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 технологий

Допустить к защите
Заведующий кафедрой
д-р физ.-мат. наук, профессор
_____ В.А. Исаев
_____2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

СОЗДАНИЕ WEB-ПРИЛОЖЕНИЯ НА ОСНОВЕ АРХИТЕКТУРЫ ВЫСОКИХ НАГРУЗОК

Работу выполнил ___Б.В. Меньшаков

Направление подготовки 09.03.02 Информационные системы и технологии

Направленность (профиль) Информационные системы и технологии

Научный руководитель
канд. физ.-мат. наук, доцент _____О.М. Жаркова

Нормоконтролер
ст. преподаватель _____ Г.Д. Цой

Краснодар
2018

СОДЕРЖАНИЕ

ОБОЗНАЧНИЯ И СОКРАЩЕНИЯ

MVC	Model View Controller
ORM	Object Role Model
REST	Representational State Transfer
API	Application Programming Interface
HTTP	HyperText Transfer Protocol
PHP	Personal Home Page Tools
AR	Active Record
VPS	Virtual Private Server

ВВЕДЕНИЕ

Web-технологии стремительно развиваются, и чтобы успевать за ско-ростью развития новых технологий, у веб-программистов часто недостаточно времени для реализации повседневных задач с нуля, таких как: аутентификация, API, тестирование, дебаг, кэширование, работа с БД и т.д. Именно для упрощения и быстрого решения этих задач были придуманы веб-фреймворки. Веб-фреймворк – это каркас приложения, набор инструментов, стандартов и, в какой-то степени, рабочее пространство.

Помимо этого, еще одной острой проблемой последних лет становится оптимизация сайтов под большое количество запросов. Пользователей сети интернет становится все больше, и все больше сайтам требуется оптимизация под высокие нагрузки.

Наступает этот момент тогда, когда текущая инфраструктура сайта начинает показывать первые признаки того, что она перестает справляться с нагрузкой. При использовании VPS на 128 Мб это может быть 10 запросов в секунду. Для кого-то это может быть 10 тысяч запросов. Основная задача разработчика заключается в том, чтобы определить существует ли необходимость для масштабирования и оптимизации инфраструктуры.

Целью данной работы является создание web-приложения заранее предполагающее работу с высокой нагрузкой и показ процесса его создания. Для достижения этой цели были сформулированы следующие задачи:

- выбор технологий и средств разработки;
- разбор средств для борьбы с высокими нагрузками;
- разработка структуры приложения;
- выявление и постановка задач для решения;
- применение оптимизации и масштабирования.

1 Теоретические аспекты изучаемой работы

1.1 Используемые технологии

Для создания приложения использовались следующие технологии: язык программирования PHP, язык запросов SQL и его разновидность MySQL, а также язык программирования JavaScript и сервер nginx. Остановимся подробнее на языке программирования PHP.

На сегодняшний день PHP является наиболее распространенным языком веб-программирования. Подавляющие большинство сайтов и веб-сервисов в интернете написано с помощью PHP. По некоторым оценкам, PHP применяется более чем на 80% сайтов, среди которых такие сервисы, как facebook.com, vk.com, baidu.com и другие. Популярность PHP обусловлена простотой языка, который позволяет быстро и легко создавать сайты и порталы различной сложности.

PHP был создан в 1994 году датским программистом Расмусом Лердорфом и изначально представлял собой набор скриптов на другом языке Perl. Позже этот набор скриптов был переписан в интерпретатор на языке Си. И с самого возникновения PHP (сокращение от PHP: Hypertext Preprocessor - PHP: Препроцессор гипертекста) представлял удобный набор инструментов для упрощенного создания веб-сайтов и веб-приложений.

Преимущества PHP:

- для всех наиболее распространенных операционных систем (Windows, MacOS, Linux) есть свои версии пакетов разработки на PHP, а это значит, что можно создавать веб-сайты на любой из этих операционных систем;

- PHP может работать в связке с различными веб-серверами: Apache, Nginx, IIS - простота и легкость освоения. Как правило, уже имея небольшой опыт в программировании на PHP, можно создавать простейшие веб-сайты;

- PHP имеет схожий синтаксис с языком Си, поэтому, зная Си или один из языков с сиподобным синтаксисом, будет проще овладеть PHP;

- PHP поддерживает работу с множеством систем баз данных (MySQL, MSSQL, Oracle, Postgre, MongoDB и другие);

- распространенность хостинговых услуг и их дешевизна;

- постоянное развитие. PHP продолжает развиваться, появляются все новые версии, которые несут новые функции, адаптируя язык программирования к новым вызовам. И, как правило, перейти на новую версию не составляет сложностей.

Теперь рассмотрим разновидность языка запросов SQL – MySQL. MySQL представляет систему управления реляционными базами данных (СУБД). На сегодняшний день это одна из самых популярных систем управления базами данных.

Изначальным разработчиком данной СУБД была шведская компания MySQL AB. В 1995 году она выпустила первый релиз MySQL. В 2008 году компания MySQL AB была куплена компанией Sun Microsystems, а в 2010 году уже компания Oracle поглотила Sun и тем самым приобрела права на торговую марку MySQL. Поэтому MySQL на сегодняшний день развивается под эгидой Oracle.

MySQL обладает кроссплатформенностью, имеются дистрибутивы под самые различные ОС, в том числе наиболее популярные версии Linux, Windows, MacOS.

Остановим подробнее на языке программирования JavaScript. Сегодняшний мир веб-сайтов трудно представить без языка JavaScript. JavaScript – это то, что делает живыми веб-страницы, которые человек каждый день просматривает в своем веб-браузере.

JavaScript был создан в 1995 году в компании Netscape в качестве языка сценариев в браузере Netscape Navigator. Первоначально язык назывался LiveScript, но на волне популярности в тот момент другого языка Java LiveScript был переименован в JavaScript [1]. Однако данный момент до сих пор иногда приводит к путанице: некоторые начинающие разработчики считают, что Java и JavaScript чуть ли не один и тот же язык. Это два абсолютно разных языка, и они связаны только по названию.

Первоначально JavaScript обладал довольно небольшими возможностями. Его цель состояла лишь в том, чтобы добавить немного поведения на веб-страницу. Например, обработать нажатие кнопок на веб-странице, произвести какие-нибудь другие действия, связанные прежде всего с элементами управления.

Однако развитие веб-среды, появление HTML5 и технологии Node.js открыло перед JavaScript гораздо большие горизонты. Сейчас JavaScript продолжает использоваться для создания веб-сайтов, только теперь он предоставляет гораздо больше возможностей.

Также он применяется как язык серверной стороны. То есть если раньше JavaScript применялся только на веб-странице, а на стороне сервера нам надо было использовать такие технологии, как PHP, ASP.NET, Ruby, Java, то сейчас благодаря Node.js мы можем обрабатывать все запросы к серверу также с помощью JavaScript.

В последнее время переживает бум сфера мобильной разработки. И JavaScript опять же не остается в стороне: увеличение мощности устройств и повсеместное распространение стандарта HTML5 привело к тому, что для создания приложений для смартфонов и планшетов также можно использовать JavaScript.

Более того, благодаря выходу нового семейства операционных систем Windows (Windows 8 / 8.1 / 10) можно использовать данный язык программирования для разработки приложений для этих операционных систем. То есть JavaScript уже перешагнул границы веб-браузера, которые ему были очерчены при его создании.

Теперь Javascript может использоваться для набирающего популярность направления разработки для IoT(Internet of Things или Интернет вещей). А значит, JavaScript можно использовать для программирования самых различных "умных" устройств, которые взаимодействуют с интернетом.

Таким образом, можно встретить применение JavaScript практически повсюду. Сегодня это действительно один из самых популярных языков.

С самого начала существовало несколько веб-браузеров (Netscape, Internet Explorer), которые предоставляли различные реализации языка. И чтобы свести

различные реализации к общему стержню и стандартизировать язык под руководством организации ECMA, был разработан стандарт ECMAScript. В принципе, сами термины JavaScript и ECMAScript являются во многом взаимозаменяемыми и относятся к одному и тому же языку.

К настоящему времени ECMA было разработано несколько стандартов языка, которые отражают его развитие.

JavaScript является интерпретируемым языком. Это значит, что код на языке JavaScript выполняется с помощью интерпретатора. Интерпретатор получает инструкции языка JavaScript, которые определены на веб-странице, выполняет их (или интерпретирует) [2].

Рассмотрим немаловажную часть любого приложения – web-сервер nginx. NGINX – программное обеспечение, написанное для UNIX-систем. Основное назначение – самостоятельный HTTP-сервер, или, как его используют чаще, фронтенд для высоконагруженных проектов. Возможно использование NGINX как почтового SMTP/IMAP/POP3-сервера, а также обратного TCP прокси-сервера.

NGINX является широко используемым продуктом в мире IT, по популярности уступая лишь Apache. Как правило, его применяют либо как самостоятельный HTTP-сервер, используя в бэкенде PHP-FPM, либо в связке с Apache, где NGINX используется во фронтенде как кеширующий сервер, принимая на себя основную нагрузку, отдавая статику из кэша, обрабатывая и отфильтровывая входящие запросы от клиента и отправляя их дальше к Apache [3]. Apache работает в бекэнде, работая уже с динамической составляющей проекта, собирая страницу для передачи её в кеш NGINX и запрашивающему её клиенту.

1.2 Выбор фреймворка для разработки приложения

1.2.1 PHP фреймворки

Фреймворк – это каркас, предназначенный для создания динамических веб-сайтов, сетевых приложений, сервисов или ресурсов. Он упрощает разработку и избавляет от необходимости написания рутинного кода. Многие каркасы упрощают доступ к базам данных, разработку интерфейса и также уменьшают дублирование кода.

Рассмотрим один из популярных фреймворков Yii 2. Yii – это универсальный фреймворк, и он может быть задействован во всех типах веб-приложений. Благодаря его компонентной структуре и отличной поддержке кэширования, фреймворк особенно подходит для разработки таких крупных проектов, как порталы, форумы, CMS, магазины или RESTful-приложения. Yii – это не проект одного человека. Он поддерживается и развивается сильной командой и большим сообществом разработчиков, которые ей помогают. Авторы фреймворка следят за тенденциями веб-разработки и развитием других проектов. Наиболее подходящие возможности и лучшие практики регулярно внедряются в фреймворк в виде простых и элегантных интерфейсов:

- как и многие другие PHP фреймворки, для организации кода Yii использует архитектурный паттерн MVC;

- Yii придерживается философии простого и элегантного кода не пытаясь усложнять дизайн только ради следования каким-либо шаблонам проектирования;

- Yii является full-stack фреймворком и включает в себя проверенные и хорошо зарекомендовавшие себя возможности: ActiveRecord для реляционных и NoSQL баз данных, поддержку REST API, многоуровневое кэширование и другие;

- Yii отлично расширяем. Можно настроить или заменить практически любую часть основного кода. Используя архитектуру расширений, легко делиться кодом или использовать код сообщества;

- одна из главных целей Yii – производительность [4].

Рассмотрим не менее популярный фреймворк Laravel 5. Laravel – это фреймворк для веб-приложений с выразительным и элегантным синтаксисом. Он позволит упростить решение основных наболевших задач, таких как аутентификация, маршрутизация, сессии и кэширование. Laravel – это попытка объединить всё самое лучшее, что есть в других PHP фреймворках, а также Ruby on Rails, ASP.NET MVC и Sinatra.

Laravel – доступный, но мощный. Располагает множеством отличных инструментов для крупных, надёжных приложений:

- как и многие другие PHP фреймворки, для организации кода Laravel использует архитектурный паттерн MVC;

- Laravel придерживается философии выразительного, красивого синтаксиса. Он предназначен для людей, которые ценят элегантность, простоту и читаемость;

- функционал Laravel является простым для понимания и использования. Если вам нравится простота и легкость CodeIgniter, вы получите удовольствие и от Laravel;

- большинство функций Laravel превосходно работают, не требуя дополнительной настройки. Они опираются на общепринятые стандарты написания кода, делая его интуитивно понятным.

- документация Laravel закончена и постоянно обновляется;

- превосходная IoC (Инверсия управления);

- удобная система миграций;

- интегрированная система модульного тестирования [5].

1.2.2 Сравнение общих параметров

В данном разделе мы проанализируем и сравним общие параметры фреймворков, а именно: требования, MVC, расширения, документацию и поддержку сообществом.

Рассмотрим основные требования к фреймворкам. Требования подразумевают необходимый набор инструментов и технологий для работы фреймворка. Обычно требования у всех современных PHP фреймворков одинаковые, но бывают и различия, ведь они реализуют в себе разные правила, паттерны и технологии. Список требований по Yii и Laravel приведены в таблице 1.2.2.1.

Таблица 1.2.2.1 – Требования к серверу

Требования	Yii 2	Laravel 5
Версия PHP	>=5.4.0	>= 5.6.4

Как видно из таблицы 1.2.2.1, для фреймворка Yii 2 требуется лишь установленный на сервере интерпретатор языка PHP версии не ниже 5.4.0, а для Laravel не ниже 5.6.4 и множество расширений. Из чего следует, что Yii менее требователен к серверу, нежели Laravel [6]. Но в последних версиях PHP почти все эти библиотеки идут в комплекте с языком программирования. Значит, что по требованиям к серверу эти два фреймворка почти идентичны и требуют установленные на сервер актуальные версии PHP. В последних версиях оба фреймоврка требуют наличие версии PHP 7.1.

Сравним архитектуру фреймворков. Оба фреймворка для организации кода используют архитектурный паттерн MVC. MVC – это схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо. При подробном рассмотрении:

- Model – это само приложение которое ничего не знает о HTTP запросах;

- View – это HTTP запрос/ответ и представление данных, которые требует клиент от сервера;

- Controller – это несколько и более классов, чья задача – абстрагирование модели от HTTP.

По архитектуре данные фрейморки идентичны, так как реализуют один и тот же архитектурный паттерн. Он позволяет быстро реализовать проект, а также легко сопровождать и масштабировать его в дальнейшем [7].

Расширения не маловажная часть фреймворка, поэтому рассмотрим их подробнее. Расширения – это возможность подключать дополнительные модули или библиотеки для фрейморка, которые позволят расширить его возможности. Yii и Laravel поддерживают такую возможность. На данный момент, расширений гораздо больше, конечно же, у Yii, так как фреймворк живет гораздо дольше, чем Laravel, однако второй, в свою очередь, развивается очень большими темпами и включает в себя много всего для работы сразу «из коробки». Расширения имеются абсолютно для любой задачи, будь то панель администратора или же платежная система [8]. Однако готовые расширения – это очень спорный вопрос, так как этот код написан другими людьми, то никаких гарантий он не несет. Как в Yii, так и в Laravel очень много базовых и необходимых расширений разработаны своими создателями, что позволяет не беспокоиться о надежности их использования, к тому же большинство расширений в Laravel перетекло из Symfony, поэтому он является конкурентом для Yii. Следовательно, в данном пункте Yii является более продвинутым в плане готовых решений.

Сравним полноту документации, предоставленную разработчиками по их продуктам. Немало важным фактором при выборе фреймворка является его документация. Из-за малого количества лет разработки или личных убеждений авторов документация Laravel очень скудна и неинформативна. Поэтому для обучения придется смотреть исходный код, и как следствие, не всегда будет понятно, что делает та или иная функция. В Yii же имеет обширная и полезная

документация, где описана каждая функция с указанием ее назначения и примера использования. Так что, несомненно, это огромный плюс фреймворка Yii, и на спорные вопросы (как это работает? есть ли тут такая функция?) уйдет очень мало времени, вследствие чего скорость разработки и обучения увеличатся в разы, что немаловажно для фрейморка. Ведь главная его задача – ускорить разработку.

Поддержка сообществом также важна, поэтому рассмотрим уровень поддержки у обоих фреймворков. Именно сообщество пишет расширения, помогает разработчикам развивать свой фреймворк и устранять баги. Да данном этапе стоит учитывать не только сообщество в целом, но и языковой сегмент. Поэтому если рассматривать мировое сообщество, то laravel и Yii стоят примерно на одной позиции, и через пару лет Laravel догонит Yii по количеству использования его в рабочих проектах. Однако в России намного популяренее именно Yii, вследствие чего учебного материала, форумов, единомышленников гораздо больше именно на нем [9].

Так что если требуется поддерживать проект, то гораздо проще найти человека, который будет уже разбираться в этом. Laravel на данный момент очень слабо популяризирована в России, поэтому и поддержка русскоязычного сообщества очень маленькая. Но сообщество активно развивается, что дает отличные шансы в будущем.

1.2.3 Сравнение по возможностям решения технических задач

В данном разделе проанализируем и сравним возможности решения технических задач, а именно: ORM, безопасность, кэширование, REST API, тестирование и отладка.

Сравним поддержу ORM. ORM (англ. Object-Relation Mapping, рус. объектно-реляционное отображение) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют

как проприетарные, являющиеся частной собственностью авторов или правообладателей и не удовлетворяющие критериям свободного ПО (наличия открытого программного кода недостаточно), так и свободные реализации этой технологии. В таблице 1.2.3.1 приведены поддерживаемые и реализованные популярные ORM:

Таблица 1.2.3.1 – Поддерживаемые фреймворками ORM

ORM	Yii 2	Laravel 5
DAO (Data Access Object)	Идет вместе с фреймворком	Идет вместе с фреймворком
Active Record Pattern	Active Record	Eloquent ORM
Doctrine 2	Через плагины	Через плагины

DAO – это объект, который предоставляет абстрактный интерфейс к какому-либо типу базы данных или механизму хранения.

Active Record (AR) – шаблон проектирования приложений. AR является популярным способом доступа к данным реляционных баз данных в ООП.

Data Mapper (DM) – это уровень доступа к данным, который выполняет двунаправленную передачу данных между постоянным хранилищем данных (часто реляционной базой данных) и представлением данных в памяти (уровень домена). Цель шаблона – сохранить представление в памяти и постоянное хранилище данных независимо друг от друга и самого модуля отображения данных. Уровень состоит из одного или нескольких картографов (или объектов доступа к данным), выполняющих передачу данных [10].

Выбор конкретного подхода Active Record vs Data Mappers + Repository должен вытекать из бизнес задач.

Дело в том, что Active Record является фактический неотъемлемой частью RAD (rapid application development), и цель его - быстро развернуть уровень доменных моделей (читай бизнес логики) по существующей схеме базы (как правило, небольшой). Фактически все RAD проекты строятся по принципу

«Database First», т.е. сначала проектируется схема БД, затем она декларируется (Json, Yaml, SQL, ini) и через эту декларацию фреймворк генерирует доменные модели с уже реализованным CRUD функционалом через механизм scaffolding-а.

Это хорошо работает на маленьких и средних проектах с небольшими зависимостями. Возникают проблемы в случае, если проект большой и людей, которые работают на нем, много, и ,чтобы поддерживать все в рабочем состоянии, люди хотят покрывать код тестами [11].

Поэтому в последнее время такую большую популярность и обрел DDD (Domain driven design), который декларирует одним из своих основных принципов persistence ignorance и «Code First» [12]. В этом случае разработчик получает преимущества: SOLID, доменные модели становятся тонкими и содержат только бизнес логику. В дальнейшем все сводится к тому, что в таком виде можно легко реализовать Unit of Work паттерн и разгрузить слой приложения (контроллеров) от обязанностей сохранять и контролировать, модели сохраняются и обновляются корректно. А эта обязанность переходит в инфраструктурный уровень. Но при этом, сложность проекта становится выше [13].

Оба фреймворка позволяют использовать обе реализации. Doctrine: в Laravel используются библиотеки Symfony, которые позволяют без проблем развернуть DM в фреймворке, с Yii тоже не возникает проблем, но принято считать, что в symfony реализовано все академически правильно. Yii AR и Eloquent: тут, бесспорно, лучшее решение - это Yii AR, потому что оно быстрее, чем Eloquent, в таблице 1.2.3.2 приведены тесты по скорости работы ORM с сайта github [14]:

Таблица 1.2.3.2 – Скорость работы ORM

ORM	Время (мс)	Память (кб)
eloquent	34.46	673.80
yii2	9.09	835.82

Безопасность - важная часть в современных приложениях, поэтому остановимся на ней подробно.

Безопасность - одна из самых главных задач, стоящих перед разработчиками, а ее реализация вручную занимает очень много времени, поэтому очень важно, чтобы в фреймворке были все инструменты для использования общепринятых стандартов безопасности. Надо отметить, что в обоих фреймворках есть инструменты для работы с паролями, авторизацией/аутентификацией, защитой от SQL-инъекций, Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF) и т.д. [15]. В таблице 1.2.3.3 представлен список используемых технологий:

Таблица 1.2.3.3 – Безопасность фреймворков

Yii 2	Laravel 5
Access Control Filter (ACF)	ACF
Role Based Access Control (RBAC)	RBAC
Расширения для авторизации по OpenID, OAuth или OAuth2 и пр.	Access Control Lists плагины

Если рассматривать базовый вариант, то в этом случае Laravel более обширен в своих возможностях, и все то, что есть в Yii можно установить в Laravel с помощью расширений [16]. Оба фреймворка реализуют базовые функции защиты, но так как разработчика интересует скорость, то Laravel будет куда более выгоден в плане скорости разработки.

Рассмотрим возможности кэширования. Кэш – это специальный промежуточный буфер с очень быстрым доступом, содержащий данные, которые могут быть запрошены с наибольшей вероятностью [17].

Для оптимизации работы с сетью применяется механизм сохранения полученных однажды по HTTP документов в кэше для их повторного использования, при этом без обращения к серверу-источнику. Документ, который сохранен в кэше, будет доступен при последующем обращении к нему, при этом без выгрузки с сервера-источника. Это призвано увеличить скорость доступа клиента

к нему, а также снизить расход трафика сети [18]. Есть множество способов кэширования, но конкретной, уникальной ее реализации в этих фреймворках нет. Поэтому разница лишь в количестве поддерживаемых ими способов. В таблице 1.2.3.4 приведен список поддерживаемых технологий:

Таблица 1.2.3.4 – Список поддерживаемых технологий кэширования

Технология кэширования	Yii 2	Laravel 5
APC	+	+
Database	+	+
File	+	+
Memcached	+	+
Redis	+	+
WinCache	+	-
XCache	+	-
Zend Data Chace	+	-

Как видно по сводной таблице 1.2.3.4, Yii поддерживает больше видов кэширования, чем Laravel.

Остановимся подробнее на поддержке REST API. REST – архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине. REST является альтернативой RPC [19].

В сети Интернет вызов удалённой процедуры может представлять собой обычный HTTP-запрос (обычно «GET» или «POST»; такой запрос называют

«REST-запрос»), а необходимые данные передаются в качестве параметров запроса.

Для веб-служб, построенных с учётом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «RESTful».

В отличие от веб-сервисов (веб-служб) на основе SOAP не существует «официального» стандарта для RESTful веб-API. Дело в том, что REST является архитектурным стилем, в то время, как SOAP является протоколом. Несмотря на то, что REST не является стандартом сам по себе, большинство RESTful-реализаций используют стандарты HTTP, URL, JSON и XML.

В конечном счете, в настоящее время все современные PHP фреймворки обязаны поддерживать REST архитектуру. Ниже приведен список возможностей фреймворков для работы с REST API.

Yii 2:

- поддержка JSON, JSONP и XML;
- роутинг в соответствии с REST-запросами;
- поддержка HATEAOS;
- кэширование запросов;
- ограничение скорости.

Laravel:

- настройка роутинга REST-запросов;
- поддержка JSON, JSONP.

В этом параметре Yii также более функционален, чем Laravel.

Наиболее важным качеством фреймворка является то, что это инструмент для тестирования приложений, сравним эти возможности.

Тестирование системы очень важно. После очередного изменения что-то перестанет работать, так как надо – это гарантированно. И тогда поиск ошибки может отнять очень много времени. Модульное тестирование может свести этот процесс к считанным минутам. Не будем также исключать переезд на другую платформу и связанные с ним «подводные камни». Чего только стоит разность

в точности вычислений на 64- и 32-разрядных системах. В обоих фреймворках реализуется базовый инструмент тестирования PHPUnit [20].

PHP Unit – это базовый тестировочный инструмент. PHPUnit был выпущен достаточно давно – в 2004 году, что означает его самодостаточность и зрелость.

Плюсы PHP Unit:

- выпущен давно, очень популярен, поэтому к нему подготовлена хорошая документационная база, можно найти множество тренингов и ссылок на форумах Q&A;

- базовый, а значит простой. С него очень легко начинать, если вы сталкиваетесь с тестированием впервые;

- PHP Unit, пожалуй, самый известный инструмент для Unit тестирования.

Минусы PHP Unit:

- PHP Unit идеально подходит для Unit тестирования, но API тестирование – это немного другое, поэтому, наверное, не стоит его использовать для API тестирования или приемочного тестирования, поскольку это на уровень выше Unit тестирования;

- возможности PHP Unit очень ограничены. Его легко понять, но по мере того, как вам будут необходимы новые уровни тестирования, вам придется комбинировать различные инструменты с PHP Unit, либо придется написать много кода, чтобы протестировать API.

Именно из-за минусов были написаны и другие средства тестирования. Различие между Yii и Laravel заключается в том, что в первом используется Codeception, а во втором средства тестирования Symfony. Оба они реализуют одно и то же, так что особой разницы в использовании нет.

Остановимся подробнее на возможности отладки приложения. При разработке отладка кода занимает большую часть времени. Поэтому наличие средств для ее выполнения очень полезно и важно для программиста. Yii и Laravel предоставляют одинаковые средства для отладки, это Консоль отладки и логирование, но из базовых решений, не прибегая к расширениям, намного удобнее и информативнее консоль отладки у Yii, потому что в Laravel зачастую сложно

понять, какая ошибка была сделана, однако это решается установкой расширений.

1.2.4 Вывод по разделу 1.2

На основе проведенного анализа был сделан выбор в сторону активно развивающегося Laravel, его структура позволяет легко масштабировать наше приложение. Yii является более стабильным, но менее масштабируемым, для нашей задачи он не подходит.

1.3 Оптимизация высоких нагрузок

Highload (высокие нагрузки) – это название состояния инфраструктуры, которая требует того, чтобы ее оптимизировали и масштабировали.

1.3.1 Основная информация

Прежде всего, highload (высокие нагрузки) – крайне относительное понятие. Оно никогда не измеряется количеством запросов или скоростью работы сайта, т.к. попросту нет такого понятия, как "средний сайт". Все сайты специфичны и одинаковое количество запросов может приводить к совершенно разным нагрузкам на разные ресурсы.

Необходимость в масштабировании появляется тогда, когда текущая инфраструктура начинает показывать первые признаки того, что она перестает справляться с нагрузкой [21].

Чтобы понять, что проблема есть, ее необходимо диагностировать. Именно поэтому при любой нагрузке нужна хорошая система мониторинга. Она поможет определить тот самый момент, когда пора масштабировать [22]. Чтобы решать проблему оптимизации нужно проверить web-сервер, базу данных (MySQL) и PHP.

Правильно настроенный Web-сервер позволит существенно разгрузить «железо». Если на сайте есть картинки и прочие файлы, которые никогда не меняются, нужно правильно настроить сервер на отдачу файлов. Наиболее часто проблемы встречаются именно в базах данных. Анализ работы приложения, в нашем случае написанном на PHP, (профилирование) позволит определить слабые места [23].

1.3.2 Основы оптимизации

Оптимизация позволит ускорить работу сервера, отображение страниц в браузере и в целом справиться с высокими нагрузками, если нет варианта улучшить физические показатели сервера [24].

1.3.2.1 Оптимизация статических ресурсов

Статика – это статические ресурсы, которые загружаются пользователем при переходе на какую-нибудь web-страницу приложения, например, js-файлы, файлы стилей (css), картинки и другая не динамическая информация, которая меняется на сайте очень редко [25]. Её оптимизация позволит ускорить загрузку страниц. Рассмотрим несколько вариантов оптимизации.

Картинки больше всего нуждаются в оптимизации, поэтому рассмотрим подробнее. Первое и самое основное, что нагружает страницы и является статичной информацией являются картинки, поэтому очень важно подобрать правильный формат изображения. PNG или JPEG [26].

PNG позволяет показывать картинки без потери мельчайших деталей и с точной передачей цвета. Существуют палитры PNG24 и PNG8, если на картинке используется небольшое количество цветов, то стоит перекодировать изображение в PNG8, что поможет существенно уменьшить размер изображения. Формат PNG удобен для:

- иконок;

- малоцветных иллюстраций;

- изображений, в которых требуется большая четкость мелких деталей (размер такого изображения может быть огромным, поэтому лучше избегать таких ситуаций).

JPEG использует максимально доступную палитру цветов. Для уменьшения размера используется специальный механизм сжатия и сглаживания. JPEG поддерживает прогрессивный формат (Progressive JPEG) [27]. При открывании такой картинки в браузере пользователь сначала увидит общие очертания, потом произойдет детализация, и качество повысится до максимального. Это составит впечатление о более быстрой загрузке сайта. Особенно важно использование progressive формата для случаев с низкой скоростью доступа в Интернет у посетителей [28]. Также JPEG позволяет указать уровень сжатия при сохранении изображения. Это снижает качество. Но иногда снижения качества незаметно, зато экономия размера может получиться довольно большой. Формат JPEG удобен для:

- фотографий;

- скриншотов;

- многоцветных иллюстраций.

В основном, оптимизация картинок направлена на снижение их размера. Например, JPEG в настоящее время содержит также множество мета-данных, которые можно вырезать. Основные правила оптимизации картинок в web:

- использовать разный уровень сжатия JPEG;

- использовать Progressive JPEG;

- использовать разные палитры для PNG;

- обрабатывать изображения только алгоритмами без потери качества;

- не сохранять текст в виде изображений;

- использовать заранее уменьшенные изображения;

- использовать css-спрайты для иконок, чтобы клиент делал только один запрос на скачивание файла;

- правильная оптимизация картинок может значительно ускорить работу

сайта.

Следующие, нуждающиеся в оптимизации статические ресурсы – это файлы. Файлы также подвержены оптимизации. Как и картинки, в основном их просто сжимают по размеру [29]. Раз их тоже загружают пользователи, то это тоже влияет на скорость загрузки сайта.

Если говорить о файлах с кодом, таких как css, js и html, то их можно минифицировать. Минификация – это простой подход для уменьшения размеров файлов. При сжатии убираются все комментарии к коду, лишние пробелы, отступы, переносы строк. Нужно обязательно минифицировать js и css файлы, а сами файлы помечать припиской к названию «.min.». Метод наглядно показан в рисунке 1.3.2.1.1, где произошло сжатие на 17,2%.

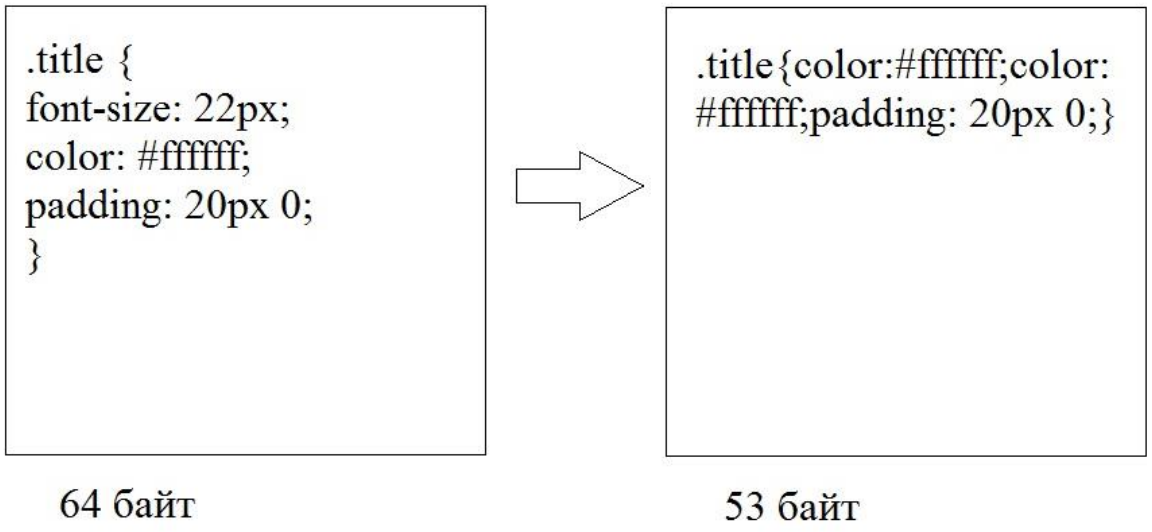


Рисунок 1.3.2.1.1 – Пример минификации

Еще одним методом оптимизации файлов является gzip-сжатие. Современные браузеры поддерживают сжатые файлы, так как является частью спецификации HTTP 1.1. Сжатие текстовых форматов (html, css, js) происходит при отдаче ответа от сервера клиенту [30]. Сервер сжимает, а браузер получив файл его распаковывает и показывает результат. Схема работы метода сжатия gzip показана на рисунке 1.3.2.1.2.

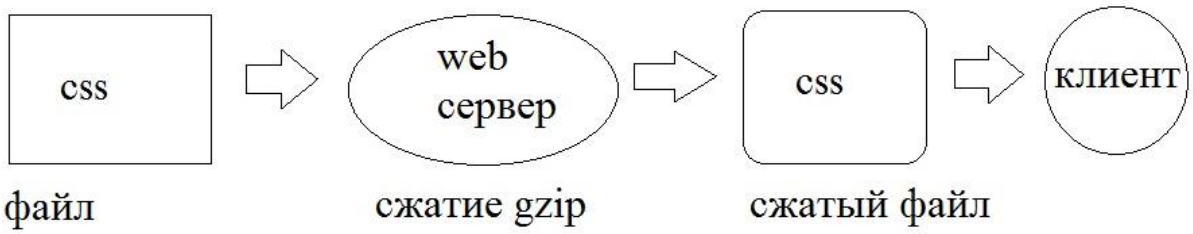


Рисунок 1.3.2.1.2 – Схема работы gzip сжатия css файла

На сервере nginx можно настроить уровень сжатия от 1 до 9 (от худшего к лучшему). Самое оптимальное значение по середине, то есть 5. Сжимать можно любые текстовые файлы:

- HTML;
- XML;
- css;
- js;
- txt;
- другие текстовые форматы.

Кэширование один из самых действенных способов ускорения работы приложения. Клиентское кэширование – это способность браузера локально сохранять файлы, чтобы не делать повторной загрузки при последующем обращении [31]. Логично использовать его для картинок, js и css файлов, так как при посещении различных страниц одного сайта клиент будет загружать много раз одни и те же файлы.

Работа запроса без кэширования показана на рисунке 1.3.2.1.3.

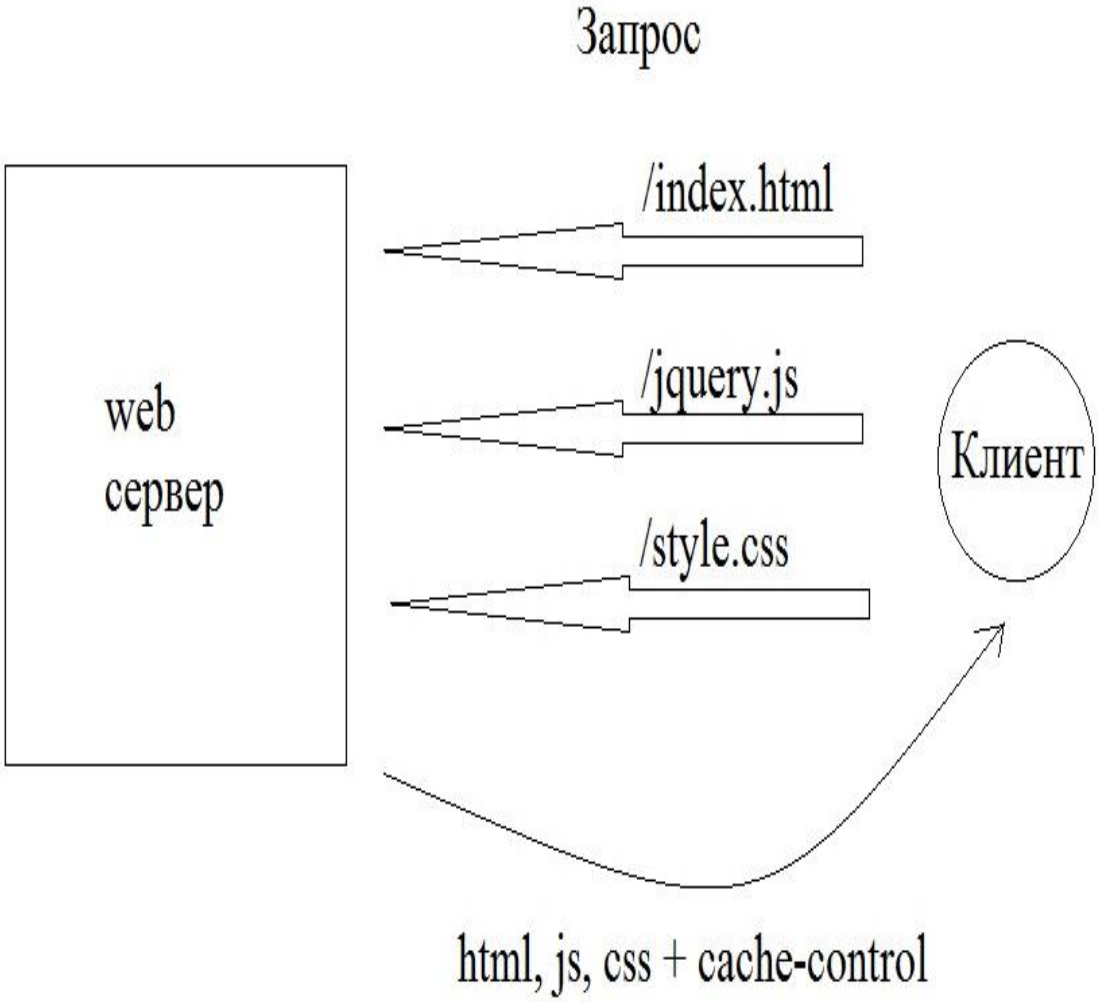


Рисунок 1.3.2.1.3 – Загрузка страницы без использования кэширования

Управление кэшированием осуществляется с помощью html-заголовков Cache-control и Expires [32]. Сервер отправляет этот заголовок вместе с файлом, так браузер понимает сохранять этот файл или нет. Если файл был сохранен, то при последующем запросе он будет загружен из кэша браузера без запроса на сервер. Работа кэширования показана на рисунке 1.3.2.1.4.

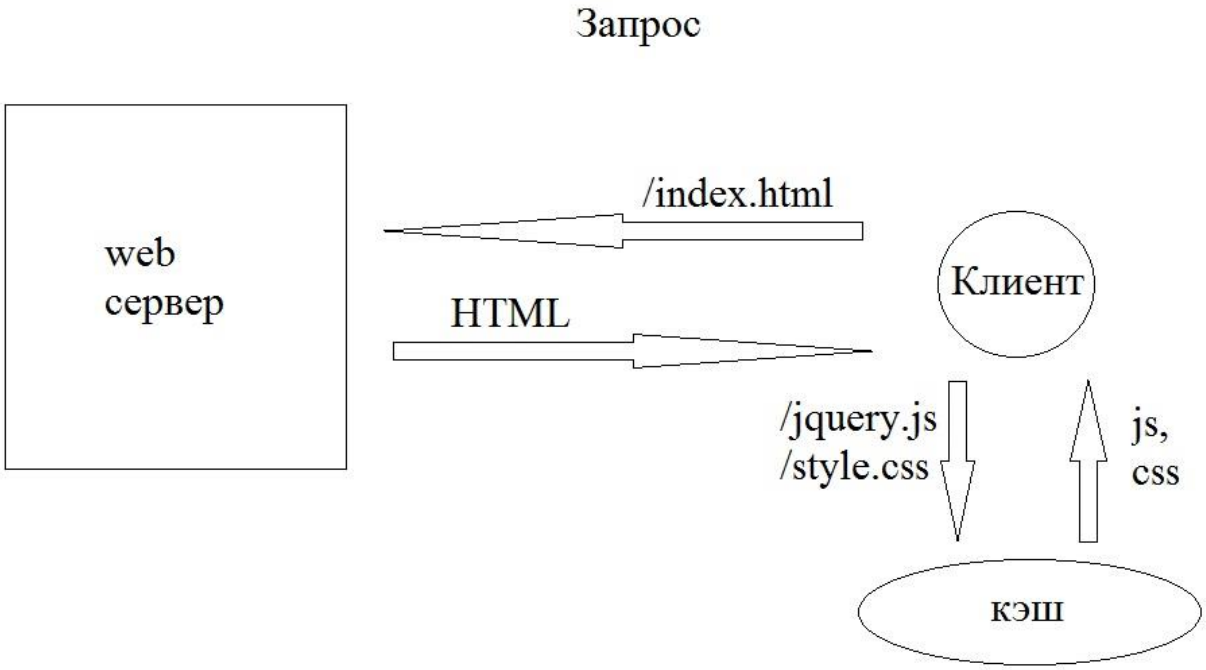


Рисунок 1.3.2.1.4 – Загрузка страницы с использованием кэширования

Заголовок Cache-control имеет следующую форму записи: Cache-control: private, max-age=0, no-cache [33]. Заголовок Exiries позволяет указать дату и время, когда браузер должен обновить кэш.

Кэшировать нужно все, что обновляется реже, чем раз в несколько минут, главное, правильно указать время кэширования.

1.3.2.2 Клиентская оптимизация

Оптимизация клиентской части позволит сэкономить значительное количество ресурсов и обеспечить высокую скорость работы ресурса для пользователей [34]. Даже если проблем на сервере нет, сайт может работать значительно быстрее.

Все методы оптимизации статических ресурсов подходят под клиентскую оптимизацию. Также в нее можно включить CDN и мобильную оптимизацию.

Остановимся на способе масштабирования – CDN системе. CDN (Content Delivery Network) – это специальная технология, которая позволяет посетителю получать содержимое сайта из разных географических мест [35]. Данная технология нужна нам тогда, когда нужно быстро доставлять контент в удаленные географические точки от физического положения сервера. Например, сервер находится в Москве, а клиенты в США, России и Англии, тогда сайт для клиентов из России будет загружаться быстро, а для США и Англии медленнее. Тут следует использовать технологию CDN [36].

Технология проста: создается еще один сервер в нужном географическом месте и туда копируется содержимое нашего сайта. Есть смысл использовать только те ресурсы, которые часто запрашиваются, но при этом редко меняются. Принцип CDN наглядно показан на рисунке 1.3.2.2.1.

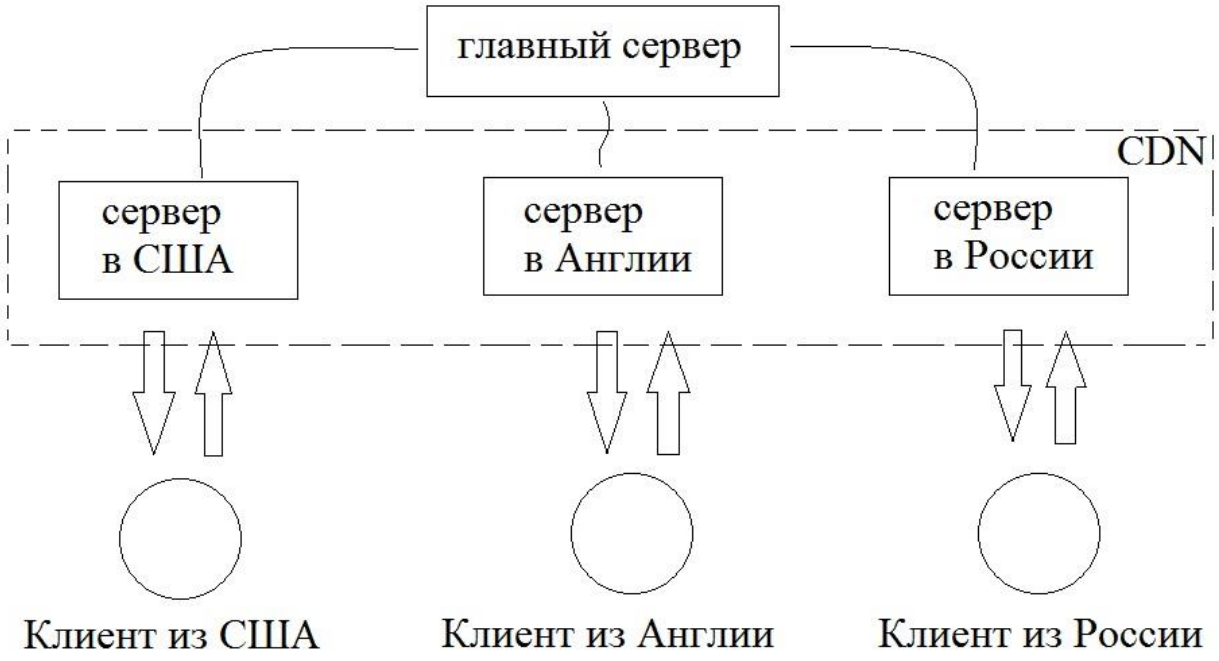


Рисунок 1.3.2.2.1 – Пример работы CDN

В современном мире половина интернет-трафика идет через мобильные устройства, поэтому остановимся на нем подробнее.

В 2018 году большая часть трафика идет через мобильные устройства. Поэтому есть смысл оптимизировать сайты под мобильные устройства, если они в этом нуждаются [37]. Есть несколько основных способов, которые помогут оптимизировать web-приложение:

- использовать асинхронную загрузку js и css файлов, потому что браузер всегда ждет полной загрузки перед отдачей страницы пользователю;

- использовать только HTML 5, flash-технологии не поддерживаются на большинстве мобильных устройств;

- настроить viewport для корректного отображения сайта по ширине экрана;

- настроить размер текста, чтобы его было удобно читать, и величину элементов управления, чтобы на них было удобно нажимать на tap-устройствах.

1.3.2.3 Серверная оптимизация

Web-сервер это по сути самое главное звено в работе веб-сайта. Он принимает запросы от пользователя, формирует и отправляет ответ. Чем больше таких запросов, тем медленнее работает сам сервер. Также важна оптимизация самого приложения и базы данных, но она происходит в процессе разработки самого приложения или сайта.

Рассмотрим варианты оптимизации сервера.

Существует несколько способов оптимизировать работу сервера:

- сжатие запросов;

- уменьшение количества запросов;

- настройка ресурсов.

Сжатие запросов gzip и уменьшение запросов разобрано в пунктах клиентской и статической оптимизации [38]. Однако без дополнительной настройки web-

сервера он скорее всего не использует все доступные ресурсы платформы. Такая настройка называется «тюнинг».

Все настройки nginx производятся в файле nginx.conf, вносятся оптимальные настройки в его работу. Максимальное количество соединений, которые nginx может обслуживать, считается по формуле « всего соединений = worker_processes x worker_connections », где worker_processes — количество рабочих процессов, а worker_connections — максимальное количество соединений одного рабочего процесса. Установка worker_processes в режим auto и worker_connections в числе 1024 (допустимые значения от 1024 до 4096) даст оптимальное распределение нагрузки. Также необходимо настроить метод соединения, для Linux это команда «use epoll», для Freebsd «use kqueue». Для обработки запросов нужно настроить максимально возможное количество соединений, включить метод отправки данных sendfile (он более эффективен, чем стандартный read+write) и настроить отправку заголовков и начала файла в одном пакете. Также лучше выключить основное логирование и оставить только запись критических ошибок, так можно сэкономить дисковое пространство.

Следующий этап после настройки сервера - это включение кэширования запросов.

Сервер nginx умеет кэшировать запросы самостоятельно. Суть серверного кеширования в том, чтобы не генерировать постоянно одни и те же скрипты, что может иногда занимать целые секунды. Вместо этого приложение генерирует страницу один раз, и результат сохраняется в память. Когда посетитель запросит ту же страницу второй раз, генерации уже не будет, а клиент получит сохраненную в памяти версию [39]. Раз в какое-то время (называемое ttl) эта сохраненная версия будет удаляться и генерироваться новая, чтобы поддерживать актуальность данных. Почти на всех сайтах можно кэшировать данные для неавторизованных пользователей, это уменьшит нагрузку на сервер [40]. Отлично подойдет для приложения по работе с изображениями, так как на нем ресурсы будут доступны публично.

Для реализации серверного кэширования необходимо задать настройки максимально возможного кэша (1 гигабайта вполне достаточно, но можно менять в зависимости от нужд). Чтобы кеширование заработало, нужно создать новый хост, который будет прослушивать 80 порт (listen :80). А основной хост перенести на какой-то другой порт (например, 81). Кеширующий хост будет посылать запросы на основной либо отдавать данные из кэша. Работа такого кэширования представлена на рисунке 1.3.2.3.1.

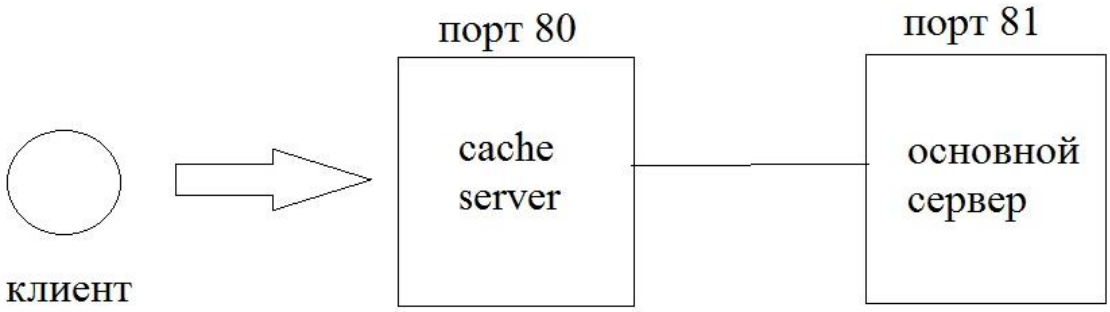


Рисунок 1.3.2.3.1 – Пример наличия кэширующего сервера

Также можно закэшировать ошибочные запросы и ответы fastCGI, для уменьшения количества ошибочных запросов на сервер, это тоже значительно сократит количество запросов к серверу.

2 Описание проведенной исследовательской работы

После анализа посещаемости различных тематических сайтов было выделено несколько направлений, которые были наиболее нагружены:

- приложения, связанные с графическим контентом, например, ArtStation;
- приложения-хранилища, например, GitHub;
- социальные сети, например, «В контакте».

В таблице 2.1 приведены сведения о посещении самых популярных ресурсов, данные взяты из открытого источника [https://a.pr-cy.ru/].

Таблица 2.1 – Сравнение посещаемости

Название web-приложения	Просмотров в сутки
ArtStation	234 935
GitHub	1 317 091
В контакте	3 489 619

Не слишком ресурсозатратная разработка и почти полное отсутствие аналогов в России, это приложения, связанные с графическим контентом. Было решено разработать web-приложение, которое бы предоставляла художникам площадку, включающую в себя поиск новых поклонников и одновременно являющаяся их портфолио, далее – Liwart (название приложения). Приложение включается следующие функции:

- система регистрации пользователей;
- личные профили;
- добавление/редактирование/удаление работ (рисунков);
- комментарии и отметки «мне нравится».

2.1 Разработка структуры приложения

Прежде чем приступить к разработке приложения, нужно обратиться к теоретическим материалам и разработать базовую схему приложения. В высоконагруженном приложении всегда самые слабые места там, к чему будет идти больше всего обращений. В нашем случае - это база данных. Поэтому нужно выбрать подходящую субд (систему управления базой данных). В работе была выбрана система MySQL. Схема приложения продемонстрирована на рисунке 2.1.1

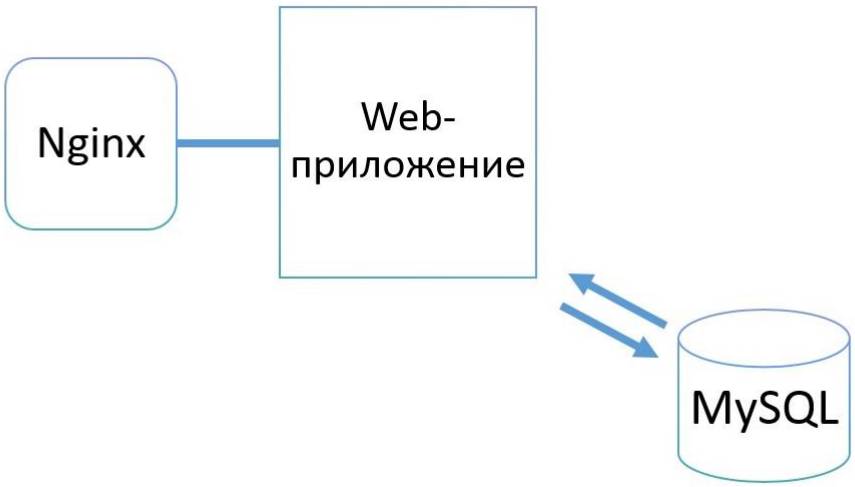


Рисунок 2.1.1 – Базовая структура приложения

Далее надо решить проблему с системой хранения данных, так как будет множество обращений к файлам(картинкам), система должна выдержать и отдавать файлы без задержек, нужно вынести СХД отдельно от сервера. В итоге была получена базовая схема приложения, устойчивая к высоким нагрузкам. Схема продемонстрирована на рисунке 2.1.2.

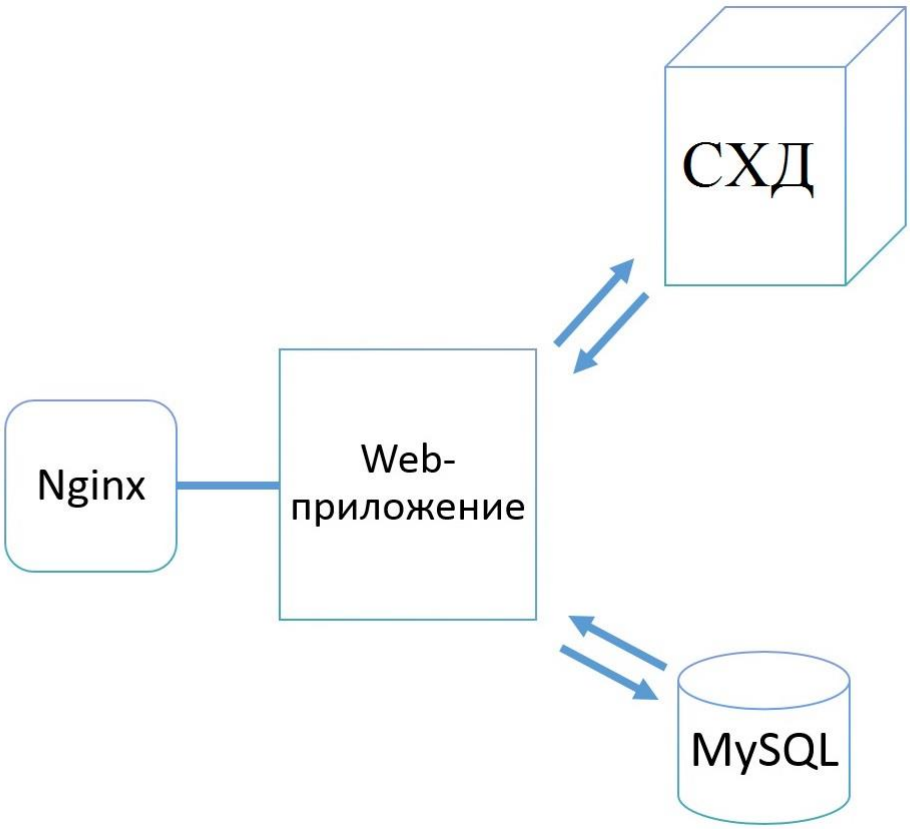


Рисунок 2.1.2 – Базовая схема приложения устойчивого к высоким нагрузкам

Следующий этап – решить, что будет лежать в основе самого web-приложения - самописное ядро или готовое. Фрейморк позволит в разы ускорить разработку и быть уверенным в том, что базовые функции приложения будут реализованы правильно и без ошибок, поэтому был выбран php-фремворк Laravel, его выбор был подробно описан в пункте 1.2.

Само приложение будет построено на принципе MVC (Model-View-Controller) и самоподгрузе классов с единой точкой входа в приложения. То есть будет папка public, в которой расположена точка входа приложения (файл index.php). За его приделам будет расположены системные файлы приложения, такие как модели, конфигурации, контроллеры и так далее. Строение проекта отображено на рисунке 2.1.3.

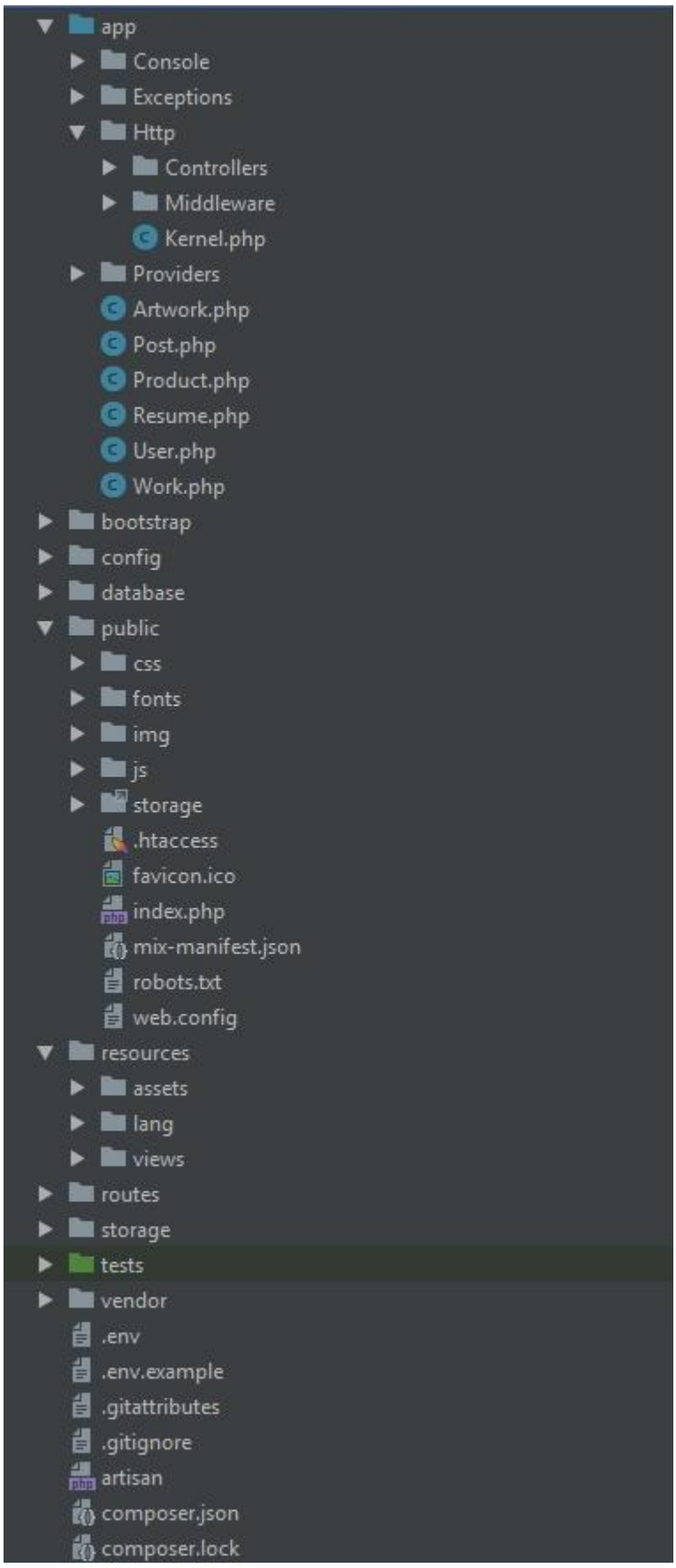


Рисунок 2.1.3 – Структура web-приложения

2.2 Выявление и постановка задач для решения

В ходе разработки приложения появляются некоторые проблемы и задачи, которые невозможно предвидеть на этапе проектирования. Для их обнаружения были использованы специальные расширения для тестирования php-приложения, запросов к MySQL и т.д.

Задачи и проблемы, которые сильно влияли на работу приложения перечислены ниже:

- большое количество простых запросов в одном методе;
- медленная загрузка файлов при большом количестве запросов к СХД;
- долгая первая загрузка приложения;
- повторяющиеся запросы к базе данных.

Данные проблемы сильно уменьшают производительность приложения и его скорость работы в целом. Поэтому необходимо их решить, применив один или несколько методов борьбы с высокими нагрузками.

2.3 Применение оптимизации и масштабирования

2.3.1 Оптимизация работы приложения

В самом приложении по проведенным тестам было все отлично, кроме времени первой загрузки. Выход из этой ситуации – кэширование таблицы маршрутов. Время первой загрузки приложения изменилось с 5с на 0,6с. Схема продемонстрирована на рисунке 2.3.1.1.

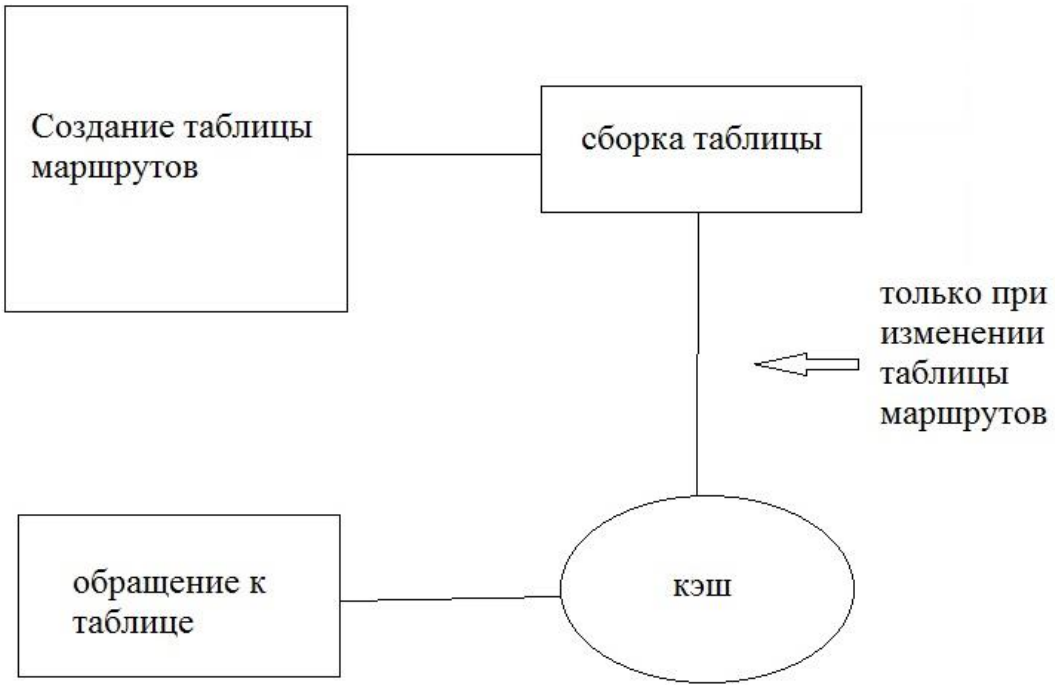


Рисунок 2.3.1.1 – Схема кэширования таблицы маршрутов

2.3.1.1 Оптимизация PHP

Важная часть оптимизации работы приложения – это оптимизация самого PHP. Настройки производятся в файле php.ini, настраиваемые параметры, которые позволили сэкономить ресурсы:

- memory_limit = 32M – максимальная выделяемая память для работы скрипта;

- zlib.output_compression = Off – сжатие применяется на стороне web-сервера;

- max_execution_time = 5 – максимальное время работы скрипта 5 секунд, позволяет отключать долгие или зависшие обращения к серверу в автоматическом режиме;

- zend.enable_gc = On – включает удаление ненужных остаточных файлов и очистку неиспользуемой памяти в фоновом режиме.

Данные параметры помогут решить проблему слишком долгого выполнения запросов или чрезмерного употребления памяти скриптами. В некоторых обстоятельствах могут значительно ускорить работу приложения.

2.3.1.2 Оптимальная настройка MySQL

На данном этапе достаточно увеличить параметр immodb_buffer_pool_size до 70-80% от общего количества оперативной памяти на сервере, так как бд кэширует данные и индексы. Прирост ответов бд вырастает примерно в 2 раза. Время выполнения запроса select * from table уменьшилось с 60мс до 6,26мс.

2.3.2 Оптимизация сервера

2.3.2.1 Настройка VPS

Настройка сервера позволяет правильно распределить ресурсы и дать возможность приложению использовать их по максимуму. Все настройки производятся в файле конфигурации nginx.conf:

- worker_processes = 1 – должен соответствовать количеству ядер на сервере;

- expires max – для кэширования статичного контента, который не будет меняться;

- access_log off – ненужные дисковые операции;

- unix socket – настраиваем php-fpm для быстрой работы php.

Данные настройки позволили освободить 100мб оперативной памяти на сервере.

2.3.2.2 Масштабирование СХД

Самой основной проблемой были работы с файлами. Расширение дискового массива или покупка дополнительного оборудования было не целесообразным в связи с дороговизной данного варианта. Поэтому было принято решение использовать облачное хранилище данных с системой CDN. Файловая система Rackspace. Управляется простыми запросами к api серверу компании, предоставляющей услуги. Данное решение исключило проблему с долгой загрузкой картинок, а также проблему с ожиданием сервера для географически удаленных клиентов. Время загрузки картинок уменьшилось с 0,5с до 0,1с.

2.3.2.3 Включение серверного кэширования

Включение серверного кэширования действительно решает множество проблем, а самое важное, он снижает нагрузку на сервер. Работу кэширующего сервера можно прочитать в описательной части.

Настройки:

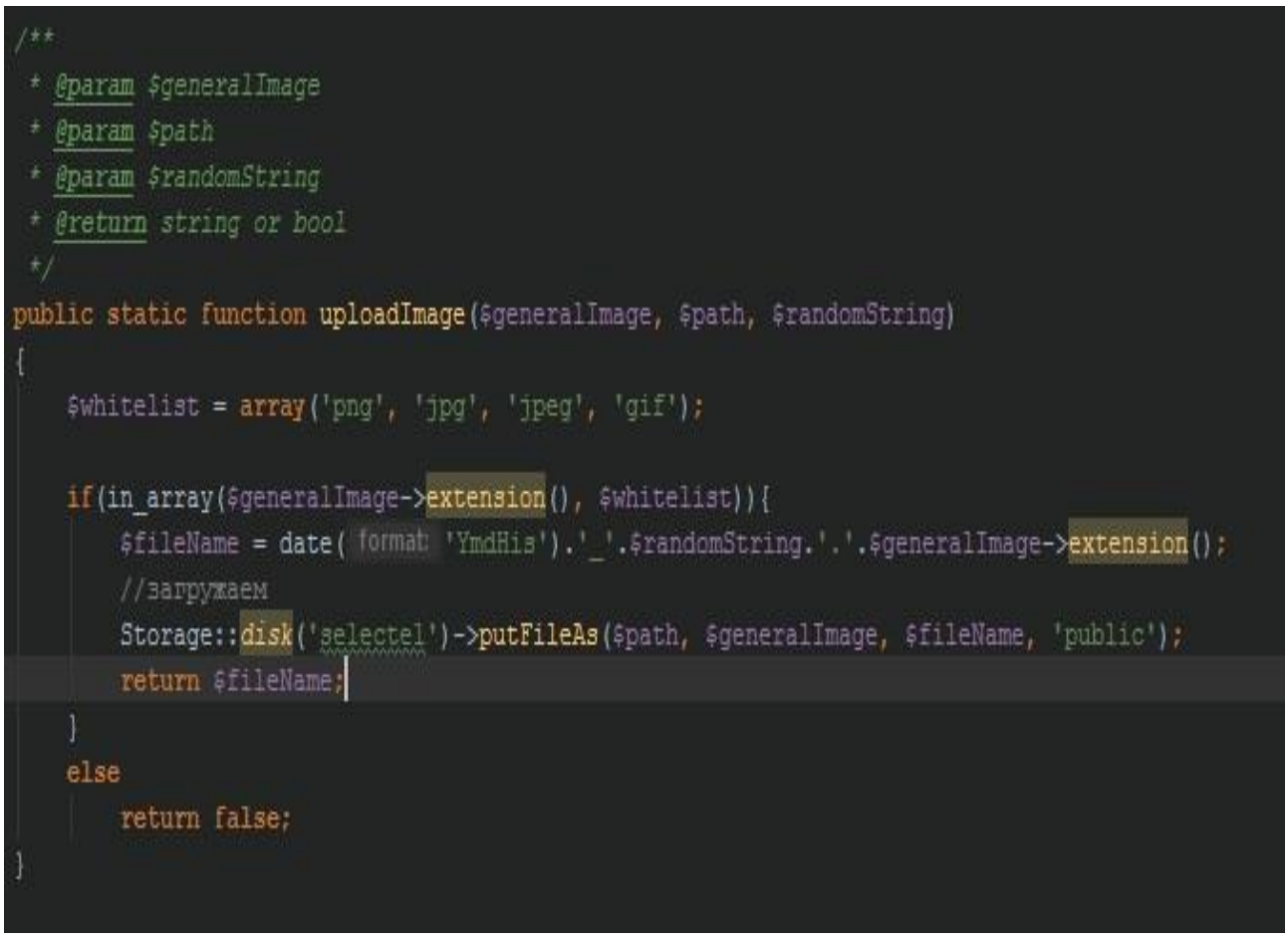
- cache_size = 1G;
- cache_valid = 2m.

Данные настройки позволят кэшировать запросы пользователей на 2 минуты, что убирает проблему многократного исполнения одинаковых запросов.

2.4 Результат работы

Были применены различные методы к решению проблем высоких нагрузок.

Для загрузки картинок и работы с СХД было выбрано решение перенести все в облачное хранилище с CDN, на рисунке 2.4.1 показан пример загрузки картинки в облако.



```
/**
 * @param $generalImage
 * @param $path
 * @param $randomString
 * @return string or bool
 */
public static function uploadImage($generalImage, $path, $randomString)
{
    $whitelist = array('png', 'jpg', 'jpeg', 'gif');

    if(in_array($generalImage->extension(), $whitelist)){
        $fileName = date( format: 'YmdHis').'_'.$randomString.'.'.$generalImage->extension();
        //загружаем
        Storage::disk('selectel')->putFileAs($path, $generalImage, $fileName, 'public');
        return $fileName;
    }
    else
        return false;
}
```

Рисунок 2.4.1 – Функция загрузки файла в облако

Для обработки запросов бд было применено увеличение размера кэширования для MySQL и создание сложных запросов взамен множеству простых. Для оптимизации самого сервера были выключены и настроены его компоненты, что позволило сэкономить оперативную память. Конфигурация web-сервера продемонстрирована на рисунке 2.4.2.

```
server {
    server_name liwart.com www.liwart.com;
    charset UTF-8;
    index index.php;
    disable_symlinks if_not_owner from=$root_path;
    include /etc/nginx/vhosts-includes/*.conf;
    include /etc/nginx/vhosts-resources/liwart.com/*.conf;
    access_log /var/www/httpd-logs/liwart.com.access.log;
    error_log /var/www/httpd-logs/liwart.com.error.log notice;
    ssi on;
    return 301 https://$host:443$request_uri;
    set $root_path /var/www/admin/data/www/liwart.com/public;
    root $root_path;
    listen 185.26.97.93:80 default_server;
    gzip on;
    gzip_comp_level 5;
    gzip_disable "msie6";
    gzip_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript application/javascript;
    expires 60s;
    location / {
        try_files $uri $uri/ /index.php?$query_string;
        location ~ [^/]\.ph(p\d*|tml)$ {
            try_files /does_not_exists @php;
        }
    }
    location @php {
        fastcgi_index index.php;
        fastcgi_param PHP_ADMIN_VALUE "sendmail_path = /usr/sbin/sendmail -t -i -f webmaster@liwart.com";
        fastcgi_pass unix:/var/www/php-fpm/admin.sock;
        fastcgi_split_path_info ^((?U).+\.ph(?:p\d*|tml))(/?.+)$;
        try_files $uri =404;
        include fastcgi_params;
    }
}
```

Рисунок 2.4.2 – Конфигурация web-сервера

Также были применены все возможные методы кэширования: серверное, клиентское и кэширование запросов. Самую большую роль в скорости сыграл именно этот пункт оптимизации.

В общей сложности приложение было протестировано при обращении 50 запросов в секунду. Для работы с более большими объемами требуется более мощные характеристики сервера. Общая скорость работы приложения повысилась на 77%. График работы приложения под нагрузкой показан на рисунке 2.4.3.

Рисунок 2.4.3 – График длительной нагрузки на сайт

Общее время ответа приложения изменилось с 1,3 секунды до 0,3-0,6 секунд. Данные приведены в таблице 2.4.1.

Таблица 2.4.1 – результаты работы

Сравниваемый параметр	До оптимизации	После оптимизации
Загрузка картинок	0,5с	0,1с
Обработка запроса бд	60мс	6,26мс
Память на сервере	680мб	470мб
Время первичной загрузки	5с	0,6с
Общее время ответа приложения	1,3с	0,3с - 0,6с

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы было создано web-приложение для «тестирования» работы сайта при высокой нагрузке, подробно рассмотрен и показан весь процесс его создания.

Основные результаты работы состоят в следующем:

- выбраны технологии и средства разработки;
- изучены средства и методы для оптимизации под высокие нагрузки;
- разработана структура приложения;
- сформулированы задачи для решения;
- применены методы оптимизации для решения задач.

Основные выводы работы:

а) выбраны следующие технологии для реализации приложения: PHP, MySQL, JavaScript, http-сервер nginx;

б) изучены направления решений для высоконагруженных проетов: масштабирование и оптимизация;

в) структура приложения включает:

1) web-сервер nginx в связке с кэширующем сервером;

2) web-приложение на php, реализующее методологию ООП, модель MVC и системы автоподгрузки классов с единой точкой входа и таблицей маршрутов;

3) CDN Cloud Storage для хранения и работы с файлами;

4) Сервер БД MySQL.

г) в качестве задач для решения сформулированы:

1) уменьшить количество небольших запросов к бд в одном месте;

2) ускорить загрузки файлов при большом количестве запросов к СХД;

3) уменьшить времени первичной загрузки приложения;

4) убрать повторяющиеся запросы к базе данных.

д) применены следующие решения поставленных задач:

1) уменьшение запросов к бд – замена сложными запросами;

2) ускорение загрузки файлов – масштабирование СХД, перенос файлов в облачное хранилище;

3) уменьшение времени первичной загрузки – кэширование таблицы маршрутов;

4) повторяющиеся запросы к бд – оптимизация запросов в коде приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Wilde E. REST: From Research to Practice / E. Wilde, C. Pautasso. — Springer Science & Business Media, 2011. — 528 с.

2 Wikipedia свободный ресурс: [Электронный ресурс]. 2017 // (Рус.). — URL: https://ru.wikipedia.org/. [5 марта 2017].

3 Yii PHP framework best for developing Web 2.0 applications: [Электронный ресурс]. 2017 // (Англ.). — URL: http://www.yiiframework.com/ [2 апреля 2017].

4 Laravel is a trademark of Taylor Otwell: [Электронный ресурс]. 2017 // (Англ.). — URL: https://laravel.com/ [20 апреля 2017].

5 Уайнсет Дж. Разработка веб-приложений в Yii 2 / Дж. Уайнсет. – М.: ДМК Пресс, 2015. – 392 с.

6 Github built for developers: [Электронный ресурс]. 2017 // (Англ.). — URL: https://github.com/ [1 мая 2017].

7 Фаулер М. Архитектура корпоративных программных приложений / М.Фаулер. – М.: Издательский дом «Вильяме», 2010. – 544 с.

8 Баас Л. Архитектура программного обеспечения на практике / Л. Баас. – СПб.: Питер, 2006. – 576 с.

9 Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма. – СПб.: Питер, 2016. – 366 с.

10 Мартин Р. Читый код: создание, анализ и рефракторинг / Р. Мартин. – СПб.: Питер, 2016. – 464 с.

11 Alex, E. Willis Web Development with Microsoft Exchange 2000 Server / Alex E. Willis, Paul R. Bebelos, Robert A. Saccone. – USA: Wiley, 2015. – 351 с.

12 Christian, Thilmany .NET Patterns: Architecture, Design, and Process / Christian Thilmany. – Москва: Мир, 2017. - 452 с.

13 Colin, Moock Action Script for Flash Mx Pocket Reference (Pocket Reference (O'Reilly)) / Colin Moock. - M.: O'Reilly Media, 2016. - 144 с.

14 Diane, Singerman Development, Change, and Gender in Cairo: A View from the Household (Indiana Series in Arab and Islamic Studies) / Diane Singerman, Homa Hoodfar. - India: Indiana University Press, 2015. - 965 с.

15 Douglas, E. Comer Internetworking with TCP/IP Vol. III Client-Server Programming and Applications-Windows Sockets Version / Douglas E. Comer, David L. Stevens. - M.: Prentice Hall, 2015. - 512 с.

16 Ed, Tittel Windows Server 2003 for Dummies / Ed Tittel, James Michael Stewart. - Москва: Гостехиздат, 2017. - 156 с.

17 Edward, P. Lazear Allocation of Income Within the Household / Edward P. Lazear, Robert T. Michael. - Москва: Машиностроение, 2017. - 668 с.

18 Gary, Olsen Windows Server 2003 on HP ProLiant Servers / Gary Olsen, Bruce Howard. - M.: Prentice Hall Ptr, 2014. - 282 с.

19 Hollosi Integrating PHP with Windows / Hollosi. - Москва: Огни, 2014. - 841 с.

20 Kalani, Kirk Hausman MCSA/MCSE Windows Server 2003 Upgrade Exams Bundle Exam Cram 2 (Exam Cram 2) / Kalani Kirk Hausman, Will Schmied. - Москва: СИНТЕГ, 2018. - 152 с.

21 Linda, F. Radke Household Careers: Nannies, Butlers, Maids & More : The Complete Guide for Finding Household Employment or "If the Dog Likes You, You're Hired!" / Linda F. Radke. - Москва: Машиностроение, 2017. - 176 с.

22 Matthew, Lockwood Fertility and Household Labour in Tanzania: Demography, Economy, and Society in Rufiji District, C.1870-1986 / Matthew Lockwood. - Москва: РГГУ, 2015. - 203 с.

23 Michael, Donahoo TCP/IP Sockets in C: Practical Guide for Programmers (The Practical Guides Series) / Michael Donahoo, Kenneth Calvert. - Москва: Огни, 2016. - 392 с.

24 Nevada, Learning Series Microsoft Outlook 2000 Quick Reference Guide with Exchange 5.5 / Nevada Learning Series. - Москва: ИЛ, 2016. - 421 с.

25 Paul, Sanna Windows 2000 Server Security for Dummies / Paul Sanna. - Москва: Наука, 2017. - 618 с.

26 Phil, Syme Sams TY C# Web Programming in 21 Days / Phil Syme, Peter Aitken. - Москва: Гостехиздат, 2018. - 648 с.

27 Richard, Carlson Don't Sweat the Small Stuff with Your Family : Simple Ways to Keep Daily Responsibilities and Household Chaos from Taking Over Your Life (Don't Sweat the Small Stuff Series) / Richard Carlson. - Москва: РГГУ, 2018. - 366 с.

28 Roseann, Cares MCSA/MCSE Windows 2000 Server Package (70-215) / Roseann Cares, Brian Alley, Charles J. Brooks. - Москва: СИНТЕГ, 2018. - 645 с.

29 Бенкен, Е.С. PHP, MySQL, XML. Программирование для Интернета (+ CD-ROM) / Е.С. Бенкен. - М.: БХВ-Петербург, 2017. - 400 с.

30 Девис, Е.М. Изучаем PHP и MySQL / Е.М. Девис. - М.: Символ-плюс, 2016. - 839 с.

31 Дунаев, В. Сценарии для Web-сайта. PHP и JavaScript / В. Дунаев. - М.: БХВ-Петербург, 2017. - 576 с.

32 Иван, Портянкин Swing. Эффектные пользовательские интерфейсы / Портянкин Иван. - М.: ЛОРИ, 2016. - 541 с.

33 Клифтон, Б. Google Analytics для профессионалов / Б. Клифтон. - М.: Диалектика / Вильямс, 2017. - 740 с.

34 Ленгсторф, Джейсон PHP и jQuery для профессионалов / Джейсон Ленгсторф. - М.: Вильямс, 2015. - 362 с.

35 Моримото, Рэнд Microsoft Exchange Server 2007. Полное руководство / Рэнд Моримото и др. - М.: Вильямс, 2014. - 884 с.

36 Ник, Рендольф Visual Studio 2010 для профессионалов / Рендольф Ник. - М.: Диалектика / Вильямс, 2016. - 356 с.

37 Рассел, Джесси Сергей Солоух / Джесси Рассел. - М.: VSD, 2017. - 444 с.

38 Скляр, Дэвид PHP. Рецепты программирования / Дэвид Скляр , Адам Трахтенберг. - М.: Питер, 2015. - 784 с.

39 Фримен, Эрик Паттерны проектирования / Эрик Фримен и др. - М.: Питер, 2015. - 656 с.

40 Яковлев, А. Раскрутка сайтов. Основы, секреты, трюки / А. Яковлев. - М.: БХВ-Петербург, 2015. - 518 с.