

PROGRAMACION WEB

Bienvenidos!!!

Julio Monetti
julio.monetti@itu.uncu.edu.ar

Horario

Comisión A.

Presencial: lunes 16:15 a 18:30

Virtual: Martes 17:50 a 19:10

Comisión B.

Presencial: lunes 18:30 a 20:30

Virtual: Martes 17:50 a 19:10

Aula Virtual: <https://aulas.itu.uncu.edu.ar/itu/login/index.php>

Evolución de la Programación Web

Web Estática (1990 – mediados 90s)

Tecnologías principales: HTML, CSS básico.

Sitios simples, informativos, sin interacción dinámica.

Todo era **estático**, cada página un archivo HTML.

Web Dinámica y CGI (1995 – 2000)

- **Tecnologías:** CGI, PHP, ASP, JSP, Perl.
- Se empiezan a generar páginas desde el **servidor**.
- Surgen bases de datos (MySQL, Oracle, SQL Server).
- Aparece **JavaScript (1995)** para darle vida al frontend.

Evolución de la Programación Web

Web 2.0 y AJAX (2000 – 2010)

Páginas interactivas sin recargar gracias a **AJAX**.

Surgen **frameworks JavaScript**: jQuery, Prototype.

Se popularizan **CMS** (WordPress, Drupal).

Aparece el concepto de **UX** y **redes sociales** (Facebook, YouTube).

Web Moderna con SPA (2010 – 2015)

SPA (Single Page Applications): aplicaciones que cargan una sola vez y luego se actualizan dinámicamente.

Frameworks **AngularJS, React, Vue**.

API REST en el backend (Node.js, Spring Boot, Django REST).

CSS moderno (Bootstrap, Sass).

Evolución de la Programación Web

Web en Tiempo Real (2015 – 2020)

Sockets y WebRTC para chat, videojuegos y videollamadas.

Microservicios en el backend.

GraphQL aparece como alternativa a REST.

Despliegues en la nube (AWS, Azure, Firebase).

Web Actual (2020 – hoy)

Jamstack: sitios estáticos potentes + APIs + microservicios.

SSR (Server Side Rendered) / SSG (Generador de sitios estáticos) (Next.js, Nuxt) para mejorar SEO y rendimiento.

PWAs (Progressive Web Apps): webs que funcionan offline y parecen apps móviles.

IA en la web: chatbots, asistentes inteligentes.

Serverless (AWS Lambda, Cloudflare Workers).

Edge Computing para contenido rápido en todo el mundo.

Arquitecturas para contenido dinámico

- CGI
- Java Servlets
- Server-Side Scripting
- JSP

Motivación

- Creación de páginas *on-the-fly* basadas en los requerimientos del cliente y datos estructurados.
- El scripting del lado del cliente no es suficiente. Solamente manipula un documento, no lo crea a partir de contenido estructurado. Solución: Arquitecturas del lado del servidor para la producción de contenido dinámico.

Servlets

Un **Servlet** es un componente de Java que se ejecuta en un **servidor web** o **servidor de aplicaciones**, y que permite manejar **peticiones HTTP** (por ejemplo, cuando un usuario envía un formulario o accede a una página web) y generar **respuestas dinámicas** (como HTML, JSON, etc.).

Forma parte de la **tecnología Java EE (ahora Jakarta EE)** y es una de las formas clásicas de construir aplicaciones web en Java.

Servlets

Cómo funciona un Servlet?

- **Carga y creación de instancia**

El contenedor carga la clase del servlet al arrancar o en la primera petición.

Se crea **una sola instancia** del servlet (Singleton).

- Durante la **inicialización (init)**, Se llama una sola vez. El servlet se configura y se prepara para manejar solicitudes.
- El usuario hace una **solicitud HTTP** desde el navegador (por ejemplo, al ingresar una URL o enviar un formulario).
- El contenedor busca el mapping en web.xml o @web servlet.
- El **servidor web** recibe la solicitud y la pasa al **Servlet** correspondiente. Por cada request, el contenedor llama al método service(HttpServletRequest, HttpServletResponse), y luego el método decide si invocar a doGet(), doPost(), doPut(), etc.

Servlets

Resumiendo, cómo funciona un Servlet?

- El Servlet procesa la solicitud (por ejemplo, consulta una base de datos).
- El Servlet genera una **respuesta**, usualmente en HTML o JSON.
- El servidor envía esa respuesta al navegador del usuario.
- Por último, en la etapa de **destrucción (destroy)**, el servlet realiza tareas de limpieza antes de ser eliminado del contenedor. Se llama una sola vez cuando el contenedor apaga la aplicación o descarga el servlet.

Servlets

Componentes Clave

- HttpServletRequest:** representa la petición del cliente (parámetros, headers, body, sesión, etc.).
- HttpServletResponse:** representa la respuesta que se enviará. (código HTTP, headers, contenido dinámico).
- ServletContext:** almacén global para toda la aplicación.
- ServletConfig:** configuración específica de un servlet.

Servlets

Cómo se define un Servlet?

Un Servlet es simplemente una clase Java que extiende la clase `HttpServlet` y sobrescribe métodos como:

- **`doGet(HttpServletRequest request, HttpServletResponse response)`**
- **`doPost(HttpServletRequest request, HttpServletResponse response)`**

Estos métodos se ejecutan cuando se recibe una solicitud HTTP de tipo GET o POST, respectivamente.

Servlets

Dónde se ejecuta?

Los servlets se ejecutan dentro de un **contenedor de servlets** (como **Apache Tomcat**, **Jetty**, **WildFly**, etc.). Estos contenedores se encargan de manejar el ciclo de vida del servlet, como:

- Su inicialización (init)
- Procesamiento de solicitudes (service)
- Destrucción (destroy)

Servlets

Ventajas de usar Servlets

- Son rápidos y eficientes.
- Se integran bien con el ecosistema Java.
- Se puede usar junto con JSP (JavaServer Pages) o frameworks como Spring.

Ver App MiServletProject
MiServletProject2 !!!

Servlets

1. Descargar de la plataforma MiServletProject.zip

2. Importar en IntelliJ:

- Abrir IntelliJ > "Open" > seleccioná la carpeta del ZIP descomprimido.
- IntelliJ detectará el proyecto Maven y lo configurará.

3. Compilar:

- Usar Maven > Lifecycle > package para generar el .war. (mvn package)

4. Desplegar en Tomcat:

- Copiá el archivo target/MiServletProject.war a webapps/ de la instalación de Tomcat.
- Iniciar Tomcat y acceder a:
<http://localhost:8080/MiServletProject/hola>

Ver App MiServletProject_con_DoPost!!!

Práctica a6. Servlets

1. Tomar y modificar el proyecto MiServletProject.
2. Agregue un servlet con métodos para atender peticiones GET y POST.
3. Agregue una página html con JQuery que invoque la ejecución del servlet.
4. El método doPost debe recibir de la página WEB un nombre y una edad, y generar como respuesta a la página “El usuario [nombre] tiene [edad] años. Es mayor/menor de edad”

Servlets

ServletConfig

Ver App MiServletProject6 !!!

Servlets

¿Qué es ServletContext?

- Es un **objeto global** que representa la **aplicación web completa** dentro del contenedor (Tomcat, Jetty, etc.).
- Se comparte entre **todos los servlets** de la misma aplicación web.
- Permite:
 - Almacenar atributos globales accesibles por todos los servlets.
 - Leer parámetros de contexto definidos en web.xml.
 - Obtener información del servidor, rutas, logs, etc.

Ver MiServletProject7 !!!

Servlets

ServletContext

Flujo

1. Usuario visita /uno → ServletUno lee parámetros de contexto y setea mensajeGlobal.
2. Usuario visita /dos → ServletDos lee mensajeGlobal desde ServletContext.
3. Ambos servlets comparten **atributos y configuración global** de la aplicación web.

Ver App MiServletProject5!!!

Práctica a6_1. Servlets

Crear una aplicación que encadene tres servlets:

1. El primer servlet recibe desde la página html los datos de un usuario para instanciar un objeto Usuario y enviárselo al segundo servlet.
2. El segundo servlet debe guardar el objeto en una base de datos Mysql y enviárselo al tercer servlet.
3. El tercer servlet debe generar una salida para el browser con los datos del usuario.

CGI

Una interface que permite al Web Server lanzar aplicaciones externas y a partir de esto, crear contenido dinámicamente.

El desarrollo del lado del servidor, también conocido como *backend*, ha experimentado una transformación significativa desde los primeros días de la web hasta las arquitecturas modernas basadas en microservicios y la computación en la nube. Este recorrido comienza con una tecnología fundamental: **CGI (Common Gateway Interface)**.

Historia de CGI

- A principio de los 90, CGI fue **especificado por NCSA** (National Center for Supercomputing Applications), que también desarrolló el servidor web **NCSA HTTPd**, uno de los primeros servidores web populares.
- Antes de CGI, las páginas web eran estáticas. CGI permitió que un navegador enviara datos a un servidor, y que este los procesara con un script en Perl, C, Shell u otro lenguaje, y devolviera una respuesta dinámica en HTML en tiempo real.
- Uno de los primeros y más comunes usos de CGI fue para **formularios web**, donde los datos ingresados por el usuario eran procesados por un script CGI.
- Aunque rudimentario, CGI fue el primer paso hacia una web dinámica e interactiva.

CGI no es

- No es un lenguaje de programación.
- No es un protocolo de comunicación.

Es una interface entre el WebServer y las aplicaciones que generan información.

Un programa CGI puede ser implementado en casi cualquier lenguaje de programación.

Evolución temprana: Servlets y PHP

Para superar las limitaciones de CGI, surgieron tecnologías más eficientes:

- **Java Servlets (finales de los 90):** Permitían mantener procesos en memoria y reutilizar instancias, reduciendo la carga del servidor.
- **PHP:** Un lenguaje embebido en HTML, fácil de aprender y muy popular. Permitió el desarrollo ágil de sitios web dinámicos y democratizó la creación de contenido dinámico.

Este período marcó la transición hacia frameworks más estructurados y lenguajes diseñados específicamente para la web.

Frameworks y arquitecturas MVC

Con el crecimiento de las aplicaciones web, surgió la necesidad de organizar mejor el código:

- **Frameworks como Ruby on Rails, Django (Python), Spring (Java), Laravel (PHP)** introdujeron el patrón **MVC (Modelo-Vista-Controlador)**.
- Estos frameworks facilitaron la reutilización de código, el mantenimiento y el trabajo en equipo.

También se incorporaron bases de datos relacionales (MySQL, PostgreSQL) y ORMs (Object-Relational Mappers) que simplificaron la interacción con los datos.

APIs, Node.js, y microservicios

En los últimos años, el backend ha evolucionado hacia modelos más flexibles y escalables:

a. Node.js

Permite ejecutar JavaScript del lado del servidor.

Es asíncrono y basado en eventos, ideal para aplicaciones en tiempo real (chat, videojuegos online, etc.).

b. APIs REST y GraphQL

Separación clara entre frontend y backend.

Permiten que diferentes tipos de clientes (web, móviles, IoT) consuman los mismos servicios.

c. Microservicios y contenedores (Docker, Kubernetes)

En lugar de una única aplicación monolítica, se dividen en pequeños servicios independientes. Esto facilita el escalado, la implementación continua y la resiliencia.

d. Serverless y cloud computing

Servicios como AWS Lambda o Firebase Functions permiten ejecutar código sin gestionar servidores.

Aumentan la eficiencia y reducen costos operativos.

Comparación entre CGI y otras tecnologías backend

Característica	CGI Clásico	PHP	Node.js	Frameworks modernos (Django, Express, etc.)
Modelo de ejecución	Cada petición lanza un proceso nuevo	Código embebido, ejecución en cada solicitud	Un solo proceso asíncrono con eventos	Persistencia y enrutamiento eficiente
Eficiencia	Muy bajo rendimiento	Medio	Alto (no bloqueante)	Alto, depende del diseño
Facilidad de uso	Difícil (requiere bajo nivel)	Fácil (HTML embebido, curva suave)	Media (requiere entender asincronismo)	Alta con frameworks bien documentados
Escalabilidad	Muy limitada	Limitada (bloqueante)	Alta (escalado horizontal simple)	Alta (soporte para microservicios, clustering)
Lenguaje	C, Perl, Shell	PHP	JavaScript	Python, JavaScript, Ruby, etc.
Manejo de estado/sesiones	Manual	Con sesiones de PHP	Librerías como express-session	Integrado en la mayoría de frameworks
Mantenimiento del código	Difícil	Moderado	Moderado	Alto (MVC, separación de responsabilidades)
Reutilización de procesos	No	Parcial	Sí (proceso persistente)	Sí
Comunidad y soporte actual	Obsoleto	Amplia pero en declive	Muy activa	Muy activa (según el framework)

Invocación de CGI

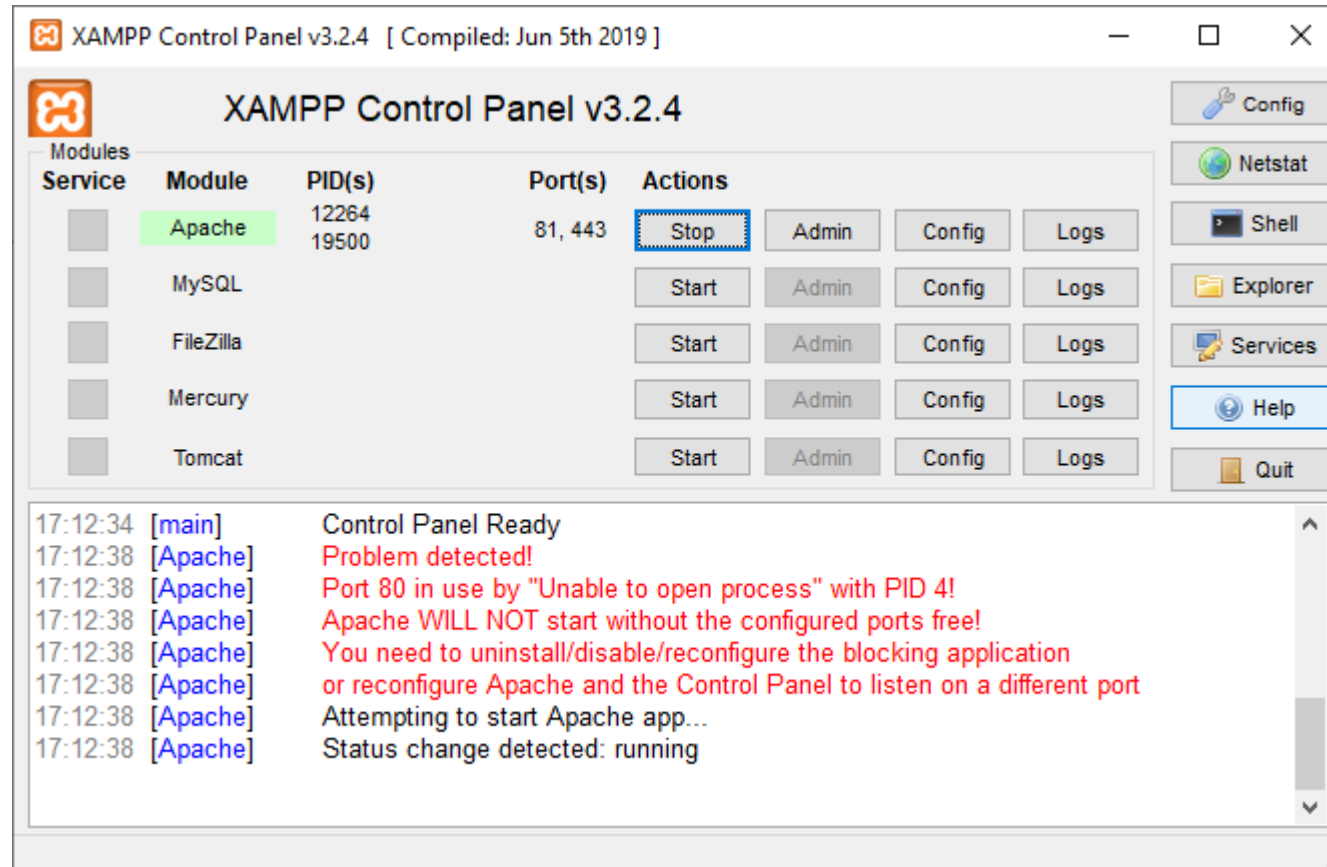
- El cliente especifica en el URI el nombre del programa a invocar.
- El programa debe estar desplegado en un lugar específico del WebServer (por ej: cgi-bin)
 - `http://mi-server.com:8000/cgi-bin/ejecutable.exe`

Ejecución de CGI

- El servidor reconoce desde el URI que el recurso solicitado es un programa ejecutable. Por lo tanto:
 - Se deben configurar permisos de ejecución en el servidor.
- El servidor decodifica los parámetros enviados en la llamada e inicializa las variables del CGI.
 - request_method, query_string, content-length, content-type.
 - <http://mi-server.com/cgi-bin/ejecutable.exe?par=val>
- El servidor lanza el programa ejecutable en un nuevo proceso.
- El programa ejecuta e imprime la respuesta a la salida estándar.
- El servidor construye la respuesta a partir los datos emitidos a la salida estándar y se las envía al cliente.

Caso de uso

Arrancar y configurar el webserver



Caso de uso

Arrancar y configurar el webserver

```
<Directory "C:/xampp/cgi-bin">  
  AllowOverride None  
  Options +ExecCGI  
  Require all granted  
  AddHandler cgi-script .bat .cgi .pl .cmd .py  
</Directory>
```


Caso de uso

En el directorio cgi-bin colocar el fichero de Python con el siguiente código

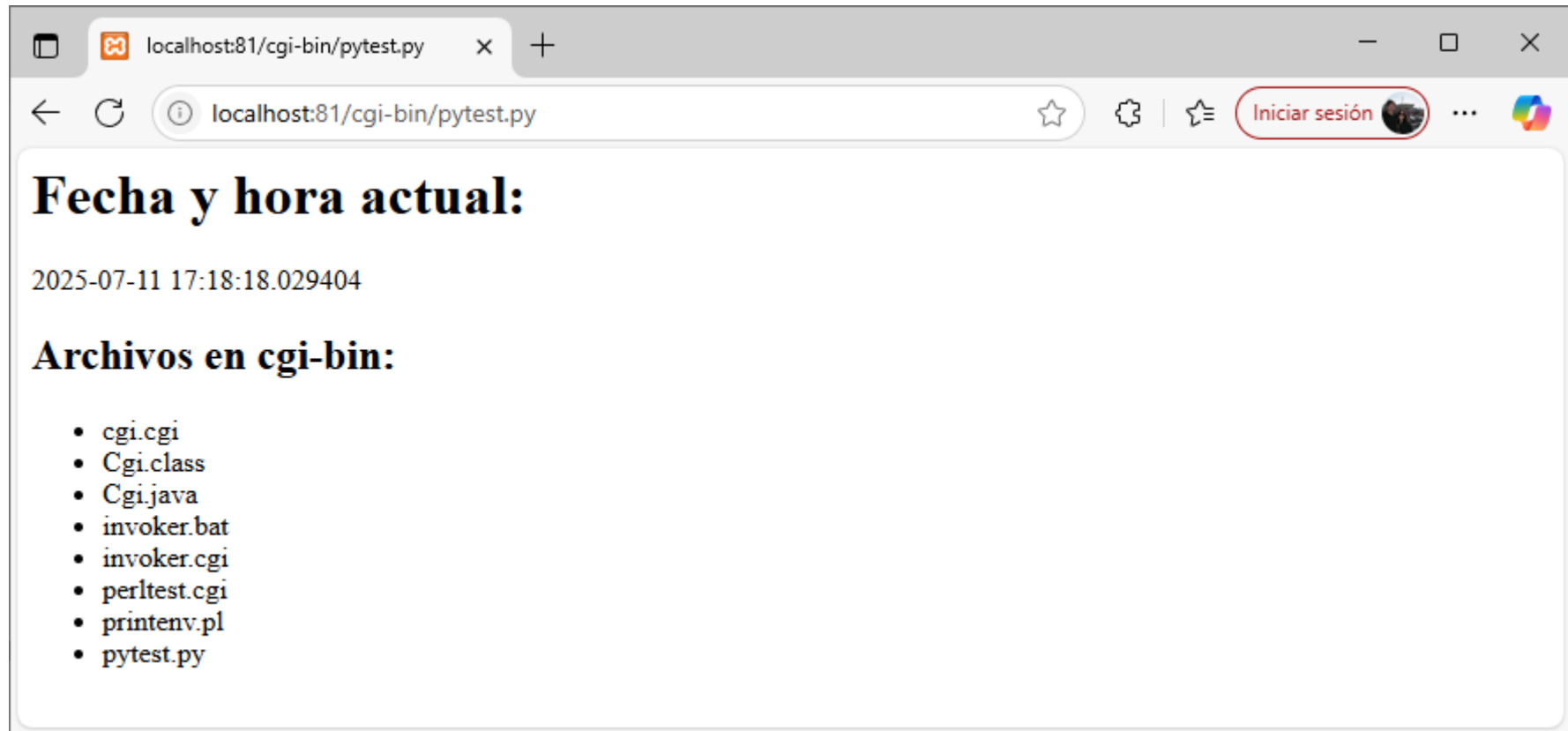
```
#!C:/Python313/python.exe
import datetime
import os

print("Content-Type: text/html")
print()
print("<html><body>")
print(f"<h1>Fecha y hora actual:</h1>")
print(f"<p>{datetime.datetime.now()}</p>")

print("<h2>Archivos en cgi-bin:</h2>")
print("<ul>")
for archivo in os.listdir():
    print(f"<li>{archivo}</li>")
print("</ul>")
print("</body></html>")
```

Caso de uso

Invocar desde el navegador la pagina/fichero pytest.py



Práctica a7. CGI

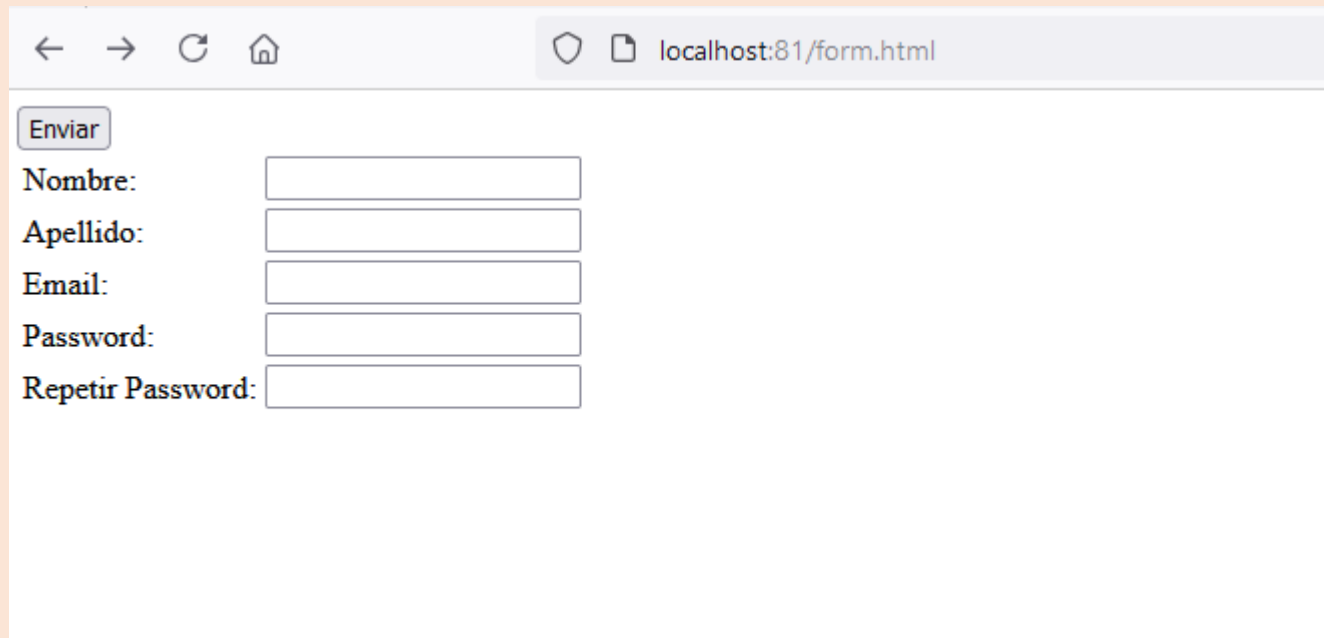
Escribir un programa cgi que obtenga el listado de *nombre* e *email* de una base de datos mysql y los muestre en una página web

Práctica a8. CGI

Ver
htdocs/form.html
cgi-bin/form.py!!!

Procesar los datos de un formulario en un cgi escrito en lenguaje Python. El programa debe:

- Mostrar los datos del formulario.
- Comparar las claves
- Mostrar las variables de ambiente del servidor.



The screenshot shows a web browser window with the address bar displaying "localhost:81/form.html". The page content includes a button labeled "Enviar" and five input fields for user registration:

- Nombre:
- Apellido:
- Email:
- Password:
- Repetir Password:

PHP

Qué es PHP?

- PHP significa "**Hypertext Preprocessor**".
- Fue creado en 1994 por Rasmus Lerdorf.
- Lenguaje de programación del lado del servidor
- El intérprete estándar de PHP, es impulsado por el motor Zend
- Se integra con HTML para crear páginas web dinámicas
- Código PHP se ejecuta en el servidor y genera HTML

PHP

Para qué se usa PHP?

- Procesamiento de datos y formularios WEB
- Sitios web dinámicos y CMS (Sistema de Gestión de Contenidos, como WordPress)
- Conexión con bases de datos (MySQL, PostgreSQL, etc.)
- Sistemas de login, sesiones, autenticación
- Aplicaciones web completas

PHP

Requisitos para programar en PHP

- Servidor web (como Apache)
- Intérprete de PHP
- Editor de código (VS Code, Sublime, etc.)
- Opcional: Base de datos (MySQL)

Ver página htdocs/2 !!!

PHP

Cómo se ejecuta PHP ?

- 1.El navegador solicita una página .php
- 2.El servidor interpreta el código PHP
- 3.PHP genera HTML como salida
- 4.El navegador muestra el resultado al usuario

(Nota: El cliente **nunca ve el código PHP**)

PHP

Ejemplo

`<h2>Formulario de contacto</h2>`

```
<form action="procesar.php" method="post">  
  Nombre: <input type="text" name="nombre"><br><br>  
  Email: <input type="email" name="email"><br><br>  
  <input type="submit" value="Enviar">  
</form>
```

PHP

Ejemplo

```
<?php
// Recibir datos del formulario
$nombre = $_POST['nombre'];
$email = $_POST['email'];

// Validar si se ingresaron datos
if(!empty($nombre) && !empty($email)) {
    echo "Gracias, $nombre. Hemos recibido tu email: $email";
} else {
    echo "Por favor completa todos los campos del formulario.";
}
?>
```

PHP

Ejemplo 2

```
<?php
$numero = 7;

if($numero > 0) {
    echo "$numero es positivo.<br>";
} elseif($numero < 0) {
    echo "$numero es negativo.<br>";
} else {
    echo "El número es cero.<br>";
}

// Loop con for
for($i = 1; $i <= 5; $i++) {
    echo "Número $i<br>";
}

// Loop con foreach
$frutas = ["Manzana", "Banana", "Cereza"];
foreach($frutas as $fruta) {
    echo "Fruta: $fruta<br>";
}
?>
```

PHP

Ejemplo 3. Funciones

```
<?php
function saludar($nombre) {
    return "Hola, $nombre!";
}

echo saludar("Juan") . "<br>";

// Función con suma
function sumar($a, $b) {
    return $a + $b;
}

echo "2 + 3 = " . sumar(2,3);
?>
```

PHP

Ejemplo 3. Arrays Asociativos

```
<?php
$persona = [
    "nombre" => "Juan",
    "edad" => 30,
    "ciudad" => "Buenos Aires"
];

echo "Nombre: " . $persona['nombre'] . "<br>";
echo "Edad: " . $persona['edad'] . "<br>";
echo "Ciudad: " . $persona['ciudad'] . "<br>";

// Recorrer array asociativo
foreach($persona as $clave => $valor) {
    echo "$clave: $valor<br>";
}
?>
```

PHP

Ejemplo 3. Sesiones y Cookies

```
<?php
// Iniciar sesión
session_start();

// Crear variable de sesión
$_SESSION['usuario'] = "Julio";

// Crear cookie que dure 1 día
setcookie("colorFavorito", "Azul", time() + 86400);

echo "Usuario en sesión: " . $_SESSION['usuario'] . "<br>";
echo "Cookie colorFavorito: " . $_COOKIE['colorFavorito'] ?? 'No definida';
?>
```

Ver página htdocs/3 !!!

Práctica a9. PHP

Hacer un programa en PHP que:

- Utilice una bd Mysql para contener los datos de las especialidades tratadas en una clínica.
- Genere un combobox con las especialidades.
- Al seleccionar una de ellas, se muestra la descripción de la misma (este dato se consulta nuevamente en el servidor)

Ver página htdocs/3_1 !!!

Práctica a9_2. PHP

Hacer un programa en PHP que extienda del ejercicio anterior para:

- Registrar nuevas especialidades médicas.
- Listar las especialidades existentes.
- Mostrar la descripción de cada especialidad.
- Guardar datos de usuarios con sesión activa.

Ver página htdocs/4!!!
(búsqueda sensitiva)

PHP. AJAX

```
// Llamada AJAX a PHP
const xhr = new XMLHttpRequest();
xhr.open("GET", "descripcion.php?id=" + id, true); //asincrona=true

xhr.onload = function () {
    if (xhr.status === 200) {
        document.getElementById("descripcion").innerHTML = xhr.responseText;
    }
};

xhr.send();
```

Práctica a10. PHP/Ajax

Hacer un programa en PHP/Javascript que realice la gestión de un carrito de compras.

Tienda

Vaciar carrito

Producto A
\$10
Agregar al carrito

Producto B
\$20
Agregar al carrito

Producto C
\$30
Agregar al carrito

🛒 Carrito (4 items)

Contenido del carrito

- Producto A - 1
- Producto B - 1
- Producto C - 2