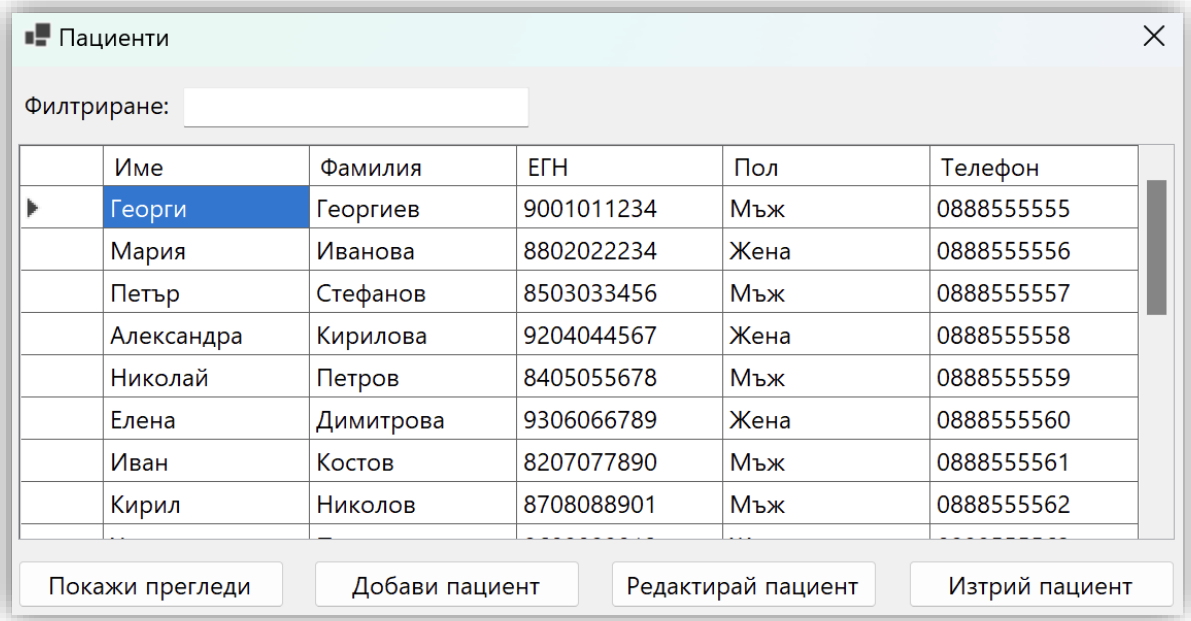
Имплементация на информационна система

Здравна информационна система

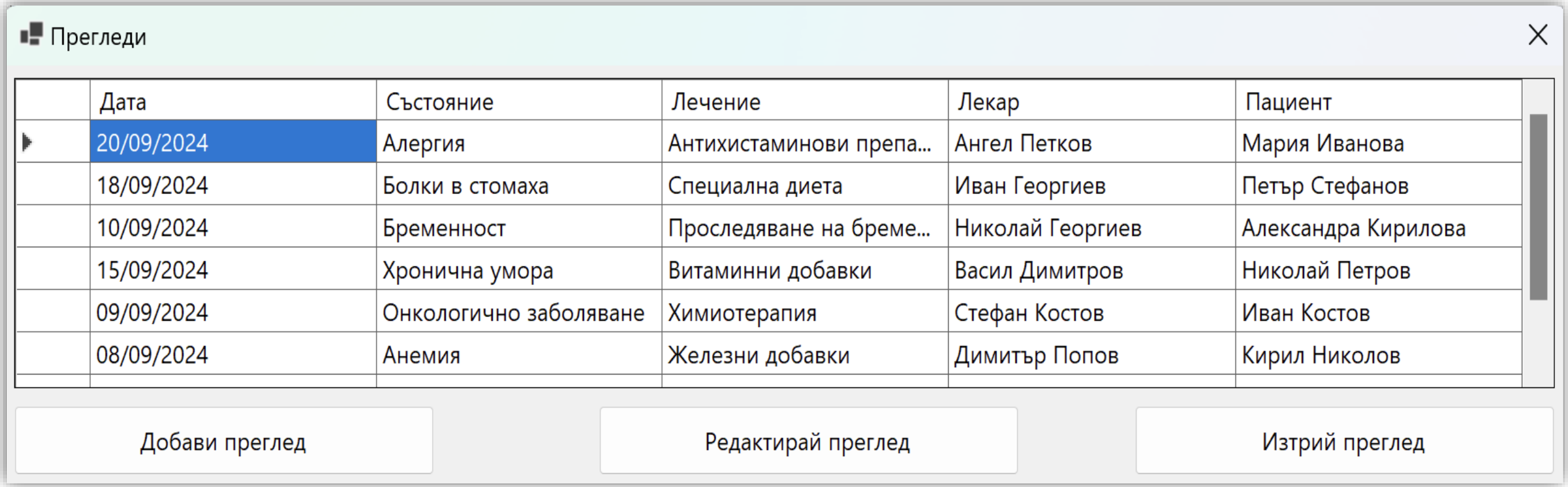
Трета част - Имплементация на отделни функционалности



- Курс "Информационни системи"
- Софтуерни и хардуерни науки

Съдържание

1. **Четене**, **добавяне**, **редактиране** и **изтриване** на **преглед**
   - Работа с **DTO (Data Transfer Object)**
   - Редактиране на **входна форма** и **главна форма** за **лекар**
   - Избиране на **лекар** спрямо **роля**
2. **Четене**, **добавяне**, **редактиране** и **изтриване** на **пациент**
   - Четене на **прегледи** на **избран пациент**
3. **Четене**, **добавяне**, **редактиране** и **изтриване** на **лекар**
4. **Четене**, **добавяне**, **редактиране** и **изтриване** на **админ**
5. Допълнителни **проверки**



Четене, добавяне, редактиране и изтриване на преглед

CRUD операции на преглед и работа с DTO

Съдържание на форми за Прегледи (1)

- Форма за **четене**
  - **Таблица** с **всички прегледи**
  - **Бутони** за **добавяне**, **редактиране** и **изтриване** на **преглед**
- Форма за **добавяне**
  - **Свойства** за **име** и **фамилия** на **пациент**, **име** и **фамилия** на **лекар**, **дата**, **състояние**, **лечение**

Съдържание на форми за Прегледи (2)

- Форма за **редактиране**
  - **Свойства** за **име** и **фамилия** на **пациент**, **име** и **фамилия** на **лекар**, **дата**, **състояние**, **лечение**
  - Задаваме **име** и **фамилия** на **пациент**, **име** и **фамилия** на **лекар**, **дата**, **състояние** и **лечение** през **конструктора**
- Форма за **изтриване**
  - Задаваме **име** и **фамилия** на **пациент**, **име** и **фамилия** на **лекар**, **дата**, **състояние** и **лечение** през **конструктора**

Съдържание на форми за Прегледи (3)

- Компоненти
  - **DataGridView - всички прегледи**
  - **Label - пациент, лекар, дата, състояние, лечение**
  - **TextBox - пациент, състояние, лечение**
  - **ComboBox - лекар**
  - **DateTimePicker - дата**
  - **Button - добавяне, редактиране, изтриване, отказ**

Работа с DTO (Data Transfer Object)

- **DTO (Data Transfer Object)** е обект за **прехвърляне** на **данни**
- Създаваме **ExaminationDto** съдържа нужните **полета** за **четене** на **преглед**

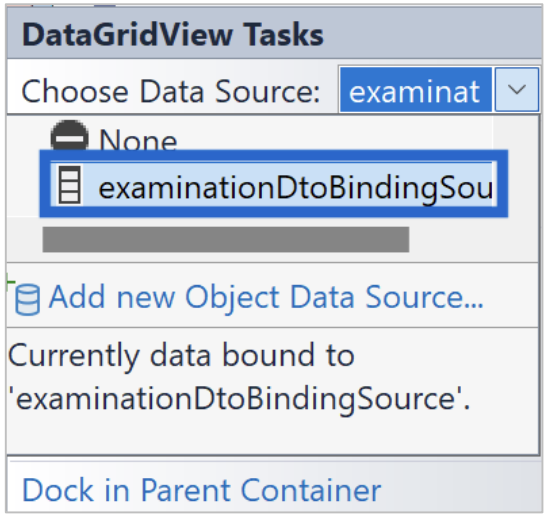
```
public class ExaminationDto
{
    public int ExaminationId { get; set; }
    public DateOnly ExaminationDate { get; set; }
    public string Condition { get; set; }
    public string Treatment { get; set; }
    public string DoctorName { get; set; }
    public string PatientName { get; set; }
    public int PatientId { get; set; }
}
```

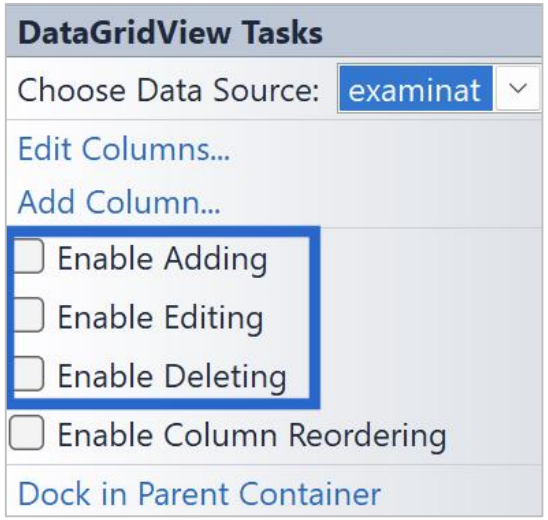
Име и фамилия на лекар

Име и фамилия на пациент

Добавяне на DataSource и стилизиране на DataGridView

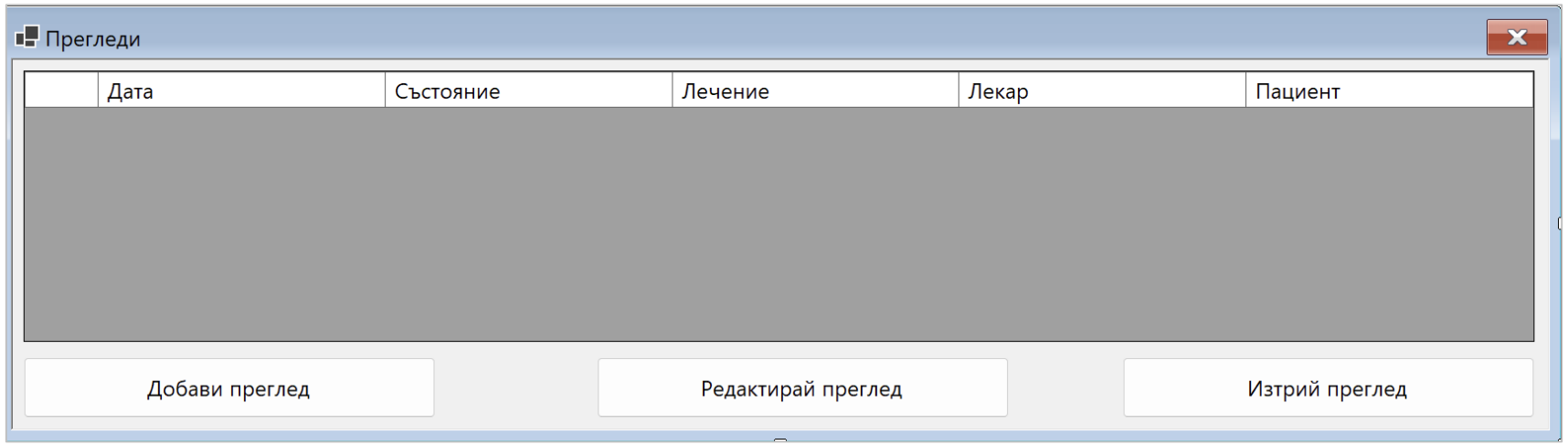
- Добавяме **DataSource** на **DataGridView** за **прегледи**
- Редактираме **имената** и **размерите** на **колоните**
- Задаваме следните **свойства** на **DataGridView**:
  - **EnableAdding** - **False**
  - **EnableEditing** - **False**
  - **EnableDeleting** - **False**





Стилизиране на форми

- Задаваме следните **свойства** на **формите**:
  - **StartPosition - CenterScreen**
  - **FormBorderStyle - Fixed3D**
  - **MaximizeBox - False**
  - **MinimizeBox - False**



Редактиране на входна форма

- Редактираме **входната форма** и **главната форма** за **лекар**, за да показваме **функционалност** за **логнат потребител лекар**
  - При **FormLogin**, **подаваме userId**

```
...
else
{
    var formMain = new FormMainDoctors(user.UserId);
    formMain.Show();
}
...
```

Подаваме userId

Редактиране на главна форма за лекари

- При **FormMainDoctor**, взимаме **userId** и го **подаваме** на **формите** за **пациенти** и **прегледи**

```
private int userId;

public FormMainDoctors(int userId)
{
    InitializeComponent();
    this.userId = userId;
}

private void buttonPatients_Click(object sender, EventArgs e)
{
    var formPatients = new FormPatients(userId);
    formPatients.ShowDialog();
}

private void buttonExaminations_Click(object sender, EventArgs e)
{
    var formExaminations = new FormExaminations(null, userId);
    formExaminations.ShowDialog();
}
```

Задаваме userId

Подаваме userId

Четене на Преглед (1)

- Добавяме **втори конструктор** на **формата**
  - Зареждат се **прегледите** на **всички пациенти** (**админ** е постъпил **прегледите** през бутонът **Прегледи** в **главната форма**)

```
public FormExaminations()
{
    InitializeComponent();
}

...
```

Четене на Преглед (2)

- Зареждат се **прегледите** при **избран пациент** (**админ** е **избрал пациент** и е кликнал върху **Покажи прегледи**)

```
private int? patientId;

...

public FormExaminations(int patientId)
{
    InitializeComponent();
    this.patientId = patientId;
}
...
```

Задаваме patientId

Четене на Преглед (3)

- Добавяме **трети конструктор** на **формата**
  - Зареждат се **прегледите** на **всички пациенти** или на **избран пациент** при **логнат лекар** (лекарят вижда **само** неговите **прегледи**)

```
private int? userId;
...
public FormExaminations(int? patientId, int? userId)
{
    InitializeComponent();
    this.patientId = patientId;
    this.userId = userId;
}
...
```

Задаваме patientId и userId

Четене на Преглед (4)

- Зареждаме **прегледите** при **отваряне** на **формата**

```
private void FormExaminations_Load(object sender, EventArgs e)
{
    ReloadExaminations();
}

private List<ExaminationDto> LoadExaminationsFormDb() {...}

private void ReloadExaminations()
{
    var examinations = LoadExaminationsFormDb();
    this.examinationDtoBindingSource.DataSource = examinations;
}
```

Използване на Examination DTO (1)

SoftUni Foundation

```
private List<ExaminationDto> LoadExaminationsFormDb()
{
    using (var db = new HospitalDbContext())
    {
        IQueryable<Examination> examinations = db.Examinations;

        // Лекарят е влязъл от Покажи прегледи на избран пациент
        if (userId != null && patientId != null)
        {
            var doctorId = db.Doctors.FirstOrDefault(d => d.UserId == userId).DoctorId;
            examinations = examinations.Where(e => e.PatientId == patientId
                                                && e.DoctorId == doctorId);
        }

        // Лекарят е влязъл от Прегледи на главната форма
        if (userId != null)
        {
            var doctorId = db.Doctors.FirstOrDefault(d => d.UserId == userId).DoctorId;
            examinations = examinations.Where(e => e.DoctorId == doctorId);
        }

...
```

Намираме прегледите на избран пациент и логнат лекар

Намираме прегледите на логнат лекар

Използване на Examination DTO (2)

```
...
        // Администраторът е влязъл от Покажи прегледи на избран пациент
        if (patientId != null)
        {
            examinations = examinations
                           .Where(e =>
                                  e.PatientId == patientId);
        }
...
```

Намираме прегледите на избран пациент

Използване на Examination DTO (3)

```
...
        // Администраторът е влязъл от Прегледи на главната форма
        return examinations
                .Include(e => e.Doctor)
                .Include(e => e.Patient)
                .Select(e => new ExaminationDto
                {
                    ExaminationId = e.ExaminationId,
                    ExaminationDate = e.ExaminationDate,
                    Condition = e.Condition,
                    Treatment = e.Treatment,
                    DoctorName = e.Doctor.FirstName + " " + e.Doctor.LastName,
                    PatientName = e.Patient.FirstName + " " + e.Patient.LastName,
                    PatientId = e.PatientId
                }).ToList();
    }
}
```

Връщаме всички прегледи

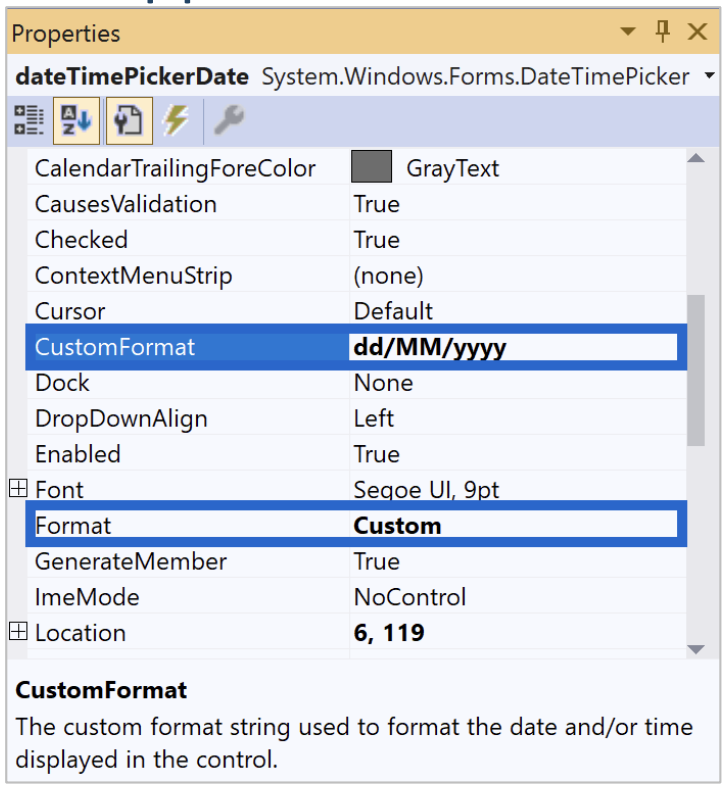
Методи-обработчици и DialogResult за бутони

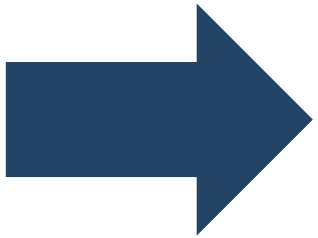
SoftUni Foundation

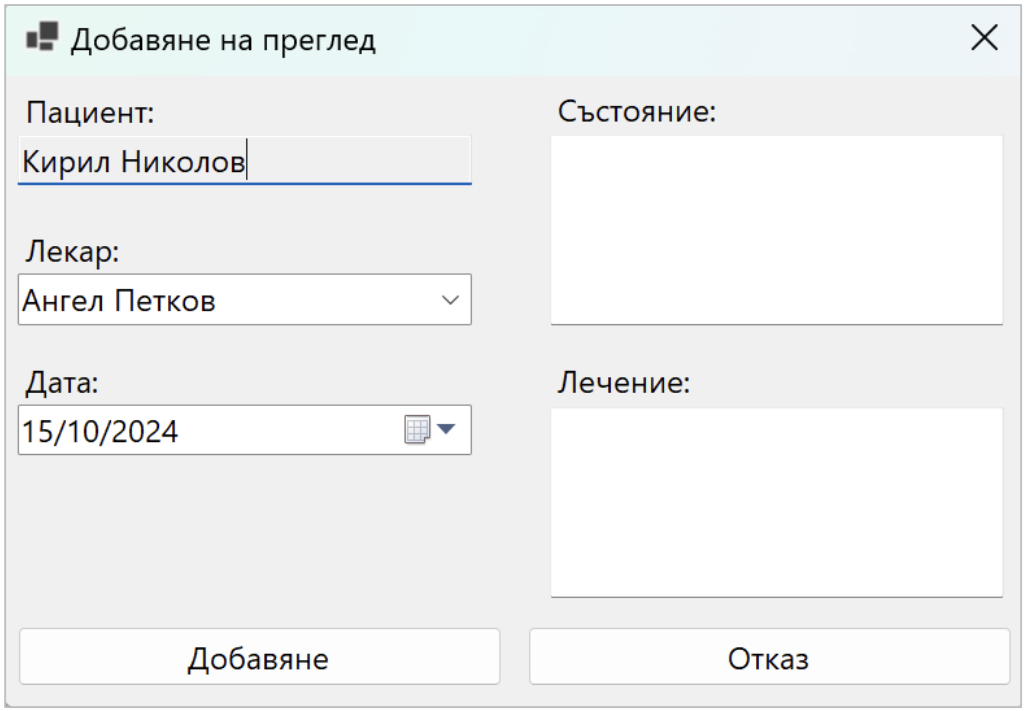
- Добавяме **методи-обработчици** към **бутоните**
  - **Добавяне на преглед**
  - **Редактиране на преглед**
  - **Изтриване на преглед**
- Задаваме подходящ **DialogResult** на **бутоните** във формите за **добавяне**, **редактиране** и **изтриване**

Форматиране на дата за преглед

- Форматираме **датата** за **преглед** при **добавяне** и **редактиране**
  - Избираме **Format** - **Custom**
  - Задаваме **CustomFormat** - **dd/MM/yyyy**







Doctor DTO и Patient DTO

- Създаваме **DoctorDto** и **PatientDto**

```
public class DoctorDto
{
    public int DoctorId { get; set; }
    public string DoctorFullName { get; set; }
}
```

Име и фамилия на лекар

```
public class PatientDto
{
    public int PatientId { get; set; }
    public string PatientFullName { get; set; }
}
```

Име и фамилия на пациент

Използване на Doctor DTO и Patient DTO (1)

SoftUni Foundation

```
private List<DoctorDto> LoadDoctorsToComboBox()
{
    using (var dbContext = new HospitalDbContext())
    {
        var doctors = dbContext.Doctors
                .Select(d => new DoctorDto
                {
                    DoctorId = d.DoctorId,
                    DoctorFullName = d.FirstName + " " + d.LastName,
                }).ToList();

         return doctors;
     }
 }
```

Задаваме име и фамилия на лекар

Използване на Doctor DTO и Patient DTO (2)

SoftUni Foundation

```
public class FormAddExamination : Form
{
    private PatientDto patient;
    private List<DoctorDto> doctors;

    public FormAddExamination(PatientDto patient, List<DoctorDto>
doctors)
    {
        ...
        this.textBoxAddExaminationPatientName.Text =
this.patient.PatientFullName;
        LoadDoctors();
    }
...
```

Задаваме име и фамилия на пациент

Използване на Doctor DTO и Patient DTO (3)

```
...
    public string PatientName => this.textBoxPatientName.Text;
    public string DoctorName => this.comboBoxDoctors.Text;
    ...

    private void LoadDoctors()
    {
        this.comboBoxAddExaminationDoctors.DisplayMember = "DoctorFullName";
        this.comboBoxAddExaminationDoctors.ValueMember = "DoctorId";
        this.comboBoxAddExaminationDoctors.DataSource = this.doctors;
    }
}
```

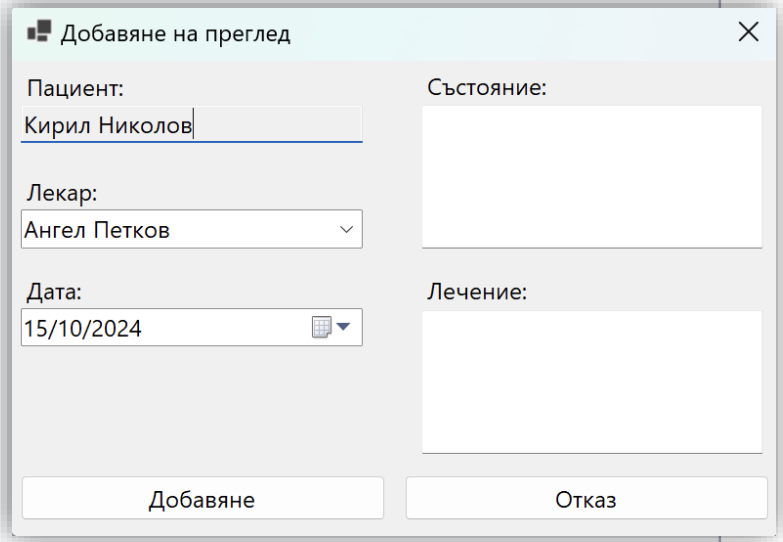
Зареждаме имената на всички лекари

Добавяне на Преглед (1)

```
private void buttonAddExamination_Click(object sender, EventArgs e)
{
    var doctors = LoadDoctorsToComboBox();

    if (doctors == null || doctors.Count == 0)
    {
        // TODO Показваме грешка - Няма
        return;
    }

    // Добавяме преглед през Пациенти - Покажи прегледи - Добави преглед
    var patient = LoadPatient();

    // Добавяме преглед през всички Прегледи - Добави преглед
    if (patientId == null)
    {
        var examination = (ExaminationDto)examinationDtoBindingSource.Current;
        patient = LoadPatient(examination);
    }
...
```

Зареждаме пациента, който сме избрали от Пациенти (по patientId)

Зареждаме пациента на избран преглед



Добавяне на Преглед (2)

SoftUni Foundation

```
...
    if (patient == null)
    {
        return; // TODO Показваме грешка - Пациентът не е намерен
    }
    var formAddExamination = new FormAddExamination(patient, doctors);
    if (formAddExamination.ShowDialog() == DialogResult.OK)
    {
        var selectedDoctor = doctors
                    .FirstOrDefault(d => d.DoctorFullName == formAddExamination.DoctorName);
        if (selectedDoctor == null)
        {
            return; // TODO Показваме грешка - Невалиден избор на лекар
        }
        if (string.IsNullOrWhiteSpace(formAddExamination.Condition) || ...)
        {
            return; // TODO Показваме грешка - Полето за състояние и лечение не може да
бъде празно
        }
...
```

Добавяне на Преглед (3)

SoftUni Foundation

```
...
        var newExamination = new Examination
        {
            PatientId = patient.PatientId,
            DoctorId = selectedDoctor.DoctorId,
            ExaminationDate = DateOnly
                                .ParseExact(formAddExamination.Date, "dd/MM/yyyy", null),
            ...
        };

        AddNewExamination(newExamination);
        ReloadExaminations();
    }
}

private List<DoctorDto> LoadDoctorsToComboBox() {...}
private PatientDto LoadPatient(Examination examination) {...}
private void AddNewExamination(Examination examination) {...}
```

Избиране на лекар спрямо роля

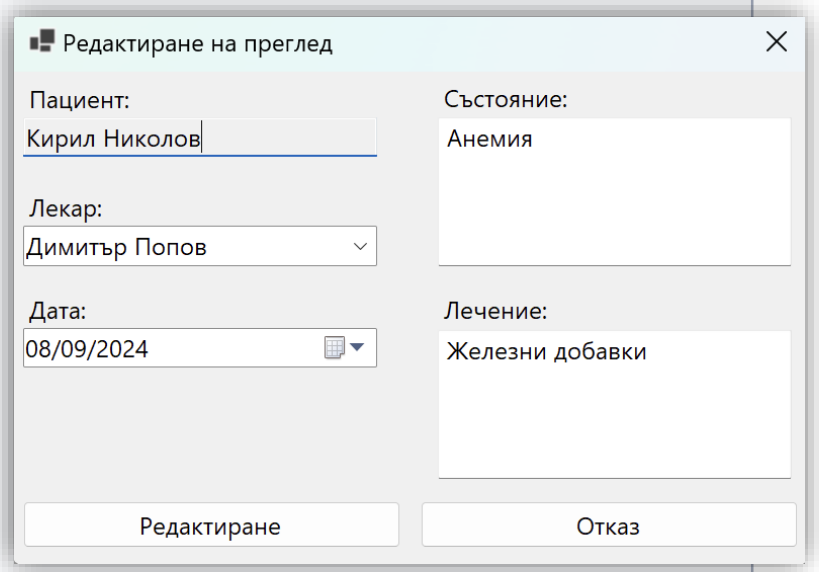
```
private List<DoctorDto> LoadDoctorsToComboBox()
{
    using (var dbContext = new HospitalDbContext())
    {
        var doctors = dbContext.Doctors.Select(d => new DoctorDto
                                {
                                    DoctorId = d.DoctorId,
                                    DoctorFullName = d.FirstName + " " + d.LastName,
                                }).ToList();
        // Лекарят не може да избира други лекари
        if (userId != null)
        {
            var doctorId = dbContext.Doctors
                        .FirstOrDefault(d => d.UserId == userId).DoctorId;
            doctors = doctors.Where(d => d.DoctorId == doctorId).ToList();
        }

        return doctors;
    }
}
```

Логнатият потребител е лекар

Лекарят има опция за лекар само себе си

Редактиране на Преглед (1)

```
private void buttonEditExamination_Click(object sender, EventArgs e)
{
    var selectedExamination = (ExaminationDto)examinationDtoBindingSource.Current;

    if (selectedExamination == null)
    {
        return; // TODO Показваме грешка - Не е избран преглед
    }

    var examination = GetExamination(selectedExamination);
    var patient = LoadPatient(examination);
    var doctors = LoadDoctorsToComboBox();

    if (doctors == null || doctors.Count == 0)
    {
        return; // TODO Показваме грешка - Няма налични лекари за избор
    }

    var doctor = doctors.FirstOrDefault(d => d.DoctorId == examination.DoctorId);
...
```



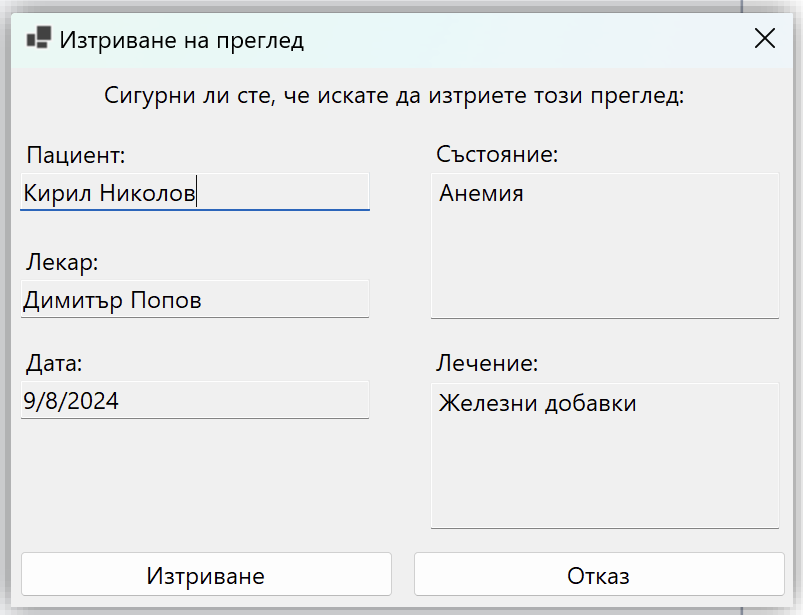
Редактиране на Преглед (2)

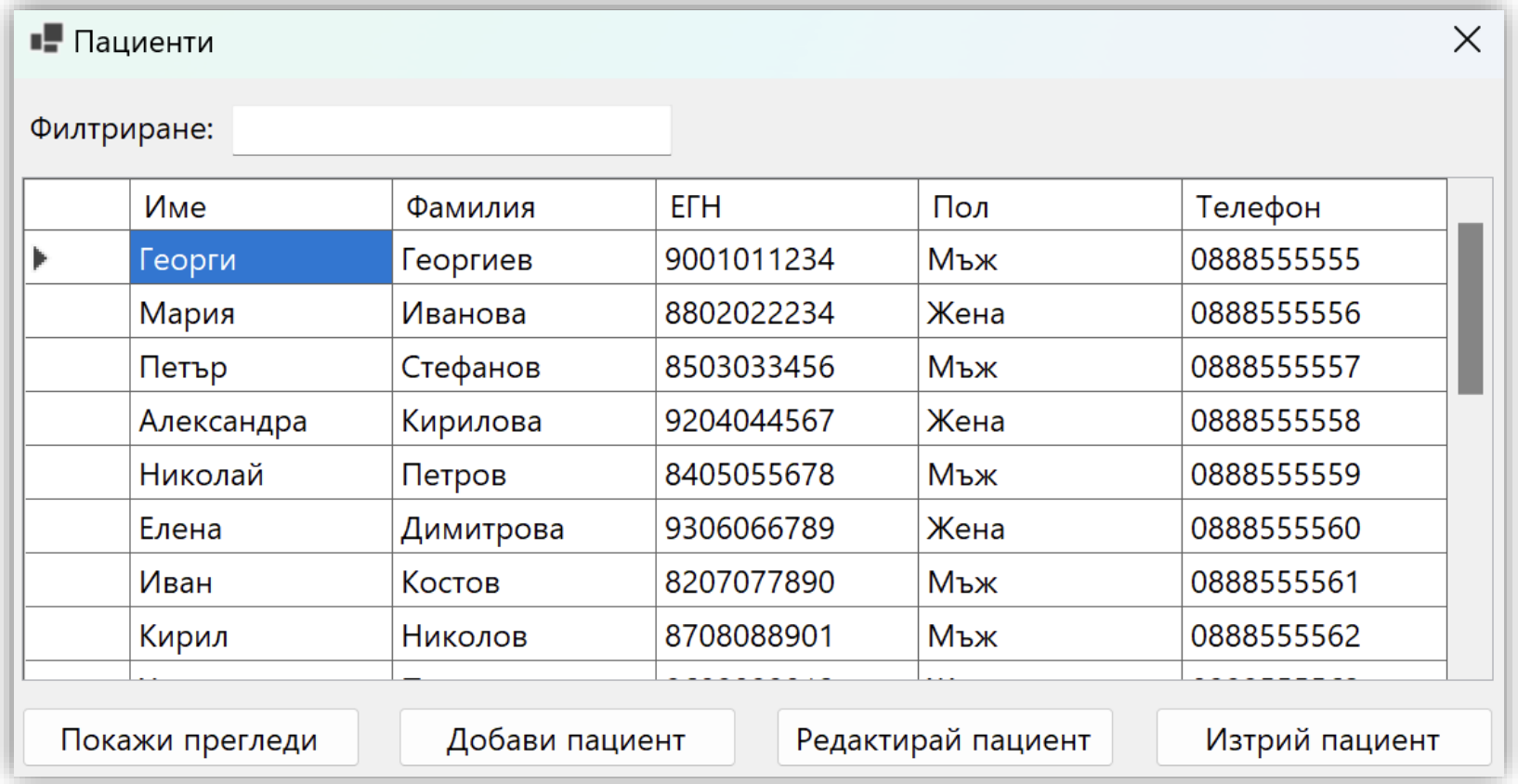
```
...
    if (doctor == null)
    {
        return; // TODO Показваме грешка - Лекарят не е намерен
    }
    var formEditExamination = new FormEditExamination(...);
    if (formEditExamination.ShowDialog() == DialogResult.OK)
    {
        if (string.IsNullOrWhiteSpace(formEditExamination.Condition) || ...)
        {
            return; // TODO Показваме грешка -
                    // Полето за състояние и лечение не може да бъде празно
        }
        examination.DoctorId = formEditExamination.DoctorId;
        ...
        EditExamination(examination);
        ReloadExaminations();
    }
}
private void EditExamination(Examination examination) {...}
```

Изтриване на Преглед

SoftUni Foundation

```
private void buttonDeleteExamination_Click(object sender, EventArgs e)
{
    var selectedExamination = (ExaminationDto)examinationDtoBindingSource.Current;
    if (selectedExamination == null)
    {
        return; // TODO Показваме грешка - Не е избран преглед
    }
    var examination = GetExamination(selectedExamination);
    var patient = LoadPatient(examination);
    var doctor = LoadDoctor(examination);
    var formDeleteExamination = new FormDeleteExamination(...);
    if (formDeleteExamination.ShowDialog() == DialogResult.OK)
    {
        DeleteExamination(examination);
        ReloadExaminations();
    }
}
private DoctorDto LoadDoctor(Examination examination) {...}
private void DeleteExamination(Examination examination) {...}
```





Четене, добавяне, редактиране и изтриване на пациент

CRUD операции на пациент

Съдържание на форми за Пациенти (1)

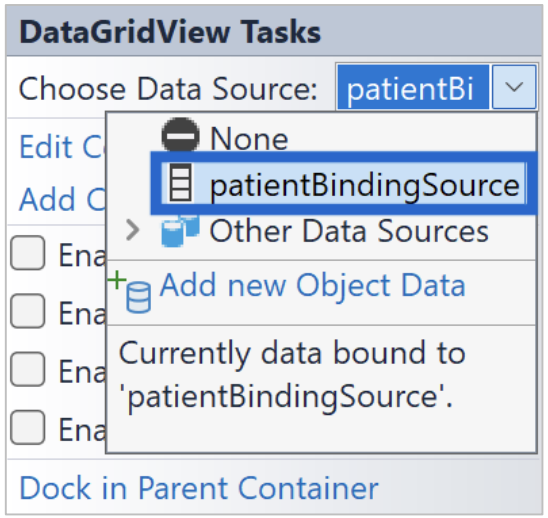
- Форма за **четене**
  - **Таблица** с **всички пациенти**
  - **Филтриране** на **пациенти**
  - **Бутони** за **показване на прегледи**, **добавяне**, **редактиране** и **изтриване на пациент**
- Форма за **добавяне**
  - **Свойства** за **име**, **фамилия**, **ЕГН, пол**, **телефон**
- Форма за **редактиране**
  - **Свойства** за **име**, **фамилия**, **ЕГН, пол**, **телефон**
  - Задаваме **име**, **фамилия**, **ЕГН**, **пол**, **телефон** през **конструктора**
- Форма за **изтриване**
  - Задаваме **име** и **фамилия** през **конструктора**

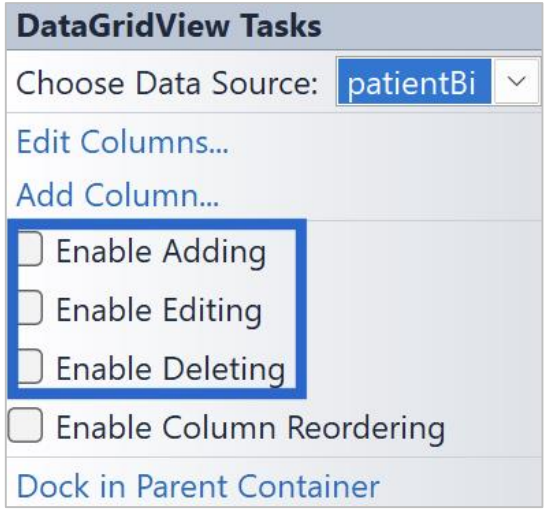
Съдържание на форми за Пациенти (2)

- Компоненти
  - **DataGridView - всички пациенти**
  - **Label - име, фамилия, ЕГН, пол, телефон, филтриране**
  - **TextBox - име, фамилия, ЕГН, телефон, филтриране**
  - **ComboBox - пол**
  - **Button - показване на прегледи, добавяне, редактиране, изтриване, отказ**

Добавяне на DataSource и стилизиране на DataGridView

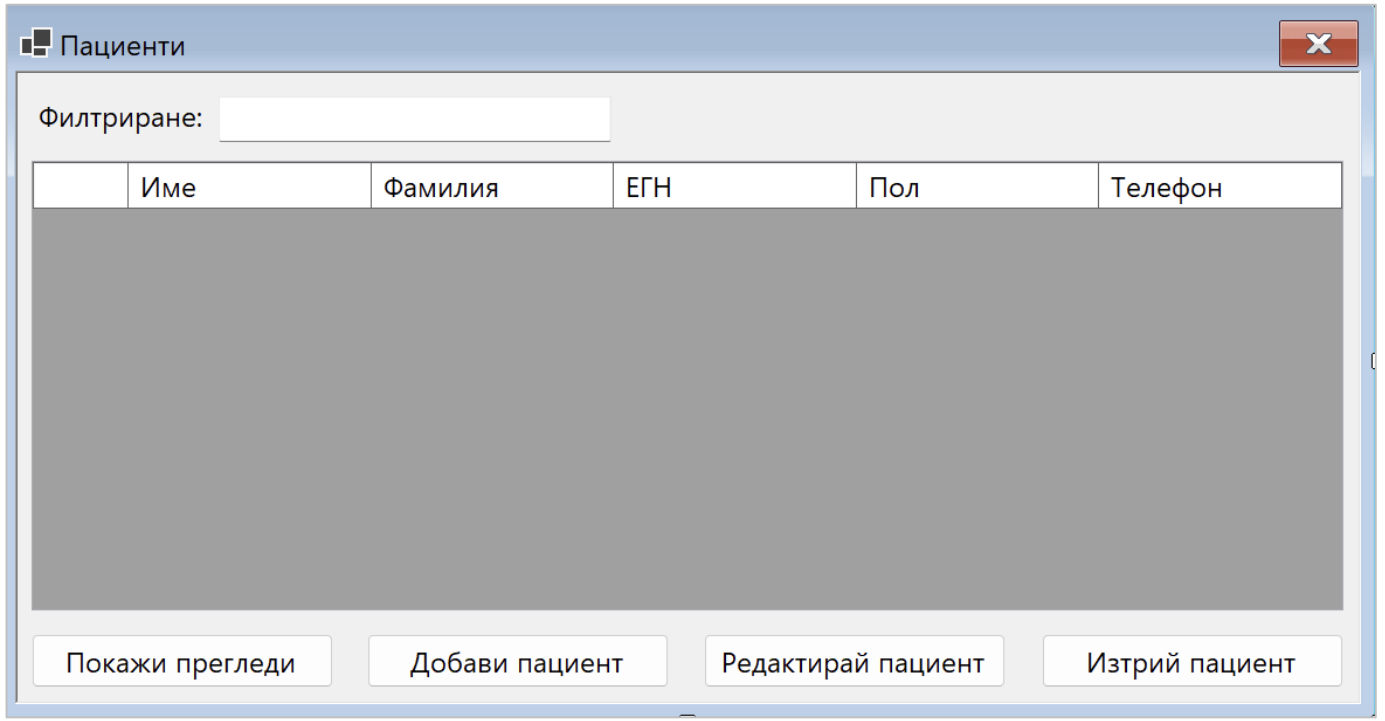
- Добавяме **DataSource** на **DataGridView** за **пациенти**
- Редактираме **имената** и **размерите** на **колоните**
- Задаваме следните **свойства** на **DataGridView**:
  - **EnableAdding** - **False**
  - **EnableEditing** - **False**
  - **EnableDeleting** - **False**





Стилизиране на форми

- Задаваме следните **свойства** на **формите**:
  - **StartPosition - CenterScreen**
  - **FormBorderStyle - Fixed3D**
  - **MaximizeBox - False**
  - **MinimizeBox - False**



Редактиране на главна форма за лекари

- При **FormMainDoctor**, взимаме **userId** и го **подаваме** на **формата за пациенти**

```
private int userId;

public FormMainDoctors(int userId)
{
    InitializeComponent();
    this.userId = userId;
}

private void buttonPatients_Click(object sender, EventArgs e)
{
    var formPatients = new FormPatients(userId);
    formPatients.ShowDialog();
}
```

Задаваме userId

Подаваме userId

Функционалност спрямо роля

- Създаваме **втори конструктор**, който приема **userId** при **логнат лекар**
- Ако **потребителят** е **лекар**, **скриваме бутоните** за манипулация на **пациенти**

```
private int? userId;

public FormPatients()
{
    InitializeComponent();
}

public FormPatients(int? userId)
{
    InitializeComponent();
    this.userId = userId;

    // Бутоните за добавяне, редактиране и изтриване на пациент няма да се достъпват от лекар
    this.buttonAddPatient.Visible = false;
    this.buttonEditPatient.Visible = false;
    this.buttonDeletePatient.Visible = false;
}
```

Четене на Пациенти

- Зареждаме **пациентите** при **отваряне** на **формата**

```
private void FormPatients_Load(object sender, EventArgs e){...}
private List<Patient> LoadPatientsFormDb()
{
    using (var db = new HospitalDbContext())
    {
        // Лекарят е влязъл от Пациенти на главната форма
        if (userId != null)
        {
            var doctorId = db.Doctors.FirstOrDefault(d => d.UserId == userId).DoctorId;
            return db.Patients.Where(p => p.Examinations
                                          .Any(e => e.DoctorId ==
doctorId)).ToList();
        }
        return db.Patients.ToList();
    }
}
private void ReloadPatients(){...}
```

Връщаме пациентите на логнатия лекар

Връщаме всички пациенти

Филтриране на пациенти

- Имплементираме **филтриране** на **пациенти** по **ЕГН**, **име**, **фамилия** или **телефон**

```
private void textBoxFilterPatients_TextChanged(object sender, EventArgs e)
{
    using (var dbContext = new HospitalDbContext())
    {
        var filterText = this.textBoxFilterPatients.Text;
        var filteredPatients = dbContext.Patients
            .Where(p => p.PersonalIdNumber.Contains(filterText) ||
                   p.FirstName.Contains(filterText) ||
                   p.LastName.Contains(filterText) ||
                   p.Phone.Contains(filterText))
           .ToList();

        this.patientBindingSource.DataSource = filteredPatients;
    }
}
```

Методи-обработчици и DialogResult за бутони

- Добавяме **методи-обработчици** към **бутоните**
  - **Прегледи на пациент**
  - **Добавяне на пациент**
  - **Редактиране на пациент**
  - **Изтриване на пациент**
- Задаваме подходящ **DialogResult** на **бутоните** във формите за **добавяне**, **редактиране** и **изтриване**

Четене на прегледи на избран пациент

- При **избран пациент**, показваме неговите **прегледи**
  - Ако логнатия потребител е **админ**, показваме **всички прегледи**
  - Ако логнатия потребител е **лекар**, показваме само **неговите прегледи** с **пациента**

```
private void buttonShowExaminations_Click(object sender, EventArgs e)
{
    var selectedPatient = (Patient)patientBindingSource.Current;

    if (selectedPatient != null)
    {
        var formExaminations = new FormExaminations(selectedPatient.PatientId);

        if (userId != null)
        {
            formExaminations = new FormExaminations(selectedPatient.PatientId, userId);
        }

        formExaminations.ShowDialog();
    }
}
```

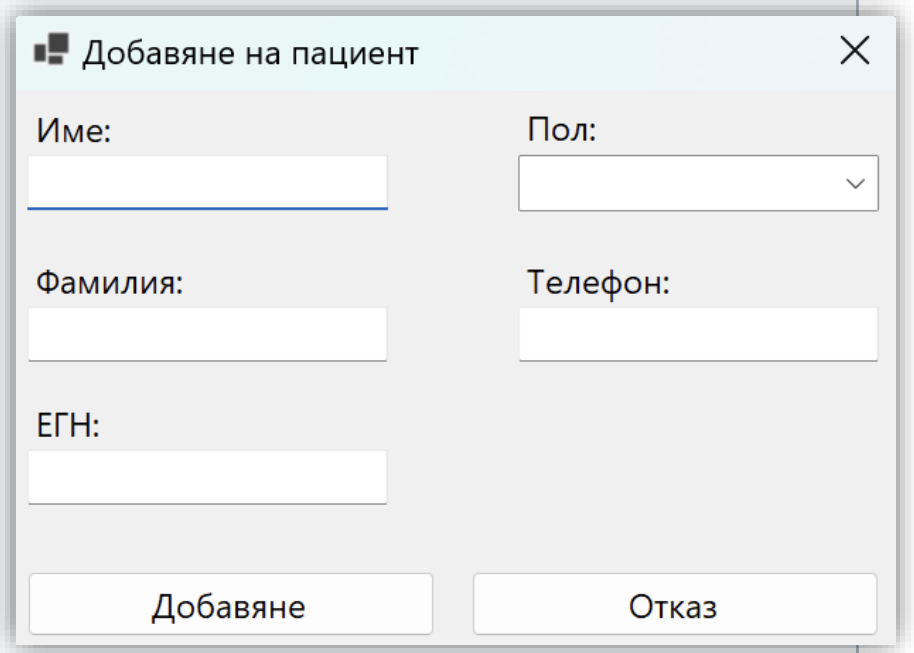
Логнатия потребител е **админ**

Логнатия потребител е **лекар**

Добавяне на Пациент

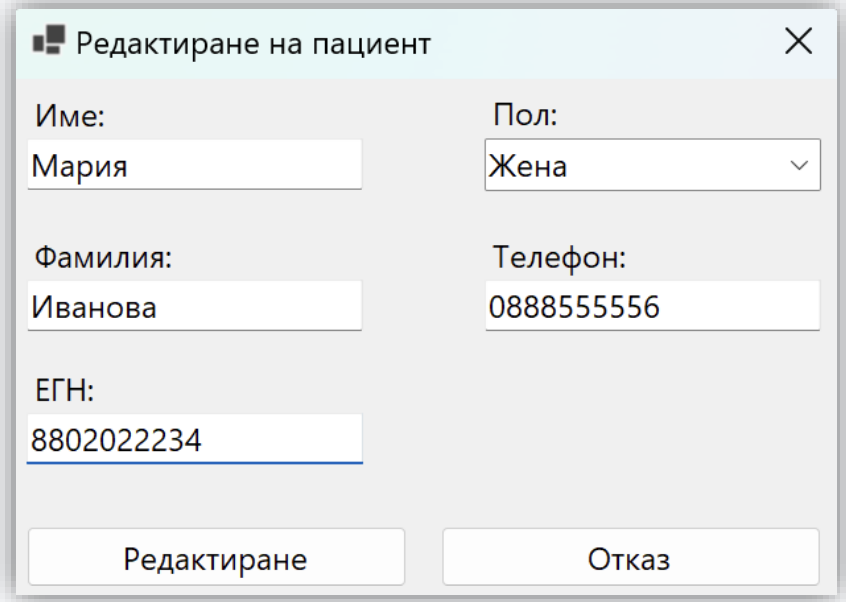
SoftUni Foundation

```
private void buttonAddPatient_Click(object sender, EventArgs e)
{
    var formAddPatient = new FormAddPatient();
    if (formAddPatient.ShowDialog() == DialogResult.OK)
    {
         if (string.IsNullOrWhiteSpace(formAddPatient.FirstName) ||...)
         {
             return; // TODO Показваме грешка - Всички полета трябва да бъдат попълнени
         }
        var newPatient = new Patient
        {
            FirstName = formAddPatient.FirstName,
            ...
        };

        AddNewPatient(newPatient);
        ReloadPatients();
    }
}

private void AddNewPatient(Patient patient) {...}
```



Редактиране на Пациент

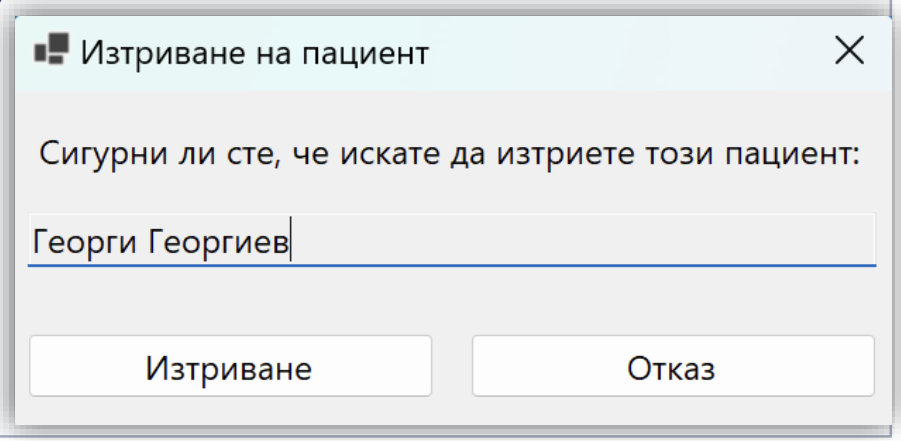
```
private void buttonEditPatient_Click(object sender, EventArgs e)
{
    var selectedPatient = (Patient)patientBindingSource.Current;

    if (selectedPatient == null)
    {
        return; // TODO Показваме грешка - Няма избран пациент
    }

    var formEditPatient = new FormEditPatient(selectedPatient.FirstName, ...);

    if (formEditPatient.ShowDialog() == DialogResult.OK)
    {
        selectedPatient.FirstName = formEditPatient.FirstName;
        ...

        EditPatient(selectedPatient);
        ReloadPatients();
    }
}

private void EditPatient(Patient patient) {...}
```

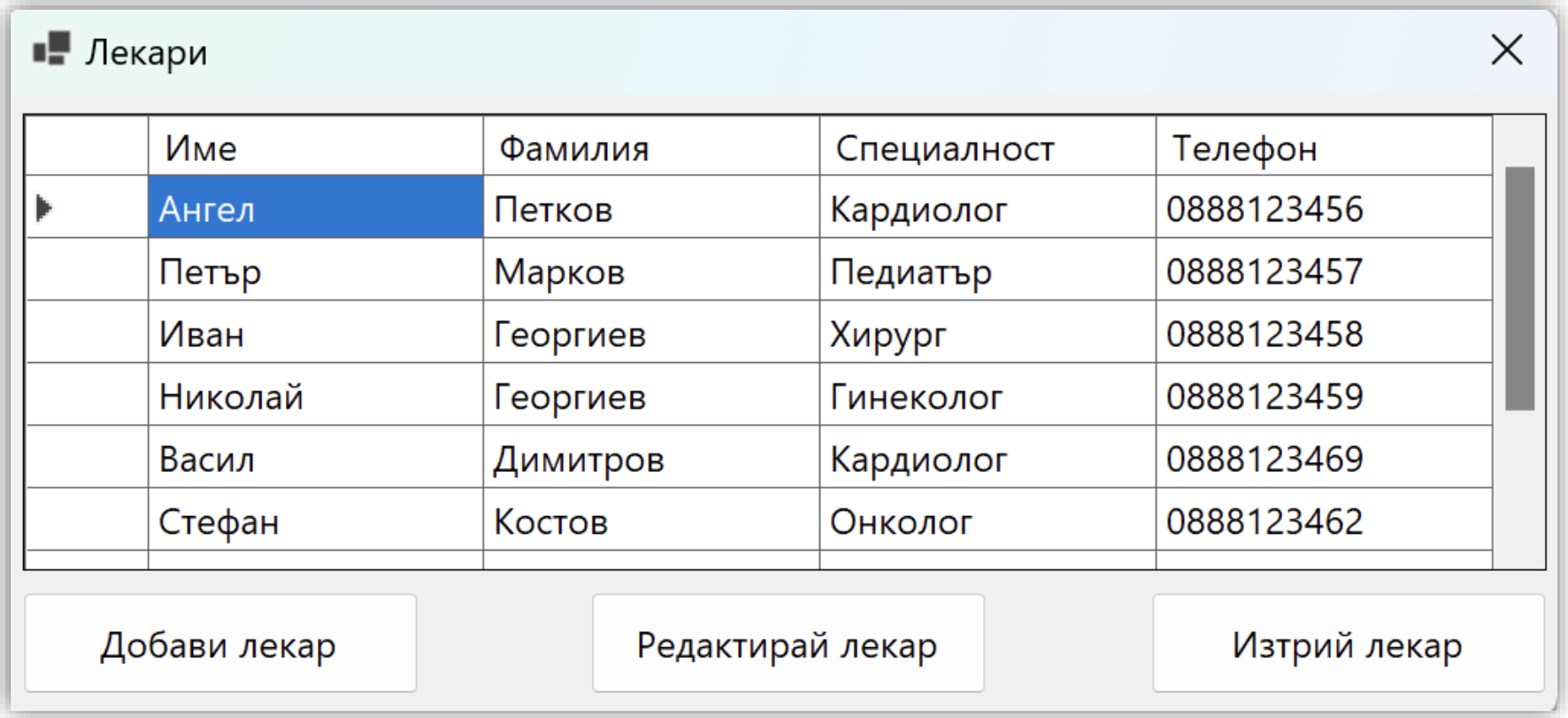


Изтриване на Пациент

```
private void buttonDeletePatient_Click(object sender, EventArgs e)
{
    var selectedPatient = (Patient)patientBindingSource.Current;

    if (selectedPatient == null)
    {
        return; // TODO Показваме грешка - Няма избран пациент
    }

    var patientName = ...;
    var formDeletePatient = new FormDeletePatient(patientName);
    if (formDeletePatient.ShowDialog() == DialogResult.OK)
    {
        DeletePatient(selectedPatient);
        ReloadPatients();
     }
}

private void DeletePatient(Patient patient) {...}
```

Изтриваме и свързаните с пациента прегледи





Четене, добавяне, редактиране и изтриване на лекар

CRUD операции на лекар

Съдържание на форми за Лекари (1)

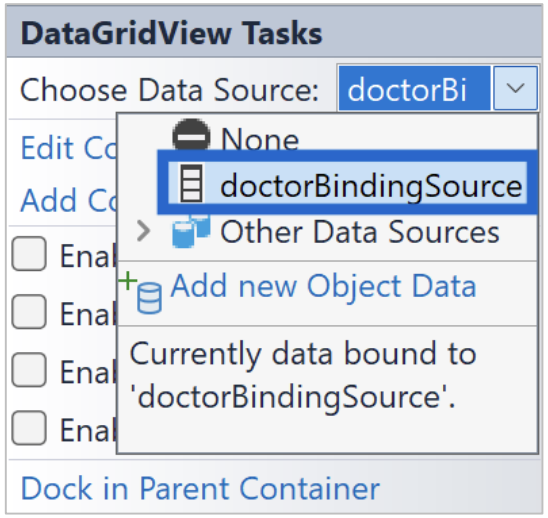
- Форма за **четене**
  - **Таблица** с **всички лекари**
  - **Бутони** за **добавяне**, **редактиране** и **изтриване** на **лекар**
- Форма за **добавяне**
  - Свойства за **име**, **фамилия**, **специалност**, **телефон**, **потребителско име**, **парола**
- Форма за **редактиране**
  - Свойства за **име**, **фамилия**, **специалност**, **телефон**, **потребителско име**, **парола**
  - Задаваме **име**, **фамилия**, **специалност**, **телефон**, **потребителско име**, **парола** през **конструктора**
- Форма за **изтриване**
  - Задаваме **име** и **фамилия** през **конструктора**

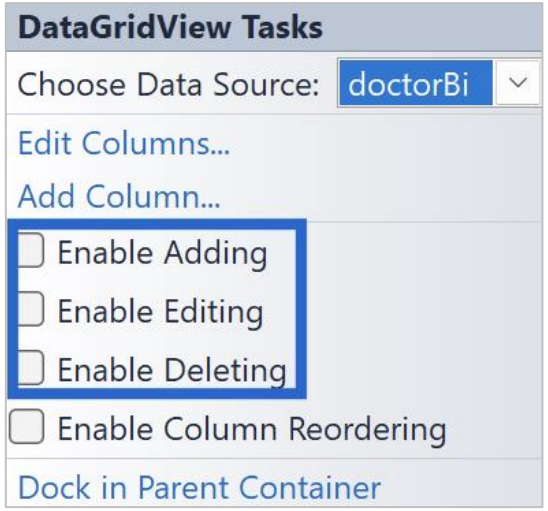
Съдържание на форми за Лекари (2)

- Компоненти
  - `DataGridView` - **всички лекари**
  - `Label` - **име, фамилия, специалност, телефон, потребителско име, парола**
  - `TextBox` - **име, фамилия, специалност, телефон, потребителско име, парола**
  - `Button` - **добавяне, редактиране, изтриване, отказ**

Добавяне на DataSource и стилизиране на DataGridView

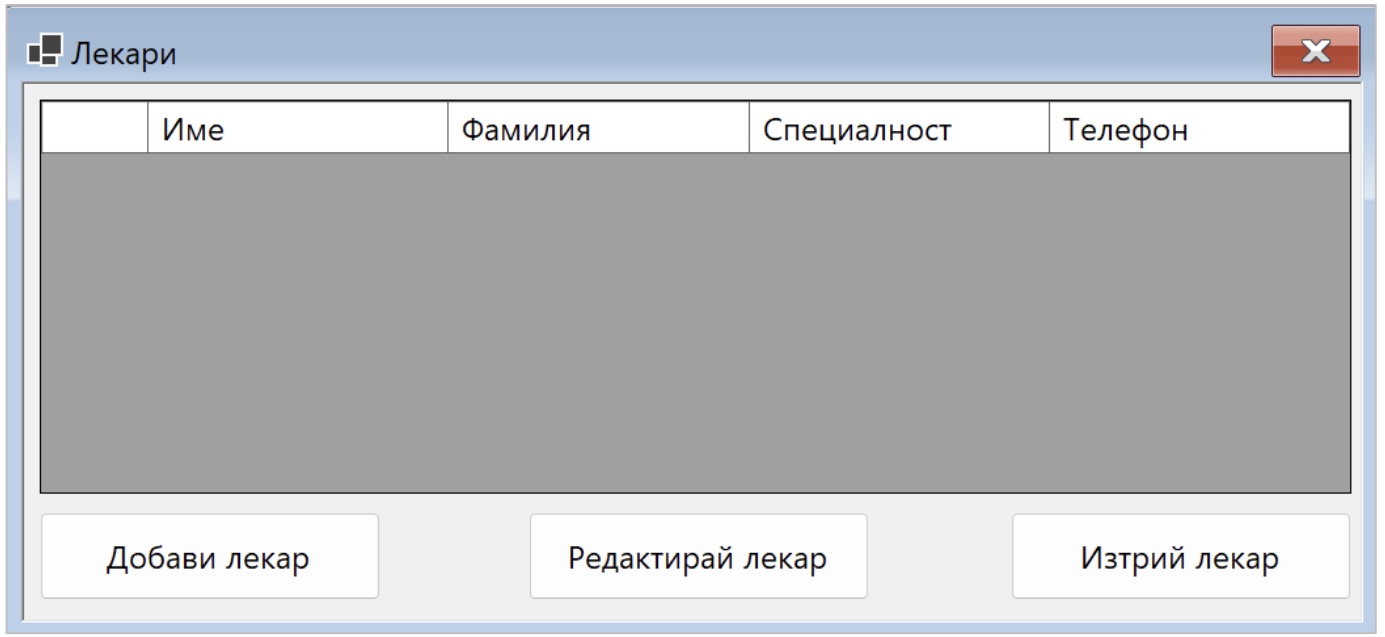
- Добавяме **DataSource** на **DataGridView** за **лекари**
- Редактираме **имената** и **размерите** на **колоните**
- Задаваме следните **свойства** на **DataGridView**:
  - **EnableAdding** - **False**
  - **EnableEditing** - **False**
  - **EnableDeleting** - **False**





Стилизиране на форми

- Задаваме следните **свойства** на **формите**:
  - **StartPosition - CenterScreen**
  - **FormBorderStyle - Fixed3D**
  - **MaximizeBox - False**
  - **MinimizeBox - False**



Четене на Лекари

- Зареждаме **лекарите** при **отваряне** на **формата**

```
private void FormDoctors_Load(object sender, EventArgs e)
{
    ReloadDoctors();
}

 private List<Doctor> LoadDoctorsFormDb()
{
    using (var db = new HospitalDbContext())
    {
        return db.Doctors.ToList();
    }
}

private void ReloadDoctors()
{
    var doctors = LoadDoctorsFormDb();
    this.doctorBindingSource.DataSource = doctors;
}
```

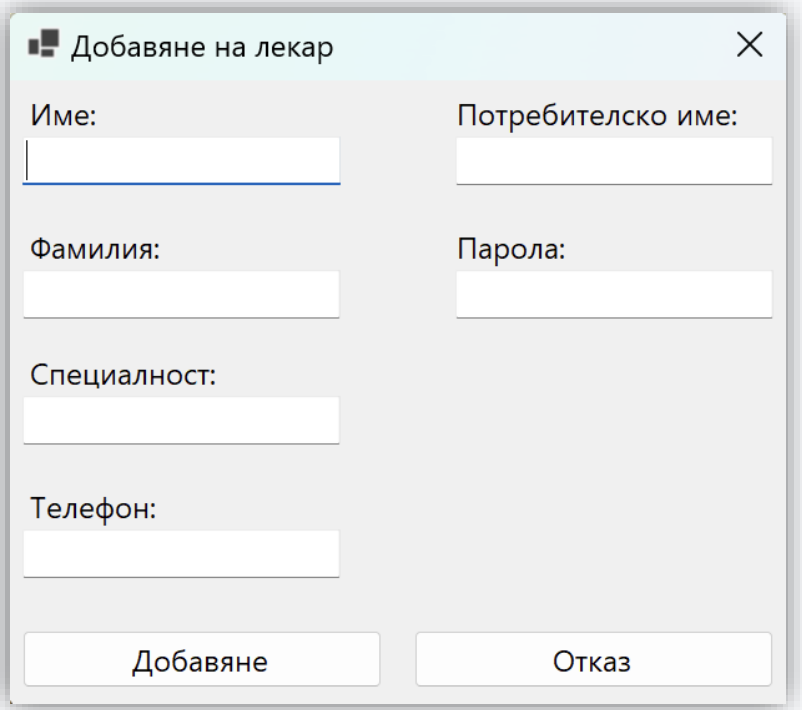
Методи-обработчици и DialogResult за бутони

SoftUni Foundation

- Добавяме **методи-обработчици** към **бутоните**
  - **Добавяне на лекар**
  - **Редактиране на лекар**
  - **Изтриване на лекар**
- Задаваме подходящ **DialogResult** на **бутоните** във формите за **добавяне**, **редактиране** и **изтриване**

Добавяне на Лекар (1)

```
private void buttonAddDoctor_Click(object sender, EventArgs e)
{
    var formAddDoctor = new FormAddDoctor();

    if (formAddDoctor.ShowDialog() == DialogResult.OK)
    {
        if (string.IsNullOrWhiteSpace(formAddDoctor.FirstName) ||...)
        {
            // TODO Показваме грешка -
            // Всички полета трябва да бъдат попълнени
            return;
        }

        var newUser = new User
        {
            Username = formAddDoctor.Username,
            PasswordHash = formAddDoctor.Password,
            RoleId = 2 // Роля на лекар
        };
...
```



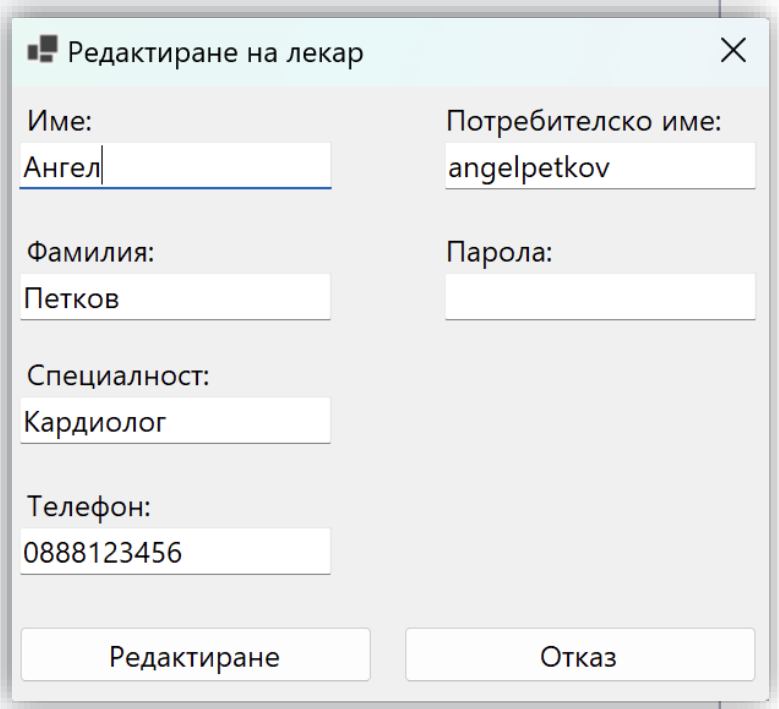
Добавяне на Лекар (2)

```
...
            // Използваме UserHelper за добавяне на потребител
            if (!UserHelper.AddNewUser(newUser))
            {
                return; // Ако потребителят вече съществува, прекратяваме
            }

            var newDoctor = new Doctor
            {
                FirstName = formAddDoctor.FirstName,
                ...,
                UserId = newUser.UserId
            };

            AddNewDoctor(newDoctor);
            ReloadDoctors();
    }
}

private void AddNewDoctor(Doctor doctor) {...}
```

Редактиране на Лекар (1)

SoftUni Foundation

```
private void buttonEditDoctor_Click(object sender, EventArgs e)
{
    var selectedDoctor = (Doctor)doctorBindingSource.Current;

    if (selectedDoctor == null)
    {
        return; // TODO Показваме грешка - Няма избран лекар
    }

    var user = UserHelper.GetUserById(selectedDoctor.UserId);
    var formEditDoctor = new FormEditDoctor(...);

    if (formEditDoctor.ShowDialog() == DialogResult.OK)
    {
        if (string.IsNullOrWhiteSpace(formEditDoctor.FirstName) || ...)
        {
            return; // TODO Показваме грешка - Всички полета трябва да бъдат попълнени
        }
...
```



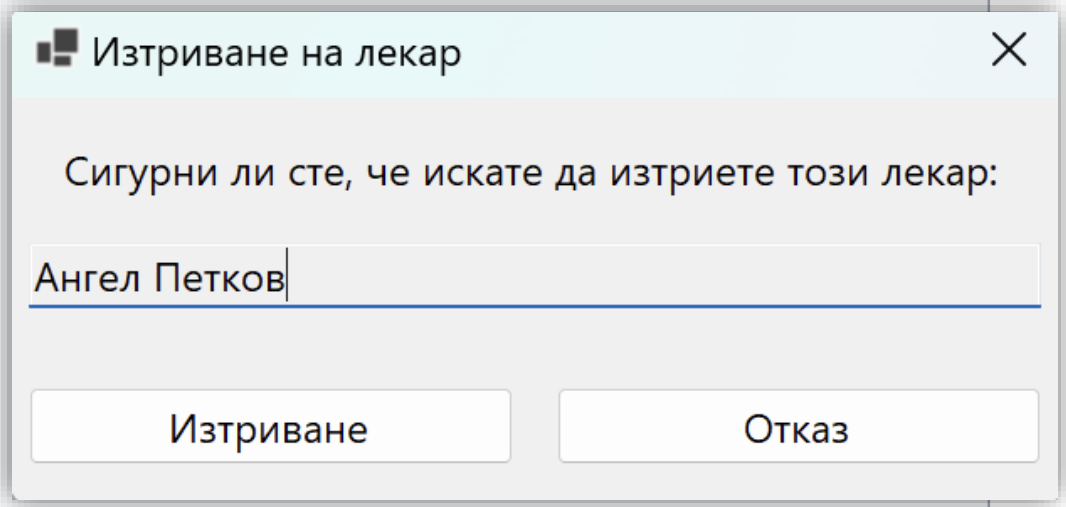
Редактиране на Лекар (2)

```
...
        selectedDoctor.FirstName = formEditDoctor.FirstName;
        ...
        EditDoctor(selectedDoctor);
        user.Username = formEditDoctor.Username;
        if (!string.IsNullOrWhiteSpace(formEditDoctor.Password))
        {
            user.PasswordHash = formEditDoctor.Password;
        }

        if (!UserHelper.EditUser(user))
        {
            return; // Ако потребителят вече съществува, прекратяваме
        }

        ReloadDoctors();
    }
}

private void EditDoctor(Doctor doctor) {...}
```

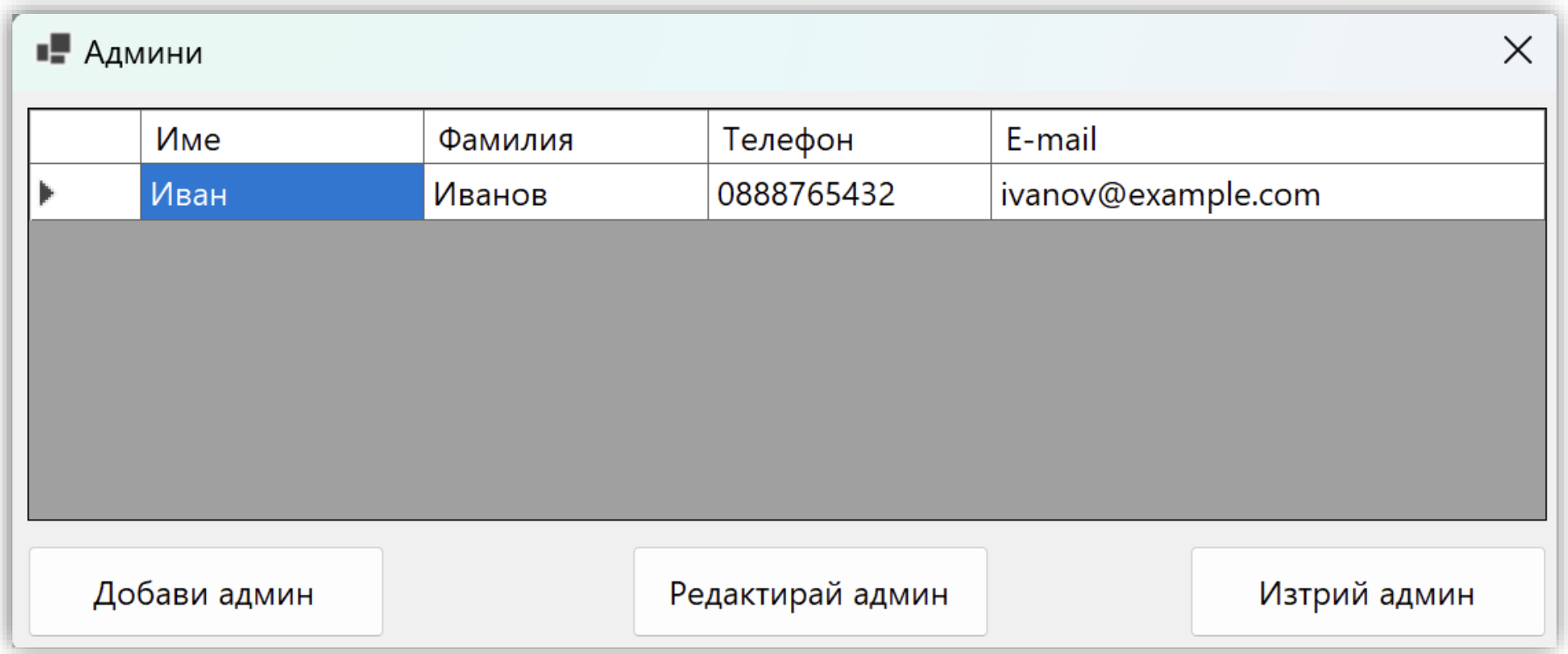
Изтриване на Лекар

SoftUni Foundation

```
private void buttonDeleteDoctor_Click(object sender, EventArgs e)
{
    var selectedDoctor = (Doctor)doctorBindingSource.Current;
    var selectedUser = UserHelper.GetUserById(selectedUser.UserId);

    if (selectedDoctor == null || selectedUser == null)
    {
        return; // TODO Показваме грешка - Няма избран лекар
    }

    var doctorName = ...;
    var formDeleteDoctor = new FormDeleteDoctor(doctorName);

    if (formDeleteDoctor.ShowDialog() == DialogResult.OK)
    {
        DeleteDoctor(selectedDoctor);
        UserHelper.DeleteUser(selectedUser);
        ReloadDoctors();
    }
}

private void DeleteDoctor(Doctor doctor) {...}
```



Изтриваме и **свързаните** с **лекаря прегледи**



Четене, добавяне, редактиране и изтриване на админ

CRUD операции на админ

Съдържание на форми за Администратори (1)

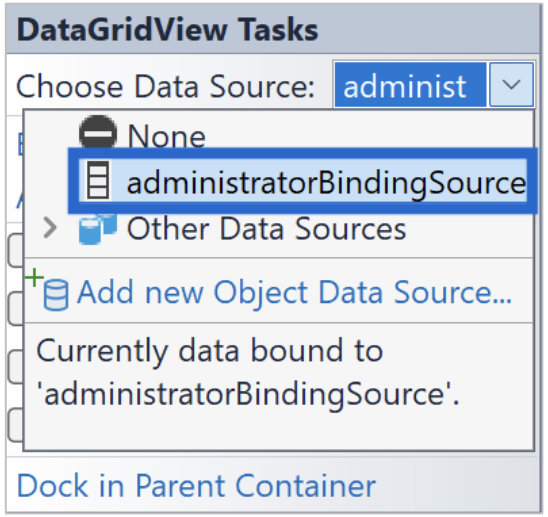
- Форма за **четене**
  - **Таблица** с **всички администратори**
  - **Бутони** за **добавяне**, **редактиране** и **изтриване** на **администратор**
- Форма за **добавяне**
  - Свойства за **име**, **фамилия**, **телефон**, **e-mail**, **потребителско име**, **парола**
- Форма за **редактиране**
  - Свойства за за **име**, **фамилия**, **телефон**, **e-mail**, **потребителско име**, **парола**
  - Задаваме за **име**, **фамилия**, **телефон**, **e-mail**, **потребителско име**, **парола** през **конструктора**
- Форма за **изтриване**
  - Задаваме **име** и **фамилия** през **конструктора**

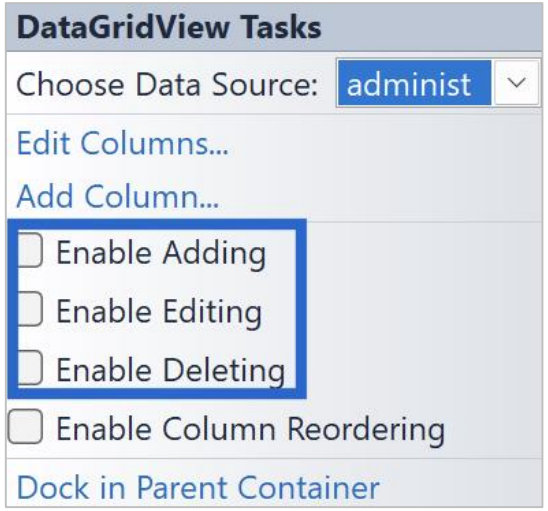
Съдържание на форми за Администи (2)

- Компоненти
  - **`DataGridView` - всички администи**
  - **`Label` - име, фамилия, телефон, e-mail, потребителско име, парола**
  - **`TextBox` - за име, фамилия, телефон, e-mail, потребителско име, парола**
  - **`Button` - добавяне, редактиране, изтриване, отказ**

Добавяне на DataSource и стилизиране на DataGridView

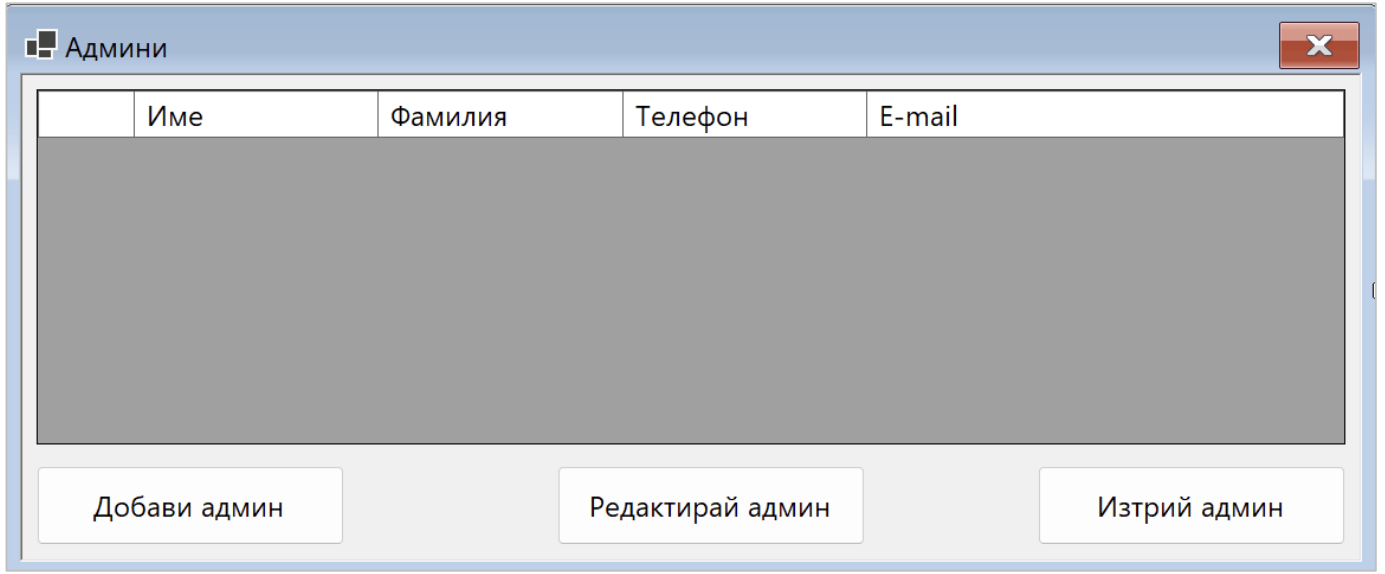
- Добавяме **DataSource** на **DataGridView** за **пациенти**
- Редактираме **имената** и **размерите** на **колоните**
- Задаваме следните **свойства** на **DataGridView**:
  - **EnableAdding** - **False**
  - **EnableEditing** - **False**
  - **EnableDeleting** - **False**





Стилизиране на форми

- Задаваме следните **свойства** на **формите**:
  - **StartPosition - CenterScreen**
  - **FormBorderStyle - Fixed3D**
  - **MaximizeBox - False**
  - **MinimizeBox - False**



Четене на Админи

- Зареждаме **администе** при **отваряне** на **формата**

```
private void FormAdmins_Load(object sender, EventArgs e)
{
    ReloadAdmins();
}

private List<Administrator> LoadAdminsFormDb()
{
    using (var db = new HospitalDbContext())
    {
        return db.Administrators.ToList();
    }
}

private void ReloadAdmins()
{
    var admins = LoadAdminsFormDb();
    this.administratorBindingSource.DataSource = admins;
}
```

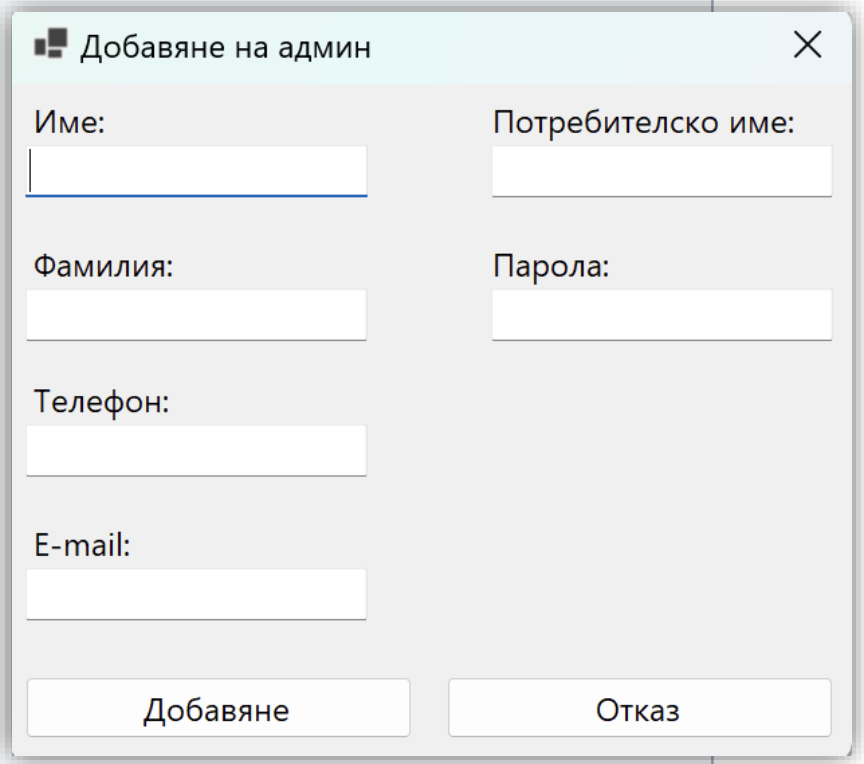
Методи-обработчици и DialogResult за бутони

SoftUni Foundation

- Добавяме **методи-обработчици** към **бутоните**
  - **Добавяне на админ**
  - **Редактиране на админ**
  - **Изтриване на админ**
- Задаваме подходящ **DialogResult** на **бутоните** във формите за **добавяне**, **редактиране** и **изтриване**

Добавяне на Админ (1)

```
private void buttonAddAdmin_Click(object sender, EventArgs e)
{
    var formAddAdmin = new FormAddAdmin();

    if (formAddAdmin.ShowDialog() == DialogResult.OK)
    {
        if (string.IsNullOrWhiteSpace(formAddAdmin.FirstName)
        {
            // TODO Показваме грешка -
            // Всички полета трябва да бъдат попълнени
            return;
        }
        var newUser = new User
        {
            Username = formAddAdmin.Username,
            PasswordHash = formAddAdmin.Password,
            RoleId = 1 // Роля на админ
        };
...
```



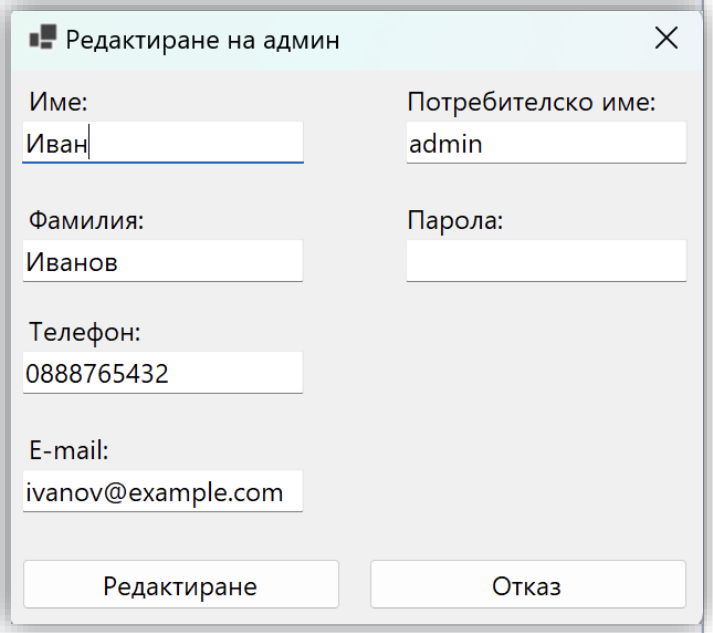
Добавяне на Админ (2)

SoftUni Foundation

```
...
        // Използваме UserHelper за добавяне на потребител
        if (!UserHelper.AddNewUser(newUser))
        {
            return; // Ако потребителят вече съществува, прекратяваме
        }
        var newAdmin = new Administrator
        {
            FirstName = formAddAdmin.FirstName,
            ...
            UserId = newUser.UserId
        };

        AddNewAdmin(newAdmin);
        ReloadAdmins();
    }
}
private void AddNewAdmin(Administrator admin) {...}
```

Редактиране на Админ (1)

```
private void buttonEditAdmin_Click(object sender, EventArgs e)
{
    var selectedAdmin = (Administrator)administratorBindingSource.Current;

    if (selectedAdmin == null)
    {
        return; // TODO Показваме грешка - Няма избран admin
    }

    var user = UserHelper.GetUserById(selectedAdmin.UserId);
    var formEditAdmin = new FormEditAdmin(...);

    if (formEditAdmin.ShowDialog() == DialogResult.OK)
    {
        if (string.IsNullOrWhiteSpace(formEditAdmin.FirstName) || ...)
        {
            return; // TODO Показваме грешка - Всички полета трябва да бъдат попълнени
        }
...
```



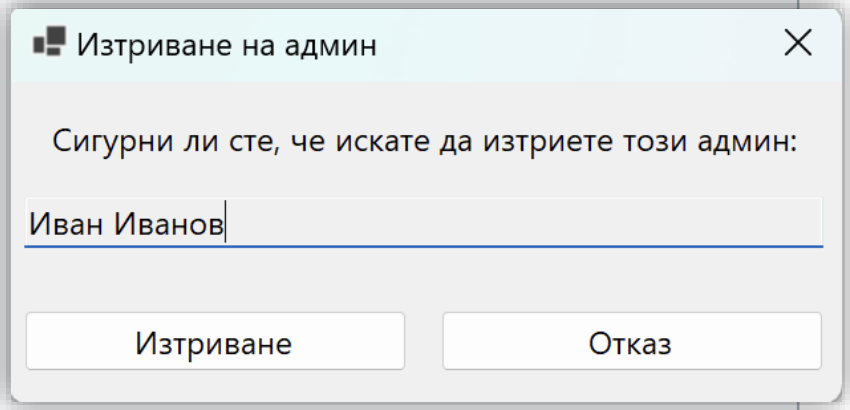
Редактиране на Админ (2)

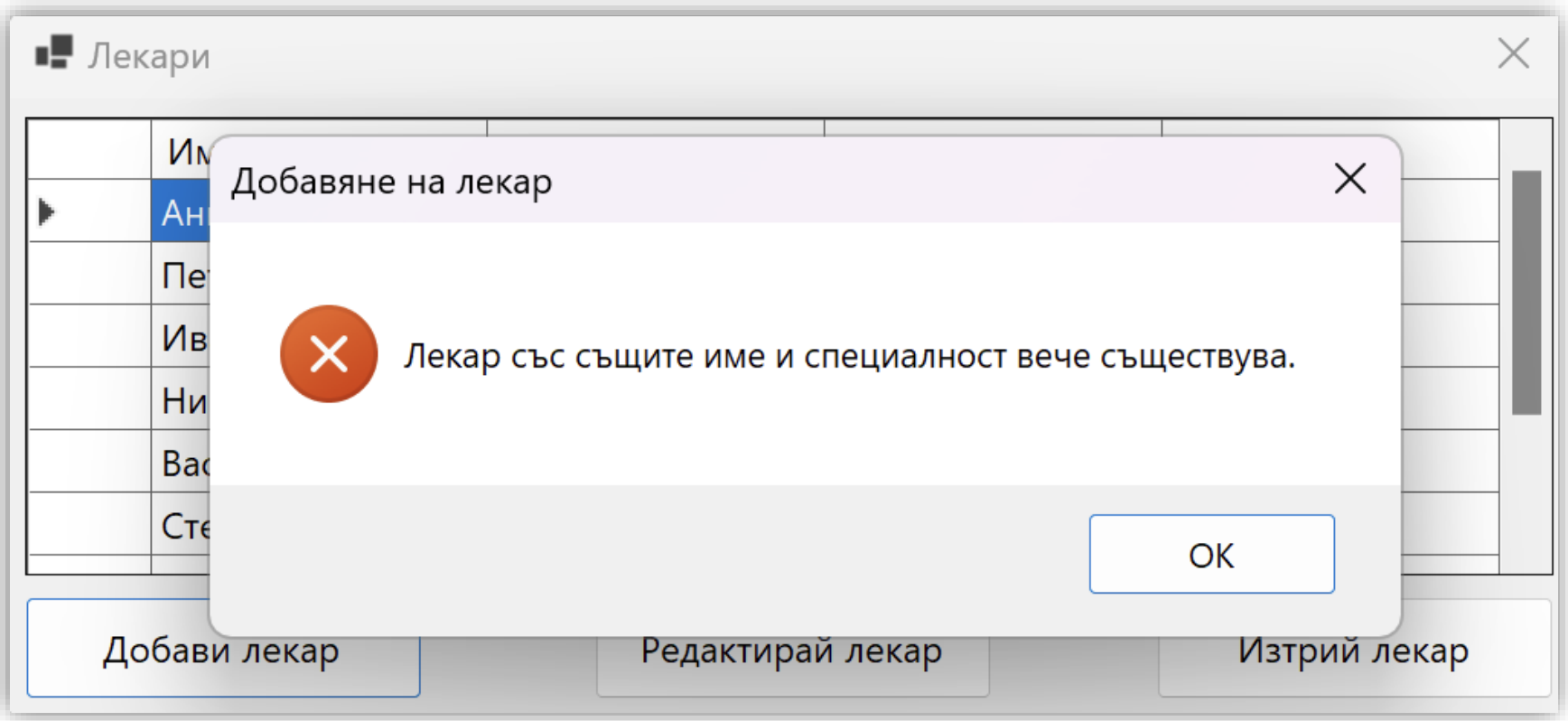
```
...
        selectedAdmin.FirstName = formEditAdmin.FirstName;
        ...
        EditAdmin(selectedAdmin);
        user.Username = formEditAdmin.Username;
        if (!string.IsNullOrWhiteSpace(formEditAdmin.Password))
        {
            user.PasswordHash = formEditAdmin.Password;
        }
        if (!UserHelper.EditUser(user))
        {
            return; // Ако потребителят вече съществува, прекратяваме
        }

        ReloadAdmins();
    }
}
private void EditAdmin(Administrator admin) {...}
```

Изтриване на Админ

SoftUni Foundation

```
private void buttonDeleteAdmin_Click(object sender, EventArgs e)
{
    var selectedAdmin = (Administrator)administratorBindingSource.Current;
    var selectedUser = UserHelper.GetUserById(selectedAdmin.UserId);
    if (selectedAdmin == null || selectedUser == null)
    {
        return; // TODO Показваме грешка - Няма избран админ
    }
    var adminName = ...;
    var formDeleteAdmin = new FormDeleteAdmin(adminName);
    if (formDeleteAdmin.ShowDialog() == DialogResult.OK)
    {
        DeleteAdmin(selectedAdmin);
        UserHelper.DeleteUser(selectedUser);
        ReloadAdmins();
    }
}
private void DeleteAdmin(Administrator admin) {...}
```





Допълнителни проверки

Уверяване за валидни данни

Допълнителни проверки (1)

- Добавяме **допълнителни проверки** в **методите** за **CRUD операции**
- **Пациенти**
  - **Телефонът** е **винаги** с **10 цифри**

```
// Проверка за валиден телефонен номер
if (string.IsNullOrWhiteSpace(patient.Phone) ||
                    patient.Phone.Length != 10 || !patient.Phone.All(char.IsDigit))
{
    // TODO Показваме грешка - Грешен телефонен номер
    return;
}
```

Допълнителни проверки (2)

- **ЕГН винаги** е **уникално**

```
// Проверка за дублиране на ЕГН
if (dbContext.Patients.Any(p => p.PersonalIdNumber == patient.PersonalIdNumber))
{
    // TODO Показваме грешка - Вече има пациент с това ЕГН
    return;
}
```

- **Винаги** има избран валиден **пол**

```
// Проверка за валиден пол
if (patient.Gender != "Мъж" && patient.Gender != "Жена")
{
    // TODO Показваме грешка - Не е избран валиден пол
    return;
}
```

Допълнителни проверки (3)

- **Имената** не са **прекалено дълги** (например до **25** символа)

```
// Проверка за прекалено дълги имена
if (patient.FirstName.Length > 25 || patient.LastName.Length > 25)
{
    // TODO Показваме грешка - Прекалено дълго име
    return;
}
```

Допълнителни проверки (4)

- **Лекари**
  - **Винаги** има само **един лекар** с избраните **имена** и **специалност**

```
// Проверка дали вече съществува лекар със същите име и специалност
var existingDoctor = dbContext.Doctors.FirstOrDefault(d =>
                     d.FirstName == doctor.FirstName &&
                     d.LastName == doctor.LastName && d.Speciality == doctor.Speciality);

if (existingDoctor != null)
{
    // TODO Показваме грешка - Лекар със същите име и специалност вече съществува
    return;
}
```

Допълнителни проверки (5)

- **Имената** не са **прекалено дълги** (например до **25** символа)

```
// Проверка за прекалено дълги имена
if (doctor.FirstName.Length > 25 || doctor.LastName.Length > 25)
{
    // TODO Показваме грешка - Прекалено дълго име
    return;
}
```

Допълнителни проверки (6)

- Admins
  - Имейлът винаги е уникален

```
// Проверка за дублиране на имейл
if (dbContext.Administrators.Any(a => a.Email == admin.Email))
{
    // TODO Показваме грешка - Admin с този имейл вече съществува
    return;
}
```

Допълнителни проверки (7)

- **Имената** не са **прекалено дълги** (например до **25** символа)

```
// Проверка за прекалено дълги имена
if (admin.FirstName.Length > 25 || admin.LastName.Length > 25)
{
    // TODO Показваме грешка - Прекалено дълго име
    return;
}
```

Обобщение

SoftUni Foundation

- Използвахме всичко **научено** до момента
- Създадохме **практически проект** с **входна**, **главни** и **модални** **форми**
- Работихме с **Data Transfer Object**
- Имплементирахме отделни **функционалности** спрямо **ролята** на **потребителя**

Въпроси?