МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет <u>информационных систем и технологий</u>
Кафедра <u>Измерительно-вычислительные комплексы</u>
Дисциплина <i>Базы данных</i>

КУРСОВАЯ РАБОТА

Тема <i>Автоматиз</i>	ированная информа	ционная системо	и центра	
генетических исслед	ований			
Выполнил студент	подпись	/	<u>Шиле</u> пы, фамилия	/
Курс <u>4</u>		·	- -	
Направление <u>09.03.02</u>	2 «Информационные	системы и техно	ологии»	
Руководитель <u>доцени</u>		•		
Родионов Виктор Вик	должность, учёная стю кторович	епень, ученое звание		
	фамилия,имя, о	тчество		
		Дата сдачи:		
		«»	20	г.
		Дата защит	Ы:	
		«»	20	г.
		Оценка:		

Ульяновск 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет <i>информационных сис</i>	стем и технологий
Кафедра <u>Измерительно-вычисл</u>	ительные комплексы
Дисциплина <u><i>Базы данных</i></u>	
ЗАДАНИЕ	НА КУРСОВУЮ РАБОТУ
студенту <u>ИСТб∂-32</u>	Шиле А.М.
еруппа	фамилия, инициалы
Тема работы <u>Автоматизиров</u>	анная информационная система центра
генетических исследований	
Срок сдачи законченной работы	«»20г.
	одические указания к выполнению курсовой вое предприятие, характер курсовой работы:
<u>работы и проведению практичес</u> задание кафедры, инициативная НИР, рег	ких занятий для студентов направления комендуемая литература, материалы практики)
09.03.03 «Информационные сист	емы и технологии» по дисциплине
«Базы данных» Родионова В.В.	
Содержание пояснительной зап	иски <u>список использованных обозначений</u>
и сокращений, введение, техниче	ское задание, информационное обеспечение
<u>системы, алгоритмическое обес</u>	печение системы, прикладное программное
обеспечение системы, руководск	пво пользователя, заключение, список
использованных источников.	
	ia
Руководитель <u>доцент каф. ИВК</u> должность	
	« » 20 г.
	
Студент	/ <u>М.А. Шиле</u> инициалы, фамилия
110011400	« » 20 г.

Содержание

	эзначении и сокращении	
	здания системы	
1.2.1 Назначение систе	МЫ	9
1.2.2 Цели создания си	стемы	9
1.3 Характеристика объек	ста автоматизации	9
1.4 Требования к системе		10
	геме в целом	
	структуре и функционированию	
1.4.1.2 Требования к	защите информации он несанкци	ионированного
	кциям, выполняемым системой	
	ам обеспечения	
-	техническому обеспечению	
-	программному обеспечению	
-	работ по созданию системы	
-	риёмки системы	
•	тированию	
•	тированию	
	а базы данных	
	ъ-связь»	
	рибуты	
2.1.2 Сущности и их ат	рноуты	12
Nº		
Разраб. Родионов Родионов		Литера Лист Листов У 49
Реценз.	I Іояснительная	
Н. Родионов Утв.	записка	ИСТ6∂-42

2.1.3 Связи между сущностями	13
2.2 Внутренняя схема базы данных	14
2.2.1 Анализ концептуальной схемы	14
2.2.2 База данных системы	15
3 Прикладное программное обеспечение системы	20
3.1 Общая характеристика прикладного программного обеспечения	20
3.2 Структура и состав прикладного программного обеспечения	21
3.3 Особенности реализации и сопровождения	23
4 Руководство пользователя	24
4.1 Общие сведения	24
4.2 Порядок и особенности работы	25
4.3 Исключительные ситуации	30
Заключение	31
Список использованных источников	32
Приложение А. Исходные тексты контроллеров	33

NΘ

Спи	сок испо	ользо	ван	іных обозначений и сокращений	
1	I. ЦГИ - Ц	ентр геі	нетич	ческих исследований.	
					Лисі
	Nº			1	5

Введение

Nº

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 молекулярногенетическая 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 справляться 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

			Лист
			6

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

			Tiucm
		·	7
	Nº		,

организации связей между сущностями, работы с отношениями "мн многим", создания миграций базы данных и конфигурации моделей с п	
Fluent API	
6. Книга М. Дж. Прайса "C# 10 и .NET 6. Современная кросс-платфо	рменная
разработка" использовалась для изучения основ создания веб-прилож	жений с
использованием платформы .NET 6, принципов объектно-ориентиро	ванного
программирования на С# и организации слоя доступа к данным;	
	Лисг
Nº N	8

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

			Лист	ı
			Q	i i
	Nº			ì

с использованием специализированного оборудования. Врачи- генетики интерпре тируют полученные результаты и формируют медицинские заключения. Админис траторы 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 он 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.

			Лист
			10
	Nº		10

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

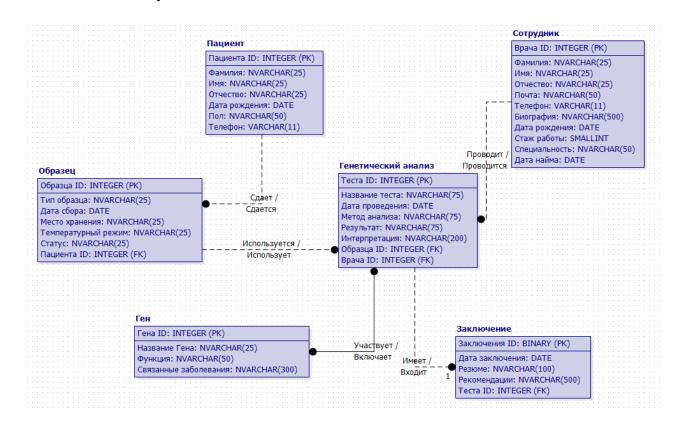
Требуется алгоритмическое обеспечение системы

			Лист
			11
	Nº		11

2 Информационное обеспечение системы

2.1 Концептуальная схема базы данных

2.1.1 Модель «сущность-связь»



2.1.2 Сущности и их атрибуты

Сущность «Пациент» содержит личную информацию о пациентах центра. Является источником биологических образцов.

Сущность "Сотрудник" содержит данные о сотрудниках центра. Атрибут "Специальность" определяет должность и специализацию сотрудника (генетик, врач, лаборант, исследователь, администратор). Атрибут "Стаж работы" измеряется в полных годах.

Сущность "Генетические анализ" содержит данные о проводимых генетических исследованиях. Атрибут "Метод анализа" описывает применяемую методику исследования. Атрибут "Интерпретация" содержит развернутое объяснение полученных результатов.

Сущность "Ген" содержит справочную информацию о исследуемых генах. Атрибут "Функция" описывает назначение гена в организме. Атрибут "Связанные

			Лист	l
			12	
	Nº		12	l

заболевания" перечисляет заболевания, ассоциированные с нарушениями в данном гене.

Сущность "Заключение" содержит итоговые отчеты по проведенным исследованиям. Атрибут "Резюме" содержит краткое резюме результатов теста.

Сущность "Образец" содержит информацию о биологических материалах пациентов. Атрибут "Тип образца" определяет вид биологического материала (кровь, слюна, волосы, биопсия, моча). Атрибут "Статус" отражает текущее состояние образца в процессе исследования (зарегистрирован, в обработке, готов к анализу, в анализе, анализ завершен, архивирован, испорчен, уничтожен).

2.1.3 Связи между сущностями

Отношение «сдает/сдается» типа один-ко-многим связывает сущности «Пациент» и «Образец». Один пациент может сдать множество образцов, но каждый образец принадлежит только одному пациенту. Сущность «Пациент» имеет минимальное кардинальное число, равное нулю, т.к. пациент может не иметь сданных образцов, и максимальное кардинальное число, равное N. Сущность «Образец» имеет минимальное и максимальное кардинальные числа, равные единице, т.к. каждый образец должен быть привязан к конкретному пациенту.

Отношение «используется/использует» типа один-ко-многим связывает сущности «Образец» и «Генетический анализ». Один образец может быть использован в нескольких тестах, но каждый тест выполняется на одном образце. Сущность «Образец» имеет минимальное кардинальное число, равное нулю, т.к. образец может не быть использован в тестах, и максимальное кардинальное число, равное N. Сущность «Генетические анализы» имеет минимальное и максимальное кардинальные числа, равные единице, т.к. каждый тест должен быть выполнен на конкретном образце.

Отношение «проводит/проводится» типа один-ко-многим связывает сущности «Сотрудник» и «Генетический анализ». Один сотрудник может проводить много генетических тестов, но каждый тест выполняется только одним сотрудником. Сущность «Сотрудники» имеет минимальное кардинальное число,

			Лист	
			13	
	NIO		13	l

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 доменноключевой 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.

			Лист
			1/1
	N IO		14

Связи «один-ко-многим» между «Генетическими образцами» и «Генетическими тестами», а также между «Сотрудниками» и «Генетическими тестами» реализованы через внешние ключи SampleId и EmployeeId соответствен -но. 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 «Заключения» 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

Nº

```
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
```

١	TUC	'	'
	1	_	,

```
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
```

Nº

Лист

16

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

Лист 1**7**

Nº

```
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]
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
```

			Лист	l
			18	l
	Nº		10	ı

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

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

ALTER TABLE [dbo].[Conclusions] CHECK CONSTRAINT

[FK_Conclusions_GeneticTests_TestId]

END:

Nº

```
@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
    BFGTN
        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;
```

Лист

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: С#.
 - 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.

			Лист
			20
	Nº		20

В системе реализованы значения по умолчанию для временных меток в четырех таблицах: HireDate в «Employee», CollectionDate в «GeneticSample», ConductDate в «GeneticTest» и ConclusionDate в «Conclusion» автоматически устанавливаются текущей датой при создании записей. Также Созданы уникальные индексы для телефонных номеров пациентов и email/телефонов сотрудников.

Во всем проекте использовались значимые имена файлов, классов и объектов.

В представлениях были использованы вспомогательные методы Html для ввода данных.

Код содержит только необходимые комментарии для пояснения сложной логики.

База данных содержит:

- 1. 7 таблиц, одна из них нужна для связи «многие-ко-многим»
- 2. 31 атрибут, без учета внешних ключей и ключей-счетчиков
- 3. Три типа связи: один-ко-многим, один-ко-одному, многие-ко-многим
- 4. Не менее 7 записей в каждой из рабочих таблиц и не менее 3 записей в справочниках.
- 5. Использовались хранимая процедура «GetEmployeeEfficiencyIndex» для подсчета уникальных клиетов, проведенных тестов, исследованных генов

В программе отсутствует аутентификация авторизация. Вместо этого предложено две точки входа в программу: Admin и User.

3.2 Структура и состав прикладного программного обеспечения

Обозреватель решений проекта:

- 1. Data папка с контекстом базы данных, миграциями и конфигурациями для сущностей.
- 2. Helpers папка с методами, которые облегчают получение имени у enum и его фотограции.

			Лист	1
			21	i
	N/O		<i>L</i> 1	

- 3. Data папка с контекстом базы данных, миграциями и конфигурациями для сущностей.
- 4. Helpers папка с методами, которые облегчают получение имени у enum и его фотограции.
 - 5. Controllers папка со всеми контроллерами приложения;
 - 6. Models папка с сущностями и моделями для представлений.
 - 7. Views папка с представлениями;
- 8. wwwroot для статических файлов: Bootstrap, CSS, JavaScript, изображения и иконки сайта.

Контроллеры:

- 1. ConclusionController управляет операциями с заключениями: отображением списка с фильтрацией по названию теста и дате, созданием, редактированием и удалением заключений. 92 значимых строк кода.
- 2. EmployeeController отвечает за управление сотрудниками: отображение списка с фильтрацией по ФИО и должности, создание, редактирование, просмотр деталей и удаление записей. 112 значимых строк кода.
- 3. GeneController управляет генами: отображение списка с фильтрацией по названию, создание, редактирование и удаление генетических записей. 63 значимых строк кода.
- 4. PatientController отвечает за управление пациентами: отображение списка, редактирование и удаление записей пациентов. 103 значимых строк кода.
- 5. GeneticSampleController— контролирует генетические образцы: отображение с фильтрацией по владельцу, типу образца и статусу, создание, редактирование и удаление. 121 значимых строк кода.
- 6. GeneticTestController— управляет генетическими тестами: отображение списка с фильтрацией по пациенту и названию теста, детальный просмотр, создание, редактирование с привязкой генов и удаление. 175 значимых строк кода

					Лист		
					22		
	NIO				22		

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.

				1
			Лист	ı
			23	l
	Nº		23	ı

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 сотрудни ках, 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.

			Лист
			24
	Nº		24

4.2 Порядок и особенности работы

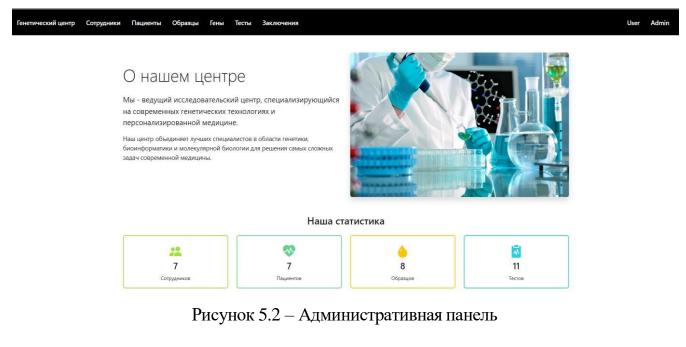
Nº

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



Рисунок 5.1 – Главная страница

2. При выборе роли «Admin» пользователь получает доступ к административному интерфейсу системы (рисунок 5.2), в котором становится доступна панель с разделами для управлений данными



Лист

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

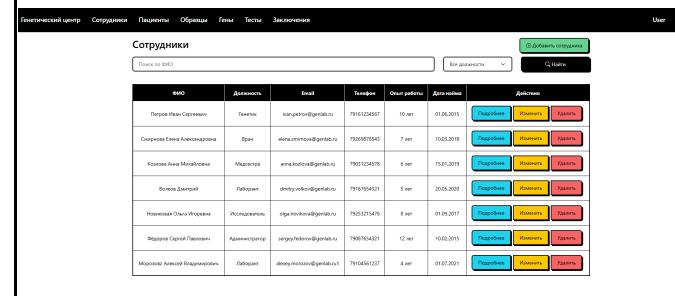


Рисунок 5.3 – Страница «Сотрудники»

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

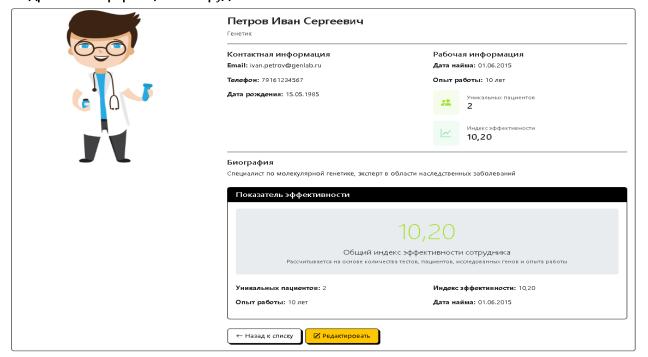


Рисунок 5.4 – Страница «Подробнее»

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

			Лист
			26
	Nº		20

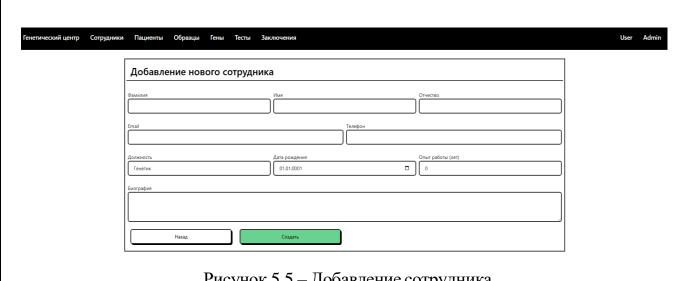


Рисунок 5.5 – Добавление сотрудника

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

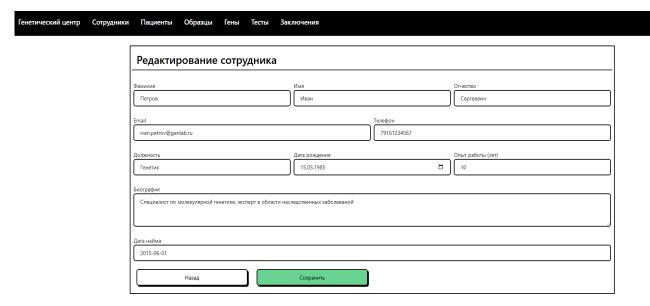


Рисунок 5.6 – Изменение сотрудника

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

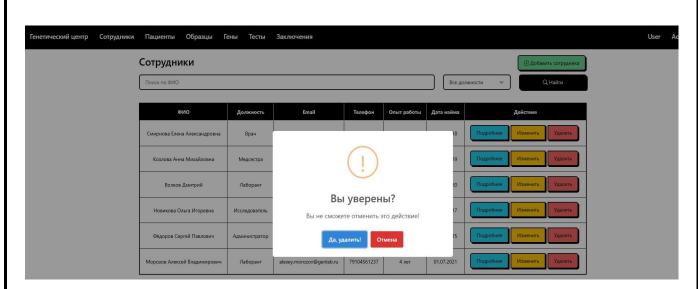


Рисунок 5.7 – Удаление сотрудника

8. Администратор может найти нужного сотрудника по его ФИО (рисунок 5.8)



Рисунок 5.8 – Поиск сотрудника

9. Выбрав нужную профессию, можно найти рабочих путем фильтрации по профессии (рисунок 5.9)

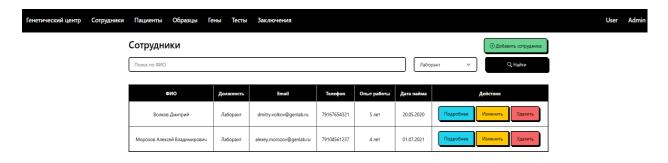


Рисунок 5.9 – Фильтрация по профессии

10. Во вкладке «Тесты» пользователь может нажать на кнопку «Подробнее». В результате чего выведется полная информация (рисунок 5.10) о тесте, данных пациента, результате теста и его заключения.

					Лист		
					28		
	Nō				20		

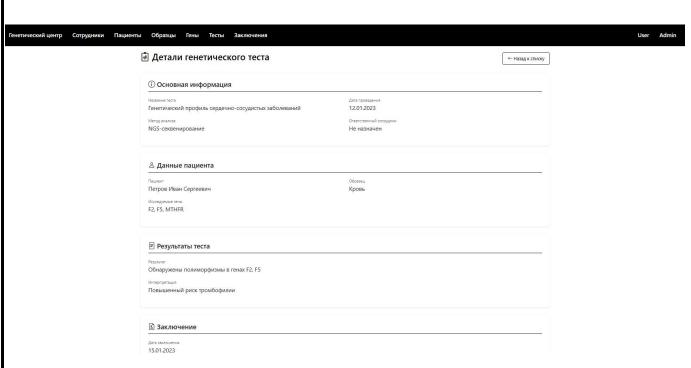


Рисунок 5.10 – Информация о тесте

11. В любой момент Администратор может переключиться на роль «User», нажав в панели на соответствующую роль, и он вернется на главную страницу генетического центра (рисунок 5.11).

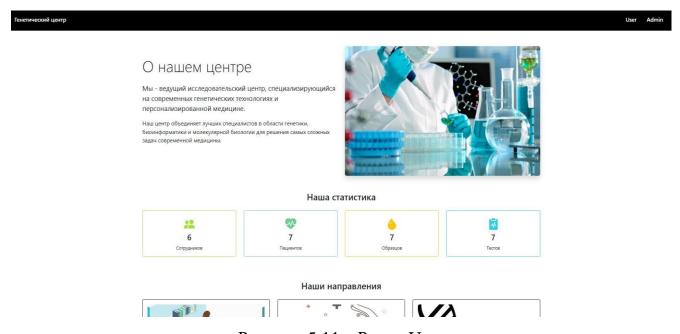


Рисунок 5.11 – Роль «User»

Несмотря на ограниченный доступ к административным данным (тестам, сотрудникам), пользователь «User» сохраняет возможность просмотра общей статистики генетического центра на главной странице.

			Лист
			29
	Nº		2)

4.3 Исключительные ситуации

- 1. Ошибка нарушения уникальности контактных данных возникает при попытке сохранения дублирующихся email или телефона сотрудника в базе данных. Приложение выполняет проверку уникальности этих полей на уровне бизнес-логики, предотвращая дублирование записей, и уведомляет пользователя о конфликте. Для решения необходимо ввести уникальные значения.
- 2. Ошибка возникает при попытке сохранить запись, не заполнив поля, обязательные для ввода. Приложение проверяет наличие значений и выводит сообщение рядом с каждым пустым полем. Для решения необходимо заполнить все обязательные поля.
- 3. Ошибка возникает при попытке указать дату сбора образца, которая превышает текущую дату. Приложение проверяет валидность даты и выводит сообщение об ошибке. Для решения необходимо ввести корректную дату.
- 4. Ошибка возникает при попытке указать значение опыта работы меньше 0 или больше 50 лет. Приложение проверяет соответствие значения заданному диапазону и выводит сообщение об ошибке. Для решения необходимо ввести корректное значение.

			Лист	
			30	
	Nº		50	

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.

		Лист
		31
NIO		\mathcal{I}_{1}

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. С# 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.c om/ru-ru/ef/ (дата обращения: 28.07.2025). Текст: электронный.

					Лист	
					32	
		V10			32	

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 = $"Tect: {conclusion.GeneticTest.TestName} (Пациент:
{conclusion.GeneticTest.GeneticSample.Patient.LastName} " +
     $"{conclusion.GeneticTest.GeneticSample.Patient.FirstName})";
  return View(conclusion);
}
[HttpPost]
public IActionResult Update(Conclusion model)
  if (!ModelState.IsValid)
  {
     return View(model);
  }
  _db.Conclusions.Update(model);
  _db.SaveChanges();
  return RedirectToAction("Index");
}
```

Nº

```
[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.lsNullOrEmpty(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);
  }
```

```
if (_db.Employees.Any(e => e.Email == model.Email))
     ModelState.AddModelError("Email", "Сотрудник с таким email уже существует");
     ViewBag.EmployeeTypes = GetEmployeeTypesSelectList();
     return View(model);
  }
  if (_db.Employees.Any(e => e.Phone == model.Phone))
     ModelState.AddModelError("Phone", "Сотрудник с таким телефоном уже существует");
     ViewBag.EmployeeTypes = GetEmployeeTypesSelectList();
     return View(model);
  }
   model.HireDate = DateTime.Now;
  _db.Employees.Add(model);
  _db.SaveChanges();
  return RedirectToAction("Index");
}
[HttpGet]
public IActionResult Update(int id)
  var employee = _db.Employees.Find(id);
  if (employee == null)
     return NotFound();
  }
  ViewBag.EmployeeTypes = GetEmployeeTypesSelectList();
  return View(employee);
}
[HttpPost]
public IActionResult Update(Employee model)
  if (!ModelState.IsValid)
  {
     ViewBag.EmployeeTypes = GetEmployeeTypesSelectList();
     return View(model);
  }
  if (_db.Employees.Any(e => e.Email == model.Email && e.Id != model.Id))
  {
     ModelState.AddModelError("Email", "Сотрудник с таким email уже существует");
     ViewBag.EmployeeTypes = GetEmployeeTypesSelectList();
     return View(model);
  }
  if (_db.Employees.Any(e => e.Phone == model.Phone && e.Id != model.Id))
  {
     ModelState.AddModelError("Phone", "Сотрудник с таким телефоном уже существует");
     ViewBag.EmployeeTypes = GetEmployeeTypesSelectList();
     return View(model);
  _db.Employees.Update(model);
```

```
_db.SaveChanges();
     return RedirectToAction("Index");
   [HttpPost]
   public IActionResult Delete(int id)
     var employee = _db.Employees.FirstOrDefault(h => h.ld == id);
     if (employée == null)
     {
        return NotFound();
     }
     _db.Employees.Remove(employee);
     _db.SaveChanges();
     return Ok();
   private SelectList GetEmployeeTypesSelectList()
     return new SelectList(Enum.GetValues<EmployeeType>()
        .Select(e => new SelectListItem
        {
           Value = e.ToString(),
        Text = e.GetDisplayName()
}), "Value", "Text");
  \underline{private} \ EmployeeStatsResult \ GetEmployeeEfficiencyIndex \ (int \ employeeId)
     try
     {
        return _db.Database
           .SqlQueryRaw<EmployeeStatsResult>("EXEC GetEmployeeEfficiencyIndex
@EmployeeId",
              new SqlParameter("@EmployeeId", employeeId))
           .AsEnumerable()
           .FirstOrDefault() ?? new EmployeeStatsResult();
     catch
     {
        return new EmployeeStatsResult();
     }
  }
}
3. GeneController
using CenterForGeneticResearch.Data;
using CenterForGeneticResearch.Models.Entities;
using CenterForGeneticResearch.Models.ViewModels;
using Microsoft.AspNetCore.Mvc;
namespace CenterForGeneticResearch.Controllers;
public class GeneController: Controller
{
   private readonly ApplicationDbContext _db;
  public GeneController(ApplicationDbContext db)
  {
     _{db} = db;
  }
  public IActionResult Index(string? nameFilter)
  {
     var genes = _db.Genes.AsQueryable();
     if (!string.IsNullOrEmpty(nameFilter))
     {
        genes = genes.Where(g => g.Name.Contains(nameFilter));
     }
```

```
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();
  }
}
                                                                                              Лисп
```

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. Entity Framework Core;
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.lsNullOrEmpty(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.lsNullOrEmpty(sampleTypeFilter))
        samples = samples.Where(gs => gs.SampleType.ToString() == sampleTypeFilter);
     if (!string.lsNullOrEmpty(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
```

NO

```
private readonly ApplicationDbContext _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.lsNullOrEmpty(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.lsNullOrEmpty(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",
```

```
PatientFullName = $"{test.GeneticSample.Patient.LastName}
{test.GeneticSample.Patient.FirstName} {test.GeneticSample.Patient.MiddleName}",
     SampleInfo = $"{test.GeneticSample.SampleType.GetDisplayName()}",
     Conclusion = test.Conclusion
  };
  return View(viewModel);
}
[HttpGet]
public IActionResult Create()
   ViewBag.Employees = GetEmployeesSelectList();
  ViewBag.GeneticSamples = GetGeneticSamplesSelectList();
  ViewBag.Genes = new SelectList(_db.Genes.ToList(), "Id", "Name");
  return View(new GeneticTest());
}
[HttpPost]
public IActionResult Create(GeneticTest model, List<int> SelectedGenes)
   if (!ModelState.IsValid)
  {
     ViewBag.Employees = GetEmployeesSelectList(model.EmployeeId);
     ViewBag.GeneticSamples = GetGeneticSamplesSelectList(model.SampleId);
     ViewBag.Genes = new SelectList(_db.Genes.ToList(), "Id", "Name");
     return View(model);
  }
  model.ConductDate = DateTime.Now;
   _db.GeneticTests.Add(model);
   _db.SaveChanges();
  if (SelectedGenes != null && SelectedGenes.Any())
     foreach (var geneld in SelectedGenes)
     {
         _db.GeneTestRelations.Add(new GeneTestRelation
        {
           TestId = model.Id,
           Geneld = geneld
     _db.SaveChanges();
  return RedirectToAction("Index");
}
[HttpGet]
public IActionResult Update(int id)
  var test = _db.GeneticTests
     .Include(t => t.GeneRelations)
     .FirstOrDefault(t => t.Id == id);
   if (test == null)
  {
     return NotFound();
```

```
ViewBag.Employees = GetEmployeesSelectList(test.EmployeeId);
  ViewBag.GeneticSamples = GetGeneticSamplesSelectList(test.SampleId);
   ViewBag.Genes = new SelectList(_db.Genes.ToList(), "Id", "Name");
  var selectedGenes = test.GeneRelations.Select(g => g.GeneId).ToList();
  var model = new GeneticTestUpdateVM
  {
     Test = test,
     SelectedGenes = selectedGenes
  };
  return View(model);
}
[HttpPost]
public IActionResult Update(GeneticTestUpdateVM model)
  if (!ModelState.IsValid)
     ViewBag.Employees = GetEmployeesSelectList(model.Test.EmployeeId);
     ViewBag.GeneticSamples = GetGeneticSamplesSelectList(model.Test.SampleId);
     ViewBag.Genes = new SelectList(_db.Genes.ToList(), "Id", "Name");
     return View(model);
  }
   var existingTest =_db.GeneticTests
      .Include(t => t.GeneRelations)
      .FirstOrDefault(t => t.Id == model.Test.Id);
  if (existingTest == null)
  {
     return NotFound();
  }
  existingTest.TestName = model.Test.TestName;
  existingTest.AnalysisMethod = model.Test.AnalysisMethod;
  existingTest.SampleId = model.Test.SampleId;
  existingTest.EmployeeId = model.Test.EmployeeId;
  existingTest.Result = model.Test.Result;
  existingTest.Interpretation = model.Test.Interpretation;
  var currentGeneIds = existingTest.GeneRelations.Select(g => g.GeneId).ToList();
  var selectedGeneIds = model.SelectedGenes ?? new List<int>();
  foreach (var relation in existingTest.GeneRelations.ToList())
     if (!selectedGeneIds.Contains(relation.GeneId))
     {
        _db.GeneTestRelations.Remove(relation);
  }
  foreach (var geneld in selectedGenelds)
     if (!currentGenelds.Contains(geneld))
     {
        _db.GeneTestRelations.Add(new GeneTestRelation
           TestId = existingTest.Id,
           Geneld = geneld
        });
     }
  }
```

Лисп

```
_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. Entity Framework Core;
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.lsNullOrEmpty(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");
  }
```

```
[HttpGet]
   public IActionResult Update(int id)
     var patient = _db.Patients.Find(id);
     if (patient == null)
        return NotFound();
     }
     ViewBag.Genders = GetGendersSelectList(patient.Gender);
     return View(patient);
  }
   [HttpPost]
  public IActionResult Update(Patient model)
     if (!ModelState.IsValid)
     {
        ViewBag.Genders = GetGendersSelectList();
        return View(model);
     }
     if (_db.Patients.Any(p => p.Phone == model.Phone && p.Id != model.Id))
        ModelState.AddModelError("Phone", "Пациент с таким телефоном уже существует");
        ViewBag.Genders = GetGendersSelectList();
        return View(model);
     _db.Patients.Update(model);
     _db.SaveChanges();
     return RedirectToAction("Index");
  }
   [HttpPost]
  public IActionResult Delete(int id)
     var patient = _db.Patients.FirstOrDefault(h => h.ld == id);
     if (patient == null)
        return NotFound();
     }
     _db.Patients.Remove(patient);
     _db.SaveChanges();
     return Ok();
  }
   private SelectList GetGendersSelectList(object selectedValue = null)
     return new SelectList(Enum.GetValues<Gender>()
        .Select(g => new SelectListItem
        {
           Value = g.ToString(),
           Text = g.GetDisplayName(),
           Selected = selectedValue != null && g.ToString() == selectedValue.ToString()
        }), "Value", "Text");
  }
}
```

7. HomeController

```
using CenterForGeneticResearch.Data;
using CenterForGeneticResearch.Models.ViewModels;
using Microsoft.AspNetCore.Mvc;
namespace CenterForGeneticResearch.Controllers;
public class HomeController: Controller
  private readonly ApplicationDbContext _db;
  public HomeController(ApplicationDbContext db)
     _{db} = db;
   public IActionResult Index()
     var model = new HomeViewModel
     {
        TotalEmployees = _db.Employees.Count(),
        TotalPatients = _db.Patients.Count(),
        TotalTests = _db.GeneticTests.Count(),
        TotalSamples = _db.GeneticSamples.Count()
     };
     return View(model);
  }
   public IActionResult UserIndex()
     var model = new HomeViewModel
        TotalEmployees = _db.Employees.Count(),
        TotalPatients = _db.Patients.Count(),
        TotalTests = _db.GeneticTests.Count(),
        TotalSamples = _db.GeneticSamples.Count()
     };
     return View(model);
  }
}
```