**Тестовое задание**

Это тестовое задание не имеет никаких технических ограничений. Ты можешь использовать любые инструменты.

Задание №1

После успешной покупки билетов на событие, данные попадают в список заказов. Список заказов сохраняется в таблице MySql в виде:

| **id** | **event\_id** | **event\_date** | **ticket\_adult\_price** | **ticket\_adult\_quantity** | **ticket\_kid\_price** | **ticket\_kid\_quantity** | **barcode** | **user\_id** | **equal\_price** | **created** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 003 | 2021-08-21 13:00:00 | 700 | 1 | 450 | 0 | [11111111](https://bitbucket.org/nevatrip/test-php/commits/11111111) | 00451 | 700 | 2021-01-11 13:22:09 |
| 2 | 006 | 2021-07-29 18:00:00 | 1000 | 0 | 800 | 2 | [22222222](https://bitbucket.org/nevatrip/test-php/commits/22222222) | 00364 | 1600 | 2021-01-12 16:62:08 |
| 3 | 003 | 2021-08-15 17:00:00 | 700 | 4 | 450 | 3 | [33333333](https://bitbucket.org/nevatrip/test-php/commits/33333333) | 00015 | 4150 | 2021-01-13 10:08:45 |

Где:

* id - int(10) - инкрементальный порядковый номер заказа
* event\_id - int(11) - уникальный ид события. У каждого события есть свое название, описание, расписание, цены и свой уникальный event\_id соответственно
* event\_date - varchar(10) - дата и время на которое были куплены билеты
* ticket\_adult\_price - int(11) - цена взрослого билета на момент покупки
* ticket\_adult\_quantity - int(11) - количество купленных взрослых билетов в этом заказе
* ticket\_kid\_price - int(11) - цена детского билета на момент покупки
* ticket\_kid\_quantity - int(11) - количество купленных детских билетов в этом заказе
* barcode - varchar(120) - уникальный штрих код заказа
* equal\_price - int(11) - общая сумма заказа
* created - datetime - дата создания заказа

**Задача: написать функцию, которая будет добавлять заказы в эту таблицу.**

Аргументы которые функция получает на входе: event\_id, event\_date, ticket\_adult\_price, ticket\_adult\_quantity, ticket\_kid\_price, ticket\_kid\_quantity

Нужно сгенерировать barcode, который будет уникальным со случайным набором цифр, он не должен быть порядковым.

Так же, существует некая сторонняя api.site.com. API писать не нужно, возвращаемые данные можно замокать и возвращать в случайном порядке. в которой нужно сделать бронь заказа отправив ей (<https://api.site.com/book>) event\_id, event\_date, ticket\_adult\_price, ticket\_adult\_quantity, ticket\_kid\_price, ticket\_kid\_quantity, barcode. На что она может вернуть либо {message: 'order successfully booked'}, либо {error: 'barcode already exists'}. В случае если получаем ошибку, нужно сгенерировать новый barcode и повторить попытку. Важно учесть, если запрос будет происходить одновременно, не должно возникнуть такой ситуации, что двум разным заказам присвоился один номер.

После успешной брони, нужно отправить на стороннюю апи запрос с подтверждением (<https://api.site.com/approve>), который принимает только barcode. Ответов может быть 2 варианта - успешный: {message: 'order successfully aproved'} и различные варианты ошибок {error: 'event cancelled'}, {error: 'no tickets'}, {error: 'no seats'}, {error: 'fan removed'}. В случае успеха, сохраняем заказ в БД

Документация

К заданию 1

Для генерации баркода используется рекурсивный метод createBarcode:

function createBarcode($mysqli, $table){  
  
 $query = "SELECT barcode FROM $table";  
 $barcode = mt\_rand(10000000, 99999999);  
  
 $res = mysqli\_query($mysqli, $query);  
  
 $arr = mysqli\_fetch\_all($res, MYSQLI\_NUM);  
  
 for($i = 0; $i < count($arr); $i++){  
  
 if($barcode == $arr[$i][0]){  
  
 return createBarcode($mysqli, $table);  
  
 }  
  
 }  
  
 return $barcode;  
  
};

на вход принимает подключение к базе данных и название таблицы, для которой генерирует штрикод. Для генерации восьмизначного штрихкода используется функция mt\_rand.

С помощью функции mysqli\_fetch\_all получаем массив из количества строк и списка штрихкодов в нашей таблице. С помощью цикла for генерируем штрихкод для каждой строчки и с помощью if проверяем его на уникальность, если такой штрихкод совпал с ШК в таблице, запускается рекурсия и генерируется новый ШК, если ШК уникален – метод возвращает ШК.

Для добавления заказов в таблицу была создана функция createOrder(); на вход принимает ($event\_id, $event\_date, $ticket\_adult\_price, $ticket\_adult\_quantity, $ticket\_kid\_price, $ticket\_kid\_quantity, $mysqli).

Для генерации ШК используем ранее созданную функцию createBarcode.

Для подключения к API инициализируем новый сеанс cURL и отправляем на адрес <https://api.site.com/book> методом POST наши данные по заказам.

После получения от нее ответа, в случае true отправляем штрихкод заказа на стороннюю api <https://api.site.com/approve> который принимает только barcode. Получаем от нее ответ, в случае true, записываем данные заказа в базу данных. В случае false функция createOrder запускается рекурсивно, для создания нового ШК для заказа и отправки повторного запроса api.

function createOrder($event\_id, $event\_date, $ticket\_adult\_price, $ticket\_adult\_quantity, $ticket\_kid\_price, $ticket\_kid\_quantity, $mysqli){  
  
 $table = 'orders';  
  
 $barcode = createBarcode($mysqli, $table);  
  
 $curl = curl\_init();  
 curl\_setopt\_array($curl, array(  
 *CURLOPT\_URL* => 'https://api.site.com/book',  
 *CURLOPT\_POST* => true,  
 *CURLOPT\_POSTFIELDS* => http\_build\_query(array(  
 'event\_id' => $event\_id,  
 'event\_date' => $event\_date,  
 'ticket\_adult\_price' => $ticket\_adult\_price,  
 'ticket\_adult\_quantity' => $ticket\_adult\_quantity,  
 'ticket\_kid\_price' => $ticket\_kid\_price,  
 'ticket\_kid\_quantity' => $ticket\_kid\_quantity,  
 'barcode' => $barcode  
 ))  
 ));  
 $response = curl\_exec($curl);  
 curl\_close($curl);  
  
 if($response === true){  
  
 $curl = curl\_init();  
 curl\_setopt\_array($curl, array(  
 *CURLOPT\_URL* => 'https://api.site.com/approve',  
 *CURLOPT\_POST* => true,  
 *CURLOPT\_POSTFIELDS* => http\_build\_query($barcode)));  
  
 $response = curl\_exec($curl);  
 curl\_close($curl);  
  
 if($response === true){  
  
 $query = "INSERT INTO $table (  
 event\_id,  
 event\_date,  
 ticket\_adult\_price,  
 ticket\_adult\_quantity,  
 ticket\_kid\_price,ticket\_kid\_quantity,barcode   
 ) VALUES ($event\_id, $event\_date, $ticket\_adult\_price,  
 $ticket\_adult\_quantity, $ticket\_kid\_price, $ticket\_kid\_quantity, $barcode)";  
  
 mysqli\_query($mysqli, $query);  
  
 }  
  
 }else{  
 createOrder();  
 }  
  
  
}

Задание №2

1. Некоторые события нужно продавать с дополнительными типами билетов - льготный и групповой, у которых будут свои цены и название. Имеется информация, что вероятно, будут другие типы билетов, которые нужно будет добавить. Нужно уметь сохранять при заказе 2 дополнительных типа билета, льготный и групповой в бд. **Задача** - Нормализовать таблицу учитывая добавленные типы билетов, показать конечный вид таблицы. Объяснить свое решение.
2. Часто посетители из одного заказа приходят не одновременно на события. Возникает необходимость проверять их билеты по отдельности. Для этого у каждого билета должен быть свой баркод. Если в одном заказе было куплено несколько билетов, 2 взрослых, 3 детских, 4 льготных - то должно быть 9 баркодов для каждого билета соответственно. **Задача** - Нормализовать таблицу, учитывая что у каждого билета свой баркод, показать конечный вид таблицы. Объяснить свое решение.

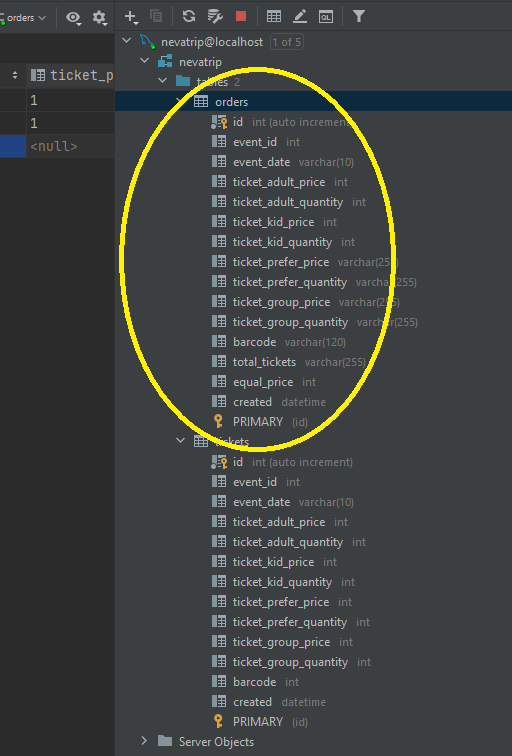
Документация к заданию 2:

1. Для групповых и льготных билет была создана функция insertColumns() добавления необходимых столбцов в нашу таблицу Orders. Для различных типов билетов создан архив $ticket\_types.

Данная функция добавляет столбцы с ценой и количеством групповых и льготных билетов, путем запросов mysqli к базе данных.

$ticket\_types = ['adult', 'kid', 'prefer', 'group'];  
  
$ticket\_type = $ticket\_types[2];  
  
function insertColumns($ticket\_type, $mysqli, $table){  
  
  
 $table = 'orders';  
   
 if($ticket\_type == 'group'){  
  
 $query = "ALTER TABLE $table ADD COLUMN ticket\_group\_price VARCHAR(255) AFTER ticket\_kid\_quantity";  
  
 mysqli\_query($mysqli, $query);  
  
 $query = "ALTER TABLE $table ADD COLUMN ticket\_group\_quantity VARCHAR(255) AFTER ticket\_group\_price";  
  
 mysqli\_query($mysqli, $query);  
  
 echo('Добавление столбцов в базу данных успешно');  
  
  
  
 };  
  
 if($ticket\_type == 'prefer'){  
  
 $query = "ALTER TABLE $table ADD COLUMN ticket\_prefer\_price VARCHAR(255) AFTER ticket\_kid\_quantity";  
  
 mysqli\_query($mysqli, $query);  
  
 $query = "ALTER TABLE $table ADD COLUMN ticket\_prefer\_quantity VARCHAR(255) AFTER ticket\_prefer\_price";  
  
 mysqli\_query($mysqli, $query);  
  
 echo('Добавление столбцов в базу данных успешно');  
  
 };  
  
}

Итоговая таблица выглядит вот так (добавилось четыре столбца):



1. Часто посетители из одного заказа приходят не одновременно на события. Возникает необходимость проверять их билеты по отдельности. Для этого у каждого билета должен быть свой баркод. Если в одном заказе было куплено несколько билетов, 2 взрослых, 3 детских, 4 льготных - то должно быть 9 баркодов для каждого билета соответственно. **Задача** - Нормализовать таблицу, учитывая что у каждого билета свой баркод, показать конечный вид таблицы. Объяснить свое решение.

Для данной задачи, будет создана новая таблица tickets, в которой будет содержаться информация по каждому билету каждого события с уникальным баркодом для каждого билета.

Также создано пять новых функций:

createAdultTickets(){};

createKidTickets(){};

createPreferTickets(){};

createGroupTickets(){};

createTicketsList(){};

Первые четыре функции выполняют одинаковые задачи (много дублирования кода, но иначе я реализацию пока не придумал). Принцип работы функций следующий:

- создаем запрос к базе данных и выдергиваем из таблицы Orders номер события(event\_id) и количество билетов по этому событию (ticket\_adult\_quantity), где количество не равно нулю. создаем из этих данных архив $arr.

- проходясь по архиву двойным циклом for, функция копирует данные каждого события в новую таблицу tickets, но для каждого билета создается отдельная строчка.

function createAdultTickets($mysqli){  
  
 $query = "SELECT event\_id, ticket\_adult\_quantity FROM orders WHERE ticket\_adult\_quantity != 0";  
 $res = mysqli\_query($mysqli, $query);  
  
  
 $arr = mysqli\_fetch\_all($res, MYSQLI\_NUM);  
  
  
 for($i = 0; $i < count($arr); $i++){  
  
 for($k = 0; $k < $arr[$i][1]; $k++){  
  
  
 $event\_id = $arr[$i][0];  
 $query = "INSERT INTO tickets (  
 event\_id,  
 event\_date,  
 ticket\_adult\_price,  
 ticket\_adult\_quantity,  
 created)  
 SELECT event\_id,  
 event\_date,  
 ticket\_adult\_price,  
 ticket\_adult\_quantity/ticket\_adult\_quantity,  
 created   
 FROM orders   
 WHERE event\_id = $event\_id";  
  
  
 mysqli\_query($mysqli, $query);  
  
 }  
  
 }  
  
  
};

function createKidTickets($mysqli){  
  
 $query = "SELECT event\_id, ticket\_kid\_quantity FROM orders WHERE ticket\_kid\_quantity != 0";  
 $res = mysqli\_query($mysqli, $query);  
  
 $arr = mysqli\_fetch\_all($res, MYSQLI\_NUM);  
  
  
 for($i = 0; $i < count($arr); $i++){  
  
 for($k = 0; $k < $arr[$i][1]; $k++){  
  
  
 $event\_id = $arr[$i][0];  
 $query = "INSERT INTO tickets (  
 event\_id,  
 event\_date,  
 ticket\_kid\_price,  
 ticket\_kid\_quantity,  
 created)  
 SELECT event\_id,  
 event\_date,  
 ticket\_kid\_price,  
 ticket\_kid\_quantity/ticket\_kid\_quantity,  
 created   
 FROM orders   
 WHERE event\_id = $event\_id";  
  
  
 mysqli\_query($mysqli, $query);  
  
 }  
  
 }  
};  
  
  
function createPreferTickets($mysqli){  
  
 $query = "SELECT event\_id, ticket\_prefer\_quantity FROM orders WHERE ticket\_prefer\_quantity != 0";  
 $res = mysqli\_query($mysqli, $query);  
  
 $arr = mysqli\_fetch\_all($res, MYSQLI\_NUM);  
  
  
 for($i = 0; $i < count($arr); $i++){  
  
 for($k = 0; $k < $arr[$i][1]; $k++){  
  
  
 $event\_id = $arr[$i][0];  
 $query = "INSERT INTO tickets (  
 event\_id,  
 event\_date,  
 ticket\_prefer\_price,  
 ticket\_prefer\_quantity,  
 created)  
 SELECT event\_id,  
 event\_date,  
 ticket\_prefer\_price,  
 ticket\_prefer\_quantity/ticket\_prefer\_quantity,  
 created   
 FROM orders   
 WHERE event\_id = $event\_id";  
  
  
 mysqli\_query($mysqli, $query);  
  
 }  
  
 }  
  
}  
  
  
function createGroupTickets($mysqli){  
  
 $query = "SELECT event\_id, ticket\_group\_quantity FROM orders WHERE ticket\_group\_quantity != 0";  
 $res = mysqli\_query($mysqli, $query);  
  
 $arr = mysqli\_fetch\_all($res, MYSQLI\_NUM);  
  
  
 for($i = 0; $i < count($arr); $i++){  
  
 for($k = 0; $k < $arr[$i][1]; $k++){  
  
  
 $event\_id = $arr[$i][0];  
 $query = "INSERT INTO tickets (  
 event\_id,  
 event\_date,  
 ticket\_group\_price,  
 ticket\_group\_quantity,  
 created)  
 SELECT event\_id,  
 event\_date,  
 ticket\_group\_price,  
 ticket\_group\_quantity/ticket\_group\_quantity,  
 created   
 FROM orders   
 WHERE event\_id = $event\_id";  
  
  
 mysqli\_query($mysqli, $query);  
  
 }  
  
 }  
  
}

Функция createTicketsList(){} генерирует финальную таблицу билетов, запустив ранее созданные методы:

createAdultTickets(){};

createKidTickets(){};

createPreferTickets(){};

createGroupTickets(){};

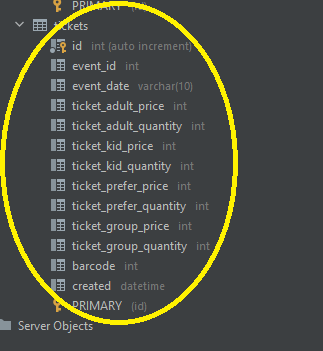
выгружаются все билеты в нашей новой таблице tickets. Далее путем запроса "SELECT id FROM $table" и функции mysqli\_fetch\_all($res, MYSQLI\_NUM);

создаем архив $arr, где хранится список id каждого билета. Затем циклом for перебираем архив и для каждого id создаем barcode, ранее созданной функцией createBarcode и записываем в таблицу.

function createTicketsList($mysqli){  
  
 $table = 'tickets';  
  
 createAdultTickets($mysqli);  
 createKidTickets($mysqli);  
 createPreferTickets($mysqli);  
 createGroupTickets($mysqli);  
  
 $query = "SELECT id FROM $table";  
 $res = mysqli\_query($mysqli, $query);  
  
 $arr = mysqli\_fetch\_all($res, MYSQLI\_NUM);  
  
  
  
 for($i = 0; $i < count($arr); $i++){  
  
 $barcode = createBarcode($mysqli, $table);  
 $id = $arr[$i][0];  
 $query = "UPDATE $table SET barcode=$barcode WHERE id=$id";  
 mysqli\_query($mysqli, $query);  
 echo ('<br>' . $barcode);  
  
 };  
  
}  
createTicketsList($mysqli);

Финальная таблица tickets выглядит так, в ней содержаться все билеты по типам, привязанные к эвентам и с уникальным штрихкодом.:





Задание №3

Сопроводить документацией своё решение.