MEMÓRIAS DAS TRINCHEIRAS

TECNOLOGIA

ARQUITETURA
>
FRAMEWORK (OU LINGUAGEM)

SOA

“Service-Oriented Architecture (SOA) é um estilo de arquitetura de software cujo princípio fundamental prega que as funcionalidades implementadas devem ser disponibilizadas na forma de serviços. Frequentemente estes serviços são conectados através de um "barramento" que disponibiliza interfaces, ou contratos, acessíveis através de web services ou outra forma de comunicação entre aplicações”
Wikipedia

12 FATORES

» Codebase

» Dependencies

» Config

» Backing Services

» Build, release, run

» Processes

» Port binding

» Concurrency

» Disposability

» Dev/prod parity

» Logs

» Admin processes

```php
namespace Inventory\Service\Trade\Order;

use Core\Service\InvalidParameterException;
use Core\Service\ParameterSet;
use Inventory\Model\OrderType;
use Person\Model\Person as PersonModel;
use Person\Service\Person\Person as PersonService;
use Inventory\Model\TradeStatusType;

class Purchase extends Order
{
    protected function doPost(ParameterSet $params)
    {
        $result = parent::doPost($params);
        $this->getEventManager()->trigger(
            'PURCHASE_CREATED',
            $this,
            [
                'trade' => $this->entityResult,
            ]
        );

        return $result;
    }

    public function doUpdate(ParameterSet $params)
    {
        $result = parent::doUpdate($params);
        $this->getEventManager()->trigger(
            'PURCHASE_UPDATED',
            $this,
            [
                'tradeBefore' => $tradeBefore,
                'tradeAfter' => $this->entityResult,
            ]
        );
```

EVENTOS

» Zend Event Manager

INJEÇÃO DE DEPENDÊNCIAS

```php
private function assertBuyerHasCreditLimit(ParameterSet $params)
{
    if (!$params->has('paymentMode')) {
        return true;
    }

    $trade = $this->buildTradeFromParameterSet($params);
    $paymentMode = $params->get('paymentMode')->getValue();

    $price = ($trade->getPrice()) ? $trade->getPrice() : $this->calcPriceOfTrade($trade);
    $this->serviceLocator
        ->get(\Inventory\Service\Trade\Order\PaymentMode::class)
        ->setTotalPrice($price)
        ->setPaymentMode($trade->getPaymentMode())
        ->setRange(isset($paymentMode['range']) ? $paymentMode['range'] : [])
        ->validate();
```

» Zend Service Manager

NÃO CRIE AMARRAS

BANCO DE DADOS

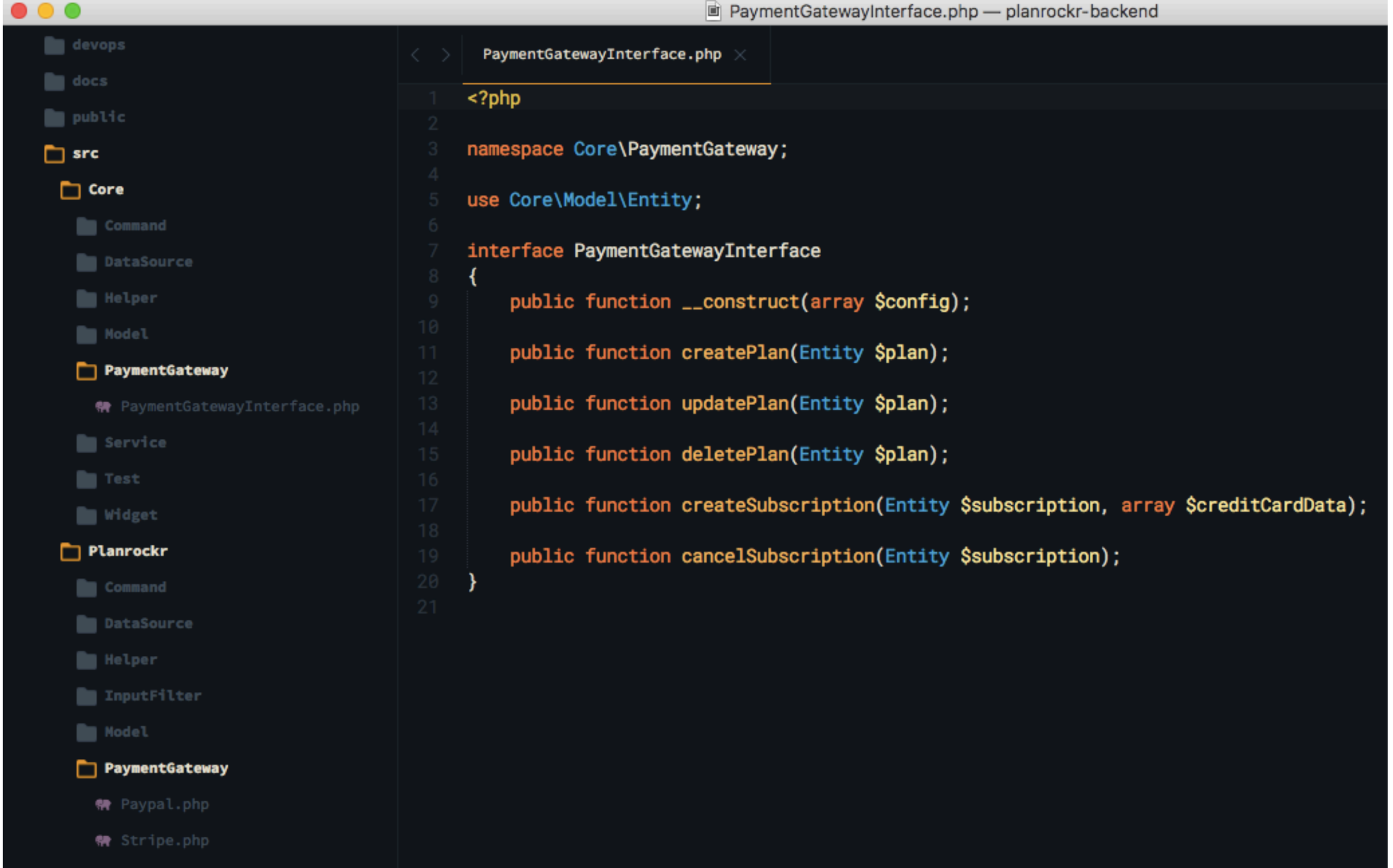
» Use um DBAL/ORM (Doctrine)

» Não crie lógica em triggers ou procedures no banco de dados

SERVIÇOS EXTERNOS

» Podem mudar, novos podem ser adicionados

» Interfaces!


PaymentGatewayInterface.php — planrockr-backend
devops
docs
public
src
Core
Command
DataSource
Helper
Model
PaymentGateway
PaymentGatewayInterface.php
Service
Test
Widget
Planrockr
Command
DataSource
Helper
InputFilter
Model
PaymentGateway
Paypal.php
Stripe.php
PaymentGatewayInterface.php
<?php
namespace Core\PaymentGateway;
use Core\Model\Entity;
interface PaymentGatewayInterface
{
public function __construct(array $config);
public function createPlan(Entity $plan);
public function updatePlan(Entity $plan);
public function deletePlan(Entity $plan);
public function createSubscription(Entity $subscription, array $creditCardData);
public function cancelSubscription(Entity $subscription);
}

```
$subscription->setGatewaySubscriptionId(
    $this->app['payment.gateway']->createSubscription(
        $subscription, $parameters['creditCard']
    )
);
```

DESENVOLVA PARA PADRÕES

» PSR-3: logs (Monolog)

» PSR-7: mensagens http (Zend Expressive)

» PSR-11: dependency injection containers (Zend Service Manager)

» PSR-15: middlewares (Zend Expressive 2)

NÃO SE APAIXONE POR LINGUAGEM OU FRAMEWORK

» PHP (Zend Framework, Drupal, Zend Expressive, Silex)

» Go

» JavaScript (AngularJS, React, React Native)

» Objective C

» Java (Android)

» Shell Script

CUIDADO COM MODISMOS

» Serviços X micro serviços

» Single Page Aplications X Sites estáticos

» Banco relacional X NoSQL

» O-novo-framework(JavaScript?)-da-semana

MÉTRICAS

COMPLEXIDADE, PADRÕES, SEGURANÇA, COBERTURA DE TESTES

» Pdepend

» PhpMetrics

» PHPUnit

» phpcs

AUTOMATIZE TUDO

» Build (Buildkite)

» Deploy (Deploybot)

» Testes (PHPUnit, Codeception)

» Análise de código (Ebert)

ATUALIZE-SE SEMPRE

» Eventos internos (Coderockr Jam)

» Meetups

» Eventos (PubTalks)

PROJETOS

AGILIDADE AO EXTREMO

» Coderockr Way = Kanban + InnerSource + ConvDev

» Slack

MÉTRICAS

LEAD TIME, CYCLE TIME, RESPONSE TIME, TAKT TIME

» Planrockr ;)

NEGÓCIOS

NÃO TESTE EM PROJETOS DE CLIENTES

» Love or Hate (conexão entre apps)

» Orçamentos (Silex)

» Planrockr (MongoDB, Go, React, Docker)

NÃO TENHA RECEIO DE "PERDER" SEUS MELHORES

Uma banda é a junção de todos os talentos dos seus músicos

SERVIÇOS É F#%A!

Mas é possível ter sucesso e uma equipe incrível!

COMO CRESCER?

» melhores clientes

» melhores processos

» automação de processos

» melhorar as pessoas

Clientes contratam uma empresa de serviços por dois motivos: para desenvolver algo que eles não querem fazer ou para desenvolver algo que eles não são especialistas.

Qual destes cenários você acha melhor??

LINKS

https://blog.planrockr.com
https://blog.coderockr.com
https://www.youtube.com/user/coderockr

CONTATO

http://eltonminetto.net
http://asemanaphp.com.br
@eminetto
eminetto@coderockr.com