МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка автоматизированной системы приёма пациентов в медицинских центрах

Специальность: 5В070400 – «Вычислительная техника и программное обеспечение»

Выполнил: Дресвянский А.С. Группа: ВТ-15-2
Научный руководитель: д.т.н. профессор Казиев Г.З.

Консультанты:
по экономической части: к.э.н., профессор _____ Ж.Г. Аренбаева
«13» мая 2019 г.

по безопасности
жизнедеятельности: д.т.н., ст. преп. _____ Ш.Ш. Бекбасаров
«14» мая 2019 г.

по применению
вычислительной техники: ст. преп. _____ М.Н. Майкотов
«14» мая 2019 г.

Нормоконтролер: ст. преп. _____ А.А. Айтказина
«15» мая 2019 г.

Рецензент: зав. каф., PhD, доцент _____ Ж.С. Есенгалиева
«____» _____ 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Дресвянскому Алексею Сергеевичу

Тема проекта: Разработка автоматизированной системы приёма пациентов в медицинских центрах.

Утверждена приказом по университету № 124 от «26» октября 2018 г.

Срок сдачи законченного проекта «21» мая 2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Руководство системы менеджмента качества на предприятии; международные стандарты ИСО-9001, данные преддипломной практики.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

- аналитическая часть;
- проектная часть;
- экспериментальная часть;
- экономическая часть;
- безопасность жизнедеятельности;
- приложение А. Техническое задание;
- приложение Б. Листинг программы;
- приложение В. Акт внедрения.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 8 таблиц, 46 иллюстраций.

Основная рекомендуемая литература:

1 Чебыкин, Ростислав Самоучитель HTML и CSS. Современные технологии / Ростислав Чебыкин. - Москва: Огни, 2012. - 624 с.

2 Дронов, Владимир HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / Владимир Дронов. - М.: БХВ-Петербург, 2013. - 416 с.

3 Чаффер, Джонатан Изучаем jQuery 1.3. Эффективная веб-разработка на JavaScript / Джонатан Чаффер , Карл Шведберг. - М.:Символ-плюс, 2010, 448 с.

4 Машнин, Тимур Web-сервисы Java / Тимур Машнин. - М.: БХВ-Петербург, 2011. - 819 с.

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	04.03.2019 – 13.05.2019	
Безопасность жизнедеятельности	Бекбасаров Ш.Ш.	12.03.2019 – 14.05.2019	
Программное обеспечение	Майкотов М.Н.	08.04 – 14.05.2019	
Нормоконтролер	Айтказина А.А.	02.04 – 505.2019	

ГРАФИК
подготовки дипломной проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть	05.11.2018 – 21.12.2018	
Проектная часть	07.01.2019. – 31.01.2018	
Экспериментальная часть	04.02.2019 – 12.04.2019	

Дата выдачи задания «29» октября 2018 г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель проекта _____ Казиев Г.З.

Задание принял к
исполнению студент _____ А.С. Дресвянский

Аңдатпа

Дипломдық жобаның тақырыбы: "Медицина орталығында пациенттерді қабылдауды автоматтандыру жүйесін әзірлеу".

Дипломдық жобаның мақсаты – ауруханаңың жұмыс процесін талдау және пайдаланушылар мен дәрігерлерге өзара іс-қимыл жасауға, қабылдауға жазылуға және қашықтықтан үзінді алуға көмектесетін бағдарламалық қамтамасыз етуді жасау.

Қойылған мақсатқа жету үшін технологиялар қолданылды:

- клиент интерфейсін әзірлеу үшін: HTML, CSS, JavaScript, Angular;
- деректер базасын жобалау үшін: MySQL;

Қорытынды зерттеу мен әзірлемелердің нәтижелерін баяндайды:

- клиент-серверлік веб-қосымшаларды әзірлеудің заманауи құралдары зерделенді және қойылған мақсатқа сәйкес әзірлеу орындалды;
- сайт толық өнімге дейін жақсартудың одан әрі стратегиясы әзірленді.

Аннотация

Тема дипломного проекта: «Разработка автоматизированной системы приёма пациентов в медицинских центрах».

Цель дипломного проекта – проанализировать рабочий процесс больницы и создать программное обеспечение, которое поможет пользователям и врачам взаимодействовать, записываться на прием и получать выписки дистанционно.

Для достижения поставленной цели использовались технологии:

– для разработки клиентского интерфейса: HTML, CSS, JavaScript, Angular;

– для проектирования базы данных: MySQL.

Заключение освещает результаты исследования и разработки/’’’;p:

– изучены современные средства разработки клиент-серверных веб-приложений и выполнена разработка согласно поставленной цели;

– разработана дальнейшая стратегия улучшения сайта до полноценного продукта.

Annotation

Theme of the graduation project: "Development of an automated patient receptiom system in medical centers."

The aim of the graduation project is to analyze the working process of the hospital and create software that will help users and doctors to interact, make an appointment and receive statements remotely.

To achieve this goal, we used technology:

- for client interface development: HTML, CSS, JavaScript, Angular;
- for database design: MySQL.

The conclusion covers the results of research and development:

- modern tools for developing client-server web applications were studied and development was carried out according to the goal;
- developed further strategy of improving the site to a full product.

Содержание

	Введение	8
1	Описание программного продукта	9
1.1	Назначение программного продукта	10
1.2	Потенциальные пользователи программного продукта	11
1.3	Обзор существующих аналогичных программных продуктов	11
1.4	Выбор средств и технологий	13
1.5	Постановка цели и задач	22
2	Проектирование программного продукта	24
2.1	Структура программного обеспечения	24
2.2	Функциональная структура веб-приложения	25
2.3	Проектирование базы данных	26
3	Разработка программного продукта	36
4	Экономическая часть	51
4.1	Трудоемкость разработки ПП	51
4.2	Расчет затрат на разработку ПП	51
4.3	Определение возможной (договорной) цены ПП	57
5	Охрана труда и безопасность жизнедеятельности	58
5.1	Расчет естественного освещения	58
5.2	Расчет искусственного освещения	61
	Заключение	67
	Список литературы	68
	Приложение А. Техническое задание	69
	Приложение Б. Листинг программы	71
	Приложение В. Акты внедрения	80

Введение

В современном мире все чаще становится актуальной проблема записи к врачу и выдачи справки.

На сегодняшний день людям часто приходиться ходить в больницу по несколько раз, дабы узнать расписание врача, записаться к нему на приём или отстоять живую очередь.

Поэтому на сегодняшний день появляется достаточно много медицинских предприятий, которые задумываются о своем сайте или приложении, позволяющем сэкономить время как пациента, так и самого лечащего врача.

Когда есть сайт, на котором можно не только посмотреть график работы врача, но и то, когда он свободен и даже записаться к нему, а затем, после приема, получить выписку о необходимом лекарстве в электронном виде, то и пациенту, и самому врачу становится гораздо легче не только в области тайм-менеджмента, но и в заполнении отчетностей, потому что все данные содержатся в электронном формате на сайте.

Актуальность выбранной темы заключается в том, чтобы создать новый сайт, функционал которого будет в корне отличаться от обычных рекламных лендингов. Он будет содержать информацию о больнице, пользователь сможет зарегестрироваться как пациент и как врач, записываться к нужному доктору и смотреть расписание своих записей, получать выписки о необходимых ему лекарствах. Врач в свою очередь сможет смотреть когда и кто к нему записан и выписывать необходимые пациенту лекарства.

Такой сайт может быть применён в небольших частных медицинских предприятиях и послужит подспорьем в создании автоматизированной системы, которая будет экономить драгоценное время и ресурсы пациентов и врачей.

Основными задачами, которые должны быть решены в дипломном проекте, являются:

- разработка дизайна сайта;
- создание макета сайта с помощью HTML и CSS;
- создание работающих форм и кнопок с помощью Java Script;
- создание базы данных, подвязанной к сайту и содержащей информаию о пациентах и врачах.

В результате разработки должен получиться сайт с понятным и приятным интерфейсом, чтоб даже пользователи, далёкие от технологий могли без проблем записаться к необходимому врачу и получить информацию о нужных им лекарствах.

1 Описание программного продукта

Перед началом создания веб-страниц необходимо ознакомиться с основными понятиями, терминами и вспомогательными инструментами.

Сайт представляет собой набор HTML-страниц на веб-сервере.

Веб-сервер-это компьютер, который постоянно подключен к Интернету, который передает эти страницы по запросу.

Сайт-это гораздо больше, чем просто набор веб-страниц. Это система гиперссылок, которая соединяет узел в единое целое, и единое стилистическое оформление узла и своевременное обновление информации на узле и скорость загрузки отдельных страниц и многое другое.

Домашняя страница-это веб-страница, которая передается сервером по запросу пользователя, указавшего в запросе имя веб-страницы, но не указано конкретное название страницы. Файл, содержащий домашнюю веб-страницу, получает индекс стандартного имени.htm или по умолчанию.htm. Домашняя страница является лицом сети, поэтому ее дизайну и макету придается особое значение, так как ее основная функция-предоставить пользователю удобные способы перехода на другие веб-страницы и документы.

Сайт, представляющий организацию, называется корпоративным или официальным сайтом организации. Официальный сайт имеет более жесткие требования с точки зрения контента, графического дизайна, навигации, хостинга.

Официальный сайт обычно имеет следующие разделы:

- информация об учереждении;
- направления деятельности;
- структура учреждения;
- персонал;
- регистрация;
- контакты.

Создание сайта - это событие, которое повышает имидж любого предприятия. Хороший сайт, вбирая всю полезную информацию, является лучшей визитной карточкой коммерческой фирмы. Конечно, это современно и престижно.

Это прекрасная возможность продемонстрировать свои достижения всем, разместить соответствующую информацию для заинтересованных сторон. Это способ рассказать о своих успехах, поблагодарить спонсоров.

Для медицинского предприятия, сайт, прежде всего, ресурс, помогающий наладить коммуникацию врач - пациент. Основные характеристики медицинских сайтов:

Содержание медицинского веб-сайта - единство всех основных элементов (текстовых и графических) информации, существующей и выраженной в виде веб-сайтов, а также единство связей этих основных элементов;

Медицинский веб-дизайн-это процесс выбора и организации графических компонентов с целью достижения определенной цели, которая может быть либо эстетической, либо иметь функциональную подоплеку, а зачастую преследовать обе эти цели;

Техническая реализация медицинского сайта - подбор компонентов, интеграция технологий, программного и аппаратного обеспечения для передачи образовательной информации пользователю;

Результативность медицинских сайтов - характеристики, информирующие об использовании медицинских сайтов пользователями.

1.1 Назначение программного продукта

Структура сайтов медицинских предприятий довольно однообразна и обычно включает в себя следующие разделы:

Главная страница. Дает пользователю представление о структуре сайта. Кроме того, на главной странице представлены новости, анонсы событий, то есть информация, которая может быть интересна для разных категорий пользователей (в нашем случае это пациенты и врачи).

Пациентам (учреждения и сам сайт) может быть отдельной страницей или в сочетании с другим разделом. Пост новости раздел только тогда, когда это действительно необходимо и доверие к его будущим обновлениям. В то же время устаревшая информация не всегда бесполезна, они должны просто находиться под другим знаком, например, "архив".

О больнице. Информация о самом медицинском предприятии.

Контакты. Может быть почтовый адрес, телефонные номера, электронную почту или форму отправки сообщения в администрацию или службу поддержки сайта.

Наши врачи (Информация, Сертификаты, Сертификация). Сайт также может предоставить возможность врачам публиковать свои материалы (вплоть до отдельного раздела по этому вопросу).

Что касается форума, то следует отметить, что только активно обновляемый "живой" сайт пользуется авторитетом и поддержкой пользователей. Если разработка сайта не предусмотрена, то запускать гостевые книги и другие интерактивные формы общения бессмысленно. Результат будет отрицательным, то есть в гостевой книге будут накапливаться различные отзывы, не придающие значения интернет-ресурсу.

1.2 Потенциальные пользователи программного продукта

Потенциальные пользователи этого продукта будут сотрудники небольших компаний и команд (например, стартапов), поскольку для таких компаний на стадии планирования своего рабочего времени это очень важно. Во-первых, это то, что людей меньше, чем задач, возложенных на них. И самое главное, что все эти задачи должны быть выполнены вовремя, и ни одна из задач не может быть забыта или пропущена.

Все сотрудники компании, независимо от их должности в компании, будут использовать этот программный продукт. Это связано с тем, что каждый сотрудник, несмотря на занимаемую должность, должен принести пользу своим товарищам по команде выполненным задачам. Каждая крупная компания, которая коммерчески успешна и узнаваема, начала с небольшой команды, в которой каждый из них отвечал за задачи.

Еще одна важная идея проекта заключается в том, что в дополнение к упрощению управления временем сотрудников этот сайт также будет более безопасным, чем аналогичные онлайн-сервисы. Никто в Интернете не может получить пароль сотрудника и прочитать его деловую переписку с коллегами. И это повлияет на уникальность развития компании, так как идея будет надежно защищена.

Единственный раз, когда утечка данных может произойти непосредственно от сотрудников компании. К сожалению, решить этот фактор программно невозможно. Все зависит от качества подбора сотрудников в компании и их мотивации. Например, сотрудники банковской системы платят большую зарплату, поэтому у них нет желания списывать деньги с банковского счета.

1.3 Обзор существующих аналогичных программных продуктов

Сегодня каждый день разрабатывается множество продуктов, каждый из которых является уникальным. Все продукты, которые делают определенную нишу в общих функциональности, имеют как аналогичные компоненты, так и что-то особенное.

Таким образом, некоторые продукты могут иметь очень большую аудиторию пользователей, и другие продукты, несмотря на их сходство, они могут оставаться неудачной попыткой решения проблемы.

При проектировании каждой системы в первую очередь должен быть пользователь, который в конечном итоге использует программное обеспечение.

Затем будет рассмотрен список продуктов, аналогичные функциям, которые выполняются с разработчиками. Что касается разработанного приложения, это будет означать плюсы и минусы подобного программного обеспечения.

Такой анализ будет искать сильные стороны на данном этапе приложение, и это поможет вам узнать, какие функции они также популярны и могут быть добавлены для использования конечным пользователем в процессе развития.

1.3.1 Изучение похожих продуктов

В современном мире каждый день разрабатывается множество сайтов, каждый из которых уникален по-своему. Все программные продукты в чём-то схожи, но и по-своему уникальны в своих целях и функционале.

Таким образом, некоторые программные продукты могут иметь огромную благодарную аудиторию пользователей, и, несмотря на их сходство, другие продукты могут продолжать быть неудачной попыткой решить задачи.

В процессе проектирования любой системы, приоритет должен быть пользователем, который в конечном итоге будет использовать программный продукт.

Затем мы рассмотрим список аналогичных программных продуктов, согласно разработчикам. Плюсы и минусы аналогичного программного обеспечения в отношении разработанного приложения будут замечены.

Этот анализ определит сильные стороны приложения на текущем этапе приложения, а также поможет вам узнать, какие функции популярны и могут быть добавлены для использования конечным пользователем в процессе разработки.

1.3.2 Damumed

“Damumed” – программа, в которой можно записываться к врачу, вызывать его на дом, есть возможность записи себя или членов семьи на прием к участковому врачу или узкому специалисту, указав повод посещения и выбрав удобное время из предложенных в приложении графиков врачей больницы.

Имеется функция просмотра своих записей, также можно посмотреть информацию о совершенных вызовах на дом.

По моему мнению, на сегодняшний день это самое актуальное и современное приложение как минимум для нашей страны.

Но у него есть и свои недостатки: как для пользователя, это приложение является сложным с точки зрения освоения и не типичности интерфейса.

Порой тяжело вызвать врача, когда это необходимо, приложение часто выдает ошибки и вылетает, но, в любом случае, приложение весьма актуальное, и, несмотря на небольшие недостатки, – это программное обеспечение является одним из лучших решений.

1.3.3 OtherSide

Также неплохое программное решение, при необходимости, также имеется функция вызова врача. Приложение работает почти без сбоев и хорошо себя показывает на деле.

Программное обеспечение также содержит перечень неотложных действий в ожидании медицинской помощи, номера телефонов экстренных служб, а также перечень содержимого аптечки первой помощи.

В нём достаточно легко разобраться, иногда возникают проблемы с регистрацией, но в условиях того, что мы имеем, этим можно пренебречь.

Есть ещё несколько минусов среди которых самым значительным является устаревший интерфейс, который предлагают нам разработчики.

1.3.4 Doq.kz

Сайт с неплохим современным дизайном и хорошим функционалом. Иногда возникают подвисания и регистрация часто не выдаёт очевидные ошибки при вбивании электронной почты.

Сайт достаточно тяжёлый, а страницы подгружаются долго. Но из плюсов – этот сайт сумел собрать в себе всех врачей города, которые работают на выезд.

Таким образом, можно сделать вывод, что у каждого из вышеперечисленных программных обеспечений имеются свои баги и недостатки.

В сегодняшних реалиях медицина – одна из самых важных сфер социума, которая должна работать оперативно и корректно.

Просмотрев баги похожих программных обеспечений, можно сделать выводы и не допустить тех же ошибок.

1.4 Выбор средств и технологий

Данный подраздел повествует о том, какие технологии были использованы при разработке дипломного проекта, их основные достоинства и недостатки. Также поясняется, почему были выбраны именно эти технологии, и за какую часть в приложении они несут ответственность.

1.4.1 Sublime text

Sublime Text - легкий текстовый редактор для программистов.

Мне лично понравилась собственная визуализация кода справа и, конечно же, возможность редактировать текст с помощью некоторых курсоров. Еще одна из частей является выделение и автозаполнение почти все, хороший

поиск без поиска и grep, автоматическая WordWrap для определенной ширины строки, spellchecker, поддержка различных кодировок и разрывов строк, регулируемая ширина отступа. Вы можете написать Плагины Python (yay, обычный язык, а не Vim!). Кроме того, существует гибкая конфигурация шрифтов и цветовых схем. Говоря о системах, дефолт не является темной темой \M/.

Это не совсем очевидный момент, когда Sublime Text - возможность открывать каталоги вместо файлов. Это делается с помощью файла → Open directory или передачи имени каталога в качестве аргумента при запуске редактора bash. В то же время у вас есть не только очень красивое дерево каталогов слева, но, например, поиск имени файла (Ctr+P) начинает работать намного круче.

1.4.2 HTML

HTML (язык разметки гипертекста) является языком по умолчанию для разметки гипертекста страниц в Интернете. Форматирование гипертекста имеет другие языки, но большинство веб-страниц помечены на языке HTML. Такие сайты успешно интерпретируются браузерами, которые отображают их на экранах различных электронных устройств в удобном для человека формате.

HTML - это язык разметки гипертекста: используйте разделители (маркеры), короткие теги для преобразования текста в гипертекст. Вот пример тега: - этот открывающий тег возвращает вывод текста жирным шрифтом до тех пор, пока закрывающий тег не будет удовлетворен.

HTML состоит из серии коротких кодов, введенных в текстовый файл автором сайта-это теги. Затем текст сохраняется в виде html-файла и просматривается через браузер, например Internet Explorer или Netscape Navigator. Этот браузер читает файл и переводит текст в видимую форму, отображая страницу, как предполагал автор. Написание собственного HTML влечет за собой правильное использование тегов для создания вашего видения. Для создания HTML-страниц можно использовать что угодно - от примитивного текстового редактора до мощного графического редактора.

Если вы создаете сайт с WordPress, вам нужно знать, что веб-сайт создается на языке HTML, но со временем вам нужно хотя бы немного освоить язык HTML.

Вот пример довольно распространенной ситуации: вы пишете текстовую страницу, и вы решили включить часть другого текста, указав эту часть как правильную цитату, чтобы поисковые системы не рассматривали это как плагиат.

Если вы перейдете из визуального режима в текстовый режим, в этом окне HTML-тега вы найдете больше, чем сам текст. Это доказательство того, что текст проходит через многие страницы, пытаясь завершить, укрепить и

"углубить" каждый текст, и в результате появился смертельный жуткий HTML-код.

Вы можете вручную удалить дополнительные теги и лучше визуально скопировать текст в окно и вставить его в текстовое окно - в результате этого простого действия все теги исчезнут и сохранят темноту времени и темноту сочных слов на адрес странных текстовых авторов.

При создании веб-сайтов во всех браузерах рекомендуется соблюдать современные стандарты языка HTML5, которые, скорее всего, обеспечат правильное представление страниц веб-сайта. Рекомендуемый валидатор для проверки соответствия стандартам.

Когда вы создаете веб-сайт с CMS WordPress, невозможно полностью полагаться на Wordpress, поскольку основная загрузка-это HTML в WordPress, но даже последняя версия имеет некоторые недостатки.

Наиболее важным из них является то, что он не может делать правильные таблицы, которые надежно контролируются валидатором. Если вы переместите таблицу из Excel на лист достаточно большой, вы получите страницу, где валидатор обнаруживает сотни ошибок! Подробные сведения о том, как редактировать такие таблицы, см. На странице таблицы.

1.4.3 CSS

CSS-это язык для описания представления веб-страниц, включая цвета, макет и шрифты. Он позволяет адаптировать презентацию к различным типам устройств, таким как экраны большие, маленькие экраны, принтеры. CSS не зависит от HTML и может использоваться с любым языком разметки на основе XML. Разделение HTML и CSS упрощает обслуживание сайтов, совместное использование таблиц стилей на страницах и адаптацию страниц к различным средам. Это называется разделение структуры (или содержание) презентации.

CSS-это язык стиля, который указывает, как отображаются HTML-документы. CSS работает со шрифтами, цветами символов и фоном, полями, строками, шириной и высотой элементов отображения, с обоями, позиционированием элементов и т. д.

Если HTML-код должен упорядочить содержимое страницы, вам нужно отформатировать CSS, чтобы отформатировать этот структурированный контент.

С помощью CSS вы можете легко создавать высококачественные веб-сайты, устанавливая стили отдельных элементов на страницах сайта в специальных файлах css, в будущем убедитесь, что все страницы оформлены в одном стиле.

Наиболее важным из всех файлов стиля является файл style.CSS, в этом файле вы указываете все основные стили элементов веб-сайта.

Пример использования одного файла style.css - это проектирование таблиц: значение ширины ячеек всех таблиц в стиле ширины файла: auto; и

высота ячеек всех таблиц высота: Auto; вы можете быть уверены, что ширина и высота всех ячеек всех таблиц страниц сайта будут автоматически распознаны содержимым ячеек.

Этот язык прошел несколько этапов своего развития, в 2014 году языковая версия 3 является современной, которая постепенно улучшается и уточняется.

1.4.4 JavaScript

HTML-страницы отлично подходят для отображения статического содержимого, например, простого изображения или текста. Однако большинство страниц в настоящее время редко статичны. Многие из сегодняшних страниц имеют меню, формы, слайд-шоу и даже изображения, которые обеспечивают взаимодействие с пользователем. Javascript-это язык, используемый веб-разработчиками для обеспечения такого взаимодействия. Поскольку JavaScript работает с HTML-страницами, разработчик должен знать HTML, чтобы использовать весь потенциал этого языка сценариев. Хотя есть и другие языки, которые можно использовать для написания сценариев в интернете, на практике это по существу все Javascript.

Существует два способа использования JavaScript в HTML-файле. Первый включает в себя встраивание всего кода JavaScript в HTML-код, в то время как второй метод использует отдельный файл JavaScript, который вызывается из элемента скрипта, т. е. заключен в теги скрипта. JavaScript-файлы идентифицируются расширение js. Хотя JavaScript в основном используется для взаимодействия с HTML-объектами, его также можно заставить взаимодействовать с другими не-HTML-объектами, такими как плагины браузера, свойства CSS (Каскадные Таблицы Стилей), текущая дата или сам браузер. Чтобы написать код JavaScript, все, что вам нужно, это базовый текстовый редактор, такой как Notepad в Windows, Gimp в Linux или BBEdit. Некоторые текстовые редакторы, такие как подсветка синтаксиса функции BBEdit для JavaScript. Это позволит вам легко идентифицировать элементы кода JavaScript. Последние версии Internet Explorer, Firefox и Opera поддерживают JavaScript.

JavaScript был создан, чтобы сделать веб-страницу "живой". Программы на этом языке называются скриптами. В браузере вам будет необходимо связаться непосредственно с HTML, и так, что страница загружается, они выполняются немедленно.

Когда язык JavaScript был создан, изначально он носил другое название: "LiveScript". Но тогда язык Java был очень популярен, и маркетологи решили, что похожее название сделает новый язык более популярным.

Он имеет множество функций, которые делают его трудно осваиваемым, но обладая умениям работы с JavaScript, он становится отличным оружием.

Для выполнения программ, не важно на каком языке, существуют два способа "компилировать" и "интерпретировать".

Компиляцией является процесс, когда исходный код программы, используя специальный инструмент, другой программы, которая называется "компилятор", преобразуется в другой язык, как правило, – в машинный код. Это машинный код, который затем распространяется и выполняться. Исходный код программы остается у разработчика.

Интерпретация - это когда исходный код программы, получает еще один инструмент под названием " Переводчик "и выполняет его "как он есть". Исходный код (скрипт) является распределенной. Этот подход применяется в браузерах с помощью JavaScript.

Современные интерпретаторы, перед выполнением преобразуют JavaScript в машинный код, оптимизируют, и только потом запускают. И даже во время выполнения пытаются оптимизировать компиляторы. Таким образом, JavaScript работает очень быстро.

Во всех основных браузерах, интерпретатор JavaScript интегрирован, так что вы можете выполнять скрипты на странице. Но, конечно, что JavaScript не может быть использован только в браузере. Это полноценный язык, в котором программы могут выполняться на сервере, и даже в стиральной машине, если исполнитель, соответствующий установлен.

1.4.5.1 Angular

AngularJS - это платформа JavaScript для создания лендингов для веб-приложений. Поскольку по-прежнему существует некоторая путаница в определении этих слов, стоит уточнить. Прежде всего, как я уже сказал, AngularJS-это фреймворк. Структура подразумевает, что вместо написания собственного кода весь разработчик "предварительно согласован с правилами". Эти правила являются стандартами и инструментами, выбранными из Yii framework.

AngularJS с большим количеством инструментов облегчает жизнь программисту. Конечно, задача разработчика, когда он узнает правила использования. В арсенал инструментов входят: фильтры, политики, двунаправленная привязка данных и многое другое.

AngularJS был первоначально разработан для создания веб-приложений. Таким образом браузер сначала открывает страницу и дополнительное содержимое загружается по мере необходимости. Такой подход экономит трафик и снижает нагрузку на сервер.

Поскольку AngularJS написан и используется в своей работе, JavaScript поэтому, если вы знакомы с этим языком программирования, то вам будет

очень легко ориентироваться и использовать в этом термине. Наконец, все контроллеры, службы, фильтры и элементы. д. - это функции JavaScript. На самом деле, они могут быть использованы с таким же успехом в других проектах, даже не связанных в AngularJS.

1.4.9 MySQL

MySQL, самая популярная система управления базами данных SQL с открытым исходным кодом, разрабатывается, распространяется и поддерживается корпорацией Oracle.

Веб-сайт MySQL (http://www.mysql.com/) предоставляет последнюю информацию о программном обеспечении MySQL.

MySQL-это система управления базами данных.

База данных представляет собой структурированный набор данных. Это может быть что угодно, от простой список покупок для картинной галереи или огромного количества информации в корпоративной сети. Для добавления, доступа и обработки данных, хранящихся в базе данных компьютера, необходима система управления базами данных, такая как MySQL Server. Поскольку компьютеры очень хорошо справляются с большими объемами данных, системы управления базами данных играют центральную роль в вычислениях, как автономные утилиты или как части других приложений.

Базы данных MySQL являются реляционными.

Реляционная база данных хранит данные в отдельных таблицах, а не помещает все данные в одну большую кладовую. Структуры базы данных организованы в физические файлы, оптимизированные для скорости. Логическая модель с такими объектами, как базы данных, таблицы, представления, строки и столбцы, предлагает гибкую среду программирования. Вы устанавливаете правила, регулирующие отношения между различными полями данных, такими как “один к одному”, "один ко многим", "уникальный", "обязательный" или "необязательный" и "указатели" между разными таблицами. База данных применяет эти правила, так что с хорошо разработанной базой данных приложение никогда не видит несогласованных, дубликатов, сирот, устаревших или отсутствующих данных.

SQL-часть "MySQL “означает”структурированный язык запросов". SQL является наиболее распространенным стандартизированным языком, используемым для доступа к базам данных. В зависимости от среды программирования можно ввести SQL напрямую (например, для создания отчетов), внедрить инструкции SQL в код, написанный на другом языке, или использовать API для конкретного языка, скрывающий синтаксис SQL.

SQL определяется стандартом ANSI / ISO SQL. Стандарт SQL развивается с 1986 года и существует несколько версий. В этом руководстве "SQL-92 “относится к стандарту, выпущенному в 1992 году,” SQL:1999

“относится к стандарту, выпущенному в 1999 году, а” SQL:2003" относится к текущей версии стандарта. Мы используем фразу "стандарт SQL" для обозначения текущей версии стандарта SQL в любое время.

MySQL software является открытым исходным кодом.

Открытый исходный код означает, что любой может использовать и изменять программное обеспечение. Любой может загрузить программное обеспечение MySQL из интернета и использовать его, ничего не платя. Если вы хотите, вы можете изучить исходный код и изменить его в соответствии с вашими потребностями. Программное обеспечение MySQL использует GPL (GNU General Public License), http://www.fsf.org/licenses/, чтобы определить, что вы можете и не можете делать с программным обеспечением в разных ситуациях. Если вы чувствуете себя некомфортно с GPL или вам нужно встроить код MySQL в коммерческое приложение, вы можете купить у нас коммерческую лицензионную версию. См. обзор лицензирования MySQL для получения дополнительной информации.

Сервер баз данных MySQL очень быстрый, надежный, масштабируемый и простой в использовании

Если это то, что вы ищете, вы должны дать ему попробовать. MySQL Server может удобно работать на рабочем столе или ноутбуке, наряду с другими приложениями, веб-серверами и т. д., не требуя особого внимания. Если вы посвятите всю машину MySQL, вы можете настроить параметры, чтобы использовать всю память, мощность процессора и доступную емкость ввода-вывода. MySQL также может масштабироваться до кластеров машин, объединенных в сеть.

MySQL Server был первоначально разработан для обработки больших баз данных намного быстрее, чем существующие решения и успешно используется в очень сложных производственных средах в течение нескольких лет. Несмотря на постоянное развитие, MySQL Server сегодня предлагает богатый и полезный набор функций. Его подключение, скорость и безопасность делают MySQL Server очень подходящим для доступа к базам данных в Интернете.

MySQL Server работает в клиент / сервер или встроенные системы.

Программное обеспечение базы данных MySQL-это клиент / серверная система, состоящая из многопоточного SQL server, поддерживающего различные серверные части, несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр интерфейсов прикладного программирования (API).

Мы также предоставляем MySQL Server как встроенную многопоточную библиотеку, которую вы можете связать с вашим приложением, чтобы получить меньший, более быстрый и простой в управлении автономный продукт.

1.4.10 Bootstrap

В общем, чтобы лучше понять, почему требуется Bootstrap, вы можете немного вернуться и ответить на вопрос "Что такое CSS framework?”

На самом деле, проще говоря, это файл или несколько файлов с предопределенным кодом, которые связаны с сайтом в основном разделе, тогда становится возможным использовать возможности этой структуры.

Рамки создаются для облегчения создания веб-сайтов для других веб-разработчиков. Я уже говорил в начале, что сегодня почти каждый разработчик думает о том, как ускорить разработку после создания пары сайтов с нуля.

Дело в том, что если мы будем делать разработку сайта с нуля, то нам придется многое позаботиться. Все стили css, все веб-скрипты должны быть написаны с нуля, но это могут быть сотни или тысячи строк кода. И вы можете сделать много ошибок в макете. Например, ваш шаблон будет выглядеть по-разному в основных браузерах или он не будет адаптивным.

Как правило, это только ради адаптивного макета и должно использоваться Bootstrap, потому что, когда дело доходит до фиксированных макетов, легко сделать их с нуля. Просто создавайте блоки, устанавливайте их и работайте над макетом.

Но в случае адаптивного макета все иногда сложнее. Вы должны убедиться, что ваш сайт хорошо отображается на каждом разрешении экрана. Для этого вам нужно использовать медиа-запросы. Для больших шаблонов таких запросов может понадобиться много, к тому же еще нужно научиться их писать.

В общем, при разработке адаптивного шаблона вам придется потрудиться с нуля, а ваши навыки в макете должны быть достаточно высокими.

Как насчет Bootstrap? Если вы изучите эту структуру, это значительно упростит макет для вас. Во-первых, структура заботится о совместимости между браузерами и адаптивности, и это самые важные вещи, которые разработчик должен заботиться. Но с bootstrap они очень просты в реализации. Он также позволяет создавать шаблон html для человека, который ранее сделал очень мало макета и не особенно знаком с css.

Во-вторых, рамки идеальны, когда вы работаете в команде. Макет на bootstrap с правильной способностью и пониманием в 3-5 раз быстрее, а единообразие кода позволяет любому из ваших коллег вносить изменения. Когда дело доходит до макета без рамки, каждый разработчик может иметь свой собственный стиль, и другой человек должен потратить время на изучение своего кода.

В качестве основных преимуществ я хотел бы также упомянуть очень большое русскоязычное сообщество и хорошую документацию на нашем языке. Из-за этой распространенности Bootstrap существует множество шаблонов, где дизайн всех основных элементов уже был переработан. Вы

можете изменить эти шаблоны и сделать свои веб-сайты, только немного немного.

Недостатки Bootstrap

На самом деле, их только два. Первый - код обычно написан в библиотеке больше, чем если бы вы написали с нуля при разработке. Потому что, если вы это сделаете сами, вы просто реализуете необходимую функциональность, и это все. Bootstrap имеет все на все случаи жизни. Даже то, что вам может не понадобиться. Но опять же, эту проблему очень легко решить, потому что вы можете выбрать, какие компоненты программы должны быть загружены в файл CSS. Например, вы можете просто загрузить сетку и сделать все остальное самостоятельно.

Второй недостаток - бесшовный паттерн. Да, на самом деле я часто захожу на разные сайты и вижу там одни и те же кнопки. И я знаю, что вы сделали в Bootstrap, потому что это слишком очевидно. Но эту проблему легко решить, потому что она существует только в том случае, если вы используете только готовые компоненты фреймворка и никогда не настраиваете что-то для себя.

И если, например, вы просто подключаете сетку Bootstrap, вы можете воссоздать любой дизайн с помощью очень удобной, гибкой сетки.

Компоненты

Bootstrap-это комплексная структура. Это означает, что он имеет много компонентов. На самом деле, все, что вам нужно при разработке стандартных сайтов.

Если вы написали код для некоторых из этих компонентов, вы, вероятно, знаете, что это не делается за 1 минуту. В Bootstrap этого достаточно, чтобы немного узнать о самой структуре, и все эти вещи вы можете использовать очень быстро.

Начать работу с bootstrap достаточно легко. У нас есть основные теги doctype, html, head и body. Тег meta name= "viewport" особенно важен для адаптивного дизайна - он гарантирует, что ваш сайт имеет соотношение 1:1 с viewport (размер экрана).

Нам необходимо просто добавить bootstrap в header сайта и он автоматически сделает наш сайт адаптивным для всех типов устройств, после того, как мы пропишем необходимые команды для соединения bootstrap с нашим проектом.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Bootstrap 101 Template</title>
    <link href="css/bootstrap.min.css" rel="stylesheet" />
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <h1>Hello, world!</h1>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

Рисунок 2.2.1 – Подключение bootstrap

1.5 Постановка цели и задач

Достижение цели не является одномоментным мероприятием. Достижение цели представляет собой решение совокупности задач, связанных между собой по смыслу и направленных для достижения единой цели.

Каждая из задач направлена на изучение или разработку каких-либо вещей в проекте.

Ниже приведены цель дипломного проекта и задачи, которые необходимо реализовать для достижения поставленной цели.

Целью дипломного проекта является разработка программного продукта, который позволит заисываться на прием к врачу, вызывать врача на дом и получать электронные выписки с необходимыми лекарствами.

Основные задачи дипломного проекта:

– изучение предметной области на примере виртуальной IT-компании;

– определение основных требований к программному продукту;

– выбор и обоснование технологий для разработки программного продукта;

– разработка пользовательского веб-интерфейса;

– проектирование базы данных для хранения данных пользователей;

– добавление функциональности на стороне сервера: регистрация, аутентификация, чат, планировщик задач;

– обоснование экономической целесообразности разрабатываемого продукта;

– предложение мероприятий по улучшению условий труда в рамках реализуемого проекта.

2 Проектирование программного продукта

Данная глава представляет собой описание структуры сайта медицинского предприятия. Она включает в себя анализ принципов построения грамотного сайта, который будет полезен для пользователей. Разделить главу можно на анализ принципов построения сайта, обоснование и выбор необходимого для построения сайта программного обеспечения, информационное обеспечение и разработка интерфейса, технологическое обеспечение задачи и саму реализацию программного продукта, которая будет финальной стадией разработки сайта.

2.1 Анализ принципов построения сайта

Интернет-услуги-одно из самых замечательных изобретений человечества, входящее в каждый дом. Интернет развивается и совершенствуется. Современные возможности сети радикально отличаются от сайтов прошлых лет. Создание сайтов сегодня базируется на основных принципах составления.

Дизайн.Главная задача веб-дизайна - объединить все информационные блоки и создать приятное впечатление о сайте для посетителей. На самом деле, проект устанавливает весь стиль сайта, помогая пользователю с первого взгляда понять, что его ждет здесь. Хорошо продуманный проект - один из важнейших факторов, влияющих на посещение вашего сайта.

В общем, дизайн сайта - это его внешний вид, который, по крайней мере, не отпугивает посетителей интернет-ресурсами и максимально уловкой. Как говорят опытные профессионалы, хороший дизайн-это невидимый, ненавязчивый дизайн, который не отвлекает зрителя от главного-от целевой аудитории определенной информации. Я думаю, что вы согласитесь, что человек, который приехал на любой ресурс в интернете, сначала ищет необходимую информацию (единственным исключением является дизайнер, который ищет новые решения и интересуется темами проекта). И в этой ситуации вся отвлекать информация (слишком яркий, привлекающий внимание дизайн сайта, излишняя анимация, всплывающие рекламные объявления) будет только мешать в достижении главной цели - получить необходимые данные, в поисках которых пришли сайта и посетителя.

Качество-основная часть создания сайта. Дизайн должен быть не только красивым, индивидуальным, но современные. Более новые концепции веб-дизайна предполагают наличие элегантной и креативной среды печати, которая прекрасно дополняет ядро дизайна и может помочь привлечь внимание посетителей к логотипам и заголовкам.

Современная технология В интернете появилось много устройств вывода. Фиксированные компьютеры, ноутбуки, планшеты и мобильные телефоны с успехом обеспечивают этот процесс, но имеют разные разрешения

экрана, что часто немного неудобно для просмотра страниц. Сегодня разработка сайта включает в себя применение современных гибких технологий проектирования. Это проект, который будет очень хорошо отображаться на любом устройстве, независимо от расширения экрана.

При таком допуске разработчики могут корректно видеть содержимое сайта на любом устройстве. Когда вы это сделаете, вы получите возможность охватить очень большого публичного пользователя, а значит сайт будет не только посещаться, но и выгоден. Особенно важно применять адаптивный дизайн к функциям с новостными блоками, потому что только посетители заинтересованы в знании свежих новостей, и поэтому блоки должны правильно отображаться на любом устройстве.

Основным отличием проекта от мобильных версий сайтов является возможность отображения страницы на различных типах устройств без создания и запуска специальных приложений. Адаптивный дизайн решает проблему разделения трафика, что положительно сказывается на статистике сайта трафик.

2.2 Обоснование и выбор программного обеспечения

Для создания Web-сайта своего дипломного проекта я выбрал программу Sublime Text 3, т.к. ее гибкие функциональные возможности позволяют создавать более совершенные Web-узлы, включающие средства для профессионального проектирования, разработки, работы с данными и публикации, необходимые для создания динамических и более сложных Web-узлов. Sublime Text 3 позволяет усовершенствовать процесс Web-разработки в следующих трех ключевых областях. Также эта программа является бесплатной для MAC и Windows.

Рисунок 2.2.1 – Sublime Text

Усовершенствованные средства Проектирования позволяют улучшить дизайн Веб-Сайтов. Новые средства разметки и графикой упрощают процесс создания веб-Сайтов, которые отвечают в полной мере в плане Пользователя.

Написать Код. С помощью средств дизайна, вы можете улучшить качество генерируемого кода и навыков программирования, повышения. Инструменты разработки, сценарии встроенные поддерживают интерактивность в продуктах, созданных. Профессиональные средства Написания кода можно работать быстрее, более эффективно и более точно.

Можно организовать общение и обмениваться данными по-новому, по data-driven с широкими интерактивными возможностями в Редакторе, который работает по принципу-то, что вы видите на экране.

2.3 Проектирование базы данных

Базы данных для веб-сайтов (БД) используются для хранения различной информации и просто представляют собой набор взаимосвязанных таблиц. Размер таблиц в базе данных разный, их количество произвольное. Это в базах данных на сервере для работы сайта информация, как клиента хранится информация, каталог продукции, статистика и так далее.

Программирование динамических сайтов выполняется с использованием различных сценариев, которые обычно делятся на сервер и клиент. Серверные скрипты позволяют пользователям веб-сайтов обрабатывать данные в веб-формах, создавать динамические страницы, отправлять и получать файлы cookie. Серверные скрипты получают доступ к базам данных для получения информации, необходимой для этих действий. Доступ к скрипту к базе данных называется запросом.

SQL (Structured Query Language) - "структурированный язык запросов" часто используется для создания запросов к базам данных. SQL позволяет добавлять, удалять, редактировать записи в таблицах базы данных, выбирать данные по различным условиям, сортировать данные и многое другое.

Для проектирования базы данных в дипломном проекте была выбрана СУБД MySQL. MySQL - это бесплатная база данных с открытым исходным кодом, которая облегчает эффективное управление базами данных, подключая их к программному обеспечению. Это стабильное, надежное и мощное решение с расширенными функциями и преимуществами:

Безопасность данных. Будучи более безопасной и надежной системой управления базами данных, MySQL известен во всем мире и используется в популярных веб-приложениях, таких как WordPress, Drupal, Joomla, Facebook и Twitter. Безопасность данных и поддержка обработки транзакций, которые сопровождают самую последнюю версию MySQL, могут принести большую пользу любому бизнесу, особенно если это бизнес электронной коммерции, который включает частые денежные переводы.

Управляемое масштабирование. MySQL предлагает непревзойденную масштабируемость, чтобы упростить управление глубоко внедренными приложениями, используя меньшие по размеру следы даже в больших хранилищах, в которых хранятся терабайты данных. Главная особенность MySQL - гибкость по требованию. Это решение с открытым исходным кодом позволяет полностью настроить предприятия электронной коммерции с уникальными требованиями к серверу баз данных.

Высокая производительность. Отличительная особенность механизма хранения MySQL позволяет системным администраторам настраивать сервер базы данных MySQL для безупречной производительности. Являясь веб-сайтом электронной коммерции, который получает миллион запросов каждый день, или высокоскоростной системой транзакционной обработки, MySQL разработан для удовлетворения даже самых требовательных приложений, обеспечивая оптимальную скорость, полнотекстовый индекс и уникальные кэш-памяти для повышения производительности.

Круглосуточность. MySQL предоставляет полную гарантию бесперебойной работы 24X7 и предлагает широкий спектр решений высокой доступности, таких как специализированные кластерные серверы и конфигурации репликации master / slave.

Комплексная поддержка транзакций. MySQL занимает первое место в списке надежных механизмов транзакционных баз данных, доступных на рынке. Благодаря таким функциям, как полная автоматическая, согласованная, изолированная и надежная поддержка транзакций, поддержка нескольких версий транзакций и неограниченная блокировка на уровне строк, это решение для полного доступа целостность данных. Это гарантирует мгновенную идентификацию тупика через принудительную ссылочную целостность сервера.

Полный контроль рабочего процесса. Поскольку среднее время загрузки и установки составляет менее 30 минут, MySQL означает удобство использования с самого первого дня. В качестве платформы используется Linux, Microsoft, Macintosh или UNIX, MySQL - это комплексное решение с функциями самостоятельного управления, которое автоматизирует все, начиная с расширения пространства. и конфигурация для проектирования данных и администрирования данных.

В сайте программирования администрирование баз данных с клиент-серверной системы управления базами данных (СУБД), таких как Oracle, MS SQL Server, PostgreSQL, MySQL и т. д. проводится. Клиент-сервер СУБД обрабатывают запросы централизованно, их преимущества-Высокая надежность баз данных, высокая доступность и высокая безопасность.

MySQL СУБД является бесплатной системой управления базами данных, один из наиболее часто используемых в программировании сайтов. База данных MySQL поддерживает большое количество существующих типов таблиц (InnoDB, MyISAM и т. д.), и благодаря открытой архитектуре и GPL-

лицензированию, MySQL-база данных, которая постоянно появляются новые типы таблиц. Управление базами данных с MySQL очень удобно, что сделало эту систему популярной.

Система управления реляционными базами данных Microsoft SQL Server предоставляется корпорацией Майкрософт на коммерческой основе (за исключением бесплатного выпуска Express Edition). Эта СУБД использует язык запросов Transact-SQL и поддерживается семейством Windows Desktop / Server операционных систем. В Microsoft SQL Server существует графическое программное обеспечение для проектирования и оптимизации запросов (SQL Management Studio или Studio Express).

Oracle Object-реляционная система управления базами данных-база данных Oracle-работает в Windows, Unix, Linux, MacOS. База данных Oracle, в отличие от MySQL, например, имеет более широкий диапазон. СУБД Oracle имеет высокую производительность, широкий функционал, уникальные технологии (RAC,СОВЕТ и т. д.). В программировании сайтов для малого и среднего бизнеса редко используется из-за его высокой стоимости. Кроме того, довольно сложно найти хостинг с поддержкой этой базы данных.

Свободная система управления базами данных PostgreSQL существует в версиях для Linux, Solaris/OpenSolaris, Win32, Win x86-64, Mac OS X, FreeBSD, QNX 4.25, QNX 6. Он основан на языке SQL. Преимущества PostgreSQL включают поддержку базы данных практически неограниченного размера, доступность надежных механизмов репликации, простую расширяемость, поддержку большого набора встроенных типов данных и многое другое.

Программирование веб-сайтов, взаимодействующих с базами данных различными способами, включает в себя несколько основных этапов работы с базой данных: создание запросов к базе данных с помощью SQL, Программирование скриптов для обработки этих запросов и модули программирования для отображения результатов обработки запросов.

Чрезмерное количество вызовов веб-сайтов в базах данных делает загрузку веб-сайтов медленнее, увеличивает нагрузку на сервер. В результате могут возникнуть ошибки в работе веб-сайтов до полного прекращения доступа. Уменьшение количества запросов к базе данных может снизить нагрузку на сервер и сократить время загрузки динамических страниц с сервера. Поэтому оптимизация взаимодействия сайтов с базами данных-одна из задач профессионального программирования сайтов.

Таблица является основным объектом для хранения информации в реляционной базе данных. Он состоит из строк и столбцов, содержащих данные, занимающих физическое пространство в базе данных и может быть постоянным или временным. Поле, также называемое столбцом в реляционной базе данных, является частью таблицы с определенным типом данных. Каждая

таблица базы данных должна содержать по крайней мере один столбец. Строка данных - это запись в таблице базы данных, содержащая поля, содержащие данные из набора таблиц.

Главное в команде создание таблицы - это определение имени таблицы и описание имен полей, указанных в соответствующем порядке. Эта команда также указывает типы данных и размеры полей таблицы.

В отличие от плоских баз реляционные базы данных состоят из нескольких таблиц, егоже связь устанавливается путем сопоставления значений одноименного поля.

Это должно быть не то, что использование реляционной базы данных модель не единственный способ представить информацию. В настоящее время существует несколько различных моделей представления данных, но они не так распространены среди разработчиков и пользователей, как реляционная модель. Это означает, что при разработке систем управления базами данных реляционная модель практически стандартная

Ключевое слово NULL используется для указания того, что этот столбец может содержать нулевые значения. Null отличается от space или zero-он используется, если вы хотите указать, что данные недоступны, опущены или недействительны. Если ключевое слово не равно NULL, любая попытка поместить нулевое значение в этот столбец будет отклонена. Если задано значение NULL, допускается установка значений NULL в столбце. По умолчанию SQL принимает ключевое слово NULL.Итак, как же будет выглядеть база данных больницы. Она состоит из таких таблиц:

- hospital (больница);
- users (врачи);
- visits (записи к врачам);
- e_queue (электронная очередь);
- patients (пациенты);
- hospital certificate (выписка);
- drugs (лекарства).

Таблица базы данных больница содержит в себе название больницы и её адрес.

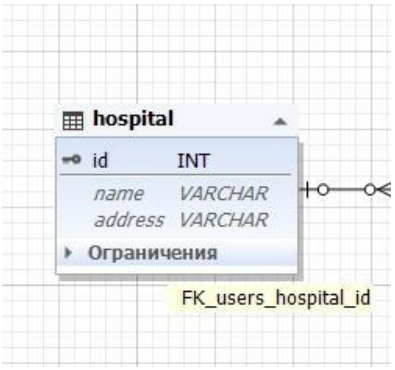

Рисунок 2.3.1 – Таблица базы данных больница

Программная реализация:

```
CREATE TABLE `hospital` (
  `id` int(11) NOT NULL,
  `name` varchar(15) NOT NULL,
  `address` varchar(15) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 2.3.2 – Реализация таблицы больница

Таблица пользователи содержит в себе айди, имя, фамилию, должность и пароль врачей, которые работают в больнице.

Она связана с больницей, в которую вносятся врачи.

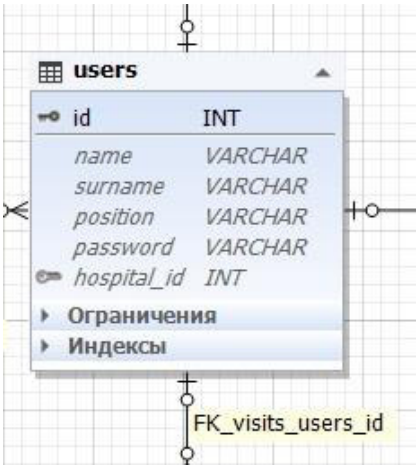

Рисунок 2.3.3 – Таблица базы данных пользователи

Программная реализация:

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `name` varchar(15) NOT NULL,
  `surname` varchar(15) NOT NULL,
  `position` varchar(20) NOT NULL,
  `password` varchar(20) NOT NULL,
  `hospital_id` int(11) NOT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 2.3.4 – Реализация таблицы пользователи

Таблица визиты содержит айди пользователя и время визита.

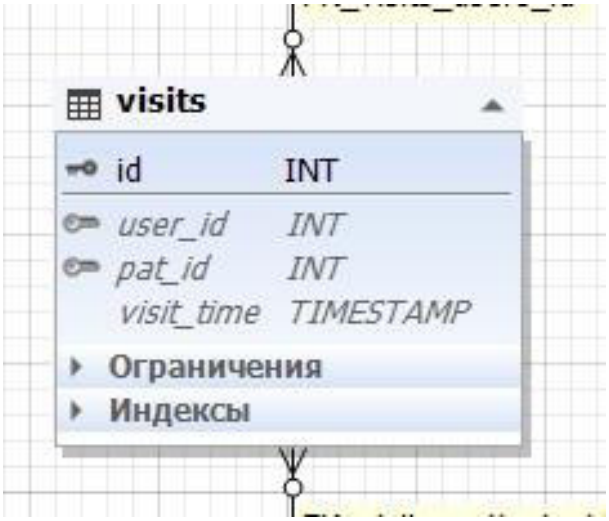

Рисунок 2.3.5 – Таблица базы данных визиты

Программная реализация:

```
CREATE TABLE `visits` (
  `user_id` int(11) NOT NULL,
  `pat_id` int(11) NOT NULL,
  `visit_time` int(20) NOT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 2.3.6 – Реализация таблицы визиты

Таблица электронная очередь содержит айди доктора и пациента, а также время, на которое он записан.

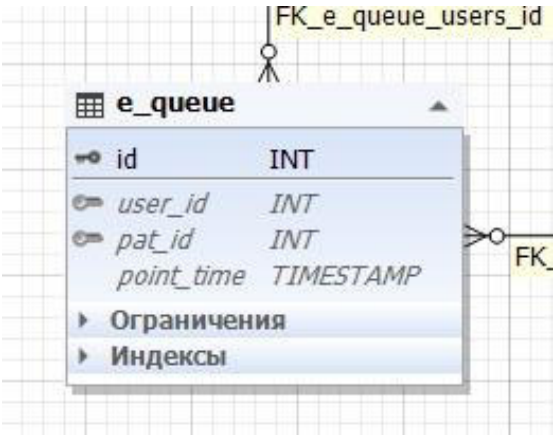

Рисунок 2.3.7 – Таблица базы данных электронная очередь

Программная реализация:

```
CREATE TABLE `e_queue` (
  `user_id` int(11) NOT NULL,
  `pat_id` int(11) NOT NULL,
  `point_time` int(20) NOT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 2.3.8 – Реализация таблицы электронная очередь

Таблица пациенты содержит в себе имя, фамилию, адрес, иин, день рождения и номер карты пациента.

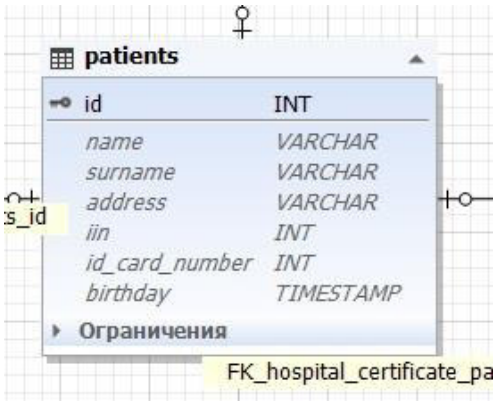

Рисунок 2.3.9 – Таблица базы данных пациенты

Программная реализация:

```
CREATE TABLE `patients` (
  `id` int(11) NOT NULL,
  `name` varchar(15) NOT NULL,
  `surname` varchar(15) NOT NULL,
  `address` varchar(15) NOT NULL,
  `iin` varchar(15) NOT NULL,
  `id_card_number` varchar(15) NOT NULL,
  `birthday` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 2.3.10 – Реализация таблицы пациенты

Таблица выписка содержит айди лекарства, айди врача, который выписал лекарство, болезнь и айди самого пациента.

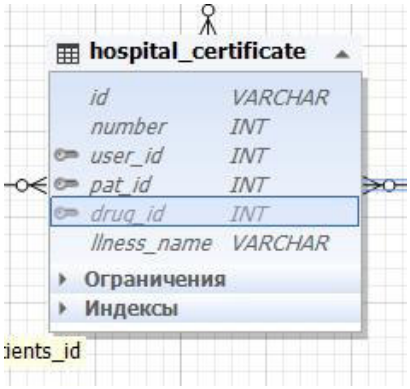

Рисунок 2.3.11 – Таблица базы данных выписка

Программная реализация:

```
CREATE TABLE `hospital_certificate` (
  `id` int(11) VARCHAR,
  `number` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `pat_id` int(11) NOT NULL,
  `drug_id` int(11) NOT NULL,
  `Ilness_name` int(20) NOT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 2.3.12 – Реализация таблицы справка

Таблица лекарство содержит в себе название лекарства и его цену и является дочерним элементом для таблицы справки.

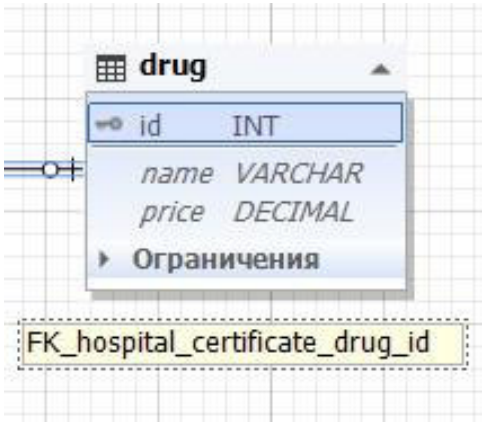

Рисунок 2.3.13 – Таблица базы данных лекарства

Программная реализация:

```
CREATE TABLE `drugs` (
  `id` int(11) NOT NULL,
  `name` varchar(20) NOT NULL,
  `price` varchar(20) DECIMAL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 2.3.14 – Реализация таблицы лекарства

Также впоследствии была добавлена таблица вызовы, дабы была возможность вызывать врача на дом.

```
CREATE TABLE `vizovy` (
  `id` int(11) NOT NULL,
  `id_pat` int(15) NOT NULL,
  `id_doc` int(15) NOT NULL,
  `number` varchar(20) NOT NULL,
  `pricina` varchar(50) NOT NULL,
  `date` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 2.3.15 – Реализация таблицы лекарства

Итак, соединив все таблицы, база данных приобретает конченый вид. Остается заполнить её врачами и лекарствами.

Для заполнения таблиц базы данных будем использовать команду INSERT INTO. В ковычках указывается название таблицы, а в скобках

значения, которые будут передаваться полям, которые содержатся в данной таблице.

На диаграмме также видно, как и по каким полям связаны таблицы базы данных.

FK(Foregn Key) используется для создания связи один ко многим и показывает, в какую таблицу передаются данные из другой. Ключи обозначают поля, по которым связаны таблицы. На основе этой базы данных и будет строиться структура сайта.

Итак, конечная базы данных нашего медицинского предприятия примет такой вид:

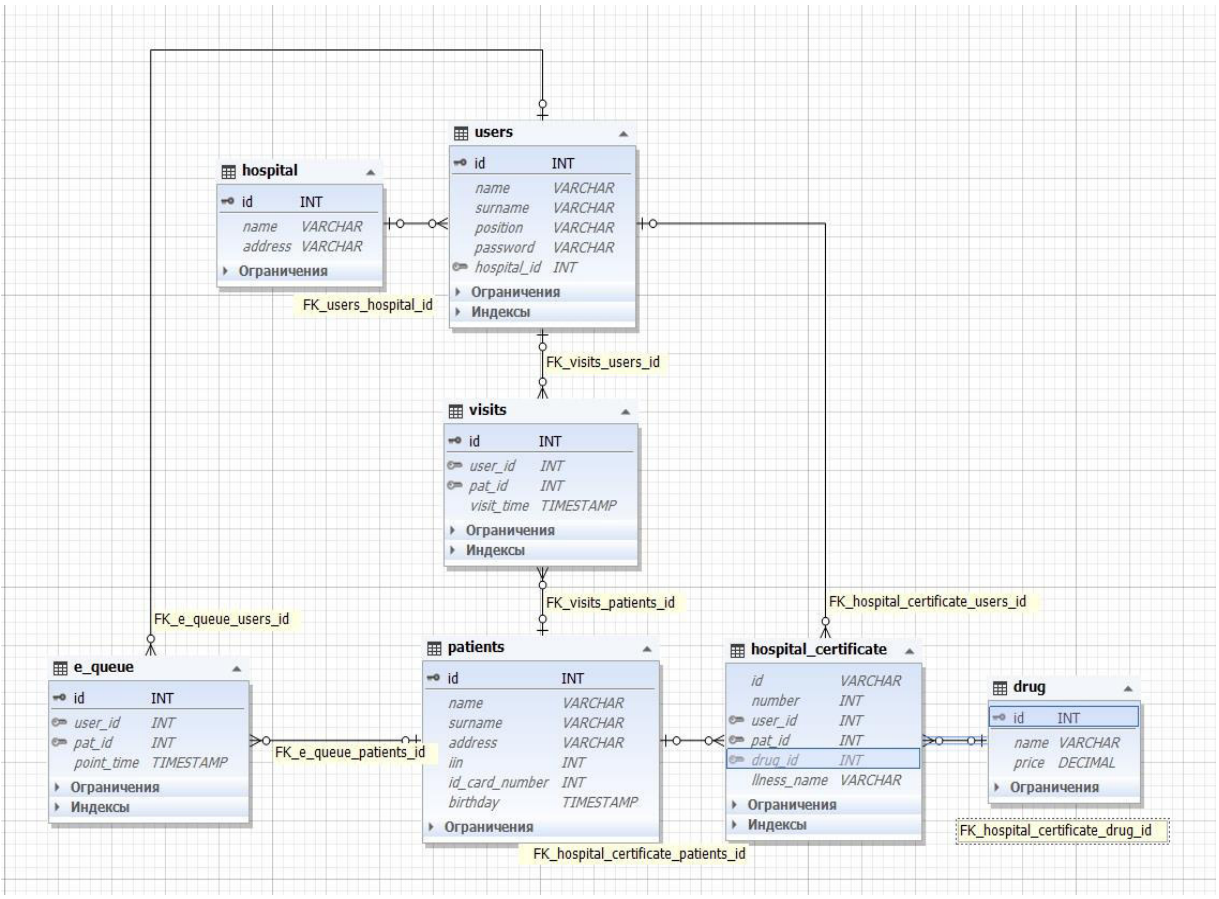

Рисунок 2.3.16 – Таблица базы данных медицинского предприятия

Заполняем таблицы “врачи” и “лекарства”. На диаграмме также видно, как и по каким полям связаны таблицы базы данных.

```
INSERT INTO `users` (`id`, `name`, `surname`, `Otchestvo`, `login`, `position`, `password`, `hospital_id`, `description`, `img`, `recenzia`) VALUES
(1, 'Артур', 'Ким', 'Баартурович', 'sago', 'стоматолог', '123321', 1, 'Стоматолог проводит осмотр полости рта пациента, выслушивает жалобы, осуществляет лечение, удаление зубов, проводит профилактику и гигиеническую чистку зубов. Рабочее время врача-стоматолога проходит ', 'static/images/13.png', 'Стоматолог проводит осмотр полости рта пациента, выслушивает жалобы, осуществляет лечение, удаление зубов, проводит профилактику и гигиеническую чистку зубов.В работе ему помогает медицинская сестра.'),
(2, 'Азиза', ' Агатова', 'Викторовна', 'aziza228', 'педиатр', '3228', 1, 'Стоматолог проводит осмотр полости рта пациента, выслушивает жалобы, осуществляет лечение, удаление зубов, проводит профилактику и гигиеническую чистку зубов. Рабочее время врача-стоматолога проходит в одной и той же позе: либо стоя, либо сидя. В работе ему помогает медицинская сестра. В процессе работы он может советоваться с пародонтологом, ортопедом, ортодонтом.', 'static/images/9.png',
'Стоматолог проводит осмотр полости рта пациента, выслушивает жалобы, осуществляет лечение, удаление зубов, проводит профилактику и гигиеническую чистку зубов.В работе ему помогает медицинская сестра.'),
(3, 'Айгуль', 'Аскарбаева', 'Избарасовна', 'ayka228', 'гинеколог', '123321', 1, 'Основная наша цель – помочь женщине обрести здоровье, почувствовать себя энергичной. Ведь женское здоровье является основополагающим фактором для рождения здоровых детей. Консультация гинеколога поможет не только избавиться от гинекологических патологий, но и достичь гармонии в интимной жизни.', 'static/images/11.png', 'Длительность приема – около 30 минут. Прием пациентов с 18 лет.'),
(4, 'Николай', 'Градский', 'Александрович', 'grad322', 'терапевт', '123321', 1, 'Оказывает квалифицированную медицинскую помощь по своей специальности, используя современные методы профилактики, диагностики, лечения и реабилитации, разрешенные для применения в медицинской практике. Определяет тактику ведения больного в соответствии с установленными правилами и стандартами. Разрабатывает план обследования больного, уточняет объем и рациональные методы обследования пациента с целью получения в минимально короткие сроки полной и достоверной диагностической информации.', 'static/images/11.jpg', 'Длительность приема – около 30 минут. Прием пациентов с 18 лет.'),
(5, 'Галина', 'Савенко', 'Михайловна', 'galina_zvezda', 'терапевт', '123321', 1, 'Оказывает квалифицированную медицинскую помощь по своей специальности, используя современные методы профилактики, диагностики, лечения и реабилитации, разрешенные для применения в медицинской практике. Определяет тактику ведения больного в соответствии с установленными правилами и стандартами. Разрабатывает план обследования больного, уточняет объем и рациональные методы обследования пациента с целью получения в минимально короткие сроки полной и достоверной диагностической информации.', 'static/images/12.png', 'Длительность приема – около 30 минут. Прием пациентов с 12 лет.');
```

Рисунок 2.3.17 – Заполнение таблицы врачей

```
INSERT INTO `drugs` (`id`, `name`, `cost`, `apteka`, `size`) VALUES
(1, 'Новакаин', 2000, 'Городская аптека №1', '250г'),
(2, 'Стрептоцид', 500, '№2 Городская аптека', '250г;300г');
(3, 'Нимулид', 1400, 'Городская аптека №3', '250г'),
(4, 'Доктор Мом', 200, 'Городская аптека №1', '250г'),
(5, 'Смекта', 60, 'Городская аптека №2', '250г'),
(6, 'Активированный уголь', 100, 'Центральная аптека', '250г'),
(7, 'Пантенол', 3000, 'Городская аптека №2', '250г'),
(8, 'Тауфон', 250, 'Городская аптека №1', '250г'),
```

Рисунок 2.3.18 – Заполнение таблицы лекарств

3 Разработка программного продукта

Веб-приложение представляет собой веб-сайт, на котором размещены страницы с частично либо полностью несформированным содержимым. Окончательное содержимое формируется только после того, как посетитель сайта запросит страницу с веб-сервера. В связи с тем что окончательное содержимое страницы зависит от запроса, созданного на основе действий посетителя, такая страница называется динамической.

Использование веб-приложений приносит определенную пользу как посетителям веб-сайтов, так и их разработчикам.

Веб-приложения позволяют посетителям быстро и легко находить требуемую информацию на веб-сайтах с большим объемом информации.

Веб-приложения позволяют собирать, сохранять и анализировать данные, полученные от посетителей сайта.

Веб-приложение может использоваться для обновления веб-сайтов с периодически меняющимся содержимым.

Веб-приложение освобождает веб-дизайнера от рутинной работы постоянного обновления HTML-страниц сайта. Поставщики содержимого, например редакторы новостей, отвечают за наличие свежего материала, а веб-приложение следит за автоматическим обновлением сайта.

Любое веб-приложение представляет собой набор статических и динамических веб-страниц. Статическая веб-страница — это страница, которая всегда отображается перед пользователем в неизменном виде. Веб-сервер отправляет страницу по запросу веб-браузера без каких-либо изменений. В противоположность этому, сервер вносит изменения в динамическую веб-страницу перед отправкой ее браузеру. По причине того что страница меняется, она называется динамической.

Элемент <Body> содержит все содержимое HTML-документа, такие как текст, гиперссылки, изображения, таблицы, списки и т. д.

Заголовок страницы задается в коде страницы с тегами <head></head>. Все между этими тегами определяет, как сторонние сервисы, такие как поисковая система Google, Twitter, Facebook и ваш браузер, видят страницу.

Рисунок 3.1 – Header на нашем сайте

```
<head>
  <title>Достижения</title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
  <script src='/socket.io/socket.io.js'></script>

  <link rel="stylesheet" type="text/css" href="static/css/css_for_vrach.css">
  <style>
    .qwer ul li{
      list-style-type: none;
      margin-left: 0;
    }
    .qwer ul{
      padding-left: 0;
      border: 1px solid black;
      width: 250px;
    }
    table td{
      border: 1px solid black;
    }
    #lec_info{
      display: none;
    }
  </style>
</head>
```

Рисунок 3.2 – Header

Важно не путать заголовок страницы с шапкой сайта. Термины почти идентичны, но один описывает раздел страницы, который люди видят и взаимодействуют с ним, в то время как другой описывает область HTML страницы (или фонового кода), которая сообщает сторонним службам, о чем страница.

Общие элементы, найденные в тегах <head></head > включают (но не ограничиваются):

Название: это относится к названию HTML-документа. Каждая страница имеет свое уникальное название.

Стиль: здесь задаются параметры стиля, которые влияют на всю страницу. Например, если вы хотите, чтобы текст был определенного цвета, это где вы бы установить, что.

Мета: мета - информация, такая как заголовок веб-страницы (Примечание: отличается от набора в теге заголовка), автор и рекомендуемое изображение помогут социальным сетям отображать любые ссылки на эту страницу наилучшим образом. Информация об авторских правах, ключевые слова и многое другое хранятся в мета-теги. Эта информация также используется, чтобы помочь поисковым системам лучше каталогизировать каждую страницу.

Ссылка: тег ссылки используется для обозначения любых внешних таблиц стилей, которые вы хотите загрузить на странице.

Содержимое нижнего колонтитула веб-страницы чаще всего передает техническую информацию.

Нижний Колонтитул (Англ. Нижний колонтитул) - важный структурный элемент любого сайта, обычно называемый "подвал". Золото веб-страницы помещается и является противоположным заголовку (заголовку) выше.

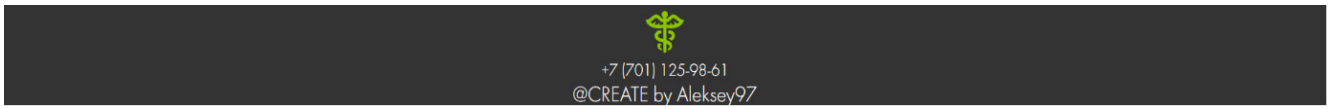

Рисунок 3.3 – Footer на нашем сайте

```
<body>
  <a href="/"><svg style="fill:#8BC500; cursor: pointer;" xmlns="http://www.w3.org/2000/svg" width="34" height="34" viewBox="0 0 24 24"><path d="M13.427
  3.021h-7.427v-3.021l-6 5.39 6 5.61v-3h7.427c3.071 0 5.561 2.356 5.561 5.427 0 3.071-2.489 5.573-5.561 5.573h-7.427v5h7.427c5.84 0 10.573-4.734
  10.573-10.573s-4.733-10.406-10.573-10.406z"/></svg></a>
  <p class="rwefk">Кабинет врача</p>
  <p class="rwefk1" id="name"></p>
  <div class="main_div_1">
    <div class="online_chat" id="online_chat">
      <button  class="signin buttons_class" id="zapis">Мои записи</button>
      <button  class="signin buttons_class" id="vizov">Мои вызовы</button>
      <button  class="signin buttons_class" id="vipiski">Мои выписки</button>
    </div>
<input class="input_calend" type="date" name="calendar" id="viz_date" max="2019-12-31">
<button class="btn-del buttons_class1" id="filter">Выбрать</button>
    <div id="zap_info">

    </div>
    <div id="viz_info">

    </div>
    <div id="vip_info">
```

Рисунок 3.4 – Body

Тег <body> определяет тело документа.

Элемент <Body> содержит все содержимое HTML-документа, такие как текст, гиперссылки, изображения, таблицы, списки и т. д.

Нижний колонтитул включает информацию об авторских правах для веб-сайта, имя автора веб-сайта, название компании и адрес, если это применимо, а также дату последнего обновления. Некоторые нижние колонтитулы веб-страниц включают ссылки на другие страницы веб-сайта, ссылку по электронной почте или ссылку RSS. В верхней части веб-страницы содержится самая ценная "недвижимость" на веб-сайте, поэтому содержимое нижнего колонтитула имеет меньшее значение.

Конструкция Нижнего Колонтитула

Дизайн нижнего колонтитула сочетается и дополняет остальную часть дизайна веб-сайта. Дизайн нижнего колонтитула использует те же шрифты, цветовые схемы и графику. Дизайн нижнего колонтитула полностью зависит от автора веб-сайта. Некоторые нижние колонтитулы содержат только одну прямоугольную строку текста, охватывающую ширину веб-сайта, в то время как другие разделены на разделы. Некоторые компании используют нижний колонтитул своего сайта для отображения дополнительных изображений или рекламных акций. Нижние колонтитулы могут содержать медиа-контент, ссылки в социальных сетях и интерактивный контент, такой как опросы.

Значение Нижнего Колонтитула

Нижний колонтитул является важным разделом веб-сайта по двум причинам. Во-первых, он отображает соответствующую техническую информацию для пользователя компьютера. Автор сайта отображает информацию и выбирает контент исходя из потребностей сайта. Нижний колонтитул также служит большой цели в создании более последовательного взгляда на веб-сайт. Это может быть достигнуто с помощью дизайна, изображений и средств массовой информации.

В случае больницы статическим является основная страница сайта, которая показывается пользователю в одном и том же виде всегда вне зависимости от того, какую информацию хочет получить пользователь.

Динамическими же являются страницы личного кабинета врача и пациента.

Для начала необходимо создать новую сессию

Теперь необходимо соединиться с базой данных. Это делается посредством команды:

```
var options = {
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'test'
}
```

Рисунок 3.5 – Соединение с базой данных

При необходимости обработать ошибки подключения.

```
connection.connect(function(error) {
    if(error){
        console.log('Error');
    } else{
        console.log('Connected');
    }
});
```

Рисунок 3.6 – Обработка ошибок

Сайт состоит из шапки, то есть верхней части (header), основной части (body), и нижней части (footer), в которой обычно содержится логотип, социальные сети компании и вся контактная информация.

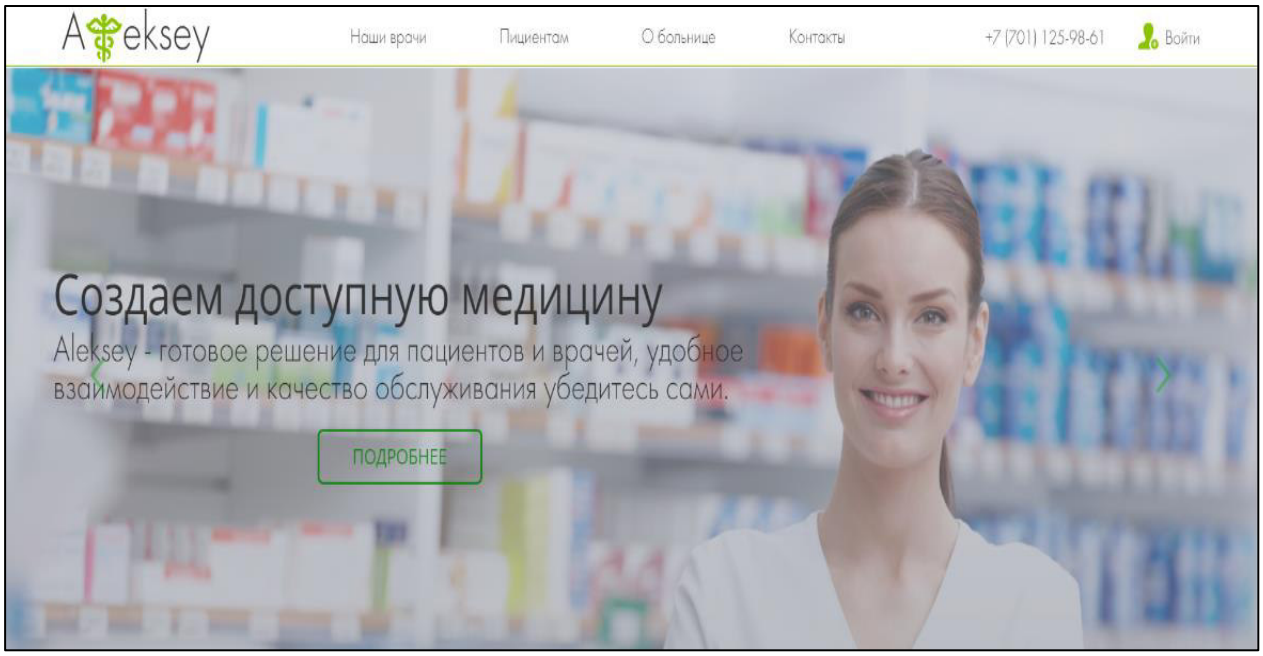

Рисунок 3.7 – Вид сайта для незарегестрированного пользователя

В header используется меню навигации, которое позволяет пользователю переходить в необходимую ему категорию в один клик, не производя переход на другую страницу.

После того, как мы нажимаем кнопку войти, появляется форма, в которой есть кнопка “регистрация”. Нажав на эту кнопку, мы увидим форму такого вида:

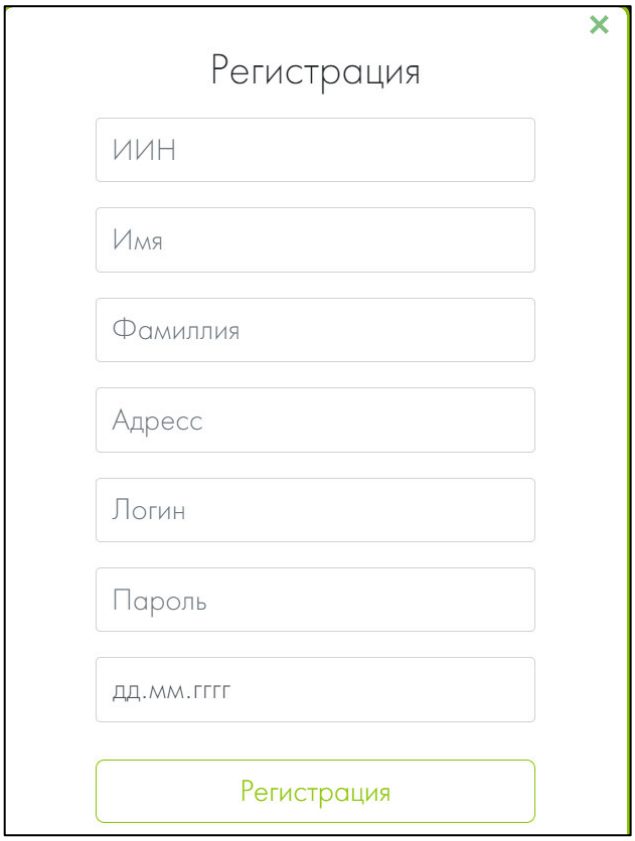

Рисунок 3.8 – Регистрация

При регистрации отправляется ajax запрос к базе данных на внесение нового пользователя через форму регистрации, и, если новый пользователь добавлен успешно, то пациент сможет войти в свой кабинет.

```
$('#form_button').click(function(){
    $.ajax({
     method:"POST",
     url:'/registr',
     cache: false,
     data: $('#form_regist').serialize(),
     success:function (datas) {
       if(datas.message == "ok"){
       window.location.reload(); }
     }
   });
})
```

Рисунок 3.9 – Запрос к базе данных на добавление пользователя

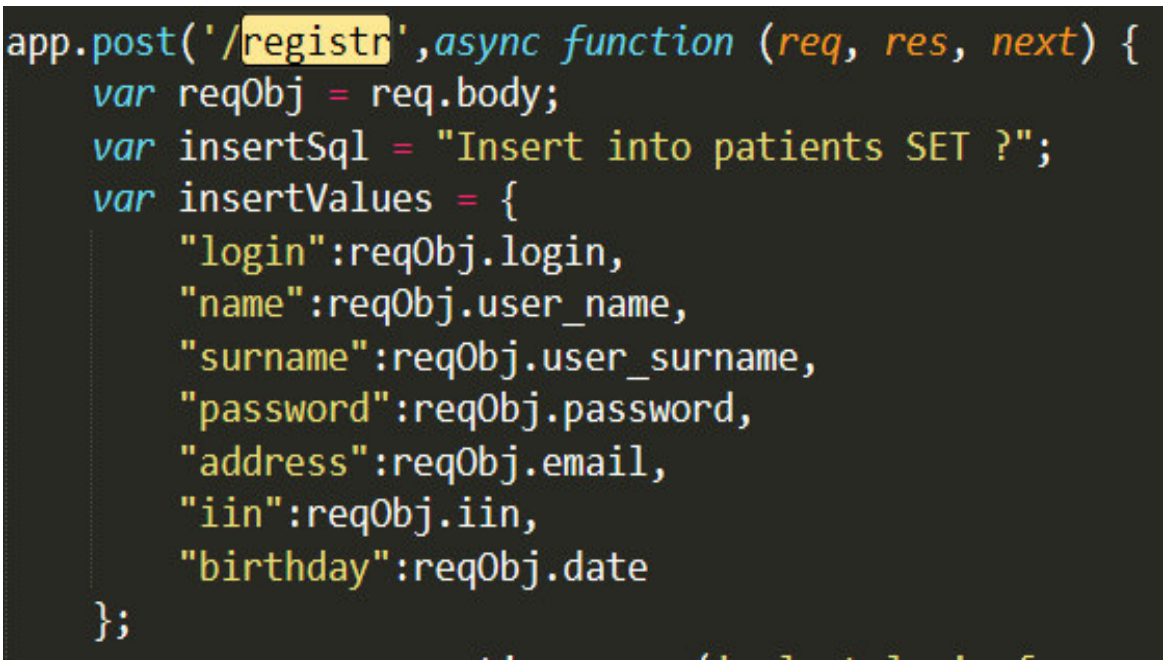

Рисунок 3.10 – Программная реализация регистрации через внесение пользователя в базу данных

Далее необходимо обработать ошибки при регистрации, чтоб пользователь мог увидеть, если он неправильно заполнил одно из необходимых полей. Также ошибки будут обрабатываться при заполнении любого из других полей. Это делается таким образом.

```
connection.query('select id, name, surname, password from patients where login = "'+req.body.login+'"', function (error, rows, fields) {
    if(!!error){
        console.log('Error in the query');
    } else
    {
        if(rows.length>0)
            {
                console.log(rows);
                if(req.body.password == rows[0].password)
            {
              console.log("eqwe")
                req.session.name = rows[0].name;
                req.session.surname = rows[0].surname;
                req.session.pid = rows[0].id;
                req.session.status = 'patient';
                res.json({message: 'ok'});

                } else
                {
                    res.status(418);
                    res.json({message: 'Неверный пароль'});
                }
            } else
            {
                res.status(202);
                res.json({message: 'Данного пользователя не существует'});
            }
        }
    });
```

Рисунок 3.11 – Обработка ошибок при регистрации

```
getLogin();
    function getLogin(){
    $.ajax({
      method:"POST",
      url:'/user',
      cache: false,
      success:function (datas) {
        $('#emptyblock').empty();
        var usename = document.createElement('p');
        var divobsh = document.createElement('div');
        var moicabinet = document.createElement('button');
        var logoutt = document.createElement('button');
        $(moicabinet).addClass("button_for_moicab");
        $(usename).addClass("text_for_moicab");
        $(divobsh).addClass("div_for_moicab");
        $(moicabinet).text("Мой кабинет");
        $(logoutt).text("Выход");
        $(logoutt).addClass("button_log_out");
       $(usename).text(datas.message.name+" "+datas.message.surname);
        moicabinet.setAttribute('onclick',"location.href='/cabinet';")
        divobsh.appendChild(usename);
        divobsh.appendChild(moicabinet);
        divobsh.appendChild(logoutt);
        var divfordivobsh = document.getElementById("emptyblock");
        divfordivobsh.appendChild(divobsh);

        $(logoutt).click(function(){
                $.ajax({
      method:"GET",
      url:'/logout',
      cache: false,
      success:function (datas) {
        if(datas.message==="ok"){
          window.location.reload();
        }
      }
```

Рисунок 3.12 – Ajax запрос на получение логина пользователя из базы данных

Далее, нажав кнопку регистрация, данные вносятся в базу данных и появляется возможность зайти, используя свой логин и пароль и появляется форма.

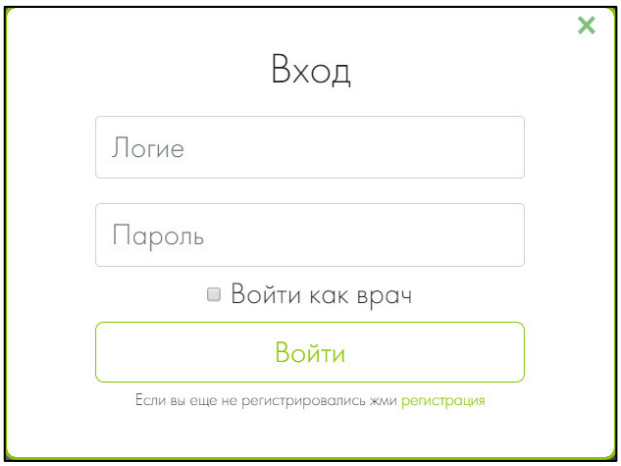

Рисунок 3.13 – Форма для регистрации

Если логин и пароль введены правильно, пациент должен увидеть в верхей части экрана свои имя, фамилию и возможность войти в личный кабинет, дабы посмотреть свои записи

Рисунок 3.14 – Что видит зарегестрированный пользователь

Пользователь может записаться на прием к врачу, либо вызвать его на дом, заполнив одну из форм, а затем посмотреть свою запись в личном кабинете.

Независимо от выбора посетителя, события отправки генерируется. Обработчик может проверить данные и, если они не правельнве, он будет выдавать ошибки и события метод preventDefault (), то форма не отправляется на сервер.

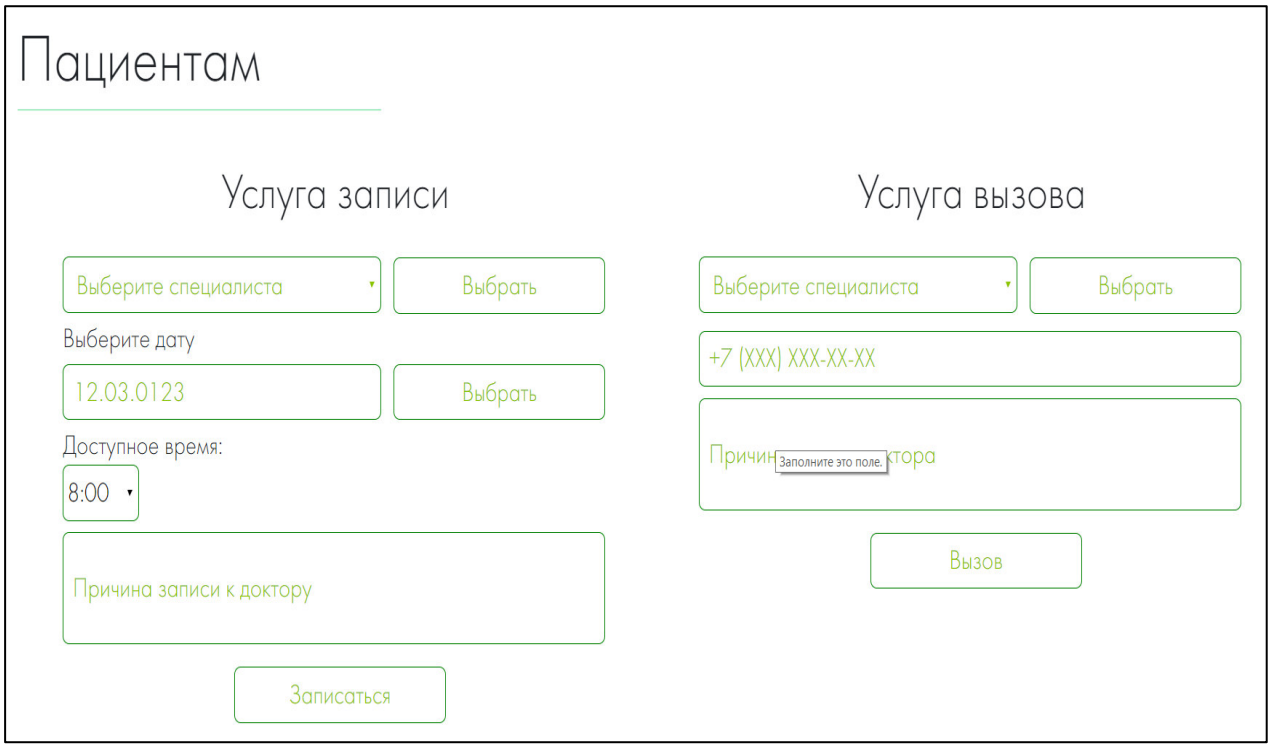

Рисунок 3.15 – Формы для записи или вызова врача на дом во вкладке пациентам

При заполнении формы, данные вносятся в базу, в которую передается ajax запрос, а затем их можно посмотреть, перейдя в личный кабинет. Передача данных реализуется таким образом:

```
$('#zapis').click(function() {
var id = ($('#curdoc').find('option:selected').attr('value'));
var time = ($('#curtime').find('option:selected').attr('value'));
var date =$("#viz_date").serialize().slice(9);
var pricina =$("#pricina").val();
var doctor = ($('#curdoc').find('option:selected').text());
var data = {
  id_doc:id,
  doctor:doctor,
  time:time,
  date:date,
  pricina:pricina
}
console.log(data)
$.ajax({
  method:"POST",
  url:'/zapis',
  cache: false,
  data: data,
  success:function (datas) {
    console.log(datas)

  }
});
```

Рисунок 3.16 – Функция и ajax запрос в базу данных для записи

```
$('#vizov').click(function() {
var id = ($('#curdoc1').find('option:selected').attr('value'));
var pricina =$("#pricina1").val();
var nomer = $('#tel').val();
var data = {
  id_doc:id,
  pricina:pricina,
  nomer:nomer
}
console.log(data)
$.ajax({
  method:"POST",
  url:'/vizov',
  cache: false,
  data: data,
  success:function (datas) {
    console.log(datas)

  }
});
```

Рисунок 3.17 – Функция и ajax запрос в базу данных для вызова врача на дом

После того, как пациент записался к врачу, он сможет увидеть свои записи и время в своем личном кабинете.

Врач, в свою очередь, увидит, что у него появился новый пациент в назначенное время и с определенной болезнью. Это будет выглядеть так:

Рисунок 3.18 – Личный кабинет и записи пациента будут выглядеть таким образом

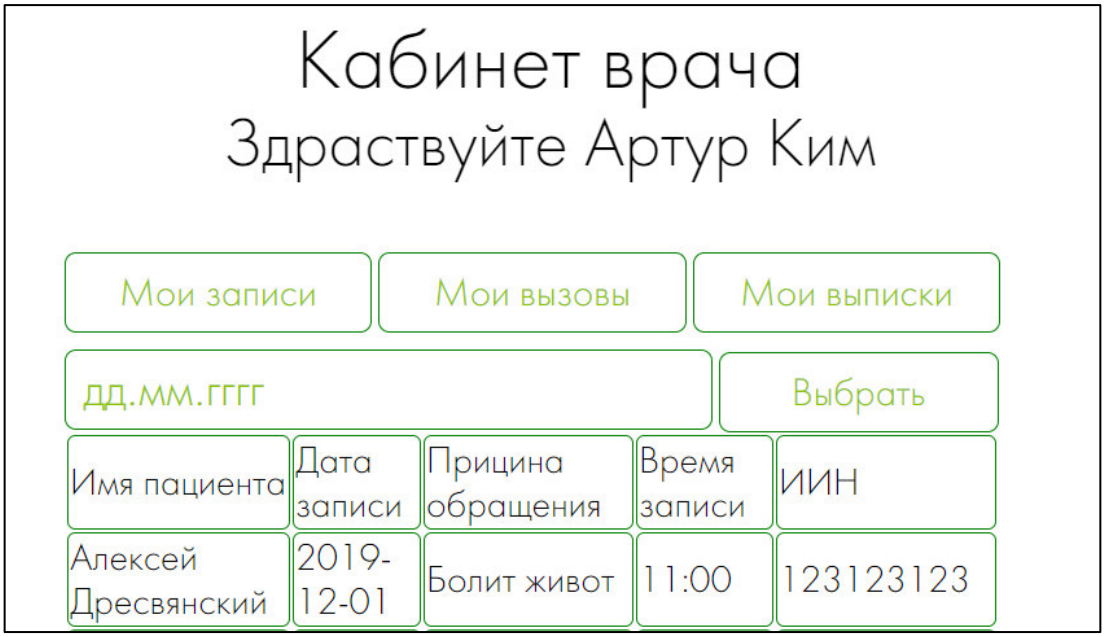

Рисунок 3.19 – Личный кабинет врача

В базу данных внесены все врачи больницы, их должности и средняя длительность приема, в зависимости от которой будет строиться их расписания.

Пользователи больницы смогут сами управлять внесением новых врачей с помощью phpmyadmin.

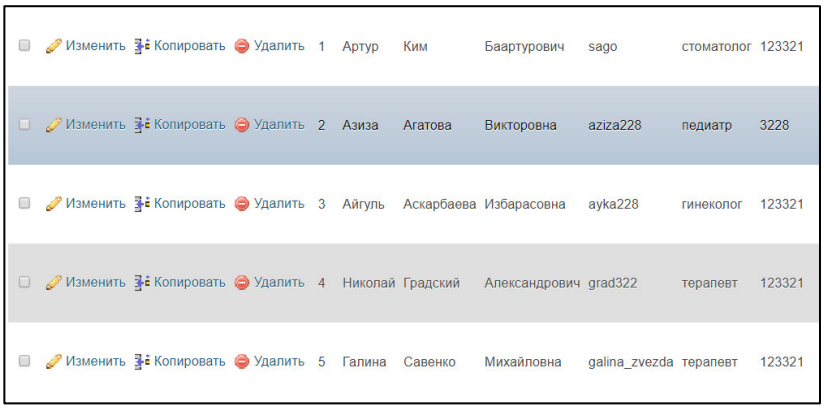

Рисунок 3.20 – База данных, в которую будут вноситься все врачи

Для добавления нового врача необходимо нажать “вставить” и заполнить все необходимые поля. Тогда врач появится на главной странице и к нему можно будет записаться. Удобство данного ПО состоит в том, что обычному пользователю не придётся разбираться с кодом, врачи добавляются легко и доступно для любого пользователя персонального компьютера.

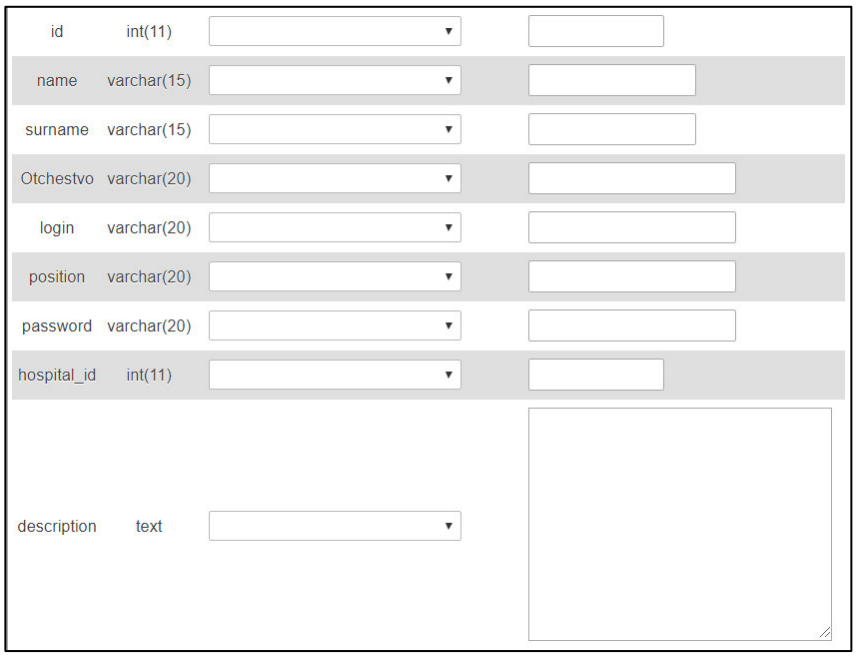

Рисунок 3.21 – Добавление врача

Таким же образом можно вносить и лекарства, удалять неактивных пользователей, изменять время визитов по просьбе клиентов.

После приема пациента врач может выписать необходимое лекарство, которое занесено в базу данных пациенту, используя его ИИН. После того, как врач введёт ИИН пациента, он увидит имя пациента, его адрес проживания, которые вносятся в базу данных при регистрации пользователя.

Это лекарство появится в личном кабинете пациента вместе с описанием, порядком его примеенения и его ценой. Также пациент сможет увидеть, в какой аптеке можно купить это лекарство и его дозировку.

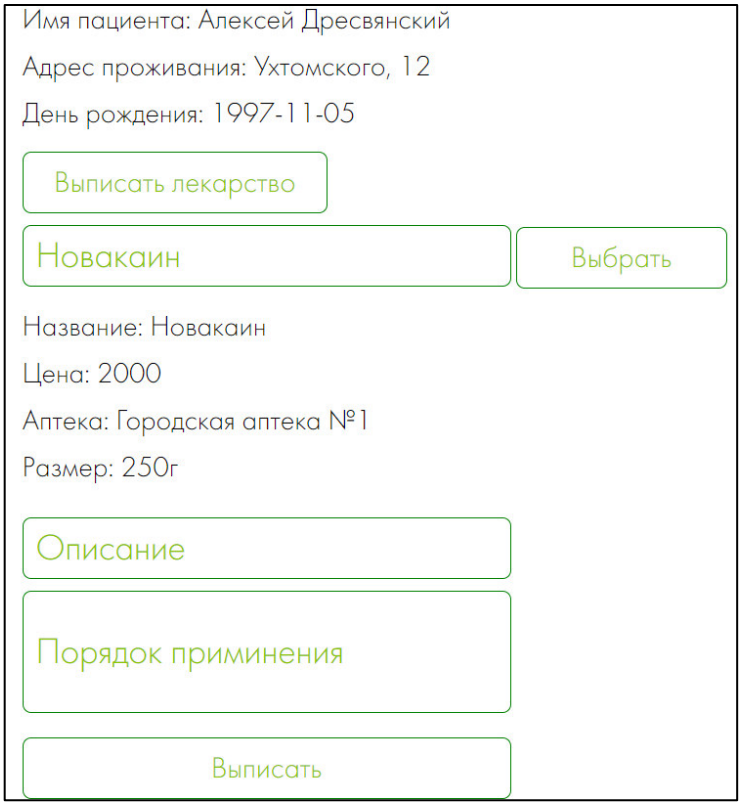

Рисунок 3.22 – Выписывание лекарства

Удобство данного программного обеспечения состоит в том, что таким же образом можно вносить и лекарства, удалять неактивных пользователей, изменять время визитов по просьбе клиентов.

После того, как врач выписывает пациенту лекарство, оно заносится в базу данных и появляется в выписках в личном кабинете пациента. Для того, чтоб посмотреть выписку пациент должен перейти во вкладку “мои выписки”. В личном кабинете это будет выглядеть таким образом:

Рисунок 3.23 – Выписка в личном кабинете пациента

Функция, которая реализует выдачу лекарства и обрабатывает ошибки:

```
app.post('/getdrugs', function(req,res){
    console.log(req.body)
    connection.query('select name, cost, apteka, size from drugs where name = "'+req.body.drug+'"', function (error, rows, fields) {
        if(!!error){
            console.log('Error in the query');
        } else
        {
            if(rows.length>0)
                {
                    res.json(rows);
            } else{
                res.json({message: 'Drug not found'});
            }
        }
        });
```

Рисунок 3.24 – Добавление врача

Для удобной работы с базой данных используется php my admin.

PhpMyAdmin (PMA) - это программный продукт для работы с базами данных, с большим количеством функций, которые представлены подробно в официальной документации проекта.

Есть три способа, чтобы получить доступ к PMA:

Как пользователь базы данных, ссылки на просмотр servername.timeweb.ru/pma/, где имя_сервера-это имя сервера, где находится.

Как пользователь базы данных.

Как пользователь общественного доступа ко всем базам данных учетной записи через ссылку phpMyAdmin в панели управления раздел "базы данных MySQL", когда переключатель"полный доступ" включена.

4 Расчет затрат на разработку ПП

Технико-экономическое обоснование

Темой дипломного проекта является «автоматизация работы медицинского предприятия».

Цель данного дипломного проекта заключается в разработке сайта, позволяющего медицинскому предприятию принимать заявки на вызов доктора на дом и записываться к необходимому врачу.

4.1 Определение сложности разработки ПО

Для определения трудоемкости разработки программного обеспечения необходимо поделить задачи на более простые и понять, сколько времени потребуется на разработку.

Таблица 1 – Этапы разработки ПО

Этапы разработки ПО	Вид работы	Трудоемкость, чел. час.
Этап 1	Определение задачи	20
Этап 2	Составление технического задания	20
Этап 3	Изучение похожих ПО	30
Этап 4	Разработка и составление дизайна для сайта	20
Этап 5	Разработка программной части	60
Этап 6	Создание БД	60
Этап 7	Тестирование ПО	20
Этап 9	Исправление недоработок	40
Итого: время на выполнение дипломного проекта		270

Продолжительность рабочего дня равна 8 часам. Поэтому, учитывая то, что итоговое время на выполнение дипломного проекта составляет 270 часов, то для реализации программного обеспечения потребуется 34 рабочих дня.

4.2 Расчет затрат на разработку ПО

Смета по программному обеспечению включает в себя следующие затраты:

- материальные затраты;
- затраты на оплату труда;
- социальный налог;
- амортизация основных фондов;
- прочие затраты.

Рассчитаем материальные затраты. Они делятся на основные и вспомогательные.

Таблица 2 – Затраты на материальные ресурсы

Наименование материала	Марка	Ед. измерения	Количество	Цена за ед. в тенге	Сумма в тенге
Бумага для офиса	International Paper	Упаковка	1	900	900
Тетрадь (96 листов)	Senner	Штук	1	200	200
Ручка	Mark Tres	Штук	2	150	300
Компьютерная мышь	BLOODY V8	Штук	1	6000	6000
Итого:					7400

Общая сумма материальных затрат ($З_м$) рассчитывается по формуле:

$$З_м = \sum P_i * Ц_i, \quad (4.1)$$

где P_i - расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ - цена за единицу i-го вида материального ресурса, тг;

i - вид материального ресурса;

n - количество видов материальных ресурсов.

Расчет затрат на необходимое оборудование и программное обеспечение производится по форме, приведенной в таблице 3.

Таблица 3 – Расчет затрат на оборудование и ПО, необходимое для проекта

Наименование материала	Марка	Ед. измерения	Количество	Цена за ед. в тенге	Сумма в тенге
Ноутбук	Xiaomi Mi Pro	Штук	1	450000	450000
Принтер	Acer	Штук	1	31540	31540
ОС	Windows 10	Штук	1	52394	52394
Итого:					533934

Зм = 7400 + 533934 = 541334 (тг)

Для реализации программного обеспечения необходимы материалы на сумму 541 334 тенге.

4.3 Расчет затрат на электроэнергию

Так как при разработке программного обеспечения нам потребуется ноутбук и принтер, то необходимо рассчитать необходимые затраты на электроэнергию.

Согласно таблице 1 для разработки программного продукта необходимо порядка 270 часов, теперь необходимо рассчитать стоимость электроэнергии, которая будет потрачена в течении 270 часов.

Расчет необходимой электроэнергии определяется по формуле:

$$З_{эл.эн.обор.} = \sum W * К_{исц} * S * T, \quad (4.3)$$

где W – потребляемая мощность, Вт;

$К_{исц}$ – коэффициент использования ($К_{исц}$ = 0,7..0,9);

T – время работы;

S – тариф (1кВт/ч = 18,32 тг).

Итоги по расчетам стоимости затрачиваемой электроэнергии представлены в таблице 4.

Таблица 4 – Затраты на электроэнергию

Наименование приборов	Паспортная мощность, кВт	Коэффициент мощности	Время работы оборудования, ч	Цена ЭЭ тг/кВтч	Сумма, тг.
Ноутбук	0,7	0,7	270	18,32	2423,74
Принтер	0,4	0,9	10	18,32	65,95
Итого:					2489,69

$З_{э\ ноутбук} = 0,7 \times 0,7 \times 270 \times 18,32 = 2423,74$ тенге

$З_{э\ принтер} = 0,4 \times 0,9 \times 10 \times 18.32 = 65,95$ тенге

$З_{э\ общие} = 2423,74 + 65,95 = 2489,69$ тенге

4.4 Расчет затрат на оплату труда

Для разработки программного обеспечения, как указывалось ранее, необходимо два работника:

- руководитель проекта – управление и корректировка рабочих процессов и времени разработчика;

- разработчик – разработка и тестирование программного обеспечения.

Сумму расходов на оплату труда можно рассчитать по следующей формуле:

$$З_{тр} = \sum ЧС_i * T_i \quad (4.5)$$

где $ЧС_i$ - часовая ставка i-го работника, тг;

T_i - трудоемкость разработки модели, чел.×ч; i - категория работника;

n - количество работников, занятых разработкой ПП.

Во время реализации проекта рабочее время участников не равномерно, поэтому имеет смысл установить часовую ставку каждого работника и общий объем заработной платы.

Часовую ставку сотрудника можно рассчитать по следующей формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \quad (4.6)$$

где $ЗП_i$ - месячная заработная плата i-го работника, тг;

$ФРВ_i$ - месячный фонд рабочего времени i-го работника, час.

Месячная заработная плата руководителя равняется 180 000 тенге и месячная заработная плата разработчика равняется 150 000 тенге. Рассчитаем часовую ставку каждого работника согласно формуле (4.6):

$$ЧС_{руководитель} = \frac{180\,000}{21 * 8} = 1071{,}43 тг/ч$$

$$ЧС_{разработчик} = \frac{150\,000}{21 * 8} = 892{,}86 тг/ч$$

Часовая ставка руководителя составляет 1071,43 (тг/ч), трудоемкость разработки равняется 120 часам. Часовая ставка разработчика составляет 892,86 (тг/ч), трудоемкость разработки равняется 270 часам. Согласно формуле (4.5) можно рассчитать сумму расходов на заработную плату работников:

$$З_{тр} = 1071{,}43 * 120 + 892{,}86 * 270 = 128571{,}6 + 241072{,}2 = 369643{,}8$$

Расчеты затрат по оплате труда показаны в таблице 5.

Таблица 5. – Расчет заработной платы

Категория работника	Квалификация	Трудоемкость разработки ПП, час.	Часовая ставка, тг/ч	Сумма, тг.
Руководитель	Инженер	120	1071,43	128 571
Разработчик	Программист	270	892,86	241072,2
Итого:				369943,8

4.5 Расчет затрат по социальному налогу

Согласно Налоговому кодексу Республики Казахстан социальный налог составляет 9,5% от фонда оплаты труда и мед страховки 1.5%. Итого 11% социальный налога. Социальный налог можно рассчитать по следующей формуле:

$$З_{по} = З_{тр} \times 0,1 = 369643,8 \times 0,1 = 36964,38$$

$$З_{тр}\,(\text{с уч п. о. }) = З_{тр} - З_{по} = \ 369643,8 - 36964,38 = 332\ 679,42$$

$$Н_{с} = З_{тр}(\text{с уч п. о. }) \times 0,11 = 332\ 679,42 \times 0,11 = 36594,74$$

4.6 Амортизация основных фондов и прочие затраты

Годовые нормы амортизации ОФ принимаются по налоговому кодексу РК или определяются, исходя из возможного срока полезного использования ОФ:

$$Н_{Ai} = \frac{100}{T_{Ni}}, \qquad (6)$$

где T_{Ni} - возможный срок использования i-го ОФ, год;

Возможный срок полезного использования ОФ примем: 5 лет для ноутбука и 7 лет для принтера. Эффективный фонда рабочего времени для компьютера и

$$З_{АМ1} = \frac{450000 \times 20 \times 270}{100 \times 1968} = 12347,56 \text{ тенге}$$

$$З_{АМ2} = \frac{31540 \times 20 \times 10}{100 \times 1968} = 32,05 \text{ тенге}$$

$$З_{АМобщие} = \ 12379,61 \text{ тенге}$$

Таблица 6 – Амортизация основных фондов

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тг
Ноутбук	450000	20	1968	270	12347,56
Принтер	31540	20	1968	10	32,05
ИТОГО					12379,61

На основе всех представленных расчетов необходимо оформить смету расходов на разработку ПО согласно форме, которая приведена в таблице

Таблица 7 – Смета затрат на разработку ПО

Статьи затрат	Сумма, тг
Затраты на оборудование	533934
Затраты на материальные ресурсы	7400
Затраты на оплату труда	369643,8
Социальные налоги	36594,74
Затраты на электроэнергию	2489,69
Амортизация основных фондов	12379,61
Итого по смете:	962441,84

Рисунок 1 - Диаграмма затрат

4.7 Определение возможной (договорной) цены ПО

Стоимость программного обеспечения определяется на основе качества разработанного продукта, сроков его разработки и производительности продукта. Стоимость $Ц_Д$ для программного обеспечения можно рассчитать по следующей формуле:

$$Ц_Д = З_{нир}\left(1 + \frac{P}{100}\right), \quad (4.9)$$

где $З_{нир}$ – затраты на разработку программного обеспечения, тг;

Р – средний уровень рентабельности ПО, (%). Данный параметр принят равным 25%.

$$Ц_Д = 962441{,}84 + 962441{,}84 * 0{,}25 = \ 1203052{,}3 \text{ тенге}$$

Далее необходимо определить стоимость реализации с учетом НДС, ставка НДС устанавливается законодательством РК. На 2019 года ставка НДС составляет 12%. Стоимость реализации учитывая НДС можно рассчитать по следующей формуле:

$$Ц_р = Ц_д + Ц_д * НДС, \quad (4.10)$$

$$Ц_р = 1203052{,}3 \ + 1203052{,}3 * 0{,}12 = 1347418{,}58 \text{ тенге}$$

Данную цену можно округлить до 1350000 тенге.

Таким образом, себестоимость программного продукта составляет - 962441,84;

Прибыль составляет – 240610,46;

Договорная цена - 1350000 .

5 Охрана труда и безопасность жизнедеятельности

Целью данного дипломного проекта является создание сайта, автоматизирующего работу медицинского предприятия.

5.1 Анализ условий труда

Рабочее место работников, сидящих в приемной должно обеспечивать комфортное положение для работы, обеспечивая высокую производительность труда. Работа над дипломным проектом велась в приемной, в которой находятся 3 работника больницы. Так как работа выполняется в приемной, то проблемы с шумом исключаются. В приемной установлен кондиционер, который обеспечивает необходимый воздухообмен и поддерживает необходимую для работы температуру. Из этого исходит, что в приемной основная работа ведется без проблем с шумом и вентиляцией. В приемной находятся 3 светильника L18 и имеется одно окно(3x1,5). Недостаток освещения неблагоприятно сказывается на комфорте работы работников медицинского центра и в этом разделе я буду приводить расчеты с целью решить проблему освещения в медицинском центре.

5.2 Расчет освещения

5.2.1 Расчет естественного освещения

Недостаточная освещенность и пониженная контрастность могут повлечь за собой напряженность зрительного анализатора, что в дальнейшем может привести к ухудшению зрения.

В данном случае, работа предполагает собой чтение, письмо и работу за компьютером, из этого следует, что освещенность, которая необходима для выполнения работ в данном помещении составляет 500 лк.

Необходимо просчитать площадь боковых световых проемов приемной, которая предназначена для создания нормируемой освещенности на рабочих местах.

Тип помещения – приемная.

Характеристики помещения:

длина L = 7 м, ширина B = 5 м, высота H = 3 м.

Высота рабочей поверхности над уровнем пола $h_{\text{рп}} = 0{,}8$, окна начинаются с высоты $h_{\text{рп}} = 0{,}8$ м, высота окна $h_0 = 1{,}5$м.

Рабочее помещение находится в IV часовом поясе – в г. Алматы (пояс светового климата – IV северной широты и южнее).

Рабочие места расположены в $1_{\text{рт}} = 0{,}5$ м, от наружной стены помещения, где проектируются оконные проемы. Минимальная освещенность будет в точке, отстоящей на расстояние 4 м от оконного проема.

Общую площадь окон S_0,м^2, можно определить по формулам 5.1 и 5.2

$$100\frac{S_0}{} = \frac{e_{\text{н}} \times \eta_0}{} \times k \quad \times k \tag{5.1}$$

$$S_0 = \frac{S_n \times e_{\text{ж}} \times \eta_0}{\tau_0 \times r_1} \times k_{\text{зд}} \times k_{\text{з}} \tag{5.2}$$

где S_n- площадь помещения; $S_n = 7 \times 5 = 35$ м^2;

$e_{\text{н}}$ – нормируемое значение КЕО;

$К_{\text{з}}$ –коэффициент запаса;

m_N - коэффициент светового климата;

Учитывая заданный световой пояс, ориентация световых проемов направлена на Север, определим по формуле 5.3.

$$e_x^{IV} = e_x \times m \times c \tag{5.3}$$

где m = 0,7;

c = 0,75 (в наружных стенах зданий);

e_n - 1,2 для работ высокой точности III разряда зрительной работы;

$$e_x^{IV} = 1{,}2 \times 0{,}7 \times 0{,}75 = 0{,}63$$

Учитывая тип помещения и находим коэффициент k_3 = 1,2 (учебные помещения, лаборатории, конструкторские бюро).

τ_0- общий коэффициент светопропускания,

равный $\tau_0 = \tau_1 \times \tau_2 \times \tau_3 \times \tau_4$;

$\tau_1 = 0{,}5$ (пустотелые стеклянные блоки);

$\tau_2 = 0{,}6$ (деревянные двойные раздельные переплеты);

τ_3 = 0,8 (железобетонные фермы и арки); τ_4 = 1 (регулируемые жалюзи и шторы);

η_0 – световая характеристика окон.

$$\eta 0 = 0{,}5 \cdot 0{,}6 \cdot 0{,}8 \cdot 1 = 0{,}24$$

Определим по формуле 5.4

$$L = B - 1$$

$$L = 5 - 1 = 4 \text{ м}$$

$$\frac{L}{l} = \frac{L}{B-1} = \frac{7}{4} = 1{,}75$$

$$h_{\text{расч}} = 0{,}8 + 1{,}5 - 0{,}8 = 1{,}5$$

$$\frac{B}{h_{\text{расч}}} = \frac{5}{1{,}5} = 3{,}33$$

По таблице 5.2 определим η_0 = 10,5.

r_1 – коэффициент, учитывающий повышение КЕО при боковом освещении благодаря свету, отраженному от поверхностей помещения и подстилающего слоя.

Средний коэффициент отражения в помещении ρср = 0,5, принимаем одностороннее боковое освещение.

$$\frac{B}{h_{\text{расч}}} = \frac{0{,}5}{4} = 0{,}125$$

Тогда r_1 = 1,05.

$k_{\text{зд}}$ – коэффициент, учитывающий затенение окон противостоящими зданиями.

Поскольку затеняющих зданий поблизости нет, $k_{\text{зд}}$ = 1. Вычислим общую площадь окон:

$$S_0 = \frac{35 \times 1{,}2 \times 10{,}5 \times 1 \times 0{,}63}{100 \times 0{,}24 \times 1{,}05} = 11{,}025 \text{ м}^2$$

Площадь световых проемов равна $S_{\text{сп}}$ = 11,025 м^2.

$$l_{ok} = \frac{S_o}{h_{ok}} = \frac{11{,}025}{3} = 3{,}625 \quad (5.5)$$

Исходя из этого, площадь световых проемов ($4{,}5 < 11{,}025\ м^2$) никак не обеспечивает необходимых условий труда на рабочих местах.

Так как основная цель – создание благоприятных условий труда в помещение с площадью S=35 м^2, исходя из этого совместно с естественным освещением используется искусственное освещение.

5.2.2 Расчет искусственного освещения

Разряд зрительной работы – V. Нормируемая освещённость – 500 лк. В качестве светильника был взят L18. Длина светильника 1514 мм, мощность – 65 ВТ. В помещении имеются 2 светильника.
Схема освещенности представлена на рисунке 1.

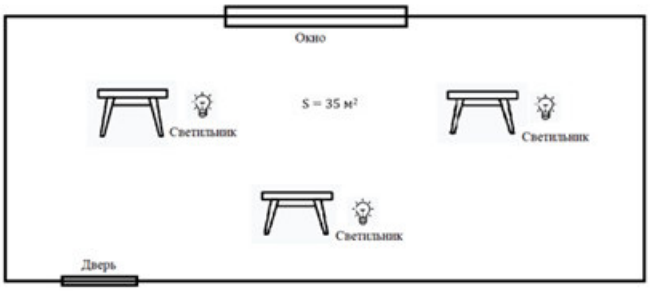

Рисунок 1 – Схема расчета освещенности

В таблице 1 изображены технические характеристики используемых светильников.

Таблица 8 - Технические характеристики ЛД65

Тип лампы	Мощность, вт	Напряжение, в	Световой поток, лм	Длина L, не более, мм	Диаметр D, мм	Тип цоколя
L18	65	110	4250	1514	38	G13d

Для начала идет проверка соответствия минимальной заданной освещенности при имеющихся 2 светильниках.

$$E = \frac{N \cdot n \cdot Ф_л \cdot \eta}{k_z \cdot S_{oc} \cdot Z}$$

Где S_{OC} −площадь помещения;

k_z − коэффициент запаса;

N – количество светильников;

Z −коэффициент неравномерности освещения, Z = 1,1;

n − количество ламп в светильнике;

$Ф_л$ − световой поток выбранной лампы, $Ф_л$ = 4250 лм

η − коэффициент использования, η = 65%

$$E = \frac{3 \cdot 2 \cdot 4250 \cdot 0{,}65}{1{,}3 \cdot 35 \cdot 1{,}1} \approx 331{,}17 \text{ лк}$$

При 2 светильниках минимальная освещенность равняется примерно 331,17 лк, что не соответствует условиям труда. В следствие чего возникает необходимость, увеличить количество светильников для обеспечения приемлемой освещенности. Для этого необходимо произвести расчеты для реконструкции:

Коэффициенты отражения от потолка стен и пола соответственно равны:

$$\rho_{пот} = 70\ \%$$

$$\rho_{ст} = 50\ \%$$

$$\rho_{пол} = 30\ \%$$

Вычислим высоту подвеса светильника над рабочей поверхностью по формуле 5.7:

$$H = h\ - h_p - h_c \qquad (5.7)$$

где: h_c – расстояние от светильника до перекрытия, h_c = 0,1 м;

h_p – высота рабочей поверхности над полом, h_p = 0,8 м;

h – высота помещения, h=3 м.

$$H = 3 - 0{,}1 - 0{,}8 = 2{,}1 \text{ м}$$

Расстояние между рядами светильников:

$$L_b = \lambda \cdot H$$

$$L_b = 1{,}2 \cdot 2{,}1 = 2{,}5\text{м}$$

Расстояние между светильниками:

$$L_a = L_b - 0{,}1 = 2{,}5 - 0{,}1 = 2{,}4 \text{ м}$$

Расстояние от стены до ближайшего светильника, когда работа у стены не проводится, определяем по формуле 5.9

$$l_1 = (0{,}4 \div 0{,}8) \cdot L \qquad (5.9)$$

$$l_1 = 0{,}6 \cdot 2{,}5 = 1{,}5 \text{ м}$$

Определяем индекс помещения по формуле 5.10

$$i = \frac{L \cdot B}{h_p \cdot (L + B)} \qquad (5.10)$$

$$i = \frac{4 \cdot 2}{2{,}175 \cdot (4 + 2)} = 1{,}319$$

Коэффициент использования в данном случае равен η = 65%, коэффициент запаса равен $k_з$ = 1,2

Определим количество люминесцентных ламп по формуле 5.11

$$N = \frac{E \cdot k_z \cdot S_{OC} \cdot Z}{n \cdot \Phi_л \cdot \eta} \qquad (5.11)$$

Где S_{OC} −площадь помещения;

k_z − коэффициент запаса;

E − заданная минимальная освещенность, Е = 500 лк;

Z −коэффициент неравномерности освещения, Z = 1,1;

n − количество ламп в светильнике;

$Ф_л$ − световой поток выбранной лампы, $Ф_л = 4250$ лм

η − коэффициент использования, $\eta = 65\%$

$$N = \frac{500 \times 1{,}5 \times 35 \times 1{,}1}{2 \times 4250 \times 0{,}65} \approx 5{,}22$$

Для создания нормируемой освещенности 500 лк необходимо 6 люминесцентных ламп серии L18, мощность каждой лампы должна быть не меньше 65 Вт, что соответствует действительности.

Для решения данной проблемы следует провести реконструкцию офиса и добавить 3 люминесцентных ламп серии L18 для обеспечения необходимого освещения в помещении. На рисунке 2 изображена приемная после реконструкции.

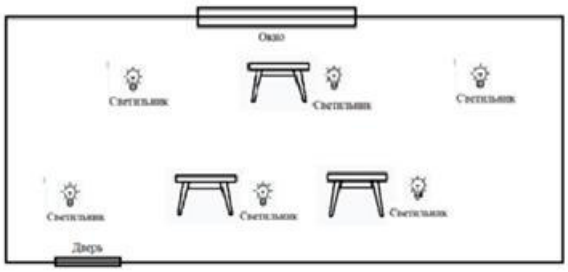

Рисунок 2 – Приемная после реконструкции

Заключение

В ходе выполнения дипломной проекта был разработан программный продукт, который автоматизирует работу медицинского предприятия.

Для успешного выполнения проекта были выполнены следующие задачи:

– проведено изучение предметной области на примере схожих приложений и сайтов;

– определены основные требования к программному продукту;

– сделаны выводы о решениях проблем других программных продуктов;

– при помощи таких технологий, как HTML, CSS, JavaScript и Angular была проведена разработка пользовательского интерфейса;

– спроектирована база данных, состоящая из 7 таблиц, предназначенная для хранения данных врачей и пациентов;

– также была изучена экономическая целесообразность разработки данного программного продукта;

– произведены работы по улучшению освещения в приемной больницы для обеспечения работникам более комфортных условий труда.

Таким образом, получен был полноценный актуальный, экономически эффективный программный продукт, готовый к использованию в любых медицинских учереждениях.

Список литературы

1 Гудман, Д. JavaScript и DHTML. Сборник рецептов. Для профессионалов / Д. Гудман. - М.: Питер, 2015. - 523 с.

2 Дакетт, Джон HTML и CSS. Разработка и дизайн веб-сайтов (+ CD-ROM) / Джон Дакетт. - М.: Эксмо, 2013. - 480 с.

3 Дакетт, Джон Основы веб-программирования с использованием HTML, XHTML и CSS / Джон Дакетт. - М.: Эксмо, 2013. - 768 с.

4 Дебольт HTML и CSS. Совместное использование / Дебольт, Вирджиния. - М.: НТ Пресс, 2013. - 512 с.

5 Дронов, В. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / В. Дронов. - М.: БХВ-Петербург, 2014. - 138 с.

6 Дронов, Владимир HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / Владимир Дронов. - М.: БХВ-Петербург, 2013. - 416 с.

7 Квинт, Игорь Создаем сайты с помощью HTML, XHTML и CSS / Игорь Квинт. - М.: Питер, 2014. - 448 с.

8 Квинт, Игорь Создаем сайты с помощью HTML, XHTML и CSS на 100% / Игорь Квинт. - М.: Питер, 2012. - 448 с.

9 Лазаро, Исси Коэн Полный справочник по HTML, CSS и JavaScript / Лазаро Исси Коэн, Джозеф Исси Коэн. - М.: ЭКОМ Паблишерз, 2014. - 938 с.

10 Мержевич, Влад HTML и CSS на примерах / Влад Мержевич. - М.: "БХВ-Петербург", 2012. - 448 с.

11 Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. - Москва: Машиностроение, 2016. - 688 с

12 Никсон, Робин Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript / Робин Никсон. - М.: Питер, 2013. - 496 с.

13 Никсон, Робин Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS / Робин Никсон. -М.: "Издательство "Питер", 2013.560 с

14 Пауэрс, Дэвид Adobe Dreamweaver, CSS, Ajax и PHP / Дэвид Пауэрс. - М.: БХВ-Петербург, 2012. - 829 с.

15 Прохоренок, Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н.А. Прохоренок, В.А. Дронов. - Москва: СПб. [и др.] : Питер, 2015. - 768 с.

16 Прохоренок, Николай HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера (+ CD-ROM) / Николай Прохоренок. - М.: БХВ-Петербург, 2012. - 912 с.

17 Пфаффенбергер HTML, XHTML и CSS. Библия пользователя / Пфаффенбергер и др. - М.: Вильямс; Издание 3-е, 2015. - 752 с.

18 Создай свой веб-сайт с помощью HTML и CSS. - М.: Питер, 2013. – 569с.

19 Ташков, Петр Веб-мастеринг HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка / Петр Ташков. - М.: Книга по Требованию, 2014. - 512 с.

20 Титтел, Эд HTML, XHTML и CSS для чайников / Эд Титтел , Джефф Ноубл. - М.: Диалектика, 2013. - 400 с.

Приложение А
(обязательное)

Техническое задание для разработки
Разработка автоматизированной системы приёма пациентов в медицинских центрах

1. Общие требования:

– наименование разрабатываемой системы:

– разработка автоматизированной системы приёма пациентов в медицинских центрах

– цель разработки:

– обеспечить возможность записи к нужному доктору в режиме онлайн;

– обеспечить возможность вызова доктора на дом;

– контролировать запись новых пациентов;

– удалять безактивных пациентов;

– создать личный кабинет пользователя;

– создать личный кабинет врача;

– добавить возможность выписки лекарства.

– предлагаемые технологии для разработки системы (на выбор разработчика):

– Nodepad++;

– Brackets.io;

– Блокнот;

– VisualStudio;

– WebStorm;

– Sublime Text;

– выбор архитектуры построения:

– Клиент-Сервер.

– предлагаемые языки и технологии программирования:

– HTML;

– CSS;

– JavaScript;

– MySQL;

– NodeJS;

– Bootstrap.

– общий объем программной части системы, Мб

– не более 100 Мб.

2. Технические требования:

– требования к программному обеспечению:

– быстрая скорость работы с информацией;

– SEO оптимизированность;

– создание ПО, доступного пользователю;

– код программы должен не быть загрязненным.

– тестирование программного обеспечения:

– проверка работоспособности программного обеспечения и всех функций сайта;

– выдавание ошибок при неправильном заполнении полей;

– тестирование системы на сбои.

3. Специфические требования

– адаптивность системы:

– сайт должен быть адаптивным под все девайсы и браузеры;

– обычный пользователь должеть быть в состоянии вручную добавлять новый персонал или лекарства.

4. Экономические требования:

– расчет стоимости системы и стоимости разработки программного обеспечения (подлежит обсуждению):

– стоимость готового продукта 1350000тг;

– стоимость разработки 1 000 000тг.

– потенциальные клиенты и области применения:

– частные больницы;

– врачи;

– пациенты.

Приложение Б
(обязательное)

Листинг программы

File Script.js

```
var express = require('express');
var mysql = require('mysql');
var bodyParser = require('body-parser');
var SessionStore = require('express-mysql-session');
var connect = require('connect');
var router = express.Router();
var cookie = require('cookie');
var cookieParser = require('cookie-parser');
var session = require('express-session');
var app = express();
var http = require('http');
var server = http.createServer(app);
var nodemailer = require('nodemailer');
var fs = require('fs');
var async = require('async');
var dateFormat = require('dateformat');
var code;

var io = require('socket.io').listen(server);

app.set('view engine', 'ejs');

app.use('/static', express.static('./static'));

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended:false}));
app.set('view engine', 'ejs');

var options = {
   host: 'localhost',
   user: 'root',
   password: '',
   database: 'test'
}
app.use(session({
   key     : 'session_name',
   secret  : 'session_secret',
```

```
  resave  : true,
   store: sessionStore,
   saveUninitialized: true ,
   cookie: {
  }
}));

var sessionStore = new SessionStore(options)

var connection = mysql.createConnection({
host: 'localhost',
user: 'root',
port: 3306,
password: '',
database: 'drugs'
});

connection.connect(function(error) {
if(error){
        console.log('Error');
} else{
        console.log('Connected');
}
});

var sess;

app.get('/',function (req, res, next) {
sess = req.session;
if(sess.name){
        res.render('index',{});
}else{
res.render('index',{});
}
});
app.post('/',function (req, res, next) {
sess = req.session;
if(sess.name){
        res.render('index',{});
}else{
res.render('index',{});
}
```

```
});
app.get('/cabinet',function (req, res, next) {
if(req.session.status){
sess = req.session;
if(sess.status=='doctor'){
        res.render('chat',{});
}else{
res.render('chat1',{});
}
}else{
res.render('index',{});
}
});


app.post('/registr',async function (req, res, next) {
var reqObj = req.body;
var insertSql = "Insert into patients SET ?";
var insertValues = {
        "login":reqObj.login,
        "name":reqObj.user_name,
        "surname":reqObj.user_surname,
        "password":reqObj.password,
        "address":reqObj.email,
        "iin":reqObj.iin,
        "birthday":reqObj.date
};
var query = connection.query('select login from patients where login =
"'+reqObj.login+'"', await function (error, rows, fields) {
        if(rows.length>0){
                res.status(418);
                return res.json({message:"Error with login"});
        }else{
        var query = connection.query(insertSql, insertValues, function (err) {
                if(err){
                        console.log('SQL error ' + err);
                }else {
                        res.status(200);
                        return res.json({message: "ok"})
                }
        })
        }
```

```
});
})
app.post('/loginu', function (req, res, next) {

connection.query('select id, name, surname, password from patients where login = "'+req.body.login+'"', function (error, rows, fields) {
        if(!!error){
                console.log('Error in the query');
        } else
        {
                if(rows.length>0)
            {
                console.log(rows);
               if(req.body.password == rows[0].password)
            {
             console.log("eqwe")
               req.session.name = rows[0].name;
               req.session.surname = rows[0].surname;
               req.session.pid = rows[0].id;
               req.session.status = 'patient';
               res.json({message: 'ok'});

               } else
               {
                  res.status(418);
                  res.json({message: 'Неверный пароль'});
               }
            } else
            {
               res.status(202);
               res.json({message: 'Данного пользователя не существует'});
            }
                }
        });
});

app.post('/logind', function (req, res, next) {
connection.query('select id, name, surname, password from clients where login = "'+req.body.login+'"', function (error, rows, fields) {
        if(!!error){
                console.log('Error in the query');
        } else
```

```
{
                    if(rows.length>0)
                {
                   if(req.body.password == rows[0].password)
                {
                   res.status(200);
                   req.session.name = req.body.login;
                   req.session.name = rows[0].name;
                   req.session.surname = rows[0].surname;
                   req.session.did = rows[0].id;
                   req.session.status = 'doctor';
                   res.json({message: 'ok'});
                   } else
                   {
                      res.status(418);
                      res.json({message: 'Неверный пароль'});
                   }
                } else
                {
                   res.status(202);
res.json({message: 'Данного пользователя не существует'});
                }
                    }
            });
    });

    app.post('/user', async function(req,res){
      sess = req.session;
      console.log(sess);
    if(sess.name) {
    return res.json({message: sess});
    } else {
       return res.status(202);
    }
    });

    app.post('/call', async function(req,res){
      sess = req.session;
    if(sess.name) {
    return res.json({message: "ok"});
    }
```

```
        });
        app.get('/logout', async function(req,res){
        var sid = req.session.id;
        req.session.destroy(function(err) {
          if(err) {
            console.log(err);
          } else {
                return res.json({message:"ok"});
                // io.sockets._events.sessreload(sid);
          }
        });
        });
        var socket;
        app.post('/join', function(req,res){
        var name = req.session;
        return res.json({name});
});
        app.post('/checkdate', function(req,res){
        var array1 = [8,9,10,11,12,13,14,15,16,17,18];
        var array2 = [];
        connection.query('select time from visits where date = "'+req.body.date+'" and
id_doc = "'+req.body.id+'"', function (error, rows, fields) {
                if(!!error){
                        console.log('Error in the query');
                } else
                {
                        if(rows.length>0)
                    {
                        for (var i=0; i < rows.length; i++) {
                        array2[i]=rows[i].time;
                        }
                       array1 = array1.filter( ( el ) => !array2.includes( el ) );
                                console.log(array1);
        res.json(array1);
                 } else{
                        res.json({message: 'Empty'});
                 }
                }
                });
        });
```

```
app.post('/getdoctors', function(req,res){
connection.query('select name, surname, Otchestvo, position, description, recenzia, img from Clients', function (error, rows, fields) {
        if(!!error){
                console.log('Error in the query');
        } else
        {
                if(rows.length>0)
           {
                        res.json(rows);
         } else{
                return res.json({message: "Error with doctor"})
         }
}
        });
});
app.post('/getdoctor', function(req,res){
connection.query('select name, surname, Otchestvo, id from Clients where position="'+req.body.spec+'"', function (error, rows, fields) {
        if(!!error){
                console.log('Error in the query');
        } else
        {
                if(rows.length>0)
           {
                        res.json({data:rows});
         } else{
                return res.json({message: "Error with doctor"})
         }
        }
        });
});
app.post('/zapis', function(req,res){
var reqObj = req.body;
console.log(reqObj);
var d_name;
var today = new Date();
var dd = today.getDate();
var mm = today.getMonth() + 1; //January is 0!
var yyyy = today.getFullYear();
        if (dd < 10) {
        dd = '0' + dd;
```

```
        }
                if (mm < 10) {
                        mm = '0' + mm;
                }
        today =  yyyy + '-' + mm + '-' + dd ;
        if(req.session.pid){
        if(reqObj.date>=today){
        var query = connection.query('select id from clients where id =
"'+reqObj.id_doc+'"', function (error, rows, fields) {
                if(!!error){
                        console.log('Error in the query');
                } else
                {
                        if(rows.length>0){
                        d_name=rows[0].id;
                        var insertSql = "Insert into visits SET ?";
                        var insertValues = {
                                "id_pat":req.session.pid,
                                "id_doc":d_name,
                                "time":reqObj.time,
                                "pricina":reqObj.pricina,
                                "date":reqObj.date
                                };
                        var query = connection.query('select time from visits where time
= "'+reqObj.time+'" and date = "'+reqObj.calendar+'"', function (error, rows, fields)
{
                                if(8<=reqObj.time && reqObj.time<=18){
                                if(rows.length==0){
                                        var query = connection.query(insertSql,
insertValues, function (err) {
                                                if(err){
                                                        console.log('SQL error ' + err);
                                                }else {
                                                        var myObject = {
                                                                "Тип":"zapis",
                                                           "name": req.session.name,
                                                           "Причина":req.body.pricina
                                                        }
                                                io.to(req.body.doctor).emit('zapis',myObject);
                                                res.status(200);
```

```
return res.json({message: "Data in database"})
                                        }
                                    })
                            } else{
                                    return res.json({message: "Error with time"})
                            }
                    } else{
return res.json({message: "Error with time(8-18)"})
                    }
                    })
                    }else{
                            return res.json({message: "Error with doctor"})
                    }
            }
});
}else{
        return res.json({message: "Error with time past"})
}
}else{
        return res.json({message: "Error with patient"})
}
});


app.post('/vizov', async function(req,res){
if(req.session.pid){
var reqObj = req.body;
console.log(reqObj);
var today = new Date();
var dd = today.getDate();
var mm = today.getMonth() + 1; //January is 0!
var yyyy = today.getFullYear();
if (dd < 10) {
  dd = '0' + dd;
}
if (mm < 10) {
  mm = '0' + mm;
}
today =  yyyy + '-' + mm + '-' + dd ;
var insertSql = "Insert into vizovy SET ?";
var insertValues = {
```

```
        "id_pat":req.session.pid,
        "id_doc":reqObj.id_doc,
        "number":reqObj.nomer,
        "pricina":reqObj.pricina,

"date":today
};
var query = connection.query(insertSql, insertValues, function (err) {
if(err){
        console.log('SQL error ' + err);
}else {
        var myObject = {
                "Тип":"zapis",
           "name": req.session.name,
           "Причина":req.body.pricina
        }
        io.to(req.body.doctor).emit('vizov',myObject);
        res.status(200);
        return res.json({message: "Data in database"})
}
```

Приложение В
(обязательное)

Акт внедрения сайта, автоматизирующего работу медицинского предприятия

Настоящий Акт свидетельствует, что сайт медицинского предприятия, разработанный Дресвянским Алексеем, внедрен в ИП «Черткова Н.О.».

Процесс внедрения проходил с 10 по 12 мая 2019 г.

Заявленные характеристики системы предполагали наличие следующих основных возможностей:

- Заполнение и редактирование данных;
- Функция записи на прием к врачу;
- Функция вызова врача на дом;
- Выписка лекарств.

В ходе опытной эксплуатации сайта подтверждено, что оно обладает всеми заявленными возможностями.

На момент подписания настоящего Акта система установлена и эксплуатируется сотрудниками данной компании. Программой пользуются 20 человек.

Начальник технического отдела