

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных систем и технологий

Кафедра Измерительно-вычислительные комплексы

Дисциплина Базы данных

## КУРСОВАЯ РАБОТА

Тема Автоматизированная информационная система центра  
генетических исследований

Выполнил студент \_\_\_\_\_ / А.М. Шиле /  
подпись инициалы, фамилия

Курс 4 Группа ИСТбд-42

Направление 09.03.02 «Информационные системы и технологии»

Руководитель доцент кафедры ИВК, к.т.н., доцент  
должность, учёная степень, учёное звание

Родионов Виктор Викторович  
фамилия, имя, отчество

Дата сдачи:

«\_\_\_» \_\_\_\_\_ 20\_\_ г.

Дата защиты:

«\_\_\_» \_\_\_\_\_ 20\_\_ г.

Оценка: \_\_\_\_\_

Ульяновск  
2025

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет информационных систем и технологий

Кафедра Измерительно-вычислительные комплексы

Дисциплина Базы данных

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

студенту ИСТбд-32  
группа

Шиле А.М.  
фамилия, инициалы

Тема работы Автоматизированная информационная система центра  
генетических исследований

Срок сдачи законченной работы «\_\_\_» \_\_\_\_\_ 20\_\_ г.

Исходные данные к работе методические указания к выполнению курсовой  
(базовое предприятие, характер курсовой работы:

работы и проведению практических занятий для студентов направления  
задание кафедры, инициативная НИР, рекомендуемая литература, материалы практики)

09.03.03 «Информационные системы и технологии» по дисциплине  
«Базы данных» Родионова В.В.

Содержание пояснительной записки список использованных обозначений  
и сокращений, введение, техническое задание, информационное обеспечение  
системы, алгоритмическое обеспечение системы, прикладное программное  
обеспечение системы, руководство пользователя, заключение, список  
использованных источников.

Перечень графического материала \_\_\_\_\_

Руководитель доцент каф. ИВК  
должность

\_\_\_\_\_  
подпись / В.В. Родионов /  
инициалы, фамилия

«\_\_\_» \_\_\_\_\_ 20\_\_ г.

Студент \_\_\_\_\_  
подпись

/ М.А. Шиле /  
инициалы, фамилия

«\_\_\_» \_\_\_\_\_ 20\_\_ г.

## Содержание

Список использованных обозначений и сокращений .....	5
Введение .....	6
1 Техническое задание .....	9
1.1 Общие сведения.....	9
1.2 Назначение и цели создания системы.....	9
1.2.1 Назначение системы .....	9
1.2.2 Цели создания системы .....	9
1.3 Характеристика объекта автоматизации.....	9
1.4 Требования к системе .....	10
1.4.1 Требования к системе в целом.....	10
1.4.1.1 Требования к структуре и функционированию системы.....	10
1.4.1.2 Требования к защите информации он несанкционированного доступа.....	10
1.4.2 Требования к функциям, выполняемым системой .....	10
1.4.3 Требования к видам обеспечения.....	11
1.4.3.1 Требования к техническому обеспечению .....	11
1.4.3.2 Требования к программному обеспечению.....	11
1.5 Состав и содержание работ по созданию системы.....	11
1.6 Порядок контроля и приёмки системы .....	11
1.7 Требования к документированию .....	11
2 Информационное обеспечение системы .....	12
2.1 Концептуальная схема базы данных.....	12
2.1.1 Модель «сущность-связь» .....	12
2.1.2 Сущности и их атрибуты .....	12

		№							
Разраб.						Литера	Лист	Листов	
Пров.	Родионов					У			49
Реценз.									
Н.	Родионов								
Утв.									

Пояснительная  
записка

ИСТбд-42



## Список использованных обозначений и сокращений

1. **ЦГИ** - Центр генетических исследований.

						Лист
						5
		№				

## Введение

Центр генетических исследований представляет собой специализированное научно-практическое учреждение, основной деятельностью которого является проведение комплексных генетических исследований и оказание услуг в области генетической диагностики. Современные ЦГИ играют ключевую роль в развитии персонализированной медицины, позволяя выявлять наследственные заболевания, определять генетические предрасположенности и разрабатывать индивидуальные подходы к лечению и профилактике различных заболеваний. В условиях роста интереса к вопросам здоровья и профилактики такие центры становятся важным звеном не только научной, но и практической медицины.

Основными направлениями деятельности ЦГИ являются молекулярно-генетическая диагностика, цитогенетические исследования, геномное секвенирование, а также медико-генетическое консультирование пациентов и их семей. Главная ценность таких организаций в том, что они соединяют науку и практику: создают базы знаний и при этом реально помогают людям справиться с важными медицинскими проблемами.

В работе ЦГИ участвуют специалисты разных профилей: врачи-генетики, молекулярные биологи, лаборанты, биоинформатики и другие эксперты. Эффективность работы ЦГИ во многом зависит от взаимодействия всех сотрудников: врачи проводят консультации и формируют рекомендации для пациентов, лаборанты выполняют лабораторные анализы и обеспечивают точность экспериментов, а специалисты по обработке данных систематизируют результаты исследований и предоставляют их в удобном для использования виде. Работа центра включает управление информацией о пациентах, биологических образцах, генетических тестах и заключениях. У каждого пациента есть медицинская карта с личными данными, историей болезни и проведёнными исследованиями. Биологические материалы хранятся в специальных условиях, где важно поддерживать температуру и порядок учёта. Даже маленькая ошибка в

маркировке образца может привести к неточности в результатах и неправильным выводам.

Проведение генетических тестов требует точного учета различных типов биологических образцов (кровь, слюна, биопсия и др.), каждый из которых проходит несколько стадий обработки - от регистрации до архивирования или утилизации. Итоги исследований оформляются в виде заключений, где собраны данные и даны индивидуальные рекомендации. Эти документы помогают врачам вырабатывать стратегии профилактики или лечения.

Персонал ЦГИ включает специалистов различных категорий: генетиков, врачей, лаборантов, исследователей и административный персонал. Каждый сотрудник имеет определенную квалификацию, опыт работы и выполняет специфические функции в процессе проведения генетических исследований.

Таким образом, ЦГИ играет ключевую роль в современной медицине и научной сфере. Они не только обеспечивают точные генетические исследования и поддерживают развитие науки, но и оказывают реальную помощь пациентам, способствуя улучшению здоровья и профилактике наследственных заболеваний.

Для описания системы ЦГИ использовались:

1. Сайт «Генокарта» использовался для изучения современных направлений генетических исследований и перечня оказываемых услуг в области генетической диагностики;

2. Сайт Медико-генетического научного центра имени академика Н.П. Бочкова применялся для анализа принципов организации работы и структуры генетического центра;

3. Статья «Профессия: генетик» с ресурса Foxford использовалась для изучения квалификационных требований к специалистам и должностных обязанностей персонала;

4. Руководство по ASP.NET Core MVC применялось для изучения архитектуры MVC, организации контроллеров, работы с представлениями Razor и реализации механизмов валидации данных.

5. Документация по Entity Framework использовалась для изучения

организации связей между сущностями, работы с отношениями "многие-ко-многим", создания миграций базы данных и конфигурации моделей с помощью Fluent API

6. Книга М. Дж. Прайса "C# 10 и .NET 6. Современная кросс-платформенная разработка" использовалась для изучения основ создания веб-приложений с использованием платформы .NET 6, принципов объектно-ориентированного программирования на C# и организации слоя доступа к данным;



# 1 Техническое задание

## 1.1 Общие сведения

Автоматизированная информационная система (далее система) «Центр генетических исследований» разрабатывается на основе заданной предметной области.

## 1.2 Назначение и цели создания системы

### 1.2.1 Назначение системы

Данная информационная система предназначена для автоматизации учета и управления основными процессами ЦГИ. Она позволяет централизованно работать с данными пациентов, биологическими образцами, результатами генетических тестов и медицинскими заключениями.

### 1.2.2 Цели создания системы

1. централизация и упрощение доступа ко всей информации о пациентах, биологических образцах, проводимых генетических тестах, заключениях врачей-генетиков и кадровом составе сотрудников центра;
2. Обеспечение удобного и быстрого поиска, фильтрации и анализа данных по всем сущностям системы;
3. Автоматизация ключевых рабочих процессов, включая регистрацию поступивших образцов, назначение и проведение тестов.

## 1.3 Характеристика объекта автоматизации

Объектом автоматизации является ЦГИ. Центр занимается проведением сложных лабораторных анализов для изучения наследственных заболеваний, индивидуальных особенностей организма и других задач современной медицины.

В работе центра задействованы сотрудники различных специальностей: генетики, врачи, лаборанты, медсестры, исследователи и администраторы. Каждый специалист выполняет определенные задачи в процессе проведения генетических исследований

Основной процесс начинается с получения биологического образца от пациента. Лаборанты и медсестры проводят подготовку образцов для исследований. Генетики и исследователи выполняют лабораторные анализы

с использованием специализированного оборудования. Врачи- генетики интерпретируют полученные результаты и формируют медицинские заключения. Администраторы обеспечивают координацию всех этапов работы.

Таким образом, система должна охватывать полный цикл работы центра: от регистрации пациентов и учета образцов до проведения исследований и формирования заключений специалистами.

## 1.4 Требования к системе

### 1.4.1 Требования к системе в целом

#### 1.4.1.1 Требования к структуре и функционированию системы

Определяется общей постановкой задачи задания на курсовую работу.

#### 1.4.1.2 Требования к защите информации от несанкционированного доступа

В системе предусмотрено два типа пользователей:

1. User имеет доступ только к главной странице сайта. Может просматривать общую информацию о центре и статистику. Любые другие действия запрещены;
2. Admin обладает полными правами на все операции с данными: просмотр, добавление, изменение и удаление записей во всех разделах системы.

Общедоступные данные включают в себя общую информацию о системе, а также информацию о статистике.

#### 1.4.2 Требования к функциям, выполняемым системой

1. Регистрация пациентов, ведение базы биологических образцов с указанием их типа, статуса и условий хранения;
2. создание медицинских заключений по результатам исследований;
3. обслуживание каталога генов и сотрудников, предоставление сводной статистики по деятельности центра.
4. назначение генетических тестов, фиксация методов анализа, результатов и их интерпретации. Установление связи тестов с генами.

### 1.4.3 Требования к видам обеспечения

#### 1.4.3.1 Требования к техническому обеспечению

Рекомендуемая конфигурация технического обеспечения:

1. Материнская плата – MSI B350M Pro-VDH (MS-7A38);
2. Процессор AMD Ryzen 5 5500, 6 ядер, 12 потоков;
3. Видеокарта – NVIDIA GeForce GTX 1070;
4. Оперативная память – 32 ГБ DDR4;
5. SSD-накопитель – AMD R5M512G8 512ГБ.

#### 1.4.3.2 Требования к программному обеспечению

1. Операционная система – Windows 10 корпоративная;
2. СУБД – Microsoft SQL Server Management Studio 19.3;
3. Среда разработки – Visual Studio 2022;
4. Программное обеспечение для создания диаграммы «сущность-связь» - ER-Constructor 2.0.

### 1.5 Состав и содержание работ по созданию системы

Определяется этапами выполнения работы задания на курсовую работу.

### 1.6 Порядок контроля и приёмки системы

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

### 1.7 Требования к документированию

Требуется алгоритмическое обеспечение системы





равное нулю, так как сотрудник может временно не проводить тесты, и максимальное кардинальное число, равное  $N$ . Сущность «Генетический анализ» имеет минимальное и максимальное кардинальные числа, равные единице, так как каждый тест обязательно проводится конкретным сотрудником.

Отношение «участвует/включает» типа многие-ко-многим связывает сущности «Ген» и «Генетический анализ». Один ген может проверяться в нескольких тестах, и один тест может включать несколько генов. Сущность «Ген» имеет минимальное кардинальное число, равное нулю, так как тест может не включать конкретный ген, и максимальное число N. Сущность «Генетический тест» имеет минимальное число, равное единице, так как тест обязательно связан хотя бы с одним геном, и максимальное число N.

Отношение «имеет/входит» типа один-к-одному связывает сущности «Генетический анализ» и «Заключение». Один тест может иметь одно заключение, а заключение относится к одному тесту. Сущность «Генетический тест» имеет минимальное кардинальное число равно 1, также как и максимальное. Сущность «Заключения» имеет минимальное и максимальное кардинальные числа, равные единице.

## 2.2 Внутренняя схема базы данных

### 2.2.1 Анализ концептуальной схемы

Все связи между сущностями предметной области приведены к доменно-ключевой нормальной форме (ДКНФ), что гарантирует отсутствие аномалий модификации. Пользовательские ограничения реализованы через атрибуты валидации и Fluent API конфигурации.

Для связи «один-ко-многим» между «Пациентами» и «Генетическими образцами» использован внешний ключ PatientId с каскадным удалением (DeleteBehavior.Cascade), что обеспечивает автоматическое удаление всех образцов пациента при его удалении.

Связи «один-ко-многим» между «Генетическими образцами» и «Генетическими тестами», а также между «Сотрудниками» и «Генетическими тестами» реализованы через внешние ключи SampleId и EmployeeId соответственно. Для связи с сотрудником установлено поведение DeleteBehavior.SetNull, что позволяет сохранить тесты в системе при удалении сотрудника.

Связь «один-к-одному» между «Генетическими тестами» и «Заключениями» реализована через общий первичный ключ, где TestId в сущности «Заклучения» является как первичным ключом, так и внешним ключом, ссылающимся на тест.

Для связи «многие-ко-многим» между «Генами» и «Генетическими тестами» создана связующая сущность «GeneTestRelations» с составным первичным ключом (GeneId, TestId), что исключает дублирование связей и обеспечивает нормализацию данных.

Все конфигурации реализованы через Fluent API, что обеспечивает четкое разделение логики предметной области и конфигурации базы данных, а также поддерживает ссылочную целостность на уровне СУБД.

## 2.2.2 База данных системы

### 1. Таблица «**Employees**» (рабочие)

```
CREATE TABLE [dbo].[Employees](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [FirstName] [nvarchar](25) NOT NULL,
    [LastName] [nvarchar](25) NOT NULL,
    [MiddleName] [nvarchar](25) NULL,
    [Email] [nvarchar](50) NOT NULL,
    [Phone] [nvarchar](11) NOT NULL,
    [EmployeeType] [nvarchar](50) NOT NULL,
    [Bio] [nvarchar](500) NOT NULL,
    [BirthDate] [datetime2](7) NOT NULL,
    [WorkExperience] [smallint] NOT NULL,
    [HireDate] [datetime2](7) NOT NULL,
    CONSTRAINT [PK_Employees] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

## 2. Таблица «**Patients**» (Пациенты)

```
CREATE TABLE [dbo].[Patients](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [FirstName] [nvarchar](25) NOT NULL,
    [LastName] [nvarchar](25) NOT NULL,
    [MiddleName] [nvarchar](25) NULL,
    [BirthDate] [datetime2](7) NOT NULL,
    [Gender] [nvarchar](50) NOT NULL,
    [Phone] [nvarchar](11) NOT NULL,
    CONSTRAINT [PK_Patients] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

## 3. Таблица «**GeneticSamples**» (образцы анализов пациентов)

```
CREATE TABLE [dbo].[GeneticSamples](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [SampleType] [nvarchar](25) NOT NULL,
    [CollectionDate] [datetime2](7) NOT NULL,
    [StorageLocation] [nvarchar](25) NOT NULL,
    [TemperatureRegime] [nvarchar](25) NOT NULL,
    [Status] [nvarchar](25) NOT NULL,
    [PatientId] [int] NOT NULL,
    CONSTRAINT [PK_GeneticSamples] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[GeneticSamples] WITH CHECK ADD CONSTRAINT
[FK_GeneticSamples_Patients_PatientId] FOREIGN KEY([PatientId])
REFERENCES [dbo].[Patients] ([Id])
ON DELETE CASCADE
GO
```

```
ALTER TABLE [dbo].[GeneticSamples] CHECK CONSTRAINT
[FK_GeneticSamples_Patients_PatientId]
GO
```



#### 4. Таблица «GeneticTests» (тесты пациентов)

```
CREATE TABLE [dbo].[GeneticTests](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [TestName] [nvarchar](75) NOT NULL,
    [ConductDate] [datetime2](7) NOT NULL,
    [AnalysisMethod] [nvarchar](75) NOT NULL,
    [Result] [nvarchar](75) NOT NULL,
    [Interpretation] [nvarchar](200) NOT NULL,
    [SampleId] [int] NOT NULL,
    [EmployeeId] [int] NULL,
    CONSTRAINT [PK_GeneticTests] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[GeneticTests] WITH CHECK ADD CONSTRAINT
[FK_GeneticTests_Employees_EmployeeId] FOREIGN KEY([EmployeeId])
REFERENCES [dbo].[Employees] ([Id])
ON DELETE SET NULL
GO

ALTER TABLE [dbo].[GeneticTests] CHECK CONSTRAINT
[FK_GeneticTests_Employees_EmployeeId]
GO

ALTER TABLE [dbo].[GeneticTests] WITH CHECK ADD CONSTRAINT
[FK_GeneticTests_GeneticSamples_SampleId] FOREIGN KEY([SampleId])
REFERENCES [dbo].[GeneticSamples] ([Id])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[GeneticTests] CHECK CONSTRAINT
[FK_GeneticTests_GeneticSamples_SampleId]
GO
```

## 5. Таблица «**Genes**» (гены)

```
CREATE TABLE [dbo].[Genes](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Name] [nvarchar](25) NOT NULL,
    [Function] [nvarchar](50) NOT NULL,
    [RelatedDiseases] [nvarchar](300) NOT NULL,
    CONSTRAINT [PK_Genes] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

### 6. Таблица «**GeneTestRelations**» (связующая таблица многие ко многим)

```
CREATE TABLE [dbo].[GeneTestRelations]( [GeneId]
[int] NOT NULL,
[TestId] [int] NOT NULL,
CONSTRAINT [PK_GeneTestRelations] PRIMARY KEY CLUSTERED
(
[GeneId] ASC,
[TestId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] GO

ALTER TABLE [dbo].[GeneTestRelations] WITH CHECK ADD CONSTRAINT
[FK_GeneTestRelations_Genes_GeneId] FOREIGN KEY([GeneId]) REFERENCES
[dbo].[Genes] ([Id])
ON DELETE CASCADE GO

ALTER TABLE [dbo].[GeneTestRelations] CHECK CONSTRAINT
[FK_GeneTestRelations_Genes_GeneId]
GO

ALTER TABLE [dbo].[GeneTestRelations] WITH CHECK ADD CONSTRAINT
[FK_GeneTestRelations_GeneticTests_TestId] FOREIGN KEY([TestId]) REFERENCES
[dbo].[GeneticTests] ([Id])
ON DELETE CASCADE GO

ALTER TABLE [dbo].[GeneTestRelations] CHECK CONSTRAINT
[FK_GeneTestRelations_GeneticTests_TestId]
GO
```

## 7. Таблица «Conclusions» (Заключения)

```
CREATE TABLE [dbo].[Conclusions](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [ConclusionDate] [datetime2](7) NOT NULL, [Summary]
    [nvarchar](100) NOT NULL, [Recommendations]
    [nvarchar](500) NOT NULL, [TestId] [int] NOT NULL,
    CONSTRAINT [PK_Conclusions] PRIMARY KEY CLUSTERED (
        [Id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
    [PRIMARY]
) ON [PRIMARY] GO
ALTER TABLE [dbo].[Conclusions] WITH CHECK ADD CONSTRAINT
[FK_Conclusions_GeneticTests_TestId] FOREIGN KEY([TestId]) REFERENCES
[dbo].[GeneticTests] ([Id])
ON DELETE CASCADE GO
ALTER TABLE [dbo].[Conclusions] CHECK CONSTRAINT
[FK_Conclusions_GeneticTests_TestId]
```

## Хранимые процедуры, триггеры и функции:

Процедура «GetEmployeeEfficiencyIndex» возвращает статистику работы сотрудника  
(Уникальных пациентов и генов, всего тестов и их даты, взрослых/детей, мужчин/женщин)

```
CREATE OR ALTER PROCEDURE GetEmployeeEfficiencyIndex
    @EmployeeId INT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @EfficiencyIndex DECIMAL(10,4) = 0;
    DECLARE @TotalTests INT = 0,
            @UniquePatients INT = 0,
            @UniqueGenes INT = 0,
            @WorkExperience INT = 0;

    -- Получаем опыт работы
    SELECT @WorkExperience = WorkExperience
    FROM Employees
    WHERE Id = @EmployeeId;
    IF @WorkExperience IS NULL
    BEGIN
        SELECT CAST(0.0 AS DECIMAL(10,4)) AS EfficiencyIndex;
        RETURN;
    END;

    -- Считаем тесты и пациентов
    SELECT
        @TotalTests = COUNT(gt.Id),
        @UniquePatients = COUNT(DISTINCT gs.PatientId)
    FROM Employees e
    LEFT JOIN GeneticTests gt ON e.Id = gt.EmployeeId
    LEFT JOIN GeneticSamples gs ON gt.SampleId = gs.Id
    WHERE e.Id = @EmployeeId;

    -- Считаем уникальные гены
    SELECT @UniqueGenes = COUNT(DISTINCT gtr.GeneId)
    FROM Employees e
    LEFT JOIN GeneticTests gt ON e.Id = gt.EmployeeId
    LEFT JOIN GeneTestRelations gtr ON gt.Id = gtr.TestId
    WHERE e.Id = @EmployeeId;

    -- Формула для индекса эффективности
    SET @EfficiencyIndex =
        (ISNULL(@TotalTests,0) * 0.5)
        + (ISNULL(@UniquePatients,0) * 1.0)
        + (ISNULL(@UniqueGenes,0) * 0.7)
        + (ISNULL(@WorkExperience,0) * 0.3);

    -- Возвращаем результат
    SELECT @EfficiencyIndex AS EfficiencyIndex;
END;
```

### 3 Прикладное программное обеспечение системы

#### 3.1 Общая характеристика прикладного программного обеспечения

Для разработки автоматизированной системы использовались следующие программные инструменты:

1. Инструментальная среда разработки: Microsoft Visual Studio Community 2022.
2. Язык программирования: C#.
3. СУБД: Microsoft SQL Server Management Studio 19.3.
4. Технология разработки: ASP.NET MVC.
5. Технология доступа к данным: ADO.NET Entity Framework.
6. Моделирование данных выполнено с помощью средства автоматизации ERConstructor 2.0.

Для всех таблиц базы данных реализована возможность просмотра данных в табличном виде, добавления, редактирования, удаления записей. При этом сохраняется логическая и ссылочная целостность.

В программе реализован поиск заключений по названию генетического теста в таблице «Conclusions». Реализованы две операции фильтрации на основе нескольких критериев:

1. Фильтрация заключений по дате заключения.
2. Фильтрация заключения по содержанию текста в название теста.

Также в программе реализован поиск сотрудников по их имени и фамилии.

Все представления созданы и оформлены с применением вспомогательных методов Html.

Реализована валидация ввода данных на уровне моделей с использованием атрибутов проверки. Клиентская же валидация обеспечена подключением библиотек jQuery Validation. Также реализована проверка ModelState перед сохранением данных.





7. HomeController– отображает главную страницу системы с статистикой по количеству сотрудников, пациентов, тестов и образцов для пользователей и администраторов. 25 значимых строк кода.

### 3.3 Особенности реализации и сопровождения

Программное обеспечение построено на основе ASP.NET MVC фреймворка с использованием подхода Code First в Entity Framework, что обеспечило:

1. Гибкое определение модели данных с ручной настройкой конфигураций с помощью Fluent API.
2. Легкое изменение схемы данных через систему миграций (Migrations) с возможностью последовательного применения изменений.
3. Наполнение базы тестовыми данными через метод HasData в конфигурациях Entity Framework, что позволяет предзаполнять таблицы корректными данными в процессе применения миграций

В системе реализовано разделение интерфейсов для администратора и пользователя через различные макеты (Layouts). Основной макет предоставляет полный доступ ко всем разделам системы (сотрудники, пациенты, образцы, гены, тесты и заключения), в то время как пользовательский макет предлагает ограниченный функционал без административных элементов. Переключение между интерфейсами осуществляется через навигационную панель, позволяя пользователям выбирать режим работы.

## 4 Руководство пользователя

### 4.1 Общие сведения

Разработанная система представляет собой специализированное веб-приложение для автоматизации деятельности центра генетических исследований. Она предназначена для комплексного управления данными о пациентах, генетических образцах, лабораторных тестах и научных заключениях в медицинских и исследовательских учреждениях генетического профиля.

Система предоставляет два режима работы: административный и пользовательский. Администраторам доступен полный набор операций для работы с данными: добавление, редактирование, просмотр и удаление записей о сотрудниках, пациентах, генетических образцах, генах, тестах и заключениях с расширенной фильтрацией и сортировкой. Пользователям доступен ограниченный режим просмотра общей информации о центре, включая статистику деятельности и ознакомительные материалы о направлениях исследований.

Для администраторов требуются минимальные знания в области генетических исследований и опыт работы с системами управления базами данных. Для обычных пользователей не требуется специальной подготовки - интерфейс предоставляет понятный доступ к общей информации о центре с статистикой и описанием услуг.



1. При открытии веб-сайта пользователь попадает на главную страницу (рисунок 5.1), которая содержит общую информацию о генетическом исследовательском центре. Пользователи могут ознакомиться с ней и выбрать роль для входа в систему



Генетический центр

Сотрудники

Пациенты

Образцы

Гены

Тесты

Заклучения

UserAdmin

О нашем центре

Мы - ведущий исследовательский центр, специализирующийся на современных генетических технологиях и персонализированной медицине.

Наш центр объединяет лучших специалистов в области генетики, биоинформатики и молекулярной биологии для решения самых сложных задач современной медицины.

Наша статистика

7

Сотрудников

7

Пациентов

8

Образцов

11

Тестов

						Лист
						25
		№				

3. При выборе вкладки «Сотрудники» (рисунок 5.3) администратор может просматривать, фильтровать, изменять, добавлять и удалять сотрудников.

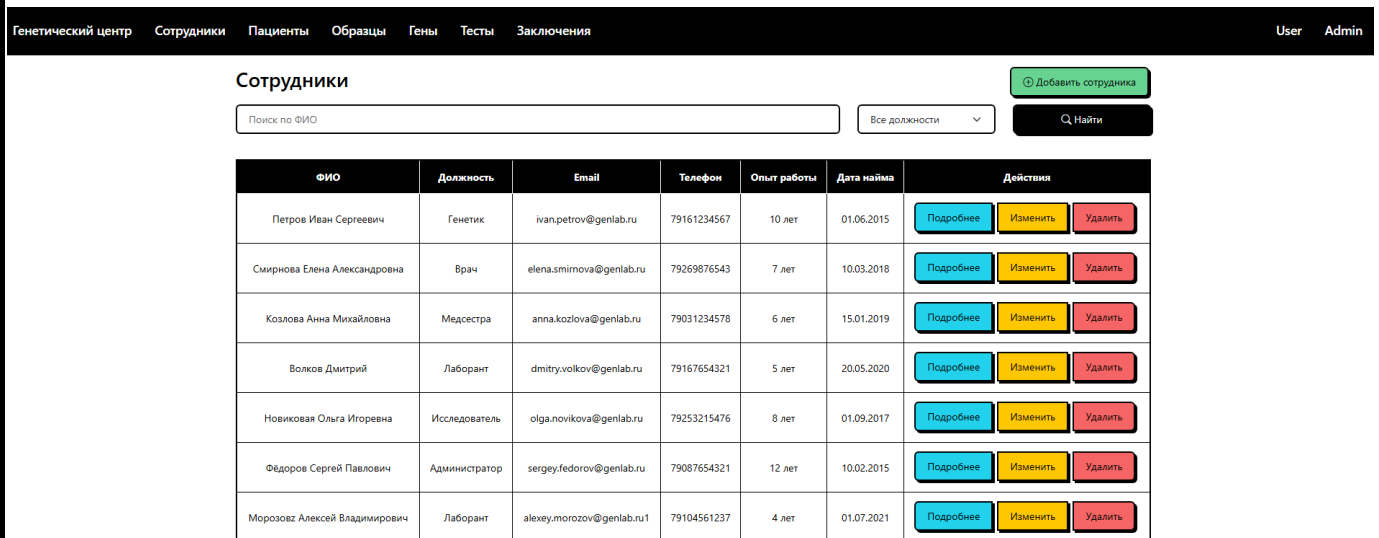



Рисунок 5.3 – Страница «Сотрудники»

4. При нажатии кнопки «Подробнее» открывается страница с полной информацией о сотруднике (рисунок 5.4), содержащая рабочую фотографию и сведения о его профессиональной деятельности в исследовательском центре.

### Подробная информация о сотруднике



Петров Иван Сергеевич

Генетик

Контактная информация

Email: ivan.petrov@genlab.ru

Телефон: 79161234567

Дата рождения: 15.05.1985

Рабочая информация

Дата найма: 01.06.2015

Опыт работы: 10 лет

Уникальных пациентов  
2

Индекс эффективности  
10,20

Биография

Специалист по молекулярной генетике, эксперт в области наследственных заболеваний

Показатель эффективности

10,20

Общий индекс эффективности сотрудника

Рассчитывается на основе количества тестов, пациентов, исследованных генов и опыта работы

Уникальных пациентов: 2

Индекс эффективности: 10,20

Опыт работы: 10 лет

Дата найма: 01.06.2015

← Назад к списку

✕ Редактировать

Рисунок 5.4 – Страница «Подробнее»

5. Нажав на кнопку «Добавить сотрудника» открывается форма (рисунок 5.5) для внесения информации о новом сотруднике.

Добавление нового сотрудника

Фамилия

Имя

Отчество

Email

Телефон

Должность

Дата рождения

Опыт работы (лет)

Генетик

01.01.0001

☐ 0

Биография

Назад

Создать

Рисунок 5.5 – Добавление сотрудника

6. Администратор также может изменять информацию о сотруднике (рисунок 5.6)

Генетический центр
Сотрудники
Пациенты
Образцы
Гены
Тесты
Заклучения

Редактирование сотрудника

Фамилия

Имя

Отчество

Петров

Иван

Сергеевич

Email

Телефон

ivan.petrov@genlab.ru

79161234567

Должность

Дата рождения

Опыт работы (лет)

Генетик

15.05.1985

☐ 10

Биография

Специалист по молекулярной генетике, эксперт в области наследственных заболеваний

Дата найма

2015-06-01

Назад

Сохранить

Рисунок 5.6 – Изменение сотрудника

7. Администратор может удалять информацию о сотруднике при этом всплывает окно с предупреждением о необратимости действия (рисунок 5.7).







## Заключение

В результате выполнения курсовой работы была успешно разработана и реализована информационная система для автоматизации деятельности центра генетических исследований. Созданное веб-приложение обеспечивает комплексное управление всеми основными бизнес-процессами учреждения, включая работу с пациентами, сотрудниками, генетическими образцами, тестами и заключениями.

Разработанная система полностью соответствует утвержденному техническому заданию и удовлетворяет всем предъявленным функциональным требованиям. Особое внимание было уделено обеспечению целостности данных и валидации вводимой информации, что реализовано через систему проверок на уровне моделей и контроллеров. Пользовательский интерфейс имеет удобные инструменты для фильтрации, поиска и просмотра данных, что ускоряет работу сотрудников.

В процессе разработки возникли трудности с организацией отображения и выбора связанных данных при создании и редактировании записей. Проблема была успешно решена с использованием механизма SelectList. Этот подход позволил организовать удобные выпадающие списки с данными.

Единственным недочетом системы можно считать упрощенную систему ролей (только два уровня доступа: пользователь и администратор). В будущем возможно расширение функционала разграничения прав для более гибкого управления доступом различных категорий сотрудников.

## Список использованных источников

1. Генокарта: генетическая энциклопедия: официальный сайт. – URL: <https://www.genokarta.ru/> (дата обращения: 23.07.2025). – Текст: электронный.
2. Медико-генетический научный центр имени академика Н.П. Бочкова: официальный сайт. – Текст: электронный // ФГБНУ «МГНЦ». – URL: <https://www.med-gen.ru/> (дата обращения: 24.07.2025).
3. Профессия: генетик. – Текст: электронный // Foxford: официальный сайт. – URL: <https://media.foxford.ru/articles/geneticist> (дата обращения: 25.07.2025).
4. Прайс, М. Дж. C# 10 и .NET 6. Современная кросс-платформенная разработка: [пер. с англ.] / М. Дж. Прайс. — Москва: Питер, 2023. — 848 с. — ISBN 978-5-4461-2166-8. – Текст: непосредственный.
5. Руководство по ASP.NET Core MVC: официальный сайт. – 2022. – URL: <https://metanit.com/sharp/aspnetmvc/> (дата обращения: 27.07.2025). – Текст: электронный.
6. Entity Framework: официальный сайт. – 2025. – URL: <https://learn.microsoft.com/ru-ru/ef/> (дата обращения: 28.07.2025). – Текст: электронный.



## Приложение А. Исходные тексты контроллеров

### 1. ConclusionController

```
using CenterForGeneticResearch.Data;
using CenterForGeneticResearch.Models.Entities;
using CenterForGeneticResearch.Models.ViewModels;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;

namespace CenterForGeneticResearch.Controllers;

public class ConclusionController : Controller
{
    private readonly ApplicationDbContext _db;

    public ConclusionController(ApplicationDbContext db)
    {
        _db = db;
    }

    public IActionResult Index(string testNameFilter, DateTime? conclusionDateFilter)
    {
        var conclusions = _db.Conclusions
            .Include(c => c.GeneticTest)
            .AsQueryable();

        if (!string.IsNullOrEmpty(testNameFilter))
        {
            conclusions = conclusions.Where(c =>
                c.GeneticTest.TestName.Contains(testNameFilter));
        }

        if (conclusionDateFilter.HasValue)
        {
            conclusions = conclusions.Where(c =>
                c.ConclusionDate.Date == conclusionDateFilter.Value.Date);
        }

        var viewModel = new ConclusionFilterVM
        {
            TestNameFilter = testNameFilter,
            ConclusionDateFilter = conclusionDateFilter,
            Conclusions = conclusions.ToList()
        };

        return View(viewModel);
    }
}
```

```

[HttpPost]
public IActionResult Create(Conclusion model)
{
    if (!ModelState.IsValid)
    {
        ViewBag.Tests = GetTestsWithoutConclusions();
        return View(model);
    }

    _db.Conclusions.Add(model);
    _db.SaveChanges();

    return RedirectToAction("Index");
}

[HttpGet]
public IActionResult Update(int id)
{
    var conclusion = _db.Conclusions
        .Include(c => c.GeneticTest)
        .ThenInclude(t => t.GeneticSample)
        .ThenInclude(s => s.Patient)
        .FirstOrDefault(c => c.Id == id);

    if (conclusion == null)
    {
        return NotFound();
    }

    ViewBag.TestInfo = $"Тест: {conclusion.GeneticTest.TestName} (Пациент:
{conclusion.GeneticTest.GeneticSample.Patient.LastName} " +
    $"{conclusion.GeneticTest.GeneticSample.Patient.FirstName})";

    return View(conclusion);
}

[HttpPost]
public IActionResult Update(C Conclusion model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

    _db.Conclusions.Update(model);
    _db.SaveChanges();

    return RedirectToAction("Index");
}

```

		№		

```

[HttpPost]
public IActionResult Delete(int id)
{
    var conclusion = _db.Conclusions.FirstOrDefault(h => h.Id == id);
    if (conclusion == null)
    {
        return NotFound();
    }

    _db.Conclusions.Remove(conclusion);
    _db.SaveChanges();

    return Ok();
}

private SelectList GetTestsWithoutConclusions()
{
    var tests = _db.GeneticTests
        .Include(t => t.GeneticSample)
        .ThenInclude(s => s.Patient)
        .Where(t => !_db.Conclusions.Any(c => c.TestId == t.Id))
        .Select(t => new SelectListItem
        {
            Value = t.Id.ToString(),
            Text = $"{t.TestName} (Пациент: {t.GeneticSample.Patient.LastName}
{t.GeneticSample.Patient.FirstName})"
        })
        .ToList();

    return new SelectList(tests, "Value", "Text");
}
}

```

## 2. EmployeeController

```

using CenterForGeneticResearch.Data;
using CenterForGeneticResearch.Helpers;
using CenterForGeneticResearch.Models.Entities;
using CenterForGeneticResearch.Models.Enums;
using CenterForGeneticResearch.Models.ViewModels;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.Data.SqlClient;
using Microsoft.EntityFrameworkCore;

namespace CenterForGeneticResearch.Controllers;

public class EmployeeController : Controller
{
    private readonly ApplicationDbContext _db;

    public EmployeeController(ApplicationDbContext db)
    {
        _db = db;
    }
}

```

		№		

```

public IActionResult Index(string nameFilter, string typeFilter)
{
    var employees = _db.Employees.AsQueryable();

    if (!string.IsNullOrEmpty(nameFilter))
    {
        employees = employees.Where(e =>
            e.FirstName.Contains(nameFilter) ||
            e.LastName.Contains(nameFilter) ||
            (e.MiddleName != null && e.MiddleName.Contains(nameFilter)));
    }

    if (!string.IsNullOrEmpty(typeFilter))
    {
        employees = employees.Where(e => e.EmployeeType.ToString() == typeFilter);
    }

    var viewModel = new EmployeeFilterVM
    {
        NameFilter = nameFilter,
        TypeFilter = typeFilter,
        Employees = employees.ToList()
    };

    return View(viewModel);
}

public IActionResult Details(int id)
{
    var employee = _db.Employees.FirstOrDefault(e => e.Id == id);
    if (employee == null) return NotFound();

    var stats = GetEmployeeEfficiencyIndex (id);
    ViewBag.EmployeeStats = stats;
    ViewBag.UniquePatientsCount = stats?.UniquePatientsCount ?? 0;

    return View(employee);
}

[HttpGet]
public IActionResult Create()
{
    ViewBag.EmployeeTypes = GetEmployeeTypesSelectList();

    return View(new Employee());
}

[HttpPost]
public IActionResult Create(Employee model)
{
    if (!ModelState.IsValid)
    {
        ViewBag.EmployeeTypes = GetEmployeeTypesSelectList();
        return View(model);
    }
}

```

		№		





```

var viewModel = new GeneFilterVM
{
    NameFilter = nameFilter,
    Genes = genes.ToList()
};

return View(viewModel);
}

[HttpGet]
public IActionResult Create()
{
    return View(new Gene());
}

[HttpPost]
public IActionResult Create(Gene model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }
    _db.Genes.Add(model);
    _db.SaveChanges();

    return RedirectToAction("Index");
}

[HttpGet]
public IActionResult Update(int id)
{
    var gene = _db.Genes.Find(id);
    if (gene == null)
    {
        return NotFound();
    }
    return View(gene);
}

[HttpPost]
public IActionResult Update(Gene model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

    _db.Genes.Update(model);
    _db.SaveChanges();

    return RedirectToAction("Index");
}

[HttpPost]
public IActionResult Delete(int id)
{
    var gene = _db.Genes.FirstOrDefault(g => g.Id == id);
    if (gene == null)
    {
        return NotFound();
    }
    _db.Genes.Remove(gene);
    _db.SaveChanges();

    return Ok();
}
}

```

#### 4. GeneticSampleController

```
using CenterForGeneticResearch.Data;
using CenterForGeneticResearch.Helpers;
using CenterForGeneticResearch.Models.Entities;
using CenterForGeneticResearch.Models.Enums;
using CenterForGeneticResearch.Models.ViewModels;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;

namespace CenterForGeneticResearch.Controllers;

public class GeneticSampleController : Controller
{
    private readonly ApplicationDbContext _db;

    public GeneticSampleController(ApplicationDbContext db)
    {
        _db = db;
    }

    public IActionResult Index(string ownerNameFilter, string sampleTypeFilter, string
statusFilter)
    {
        var samples = _db.GeneticSamples
            .Include(gs => gs.Patient)
            .AsQueryable();

        if (!string.IsNullOrEmpty(ownerNameFilter))
        {
            samples = samples.Where(gs =>
                gs.Patient.FirstName.Contains(ownerNameFilter) ||
                gs.Patient.LastName.Contains(ownerNameFilter) ||
                (gs.Patient.MiddleName != null &&
gs.Patient.MiddleName.Contains(ownerNameFilter)));
        }

        if (!string.IsNullOrEmpty(sampleTypeFilter))
        {
            samples = samples.Where(gs => gs.SampleType.ToString() == sampleTypeFilter);
        }
        if (!string.IsNullOrEmpty(statusFilter))
        {
            samples = samples.Where(gs => gs.Status.ToString() == statusFilter);
        }
        var viewModel = new GeneticSampleFilterVM
        {
            OwnerNameFilter = ownerNameFilter,
            SampleTypeFilter = sampleTypeFilter,
            StatusFilter = statusFilter,
            GeneticSamples = samples.ToList()
        };

        return View(viewModel);
    }
}
```



```
[HttpGet]
public IActionResult Create()
{
    ViewBag.Patients = GetPatientsSelectList();
    ViewBag.SampleTypes = GetSampleTypesSelectList();
    ViewBag.Statuses = GetStatusesSelectList();

    return View();
}

[HttpPost]
public IActionResult Create(GeneticSample model)
{
    if (!ModelState.IsValid)
    {
        ViewBag.Patients = GetPatientsSelectList(model.PatientId);
        ViewBag.SampleTypes = GetSampleTypesSelectList();
        ViewBag.Statuses = GetStatusesSelectList();

        return View(model);
    }

    model.CollectionDate = DateTime.Now;
    _db.GeneticSamples.Add(model);
    _db.SaveChanges();

    return RedirectToAction("Index");
}

[HttpGet]
public IActionResult Update(int id)
{
    var sample = _db.GeneticSamples.Find(id);
    if (sample == null)
    {
        return NotFound();
    }

    ViewBag.Patients = GetPatientsSelectList(sample.PatientId);
    ViewBag.SampleTypes = GetSampleTypesSelectList(sample.SampleType);
    ViewBag.Statuses = GetStatusesSelectList(sample.Status);

    return View(sample);
}

[HttpPost]
public IActionResult Update(GeneticSample model)
{
    if (!ModelState.IsValid)
    {
        ViewBag.Patients = GetPatientsSelectList(model.PatientId);
        ViewBag.SampleTypes = GetSampleTypesSelectList();
        ViewBag.Statuses = GetStatusesSelectList();

        return View(model);
    }

    _db.GeneticSamples.Update(model);
    _db.SaveChanges();

    return RedirectToAction("Index");
}
```

```

[HttpPost]
public IActionResult Delete(int id)
{
    var sample = _db.GeneticSamples.FirstOrDefault(h => h.Id == id);
    if (sample == null)
    {
        return NotFound();
    }

    _db.GeneticSamples.Remove(sample);
    _db.SaveChanges();

    return Ok();
}

private SelectList GetPatientsSelectList(object selectedValue = null)
{
    return new SelectList(_db.Patients
        .Select(p => new {
            Id = p.Id,
            FullName = $"{p.LastName} {p.FirstName} {p.MiddleName}"
        }),
        "Id", "FullName", selectedValue);
}

private SelectList GetSampleTypesSelectList(object selectedValue = null)
{
    return new SelectList(Enum.GetValues<SampleType>()
        .Select(t => new SelectListItem
        {
            Value = t.ToString(),
            Text = t.GetDisplayName(),
            Selected = selectedValue != null && t.ToString() == selectedValue.ToString()
        }), "Value", "Text");
}

private SelectList GetStatusesSelectList(object selectedValue = null)
{
    return new SelectList(Enum.GetValues<SampleStatus>()
        .Select(s => new SelectListItem
        {
            Value = s.ToString(),
            Text = s.GetDisplayName(),
            Selected = selectedValue != null && s.ToString() == selectedValue.ToString()
        }), "Value", "Text");
}
}

```

## 5. GeneticTestController

```

using CenterForGeneticResearch.Data;
using CenterForGeneticResearch.Helpers;
using CenterForGeneticResearch.Models.Entities;
using CenterForGeneticResearch.Models.ViewModels;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;

namespace CenterForGeneticResearch.Controllers;

public class GeneticTestController : Controller

```

```

{
    private readonly ApplicationDbContextContext _db;

    public GeneticTestController(ApplicationDbContext db)
    {
        _db = db;
    }

    public IActionResult Index(string patientFilter, string testNameFilter)
    {
        var tests = _db.GeneticTests
            .Include(t => t.GeneticSample)
            .ThenInclude(s => s.Patient)
            .Include(t => t.GeneRelations)
            .ThenInclude(gr => gr.Gene)
            .AsQueryable();

        if (!string.IsNullOrEmpty(patientFilter))
        {
            tests = tests.Where(t =>
                t.GeneticSample.Patient.LastName.Contains(patientFilter) ||
                t.GeneticSample.Patient.FirstName.Contains(patientFilter) ||
                (t.GeneticSample.Patient.MiddleName != null &&
                 t.GeneticSample.Patient.MiddleName.Contains(patientFilter)));
        }

        if (!string.IsNullOrEmpty(testNameFilter))
        {
            tests = tests.Where(t => t.TestName.Contains(testNameFilter));
        }

        var viewModel = new GeneticTestFilterVM
        {
            PatientFilter = patientFilter,
            TestNameFilter = testNameFilter,
            GeneticTests = tests.Select(t => new GeneticTestWithGenes
            {
                Test = t,
                GeneNames = t.GeneRelations.Select(gr => gr.Gene.Name).ToList()
            }).ToList()
        };

        return View(viewModel);
    }

    public IActionResult Details(int id)
    {
        var test = _db.GeneticTests
            .Include(t => t.GeneticSample)
            .ThenInclude(s => s.Patient)
            .Include(t => t.Employee)
            .Include(t => t.Conclusion)
            .Include(t => t.GeneRelations)
            .ThenInclude(gr => gr.Gene)
            .FirstOrDefault(t => t.Id == id);

        if (test == null)
        {
            return NotFound();
        }

        var viewModel = new GeneticTestDetailsVM
        {
            Test = test,
            GeneNames = test.GeneRelations?.Select(gr => gr.Gene.Name).ToList() ?? new
            List<string>(),
            EmployeeInfo = test.Employee != null ?
                $"{test.Employee.LastName} {test.Employee.FirstName} {test.Employee.MiddleName}
                ({test.Employee.EmployeeType.GetDisplayName()})" : "Не назначен",
        };
    }
}

```





```

        _db.SaveChanges();

        return RedirectToAction("Index");
    }

    [HttpPost]
    public IActionResult Delete(int id)
    {
        var test = _db.GeneticTests.FirstOrDefault(g => g.Id == id);
        if (test == null)
        {
            return NotFound();
        }

        _db.GeneticTests.Remove(test);
        _db.SaveChanges();

        return Ok();
    }

    private SelectList GetEmployeesSelectList(object selectedValue = null)
    {
        var employees = _db.Employees
            .Select(e => new SelectListItem
            {
                Value = e.Id.ToString(),
                Text = $"{e.LastName} {e.FirstName} ({e.EmployeeType.GetDisplayName()})",
                Selected = selectedValue != null && e.Id.ToString() == selectedValue.ToString()
            }).ToList();

        return new SelectList(employees, "Value", "Text");
    }

    private SelectList GetGeneticSamplesSelectList(object selectedValue = null)
    {
        var samples = _db.GeneticSamples
            .Include(s => s.Patient)
            .Select(s => new SelectListItem
            {
                Value = s.Id.ToString(),
                Text = $"ID: {s.Id} | {s.SampleType.GetDisplayName()} | Пациент: {s.Patient.LastName} {s.Patient.FirstName}",
                Selected = selectedValue != null && s.Id.ToString() == selectedValue.ToString()
            }).ToList();

        return new SelectList(samples, "Value", "Text");
    }
}

```

## 6. PatientController

```
using CenterForGeneticResearch.Data;
using CenterForGeneticResearch.Helpers;
using CenterForGeneticResearch.Models.Entities;
using CenterForGeneticResearch.Models.Enums;
using CenterForGeneticResearch.Models.ViewModels;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
namespace CenterForGeneticResearch.Controllers;
```

```

public class PatientController : Controller
{
    private readonly ApplicationDbContext _db;

    public PatientController(ApplicationDbContext db)
    {
        _db = db;
    }

    public IActionResult Index(string nameFilter, string genderFilter)
    {
        var patients = _db.Patients.AsQueryable();

        if (!string.IsNullOrEmpty(nameFilter))
        {
            patients = patients.Where(p =>
                p.FirstName.Contains(nameFilter) ||
                p.LastName.Contains(nameFilter) ||
                (p.MiddleName != null && p.MiddleName.Contains(nameFilter)));
        }

        if (!string.IsNullOrEmpty(genderFilter))
        {
            patients = patients.Where(p => p.Gender.ToString() == genderFilter);
        }

        var viewModel = new PatientFilterVM
        {
            NameFilter = nameFilter,
            GenderFilter = genderFilter,
            Patients = patients.ToList()
        };

        return View(viewModel);
    }

    [HttpGet]
    public IActionResult Create()
    {
        ViewBag.Genders = GetGendersSelectList();

        return View(new Patient());
    }

    [HttpPost]
    public IActionResult Create(Patient model)
    {
        if (!ModelState.IsValid)
        {
            ViewBag.Genders = GetGendersSelectList();
            return View(model);
        }

        if (_db.Patients.Any(p => p.Phone == model.Phone))
        {
            ModelState.AddModelError("Phone", "Пациент с таким телефоном уже существует");
            ViewBag.Genders = GetGendersSelectList();
            return View(model);
        }

        _db.Patients.Add(model);
        _db.SaveChanges();

        return RedirectToAction("Index");
    }
}

```





