Definir relações que vamos serializar

Api ClientCheckoutController

*//definir as relações que precisamos serializar***private $with** = [**'items'**, **'client'**, **'cupom'**];

Chamando na função de listagem

**public function** index()  
{  
 $id = Authorizer::*getResourceOwnerId*();  
 $clientId = $this->**userRepository**->find($id)->**client**->**id**;  
 $orders = $this->**orderRepository** ->skipPresenter(**false**)  
 ->with($this->**with**)->scopeQuery(**function**($query) **use** ($clientId) {  
 **return** $query->where(**'client\_id'**, **'='**, $clientId);  
 })->paginate();  
  
 **return** $orders;  
}

Chamando no método Store

**public function** store(CheckoutRequest $request)  
{  
 $data = $request->all();  
 $id = Authorizer::*getResourceOwnerId*();  
 $clientId = $this->**userRepository**->find($id)->**client**->**id**;  
 $data[**'client\_id'**] = $clientId;  
 $order = $this->**orderService**->create($data);  
 *//$order = $this->orderRepository->with(['items'])->find($order->id);* **return** $this->**orderRepository**->with($this->**with**)->find($order->**id**);  
}

Chamando no método Show

**public function** show($id)  
 {  
 **return** $order = $this->**orderRepository** ->skipPresenter(false)  
 ->with($this->**with**)->find($id);  
 *//COLLECTION DO LARAVEL - EACH PODEMOS APLICAR COISAS PARA CADA REGISTRO  
// $order->items->each(function($item){  
// $item->product;  
// });  
 //return $order;* }

Criar Transformer e Presenter de CLient

rafael.franco@NOTWEB-O1 MINGW64 /e/PROGRAMAS/www/code-delivery-laravel-ionic (master)

$ php artisan make:transform Client

Transformer created successfully.

rafael.franco@NOTWEB-O1 MINGW64 /e/PROGRAMAS/www/code-delivery-laravel-ionic (master)

$ php artisan make:presenter Client

CodeDelivery\Transformers\ClientTransformerPresenter created successfully.

Atribuir o relacionamento em OrderTransform

**protected $availableIncludes** = [**'cupom'**, **'items'**, **'client'**];

Métodos de Inclusão

**public function** includeClient(Order $model)  
{  
 **return** $this->item($model->**client**, **new** ClientTransformer());  
}  
  
*//Many to One - CUmpo  
/\*****TODO MEDOTO PARA SERIALZAR O RELACIONAMENTO****\*/***public function** includeCupom(Order $model)  
{  
 *//VALIDANDO O CUPOM - PQ O CUPOM PODE OU NAO PODE EXISTIR* **if**(!$model->**cupom**)  
 **return null**;  
  
 **return** $this->item($model->**cupom**, **new** CupomTransformer());  
}  
  
*//One To Many - Items***public function** includeItems(Order $model)  
{  
 **return** $this->collection($model->**items**, **new** OrderItemTransformer());  
}

Moder Order atribuir o filable de cupom\_id

**protected $fillable** = [  
 **'client\_id'**,  
 **'user\_deliveryman\_id'**,  
 **'total'**,  
 **'status'**,  
 **'cupom\_id'**];

Validar no Método Create do OrderService

**public function** create(**array** $data)  
{  
 *//TRANSACAO* \DB::*beginTransaction*();  
 **try** {  
 $data[**'status'**] = 0;  
  
 **if**(**isset**($data[**'cupom\_id'**]))  
 {  
 **unset**($data[**'cupom\_id '**]);  
 }  
 **if**(**isset**($data[**'cupom\_code'**]))  
 {  
 $cupom = $this->**cupomRepository**->findByField(**'code'**, $data[**'cupom\_code'**])->first();  
 $data[**'cupom\_id'**] = $cupom->**id**;  
 $cupom->**used** = 1;  
 $cupom->save();  
 **unset**($data[**'cupom\_code'**]);  
 }  
  
 $items = $data[**'items'**];  
 **unset**($data[**'items'**]);  
  
 $order = $this->**orderRepository**->create($data);  
 $total = 0;  
  
 **foreach**($items **as** $item)  
 {  
 $item[**'price'**] = $this->**productRepository**->find($item[**'product\_id'**])->**price**;  
 $order->items()->create($item);  
 $total += $item[**'price'**] \* $item[**'qtd'**];  
 }  
  
 $order->**total** = $total;  
  
 *//SE TIVER UTILIZANDO CUPOM* **if**(**isset**($cupom))  
 {  
 $order->**total** = $total - $cupom->**value**;  
 }  
  
 $order->save();  
 \DB::*commit*();  
 **return** $order;  
 } **catch** (\Exception $e) {  
 \DB::*rollback*();  
 **throw** $e;  
 }  
}