Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

ОТЧЕТ

к лабораторной работе №1 по дисциплине:

|  |
| --- |
| «Объектно-ориентированные базы данных» |
| База данных DB4O |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | ИСМб-19-1 |  |  |  | Вовиков Д. Е. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил: | доцент |  |  |  | Харахинов В. А. |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск. 2022 г.

Содержание

[Индивидуальное задание 3](#_Toc116088454)

[База данных и объектные классы 4](#_Toc116088455)

[Клиентская часть приложения 13](#_Toc116088456)

[Листинг программы 21](#_Toc116088457)

Индивидуальное задание

**Задание**: разработать настольное приложение с использованием объектно-ориентированной базы данных.

**Тренажерный зал**

Имеются сотрудники (id, адрес сотрудника, телефон сотрудника, имя сотрудника, фамилия сотрудника, отчества), зал (id, зал), клиент (id, адрес клиента, телефон клиента, имя клиента, фамилия клиента, имя клиента, отчество клиента), абонемент (id, клиент, услуга, старт карты, конец карты, цена абонемента), расписание (id, сотрудник, зал, клиент, дата занятия, продолжительность занятия).

Согласно разработанным таблицам, была спроектирована соответствующая схема базы данных.

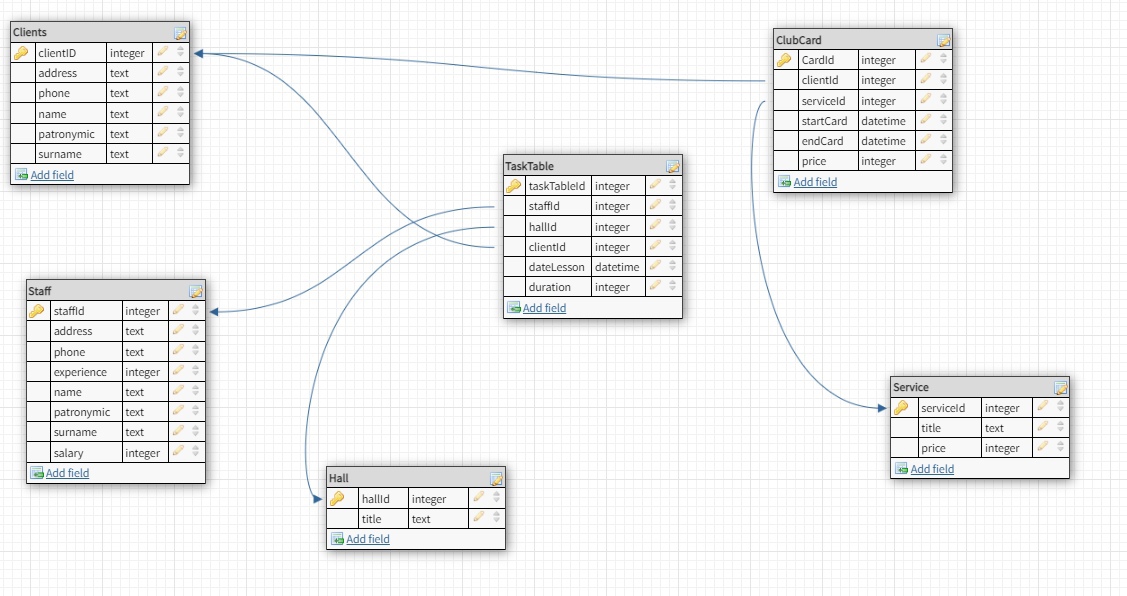


Рисунок 1 – Схема базы данных

База данных и объектные классы

В качестве объектно-ориентированной базы данных данных используется db4o.

Код подключения к БД:

DataBase dataBase;

dataBase = new DataBase();

dataBase.dbConnect();

Механизмы удаления, добавления, обновления и поиска, а также полноценное подключение к базе данных и отключение от нее реализовано в классе DataBase

Класс DataBase:

internal class DataBase

{

const String filename = @"gym.yap";

IObjectContainer db;

public void dbConnect()

{

db = Db4oFactory.OpenFile(filename);

}

public void dbClose()

{

db.Close();

}

#region Работа с услугой

public void AddService(Service service)

{

db.Store(service);

}

public List<Service> GetService()

{

return db.Query<Service>().ToList();

}

public void DeleteService(string id)

{

var service = db.Query<Service>(src => src.id == id)[0];

db.Delete(service);

}

public void UpdateService(Service service)

{

var found = db.Query<Service>(src => src.id == service.id)[0];

found.Price = service.Price;

found.Title = service.Title;

db.Store(found);

}

#endregion

#region Работа с залами

public void AddHall(Hall hall)

{

db.Store(hall);

}

public void DeleteHall(string id)

{

var found = db.Query<Hall>(hl => id == hl.id)[0];

db.Delete(found);

}

public List<Hall> GetHall()

{

var list = db.Query<Hall>().ToList();

return list;

}

public void UpdateHall(Hall hall)

{

var found = db.Query<Hall>(hl => hl.id == hall.id)[0];

found.title = hall.title;

db.Store(found);

}

#endregion

#region работа с клиентами

public void AddClient(Client client)

{

db.Store(client);

}

public List<Client> GetClient()

{

return db.Query<Client>().ToList();

}

public void DeleteClient(string id)

{

var found = db.Query<Client>(clt => clt.id == id)[0];

db.Delete(found);

}

public void UpdateClient(Client client)

{

var found = db.Query<Client>(clt => clt.id == client.id)[0];

found.phone = client.phone;

found.adress = client.adress;

found.surname = client.surname;

found.name = client.name;

found.patronymic = client.patronymic;

db.Store(found);

}

#endregion

#region Работа с абониментом

public void AddClubCard(ClubCard clubCard)

{

db.Store(clubCard);

}

public void DeleteClubCard(string id)

{

var found = db.Query<ClubCard>(cc => cc.id == id)[0];

db.Delete(found);

}

public List<ClubCard> GetClubCard()

{

return db.Query<ClubCard>().ToList();

}

#endregion

#region

public void AddStaff(Staff staff)

{

db.Store(staff);

}

public List<Staff> GetStaff() => db.Query<Staff>().ToList();

public void DeleteStaff(string id)

{

var found = db.Query<Staff>(stf => stf.id == id)[0];

db.Delete(found);

}

public void UpdateStaff(Staff staff)

{

var found = db.Query<Staff>(stf => stf.id == staff.id)[0];

found.name = staff.name;

found.phone = staff.phone;

found.surname = staff.surname;

found.adress = staff.adress;

found.patronymic = staff.patronymic;

db.Store(found);

}

#endregion

#region

public void AddTaskTable(TaskTable taskTable)

{

db.Store(taskTable);

}

public void DeleteTaskTable(string id)

{

var found = db.Query<TaskTable>(tt => tt.id == id)[0];

db.Delete(found);

}

public List<TaskTable> GetTaskTable() => db.Query<TaskTable>().ToList();

#endregion

#region Поиски

public List<ClubCard> ClubCardSearch(int dayCount)

{

var ans = (from ClubCard cc in db.Cast<ClubCard>()

where (cc.endCard - DateTime.Now).TotalDays <= dayCount

select cc).ToList<ClubCard>();

return ans;

}

public List<Service> ServiceSearch(int price, string title)

{

IQuery query = db.Query();

query.Constrain(typeof(Service));

IQuery titleQuery = query.Descend("\_title");

query.Descend("\_price").Constrain(price).Greater().Equal().And(

titleQuery.Constrain(title).Like());

IObjectSet result = query.Execute();

var soda = new List<Service>();

foreach (var r in result)

{

soda.Add(r as Service);

}

return soda;

}

public List<Search> Search() {

var asn = from TaskTable tt in db.Cast<TaskTable>()

group tt by tt.staff into t

select new { Name = t.Key,

Count = t.Count() }; ;

List<Search> lists = new List<Search>();

foreach(var an in asn)

{

lists.Add(new Classes.Search(an.Count, an.Name));

}

return lists;

}

internal void ServiceSearch(decimal value)

{

throw new NotImplementedException();

}

#endregion

}

Приложение содержит в себе шесть объектных классов.

Класс Client:

internal class Client

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Адрес клиента")]

public string adress { get; set; }

[DisplayName("Телефон клиента")]

public string phone { get; set; }

[DisplayName("Имя клиента")]

public string name { get; set; }

[DisplayName("Фамилия клиента")]

public string surname { get; set; }

[DisplayName("Отчество клиента")]

public string patronymic { get; set; }

public Client()

{

id = Guid.NewGuid().ToString();

}

public Client(string adress, string phone, string name, string surname, string patronymic)

{

id = Guid.NewGuid().ToString();

this.adress = adress;

this.phone = phone;

this.name = name;

this.surname = surname;

this.patronymic = patronymic;

}

public override bool Equals(object obj)

{

return obj is Client client &&

id == client.id &&

adress == client.adress &&

phone == client.phone &&

name == client.name &&

surname == client.surname &&

patronymic == client.patronymic;

}

public override string ToString()

{

return surname;

}

}

Класс ClubCard:

internal class ClubCard

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Клиент")]

public Client client { get; set; }

[DisplayName("Услуга")]

public Service service { get; set; }

[DisplayName("Старт карты")]

public DateTime startCard { get; set; }

[DisplayName("Конец карты")]

public DateTime endCard { get; set; }

[DisplayName("Цена абонемента")]

public int price { get; set; }

public ClubCard()

{

id = Guid.NewGuid().ToString();

}

public ClubCard(Client client, Service service, DateTime startCard, DateTime endCard, int price)

{

id = Guid.NewGuid().ToString();

this.client = client;

this.service = service;

this.startCard = startCard;

this.endCard = endCard;

this.price = price;

}

}

Класс Hall:

internal class Hall

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Зал")]

public string title { get; set; }

public Hall()

{

id = Guid.NewGuid().ToString();

}

public Hall(string title)

{

id = Guid.NewGuid().ToString();

this.title = title;

}

public override bool Equals(object obj)

{

return obj is Hall hall &&

id == hall.id &&

title == hall.title;

}

public override string ToString()

{

return title;

}

}

Класс Service:

internal class Service

{

[DisplayName("id")]

public string id { get; set; }

private string \_title;

[DisplayName("Название услуги")]

public string Title

{

get { return \_title; }

set { \_title = value; }

}

private int \_price;

[DisplayName("Цена")]

public int Price

{

get { return \_price; }

set { \_price = value; }

}

public Service()

{

id = Guid.NewGuid().ToString();

}

public Service(string title, int price)

{

id = Guid.NewGuid().ToString();

this.\_title = title;

this.\_price = price;

}

public override bool Equals(object obj)

{

return obj is Service service &&

id == service.id &&

\_title == service.Title &&

\_price == service.\_price;

}

public override string ToString()

{

return Title;

}

}

Класс Staff:

internal class Staff

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Адрес сотрудника")]

public string adress { get; set; }

[DisplayName("Телефон сотрудника")]

public string phone { get; set; }

[DisplayName("Имя сотрудника")]

public string name { get; set; }

[DisplayName("Фамилия сотрудника")]

public string surname { get; set; }

[DisplayName("Отчество сотрудника")]

public string patronymic { get; set; }

public Staff()

{

id = Guid.NewGuid().ToString();

}

public Staff(string adress, string phone, string name, string surname, string patronymic)

{

id = Guid.NewGuid().ToString();

this.adress = adress;

this.phone = phone;

this.name = name;

this.surname = surname;

this.patronymic = patronymic;

}

public override bool Equals(object obj)

{

return obj is Staff staff &&

id == staff.id &&

adress == staff.adress &&

phone == staff.phone &&

name == staff.name &&

surname == staff.surname &&

patronymic == staff.patronymic;

}

public override string ToString()

{

return surname;

}

}

Класс TaskTable:

internal class TaskTable

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Сотрудник")]

public Staff staff { get; set; }

[DisplayName("Зал")]

public Hall hall { get; set; }

[DisplayName("Клиент")]

public Client client { get; set; }

[DisplayName("Дата занятия")]

public DateTime dateLesson { get; set; }

[DisplayName("Продолжительность занятия")]

public int duration { get; set; }

public TaskTable()

{

id = Guid.NewGuid().ToString();

}

public TaskTable(Staff staff, Hall hall, Client client, DateTime dateLesson, int duration)

{

id = Guid.NewGuid().ToString();

this.staff = staff;

this.hall = hall;

this.client = client;

this.dateLesson = dateLesson;

this.duration = duration;

}

}

Клиентская часть приложения

Для реализации клиентской части приложения был использован язык C# со средой разработки Visual Studio Community, с подключенным модулем db4o для работы с объектно-ориентированной базой данных.

Интерфейс приложения предоставляет возможность взаимодействия с шестью таблицами: таблицей Услуги (Service), таблицей Залы (Hall), таблицей Клиент (Client), таблицей Абонемент (ClubCard), таблицей Сотрудники (Staff), таблицей Расписание (TaskTable) и таблицей поиска с использованием Soda-запроса.

Для переключения между таблицами используется верхнее меню (с помощью элемента управления MenuStrip).

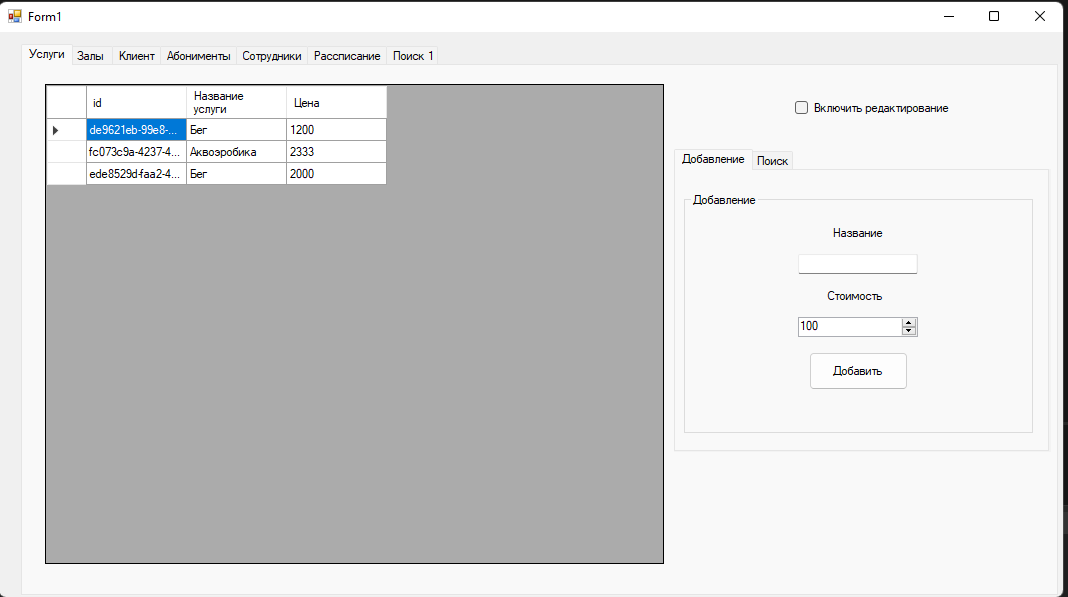


Рисунок 2 – Интерфейс для взаимодействия с таблицей Услуги

При попытке добавить запись с незаполненными полями отобразится MessageBox с ошибкой.

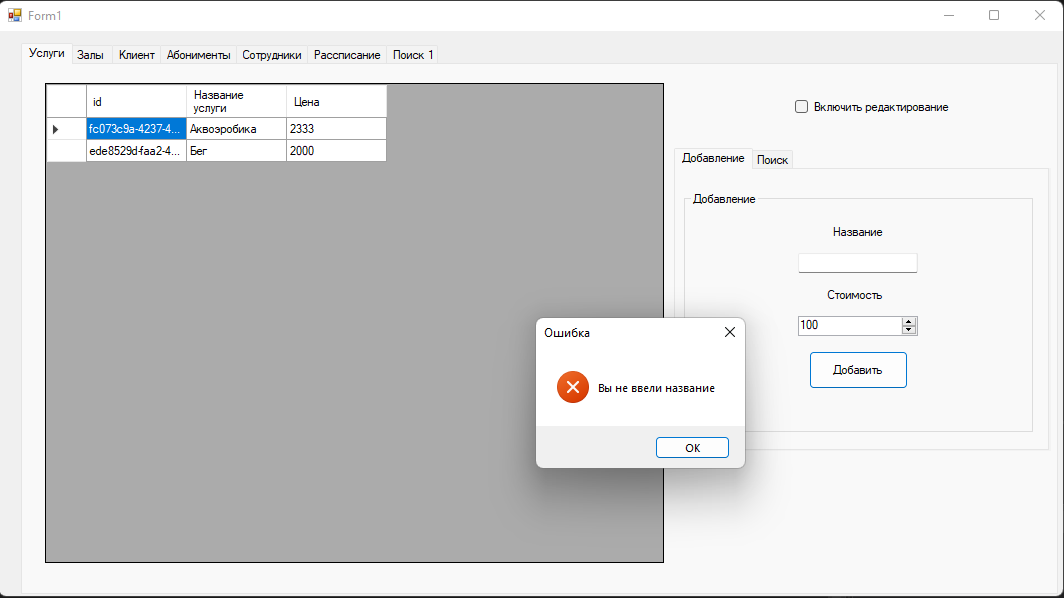


Рисунок 3 – Неверное добавление записи

При удачном добавлении записи отобразится MessageBox, уведомляющий об успешной записи.

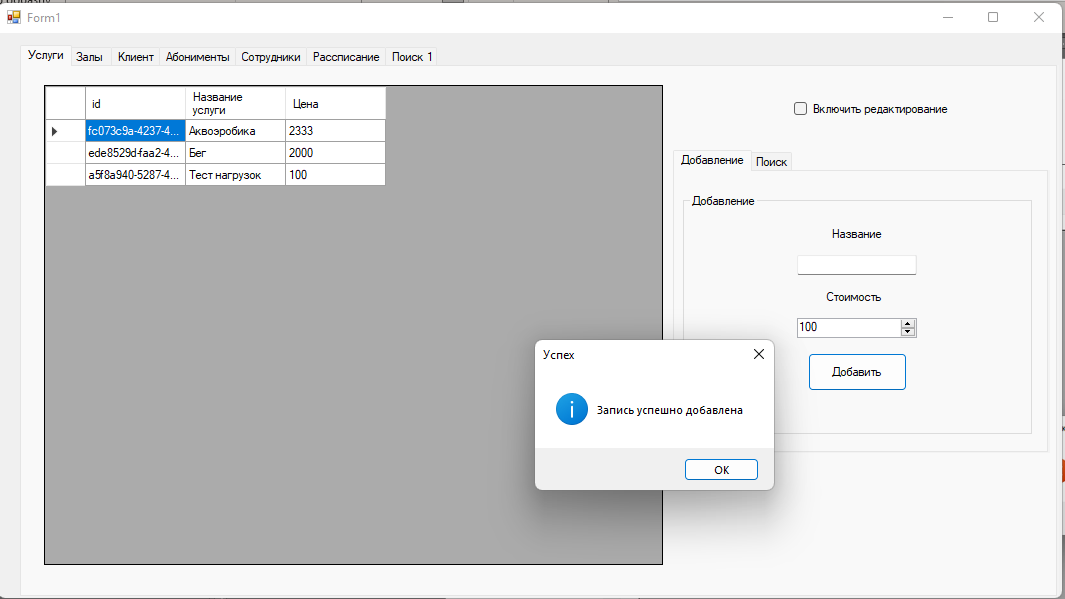


Рисунок 4 – Успешное добавление записи

Редактирование записей активируется с помощью соответствующего чекбокса.

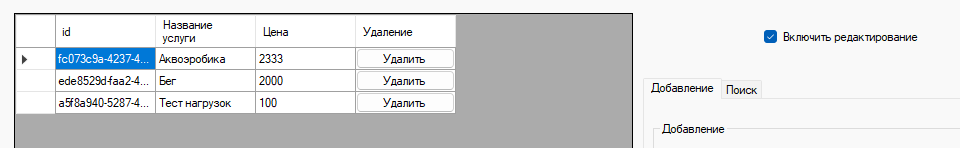


Рисунок 5 – Режим редактирования таблицы

В режиме редактирования доступна возможность редактирования полей (кроме поля id) и удаления записей.

При редактировании появляется MessageBox с согласием на изменение записи. На рисунке 6 показан процесс изменения значения цены первой строки.

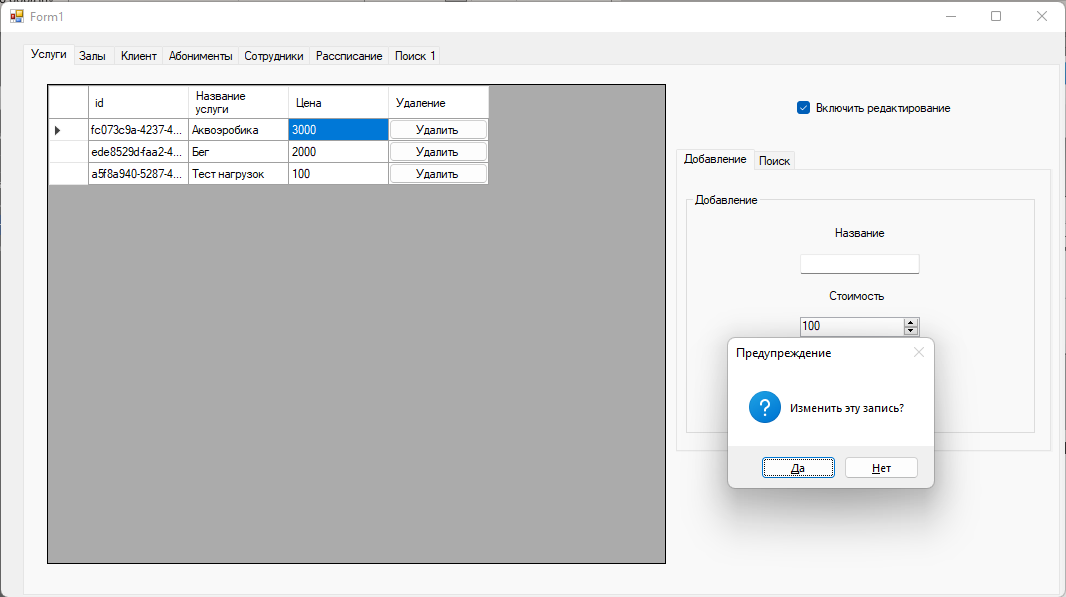


Рисунок 6 – Редактирование записи

В случае согласия на изменение запись в таблице и базе данных изменится.

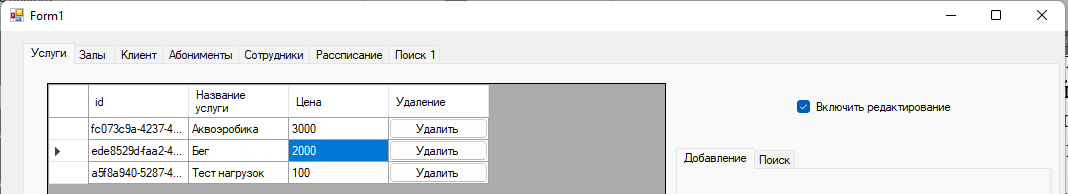


Рисунок 7 – Успешное изменение записи

Аналогичное сообщение отображается и при удалении записи.

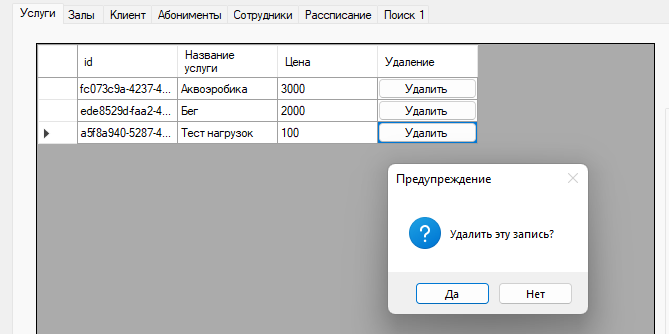


Рисунок 8 – Удаление записи

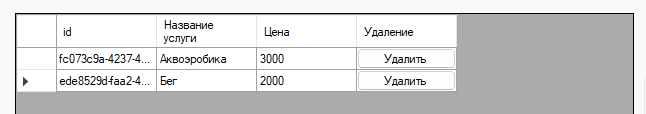


Рисунок 9 – Успешное удаление записи

Панель поиска предлагает возможность услуг от заданной стоимости или по названию услуги.

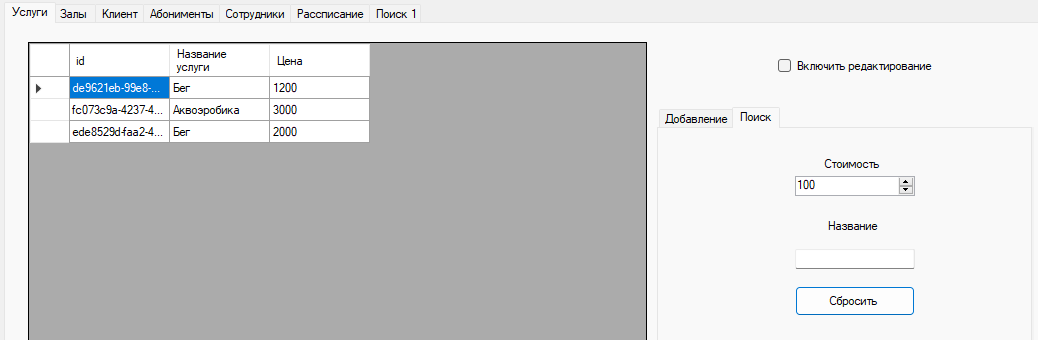


Рисунок 10 – Панель поиска

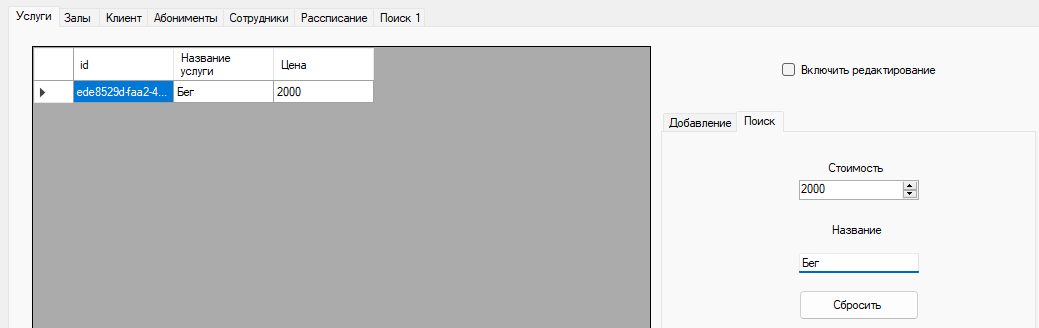


Рисунок 11 – Поиск записей с ценой от 2000 и названием «Бег»

Таблицы Залы, Клиент и Сотрудники включают в себя аналогичные возможности добавления, редактирования и удаления записей. Аналогично таблице Услуги функции редактирования и удаления сопровождаются выводом соответствующих сообщений. Механизмов поиска данные таблицы не предусматривают.

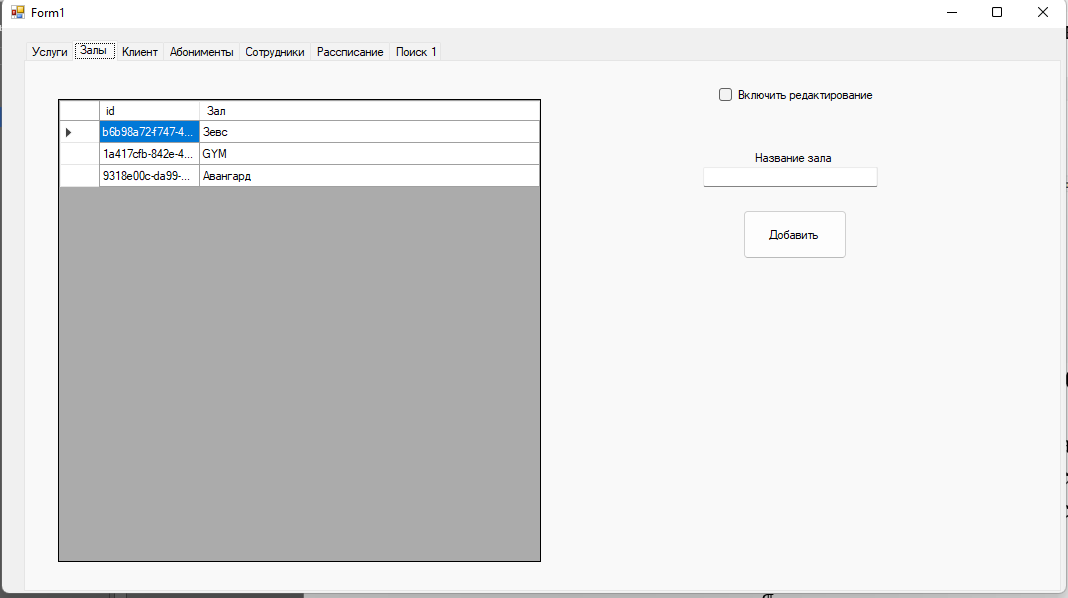


Рисунок 12 – Таблица Залы

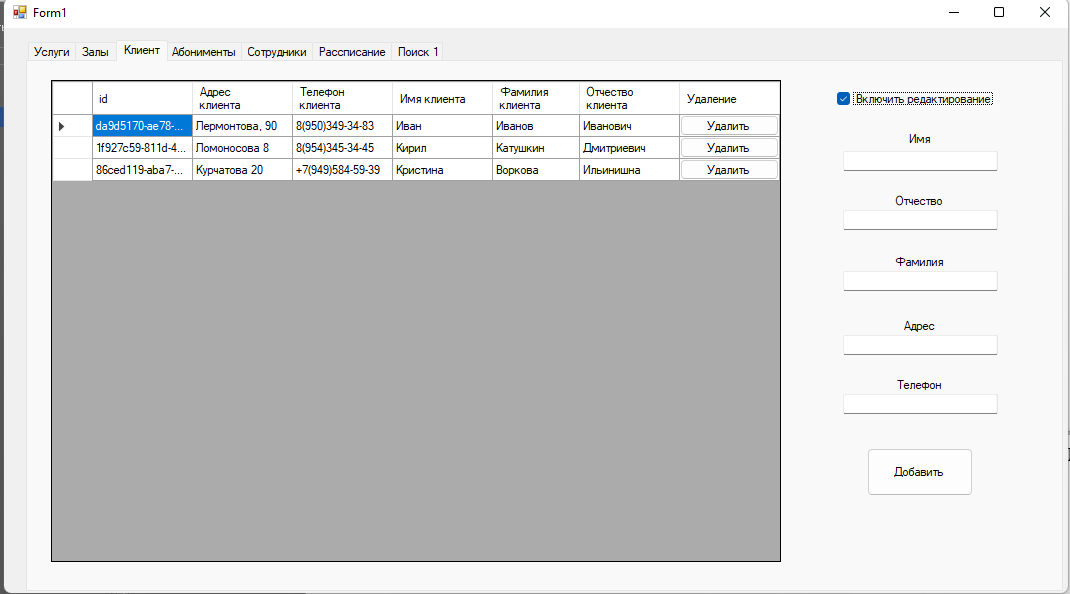


Рисунок 13 – Таблица Клиент

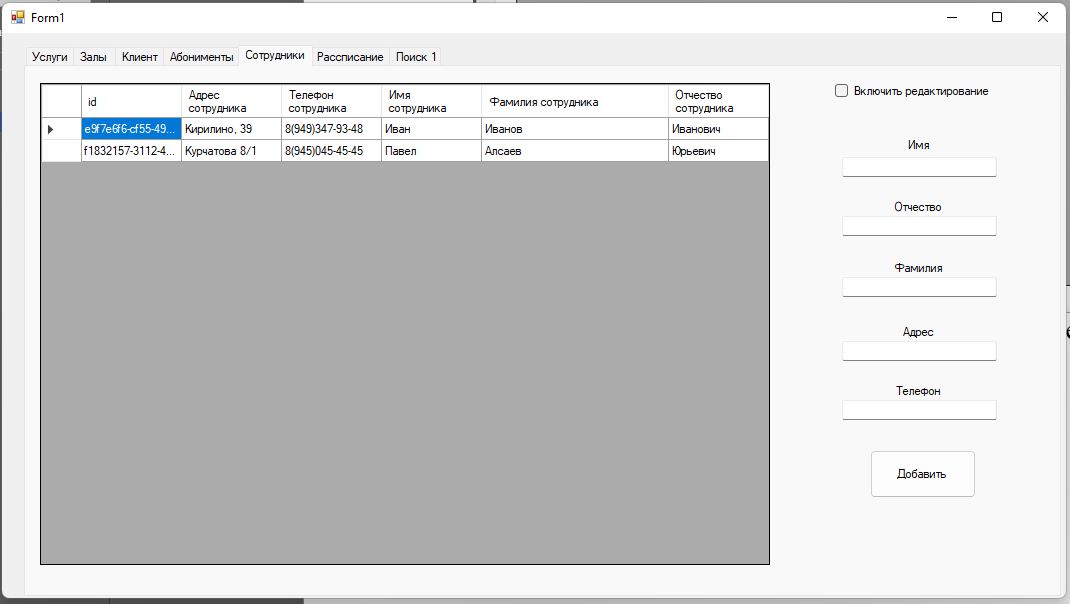


Рисунок 14 – Таблица Сотрудники

Интерфейс таблицы Абонемент несколько отличается: данная таблица связывает Клиентов и их Услуги. Их поля представляют из себя выпадающие списки с именами и услугами.

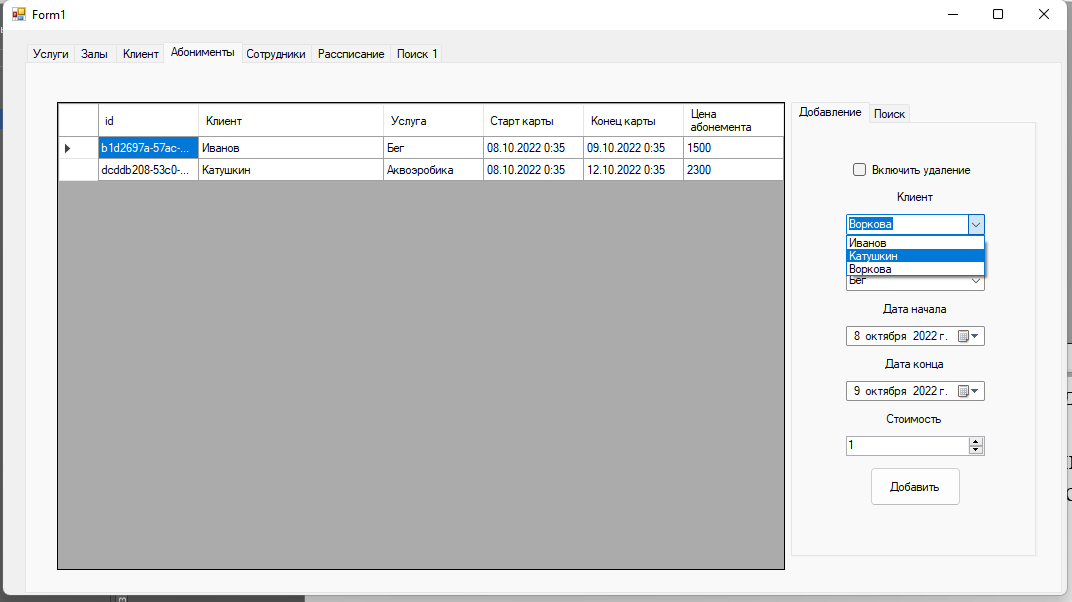


Рисунок 15 – Таблица Абонемент

Также с помощью чекбокса включается возможность удаления абонемента.

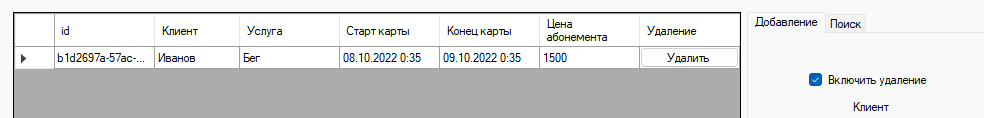


Рисунок 16 – Удаление абонемента

В данной таблице можно осуществлять поиск по таблице по остатку дней действия абонемента.

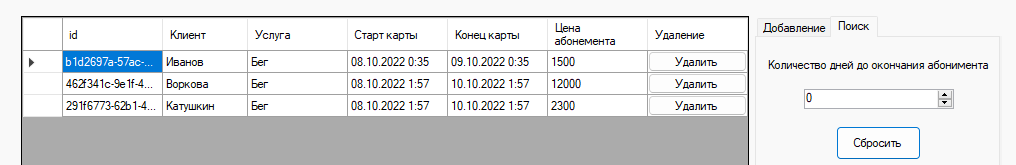


Рисунок 17 – Интерфейс поиска

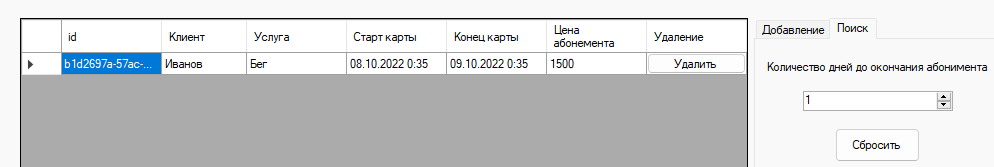


Рисунок 18 – Поиск по таблице

Таблица Расписание схожа по функционалу с таблицей Абонемента: в данной таблице можно связывать Сотрудников, Залы и Клиентов. Также задавать дату тренировки и ее продолжительность.

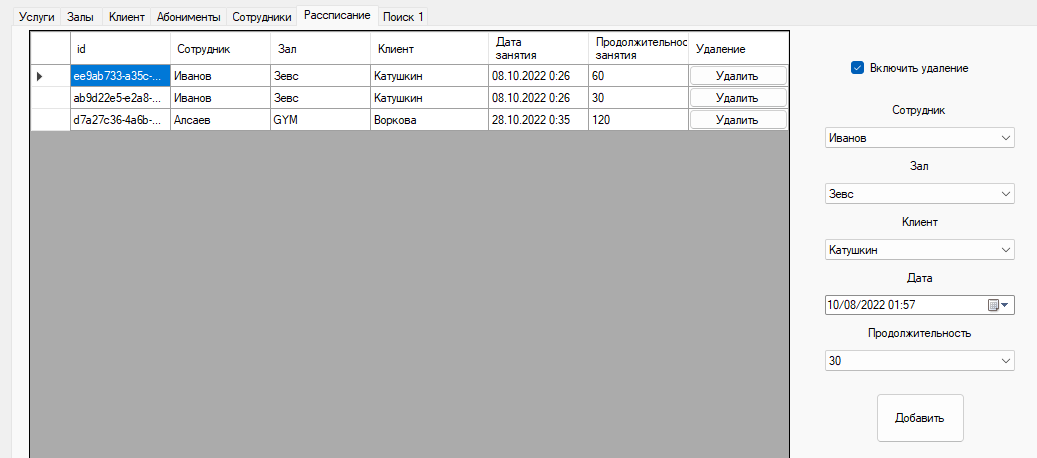


Рисунок 19 – Таблица Расписание

При включении чекбокса удаления активируется соответствующая возможность. Данный процесс сопровождается соответствующим предупреждением пользователя.

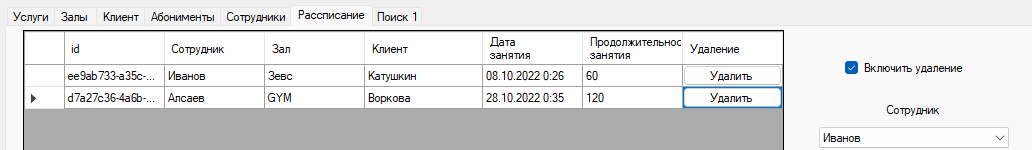


Рисунок 20 – Таблица после удаления записи

В таблице Поиск 1 отображаются задействованные тренера и количество занимающихся у них человек.

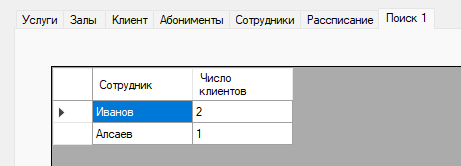


Рисунок 21 – Таблица Поиск 1

Листинг программы

Класс DataBase:

internal class DataBase

{

const String filename = @"gym.yap";

IObjectContainer db;

public void dbConnect()

{

db = Db4oFactory.OpenFile(filename);

}

public void dbClose()

{

db.Close();

}

#region Работа с услугой

public void AddService(Service service)

{

db.Store(service);

}

public List<Service> GetService()

{

return db.Query<Service>().ToList();

}

public void DeleteService(string id)

{

var service = db.Query<Service>(src => src.id == id)[0];

db.Delete(service);

}

public void UpdateService(Service service)

{

var found = db.Query<Service>(src => src.id == service.id)[0];

found.Price = service.Price;

found.Title = service.Title;

db.Store(found);

}

#endregion

#region Работа с залами

public void AddHall(Hall hall)

{

db.Store(hall);

}

public void DeleteHall(string id)

{

var found = db.Query<Hall>(hl => id == hl.id)[0];

db.Delete(found);

}

public List<Hall> GetHall()

{

var list = db.Query<Hall>().ToList();

return list;

}

public void UpdateHall(Hall hall)

{

var found = db.Query<Hall>(hl => hl.id == hall.id)[0];

found.title = hall.title;

db.Store(found);

}

#endregion

#region работа с клиентами

public void AddClient(Client client)

{

db.Store(client);

}

public List<Client> GetClient()

{

return db.Query<Client>().ToList();

}

public void DeleteClient(string id)

{

var found = db.Query<Client>(clt => clt.id == id)[0];

db.Delete(found);

}

public void UpdateClient(Client client)

{

var found = db.Query<Client>(clt => clt.id == client.id)[0];

found.phone = client.phone;

found.adress = client.adress;

found.surname = client.surname;

found.name = client.name;

found.patronymic = client.patronymic;

db.Store(found);

}

#endregion

#region Работа с абониментом

public void AddClubCard(ClubCard clubCard)

{

db.Store(clubCard);

}

public void DeleteClubCard(string id)

{

var found = db.Query<ClubCard>(cc => cc.id == id)[0];

db.Delete(found);

}

public List<ClubCard> GetClubCard()

{

return db.Query<ClubCard>().ToList();

}

#endregion

#region

public void AddStaff(Staff staff)

{

db.Store(staff);

}

public List<Staff> GetStaff() => db.Query<Staff>().ToList();

public void DeleteStaff(string id)

{

var found = db.Query<Staff>(stf => stf.id == id)[0];

db.Delete(found);

}

public void UpdateStaff(Staff staff)

{

var found = db.Query<Staff>(stf => stf.id == staff.id)[0];

found.name = staff.name;

found.phone = staff.phone;

found.surname = staff.surname;

found.adress = staff.adress;

found.patronymic = staff.patronymic;

db.Store(found);

}

#endregion

#region

public void AddTaskTable(TaskTable taskTable)

{

db.Store(taskTable);

}

public void DeleteTaskTable(string id)

{

var found = db.Query<TaskTable>(tt => tt.id == id)[0];

db.Delete(found);

}

public List<TaskTable> GetTaskTable() => db.Query<TaskTable>().ToList();

#endregion

#region Поиски

public List<ClubCard> ClubCardSearch(int dayCount)

{

var ans = (from ClubCard cc in db.Cast<ClubCard>()

where (cc.endCard - DateTime.Now).TotalDays <= dayCount

select cc).ToList<ClubCard>();

return ans;

}

public List<Service> ServiceSearch(int price, string title)

{

IQuery query = db.Query();

query.Constrain(typeof(Service));

IQuery titleQuery = query.Descend("\_title");

query.Descend("\_price").Constrain(price).Greater().Equal().And(

titleQuery.Constrain(title).Like());

IObjectSet result = query.Execute();

var soda = new List<Service>();

foreach (var r in result)

{

soda.Add(r as Service);

}

return soda;

}

public List<Search> Search() {

var asn = from TaskTable tt in db.Cast<TaskTable>()

group tt by tt.staff into t

select new { Name = t.Key,

Count = t.Count() }; ;

List<Search> lists = new List<Search>();

foreach(var an in asn)

{

lists.Add(new Classes.Search(an.Count, an.Name));

}

return lists;

}

internal void ServiceSearch(decimal value)

{

throw new NotImplementedException();

}

#endregion

}

Класс Client:

internal class Client

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Адрес клиента")]

public string adress { get; set; }

[DisplayName("Телефон клиента")]

public string phone { get; set; }

[DisplayName("Имя клиента")]

public string name { get; set; }

[DisplayName("Фамилия клиента")]

public string surname { get; set; }

[DisplayName("Отчество клиента")]

public string patronymic { get; set; }

public Client()

{

id = Guid.NewGuid().ToString();

}

public Client(string adress, string phone, string name, string surname, string patronymic)

{

id = Guid.NewGuid().ToString();

this.adress = adress;

this.phone = phone;

this.name = name;

this.surname = surname;

this.patronymic = patronymic;

}

public override bool Equals(object obj)

{

return obj is Client client &&

id == client.id &&

adress == client.adress &&

phone == client.phone &&

name == client.name &&

surname == client.surname &&

patronymic == client.patronymic;

}

public override string ToString()

{

return surname;

}

}

Класс ClubCard:

internal class ClubCard

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Клиент")]

public Client client { get; set; }

[DisplayName("Услуга")]

public Service service { get; set; }

[DisplayName("Старт карты")]

public DateTime startCard { get; set; }

[DisplayName("Конец карты")]

public DateTime endCard { get; set; }

[DisplayName("Цена абонемента")]

public int price { get; set; }

public ClubCard()

{

id = Guid.NewGuid().ToString();

}

public ClubCard(Client client, Service service, DateTime startCard, DateTime endCard, int price)

{

id = Guid.NewGuid().ToString();

this.client = client;

this.service = service;

this.startCard = startCard;

this.endCard = endCard;

this.price = price;

}

}

Класс Hall:

internal class Hall

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Зал")]

public string title { get; set; }

public Hall()

{

id = Guid.NewGuid().ToString();

}

public Hall(string title)

{

id = Guid.NewGuid().ToString();

this.title = title;

}

public override bool Equals(object obj)

{

return obj is Hall hall &&

id == hall.id &&

title == hall.title;

}

public override string ToString()

{

return title;

}

}

Класс Service:

internal class Service

{

[DisplayName("id")]

public string id { get; set; }

private string \_title;

[DisplayName("Название услуги")]

public string Title

{

get { return \_title; }

set { \_title = value; }

}

private int \_price;

[DisplayName("Цена")]

public int Price

{

get { return \_price; }

set { \_price = value; }

}

public Service()

{

id = Guid.NewGuid().ToString();

}

public Service(string title, int price)

{

id = Guid.NewGuid().ToString();

this.\_title = title;

this.\_price = price;

}

public override bool Equals(object obj)

{

return obj is Service service &&

id == service.id &&

\_title == service.Title &&

\_price == service.\_price;

}

public override string ToString()

{

return Title;

}

}

Класс Staff:

internal class Staff

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Адрес сотрудника")]

public string adress { get; set; }

[DisplayName("Телефон сотрудника")]

public string phone { get; set; }

[DisplayName("Имя сотрудника")]

public string name { get; set; }

[DisplayName("Фамилия сотрудника")]

public string surname { get; set; }

[DisplayName("Отчество сотрудника")]

public string patronymic { get; set; }

public Staff()

{

id = Guid.NewGuid().ToString();

}

public Staff(string adress, string phone, string name, string surname, string patronymic)

{

id = Guid.NewGuid().ToString();

this.adress = adress;

this.phone = phone;

this.name = name;

this.surname = surname;

this.patronymic = patronymic;

}

public override bool Equals(object obj)

{

return obj is Staff staff &&

id == staff.id &&

adress == staff.adress &&

phone == staff.phone &&

name == staff.name &&

surname == staff.surname &&

patronymic == staff.patronymic;

}

public override string ToString()

{

return surname;

}

}

Класс TaskTable:

internal class TaskTable

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Сотрудник")]

public Staff staff { get; set; }

[DisplayName("Зал")]

public Hall hall { get; set; }

[DisplayName("Клиент")]

public Client client { get; set; }

[DisplayName("Дата занятия")]

public DateTime dateLesson { get; set; }

[DisplayName("Продолжительность занятия")]

public int duration { get; set; }

public TaskTable()

{

id = Guid.NewGuid().ToString();

}

public TaskTable(Staff staff, Hall hall, Client client, DateTime dateLesson, int duration)

{

id = Guid.NewGuid().ToString();

this.staff = staff;

this.hall = hall;

this.client = client;

this.dateLesson = dateLesson;

this.duration = duration;

}

}

Класс Search

internal class Search

{

[DisplayName ("Сотрудник")]

public Staff Staff { get; set; }

[DisplayName("Число клиентов")]

public int Count { get; set; }

public Search(int count, Staff staff)

{

Count = count;

Staff = staff;

}

}

Класс Form1:

public partial class Form1 : Form

{

#region Настройка формы

DataBase dataBase;

static Random random = new Random();

string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789" + "ABCDEFGHIJKLMNOPQRSTUVWXYZ".ToLower();

public Form1()

{

InitializeComponent();

dataBase = new DataBase();

dataBase.dbConnect();

bindingSourceSetting();

cmbTaskTableDuraction.SelectedIndex = 0;

dtpClubCardStart.MinDate = DateTime.Now;

dtpClubCardEnd.MinDate = DateTime.Now.AddDays(1);

dtpTaskTableDate.Format = DateTimePickerFormat.Custom;

dtpTaskTableDate.CustomFormat = "MM/dd/yyyy HH:mm";

dtpTaskTableDate.MinDate = DateTime.Now;

dgvSearch.DataSource = dataBase.Search();

}

void bindingSourceSetting()

{

foreach (var s in dataBase.GetService())

{

serviceBindingSource.Add(s);

}

foreach(var h in dataBase.GetHall())

{

hallBindingSource.Add(h);

}

foreach(var c in dataBase.GetClient())

{

clientBindingSource.Add(c);

}

foreach(var cc in dataBase.GetClubCard())

{

clubCardBindingSource.Add(cc);

}

foreach (var staff in dataBase.GetStaff())

{

staffBindingSource.Add(staff);

}

foreach(var tt in dataBase.GetTaskTable())

{

taskTableBindingSource.Add(tt);

}

}

#endregion

#region Утилиты

void messageBoxSuccessAdd()

{

MessageBox.Show("Запись успешно добавлена",

"Успех", MessageBoxButtons.OK,

MessageBoxIcon.Information);

}

void messageBoxError(string message)

{

MessageBox.Show(message,

"Ошибка", MessageBoxButtons.OK,

MessageBoxIcon.Error);

}

DialogResult messageBoxClickResult(string message)

{

var dialogResult = MessageBox.Show(message,

"Предупреждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

return dialogResult;

}

string generateString(int from, int to)

{

return new string(Enumerable.Repeat(chars, random.Next(from, to))

.Select(s => s[random.Next(s.Length)]).ToArray());

}

//Отменение изменений

private void tableValidating(object sender, DataGridViewCellValidatingEventArgs e)

{

if (string.IsNullOrEmpty(e.FormattedValue.ToString()))

{

((DataGridView)sender).CancelEdit();

((DataGridView)sender).EndEdit();

}

}

#endregion

#region Работа с услугой

private void cbIsServiceEdit\_CheckedChanged(object sender, EventArgs e)

{

if(cbIsServiceEdit.Checked == true)

{

dgvService.Columns[1].ReadOnly = false;

dgvService.Columns[2].ReadOnly = false;

dgvService.Columns[3].Visible = true;

}

else

{

dgvService.Columns[1].ReadOnly = true;

dgvService.Columns[2].ReadOnly = true;

dgvService.Columns[3].Visible = false;

}

}

private void dgvService\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvService.Columns[e.ColumnIndex].Index == dgvService.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvService.Rows[e.RowIndex].Cells[0].Value.ToString();

serviceBindingSource.RemoveAt(e.RowIndex);

dataBase.DeleteService(id);

}

}

}

Service selectedService;

private void dgvService\_CellBeginEdit(object sender, DataGridViewCellCancelEventArgs e)

{

selectedService = new Service();

selectedService.id = dgvService.Rows[e.RowIndex].Cells[0].Value.ToString();

selectedService.Title = dgvService.Rows[e.RowIndex].Cells[1].Value.ToString();

selectedService.Price = (int)dgvService.Rows[e.RowIndex].Cells[2].Value;

}

private void dgvService\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

var service = new Service();

if((int)dgvService.Rows[e.RowIndex].Cells[2].Value < 100)

{

messageBoxError("Некорректная цена");

return;

}

service.id = dgvService.Rows[e.RowIndex].Cells[0].Value.ToString();

service.Title = dgvService.Rows[e.RowIndex].Cells[1].Value.ToString();

service.Price = (int)dgvService.Rows[e.RowIndex].Cells[2].Value;

if (service.Equals(selectedService))

{

return;

}

var dialogResult = messageBoxClickResult("Изменить эту запись?");

if (dialogResult == DialogResult.No)

{

dgvService.Rows[e.RowIndex].Cells[1].Value = selectedService.Title;

dgvService.Rows[e.RowIndex].Cells[2].Value = selectedService.Price;

return;

}

if (dialogResult == DialogResult.Yes)

{

dataBase.UpdateService(dgvService.CurrentRow.DataBoundItem as Service);

}

}

private void dgvService\_DataError(object sender, DataGridViewDataErrorEventArgs e)

{

messageBoxError("Введите числовое значение");

dgvService.Rows[e.RowIndex].Cells[2].Value = selectedService.Price;

}

private void btnServiceAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtServiceTitle.Text)){

messageBoxError("Вы не ввели название");

return;

}

var service = new Service();

service.Price = (int)nudServicePrice.Value;

service.Title = txtServiceTitle.Text;

dataBase.AddService(service);

txtServiceTitle.Clear();

serviceBindingSource.Add(service);

messageBoxSuccessAdd();

}

#endregion

#region работа с залами

private void dgvHall\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvHall.Columns[e.ColumnIndex].Index == dgvHall.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvHall.Rows[e.RowIndex].Cells[0].Value.ToString();

hallBindingSource.RemoveAt(e.RowIndex);

dataBase.DeleteHall(id);

}

}

}

private void btnHallAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtHallTitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

var hall = new Hall();

hall.title = txtHallTitle.Text;

dataBase.AddHall(hall);

hallBindingSource.Add(hall);

txtHallTitle.Clear();

messageBoxSuccessAdd();

}

private void cbHallIsEdit\_CheckedChanged(object sender, EventArgs e)

{

if (cbHallIsEdit.Checked)

{

dgvHall.Columns[1].ReadOnly = false;

dgvHall.Columns[2].Visible = true;

}

else

{

dgvHall.Columns[1].ReadOnly = true;

dgvHall.Columns[2].Visible = false;

}

}

private void dgvHall\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

var hall = new Hall();

hall.id = dgvHall.Rows[e.RowIndex].Cells[0].Value.ToString();

hall.title = dgvHall.Rows[e.RowIndex].Cells[1].Value.ToString();

if (hall.Equals(selectedHall))

{

return;

}

var dialogResult = messageBoxClickResult("Изменить эту запись?");

if (dialogResult == DialogResult.No)

{

dgvHall.Rows[e.RowIndex].Cells[1].Value = selectedHall.title;

return;

}

if (dialogResult == DialogResult.Yes)

{

dataBase.UpdateHall(dgvHall.CurrentRow.DataBoundItem as Hall);

}

}

Hall selectedHall;

private void dgvHall\_CellBeginEdit(object sender, DataGridViewCellCancelEventArgs e)

{

selectedHall = new Hall();

selectedHall.id = dgvHall.Rows[e.RowIndex].Cells[0].Value.ToString();

selectedHall.title = dgvHall.Rows[e.RowIndex].Cells[1].Value.ToString();

}

#endregion

#region работа с клиентом

private void cbClientIsEdit\_CheckedChanged(object sender, EventArgs e)

{

if(cbClientIsEdit.Checked == true)

{

dgvClient.Columns[1].ReadOnly = false;

dgvClient.Columns[2].ReadOnly = false;

dgvClient.Columns[3].ReadOnly = false;

dgvClient.Columns[4].ReadOnly = false;

dgvClient.Columns[5].ReadOnly = false;

dgvClient.Columns[6].Visible = true;

}

else

{

dgvClient.Columns[1].ReadOnly = true;

dgvClient.Columns[2].ReadOnly = true;

dgvClient.Columns[3].ReadOnly = true;

dgvClient.Columns[4].ReadOnly = true;

dgvClient.Columns[5].ReadOnly = true;

dgvClient.Columns[6].Visible = false;

}

}

private void dgvClient\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvClient.Columns[e.ColumnIndex].Index == dgvClient.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvClient.Rows[e.RowIndex].Cells[0].Value.ToString();

clientBindingSource.RemoveAt(e.RowIndex);

dataBase.DeleteClient(id);

}

}

}

private void btnClientAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtClientName.Text))

{

messageBoxError("Вы не ввели имя");

return;

}

if (string.IsNullOrEmpty(txtClientSecondName.Text))

{

messageBoxError("Вы не ввели отчество");

return;

}

if (string.IsNullOrEmpty(txtClientLastName.Text))

{

messageBoxError("Вы не ввели фамилию");

return;

}

if (string.IsNullOrEmpty(txtClientAddress.Text))

{

messageBoxError("Вы не ввели адрес");

return;

}

if (string.IsNullOrEmpty(txtClientPhone.Text))

{

messageBoxError("Вы не ввели телефон");

return;

}

Regex phoneRegex = new Regex(@"^[+]{0,1}[7-8]{1}[(]9{1}\d{2}[)]\d{3}[-]\d{2}[-]\d{2}$");

MatchCollection matches = phoneRegex.Matches(txtClientPhone.Text);

if (matches.Count == 0)

{

messageBoxError("Не верный формат телефона");

return;

}

var client = new Client();

client.adress = txtClientAddress.Text;

client.surname = txtClientLastName.Text;

client.phone = txtClientPhone.Text;

client.patronymic = txtClientSecondName.Text;

client.name = txtClientName.Text;

dataBase.AddClient(client);

clientBindingSource.Add(client);

txtClientAddress.Clear();

txtClientLastName.Clear();

txtClientPhone.Clear();

txtClientSecondName.Clear();

txtClientName.Clear();

messageBoxSuccessAdd();

}

private void txtClientPhone\_TextChanged(object sender, EventArgs e)

{

//Добавление первой скобки

if (txtClientPhone.Text.Length == 1 && (txtClientPhone.Text[0] != '+'))

{

txtClientPhone.Text += "(9";

txtClientPhone.SelectionStart = 3;

}

else if (txtClientPhone.Text.Length == 2 && txtClientPhone.Text[1] == '7')

{

txtClientPhone.Text += "(9";

txtClientPhone.SelectionStart = 4;

}

//Добавление второй скобки

if (txtClientPhone.Text.Length == 5 && (txtClientPhone.Text[0] != '+'))

{

txtClientPhone.Text += ")";

txtClientPhone.SelectionStart = 6;

}

else if (txtClientPhone.Text.Length == 6 && txtClientPhone.Text[0] == '+')

{

txtClientPhone.Text += ")";

txtClientPhone.SelectionStart = 7;

}

//Добавление -

if (txtClientPhone.Text.Length == 9 && (txtClientPhone.Text[0] != '+'))

{

txtClientPhone.Text += "-";

txtClientPhone.SelectionStart = 10;

}

else if (txtClientPhone.Text.Length == 10 && txtClientPhone.Text[0] == '+')

{

txtClientPhone.Text += "-";

txtClientPhone.SelectionStart = 11;

}

//Добавление -

if (txtClientPhone.Text.Length == 12 && (txtClientPhone.Text[0] != '+'))

{

txtClientPhone.Text += "-";

txtClientPhone.SelectionStart = 13;

}

else if (txtClientPhone.Text.Length == 13 && txtClientPhone.Text[0] == '+')

{

txtClientPhone.Text += "-";

txtClientPhone.SelectionStart = 14;

}

}

Client selectedClient;

private void dgvClient\_CellBeginEdit(object sender, DataGridViewCellCancelEventArgs e)

{

selectedClient = new Client();

selectedClient.id = dgvClient.Rows[e.RowIndex].Cells[0].Value.ToString();

selectedClient.adress = dgvClient.Rows[e.RowIndex].Cells[1].Value.ToString();

selectedClient.phone = dgvClient.Rows[e.RowIndex].Cells[2].Value.ToString();

selectedClient.name = dgvClient.Rows[e.RowIndex].Cells[3].Value.ToString();

selectedClient.patronymic = dgvClient.Rows[e.RowIndex].Cells[4].Value.ToString();

selectedClient.surname = dgvClient.Rows[e.RowIndex].Cells[5].Value.ToString();

}

private void dgvClient\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

var client = new Client();

client.id = dgvClient.Rows[e.RowIndex].Cells[0].Value.ToString();

client.adress = dgvClient.Rows[e.RowIndex].Cells[1].Value.ToString();

client.phone = dgvClient.Rows[e.RowIndex].Cells[2].Value.ToString();

client.name = dgvClient.Rows[e.RowIndex].Cells[3].Value.ToString();

client.patronymic = dgvClient.Rows[e.RowIndex].Cells[4].Value.ToString();

client.surname = dgvClient.Rows[e.RowIndex].Cells[5].Value.ToString();

if (selectedClient.Equals(client))

{

return;

}

if (selectedClient.phone != client.phone)

{

Regex phoneRegex = new Regex(@"^[+]{0,1}[7-8]{1}[(]9{1}\d{2}[)]\d{3}[-]\d{2}[-]\d{2}$");

MatchCollection matches = phoneRegex.Matches(client.phone);

if (matches.Count == 0)

{

messageBoxError("Не верный формат телефона");

dgvClient[2, e.RowIndex].Value = selectedClient.phone;

return;

}

}

var dialogResult = messageBoxClickResult("Изменить эту запись?");

if (dialogResult == DialogResult.No)

{

dgvClient[1, e.RowIndex].Value = selectedClient.adress;

dgvClient[2, e.RowIndex].Value = selectedClient.phone;

dgvClient[3, e.RowIndex].Value = selectedClient.name;

dgvClient[4, e.RowIndex].Value = selectedClient.patronymic;

dgvClient[5, e.RowIndex].Value = selectedClient.surname;

return;

}

if (dialogResult == DialogResult.Yes)

{

dataBase.UpdateClient(dgvClient.CurrentRow.DataBoundItem as Client);

}

}

#endregion

#region Работа с абониментом

private void dgvClubCard\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvClubCard.Columns[e.ColumnIndex].Index == dgvClubCard.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvClubCard.Rows[e.RowIndex].Cells[0].Value.ToString();

clubCardBindingSource.RemoveAt(e.RowIndex);

dataBase.DeleteClubCard(id);

}

}

}

private void cbIsClubCardEdit\_CheckedChanged(object sender, EventArgs e)

{

if (cbIsClubCardEdit.Checked)

{

dgvClubCard.Columns[6].Visible = true;

}

else

{

dgvClubCard.Columns[6].Visible = false;

}

}

private void btnClubCardAdd\_Click(object sender, EventArgs e)

{

var dateStart = dtpClubCardStart.Value;

var dateEnd = dtpClubCardEnd.Value;

if (dateStart.Day == dateEnd.Day

&& dateStart.Month == dateEnd.Month

&& dateStart.Year == dateEnd.Year)

{

messageBoxError("Не верные даты");

return;

}

var clubCard = new ClubCard();

clubCard.price = (int)nudClubCardPrice.Value;

clubCard.client = cmbClubCardClient.SelectedItem as Client;

clubCard.startCard = dateStart;

clubCard.endCard = dateEnd;

clubCard.service = cmbClubCardService.SelectedItem as Service;

dataBase.AddClubCard(clubCard);

clubCardBindingSource.Add(clubCard);

messageBoxSuccessAdd();

}

#endregion

#region Работа с сотрудником

private void dgvStaff\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvStaff.Columns[e.ColumnIndex].Index == dgvStaff.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvStaff.Rows[e.RowIndex].Cells[0].Value.ToString();

staffBindingSource.RemoveAt(e.RowIndex);

dataBase.DeleteStaff(id);

}

}

}

private void cbIsStaffEdit\_CheckedChanged(object sender, EventArgs e)

{

if (cbIsStaffEdit.Checked)

{

dgvStaff.Columns[1].ReadOnly = false;

dgvStaff.Columns[2].ReadOnly = false;

dgvStaff.Columns[3].ReadOnly = false;

dgvStaff.Columns[4].ReadOnly = false;

dgvStaff.Columns[5].ReadOnly = false;

dgvStaff.Columns[6].Visible = true;

}

else

{

dgvStaff.Columns[1].ReadOnly = true;

dgvStaff.Columns[2].ReadOnly = true;

dgvStaff.Columns[3].ReadOnly = true;

dgvStaff.Columns[4].ReadOnly = true;

dgvStaff.Columns[5].ReadOnly = true;

dgvStaff.Columns[6].Visible = false;

}

}

private void btnStaffAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtStaffName.Text))

{

messageBoxError("Вы не ввели имя");

return;

}

if (string.IsNullOrEmpty(txtStaffSecondName.Text))

{

messageBoxError("Вы не ввели отчество");

return;

}

if (string.IsNullOrEmpty(txtStaffLastName.Text))

{

messageBoxError("Вы не ввели фамилию");

return;

}

if (string.IsNullOrEmpty(txtStaffAddress.Text))

{

messageBoxError("Вы не ввели адрес");

return;

}

if (string.IsNullOrEmpty(txtStaffPhone.Text))

{

messageBoxError("Вы не ввели телефон");

return;

}

Regex phoneRegex = new Regex(@"^[+]{0,1}[7-8]{1}[(]9{1}\d{2}[)]\d{3}[-]\d{2}[-]\d{2}$");

MatchCollection matches = phoneRegex.Matches(txtStaffPhone.Text);

if (matches.Count == 0)

{

messageBoxError("Не верный формат телефона");

return;

}

var staff = new Staff();

staff.adress = txtStaffAddress.Text;

staff.surname = txtStaffLastName.Text;

staff.phone = txtStaffPhone.Text;

staff.patronymic = txtStaffSecondName.Text;

staff.name = txtStaffName.Text;

dataBase.AddStaff(staff);

staffBindingSource.Add(staff);

txtStaffAddress.Clear();

txtStaffPhone.Clear();

txtStaffSecondName.Clear();

txtStaffName.Clear();

txtStaffLastName.Clear();

messageBoxSuccessAdd();

}

private void txtStaffPhone\_TextChanged(object sender, EventArgs e)

{

//Добавление первой скобки

if (txtStaffPhone.Text.Length == 1 && (txtStaffPhone.Text[0] != '+'))

{

txtStaffPhone.Text += "(9";

txtStaffPhone.SelectionStart = 3;

}

else if (txtStaffPhone.Text.Length == 2 && txtStaffPhone.Text[1] == '7')

{

txtStaffPhone.Text += "(9";

txtStaffPhone.SelectionStart = 4;

}

//Добавление второй скобки

if (txtStaffPhone.Text.Length == 5 && (txtStaffPhone.Text[0] != '+'))

{

txtStaffPhone.Text += ")";

txtStaffPhone.SelectionStart = 6;

}

else if (txtStaffPhone.Text.Length == 6 && txtStaffPhone.Text[0] == '+')

{

txtStaffPhone.Text += ")";

txtStaffPhone.SelectionStart = 7;

}

//Добавление -

if (txtStaffPhone.Text.Length == 9 && (txtStaffPhone.Text[0] != '+'))

{

txtStaffPhone.Text += "-";

txtStaffPhone.SelectionStart = 10;

}

else if (txtStaffPhone.Text.Length == 10 && txtStaffPhone.Text[0] == '+')

{

txtStaffPhone.Text += "-";

txtStaffPhone.SelectionStart = 11;

}

//Добавление -

if (txtStaffPhone.Text.Length == 12 && (txtStaffPhone.Text[0] != '+'))

{

txtStaffPhone.Text += "-";

txtStaffPhone.SelectionStart = 13;

}

else if (txtStaffPhone.Text.Length == 13 && txtStaffPhone.Text[0] == '+')

{

txtStaffPhone.Text += "-";

txtStaffPhone.SelectionStart = 14;

}

}

Staff selectedStaff;

private void dgvStaff\_CellBeginEdit(object sender, DataGridViewCellCancelEventArgs e)

{

selectedStaff = new Staff();

selectedStaff.id = dgvStaff.Rows[e.RowIndex].Cells[0].Value.ToString();

selectedStaff.adress = dgvStaff.Rows[e.RowIndex].Cells[1].Value.ToString();

selectedStaff.phone = dgvStaff.Rows[e.RowIndex].Cells[2].Value.ToString();

selectedStaff.name = dgvStaff.Rows[e.RowIndex].Cells[3].Value.ToString();

selectedStaff.patronymic = dgvStaff.Rows[e.RowIndex].Cells[4].Value.ToString();

selectedStaff.surname = dgvStaff.Rows[e.RowIndex].Cells[5].Value.ToString();

}

private void dgvStaff\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

var staff = new Staff();

staff.id = dgvStaff.Rows[e.RowIndex].Cells[0].Value.ToString();

staff.adress = dgvStaff.Rows[e.RowIndex].Cells[1].Value.ToString();

staff.phone = dgvStaff.Rows[e.RowIndex].Cells[2].Value.ToString();

staff.name = dgvStaff.Rows[e.RowIndex].Cells[3].Value.ToString();

staff.patronymic = dgvStaff.Rows[e.RowIndex].Cells[4].Value.ToString();

staff.surname = dgvStaff.Rows[e.RowIndex].Cells[5].Value.ToString();

if (selectedStaff.Equals(staff))

{

return;

}

if (selectedStaff.phone != staff.phone)

{

Regex phoneRegex = new Regex(@"^[+]{0,1}[7-8]{1}[(]9{1}\d{2}[)]\d{3}[-]\d{2}[-]\d{2}$");

MatchCollection matches = phoneRegex.Matches(staff.phone);

if (matches.Count == 0)

{

messageBoxError("Не верный формат телефона");

dgvStaff[2, e.RowIndex].Value = selectedStaff.phone;

return;

}

}

var dialogResult = messageBoxClickResult("Изменить эту запись?");

if (dialogResult == DialogResult.No)

{

dgvStaff[1, e.RowIndex].Value = selectedStaff.adress;

dgvStaff[2, e.RowIndex].Value = selectedStaff.phone;

dgvStaff[3, e.RowIndex].Value = selectedStaff.name;

dgvStaff[4, e.RowIndex].Value = selectedStaff.patronymic;

dgvStaff[5, e.RowIndex].Value = selectedStaff.surname;

return;

}

if (dialogResult == DialogResult.Yes)

{

dataBase.UpdateStaff(dgvStaff.CurrentRow.DataBoundItem as Staff);

}

}

#endregion

#region Работа с расписанием

private void btnTaskTabkeAdd\_Click(object sender, EventArgs e)

{

var tt = new TaskTable();

tt.staff = cmbTaskTableStaff.SelectedItem as Staff;

tt.client = cmbTaskTableClient.SelectedItem as Client;

tt.dateLesson = dtpTaskTableDate.Value;

tt.duration = int.Parse(cmbTaskTableDuraction.Text);

tt.hall = cmbTaskTableHall.SelectedItem as Hall;

dataBase.AddTaskTable(tt);

taskTableBindingSource.Add(tt);

messageBoxSuccessAdd();

dgvSearch.DataSource = dataBase.Search();

}

private void cbIsTaskTableEdit\_CheckedChanged(object sender, EventArgs e)

{

if (cbIsTaskTableEdit.Checked)

{

dgvTaskTable.Columns[6].Visible = true;

}

else

{

dgvTaskTable.Columns[6].Visible = false;

}

}

private void dgvTaskTable\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvTaskTable.Columns[e.ColumnIndex].Index == dgvTaskTable.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvTaskTable.Rows[e.RowIndex].Cells[0].Value.ToString();

taskTableBindingSource.RemoveAt(e.RowIndex);

dataBase.DeleteTaskTable(id);

}

}

}

#endregion

private void nudClubCardSearch\_ValueChanged(object sender, EventArgs e)

{

var ans = dataBase.ClubCardSearch((int)nudClubCardSearch.Value);

clubCardBindingSource.Clear();

foreach(var cc in ans)

{

clubCardBindingSource.Add(cc);

}

Console.WriteLine();

}

private void btnClubCardReset\_Click(object sender, EventArgs e)

{

nudClubCardSearch.Value = 0;

clubCardBindingSource.Clear();

foreach (var cc in dataBase.GetClubCard())

{

clubCardBindingSource.Add(cc);

}

}

private void nudServiceSerach\_ValueChanged(object sender, EventArgs e)

{

var ans = dataBase.ServiceSearch((int)nudServiceSerach.Value, txtServiceSearch.Text);

serviceBindingSource.Clear();

foreach(var a in ans)

{

serviceBindingSource.Add(a);

}

}

private void txtServiceSearch\_TextChanged(object sender, EventArgs e)

{

var ans = dataBase.ServiceSearch((int)nudServiceSerach.Value, txtServiceSearch.Text);

serviceBindingSource.Clear();

foreach (var a in ans)

{

serviceBindingSource.Add(a);

}

}

private void ResetServiceBtn\_Click(object sender, EventArgs e)

{

nudServiceSerach.Value = 100;

txtServiceSearch.Text = "";

serviceBindingSource.Clear();

foreach (var serv in dataBase.GetService())

{

serviceBindingSource.Add(serv);

}

}

}