

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования  
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
Кафедра «Измерительно-вычислительные комплексы»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
«Информационная система планирования переговоров»

Выполнил:  
студент гр. ИСТбд-41  
Шаповалов Александр Петрович  
Проверил:  
Шишкин Вадим Викторович

Ульяновск

2023

## Введение

В данной работе разработана и представлена система для бронирования переговоров, основанная на важнейших условиях для наилучшего понимания события: высокая скорость доступа к информации о событии.

Переговоры – один из ключевых аспектов любого бизнеса. При этом количество помещений (переговорных комнат) практически в любой компании сильно ограничено и от эффективности использования данного ресурса в компании зависит эффективность ее бизнеса. Несмотря на это, бронирование переговорных комнат в большинстве компаний происходит либо с помощью записей на бумаге, с использованием секретаря или, в лучшем случае, в корпоративных порталах, а иногда и по принципу «кто первый занял». По статистике, средняя загрузка переговорных комнат без использования автоматизированных систем бронирования составляет не более 50%, а внедрение таких систем позволяет увеличить ее минимум на 20%.

Преимущества систем бронирования переговорных:

Повышение эффективности использования офисных помещений;

Удобство бронирования переговорных и распределение загрузки переговорных комнат и конференц-залов;

Предоставление информации о запланированных мероприятиях, сбор статистики об использовании;

Сокращение расходов на электроэнергию и аудио-видео оборудование за счет эффективного использования ресурсов.

Так же стоит отметить стандартный перечень задач, которое предоставляет система бронирования:

- добавление новых событий;
- сортировка события по дате и времени;
- вывод подробной информации о событии;
- изменение и удаление событий;

С помощью данной информационной системы можно улучшить управление бронированием переговорки, путем сортировки событий, а также получение подробной информации о событии.

Данная база должна хранить в себе информацию о событии, его участниках, дате, а так же подробной информации. Так же в ней должна иметься возможность редактирования любой внутренней информации, а так

жедобавление новой и удаление старой/дублирующей информации. Вся совокупность этих данных позволит упорядочить рабочую неделю.

Для пользователя должна быть предоставлена возможность просмотра основной информации о событии, его участниках и подробной информации.

# **1 Техническое задание**

## **1.1 Общие сведения**

Система для бронирования переговоров, которая представляет из себя таблицу, где в качестве колонок числа и дни недели, а в качестве строк почасовой список (формат 00:00 – 00:24).

### **1.2 Назначение и цели создания системы**

#### **1.2.1 Назначение системы**

Система предназначена для удобного бронирования переговорки. Она будет доступна только сотрудникам. Они могут бронировать дату и время, изменять уже существующую бронь и просматривать подробную информацию о брони.

#### **1.2.2 Цели создания системы**

Цели, которые будут достигнуты в результате применения системы: 1)  
Предоставление сотрудникам возможности забронировать переговорку в определенную дату и время.

2) Упрощение получения информации о забронированной переговорки.

### **1.3 Требования к системе**

#### **1.3.1 Требования к системе в целом**

##### **1.3.1.1 Требования к структуре и функционированию системы**

Определяется общей постановкой задачи задания на курсовую работу.

##### **1.3.1.2 Требования к защите информации от несанкционированного доступа**

Типы пользователей в системе:

1) Сотрудник: он может просматривать, добавлять, изменять и удалять бронь переговоров. Требуется авторизация.

##### **1.3.2 Требования к функциям, выполняемым системой**

Функции, выполняемые системой:

1) Добавление брони на определённую дату и время;

2) Редактирование уже существующей брони;

3) Предоставление полной информации о существующей брони.

#### 1.4 Состав и содержание работ по созданию системы

Определяется этапами выполнения работы задания на курсовую работу.

#### 1.5 Порядок контроля и приёмки системы

Определяется порядком защиты и критериями оценки работы задания на курсовую работу.

## 2 Проектное решение

Особенности варианта:

1. Хранить данные пользователя, карточек событий и другие данные в **JavaCopyOnWriteArrayList** списках. Списки должны создаваться и заполняться тестовыми данными при старте приложения.
2. Фон страниц должен быть белого цвета (background)
3. Детали события карточки отображаются в всплывающем окне
4. Предусмотреть возможность отображения даты в виде timestamp

Хранить данные пользователя, карточек событий и другие данные в **JavaCopyOnWriteArrayList** списках.

Аналогом **JavaCopyOnWriteArrayList** в синтаксисе языка **C#** является **C# SynchronizedCollection**.

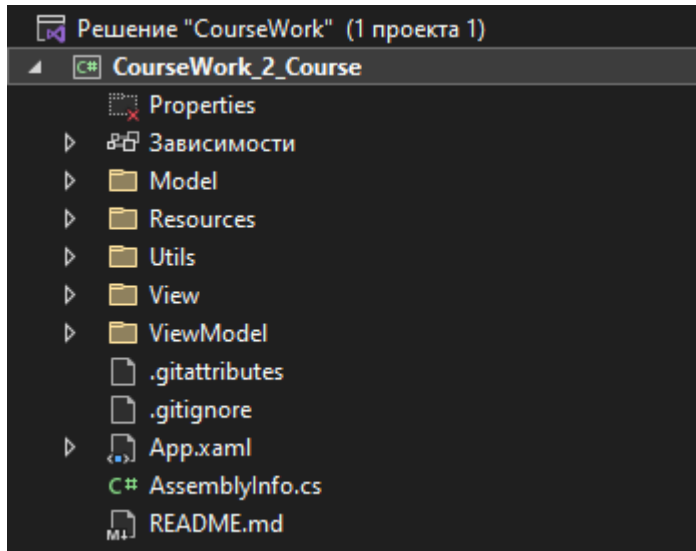
```
public static SynchronizedCollection<User> users = new();
public static bool AuthUser(string nickname, string password)
{
    using (ApplicationContextdb = new())
    {
        bool isAuthenticated = db.Users.Any(u => u.Nickname == nickname & u.Password ==
password);
        if (isAuthenticated) return true;
    }
    return false;
}
```

**SynchronizedCollection** предоставляет потокобезопасную коллекцию, содержащую объекты типа, заданного универсальным параметром, в качестве элементов

### 3. Реализация

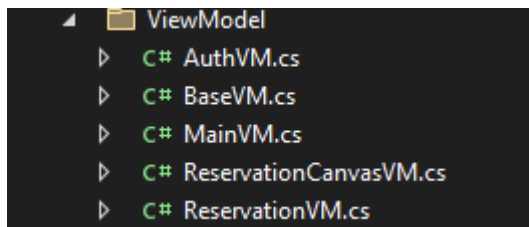
#### 3.1 Структура программного обеспечения

Структура проекта VisualStudio:



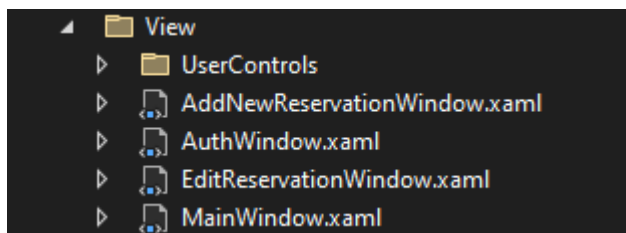
Всего в приложении 5 контроллеров, называемые в проекте «вью-модели» - паттерн MVVM (Model-View-ViewModel).

ViewModels

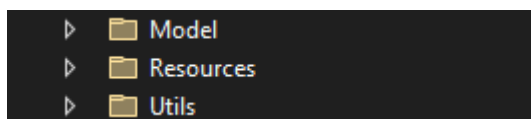


Каждый из них отвечает за одноимённую страницу(view).

В каталоге View лежат соответствующие представления (add, edit, index, show) для контроллера.



Каталоги Model, Resources и Utils содержат вспомогательные классы для работы с данными, ресурсы, необходимые для работы приложения и классы-утилиты соответственно. Контроль доступных для бронирования дней осуществляется в утилитах.





## 4 Руководство пользователя

### 4.1 Общие сведения

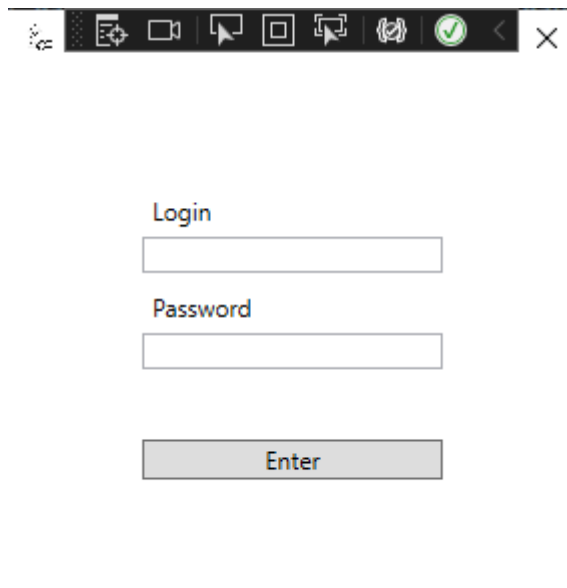
Система «Бронирования переговоров» предназначена для упрощения бронирования переговорки, получения подробной информации о брони. Сотрудники могут добавлять, редактировать и удалять брони.

Сотрудник должен иметь навык работы с:

- 1) ОС Windows;
- 2) Интернет-браузером;

### 4.2 Порядок и особенности работы

Страница авторизации выглядит следующим образом:

A screenshot of a web browser window showing a login form. The browser's address bar is at the top with various icons. The login form is centered and contains three elements: a text label 'Login' above a text input field, a text label 'Password' above another text input field, and a button labeled 'Enter' below the password field. The entire form is enclosed in a thin black border.

После ввода логина и пароля переходим на главную страницу. Главная страница выглядит следующим образом:

# Meet Room

Previous week

Next week

Admin



	Tuesday 14.11.2023	Wednesday 15.11.2023	Thursday 16.11.2023	Friday 17.11.2023	Saturday 18.11.2023	Sunday 19.11.2023	Monday 20.11.2023
00:00	+	+	+	+	+	+	+
00:01	+	+	+	+	+	+	+
00:02	+	+	+	+	+	+	+
00:03	+	+	+	+	+	+	+
00:04	+	+	+	+	+	Pavel	+
00:05	+	+	+	+	+	+	+
00:06	+	+	+	+	+	+	+
00:07	+	+	+	+	+	+	+
00:08	+	+	+	+	+	+	+
00:09	+	+	+	+	+	+	+
00:10	+	+	+	+	+	+	+
00:11	+	+	+	+	+	+	+
00:12	+	+	+	+	+	+	+
00:13	+	+	+	+	+	+	+
00:14	+	+	+	+	+	+	+
00:15	+	+	+	+	+	+	+
00:16	+	+	+	+	+	+	+
00:17	+	+	+	+	+	+	+
00:18	+	+	+	+	+	+	+
00:19	+	+	+	+	+	+	+
00:20	+	+	+	+	+	+	+
00:21	+	+	+	+	+	+	+
00:22	+	+	+	+	+	+	+
00:23	+	+	+	+	+	+	+
00:24	+	+	+	+	+	+	+

Страница добавления брони выглядит следующим образом:



Time from

00:00:00

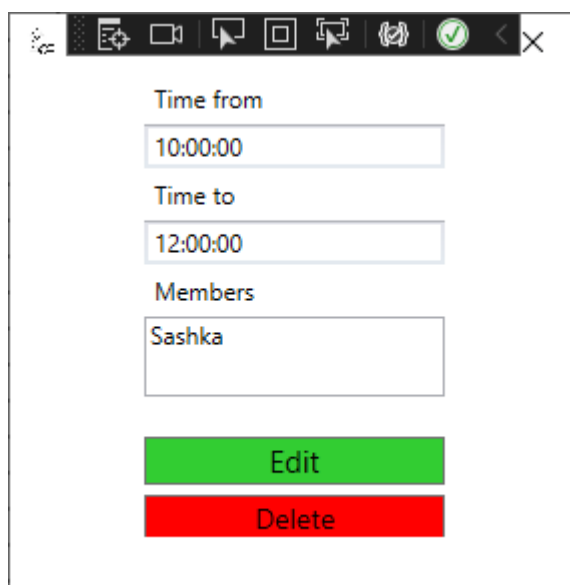
Time to

00:00:00

Members

Add

Страница подробной информации выглядит следующим образом:



The screenshot shows a web application window with a dark toolbar at the top containing icons for settings, video, screen sharing, window management, and a checkmark. The main content area is white and contains a form with the following fields:

- Time from**: A text input field containing "10:00:00".
- Time to**: A text input field containing "12:00:00".
- Members**: A text input field containing "Sashka".

Below the form fields are two buttons:

- Edit**: A green button with white text.
- Delete**: A red button with white text.

## 5 Тестирование

При нажатии кнопки «Add» создается новая бронь для Sasha и пользователя перемещает на главную страницу

Reservation

Meet Room

Previous week Next week

Admin

	Tuesday 14.11.2023	Wednesday 15.11.2023	Thursday 16.11.2023	Friday 17.11.2023	Saturday 18.11.2023	Sunday 19.11.2023	Monday 20.11.2023
00:00	+	+	+	+	+	+	+
00:01	+	+	+	+	+	+	+
00:02	+	+	+	+	+	+	+
00:03	+	+	+	+	+	+	+
00:04	+	+	+	Sashka	+	Pavel	+
00:05	+	+	Vital	+	+	+	+
00:06	+	+	+	+	+	+	+
00:07	+	+	+	+	+	+	+
00:08	+	+	+	+	+	+	+

Если пользователь попытается добавить бронь, которая меньше 30 мин или на время, которое уже занято, то появиться ошибка.

Time from

10:00:00

Time to

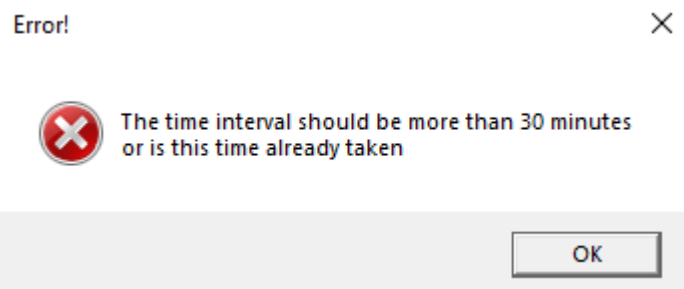
10:20:00

Members

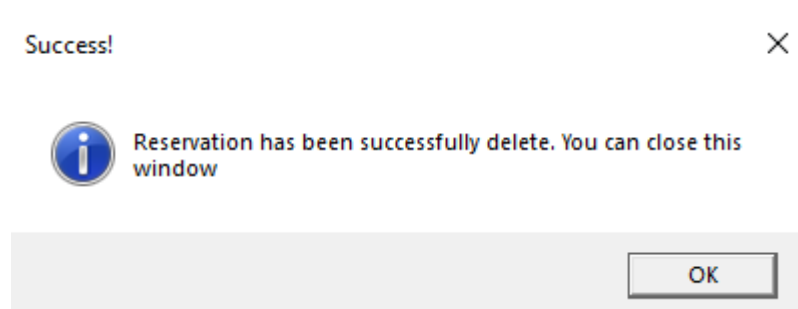
Maxim

Add

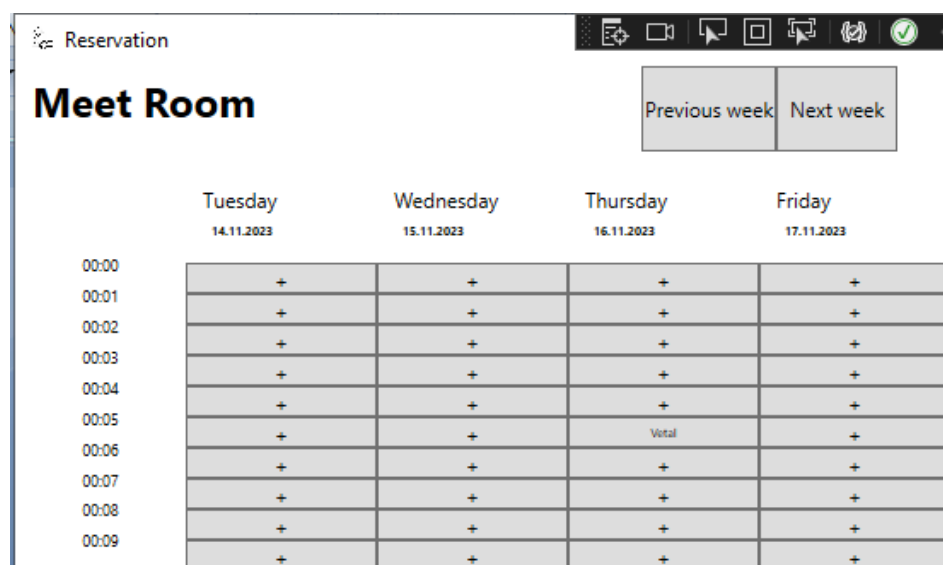
Ошибка:



Если пользователь нажмет на кнопку «Delete» то выбранное событие удалится



Если пользователь нажмет на кнопку «Previousweek» его перебросит на прошлую неделю





## Meet Room

Previous week

Next week

Admin



	Tuesday 07.11.2023	Wednesday 08.11.2023	Thursday 09.11.2023	Friday 10.11.2023	Saturday 11.11.2023	Sunday 12.11.2023	Monday 13.11.2023
00:00	+	+	+	+	+	+	+
00:01	+	+	+	+	+	+	+
00:02	+	+	+	+	+	+	+
00:03	+	+	+	+	+	+	+
00:04	+	+	+	+	+	+	+
00:05	+	+	+	+	+	+	+
00:06	+	+	+	+	+	+	+
00:07	+	+	+	+	+	+	+
00:08	+	+	+	+	+	+	+
00:09	+	+	+	+	+	+	+
00:10	+	+	+	+	+	+	+
00:11	+	+	+	+	+	+	+
00:12	+	+	+	+	+	+	+
00:13	+	+	+	+	+	+	+
00:14	+	+	+	+	+	+	+
00:15	+	+	+	+	+	+	+
00:16	+	+	+	+	+	+	+
00:17	+	+	+	+	+	+	+
00:18	+	+	+	+	+	+	+
00:19	+	+	+	+	+	+	+
00:20	+	+	+	+	+	+	+
00:21	+	+	+	+	+	+	+
00:22	+	+	+	+	+	+	+
00:23	+	+	+	+	+	+	+
00:24	+	+	+	+	+	+	+

Если пользователь нажмет на кнопку «Nextweek» его перебросит на следующую неделю

# Meet Room

[Previous week](#) [Next week](#)

Admin



	Tuesday 21.11.2023	Wednesday 22.11.2023	Thursday 23.11.2023	Friday 24.11.2023	Saturday 25.11.2023	Sunday 26.11.2023	Monday 27.11.2023
00:00	+	+	+	+	+	+	+
00:01	+	+	+	+	+	+	+
00:02	+	+	+	+	+	+	+
00:03	+	+	+	+	+	+	+
00:04	+	+	+	+	+	+	+
00:05	+	+	+	+	+	+	+
00:06	+	+	+	+	+	+	+
00:07	+	+	+	+	+	+	+
00:08	+	+	+	+	+	+	+
00:09	+	+	+	+	+	+	+
00:10	+	+	+	+	+	+	+
00:11	+	+	+	+	+	+	+
00:12	+	+	+	+	+	+	+
00:13	+	+	+	+	+	+	+
00:14	+	+	+	+	+	+	+
00:15	+	+	+	+	+	+	+
00:16	+	+	+	+	+	+	+
00:17	+	+	+	+	+	+	+
00:18	+	+	+	+	+	+	+
00:19	+	+	+	+	+	+	+
00:20	+	+	+	+	+	+	+
00:21	+	+	+	+	+	+	+
00:22	+	+	+	+	+	+	+
00:23	+	+	+	+	+	+	+
00:24	+	+	+	+	+	+	+

Если пользователь при авторизации введет неверные логин или пароль  
появится ошибка

Error



WRONG DATA!

OK

**Код:**

## **ReservationService.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace CourseWork.Model.Data.Service
{
    public class ReservationService
    {
        public static List<Reservation> GetAllReservations()
        {
            using ApplicationContext db = new();
            List<Reservation> reservations = db.Reservations.ToList();
            return reservations;
        }

        public static void CreateReservation(int gridRow,
                                             int gridColumn,
                                             string page,
                                             string user,
                                             string members,
                                             TimeSpan timeFrom,
                                             TimeSpan timeTo)
        {
            using ApplicationContext db = new();
            Reservation reservation = new()
            {
                GridColumn = gridColumn,
                GridRow = gridRow,
                Page = page,
                User = user,
                Members = members,
                TimeFrom = timeFrom,
                TimeTo = timeTo
            };
            db.Reservations.Add(reservation);
            db.SaveChanges();
        }

        public static void EditReservation(int gridRow,
                                           int gridColumn,
                                           string page,
                                           string user,
                                           string members,
                                           TimeSpan timeFrom,
                                           TimeSpan timeTo)
        {
            using (ApplicationContext db = new())
            {
                #pragma warning disable CS8600
                Reservation reservation = db.Reservations.FirstOrDefault(r => r.GridRow == gridRow
                                                                              && r.GridColumn == gridColumn
                                                                              && r.Page.ToLower() == page.ToLower()
                                                                              && r.User == user);

                reservation.Members = members;
                reservation.TimeFrom = timeFrom;
                reservation.TimeTo = timeTo;
                db.SaveChanges();
            }
        }

        public static bool isReservationExist(int gridRow, int gridColumn, string page, string user)
        {
            using (ApplicationContext db = new())
```



```

    {
        bool isExist = db.Reservations.Any(r => r.GridRow == gridRow
            && r.GridColumn == gridColumn
            && r.Page.ToLower() == page.ToLower()
            && r.User == user);
        if (isExist) return true;
    }
    return false;
}
public static Reservation GetReservationInfo(int gridRow, int gridColumn, string page, string user)
{
    using (ApplicationContext db = new())
    {
        Reservation reservation = db.Reservations.FirstOrDefault(r => r.GridRow == gridRow
            && r.GridColumn == gridColumn
            && r.Page.ToLower() == page.ToLower()
            && r.User == user);
        return reservation ?? new Reservation() { GridColumn = gridColumn, GridRow = gridRow, Page =
page, User = user };
    }
}
public static void DeleteReservation(int gridRow, int gridColumn, string page, string user)
{
    using (ApplicationContext db = new())
    {
        Reservation reservation = db.Reservations.FirstOrDefault(r => r.GridRow == gridRow
            && r.GridColumn == gridColumn
            && r.Page.ToLower() == page.ToLower()
            && r.User == user);

        db.Reservations.Remove(reservation);
        db.SaveChanges();
    }
}
}
}
}

```

## UserService.cs

```

using System.Linq;

namespace CourseWork.Model.Data.Service
{
    public class UserService
    {
        public static bool AuthUser(string nickname, string password)
        {
            using (ApplicationContext db = new())
            {
                bool isAuthenticated = db.Users.Any(u => u.Nickname == nickname & u.Password == password);
                if (isAuthenticated) return true;
            }
            return false;
        }
    }
}

```

```
}  
}
```

## ApplicationContext.cs

```
using Microsoft.EntityFrameworkCore;  
using System;  
  
namespace CourseWork.Model.Data  
{  
    public class ApplicationContext : DbContext  
    {  
        public DbSet<User> Users { get; set; }  
        public DbSet<Reservation> Reservations { get; set; }  
        public ApplicationContext()  
        {  
            Database.EnsureCreated();  
        }  
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  
        {  
            optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=CourseWork;Trusted_Connection=True;");  
        }  
    }  
}
```

## RelayCommand.cs

```
using System;  
using System.Windows.Input;  
  
namespace CourseWork.Model  
{  
    public class RelayCommand : ICommand  
    {  
        private Action<object> _execute;  
        private Func<object, bool> _canExecute;  
        public event EventHandler? CanExecuteChanged  
        {  
            add { CommandManager.RequerySuggested += value; }  
            remove { CommandManager.RequerySuggested -= value; }  
        }  
    }  
}
```

```

public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)
{
    _execute = execute;
    _canExecute = canExecute;
}

public bool CanExecute(object? parameter)
{
    return _canExecute == null || _canExecute(parameter);
}

public void Execute(object? parameter)
{
    _execute(parameter);
}
}
}

```

## Reservation.cs

```

using System;

namespace CourseWork.Model
{
    public class Reservation
    {
        public int Id { get; set; }
        public int GridColumn { get; set; }
        public int GridRow { get; set; }
        public string Page { get; set; }
        public string User { get; set; }
        public TimeSpan TimeFrom { get; set; }
        public TimeSpan TimeTo { get; set; }
        public string? Members { get; set; }
    }
}

```

```
}  
  
}
```

## User.cs

```
using Microsoft.EntityFrameworkCore;  
  
namespace CourseWork.Model  
{  
    public class User  
    {  
        public int Id { get; set; }  
  
        [Comment("The nickname of user")]  
        public string Nickname { get; set; }  
  
        [Comment("The password of user")]  
        public string Password { get; set; }  
    }  
}
```

## CommonUtils.cs

```
using System.Windows;  
  
namespace CourseWork.Utils  
{  
    public class CommonUtil  
    {  
        public static void OpenWindow(Window window)  
        {  
            window.Owner = Application.Current.MainWindow;  
            window.Show();  
        }  
    }  
}
```

## FillUtils.cs

```
using CourseWork.Model.Data;  
using CourseWork.ViewModel;  
using System;  
using System.Windows;  
using System.Windows.Controls;  
using System.Windows.Controls.Primitives;  
using CourseWork.View;  
using CourseWork.Model.Data.Service;  
  
namespace CourseWork.Utils  
{  
    public class FillUtil  
    {  
        private readonly static DateTime TODAY = DateTime.Today;  
        /// <summary>
```

```

/// Pointer to current content grid
/// </summary>
public static Grid CONTENT_GRID;
/// <summary>
/// Pointer to current page
/// </summary>
public static string PAGE;

public static int COLUMN, ROW;
public static void FillContentGrid(Grid contentGrid, string page)
{
    CONTENT_GRID = contentGrid;
    UIElement ui = new Button();
    for (int row = 0; row < 24; row++)
    {
        PAGE = page;
        for (int column = 0; column < 7; column++)
        {
            var dayOfWeek = TODAY.AddDays(column).DayOfWeek;
            if (dayOfWeek == DayOfWeek.Saturday | dayOfWeek == DayOfWeek.Tuesday | dayOfWeek ==
DayOfWeek.Thursday | dayOfWeek == DayOfWeek.Wednesday | dayOfWeek == DayOfWeek.Friday |
dayOfWeek == DayOfWeek.Sunday | dayOfWeek == DayOfWeek.Monday)
            {
                if (ReservationService.isReservationExist(row, column, page, AuthVM.Nickname))
                {
                    ui = new Button() { Content = ReservationService.GetReservationInfo(row, column, page,
AuthVM.Nickname).Members, FontSize = 7 };
                    ((Button)ui).Click += OpenEditReservationWnd;
                }
                else
                {
                    ui = new Button() { Content = "+" };
                    ((Button)ui).Click += OpenNewReservationWnd;
                }
                contentGrid.Children.Add(ui);
                Grid.SetColumn(ui, column);
                Grid.SetRow(ui, row);
            }
        }
    }
}

public static void FillDaysOfWeek(UniformGrid contentHeaderGrid, int from, int to)
{
    for (int i = from; i < to; i++)
    {
        StackPanel stackPanel = new() { Orientation = Orientation.Vertical, Margin = new Thickness(10, 0, 0,
0) };
        stackPanel.Children.Add(new TextBlock() { Text = TODAY.AddDays(i).DayOfWeek.ToString() });
        stackPanel.Children.Add(new Label() { Content = TODAY.AddDays(i).ToString("d"), FontWeight =
FontWeights.Bold, FontSize = 7 });
        contentHeaderGrid.Children.Add(stackPanel);
    }
}

public static void FillHoursStackPanel(StackPanel stackPanel)
{
    for (int i = 0; i < 25; i++)
    {
        TextBlock tb = new()
        {
            FontSize = 9,
            HorizontalAlignment = HorizontalAlignment.Center,
            Margin = new Thickness(0, 0, 0, 6)
        };
        switch (i)
        {
            case < 10:

```

```

        tb.Text = $"00:0{i}";
        break;
    case >= 10:
        tb.Text = $"00:{i}";
        break;
    default:
    }
    stackPanel.Children.Add(tb);
}

}

private static void OpenNewReservationWnd(object sender, RoutedEventArgs e)
{
    COLUMN = Grid.GetColumn(sender as Button);
    ROW = Grid.GetRow(sender as Button);

    AddNewReservationWindow newReservationWindow = new();
    newReservationWindow.ShowDialog();
}

private static void OpenEditReservationWnd(object sender, RoutedEventArgs e)
{
    COLUMN = Grid.GetColumn(sender as Button);
    ROW = Grid.GetRow(sender as Button);

    EditReservationWindow editReservationWindow = new();
    editReservationWindow.ShowDialog();
}
}
}

```

## AddNewReservationWindow.xaml

```

using CourseWork.ViewModel;
using System.Windows;

namespace CourseWork.View
{
    public partial class AddNewReservationWindow : Window
    {
        public AddNewReservationWindow()
        {
            InitializeComponent();
            DataContext = new ReservationVM();
        }
    }
}

```

## AuthWindow.xaml

```
using CourseWork.ViewModel;
using System.Windows;

namespace CourseWork.View
{
    public partial class AuthWindow : Window
    {
        public AuthWindow()
        {
            InitializeComponent();
            DataContext = new AuthVM();
        }
    }
}
```

### **EditReservationWindow.xaml**

```
using CourseWork.ViewModel;
using System.Windows;

namespace CourseWork.View
{
    public partial class EditReservationWindow : Window
    {
        public EditReservationWindow()
        {
            InitializeComponent();
            DataContext = new ReservationVM();
        }
    }
}
```

### **MainWindow.xaml**

```
using CourseWork.ViewModel;
using System.Windows;

namespace CourseWork.View
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            DataContext = new MainVM();
        }
    }
}
```

## **Заключение**

Разработанная в рамках курсовой работы система полностью соответствует предъявляемым к ней требованиям технического задания. Большая часть заявленных функций реализованы и корректно работают, некоторые алгоритмы функционируют с небольшими оговорками без существенного нарушения логики.

Поддерживается логическая и ссылочная ценность, корректный ввод данных, доступ к изменению данных. Поэтому серьезных ошибок в работе приложения возникать не должно.



Особые трудности возникали только при работе с данными, а также при обеспечении их корректной работы. Также сложности были при создании представлений для этих данных (таких, чтобы они были удобны и интуитивно понятны для пользователей), а также при передаче модели в Get- и Post- методах (не всегда данные передавались корректно), но эти трудности в основном были преодолены.

### **Список использованной литературы**

1)Руководство по программированию на C# [Электронный ресурс] - <https://docs.microsoft.com/ru-ru/dotnet/csharp/progra..> (Дата обращения 17.11.2021)

2)Введение в WPF [Электронный ресурс] - <https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/i..> (Дата обращения 27.11.2021)

3)MVVM: полное понимание [Электронный ресурс] - <https://habr.com/ru/post/338518/> (Дата обращения 27.11.2021)

4)Введение в EntityFramework [Электронный ресурс] -

HYPERLINK

"<https://vk.com/away.php?utf=1&to=https%3A%2F%2Fmetanit.com%2Fsharp%2Fentityframe>

work%2F1.1.php" \n \_blank<https://metanit.com/sharp/entityframework/1.1.php> (Дата

о  
б  
р  
а  
щ  
е  
н  
и  
я

0  
5

·  
1  
2

·  
2  
0  
2