**PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL**

**FORMATO GUÍA DE APRENDIZAJE**

**IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE**

- **Denominación del Programa de Formación:** TGO Análisis y Desarrollo de Sistemas de Información
- **Código del Programa de Formación:** 228106 V102
- **Nombre del Proyecto:** Construcción de un sistema de información que cumpla con los requerimientos del cliente en procesos que se lleven a cabo en el sector productivo del departamento de Caldas
- **Fase del Proyecto:** IMPLEMENTACIÓN
- **Actividad de Proyecto:** Seleccionar la alternativa de solución que cumpla con los requerimientos establecidos por el cliente
- **Competencia:** Construir el sistema que cumpla con los requisitos de la solución informática.
- **Resultados de Aprendizaje Alcanzar:** Realizar la codificación de los módulos del sistema y el programa principal, a partir de la utilización del lenguaje de programación seleccionado, de acuerdo con las especificaciones del diseño
- **Duración de la Guía:** 40 horas

**2. PRESENTACIÓN**

*PHP e*s un lenguaje de programación del lado del servidor y de alto nivel, embebido en páginas HTML. Ha tenido una gran aceptación en la comunidad de desarrolladores, debido a la potencia y simplicidad que lo caracterizan, así como al soporte generalizado en la mayoría de los servidores de hosting, hasta los más simples y económicos.

Otra de las claves del éxito de PHP es que la mayoría de los CMS más populares (WordPress, Joomla!, Drupal) y los sistemas de comercio electrónico (Prestashop, Woocommerce, Magento), así como otros cientos de herramientas, están desarrollados en PHP. Por lo tanto, usar PHP es sinónimo de ser capaz de introducirnos en muchas herramientas gratuitas y de código abierto para realizar cualquier cosa en el ámbito de la web.

**3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE**

- **Descripción de la(s) Actividad(es):**
    - **Actividades de aprendizaje:**
        - Nueva sintaxis de PHP 8

- Dominar la creación y uso de funciones en PHP para modularizar el código y reutilizarlo.
- Familiarizarse y utilizar variables superglobales como $_GET, $_POST y $_SESSION
- Crear formularios HTML y procesar los datos enviados por los usuarios utilizando PHP 8
- Conexión y manipulación de bases de datos MySQL incluyendo la ejecución de consultas SQL y la protección contra ataques de inyección SQL
- Comprender los conceptos de POO en PHP, como clases, objetos, propiedades y métodos
- Manejo de errores y excepciones de manera efectiva en PHP para depurar y mejorar la robustez del código
- Implementar sistemas de autenticación de usuarios y control de acceso en aplicaciones web PHP 8.

- **Actividad de Reflexión inicial**

**Actividad de reflexión inicial**

Antes de comenzar nuestro proyecto en PHP 8, los aprendices deberán conocer la evolución de PHP. Puede parecer que no, pero PHP ha cambiado muchísimo en este tiempo. Vamos a ver cómo era construir una clase con PHP 5.3 y cómo lo es con PHP 8.2:

**De PHP 5.3 a PHP 8.2**
https://www.youtube.com/watch?v=BHAYO6esXlw

- **Actividades de contextualización e identificación de conocimientos necesarios para el aprendizaje**

## Definición de la función editar y eliminar en la aplicación

En este punto ya tenemos una aplicación que nos permite adicionar y listar elementos de los diferentes modelos con sus respectivos controladores. Es hora de implementar la función `editar` y la función `eliminar` en nuestro proyecto.

Para ello modificaremos la lógica de nuestro `public/index.php` y comenzaremos eliminando todo el siguiente código:

```php
// Verificación de las rutas válidas
if(array_key_exists($url, $routes)){
    $appRoute = $routes[$url];
    $controllerName = $appRoute['controller'];#Definimos controlador
    $action = $appRoute['action'];#Definimos la acción
    // Verificamos que el controlador y la acción existan
    if(class_exists($controllerName) && method_exists($controllerName,$action)){
        // Instanciar el controlador
        $controller = new $controllerName;
        $controller->$action();
        exit;
    }
}
```

Y lo reemplazamos primero por el siguiente código:

```php
# Verificar si la ruta es válida
$matchedRoute = null;
foreach ($routes as $route => $routeConfig) {
    # preg_match realiza una coincidencia de patrón con una expresión regular
    if (preg_match("#^$route$#", $url, $matches)) {
        $matchedRoute = $routeConfig;
        break;
    }
}
```

La función `preg_match("#^$route$#", $url, $matches)` se utiliza para realizar una coincidencia de patrón con expresiones regulares en PHP. Así es cómo funciona:

- **preg_match:** Es una función de PHP que realiza una coincidencia de patrón con una expresión regular.
- **"#^$route$#":** Es la expresión regular que se utilizará para realizar la coincidencia de patrón. La expresión regular está delimitada por caracteres **#**, y **^** y **$** representan el comienzo y el final de la cadena, respectivamente. `$route` es la variable que contiene el patrón que se va a comparar.
- **$url:** Es la cadena en la que se realizará la coincidencia de patrón.
- **$matches:** Es una variable de salida opcional que contendrá los resultados de la coincidencia. Después de llamar a preg_match, los resultados de la coincidencia se almacenarán en el arreglo `$matches`.

Entonces, al utilizar `preg_match("#^$route$#", $url, $matches)`, estamos verificando si la cadena `$url` coincide con el patrón definido en `$route`. Si hay una coincidencia, el resultado se almacenará en `$matches`.

*Esta función es comúnmente utilizada para verificar si una URL coincide con un patrón determinado y tomar acciones en consecuencia, como enrutamiento en aplicaciones web o análisis de URLs estructuradas.*

*Las expresiones regulares son una poderosa herramienta para realizar coincidencias y manipulaciones de texto, pero también pueden ser complejas. Es importante comprender cómo funcionan y asegurarse de utilizarlas de manera adecuada y segura en el código.*

Después del anterior bloque de código, crearemos este condicional:

```php
if ($matchedRoute) {
    $controllerName = $matchedRoute['controller'];
    $action = $matchedRoute['action'];

    if (class_exists($controllerName) && method_exists($controllerName, $action)) {
        // Obtener los parámetros capturados de la URL
        $parameters = array_slice($matches, 1);

        // Instanciar el controlador y llamar a la acción con los parámetros
        $controller = new $controllerName();
        $controller->$action(...$parameters);
        exit;
    }
}
```

`(...$parameters)` pasa los parámetros a la acción del controlador utilizando la sintaxis de desempaquetado de argumentos (...). Esto permite pasar un número variable de parámetros a la acción del controlador.

*El desempaquetado de argumentos, también conocido como "splat operator" en PHP, es una característica que permite pasar un número variable de argumentos a una función o método. Permite que un array o iterable se desempaquete en argumentos individuales.*

El desempaquetado de argumentos nos permite pasar un número variable de argumentos a una función o método, tomando los valores de un array o iterable y pasándolos como argumentos individuales. Esto es útil cuando se necesita pasar un conjunto variable de argumentos a una función o método sin tener que especificarlos individualmente.

Luego hacemos pruebas para ver que todo sigue funcionando correctamente, es decir, miramos todos los controladores y acciones y todo debería seguir funcionando como antes.

Luego de estas pruebas, vamos al archivo `$routes` y configuaramos la siguiente ruta:

```
'/paciente/show/(\d+)/(\d+)' => [
    'controller' => 'App\Controllers\PacienteController',
    'action' => 'getPaciente'
]
```

En este ejemplo, la ruta `/paciente/show/(\d+)` utiliza una expresión regular para capturar un valor numérico en la URL. Este valor se pasará como parámetro a la acción `getPaciente` del controlador `PacienteController`.

---

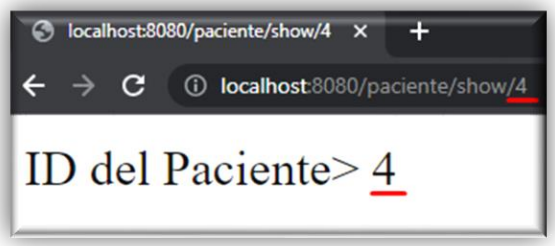
*Recordemos que el patrón de la expresión regular `(\d+)` coincide con cualquier número entero de un dígito o más.*

---

Ahora incluimos un nuevo método en `PacienteController` llamado `getPaciente` para obtener la información de un paciente. El método recibe el ID que llega a través de la URL y que posteriormente fue procesado y obtenido por la lógica del archivo `public/index.php`

```
public function getPaciente($id){
    echo "ID del Paciente> $id";
}
```

Si accedemos a través del navegador con la ruta: `/paciente/show/4` podemos ver que corresponde al número de id de paciente que queremos consultar. Por ahora lo único que está haciendo el método es mostrar el número



**Actividad 1**

Los aprendices deberán incluir al Proyecto **Senamás_EPS**, la vista para visualizar la información de un paciente específico. Para ello se deberá crear la vista requerida y también modificar el nuevo método **getPaciente**, para llamar al modelo correspondiente y traer la información del paciente para posteriormente incluirla en la vista de editar

**Actividad 2**

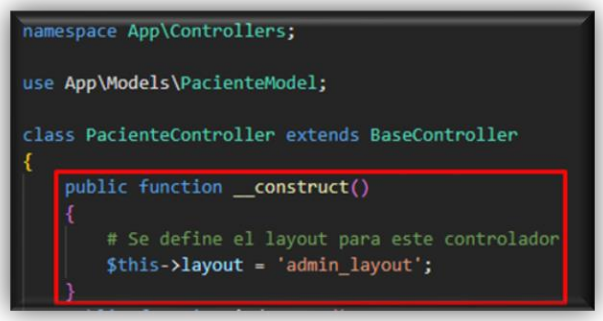
Ahora los aprendices deberán realizar las acciones necesarias para que cuando el usuario le de clic al botón "**Editar**", los nuevos datos del paciente sean editados en la base de datos. Para ello también se deberá crear la ruta correspondiente en el Controlador de Paciente para cumplir con dicha funcionalidad.

# Definición de plantilla o layout del proyecto

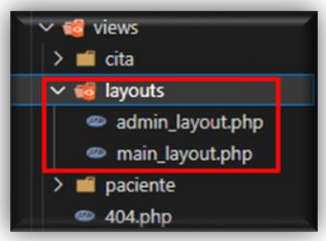
Actualmente nuestra estructura de proyecto formativo tiene varias vistas que están repitiendo el código lo cual se puede constituir en un problema porque a la hora de hacer algún cambio si tenemos por ejemplo 20 vistas tendremos que cambiar el código en 20 archivos. Para resolver este problema integraremos a nuestra base de proyecto una plantilla o layout que nos permitirá optimizar este proceso.

Primero se define en el controlador base, un atributo protegido `$layout` que nos permite determinar la plantilla general y por defecto que tendrá el proyecto.


Luego modificamos el controlador específico y creamos un constructor donde se define la plantilla específica para utilizar en dicho controlador. En el caso de `PacienteController`, se incluirá la plantilla *"*`admin_layout.php`*"*.




La carpeta `views` ahora tendrá una carpeta 'layouts' donde guardaremos todas las plantillas que requiera el proyecto.




Cambiaremos nuestra **vista** del ejemplo y solo dejaremos el contenido como tal (El DIV container) para renderizar los registros de pacientes y el formulario de ingreso. El archivo `paciente/indexView.php` quedará de la siguiente:

```php
<div class="container">
    <h1>Lista pacientes</h1>
    <p>
        <?php
        foreach ($patients as $item) :
            echo "<br>Nombre: $item->nombre";
        endforeach;
        ?>
    </p>
</div>
```

La definición de nuestro layout tendrá todos los elementos comunes a todas las vistas, es decir, que tendrá las etiquetas `<html>, <head> y <body>` al igual que la definición de los estilos, scripts y pie de página (Footer) que se repetirá en todas las vistas. Utilizaremos el siguiente código para poder incluir la vista dentro del layout.

```php
<!-- Agregar aquí el contenido de la vista específica -->
<?php include_once $content; ?>
```

El siguiente código irá en el archivo `admin_layout.php`:

```php
<!-- layout.php -->
<!DOCTYPE html>
<html>
<head>
    <title><?php echo $title; ?></title>
    <!-- Agregra aquí estilos y scripts comunes -->
</head>
<body>
    <!-- Agregar aquí el contenido de la vista específica -->
    <?php include_once $content; ?>

    <!-- Agregar aquí pie de página común -->
    <footer>
        <!-- Contenido del pie de página -->
    </footer>
</body>
</html>
```

Ahora cambiamos el método `render` del controlador base, donde se incluirá directamente el layout, y a su vez el layout incluirá el contenido de la vista requerida a través de la variable `$content`

```php
protected function render($view, $data = []): void
{
    // Lógica para renderizar la vista
    //extract($data);
    foreach ($data as $key => $value) {
        # Se extraen y crean las variables traidas en $data
        $$key = $value;
    }

    $content = MAIN_APP_ROUTE . "views/$view" . "View.php";
    $layout = MAIN_APP_ROUTE . "views/layouts/{$this->layout}.php";

    include_once $layout;
}
```

La acción `index` de nuestro controlador `PacienteController`, quedará de la siguiente manera (tener en cuenta que ahora se está enviando el valor `title`:

```php
public function index()
{
    // Crear una instancia del modelo Paciente
    $pacienteObj = new \App\models\PacienteModel();
    // Obtener todos los pacientes desde el modelo
    $pacientes = $pacienteObj->getAll();
    // Pasar los datos a la vista
    $data = [
        'title' => 'Lista de pacientes',
        'patients' => $pacientes
    ];
    # Renderizar la vista a través del método de BaseController
    $this->render('paciente/index', $data);
}
```

**Actividad 3**

Los aprendices deberán incluir al Proyecto base **Senamás_EPS**, toda la parte de **layouts** tal como está especificado en esta guía, teniendo el CRUD completo y funcional de la tabla Pacientes

**Actividad 4**

Luego de esto, se procederá a realizar el CRUD completo de las tablas **médico** y **especialidad,** con todas las rutas, vistas, modelos y ajustes en controladores requeridos para dicha implementación

- **Actividades de transferencia del conocimiento**
  Los aprendices deberán demostrar los conocimientos conceptuales y procedimentales que adquirieron a partir de las actividades de apropiación mediante el siguiente instrumento de evaluación:

Los aprendices deberán realizar cada una de las actividades propuestas para adquirir dominio en los conocimientos relacionados con los diferentes temas. Se evaluarán los conocimientos adquiridos mediante instrumentos de evaluación de Desempeño y Producto relacionados en la presente guía

- **Ambiente Requerido**
  - Ambiente de SISTEMAS con conexión eléctrica e internet
- **Materiales**
  - Computadores (30)
  - Sillas (3)
  - Televisor (1)
  - Resma tamaño carta (1)
  - Marcadores (3)
  - Lápiz (1)
  - Lapicero (1)

## 4. ACTIVIDADES DE EVALUACIÓN

| Evidencias de Aprendizaje | Criterios de Evaluación | Técnicas e Instrumentos de Evaluación |
|---|---|---|
| **Evidencias de Conocimiento :**<br><br>**Evidencias de Desempeño**<br><br>**Asistencia y participación activa en las diferentes actividades propuestasz**<br><br>**Evidencias de Producto:** | Crea la base de datos en el motor de base de datos seleccionado, siguiendo especificaciones técnicas del informe, según normas y protocolos de la empresa. | **Observación:** EXC_D_01<br><br>**Valoración del Producto:** EXC_P_01 |

## 5. GLOSARIO DE TÉRMINOS

**Sistema de información:** es un conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su uso posterior, generados para cubrir una necesidad o un objetivo

**Sistema operativo** – Es un conjunto de programas que sirven para manejar un ordenador.

**Software** - El conjunto de programas, procedimientos y documentación asociado a un sistema informático.

**Javascript:** es un lenguaje de programación del lado del cliente que se utiliza con frecuencia en diseño WEB para generar efectos más complejos que no se puedan alcanzar usando HTML.

**HTML:** Siglas de las palabras inglesas: Hypertext Markup Language. Es decir, lenguaje de marcado de hipertexto. Lenguaje informático para crear páginas web. Conjunto de etiquetas o instrucciones que permiten estructurar el contenido de una web e incluir los hipervínculos o enlaces enlaces a otras páginas. Este lenguaje lo inventó en 1991 el Doctor Berners-Lee del CERN en Suiza.

**HTTPS:** Siglas de las palabras inglesas: HyperText Transfer Protocol Secure o versión segura del protocolo HTTP. Es el protocolo empleado para la transferencia de ficheros HTML cifrados que puedan contener información confidencial.

**HTTP:** siglas de las palabras inglesas: Hypertext Transfer Protocol. A saber en español: Protocolo de Transmisión de Hipertexto. Protocolo estándar de transferencia de hipertexto. Es decir: el protocolo de comunicaciones en el que está basado la Word Wide Web.

**Script:** es un archivo de órdenes o archivo de procesamiento por lotes. Es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.

**MySQL:** es un sistema de gestión de bases de datos de código abierto que, junto con PHP, permite darle a las páginas web cierto dinamismo, es decir, disponer de manera adecuada los datos solicitados por los navegadores. Es un sistema multiplataforma y su uso está tan extendido en las bases de datos que podría considerarse un estandar.

**SEO (Search Engine Optimisation) Optimización en buscadores:** técnica utilizada para asegurar que una página Web es compatible con los motores de búsqueda y así tener la posibilidad de aparecer en las posiciones más altas en los resultados de búsqueda.

**Diseño web adaptable (responsive web design):** se llama así al diseño web de aquellas páginas que se adaptan al tamaño de la pantalla o ventana en que se despliegan, por medio del uso de, idealmente, un solo documento HTML y un solo documento CSS. Esto permite hacer una sola página web para smartphones, phablets, tablets y PC.

**Diagrama o Modelo Entidad Relación (DER)**: denominado por sus siglas en inglés, E-R "Entity relationship", o del español DER "Diagrama de Entidad Relación") es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades

**Bases de Datos (BD):** es un banco de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto

## 6. REFERENTES BILBIOGRÁFICOS


- Documentos técnicos relacionados en la plataforma

| | |
|---|---|
| https://www.youtube.com/watch?v=6ERdu4k62wI<br><br>https://www.freecodecamp.org/news/create-an-mvc-framework-from-scratch-with-php/ | Use PHP to Create an MVC Framework - Full Course |

| | |
|---|---|
| https://codeburst.io/mvc-design-pattern-analogy-to-an-old-school-landline-3dcd2e994063 | MVC Design pattern: analogy to an old school landline |
| https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/ | The Model View Controller Pattern – MVC Architecture and Frameworks Explained |

**7. CONTROL DEL DOCUMENTO**

| | **Nombre** | **Cargo** | **Dependencia** | **Fecha** |
|---|---|---|---|---|
| **Autor (es)** | **Julian Salazar Pineda** | **Instructor** | **CPIC** | **14-11-2022** |

**8. CONTROL DE CAMBIOS** (diligenciar únicamente si realiza ajustes a la guía)

| | **Nombre** | **Cargo** | **Dependencia** | **Fecha** | **Razón del Cambio** |
|---|---|---|---|---|---|
| **Autor (es)** | | | | | |