Membuat response formatter untuk API

helper response formatter untuk API ini berguna untuk mempermudah kita untuk mengetahui apakah api yang dikeluarkan sukses apa error :

https://github.com/belajarkoding/laravel-response-formatter

cara menggunakan :

1. clone or copy dan paste di app\Helpers\ResponseFormatter.php

---

Membuat API Login

cara membuat :

1. $php artisan make:controller API\UserController
2. pada file UserController.php buat function bernama login

```
public function login(Request $request)
{

}
```

3. buat validasi apakah loginnya berhasil atau tidak , maka buat code trycatch.
4.
```
public function login(Request $request)
{

}
```

5. buat validasi input pada login di dalam block try

```
$request->validate([
    'email'=>'email|required',
    'password'=>'required'
]);
```

6. cek credentials login

```
$credentials = $request(['email', 'password']);
if(!Auth::attempt([$credentials]))
{
    return ResponseFormatter::error([
        'message' => 'Unauthorized'
    ], 'Authentication Faileds', 500);
}
```

7. jika hash tidak sesuai maka loginkan

```
//Jika hash/akun tidak sesuai maka beri error
$user = User::where('email', $request->email)->first();
if(!Hash::check($request->password, $user->password, []))
{
    throw new \Exception('Invalid Credentials');
}
```

8. jika berhasil maka loginkan

```
//jika berhasil maka loginkan
$tokenResult = $user->createToken('authToken')->plainTextToken;
return ResponseFormatter::success([
    'access_token' => $tokenResult,
    'token_type' => 'Bearer',
    'user'=> $user
],'Authenticated');
```

9. buat code dalam block catch (jika auth nya gagal)

```
catch(Exception $error){
return ResponseFormatter::error([
        'message' => 'Something went wrong',
        'error' => $error
    ], 'Authentication Failed', 500);
}
```

10.
```
public function login(Request $request)
{
    try {
        //validasi Input
        $request->validate([
            'email'=>'email|required',
            'password'=>'required'
        ]);

        //cek credentials
        $credentials = $request(['email', 'password']);
        if(!Auth::attempt([$credentials]))
        {
            return ResponseFormatter::error([
                'message' => 'Unauthorized'
            ], 'Authentication Faileds', 500);
        }

        //Jika hash tidak sesuai maka beri error
        $user = User::where('email', $request->email)->first();
        if(!Hash::check($request->password, $user->password, []))
        {
            throw new \Exception('Invalid Credentials');
        }

        //jika berhasil maka loginkan
```

```
        $tokenResult = $user->createToken('authToken')->plainTextToken;
        return ResponseFormatter::success([
            'access_token' => $tokenResult,
            'token_type' => 'Bearer',
            'user'=> $user
        ],'Authenticated');
    }
    catch(Exception $error){
        return ResponseFormatter::error([
            'message' => 'Something went wrong',
            'error' => $error
        ], 'Authentication Failed', 500);
    }
}
```

Membuat API Register

1. Buat function pada file UserController.php dengan nama function register dan code block trycatch

```
public function register(Request $request)
{
    try{

    }catch(Exception $error){

    }
}
```

2. pada code block try, buat validasi request:

```
$request->validate([
    'name' => ['required', 'string', 'max:255'],
    'email' => ['required', 'string', 'email', 'max:255'],
    'password' =>  $this -> passwordRules()
]);
```

3. Pada Field Password , import libarry untuk validasinya, sehingga kita hanya pakai saja.

```
//gunakan paling atas
use App\Actions\Fortify\PasswordValidationRules;

//gunakan di dalam class
class UserController extends Controller
{
    use PasswordValidationRules;
}
```

4. pada codeblock try, buatkan User itu sendri dengan User::create

```
User::create([
    'name'=> $request-> name ,
    'email'=> $request-> email ,
    'address'=> $request->address  ,
    'houseNumber'=> $request->houseNumber  ,
    'phoneNumber'=> $request-> phoneNumber ,
    'city'=> $request-> city ,
    'password'=> Hash::make($request->password)
]);
```

5. lalu ambil data user yang sudah dibuat , dan buatkan tokenresultnya

```
$user = User::where('email', $request->email)-> first();
$tokenResult = $user -> createToken('authToken')->plainTextToken;
```

6. return hasilnya, yaitu tokenresultnya dan user tersebut,

```
return ResponseFormatter::success([
    'access_token' => $tokenResult,
    'token_type' => 'Bearer',
    'user'=> $user
]);
```

7. jika sudah semua, masukkan catch nya, seperti pada catch dalam membuat api .

Membuat API Logout

Cara membuat API Logout :

1. buat variable RevokedToken dari $request user yang telah login menggunakan currentaccesstoken yang telah diberikan dan delete
2. return response success dengan parameter dari variable diatas dan string ('revoked token')

```
public function logout(Request $request)
{
    $tokenRevoked = $request->user()->currentAccessToken()->delete();
    return ResponseFormatter::success($tokenRevoked, 'Token has been revoked');
}
```

Membuat API UpdateProfile

Update profile itu , si user yang telah login pada aplikasi react native , memberikan data yang akan dikirim ke server laravel untuk diubah.

caranya :

1. Tentuin data nya apa aja yang mau diubah
2. Cari , data tersebut milik akun siapa
3. Update data pada akun tersebut
4. berikan responseformatter success dengan parameter user dan pesannya apa.

```
public function updateProfile(Request $request)
{
    $data = $request->all();
    $user = Auth::user();
    $user->update($data);

    return ResponseFormatter::success($user, 'Profile has been updated');
}
```

Membuat API User

Bagaimana cara mengambil data user dari database laravel menggunakan API ?

1. buat function pada class UserController dengan nama **fetch()** dan return requestnya .

```
public function fetch(Request $request)
{
    return ResponseFormatter::success($request->user(), 'Data profile
berhasil          diambil');
}
```

Membuat API Upload Photo

Bagaimana cara update photo profile user ?

1. jalankan >$ php artisan storage:link
2. Buat function updatephoto dengan parameter Request pada class UserController.
3. Buat variable validator untuk requirements ketentuan photonya , contoh mau maks brp Mb, file typenya apa, mandatory apa ngga, menggunakan library library validator

```
use Illuminate\Support\Facades\Validator;
```

4. pada library tersebut , gunakan function make() dgn parameter , isi semua requestnya dan requirementsnya.

```
$validator = Validator::make($request->all(), [
    'file' => 'required|image|max:2048'
]);
```

5. Jika validatornya gagal cek menggunakan variable validators dan function fail bawaan validator library, maka return responseformatter error
6. jika berhasil, maka uploadkan photonya ke dalam folder dan , simpan photonya ke dalam database.
7. return responseformatter sukses.

```
//2.
public function updatePhoto(Request $request)
{
    $validator = Validator::make($request->all(), [
    'file' => 'required|image|max:2048'
    ]);

    if($validator->fails())
    {
        return ResponseFormatter::error(
            ['error'=>$validator->errors()],
            'update photo fails',
            401
        );
    }
    if($request->file('file'))
    {
        $file = $request->file;
        $file->store('assets/user', 'public');

        $user = Auth::user();
        $user->profile_photo_path = $file;
        $user->update();

        return ResponseFormatter::success([$file], 'file            successfully
updated')
    }

}
```

Pembuatan Route oleh UserController

Route yang dapat diakses secara public :

1. **Login**
2. **Register**

Route yang dapat diakses hanya ketika sudah login, harus dimasukkan kedalam rotue group Middleware

1. **fetch()**
2. **updateProfile()**
3. **updatePhoto()**
4. **logout()**

```
Route::middleware('auth:sanctum')->group(function(){
    Route::get('user', [UserController::class, 'fetch']);
    Route::post('user', [UserController::class,
'updateProfile']);
    Route::post('user/photo', [UserController::class,
'updatePhoto']);
    Route::post('logout', [UserController::class, 'logout']);
});

//route yang diakses secara public
Route::post('login', [UserController::class,'login']);
Route::post('register', [UserController::class,'register']);
```

Membuat API Food

cara membuat API untuk data data food, kita ingin mengambil data berdasarkan :

1. ID
2. Name
3. price from
4. price to
5. rate from
6. rate to
7. types

cara nya yaitu ,

1. Buat controller api nya dengan nama FoodController di dalam folder API.
2. Buat function bernama all dengan parameter Request
3. buat variable dengan nama nama berdasarkan variable di atas yang berisikan sesuai dengan request oleh request client.
4. Buat conditional searching berdasarkan variable yang telah di buat di nomor 3.
    - jika request inputnya id , maka food::find($id), jika food maka return respons food, else , return error

```
if($id)
{
    $food = Food::find($id);
    if($food)
    {
        return ResponseFormatter::success($food, 'Data produk berhasil
diambil');
    }
    else
    {
        return ResponseFormatter::error(null, 'Data Produk Tidak Ada',
404);
    }
}
```

notes :

Food::find($id) -> hai Model food, tolong carikan data berdasarkan ID

Food::query(); -> hai model food, tolong carikan data berdasarkan query

```
Food::query()->where('name', 'like' , '%' . 'name' . '%');
```

contoh all code :

```
public funtion all(Request $request)
{
    $id = $request->input('id');
    $limit = $request->input('limit');
    $name = $request->input('name');
    $types = $request->input('types');
    $price_from = $request->input('price_from');
    $price_to = $request->input('price_to');
    $rate_from = $request->input('rate_from');
    $rate_to = $request->input('rate_to');

    if($id)
    {
        $food = Food::find($id);

        if($food)
        {
            return ResponseFormatter::success($food, 'Data produk berhasil
diambil');
        }
        else
        {
            return ResponseFormatter::error(null, 'Data Produk Tidak Ada',
404);
        }
    }

    $food = Food::query();
```

```
    if($name)
    {
        Food::query()->where('name', 'like' , '%' . 'name' . '%');
    }
    if($types)
    {
        Food::query()->where('name', 'like' , '%' . 'name' . '%');
    }
    if($price_from)
    {
        Food::query()->where('price', '>=' , $price_from);
    }
    if($price_to)
    {
        Food::query()->where('price', '<=' , $price_to);
    }
    if($rate_from)
    {
        Food::query()->where('rate', '>=' , $rate_from);
    }
    if($rate_to)
    {
        Food::query()->where('rate', '>=' , $rate_to);
    }

    return ResponseFormatter::success(
        $food->paginate($limit),
        'Data berhasil diambil'
    )
}
```

Membuat API Transaction

API transactions itu, kita ingin mengambil transaksi , statusnya apa, foodnya apa, terus usernya juga apa .

jadi contoh . "Hai transaction model, tolong dong cariin transaksi oleh user yang udah login".

intinya sama seperti di atas, hanya variable nya aja yang dibedain.

Cara membuatnya :

1. Buat Controller dengan nama TransactionController
2. Buat function all() dengan parameter Request
3. buat variable dari request input yang diminta oleh client.
4. Jika inputnya by id tentukan transaksinya , berdasarkan relasi dari user dan food.

```
$id = $request -> input('id');
if($id)
{
    $transaction = Transaction::with(['food', 'user'])->find($id);
```

```
    if($transaction)
    {
        return ResponseFormatter::success($transaction,
        'Data Transaksi berhasil diambil');
    }
    else
    {
        Return ResponseError::error(null, 'Data transaksi tidak ada', 404)
    }
}
```

5. return transaksinya jika yang dicari transaksi ID
6. Buat variable transaction untuk mencari Transaction berdasarkan userid yang sedang login

```
$transaction = Transaction::with(['food', 'user'])
                ->where('user_id', Auth::user()->id);
```

7. jika requestnya yang diminta itu foodid , maka yang dicari adalah transaction berdasarkan userid yang sedang login dan food_id dari transakti yang ingin dicari berdasarkan request input dari client.

```
if($food_id)
{
    $transaction -> where('food_id', $food_id)
}

//notes : code di atas itu sebenernya kaya gini, kalau mau simple or
dimengerti

if($request->input('food_id'))
{
    Transaction::with(['food', 'user'])->where('user_id', Auth::user()->id)
                                       ->where('food_id', $food_id);
}

psudocode :
jika request input yang diminta user itu food_id
    Transaksi , tolong carikan data table kamu dengan kondisi
    kolom user_id nya user yang sedang login dan
    kolom food_id nya itu sesuai requestan input food_id
```

8. Jika request input dari client itu status , maka carikan data transaksi dengan kolom status berdasarkan request input status.

```
if($status)
{
    $transaction -> where('status', $status);
}

//notes : code di atas itu sebenernya kaya gini, kalau mau simple or
dimengerti
if($request->input('stauts'))
```

```
{
    Transaction::with(['food', 'user'])->where('user_id', Auth::user()->id)
                                       ->where('status', $status);
}

psudocode :
jika request input yang diminta user itu berdasarkan status
    Table Transaksi , tolong carikan data table kamu dengan kondisi
    kolom user_id nya user yang sedang login dan
    kolom status nya itu sesuai requestan input 'status'
```

Membuat API Transaction : POST (UPDATE)

pada controller transaction, kita ingin mencari transactionnya berdasardkan ID nya,

lalu pada transaction dengan ID tersebut , kita update dengan update an request->all() dari si client.

cara membuatnya :

1. Buat function bernama update dengan parameter Request dan id.
2. di dalam function, cari transaction berdasarkan id .
3. lalu update transaksi tersebut sesuai dengan request yang diminta oleh si client.
4. return ResponsFormatternya
5. buat routenya dengan link 'transaction/{id}' di dalam route group middleware

contoh :

1. ```
   public function update(Reqeust $request, $id)
   {
       $transaction = Transaction::findOrFail($id);
       $transaction -> update($request->all());
       return ResponseFormatter::success($transaction, 'Transaksi berhasil
   diupdate')
   }
   ```

2. ```
   Route::post('transaction/{id}', [TransactionController::class, 'update']);
   ```

Membuat API Checkout (Install dan Daftar Midtrans)

https://github.com/Midtrans/midtrans-php

cara install midtrans :

1. composer require midtrans/midtrans-php
2. Semua API Key including server key ada di bagian settings

3. Setting Redirections Settings
    - Finish Url , Unfinish URL, dan Error payment URL. (later)
    - Payment Notification URL (later)
4. Setelah daftar midtrans dan install midtrans ,
5. Edit file **.env** dan **config\services.php**

```
//services.php
'midtrans' => [
    'serverKey' => env('MIDTRANS_SERVER_KEY'),
    'clientKey' => env('MIDTRANS_CLIENT_KEY'),
    'isProduction' => env('MIDTRANS_IS_PRODUCTION', false),
    'isSanitized' => env('MIDTRANS_IS_SANITIZED', true),
    'is3ds' =>  env('MIDTRANS_IS_3DS', true)
]
```

```
//.env
MIDTRANS_SERVER_KEY = ""
MIDTRANS_CLIENT_KEY = ""
MIDTRANS_IS_PRODUCTION = false
MIDTRANS_IS_SANITIZED true
MIDTRANS_IS_3DS = true
```

Membuat API (Checkout Controller) Post dengan midtrans.

Di sini itu maksudnya, gimana caranya , data data transaksi yang ada di database kita, bisa ke isi di database midtrans juga .

jadi , kita mengirimkan data transaksi ke database midtrans.

Alur Flow :

1. Buat function checkout dengan parameter request
2. Validasi request yang diisi oleh user client
3. buat data di Table transaction dengan Transaction::create dengan filed yang sudah diisi oleh client
4. Konfig midtransnya , (serverKey, isProduction, isSanitized, is3ds)
5. Panggil transaksi yang udah dibuat di nomor 3 tadi, (cari id nya sehingga dapat datanya)
6. Buat Array Transaksi untuk kita taro di midtrans

   (https://snap-docs.midtrans.com/#request-body-json-parameter)
7. (try) buat payment url nya dengan cara : di midtrans dengan function Snap::createTransaction($arrayNomor5)-> redirectUrl,
8. (try) masukkan payment urlnya ke dalam database, lalu save
9. (try) return ResponseFormatter jika berhasil

10. (catch) return ResponseFormater::error jika gagal
11. masukkan route nya , ke dalam middleware group dgn method post.

1.
```
public function checkout(Request $request)
{

}
```

2.
```
//2.
$request->validate([
    'food_id' => 'required|exists:food,id'
    'user_id' => 'required|exists:user, id',
    'quantity' => 'requiered',
    'total' => 'required',
    'status' => 'required'
]);
```

3.
```
$transaction = Transaction::create([
    'food_id' => $request->food_id,
    'user_id' => $request->user_id,
    'quantity' => $request->quantity,
    'total' => $request->total,
    'status' => $request->status,
]);
```

4.
```
Config::$serverKey = config('services.midtrans.serverKey');
Config::$isProduction = config('services.midtrans.isProduction');
Config::$isSanitized = config('services.midtrans.isSanitized');
Config::$is3ds = config('services.midtrans.is3ds');
```

5.
```
$transaction = Transaction::with(['user','food'])->find($transaction->id);
```

6.
```
$transactioninMidtrans = [
    "transaction_details"=> [
        "order_id"=> $transaction->id,
        "gross_amount"=> (int)$transaction->total
    ],
    customer_details" => [
        "first_name"=> $transaction->user->name,
        "email"=> $transaction->user->email
    ],
    'enabled_payments'=> ["gopay", "bca_va"],
    'vtweb'=> []
]
```

7.
```
try{
    $paymentUrl = Snap::createTransaction($transactioninMidtrans)-
>redirect_url;
}
```

8.
```
try{
    //code nomor 7
    $transaction->payment_url = $payment_url
    $transaction->save();
}
```

9.
```
try{
    return ResponseFormatter::success($transaction, 'Transaksi Berhasil');
}
```

10.
```
catch(Exception $e){
    return ResponseFormatter::error($e->getMessage(), 'Transaksi Gagal');
}
```

11.
```
Route::post('checkout', [TransactionController::class, 'checkout']);
```

Kesimpulan :

jadi ini itu intinya, ngebuat transaksi di database kita, tapi , di mitrans juga terisi.

Callback Notification API from Midtrans

Di bab ini maksudnya kita ingin menyimpan status pembayaran (di table Transaction database kita) dari midtrans ke dalam database kita . kita butuh Notification instance.

1. Import library Notification dari Midtrans
2. Buat Controller dengan nama MidtransController dan buat function named callback
3. Set Konfigrusi Midtrans
4. buat variable $transaction (instans new) Notifications
5. assign ke variable , (status, type, fraud, order_id)
6. cari transaksi berdasarkan id yang mana , id yang di cari itu berdasarkan variable order_id di nomor 5
7. Handle transaksi status midtrans , contoh

```
if($notification->status == 'pending'){
    $transaction->status = 'PENDING'
}
```

8. Simpan transaction status

1.
```
use Midtrans\Notification;
```

2.
```
$> php artisan make:controller MidtransController
```

```
public function callback(Request $request)
{
}
```

3. ```
   Config::$serverKey = config('services.midtrans.serverKey');
   Config::$isProduction = config('services.midtrans.isProduction');
   Config::$isSanitized = config('services.midtrans.isSanitized');
   Config::$is3ds = config('services.midtrans.is3ds');
   ```

4. ```
   $notification = new Notification();
   ```

5. ```
   $status = $notification -> transaction_status;
   $type = $notification -> payment_type;
   $fraud = $notification -> fraud_status;
   $order_id = $notification -> order_id;
   ```

6. ```
   $transaction = Transaction::findOrFail($order_id);
   ```

7. ```
   if($status == 'capture'){
       if($type == 'credit_card'){
           if($fraud == 'challange')
           {
               $transaction->status = 'PENDING'
           }
           else
           {
               $transaction->status = 'SUCCESS'
           }
       }
   }
   else if($status == 'settlement')
   {
       $transaction->status = 'SUCCESS';
   }
   else if($status == 'pending')
   {
       $transaction->status = 'PENDING';
   }
   else if($status == 'deny')
   {
       $transaction->status = 'CANCELLED';
   }
   else if($status == 'expire')
   {
       $transaction->status = 'CANCELLED';
   }
   else if($status == 'cancel')
   {
       $transaction->status = 'CANCELLED';
   }
   ```
   ```

8.
```
$transaction->save();
```

Membuat view feedback transaksi (unfinish/sukses/error)

jadi , kita buat halaman untuk memberi tahu user , bahwa transaksi nya apa.

cara nya :

1. Buat function named error, success, dan unfinish.
2. return view dalam function tersebut
3. buat routenya di web.php dengan method get.
4. buat file di dalam folder resource\view\midtrans\unfinish.blade.php , dst
5. percantik file blade.php nya (simple aja)

1.
```
public function unfinish()
{
    return view('midtrans.unfisih')
}
public function success()
{
    return view('midtrans.success')
}
public function error()
{
    return view('midtrans.error')
}
```
