

Guía práctica de ejercicios

Introducción a PHP

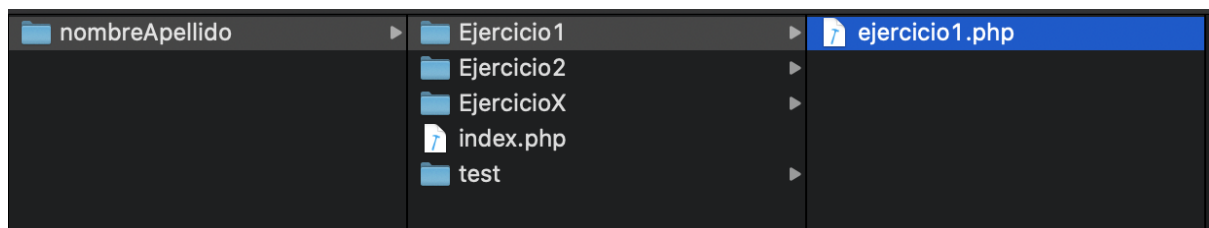
Pasos preliminares:

- Configure un servidor web que permita interpretar PHP
- Configure una IDE que permita desarrollar fácilmente (PHPStorm)

Pautas de resolución:

Todos los ejercicios constan de un enunciado, el cual se debe resolver del siguiente modo:

- La resolución del ejercicio debe estar en una función o clase, que se llamará desde el `index.php`. Dicha función o clase debe estar en otro archivo e incluirse en el `index.php` mediante un `require_once()`



Ejercicio 1: Semáforo

Cree una función llamada Semaforo, que recibe por parametro un color como texto ("rojo", "amarillo", "verde"). Dicha función devolverá el estado que corresponde: "freno", "precaución", "avance" o "estado desconocido" ante un caso no esperado.

- a) función semaforo_a(\$color): Resuelva la solución utilizando if else
- b) función semaforo_b(\$color): Resuelva la solución utilizando if inline (return ? :)
- c) función semaforo_c(\$color): Resuelva la solución utilizando switch

Ejercicio 2: Binomio cuadrado perfecto

Cree una función llamada binomioCuadradoPerfecto que realice la ecuación de dicha problemática: recibe dos parámetros y devuelve el cuadrado de la suma de ambos $(a+b)^2$

- a) función binomioCuadradoPerfecto_a(\$a, \$b): Resuelva la solución utilizando la función de potencia
- b) función binomioCuadradoPerfecto_b(\$a, \$b): Resuelva la solución utilizando la fórmula desarrollada del binomio: $a^2 + 2*a*b + b^2$

Ejercicio 3: Concatenar textos

Cree una función concatenar(\$texto1, \$texto2) que reciba dos textos como parámetro y devuelva ambos textos concatenados como uno solo.

Ejercicio 4: Incrementar

Cree una función llamada incrementar, que reciba una variable y sin devolver nada como retorno de la función, el valor del parámetro haya sido incrementado en 1
(Ver pasaje de parámetros por referencia)

Ejercicio 5: Sumatoria

Cree una función sumatoria que reciba un vector como parámetro, y devuelva la suma de todos sus valores.

- a) función sumatoria_a(\$array): Resuelva la solución utilizando la estructura de control for
- b) función sumatoria_b(\$array): Resuelva la solución utilizando la estructura for each
- c) función sumatoria_c(\$array): Resuelva la solución utilizando la estructura de control while

Ejercicio 6: Nombre completo

Cree una clase llamada Saludar, la misma tendrá un constructor que reciba nombre, apellido de una persona.

Dicha clase debe implementar el método saludoFormal(\$horario) el cual debe responder concatenado al nombre un prefijo dependiendo del horario:

05hs a 13hs “Buenos días”

13hs a 21hs “Buenas tardes”

21hs a 05hs “Buenas noches”

Ej. para clase instanciada para Ezequiel Perez, y parámetro 9hs: “Buenos días Ezequiel Perez”

Dicha clase debe implementar también el método saludoInformal(\$horario) el cual debe responder sin el apellido, iniciando con un “hola” por delante y al finalizar concatenar “que tengas un ...” saludo perteneciente al horario .

Ej. para clase instanciada para Ezequiel Perez, y parámetro 9hs:

“¡Hola Ezequiel! Que tengas un buen día”

Ejercicio 7: Integrando PHP, HTML y CSS

Cree una web a modo de demostrar las funciones resueltas en el trabajo práctico.

Cree un archivo index.php, que contenga cómo título su nombre, y apellido.

A continuación liste los ejercicios a resolver, y en cada caso comente con qué valores se llama a la función, la llame efectivamente y muestre por pantalla el resultado utilizando la función “echo” .

Cómo resultado se debería visualizar algo similar al ejemplo al ingresar a localhost:

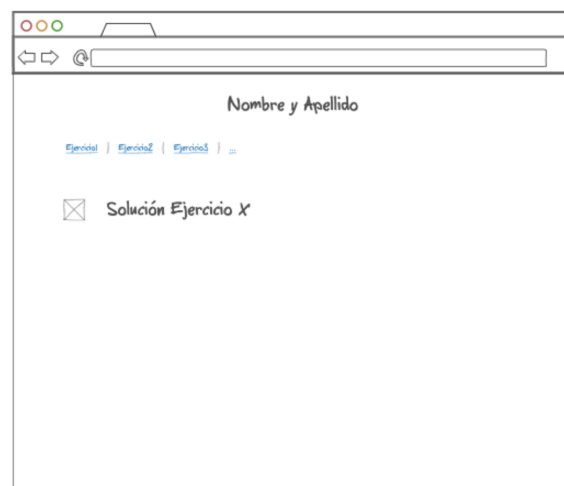
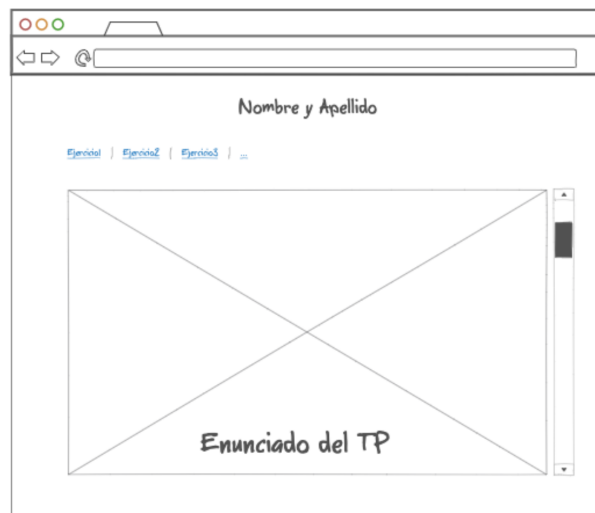


Algoritmos y funciones del lenguaje

Ejercicio 8: Punto de partida

Realizar una web home en el archivo index.php, que contenga:

- Su nombre y apellido como título
- Un menú horizontal para acceder a los ejercicios por su nombre
- Un iframe con el archivo PDF que contiene el enunciado
- La pantalla de cada ejercicio debe ser igual a ésta, sólo que mostrar en el body la resolución.
(Utilizar includes para el header y footer)



Ejercicio 9: Insta-gramo

Realizar una web que muestre todas las imágenes que contiene en la carpeta “/imagenes” con su respectivo nombre de archivo como pié de imagen. Al final de dicha web debe haber un formulario que permita subir una imagen con un nombre a designar.

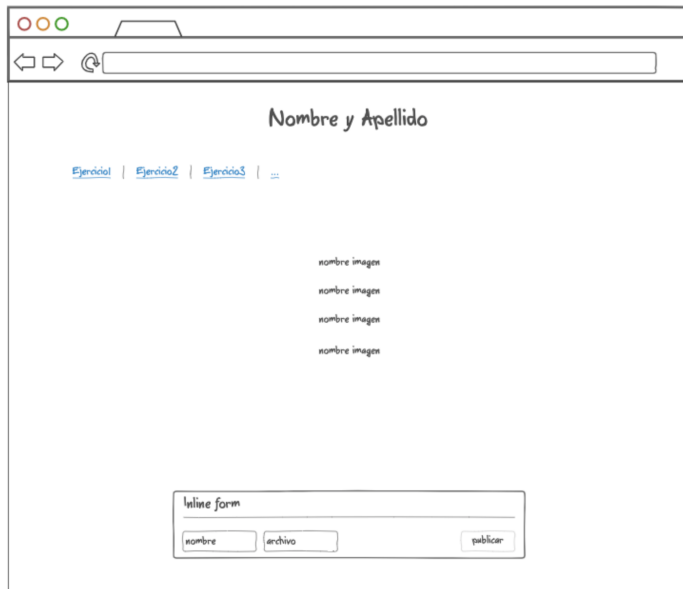
Tip: Luego de cargar la imagen, debe volver a la misma página, sin tener que tocar links “volver” o cosas similares (Como en todo insta-gramo!)

Tip: Para facilitar la resolución, tener 3 imágenes cargadas en la carpeta



Ejercicio 10: Insta-reciclado (para cuidar el medio ambiente):

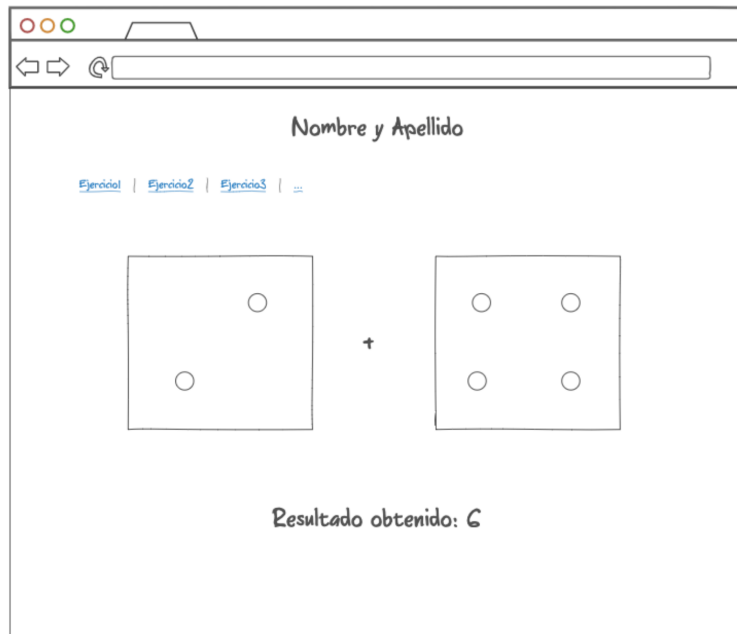
Reutilizando el ejercicio anterior, realizar una web que liste todos los nombres de imágenes que contiene en la carpeta /imagenes cómo link, que al hacer clic, lleve a una segunda pantalla donde efectivamente se muestre dicha imagen.



Ejercicio 11: Lanzar dados

Realizar una web que permita indicar la cantidad de dados a lanzar (mediante input de tipo option) y al tocar un botón “lanzar dados” pase a una segunda pantalla donde muestre los dados lanzados como imagen y la suma de sus valores como puntaje obtenido.

Tip: Buscar una función en php que devuelve un número random para resolver el ejercicio.



Ejercicio 12: Contador de visitas... extraterrestres

Crear una web que permita ingresar en un formulario, los datos de ingreso a nuestro planeta: nombre del visitante y planeta perteneciente (mediante un combo).

Una vez enviado el formulario, muestra una segunda página que liste todos los visitantes que cruzaron este control interplanetario, y un contador del total de visitas, que no pertenecen al planeta Tierra.

Tip: Almacenar en un archivo (en el servidor) cada una de las visitas, para dicho almacenado, pueden utilizar formato json, archivo separado por comas, cada visita en un archivo o algún otro tipo de su preferencia. No utilizar base de datos

Tip: Es recomendable, realizar el cálculo de visitas, cuando leen el archivo, justo antes de mostrarlo y no persistir esta sumatoria

Ejercicio 13: ConociendINIls con el Menú no saludable

Mostrar en pantalla un formulario con checkbox que permita elegir más de un paso en el menú de hoy: Entrada, plato principal, acompañamiento y postre.

Una vez enviado el formulario, lea el archivo menu.ini, y muestre en pantalla sólo los pasos del menú que solicitó el comensal:

```
menu.ini
entrada=Snacks
plato_principal=Milanesa
acompañamiento=Papas fritas
postre=helado
```

Tip: Utilizar función `parse_ini_file` que permite leer archivos .ini de modo sencillo

Tip: Si la letra ñ es un problema, no la utilice

Tip: Puede cambiar los menú de comida por lo que prefiera, la intención es aprender a leer un archivo .ini y utilizar el array asociativo resultante

Ejercicio 14: La Matrix... digo, Matriz

Solicite mediante un formulario, la dimensión de una matriz cuadrada.

En la siguiente página, cree esa matriz de NxN y realice las siguientes acciones:

- Recorrer la matriz con un sólo FOR y mostrar en pantalla los valores que componen la diagonal principal (1,12,23,etc)
- Recorrer la matriz con un sólo FOR y mostrar en pantalla los valores que componen la diagonal secundaria (10,19,28,etc)
- Recorra la matriz (Ahora si con 2 for) y sume todos los valores que contiene. Muestre el resultado

Tip: En los pasos a y b, no utilice un while, ni un flag, sólo con el contador del for debe alcanzar. Analice las posiciones de la matriz que componen las diagonales ;)

Ejercicio 15: Buscando una palabra en un pajar, más bien en un String:

Crear una función que reciba una palabra a buscar y un texto, y devuelva cuantas ocurrencias hubo de la misma. Recorrer el texto String caracter a caracter sin utilizar funciones de String, la firma de la función será similar a: `buscar(clave, texto)`

Paradigma orientado a objetos

Los siguientes tres ejercicios tienen la intención de conocer el paradigma de objetos. Donde no debemos valernos de estructuras de control if-while-for para modelar la solución a los problemas, si no, de familias de objetos polimórficos que envían mensajes e interactúan entre sí. El mismo enunciado dará los pasos a seguir.

Ejercicio 16: Piedra, papel o tijera:

Crear las clases Piedra, Papel y Tijera. Todas ellas deberán tener los siguientes métodos

- constructor(nombreJugador)
- competirContra(otro)
- teCompiteTijera(otro)
- teCompitePiedra(otro),
- teCompitePapel(otro),
- ganador() -> devuelve el nombre del jugador "o la palabra Empate, si lo hubo"

Las funciones competirContra() buscan implementar el patrón **double dispatch**, y las otras tres se limitan a responder si ganan ellos, el contrincante o hay Empate (\$this, \$otro o new Empate()). Al finalizar, se le indicará que indique el ganador(). Algunos ejemplos para ayudar:

```
Tijera{
    ...

    funcion competirContra(otro){
        return otro.teCompiteTijera( this )
    }
    ...
}

-----

$jugadaFacu = new Tijera("Facu");
$jugadaAle = new Piedra("Ale");

$resultado = $jugadaFacu.compiteContra($jugadaAle);
echo $resultado.ganador(); // Imprime Ale

-----
```

Ejercicio 17: Calcula Doris - No olvidemos imprimir el resultado final!

Crear los objetos Operando, Sumar, Restar. Todos deben responder al método `resolverEcuacion()` de modo tal de poder escribir:

```
$resultado = new Sumar(
    new Restar( new Operando(5), new Operando(3)) ,
    new Sumar( new Operando(3), new Operando(4))
).resolverEcuacion();
```

y resultado debe ser 9. Crear ahora el objeto Multiplicar, agregarlo a la ecuación y multiplicando todo por 2, ver que sucede ¿Tuvimos que modificar algo de lo existente?

Ejercicio 18: Pacman come tuti

Crear los objetos packman, fantasma , píldora y fantasma comestible.

Packman debe responder al método: `chocaContra(elemento)` y lanzar una excepción si se queda sin vidas. Dicha Excepcion debe llamarse `FinPartida`, y debe indicar la cantidad de puntos obtenidos.

Por defecto packman comienza con 3 vidas y cero puntos.

Si packman choca un fantasma, resta una vida

Si packman choca un fantasma comestible, suma 100 puntos

Si packman choca una píldora, suma 10 puntos

Si packman se queda sin vidas, finaliza la partida

Imprimir en pantalla los puntos obtenidos (Ej: Partida activa - 600 puntos - 5 vidas)

Ejemplo clarificador (se supone):

```
$partida = new Partida()
$packman = new Packman($puntaje);
$pildora = new Pildora();
$fantasma = new Fantasma();
$fantasmaComestible = new FantasmaComestible();
```

```
$packman.chocaContra($fantasmaComestible);
$partida->mostrarPuntaje();
$packman.chocaContra($fantasmaComestible);
$packman.chocaContra($fantasmaComestible);
$partida->mostrarPuntaje();
```

```
$packman.chocaContra($pildora);
$partida->mostrarPuntaje();
$packman.chocaContra($pildora);
$packman.chocaContra($pildora);
$partida->mostrarPuntaje();
```

```
$packman.chocaContra($fantasma);
$partida->mostrarPuntaje();
$packman.chocaContra($fantasma);
$packman.chocaContra($fantasma);
```

```
$partida->mostrarPuntaje();
```

Resolución de problemas aplicando TDD

Ejercicio 19: FizzBuzz Kata

Escribe un programa que imprima los números del 1 al 100
Los múltiplos de 3 mostrarán "Fizz" en vez de el número
Los múltiplos de 5 mostrarán "Buzz" en vez de el número
Los múltiplos de 3 y 5 mostrarán "FizzBuzz" en vez de el número

<https://kata-log.rocks/fizz-buzz-kata>

Ejercicio 20: Reglas de bolos (Bowling)

El juego consta de 10 Frames (cuadros). En cada cuadro, el jugador tiene dos tiradas para derribar 10 bolos. La puntuación del marco es el número total de bolos derribados, más bonificaciones por strikes y repuestos (chuzas).

Un repuesto es cuando el jugador derriba los 10 bolos en dos tiradas. La bonificación para ese marco es la cantidad de bolos derribados en el siguiente lanzamiento.

Un strike es cuando el jugador derriba los 10 bolos en su primera tirada. El marco se completa entonces con un solo rollo. La bonificación para ese marco es el valor de las próximas dos tiradas.

En el décimo cuadro, un jugador que lanza un repuesto o un strike puede tirar las bolas adicionales para completar el cuadro. Sin embargo, no se pueden rodar más de tres bolas en el décimo cuadro.

Requisitos

Escribe un juego de clase que tenga dos métodos.

`void roll(int)`

se llama cada vez que el jugador hace rodar una bola. El argumento es el número de bolos derribados.

`int score()`

devuelve la puntuación total de ese juego.

<https://kata-log.rocks/bowling-game-kata>

Trabajos prácticos

TP N°1: Repaso Programación Web 2

Ejercicio 1:

Realizar un análisis sobre la Plataforma MleL, su contenido estético, su funcionalidad y la tecnología involucrada (donde la pueda determinar).

- Detalle el análisis realizado
- A partir de ello proponga al menos una mejora en cualquiera de los tres aspectos mencionados.
- La propuesta debe ser escrita (no es necesario maquetar ni programar) y debe establecer las características de la mejora, su objetivo, y una descripción de las deficiencias que resolvería.

Ejercicio 2:

Según lo observado por Usted en el Ejercicio 1 se le solicita construir una maqueta en HTML y CSS, poniendo en juego sus propuestas, de la Home Page de MleL y de la primera página interna.

TP N°2: Pokédex (ABML con login por sesión y persistencia a BD)

Realizar una web que represente una Pokédex (listado de pokémon) y permita realizar búsqueda sobre la misma. Además, el sistema debe permitir que el administrador se loguee al mismo y pueda realizar altas, bajas y modificaciones de los elementos. Ejemplo de donde pueden tomar datos: http://es.pokemon.wikia.com/wiki/Lista_de_Pok%C3%A9mon

Tip: Si no les agrada o no conocen pokémon, pueden reemplazar el contenido por el de otro tema que los motive, siempre que se mantengan todas las funcionalidades (Ej: Música, Jugadores de Fútbol, auto fantástico, Meteoro, etc.)

Casos de uso:

- Si se busca un pokemon existente, mostrar sólo ese.
- Si se busca un pokemon inexistente, informar "pokemon no encontrado" y - mostrar la lista de todos ellos.
- Si no se busca nada, mostrar la lista de todos ellos
- Al ingresar al pokemon buscado, mostrar todos sus datos en una página completa
- El sistema debe permitir loguear usuarios a modo de administradores
- El tipo de pokemon (fuego, agua, hierba) debe mostrarse con una imagen. No como texto

- El usuario logueado: puede dar de alta un pokemon
- El usuario logueado: puede dar de baja un pokemon
- El usuario logueado: puede editar un pokemon

Guía de maquetado a modo de ejemplo:

Página principal de Búsqueda:

Logo Pokedex

Usuario Password Ingresar

Ingrese el nombre, tipo o número de pokémon Quien es este pokémon?

imagen	tipo	número	nombre
		1	Charmander
		2	Charmeleon

Página principal de Búsqueda para administrador logueado:

Logo Pokedex Usuario ADMIN

Ingrese el nombre, tipo o número de pokémon Quien es este pokémon?

imagen	tipo	número	nombre	acciones
		1	Charmander	Modificación Baja
		2	Charmeleon	Modificación Baja

Nuevo pokémon

Página de visualización del pokemon (interna):



Requisitos técnicos:

- La web debe ser visualizada correctamente desde un celular o computadora (ser responsive). Utilice w3css, bootstraps, materialize o similar
- Los datos mínimos de cada pokémon serán:
 - id de base de datos autoincremental (no puede modificarse)
 - Número identificador único (no autoincremental, se debe cargar a mano y se puede editar por el administrador)
 - Imagen (subir imagen al servidor y guardar en la base de datos la ruta a la carpeta y nombre del archivo)
 - Nombre
 - Tipo (seleccione al menos 4 valores posibles)
 - Descripción
 - agregar todos los datos extras que deseen
- Debe trabajarse sobre una base de datos MySQL (MariaDb o postgre si prefieren).
- El backend debe realizarse en PHP
- Toda la lógica debe residir del lado del servidor en PHP
- No utilice Ajax ni funciones complejas de js
- Entregar un archivo comprimido .zip que contenga:
 - código del sitio
 - sql para crear la base del sitio y sus datos
 - debe poder corregirse descomprimiendo el zip sobre el raíz del apache y accediendo a localhost<puerto>:// chequee en la computadora de algún compañero que esto funcione