Instalando Pusher

<https://laravel.com/docs/5.1/events#broadcasting-events>

**Broadcast Prerequisites**

The following dependencies are needed for event broadcasting:

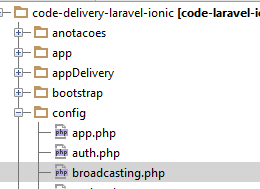
* Pusher: pusher/pusher-php-server ~2.0
* Redis: predis/predis ~1.0

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

$ composer require pusher/pusher-php-server ~2.0

Broadcasting do Laravel

Fazer comunicação com outros tipos de serviços que a gente precisar



Pegar credenciais no dashboar Pusher

KEY  
SECRET  
APP\_ID

Adicionar no .ENV no laravel

PUSHER\_KEY=7da988dd3befceff7414  
PUSHER\_SECRET=db8419ef87e38e4cbe41  
PUSHER\_APP\_ID=302228

Depois das configurações – Configurar um Evento

Criar Evento GetLocationDeliveryman

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

$ php artisan make:event GetLocationDeliveryman

Event created successfully.

**<?php  
  
namespace** CodeDelivery\Events;  
  
**use** CodeDelivery\Events\Event;  
**use** CodeDelivery\Models\Geo;  
**use** CodeDelivery\Models\Order;  
**use** Illuminate\Queue\SerializesModels;  
*//IMPLEMENTAR ShouldBroadcast - QUE O LARAVEL VAI SER QUE VAMOS UTILIZAR O PUSHER***use** Illuminate\Contracts\Broadcasting\ShouldBroadcast;  
  
**class** GetLocationDeliveryman **extends** Event **implements** ShouldBroadcast  
{  
 **use** SerializesModels;  
  
 *//SERIALIZAR O GEO* **public $geo**;  
  
 **private $model**;  
  
 */\*\*  
 \* Create a new event instance.  
 \*  
 \** ***@return*** *void  
 \*/* **public function** \_\_construct(Geo $geo, Order $order)  
 {  
 $this->**geo** = $geo;  
 $this->**model** = $order;  
 }  
  
 */\*\*  
 \* Get the channels the event should be broadcast on.  
 \*  
 \** ***@return*** *array  
 \*/* **public function** broadcastOn()  
 {  
 *//QUANDO O ENTREGADOR SAI RPARA ENTREGAR ELE TEM QUE AAPERTAR O BOTAO  
 //STATUS DA ORDER ENTRA EM TRAG  
 //VOU GERAR UM HASH E ESSE HASH VAI SER O CANAL DE COMUNICAO ENTRE O ENTREGADOR E O CLIENT* **return** [$this->**model**->**hash**];  
 }  
}

Já Criamos o Evento

Vamos Acrescentar o Campo Hash

Criar Migration

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

$ php artisan make:migration --table=orders add\_hash\_to\_orders

Created Migration: 2017\_02\_14\_171933\_add\_hash\_to\_orders

**<?php  
  
use** Illuminate\Database\Schema\Blueprint;  
**use** Illuminate\Database\Migrations\Migration;  
  
**class** AddHashToOrders **extends** Migration  
{  
 */\*\*  
 \* Run the migrations.  
 \*  
 \** ***@return*** *void  
 \*/* **public function** up()  
 {  
 Schema::*table*(**'orders'**, **function** (Blueprint $table) {  
 $table->string(**'hash'**);  
 });  
 }  
  
 */\*\*  
 \* Reverse the migrations.  
 \*  
 \** ***@return*** *void  
 \*/* **public function** down()  
 {  
 Schema::*table*(**'orders'**, **function** (Blueprint $table) {  
 $table->dropColumn(**'hash'**);  
 });  
 }  
}

Rodando a Migration

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

$ php artisan migrate

Migrated: 2017\_02\_14\_171933\_add\_hash\_to\_orders

Schema::*table*(**'orders'**, **function** (Blueprint $table) {  
 $table->string(**'hash'**)->nullable();  
 });

Adicionar campo hash da migration como nullable

Voltar e Criar de Novo

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

$ php artisan migrate:rollback

Rolled back: 2017\_02\_14\_171933\_add\_hash\_to\_orders

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

$ php artisan migrate

Migrated: 2017\_02\_14\_171933\_add\_hash\_to\_orders

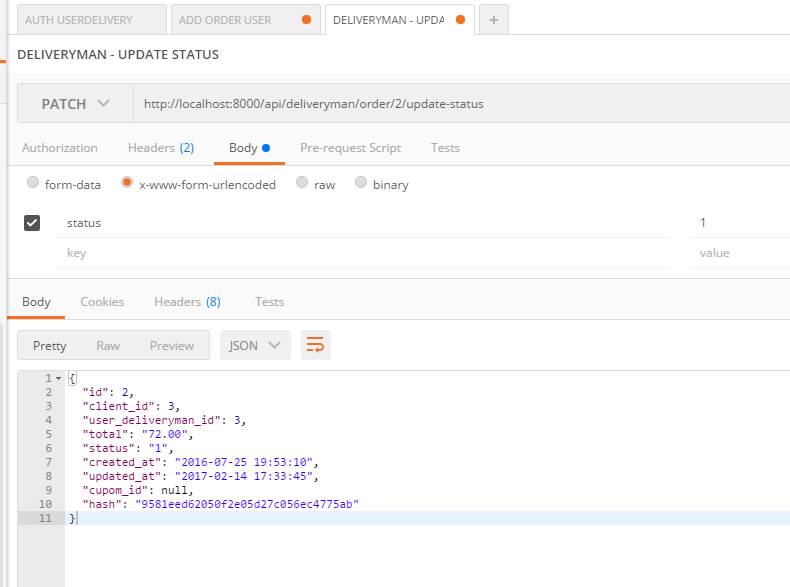
Alterar OrderService

**public function** updateStatus($id, $idDeliveryman, $status)  
**public function** updateStatus($id, $idDeliveryman, $status)  
{  
 $order = $this->**orderRepository**->getByIdAndDeliveryman($id, $idDeliveryman);  
 $order->**status** = $status;  
 *//GERAR HASH CASO NAO TENHA E O ESTATUS FOR ENTREGANDO* **if**((int)$order->**status** == 1 && !$order->**hash**)  
 {  
 $order->**hash** = *md5*((**new** \DateTime())->getTimestamp());  
 }  
 $order->save();  
 **return** $order;  
}

Em Api Controller DeliverymanCheckoutController.php

**public function** updateStatus(Request $request, $id)  
{  
 $idDeliveryman = Authorizer::*getResourceOwnerId*();  
 **return** $this->**orderService**->updateStatus($id, $idDeliveryman, $request->get(**'status'**));  
}

Testando



Em Api Deliveryman ChecoutController

Adicionar Evento

**public function** geo(Request $request, Geo $geo, $id)  
{  
 $idDeliveryman = Authorizer::*getResourceOwnerId*();  
 $order = $this->**orderRepository**->getByIdAndDeliveryman($id, $idDeliveryman);  
 $geo->**lat** = $request->get(**'lat'**);  
 $geo->**long** = $request->get(**'long'**);  
 event(**new** GetLocationDeliveryman($geo, $order));  
 **return** $geo;  
}

Testando o Evento

