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 общества Красного Креста»*

Выполнила студентка _____ */ Н.М. Бакунькина /*

подпись *инициалы, фамилия*

Курс *2* Группа *ИСТбд-21*

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

2021

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

студентке *ИСТбд-21* *Бакунькина Н.М.*
группа *фамилия, инициалы*

Тема работы «*Автоматизированная информационная система общества Красного Креста»*

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 материла _____

_____

Руководитель *доцент каф. ИВК* _____ */ В.В. Родионов /*
должность *подпись* *инициалы, фамилия*

«___» _____ 20___ г.

Студентка _____ */ Н.М.Бакунькина /*
подпись *инициалы, фамилия*

«___» _____ 20___ г

Содержание

Введение

Международный комитет Красного Креста (МККК) – некоммерческая независимая организация, которая учреждена на основе добровольных имущественных взносов граждан, юридических лиц разных стран, государств-участниц Женевской конвенции. На собранные средства пожертвований комитет реализует гуманитарные проекты, направленные на защиту и помощь участникам и жертвам катастроф, стихийных бедствий, вооруженных конфликтов, независимо от принадлежности к той или иной стороне конфликта, национальности, вероисповедания или социального положения пострадавших. Реализуя свои программы МККК выступает с законотворческой инициативой по развитию гуманитарного права, а также контролирует исполнение сторонами конфликта норм международного гуманитарного законодательства. В случае его неисполнения обращается за содействием в международные общественные и национальные правительственные организации.

Принципы МККК: гуманность, беспристрастность, нейтральность, независимость, добровольность, единство, универсальность.

Деятельность Международного комитета Красного Креста направлена на:

- оказание помощи людям, пострадавшим в результате локальных и глобальных военных столкновений, военнослужащим (независимо от стороны конфликта), попавшим в плен, гражданскому населению, находящемуся в зоне вооруженного конфликта и др.;

- оказание гуманитарной помощи (в том числе в рамках спасательных операций) населению любого региона, ставшему жертвой катастроф различного характера (природных, антропогенных, техногенных);

- оказание безвозмездной медицинской помощи незащищенным и нуждающимся социальным слоям населениях различных государств. Обеспечение этой категории населения продовольствием, водой, предметами первой необходимости. [3]

МККК основан в 1863 году швейцарским бизнесменом и писателем Анри Дюнаном, который использовал помещение местной церкви в качестве госпиталя и призвал местное население оказывать первую помощь всем раненным военным. Позже Дюнан описал эти события в ставшей популярной книге «Воспоминание о битве при Сольферино». А через год, в 1864 году, он собрал первую международную Конференцию МККК, которая состоялась в Женеве.

В течение последующих 150 лет общество Международного Комитета Красного Креста сохраняет статус нейтрального посредника, который обеспечивает правовую защиту и поддержку жертвам военных конфликтов и насилия внутри стран, контролирует соблюдение правового режима и условий содержания военнопленных, помогает тысячам людей, нуждающимся в помощи по всему миру. [1]

В наше время у Красного Креста появились новые задачи, не связанные с последствиями военных действий. Обществом реализуется комплекс программ по формированию и развитию национальных проектов здравоохранения, прежде всего направленных на недопущение распространения и профилактику особо опасных инфекций, таких как СПИД, чума, холера, сибирская язва, а также «coronavirus disease 2019». Также реализуются гуманитарные проекты по соблюдению и ужесточению мер безопасности населения и специальные просветительские программы.

МККК не является межправительственной организацией, но ее статус и признание отражено в значимых международных договорах и нормах международного права. Международный статус комитета закреплен в Женевской конвенции. Предоставленные МККК полномочия и иммунитеты (такие как освобождение от налогов и таможенных сборов, неприкосновенность сотрудников и имущества Общества, освобождение от судебного преследования) обеспечивают безопасность деятельности самой организации.

В обществе Красного Креста запрещены ущемления или привилегии по национальному, расовому, религиозному, политическому признаку. С целью утверждения и поддержания этих принципов общество Красного Креста никогда

не признает чью-либо сторону конфликта и не участвует в спорах политического или иного характера.

Международное движение Красного Креста, вызванное рвением содействовать в помощи всем пострадавшим на поле боя без привилегий или преимущества, пытается при любом случае как на международном, так и на мировом уровне устранять и облегчать страдания человека. Движение отстаивает жизнь и здоровье людей и содействует обеспечению уважения к личности человека, приобретению взаимопонимания, дружбы, партнёрства и долговечного мира среди народов. [4]

Целью аналитического обзора является выяснение принципов оказания помощи жертвам различных конфликтов подразделениями Международного Комитета Красного Креста.

В статье Зябкина А. И. «К вопросу о правовой природе Международного Комитета Красного Креста» описывается деятельность Международного Комитета Красного Креста на протяжении всего его существования, с момента создания и до наших дней. Автором изучена деятельность Международного Комитета Красного Креста, начиная с непосредственного оказания помощи пострадавшим, раненным, обеспечения их необходимыми продуктами и медикаментами, и заканчивая законотворческой деятельностью организации, теоретическими разработками документов, имеющих международное значение, а также принятие мер по ограничению распространения оружия, причиняющего излишние страдания, и запрещению бесчеловечных методов ведения войн.

По вопросу основополагающих принципов Комитета Красного Креста были рассмотрены: электронный ресурс «Международный комитет Красного Креста» и общественно-политическое издание «Основополагающие принципы Красного Креста». В электронном ресурсе рассматриваются, основные понятия, цели и задачи Комитета Красного Креста. В тексте общественно-политического издания описаны основные положения комитета, к которым должен придерживаться каждый член Красного Креста.

Результаты литературного обзора показали, что помощь пострадавшим – не единственная цель Красного Креста. Оказывая поддержку, он служит не менее важной задаче: отстоять во время войны идею солидарности людей и уважения человеческого достоинства, когда реальные или мнимые нужды войны отодвигают моральные ценности на второй план. За долгие годы работы Комитет Красного Креста накопил уникальный опыт работы и стал поистине крупнейшей в мире гуманитарной организацией. Тесное сотрудничество между региональными организациями позволяет быстро и эффективно реагировать на проблемные ситуации, возникающие по всему миру.

Также использовались электронные ресурсы для реализации программного и информационного обеспечения системы. В электронном ресурсе «Майкрософт» описывается модель MVC ASP.NET Core с контроллерами и представлениями. Рассматриваются основы создания веб-приложения ASP.NET MVC с помощью Visual Studio. В электронном ресурсе «Уроки по C# и платформе .NET Framework» рассказывается, что собой представляет платформа ASP.NET. Описываются базовые части инфраструктуры ASP.NET: модель веб-страниц, конфигурация приложений и управление состоянием. Также показаны основные возможности LINQ, описывается как запрашивать данные из базы, вставлять, модифицировать и удалять данные с помощью Entity Framework.

1 Техническое задание

1 Общие сведения

Автоматизированная информационная система, далее система «Общество красного креста».

2 Назначение и цели создания системы

2.1 Назначение системы

Удовлетворение информационных потребностей и управление функционированием подразделений «Общества Красного Креста». Система содержит персональные данные и предназначена только для служебного пользования.

2.2 Цели создания системы

1. Целью информационной системы является обеспечение подразделений Красного Креста необходимыми для текущей деятельности документами, информационными массивами и оказанием информационных услуг. Автоматизация процессов обеспечения выполнение задач структурных подразделений Красного Креста.
2. Целью информационной системы является сбор, обработка, хранение данных о кадровом составе, мерах безопасности, о источниках финансирования и закупках Красного Креста для удобства использования данных, а также оптимизации принятия управленческих решений.

3 Характеристика объекта автоматизации

Объектом автоматизации является российское отделение международного комитета Красного Креста. Основным направлением деятельности отделения комитета Красного Креста является оказание старшим и младшим медицинским персоналом помощи жертвам стихийным бедствий и вооруженных конфликтов с целью защиты жизни и здоровья людей. Так же обеспечение и соблюдения прав жертв вооруженных конфликтов, содействие в развитии международного гуманитарного права. Помощь гражданскому населению, розыск пропавших без вести.

Объектом автоматизации являются процессы по управлению сотрудниками комитетом Красного Креста и привлечённым персоналом, управление процессами закупок материальных ценностей, их распределение, финансовый контроль и учёт, а также контроль эффективности выполнения указанных процессов. Данные процессы включают в себя планирования структуры общества, его штатного расписания, кадровую, финансово-экономическую политику, обеспечение мер безопасности, произведение расчета заработной платы, оперативного учета движения кадров, ведения административного документооборота по персоналу и учету труда, (обучение, повышение квалификации) работников.

Следующим направлением автоматизации является накопление персональных данных беженцев и пациентов, определение категорий, нуждающихся в помощи. Ведение архивов без ограничения сроков давности о датах командировок, о дате назначения приёмов пациентам и нуждающимся лицам, побывавшим в чрезвычайных ситуациях. Присутствует вовлечение в волонтёрскую деятельность и публикуется открытая часть информации о повседневной работе Красного Креста.

4 Требования к системе

4.1 Требования к системе в целом

4.1.1 Требования к структуре и функционированию системы

Определяется общей постановкой задачи задания на курсовую работу

4.1.2 Требования к защите информации от несанкционированного доступа

Администратор имеет неограниченный доступ ко всему сайту в том числе и к данным базы данных. Для него требуется авторизация в системе, для защиты информации от посторонних. Под администратором может заходить любой сотрудник, для получения информации по базам данных.

Пользователь имеет общедоступный доступ к сайту, но не имеет доступа к разделу «Базы данных», поскольку в нём содержится информация ограниченного доступа, доступная только сотрудникам.

4.2 Требования к функциям, выполняемым системой

1. Сбор и регистрация информационных ресурсов о кадровых сотрудниках, старшем и младшем медицинском персонале, привлекаемых специалистах, на временной основе, волонтёрах, нуждающихся в помощи лицах, материальных ценностях, о медицинских препаратах и о оборудовании, и о изделии медицинского назначения, финансировании.
2. Предоставление информационных ресурсов пользователям: службе безопасности, кадровому подразделению, финансово-экономическому подразделению, уполномоченным медицинским сотрудникам.
3. Хранение информационных ресурсов: ведение справочника штатных сотрудников, общества Красного Креста, в том числе медицинского персонала по специальностям, привлекаемых специалистов, волонтеров.
4. Обработка информационных ресурсов: учет квалификации медицинского персонала и других штатных сотрудников общества Красного Креста, приобретаемых и реализуемых материальных ценностей, затрат на реализуемые проекты.
5. Вычисление фонда оплаты труда, финансовых затрат (потребностей) на реализацию гуманитарных программ.

4.3 Требования к видам обеспечения

4.3.1 Требования к техническому обеспечению

Рекомендуемая конфигурации технического обеспечения с указанием производителя того или иного компонента, названия модели и её кратких характеристик.

Материнская плата PK, Sleepy_PK, версия V1.09

Процессор AMD, Ryzen 3 3200U with Radeon Vega Mobile Gfx, 2600 МГц, ядер: 2, логических процессоров: 4

Оперативная память Kingston, 8GB Value RAM, (KVR26N19S8/8)

Жёсткий диск HGST, HFM256GDJTNG-8310A, размер 238,47 ГБ (256 052 966 400 байт)

Видеокарта AMD, Radeon(TM) Vega 3 Graphics, (2 147 483 648) байт

Звуковая карта Realtek Semiconductor Corp., (Realtek(R) Audio), версия 6.00.8971.0001

Монитор Acer, Aspire A315-42G, на базе x64

Привод отсутствует

Принтер отсутствует

4.3.2 Требования к программному обеспечению

Версия операционной системы Майкрософт Windows 10 (x64). СУБД и инструментальные средства Microsoft SQL Server Management Studio 2012, Visual Studio 2019, а также программное обеспечение ERConstructor 2.0, использованное при создании модели «сущность-связь».

5 Состав и содержание работ по созданию системы

Определяется этапами выполнения работы задания на курсовую работу.

6 Порядок контроля и приёмки системы

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

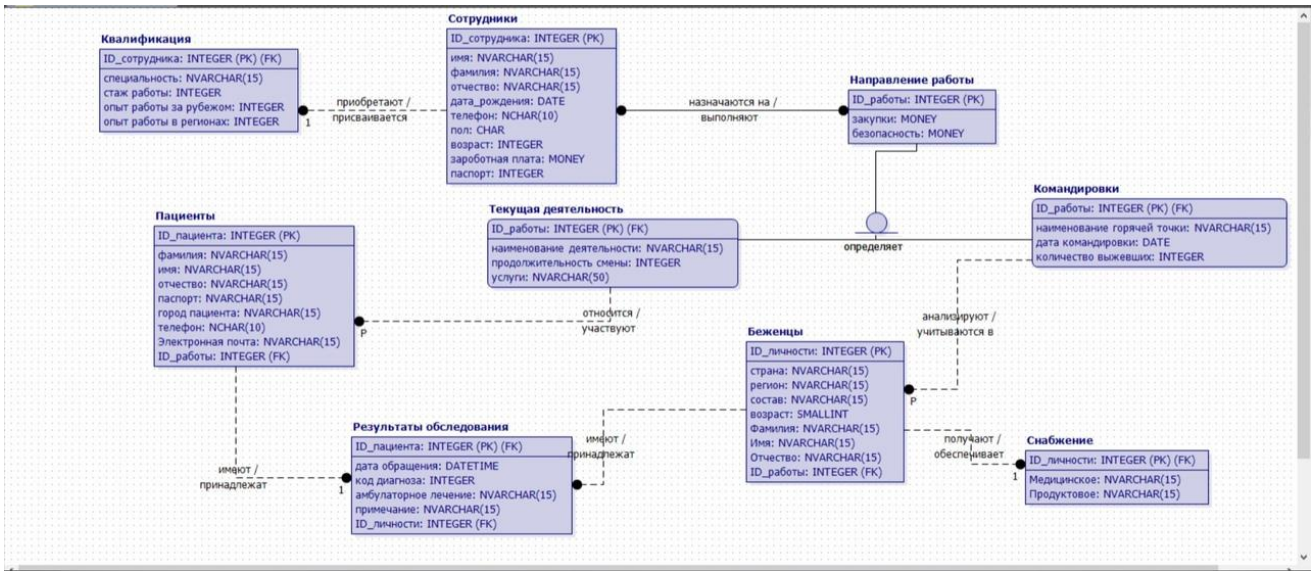
7 Требования к документированию

Для «уровня 4» необязательны структурные компоненты: «Список использованных обозначений и сокращений» и «Заключение», а также «Анализ концептуальной схемы». Изменение названий структурных компонентов не допускается.

2 Информационное обеспечение системы

2.1 Концептуальная схема базы данных

2.1.1 Модель «сущность-связь»



«Рисунок 1.2 – Модель “Сущность-связь”»

2.1.2 Сущности и их атрибуты

В Сущности «Сотрудники» описываются личные данные сотрудников, работающих в Красном кресте.

В Сущности «Квалификация» описывается уровень профессиональной подготовленности соискателя на должность.

Сущность «Направление работы» распределяет вид деятельности сотрудников в иерархию. В каждом виде деятельности присутствуют далее перечисленные атрибуты. Атрибут «Закупки» отвечает за выполнение административно-распорядительных функций в сфере закупок. Атрибут «Безопасность» отвечает за выполнение административно-распорядительных функций в сфере безопасности.

В Сущности «Текущая деятельность» описывается повседневная региональная деятельность отделения «Красного креста». Атрибут «Услуги» описывает виды волонтёрской деятельности Красного креста.

В Сущности «Командировки» описываются командировки, в которых отправляют сотрудников для спасения и оказания гуманитарной помочи беженцам и всем нуждающимся.

В Сущности «Беженцы» описываются личные данные беженцев, которые покинули страну своего постоянного проживания в силу чрезвычайных обстоятельств. Атрибут «Состав» обозначает семейный состав беженцев, а именно количество родственных связей.

В Сущности «Снабжение» описываются виды снабжения необходимые для обеспечения материальных потребностей беженцев.

В Сущности «Пациенты» описываются личные данные пациента, которые необходимы для прохождения обследования или выполнения медицинской услуги.

В Сущности «Результаты обследования» описываются результаты обследования как пациентов, так и беженцев.

2.1.3 Связи между сущностями

Между сущностями «Сотрудники» и «Квалификация» используется тип связи один-к-одному, поскольку у одного сотрудника присутствует только одна квалификация и специальность. Квалификация может быть только у одного сотрудника, так как у всех разный уровень профессиональной подготовки, также сотрудник может быть принят на работу только на одну специальность.

Между сущностями «Сотрудники» и «Направление работы» используется тип связи многие-ко-многим, поскольку группа сотрудников распределяется в разные даты по разным направлениям работы. В разные направления работы распределяют группы сотрудников по разным датам.

Между сущностями «Направление работы», «Текущая деятельность» и «Командировки» используется тип связи иерархия. «Направление работы» распределяет вид деятельности группы сотрудников. В «Текущую деятельность» назначают сотрудников. «Командировки» распределяют сотрудников в горячие точки.

Между сущностями «Текущая деятельность» и «Пациенты» используется тип связи один-к-одному-или-многим, поскольку одна текущая деятельность (организация) обрабатывает данные сразу нескольких пациентов с множеством различных запросов, также в эту же организацию, может обратится один пациент. Один или несколько пациентов могут нуждаться в нескольких услугах, при этом они обращаются в организацию и предоставляют данные: Ф.И.О., серию и номер паспорта, контактные данные для связи, которые корректирует текущая деятельность.

Между сущностями «Пациенты» и «Результаты обследования» используется тип связи один-к-одному, поскольку одному пациенту выдаётся конкретно его результат обследования. Результат обследования предоставляется одному (конкретному) пациенту, по его личному запросу.

Между сущностями «Беженцы» и «Снабжение» используется тип связи один-к-одному, поскольку на каждого беженца устанавлен норматив материального обеспечения. Норматив снабжения устанавливается на одного беженца.

Между сущностями «Командировки» и «Беженцы» используется тип связи один-к-одному-или-многим, поскольку в каждой командировке сотрудники оказывают помощь либо группе беженцев, либо одному. Помощь беженцам сотрудники могут оказать только в одной командировке.

Между сущностями «Беженцы» и «Результаты обследования» используется тип связи один-ко-многим, поскольку один беженец может иметь несколько результатов обследования. Результаты обследования предоставляются только одному (конкретному) беженцу.

2.2 Внутренняя схема базы данных

2.2.2 База данных системы

Описание структуры таблиц базы данных системы на языке Transact-SQL.

Таблица «Беженцы»

```
USE [Красный Крест]
GO

/****** Object:  Table [dbo].[Беженцы]    Script Date: 31.05.2021 16:58:39 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Беженцы](
	[страна] [nvarchar](40) NULL,
	[регион] [nvarchar](40) NULL,
	[состав] [nvarchar](40) NULL,
	[возраст] [smallint] NULL,
	[ID_личности] [int] NOT NULL,
	[ID_сотрудника] [int] NOT NULL,
	[Фамилия] [nvarchar](40) NULL,
	[Имя] [nvarchar](40) NULL,
	[Отчество] [nvarchar](40) NULL,
 CONSTRAINT [PK_Беженцы] PRIMARY KEY CLUSTERED
(
	[ID_личности] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Беженцы] ADD  CONSTRAINT [DF_Беженцы_ID_личности]  DEFAULT (NULL) FOR
[ID_личности]
GO

ALTER TABLE [dbo].[Беженцы]  WITH CHECK ADD  CONSTRAINT [Беженцы_Командировки_FK_Rule] FOR-
EIGN KEY([ID_сотрудника])
REFERENCES [dbo].[Командировки] ([ID_работы])
GO

ALTER TABLE [dbo].[Беженцы] CHECK CONSTRAINT [Беженцы_Командировки_FK_Rule]
GO
```

Таблица «Квалификация»

```
USE [Красный Крест]
GO

/****** Object:  Table [dbo].[Квалификация]    Script Date: 31.05.2021 16:59:02 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Квалификация](
	[специальность] [nvarchar](40) NULL,
	[стаж работы] [int] NULL,
	[опыт работы за рубежом] [int] NULL,
	[опыт работы в регионах] [int] NULL,
	[ID_сотрудника] [int] NOT NULL,
```

```
 CONSTRAINT [PK_Квалификация] PRIMARY KEY CLUSTERED
(
	[ID_сотрудника] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

ALTER	TABLE	[dbo].[Квалификация]	WITH	CHECK	ADD	CONSTRAINT
[Квалификация_Сотрудники_FK_Rule] FOREIGN KEY([ID_сотрудника])
REFERENCES [dbo].[Сотрудники] ([ID_сотрудника])
GO

ALTER TABLE [dbo].[Квалификация] CHECK CONSTRAINT [Квалификация_Сотрудники_FK_Rule]
GO
```

Таблица «Командировки»

```
USE [Красный Крест]
GO

/****** Object:  Table [dbo].[Командировки]    Script Date: 31.05.2021 16:59:21 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Командировки](
	[наименование горячей точки] [nvarchar](40) NULL,
	[дата командировки] [datetime] NULL,
	[количество выжевших] [int] NULL,
	[ID_работы] [int] NOT NULL,
 CONSTRAINT [PK_Командировки] PRIMARY KEY CLUSTERED
(
	[ID_работы] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Командировки]  WITH CHECK ADD  CONSTRAINT [FK_Командировки_Направление ра-
боты1] FOREIGN KEY([ID_работы])
REFERENCES [dbo].[Направление работы] ([ID_работы])
GO

ALTER TABLE [dbo].[Командировки] CHECK CONSTRAINT [FK_Командировки_Направление работы1]
GO

USE [Красный Крест]
GO
```

Таблица «Направление работы»

```
/****** Object:  Table [dbo].[Направление работы]    Script Date: 31.05.2021 16:59:41 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Направление работы](
```

```
	[закупки] [money] NULL,
	[безопасность] [money] NULL,
	[ID_работы] [int] NOT NULL,
 CONSTRAINT [PK_Направление работы] PRIMARY KEY CLUSTERED
(
	[ID_работы] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Направление работы] ADD  CONSTRAINT [DF_Направление работы_закупки]  DE-
FAULT ((15000000)) FOR [закупки]
GO

ALTER TABLE [dbo].[Направление работы] ADD  CONSTRAINT [DF_Направление работы_безопасность]
DEFAULT ((6000000)) FOR [безопасность]
GO
```

Таблица «Направление работы_сотрудники»

```
USE [Красный Крест]
GO

/****** Object:  Table [dbo].[Направление работы_сотрудники]    Script Date: 31.05.2021
16:59:54 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Направление работы_сотрудники](
	[ID_сотрудника] [int] NOT NULL,
	[ID_работы] [int] NOT NULL
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Направление работы_сотрудники]  WITH CHECK ADD  CONSTRAINT [FK_Направление
работы_сотрудники_Направление работы] FOREIGN KEY([ID_работы])
REFERENCES [dbo].[Направление работы] ([ID_работы])
GO

ALTER TABLE [dbo].[Направление работы_сотрудники] CHECK CONSTRAINT [FK_Направление рабо-
ты_сотрудники_Направление работы]
GO

ALTER TABLE [dbo].[Направление работы_сотрудники]  WITH CHECK ADD  CONSTRAINT [FK_Направление
работы_сотрудники_Сотрудники] FOREIGN KEY([ID_сотрудника])
REFERENCES [dbo].[Сотрудники] ([ID_сотрудника])
GO

ALTER TABLE [dbo].[Направление работы_сотрудники] CHECK CONSTRAINT [FK_Направление рабо-
ты_сотрудники_Сотрудники]
GO
```

Таблица «Пациенты»

```
USE [Красный Крест]
GO

/****** Object:  Table [dbo].[Пациенты]    Script Date: 31.05.2021 17:00:06 ******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Пациенты](
	[фамилия] [nvarchar](40) NULL,
	[имя] [nvarchar](40) NULL,
	[отчество] [nvarchar](40) NULL,
	[паспорт] [nvarchar](40) NULL,
	[город пациента] [nvarchar](40) NULL,
	[телефон] [nvarchar](40) NULL,
	[ID_сотрудника] [int] NOT NULL,
	[ID_пациента] [int] NOT NULL,
	[Электронная почта] [nvarchar](40) NULL,
 CONSTRAINT [PK_Пациенты] PRIMARY KEY CLUSTERED
(
	[ID_пациента] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Пациенты]  WITH CHECK ADD  CONSTRAINT [FK_Пациенты_Текущая деятельность1]
FOREIGN KEY([ID_сотрудника])
REFERENCES [dbo].[Текущая деятельность] ([ID_работы])
GO

ALTER TABLE [dbo].[Пациенты] CHECK CONSTRAINT [FK_Пациенты_Текущая деятельность1]
GO
```

Таблица «Результаты обследования»

```
USE [Красный Крест]
GO

/****** Object:  Table [dbo].[Результаты обследования]    Script Date: 31.05.2021 17:00:25
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Результаты обследования](
	[дата обращения] [datetime] NULL,
	[код диагноза] [int] NULL,
	[амбулаторное лечение] [nvarchar](40) NULL,
	[примечание] [nvarchar](40) NULL,
	[ID_пациента] [int] NOT NULL,
	[ID_личности] [int] NULL,
 CONSTRAINT [PK_Результаты обследования] PRIMARY KEY CLUSTERED
(
	[ID_пациента] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Результаты обследования] ADD  CONSTRAINT [DF_Результаты обследова-
ния_амбулаторное лечение]  DEFAULT (N'отсутствует') FOR [амбулаторное лечение]
GO

ALTER TABLE [dbo].[Результаты обследования] ADD  CONSTRAINT [DF_Результаты обследова-
ния_примечание]  DEFAULT (N'отсутствует') FOR [примечание]
```

```
GO

ALTER TABLE [dbo].[Результаты обследования]  WITH CHECK ADD  CONSTRAINT [FK_Результаты обсле-
дования_Беженцы] FOREIGN KEY([ID_личности])
REFERENCES [dbo].[Беженцы] ([ID_личности])
GO

ALTER  TABLE  [dbo].[Результаты  обследования]  CHECK  CONSTRAINT  [FK_Результаты  обследова-
ния_Беженцы]
GO

ALTER TABLE [dbo].[Результаты обследования]  WITH CHECK ADD  CONSTRAINT [FK_Результаты обсле-
дования_Пациенты] FOREIGN KEY([ID_личности])
REFERENCES [dbo].[Пациенты] ([ID_пациента])
GO

ALTER  TABLE  [dbo].[Результаты  обследования]  CHECK  CONSTRAINT  [FK_Результаты  обследова-
ния_Пациенты]
GO
```

Таблица «Снабжение»

```
USE [Красный Крест]
GO

/****** Object:  Table [dbo].[Снабжение]    Script Date: 31.05.2021 17:00:47 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Снабжение](
	[Медицинское] [nvarchar](40) NULL,
	[Продуктовое] [nvarchar](40) NULL,
	[ID_личности] [int] NOT NULL,
 CONSTRAINT [PK_Снабжение] PRIMARY KEY CLUSTERED
(
	[ID_личности] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Снабжение] ADD  CONSTRAINT [DF_Снабжение_Медицинское]  DEFAULT (N'отказ от
услуги') FOR [Медицинское]
GO

ALTER TABLE [dbo].[Снабжение] ADD  CONSTRAINT [DF_Снабжение_Продуктовое]  DEFAULT (N'отказ от
услуги') FOR [Продуктовое]
GO

ALTER TABLE [dbo].[Снабжение]  WITH CHECK ADD  CONSTRAINT [Снабжение_Беженцы_FK_Rule] FOREIGN
KEY([ID_личности])
REFERENCES [dbo].[Беженцы] ([ID_личности])
GO

ALTER TABLE [dbo].[Снабжение] CHECK CONSTRAINT [Снабжение_Беженцы_FK_Rule]
GO
```

Таблица «Сотрудники»

```
USE [Красный Крест]
GO
```

```
/****** Object:  Table [dbo].[Сотрудники]    Script Date: 31.05.2021 17:01:21 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_PADDING ON
GO

CREATE TABLE [dbo].[Сотрудники](
	[ID_сотрудника] [int] NOT NULL,
	[имя] [nvarchar](40) NULL,
	[фамилия] [nvarchar](40) NULL,
	[отчество] [nvarchar](40) NULL,
	[дата_рождения] [datetime] NULL,
	[телефон] [nvarchar](40) NULL,
	[пол] [char](40) NULL,
	[возраст] [int] NULL,
	[заработная плата] [money] NULL,
	[паспорт] [int] NULL,
 CONSTRAINT [PK_Сотрудники] PRIMARY KEY CLUSTERED
(
	[ID_сотрудника] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

SET ANSI_PADDING OFF
GO
```

Таблица «Текущая деятельность»

```
USE [Красный Крест]
GO

/****** Object:  Table [dbo].[Текущая деятельность]    Script Date: 31.05.2021 17:01:34
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Текущая деятельность](
	[наименование деятельности] [nvarchar](40) NULL,
	[ID_работы] [int] NOT NULL,
	[продолжительность смены] [int] NULL,
	[услуги] [nvarchar](40) NULL,
 CONSTRAINT [PK_Текущая деятельность] PRIMARY KEY CLUSTERED
(
	[ID_работы] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Текущая деятельность] WITH CHECK ADD CONSTRAINT [FK_Текущая
деятельность_Направление работы1] FOREIGN KEY([ID_работы])
REFERENCES [dbo].[Направление работы] ([ID_работы])
GO
```

```
ALTER TABLE [dbo].[Текущая деятельность] CHECK CONSTRAINT [FK_Текущая деятель-
ность_Направление работы1]
GO
```

Хранимая процедура №1

Алгоритм выполняет удаление из базы данных сотрудников, которых отправляют в отставку при достижении ими возраста старше 65 лет и опыта работы за рубежом меньше установленного нами ограничения.

```
USE [Красный Крест]
GO
/****** Object:  StoredProcedure [dbo].[procedura]    Script Date: 31.05.2021 16:57:03
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[procedura] (@surname NVARCHAR(40),@ogr FLOAT) --значение по умолчанию
фамилии, ограничение, которое далее мы сами вписы-ваем в запрос.
AS
BEGIN
	DECLARE @id int = (SELECT ID_сотрудника FROM dbo.Сотрудники WHERE @surname=Фамилия) --
объявляем @id из таблицы сотрудники, присваиваем  @surname к столбцу Фамилия
	DECLARE @rubeg int = (SELECT [опыт работы за рубежом] FROM dbo.Квалификация WHERE
@id=ID_сотрудника) --объявляем @rubeg из таблицы опыт работы за рубежом, присваиваем @id
столбец ID_сотрудника
BEGIN TRY
		IF (SELECT MAX(возраст) FROM dbo.Сотрудники WHERE Фамилия IN (@surname))
> 65 --прописываем условие: максимальный возраст сотрудника из таблицы сотруд-ники больше 65,
поиск по фамилии.
BEGIN
DELETE FROM dbo.[Направление работы_сотрудники] WHERE ID_сотрудника IN (@id) --удаление из
таблицы направление работы_сотрудники ID_сотрудника
DELETE FROM dbo.Сотрудники WHERE ID_сотрудника IN (@id) --удаление из таблицы сотрудники
ID_сотрудника
END
ELSE
BEGIN
PRINT 'Сотрудник не найден' --если условия не выполяются, то выводит 'Сотрудник не найден'
END
END TRY
BEGIN CATCH
DELETE FROM dbo.Квалификация WHERE [опыт работы за рубежом] < @ogr AND ID_сотрудника IN (@id)
--удаляем из таблицы квалификация ID_сотрудника, если опыт работы за рубе-жом меньше вписыва-
емого нами ограничения.
DELETE FROM dbo.Сотрудники WHERE ID_сотрудника IN (@id) --удаляем из таблицы сотрудники
ID_сотрудника.
PRINT 'Сотрудник удалён' --при выполнении условий выводит 'Сотрудник удалён'
END CATCH
END
```

Хранимая процедура №2

Алгоритм выполняет обновление базы данных сотрудников, при условии, если в установленной нами горячей точке количество выживших превышает 100 человек, то отделу назначается премия в размере установленной нами заработной платы.

```
USE [Красный Крест]
GO
/****** Object:   StoredProcedure [dbo].[procedura_2]      Script Date: 31.05.2021 16:57:47
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[procedura_2] (@tochka NVARCHAR(40),@newZarplata MONEY) -- значение по
умолчанию горячей точки, вписываем новую зарплату.
AS
BEGIN
       DECLARE @visivsie int = (SELECT [количество выжевших] FROM dbo.Командировки WHERE
@tochka=[наименование горячей точки]) --объявляем @visivsie из таблицы командировки, присваи-
ваем @tochka к столб-цу наименование горячей точки.

                    IF (SELECT MAX([количество выжевших])FROM dbo.Командировки WHERE [количе-
ство выжевших] IN (@visivsie))> 100 --прописываем условие: максимальное количество выживших
из таблицы ко-мандировки превышает 100.
BEGIN
     UPDATE dbo.Сотрудники --обновляем таблицу сотрудники
        SET [заработная плата] = @newZarplata --присваиваем вписываемое нами значение столбцу
заработная плата
        PRINT 'Премия отделу' --если условия выполняются вывод 'Премия отделу'
END
ELSE
BEGIN
RAISERROR ('Не является максимальным количеством выживших',1,2) --создаём сообщение об ошиб-
ке, при невынолнении условий.
END
END
```

Текст пользовательской табличной функции.

Функция выводит таблицу, в которой отображется результат ввиде общей суммы закупок и безопасности.

```
USE [Красный Крест]
GO
ALTER FUNCTION funkcia1 (@id INT) --значение по умолчанию @id
RETURNS @return TABLE ("результат" MONEY) --возвращаем таблицу "результат" тип данных MONEY
AS
BEGIN
       DECLARE @zakupki money = (SELECT закупки FROM dbo.[Направление работы] WHERE
@id=ID_работы) --объявляем @zakupki из таблицы направление работы, присваиваем @id столбцу
ID_работы.
       DECLARE @bezopasnoct money = (SELECT безопасность FROM dbo.[Направление работы] WHERE
@id=ID_работы) --объявляем @bezopasnoct из таблицы Направление работы, присваиваем @id столб-
цу ID_работы.
       DECLARE @result MONEY --объявляем @result тип данных MONEY
       INSERT INTO @return --вставляем @return
       SELECT @zakupki+@bezopasnoct; --возвращаем сумму закупок с безопасностью.
       RETURN
END
```

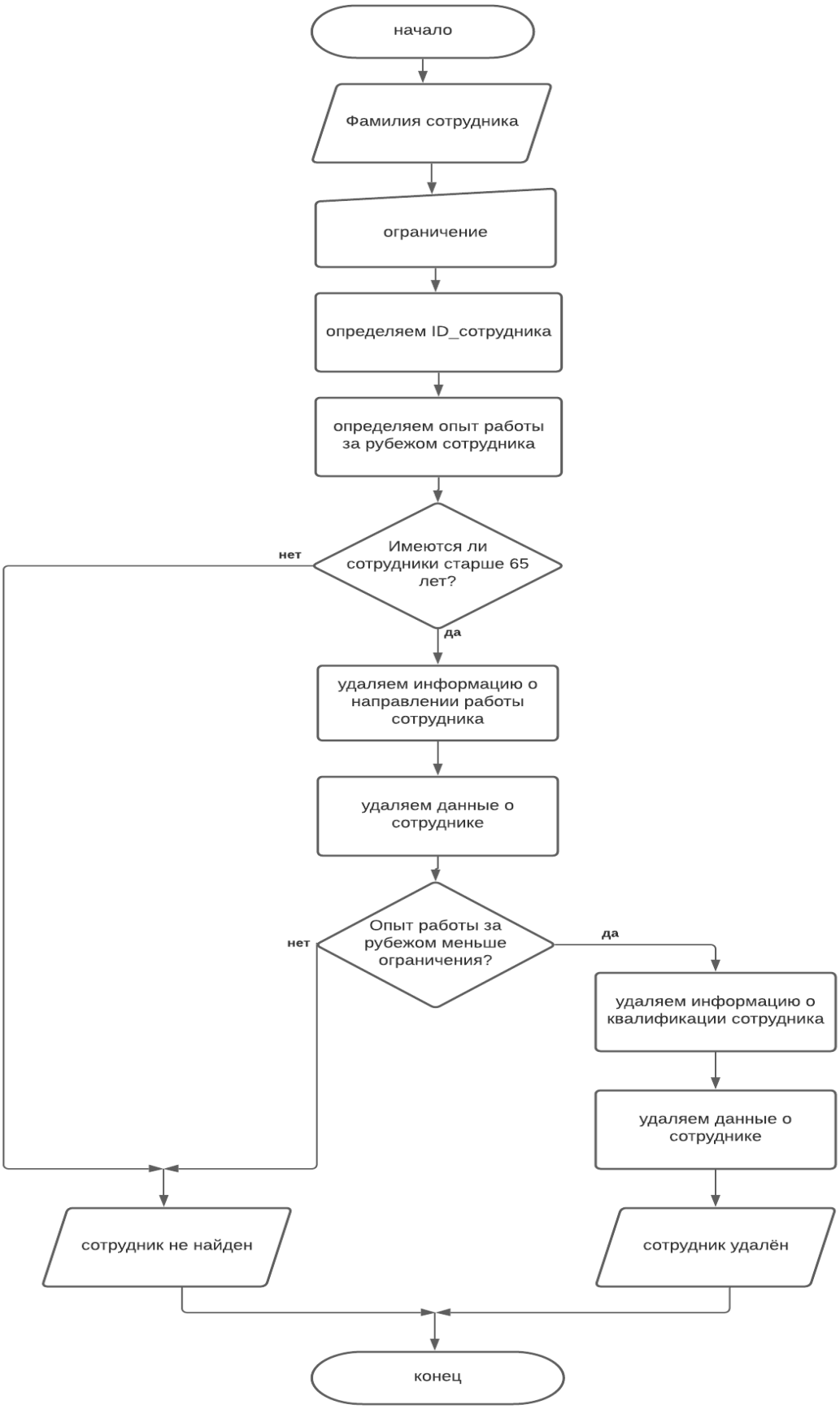
3 Алгоритмическое обеспечение системы

1. **Общая характеристика**: алгоритм выполняет удаление из базы данных сотрудников, которых отправляют в отставку при достижении ими возраста старше 65 лет и опыта работы за рубежом меньше установленного нами ограничения.

2. **Используемые данные**: у таблицы «Сотрудники» атрибуты ID_сотрудника, Фамилия, у таблицы «Квалификация» атрибуты ID_сотрудника, опыт работы за рубежом, у таблицы «Направление работы_сотрудники» атрибут ID_сотрудника.

3. **Результаты выполнения**: изменяются таблицы «Сотрудники» и «Квалификация» удаляется атрибут ID_сотрудника, а вместе с ним все атрибуты в назначенной строке, у таблицы «Направление работы_сотрудники» удаляется атрибут ID_сотрудника.

3. **Логическое описание**:



«Схема 1 – Логическое описание»

4 Прикладное программное обеспечение системы

4.1 Общая характеристика прикладного программного обеспечения

Прикладное программное обеспечение соответствует предъявляемым требованиям – общим и согласно выбранному «уровню 4».

1. Проведён анализ заданной предметной области «Красный Крест» и разработано техническое задание на создаваемую автоматизированную систему по ГОСТ 34.602-89 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы».

2. Разработана автоматизированная система на основе следующих общих требований.

- Инструментальная среда разработки – Visual Studio Express 2019 for Web.
- СУБД – SQL Server 2012 Express.
- Базовые возможности интерфейса для всех таблиц базы данных:

– просмотр данных в табличном виде;

– ввод новых записей, редактирование записей – в виде бланка;

– удаление записей.

Все операции должны реализовываться корректно (с сохранением логической и ссылочной целостности).

- Фильтрация отображаемых данных в таблице «Квалификация» реализуется методом Index, она сортирует опыт работы в регионах, между установленным нами диапазоном, с помощью двух CheckBox, в первом записывается первое значение, во втором второе. Фильтрация отображаемых данных в таблице «Беженцы» реализуется методом Index и сортирует по первой букве страны и имени беженца, с помощью CheckBox.
- Наличие поиска по базе данных по фамилии в таблице «Пациенты» реализуется с помощью метода Index, с помощью CheckBox.

- Моделирование данных для предметной области «Общество Красного Креста» выполнено с помощью средства автоматизации ERConstructor 2.0. Таблицы базы данных заполняются правдоподобными и согласованными между собой данными (строковые данные – на русском языке). Размер каждой из рабочих таблиц –7 записей.

К разработке также предъявляются следующие дополнительные требования.

- «Уровень 4»

– технология разработки –ASP.NET MVC;

– количество таблиц в базе данных – 9, с суммарным количеством атрибутов – 42;

– количество типов связей 4;

– технология доступа к данным – «классический» ADO.NET Entity Framework;

– наличие самостоятельно реализованной аутентификации по логину и паролю, данные сохраняются и возвращаются через SQL.

– создание в таблице «Пациенты» уникальных индексов паспорта, телефона, эоектронной почты и создание в таблице «Снабжение» значения по умолчанию «отказ от услуги», в случае отказа беженца от предоставленных услуг.

– создание и использование хранимых процедур –2 и определяемой пользователем табличной функции – 1;

– применение в программе значимых имён файлов, классов и объектов; – обработка исключительных ситуаций в таблицах «Пациенты», «Командировки», «Сотрудники», «Направление работы», «Текущая деятельность», «Беженцы».

– выполнение работы вовремя.

4.2 Структура и состав прикладного программного обеспечения

В Visual Studio Express 2019 for Web в обозревателе решений в проекте «Красный Крест» используются:

1. Properties управляет общими сведениями о сборке. Сведения о версии сборки состоят из основного и дополнительного номера версии, номера сборки и редакции.
2. Ссылки содержат сведения о целевой платформе и её сборке.
3. App_Data содержит файлы данных приложения, включая файлы базы данных mdf, файлы XML и другие файлы хранилища данных.
4. App_Start содержит файлы конфигурации:
    - BungleConfig, который регистрирует CSS и JS, чтобы их можно было объединить и минимизировать.
    - В FilterConfig зарегистрированны глобальные фильтры. Эти фильтры применяются ко всем действиям и контроллерам.
    - RouteConfig указан маршрут нахождения информации.
5. Content содержит файлы css, реализуемые далее в проекте.
6. Controllers содержит контроллеры, которые отвечают за управление способом взаимодействия пользователя с приложением MVC.
7. Fonts содержит шрифты.
8. Models содержит модели и контекст данных.
9. Scripts содержит скрипты js, реализуемые в проекте.
10. Views содержит представления общих данных проекта.
11. Favicon отображает знак в браузере.
12. Global содержит таблицу маршрутов.
13. Model1 описывает рабочий, удаленный и связующий слой архитектуры Entity Framework на языках CSDL, SSDL и MDL, основанных на синтаксисе XML, но имеющих различную семантику.
14. packages используется в проектах некоторых типов для ведения списка пакетов, на которые ссылается проект.
15. Web.config определяет параметры для ASP.NET web-приложения. В нём хранится информация о параметрах поставщиков состояний сеансов, членства, определяются ссылки на страницы ошибок.

Контроллеры:

AccountController отвечает за вывод страниц регистрации пользователей и входа в систему. Предоставляет данные о поиске и наличии пользователя в системе. Значимых строк 48.

HomeController предоставляет данные о выводе страниц. Также выводит домашнюю страницу. Значимых строк 19.

RoleController содержит данные о выводе страниц роли администратора и пользователя. Значимых строк 11.

БеженцыController предоставляет данные о беженцах. Позволяет осуществить просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Значимых строк 111.

КвалификацияController предоставляет данные о квалификации сотрудников. Позволяет осуществить просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Значимых строк 90.

КомандировкиController предоставляет данные о командировках, куда направляют сотрудников. Позволяет осуществить просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Значимых строк 103.

Направление_работыController предоставляет данные о направлении работы сотрудников. Позволяет осуществить просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Используется функция, которая определяет общую сумму закупок, в том числе и в области безопасности. Значимых строк 94.

ПациентыController предоставляет данные о пациентах. Позволяет осуществить просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Значимых строк 97.

Результаты_обследованияController предоставляет данные о результатах обследования пациентов и беженцев. Позволяет осуществить просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Значимых строк 86.

СнабжениеController предоставляет данные о снабжении беженцев. Позволяет осуществить просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Значимых строк 82.

СотрудникиController предоставляет данные о сотрудниках организации. Позволяет осуществить просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Значимых строк 107.

Текушая_деятельностьController предоставляет данные о текущей деятельности сотрудников. Позволяет осуществить просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Значимых строк 87.

4.3 Особенности реализации и сопровождения

Благодаря простому веб-дизайну отсутствует лишняя информация, и сам дизайн намного лучше воспринимается пользователем. Это помогает быстрее находить необходимые навигационные элементы. За счет этого навигация становится более легкой.

Благодаря несложному коду реализации, разработчик избегает включения нескольких списков стилей, вызова нескольких Javascript файлов или же большого количества иного контента, повышающего количество HTTP запросов. В итоге весь сайт начинает работать значительно быстрее. И если медленные ресурсы только раздражают пользователей, то «летающие» сайты всегда воспринимаются положительно.

С помощью аутентификации мы создали дополнительную таблицу в SQL, в которой содержатся данные для входа администратора, они автоматически передаются в систему. Благодаря этому не пришлось использовать громоздкий код в Visual Studio для привязки данных для входа администратору.

5 Руководство пользователя

5.1 Общие сведения

Система предназначена для удовлетворения информационных потребностей пользователей и управления функционированием подразделений «Общества Красного Креста» и позволяет решать следующие задачи:

- предоставление информации пользователям о «Сообществе Красного Креста»;
- учет данных о сотрудниках;
- учет данных о пациентах и беженцах;
- планирование командировок, программ и проектов деятельности;
- учет данных о предоставляемых услугах.

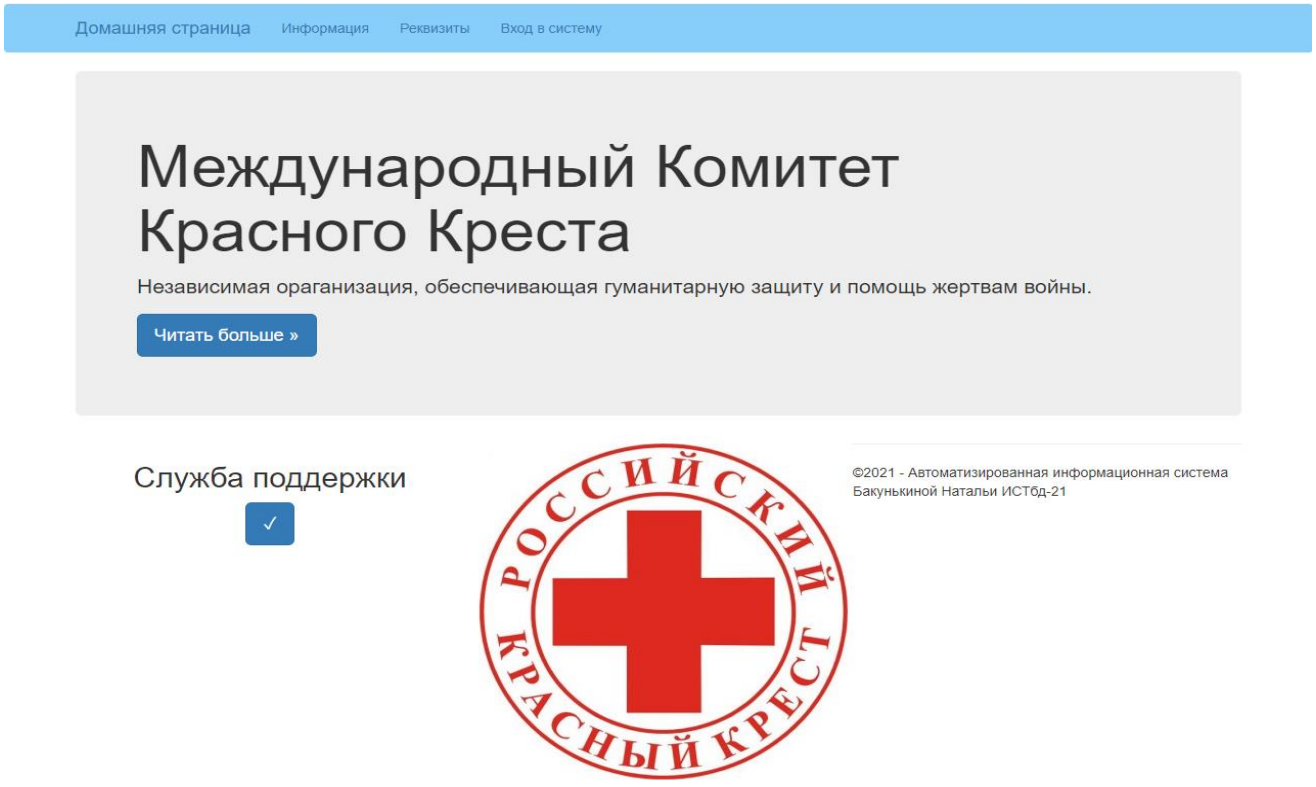
Система представляет собой веб-приложение, разработанное на платформе ASP.NET. Для решения задач (см. выше) и хранения данных используется база данных SQL общества Красного Креста.

Система позволяет просматривать, редактировать и управлять информацией, хранящейся в базе данных, через веб интерфейс. В системе присутствуют хранимые процедуры, функции для высокопроизводительной работы пользователя. Система проверяет корректность вводимых данных пользователем и сообщает об ошибках, при вводе неверных данных, что позволяет обеспечить высокую доступность к приложению. Таким образом, обеспечивается экономичное и масштабируемое использование ресурсов при одновременной автоматизации трудоемких задач администрирования, таких как настройка базы данных.

Пользователь с любым уровнем подготовки может использовать систему, поскольку система интерактивная и динамичная, то есть она реагирует на действия пользователя, обрабатывает его запросы и выдаёт результат.

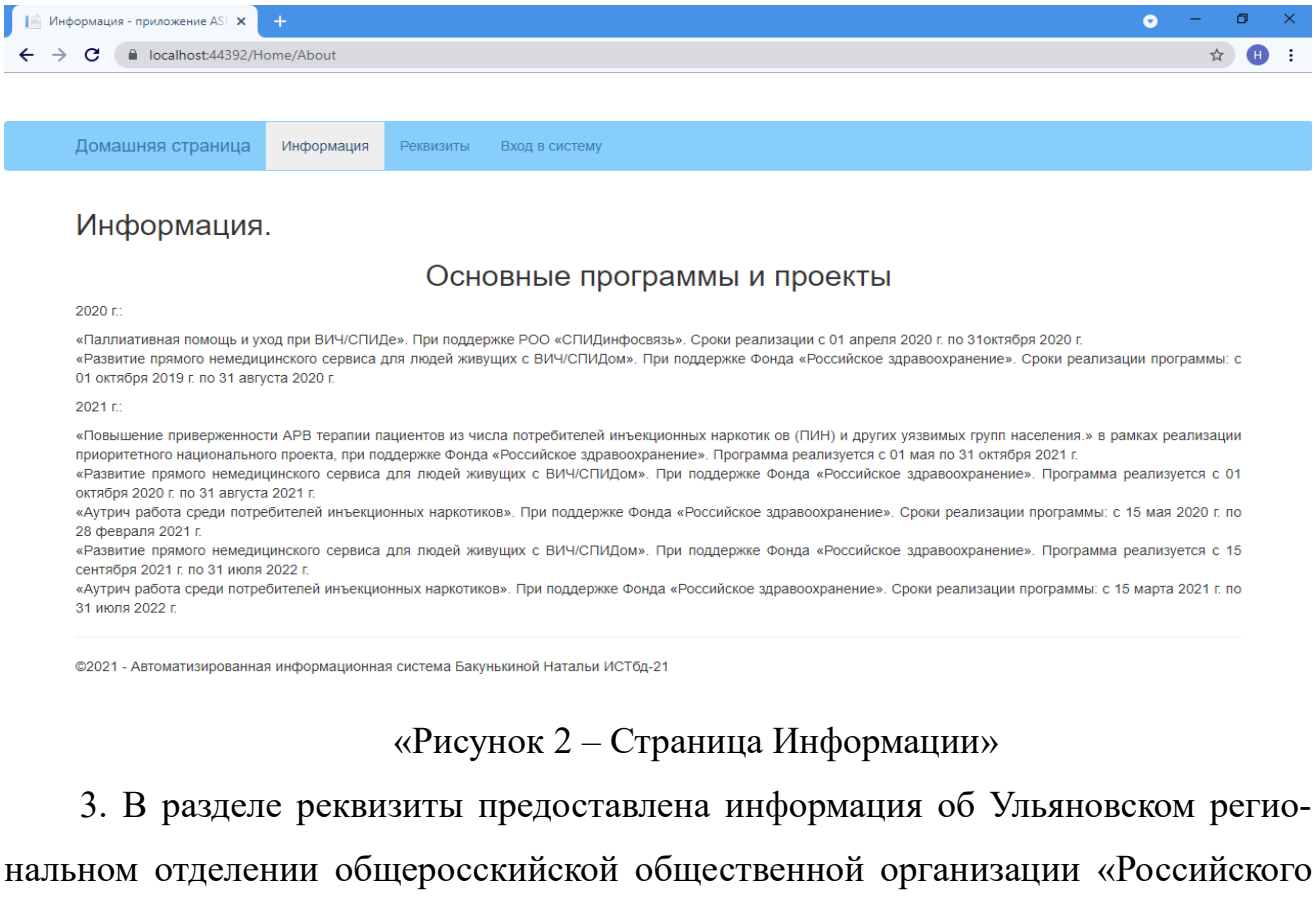
5.2 Порядок и особенности работы

1. При запуске программы пользователь перейдёт на главную страницу автоматизированной информационной системы по предметной области «Сообщества Красного Креста». В верхней части главной страницы реализован переход на дополнительные страницы сайта, в том числе и входа в систему. На странице предоставлена краткая информация об организации, также, перейдя по ссылке «Читать больше», пользователь перейдёт на сайт Ульяновского регионального отделения общероссийской общественной организации и сможет ознакомится с более обширной информацией. При нажатии на изображение, пользователь перейдёт на сайт общероссийской общественной организации Комитета Красного Креста и ознакомится с более глобальной информацией по своей специальности.



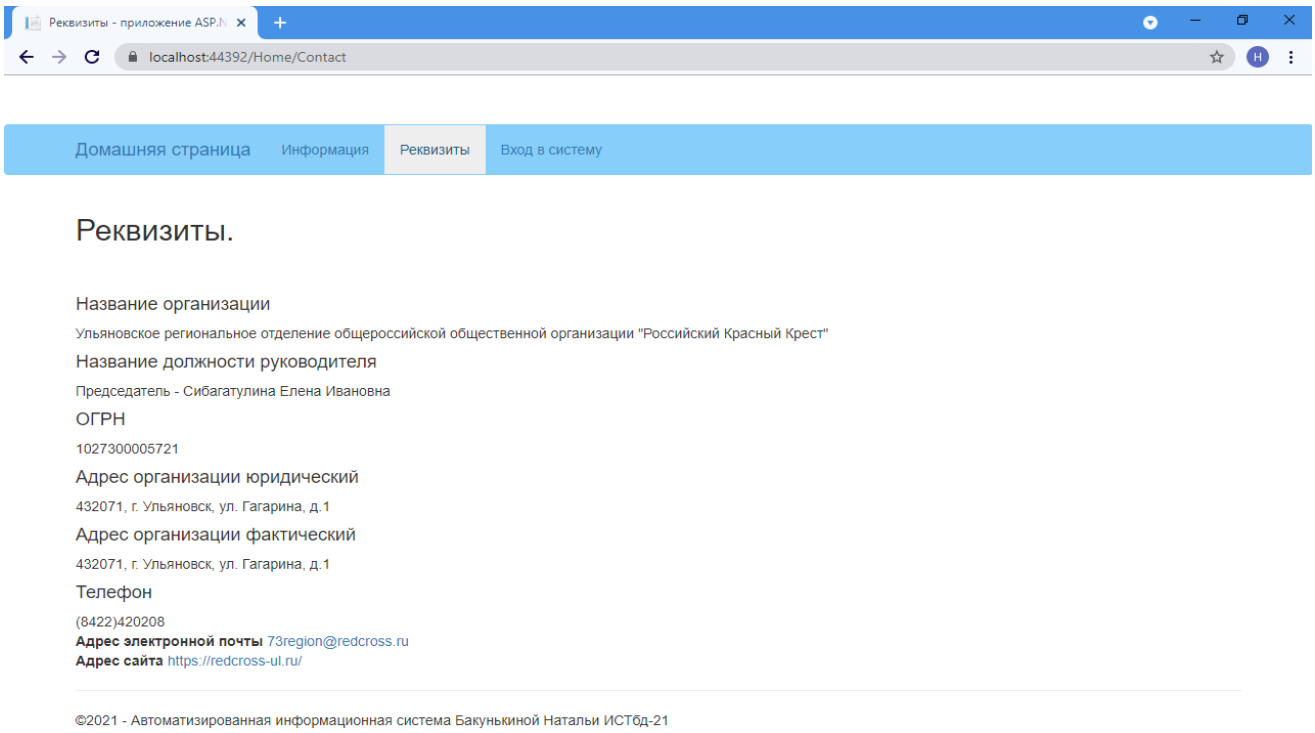
«Рисунок 1 – Главная страница»

2. Перейдя на страницу информации пользователь может ознакомиться с основными программами и проектами своей деятельности. При чём здесь предоставлены задачи, которые уже завершились по истечению времени, также предоставлены задачи на будущие проекты и программы, которые должны быть реализованы до определённого срока, который прописывается в каждом пункте в конце предложения.



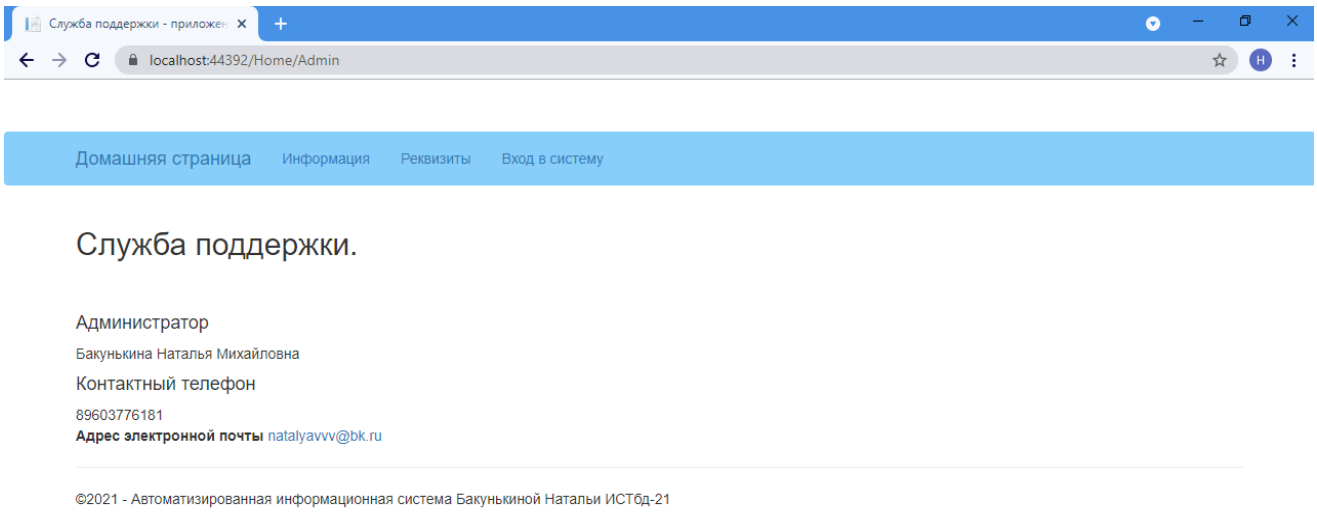
«Рисунок 2 – Страница Информации»

3. В разделе реквизиты предоставлена информация об Ульяновском региональном отделении общероссийской общественной организации «Российского Красного Креста», а именно указана должность и ФИО руководителя, основной государственный регистрационный номер записи о создании юридического лица, юридический и фактический адресс организации, контактный телефон, адрес электронной почты для связи с организацией.



«Рисунок 3 – Страница Реквизитов»

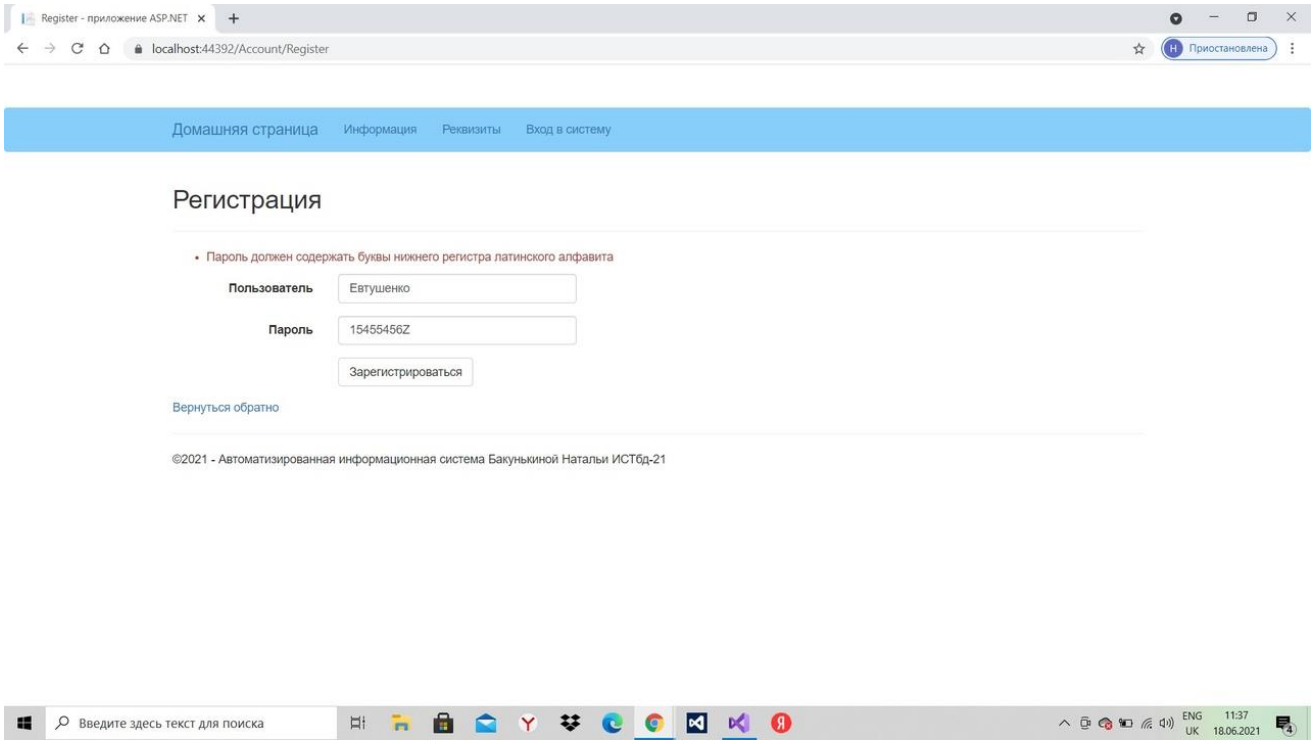
4. Если у пользователя возникают проблемы с эксплуатацией сайта, то он может обратиться в раздел службы поддержки для связи с администратором, где указаны ФИО администратора, его контактные данные, адрес электронной почты. Задача администратора принимать обращения пользователей, у которых возникают проблемы, фиксировать их и решать. Иногда для решения проблемы пользователя достаточно ответа на вопрос, а в других случаях требуется разбираться в проблеме, предоставлять развернутое объяснение и возвращать работоспособность решению. Каким бы ни было обращение, от службы поддержки требуется восстанавливать работу обслуживаемой инфраструктуры в кратчайшие сроки.



«Рисунок 4 – Страница Службы поддержки»

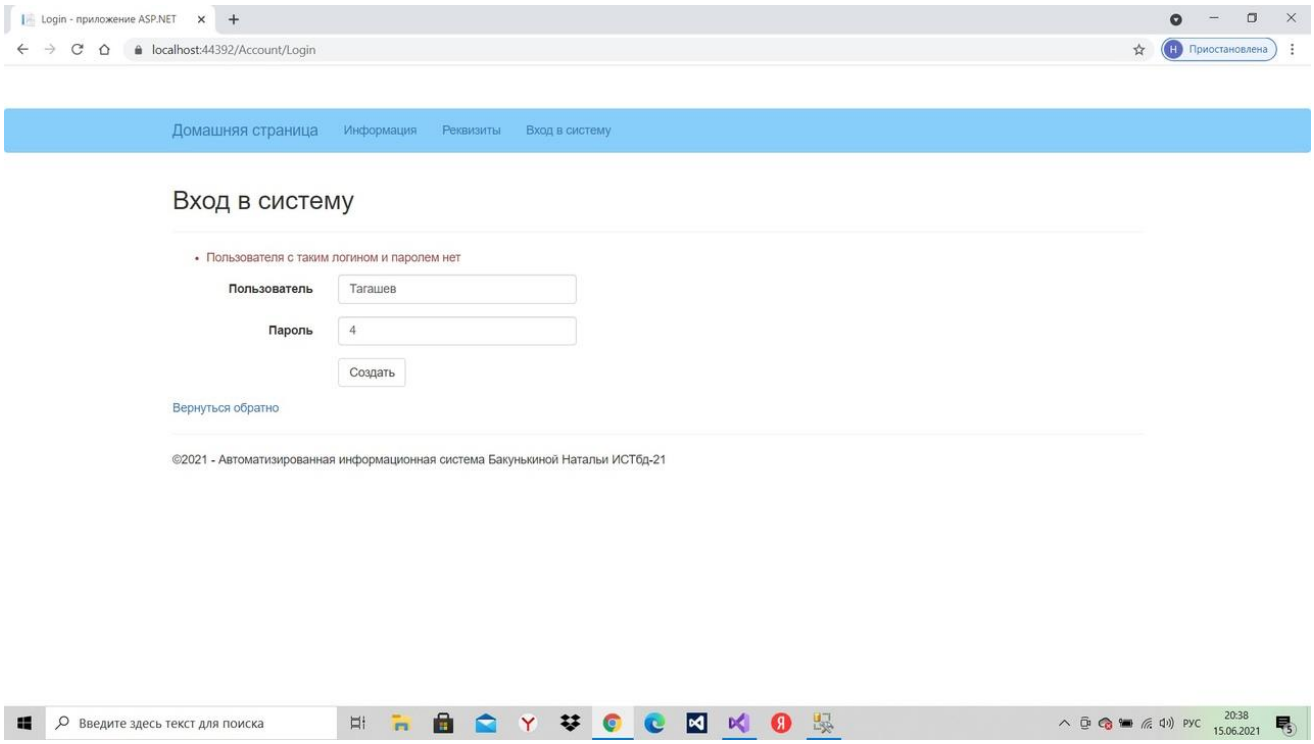
5. Чтобы получить доступ к таблицам базы данных, пользователю необходимо зарегистрироваться в системе, придумая свой логин и пароль. Если вы уже зарегистрированы в системе и пытаетесь зарегистрироваться под тем же логином и паролем, то система выдаст сообщение «Пользователь с таким логином уже существует», тогда вам нужно будет перейти на страницу входа на сайт и ввести уже зарегистрированные данные. Если у вас только первая регистрация, то она пройдёт успешно. Также на ввод пароля есть ограничения, если вы вводите меньше 8 символов, то программа выдаст сообщение «Пароль должен содержать минимум 8 символов», если не ввели заглавную букву латинского алфавита, то

выводит «Пароль должен содержать заглавные буквы верхнего регистра латинского алфавита», если не ввели строчкую букву латинского алфавита, то выводит «Пароль должен содержать строчные буквы нижнего регистра латинского алфавита»



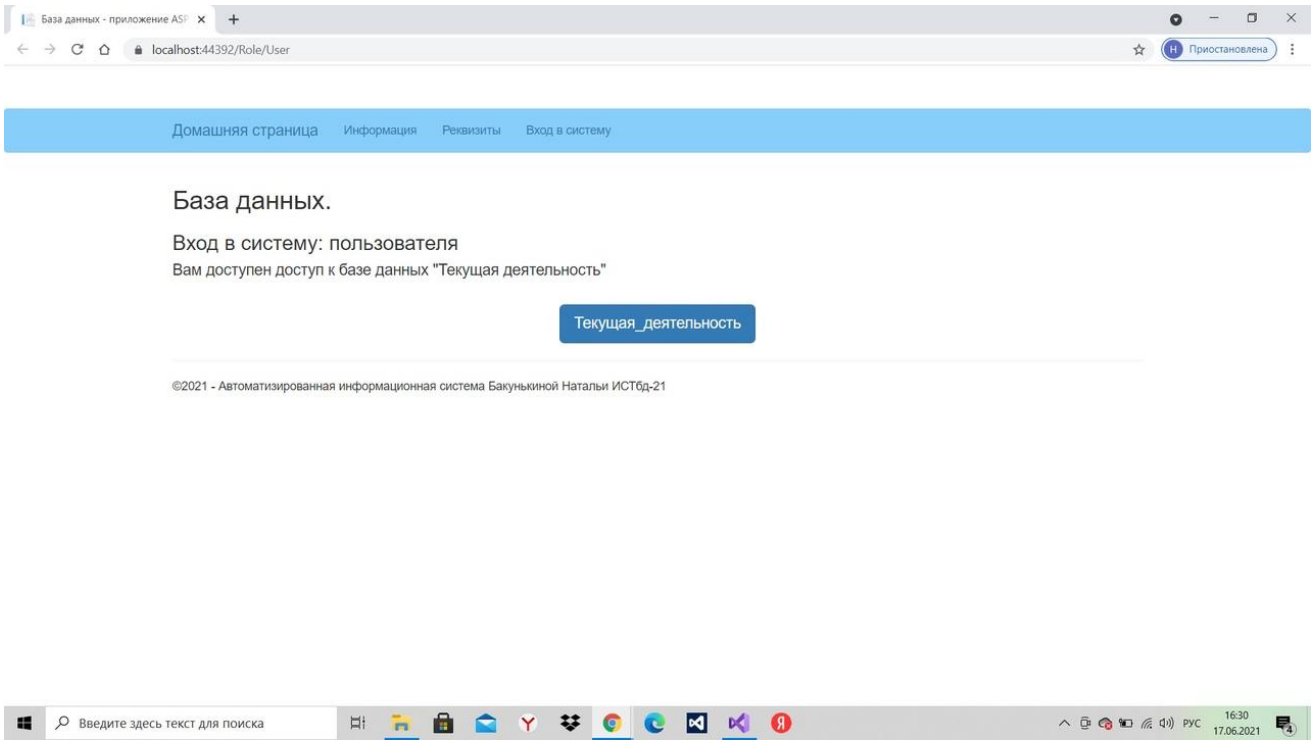
«Рисунок 5 – Регистрация»

6. Если вы не зарегистрировались, при пытке входа в систему будет выведено сообщение «Пользователя с таким логином и паролем нет». Вам нужно будет пройти сначала регистрацию, только потом производить вход в систему. При входе в систему, пользователя перенаправит на страницу базы данных.



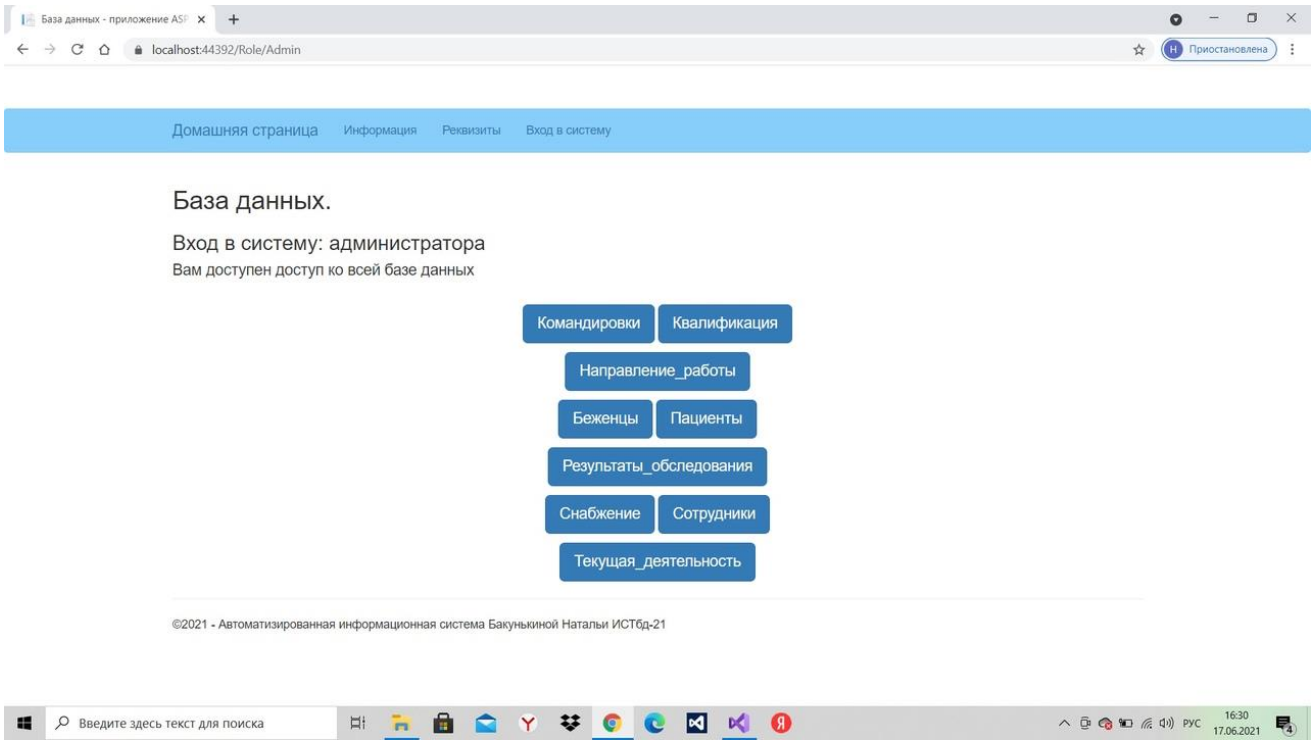
«Рисунок 6 – Вход в систему»

7. Перейдя на страницу базы данных, обычный пользователь увидит на странице, что вход произошёл под именем пользователя, и что он имеет доступ к базам данных таблицы «Текущая_деятельность», так как в ней не содержится информации ограниченного доступа. В ней пользователь сможет ознакомится с деятельностью подразделений общества Красного Креста, проосмотреть продолжительность смены сотрудников, изучить услуги. (См. текущую деятельность «Рисунок 25 – Страница Текущая деятельность»)



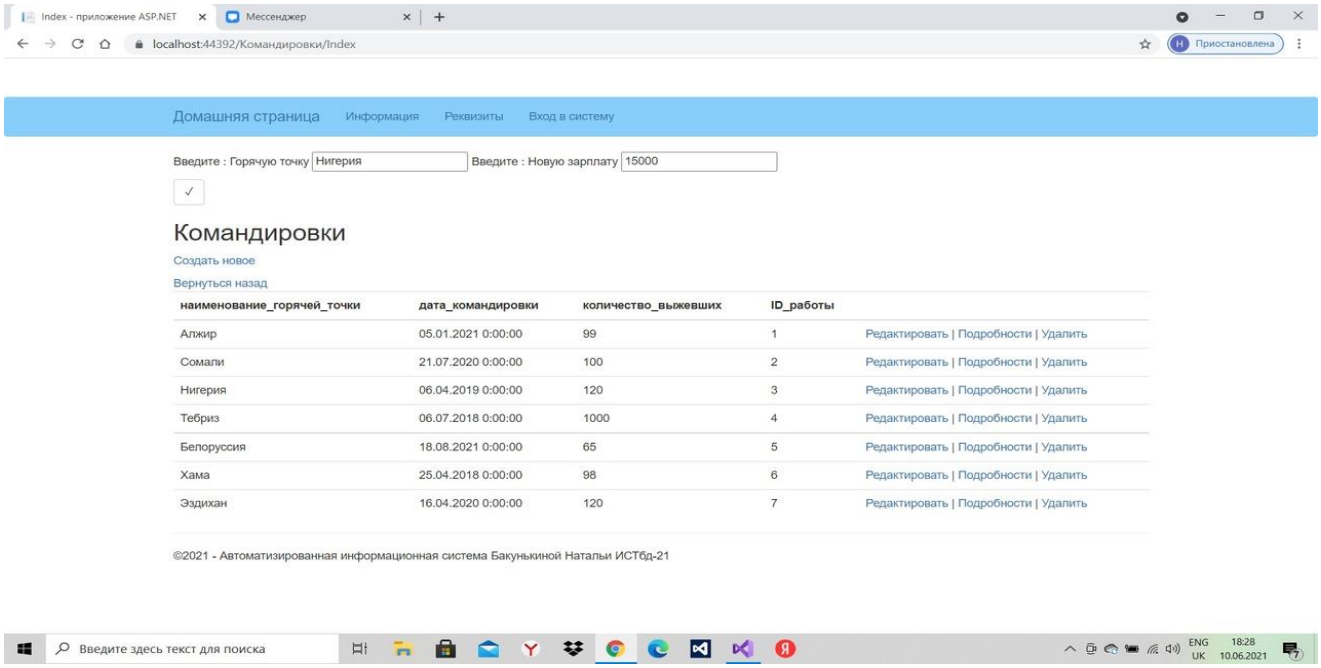
«Рисунок 7 – Вход под пользователем»

8. Перейдя на страницу базы данных, администратор увидит на странице, что вход произошёл под именем администратора, и что он имеет открытый доступ ко всей базе данных таблиц: «Командировки», «Квалификация», «Направление_работы», «Беженцы», «Пациенты», «Результаты_обследования», «Снабжение», «Сотрудники», «Текущая_деятельность».



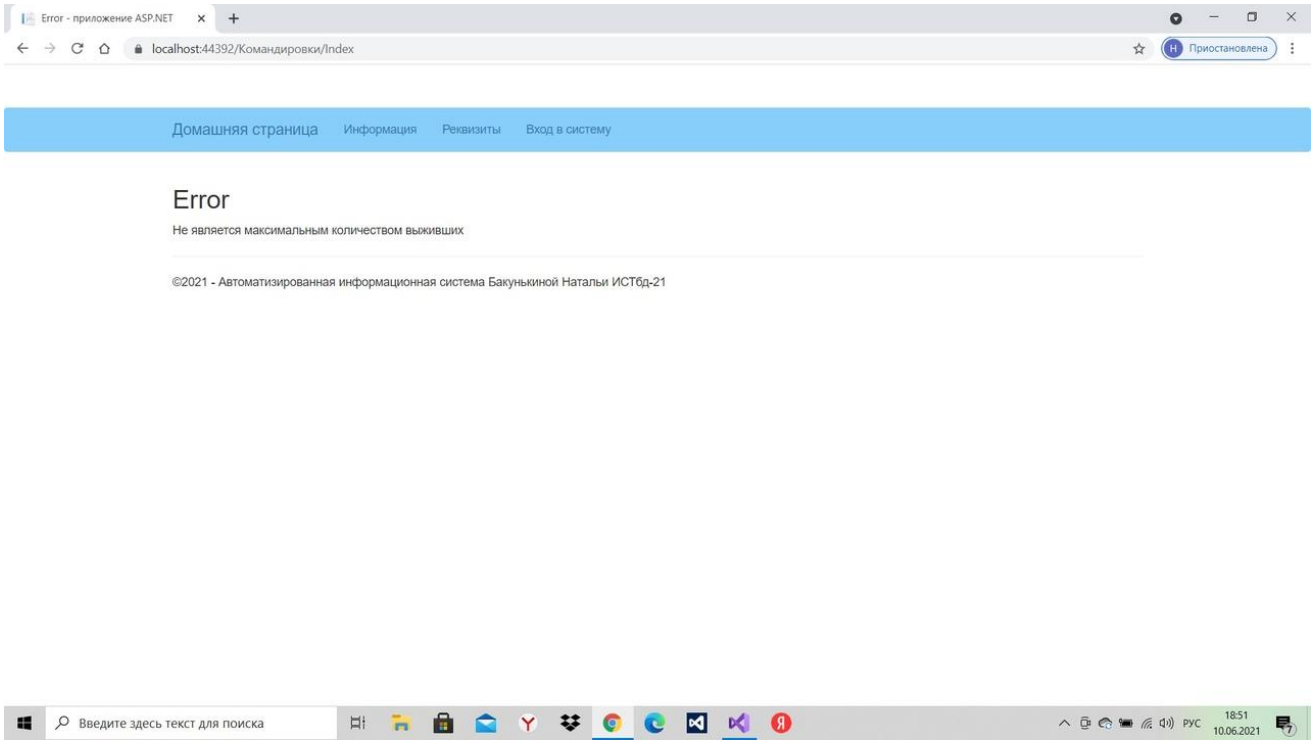
«Рисунок 8 – Вход под администратором»

9. В таблице командировки применяется просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Также присутствует хранимая процедура, которая обновляет данные таблицы «Сотрудники», при условии, если в установленной нами горячей точке количество выживших превышает 100 человек, то отделу назначается премия в размере установленной нами заработной платы.



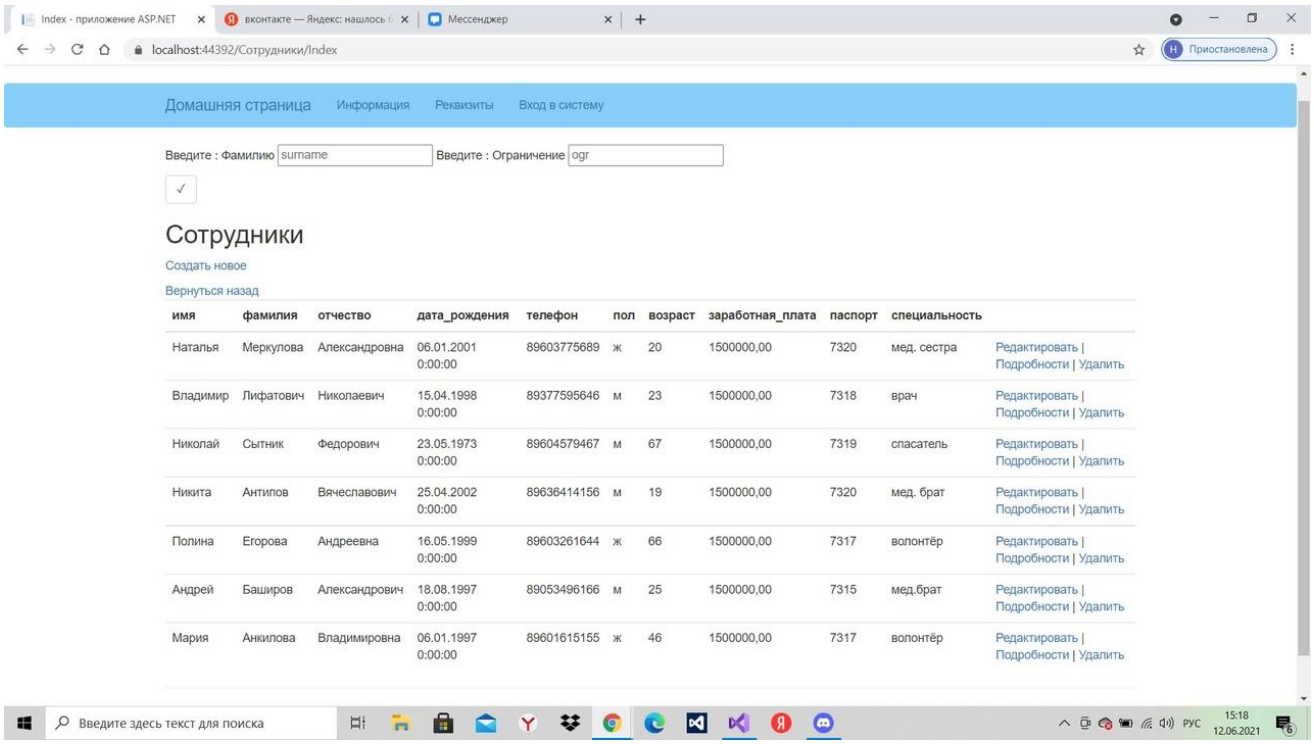
«Рисунок 9 – Страница Командировки»

10. Если вы ввели некоректные данные, или если в установленной нами горячей точке количество выживших не превышает 100 человек, тогда сайт перенаправляет нас на страницу ошибки и выдаёт сообщение «Не является максимальным количеством выживших».



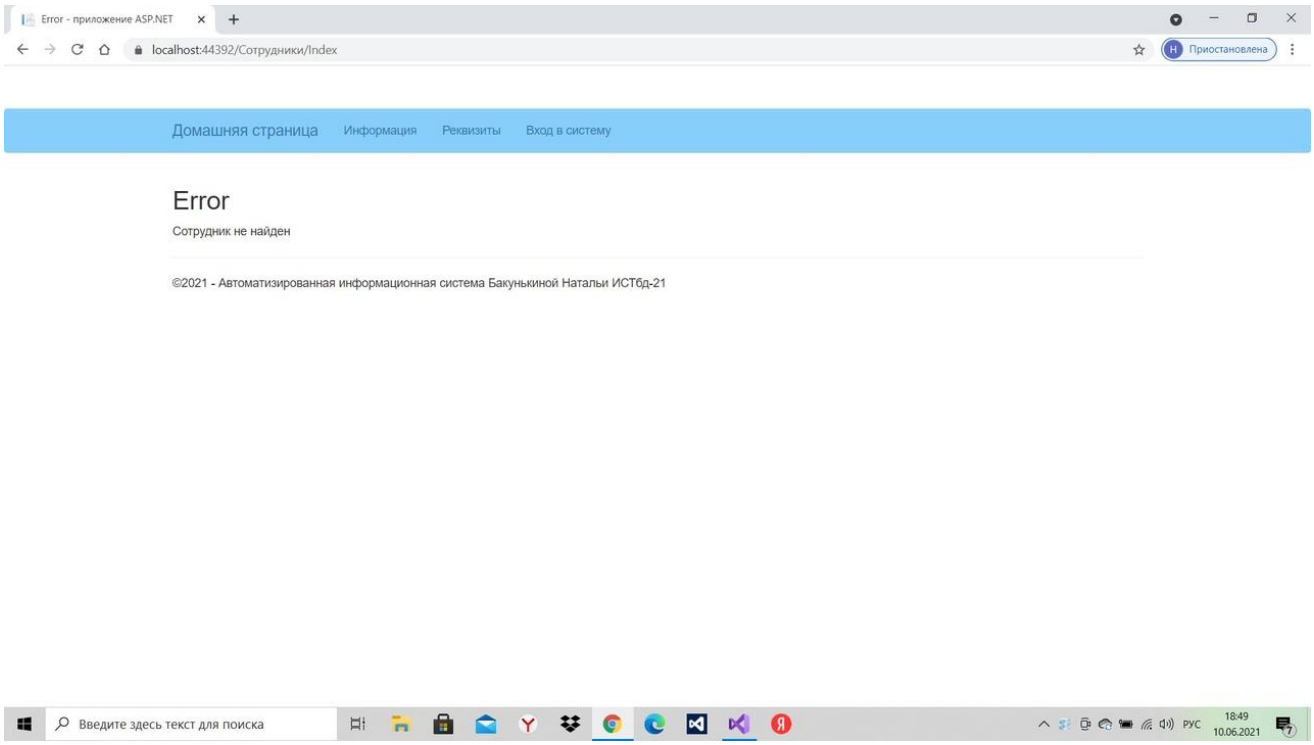
«Рисунок 10 – Страница Ошибки»

11. В таблице «Сотрудники» применяется просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Также присутствует хранимая процедура, которая выполняет удаление из базы данных сотрудников, которых отправляют в отставку при достижении ими возраста старше 65 лет и опыта работы за рубежом, меньше установленного нами ограничения.



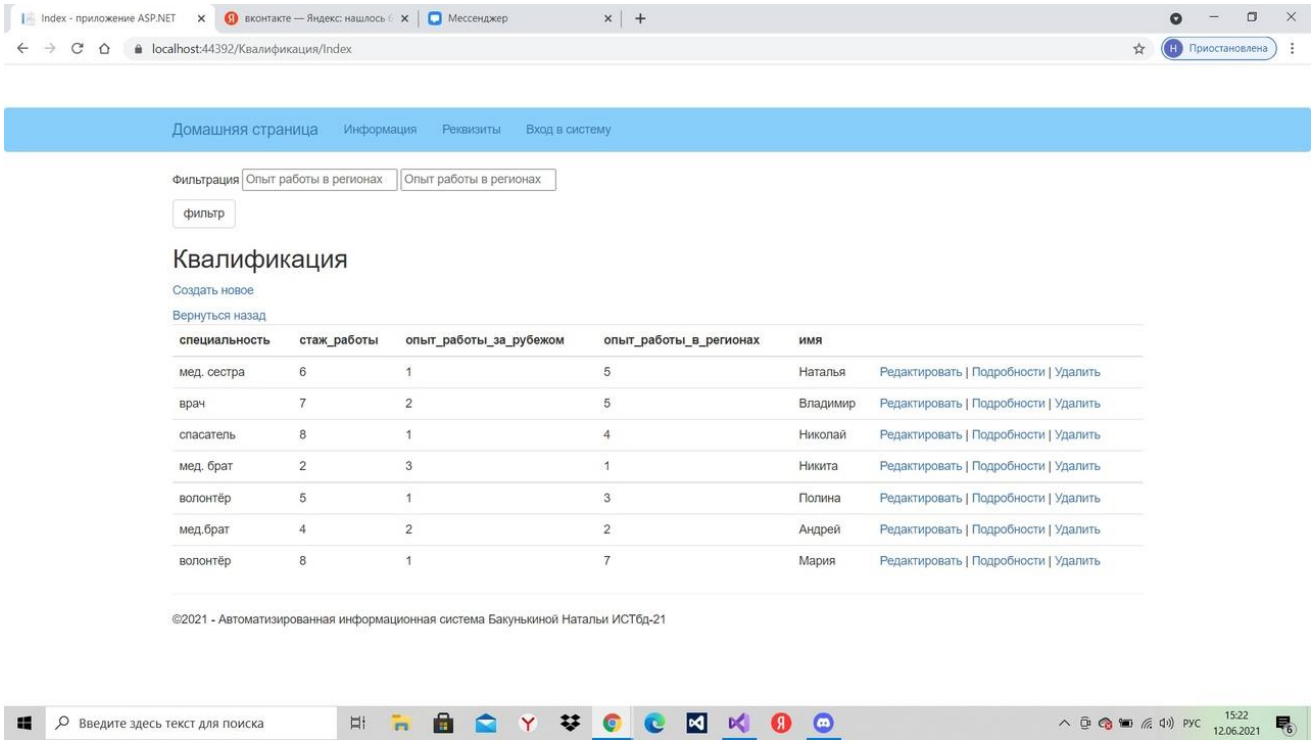
«Рисунок 11 – Страница Сотрудники»

12. Если вы ввели некоректные данные, или если вы ввели возраст меньше 65 лет и опыт работы за рубежом больше, установленного нами ограничения, тогда сайт перенаправляет нас на страницу ошибки и выдаёт сообщение «Сотрудник не найден».



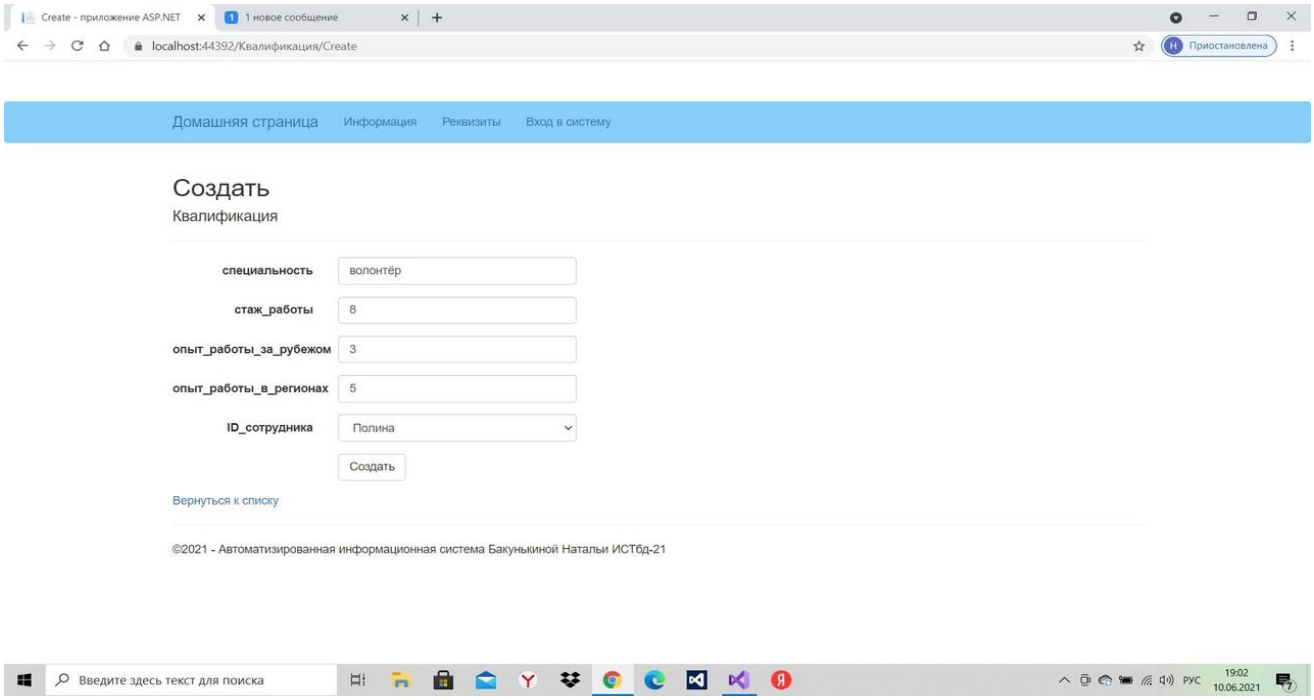
«Рисунок 12 – Страница Ошибки»

13. В таблице квалификация применяется просмотр данных в табличном виде. На примере этой таблицы далее продемонстрируется ввод новых записей, редактирование записей – в виде бланка, удаление записей. Подобные действия воспроизводятся с каждой таблицей.



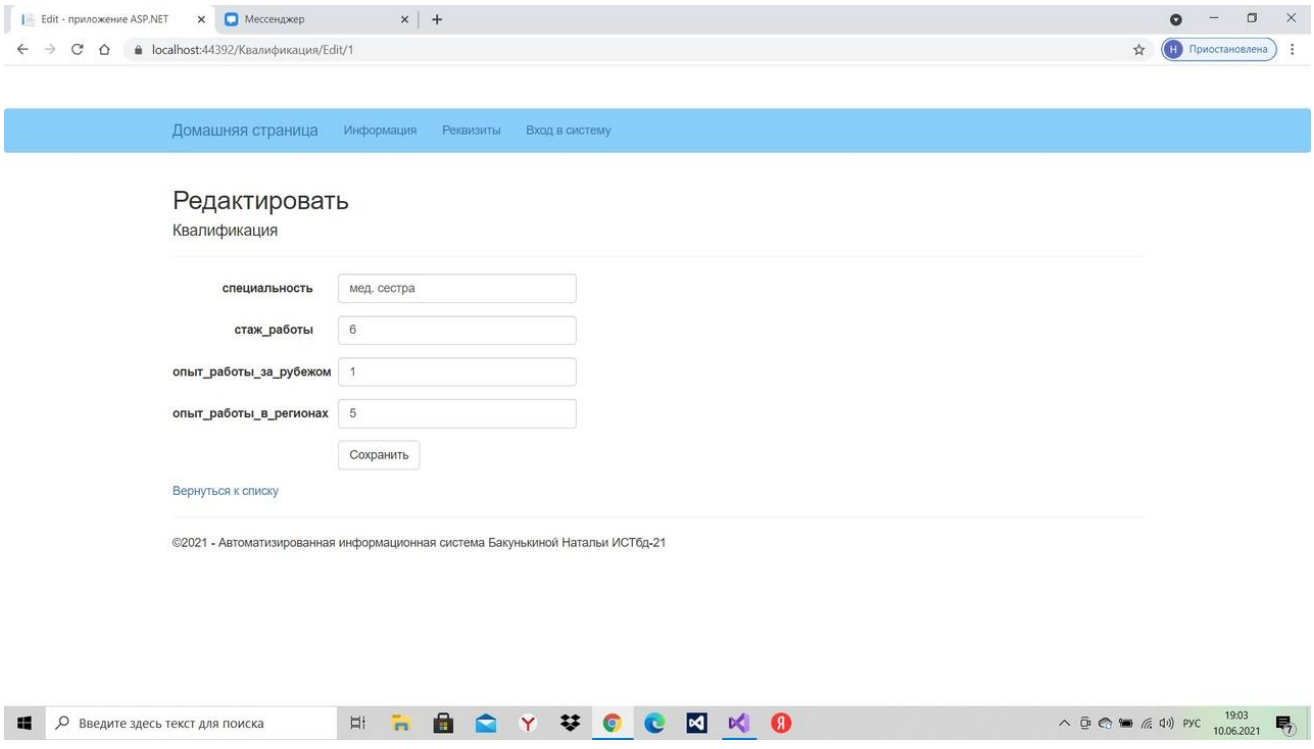
«Рисунок 13 – Страница Квалификации»

14. Нажимая на кнопку «создать новое» мы переходим на страницу, где вводим новые данные в таблицу. В таблицу «Квалификация» мы вводим новые данные для столбцов: специальность, стаж работы, опыт работы за рубежом, опыт работы в регионах. Затем нажимаем кнопку «Создать» и данные благополучно разместятся в таблице.



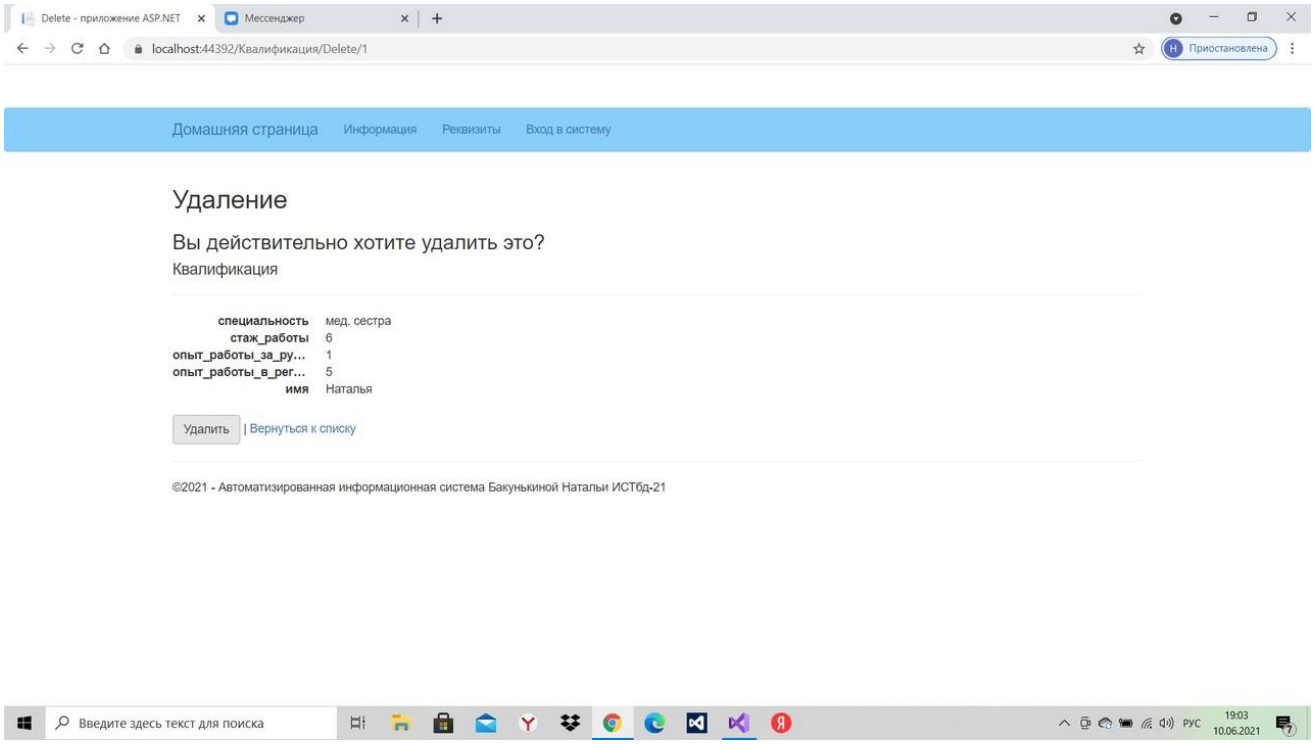
«Рисунок 14 – Страница Создать»

15. Нажимая на кнопку «редактировать» мы переходим на страницу, где редактируем данные в таблице. В таблице «Квалификация» мы редактируем данные для столбцов: специальность, стаж работы, опыт работы за рубежом, опыт работы в регионах. Затем нажимаем кнопку «Сохранить» и обновлённые данные благополучно разместятся в таблице.



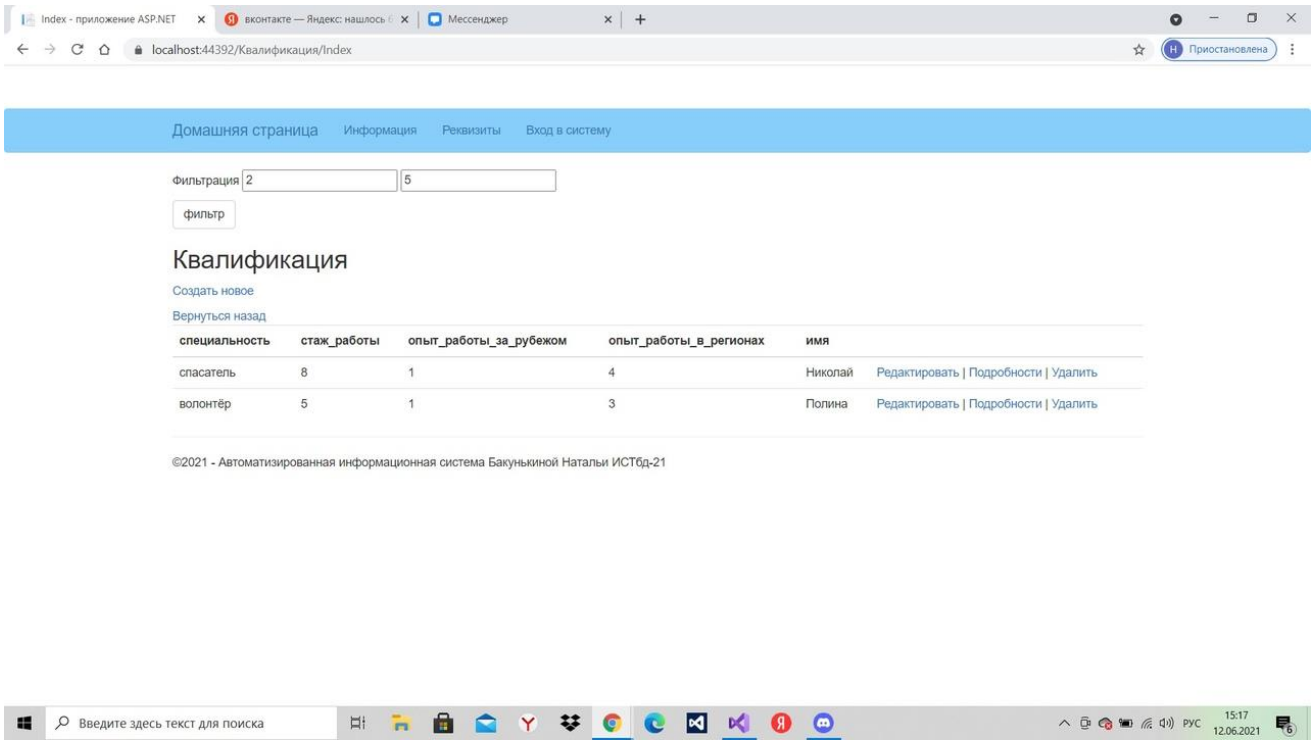
«Рисунок 15 – Страница Редактировать»

16. Нажимая на кнопку «удалить» мы переходим на страницу, где удаляем данные из таблицы. В таблице «Квалификация» мы удаляем данные для столбцов: специальность, стаж работы, опыт работы за рубежом, опыт работы в регионах. Затем нажимаем кнопку «удалить» и даннве успешно удаляться из таблицы.



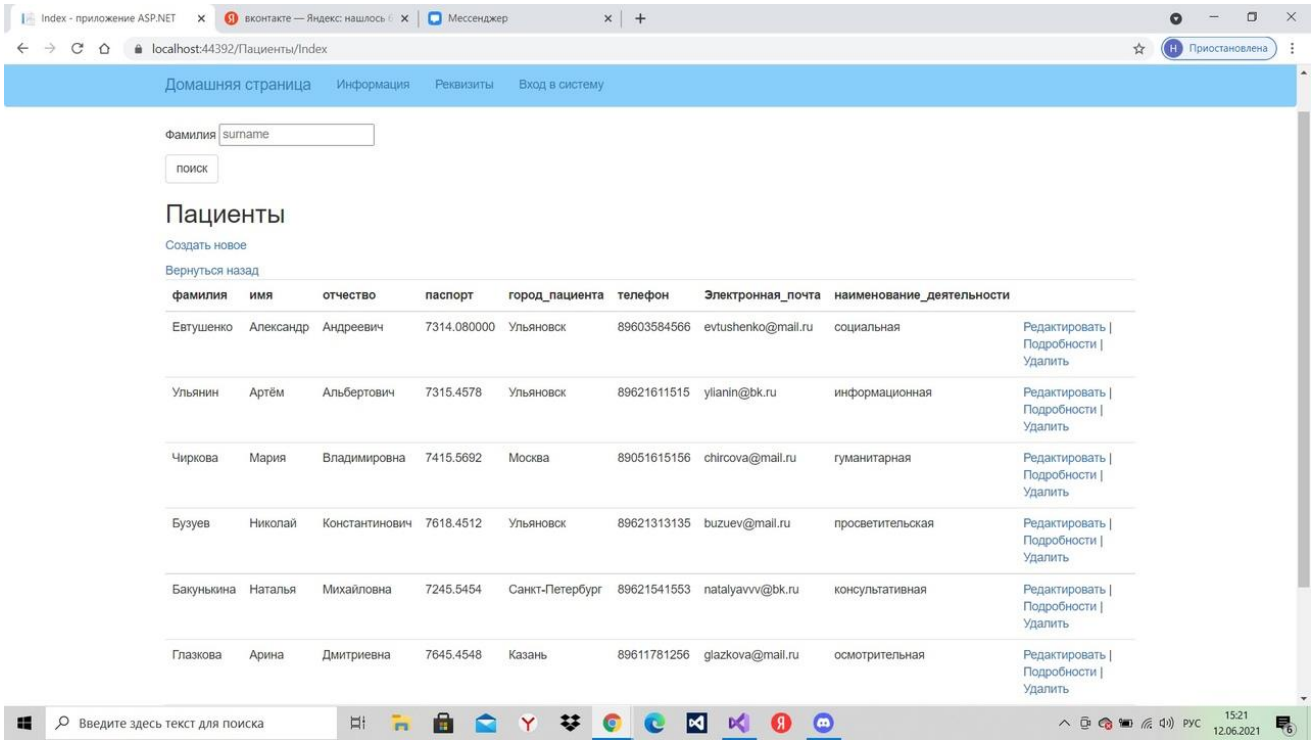
«Рисунок 16 – Страница Удаление»

17. В таблице «Квалификация» также применяется операция фильтрации. Она сортирует опыт работы в регионах, между установленным нами диапазоном, не включая его. В первое окошко мы вписываем первое значение диапозона, во второе окошко второе значение, после этого мы нажимаем на кнопку «Фильтр» и получаем отсортированный опыт работы в регионах.



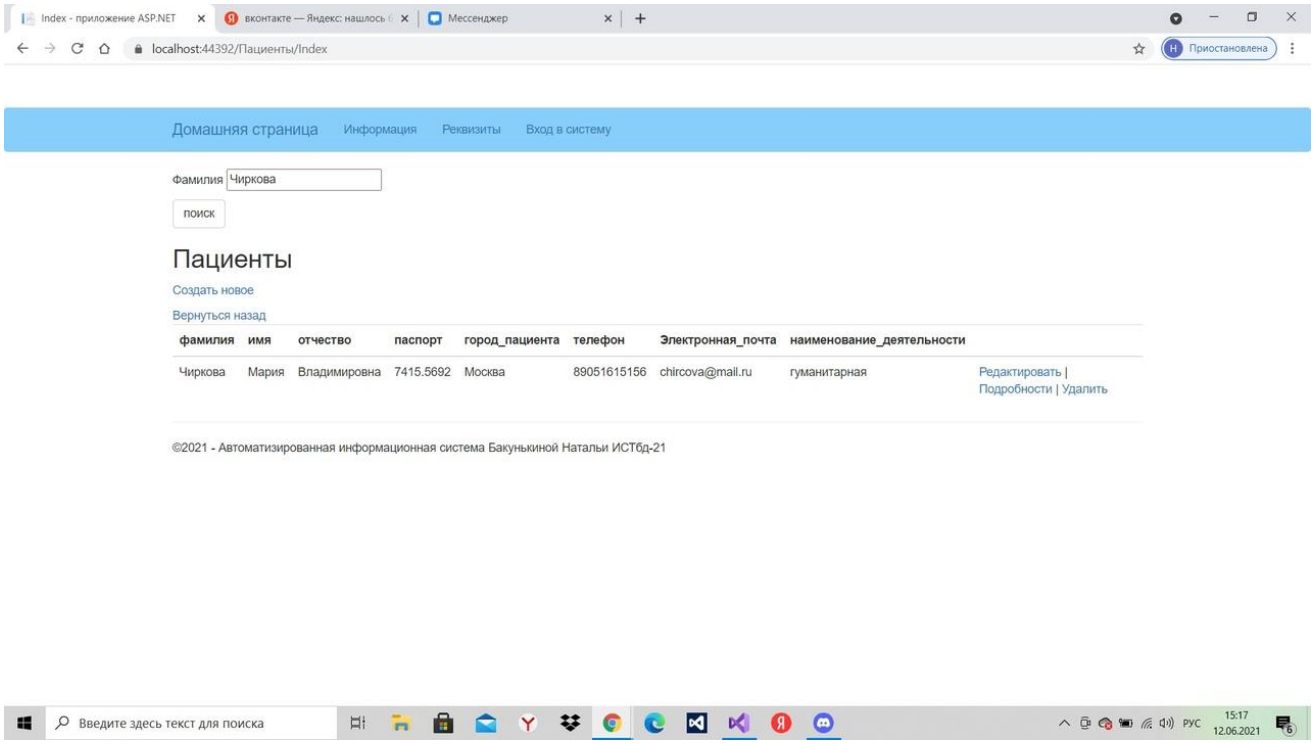
«Рисунок 17 – Страница фильтрации»

18. В таблице пациенты применяется просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Эти действия применяются для столбцов: фамилия, имя, отчество, паспорт, город пациента, телефон, электронная почта, наименование деятельности.



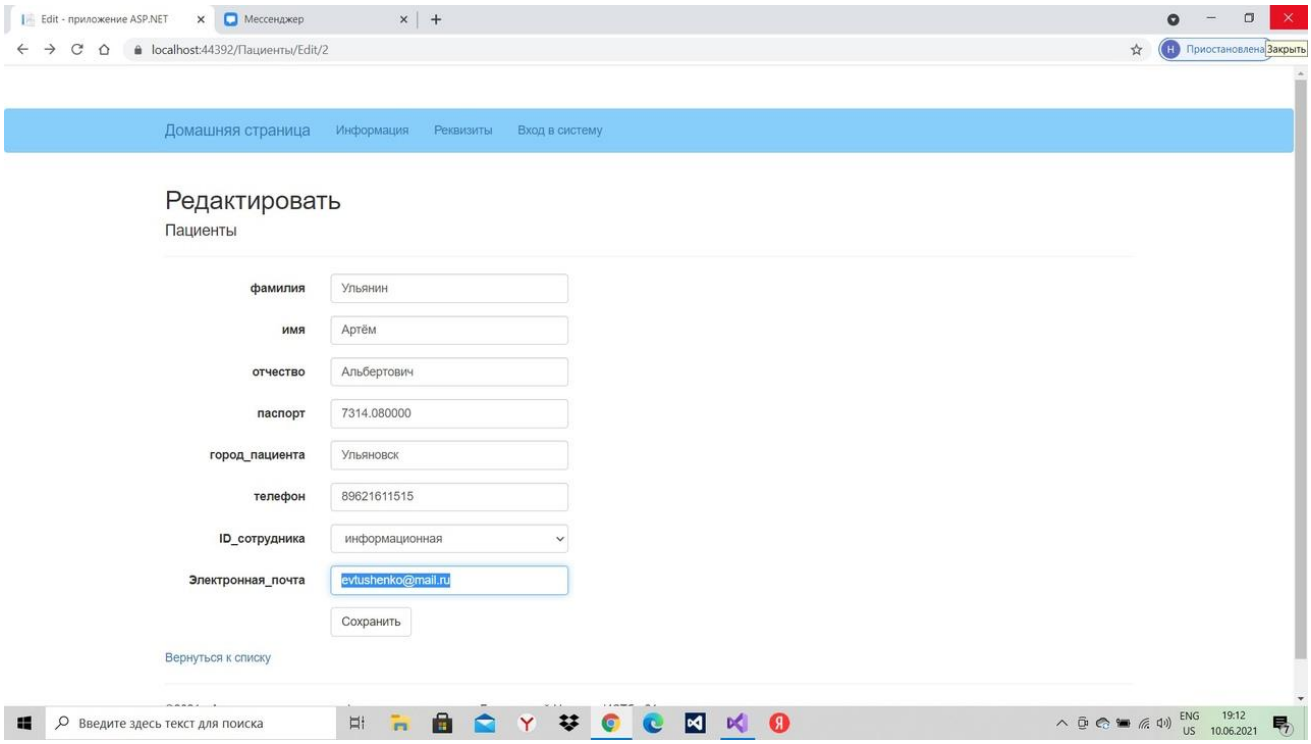
«Рисунок 18 – Страница Пациенты»

19. В таблице «Пациенты» также применяется операция поиска по фамилии. В окошко мы вводим фамилию пациента, затем нажимаем на кнопку «Поиск» и получаем данные пациента с фамилией, которую мы искали.



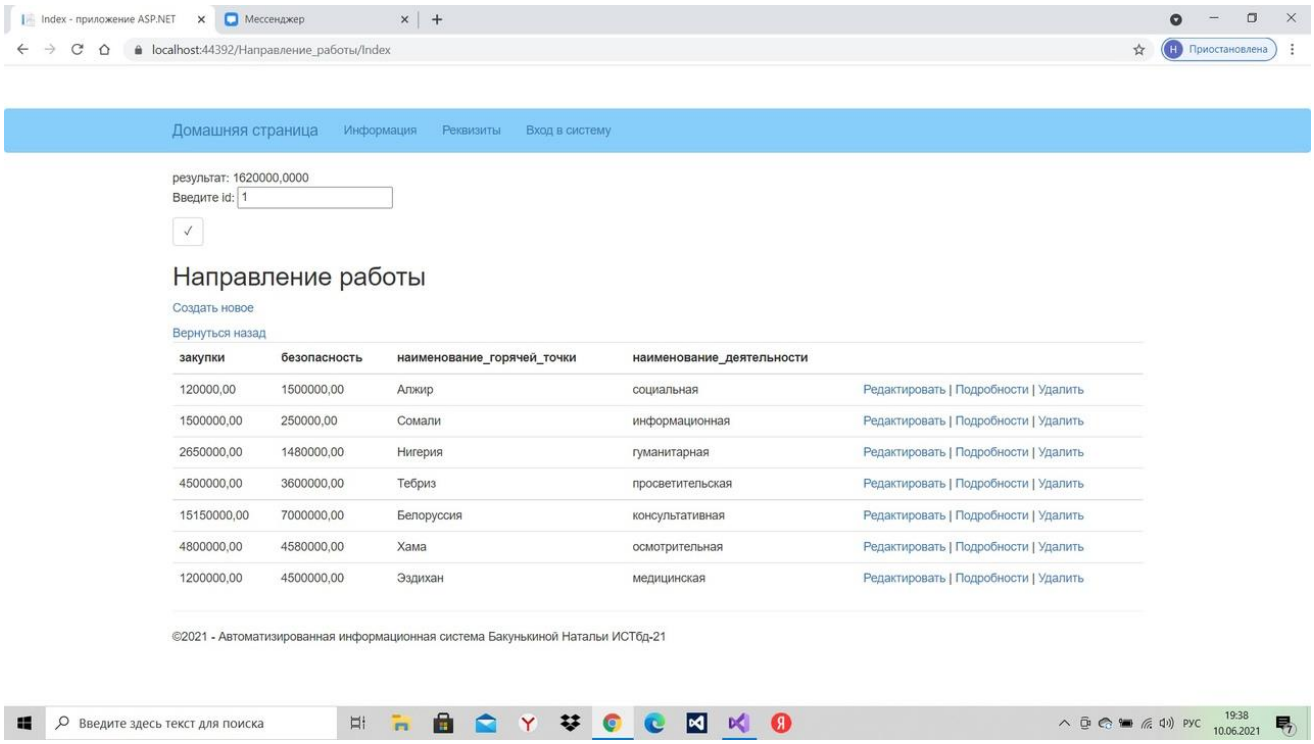
«Рисунок 19 – Страница поиска»

20. При редактировании данных, если чьи-то данные паспорта, телефона, электронной почты, которые присутствуют в таблице, присвоить другому пациенту, тогда произойдёт ошибка, так как эти данные уникальные. Система нас перенаправит на страницу ошибки, и выдаст сообщение «Повторные данные».



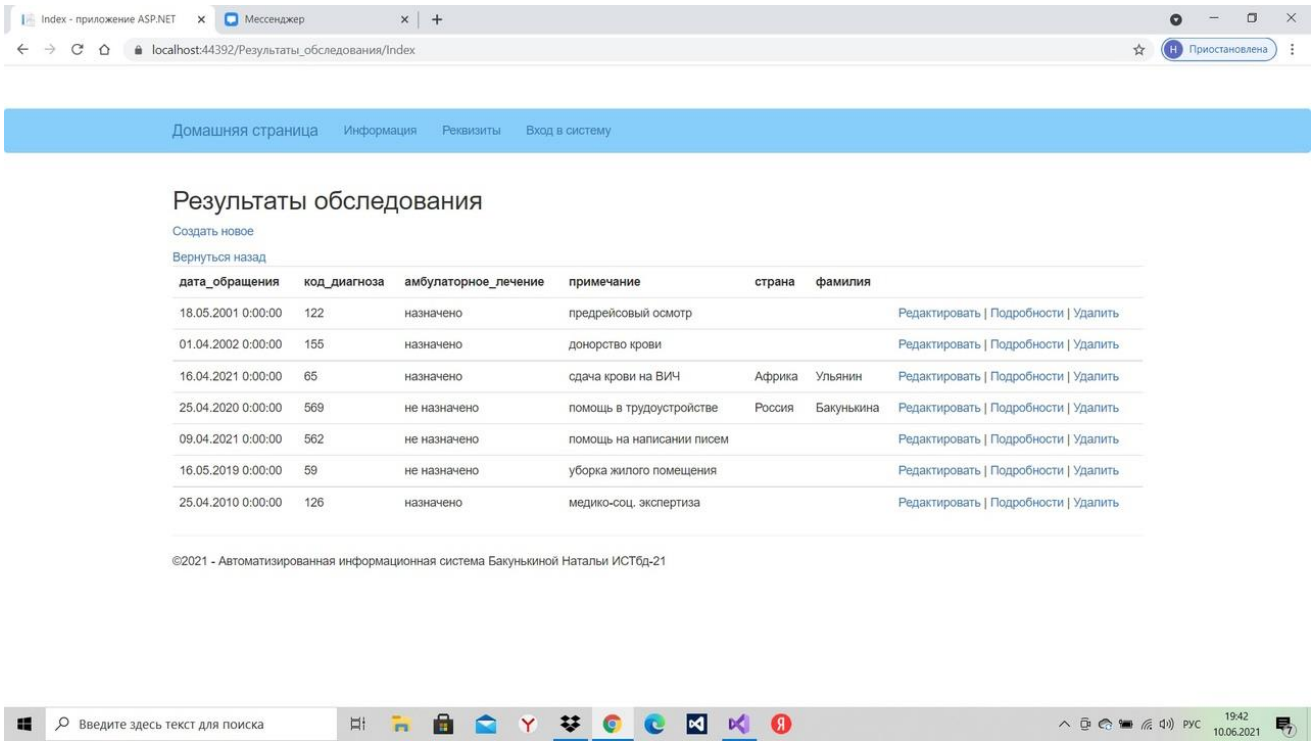
«Рисунок 20 – Страница Пациенты»

21. В таблице «Направление работы» применяется просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Эти действия применяются для столбцов: закупки, безопасность, наименование горячей точки, наименование деятельности. Также присутствует функция, в которой отображается результат в виде общей суммы закупок и безопасности. Для использования функции необходимо в окошко вписать номер строки, к которой вы хотите применить её, затем функция выдаст результат суммы.



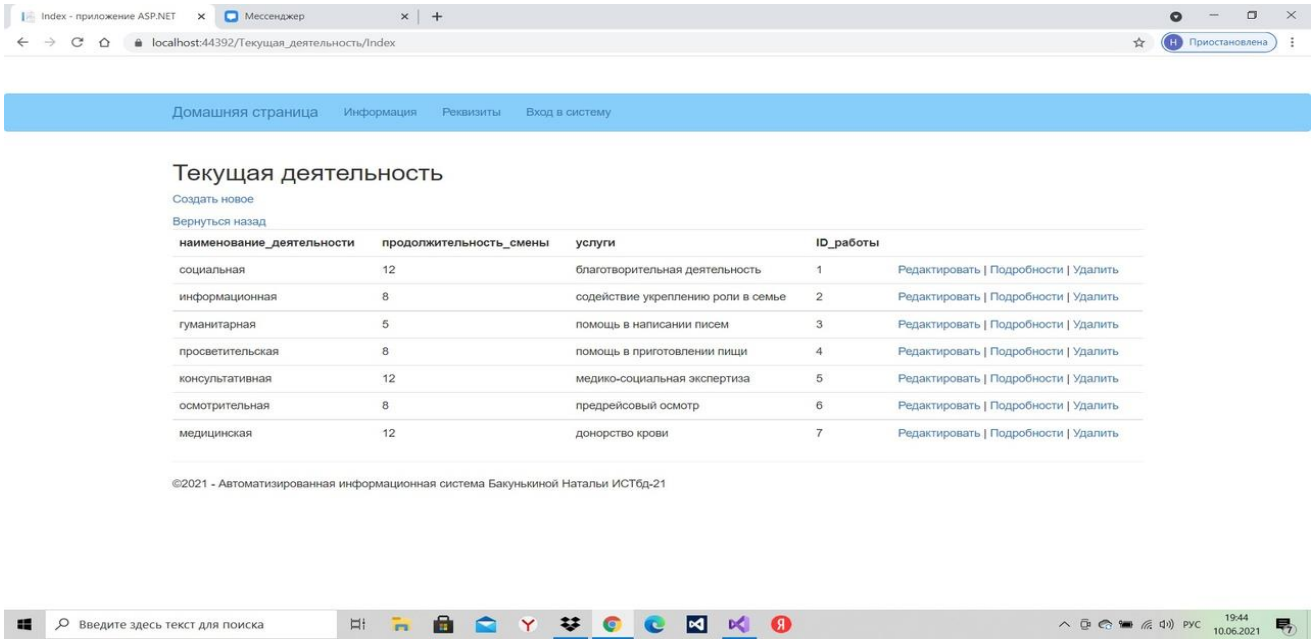
«Рисунок 21 – Страница Направление работы»

22. В таблице «Результаты обследования» применяется просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Эти действия применяются для столбцов: дата обращения, код диагноза, амбулаторное лечение, примечание, страна, фамилия.



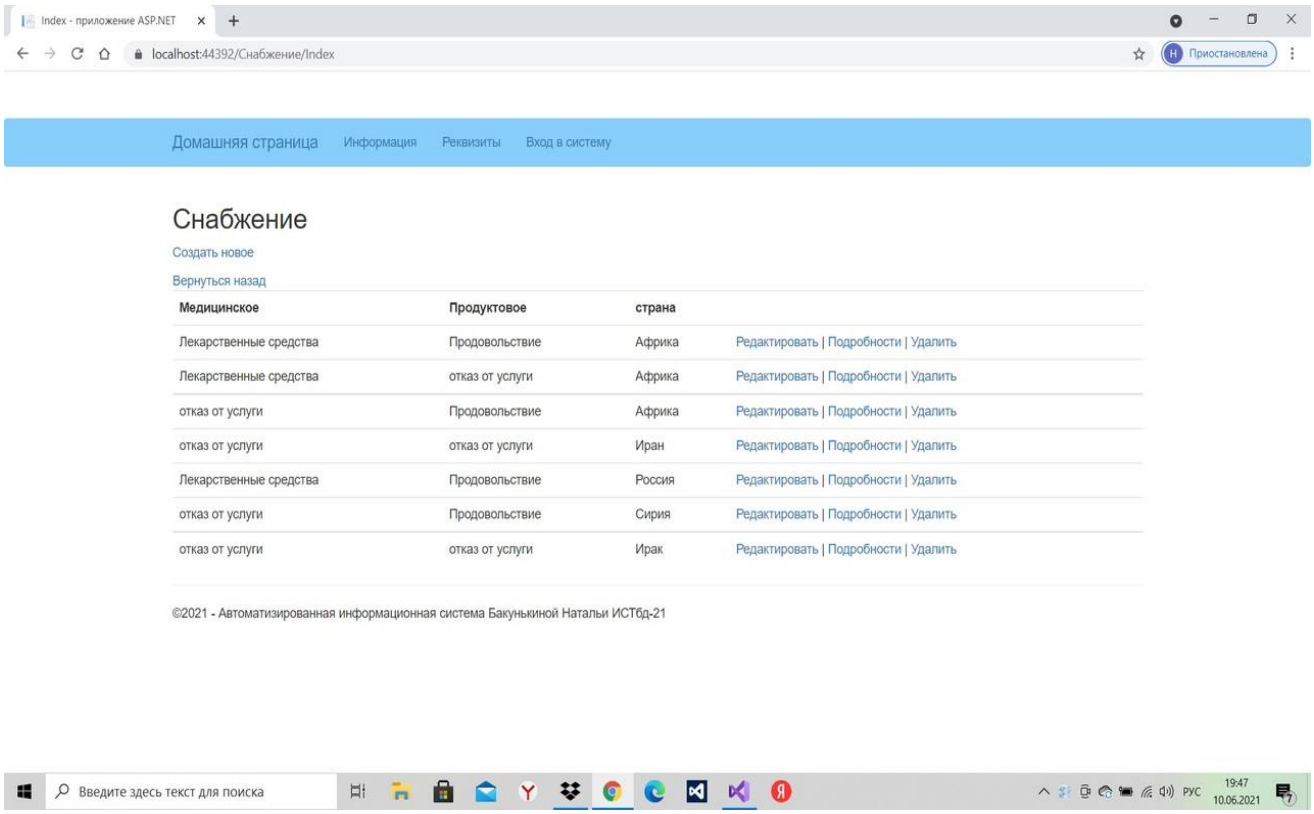
«Рисунок 22 – Страница Результаты обследования»

23. В таблице «Текущая деятельность» применяется просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Эти действия применяются для столбцов: наименование деятельности, продолжительность смены, услуги, ID_работы.



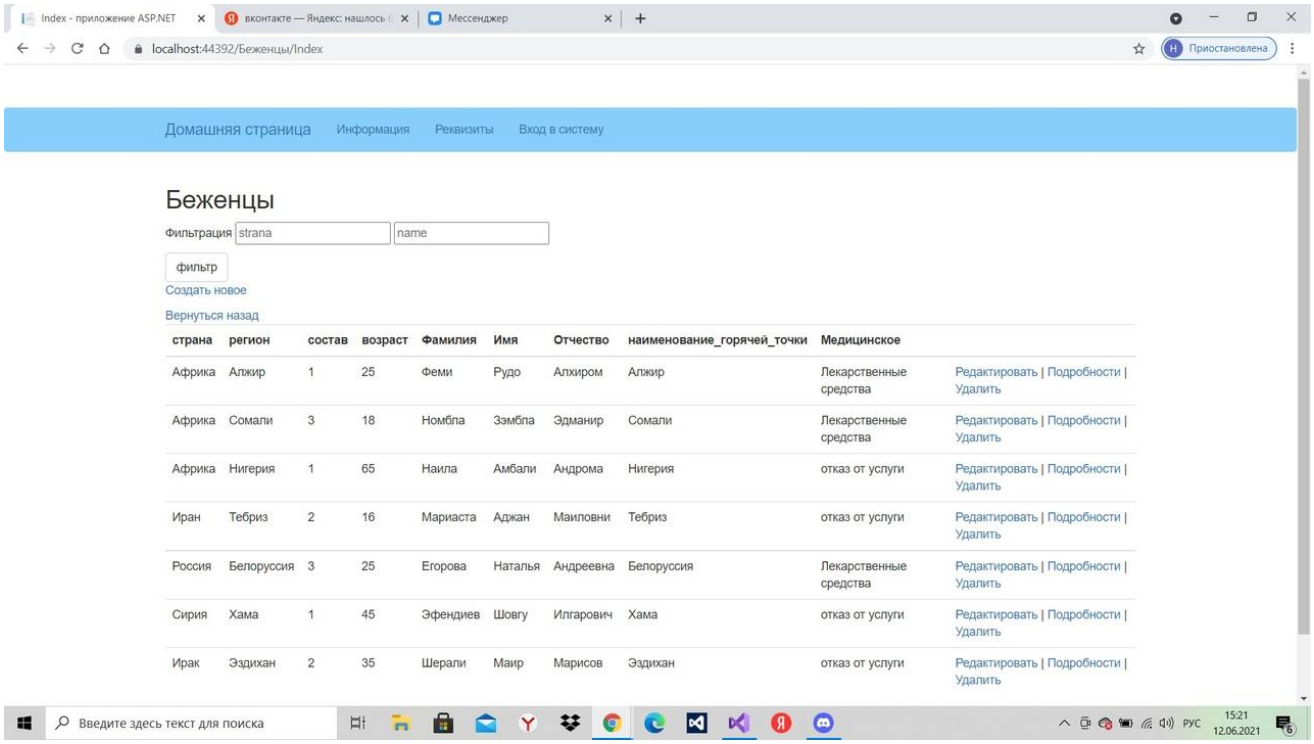
«Рисунок 23 – Страница Текущая деятельность»

24. В таблице «Снабжение» применяется просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Эти действия применяются для столбцов: медицинское, продуктовое, страна.



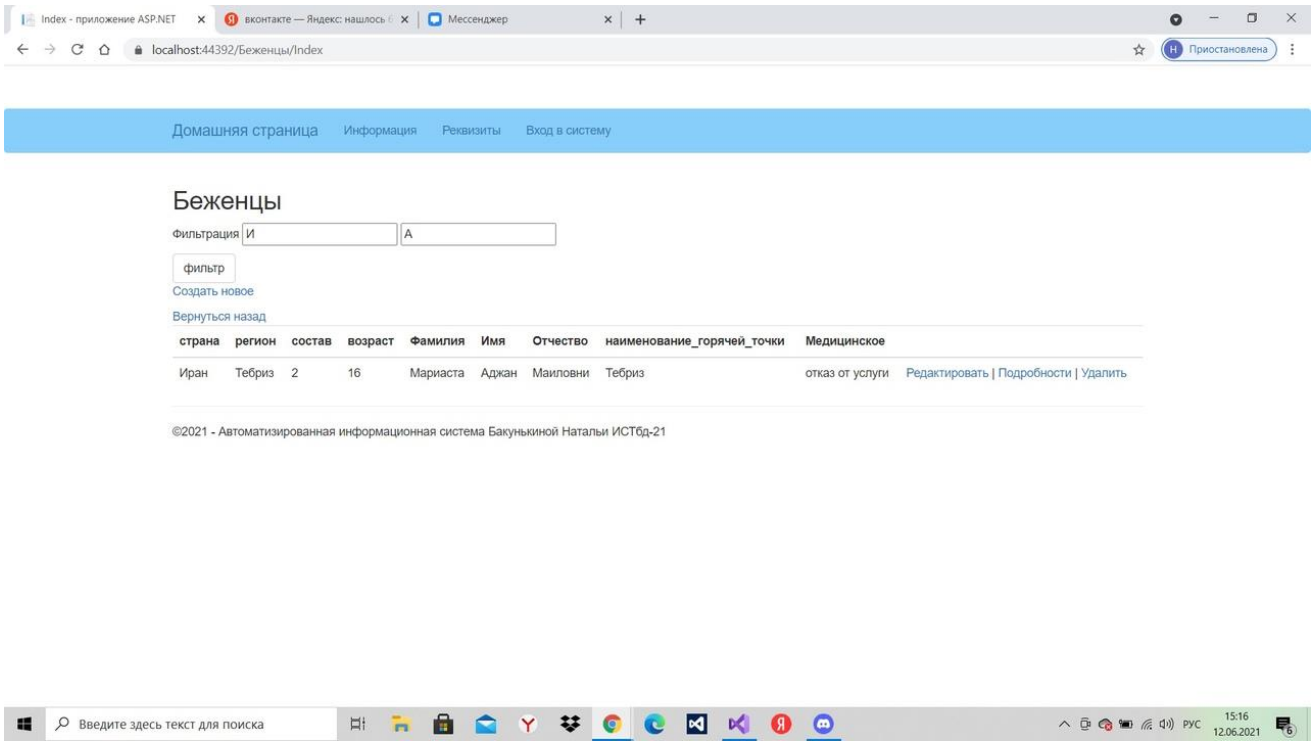
«Рисунок 24 – Страница Снабжение»

25. В таблице «Беженцы» применяется просмотр данных в табличном виде, ввод новых записей, редактирование записей – в виде бланка, удаление записей. Эти действия применяются для столбцов: страна, регион, состав, возраст, фамилия, имя, отчество, наименование горячей точки, медицинское.



«Рисунок 25 – Страница Беженцы»

26. В таблице «Беженцы» также применяется операция фильтрации. Она сортирует по первой букве страны и имени. В первое окошко мы вписываем первую букву страны, во второе окошко первую букву имени, после этого мы нажимаем на кнопку «Фильтр» и получаем отсортированные данные.



«Рисунок 26 – Страница фильтрации»

5.3 Исключительные ситуации

1. При попытке входа в систему незарегистрированого пользователя, система выдаст сообщение: «Пользователя с таким логином и паролем нет». Чтобы это устранить, необходимо зарегистрироваться в системе.
2. Если пользователь уже зарегистрирован в системе и пытается снова зарегистрироваться под тем же логином и паролем, система выдаст ему сообщение: «Пользователь с таким логином уже существует». При такой ситуации нужно осуществить вход на сайт без повторной регистрации.
3. При попытке редактирования таблицы «Пациенты» ввести повторные данные паспорта, телефона, электронной почты других пациентов невозмож-

но, так как эти данные являются уникальными для каждого пациента. Система перенаправит пользователя на страницу ошибки и выдаст сообщение «Повторные данные». Чтобы избежать ситуацию, необходимо не вводить повторные данные.

4. При попытке удалить сотрудника из системы, необходимо сначала удалить его данные из таблиц «Квалификация» и «Направление работы», только после этого сотрудник может быть удалён.
5. При попытке удалить данные пациента из таблицы «Текущая деятельность», необходимо сначала удалить его данные из таблицы «Пациенты», только после этого пациент может быть удалён. Иначе пользователя перенаправит на страницу ошибки.
6. При попытке удалить пациента из системы, необходимо сначала удалить его данные из таблицы «Результаты обследования», только после этого пациент может быть удалён. Иначе пользователя перенаправит на страницу ошибки.
7. При попытке удалить данные беженца из таблицы «Командировки», необходимо сначала удалить его данные из таблицы «Беженцы», только после этого беженец может быть удалён. Иначе пользователя перенаправит на страницу ошибки.
8. При попытке удалить беженца из системы, необходимо сначала удалить его данные из таблиц «Результаты обследования» и «Снабжение», только после этого беженец может быть удалён. Иначе пользователя перенаправит на страницу ошибки.
9. При попытке удалить данные из таблицы «Направление работы», необходимо сначала удалить его данные из таблиц «Текущая деятельность» и «Командировки», только после этого данные могут быть удалены. Иначе пользователя перенаправит на страницу ошибки.
10. Если в хранимую процедуру в таблице «Командировки» ввести некорректные данные, тогда система перенаправит пользователя на страницу ошибки и выдаст сообщение «Не является максимальным количеством выживших».

11. Если в хранимую процедуру в таблице «Сотрудники» ввести некорректные данные, тогда система перенаправит пользователя на страницу ошибки и выдаст сообщение «Сотрудник не найден».

Список использованных источников

1. Зябкин, А. И. К вопросу о правовой природе Международного Комитета Красного Креста / А. И. Зябкин. – Текст : электронный // Евразийский юридический журнал : электронный журнал. – URL: https://eurasialaw.ru/. – Дата публикации: 18 ноября 2019.
2. Майкрософт: сайт. – Амстердам, 2004. – URL: https://www.microsoft.com/ru-ru/ (дата обращения: 23.05.2021). – Текст : электронный.
3. Международный комитет Красного Креста: сайт. – Москва, 2004. – URL: https://www.icrc.org/ru/ (дата обращения: 23.05.2021). – Текст : электронный.
4. Основополагающие принципы Красного Креста: Комментарий Ж. Пикте - М.: МККК, 1997. –7 с.
5. Уроки по C# и платформе .NET Framework. – Санкт-Петербург, 2011. – URL: https://professorweb.ru (дата обращения: 23.05.2021). – Текст : электронный.

Приложение А. Исходные тексты программных модулей

UserContext

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace Красный_крест.Models
{
    public class UserContext : DbContext
    {
        public UserContext() :
            base("Красный Крест")
        { }

        public DbSet<User> Users { get; set; }
    }
}
```

User

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

namespace Красный_крест.Models
{
    public class User
    {
        public int Id { get; set; }
        public string Email { get; set; }
        public string Пароль { get; set; }
        public string Role { get; set; }
    }
}
```

Models

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

namespace Красный_крест.Models
{
    public class LoginModel
    {
        public string Пользователь { get; set; }
        public string Пароль { get; set; }
    }
    public class RegisterModel
    {
        public string Пользователь { get; set; }
        public string Пароль { get; set; }
        public string Role { get; set; }
    }
}
```

AccountController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;
using Красный_крест.Models;

namespace Красный_крест.Controllers
{
    public class AccountController : Controller
    {
        public ActionResult Login()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Login(LoginModel model)
        {
            // поиск пользователя в бд
            using (UserContext db = new UserContext())
                try
                {
                    var user = db.Users.Any(u => u.Email == model.Пользователь && u.Пароль ==
model.Пароль && u.Роль == "user");
                    var admin = db.Users.Any(u => u.Email == model.Пользователь && u.Пароль
== model.Пароль && u.Роль == "admin");

                    if (admin)
                    {
                        return RedirectToAction("Admin", "Role");
                    }
                    else if (user)
                    {
                        return RedirectToAction("Users", "Role");
                    }
                }
                catch { }
            ModelState.AddModelError("", "Роль не найдена или пользователя с таким логином и
паролем нет");
            return View(model);
        }

        public ActionResult Register()
        {
            return View();
        }
        [HttpPost]
        public ActionResult Register(RegisterModel model)
        {
            if (ModelState.IsValid) // проверка на корректность
            {
                string s = model.Пароль;
                int i = s.Length;
                bool c = false;
                bool d = false;
                bool e = false;
                for (int a = 0; a < i; a = a + 1)
                {
                    for (int b = 48; b < 58; b++) //Проверка наличия цифр
                    {
```

```
                    if (s[a] == b)
                    {
                        c = true;
                        break;
                    }
                }
                for (int b = 65; b < 91; b = b + 1) //Проверка наличия букв A…Z
                {
                    if (s[a] == b)
                    {
                        d = true;
                        break;
                    }
                }
                for (int b = 97; b < 123; b = b + 1)//Проверка наличия букв a…z
                {
                    if (s[a] == b)
                    {
                        e = true;
                        break;
                    }
                }
            }

            if (i < 8) { ModelState.AddModelError("", "Пароль должен содержать минимум 8
символов"); return View(model); }
            if (d == false) { ModelState.AddModelError("", "Пароль должен содержать за-
главные буквы верхнего регистра латинского алфавита"); return View(model); }
            if (c == false) { ModelState.AddModelError("", ""); return View(model); }
            if (e == false) { ModelState.AddModelError("", "Пароль должен содержать буквы
нижнего регистра латинского алфавита"); return View(model); }

            User user = null;
            using (UserContext db = new UserContext()) // обращаемся к контексту данных
            {
                user = db.Users.FirstOrDefault(u => u.Email == model.Пользователь); //
находим пользователя по логину
            }
            if (user == null)
            {
                // создаем нового пользователя
                using (UserContext db = new UserContext())
                {
                    db.Users.Add(new User { Email = model.Пользователь, Пароль =
model.Пароль }); // создаём user по данным, которые переданы через registermodel
                    db.SaveChanges();

                    user = db.Users.Where(u => u.Email == model.Пользователь && u.Пароль
== model.Пароль).FirstOrDefault();
                   // получаем пользователя из бд
                }
                // если пользователь добавлен в бд
                if (user != null)
                {
                    return RedirectToAction("Login", "Account");
                }
            }
            else
            {
                ModelState.AddModelError("", "Пользователь уже зарегистрирован в систе-
ме");
            }
        }

        return View(model); //возврат в представление (главная страница)
```

```
        }
    }
}
```

Role

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

namespace Красный_крест.Controllers
{
    public class RoleController : Controller
    {
        // GET: Role
        public ActionResult Admin()
        {
            return View();
        }
        public ActionResult Users()
        {
            return View();
        }
    }
}
```

HomeCotrroller

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

namespace Красный_крест.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Index()
        {
            return View();
        }
        // GET: Home
        public ActionResult About()
        {
            return View();
        }
        // GET: Home
        public ActionResult Contact()
        {
            return View();
        }
        public ActionResult Entrance()
        {
            return View();
        }

        // GET: Home
        public ActionResult Service()
        {
            return View();
        }
        public ActionResult Slusba()
```

```
        {
            return View();
        }
    }
}
```

БеженцыController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using Красный_крест;

namespace Красный_крест.Controllers
{
    public class БеженцыController : Controller
    {
        private Красный_КрестEntities1 db = new Красный_КрестEntities1();
        [HttpPost]
        public ActionResult Index(string strana, string name)
        {
            IQueryable<Беженцы> per = db.Беженцы.Include(p => p.Командировки).Include(m =>
m.Снабжение);
            if (strana != null)
            {
                per = per.AsEnumerable().Where(p => p.страна[0] == stra-
na[0]).AsQueryable().Where(p => p.Имя[0] == name[0]);
            }
            ViewBag.k = "";
            return View(per.ToList());
        }        // GET: Беженцы
        public ActionResult Index()
        {
            var беженцы = db.Беженцы.Include(б => б.Командировки).Include(б => б.Снабжение);
            return View(беженцы.ToList());
        }

            // GET: Беженцы/Details/5
            public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Беженцы беженцы = db.Беженцы.Find(id);
            if (беженцы == null)
            {
                return HttpNotFound();
            }
            return View(беженцы);
        }

        // GET: Беженцы/Create
        public ActionResult Create()
        {
            ViewBag.ID_сотрудника = new SelectList(db.Командировки, "ID_работы", "наименова-
ние_горячей_точки");
            ViewBag.ID_личности = new SelectList(db.Снабжение, "ID_личности", "Медицинское");
            return View();
        }
```

```
        // POST: Беженцы/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "стра-
на,регион,состав,возраст,ID_личности,ID_сотрудника,Фамилия,Имя,Отчество")] Беженцы беженцы)
        {
            if (ModelState.IsValid)
            {
                db.Беженцы.Add(беженцы);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.ID_сотрудника = new SelectList(db.Командировки, "ID_работы", "наименова-
ние_горячей_точки", беженцы.ID_сотрудника);
            ViewBag.ID_личности = new SelectList(db.Снабжение, "ID_личности", "Медицинское",
беженцы.ID_личности);
            return View(беженцы);
        }

        // GET: Беженцы/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Беженцы беженцы = db.Беженцы.Find(id);
            if (беженцы == null)
            {
                return HttpNotFound();
            }
            ViewBag.ID_сотрудника = new SelectList(db.Командировки, "ID_работы", "наименова-
ние_горячей_точки", беженцы.ID_сотрудника);
            ViewBag.ID_личности = new SelectList(db.Снабжение, "ID_личности", "Медицинское",
беженцы.ID_личности);
            return View(беженцы);
        }

        // POST: Беженцы/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "стра-
на,регион,состав,возраст,ID_личности,ID_сотрудника,Фамилия,Имя,Отчество")] Беженцы беженцы)
        {
            if (ModelState.IsValid)
            {
                db.Entry(беженцы).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.ID_сотрудника = new SelectList(db.Командировки, "ID_работы", "наименова-
ние_горячей_точки", беженцы.ID_сотрудника);
            ViewBag.ID_личности = new SelectList(db.Снабжение, "ID_личности", "Медицинское",
беженцы.ID_личности);
            return View(беженцы);
        }
```

```
        // GET: Беженцы/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Беженцы беженцы = db.Беженцы.Find(id);
            if (беженцы == null)
            {
                return HttpNotFound();
            }
            return View(беженцы);
        }

        // POST: Беженцы/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Беженцы беженцы = db.Беженцы.Find(id);
            if (ModelState.IsValid)
            {
                try
                {
                    db.Беженцы.Remove(беженцы);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    string text = "Ошибка";
                    ViewData["text"] = text;
                    return View("Error");
                }
            }
            return View(беженцы);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

КвалификацияController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Красный_крест;

namespace Красный_крест.Controllers
{
    public class КвалификацияController : Controller
```

```
    {
        private Красный_КрестEntities1 db = new Красный_КрестEntities1();
        [HttpPost]
        public ActionResult Index(int? a, int? b)
        {
            var квалификация = db.Квалификация.Include(к => к.Сотрудники).Where(с =>
с.опыт_работы_в_регионах > a && с.опыт_работы_в_регионах < b);
            ViewBag.d = "";
            return View(квалификация.ToList());
        }
        // GET: Квалификация
        public ActionResult Index()
        {
            var квалификация = db.Квалификация.Include(к => к.Сотрудники);
            return View(квалификация.ToList());
        }

        // GET: Квалификация/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Квалификация квалификация = db.Квалификация.Find(id);
            if (квалификация == null)
            {
                return HttpNotFound();
            }
            return View(квалификация);
        }

        // GET: Квалификация/Create
        public ActionResult Create()
        {
            ViewBag.ID_сотрудника = new SelectList(db.Сотрудники, "ID_сотрудника", "имя");
            return View();
        }

        // POST: Квалификация/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "специаль-
ность,стаж_работы,опыт_работы_за_рубежом,опыт_работы_в_регионах,ID_сотрудника")] Квалификация
квалификация)
        {
            if (ModelState.IsValid)
            {
                db.Квалификация.Add(квалификация);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.ID_сотрудника = new SelectList(db.Сотрудники, "ID_сотрудника", "имя",
квалификация.ID_сотрудника);
            return View(квалификация);
        }

        // GET: Квалификация/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
```

```
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Квалификация квалификация = db.Квалификация.Find(id);
            if (квалификация == null)
            {
                return HttpNotFound();
            }
            ViewBag.ID_сотрудника = new SelectList(db.Сотрудники, "ID_сотрудника", "имя",
квалификация.ID_сотрудника);
            return View(квалификация);
        }

        // POST: Квалификация/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "специаль-
ность,стаж_работы,опыт_работы_за_рубежом,опыт_работы_в_регионах,ID_сотрудника")] Квалификация
квалификация)
        {
            if (ModelState.IsValid)
            {
                db.Entry(квалификация).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.ID_сотрудника = new SelectList(db.Сотрудники, "ID_сотрудника", "имя",
квалификация.ID_сотрудника);
            return View(квалификация);
        }

        // GET: Квалификация/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Квалификация квалификация = db.Квалификация.Find(id);
            if (квалификация == null)
            {
                return HttpNotFound();
            }
            return View(квалификация);
        }

        // POST: Квалификация/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Квалификация квалификация = db.Квалификация.Find(id);
            db.Квалификация.Remove(квалификация);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
```

```
            }
            base.Dispose(disposing);
        }
    }
}
```

КомандировкиController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Data.SqlClient;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Красный_крест;

namespace Красный_крест.Controllers
{
    public class КомандировкиController : Controller
    {
        private Красный_КрестEntities1 db = new Красный_КрестEntities1();
        [HttpPost]
        public ActionResult Index(string tochka, decimal? newZarplata)
        {
            var командировки = db.Командировки.Include(с => с.Направление_работы);
            var check = командировки.Any(х => х.наименование_горячей_точки == tochka);
            if (check == true)
            {
                SqlParameter a =
                  new SqlParameter("@tochka", tochka);
                SqlParameter b =
                new SqlParameter("@newZarplata", newZarplata);
                db.Database.ExecuteSqlCommand("EXEC procedura_2 @tochka, @newZarplata", a,
b);
                return View(командировки.ToList());
            }
            else
            {
                string text = "Не является максимальным количеством выживших";
                ViewData["text"] = text;
                return View("Error");
            }

        }
        // GET: Командировки
        public ActionResult Index()
        {
            var командировки = db.Командировки.Include(к => к.Направление_работы);
            return View(командировки.ToList());
        }

        // GET: Командировки/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Командировки командировки = db.Командировки.Find(id);
            if (командировки == null)
            {
                return HttpNotFound();
```

```
            }
            return View(командировки);
        }

        // GET: Командировки/Create
        public ActionResult Create()
        {
            ViewBag.ID_работы = new SelectList(db.Направление_работы, "ID_работы",
"ID_работы");
            return View();
        }

        // POST: Командировки/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "наименова-
ние_горячей_точки,дата_командировки,количество_выжевших,ID_работы")] Командировки командиров-
ки)
        {
            if (ModelState.IsValid)
            {
                db.Командировки.Add(командировки);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.ID_работы = new SelectList(db.Направление_работы, "ID_работы",
"ID_работы", командировки.ID_работы);
            return View(командировки);
        }

        // GET: Командировки/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Командировки командировки = db.Командировки.Find(id);
            if (командировки == null)
            {
                return HttpNotFound();
            }
            ViewBag.ID_работы = new SelectList(db.Направление_работы, "ID_работы",
"ID_работы", командировки.ID_работы);
            return View(командировки);
        }

        // POST: Командировки/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "наименова-
ние_горячей_точки,дата_командировки,количество_выжевших,ID_работы")] Командировки командиров-
ки)
        {
            if (ModelState.IsValid)
            {
                db.Entry(командировки).State = EntityState.Modified;
                db.SaveChanges();
```

```
                return RedirectToAction("Index");
            }
            ViewBag.ID_работы = new SelectList(db.Направление_работы, "ID_работы",
"ID_работы", командировки.ID_работы);
            return View(командировки);
        }

        // GET: Командировки/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Командировки командировки = db.Командировки.Find(id);
            if (командировки == null)
            {
                return HttpNotFound();
            }
            return View(командировки);
        }

        // POST: Командировки/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Командировки командировки = db.Командировки.Find(id);
            if (ModelState.IsValid)
            {
                try
                {
                    db.Командировки.Remove(командировки);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    string text = "Ошибка";
                    ViewData["text"] = text;
                    return View("Error");
                }
            }
            return View(командировки);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Направление_работыController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
```

```
using System.Web;
using System.Web.Mvc;
using Красный_крест;

namespace Красный_крест.Controllers
{
    public class Направление_работыController : Controller
    {
        private Красный_КрестEntities1 db = new Красный_КрестEntities1();

        // GET: Направление_работы
        public ActionResult Index(int ?id)
        {
            decimal? a = db.funkcia1(id).First().результат;
            ViewBag.a = "результат: " + a;
            var направление_работы = db.Направление_работы.Include(н =>
н.Командировки).Include(н => н.Текущая_деятельность);
            return View(направление_работы.ToList());
        }

        // GET: Направление_работы/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Направление_работы направление_работы = db.Направление_работы.Find(id);
            if (направление_работы == null)
            {
                return HttpNotFound();
            }
            return View(направление_работы);
        }

        // GET: Направление_работы/Create
        public ActionResult Create()
        {
            ViewBag.ID_работы = new SelectList(db.Командировки, "ID_работы", "наименова-
ние_горячей_точки");
            ViewBag.ID_работы = new SelectList(db.Текущая_деятельность, "ID_работы", "наиме-
нование_деятельности");
            return View();
        }

        // POST: Направление_работы/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "закупки,безопасность,ID_работы")] Направ-
ление_работы направление_работы)
        {
            if (ModelState.IsValid)
            {
                db.Направление_работы.Add(направление_работы);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.ID_работы = new SelectList(db.Командировки, "ID_работы", "наименова-
ние_горячей_точки", направление_работы.ID_работы);
            ViewBag.ID_работы = new SelectList(db.Текущая_деятельность, "ID_работы", "наиме-
нование_деятельности", направление_работы.ID_работы);
```

```
            return View(направление_работы);
        }

        // GET: Направление_работы/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Направление_работы направление_работы = db.Направление_работы.Find(id);
            if (направление_работы == null)
            {
                return HttpNotFound();
            }
            ViewBag.ID_работы = new SelectList(db.Командировки, "ID_работы", "наименова-
ние_горячей_точки", направление_работы.ID_работы);
            ViewBag.ID_работы = new SelectList(db.Текущая_деятельность, "ID_работы", "наиме-
нование_деятельности", направление_работы.ID_работы);
            return View(направление_работы);
        }

        // POST: Направление_работы/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "закупки,безопасность,ID_работы")] Направле-
ние_работы направление_работы)
        {
            if (ModelState.IsValid)
            {
                db.Entry(направление_работы).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.ID_работы = new SelectList(db.Командировки, "ID_работы", "наименова-
ние_горячей_точки", направление_работы.ID_работы);
            ViewBag.ID_работы = new SelectList(db.Текущая_деятельность, "ID_работы", "наиме-
нование_деятельности", направление_работы.ID_работы);
            return View(направление_работы);
        }

        // GET: Направление_работы/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Направление_работы направление_работы = db.Направление_работы.Find(id);
            if (направление_работы == null)
            {
                return HttpNotFound();
            }
            return View(направление_работы);
        }

        // POST: Направление_работы/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Направление_работы направление_работы = db.Направление_работы.Find(id);
```

```
            if (ModelState.IsValid)
            {
                try
                {
                    db.Направление_работы.Remove(направление_работы);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    string text = "Ошибка";
                    ViewData["text"] = text;
                    return View("Error");
                }
            }
            return View(направление_работы);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

ПациентыController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Красный_крест;

namespace Красный_крест.Controllers
{
    public class ПациентыController : Controller
    {
        private Красный_КрестEntities1 db = new Красный_КрестEntities1();
        [HttpPost]
        public ActionResult Index(string фамилия)
        {
            var пациенты = db.Пациенты.Include(п => п.Текущая_деятельность).Where(с =>
с.фамилия == фамилия);
            ViewBag.b = "";
            return View(пациенты.ToList());
        }

        // GET: Пациенты
        public ActionResult Index()
        {
            var пациенты = db.Пациенты.Include(п => п.Текущая_деятельность);
            return View(пациенты.ToList());
        }

        // GET: Пациенты/Details/5
        public ActionResult Details(int? id)
        {
```

```
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Пациенты пациенты = db.Пациенты.Find(id);
            if (пациенты == null)
            {
                return HttpNotFound();
            }
            return View(пациенты);
        }

        // GET: Пациенты/Create
        public ActionResult Create()
        {
            ViewBag.ID_сотрудника = new SelectList(db.Текущая_деятельность, "ID_работы",
"наименование_деятельности");
            return View();
        }

        // POST: Пациенты/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "фами-
лия,имя,отчество,паспорт,город_пациента,телефон,ID_сотрудника,ID_пациента,Электронная_почта")
] Пациенты пациенты)
        {
            if (ModelState.IsValid)
            {
                db.Пациенты.Add(пациенты);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.ID_сотрудника = new SelectList(db.Текущая_деятельность, "ID_работы",
"наименование_деятельности", пациенты.ID_сотрудника);
            return View(пациенты);
        }

        // GET: Пациенты/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Пациенты пациенты = db.Пациенты.Find(id);
            if (пациенты == null)
            {
                return HttpNotFound();
            }
            ViewBag.ID_сотрудника = new SelectList(db.Текущая_деятельность, "ID_работы",
"наименование_деятельности", пациенты.ID_сотрудника);
            return View(пациенты);
        }

        // POST: Пациенты/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
```

```
        public ActionResult Edit([Bind(Include = "фами-
лия,имя,отчество,паспорт,город_пациента,телефон,ID_сотрудника,ID_пациента,Электронная_почта")
] Пациенты пациенты)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    db.Entry(пациенты).State = EntityState.Modified;
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    string text = "Повторные данные";
                    ViewData["text"] = text;
                    return View("Error");
                }
            }
            ViewBag.ID_сотрудника = new SelectList(db.Текущая_деятельность, "ID_работы",
"наименование_деятельности", пациенты.ID_сотрудника);
            return View(пациенты);
        }

        // GET: Пациенты/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Пациенты пациенты = db.Пациенты.Find(id);
            if (пациенты == null)
            {
                return HttpNotFound();
            }
            return View(пациенты);
        }

        // POST: Пациенты/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Пациенты пациенты = db.Пациенты.Find(id);
            if (ModelState.IsValid)
            {
                try
                {
                    db.Пациенты.Remove(пациенты);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    string text = "Ошибка";
                    ViewData["text"] = text;
                    return View("Error");
                }
            }
            return View(пациенты);
        }

        protected override void Dispose(bool disposing)
        {
```

```
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Результаты_обследованияController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Красный_крест;

namespace Красный_крест.Controllers
{
    public class Результаты_обследованияController : Controller
    {
        private Красный_КрестEntities1 db = new Красный_КрестEntities1();

        // GET: Результаты_обследования
        public ActionResult Index()
        {
            var результаты_обследования = db.Результаты_обследования.Include(p =>
p.Беженцы).Include(p => p.Пациенты);
            return View(результаты_обследования.ToList());
        }

        // GET: Результаты_обследования/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Результаты_обследования результаты_обследования =
db.Результаты_обследования.Find(id);
            if (результаты_обследования == null)
            {
                return HttpNotFound();
            }
            return View(результаты_обследования);
        }

        // GET: Результаты_обследования/Create
        public ActionResult Create()
        {
            ViewBag.ID_личности = new SelectList(db.Беженцы, "ID_личности", "страна");
            ViewBag.ID_пациента = new SelectList(db.Пациенты, "ID_пациента", "фамилия");
            return View();
        }

        // POST: Результаты_обследования/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
```

```
        public ActionResult Create([Bind(Include = "да-
та_обращения,код_диагноза,амбулаторное_лечение,примечание,ID_пациента,ID_личности")] Резуль-
таты_обследования результаты_обследования)
        {
            if (ModelState.IsValid)
            {
                db.Результаты_обследования.Add(результаты_обследования);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.ID_личности = new SelectList(db.Беженцы, "ID_личности", "страна", резуль-
таты_обследования.ID_личности);
            ViewBag.ID_пациента = new SelectList(db.Пациенты, "ID_пациента", "фамилия", ре-
зультаты_обследования.ID_пациента);
            return View(результаты_обследования);
        }

        // GET: Результаты_обследования/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Результаты_обследования результаты_обследования =
db.Результаты_обследования.Find(id);
            if (результаты_обследования == null)
            {
                return HttpNotFound();
            }
            ViewBag.ID_личности = new SelectList(db.Беженцы, "ID_личности", "страна", резуль-
таты_обследования.ID_личности);
            ViewBag.ID_пациента = new SelectList(db.Пациенты, "ID_пациента", "фамилия", ре-
зультаты_обследования.ID_пациента);
            return View(результаты_обследования);
        }

        // POST: Результаты_обследования/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "да-
та_обращения,код_диагноза,амбулаторное_лечение,примечание,ID_пациента,ID_личности")] Резуль-
таты_обследования результаты_обследования)
        {
            if (ModelState.IsValid)
            {
                db.Entry(результаты_обследования).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.ID_личности = new SelectList(db.Беженцы, "ID_личности", "страна", резуль-
таты_обследования.ID_личности);
            ViewBag.ID_пациента = new SelectList(db.Пациенты, "ID_пациента", "фамилия", ре-
зультаты_обследования.ID_пациента);
            return View(результаты_обследования);
        }

        // GET: Результаты_обследования/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
```

```
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Результаты_обследования результаты_обследования =
db.Результаты_обследования.Find(id);
            if (результаты_обследования == null)
            {
                return HttpNotFound();
            }
            return View(результаты_обследования);
        }

        // POST: Результаты_обследования/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Результаты_обследования результаты_обследования =
db.Результаты_обследования.Find(id);
            db.Результаты_обследования.Remove(результаты_обследования);
            db.SaveChanges();
            return RedirectToAction("Index");

            }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

СнабжениеController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Красный_крест;

namespace Красный_крест.Controllers
{
    public class СнабжениеController : Controller
    {
        private Красный_КрестEntities1 db = new Красный_КрестEntities1();

        // GET: Снабжение
        public ActionResult Index()
        {
            var снабжение = db.Снабжение.Include(с => с.Беженцы);
            return View(снабжение.ToList());
        }

        // GET: Снабжение/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
```

```
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Снабжение снабжение = db.Снабжение.Find(id);
            if (снабжение == null)
            {
                return HttpNotFound();
            }
            return View(снабжение);
        }

        // GET: Снабжение/Create
        public ActionResult Create()
        {
            ViewBag.ID_личности = new SelectList(db.Беженцы, "ID_личности", "страна");
            return View();
        }

        // POST: Снабжение/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "Медицинское,Продуктовое,ID_личности")]
Снабжение снабжение)
        {
            if (ModelState.IsValid)
            {
                db.Снабжение.Add(снабжение);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.ID_личности = new SelectList(db.Беженцы, "ID_личности", "страна", снабже-
ние.ID_личности);
            return View(снабжение);
        }

        // GET: Снабжение/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Снабжение снабжение = db.Снабжение.Find(id);
            if (снабжение == null)
            {
                return HttpNotFound();
            }
            ViewBag.ID_личности = new SelectList(db.Беженцы, "ID_личности", "страна", снабже-
ние.ID_личности);
            return View(снабжение);
        }

        // POST: Снабжение/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "Медицинское,Продуктовое,ID_личности")]
Снабжение снабжение)
        {
```

```
            if (ModelState.IsValid)
            {
                db.Entry(снабжение).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.ID_личности = new SelectList(db.Беженцы, "ID_личности", "страна", снабже-
ние.ID_личности);
            return View(снабжение);
        }

        // GET: Снабжение/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Снабжение снабжение = db.Снабжение.Find(id);
            if (снабжение == null)
            {
                return HttpNotFound();
            }
            return View(снабжение);
        }

        // POST: Снабжение/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Снабжение снабжение = db.Снабжение.Find(id);
            db.Снабжение.Remove(снабжение);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

СотрудникиController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Data.SqlClient;
using System.Linq;
using System.Net;
using System.Runtime.Remoting.Contexts;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using Красный_крест;

namespace Красный_крест.Controllers
```

```
{
    public class СотрудникиController : Controller
    {
        private Красный_КрестEntities1 db = new Красный_КрестEntities1();

        // GET: Сотрудники
        [HttpPost]
        public ActionResult Index(string surname, float ogr)
        {

            var сотрудники = db.Сотрудники.Include(c => c.Квалификация);
            var check = сотрудники.Any(x => x.фамилия == surname);
            if (check == true)
            {
                SqlParameter a =
                  new SqlParameter("@surname", surname);
                SqlParameter b =
                new SqlParameter("@ogr", ogr);
                db.Database.ExecuteSqlCommand("EXEC procedura @surname, @ogr", a, b);
                return View(сотрудники.ToList());
            }
            else
            {
                string text = "Сотрудник не найден";
                ViewData["text"] = text;
                return View("Error");
            }
        }
        // GET: Сотрудники
        public ActionResult Index()
        {
            var сотрудники = db.Сотрудники.Include(c => c.Квалификация);
            return View(сотрудники.ToList());
        }

        // GET: Сотрудники/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Сотрудники сотрудники = db.Сотрудники.Find(id);
            if (сотрудники == null)
            {
                return HttpNotFound();
            }
            return View(сотрудники);
        }

        // GET: Сотрудники/Create
        public ActionResult Create()
        {
            ViewBag.ID_сотрудника = new SelectList(db.Квалификация, "ID_сотрудника", "специ-
альность");
            return View();
        }

        // POST: Сотрудники/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
```

```
        public ActionResult Create([Bind(Include =
"ID_сотрудника,имя,фамилия,отчество,дата_рождения,телефон,пол,возраст,заработная_плата,паспор
т")] Сотрудники сотрудники)
        {
            if (ModelState.IsValid)
            {
                db.Сотрудники.Add(сотрудники);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.ID_сотрудника = new SelectList(db.Квалификация, "ID_сотрудника", "специ-
альность", сотрудники.ID_сотрудника);
            return View(сотрудники);
        }

        // GET: Сотрудники/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Сотрудники сотрудники = db.Сотрудники.Find(id);
            if (сотрудники == null)
            {
                return HttpNotFound();
            }
            ViewBag.ID_сотрудника = new SelectList(db.Квалификация, "ID_сотрудника", "специ-
альность", сотрудники.ID_сотрудника);
            return View(сотрудники);
        }

        // POST: Сотрудники/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"ID_сотрудника,имя,фамилия,отчество,дата_рождения,телефон,пол,возраст,заработная_плата,паспор
т")] Сотрудники сотрудники)
        {
            if (ModelState.IsValid)
            {
                db.Entry(сотрудники).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.ID_сотрудника = new SelectList(db.Квалификация, "ID_сотрудника", "специ-
альность", сотрудники.ID_сотрудника);
            return View(сотрудники);
        }

        // GET: Сотрудники/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Сотрудники сотрудники = db.Сотрудники.Find(id);
            if (сотрудники == null)
            {
                return HttpNotFound();
```

```
            }
            return View(сотрудники);
        }

        // POST: Сотрудники/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Сотрудники сотрудники = db.Сотрудники.Find(id);
            if (ModelState.IsValid)
            {
                try
                {
                    db.Сотрудники.Remove(сотрудники);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    string text = "Ошибка";
                    ViewData["text"] = text;
                    return View("Error");
                }
            }
            return View(сотрудники);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Текущая_деятельностьController

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Красный_крест;

namespace Красный_крест.Controllers
{
    public class Текущая_деятельностьController : Controller
    {
        private Красный_КрестEntities1 db = new Красный_КрестEntities1();

        // GET: Текущая_деятельность
        public ActionResult Index()
        {
            var текущая_деятельность = db.Текущая_деятельность.Include(т =>
т.Направление_работы);
            return View(текущая_деятельность.ToList());
        }
```

```
        // GET: Текущая_деятельность/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Текущая_деятельность текущая_деятельность = db.Текущая_деятельность.Find(id);
            if (текущая_деятельность == null)
            {
                return HttpNotFound();
            }
            return View(текущая_деятельность);
        }

        // GET: Текущая_деятельность/Create
        public ActionResult Create()
        {
            ViewBag.ID_работы = new SelectList(db.Направление_работы, "ID_работы",
"ID_работы");
            return View();
        }

        // POST: Текущая_деятельность/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "наименова-
ние_деятельности,ID_работы,продолжительность_смены,услуги")] Текущая_деятельность теку-
щая_деятельность)
        {
            if (ModelState.IsValid)
            {
                db.Текущая_деятельность.Add(текущая_деятельность);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.ID_работы = new SelectList(db.Направление_работы, "ID_работы",
"ID_работы", текущая_деятельность.ID_работы);
            return View(текущая_деятельность);
        }

        // GET: Текущая_деятельность/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Текущая_деятельность текущая_деятельность = db.Текущая_деятельность.Find(id);
            if (текущая_деятельность == null)
            {
                return HttpNotFound();
            }
            ViewBag.ID_работы = new SelectList(db.Направление_работы, "ID_работы",
"ID_работы", текущая_деятельность.ID_работы);
            return View(текущая_деятельность);
        }

        // POST: Текущая_деятельность/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свой-
ства, для которых следует установить привязку. Дополнительные
```

```
        // сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "наименова-
ние_деятельности,ID_работы,продолжительность_смены,услуги")] Текущая_деятельность теку-
щая_деятельность)
        {
            if (ModelState.IsValid)
            {
                db.Entry(текущая_деятельность).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.ID_работы = new SelectList(db.Направление_работы, "ID_работы",
"ID_работы", текущая_деятельность.ID_работы);
            return View(текущая_деятельность);
        }

        // GET: Текущая_деятельность/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Текущая_деятельность текущая_деятельность = db.Текущая_деятельность.Find(id);
            if (текущая_деятельность == null)
            {
                return HttpNotFound();
            }
            return View(текущая_деятельность);
        }

        // POST: Текущая_деятельность/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Текущая_деятельность текущая_деятельность = db.Текущая_деятельность.Find(id);
            if (ModelState.IsValid)
            {
                try
                {
                    db.Текущая_деятельность.Remove(текущая_деятельность);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch
                {
                    string text = "Ошибка";
                    ViewData["text"] = text;
                    return View("Error");
                }
            }
            return View(текущая_деятельность);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
```