Оглавление

[Введение 3](#_Toc514795040)

[1. Постановка задачи 4](#_Toc514795041)

[1.1 Актуальность темы 4](#_Toc514795042)

[1.2 Цель работы 5](#_Toc514795043)

[1.3 Задачи, подлежащие решению для достижения цели 5](#_Toc514795044)

[1.4 Методология, средства решения задач 5](#_Toc514795045)

[1.5 Стартовые данные для выполнения работы 5](#_Toc514795046)

[2. Обзор существующих аналогов 5](#_Toc514795047)

[2.1 Общие требования и процесс получения консультаций 6](#_Toc514795048)

[2.2 Яндекс.Здоровье 6](#_Toc514795049)

[2.3 Doc+ Онлайн 7](#_Toc514795050)

[2.4 Доктор рядом 7](#_Toc514795051)

[2.5 Онлайн Доктор 8](#_Toc514795052)

[2.6 Итоги сравнения 8](#_Toc514795053)

[3. Функционал приложения 10](#_Toc514795054)

[3.1 Аутентификация, авторизация, регистрация 10](#_Toc514795055)

[3.2 Обзор главных страниц 14](#_Toc514795056)

[3.2 Пути создания консультаций 16](#_Toc514795057)

[3.3 Окно консультаций 17](#_Toc514795058)

[3.4 Тестирование требований 18](#_Toc514795059)

[4. Архитектура и реализация функционала приложения 18](#_Toc514795060)

# Введение

В последние годы с появлением быстрого, стабильного и доступного интернета, мощных телефонов и компьютеров появилась возможность создавать автоматизированные сервисы для уменьшения стоимости услуг путем ликвидации промежуточных, ненужных звеньев в бизнес-процессах. Например, сервисы заказа такси, такие как Uber, Yandex Taxi и Get пришли на смену традиционным диспетчерским службам всего за несколько лет и стоимость услуг такси снизилась на порядок, а качество обслуживания в свою очередь только улучшилось. На данный момент активно развивается рынок оказания медицинских услуг удаленно. Конечно, полностью очную медицину подобный сервис на ликвидирует, однако однозначно поменяет представление об оказании медицинских услуг. Сервис удаленных консультаций поможет пациентам, проживающим в труднодоступных местах, неподвижным пациентам, пациентам, кому необходима срочная мед. консультация. Для получения услуг нет необходимости стоять в очереди, необходимо лишь записаться на прием, либо связаться со свободным врачом в данный момент. Также данный сервис потенциально может сохранять историю пациента, его обращения, рекомендации от врачей, и, даже, анализировать состояние здоровья пациента и делать на основе этого автоматические рекомендации.

# Постановка задачи

## Актуальность темы

Востребованность телемедицинских услуг становится очевидной при рассмотрении результатов исследования, проведенного в высококвалифицированном учреждении – Институте хирургии им. А.В. Вишневского РАМН. Из 12–15 тыс. обратившихся за год больных лишь 10% нуждались в госпитализации и хирургическом лечении. Следовательно, остальные 90% обратившихся тратили деньги, и немалые, время и здоровье лишь на то, чтобы получить квалифицированную консультацию. в США телемедицину начали внедрять в 2008 году. Но уже через восемь лет, к 2016 году, в стране уменьшилась обращаемость пациентов за медпомощью амбулаторно на 70 процентов, число койко-мест - на 19 процентов, стационарных больниц - на 26 процентов. Таким образом, телемедицина высвобождает огромные ресурсы.

Однако существует ряд проблем. На данный момент эта область находится вне правового поля. Возникает острый вопрос - можно ли врачам оказывать консультативные услуги через интернет или по телефону. Можно предположить, что легализация телемедицины сильно изменит структуру медицинского рынка, так как позволит получать лицензию не на целую компанию, а на частное лицо.Врачи давно могут консультировать и отвечать на вопросы пользователей на форумах и в соцсетях, не неся за это какой-либо ответственности. Лицензирование онлайн-консультаций позволит повысить персональную ответственность врачей и сделать такого рода услуги доступными для широкой аудитории.

## Цель работы

Конечной целью данного проекта является разработка сервиса для организации удаленных консультаций пациентов врачами с помощью видеосвязи. Проект станет инструментом реализации телемедицины.

## Задачи, подлежащие решению для достижения цели

Для достижения поставленной цели были определены следующие задачи:

1. Произвести поиск и анализ существующих аналогичных сервисов
2. Изучить предметную область
3. Спроектировать архитектуру приложения
4. Выбрать технологии для реализации как клиента, так и сервера
5. Выбрать один из методов передачи потока данных между пользователями
6. Написать клиент-серверное приложение с использованием выбранных технологий

## Методология, средства решения задач

## Стартовые данные для выполнения работы

# Обзор существующих аналогов

Для обзора были выбраны 4 сервиса по оказанию удаленных медицинских услуг, воспользоваться которыми может каждый - “Doc+ Онлайн”, “Доктор рядом”, “Онлайн Доктор” и “Яндекс.Здоровье”.

## 2.1 Общие требования и процесс получения консультаций

Для получения консультации необходимо мобильное устройство или персональный компьютер, мобильное приложение или веб-браузер, стабильное соединение с интернет, микрофон и, желательно, но не обязательно видео камера.

Далее необходимо зарегистрироваться в сервисе через мобильное приложение или веб-интерфейс, определиться с врачом и временем консультации забронировав услугу, после чего оплатить.

## 2.2 Яндекс.Здоровье

Компания Яндекс запустила сервис «Яндекс.Здоровье» в 2016 году, которое предоставляет возможность оказывать медицинские услуги от разных врачей с помощью видеоконференций. Имеется возможность получить консультации от терапевта, педиатра, гинеколога и др. врачей, а также получить услугу расшифровку результатов анализов. Плюс ко всему есть консультации ветеринара.

Одной из особенностей сервиса является технология передачи заявку ближайшему необходимому врачу, что сокращает время ожидания консультации. Данная особенность имеет и недостаток в виде получения консультации от ранее незнакомого врача, что может ухудшить качество услуги.

**Преимущества**

1. Низкие цены
2. Отсутствие очередей
3. Множество врачей
4. История приёмов
5. Возможность напоминаний

**Недостатки**

1. Не всегда “свой” врач
2. Отсутствие своего мед.центра

## 2.3 Doc+ Онлайн

Серсис по оказанию медицинских услуг на базе частного медицинского учреждения. Одно из направлений – видео консультации врачей на расстоянии.

**Преимущества**

1. Отсутствие очередей
2. Множество врачей
3. Свой медицинский центр
4. Низкие цены

**Недостатки**

1. Неудобный интерфейс приложения
2. Нет возможности получить консультацию в выходной
3. Видео-консультаций не являются основным направлением

## 2.4 Доктор рядом

Также как и «Doc+ Онлайн» является частной медицинский клиникой и предоставляет услуги удалённых консультаций. Имеет множество врачей различных направлений и специализаций.

**Преимущества**

1. Отсутствие очередей
2. Множество врачей
3. Свой медицинский центр

**Недостатки**

1. Нет возможности получить консультацию в выходной и ночью
2. Отсутствие мобильного приложения

## 2.5 Онлайн Доктор

В отличии от “Doc+ Онлайн” и “Доктор рядом” есть возможность получить консультацию в любое время суток. Есть удобное мобильное приложение, история консультаций, напоминания о предстоящей консультации и др. функционал. Однако, стоимость услуг сильно зависит от опыта и специализации врача и сильно отличается от других аналогичных сервисов.

**Преимущества**

1. Отсутствие очередей
2. Множество врачей
3. Многофункционалное приложение
4. Круглосуточно

**Недостатки**

1. Высокие цены

## 2.6 Итоги сравнения

Подводя итоги можно с уверенностью сказать, что “Яндекс.Здоровье” является серьезным сервисом, который благодаря своим ресурсам может предоставлять качественные (с технической точки зрения) и недорогие услуги. Другие же сервисы предоставляют качественные мед.услуги, однако цены на эти услуги совсем немного ниже очных медицинских консультаций.

Ниже приведена сравнительная таблица цен сервисов. Стоимость варьируется в зависимости от сервиса и часто сильно зависит от квалификации врача и его специализации.

|  |  |  |
| --- | --- | --- |
| Сервис | Стоимость услуг терапевта, рублей | Стоимость услуг специализированного врача, рублей |
| “Doc+ Онлайн” | 499 | 799 |
| “Доктор рядом” | 1200 | 1200 |
| “Онлайн Доктор” | 800-1800\* | 800-2500\* |
| “Яндекс.Здоровье” | Первая консультация - 199, последующие - 499 | |

# Функционал приложения

В ходе выполнения дипломной работы помимо основного функционала, такого как видеосвязь и регистрация, были реализованы сопутствующие удобства, например, поиск врачей, проверка соответствия требованиям для оказания услуг видео-консультаций, чат с возможностью прикрепления материала к консультации и т.д. Ниже приведена диаграмма, описывающие наиболее примечательный функционал приложения.

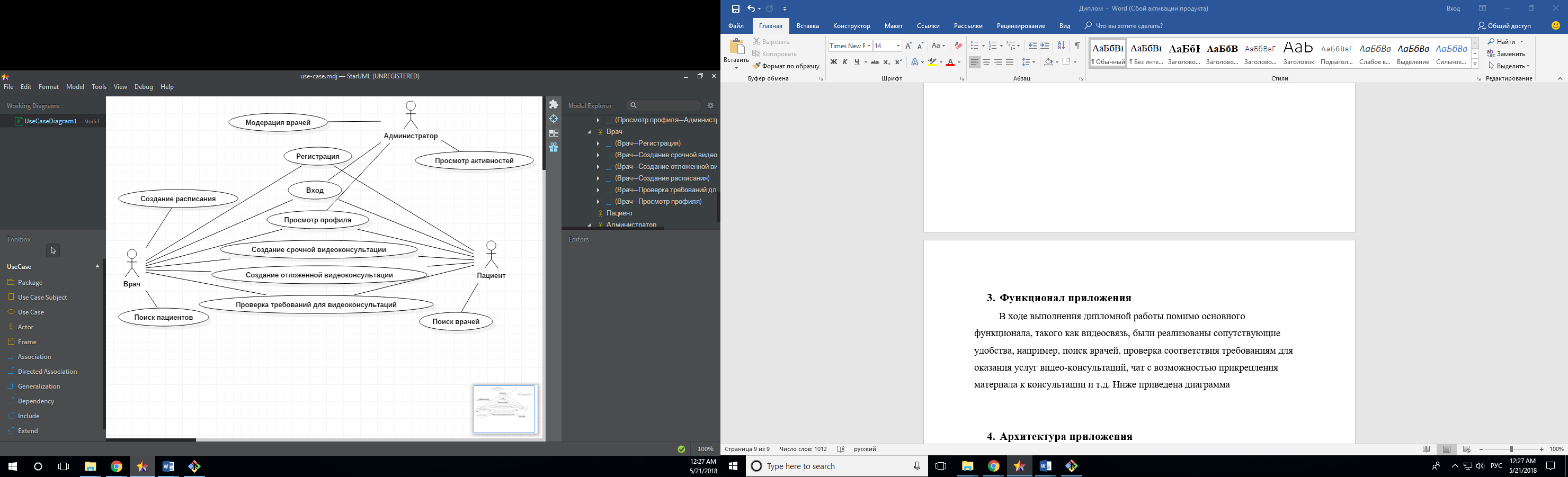


Рисунок 1 - use-case диаграмма

## Аутентификация, авторизация, регистрация

Для любого приложения где имеются пользователи необходимо реализовать как мимимум аутентификацию пользователя, а так как в реализуемом приложении имеются на данный момент три роли (врачи, пациенты, администраторы), то необходима авторизация пользователей. Процесс регистрации будет реализован только для пациентов. Регистрация врачей является несколько более сложным процессом, выходящим за рамки работы, т.к. необходимо определить каким образом врач будет подтвержать свой статус. Предполагается что администраторы и врачи будут внесены в базу данных напрямую.

Для ясности дадим определения терминам аутентфиикация и авторизация.

*Аутентификация* – процедура проверки подленности. Т.е. процедура установления личности на основе предоставленных данных. Например, логина/пароля.

*Авторизация* - процесс провекри наличия прав на совершаемое действие на основе аутентификации.

Ниже приведена диаграмма деятельности процесса авторизации и регистрации пользователей.

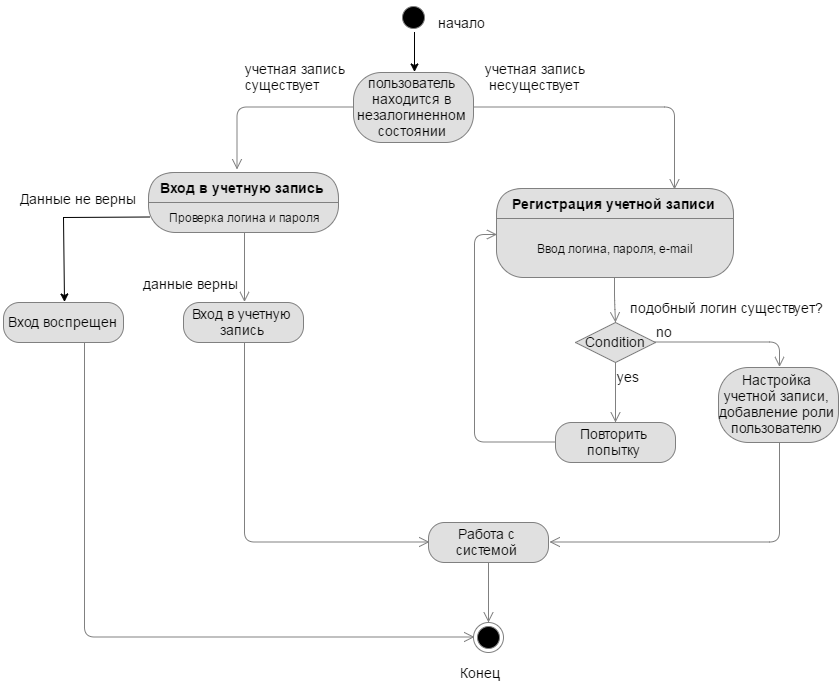


Рисунок 2 - – диаграмма деятельности для авторизации и регистрации пользователей

Так как приложение реализуется по RESTful архитектуре и имеются как клиент так и сервер то, функционал необходимо было его реализовать с обоих сторон.

Процесс регистрации пользователей является стандартным: пользователь отсылает данные для регистрации и в случае успеха сервер ответчает 200 статус-кодом. Данные посылаются на сервер клиентом в теле запроса через HTTP протокол методом POST.

Вводимые пользователем данные валидируются. Валидация данных – процесс проверки корректности вводимых данных. Может быть исполнен как до отправки данных на клиенте, так и в момент обработки данных на сервере.

После успешной регистрации пользователь может воспользоваться своими данными и войти в приложение. Клиент отправляет данные пользователя серверу. Сервер проверяет соответствие логина/пароля пользователя и в случае успеха отсылает клиенту токен доступа – JWT-токен. На основе этого токена клиент в дальнейшем может получать доступ к личным данным пользователя на сервере.

Ниже приведена схема получения JWT-токена и дальнейшее его использование клиентом.

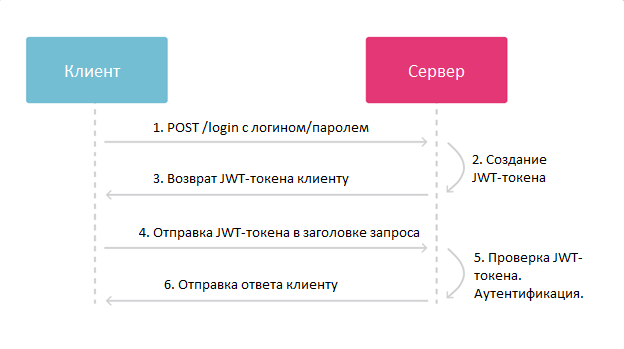


Рисунок 3 - схема получения токена и его использование

JWT-токен – это сгенерированная на сервере последовательность символов, содержащая в себе метаданные о пользователе, например, можно «зашить» в токен роль пользователя, логин, почтовый ящик и др. данные, что и было сделанно. Токен имеет срок жизни, после чего его необходимо обновить. Срок жизни токена также конфигурируется на сервере. Далее получнный токен сохраняется на клиенте и при каждом запросе передается серверу для авторизации.

Для хранения токена на клиенте использовалось стандартное браузерное хранилище – localStorage.

## 3.2 Обзор главных страниц

Зайдя на сайт, неавторизованный пользователь попадает на главную страницу с поиском врача по специализации и/или имени. У пользователя также есть возможность войти в личный кабинет. Практикующий, но не зарегистрированный врач может подать заявление на регистрацию в системе.

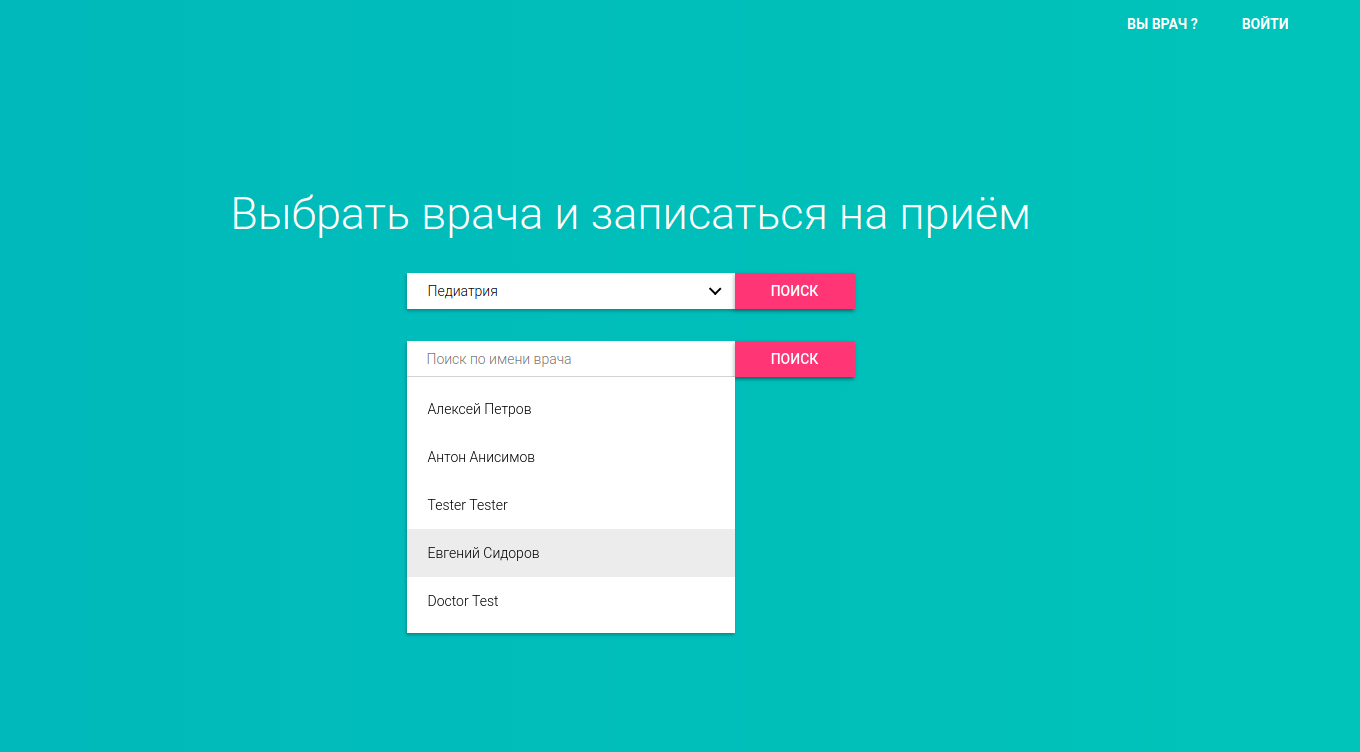
****

Рисунок 4 - главная страница

Интересной особенностью поиска является автозаполнение и поиск без перезагрузки страницы. Достигается это с помощью технологии фоновых запросов *XMLHttpRequest*. При изменении поля ввода или после простоя в одну секунду посылается фоновый запрос на сервер. Поиск происходит по следующим полям:

1. Имени
2. Фамилии
3. Имени и фамилии

Скорость поиска достигается индексированием полей имени и фамилии как вместе, так и по отдельности в базе данных.

После выбора врача происходит переход на страницу с результатом поиска, где можно выбрать время будущей консультации в соответствии с расписанием врача.

Для отображения врачей был сделан список с возможностью фильтрации по статусу и поиску по имени врача. Добавлена постраничная навигация по списку.

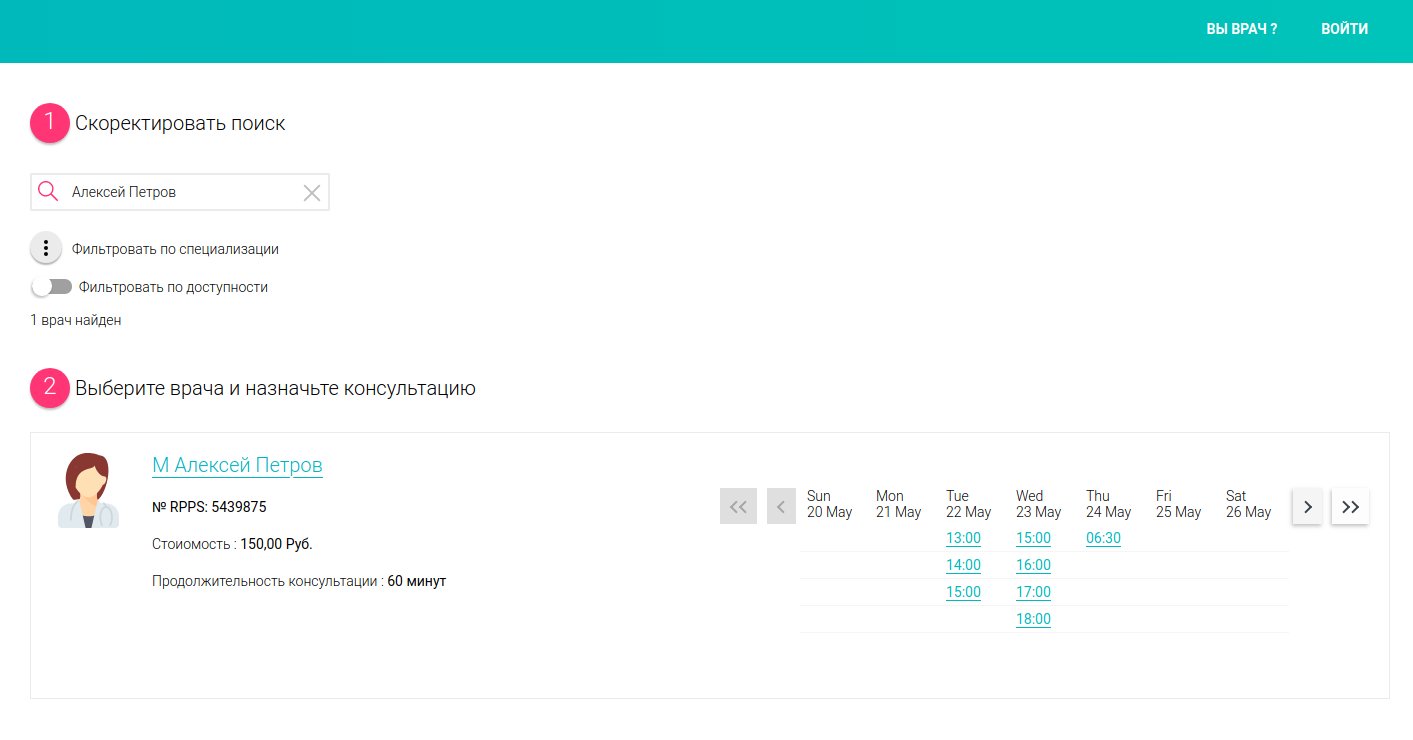


Рисунок 5 – результаты поиска врача

Из интересных технологий можно отметить применение вебсокетов. Вебсокеты были использованы для оторбажения статусов врачей - в сети или нет.

Третьим шагом необходимо указать почтовый адрес на который придут дальнейшие инструкции и напоминания о запланированной консультации пациенту.

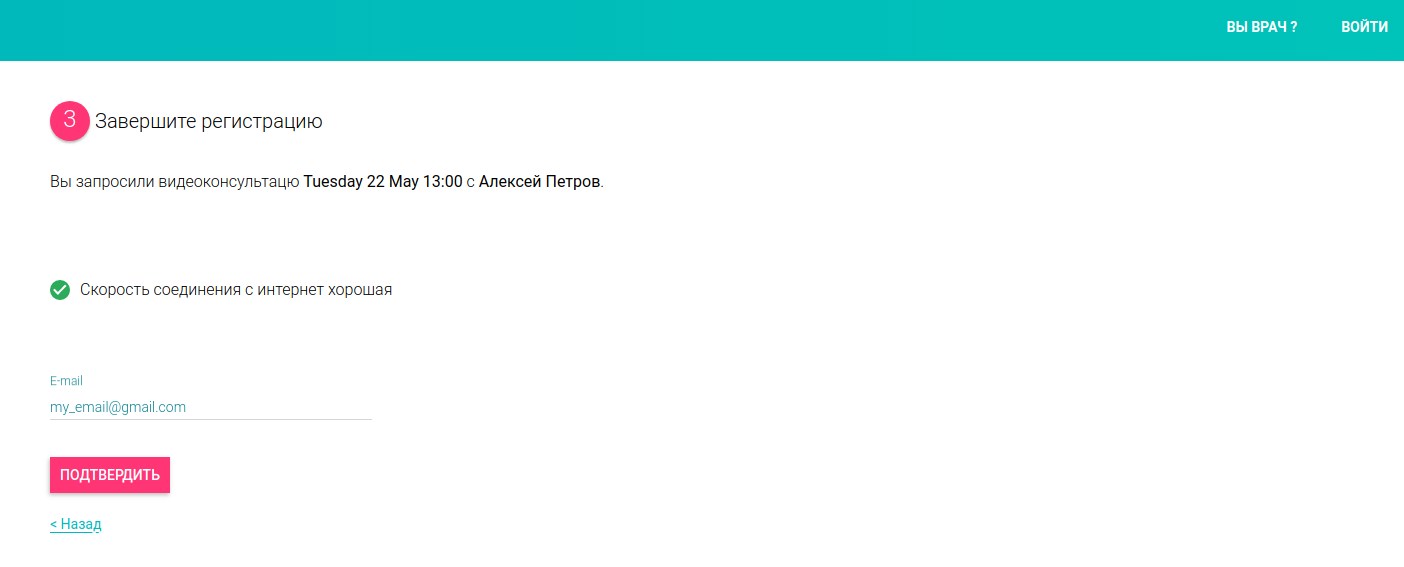


Рисунок 6 – завершение регистрации

## 3.2 Пути создания консультаций

В приложении предусмотрено два пути создания консультаций:

1. запланированные – путь создания был описан выше;
2. моментальные - врач создает консультацию и получает уникальный код, который передает пациенту. Пациент вводит код в форму и попадает в окно консультации с врачом;

Для описания процесса моментальной ниже приведена диаграмма последовательности:

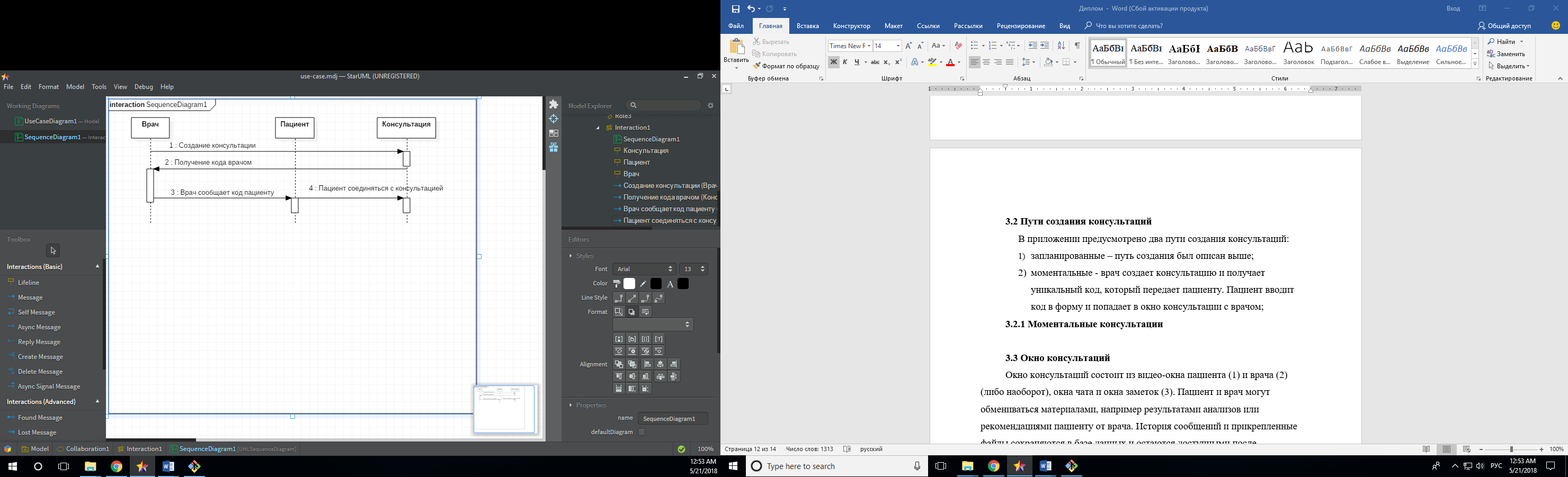


Рисунок 7 – процесс моментальной консультации

## 3.3 Окно консультаций

Окно консультаций состоит из видео-окна пациента (1) и врача (2) (либо наоборот), окна чата и окна заметок (3). Пациент и врач могут обмениваться материалами, например результатами анализов или рекомендациями пациенту от врача. История сообщений и прикрепленные файлы сохраняются в базе данных и остаются доступными после консультации. Имеются кнопки управления: развернуть видео на весь экран, отключить видео. Время консультации фиксируется и отображается на экране.

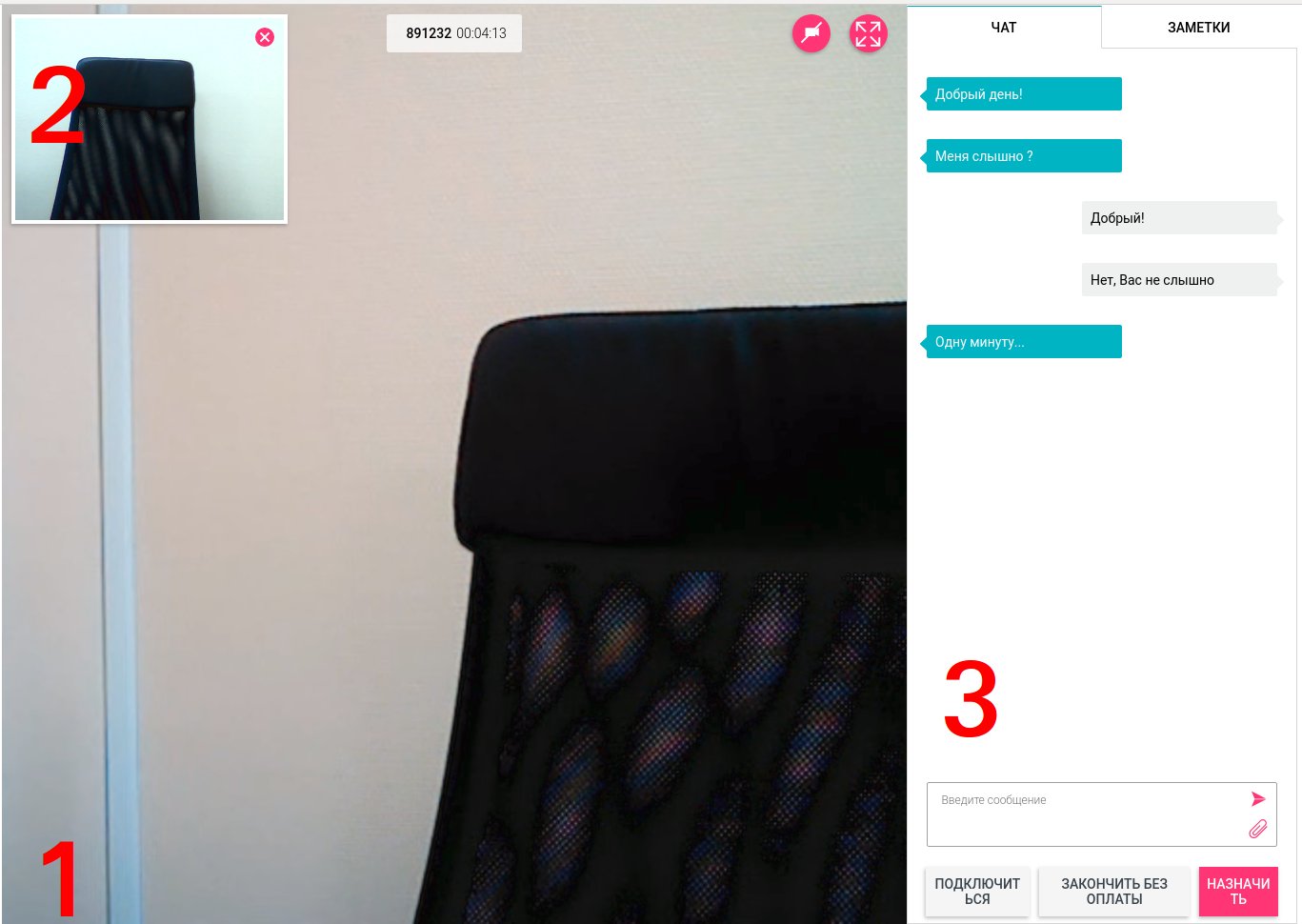


Рисунок 8 - окно видео-консультаций

## 3.4 Тестирование требований

Пользователь приложения (врач, пациент) перед консультацией имеет возможность проверить удовлетворяет ли его браузер, скорость соединения с интернет, веб-камера и микрофон требованиям.

# Архитектура и реализация функционала приложения

Серверная сторона проекта будет реализована на NodeJS с использованием фреймворка Express. В качестве базы данных будет использоваться MongoDB – нереляционная база данных документного типа. Для платформонезависимой разработки выбран Docker.

В качестве языка для разработки клиентского приложения был выбран набирающий популярность скриптовый язык с динамической типизацией javascript. Для построения графических элементов выбрана библиотека от facebook react в связке с менеджером состояний приложения redux и библиотекой создания сторонних эффектов saga. В проекте планируется использовать стандарт ECMAScript 6, который на данный момент поддерживается очень не многими браузерами. Для решения этой проблемы будет использоваться промежуточный траспилятор Babel. Для сборки клиентской части проекта выбран инструмент Webpack.

Клиент-серверное взаимодействие будет происходить в формате JSON поверх HTTP протокола. Для этих целей была реализована RESTful-архитектура.

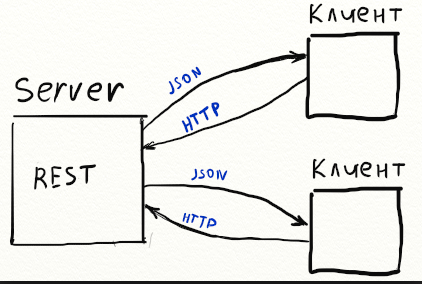
****

Рисунок 9 – RESTFul архитектура

Такая архитектура позволяет отделить бизнес-логику приложения от представления. В рамках проекта планируется реализовать только веб-клиент. Однако, такой подход даёт возможность в дальнейшем без проблем подключить, например, клиенты мобильных приложений.

**Разработка клиентской части**

В качестве языка для разработки клиентского приложения был выбран набирающий популярность скриптовый язык с динамической типизацией javascript. Других аналогов, для создания пользовательских приложений в веб-браузере на данный момент нет. Плюс ко всему, JS настолько универсален, что позволяет создавать в том числе и серверные части приложений на платформе nodeJS, а также,  мобильные приложения путём компиляции написанного кода на нативный язык (Java для android и Objective c для IOS) что обуславливает популярность языка в последнее время. Для построения графических элементов выбрана библиотека от facebook react в связке с менеджером состояний приложения redux.

# Flux-паттерн

Клиентское приложение было построено по паттерну Flux, а точнее его конкретной реализации в виде библиотеки от facebook redux.

# Проблема

Предположим следующую архитектуру приложения (рис…), которая имеет несколько моделей и представлений. Представления в свою очередь могут быть зависимы от нескольких моделей, и модели также могут взаимодействовать между собой и изменять друг-друга. Модели отправляют данные на представления после того как что-то измениться. Появляются некоторые трудности в отладке приложения если что-то пошло не так. Трудности в отслеживании потока данных добавляет еще и то что изменения могут быть асинхронными. Одно изменение может вызвать целый каскад других.

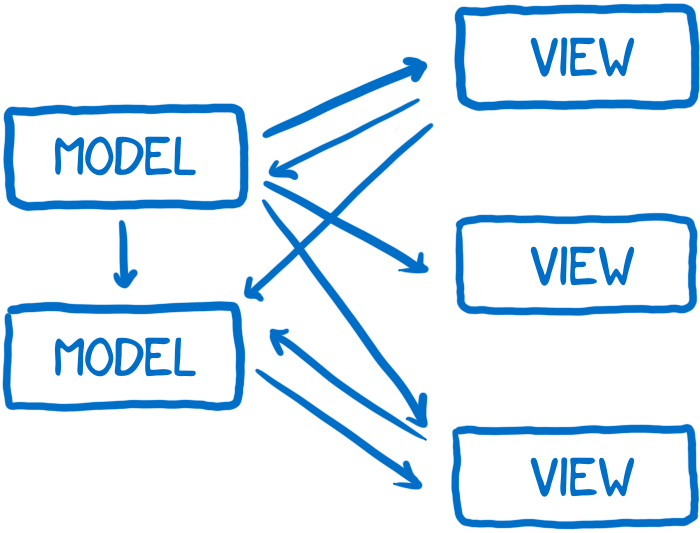
****

Рисунок 10 - путаница во взаимодействии компонентов

Для решения проблемы в компании Facebook придумали и начали использовать другой тип архитектуры - Flux. Во Flux поток данных в приложении является максимально предсказуемым и очевидным. Данные движутся только в одном направлении. На рис (...) представлена схема движения потока данных. В примере рассмотрим конкретную реализацию паттерна Flux - библиотеку Redux. В данной системе фигурируют 4 основных элемента - Action, Reducer, Store, View.

*Action* – далее экшн, отвечает за создание действий. В действии содержится информация о том, что произошло в приложении, а также сам тип действия - т.е. какое действие произошло. Обычно, типы действий в приложении определяются константами в одном месте для дальнейшего переиспользования.

*Reducer* – далее редюсер. Экшн содержит информацию о событии, но при этом не знает что делать. Что делать с этой информацией знает редюсер. Редюсер в соответствии с типом вызванного экшена меняет состояние хранилища, о котором речь пойдет дальше.

*Store* – далее хранилище. Единое хранилище общего состояния приложения. Хранилище выполняет две основные роли - хранит состояние и при изменении состояния информирует об этом компоненты представления (*View*).

*View* - представление приложения. Обычно сгенерированная HTML страница на основе JS. Подписывается на состояние хранилища и при изменении получает новые данные. После получения данных от хранилища обновляет соответствующие элементы представления.

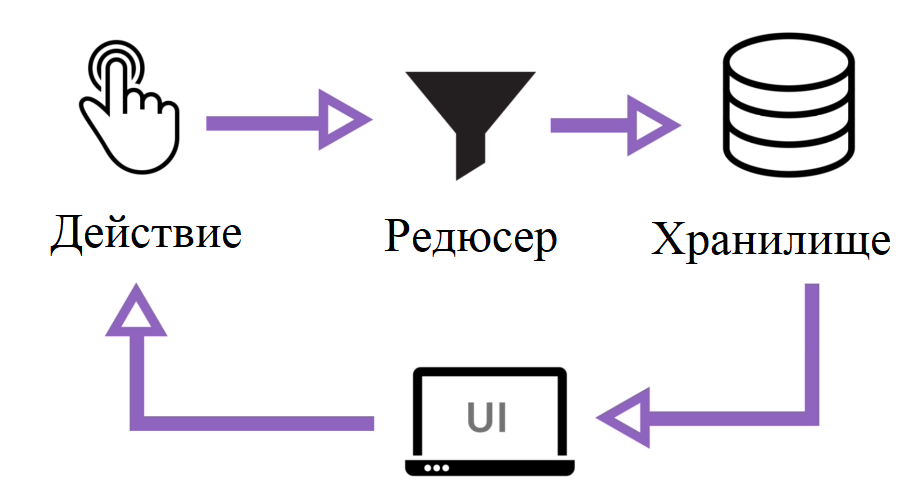


Рисунок 11 – схема взаимодействия компонентов

Другими словами схему можно описать так: пользователь кликает на элемент пользовательского интерфейса (UI) и вызывает действие в приложении. Действие содержит информацию о том что произошло и тип события. Редюсер в соответствии с типом действия изменяет текущее состояние приложения. После изменения состояния, все элементы представления которые были подписаны на хранилище получаются новые данные и в соответствии с ними обновляются.

**Сторонние сервисы**

Twilio

**Технология WebRTC**

**Технология Вебсокетов**

Для списка врачей и отображения их текущих статусов использовались вебсокеты. Вебсокеты позволяют создать соединение в режиме реального времени и даёт возможность обмениваться данными между клиентом и сервером. В отличии от HTTP, вебсокеты могут работать двунаправленно, не разрывая связи между узлами. Ниже приведен пример работы HTTP протокола:

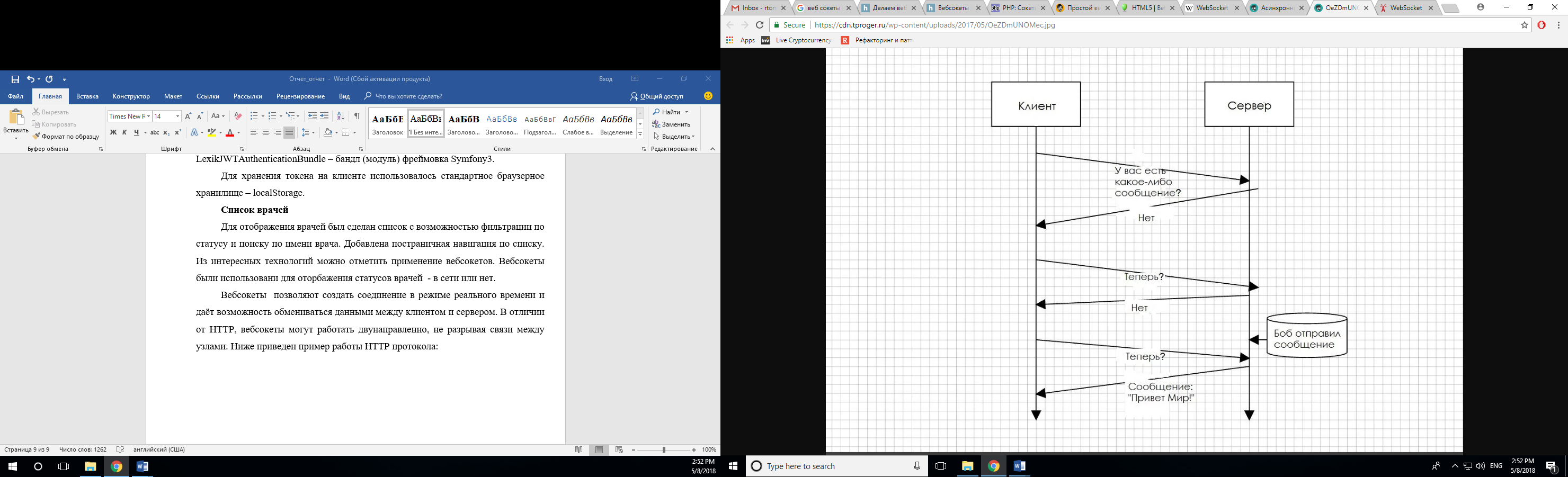


Рисунок 12 – схема работы HTTP-протокола

Как видно из схемы на рис.3 для того, чтобы узнать появилось ли новое сообщение от сервера клиент должен постоянно проверять наличие обновлений. Из этого вытекают две проблемы: либо слишком большая нагрузка на сервер с большим количество обращений от клиента, либо недостаточная интерактивность для клиента.

Вебсокеты, в отличии от HTTP, устанавливают соединение единожды и в момент появления обновлений сервер сигнализирует клиенту об этом и посылает сообщение.

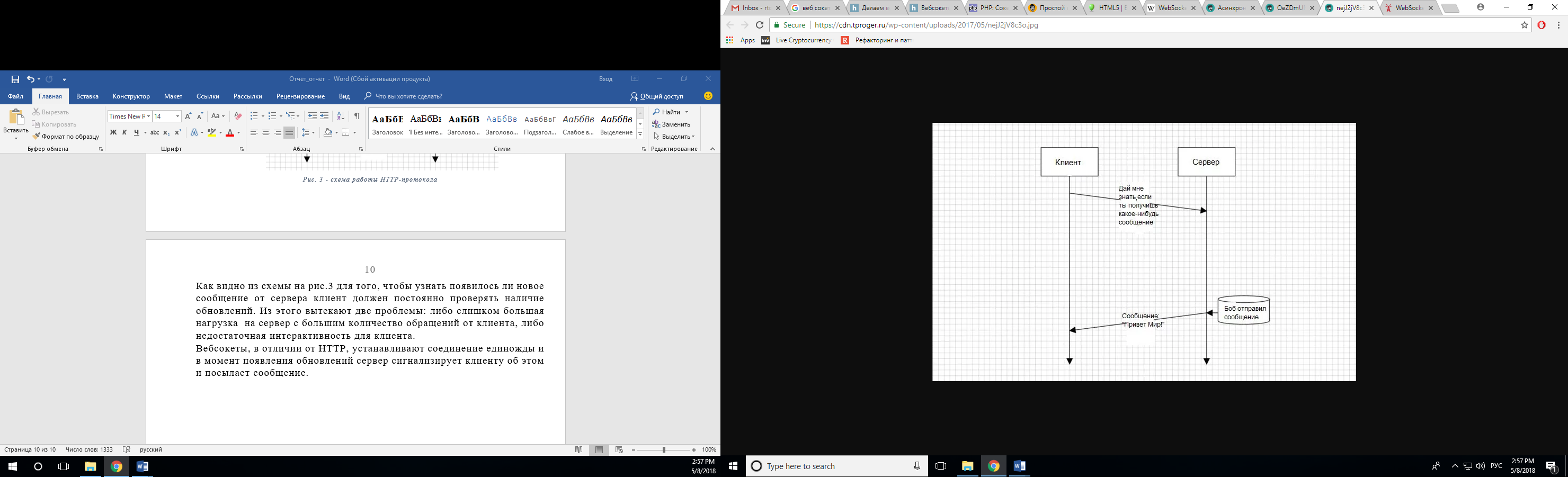


Рисунок 13 – схема работы вебсокета