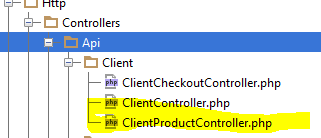
Criar ClientProductController

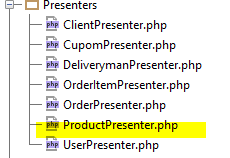


**<?php  
  
namespace** CodeDelivery\Http\Controllers\Api\Client;  
  
**use** CodeDelivery\Http\Controllers\Controller;  
**use** CodeDelivery\Repositories\ProductRepository;  
  
  
**class** ClientProductController **extends** Controller  
{  
  
 **private $productRepository**;  
  
 **public function** \_\_construct(ProductRepository $productRepository)  
 {  
 $this->**productRepository** = $productRepository;  
 }  
  
 **public function** index()  
 {  
 $products = $this->**productRepository**->skipPresenter(**false**)->all();  
 **return** $products;  
 }  
}

Chamar a Rota dentro do Middlware de cliente

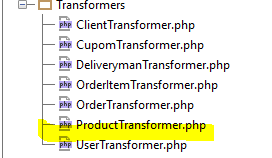
Route::*get*(**'products'**, **'Api\Client\ClientProductController@index'**);

Criar ProductPresenter



**<?php  
  
namespace** CodeDelivery\Presenters;  
  
**use** CodeDelivery\Transformers\ProductTransformer;  
**use** Prettus\Repository\Presenter\FractalPresenter;  
  
*/\*\*  
 \* Class ProductPresenter  
 \*  
 \** ***@package*** *namespace CodeDelivery\Presenters;  
 \*/***class** ProductPresenter **extends** FractalPresenter  
{  
 */\*\*  
 \* Transformer  
 \*  
 \** ***@return*** *\League\Fractal\TransformerAbstract  
 \*/* **public function** getTransformer()  
 {  
 **return new** ProductTransformer();  
 }  
}

Criar ProductTransformer



**<?php  
  
namespace** CodeDelivery\Transformers;  
  
**use** League\Fractal\TransformerAbstract;  
**use** CodeDelivery\Models\Product;  
  
*/\*\*  
 \* Class ProductTransformer  
 \** ***@package*** *namespace CodeDelivery\Transformers;  
 \*/***class** ProductTransformer **extends** TransformerAbstract  
{  
  
 */\*\*  
 \* Transform the \Product entity  
 \** ***@param*** *\Product $model  
 \*  
 \** ***@return*** *array  
 \*/* **public function** transform(Product $model)  
 {  
 **return** [  
 **'id'** => (int) $model->**id**,  
 **'name'** => $model->**name**,  
 **'description'** => $model->**description**,  
 **'price'** => $model->**price** ];  
 }  
}

Chamar ProductPresenter no ProductRepositoryEloquent

**<?php  
  
namespace** CodeDelivery\Repositories;  
  
**use** Prettus\Repository\Eloquent\BaseRepository;  
**use** Prettus\Repository\Criteria\RequestCriteria;  
**use** CodeDelivery\Repositories\ProductRepository;  
**use** CodeDelivery\Models\Product;  
**use** CodeDelivery\Validators\ProductValidator;  
  
*/\*\*  
 \* Class ProductRepositoryEloquent  
 \** ***@package*** *namespace CodeDelivery\Repositories;  
 \*/***class** ProductRepositoryEloquent **extends** BaseRepository **implements** ProductRepository  
{  
  
 *//DESATIVOU O PRESENTER POR PADRÂO  
 //QUANDO QUISER UTILIZAR TERA QUE ATIVAR* **protected $skipPresenter** = **true**;  
 */\*\*  
 \* Specify Model class name  
 \*  
 \** ***@return*** *string  
 \*/* **public function** model()  
 {  
 **return** Product::***class***;  
 }  
  
 **public function** listsOrder()  
 {  
 **return** $this->**model**->get([**'price'**, **'name'** ,**'id'**]);  
 }  
  
 */\*\*  
 \* Boot up the repository, pushing criteria  
 \*/* **public function** boot()  
 {  
 $this->pushCriteria(app(RequestCriteria::***class***));  
 }  
  
 **public function** presenter()  
 {  
 **return** \CodeDelivery\Presenters\ProductPresenter::***class***;  
 }  
  
  
}