Geekbrains

**Создание веб-приложения**

**(лог анализатора)**

**для проекта игровых серверов Magic Rust**

IT-специалист:

Программист Разработчик

Пантюхов И.А.

Курск

2024

**Содержание**

ВВЕДЕНИЕ 3

Глава 1. Теоретическая часть

Описание технических особенностей проекта 6

1.1 Что такое проект Magic Rust 6

1.2 Типы Серверов в “Rust” 7

1.3 Описание плагинов для модифицированных серверов 8

1.4 Причины выбора языков для клиент-сервера лог-анализатора 9

1.4.1 Выбор языка программирования плагина на игровом-сервере 9

1.4.2 Выбор языка программирования веб-сервера 10

1.4.3 Компоненты динамического сайта на PHP 11

1.4.4 Выбор языка программирования клиента 12

1.4.5 В чем особенности JavaScript 12

1.5 Анализ технического задания 14

1.6 Обоснование выбора СУБД 15

1.7 Выбор среды разработки 15

1.8 Разработка клиент-серверного приложения 16

1.9 Выбор фреймворка 16

1.10 Выбор хостинга веб-сервиса 17

Глава 2 Практическая часть разработки лог анализатора 18

2.1.1 Разработка плагина 18

2.1.2 Описание работы плагина 19

2.2 Описание структуры базы данных 21

2.4 Настройка аутентификации с помощью сервиса Steamя 29

2.5 Инициализация сайта 31

2.6 Работа с сайтом с точки зрения бекэнда 33

2.7 Отслеживание ошибок на странице сайта 35

2.8 Создание своего фреймворка mysql.php 37

2.9 Описание класса GetLogs 43

2.10 Описание сайтового фреймворка 44

5. Заключение 52

Список использованной литературы и ресурсов 53

**ВВЕДЕНИЕ**

●Проект — система из плагина автоматически собирающего информацию из игры и передающий её в базу данных, базы данных собирающая данные с разных одновременно активных игровых серверов, а также веб-сервиса лог-анализатора который даёт быстрый и удобный доступ к данным для сотрудников технической поддержки которые проводят постмодерацию сообщений и разрешают жалобы на встроенный магазин заказчика.

● Актуальность проекта — состоит в том что нет удобного инструмента для часто появляющейся задачи.

● Цель проекта — создать систему из интегрируемого плагина для логирования действий пользователей приложения клиента, базу данных для хранения логов со всех серверов клиента, веб-приложение для удобного просмотра логов сотрудниками заказчика.

● План работы — каскадный метод разработки, создание полностью рабочей версии со всеми оговоренными возможностями, тестирование и верификация системы, затем поддержка по мере запросов.

● Моя роль в этом проекте - Программист разработчик бекэнда.

● Умения для решения этой задачи были получены при прохождении курсов Geekbrains, так что я решил это возможностью проверить себя работой над этим проектом.

● Во время работы использовались среды разработки Visual Studio Code, а также мощный редактор текста notepad++.

● В разработке проекта использованы следующие технологии:

1. Разработку плагина решено реализовать на основе фреймворка Oxide на .NET
2. База данных решено реализовать на подсистеме СУБД MySQL – InnoDB с системой администрирования phpMyAdmin
3. Бекэнд веб-приложения решено реализовать на языке php потому что он имеет более удобные инструменты взаимодействиями с выбранной базой данных
4. Фронтэнд веб приложения решено реализовать на JavaScript потому что он имеет более удобные инструменты взаимодействиями с форматом JSON, а также позволяет сделать удобный и интерактивный сайт.

● Участвовал в командном проекте в роли backend-разработчика. Также на проекте были задействованы: технический директор и интерфейс дизайнер.

# Глава 1 Теоретическая часть

# Описание технических особенностей проекта

## 

## 1.1 Что такое проект Magic Rust

## к которому будет создана система лог-анализатора

Ни для кого не секрет, что в наше время компьютерные сетевые игры занимают значительную часть времени большинства людей. Почти любой игровой проект способен приносить не малые деньги, причем не только разработчикам и издателям, но и сообществу которое предлагает изменённый опыт взаимодействия.

Несмотря на то, что разработчики дают полное разрешение сообществу на изменение своих игр, они не всегда дают инструментарий для создания этого контента, поэтому энтузиастам приходиться изобретать свои собственные решения для работы с игровым контентом.

Для этого диплома, а также по заказу проекта “MAGIC RUST”, был разработан инструментарий, который поможет с отслеживанием действий игрока в многопользовательской среде.

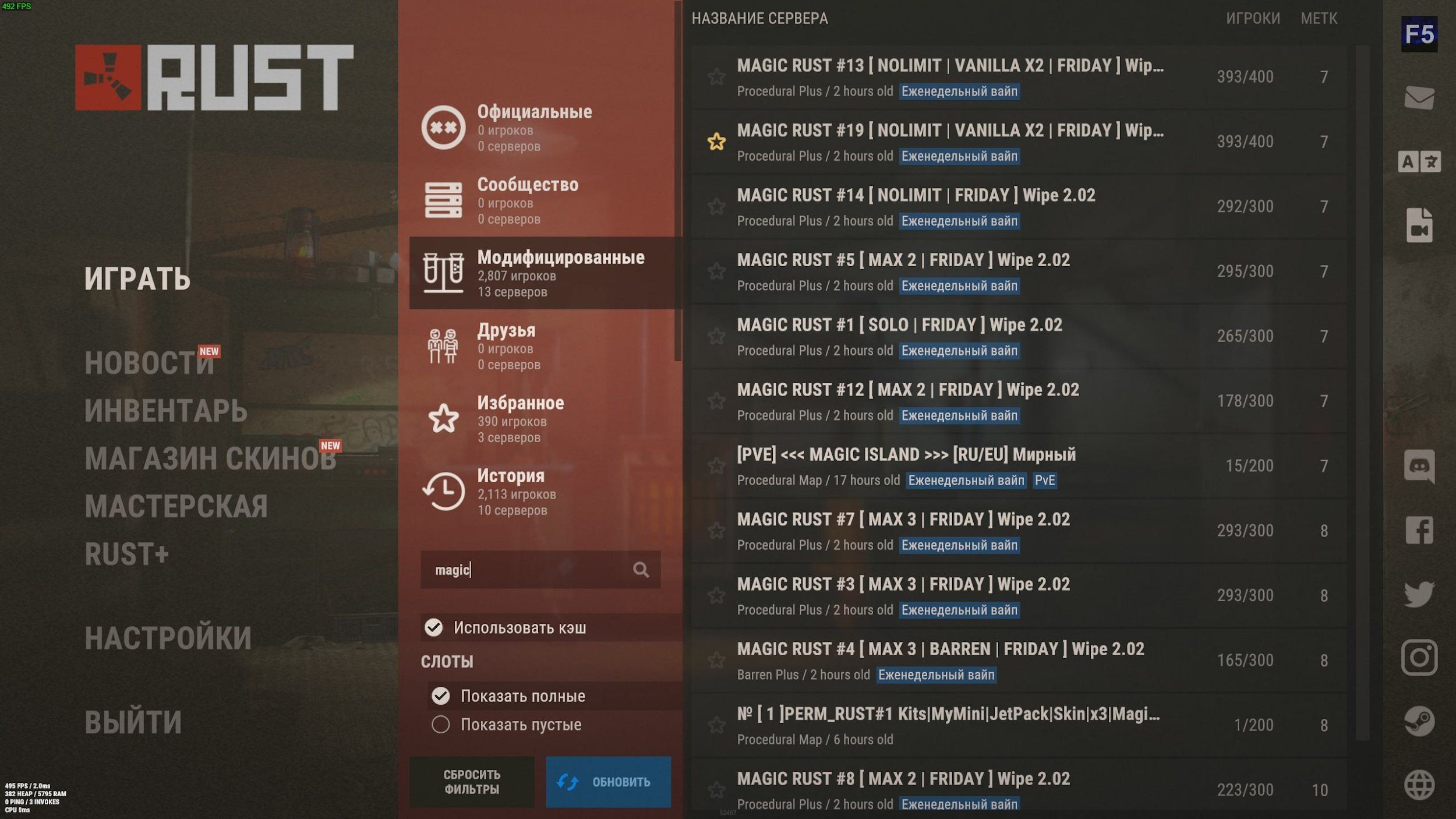
Решение, рассматриваемое в рамках этого диплома, служит как раз для обеспечения понимания структуры конфигурации плагинов и облегчения работы с ними, не требуя от пользователя особых знаний, а также отработке таких повсеместных действий в веб-разработке как безопасное обращение к базе данных и создание удобного представления данных.

## 1.2 Типы Серверов в “Rust”

“Rust” — это игра в жанре выживания, в которой игровые сервера могут быть предоставлены не только от разработчиков, но и пользовательскими от сообщества.

Есть 3 типа серверов:

* Официальные - сервера, запущенные или одобренные разработчиком игры.
* Классические от сообщества - сервера, запущенные сообществом, незначительно сильно отличающиеся от оригинальных.
* Модифицированные - сервера, так же запущенные сообществом, но имеющие сильные изменения, достаточно влияющие на игровой процесс или полностью его меняя.

(рисунок 1) Меню выбора игрового сервера, со стороны игрока требуется только оригинальный дистрибутив и никаких самостоятельных изменений

## 1.3 Описание плагинов для модифицированных серверов

Последний указанный выше тип серверов обычно имеет установленную модификацию Oxide, позволяющую в реальном времени загружать дополнительные плагины, на которых работают все модифицированные сервера.

Модификация Oxide позволяет использовать для написания плагинов пять языков программирования, среди которых :

* Lua - Скриптовый язык программирования
* Python - Высокоуровневый язык программирования
* JavaScript - Мультипарадигменный язык программирования
* CoffeeScript - Язык программирования, транслируемый в JavaScript
* Пятым языком стал C#, который имел наивысшую производительность, поскольку сама игра написана на том же языке.

Независимо от языка программирования, на котором написаны плагины, всегда используется одна и та же система хранения данных, а именно файлы в формате Json.

Json (JavaScript Object Notation) - формат для хранения и обмена информацией, доступной для чтения человеком. Файл содержит только текст и использует расширение .json. Формат хоть и доступен для чтения, но вносить изменения бывает сложно из-за отсутствия понимания и возможности нарушить структуру формата.

Этой модификации есть альтернатива - Carbon, новейший фреймворк для внедрения внутриигровых изменений, но он еще находится на стадии после релизной доработки и не имеет всех необходимых функций уже отработанного Oxide, а также обладает менее разработанной документацией и не собрал большого сообщества активных разработчиков овладевших им и оказывающих помощь на технических форумах.

## 1.4 Причины выбора языков для клиент-сервера лог-анализатора

## 1.4.1 Выбор языка программирования плагина на игровом-сервере

В настоящее время существует действительно огромное количество доступных инструментов для разработки плагинов, и это делает выбор как никогда сложным.

Язык программирования C# считается одним из самых универсальных. Он применяется в самых разных сферах. Например, для создания продвинутых бизнес-приложений, видеоигр, функциональных веб-приложений.

Данный язык получил широкое распространение среди разработчиков видеоигр. С# применяется для создания игр под Windows, macOS, Android и iOS. Объясняется это тем, что этот язык лучше всего подходит для работы с Unity (движком для создания игр). Именно по этой причине разработчики столь часто применяют комбинацию из Unity и C#.

Все это означает, что будущий разработчик плагина, выбрав C#, с высокой степенью вероятности сможет найти учебные материалы, удобные инструменты разработки, ответы на свои вопросы и решения проблем, готовые плагины и интеграции. И это не пустые слова: в случае выбора действительно нишевого решения плагины для интеграции с платформами и сторонними сервисами, вероятно, придется писать или адаптировать самому.

Для Разработки на C# существует множество IDE, включая, как платные, так и бесплатные. В наличии также продвинутые инструменты для отладки и профайлинга. Большая часть инструментов одинаково доступна на всех системах на основе современных версий Windows, Linux и MacOS. Таким образом, начать погружаться в разработку можно абсолютно бесплатно и почти на любом программном и аппаратном обеспечении.

## 1.4.2 Выбор языка программирования веб-сервера

Для разработки логики сайта используется один из самых популярных и востребованных на сегодня языков программирования – PHP. Одним из ключевых его преимуществ выступает сравнительная простота практического применения.

Зона ответственности PHP – так называемый backend или функционирование скрытой от пользователя внутренней части интернета. Она представлена программно-аппаратным обеспечением работы серверов, включая хранение, обработку и управление базами данных. PHP заслуженно считается самым популярным инструментом для программирования динамических сайтов.

Сложные проблемы и задачи обычно уже имеют готовые решения, которые создает сообщество в виде готовых частей кода — библиотек. Это позволяет не решать с нуля типовые задачи, а переходить к закрытию потребностей конкретного проекта. Каждая библиотека при этом решает свою часть задач: одна отвечает за безопасное соединение с базами данных, другая — за отправку сообщений в Telegram-бот, а третья — за анализ данных.

## 

## 1.4.3 Компоненты динамического сайта на PHP

Можно выделить четыре ключевых категории файлов, выступающих

компонентами динамического сайта :

1. В их числе базовый элемент проекта – файл index.php. Содержит наиболее значимые характеристики ресурса.
2. Шаблоны, которые описывают отдельные части интернет-страницы, например, шапку или подвал. Их количество определяется структурой и объемом контента, размещенного на сайте.
3. Набор файлов CSS, содержащих подробное описание визуальных составляющих ресурса – стилей оформления, шрифтов и других подобных элементов.
4. Базы данных с кодами функций и другими исходными программными кодами PHP. Обычно управляются СУБД MySQL, так как такая комбинация, как и в случае PHP с HTML, обеспечивает максимальное полное использование возможностей каждого элемента из парной связки.

## 1.4.4 Выбор языка программирования клиента

Выбор языка пал на JavaScript по следующим причинам:

* Доступен во всех браузерах
* Удовлетворяющая скорость и производительность
* Гибкость и удобство использования
* Полностью соответствуют нашим требованиям

JavaScript — это язык скриптов, на котором держится весь frontend веб-разработки. Он позволяет перехватывать события и выполнять различные действия. Например, пользователь кликнул по какой-нибудь кнопке — сработало событие click. И, связав с ним, мы можем выполнить нужную нам функцию — открыть модальное окно или изменить цвет элемента.

JavaScript используется для того, чтобы делать страницы интерактивными, то есть дать пользователю возможность взаимодействовать с элементами.

**1.4.5 В чем особенности JavaScript**

Интеграция с HTML и CSS

HTML — язык разметки, а CSS — таблицы стилей. На них нельзя писать полноценный код: можно только размещать элементы на странице сайта.

А вот на JavaScript уже можно писать код, и этот язык интегрирован с языками разметки. То есть если нужно добавить в разметку скрипт, туда вписывают код на JavaScript.

Но чаще на страницу просто вставляют ссылку на файл: это позволяет использовать один скрипт на многих страницах и не редактировать каждую из них.

Преимущества

* Нагрузка на сервер – JavaScript является клиентским языком, который может динамически снижать требования к серверу. Некоторые приложения могут вообще не требовать сервера для своей работы.
* Богатые интерфейсы – JavaScript может быть использован для разработки таких функций, как перетаскивание слайдеров. Это может радикально улучшить пользовательский интерфейс и пользовательский опыт на сайте.
* Расширенная функциональность – Разработчики JavaScript могут расширять функциональность веб-страницы, создавая сниппеты JavaScript для множества сторонних дополнений.

Недостатки

* Безопасность на стороне клиента – код JavaScript выполняется на компьютере пользователя, что часто приводит к проблемам безопасности из-за вредоносных действий. Именно поэтому многие разработчики отключают JavaScript.
* Поддержка браузеров – JavaScript может интерпретироваться по-разному в зависимости от того, какой браузер его интерпретирует. В результате создание кросс-браузерного кода может стать проблемой для некоторых разработчиков.

## 

## 1.5 Анализ технического задания

Обговорив всё с заказчиком, он передал техническое задание, относящихся к следующему веб-сервису:

* Внутриигровой плагин на игровом сервере,доступный к редактированию только техническому директору заказчика, который собирает и передаёт информацию в базу данных в приблизительно реальном времени
* База данных,доступная только команде разработки и поддержки продукта, в которой коллекционируются данные со всех серверов с вышеуказанным плагином
* Лог анализатор,доступный только для технического персонала, который посылает запрос в базу данных сформированный при помощи интуитивно понятного графического интерфейса и получает отформатированную информацию о действиях пользователей на сервере.

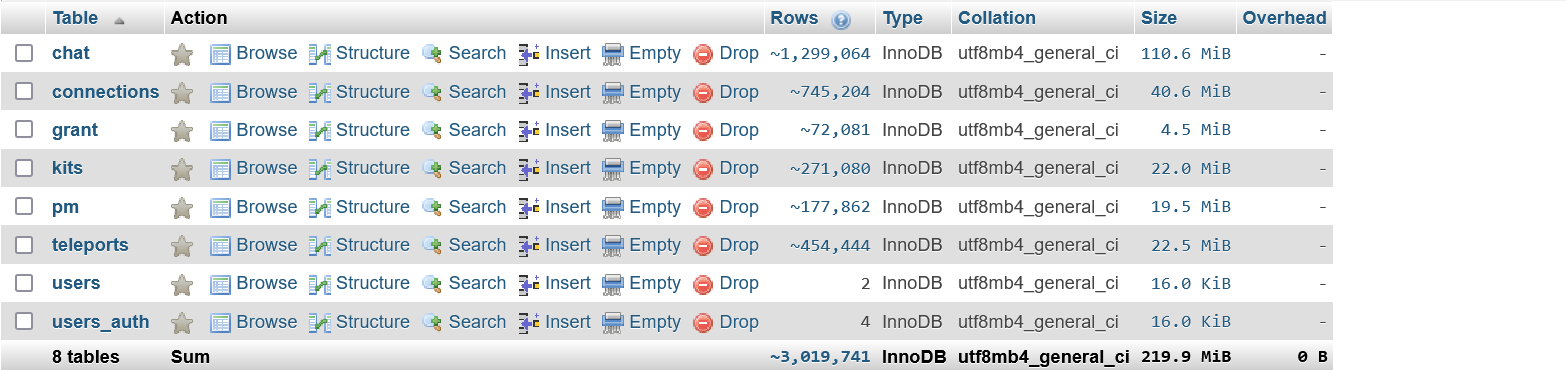
Также была передана тестовая база данных, которые никак не будут меняться, но будет использоваться в тестировании анализатора, также это база данных используется для представления этого проекта для приёмной комиссии.

## 1.6 Обоснование выбора СУБД

InnoDB это транзакционный, реляционный движок работающий на основе MySQL сервера. Начиная с 2001 года он поставляется в стандартной сборке, а с версии 5.1 может устанавливаться в качестве плагина (без необходимости перекомпилировать ядро сервера). Синтаксис очень простой.

InnoDB предназначается для получения максимальной производительности при обработке больших объемов данных. По эффективности использования процессора этот тип намного превосходит другие модели реляционных баз данных с памятью на дисках.

Данные InnoDB в настройках по умолчанию хранятся в больших совместно используемых файлах , что позволяет использовать постраничный кэш страниц базы данных. Формат данных InnoDB обеспечивает надежное хранение данных за счет транзакционности и блокировки данных на уровне строки.

(рисунок 2 — тестовая структура БД с данными интересующими заказчика)

## 1.7 Выбор среды разработки

Для написания проекта была выбрана программа Visual Studio Code. Она поддерживает синтаксис большинства языков, способна выдавать подсказки при написании и в целом оптимизировать работу за счет присутствия автозаполнения, например, закрытия скобок и постановки запятых. Но главная особенность редактора, ради которой был сделан такой выбор - возможность запустить живой сервер сайта, который будет автоматически обновлять страницу при каждом сохранении файла, за счёт чего отпадает необходимость вручную переключаться на окно браузера и обновлять страницу.

## 1.8 Разработка клиент-серверного приложения

После того как были решены проблемы сбора данных с помощью плагина на основе модификации Oxide и коллекционирования её в базе данных, подошла задача быстрого и удобного доступа к агрегированным данным без необходимости вручную составлять запросы к системе администрирования БД.

В первую очередь предстояло выбрать способ связи сервиса с базой данных, ввиду нежелания использовать существующие решения из-за излишней функциональности было решено написать свой инструментарий.

Завершив написание серверного бекэнда, приступили к созданию лёгкого для загрузки и использования одностраничного веб-приложения, для этого мы создали фреймворк построения функциональных блоков в контейнерах, а затем сверстали сайт с упрощенным дизайном и несколькими цветовыми схемами.

## 1.9 Выбор фреймворка

От готовых решений было принято отказаться в пользу разработки собственного фреймворка, поскольку большинство фреймворков содержат гораздо больше функционала, чем требуется для нашего проекта. А это создаст дополнительную нагрузку и увеличит время загрузки страницы, что является нежелательным для веб приложения. Ну и самая главная причина отказа от готовых решений, даже в вспомогательных целях, что в готовых решениях не будет тех функций, которые нам потребуются. Работа со своим собственным фреймворком значительно облегчит добавление нового функционала и редактирование текущего, поскольку ориентироваться в своем продукте гораздо проще, да и не все публичные фреймворки предоставляют открытый код. Чаще всего публичные фреймворки проходят процесс обфускации, что приводит код в нечитаемый вид. Сделано это с целью защиты своих наработок, во избежание кражи в чужие проекты.

## 1.10 Выбор хостинга веб-сервиса

Так как все вычисления проходят на стороне клиента, вычислительные мощности хостинга нас не интересуют. Выбор хостинга нужно делать по следующим критериям:

* Пропускная способность - чем быстрее скорость, тем быстрее будет загружаться сервис.
* Доступное дисковое пространство - для каждой отслеживаемой категории нужна таблица в базе данных.
* Стабильность - если вдруг сайт по какой-то причине сайт станет недоступен, то и сервер не сможет на него отправлять данные, из-за чего он будет вынужден их копить что может привести к сильному увеличению объема данных и затруднению разовой отправки и мы потеряем доступ к сервису.
* Сайт должен иметь стабильную связь с рабочим сервером(в данном случае игровым). Желательно, чтоб хостинг сервера и хостинг сайта находились в одном городе. Это обеспечит наиболее быструю и стабильную отправку накопившихся данных с сервера на сайт с минимальными задержками.

# Глава 2 Практическая часть разработки лог анализатора

## 2.1.1 Разработка плагина

Так как оригинальный функционал не предполагает сбор и пересылку информации о действиях игрока на сторонние ресурсы пришлось самостоятельно создавать плагин для этого.

Плагин должен собирать как оригинально присутствующую информацию в немодифицированной версии серверов, такую как публичный чат, командный чат, клановый чат, карточный чат, подключения и отключения, так и в модифицированной, которая добавляет ранее не присутствующий функционал, такой как приватный чат, получение наборов, телепортация, выдача и окончание услуг.

После собирании и упаковывании информации в Json формате плагин должен отправлять их в подготовленную базу данных без потерь и порчи данных, так как от них зависят решения технических сотрудников о операциях\возвратах в магазине заказчика.

### **2.1.2 Описание работы плагина**

В первую очередь нам требуется как-то загружать данные из игры. Файлов много, поэтому чтобы не повторять один и тот же код много раз, была создана упрощенная функция загрузки из плагина работающего внутри игры

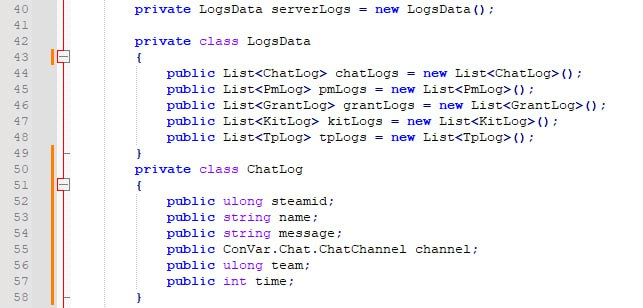
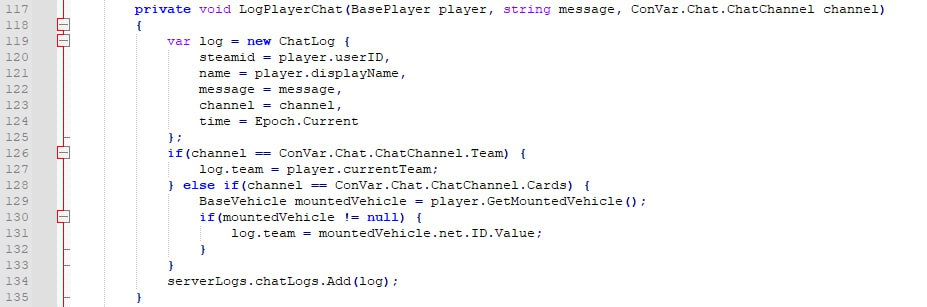
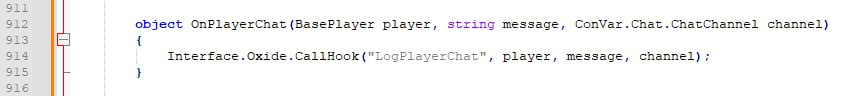
(рисунок 3) Вырезка из плагина “DataBase” класс “ChatLog”

Таблица 1 - Разбор вырезки из плагина “DataBase” класс “ChatLog”

| Переменная | Значение |
| --- | --- |
| steamid | уникальный идентификатор имеющийся у каждого игрока, обозначающий номер аккаунта на платформе Steam |
| name | имя игрока на игровом сервере |
| message | сообщение игрока, автоматически профильтрованное плагином |
| channel | канал чата. есть общий канал, командный, клановый, карточный |
| team | команда в зависимости от канала, означает разное. в командном чате - номер команды, в клановом - номер клана, в карточном - сетевой номер карточного стола. |
| time | время когда было отправлено сообщение |

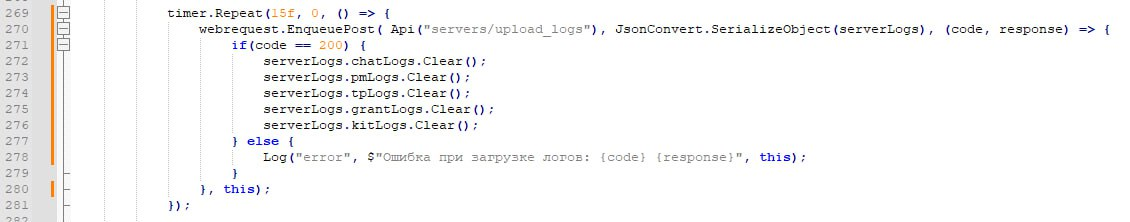
(рисунок 4) Вырезка из конфигурации плагина “DataBase”

Это функция добавления сообщения чата в логи, тут собирается вся вышеописанная информация.

(рисунок 5) Вырезка из конфигурации плагина “DataBase”

Здесь вышеописанная функция вызывается.

Метод OnPlayerChat вызывается внутриигровой функцией модификации Oxide, когда игрок отправляет какое либо сообщение

(рисунок 6) Вырезка из конфигурации плагина “DataBase”

Это таймер, который каждые 15 секунд отправляет все накопившиеся логи на сайт. если код запроса 200, то есть успешный, то логи очищаются.

Если нет, то в файлах сервера логируем ошибку и логи не очищаются, чтобы они загрузились при следующем срабатывании таймера.

## 2.2 Описание структуры базы данных

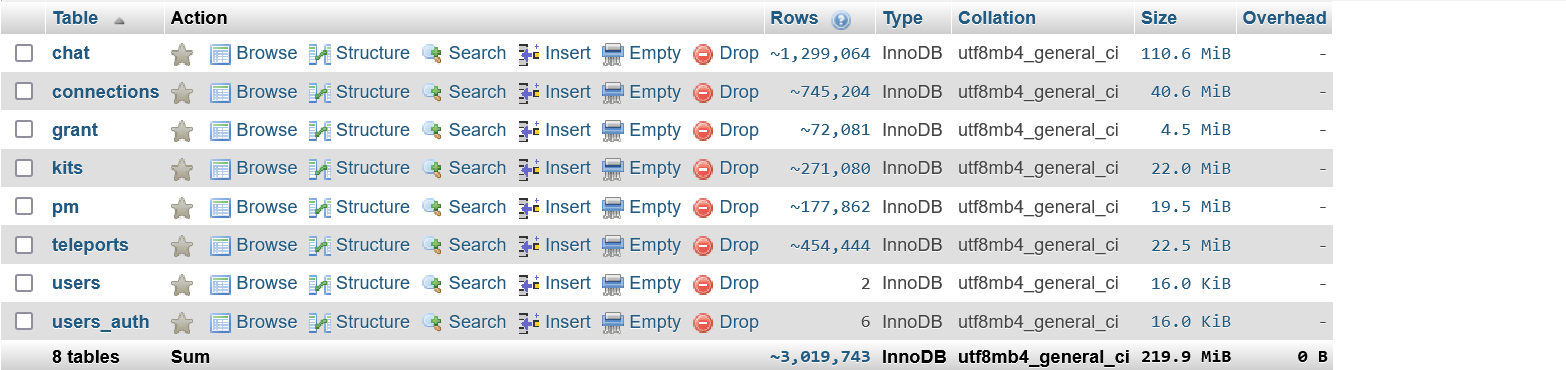
(рисунок 7) Вырезка из структуры базы данных

Таблица 2- Разбор вырезки из структуры базы данных

| Таблица | Описание |
| --- | --- |
| chat | собранная со всех серверов история сообщений отправленных в общую переписку |
| connection | собранная со всех серверов информация о подключениях и отключениях от сервера |
| grant | собранная со всех серверов информация о активировании услуг приобретаемых в магазине заказчика |
| kits | собранная со всех серверов информация о получении/активации наборов наборов предметов приобретаемых в магазине заказчика |
| pm | собранная со всех серверов информация о сообщениях отправленных в личную переписку |
| teleports | собранная со всех серверов информация о телепортациях одних игроков к другим |
| users | список сотрудников допущенных к пользованию лог-анализатору и уровней их допуска к его функциям |
| users\_auth | журнал сессий на веб-сервисе |

(рисунок 8) Вырезка из таблицы истории сообщений в общем чате

Таблица 3- Разбор вырезки из таблицы истории сообщений в общем чате

| Таблица | Описание |
| --- | --- |
| id | порядковый номер сообщения в базе данных |
| server | сервер с которого были отправлены логи |
| steamid | уникальный идентификатор аккаунта игрока на платформе Steam |
| name | имя игрока на момент написания сообщения |
| message | сообщение игрока |
| channel | канал в который было отправлено сообщение в зависимости от канала, означает разное. в командном чате - номер команды, в клановом - номер клана, в карточном - сетевой номер карточного стола. |
| team | айди команды в которую было направлено сообщение |
| time | время в формате UNIX time |

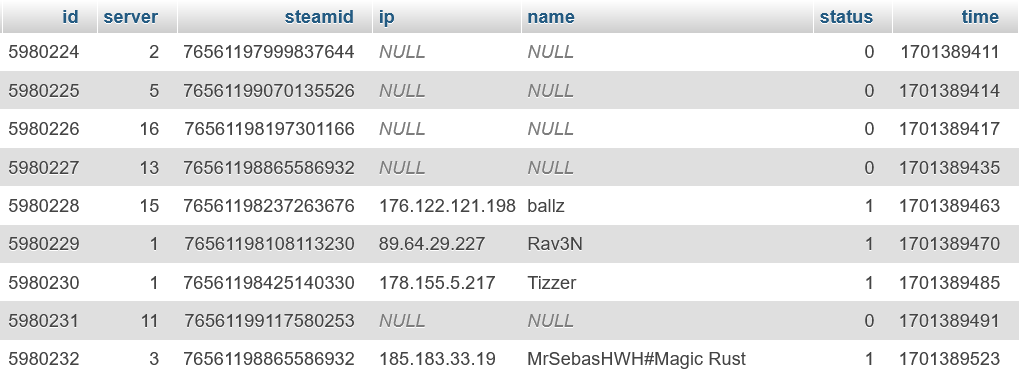
(рисунок 9) Вырезка из таблицы подключений к серверам

Таблица 4- Разбор вырезки из таблицы подключений к серверам

| Таблица | Описание |
| --- | --- |
| id | порядковый номер подключения в базе данных |
| server | сервер на который было совершено подключение |
| steamid | уникальный идентификатор аккаунта игрока на платформе Steam |
| ip | ip адрес игрока совершившего подключение |
| name | имя игрока использованное на игровом сервере |
| status | 0 - отключился, 1 - подключился |
| time | время в формате UNIX time |

(рисунок 10) Вырезка из таблицы активации услуг

Таблица 5- Разбор вырезки из таблицы активации услуг

| Таблица | Описание |
| --- | --- |
| id | порядковый номер лога |
| server | сервер на который была активация услуги |
| steamid | уникальный идентификатор аккаунта игрока на платформе Steam |
| service | внутреннее название услуги в магазине заказчика |
| isgroup | если true, значит это комплектная услуга (содержащая в себе несколько других). если false, то это одиночная самостоятельная услуга. |
| seconds | длительность услуги в количестве секунд |
| time | время в формате UNIX time |

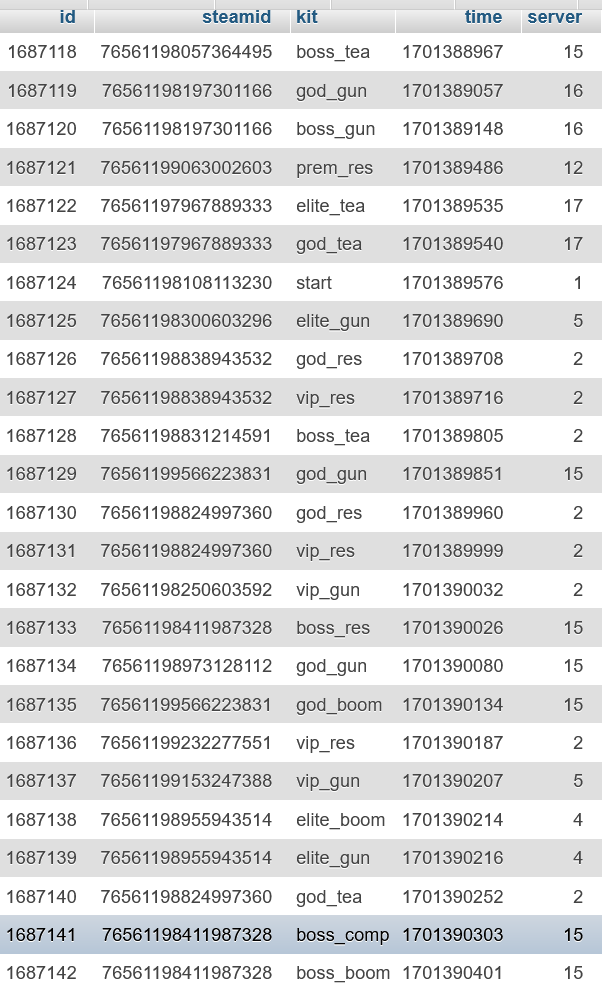
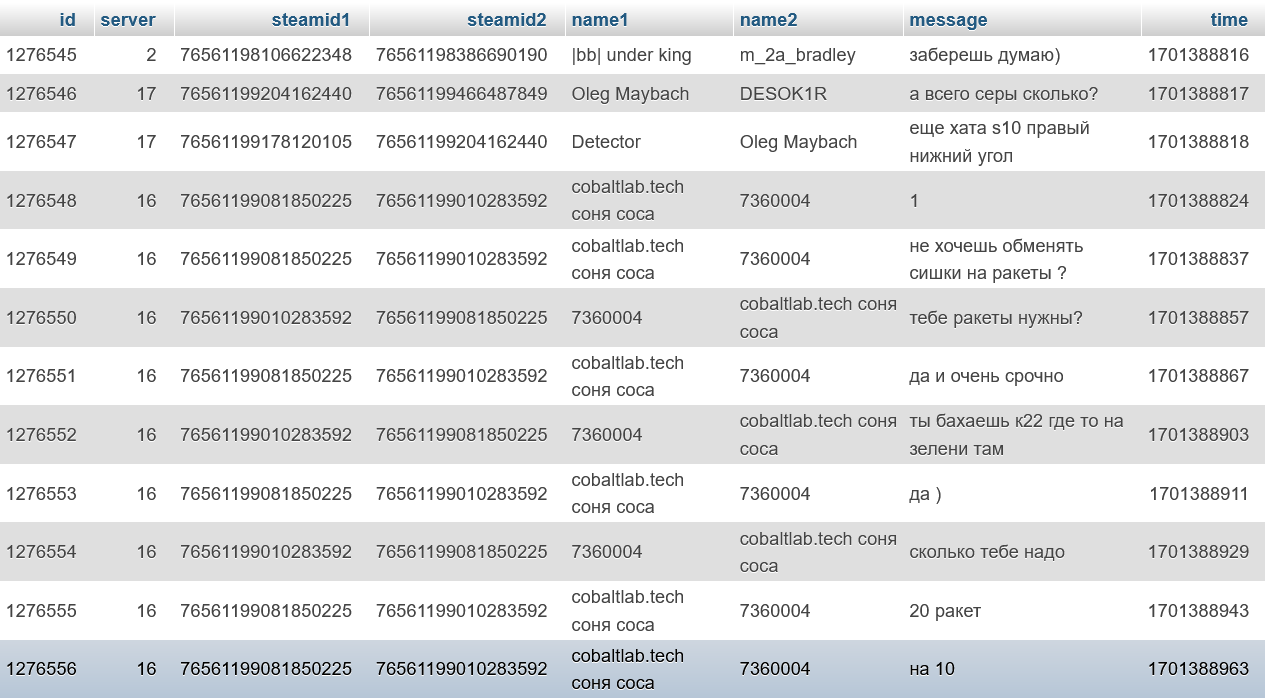
(рисунок 11) Вырезка из таблицы активации наборов

Таблица 6 - Разбор вырезки из таблицы активации наборов

| Таблица | Описание |
| --- | --- |
| id | порядковый номер лога |
| steamid | уникальный идентификатор аккаунта игрока на платформе Steam |
| kit | внутреннее название набора предметов (кита) в магазине заказчика |
| time | время открытие набора в формате UNIX time |
| server | сервер на котором было совершено открытие набора |



(рисунок 12) Вырезка из таблицы истории сообщений личного чата

Таблица 7- Разбор вырезки из таблицы истории сообщений личного чата

| Таблица | Описание |
| --- | --- |
| id | порядковый номер сообщения в базе данных |
| server | сервер на котором было послано сообщение |
| steamid | уникальный идентификатор аккаунта игрока на платформе Steam, который отправлял сообщение |
| steamid2 | уникальный идентификатор аккаунта игрока на платформе Steam, который получал сообщение |
| name | имя игрока, который отправлял сообщение, использованное на игровом сервере |
| name2 | имя игрока, который получал сообщение, использованное на игровом сервере |
| message | сообщение игрока отправленное в личную переписку |
| time | время в формате UNIX timе |

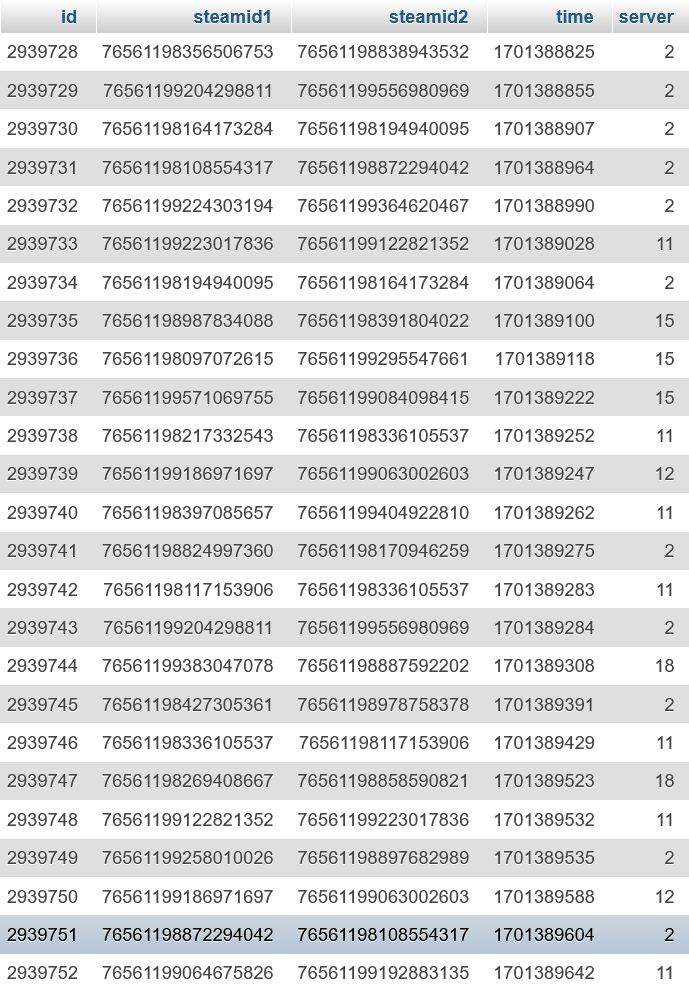
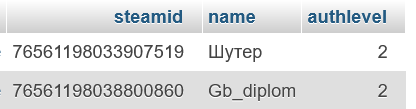
(рисунок 13) Вырезка из таблицы телепортаций 

Таблица 8 - Разбор вырезки из таблицы телепортаций

| Таблица | Описание |
| --- | --- |
| id | порядковый номер лога телепортаций |
| steamid | уникальный идентификатор аккаунта игрока на платформе Steam, который телепортируется ко второму |
| steamid2 | уникальный идентификатор аккаунта игрока на платформе Steam, к которому телепортируется первый |
| time | время в формате UNIX time |
| server | сервер с которого пришёл лог о телепортации |

(рисунок 14) Вырезка из таблицы сотрудников

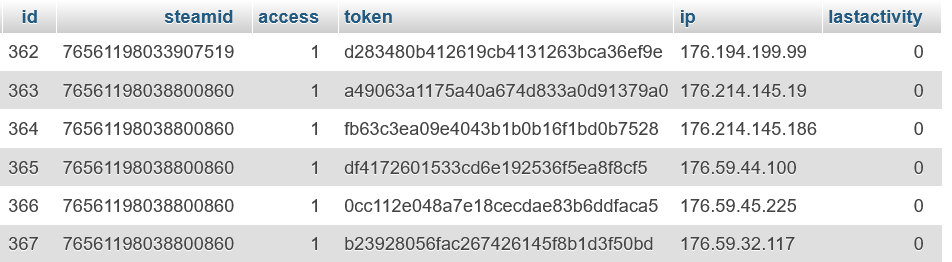
(рисунок 15) Вырезка из таблицы сессий на веб сервисе

Таблица 9 - Разбор вырезки из таблицы сессий на веб сервисе

| Поле | Описание |
| --- | --- |
| id | идентификатор строки автоинкремент |
| steamid | уникальный идентификатор аккаунта сотрудника на платформе Steam |
| access | Уровень доступа, который планировался, но не реализовался. |
| token | уникальный авторизационный токен. хранится в куках пользователя и сравнивается при каждом запросе. |
| ip | ай-пи адрес с которого был произведён вход |
| lastactivity | Должно было показывать, когда от работника была последняя активность, но применение этому не нашлось. Остаток от заброшенных идей. |

## 2.4 Настройка аутентификации с помощью сервиса Steam

Было важно сделать аутентификацию через сервис Steam, так как проект делается для игрового проекта, в котором везде и всё работает через Steam и SteamID.

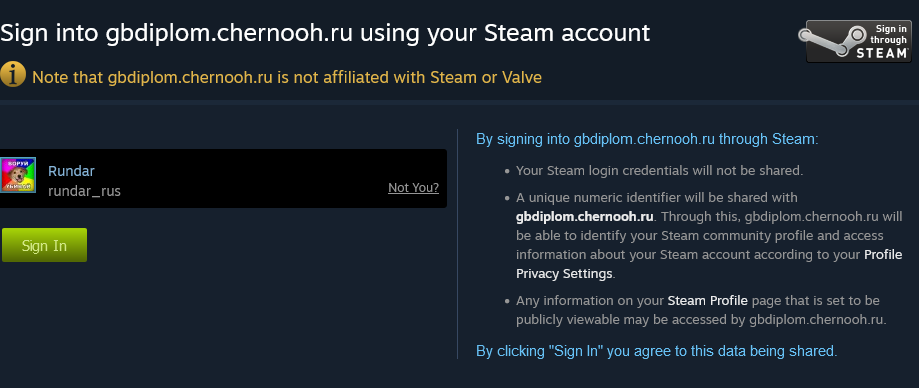
Мы будем использовать PHP-класс SteamAuth который был предоставлен Заказчиком. Это готовое решение по авторизации, работающее по стандарту OpenID и класс PHP, который используется для взаимодействия с аутентификацией OpenID.

Это все довольно просто. Сначала он включает скачанный нами класс SteamAuth (в который также входит openid.php).

Если сайт не обнаружил у пользователя активной сессии, то ему покажут кнопку "Войти через Steam":

(рисунок 16) Скриншот кнопки авторизации на сайте лог-анализатора

Так как сайт сделан для частного использования и не предназначен для третьих лиц, без авторизации он отображает только кнопку "Войти через Steam" и ничего больше.

(рисунок 17) Скриншот перехода на сайт сервиса Steam с проверенной системой безопасности и двухфакторной аутентификацией

Авторизация работает по стандарту OpenID. При нажатии на кнопку "Войти через Steam", пользователя перенаправляет на страницу Steam, где просят либо ввести данные для авторизации, либо сразу подтвердить вход кнопкой "Войти"

Подтвердив вход, Steam передаёт сайту SteamID пользователя, после чего происходит проверка прав из таблицы users в базе данных. Если пользователь присутствует и имеет достаточные права, сайт открывается в полноценном виде, а в таблицу users\_auth в базе данных создаётся новая запись, в которой указан SteamID, сгенерированный токен, ip адрес и уровень доступа. Токен также записывается в куки пользователя для последующей проверки авторизации при каждом запросе.

Если пользователь отсутствует или не имеет права, его перенаправит снова на пустую страницу с кнопкой "Войти через Steam".

## 2.5 Инициализация сайта

После прохождения аутентификации на странице пользователя прогрузятся данные сайта, а затем будут отрисованы контейнеры с контентом и навигационными панелями, а для этого:

# 

(рисунок 18) Вырезка из script.js

Функция которая даёт отрисовать страницу с интерактивными панелями только после полной прогрузки сайта

(рисунок 19) Вырезка из script.js

Для этого сначала подгружаются настройки



(рисунок 20) Вырезка из script.js

Которые до этого были сохранены этими функциями



(рисунок 21) Вырезка из script.js

Затем подготавливается меню выбора темы сайта

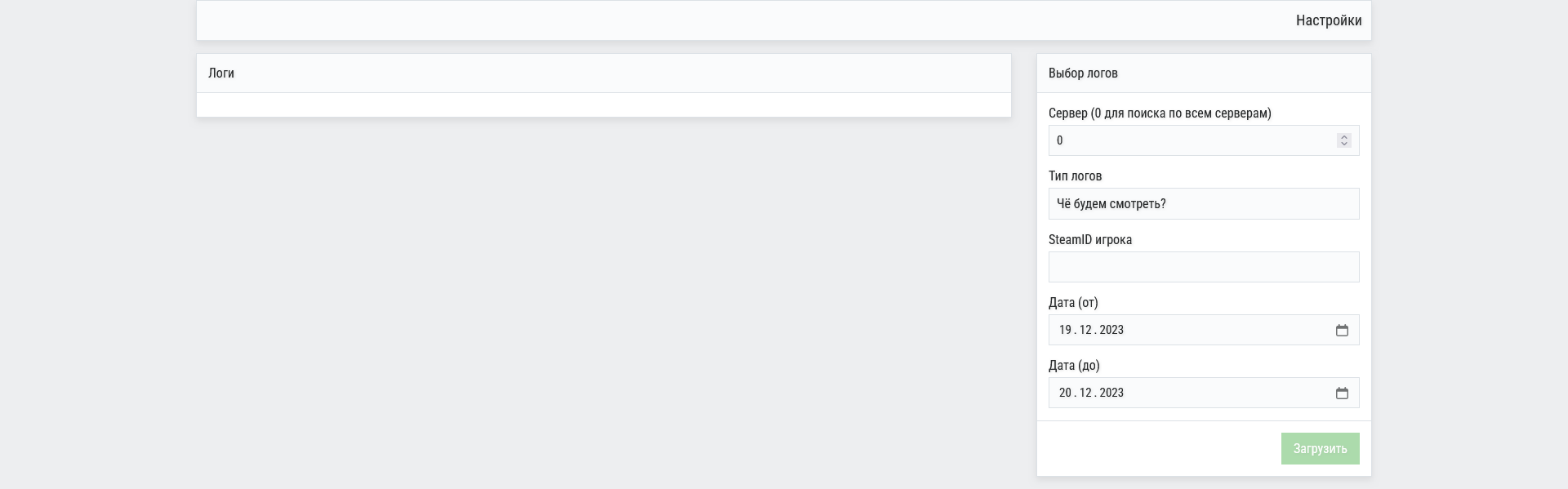


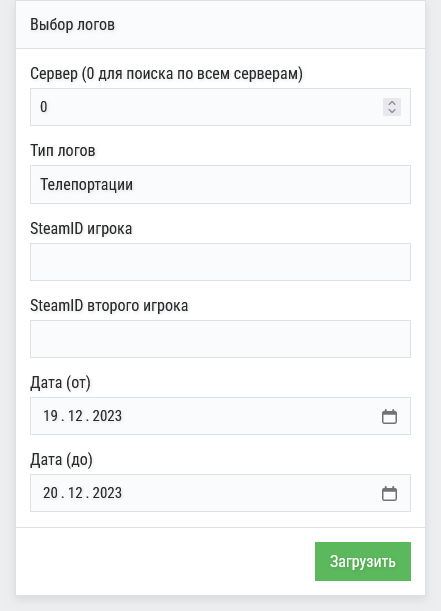
(рисунок 22) Вырезка из script.js

И в конце функции подгрузки вызывается вызов главного окна вывода логов и панели ввода настроек запроса в базу данных для вышеописанного окна

## 2.6 Работа с сайтом с точки зрения бекэнда

После построения сайта, он полностью готов к началу работы.

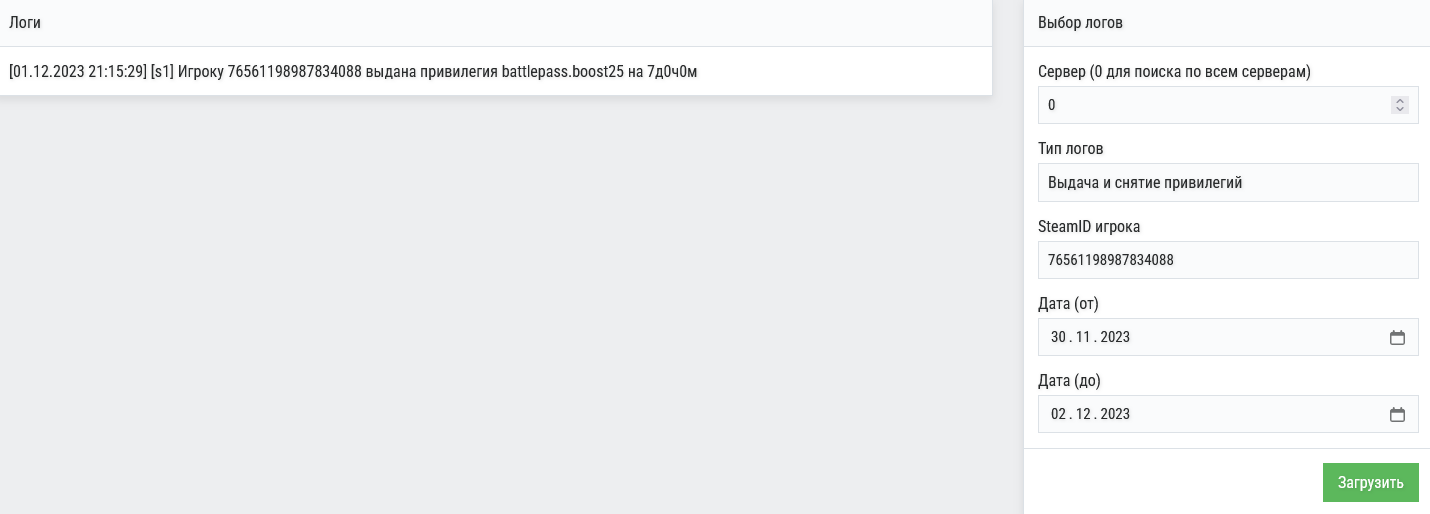


(рисунок 23) Скриншот сайта сразу после первого успешного входа

# 

(рисунок 24) Скриншот меню после выбора категории для которой нужны уточняющие данные и вырезка из script.js функции которая добавляет поля

в контейнер

(рисунок 25) Скриншот контекстного контейнера после составления и отправки запроса

# 

# (рисунок 26) Скриншот GET запроса с логом приведенным в читаемый вид выше

## 

## 2.7 Отслеживание ошибок на странице сайта

Для удобного отслеживания ошибок редактора был добавлен перехват консольных ошибок и их вывод прямо на сайте среди уведомлений (Рисунок 27). В ходе тестирования выяснилось, что ошибки иногда возникают в тех ситуациях, где нет никакого влияния на конечный результат, а значит без вывода ошибок было бы невозможно догадаться, что в работе анализатора есть какие-то сбои.

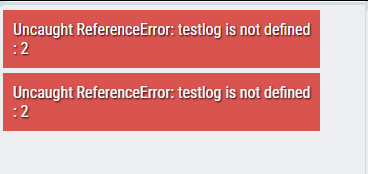
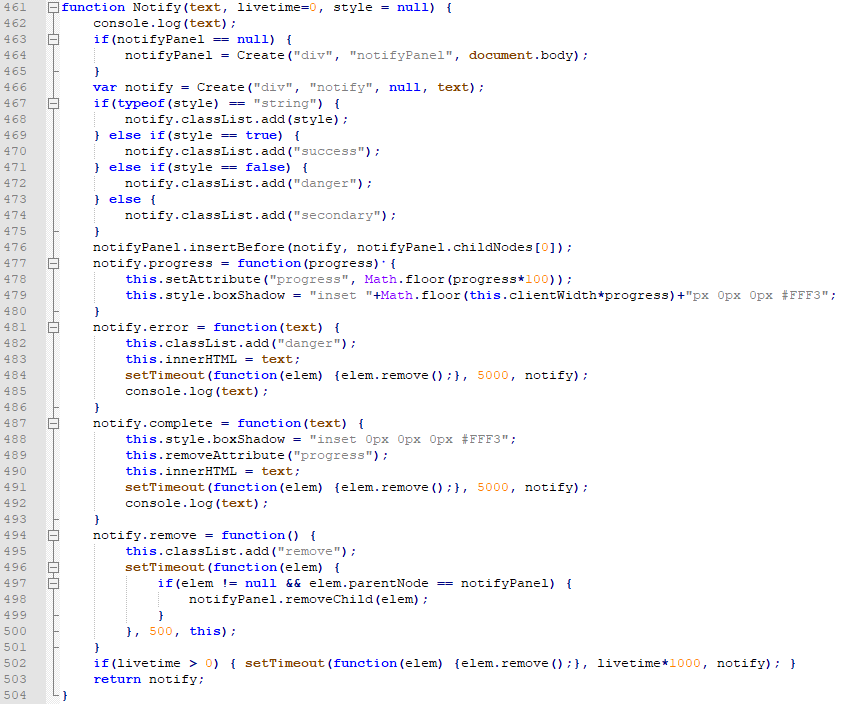


Рисунок 27- Вывод ошибки в уведомлениях сайта



(рисунок 28) Вырезка из script.js

Это функция которая вызывает окошко уведомления при ошибке на сайте, даже если она не была замечена пользователем.



(рисунок 29) Вырезка из script.ui.js

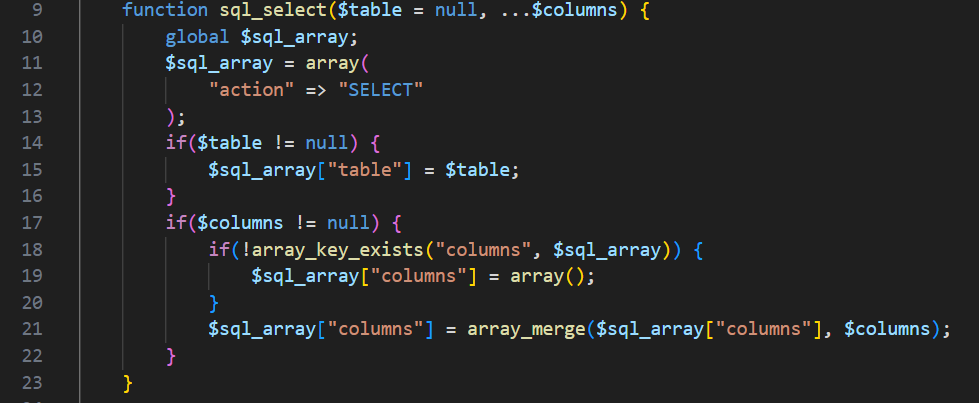
Эта функция для выбрасывания панели уведомлений, также она отслеживает положение уже вызванных уведомлений и располагает их одно за одним в порядке поступления и времени пропадания.

## 

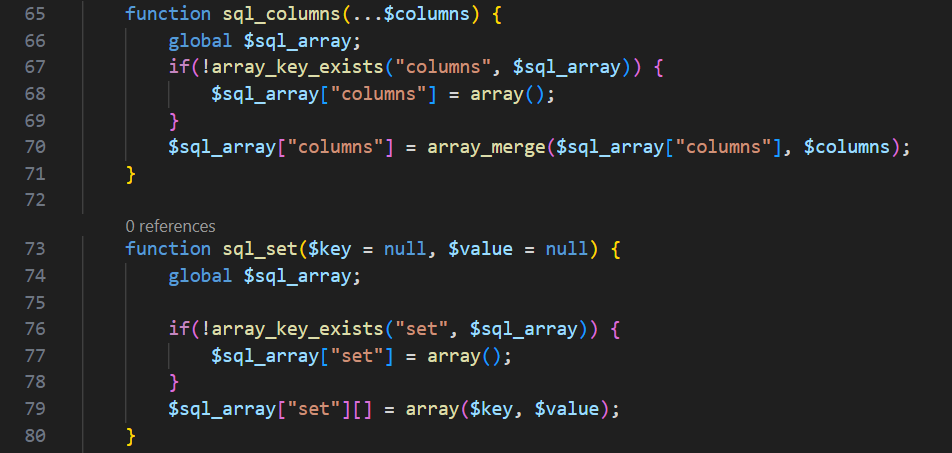
## 2.8 Создание своего фреймворка mysql.php

Для большего понимания внутренней работы сайта было решено написать свои функции запросов к базе данных, а также свой парсинг полученных Jsonов

В этой части подробно разобраны исключительно методы составления SELECT запросов так как функционал сайта создан в первую очередь для быстрого доступа к логам базы данных без необходимости вручную составлять SQL запрос и не давать доступ к системе администрирования из за опасности возможности вносить изменения и добавления в БД даже если и не со злым умыслом так для предотвращения возможности её повредить.

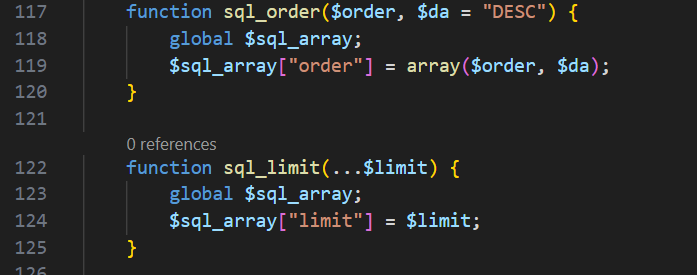
(рисунок 30) Вырезка из mysql.php метод sql\_select

Функция sql\_select инициализируют процесс создания запроса. в функции можно в качестве первого аргумента указать название таблицы, с которой будем работать.

(рисунок 31) Вырезка из mysql.php методы sql\_columns и sql\_set 

Метод sql\_columns используется в select для указания, какие столбцы нужно вернуть.

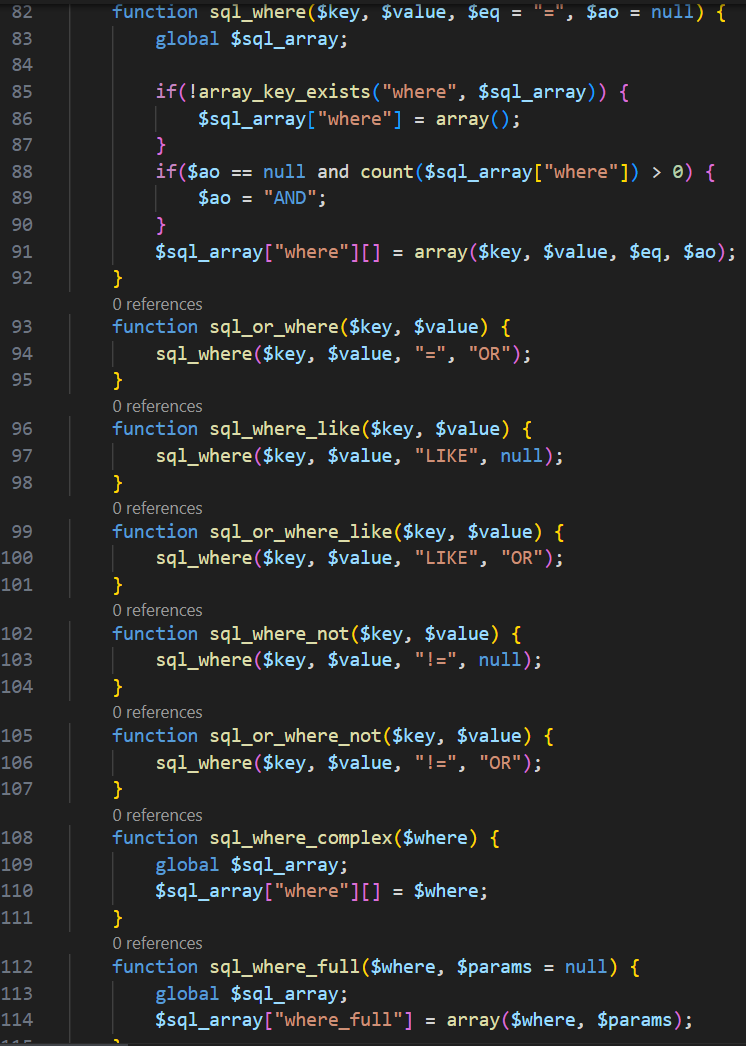
Метод sql\_set позволяет задать значение столбцу.



(рисунок 32) Вырезка из mysql.php метод sql\_order и sql\_limit

Метод sql\_order используется для определения нисходящего порядка выдачи.

Метод sql\_limit используется для выставления отсечки выдачи запроса.

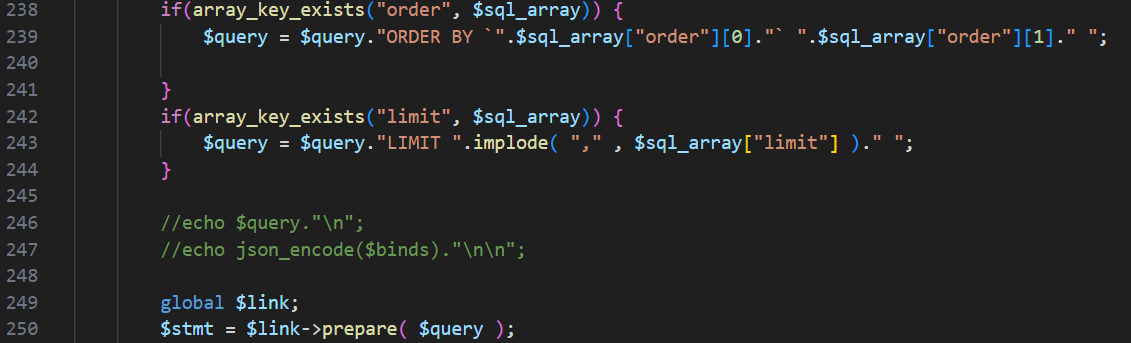
(рисунок 33) Вырезка из mysql.php методы where

Здесь представлены множественные разновидности where для уточнения запроса

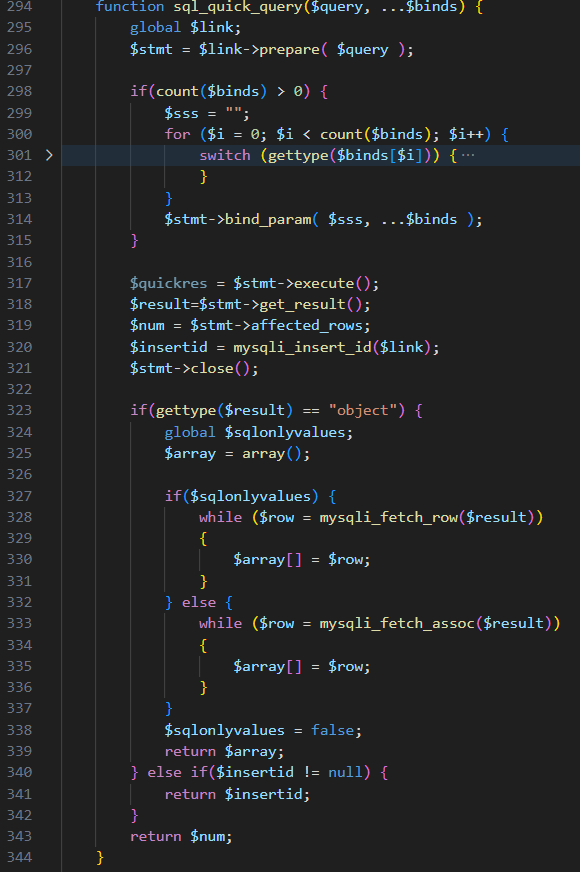
(рисунок 34) Вырезка из mysql.php часть метода sql\_execute

Функция, которая собирает и обрабатывает все накопившиеся данные после прошлых функций, формирует запрос и выполняет.

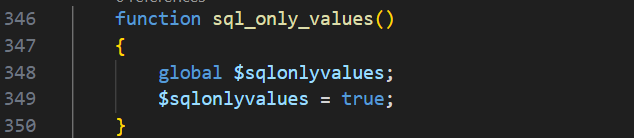
Варианты кроме SELECT скрыты поскольку не используются в текущей версии

(рисунок 35) Вырезка из mysql.php часть метода sql\_execute

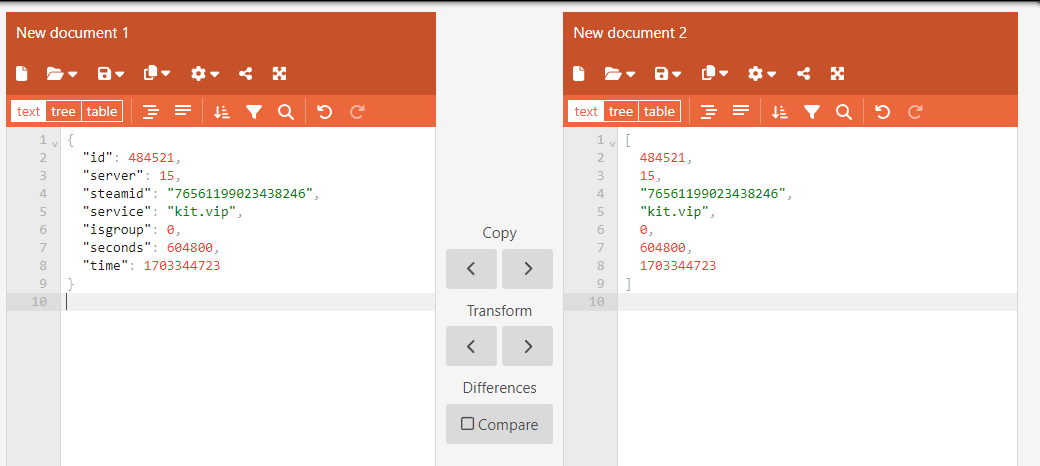
Для предотвращения злонамеренных инъекций в sql запрос он не формируется,а подготавливается в ожидаемой форме.

(рисунок 36) Вырезка из mysql.php метод sql\_quick\_query

Упрощенный метод создания запроса с малым количеством уточнений.

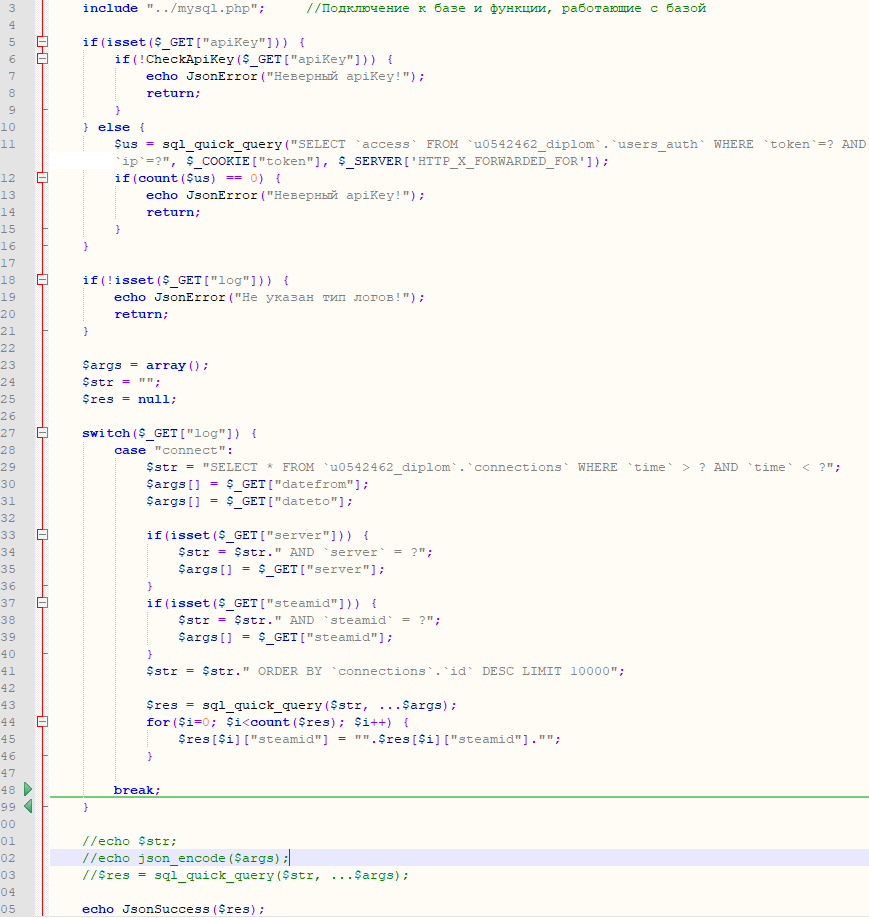
(рисунок 37) Вырезка из mysql.php функция sql\_only\_values

Функция для того чтобы облегчить Json для выдачи клиенту, обрезая названия полей как оно видно здесь :



(рисунок 38) Вырезка из Json то как информация приходит в необрезанном виде и то как информация обрезается для загрузки большого запроса

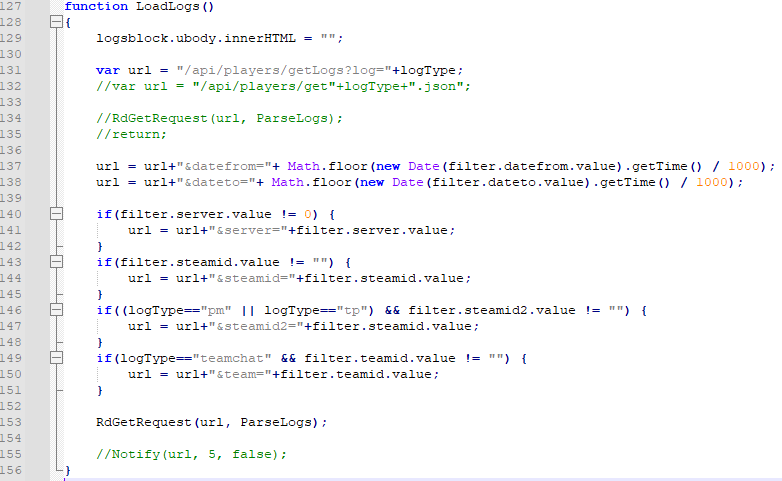
## 2.9 Описание класса GetLogs



(рисунок 39) Вырезка из getLogs

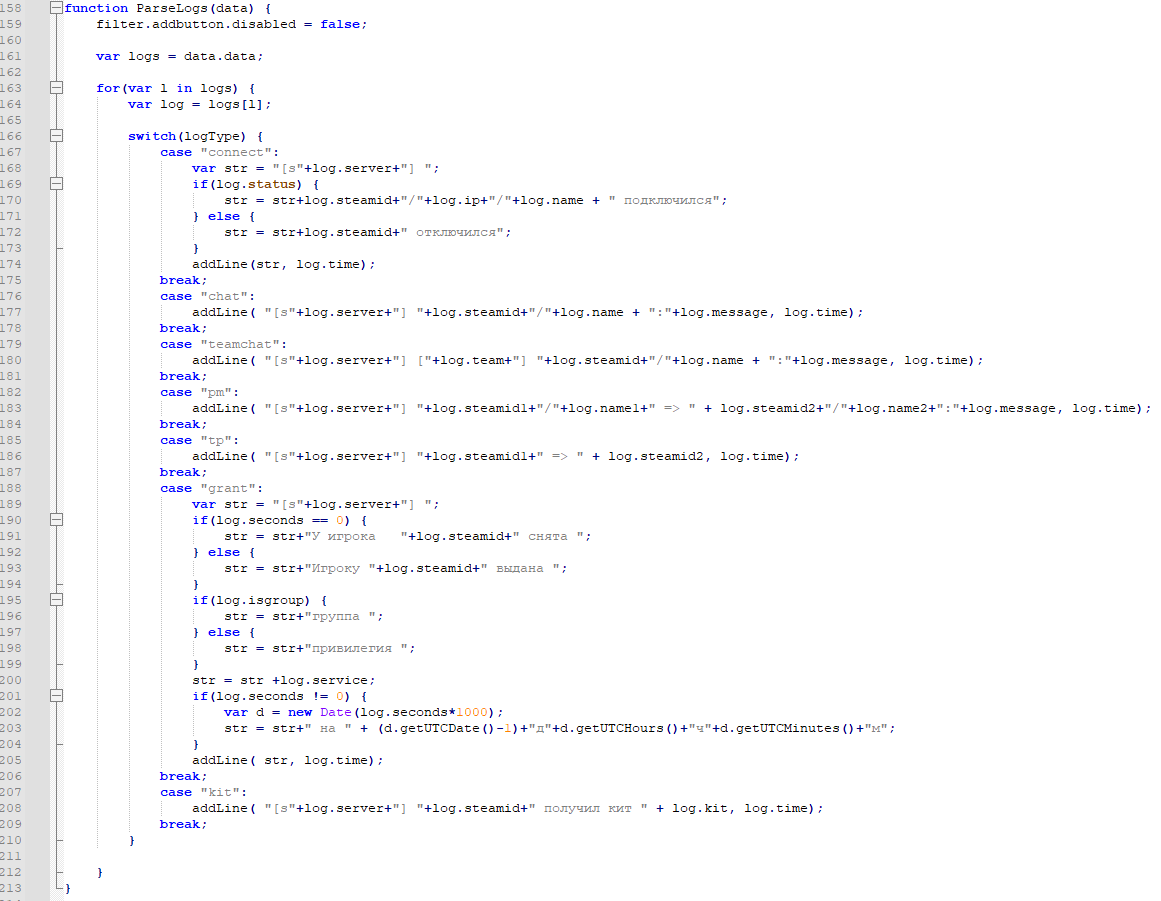
Здесь представлены switch-case для построения запроса

## 2.10 Описание сайтового фреймворка



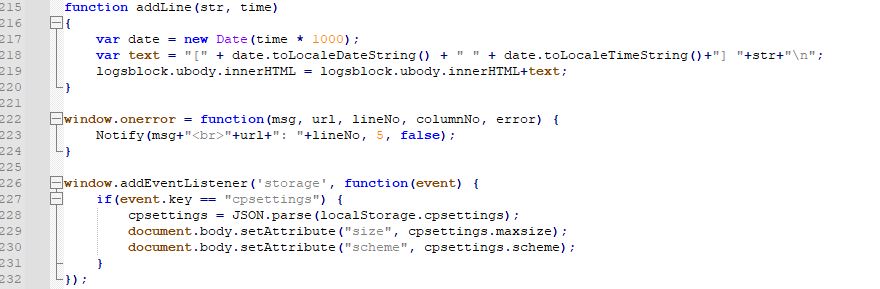
(рисунок 40) Вырезка из script.js

Это функция активируется при нажатии кнопки “Загрузить” на сайте, оно берёт данные введенные в боковое меню ранее и направляет их в метод запроса к базе данных.



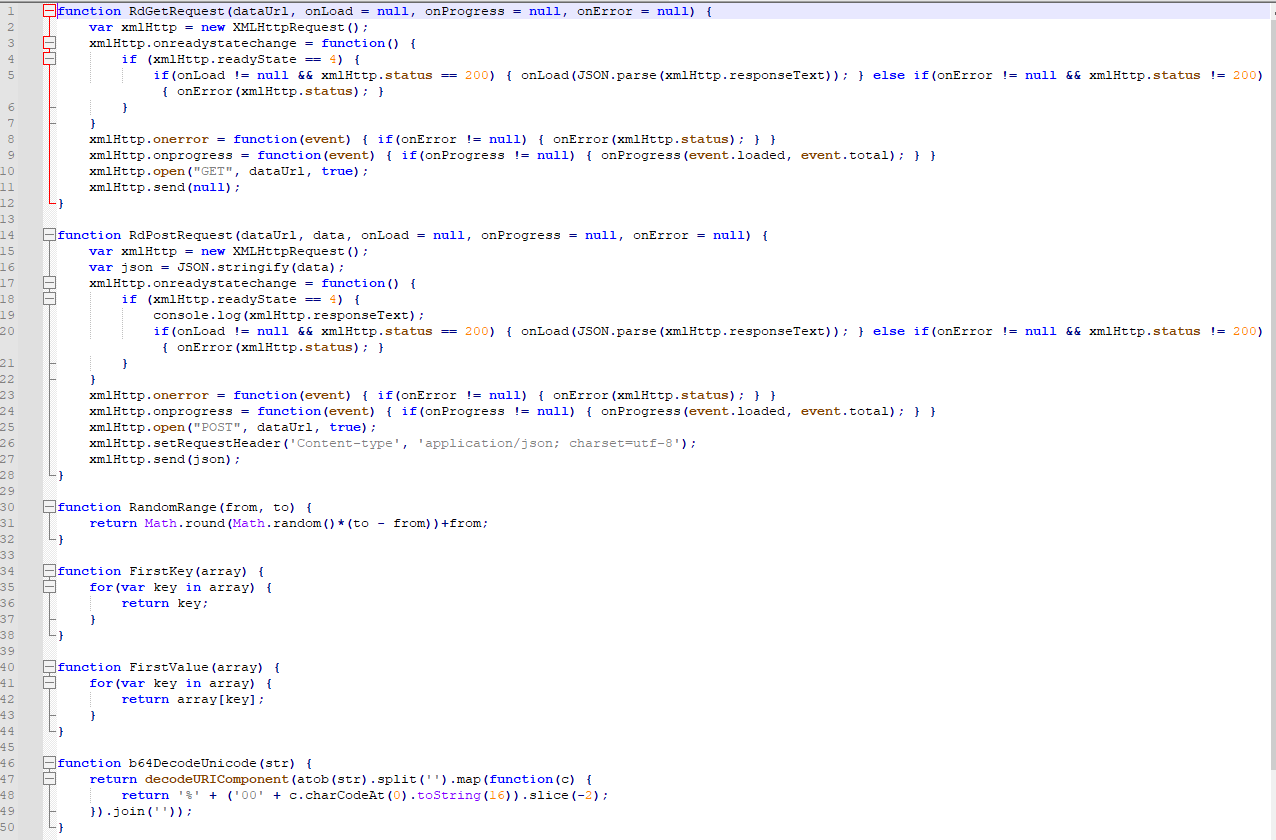
(рисунок 41) Вырезка из script.js

ParseLogs парсит полученные JSON для придачи читаемого вида в рабочем окне.



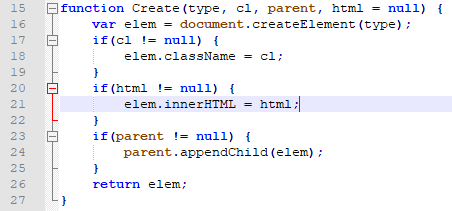
(рисунок 42) Вырезка из script.js

addLine добавляет строку лога в вывод на главном окне, в результате большой выдачи получается одна очень большая строка на все результаты вывода, что позволяет вывести много результатов, если же иначе для каждого результата выделять по отдельному объекту то сайт быстро сталкивается с проблемами быстродействия и работоспособности.



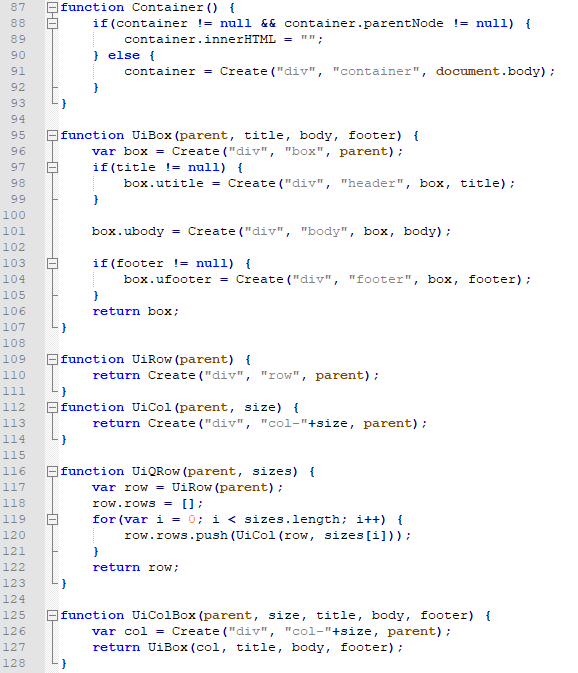
(рисунок 43) Вырезка из script.helper.js

Здесь представлены POST и GET запросы а также несколько утилитарных методов, приписка Rd означает RustDev - фреймворк для работы других внутренних проектов команды



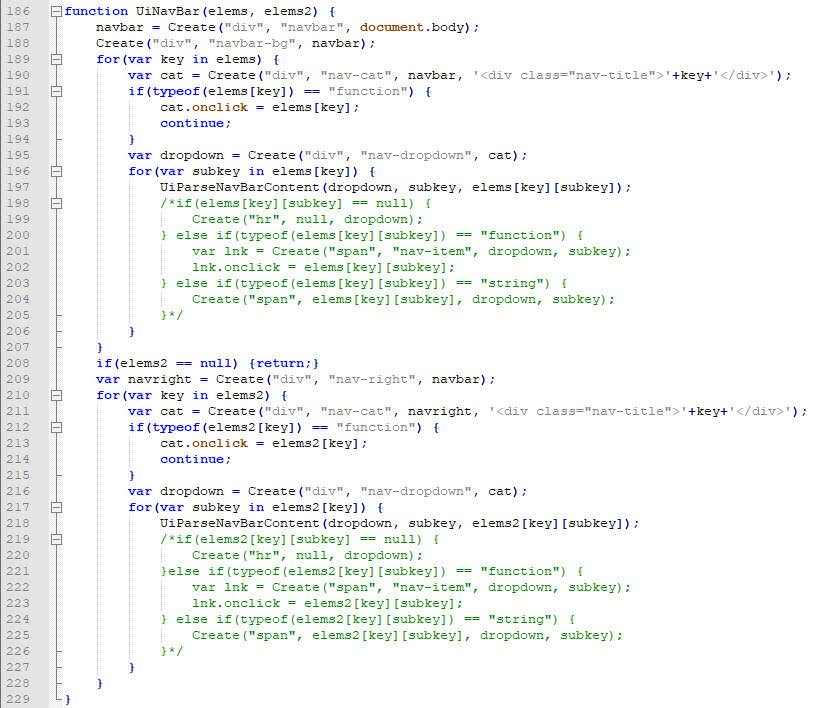
(рисунок 44) Вырезка из script.ui.js

Эта функция создаёт элемент указанного типа и класса,а также при необходимости встраивает его в контейнер-родитель



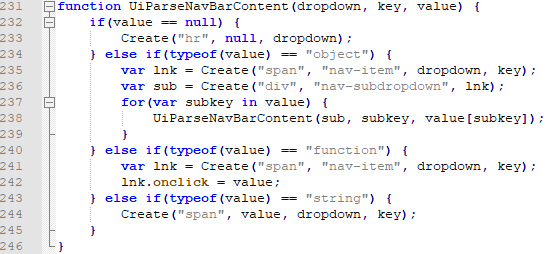
(рисунок 45) Вырезка из script.ui.js

Эти функции создают блоки. Созданы лишь для облегчения написания сайта, так как требуют меньше данных в аргументах.



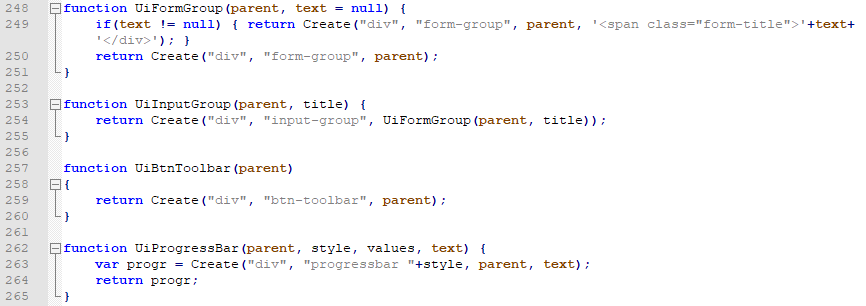
(рисунок 46) Вырезка из script.ui.js

Эта функция создаёт навигационное меню с выпадающими списками, такие как меню выбора цветовой схемы



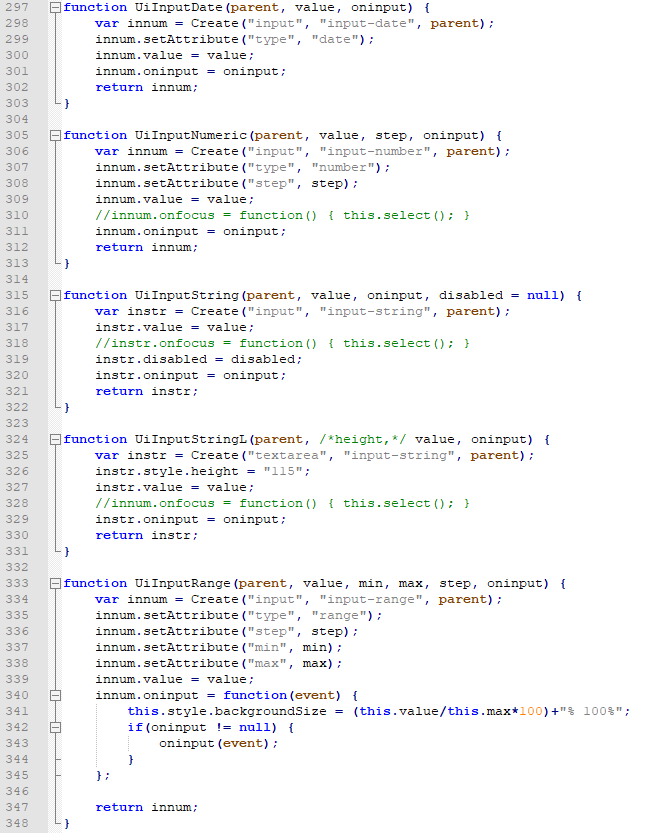
(рисунок 47) Вырезка из script.ui.js

Эта функция создана для построения контента выпадающего меню



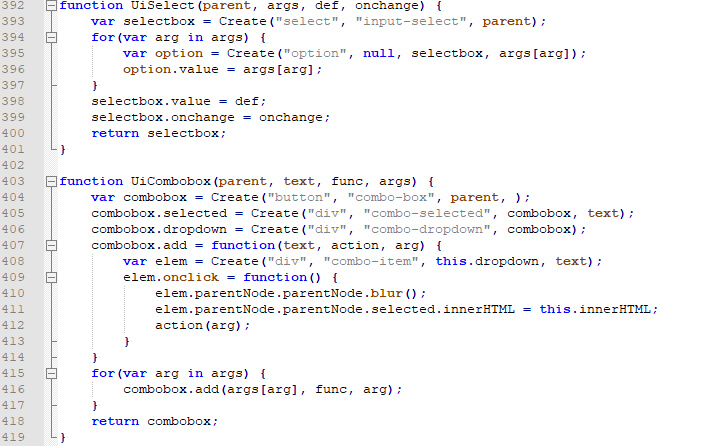
(рисунок 48) Вырезка из script.ui.js

Эти функции формируют такие группы как поля для ввода и кнопки, а также прогресс бары



(рисунок 49) Вырезка из script.ui.js

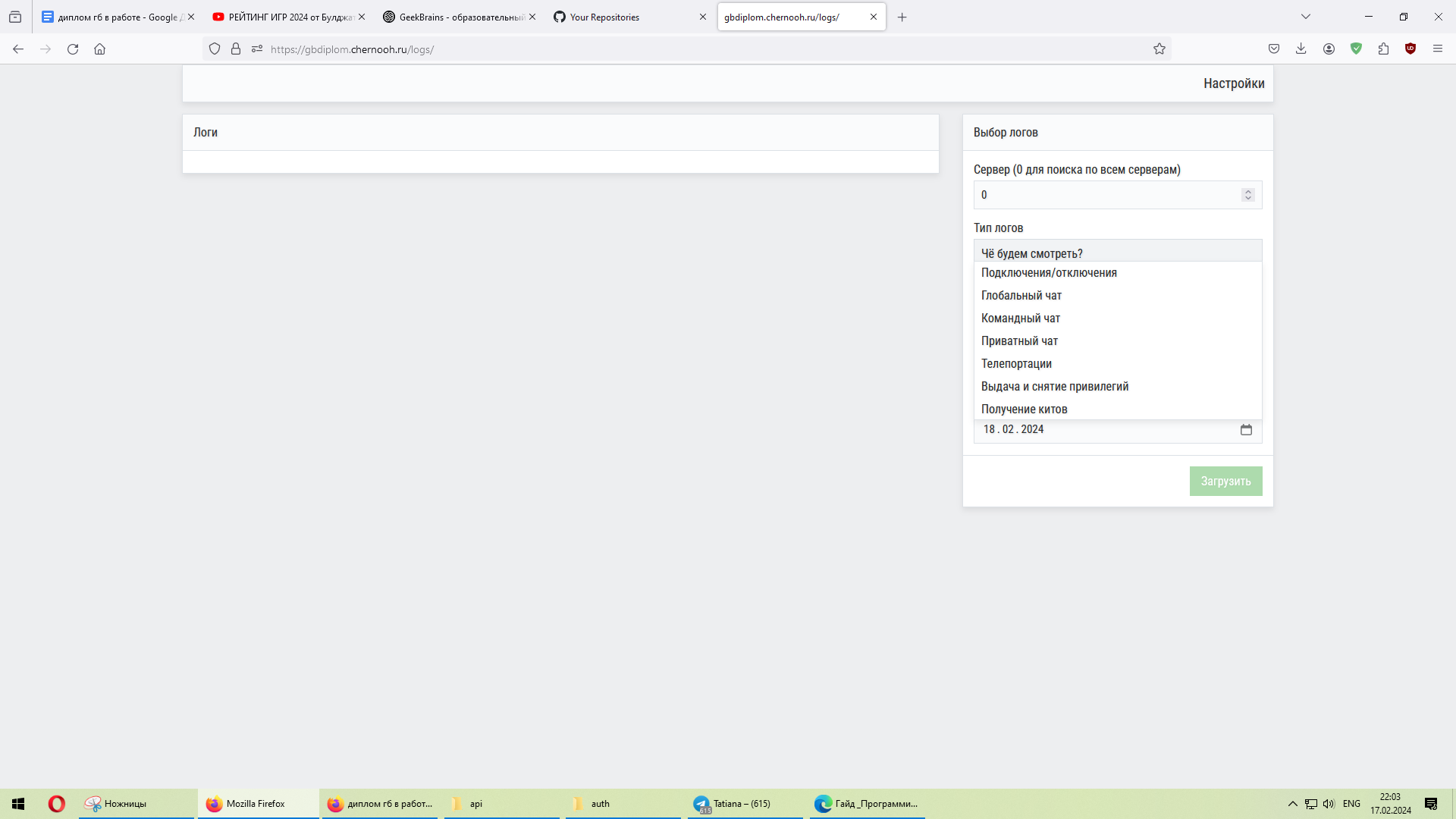
Эти функции для создания нужного типа данных ввода, не позволяющих вводить данные не того типа в поля ввода



(рисунок 50) Вырезка из script.ui.js

Эта функция создаёт выпадающие списки в панели опций.

Такие как на этом скриншоте:



# (рисунок 51) Скриншот рабочей панели сайта

# 3. Заключение

В рамках выпускной квалификационной работы создан вспомогательный инструментарий для разработчиков и администраторов игровых решений, конкретно в игре Rust.

В ходе выполнения выпускной квалификационной работы были решены следующие задачи.

* Был создан плагин для сбора и загрузки информации в БД
* Реализован лог-анализатор для разного рода записываемой информации
* Сайт был полностью протестирован на наличие критических проблем и неточностей

Использование разработанного инструментария упрощает процесс работы с модерирования серверов за счёт более удобного доступа к логам. Позволяет пользователям инструментария проводить мониторинг действий пользователей. А самое главное увеличивает продуктивность.

Инструментарий был протестирован заказчиком и полностью удовлетворяет его потребности. Вследствие чего, считаю, что выпускная квалификационная работа выполнена.

# Список использованной литературы и ресурсов

<https://www.w3schools.com/> - документация по всем указанным языкам

<https://www.php.net/manual/ru/langref.php> - документация по PHP

<https://learn.microsoft.com/ru-ru/dotnet/csharp/> - документация по C#

<https://developer.mozilla.org/ru/docs/Web/JavaScript> - документация по JS

<https://partner.steamgames.com/doc/features/auth> - Документация Steamworks Аутентификация и проверка владения

<https://dev.mysql.com/doc/refman/8.0/en/innodb-storage-engine.html> - документация InnoDB

<https://www.phpmyadmin.net/docs/> - документация PhpMyAdmin

<https://umod.org/documentation/games/rust> - документация модификации Oxide

<https://wiki.facepunch.com/rust/modding> - документация игры Rust

<https://www.telerik.com/products/decompiler.aspx> - декомпилятор кода игры