**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНСТИТУТ**

**МЕЖДУНАРОДНЫХ ОТНОШЕНИЙ (УНИВЕРСИТЕТ)**

**МИНИСТЕРСТВА ИНОСТРАННЫХ ДЕЛ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**ОДИНЦОВСКИЙ ФИЛИАЛ**

**Колледж МГИМО**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Тема Разработка информационной системы для автоматизации деятельности регистратуры в медицинском учреждении  

Выполнил Мыра Артём Валерьевич

(Фамилия Имя Отчество) (подпись)

Специальность Прикладная информатика Группа И-4.2

Руководитель Терпугова О.А.

(Фамилия И. О.) (подпись)

(место работы, должность, ученое звание, степень)

**К защите допустить**

Выпускная квалификационная работа выполнена с оценкой

Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Секретарь ГИА

(Фамилия И. О.) (подпись)

**Содержание**

[Введение 3](#_Toc73486640)

[1.Теоретическая часть 5](#_Toc73486641)

[1.1 Анализ предметной области 5](#_Toc73486642)

[1.2 Исследование существующих подходов к решению поставленных задач 6](#_Toc73486643)

[1.3 Описание среды разработки 9](#_Toc73486644)

[Вывод по теоретической части 13](#_Toc73486645)

[2.Практическая часть 15](#_Toc73486646)

[2.1 Разработка сайта 15](#_Toc73486647)

[2.2 Разработка базы данных 21](#_Toc73486648)

[2.3 Создание модуля регистрации пользователя 27](#_Toc73486649)

[2.4 Создание модуля авторизации пользователя 34](#_Toc73486650)

[2.5 Создание модуля регистрации пользователя на прием 39](#_Toc73486651)

[Вывод по практической части 44](#_Toc73486652)

[Заключение 45](#_Toc73486653)

[Список литературы 46](#_Toc73486654)

Введение

На сегодняшний день как никогда необходима автоматизация информационной системы в муниципальной поликлинике в связи с необходимостью работы с большими и постоянно растущими объемами информации при решении диагностических, терапевтических, статистических, управленческих и других задач.

Основное время, которое уходит на выполнение процесса приема пациентов, занимает не само решение каких-то клинических вопросов, а скорее больше сопроводительная работа, а именно оформление поликлинических талонов и иных видов отчетной документации, также записей в амбулаторной карте или истории болезни, а также назначений консультаций или обследований и тому подобное.

Наиболее эффективным инструментом для облегчения труда медицинских сотрудников и повышения его эффективности являются компьютерные технологии.

Автоматизация может не только облегчить работу, но и освободить персонал от рутины и предоставить им принципиально новый инструмент, который прямо или косвенно приведет к снижению неуместного потребления интеллектуального багажа, исполнению желания работать и заниматься медициной.

Первое знакомство посетителей с клиникой начинается на ресепшене. Это основное структурное подразделение для приема пациентов в поликлинику. Организация работы регистратуры во многом определяет ритм работы всех отделений поликлиники, чтобы обеспечить оптимальное распределение потока посетителей и сократить время посещения поликлиники.

Объектом исследования является автоматизация деятельности медицинского учреждения.

Предметом же в данном исследовании является автоматизация деятельности регистратуры медицинского учреждения, а конкретнее процесс записи на прием пациентов.

Целью данной работы является разработка программного комплекса для поликлиники, позволяющего повысить эффективность работы регистратуры за счет сокращения временных и трудозатрат, а также повышения качества работы.

Задачи:

* Проанализировать предметную область
* Проанализировать существующие проекты
* Разработать информационную систему
* Протестировать информационную систему

Актуальность выбранной задачи обусловлена ​​тем, что в настоящее время все более необходимо разработать информационную систему, которая обеспечивает лучшую видимость организации и деятельности и, следовательно, более удобна для последующей реализации. Актуальность подтверждается наличием нескольких систем для выполнения этих задач.

Разрабатываемая информационная система отличается от остальных тем, что форма для записи на прием находится на главной странице и нет необходимости переходов на другие страницы.

1. Теоретическая часть
   1. Анализ предметной области

Изначально регистрационный отдел больницы работал с использованием большого количества информации на материальных носителях, что не могло не тормозить работу отдела. База данных так же содержала информацию на материальных носителях, что значительно уменьшает скорость выполнения таких простых задач, как поиск, внесение, изменение информации в базе данных. Решить эти проблемы возможно с помощью программного модуля, который будет автоматизировать этот процесс.

Автоматизация – это процесс внедрения совокупности средств, которые минимизируют рутинные действия в процессе выполнения задачи. Автоматизация процесса будет осуществляться путем создания web интерфейса, который облегчит потенциальному клиенту больницы записаться на прием. Сотруднику больницы не придется вносить изменения в базу данных вручную, так как база данных будет находиться на электронном носителе и сама система зарегистрирует клиента. При таком подходе снижается нагрузка на регистрационный отдел больницы, а также упрощается и ускоряется сам процесс записи клиента.

В области практической реализации выпускной квалификационной работы лежит информационная система, реализованная в виде сайта, разработанного с помощью языка программирования PHP. Сайт состоит из следующих страниц:

– Главная страница

Главная страница отображает основную информацию о муниципальной больнице. Также присутствуют блоки с данными о количестве сотрудников больницы, количестве успешно выполненных операций.

Основной частью главной страницы является информационный блок, отвечающий за запись на прием пациента. Данный блок будет более подробно описан в практической части данной работы.

– Страница отделения

Данная страница содержит в себе все отделения, доступ к которым может предоставить больница.

– Страница врачи

Переходя на данную страницу, пользователь может увидеть список врачей, работающих в данном учреждении. Нажав на какого-либо врача, пользователь переходит на страницу с более подробной информацией об этом враче, в частности: возраст, образование, специальность и т.д.

Объединение информации в общее хранилище данных гарантирует целостность данных и возможность распределенного и одновременного доступа к ним. Создание базы данных также приведет к стабильной формализации данных и сокращению бумажного документооборота между отделами.

Экономическая эффективность работы следующая:

* Увеличение количества обслуживаемых пациентов;
* Сокращение сроков обработки медицинских карт;
* Уменьшение количества возможных ошибок.
  1. Исследование существующих подходов к решению поставленных задач

В области автоматизации процесса записи на прием немалую роль играет государство. Примером реализации данного процесса государством является сайт Госуслуги. Чтобы перейти к записи на прием, необходимо заполнить данные своего полиса, а затем выбрать нужного врача и время.

Страница с записей на прием изображена на рисунке (Рис. 1).

После ввода номера полиса и даты рождения система определяет участок пользователя, терапевта и поликлинику, в которой обслуживается данный участок, а также есть возможность либо вызвать врача на дом, либо записаться на прием в зависимости от отделения.

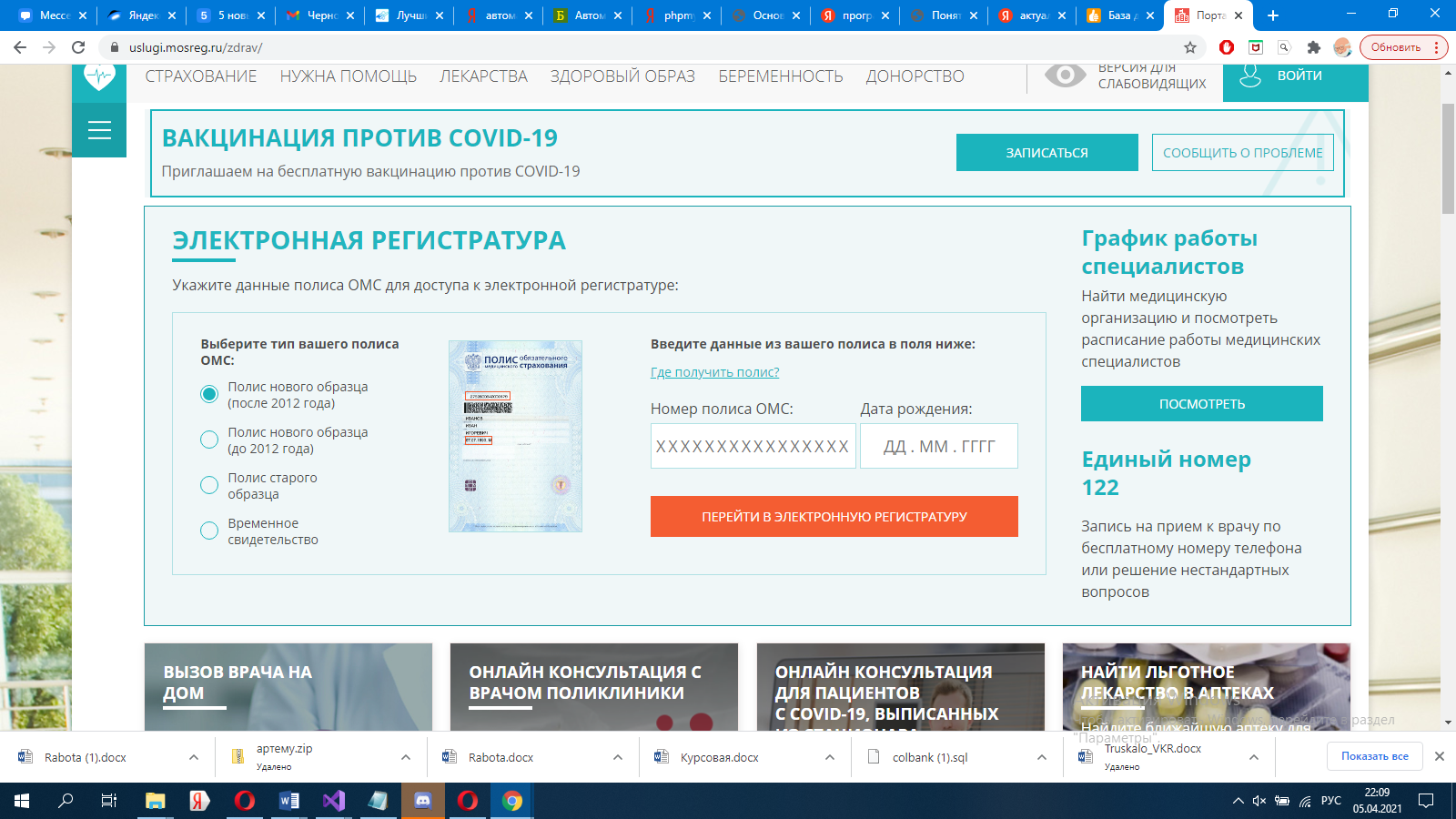


Рис. . Сайт Госуслуги

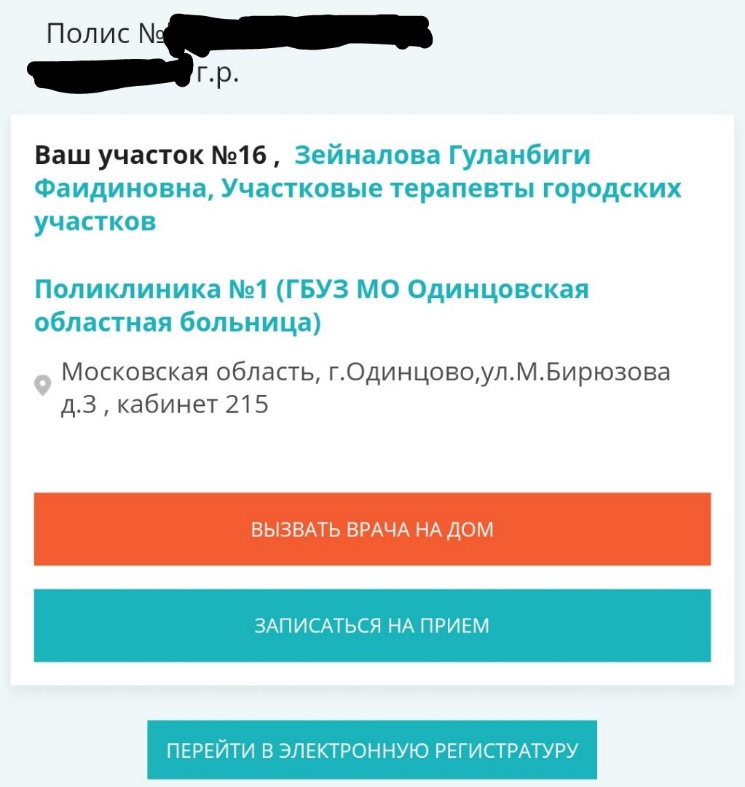


Рис. . Сайт Госуслуги после ввода

Главным минусом данной информационной системы является сессионное хранение данных, то есть при перезаходе на страницу необходимо вводить данные полиса, что крайне неудобно.

Также в деле автоматизации от государственного сектора не отстает и частный сектор. Тысячи частных поликлиник разрабатывают и администрируют свои сайты с возможностью удаленной записи на прием. Примером такого сайта можно назвать главный сайт сети частных поликлиник «Семейный доктор». На главной странице присутствует карта с отделениями, списком врачей и т.д. Также на данной странице присутствует кнопка «Записаться на прием», которая открывает форму с данными для записи на прием.

Главная страница изображена на рисунке (Рис. 3).

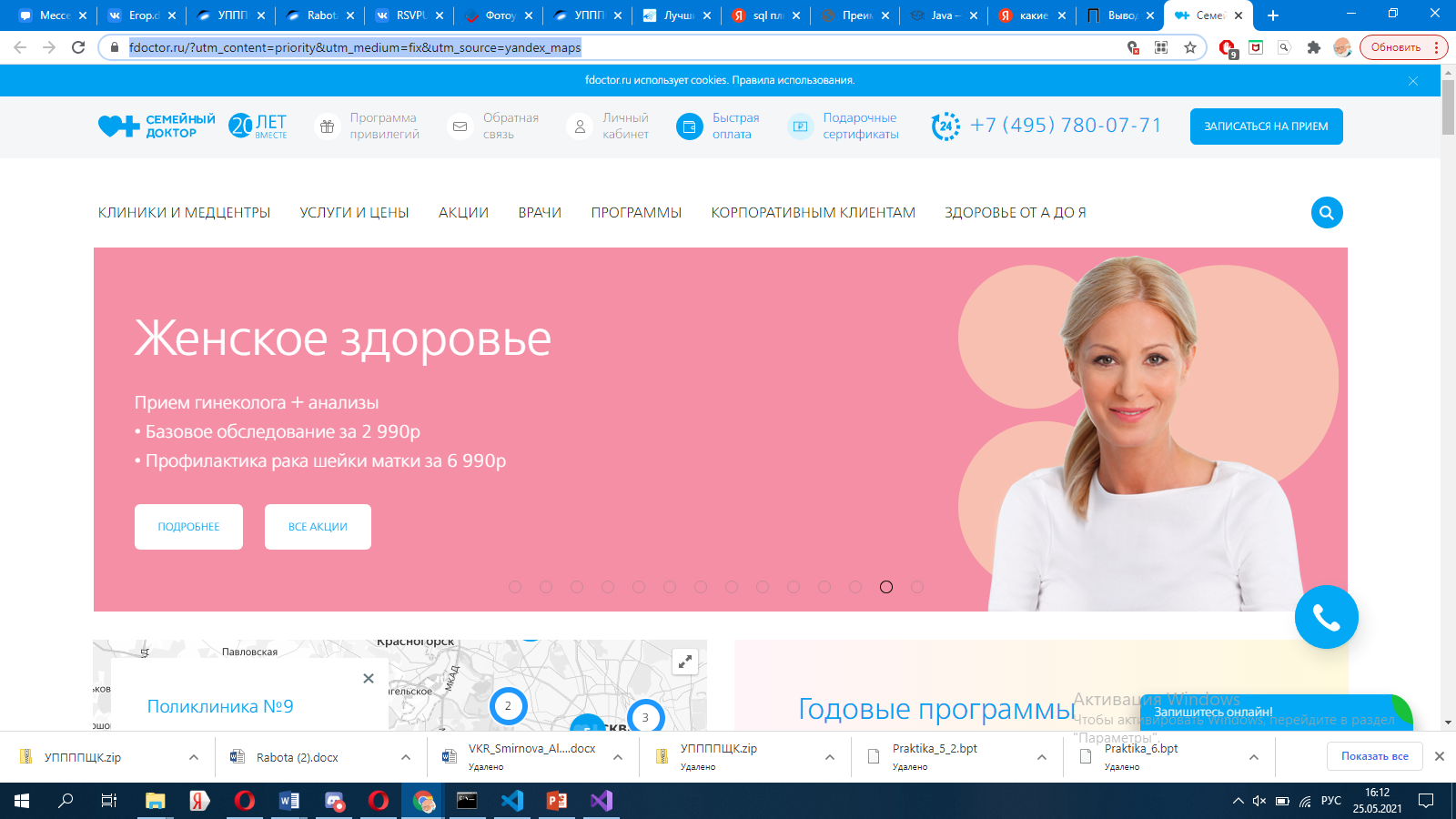


Рис. . Главная страница

Главная

Нажав на кнопку «Записаться на прием» пользователю открывается форма с данными для записи на прием. Сюда включены: Номер телефона, ФИО, специальность врача и желаемая дата приема. Также присутствует рекламный блок со скидками на отдельные процедуры.

Форма записи на прием изображена на рисунке (Рис. 4).

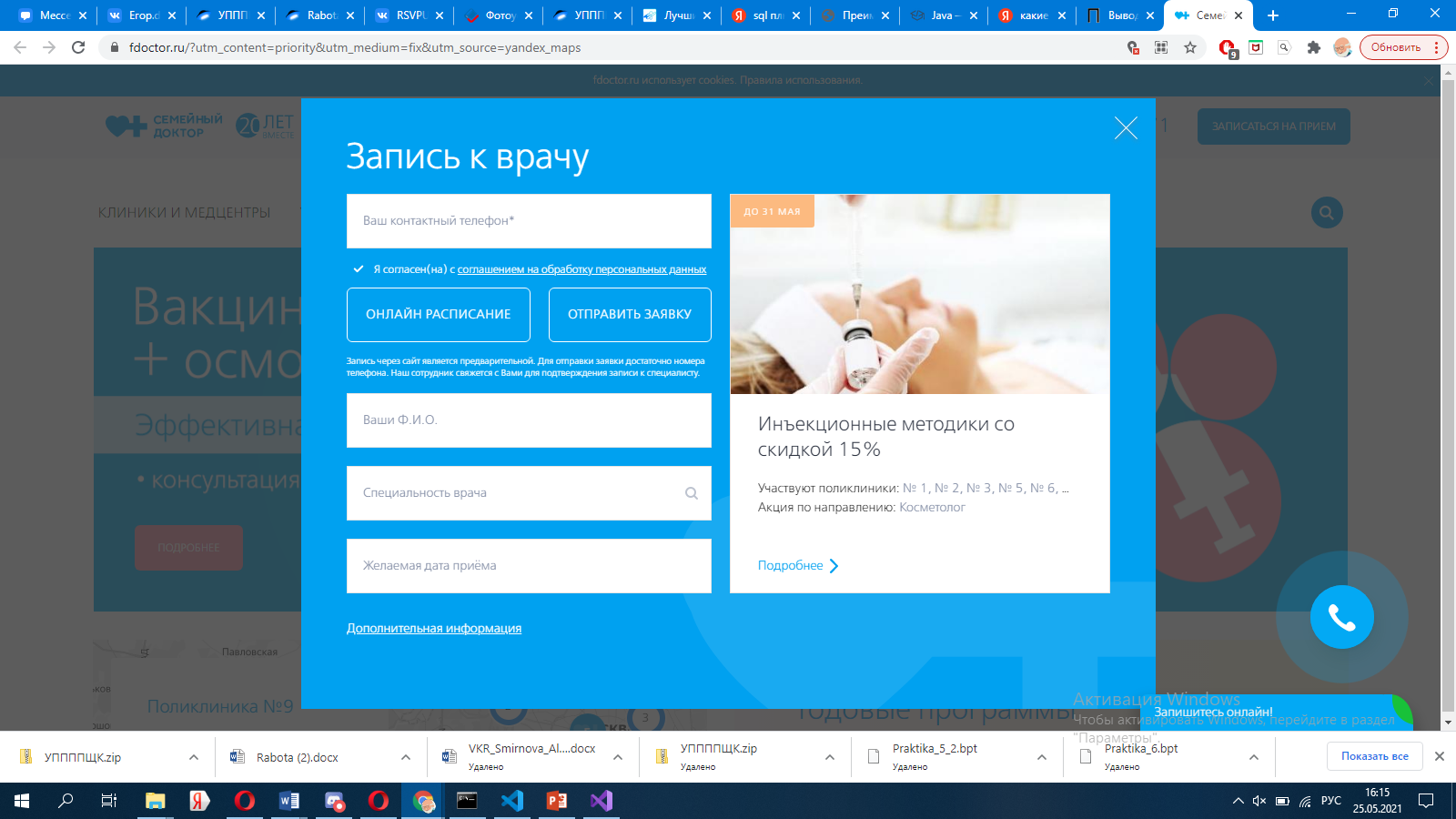


Рис. . Форма записи на прием

Одним из существенных минусов данной ИС является неудобный процесс ввода данных в виде всплывающей формы, занимающей большую площадь и блокирующую доступ к остальной части экрана. В случае, если случайно нажать вне формы, форма автоматически закроется и данные не сохранятся, что приведет к повторному вводу тех же данных.

* 1. Описание среды разработки

Разработка информационной системы проводилась в системе разработки Visual Code и на хостинге 000webhost.

Visual Studio Code – один из самых популярных редакторов кода, разработанный корпорацией Microsoft. Он распространяется бесплатно и поддерживается всеми текущими операционными системами: Windows, Linux и macOS. Данный редактор так же характеризуют как простой текстовый редактор, с возможностью подключения различных плагинов, что дает возможность работать со всеми видами языков программирования для разработки любого ИТ-продукта.[1]

Преимущества

* Позиционируется как «легкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Visual Studio Code бесплатен и разработан как программное обеспечение с открытым исходным кодом.
* Visual Studio Code позволяет разрабатывать как консольные, так и графические приложения, включая приложения с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-сервисы как в собственном, так и в управляемом коде для всех платформ.
* Редактор содержит встроенный отладчик, инструменты для работы с Git и инструменты для рефакторинга, навигации по коду, автозаполнения типовых конструкций и контекстных подсказок. Продукт поддерживает разработку для платформ ASP.NET и Node.js и считается легким решением, исключающим необходимость в полноценной среде IDE. Большой плюс редактора – поддержка большого количества языков, таких как C ++, C #, Python, PHP, JavaScript и других.[2]

Хостинг – это услуга, которая физически размещает страницы веб-сайта на сервере. Другими словами, это удаленный компьютер, работающий круглосуточно и без выходных, на котором хранится сайт, а также база данных. Даже простую посадочную страницу нужно где-то сохранить. Персональный компьютер не подходит для такой деятельности: он может потерять подключение к Интернету, а его производительность далеко не настолько, чтобы выдержать большой поток посетителей.

000webhost новый веб-хостинг разработан в 2007 году той же командой, которая разрабатывала сервисы Hostinger. 000webost был создан для предоставления бесплатного хостинга веб-сайтов с PHP, MySQL, Cpanel и без рекламы! Проще говоря, 000webost стремится предоставлять бесплатные услуги веб-хостинга.

000webhost предоставляет ведущие бесплатные услуги хостинга Cpanel с 2007 года. Все это стало возможным благодаря их платным услугам хостинга, включая обновления для тех, кому нужно больше места для роста.

Данный хостинг предоставляет самые новые возможности PHP вместе с их индивидуальными запатентованными бесплатными Easy Website Builder.

Таблица

Бесплатные услуги веб-хостинга

|  |  |
| --- | --- |
| Цены / подписки | бесплатно |
| Дисковое пространство | 1 GB |
| Пропускная способность | 10 GB |
| Веб-сайты разрешены | 2 |
| Панель управления | бесплатно |
| Ваша собственная реклама? | Разрешено бесплатно |
| Субдомена | Доступно |
| WordPress Хостинг | Доступно |
| Мест | 1 час в день |

Таблица

Детали

|  |  |
| --- | --- |
| Бесплатный сайт строитель | Доступно |
| Установщик WordPress | Доступно |
| Практически неограниченная поддержка PHP | Доступно |
| MySQL базы данных | Доступно |
| Мгновенное резервное копирование | Доступно |
| Поддержка FTP | Доступно |
| Веб-файловый менеджер | Доступно |
| Время безотказной работы 99% | Доступно |
| Мгновенная активация | Доступно |

Таблица

Поддерживаемые функции

|  |  |
| --- | --- |
| PHP и MySQL | Доступно |
| Работа в Крон | Доступно |
| Виться | Доступно |
| GD | Доступно |

Таблица

Функции панели управления

|  |  |
| --- | --- |
| Последняя Cpanel | Доступно |
| Бесплатный конструктор сайтов с сотнями бесплатных веб-шаблонов | Доступно |
| Автоустановщик WordPress | Доступно |
| Мастер начала работы | Доступно |
| Поддержка нескольких языков | Доступно |
| Защита от хотлинков | Доступно |
| Менеджер запрета IP | Доступно |
| URL перенаправления | Доступно |
| Каталоги, защищенные паролем | Доступно |
| Веб-файловый менеджер | Доступно |
| PHPMYADMIN | Доступно |

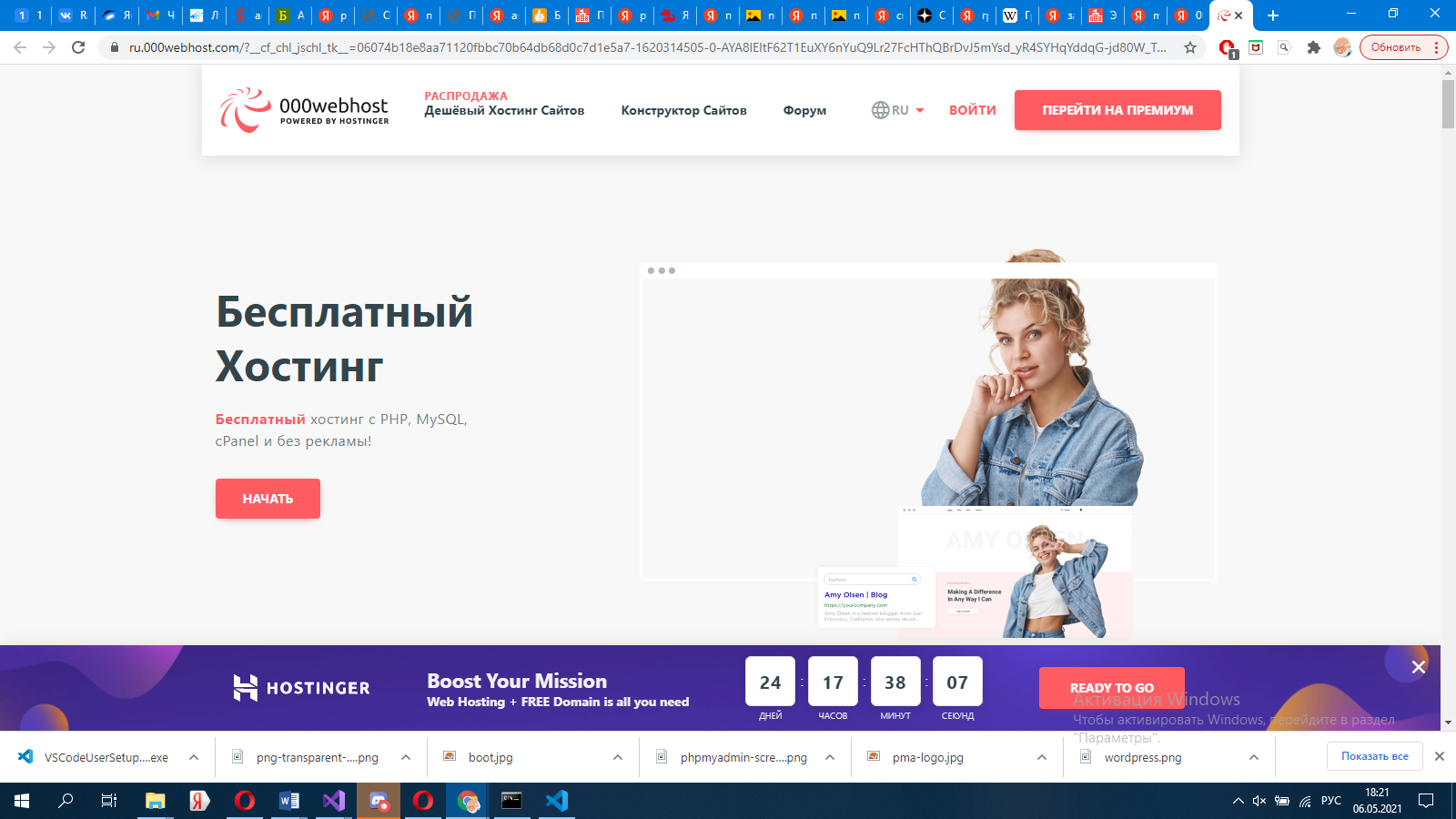


Рис. . Главная страница хостинга

Для создания информационной системы необходимо определить техническое обеспечение. IT отдел оборудован ноутбуками ASUS X570:

Основные характеристики ноутбука:

* Процессор: Core i5, 8250U
* Оперативная память: 16gb ddr4
* Постоянная память: 256gb ssd и 1tb hdd
* Видеокарта: NVIDIA GeForce GTX 1050
* Вес: 1.9 кг
* Дисплей: IPS 15.6 дюйма

Данные ноутбуки оснащены достаточно мощным процессором и видеокартой для работы. Вышеописанные ноутбуки не будут зависать при множествах параллельных процессах, так как оснащены большим количеством оперативной памяти и высокоскоростным SSD для быстрого запуска программ. Помимо данных преимуществ стоит упомянуть также и достаточно малый вес, что позволяет с легкостью портативно работать.

На ноутбуках установлена операционная система Windows 10 pro. Для удобной работы был установлен Visual Studio Code. VS Code включает в себя следующие плагины:

* HTML IntelliSense;
* PHP IntelliSense
* JS IntelliSense
* SQL IntelliSense
* CSS IntelliSense
* Live Sass Compiller

HTML IntelliSense – помощник для написания разметки в языке html. Визуально помогает разобраться в коде путем подсвечивания синтаксиса (структуры).

PHP IntelliSense – помощник для написания кода в языке php. Визуально помогает разобраться в коде путем подсвечивания синтаксиса, а так же позволяет выполнять автозаполнение и анализ кода.

JS IntelliSense – помощник для написания кода в языке php. Позволяет выполнять псевдо-код по мере ввода текста, таким образом intellisense сохраняет аккуратность при авто-дополнении.

SQL IntelliSense – помощник для написания запросов в языке sql.

CSS IntelliSense – помощник для написания стилей в языке css. По мере набора текста происходит выделение синтаксиса, а также предварительный просмотр цветов в контексте. Предложения содержат обширную документацию, включая список браузеров, поддерживающих свойство css.

Live Sass Compiller – это расширение, в интерактивном режиме, компилирует SCSS-файлы в CSS-файлы. Предварительный просмотр результатов применения скомпилированных стилей в браузере можно запустить кнопкой Watch my sass в панели задач.

Так же на компьютер установлен локальный сервер EasyPhp, который включает в себя:

* PHP
* Apache
* MySQL

PHP – скриптовый язык программирования общего назначения, который широко применяется для разработки веб-приложений.

Apache – Веб сервер.

MySQL – система управления реляционными базами данных с открытым исходным кодом.

Вывод по теоретической части

Опираясь на информацию, изложенную в теоретической части, можно сделать следующие выводы:

* Необходимо создавать уникальную информационную систему, ориентируясь на рассмотренные варианты реализации данной системы;
* Разработка дипломной работы будет проходить в среде разработки VS Code.
* Также информационная система будет загружена на хостинг 000webhost. Так как 000webhost обладает необходимым функционалом для разработки данной ИС.

1. Практическая часть
   1. Разработка сайта

Верстка сайта – это описание программным кодом визуальной части сайта. Другими словами – это создание и расположение всех элементов, находящихся на странице. К таких элементам относятся: заголовки, параграфы, блоки, фигуры, таблицы и многое другое.

На разрабатываемом сайте будет использоваться блочная верстка. Блочная верстка – это подход, с помощью которого сайт строят на основе блоков. Блочный элемент описывает тег div, который изначально задумывался универсальным, что помогает в индексировании сайта.

Любая страница сайта построена так, что изначально необходимо подключить все необходимые универсальные блоки. Блоки header и footer не изменяются ни на одной из страниц. Правильным решением будет создание подключаемого блока страницы, а далее использование его на страницах. При использовании этого решения есть возможность изменить код этих блоков в одном файле и изменения применятся на всех страницах сайта. Первым делом подключается шапка сайта (Рис. 6), далее создается блок с основным контентом страницы (Рис. 7). Как правило данный блок имеет тег div с классом «content». И в заключение подключается подвал сайта, то есть footer.



Рис. . Подключение динамического блока

На (Рис. 8) представлена структурная схема сайта, в которой учитывается вся специфика разрабатываемого сайта.

* На главной странице отображается основная информация о больнице, что позволяет быстро найти нужную информацию на сайте.
* Страница «Направления» содержит список всех направлений в больнице.



Рис. . Пример блока с контентом

* Страница «Информация о направлении» описывает выбранное направление, а так же выводит все отделения, работающие с текущем направлением и позволяет быстро перейти к отделению.
* Страница «Отделения» содержит список всех отделений в больнице
* Страница «Информация об отделении» описывает выбранное отделение, а так же выводит краткую информацию о заведующего отделением. К подробной информации так же можно перейти по ссылке на странице.
* Страница «Врачи» выводит список всех врачей больницы. Страница разделена на несколько вкладок, что позволяет не нагружать систему. В данном модуле присутствует сортировка врачей по отделениям, а так же поиск врача по ФИО.
* Страница «Информация о враче» содержит не конфиденциальную информацию о враче, которая может понадобиться пользователю.
* Страница «Контакты» содержит контактную информацию о больнице.

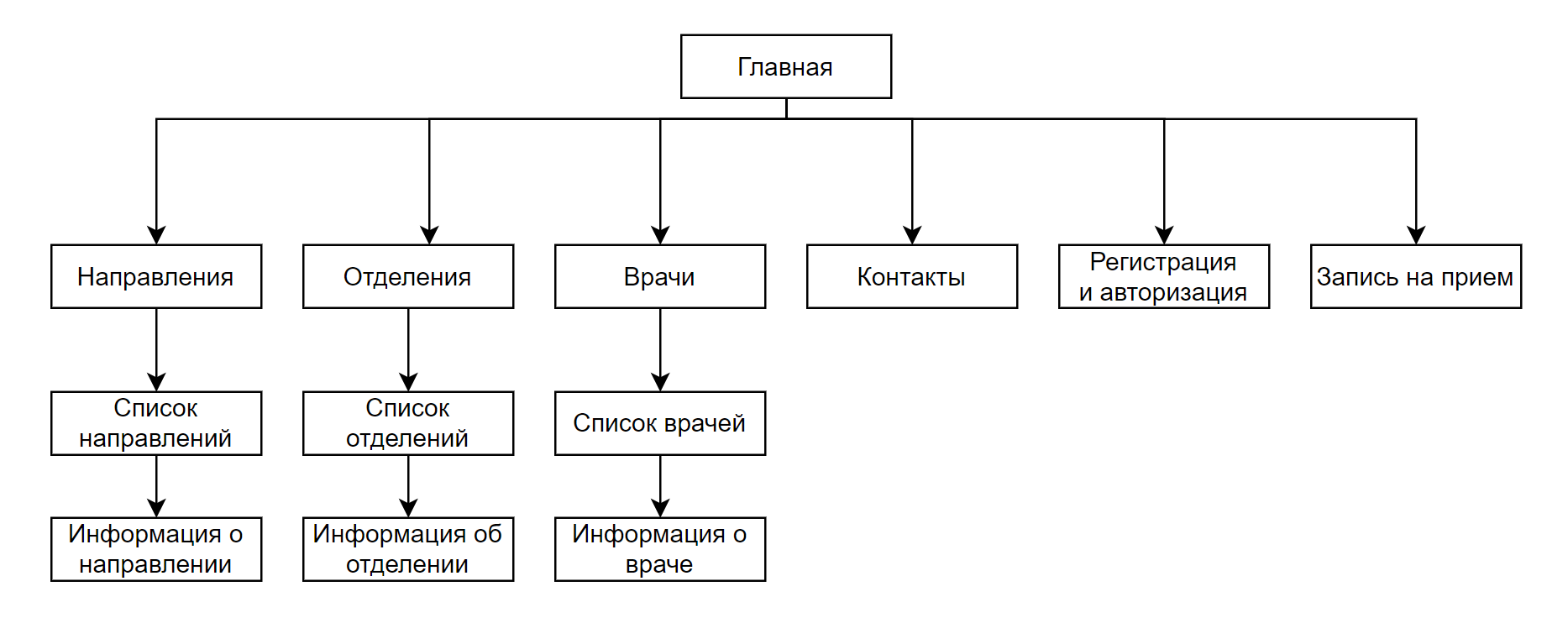


Рис. . Структурная схема сайта

В системе присутствует система разделения данных приложения MVC. MVC (Model-View-Controller «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо.

Зная определение можно понять, как устроен данный паттерн, а именно принцип работы будет состоять из следующих этапов:

1. Обработка запроса пользователя роутером системы.

2. Обработка данных контроллером страницы.

3. Получение данных через модели.

Первым этапом необходимо принять данные, отправленные пользователем, и задача роутера системы заключается в определении типа запроса. В информационной системе существуют следующие типы запросов:

* POST запросы
* GET запросы

Post запросы в основном понадобятся для передачи и принятия асинхронных данных сайта. «Get» запросы будут использоваться для большинства запросов. К «Get» запросам относятся и простая загрузка сайта.

После определения типа запроса он проходит следующие проверки:

* Существование контроллера страницы в списке ссылок
* Наличие параметров передаваемых в запросе

После прохождения всех проверок, роутером вызывается метод из запрашиваемого контроллера с передачей обработанных параметров.

Следующим этапом необходимо обработать данные контроллером конкретной страницы. В данной системе для каждой группы страниц создается отдельный контроллер. Группа состоит из страниц, которые работают с одной и той же категорией данных. Контроллеру необходимо подключить необходимые библиотеки и выполнить все проверки данных для исключения ошибок и выдать результат. При возникновении необходимости в получении данных, контроллер должен обратиться к соответствующей модели для получения данных.

Модель представляет из себя данные и условия, необходимые для работы с данными, которые отображают управляющую концепцию приложения. Модель передает контроллеру данные, которые были запрошены.

В модели содержится более важная часть логики, которая обрабатывает данные полученные от контроллера. Модель определяет какие данные будут записаны в базу данных и какие будут получены из нее и впоследствии вернутся контроллеру. Контроллер содержит основную логику для самого приложения.

Последним этапом работы контроллера является создание ответа, в который входит код HTTP-ответа, а так же само сообщение формате JSON, завершающим этапом является отправка его клиенту.

JSON (JavaScript Object Notation, объектная нотация JavaScript) – формат представления структурированных данных, используемый для передачи данных через Интернет.

Сервер сайта настроен так, что все запросы перенаправляются на файл, в котором происходит базовая настройка сервера, а так же подключение всех необходимых компонентов. На изображении (Рис. 9) показаны базовые настройки сервера, а именно включение или отключение показа ошибок сервера, подключение компонента router, который выбирает какой контроллер будет обрабатывать запрос и вызывает метод нужной страницы.

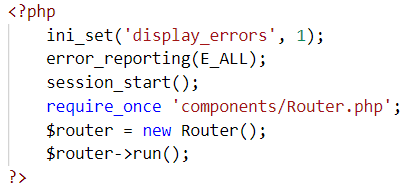


Рис. . Настройки сервера

Пример контроллера главной страницы (Рис. 10). На начальном этапе работы метода контроллера происходит подключение всех нужных для работы объектов. Для получения данных для страниц необходимо подключить модель, для работы с определенными данными.

Модель представляет данные и методы работы с запросами в базу данных, а так же производит проверку на валидность. Модель не знает ничего о контроллере, а лишь работает с данными, просто предоставляя доступ и управление ими.

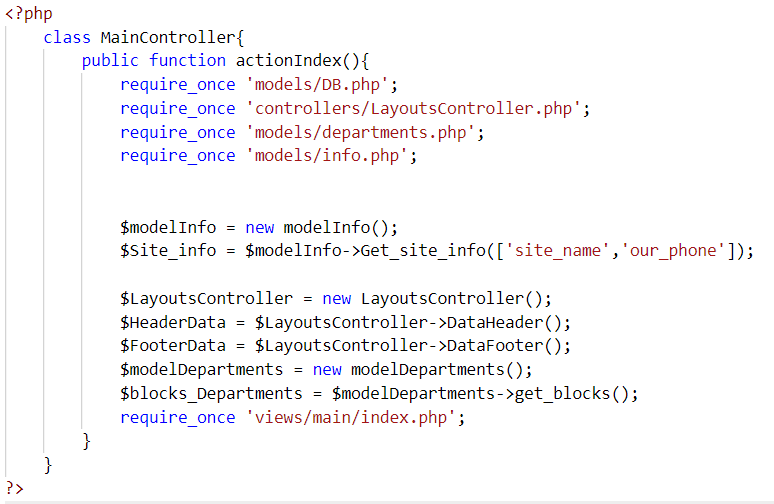


Рис. . Пример контроллера

Пример работы модели, связанной с получением данных об основной информации сайта (Рис. 11). В примере работы модели данных представлен метод получения данных с таблицы с информацией о больнице. В ходе работы метода происходит подключение модели работы с базой данных, совершаются проверки передаваемых параметров и в итоге выполняется запрос к базе данных. Результатом данного метода являются отсортированные данные.



Рис. . Модель данных

Последним этапом является представление (Визуализация). Представление отвечает за получение данных из модели или контроллера и отправляет их пользователю. Представление никак не обрабатывает полученные данные, а лишь визуализирует их. В примере (Рис. 12) показано получение данных из контроллера и отображение их на странице.



Рис. . Визуализация данных

Обрабатывание запроса на сервере происходит в функции, которая определяет дальнейший метод обработки данных на сервере (Рис. 13). При отсутствии метода для обработки запроса, функция вернет ошибку, где будет указаны необходимые для разработчика данные.



Рис. . Серверная функция распределения запросов

* 1. Разработка базы данных

Для начала данного подраздела необходимо описать само понятие базы данных.

База данных (БД) – совокупность данных, организованных по определённым правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ. Эти данные относятся к определённой предметной области и организованы таким образом, что могут быть использованы для решения многих задач многими пользователями.

В разрабатываемой системе используется реляционная база данных MySQL. Использование базы данных позволит в дальнейшем быстро изменять все данные на сайте не прибегая к вызову специалистов работы с сайтами.

MySQL — это система управления реляционными базами данных. Где «реляционные» значит, что данные хранятся в виде таблиц.

База данных на сайте будет содержать следующие таблицы:

1. Info
2. General\_menu
3. Directions
4. Departments
5. Doctors
6. Register\_appoinments
7. Users

Таблица «info» (Рис. 14), необходима для хранения информационных данных больницы.

В таблицу входят следующие поля:

1. id – уникальный идентификатор записи в таблице «info»
2. name – поле, которое служит для удобного поиска записи в таблице
3. value – поле, которое содержит значение конкретной записи в таблице

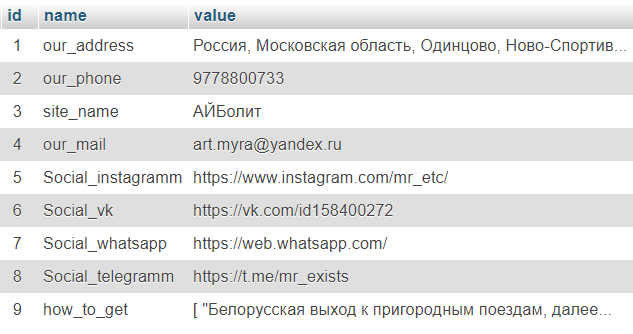


Рис. . Таблица "Info"

Таблица «General\_menu» необходима для хранения данных о меню сайта (Рис. 15).

В таблицу входят следующие поля:

1. id – уникальный идентификатор записи в таблице «General\_menu»
2. name – название пункта меню
3. data – содержит данные для пункта меню. Данные в этом пункте представлены в виде строки JSON, которая может содержать следующие параметры:

* Path – обязательный параметр, который указывает путь для пункта меню
* Subitems\_from\_base – необязательный параметр, который указывает на подпункты меню из базы данных.
* Subitems\_from\_list – необязательный параметр, который указывает на подпункты меню. Все пути для данного параметра указываются вручную.

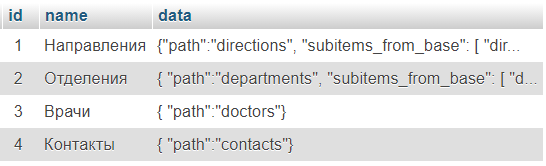


Рис. . Таблица "general\_menu"

Таблица «Directions» (Рис. 16), необходима для хранения данных о направлениях, существующих в больнице.

В таблицу входят следующие поля:

1. id – уникальный идентификатор записи в таблице «Directions»
2. name – название направления
3. descrioption – содержит описание направления. Данные представлены в виде строки JSON.
4. departments – Содержит JSON список уникальных идентификаторов полей таблицы с отделениями больницы.

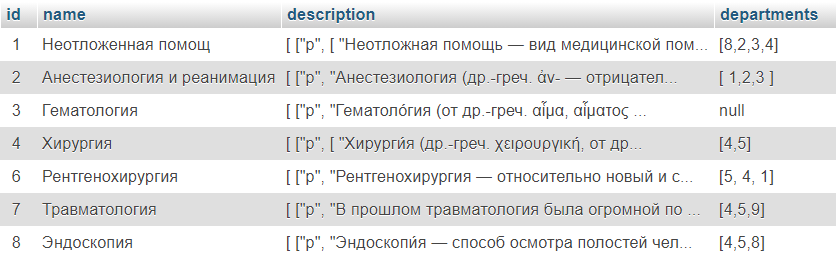


Рис. . Таблица "Directions"

Таблица «Departments» (Рис. 17), необходима для хранения данных о отделениях, существующих в больнице.

В таблицу входят следующие поля:

1. id – уникальный идентификатор записи в таблице «Departments»
2. name
3. department\_head
4. short\_description
5. description



Рис. . Таблица "Departments"

Таблица «Doctors» (Рис. 18), необходима для хранения данных о докторах, работающих в больнице.

В таблицу входят следующие поля:

1. id – уникальный идентификатор записи в таблице «Doctors»
2. name – имя врача
3. surname – фамилия врача
4. lastname – отчество врача
5. sex – пол врача
6. birthdate – дата рождения врача
7. phone – номер телефона врача
8. education – образование врача
9. graduation\_date – дата окончания учебного заведения
10. cabinet – личный кабинет врача
11. departments\_id – уникальный идентификатор поля таблицы с отделениями больницы, к которому относится врач
12. password – пароль для доступа врача в систему. Данные этого поля хранятся в зашифрованном виде. Шифрование происходит md5 алгоритмом, который вычисляет 128-битный хеш для произвольного набора данных, поступающих на его вход.
13. photo – содержит путь до фотографии врача

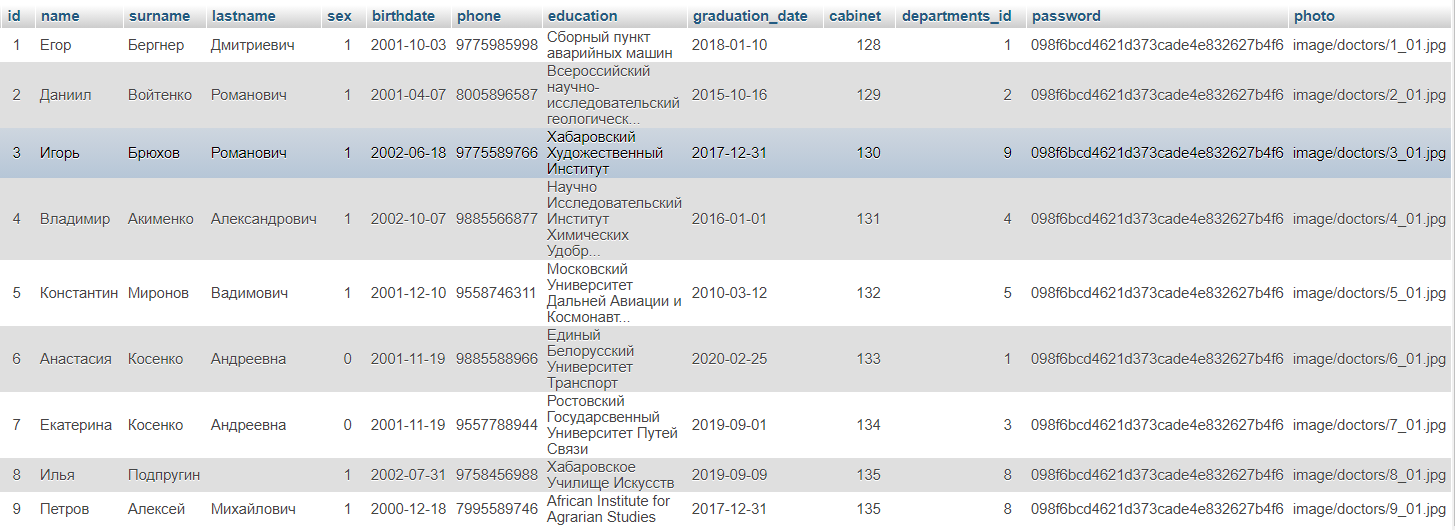


Рис. . Таблица "Doctors"

Таблица «Register\_appoinments» (Рис. 19) используется для хранения данных о регистрациях пациентов на прием.

В таблицу входят следующие поля:

1. id – уникальный идентификатор записи в таблице «Register\_appoinments»
2. id\_doctor – уникальный идентификатор поля таблицы с докторами больницы
3. id\_user – уникальный идентификатор поля таблицы с пользователями больницы
4. status – идентификатор, определяющий статус выполнения приема
5. date – дата записи на прием



Рис. . Таблица "Register\_appoinments"

Таблица «Users» (Рис. 20) используется для хранения данных о клиентах больницы.

В таблицу входят следующие поля:

1. id – уникальный идентификатор записи в таблице «Users»
2. name – имя пациента
3. surname – фамилия пациента
4. lastname – отчество пациента
5. birthdate – дата рждения пациента
6. sex – пол пациента. Принимает значение типа bool. Логическое 0 этого поля означает женский пол, а логическое 1 — мужской пол
7. mail – адрес электронной почты пациента
8. phone – номер телефона пациента
9. snils – номер СНИЛС пациента
10. password – пароль для входа в систему. Пароль в базе данных хранится в зашифрованном виде.



Рис. . Таблица «Users»

Все запросы к базе данных будут выполняться через модель «DB». В этой модели содержатся следующие методы:

* Connect – используется для создания соединения с базой данных
* Close\_Connect – используется для закрытия соединения с базой данных
* Query – используется для отправки запроса к базе данных
* Sort\_Result – используется для сортировки ответа от базы данных
  1. Создание модуля регистрации пользователя

Регистрация пользователя на сайте – это действия, которые направлены на создание личной учетной записи на сайте, с целью получения расширенного функционала на нем. Регистрация пользователя упрощает в дальнейшем работу с самим сайтом, так как внесенные данные в дальнейшем не надо будет вводить, так как они будут заноситься автоматически системой. Все данные пользователей хранятся в отдельной таблице на сервере, доступ к которому открыт только для администратора сайта.

Для регистрации пользователя была создана отдельная страница, доступ к которой можно получить перейдя по ссылке (Рис. 21) со страницы авторизации пользователя.



Рис. . Ссылка на страницу регистрации пользователя

После перехода по ссылке пользователь попадает на страницу регистрации (Рис. 22).

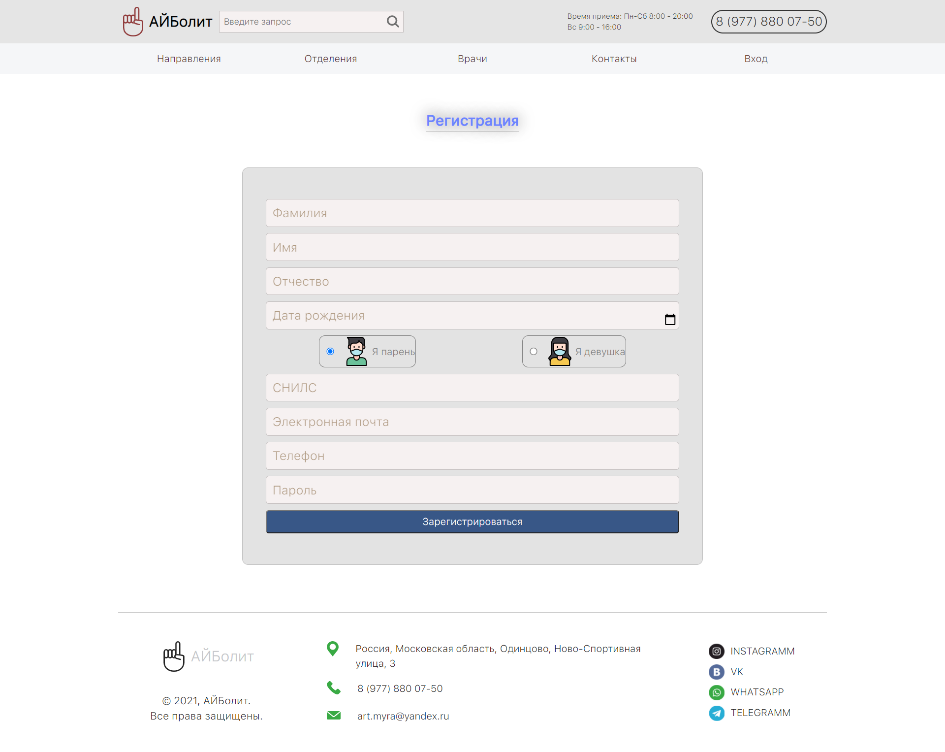


Рис. . Страница регистрации пользователя

На странице регистрации (Рис. 22) присутствует форма заполнения необходимых для регистрации данных. Для работы вышеуказанной формы необходимо подключить файлы со скриптами.

На (Рис. 23) показано подключение необходимых скриптов для обработки данных пользователя, а именно:

* JQuery – это кроссплатформенная JavaScript-библиотека, предназначенная для упрощения работы с языком разметки HTML.
* Notifications – это написанная самостоятельно библиотека для отображения пользователю уведомлений.
* RegisterUser – это основной скрипт для работы в форме регистрации.

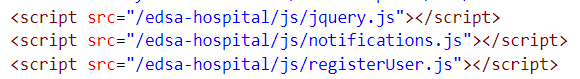


Рис. . Подключение скриптов для работы формы регистрации

Notifications.js – данная библиотека разработана для показа уведомлений пользователю. На (Рис. 24) отображен весь код данного скрипта.

В функцию «CreateNotify» (Рис. 24) передаются два параметра:

* Header – данный параметр определяет заголовок уведомления.
* Text – данный параметр содержит текст уведомления.

После вызова данной функции с указанием вышеуказанных параметров выполняется следующие действия:

* Добавляется объект уведомления в список HTML уведомлений.
* Выполняется функция плавного показа уведомления.
* Запускается обратный отсчет для показа уведомления. По истечению таймера уведомление удаляется.

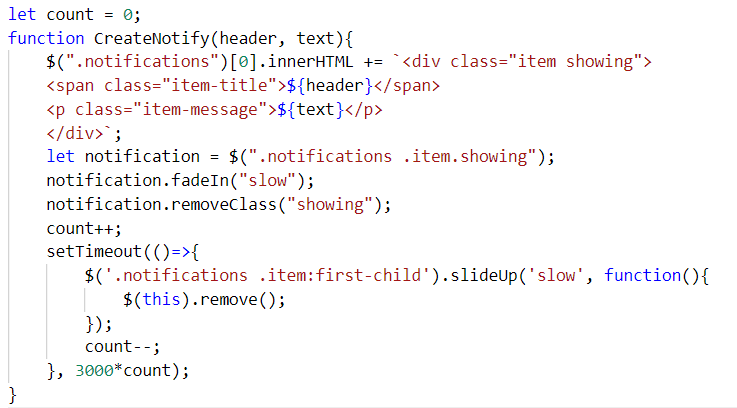


Рис. . Содержимое "Notifications.js"

Принцип работы достаточно простой. В HTML изначально встроен специальный блок для уведомлений. Так же для данного блока описаны стили, что делает восприятие уведомлений пользователю более приятным. Пример уведомления показан на (Рис. 25).

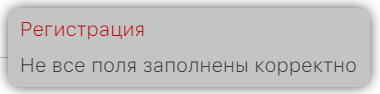


Рис. . Интерфейс уведомления

RegisterUser.js – основной скрипт, в котором происходит инициализация всех компонентов (Рис. 26) и обрабатываются действия на форме регистрации. Функция инициализации компонентов необходима для подключения, активации, подготовки к работе всех компонентов модуля. При инициализации включается маска для ввода определенных полей формы, создаются функции для обработки наведения на поле ввода и обработка самого нажатия на кнопки в форме. Функции, обрабатывающие события наведения на поля используются для визуального оформления поля. Изменения поля в статичном и активном состоянии можно увидеть на   
(Рис. 27) и (Рис. 28).

Маска ввода — это строка символов, указывающая формат допустимых значений входных данных.



Рис. . Функция инициализации компонентов

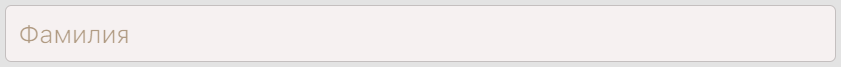


Рис. . Поле для ввода в статичном состоянии

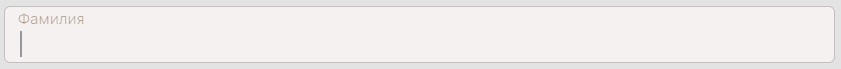


Рис. . Поле для ввода в активном состоянии

При нажатии на кнопку вызывается обработчик событий   
(Рис. 26 строка 18), в котором вызывается функция проверки полей на валидность введенных данных (Рис. 29).

Функция «checkFields» (Рис. 29) принимает в качестве параметра массив с объектами проверяемых полей. Процесс проверки производится путем перебора элементов вышеуказанного массива и проверки этих самых элементов через специальные регулярные выражения. Данная проверка предназначена для ускорения работы системы, т.к. система обрабатывает данных на стороне клиента, что исключает излишнее выполнение запросов к серверу. По завершению, вышеописанная функция возвращает id элементов, которые не прошли проверку на валидность.

Регулярные выражения – язык поиска и манипуляций с подстроками в тексте, основанный на использовании метасимволов. Для поиска используется паттерн, то есть строка-образец, состоящая из символов и метасимволов задающего правило поиска.

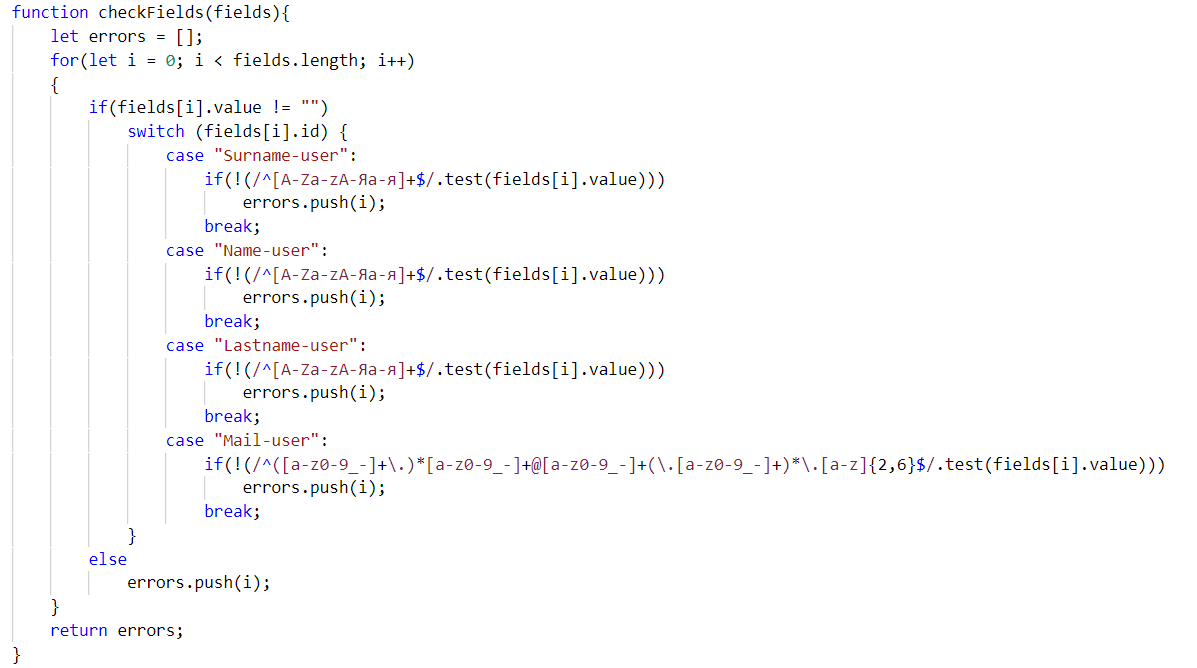


Рис. . Функция проверки валидности данных

После успешной проверки всех полей, вызывается функция отправки данных на сервер (Рис. 30).



Рис. . Функция отправки данных на сервер

В ходе работы функции (Рис. 30) выполняются следующие действия:

* Собираются значения всех полей
* Создается запрос к серверу
* Выполняется асинхронный запрос

Запрос к серверу выполняется при помощи функции «ajax», которая выполняется асинхронно посылает POST запрос. После выполнения запроса вызывается функция, которая проверяет код ответа сервера, а после уже и сам ответ.

Функция обработки данных на сервере (Рис. 31). Данная функция проводит проверку всех полей, что исключает ввод некорректных данных пользователем. Так как у пользователя есть два уникальных поля, а именно номер телефона пользователя и его СНИЛС. В этом случае необходимо проверить существование пользователя со схожими данными. При успешной проверки данных, функция вызывает метод из модели, которая предназначена для работы с данными из таблицы пользователей. После успешного выполнения метода, пользователю возвращается ответ от сервера с текущим статусом выполнения запроса.

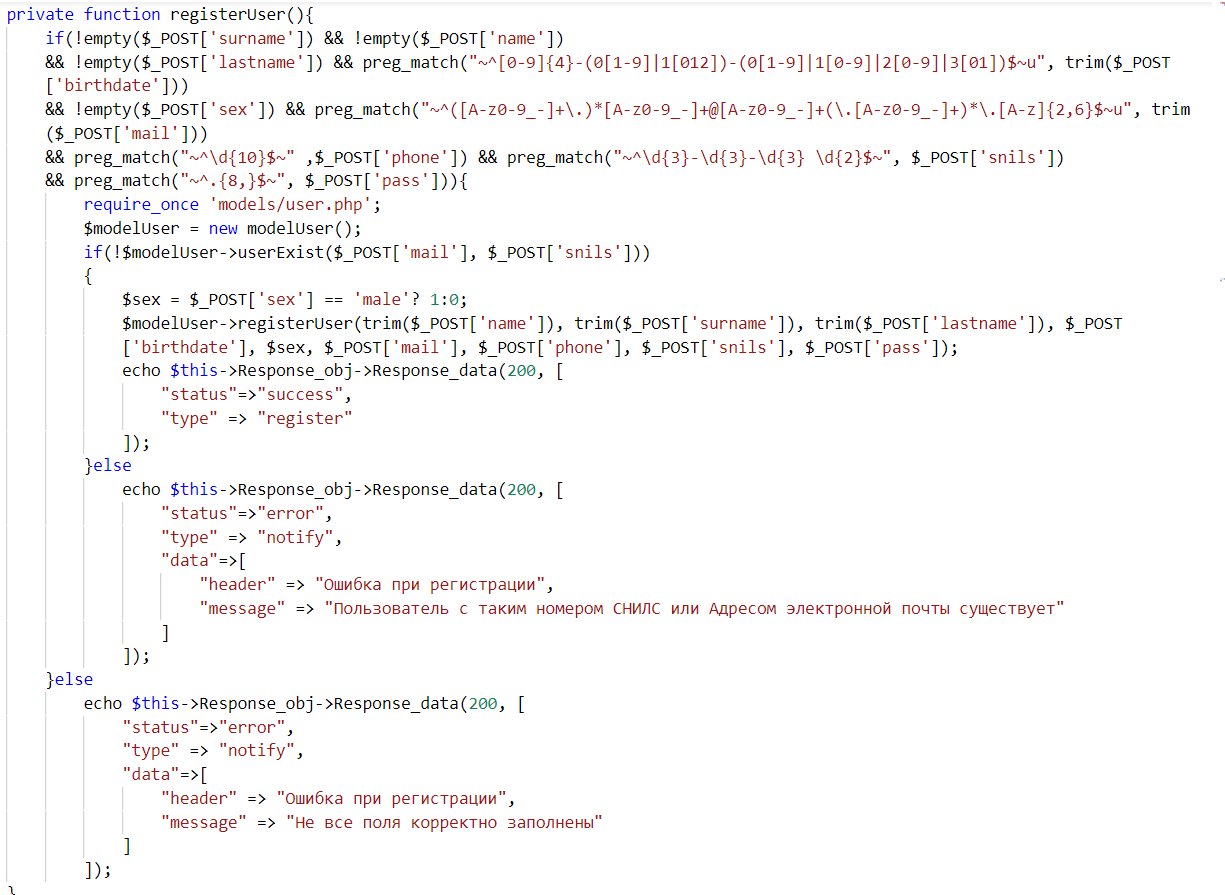


Рис. . Серверная функция обработки регистрации пользователя

Функция занесения данных пользователя представляет из себя обычный запрос к базе данных, который шифрует пароль и отправляет сам sql запрос. В результате выполнения функции мы получаем логический ответ true или false. Подобный ответ отображает успешность выполнения запроса.

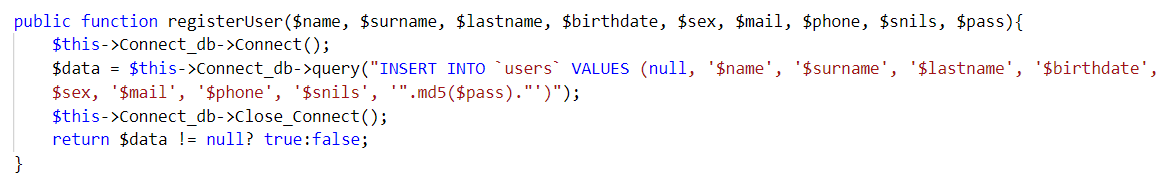


Рис. . Функция добавления данных пользователя в базу данных

Результатом выполнения модуля является зарегистрированный пользователь.

* 1. Создание модуля авторизации пользователя

Авторизация – это процесс определения прав доступа пользователю или группе лиц на выполнение определенных действий. Авторизация на сайте позволяет войти в свою учетную запись и выполнять операции от своего имени.

Для авторизации пользователя была создана отдельная страница, доступ к которой можно получить перейдя по ссылке с любой страницы сайта. Ссылка находится в шапке сайта (Рис. 33), что упрощает поиск страницы авторизации.

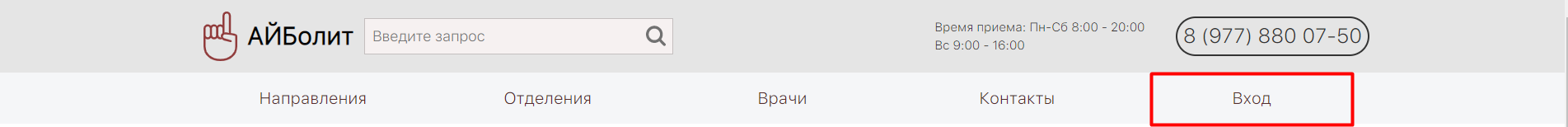


Рис. 33. Расположение ссылки на страницу авторизации

После перехода по ссылке, пользователь попадает на страницу авторизации (Рис. 34). На данной странице присутствует форма, состоящая из двух полей, кнопки и ссылки на страницу регистрации. Для работы данной страницы необходимо подключить те же скрипты, которые были описаны в главе. При прохождении авторизации пользователю необходимо заполнить поле номер телефона и пароль. После введения корректных данных необходимо нажать на кнопку «Войти», после чего начнет свое выполнение обработчик события нажатой кнопки (Рис. 35).

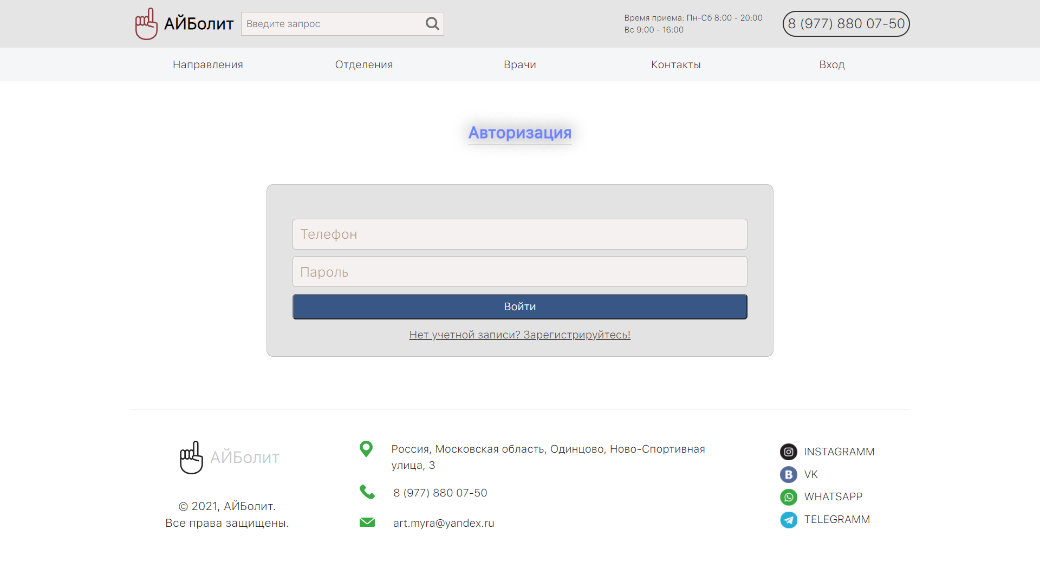


Рис. . Страница авторизации

В ходе обработки нажатия на кнопку происходит проверка на валидность полей формы (Рис. 36).

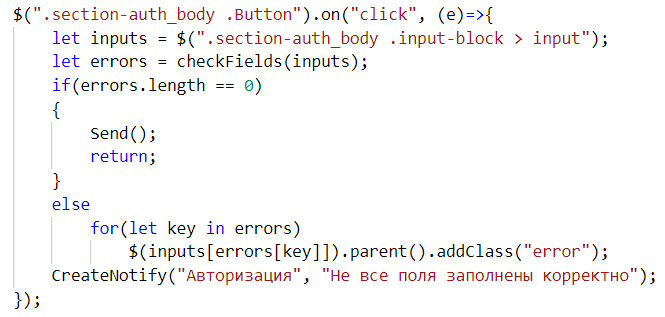


Рис. . Обработчик нажатия на кнопку "Войти"

Основной задачей проверки полей является выявление полей, содержание которых не соответствуют по критериям проверки регулярным выражением. При неудачном результате, уникальный id поля заносится в список не прошедших проверку полей, после чего эти поля подсвечиваются красным цветом, что указывает пользователю на конкретное поле для проверки (Рис. 37).

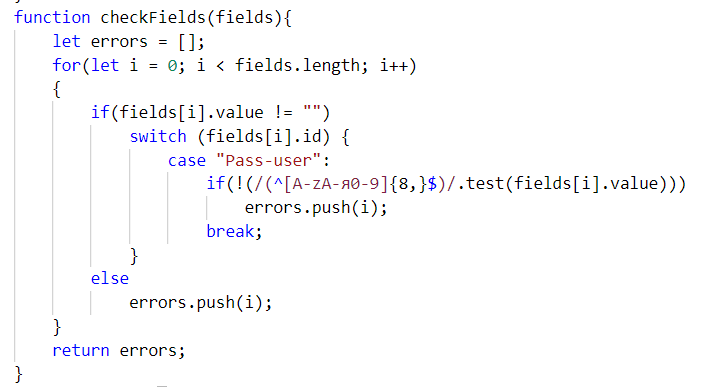


Рис. . Проверка полей формы

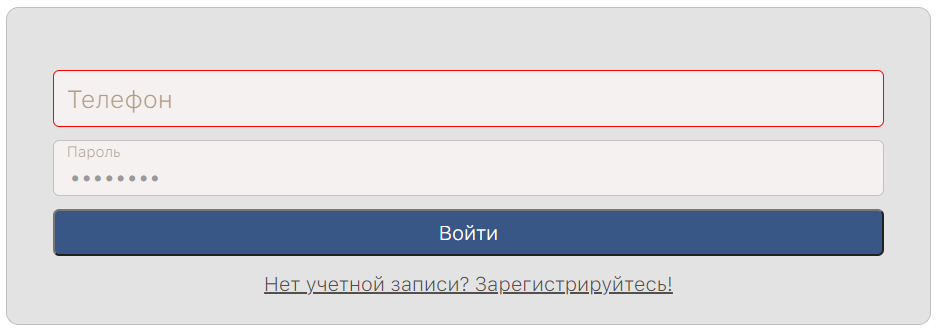


Рис. . Пример отображения ошибки при некорректном заполнении поля

После прохождения всех проверок, вызывается функция обработки авторизации (Рис. 38). В ходе выполнения данной функции происходит:

* Обработка введенных пользователем данных.
* Формирование на основе обработанных данных запроса.
* Отправка запроса на сервер.
* Обработка ответа от сервера.



Рис. . Функция обработки авторизации

Обработка запроса на сервере начинается с выполнения соответствующий функции (Рис. 39). Данная функция проводит проверку передаваемы полей, что ограждает нас от множества ошибок. Номер телефона состоит из 10 цифр а пароль из любых букв и цифр русского и английского языка. Пароль должен содержать минимум 8 символов. При несоблюдении хотя бы одного условия, пользователю вернется ошибка, а выполнение скрипта на этом прекратится. В случае успешного прохождения вышеописанных проверок, вызывается сама функция проверки существования пользователя с такими данными авторизации (Рис. 40).



Рис. . Функция обработки запроса авторизации на сервере

В ходе выполнения авторизации совершается запрос к базе данных, в котором проверяется существование пользователя с указанным номером телефона и пароля. Для проверки пароля происходит шифрование переданного пользователем пароля и сравнение с находящимся в базе данных. При нахождении совпадения, в сессию пользователя вносится персональный id. Данный id нам понадобится для выполнения всех запросов связанных с аккаунтом. При успешно выполнении всех действий, функция возвращает положительный результат, а пользователю возвращается соответствующее сообщение.

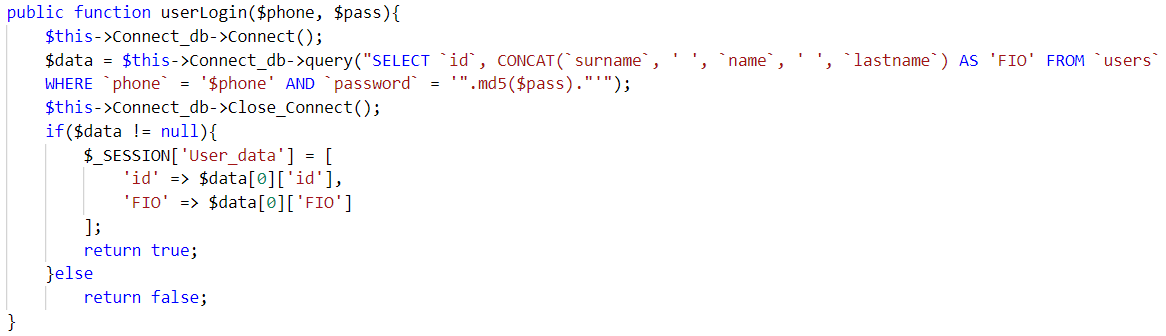


Рис. . Функция проверки данных авторизации

* 1. Создание модуля регистрации пользователя на прием

Визуальная часть очень важна, ведь с ней придется взаимодействовать пользователю. Интерфейс должен быть прост и понятен, так как чем он проще для восприятия, тем больше вероятность успешной регистрации.

Регистрация на прием возможна всем зарегистрированным пользователям. Для начала регистрации, пользователю необходимо открыть главную страницу и нажать на кнопку «Записаться на прием» (Рис. 41).

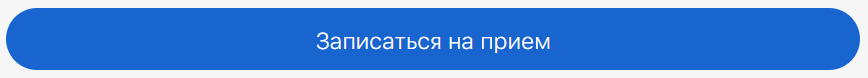


Рис. . Кнопка открытия формы записи на прием

После нажатия на соответствующую кнопку откроется форма записи на прием пользователя (Рис. 42). На форме содержится два выпадающих списка с перечислением всех отделений и врачей относящихся к нему. Таблица с датой для записи выполнена в простом и понятным для пользователя виде. Недоступные для записи даты имеют схожий фон с основной таблицей. Для доступных дат регистрации использован специальный стиль, который выделяется на основном фоне формы, что упрощает выбор даты для приема.

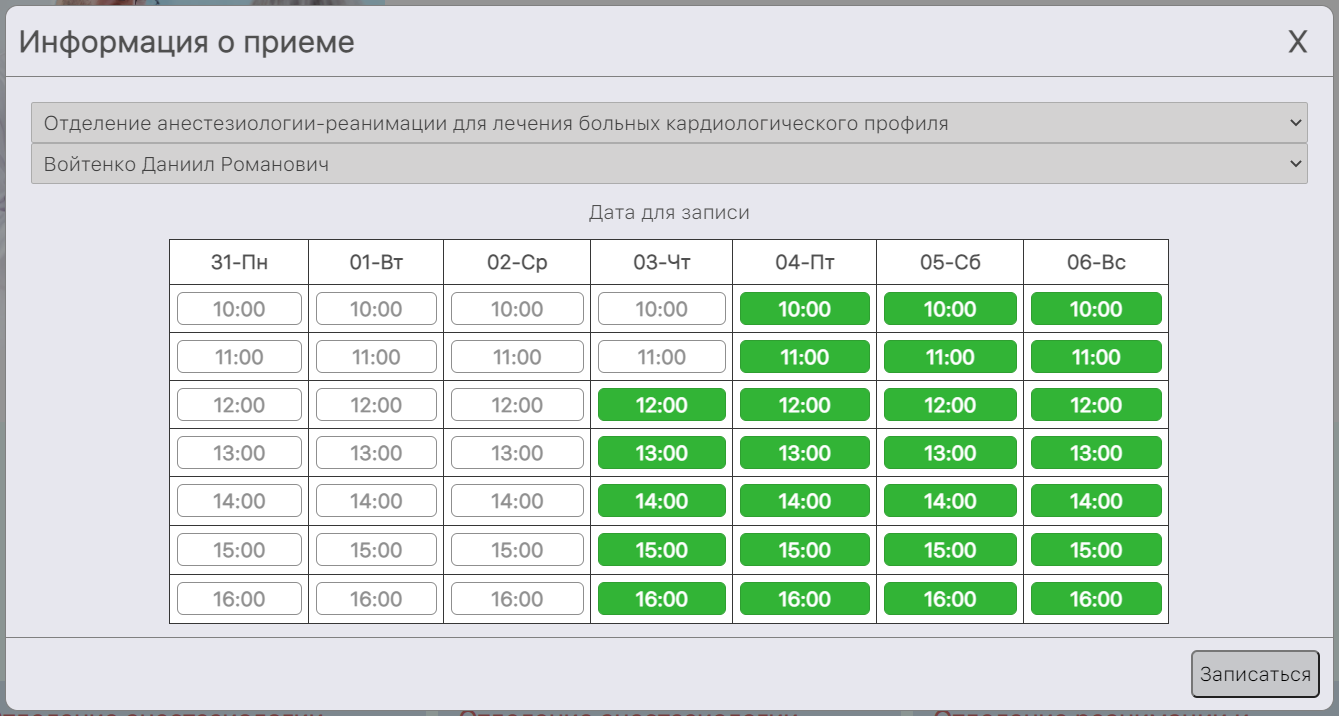


Рис. . Форма записи на прием

За инициализацию вышеописанной формы отвечает функция «drawSecondBlock» (Рис. 43). При инициализации происходит присвоение обработчиков событий выпадающим спискам.

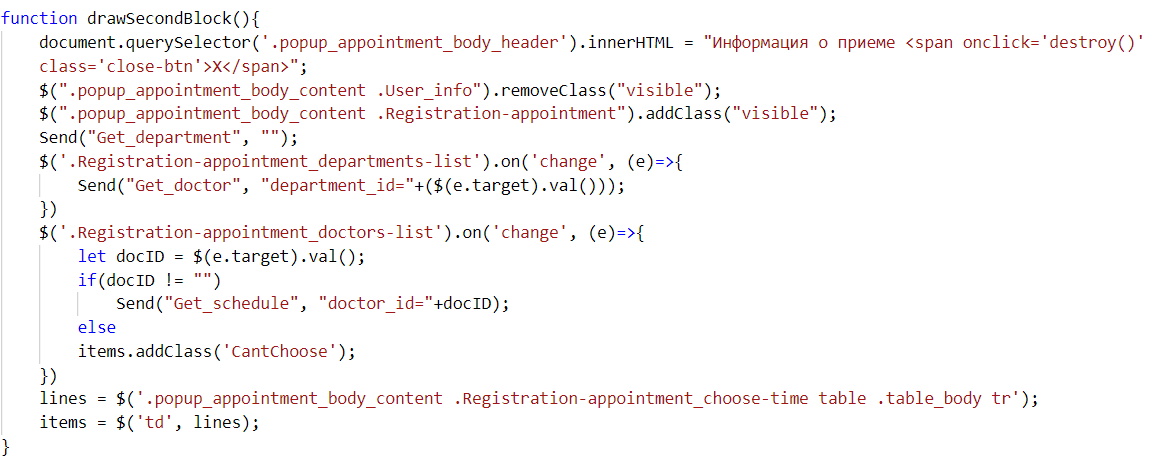


Рис. . Функция инициализации компонентов формы

При выборе отделения из соответствующего списка, вызывается функция с указанием двух параметров (Рис. 44). Первый параметр необходим для определения типа возвращаемых данных, второй же содержит строку запроса к серверу.

Запросы на сервер отправляются в асинхронном типе, таким образом страница при запросах на сервер не зависает. В ходе выполнения запроса форма притемняется и все кнопки становятся недоступными для взаимодействия (Рис. 45). При успешном выполнении запроса, в соответствии с возвращаемыми данными происходит выбор соответствующей функции для обработки данных.



Рис. . Функция отправления запросов на сервер



Рис. . Пример заблокированной для взаимодействия формы

Отрисовывание таблицы происходит при помощи отдельной функции (Рис. 46). Логика данной функции достаточно проста, но в то же время реализация не такая. Изначально вся таблица доступна для выбора даты приема, но в процессе работы возвращаются данные с указанием недоступных дат, что в результате отображается на экран пользователя.

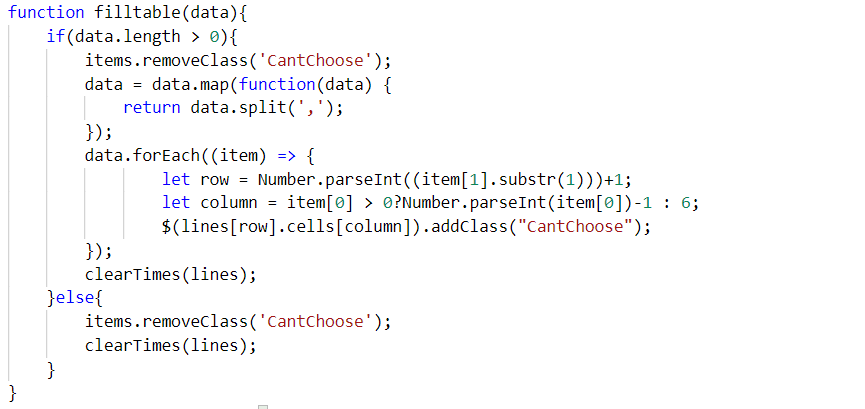


Рис. . Функция отрисовывания таблицы

Все запросы на регистрацию приходят в один контроллер, после чего происходит обработка поступивших данных. Обработка запроса на сервере заключается в проверке поступивших данных и выбор метода работы с этими данными (Рис. 47).



Рис. . Функция обработки запроса модуля регистрации на прием

Со стороны сервера процесс выбора данных выглядит следующим образом (Рис. 48). В данной функции происходит выбор набора дат из базы данных. Крайняя отбираемая дата не должна превышать семи дней с текущей даты. Эти данные возвращаются клиенту, после чего клиентская часть программы обрабатывает их.

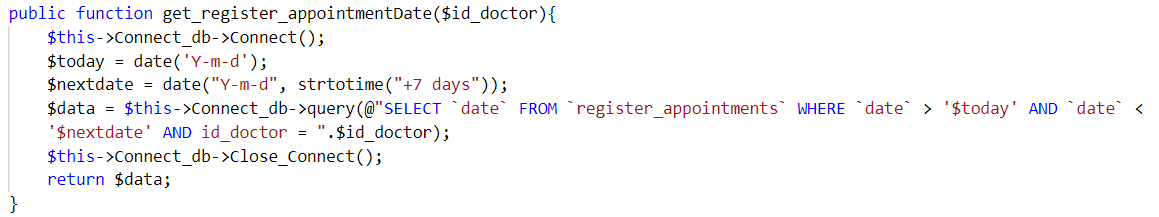


Рис. . Функция получения данных о забронированных датах для приема

Сам процесс регистрации на прием отображен на изображении (Рис. 49). Первым делом происходит проверка на доступность выбранного доктора в выбранный день, следующим этапом все передаваемые параметры проверяются на валидность и происходит сам процесс записи в базу данных. Заключительным этапом является возврат ответа сервера к пользователю.



Рис. . Функция записи на прием

Вывод по практической части

В ходе разработки информационной системы, которая описана в практической части, была выполнена основная цель данной выпускной квалификационной работы, заключающейся в разработке программного комплекса для поликлиники, позволяющего повысить эффективность работы регистратуры за счет сокращения временных и трудозатрат, а также повышения качества работы.

Заключение

В ходе выполнения выпускной квалификационной работы был проведен анализ как предметной области, так и способов реализации информационной системы. В теоретической части был проведен сбор данных о процессе регистрации в поликлинике, были рассмотрены среды разработки и языки программирования и сделаны следующие выводы:

* Проанализировав другие примеры реализации данной информационной системы, возникла более четкая картина относительно того, как сделать уникальную информационную систему;
* Рассмотрев среды разработки, был сделан выбор в пользу среды разработки VS Code.

В практической части отображена разработка информационной системы в выбранных ранее средах и на выбранных ранее языках программирования. В ходе разработки информационной системы, была выполнена основная цель данной выпускной квалификационной работы, заключающейся в разработке программного комплекса для поликлиники, позволяющего повысить эффективность работы регистратуры за счет сокращения временных и трудозатрат, а также повышения качества работы.

Список литературы

1. Белов В.В. Проектирование информационных систем: учебник. М.: Академия, 2013. – 352 с.
2. Гагарина Л.Г. Технология разработки программного обеспечения: учеб. пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул; под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2018. – 400 с. http://znanium.com/catalog.php?bookinfo=924760
3. Грегори Р. Эндрюс. Основы многопоточного, параллельного и распределенного программирования Автор: Грегори Р. Эндрюс, Москва: Вильямс 2003г. 512 стр.
4. Заботина Н.Н. Проектирование информационных систем: [Электронный ресурс] Учебное пособие / Н.Н. Заботина. – М.: НИЦ Инфра-М, 2014. – 331 с. http://znanium.com/catalog.php?bookinfo=454282.
5. <https://timeweb.com/ru/community/articles/chto-takoe-visual-studio-code>

1. <https://loftblog.ru/broadcast/1-luchshie-redaktory-koda-i-ide-sublime-text-atom-visual-studio-code-brackets-webstorm-caret-zed/>