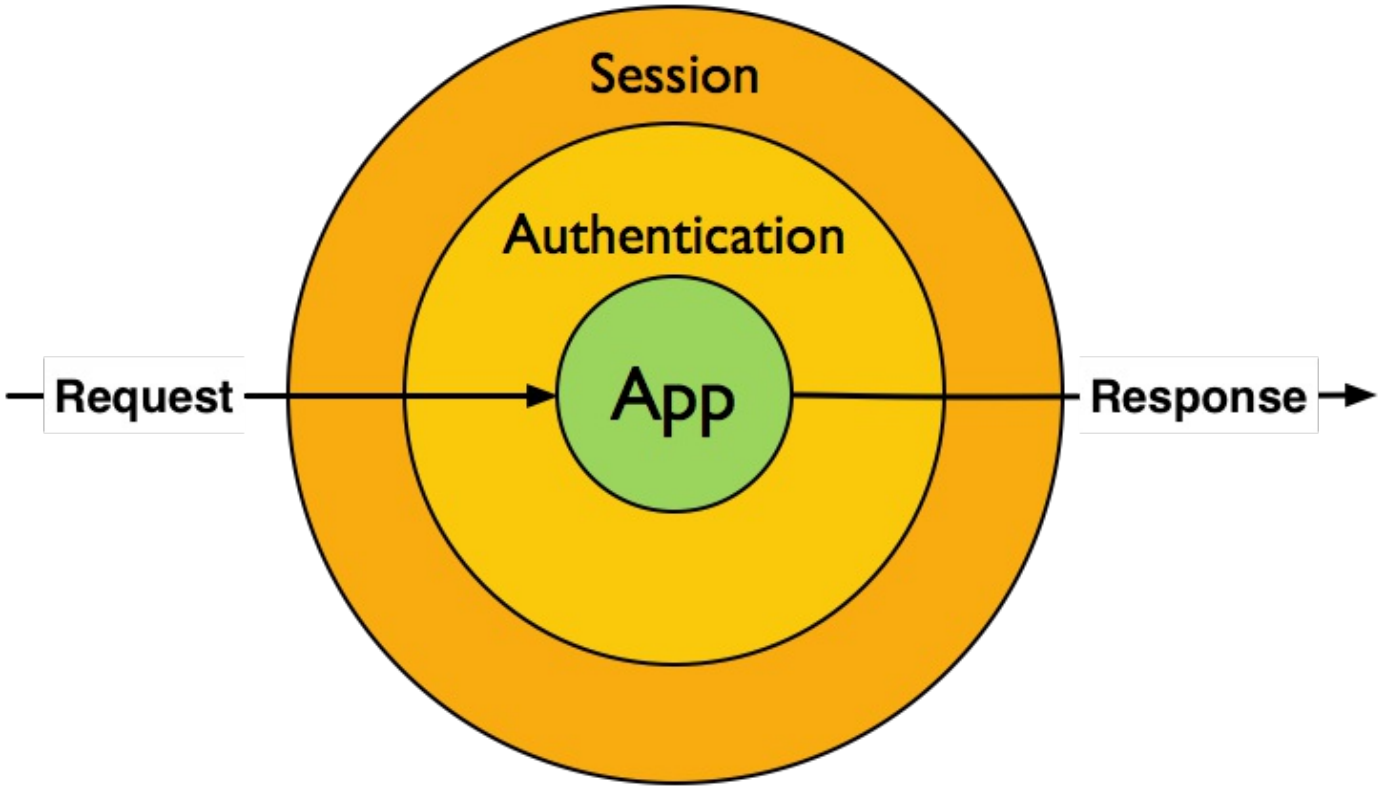
Middlewares

Middlewares são códigos intermediários entre a requisição, a sua aplicação e a resposta para o cliente. Este padrão passou a ser implementado em inúmeros frameworks php's e passou a ser aceito o conceito pela **PSR-7**.



Como no exemplo acima, os blocos `Session` (sessão) e `Authentication` (autenticação) envolvem a `App` (aplicação). Este padrão é chamado de `Onion style` (estilo cebola), pois cada camada da cebola é como se fosse uma camada de `middleware`.

Os middlewares são trechos de códigos que recebem uma `Request` (requisição), uma `Response` (resposta) e uma `callable` (função que representa o próximo middleware).

```php
use Psr\Http\Message\ResponseInterface as Response;
use Psr\Http\Message\RequestInterface as Request;

function (
    Request $request,   // the request
    Response $response, // the response
    callable $next      // the next middleware
) {

    if ($next) {
        return $next($request,$response);
    }

    return $response;
}
```

Vamos traduzir a imagem a o código acima em palavras.

O lado do **cliente** cria uma `request` (requisição) para o uri **exemplo.dev/app** , essa requisição vai passar por cada camada.

Ao encontrar a rota disponível na aplicação, verifica que existem 2 middlewares e antes de escrever o `echo 'Hello World'` irá executar as middlewares nas ordens específicas.

```php
// $request - representa uma nova requisição.
```

```
// Middlewares - SessionMiddleware.php
function __invoke (Request $request, Response $response, ca
llable $next) {
    ...
    /*
        Executa tudo relacionado a Sessões e então antes de
 retornar a resposta faz uma verificação se existe um outro
 middleware que deva ser executado.
    */
    if ($next) { // nesse caso, verifica que existe o Authe
nticationMiddleware e o chamaa
        return $next($request,$response);
    }
    return $response;
};

// Middlewares - AuthenticationMiddleware.php
function __invoke (Request $request, Response $response, ca
llable $next) {
    ...
    /*
        Executa tudo relacionado a Authenticação e então an
tes de retornar a resposta faz uma verificação se existe um
 outro middleware que deva ser executado.
    */
    if ($next) { // nesse caso, verifica que existe o bloco
 relacionado ao contexto da aplicação e chama o bloco que i
rá escrever o echo 'Hello World'
```

```
        return $next($request,$response);
    }
    return $response;
};

// public/index.php
$app->get('/app',["SessionMiddleware","AuthenticationMiddlewares"],function (Request $request, Response $response) {
    echo 'Hello world';
});
```

O trecho do `function () { ... }` em `$app->get('/app',function() { ... }` também é tratado como um middleware, mas como sendo o `terminal middleware` (middleware terminal/final), ou seja, o último middleware que deverá ser executado.

Contudo, todos os middlewares podem ser terminais desde que o `$next` seja `NULL` e irá seguir para a próxima instrução que retornará a `response` (resposta).