Центросоюз РФ

Нижегородский областной союз потребительских обществ

Нижегородский экономико-технологический колледж

ЗАДАНИЕ НА ПРАКТИКУ

УП.05 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ

**Ананьев Артём Алексеевич**

(Фамилия, имя, отчество обучающегося)

|  |  |  |
| --- | --- | --- |
| Специальность | *09.02.07 Информационные системы и программирование* | |
|  | Код и наименование специальности | |
| *3* курс *34П* группа | |  |

Место прохождения практики:

*Лаборатория организации и принципов построения информационных систем*

(Название лаборатории, кабинета)

Сроки практики с « *17* » *марта* 2025 г. по « *21* » *марта* 2025 г.

с « *14* » *апреля*  2025 г. по « *18* » *апреля* 2025 г.

Всего часов по практике *72* час.

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Задание, раскрывающее содержание практики** | **Даты выполнения задания** | **Форма отчетности** |
| 1 | Определение требований, проектирование функционала, разработка пользовательских сценариев | 17.03.25 | Описание выполненной работы, диаграммы Use case и Activity, таблицы с текстовым описанием сценариев. |
| 2 | Проектирование интерфейса для каждой роли: окна, страницы, меню, кнопки. | 18.03.25 | Описание выполненной работы, wireframe-диаграммы |
| 3 | Проектирование БД: создание ERD-модели, словаря данных, скрипта для восстановления базы данных | 19.03.25 | Описание выполненной работы, ERD, словарь данных, листинг SQL-скрипта восстановления (схема; схема + данные) |
| 4 | Разработка подсистемы регистрации/авторизации, ПИ администратора | 20.03.25 | Описание выполненной работы, листинги кода, скриншоты |
| 5 | Реализация проверки данных пользователя, подключение библиотеки с методами валидации данных | 21.03.25 | Описание выполненной работы, листинги библиотечного метода и unit-тестов, скриншоты приложения. |
|  | Проектировании и разработка мобильного приложения | 14.04.25 – 17.04.25 | Описание выполненной работы, листинги кода, скриншоты |
| 5 | Оформление отчетной документации по практике | 18.04.25 | Задание на практику, дневник, отчет, аттестационный лист |
| 6 | Дифференцированный зачет | 18.04.25 | Защита дневника-отчета |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Руководитель практики: | *преподаватель* |  |  |  | *Кисарова М.В.* |
|  | должность |  | подпись |  | ФИО |

Арзамас, 2025

Центросоюз РФ

Нижегородский областной союз потребительских обществ

Нижегородский экономико-технологический колледж

ДНЕВНИК учебной ПРАКТИКИ

*ПМ.05 Проектирование и разработка информационных систем*

Индекс и название профессионального модуля

Специальность

*09.02.07 Информационные системы и программирование*

Код и наименование специальности

Обучающегося(йся) *3* курса *34П* группы

**Ананьев Артём Алексеевич**

Фамилия, имя, отчество обучающегося

Место практики: *Нижегородский экономико-технологический колледж*

*Лаборатория организации и принципов*

*построения информационных систем*

Название организации, лаборатории, кабинета

Сроки практики с « *17* » *марта*  2025 г. по « *21* » *марта* 2025 г.

с « *14* » *апреля*  2025 г. по « *18* » *апреля* 2025 г.

Всего часов по практике *72* час.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Руководитель практики: | *преподаватель* |  |  |  | *Кисарова М.В.* |
|  | должность |  | подпись |  | ФИО |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Зав.производственной практикой: |  |  |  | *Цапаева А.А.* |
|  |  | подпись |  | ФИО |

Арзамас, 2025

СОДЕРЖАНИЕ ДНЕВНИКА

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Описание выполненной работы | Кол-во  часов | Подпись руководителя практики |
| 1 | 2 | 3 | 4 |
| 17.03.25 | Анализ предметной области |  |  |
| 17.03.25 | Функциональное моделирование. Разработка диаграммы Use Case |  |
| 17.03.25 | Составление сценариев действий пользователей. Разработка диаграммы Activity |  |
| 18.03.25 | Проектирование пользовательского интерфейса. |  |  |
| 18.03.25 | Моделирование экранов ПИ для решения |  |
| 18.03.25 | задач каждой категории пользователей системы |  |
| 18.03.25 | Разработка диаграмм wireframe |  |
| 19.03.25 | Проектирование базы данных |  |  |
| 19.03.25 | Определение сущностей, связей, атрибутов, |  |
| 19.03.25 | типов данных, ключей. Создание ERD |  |
| 19.03.25 | Создание словаря базы данных |  |
| 20.03.25 | Разработка подсистемы регистрации/авторизации |  |  |
| 20.03.25 | Создание и подключение базы данных |  |
| 20.03.25 | Создание экранов входа в систему |  |
| 20.03.25 | Реализация базовой логики авторизации |  |
| 21.03.25 | Разработка библиотеки классов для валидации |  |  |
| 21.03.25 | данных пользователя при регистрации |  |
| 21.03.25 | Подключение библиотеки к проекту. Тестирование разработанного функционала |  |
| 14.04.25 | Создание кроссплатформенного приложения |  |  |
| 14.04.25 | с использованием платформы .NET MAUI |  |
| 14.04.25 | Добавление визуальных элементов управления |  |
| 15.04.25 | Создание пользовательского интерфейса (UI) |  |  |
| 15.04.25 | с помощью языка разметки XAML |  |
| 15.04.25 | Добавление поведения на страницу XAML |  |
| 15.04.25 | Тестирование и отладка приложения |  |
| 16.04.25 | Настройка макета на страницах |  |  |
| 16.04.25 | приложения .NET MAUI XAML |  |
| 16.04.25 | Разметка макета с помощью StackLayout |  |
| 16.04.25 | Разметка макета представления с помощью Grid |  |
| 17.04.25 | Добавление привязки данных |  |  |
| 17.04.25 | для представления и обновления данных в пользовательском интерфейсе |  |
| 17.04.25 | Разработка класса-интерфейса для конвертирования значений разных типов |  |
| 17.04.25 | Тестирование и отладка кроссплатформенного приложения |  |
| 18.04.25 | Разработка отчетной документации по УП.05 |  |  |
| 18.04.25 | «Проектирование и разработка информационных систем» |  |
| 18.04.25 | Дифференцированный зачет |  |

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_ / Ананьев А.А. /

Руководитель практики \_\_\_\_\_\_\_\_\_\_\_\_\_ / *Кисарова М.В.* /

подпись Ф.И.О.

Центросоюз РФ

Нижегородский областной союз потребительских обществ

Нижегородский экономико-технологический колледж

ОТЧЕТ О прохождении учебной ПРАКТИКИ

*ПМ.05 Проектирование и разработка информационных систем*

Индекс и название профессионального модуля

Специальность

*09.02.07 Информационные системы и программирование*

Код и наименование специальности

Обучающегося(йся) *3* курса *34П* группы

**Ананьев Артём Алексеевич**

Фамилия, имя, отчество обучающегося

Место практики: *Нижегородский экономико-технологический колледж*

*Лаборатория организации и принципов*

*построения информационных систем*

Название организации, лаборатории, кабинета

Сроки практики с « *17* » *марта* 2025 г. по « *21* » *марта* 2025 г.

с « *14* » *апреля*  2025 г. по « *18* » *апреля* 2025 г.

Всего часов по практике *72* час.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Руководитель практики: | *преподаватель* |  |  |  | *Кисарова М.В.* |
|  | должность |  | подпись |  | ФИО |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Зав.производственной практикой: |  |  |  | *Цапаева А.А.* |
|  |  | подпись |  | ФИО |

Оценка по практике

Арзамас, 2025

СОДЕРЖАНИЕ ОТЧЕТА

1. ПРОЕКТИРОВАНИЕ ИС
   1. Проектирование функционала ИС 3
   2. Проектирование пользовательского интерфейса 3
   3. Проектирование базы данных 5
2. РАЗРАБОТКА ИС
   1. Подсистема авторизации/регистрации 6
   2. Библиотека с методами валидации данных 6
3. ТЕСТИРОВАНИЕ ИС
   1. Модульное тестирование 8
   2. Тестирование пользовательского интерфейса 10
4. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ 11

ВЫВОДЫ 27

ПРИЛОЖЕНИЯ

* + - 1. ПРОЕКТИРОВАНИЕ ИС
  1. Проектирование функционала ИС

Необходимо провести исследование описания предметной области «Агентство недвижимости» и сформулировать функциональную спецификацию разрабатываемой информационной системы. Исходное описание содержится в Приложении А.

Система предназначена для трех основных категорий пользователей: администраторов, риелторов и клиентов. Каждая категория имеет свои уникальные задачи и требования к функционалу.

Администраторы отвечают за управление всей системой, включая:

добавление и удаление пользователей, утверждение сделок, контроль за доступностью объектов недвижимости.

Риелторы выполняют основную работу по взаимодействию с клиентами: добавляют новые объекты, обрабатывают заявки, формируют отчеты по сделкам.

Клиенты имеют доступ к основным функциям поиска и подачи заявок: фильтрация объектов по различным критериям, отправка заявок на просмотр объектов, отслеживание статуса заявок.

Все функциональные требования были систематизированы и представлены на Use Case-диаграмме (см. Приложения Б). Для каждой группы пользователей были разработаны Activity-диаграммы, отражающие последовательность действий при выполнении различных операций (см. Приложение В).

* 1. Проектирование пользовательского интерфейса

При проектировании пользовательского интерфейса важно учитывать особенности предметной области, чтобы создать эффективное и удобное решение для конечных пользователей. В контексте агентства недвижимости ключевыми аспектами являются взаимодействие с клиентами, управление объектами недвижимости и обеспечение прозрачности процессов. Макет пользовательского интерфейса находится в Приложении Г.

Экран авторизации содержит заголовок «Агентство недвижимости», два поля для ввода логина и пароля, а также кнопки «Войти» и «Регистрация».

Регистрационная форма включает поля для ввода логина, пароля, повторения пароля, а также кнопку «Вход».

Интерфейс страницы клиента имеет две кнопки: «Продажа объекта недвижимости» и «Покупка объекта недвижимости».

Макеты продажи и покупки объектов недвижимости похожи и содержат заголовки, левую колонку с вкладками ФИО, номер, тип недвижимости и цена, правую колонку с кнопками риэлтор, покупатель/пользователь и комиссия при нажатии которых будет выводится сообщение, а также кнопку «Заключить сделку».

Интерфейс для риэлторов включает в себя два основных раздела: первый раздел список клиентов, который можно отфильтровать в зависимости от запроса. Во втором разделе представлены списки объектов и покупателей, которые так же можно отфильтровать по типу недвижимости и цене. Ниже расположена кнопка «Завершить».

Интерфейс для директоров также состоит из двух разделов. В первом разделе директор может просматривать список сотрудников и редактировать данные. Во втором разделе директор может просматривать отчеты о сделках, которые фильтруются по дате и цене.

Интерфейс для администраторов предлагает полный контроль над системой управления агентством недвижимости. Первый раздел посвящен управлению пользователями, где администратор может просматривать списки зарегистрированных сотрудников и клиентов, редактировать их профили, добавлять новых пользователей или удалять ненужные аккаунты. Во втором разделе представлена подробная информация обо всех объектах, находящихся в базе данных агентства. Администратор может обновлять данные об объектах. Функционал удаления и добавления объектов позволяет поддерживать базу данных актуальной и соответствующей реальным предложениям на рынке. Внизу располагается кнопка «Назад».

* 1. Проектирование базы данных

Между таблицами существуют четкие связи, позволяющие сохранять консистентность данных и предотвращать потерю информации. Основные связи включают свойства недвижимости связаны с риелторами, которые добавили эти объекты в систему, заявки клиентов на просмотр объектов связаны с самими объектами и клиентами, подавшими заявку, сделки (транзакции) привязаны к объектам недвижимости и покупателям.

Это достигается за счет использования внешних ключей, что позволяет чтобы поддерживать целостность данных и избегать дублирования информации структура базы данных построена в соответствии с принципами нормализации данных. Это означает что минимизирована избыточность данных исключены аномалии обновления вставки и удаления упрощён процесс внесения изменений и добавления новой информации. Благодаря этому база данных становится более устойчивой к ошибкам и легче масштабируема.

Целостность данных обеспечивается несколькими механизмами. Внешние ключи REFERENCES гарантируют что записи в одной таблице ссылаются только на существующие записи в другой таблице. Ограничения на типы данных ENUM TINYINT DECIMAL защищают от ввода недопустимых значений. Поля с датами DATE автоматически контролируют соответствие формату даты.

Предложенная архитектура базы данных предоставляет прочную основу для дальнейшей эволюции системы. Она легко адаптируется под новые требования такие как интеграция с внешними системами аналитики подключение платежных шлюзов или внедрение геопространственных сервисов для более точного определения.

Мы также создали словарь данных (Приложение Е), который описывает структуру базы данных, включая типы данных, размеры полей и ограничения, такие как уникальность первичного ключа и наличие обязательных полей.

* + - 1. РАЗРАБОТКА ИС
  1. Подсистема авторизации/регистрации

Процесс авторизации и регистрации пользователей был разработан с учетом высоких стандартов безопасности и удобства. Пользователям предлагаются два сценария для входа в систему: через существующую учетную запись или путем создания новой. При регистрации новый пользователь заполняет специальную форму, указывая обязательные данные: имя, электронную почту и пароль.

Пароли защищены механизмом хэширования, что предотвращает возможность несанкционированного доступа к аккаунтам. После успешного завершения регистрации пользователю отправляется письмо-подтверждение на указанный e-mail, которое необходимо активировать перед первым входом в систему. Для дополнительной защиты данных и предотвращения мошенничества внедрена двухфакторная аутентификация.

При входе в систему после ввода имени пользователя и пароля система запрашивает дополнительный одноразовый код, который генерируется специальным приложением или отправляется на телефон пользователя.

В итоге была создана эффективная и безопасная подсистема авторизации и регистрации, которая соответствует современным стандартам разработки и обеспечивает надежную защиту данных пользователей (Приложение Ж).

* 1. Библиотека с методами валидации данных

Мы столкнулись с задачей разработки информационной системы для агентства недвижимости, и одной из ключевых задач стало создание надёжного механизма для проверки паролей. Ведь именно от качества паролей зависит безопасность пользовательских данных.

Чтобы решить эту проблему, мы разработали специальную библиотеку методов для валидации данных. Центральным звеном этой библиотеки стал класс PasswordValidator, который отвечает за проверку соответствия паролей ряду важных критериев безопасности (Листинг 1).

Листинг 1 – Метод валидации данных

public class PasswordValidator

{

private readonly int minLength = 8; // Минимальная длина пароля

public (bool IsValid, string Message) Validate(string password)

{

if (string.IsNullOrEmpty(password))

{

return (false, "Пароль не может быть пустым.");

}

if (password.Length < minLength)

{

return (false, $"Пароль должен содержать минимум {minLength} символов.");

}

bool hasUpperCase = password.Any(char.IsUpper);

if (!hasUpperCase)

{

return (false, "Пароль должен содержать хотя бы одну заглавную букву.");

}

bool hasDigit = password.Any(char.IsDigit);

if (!hasDigit)

{

return (false, "Пароль должен содержать хотя бы одну цифру.");

}

return (true, "Пароль валиден.");

}

}

Основной принцип работы библиотеки заключается в последовательной проверке каждого критерия, начиная с минимальной длины пароля и заканчивая наличием специальных символов. Мы установили минимальную длину пароля в восемь символов, чтобы повысить его стойкость к взлому. Помимо этого, мы обязали пользователей использовать хотя бы одну заглавную букву и одну цифру, что делает пароли сложнее для подбора.

Эта библиотека стала неотъемлемой частью нашей системы авторизации и регистрации. Благодаря ей мы смогли значительно повысить уровень безопасности системы. Теперь наши пользователи могут чувствовать себя увереннее, зная, что их данные защищены более сложными паролями.

3. ТЕСТИРОВАНИЕ ИС

* 1. Модульное тестирование

Модульное тестирование являлось ключевым этапом разработки информационной системы. Оно направлено на проверку отдельных компонентов программы (модулей) на соответствие ожидаемому поведению. Для проведения модульного тестирования использовались специализированные инструменты, такие как MSTest, которые позволяли имитировать поведение зависимостей и проверять функциональность изолированно.

На первом этапе мы определили, какие части системы требуют тестирования. Основное внимание уделили критически важным компонентам, таким как валидация паролей, работа с базой данных и механизмы авторизации.

Подготовили набор тестовых данных, которые покрывают различные случаи использования.

На следующем этапе разработали тест-кейсы, которые включают как положительные, так и негативные сценарии. Тест-кейс для положительного тестирования можно увидеть в таблице 1.

Таблица 1 – Тест-кейс для положительного тестирования

|  |  |  |
| --- | --- | --- |
| Что проверяем | Тестовые данные | Ожидаемый результат |
| Минимальная длина | "Aa123456" | Валидный пароль |
| Наличие заглавной буквы | "aA123456" | Валидный пароль |
| Наличие цифры | "Abcd12345" | Валидный пароль |
| Все условия соблюдены | "Test1234Q" | Валидный пароль |

Положительные тесты проверяют правильность работы системы при нормальных условиях. Тест-кейс для негативного тестирования можно увидеть в таблице 2.

Таблица 2 - Тест-кейс для негативного тестирования

|  |  |  |
| --- | --- | --- |
| Что проверяем | Тестовые данные | Ожидаемый результат |
| Недостаточно длинная строка | "Aa12" | Сообщение: «Пароль должен содержать минимум 8 символов» |
| Нет заглавной буквы | "abcd123456" | Сообщение: «Пароль должен содержать хотя бы одну заглавную букву» |
| Нет цифры | "Abcdefgh" | Сообщение: «Пароль должен содержать хотя бы одну цифру» |
| Пустая строка | "" | Сообщение: «Пароль не может быть пустым» |

Негативные — выявляют ошибки при неправильных данных или непредвиденных ситуациях.

Проведённое юнит-тестирование позволило убедиться в правильной работе модуля валидации паролей. Было создано несколько тестов, каждый из которых проверял различные аспекты валидации (Листинг 2).

Листинг 2 – Пример положительного тестирования

[TestMethod]

public void PasswordValidator\_all\_valid()

{

var validator = new PasswordValidator();

string password = "Test1234Q";

bool expectedIsValid = true;

string expectedMessage = "Пароль валиден.";

var result = validator.Validate(password);

Assert.AreEqual(expectedIsValid, result.IsValid);

Assert.AreEqual(expectedMessage, result.Message);

}

Первый тест проверяет ситуацию, когда пароль соответствует всем установленным правилам. Второй тест проверяет ситуацию, когда пароль не соответствует одному из установленных правил (Листинг 3).

Листинг 3 -Пример негативного тестирования

[TestMethod]

public void PasswordValidator\_NoAZ\_valid()

{

var validator = new PasswordValidator();

string password = "abcd123456";

bool expectedIsValid = false;

string expectedMessage = "Пароль должен содержать хотя бы одну заглавную букву.";

var result = validator.Validate(password);

Assert.AreEqual(expectedIsValid, result.IsValid);

Assert.AreEqual(expectedMessage, result.Message);

}

В этом тесте пароль «abcd123456» содержит 10 символов и цифры, но не имеет заглавных букв. Соответственно, валидатор должен вернуть результат IsValid = false и сообщение «Пароль должен содержать хотя бы одну заглавную букву.». Тест подтверждает, что при отсутствии заглавных букв пароль отвергается.

Модульное тестирование оказалось крайне полезным инструментом в нашем проекте. Оно помогло выявить потенциальные проблемы заранее и обеспечило уверенность в работоспособности системы. Такой подход позволил снизить риски возникновения ошибок в рабочем приложении и ускорить процесс разработки благодаря автоматизации тестирования.

* 1. Тестирование пользовательского интерфейса

Тестирование пользовательского интерфейса проводилось с целью оценки удобства использования системы, выявления возможных ошибок в представлении данных и проверки совместимости с различными браузерами и устройствами. Навигация и доступность. Оценивалась простота перемещения по системе, а также доступность всех важных функций для пользователей с разными ролями.

Оформление и отображение данных, проводилась проверка корректности отображения текста, изображений и прочих элементов интерфейса на различных экранах и разрешениях. Работа с формами, тестировалась корректность работы всех интерактивных элементов, таких как кнопки, выпадающие списки, текстовые поля и календари. Особое внимание уделялось проверке обязательных полей и валидации введенных данных

4. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ

.NET MAUI — это технология от Microsoft, позволяющая программистам писать мобильные и десктоп-приложения сразу для нескольких платформ с использованием одного набора инструментов и единого кода на C# и XAML. Это значительно ускоряет разработку и снижает затраты на поддержку множества версий продукта, сохраняя при этом уникальный внешний вид и производительность каждого устройства.

Первое упражнение заключалось в создании простого приложения на платформе .NET MAUI, предназначенного для изучения основ разработки кросс-платформенных приложений, совместимых с различными устройствами и системами, такими как Windows и Android.

Процесс начался с установки необходимых инструментов — среды Visual Studio с поддержкой .NET MAUI. Затем был создан новый проект, основанный на шаблоне .NET MAUI, который получил название «Phoneword».

Дальше последовал важный этап тестирования: приложение собрано и запущено на Windows. Результат получился предсказуемым — простая форма с кнопкой «Click me!», которая отслеживала количество нажатий.

Но амбициозному разработчику недостаточно простых результатов, поэтому решили усложнить задачу: перенастроили обработку нажатий таким образом, чтобы каждое касание приводило к увеличению числа на 5. Это достигнуто изменением метода OnCounterClicked() в файле MainPage.xaml.cs (Листинг 4).

Листинг 4 – Метод для прибавления 5 очков

private void OnCounterClicked(object sender, EventArgs e)

{

count += 5; // Увеличим счетчик на 5

if (count == 1)

CounterBtn.Text = $"Clicked {count} time";

else

CounterBtn.Text = $"Clicked {count} times";

SemanticScreenReader.Announce(CounterBtn.Text); // Читаем вслух текущее состояние

}

Важнейшим аспектом .NET MAUI является возможность одновременной работы на разных платформах. Поэтому следующим испытанием стало запуск приложения на эмуляторе Android.

Прошло некоторое время ожидания, пока загрузится виртуальное устройство, и вот оно — наше приложение вновь предстало перед нами, на этот раз на Android-экранчике. Готовое приложение можно увидеть на рисунке 1.

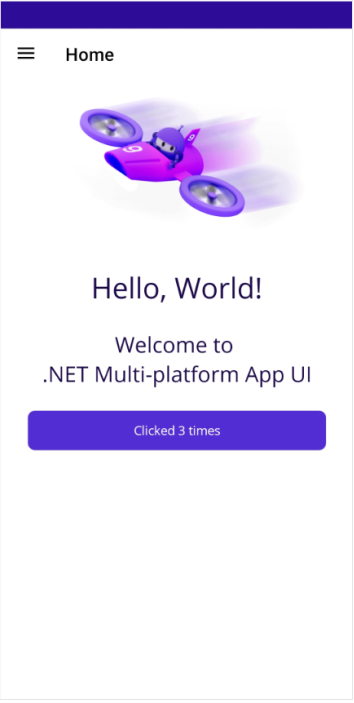


Рисунок 1 – Готовое приложение из первого упражнения

Второе упражнение заключалось в доработке существующего приложения Phoneword, созданного в предыдущем занятии. Цель задания — реализовать полноценную логику перевода буквенного текста в телефонный номер и обеспечить возможность совершить реальный звонок с использованием соответствующих API.

Прежде всего, нужно подготовить существующую заготовку проекта Phoneword, созданную в предыдущем уроке. Основное внимание уделяется улучшению интерфейса и функциональности приложения.

В файле MainPage.xaml был удалён лишний элемент управления <ScrollView> и вся находившаяся там разметка. Осталась только пустая страница <ContentPage>.

Вместо старого содержания добавляем контейнер <VerticalStackLayout>, который размещает элементы вертикально друг под другом с отступами.

Внутри контейнера размещаем четыре основных компонента:

<Label> — служит для надписи «Enter a Phoneword».

<Entry> — предназначен для ввода текста пользователем.

Две кнопки: одна для перевода введённого текста в телефонный номер, другая для осуществления звонка.

Свойство x:Name используется для присвоения уникальных идентификаторов различным элементам управления. Эти идентификаторы нужны для обращения к ним в программном коде.

Следующая задача — организовать перевод введённого текста в подходящий телефонный номер. Для этого добавляется новый класс PhonewordTranslator.cs, содержащий основную логику преобразования:

Реализован статический метод ToNumber, принимающий строку и возвращающий её цифровой эквивалент. Алгоритм основан на сопоставлении букв алфавита с соответствующими цифрами на классической телефонной клавиатуре.

Интерфейс теперь организован, и нужно связать кнопки с соответствующей логикой:

Метод OnTranslate: обрабатывает нажатие кнопки «Перевести». В нём выполняются следующие действия:

* + Чтение введённого текста из элемента <Entry>.
  + Обращение к классу PhonewordTranslator для преобразования текста.
  + Проверка полученного результата: если номер успешно переведён, кнопка «Позвонить» активируется и выводит сформированный номер телефона.

Метод OnCall: работает по нажатию кнопки «Позвонить».

Чтобы приложение могло вызывать телефонные номера, подключаются специальные методы и события:

* Использование PhoneDialer: телефонный номер отправляется с помощью объекта PhoneDialer, который поддерживает вызов на большинстве современных платформ (Windows, Android, iOS).
* Механизм уведомлений: до попытки набора номера предлагается предупреждение с запросом согласия пользователя.

Поскольку некоторые случаи могут привести к проблемам, предусмотрены блоки try-catch для обработки потенциальных ошибок.

После завершения работы приложение собирается и тестируется на различных платформах. На рисунке 2 представлено итоговое приложение.

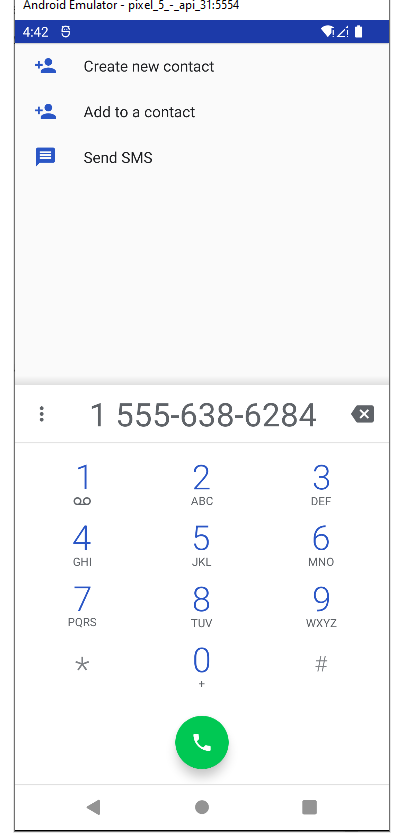


Рисунок 2 – Готовое приложение Phoneword

После этого мы начали создавать первую страницу XAML. Начали работу с клоном официального репозитория с сайта GitHub, скачав его локально на компьютер. После этого перешли в директорию exercise1, открыли там файл решения в среде разработки Visual Studio.

Анализ исходного кода открыв файлы MainPage.xaml и MainPage.xaml.cs, мы увидели структуру приложения, которое создавало пользовательский интерфейс путем программного добавления контролов. Вся логика формирования интерфейса была сосредоточена именно в файле .cs.

Чтобы перенести визуальную часть приложения на XAML, мы сначала добавили на главную страницу контент вертикального стэка, используя соответствующий тег XML. Затем последовательно заполнили этот контейнер элементами интерфейса согласно структуре исходного кода: Label, Editor и Grid с двумя кнопками («Сохранить» и «Удалить»).

Создали разделы для заголовка приложения, поля ввода текста и панели с кнопками. Задали необходимые атрибуты каждому элементу. Связали обработчики событий с соответствующими методами в коде (OnSaveButtonClicked и OnDeleteButtonClicked).

Перейдя обратно в файл MainPage.xaml.cs, убрали старый код создания интерфейса и оставили лишь метод инициализации, а также чтение сохраненных ранее заметок из файла. Запустили приложение на платформе Windows и убедились, что интерфейс отображается правильно и работает аналогично предыдущему варианту. Протестировали приложение на эмуляторе Android, чтобы убедиться в правильной работе на мобильной платформе.

Все изменения были успешно интегрированы, интерфейс приложения теперь определяется исключительно через XAML-разметку, что делает его проще для поддержки и модификации в будущем. Готовый результат можно увидеть на рисунке 3.

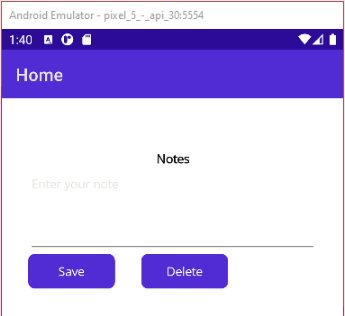


Рисунок 3 – Первая страница XAML

Целью данного этапа было добавить возможность централизованного управления цветами шрифтов и фонов в приложении Notes посредством XAML. Для этого потребовалось создать отдельный статический класс с публичными свойствами, представляющими нужные цвета, а затем использовать их в разметке страниц.

Сначала мы создали новый класс под названием SharedResources.cs. Этот класс предназначен для хранения глобальных переменных, используемых в приложении. Внутри класса объявили два статических поля типа Color: одно для цвета шрифта, другое для цвета фона.

Далее мы открыли основной файл разметки и настроили пространство имен таким образом, чтобы класс SharedResources стал доступен в XAML-коде. Теперь стало возможным указывать цвет шрифта и фона для каждого элемента управления, применяя специальное расширение разметки x:Static, которое позволяло получать значения цветов из статического класса.

Цвет шрифта был установлен для всех основных элементов. Благодаря этому любые изменения цветовой схемы будут сразу отражаться во всём приложении, достаточно обновить соответствующие значения в классе SharedResources.

Для повышения удобства использования приложения на устройствах с разными размерами экранов, мы установили различную высоту для элемента редактирования текста на мобильных платформах, оставив стандартную высоту для настольных систем. Это достигается применением конструкции OnPlatform, позволяющей устанавливать размеры, специфичные для каждой платформы.

Завершив настройку, мы собрали и запустили приложение на Windows, проверив правильность отображения цветов и размеров. Дополнительно провели тестирование на виртуальном устройстве Android, убедившись, что различия в размерах работают корректно (Рисунок 4).

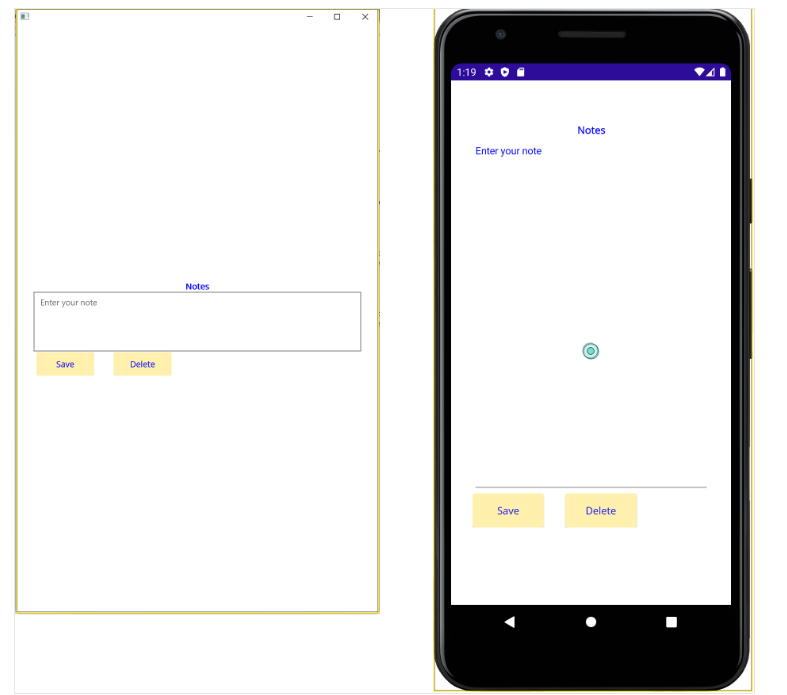


Рисунок 4 – Добавление стиля для страницы XAML

Мы скачали материалы урока с сайта GitHub и распаковали архив в удобную директорию на своём ПК. Затем запустили среду разработки Visual Studio и открыли исходники проекта из предоставленной папки.

Перед запуском приложения мы внимательно изучили интерфейс формы, заметил наличие элементов управления, отвечающих за изменение ориентации и расположения содержимого на экране.

Пошагово нажимали каждую кнопку, соответствующую различным параметрам выравнивания.

Во время экспериментов наблюдали за изменениями положения и размеров главного элемента интерфейса.

После завершения серии тестов мы записали наблюдения относительно влияния параметров LayoutOptions на общий дизайн приложения. Оценивали привлекательность и практичность каждого варианта в зависимости от предполагаемых сценариев использования (Рисунок 5).

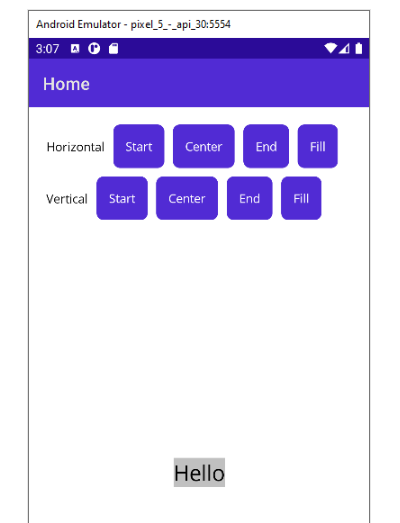


Рисунок 5 – Обзор параметров выравнивания

Основная цель состояла в том, чтобы улучшить внешний вид и юзабилити приложения калькулятора чаевых, сделав его более удобным и эстетичным. Изначально приложение имело простейший интерфейс, состоящий из одного вертикального стека, где все элементы располагались друг под другом.

Чтобы улучшить ситуацию, мы решили разделить элементы интерфейса на группы, объединив их с помощью горизонтальных стеков. Такое разделение позволило более аккуратно расположить надписи и поля ввода, выровнять кнопки и улучшить общее впечатление от приложения (Рисунок 6).

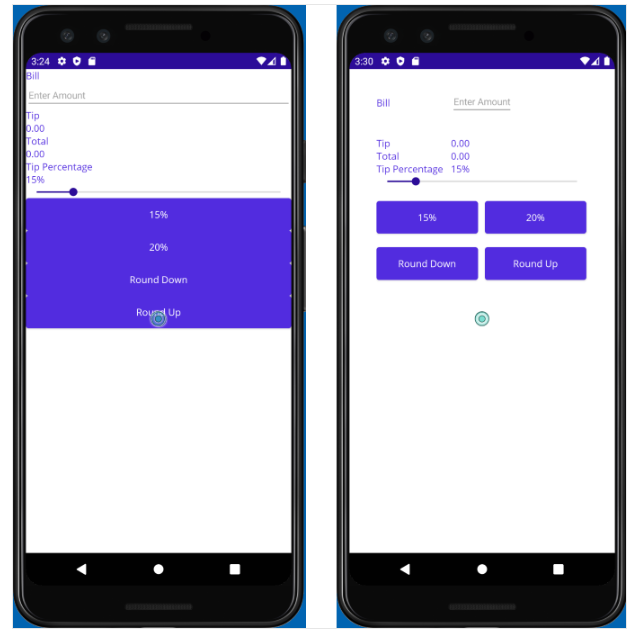


Рисунок 6 – StackLayout для интерфейса

В данном упражнении использовалась сетка для упорядочивания элементов пользовательского интерфейса. Работа началась с новой версии проекта TipCalculator, которую следовало настроить для придания ей большей интуитивности и удобства.

Первоначально пользовательский интерфейс представлял собой простую последовательность элементов, расположенных вертикально. Для улучшения компоновки и организации содержимого я использовал сеточный макет.

Первым шагом было преобразование макета страницы, построив её структуру на основе строк и колонок сетки. Каждый элемент был размещён в своей ячейке сетки, обеспечивая правильное выравнивание и размещение на экране. Особенно важным изменением стало перемещение кнопок управления вниз страницы, что улучшило общую композицию и облегчило взаимодействие с ними.

В результате выполненных шагов пользовательский интерфейс стал более логичным и удобочитаемым. Компоненты оказались грамотно распределены по пространству страницы, а сам интерфейс приобрёл современный и привлекательный внешний вид. Таким образом, переход на сеточную систему оказался эффективным решением, позволившим заметно улучшить качество и функциональность приложения. Код для Grid представлен на Листинге 5.

Листинг 5 – Код Grid

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:TipCalculator"

x:Class="TipCalculator.MainPage">

<Grid RowDefinitions="Auto, Auto, Auto, \*, Auto, Auto, Auto"

ColumnDefinitions="\*, \*"

Padding="40">

<Label Text="Bill" VerticalOptions="Center" Grid.Row="0" Grid.Column="0"/>

<Entry x:Name="billInput" Placeholder="Enter Amount" Keyboard="Numeric" Grid.Row="0" Grid.Column="1"/>

<Label Text="Tip" Grid.Row="1" Grid.Column="0"/>

<Label x:Name="tipOutput" Text="0.00" Grid.Row="1" Grid.Column="1"/>

<Label Text="Total" Grid.Row="2" Grid.Column="0"/>

<Label x:Name="totalOutput" Text="0.00" Grid.Row="2" Grid.Column="1"/>

<Label Text="Tip Percentage" VerticalOptions="End" HorizontalOptions="Start" Grid.Row="3" Grid.Column="0"/>

<Label x:Name="tipPercent" Text="15%" VerticalOptions="End" HorizontalOptions="End" Grid.Row="3" Grid.Column="1"/>

<Slider x:Name="tipPercentSlider" Minimum="0" Maximum="100" Value="15" Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="2"/>

<Button Text="15%" Clicked="OnNormalTip" Margin="5" Grid.Row="5" Grid.Column="0"/>

<Button Text="20%" Clicked="OnGenerousTip" Margin="5" Grid.Row="5" Grid.Column="1"/>

<Button x:Name="roundDown" Margin="5" Text="Round Down" Grid.Row="6" Grid.Column="0"/>

<Button x:Name="roundUp" Margin="5" Text="Round Up" Grid.Row="6" Grid.Column="1"/>

</Grid>

</ContentPage>

Решение WeatherClient.sln было открыто в Visual Studio, и была предпринята следующая модификация кода.

Ранее обработчик события нажатия кнопки «Обновить» получал данные о погоде и напрямую обновлял элементы управления. Такая архитектура усложняла масштабируемость и затрудняла сопровождение кода.

Изменённый подход заключался в получении погодных данных и назначении их в качестве контекста привязки для всей страницы. Такой подход значительно улучшает отделённость данных от пользовательского интерфейса, позволяя более эффективно управлять состоянием приложения и обеспечивать большую гибкость при дальнейшей доработке. Результат можно увидеть на рисунке 7.

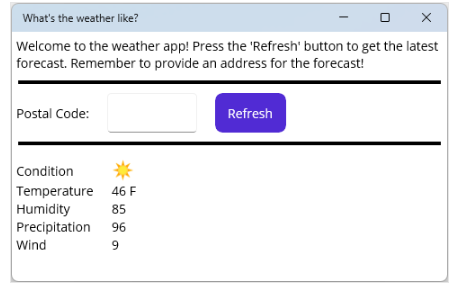


Рисунок 7 – Готовое приложение WeatherClient

Мы кратко остановимся на примере кадрового приложения и переключимся на демонстрационное приложение, посвящённое фильмам. В нём уже предусмотрены несколько ключевых страниц и модели представлений с основными связями данных. Есть две отдельные страницы: одна показывает список фильмов, а вторая отображает подробную информацию о конкретном фильме.

Наша задача заключается в оптимизации процесса выбора фильма. Ранее при выборе фильма на странице со списком использовался иной подход к передаче данных на страницу подробностей. Сейчас мы хотим, чтобы обе страницы использовали одно и то же свойство в модели представления, тем самым исключая дублирование и повышая эффективность передачи данных между страницами (Рисунок 8).Начало формы

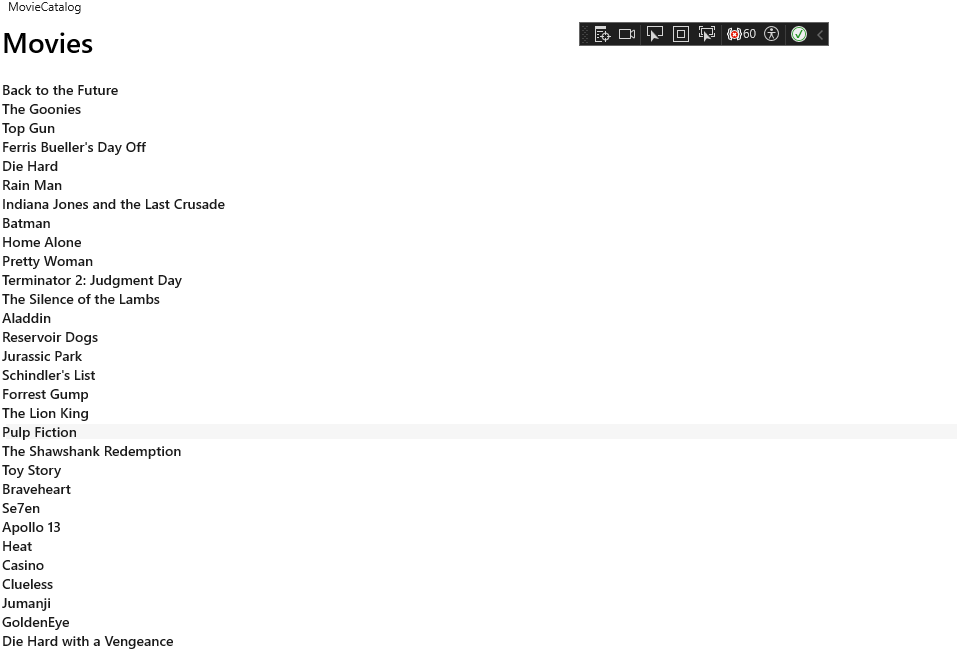


Рисунок 8 – Готовое приложение MovieCatalog

Цель данного упражнения — реализовать доступность ресурсов на множестве страниц в приложении .NET MAUI, перенеся их в словарь ресурсов класса приложения «Калькулятор чаевых». Это упражнение продолжает предыдущие этапы и направлено на дальнейшее совершенствование архитектуры приложения.

За основу взят предыдущий проект, разработанный в рамках прошлых заданий, либо можно начать с готового решения, расположенного в папке exercise4/TipCalculator репозитория, клонированного ранее. Основная задача — переместить существующие ресурсы в словарь ресурсов уровня приложения, чтобы они могли использоваться всеми страницами приложения, избегая избыточности и повышая уровень повторного использования компонентов. Готовое приложение можно увидеть на рисунке 9.

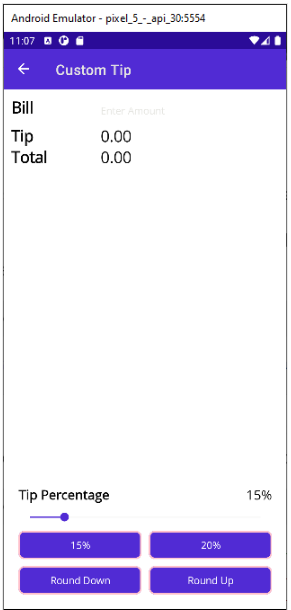
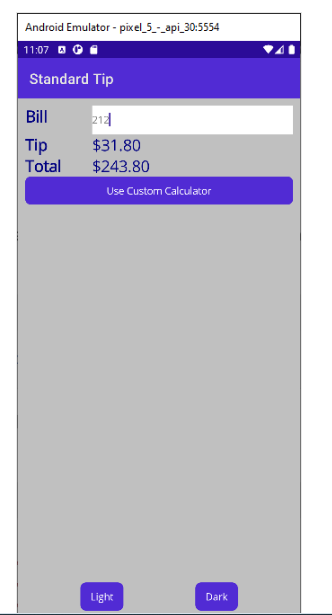


Рисунок 9 – Готовое приложение TipCalculator

Представьте себе сценарий, в котором ваше приложение .NET MAUI включает несколько самостоятельных страниц, таких как информация об астрономических объектах, фазы Луны и расписание восхода и захода солнца. Кроме того, имеется отдельная страница «О программе». Каждая из этих страниц функционирует изолированно, однако вам хочется предложить пользователям удобное средство для плавного перехода между этими разделами.

В рамках данного упражнения наша задача — внедрить навигационную панель с выпадающим меню, что позволит пользователю быстро и естественно перемещаться между указанными страницами, создавая целостный опыт взаимодействия с приложением. Результат представлен на рисунке 10.

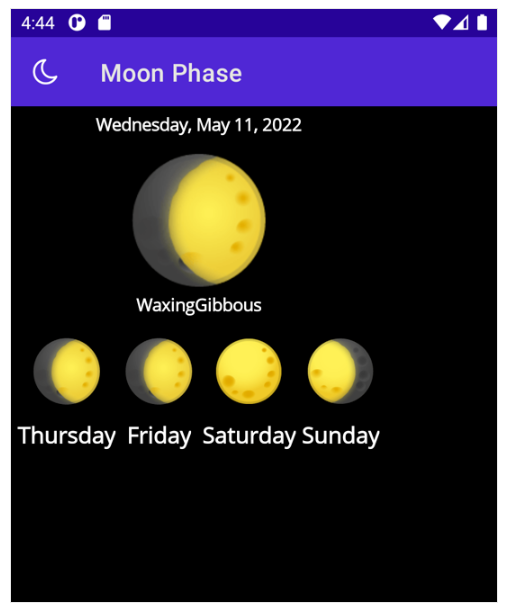
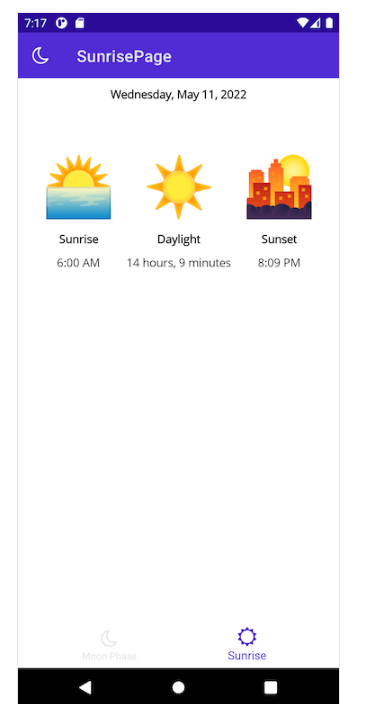
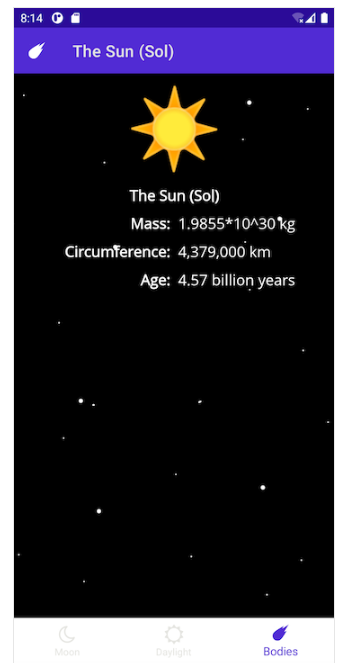


Рисунок 10 – Готовое приложение Astronomy

Во время выполнения следующего задания мы столкнулись с задачей разработки полезного инструмента для инженеров, занятых выездами на объекты заказчиков. Одной из ключевых функций приложения стала система поиска данных о различных электротехнических компонентах. Эти сведения решено было размещать в отдельной базе данных, доступ к которой осуществляется через специальную службу REST API, размещённую в облаке Microsoft Azure.

Помимо самой поисковой системы, важной частью приложения стала административная составляющая. Она даёт право авторизованным специалистам оперативно пополнять базу новыми изделиями, своевременно устранять устарелые или ошибочные записи, а также редактировать уже существующие позиции. Управление базой предусмотрено через ту же самую службу REST API.

В процессе выполнения задания мы пошагово освоилм технологию развёртывания собственной службы REST API в облаке Azure, проверку её работоспособности браузером, а затем интеграцию реализованных сервисов в готовую мобильную платформу. Мобильное приложение получило возможности поиска запчастей, добавления новых позиций, удаления старых записей и коррекции существующих данных (Рисунок 11).

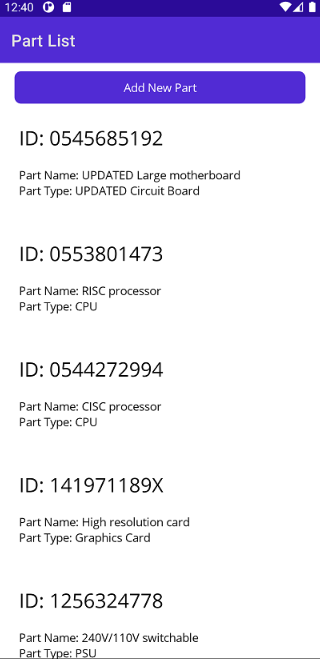


Рисунок 11 – REST API

В этом задании я применял библиотеку SQLite для локального сохранения данных в моём приложении. Представьте ситуацию, когда необходимо ускорить отклик своего социального приложения за счёт кеширования информации о пользователях. Именно для этого я и решил создать локальную базу данных SQLite, в которой буду сохранять все требуемые сведения.

Физически файл базы данных располагается в специальном каталоге локального хранилища устройства. Сам процесс создания и последующего использования такой базы данных проходил поэтапно: начиная с установки необходимой библиотеки, продолжая созданием таблицы для хранения нужных данных и заканчивая реализацией запросов для добавления, извлечения и изменения информации.

Таким образом, благодаря интеграции SQLite в своё приложение, мы добились необходимого ускорения реакции интерфейса, поскольку многие запросы теперь выполняются локально, без необходимости постоянного обращения к серверу (Рисунок 12).

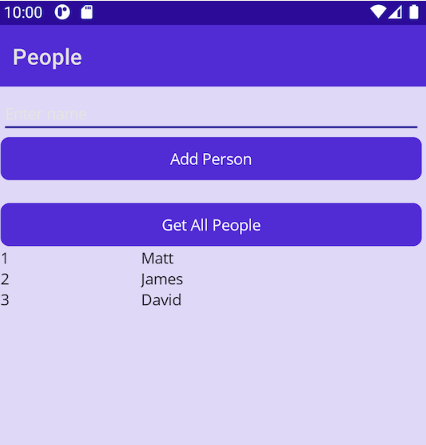


Рисунок 12 – Хранение данных с помощью SQLite

Результатом освоения курса становится умение создавать эффективные кроссплатформенные приложения с качественным дизайном и высокой производительностью.

ВЫВОДЫ

Выполнение данной работы стало значимым этапом профессионального роста и приобретения важных компетенций в сфере информационных технологий. Были освоены ключевые аспекты жизненного цикла разработки программного обеспечения, начиная от анализа требований и заканчивая развертыванием готового решения.

В ходе исследования получены глубокие знания в следующих областях:

* Методология проектирования: изучены принципы построения архитектуры информационной системы, рассмотрены подходы к проектированию функциональности, пользовательского интерфейса и структурирования базы данных.
* Разработка прикладного ПО: реализованы модули авторизации и регистрации пользователей, разработана библиотека методов валидации данных, обеспечивающая целостность и достоверность вводимой информации.
* Процесс тестирования: применялись методики модульного тестирования и тестирования пользовательского интерфейса, что помогло повысить качество разработанного программного продукта путем выявления и устранения ошибок на ранних этапах разработки.
* Создание мобильного приложения: получили практический опыт адаптации функциональных возможностей веб-приложений для различных платформ, что существенно расширило круг потенциальных пользователей и повысило удобство взаимодействия с системой.

Эти компетенции позволят успешно решать задачи современной индустрии программирования, уверенно ориентироваться в новых технологиях и инструментах разработки, эффективно организовывать процессы тестирования и сопровождения проектов любой сложности. Практика подтвердила готовность применять полученные знания и навыки в реальных условиях производственной среды, заложив основу для дальнейшего профессионального развития и успешной карьеры в сфере информационных технологий.

ПРИЛОЖЕНИЕ А

Описание предметной области «Агентство недвижимости»

Данный документ описывает предметную область.

**Основные понятия**

**Пользователь** - пользователь системы. В рамках данного задания вам предстоит реализовать систему, упрощающую работу риэлтора.

**Риэлтор** - сотрудник компании агентства недвижимости. Выступает в качестве посредника между клиентом и компанией.

**Клиент** - человек, который желает купить либо продать объект недвижимости.

**Объект недвижимости** - недвижимое имущество, предмет сделки

между клиентами. Обобщающее понятие для разных типов недвижимого имущества.

**Тип объекта недвижимости** - квартира, дом, либо земельный участок.

**Потребность** - желание клиента купить объект недвижимости, соответствующего указанным параметрам.

**Предложение** - желание клиента продать указанный объект недвижимости за указанную цену.

**Сделка** - факт осуществления продажи недвижимого имущества. В сделке участвуют две стороны: клиент-покупатель с потребностью и клиент-продавец с предложением.

**Стоимость услуг** или **комиссия** - количество денег, которое должен заплатить клиент за оказанные ему компанией услуги.

**Основные бизнес процессы компании**

Упрощая, приведем описание основного бизнес-процесса агентства недвижимости и участие в нем разрабатываемой системы:

**Продажа объекта недвижимости**

1. Клиент обращается в компанию с желанием продать объект недвижимости. В системе создаются три сущности: “Клиент”, “Объект недвижимости”, “Предложение”
2. “Предложению” клиента назначается ответственный риэлтор. Он осуществляет поиск подходящих потребностей

Окончание прил. А

1. Клиент выбирает потребность и заключается сделка.
2. Система рассчитывает размер комиссии, происходит оплата комиссии и перечисление средств риэлтору клиента-продавца и клиента-покупателя.

**Покупка объекта недвижимости**

1. Клиент обращается в компанию с желанием купить объект недвижимости. В системе создаются две сущности: “Клиент”, “Потребность”
2. “Потребности” клиента назначается ответственный риэлтор. Он осуществляет поиск подходящих предложений
3. Клиент выбирает предложение и заключается сделка.
4. Система рассчитывает размер комиссии, происходит оплата комиссии и перечисление средств риэлтору клиента-продавца и клиента-покупателя.

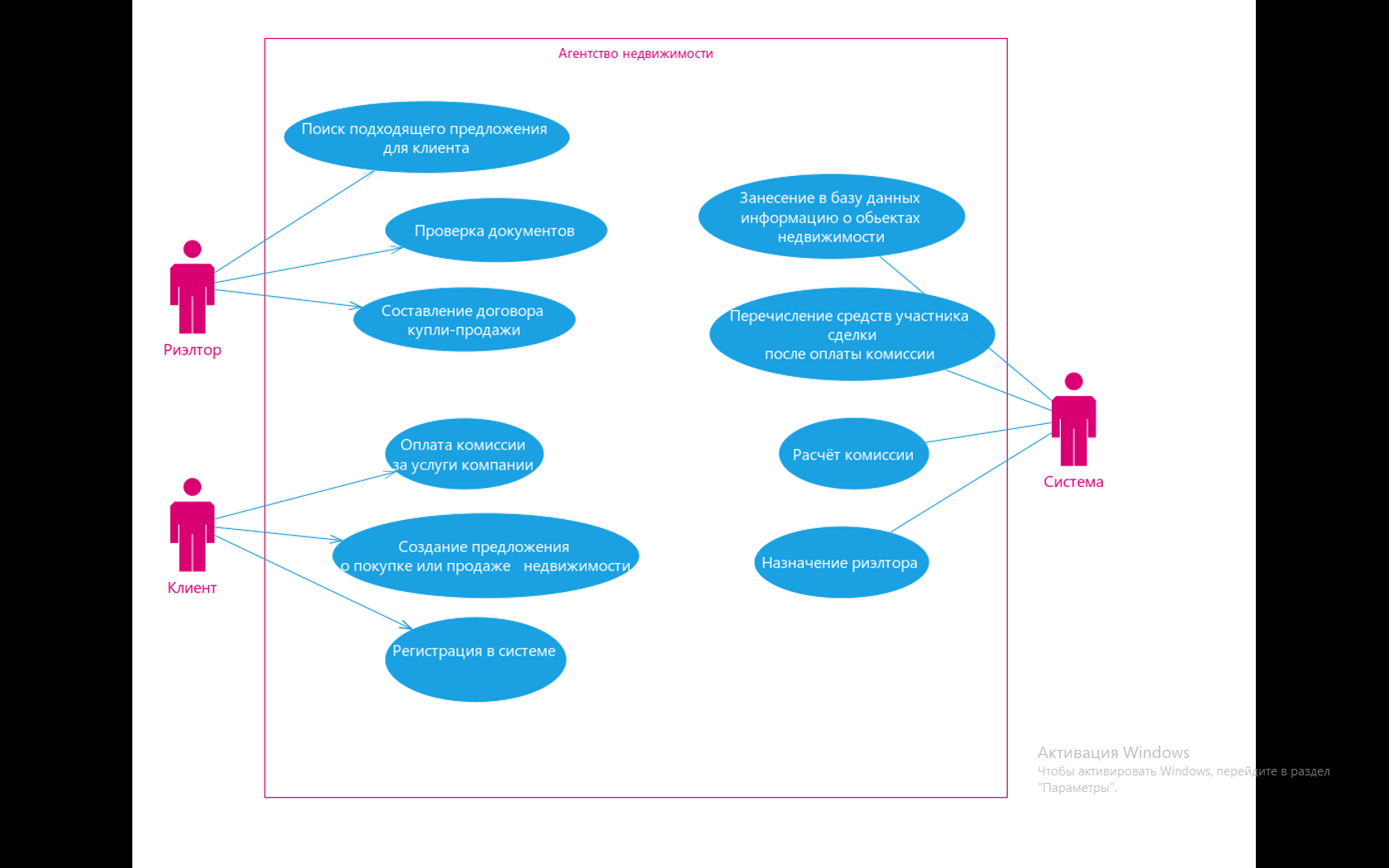
Определим отношение “удовлетворительности” между потребностью и предложением:

Предложение S удовлетворяет потребность D, если выполнены все условия:

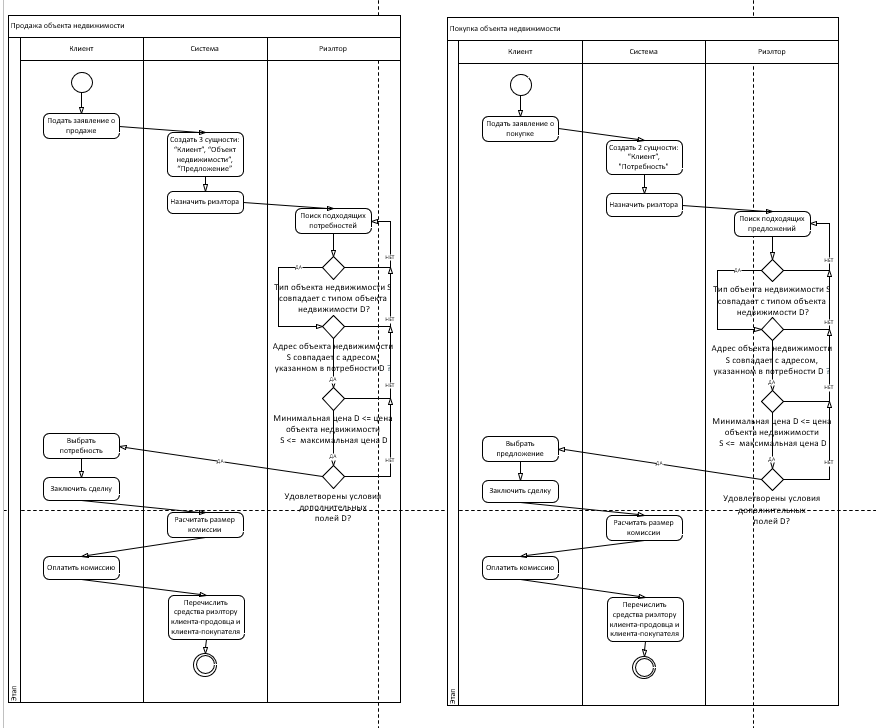
1. Тип объекта недвижимости S совпадает с типом объекта недвижимости D
2. Адрес объекта недвижимости S совпадает с адресом, указанном в потребности D (за исключением незаполненных полей в D).
3. Минимальная цена D <= цена объекта недвижимости S <=  максимальная цена D
4. Удовлетворены условия дополнительных полей D. Например для квартиры: минимальный этаж D <= этаж объекта недвижимости S <= максимальный этаж D

Данное отношение является симметричным: если предложение S удовлетворяет потребность D, то говорят, что потребность D удовлетворяется предложением S.

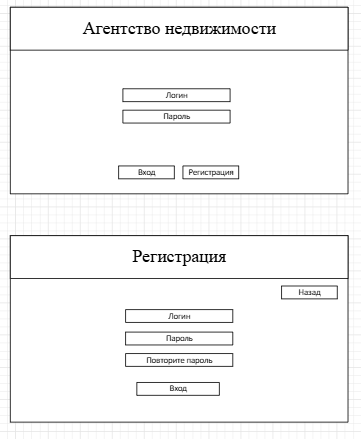
ПРИЛОЖЕНИЕ Б



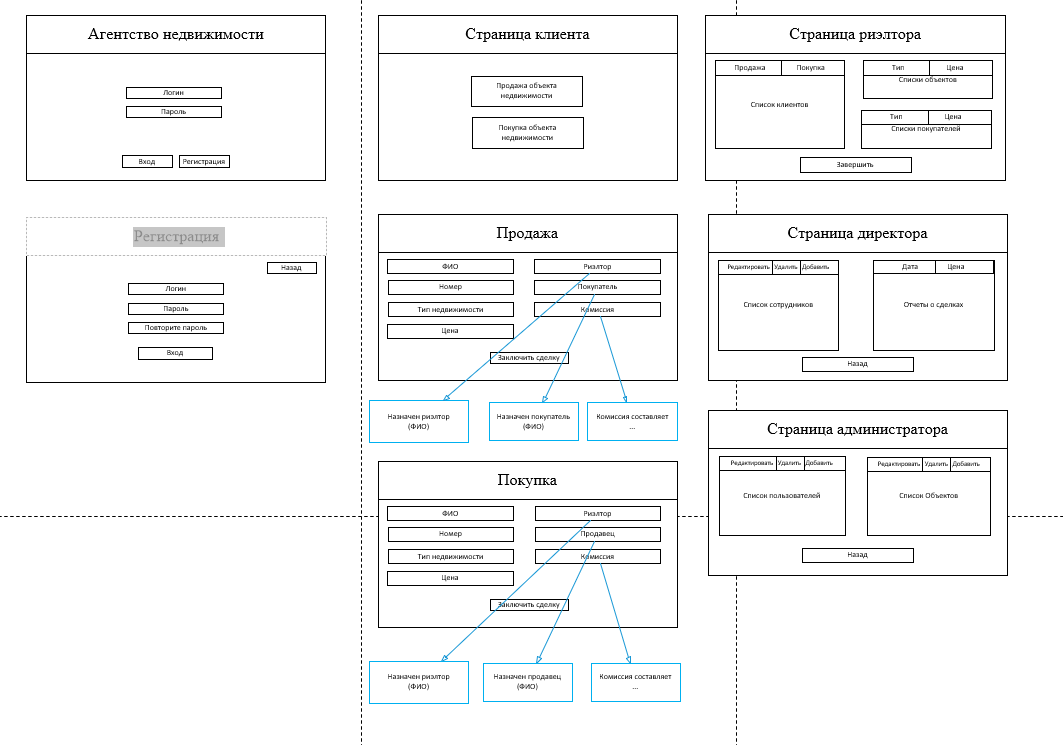
ПРИЛОЖЕНИЕ В



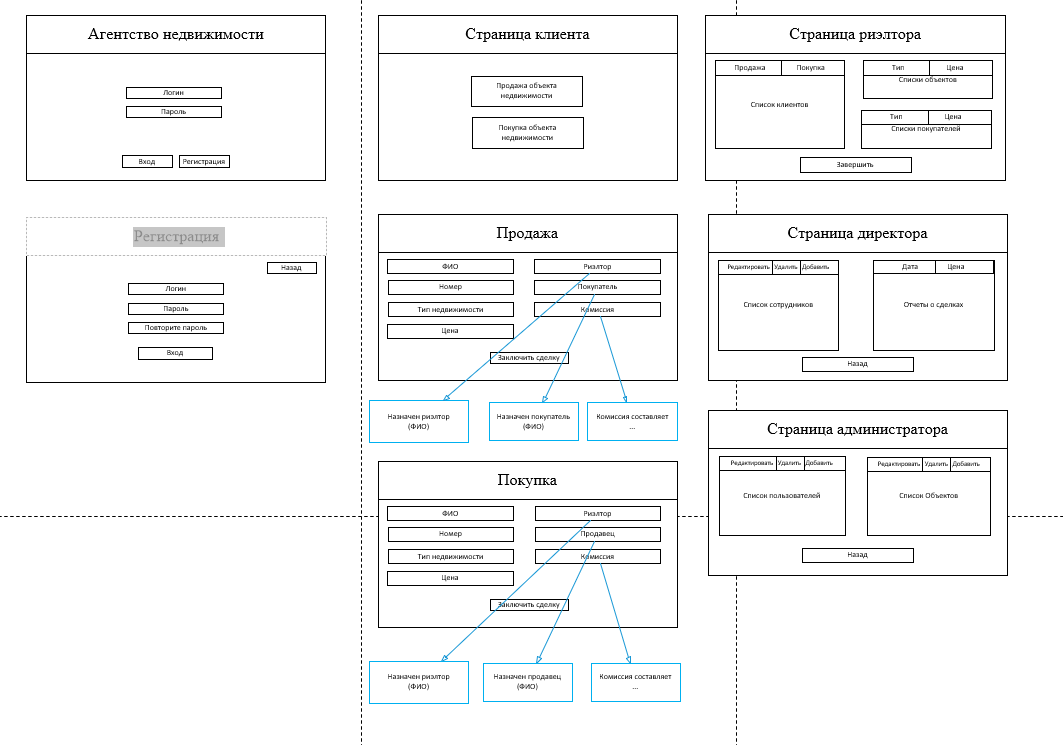
ПРИЛОЖЕНИЕ Г



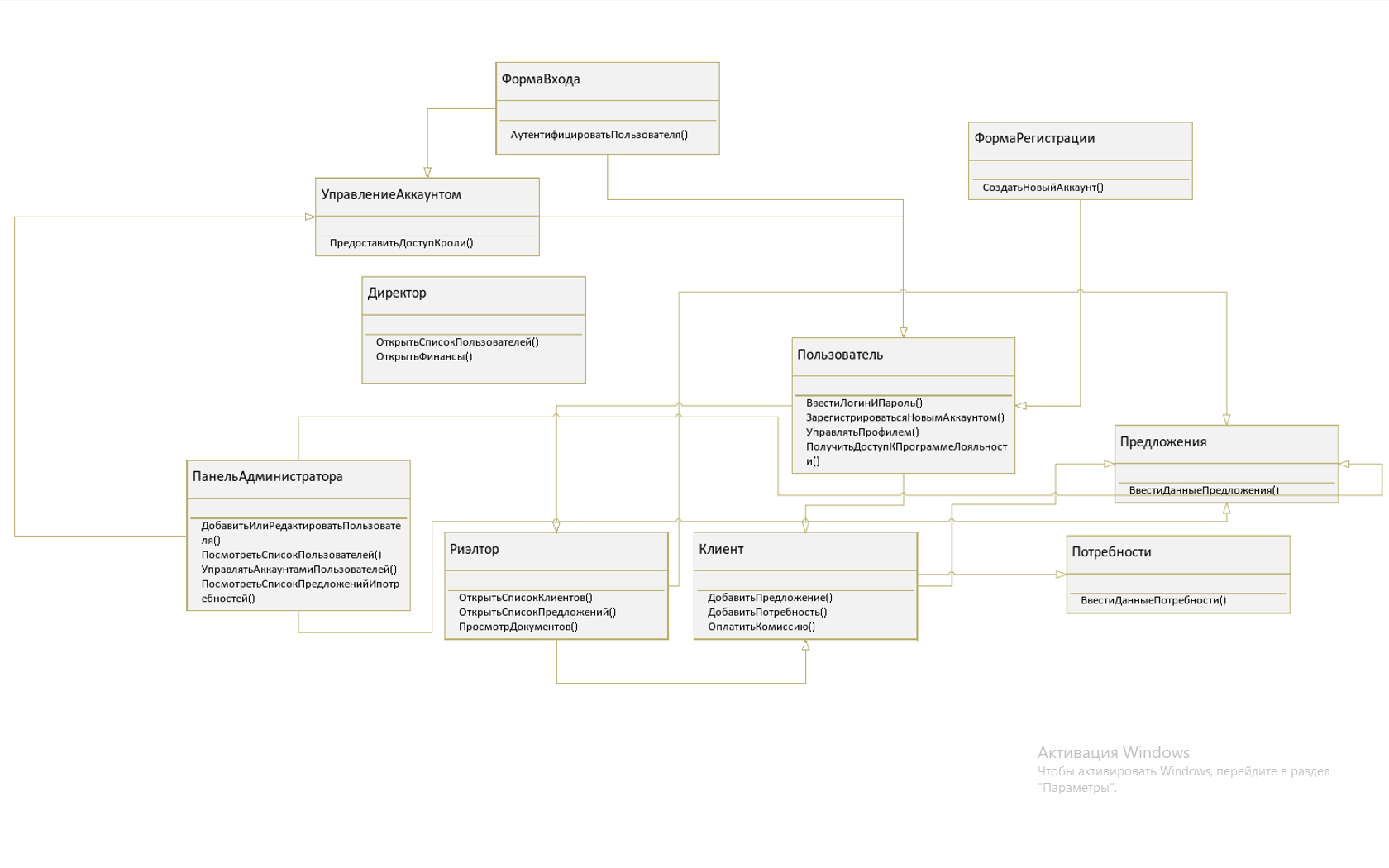
Продолжение прил. Г



Окончание прил. Г



ПРИЛОЖЕНИЕ Д

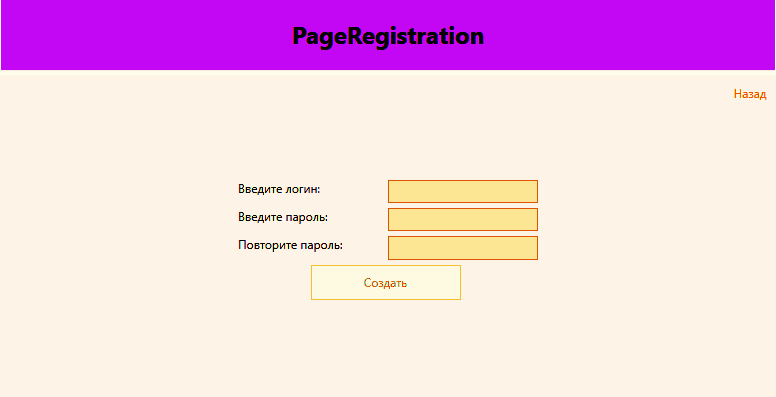


ПРИЛОЖЕНИЕ Е

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Dictionary | | | | |
| **Client** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | **Id** | INT | Y | Auto Increment |
|  | **Login** | NCHAR (100) | Y |  |
|  | **Password** | NCHAR (20) | Y |  |
|  | **IdRole** | INT | Y |  |
| **Roaltor** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | **Id** | INT | Y | Auto Increment |
|  | **Login** | NCHAR (100) | Y |  |
|  | **Password** | NCHAR (20) | Y |  |
|  | **IdRole** | INT | Y |  |
| **Role** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | **Id** | INT | Y | Auto Increment |
|  | **Name** | NCHAR (10) | Y |  |
| **PropertyType** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | **Id** | INT | Y | Auto Increment |
|  | **Name** | NCHAR (10) | Y |  |
| **Buyers** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | **Id** | INT | Y | Auto Increment |
|  | **Name** | NCHAR (10) | Y |  |
|  | **IdType** | INT | Y |  |
|  | **Price** | INT | Y |  |
| **Objects** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | **Id** | INT | Y | Auto Increment |
|  | **Name** | NCHAR (10) | Y |  |
|  | **IdType** | INT | Y |  |
|  | **Price** | INT | Y |  |
| **Sale** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | **Id** | INT | Y | Auto Increment |
|  | **Name** | NCHAR (10) | Y |  |
|  | **Price** | INT | Y |  |
|  | **Commission** | INT | Y |  |
|  | **IdPropertyType** | INT | Y |  |
|  | **IdRealtor** | INT | Y |  |
|  | **IdUser** | INT | Y |  |
|  | **Data** | DATA TYPE / FIELD SIZE | Y |  |
| **Purchase** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | **Id** | INT | Y | Auto Increment |
|  | **Name** | NCHAR (10) | Y |  |
|  | **Price** | INT | Y |  |
|  | **Commission** | INT | Y |  |
|  | **IdPropertyType** | INT | Y |  |
|  | **IdRealtor** | INT | Y |  |
|  | **IdUser** | INT | Y |  |
|  | **Data** | DATA TYPE / FIELD SIZE | Y |  |

ПРИЛОЖЕНИЕ Ж





**Аттестационный лист**

Обучающийся Ананьев Артём Алексеевич

ФИО

*3* курса по специальности *09.02.07 Информационные системы и программирование*

код и наименование специальности

прошел(ла) учебную практику по профессиональному модулю

*ПМ.05. Проектирование и разработка информационных систем*

индекс и название профессионального модуля

в объеме *72* часа с « *17* »  *марта*  2025 г. по « *21* »  *марта*  2025 г.

с « *14* »  *апреля*  2025 г. по « *18* »  *апреля*  2025 г.

*в частном профессиональном образовательном учреждении*

*«Нижегородский экономико-технологический колледж»*

\_\_*в лаборатории организации и принципов построения информационных систем*

название лаборатории, кабинета

Критерии оценки:

5 баллов – качество выполненных заданий высокое, ярко выраженное, проявляется всегда;

4 балла – качество выполненных заданий среднее, проявляется в зависимости от ситуации;

3 балла – качество выполненных заданий умеренно выраженное, проявляется редко;

2 балла – качество выполненных заданий выражено слабо, не выражено.

**Виды и качество выполнения работ в период учебной практики**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Умения, формируемые в составе профессиональных компетенций** | **Виды работ,**  **выполненных обучающимся во время практики** | **Объем работ, час.** | **Оценка** | |
| ПК 5.1, 5.2:   * создавать и управлять проектом по разработке приложения; * проводить анализ предметной области; * осуществлять выбор модели и средства построения информационной системы и программных средств; | * Выполнение анализа предметной области * Разработка модели архитектуры информационной системы | 18 |  | |
| ПК 5.3, 5.4:   * проектировать и разрабатывать систему по заданным требованиям и спецификациям; * разрабатывать графический интерфейс приложения; | * Проектирование и разработка пользовательского интерфейса в соответствии со спецификациями; * Интеграция модуля в информационную систему | 24 | |  |
| ПК 5.5:   * решать прикладные вопросы программирования и языка сценариев для создания программ; | * Разработка логики приложения * Разработка классов для решения вычислительных задач | 12 | |  |
| ПК 5.6, 5.7   * осуществлять постановку задач по обработке информации; * использовать алгоритмы обработки информации для различных приложений. | * Обработка исключений * Тестирование программного обеспечения | 18 | |  |

Программа учебной практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

освоена / не освоена

Итоговая оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ *Кисарова М.В.*

«  *18*  »  *апреля*  2025 г.

Зав. производственной практикой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ *Цапаева А.А.*

Ознакомлен: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Ананьев А.А.

«  *18*  »  *апреля*  2025 г.