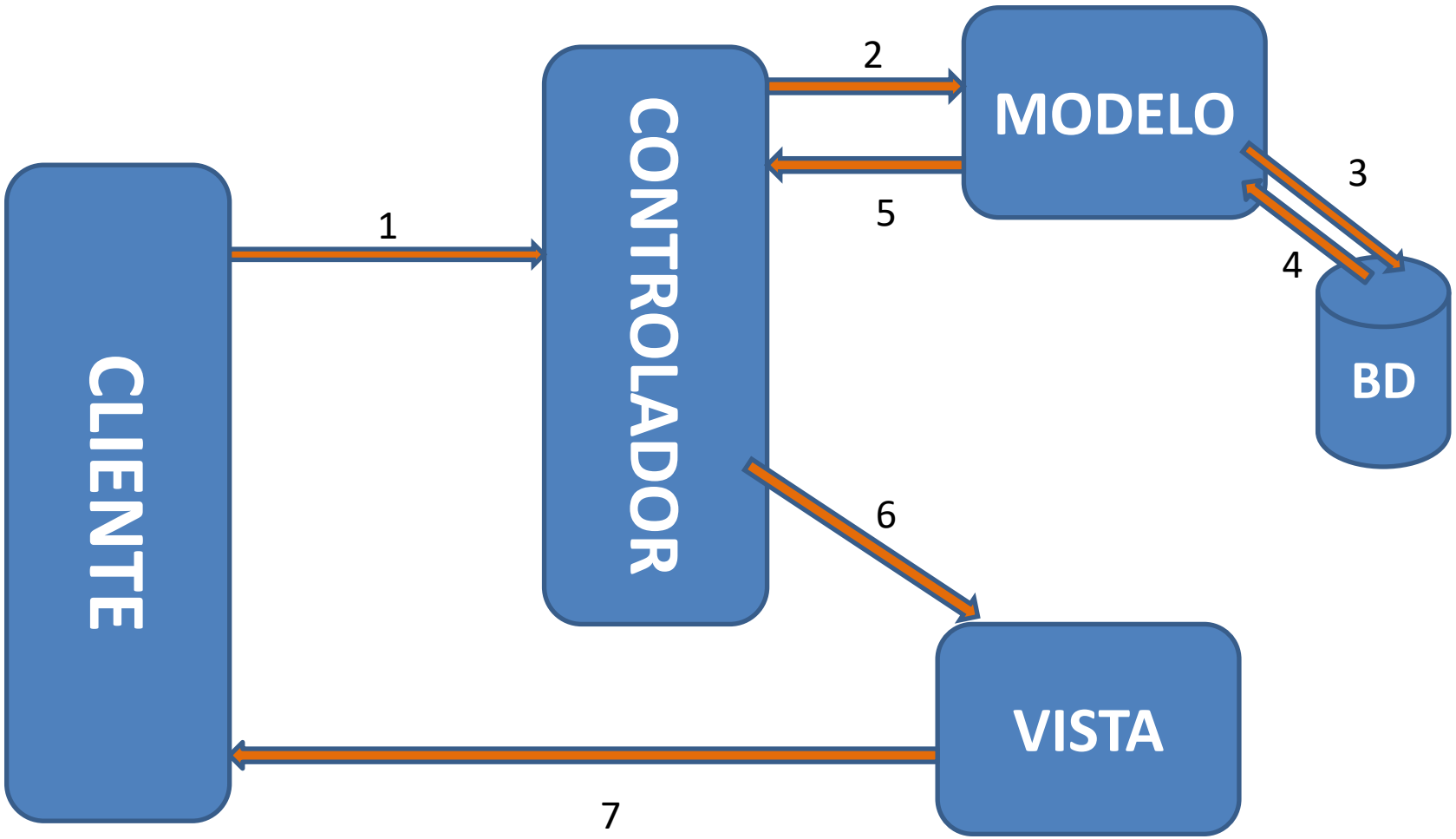
Modelo Vista Controlador

Antonio Espín Herranz

Esquema MVC



Secuencia

1. El cliente hace una petición / solicita una acción a la aplicación. Puede ser de tipo GET / POST.
2. El controlador la analiza y le solicita al modelo unos datos.
3. y 4. El modelo conecta con la BD y ejecuta la sentencia SQL necesaria para recuperar los datos.
5. El modelo devuelve los datos al controlador. En forma de objetos.
6. El controlador selecciona la lista y le pasa la información.
7. La vista se devuelve al cliente en distintos formatos: HTML, XML, JSON, PDF, etc.

3 Partes

- El **controlador**:
  - Es el cerebro de la aplicación, también recibe el nombre de controlador frontal.
  - Organiza el flujo de la aplicación. Hace las veces de guardia de tráfico, centraliza todas las peticiones. Los datos que necesita se los solicita al modelo y después pasa la información a la vista.
  - Soporta la lógica de la aplicación.

3 Partes

- El **Modelo**:
  - Representa la lógica de negocio. Responde a la aplicación ¿qué hace la aplicación?
  - Se suele aplicar un patrón DAO: Data Access Object para acceder a la BD.
  - Y también se utilizan clases ayudantes / beans o value Object para representar los objetos de datos: el cliente, el proveedor, el usuario …

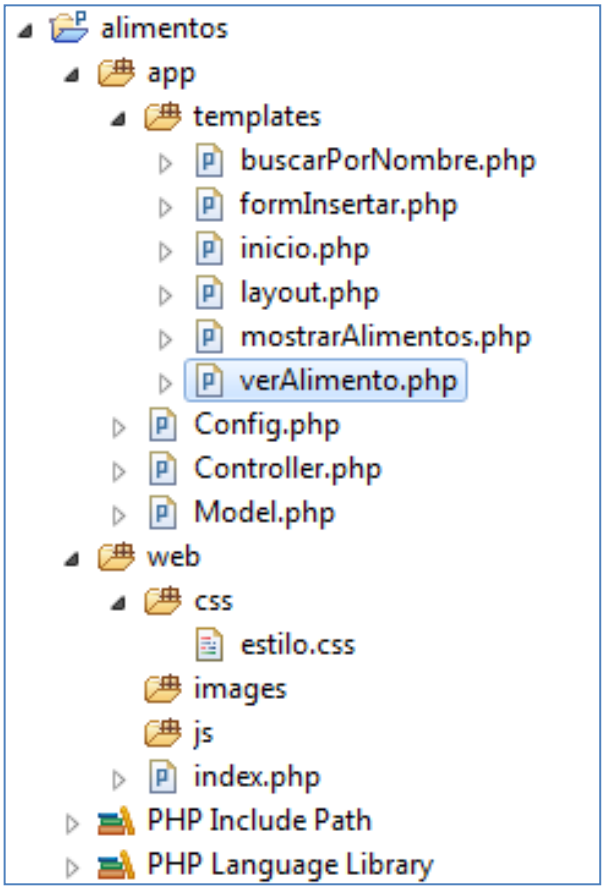
3 Partes

- La **vista**:
  - Recibe la información del controlador, representada por objetos y se encargar de pintar los datos en el navegador en formato adecuado o requerido.

Ejemplo

- **MVC**:
  - App web para el control de los alimentos.
  - Con las siguientes operaciones:

URL	Acción
http://tu.servidor/alimentos/index.php?ctl=inicio	mostrar pantalla inicio
http://tu.servidor/alimentos/index.php?ctl=listar	listar alimentos
http://tu.servidor/alimentos/index.php?ctl=insertar	insertar un alimento
http://tu.servidor/alimentos/index.php?ctl=buscar	buscar alimentos
http://tu.servidor/alimentos/index.php?ctl=ver&id=x	ver el alimento *x*



Base de datos

- BD:
  - La tabla **alimento** tendrá los siguientes campos:
    - `**id**` int(11) **NOT NULL** AUTO_INCREMENT,
    - `**nombre**` varchar(255) **NOT NULL**,
    - `**energia**` decimal(10,0) **NOT NULL**,
    - `**proteina**` decimal(10,0) **NOT NULL**,
    - `**hidratocarbono**` decimal(10,0) **NOT NULL**,
    - `**fibra**` decimal(10,0) **NOT NULL**,
    - `**grasatotal**` decimal(10,0) **NOT NULL**,

index

- La página **index**:
  - Carga el modelo, el controlador y la configuración (parámetros de acceso a la BD).
  - Mediante un array asociativo almacena las acciones y el controlador asociado, según la petición que venga.
  - En caso de que la encuentre la lanza y si no, será un error.
  - Utiliza las funciones de PHP:
    - **method_exists**(objeto, nombre_metodo)
      - http://php.net/manual/es/function.method-exists.php
      - Para comprobar si existe el método en el controlador.
    - **call_user_func**(call_back, argumentos)
      - http://php.net/manual/es/function.call-user-func.php
      - Llamar y retornar un método del objeto y se le pueden pasan argumentos.

Templates

- **Ejemplo de plantilla:**

```
<?php ob_start() ?>
//Activa el almacenamiento en búfer de la salida, no vuelca nada.
http://php.net/manual/es/function.ob-start.php

 <h1>Inicio</h1>
<h3> Fecha: <?php echo $params['fecha'] ?> </h3>
<?php echo $params['mensaje'] ?>
<?php $contenido = ob_get_clean() ?>
// Obtener el contenido del búfer actual y eliminar el búfer de salida actual
http://php.net/manual/es/function.ob-get-clean.php

<?php include 'layout.php' ?>
```

- La variable $contenido vuelca todo el contenido generado después dentro del layout.
- Se pasa del buffer a la variable.

Controller

- El controlador se representa por una clase "**Controller**" que se encarga de tener tantos métodos públicos como acciones tenga la web.

- Cada método del controlador representa una acción.
  - En cada acción **accede al modelo** si necesita datos y por último **llama la vista (template)** que muestra la información.
    - **require __DIR__ . '/templates/una vista.php';**
  - Cada acción carga un **array** llamado **params** para pasar información a la vista.

Layout

- Es una página con el código HTML que incrusta el contenido.
- Esta mantiene el menú y la estructura principal de la app.
- Siempre vuelca en el contenido a través de una variable que se ha cargado en el template.