

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
Институт заочно-вечернего обучения

Допускаю к защите

Руководитель Ю. Р. Басиров
подпись, И. О. Фамилия

Разработка модуля автооплаты

Наименование темы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к лабораторной работе по дисциплине
«Технология программирования»

1.011.00.00 ПЗ

обозначение документа

Разработал студент группы ЭВМбз-16-1

подпись

А. А. Михиденко

И. О. Фамилия

Нормоконтроль

подпись

Ю. Р. Басиров

И. О. Фамилия

Лабораторная работа защищена с оценкой

Иркутск 2021

Содержание

1 Теоретический материал.....	3
2 Постановка задачи.....	4
3 Выполнение задания.....	5
Вывод.....	13

1 Теоретический материал

REST архитектура

Один из стилей архитектуры ПО для распределенных систем. Термин был определен в 2000г. одним из авторов протокола HTTP, Роем Филднигом.

Данная архитектура получила очень широкое распространение в построении сетевых служб (www, веб). Системы которые поддерживают REST - называются RESTfull-системами. В общем случае, REST является очень простым интерфейсом управления информацией. Каждая единица информации однозначно определяется глобальным идентификатором URL. В свою очередь каждая URL имеет строго заданный формат.

На сегодняшний день, в веб технологиях самым распространенным протоколом является HTTP, HTTP2 и его производные: такие как H2, поддерживающий соединение поверх TLS; H2C, поверх TCP. В тоже время HTTP простой текст, а H2 - это уже протокол бинарный.

Так вот, действия для HTTP задаются при помощи запросов, POST(добавить, заменить, удалить), GET(получить), PUT(добавить, заменить), DELETE(удалить).

Альтернативные платежные системы

Это инструменты для организации переводов цифрового эквивалента денежных средств, осуществляемые на основе имеющихся технологий современного веб-интернета.

Большинство современных платежных систем уже нашли широкое применение в потребительском секторе банковских услуг. За последние 10 лет, благодаря популяризации технологии ВС, платежные системы получили еще более широкое распространение. Постоянный рост качества и производительности в технологиях коммуникаций и носителях, обеспечили бум мобильных и веб приложений. Это позволило некоторым участникам венчурного капитализма перестроить привычные для нас понятия о банковских услугах. Появились услуги мгновенных платежей.

2 Постановка задачи

Задача: разработать модуль платежной системы.

Цель: автоматизировать процесс оплаты продукта.

Анализ задания

В качестве системы приема платежей была выбрана платформа qiwі, которая соответствует всем необходимым условиям для разработки современного веб-приложения с поддержкой автоматической оплаты.

Необходимые условия для подключения к системе:

1. Выделенный домен и действительный TSL сертификат
2. Подтверждения заявки от qiwі, на регистрацию в системе
3. Иметь действующие токены для авторизации на qiwі api

Поскольку процесс состоял из ручных операций: приема денежных средств, записи данных о покупателе и товаре, временных меток действия гарантии на товар и пр., было разработан алгоритм который полностью перевел процесс оплаты в автоматический режим.

3 Выполнение задания

И так. Имея на руках готовый движек, изучив его можно спокойно приступить к написанию своего модуля. Я нарочно упускаю описание ядра движка и все его нюансы, которые необходимо знать для того что бы заставить этот модуль работать.

Проектирование объекта информационной системы

Описание характеристик нотации:

- *Database* – база данных, в качестве хранения данных о заказе
- *Recaptcha v3* – внешняя API модуля системы защиты от спама
- *P2P Qiwi payments* – внешняя RESTfull API системы приема платежей

Согласно нотации EPC можно определить диаграмму, которая наглядно покажет схему работы алгоритма оплаты платежей (рис. 1).

Этапы информационных технологий:

Сбор информации о заказе. Выполняется при помощи веб-сайта, формы которого способны содержать в себе данные как из статического, так и из динамического набора ограниченного диапазона.

Обработка собранных данных. При подтверждении заказа, клиентский браузер передает данные веб-серверу на обработку, где производится проверка валидности токена внешней API модуля системы защиты от спама, затем обрабатываются данные покупателя и формируется уникальный номер заказа. После чего подготовленная структура данных отправляется внешней REST API системы приема платежей.

Сервер все еще ожидает ответа от этой системы, и по результатам её ответа выполняет переадресацию пользователя на страницу оплаты.

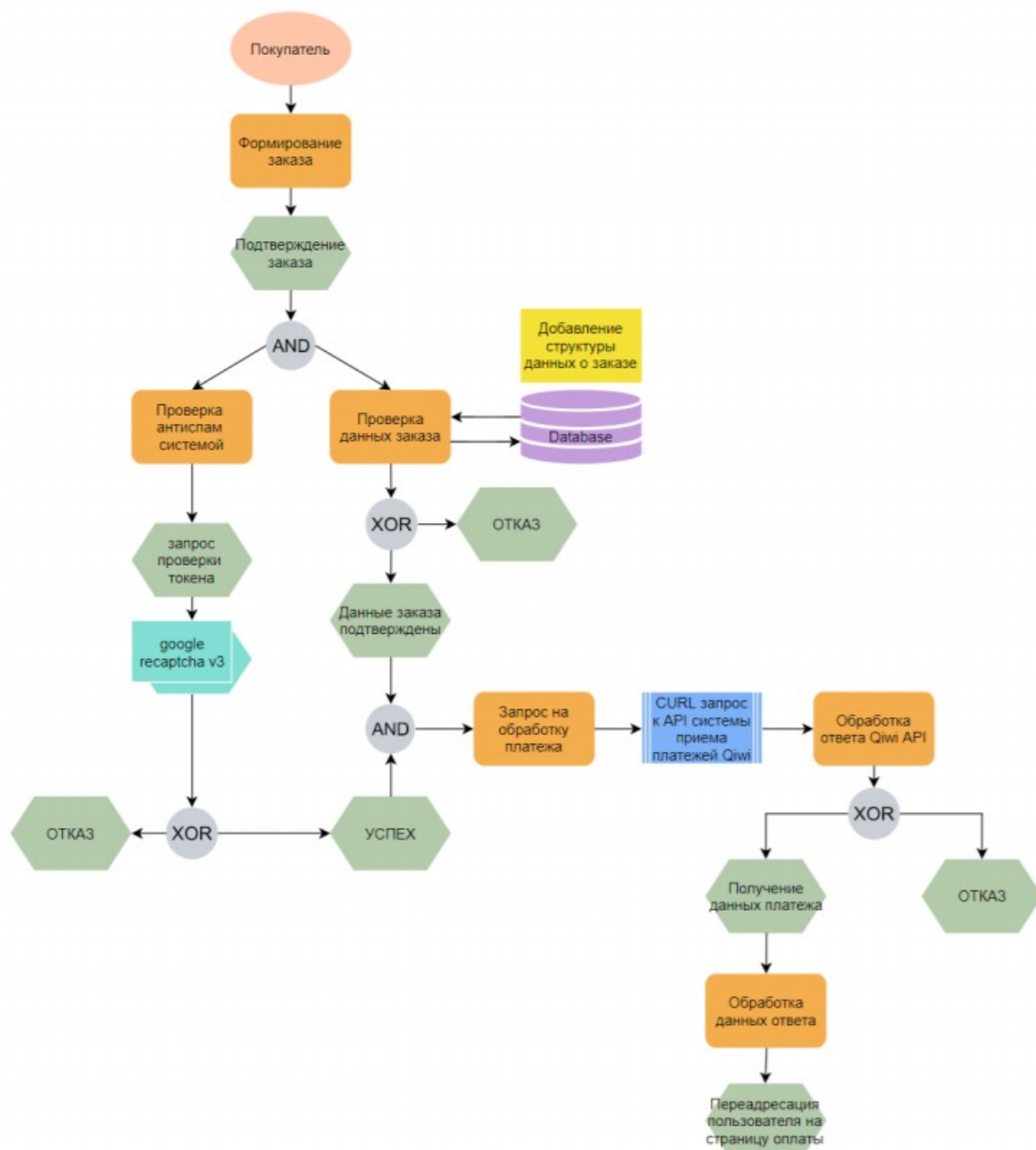


Рисунок 1 - Описание информационного процесса автоматического приема оплаты платежей в нотации EPC.

Пользовательский интерфейс

Демонстрация исходного кода пользовательского интерфейса не имеет смысла, по причине того что займет очень много места. В свою защиту могу сказать лишь только то, что при написании FE я придерживался стандартов разметки HTML, каскадных таблиц CSS и Native JS.

Внешний вид FE можно наблюдать на рисунке 2 и рисунке 3.

Привет, Alex Deroza

Введите стим/ссылку на профиль
STEAM_1:0:152514705

Дата активации
02.04.2021

Ваше мнение

1 дн. навсегда*

VIP ☐ Навсегда*

Подтвердить

Скидка: 0.00% Комиссия: +2% Qiwi

Итого к оплате:
1.02 RUB

2021 © this script was developed by that kitty
<https://github.com/M0st1ce>

Рисунок 2 - Мобильная версия шаблона

Привет, Alex Deroza

Если хотите сделать кому-либо ? подарок, тогда введите ссылку на его профиль* steam. Либо оставьте поле как-есть, если хотите сделать подарок ❤ себе ;) *0 том как получить ссылку на профиль читай здесь

STEAM_1:0:152514705

Дата активации 02.04.2021 Ваше мнение

1 дн. навсегда*

VIP ☐ Навсегда* Подтвердить

Скидка: 0.00% Комиссия: +2% Qiwi

Итого к оплате:
1.02 RUB

Рисунок 3 - Обычная версия шаблона

Структура БД

Для внешнего хранения данных была выбрана реляционная БД MariaDB. Фактически, структуру БД можно описать при помощи *dbdesinger* (рис. 4).

В системе игрового сервера, определение клиента происходит по внутреннему идентификатору *valve*, *steamid*. В свою очередь идентификаторы платежей со стороны *qiwi* необходимо хранить в формате *uuid*, это требования *qiwi*. По этому идентификатору *qiwi api* определяет для какого именно платежа необходимо произвести оплату.

Не вдаваясь в подробности структуры текущей версии таблиц БД, предполагается что эта модель позволит понять основную суть этой структуры.

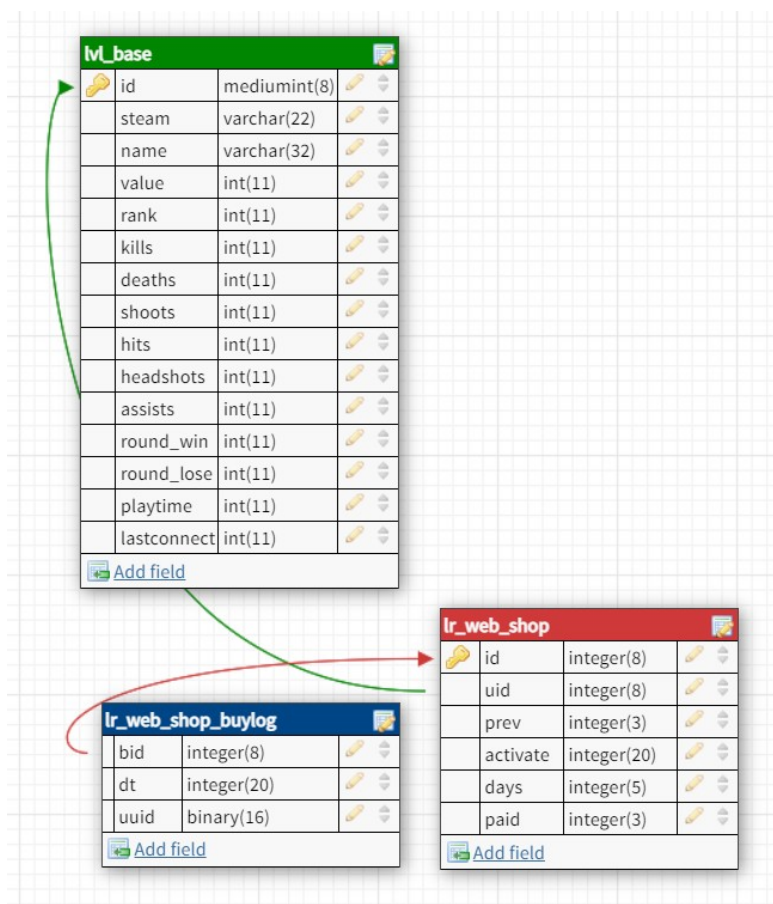


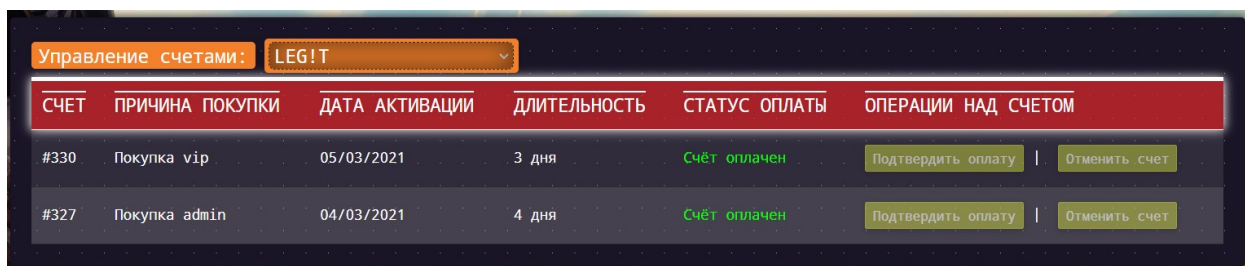
Рисунок 4 - Структура БД автооплаты

Структура пользовательского интерфейса

При реализации FE, для работы с DOM и асинхронной передачей данных, вызовами внешних систем, используется jQuery.

Страницы для работы с платежами:

1. */?page=shop*, страница выбора параметров оплаты(рис. 2, рис. 3)
2. */?page=shop&viewbills=1*, страница просмотра платежей(рис. 5)



The screenshot shows a web interface for managing accounts. At the top, there is a header with 'Управление счетами:' and a dropdown menu showing 'LEGIT'. Below this is a table with columns: 'СЧЕТ', 'ПРИЧИНА ПОКУПКИ', 'ДАТА АКТИВАЦИИ', 'ДЛИТЕЛЬНОСТЬ', 'СТАТУС ОПЛАТЫ', and 'ОПЕРАЦИИ НАД СЧЕТОМ'. The table contains two rows of data.

СЧЕТ	ПРИЧИНА ПОКУПКИ	ДАТА АКТИВАЦИИ	ДЛИТЕЛЬНОСТЬ	СТАТУС ОПЛАТЫ	ОПЕРАЦИИ НАД СЧЕТОМ
#330	Покупка vip	05/03/2021	3 дня	Счёт оплачен	Подтвердить оплату Отменить счет
#327	Покупка admin	04/03/2021	4 дня	Счёт оплачен	Подтвердить оплату Отменить счет

Рисунок 5 - Страница просмотра платежей

Методы управления счетами

Выпадающий список *управление счетами*, имеет работает в привилегированном режиме(доступны к просмотру все активные счета клиентов), что позволяет администратору контролировать статус платежей.

Для обычных пользователей этот режим недоступен, и они могут контролировать состояние только своих платежей.

Возможности контроля счетов включают:

1. *Подтверждение оплаты*. Вызывает метод проверки статуса оплаты. Если счет неоплаченный JS направляет(перенаправляет) клиента на страницу оплаты в системе *qiwi*.
2. *Отмена счета*. Позволяет отменить неоплаченный счет. Когда счет находится в статусе *просрочен*, - кнопка отмены счета устанавливается в состояние - *удаление счета*.
3. *Удаление счета*. Удалить счет из БД можно только когда он был отменен или счет из-за неуплаты оказывается просроченным. Отмена счета автоматически помещает счет в статус - *просрочен*.

Фрагменты кода класса Shop

Метод *GetPaymentStatusBySteamID* определяет состояние счета посредством обращения SQL запросом к БД. В запросе продемонстрирована логики выборки в результате абстрактного объединения трех таблиц.

```
private function GetPaymentStatusBySteamID($paymentid, $steamid, $is_admin = false) {
    if(!isset($paymentid) || !isset($steamid)) {
        return NULL;
    }
    $qparams = [
        'a' => $paymentid,
        'b' => $steamid
    ];
    return $this->Db->query('Core', 0, 0, '
        SELECT `a`.`id`, `b`.`bid`, UuidFromBin(`b`.`uuid`) AS `uuid`, `b`.`dt`, `c`.`days`, `c`.`paid`
        FROM `lvl_base` AS `a`
            INNER JOIN `lr_web_shop_buylog` AS `b`
            INNER JOIN `lr_web_shop` AS `c`
        WHERE `a`.`steam` LIKE :b ' . (!$is_admin ? 'AND `a`.`id` = `c`.`uid` ' : ''). 'AND `b`.`bid` = `c`.`id` AND `b`.`bid`
        = :a
        LIMIT 1;',
        $qparams); // is_admin ignore user bid owning
}
```

Рисунок 6 - Метод *GetPaymentStatusBySteamID*

На рисунке 7 показан фрагмент кода из метода *CancelBill*, который применяется для изменения статуса счета, т.е. его отмены на стороне *qiwi*.

```
$url = 'https://api.qiwi.com/partner/bill/v1/bills/' . $uuid . '/reject';
$ch = curl_init();
if($ch) {
    curl_setopt($ch, CURLOPT_USERAGENT, 'PHP Curl/1.9 (+https://github.com/php-mod/curl)');
    curl_setopt($ch, CURLOPT_HEADER_OUT, true);
    curl_setopt($ch, CURLOPT_HEADER, false);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    //curl_setopt($ch, CURLOPT_HEADERFUNCTION, array($ch, 'addResponseHeaderLine'));
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        'Authorization: Bearer ' . $this->ShowSecret(),
        'Accept: application/json',
        'Content-Type: application/json;charset=UTF-8'
    ]);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'POST');
    curl_setopt($ch, CURLOPT_TIMEOUT, 3);
    $result = curl_exec($ch);
    if(!$result) {
        //echo(curl_error($ch).', '.curl_errno($ch));
        die($this->RetInJson('Curl result error..'));
    }
    curl_close($ch);
}
$json = json_decode($result, true);
```

Рисунок 7 - Обращение к *QIWI API* (отмена счета)

При попытке добавлении счета в БД, в логике метода *AddNewPrivilege* необходимо сначала проверить на какого пользователя будет оформлен этот счет:

```
$uid = intval($this->Db->queryOneColumn(
    'Core', 0, 0,
    'SELECT `id` FROM `lvl_base` WHERE `steam` LIKE :a LIMIT 1;',
    $qparams)
);
```

если пользователь не существует, проверить его ключь *steamid* через внешней API ИС steampowered:

```
$response = file_get_contents(
    'http://api.steampowered.com/ISteamUser/GetPlayerSummaries/v0002/?key='
    . $_STEAMAPI . '&steamids=' . con_steam32to64($steam_id)
);
```

по результатам которой определится факт принадлежности пользователя и будет получена необходимая по нему информация. После чего можно добавить его в БД:

```
$qparams = [
    'a' => $steam_id,
    'b' => $json['response']['players'][0]['personaname']
];

$res = $this->Db->insertion(
    'LevelsRanks', 0, 0,
    'INSERT INTO `'. $this->Db->db_data['LevelsRanks'][0]['Table'] . '` (
        `steam`, `name`, `value`, `rank`, `kills`, `deaths`, `shoots`, `hits`,
        `headshots`, `assists`, `round_win`, `round_lose`, `playtime`, `lastconnect`
    ) VALUES (:a, :b, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);', $qparams);

if($res) {
    $uid = $this->Db->lastInsertId(
        'LevelsRanks', 0, 0, $this->Db->db_data['LevelsRanks'][0]['Table']);
}
```

В случае если пользователь был добавлен, перед созданием записи нового счета, необходимо убедиться количество счетов для пользователя не превышает допустимого лимита. Если все в порядке, выполнить запись о новом счете в БД:

```
$res = $this->Db->insertion(
    'LevelsRanks', 0, 0, 'INSERT INTO `lr_web_shop` (
        `uid`, `prev`, `activate`, `days`
    ) VALUES (:a, :b, :c, :d);', $qparams);
```

при этом, данный *insert* запрос имеет один немаловажный момент: процесс запроса *insert* сопровождается триггером в базе данных, который при условии *after insert* производит вставку записи в таблицу *lr_web_shop_buylog*, в которую записываются данные *uuid* и время покупки привилегии.

После запроса *insert*, когда будет создана запись служебным триггером, будет запрошен генерированный *uuid* пользователя, который в свою очередь будет использован для формирования запроса оплаты на стороне внешней API системы оплаты платежей.

Каким образом обратимся к системе *qiwі*, уже было продемонстрировано на рисунке 7. Вся остальная логика действует примерно таким же образом, лишь с небольшими отличиями согласно своего алгоритма.

Вывод

Разработка или модернизация блоков ИС для достижения автоматизации определенных целей выводит продукт на новый уровень представления. Такие продукты становятся интерактивными, что повышает к ним интерес конечного пользователя, которая позволяет осуществить многообразие выбора. Для пользователя выбор - это маленький синоним или элемент свободы в его действиях. Пользователи очень любят этот феномен. К тому же для управления данными на основе их действий так же необходимо уделять внимание, что позволяет администраторам более точно и действительно оценивать текущую ситуацию, а так же очень важно - экономить драгоценное время.