Реферат

Пояснительная записка курсового проекта 39 с., 10 рис., 14 источников, 2 приложения.

ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, СИСТЕМА ПОДДЕРЖКИ СОСТАВЛЕНИЯ РАСПИСАНИЯ ЗАНЯТИЙ, ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ, ДИАГРАММА КЛАССОВ, C#, WINDOWS FORMS

Объектом исследования являются программное обеспечение способное работать в качестве календаря с расписанием занятий и возможностью составления множества расписаний двухнедельных расписаний занятий.

Целью курсового проекта является проектирование и разработка программного обеспечения «Система поддержки составления расписания занятий».

К полученным результатам относится приложение WinForms, написанное на языке программирования C# в формате исполняемых файлов (.exe). Данное программное обеспечение было создано в среде разработки Microsoft Visual Studio.

При выполнении курсовой работы были выполнены все этапы создания программного продукта: от постановки задачи до практической реализации. В ходе выполнения курсовой работы были приобретены навыки работы со специализированной литературой, справочниками, стандартами

Содержание

Введение

Целью курсового проекта является проектирование и разработка программного продукта, который будет предоставлять возможность просматривать расписание занятий в специальном календаре, а также редактирование расписания и создание множества других двухнедельных расписаний занятий.

Задачи курсового проекта:

– изучить предметную область и провести анализ существующих решений;

– составить техническое задание проекта;

– спроектировать программный продукт;

– реализовать программный продукт;

– составить руководство пользователя.

Предметом исследования курсового проекта является разработка сайта на языках html и css, взаимодействующего с базой данных, созданной на языке SQL.

В первой главе проекта указаны нормативные ссылки, которые были использованы при написании курсового проекта.

Во второй главе перечислены основные цель разработки, форма ее реализации, перечень потребностей конечного пользователя приложения. В конце главы представлена диаграмма вариантов использования приложения.

В третьей главе содержится техническое задание проекта, на основе которого строится работа учебного комплекса.

В четвертой главе описан процесс проектирования программного продукта, выбор средств разработки.

В пятой главе описан процесс разработки программного продукта и руководство пользования готовым программным продуктом.

В заключении приводятся основные выводы из проведенного исследования, а также основные рекомендации по использованию ПО.

1 Нормативные ссылки

В данном курсовом проекте использованы следующие нормативные ссылки:

ГОСТ Р 1.5-2004 Стандартизация в Российской Федерации. Стандарты национальные Российской Федерации. Правила построения, изложения, оформления и обозначения.

ГОСТ Р 1.12-2004 Стандартизация в Российской Федерации. Термины и определения.

ГОСТ Р 7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления.

ГОСТ Р ИСО 9000-2008 Системы менеджмента качества. Основные положения и словарь.

ГОСТ Р ИСО/МЭК 12207-99 Информационная технология. Процессы жизненного цикла программных средств.

ГОСТ 2.001-93 ЕСКД. Общие положения.

ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных продуктов.

ГОСТ 19.105-78 ЕСПД. Обозначение программ и программных продуктов.

ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению.

ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.

2 Анализ предметной области

2.1 Цель приложения

Class Schedule Support System – программное обеспечение для просмотра, редактирования и добавления двухнедельного расписания занятий.

Ожидается, что программное обеспечение будет использоваться в целях контроля и планирования времени, как календарь; как система для составления собственного двухнедельного расписания.

2.2 Диаграмма вариантов использования

Вариант использования (use case) представляет собой последовательность действий, выполняемых системой в ответ на событие, инициализируемое некоторым внешним объектом. Вариант использования описывает типичное взаимодействие между пользователем и системой. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать. Диаграмма вариантов использования изображена на рисунке 2.2.1.

Рисунок 2.2.1 – Диаграмма вариантов использования

Диаграмма вариантов использования была построена при помощи ресурса [2].

2.3 Аналоги

Первым аналогом приведу приложение для смартфонов Timetable. Красивое и интуитивно понятное приложение для управления школьной жизнью. Можно внести расписание, домашние задания, экзамены и даже каникулы. Приложение может синхронизироваться со всеми вашими Android-устройствами, а во время занятий само перейдёт в беззвучный режим. Интерфейс приложения изображен на рисунке 2.3.1.

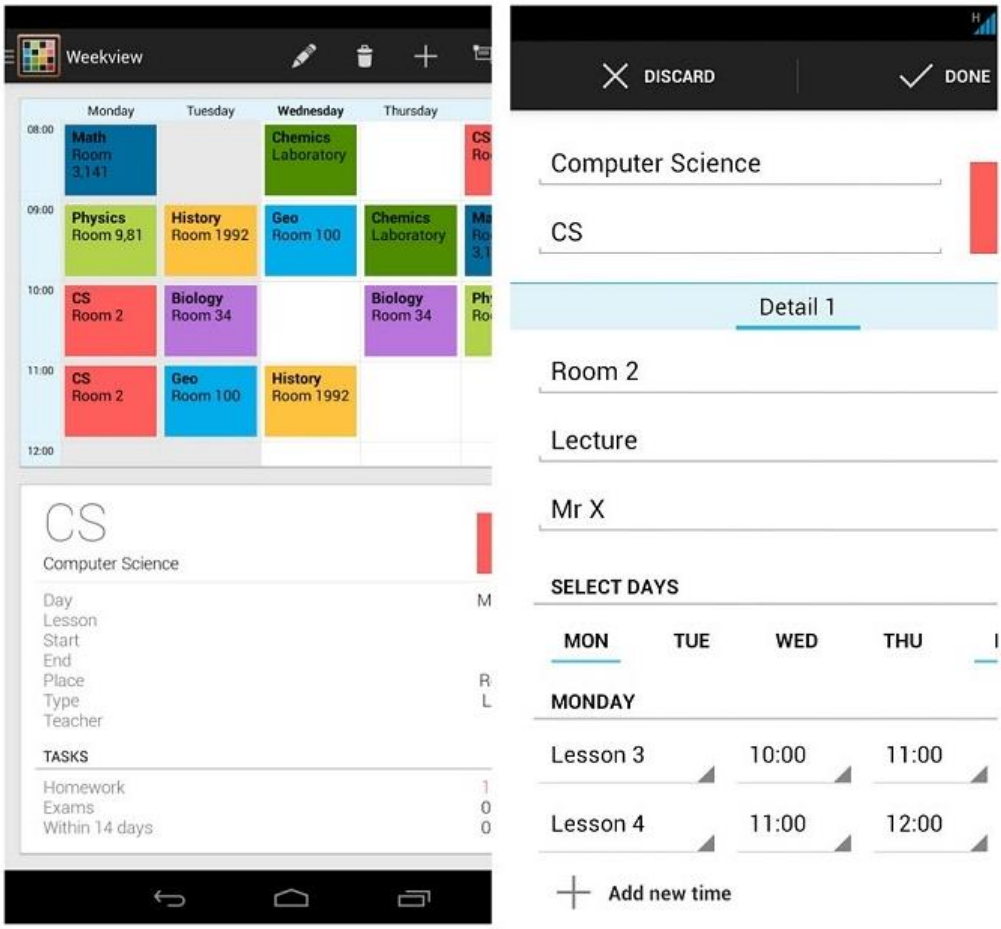

Рисунок 2.3.1 – Интерфейс приложения Timetable

Вторым аналогом будет приложение под iOS – Grade Hound. Календарь для школьников и студентов с возможностью маркировки предметов по цветам и проставлением оценок по предметам. Изюминка: временные графики, показывающие, сколько времени вы потратите на тот или иной предмет. Минус: приложение не поддерживает русский язык. Интерфейс приложения изображен на рисунке 2.3.2. Все аналоги, приведенные в данном разделе были найдены на электронном ресурсе [3].

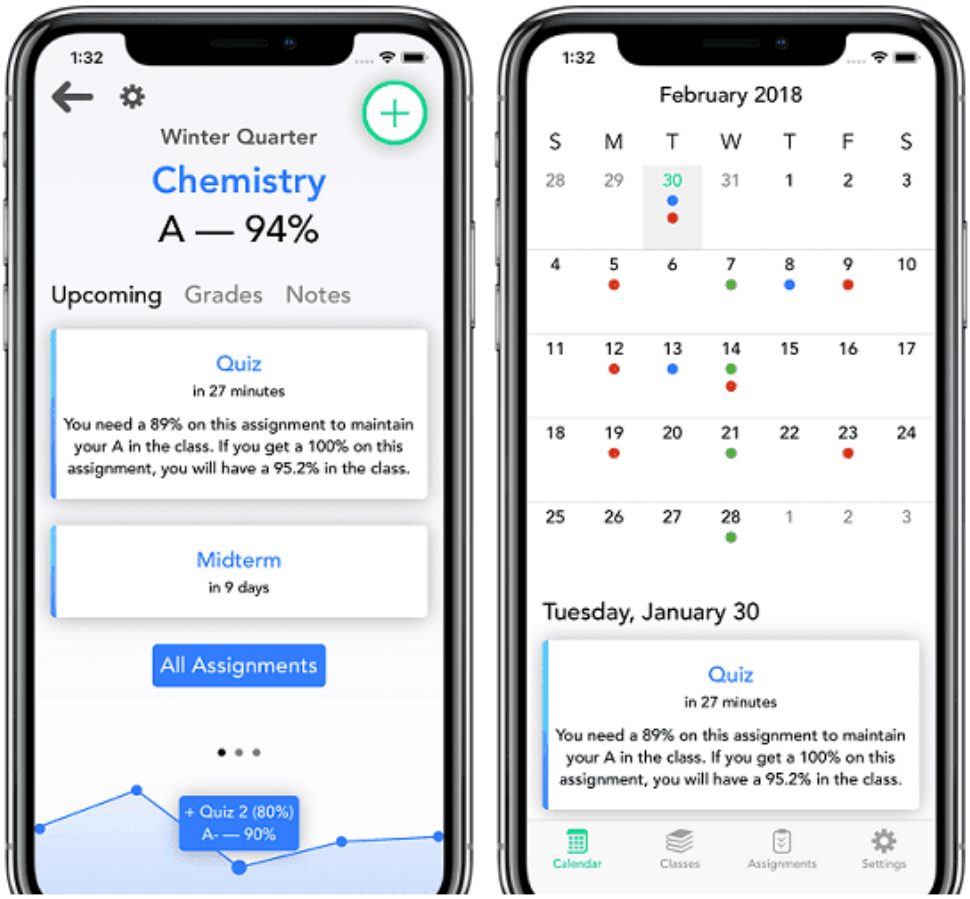

Рисунок 2.3.2 – Интерфейс приложения Grade Hound

3 Техническое задание

3.1 Введение

Настоящее техническое задание распространяется на разработку системы поддержки составления расписания занятий.

Разработанное программное обеспечение позволит пользователям с компьютерами просматривать двухнедельное расписание занятий, редактировать и составлять новые в разном количестве.

3.2 Основания для разработки

Указание о закреплении тем курсовых проектов №24-КТ от 23.09.2020.

3.3 Назначение разработки и область применения программы

Функциональным назначением программного продукта является предоставление пользователю возможности просматривать расписание занятий в календаре и редактировать его под свои нужды, а также составлять собственное расписание.

Программа должна эксплуатироваться студентами и преподавателями в практическом и лабораторном курсе дисциплин кафедры ИСП. Программа позволит повысить уровень и качество учебного процесса студентов кафедры ИСП, упростить процесс редактирования и просмотра расписания занятий.

3.4 Требование к программе или программному изделию

Программа должна предоставлять пользователю следующий функционал:

– выбор расписания для конкретной группы;

– выбор учебной недели;
– выбор дня недели;
– просмотр расписания на день;
– редактирование расписания на день;
– удаление расписания;
– добавления новой группы расписания.

3.5 Требования к надежности

– ПО должно иметь защиту от некорректных действий пользователей и ошибочных исходных данных;
– ПО не должно во время работы модифицировать свой код или код других программ;
– количество отказов ПО из-за не выявленных ошибок не должно превышать 1 отказа на 1000 сеансов работы с программой.

3.6 Время восстановления программы после отказа

– максимальное время устранения неисправностей в работе оборудования сервера системы – 24 часа;
– максимальное время устранения неисправностей в работе по сервера системы – 12 часов;
– максимальное время устранения неисправностей в работе sql сервера бд системы – 12 часов;
– время устранения сбоев в работе системного и прикладного по рабочих станций – 12 часов.

3.7 Информационное обеспечение

– средства ввода данных в систему должны обеспечивать контроль правильности данных по типу;

– при модификации и удалении данных средства ведения ПО должны запрашивать подтверждение правильности выданных команд и временно сохранять предыдущую версию данных;

– при выполнении запросов, время которых превышает 10 секунд, на монитор пользователя должен выдаваться транспарант с извещением о вынужденном ожидании.

3.9 Требования к составу и параметрам технических средств

В состав технических средств должны входить IBM-совместимый персональный компьютер, выполняющий роль сервера, включающий в себя:

– 32-разрядный или 64-разрядный процессор с тактовой частотой 1 ГГц или выше;

– 1 ГБ ОЗУ для 32-разрядного процессора или 2 ГБ ОЗУ для 64-разрядного процессора;

– 16 ГБ для 32-рязрядной системы или 20 ГБ для 64-разрядной системы свободного места на жестком диске;

– операционную систему Windows 7 или новее.

3.10 Предварительный состав программной документации

Состав программной документации должен включать в себя:

– техническое задание;

– программный продукт и методику испытаний;

– руководство оператора.

3.11 Экономические преимущества разработки

Ориентировочная экономическая эффективность не рассчитывается в виду уникальности предъявляемых требований к разработке.

3.12 Стадии разработки

Разработка должна быть проведена в 3 стадии:

– разработка технического задания;
– рабочее проектирование;
– внедрение.

3.13 Этапы разработки

На стадии разработки ТЗ должен быть выполнен этап разработки, согласования и утверждение технического задания. На стадии рабочего проектирования должны быть выполнены следующие этапы работ:

– разработка программы;
– разработка программной документации;
– испытания программы.

4 Проектирование программного обеспечения

4.1 Проектирование интерфейса программы

Одним из важных этапов разработки – это проектирование интерфейса программы. Это то, как будет выглядеть конечный продукт для пользователя, но без детализации, просто в виде схематичных блоков. Данный макет может подсказать нам, какие элементы управления необходимо разработать, какие контейнеры необходимы для размещения всей информации. Макет интерфейса приложения изображен на рисунке 4.1.1. Проектирование основного интерфейса приложения было реализовано при помощи ресурса [2].

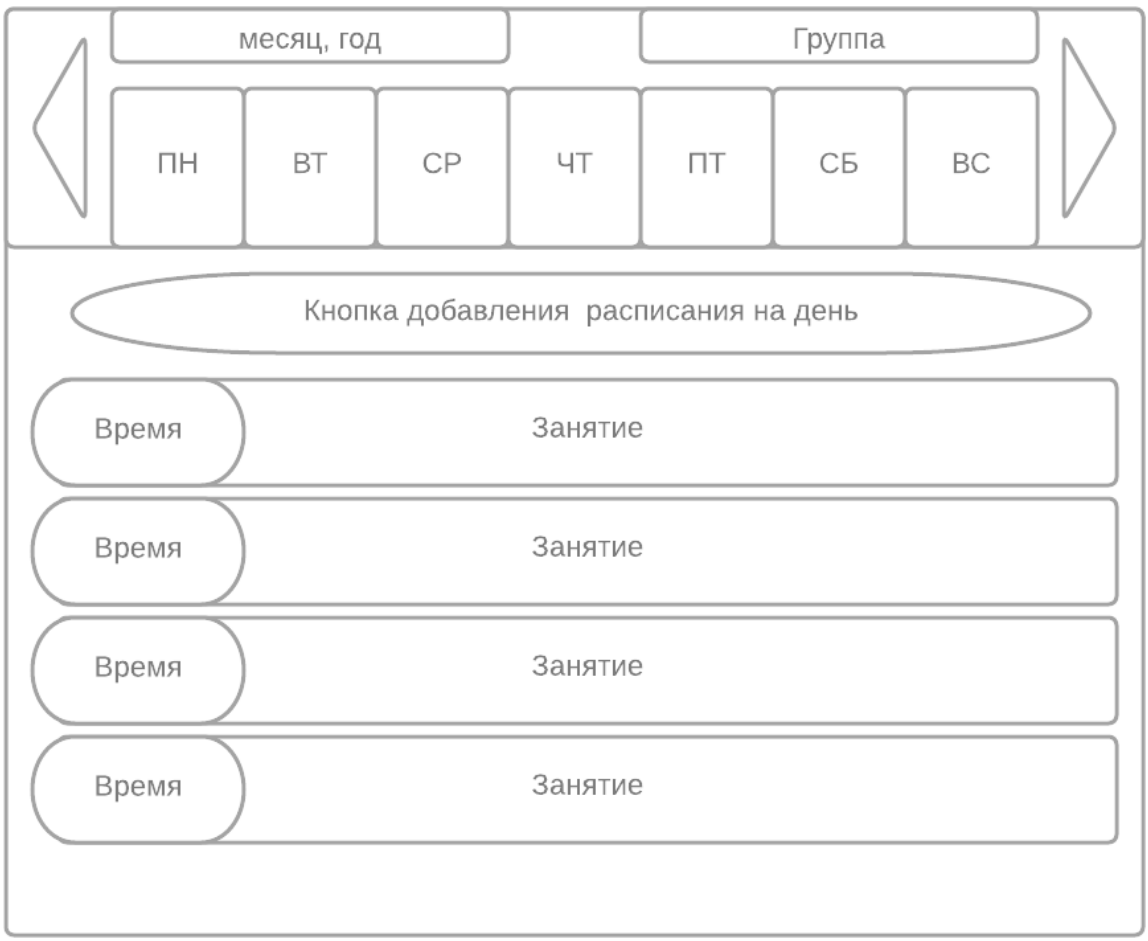

Рисунок 4.1.1 – Макет интерфейса окна приложения

4.2 Выбор языка программирования

После того, как мы увидели то, как схематически должен выглядеть готовый продукт для пользователя, нужно определиться при помощи каких средств программирования этого можно достичь. Для написания данного проекта будет использоваться язык программирования C#, библиотеки для работы с средствами построения графического интерфейса Windows Forms и среда разработки Microsoft Visual Studio 2019 для удобного построения графического дизайна и упрощения написания кода.

4.3 Проектирование классов

В приложении, а именно в графическом интерфейсе существует множество элементов управления. Для упрощения спроектируем так, чтобы группа маленьких элементов составляла более крупный, а он в свою очередь еще более крупный, таким образом, у нас не будет большого количества переменных в которых мы будем путаться. У ас все элементы будут перегружены при помощи пользовательских элементов управления. Диаграмма классов данного проекта изображена на рисунке 4.3.1. Проектирование диаграммы классов осуществлялось при помощи электронного ресурса [2].

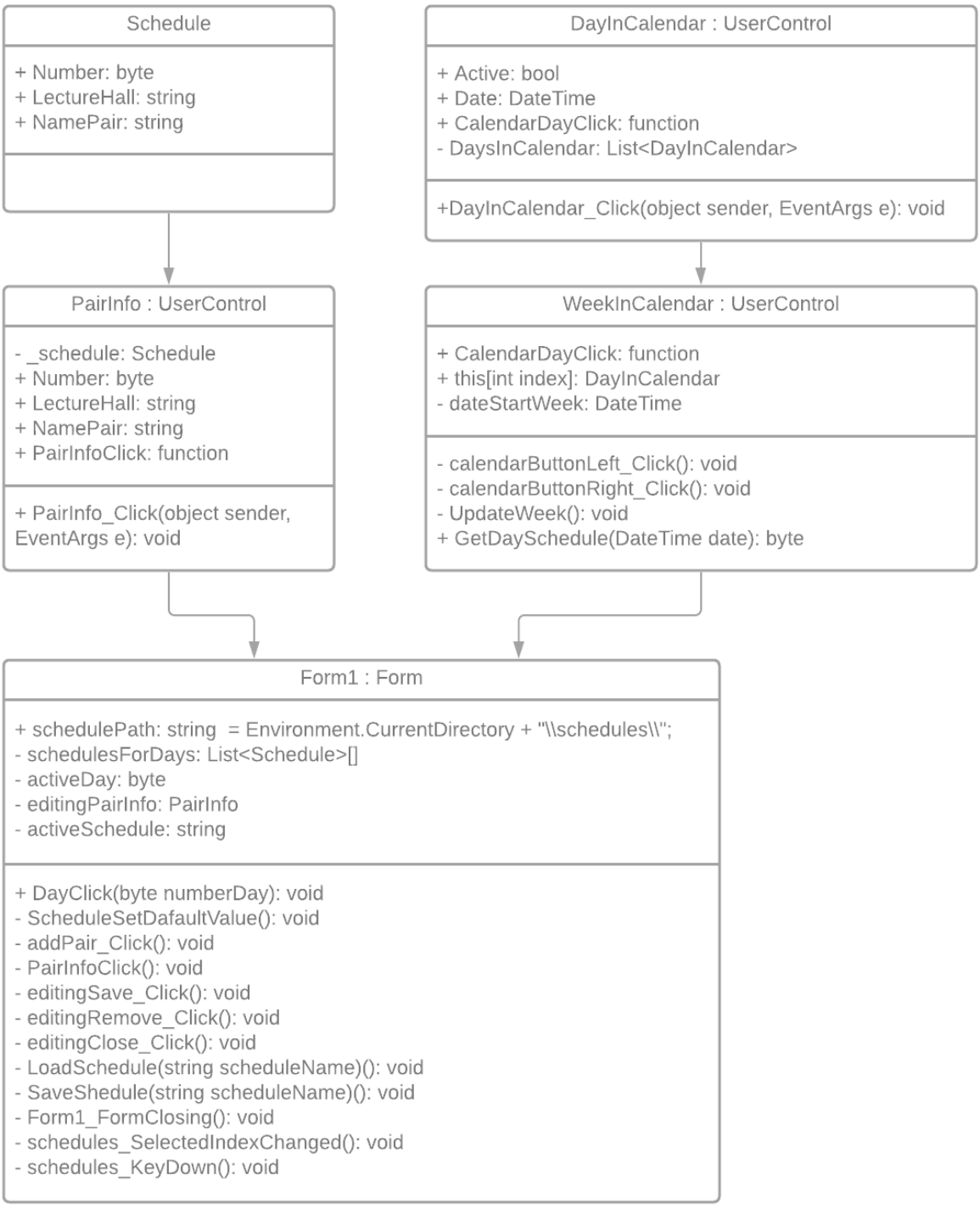

Рисунок 4.3.1 – Диаграмма классов

5 Реализация программного продукта

5.1 Описание методов программирования

Одним из самых главных компонентов, Которые были использованы при разработке является класс DateTime. Он предоставляет удобный интерфейс для взаимодействия со временем и датой. Фрагмент кода с использованием возможностей данного класса:

```
[DisplayName(@"Date"), Description("Описание"),
Category("Данные"), DefaultValue(30)]
    public DateTime Date
    {
      get { return _date; }
      set
      {
        _date = value;
        labelNumberDay.Text = _date.Day.ToString();
        labelText.Text = _date.ToString("ddd").ToUpper();
      }
    }
```

На примере выше представлено свойство, при изменении которого происходит запись на пользовательский элемент управления дня недели в сокращенном формате (ПН, ВТ и т.д.). Информация по работе с классом DateTime найдена на электронном ресурсе [4, 5].

Для взаимодействия большого количества форм была необходимость передавать функции как параметры, для этого использовались делегаты. Пример делегата для передачи события клика мыши по элементу управления, для того чтобы осуществить связь данного элемента управления с основной формой приведен ниже:

```
public delegate void HandlerCalendarDayClick(byte numberDay);
```

Информация по работе с делегатами найдена на электронном ресурсе [6]. Также для обработки большого количества событий, таких как: нажатия на элемент управления, наведения на него, выбора значения, изменения значения и многих других, было необходимо основательно разобраться в

создании своих обработчиков событий (эвентов). Информация по событиям и их обработке была найдена на электронном ресурсе [7, 8, 9].

Для вывода пользователю быстрых уведомлений в виде отдельного окна, иногда с возможностью выбора варианта ответа (Да/Нет) использовался класс MessageBox. Пример кода вызова диалогового окна для создания нового расписания:

```
// Спрашиваем, если такого расписания нет
  DialogResult result = MessageBox.Show(
    "Создать новое рассписание?",
    "Сообщение",
    MessageBoxButtons.YesNo,
    MessageBoxIcon.Question,
    MessageBoxDefaultButton.Button1,
    MessageBoxOptions.DefaultDesktopOnly);


// Создаем новое рассписание
  if (result == DialogResult.Yes)
  {
    if (activeSchedule != null)
      SaveShedule(activeSchedule);

    Directory.CreateDirectory(path);
    activeSchedule = schedules.Text;

    ScheduleSetDafaultValue();
    schedules.Items.Add(schedules.Text);
    schedules.SelectedItem = schedules.Items.Count - 1;
  }
```

Информация по работе с классом MessageBox была найдена на электронном ресурсе [10].

Для сохранения расписания необходимо было разобраться с работой библиотек для работы с файлами, директориями и потоками ввода/вывода. Информация по работе с каталогами была найдена на электронном ресурсе [11], информация по работе с файлами была найдена на электронном ресурсе [12], информация по сериализации объектов была найдена на электронных ресурсах [13, 14].

При помощи данных технологий производится эффективное в плане размера и сложности кода хранение расписания всех занятий в одном файле на каждую группу. Методы сохранения и загрузки данных:

```
private void LoadSchedule(string scheduleName)
{
     string path = schedulePath + scheduleName + "\\" +
scheduleName + ".sched";
     BinaryFormatter binFormat = new BinaryFormatter();

     // Сохранить объект в локальном файле.
     Stream fStream = new FileStream(path, FileMode.Open,
                       FileAccess.Read, FileShare.ReadWrite);
     schedulesForDays =
               (List<Schedule>[])binFormat.Deserialize(fStream);
     fStream.Close();
}
 // Сохранение рассписания в файл
private void SaveShedule(string scheduleName)
{
     string path = schedulePath + scheduleName + "\\" +
                                          scheduleName + ".sched";
     BinaryFormatter binFormat = new BinaryFormatter();

     // Сохранить объект в локальном файле.
     Stream fStream = new FileStream(path, FileMode.Create,
                          FileAccess.Write, FileShare.ReadWrite);
     binFormat.Serialize(fStream, schedulesForDays);
     fStream.Close();
}
```

Весь код программного продукта представлен в Приложении А.

5.2 Руководство пользователя

Как только пользователь запустит приложение «Class Schedule Support System» перед ним откроется главное окно программы (см. пример на рисунке 5.2.1). По началу на главном окне не будет ничего кроме календаря. Для того чтобы отобразился интерфейс занятий необходимо выбрать в выпадающем меню готовое расписание или создать свое (см. пример на рисунке 5.2.2).

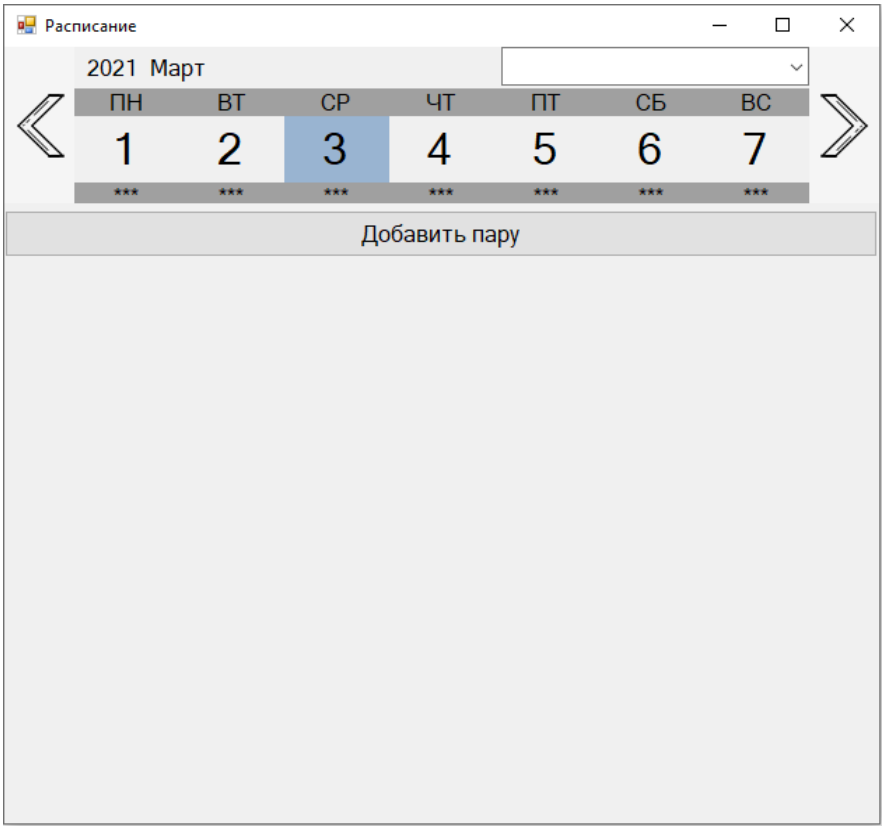

Рисунок 5.2.1 – Главное окно приложения

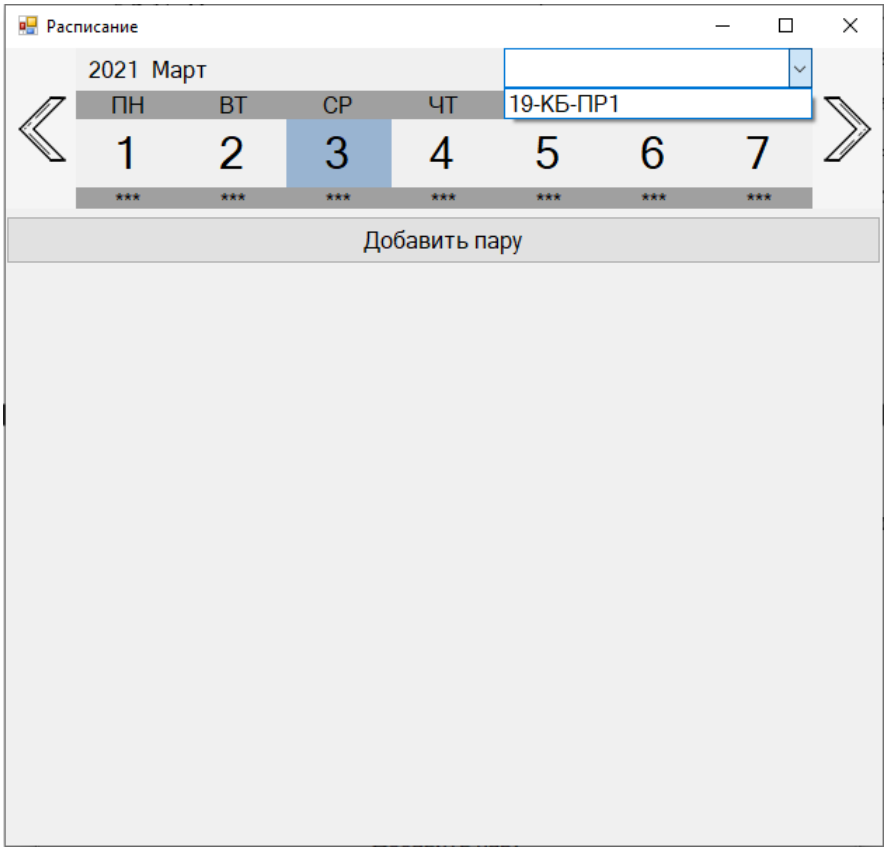

Рисунок 5.2.2 – Выбор расписания из списка существующих

Для того чтобы создать свое расписание необходимо вбить его название в поле и нажать клавишу «Enter». После чего будет высвечено сообщение с вариантом выбора «Создать расписание?» по нажатии кнопки «Да» будет создано новое расписание (см. пример на рисунке 5.2.3).

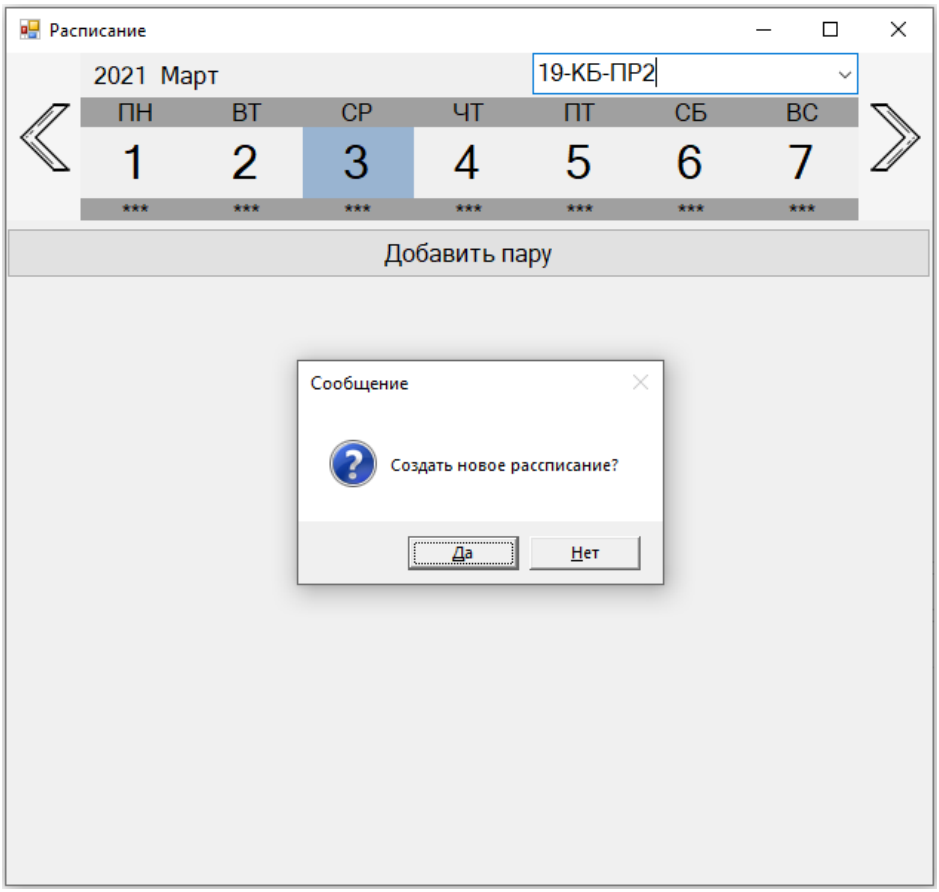

Рисунок 5.2.3 – Создание нового расписания

После того как было выбрано существующее расписание, оно сразу отобразиться на панели снизу (см. пример на рисунке 5.2.4). В случае, если необходимо отредактировать расписание нужно нажать на соответствующую пару и в появившейся форме изменить информацию о занятии или вовсе его удалить (см. пример на рисунке 5.2.5).

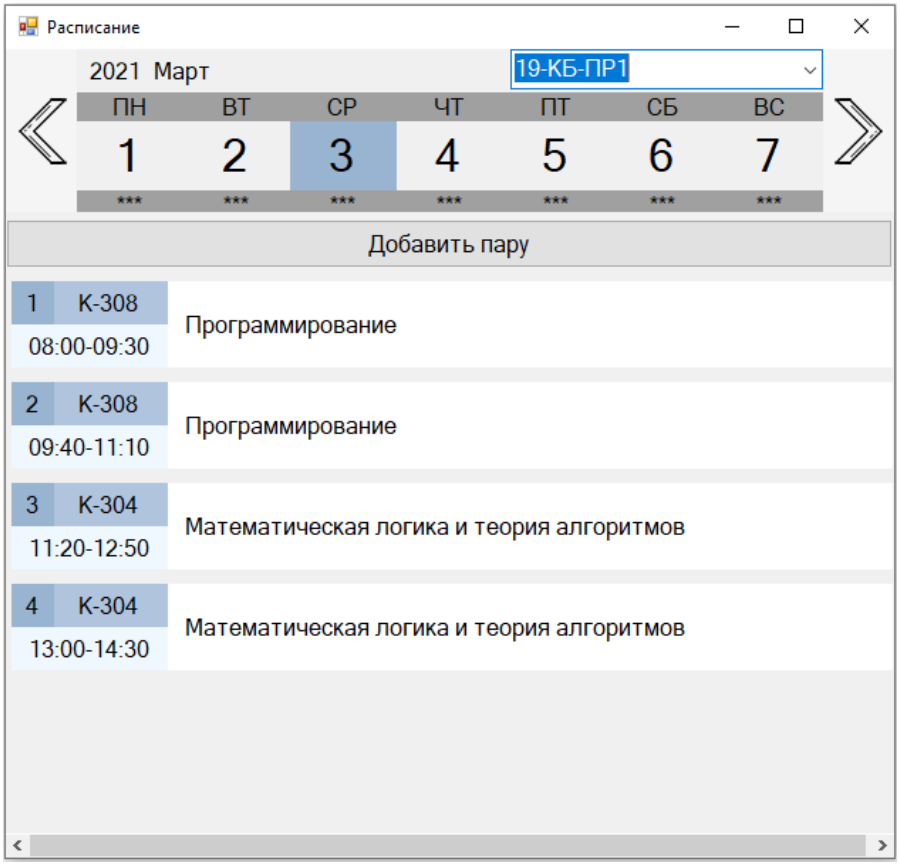

Рисунок 5.2.4 – Пример с загруженным расписанием

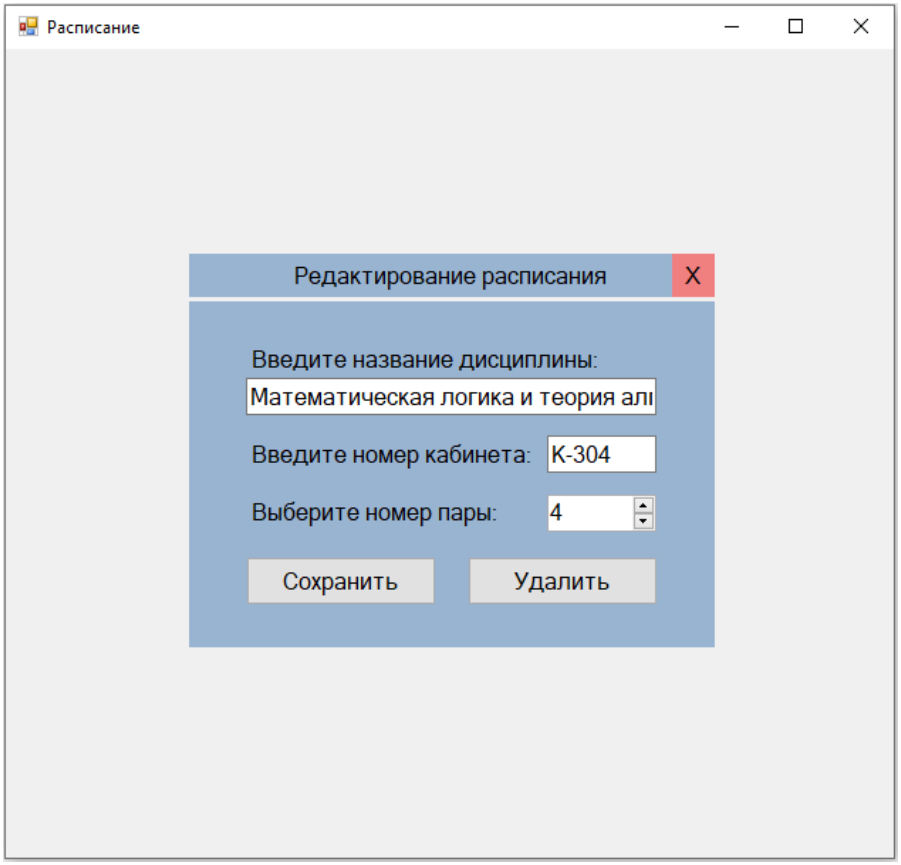

Рисунок 5.2.5 – Пример редактирования расписания занятия

Для того, чтобы добавить новое занятие в расписание дня необходимо нажать на кнопку «Добавить пару», после чего откроется уже знакомая форма с информацией о занятии. Для того, чтобы посмотреть расписание на другой день недели, необходимо просто нажать на соответствующий день мышкой. Если необходимо посмотреть расписание на следующей неделе, или предыдущей, нужно нажать на стрелочки справа и слева соответственно.

Заключение

В результате курсового проекта по дисциплине «Технологии разработки программного обеспечения» разработан программное обеспечение, позволяющее просматривать расписание занятий, редактировать, удалять и добавлять новое двухнедельное расписание.

Цель курсового проекта, которая заключается в проектировании и разработке учебного комплекса, выполнена.

В ходе работы были выполнены следующие задачи:

- произведено изучение предметной области;
- произведен анализ существующих решений и аналогов;
- техническое задание проекта составлено;
- программный продукт спроектирован;
- программный продукт разработан;
- составлено руководство пользователя и пояснительная записка проекта.

В ходе разработки использовались такие программные средства, как язык программирования C#, библиотека для работы с графическим интерфейсом управления Windows Forms и среда, которая все это в себе совмещает и предоставляет быстрый и простой доступ ко всем средствам тестирования и разработки программного кода Microsoft Visual Studio.

Список используемых источников

1. Техническое задание. Требование к содержанию и оформлению. ГОСТ 19.201-78 [Текстовый документ]
2. Lucid [Электронный ресурс] https://lucid.app/
3. 10 приложений для составления школьного расписания [Электронный ресурс] https://externat.foxford.ru/polezno-znat/10
4. Форматирование дат и времени [Электронный ресурс] https://metanit.com/sharp/tutorial/19.2.php
5. Практическое руководство. Извлечение дня недели из конкретной даты [Электронный ресурс] https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/how-to-extract-the-day-of-the-week-from-a-specific-date
6. Делегаты [Электронный ресурс] https://metanit.com/sharp/tutorial/3.13.php
7. События C# по-человечески [Электронный ресурс] https://habr.com/ru/post/213809/
8. How to create event button click in usercontrol into usercontrol on winform C# [Электронный ресурс] https://stackoverflow.com/questions/23003862/how-to-create-event-button-click-in-usercontrol-into-usercontrol-on-winform-c-sh
9. События [Электронный ресурс] https://metanit.com/sharp/tutorial/3.14.php
10. Окно сообщения MessageBox [Электронный ресурс] https://metanit.com/sharp/windowsforms/4.19.php
11. Работа с каталогами [Электронный ресурс] https://metanit.com/sharp/tutorial/5.2.php
12. Работа с файлами. Классы File и FileInfo [Электронный ресурс] https://metanit.com/sharp/tutorial/5.3.php

13. Formatter сериализации [Электронный ресурс] https://professorweb.ru/my/csharp/thread_and_files/level4/4_3.php

14. Сериализация. Введение в сериализацию объектов [Электронный ресурс] https://metanit.com/sharp/tutorial/6.1.php

Приложение А

Листинг

Файл Program.cs

```
using System;
using System.Windows.Forms;

namespace ClassScheduleSupportSystem
{
  static class Program
  {
    /// <summary>
    /// Главная точка входа для приложения.
    /// </summary>
    [STAThread]
    static void Main()
    {
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);
      Application.Run(new Form1());
    }
  }
}
```

Файл Form1.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.IO;
using System.Linq;
using System.Runtime.Serialization.Formatters.Binary;
using System.Windows.Forms;

namespace ClassScheduleSupportSystem
{
  public partial class Form1 : Form
  {
    // Делегат для передачи функции редактирования расписания
    public delegate void HandlerCalendarDayClick(byte numberDay);
    public delegate void HandlerPairInfoClick(PairInfo pairInfo);

    List<Schedule>[] schedulesForDays;
    private byte activeDay;

    string schedulePath = Environment.CurrentDirectory +
"\\schedules\\";
```

```
    public Form1()
    {
      InitializeComponent();

      // Добавление обработчика клика на день в календаре
      weekInCalendar1.CalendarDayClick = DayClick;
      //
      foreach (var pI in schedulePanel.Controls)
        if (pI.GetType() == typeof(PairInfo))
          ((PairInfo)pI).PairInfoClick = PairInfoClick;

      // Создаем структуру для хранения расписания группы
      ScheduleSetDafaultValue();

      // Определяем существование каталога с рассписаниями
      if (!Directory.Exists(schedulePath))        // Если не
существует
        Directory.CreateDirectory(schedulePath);  // Создаем каталог

      // Получаем список каталогов с различными рассписаниями
      List<string> scheduleNames = new List<string>(from path in
Directory.EnumerateDirectories(schedulePath)
                                                    select path.Remove(0,
path.LastIndexOf('\\') + 1));

      // Добавление всех рассписаний на форму
      foreach (var schedule in scheduleNames)
      {
        if (!schedules.Items.Contains(schedule))
          schedules.Items.Add(schedule);
      }

      // Установка текущего дня в календаре
      activeDay = weekInCalendar1.GetDaySchedule(DateTime.Now);
      weekInCalendar1[6 - activeDay % 7].Active = true;
      DayClick(activeDay);
    }


    // Отрисовка расписания на день
    public void DayClick(byte numberDay)
    {
      activeDay = numberDay;
      // Редактировать расписание
      //MessageBox.Show(numberDay.ToString());

      // Удаляем старое расписание на день
      for (byte i = 0; i < schedulePanel.Controls.Count; i++)
        if (schedulePanel.Controls[i].GetType() == typeof(PairInfo))
          schedulePanel.Controls.RemoveAt(i--);
```

```
      // Загружаем новое расписание на день \\
      // Если в расписании есть пары
      if (schedulesForDays[numberDay].Count > 0)
      {
        // Сортируем пары
        schedulesForDays[activeDay] = (from s in
schedulesForDays[activeDay] orderby s.Number select s).ToList();
        // Добавляем на панельку
        foreach (Schedule s in schedulesForDays[numberDay])
        {
          schedulePanel.Controls.Add(new PairInfo(s));

((PairInfo)schedulePanel.Controls[schedulePanel.Controls.Count -
1]).PairInfoClick = PairInfoClick;
        }
      }
    }

    private void ScheduleSetDafaultValue()
    {
      schedulesForDays = new List<Schedule>[14];
      for (byte i = 0; i < 14; i++)
        schedulesForDays[i] = new List<Schedule>(8);
    }
    private PairInfo editingPairInfo;
    // Добавление новой пары в расписание дня
    private void addPair_Click(object sender, EventArgs e)
    {
      if (activeSchedule == null) return;

      Schedule newSched = new Schedule(1, "", "");
      schedulesForDays[activeDay].Add(newSched);

      PairInfo newPairInfo = new PairInfo(newSched);
      newPairInfo.PairInfoClick = PairInfoClick;
      schedulePanel.Controls.Add(newPairInfo);

      PairInfoClick(newPairInfo);
    }
    private void PairInfoClick(PairInfo pairInfo)
    {
      editingNumberPair.Value = pairInfo.Number;
      editingLectureHall.Text = pairInfo.LectureHall;
      editingNamePair.Text = pairInfo.NamePair;

      editingPairInfo = pairInfo;
      panelEditingSchedule.Visible = true;  // Отображение панели
редактора
    }
```

```
    // Сохранить изменения
    private void editingSave_Click(object sender, EventArgs e)
    {
      panelEditingSchedule.Visible = false;

      editingPairInfo.Number = (byte)editingNumberPair.Value;
      editingPairInfo.LectureHall = editingLectureHall.Text;
      editingPairInfo.NamePair = editingNamePair.Text;

      DayClick(activeDay);  // Отрисовать
    }
    // Удалить пару из расписания
    private void editingRemove_Click(object sender, EventArgs e)
    {
      panelEditingSchedule.Visible = false;

      for (byte i = 0; i < schedulesForDays[activeDay].Count; i++)
        if
(editingPairInfo.Schedule.Equals(schedulesForDays[activeDay][i]))
          schedulesForDays[activeDay].RemoveAt(i);

      editingPairInfo.Dispose();
    }
    // Закрыть окно редактирования без сохранения изменений
    private void editingClose_Click(object sender, EventArgs e)
    {
      panelEditingSchedule.Visible = false;
    }

    private string activeSchedule;  // Активное расписание
    // Загрузка расписания из файла
private void LoadSchedule(string scheduleName)
{
  string path = schedulePath + scheduleName + "\\" + scheduleName +
".sched";
  BinaryFormatter binFormat = new BinaryFormatter();

  // Сохранить объект в локальном файле.
  Stream fStream = new FileStream(path, FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
  schedulesForDays = (List<Schedule>[])binFormat.Deserialize(fStream);
  fStream.Close();
}
// Сохранение расписания в файл
private void SaveShedule(string scheduleName)
{
  string path = schedulePath + scheduleName + "\\" + scheduleName +
".sched";
  BinaryFormatter binFormat = new BinaryFormatter();
```

```
  // Сохранить объект в локальном файле.
  Stream fStream = new FileStream(path, FileMode.Create,
FileAccess.Write, FileShare.ReadWrite);
  binFormat.Serialize(fStream, schedulesForDays);
  fStream.Close();
}

    // Сохранение данных перед закрытием
    private void Form1_FormClosing(object sender, FormClosingEventArgs
e)
    {
      if (activeSchedule != null)
        SaveShedule(activeSchedule);
    }
    // Обработчик выбора расписания (comboBox)
    private void schedules_SelectedIndexChanged(object sender,
EventArgs e)
    {
      if (activeSchedule != null)
        SaveShedule(activeSchedule);

      if (activeSchedule != schedules.SelectedItem.ToString())
      {
        activeSchedule = schedules.SelectedItem.ToString();
        LoadSchedule(activeSchedule);
        DayClick(activeDay);
      }
    }
    // Клик по выбору расписания (comboBox)
    private void schedules_KeyDown(object sender, KeyEventArgs e)
    {

      // Если в окне выбора расписания ввели новое и нажали Enter
      if (e.KeyCode != Keys.Enter)
        return;

      schedules.Text = schedules.Text.Trim();
      string path = schedulePath + schedules.Text + "\\";
      // Проверка на существование расписания
      if (Directory.Exists(path))
      {
        foreach (var i in schedules.Items)
          if (i.ToString() == schedules.Text)
            schedules.SelectedItem = i;
        return;
      }

      // Спрашиваем, если такого расписания нет
      DialogResult result = MessageBox.Show(
        "Создать новое расписание?",
        "Сообщение",
```

```
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Question,
            MessageBoxDefaultButton.Button1,
            MessageBoxOptions.DefaultDesktopOnly);

          // Создаем новое рассписание
          if (result == DialogResult.Yes)
          {
            if (activeSchedule != null)
              SaveShedule(activeSchedule);

            Directory.CreateDirectory(path);
            activeSchedule = schedules.Text;

            ScheduleSetDafaultValue();
            schedules.Items.Add(schedules.Text);
            schedules.SelectedItem = schedules.Items.Count - 1;
          }
          // Иначе очищаем поле
          else { schedules.Text = ""; }
        }
      }
    }
```

Файл PairInfo.cs

```
using System;
using System.Windows.Forms;

namespace ClassScheduleSupportSystem
{
  public partial class PairInfo : UserControl
  {
    // Метод обработчик клика на пару в расписании
    public Form1.HandlerPairInfoClick PairInfoClick;

    private Schedule _schedule;
    public Schedule Schedule { get { return _schedule; } }
    public byte Number
    {
      get { return _schedule.Number; }
      set
      {
        _schedule.Number = value;
        number.Text = value.ToString();
      }
    }

    public string LectureHall
    {
      get { return _schedule.LectureHall; }
      set
      {
```

```
            _schedule.LectureHall = value;
            lectureHall.Text = value;
          }
        }

        public string NamePair
        {
          get { return _schedule.NamePair; }
          set
          {
            _schedule.NamePair = value;
            namePair.Text = value;
          }
        }

        public PairInfo(Schedule schedule)
        {
          InitializeComponent();
          _schedule = schedule;

          number.Text = schedule.Number.ToString();
          namePair.Text = schedule.NamePair;
          lectureHall.Text = schedule.LectureHall;

          var t = Schedule.GetTimePair(schedule.Number);
          timeStartEnd.Text = string.Format("{0}-{1}",
                                            t.Start.ToString("HH:mm"),
                                            t.End.ToString("HH:mm"));

          number.Click += PairInfo_Click;
          lectureHall.Click += PairInfo_Click;
          timeStartEnd.Click += PairInfo_Click;
          namePair.Click += PairInfo_Click;
        }

        public void PairInfo_Click(object sender, EventArgs e)
        {
          PairInfoClick(this);
        }
      }
    }
```

Файл Schedule.cs

```
using System;

namespace ClassScheduleSupportSystem
{
  [Serializable]
  public class Schedule
  {
```

```
    public byte Number { get; set; }
    public string LectureHall { get; set; }

    public string NamePair { get; set; }
    public static (DateTime Start, DateTime End) GetTimePair(byte
number)
    {
      DateTime t = DateTime.Parse("08:00");
      t = t.AddMinutes(100 * (number - 1));

      return (t, t.AddMinutes(90));
    }

    public Schedule(byte number, string lectureHall, string namePair)
    {
      Number = number;
      LectureHall = lectureHall;
      NamePair = namePair;
    }
  }
}
```

Файл WeekInCalendar.cs

```
using System;
using System.Globalization;
using System.Windows.Forms;

namespace ClassScheduleSupportSystem
{
  public partial class WeekInCalendar : UserControl
  {
    // Обработчик клика на конкретный день в расписании
    public Form1.HandlerCalendarDayClick CalendarDayClick
    {
      set
      {
        foreach (var control in calendarPanelDays.Controls)
          if (control.GetType() == typeof(DayInCalendar))
            ((DayInCalendar)control).CalendarDayClick = value;
      }
    }

    public DayInCalendar this[int index]
    {
      get
      {
        return (DayInCalendar)calendarPanelDays.Controls[index];
      }
    }
```

```
    DateTime dateStartWeek;
    public WeekInCalendar()
    {
      InitializeComponent();

      dateStartWeek = DateTime.Today.AddDays(-
(int)DateTime.Today.DayOfWeek + (int)DayOfWeek.Monday);
      UpdateWeek();
    }

    private void calendarButtonLeft_Click(object sender, EventArgs e)
    {
      dateStartWeek = dateStartWeek.AddDays(-7);
      UpdateWeek();

      for (byte i = 0; i < calendarPanelDays.Controls.Count; i++)
        if (this[i].Active) this[i].DayInCalendar_Click(null, null);
    }

    private void calendarButtonRight_Click(object sender, EventArgs e)
    {
      dateStartWeek = dateStartWeek.AddDays(7);
      UpdateWeek();

      for (byte i = 0; i < calendarPanelDays.Controls.Count; i++)
        if (this[i].Active) this[i].DayInCalendar_Click(null, null);
    }

    private void UpdateWeek()
    {
      calendarYear.Text = dateStartWeek.Year.ToString() +
                          ((dateStartWeek.Year ==
dateStartWeek.AddDays(6).Year) ?
                                  "" : " - " +
dateStartWeek.AddDays(6).Year.ToString());

      calendarMonth.Text = dateStartWeek.ToString("MMMM") +
                          ((dateStartWeek.Month ==
dateStartWeek.AddDays(6).Month) ?
                                  "" : " - " +
dateStartWeek.AddDays(6).ToString("MMMM"));

      byte i = 6;
      foreach (DayInCalendar day in calendarPanelDays.Controls)
        day.Date = dateStartWeek.AddDays(i--);
    }

    public byte GetDaySchedule(DateTime date)
    {
      var cal = new GregorianCalendar();
```

```
      byte weekNumber = (byte)cal.GetWeekOfYear(dateStartWeek,
CalendarWeekRule.FirstFullWeek, DayOfWeek.Monday);
      byte day = (byte)(date.DayOfWeek - 1);
      return (byte)(day + (((weekNumber & 1) == 1) ? 7 : 0));
    }
  }
}
```

Файл DayInCalendar.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Globalization;
using System.Windows.Forms;

namespace ClassScheduleSupportSystem
{
  public partial class DayInCalendar : UserControl
  {
    private static List<DayInCalendar> DaysInCalendar = new
List<DayInCalendar>();
    public Form1.HandlerCalendarDayClick CalendarDayClick;  // Метод
обработчик клика на день в календаре

    private bool _active;
    public bool Active
    {
      get { return _active; }
      set
      {
        if (_active != value)
        {
          if (value)
          {
            foreach (var dayInCalendar in DaysInCalendar)
              if (dayInCalendar.Active)
                dayInCalendar.Active = false;

            labelNumberDay.BackColor = SystemColors.ActiveCaption;
          }
          else labelNumberDay.BackColor = SystemColors.Control;
          _active = value;
        }
      }
    }

    private DateTime _date;
```

```
    [DisplayName(@"Date"), Description("Описание"),
Category("Данные"), DefaultValue(30)]
    public DateTime Date
    {
      get { return _date; }
      set
      {
        _date = value;
        labelNumberDay.Text = _date.Day.ToString();
        labelText.Text = _date.ToString("ddd").ToUpper();
      }
    }

    public DayInCalendar()
    {
      DaysInCalendar.Add(this);
      InitializeComponent();
      for (int i = 0; i < Controls.Count; i++)
        Controls[i].Click += DayInCalendar_Click;
    }

    public void DayInCalendar_Click(object sender, EventArgs e)
    {
      var cal = new GregorianCalendar();
      // Получаем номер недели
      byte weekNumber = (byte)cal.GetWeekOfYear(_date,
CalendarWeekRule.FirstFullWeek, DayOfWeek.Monday);
      // Номер дня недели
      byte day = (byte)(cal.GetDayOfWeek(_date));
      day = (byte)((day == 0) ? 6 : day - 1);
      // День по расписанию
      byte scheduleDay = (byte)(day + (((weekNumber & 1) == 1) ? 7 :
0));

      // Вызов метода редактирования расписания
      Active = true;
      CalendarDayClick(scheduleDay);
    }
  }
}
```

Приложение Б