1. Анализ поставленной задачи
   1. Формулировка поставленной задачи

Целью проекта является разработка клиентской части АИС для стоматологической клиники «Доктор Вознесенский» (url: <https://doc-stom.ru/>, дата обращения: 04.02.2021), которая позволит автоматизировать учёта клиентов и оказанных им услуг.

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.

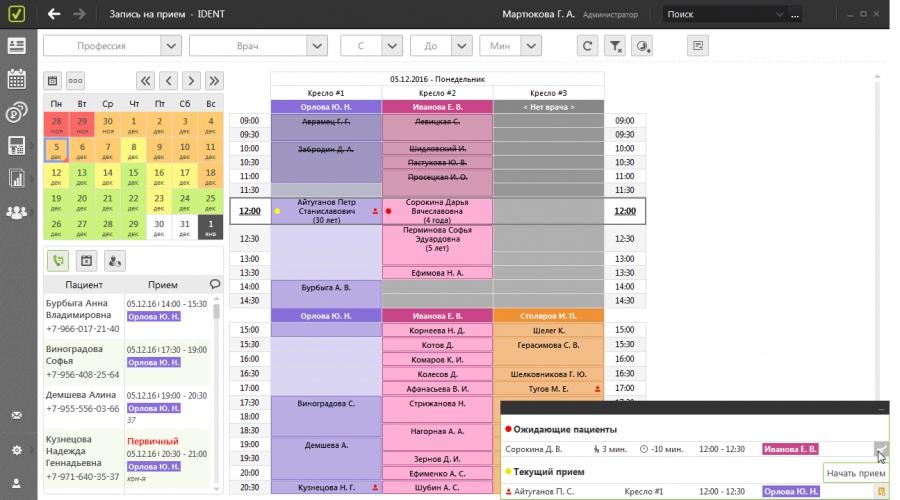
Так как большая часть действий в рабочем процессе связана с манипуляциями данными - заказчик использует для их выполнения АИС. Но используемая в данный момент система обладает рядом недостатков, главными из которых являются высокая стоимость системы, отсутствие мобильного приложения для работы с АИС, отсутствие интеграции с IP-телефонией.

* 1. Обзор и сравнение аналогичных решений

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

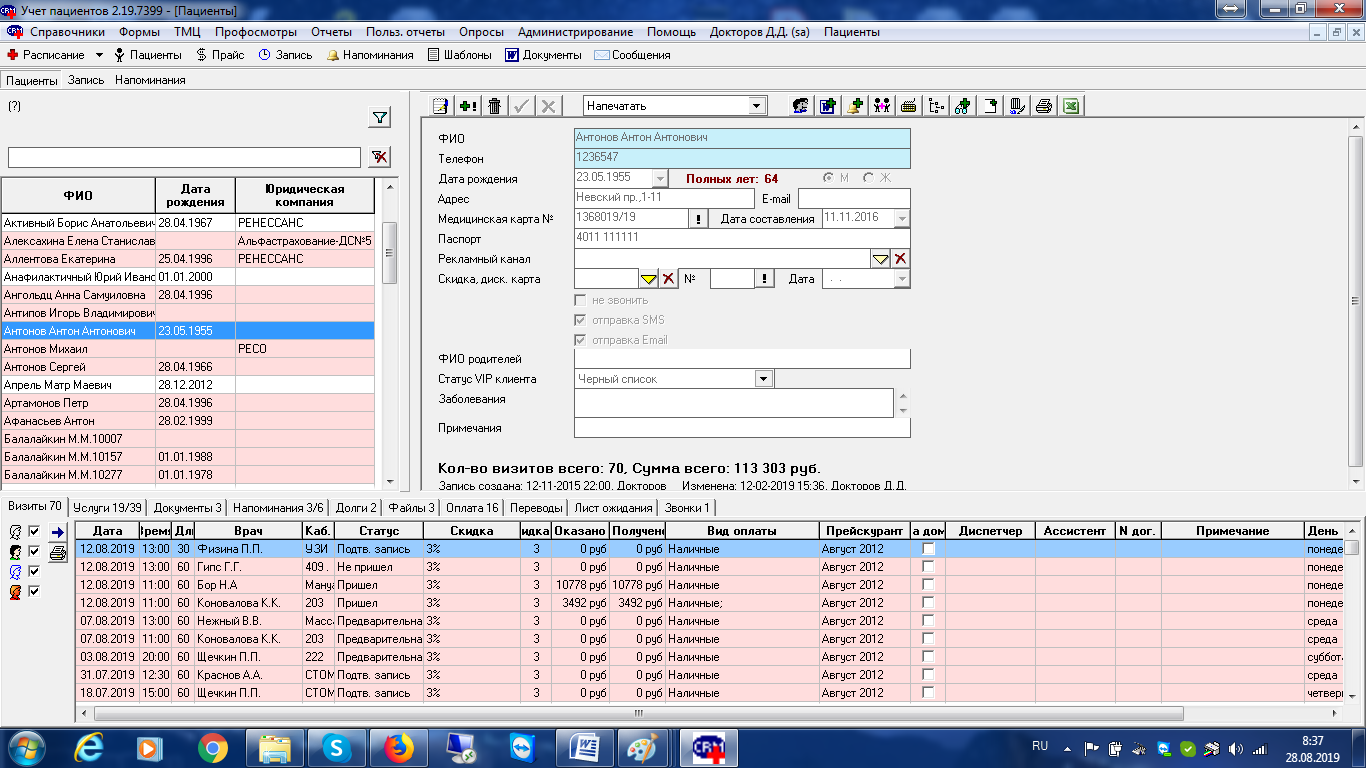
Для сравнения были выбраны использовавшиеся заказчиком системы «iDent» (url: <https://dent-it.ru/>, дата обращения: 04.02.2021) и МИС «Учет пациентов» от «Dr. Soft» (url: <https://crm-soft.ru/>, дата обращения 04.02.2021), а также системы «iStom» (url: <https://i-stom.ru/>, дата обращения 04.02.2021) 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;



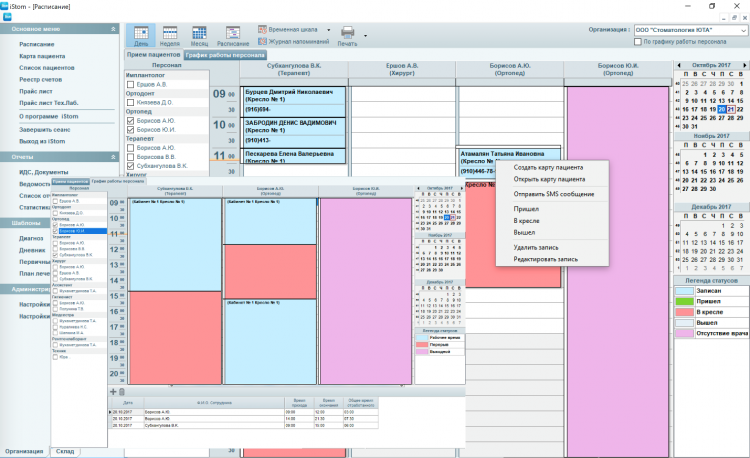
*Рисунок 1 – «Интерфейс программы «Ident»*

* 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;



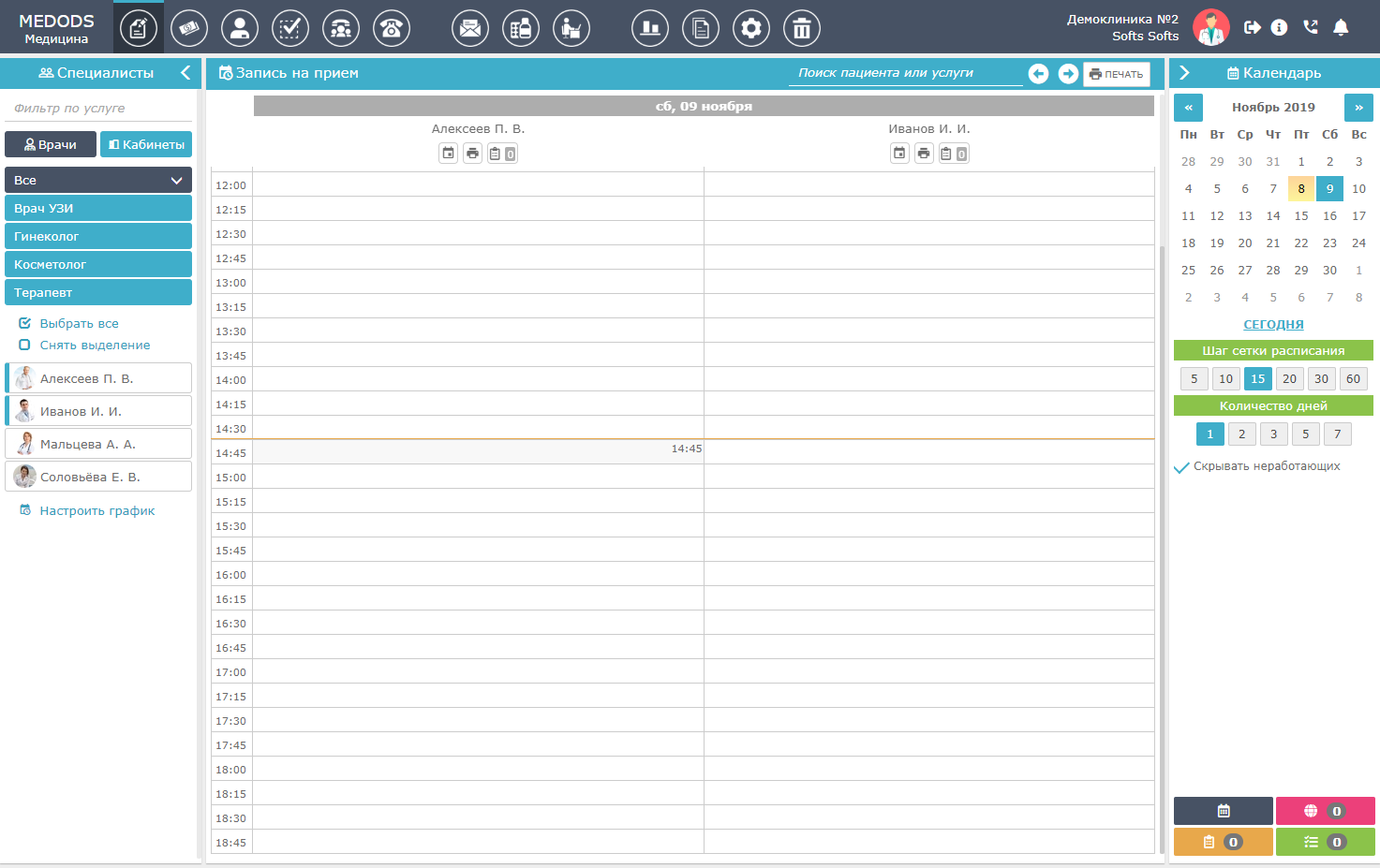
*Рисунок 2 – «Интерфейс программы «Учёт пациентов»*

* 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, создать возможность пользоваться системой удалённо, требуется реализовать авторизацию, ведение карт пациентов, запись пациентов на приём, ведение отчётности вносимых средств пациентами, печать документом и чеков, сбор статистики. При этом всем веб-сервис должен предоставлять исчерпывающую информацию об услугах.

* 1. Функциональные требования к разрабатываемому приложению

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

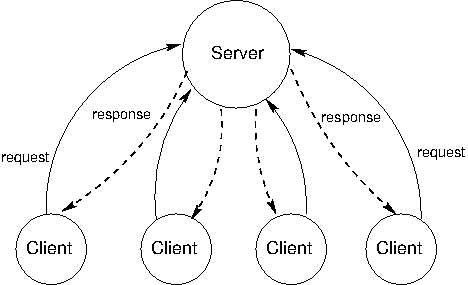
* авторизация;
* ведение карты пациента;
* запись пациентов на приём;
* ведение отчётности вносимых средств пациентами;
* печать документов;
* сбор статистики.

1. Проектирование 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;
* печать документов;
  + входные данные: информация о пациенте;
  + выходные данные: печать документа по форме;
* сбор статистики;
  + входные данные: информация о пациентах;
  + выходные данные: графики и сводки о работе стоматологической клиники;
* печать чеков;
  + входные данные: внесение денежных средств.
  1. Выбор архитектуры системы

Веб-приложения, может быть, спроектировано с помощью различных шаблонов архитектур: клиент-сервер, MVC (Model-View-Controller, Модель-Представление-Контроллер), многоуровневый шаблон, шаблон посредника. Так как при разработке будет использовать один сервер, то самым подходящим шаблоном является двухзвенный шаблон «клиент-сервер» с тонким клиентом. Приблизительная структура системы представлена на рисунке 5.



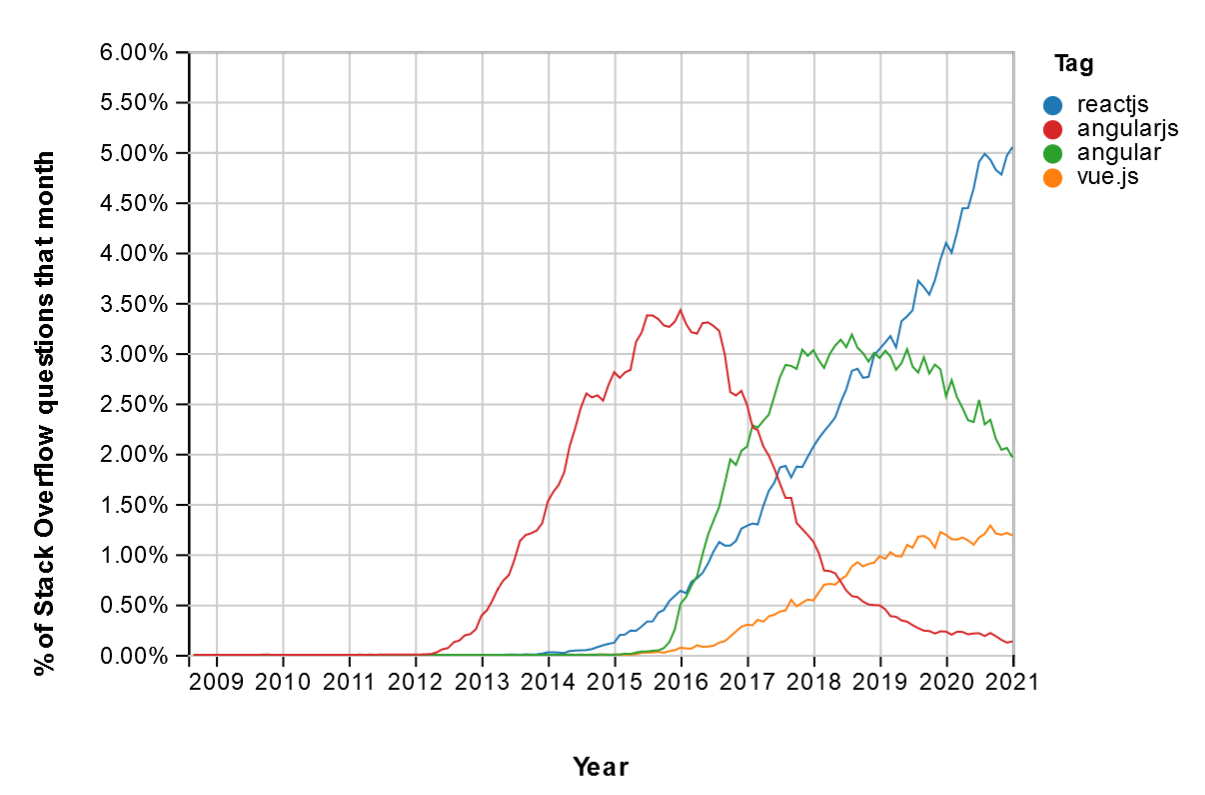
*Рисунок 5 – Архитектурны шаблон «Клиент-Сервер»*

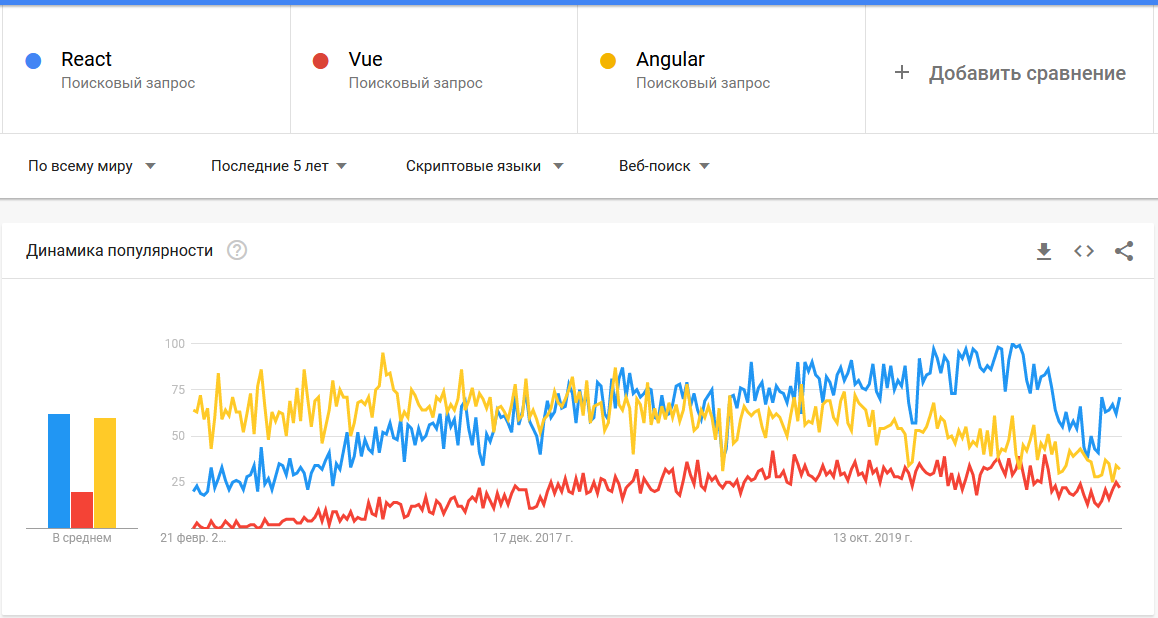
* 1. Выбор и обоснование средств и технологий реализации

Языка программирования JavaScript широко используется для реализации взаимодействия пользователя с веб-сайтами и приложениями [1]. Для упрощения разработки используются различные библиотеки и фреймворки:

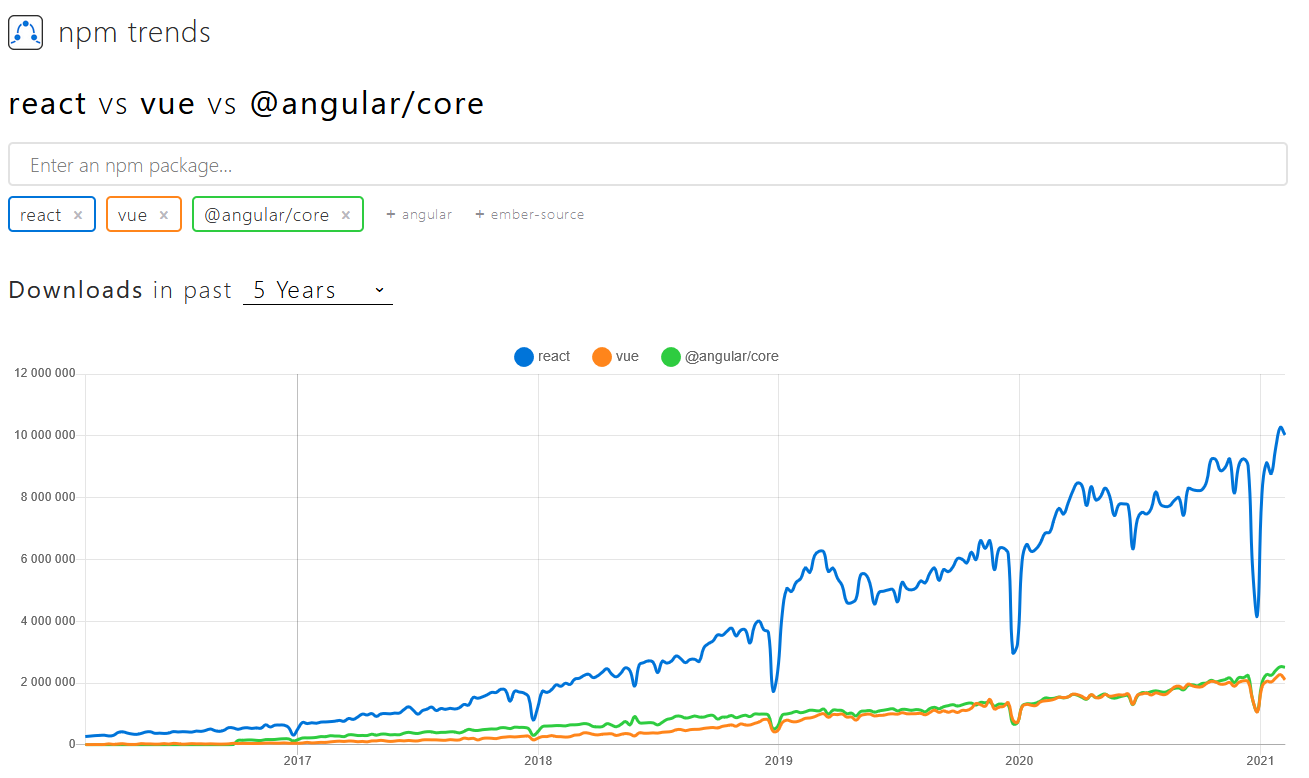
* фреймворк VueJs;
* фреймворк AngularJs;
* библиотека ReactJs.

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 6, 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 7, на площадке npm trends (url: <https://www.npmtrends.com/react-vs-vue-vs-@angular/core>, дата обращения 15.02.2020) на рисунке 8.

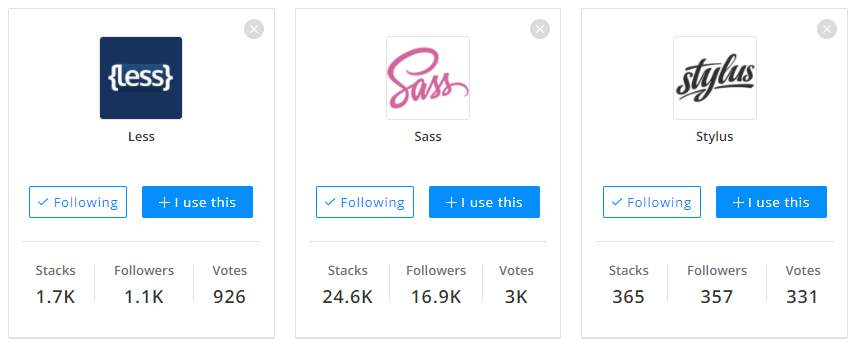


*Рисунок 6 – Сравнение частоты запросов на Stack Overflow.*

*Рисунок 7 – Сравнение частоты запросов в поисковой системе Google.*

*Рисунок 8 – 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 8 (url: <https://stackshare.io/stackups/less-vs-sass-vs-stylus>, дата обращения 15.02.2020).



*Рисунок 8 – Сравнение частоты запросов.*

Исходя из данных на рисунке 8 можно сделать вывод, что популярность препроцессора Sass намного выше, чем популярность Less и Stylus. Scss является частью Sass поэтому он отдельно не рассматривается.

Итого, приложение должно быть реализовано при помощи следующих средств:

* язык программирования JavaScript версии не ниже ES6;
* библиотека 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 [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 [3] (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 [4];
  + тестирование безопасности по OWASP testing guide [6];

Использованные источники

Веб-технологии для разработчиков | MDN // MDN Web Docs, URL: <https://developer.mozilla.org/ru/docs/Web> (дата обращения: 20.03.2021);

Методика тестирования web приложения // Тестирование Программного Обеспечения, URL: <https://software-testing.org/testing/metodika-testirovaniya-web-prilozheniya.html> (дата обращения: 02.03.2021);

Markup Validation Service, URL: <https://validator.w3.org> (дата обращения 23.03.2021);

PageSpeed Insights, URL: <https://developers.google.com/speed/pagespeed/insights/?hl=ru> (дата обращения: 23.03.2021);

Онлайн валидатор html // involta, URL: <https://involta.ru/tools/validator-html/> (дата обращения: 23.03.2021);

OWASP Web Security Testing Guide // OWASP, URL: <https://owasp.org/www-project-web-security-testing-guide/> (дата обращения: 23.03.2021).

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 программы является предоставление заказчику возможности автоматизации учёта пациентов стоматологической клиники (далее СК).

Программа должна эксплуатироваться на объекте заказчика (СК). Пользователями программы должны являться администраторы, директор, врачи.

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:

* разработка требований к системе;
* детальное проектирование системы;
* программная реализация;
* тестирование.

1. ТРЕБОВАНИЯ К системе
   1. Требования к функциональным характеристикам

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 товаров и материалов на складе;
    - 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;
    - учет дисконтных карт клиентов;
    - 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;
* постоянная скидка.

О визитах:

* пациент;
* дата;
* статус первичности;
* 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;
* фактическая цена;
* посещение, в которое услуга оказана.

О товарах:

* цена;
* наименование;
* группа;
* количество.

О проданных товарах:

* товар;
* клиент;
* фактическая цена;
* дата продажи;
* факт использования дисконтной карты;

О дисконтных картах:

* клиент;
* размер скидки.

О транзакциях:

* клиент;
* изменение баланса.
  1. Модель приложения

Функциональная модель поведения приложения должна соответствовать модели на рисунках %number of picture%.

Модель поведения пользователей должна соответствовать модели, представленной на рисунке %number of picture%.

Модель прикладного программного интерфейса должна соответствовать модели на рисунке %number of picture%. и в таблицах %number of picture%.

* 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).

* 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;
* услуги – модуль, содержащий информацию о возможных услугах, предоставляемых клиникой;
* статистика – модуль, содержащий статистическую информацию о работе клиники и о клиентах.

* 1. Требования к информационному обеспечению системы

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

* 1. Требования к технологиям разработки

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

* 1. Требования к программным средствам и технологиям разработки

Компонента клиентского интерфейса должно быть реализовано при помощи следующих средств:

* язык программирования JavaScript (версии ECMAScript 6 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 должны быть установлены:

* любой современный браузер (Mozilla Firefox, Opera, Google Chrome, Safari, Yandex Browser).
  1. Требования к защите от несанкционированного доступа

Контроль защиты информации от несанкционированного доступа осуществляется средствами СУБД. Правила валидации полей также определяются СУБД.

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.

1. ТРЕБОВАНИЯ К МЕТОДАМ 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 [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 [3] (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 [4];
  + тестирование безопасности по OWASP testing guide [6];

Backend:

Любая программа подлежит тестированию на этапе разработки. Разрабатываемая система будет тестироваться с помощью технологии белого ящика и включать:

* модульное тестирование;
* интеграционное тестирование.

Тестирование системы проходит через несколько этапов:

* функциональное тестирование;
* тестирование безопасности;
* нагрузочное тестирования.