ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ГАГАРИНА Ю.А.»

Институт прикладных информационных технологий и коммуникаций

Кафедра «Прикладные информационные технологии»

Направление 09.03.03 «Прикладная информатика»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«Разработка программного обеспечения отслеживания приема прописанных лекарств пациентом. Подсистема врача.»

Студент (ка) _____Беляев Матвей Александрович_____
фамилия, имя, отчество

курс 4 группа б2-ПИНФ41

Руководитель
доцент каф. ПИТ, к.т.н. 11.06.20 А.В. Ермаков
должность, ученая степень, уч. звание | подпись, дата | Инициалы Фамилия

Допущен к защите
Протокол № 12 от «11» июня 2020 года

Зав. кафедрой «Прикладные информационные технологии»

кандидат технических наук, доцент 11.06.20 О.А. Торопова
ученая степень, уч. звание | подпись, дата | Инициалы

Саратов 2020г

Содержание

Введение

В настоящее время информационные технологии используются во всех сферах жизни общества, в том числе они используются в медицине [1]. Некоторые болезни требуют четкого следования графику приема медикаментов, определяемого врачом. При стационарном лечении пациента это не является проблемой, с этим справляется младший медицинский персонал, но некоторые болезни не требуют нахождения пациента в больнице, при этом соблюдение режима приема лекарств остается необходимым условием успешного лечения [2]. При нахождении не под присмотром персонала больницы пациент может начать пропуск приема необходимых препаратов, что может привести к снижению эффективности лечения, а так же тяжелым последствиям, вплоть до летального исхода, так, по статистике, каждый год более 100 000 человек умирает от сердечно-сосудистых заболеваний из-за несоблюдения режима приема лекарств. В большинстве случаев пациенты пропуск приема медикаментов объясняют забывчивостью.

Эту проблему можно решить с помощью информационных технологий. Если предоставить врачам комплекс программных и аппаратных средств, которые позволят отслеживать прием медикаментов каждым из пациентов, а так же оповестят пациента о необходимости принять тот или иной препарат, получится осуществить достаточный уровень контроля хода лечения. С помощью

этого врачи смогут делать выводы об эффективности курса лечения без необходимости личной встречи с пациентом и корректировать курс лечения, если это требуется.

Целью ВКР является разработка информационной системы которая позволит врачу вести историю болезни пациента автоматически отображая в ней статистику приема лекарств амбулаторным больным на основе данных полученных с сервера устройства отслеживания приема лекарств

Для достижения этой цели необходимо решить следующие задачи:

1. Провести анализ предметной области, выявить проблематику и сформировать требования к подобной информационной системе.
2. Провести обзор существующих информационных систем для работы врача, и определить пригодность существующих систем для решения поставленной задачи.
3. Провести планирование информационной системы, определить перечень входных и выходных данных, информационные потоки
4. Определить требуемую архитектуру информационной системы
5. Провести разработку программного обеспечения.

1 Обзор предметной области и существующих аналогов

1.1 Формирование требований к системе

Целью создания системы является автоматизация контроля со стороны лечащего врача приема медикаментов пациентом. Необходимо предоставить врачу возможность работы с пациентом (добавление новых пациентов, назначение им курсов лечения, просмотр истории приема пациентами назначенных медикаментов, составление отчетов о ходе лечения). Информация о ходе приема будет собираться с помощью «Умной банки» и будет включать в себя: информацию о принятом медикаменте, дате и времени открытия. Отчет о ходе лечения будет формироваться на основе данных о приеме медикаментов пациентом на текущий момент времени, и будет позволять оценить то, на сколько пациент следует рекомендациям врача.

Таким образом, можно сформировать основные требования к системе:

- Возможность регистрации пациента врачом;
- Возможность создания курса лечения врачом;
- Добавление приема медикамента в курс лечения;
- Установка графика приема медикамента;
- Возможность просмотра хода приема медикаментов в курсе лечения;
- Хранение информации о всех созданных в системе курсах лечения;

- Составление отчета о ходе лечения;
- Предоставление врачу возможности управлять назначенными курсами лечения;
- Оповещение пациента о необходимости принять препарат;
- Обеспечение контроля приема медикаментов пациентом, находящимся не на стационарном лечении.

1.2 Сравнительный анализ существующих систем

На данный момент времени существует достаточно большое количество информационных систем, упрощающих деятельность врачей. Далее будут описаны некоторые из уже существующих информационных систем для медицинских учреждений и систем, напоминающих пациентам о необходимости приема медикаментов.

Медицинская информационная система VS Clinic

VS Clinic – интегрированное решение для медицинских стационаров, разработанное компанией ВитаСофт [4].

Решение позволяет эффективно управлять основными процессами деятельности медицинского учреждения по оказанию медицинской помощи, начиная с регистрации первичного обращения пациента и до момента его выписки.

Решение построено на базе Microsoft Dynamics AX, что позволяет воспользоваться всеми преимуществами централизованной ERP системы управления предприятием.

Данная система позволяет вести учет курсов лечения пациентов, проводить оценку эффективности лечения, контролировать расход медицинских препаратов и оборудования, позволяет составлять отчет о ходе лечения пациентов. К сожалению, данная система предназначена для стационарного лечения и не позволяет вести контроль за лечением пациента удаленно.

Медицинская информационная система qMS

МИС qMS – это инструмент управления качеством оказания медицинской помощи и ресурсами медицинской организации (комплекса медицинских организаций, вплоть до региональной и национальной систем здравоохранения) [5]. Медицинская информационная система qMS осуществляет такие функции, как интеграция с медицинским оборудованием, хранение полной информации о пациенте в электронной медицинской карте, фиксация всех действий врачей, управление потоком пациентов и ресурсами учреждения, ведение финансовой отчетности, аналитическая обработка данных и выявление причинно-следственных связей для доказательной медицины.

Система реализует большое количество возможностей, необходимых при работе с пациентами, в частности, с помощью этой информационной системы можно хранить данные о пациентах, которые когда-либо проходили лечение в медицинском учреждении, информацию о лечащем враче для каждого курса лечения пациента, всех проведенных в рамках курса процедурах.

Данная система представляет собой средство для управления медицинской организацией и не предоставляет возможности для контроля за ходом приема медикаментов со стороны лечащего врача.

Автоматизированная система MedLink

Автоматизированная система MedLink позволяет осуществлять единый контроль и управление медицинским учреждением [6]. Автоматизированные медицинские информационные средства позволяют организовать качественный учет пациентов, хранение данных о сотрудниках, медицинских исследованиях и многое другое.

Информационная система позволяет централизовать управление и контроль над медицинскими учреждениями. Она дает возможность руководителям и врачам в режиме реального времени оценивать состояние дел на каждом локальном участке и во всей сети, анализировать процессы за любой интервал времени с целью повышения качества обслуживания пациентов, оптимизации складского учета и повышения информативности финансово-управленческой информации.

Информационная система объединяет в себе:

- все медицинские данные о пациентах (истории болезни, бюллетени, процедурные манипуляции, инструментально-лабораторная диагностика, мониторинг состояния пациента) в электронном виде,

- сообщения между структурами медицинского учреждения, включая в себя финансовую и административную стороны.

Возможности:

- Регистрация пациента
- Возможность создания курса лечения врачом;
- Добавление приема медикамента в курс лечения;
- Установка графика приема медикамента;
- Хранение информации о всех созданных в системе курсах лечения;

Как и предыдущие системы, эта система не соответствует всем предъявленным требованиям. В частности, в этой системе врачу не предоставляется информация о ходе приема медикаментов, а так же не предусмотрено никаких средств по оповещению пациента о необходимости приема лекарств.

MedReady 1700

Система распределения таблеток MedReady 1700 представляет собой дозатор, который содержит 28 ячеек для недельного набора лекарств и веб-приложение для его настройки [7].

Когда подходит время приема лекарства, раздастся звуковой и световой сигнал. Нужные таблетки оказываются под небольшой крышкой в верхней части дозатора, и чтобы их взять достаточно сдвинуть крышку. Если, несмотря на сигналы, лекарство не используется, устройство через беспроводную сеть посылает соответствующее сообщение

врачу или лицу, ухаживающему за пациентом. Для этого MedReady снабжена модемом мобильной связи, позволяющим связываться с системой для получения информации о правилах приема таблеток пациентом.

Веб-приложение не предоставляет информации о том, какие конкретно медикаменты были приняты, что делает невозможным построение точного отчета о приеме более чем 1 медикамента. Так же, система не имеет интерфейса для работы врача, врач не может просматривать курсы, которые назначил, ход приема медикаментов для каждого пациента и т.д., система позволяет только настроить отправку уведомлений для лечащего врача с помощью e-mail или номера телефона. Веб-приложение позволяет настроить дозатор на выдачу до 4 разных препаратов ежедневно, для приема большего количества препаратов необходимо использовать дополнительные дозаторы. Просмотр истории использования возможен только для каждого дозатора отдельно, что исключает возможность сразу получить информацию о ходе приема 5 и более медикаментов одним пациентом.

GlowCap

GlowCap - это специальная крышка, которая подходит к большинству баночек с лекарствами в США. Если пришло время приема лекарства, то крышка начинает мигать оранжевым светом. Затем через час, если за это время лекарство не было принято, GlowCap начинает проигрывать мелодию. Она поставляется вместе ночником, который находится на беспроводной связи с крышкой и также

начинает мигать в нужное время. Находящийся внутри крышки чип определяет, когда открывается баночка и отправляет соответствующее сообщение по беспроводной сети в приложение, которое позволяет врачу контролировать насколько точно выполняется режим приема лекарств.

Если человек все равно забудет принять лекарство после напоминания, то он через пару часов получит дополнительное сообщение на свой телефон.

В отличии от предыдущего решения, данная система не имеет веб-приложения для просмотра хода приема медикаментов. Врач так же оповещается только с помощью сообщения, присылаемого на электронную почту или телефон и не может получить достаточно развернутую информацию о ходе приема лекарств для составления отчета и корректировки курса.

1.3 Обзор работы с данными в медицинских информационных системах

Важной частью работы врача является оформление истории болезни. История болезни является ключевым документом при осуществлении врачебной деятельности [8]. В истории болезни указывается вся необходимая информация для выбора средств и методов лечения, кроме того история болезни позволяет оценить качество оказания медицинской помощи врачом, то, насколько верно и подробно врач извлекает информацию о заболевании исходя из жалоб пациента. История болезни должна

содержать информацию о пациенте: фамилию, имя, отчество, пол, дату рождения и возраст на момент оформления истории болезни. Кроме этого в истории болезни указываются жалобы, которые озвучивает пациент, составляется анамнез жизни пациента- краткая биография, описываются заболевания, которые пациент перенес за время своей жизни, указывается наличие вредных привычек, ставиться диагноз, определяемый исходя из полученной информации и пишется краткое пояснение диагноза, в котором врач обосновывает поставленный диагноз. После этого составляется план лечения пациента, включающий в себя список необходимых медикаментов, план их приема.

Таким образом можно с уверенностью сказать, что приложение осуществляющее поддержку работы врача должно содержать функционал по составлению истории болезни для пациента. При проектировании инструментов для создания истории был проведен анализ схожих по назначению средств, присутствующих в других информационных системах. Примеры интерфейса подобных систем представлены на рисунках 1, 2, 3.

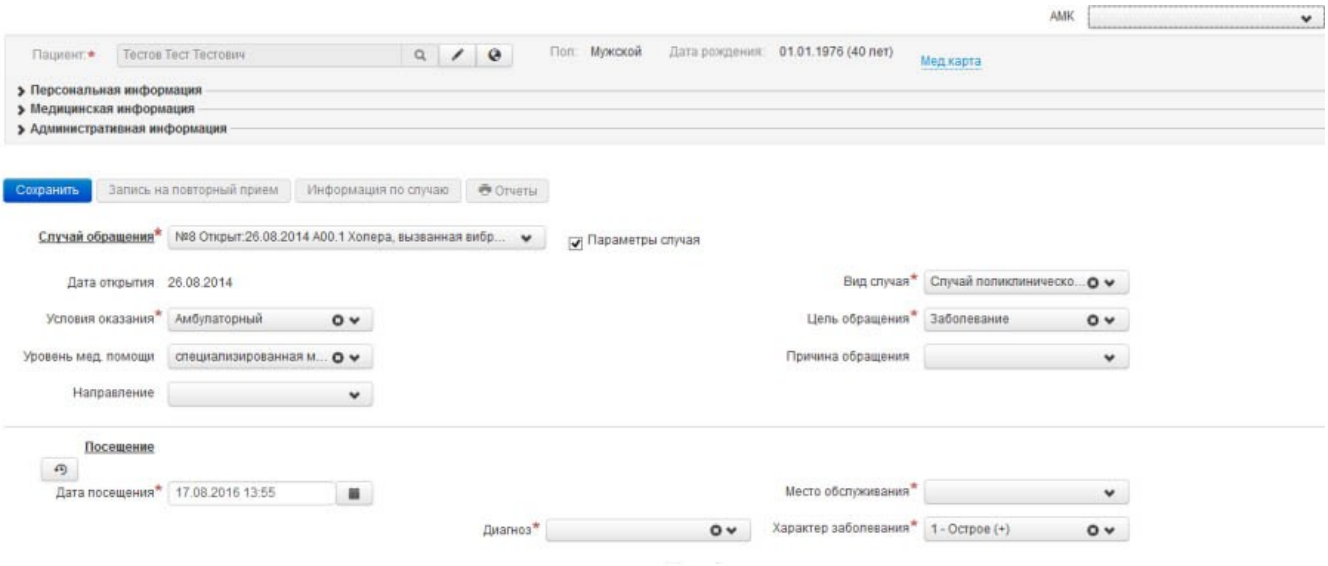


Рисунок 1 — Создание истории болезни в системе «Confluence»

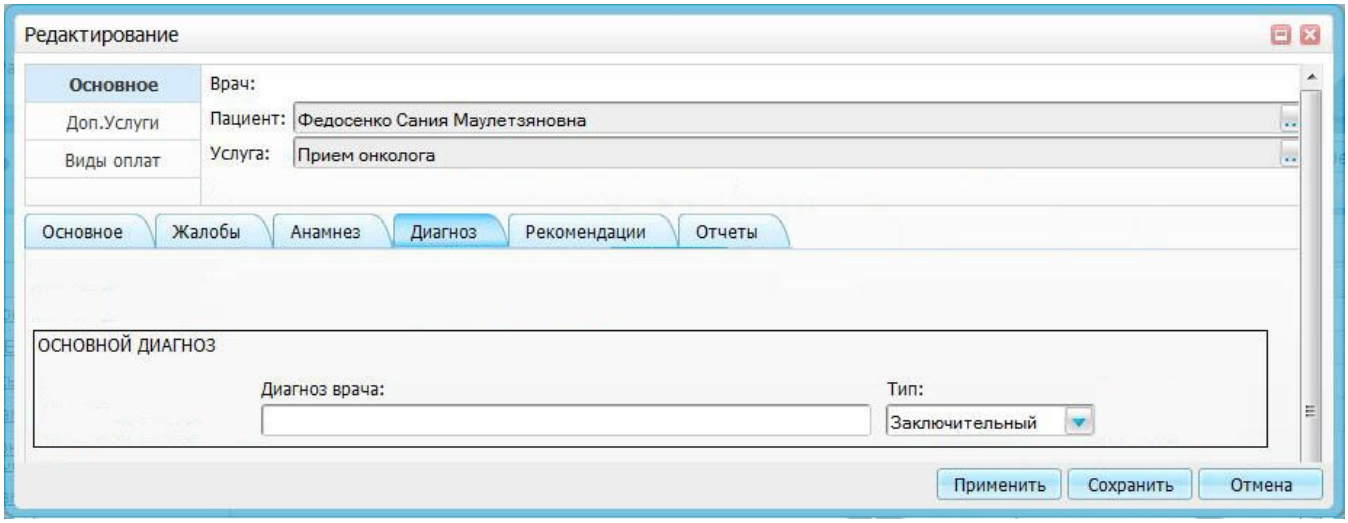


Рисунок 2 — Создание истории болезни в системе «Барс»

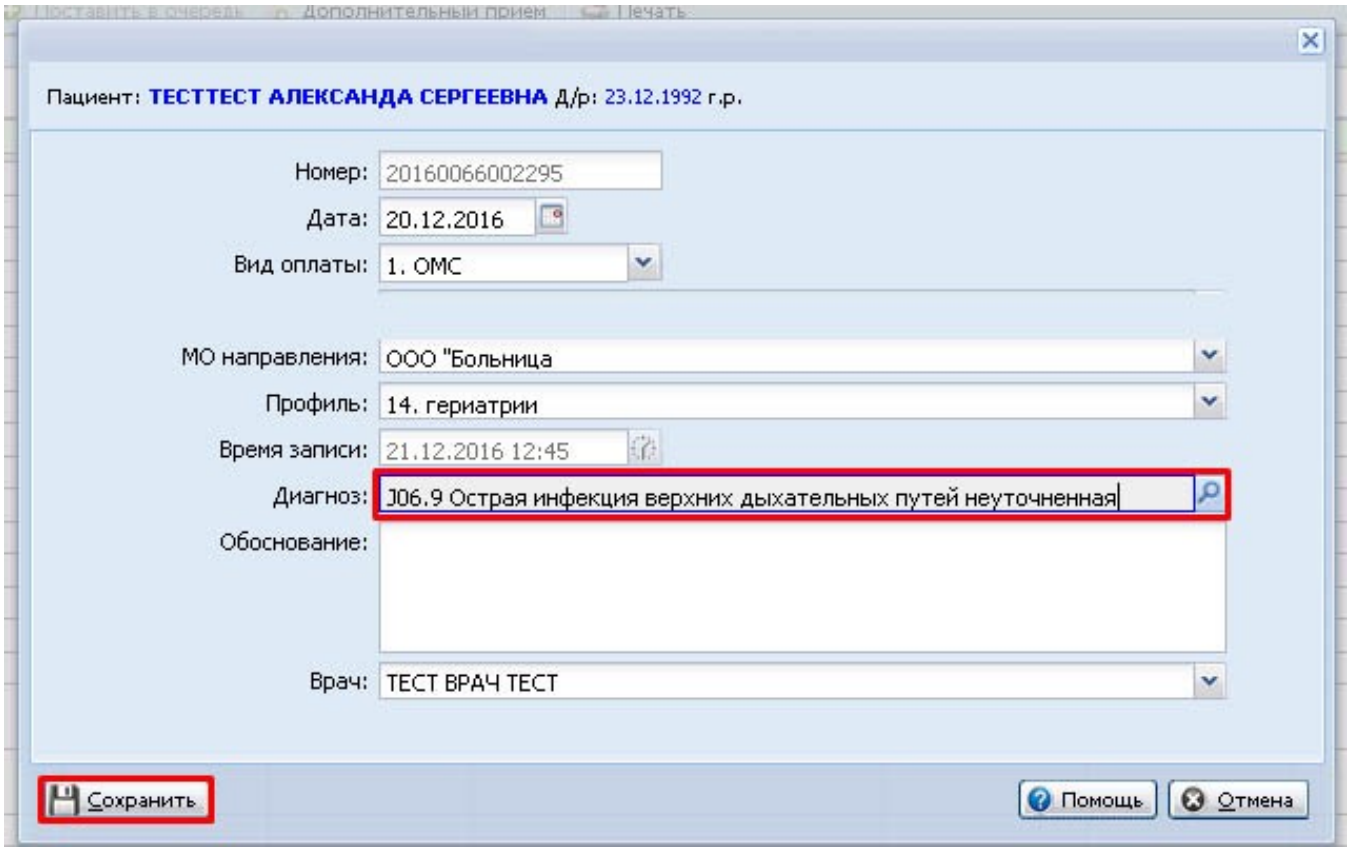


Рисунок 3 — Создание истории болезни в системе «Redmine»

1.4 Результаты сравнительного анализа

После анализа собранной информации можно сделать вывод, что система VS Clinic не соответствует требованиям по причине отсутствия возможности отследить ход приема медикаментов пациентом, не находящимся в медицинском учреждении, высокой стоимости внедрения и обслуживания системы, сложности в обучении пользователей системы.

Система qMS так же не реализует возможность отследить ход лечения пациента дистанционно, имеет достаточно высокую стоимость внедрения, не имеет

возможности уведомлять пациента о необходимости приема медикамента.

Система MedLink так же не удовлетворяет поставленным требованиям из-за невозможности контроля лечащим врачом хода приема медикаментов пациентом.

Система MedReady и GlowCap не удовлетворяют поставленным требованиям, так как не предоставляют врачу возможность удаленно просматривать историю приема лекарств, список назначенных курсов и управлять ими.

Исходя из вышеизложенного, можно сделать вывод, что ни одна из существующих систем не реализует все предъявленные требования и необходимо создать новую систему представляющую собой комплекс программных и аппаратных средств, позволяющую врачу осуществлять все функции, обозначенные в требованиях (регистрация пациента, назначение курса лечения и т.д.) и напоминающую пациентам о необходимости приема прописанных лекарств в необходимое время.

2 Проектирование информационной системы

2.1 Общее описание архитектуры информационной системы

Для описания принципа работы системы будет использоваться методология IDEF0. Данная методология предназначена для описания бизнес-процессов, происходящих в системе, структуры и функций, из которых состоит система. Диаграмма уровня А-0 представлена на рисунке 4.

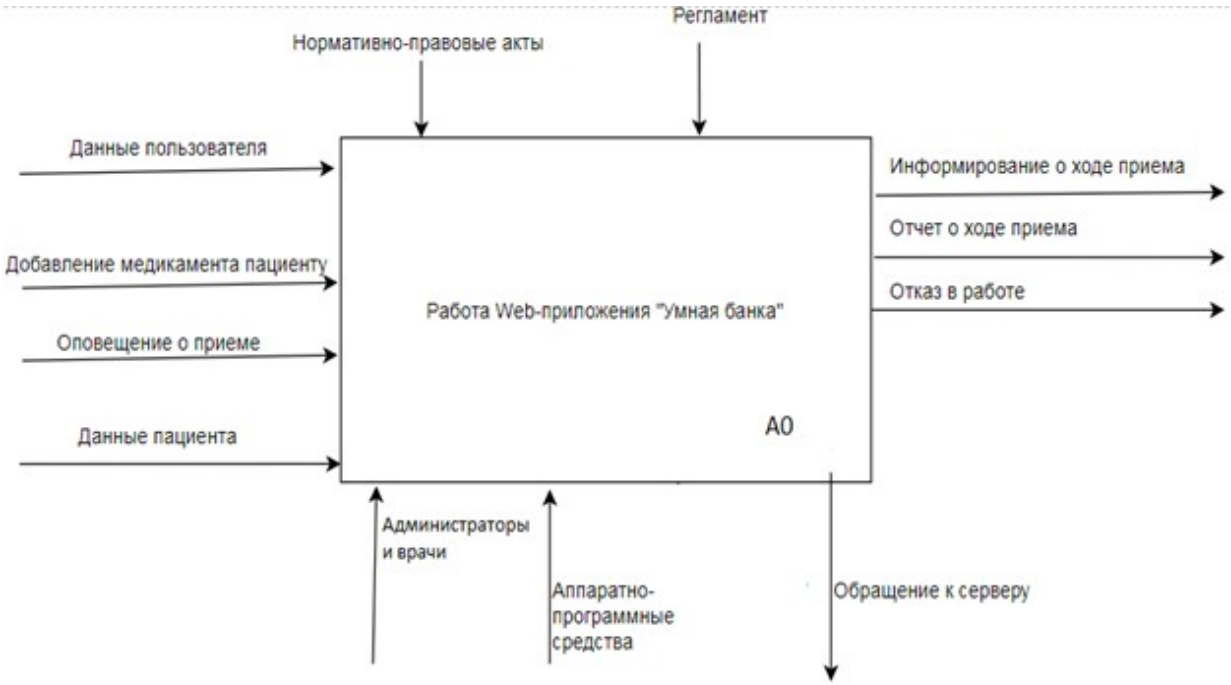


Рисунок 4 — Диаграмма уровня А-0 (методология IDEF0)

В общем виде система будет представлять собой веб-приложение, связанное с системой «Умная банка» и базой данных. База данных будет содержать информацию о пользователях системы, пациентах, курсах лечения, банках,

использующихся в системе, медикаментах, назначенных для всех пациентов. Так же в базе данных будет содержаться информация о каждом открытии банки с указанием времени.

Веб-приложение является достаточно удобным решением, так как позволяет избежать проблем с установкой и обновлением программного обеспечения на компьютерах конечных пользователей. Так же это позволит врачам осуществлять работу в системе не только на месте работы, но и, при необходимости, из дома. Это даст возможность врачу наблюдать за приемом медикаментов теми пациентами, состояние которых сильно зависит от своевременности приема лекарств, так как врач сможет контролировать их лечение не только в рабочее время. Помимо прочего, использование веб-приложения возможно на самых различных устройствах, в том числе на смартфонах и планшетах, что позволит сделать систему достаточно гибкой в применении.

Описание общей архитектуры приложения можно осуществлять с помощью диаграмм UML, в частности диаграммы развертывания. Данный вид диаграмм позволяет описать основные компоненты системы, способы взаимодействия, которые существуют между ними. Диаграммы развертывания применяются для более наглядной демонстрации архитектуры системы. При проектировании приложения была создана диаграмма развертывания, представленная на рисунке 5.

Как видно на диаграмме, пользователи системы используя браузер будут обращаться к серверу приложения. Сервер приложения будет обрабатывать запросы пользователей, предоставляя в ответ информацию сформированную по шаблону, определяемую файловым сервером. Информацию для предоставления пользователю сервер приложения будет брать обращаясь к серверу базы данных. Система «Умная банка» будет также обращаться к серверу приложения с запросом на добавление новой записи о действиях с банкой, запрос будет обрабатываться, после чего сервер веб-приложения отправит запрос на добавление записи серверу базы данных.

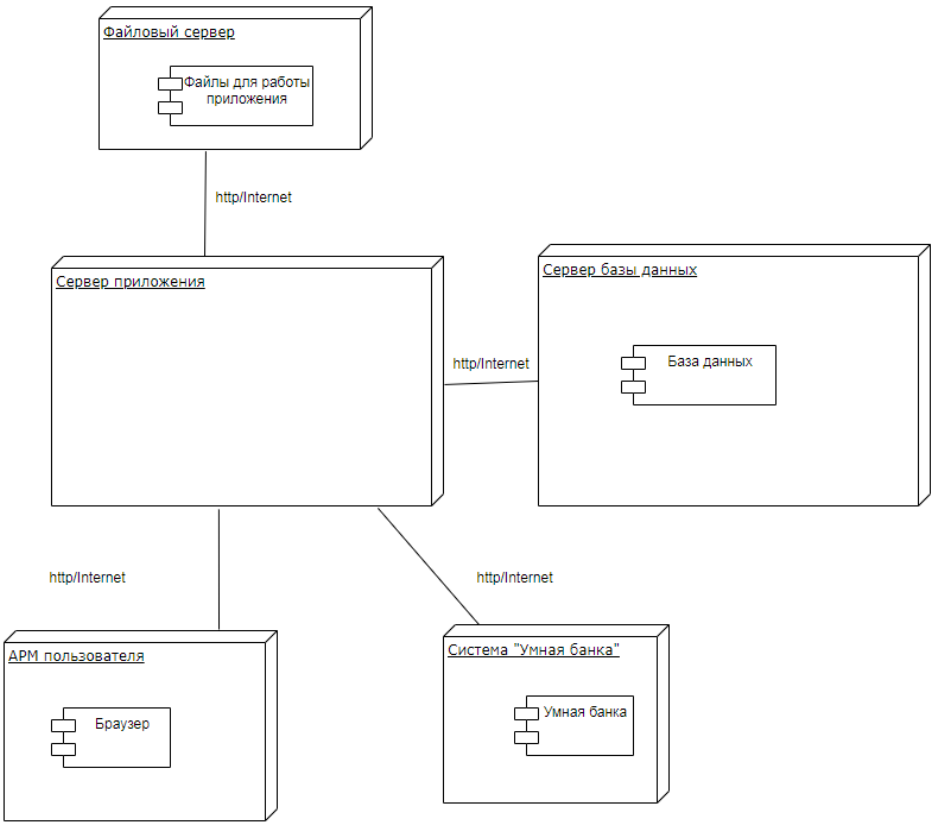


Рисунок 5 — Диаграмма развертывания

Для уточнения деталей работы системы была создана диаграмма DFD. DFD - диаграммы потоков данных, которые являются средством моделирования функциональных требований проектируемой системы. [9] С их помощью эти требования разбиваются на компоненты и представляются в виде сети, связанной потоками данных [10]. Главная цель таких средств – продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

Нотации DFD – удобное средство для формирования контекстной диаграммы [11], то есть диаграммы, показывающую разрабатываемую информационную систему в коммуникации с внешней средой. Назначение диаграммы верхнего уровня – ограничить рамки создаваемой системы [12], определить, где заканчивается разрабатываемая система и начинается среда. Одним из преимуществ нотации является возможность отображения на диаграмме места бизнес-процесса, в которых хранится информация, либо материальные ресурсы.

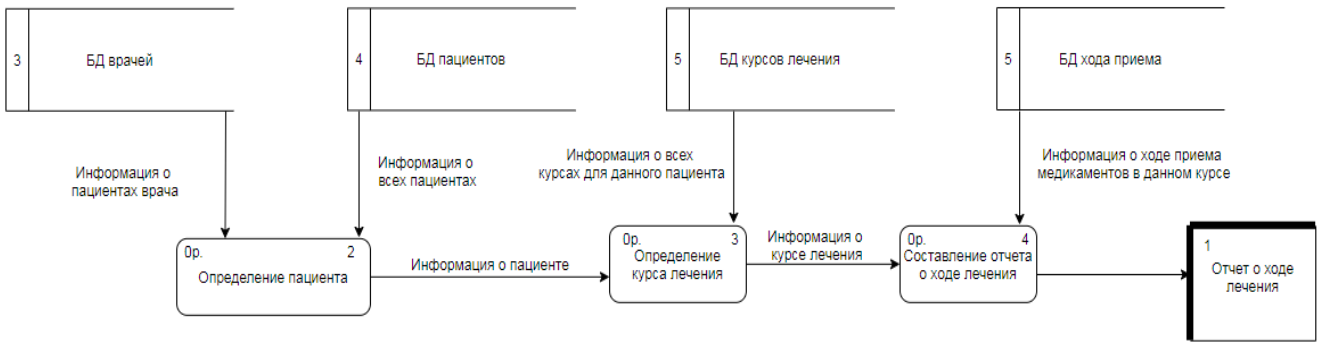


Рисунок 6 — Диаграмма DFD «Составление отчета»

Рассмотрим основные хранилища данных создаваемого приложения. В состав приложения будут входить БД врачей с данными всех врачей, зарегистрированных в приложении, в БД пациентов будут находится данные всех пациентов, в БД курсов лечения будет находится информация о всех курсах лечения, существующих на данный момент, в БД хода приема будут данные о ходе приема медикамента, в том числе дата приема, название медикамента, номер курс лечения. На выходе будет составляться отчет о ходе лечения пациента.

Для уточнения описания предметной области строятся модели на основе методологии объектного анализа (UML) Unified Modeling Language -унифицированного языка моделирования [13].

В результате моделирования формируется более точное описание предметной области. Использование языка UML для моделирования позволяет сделать модель понятной для всех участников проекта [14].

Прежде чем начать разработку системы, необходимо на уровне моделей детально описать предметную область, для которой эта система разрабатывается. Язык UML включает в себя диаграмму бизнес-процесса, диаграмму прецедентов и иные [15]. Он позволяет построить модели, дополняющие описание предметной области.

Для более наглядной демонстрации функционала персонала системы была составлена диаграмма вариантов использования, которая отражает возможности

проектируемой системы. На ней представлены сущности, взаимодействующие с системой с помощью вариантов использования, так называемые «actor». При этом «actor» - это любая сущность, взаимодействующая с системой из внешней среды. Варианты использования в этой диаграмме определяют некоторый набор действий, совершаемый системой при диалоге с «эктором» [16]. Между «экторами» может существовать наследование. Для каждого «actor» существует набор действий, которые он может совершать. Диаграмма видов деятельности представлена на рисунке 7.

Пациент может авторизоваться в системе, посмотреть историю болезни, которая была составлена на основе его жалоб, составить отчет о курсе лечения, изменить персональные данные в личном кабинете.

Врач так же может авторизоваться, менять данные в личном кабинете, посмотреть список всех пациентов, проходящих у него лечение, детальную информацию о лечении каждого из пациентов, составить отчет о лечении пациента, при необходимости, сохранить его в формате docx.

Главный врач может авторизоваться, изменить свои персональные данные в личном кабинете, посмотреть информацию о деятельности врача, которая включает в себя список всех пациентов, проходящих или прошедших лечение у данного специалиста, информацию о каждом заболевании, лечение которого определил врач, составить отчет на основе этой информации. Кроме этого, ему так же

доступна возможность просмотра детальной информации о любом курсе лечения, с составлением отчета.

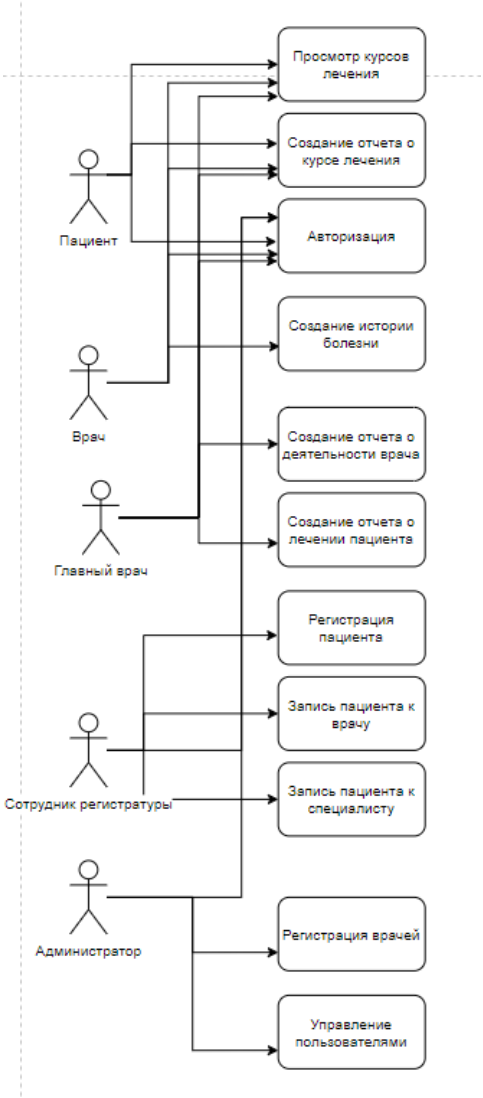


Рисунок 7 — Диаграмма видов деятельности

Сотрудник регистратуры может авторизоваться, осуществить регистрацию пациента, изменить свои данные в личном кабинете, осуществить запись пациента к какому-либо конкретному врачу или любому из специалистов.

Администратор может авторизоваться, изменить данные в личном кабинете, изменить данные какого-либо

из пользователей, сбросить пароль для любого из пользователей, при возникновении необходимости.

Для более детального описания предметной области была составлена диаграмма последовательностей, представляющая собой описание жизненного цикла основных объектов системы [17]. Данная диаграмма представлена на рисунке 8.

2.2 Входные и выходные параметры

Как видно из диаграммы, в качестве входных параметров система будет принимать данные пользователя, т.е. врача, которые будут использоваться при регистрации и авторизации врачей.

Данные врачей будут включать в себя:

- Фамилию, имя и отчество врача;
- Контактный номер телефона;
- Паспортные данные врача;
- Адрес места жительства;
- Специализацию врача.

Кроме этого, при регистрации новых пациентов, на вход системе будут даваться данные пациентов, состоящие из:

- Фамилии, имени и отчества пациента;
- Номера полиса обязательного медицинского страхования;
- Информация для авторизации пациента;

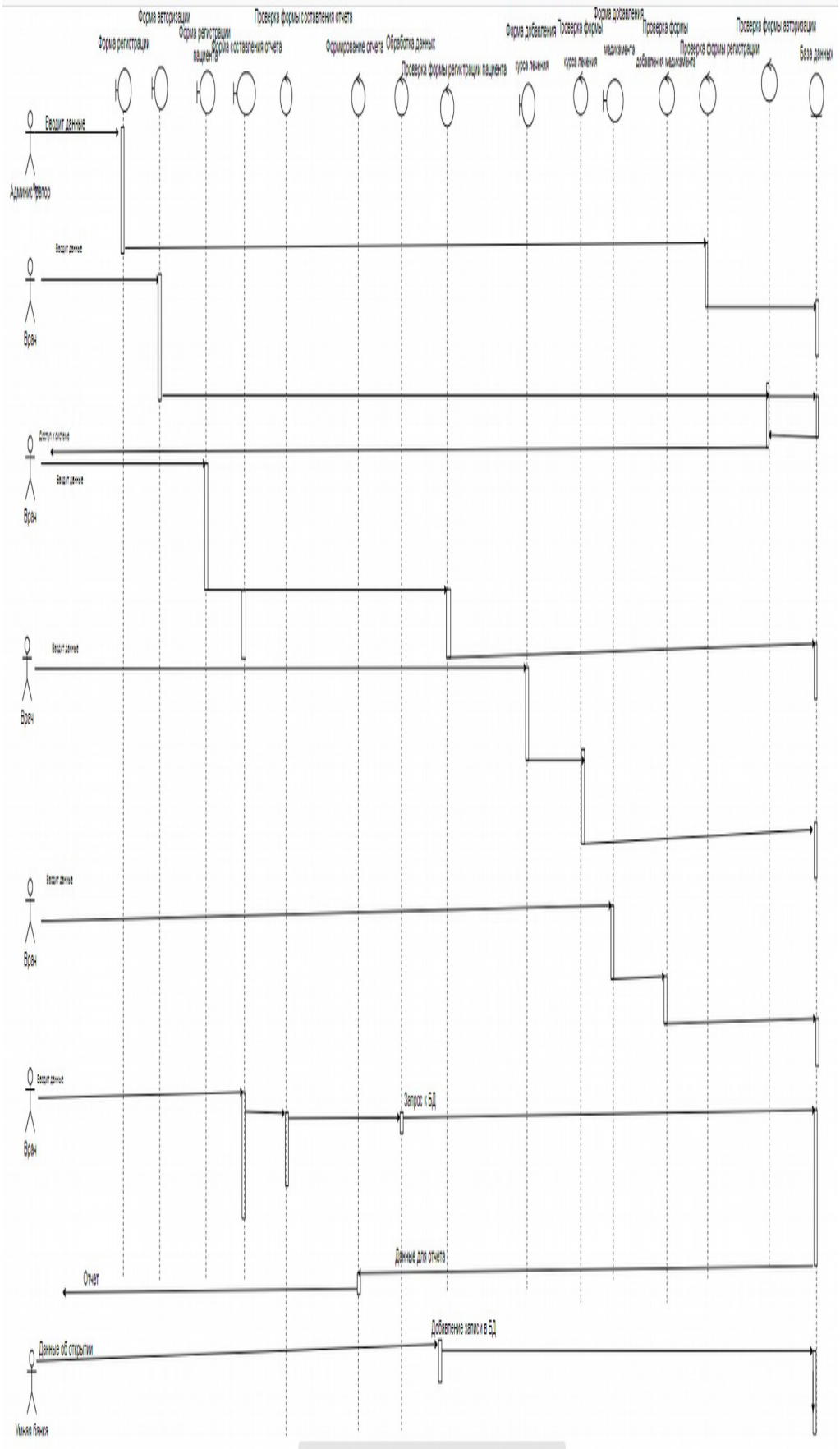

Форма авторизации
Проверка формы составления отчета
Форма регистрации
Форма регистрации пациента
Форма составления отчета
Формирование отчета
Обработка данных
Проверка формы регистрации пациента
Форма добавления курса лечения
Проверка формы курса лечения
Форма добавления назначения
Проверка формы добавления назначения
Проверка формы регистрации
Проверка формы авторизации
База данных
Вводит данные
Администратор
Врач
Данные о назначении
Запрос к БД
Данные для отчета
Отчет
Добавление записи в БД
Данные об открытии
Ушаков Ярослав

Рисунок 8 — Диаграмма последовательностей

Помимо этого, система будет принимать информацию, получаемую от «умных банок», о времени использования того или иного препарата. Так же на вход будет подаваться информация для приема медикамента, добавляемого в курс лечения, в частности его название, график приема и, при необходимости, дозировка.

В результате своей работы система будет информировать врача о приеме медикаментов пациентами, оповещать пациента о необходимости принять тот или иной препарат, создавать отчеты о ходе лечения пациентов.

Данные для формирования отчета будут браться из таблиц базы данных. В системе будет возможность создания различных отчетов, тип создаваемого отчета будет зависеть от роли пользователя.

Отчеты, создаваемые пациентом

Пациент сможет увидеть отчет о ходе лечения в рамках любого из курсов, которые когда-либо были назначены ему в системе. Такой отчет будет содержать в себе:

- дату начала лечения;
- планируемую дату окончания лечения, если курс не окончен;
- дату окончания лечения для завершенных курсов;
- данные о лечащем враче;
- диагноз, который определил врач;

- информацию о том, какие медикаменты принимает пациент;
- план приема каждого медикамента;
- сводку о качестве приема медикаментов (сколько принял, сколько пропустил и т. д.).

Отчеты, создаваемые врачом

Врачу будет доступно создание отчетов для любого из курсов, в котором данный специалист назначен лечащим врачом.

Отчет будет содержать:

- дату начала лечения;
- планируемую дату окончания лечения, если курс не окончен;
- дату окончания лечения для завершенных курсов;
- данные о лечащем враче;
- данные о пациенте, для которого назначен курс;
- информацию о том, какие медикаменты принимает пациент;
- время приема каждого из медикаментов.

Отчет будет предоставлять информацию для лечащего врача о том, как пациент следует рекомендациям и позволит сделать вывод о необходимости продолжения данного курса лечения либо необходимости изменения набора медикаментов, если пациент исправно принимает лекарства, но его состояние не улучшается.

Так же врачу будет предоставлена возможность составить отчет о приеме медикаментов пациентом за все

курсы, которые были ему назначены. С помощью этого отчета врач сможет предположить то, как пациент будет принимать прописанные лекарства. Такой отчет будет содержать:

- данные о пациенте, для которого назначен курс;
- список курсов, которые были назначены пациенту за все время;
- дату начала курса;
- планируемую дату окончания лечения, если курс не окончен;
- дату окончания лечения для завершенных курсов;
- сводку о приеме медикаментов в рамках каждого из курсов (сколько принял, сколько пропустил и т. д.).

Отчеты, создаваемые главным врачом

Главный врач сможет создавать отчеты, аналогичные тем, которые доступны врачу. Помимо этого, главный врач будет иметь возможность создавать отчеты, содержащие информацию о курсах лечения, назначенных определенным врачом, такие отчеты будут содержать:

- данные о лечащем враче;
- список курсов, которые были назначены врачом за все время;
- дату начала курса;
- планируемую дату окончания лечения, если курс не окончен;
- дату окончания лечения для завершенных курсов;

На основе этих отчетов главный врач сможет сделать вывод о качестве курсов лечения, которые назначает врач, и получить статистику о распространении определенных заболеваний в определенные промежутки времени (сезонность, количество заболевших и т.д.).

2.3 Информационные потоки

Для входа в систему пользователь должен будет ввести логин и пароль, которые будут храниться в базе данных. В целях безопасности, в базе данных пароль будет храниться в хэшированном виде.

После авторизации врач сможет увидеть список всех неоконченных курсов лечения, которые назначил. В отдельной вкладке врач сможет увидеть список уже завершенных курсов. Для любого курса, действующего или завершенного, можно будет посмотреть детальную информацию, которая будет содержать дату начала курса, дату окончания курса, данные пациента, которому был назначен курс, список медикаментов и план их приема, определенные в рамках курса. Для получения детальной информации врачу будет необходимо нажать на кнопку «Детали курса», находящуюся в таблице со списком курсов. При просмотре детальной информации врач сможет увидеть ход приема любого из назначенных медикаментов, нажав на название этого медикамента, при этом в новой вкладке будет выведен список всех приемов препарата с датой и временем приема.

Для создания нового пациента необходимо будет заполнить специальную форму, в которой будут поля для ввода всей необходимой информации. После обработки формы, данные из нее будут занесены в соответствующие столбцы таблицы базы данных, содержащей информацию о пациентах. Сразу после регистрации пациента врачу будет предложено добавить новый курс лечения для нового пациента.

Создание курса лечения будет осуществляться с помощью заполнения формы, которая будет содержать дату окончания курса, диагноз, который врач поставит пациенту, комментарий к курсу лечения, если это необходимо. Данные, после обработки формы, попадут в таблицу с курсами лечения базы данных, если все поля формы будут заполнены правильно, в ином случае врачу будет необходимо исправить обнаруженные недостатки. Создание курса лечения так же возможно и для уже зарегистрированного пациента, для чего врач должен будет найти необходимого пациента из списка и нажать на кнопку «Добавить курс».

После создания курса лечения врач будет перенаправлен на страницу, содержащую детальную информацию о курсе лечения, где сможет назначить прием одного или нескольких медикаментов. Добавление медикамента будет осуществляться с помощью формы, в которой врач выберет банку, в которой будет храниться данный препарат, название препарата, дозировку, если это необходимо, и частоту приема данного лекарства. После

обработки формы данные из нее будут занесены в таблицу, содержащую информацию о выдаче банок.

Для формирования отчета врач или администратор должен будет выбрать курс, для которого необходимо сформировать отчет, перейти во вкладку “детали курса” и нажать на кнопку «Сформировать отчет».

После авторизации администратор сможет увидеть список всех зарегистрированных в системе врачей. На отдельной вкладке он сможет посмотреть всех зарегистрированных пациентов. Администратор так же сможет регистрировать пациентов, кроме этого ему будет предоставлена возможность регистрации новых врачей. Регистрация врача будет напоминать регистрацию пациента, но форма регистрации врача будет содержать больше полей для ввода, что обусловлено необходимостью указания специализации врача, его адреса проживания и т.д.

Администратор сможет посмотреть информацию о всех курсах лечения, назначенных каким-либо специалистом, для чего ему будет необходимо нажать на кнопку «Курсы врача», находящуюся в таблице со списком всех врачей.

Помимо прочего, администратор сможет назначить курс лечения, в котором он выберет врача, который будет осуществлять лечение пациента. Данный процесс будет напоминать то, как врач добавляет курс лечения для пациента, но с дополнительным полем, в котором администратор выберет лечащего врача.

Взаимодействие с системой «Умная банка» будет происходить по следующему сценарию: система будет передавать указание на то, какая банка должна начать оповещение пациента о времени приема препарата, после открытия требуемой банки пациентом система получит уведомление о совершенном действии, получит текущее время и добавит новую запись в таблицу базы данных, содержащую историю использования всех банок.

3 Описание реализации программного обеспечения

3.1 Обоснование выбора программного обеспечения

При разработке программного обеспечения важную роль играет выбор средств разработки. На данный момент существует большое количество платформ для создания приложений, которые имеют свои преимущества и недостатки, предназначены для работы с теми или иными СУБД, используют определенные языки программирования в качестве основных.

Для разработки веб-приложения были выбраны продукты компании Microsoft, а именно среда разработки MS Visual Studio 2019, СУБД MS SQL Server.

Для выполнения всех необходимых функций разрабатываемому приложению требуется работа с базой данных. На данный момент существует большое количество систем управления базами данных, которые позволяют выполнять необходимые операции хранения, добавления, удаления, выборки данных по определенным критериям. Реляционные базы данных являются одним из наиболее популярных подходов к разработке БД. Данная концепция подразумевает хранение всей необходимой для работы приложения информации в связанных между собой таблицах. Преимуществом данной модели является возможность хранить информацию для информационных систем любой сложности, также при использовании

данного подхода облегчается поиск и выбор данных по определенным критериям.

В качестве СУБД для разрабатываемой системы был выбран продукт компании Microsoft MS SQL Server. Данная СУБД предоставляет весь необходимый перечень инструментов для работы с базами данных, их создания, управления, администрирования [18]. Работа с этой СУБД возможна как с использованием графического интерфейса, так и с помощью языка T-SQL. Язык T-SQL является одной из разновидностей стандарта SQL- структурированного языка запросов. Данная разновидность языка SQL позволяет выполнять запросы любой сложности, вложенные запросы и т.д.

Среда разработки MS Visual Studio позволяет создавать приложения для Платформы ASP.NET Core. Данная платформа представляет технологию от компании Microsoft, предназначенную для создания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов.

ASP.NET Core работает на основе кросс-платформенной среды .NET Core [19], которая может быть развернута на наиболее популярных операционных системах: Windows, Mac OS, Linux. Несмотря на то, что большинство приложений ASP.NET по прежнему создаются для работы с ОС Windows, теперь разработчики приложений не ограничены только этой операционной системой. То есть, при необходимости, возможен запуск

веб-приложения не только на ОС Windows, но и на Linux и Mac OS.

Фреймворк является достаточно гибким в настройке и применении, в частности, все необходимые модули для разработки веб-приложения могут загружаться как отдельные компоненты с помощью пакетного менеджера Nuget [20]. Для установки доступно большое количество пакетов, в том числе для разработки веб-приложений, среди них пакеты, обеспечивающие работу библиотеки jQuery, Ajax, позволяющие проводить проверку корректности форм на стороне клиента.

На данный момент ASP.NET Core включает в себя фреймворк MVC, который объединяет функциональность MVC, Web API и Web Pages [21]. В предыдущих версиях платформы данные технологии реализовались отдельно и поэтому содержали много дублирующихся функций. Сейчас же они объединены в одну программную модель ASP.NET Core MVC, что позволяет создавать приложения реализующие функции всех предшествующих моделей.

Кроме объединения вышеупомянутых технологий в одну модель в MVC был добавлен ряд дополнительных функций. Одной из таких функций является возможность использовать тэг-хелперы (tag helper), которые позволяют создавать разметку html для страницы, используя синтаксис C#.

Исходя из выше перечисленного, можно сделать вывод, что среда разработки Microsoft Visual Studio

позволяет создавать приложения любой сложности, которые впоследствии могут быть развернуты на практически любой операционной системе [22]. Помимо прочего, стоит отметить, что компания Microsoft уделяет большое внимание интеграции своих продуктов при разработке программного обеспечения, таким образом, разработка приложения на платформе ASP.NET, которое будет осуществлять работу с базой данных, управляемой СУБД MS SQL Server, существенно упрощается. Кроме того, создание веб-приложений на платформе ASP.NET не требует от разработчика сложной настройки разметки для каждой страницы, при необходимости, среда разработки сама формирует шаблон, что позволяет уделить больше внимания проработке бизнес-процессов, происходящих в приложении.

Также для разработки приложения был использован пакет Entity Framework. Данный пакет облегчает работу с базой данных, в частности, при использовании этого пакета, для работы с базой данных нет необходимости составления запросов, которые впоследствии будет необходимо обработать СУБД [23]. Разработчик может работать с записями в базе данных как с коллекциями и отбирать, добавлять, изменять их используя специальные методы, работающие с наборами данных.

Исходя из вышесказанного, можно сделать вывод, что выбор средств разработки обусловлен в том числе простотой комбинации всех средств разработки.

3.2 Описание базы данных созданного приложения

При создании приложения была использована реляционная модель базы данных. Реляционные БД представляют собой совокупность взаимосвязанных таблиц [24], каждая из которых содержит информацию об объектах определенного типа. Строка таблицы содержит данные об одном объекте, а столбцы таблицы описывают различные характеристики этих объектов — атрибутов [25]. Записи, т. е. строки таблицы, имеют одинаковую структуру — они состоят из полей, хранящих атрибуты объекта. Каждое поле, т. е. столбец, описывает только одну характеристику объекта и имеет строго определенный тип данных. Все записи имеют одни и те же поля, только в них отображаются различные информационные свойства объекта.

Для реализации необходимого функционала создаваемому приложению нужно взаимодействовать с базой данных. База данных приложения должна хранить информацию о пользователях системы, созданных историях болезни, всех назначенных приемах медикаментов, процедур, осуществлении пользователями приема медикаментов. Исходя из этих требований была сформирована база данных, схема которой представлена на рисунке 9.

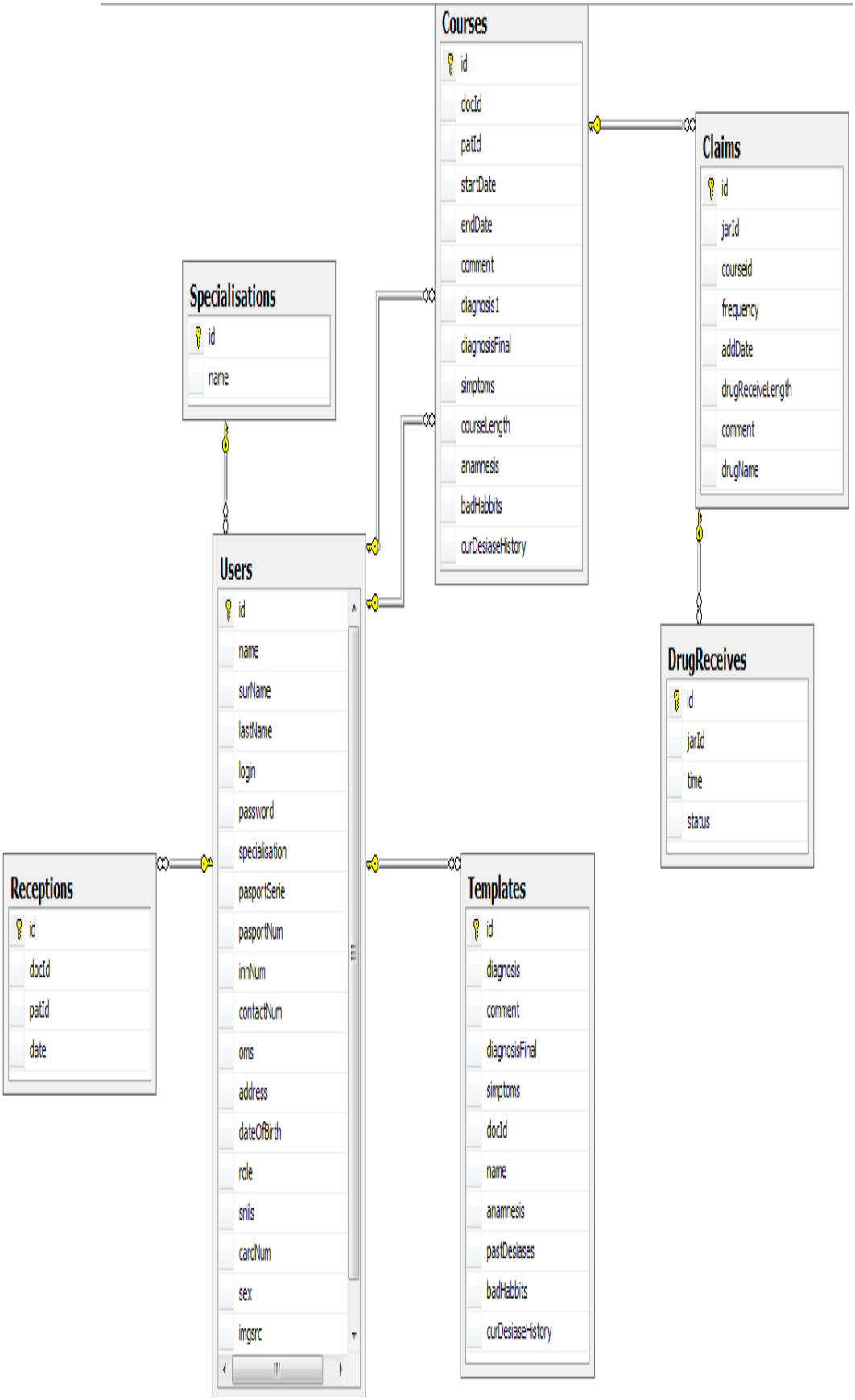


Рисунок 9 — Схема БД

Как видно на схеме, база данных состоит из следующих таблиц: Users, Courses, Claims, Shablons, Receptions, DrugReceives, Specialisations. Таблица Users содержит информацию о пользователях системы, ее структура представлена на рисунке 10.

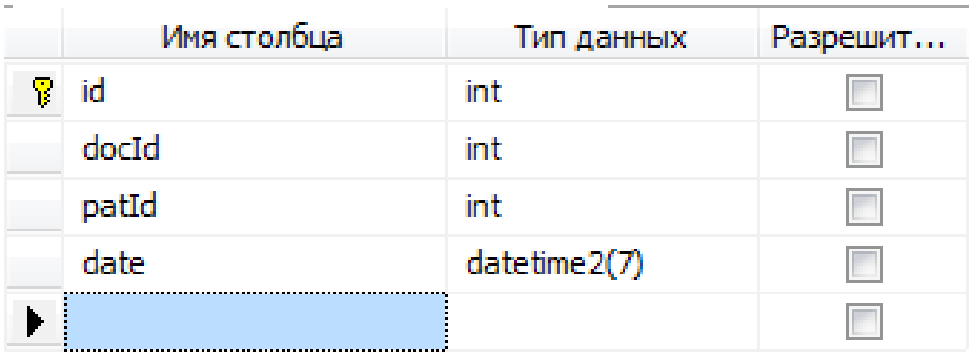
Имя столбца	Тип данных	Разрешить значения NULL
id	int	☐
name	varchar(50)	☐
surName	varchar(50)	☐
lastName	varchar(50)	☑
login	varchar(50)	☐
password	varchar(50)	☐
specialisation	int	☑
pasportSerie	int	☑
pasportNum	varchar(50)	☑
innNum	varchar(50)	☑
contactNum	varchar(50)	☑
oms	varchar(50)	☑
address	varchar(50)	☑
dateOfBirth	datetime2(7)	☐
role	int	☐
snils	varchar(50)	☑
cardNum	varchar(50)	☑
sex	int	☐
imgsrc	varchar(50)	☑
		☐

Рисунок 10 — Таблица Users

Таблица имеет поля для хранения фамилии, имени, отчества пользователя, в качестве типа данных выбран varchar(50), данный тип данных позволяет хранить

значения переменной длины, параметр 50 определяет количество байт, которое отводится под хранение строки [26], при этом, если хранимая строка занимает меньше 50 байт, то формат varchar позволяет не занимать лишнее место в памяти и под хранение строки будет отведено необходимое количество байт. Помимо этого, таблица хранит данные пользователя для авторизации, информацию о паспортных данных, данных некоторых других документов. Набор документов, необходимых пользователю для работы в системе различается, поэтому все поля, содержащие информацию о документах, позволяют хранить значения NULL, контроль за тем, чтобы все поля были заполнены правильно и недопущение NULL-значений в определенных полях осуществляется приложением. Поле role определяет роль пользователя. Роль пользователя определяет список доступных для пользователя действий. Поле specialisation является внешним ключом, соединяющим таблицы Users и Specialisations. Таблица Specialisations содержит только два поля: id- поле, содержащее первичный ключ для каждой записи и name-поле, содержащее название специализации. В данной таблице содержаться записи о специализации всех врачей, работающих в системе, и других сотрудников.

На рисунке 11 представлена структура таблицы Receptions.

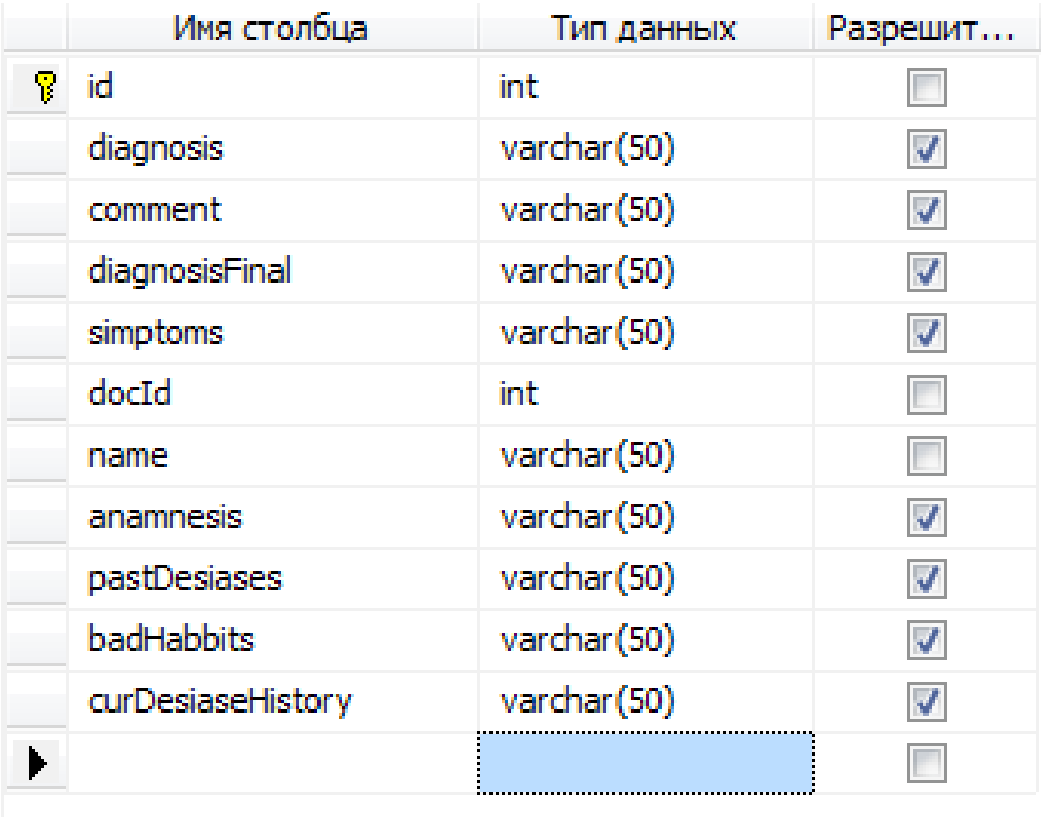


	Имя столбца	Тип данных	Разрешит…
🔑	id	int	☐
	docId	int	☐
	patId	int	☐
	date	datetime2(7)	☐
▶			☐

Рисунок 11 — Таблица Receptions

Данная таблица предназначена для хранения информации о записи пациентов к врачам. В таблице содержится два внешних ключа, определяющих записи таблицы Users, содержащие информацию о пациенте и враче, которые были определены при создании записи. Помимо этого, содержится поле, хранящее информацию о дате и времени приема.

На рисунке 12 представлена структура таблицы Templates.



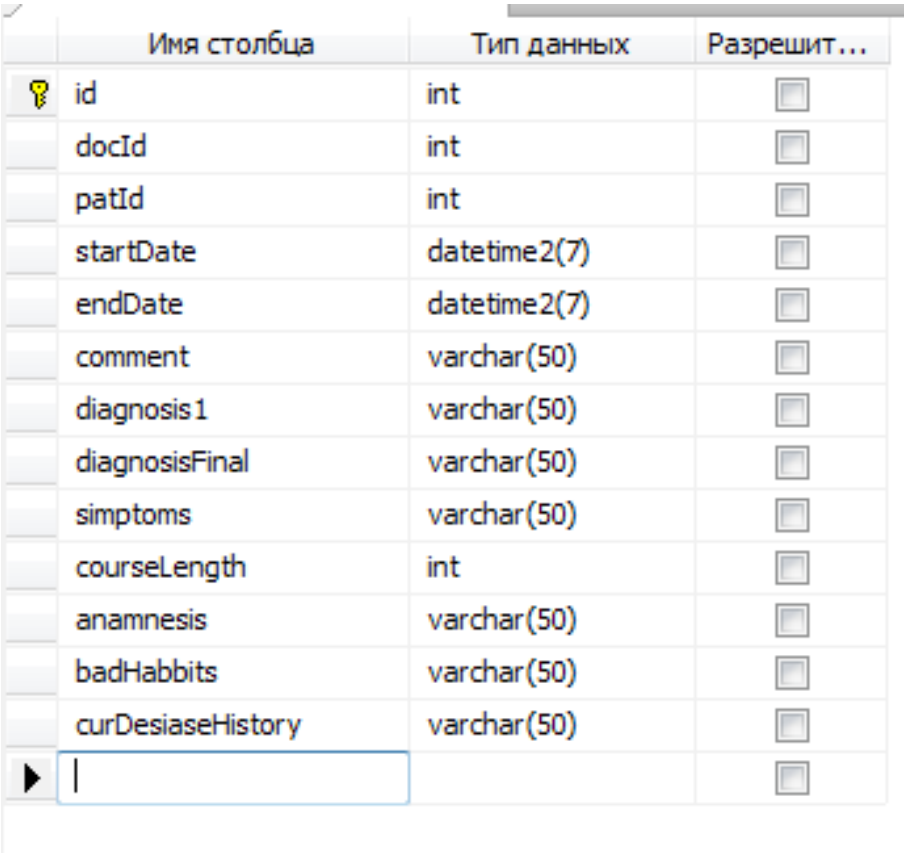
Имя столбца	Тип данных	Разрешит...
id	int	☐
diagnosis	varchar(50)	☑
comment	varchar(50)	☑
diagnosisFinal	varchar(50)	☑
simptoms	varchar(50)	☑
docId	int	☐
name	varchar(50)	☐
anamnesis	varchar(50)	☑
pastDesiases	varchar(50)	☑
badHabbits	varchar(50)	☑
curDesiaseHistory	varchar(50)	☑
		☐

Рисунок 12 — Таблица Templates

Данная таблица предназначена для хранения шаблонов, создаваемых врачом. Шаблоны представляют собой набор заготовленных значений для вставки при создании новой истории болезни. Шаблон предназначен для упрощения работы врача, предоставляя ему возможность ускорить процесс заполнения формы создания новой истории болезни путем подстановки уже заготовленного значения в поля. Каждый шаблон позволяет хранить заранее приготовленные значения для полей: жалобы пациента, диагноз, пояснение к поставленному диагнозу, наличие вредных привычек, истории текущего заболевания, анамнеза. Данные поля необходимы при создании истории болезни и позволяют отметить важные особенности пациента, которые

определяют методы и средства его лечения. Каждый врач может в любой момент времени определить для себя новый шаблон и, впоследствии, его использовать.

На рисунке 13 представлена структура таблицы Courses.



Имя столбца	Тип данных	Разрешит...
id	int	
docId	int	
patId	int	
startDate	datetime2(7)	
endDate	datetime2(7)	
comment	varchar(50)	
diagnosis1	varchar(50)	
diagnosisFinal	varchar(50)	
simptoms	varchar(50)	
courseLength	int	
anamnesis	varchar(50)	
badHabbits	varchar(50)	
curDesiaseHistory	varchar(50)	

Рисунок 13 — Таблица Courses

Данная таблица предназначена для хранения информации об историях болезни, созданных с использованием системы. Для хранения данной информации в таблице определены поля для хранения даты начала лечения, окончания лечения, жалоб пациента при обращении к врачу, наличия вредных привычек у пациента, анамнеза жизни пациента, диагноза, определенного врачом

и пояснения к нему. Помимо этого, история болезни должна содержать информацию о пациенте и враче, осуществляющем его лечение, для этого в таблице есть два поля являющиеся внешними ключами, которые содержат информацию о записях в таблице Users, определяющих лечащего врача и пациента, которому необходимо лечение.

На рисунке 14 представлена структура таблицы Claims.

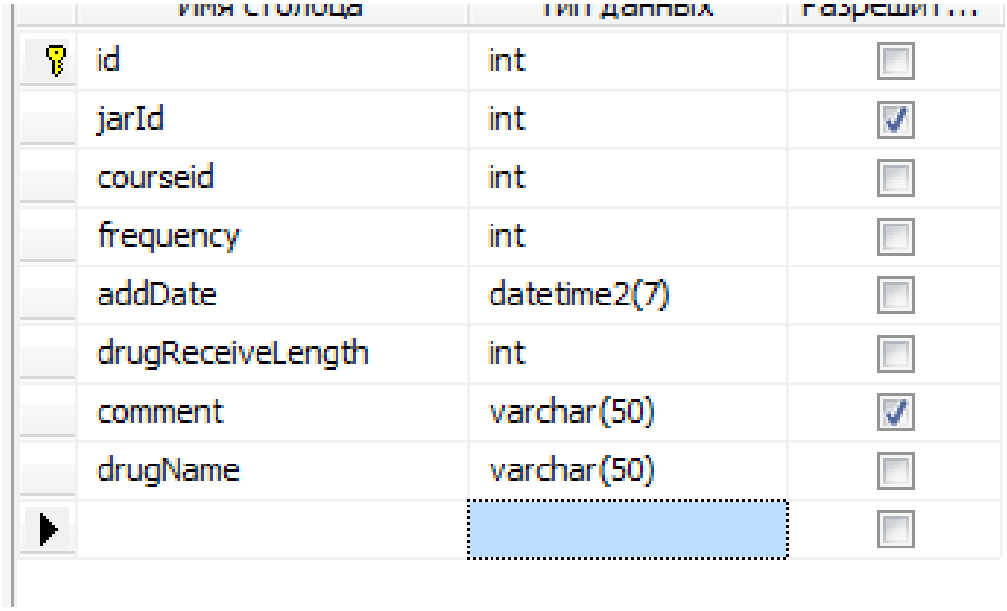


Рисунок 14 — Таблица Claims

Таблица предназначена для хранения информации о всех случаях выдачи банок с лекарствами для пациентов. Каждая такая запись должна содержать информацию о номере банки, которая содержит медикамент, названии действующего вещества медикамента, которое содержится в банке, дате выдачи банки и продолжительности приема медикамента - это необходимо для тех случаев, когда прием определенного медикамента необходимо осуществлять не

на протяжении всего лечения, а в течении какого-либо отрезка времени. Поле comment предназначено для хранения дополнительной информации о правилах приема медикамента, определяемых врачом.

Для хранения информации об использовании банок в базе данных определена таблица DrugReceives, схема которой представлена на рисунке 15.

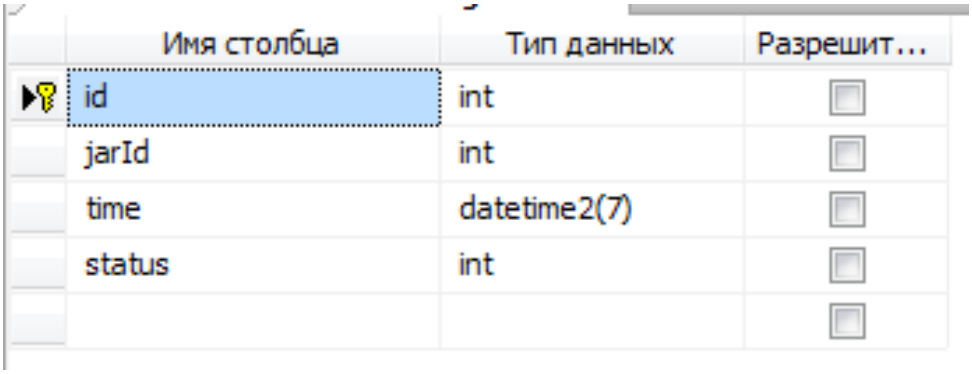


Рисунок 15 — Таблица DrugReceives

Таблица позволяет сохранить номер банки, которая была открыта или, напротив, не открыта в назначенное время, времени, которое задается на основе информации, получаемой от «Умной банки», статуса банки. Статус банки определяет то, что было осуществлено с банкой: открытие, пропуск открытия в назначенное время.

3.3 Описание технологий, использованных при создании

При создании приложения была использована технология MVC. Данная концепция подразумевает

разделение функционала приложения между тремя группами элементов: модель, представление, контроллер.

- **Модель** — представляет собой объектную модель предметной области и включает в себя данные и методы работы с этими данными, реагирует на запросы из контроллера, возвращая данные. При этом модель не содержит в себе информации о способах визуализации данных или форматах их представления, а также не взаимодействует с пользователем напрямую.
- **Представление** — отвечает за то, как информация будет представлена пользователю. Одни и те же данные могут представляться различными способами и в различных форматах. Например, коллекцию объектов при помощи разных представлений можно представить на уровне пользовательского интерфейса как в табличном виде, так и списком.
- **Контроллер** — обеспечивает связь между пользователем и системой, использует модель и представление для реагирования на действия пользователя. Как правило, на уровне контроллера осуществляется фильтрация полученных данных и авторизация — проверяются права пользователя на выполнение действий или получение информации.

Преимуществом данного подхода является упрощение разработки приложений, в особенности крупных [27]. Так же упрощается работа по развитию системы, переопределению некоторых механик, осуществление

поддержки созданного программного обеспечения. Это возможно благодаря грамотному разделению функционала приложения, что упрощает поиск и устранение ошибок и других проблем. Так, если необходимо изменить то, как пользователю будут отображены определенные данные, нужно лишь исправить соответствующее представление [28], а если необходимо ограничить доступ какой-либо группы пользователей к определенному функционалу внести проверку доступности функционала для роли в контроллере. Выигрыш от использования такой архитектуры заключается в том, что она позволяет упорядочить код, распределив его по уровням, каждый из которых определяет сферу ответственности [29]. Это минимизирует взаимозависимость программных компонент, что в свою очередь облегчает их последующую модификацию. Доработка и развитие такой системы становится проще [30].

Так же стоит упомянуть, что в приложении использована технология AJAX-запросов. Взаимодействие клиента и сервера при работе веб-приложения осуществляется следующим образом: на странице клиент пользователь совершает какое-либо действие, после чего формируется запрос к серверу, сервер, в свою очередь, обрабатывает этот запрос и возвращает пользователю полностью сформированную страницу с необходимой информацией [31]. Данный подход позволяет реализовать любой функционал приложения, но при этом содержит некоторые незначительные недостатки, одним из таких

недостатков является то, что при использовании такого подхода после совершения определенных действий пользователь вынужден ждать, когда сервер обработает запрос, отправит ответ и браузер отобразит полученную от сервера информацию, что делает работу в приложении менее удобным. Использование технологии AJAX позволяет избежать такой проблемы, как ожидание формирования полной DOM-модели страницы [32]. При использовании данного подхода, по-прежнему после действий пользователя формируется запрос к серверу, но в ответ сервер предоставляет не всю страницу целиком, а ту часть, которая должна быть изменена. Реализация данного подхода совместно с MVC выглядит так: в представлении определен код на языке java-script, который реагирует на действия пользователя и формирует запрос к методу контроллера с определенным набором параметров, запрос обрабатывается и в ответ пользователю отправляется частичное представление, которое заменяет собой блок html, который пользователь хотел изменить. Для частичного представления в ASP.NET, в отличии от обычного, не определена мастер-страница, страница, содержащая общие для всех страниц приложения стили, скрипты, и отвечающая за формирование единообразного вида всех страниц приложения [33].

Помимо прочего, при создании приложения использовался дополнительный пакет Entity Framework. Данный пакет упрощает работу с базой данных в приложении. Как правило, для работы с базой данных

используются запросы, написанные на языке, который используется в СУБД, осуществляющей работу с базой данных, в частности в СУБД MS SQL Server используется разновидность языка SQL, называемая T-SQL [34]. Данный диалект позволяет создавать любые, даже очень сложные запросы [35], содержащие вложенные подзапросы и так далее, при этом для работы приложения необходимо формировать запрос на основе выбранных пользователем параметров, что может быть не так просто, ведь необходимо предусмотреть все комбинации входных параметров, которые может выбрать пользователь и на основе этого сформировать скрипт, который будет формировать строку запроса из определенных, уже готовых, блоков, после чего отправить запрос на исполнение СУБД. При использовании пакета Entity Framework отпадает необходимость формирования запросов к СУБД, а взаимодействие с базой данных становится похожим на работу с коллекциями [36]. Запросы формируются не разработчиком ПО, а инструментами пакета на основе той логики, которую заложил разработчик. Entity Framework позволяет абстрагироваться от базовых понятий реляционных БД и работать с понятиями более приближенными к программированию такими, как объекты, их поля и так далее. Важно отметить, что Entity Framework предоставляет широкий функционал по работе с базами данных, на данный момент существует три способа взаимодействия с базами данных [37]:

•Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных.

•Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере.

•Code first: разработчик создает класс модели данных, которые будут храниться в БД, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

3.4 Описание разработанного приложения

Разработанная система представляет собой веб-приложение для поликлиники. Приложение позволяет любому потенциальному пациенту узнать информацию о клинике, контактный телефон, возможность записаться на прием к врачу.

На главной странице содержится информация о клинике, список врачей, осуществляющих врачебную деятельность в данном учреждении. Записаться на прием можно позвонив по телефону, указанному на главной странице.

Приложение будет позволять сотрудникам регистратуры осуществлять следующие функции: регистрация пациентов, запись пациентов на прием к определенному врачу или к любому из специалистов, необходимых пациенту.

С помощью приложения врач может увидеть все запланированные приемы, информацию о пациенте, дату

приема. Для пришедшего пациента врач может создать новую историю болезни, в которой необходимо указать анамнез жизни пациента, наличие вредных привычек, историю текущего заболевания, диагноз и объяснить причину того, почему врач определил именно такой диагноз. Для упрощения данных действий врач может применить шаблон для создания истории болезни. Врач сам может создать необходимый ему шаблон и, сохранив его, применить при создании истории болезни. Шаблон позволяет создать стандартное содержимое для пунктов: анамнез жизни пациента, наличие вредных привычек, диагностированное заболевание, и описание причин выбора заболевания. Применение шаблона реализуется с помощью технологии Ajax. Данная технология примечательна тем, что в отличии от привычного принципа обмена информацией клиента и сервера, при которой для получения новой информации необходимо осуществить запрос к серверу, в ответ на который сервер отправит заново сформированную страницу с необходимой информацией, при использовании Ajax осуществляется асинхронный запрос к серверу, и сервер предоставляет не полностью сформированное представление, а только часть общего представления, которую необходимо заменить. Использование данной технологии делает работу пользователя более удобной, исключая необходимость ожидания формирования новой DOM-модели после отправки запроса на сервер.

На основе информации, полученной при анализе структуры истории болезни, который был проведен ранее, была создана форма создания новой истории болезни. Форма содержит поля для ввода причин обращения к врачу, истории заболевания, наличия вредных привычек, постановки и обоснования диагноза. Информация о имени, фамилии пациента, номера полиса ОМС берется из базы данных, для упрощения ввода врачом.

Для создания истории болезни врач должен заполнить специальную форму в представлении. Специально для представления, в котором осуществляется создание истории болезни, была создана модель DocAddCourseVM. Зачастую, представление тех или иных сущностей в приложении отличается от того, как пользователь привык работать с ними в повседневной жизни, поэтому для того, чтобы представить пользователю весь необходимый функционал требуется объединить несколько моделей приложения. Так модель DocAddCourseVM совмещает в себе элементы базовых моделей приложения, отвечающих за работу с базой данных и выполнение других функций.

На основе полей этой модели формируется представление, которое содержит форму для заполнения врачом. При формировании страницы с формой можно использовать возможности предоставляемые платформой ASP.NET, в частности, проверка корректности данных формы на стороне клиента возможна на основе правил, задаваемых при описании модели. Правила могут быть

различными от указания на обязательное заполнение поля, задания шаблона введенного выражения, до указания максимальной длины содержимого элемента. Форма представлена на рисунке 16.

После заполнения истории болезни врач нажимает кнопку «Сохранить», после чего формируется POST-запрос, который обрабатывается методом DocAddCourse контроллера HomeController.

Контроллер осуществляет валидацию полученной из представления модели с помощью свойства ModelState.IsValid. Это свойство позволяет провести валидацию модели на основе описания этой модели, т.е. нет необходимости проверять значение каждого поля, можно осуществить проверку заполнения всех полей правильными значениями, используя данное свойство.

После проверки корректности данных полученной модели пользователю будет возвращена страница с формой добавления курса, если форма была заполнена неверно, иначе будет вызван метод Save() модели DocAddCourseVM, который сохранит полученные данные в соответствующие таблицы базы данных.

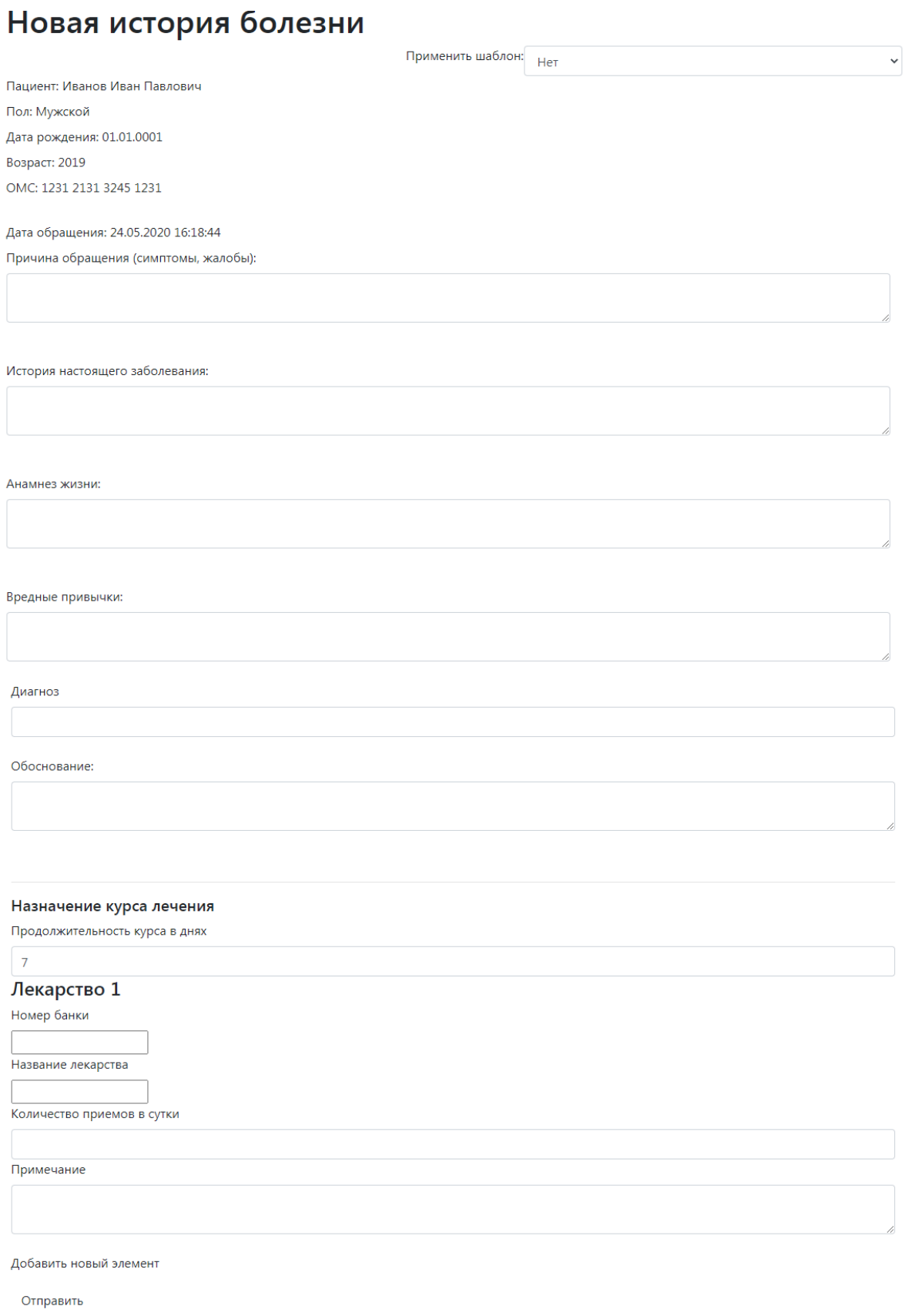


Рисунок 16 — Оформление истории болезни в созданном приложении

Помимо прочего, при использовании пакета Entity Framework, работа с базой данных претерпевает некоторые изменения.

Так, при использовании данного пакета, для добавления записи не составляется запрос на языке SQL с последующим запуском запроса на исполнение СУБД, вместо этого добавление записи выглядит как работа с коллекциями в некоторых языках программирования, в частности добавление реализуется с помощью строки кода “db.Courses.Add(cou);” и последующим сохранением изменений в базе данных. Если разобрать данную строку кода, то можно понять, как устроена работа с базой данных: db- контекст базы данных, экземпляр класса DBCon, содержащий строку подключения к базе данных, Courses- таблица базы данных, содержащая записи о созданных историях болезни, метод Add, осуществляющий добавление новой записи и создающий первичный ключ для новой записи и объект cou- экземпляр класса Course.

Шаблон содержит базовую информацию для заполнения, такую как: причина обращения в медицинское учреждение, история настоящего заболевания, вредные привычки пациента, анамнез, диагноз и его обоснование. В некоторых случаях, пациенты обращаются за медицинской помощью с достаточно типичными жалобами и, соответственно, врач пишет в истории болезни информацию, подвергая ее лишь незначительным изменениям. Для упрощения этой работы в системе

присутствует возможность применения шаблонов заполнения.

Так же врачу доступно формирование шаблонов для заполнения. Шаблон для заполнения содержит заготовленные значения для полей: диагноз, обоснование, наличие вредных привычек, причина обращения. Врач сам может составить те шаблоны, которые посчитает необходимыми и заполнить их на свое усмотрение. В дальнейшем врач при создании истории болезни может выбрать шаблон, который хочет применить и, таким образом, часть полей формы будет заполнена автоматически. Данная возможность осуществляется с помощью технологии Ajax. При выборе нужного шаблона формируется запрос к серверу, в ответ на который сервер возвращает частичное представление, которое заменяет имеющиеся поля формы. Добавление медикаментов в курс лечения осуществляется динамически на странице с основной информацией о пациенте и заболевании. При добавлении медикамента врач указывает номер банки, в которой будет храниться медикамент, его название, частоту приема, время приема и примечание, которое врач может добавить при необходимости.

Форма добавления шаблона представлена на рисунке 17.

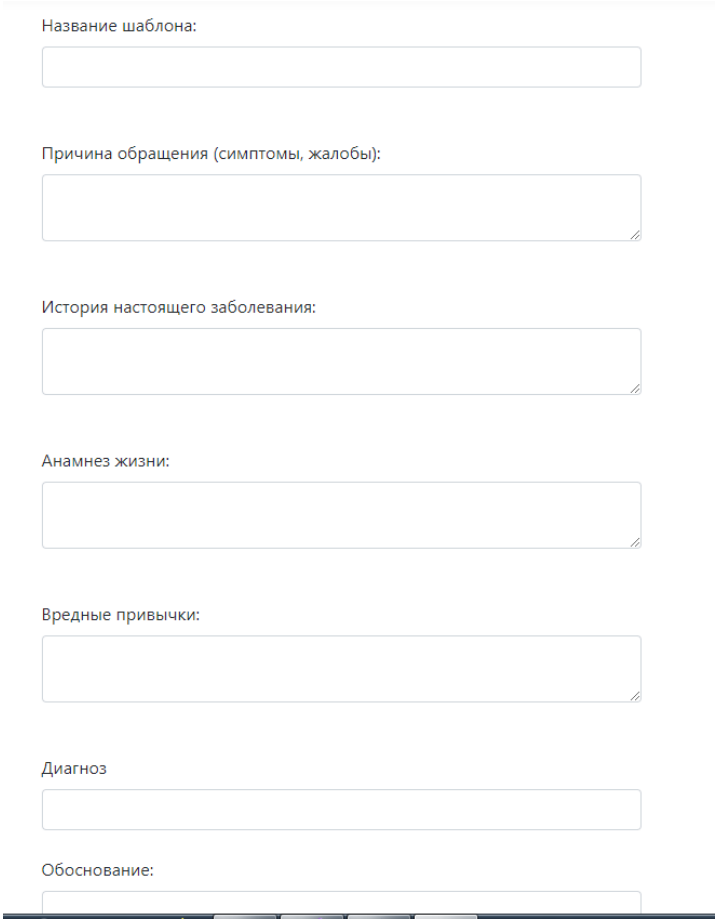


Рисунок 17 — Создание шаблона

Кроме этого, врач может посмотреть информацию, содержащуюся в любой созданной им истории болезни. Пример показан на рисунке 18.

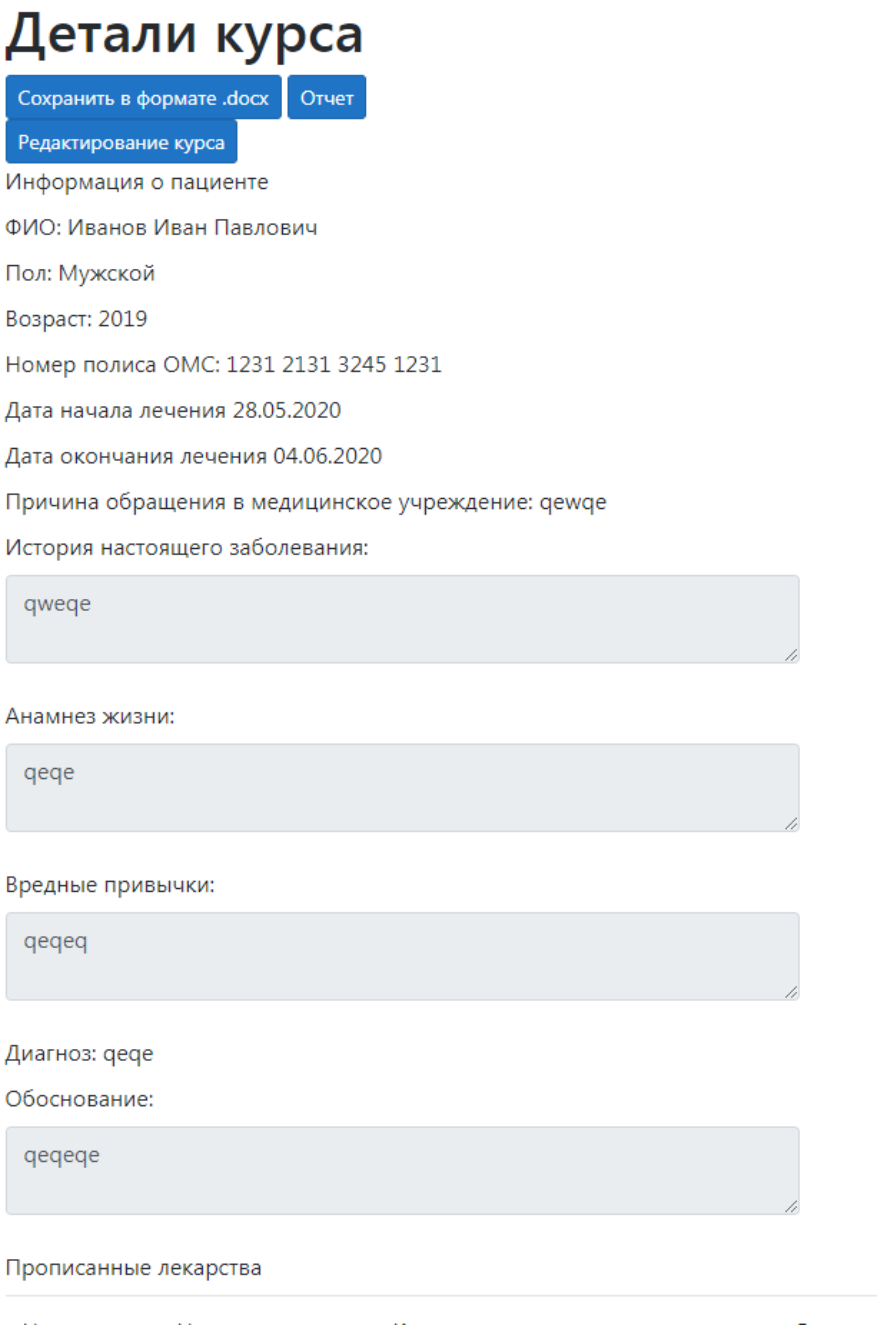


Рисунок 18 — Просмотр истории болезни

В некоторых случаях врачу требуется изменить некоторую информацию, содержащуюся в истории болезни, в частности диагноз. Это может случиться из-за неправильной постановки диагноза в следствии ошибки врача, неполноты информации о пациенте, некоторых особенностях его организма. Помимо прочего, случается

так, что уже назначенное лечение оказывается неэффективным, для решения этих проблем необходимо предоставить врачу возможность исправить ошибочный диагноз и скорректировать курс лечения пациента. Корректировка истории болезни представлена на рисунке 19.

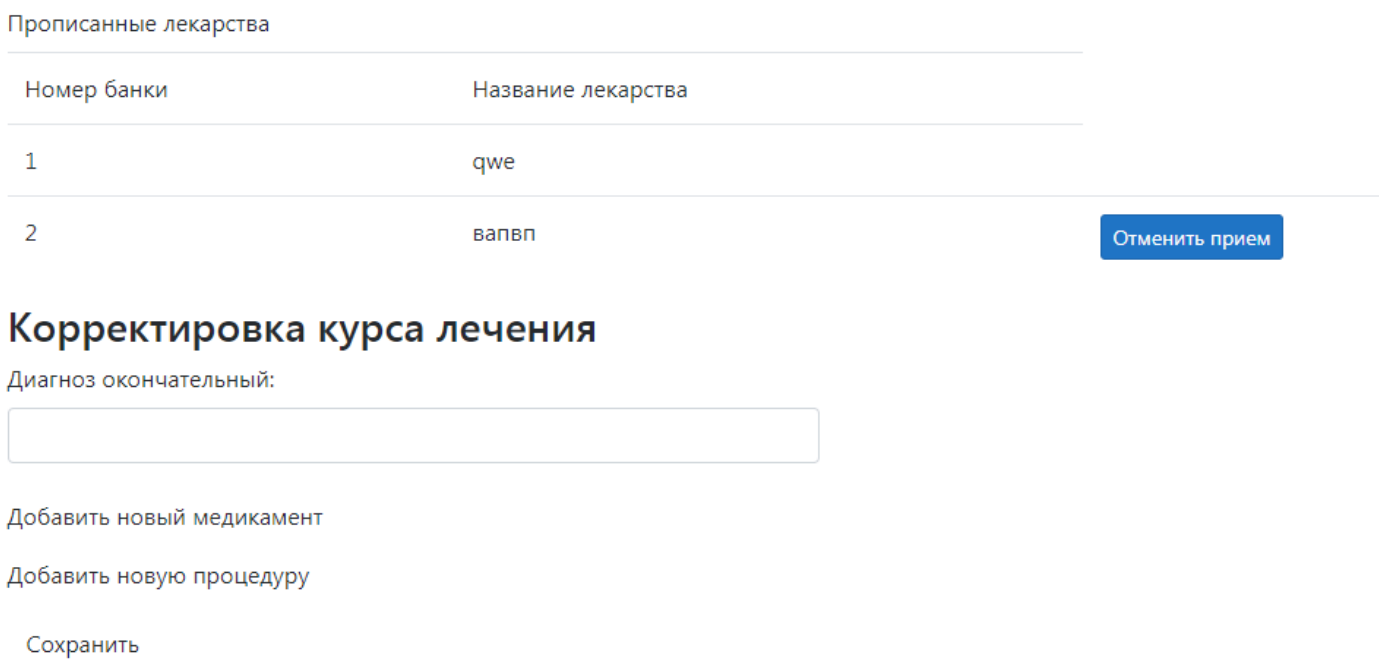
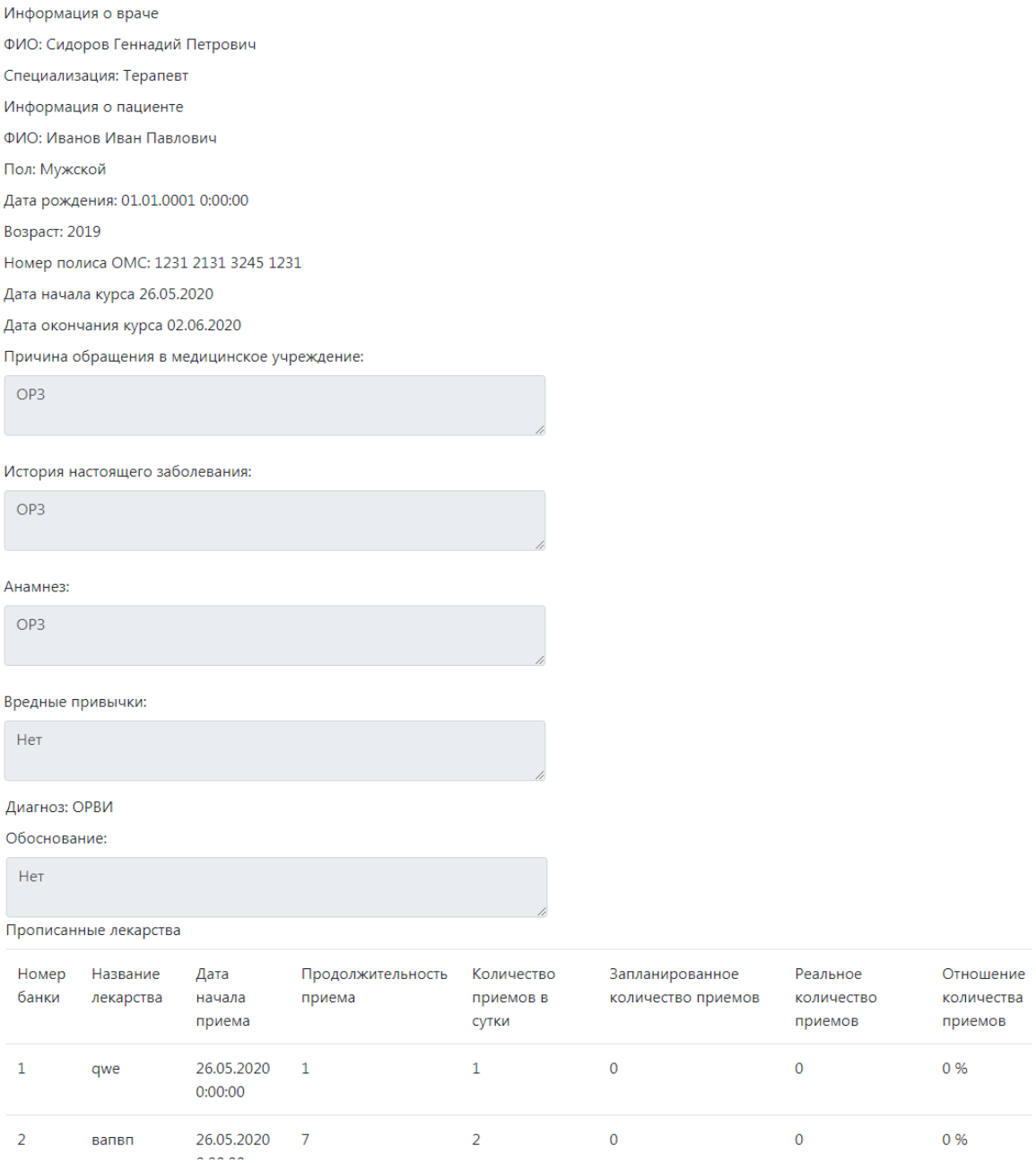


Рисунок 19 — Редактирование истории болезни

При редактировании врачу предоставляется возможность поставить окончательный диагноз, добавить новые медикаменты в курс лечения и отменить прием тех медикаментов, прием которых врач сочтет необоснованным, при этом врач может отменить прием только тех медикаментов, прием которых не окончен. После отмены приема информация о ходе приема медикамента остается доступной, что сделано для того, чтобы не исказить картину лечения пациента.

Для любого курса лечения доступна функция формирования отчета. Отчет о курсе лечения содержит информацию о лечащем враче, пациенте, заболевании, прописанных медикаментах, статистику их приема. Пример отчета представлен на рисунке 20.



Информация о враче

ФИО: Сидоров Геннадий Петрович

Специализация: Терапевт

Информация о пациенте

ФИО: Иванов Иван Павлович

Пол: Мужской

Дата рождения: 01.01.0001 0:00:00

Возраст: 2019

Номер полиса ОМС: 1231 2131 3245 1231

Дата начала курса 26.05.2020

Дата окончания курса 02.06.2020

Причина обращения в медицинское учреждение:

ОРЗ

История настоящего заболевания:

ОРЗ

Анамнез:

ОРЗ

Вредные привычки:

Нет

Диагноз: ОРВИ

Обоснование:

Нет

Прописанные лекарства

Номер банки	Название лекарства	Дата начала приема	Продолжительность приема	Количество приемов в сутки	Запланированное количество приемов	Реальное количество приемов	Отношение количества приемов
1	qwe	26.05.2020 0:00:00	1	1	0	0	0 %
2	вапвп	26.05.2020	7	2	0	0	0 %

Рисунок 20 — Просмотр подробной информации об истории болезни

При этом также доступна возможность сохранить полученный отчет в формате docx. Это реализуется с помощью пакета GemBox.Document. Данный пакет разработан компанией GemBox d.o.o. и позволяет формировать документы в форматах: docx, pdf, rtf, xml, html, и других [38]. Помимо этого, компания разработала ряд других продуктов, позволяющих работать с таблицами Excel и многим другим. Пакет GemBox.Document существенно упрощает работу с документами, позволяет создавать документы, включая составление разметки документа, редактирование документов и многое другое.

Любой пользователь системы может поменять личную информацию, в том числе паспортные данные, данные для авторизации. Это сделано для удобства пользователей, так как некоторые документы должны меняться с течением времени или быть утрачены и впоследствии восстановлены. Данные для входа пользователь может изменить, если сочтет их неудобными для ввода или запоминания.

Важно отметить, что заполнение форм является одним из слабых мест любого приложения, поэтому необходимо строго контролировать вводимые пользователем данные, чтобы вся информация, хранимая и обрабатываемая приложением была правильной, не содержала ошибок. Для контроля вводимых пользователем данных существует процесс валидации- проверки введенных значений, их соответствия определенным требованиям. Валидация может происходить на стороне сервера, как это было

показано на рисунке 6. В этом случае проверка происходит только после отправки формы на сервер, но некоторые ошибки заполнения формы можно «поймать» до отправки на сервер. В HTML5 для этого используется атрибут pattern, который содержит шаблон заполнения поля формы. В ASP.NET данный функционал можно определить в классе модели [39]. Для этого к каждому полю необходимо дописать специальные атрибуты. В частности, можно сделать поле обязательным для заполнения с помощью атрибута Required [40], так же можно ввести определенные ограничения на вводимые значения с помощью атрибута RegularExpression, в котором можно указать допустимые символы, длину последовательности символов, ввести ограничение на первый или какой-либо другой символ. В частности, для корректного заполнения поля, содержащего фамилию, можно использовать следующее регулярное выражение: [RegularExpression("^[А-Я||Э-Я]+[а-я]*$")]. Данное выражение вводит правило, что фамилия должна начинаться с заглавной буквы, состоять из букв русского алфавита, и не может начинаться с определенных букв. Помимо этого, есть и другие атрибуты, ограничивающие длину введенной строки, максимальное и минимальное значение поля и другие. Для любого атрибута можно задать свое сообщение об ошибке. Для осуществления валидации на стороне клиента необходимо указать все нужные атрибуты и в представлении, где будет происходить заполнение данных формы указать правила валидации формы [40]. После чего для каждого поля добавляется тег

span, выводящий сообщение о неправильном заполнении, если такое имело место быть. Кроме того, перед формой добавляется блок, который будет содержать список всех ошибок заполнения формы.

Запись к врачу

Важной частью работы поликлиники является планирование посещений пациентов. Планирование посещений позволяет организовать работу врачей и других сотрудников более удобным способом, избежать излишней нагрузки на специалистов в одни дни и недостаток пациентов в другие. Данная задача лежится на сотрудников регистратуры. Сотрудники регистратуры должны принимать заявки от пациентов с просьбой на оказание медицинской помощи тем или иным специалистом. Заявки могут поступать различными способами, как правило запись осуществляется по контактному номеру телефона учреждения. Важно отметить, что пациент может попросить записать его к какому-либо конкретному врачу или же к любому свободному специалисту, после чего оговаривается дата и время приема. Для реализации поставленных задач в приложении предусмотрены специальные инструменты.

Пример записи к одному из специалистов представлен на рисунке 23.

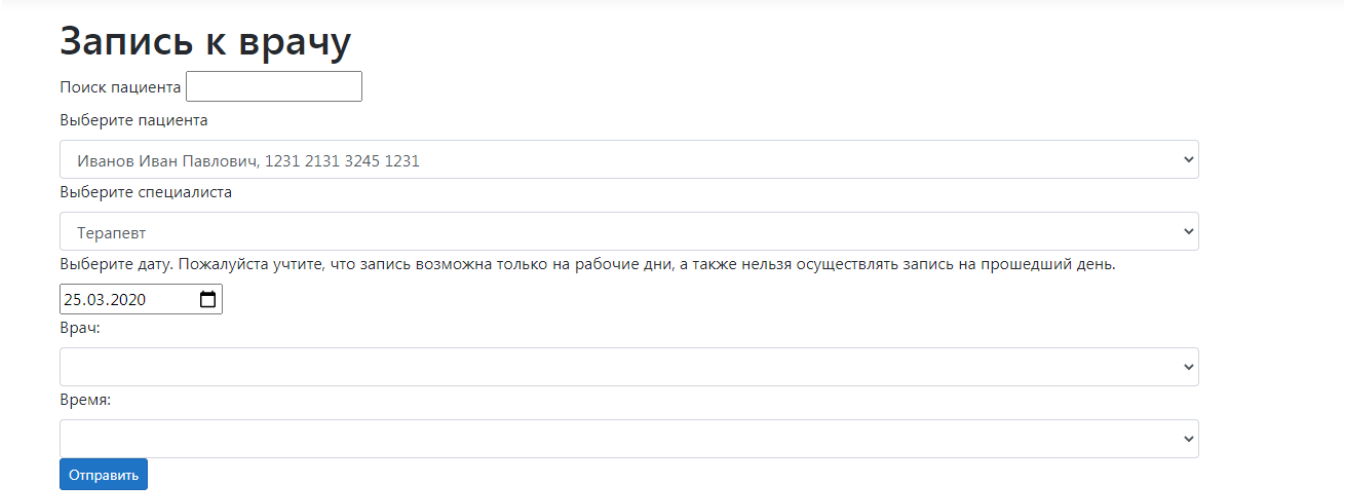


Рисунок 23 — Запись пациента к специалисту

Запись на прием к специалисту осуществляется после регистрации пациента в специальной форме. Сотрудник регистратуры должен выбрать необходимого пациента из списка всех пациентов. Для удобства сотрудник может осуществить поиск по фамилии, имени, отчеству пациента, номеру полиса ОМС, либо по части данной информации. Данный функционал осуществляется с помощью технологии Ajax. После выбора пациента необходимо выбрать специальность врача, к которому хочет попасть пациент и дату приема. Выбор даты, для удобства сотрудника, осуществляется с помощью специального элемента HTML5- DatePicker. Это позволяет избежать проблем с правильным форматированием даты, ошибок во введенных данных, разделителях значений. После выбора даты в список доступных специалистов загружаются все врачи, соответствующие следующим критериям: имеют

необходимую специальность, имеют по меньшей мере одно свободное место в графике на данную дату. В другой список- список доступных времен приема записываются все доступные времена приема. Если, по каким-либо причинам, найти свободного специалиста не удается, то из формы исчезают поля для выбора врача и времени приема, и сотруднику выводится сообщение с просьбой изменить входные данные: дату приема, врача, и т.д. Это осуществляется с помощью AJAX-запросов, для примера на рисунке 26 приведен java-script код, отвечающий за поиск пациентов.

Сформированный запрос обрабатывается контроллером, после чего пользователю возвращается частичное представление, заменяющее требуемый блок.

Помимо этого, существует возможность осуществить запись к конкретному врачу, если этого желает пациент. Для этого в списке всех специалистов необходимо найти нужного и нажать на кнопку «Добавить курс», данная процедура представлена на рисунке 24.

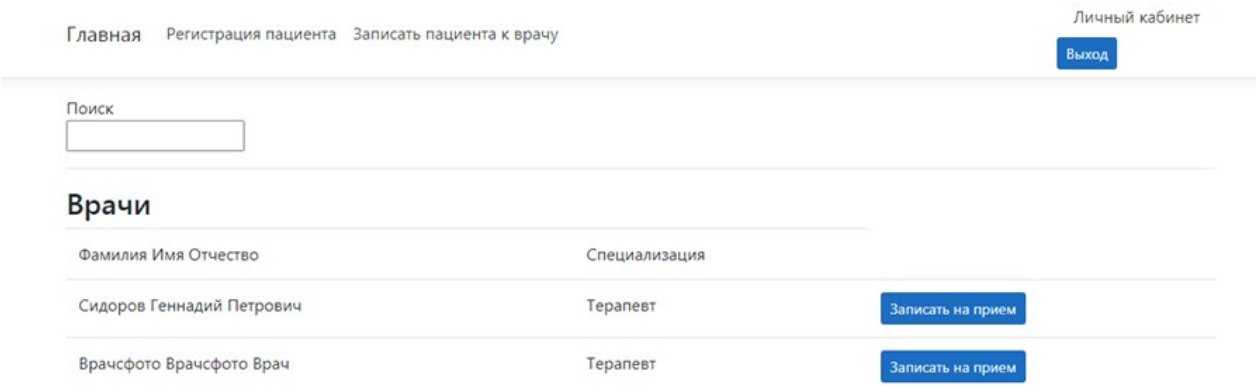


Рисунок 24 — Список специалистов

После чего пользователь будет перенаправлен к форме регистрации приема. Форма мало чем отличается от записи на прием к специалисту за исключением того, что сотруднику не предоставляется возможность выбора специальности и врача, ведь это было осуществлено на предыдущем шаге. Форма представлена на рисунке 25.

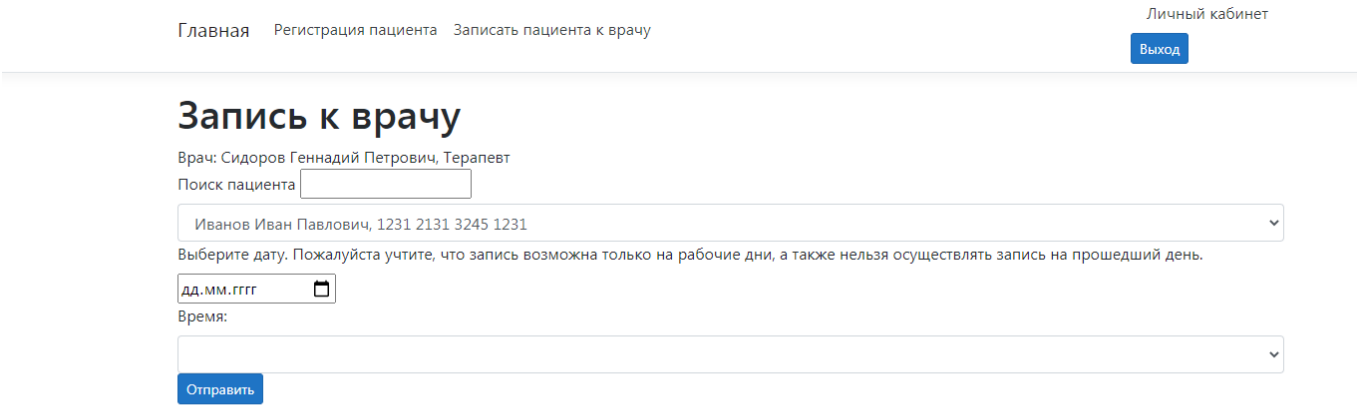


Рисунок 25 — Запись к врачу

Помимо прочего, в работе любого медицинского учреждения важной частью является учет сотрудников и пациентов. Любое обращение в медицинское учреждение подробно документируется, каждый из пациентов и, в особенности, сотрудников должен быть учтен, о нем обязательно должны быть составлены записи, содержащие основную информацию: фамилию, имя, отчество, данные некоторых документов. Для приложения, предназначенного для использования в медицинском учреждении, реализация подобного функционала является

чрезвычайно важной, поэтому в созданном приложении существует возможность регистрации сотрудников и пациентов.

Регистрацию пациентов осуществляют сотрудники регистратуры. При регистрации пациента обязательно указываются его фамилия, имя, отчество, дата рождения, пол, номер полиса ОМС, серия и номер паспорта. Эти данные необходимы как для точной постановки диагноза, так и для осуществления различных финансовых операций. Страховка по полису ОМС позволяет пациенту получить некоторые услуги, предоставляемые медицинским учреждением бесплатно. Регистрация пациентов осуществляется в специальной форме, в которой сотрудник регистратуры указывает необходимую информацию о пациенте, при этом важно предусмотреть случаи, когда в базу данных вносится информация о человеке, который уже был зарегистрирован ранее. Для этого, при заполнении полей формы, содержащих информацию о номере полиса, серии и номере паспорта выполняются запросы к серверу, осуществляемые с помощью технологии AJAX. Это позволяет пользователю получить информацию о том, что такой пациент уже зарегистрирован еще до того, как форма будет отправлена. Кроме этого, подобная проверка осуществляется и после отправки формы на стороне сервера, после чего пациент добавляется в базу данных, либо пользователю возвращается форма с указанием на то, что, либо пациент уже зарегистрирован, либо при заполнении формы возникла ошибка.

Любой зарегистрированный пользователь, пациент или сотрудник, получает возможность авторизации в приложении. При регистрации указывается только логин нового пользователя, пароль изначально устанавливается один для всех, что сделано для упрощения процедуры регистрации пользователей. Впоследствии пароль может быть изменен пользователем в любой момент в личном кабинете, там же пользователь может изменить некоторую другую информацию. Форма регистрации пациента и сотрудника отличается набором полей, которые должны быть заполнены, так, при регистрации пациента, нет необходимости указывать его специализацию

Форма регистрации пациента представлена на рисунке 26.

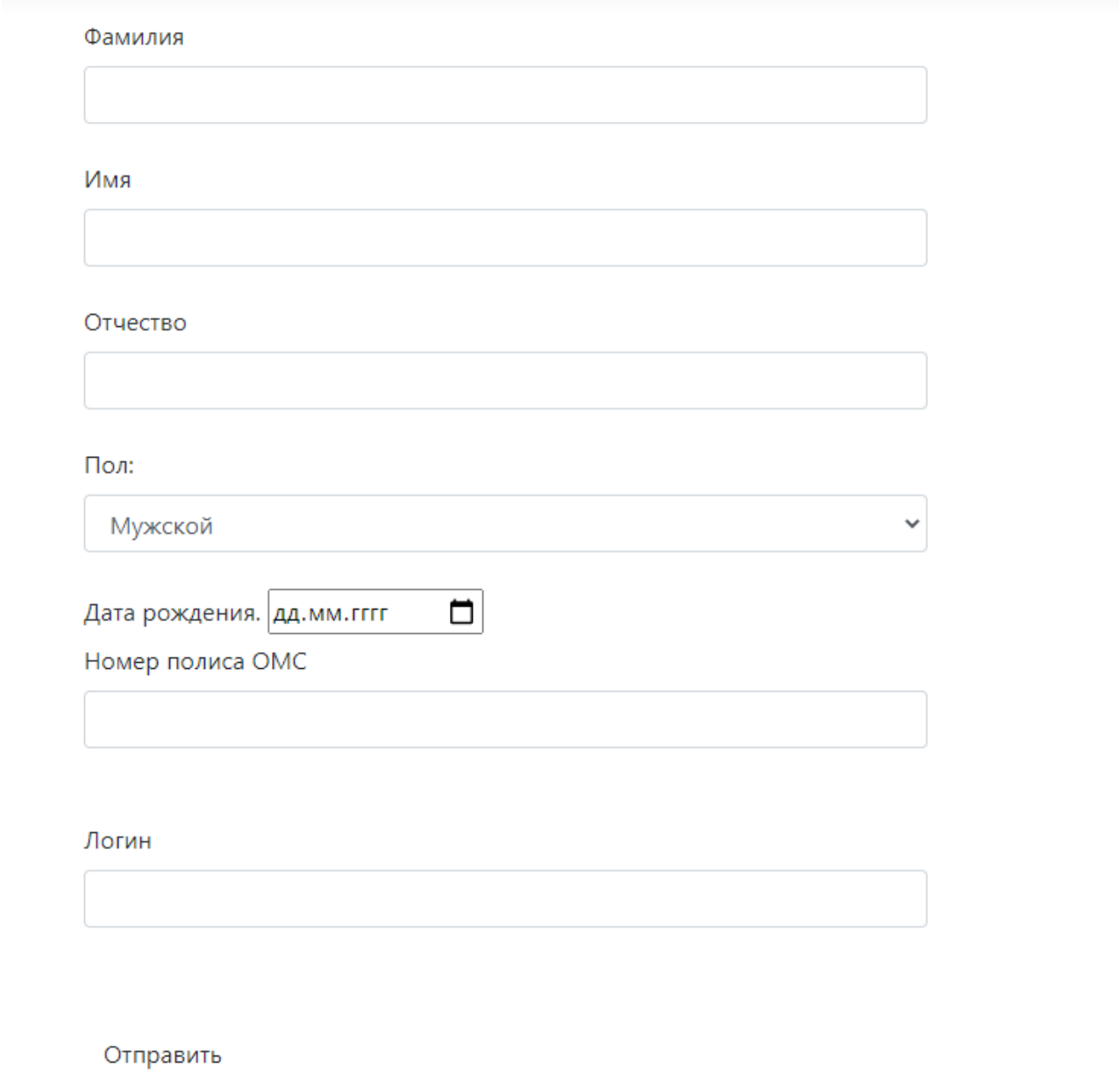


Рисунок 26 — Регистрация пациента

В личном кабинете пользователь может посмотреть и, при необходимости, изменить информацию о себе, данные некоторых документов. Эта возможность необходима при возникновении ошибок при регистрации пациента, смене некоторых документов, которая может происходить по причине истечения срока действия предыдущего документа

или его утраты. Личный кабинет представлен на рисунке 27.

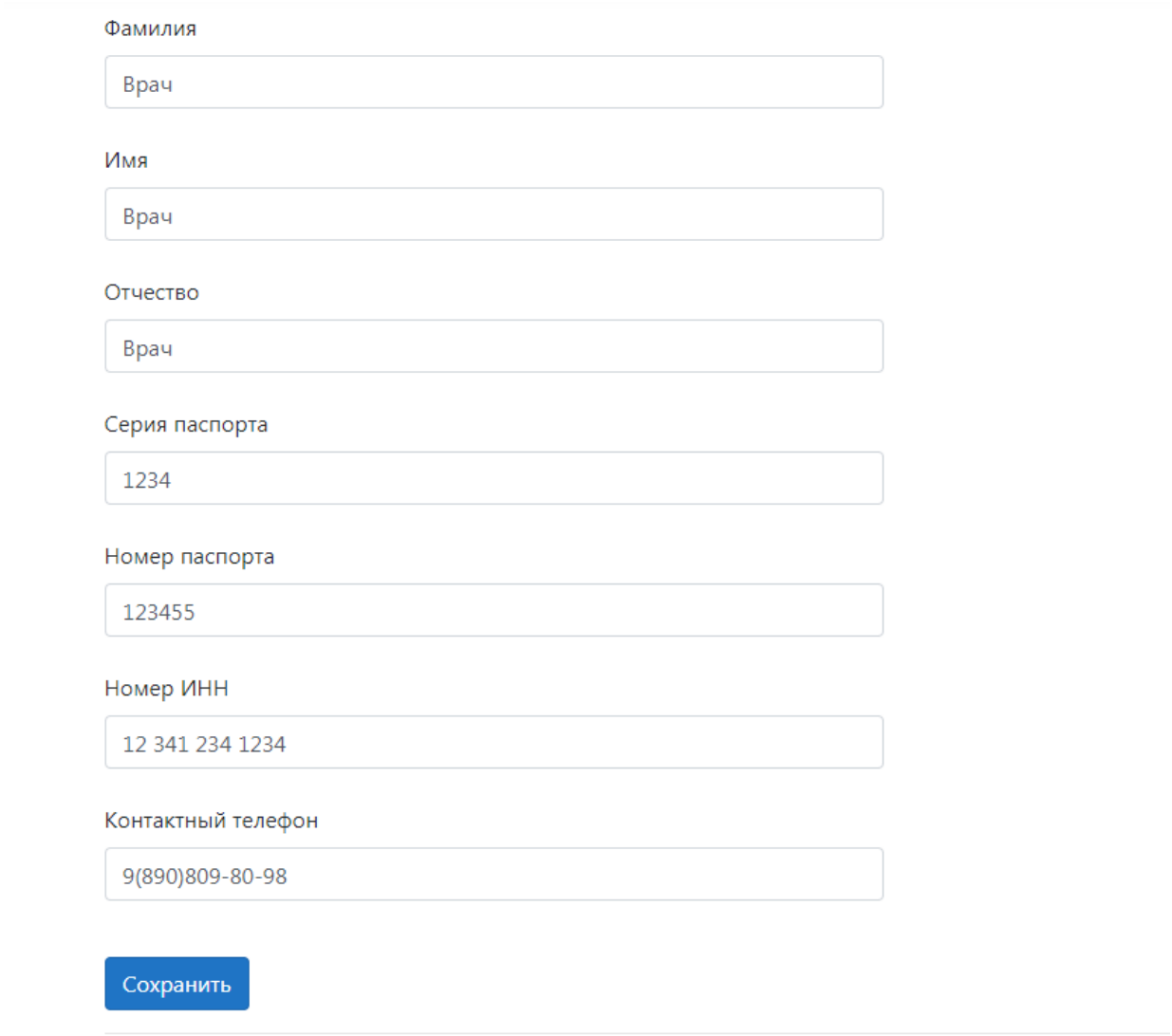


Рисунок 27 — Личный кабинет сотрудника

Помимо прочего, в личном кабинете предусмотрена возможность смены пароля. Форма смены пароля представлена на рисунке. При смене пароля пользователю необходимо дважды ввести новый пароль, для исключения ошибок при вводе пароля, а также ввести старый пароль, чтобы исключить возможность смены пароля кем-то посторонним. Для повышения безопасности системы в базе

данных хранится не пароль в чистом виде, а его хэш. Для получения хэш формируется путем применения алгоритма SHA512. Размер хэша, получаемого при применении данного алгоритма составляет 512 бит, что обеспечивает достаточно высокий уровень надежности. Пример приведен на рисунке 28.

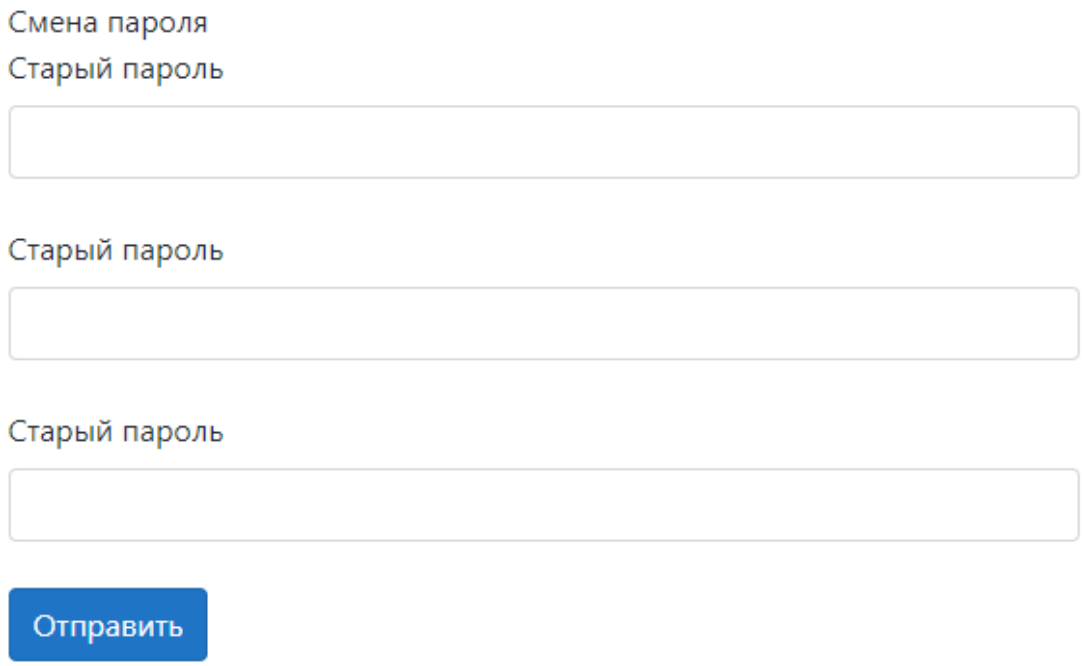


Рисунок 28 — Смена пароля

Для работы с учетными записями в системе предусмотрена роль администратора. Администратор имеет возможность регистрации пациента, врача, сотрудника регистратуры, редактирования данных пользователей и осуществляет восстановление доступа к учетной записи пользователя. Редактирование данных необходимо при возникновении ситуации, когда пользователь по тем или иным причинам не может изменить данные, необходимые для работы с ним. Восстановление доступа к учетной

записи подразумевает установку определенного пароля от учетной записи, при этом, для исключения ситуаций, когда новый пароль так же утрачивается, доступна возможность сброса пароля к стандартному значению, аналогичному тому, которое задается при регистрации пользователя. Форма редактирования данных пользователя администратором представлена на рисунке 29.

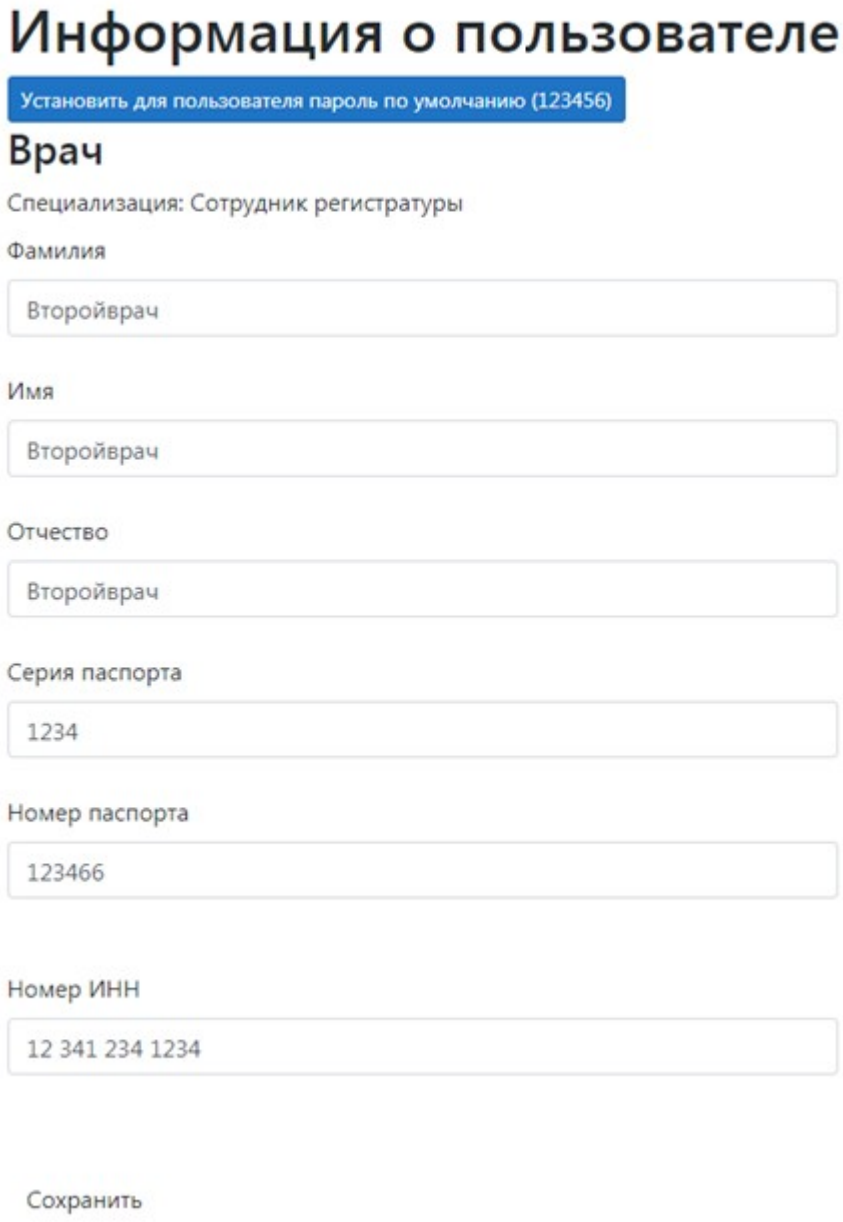


Рисунок 29 — Редактирование данных пользователя

Регистрация врачей доступна только администратору, это сделано для исключения возможности регистрации в качестве врача человека, не имеющего знаний и навыков, необходимых для осуществления врачебной деятельности. При регистрации администратор указывает данные нового сотрудника и его специализацию. Форма регистрации врача представлена на рисунке 30.

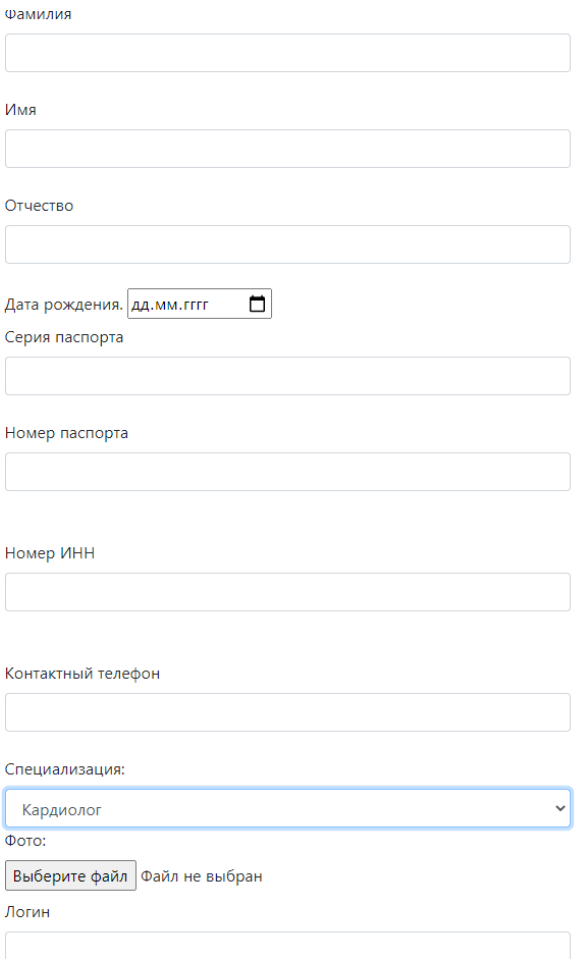


Рисунок 30 — Регистрация врача

Рисунок 30 завершает обзор реализованных в информационной системе форм.

Заключение

В результате проделанной работы был проведен анализ уже существующих медицинских информационных систем. На основе результатов анализа были сформированы требования к проектируемой информационной системе. Была спроектирована архитектура базы данных для будущего приложения, осуществлена формализация требований к функционалу пользователей с помощью диаграмм UML. Выбрано программное обеспечение для реализации приложения с заявленным функционалом, изучены и описаны некоторые технологии, применяемые при создании веб-приложений, приложений, взаимодействующих с базами данных, приложений, осуществляющих работу с текстовыми документами в некоторых форматах.

После выбора программного обеспечения, определения и формализации требований к программному обеспечению были созданы база данных и приложение, реализующее запланированный функционал. Приложение позволяет взаимодействовать нескольким группам пользователей, осуществлять основную часть функционала персонала медицинского учреждения. Помимо этого, приложение взаимодействует с системой «умная банка», что открывает новые возможности в деятельности врачей, их взаимодействии с пациентами.

Список использованных источников

1. Robomed Network [Электронный ресурс] Режим доступа: https://robomed.io (дата обращения: 23.04.2020)
2. The Trusted Provider of Medical Information since 1899 [Электронный ресурс] Режим доступа: https://msdmanuals.com (дата обращения: 24.04.2020)
3. Смертность от болезней системы кровообращения и продолжительность жизни в россии // научная электронная библиотека «киберленинка» [Электронный ресурс] Режим доступа: https://cyberleninka.ru/article/n/smertnost-ot-bolezney-sistemy-krovoobrascheniya-i-prodolzhitelnost-zhizni-v-rossii (дата обращения: 24.04.2020).
4. Корпоративные информационные системы (КИС), комплексная автоматизация бизнеса, внедрение ERP систем – ВитаСофт [Электронный ресурс] Режим доступа: https://vita-soft.ru (дата обращения: 24.04.2020)
5. Медицинские информационные системы - СП.АРМ [Электронный ресурс] Режим доступа: https://sparm.com (дата обращения: 24.04.2020)
6. MedLink Комплексная автоматизация всех видов деятельности медицинских учреждений [Электронный ресурс] Режим доступа: https://medlink.su (дата обращения: 24.04.2020)
7. MedReady Automated Medication Dispensers [Электронный ресурс] Режим доступа: https://medreadyinc.net (дата обращения: 26.04.2020)

8. Функции медицинского документа «История болезни» [Электронный ресурс] Режим доступа: https://cyberleninka.ru (дата обращения: 4.05.2020)
9. Цуканова О. А. Методология и инструментарий моделирования бизнес-процессов: учебное пособие – СПб.: Университет ИТМО, 2015. – 100 с.
10. А. Н. Калашян, Г. Н. Калянов Структурные модели бизнеса: DFD-технологии– М.: Финансы и статистика, 2009. - 256 с.
11. НОУ ИНТУИТ | Лекция | Расширенный анализ требований. Моделирование [Электронный ресурс] Режим доступа: http://intuit.ru. (дата обращения: 20.05.2020)
12. DFD - диаграмма потоков данных [Электронный ресурс] Режим доступа: http://lektsia.com. (дата обращения: 20.05.2020)
13. Крэг Ларман Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку - М.: Вильямс, 2013. - 736 с.
14. EMR Integration Software | Connected Care by NantHealth [Электронный ресурс] Режим доступа: https://nanthealth.com (дата обращения: 26.04.2020)
15. Фиайли К. SQL: Пер. с англ. – М.: ДМК Пресс, 2013. – 456 с.
16. METANIT.COM [Электронный ресурс] Режим доступа: https://metanit.com (дата обращения: 30.04.2020)

17. Техническая документация, материалы по API и примеры кода | Microsoft Docs [Электронный ресурс] Режим доступа: https://docs.microsoft.com (дата обращения: 02.05.2020)
18. ASP.NET Web Forms | .NET [Электронный ресурс] Режим доступа: https://dotnet.microsoft.com (дата обращения: 02.05.2020)
19. Джеффри Рихтер CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# - СПб.: Питер, 2017.- 896 с.
20. Entity Framework Core DbContext [Электронный ресурс] Режим доступа: https://entityframeworktutorial.net (дата обращения: 04.05.2020)
21. Меломед Эдвард, Степаненко Виталий П., Щербинин Владислав А. Microsoft SQL Server 2005 Analysis Services. OLAP и многомерный анализ данных - СПб.: БХВ-Петербург, 2007.- 928 с.
22. Нормализация баз данных, 1, 2 и 3 нормальная форма [Электронный ресурс] Режим доступа: https://office-menu.ru (дата обращения: 05.05.2020)
23. Char и varchar Интерактивный учебник по SQL [Электронный ресурс] Режим доступа: https://sql-tutorial.ru (дата обращения: 06.05.2020)
24. Фримен Адам ASP.NET MVC 5 с примерами на C# 5.0 для профессионалов - М.: Вильямс, 2018. - 736 с.
25. MVC: что это такое [Электронный ресурс] Режим доступа: https://tproger.ru (дата обращения: 06.05.2020)

26. MVC — модель-представление-контроллер [Электронный ресурс] Режим доступа: https://web-creator.ru (дата обращения: 10.05.2020)
27. Что такое MVC и как его использовать [Электронный ресурс] Режим доступа: https://skillbox.ru (дата обращения: 12.05.2020)
28. Клиент-серверные приложения [Электронный ресурс] Режим доступа: https://geekbrains.ru (дата обращения: 13.05.2020)
29. Торопова О.А. Технологии разработки web-приложений: учеб. пособие / О.А. Торопова, Е.В. Кушникова, Ю.М. Урасова, под общ. ред. О.А. Тороповой. Саратов: Сарат. гос. техн. ун-т, 2016. 356 с.
30. Мастер-страницы - Проектирование и разработка web-приложений [Электронный ресурс] Режим доступа: https://studme.org (дата обращения: 16.05.2020)
31. Бен-Ган, И. Microsoft® SQL Server® 2012. Создание запросов. Учебный курс Microsoft: Пер. с англ. / И. Бен-Ган, Д. Сарка, Р. Талмейдж. — М.: Издательство «Русская редакция», 2014. — 720 с.
32. Поль Дюбуа MySQL - М.: Вильямс, 2007. - 1168 с.
33. Entity Framework 6 и коллекции [Электронный ресурс] Режим доступа: https://entityframework.net (дата обращения: 19.05.2020)
34. Запросы к базе данных | Entity Framework | Brainoteka [Электронный ресурс] Режим доступа: https://brainoteka.com (дата обращения: 20.05.2020)

35. Даниэль, Арсеновски Рефакторинг в C# и ASP.NET для профессионалов / Арсеновски Даниэль. - М.: Диалектика, 2018. - 265 с.
36. Гэри, Маклин Холл Адаптивный код на C#. Проектирование классов и интерфейсов, шаблоны и принципы SOLID / Гэри Маклин Холл. - М.: Вильямс, 2015. - 432 с.
37. Зелковиц, М. Принципы разработки программного обеспечения / М. Зелковиц, А. Шоу, Дж. Гэннон. - М.: Мир, 2018. - 364 с.
38. GemBox.Spreadsheet [Электронный ресурс] Режим доступа: https://syssoft.ru (дата обращения: 25.05.2020)
39. Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5. - М.: Вильямс, 2015. - 1312 с.

ПРИЛОЖЕНИЕ А

Исходный код программного обеспечения веб-приложения

Листинг Б.1 – Исходный код методов контроллера HomeController, которые отвечают за создание истории болезни.

```
[HttpGet]
        public async Task<IActionResult> DocAddCourse(int patId)
    {
        int.TryParse(User.Identity.Name, out int a);
         User user = await db.Users.FirstOrDefaultAsync(u => u.id == a);
          User pat = await db.Users.FirstOrDefaultAsync(p => p.id == patId);
                        DocAddCourseVM dACVM = new DocAddCourseVM(user,pat,db);
        return View(dACVM);
    }
[HttpPost]
                        public async Task<IActionResult> DocAddCourse(DocAddCourseVM dACVM){
    if (ModelState.IsValid){
        try{
          dACVM.save();
```

```
            return RedirectToAction("ToMain");
        }catch((System.IO.IOException e){
                                return RedirectToAction("Error",e);
                        }
            }
                    ViewBag.Message = "Форма была заполнена неправильно.";
            return View(dACVM);
        }
```

Листинг Б.2 – Исходный код модели DocAddCourseVM.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Mvc.Rendering;
using WebApplication8.Models;

namespace WebApplication8.Models
{
    public class DocAddCourseVM
    {
        public string patName { get; set{
                    if(!string.IsNullOrEmpty(value)){
                        this.patName=value;
```

```
            }else{
                throw new ArgumentException();
            }
        } }
public string patSurName { get; set{
            if(!string.IsNullOrEmpty(value)){
                this.patName=value;
            }else{
                throw new ArgumentException();
            }
        }  }
public string patLastName { get; set{
            if(!string.IsNullOrEmpty(value)){
                this.patName=value;
            }else{
                throw new ArgumentException();
            }
        }  }
public int docId { get; set; }
public int patId { get; set; }
public string diagnosis { get; set{
            if(!string.IsNullOrEmpty(value)){
                this.patName=value;
            }else{
                throw new ArgumentException();
            }
        }  }
```

```
public DateTime endDate { get; set; }
public DateTime startDate { get; set; }
public int courseLength { get; set; }
public string comment { get; set{
            if(!string.IsNullOrEmpty(value)){
                  this.patName=value;
            }else{
                  throw new ArgumentException();
            }
      }  }
public string simptoms { get; set{
            if(!string.IsNullOrEmpty(value)){
                  this.patName=value;
            }else{
                  throw new ArgumentException();
            }
      }  }
public string cardNum { get; set; }
public string oms { get; set; }
public int sex { get; set; }
public string dateOfBirth { get; set; }
public string age { get; set; }
public List<AddJarVM> jars { get; set; }
public List<AddJarVM> procs { get; set; }
public SelectList shabloons { get; set; }
public string anamnesis { get; set{
            if(!string.IsNullOrEmpty(value)){
```

```
                this.patName=value;
            }else{
                throw new ArgumentException();
            }
        }  }
public string badHabbits { get; set{
            if(!string.IsNullOrEmpty(value)){
                this.patName=value;
            }else{
                throw new ArgumentException();
            }
        }  }
public string curDesiaseHistory { get; set{
            if(!string.IsNullOrEmpty(value)){
                this.patName=value;
            }else{
                throw new ArgumentException();
            }
        }  }
public class AddJarVM
{
    public int jarId { get; set; }
    public int courseid { get; set; }
    public int frequency { get; set; }
    public DateTime addDate { get; set; }
    public int drugReceiveLength { get; set; }
    public string comment { get; set; }
```

```
        [RegularExpression("^[А-Я]+[а-я]*$", ErrorMessage =
"Некорректное название")]
        public string drugName { get; set; }
        public SelectList jars { get; set; }
    }


        public DocAddCourseVM(User user, User pat, DBCon
db){
            this.patId = patId;
        this.docId = user.id;
        this.patLastName = pat.lastName;
        this.patName = pat.name;
        this.patSurName = pat.surName;
                                                this.dateOfBirth =
pat.dateOfBirth.ToString("dd/MM/yyyy");
        this.age = DateTime.Now.Year - pat.dateOfBirth.Year +
"";
        this.oms = pat.oms;
        this.cardNum = pat.cardNum;
        this.sex = pat.sex;
        this.startDate = DateTime.Now.AddDays(1);
        List<Shabloon> shab = await db.Shabloons.Where(s
=> s.docId == user.id).ToListAsync();
        shab.Insert(0, new Shabloon { id = 0, name = "Нет" });
        this.shabloons = new SelectList(shab, "id", "name");


        }


        public void  save(){
```

```
                Course cou = new Course();
            cou.patId = this.patId;
            cou.docId = this.docId;
                                                    cou.endDate    =
this.startDate.AddDays(this.courseLength);
            cou.startDate = this.startDate;
            cou.comment = this.comment;
            cou.diagnosis1 = this.diagnosis;
            cou.simptoms = this.simptoms;
            cou.cardNum = this.cardNum;
            cou.anamnesis = this.anamnesis;
            cou.curDesiaseHistory = this.curDesiaseHistory;
            cou.badHabbits = this.badHabbits;
            db.Courses.Add(cou);
            await db.SaveChangesAsync();
            foreach (var jar in this.jars)
            {
               jar.courseid = cou.id;
            }
            foreach (var proc in this.procs)
            {
               proc.courseid = cou.id;
            }
            foreach (var proc in this.procs)
            {
               if (proc.drugReceiveLength > 0)
               {
                  JarClaim jc = new JarClaim();
```

```
                jc.addDate = cou.startDate;
                jc.courseid = proc.courseid;
                jc.frequency = proc.frequency;
                jc.drugName = proc.drugName;
                jc.comment = proc.comment;
                db.Claims.Add(jc);
                await db.SaveChangesAsync();
            }
        }
        foreach (var jar in this.jars)
        {
            if (jar.drugReceiveLength > 0)
            {
                JarClaim jc = new JarClaim();
                jc.addDate = cou.startDate;
                jc.courseid = jar.courseid;
                jc.frequency = jar.frequency;
                jc.jarId = jar.jarId;
                jc.drugName = jar.drugName;
                jc.comment = jar.comment;
                db.Claims.Add(jc);
                await db.SaveChangesAsync();
            }
        }
    }
}
```

Листинг Б.3 – Исходный код функции в представлении, осуществляющей AJAX- запрос

```
<script type="text/javascript">
        $(function () {
            $('#innNum').change(function ()
            {
                var s = $("#innNum").val();
                //alert(s);
                $.ajax({
                    type: 'GET',
                    url: '@Url.Action("innAvailible")/?s=' + s,
                    success: function (data) {
                        $('#inninInUse').replaceWith(data);
                    }
                });
            }
          );
        }
          )</script>
```

Листинг Б.4 – Исходный код модели CourseReportVM

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using WebApplication8.Models;

namespace WebApplication8.ViewModels
```

```
{
   public class CourseReportVM
   {
      public User doc;
      public User pat;
      public Course cou;
      public Diagnosis diag;
      public Specialisation sp;
      public List<JarUsage> allJars;
      public List<JarClaim> allProcs;
      public string prof { get; set; }
      public string workPlace { get; set; }


            public CourseReportVM(int courseId,DBCon db){
                  Course cou = await
db.Courses.FirstOrDefaultAsync(c => c.id == courseId);
           User doc = await db.Users.FirstOrDefaultAsync(u =>
u.id == cou.docId);
           User pat = await db.Users.FirstOrDefaultAsync(u =>
u.id == cou.patId);
          List<JarClaim> claims = await db.Claims.Where(c =>
c.courseid == courseId).ToListAsync();
                                    Specialisation sp = await
db.Specialisations.FirstOrDefaultAsync(s => s.id ==
doc.specialisation);


        CourseReportVM crvm = new CourseReportVM();
        crvm.doc = doc;
```

```
crvm.pat = pat;
crvm.cou = cou;
crvm.sp = sp;
crvm.prof = cou.prof;
crvm.workPlace = cou.workPlace;
List<JarUsage> jarsUsage = new List<JarUsage>();
foreach (var drug in claims) {
   if (drug.jarId != 0)
   {
      JarUsage ju = new JarUsage();
                  IQueryable<DrugReceive> receives = db.DrugReceives.Where(c => c.jarId == drug.jarId && c.time > drug.addDate && c.time < drug.addDate.AddDays(drug.drugReceiveLength) && c.status == 2);
      ju.realNumber = receives.Count();
      ju.jarId = drug.jarId;
          ju.plannedNumber = db.DrugReceives.Where(c => c.jarId == drug.jarId && c.time > drug.addDate && c.time < drug.addDate.AddDays(drug.drugReceiveLength)).Count();

      if (ju.plannedNumber > 0)
      {
            ju.percentage = Math.Floor(ju.realNumber / ju.plannedNumber * 100);
      }
      ju.length = drug.drugReceiveLength;
      ju.addDate = drug.addDate;
      ju.freq = drug.frequency;
      ju.name = drug.drugName;
```

```
                jarsUsage.Add(ju);
            }
        }
        List<JarClaim> procs = new List<JarClaim>();
        foreach (var drug in claims)
        {
            if (drug.jarId == 0) {
                procs.Add(drug);
            }
        }
        crvm.allJars = jarsUsage;
        crvm.allProcs = procs;

            }
    }
}
```

Листинг Б.5 – Исходный код контроллера AccountController

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using WebApplication8.Models;
using WebApplication8.ViewModels;
using Microsoft.EntityFrameworkCore;
```

```
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using System.Security.Claims;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using System.Text;
using System.Security.Cryptography;

namespace WebApplication8.Controllers
{
    public class AccountController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        private DBCon db;
        public AccountController(DBCon context)
        {

            db = context;
        }



        [HttpGet]
        public IActionResult Login()
        {
            return View();
```

```
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
            public async Task<IActionResult> Login(LoginModel
model)
        {
           if (ModelState.IsValid)
           {
              string hash;
                                                    var    data    =
Encoding.UTF8.GetBytes(model.password1);
              using (SHA512 shaM = new SHA512Managed())
              {
                                                          hash    =
Convert.ToBase64String(shaM.ComputeHash(data));
              }
                 User user = await db.Users.FirstOrDefaultAsync(u
=>              u.login.Equals(model.login1)              &&
u.password.Equals(hash));
              if (user != null)
              {
                 await Authenticate(user); // аутентификация
                 return RedirectToAction("Index", "Home");


              }
                    ModelState.AddModelError("", "Некорректные
логин и(или) пароль");
           }
            ViewBag.Message = "Пользователь с таким логином
и паролем не существует";
```

```
            return View(model);
        }
        private async Task Authenticate(User user)
        {
            // создаем один claim
            var claims = new List<Claim>
            {
                    new Claim(ClaimsIdentity.DefaultNameClaimType,
user.id.ToString()),
                     new Claim(ClaimsIdentity.DefaultRoleClaimType,
user.role.ToString())

            };

ClaimsIdentity id = new ClaimsIdentity(claims,
"ApplicationCookie",
ClaimsIdentity.DefaultNameClaimType,
ClaimsIdentity.DefaultRoleClaimType);
                                                                    await
HttpContext.SignInAsync(CookieAuthenticationDefaults.Authe
nticationScheme, new ClaimsPrincipal(id));
        }
        public async Task<IActionResult> Logout()
        {
                                                                    await
HttpContext.SignOutAsync(CookieAuthenticationDefaults.Auth
enticationScheme);
            return RedirectToAction("Login", "Account");
        }
```

}}

Листинг Б.6 – Исходный код методов контроллера HomeController, которые отвечают за сохранение данных, получаемых от устройств «Умная банка», в базе данных.

```
public bool AddRecordToDB(int id, int status,DateTime data,DateTime time) {
        DrugReceive dr = new DrugReceive();
        dr.jarId = id;
                                                dr.time = new DateTime(data.Year,data.Month,data.Day,time.Hour,time.Minute,time.Second);
        dr.status = status;
        db.DrugReceives.Add(dr);
        db.SaveChanges();
        return true;        }
```