
# PHP moderno

## e a saga contra as amarras dos frameworks (e bibliotecas)

yksilva

in/yksilva

Yuri Silva

# Antes de tudo...

Alinhamento de expectativas

# O que essa palestra NÃO é sobre:

# O que essa palestra é sobre:

# O que essa palestra é sobre:

TDD

# O que essa palestra é sobre:

SOLID

TDD

# O que essa palestra é sobre:

SOLID

TDD

**Design Patterns**

# O que essa palestra é sobre:

SOLID

TDD

**KISS** 

**Design Patterns**

# O que essa palestra é sobre:

SOLID

**KISS** 

TDD

Design Patterns

**Abstrações**

# O que essa palestra é sobre:

AR
QUI
TE
TURA

SOLID

TDD

**Abstrações**

**KISS** 

**Design Patterns**

# O que essa palestra é sobre:

AR
QUI
TE
TURA

SOLID

TDD

**Abstrações**

**KISS**

**Design Patterns**

# BOAS PRÁTICAS

# PHP

# PHP e os marcos para sua modernidade

# PHP e os marcos para sua modernidade

5.6 ⟶ 7.0

- Scalar type declarations
- Return type declarations

https://www.php.net/manual/en/migration70.new-features.php

# PHP e os marcos para sua modernidade

7.3 ⟶ 7.4

Typed properties

https://www.php.net/manual/en/migration74.new-features.php

# PHP e os marcos para sua modernidade

7.4 ⟶ 8.0

- Named Arguments
- Attributes
- Constructor Property Promotion
- Union Types

[...]

https://www.php.net/manual/en/migration80.new-features.php

# Tipagem

# Outros fatores de sucesso do PHP


COMPOSER


COMPOSER


Slim
a micro framework for PHP


COMPOSER


Slim
a micro framework for PHP

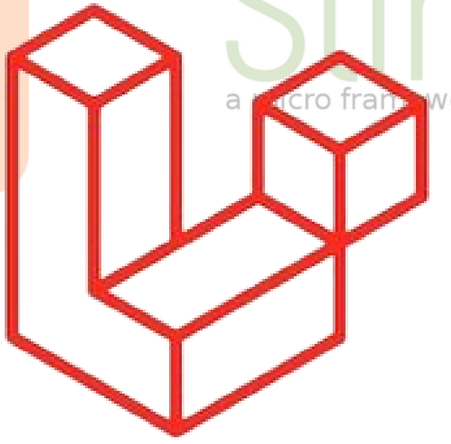

COMPOSER


PHP-FIG


Slim
a micro framework for PHP


COMPOSER


PHP-FIG




Laravel




# Mas...

Até que ponto é saudável
estar dependente
de uma ferramenta?


Running feature
@programmerbay
New changes

# Algumas **ades** do nosso interesse

# Algumas **ades** do nosso interesse

Versatilidade

# Algumas **ades** do nosso interesse

Versatilidade

Manutenibilidade

# Algumas **ades** do nosso interesse

Versatilidade

Manutenibilidade

Reusabilidade

# Algumas **ades** do nosso interesse

Versatilidade

Manutenibilidade

Reusabilidade

Testabilidade

# Algumas **ades** do nosso interesse

Versatilidade

Manutenibilidade

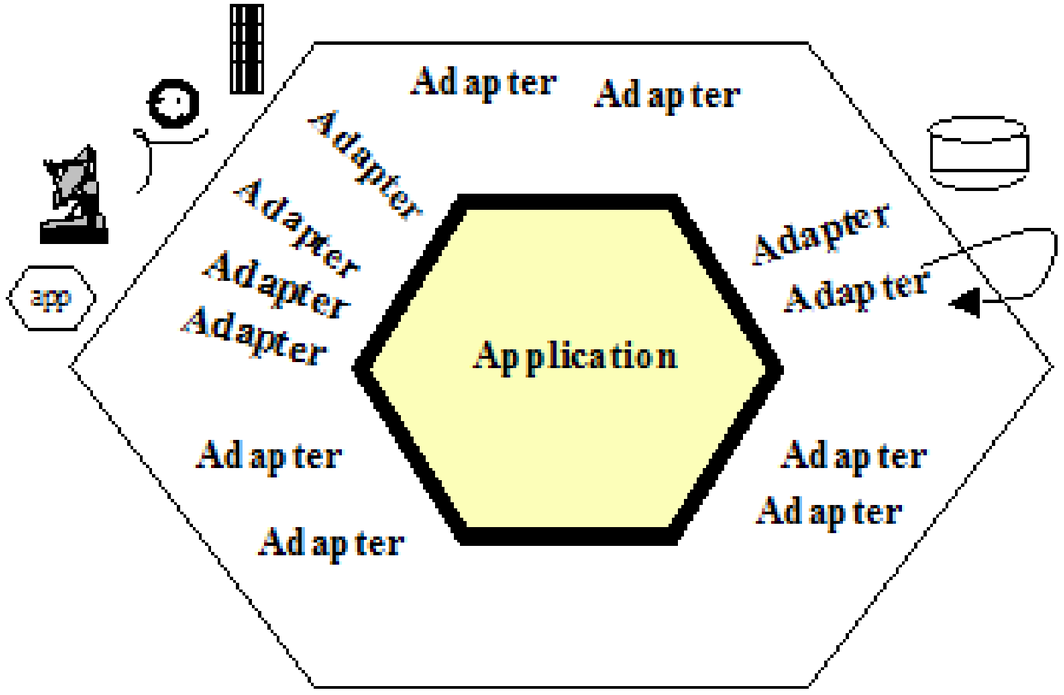
**Modularidade**

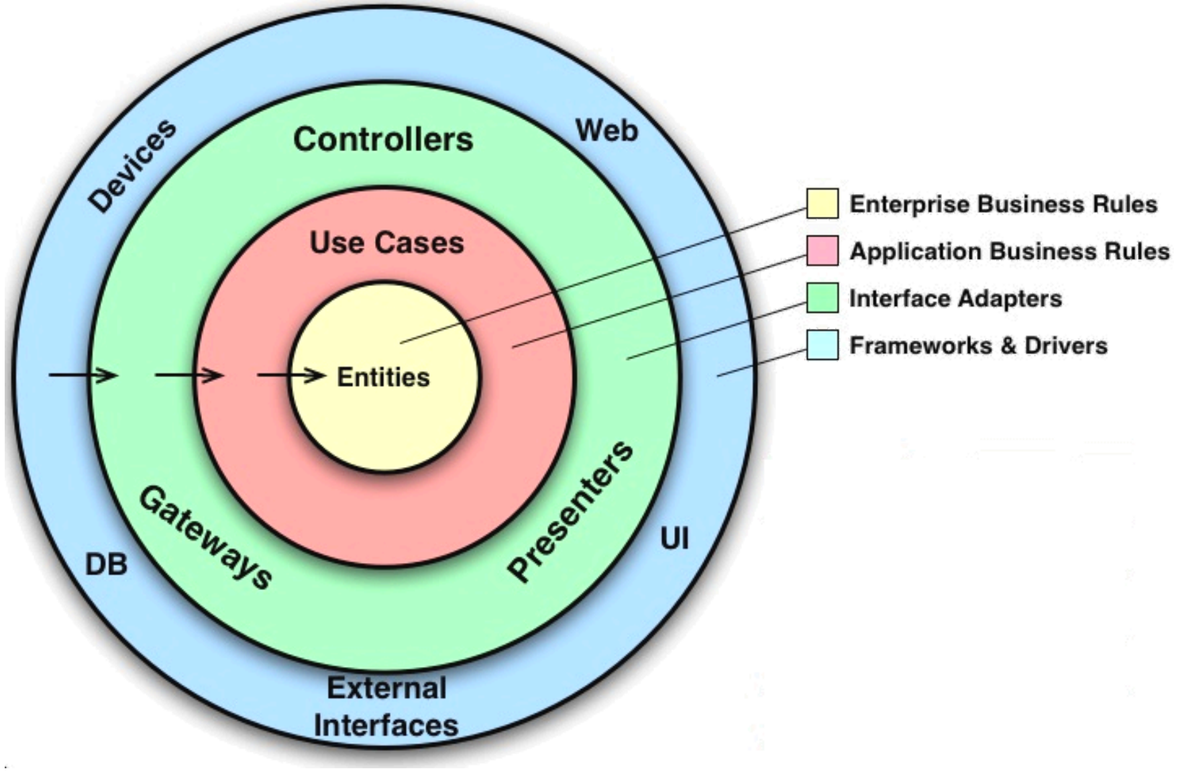
Reusabilidade

Testabilidade

# Isolamento

# Independência


Adapter
Adapter
Adapter
Adapter
Adapter
Adapter
Adapter
Adapter
Adapter
Adapter
Adapter
Adapter
Adapter
app
Application


Devices
Controllers
Web
Use Cases
Entities
Gateways
Presenters
DB
UI
External Interfaces
Enterprise Business Rules
Application Business Rules
Interface Adapters
Frameworks & Drivers

Como o **PHP** moderno
nos ajuda?

# Abstrações

# Situação-problema

**Contexto de negócio**

- Processamento de pagamentos

**Desafio da Engenharia:**

Estruturar um módulo versátil para integrar com diversos gateways

# Ok!

# Vamos recorrer às bibliotecas

Omnipay MULTI-GATEWAY PAYMENT PROCESSING LIBRARY

Presented by The League of Extraordinary Packages



# Introduction

source league/omnipay | 404 badge not found | license MIT | packagist v3.3.0 | downloads 14.83 M

Omnipay is a payment processing library for PHP. It has been designed based on ideas from Active Merchant, plus experience implementing dozens of gateways for CI Merchant. It has a clear and consistent API, is fully unit tested, and even comes with an example application to get you started.

## Why use Omnipay?

So, why use Omnipay instead of a gateway's official PHP package/example code?

- Because you can learn one API and use it in multiple projects using different payment gateways
- Because if you need to change payment gateways you won't need to rewrite your code
- Because most official PHP payment gateway libraries are a mess
- Because most payment gateways have exceptionally poor documentation
- Because you are writing a shopping cart and need to support multiple gateways

https://omnipay.thephpleague.com

Details

github.com/thephpleague/omnipay
Homepage
Source
Issues

Fund package maintenance!
barryvdh

| | |
|---|---|
| Installs: | 7 047 028 |
| Dependents: | 156 |
| Suggesters: | 5 |
| Security: | 0 |
| Stars: | 5 929 |
| Watchers: | 247 |
| Forks: | 927 |
| Open Issues: | 126 |
| Type: | metapackage |

https://packagist.org/packages/league/omnipay

Details

github.com/thephpleague/omnipay-c...
Homepage
Source
Issues

Fund package maintenance!
barryvdh

| | |
|---|---|
| Installs: | 14 849 552 |
| Dependents: | 1 246 |
| Suggesters: | 1 |
| Security: | 0 |
| Stars: | 330 |
| Watchers: | 24 |
| Forks: | 244 |
| Open Issues: | 52 |

https://packagist.org/packages/omnipay/common

```php
use Omnipay\Omnipay;

$gateway = Omnipay::create('Stripe');
$gateway->setApiKey('abc123');

$formData = array('number' => '4242424242424242', 'expiryMonth' => '6', 'expiryYear' => '2030', 'cvv' => '123');
$response = $gateway->purchase(array('amount' => '10.00', 'currency' => 'USD', 'card' => $formData))->send();

if ($response->isRedirect()) {
    // redirect to offsite payment gateway
    $response->redirect();
} elseif ($response->isSuccessful()) {
    // payment was successful: update database
    print_r($response);
} else {
    // payment failed: display message to customer
    echo $response->getMessage();
}
```

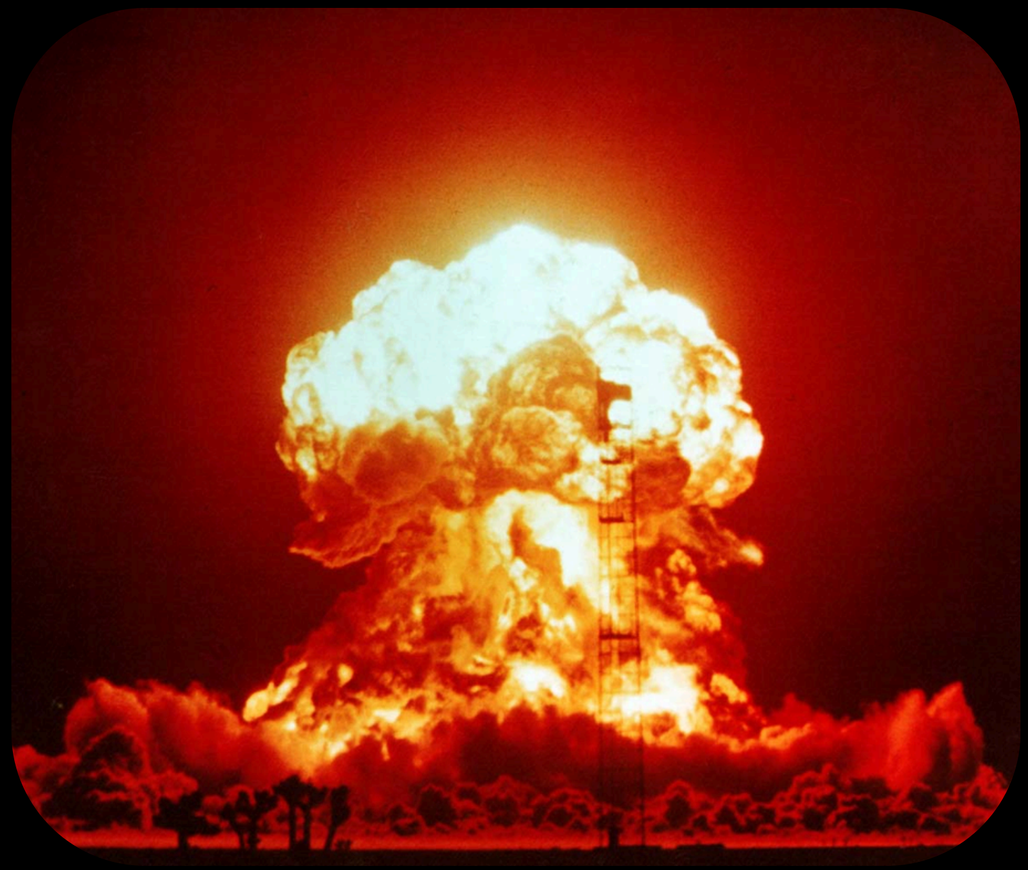
https://packagist.org/packages/league/omnipay

# Error handling

You can test for a successful response by calling `isSuccessful()` on the response object. If there was an error communicating with the gateway, or your request was obviously invalid, an exception will be thrown. In general, if the gateway does not throw an exception, but returns an unsuccessful response, it is a message you should display to the customer. If an exception is thrown, it is either a bug in your code (missing required fields), or a communication error with the gateway.

You can handle both scenarios by wrapping the entire request in a try-catch block:

https://omnipay.thephpleague.com/api/error-handling



```
die('kboom!');
```

# Recurring billing

At this stage, automatic recurring payments functionality is out of scope for this library. This is because there is likely far too many differences between how each gateway handles recurring billing profiles. Also in most cases token billing will cover your needs, as you can store a credit card then charge it on whatever schedule you like. Feel free to get in touch if you really think this should be a core feature and worth the effort.

https://omnipay.thephpleague.com/api/recurring-billing

# DO IT YOURSELF!