МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Физико-технический факультет

Кафедра теоретической физики и компьютерных наук

К У Р С О В О Й П Р О Е К Т

АНАЛИЗ СРЕДСТВ РАЗРАБОТКИ ВЫСОКОНАГРУЖЕННЫХ ВЕБ-ПРИЛОЖЕНИЙ

Работу выполнила _____ Снопкова Алла Михайловна

Курс 3

Направление 09.03.02 Информационные системы и технологии

Научный руководитель
канд. биолог. наук, доцент _____ Н. Н. Куликова

Нормоконтролер ст. преподаватель _____ Г. Д. Цой

Краснодар 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность данного курсового проекта заключается в том, что в настоящее время существует огромное количество средств разработки высоконагруженных систем. Поэтому перед разработкой нового приложения необходимо тщательно проанализировать всевозможные средства разработки систем с высокой нагрузкой в соответствии с теми параметрами и функциями, которые будут присутствовать в данном приложении.

Цель проекта — проанализировать все средства разработки веб-приложений исходя из концепции разрабатываемого веб-приложения. Для достижения данной цели необходимо решить следующие задачи:

- освоить теорию о высоконагруженных системах;
- изучить архитектуру и все ее компоненты высоконагруженных систем;
- определить концепцию высоконагруженного веб-приложения;
- рассмотреть всевозможные технологии разработки, используемые в создании систем с высокой нагрузкой;
- сделать анализ средств разработки высоконагруженного веб-приложения с учетом ее функционала;
- на основе результата анализа рассмотренных средств разработать прототип веб-приложения с высокой нагрузкой.

Объект исследования — высоконагруженные системы.

Теоретической основой данного курсового проекта являются статьи и книги по разработке высоконагруженных систем.

1 Общие сведения о высоконагруженных системах

Высоконагруженными (от англ. Highload) называют системы безостановочного доступа, то есть те структуры, запрос данных из которых позволяет получать информацию без длительной задержки при беспрерывной работе [1]. Иными словами, высоконагруженные системы – это, в основном, те же веб-сайты (либо веб-приложения), только с очень большим количеством пользователей и как следствие с большой нагрузкой, требующей оптимизированной серверной части веб-сайта (либо веб-приложения). Качественной характеристикой для высоконагруженной системы является пропускная способность этой системы, она описывает количество работы, которую должна уметь выполнять система в единицу времени.

1.1 Архитектура высоконагруженных систем

Для того чтобы описать архитектуру высоконагруженных систем необходимо для начала дать определение этому термину. Архитектура - это организационная структура системы. Иными словами, это фундамент концепции. И точно так же, как и в строительстве от прочности фундамента зависит качество здания, так в разработке – успешность и жизнеспособность системы.

Исходя из определения, архитектура высоконагруженных систем может быть представлена следующим образом (рисунок 1) [2].

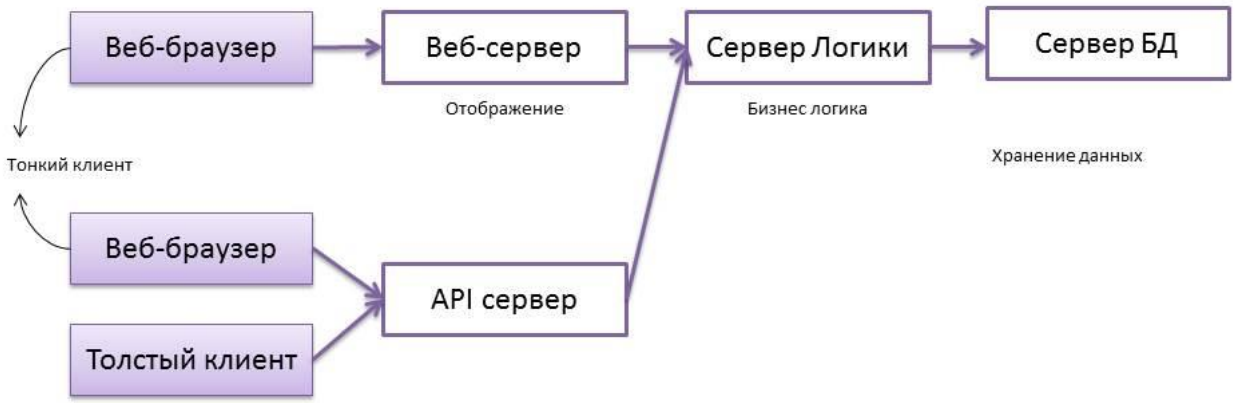

Рисунок 1 – Архитектура высоконагруженных систем

Рассмотрим все компоненты данной архитектуры.

Толстый клиент (англ. rich client) – это приложение, обеспечивающее расширенную функциональность независимо от центрального сервера. Часто сервер в этом случае является лишь хранилищем данных, а вся работа по обработке и представлению этих данных переносится на машину клиента. На рисунке 2 показан принцип работы данного клиента.

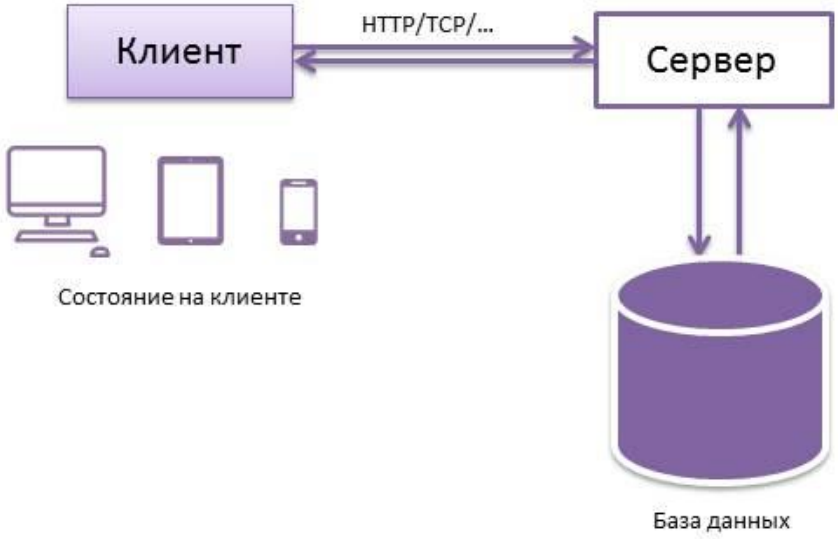

Рисунок 2 – Принцип работы толстого клиента

Тонкий клиент (англ. thin client) – это компьютер или программа-клиент в сетях с клиент-серверной или терминальной архитектурой, который переносит большую часть задач по обработке информации на сервер. Примером тонкого клиента может служить обыкновенный компьютер или мобильное устройство с браузером. Принцип работы данного клиента представлен на рисунке 3.

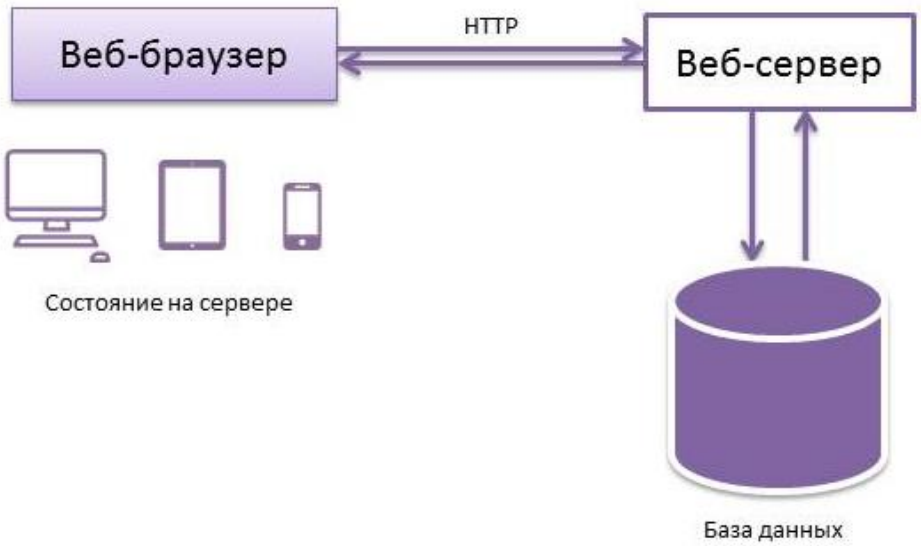

Рисунок 3 – Принцип работы тонкого клиента

Далее выделим преимущества и недостатки тонкого клиента.

Преимущества тонкого клиента:

- обновление;

- скорость разработки;

- не требует совместимость с API;

- свободное проведение экспериментов (проверка работоспособности новой функции или нового модуля системы на конечном пользователе).

Недостатки тонкого клиента:

- большая затрата трафика;

- больше задержки ответа на запрос клиента;

- состояние на сервере;

- отсутствие push – уведомлений (это короткие уведомления, которые посылаются сервером автоматически (не ожидая запроса пользователя) клиенту, появляется обычно вверху экрана планшета или смартфона, либо на панели задач рабочего стола компьютера);

- только online режим.

У толстого, как и у тонкого клиента, также имеются свои достоинства и недостатки.

Достоинства толстого клиента:

- минимальная затрата на трафик;

- меньше задержки ответа на запрос клиента;

- кеширование - размещение данных в специально отведенном месте для ускоренного доступа к ним в будущем;

- постоянное соединение;

- имеются push – уведомления;

- возможность работы в offline режиме;

- сервер без состояния.

Недостатки толстого клиента:

- обновление приложения;

- совместимость API (написание новых интерфейсов и поддержание старых версий интерфейса);

- консистентность состояния - это согласованность состояний друг с другом (необходима для поддержки кеширования), то есть получение обновлений с сервера (частый запрос на получения обновления данных может усложнить протокол, а редкий запрос может и вовсе не донести актуальную информацию конечному пользователю);

- трудность в проведении экспериментов.

Веб-сервер - это программа, которая принимает входящие HTTP-запросы, обрабатывает эти запросы, генерирует HTTP-ответ и отправляет его клиенту. Общий алгоритм работы веб-сервера представлен на рисунке 4 [3].

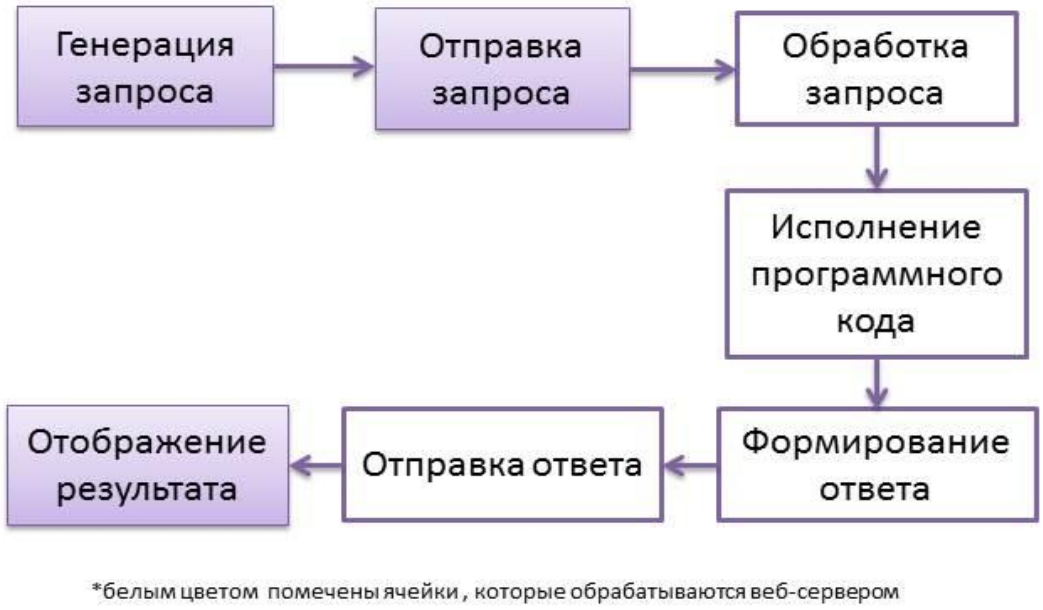

Рисунок 4 – Общий алгоритм работы веб-сервера

Сервер логики (или бизнес – логика) - это реализация правил и ограничений автоматизируемых операций. Другими словами бизнес - логика – это подсистема, которая отвечает за логическую обработку всех действий конечного пользователя.

API сервер (от англ. Application Programming Interface - программный интерфейс приложения) - это определенное представление данных для взаимодействия между приложениями. В данном случае, в качестве ответного приложения выступает сервер.

Сервер базы данных - обслуживает базу данных и отвечает за целостность и сохранность данных, а также обеспечивает операции ввода-вывода при доступе клиента к информации.

Несмотря на то, что в данной схеме описано всего несколько клиентов (два тонких и один толстый), на практике таких клиентов гораздо больше, которые собственно и делают данную систему высоконагруженной, работая одновременно.

Таким образом, highload – это система с высокой нагрузкой, которая возникает связи:

а) с большим количеством одновременных пользователей;

б) с большим объемом обработки данных;

в) с наличием многочисленных сложных расчетов и вычислений.

1.2 Особенности хранения данных в системах с большой нагрузкой

На сегодняшний день достаточно трудно вообразить себе какое-либо приложение, которое не применяло бы базы данных, будь то сервера, персональные компьютеры или мобильные устройства. От простых веб-приложений до серьезных корпоративных систем. Все они обрабатывают, читают и записывают определенный набор данных.

Система управления базами данных (DBMS/СУБД) - программное обеспечение, предназначенное для хранения и управления данными. Для решения различных задач разрабатывалось всё больше и больше различных СУБД (Реляционные и NoSQL) и программ для работы с ними (MySQL, MongoDB, PostgreSQL, Redis и другие).

На данный момент используют один из двух подходов к хранению и работе с данными SQL и NoSQL.

SQL (от англ. Structured Query Language) — язык структурированных запросов, применяемый для создания, модификации и управления информацией в реляционных базах данных, основанных на реляционной модели данных. Понятие «реляционный» был основан на английском языке, как relation («отношение», «зависимость», «связь») [4].

NoSQL (от англ. Not only SQL - не только SQL) — ряд подходов, направленных на реализацию моделей баз данных, имеющих существенные отличия от средств языка SQL, который характерен для традиционных реляционных баз данных [5].

Организация доступа к данным в SQL решается с помощью интерактивной аналитической обработки и реляционного подхода. А в NoSQL решается четырьмя основными способами: с помощью документной ориентации, расширяемых записей (разреженных матриц), ключей доступа и теории графов (рисунок 5).

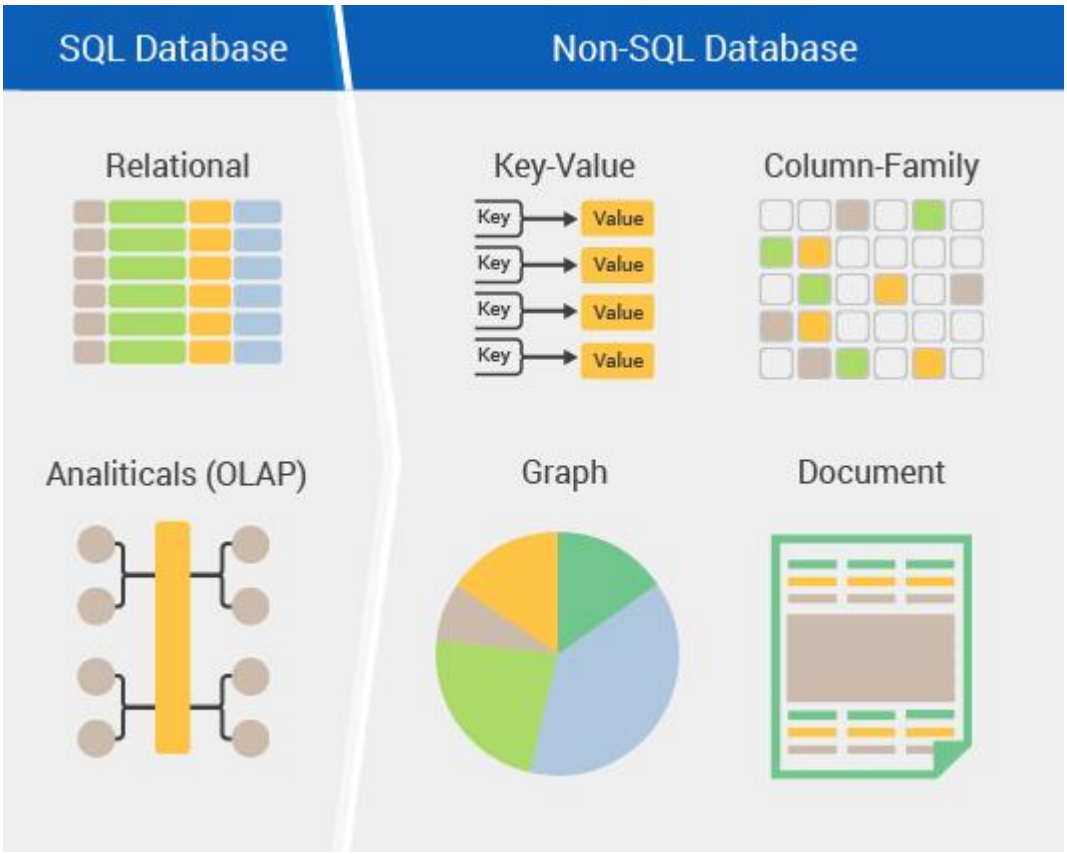

Рисунок 5 – Способы организации доступа к данным NoSQL и SQL

Сравним более детально данные типы СУБД и отметим из особенности:

- структуры данных и их типы - реляционные базы данных используют строгие схемы данных, NoSQL базы данных допускает любой тип данных;

- запросы - вне зависимости от типа лицензии, реляционные базы данных хоть в какой-то мере соответствуют стандартам SQL, поэтому данные из них можно получать при помощи языка SQL, а NoSQL базы данных использует характерные способы запросов к данным;

- масштабируемость - оба эти типа СУБД довольно легко поддаются вертикальному масштабированию (то есть увеличение системных ресурсов), но, во всяком случае, NoSQL является современным продуктом, именно такие СУБД предлагают более простые способы горизонтального масштабирования;

- надежность – когда приходит время до сохранности данных и гарантии выполнения транзакций SQL базы данных по-прежнему занимают лидирующие позиции;

- поддержка - реляционные СУБД имеют огромный опыт, поэтому при возникновении проблем, все же гораздо проще и эффективнее решить их, если дело касается реляционных систем, чем NoSQL, особенно если решение довольно сложное по своей природе;

- хранение и доступ к сложным структурам данных - изначально реляционные системы предполагали работу со сложными структурами, именно поэтому они превосходят остальные решения по производительности.

Вообще, работа в NoSQL требует от программиста как можно больше знаний и опыта, но результаты получаются куда более эффективными. Именно поэтому считается, что SQL уже сейчас начинает постепенно уходить в историю, а NoSQL – будущее всех БД.

Впрочем, данное предсказание разрушается при наличии того факта, что использование реляционного подхода для небольших баз куда продуктивнее. Поэтому необходимо думать о практических вещах, а именно непосредственно о наиболее популярных БД.

В общем, рейтинг 10 наиболее популярных баз данных, согласно ресурсу DB-Engines, выглядит таким образом:

- Oracle Database;
- MySQL;
- Microsoft SQL Server;
- PostgreSQL;
- MongoDB;
- DB2;

– Microsoft Access;
– Redis;
– Elasticsearch;
– Cassandra.

Оценки в рейтинге в данном ресурсе выставляются согласно шести параметрам, которые включают в себя популярность в поисковых системах, социальных сетях и на форумах, частота упоминание в резюме, количество вакансий. Подробные характеристики выше упомянутых популярных баз данных представлены в виде таблицы 1.

Таблица 1 – Основные характеристики баз данных

База данных	Модель базы данных	Язык управления БД
Oracle Database	Реляционная	SQL
MySQL	Реляционная	SQL
Microsoft SQL Server	Реляционная	SQL
PostgreSQL	Реляционная	SQL
MongoDB	Документоориентированная	NoSQL
DB2	Реляционная	SQL
Microsoft Access	Реляционная	SQL
Redis	Ключ-значение	NoSQL
Elasticsearch	Графы	NoSQL
Cassandra	Распределённые значения	NoSQL

Отсюда следует, что 6 из 10 представителей рейтинга – реляционные базы данных, а также по одному экземпляру документоориентированной БД (MongoDB), с распределёнными значениями (Cassandra), использующей подход «ключ-значение» (Redis) и использование графов (Elasticsearch). Следовательно, на данный момент доминируют реляционные базы данных, но в 2016 году, таких представителей реляционных БД было на одного больше.

1.3 Браузерные игры, как системы с большой нагрузкой

На данный момент времени, для того чтобы поиграть в игру, всего лишь достаточно иметь доступ в сеть Интернет. В целом, все игры, где необходим доступ в сеть Интернет, называют online-играми. В зависимости от того, как будет игра запускается, online-игры можно разделить на клиентские и браузерные. В данном параграфе будет идти речь о браузерных играх. Итак, браузерная игра — online-игра, использующая браузерный интерфейс и обычно не требующая установки дополнительных приложений.

Браузерные игры разделяют на следующие типы:

- ролевые online-игры – игры, которые представляют собой моделирование событий, происходящих в данный момент времени в определенном пространстве;

- стратегии – игры, которые базируются на стратегическом планировании и тактику для достижения необходимых целей (делятся на 2 типа: стратегии реального времени и пошаговые стратегии);

- социальные online-игры – многопользовательские игры, которые основаны на взаимодействии реальных игроков в определенный момент времени и на определенной локации;

- квест – игры, представляющий собой приключенческую историю, главный герой которой является сам игрок;

- симуляторы – игры, имитирующие жизнь живых организмов, управление техникой или транспортным средством;

- экшн – игры, в которых победа игрока целиком и полностью зависит от скорости реакции и быстрого принятия решения;

- настольные online-игры – игры, такие как шашки, шахматы, нарды и другие.

Браузерные игры также можно разделить на 3 категории: однопользовательские, многопользовательские и массово-многопользовательские.

Однопользовательская браузерная игра – это online-игра, в которой принимает участие один пользователь. Наиболее распространёнными являются флэш-игры. На данный момент популярность пришлась разработку игр под CANVAS на языке программирования JavaScript. Поэтому, сейчас можно встретить достаточно много однопользовательских игр, написанные на данной платформе.

Многопользовательские браузерные игры – это игры, при котором одновременно играют несколько пользователей [6]. Браузер обеспечивает связь игроков между собой посредством игрового сервера.

Массово-многопользовательские браузерные игры – это игры, в которых сотни, тысячи, или десятки тысяч игроков взаимодействуют друг с другом. Игры, относящиеся к данной подгруппе, «простыми» назвать нельзя. Также как и у многопользовательских игр, браузер обеспечивает связь всех игроков между собой посредством игрового сервера.

Многопользовательские и особенно массово-многопользовательские браузерные игры можно отнести к системам с высокой нагрузкой. Как и в высоконагруженных приложениях, так и в таких категориях игр имеют большое количество игроков, и как следствие на сервер будет производиться огромная нагрузка. Поэтому необходимо максимально оптимизировать серверную часть игры.

Архитектура многопользовательских и массово-многопользовательских игр в точности совпадает с архитектурой высоконагруженных систем, только в качестве пользователя системы выступает только тонкий клиент (веб-браузер).

2 Обзор средств разработки высоконагруженных систем

2.1 Знакомство с API Вконтакте

API (от англ. Application Programming Interface) — это посредник между разработчиком приложений и какой-либо средой, с которой это приложение должно взаимодействовать. API упрощает создание кода, поскольку предоставляет набор готовых классов, функций или структур для работы с имеющимися данными. Далее ознакомимся с основными принципами работы API Вконтакте.

API ВКонтакте — это интерфейс, который позволяет получать информацию из базы данных социальной сети vk.com с помощью http-запросов к специальному серверу. Разработчику не требуется знать в подробностях, как устроена база, из каких таблиц и полей каких типов она состоит — достаточно того, что API-запрос об этом «знает».

Для того чтобы использовать все возможности API ВКонтакте, необходимо зарегистрировать свое приложение.

Откройте страницу «Управление» в левом меню, затем нажмите «Создать приложение» — произойдет переход на страницу https://vk.com/editapp?act=create

Нужно выбрать один из трех ниже перечисленных типов приложений.

Standalone-приложение — это API_ID для мобильного или десктопного клиента, внешнего сайта, где работа с API будет вестись из Javascript. Основная мысль в том, что запросы к API должны осуществляться с устройства пользователя. В интерфейсе приложения с таким типом доступны настройки SDK и подключение сертификатов для push-уведомлений.

Веб-сайт — регистрация API_ID для внешнего сайта и работы с API с сервера. Например, если разработчик хочет написать скрипт на PHP с использованием API ВК, ему нужен именно этот вариант.

IFrame/Flash приложение — это те самые приложения, которые пользователь может видеть в нашем каталоге https://vk.com/apps. Они загружаются непосредственно на сервер ВКонтакте (Flash) или встраиваются во фрейме с внешнего сайта.

После подтверждения действия происходит переход на страницу с информацией о приложении. Если открыть вкладку «Настройки» в меню слева, можно увидеть поле "ID приложения", в котором будет указано число, например, 5490057. Это число — идентификатор приложения, он же API_ID, APP_ID, client_id, оно потребуется разработчику в дальнейшей работе.

Весь список методов, а также подробная документация API Вконтакте представлена на сайте https://vk.com/dev/manuals.

Таким образом, Вконтакте является одной из удобных, популярных и доступных площадок в России, где разработчик может разместить созданную многопользовательскую браузерную игру, посредством API данной социальной сети и заняться его продвижением.

2.2 Технологии программирования для разработки систем с высокой нагрузкой

Разработка высоконагруженного веб-приложения, как правило, делится на 2 типа:

- frontend – разработка публичной части приложения (то есть разработка пользовательского интерфейса, с которым будет взаимодействовать конечный потребитель);

- backend – это разработка и администрирование серверной части веб-приложения (то есть работа с БД, обработка данных, и т.д.).

Рассмотрим методы frontend – разработки многопользовательских браузерных игр. В создании online-игр в основном используют технологии Adobe Flash или HTML 5 (CANVAS) [7].

Adobe Flash (ранее Macromedia Flash, или просто Flash) —

мультимедийная платформа компании Adobe Systems для создания веб-приложений или мультимедийных презентаций (рисунок 6). Широко используется для создания рекламных баннеров, анимации, игр, а также воспроизведения на веб-страницах видео- и аудиозаписей. Flash использует язык программирования ActionScript. Для воспроизведения Flash-приложений в браузере необходимо иметь дополнительно плагин Flash Player. На данный момент Flash-технология доживает свои последние дни, а на смену ей приходит HTML5, так как Flash Player тормозит загрузку сайтов, приводит к чрезмерной нагрузке на центральный процессор, а также быстро разряжает батарею мобильный устройств, поэтому Flash уже не поддерживается на базах Android OS и iOS.

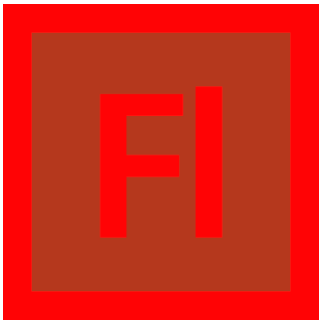

Рисунок 6 – Логотип Adobe Flash

HTML5 не является продолжателем языка разметки гипертекста, а это новая открытая платформа, предназначенная для создания веб-приложений использующих аудио, видео, графику, анимацию и многое другое (рисунок 7). В последние годы HTML5 активно набирает популярность среди разработчиков игр [8]. Причина такого выбора является ее адаптивность на любых устройствах, то есть игры одинаково отображаются на планшетах и телефонах любых моделей, без необходимости подключения дополнительных плагинов. Следовательно, HTML5 теперь считается серьезной игровой платформой. Платформа HTML5 в основном использует язык программирования JavaScript [9].

Рисунок 7 – Логотип HTML5

Для созданий игр существуют движки, которые предоставляют дополнительные возможности: упрощение часто встречающихся задач или загрузка ресурсов, физика, звук, видео, bitmap’ы и так далее. Вообще, большинство движков служат для сокращения временных затрат на разработку полноценной игры, хотя многие разработчики предпочитают создавать свой проект полностью с нуля. Существует несколько HTML5+JS движков, которые действительно чего-то стоят, однако и у них может быть один большой недостаток: они больше не поддерживаются или близки к прекращению поддержки. Поэтому, выбирая движок, необходимо выбрать тот продукт, поддержка которого будет длиться достаточно продолжительное время [10].

Итак, рассмотрим несколько популярных игровых движков для разработки браузерных игр на платформе HTML5.

Игровые движки для создания HTML5 2D игр: PixiJS, Phaser и ImpactJS.

PixiJS - это библиотека, реализующая быстрый и простой механизм визуализации (рисунок 8). Основным характеристическим свойством данного движка является скорость рендера. Авторы утверждают, что на данный момент

это самый быстрый движок рендеринга 2D. В основном используется WebGL рендер, а в старых браузерах автоматически включается рендер в CANVAS. На официальном сайте представлено множество примеров и также документация.

Стоит отметить, что если сравнивать Pixi с игровым движком, предоставляющие разработчику еще больше возможностей по созданию игры (например, Phaser), то здесь необходимо самостоятельно находить и подключать такие библиотеки как физику, твининг и так далее.

Рисунок 8 – Логотип PixiJS

Phaser - игровой движок на платформе HTML5 с открытым исходным кодом (рисунок 9). Для рендера используется PixiJS, поэтому игра может работать либо в CANVAS, либо в WebGL. За скорость, как утверждают создатели, можно не беспокоиться [11].

Множество примеров, уроков и документация присутствуют на официальном сайте. В содержимом файла phaser.js уже имеется пара физических движков, библиотека для твининга, библиотека математики и многое другое.

Рисунок 9 – Логотип Phaser

ImpactJS - это игровой движок на JavaScript на платформе HTML5 (рисунок 10). Данный движок поставляется вместе с фреймворком Ejecta, который помогает публиковать игры на iOS. Ejecta может данные исходники в js использовать под OpenGL. По словам разработчиков, игры ничем не будут отличаться от игр на Objective-C. Данный инструмент не является абсолютно бесплатным, за использование некоторых функций необходимо купить расширенную версию.

Рисунок 10 – Логотип Phaser

Игровые движки для создания HTML5 3D игр: Unity 5, Unreal Engine и Three.js.

Unity 5 - многоплатформенный инструмент для создания трехмерных и двухмерных пользовательских игр (рисунок 11). Является одним из самых популярных трехмерных движков. Так же поддерживает экспорт в HTML5 WebGL. Этот инструмент является бесплатным, но имеются платные лицензии.

Рисунок 11 – Логотип Unity

Unreal Engine - представляет собой набор интегрированных инструментов для разработки трехмерных игр, поддерживает экспорт в HTML5 WebGL начиная с версии 4.7 (рисунок 12). Также как и Unity, движок является бесплатным, но с платными лицензиями.

Рисунок 12 – Логотип Unreal Engine

Three.js - легковесная бесплатная кросс-браузерная библиотека JavaScript, используемая для создания и отображения анимированной компьютерной 3D графики при разработке веб-приложений (рисунок 13). Three.js скрипты могут использоваться совместно с элементом HTML5 (CANVAS), SVG или WebGL.

Рисунок 13 – Логотип Three.js

Таким образом, на данный момент времени HTML5 стал основной игровой платформой для веб-игр. Разработчикам больше не надо полагаться на сторонние плагины. HTML5 работает во всех браузерах, в том числе мобильных, он кроссплатформенный и для него уже создано множество игровых движков.

Далее рассмотрим методы Backend – разработки многопользовательских браузерных игр. Итак, существуют множество языков программирования

backend стороны, но чаще выделяют следующие языки: PHP, Python и Node.js.

PHP (от англ. Hypertext PreProcessor) - язык программирования, используемый на стороне WEB-сервера для динамической генерации HTML-страниц. Является одним из самых популярных инструментов веб-программирования на стороне сервера [12]. Существуют три самых популярных фреймворка, которые строятся на базе данного языка: Laravel, Symphony, Yii.

Python - представляет популярный высокоуровневый язык программирования, который предназначен для создания приложений различных типов. Это и веб-приложения, и игры, и настольные программы, и работа с базами данных. В разработке серверной части веб-приложений используется фреймворк Django.

Node.js – это серверный язык JavaScript. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода.

Таким образом, в Backend’е существует огромное количество средств разработки серверной части браузерных игр. Но стоит подчеркнуть, что необходимо выбирать тот фреймворк, который будет поддерживаться как можно дольше.

3 Анализ средств разработки высоконагруженных веб-приложений и их использование на практике

3.1 Анализ средств разработки и определение концепции разрабатываемого веб-приложения

Разрабатываемое веб-приложение будет иметь формат браузерной игры, потому что:

а) не требует установки программного обеспечения;

б) доступен для всех устройств (то есть не зависит от того насколько мощным является тот или иной компьютер), где установлен браузер;

в) такая игра без обновления «живет» намного дольше, чем клиентские online-игры.

Структура браузерной игры представлена на рисунке 14.

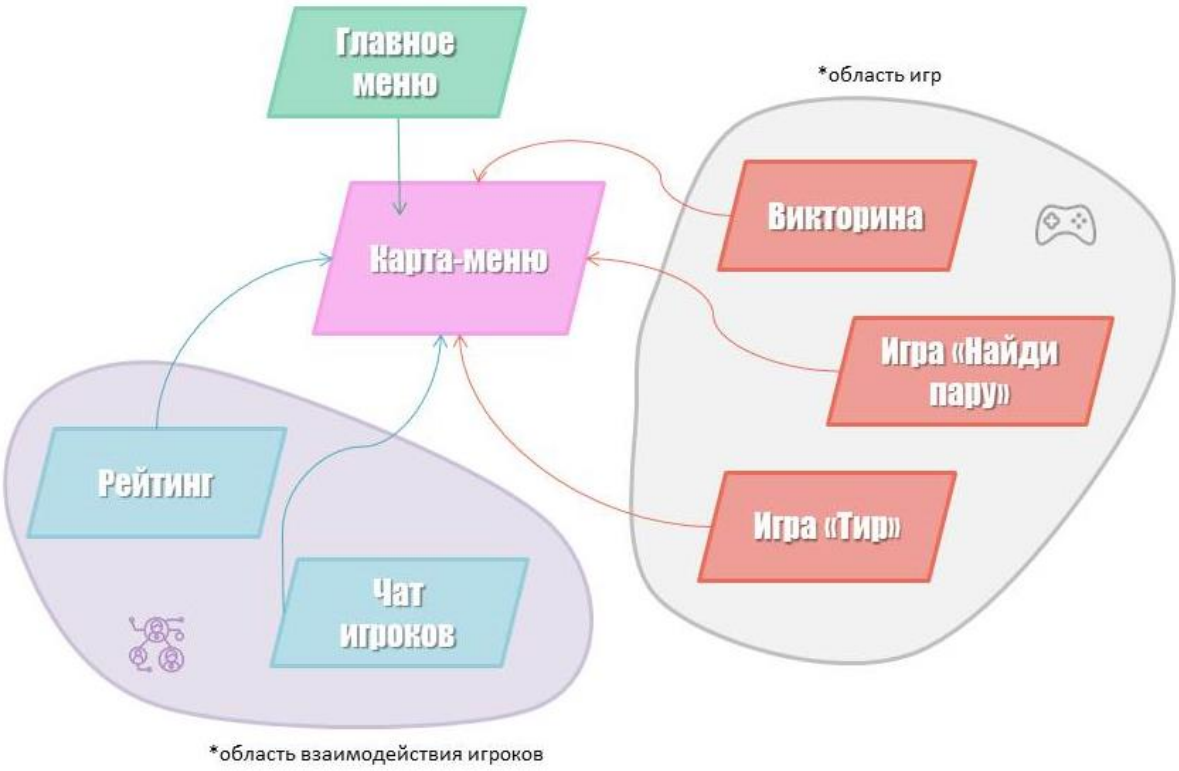

Рисунок 14 – Структура разрабатываемого веб-приложения

Веб-приложение будет состоять из 7 модулей (рис. 14), которые подразделяются на 3 категории:

- главная (главное меню и карта-меню);

- игровая (викторина, игры «Тир» и «Найди пару»);

- социальная (чат игроков и рейтинг).

Помимо данных модулей в приложении также будут присутствовать 2 информационные страницы: «об авторе» и «об игре».

Позже будут разработаны дополнительные игровые модули: «Поймай преступника» и «Поиск предметов».

Для данной игры определена следующая концепция:

- по количеству игроков, приложение будет иметь вид многопользовательской игры;

- мир игры будет построен в двухмерном пространстве (2D);

- тип игры - квест (то есть в рамках данной игры будет разработана мини-история, в которой игрок будет главным героем).

Исходя из предложенной концепции, а также из представленных средств разработки веб-приложений, клиентская сторона игры будет разработана на языке программирования JavaScript, используя технологию платформы HTML5.

На основе того, что интерфейс игры будет представлен в двухмерном пространстве, необходимо использовать игровой движок, предназначенный именно для разработки 2D-игр. Если придерживаться данного замечания, то браузерная игра будет иметь высокую производительность.

Итак, для создания 2D-игры был выбран игровой движок «Phaser» так как в отличие от других библиотек:

а) имеется понятная документация, предоставленная на официальном сайте данной библиотеки;

б) на официальном сайте также присутствует огромное количество примеров и уроков по разработке;

в) имеется чат-форум и техническая поддержка;

г) движок удобен в использовании;

д) данная библиотека может поддерживать мобильные устройства.

Таким образом, клиентской сторона многопользовательской браузерной игры будет разработана на платформе HTML5 на языке программирования JavaScript с использованием библиотеки «Phaser».

Серверная часть браузерной игры будет разработана на языке программирования PHP+MySql. У этого языка программирования есть ряд преимуществ:

- практичность: язык программирования должен предоставить программисту средства для решения поставленной задачи, поэтому благодаря своей богатой функциональности PHP отлично подходит для решения широкого спектра задач;

- простота в изучении: отсутствие строгой типизации, такой как в Java или C++, делает этот язык простым в изучении;

- традиционность: изначально PHP создавался как настройка на Perl, поэтому язык сочетает в себе достоинства Perl и C, следовательно, код PHP очень похож на написанный код на C, что заметно снижает усилия при изучении этого языка;

- эффективность: один из важнейших факторов при выборе языка программирования, поэтому благодаря своему “движку”, сценарии в PHP выполняются с большой скоростью, что позволяет создавать на PHP серьезные веб-приложения;

- гибкость: так как PHP является встраиваемым языком, это дает чрезвычайную гибкость в процессе разработки, поэтому чаще всего сценарии PHP интегрируются в HTML страницы, но при необходимости могут встраиваться и в JavaScript, WML, XML и другие языки;

- безопасность: в PHP реализованы гибкие и эффективные средства безопасности, в том числе ряд надежных механизмов шифрования;

- базы данных: одним из самых больших преимуществ PHP является поддержка большого числа баз данных (более 20 видов).

Подводя итоги, можно с уверенностью сказать, что PHP обладает

большими возможностями. Конечно, у этого языка есть и слабые места, если злоупотреблять его гибкостью и стараться применять его повсеместно, то это приводит к усложнению структуры кода и некорректному выполнению программы. Поэтому необходимо подбирать под свои задачи соответствующие языки программирования, и самое главное, использовать ресурсы с умом.

3.2 Создание прототипа браузерной игры на основе проанализированных средств разработки

Модуль 1: главное меню (рисунок 15). Данный модуль содержит 3 кнопки, которые были созданы с помощью метода `game.add.button(x, y, 'button', click, this, 0, 1, 2)`:

Рисунок 15 – Главное меню

- «играть» - с помощью метода `game.state.start('Locations')` переходит на локацию модуля «карта-меню»;

- «об игре» - с помощью метода `game.state.start('aboutGame')` переходит на информационную страницу, где находится описание разрабатываемой игры;

- «об авторе» - с помощью метода `game.state.start('aboutMe')` переходит на информационную страницу, где находится основная информация и контакты разработчика.

Код модуля представлен в листинге 1.

```
var Menu =
 {
   preload: function (){
    game.load.image('background','./assets/files/menu/background.jpg');
    game.load.spritesheet('button','./assets/files/menu/button.png', 235, 44);
    game.load.bitmapFont('font','./assets/files/menu/font.png','./assets/files/menu/font.fnt');
    game.load.audio('audio','./assets/files/menu/music.mp3');
    },
   create: function (){
     game.add.image(0,0, 'background');
     music = game.add.audio('audio');
     music.loopFull();

     makeButton('ИГРАТЬ', 395, 300);
     makeButton('ОБ ИГРЕ', 395, 370);
     makeButton('ОБ АВТОРЕ', 395, 440);
    }
 };

function makeButton(name, x, y) {
     var button = game.add.button(x, y, 'button', click, this, 0, 1, 2);
     button.name = name;
     button.scale.set(1, 1);
     button.smoothed = true;
     var text = game.add.bitmapText(x, y + 7, 'font', name, 25);
     text.x += (button.width / 2) - (text.textWidth / 2);
};
```

```
function click(button) {
     switch(button.name) {
         case 'ИГРАТЬ':
             game.state.start('Locations');
             music.stop();
             break;
         case 'ОБ ИГРЕ':
             game.state.start('aboutGame');
             music.stop();
             break;
         case 'ОБ АВТОРЕ':
             game.state.start('aboutMe');
             music.stop();
             break;
     }
};
```

Листинг 1 – Главное меню

Модуль 2: карта-меню (рисунок 16). Данный модуль содержит 5 областей выбора игровых зон.

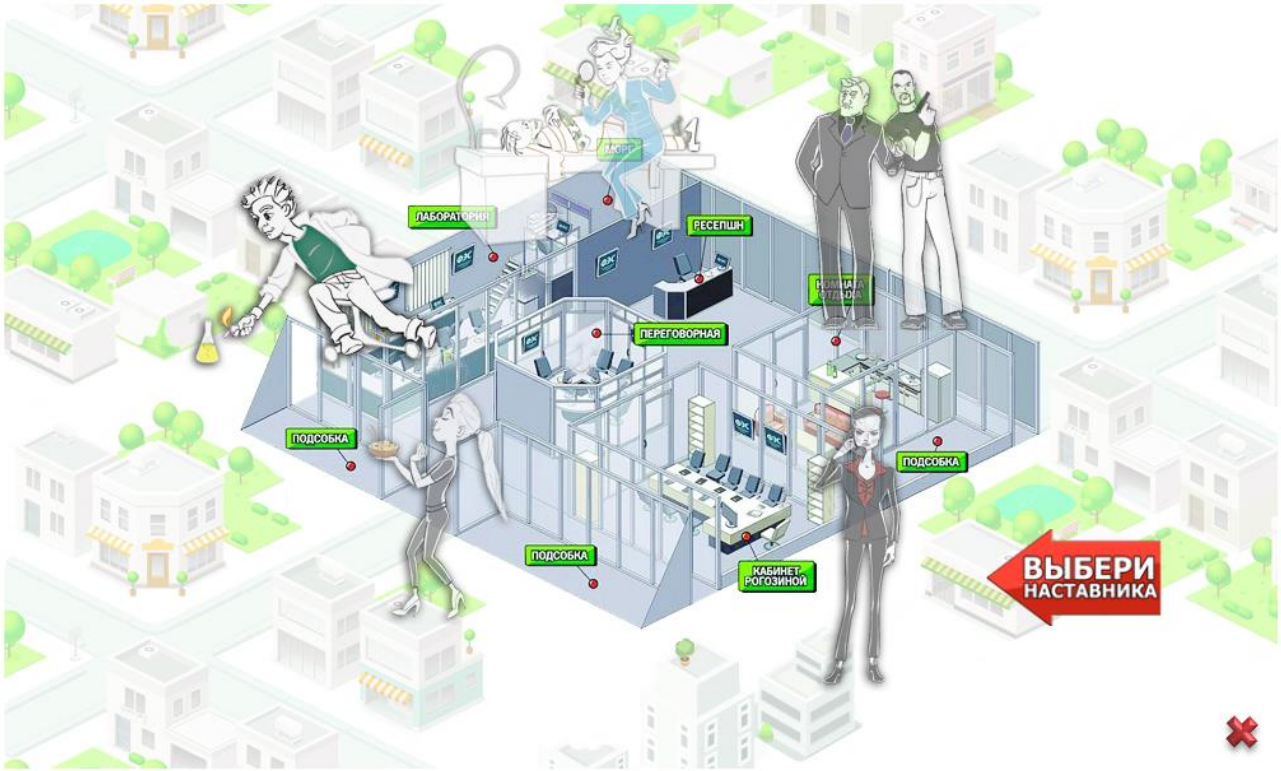

Рисунок 16 – Карта-меню

В листинге 2 представлен отрывок программного кода, при помощи которого разработана каждая область выбора игровой зоны.

```
    playQuiz = game.add.image(280,300, 'rogozina');
    playQuiz.alpha = 0.7;
    playQuiz.inputEnabled = true;
    playQuiz.events.onInputUp.add(function(){
window_game('playrogozina'); });
    function window_game(namegames) {
        game.paused = true;
        namegame = namegames;
        menu = game.add.image(490, 300, namegames);
        menu.anchor.setTo(0.5, 0.5);
        exitLabel = game.add.bitmapText(500, 480, 'font',
'ВЕРНУТЬСЯ В МЕНЮ', 25);
        exitLabel.anchor.setTo(0.5, 0.5);
        playLabel = game.add.bitmapText(480, 365,
'fontplay', 'ИГРАТЬ', 40);
        game.input.onDown.add(unpause, self);
     };
```

Листинг 2 – Разработка выбора области зон

С помощью метода `game.add.image()` загружено изображение в формате *.png. Свойство `playQuiz.alpha` добавляет стилевой эффект для кликабельной области выборки игровой зоны. С помощью функции `playQuiz.events.onInputUp.add(window_game('playrogozina') ;)` создает всплывающее окно при выборе игровой зоны, который дает игроку краткую информацию и согласие на переход в выбранный игровой модуль игры и отказ, который возвращает в меню выбора локации. Данное окно изображено на рисунке 17.

Рисунок 17 – Всплывающее окно меню

Переход в игровой модуль, а также и отказ осуществляется с помощью кода в листинге 3.

```
if (game.paused)
         {
         var x1 = 480, x2 = 645, y1 = 365, y2 = 405;
        if (event.x > x1 && event.x < x2 && event.y > y1
&& event.y < y2){
             switch(namegame) {
                 case 'playbelaya':
                     game.paused = false;
                     game.state.start('Shoot');
                     music.stop();
                     break;
                 case 'playrogozina':
                     game.paused = false;
                     game.state.start('Quiz');
                     music.stop();
                     break;
                 case 'playtikhonov':
                     game.paused = false;
```

```
                    game.state.start('MatchingPairs');
                    music.stop();
                    break;
                }
            }
            else {
                menu.destroy();
                exitLabel.destroy();
                playLabel.destroy();
                game.paused = false;
            }
};
```

Листинг 3 – Разработка выбора области зон

Модуль 3: викторина (рисунок 18). Суть данный игры заключается в том, чтобы игрок ответил на некоторое количество вопросов за короткий промежуток времени, отведенный на каждый вопрос.

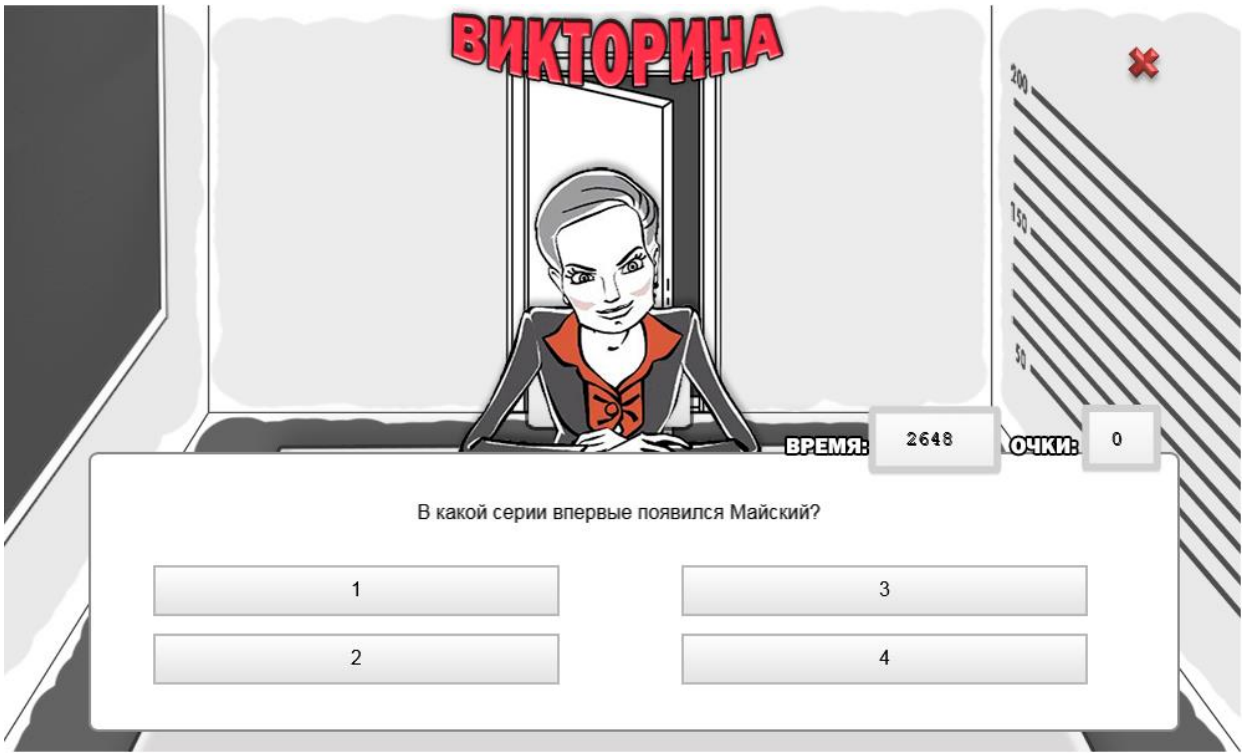

Рисунок 18 – Викторина

Игроку дается 10 вопросов. На данном этапе разработки все вопросы хранятся в текстовом файле. Чтение приложением каждого вопроса происходит последовательно. Структура каждого вопроса в файле следующая:

- вопрос;
- первый вариант ответа;
- второй вариант ответа;
- третий вариант ответа;
- четвертый вариант ответа;
- правильный ответ.

На каждый вопрос у игрока имеется 10 секунд, чтобы выбрать правильный ответ из предложенных вариантов. Среди четырех предложенных на выбор ответов правильным является только один. Подсчет очков производится по количеству вопросов, на которые игрок ответил верно.

Модуль 4: игра «Тир» (рисунок 19). Суть данной игры состоит в том, чтобы игрок научился максимально точно попадать в цель.

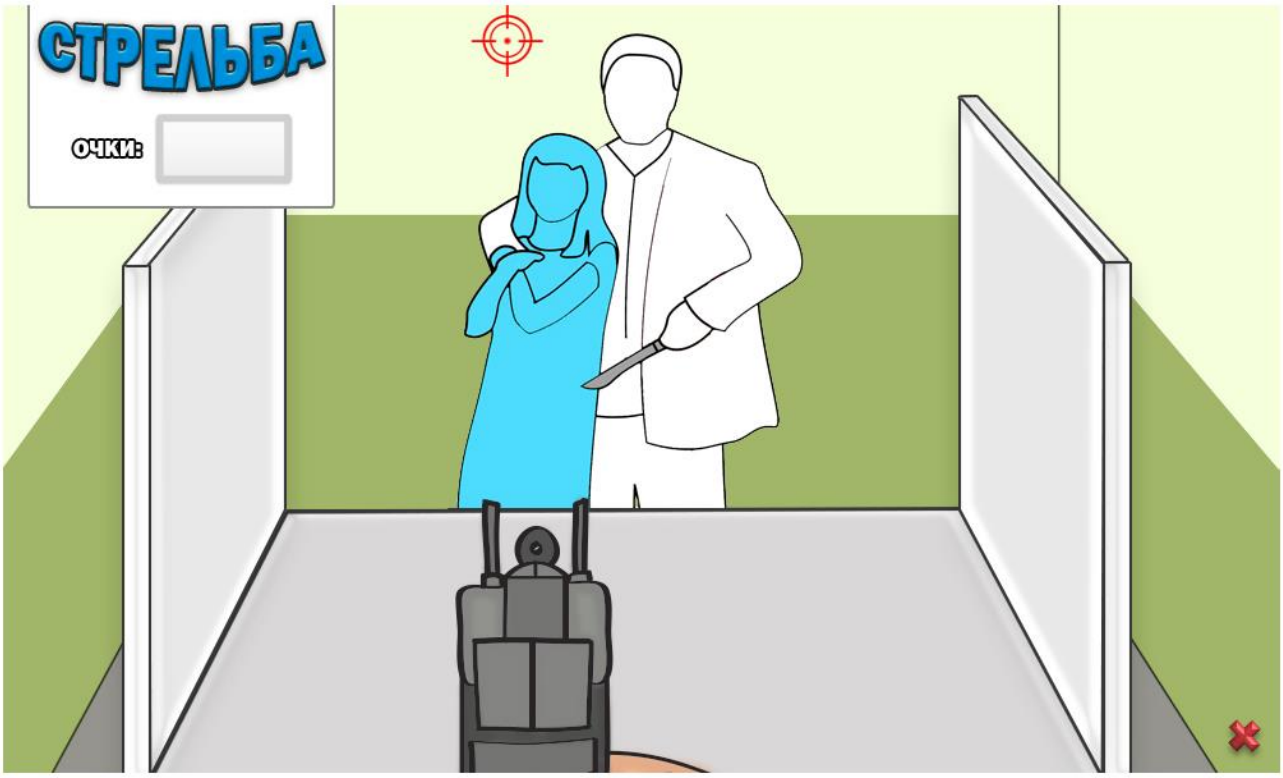

Рисунок 19 – Тир

Игроку дается 3 попытки, чтобы обезвредить преступника и спасти заложника. Прицел передвигается по декартовой системе координат: сначала по оси x, потом по оси y. Переход от горизонтали к вертикали осуществляется путем нажатия клавиши ENTER. Очки начисляются в зависимости от того, в какую область попал игрок. Если игрок попал в заложника, то игра завершается. Определение количества баллов для конкретной области рабочей поверхности осуществляется с помощью технологии JSON.

Модуль 5: игра «Найди пару» (рисунок 20). Суть данной игры заключается в том, чтобы игрок нашел и обозначил пару одинаковых вещей в игровой области модуля за определенный промежуток времени.

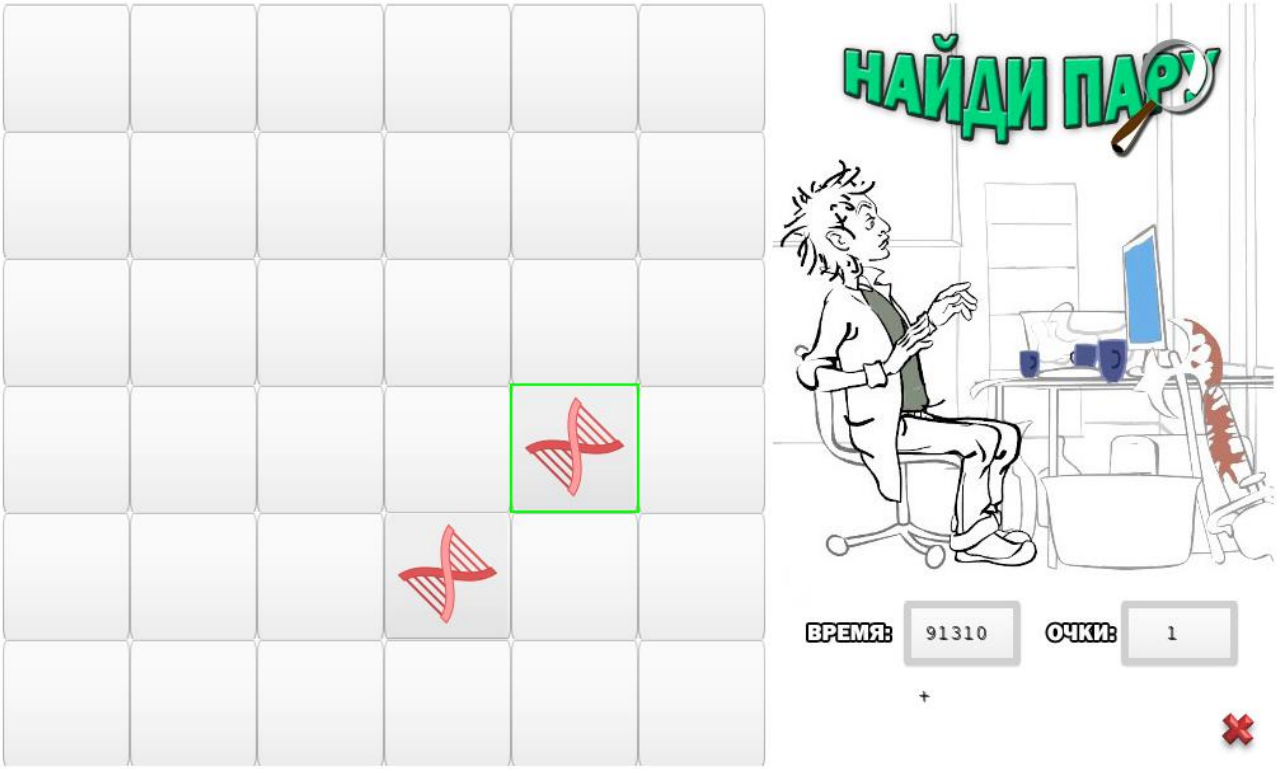

Рисунок 20 – Найди пару

Игроку дается 2 минуты для того, чтобы найти максимальное число одинаковых пар. В общем счете успешность игрока зависит от количества найденных пар и от оставшегося времени. Сетка игровой области, как в

предыдущем модуле, была разработана с помощью технологии JSON.

В рамках данного курсового проекта создана область игр со стороны клиента веб-приложения. На данный момент разработки приложение имеет вид однопользовательской игры. Исходя из концепции приложения для того, чтобы игра была многопользовательской, а соответственно и высоконагруженной системой, необходимо добавить чат и рейтинг (рисунок 21).

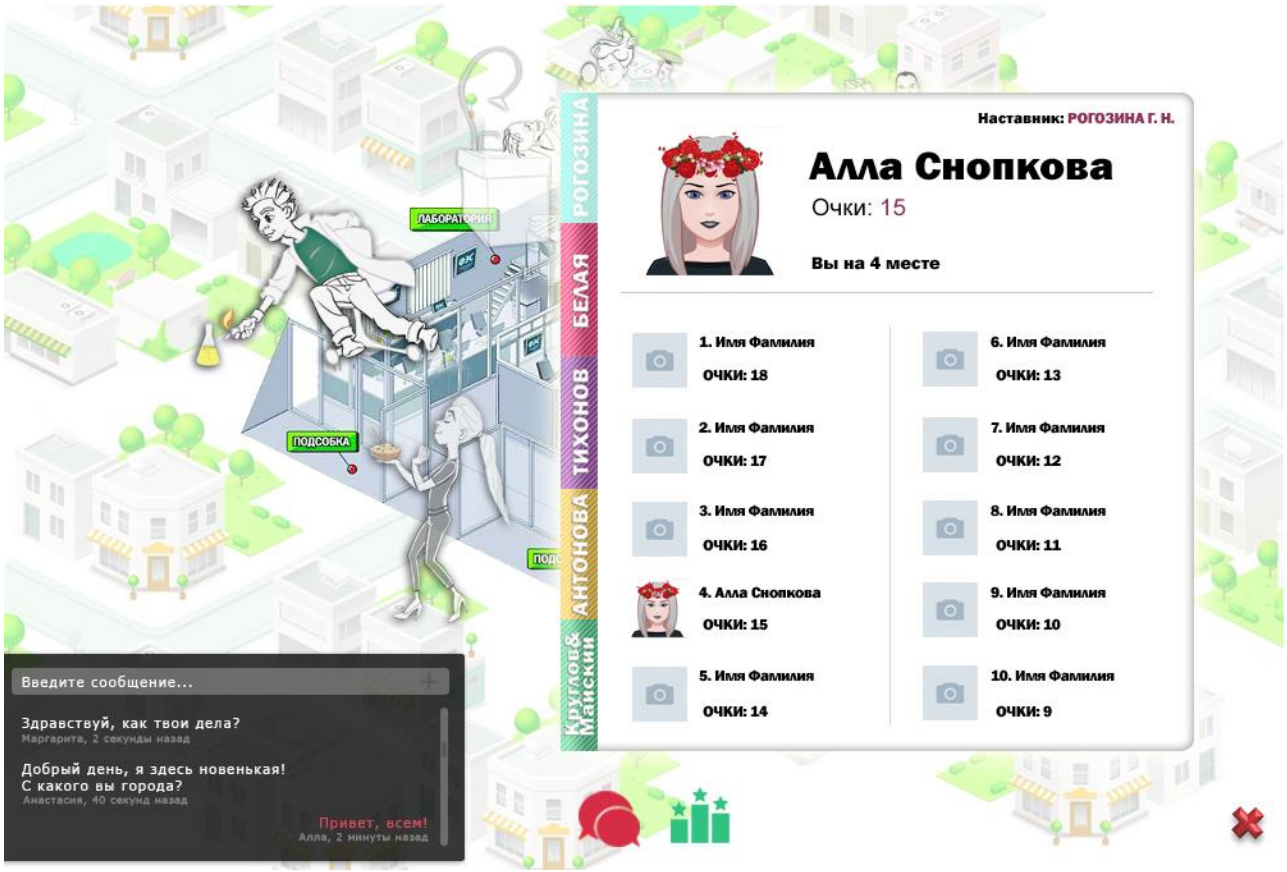

Рисунок 21 – Концепт чата и рейтинга online-игры

Таким образом, в настоящее время была разработана клиентская сторона браузерной многопользовательской игры, а конкретно зона игр. Дальнейшие разработки будут проводиться в зоне взаимодействия игроков и непосредственно серверной части игровой системы. Все пользователи этой игры будут синхронизироваться из социальной сети vk.com с помощью API ВКонтакте.

ЗАКЛЮЧЕНИЕ

Основные результаты данного курсового проекта состоят в следующем:

1 Освоена теория о высоконагруженных системах, используя теоретическую основу данного курсового проекта.

2 Изучена архитектура высоконагруженных систем. Именно здесь более детально рассмотрены все компоненты данной архитектуры и выявлены плюсы и минусы толстого и тонкого клиента.

3 Определена концепция высоконагруженного веб-приложения. Именно здесь подробно проработана схема веб-приложения. Данное приложение будет иметь вид многопользовательской браузерной двумерной квест-игры.

4 Рассмотрены всевозможные технологии разработки, используемые в создании систем с высокой нагрузкой, как и со стороны клиента, так и со стороны сервера.

5 Сделан анализ всех средств разработки высоконагруженных веб-приложений. Здесь подробно были описаны сильные и слабы стороны рейтинговых средств разработки и выявлены самые оптимальные инструменты для создания веб-приложения.

6 На основе результата анализа рассмотренных средств разработан прототип многопользовательской браузерной игры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Бунин О. Разработка высоконагруженных систем / О. Бунин— М.: Издательство Олега Бунина, 2014. – 414 с.

2 Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка / М. Клеппман – СПб.: Питер, 2018. — 640 с.

3 Бунин О. Материалы курса: HighLoad++ [Электронный ресурс] / О. Бунин – (Рус.). – Москва. – URL: http://highload.guide/blog/ [28 марта 2018].

4 Фиайли К. SQL / К. Фиайли – М.: ДМК Пресс, 2014. – 456 с.

5 Redmond E., Wilson R. J. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement / E. Redmond, R. J. Wilson – NC, USA: The Pragmatic Programmers, 2015. – 335 с.

6 Глейзер Д., Мадхав С. Многопользовательские игры. Разработка сетевых приложений / Д. Глейзер, С. Мадхав — СПб.: Питер, 2017. — 368 с.

7 Алексеев А. А., Савельев А. О. HTML5. Основы клиентской разработки / А. А. Алексеев, А. О. Савельев — М.: НОИ Интуит, 2016. — 271с.

8 Роббинс Д. HTML5. Карманный справочник / Д. Роббинс — М.: Вильямс, 2015. — 192 с.

9 Фримен Э., Робсон Э. Изучаем программирование на JavaScript / Э. Фримен, Э. Робсон — СПб.: Питер, 2015. — 640 с.

10 Беляев С. А. Разработка игр на языке JavaScript / С. А. Беляев — СПб.: Лань, 2016. – 128 с.

11 Davey R. A Guide to the Phaser Tween Manager / R. Davey — Victoria, Canada: Leanpub, 2015. – 150 с.

12 Колисниченко Д. PHP и MySQL: разработка Web-приложений / Д. Колисниченко – СПб.: БХВ-Петербург, 2015. — 593 с.