МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

Кафедра систем управления и информационных технологий в строительстве

КУРСОВОЙ ПРОЕКТ

по дисциплине: «Объектно-ориентированное программирование»
на тему: «Разработка приложения для записи на приём в медицинское учреждение»

Выполнил: студент группы бПИЭ-202

_____ Кожина А.Г.
(подпись)

Руководитель: д.т.н., проф. Кононов А.А.

Работа защищена « ___ »_____ ____ г.

с оценкой _____ _____
(подпись)

Члены комиссии _____
Подпись, дата Инициалы, фамилия

_____
Подпись, дата Инициалы, фамилия

Нормоконтролер _____
Подпись, дата Инициалы, фамилия

2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГТУ», ВГТУ)

Кафедра систем управления и информационных технологий в строительстве

ЗАДАНИЕ
на курсовой проект

по дисциплине: «Объектно-ориентированное программирование»
Тема: «Разработка приложения для записи на приём в медицинское учреждение»

Студент бПИЭ-202, Кожина Анастасия Геннадьевна
Группа, фамилия, имя, отчество

Вариант _____

Технические условия операционная система Windows 10, ОЗУ 2048 МБ

Сроки выполнения этапов: анализ и постановка задачи (15.09-10.10);
изучение теоретического обоснования работы (10.10-10.11);
реализация программного решения (10.11-10.12);
оформление пояснительной записки (10.12-20.12).

Срок защиты курсового проекта: декабрь 2022 года

Руководитель _____ А.А. Кононов
Подпись, дата Инициалы, фамилия

Задание принял студент _____ А.Г. Кожина
Подпись, дата Инициалы, фамилия

СОДЕРЖАЕНИЕ

ВВЕДЕНИЕ 3
1. Описание предметной области 5
2. Постановка задачи 6
3. Выбор языка программирования разработки 6
4. Выбор среды разработки 11
5. Проектирование структуры приложения 11
6. Разработка базы данных 13
7. Разработка приложения 15
8. Демонстрация разработанного приложения 23
ЗАКЛЮЧЕНИЕ 25
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 26
ПРИЛОЖЕНИЕ А 28

ВВЕДЕНИЕ

Объектно-ориентированное программирование (сокр. ООП) — методология программирования, основанная на представлении программы в виде совокупности взаимодействующих объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

C# — объектно-ориентированный язык программирования (ООП). Всё взаимодействие в нём происходит через объекты. Это в целом похоже на то, что творится в реальном мире. Все эти сущности описывают в коде и учат взаимодействовать друг с другом. В итоге программа в стиле ООП состоит из отдельных блоков, которые хорошо расширяются и масштабируются. Поэтому язык C# подходит для разработки программ, которые планируют долго использовать и постоянно развивать.

C# берёт лучшее из компилируемых и интерпретируемых языков. Чтобы разобраться в этом свойстве, нужно шагнуть ещё немного назад. Язык программирования — это язык, на котором программист и процессор договариваются, как выполнять команды. Так вот процессор не полиглот и не обязан знать все языки, на которых им хотят покомандовать. Поэтому язык программирования нужно переводить на язык процессора. Делается это двумя способами — интерпретированием и компилированием.

Целью данной курсовой работы является разработка программы с использованием принципов объектно-ориентированного программирования на ООП языке – C# средствами среды разработки Visual Studio.

Задачами данной курсовой работы являются:

— Выделение классов, необходимых для решения задачи;
— Выделение основного действия в задаче и построение алгоритма его реализации;

— Определение критерий поиска, применение которого необходимо в задаче.

1. Описание предметной области

Концепции ООП являются основополагающими элементами и составляют основу языка программирования C#. В рамках данного подхода выделяют следующие термины: абстракция, инкапсуляция, наследование и полиморфизм. Понимание данных принципов служит ключом к построению целостной картины того, как работают программы, написанные на C#. По большому счету, объектно-ориентированный подход позволяет нам описывать классы, определять методы и переменные таким образом, чтобы затем использовать их вновь, частично либо полностью, без нарушения безопасности.

Абстракция. Абстракция означает использование простых вещей для описания чего-то сложного. В C# под абстракцией подразумеваются такие вещи, как объекты, классы и переменные, которые в свою очередь лежат в основе более сложного кода. Использование данного принципа позволяет избежать сложности при разработке ПО.

Инкапсуляция. Под инкапсуляцией подразумевается сокрытие полей внутри объекта с целью защиты данных от внешнего, бесконтрольного изменения со стороны других объектов. Доступ к данным (полям) предоставляется посредством публичных методов (геттеров/сеттеров). Это защитный барьер позволяет хранить информацию в безопасности внутри объекта.

Наследование. Это особая функциональность в объектно-ориентированных языках программирования, которая позволяет описывать новые классы на основе уже существующих. При этом поля и методы класса-предка становятся доступны и классам-наследникам. Данная фича делает классы более чистыми и понятным за счет устранения дублирования программного кода.

Полиморфизм. Данный принцип позволяет программистам использовать одни и те же термины для описания различного поведения, зависящего от контекста. Одной из форм полиморфизма в C# является переопределение метода, когда различные формы поведения определяются объектом, из которого

данный метод был вызван. Другой формой полиморфизма является перегрузка метода, когда его поведение определяется набором передаваемых в метод аргументов.

2. Постановка задачи

В рамках данной курсовой работы требуется разработать приложение, программный код которого будет основываться на парадигме объектно-ориентированного программирования и его основных принципах: наследовании, инкапсуляции, полиморфизме, абстракции.

Основная задача проектируемого приложения заключается в организации записи к врачам-специалистам в соответствии с запрошенными критериями и пожеланиями во времени

Программа должна быть отказоустойчивой, то есть корректно реагировать и обрабатывать ошибки, которые могут возникнуть в ходе использования приложения: некорректно введенные данные, деление на ноль, ошибка чтения системных файлов и так далее.

Разрабатываемое приложение должно иметь простой и органичный интерфейс, предоставлять своим пользователям доступный и отчётливый результат своего использования

3. Выбор языка программирования разработки

Созданное приложение должно полностью соответствующего поставленным требованием и задачам, в следствии чего при выборе языка разработке должно учитываться множество критериев.

К наиболее популярным на данный момент языкам программирования, реализующим принципы ООП, можно отнести:

— C++

— Java

— C#

C++

Достоинства языка:

1. Масштабируемость. На языке C++ разрабатывают программы для самых различных платформ и систем.
2. Возможность работы на низком уровне с памятью, адресами, портами. Что, при неосторожном использовании, может легко превратиться в недостаток.
3. Возможность создания обобщенных алгоритмов для разных типов данных, их специализация, и вычисления на этапе компиляции, используя шаблоны.

Недостатки языка:

1. Наличие множества возможностей, нарушающих принципы типобезопасности приводит к тому, что в C++-программы может легко закрасться трудноуловимая ошибка. Вместо контроля со стороны компилятора разработчики вынуждены придерживаться весьма нетривиальных правил кодирования. По сути эти правила ограничивают C++ рамками некоего более безопасного подъязыка. Большинство проблем типобезопасности C++ унаследовано от C, но важную роль в этом вопросе играет и отказ автора языка от идеи использовать автоматическое управление памятью (например, сборку мусора). Так визитной карточкой C++ стали уязвимости типа "переполнение буфера".
2. Плохая поддержка модульности. Подключение интерфейса внешнего модуля через препроцессорную вставку заголовочного файла (#include) серьёзно замедляет компиляцию, при подключении большого количества модулей. Для устранения этого недостатка, многие компиляторы

реализуют механизм прекомпиляции заголовочных файлов Precompiled Headers.

3. Недостаток информации о типах данных во время компиляции (CTTI).
4. Язык C++ является сложным для изучения и для компиляции.
5. Некоторые преобразования типов неинтуитивны. В частности, операция над беззнаковым и знаковым числами выдаёт беззнаковый результат.

Java

Достоинства языка:

1. Независимый код. Любая платформа, которая поддерживает виртуальную машину Java, воспроизведет ваш код.
2. Надежный код. Строгая статистическая, типизация дает главное преимущество — надежность вашего кода.
3. Высокая функциональность. На Java можно написать практически все: от простого приложения на смартфон до программ по машинному обучению для беспилотных автомобилей.
4. Синтаксис средней сложности. Данный язык поддается изучению новичкам, которые раньше вообще не имели дела с программированием.
5. Java для Андроид. Андроид — самая популярная ОС для смартфонов, а Java — самый популярный язык для приложений на Андроид, соответственно, изучив Java, будет очень широкое поле для деятельности.

Недостатки языка:

1. Более низкая производительность. За счет свой специфики Java во многих случаях работает медленнее, чем другие языки, такие как: C, C#, C++, Python.
2. Потребляет память. Опять же, за счет своей специфики работы данный язык требует больше памяти, чем многие сторонние языки.

3. Платность. Буквально с 2019 года для коммерческо-юридических проектов язык Java стал платным, но для частного использования он абсолютно бесплатен.

C#

Достоинства языка:

1. Независимость от железа. Программисту не надо адаптировать программу под разные платформы и системы — за него это делает виртуальная машина, вшитая в .NET Framework. В итоге один и тот же код можно запускать на любых устройствах — смартфонах, компьютерах, серверах, банкоматах и даже умных часах.
2. Отличная совместимость с Windows. Не зря же язык разработали именно в Microsoft. Так же как Swift идеально подходит для программирования под экосистему Apple, C# прекрасно вписывается в экосистему Windows.
3. Управление памятью. Чтобы программа работала стабильно, её надо иногда чистить от ненужных объектов, ссылок, кэша и прочего мусора. В C# это происходит автоматически — разработчику не надо следить за расходом памяти, бороться с её утечками или удалять мёртвые куски кода.
4. Строгая типизация. Когда вы объявляете переменную в C#, надо сначала указать, что в ней лежит — строка, число или массив. Так разрабатывать чуть дольше, зато ваш код работает предсказуемо — числа взаимодействуют с числами, строки со строками и так далее. В языках со слабой типизацией свободы и драйва больше, но есть шанс пропустить ошибку, которая всплывёт в готовой программе.
5. Большое сообщество. На C# пишут более миллиона программистов по всему миру. В соцсетях полно чатов и сообществ, где можно задать вопрос, обсудить сложную тему или найти готовое решение. В теории

можно даже найти ментора, который поделится знаниями и поможет быстрее освоить язык.

6. Синтаксический сахар. В C# есть много способов сократить код, не нарушая логику программы. Программисты называют такие приёмы «синтаксическим сахаром» — они помогают сделать код проще, понятнее и в целом симпатичнее. Сравните, например, как выглядит сложение чисел с «сахаром» и без.

Недостатки языка:

1. Скорость. Когда мы запускаем программу на C#, код исполняется не сразу, а сначала адаптируется под нужное железо. Так мы охватываем больше платформ, но теряем в скорости — программе нужно сделать двойную работу, чтобы просто стартовать. Из-за этого интерфейсы на C# иногда подтормаживают при первом запуске.
2. Безопасность. Эксперты говорят, что код на C# легко декомпилировать — то есть перевести из машинного обратно в человеческий. Проблема в том, что так программу может легко прочитать хакер или конкурент — и изучить её уязвимости, украсть фрагменты кода или написать для неё вредоносный софт.
3. Мало доступа к железу. Так как C# — язык высокого уровня, на нём редко пишут проекты, где нужно полное взаимодействие с железом, — игровые движки, операционные системы, авиационный софт и так далее. Та же Unity целиком написана на низкоуровневом языке C++, хотя и умеет исполнять C#-команды.

В ходе проведения анализа, учитывая все плюсы и минусы, среди представленных языков был выбран C#(.net) и фреймворк WinForms(.net 6). На данном языке программирования довольно просто реализовать принципы ООП, он безопасный, а также обеспечивает автоматическое управление памятью и обладает сборщиками мусора.

4. **Выбор среды разработки**

Для комфортного и органичного использования языка C# была выбрана среда разработки Visual Studio.

Visual Studio 2022 Community - лучшая комплексная среда IDE для разработчиков .NET и C++ в Windows. Полноценный набор инструментов и функций для улучшения и усовершенствования каждого этапа разработки программного обеспечения.

5. **Проектирование структуры приложения**

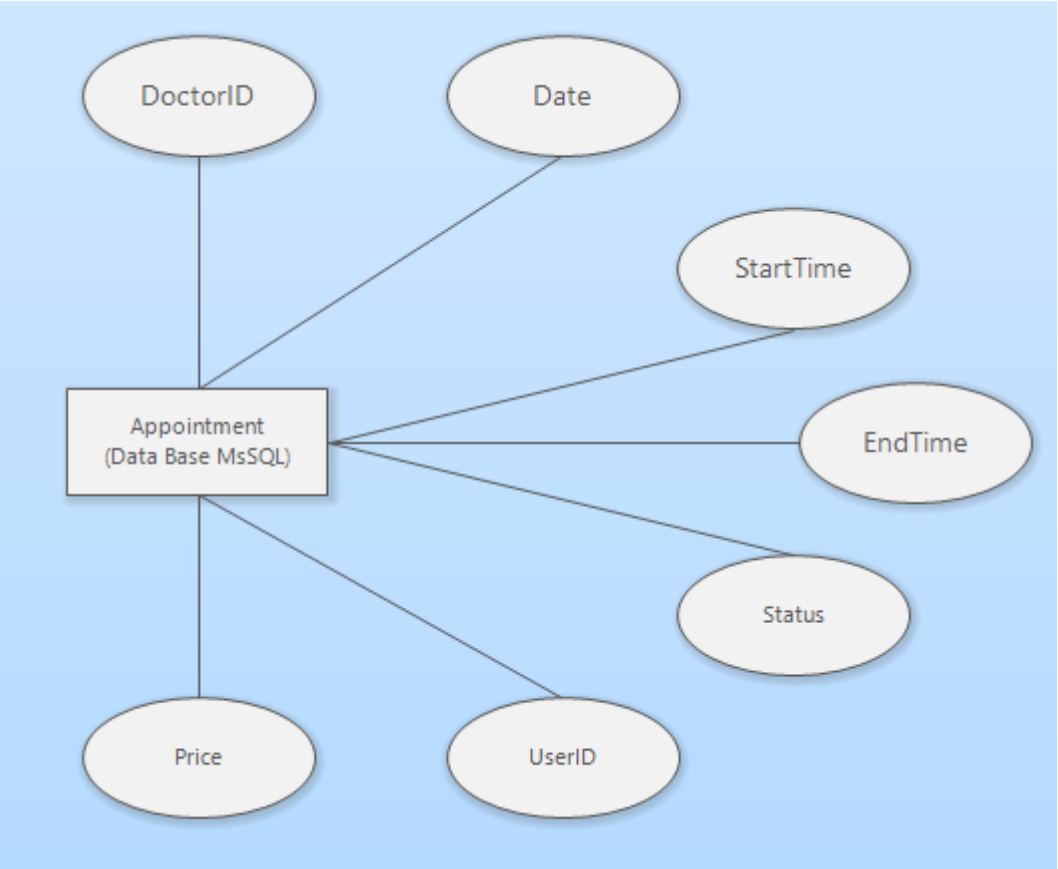

Рисунок 1 – Концептуальная модель класса Базы данных

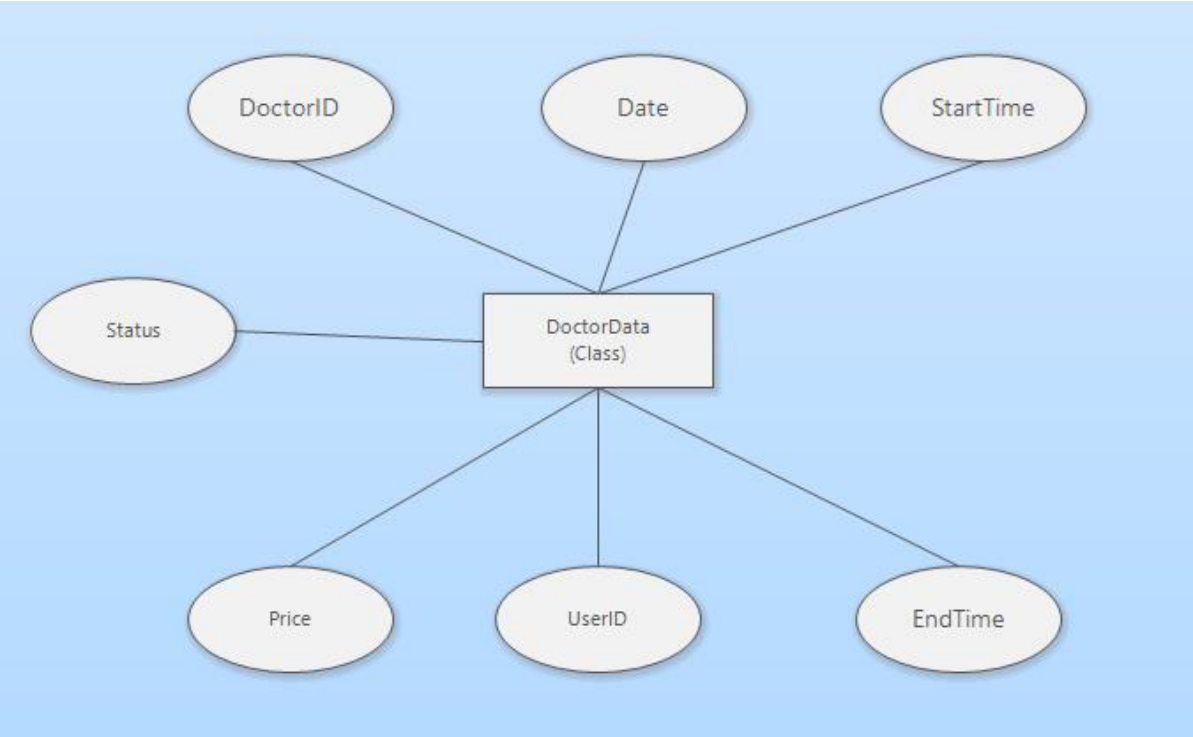

Рисунок 2 – Концептуальная модель класса DoctorData

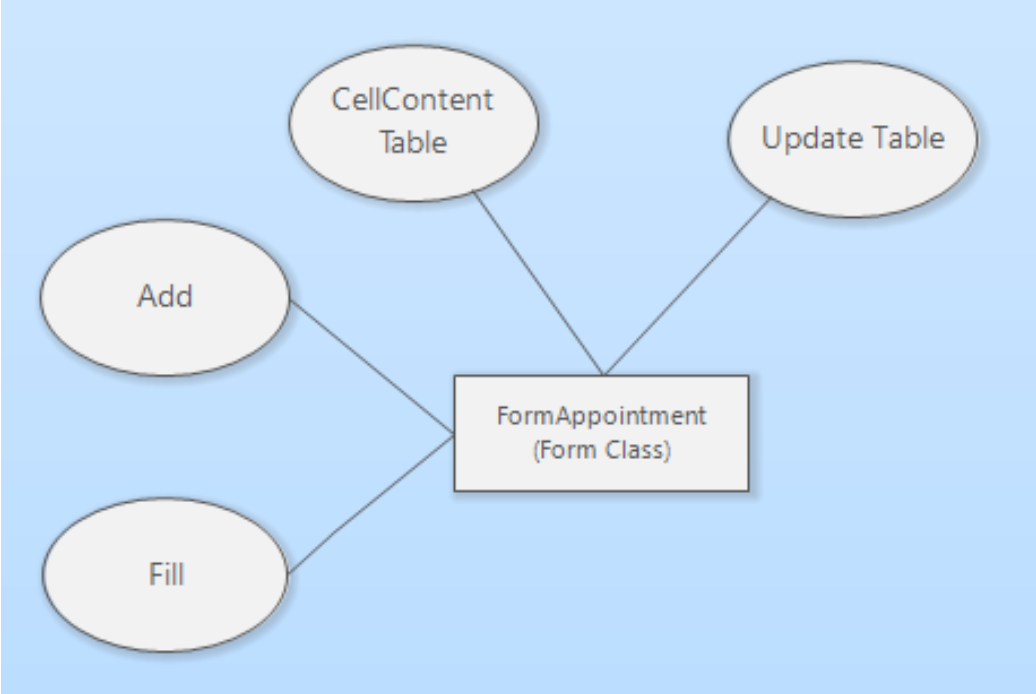

Рисунок 3 – Концептуальная модель класса FormAppointment

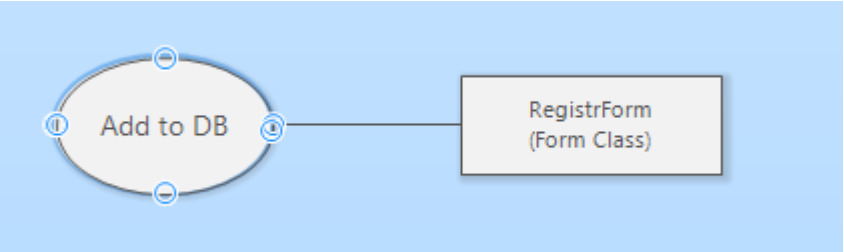

Рисунок 4 – Концептуальная модель класса RegistrForm

На рисунках 1-4 показана концептуальная часть разрабатываемой программы. На основе данной модели происходит создание логической модели. Концептуальная модель помогает определить сущности и их атрибуты.

6. Разработка базы данных

В разрабатываемом приложении необходима база данных, которая будет содержать информацию о докторах, клиентах и записях к врачам-специалистам.

Всё это можно записать в одной таблице под названием Appointment. Так же понадобится дополнительная таблица для генерации времени при записи - Numbers

В таблице Appointment создаются поля (DoctorID, [Date], StartTime, EndTime, Status, UserID, Pric).

```
CREATE TABLE Appointment (
    DoctorID char(1) NOT NULL,
    [Date] date NOT NULL,
    StartTime time(0) NOT NULL CONSTRAINT CHK_StartTime_TenMinute
                               CHECK (DATEPART(MINUTE, StartTime)%10 = 0 AND DATEPART(SECOND, StartTime) = 0),
    EndTime time(0) NOT NULL CONSTRAINT CHK_EndTime_TenMinute
                               CHECK (DATEPART(MINUTE, EndTime)%10 = 0 AND DATEPART(SECOND, EndTime) = 0),
    Status char(1) NOT NULL,
    UserID char(1) NOT NULL,
    Price int NOT NULL,
    CONSTRAINT PK_Appointment PRIMARY KEY CLUSTERED (DoctorID, [Date], StartTime),
    CONSTRAINT CHK_StartTime_BusinessHours
                CHECK (DATEPART(HOUR, StartTime) > = 9 AND DATEPART(HOUR, StartTime) < = 16),
    CONSTRAINT CHK_EndTime_BusinessHours
                CHECK (DATEPART(HOUR, EndTime) > = 9 AND DATEPART(HOUR, DATEADD(SECOND, -1, EndTime)) < = 16),
    CONSTRAINT CHK_EndTime_After_StartTime
                CHECK (EndTime > StartTime));
CREATE INDEX iDoctor_End ON Appointment (DoctorID, [Date], EndTime);
```

Рисунок 5 – Реализация таблицы Appointment

В таблице Numbers создаётся поле – Numbe, что продемонстрировано на рисунке 6.

```
CREATE TABLE Numbers (Number int PRIMARY KEY CLUSTERED);
DECLARE @number int = 0;
WHILE @number < 1000
BEGIN
    INSERT INTO Numbers VALUES (@number);
    SET @number += 1;
END
```

Рисунок 6 – Реализация таблицы Number

Так же для функционирования приложения нужно разработать sql-скрипт, который будет генерировать свободные даты для записи к врачам в необходимом количестве. Необходимо предусмотреть, чтобы пользователю не отображалось время, которое уже прошло, и чтобы нельзя было записаться на одну и туже дату дважды.

Для этого создаётся скрипт, показанный на рисунке 7.

```
DECLARE @doctorID char(1) = 'A';
DECLARE @length tinyint = 20;
WITH Slots AS (
    SELECT StartTime =
    DATEADD(MINUTE, ((DATEPART(MINUTE, GETDATE())/10)+1+Number)*10,
    DATEADD(HOUR, DATEPART(HOUR, GETDATE()), CONVERT(smalldatetime, CONVERT(date, GETDATE()))))),
           EndTime =
           DATEADD(MINUTE, @length, DATEADD(MINUTE, ((DATEPART(MINUTE, GETDATE())/10)+1+Number)*10,
           DATEADD(HOUR, DATEPART(HOUR, GETDATE()), CONVERT(smalldatetime, CONVERT(date, GETDATE()))))))
      FROM Numbers)
SELECT TOP 15 DoctorID = @doctorID,
    s.StartTime,
    s.EndTime
  FROM Slots AS s
  WHERE NOT EXISTS (SELECT 1
                    FROM Appointment AS a
                    WHERE (CONVERT(time(0), s.StartTime) < a.EndTime AND CONVERT(time(0), s.EndTime) > a.StartTime)
                      AND a.DoctorID = @doctorID
                      AND a.[Date] = CONVERT(date, s.StartTime))
    AND DATEPART(HOUR, s.StartTime) >= 9
    AND DATEPART(HOUR, DATEADD(MINUTE, -1, s.EndTime)) <= 16
ORDER BY s.StartTime;
```

Рисунок 7 – Скрип для генерации времени записи к врачу

Также необходим скрип, который будет вставлять данные в таблицу Appointment.

```
INSERT INTO Appointment VALUES
('A', '20221217', '09:00:00', '09:10:00', 'P', '1', '1500');
```

Рисунок 8 – Скрип для вставки данных в таблицу Appointment

7. Разработка приложения

Сначала реализуется класс Program, используемый для запуска программы.

```
1   using WinFormsApp2.View;
2
3   namespace WinFormsApp2
4   {
        Ссылок: 0
5       internal static class Program
6       {
7           /// <summary>
8           ///  The main entry point for the application.
9           /// </summary>
10          [STAThread]
            Ссылок: 0
11          static void Main()
12          {
13              // To customize application configuration such as set high DPI settings or default font,
14              // see https://aka.ms/applicationconfiguration.
15              ApplicationConfiguration.Initialize();
16              Application.Run(new FormAppointment());
17          }
18      }
19  }
```

Рисунок 5 – Реализация класса Program

```
Ссылок: 2
public class DoctorData
{
    public string DoctorID = ""; // clickCellTableDoctorID
    public DateTime Date;  // dt
    public string StartTime = "";  //  dccstart
    public string EndTime = "";  //dccend
    public char Status = ' ';  // status
    public char UserID = ' ';  //// i
    public int Price; // prise

    // command.Parameters.AddWithValue("@DoctorID", clickCellTableDoctorID);
    // command.Parameters.AddWithValue("@Date", dt);
    // command.Parameters.AddWithValue("@StartTime", dccstart);
    // command.Parameters.AddWithValue("@EndTime", dccend);
    // command.Parameters.AddWithValue("@Status", status);
    // command.Parameters.AddWithValue("@UserID", i);
    // command.Parameters.AddWithValue("@Price", prise);
}
```

Рисунок 6 – Реализация класса DoctorData

Данный класс будет реализовывать поля базы данных.

```
Ссылок: 5
public partial class FormAppointment : Form
{
    private string connectDB =
        @"Data Source=DESKTOP-432U1GM\\SQLEXPRESS;Initial Catalog=Kojina;Integrated
        Security=True;MultipleActiveResultSets=True";

    private string clickCellTableStart = "";
    private string clickCellTableEnd = "";
    private string clickCellTableDoctorID = "";
    private string clickCellTableUserID = "";

    Ссылок: 2
    public FormAppointment()
    {
        InitializeComponent();

        comboBox2.Items.Add("5");
        comboBox2.Items.Add("10");
        comboBox2.Items.Add("20");
        comboBox2.Items.Add("30");
        comboBox2.Items.Add("40");
        comboBox2.Items.Add("50");
        comboBox2.SelectedIndex = 0;
    }
```

Рисунок 7 – Реализация класса FormAppointment

На рисунке 7 создаются приватные переменные для внутренней работы с данными, которые в дальнейшем получается из базы данных. Так же в конструкторе класса мы задаются значения по умолчанию для comboBox2, с помощью которого будет регулироваться количество отображаемых записей к определённому врачу.

```
Ссылок: 2
private void FormAppointment_Load(object sender, EventArgs e)
{
    string squery = @$"
        DECLARE @doctorID char(1) = '{comboBox1.Text}';
        DECLARE @length tinyint = 20;
        WITH Slots AS (
            SELECT StartTime = DATEADD(MINUTE, ((DATEPART(MINUTE, GETDATE())/10)+1+Number)*10, DATEADD(HOUR, DATEPART(HOUR, G
                   EndTime = DATEADD(MINUTE, @length, DATEADD(MINUTE, ((DATEPART(MINUTE, GETDATE())/10)+1+Number)*10, DATEADD
              FROM Numbers)
        SELECT TOP {comboBox2.Text} DoctorID  = @doctorID,
            s.StartTime as 'Начало приёма',
            s.EndTime as 'Конец приёма'
          FROM Slots AS s
          WHERE NOT EXISTS (SELECT 1
                            FROM Appointment AS a
                            WHERE (CONVERT(time(0), s.StartTime) < a.EndTime AND CONVERT(time(0), s.EndTime) > a.StartTime)
                              AND a.DoctorID = @doctorID
                              AND a.[Date] = CONVERT(date, s.StartTime))
            AND DATEPART(HOUR, s.StartTime) >= 9
            AND DATEPART(HOUR, DATEADD(MINUTE, -1, s.EndTime)) <= 16
        ORDER BY s.StartTime;
        ";
    try
    {
        SqlConnection sconn = new SqlConnection(connectDB);
        SqlCommand scomm = new SqlCommand(squery, sconn);
        DataTable stable = new DataTable();
        SqlDataAdapter sadapter = new SqlDataAdapter(scomm);
        sadapter.Fill(stable);
        dataGridView1.DataSource = stable;
        //dataGridView1.Columns[0].Visible = false;
        dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
        dataGridView1.Columns[1].DefaultCellStyle.BackColor = Color.Green;
        dataGridView1.Columns[2].DefaultCellStyle.BackColor = Color.Green;
    }
```

Рисунок 8 – Реализация метода FormAppointment_Load

На рисунке 8 обозначается переменная, в которую вставляется sql-запрос, используемый для обряжения к БД в блоке try, результат запроса записывается в таблицу dataGridView1.

```
catch (Exception exception)
{
    MessageBox.Show(exception.Message, "|error", MessageBoxButtons.OK, MessageBoxIcon.Information);
    throw;
}

squery = @"select DISTINCT DoctorID from Appointment";
try
{
    SqlConnection sconn = new SqlConnection(connectDB);
    SqlCommand scomm = new SqlCommand(squery, sconn);
    DataTable stable = new DataTable();
    SqlDataAdapter sadapter = new SqlDataAdapter(scomm);
    sadapter.Fill(stable);

    comboBox1.DataSource = stable;
    comboBox1.DisplayMember = "DoctorID";
}
catch (Exception exception)
{
    MessageBox.Show(exception.Message, "|error", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Рисунок 9 – Реализация метода FormAppointment_Load

На рисунке 9 присваиваются значения sql-запроса, из которого достаются данные о докторах в comboBox1.

```
Ссылок: 1
private void button1_Click(object sender, EventArgs e)
{
    string squery = @"select DoctorID as 'id доктора', [Date] as 'Дата', StartTime as 'Начало приёма',
                        EndTime as 'Конец приёма', Status as 'Статус',
                        UserID as 'id пользователя', Price as 'Стоимость'
                        from Appointment";
    try
    {
        SqlConnection sconn = new SqlConnection(connectDB);
        SqlCommand scomm = new SqlCommand(squery, sconn);
        DataTable stable = new DataTable();
        SqlDataAdapter sadapter = new SqlDataAdapter(scomm);
        sadapter.Fill(stable);
        dataGridView1.DataSource = stable;
        //dataGridView1.Columns[0].Visible = false;
        dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
        dataGridView1.Columns[2].DefaultCellStyle.BackColor = Color.Green;
        dataGridView1.Columns[3].DefaultCellStyle.BackColor = Color.Green;
    }
    catch (Exception exception)
    {
        MessageBox.Show(exception.Message, "|error", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

Рисунок 10 – Реализация метода button1_Click

В методе button1_Click с помощью sql-запроса достаются данные о тех записях к врачам, которые уже есть, и отображаются в таблицу dataGridView.

```
Ссылок: 1
private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        if (dataGridView1.Columns[dataGridView1.CurrentCell.ColumnIndex].HeaderText.ToString() ==
            "Начало приёма")
        {
            clickCellTableStart = dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString();
            clickCellTableEnd = dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex + 1].Value.ToString();
            clickCellTableDoctorID = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();

            textBox1.Text += "S| " + dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString()
                                  + dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex + 1].Value.ToString()
                                  + dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString()
                                  + "\r \n";
        }
        else if (dataGridView1.Columns[dataGridView1.CurrentCell.ColumnIndex].HeaderText.ToString() ==
                 "Конец приёма")
        {
            clickCellTableEnd = dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString();
            clickCellTableStart = dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex - 1].Value.ToString();
            clickCellTableDoctorID = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();

            textBox1.Text += "E| " + dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex - 1].Value.ToString()
                                  + dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString()
                                  + dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString()
                                  + "\r \n";
        }
        else{}
    }
    catch (Exception exception)
    {
        MessageBox.Show("cell content exp | " + exception);
        throw;
    }
}
```

Рисунок 11 – Реализация метода button1_Click

В методе button1_Click dataGridView1_CellContentClick реализуется функционал выбора конкретного врача-специалиста путём нажатия на свободное время его записи.

```
Ссылок: 1
private void button2_Click(object sender, EventArgs e)
{
    if (clickCellTableStart != null)
    {
        textBox1.Text += "BTN| " + clickCellTableStart + " | " + clickCellTableEnd + "\r \n";
        RegistrForm registrForm =
            new RegistrForm(clickCellTableStart, clickCellTableEnd, clickCellTableDoctorID);
        registrForm.ShowDialog();
    }
    else
    {
        textBox1.Text += "NULL \r";
    }
}

Ссылок: 1
private void button3_Click(object sender, EventArgs e)
{
    FormAppointment_Load(sender, e);
}
```

Рисунок 12 – реализация метода button1_Click2 и button1_Click3

В методе button1_Click2 реализуется событие нажатия на кнопку записи к врачу и открытия окна для подтверждения записи.

В методе button1_Click3 предусмотрена возможность для обновления данных таблицы dataGridView1.

```
Ссылок: 4
public partial class RegistrForm : Form
{
    private string connectDB =
        @"Data Source=DESKTOP-432U1GM\\SQLEXPRESS;Initial Catalog=Kojina;Integrated Security=True;MultipleActiveResultSets=True";

    private DoctorData doctorData = new DoctorData();

    private string clickCellTableStart = "";
    private string clickCellTableEnd = "";
    private string clickCellTableDoctorID = "";
    private string datecc = "";
    private string dccstart = "";
    private string dccend = "";
    private DateTime dt;

    Ссылок: 1
    public RegistrForm(string cCTS, string cCTE, string cCTDid)
    {
        InitializeComponent();

        clickCellTableStart = cCTS;
        clickCellTableEnd = cCTE;
        clickCellTableDoctorID = cCTDid;

        string[] ccstart = clickCellTableStart.Split(new char[] { ' ' }); // 0 - date / 1 - timestart
        string[] ccend = clickCellTableEnd.Split(new char[] { ' ' }); // 0 - date / 1 - timeend
        dccstart = ccstart[1];
        dccend = ccend[1];

        var test = ccstart[0].Split(new char[] { '.' });
        datecc = test[2] + "-" + test[1] + "-" + test[0];
```

Рисунок 13 – Реализация класса RegistrForm

На рисунке 13 создаются приватные переменные для внутренней работы с данными записи, а также обозначается экземпляр класса DoctorData.

В конструкторе класса полученные из другого класса переменные присваиваются к глобальным приватным, а также преобразовываются данные для дальнейшего отправления их в виде запроса в БД.

```
try
{ // "2022-12-17 14:40:52", "yyyy-MM-dd HH:mm:ss"
    dt = DateTime.ParseExact( datecc + " 14:40:52", "yyyy-MM-dd HH:mm:ss",
        System.Globalization.CultureInfo.InvariantCulture);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "error", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

label1.Text += " " + ccstart[1]+ " | " + datecc;
label2.Text += clickCellTableDoctorID;

toolTip1.SetToolTip(comboBox1, @"Выбирите цену в зависимости от ваших условий
                                        *Взрослый - 1500
                                        *Детский - 1000
                                        *Для пожмлых - 1200");
                                        // id 1500 => 'S'(standatr), 'C'(children), 'P'(Old)
comboBox1.Items.Add("Взрослый - 1500");
comboBox1.Items.Add("Детский - 1000");
comboBox1.Items.Add("Для пожмлых - 1200");
comboBox1.SelectedIndex = 0;
```

Рисунок 14 – Код конструктора класса RegistrForm

На рисунке 14 преобразовываются данные для дальнейшего отправления их в БД в виде запроса, задаются значения по умолчанию для comboBox1.

```
Ссылок: 1
private void button1_Click(object sender, EventArgs e)
{
    int i = 1;
    char status = 'S'; // id 1500 => 'S'(standatr), 'C'(children), 'P'(Old)
    int prise = 0;
    if (comboBox1.Text == "Детский - 1000")
    {
        status = 'C';
        prise = 1000;
    }
    else if (comboBox1.Text == "Для пожмлых - 1200")
    {
        status = 'P';
        prise = 1200;
    }
    else
    {
        status = 'S';
        prise = 1500;
    }
```

Рисунок 15 – Реализация метода button1_Click

```
string query = @"INSERT INTO Appointment
                (DoctorID, [Date], StartTime, EndTime, Status, UserID, Price) VALUES
                (@DoctorID, @Date, @StartTime, @EndTime, @Status, @UserID, @Price)";
try
{
    SqlConnection sconn = new SqlConnection(connectDB);
    sconn.Open();
    SqlCommand command = new SqlCommand(query, sconn);

    doctorData.DoctorID = clickCellTableDoctorID;
    doctorData.Date = dt;
    doctorData.StartTime = dccstart;
    doctorData.EndTime = dccend;
    doctorData.Status = status;
    doctorData.UserID = char.Parse(i.ToString());
    doctorData.Price = prise;

    command.Parameters.AddWithValue("@DoctorID", doctorData.DoctorID);
    command.Parameters.AddWithValue("@Date", doctorData.Date);
    command.Parameters.AddWithValue("@StartTime", doctorData.StartTime);
    command.Parameters.AddWithValue("@EndTime", doctorData.EndTime);
    command.Parameters.AddWithValue("@Status", doctorData.Status);
    command.Parameters.AddWithValue("@UserID", doctorData.UserID);
    command.Parameters.AddWithValue("@Price", doctorData.Price);

    int result = command.ExecuteNonQuery();

    if (result < 0)
        MessageBox.Show("Ошибка добавления строки в базу данных! " + result.ToString());

    sconn.Close();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "error", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Рисунок 16 – Реализация отправки данных в БД

На рисунке 16 создаётся sql-запрос в виде строки, параметрам запроса присваиваются данные из класса DortorData, и отправляется запрос к БД.

8. Демонстрация разработанного приложения

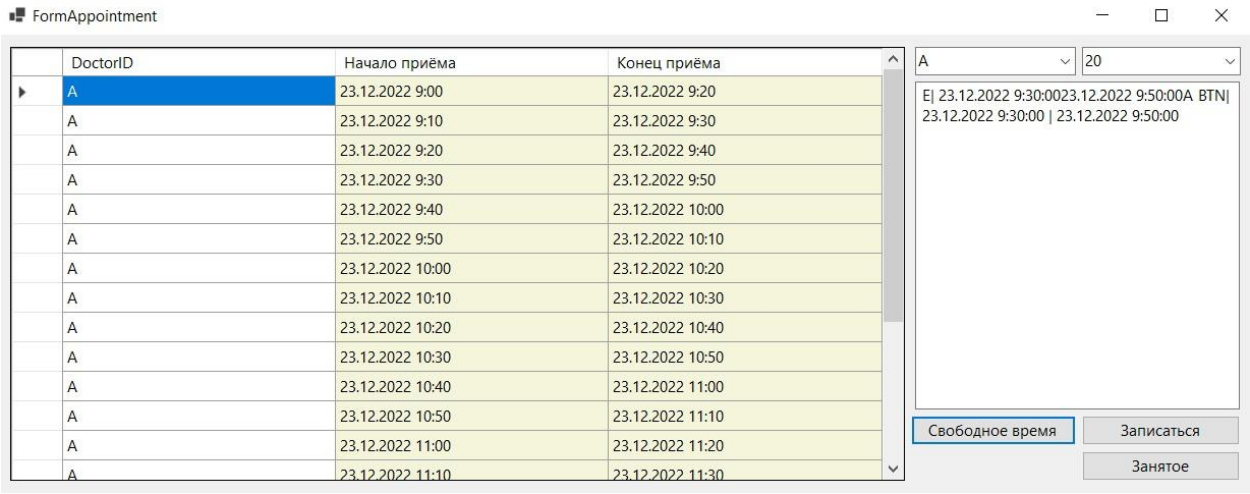

Рисунок 17 – Пример работы формы FormAppointment

На рисунке 17 показана работа начальной формы FormAppointment, на ней можно выбрать id врача, время записи, которую хочет посмотреть пользователь, предусмотрена возможность выбрать количество отображаемых записей.

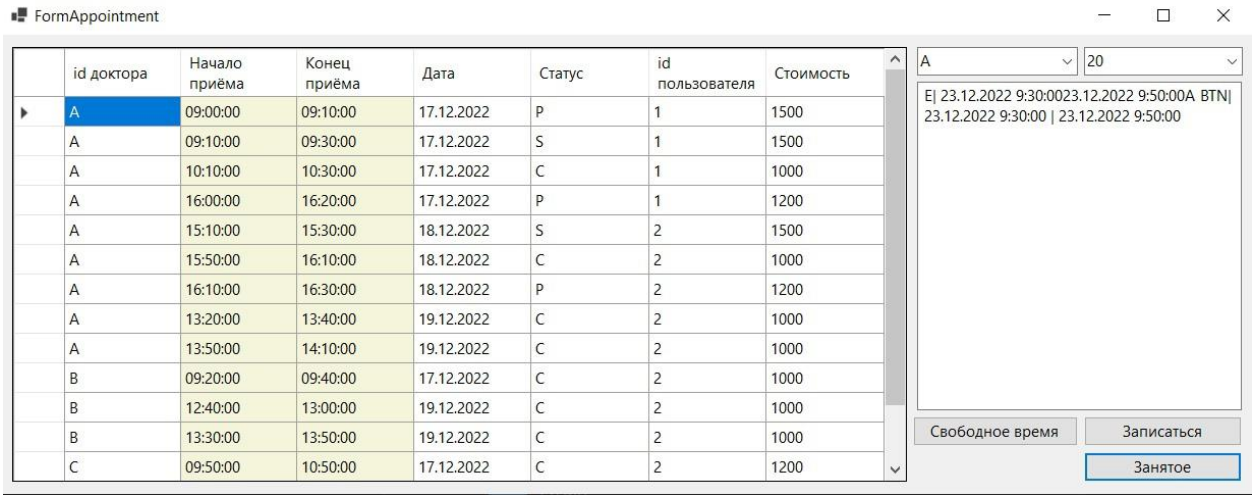

Рисунок 18 – Пример работы формы FormAppointment

На рисунке 18 отображены все записи, которые уже заняты, такой результат был получен нажатием кнопки «Занятое».

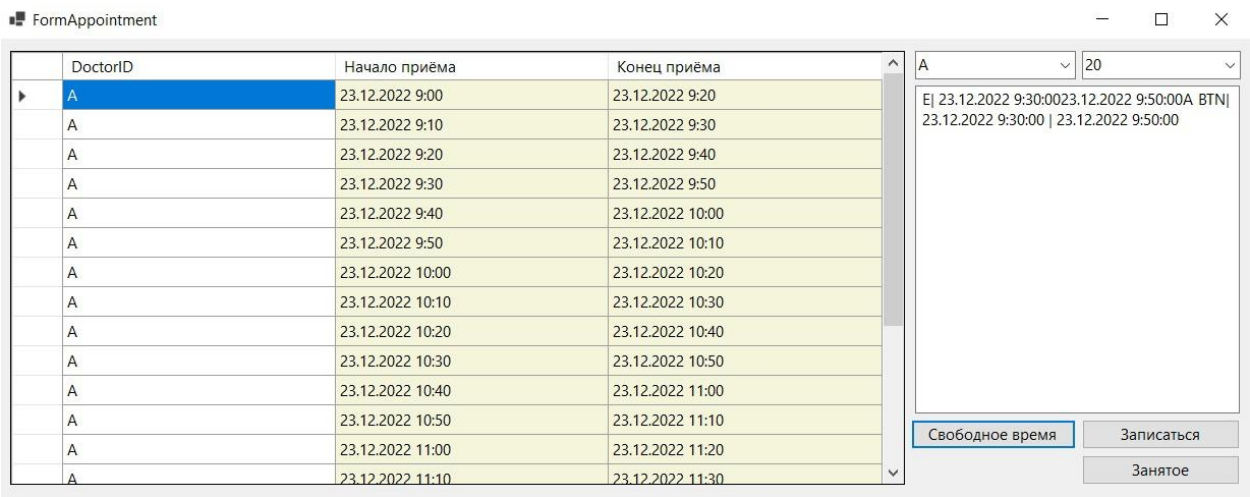

Рисунок 19 – Пример работы формы FormAppointment

Если на форме FormAppointment выбрать время и нажать кнопку «записаться», то откроется окно RegistrForm, в котором нужно будет подтвердить запись.

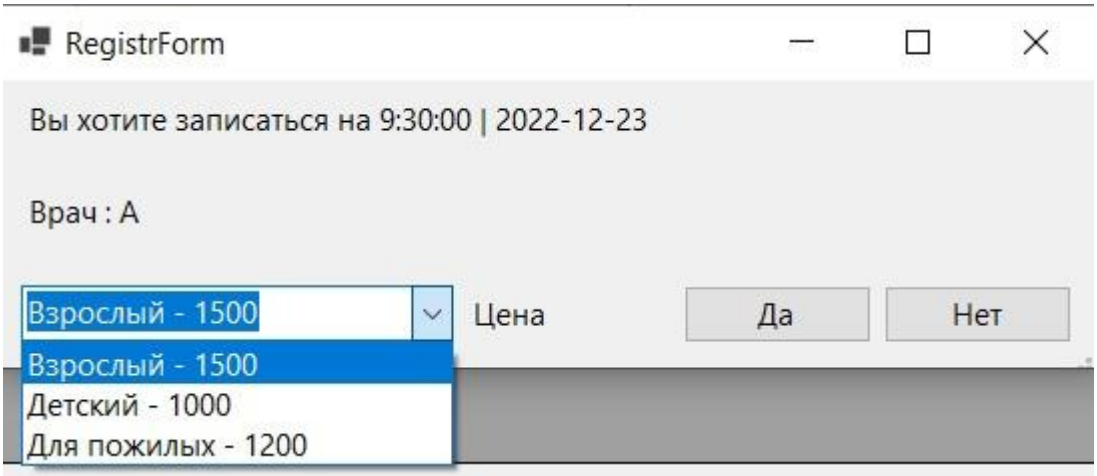

Рисунок 20 – Пример работы формы RegistrForm

На форме RegistrForm выбирается тип записи в зависимости от параметров, на данном этапе запись можно отменить.

ЗАКЛЮЧЕНИЕ

В курсовой работе было разработано приложение на WinForms с использованием объектно-ориентированного программирования (ООП) и его столпов: наследования, инкапсуляции, полиморфизма, абстракции.

В ходе разработки была взята специфика десктопного приложения, средой разработки послужила Visual Studio, а технологией разработки выступила WinForms.NET(C#).

Разработанная программа обеспечивает просмотр свободного времени для записи к врачам-специалистам и непосредственную запись к ним.

Данная курсовая работа закрепляет знания в области проектирования программ, а также навыков разработки и реализации пользовательских требований.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Разработка пользователя программного обеспечения – Текст: электронный // CoderLessons.com [сайт] – URL: https://coderlessons.com/tutorials/akademicheskii/programmnaia-inzheneriia/razrabotka-interfeisa-polzovatelia-programmnogo-obespecheniia (дата обращения: 02.12.2022).
2. Плюсы и минусы C++ – Текст: электронный // cjblogr.blogspot.com [сайт] – URL: https://cjblogr.blogspot.com/2015/02/blog-post.html (дата обращения: 02.12.2022).
3. Логанов С.В. Объектно-ориентированное программирование: учебное пособие для СПО / Логанов С.В., Моругин С.Л.. — Саратов, Москва: Профобразование, Ай Пи Ар Медиа, 2022. — 215 с. — ISBN 978-5-4488-1355-9, 978-5-4497-1586-9. — Текст: электронный // IPR SMART: [сайт]. — URL: https://www.iprbookshop.ru/118969.html (дата обращения: 02.10.2022). — Режим доступа: для авторизир. пользователей. - DOI: https://doi.org/10.23682/118969
4. Stack Exchange Q&A communities are different – Текст: электронный // https://stackexchange.com/ [сайт] – URL: https://stackexchange.com/ (дата обращения: 02.12.2022).
5. Adonet – Текст: электронный // metanit.com [сайт] – URL: https://metanit.com/sharp/adonet/2.12.php (дата обращения: 02.12.2022).
6. WITH в T-SQL // info-comp.ru [сайт] – URL: https://info-comp.ru/obucheniest/495-the-with-in-t-sql-or-common-table-expression.html (дата обращения: 02.12.2022).
7. Операции со строками – Текст: электронный // metanit.com [сайт] – URL: https://metanit.com/sharp/tutorial/7.2.php#:~:text=Обрезать%20определенную%20часть%20строки%20позво-

ляет,%2F%2F%20результат%20%22ро-
ший%20де%22%20Console.WriteLine(text) (дата обращения:
02.12.2022).

8. With common table – Текст: электронный // learn.microsoft.com [сайт] – URL: https://learn.microsoft.com/ru-RU/sql/t-sql/queries/with-common-table-expression-transact-sql?view=sql-server-linux-ver16 (дата обращения: 02.12.2022).
9. Sqlserver – Текст: электронный // metanit.com [сайт] – URL: https://metanit.com/sql/sqlserver/3.3.php (дата обращения: 02.12.2022).
10. SELECT предложение ORDER BY (Transact-SQL) – Текст: электронный // learn.microsoft.com [сайт] – URL: https:// https://learn.microsoft.com/ru-ru/sql/t-sql/queries/select-order-by-clause-transact-sql?view=sql-server-linux-ver16 (дата обращения: 02.12.2022).

ПРИЛОЖЕНИЕ А

```
using WinFormsApp2.View;

namespace WinFormsApp2
{
   internal static class Program
   {
      /// <summary>
      ///  The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         // To customize application configuration such as set high DPI settings or
default font,
         // see https://aka.ms/applicationconfiguration.
         ApplicationConfiguration.Initialize();
         Application.Run(new FormAppointment());
      }
   }
       }

namespace WinFormsApp2;

public class DoctorData
{
   public string DoctorID = ""; // clickCellTableDoctorID
   public DateTime Date;  // dt
   public string StartTime = "";  //  dccstart
   public string EndTime = "";  //dccend
   public char Status = ' ';  // status
   public char UserID = ' ';  //// i
   public int Price; // prise

}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WinFormsApp2.View
{
   public partial class FormAppointment : Form
   {
      private string connectDB =
         "Data Source=DESKTOP-432U1GM\\SQLEXPRESS;Initial Cata-
log=Kojina;Integrated Security=True;MultipleActiveResultSets=True";

      private string clickCellTableStart = "";
      private string clickCellTableEnd = "";
      private string clickCellTableDoctorID = "";
      private string clickCellTableUserID = "";

      public FormAppointment()
      {
         InitializeComponent();

         comboBox2.Items.Add("5");
         comboBox2.Items.Add("10");
         comboBox2.Items.Add("20");
         comboBox2.Items.Add("30");
         comboBox2.Items.Add("40");
         comboBox2.Items.Add("50");
         comboBox2.SelectedIndex = 0;
      }

      private void FormAppointment_Load(object sender, EventArgs e)
      {
         string squery = @$"
            DECLARE @doctorID char(1) = '{comboBox1.Text}';
            DECLARE @length tinyint = 20;
            WITH Slots AS (
               SELECT StartTime = DATEADD(MINUTE, ((DATEPART(MI-
NUTE, GETDATE())/10)+1+Number)*10, DATEADD(HOUR, DATE-
PART(HOUR, GETDATE()), CONVERT(smalldatetime, CONVERT(date, GET-
DATE()))),
```

```
                    EndTime = DATEADD(MINUTE, @length, DATEADD(MI-
NUTE, ((DATEPART(MINUTE, GETDATE())/10)+1+Number)*10, DATE-
ADD(HOUR, DATEPART(HOUR, GETDATE()), CONVERT(smalldatetime,
CONVERT(date, GETDATE()))))
                FROM Numbers)
            SELECT TOP {comboBox2.Text} DoctorID  = @doctorID,
              s.StartTime as 'Начало приёма',
              s.EndTime as 'Конец приёма'
             FROM Slots AS s
             WHERE NOT EXISTS (SELECT 1
                          FROM Appointment AS a
                          WHERE (CONVERT(time(0), s.StartTime) < a.EndTime
AND CONVERT(time(0), s.EndTime) > a.StartTime)
                            AND a.DoctorID = @doctorID
                            AND a.[Date] = CONVERT(date, s.StartTime))
              AND DATEPART(HOUR, s.StartTime) >= 9
              AND DATEPART(HOUR, DATEADD(MINUTE, -1, s.EndTime))
<= 16
            ORDER BY s.StartTime;
            ";
        try
        {
            SqlConnection sconn = new SqlConnection(connectDB);
            SqlCommand scomm = new SqlCommand(squery, sconn);
            DataTable stable = new DataTable();
            SqlDataAdapter sadapter = new SqlDataAdapter(scomm);
            sadapter.Fill(stable);
            dataGridView1.DataSource = stable;
            //dataGridView1.Columns[0].Visible = false;
            dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeCol-
umnsMode.Fill;
            dataGridView1.Columns[1].DefaultCellStyle.BackColor = Color.Green;
            dataGridView1.Columns[2].DefaultCellStyle.BackColor = Color.Green;
        }
        catch (Exception exception)
        {
            MessageBox.Show(exception.Message, "|error", MessageBoxBut-
tons.OK, MessageBoxIcon.Information);
            throw;
        }

        squery = @"select DISTINCT DoctorID from Appointment";
        try
```

```
        {
            SqlConnection sconn = new SqlConnection(connectDB);
            SqlCommand scomm = new SqlCommand(squery, sconn);
            DataTable stable = new DataTable();
            SqlDataAdapter sadapter = new SqlDataAdapter(scomm);
            sadapter.Fill(stable);

            comboBox1.DataSource = stable;
            comboBox1.DisplayMember = "DoctorID";
        }
        catch (Exception exception)
        {
            MessageBox.Show(exception.Message, "|error", MessageBoxBut-
tons.OK, MessageBoxIcon.Information);
        }
    }

    private void button1_Click(object sender, EventArgs e)
    {
        string squery = @"select DoctorID as 'id доктора', [Date] as 'Дата', Start-
Time as 'Начало приёма',
                    EndTime as 'Конец приёма', Status as 'Статус',
                    UserID as 'id пользователя', Price as 'Стоимость'
                    from Appointment";
        try
        {
            SqlConnection sconn = new SqlConnection(connectDB);
            SqlCommand scomm = new SqlCommand(squery, sconn);
            DataTable stable = new DataTable();
            SqlDataAdapter sadapter = new SqlDataAdapter(scomm);
            sadapter.Fill(stable);
            dataGridView1.DataSource = stable;
            //dataGridView1.Columns[0].Visible = false;
            dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeCol-
umnsMode.Fill;
            dataGridView1.Columns[2].DefaultCellStyle.BackColor = Color.Green;
            dataGridView1.Columns[3].DefaultCellStyle.BackColor = Color.Green;
        }
        catch (Exception exception)
        {
            MessageBox.Show(exception.Message, "|error", MessageBoxBut-
tons.OK, MessageBoxIcon.Information);
        }
```

```
    }

    private void dataGridView1_CellContentClick(object sender, DataGridView-
CellEventArgs e)
    {
        try
        {
            if (dataGridView1.Columns[dataGridView1.CurrentCell.Column-
Index].HeaderText.ToString() ==
                "Начало приёма")
            {
                clickCellTableStart = data-
GridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString();
                clickCellTableEnd = data-
GridView1.Rows[e.RowIndex].Cells[e.ColumnIndex + 1].Value.ToString();
                clickCellTableDoctorID = data-
GridView1.Rows[e.RowIndex].Cells[0].Value.ToString();

                textBox1.Text += "S| " + data-
GridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString()
                                + dataGridView1.Rows[e.RowIndex].Cells[e.Column-
Index + 1].Value.ToString()
                                + data-
GridView1.Rows[e.RowIndex].Cells[0].Value.ToString()
                                + "\r \n";
            }
            else if (dataGridView1.Columns[dataGridView1.CurrentCell.Column-
Index].HeaderText.ToString() ==
                    "Конец приёма")
            {
                clickCellTableEnd = data-
GridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString();
                clickCellTableStart = data-
GridView1.Rows[e.RowIndex].Cells[e.ColumnIndex - 1].Value.ToString();
                clickCellTableDoctorID = data-
GridView1.Rows[e.RowIndex].Cells[0].Value.ToString();

                textBox1.Text += "E| " + data-
GridView1.Rows[e.RowIndex].Cells[e.ColumnIndex - 1].Value.ToString()
                                + dataGridView1.Rows[e.RowIndex].Cells[e.Column-
Index].Value.ToString()
                                + data-
GridView1.Rows[e.RowIndex].Cells[0].Value.ToString()
```

```
                                + "\r \n";
                }
                else{}
            }
            catch (Exception exception)
            {
                MessageBox.Show("cell content exp | " + exception);
                throw;
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (clickCellTableStart != null)
            {
                textBox1.Text += "BTN| " + clickCellTableStart + " | " + clickCellTa-
bleEnd + "\r \n";
                RegistrForm registrForm =
                    new RegistrForm(clickCellTableStart, clickCellTableEnd,
clickCellTableDoctorID);
                registrForm.ShowDialog();
            }
            else
            {
                textBox1.Text += "NULL \r";
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            FormAppointment_Load(sender, e);
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Globalization;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WinFormsApp2.View
{
   public partial class RegistrForm : Form
   {
      private string connectDB =
         @"Data Source=DESKTOP-432U1GM\\SQLEXPRESS;Initial Cata-
log=Kojina;Integrated Security=True;MultipleActiveResultSets=True";

      private DoctorData doctorData = new DoctorData();

      private string clickCellTableStart = "";
      private string clickCellTableEnd = "";
      private string clickCellTableDoctorID = "";
      private string datecc = "";
      private string dccstart = "";
      private string dccend = "";
      private DateTime dt;

      public RegistrForm(string cCTS, string cCTE, string cCTDid)
      {
         InitializeComponent();

         clickCellTableStart = cCTS;
         clickCellTableEnd = cCTE;
         clickCellTableDoctorID = cCTDid;

         string[] ccstart = clickCellTableStart.Split(new char[] { ' ' }); // 0 - date / 1 -
timestart
         string[] ccend = clickCellTableEnd.Split(new char[] { ' ' }); // 0 - date / 1 -
timeend
         dccstart = ccstart[1];
         dccend = ccend[1];

         var test = ccstart[0].Split(new char[] { '.' });
         datecc = test[2] + "-" + test[1] + "-" + test[0];

         try
         { // "2022-12-17 14:40:52", "yyyy-MM-dd HH:mm:ss"
```

```
            dt = DateTime.ParseExact( datecc + " 14:40:52", "yyyy-MM-dd
HH:mm:ss",
                System.Globalization.CultureInfo.InvariantCulture);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "error", MessageBoxButtons.OK, Mes-
sageBoxIcon.Information);
        }

        label1.Text += " " + ccstart[1]+ " | " + datecc;
        label2.Text += clickCellTableDoctorID;

        toolTip1.SetToolTip(comboBox1, @"Выбирите цену в зависимости от
ваших условий
                                        *Взрослый - 1500
                                        *Детский - 1000
                                        *Для пожмлых - 1200");
                                        // id 1500 => 'S'(standatr), 'C'(children), 'P'(Old)
        comboBox1.Items.Add("Взрослый - 1500");
        comboBox1.Items.Add("Детский - 1000");
        comboBox1.Items.Add("Для пожмлых - 1200");
        comboBox1.SelectedIndex = 0;
    }


    private void button2_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        int i = 1;
        char status = 'S'; // id 1500 => 'S'(standatr), 'C'(children), 'P'(Old)
        int prise = 0;
        if (comboBox1.Text == "Детский - 1000")
        {
            status = 'C';
            prise = 1000;
        }
        else if (comboBox1.Text == "Для пожмлых - 1200")
        {
```

```
            status = 'P';
            prise = 1200;
        }
        else
        {
            status = 'S';
            prise = 1500;
        }

        //INSERT INTO Appointment VALUES ('A', '20221217', '09:00:00',
'09:10:00', 'P', '1', '1500');
        //"INSERT INTO Appointment (DoctorID, [Date], StartTime, EndTime,
Status, UserID, Price) VALUES (@surname,@kind_sport,@place, @id_country)";
        string query = @"INSERT INTO Appointment
                    (DoctorID, [Date], StartTime, EndTime, Status, UserID, Price)
VALUES
                    (@DoctorID, @Date, @StartTime, @EndTime, @Status,
@UserID, @Price)";
        try
        {
            SqlConnection sconn = new SqlConnection(connectDB);
            sconn.Open();
            SqlCommand command = new SqlCommand(query, sconn);

            doctorData.DoctorID = clickCellTableDoctorID;
            doctorData.Date = dt;
            doctorData.StartTime = dccstart;
            doctorData.EndTime = dccend;
            doctorData.Status = status;
            doctorData.UserID = char.Parse(i.ToString());
            doctorData.Price = prise;

            command.Parameters.AddWithValue("@DoctorID", doctorData.Doc-
torID);
            command.Parameters.AddWithValue("@Date", doctorData.Date);
            command.Parameters.AddWithValue("@StartTime", doctorData.Start-
Time);
            command.Parameters.AddWithValue("@EndTime", doctorData.End-
Time);
            command.Parameters.AddWithValue("@Status", doctorData.Status);
            command.Parameters.AddWithValue("@UserID", doctorData.UserID);
            command.Parameters.AddWithValue("@Price", doctorData.Price);
```

```
            int result = command.ExecuteNonQuery();

            if (result < 0)
                MessageBox.Show("Ошибка добавления строки в базу данных! " +
result.ToString());

            sconn.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "error", MessageBoxButtons.OK, Mes-
sageBoxIcon.Information);
        }

        this.Close();
    }
  }
}
```