

## NAMESPACE PHP:

Iniciamos composer sobre el directorio raíz:

```
PS C:\daw2RSH\desarrollo_servidor\Primera evaluación\AP2> composer init -n --name "daw2_raulsales/ap2-mve" []
```

Cuando ejecutamos esto se crea la siguiente estructura de ficheros:

```
> composer
autoload.php
{} composer.json
{} composer.lock
```

-Modificamos el fichero composer.json con las credenciales del nombre de proyecto:

```
{ } composer.json > ...
1  {
2      "name": "daw2_raulsales/ap2-mve",
3      "require": {},
4      "autoload": {
5          "psr-4": {
6              "app\\": "src/"
7          }
8      }
9  }
10
```

-Ahora ejecutamos en la terminal lo siguiente:

```
PS C:\daw2RSH\desarrollo_servidor\Primera evaluación\AP2> composer install[]
```

AHORA VAMOS A PONER UN EJEMPLO DE NAMESPACE

EJEMPLOS DE USO DE NAMESPACE:

MODIFICAREMOS EL REQUIRE SOBRE EL AUTOLOAD DE COMPOSER Y LE DIREMOS QUE NAMESPACE VAMOS A UTILIZAR

```

1  <?php
2  require_once "../vendor/autoload.php";
3
4  $accion=$_GET["action"];
5  $n=$_GET["n"];
6  use app\controller\{controllerbye,controllerhello,controllersaying};
7
8  switch($accion){
9      case "saying":
10         //require "../controller/controllersaying.php";
11         $controllerSaying = new controllersaying();
12         $controllerSaying -> sayingWord($n-1);
13         break;
14      case "hello":
15         //require "../controller/controllerhello.php";
16         $controllerHello = new controllerhello();
17         $controllerHello -> helloTime();
18         break;
19      case "bye":
20         //require "../controller/controllerbye.php";
21         $controllerbye = new controllerbye();
22         $controllerbye -> byeTime();
23         break;
24      default:
25         echo "hay un error cara pinga";
26         break;
27  }
28  /* Numero aleatorio: $posicion=rand(0,count($sayings) -1);
29  echo $sayings[$posicion]; */
30

```

-EJEMPLO DE UN CONTROLLER:

src > controller > controllerbye.php > ...

```
1  <?php
2  namespace app\controller;
3
4  use app\model\timemodel;
5  use app\view\GenericView;
6
7  class controllerbye
8  {
9      public function byeTime()
10     {
11         $timeBack = new timemodel();
12         //require_once "../view/genericview.php";
13         $genericViewBye = new GenericView();
14         $currentTime = $timeBack->getCurrentTime();
15         $genericViewBye -> genericView("Adios vro ", $currentTime);
16     }
17 }
18
```

AP2

- > clases
- > config
- ▼ public
  - 🐘 index.php M
- ▼ src
  - ▼ controller
    - 🐘 controllerbye... M
    - 🐘 controllerhell... M
    - 🐘 controllersayi... M
  - ▼ model
    - 🐘 sayingmodel.... M
    - 🐘 timemodel.php M
  - ▼ view
    - 🐘 genericview.p... M
    - 🐘 sayingview.php M
- > templates
- ▼ vendor
  - > composer
    - 🐘 autoload.php M
  - { } composer.json
  - { } composer.lock

src > model > sayingmodel.php > PHP Intelephense > { } app\model

```
1  <?php
2  namespace app\model;
3  class sayingmodel{
4      public function getSayings($n){
5          $sayings =[
6              "A quien madruga, Dios le ayuda",
7              "No hay mal que por bien no venga",
8              "De tal palo, tal astilla",
9              "En casa del herrero cuchara de palo",
10             "El que no corre, vuela",
11             "A lo hecho, pecho",
12             "Ojo por ojo, diente por diente",
13             "A rey muerto, rey puesto"
14         ];
15         return $sayings[$n];
16     }
17 }
18
19 ?>
```

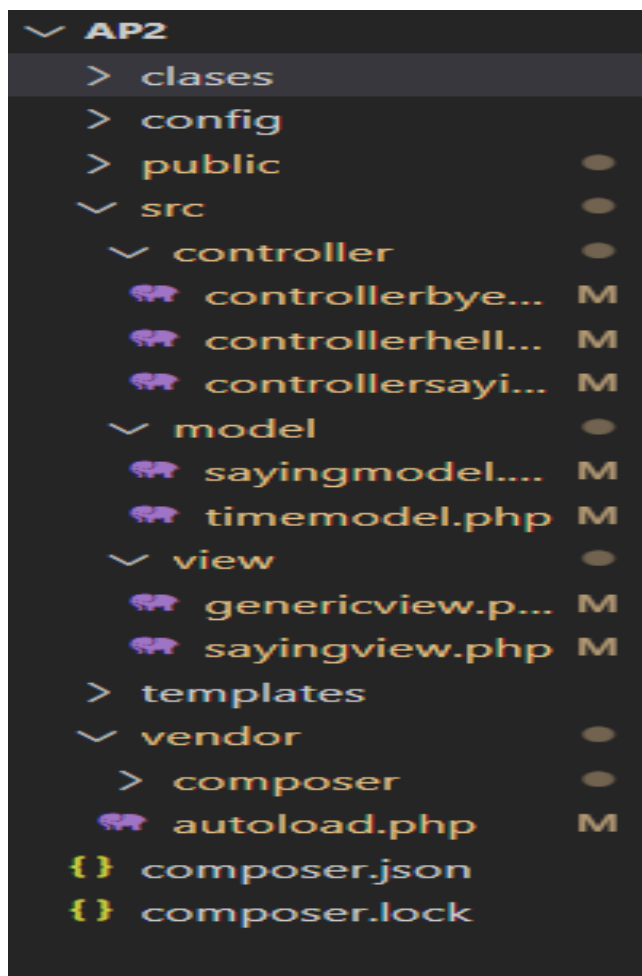
The screenshot shows the VS Code interface with a project named 'AP2'. The Explorer sidebar on the left displays the following structure:

- AP2
  - clases
  - config
  - public
    - index.php M
  - src
    - controller
      - controllerbye... M
      - controllerhell... M
      - controllersayi... M
    - model
      - sayingmodel.... M
      - timemodel.php M
    - view
      - genericview.p... M
      - sayingview.php M
  - templates

The main editor shows the file 'genericview.php' with the following PHP code:

```
src > view > genericview.php > PHP Intelephense > {} app\view
1  <?php
2  /* //soltamos chorro de código html en una variable
3  $template = file_get_contents('templates/template.html');
4  //le pasamos a clave el contenido del nombre de array asociado
5  foreach ($sayings as $refran=>$contenido) {
6      //el '{' delimita lo que contenga {.
7      $template = str_replace('{'.$refran.'}', $contenido, $template);
8  };
9  echo $template; */
10 namespace app\view;
11
12 class genericview
13 {
14     public function genericView($greetings, $currentTime)
15     {
16         echo $greetings . $currentTime;
17     }
18 }
```

LA ESTRUCTURA QUEDA TAL QUE ASÍ:



-IMPORTANTE: UTILIZAR composer dump-autoload CUANDO TENGAMOS LOS CAMBIOS SOBRE EL DIRECTORIO RAÍZ DE LA CARPETA.

EN ESTE CASO NUESTRO DIRECTORIO RAÍZ ES AP2 Y LO EJECUTAMOS AHÍ.

```
PS C:\daw2RSH\desarrollo_servidor\Primera evaluación\AP2\public> cd ..
PS C:\daw2RSH\desarrollo_servidor\Primera evaluación\AP2> composer dump-autoload
Generating autoload files
Generated autoload files
```

-FUNCIONA:

