Instalando

```
curl -s http://getcomposer.org/installer | p
hp
php composer.phar create-project zendframewo
rk/zend-expressive-skeleton restbeer
```

Escolher:
[3]

- Zend ServiceManager
- FastRoute
- Zend View
- Woops

Testando

```
cd restbeer
php -S localhost:8080 -t public
```

Modelos

Copiar o beers.db de http://cl.ly/2e473b2M2k1Z e salvar no diretório do projeto

```
cd restbeer
php ../composer.phar require zendframework/z
end-db
```

Escolher:
[1]
[y]

Criar o src/App/src/Model/Beer.php

```
<?php
namespace App\Model;

use Zend\InputFilter\InputFilter;

class Beer
{
    public $id;
    public $name;
    public $style;
    public $img;

    /**
     * Configura os filtros dos campos da cl
```

```
asse
     *
     * @return Zend\InputFilter\InputFilter
     */
    public function getInputFilter()
    {
        $inputFilter = new InputFilter();

        $inputFilter->add([
            'name'     => 'id',
            'required' => false,
            'filters'  => [
                ['name' => 'Int'],
            ],
        ]);

        $inputFilter->add([
            'name'     => 'name',
            'required' => true,
            'filters'  => [
                ['name' => 'StripTags'],
                ['name' => 'StringTrim'],
            ],
            'validators' => [
                [
                    'name'    => 'StringLeng
th',
                    'options' => [
                        'encoding' => 'UTF-8
',
                        'min'      => 1,
                        'max'      => 100,
                    ],
                ],
            ],
        ]);

        $inputFilter->add([
            'name'     => 'style',
            'required' => true,
            'filters'  => [
                ['name' => 'StripTags'],
                ['name' => 'StringTrim'],
            ],
            'validators' => [
                [
                    'name'    => 'StringLeng
th',
                    'options' => [
                        'encoding' => 'UTF-8
',
                        'min'      => 1,
                        'max'      => 100,
                    ],
                ],
            ],
        ]);
        $inputFilter->add([
            'name'     => 'img',
            'required' => false,
            'filters'  => [
                ['name' => 'StripTags'],
                ['name' => 'StringTrim'],
            ],
        ]);

        return $inputFilter;
    }
}
```

Configurando

Criar o config/autoload/db.global.php e incluir:

```
    <?php
    //https://zendframework.github.io/zend-db/adapter/#creating-an-adapter-using-configuration
    return [
        'db' => [
            'driver' => 'Pdo_Sqlite',
            'database' => 'beers.db',
        ],
    ];
```

Alterar o config/autoload/dependencies.global.php e incluir:

```
// Use 'factories' for services provided by callbacks/factory classes.
        'factories' => [
            Application::class => ApplicationFactory::class,
            Helper\UrlHelper::class => Helper\UrlHelperFactory::class,
                      //adicionar
            App\Factory\Db\Adapter\Adapter::class => App\Factory\Db\Adapter\Adapter::class
        ],
```

Criar o src/App/src/Factory/Db/Adapter/Adapter.php com:

```
<?php

namespace App\Factory\Db\Adapter;
```

```php
use Interop\Container\ContainerInterface;
use Zend\Db\Adapter\Adapter as ZendAdapter;

class Adapter
{
    public function __invoke(ContainerInterface $container)
    {
        $config = $container->get('config');
        return new ZendAdapter($config['db']);
    }
}
```

Crud de cervejas

Configurar as rotas

No config/routes.php:

```php
<?php

$app->get('/', App\Action\HomePageAction::class, 'home');
$app->get('/api/ping', App\Action\PingAction::class, 'api.ping');
```

```
$app->get('/beer', App\Action\Beer\Index::class, 'beer.index');
$app->get('/beer/{id}', App\Action\Beer\View::class, 'beer.view');
$app->post('/beer', App\Action\Beer\Create::class, 'beer.create');
$app->put('/beer/{id}', App\Action\Beer\Update::class, 'beer.update');
$app->delete('/beer/{id}', App\Action\Beer\Delete::class, 'beer.delete');
```

Adicionar as classes em config/autoload/dependencies.global.php, dentro do array de factories:

```
App\Action\HomePageAction::class => App\Action\HomePageFactory::class,
App\Action\Beer\Index::class => App\Factory\Action\Beer::class,
App\Action\Beer\Update::class => App\Factory\Action\Beer::class,
App\Action\Beer\Create::class => App\Factory\Action\Beer::class,
App\Action\Beer\Delete::class => App\Factory\Action\Beer::class,
```

Criar a factory

Criar o src/App/src/Factory/Action/Beer.php

```php
<?php

namespace App\Factory\Action;

use Interop\Container\ContainerInterface;
use Zend\Db\TableGateway\TableGateway;

class Beer
{
    public function __invoke(ContainerInterface $container, $requestedName)
    {
        $adapter = $container->get('App\Factory\Db\Adapter\Adapter');
        $tableGateway = new TableGateway('beer', $adapter);

        return new $requestedName($tableGateway);
    }
}
```

Criar as actions

Criar o src/App/src/Action/Beer/Index.php

```
<?php

namespace App\Action\Beer;

use Interop\Http\ServerMiddleware\DelegateIn
terface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\Middleware
Interface;

class Index implements MiddlewareInterface
{
    private $tableGateway;

    public function __construct($tableGatewa
y)
    {
        $this->tableGateway    = $tableGatewa
y;
    }

    public function process(ServerRequestInt
erface $request, DelegateInterface $delegate
)
    {
        $beers = $this->tableGateway->select
()->toArray();

        return $delegate->process(
            $request->withParsedBody($beers)
        );
    }
}
```

Criar o src/App/src/Action/Beer/Create.php

```
<?php

namespace App\Action\Beer;

use Interop\Http\ServerMiddleware\DelegateIn
terface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\Middleware
Interface;

class Create implements MiddlewareInterface
{
    private $tableGateway;
    public function __construct($tableGatewa
y)
    {
        $this->tableGateway   = $tableGatewa
y;
    }

    public function process(ServerRequestInt
erface $request, DelegateInterface $delegate
)
    {
        $data = $request->getParsedBody();

        $this->tableGateway->insert($data);

        return $delegate->process(
            $request->withParsedBody(['id' =
> $this->tableGateway->getLastInsertValue()]
)
        )->withStatus(201);
    }
}
```

Criar o src/App/src/Action/Beer/Update.php

```
<?php
```

```
namespace App\Action\Beer;

use Interop\Http\ServerMiddleware\DelegateInterface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\MiddlewareInterface;

class Update implements MiddlewareInterface
{
    private $tableGateway;

    public function __construct($tableGateway)
    {
        $this->tableGateway   = $tableGateway;
    }

    public function process(ServerRequestInterface $request, DelegateInterface $delegate)
    {
        $id = $request->getAttribute('id');
        $beer = $this->tableGateway->select(['id' => $id]);
        if (count($beer) == 0) {
            return $delegate->process($request)
```

```
                                                  ->withStatus(
404);
        }

        parse_str(file_get_contents("php://i
nput"), $data);
        $this->tableGateway->update($data, [
'id' => $id]);

        return $delegate->process(
            $request->withParsedBody(['id' =
> id])
        );
    }
}
```

Criar o src/App/src/Action/Beer/Delete.php

```
<?php

namespace App\Action\Beer;

use Interop\Http\ServerMiddleware\DelegateIn
terface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\Middleware
Interface;

```

```
class Delete implements MiddlewareInterface
{
    private $tableGateway;

    public function __construct($tableGatewa
y)
    {
        $this->tableGateway   = $tableGatewa
y;
    }

    public function process(ServerRequestInt
erface $request, DelegateInterface $delegate
)
    {
        $id = $request->getAttribute('id');
        $beer = $this->tableGateway->select(
['id' => $id]);
        if (count($beer) == 0) {
            return $delegate->process($reque
st)
                                ->withStatus(
404);
        }

        $this->tableGateway->delete(['id' =>
 $id]);

        return $delegate->process($request)
```

```
                         ->withStatus(204);
    }
}
```

Serialização

Alterar o config/pipeline.php

```php
<?php

use Zend\Expressive\Helper\ServerUrlMiddleware;
use Zend\Expressive\Helper\UrlHelperMiddleware;
use Zend\Expressive\Middleware\ImplicitHeadMiddleware;
use Zend\Expressive\Middleware\ImplicitOptionsMiddleware;
use Zend\Expressive\Middleware\NotFoundHandler;
use Zend\Stratigility\Middleware\ErrorHandler;

$app->pipe(ErrorHandler::class);
$app->pipe(ServerUrlMiddleware::class);
$app->pipeRoutingMiddleware();
$app->pipe(ImplicitHeadMiddleware::class);
```

```
$app->pipe(ImplicitOptionsMiddleware::class)
;
$app->pipe(UrlHelperMiddleware::class);
$app->pipeDispatchMiddleware();
$app->pipe(App\Middleware\Format\Json::class
);
$app->pipe(App\Middleware\Format\Html::class
);
$app->pipe(NotFoundHandler::class);
```

e config/autoload/dependencies.global.php

```
'dependencies' => [
    'invokables' => [
        App\Middleware\Format\Json::class =>
 App\Middleware\Format\Json::class,
    ],
    'factories' => [
        App\Middleware\Format\Html::class =>
 App\Factory\Middleware\Format\Html::class
    ],
],
```

Criar o src/App/src/Middleware/Format/Json.php

```php
<?php

namespace App\Middleware\Format;

use Interop\Http\ServerMiddleware\DelegateIn
terface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\Middleware
Interface;
use Zend\Diactoros\Response\JsonResponse;

class Json implements MiddlewareInterface
{
    public function process(ServerRequestInt
erface $request, DelegateInterface $delegate
)
    {
        $content = $request->getParsedBody()
;
        $header = $request->getHeader('accep
t');
        $accept = null;
        if (isset($header[0])) {
            $accept = $header[0];
        }
        if (!$accept || $accept != 'applicat
ion/json') {
                return $delegate->process($r
equest);
```

```
        }

        return new JsonResponse($content);
    }
}
```

Criar o
src/App/src/Factory/Middleware/Format/Html.php

```php
<?php

namespace App\Factory\Middleware\Format;

use Interop\Container\ContainerInterface;
use Zend\Expressive\Template\TemplateRendererInterface;
use App\Middleware\Format\Html as HtmlMiddleware;

class Html
{
    public function __invoke(ContainerInterface $container)
    {
        return new HtmlMiddleware($container->get(TemplateRendererInterface::class));
    }
}
```

Instalar o Zend View

```
php ../composer.phar require zendframework/z
end-expressive-zendviewrenderer
```

Criar o src/App/src/Middleware/Format/Html

```

<?php

namespace App\Middleware\Format;

use Interop\Http\ServerMiddleware\DelegateIn
terface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\Middleware
Interface;
use Zend\Expressive\Template\TemplateRendere
rInterface;
use Zend\Diactoros\Response\HtmlResponse;

class Html implements MiddlewareInterface
{
    private $template;

    public function __construct($template)
    {
        $this->template = $template;
    }

    public function process(ServerRequestInterface $request, DelegateInterface $delegate)
    {
        $content = $request->getParsedBody();

            return new HtmlResponse($this->template->render('beer::index', ['content' => $content]));
    }
}
```

Criar src/App/templates/beer/index.phtml

```
<table class="table table-striped">
  <thead>
    <tr>
      <th>#</th>
      <th>Id/th>
      <th>Name</th>
      <th>Style</th>
      <th>Img</th>
    </tr>
  </thead>
  <tbody>
```

```
    <?php foreach ($this->content as $beer):
 ?>
      <tr>
        <th scope="row"><?php echo $beer['id
'];?></th>
        <td><?php echo $beer['name'];?></td>
        <td><?php echo $beer['style'];?></td
>
        <td><img src="<?php echo $beer['img'
];?>"></td>
      </tr>
    <?php endforeach; ?>
  </tbody>
</table>
```

Adicionar os templates em src/App/src/ConfigProvider.php

```
    public function getTemplates()
    {
        return [
            'paths' => [
                'app'    => [__DIR__ . '/../
templates/app'],
                'error'  => [__DIR__ . '/../
templates/error'],
                'layout' => [__DIR__ . '/../
templates/layout'],
                    'beer' => [__DIR__ . '/../te
mplates/beer'],
               ],
          ];
     }
```

Validando

Instalar o zend-validator e o zend-filter

```
php ../composer.phar require zendframework/z
end-validator zendframework/zend-filter zend
framework/zend-inputfilter
```

Criar o src/App/src/Middleware/Validate.php

```
<?php

namespace App\Middleware;

use Interop\Http\ServerMiddleware\DelegateIn
terface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\Middleware
Interface;
```

```
use Zend\Diactoros\Response\JsonResponse;
use App\Model\Beer;

class Validate implements MiddlewareInterfac
e
{
    public function process(ServerRequestInt
erface $request, DelegateInterface $delegate
)
    {
        $beer = new Beer;
        $inputFilter = $beer->getInputFilter
();
        $inputFilter->setData($this->parseBo
dy($request));
        if (!$inputFilter->isValid()) {
            $errors = [];
            foreach ($inputFilter->getMessag
es() as $key => $values) {
                $messages = [];
                foreach ($values as $message
) {
                    $messages[] = $message;
                }
                $errors[] = [
                    'input' => $key,
                    'messages' => $messages,
                ];
            }
            throw new \Exception("Error Proc
essing Request", 422);
        }

        return $delegate->process($request);
    }

    private function parseUri($request)
    {
        return substr($request->getUri()->ge
tPath(), 0, 5);
    }

    private function parseBody($request)
    {
        switch ($request->getMethod()) {
            case 'POST':
                return $request->getParsedBo
dy();
                break;
            case 'PUT':
                parse_str(file_get_contents(
"php://input"),$data);
                return $data;
                break;
        }

        return [];
    }
```

```
}
```

Alterar o dependencies.global.php e colocar no invokables

```
App\Middleware\Validate::class => App\Middleware\Validate::class,
```

e colocar o middleware no config.pipeline.php

```php
<?php

use Zend\Expressive\Helper\ServerUrlMiddleware;
use Zend\Expressive\Helper\UrlHelperMiddleware;
use Zend\Expressive\Middleware\ImplicitHeadMiddleware;
use Zend\Expressive\Middleware\ImplicitOptionsMiddleware;
use Zend\Expressive\Middleware\NotFoundHandler;
use Zend\Stratigility\Middleware\ErrorHandler;

$app->pipe(ErrorHandler::class);
$app->pipe(ServerUrlMiddleware::class);
```

```php
$app->pipeRoutingMiddleware();
$app->pipe(ImplicitHeadMiddleware::class);
$app->pipe(ImplicitOptionsMiddleware::class);
$app->pipe(UrlHelperMiddleware::class);
$app->pipe(App\Middleware\Validate::class);
$app->pipeDispatchMiddleware();
$app->pipe(App\Middleware\Format\Json::class);
$app->pipe(App\Middleware\Format\Html::class);
$app->pipe(NotFoundHandler::class);
```

Authenticação

Criar o src/App/src/Middleware/Auth.php

```php
<?php
namespace App\Middleware;

use Interop\Http\ServerMiddleware\DelegateInterface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\MiddlewareInterface;
use Zend\Diactoros\Response\JsonResponse;
```

```php
class Auth implements MiddlewareInterface
{
    public function process(ServerRequestInterface $request, DelegateInterface $delegate)
    {
        if(! $request->hasHeader('authorization')){
            return $response->withStatus(401);
        }

        if (!$this->isValid($request)) {
            return $response->withStatus(403);
        }

        return $delegate->process($request);
    }

    private function isValid(ServerRequestInterface $request)
    {
        $token = $request->getHeader('authorization');
        //validar o token de alguma forma...
        return true;
    }
}
```

Adicionar em config/pipeline.php

```
$app->pipe(App\Middleware\Auth::class);
$app->pipe(App\Middleware\Validate::class);
$app->pipeDispatchMiddleware();
```

Desta forma todas as rotas vão executar primeiro este Middleware

Trabalho