**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К ДИПЛОМНОМУ ПРОЕКТУ**

по специальности 09.02.07 «Информационные системы и программирование»

на тему:

«Разработка клиентской части автоматизированной информационной системы для клиники «Стоматология Доктора Вознесенского»»

**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«Национальный исследовательский университет ИТМО»**

Факультет среднего профессионального образования

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кунаев Н.Д.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Говорова М.М.

Защиту дипломного проекта разрешаю:

Директор факультета СПО Университета ИТМО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зленко А.Н.

«07» июня 2021г.

Содержание

Введение 4

1 Анализ поставленной задачи 5

1.1 Формулировка поставленной задачи 5

1.2 Описание предметной области 5

1.3 Обзор и сравнение аналогичных решений 8

1.4 Функциональные требования 18

2 Проектирование и программная реализация системы 19

2.1 Описание входных и выходных данных 19

2.2 Моделирование системы 20

2.3 Выбор архитектуры системы 23

2.4 Обоснование выбора технологий и программных средств 24

2.5 Выбор методов тестирования 28

2.6 Программная реализация 29

3 Экономическая оценка разработки 32

3.1 Расчет стоимости оборудования 32

3.2 Расчет стоимости оборотных средств 33

3.3 Расчет заработной платы 34

3.4 Расчет взносов во внебюджетные фонды 35

3.5 Расчет непредвиденных расходов 36

3.6 Общая себестоимость разработки 36

Заключение 38

Список использованных источников 39

Приложение А 43

Приложение Б 56

Приложение В 72

Введение

Целью дипломного проекта является разработка клиентской части автоматизированной информационной системы для клиники «Стоматология Доктора Вознесенского» Фрунзенского района Санкт-Петербурга.

Для разработки системы требуется выполнить следующие задачи:

* анализ предметной области;
* формирование требований к системе;
* определение входных и выходных данных;
* выбор и обоснование средств реализации и архитектуры системы;
* выбор методов тестирования;
* формирование технического задания;
* разработка клиентской части;
* тестирование.

Система должна предоставлять возможность хранить информацию о пациентах, сотрудниках, услугах, ограничениях на запись пациентов к специалистам стоматологии и осуществлять запись клиентов на занятия.

Пояснительная записка к дипломному проекту содержит следующие разделы:

Введение.

Раздел 1 Анализ поставленной задачи.

Раздел 2 Проектирование и программная реализация системы.

Раздел 3 Экономическая оценка разработки.

Заключение.

Список источников.

Приложение А Техническое задание.

Приложение Б Модели разработки.

Приложение В Техническое описание.

1. Анализ поставленной задачи
   1. Формулировка поставленной задачи

Целью дипломного проекта является разработка клиентской части автоматизированной информационной системы для клиники «Стоматология Доктора Вознесенского» Фрунзенского района Санкт-Петербурга, которая позволит автоматизировать учёт клиентов.

Пользователями АИС должны быть врачи, их ассистенты, администраторы, бухгалтеры, директор, и прочий персонал клиники.

* 1. Описание предметной области

Стоматологическая клиника «Доктор Вознесенский» предоставляет весь спектр услуг, присущий стоматологической клинике.

Работа клиники с клиентами в текущий момент происходит так: клиент записывается на прием по одному из каналов связи (телефон, СМС, почта, мессенджеры), в ходе личного общения с администратором, либо записывается сразу после посещения на следующий раз.

В назначенное время клиент должен прийти в клинику. Если клиент опаздывает, он может сообщить об этом администратору, чтобы перенести визит на свободное время. Если же клиент опаздывает без предупреждения, его место может занять другой клиент (например, пришедший лично в это время). Клиент также может заранее сообщить о невозможности прихода, после чего ему будет предложено выбрать другой день и время.

Приход пациента в назначенное время отмечается администратором в программе для учета пациентов. Если пациент пришел в первый раз, администратор распечатывает договоры и документы, которые клиент должен подписать. Если клиент отказывается, прием отменяется.

Затем пациенту сообщается номер кабинета. Врач проводит прием, после чего заносит в систему оказанные услуги, потраченные материалы и проданные товары. Пациент возвращается на ресепшн, администратор сообщает ему состояние его счета на данный момент и его постоянную скидку. После этого клиент может оплатить некоторую часть задолженности перед клиникой (или авансом «положить» деньги на счет), используя или не используя при этом дисконтную карту. Если у клиента нет карты – он может ее оформить.

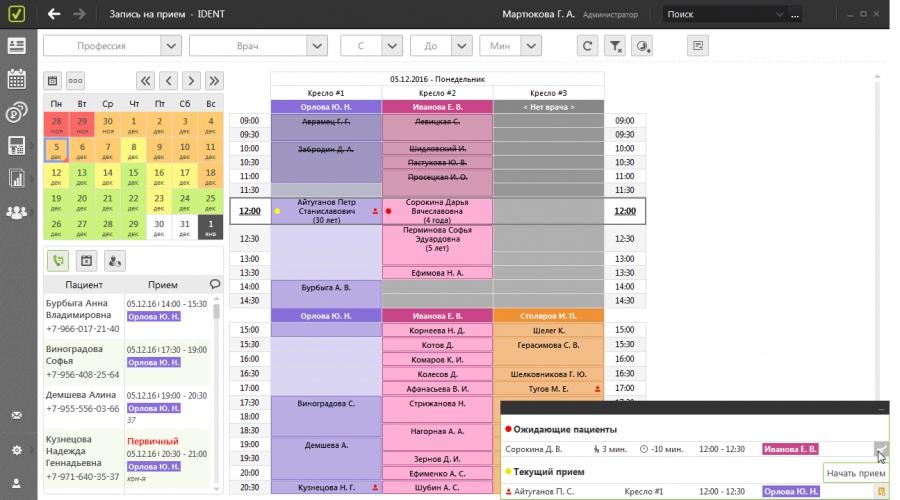
Так как большая часть действий в рабочем процессе связана с манипуляциями данными, заказчик использует для их выполнения АИС. Но используемая в данный момент существующая информационная система обладает рядом недостатков, главными из которых являются высокая стоимость системы и большая загруженность пользовательского интерфейса.

* 1. Обзор и сравнение аналогичных решений

Для выявления актуальности разработки и необходимых функций проведено сравнение аналогичных решений.

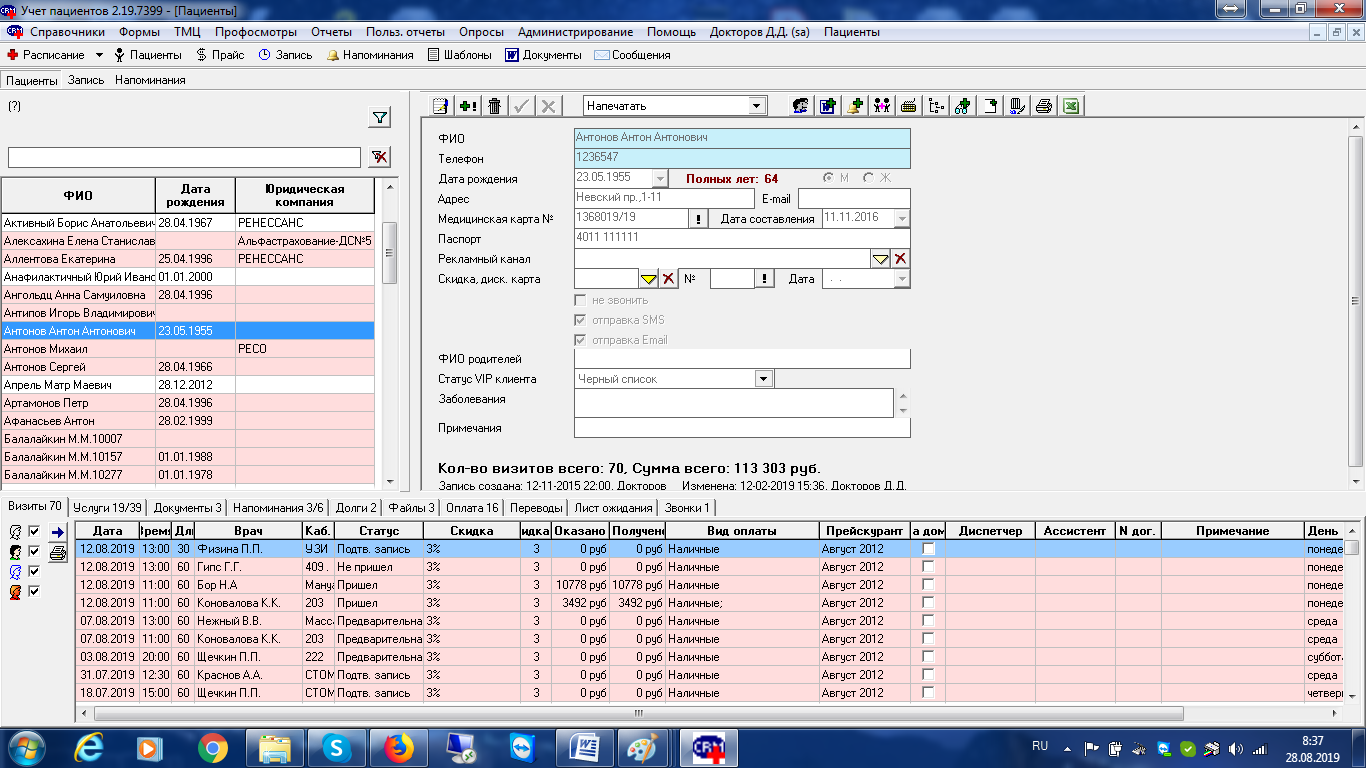
Для сравнения были выбраны использовавшиеся заказчиком системы «iDent» (url: <https://dent-it.ru/>, дата обращения: 04.02.2021) и МИС «Учет пациентов» от «Dr. Soft» (url: <https://crm-soft.ru/>, дата обращения 04.02.2021), а также системы «iStom» (url: <https://i-stom.ru/>, дата обращения 04.02.2021) и «MEDODS» (url: <https://medods.ru/>, дата обращения 04.02.2021) из рейтинга лучших CRM для стоматологий «CRMindex» (url: <https://crmindex.ru/products>, дата обращения 04.02.2021).

* Компьютерная программа «IDENT» для управления стоматологической клиникой – решение для ведения учета в стоматологии. Своими плюсами считают цену, не зависящую от размера клиники, понятный интерфейс, обучение сотрудников пользованию системой и хорошую техподдержку. Пример интерфейса можно увидеть на рисунке 1;



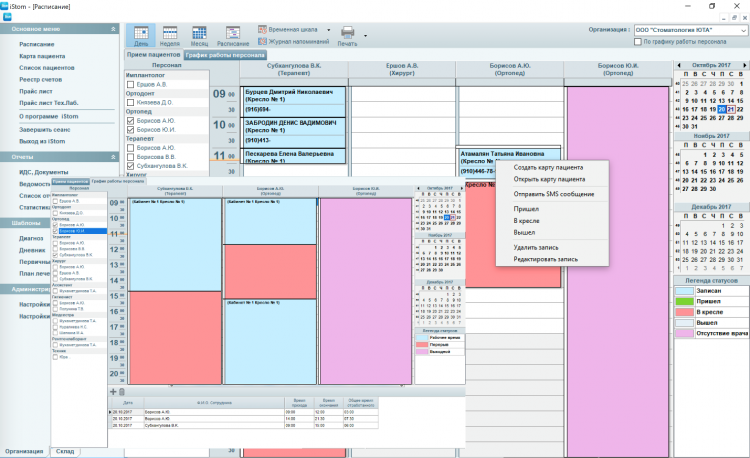
*Рисунок 1 – «Интерфейс программы «Ident»*

* МИС «Учет пациентов» - CRM для медицинской клиники любого профиля. Имеет только on-prem версию (т.е. устанавливается на компьютер клиента, а не в облако). Продукт на рынке уже больше 13 лет. Также из плюсов выделяют простоту освоения и бесплатное обучение пользованию программой и «демократичную цену». Пример интерфейса можно увидеть на рисунке 2;



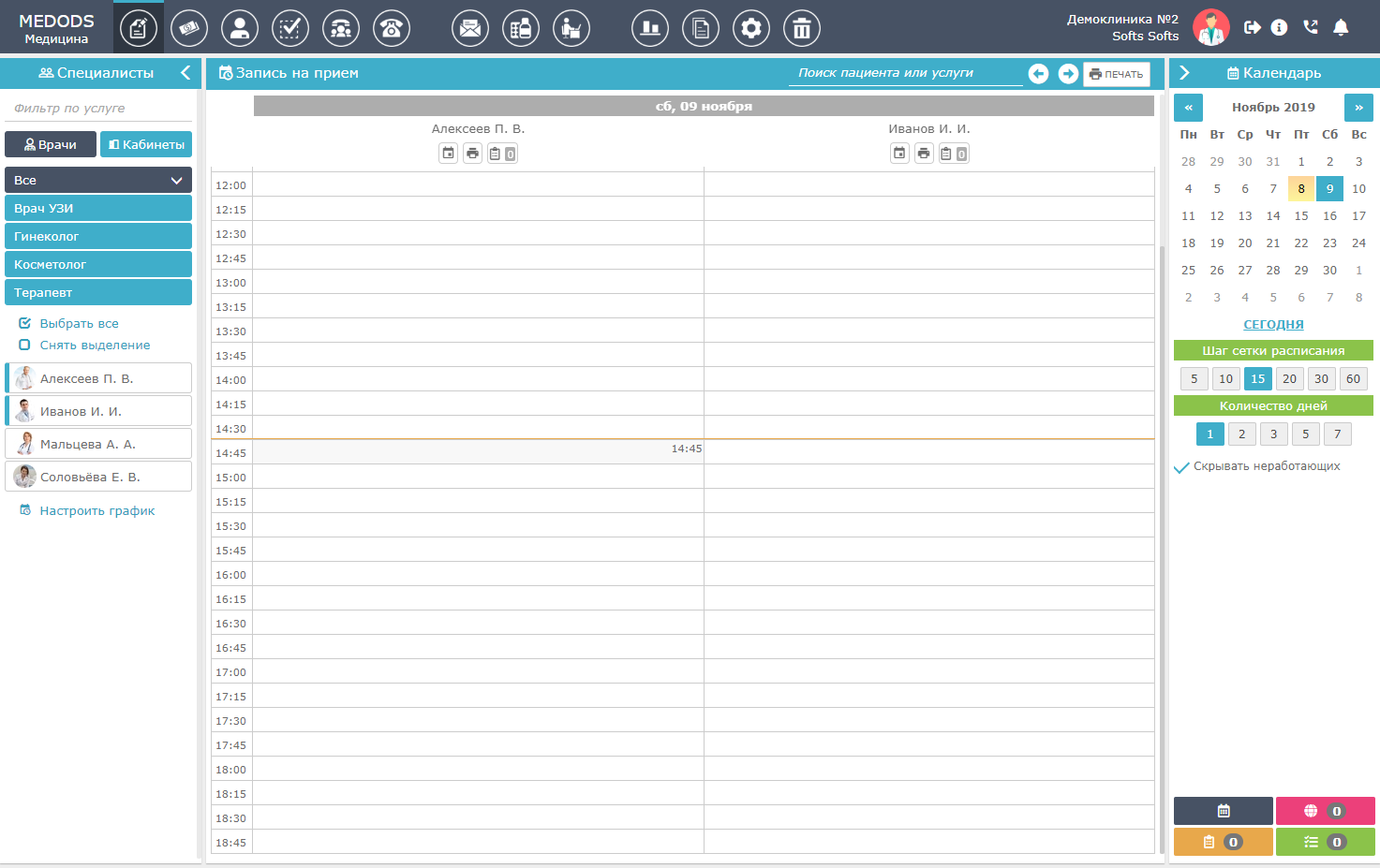
*Рисунок 2 – «Интерфейс программы «Учёт пациентов»*

* Программа для автоматизации стоматологии «iStom» - стоматологическая CRM, занимающая первое место в рейтинге www.crmindex.ru. Обладает как облачной, так и on-prem версиями. Как преимущества, выделяют соответствие системы законодательству и стандартам РФ в области медицины, легкость в освоении, а также масштабируемость (возможность применения как в небольшой клинике, так и в сети стоматологий). Пример интерфейса можно увидеть на рисунке 3;



*Рисунок 3 – «Интерфейс программы «iStom»*

* Медицинская информационная система «MEDODS» - решение для медицинских центров и стоматологий. Имеют on-prem и облачную версию. В систему входит интерфейс клиента для самостоятельной записи. Поддерживают работу с кассовыми аппаратами «Атол» и «Штрих-М». Также выделяют простоту систему и шифрование данных при передаче и хранении, как плюсы. Пример интерфейса можно увидеть на рисунке 4;



*Рисунок 4 – «Интерфейс программы «MEDODS»*

В таблице 1 приведено сравнение аналогичных решений.

Таблица 1 − Обзор аналогичных решений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование аналогов  Функция | iDent | Учет пациентов | iStom | MEDODS |
| “User friendly” интерфейс | Да | Нет | Нет | Да |
| База знаний (инструкция) | Да | Да/нет | Нет | Нет |
| Лаконичные дизайнерские решения (UX) | Да | Нет | Нет | Да |
| Мобильное приложение | Нет | Нет | Да | Да |
| Достаточная информация на сайте | Да | Нет | Да | Нет |
| Функции облачных сервисов | Нет | Нет | Да | Нет |
| Удалённая помощь | Да | Да | Нет | Нет |

В результате анализа приложений выделены следующие необходимые функциональные возможности:

* требуется создать понятный интерфейс;
* требуется реализовать авторизацию;
* ведение карт пациентов;
* запись пациентов на приём;
* ведение отчётности вносимых средств пациентами;
* сбор статистики.

При этом всем веб-сервис должен предоставлять исчерпывающую информацию об услугах.

* 1. Функциональные требования

На основе анализа предметной области и аналогичных решений были выявлены следующие функциональные возможности, которые будут представлены:

* авторизация;
* ведение карты пациента;
* запись пациентов на приём;
* ведение отчётности вносимых средств пациентами;
* печать документов;
* сбор статистики.

1. Проектирование и программная реализация системы
   1. Описание входных и выходных данных

Функциональные возможности клиентской части системы:

* аутентификация;
  + входные данные: логин и пароль;
  + выходные данные: сессия в системе;
* ведение карты пациента;
  + входные данные: информация о пациенте;
  + выходные данные: сохранённая в системе карта пациента;
* запись пациентов на приём;
  + входные данные: информация о пациенте и дата приёма;
  + выходные данные: новая запись на существующий приём;
* ведение отчётности вносимых средств пациентов;
  + входные данные: транзакции денежных средств;
  + выходные данные: баланс пациента;
* сбор статистики;
  + входные данные: информация о пациентах;
  + выходные данные: графики и сводки о работе стоматологической клиники;

Все входные данные записываются в базу данных. Выходные данные формируются из запросов к БД.

* 1. Моделирование системы

Для моделирования системы были использованы следующие методологии и нотации:

* BPMN [1] – система условных обозначений (Business Process Model and Notation, нотация и модель бизнес-процессов) и их описания в XML для моделирования бизнес-процессов. Разработана Business Process Management Initiative и поддерживается Object Management Group, после слияния обеих организаций в 2005 году. Последняя версия BPMN – 2.0 (2.0.2);
* User Story (пользовательские истории) [2] – способ описания требований к разрабатываемой системе, сформулированный как одно или более предложений на повседневном или деловом языке пользователей.

Функциональная модель системы в нотации BPMN представлена на рисунке Б.1.

Концептуальная модель системы в нотации User Story представлена на рисунке Б.2.

Концептуальная модель системы (диаграмма классов) представлена на рисунке Б.3.

Модель прикладного интерфейса на рисунке Б.4.

* 1. Выбор архитектуры системы

Веб-приложение – приложение, в котором клиентом выступает браузер, а сервером – веб-сервер [3]. Архитектура «клиент-сервер» бывает двухзвенной и трехзвенной. При двухзвенной архитектуре сервер обрабатывает клиентские запросы и обращается к базе данных (рисунок 5).

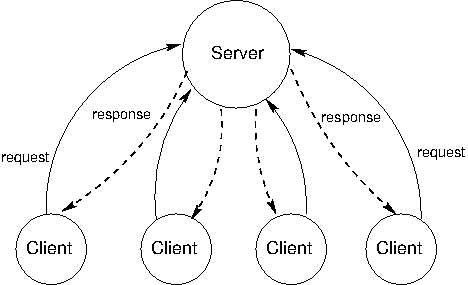


Рисунок 5 – Двухзвенная архитектура клиент-серверного приложения

При трехзвенной архитектуре сервер разделен на две части: первая обрабатывает клиентские запросы, а вторая часть обращается к базе данных (рисунок 6).

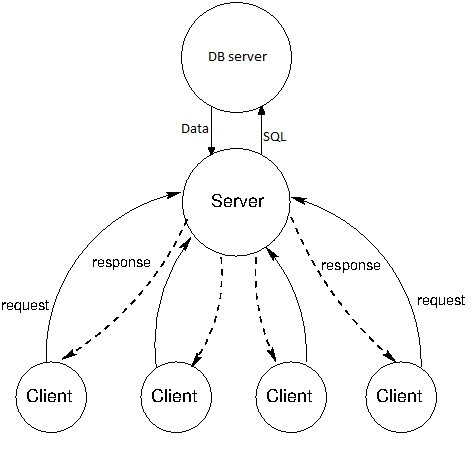


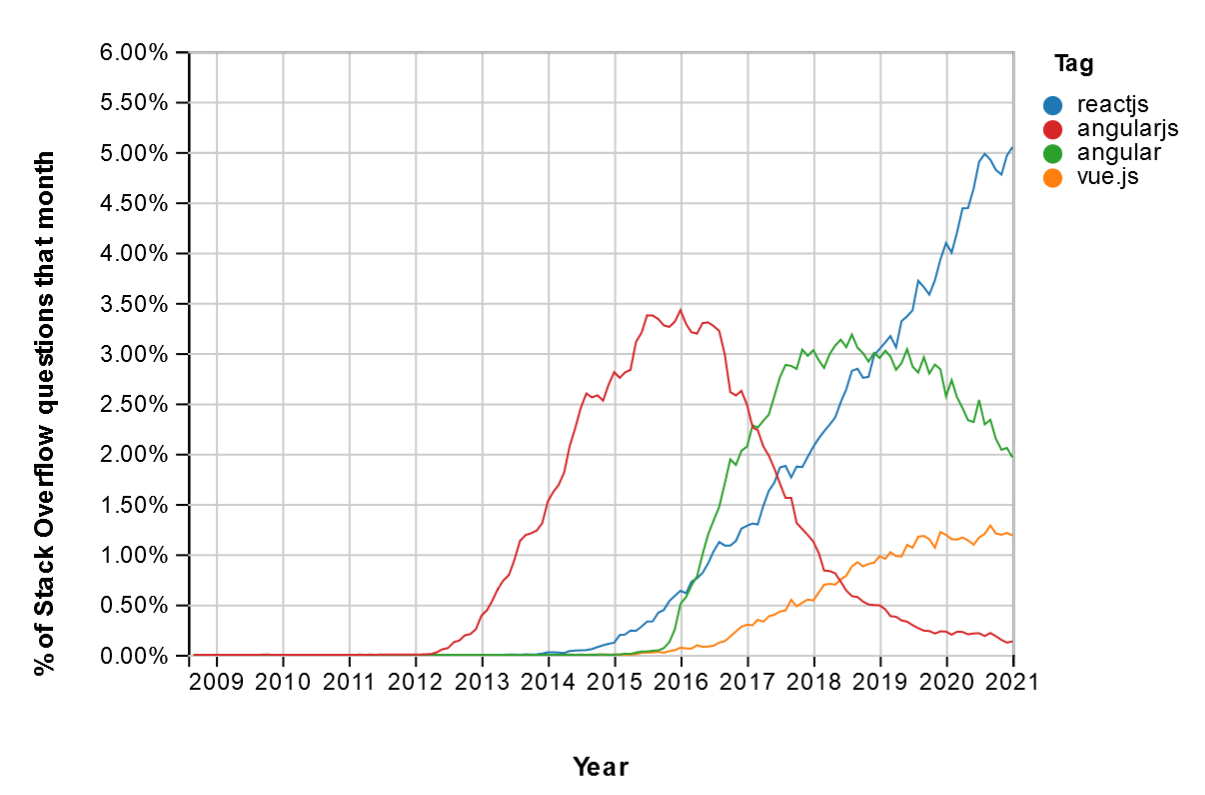
Рисунок 6 – Трехзвенная архитектура клиент-серверного приложения

* 1. Обоснование выбора технологий и программных средств

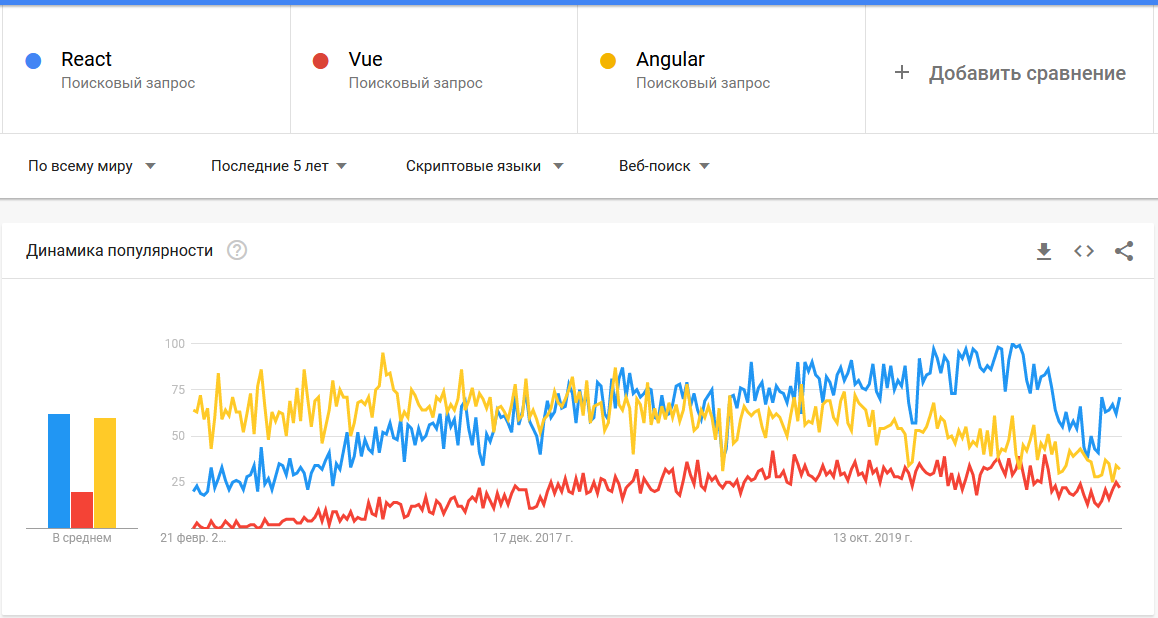
Языка программирования JavaScript широко используется для реализации взаимодействия пользователя с веб-сайтами и приложениями [4]. Для упрощения разработки используются различные библиотеки и фреймворки:

* фреймворк VueJs;
* фреймворк AngularJs;
* фреймворк ReactJs.

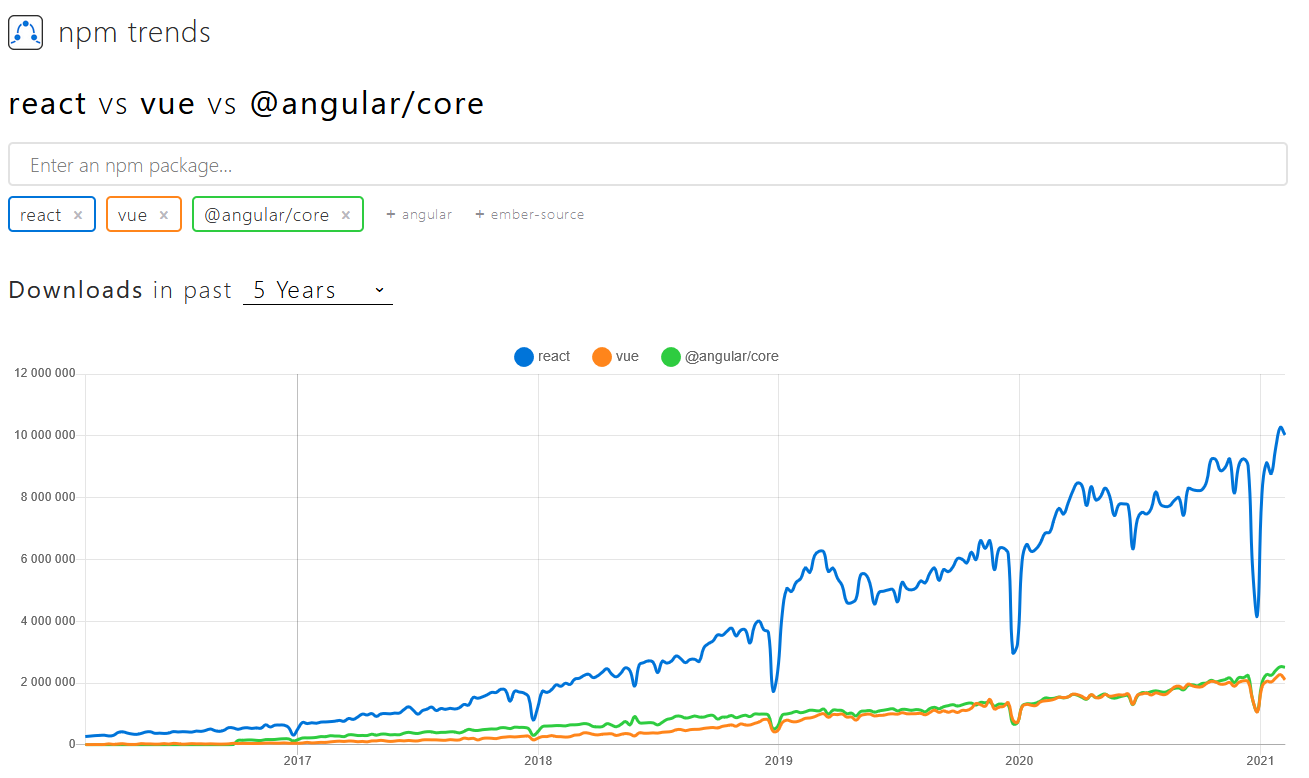
Ниже представлены сводки частоты поиска разных средств разработки на различных площадках: на площадке Stack Overflow (url: <https://insights.stackoverflow.com/trends?tags=reactjs%2Cvue.js%2Cangular%2Cangularjs>, дата обращения 15.02.2020) на рисунке 7, на площадке Google.Trends (url: <https://trends.google.com/trends/explore?cat=733&date=today%205-y&q=React,Vue,Angular>, дата обращения 15.02.2020) на рисунке 8, на площадке npm trends (url: <https://www.npmtrends.com/react-vs-vue-vs-@angular/core>, дата обращения 15.02.2020) на рисунке 9.



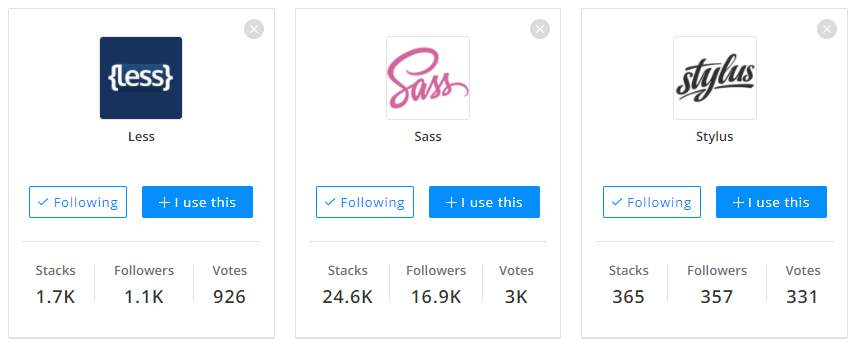
*Рисунок 7 – Сравнение частоты запросов на Stack Overflow.*

**

*Рисунок 8 – Сравнение частоты запросов в поисковой системе Google.*

*Рисунок 9 – Сравнение частоты скачиваний различных дистрибутивов с помощью пакетного менеджера NPM.*

Кроме вспомогательных средств (фреймворков и библиотек) для языка программирования JavaScript требуется использовать язык разметки HTML5 (альтернатив у него нет), а также язык стилей CSS. Для языка стилей можно использовать различные препроцессоры, упрощающие взаимодействие со стилями: Sass, Scss, Stylus, Less. Сравнение частоты запросов по ним представлено на рисунке 10 (url: <https://stackshare.io/stackups/less-vs-sass-vs-stylus>, дата обращения 15.02.2020).



*Рисунок 10 – Сравнение частоты запросов.*

Исходя из данных на рисунке 10 можно сделать вывод, что популярность препроцессора Sass намного выше, чем популярность Less и Stylus. Scss является частью Sass поэтому он отдельно не рассматривается.

Таким образом, приложение должно быть реализовано при помощи следующих средств:

* язык программирования JavaScript версии не ниже ES6;
* фреймворк ReactJs версии не ниже 17;
* язык разметки HTML5;
* метаязык стилей Sass;
* интегрированная среда разработки (IDE) WebStorm версии 2020.3.2 от компании JetBrains;
* сервер для разработки Node.Js версии не ниже 14.15.4;
* пакетный менеджер yarn версии не ниже 1.22.10.

При эксплуатации приложения не требуется никакого дополнительного программного обеспечения.

* 1. Выбор методов тестирования

Любая программа подлежит тестированию на этапе разработки. Разрабатываемое веб-приложение будет тестироваться с помощью технологий белого и чёрного ящиков.

С помощью метода белого ящика будут проведены:

* модульное тестирование;
* интеграционное тестирование.

Тестирование веб-приложений проходит через несколько этапов [5]:

* на этапе функционального тестирования проверяются:
  + функциональные требования (с помощью метода чёрного ящика);
  + загрузка требуемых файлов (с помощью метода чёрного ящика);
* на этапе тестирования вёрстки проверяются:
  + соответствие вёрстки макету (чёрный ящик);
  + оптимизация графических изображений (чёрный ящик);
  + валидность кода [6] (белый ящик);
  + совместимость (чёрный ящик);
* на этапе тестирования удобства интерфейса:
  + собираются данные и отзывы пользователей (чёрный ящик);
  + анализ собранных данных и применение результатов анализа в последующей разработке;
* тестирование быстродействия сайта [7];
  1. Программная реализация

Система разработана в соответствии с техническим заданием, представленным в приложении A.

Техническое описание представлено в приложении В.

Реализованы следующие функции:

Для администратора:

* авторизация;
* запись на приём:
  + выбор даты;
  + выбор времени;
  + выбор пациента;
  + выбор врача;
  + описание приёма: причина, комментарий;
* просмотр расписания:
  + просмотр расписания на определенную дату;
* регистрация новых пациентах.

Для директора:

* авторизация;
* регистрация новых врачей;
* регистрация новых услуг.

Функциональная модель поведения приложения соответствует модели на рисунке Б.1.

Поведение пользователей системы соответствует диаграмме пользовательских историй, представленной на рисунке Б.2.

Система реализована в виде веб-приложения. Для информационного обеспечения системы использована технология реляционных баз данных.

Для взаимодействия сайта с сервером был разработан интерфейс взаимодействия – API [8]. Принцип действия REST API представлен на рисунке 11. В качестве формата входных и выходных данных использован JSON.

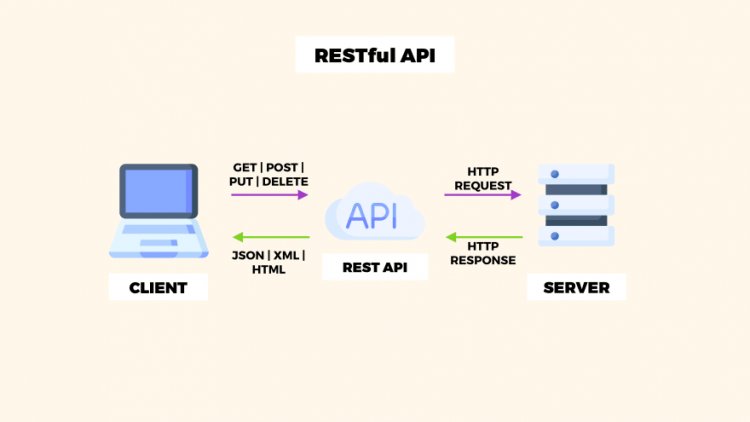


Рисунок 11 – Модель работы REST API

1. Экономическая оценка разработки

В настоящем разделе рассматриваются экономические аспекты разработки программного продукта, являющиеся элементами планирования ресурсных затрат (себестоимости разработки).

Себестоимость – это стои­мо­ст­ная оцен­ка те­ку­щих за­трат пред­при­ятия на про­из­вод­ст­во и реа­ли­за­цию про­дук­ции (то­ва­ров, ра­бот, ус­луг) [9].

Все затраты, учитываемые в себестоимости, классифицируются различными способами. Наиболее адекватной целям дипломного проекта является классификация затрат по элементам.

Затраты, образующие себестоимость системы, группируются в соответствии с их экономическим содержанием по следующим элементам:

* материальные затраты (за вычетом стоимости возвратных отходов);
* затраты на оплату труда (заработная плата работников предприятия);
* отчисления на социальные нужды (взносы во внебюджетные фонды/отчисления на социальные нужды);
* амортизация оборудования;
* непредвиденные расходы. [10]
  1. Расчет стоимости оборудования

При создании настоящего программного продукта не требуется использования средств и предметов труда стоимостью более *100* тысяч рублей за каждую единицу. Однако, перечисленные ниже объекты техники и мебели были и будут использованы в дальнейшем (не только в рамках описываемого проекта). Поэтому расчёт их стоимости, которую логично было бы включить в себестоимость проекта, следует проводить по аналогии с линейным методом начисления амортизации основных фондов, используя формулу (1):

(1)

где – сумма амортизационных отчислений,

– первоначальная стоимость объекта,

– норма амортизации.

Норма амортизации рассчитывается по формуле (2).

(2)

где – срок полезного использования объекта, выраженный в месяцах.

Используя данные формулы, можно рассчитать стоимость задействованного в разработке ПП оборудования.

Результаты всех расчетов представлены в таблице 6.

Таблица 2 – Стоимость оборудования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование оборудования | Первоначальная стоимость, руб. | Срок полезного использования, мес. | Стоимость оборудования в месяц, руб. | Срок использования для проекта, мес. | Стоимость оборудования на проект, руб. |
| Ноутбук | 49990 | 36 | 1400 | 6 | 8400 |
| Стул | 10000 | 84 | 120 | 6 | 720 |
| Стол письменный | 10000 | 84 | 120 | 6 | 720 |
| Итого: | | | | | 9840 |

Исходя из расчетов, представленных выше, стоимость оборудования на время выполнения проекта составляет 9840 руб.

* 1. Расчет стоимости оборотных средств

Оборотные средства (оборотный капитал) предприятий – это совокупность средств предприятий, выраженных в денежной форме, одноразово участвовавших в производственном процессе, видоизменивших свою натурально-вещественную форму и полностью переносящую свою стоимость на производственный продукт.

Стоимость оборотного капитала рассчитывается по формуле (3).

(3)

где – сумма оборотного капитала,

– цена единицы ресурса,

К – количество потраченных ресурсов.

Стоимость электроэнергии рассчитывается по формуле (4).

(4)

где – стоимость электроэнергии,

– мощность блока питания ноутбука в кВт,

– стоимость кВт\*ч,

– время в часах.

Результаты всех расчетов стоимости оборотных средств представлены в таблице 3.

Таблица 3 – Расчет стоимости оборотных средств

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена единицы, руб. | Количество ресурса «на проект» | Стоимость |
| Электроэнергия | 4,97 | 27 | 135 |
| Интернет | 450 | 6 | 2700 |
| USB - накопитель | 700 | 1 | 700 |
| Папка-скоросшиватель | 65 | 1 | 65 |
| Итого: | | | 3600 |

Исходя из расчетов, произведенных выше, стоимость оборотных средств на проект составляет 3600.

* 1. Расчет заработной платы

Основная заработная плата разработчиков рассчитывается по формуле (7):

(7)

где – основная заработная плата,

– среднечасовая заработная плата работника в рублях,

P – продолжительность работы, выполняемой работником, в часах.

Результаты всех расчетов основных ЗП участников проекта представлены в таблице 4.

Таблица 4 – Расчет ФОТ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Нагрузка, часов | Стоимость часа работы, руб. | Фонд оплаты труда, руб. | В т. ч. НДФЛ (13% от оклада) |
| Разработчик | 300 | 600 | 180000 | 23400 |
| Руководитель дипломного проекта | 30 | 120 | 3600 | 468 |
| Консультант по обоснованию | 4 | 200 | 800 | 104 |
| Консультант по экономике | 3 | 250 | 750 | 98 |
| Итого: | | | 185150 | 24070 |

Исходя из расчетов, представленных в таблице 4, фонд оплаты труда составляет 185150 рублей, а НДФЛ с этой суммы – 24070 рублей.

* 1. Расчет взносов во внебюджетные фонды

Государственный внебюджетный фонд – фонд денежных средств, образуемый вне федерального бюджета и бюджетов субъектов Российской Федерации и предназначенный для реализации конституционных прав граждан на пенсионное обеспечение, социальное страхование, охрану здоровья и медицинскую помощь.

Основными по размерам и значению являются социальные внебюджетные фонды:

* Пенсионный фонд Российской Федерации (ПФ);
* Фонд социального страхования Российской Федерации (ФСС);
* Федеральный фонд обязательного медицинского страхования (ФФОМС).

Общий процент отчислений во внебюджетный фонды составляет 30%, из них: 22% в Пенсионный фонд, 2,9% в Фонд социального страхования, 5,1% в Фонд медицинского страхования.

Таким образом, взносы во внебюджетные фонды рассчитываются по формуле (8):

(8)

где В – взнос во внебюджетный фонд,

ФОТ – фонд оплаты труда,

– процентная ставка отчислений.

Результаты всех расчетов отчислений во внебюджетные фонды представлены в таблице 5.

Таблица 5 – Расчёт взносов во внебюджетные фонды

|  |  |  |
| --- | --- | --- |
| ФОНД | % ОТ ФОТ | СУММА В ПРОЕКТЕ, РУБ. |
| Пенсионный фонд | 22 | 40645 |
| Фонд социального страхования | 2,9 | 5357,75 |
| Фонд медицинского страхования | 5,1 | 9422,25 |
| ИТОГО | 30 | 55425 |

Исходя из расчетов, представленных выше, отчисления во внебюджетные фонды составляют 55425 рублей.

* 1. Расчет непредвиденных расходов

Непредвиденные расходы — расходы, не предусмотренные планами, программами, бюджетами, сметами, возникшие неожиданным образом; включают непредвиденное увеличение издержек производства и обращения, возмещение убытков, штрафы, пени, неустойки [11].

Непредвиденные расходы рассчитываются как 10% от фонда оплаты труда и составляют (формула 9).

(9)

* 1. Общая себестоимость разработки

Сложив расходы по всем элементам (таблица 6), получим сумму себестоимости разработки ПП:

Таблица 6 – Себестоимость разработки ПП

|  |  |  |
| --- | --- | --- |
| Элемент затрат | Сумма, руб. | Доля в общей сумме, % |
| Оборудование | 9840 | 3.62 |
| Оборотные средства | 3600 | 1.3 |
| Фонд оплаты труда | 185150 | 67.92 |
| Взносы во внебюджетные фонды | 55425 | 20.37 |
| Непредвиденные расходы | 18475 | 6,79 |
| Итого: | 272490 | 100 |

Долевое распределение затрат на проект представлено на круговой диаграмме на рисунке 12.

Рисунок 12 – Долевое распределение затрат на разработку ПП

В ходе оценки ресурсов, необходимых для разработки программного продукта, были выполнены расчеты и выявлено, что общая себестоимость создания программного продукта составляет 272490 рубля. Наибольший удельный вес имеет фонд оплаты труда, что связано с длительным вовлечением разработчика в проект.

Заключение

По завершении работы над дипломным проектом разработана клиентская часть веб-приложение для медицинской клиник «Стоматология Доктора Вознесенского» Фрунзенского района Санкт-Петербурга.

В ходе работы была проанализирована предметная область, поведение системы и данные, необходимые для работы системы. По полученным данным были сформированы требования к системе, на основании которых система была детально спроектирована, и составлено техническое задание. Затем были выбраны технологии для реализации и разработана клиентская части приложения для частной стоматологии Фрунзенского района. Система была протестирована для выявления ошибок и их исправления.

Все поставленные задачи выполнены в полном объеме.

Список использованных источников

1. Спецификация BPMN 2.0 [Электронный ресурс] // Object Management Group [сайт], URL: <http://www.omg.org/spec/BPMN/2.0.2/> (дата обращения: 20.04.2021);
2. A comparative study of software tools for user story management [Электронный ресурс] // Science Direct [сайт], URL: <https://www.sciencedirect.com/science/article/abs/pii/S0950584914001293?via%3Dihub> (дата обращения: 20.04.2021);
3. Архитектура клиент-сервер. Курс лекций «Тестирование программного обеспечения» [Электронный ресурс] // GitBook – инструмент для разработчиков для создания документации [сайт], URL: <https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-6-ch1-arhitektura-klient-server.html> (дата обращения: 12.02.2021);
4. Веб-технологии для разработчиков | MDN [Электронный ресурс] // MDN Web Docs [сайт], URL: <https://developer.mozilla.org/ru/docs/Web> (дата обращения: 20.03.2021);
5. Методика тестирования web приложения [Электронный ресурс] // Тестирование Программного Обеспечения [сайт], URL: <https://software-testing.org/testing/metodika-testirovaniya-web-prilozheniya.html> (дата обращения: 02.03.2021);
6. Markup Validation Service [Электронный ресурс, сайт], URL: <https://validator.w3.org> (дата обращения 23.03.2021);
7. PageSpeed Insights [Электронный ресурс, сайт], URL: <https://developers.google.com/speed/pagespeed/insights/?hl=ru> (дата обращения: 23.03.2021);
8. Что такое REST API? [Электронный ресурс] // Курс по документированию API [сайт], URL: <https://starkovden.github.io/what-is-rest-api.html> (дата обращения 18.05.2021);
9. Себестоимость [Электронный ресурс] // Большая российская энциклопедия [сайт], URL: <https://bigenc.ru/economics/text/3542615> (дата обращения: 09.05.2021);
10. Постановление Правительства РФ от 5 августа 1992 г. N 552 [Электронный ресурс] // ГАРАНТ - Законодательство (кодексы, законы, указы, постановления) РФ, аналитика, комментарии, практика [сайт], URL: <http://base.garant.ru/10104801/> (дата обращения: 09.05.2021);
11. Непредвиденные расходы [Электронный ресурс] // Энциклопедия по экономике [сайт], URL: <https://economy-ru.info/info/71274/> (дата обращения: 09.05.2021).

Приложение А

Техническое задание

ТЕРМИНЫ, ПОНЯТИЯ И СОКРАЩЕНИЯ

В данном разделе представлены основные термины, понятия и сокращения, используемые в настоящем документе – смотреть [Таблица 1].

Таблица 1 – Термины, понятия и сокращения

|  |  |
| --- | --- |
| **Термин, понятие или сокращение** | **Определение** |
| БД | База данных |
| ОС | Операционная система |
| ПО | Программное обеспечение |
| Система | Разрабатываемый программный продукт |
| СУБД | Система управления базой данных |
| ТЗ | Техническое задание |
| GUI | Global User Interface – Глобальный пользовательский интерфейс |

1. Назначение разработки

Назначением разработки является создание веб-приложения для автоматизации учёта пациентов медицинской клиники «Стоматология Доктора Вознесенского».

Программа должна эксплуатироваться на объекте заказчика. Пользователями программы должны являться администраторы, директор и врачи. Пациенты не должны иметь доступ к системе.

1. Цель и задачи, решаемые в процессе достижения цели

Целью является создание клиент-серверной автоматизированной информационной системы (АИС) для организации работы СК.

Задачи, решаемые в процессе достижения цели:

1. Уточнение требований к системе (при необходимости).
2. Детальное проектирование системы.
3. Программная реализация.
4. Тестирование.
5. Требования к приложению
   1. Требования к функциональным характеристикам. Описание функциональности разрабатываемой системы

Система должна обеспечить хранение информации о клиентах, предоставленных услугах и графике работы сотрудников.

В системе должны быть определены следующие категории пользователей:

* администратор;
* директор;
* врач.

В системе должны быть реализованы следующие функциональные возможности:

* + - работа с расписаниями сотрудников, изменение графиков работы;
    - добавление, удаление и изменение информации о приемах пациентов;
    - учет оказываемых клиникой услуг, их классификация по номенклатуре медицинских услуг;
    - учет клиентов клиники, изменение и добавление информации о них, добавление клиента в архив и удаление из него;
    - возможность ограничения прав.
  1. Описание входных и выходных данных

Входными данными приложения являются следующие данные.

О пациентах:

* паспортные данные;
* ФИО;
* дата рождения;
* адрес регистрации;
* адрес проживания;
* профессия;
* телефон;
* пол;
* место рождения;
* статус архивации;
* откуда узнал;
* адрес электронной почты;
* комментарий;
* баланс.

О визитах:

* пациент;
* дата;
* причина;
* посещённые врачи;
* комментарий;
* статус.

О сотрудниках:

* ФИО;
* Дата рождения;
* должность;
* контактные данные.
* логин;
* пароль;
* права доступа.

Об услугах:

* цена;
* наименование;
* группа.

О предоставленных услугах:

* услуга;
* посещение, в которое услуга оказана.

Все входные данные записываются в БД. Выходные данные формируются из запросов к БД.

* 1. Модель приложения

Функциональная модель поведения приложения должна соответствовать модели на рисунках Б.1.

Модель поведения пользователей должна соответствовать модели, представленной на рисунке Б.2.

Модель прикладного программного интерфейса должна соответствовать модели на рисунке Б.3.

* 1. Эргономические и технико-эстетические требования

При проектировании форм необходимо стремиться к использованию ограниченного набора цветов и уделять внимание их правильному сочетанию. По возможности должны использоваться системные цвета и свойственные системе элементы управления. Набор из трёх цветов продиктован основными цветами логотипа стоматологической клиники.

При неправильно веденных данных пользователю должно быть показано диалоговое окно с ошибкой (язык ошибки в данном случае может быть русским или английским).

Правила валидации полей определяются на стороне клиента.

* 1. Структура системы

Система должна включать в себя следующие компоненты клиентского интерфейса (подсистемы):

* пациенты – модуль, содержащий информацию о клиентах в системе;
* сотрудники – модуль, содержащий информацию о пользователях системы;
* визиты – модуль, содержащий информацию о посещениях пациентов клиники и посещённых врачах;
* статистика – модуль, содержащий статистическую информацию о работе клиники и о клиентах.
  1. Требования к информационному обмену между компонентами системы

Для обеспечения целостности данных должны использоваться системы валидации форм. Доступ к данным должен быть предоставлен только авторизованным пользователям с учетом их прав доступа.

* 1. Требования к технологиям разработки и использования

При разработке приложения должны использоваться парадигмы реактивного, декларативного, объектно(прототипно)-ориентированного и функционального программирования. Архитектура реализуемой системы – клиент-серверное веб-приложение.

* 1. Требования к защите от несанкционированного доступа

Контроль защиты информации от несанкционированного доступа осуществляется на стороне клиента по средствам использования токена авторизации.

Компонент прикладного программного интерфейса должен обеспечивать защиту от несанкционированного доступа:

* аутентификация пользователя;
* проверку полномочий пользователя при работе с системой.

Защищённая часть системы должна использовать пароли, символы должны маскироваться одним типом символов (\*).

Директор имеет доступ ко всем функциям системы.

Администраторы не могут вносить изменения в прайс-листы, не может добавлять новых пользователей, не может изменять существующих пользователей.

Врачи могут получить доступ только к информации о пациентах, их визитах и выполненных услугах.

* 1. Требования к программным средствам разработки

Для реализации приложения должны использоваться следующие средства разработки:

Компонента клиентского интерфейса должно быть реализовано при помощи следующих средств:

* язык программирования JavaScript (версии ECMAScript 9 или выше);
* язык размети HTML5;
* язык стилей CSS3;
* сервер для разработки Node.Js версии не ниже 14.15.4;
* пакетный менеджер yarn версии не ниже 1.22.10;
* интегрированная среда разработки Web Storm версии 2020.3.3.

Также для реализации потребуются некоторые библиотеки и метаязыки:

* библиотека ReactJs;
* метаязык SASS;

При эксплуатации приложения должен быть установлен любой современный браузер (Google Chrome 87+, Edge 87+, Firefox 84+, Safari 13.1+, Opera 72+).

* 1. Требования к составу и параметрам технических средств, применяемых при разработке системы, с одной стороны, и при использовании системы, с другой стороны

При разработке системы технические средства должны удовлетворять следующим требованиям:

* OC Windows 8 и выше;
* RAM - не менее 2Гб.

При использовании системы технические средства должны удовлетворять следующим требованиям:

* ОС Microsoft Windows (7, 8, 8.1, 10) или Apple OS X (macOS) (10.9-10.14);
* Веб-браузер (Google Chrome 87+, Edge 87+, Firefox 84+, Safari 13.1+, Opera 72+);
* RAM - не менее 2Гб.

1. Требования к методам тестирования

Тестирование должно быть проведено на следующих уровнях:

Разрабатываемое веб-приложение будет тестироваться с помощью технологий белого и чёрного ящиков.

С помощью метода белого ящика будут проведены:

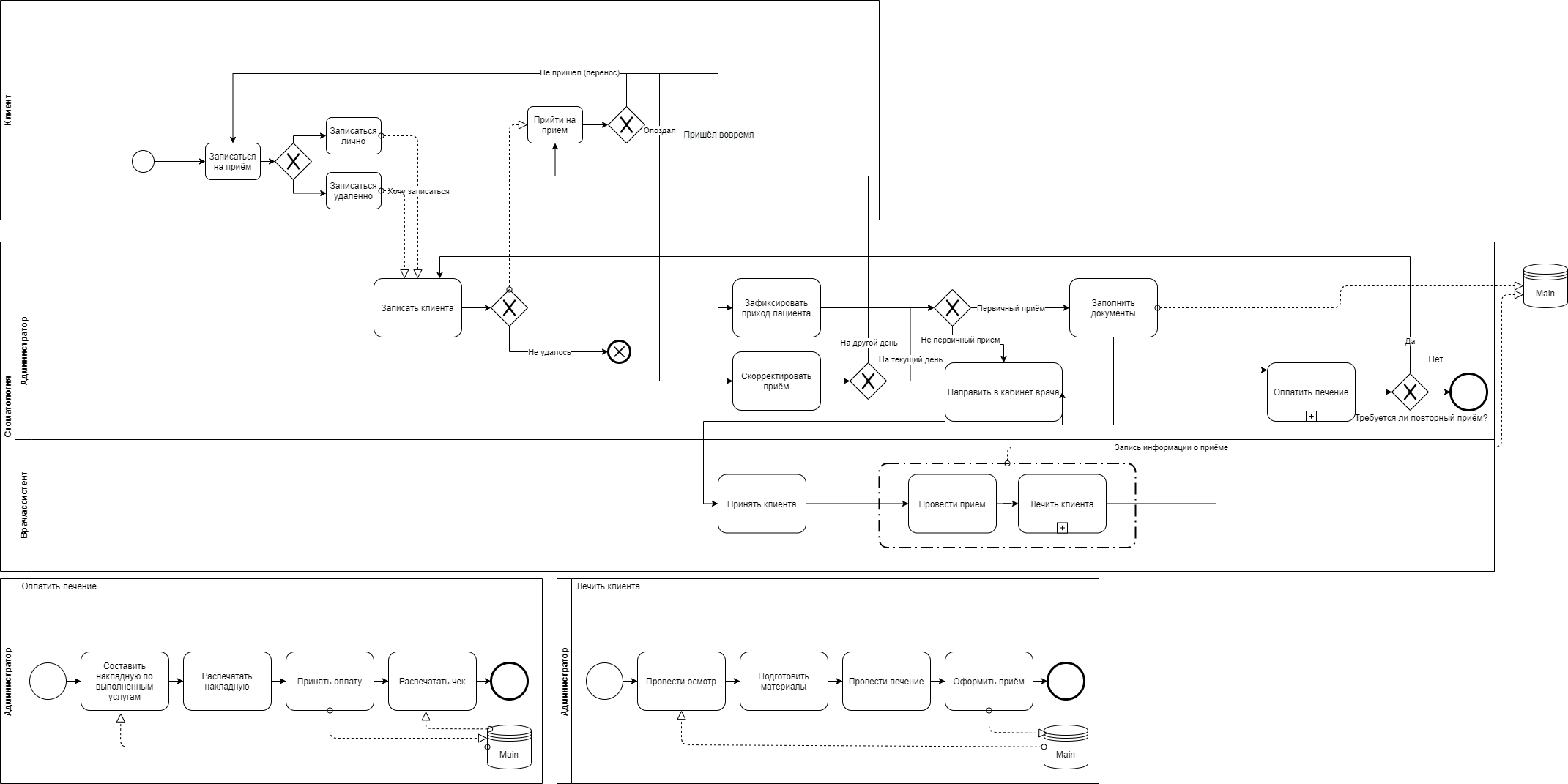
* модульное тестирование;
* интеграционное тестирование.

Тестирование веб-приложений проходит через несколько этапов [2]:

* на этапе функционального тестирования проверяются:
  + функциональные требования (с помощью метода чёрного ящика);
  + загрузка требуемых файлов (с помощью метода чёрного ящика);
* на этапе тестирования вёрстки проверяются:
  + соответствие вёрстки макету (чёрный ящик);
  + оптимизация графических изображений (чёрный ящик);
  + валидность кода [6] (белый ящик);
  + совместимость (чёрный ящик);
* на этапе тестирования удобства интерфейса:
  + собираются данные и отзывы пользователей (чёрный ящик);
  + анализ собранных данных и применение результатов анализа в последующей разработке;
* на этапе тестирования безопасности:
  + тестирование быстродействия сайта [7].

Приложение Б

Модель приложения

Рисунок Б.1 - Функциональная модель системы в нотации BPMN

Изображение выглядит как стол

Автоматически созданное описание

Рисунок Б.2 – Концептуальная модель: диагрмма User Story

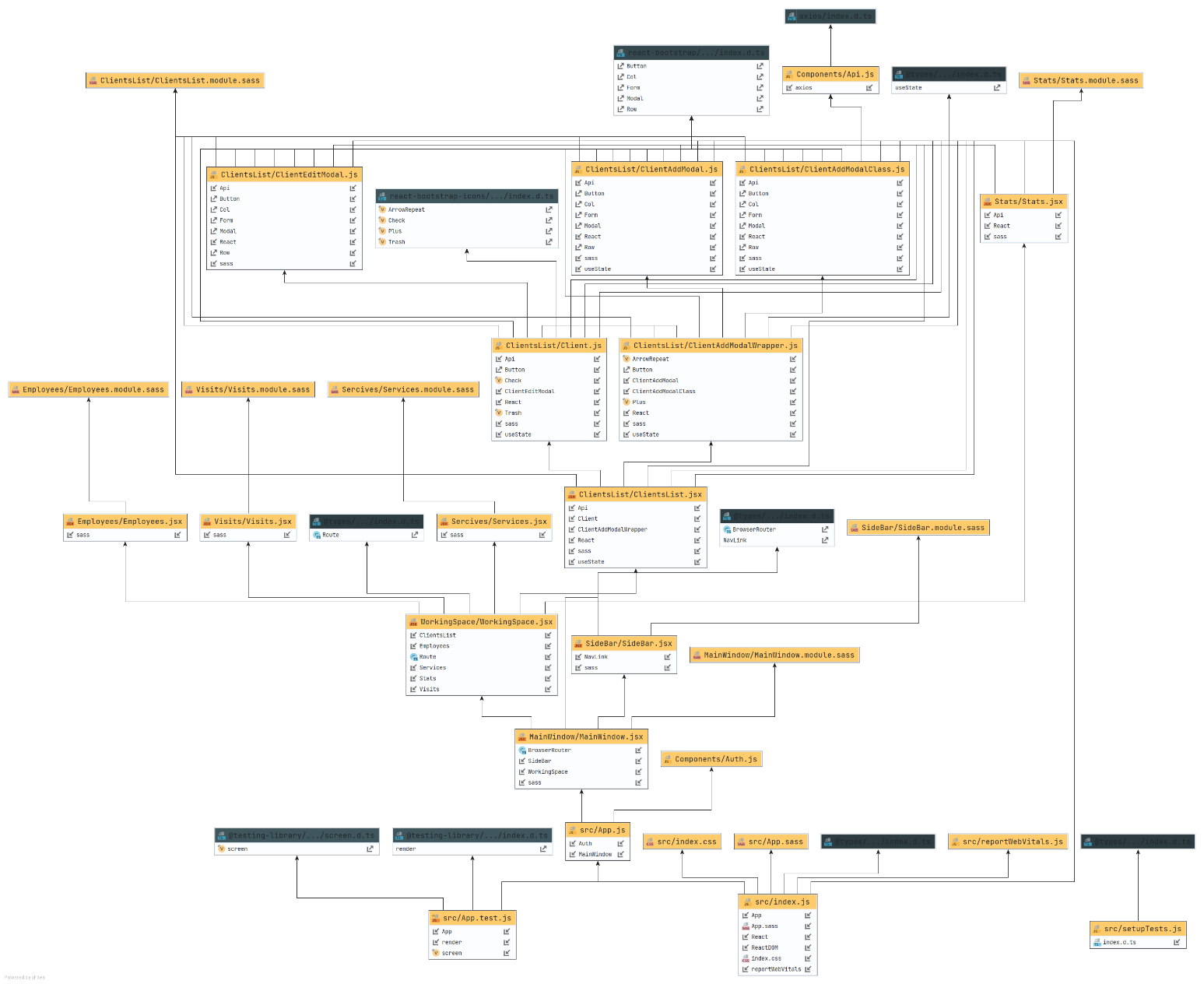


Рисунок Б.3 – Концептуальная модель: диаграмма классов моделей сущностей

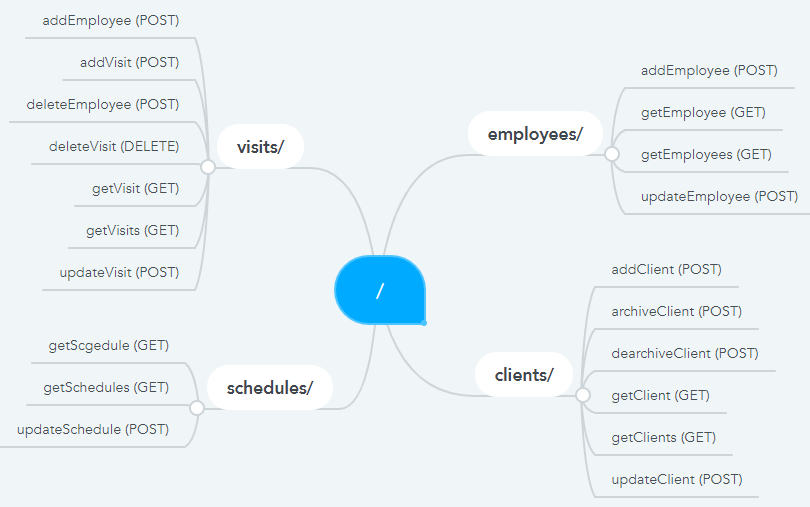


Рисунок Б.4 – Модель прикладного программного интерфейса

Приложение В

Техническое описание

1. Общие сведения о разработке
   1. Наименование программы

АИС для медицинской клиники «Стоматология Доктора Вознесенского» (клиентская часть).

* 1. Программное обеспечение, необходимое для функционирования программы

Для функционирования программы необходимы следующие веб-браузеры:

* Google Chrome, версии 90.0.4430.212+;
* Mozilla Firefox Developer Edition, версии 89.0 x64+;
* Microsoft Edge, версии 90.0.818.56+;
* Safari, версии 14.1+;
* Opera, версии 76.0.4017.123+.
  1. Программные средства

В приложении использованы парадигмы реактивного, декларативного, объектно(прототипно)-ориентированного и функционального программирования.

Для реализации были использованы:

Во время разработки клиентской части:

NodeJs Server, версии 14.15.4;

Пакетный менеджер Yarn, версии 1.22.10;

Метаязык SASS, версии 1.32.5;

ReactJS, версии 17.0.1;

Bootstrap, версии 4.6.0;

React-bootstrap, версии 1.5.0.

В продакшн версии веб-приложения:

HTML, версии 5.3;

CSS, версии 2.1;

JS, версии ES9 (ECMAScript 2018).

1. Функциональное назначение
   1. Описание решаемых задач

Система предоставляет различный функционал для разных групп пользователей (директор/администратор/врач).

Врач может просматривать своё расписание, просматривать и вносить изменения в медицинские карты пациентов.

Администратор может делать то, что может делать врач, а также вносить изменения в расписание, добавлять новых пациентов, вписывать пациентов в расписание врача (записывать на приём).

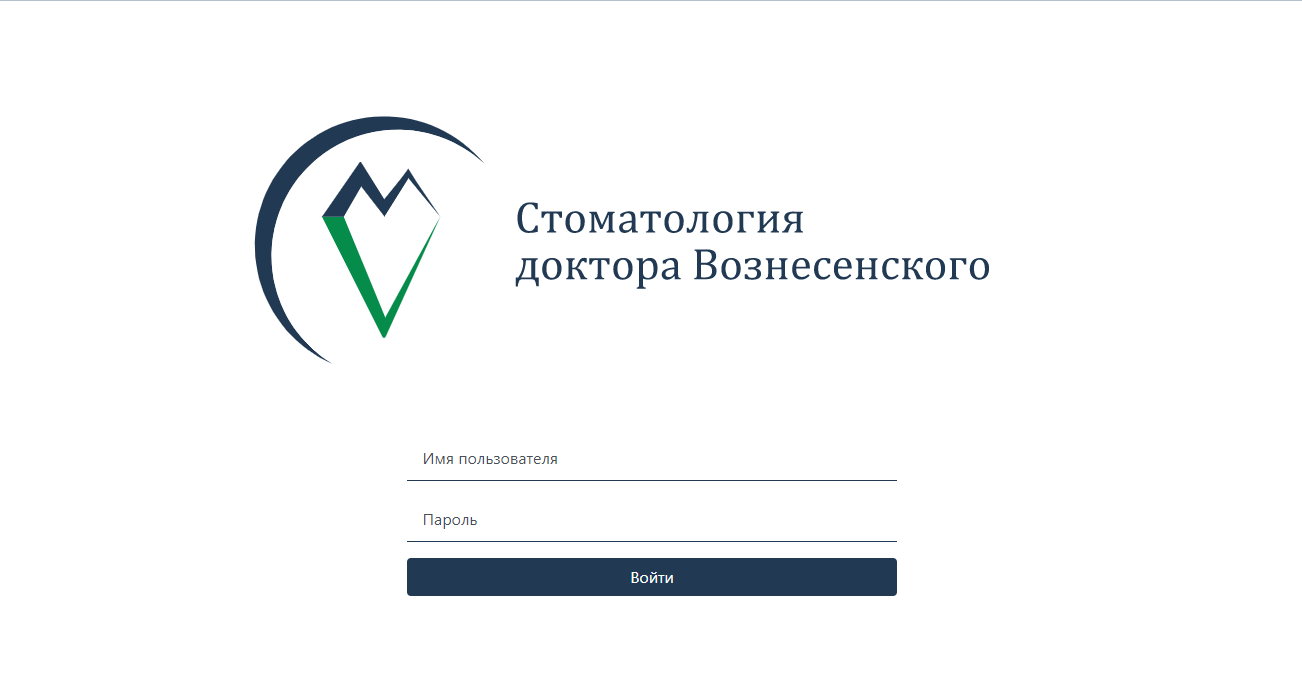
Директору доступен весь функционал других категорий пользователей, а также он может добавлять новые услуги, изменять их цену, добавлять новых пользователей, изменять любые данные пользователей, назначать рабочие дни для администраторов и врачей.

Просмотр расписания реализован посредством таблицы, отображающей календарь с метками врачей, работающих в определённые дни.

* 1. Описание функций

При входе на сайт открывается форма авторизации (рисунок В.1). После нажатия на кнопку «Войти» определяется роль пользователя и осуществляется перенаправление на главную страницу.

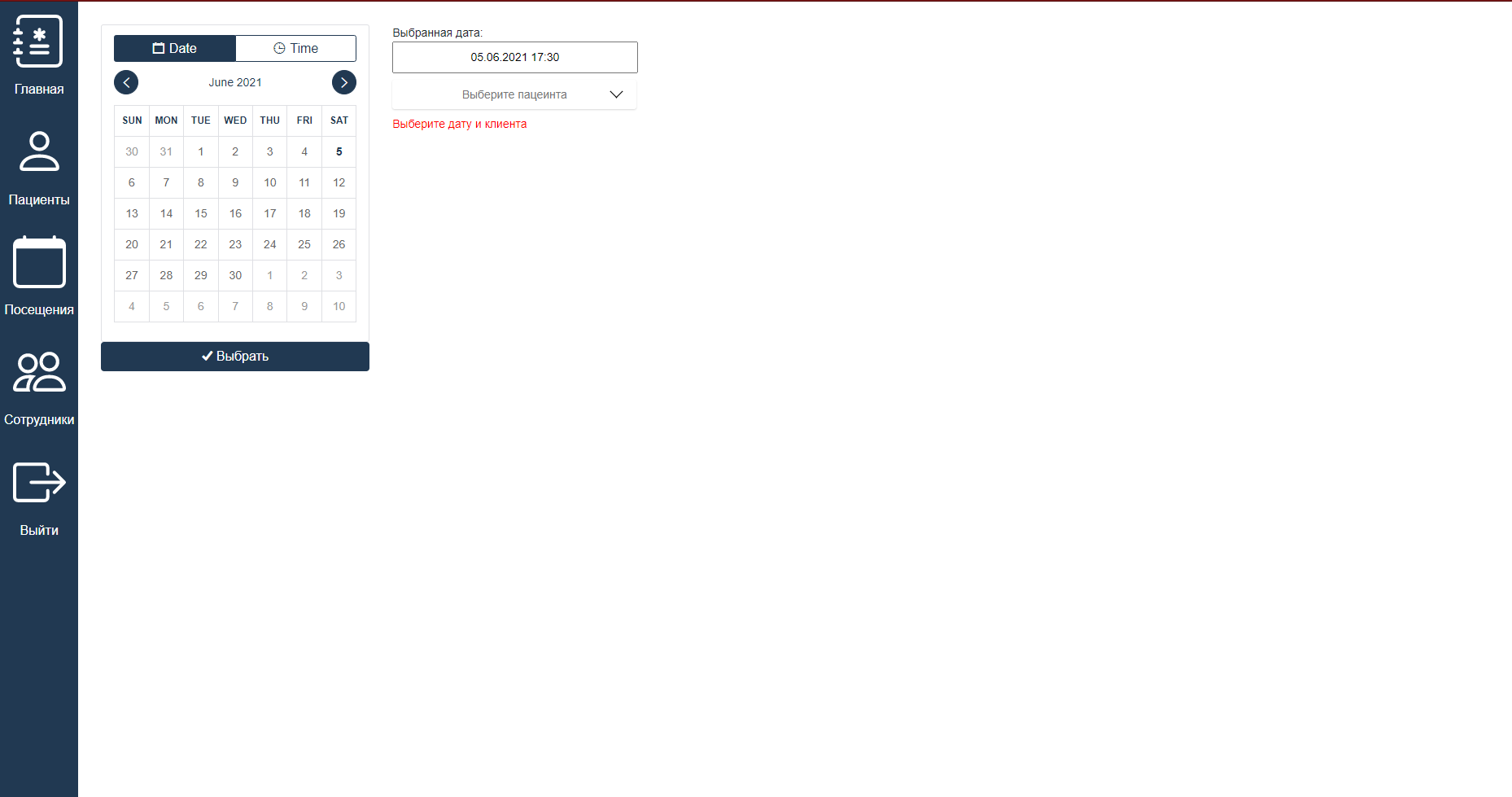
При входе в веб-приложение открывается форма авторизации (рисунок 1). После ввода корректных данных (логин и пароль) и нажатия на кнопку «Войти», исходя из введенных данных, определяется к какой группе прав относится пользователь, в зависимости от неё пользователь будет видеть разные варианты интерфейса (пока реализован только интерфейс директора). Форма входа показана на рисунке В.1.



*Рисунок В.1 – Окно аутентификации пользователей*

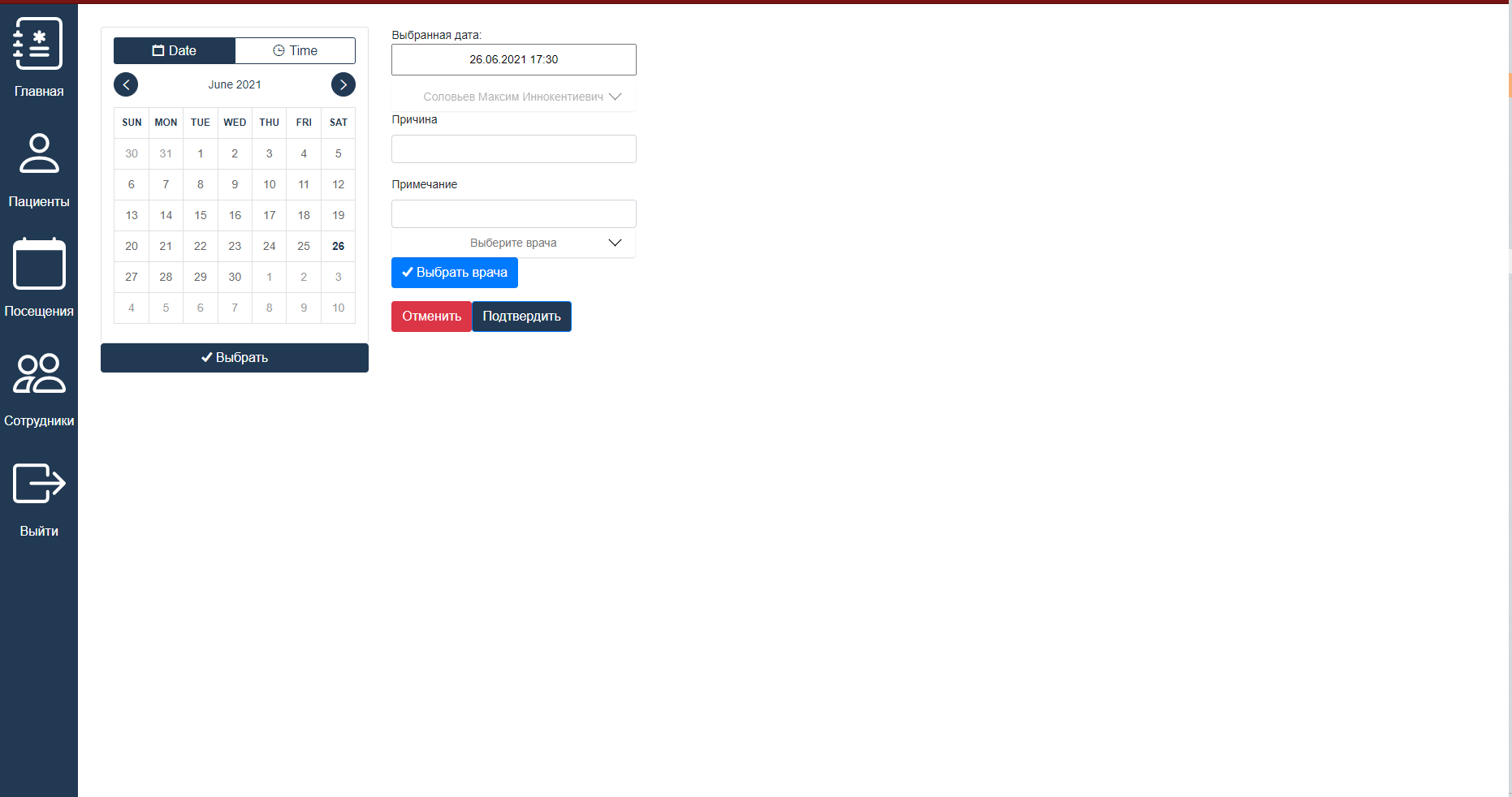
После удачной аутентификации происходит переадресация на главную страницу, на которой можно увидеть меню навигации, позволяющее перейти в другие разделы веб-приложения.

Главная страница представлена на рисунке В.2.



*Рисунок В.2 – Страница «Главная»*

После выбора даты, времени и пациента появится дополнительная форма, представленная на рисунке В.3.



*Рисунок В.3 – Страница «Главная»*

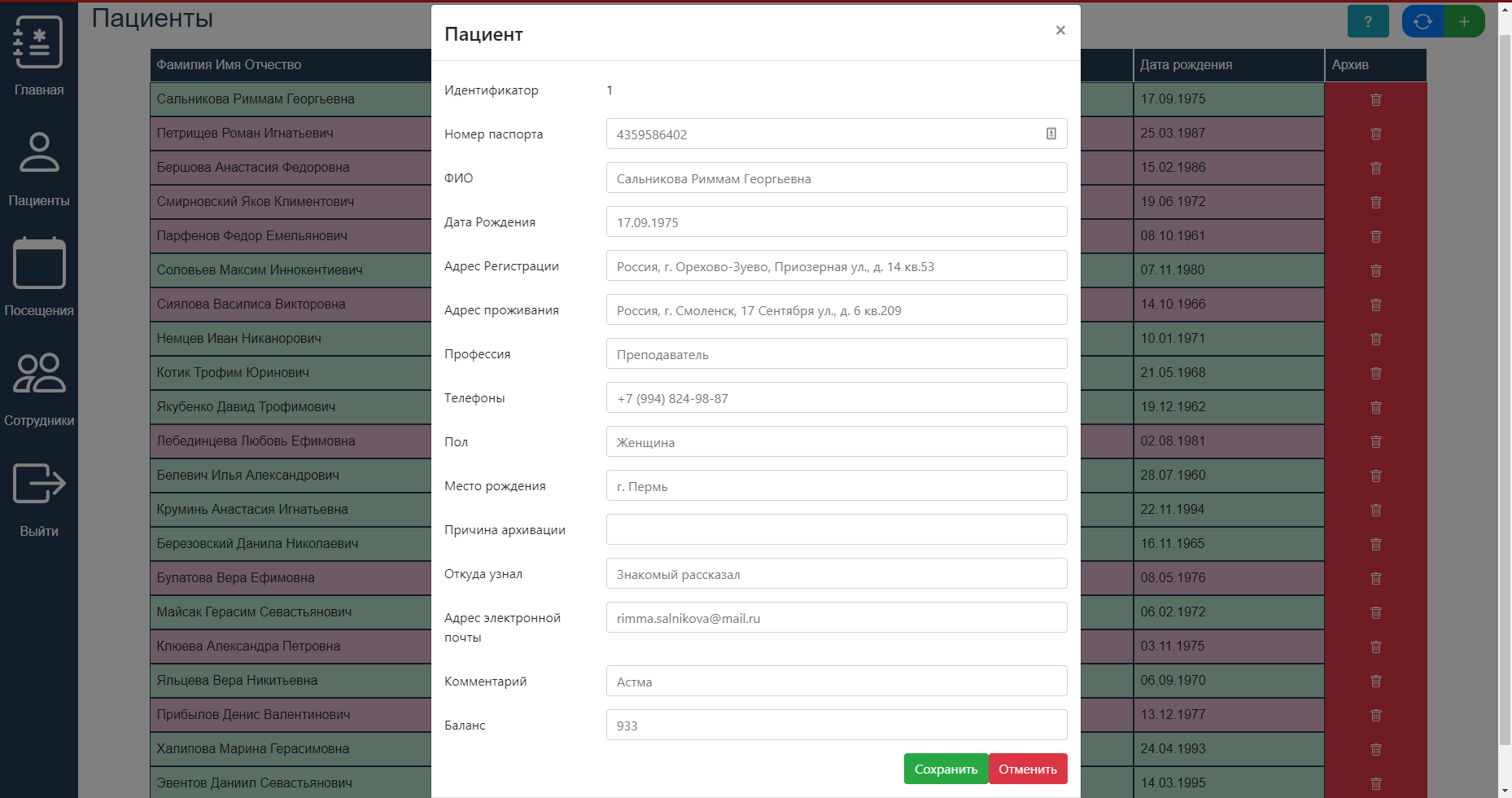
Перейдя по ссылке «Пациенты», открывается окно со списком всех пациентов, которое показано на рисунке В.4.

Изображение выглядит как стол

Автоматически созданное описание

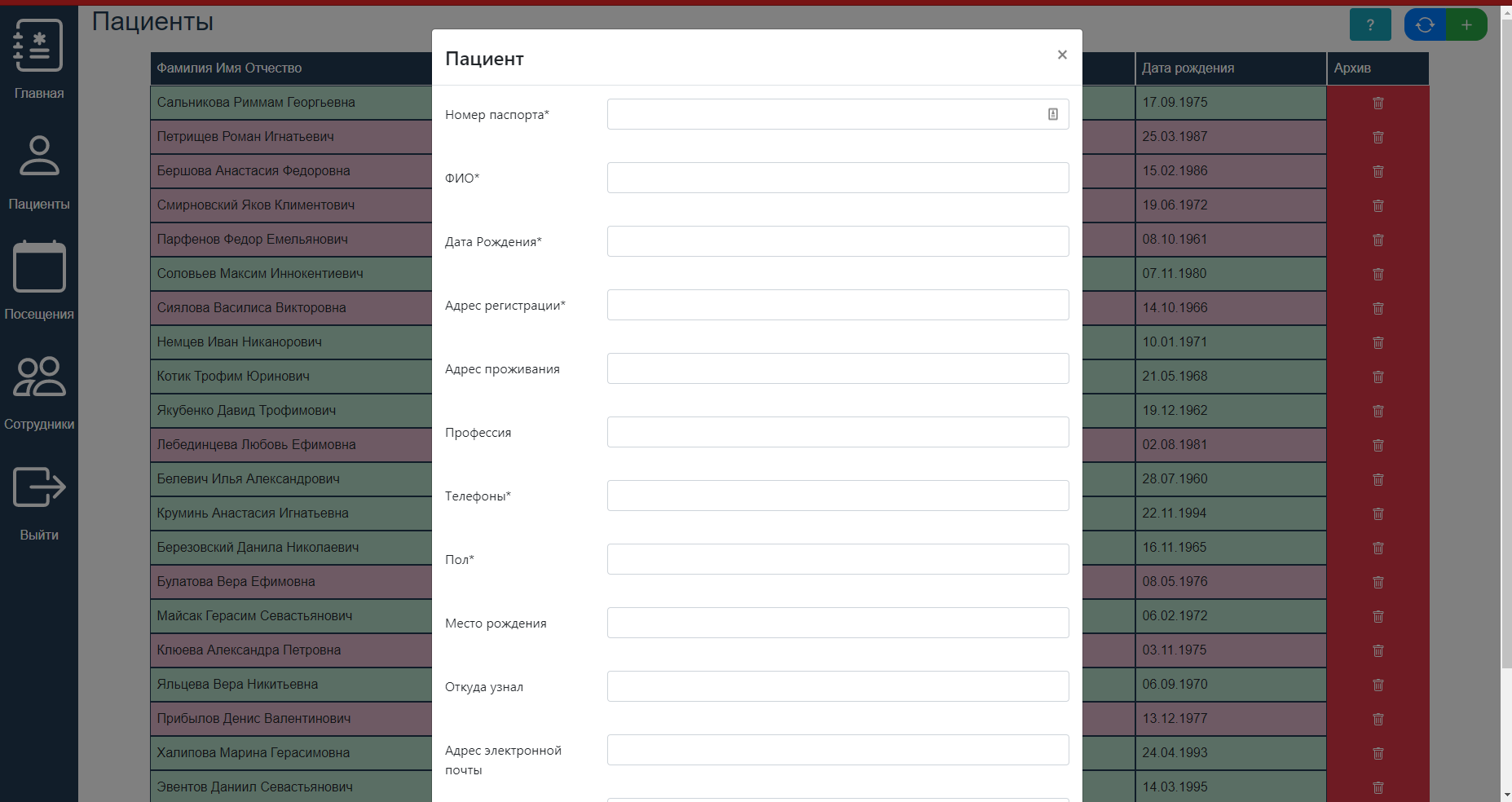
*Рисунок В.4 – Страница «Пациенты»*

При нажатии на строку с конкретным пациентов открывается модальное окно изменения, которое представлено на рисунке В.5.



*Рисунок В.5 – окно изменения пациент*

При нажатии на кнопку «+» появляется форма добавления нового пациента, которая представлена на рисунке В.6.



*Рисунок В.6 – окно добавления пациентов*

При переходе по ссылке «Посещения» выводится страница, которая показанная на рисунках В.7.

Изображение выглядит как стол

Автоматически созданное описание

*Рисунок В.7 – Страница «Посещения»*

При переходе по ссылке «Сотрудники» открывается список всех зарегистрированных сотрудников, представленный на рисунке В.8.

Изображение выглядит как стол

Автоматически созданное описание

*Рисунок В.8 – Страница «Сотрудники»*

При нажатии на строку с конкретным пациентов открывается модальное окно изменения, которое представлено на рисунке В.9.

Изображение выглядит как текст, снимок экрана, внутренний

Автоматически созданное описание

*Рисунок В.9 – окно изменения сотрудников*

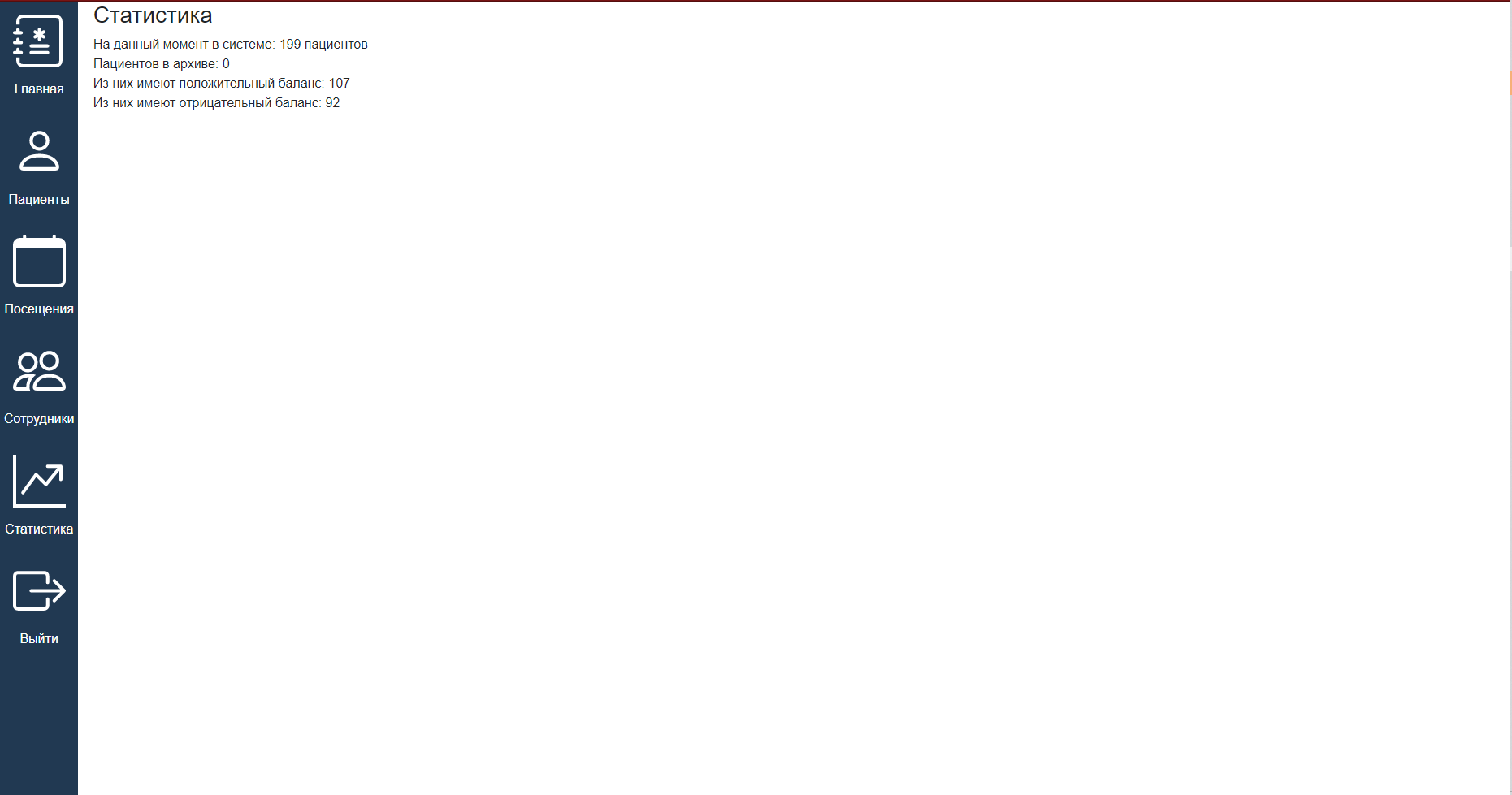
При нажатии на кнопку «+» появляется форма добавления нового сотрудника, которая представлена на рисунке В.10.

Изображение выглядит как текст, снимок экрана, компьютер, ноутбук

Автоматически созданное описание

*Рисунок В.10 – окно добавления сотрудников*

При переходе по ссылке «Статистика» открывается окно с базовой статистикой, которая представлена на рисунке В.11.



*Рисунок В.11 – Страница «Статистика»*

* 1. Ограничения области применения программы

Программа может быть использована только в медицинской клинике «Стоматология Доктора Вознесенского» (Санкт-Петербург, Фрунзенский район).

1. Описание логической структуры
   1. Алгоритм программы

Функциональная модель представлена в BPMN-диаграмме, на рисунке Б.1, включающей основной бизнес-процесс, который описывает процесс записи пациентов на приём, который включает в себя проведение лечения и оплаты. Описание поведения пользователей представлено на рисунке Б.2 в формате «Пользовательская история» (User story).

* 1. Используемые методы при разработке программы

В процессе разработки системы были использованы:

* git – для контроля версий файлов, версии 2.29.2.windows.3;
* https – протокол для обмена данными между клиентом и сервером;
* json – универсальный формат данных;
* yarn – пакетный менеджер, версии 1.22.10;
* sass – мета-язык для каскадных таблиц стилей (css), версии 1.32.5;
* html – стандартизированный язык разметки веб-документов, версии 5.3;
* ESLint – линтер для JavaScript, статический анализатор кода, версии 0.2.1.

При разработке были использованы следующие паттерны проектирования:

* Простые компоненты (stateless component);
* Условный рендеринг (conditional rendering);
* Типы потомков (children types);
* Массив как потомок (array as children);
* Функция как потомок (function as children);
* Функция в render (render callback);
* Перенаправление компонента (proxy component);
* Компоненты высшего порядка (higher-order component);
* Передача свойств вниз по дереву компонентов (passing down props);

После разработки производится сборка и компиляция dev-проекта в продакшн билд, который более оптимизирован для браузеров и работы в интернете: все js- и css-/sass-файлы собираются в набор минимизированных файлов, чтобы браузер загружал сразу весь функционал и не осталось не подгруженных файлов.

* 1. Структура программы

Состав файлов и распределение функций программы представлено в таблице В.1.

Таблица В.1 – Структура программы

|  |  |  |
| --- | --- | --- |
| Файл | Функции | Назначение |
| 1 | 3 | 2 |
| Auth.jsx | Auth | Функция, возвращающая визуальный компонент окна авторизации |
| Auth.setField | Функция, для логирования изменений поля и записи данных в структуру |
| Auth.setAuthorization | Функция, устанавливающая флаг авторизации |
| Auth.module.sass | - | Файл стилей |
| SideBar.jsx | SideBar | Возвращает визуальный компонент бокового меню |
| SideBar.module.sass | - | Файл стилей |

Продолжение таблицы В.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Client.jsx | Client | Функция, возвращающая визуальный интерактивный компонент с информацией о пациенте |
| Client.archiveClient | Функция архивации клиентов |
| Client.dearchiveClient | Функция разархивации клиентов |
| ClientList.jsx | ClientList.componentDidMount | Функция, запускающаяся в момент загрузки компонента Client, которая вызывает функцию getClietns |
| ClientList.getClients | Функция, получающая данные пользователей из Базы Данных по средствам API |
| ClietnList.render | Функция, возвращающая разметку списка клиентов и передающая данные в класс Client |
| ClientList.module.sass | - | Файл стилей |
| ClientAddModalWrapper.jsx | ClientAddModalWrapper | Функция-обёртка для модальных окон, добавляющая возможность обновления и добавления клиентов |
| ClientEditModal.jsx | ClientEditModal | Функция, которая возвращает модальное окно с формой изменения существующих пациентов |
| ClientEditModal.setField | Функция, для логирования изменений поля и записи данных в структуру |
| ClientEditModal.handleSubmit | Функция, собирающая данные и проверяющая на наличие ошибок |
| ClientEditModal.findFromErrors | Функция поиска ошибок во введённых данных |

Продолжение таблицы В.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| ClientAddModalClass.jsx | ClientAddModal | Функция, которая возвращает модальное окно с формой добавление пациентов |
| ClientAddModal.setField | Функция для логирования изменений поля и записи данных в структуру |
| ClientAddModal.handleSubmit | Функция, собирающая данные и проверяющая на наличие ошибок |
| ClientAddModal.findFromErrors | Функция поиска ошибок во введённых данных |
| Employee.module.sass | - | Файл стилей |
| EmployeesList.jsx | Employee | Функция, возвращающая интерактивный визуальный компонент со списком сотрудников |
| EmployeesList.module.sass | - | Файл стилей |
| EmployeeAddModal.jsx | EmployeeAddModal | Функция, которая возвращает модальное окно с формой добавление сотрудников |
| EmployeeAddModal.setField | Функция, для логирования изменений поля и записи данных в структуру |
| ClientAddModal.handleSubmit | Функция, собирающая данные и проверяющая на наличие ошибок |
| ClientAddModal.findFromErrors | Функция поиска ошибок во введённых данных |
| EmployeeAddModalWrapper.jsx | EmployeeAddModalWrapper | Функция-обёртка для модальных окон, добавляющая возможность обновления и добавления сотрудников |

Продолжение таблицы В.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| EmployeeEditModal.jsx | ClientEditModal | Функция, которая возвращает модальное окно с формой изменения существующих сотрудников |
| ClientEditModal.setField | Функция, для логирования изменений поля и записи данных в структуру |
| ClientEditModal.handleSubmit | Функция, собирающая данные и проверяющая на наличие ошибок |
| ClientEditModal.findFromErrors | Функция поиска ошибок во введённых данных |
| Employee.jsx | Employee | Функция, возвращающая интерактивный визуальный компонент интерфейса, которые выводит таблицу всех сотрудников |
| SideBar.module.sass | - | Файл стилей |
| SideBar.jsx | SideBar | Функция, возвращающая разметку меню боковой навигации |
| Services.module.sass | - | Файл стилей |
| Stats.jsx |  | Функция, возвращающая статистические данные |
| Visits.jsx | Visits | Функция, возвращающая список визитов |
| Visits.module.sass | - | Файл стилей |
| WorkingSpace.jsx | WorkingSpace | Функция-маршрутизатор, определяет, что должно быть выведено, в зависимости от адреса |
| WorkingSpace.module.sass | - | Файл стилей |
| Services.jsx | Service | Функция, возвращающая список всех предоставляемых услуг |
| Stats.module.sass | - | Файл стилей |
| Api.js | axios.create | Обёртка функции из библиотеки axios |
| App.js | App | Функция-маршрутизатор, которая не позволяет неавторизованным пользователям получать доступ к системе |

Продолжение таблицы В.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| App.sass | - | Файл стилей |
| index.css | - | Файл стилей |
| index.js | ReactDom.render | Функция, запускающая всё веб-приложение |
| package.json |  | Файл, в котором перечислены зависимости |
| yarn.lock |  | Файл, в котором перечислены зафиксированные версии зависимостей |
| .gitignore |  | Файл, в котором перечислены игнорируемые файлы |
| public/index.html | - | Файл разметки стартового веб-документа |
| public/index.css | - | Файл стилей |

В таблице В.2 представлена структура программы после продакшн сборки.

Таблица В.2 – Структура клиентской части веб-приложения

|  |  |  |
| --- | --- | --- |
| Файл | Функции | Назначение |
| index.html | - | Файл разметки стартового веб-документа |
| index.css | - | Файл стилей |
| static/css/\*.\*.chunk.css | - | Минимизированные файлы стилей |
| static/js/\*.\*.chunk.js | Их слишком много, чтобы это было возможным описать | Минимизированные файлы js, содержащий основной функционал |
| static/js/\*.\*.chunk.js.LICENSE | - | Файлы лицензий |
| static/js/main.\*.chunk.js | Их слишком много, чтобы это было возможным описать | Минимизированный файл webpack-скрипта, содержащий фундаментальные настройки сборки окружения |
| static/js/runtime-main.\*.js | Их слишком много, чтобы это было возможным описать | Минимизированный файл js |

* 1. Форматы обмена данными

Основной формат обмена данными – обращение к Rest API в виде JSON-запросов. Схема API представлена на рисунке Б.4.

1. Сведения о входных и выходных данных

Входные и выходные данные приложения описаны в приложения А «Техническое задание» в разделе 3.2.

1. Используемые технические средства

Для использования разрабатываемого веб-приложения устройство пользователя должно обладать следующими техническими требованиями, соответствующим требованиям использования основных браузеров (Chrome, Firefox, Internet Explorer, Opera, Safari и т.п.) (URL: <https://support.google.com/chrome/a/answer/7100626>, дата обращения: 18.05.21):

* ОС Windows 7 (поддержка завершится не ранее 15.01.2022), Windows 8, Windows 8.1, Windows 10 или более поздней версии или OS X El Capitan 10.11 или более поздней версии.
* процессор Intel Pentium 4 или более поздней версии с поддержкой SSE3.

1. Вызов и загрузка программы
   1. Конфигурирование и загрузка программы

Развертывание системы производится по следующему алгоритму:

1. Запуск сервера компонента логирования;
2. Запуск сервера компонента хранения;
3. Запуск компонента API;
4. Запуск веб-сервера.

Алгоритм также может быть описан следующей последовательностью команд (при использовании описанных в разделе 5 технических средств):

systemctl influxd start;

systemctl mysqld start;

java -jar ./voz-backend.jar;

systemctl nginx start.

* 1. Способ вызова и завершения работы программы

Разрабатываемая программа – веб-приложение, поэтому для начала работы требуется перейти на определённый URL, который будет назначен заказчиком на рабочем месте.

Для прекращения работы необходимо покинуть браузер или вкладку, в которой открыто веб-приложение.

* 1. Режим работы программы

Пользователи взаимодействуют с веб-приложением в интерактивном режиме.