**Содержание**

[Введение 3](#_Toc96585234)

[1. Предварительное планирование и проверка осуществимости 4](#_Toc96585235)

[1.1. Требования к системе в целом 4](#_Toc96585236)

[1.2. Требования к функциям, выполняемым системой 6](#_Toc96585237)

[1.3. Требования к видам обеспечения 7](#_Toc96585238)

[2. Постановка задачи 10](#_Toc96585239)

[2.1. Общие сведения 10](#_Toc96585240)

[2.2. Назначение и цели создания системы 11](#_Toc96585241)

[2.3. Характеристики объекта автоматизации 12](#_Toc96585242)

[2.4. Состав и содержание работ по созданию системы 13](#_Toc96585243)

[2.5. Порядок контроля и приемки программного обеспечения 14](#_Toc96585244)

[2.6. Требования к документированию 14](#_Toc96585245)

[3. Разработка приложения 15](#_Toc96585246)

[4. Реализация 16](#_Toc96585247)

[4.1. Выбор и обоснование IDE 16](#_Toc96585248)

[4.2. Разработка логики приложения. 16](#_Toc96585249)

[4.3. Написание кода 16](#_Toc96585250)

[4.4. Создание UI 16](#_Toc96585251)

[Заключение 17](#_Toc96585252)

[Список использованных источников 18](#_Toc96585253)

[Приложение 19](#_Toc96585254)

# Введение

Темой данного курсового проекта является «Разработка модуля информационной системы для агентства недвижимости».

Агентство недвижимости – специализированная коммерческая организация, осуществляющая как, правило посреднические услуги между продавцом и конечным покупателем на рынке недвижимости. С целью получения максимальной прибыли.

На постоянной основе в агентствах недвижимости ведение учета клиентов и их заявок и составление базы данных недвижимости проводится вручную с помощью различных книг учета. Информация структурирована частично, отсутствует возможность быстрого поиска недвижимости и оперативности в общем и автоматизированного формирования итоговых отчетов, нет возможности автоматического получения отчета о деятельности агентства за выбранный период времени.

**Актуальность темы** – потребность в автоматизации процесса приобретения, подбора квартир в агентстве недвижимости.

**Цель** – разработка модуля информационной системы для агентства недвижимости.

**Объект** – продажа недвижимости, поиск продавца покупателей, составление списка недвижимости.

**Предмет** – модуль для агентства недвижимости.

**Для достижения цели, требуется выполнить следующие задачи:**

1. Провести предварительное планирование и проверку осуществимости.
2. Определить требования к программному продукту.
3. Разработать прототип приложения.
4. Реализовать приложение в выбранной среде IDE.
5. Разместить проект на веб-сервисе GitHub или аналогичном.

# Предварительное планирование и проверка осуществимости

# 1.1. Требования к системе в целом

1.1.1. Требования к структуре и функционированию системе

Система должна состоять из сервера баз данных, приложения.

Сервер БД отвечает за хранение базы данных. Приложение организует работу обмен данными с сервером баз данных. Приложение – основная часть системы – обрабатывает данные полученные от сервера БД и выводит их в понятном для конечного пользователя виде, а также формирует запросы от пользователя к серверу БД.

1.1.2. Требования к численности и квалификации персонала

Пользователями приложения являются потенциальные клиенты. Пользователи используют систему через приложение, поэтому количество людей, желающих просмотреть приложение или воспользоваться услугами неограниченно.

Обслуживающий персонал:

• системный администратор – 1 человек;

Пользователи должны иметь минимальные навыки работы на компьютере и в приложениях.

Обслуживающий персонал должен являться продвинутыми пользователями в компьютере и в глобальной сети Интернет, а также иметь опыт работы с SQL Server и владеть языком программирования C#.

Системный администратор должен иметь опыт работы с SQL Server, C#.

1.1.3. Требования к надежности

Программное обеспечение системы должно функционировать и не приводить к зависанию, к аварийному завершению, критическим ошибкам системы при любых ошибочных действиях в данном приложении.

1.1.4. Требования к эргономике и технической эстетике

Программное обеспечение и технические средства должны иметь эстетически привлекательный вид и быть удобными в использовании.

1.1.5. Требования к эксплуатации и техническому обслуживанию

Для нормальной работы системы необходимо произвести ряд действий

Приложение:

• установить лицензионную копию ОС;

• запустить программу установки приложения.

• на сервере базы данных:

• установить лицензионную копию ОС;

• установить копию SQL Server;

• запустить программу установки сервера БД.

1.1.6. Требования к защите информации от несанкционированного доступа

Защита ИС и информации, с которой она работает, от несанкционированного разграничение доступа по ролям и частично возлагается на операционную систему. Доступ к данным предоставляется только легальным пользователям.

1.1.7. Требования по сохранности информации при авариях

Сохранность данных должна быть обеспеченна:

• при сбоях в работе оборудования сети в любой момент времени;

• при сбоях электропитания или программного обеспечения не ведущих к физическому уничтожению или повреждению информации.

1.1.8. Требования к защите от влияния внешних воздействий

Автоматизированная система должна быть устойчивой к наличию и параллельной с ней работе на том же ПК другого программного обеспечения. Должна быть устойчивой к одновременной работе пользователей.

1.1.9. Требования к стандартизации и унификации

Автоматизированная система должна предоставлять пользователю интуитивно понятный интерфейс. Программная документация, поставляемая с программой, должна быть оформлена в соответствии со стандартом ЕСПД.

# 1.2. Требования к функциям, выполняемым системой

Функции подсистемы «Сервер базы данных»:

• получение запросов от «клиента БД»

• проверка легальности выполнения запроса на основе идентификационной информации, вложенной в запрос;

• выполнение SQL-запросов к БД;

• передача результата SQL-запроса «Приложение».

Функции подсистемы «Приложение»:

• передача пользователю следующей информации:

• информация о разделах;

• информация об недвижимости;

• визуальная информация о недвижимости.

• изменение информации:

• изменение информации о разделах;

• информация об адресах и телефонов магазинов и о людях, имеющих отношение к организации свадеб;

• формирование и передача запросов серверу БД;

• получение от сервера БД результатов запросов.

# 1.3. Требования к видам обеспечения

1.2.1. Требования к математическому обеспечению

В основе разрабатываемого ПО должна лежать реляционная модель базы данных. Все таблицы должны находиться в третьей или четвертой нормальной форме. Связи между таблицами должны быть вида: «один к одному» или «один ко многим». При проектировании используется метод «сущность – связь».

1.3.2. Требования к информационному обеспечению

Данные в системе должны быть организованны в виде связанных таблиц. Хранение и доступ к данным осуществляется с применением реляционные систем управления базами данных. Информационный обмен между компонентами системы осуществляется по технологии приложение-сервер. Желательно обеспечить возможность резервного копирования и восстановления данных.

1.3.3. Требования к лингвистическому обеспечению

В качестве языка программирования используются – МYSQL, С#, взаимодействие пользователя и ПО осуществляется через оконный интерфейс.

1.3.4. Требования к программному обеспечению

Система должна иметь следующее программное обеспечение: ОС MS Windows 7 или выше, C#, WPF, SQL Server

1.3.5. Требования к техническому обеспечению

Для работоспособности приложение необходимы следующие программные и технические средства:

• клавиатура;

• мышь;

• процессор Intel core Pentium или выше;

• оперативная память 2GB и больше;

• видеокарта и монитор, поддерживающего разрешение 1280x720точек и больше (рекомендуется 1920х1080 точек) при глубине цвета 16 бит и более;

• дисковое пространство 1 Гб и больше.

# 2. Постановка задачи

# 2.1. Общие сведения

2.1.1. Полное наименование программного обеспечения и его условное обозначение

Полное наименование – "Разработка приложения «Агентство недвижимости»”.

2.1.3. Наименование предприятий разработчика и заказчика системы

Разработчик - студент группы 402ИСП, Филиппов К.П.

Заказчик – КГБ ПОУ АПТ, в лице преподавателя Тихонов Е.О.

2.1.4. Перечень документов, на основании которых создается программное обеспечение

Перечень документов, на основании которых создается программное обеспечение:

• задание на курсовое проектирование;

• техническое задание;

• календарный план-график выполнения этапов работы (см. п. 2.4 данного технического задания).

2.1.5. Плановые сроки начала и окончания работы по созданию системы

Плановый срок начала работ: 23.01.2023 г.

Плановый срок окончания работ: 04.03.2023 г.

2.1.6. Сведения об источниках и порядке финансирования работ

Данная работа выполняется в рамках курсового проектирования, финансирование не производится.

2.1.7. Порядок оформления и предъявления заказчику результатов работы по созданию программного обеспечения

Порядок следующий:

• после подписания технического задания создается рабочий вариант программного обеспечения. Работа осуществляется в соответствии с графиком (см. п. 2.4 данного технического задания). По окончании работ программное обеспечение тестируется и передается заказчику для эксплуатации;

• также заказчику передается комплект технической документации, оформленный в соответствии с ГОСТ 34.201 и ЕСПД.

# 2.2. Назначение и цели создания системы

2.2.1. Назначение системы

Автоматизация процесса поиска и продажи недвижимости для клиентов агентства недвижимость.

2.2.2. Цели создания системы

Цели создания приложения:

• помощь в продаже недвижимости;

• составление списка недвижимости;

• повышение работоспособности сотрудников и оперативности.

• привлечения большего числа клиентов за счет удобности

# 2.3. Характеристики объекта автоматизации

2.2.1. Краткие сведения об объекте автоматизации

Объекты автоматизации:

• процесс получения информации о недвижимости на продажу;

• сортировка информации;

• составление базы данных недвижимости;

• составление клиентской базы данных.

В процессе выдачи информации разделах должны отображаться следующие данные:

• название раздела;

• полное описание раздела.

В процессе выдачи информации об клиентской базе данных отображается:

• Ф.И.О;

• адрес, телефон, e-mail;

2.3.2. Сведения об условиях эксплуатации объекта автоматизации и характеристик окружающей среды

Условия эксплуатации объекта автоматизации (на стороне приложения):

• операционная система MS Windows 7 и выше;

Условия эксплуатации объекта автоматизации (на стороне сервера БД):

• операционная система MS Windows 7 и выше;

• SQL Server;

Условия эксплуатации объекта автоматизации (на стороне клиента):

• операционная система MS Windows 7 или выше;

Физические условия эксплуатации и характеристики окружающей среды определяются соответствующими характеристиками окружающей среды используемого аппаратного обеспечения, указанными в документации на него.

# 2.4. Состав и содержание работ по созданию системы

Этапы выполнения работ по созданию программного обеспечения приведены в таблице 1.1.

Таблица 2.1 – Этапы выполнения работ

|  |  |  |  |
| --- | --- | --- | --- |
| Этап | Дата начала | Срок окончания | |
| Минима-льный | Максима-льный |
| 1. Введение | 23.01 | 23.01 | 23.01 |
| 2. Предварительное планирование и проверка осуществимости | 26.01 | 23.01 | 24.01 |
| 3. Постановка задачи | 26.01 | 26.01 | 27.01 |
| 4. Разработка приложения | 30.01 | 01.02 | 02.02 |
| 5. Реализация | 02.02 | 05.02 | 08.02 |
| 5.1. Выбор и обоснование IDE | 02.02 | 03.02 | 04.02 |
| 5.2. Разработка логики приложения. | 05.02 | 06.02 | 07.02 |
| 5.3. Написание кода | 08.02 | 14.02 | 15.02 |
| 5.4. Создание UI | 16.02 | 25.02 | 26.02 |
| 6. Заключение | 27.02 | 01.02 | 02.03 |
| 7. Список использованных источников | 1.03 | 03.03 | 4.03 |

# 2.5. Порядок контроля и приемки программного обеспечения

Контроль системы и ее приемка осуществляется в следующей последовательности:

1. тестирование системы по следующим параметрам:

• стабильное отображения интерфейса и списка приложения

• стабильность работы системы при выполнении любых доступных функций на каждом шаге ее работы;

2. исправление недостатков и ошибок системы, выявленных в результате тестирования;

3. повторное тестирование и исправление ошибок;

4. при устранении всех выявленных ошибок сдача системы в эксплуатацию.

# 2.6. Требования к документированию

Пояснительная записка оформляется в соответствии с ГОСТ 19.106-78 от 01.01.80 «Требования к программным документам, выполненным печатным способом», ГОСТ 19.101-77 от 01.01.80 «Виды программ и программных документов», ГОСТ 2.304-81 от 01.01.82 «Шрифты чертежные», ГОСТ 2.301-68 от 01.01.71 «Форматы ЕСКД», ЕСПД ГОСТ 19.001-77 от 1.01.80.

# 3. Разработка приложения

Прототип страницы Главная – показывает возможный вариант страницы Главная. Показано окно авторизации, регистрации и предпросмотра. (см. Рисунок 3.1)

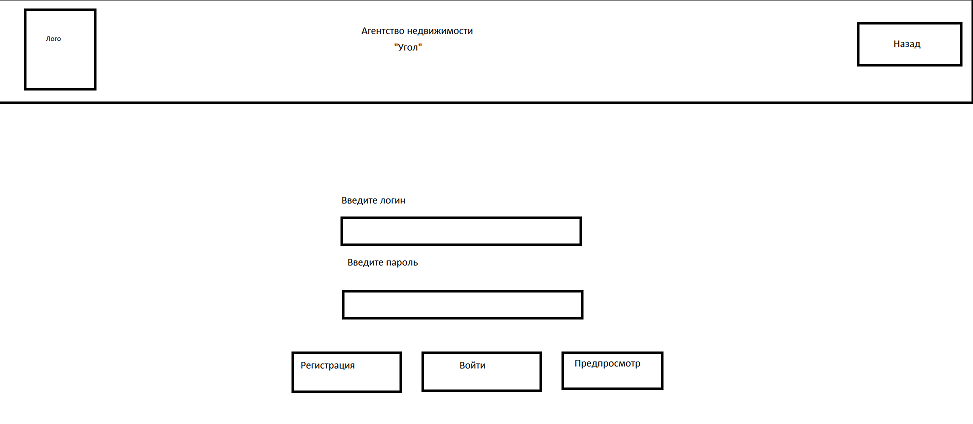


Рисунок 3.1 - Прототип страницы Главная

Прототип страницы Регистрация – показывает интерфейс страницы Регистрации. Показано кнопка создания аккаунта и места для ввода данных (см. Рисунок 3.2)

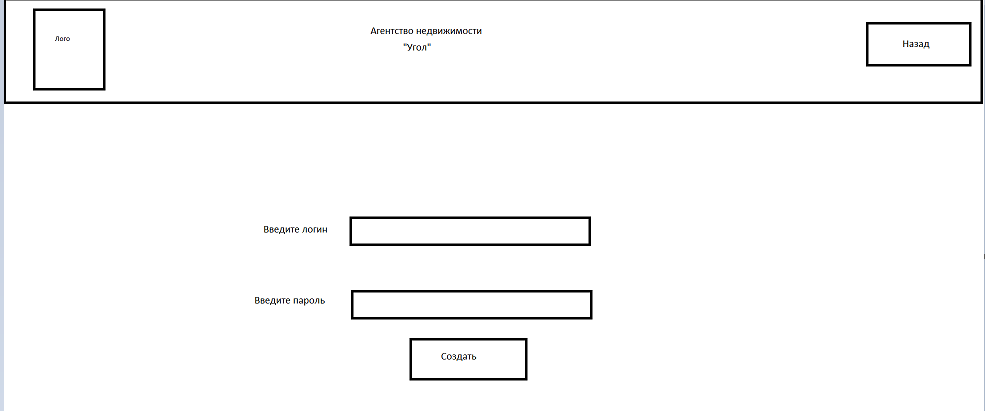


Рисунок 3.2 - Прототип страницы Регистрация

Прототип страницы Список – показывает список недвижимости на продажу в агентстве недвижимости (см. Рисунок 3.3)

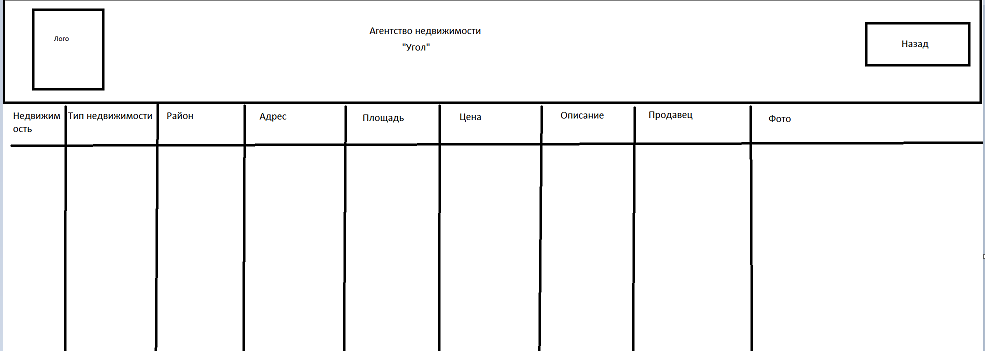


Рисунок 3.3 - Прототип страницы Список

# 4. Реализация

# 4.1. Выбор и обоснование IDE

* Visual Studio – представляет собой полностью интегрированную среду разработки. Она спроектирована таким образом, чтобы делать процесс написания кода, его отладки и компиляции в сборку для поставки конечным потребителям как можно более простым. Является наиболее функционально насыщенной IDE-средой, в которой можно разрабатывать приложения на C#. Данный продукт поставляется с конструкторами графических пользовательских интерфейсов, с возможностями работы с базами данных, с поддержкой фрагментов кода, с возможностями для просмотра всего проекта в целом, с просмотром свойств объектов.
* Standard Widget Toolkit – библиотека графических пользовательских элементов с открытым исходным кодом для разработки интерфейсов пользователя на языке Java. Предоставляет из себя переносимый API для разработчиков Java. Использует собственные библиотеки операционной системы через собственный интерфейс Java (Java Native Interface) для визуализации элементов графического интерфейса, аналогично программам, написанным с использованием собственного API операционной системы.
* Qt Creator – кроссплатформенная свободная IDE для разработки на С, С++ и QML с помощью фреймворка Qt. Разработана Trolltech для работы с фреймворком Qt. Включает в себя графический интерфейс отладчика и визуальные средства разработки интерфейса как с использованием QtWidgets, так и QML. Зачастую используется при написания программного обеспечения для роботов, автомобилей и кораблей, которые используются на производствах, в перевозке грузов и других похожих отраслях. Qt Creator позволяет добиться высокой производительности при прототипировании интерфейса в приложениях любой сложности.
* CLion – продукт JetBrains. Многофункциональная IDE. В которой поддерживается разработка кода на C и C++, а также web-разработка на HTML (5), CSS, JavaScript, XML. А также на некоторых других языках доступнен в виде плагинов (например, Lua). Основным преимуществом является статический и динамический анализ с безопасный рефакторингом кода. Позволяет кастомизировать редактор так, как удобно пользователю.
* IntelliJ IDEA – ещё один продукт от JetBrains. Комплекс программных средств, который используется для написания, исполнения, отладки и оптимизации кода для Java, JavaScript, Python и других языков программирования от компании JetBrains. Отличается обширным набором инструментов для рефакторинга (перепроектирования) и оптимизации кода.  
  Интеграция с серверами приложений, включая Tomcat, TomEE, GlassFish и другие. Имеет набор инструментов для работы с базами данных, SQL файлами и инструменты для запуска тестов и анализа покрытия кода, включая поддержку всех популярных фреймворков для тестирования.

Для разработки программного обеспечения мной было принято решение реализовывать информационную системы с нуля в среде Visual Studio на платформе Windows Presentation Foundation (WPF) используя язык C#. Ниже описано каждое средство реализации, вместе с их преимуществами среди других.

* Поддержка контроля версий GIT;
* Технология автодополнения IntelliSense, которая умеет исправлять ошибки разработчика, предлагая ряд вариантов, но, в отдельных случаях, может и сгенерировать фрагменты кода;
* Расширенный функционал, который можно дополнятся плагинами;
* Интегрированная среда разработки (IDE);
* Удобный и интуитивно понятный интерфейс, который можно кастомизировать под каждого пользователя индивидуально;
* Присутствует встроенный отладчик;
* Наличие встроенного средства для быстрого рефакторинга;
* Множество функций, которые позволяют изменять редактор в соответствии с использованием, что означает, что пользователь может загружать библиотеки из Интернета и интегрировать его с кодом в соответствии со своими требованиями.

# 4.2. Разработка логики приложения.

ER-диаграмма представлена на рисунке 4.1



Рисунок 4.1 - ER-диаграмма

Таблица «Sdelki» – сделки клиентов агентства недвижимости и их данные. Данные таблицы «Sdelki» рисунок 4.2.

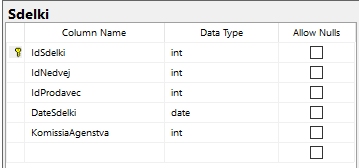


Рисунок 4.2 - данные таблицы Sdelki

Таблица 1. Описание структуры «Sdelki».

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| IdSdelki | int | Идентификатор сделки |
| IdNedvej | int | Идентификатор недвижимости |
| IdProdavec | int | Идентификатор продавца |
| DateSdelki | date | Дата сделки |
| KomissiaAgenstva | int | Комиссия агентства за продажу |

Таблица «Prodavec» - данные о продавцах, которые воспользовались агентством недвижимости. Данные таблицы «Prodavec» рисунок 4.3.

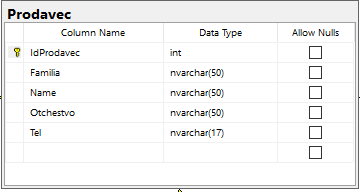


Рисунок 4.3 - данные таблицы Prodavec

Таблица 2. Описание структуры «Prodavec».

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| IdProdavec | int | Идентификатор продавца |
| Familia | nvarchar(50) | Фамилия |
| Name | nvarchar(50) | Имя |
| Otchestvo | nvarchar(50) | Отчество |
| Tel | nvarchar(17) | Номер телефона |

Таблица «Users» - данные о пользователях, которые используют приложения агентства недвижимости. Данные таблица «Users» рисунок 4.4.

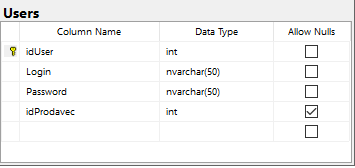


Рисунок 4.4 - данные таблицы Users

Таблица 3. Описание структуры «Users».

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| IdUser | int | Идентификатор пользователя |
| Login | nvarchar(50) | Логин |
| Password | nvarchar(50) | Пароль |
| IdProdavec | int | Идентификатор продавца |

Таблица «TypeNedvez» - тип недвижимости. Данные таблица «TypeNedvez» рисунок 4.6.

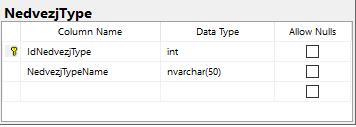


Рисунок 4.6 - данные таблица TypeNedvez

Таблица 4. Описание структуры «TypeNedvez».

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| IdNedvezjType | int | Идентификатор типа недвижимости |
| NedvezjTypeName | nvarchar(50) | Наименование типа недвижимости |

Таблица «Rayon» - данные о районах города. Данные таблица «Rayon» рисунок 4.7.

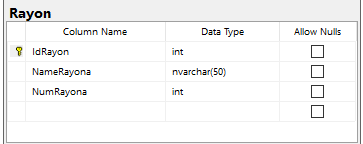


Рисунок 4.7 - данные таблицы Rayon

Таблица 5. Описание структуры «Rayon».

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| IdRayon | int | Идентификатор района |
| NameRayona | nvarchar(50) | Название района |
| NumRayona | int | Номер района |

Таблица «Nedvezj» - данные о недвижимости. Данные таблица «Nedvezj» рисунок 4.8.

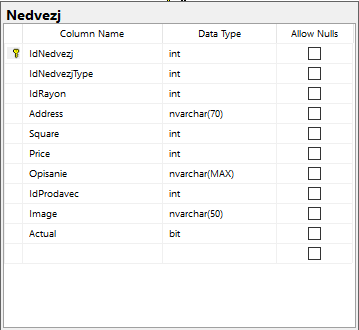


Рисунок 4.8 - данные таблицы Nedvezj

Таблица 6. Описание структуры «Nedvezj».

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| IdNedvezj | int | Идентификатор недвижимости |
| IdNedvezjType | int | Идентификатор типа недвижимости |
| IdRayon | int | Идентификатор района |
| Address | nvarchar(70) | Адрес недвижимости |
| Square | int | Площадь недвижимости |
| Price | int | Цена недвижимости |
| Opisanie | nvarchar(MAX) | Описание недвижимости |
| IdProdavec | int | Идентификатор продавца |
| Image | nvarchar(50) | Фотография недвижимости |
| Actual | bit | Актуально ли предложение |

# 4.3. Написание кода

# Таблица 7. Листинг и описание интерфейса окна MainWindow

|  |  |
| --- | --- |
| Листинг кода | Комментарий к блоку кода |
| private void btnIn\_Click(object sender, RoutedEventArgs e)  {  try  {  var userData = AgenstvoNedvezjEntities2.GetContext().Users.FirstOrDefault(x => x.Login == txbLogin.Text && x.Password == psbPassword.Password);  if (userData == null)  {  MessageBox.Show("Такого пользователя нет!", "Ошибка авторизации!", MessageBoxButton.OK, MessageBoxImage.Error);  }  else  {  Manager.MainFrame.Navigate(new List());  MessageBox.Show("Здравствуйте, " + userData.Login + "!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);  }  }  catch (Exception Ex)  {  MessageBox.Show("Ошибка" + Ex.Message.ToString() + "Критическая ошибка приложения!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);  } } |  |

# 4.4. Создание UI

При запуске при приложения, клиент видит страницу “Главная”. На ней находятся два поля для ввода данных и три кнопки. Кнопка “Войти” после ввода данных, которая позволяет перейти к списку недвижимости, кнопка “Предпросмотр” для входа без ввода данных и кнопка “Регистрация”, которое переходит на страницу регистрации. Состоит из трех блоков: header, footer, main block. Страница Главная (см. Рисунок 4.9)

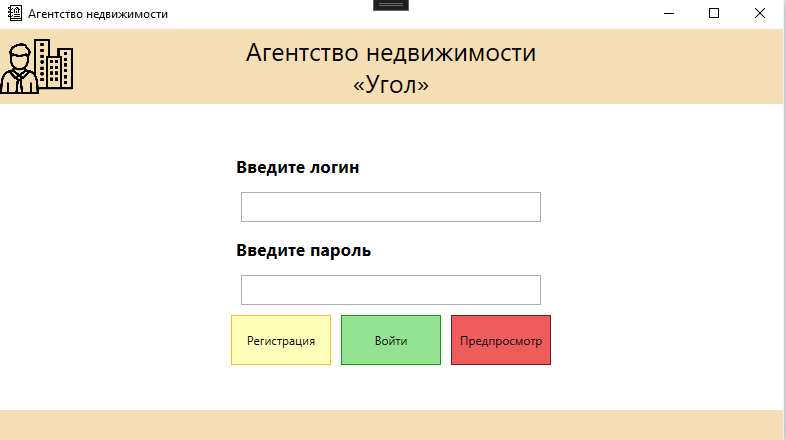


Рисунок 4.9 - страница Главная

При нажатии кнопки регистрация, клиент переходит на окно регистрации, где ему нужно придумать и ввести данные для того чтобы зарегистрированным пользователем. После ввода данных нужно нажать кнопку “Cоздать”, после этого под этим логином и пароль можно произвести вход. Состоит из трех блоков: header, footer, main block. Так же есть кнопка “Назад” чтобы вернуться на страницу “Главная” (см. Рисунок 4.10)

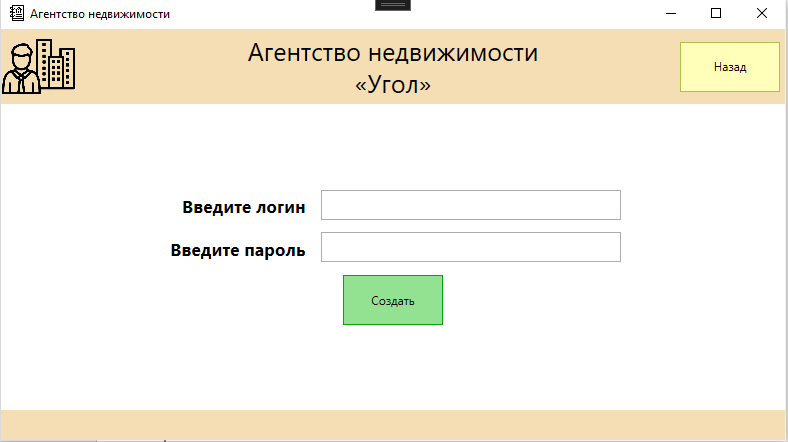


Рисунок 4.10 – страница Регистрация

При нажатии кнопки “Войти” или “Предпросмотр” идет переход на страницу главная на которой можно посмотреть список недвижимости, которая есть в агентстве. Состоит из трех блоков: header, footer, main block. Так же есть кнопка “Назад” чтобы вернуться в страницу “Главная” (см. Рисунок 4.11)

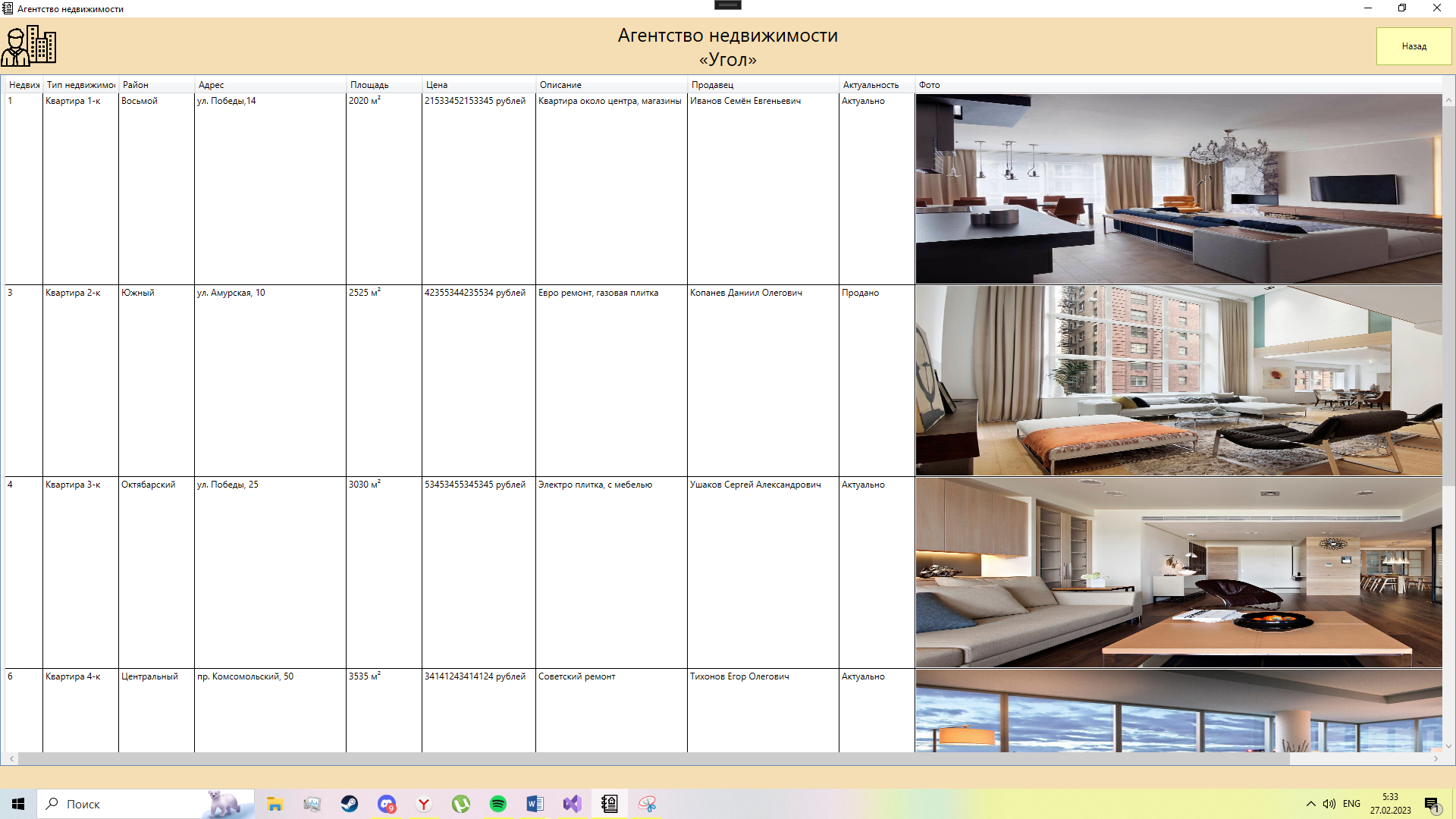


Рисунок 4.11 – страница Список

# Заключение

Заключение по целям и задачам из введения(п2.2): В ходе курсового проекта, я произвел анализ своей предметной области, выделил основные сущности и спроектировал их. Создал макет приложения, в котором реализовал регистрацию, авторизацию, просмотр списка недвижимости, тем самым автоматизировал процесс поиска и продажи недвижимости для клиентов агентства недвижимость.

Цель выполнена(введение), задачи выполнены

1. Провел предварительное планирование и проверку осуществимости.
2. Определил требования к программному продукту.
3. Разработал прототип приложения.
4. Реализовал приложение в выбранной среде IDE.
5. Создал сборку для установки.
6. Разместил проект на веб-сервисе GitHub -https://github.com/NekoYo1/Kursovoi\_Filippov.git

# Список использованных источников

1. **БардзеллД.** Macromedia Dreamweaver MX 2004 с ASP, Cold Fusionи PHP. Из первых рук (+ CD-ROM); Эком - М., 2016. - 560 c. ISBN: 5-7163-0141-X, 0-321-24157-6

2. **Бенкен Елена** PHP, MySQL, XML. Программирование для Интернета; БХВ-Петербург - М., 2017. - 336 c. ISBN 978-5-9775-0724-0

3. **Гизберт Дамашке** PHP и MySQL; НТ Пресс - М., 2016. - 569 c. ISBN: 978-5-9775-0125-5

**Интеренет-источники:**

1. Волобуева, Т. В. Информатика. Основы алгоритмизации: учебное пособие / Т. В. Волобуева. — Воронеж: Воронежский государственный технический университет, ЭБС АСВ, 2019. — 73 c. — ISBN 978-5-7731-0740-8. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование: [сайт]. — URL: <https://profspo.ru/books/93316> (дата обращения: 28.02.2022). — Режим доступа: для авторизир. пользователей

# Приложение

**Приложение А**

**Листинги программного кода**

|  |  |
| --- | --- |
| **MainWindow** | |
| **xaml** | **cs** |
| Title="Агентство недвижимости" Height="450" Width="800" Icon="Resources/ico.png" >  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="75"/>  <RowDefinition Height="\*"/>  <RowDefinition Height="30"/>  </Grid.RowDefinitions>  <Grid Background="Wheat"></Grid>  <Image Source="/Resources/logo.png" HorizontalAlignment="Left"></Image>  <TextBlock Text="Агентство недвижимости" FontSize="25" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0 5 0 0"></TextBlock>  <TextBlock Text="«Угол»" FontSize="25" HorizontalAlignment="Center" VerticalAlignment="Bottom" Margin="0 0 0 5"></TextBlock>  <Frame NavigationUIVisibility="Hidden" Grid.Row="1" ContentRendered="MainFrame\_ContentRendered" Name="MainFrame"></Frame>  <Grid Grid.Row="2" Background="Wheat"></Grid>  <Button Content="Назад" Style="{StaticResource ButtonGoBack}" Name="BtnBack" HorizontalAlignment="Right" Click="BtnBack\_Click"></Button>  </Grid>  </Window> | public partial class MainWindow : Window  {  public MainWindow()  {  InitializeComponent();  MainFrame.Navigate(new Authorization());  Manager.MainFrame = MainFrame;  }  private void MainFrame\_ContentRendered(object sender, EventArgs e)  {  if (MainFrame.CanGoBack)  {  BtnBack.Visibility = Visibility.Visible;  }  else  {  BtnBack.Visibility = Visibility.Hidden;  }  }  private void BtnBack\_Click(object sender, RoutedEventArgs e)  {  Manager.MainFrame.GoBack();  }  }  } |
|  | |

|  |  |
| --- | --- |
| **List** | |
| **xaml** | **cs** |
| Title="List">  <Grid>  <Grid.RowDefinitions>  </Grid.RowDefinitions>  <DataGrid x:Name="DGridList" AutoGenerateColumns="False" IsReadOnly="True">  <DataGrid.Columns>  <DataGridTextColumn Header="Недвижимость" Binding="{Binding IdNedvezj}" Width="50"></DataGridTextColumn>  <DataGridTextColumn Header="Тип недвижимости" Binding="{Binding TypeNJ}" Width="100"></DataGridTextColumn>  <DataGridTextColumn Header="Район" Binding="{Binding RayonName}" Width="100"></DataGridTextColumn>  <DataGridTextColumn Header="Адрес" Binding="{Binding Address}" Width="200"></DataGridTextColumn>  <DataGridTextColumn Header="Площадь" Width="100" Binding="{Binding SquareM}"></DataGridTextColumn>  <DataGridTextColumn Header="Цена" Width="150" Binding="{Binding PriceRub}"></DataGridTextColumn>  <DataGridTextColumn Header="Описание" Width="200" Binding="{Binding Opisanie}"></DataGridTextColumn>  <DataGridTextColumn Header="Продавец" Width="200" Binding="{Binding ProdavecName}"></DataGridTextColumn>  <DataGridTextColumn Header="Актуальность" Width="100" Binding="{Binding Actuality}"></DataGridTextColumn>  <DataGridTemplateColumn Width="auto" Header="Фото">  <DataGridTemplateColumn.CellTemplate>  <DataTemplate >  <Image Source="{Binding ImagePath}" Stretch="Fill" Width="900" Height="250"></Image>  </DataTemplate>  </DataGridTemplateColumn.CellTemplate>  </DataGridTemplateColumn>  </DataGrid.Columns>  </DataGrid>  </Grid>  </Page> | public partial class List : Page  {  public List()  {  InitializeComponent();  DGridList.ItemsSource = AgenstvoNedvezjEntities2.GetContext().Nedvezj.ToList();  }  }  } |
|  | |

|  |  |
| --- | --- |
| **Authorization** | |
| **xaml** | **cs** |
| Title="Authorization">  <Grid>  <StackPanel Grid.Row="1" VerticalAlignment="Center" HorizontalAlignment="Center">  <Label Content="Введите логин" Style="{StaticResource Label}"/>  <TextBox Text="" Style="{StaticResource TextBox}" x:Name="txbLogin"/>  <Label Content="Введите пароль" Style="{StaticResource Label}"/>  <PasswordBox Style="{StaticResource PasswordBox}" x:Name="psbPassword"/>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">  <Button Content="Регистрация" Style="{StaticResource ButtonReg}" Click="btnRegIn\_Click"/>  <Button Content="Войти" Style="{StaticResource ButtonIn}" Click="btnIn\_Click"/>  <Button Content="Предпросмотр" Style="{StaticResource ButtonPred}" Click="Button\_Click"></Button>  </StackPanel>  </StackPanel>  </Grid>  </Page> | public partial class Authorization : Page  {  public Authorization()  {  InitializeComponent();  }  private void btnIn\_Click(object sender, RoutedEventArgs e)  {  try  {  var userData = AgenstvoNedvezjEntities2.GetContext().Users.FirstOrDefault(x => x.Login == txbLogin.Text && x.Password == psbPassword.Password);  if (userData == null)  {  MessageBox.Show("Такого пользователя нет!", "Ошибка авторизации!", MessageBoxButton.OK, MessageBoxImage.Error);  }  else  {  Manager.MainFrame.Navigate(new List());  MessageBox.Show("Здравствуйте, " + userData.Login + "!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);  }  }  catch (Exception Ex)  {  MessageBox.Show("Ошибка" + Ex.Message.ToString() + "Критическая ошибка приложения!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);  }  }  private void btnRegIn\_Click(object sender, RoutedEventArgs e)  {  Manager.MainFrame.Navigate(new Registration());  }  private void Button\_Click(object sender, RoutedEventArgs e)  {  Manager.MainFrame.Navigate(new List());  }  }  } |
|  | |

|  |  |
| --- | --- |
| **Registration** | |
| **xaml** | **cs** |
| Title="Registration">  <Grid>  <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">  <StackPanel Orientation="Horizontal" HorizontalAlignment="Right"/>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Right">  <Label Content="Введите логин" Style="{StaticResource Label}"/>  <TextBox x:Name="txbLogin" Text="" Style="{StaticResource TextBox}"/>  </StackPanel>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Right">  <Label Content="Введите пароль" Style="{StaticResource Label}"/>  <StackPanel>  <TextBox x:Name="txbPass" Text="" Style="{StaticResource TextBox}"/>  </StackPanel>  </StackPanel>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">  <Button x:Name="btnCreate" Content="Создать" Style="{StaticResource ButtonIn}" Click="btnCreate\_Click"/>  </StackPanel>  </StackPanel>  </Grid>  </Page> | public partial class Registration : Page  {  public Registration()  {  InitializeComponent();  }  private void btnCreate\_Click(object sender, RoutedEventArgs e)  {    if (AgenstvoNedvezjEntities2.GetContext().Users.Count(x => x.Login == txbLogin.Text) > 0)  {  MessageBox.Show("Пользователь с таким логином есть!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);  return;  }    try  {  Users userData = new Users()  {  Login = txbLogin.Text,  Password = txbPass.Text,  };  AgenstvoNedvezjEntities2.GetContext().Users.Add(userData);  AgenstvoNedvezjEntities2.GetContext().SaveChanges();  MessageBox.Show("Данные успешно добавлены!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);  }  catch  {  MessageBox.Show("Ошибка при добавлении данных!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Error);  }  }  }  } |
|  | |