ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»

(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УЧЕТА ДОНОРОВ БЕЛГОРОДСКОЙ СТАНЦИИ ПЕРЕЛИВАНИЯ КРОВИ

Выпускная квалификационная работа
обучающегося по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем
очной формы обучения, группы 07001402
Саградян Карена Агасовича

Научный руководитель
доцент Румбешт В.В.

БЕЛГОРОД 2018

Оглавление

ВВЕДЕНИЕ

21 век – век развития и совершенствования информационных технологий. На сегодняшний день, существует бесчисленное множество различных информационных технологий, которые используются повсеместно, что значительно упрощает, упорядочивает и делает более доступными те или иные услуги, в зависимости от потребностей людей. Можно смело сказать, что мы живем в «информационную эпоху».

Информация распространяется крайне молниеносно, поэтому возникает необходимость упорядочивать, структурировать и взаимодействовать с ней, такой подход мог бы пригодится в случаях, когда возникает резкая необходимость использовать упорядоченные данные, как, например, в случаях, когда необходимо найти доноров с определенной группой крови, связаться с ними и в кратчайшие сроки получить бесценную кровь. На современном этапе развития медицины, использование в лечении больных донорской крови и ее компонентов является непременным условием в борьбе за жизнь человека и восстановления его работоспособности. Кровь и ее компоненты — незаменимые средства при лечении заболеваний крови, онкологических заболеваний, при родовспоможении, травмах и ранениях.

Актуальность данной работы обуславливается отсутствием функций удаленного доступа для доноров, которые могли бы ускорить и упростить работу с их данными на самой станции переливания крови.

Целью данной дипломной работы является создание автоматизированной системы учета доноров Белгородской станции переливания крови. Основными функциями данной системы будут являться возможность автоматизированной, удобной, структурированной и удаленной работы с донорами, используя всемирную систему объединённых компьютерных сетей для хранения и передачи информации – интернет.

Для того чтобы разработать автоматизированную систему, описанную выше, следует выделить и указать ряд ключевых задач, которые включают в себя:

1) анализ деятельности организации;
2) выявление ключевых направлений деятельности организации, требующих автоматизации;
3) обзор и подробный анализ систем, выполняющих схожие задачи;
4) постановка требований к разрабатываемой системе;
5) проектирование системы учета;
6) реализация системы учета;
7) проведение испытаний.

Данная выпускная квалификационная работа состоит из введения, трех глав, заключения списка литературы и приложения.

Введение содержит общие сведения о работе, ее актуальность, цели, задачи и способы их достижения.

Первая глава содержит описание организации, анализ ее деятельности и постановку задач, требующих решения.

Вторая глава посвящена проектированию и реализации системы учета доноров.

Третья глава содержит методику тестирования системы и сам процесс тестирования.

В заключении подводится итог проведенной работе.

Дипломная работа состоит из 75 страниц, 45 рисунков, 1 таблицы и приложения, включающего 10 страниц.

ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ

1.1 Особенности организации

В настоящее время ОГБУЗ "БОСПК" является многофункциональным лечебным учреждением и фабрикой по переработке крови. Выпускается более 20 наименований продукции, осуществляется организационно-методическое руководство службы крови области (3 СПК, 2 ОПК), медицинские организации области в самом полном объеме совершенно бесплатно обеспечиваются компонентами и препаратами крови, оказывается консультативная помощь медицинским учреждениям по вопросам клинической трансфузиологии. В данной организации работает цецлое множество людей, все они являются классифицированными специалистами, поскольку это областное государственное бюджетное учреждение здравоохранения.

Ниже приведена деятельность данной станции:

- координирование работы службы крови в регионе;
- планирование, комплектация, учёт и обследование донорских кадров;
- заготовка и переработка крови;
- контроль заготовки консервированной крови, её компонентов и препаратов;
- производство необходимых лабораторных исследований крови;
- оказание консультативной трансфузиологической помощи, подготовка кадров.

Служба крови — это структура, которая объединяет по всей стране медицинские учреждения и их структурные подразделения, основным видом деятельности которых является заготовка, переработка, хранение и обеспечение безопасности донорской крови и ее компонентов.

Это чрезвычайно важная служба, поскольку у крови нет никаких аналогов, используемых в лечении. Здесь приведен официальный прогноз Росстата до 2036 года, который предполагает изменение численности населения страны в диапазоне от 136,7 млн человек (низкая рождаемость и низкая миграция) до 157,1 млн человек (высокая рождаемость и высокая миграция). Можно сделать незамысловатый вывод, что банку крови потребуется все больше и больше ресурсов. Ведь кровь никогда не бывает в избытке.

Вернемся непосредственно к самой станции, чтобы правильно создать автоматизированную систему учета доноров необходимо проанализировать текущую структуру данной станции. Я сам являюсь донором и посещаю данную станцию приблизительно каждые 2 месяца, поэтому я могу описать сам процесс становления донора на учет. Персоналу, который ставит на учет вновь прибывших доноров, необходим перечень документов для того, чтобы поставить на учет донора, причем, учитывая, что вся эта информация может меняться с течением времени, периодически возникает необходимость перепроверять и сверять эти данные с актуальными данными. После этого донор каждый раз проходит процедуру определения состава крови.

Здоровые доноры, которые сдают кровь регулярно(активные доноры) — это основа донорского движения и надежный гарант того, что в нужный момент донорской крови будет достаточно для спасения жизни нуждающегося в переливании. Поэтому одним из главных приоритетов учреждений Службы крови является полное обеспечение 100 %-й безопасности процедуры сдачи крови для донора. Для решения этой задачи принимаются следующие меры:

- За состоянием здоровья доноров ведется непрерывное и качественное медицинское наблюдение[8]. Каждый первичный донор отправляется на прием к врачу-трансфузиологу после медицинского обследования и клинико-лабораторного исследования крови. Активные доноры каждый год проходят медицинское обследование, которые включает в себя: сдачу анализов,

проведение рентгеноскопического (или флюорографического) обследования органов грудной клетки, электрокардиографии и многие другие.

Помимо этого, каждый раз необходимо заполнить анкету, которая так же помогает установить, является ли возможным сдача крови на данный момент для донора или нет.

Таким образом, деятельность данной станции переливания крови имеет широкий круг задач, которые нуждаются в автоматизации и информационном сопровождении. Исходя из вышеперечисленного, следует перечень аспектов деятельности станции переливания крови, подлежащих автоматизации:

1) Построение отчетности о текущей деятельности станции. Отчетность о деятельности должна быть представлена в виде блога, который содержит в себе множество статей, добавляемых после участия в мероприятиях, или в случае необходимости информационной поддержки какого – либо события. Структура блога должна представлять собой страницу со списком уже имеющихся отчетов о мероприятиях, открываемой при выборе какого-то конкретного мероприятия и отображающей достаточно подробную информацию о данном событии.

2) Объявление приближающихся мероприятий и предоставление необходимой информации, такой как место и время проведения мероприятия, а так же информирование о подробностях. В случае выбора определённого мероприятия, должна открываться новая страница с детальной информацией о выбранном событии. Страница с объявлением выбранного мероприятия должна содержать такую информацию, как дата и время проведения мероприятия, время начала мероприятия, все требования, предъявляемые к участникам мероприятия, фотографии места проведения мероприятия и другую информацию по необходимости.

3) Предоставление информации, которая необходима для изучения процедуры сдачи крови и ее необходимости. К информации такого рода могут относиться следующие понятия: правила и требования, безопасность,

подробное и понятное описании используемой технологии. Такая информация должна быть организована в виде страницы с множеством статей, из которых будет возможность выбрать интересующую пользователя информацию для более подробного изучения.

4) Возможность хранить и изменять данные о донорах, осуществлять быстрый доступ к данным по средствам сети интернет. Сюда входит информация о деятельности донора на Белгородской областной странции переливания крови, т.е. количества донаций и даты осуществления , а так же даты следующего возможного посещения с учетом периода осуществления донаций, который установлен законом.

5) Возможность удаленного анкетирования, как мы уже выяснили, каждый раз, приходя на станцию, донор должен заполнить анкету, чтобы специалист мог выявить проблемы со здоровьем, либо же установить, является ли возможным сдать кровь в данный момент.

6) Возможность пользователям осуществлять запись на прием и обращения к сотрудникам станции через личный кабинет, т.е. удаленно.

Все перечисленные пункты необходимо реализовать так, чтобы персонал станции переливания крови, имел возможность управлять информацией на страницах веб-приложения, не обладая особыми знаниями в сфере программирования, а также пользователи могли использовать описанный ранее функционал личного кабинета.

1.2 Обзор и анализ существующих систем

В век цифровых технологий и проникновения интернета во все сферы человеческой деятельности невозможно успешно вести какую-либо работу, ориентированную на взаимодействие с людьми, не имея представительства в сети.. Однако все эти сайты в той или иной мере не отвечают тем

требованиям, которые необходимо выполнять сайту, чтобы подобающим образом упростить ряд аспектов работы Белгородской областной станции переливания крови, требующих автоматизации. Чтобы убедиться в этом, можно рассмотреть несколько таких сайтов и оценить то, как они справляются с поставленными задачами.

Система 1. Сайт Иркутской станции переливания крови

Первой рассмотренной системой будет сайт Иркутской станции переливания крови, расположенный по интернет адресу: http://www.donor38.ru. На рисунке 1.1 отображена главная страница сайта.

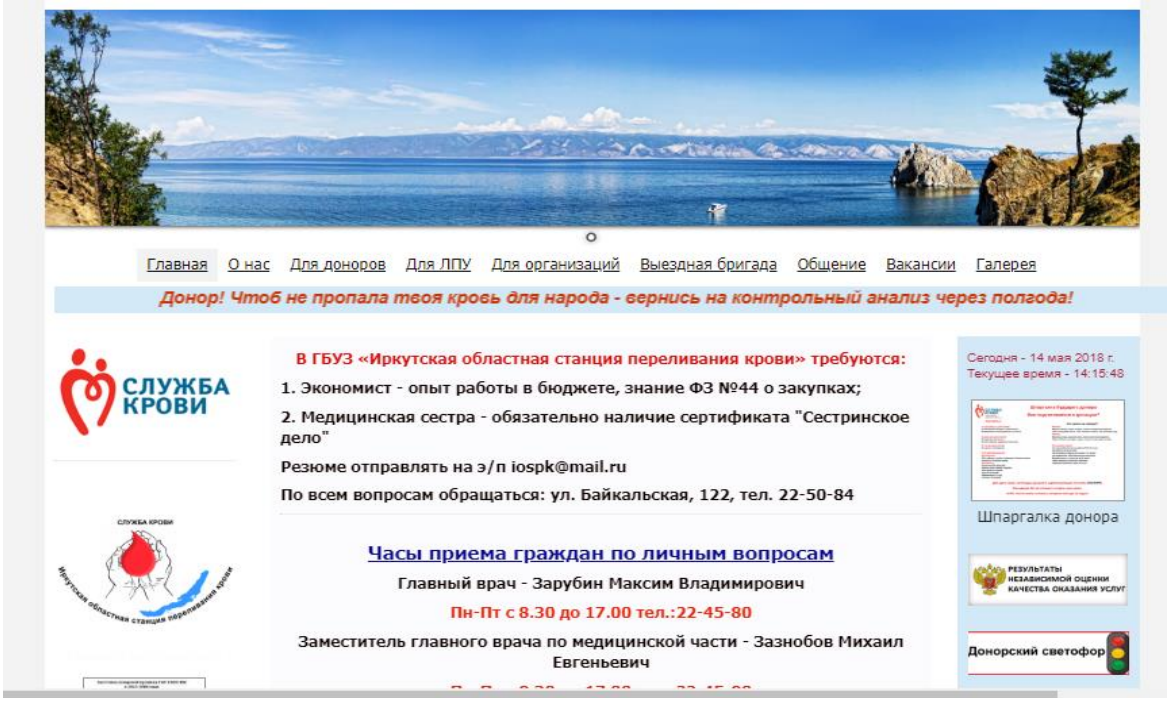

Рис. 1.1. Главная страница

Рассмотрим то, как сайт реализует приведенные выше аспекты требующие автоматизации:

1) Построение отчетности о текущей деятельности станции. На рисунке 1.1 изображена главная страница, на которой реализовано предоставление отчетности о деятельности этой станции. Новостной раздел представлен в виде блога с перечнем новостей, однако он сделан так, что

новость отображается либо полностью, что затрудняет пользователю искать интересующую его информацию. Помимо всего этого, все новости отображаются единовременно на одной странице, тем самым растягивая ее до запредельных размеров, что является крайней нерационально и неудобно, отсутствуют какие либо категории.

2) Способ анонсирования грядущих мероприятий и предоставление информации о месте и времени проведения мероприятия, а так же информирование о подробностях на сайте присутствует. Единственной страницей, выполняющей малую часть необходимого функционала, является страница с новостями которая не содержит каких либо категорий и все новости отображены на главной странице, что делает ее трудно читаемой, отсутствует поиск. Однако даже тот функционал, который имеется, реализован достаточно неудобно. Использование подобного способа информирования явный минус.

3) Предоставление информации необходимой для изучения процесса осуществления кроводачи реализовано. Страница сайта, которая частично реализует этот аспект, показана на рисунке 1.2. Отсутствует личный кабинет.

Рис. 1.2. Информация для доноров

4) Личный кабинет отсутствует, впрочем, как и регистрация. Возможность удаленного анкетирования и записи на прием, очевидно, так же осутствует.

Рассмотрев данную систему и оценив все возможности, можно прийти к выводу, что данный сайт слабо организует все аспекты деятельности станции переливания крови, подлежащей к автоматизации.

Система 2. Сайт Белгородской станции переливания крови

Следующий рассмотренный сайт располагается по адресу: www.donor-bel.belzdrav.ru. Как видно из рисунка 1.3 дизайн главной страницы сайта выглядит привлекательно, однако для того чтобы оценить насколько этот сайт отвечает требованиям, которые необходимо выполнять для корректной автоматизации учета доноров на Белгородской станции переливания крови, следует рассмотреть функционал сайта и оценить то на сколько реализованы требуемые аспекты:

Рис. 1.3. Главная страница

1) Построение отчетности о текущей деятельности Белгородской станции переливания крови. Сразу после перехода на сайт перед пользователем на главной странице предстает перечень свежих новостей. Как показано на рисунке 1.4 пользователь имеет возможность нажать новость

для того чтобы открыть подробности о событии. Однако после нажатия на новость, она так и останется открытой, нет возможности закрыть ее.

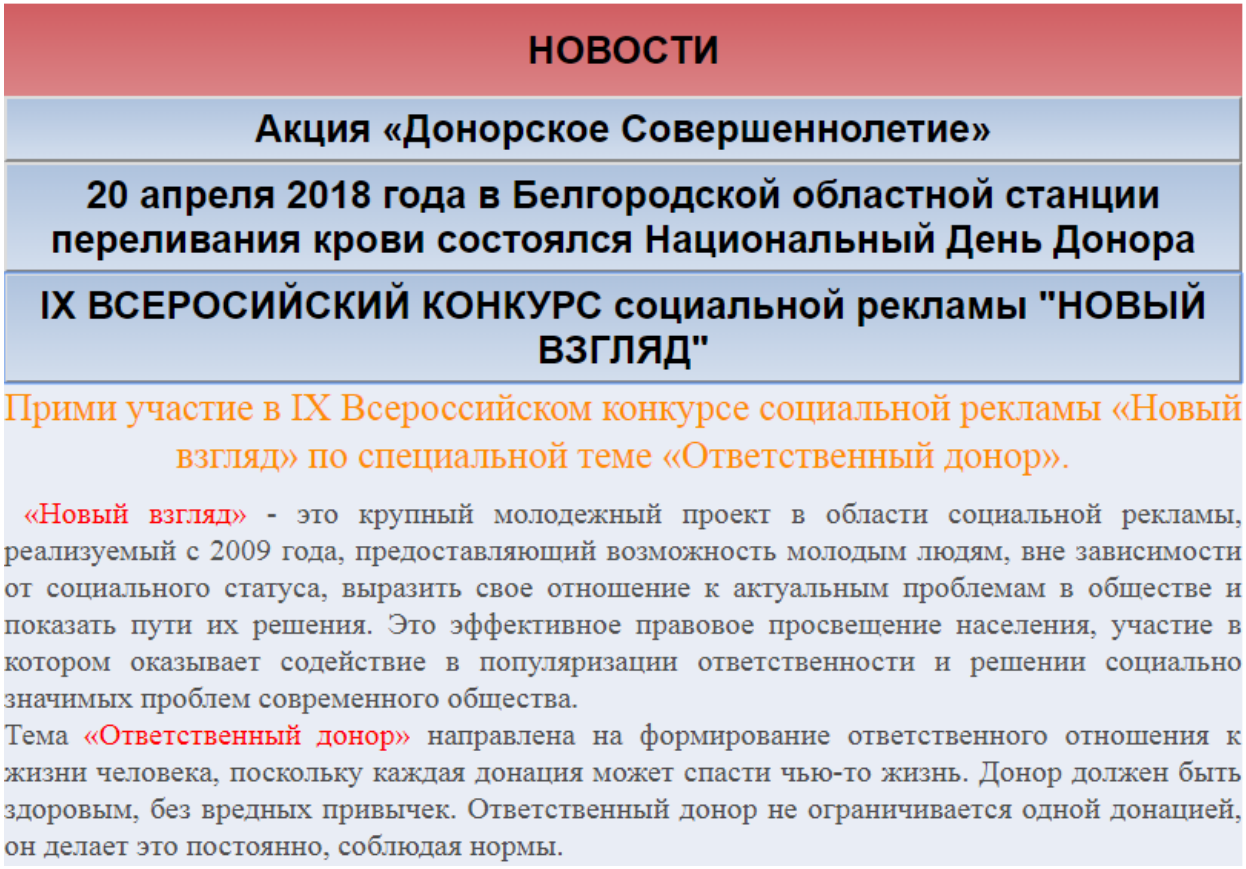

Рис. 1.4. Новости

На рисунке 1.5 показано отображение информации в нижней части главной страницы, слева в блоке размещены неактивные ссылки, которые выходят за границы разметки и сливаются с текстом, делая его трудночитаемым либо же вовсе не читабельным.

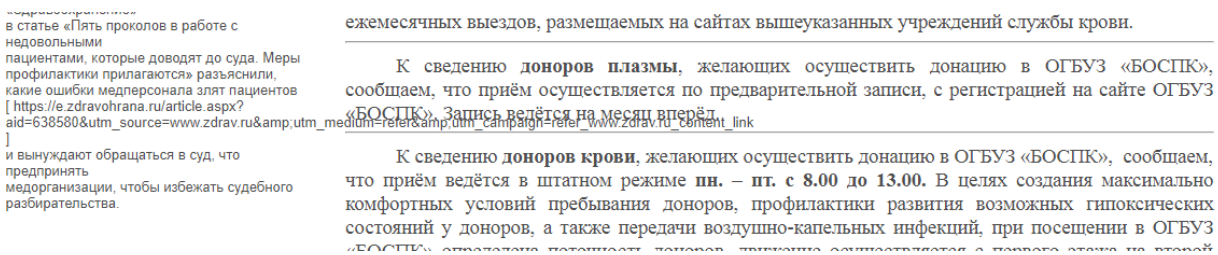

Рис. 1.5. Нижняя часть главной страницы

2) Анонсирование грядущих мероприятий и предоставление информации о месте и времени проведения мероприятия, а так же информирование о подробностях на сайте присутствует, это наглядно проиллюстрировано на рисунке 1.4.

Рис.1.6. Информация для доноров

3) Предоставление информации необходимой для изучения процедуры кроводачи в той или иной степени реализовано. Страница сайта, которая реализует этот аспект, показана на рисунке 1.6 – это страница «Информация для доноров».

4) Имеется личный кабинет. Он показан на рисунке 1.7

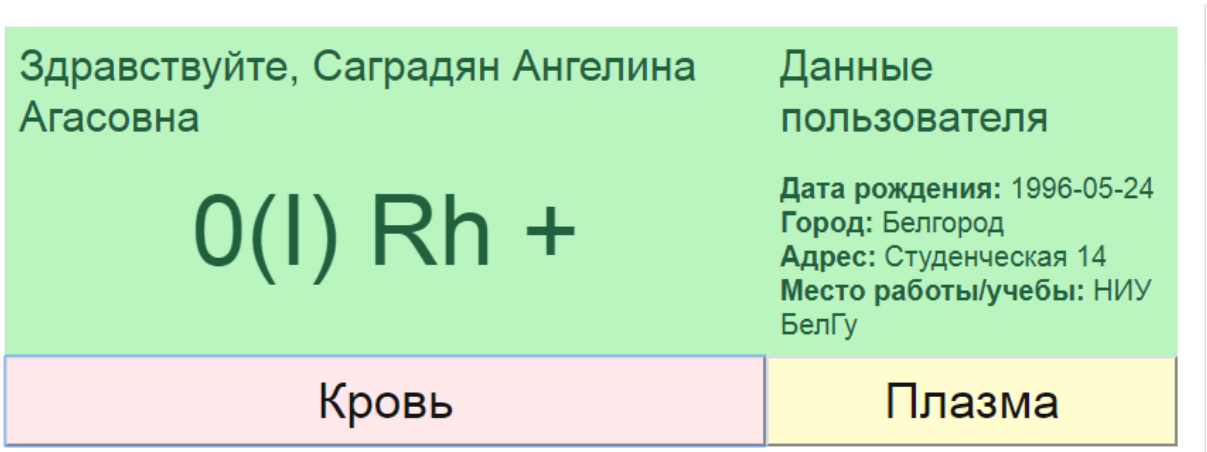

Рис.1.7. Личный кабинет

Весь его функционал сводится к электронной записи, также здесь можно просмотреть некоторую информацию о себе. Однако и тут есть свои недоработки, одна из них показана на рисунке 1.8

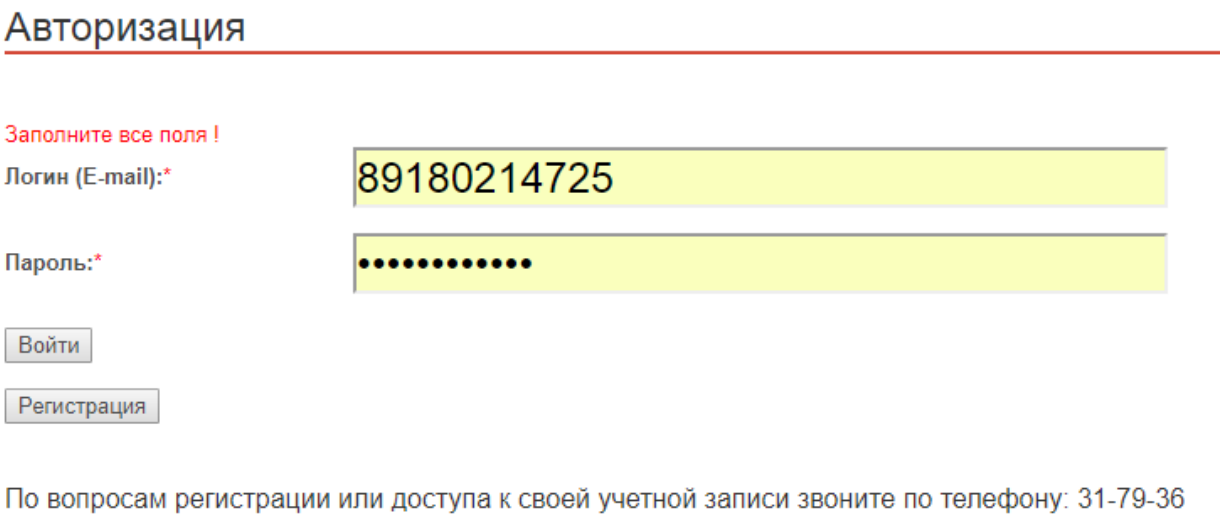

Рис. 1.8. Авторизация

Во первых, следует обратить внимание на то, что отсутствует кнопка выхода из личного кабинета. Выход осуществляется, как только осуществляется переход на любую другую часть сайта, пользователь автоматически покидает личный кабинет и для повторного входа нужно вводить данные заново. Помимо этого, отсутствует восстановление пароля, как сказано в форме, вам необходимо обратиться по номеру, что тоже нельзя назвать удобным. Сайт не связан с базой данных Белгородской станции переливания крови, он используется в качестве представительства станции в сети, соответственно, история донаций недоступна в личном кабинете, ровным счетом как и любая другая информация, связанная с деятельностью пользователя на станции.

Система 3. Сайт Краснодарской станции переливания крови.

Еще одной рассмотренной системой будет сайт Краснодарской станции переливания крови расположенный по интернет адресу: http://www.gbuzspk.ru.

1) Открыв главную страницу данного сайта, которая показана на рисунке 1.9 видно, что данный сайт представляет определенную информацию, необходимую донорам и посетителям ресурса.

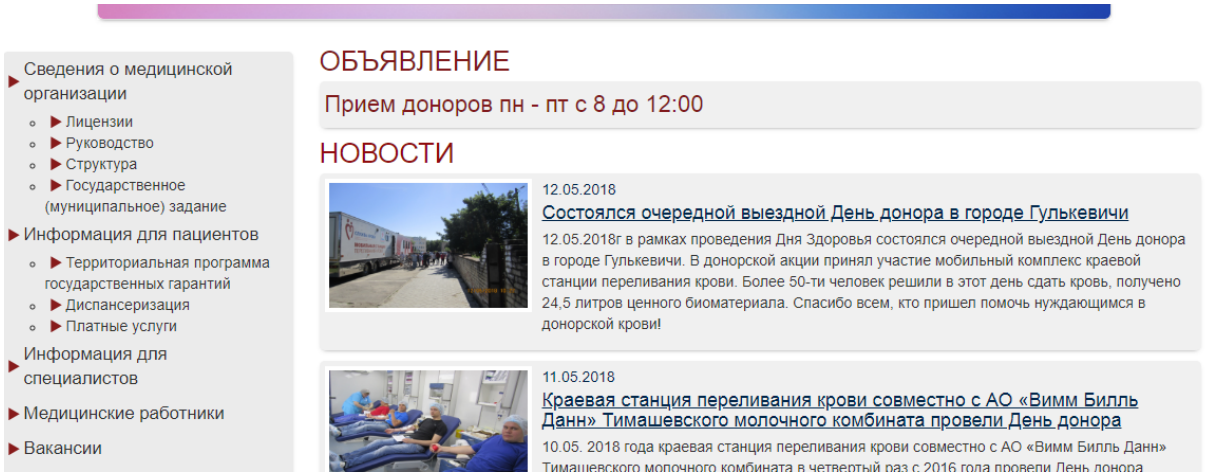

Рис. 1.9. Главная страница сайта Краснодарской станции переливания крови

Но открыв несколько разделов, я обнаружил их пустыми, это показано на рисунке 1.10. Сайт грузится весьма медленно, а информация в разделах для пациентов отсутствует на момент составления этого отчета. Однако составление отчетности на данный момент можно считать частично реализованным.

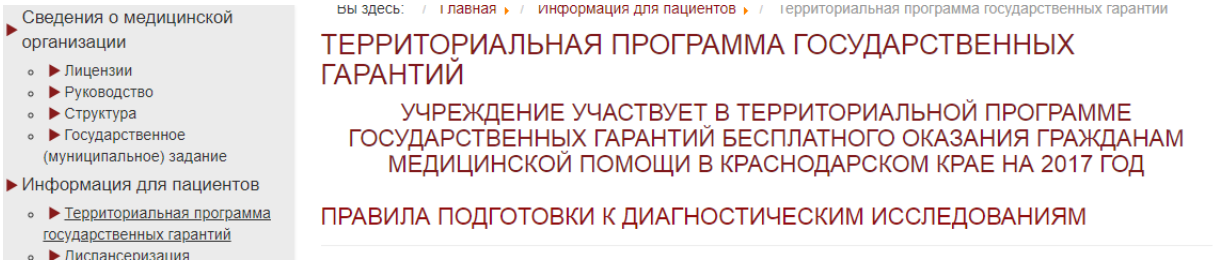

Рис. 1.10. Информация

2) Способ анонсирования мероприятий реализован на главной странице, рисунок 1.9;

3) предоставление информации необходимой для изучения процесса кроводачи не реализовано. Это показано на рисунке 1.10. Разделы не содержат никакой информации.
4) Личный кабинет и регистрация отсутствуют.

Таким образом, рассмотрев ряд систем, три из которых представлены выше, можно прийти к выводу, что те системы, которые уже существуют – не способный в полной мере удовлетворить требования, предъявленные к системе. Анализ этих систем основывался на пяти критериях, указанных в круге задач деятельности станций. Все эти критерии подлежат автоматизации. Для большей наглядности результаты анализа представлены в таблице 1.1.

Таблица 1.1

Анализ существующих систем

Станция переливания крови	Отчет о деятельности	Анонс мероприятий	Образовательные материалы	Личный кабинет	Анкетирование
«Иркутская»	+	+/-	+/-	-	-
«Белгородская»	+	+	+	+/-	-
«Краснодарская»	+/-	+	-	-	-

Исходя из анализа, возникает потребность разработать собственную систему, которая будет автоматизировать деятельность станции, в соответствии с описанными выше аспектами, подлежащими автоматизации.

1.3 Требования к автоматизированной системе

Требования к функциональным характеристикам

1) Автоматизированная система обязана обеспечивать выполнения перечисленных функций:

- предоставление отчетности о текущей деятельности станции. Отчетность должна включать в себя информацию о новостях и изменениях;
- предоставление посетителям объявления приближающихся событий;
- предоставление доступа к информации необходимой посетителям ресурса и донорам, такой как процедура сдачи крови, ее необходимость, льготы для доноров и так далее;
- реализовать возможность хранения и изменения информации о донорах и их деятельности, а также удаленного доступа к этой информации;
- реализовать возможность пользователям удаленно осуществлять запись на прием и обращения к сотрудникам станции через личный кабинет;
- защита базы данных от несанкционированного доступа к данным.

2) Требования к организации входных данных

Входными данными в программе являются данные, которые сторонние пользователи вводят в процессе регистрации – такие как:

- email;
- пароль.

Для сотрудников станции входные данные должны быть аналогичными, разделение на сотрудников и пользователей должно осуществляться путем разделения прав доступа.

На странице добавления новостей администратор сайта должен иметь возможность ввести следующие данные:

- название заголовка;
- название новости;

- название категории;

- краткое описание;

- полное описание;

- изображения для содержимого новости при необходимости;

- мета заголовок, для нахождения ресурса через интернет поисковики;

- мета описание, для нахождения ресурса через интернет поисковики;

- ключевые слова для поиска через интернет поисковики;

На странице добавления приближающихся мероприятий администратор сайта должен иметь возможность ввести следующие данные:

- название заголовка;

- название мероприятия;

- название категории;

- краткое описание мероприятия;

- полное описание мероприятия;

- изображения для содержимого новости при необходимости;

- мета заголовок, для нахождения ресурса через интернет поисковики;

- мета описание, для нахождения ресурса через интернет поисковики;

- ключевые слова для поиска, для нахождения ресурса через интернет поисковики.

На странице добавления обучающей информации администратор сайта должен иметь возможность ввести следующие данные:

- название статьи;

- содержимое статьи;

- родительскую категорию статьи при необходимости.

На странице добавления донора в список администратор сайта должен иметь возможность ввести следующие данные:

- имя;
- отчество;
- фамилия;
- пароль(при необходимости);
- email;
- дата рождения;
- место проживания;
- паспортные данные(серия и номер);
- место работы;
- номер телефона;
- группу крови.

Общие требования к входным данным, вводимым пользователем в программу: данные должны вводиться посредством специальный полей, которые проверяют данные прежде чем отправить их на сервер, в случае некорректного ввода, пользователь должен получать сообщение об ошибке при вводе и подсказку для правильного ввода.

3) Требования к организации выходных данных

Вывод данных должен осуществляться путем отображения информации на страницах веб-приложения.

На главной странице приложения должен быть отображен донорский светофор и краткий список последних новостей.

На странице новости должны отображаться данные из таблицы новостей, которые формируют список, состоящий из:

- заголовка новости;
- краткого содержания.

На странице информации должны отображаться данные из таблицы новостей, которые представляют собой статьи с необходимой информацией, формирующие список, состоящий из:

- заголовка статьи;
- краткого содержания статьи.

В личном кабинете должна быть отображена информация об авторизированном пользователе, его персональные данные, имеющиеся в базе, а так же количество донаций, дата следующей возможной донации , количество необходимых донаций до получения звания почетного донора и журнал донаций, осуществленных этим пользователем.

А также, применимыми ко всем вводимым данным, являются следующие требования:

- доступ к таблицам должен зависеть от роли пользователей, которые определяют права доступа;
- страницы веб-приложения должны формироваться динамически, в зависимости от тех данных, которые находятся в базе данных и в зависимости от действий пользователей формирующих запросы.

4) Требования к временным характеристикам

Требования к временным характеристикам зависит от количества запрашиваемых запросов и пропускной способности сервера. Автоматизированная система должна осуществлять запросы к серверу только по необходимости.

5) Требования к надежности

Надежная работа автоматизированной системы должна обеспечиваться выполнением следующих организационно-технических мероприятий:

- обеспечением бесперебойного питания сервера;
- обеспечением устойчивости системы путем разделения на независимые части (модули);
- обеспечением защищенности персональной информации пользователей путем хеширования паролей и ограничением доступа к данной информации неуполномоченных сотрудников;

6) Отказы из-за некорректных действий оператора

Отказы программы вследствие некорректных действий оператора (пользователя) при взаимодействии с автоматизированной системой. Во избежание возникновения всяческих отказов автоматизированной системы, по выше-перечисленной причине, необходимо обеспечить работу пользователя без предоставления ему административных привилегий.

7) Климатические условия эксплуатации

Климатические эксплуатационные условия, которые должны обеспечивать заданные характеристики, должны соответствовать требованиям, предъявляемым к техническим средствам в части условий их непосредственной эксплуатации.

8) Требования к численности и квалификации персонала

Для управления автоматизированной системой требуется как минимум один человек, который будет заносить актуальную информацию. Так же требуется персонал, который будет обслуживать сервер, на котором система будет размещена.

9) Требования к составу и параметрам технических средств

Для работы автоматизированной системы на стороне клиента требуется персональный компьютер с доступом в интернет. Для работы приложения на стороне сервера требуется персональный компьютер или специализированное оборудование, которые способны выполнять задачи сервера, а также имеющие доступ к сети интернет

10) Требования к информационным структурам и методам решения

Пользовательский интерфейс должен быть предельно ясным и содержать подсказки в случаях, когда это необходимо. Должен существовать программный доступ к управлению содержимым. Новые страницы с данными должны отображаться по следующему принципу – последняя запись отображается первой в списке.

11) Требования к исходным кодам и языкам программирования

Языком программирования, реализующим исходные коды стороне сервера, должен быть язык программирования PHP с использованием Laravel

Framework. Интегрированной средой разработки автоматизированной системы должна быть использована среда PHP Storm. В качестве СУБД должна использоваться база данных PostgreSQL. Для взаимодействия с СУБД и создания базы данных должно использоваться объектно-реляционное отображение Eloquent ORM Laravel Framework и PgAdmin 3. Языки программирования на стороне клиента: HTML и Javascript , включая библиотеку Jquery.

12) Требования к программным средствам, используемым программой

У клиента, для работы с автоматизированной системой должна быть установлена любая операционная система, поддерживающая взаимодействие с всемирной сетью интернет и работу любого современного веб-браузера..

13) Требования к защите информации и программ

В системе должен быть обеспечен надлежащий уровень защиты персональной информации от несанкционированного доступа.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Технология построения приложения

Шаблон проектирования «Модель–Представление–Контроллер» «Model–View–Controller» является схемой[1], которая основана на разделении данных, пользовательского интерфейса и управляющей логики в виде трех независимых составляющих: модель, представление и контроллер, при этом, если возникнет необходимость модифицировать любой из них, не придется модифицировать другие см. рисунок 2.1.

Рис 2.1. MVC архитектура

Сама концепция MVC довольно стара, она была описана еще в далеком 1978 сотрудником научно-исследовательского центра «Xerox PARC» Трюгве, который работал над языком программирования «Smalltalk».Только в 1988 году была напечатана финальная версия концепции MVC архитектуры в журнале «Technology Object». Позже компания «Apple», внедрила

технологию «WebObjects», реализованную на «Objective-C» и стала популяризировать шаблон и в вебе.

Структура MVC приложения показана на рисунке 2.2.

Рис. 2.2. Структура взаимодействия MVC

Особенности архитектурного шаблона MVC

При использовании архитектурного шаблона MVC предполагается, разделение приложения MVC, как минимум, на три части, которые описаны ниже.

- Модели, которые хранят данные, с которыми работают пользователи, или представляют эти данные[2]. Модели могут быть, как обыкновенными моделями представлений, отображающими данные, которые передаются между представлениями и контроллерами, так и моделями предметной области, содержащими бизнес-данные, а так же операции, для преобразования и правила для обработки этих данных.

- Представления, которые применяются для визуализации конкретной части модели в виде пользовательского интерфейса.

- Контроллеры, обрабатывающие запросы, которые к ним поступают - выполняют операции над моделью и подбирают то представление, которое требуется для визуализации пользователю[3].

Все составляющие архитектуры MVC точно определены и независимы – они реализуют принцип разделения ответственности. Логика, управляющая данными модели, содержится только в ней, логика, которая отображает данные, находиться только в представлении, а код, который обрабатывает

пользовательский ввод и запросы содержится в контроллере. Таким образом, четко разграничивая все части в приложении, сопровождение и расширение в течение срока его существования происходит значительно легче, вне зависимости от его увеличения и усложнения.

На рисунке 2.3 показана диаграмма вариантов использования разрабатываемой системы для пользователей.

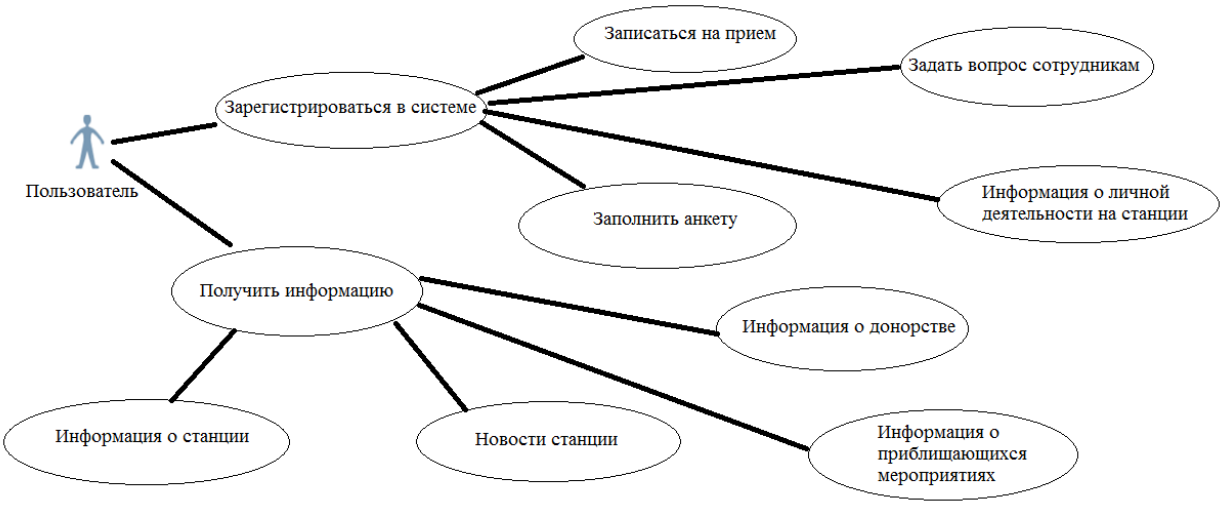

Рис. 2.3. Диаграмма вариантов использования системы пользователями

На следующем рисунке 2.4, показана диаграмма вариантов использования системы сотрудниками и администраторами

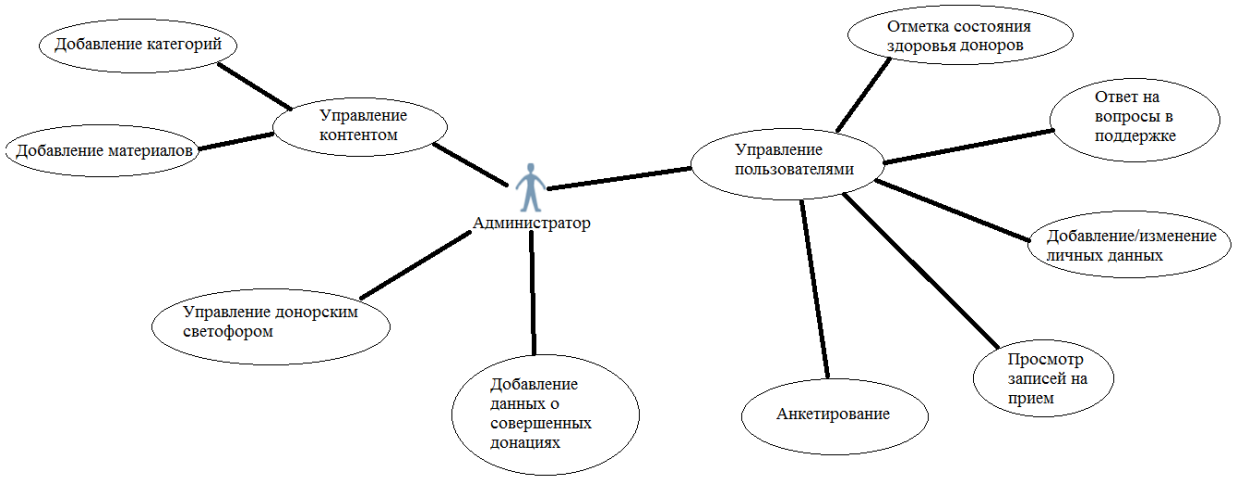

Рис. 2.4. Диаграмма вариантов использования системы администраторами

С учетом MVC архитектуры, структура приложения будет выглядеть следующим образом.

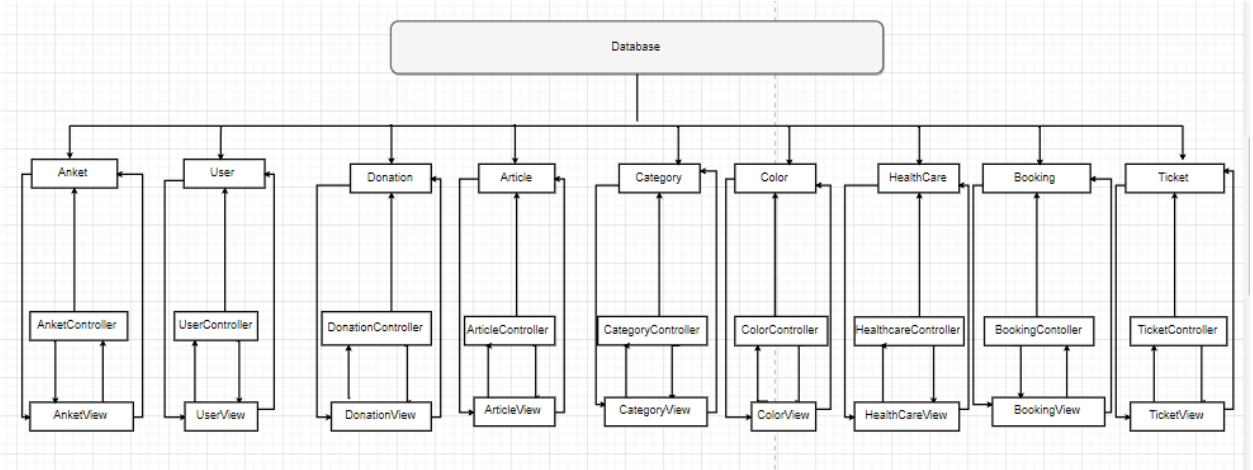

Рис. 2.5. MVC схема приложения

На рисунке 2.5 показана MVC схема разрабатываемого приложения, всего на ней отображено 4 группы компонентов, такие как: база данных, модель или класс, контроллер и представление.

2.2 Проектирование базы данных

Основой любого приложения являются данные, которыми оно управляет, поэтому проектирование системы стоит начать именно с них. Для того чтобы реализовать установленный требованиями функционал, нужна база данных, которая будет хранить в себе всю необходимую информацию для работы системы.

В качестве базы данных для разрабатываемого мною приложения должна использоваться СУБД Postgres.

PostgreSQL — это объектно-реляционная система управления базами данных (ОРСУБД, ORDBMS), основанная на POSTGRES, Version 4.2 — программе, разработанной на факультете компьютерных наук в стенах

Калифорнийского университета в Беркли. В POSTGRES появилось множество новшеств, которые были реализованы в некоторых коммерческих СУБД значительно позднее.

PostgreSQL — СУБД с открытым исходным кодом, основой которого был код, написанный в Беркли[5]. Она поддерживает большую часть стандарта SQL и предлагает множество современных функций, такие как:

А благодаря свободной лицензии, эту систему управления базами данных разрешено бесплатно использовать, изменять и распространять для совершенно любых целей — будь то личные, коммерческие или учебные.

В Laravel можно чрезвычайно просто взаимодействовать с БД на разных «движках», будь то сырой SQL, гибкий конструктор запросов или Eloquent ORM. На данный момент Laravel поддерживает следующие четыре системы баз данных:

- MySQL
- Postgres
- SQLite
- SQL Serv

Работа с базой данных в Laravel осуществляется посредством миграций. Миграции — это своего рода системы контроля версий для вашей базы данных. Они позволяют вам и вашей команде изменять структуру БД, в то же время отслеживая изменения других участников. Миграции, как правило, идут вместе с построителем структур для более простого обращения с архитектурой вашей базы данных. Если вам когда-нибудь приходилось вручную добавлять столбец в не локальную БД, значит вы сталкивались с проблемой, которую решают миграции БД.

Фасад Laravel Schema обеспечивает поддержку создания и изменения таблиц в независимости от любой используемой СУБД из числа тех, что поддерживаются в Laravel.

Для создания миграций используется Artisan. Artisan — интерфейс командной строки, который поставляется с Laravel. Он содержит набор весьма полезных команд, помогающих при разработке приложений. На

рисунке показаны команды, через которые мы можем взаимодействовать с Artisan.

Это очень удобно и эффективно, поскольку одна команда в консоли способна заменить целый ряд действий в построении базы данных либо же самого приложения.

Каждая команда имеет описание, в котором указаны её доступные аргументы и ключи. Для просмотра описания необходимо добавить перед командой слово help; Ниже показан пример использования help для команды выполнения миграций.

Исходя из всего выше сказанного, для работы с БД необходимо использовать Artisan, посредством которого создаются и выполняются миграции.

Но для начала необходимо спроектировать инфологическую модель базы данных. В соответствии с требованиями, необходимо выделить следующие сущности, чтобы решить поставленную задачу:

- пользователи, для хранения информации о пользователях;
- новости, для хранения содержимого новостей и статей;
- категории, для группирования новостей и статей;
- состояние здоровья доноров, для регистрации данных о состоянии здоровья в день осуществления донации.
- донации, для хранения информации о донациях, осуществляемых на Белгородской станиции переливания крови;
- электронная запись, для осуществления процедуры электронной записи доноров;
- анекирование, для осуществления процедуры прохождения анкет;
- светофор, для хранения информации о текущем состоянии донорского светофора, который является индикатором количества запасов той или иной группы крови в банке крови.

Теперь можно спроектировать инфологическую модель базы данных. В качестве среды проектирования использовался Erwin Data Modeler. Модель отображена на рисунке 2.6.

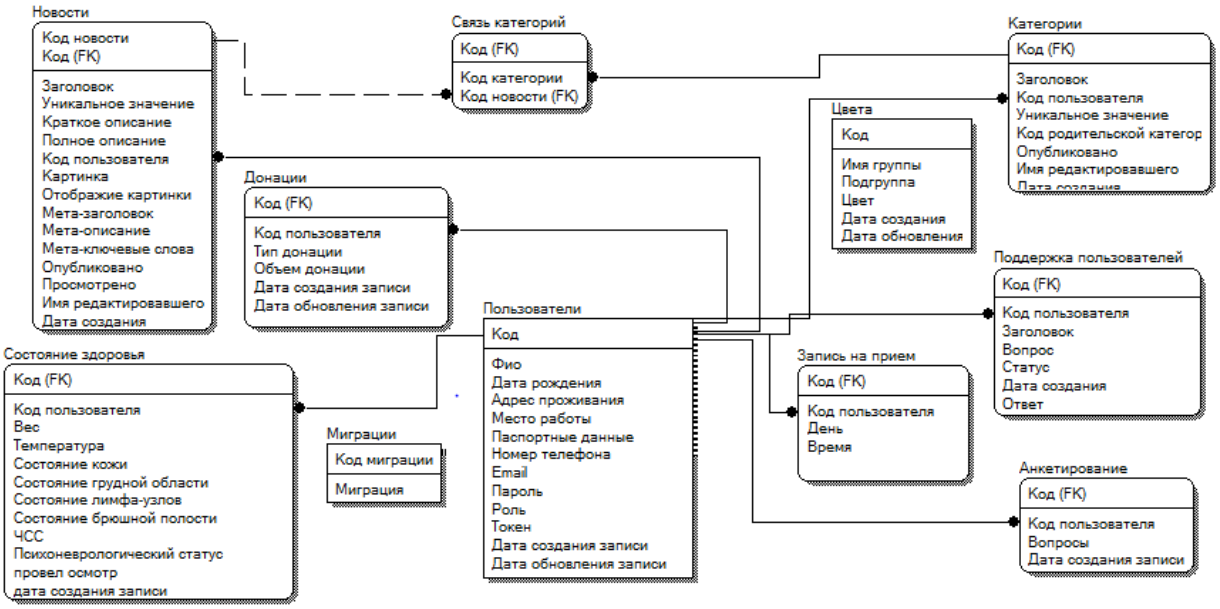

Рис. 2.6. Инфологическая модель базы данных

В Laravel, создание этих сущностей это выполнение миграций[4], но прежде чем приступать к созданию и выполнению миграций, нужно четко обозначить типы всех данных, которые указаны в инфологической модели.

1) Таблица Пользователи предназначена для хранения информации о донорах и сотрудниках, в связи с этим вводится целый ряд свойств:

- целочисленное свойство «id» является первичным ключом, т.е. уникальным значением для каждого пользователя;
- строковое свойство «fullname» содержит в себе ФИО пользователя;
- строковое свойство «liveadress» хранит адрес проживания пользователя;
- строковое свойство «workadress» хранит в себе место работы пользователя, либо адрес места работы;
- строковое свойство «passport» содержит в себе паспортные данные пользователя, а именно серию и номер паспорта;
- строковое свойство «phonenumber» хранит в себе номер телефона пользователя, выбран строковый тип, поскольку здесь могут быть

указаны, как личный номер мобильного телефона, так и номер стационарного телефона с места работы;
- строковое свойство «email» хранит email адрес пользователя, который и будет использоваться, как логин для входа в личный кабинет пользователями;
- строковое свойство «password» хранит пароль пользователя, пароль будет хранится в зашифрованном виде, такие требования к защите информации на сегодняшний день являются безоговорочно обязательными;
- целочисленное свойство «role» хранит значения, которые будут являться статусом пользователя, то есть по умолчанию при регистрации пользователю присваивается значение 0, что означает обычный пользователь;
- строковое свойство «token» хранит содержимое токена, который используется в Laravel для того, чтобы запоминать данные пользователя по желанию, что избавит их от постоянного ввода данных при авторизации;
- свойство «created_at» хранит дату и время создания пользователя;
- свойство «updated_at» хранит дату и время обновления данных пользователя.

2) Таблица Донации предназначена для хранения информации о донациях(кроводачах) совершаемых пользователями в Белгородской областной станции переливания крови, она содержит следующие свойства:
- целочисленное свойство «id» которое является первичным ключом, а по сути номером донации;
- целочисленное свойство «user_id» является вторичным(внешним) ключом, который содержит код пользователя, совершающего донацию;
- строковое свойство «type» хранит тип осуществляемой донации;
- целочисленное свойство «amountofblood» хранит объем осуществляемой донации в миллилитрах (мл);
- свойство «created_at» хранит дату и время создания донации;
- свойство «updated_at» хранит дату и время обновления записи о донации.

3) Таблица Цвета предназначена для хранения информации о состоянии «донорского светафора», который используется на странице приложения для отображения дефицита или профицита крови определенной группы или резуса, в связи с этим создаются следующие свойства:

- целочисленное свойство «id» которое является первичным ключом, т.е. номером индикатора светефора;
- строковое свойство «nameofgroup» хранит название группы крови, за которой закреплен индикатор;
- строковое свойство «subgroup» хранит название резуса-фактора группы крови, за которой закреплен индикатор;
- строковое свойство «color» хранит текущий цвет индикатора, который на используется и отображается светофором на главной странице приложения;
- свойство «created_at» хранит дату и время создания индикатора;
- свойство «updated_at» хранит дату и время обновления цвета индикатора.

4) Таблица Новости, содержит в себе новости и статьи, которые будут создаваться для того, чтобы динамически управлять содержим веб-сайта, для этого потребуются следующие свойства;

- целочисленное свойство «id» которое является первичным ключом, т.е. уникальным номером новости;
- строковое свойство «title» хранит заголовок новости или статьи;
- строковое свойство «slug» хранит в себе значение семантического URL новости или статьи;
- строковое свойство «description_short» хранит содержимое краткого описания новости или статьи;
- строковое свойство «description» хранит в себе полное содержимое публикации;
- строковое свойство «image» хранит URL адрес изображения, используемого в публикации;
- целочисленное свойство «image_show» хранит статус изображения, 0 если не отображено и 1 если отображено;
- строковое свойство «meta_title» хранит заголовок страницы в браузере, который отображается в выдаче поисковых систем;
- строковое свойство «meta_ description» хранит мета-описание содержимого публикации;
- строковое свойство «meta_ keyword» хранит ключевые слова, которые нужны для отображения публикации в поисковиках.
- целочисленное свойство «published» хранит статус публикации, 0 если не опубликована и в обратном случае 1;

- целочисленное свойство «viewed» хранит количество просмотров страницы;
- целочисленное свойство «created_by» хранит код пользователя, создавшего публикацию;
- целочисленное свойство «modified_by» хранит код пользователя, редактировавшего запись в последний раз;
- свойство «created_at» хранит дату и время создания публикации;
- свойство «updated_at» хранит дату и время внесения изменений в публикацию.

5) Таблица Категории предназначена для хранения категорий, за которыми закрепляются новости либо статьи, для этого нужны следующие свойства:

- целочисленное свойство «id» которое является первичным ключом, т.е. номером категории;
- строковое свойство «title» хранит заголовок категории;
- строковое свойство «slug» хранит в себе значение семантического URL категории;
- целочисленное свойство «parent_id» хранит значение родительской категории, если она существует;
- целочисленное свойство «published» хранит статус категории, 0 если не опубликована и в обратном случае 1;
- целочисленное свойство «created_by» хранит код пользователя, создавшего категорию;
- целочисленное свойство «modified_by» хранит код пользователя, редактировавшего категорию в последний раз;
- свойство «created_at» хранит дату и время создания категории;
- свойство «updated_at» хранит дату и время внесения изменений в категорию.

6) Таблица Категория-новость предназначена для регулирования отношения «многие ко многим», которое образовалось в связи с тем, что одной новости или статье может соответствовать несколько категорий и одна категория, может содержать несколько статей или новостей, в связи с этим выделяются следующие свойства:

- целочисленное свойство «category_id» которое является номером категории;
- целочисленное свойство «categoryable_id» которое является номером новости;

- строковое свойство «categoryable_type» хранит путь в приложении к статье либо новости.

7) Таблица Миграции предназначена для хранения информации о выполненных миграциях, это необходимо для того, чтобы контролировать миграции и при необходимости производить откат миграций, для этого выделены следующие свойства:

- целочисленное свойство «id» которое является первичным ключом, т.е. номером миграции;
- строковое свойство «migration» хранит информацию о миграции;
- целочисленное свойство «batch» хранит информацию о количестве откатов данной миграции.

8) Таблица Восстановление пароля предназначена для хранения уникального токена, по которому будет осуществляться переход на страницу для восстановления пароля, для этого нужны следующие свойства:

- строковое свойство «email» хранит email адрес пользователя, на которые будет отправлено письмо с ссылкой для восстановления пароля;
- строковое свойство «token» хранит ссылку, по которой в течении 1 часа пользователь сможет восстановить пароль;
- свойство «created_at» хранит дату и время отправки письма на email.

9) Таблица Контроль здоровья предназначена для хранения информации о состоянии доноров в день сдачи крови, для этого необходимы следующие свойства:

- целочисленное свойство «id» которое является первичным ключом, то есть номером записи;
- целочисленное свойство «user_id» которое является внешним ключом на таблицу Users;
- целочисленное свойство «weight», вес тела человека в день сдачи крови;
- свойство с плавающей точкой «temperature», для хранения информации о температуре тела;
- строковое свойство «skin», для описания состояния кожи;
- строковое свойство «chss», показывающее количество сердечных сокращений;
- строковое свойство «limfa», для описания состояния лимфоузлов;

- строковое свойство «belly», для описания брюшной полости;
- строковое свойство «chest», для описания грудной области;
- строковое свойство «neuropsychiatric», для описания психоневрологического статуса донора;
- строковое свойство «cooperator», содержащее фио врача, проводившего осмотр;
- свойство «created_at», содержащее дату создания записи.

10) Таблица Поддержка пользователей, для ответа на вопросы пользователей, задаваемые через личный кабинет. Для этого необходимо выделить следующие свойства:

- целочисленное свойство «id» которое является первичным ключом, то есть номером записи;
- целочисленное свойство «user_id» которое является внешним ключом на таблицу Users;
- строковое свойство «title», содержащее заголовок вопроса или его тип;
- строковое свойство «issue», содержащее сам вопрос;
- целочисленное свойство «status» которое показывает статус вопроса, то был ли на него отправлен ответ или нет;
- свойство «created_at», хранящее в себе дату создания записи;
- строковое свойство «answer», содержащее ответ на вопрос.

11) Таблица онлайн записи на прием, для ее реализации нужны следующие свойства:

- целочисленное свойство «id» которое является первичным ключом, то есть номером записи;
- целочисленное свойство «user_id» которое является внешним ключом на таблицу Users;
- свойство «day», которое будет хранить зарезервированную дату;
- свойство «time», которое будет хранить время;

12) Таблица анкетирования, куда будут сохраняться данные после прохождения пользователями анкетирования, для этого нужны следующие свойства:

- целочисленное свойство «id» которое является первичным ключом, то есть номером записи;
- целочисленное свойство «user_id» которое является внешним ключом на таблицу Users;
- строковое свойство «question», которое содержит вопрос

- свойство «created_at», которое содержит дату прохождения анкетирования.

2.3 Программная реализация приложения

Как я уже отмечал ранее, для того, чтобы создать базу данных, необходимо использовать миграции. На рисунке 2.7 приведен список миграций, необходимых для создания описанной мною выше базы данных.

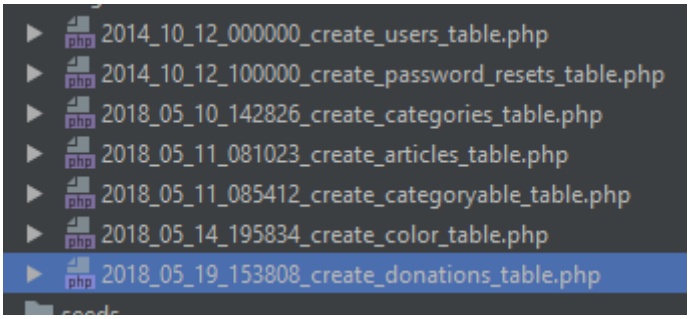

Рис. 2.7 Список миграций

Создание миграций по отдельности обусловлено тем, что по необходимости, можно будет откатывать изменения. В миграциях указываются поля таблиц, их модификаторы, значения по умолчанию, связи с другими таблицами.

Использование такого подхода позволяет контролировать версии приложения, отменять нежелательные изменения, которые появились в результате ошибочных изменений приложения или его структуры. Пример одной из миграций приведен на листинге 2.1

Листинг 2.1 Реализация миграции для создания таблицы Пользователи

```
public function up()
 {
```

```
Schema::create('users', function (Blueprint $table) {
    $table->increments('id');
    $table->string('fullname');
    $table->string('dateofbirth');
    $table->string('liveadress');
    $table->string('groupofblood')->nullable()->default('newdonor');
    $table->string('workadress')->nullable();
    $table->string('passport')->unique();
```

Конец листинга 2.1.

Такой подход, очевидно, крайне удобен и прост в использовании, метод increments автоматически сгенерирует последовательность для первичного ключа, который этот же метод и создает. Метод unique задает уникальность для выбраного поля, в данной системе уникальными являются следующие поля: email, номер телефона, паспортные данные. Метод default задает значение по умолчанию.

В классе User описаны переменные, которые мы будем передавать через контроллер, они разделены на 2 группы, общие и скрытые, это нужно для того, чтобы пароль и токен были защищены.

Теперь необходимо создать контроллер, который будет управлять моделью User. Для создания контроллера можно использовать команду php artisan make:controller либо же просто создать новый файл. В контроллере необходимо подключить уже созданный нами ранее класс и описать методы для работы с данными.

Создание контролера UserController можно осуществить несколькими путями, создать пустой файл с именем UserController и поместить в директиву с контроллерами либо же выполнить команду php artisan make:controller UserController –resource, получить готовый контролер с подключенными фасадами для работы с данными и объявленными методами для работы с моделью User. Для работы с этой моделью нам потребуется 5

методов, index() для передачи данных о пользователях в представление, store() для добавления новых пользователей, update() для редактирования пользователей, edit(), для передачи данных пользователя в форму для редактирования и destroy() для удаления пользователя.

Описав методы работы контролера с моделью, теперь необходимо настроить взаимодействие контролера с представлением. Для этого необходимо создать это представление. В Laravel поддерживается шаблонизация приложения, поэтому достаточно лишь создать некий шаблон, который можно будет использовать на любых страницах приложения. Поскольку контроллер UserControl создан для работы с данными пользователей, необходимо ограничить доступ к будущим страницам системы:

Листинг 2.2 Реализация ограничения доступа пользователей

```
@if(Auth::user()->role !== '1' and Auth::user()->role !== '2' )
{{exit("Доступ ограничен")}}
@endif
```

Конец листинга 2.2.

По умолчанию при создании пользователя, значения атрибута role равняется 0, что свидетельствует о статусе обычного пользователя, так же имеются роли со значениями 1 и 2, модератор и администратор соответственно. Уставив данную проверку, был ограничил доступ обычным пользователям к разделу администрирования системы, поскольку все страницы админ панели будут наследоваться от представления app_admin.blade.php, в котором установлена эта проверка, что делает недоступной админ панель для всех пользователей со нулевой ролью.

Представление – есть ни что иное как страница с HTML разметкой. Для отображения данных пользователей удобно использовать таблицу, которая будет динамически генерироваться в зависимости от загружаемых данных.

Чтобы реализовать динамическое формирование таблицы нужно использовать цикл foreach(), в котором будет содержаться разметка вместе с получаемыми данными. Ниже приведет листинг динамического формирования таблицы.

Листинг 2.3 Реализация динамического формирования таблицы

```
@forelse($users as $user)
<tr style="background-color: whitesmoke;color: black;border-color: blue">
<td style="border-color: blue" align="center">{{$user->id}}</td>
<td style="border-color: blue">{{$user->fullname}}</td>
<td style="border-color: blue">{{$user->email}}</td>
…
 @endforelse
```

Конец листинга 2.3.

Таким образом, в цикле динамически формируется таблица, в которую передаются данные из базы данных.

Аналогичным образом создаются представления для других моделей. Реализация отчетности Белгородской областной станции переливания крови, требует создания двух моделей и контролеров: категории и материалы. Аналогично выше созданному классу создаются классы категорий и материалов, в которых указываются данные, с которыми классы взаимодействуют. Одной из сложностей реализации взаимодействия этих двух моделей является тот факт, что связь между ними классифицируется как связь «многие ко многим» потому , как одна категория может содержать несколько материалов, и в то же время, один материал может быть закреплен за несколькими категориями. Есть несколько решений, с помощью которых можно уйти от связи многие ко многим, однако в Laravel все намного проще.

Существуют готовые методы, которые помогут установить взаимосвязь между данными моделями, такие методы, как: morphedByMany(), что

обеспечит категориям связь «многие ко многим» с новостями и morphToMany(), что обеспечит связь новостям с категориями.
Пример использования приведен ниже на листинге 2.4

Листинг 2.4 Реализация связи между моделями категории и материалы

```
public function categories()
   {
      return $this->morphToMany('App\Category','categoryable');
   }
 public function articles()
   {
      return $this->morphedByMany('App\Article','categoryable');
   }
```

Конец листинга 2.4.

Таким образом, категории и материалы теперь являются взаимосвязанными. Немаловажной частью реализации категорий и материалов является создания уникального атрибута slug, который является URL маршрутом, по которому можно будет просмотреть содержимое материала либо категории. Но в Laravel это не является сложной задачей, ниже приведена функция, которая будет генерировать необходимый slug. Реализация функции показана на листинге 2.5

Листинг 2.5 Реализация функции генерации URL маршрута slug

```
 public function setSlugAttribute($value) {
 $this->attributes['slug'] = Str::slug(mb_substr($this->title,0,40) . "-" .
\Carbon\Carbon::now()->format('dmyHi'),'-');
 }
```

Конец листинга 2.5.

Таким образом, выполняется требование, чтобы каждая новость открывалась на отдельной странице при выборе.
Благодаря динамической загрузки состояния индикаторов светофора, без особого труда получатся менять их при помощи Update запроса.

На очереди реализация личного кабинета пользователя, в котором будет хранится его персональная данные и информация о его деятельности на Белгородской станции переливания крови. Все так же создаем представление при помощи HTML разметки. Для того, чтобы отобразить его персональные данные, вместо обычного запроса к базе данных, я воспользуюсь удобной альтернативой. Классом Авторизации пользователей.
В Laravel, когда пользователь успешно проходит авторизацию, данный класс хранит всю информацию о данном пользователе, которая хранится в таблице Пользователи. Поэтому нужно всего лишь обратиться к нему за этой информацией и отобразить ее. Реализация показана на листинге 2.8

Листинг 2.6 Реализация отображения информации в личном кабинете

```
{{Auth::user()->phonenumber}}
{{Auth::user()->email}}
{{Auth::user()->groupofblood}}
{{Auth::user()->workadress}}
```

Конец листинга 2.6.

Фигурные скобки используются в Larvel для того, чтобы использовать PHP код в HTML разметке. Однако для того, чтобы посчитать количество донаций, совершенных данным пользователем все же придется выполнить запрос к базе данных. Выполнение запроса показано на листинге 2.7

Листинг 2.7 Запрос на получение данных о количестве донаций

```
{{$stat1 = DB::table('donations')->select(DB::raw('count(*)'))-
>where('user_id', Auth::user()->id)->where('type','Кроводача')-
>value('count(*)')}}">
{{$stat2 = DB::table('donations')->select(DB::raw('count(*)'))-
>where('user_id', Auth::user()->id)->where('type','Плазмаферез')-
>value('count(*)')}}">
```

Конец листинга 2.7.

Таким образом пользователь сможет узнать свою статистику на Белгородской областной станиции переливания крови.

Следующим шагом является реализация рассылки уведомлений на почту, для этого необходимо создать новый контроллер. Реализация функции показана на листинге 2.8

Листинг 2.8 Функция рассылки уведомлений

```
  public function send(Request $a)
{
$input = $a->only('a');
$user1 = DB::table('users')->where('id',$input)->value('email');
Mail::send(['text' => 'mail'],['name','Администрация ОБГУЗ'],function
($message) use ($user1){
$message->to($user1)->subject('test email');
$message->from('sagradyan96@gmail.com','Администрация ОБГУЗ1');});
return redirect()->route('admin.user_managment.control.index'); }
```

Конец листинга 2.8.

Фасад mail обеспечивает работу с рассылкой сообщений на почту через SMTP.

ГЛАВА 3. ТЕСТИРОВАНИЕ

3.1 Программа и методика тестирования

В данной главе описывается алгоритм и проведение тестирования разработанной ранее «Автоматизированной системы учета доноров Белгородской станции переливания крови».

В первой главе был указан весь перечень требований к разрабатываемой системе, ниже приведены ключевые требования:

- построение отчетности деятельности станции;
- объявление приближающихся мероприятий и предоставление необходимой о них информации;
- предоставление информации, которая необходима для изучения процедуры сдачи крови и ее необходимости;
- возможность хранить и изменять данные о донорах, осуществлять быстрый доступ к данным по средствам сети интернет;
- возможность удаленного анкетирования;
- возможность пользователям удаленно осуществлять запись на прием и обращения к сотрудникам станции через личный кабинет.

В соответствии с указанными требованиями был разработан следующий алгоритм проведения тестирования «Автоматизированной системы учета доноров Белгородской станции переливания крови»:

1. Протестировать работоспособность системы для неавторизованного пользователя:
    1.1 Протестировать возможность просмотра новостей.
    1.2 Протестировать возможность просмотра статей.

1.3 Протестировать возможность регистрации.

2. Протестировать работоспособность системы для авторизованного пользователя в роли донора:
    2.1 Протестировать возможность перехода в личный кабинет.
    2.2 Проверить отображение личных данных донора в кабинете.
    2.3 Проверить отображение истории донаций, осуществленных донором.
    2.4 Протестировать возможность предварительной записи на прием через личный кабинет.
    2.5 Протестировать возможность обращения к сотрудникам через поддержку в личном кабинете.
    2.6 Протестировать возможность анкетирования в личном кабинете.
3. Протестировать работоспособность системы для авторизованного пользователя в роли администратора.
    3.1 Протестировать возможность перехода в панель управления системой.
    3.2 Протестировать модуль управления контентом, а именно добавление новостей, статей, категорий.
    3.3 Протестировать модуль управления донорским светофором, размещенным на главной странице веб-приложения.
    3.4 Протестировать модуль работы контроля личных данных доноров.
    3.5 Протестировать модуль контроля данных о здоровье доноров.
    3.6 Протестировать модуль поддержки пользователей.
    3.7 Протестировать модуль удаленного анкетирования для доноров.

3.8 Протестировать модуль электронной записи доноров на прием.

3.2 Тестирование

Тестирование «Автоматизированной системы учета доноров Белгородской станции переливания крови» необходимо выполнить согласно алгоритму проведения тестирования, который был описан ранее.

На рисунке 3.1 показана главная страница приложения, вход на которую осуществляется путем ввода адреса в браузерной строке, либо через любой интернет поисковик.

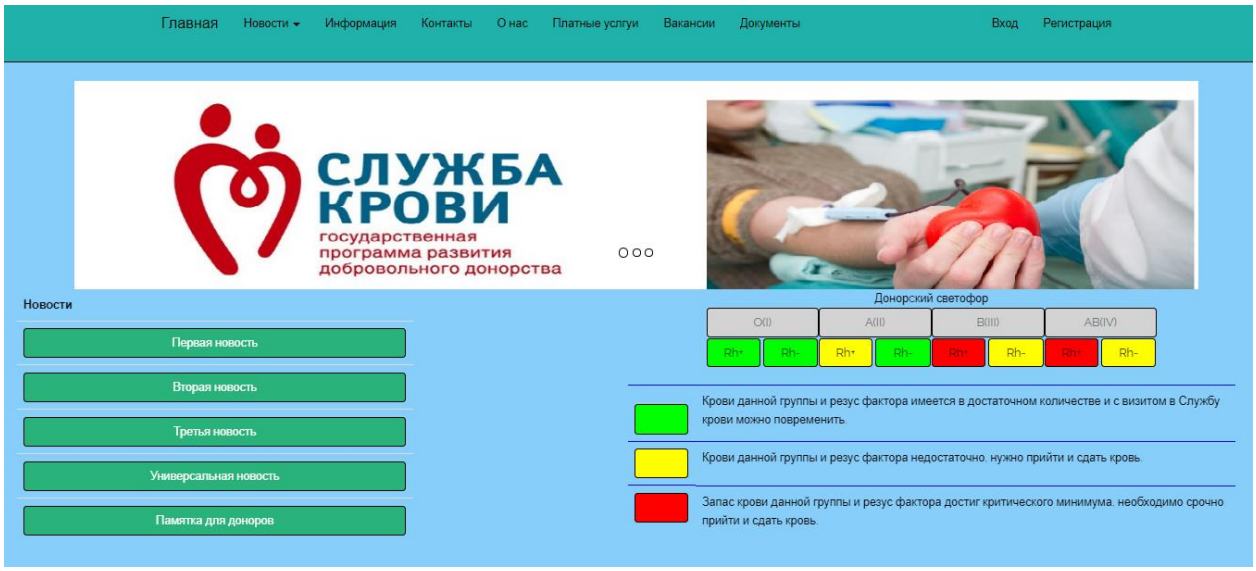

Рис. 3.1. Главная страница приложения

Неавторизованному пользователю доступа вся информация, отображенная на рисунке 3.1. В левой нижней части экрана отображены 5 последних добавленных новостей. На рисунке 3.2 показан просмотр содержимого новости «Памятка для доноров».

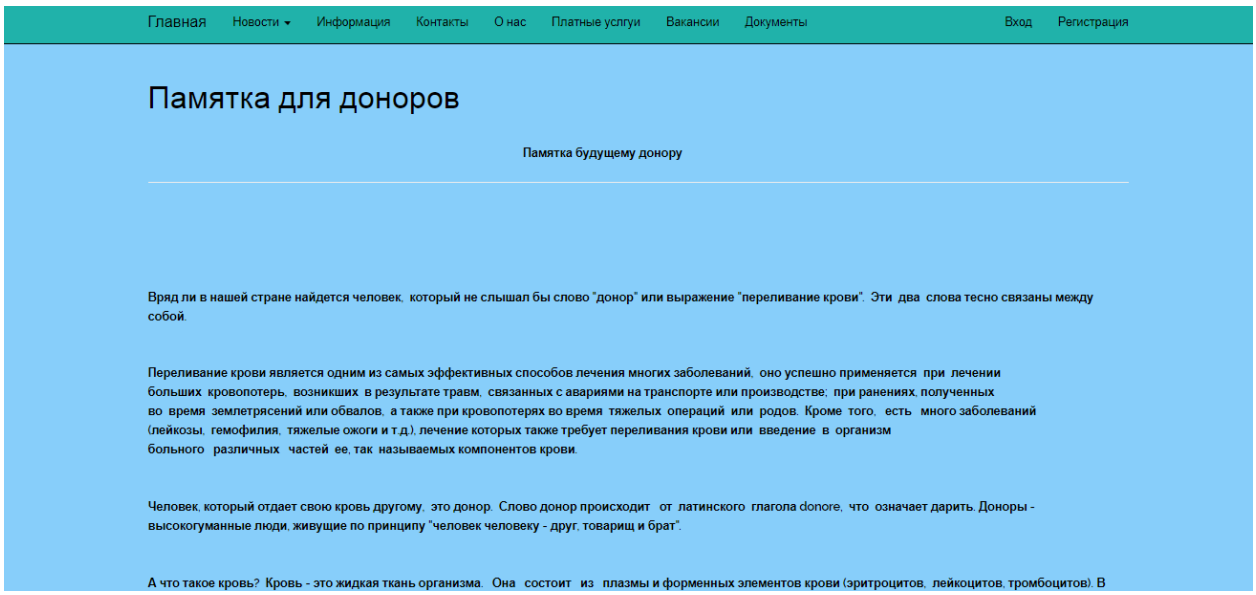

Рис 3.2. Содержимое новости «Памятка для доноров»

Следующим шагом будет тестирование просмотра информационных статей, доступным всем посетителям веб-приложения. На рисунке 3.3 отображается содержимое вкладки «Информация», которая размещена в верхней части любой страницы веб-приложения.

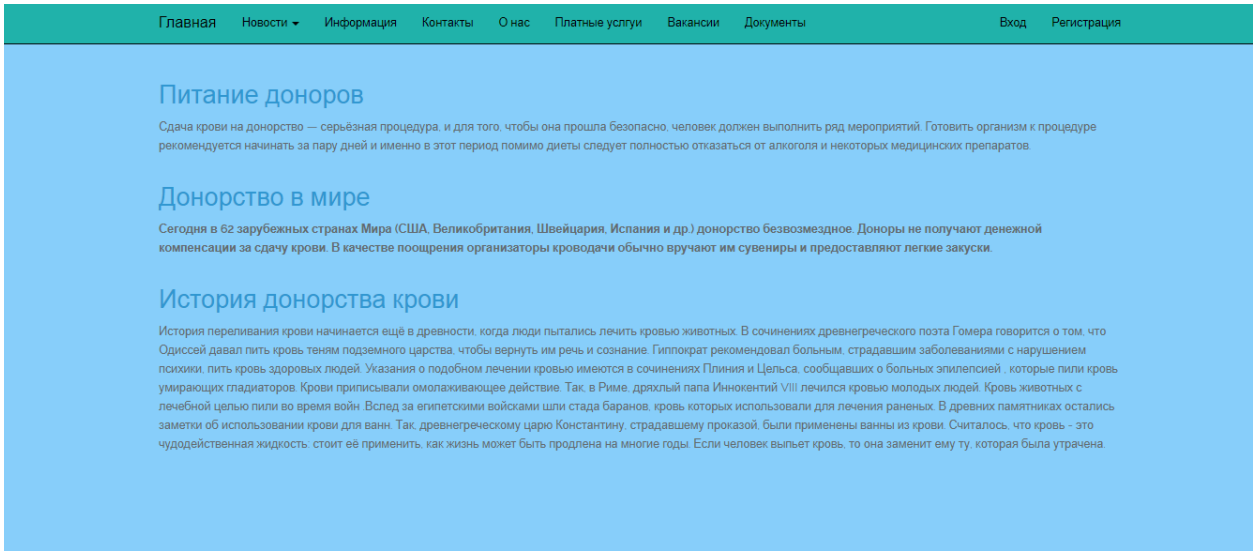

Рис. 3.3. Просмотр статей в категории «Информация»

Теперь необходимо протестировать возможность регистрации пользователей, для этого необходимо нажать на ссылку «Регистрация» размещенной в

верхней правой части приложения. На рисунке 3.4 показан результат перехода по данной ссылке.

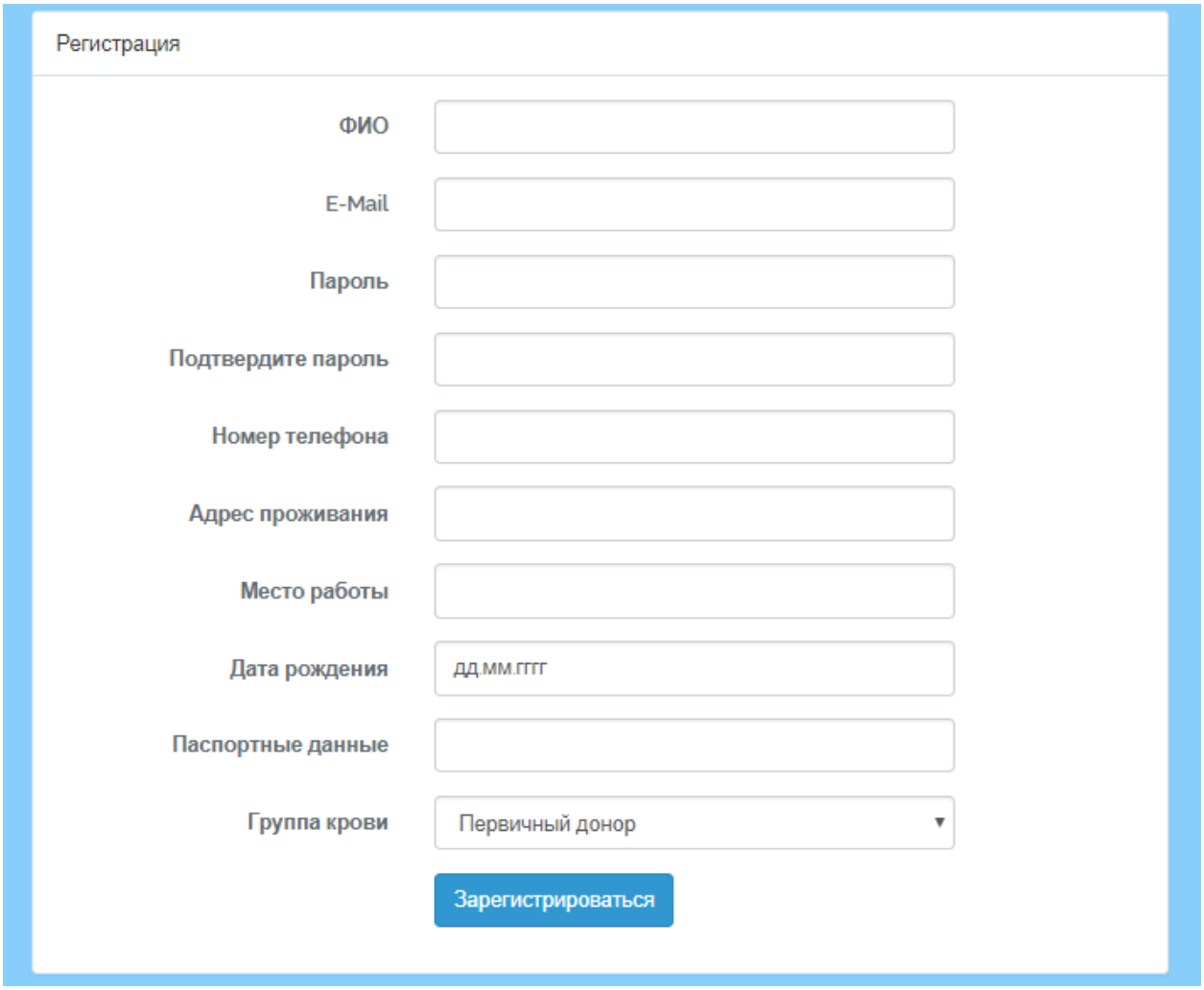

Рис. 3.4. Страница регистрации пользователей

Теперь введем тестовые данные и проверим работоспособность данной функции. В качестве тестовых данных будут использованы вымышленные данные, но корректно введенные и соответствующие требованиями показанных выше полей формы регистрации пользователей в разработанной системе учета доноров. На рисунке 3.5 показана заполненная данными страница.

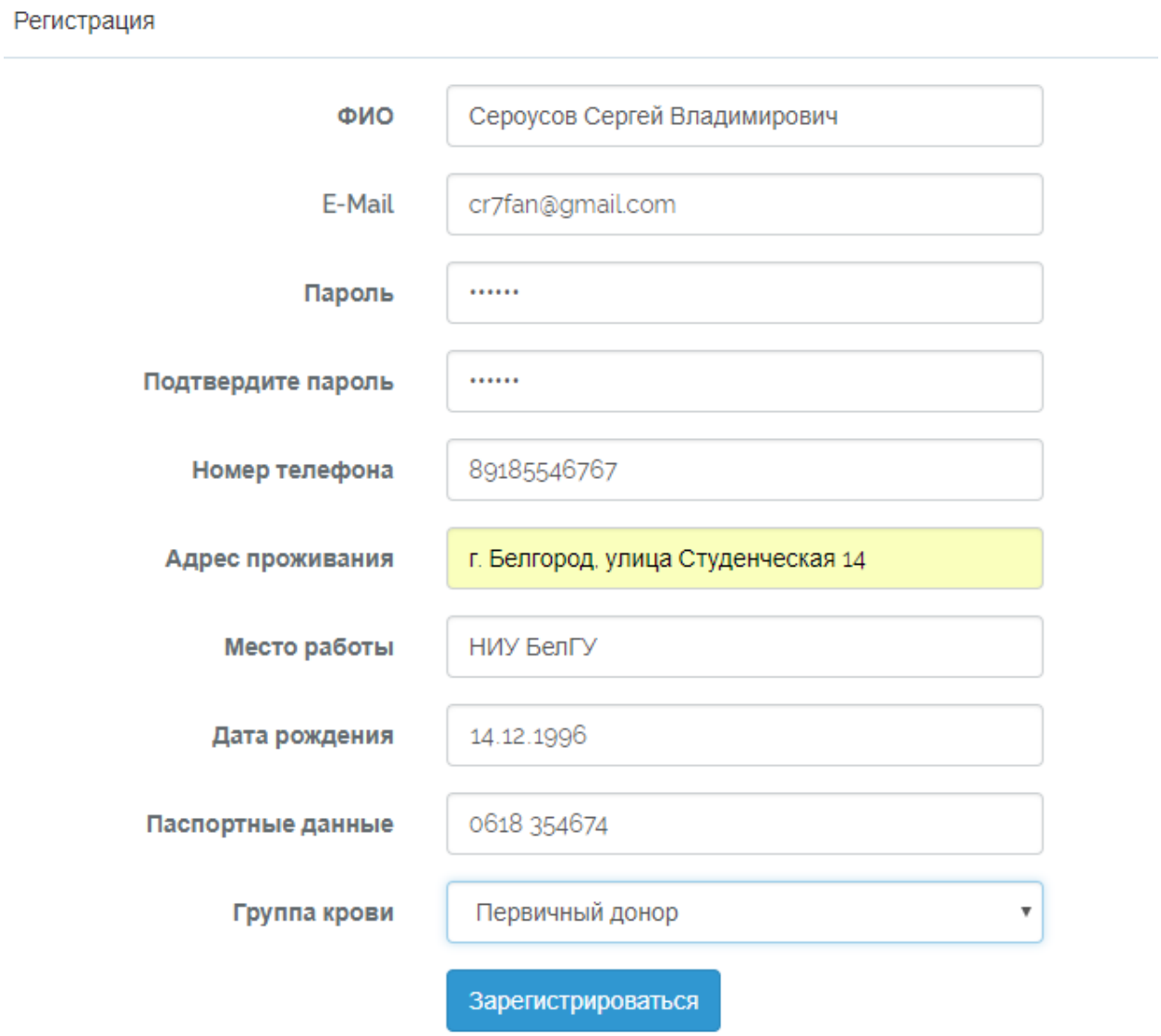

Рис. 3.5. Прохождение регистрации

При ошибочном вводе каких-либо данных, либо при пропуске, после перезагрузки страницы появится сообщение, в котором будет сказано где именно допущена ошибка, а так же все введенные данные, кроме пароля, будут отображены, что позволит сэкономить время. Позже эти данные будут уточнены, когда пользователь явится на станцию и предъявит документы требуемые документы.

После нажатия на кнопку зарегистрироваться пользователь перенаправляется в личный кабинет, где отображается имеющаяся информация о нем. Это продемонстрированно на рисунке 3.6

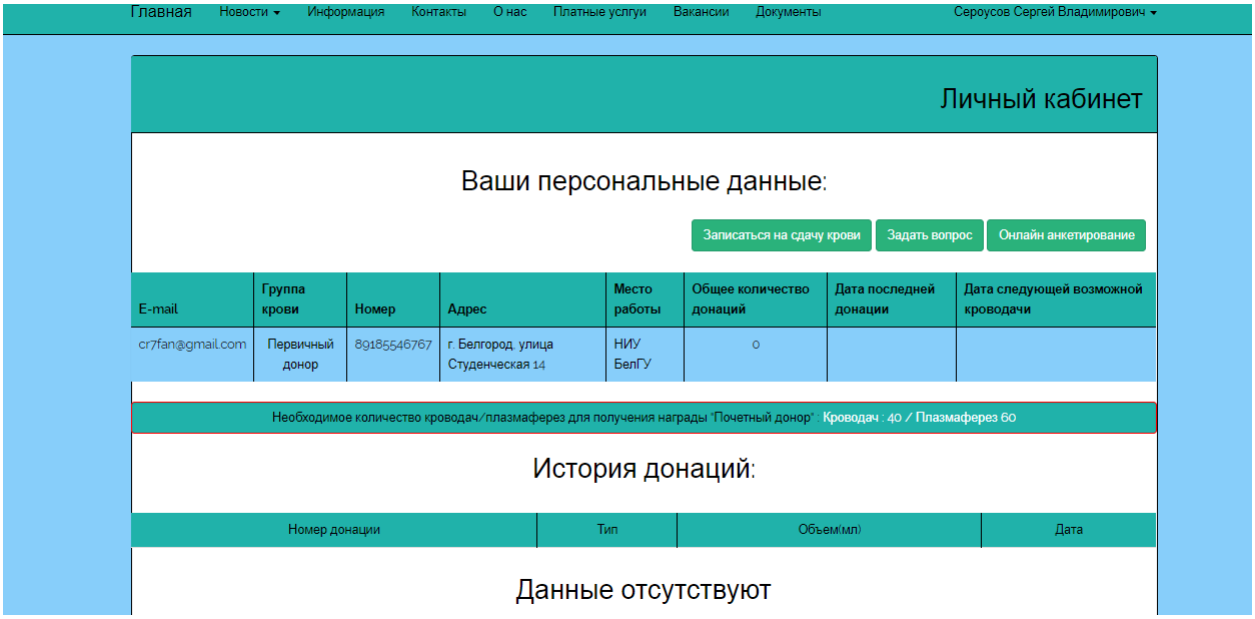

Рис. 3.6. Личный кабинет пользователя

Как показано на рисунке 3.6, в личном кабинете отображается личная информация пользователя, так же здесь имеется история донаций, которая на данный момент пустая, поскольку пользователь только что зарегистрировался. Как видно на этом рисунке, пользователю предлагаются следующие функции: записаться на сдачу крови, задать вопрос и пройти онлайн анкетирование. Начнем с тестирования функции предварительной записи на сдачу крови. Результат перехода показан на рисунке 3.7.

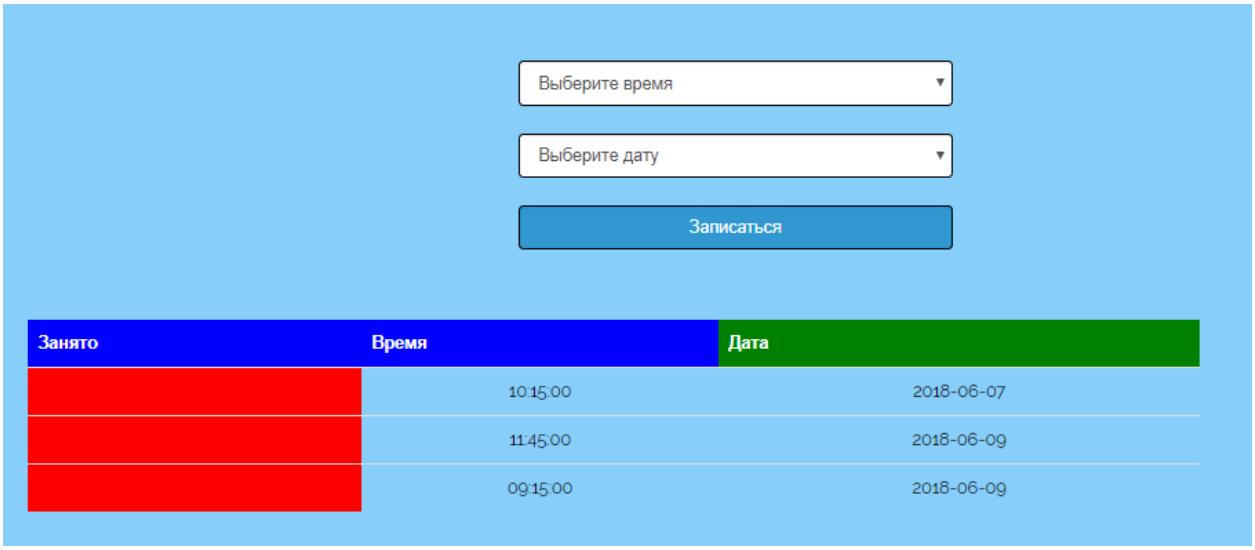

Рис. 3.7. Запись на сдачу крови через личный кабинет

Как показано на рисунке 3.7, пользователю доступна форма, где предлагается выбрать время и дату для записи на прием, также ниже приведена таблица, где отображены уже занятые время и дата для записи. Осуществим запись на завтрашний день на 12:00 часов. Результат показан на рисунке 3.8.

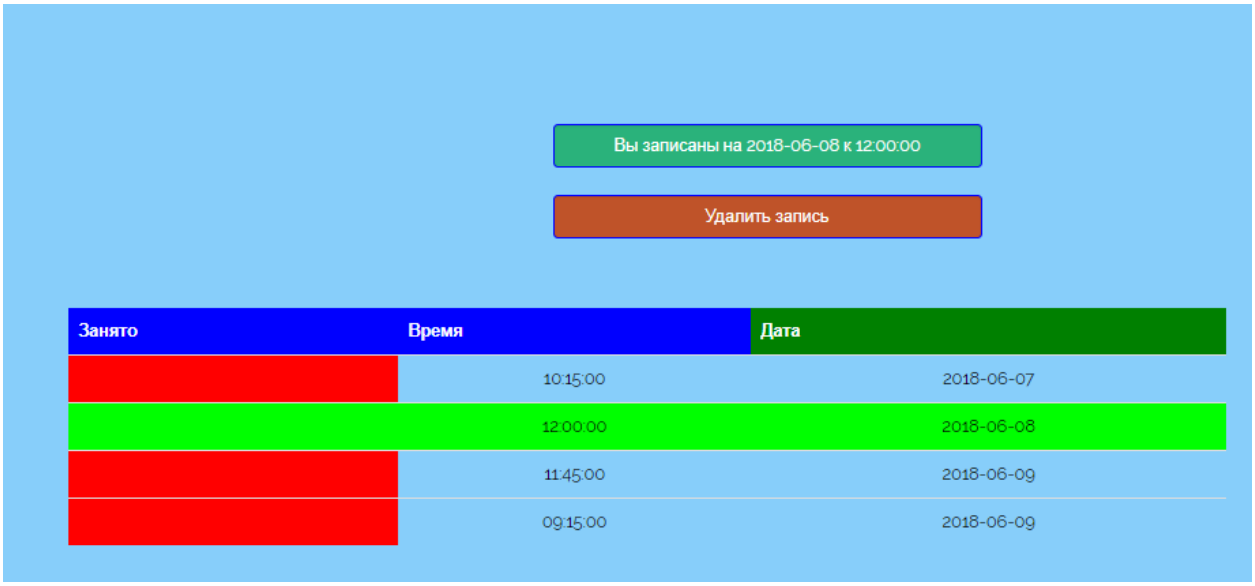

Рис. 3.8. Тестирование записи на прием.

Как показано на рисунке 3.8, удалось записаться на выбранное время и запись отображается в общей таблице записей. Помимо этого ваша запись подсвечивается зеленым цветом. В верхней части страницы показано сообщение, в котором сообщается на какой день и время вы записались, так же предлагается функция удаления записи. Если у пользователя возникнет необходимость изменить время, либо отменить запись вовсе, то эта функция будет ему весьма кстати. Записаться можно только на текущий день, завтрашний либо 2 дня вперед, система сама высчитывает даты, пользователю нужно лишь выбрать сегодня, завтра либо послезавтра. Если эти даты приходятся на выходные, форма предупредит об этом и не зарегистрирует пользователя на указанное время. Теперь протестируем функцию удаления записи из очереди. Результат отображен на рисунке 3.9.

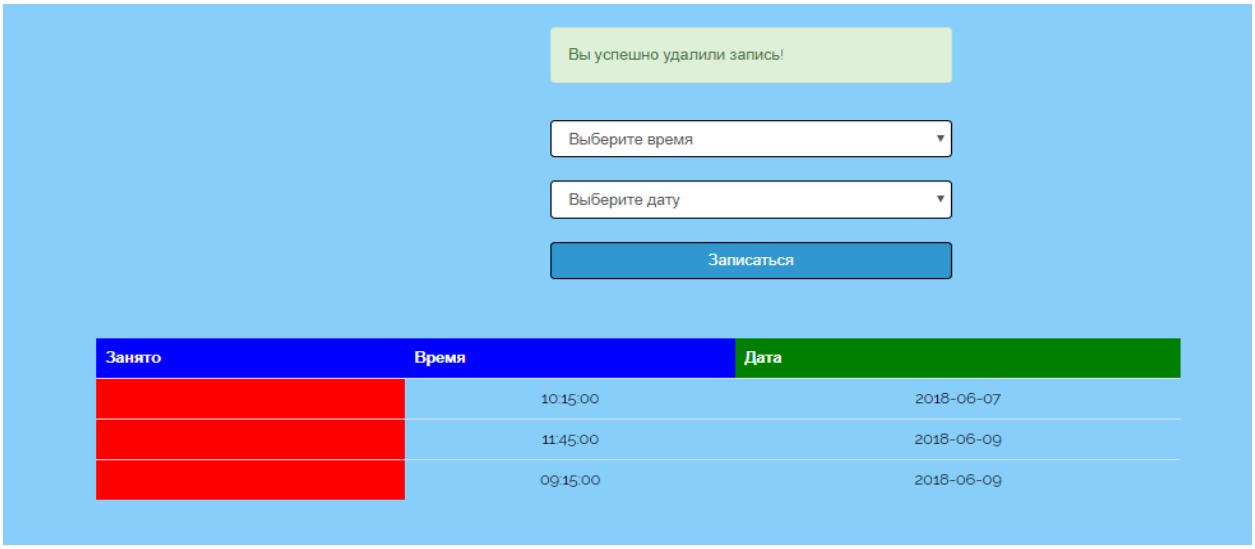

Рис. 3.9. Результат удаления записи.

Как видно на рисунке 3.9, пользователю показано сообщение о том, что он успешно удалил запись и ему вновь становится доступной возможность записаться.

Теперь необходимо протестировать функцию обращения к сотрудникам, для этого нужно перейти в личный кабинет и нажать на кнопку «Задать вопрос». На рисунке 3.10 показана страница поддержки пользователей.

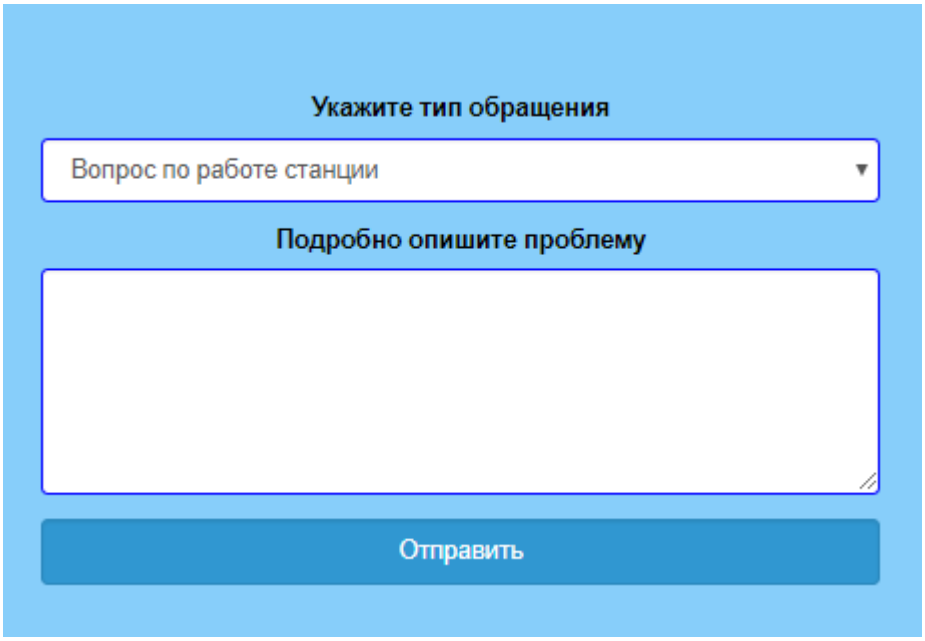

Рис. 3.10. Форма отправки обращения.

На рисунке 3.10 показана форма в которой предлагается выбрать тип обращения и описать проблему, для тестирования укажем тип «Другой» и отправим следующий вопрос « До скольки часов можно пройти процедуру сдачи крови завтра?». Результат отправки данного обращения показан на рисунке 3.11.

Рис. 3.11. Результат отправки обращения.

Как показано на рисунке, вопрос был отправлен и отображается в таблице на той же странице, так же здесь присутствует поле «Ответ», в котором будет отображен ответ сотрудников.

Далее протестируем функцию удаленного анкетирования. Для этого нужно перейти в личный кабинет и нажать на кнопку «Онлайн анкетирование». Результат показан на рисунке 3.12.

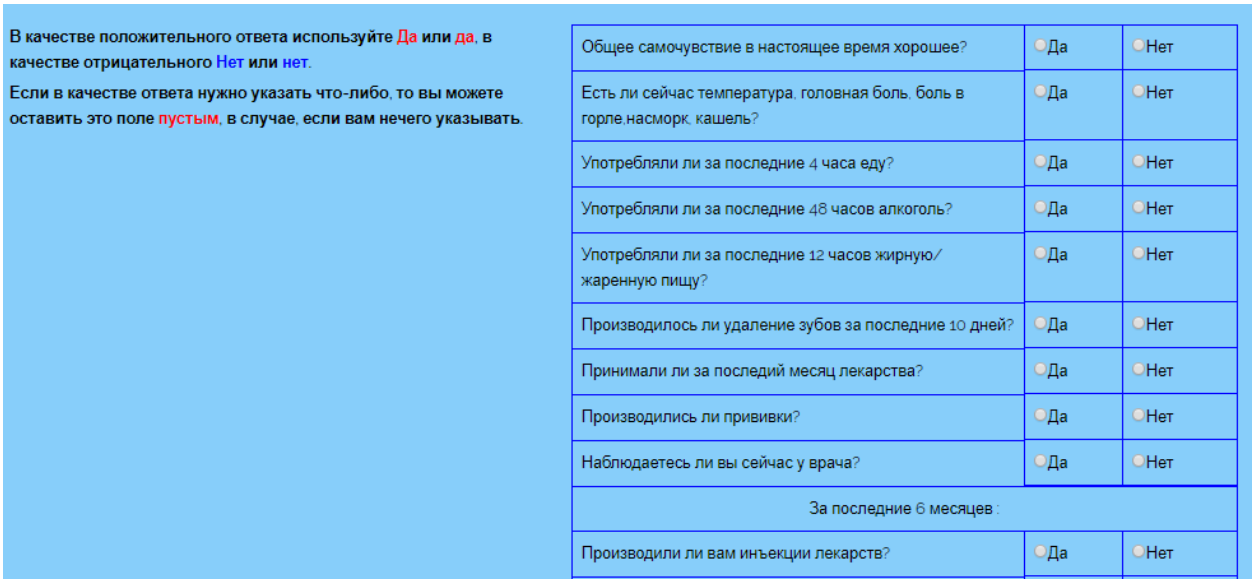

Рис. 3.12. Функция удаленного анкетирования

Как показано на рисунке 3.12, на странице анкетирования отображается форма для заполнения анкеты, а слева от нее подсказка для ввода. Нужно заполнить анкету и нажать на кнопку «Отправить». После отправки, пользователь перемещается в личный кабинет. При повтором нажатии на кнопку, пользователю показывается сообщение о том, что прошло недостаточно времени с момента последней кроводачи. Если пользователь заполнил анкету случайно, то он может написать в поддержку с просьбой удалить его анкету. Поскольку анкетирование является обязательной процедурой перед осуществлением кроводачи, позволить пользователю удалять их не является целесообразным.

Следующий этап тестирования это проверка работоспособности панели управления приложением. При регистрации пользователю автоматически присваивается нулевой статус, что определяет его в группу простых пользователей, администраторы обладают статусом равным двум. При нажатии на ссылку именованную именем пользователя и расположенную в верхней правой части приложения, появляется выпадающий список, в котором обычному пользователю доступны личный кабинет и выход из системы. Однако у администраторов появляется еще одна ссылка, которая перемещает его в панель управления приложением, это показано на рисунке 3.13.

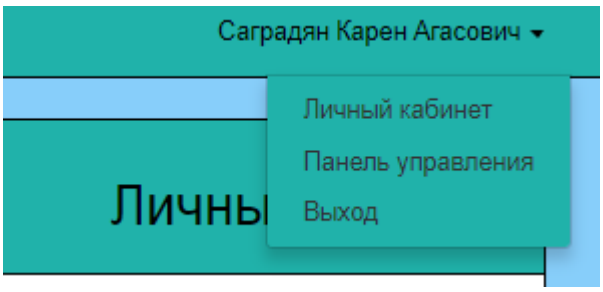

Рис. 3.13 Ссылка для перехода в панель управления приложением

После перехода по этой ссылке отрывается панель управления. Переход между модулями управления приложением осуществляется путем выбора

модулей, ссылки на которые расположены в верхней части веб-приложения. Это продемонстрировано на рисунке 3.14.

Рис. 3.14. Панель управления приложением

Как показано на рисунке, в верхней части приложения расположены следующие модули: управление новостями, контроль донаций, управление светофором и управление пользователями.

Начнем тестирование с управления новостями, как показано на рисунке 3.14, администратору предлагается создать категорию либо материл, также выведено общее количество категорий и материалов. Кроме того выводятся 5 последних добавленных материалов и категорий, для быстрого редактирования в случае необходимости. Под каждой категорией показано количество материалов, прикрепленных к ней, а под каждым материалом показана категория, за которой материал закреплен.

Протестируем функцию добавления категорий, для этого нажнем на кнопу «Создать категорию». Результат показан на рисунке 3.15.

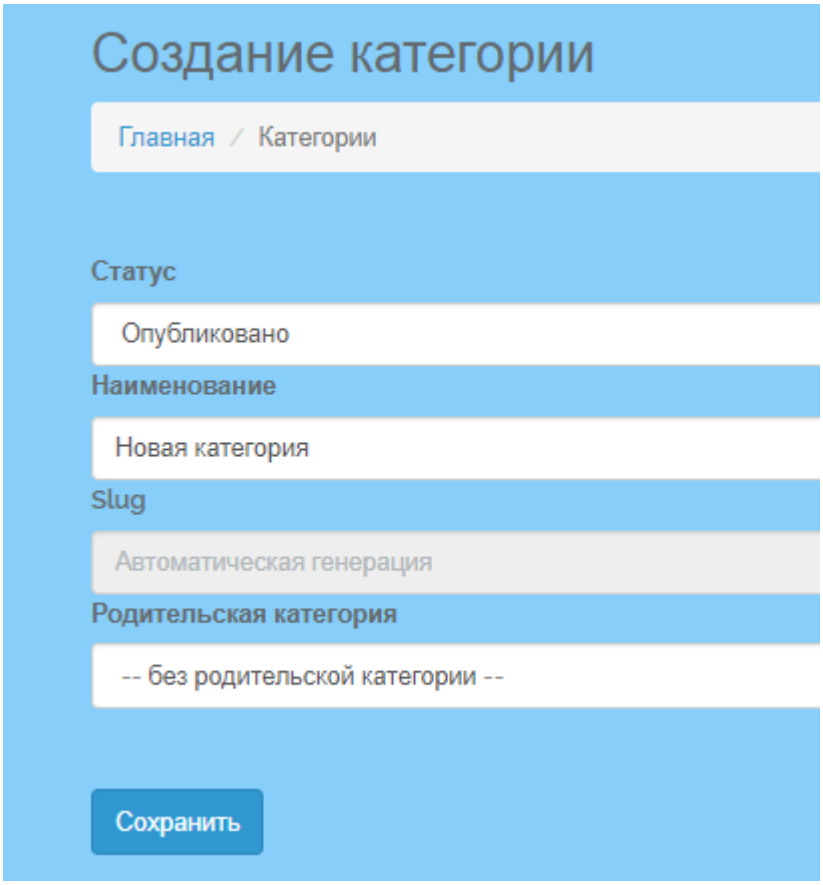

Рис. 3.15. Создание категории

На рисунке показано, что администратору предоставляется возможность выбрать статус опубликовано или не опубликовано, что влияет на отображение категории и вложенных в нее материалов на странице веб-приложения.

Кроме того, необходимо ввести наименование категории и при необходимости выбрать родительскую категорию для данной категории. Slug, то есть URL маршрут к просмотру содержимого категории генерируются автоматически.

После нажатия на кнопку сохранить, администратор переходит на страницу, где отображены все существующие категории. Это показано на рисунке 3.16.

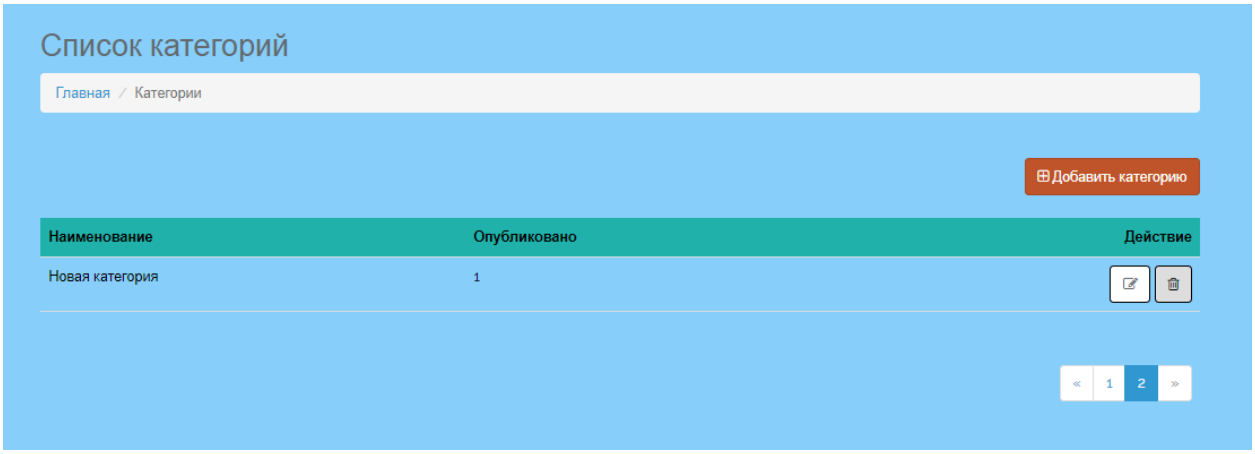

Рис. 3.16 Просмотр созданной категории

Администратору предоставляются функции редактирования и удаления категорий.

Теперь необходимо протестировать добавление материала. Для этого нужно перейти на главную страницу модуля управления контентом и нажать на кнопку «Добавить материал». На рисунке 3.17 показана форма добавления материала.

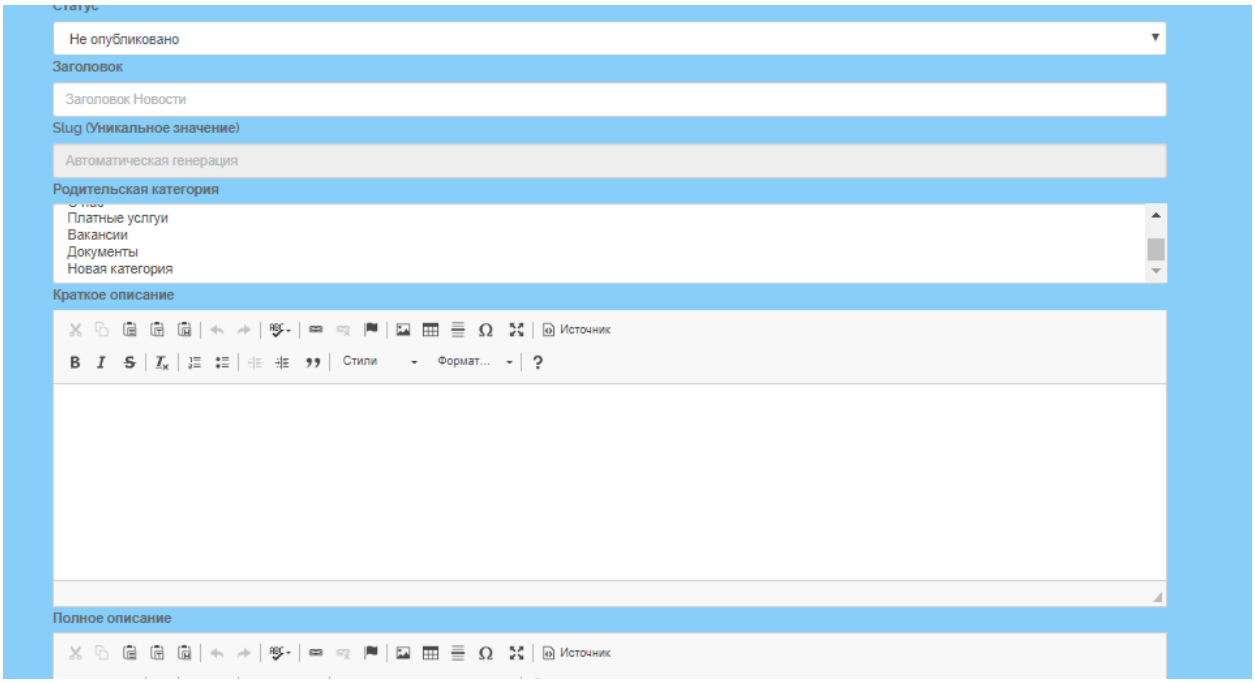

Рис. 3.17. Добавление материала

Здесь также предлагается выбрать статус материала, указать заголовок и категорию, ввести краткое описание, которое будет отображаться на странице содержания категории, затем ввести полное описание и указать мета данные для того, чтобы страницу с содержанием могли находить интернет-поисковки.

После нажатия на кнопку «сохранить», администратора перенаправляет на страницу, где отображены все материалы, это продемонстрировано на рисунке 3.18

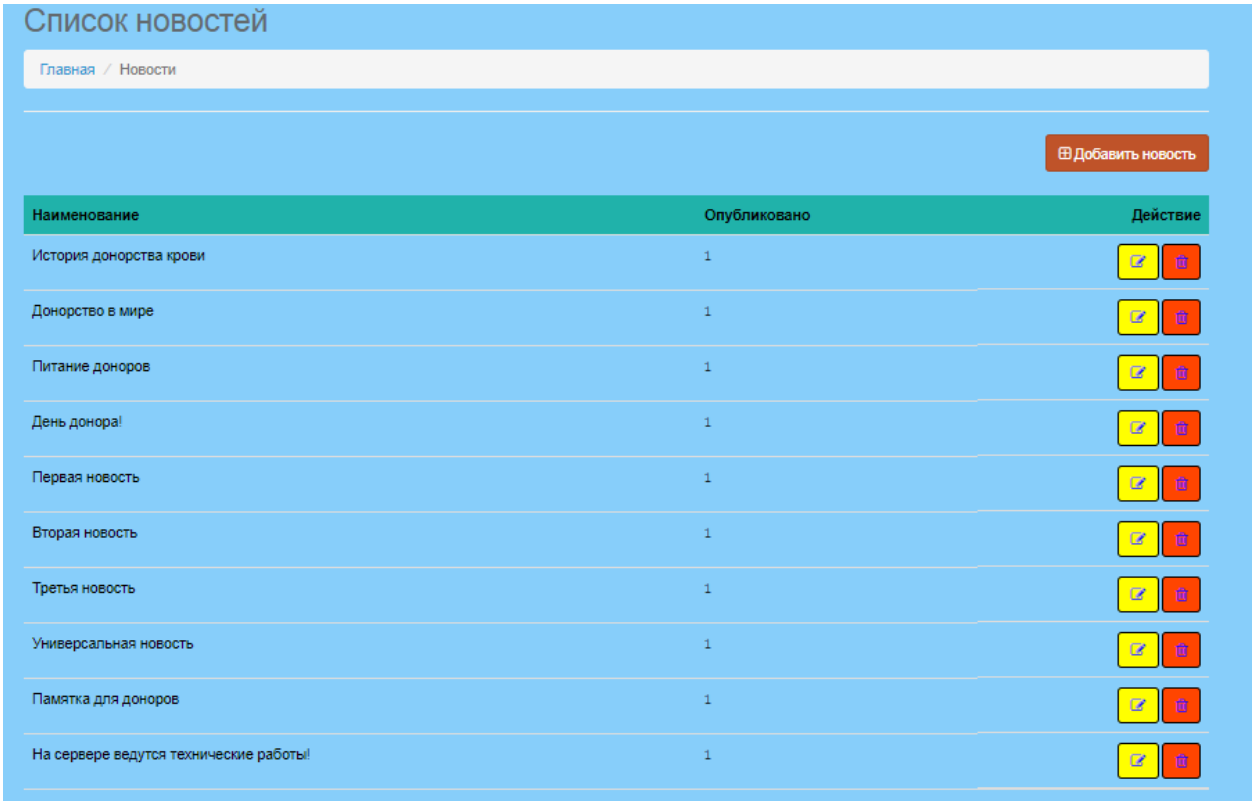

Рис. 3.18 Отображение всех новостей

Как показано на рисунке 3.18, администратор может добавить, удалить, либо отредактировать уже созданный материал.

Далее, согласно алгоритму тестирования, необходимо протестировать модуль управления донорским светофором, который расположен на главной странице веб-приложения. Страница модуля показана на рисунке 3.19.

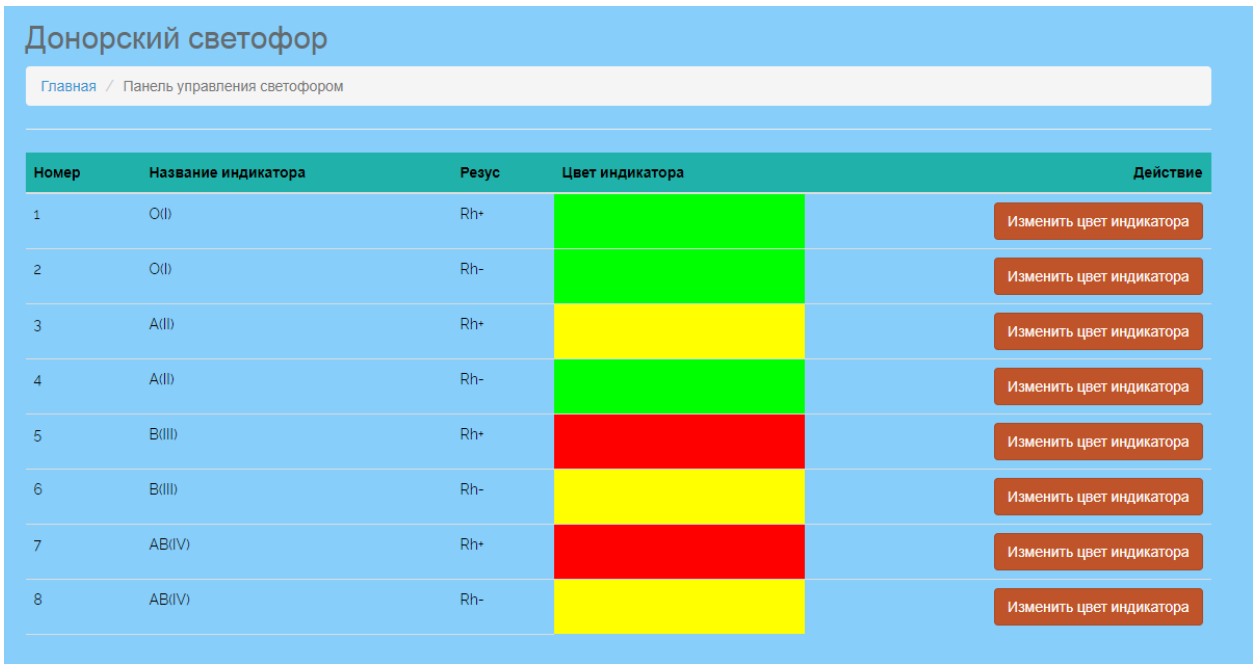

Рис. 3.19. Модуль управления донорским светофором

На данной странице отображается текущие состояние донорского светофора и предоставляется возможность изменить его, согласно актуальной информации, поступающей из банка крови. Форма изменения цвета показана на рисунке 3.20.

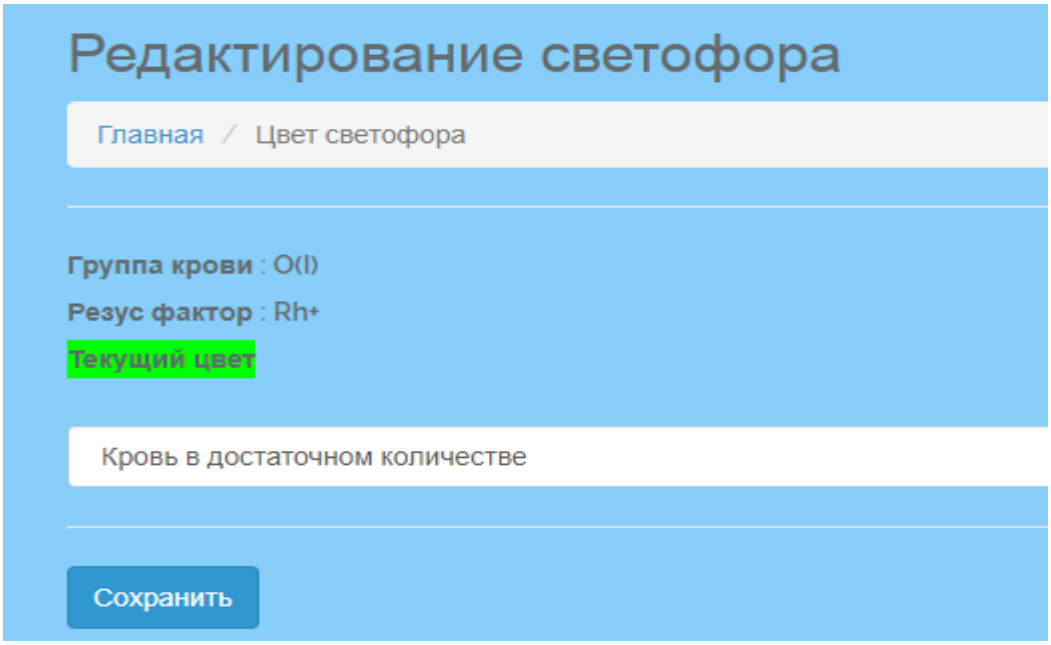

Рис. 3.20. Изменение состояния светофора

После изменения цвета индикатора светофора, в модуле управления светофором изменяется цвет состояния, так же как и на главной странице приложения.

Следующий шаг, согласно алгоритму тестирования, это тестирование модуля контроля личных данных пользователей. Для этого необходимо перейти по ссылке «Контроль данных», результат показан на рисунке 3.21.

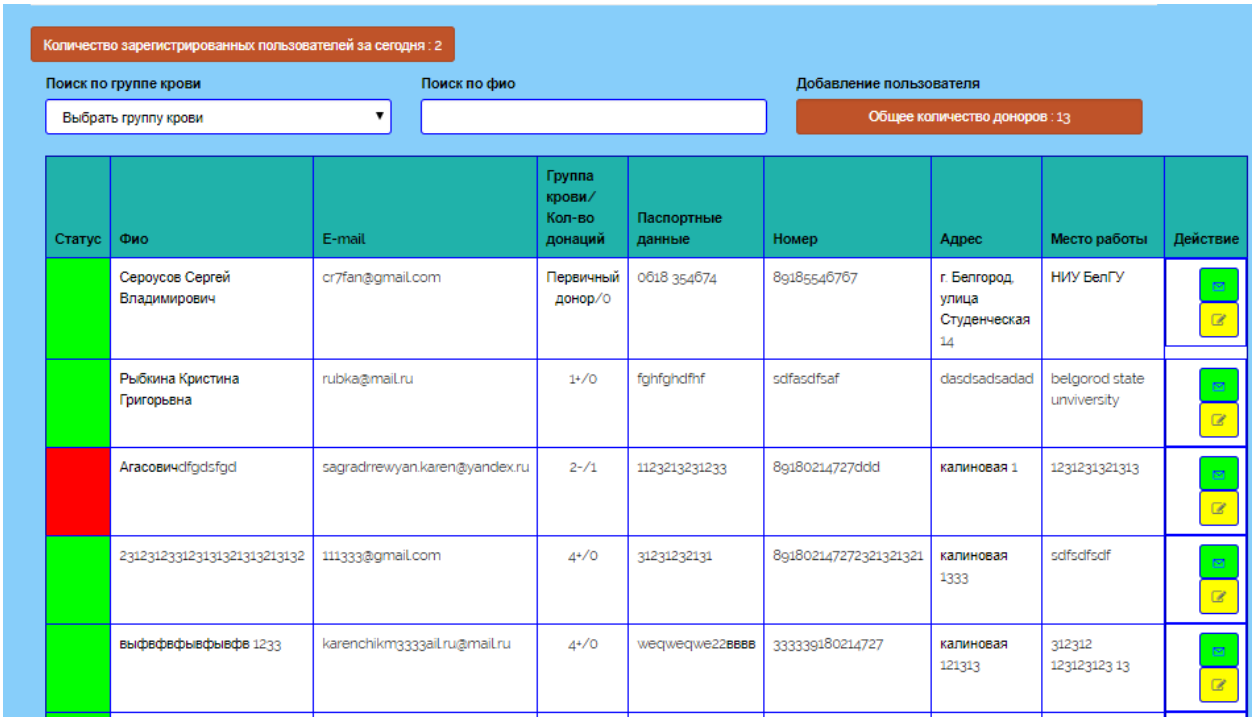

Рис. 3.21. Модуль контроля личных данных пользователей

В данном модуле хранятся такие данные пользователей, как Email, группа крови, общее количество донаций, паспортные данные, номер, адрес и место работы. В крайней левой части таблицы находится столбец «Статус», который имеет два цвета: красный и зеленый. Если пользователь осуществлял кроводачи менее 60-ти дней назад, индикатор имеет красный цвет. В крайней правой части расположены две кнопки: желтая и зеленая. Первая соответственно открывает форму изменения данных пользователя, а вторая отправляет на его email адрес уведомление о том, что донору нужно посетить Белгородскую станцию переливания крови.

Помимо этого в верхней части страницы модуля доступны такие функции, как добавление нового пользователя, поиск по фио и поиск по группе крови. Также выведена статистика, в которой содержится информация о количестве всех пользователей, зарегистрированных в системе и о количестве пользователей зарегистрированных за текущий день.

Осуществим тестирование поиска по группе крови, указав первую положительную группу крови. Результат показан на рисунке 3.22.

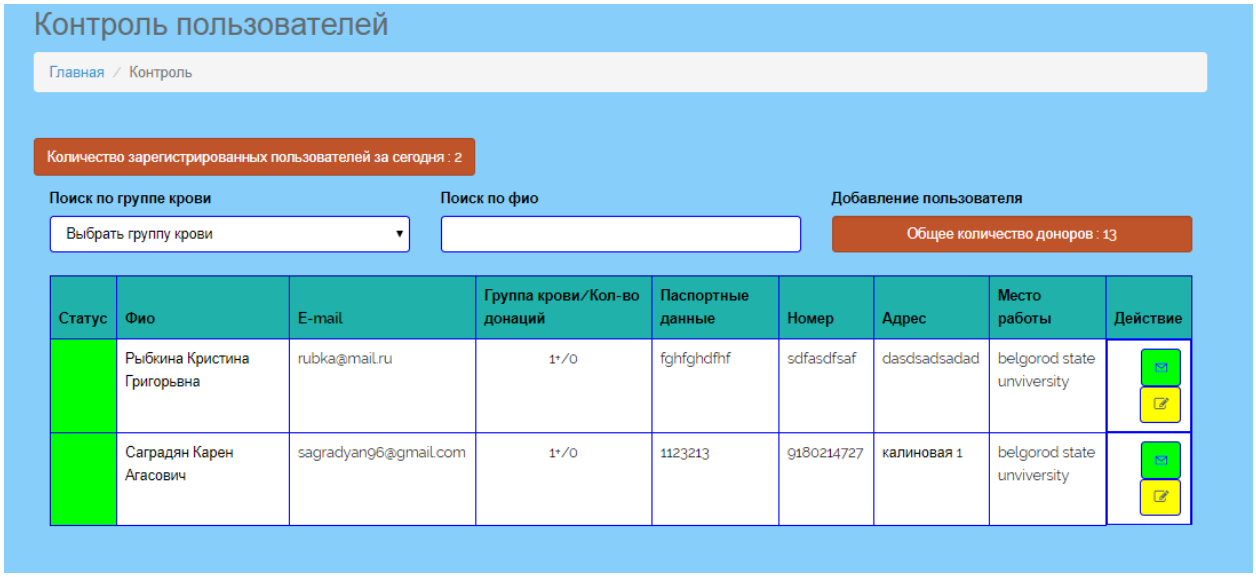

Рис. 3.22. Поиск пользователей по 1+ группе крови

Поиск осуществлен, теперь протестируем отправку уведомления на электронный почтовый адрес. Просмотр содержимого сообщения показан на рисунке 3.23.

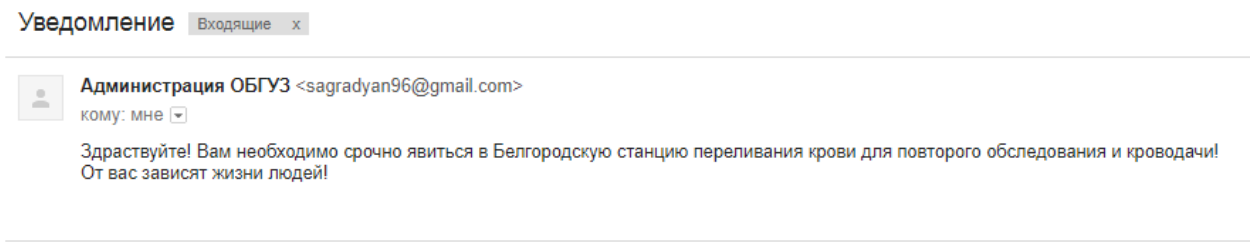

Рис. 3.23. Просмотр полученного сообщения-уведомления

Следующим шагом тестирования является проверка работоспособности модуля поддержки пользователей, то есть ответа на вопросы, которые они могут присылать из личного кабинета. На рисунке 3.24 показана страница с данным модулем.

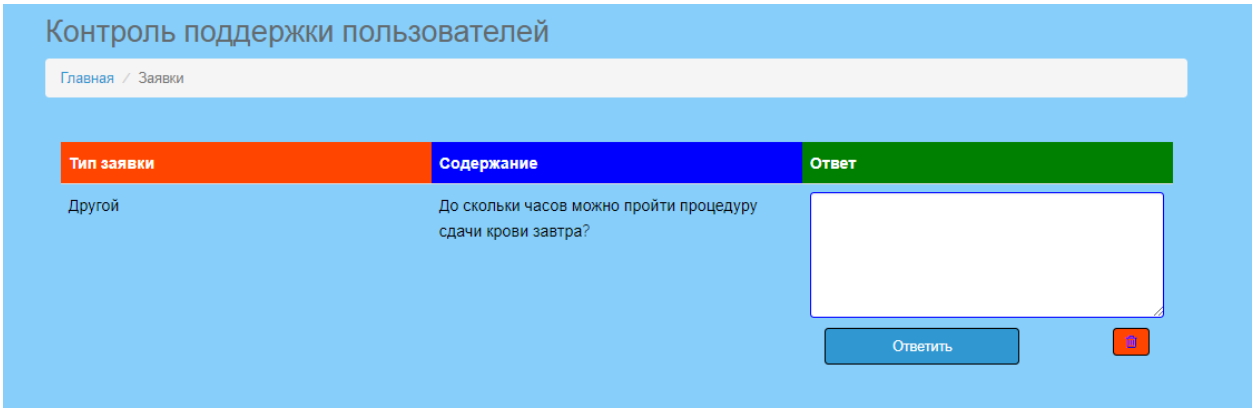

Рис. 3.24. Модуль поддержки пользователей

Как показано на рисунке 3.24, здесь отображается тип заявки, содержание и возможность ввода ответа, либо удаления заявки. Пользователь, отправивший вопрос, может увидеть ответ своем личном кабинете.

Теперь необходимо проверить работоспособность модуля удаленного анкетирования, для этого нужно перейти на страницу модуля. На рисунке 3.25 показана данная страница

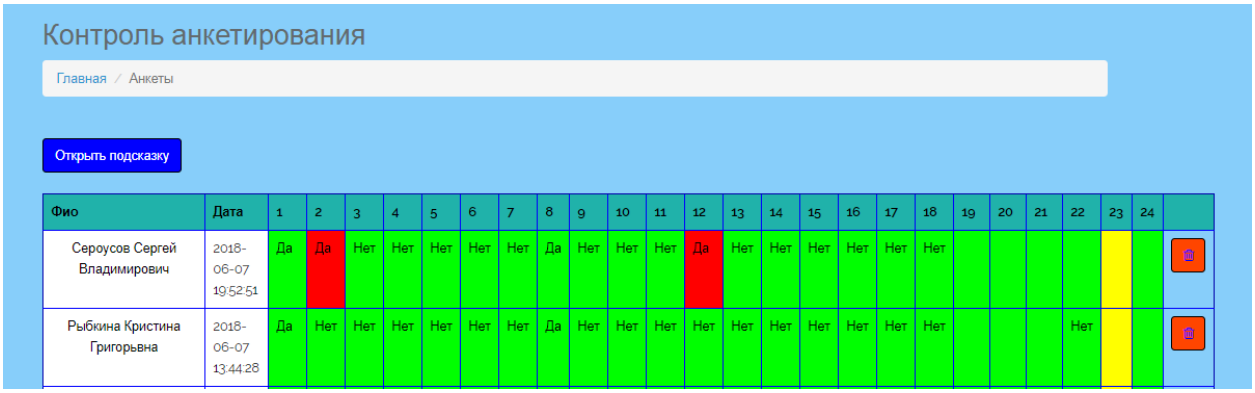

Рис. 3.25. Модуль удаленного анкетирования

Как показано на рисунке 3.25, на данной странице отображается фио пользователя, дата отправки анкеты, ее содержимое, кнопки удаления и подсказки. Поскольку многие ответы на вопросы представляют собой выбор да или нет вариантов ответа, то была разработана подсветка ответов и в случае неудовлетворительного ответа, он помечается красным цветом. В подсказке содержатся пронумерованные вопросы, на которые отвечал пользователь, содержимое подсказки показано на рисунке 3.26.

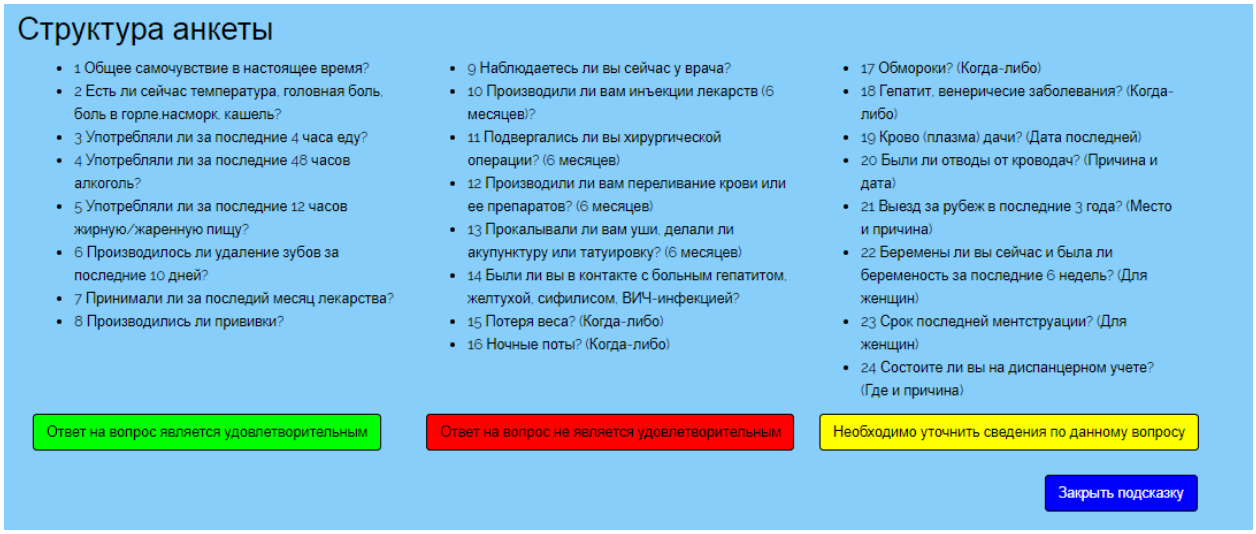

Рис. 3.26. Открытие подсказки

После открытия подсказки, в верхней части страницы подгружается список, который показан на рисунке 3.26, а также указано значение цветов подсветки вариантов ответов.

Следующий этап тестирования, это тестирование модуля удаленной записи на прием. Для этого нужно перейти на страницу модуля. На рисунке 3.27 показана страница модуля.

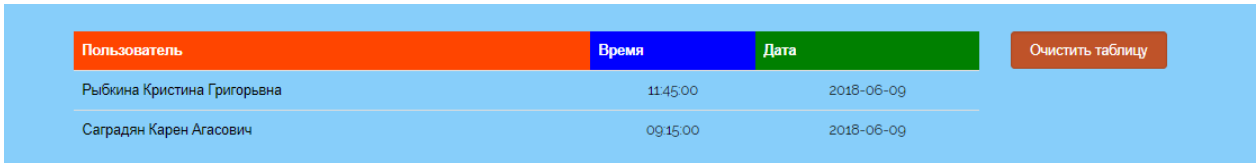

Рис. 3.27. Модуль удаленной записи на прием

Здесь отображаются записи, оставленные пользователями. Таким образом сотрудники могут принимать пользователей, которые записались удаленно, с особым приоритетом. Также имеется кнопка очистки таблицы.

Последним этапом тестирования является тестирование модуля добавления донаций. Страница модуля отображена на рисунке 3.28.

Рис. 3.28. Модуль контроля донаций

Как видно на рисунке 3.28, на странице модуля отображена информация не только о донациях, но и показана статистика, которая содержит в себе: количество донаций, осуществленных за текущий день, количество донаций, осуществленных за весь период работы системы, общий объем крови и ее компонентов, полученных за сегодняшний день и за весь период работы системы, кроме того, отдельно высчитывается объем компонентов крови. Также отображены кнопки добавления и удаления записей. Перейдем на страницу добавления донации. Результат показан на рисунке 3.29.

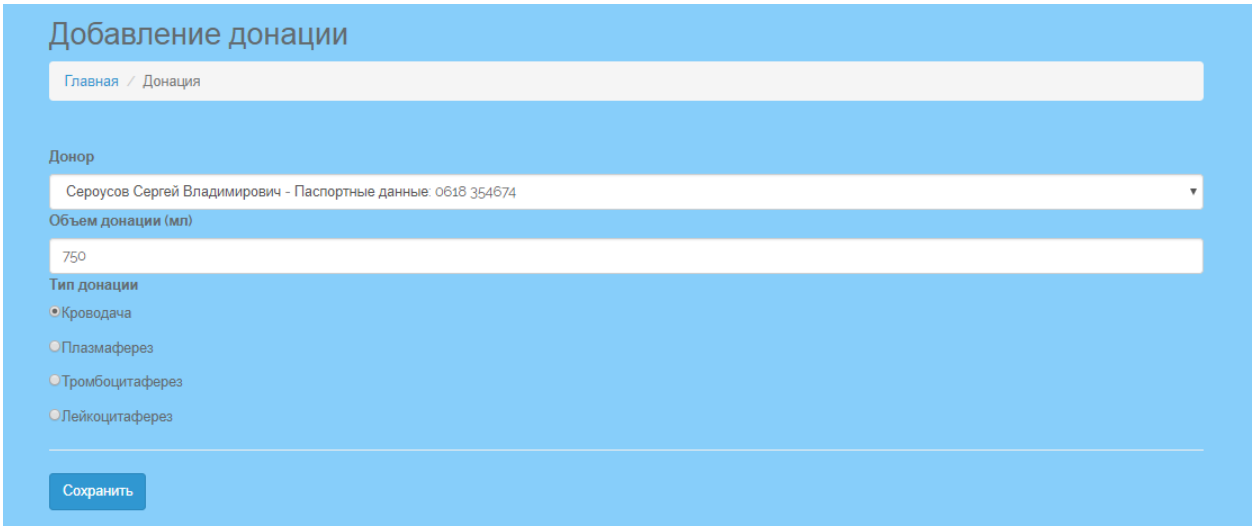

Рис. 3.29. Добавление донации

На странице добавления предлагается выбрать донора из списка всех доноров, зарегистрированных в системе, объем осуществелнной донации в миллилитрах и тип донации. После нажатия на кнопку «сохранить», администратор автоматически переход на страницу донаций, где уже присутствуют новые данные.

По результатам проведения тестирования разработанной «Автоматизированной системы учета доноров Белгородской станции переливания крови» было установлено, что система является работоспособной и готова к использованию. Функциональные возможности данный системы в полном объеме удовлетворяют все требования, предъявленные к разрабатываемой системе в первой главе документа.

ЗАКЛЮЧЕНИЕ

В начале проектирования системы была поставлена следующая цель – спроектировать и разработать автоматизированную систему учета доноров Белгородской станции переливания крови. Для этого были выделены следующие задачи:

- анализ деятельности организации;
- выявление ключевых направлений деятельности организации, требующих автоматизации;
- обзор и подробный анализ систем, выполняющих схожие задачи;
- постановка требований к разрабатываемой системе;
- проектирование системы учета;
- реализация системы учета;
- проведение испытаний.

По итогу выполнения дипломной работы была спроектирована и разработана автоматизированная система учета доноров Белгородской станции переливания крови.

Данная система реализована в виде веб-приложения, которое выполняет не только функции веб-представительства данной станции переливания, но и позволяет пользователем удаленно осуществлять работу с ней. Сотрудникам предоставлен удобный и понятный функционал для ведения учета доноров и их данных. Возможность находить доноров определенной группы крови и рассылать им уведомления может помочь получить драгоценную кровь в кротчайшие сроки.

По итогам тестирования разработанной системы учета, можно сделать следующий вывод – веб приложение полностью удовлетворяет всем требования, предъявленным к системе на этапе постановки задачи.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Шаблон проектирования «Модель Представление Контроллер» // Википедия: свободная энциклопедия - 2012 [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Model-View-Controller (дата обращения: 07.10.2017).
2. Архитектурный паттерн MVC // Около программирования - тьюториалы и статьи по веб-программированию. - 2011 [Электронный ресурс]. – URL: http://artanovy.com/2011/03/arhitekturnyj-pattern-mvc/ (дата обращения: 21.10.2017).
3. Ильичев С. MVC: что это такое и какое отношение имеет к пользовательскому интерфейсу // Типичный программист – 2015 [Электронный ресурс]. – URL: https://tproger.ru/articles/mvc/ (дата обращения: 03.11.2017).
4. Документация Laravel версии 5+ // Работа с фреймворком Laravel. - 2017 [Электронный ресурс]. – URL: https://laravel.ru/docs/v5 (дата обращения: 21.12.2017).
5. Моргунов Е. П. Язык SQL. Базовый курс: учеб.-практ. Пособие, М., 2017. — 257 с. (дата обращения: 14.01.2018).
6. Язык программирования PHP // The PHP Group [Электронный ресурс]. – URL: http://php.net/ (дата обращения: 25.01.2018).
7. Свободная энциклопедия Википедия, статья “ORM”. [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/ORM (дата обращения: 11.02.2018).
8. Служба крови России. [Электронный ресурс]. Режим доступа: http://yadonor.ru/about.htm. (дата обращения: 13.03.2018).

ПРИЛОЖЕНИЕ

Листинг 1 – файл маршрутизации веб приложения

```
<?php
Route::get('/news/category/{slug?}', 'NewsController@category')->name('category');
Route::get('/news/article/{slug?}', 'NewsController@article')->name('article');
Route::get('/', function () {
   return view('welcome');
});
Auth::routes();
Route::get('/onlinebooking', 'onlinebookingController@index')->name('onlinebooking');
Route::post('/onlinebooking', 'onlinebookingController@store')->name('onlinebooking.store');
Route::delete('/onlinebooking', 'onlinebookingController@destroy')->name('onlinebooking.destroy');
Route::get('/home', 'HomeController@index')->name('home');
Route::get('/anketirovanie', 'AnketaControl@homepage')->name('anketirovanie');
Route::post('/anketirovanie', 'AnketaControl@store')->name('anketirovanie.store');
Route::delete('/anketirovanie', 'AnketaControl@destroy')->name('anketirovanie.destroy');
Route::get('/support', 'TicketController@index')->name('ticket');
Route::post('/home', 'TicketController@store')->name('ticket.store');
Route::any('/send/{a}','mailController@send')->name('send');
Route::group(['prefix' => 'admin','namespace'=>'Admin', 'middleware'=>['auth']], function (){
   Route::get('/','DashboardController@dashboard')->name('admin.index');
   Route::resource('/category','CategoryController',['as'=>'admin']);
   Route::resource('/article','ArticleController',['as'=>'admin']);
   Route::resource('/color', 'ColorController', ['as' => 'admin']);
   Route::resource('/donation', 'DonationController', ['as' => 'admin']);
   Route::group(['prefix' => 'user_managment','namespace' =>'UserManagment'], function (){
      Route::resource('/user','UserController', ['as' => 'admin.user_managment']);
       Route::resource('/control','Control', ['as' => 'admin.user_managment']);
```

```
        Route::resource('/support','Support', ['as' => 'admin.user_managment']);
        Route::resource('/ankets','Ankets', ['as' => 'admin.user_managment']);
        Route::resource('/onlinebooking','Onlinebooking', ['as' => 'admin.user_managment']);
        Route::resource('/healthcare','HealthController', ['as' => 'admin.user_managment']);
        Route::put('/support/update','Support@update');
    });
});
```

Листинг 2 – файл app.blade.php

```
<!DOCTYPE html>
<html lang="{{ app()->getLocale() }}">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- CSRF Token -->
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>@yield('title')</title>
    <meta name="keyword" content="@yield('meta_keyword')">
    <meta name="keyword" content="@yield('meta_description')">
    <!-- Styles -->
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
</head>
<body style="background-color: lightskyblue">
    <div id="app">
        @include('layouts.header')

        @yield('content')
    </div>
    <!-- Scripts -->
    <script src="{{ asset('js/app.js') }}"></script>

</body>
</html>
```

Листинг 3 – вывод категорий, содержащих информацию, на страницы веб-прилоежния

```
@foreach($categories as $category)

    @if ($category->children->where('published', 1)->count())
        <li style="color: black" class="nav-item dropdown">
            <a style="color: black" href="{{url("/news/category/$category->slug")}}" class="nav-link dropdown-toggle"
               data-toggle="dropdown" role="button" aria-expanded="false">
                {{$category->title}} <span class="caret"></span>
            </a>

            <ul style="color: black" class="dropdown-menu" role="menu">
                <li style="color: black" ><a class="dropdown-item" href="{{url("/news/category/$category->slug")}}">
                    {{ $category->title }}
                </a></li>
                @foreach($category->children as $child)
                    <li style="color: black"><a class="dropdown-item" href="{{url("/news/category/$child->slug")}}">
                        {{ $child->title }}
                    </a></li>
                @endforeach
            </ul>

        @else
            <li>
                <a style="color: black" class="nav-link" href="{{url("/news/category/$category->slug")}}">{{$category->title}}</a>
                @endif
            </li>

            @endforeach
```

Листинг 4 – header.blade.php

```
<nav class="navbar navbar-default navbar-static-top" style="border-color: black;background-color: lightseagreen">
    <div class="container" style="border-color: black">
        <div class="navbar-header" style="border-color: black">

            <!-- Collapsed Hamburger -->
            <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#app-navbar-collapse" aria-expanded="false">
                <span class="sr-only">Toggle Navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>

            <!-- Branding Image -->
            <a  style="color: black" class="navbar-brand" href="{{ url('/') }}">
                Главная
            </a>
        </div>

        <div class="collapse navbar-collapse" id="app-navbar-collapse">
            <!-- Left Side Of Navbar -->
            <ul style="color: black" class="nav navbar-nav">
                @include('layouts.top_menu',['categories' => $categories])

            </ul>

            <!-- Right Side Of Navbar -->
            <ul style="color: black" class="nav navbar-nav navbar-right">
                <!-- Authentication Links -->
                @guest
                <li><a style="color: black" href="{{ route('login') }}">Вход</a></li>
                <li><a style="color: black" href="{{ route('register') }}">Регистрация</a></li>
                @else
```

```
                    <li class="dropdown" style="color: black;background-color: lightseagreen">
                        <a style="color: black;background-color: lightseagreen" href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-expanded="false" aria-haspopup="true" v-pre>
                            {{ Auth::user()->fullname }} <span class="caret"></span>
                        </a>

                        <ul class="dropdown-menu" style="color: black;background-color: lightseagreen">
                            <li><a href="{{route('home')}}">Личный кабинет</a></li>
                            @if(Auth::user()->role == '1')
                                <li><a href="{{route('admin.index')}}">Панель управления</a></li>
                            @endif
                            <li>
                                <a href="{{ route('logout') }}"
                                   onclick="event.preventDefault();
                                                 document.getElementById('logout-form').submit();">
                                    Выход
                                </a>

                                <form id="logout-form" action="{{ route('logout') }}" method="POST" style="display: none;">
                                    {{ csrf_field() }}
                                </form>
                            </li>

                        </ul>
                    </li>
                    @endguest
                </ul>
            </div>
        </div>
    </nav>
```

Листинг 5 – описание класса User

```
<?php

namespace App;

use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use Notifiable;

    protected $fillable = [
        'fullname', 'email', 'password','phonenumber',
        'liveadress', 'workadress','passport',
        'groupofblood', 'dateofbirth', 'role',
    ];



    protected $hidden = [
        'password', 'remember_token',
    ];
}
```

Листинг 6 – UserController.php

```
<?php

namespace App\Http\Controllers\Admin\UserManagment;

use App\User;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;

class UserController extends Controller
{
```

```
public function index()
{
  return view('admin.user_managment.users.index',[
     'users' => User::paginate(10)
  ]);
}


public function create()
{
   return view('admin.user_managment.users.create',[
      'user' => []
   ]);
}

public function store(Request $request)
{
   $validator = $request->validate([
      'fullname' => 'required|string|max:255',
      'email' => 'required|string|email|max:255|unique:users',
      'password' => 'required|string|min:6|confirmed',
      'liveadress' => 'required|string|max:255|',
      'passport' => 'required|string|max:255|unique:users',
      'phonenumber' => 'required|unique:users',
   ]);

   User::create([
      'fullname' => $request['fullname'],
      'email' => $request['email'],
      'password' => bcrypt($request['password']),
      'liveadress' => $request['liveadress'],
      'passport' => $request['passport'],
      'phonenumber' => $request['phonenumber'],
      'workadress' => $request['workadress'],
```

```
        'dateofbirth' => $request['dateofbirth'],
        'groupofblood' => $request['groupofblood'],
        'role' => $request['role']
    ]);
        return redirect()->route('admin.user_managment.control.index');
}



public function edit(User $user)
{
    return view('admin.user_managment.users.edit',[
      'user' => $user
    ]);
}

public function update(Request $request, User $user)
{
    $validator = $request->validate([
        'fullname' => 'required|string|max:255',
        'email' => [
            'required',
            'string',
            'email',
            'max:255',
            \Illuminate\Validation\Rule::unique('users')->ignore($user->id),],
        'password' => 'nullable|string|min:6|confirmed',
        'passport' => [
            'required',
            'string',
            'max:255',
            \Illuminate\Validation\Rule::unique('users')->ignore($user->id),],
        'phonenumber' => [
            'required',
            'string',
```

```
            'max:255',
            \Illuminate\Validation\Rule::unique('users')->ignore($user->id),]
        ]   );

        $user->fullname = $request['fullname'];
        $user->email = $request['email'];
        $request['password'] == null ?: $user->password = bcrypt($request['password']);
        $user->liveadress = $request['liveadress'];
        $user->passport = $request['passport'];
        $user->phonenumber = $request['phonenumber'];
        $user->workadress = $request['workadress'];
        $user->dateofbirth = $request['dateofbirth'];
        $user->groupofblood = $request['groupofblood'];
        $user->role = $request['role'];
        $user->save();

        return redirect()->route('admin.user_managment.control.index');

    }

    public function destroy(User $user)
    {
        $user->delete();
        return redirect()->route('admin.user_managment.user.index');
    }
}
```

Листинг 7 – файл миграции users_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration
{
```

```
void

    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('fullname');
            $table->string('dateofbirth');
            $table->string('liveadress');
            $table->string('groupofblood')->nullable()->default('newdonor');
            $table->string('workadress')->nullable();
            $table->string('passport')->unique();
            $table->string('phonenumber')->unique();
            $table->string('email')->unique();
            $table->string('password');
            $table->string('role')->default('0');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```