Definir os Relacionamentos de Serialização

DeliverymanCheckoutController

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

Chamar no Método Index

**public function** index()  
{  
 $id = Authorizer::*getResourceOwnerId*();  
 $orders = $this->**orderRepository** ->skipPresenter(**false**)  
 ->with([**'items'**])->scopeQuery(**function**($query) **use** ($id) {  
 **return** $query->where(**'user\_deliveryman\_id'**, **'='**, $id);  
 })->paginate();  
  
 **return** $orders;  
}

Chamar no Método Show

**public function** show($id)  
{  
 $idDeliveryman = Authorizer::*getResourceOwnerId*();  
 **return** $this->**orderRepository** ->skipPresenter(**false**)  
 ->getByIdAndDeliveryman($id, $idDeliveryman);  
 **return** $order;  
}

Chamar no Método UpdateStatus

Corrigir método em OrderRepositoryEloquent

**public function** getByIdAndDeliveryman($id, $idDeliveryman)  
 {  
 $result = $this->with([**'client'**, **'items'**, **'deliveryman'**])->findWhere([**'id'** => $id, **'user\_deliveryman\_id'** => $idDeliveryman]);  
  
 **if**($result **instanceof** Collection)  
 {  
 $result = $result->first();  
 } **else** {  
 **if**(**isset**($result[**'data'**]) && *count*($result[**'data'**]) == 1)  
 {  
 $result = [  
 **'data'** => $result[**'data'**][0]  
 ];  
 } **else** {  
 **throw new** ModelNotFoundException(**"Order não existe"**);  
 }  
 }  
  
*// if($result){  
// $result->items->each(function($item){  
// $item->product;  
// });  
// }* **return** $result;  
 }