Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и Информационных Технологий
институт
Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ Л.С. Троценко
подпись инициалы, фамилия
«13»___июня____ 2018 г.

# БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии

Разработка веб-сервиса для уменьшения денежных затрат на приобретение игрового контента

Руководитель _____ С.А. Виденин
подпись, дата инициалы, фамилия

Выпускник _____ В.В. Акулов
подпись, дата инициалы, фамилия

Нормоконтролер _____ Ю.В. Шмагрис
подпись, дата инициалы, фамилия

Красноярск 2018

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и Информационных Технологий
институт
Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ С.А. Виденин
подпись инициалы, фамилия
«_05_» __03__ 2018г.

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

**в форме бакалаврской работы**

Студенту _____Акулову Вячеславу Вадимовичу_____

фамилия, имя, отчество

Группа КИ14-13Б Направление 09.03.02_____

номер код

_____Информационные системы и технологии_____

наименование

Тема выпускной квалификационной работы: Разработка веб-сервиса для уменьшения денежных затрат на приобретение игрового контента

Утверждена приказом по университету № 4896/с от 05.04.2018г.

Руководитель ВКР С. А. Виденин, кандат пед. наук, доцент, заведующий кафедрой «Информационные системы» ИКИТ _____

инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР: Требования к разрабатываемому веб-сервису, рекомендации руководителя

Перечень разделов ВКР: Введение, обзор предметной области и выбранных технологий для создания веб-сервиса, проектирование веб-сервиса, разработка веб-сервиса, демонстрация функционала веб-сервиса, заключение, список использованных источников, приложение А

Перечень графического материала: Презентация, выполненная в Microsoft Office PowerPoint 2013

Руководитель ВКР _____ С. А. Виденин

подпись инициалы и фамилия

Задание принял к исполнению _____ В. В. Акулов

подпись инициалы и фамилия

« 05 » 03 2018г.

# РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка веб-сервиса для уменьшения денежных затрат на приобретение игрового контента» содержит 50 страницы текстового документа, 28 иллюстраций, 20 использованных источников.

ВЕБ-СЕРВИС, LARAVEL, MVC, PHP, HTML, JS_QUERY, NODE_JS, BOOTSTRAP, БАЗА ДАННЫХ, MYSQL.

Целью работы — веб-сервис для уменьшения денежных затрат на приобретение внутриигрового контента.

Для достижения поставленной цели были определенны следующие задачи:

– изучение и выбор инструментов для реализации веб-сервиса;

– проектирование веб-сервиса;

– разработка веб-сервиса.

В результате выполнения выпускной квалификационной работы был создан веб-сервис, позволяющий уменьшить денежные затраты на приобретение внутриигрового контента.

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ ..... 4

1 Обзор предметной области и выбранных технологий для создания веб-сервиса ..... 5

1.1 Краткий обзор предметной области ..... 5

1.2 Выбранные технологии для реализации веб-сервиса ..... 6

2 Проектирование веб-сервиса ..... 14

2.1 Функциональная модель ..... 14

2.2 Проектирование базы данных ..... 20

3 Разработка веб-сервиса ..... 22

4 Демонстрация функционала веб-сервиса ..... 33

ЗАКЛЮЧЕНИЕ ..... 37

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ ..... 38

ПРИЛОЖЕНИЕ А ..... 40

# ВВЕДЕНИЕ

На сегодняшний день очень популярны многопользовательские онлайн игры, одной из таких является Counter Strike: Global Offensive. Игроки CS:GO очень часто приобретают внутриигровой контент. В настоящее время в интернете можно увидеть множество сервисов, которые позволяют пользователям получать предметы отличным от основного способа, которые задумывали разработчики данной игры.

Основываясь на анализе проведенным Бобиным Максимом в рамках ВКР, был сделан вывод, что существующие веб-сервисы не в полной мере удовлетворяют потребность пользователей, получать игровой предмет за меньшие денежные затраты, чем предлагает производитель игры.

Исходя из вышеописанной проблемы целью выпускной квалификационной работы является: веб-сервис для уменьшения денежных затрат на приобретение внутриигрового контента.

Для достижения поставленной цели были определенны следующие задачи:

– изучение и выбор инструментов для реализации веб-сервиса;

– проектирование веб-сервиса;

– разработка веб-сервиса.

# 1 Обзор предметной области и выбранных технологий для создания веб-сервиса

## 1.1 Краткий обзор предметной области

В настоящее время большинство игроков в онлайн игры стремятся превосходить других игроков не только в умение играть, но и в качестве игровых предметов. Пользователи готовы тратить суммы на покупку внутриигровых предметов неоднократно превышающие цену на саму игру, например, стоимость одного из самых дорогих предметов можно увидеть на рисунке 1.

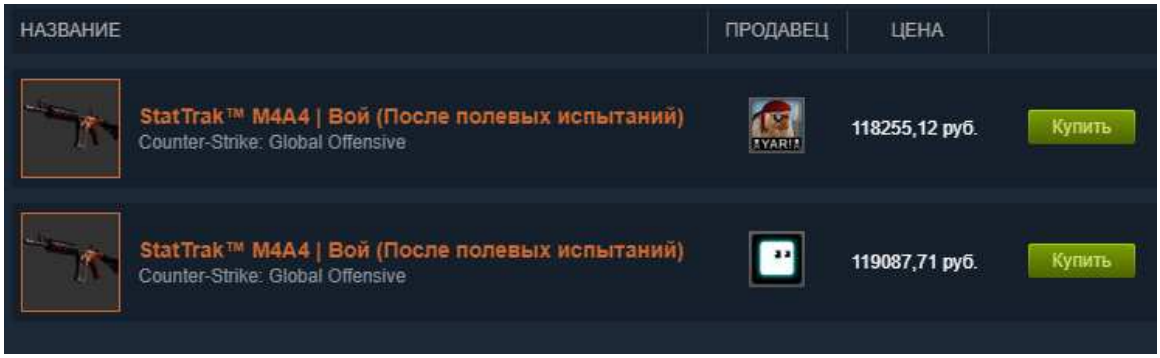


Рисунок 1 — Игровой предмет

Благодаря этому начали активно создаваться веб-сервисы, позволяющие пользователям уменьшить затраты на приобретение внутриигровых предметов.

По результатам обзора аналогичных веб-сервисов для онлайн игры Counter-Strike: Global Offensive Бобиным Максимом в рамках ВКР был сделан вывод, что данные веб-сервисы работают по принципу, проиграть больше шансов, чем выиграть, так как выигрыш полностью зависит от размера депозита. Данный факт и побудил создать веб-сервис, работающий по принципу, у всех пользователей одинаковый шанс на победу и не зависит от суммы депозита.

## 1.2 Выбранные технологии для реализации веб-сервиса

Для реализации данного проекта были выбраны следующие технологии: языки программирования PHP, JavaScript, framework Laravel, Bootstrap, веб-сервер Apache, база данных MySQL, а также программная платформа NodeJS.

Для удобства и быстроты современное сообщество используют готовый набор серверного программного обеспечения – LAMP, включающий в себя следующие компоненты:

– Linux – операционная система Linux;

– Apache – веб-сервер;

– MySQL – СУБД;

– PHP – язык программирования.

Изначально перечисленные программные продукты не разрабатывались специально для работы друг с другом, лишь в последствие данная комбинация стала весьма популярна в веб-разработке.

Apache — полнофункциональный, расширяемый веб-сервер с открытым кодом.

На сегодняшний день существуют огромное количество веб-серверов. Apache является самым популярным среди них, примерно на 70% серверов мира он используется.

Плюсы:

- Возможность простой аутентификации;
- Поддержка технологии SSI;
- Возможность создания на сервере пользовательских директорий;
- Возможность настройки виртуальных серверов;
- Работа с различными скриптами;
- Безопасность;
- Ведение журнала событий на сервере.

MySQL — это самая распространенная полноценная серверная СУБД. MySQL очень функциональная, свободно распространяемая СУБД, которая успешно работает с различными сайтами и веб приложениями.

Плюсы:

- Простота и легкость освоения.
- Большое сообщество пользователей и разработчиков.
- Богатый функционал.
- Скорость.
- Безопасность
- Масштабируемость.

PHP — известный язык программирования, с открытым исходным кодом, интенсивно применяющийся в веб-среде. Был разработан для создания скриптов работающих на сервере, так же обрабатывать данные html-форм динамически генерировать html-страницы и выполнения работ с различными базами данных.

Плюсы:

- Простота и легкость освоения.
- Работа с различными базами данных.
- Работа с различными веб-серверами.
- Большое сообщество пользователей и разработчиков.
- Имеется огромное количество библиотек и расширений языка.

Для удобства и быстроты разработки было решено использовать php-фреймворк.

На сегодняшний день создано очень большое количество фреймворков базирующиеся на php. Но в данной работе мы рассмотрим два laravel, codeigniter. Laravel — бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC. Пакеты — позволяют создавать и подключать модули в формате Composer к приложению на Laravel. Eloquent ORM — реализация шаблона проектирования ActiveRecord на PHP.

Позволяет строго определить отношения между объектами базы данных. Стандартный для Laravel построитель запросов Fluent поддерживается ядром Eloquent.

Логика приложения — часть разрабатываемого приложения, объявленная либо при помощи контроллеров, либо маршрутов (функций-замыканий). Синтаксис объявлений похож на синтаксис, используемый в каркасе Sinatra.

Обратная маршрутизация связывает между собой генерируемые приложением ссылки и маршруты, позволяя изменять последние с автоматическим обновлением связанных ссылок.

REST-контроллеры — дополнительный слой для разделения логики обработки GET- и POST-запросов HTTP.

Автозагрузка классов — механизм автоматической загрузки классов PHP без необходимости подключать файлы их определений в include. Загрузка по требованию предотвращает загрузку ненужных компонентов; загружаются только те из них, которые действительно используются.

Составители представлений — блоки кода, которые выполняются при генерации представления (шаблона).

Инверсия управления — позволяет получать экземпляры объектов по принципу обратного управления. Также может использоваться для создания и получения объектов-одиночек.

Миграции — система управления версиями для баз данных. Позволяет связывать изменения в коде приложения с изменениями, которые требуется внести в структуру БД, что упрощает развёртывание и обновление приложения.

Модульное тестирование (юнит-тесты) — играет очень большую роль в Laravel, который сам по себе содержит большое число тестов для предотвращения регрессий (ошибок вследствие обновления кода или исправления других ошибок).

Страничный вывод — упрощает генерацию страниц, заменяя различные способы решения этой задачи единым механизмом, встроенным в Laravel.

CodeIgniter — популярный MVC фреймворк с открытым исходным кодом, написанный на языке программирования PHP, для разработки полноценных веб-систем и приложений.

Поддержка баз данных MySQL, PostgreSQL, MSSQL, SQLite, Oracle.

Поддержка псевдо-ActiveRecord, который по большей части повторяет синтаксис языка SQL

Легко расширяемая система за счет возможности использования сторонних и самописных библиотек, а также дополнения или переопределения существующих.

Поддержка как сегментированных ЧПУ, так и обычных URL-ов с передачей параметров.

Фреймворк содержит в себе множество необходимых библиотек, которые создают функциональность для работы с файлами, отправки электронных писем, валидации форм, поддержки сессий, работы с изображениями и так далее.

Основываясь на вышенаписанном и мнении комьюнити был сделан вывод, что Laravel обладает большей гибкостью, расширяемостью, безопасностью и лучшей производительностью. Исходя из этих плюсов было решено использовать в рамках нашего проекта framework laravel.

Выбор framework laravel в свою очередь определил архитектуру проектирования model-view-controller.

Основная цель данного паттерна заключается в разделение приложения на три уровня: модель данных (model), представление (view) и контроллер (controller). Таким образом, изменения, вносимые в один из компонентов, оказывают минимально возможное воздействие на другие компоненты или не оказывают вовсе. Рассмотрим этот паттерн подробнее.

Model (модель) – это часть, в которой содержится бизнес-логика приложения. Она никак не зависит от остальных частей. Модельный слой ничего не должен знать об элементах дизайна (View), и каким образом он будет отображаться.

View (вид, представление) – данная часть отвечает за визуализацию данных, полученных из контроллера и модели. так, например, одни и те же данные могут представляться разными способами. Она содержит HTML-разметку и небольшие вставки PHP-кода.

Controller (контроллер) — часть, связывающая модели, виды и друге компоненты в работающие приложение. То есть обрабатывает поступающие запросы, выполняет операции с моделью и выбирает представления для визуализации пользователю.

На рисунке 2 показано взаимодействие между уровнями приложения.


Рисунок 2 — Model-view-controller

Так как в проекте должна была присутствовать интерактивность встал вопрос использовать чистый JavaScript или библиотеку jQuery, для понимания jQuery является библиотекой JavaScript. Основные отличительные свойства представлены в таблице 1

Таблица 1 – JS vs jQuery

| JavaScrip | jQuery |
|---|---|
| Слабо типизированный язык динамического программирования | Быстрая и сжатая библиотека JavaScript |
| Язык сценариев взаимодействия интерфейса и управления содержимым документа | Рамка, которая упрощает и ускоряет обработку событий, анимацию и Ajax |
| Интерпретируемый язык | Использует ресурсы, предоставленные JavaScript, чтобы упростить работу |
| Нужно писать собственные сценарии, которые могут занять время | не нужно писать много скриптов, которые уже существуют в jQuery |
| Разработчики должны обрабатывать совместимость с несколькими браузерами, написав собственный код JavaScript | Является многосерверной библиотекой JavaScript, которая уменьшает работу разработчиков во время развертывания |
| Разработчики склонны делать много распространенных ошибок, связанных с браузером. | Разработчикам не нужно беспокоиться о проблемах совместимости браузеров. |
| Не нужно ничего включать в браузер, поскольку все современные браузеры поддерживают JS | Необходимо включить URL-адрес библиотеки jQuery в заголовок страницы |
| Больше строк кода | Меньше строк кода |
| Быстрее в доступе к DOM | Подходит для сложных операций, в которых разработчики склонны к ошибкам и написанию плохих строк кода. |

Таким образом выбор пал на jQuery, т.к хорошо подходит для большинства приложений, которые требуют быстроты действий. jQuery заботится об общих ошибках браузера, а также заботится о проблеме совместимости браузера с программным продуктом.

Чтобы веб-сервис обладал адаптивностью было решено при разработке front-end использовать framework bootstrap

Framework bootstrap это простой и легко настраиваемый HTML, CSS и Javascript фреймворк для более быстрой и удобной разработки.

Включает в себя HTML и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Основные инструменты:

Сетки - это заранее заданные размеры колонок, которые можно сразу применить. Сетка включает в себя 12 колонок для различных девайсов, что позволяет веб-странице быть масштабируемой, в результате чего создаётся адаптивный дизайн веб-приложения.

Шаблоны - это фиксированный или резиновый шаблон документа.

Формы - это классы для оформления форм и некоторых событий, происходящих с ними.

Медиа - представляет некоторое управление изображениями и видео.

Таблицы - это средства оформления таблиц, вплоть до добавления функциональности сортировки.

Типографика - это описание шрифтов, определение некоторых классов для шрифтов.

Навигация явлется классами оформления для табов, вкладок, страничности (пагинации), меню и панели инструментов.

Алерты – это оформление подсказок, диалоговых и всплывающих окон.

Все эти инструменты помогают настроить Bootstrap индивидуально под каждый проект, веб-сайт или веб-приложение.

Так же в рамках проекта нужно было реализовать программу, имитирующую деятельность человека. Так как, одной из основных функций было взаимодействие с API steam. При рассмотрение данной задачи было выявлено, что данную программу можно реализовать двумя способами, использовав python или NodeJS.

Но так как в нашем проекте за основу был взят framework Laravel, написанный на php, то использовать python не рационально. Поэтому выбор пал на NodeJS.

Node.js — программная платформа, на основе движка JavaScript Chrome V8, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. 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. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения.

# 2 Проектирование веб-сервиса

Чтобы упростить этап разработки было решено изначально спроектировать функциональную модель SADT и спроектировать базу данных.

## 2.1 Функциональная модель

Методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели системы какой-либо предметной области. Функциональная модель SADT отображает структуру процессов функционирования системы и ее отдельных подсистем, т. е. выполняемые ими действия и связи между этими действиями. Для этой цели строятся специальные модели, которые позволяют в наглядной форме представить последовательность определенных действий.

Исходными строительными блоками любой модели IDEF0 процесса являются деятельность (activity) и стрелки (arrows).

Для наглядности, и чтобы в дальнейшем не возникло проблем с разработкой было решено сначала построить SADT-диаграммы.

Контекстная диаграмма A0: специальный вид (контекстной) диаграммы IDEF0, состоящей из одного блока, описывающего функцию верхнего уровня, ее входы, выходы, управления, и механизм мы, вместе с формулировками цели модели и точки зрения, с которой строится модель.

На рисунке 3 показана диаграмма, описывающая функцию получения предмета.

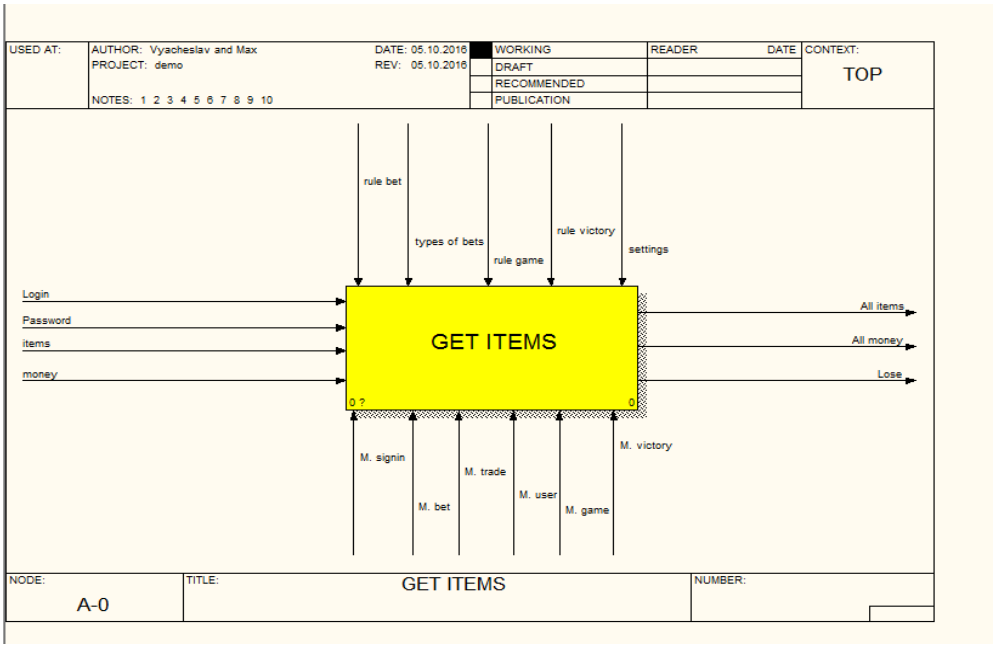


Рисунок 3 — Система получение предмета

На рисунке 4 представлена проведенная декомпозиция основной функции.

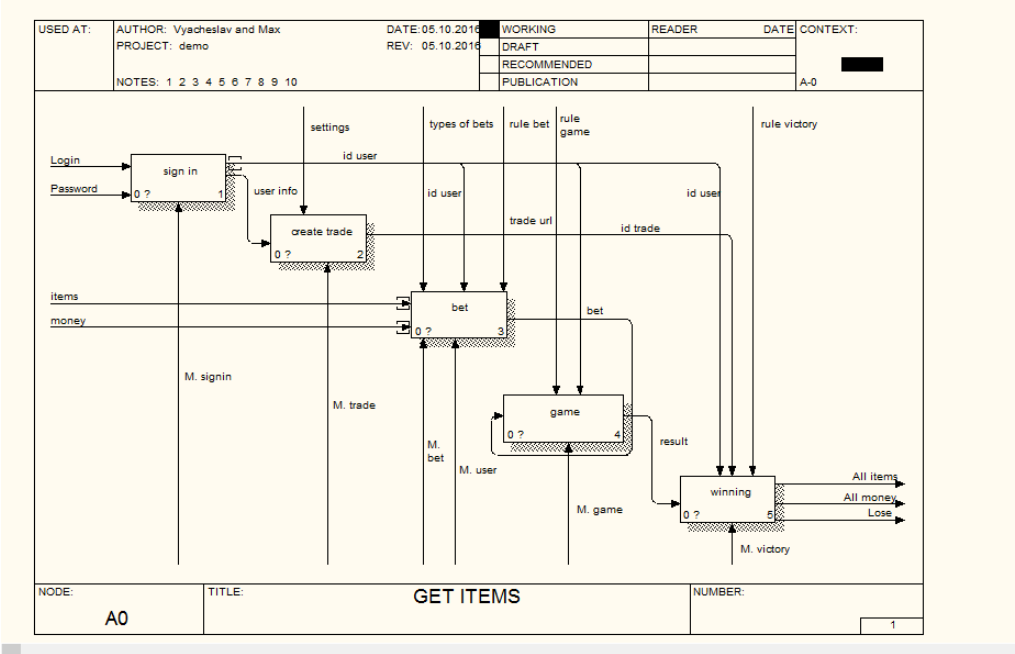


Рисунок 4 — Декомпозиция системы получение предмета

Система получения предмета состоит из следующих подсистем:

– sign in;

– create trade;

– bet;

– game;

– winning.

Диаграмма A1 – Авторизация, показана на рисунке 10.

A11(Авторизация через steam): пользователь через м.user переходит на форму авторизации через steam.

A12(Авторизация): Ввод логина и пароля.

A13(Подтверждение авторизации): Подтверждение, используя SteamGuard.

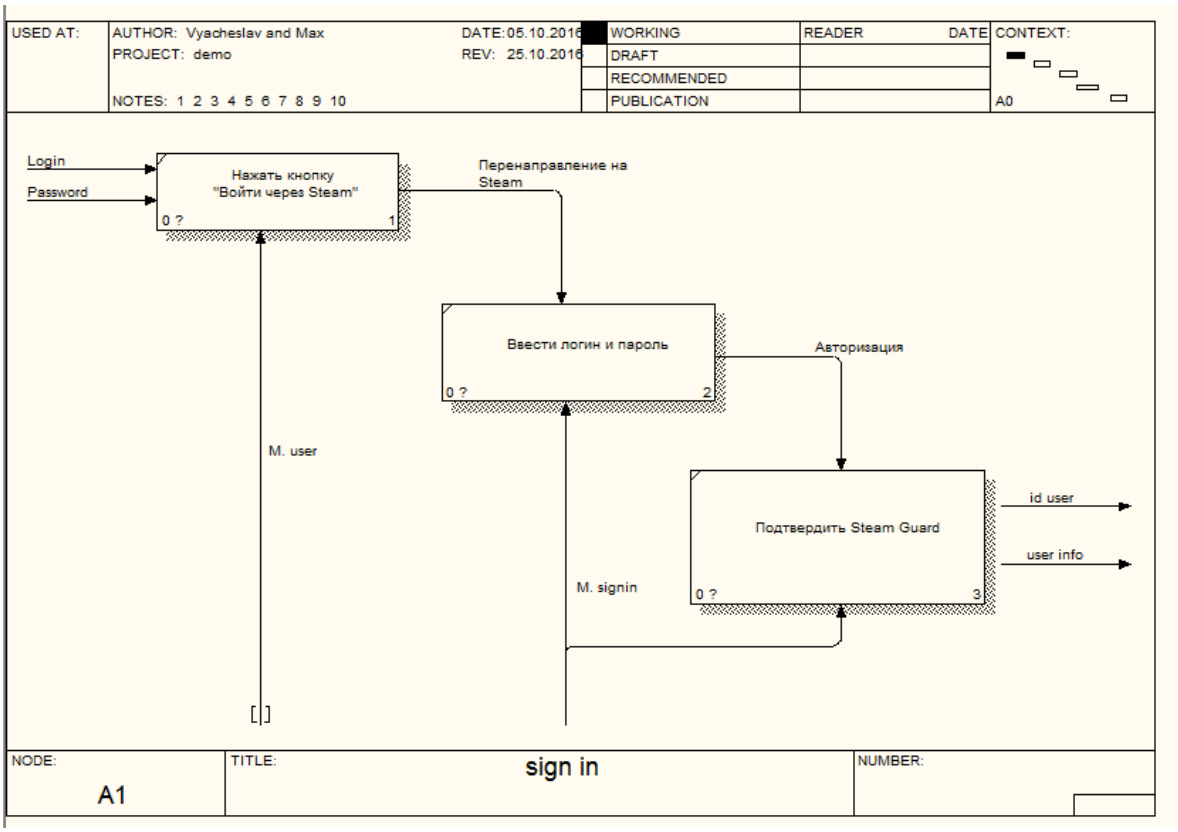


Рисунок 5 — Подсистема авторизации

Диаграмма A3 – Create trade, показана на рисунке 11.

Пользователь должен перейти во вкладку настройка профиля, после, используя подсказки, добавить trade ссылку и нажать на кнопку сохранить изменения.

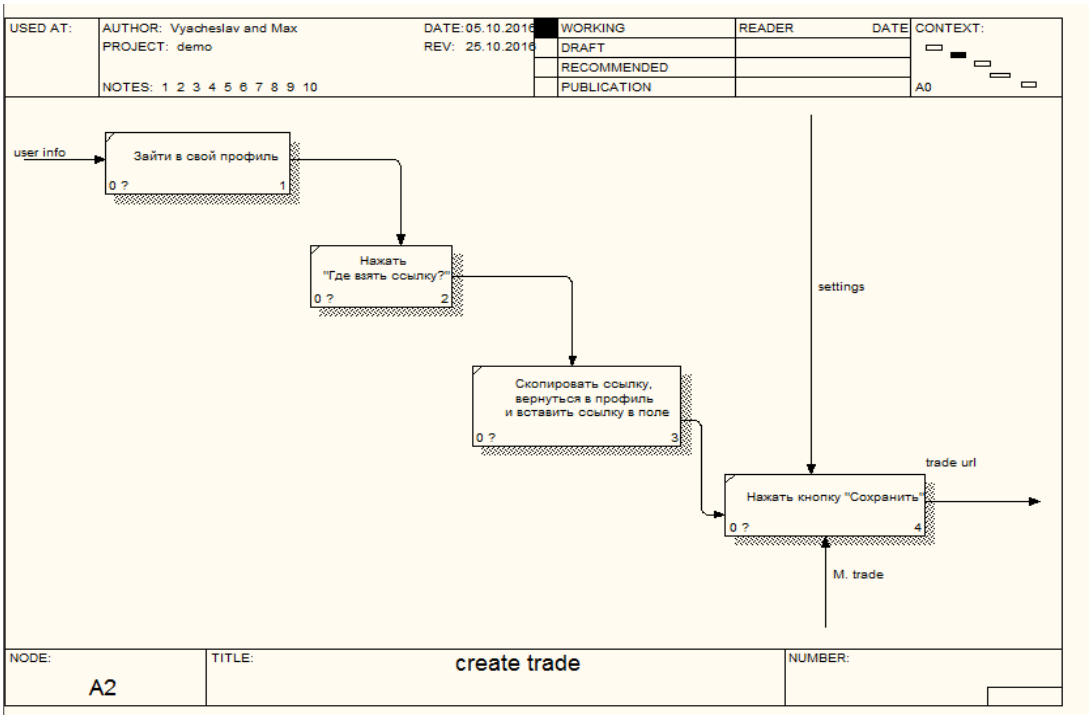


Рисунок 6 — Подсистема настройки аккаунта

Диаграмма A4 – Bet, показана на рисунке 7.

A41(выбор способа ставки): пользователь через м.user выбирает интересующий его способ ставки.

A42(выбор способа оплаты): пользователь через м.user вносит ставку.

A43(Подтверждение ставки): пользователь через м.bet подтверждает ставку.

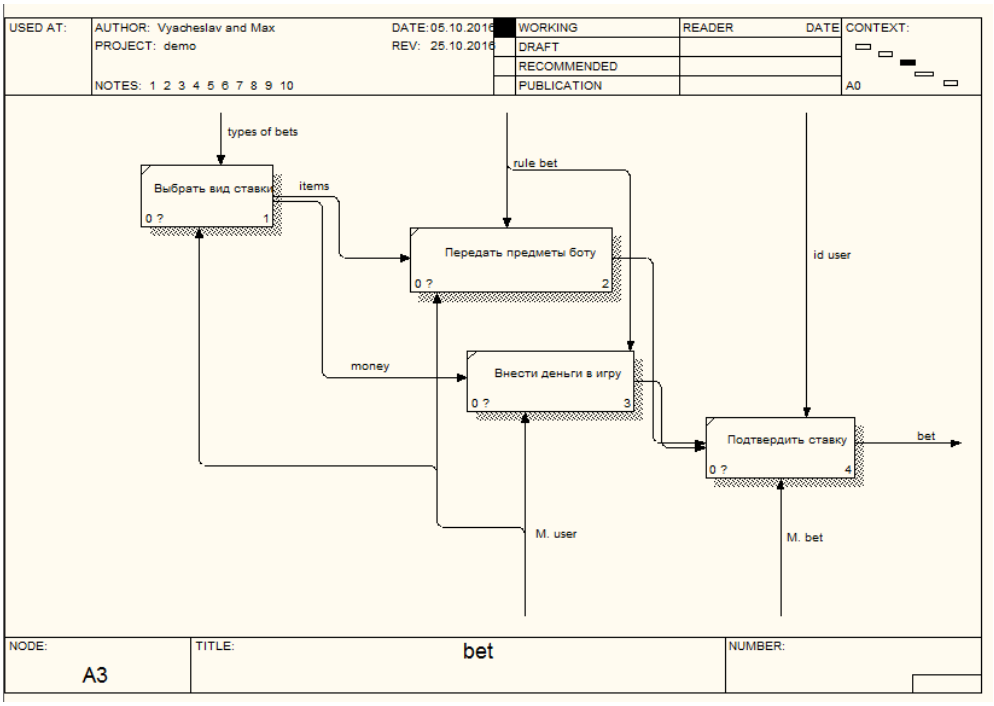


Рисунок 7 — Подсистема депозита

Диаграмма A5 –Game, показана на рисунке 8.

A51(принятие ставки)

A52(подсчет кол-ва игроков)

A53(запуск игры)

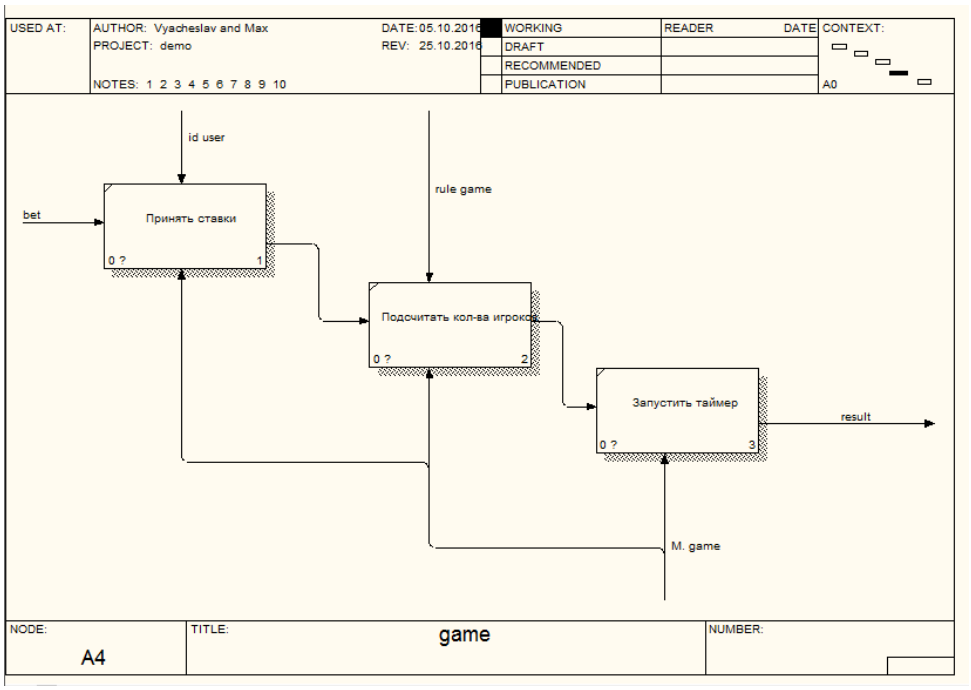


Рисунок 8— Подсистема игры

Диаграмма A6 –Game, показана на рисунке 9.

A61(результат игры)

A62(получение выигрыша)

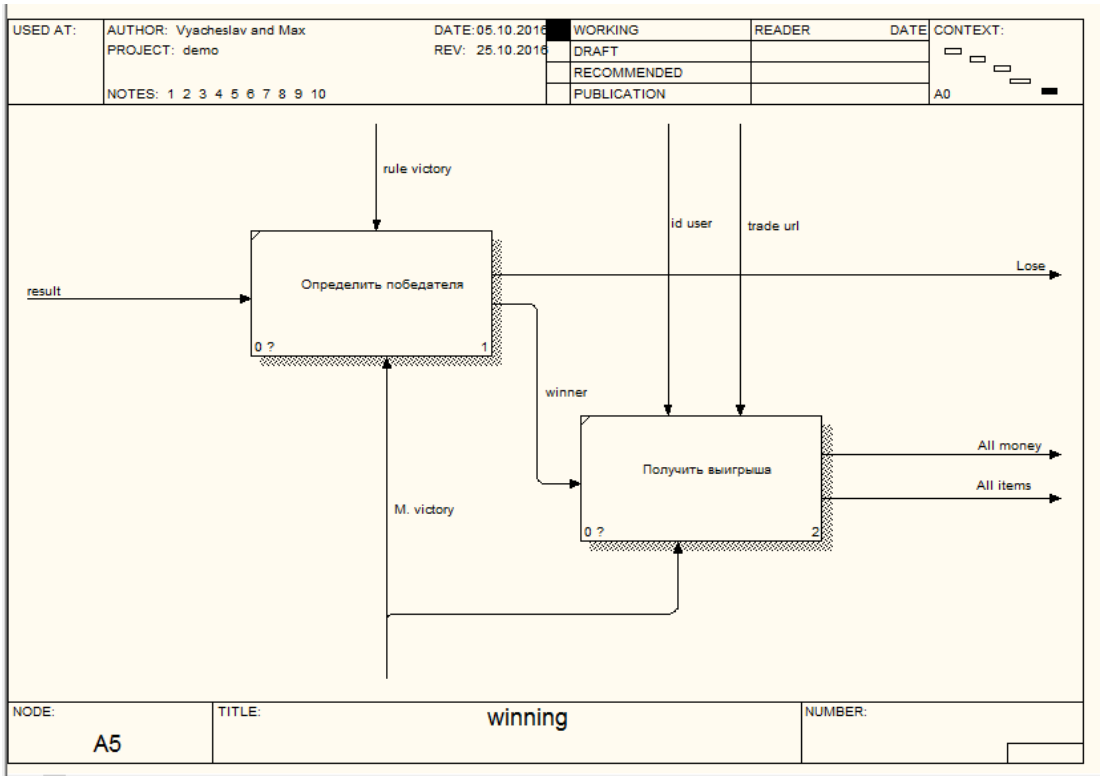


Рисунок 9 — Подсистема определение победителя

## 2.2 Проектирование базы данных

Перед созданием сайта необходимо разработать структуру будущей базы данных. Используя программный продукт ErWinDataModeller, была спроектирована схема данных веб-сервиса.

Реляционная база данных – база данных, построенная на основе реляционной модели. В реляционной базе каждый объект задается записью (строкой) в таблице. Реляционная база создается и затем управляется с помощью реляционной системы управления базами данных. Фактически реляционная база данных это тело связанной информации, сохраняемой в двухмерных таблицах. Связь между таблицами может находить свое отражение в структуре данных, а может только подразумеваться, то есть присутствовать на неформализованном уровне. Каждая таблица БД представляется как совокупность строк и столбцов, где строки соответствуют экземпляру объекта, конкретному событию или явлению, а столбцы - атрибутам (признакам, характеристикам, параметрам) объекта, события, явления.

Реляционные базы данных предоставляют более простой доступ к оперативно составляемым отчетам (обычно через SQL) и обеспечивают повышенную надежность и целостность данных благодаря отсутствию избыточной информации.

На рисунке 10 представлена структура базы данных.


Рисунок 10 — Схема данных

В таблице «withdraws» хранятся основные характеристики для вывода предмета.

В таблице «bots» хранится информация для бота.

В таблице «double» хранится подробная информация об игре.

В таблице «settings» хранятся настройки веб-сайта.

В таблице «user» хранится информация о пользователях.

В таблице «payments» храниться информация для внесения депозита.

## 3 Разработка веб-сервиса

Для серверной части на понадобится установить веб-сервер, который имеет все необходимое для работы с фреймворком. Устанавливаем менеджер зависимостей для PHP Composer, который достаточно просто установить для Windows. Далее с помощью командной строки устанавливаем Larvael командой composer global require "laravel/installer" и создаем проект laravel new reg. На рисунке 11 представлена структура проекта.

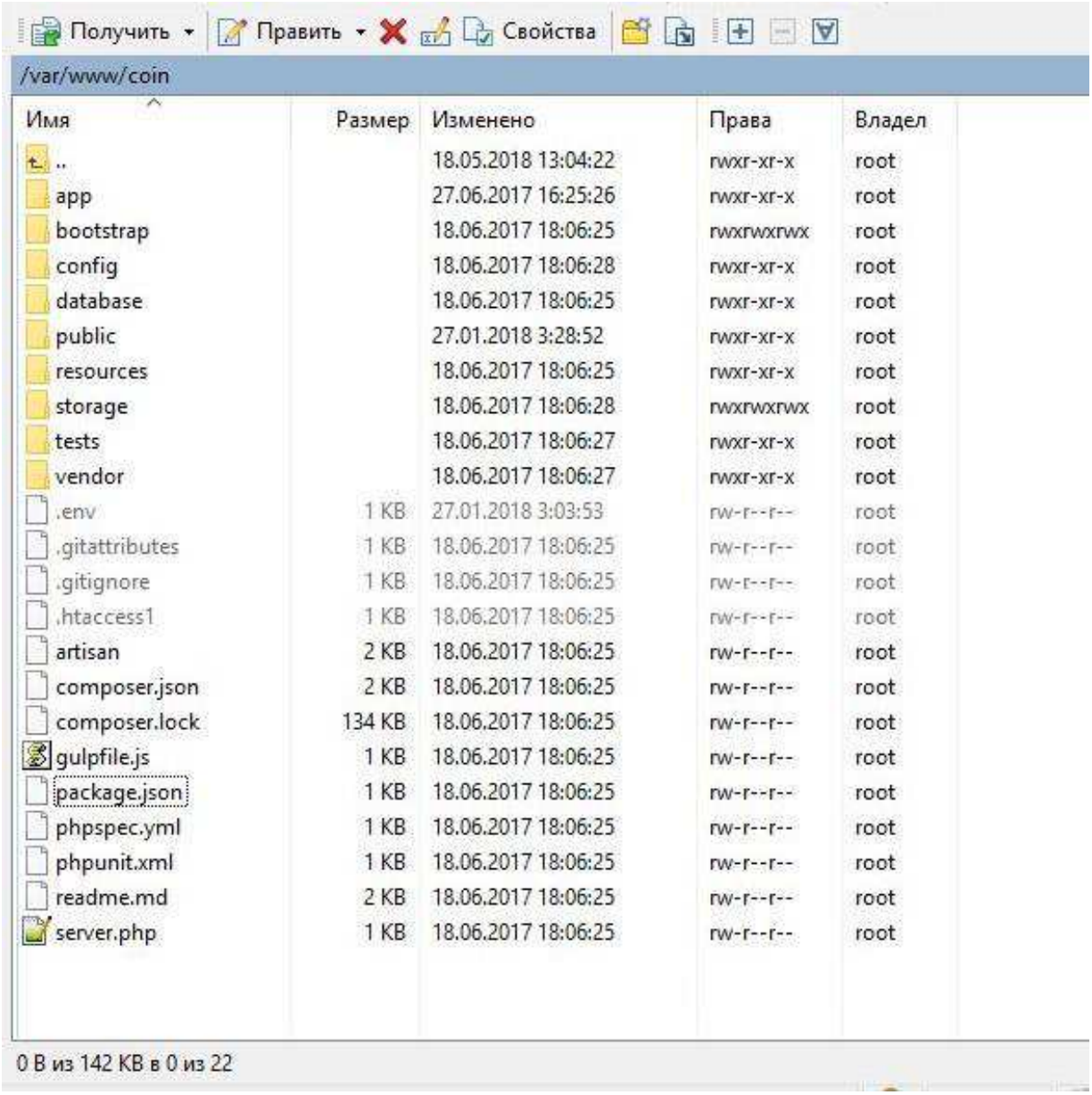


Рисунок 11 — Структура проекта

Точка входа, иначе говоря, index.php, располагается в каталоге public. По задумке разработчиков фреймворка в каталоге public хранятся файлы, к которым разрешен публичный доступ. Для сохранения структуры проекта прописываем в файле .htaccess правила RewriteEngine On, RewriteRule ^$ public/ [L], RewriteRule (.*) public/$1 [L], то есть все запросы перенаправляются в каталог public.

В каталоге app располагаются основная логика приложения, например, модели, контроллеры и так далее. Непосредственно в корне каталога app располагаются модели user.php (создана по умолчанию). В каталоге app/http/controllers хранятся контроллеры. На рисунке 12 представлена основная логика приложения.

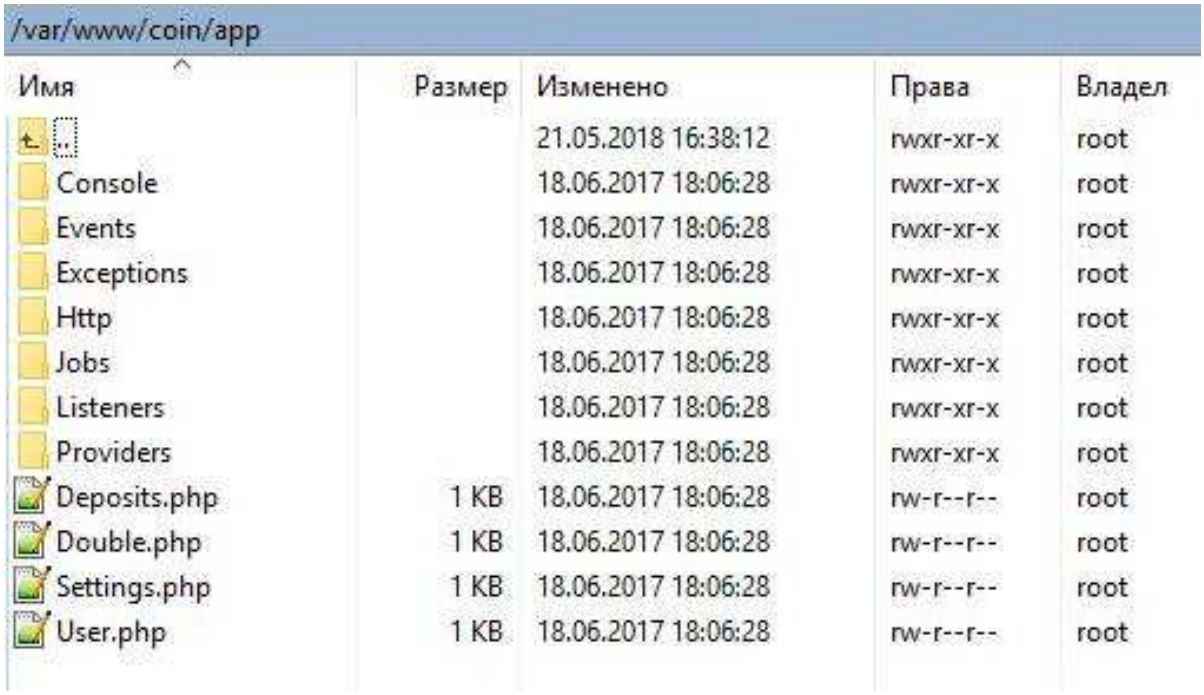


Рисунок 12 — Основная логика приложения

В каталоге resources/view находятся виды, которые переведены в php с использование шаблонизатора Blade. Все шаблоны компилируются в PHP код и кешируются до следующих изменений, что означает — Blade не добавляет накладных расходов в приложение. Два основных преимущества Blade: наследование и секции. Основные view представлены на рисунке 13.

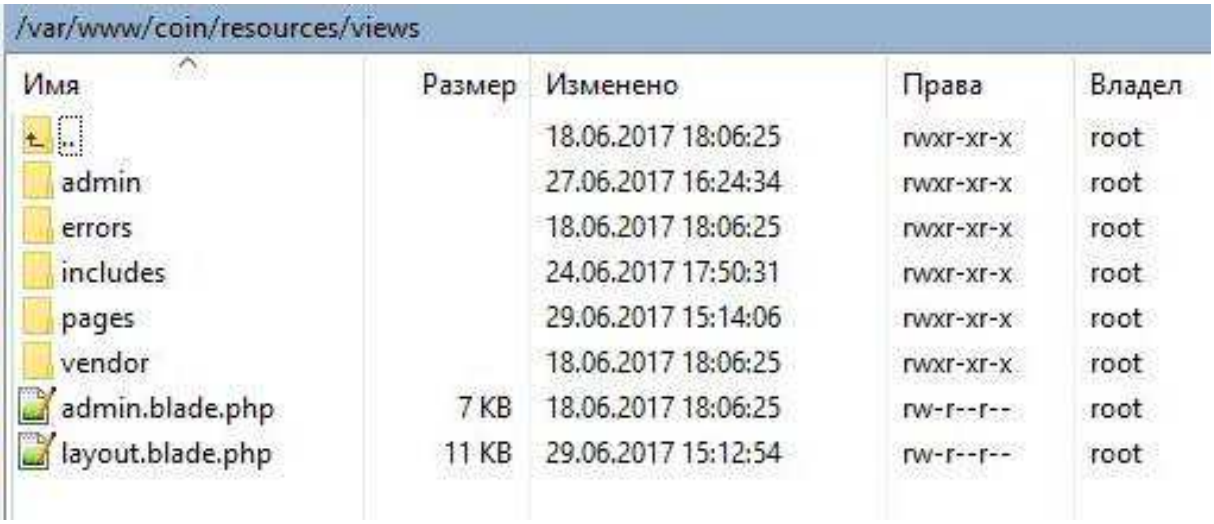


Рисунок 13 — Views

На рисунке 14 показаны контроллеры, отвечающие за логику веб-сервиса.

Основные из них, DepositController, ChatController, DoubleController, AuthController, ShopController.

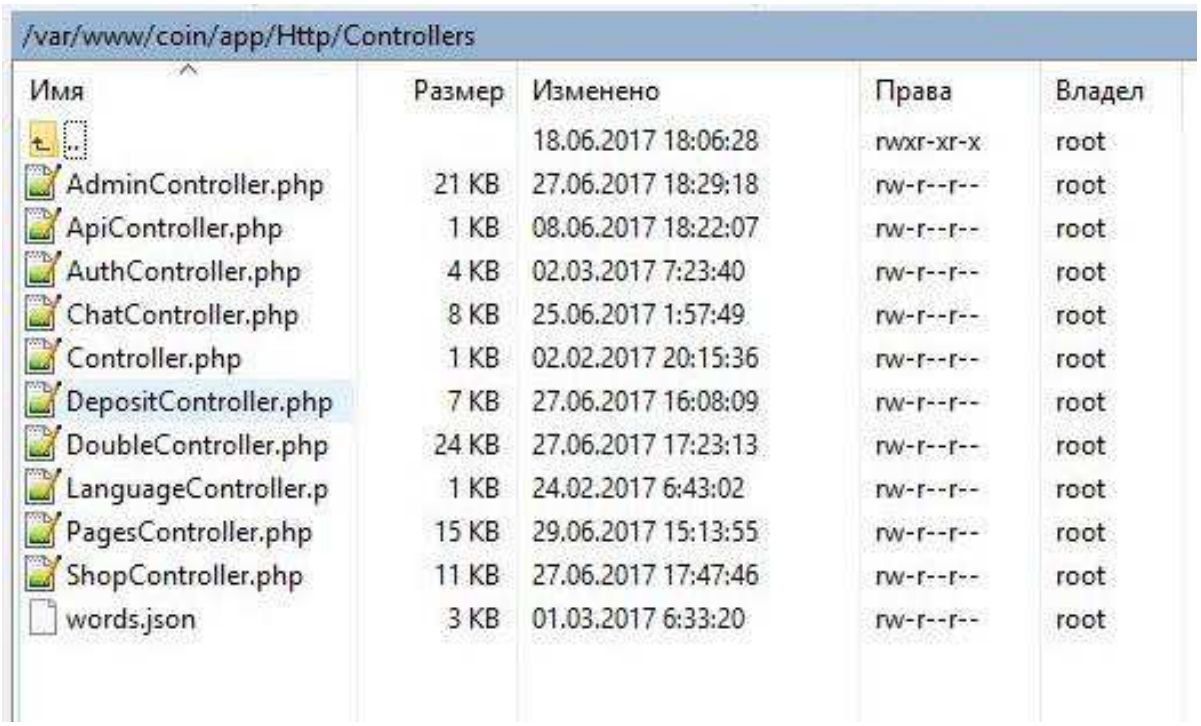


Рисунок 14 — Контроллеры

Логика контроллера, отвечающего за внесение депозита показана на рисунке 15.

```
class DepositController extends Controller

    public function __construct()
    {
        $this->redis = Redis::connection();
        $this->config = Settings::first();
        if(Auth::check()) {
            $this->user = Auth::user();
            view()->share('u', $this->user);
        }
        $this->prices = json_decode(File::get('prices.txt'), true);
        view()->share('config', $this->config);
    }

    public function index()
    {
        view()->share('title', 'Deposit');
        return view('pages.refill');
    }

    public function parseItems()
    {
        if(Auth::guest()) return;

        $res = json_decode(file_get_contents('http://steamcommunity.com/inventory/'.$this->user->steamid64.'/730/2'));
        if(!isset($res->assets)) return;

        $items = [];
        foreach($res->assets as $item) {
            $items[] = [
                'classid'       => $item->classid,
                'instanceid'    => $item->instanceid,
                'assetid'       => $item->assetid
            ];
        }

        foreach($items as $i => $item) {
            $found = false;
            foreach($res->descriptions as $data) {
                if((!$found) && ($data->classid == $item['classid']) && ($data->instanceid == $item['instanceid'])) {
                    $found = true;
                    $items[$i]['market_hash_name'] = $data->market_hash_name;
                }
            }
        }

        $list = [];
        foreach($items as $i => $item) {
            if(isset($this->prices[$item['market_hash_name']])) {
                $items[$i]['price'] = floor($this->prices[$item['market_hash_name']]*$this->config->curs);
                $list[] = $items[$i];
            }
        }
```

Рисунок 15 — Логика контроллера

Логика контроллера, отвечающего за чат показана на рисунке 16.

```php
class ChatController extends Controller
{
    const CHAT_CHANNEL = 'chat.message';
    const NEW_MSG_CHANNEL = 'new.msg';
    const DELETE_MSG_CHANNEL = 'del.msg';

    public function __construct()
    {
        parent::__construct();
        $this->redis = Redis::connection();
    }

    public function banchat(Request $request) {
        $steamid64 = $request->get('steamid64');
        $user = User::where('steamid64', $steamid64)->first();

        if($user->banchat == 0){
            User::where('steamid64', $user->steamid64)->update(['banchat' => '1']);
            return response()->json(['status' => 'success', 'reason' => 'Пользователь заблокирован.']);
        }else{
            return response()->json(['status' => 'error', 'reason' => 'Пользователь уже заблокирован.']);
        }
    }

    public function unbanchat(Request $request) {
        $steamid64 = $request->get('steamid64');
        $user = User::where('steamid64', $steamid64)->first();

        if($user->banchat == 1){
            User::where('steamid64', $user->steamid64)->update(['banchat' => '0']);
            return response()->json(['status' => 'success', 'reason' => 'Пользователь разблокирован.']);
        }else{
            return response()->json(['status' => 'error', 'reason' => 'Пользователь не заблокирован.']);
        }
    }
    public static function chat()
    {
        $redis = Redis::connection();

        $value = $redis->lrange(self::CHAT_CHANNEL, 0, -1);
        $i = 0;
        $returnValue = NULL;
        $value = array_reverse($value);

        foreach ($value as $key => $newchat[$i]) {
            if ($i > 20) {
                break;
            }
            $value2[$i] = json_decode($newchat[$i], true);

            $user = DB::table('users')->where('steamid64', $value2[$i]['steamid64'])->first();

            $value2[$i]['username'] = htmlspecialchars($value2[$i]['username']);

            $returnValue[$i] = [
                'userid' => $value2[$i]['userid'],
                'steamid64' => $value2[$i]['steamid64'],
                'avatar' => $value2[$i]['avatar'],
                'colors' => $value2[$i]['colors'],
                'time' => $value2[$i]['time'],
                'time2' => $value2[$i]['time2'],
                'support' => $value2[$i]['support'],
                'ban' => $value2[$i]['ban'],
                'mute' => $user->banchat,
                'messages' => $value2[$i]['messages'],
                'username' => $value2[$i]['username'],
                'admin' => $value2[$i]['admin']];

            $i++;
```

Рисунок 16 — Логика контроллера

Функция, отвечающая за внесение поинтов в игру показана на рисунке 17.

```
public function addBet(Request $r)
{
    if(Auth::guest()) return;

    if(Session::get('applocale') == 'ru') {
        if($r->get('value') < $this->config->double_min_bet) return [
            'msg'       => 'Минимальная сумма ставки - '.$this->config->double_min_bet,
            'type'      => 'error'
        ];

        if($r->get('value') > $this->config->double_max_bet) return [
            'msg'       => 'Максимальная сумма ставки - '.$this->config->double_max_bet,
            'type'      => 'error'
        ];

        if ($r->get('real') == 'false')
        {
            if($this->user->money < $r->get('value')) return [
                'msg'       => 'Недостаточно Free Coin на балансе!',
                'type'      => 'error'
            ];
        }
        else
        {
            if($this->user->real_money < $r->get('value')) return [
                'msg'       => 'Недостаточно Real Coin на балансе!',
                'type'      => 'error'
            ];
        }

        if($this->game->status > 1) return [
            'msg'       => 'Ставки в эту игру уже не принимаются!',
            'type'      => 'error'
        ];
    } else {
        if($r->get('value') < $this->config->double_min_bet) return [
            'msg'       => 'The minimum bet amount - '.$this->config->double_min_bet,
            'type'      => 'error'
        ];

        if($r->get('value') > $this->config->double_max_bet) return [
            'msg'       => 'The maximum bet amount - '.$this->config->double_max_bet,
            'type'      => 'error'
        ];

        if($this->user->money < $r->get('value')) return [
            'msg'       => 'Not enough PT in the balance!',
            'type'      => 'error'
        ];

        if($this->game->status > 1) return [                                    rt();
            'msg'       => 'Bets in this game are no longer accepted!',
            'type'      => 'error'
        ];
    }
```

Рисунок 17 — Функция «addBet»

Для автоматизации процессов на сайте был реализован бот на nodeJS с использованием специальной библиотекой steam. Изначально настройки веб-сервиса мы получаем напрямую из бд, соединив бота с субд, данная функция показана на рисунке 18.

```
/* — Socket Server — */
var config = require('./config.js'),
app = require('express')(),
server = require('http').Server(app),
io = require('socket.io')(server);

/* — Dev — */
var double_timer = 0,
rotate = 0;

var online = 0;

var timer, ngtimer;

/* — Log Function — */
function log(log) { console.log('[APP] ' + log); }

/* — Etc. init — */

var mysql = require('mysql');

var client = mysql.createConnection({
host : config.mysql.host,
port : config.mysql.port,
user : config.mysql.user,
password : config.mysql.password,
database : config.mysql.database
});

client.query('select * from settings', function(err, res) {
if(err) {
log('Ошибка при выполнении mysql запроса.');
return;
}
if(typeof res[0] == 'undefined') {
log('Не удалось найти настройки сайта.');
} else {
res = res[0];
server.listen(res.socket_port);
log('Socket сервер запущен на порте ' + res.socket_port);
}
```

Рисунок 18 — Соединение бота с БД

После бот в автоматическом режиме контролирует процесс игры, основные из них: проверяет ставки, начинает игру, определяет победителя, создает новую игру, выводит предметы

Функция внесения депозита показана на рисунке 19.

```
function deposit(data) {
if(!cookiesSet) return setTimeout(function() {
deposit(data);
}, 5000);
var sendItems = [];
for(var i = 0; i < data.items.length; i++) {
if(typeof data.items[i] != 'undefined') sendItems.push({
appid : 730,
contextid : 2,
assetid : data.items[i]
});
}

var offer = Manager.createOffer(data.url);
offer.addTheirItems(sendItems);
sendDepOffer(offer, data, 5);
}
```

Рисунок 19 — Функция внесения депозита

Функция вывода предметов показана на рисунке 20.

```
function withdraw(data) {
if(!cookiesSet) return setTimeout(function() {
withdraw(data);
}, 5000);
var sendItems = [];
for(var i = 0; i < data.items.length; i++) {
if(typeof data.items[i] != 'undefined') sendItems.push({
appid : 730,
contextid : 2,
assetid : data.items[i]
});
}

var offer = Manager.createOffer(data.url);
offer.addMyItems(sendItems);
offer.setMessage('Вывод с сайта ' + config.siteName + ' на сумму ' + data.value + ' PTS');
sendOffer(offer, data, 5);
}
```

Рисунок 20 — Функция вывода предметов

Функция авторизации показана на рисунке 21.

```
function login(data) {
Steam.login({
accountName : data.username,
password : data.password,
twoFactorCode : SteamTotp.generateAuthCode(data.shared_secret)
}, function(err, sessionID, cookies) {
if(err) {
log('Ошибка при авторизации в STEAM. Повторяем через 10 сек. ('+err.message+')');
setTimeout(function() {
login(data);
}, 10000);
return;
}
log('Авторизировались в STEAM!');
Manager.setCookies(cookies, function(err) {
if(err) {
log('Ошибка при подключении куки к SteamTradeofferManager. Перезаходим в STEAM через 10 сек.');
setTimeout(function() {
login(data);
}, 10000);
return;
}
log('Подключили куки к SteamTradeofferManager!');
cookiesSet = true;
});
Steam.startConfirmationChecker(10000, data.identity_secret);
});

Steam.on('confKeyNeeded', function(tag, callback) {
var time = Math.floor(Date.now() / 1000);
callback(null, time, SteamTotp.getConfirmationKey(data.identity_secret, time, tag));
});

Steam.on('newConfirmation', function(confirmation) {
var time = Math.floor(Date.now() / 1000);
var key = SteamTotp.getConfirmationKey(data.identity_secret, time, 'allow');
confirmation.respond(time, key, true, function(err) {
if(err) {
log(err)
return;
}
log('Успешно подтвердили трейд');
});
});
}

Manager.on('newOffer', function(offer) {
if(offer.itemsToReceive.length !== 0 && config.admins.indexOf(offer.partner.getSteamID64()) != -1) {
offer.accept(function (err, status) {
if(!err) {
log("Трейд от админа #" + offer.id + " Принят");
} else {
log("Трейд от админа #" + offer.id + " Ошибка");
}
});
}
if(offer.itemsToGive.length !== 0) {
offer.decline();
return;
}
});
```

Рисунок 21 — Функция авторизации

Для хранения информации была создана база данных, структура базы данных показана на рисунке 22.

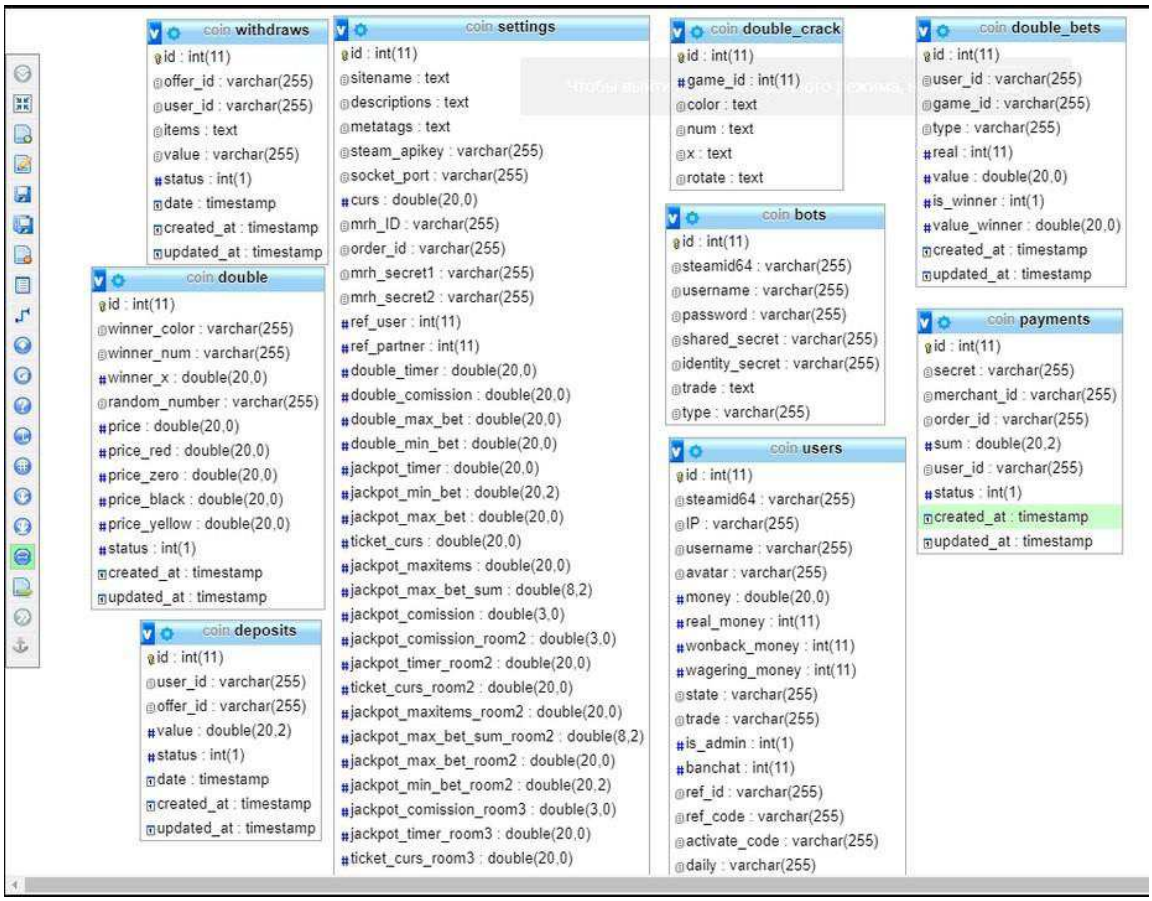


Рисунок 22 — Схема данных БД

Главной особенностью в созданной базе данных является отсутствие связей между таблицами. Данное решение было принято исходя из двух пунктов. Первый пункт заключается в том, что изначально базы данных проектировались без связей между таблицами, позже было решено реализовать возможность построения связей, одной из причин является каскадное удаление данных, т.к объемы памяти были малы и не позволяли хранить большой объем данных, но на сегодняшний день такой проблемы нет. Второй пункт заключается в том, что существующие связи между таблицами существенно затормаживают систему, поэтому было решено отказаться от их построения, что привело к более быстрой работе системы.

Интерфейс сайта создавался на основе готового макета с использованием framework bootstrap 3, использование готового макета позволило существенно сэкономить время разработки конечного продукта.

В качестве меню мы использовали стандлартные классы bootstrap <nav class="navbar navbar-default navbar-static-top" role="navigation">.

Дополнительно для мобильной адаптации мы использовали классы:

– col-xs-n;

– col-sm-n;

– col-md-n;

– col-lg-n;

Где, n - ширина в % от 1, до 12 в каждом классе своя ширина, начиная от 3%, xs - мобильные устройства, sm – планшеты, md - стандартная ширина монитора, lg - широкоформатные мониторы.

## 4 Демонстрация функционала веб-сервиса

В данной главе будут рассмотрены результаты основных действий пользователей.

При авторизации пользователь попадает на сайт steam, где вводит логин и пароль, при этом подтверждает, что его данные аккаунта будут публичны, после этого он перенаправляется на наш веб-портал. Форма авторизации продемонстрирована на рисунке 23.

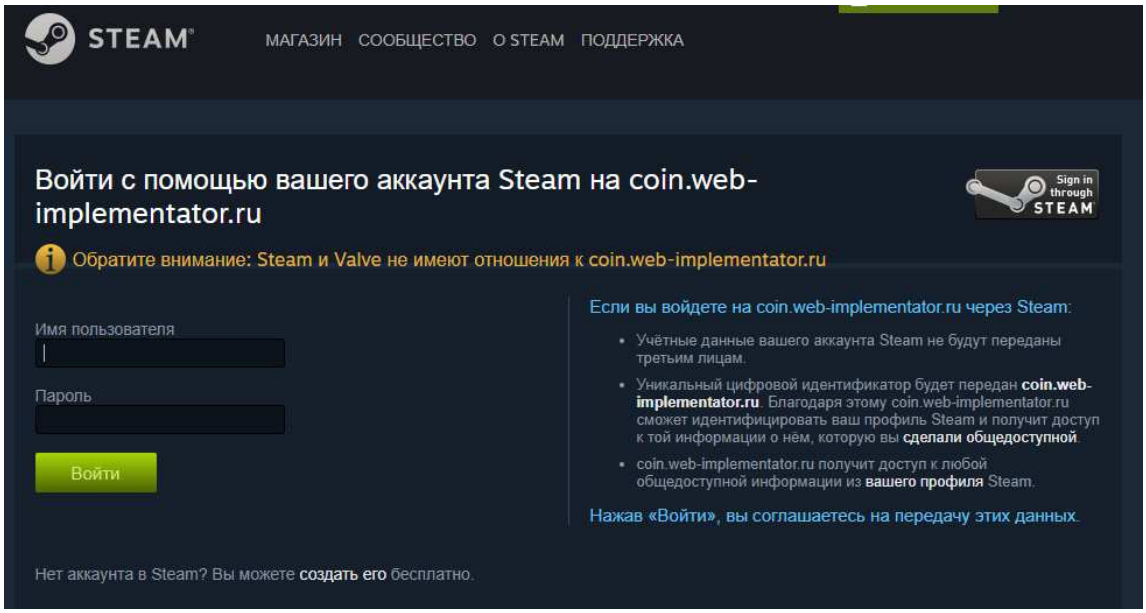


Рисунок 23 — Форма авторизация

Для внесений предметов пользователь должен перейти на вкладку «deposit». После перехода сперва идёт загрузка инвентаря аккаунта. После того как загрузка окончена, пользователь выбирает предметы для обмена на поинты и нажимает депозит. При этом бот отправляет данный депозит на подтверждение. После того как пользователь подтвердил операцию у него на счету появляется free coin и он может принимать участие в розыгрыше. Данная форма продемонстрирована на рисунке 24.

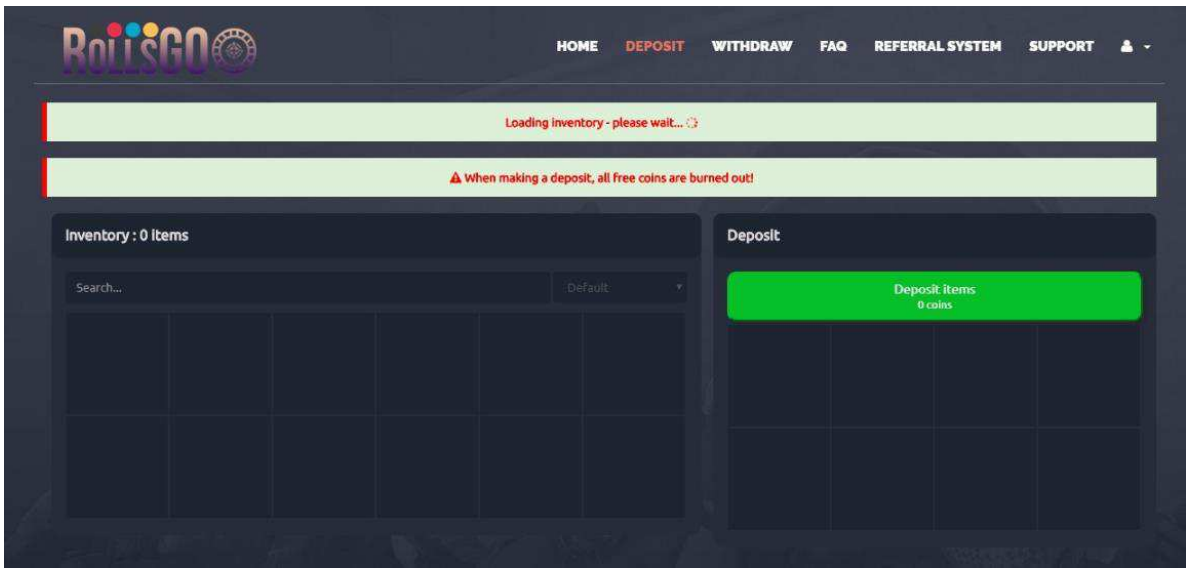


Рисунок 24 — Форма внесения депозита

На данном веб-портале используется следующий алгоритм, при котором пользователю доступны 4 цвета, он выбирает один из них и проводиться розыгрыш, если участник угадывает, то его выигрыш зависит от выбранного им цвета. На рисунке 25 продемонстрирована данная процедура.

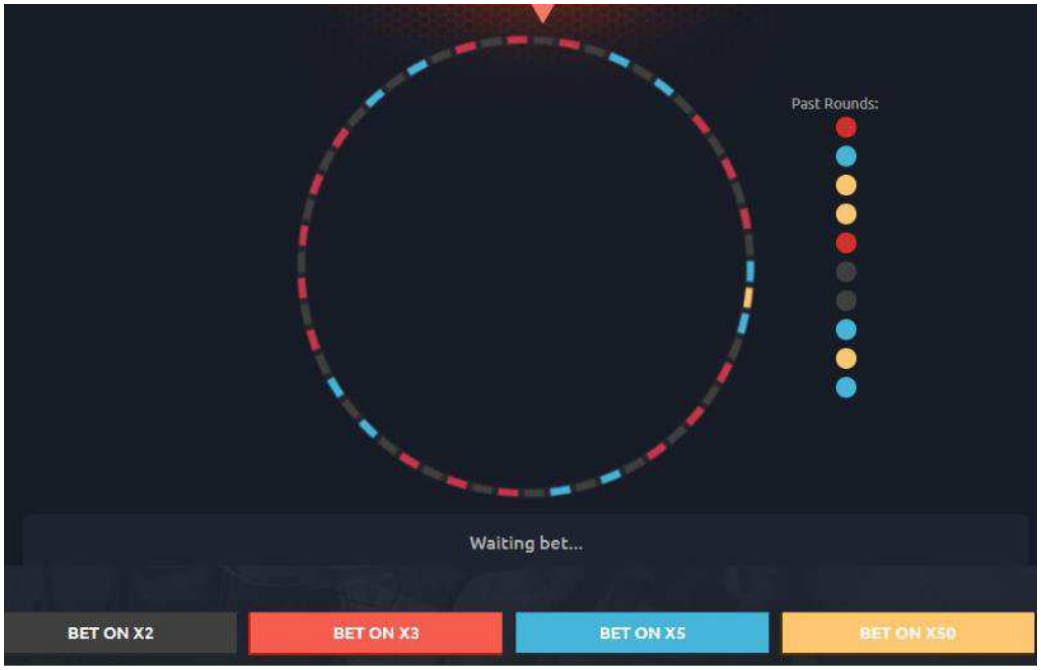


Рисунок 25 — Процедура игры

После выигрыша пользователь может обменять free coin на real coin, для того чтобы можно было совершать покупку игровых товаров. Данная процедура показана на рисунке 26.


Рисунок 26 — Функция обмена free coin на real coin

Передача игровых предметов на аккаунт пользователя проходит на вкладке «withdraw». После перехода по ней, юзер выбирает интересующие его предметы и подтверждает покупку, после этого бот автоматически передает скин в инвентарь пользователя. Функция покупки предмета продемонстрирована на рисунке 27.

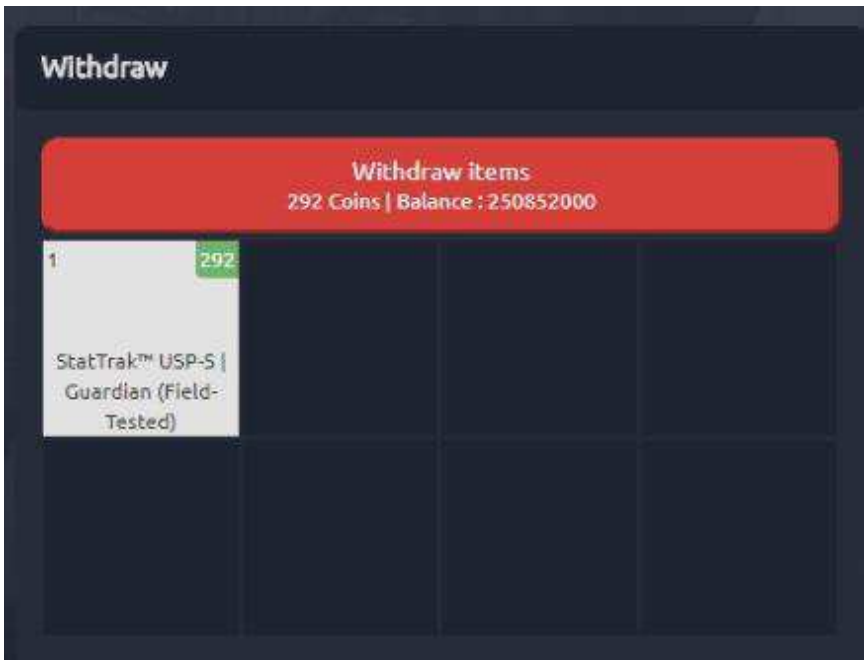


Рисунок 27 — Функция вывода предметов

Так же была реализована функция «чат», чтобы игроки могли коммуницировать друг с другом, данная функция показана на рисунке 28.

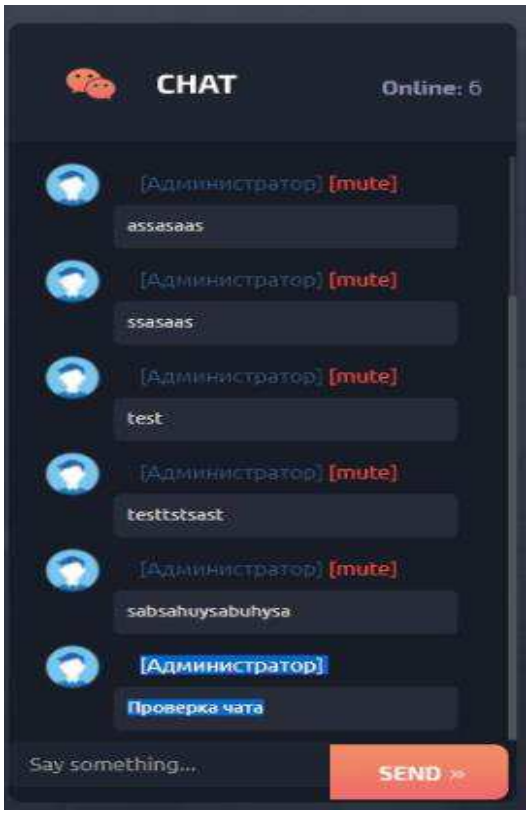


Рисунок 28 — Чат

# ЗАКЛЮЧЕНИЕ

В результате бакалаврской работы был разработан веб-сервис, позволяющий пользователям уменьшить затраты на приобретение внутриигрововo контента.

Для достижения поставленной цели были решены следующие задачи:

– Выбор инструментов для реализации веб-сервиса;

– Проектирование веб-сервиса;

– Разработка веб-сервиса.

В результате решения первой задачи были выбраны инструменты для реализации веб-сервиса.

В рамках решения второй задачи были построены функциональная модель, а также спроектирована база данных.

При разработке было выявлено что, выбор технологий для реализации веб-сервиса сделанный при решение первой задачи полностью себя оправдал.

Разработка велась с использованием Framework Laravel, реализующий архитектуру model-view-controller, на языке программирования PHP с использованием HTML, CSS, JS, база данных MySQL, веб-сервера Apache, а также при разработке бота применялся nodeJS.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Бейли, Л. Изучаем SQL. / Л. Бейли – Санкт-Петербург: 2012. – 592 с.

2. Введение в JavaScript [Электронный ресурс] : особенности и уникальность. / Современный учебник Javascript. – Режим доступа: https://learn.javascript.ru/intro

3. Зандстра, М. PHP. Объекты, шаблоны и методики программирования / Мэт Зандстра – Москва : Издательский дом «Вильямс», 2016. – 576 с.

4. Крупнейший в Европе ресурс для IT-специалистов [Электронный ресурс] : Реализация MVC паттерна на примере создания сайта-визитки на PHP // «Хабрахабр». – Режим доступа: https://habrahabr.ru/post/150267/

5. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript / Р. Никсон. – Санкт-Петербург : Питер, 2013. – 496 с.

6. Никсон, Р. Learning PHP, MySQL, JavaScript, CSS & HTML5 : A Step-by-Step Guide to Creating Dynamic Websites / Р. Никсон. – Москва : Питер, 2016. – С. 180–197.

7. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы : учебник для вузов / В. Г. Олифер, Н. А. Олифер. – Санкт-Петербург : Питер, 2010. – 944 с.

8. Олищук, А. Разработка Web-приложений на PHP / А. Олищук. – Москва : Вильямс, 2006. – 352 с.

9. Пирогов, В. Ю. Информационные системы и базы данных: организация и проектирование / В. Ю. Пирогов. – Санкт-Петербург : 2009. – 87 с.

10. Полонская, Е. Л. Язык HTML. Самоучитель. / Е. Л. Полонская – Москва : Издательский дом «Вильяме», 2005. – 320 с.

11. Попов, В. К. Практикум по Интернет-технологиям / В. К. Попов. – Санкт-Петербург: Питер, 2002. – 162 с.

12. Работа с базой данных. MySQL и phpMyAdmin [Электронный ресурс] : / – Режим доступа: https://metanit.com/web/php/7.1.php.

13. Реляционная база данных [Электронный ресурс] : / – Режим доступа: https://ru.bmstu.wiki/

14. Рязанцева, Л. Что нам стоит сайт построить / Л. Рязанцева // Библиополе. – 2008. – № 8. – С. 20–21.

15. СТО 4.2–07–2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. – Введ. 30.12.2013. – Красноярск : СФУ, 2014. – 60 с.

16. Строительство Web-сайтов / В. А. Фридман, А. В. Александров, Г. Г. Сергеев, С. П. Костин. – Москва : Триумф, 2011. – 288 с.

17. Титоров, Д. Ю. Технология создания интерактивных сайтов / Д. Ю. Титоров // Информатика : [газ. Изд. дома "Первое сентября"]. – 2010. – № 3 (февр.). – С. 13–18.

18. Чебыкин, Р. И. Разработка и оформление текстового содержания сайтов / Р. И. Чебыкин. – Москва : БХВ–Петербург, 2014. – 528 с.

19. Bootstrap по-русски [Электронный ресурс] : описание фреймворка, используемые технологии. – Режим доступа: http://mybootstrap.ru/

20. SEO — искусство раскрутки сайтов / Э. Энж [и др.]. – Москва : БХВ-Петербург, 2011. – 592 с.

## ПРИЛОЖЕНИЕ А

### Реализация процесса игры

```
namespace App\Http\Controllers;

use DB;
use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Session;
use Auth;
use App\Double;
use App\Settings;
use Redis;
use Carbon\Carbon;

class DoubleController extends Controller
{
   public function __construct()
   {
      if(Auth::check()) {
         $this->user = Auth::user();
         view()->share('u', $this->user);
      }
      $this->game     = Double::orderBy('id', 'desc')->first();
      $this->redis    = Redis::connection();
      $this->rotate   = $this->redis->get('rotate');
      $this->config   = Settings::first();
      view()->share('rotate', $this->rotate);
      view()->share('config', $this->config);
   }
   public function index()
   {
      view()->share('games', $this->getLastGamesData());
```

```
        view()->share('bets', $this->getBets());
        if(Auth::check()) view()->share('person', $this->getPerson());
        view()->share('game', $this->game);
        view()->share('title', 'Double');
        view()->share('time', $this->config->double_timer);
        return view('pages.double.index');
    }

    public function addBet(Request $r)
    {
        if(Auth::guest()) return;

                    if(Session::get('applocale') == 'ru') {
                        if($r->get('value') < $this->config->double_min_bet) return [
                            'msg'        => 'Минимальная сумма ставки - '.$this->config->double_min_bet,
                            'type'      => 'error'
                        ];

                        if($r->get('value') > $this->config->double_max_bet) return [
                            'msg'        => 'Максимальная сумма ставки - '.$this->config->double_max_bet,
                            'type'      => 'error'
                        ];

                        if ($r->get('real') == 'false')
                        {
                            if($this->user->money < $r->get('value')) return [
                                'msg'       => 'Недостаточно Free Coin на балансе!',
                                'type'      => 'error'
                            ];
                        }
                        else
                        {
                            if($this->user->real_money < $r->get('value')) return [
```

```
                    'msg'       => 'Недостаточно Real Coin на балансе!',
                    'type'      => 'error'
                ];
            }

            if($this->game->status > 1) return [
                'msg'       => 'Ставки в эту игру уже не принимаются!',
                'type'      => 'error'
            ];
        } else {
            if($r->get('value') < $this->config->double_min_bet) return [
                'msg'           => 'The minimum bet amount - '.$this->config->double_min_bet,
                'type'      => 'error'
            ];



            if($r->get('value') > $this->config->double_max_bet) return [
                'msg'           => 'The maximum bet amount - '.$this->config->double_max_bet,
                'type'      => 'error'
            ];

            if($this->user->money < $r->get('value')) return [
                'msg'       => 'Not enough PT in the balance!',
                'type'      => 'error'
            ];

            if($this->game->status > 1) return [
                'msg'       => 'Bets in this game are no longer accepted!',
                'type'      => 'error'
            ];
        }
    // Проверки на два цвета
```

```
$bet = DB::table('double_bets')->where('game_id', $this->game->id)->where('user_id', $this->user->id)->first();

if(!is_null($bet)) {
                if(Session::get('applocale') == 'ru') {
                        if($bet->type != $r->get('type')) return [
                                'msg' => 'Вы можете поставить только на : '.$bet->type,
                                'type' => 'error'
                        ];
                } else {
                        if($bet->type != $r->get('type')) return [
                                'msg' => 'You can put only : '.$bet->type,
                                'type' => 'error'
                        ];
                }
}

        if ($r->get('real') == 'false') { $this->user->money -= $r->get('value'); $new_money = $this->user->money; $real = 0; }
        else { $this->user->real_money -= $r->get('value'); $new_money = $this->user->real_money; $real = 1; }
$this->user->save();

DB::table('double_bets')->insert([
   'user_id'   => $this->user->id,
   'game_id'   => $this->game->id,
   'type'      => $r->get('type'),
   'real'      => $real,
   'value'     => $r->get('value')
]);

switch($r->get('type')) {
   case 'red' :
      $this->game->price_red += $r->get('value');
```

```
            break;
            case 'blue' :
                $this->game->price_zero += $r->get('value');
            break;
            case 'black' :
                $this->game->price_black += $r->get('value');
            break;
                        case 'yellow' :
                $this->game->price_yellow += $r->get('value');
            break;
        }

        $this->game->price += $r->get('value');
        $this->game->save();

        $this->getPersonValue($this->user->id);
        $this->getTimer();

        $returnValue = [
            'user_id'   => $this->user->id,
            'username'  => $this->user->username,
            'avatar'    => $this->user->avatar,
            'type'      => $r->get('type'),
            'value'     => $r->get('value'),
            'allValues' => [
                'red'       =>      DB::table('double_bets')->where('user_id',      $this->user->id)-
>where('game_id', $this->game->id)->where('type', 'red')->sum('value'),
                'blue'      =>      DB::table('double_bets')->where('user_id',      $this->user->id)-
>where('game_id', $this->game->id)->where('type', 'blue')->sum('value'),
                'black'     =>      DB::table('double_bets')->where('user_id',      $this->user->id)-
>where('game_id', $this->game->id)->where('type', 'black')->sum('value'),
                'yellow'    =>      DB::table('double_bets')->where('user_id',      $this->user->id)-
>where('game_id', $this->game->id)->where('type', 'yellow')->sum('value'),
            ],
            'global'    => [
```

```
            'red'   => $this->game->price_red,
            'blue'  => $this->game->price_zero,
            'black' => $this->game->price_black,
            'yellow' => $this->game->price_yellow,
        ]
    ];

    $this->redis->publish('double.newBet', json_encode($returnValue));

    $this->redis->publish('updateBalance', json_encode([
        'id'    => $this->user->id,
        'money' => $new_money
    ]));
    if(Session::get('applocale') == 'ru') {
                        return [
                                'msg'   => 'Ваша ставка зашла в игру!',
                                'type'  => 'success'
                        ];
                } else {
                        return [
                                'msg'   => 'Your bet has come into game!',
                                'type'  => 'success'
                        ];
                }
}

public function getPerson()
{
    $array = [
        'red'   => 0,
        'blue'  => 0,
        'yellow'  => 0,
        'black' => 0
    ];
```

```
        $bets = DB::table('double_bets')->where('user_id', $this->user->id)->where('game_id', $this->game->id)->get();
        foreach($bets as $bet) $array[$bet->type] += $bet->value;

        return $array;
    }

    public function getPersonValue($id) {
        $array = [
            'red'     => 0,
            'blue'    => 0,
            'black'   => 0,
                        'yellow'  => 0,
            'user_id' => $id
        ];

        $bets = DB::table('double_bets')->where('user_id', $id)->where('game_id', $this->game->id)->get();
        foreach($bets as $bet) $array[$bet->type] += $bet->value;

        $this->redis->publish('double.updatePersonaValue', json_encode($array));
    }

                public function getCheckTwist()
            {
                $double_crack = DB::table('double_crack')->where('game_id', $this->game->id)->first();

                if(!$double_crack) return;

                $this->game->winner_color   = $double_crack->color;
                $game_color                 = $double_crack->color;

                $this->game->winner_num     = $double_crack->num;
                $game_number                = $double_crack->num;
```

```
                $this->game->winner_x       = $double_crack->x;
                $rotate   =   360-($this->rotate-floor($this->rotate/360)*360)+$double_crack->rotate+$this->rotate+1080;
                                $this->game->status         = 1;
                $this->game->save();
                $winner_value   =   DB::table('double_bets')->where('game_id',   $this->game->id)->where('type', $this->game->winner_color)->sum('value')*$this->game->winner_x;
                $returnValue = [
                        'time'          => $this->config->double_timer,
                        'rotate'        => $rotate,
                        'color'         => $game_color,
                        'number'        => $game_number,
                        'random_number' => $this->game->random_number,
                        'winner_value'  => $winner_value
                ];
            $this->redis->publish('double.timer', json_encode($returnValue));
        }
   public function updateStatus(Request $r)
   {
      $this->game->status = $r->get('status');
      $this->game->save();
   }
    public function newGame()
   {
      $this->game->status = 3;
      $this->game->save();
            // Отдаем выигрыш победителям.
      $bets = DB::table('double_bets')->where('game_id', $this->game->id)->get();
      foreach($bets as $bet) {
         if($bet->type == $this->game->winner_color) {
            $user = User::where('id', $bet->user_id)->first();
            if($user) {
               DB::table('double_bets')->where('id', $bet->id)->update([
                  'is_winner'     => 1,
```

```
                'value_winner'  => $bet->value*$this->game->winner_x
            ]);
                        if ($bet->real == 0) { $user->money += $bet->value*$this->game->winner_x; $new_money = $user->money; }
                        else { $user->real_money += $bet->value*$this->game->winner_x; $new_money = $user->real_money; }
            $user->save();

                                $this->redis->publish('updateBalance', json_encode([
                                        'id'    => $user->id,
                                        'real'    => $bet->real,
                                        'money' => $new_money
                                ]));            }           }       }

            // Отыгрышь
            foreach($bets as $bet) {
        $user = User::where('id', $bet->user_id)->first();
        if($user) {
           $itogo = $user->wagering_money - $user->wonback_money;
                        if ($bet->value < $itogo) {
                                $w_money = $user->wonback_money + $bet->value;
                                DB::table('users')->where('id', $user->id)->update([
                                        'wonback_money'   => $w_money
                                ]);
                        }
                        else
                        {
                                 DB::table('users')->where('id', $user->id)->update([
                                        'wonback_money'   => $user->wagering_money
                                ]);
                        }
                    }
    }
```

```
        // Создаем новую игру
        Double::insert([
            'random_number' => '0.'.mt_rand(100000000, 999999999).mt_rand(100000000, 999999999).mt_rand(100000000, 999999999).mt_rand(100, 999)
        ]);

        $this->game = Double::orderBy('id', 'desc')->first();

        $returnValue = [
            'time' => $this->config->double_timer
        ];

        $this->redis->publish('double.newGame', json_encode($returnValue));
    }

    public function getGameId()
    {
        $str = '';
        $strlen = strlen($this->game->id);
        $u = 7-$strlen;
        for($i = 0; $i < $u; $i++) {
            $str .= '0';
        }
        $str .= $this->game->id;
        return $str;
    }

    public function getLastGamesData()
    {
        $list = [];

        $games = Double::where('status', 3)->orderBy('id', 'desc')->limit(10)->get();
        foreach($games as $game) {
            $list[] = [
                'num'   => $game->winner_num,
```

```
                'color' => $game->winner_color
            ];
        }

        return $list;
    }

    public function getBets()
    {
        $bets = DB::table('double_bets')->where('game_id', $this->game->id)->orderBy('id',
'desc')->get();
        foreach($bets as $i => $bet) {
            $user = User::where('id', $bet->user_id)->first();
            $bets[$i]->username = $user->username;
            $bets[$i]->avatar = $user->avatar;
        }
        return $bets;
    }

    public static function getLastGame() {
        $games = Double::orderBy('id', 'desc')->first();
        return $games;
    }
```

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и Информационных Технологий
институт
Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ Л.С. Троценко
подпись инициалы, фамилия
«13» июня 2018 г.

# БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии

Разработка веб-сервиса для уменьшения денежных затрат на приобретение игрового контента

Руководитель _____ 13.06.18 С.А. Виденин
подпись, дата инициалы, фамилия

Выпускник _____ 13.06.18 В.В. Акулов
подпись, дата инициалы, фамилия

Нормоконтролер _____ 13.06.18 Ю.В. Шмагрис
подпись, дата инициалы, фамилия

Красноярск 2018