**Содержание**

1 Восстановление базы данных из скрипта 5

2 Импорт данных 6

3Разработка desktop-приложений. Список агентов 8

4 Разработка desktop-приложений. Добавление агентов 10

5 Разработка desktop-приложений. Редактирование агентов 12

6Изучение и описание современных методов разработки программного обеспечения 15

7 Применение стандартов кодирования при разработке приложения 16

8 Применение систем контроля версий 18

**1 Восстановление базы данных из скрипта**

Создаю базу данных в Visual Studio

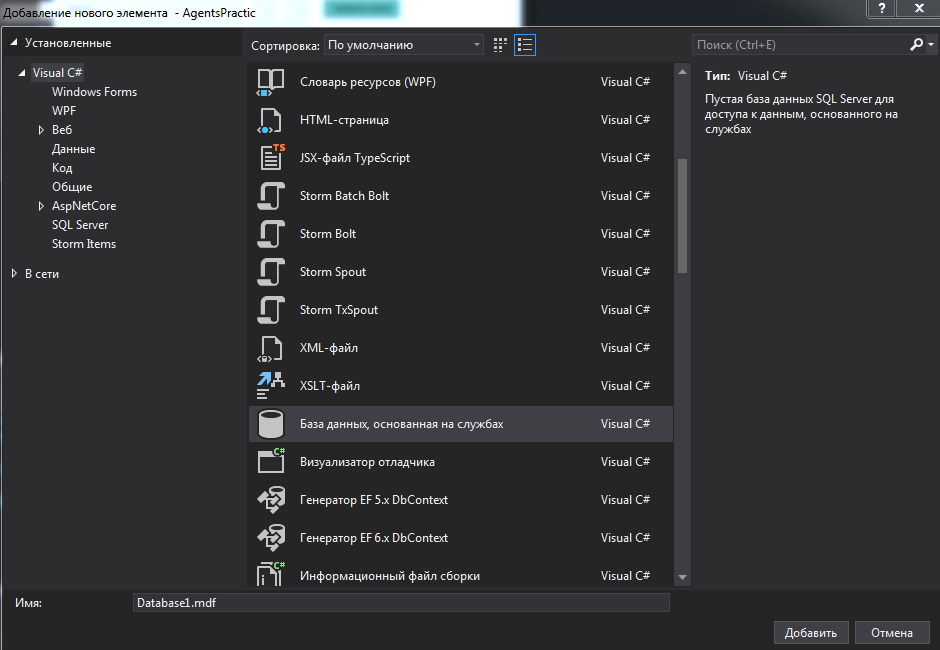


Рисунок 1 – создание БД

После просмотра данных для импорта решаю делать скрипт только для 3 таблиц

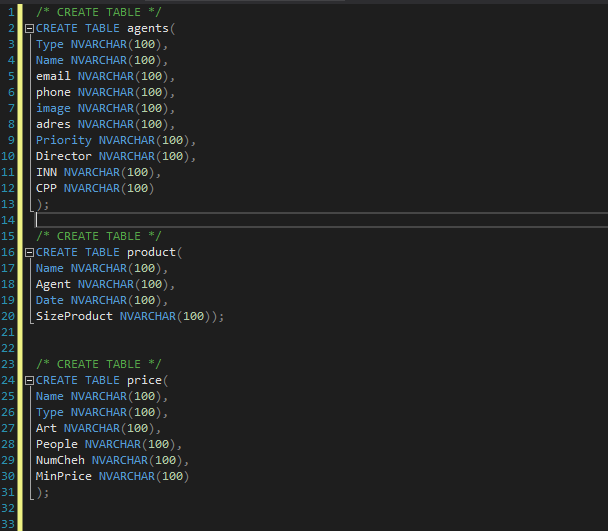
****

Рисунок 2 – создание таблиц

**2 Импорт данных**

Нормализую таблицы Excel

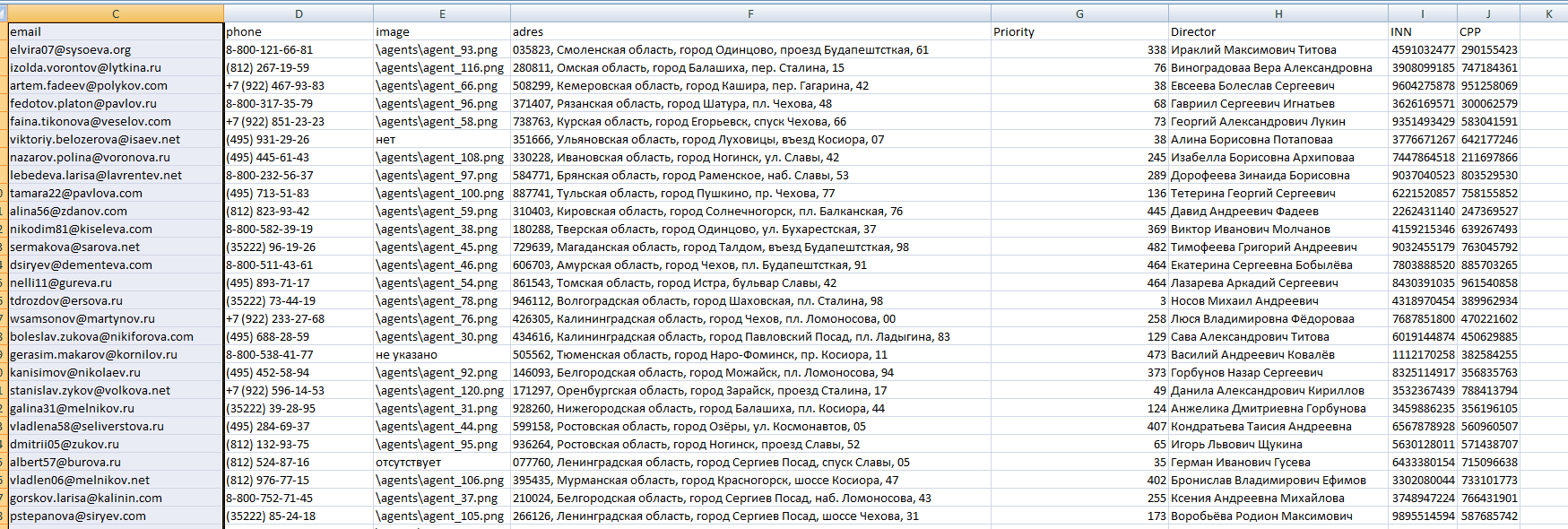


Рисунок 3 – нормализация таблицы

С помощью онлайн конвертора создаю SQL запрос из таблицы Excel

****

Рисунок 4 – конвертация

Заполнение таблиц БД запросом

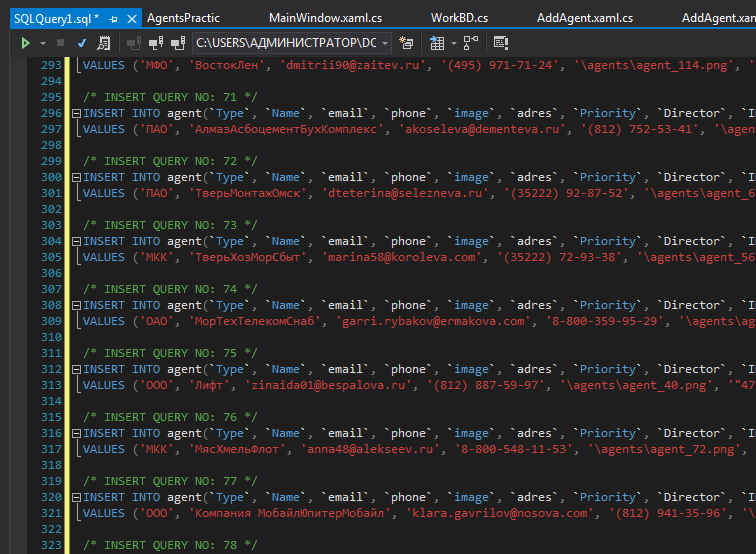
****

Рисунок 5 – заполнение таблиц

Проверка заполненных таблиц в БД

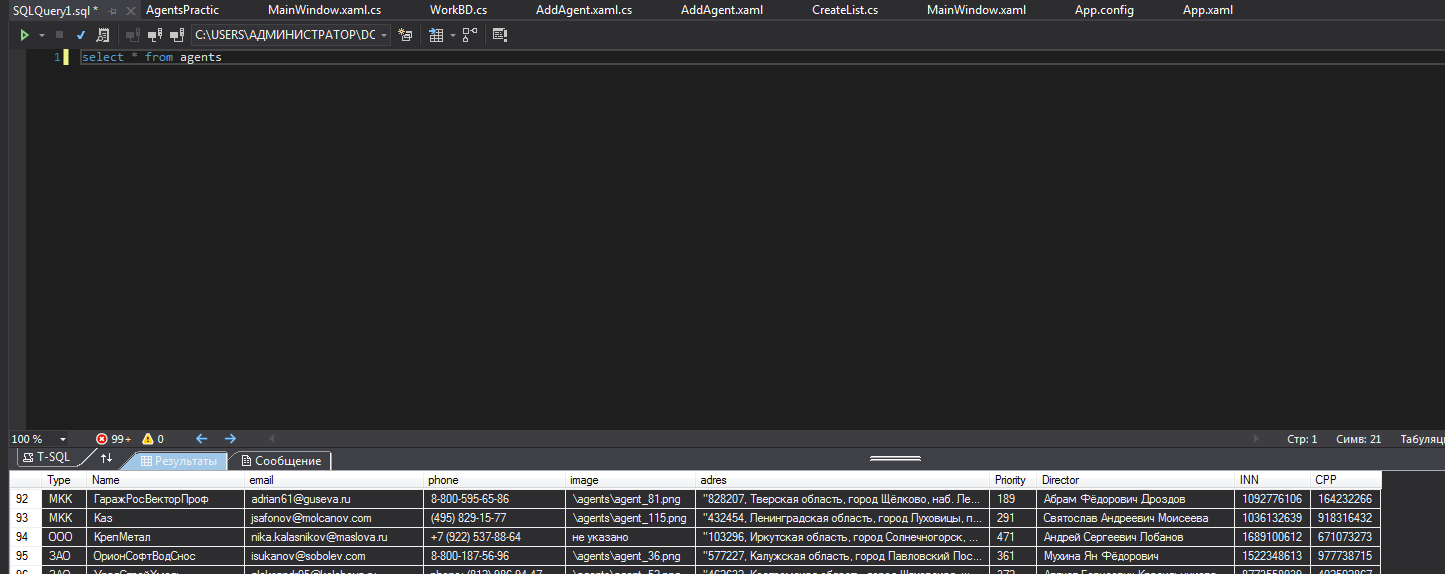
****

Рисунок 6 – проверка таблиц

**3 Разработка desktop-приложений. Список агентов**

Создаю элементы для шаблона списка агентов

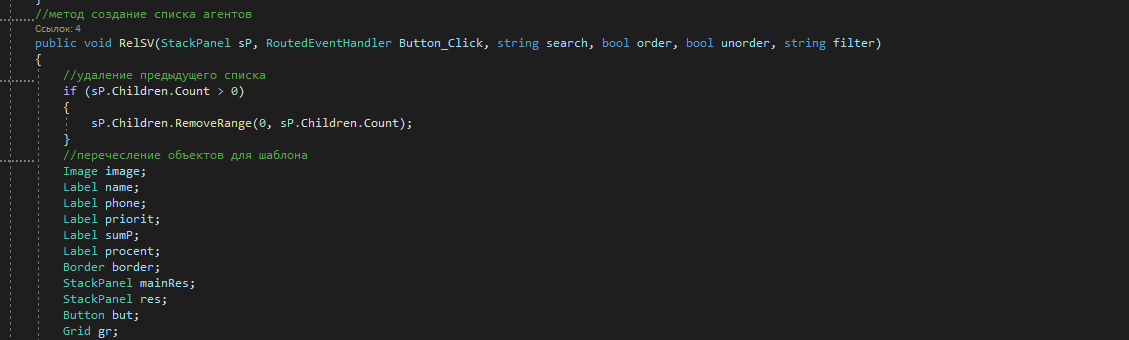


Рисунок 7 – создание элементов шаблона

Заполнение элементов шаблона данными из таблиц

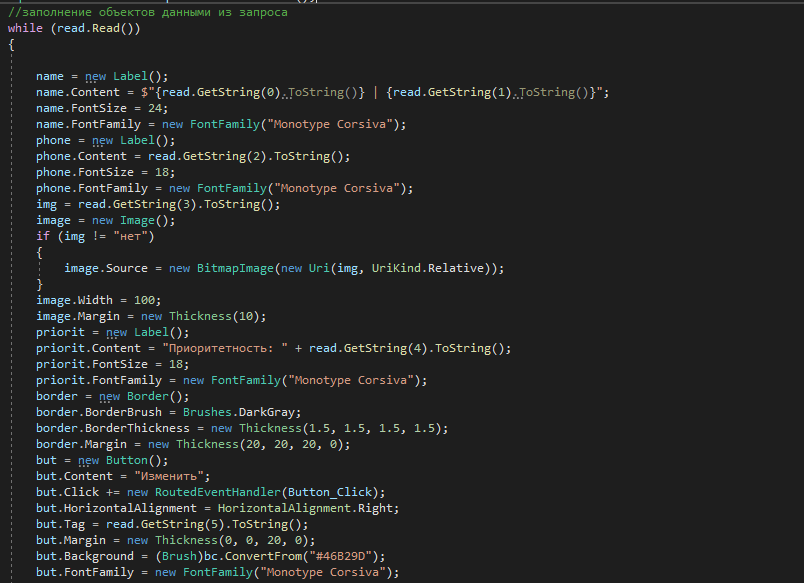
****

Рисунок 8 – заполнение элементов

Создание заготовок запросов с сортировкой, с фильтрацией и поиском

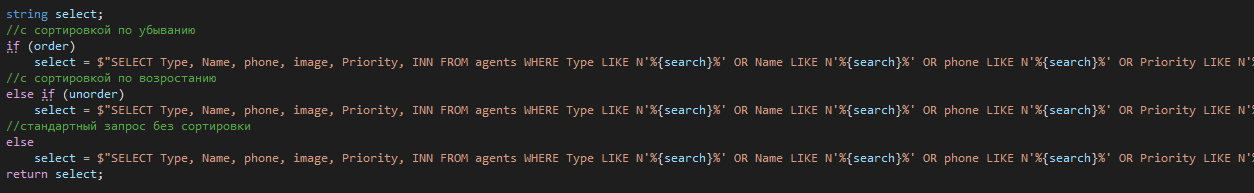
****

Рисунок 9 – создание запросов SELECT

Проверяю вывод агентов с поиском ПАО, с сортировкой по убыванию и фильтром по имени

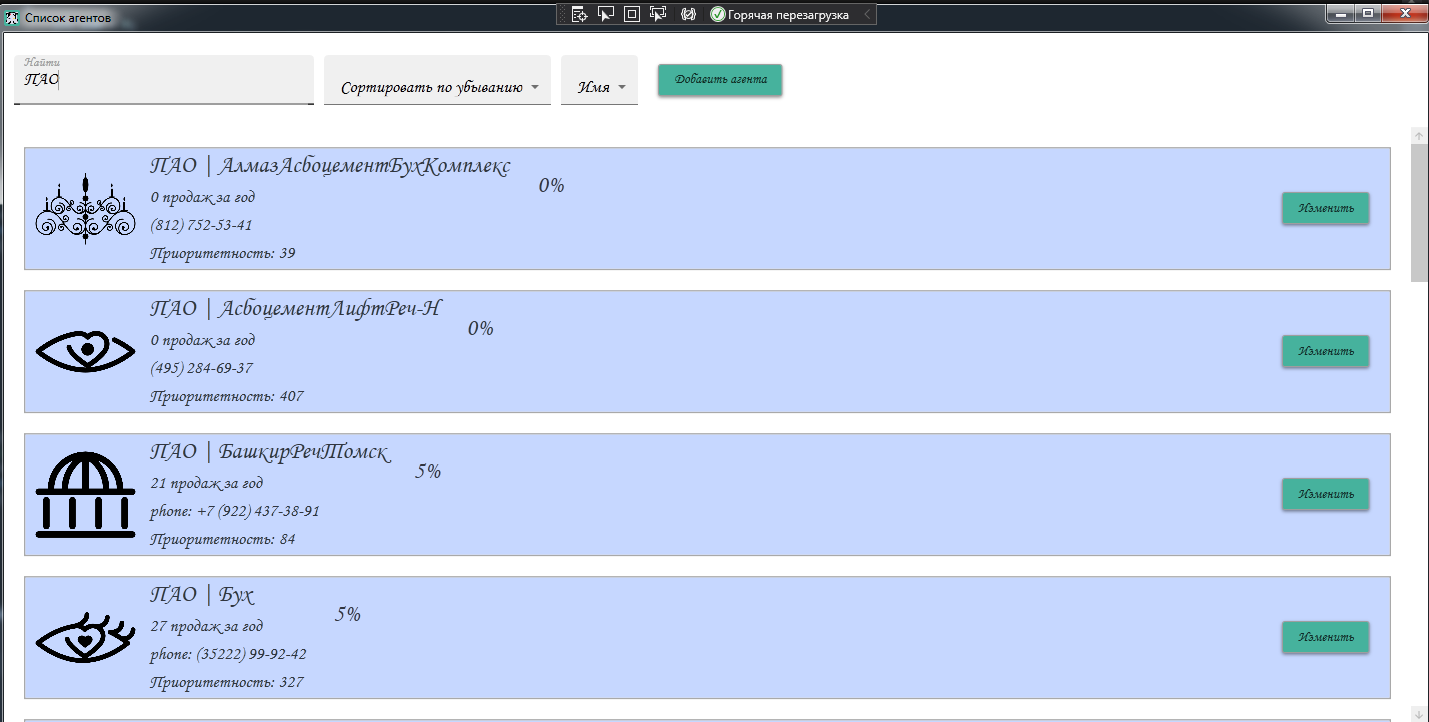
****

Рисунок 10 – вид списка агентов

**4 Разработка desktop-приложений. Добавление агентов**

Создаю метод для добавления агентов

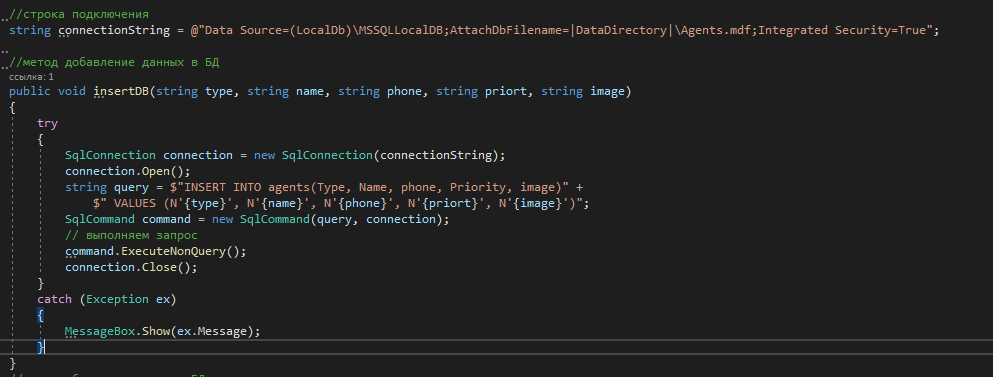


Рисунок 11 – создание запроса заполнения

Создаю обработчик по нажатию кнопки добавить

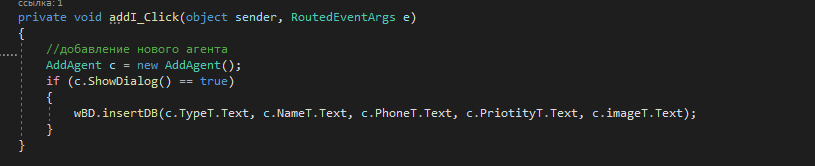
****

Рисунок 12 – обработка добавления

Создаю форму заполнения и заполняю форму

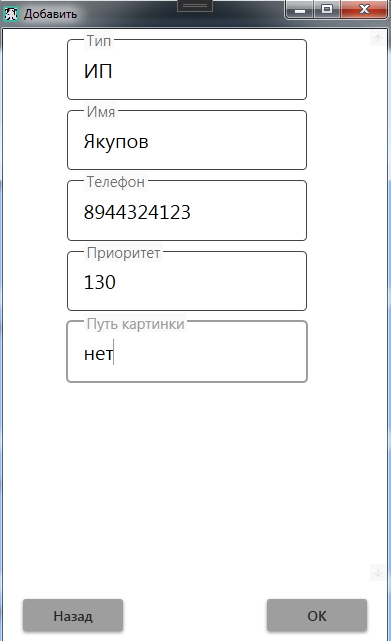
****

Рисунок 13 –форма для добавления

Выполняю поиск созданного агента

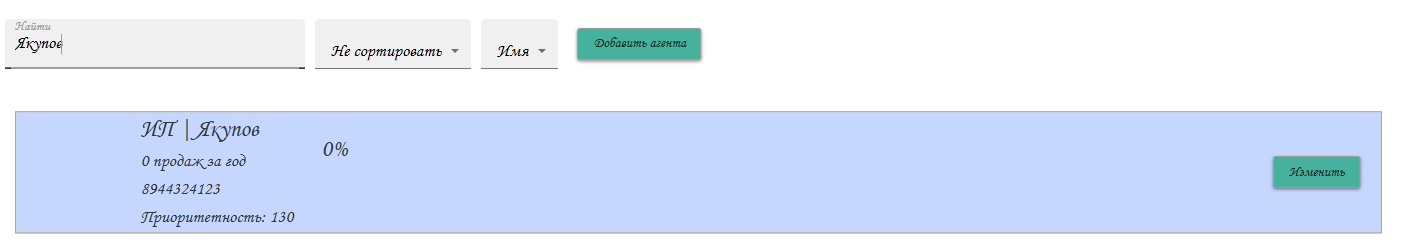
****

Рисунок 14 – добавленный агент

**5 Разработка desktop-приложений. Редактирование агентов**

Создаю метод обновления агента

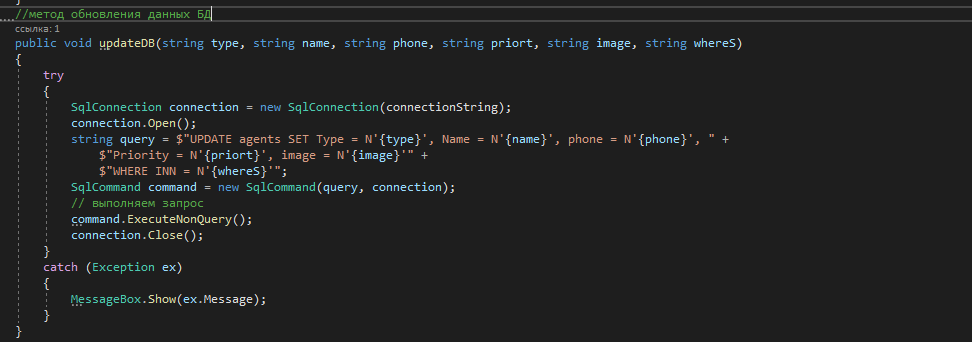
****

Рисунок 15 – запрос на обновление агента

Создаю конструктор для формы изменения агента

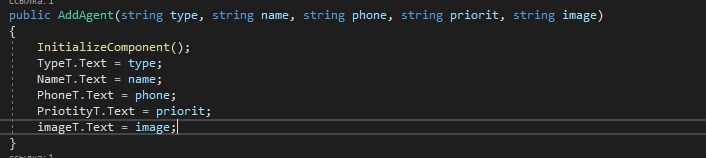
****

Рисунок 16 – конструктор для обновления

Создаю метод для заполнения полей выбранного агента

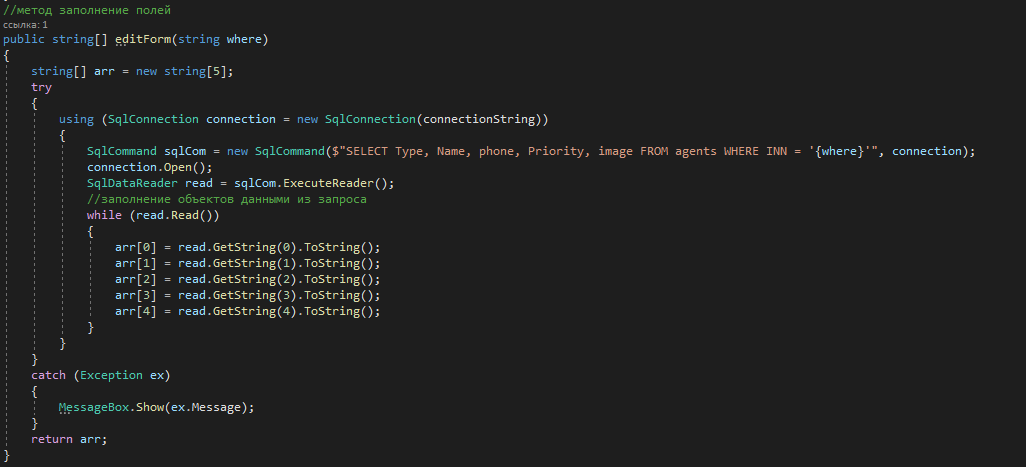
****

Рисунок 17 – функция поиска данных для обновления

Добавляю в шаблон кнопку для каждого агента

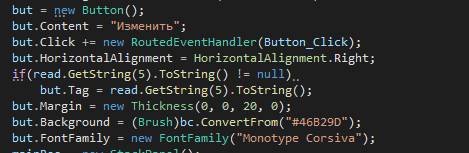
****

Рисунок 18 – кнопка изменения для шаблона

Делаю обработчик нажатия на кнопку изменить

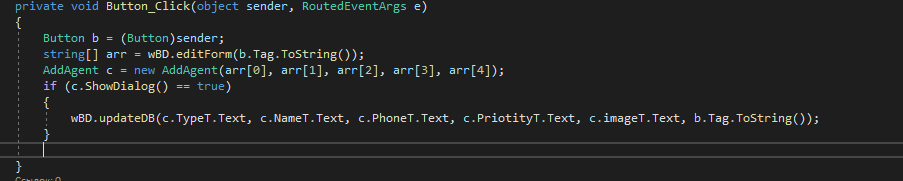
****

Рисунок 19 – обработка обновления

Изменяю выбранного агента

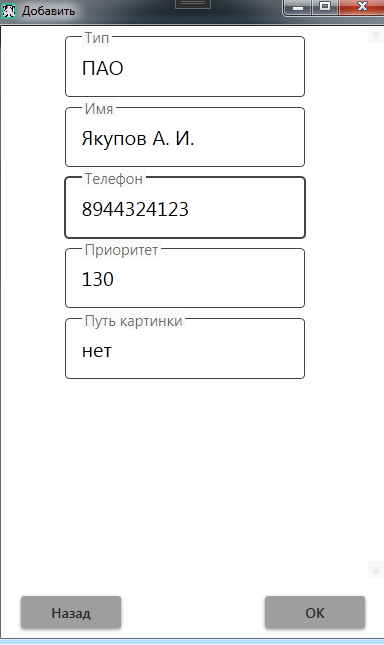


Рисунок 20 – форма обновления агента

Выполняю поиск для просмотра изменений

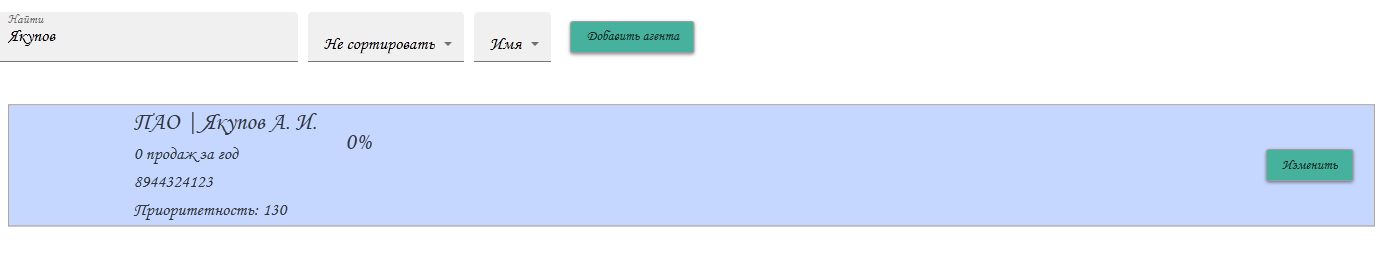
****

Рисунок 21 – измененный агент

**6Изучение и описание современных методов разработки программного обеспечения**

|  |  |
| --- | --- |
| **Технология** | **Описание** |
| WPF | Аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая подсистема в составе .NET Framework, использующая язык XAML. |
| Расширение SQL Server для Visual Studio | Расширение mssql для Visual Studio Code позволяет подключаться к SQL Server, выполнять запросы с помощью Transact-SQL (T-SQL) и просматривать результаты. Создание или открытие файла SQL. Чтобы выполнять команды mssql и пользоваться технологией IntelliSense для T-SQL в редакторе кода, необходимо выбрать языковой режим SQL. |

**7Применение стандартов кодирования при разработке приложения**

Использованные стандарты:

1. Ограниченное использование глобалов:  
   Эти правила говорят о том, какие типы данных могут быть объявлены глобальными, а какие - нет.
2. Стандартные заголовки для разных модулей:  
   Для лучшего понимания и поддержки кода заголовки различных модулей должны соответствовать некоторому стандартному формату и информации. Формат заголовка должен содержать следующие вещи, которые используются в различных компаниях:
   * название модуля;
   * дата создания модуля;
   * автор модуля;
   * история изменений;
   * краткое описание модуля о том, что модуль делает;
   * различные функции, поддерживаемые в модуле вместе с их входными и выходными параметрами;
   * глобальные переменные, к которым осуществляется доступ или которые изменяются модулем.
3. Соглашения об именах для локальных переменных, глобальных переменных, констант и функций:  
   Некоторые соглашения об именах приведены ниже:
   * осмысленное и понятное название переменных помогает понять причину их использования;
   * локальные переменные должны быть названы с использованием букв верблюжьего регистра, начиная с маленькой буквы (например, localData ), тогда как имена глобальных переменных должны начинаться с заглавной буквы (например, GlobalData ). Постоянные имена должны быть сформированы только заглавными буквами (например, CONSDATA );
   * лучше избегать использования цифр в именах переменных;
   * названия функций следует писать верблюжьим регистром, начиная с маленьких букв;
   * название функции должно четко и кратко описывать причину использования функции.
4. Отступ:  
   Правильный отступ очень важен для повышения читабельности кода. Чтобы код был читабельным, программисты должны правильно использовать белые пробелы. Некоторые условные обозначения интервалов приведены ниже:
   * после запятой между двумя аргументами функции должен быть пробел;
   * каждый вложенный блок должен иметь правильный отступ и интервал;
   * правильный отступ должен быть в начале и в конце каждого блока программы;
   * все фигурные скобки должны начинаться с новой строки, а код, следующий за концом фигурных скобок, также начинается с новой строки.
5. Возвращаемые значения ошибок и соглашения об обработке исключений:  
   Все функции, которые сталкиваются с ошибкой, должны возвращать 0 или 1 для упрощения отладки.
6. Избегайте использования стиля кодирования, который слишком сложен для понимания:  
   Код должен быть легко понятным. Сложный код делает обслуживание и отладку сложными и дорогостоящими.
7. Избегайте использования идентификатора для нескольких целей:  
   Каждой переменной следует дать описательное и содержательное имя, указывающее на причину ее использования. Это невозможно, если идентификатор используется для нескольких целей и, таким образом, может ввести читателя в заблуждение. Более того, это усложняет будущие улучшения.
8. Код должен быть хорошо документирован:  
   Код должен быть правильно прокомментирован для облегчения понимания. Комментарии к утверждениям повышают понятность кода.
9. Длина функций не должна быть очень большой:  
   Длинные функции очень трудно понять. Вот почему функции должны быть достаточно маленькими, чтобы выполнять небольшую работу, а длинные функции должны быть разбиты на мелкие для выполнения небольших задач.

**8Применение систем контроля версий**

В проекта использовалась система контроля версий GitHub

GitHub – это веб-интерфейс, в котором можно хранить свои репозитории Git, а также эффективно отслеживать и управлять своими изменениями. С его помощью разные разработчики имеют доступ к коду одного проекта. У вас есть возможность вносить свои собственные изменения в проект одновременно с другими разработчиками.

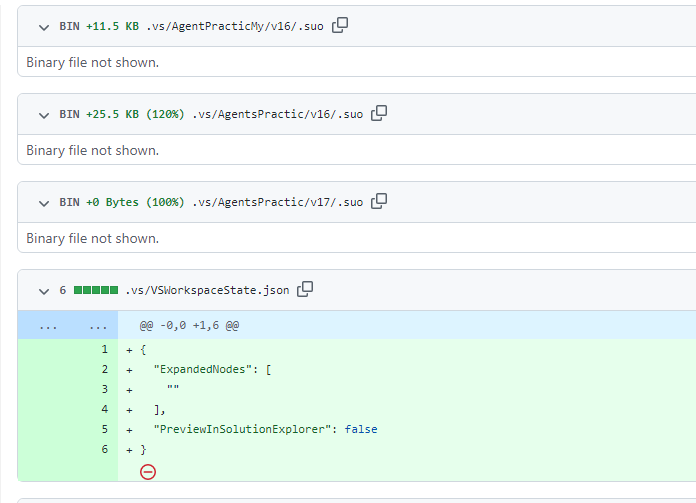
****

Рисунок 22 – использование GitHub

**Приложение А**