ОПИСАНИЕ

В период практической подготовки вам необходимо проделать все действия, приведенные в примере. Итоговые результаты необходимо представить в формате отчетной презентации PowerPoint, с визуализацией каждого этапа выполненной работы.

Скриншоты: 103, 107, 113, 115, 120, 121 должны отображать картинки и изображения, связанные с вашей предметной областью. Логику проектирования по ранее выбранной предметной области в учебной практике по профессиональному модулю ПМ.03 [Проектирование](https://lms.synergy.ru/lntools/versiongroupassign/contents/student/13057821) и разработка информационных систем в текущем периоде практической подготовки обязательно необходимо продолжить. Скриншоты не должны быть одинаковы у нескольких студентов.

# Этап 1.

## Установка JDK

Для того, чтобы установить необходимое программное обеспечение, необходимо скачать, разархивировать необходимые файлы, показанные на рисунке 1, и запустить установку.



Рисунок. 1. Расположение установщиков

Первым необходимо установить Java Development Kit. Двойным кликом по установщику, после появится диалоговое окно с приветствием, как показано на рисунке 2. После чего необходимо нажать на Next.

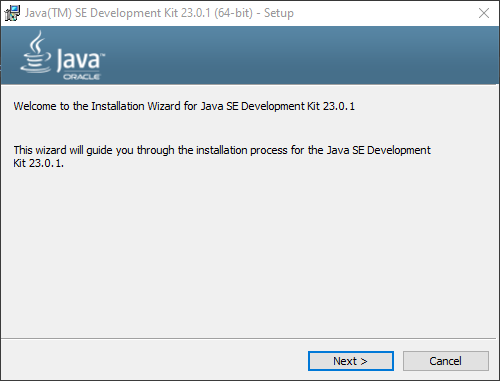


Рисунок. 2. Приветствие

В следующем окне, показанном на рисунке 3, необходимо выбрать путь для загрузки программ. В данном случае нужно не вносить изменения и нажать кнопку Next.

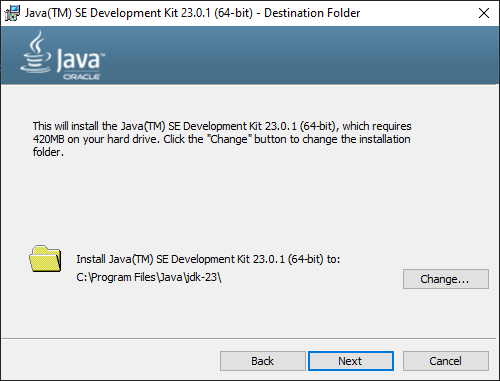


Рисунок. 3. Путь установки

После чего начнется установка программы, как показано на рисунке 4.

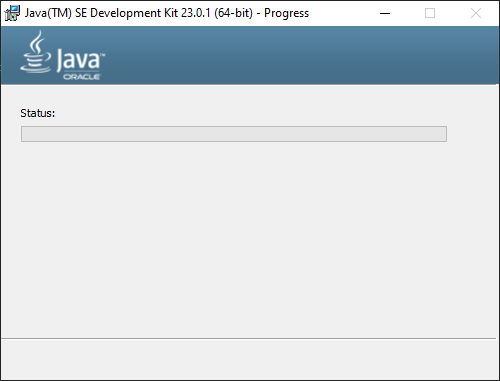


Рисунок. 4. Установка программы

После этого появится первое с сообщением об успешной установке программы, как показано на рисунке 5.

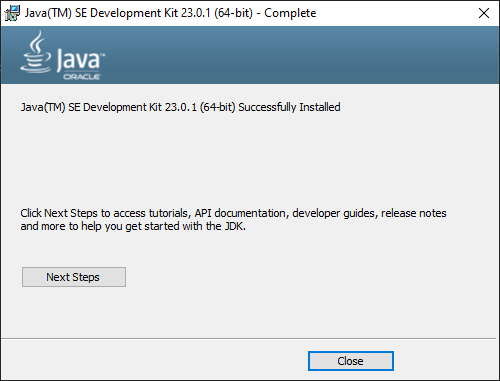


Рисунок. 5. Завершение установке

При необходимости можно ознакомиться с документацией, которая откроется после нажатия на кнопку Next Steps, как показано на рисунке 6.

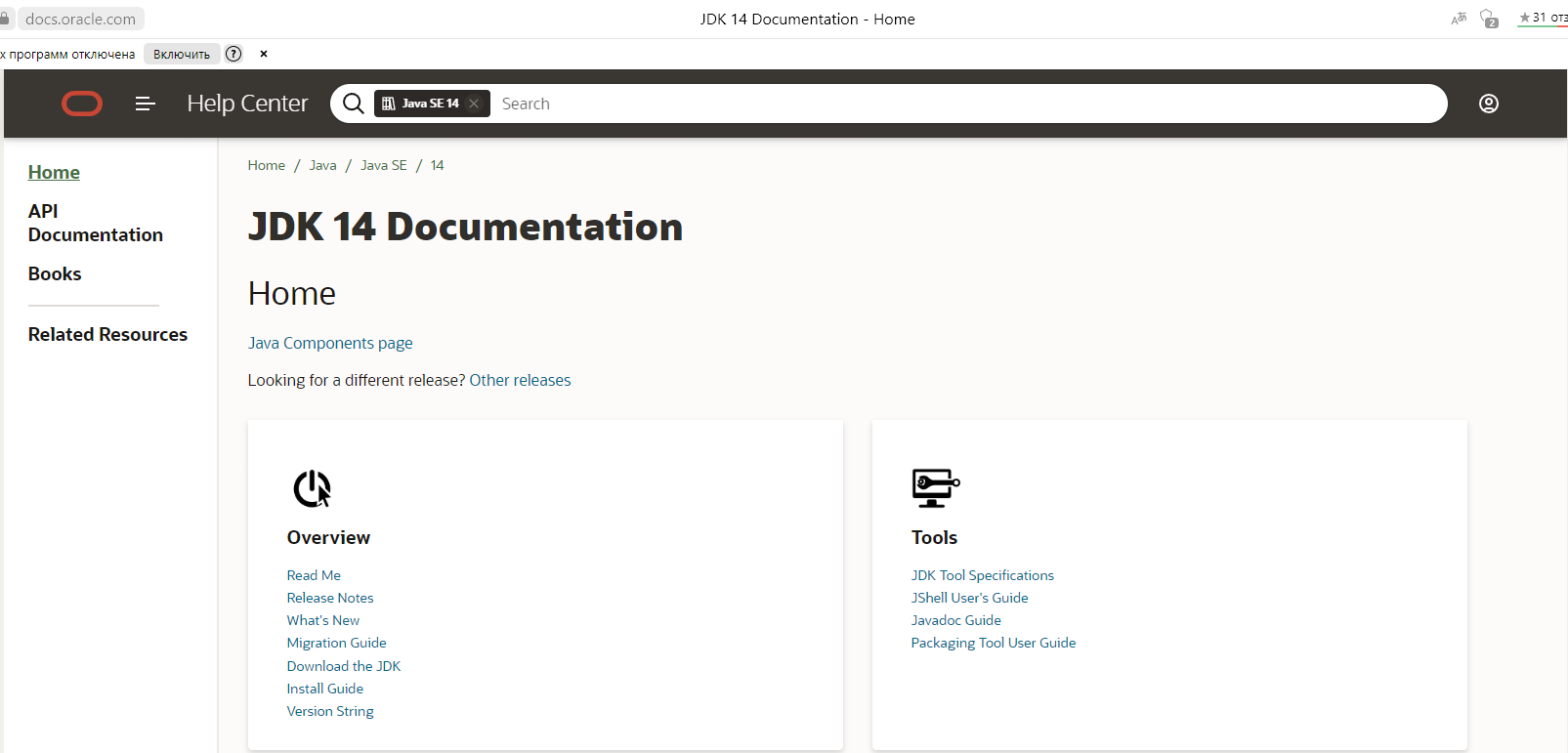


Рисунок. 6. Документация

## Установка Eclipse

Необходимо два раза кликнуть на файл-установщик, после чего появится следующее изображение, показанное на рисунке 7.



Рисунок. 7. Запуск установщика

Далее требуется выбрать нужный тип среды разработчика. Необходимо выбрать самый первый, как показано на рисунке 8.

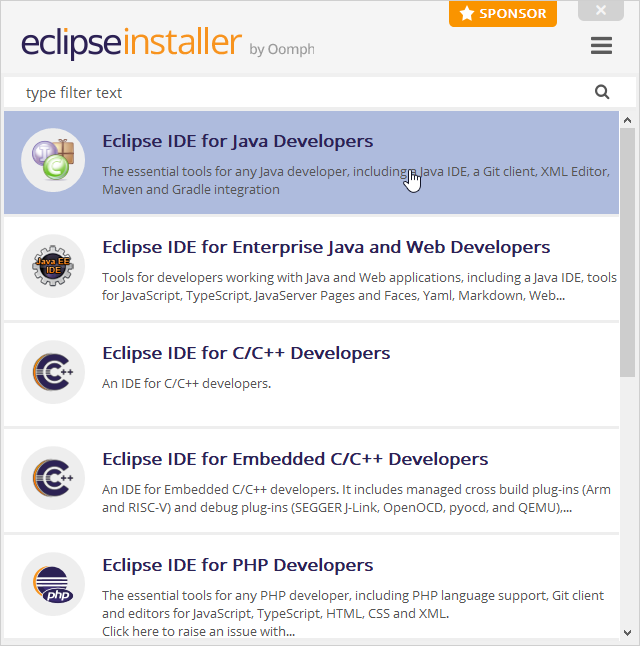


Рисунок. 8. Выбор ПО

Затем предлагается выбрать нужные опции, после чего нужно нажать на Install, как показано на рисунке 9.

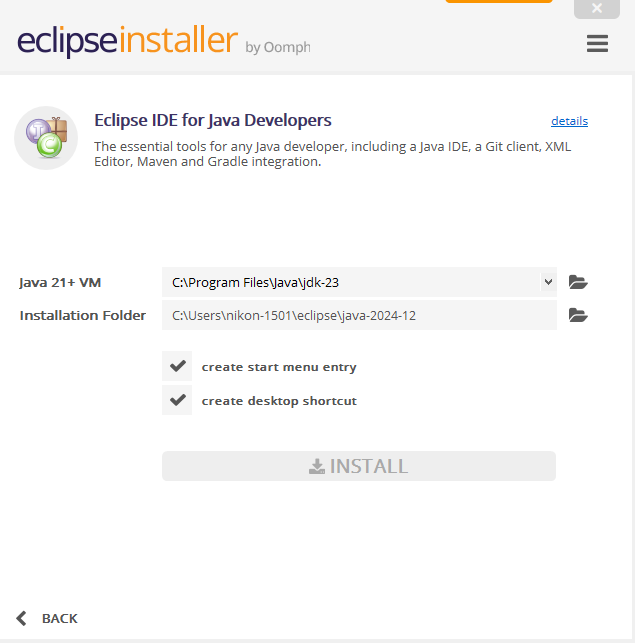


Рисунок. 9. Выбор расположения

После этого будет открыто дополнительное диалоговое окно с лицензионным соглашением, как показано на рисунке 10.

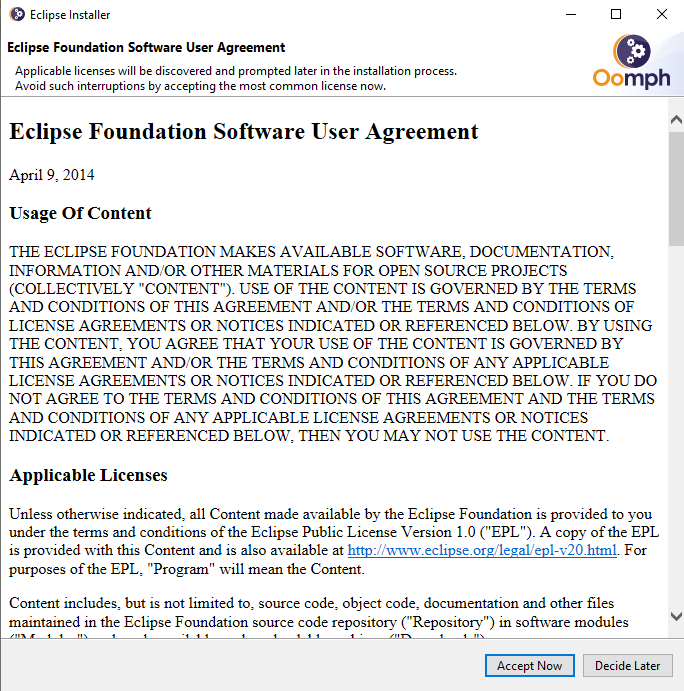


Рисунок. 10. Согласие

Далее последует непосредственная установка программы, как показано на рисунке 11.

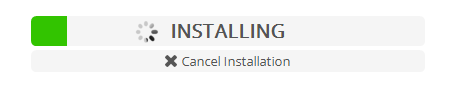


Рисунок. 11. Установка

Если выбрать Launch, будет показан следующее изображение, как показано рисунке 13.



Рисунок. 13. Запуск программы

Далее будет открыто диалоговое окно, предлагающее выбрать рабочую область, как показано рисунке 14. Можно всё оставить без изменений и нажать Launch.

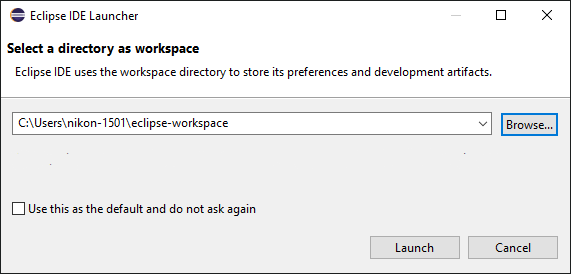


Рисунок. 14. Выбор директории

Далее снова будет показан баннер, показываемый при запуске программы, как показано на рисунке 15.



Рисунок. 15. Начало

После этого программа будет запущена, как показано на рисунке 20.

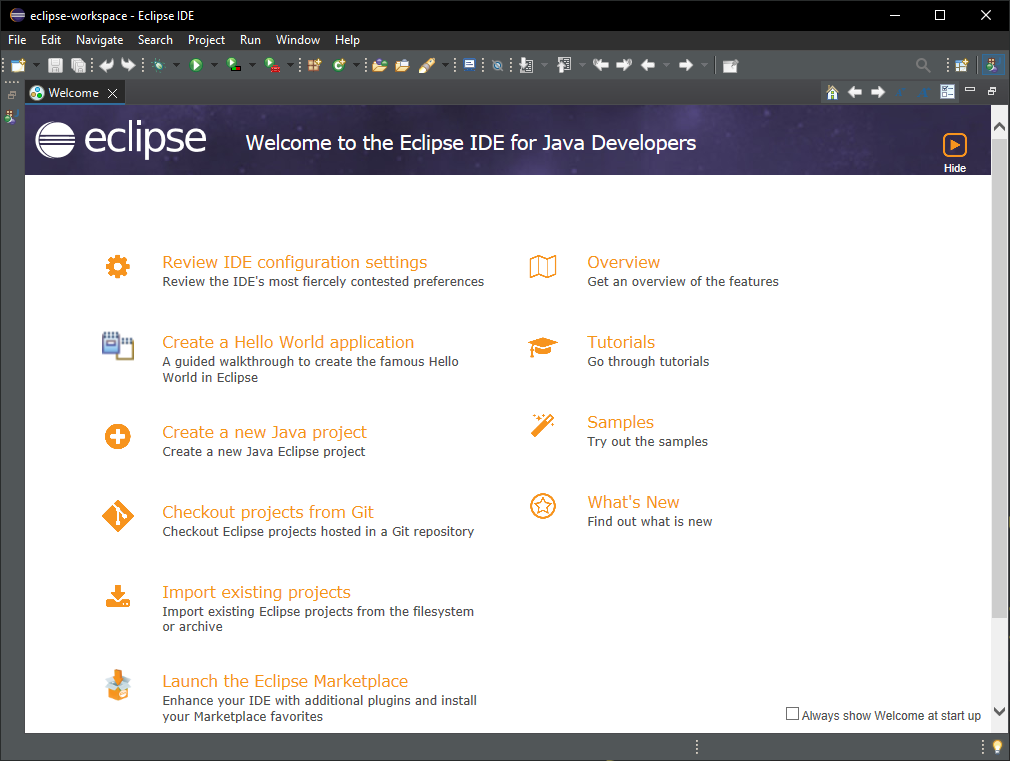


Рисунок. 16. Установка программы

## Установка Apache NetBeans

Открываем установщик Apache, после чего появляется окно с приветствием, для продолжения необходимо нажать Next, как показано на рисунке 17.



Рисунок. 17. Открытие установщика

После подтверждаем лицензионное соглашение и нажимаем Next, как показано на рисунке 18.

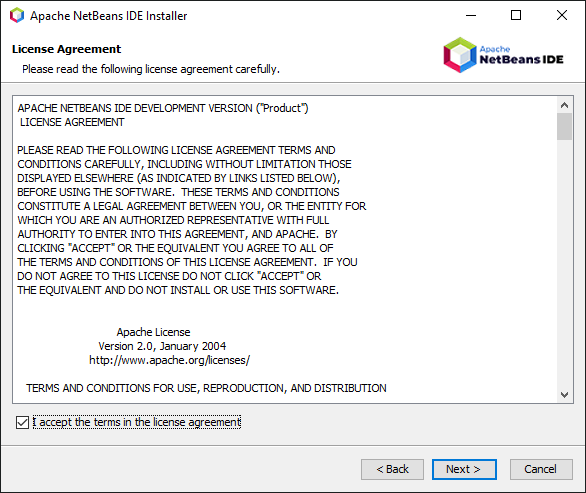


Рисунок. 18. Соглашение с лицензией

Далее необходимо выбрать путь, куда будет установлен Apache. Оставляем без изменений. Нажимаем Next, как показано ан рисунке 19.

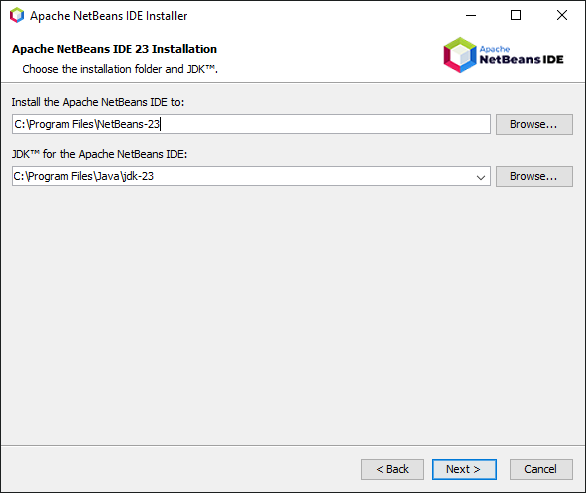


Рисунок. 19. Выбор пути

Для того, чтобы начать установку, нужно нажать кнопку установить.

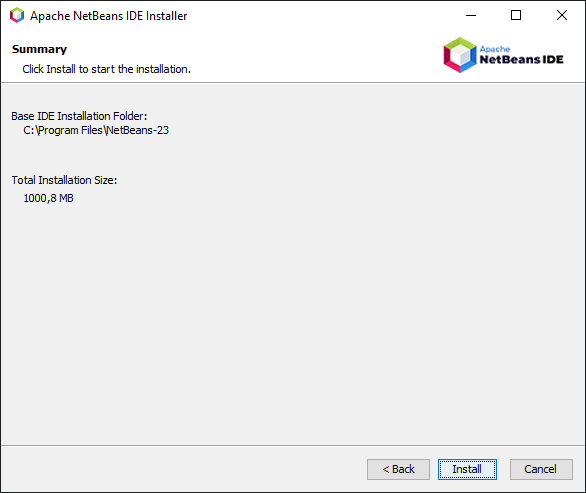


Рисунок. 20. Начало установки

После выполненных действий появится окно загрузки, показанное на рисунке 21.

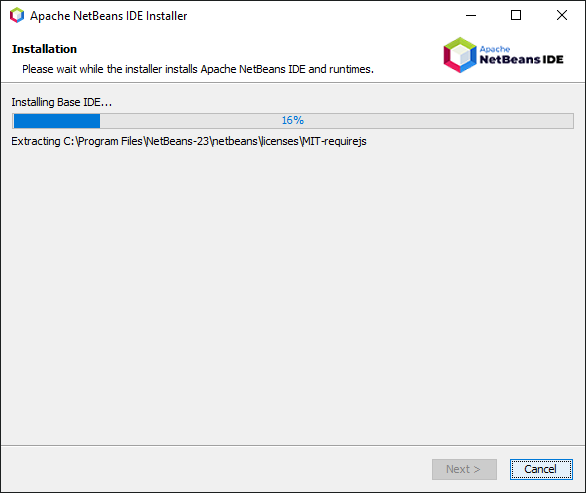


Рисунок. 21. Установка

Установка прошла успешно, нажимаем Finish.

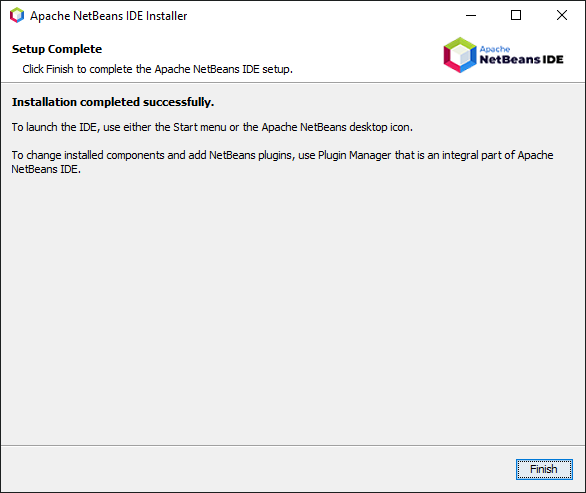


Рисунок 1. Установка завершена.

# Этап 2.

## Запуск Eclipse и создание проекта

Для начала необходимо запустить Eclipse, после необходимо создать проект, нажав на File, затем New, Java Project, как изображено на рисунке 22.

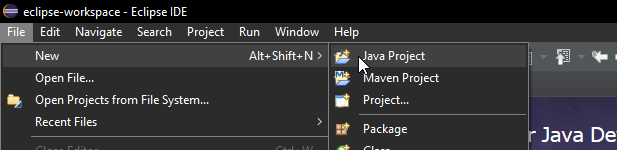


Рисунок. 22. Создание проекта

Далее появится диалоговое окно, в котором требуется выбрать нужные настройки, как показано на рисунке 23. В данном случае необходимо ввести имя проекта, выбрать JRE (JavaSE-13).

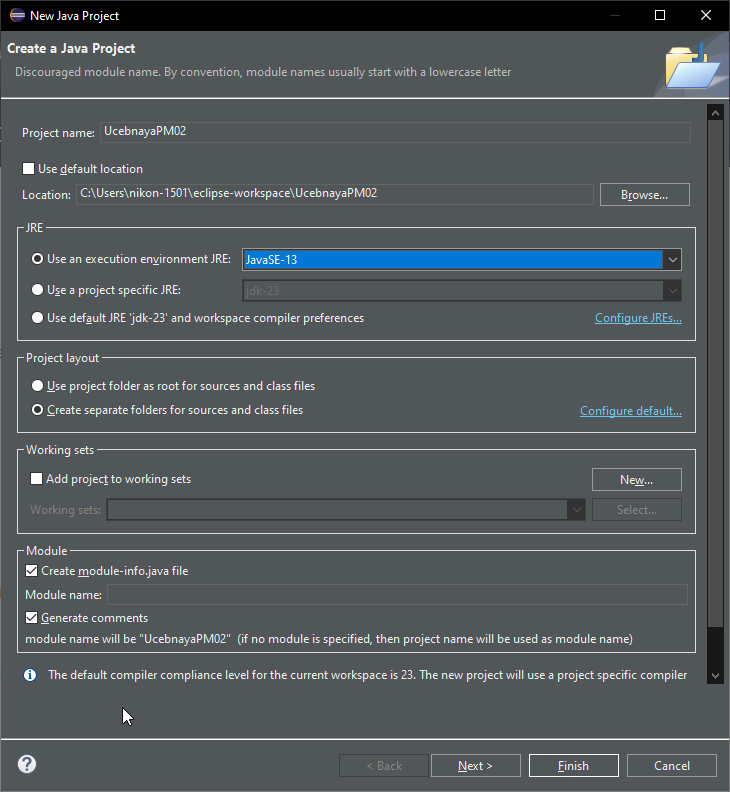


Рисунок. 23. Необходимые настройки

После этого появится созданный проект, как показано на рисунке 24..

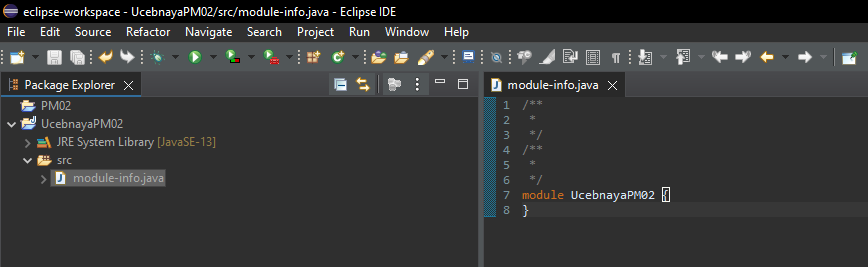


Рисунок. 24. Навигатор

## Создание класса

Далее необходимо нажать правой кнопкой мыши на src, затем выбрать New, после Class, как показано на рисунке 25.

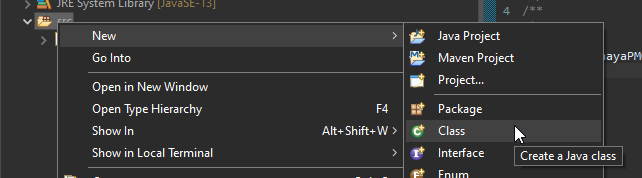


Рисунок. 25. Создание класса

После этого будет откроется диалоговое окно с выбором параметров, где необходимо ввести имя класса и установить галочку (public static void main), как показано на рисунке 26.

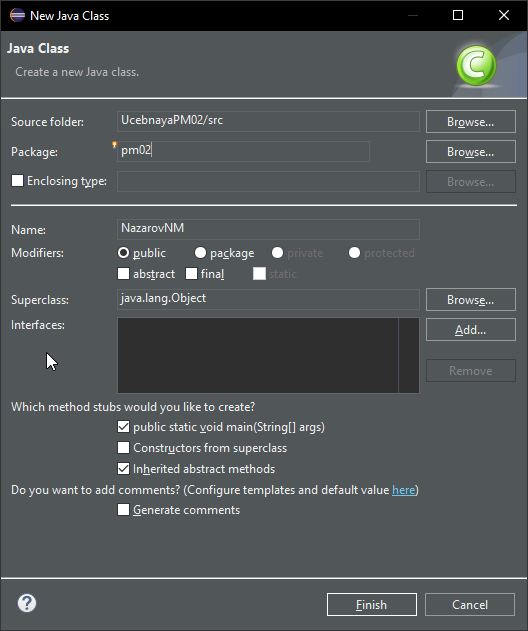


Рисунок. 26. Настройки класса

После этого класс появится в проекте, как показано на рисунке 27.

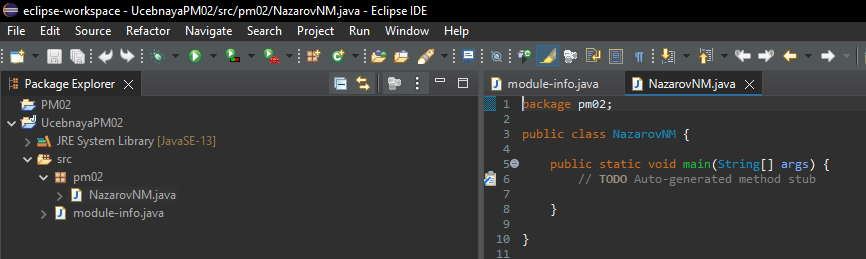


Рисунок. 27. Созданный класс

## Вывод надписи в консоль

Для того, чтобы вывести надпись в консоль необходимо использовать конструкцию system.out.println(), как показано на рисунке 28.

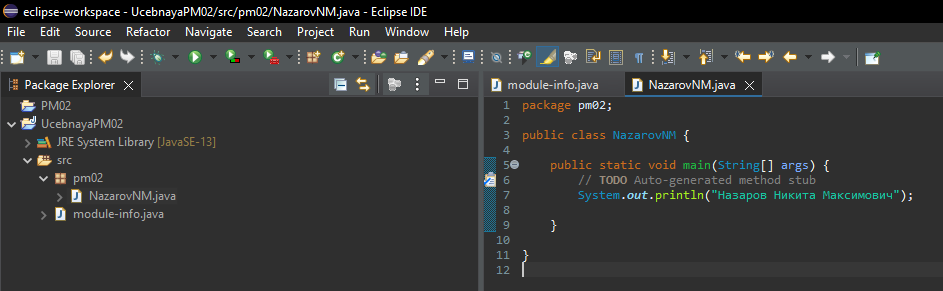


Рисунок. 28. Код первого задания

Для запуска проекта необходимо нажать на кнопку запуска в левом верхнем углу, как показано на рисунке 29.

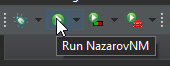


Рисунок. 29. Запуск

После запуска программы в консоли появится результат работы.

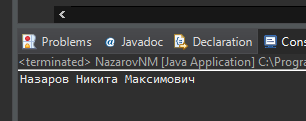


Рисунок. 30. Результат первого задания

## Вывод слов в столбец

Для того, чтобы построчно вывести слова, необходимо использовать команду system.out.println(). Код задания показан на рисунке 31.

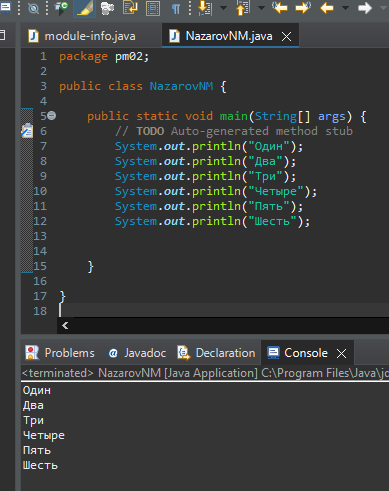


Рисунок. 31. Код второго задания

## Вывод возраста в одной строке

Для выполнения задания необходимо обозначить новую переменную, в которой будет храниться возраст. Ниже показан код выполнения задания двумя способами.

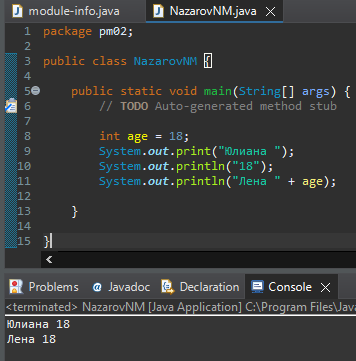


Рисунок. 33. Код третьего задания

# Этап 3.

## Создание проекта в NetBeans

Перед началом работы необходимо создать новый проект, как показано на рисунке 35.

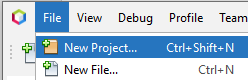


Рисунок. 35. Создание проекта

После откроется диалоговое окно, в котором необходимо выбрать категорию и тип проекта так, как показано на рисунке 36.

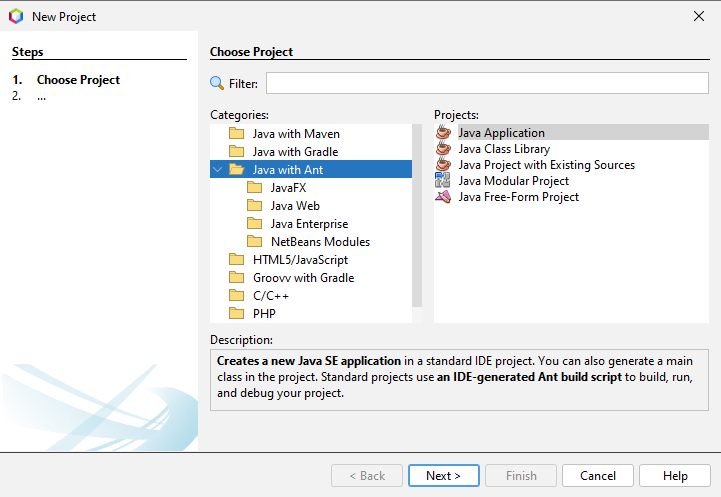


Рисунок. 36. Выбор типа проекта

Далее нужно выбрать имя проекта и путь, а так же убрать галочку, как показано на рисунке 37.



Рисунок. 37. Имя и расположения проекта

Созданный проект показан на рисунке 38.



Рисунок. 38. Созданный проект

## Создание формы

Далее необходимо создать форму, нажав правой кнопкой мыши на название проекта, как показано на рисунке 39.

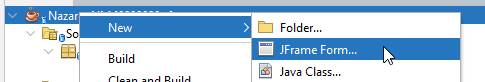


Рисунок. 39. Создание формы

В диалоговом окне требуется выбрать имя и расположение формы.

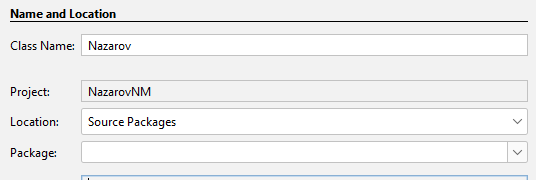


Рисунок. 40. Параметры формы

После чего создается форма, как показано на рисунке 41.

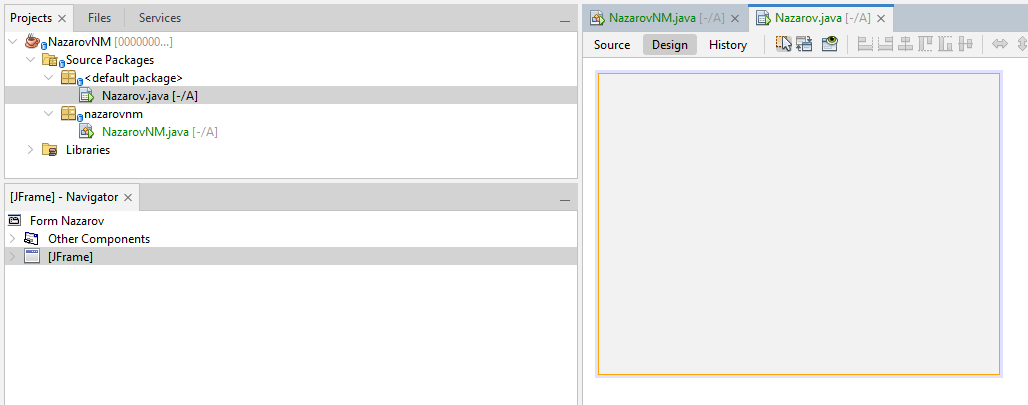


Рисунок. 41. Созданная форма

## Создание элементов формы

Далее нужно перетащить кнопку на форму, как показано на рисунке 42.



Рисунок. 42. Добавление кнопки

Для того, чтобы изменить текст кнопки, необходимо поменять значение в свойствах элемента, как показано на рисунке 43.



Рисунок. 43. Текст кнопки

В обработчике событий требуется создать код, показанный на рисунке 44.



Рисунок. 44. Обработчик события

Готовый проект показан на рисунке 45.

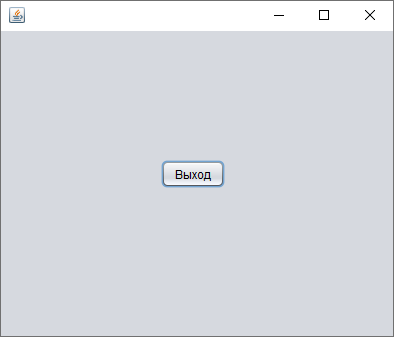


Рисунок. 45. Запущенная программа

После закрытия программы в консоли будет показано, что сборка завершена.

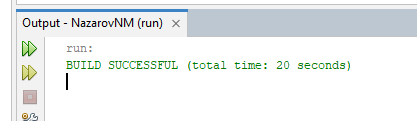


Рисунок. 46. Сборка завершена

Далее нужно добавить еще элементы на форму, а так же изменить в них текст, как показано на рисунке 47.

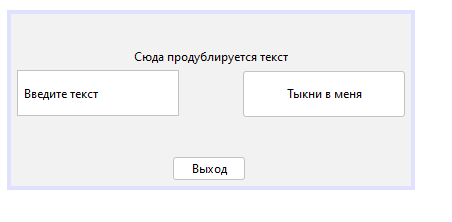


Рисунок. 47. Новые элементы

На рисунке 48 показан код в обработчике событий новой созданный кнопки.



Рисунок. 48. Обработчик события

При запуске программы в текстовое поле можно вводить значения, а нажав на кнопку, введенный текст перенесется в лейбл, как показано на рисунке 49.

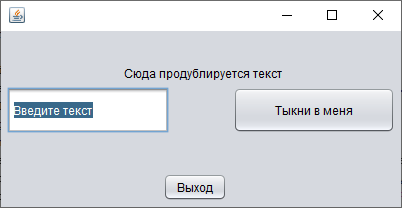


Рисунок. 49. Результат работы

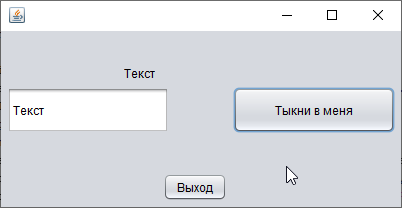


Рисунок. 50. Результат после нажатия

После нужно добавить еще элементы на форму, как показано на рисунке 51.

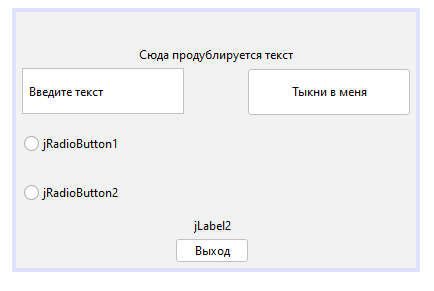


Рисунок. 51. Добавление новых элементов

Далее необходимо добавить элемент buttonGroup, который видно в свойствах, как показано на рисунке 52.

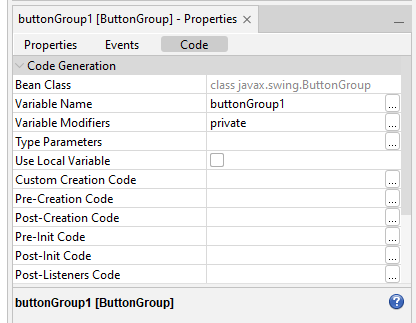


Рисунок. 52. Свойства группы

Для группировки переключателей необходимо им обоим в свойство buttonGroup добавить созданный элемент buttonGroup1, как показано на рисунке 53.

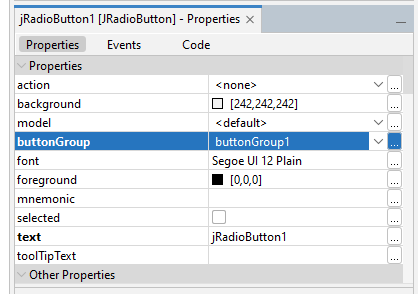


Рисунок. 53. Группировка

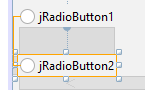


Рисунок. 54. Сгруппированы

Также нужно изменить текст элементов, как показано на рисунке 55.

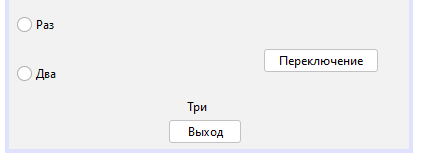


Рисунок. 55. Текст элементов

Код обработчика событий показан на рисунке 56.

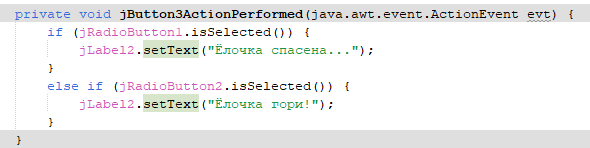


Рисунок. 56. Обработчик события

В результате при выборе переключателя и нажатия на кнопку будет меняться текст в метке, как показано на рисунках 57 и 58.

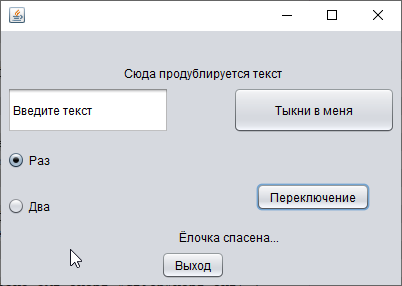


Рисунок. 57. Выбор первого радио



Рисунок. 58. Выбор второго радио

Готовое и запущенное приложение показано на рисунке 59.

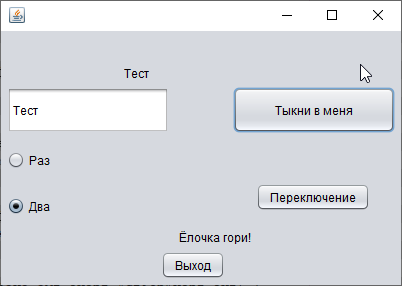


Рисунок. 59. Готовый запущенный проект с графическим интерфейсом

# Этап 4.

## Создание мини-калькулятора.

Для создания мини-калькулятора понадобиться написание кода, показанного на рисунке 60.

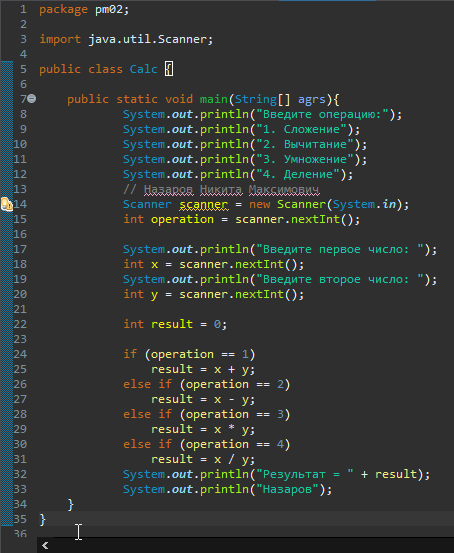


Рисунок. 60. Код программы

Результат работы программы при выборе операции «Сложение», показан рисунке 61.

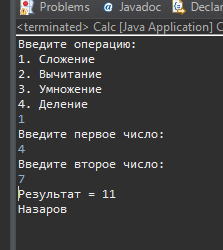


Рисунок. 61. Результат работы операции сложение

Результат работы программы при выборе операции «Вычитание», показан рисунке 62.

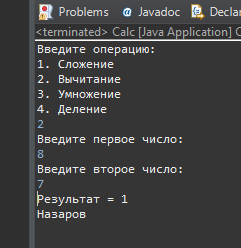


Рисунок. 62. Результат работы операции вычитание

Результат работы программы при выборе операции «Умножение», показан рисунке 63.

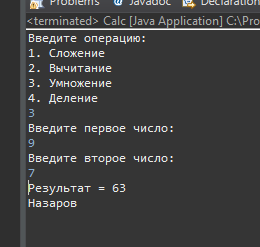


Рисунок. 63. Результат работы операции умножение

Результат работы программы при выборе операции «Деление», показан рисунке 64.

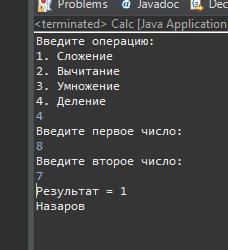


Рисунок. 64. Результат работы операции деление

# Этап 5.

## Установка символов табуляции в качестве отступов

Перед началом работы необходимо запустить Eclipse, после чего в меню выбрать Window Preferences, как показано на рисунке 65.

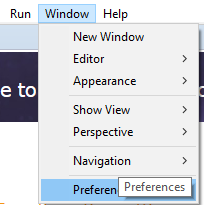


Рисунок. 65. Выбор Preferences

Далее будет открыто диалоговое окно, в котором нужно открыть Java -> Code Style -> Formatter, как показано на рисунке 66.

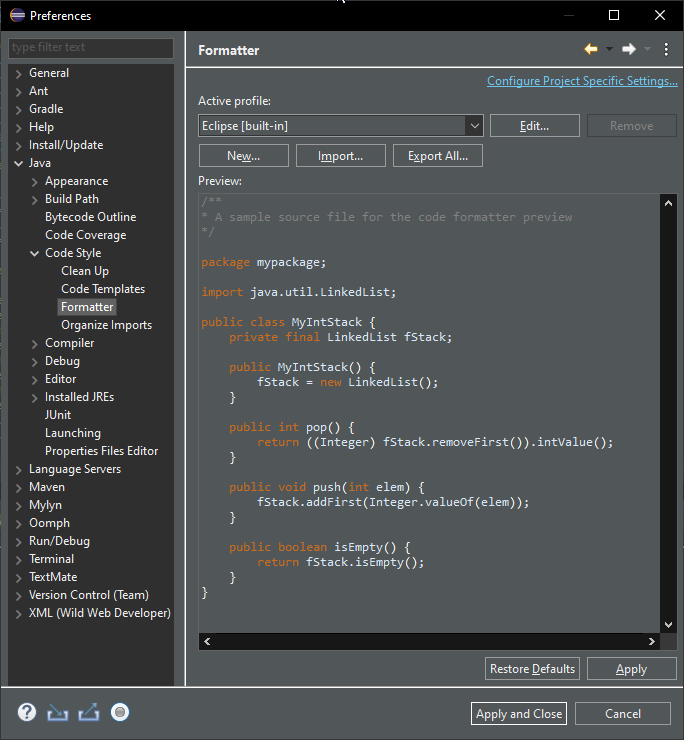


Рисунок. 66. Раздел Formatter

В качестве Active profile необходимо выбрать Java Conventions [built-in], как показано на рисунке 67.

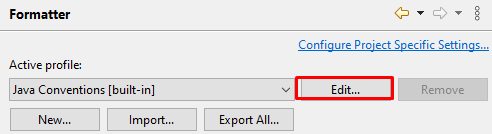


Рисунок. 67. Выбор Java Conventions

После чего будет открыто диалоговое окно, показанное на рисунке 68.

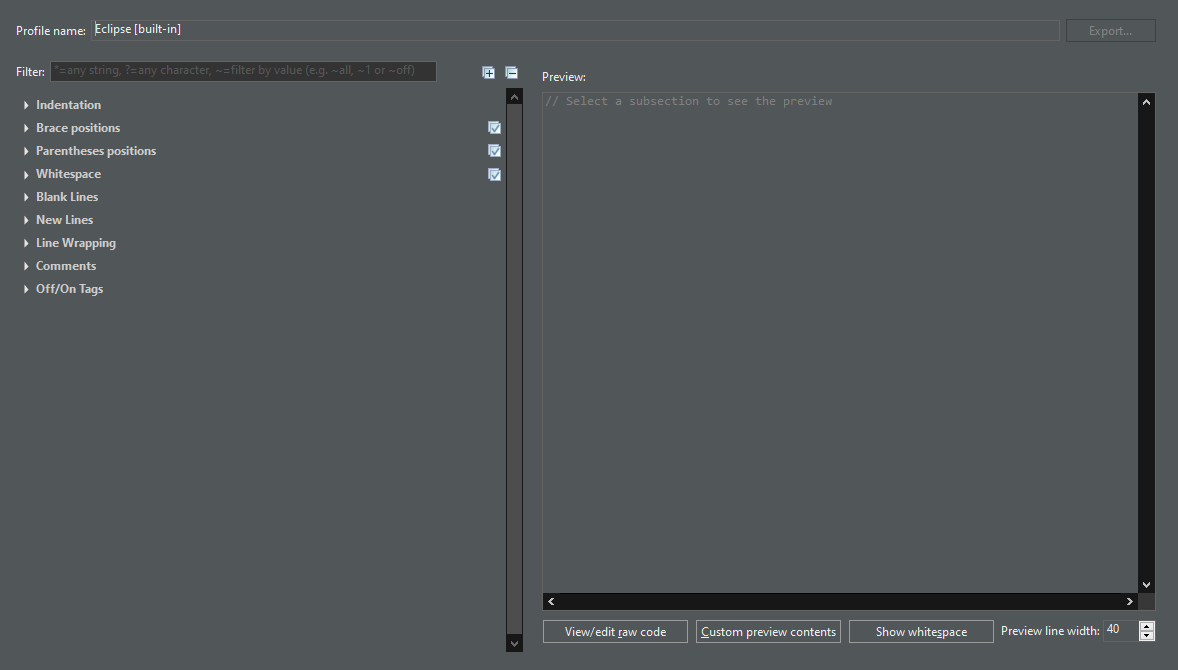


Рисунок. 68. Диалоговое окно Java Conventions

В нем нужно открыть вкладку Indentation и установить значение Tabs only, как показано на рисунке 69.

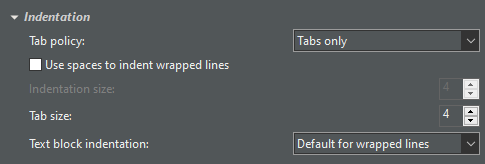


Рисунок. 69. Установка Tabs only

После этого нужно закрыть диалоговое окно Preferences, как показано на рисунке 70.

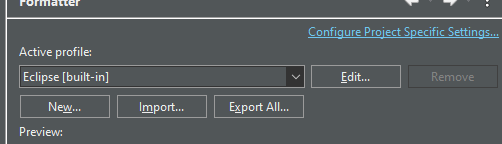


Рисунок. 70/ Закрытие диалогового окна Preferences

В результате отступами в коде приложения будут только символы табуляции.

## Переименование элементов проекта

Для того, чтобы переименовать класс нужно нажать на файл класса правой кнопкой мыши, затем Refractor -> Rename, как показано на рисунке 71.

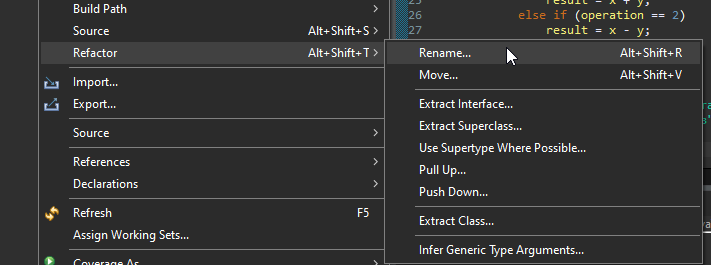


Рисунок. 71. Выбор Rename

После этого появится диалоговое окно, в котором нужно записать новое имя класса, как показано на рисунке 72.

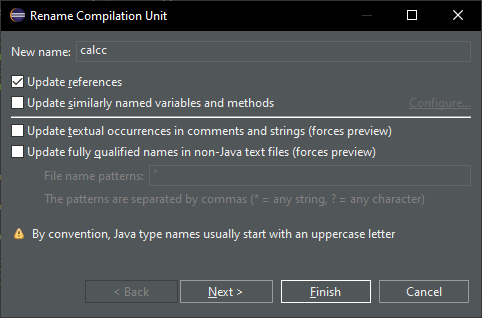


Рисунок. 72. Изменение имени класса

После появится предупреждающее о возможных проблемах в проекте, как показано на рисунке 73.

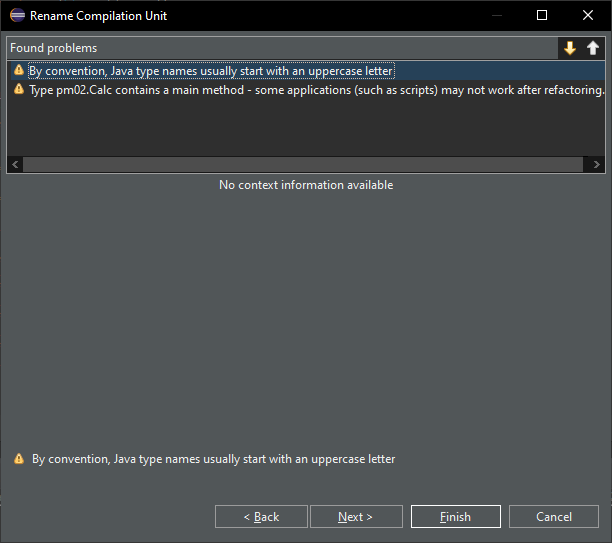


Рисунок. 73. Диалоговое окно предупреждения

Далее нужно переименовать переменную, выделив название, нажать правой кнопкой мыши, затем Rename, как показано на рисунке 74.

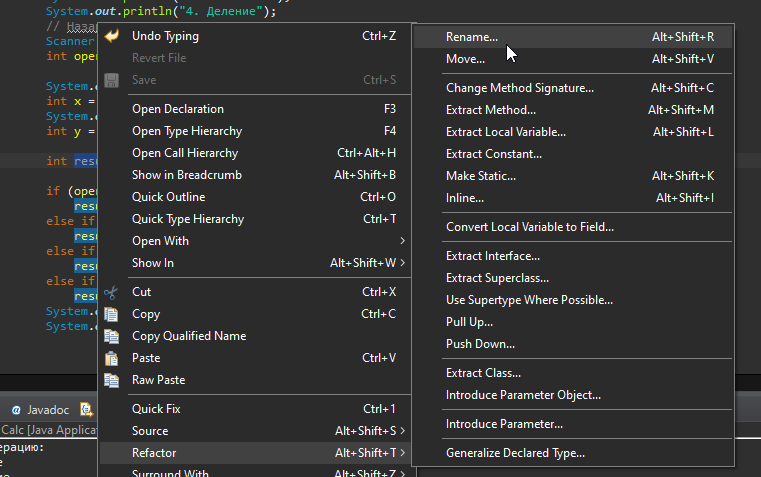


Рисунок. 74. Выбор Rename

После этого надо ввести новое название переменной и после этого нажать Enter (рисунок 93).

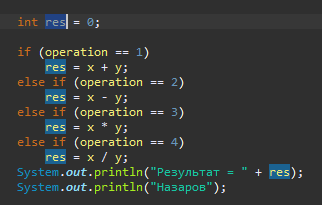


Рисунок. 75. Изменение названия переменной

## Создание нового пакета

Для того, чтобы создать новый пакет нужно выбрать проект, затем New, Package, как показано на рисунке 76.

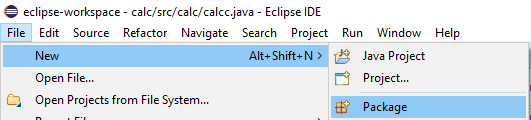


Рисунок. 76. Выбор создания нового пакета

После этого будет открыто диалоговое окно, в котором нужно ввести название нового пакета.

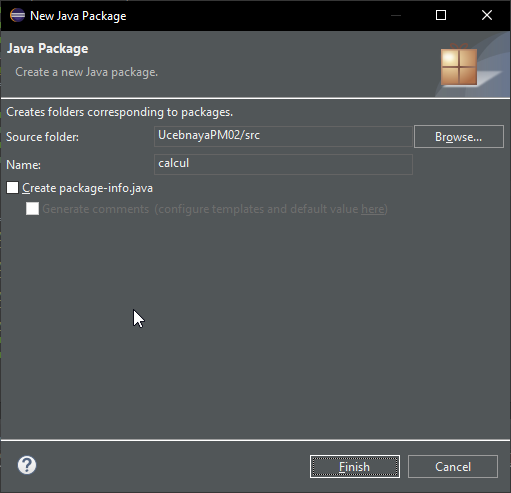


Рисунок. 77. Диалоговое окно создания пакета

После этого новый пакет будет создан.



Рисунок. 78. Новый пакет создан

## Перемещение класса из одного пакета в другой

Для того, чтобы переместить класс из одного пакета в другой, требуется нажать правой кнопкой мыши на название класса, потом Refractor, Move, как показано на рисунке 79.

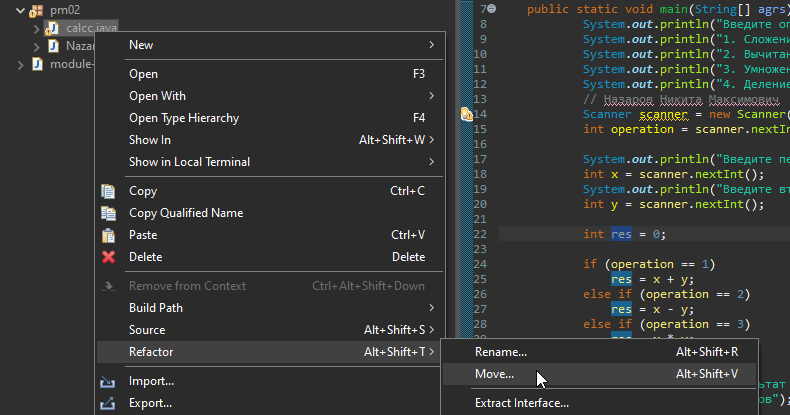


Рисунок. 79. Выбор Move

После этого будет открыто диалоговое окно, в котором нужно выбрать пакет, в который будет перенесен класс, как показано на рисунке 80.

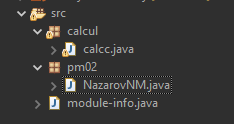


Рисунок. 80. Выбор пакета

На рисунке 81 можно увидеть, что класс был перенесен успешно.

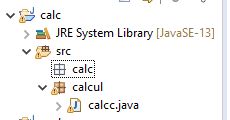


Рисунок. 81. Класс перенесен

## Экспорт проекта в архив zip

Для экспорта необходимо выбрать Export, как показано на рисунке 82.

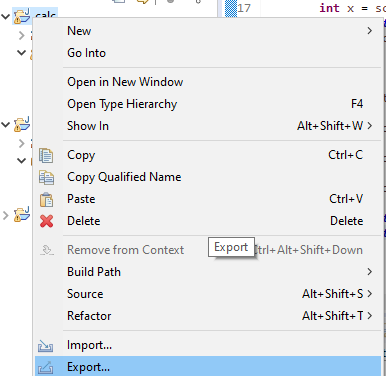


Рисунок. 82. Выбор Export

После чего будет открыто диалоговое окно, в котором нужно выбрать General -> Archive File.

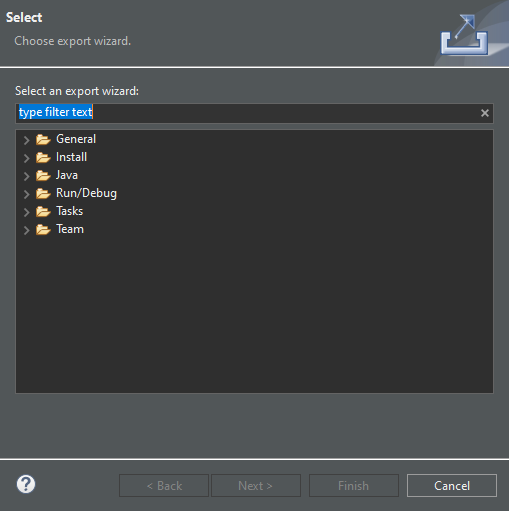


Рисунок. 83. Выбор Archive File

После этого необходимо выбрать путь и название создаваемого архива, как показано на рисунке 84.

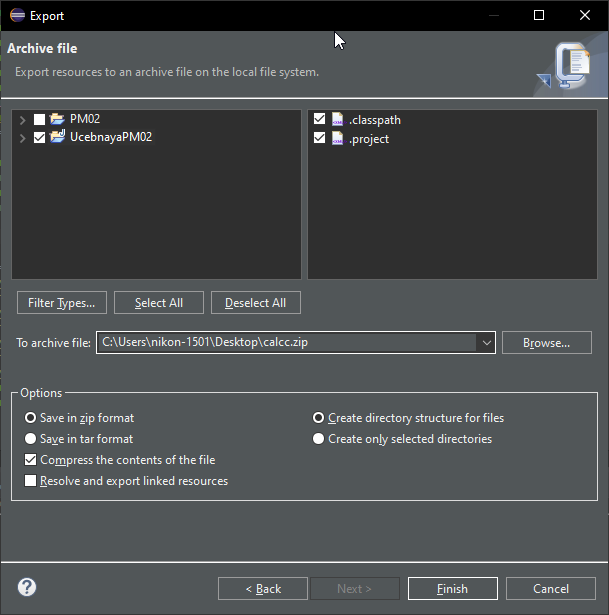


Рисунок. 84. Выбор пути

На рисунке 85 показано, что архив был создан успешно.

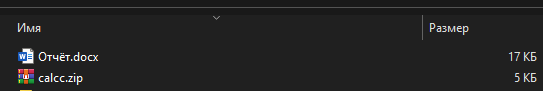


Рисунок. 85. Созданный архив

## Удаление проекта

Для удаления проекта необходимо выбрать в контекстном меню Delete, как показано на рисунке 86.

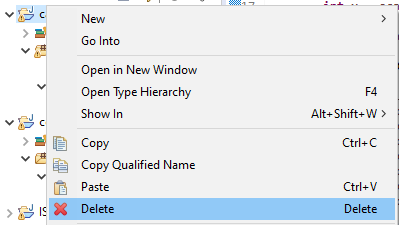


Рисунок. 86. Выбор Delete

В открывшемся окне необходимо установить галочку, как показано на рисунке 87.

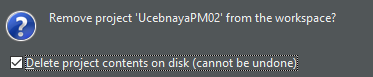


Рисунок. 87. Диалоговое окно удаления

## Импорт проекта из архива zip

Для импорта проекта необходимо в программе нажать File, а затем Import, как показано на рисунке 88.

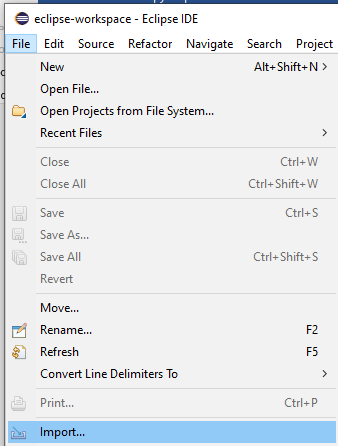


Рисунок. 88. Импорт

После этого в открывшемся окне нужно выбрать General, Existing project into Workspace, как показано на рисунке 89.

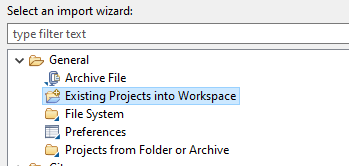


Рисунок. 89. Выбор импорта существующего проекта

Далее необходимо выбрать путь к архиву с проектом, как показано на рисунке 90.

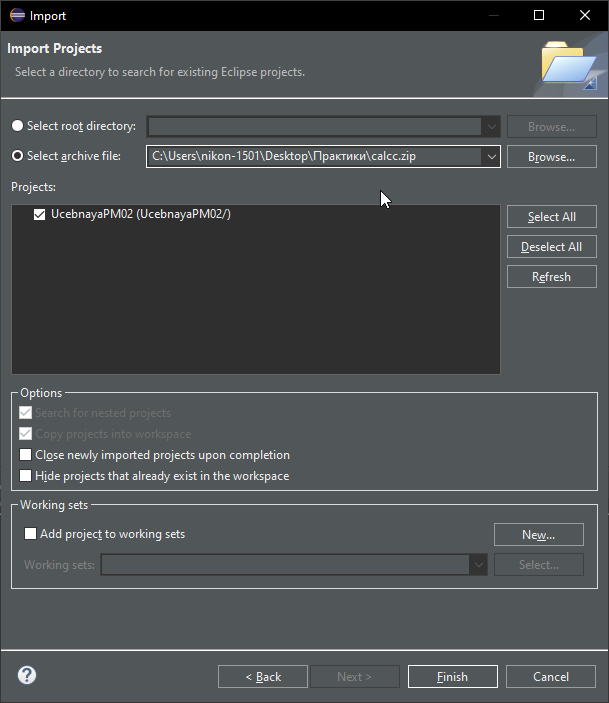


Рисунок. 90. Выбор пути к архиву с проектом

После проект будет импортирован в программу, как показано на рисунке 91.

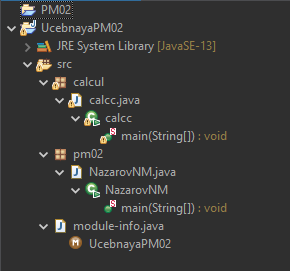


Рисунок. 91. Импортированный проект

# Этап 6.

Я выбрал для практики C# и WPF.

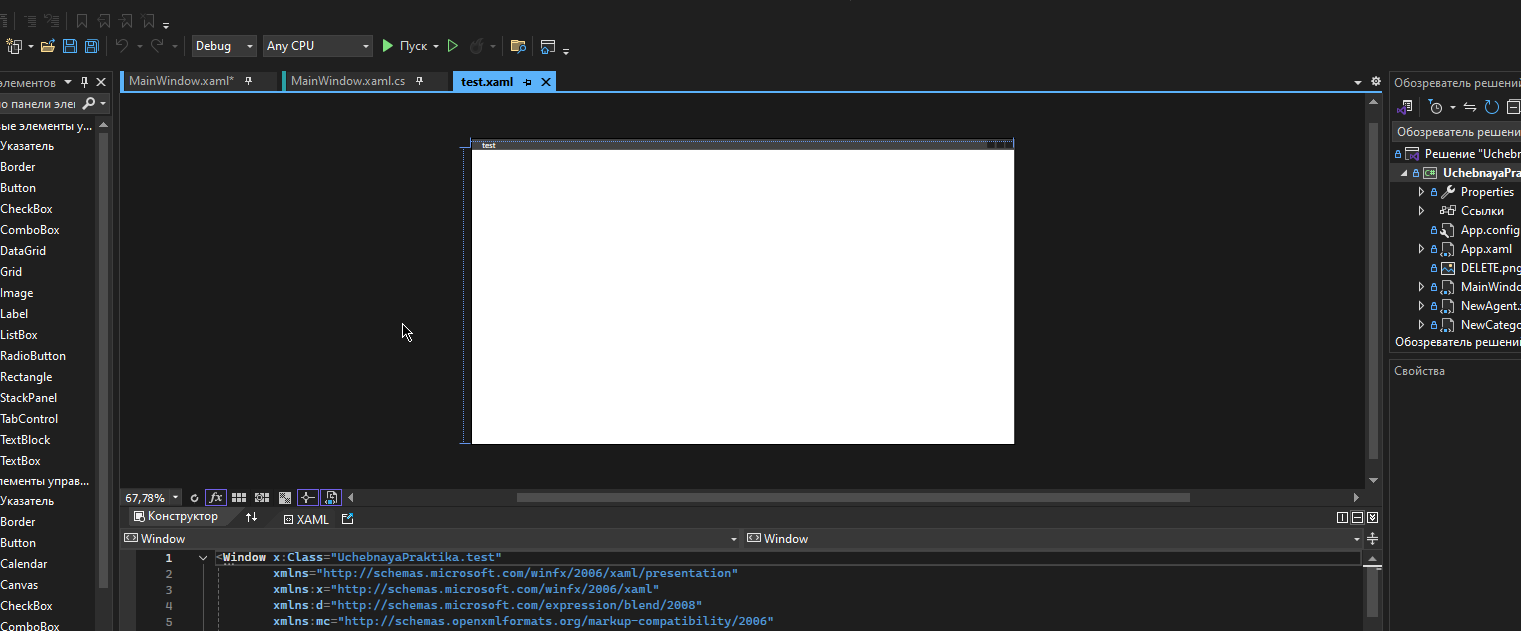


Рисунок 92. Пустое окно

Далее, так как у меня был некоторый план, как бы все элементы управления расположить, и какие они вообще должны быть, я расположил в черновом варианте всё, что пока что хотел:

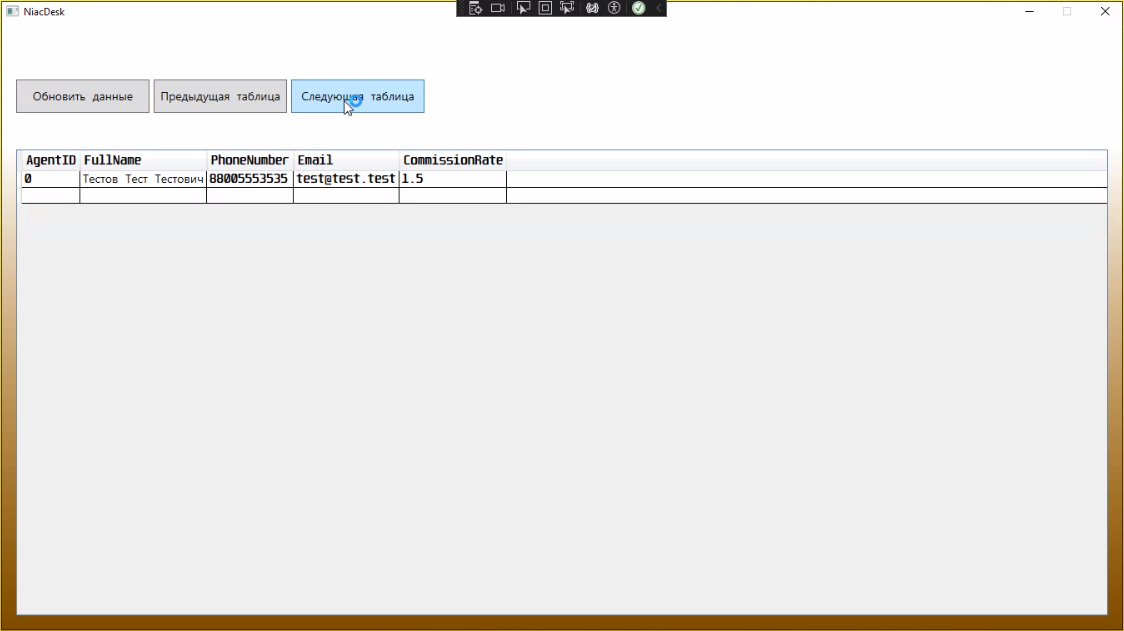


Рисунок 93. Начало работы

Сначала и кнопок навигации не было, поэтому для проверки DataGrid была кнопка «Обновить данные», позже, когда появились кнопки навигации, в них было прописано обновление данных при каждом нажатии:

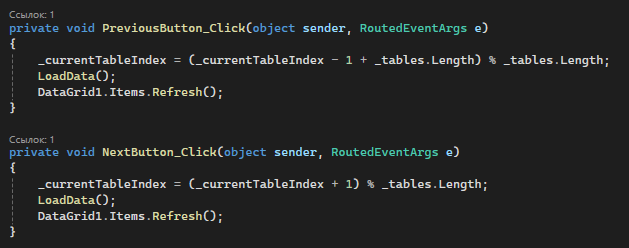


Рисунок 94. Кнопки «Следующая» и «Предыдущая таблица»

Сам DataGrid и кнопки навигации были реализованы следующим образом:

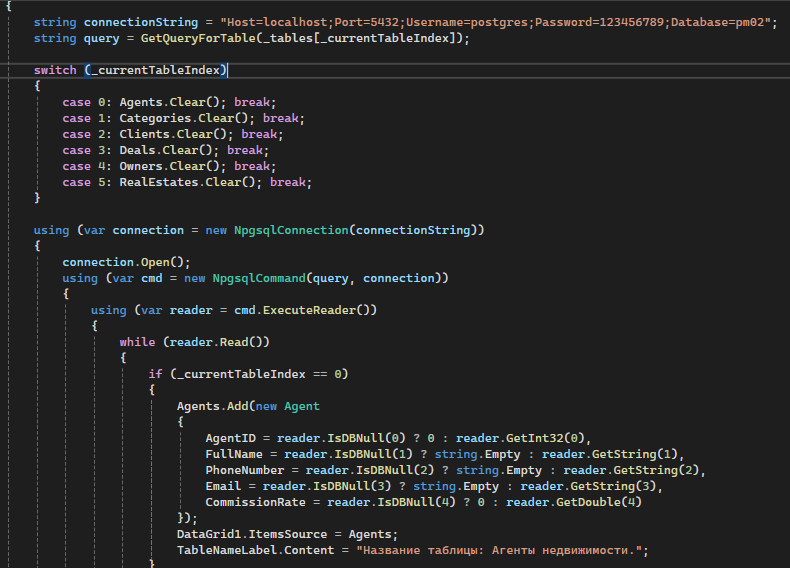


Рисунок 95. Метод переключения таблиц 1/4

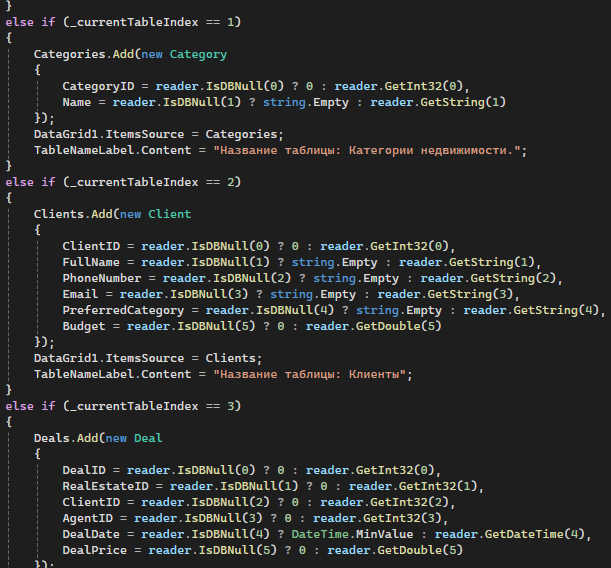


Рисунок 96. Метод переключения таблиц 2/4



Рисунок 97. Метод переключения таблиц 3/4

Определение SQL запроса для отрисовки нужной таблицы в DataGrid:

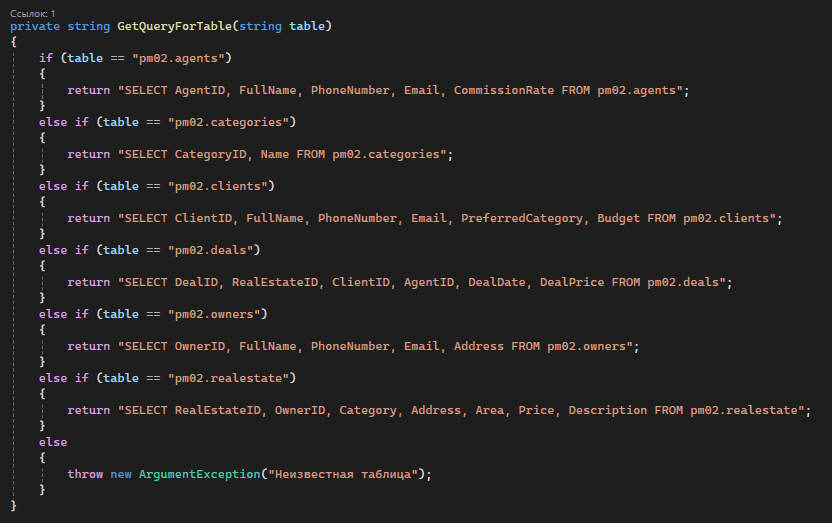


Рисунок 98. Метод переключения таблиц 4/4

Затем было добавлено меню:

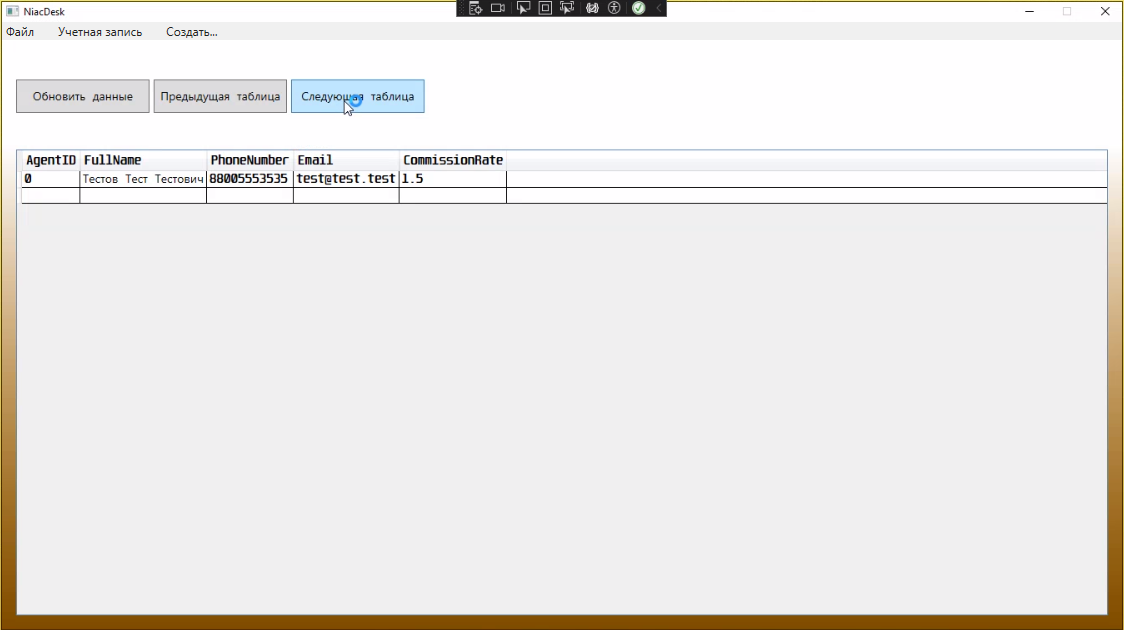


Рисунок 99. Меню

Со следующими выборами:

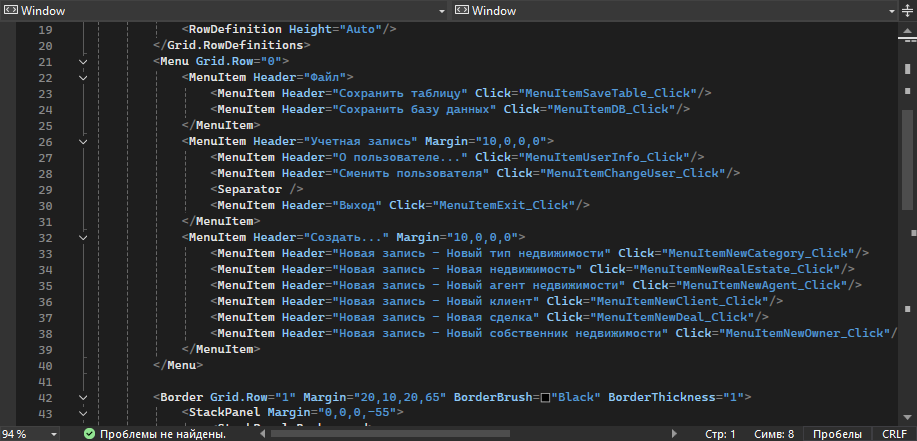


Рисунок 100. Окошки подменю

После чего была небольшая перестановка, добавлены изображения кнопкам, добавлена кнопка «Удалить запись» и я решил, что кнопка «Обновление данных» всё же останется, а также заполнил немного БД, чтобы пусто не выглядело:

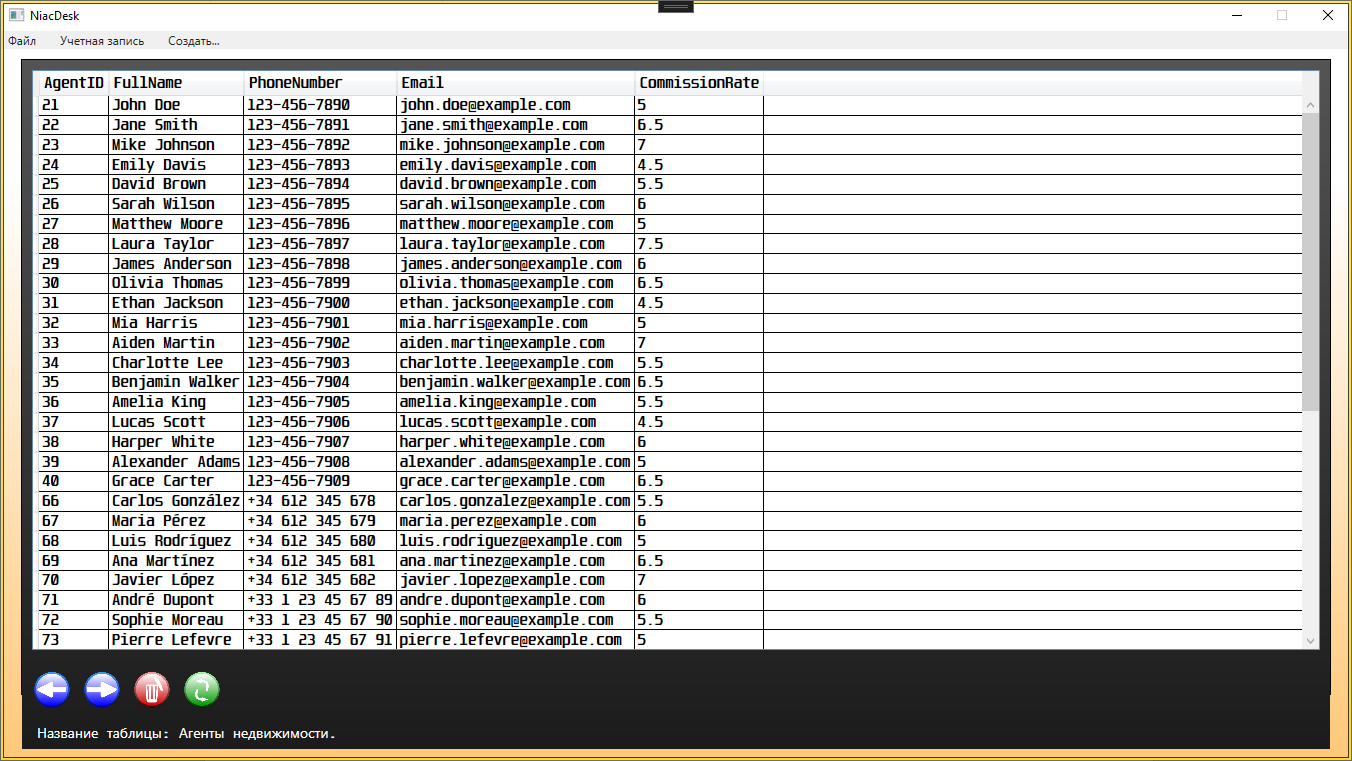


Рисунок 101. Конечный дизайн приложения

Кнопка «Удалить запись»:



Рисунок 102. Удаление записи

Если то, что нужно удалить где то используется, то БД не даст этого сделать, и выдаст ошибку, которая отобразиться в программе:

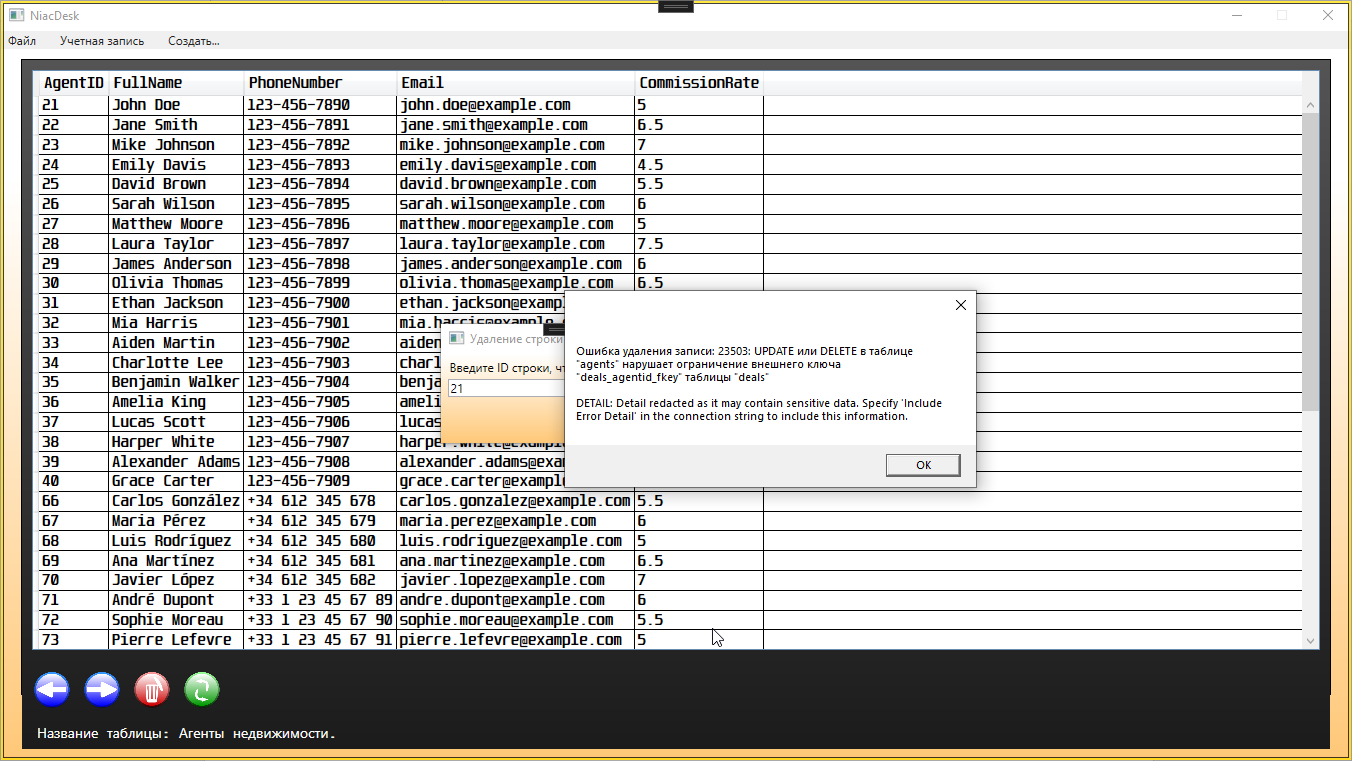


Рисунок 103. Попытка удаления связанных записей

Кнопки Меню:

Добавить новую категорию недвижимости:

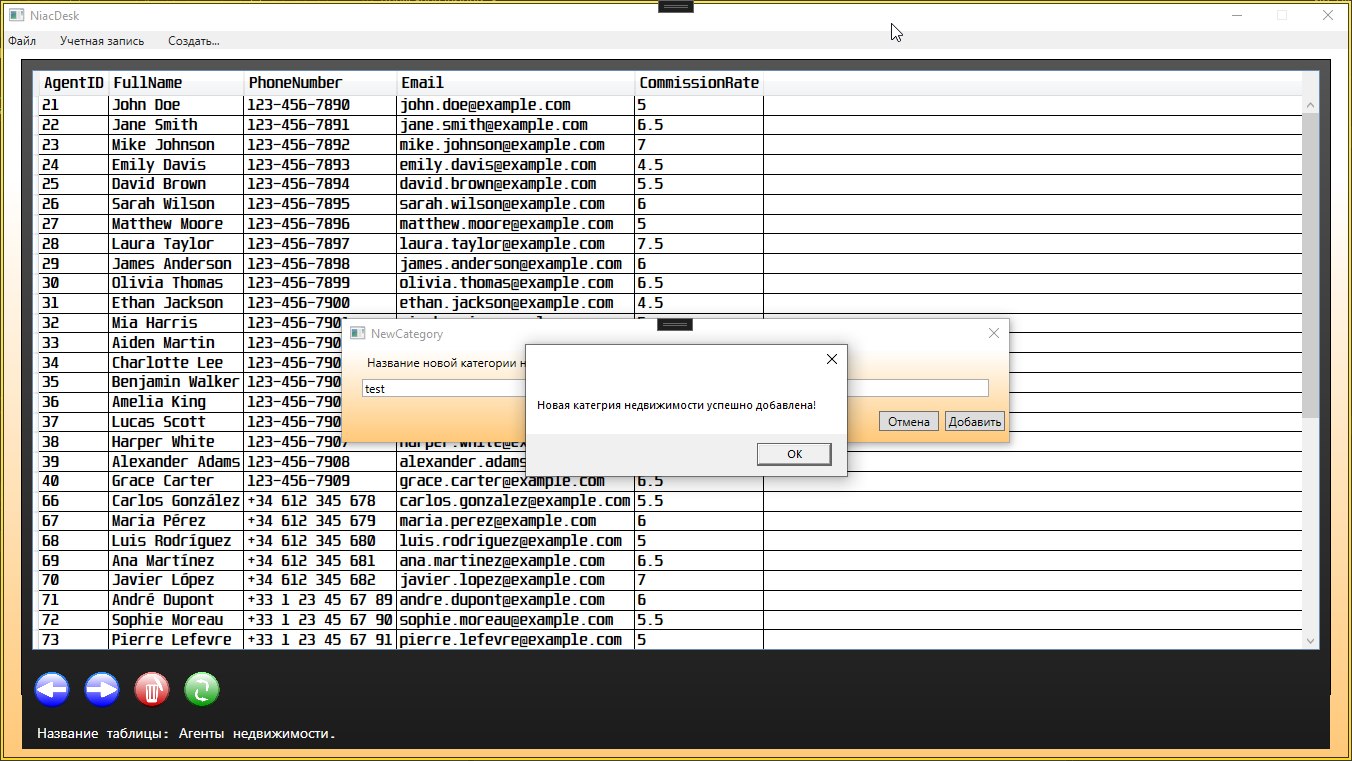


Рисунок 104. Добавление новой категории недвижимости

Добавить новой недвижимости:

Исключение:

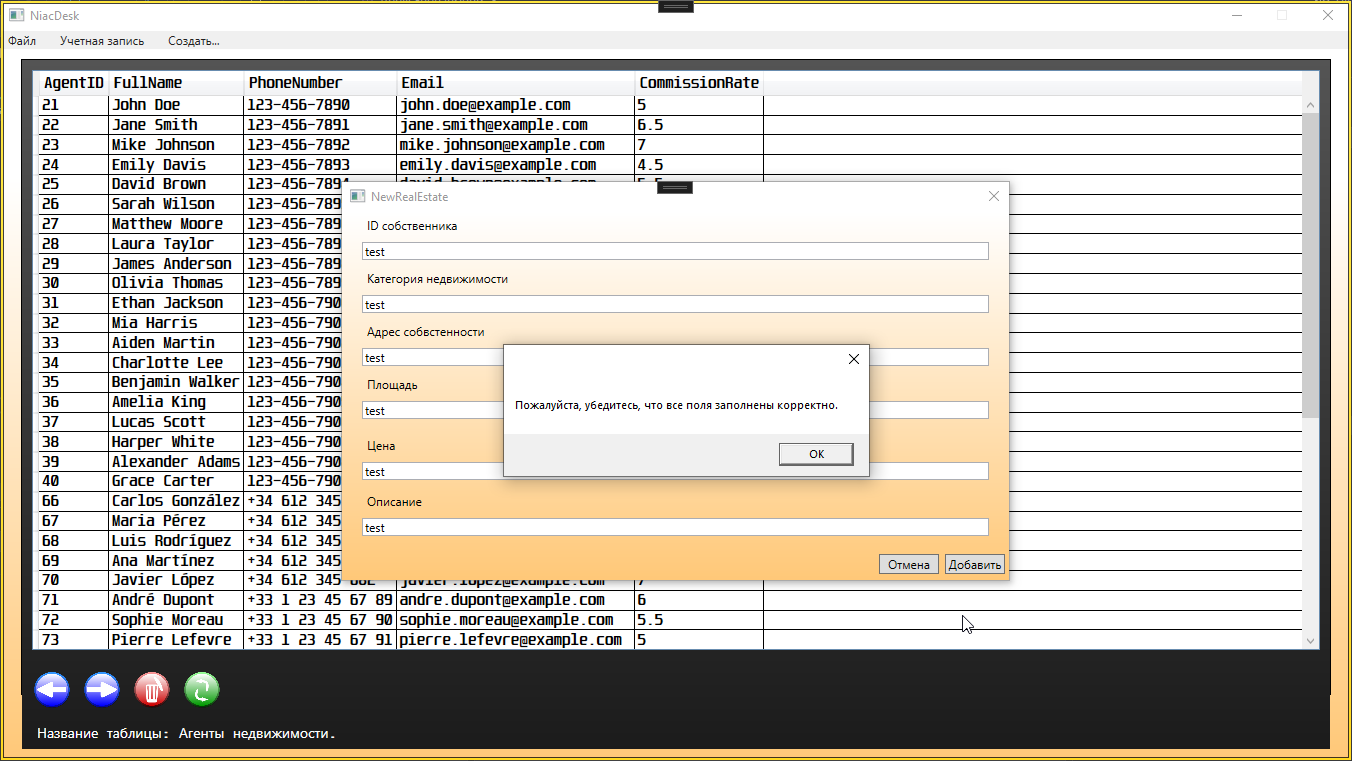


Рисунок 105. Попытка добавить новую недвижимость с неверными данными

Правильная работа:

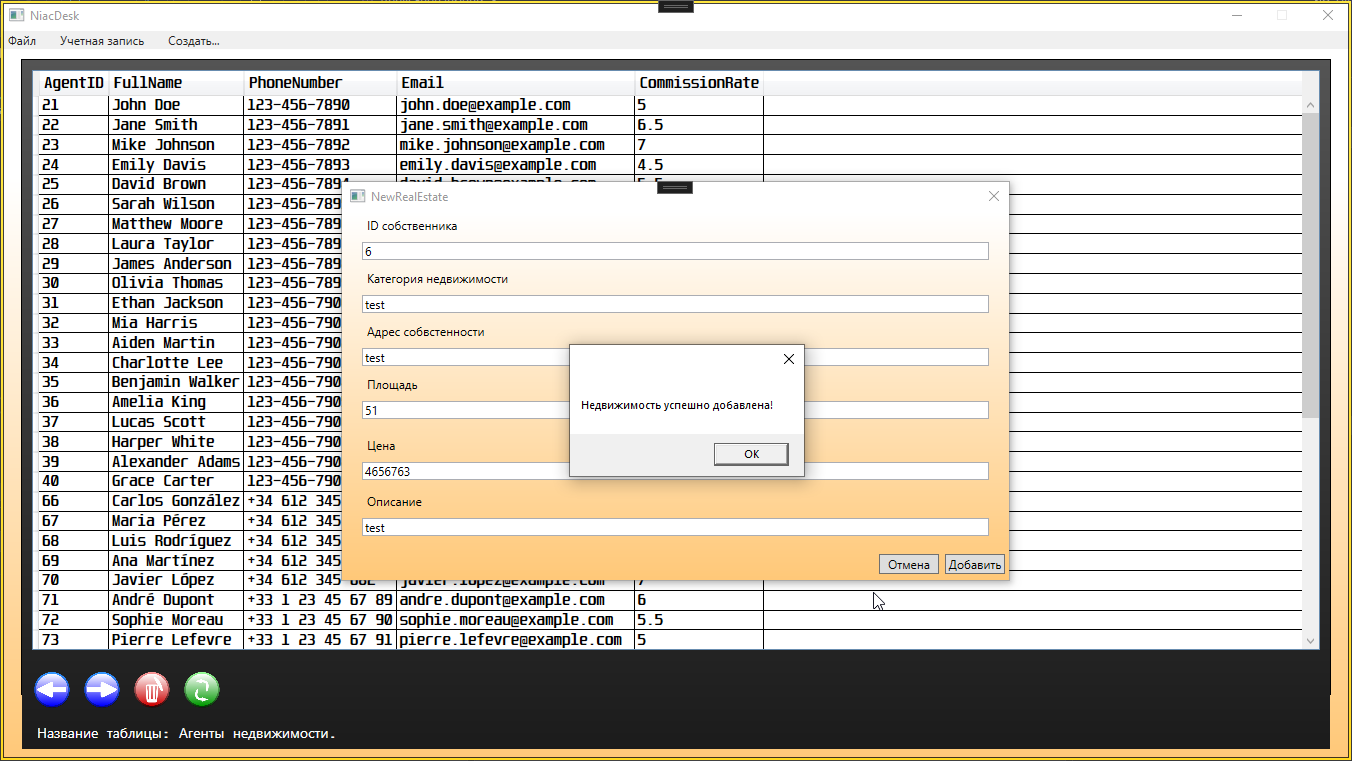


Рисунок 106. Добавление новой недвижимости

Добавить нового агента недвижимости:

Исключение:

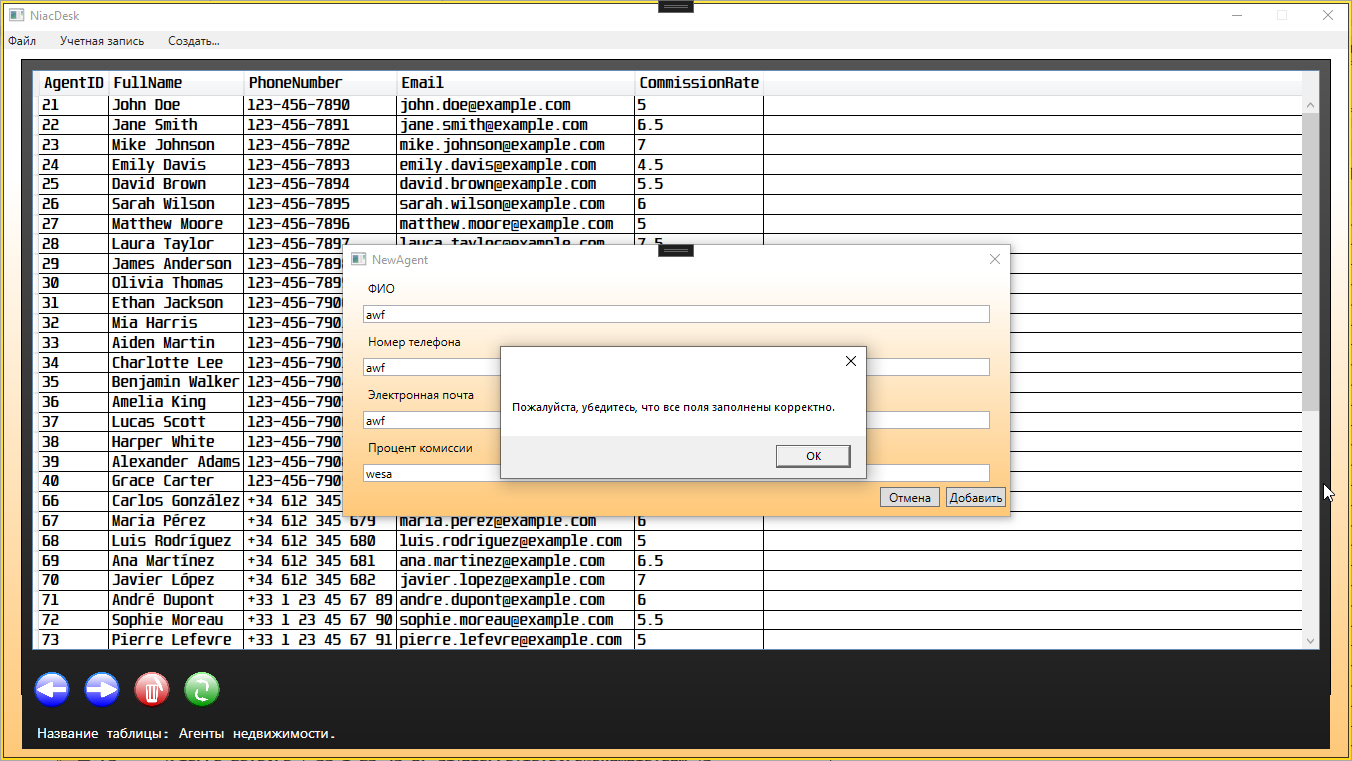


Рисунок 107. Попытка добавить нового агента с неверными данными

Правильная работа:

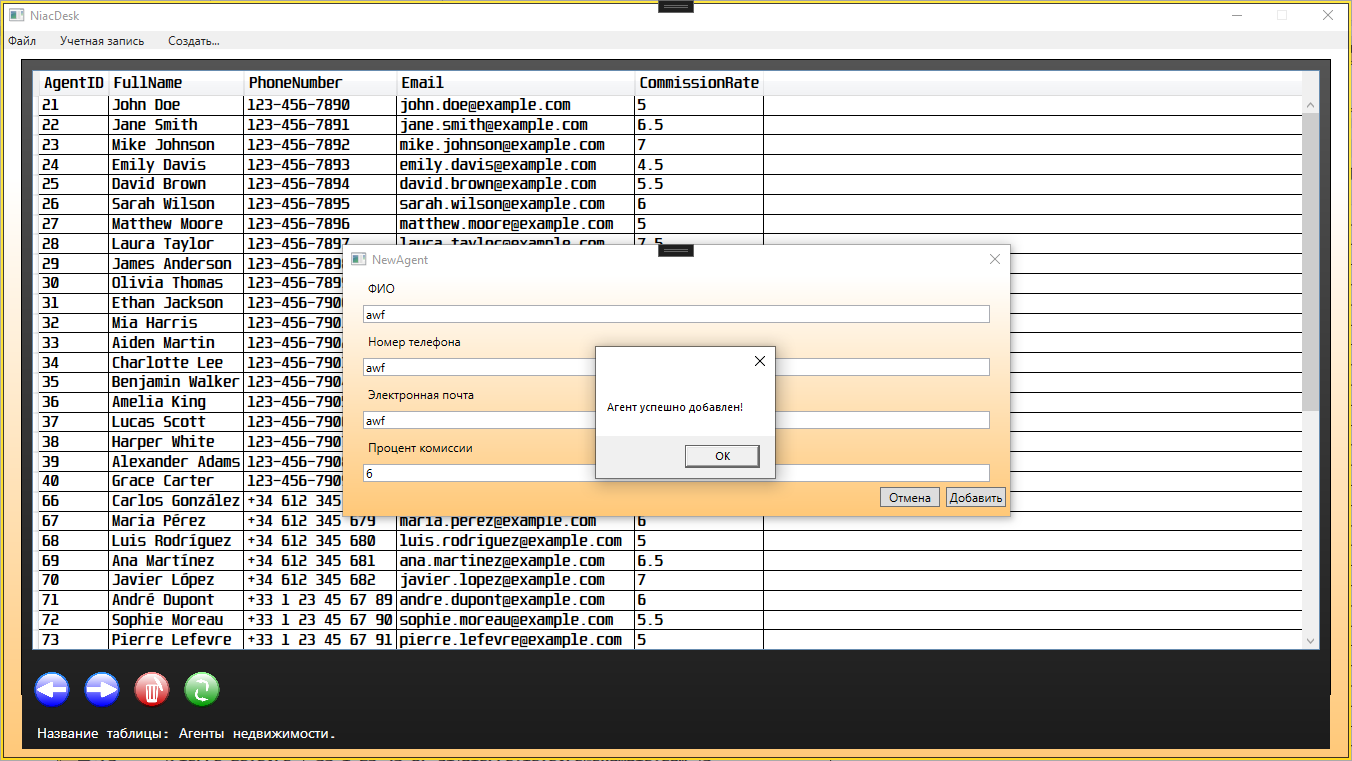


Рисунок 108. Добавление нового агента недвижимости

Добавить нового клиента:

Исключение:

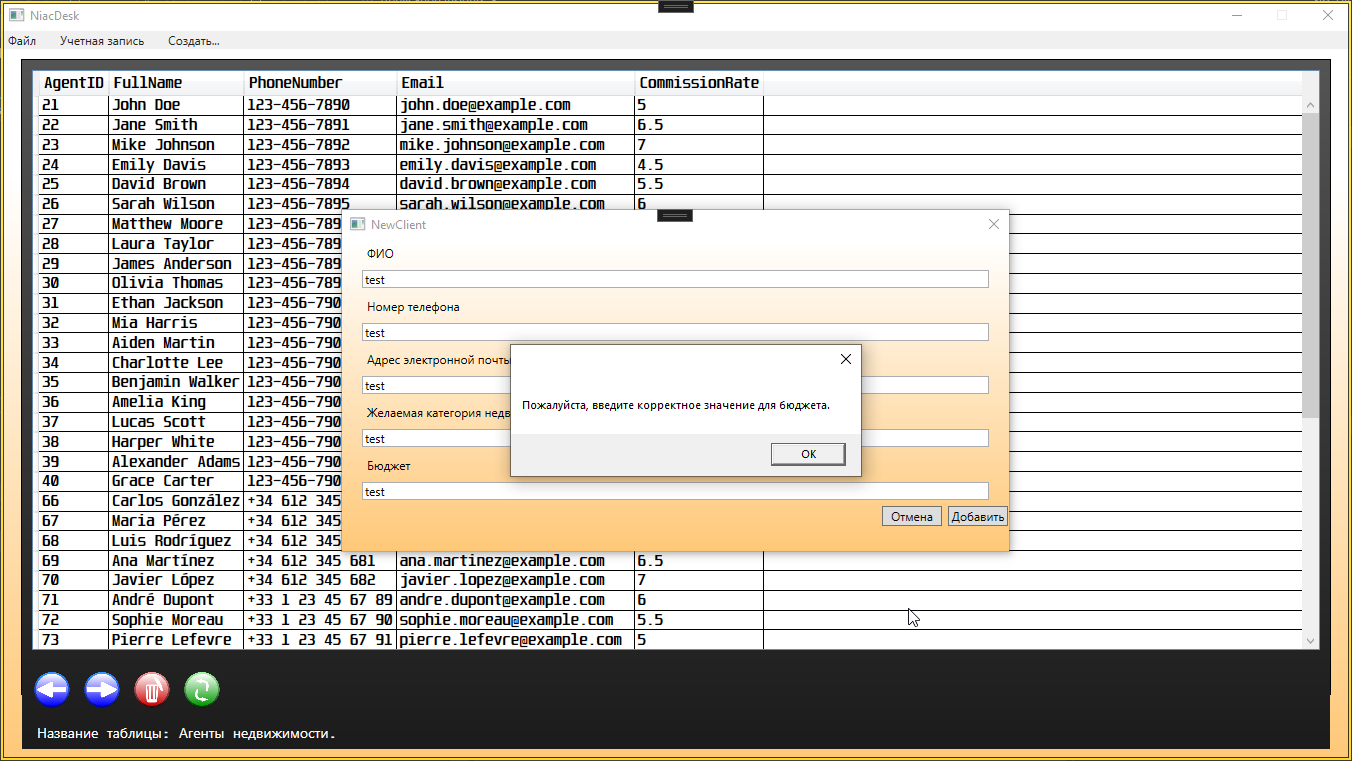


Рисунок 109. Попытка добавить нового клиента с неверными данными

Правильная работа:

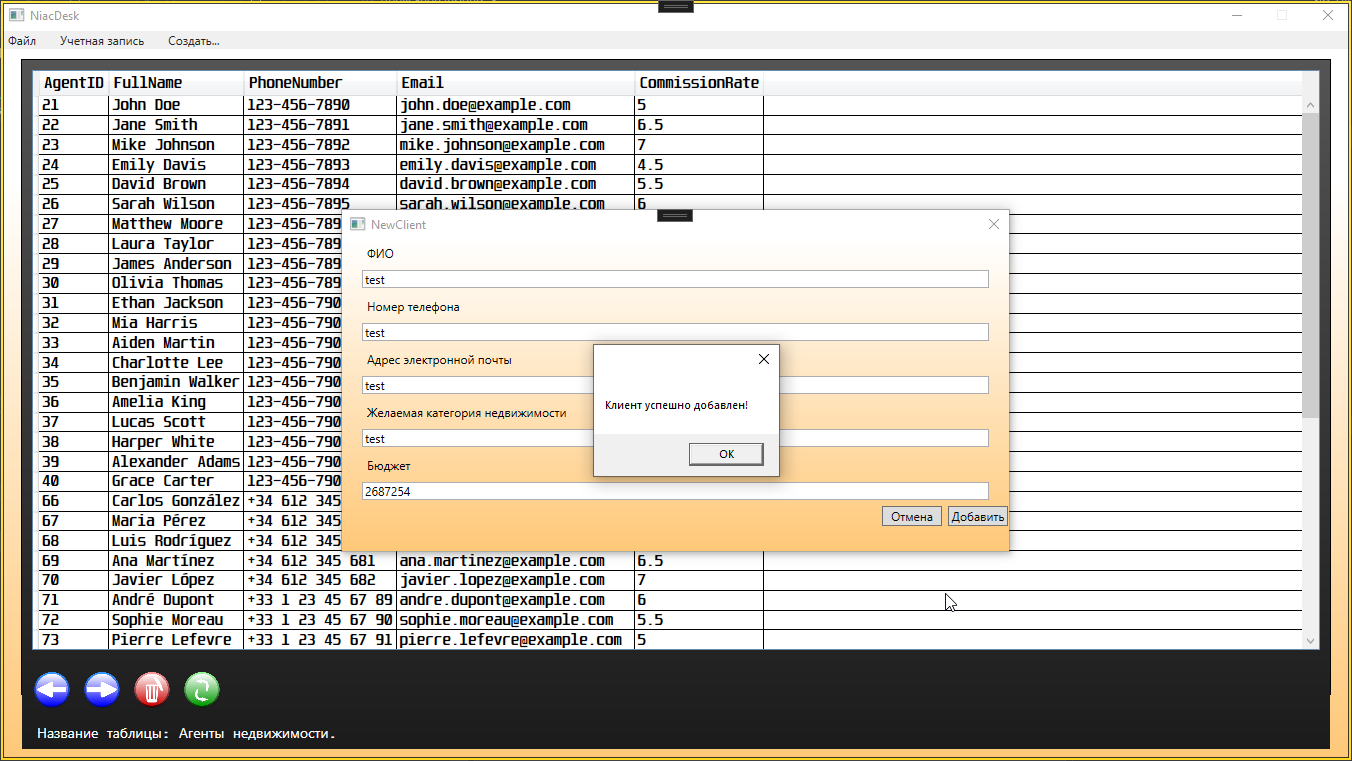


Рисунок 110. Добавление нового клиента

Добавить новую сделку:

Исключения:

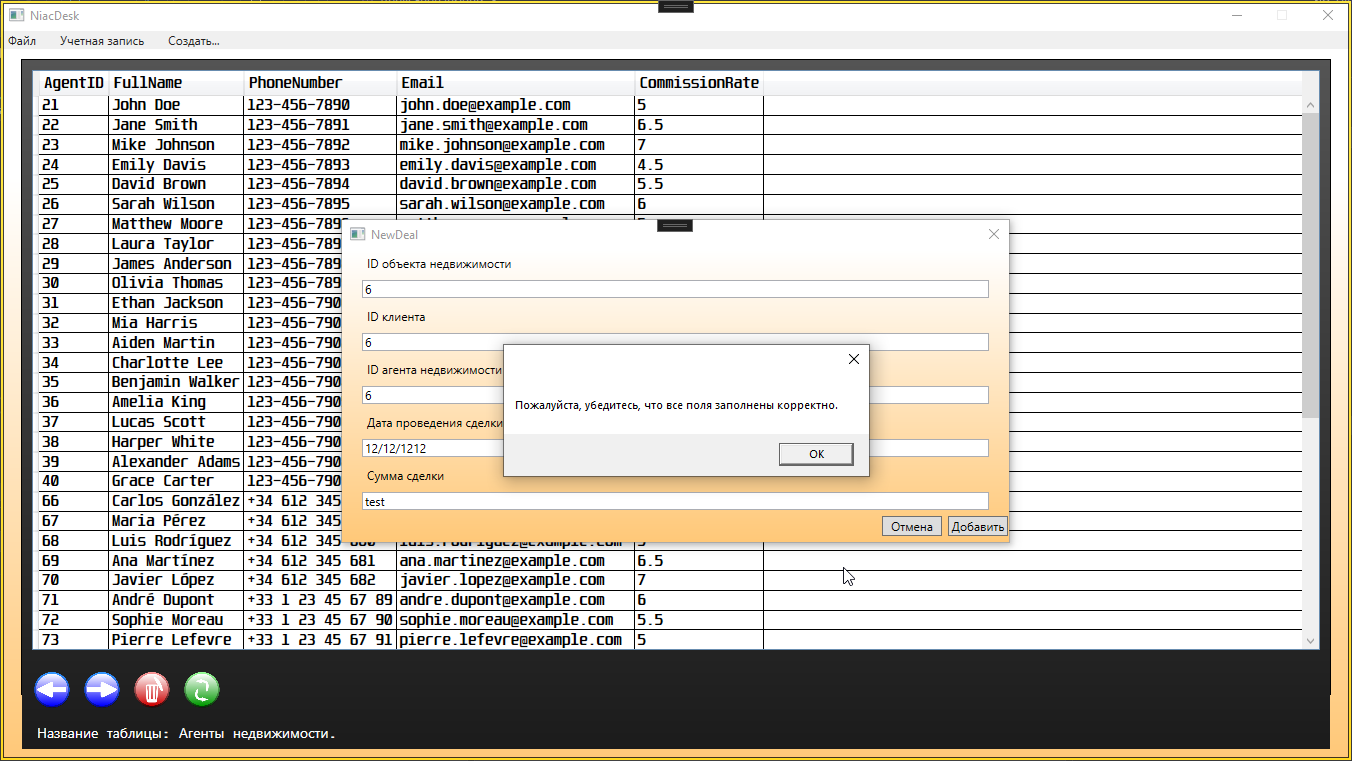


Рисунок 111. Попытка добавить новую сделку с неверными данными

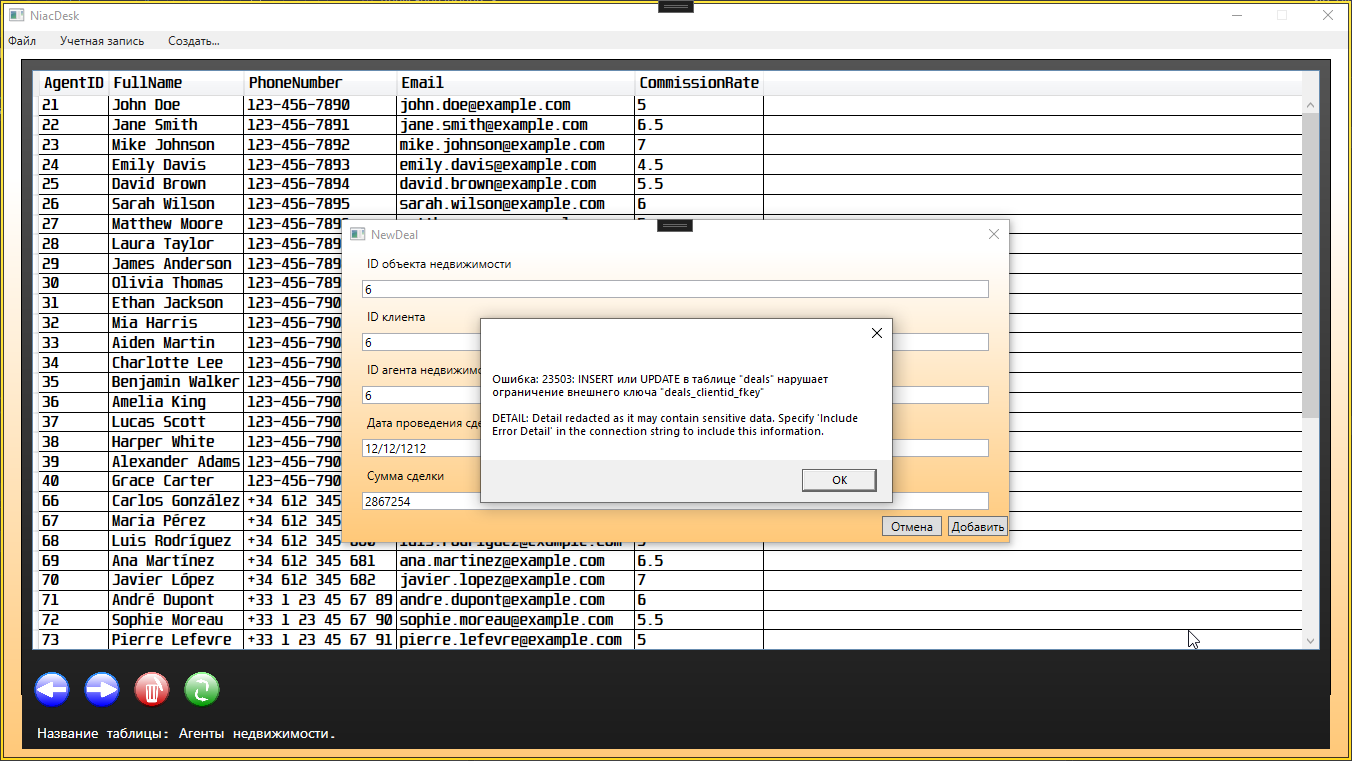


Рисунок 112. Попытка добавить новую сделку с несуществующими данными

Правильная работа:

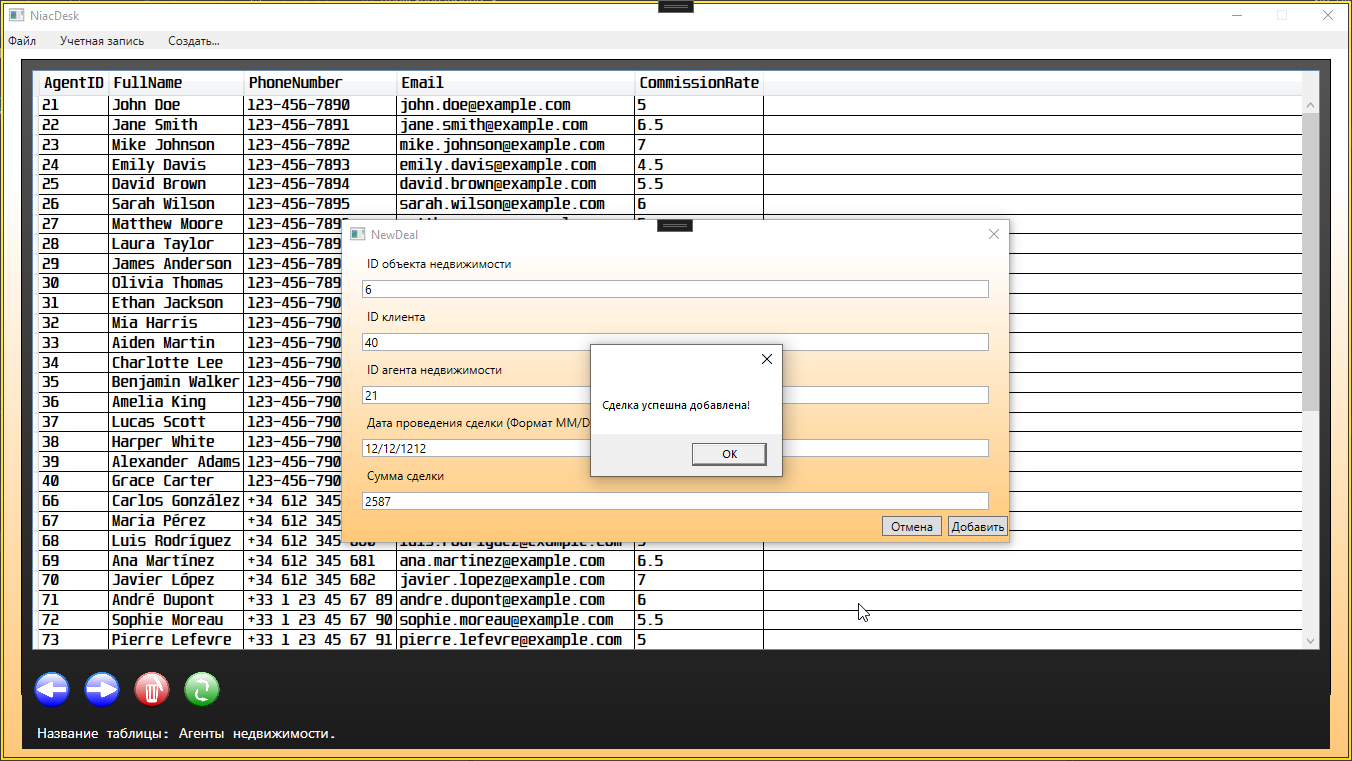


Рисунок 113. Добавление новой сделки

Добавить нового собственника:

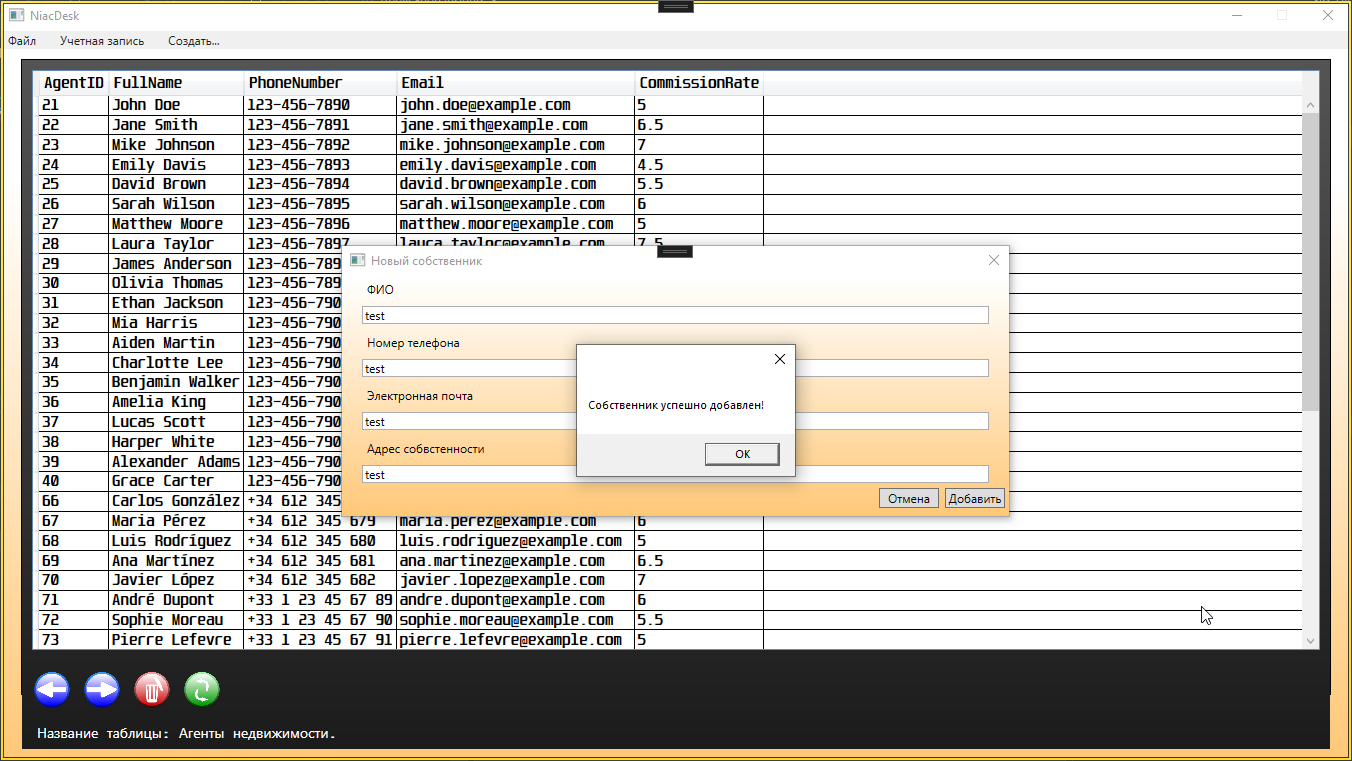


Рисунок 114. Добавление нового собственника

Проверка, чтобы все тестовые новые данные появились в DataGrid:

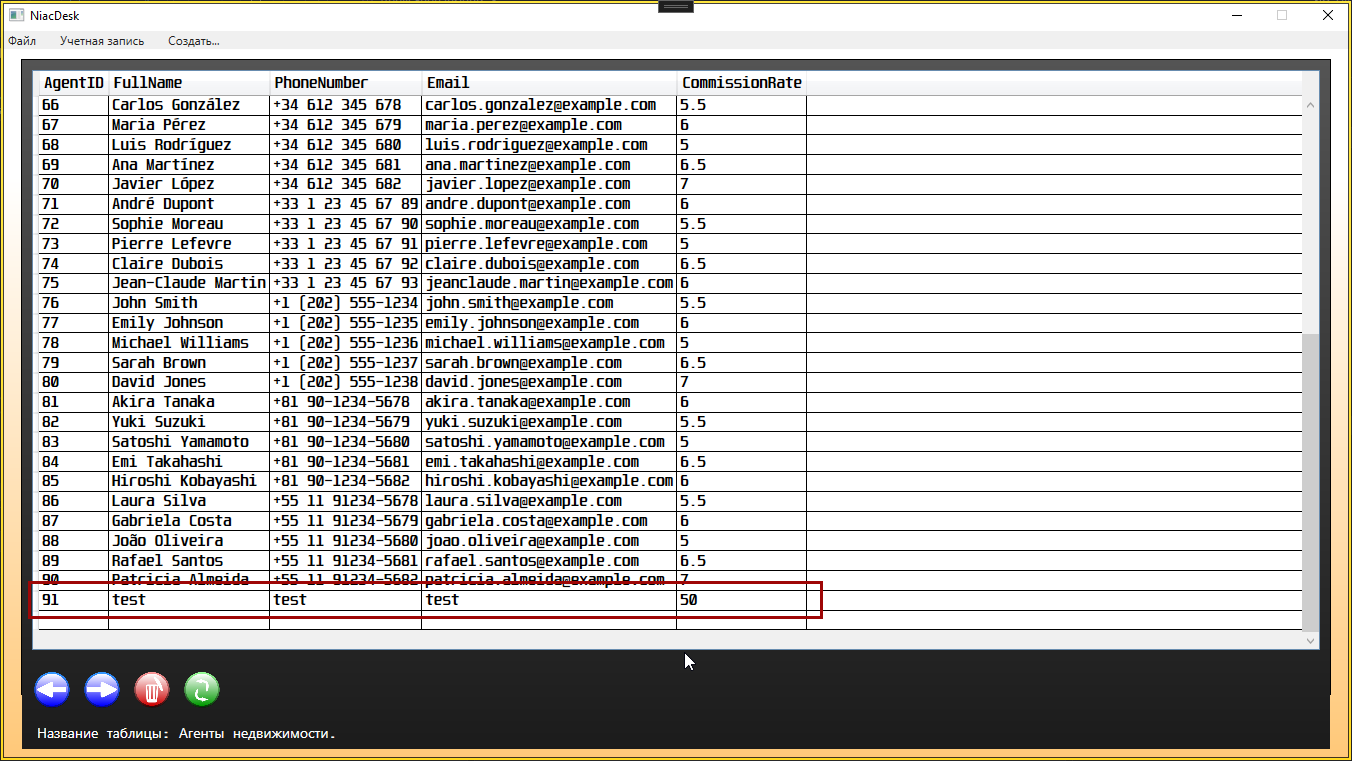


Рисунок 115. Новый агент недвижимости

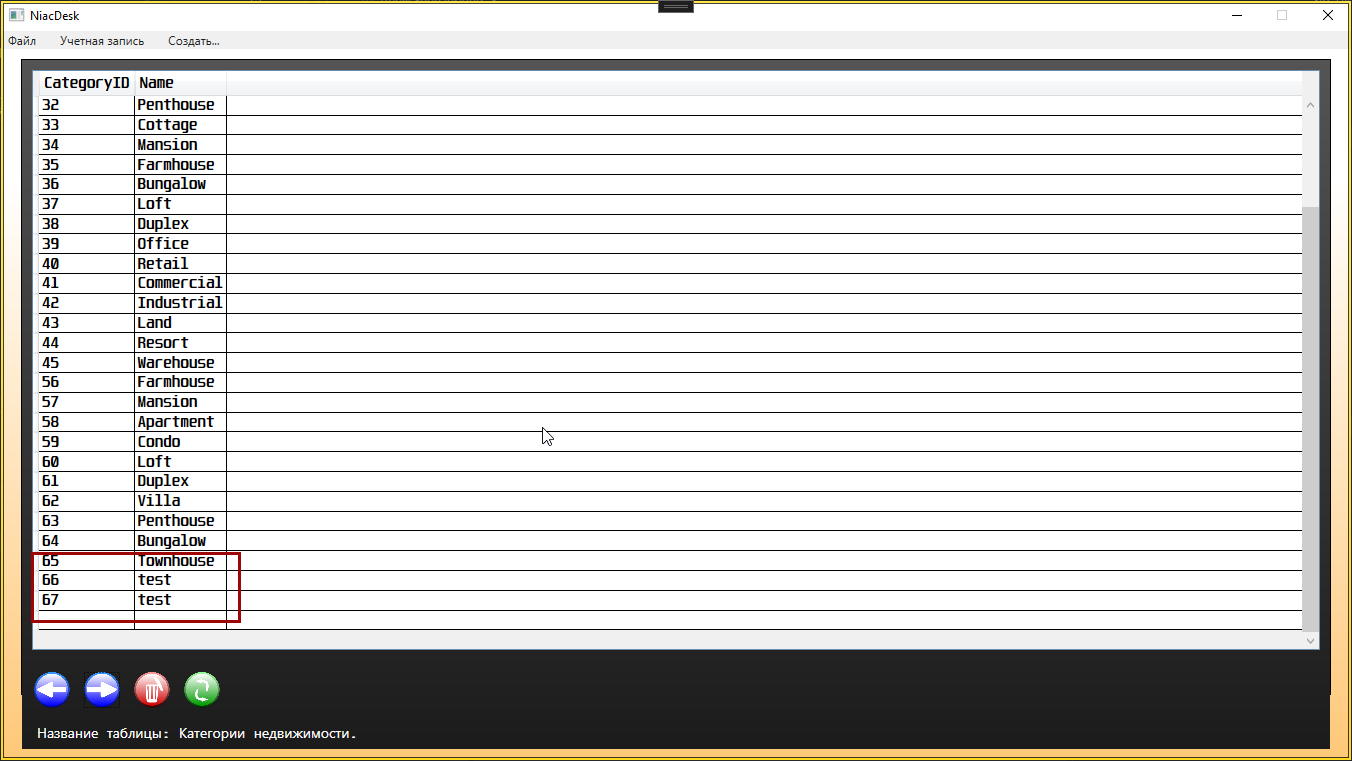


Рисунок 116. Новая категория недвижимости

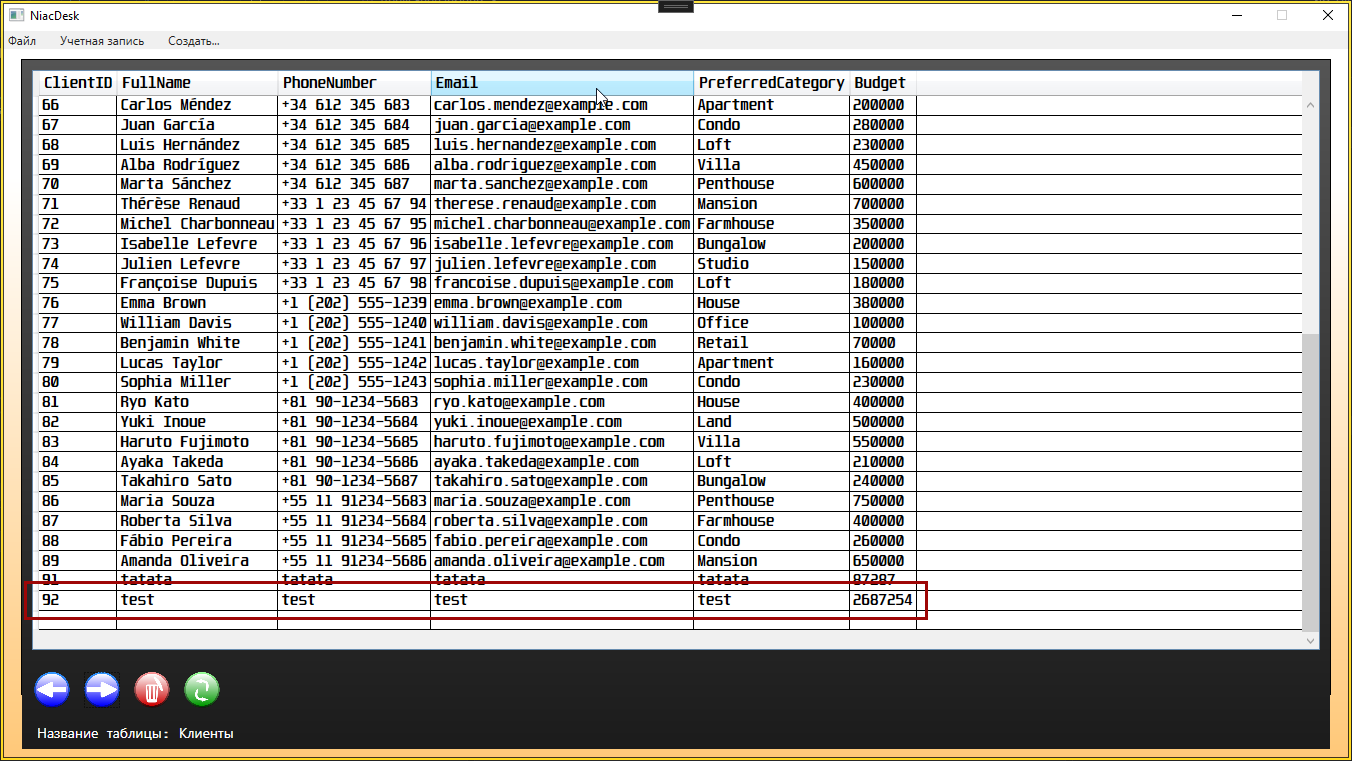


Рисунок 117. Новый клиент

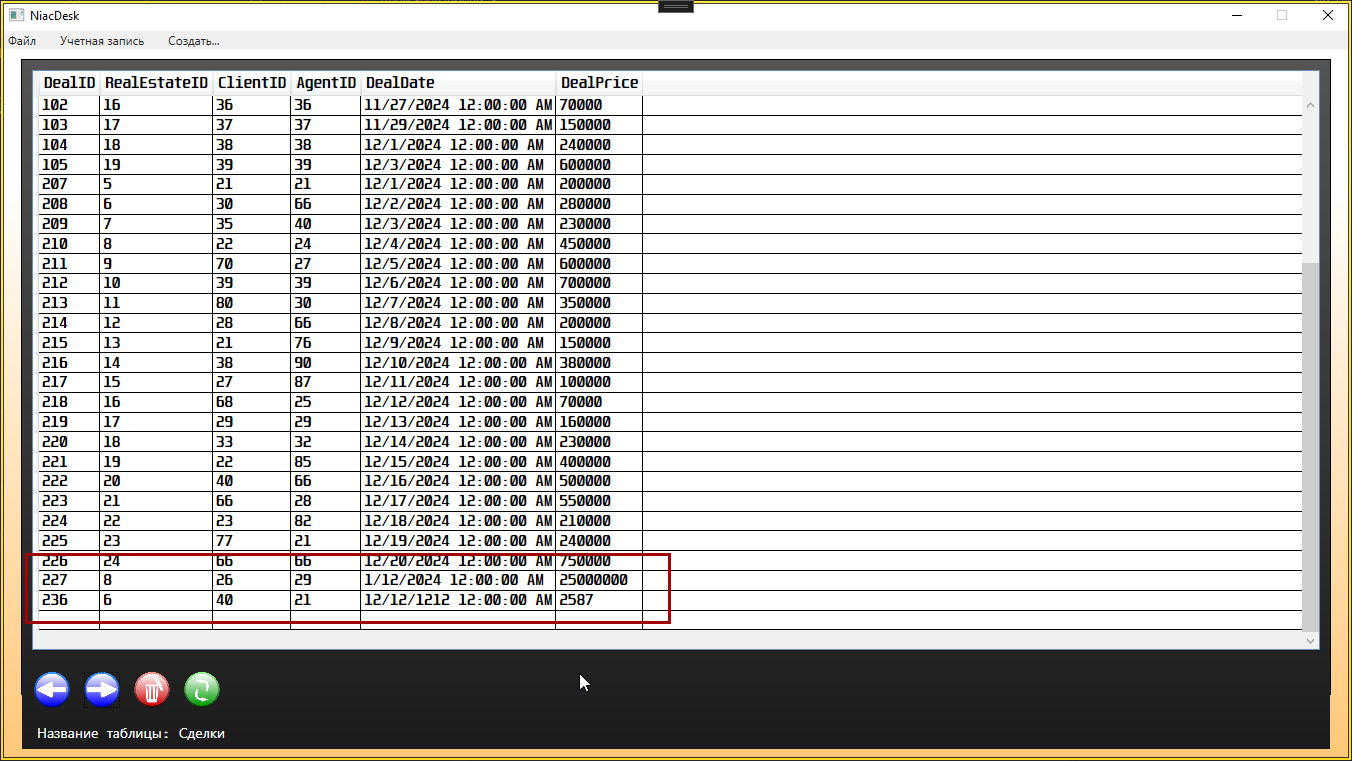


Рисунок 118. Новая сделка

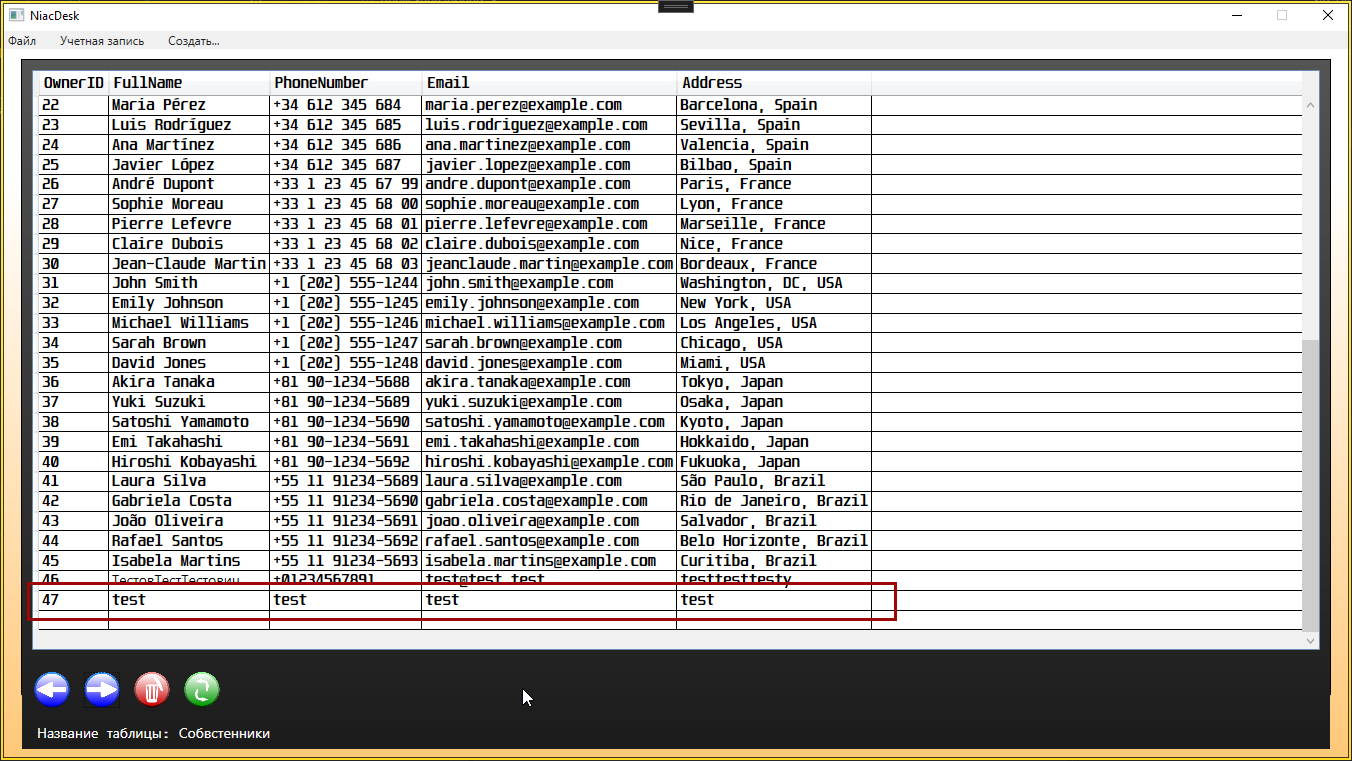


Рисунок 119. Новый собственник

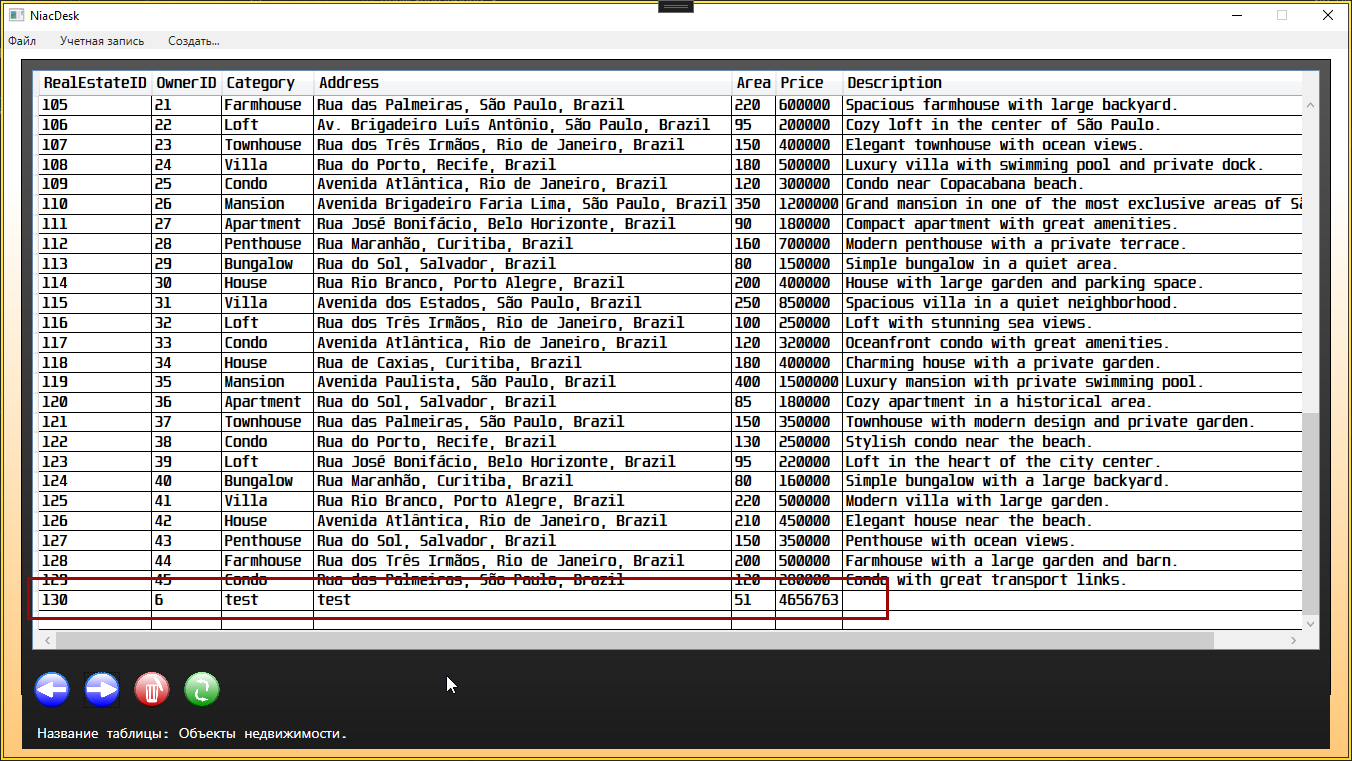


Рисунок 120. Новый объект недвижимости

Так же кнопки в подменю «Файл» - «Сохранить Таблицу» и «Сохранить базу данных»

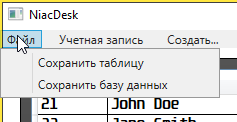


Рисунок 121. Подменю файл

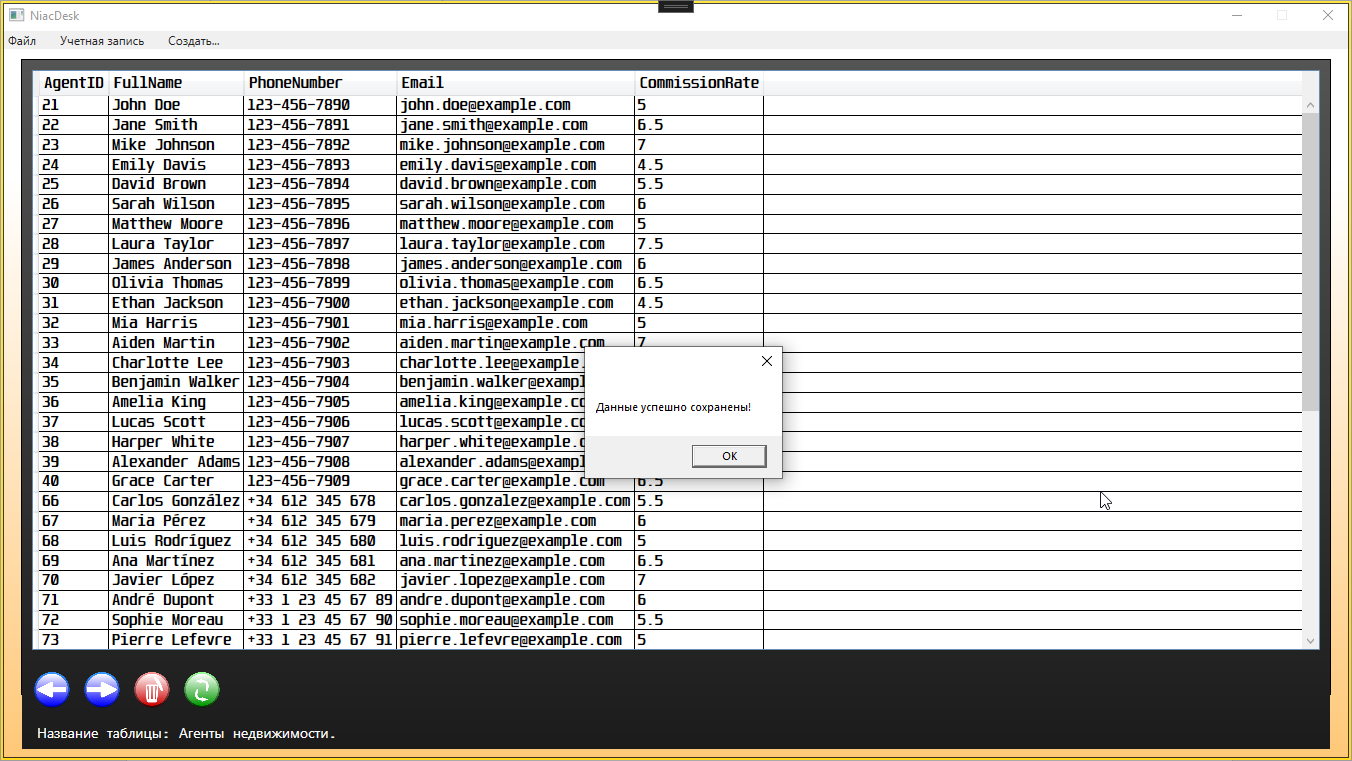


Рисунок 122. Таблица сохранена

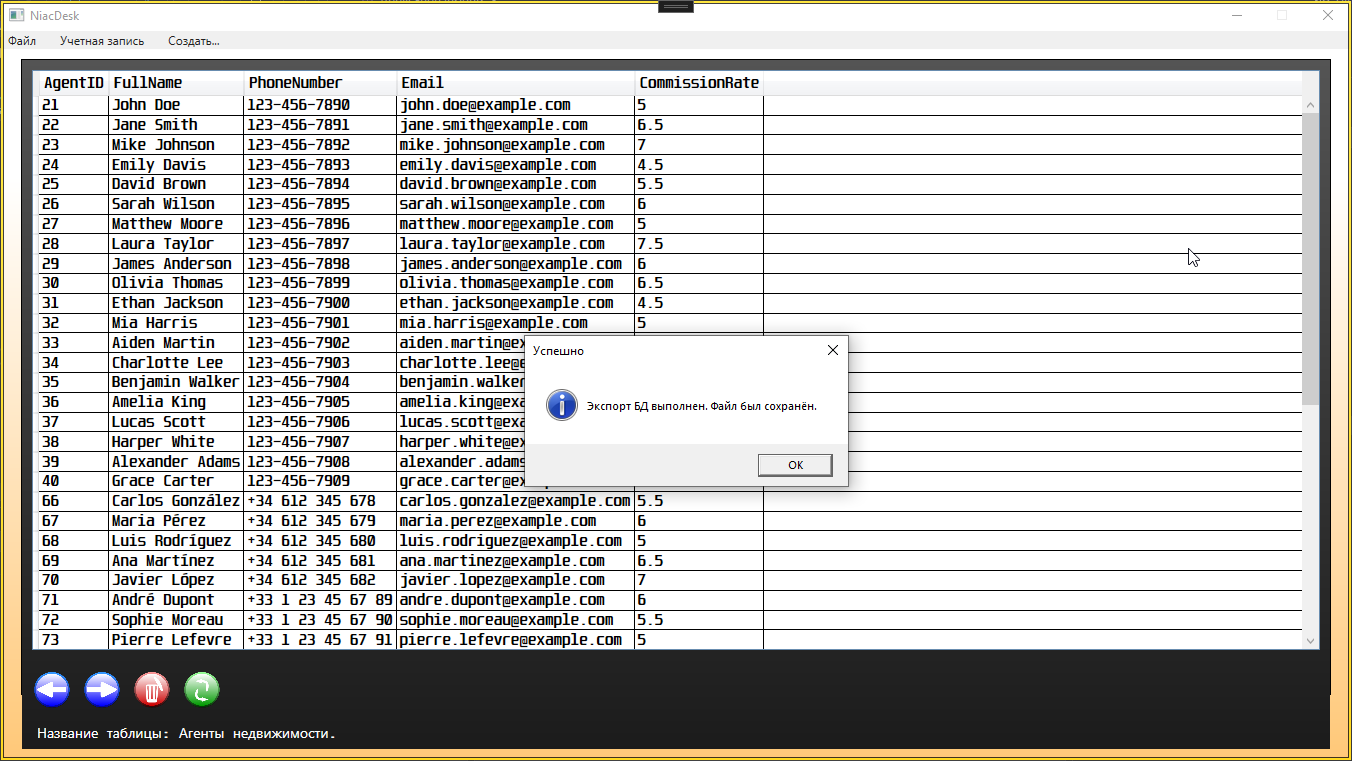


Рисунок 123. БД сохранена

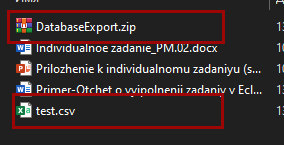


Рисунок 124. Сохранённые БД и таблица

**Код основного окна:**

using System;

using System.Windows;

using Npgsql;

using System.Collections.ObjectModel;

using Microsoft.Win32;

using System.Collections.Generic;

using System.IO;

using System.Data;

using System.IO.Compression;

namespace UchebnayaPraktika

{

public partial class MainWindow : Window

{

public ObservableCollection<Agent> Agents { get; set; }

public ObservableCollection<Category> Categories { get; set; }

public ObservableCollection<Client> Clients { get; set; }

public ObservableCollection<Deal> Deals { get; set; }

public ObservableCollection<Owner> Owners { get; set; }

public ObservableCollection<RealEstate> RealEstates { get; set; }

private int \_currentTableIndex = 0;

private readonly string[] \_tables = new string[] {

"pm02.agents",

"pm02.categories",

"pm02.clients",

"pm02.deals",

"pm02.owners",

"pm02.realestate"

};

public MainWindow()

{

InitializeComponent();

Agents = new ObservableCollection<Agent>();

Categories = new ObservableCollection<Category>();

Clients = new ObservableCollection<Client>();

Deals = new ObservableCollection<Deal>();

Owners = new ObservableCollection<Owner>();

RealEstates = new ObservableCollection<RealEstate>();

this.DataContext = this;

LoadData();

}

private void LoadData()

{

string connectionString = "Host=localhost;Port=5432;Username=postgres;Password=1234;Database=pm02";

string query = GetQueryForTable(\_tables[\_currentTableIndex]);

switch (\_currentTableIndex)

{

case 0: Agents.Clear(); break;

case 1: Categories.Clear(); break;

case 2: Clients.Clear(); break;

case 3: Deals.Clear(); break;

case 4: Owners.Clear(); break;

case 5: RealEstates.Clear(); break;

}

using (var connection = new NpgsqlConnection(connectionString))

{

connection.Open();

using (var cmd = new NpgsqlCommand(query, connection))

{

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

if (\_currentTableIndex == 0)

{

Agents.Add(new Agent

{

AgentID = reader.IsDBNull(0) ? 0 : reader.GetInt32(0),

FullName = reader.IsDBNull(1) ? string.Empty : reader.GetString(1),

PhoneNumber = reader.IsDBNull(2) ? string.Empty : reader.GetString(2),

Email = reader.IsDBNull(3) ? string.Empty : reader.GetString(3),

CommissionRate = reader.IsDBNull(4) ? 0 : reader.GetDouble(4)

});

DataGrid1.ItemsSource = Agents;

TableNameLabel.Content = "Название таблицы: Агенты недвижимости.";

}

else if (\_currentTableIndex == 1)

{

Categories.Add(new Category

{

CategoryID = reader.IsDBNull(0) ? 0 : reader.GetInt32(0),

Name = reader.IsDBNull(1) ? string.Empty : reader.GetString(1)

});

DataGrid1.ItemsSource = Categories;

TableNameLabel.Content = "Название таблицы: Категории недвижимости.";

}

else if (\_currentTableIndex == 2)

{

Clients.Add(new Client

{

ClientID = reader.IsDBNull(0) ? 0 : reader.GetInt32(0),

FullName = reader.IsDBNull(1) ? string.Empty : reader.GetString(1),

PhoneNumber = reader.IsDBNull(2) ? string.Empty : reader.GetString(2),

Email = reader.IsDBNull(3) ? string.Empty : reader.GetString(3),

PreferredCategory = reader.IsDBNull(4) ? string.Empty : reader.GetString(4),

Budget = reader.IsDBNull(5) ? 0 : reader.GetDouble(5)

});

DataGrid1.ItemsSource = Clients;

TableNameLabel.Content = "Название таблицы: Клиенты";

}

else if (\_currentTableIndex == 3)

{

Deals.Add(new Deal

{

DealID = reader.IsDBNull(0) ? 0 : reader.GetInt32(0),

RealEstateID = reader.IsDBNull(1) ? 0 : reader.GetInt32(1),

ClientID = reader.IsDBNull(2) ? 0 : reader.GetInt32(2),

AgentID = reader.IsDBNull(3) ? 0 : reader.GetInt32(3),

DealDate = reader.IsDBNull(4) ? DateTime.MinValue : reader.GetDateTime(4),

DealPrice = reader.IsDBNull(5) ? 0 : reader.GetDouble(5)

});

DataGrid1.ItemsSource = Deals;

TableNameLabel.Content = "Название таблицы: Сделки";

}

else if (\_currentTableIndex == 4)

{

Owners.Add(new Owner

{

OwnerID = reader.IsDBNull(0) ? 0 : reader.GetInt32(0),

FullName = reader.IsDBNull(1) ? string.Empty : reader.GetString(1),

PhoneNumber = reader.IsDBNull(2) ? string.Empty : reader.GetString(2),

Email = reader.IsDBNull(3) ? string.Empty : reader.GetString(3),

Address = reader.IsDBNull(4) ? string.Empty : reader.GetString(4)

});

DataGrid1.ItemsSource = Owners;

TableNameLabel.Content = "Название таблицы: Собвстенники";

}

else if (\_currentTableIndex == 5)

{

RealEstates.Add(new RealEstate

{

RealEstateID = reader.IsDBNull(0) ? 0 : reader.GetInt32(0),

OwnerID = reader.IsDBNull(1) ? 0 : reader.GetInt32(1),

Category = reader.IsDBNull(2) ? string.Empty : reader.GetString(2),

Address = reader.IsDBNull(3) ? string.Empty : reader.GetString(3),

Area = reader.IsDBNull(4) ? 0 : reader.GetDouble(4),

Price = reader.IsDBNull(5) ? 0 : reader.GetDouble(5),

Description = reader.IsDBNull(6) ? string.Empty : reader.GetString(6)

});

DataGrid1.ItemsSource = RealEstates;

TableNameLabel.Content = "Название таблицы: Объекты недвижимости.";

}

}

}

}

}

DataGrid1.Items.Refresh();

}

private string GetQueryForTable(string table)

{

if (table == "pm02.agents")

{

return "SELECT AgentID, FullName, PhoneNumber, Email, CommissionRate FROM pm02.agents";

}

else if (table == "pm02.categories")

{

return "SELECT CategoryID, Name FROM pm02.categories";

}

else if (table == "pm02.clients")

{

return "SELECT ClientID, FullName, PhoneNumber, Email, PreferredCategory, Budget FROM pm02.clients";

}

else if (table == "pm02.deals")

{

return "SELECT DealID, RealEstateID, ClientID, AgentID, DealDate, DealPrice FROM pm02.deals";

}

else if (table == "pm02.owners")

{

return "SELECT OwnerID, FullName, PhoneNumber, Email, Address FROM pm02.owners";

}

else if (table == "pm02.realestate")

{

return "SELECT RealEstateID, OwnerID, Category, Address, Area, Price, Description FROM pm02.realestate";

}

else

{

throw new ArgumentException("Неизвестная таблица");

}

}

private void PreviousButton\_Click(object sender, RoutedEventArgs e)

{

\_currentTableIndex = (\_currentTableIndex - 1 + \_tables.Length) % \_tables.Length;

LoadData();

DataGrid1.Items.Refresh();

}

private void NextButton\_Click(object sender, RoutedEventArgs e)

{

\_currentTableIndex = (\_currentTableIndex + 1) % \_tables.Length;

LoadData();

DataGrid1.Items.Refresh();

}

private void DeleteLineButton\_Click(object sender, RoutedEventArgs e)

{

SecondWindow secondWindow = new SecondWindow();

secondWindow.DeleteLineEvent += DeleteLine;

secondWindow.ShowDialog();

DataGrid1.Items.Refresh();

}

private void DeleteLine(int id)

{

string columnName = GetColumnNameForCurrentTable();

string tableName = GetTableNameForCurrentIndex();

DeleteFromDatabase(columnName, id, tableName);

DataGrid1.Items.Refresh();

}

private string GetColumnNameForCurrentTable()

{

switch (\_currentTableIndex)

{

case 0: return "AgentID";

case 1: return "CategoryID";

case 2: return "ClientID";

case 3: return "DealID";

case 4: return "OwnerID";

case 5: return "RealEstateID";

default: return null;

}

}

private string GetTableNameForCurrentIndex()

{

switch (\_currentTableIndex)

{

case 0: return "Agents";

case 1: return "Categories";

case 2: return "Clients";

case 3: return "Deals";

case 4: return "Owners";

case 5: return "RealEstates";

default: return null;

}

}

private void DeleteFromDatabase(string columnName, int id, string tableName)

{

string connectionString = "Host=localhost;Port=5432;Username=postgres;Password=1234;Database=pm02";;

using (var connection = new NpgsqlConnection(connectionString))

{

connection.Open();

string deleteQuery = $"DELETE FROM pm02.{tableName} WHERE {columnName} = @id";

using (var command = new NpgsqlCommand(deleteQuery, connection))

{

command.Parameters.AddWithValue("@id", id);

try

{

command.ExecuteNonQuery();

MessageBox.Show("Запись успешно удалена!");

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка удаления записи: {ex.Message}");

}

}

}

DataGrid1.Items.Refresh();

}

private void MenuItemNewCategory\_Click(object sender, RoutedEventArgs e)

{

NewCategory newCategory = new NewCategory();

newCategory.ShowDialog();

}

private void MenuItemNewRealEstate\_Click(object sender, RoutedEventArgs e)

{

NewRealEstate newRealEstate = new NewRealEstate();

newRealEstate.ShowDialog();

}

private void MenuItemNewAgent\_Click(object sender, RoutedEventArgs e)

{

NewAgent newAgent = new NewAgent();

newAgent.ShowDialog();

}

private void MenuItemNewClient\_Click(object sender, RoutedEventArgs e)

{

NewClient newClient = new NewClient();

newClient.ShowDialog();

}

private void MenuItemNewDeal\_Click(object sender, RoutedEventArgs e)

{

NewDeal newDeal = new NewDeal();

newDeal.ShowDialog();

}

private void MenuItemNewOwner\_Click(object sender, RoutedEventArgs e)

{

NewOwner newOwner = new NewOwner();

newOwner.ShowDialog();

}

private void MenuItemSaveTable\_Click(object sender, RoutedEventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "CSV files (\*.csv)|\*.csv|All files (\*.\*)|\*.\*";

if (saveFileDialog.ShowDialog() == true)

{

string filePath = saveFileDialog.FileName;

try

{

SaveTableToCSV(filePath, \_currentTableIndex);

MessageBox.Show("Данные успешно сохранены!");

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении файла: {ex.Message}");

}

}

}

private void SaveTableToCSV(string filePath, int tableIndex)

{

var lines = new List<string>();

switch (tableIndex)

{

case 0:

lines.Add("AgentID,FullName,PhoneNumber,Email,CommissionRate");

foreach (var agent in Agents)

{

lines.Add($"{agent.AgentID},{agent.FullName},{agent.PhoneNumber},{agent.Email},{agent.CommissionRate}");

}

break;

case 1:

lines.Add("CategoryID,Name");

foreach (var category in Categories)

{

lines.Add($"{category.CategoryID},{category.Name}");

}

break;

case 2:

lines.Add("ClientID,FullName,PhoneNumber,Email,PreferredCategory,Budget");

foreach (var client in Clients)

{

lines.Add($"{client.ClientID},{client.FullName},{client.PhoneNumber},{client.Email},{client.PreferredCategory},{client.Budget}");

}

break;

case 3:

lines.Add("DealID,RealEstateID,ClientID,AgentID,DealDate,DealPrice");

foreach (var deal in Deals)

{

lines.Add($"{deal.DealID},{deal.RealEstateID},{deal.ClientID},{deal.AgentID},{deal.DealDate:yyyy-MM-dd},{deal.DealPrice}");

}

break;

case 4:

lines.Add("OwnerID,FullName,PhoneNumber,Email,Address");

foreach (var owner in Owners)

{

lines.Add($"{owner.OwnerID},{owner.FullName},{owner.PhoneNumber},{owner.Email},{owner.Address}");

}

break;

case 5:

lines.Add("RealEstateID,OwnerID,Category,Address,Area,Price,Description");

foreach (var realEstate in RealEstates)

{

lines.Add($"{realEstate.RealEstateID},{realEstate.OwnerID},{realEstate.Category},{realEstate.Address},{realEstate.Area},{realEstate.Price},{realEstate.Description}");

}

break;

default:

throw new Exception("Неизвестная таблица.");

}

File.WriteAllLines(filePath, lines);

}

private void MenuItemDB\_Click(object sender, RoutedEventArgs e)

{

try

{

string connectionString = "Host=localhost;Port=5432;Username=postgres;Password=1234;Database=pm02";

string tempFolder = Path.Combine(Path.GetTempPath(), "DatabaseExport");

if (Directory.Exists(tempFolder))

Directory.Delete(tempFolder, true);

Directory.CreateDirectory(tempFolder);

using (var connection = new NpgsqlConnection(connectionString))

{

connection.Open();

DataTable tables = connection.GetSchema("Tables", new string[] { null, "pm02" });

foreach (DataRow row in tables.Rows)

{

string tableName = row["table\_name"].ToString();

string csvFilePath = Path.Combine(tempFolder, $"{tableName}.csv");

using (var cmd = new NpgsqlCommand($"SELECT \* FROM pm02.{tableName}", connection))

using (var reader = cmd.ExecuteReader())

using (var writer = new StreamWriter(csvFilePath))

{

for (int i = 0; i < reader.FieldCount; i++)

{

if (i > 0) writer.Write(",");

writer.Write(reader.GetName(i));

}

writer.WriteLine();

while (reader.Read())

{

for (int i = 0; i < reader.FieldCount; i++)

{

if (i > 0) writer.Write(",");

writer.Write(reader.GetValue(i).ToString());

}

writer.WriteLine();

}

}

}

}

string zipFilePath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), "DatabaseExport.zip");

if (File.Exists(zipFilePath))

File.Delete(zipFilePath);

ZipFile.CreateFromDirectory(tempFolder, zipFilePath);

MessageBox.Show("Экспорт БД выполнен. Файл был сохранён.", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

Directory.Delete(tempFolder, true);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка: " + ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void MenuItemUserInfo\_Click(object sender, RoutedEventArgs e)

{

}

private void MenuItemChangeUser\_Click(object sender, RoutedEventArgs e)

{

}

private void MenuItemExit\_Click(object sender, RoutedEventArgs e)

{

}

private void UpdateButton\_Click(object sender, RoutedEventArgs e)

{

LoadData();

DataGrid1.Items.Refresh();

}

}

public class Agent

{

public int AgentID { get; set; }

public string FullName { get; set; }

public string PhoneNumber { get; set; }

public string Email { get; set; }

public double CommissionRate { get; set; }

}

public class Category

{

public int CategoryID { get; set; }

public string Name { get; set; }

}

public class Client

{

public int ClientID { get; set; }

public string FullName { get; set; }

public string PhoneNumber { get; set; }

public string Email { get; set; }

public string PreferredCategory { get; set; }

public double Budget { get; set; }

}

public class Deal

{

public int DealID { get; set; }

public int RealEstateID { get; set; }

public int ClientID { get; set; }

public int AgentID { get; set; }

public DateTime DealDate { get; set; }

public double DealPrice { get; set; }

}

public class Owner

{

public int OwnerID { get; set; }

public string FullName { get; set; }

public string PhoneNumber { get; set; }

public string Email { get; set; }

public string Address { get; set; }

}

public class RealEstate

{

public int RealEstateID { get; set; }

public int OwnerID { get; set; }

public string Category { get; set; }

public string Address { get; set; }

public double Area { get; set; }

public double Price { get; set; }

public string Description { get; set; }

}

}

**XAML код основного окна:**

<Window x:Class="UchebnayaPraktika.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:UchebnayaPraktika"

mc:Ignorable="d"

Title="NiacDesk" Height="768" Width="1366" ResizeMode="CanMinimize" WindowStartupLocation="CenterScreen" FontFamily="Unispace" FontSize="14">

<Window.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="White" Offset="0.2"/>

<GradientStop Color="#FFFFC878" Offset="1"/>

</LinearGradientBrush>

</Window.Background>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Menu Grid.Row="0">

<MenuItem Header="Файл">

<MenuItem Header="Сохранить таблицу" Click="MenuItemSaveTable\_Click"/>

<MenuItem Header="Сохранить базу данных" Click="MenuItemDB\_Click"/>

</MenuItem>

<MenuItem Header="Учетная запись" Margin="10,0,0,0">

<MenuItem Header="О пользователе..." Click="MenuItemUserInfo\_Click"/>

<MenuItem Header="Сменить пользователя" Click="MenuItemChangeUser\_Click"/>

<Separator />

<MenuItem Header="Выход" Click="MenuItemExit\_Click"/>

</MenuItem>

<MenuItem Header="Создать..." Margin="10,0,0,0">

<MenuItem Header="Новая запись - Новый тип недвижимости" Click="MenuItemNewCategory\_Click"/>

<MenuItem Header="Новая запись - Новая недвижимость" Click="MenuItemNewRealEstate\_Click"/>

<MenuItem Header="Новая запись - Новый агент недвижимости" Click="MenuItemNewAgent\_Click"/>

<MenuItem Header="Новая запись - Новый клиент" Click="MenuItemNewClient\_Click"/>

<MenuItem Header="Новая запись - Новая сделка" Click="MenuItemNewDeal\_Click"/>

<MenuItem Header="Новая запись - Новый собственник недвижимости" Click="MenuItemNewOwner\_Click"/>

</MenuItem>

</Menu>

<Border Grid.Row="1" Margin="20,10,20,65" BorderBrush="Black" BorderThickness="1">

<StackPanel Margin="0,0,0,-55">

<StackPanel.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF545454"/>

<GradientStop Color="#FF1B1B1B" Offset="1"/>

</LinearGradientBrush>

</StackPanel.Background>

<DataGrid x:Name="DataGrid1" AutoGenerateColumns="True" Height="580" Margin="10,10,10,10" ItemsSource="{Binding Agents}" ColumnWidth="auto" />

<StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="0,10">

<Button x:Name="PreviousTableButton" Width="40" Height="40" Margin="10,0,5,0" Click="PreviousButton\_Click" Background="#00DDDDDD" BorderBrush="#00707070">

<Image Source="/PREVIOUS.png" Stretch="Uniform" OpacityMask="Black" />

</Button>

<Button x:Name="NextTableButton" Width="40" Height="40" Margin="5,0,5,0" Click="NextButton\_Click" OpacityMask="Black" Background="#00DDDDDD" BorderBrush="#00707070">

<Image Source="/NEXT.png" Stretch="Uniform" OpacityMask="Black" />

</Button>

<Button x:Name="DeleteLineButton" Width="40" Height="40" OpacityMask="Black" Background="#00DDDDDD" BorderBrush="#00707070" Click="DeleteLineButton\_Click" Margin="5,0,5,0">

<Image Source="/DELETE.png" Stretch="Uniform" OpacityMask="Black" />

</Button>

<Button x:Name="UpdateButton" Width="40" Height="40" OpacityMask="Black" Background="#00DDDDDD" BorderBrush="#00707070" Click="UpdateButton\_Click" Margin="5,0,0,0">

<Image Source="/UPDATE.png" Stretch="Uniform" OpacityMask="Black" />

</Button>

</StackPanel>

<Label x:Name="TableNameLabel" Content="Выбранная таблица: Агенты недвижимости" Margin="10,0,10,5" Background="Transparent" Foreground="White"/>

</StackPanel>

</Border>

</Grid>

</Window>