

INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	PROGRAMACIÓN WEB 1				
TÍTULO DE LA PRÁCTICA:	Laboratorio 05 :Calculadora con Expresiones Regulares en Perl				
NÚMERO DE PRÁCTICA:	05	AÑO LECTIVO:	2024 - B	NRO. SEMESTRE:	II
FECHA DE PRESENTACIÓN	27/10/2024	HORA DE PRESENTACIÓN			
INTEGRANTE (s):				NOTA:	
DOCENTE: M.Sc. Richart Escobedo Quispe					

INFORME
I. ÍNDICE
II. RESUMEN
<p>Este informe describe el proceso de creación de una calculadora interactiva basada en la web, desarrollada en HTML y que emplea código Perl para llevar a cabo los cálculos. El objetivo de este proyecto fue combinar diversas tecnologías de programación y despliegue dentro de un entorno controlado utilizando Docker.</p> <p>Se diseñó un Dockerfile que permite generar una imagen adecuada para alojar la calculadora en un servidor web, el cual está configurado para utilizar CGI (Common Gateway Interface) para la ejecución de scripts en Perl. Asimismo, se incorporó un archivo de configuración httpd-cgi.config, que garantiza el manejo correcto de las solicitudes del servidor.</p> <p>Además, se implementaron constructores y ejecutores en el Dockerfile, lo que simplifica el proceso de construcción y despliegue de la aplicación web. Este informe ofrece un análisis exhaustivo de cada uno de los componentes, los desafíos encontrados a lo largo del desarrollo y las soluciones adoptadas, así como sugerencias para mejoras futuras.</p>

III. INTRODUCCIÓN

- **Objetivo**
Este informe tiene como objetivo detallar el proceso de desarrollo de una calculadora interactiva en la web, que combina HTML y código Perl. La intención es documentar cada fase del proyecto, desde su concepción inicial hasta la implementación final, con el fin de ofrecer una guía completa sobre las tecnologías empleadas y los retos que se presentaron en el camino.
- **Alcance**
Este informe pretende que sea accesible para cualquier persona, especialmente para la familia Agustina(UNSA), reflejando en el informe aspectos como el diseño visual de la interfaz o la optimización del rendimiento, ya que el enfoque principal se centra en la funcionalidad y el uso de Docker como entorno de desarrollo.
- **Contexto**
Vivimos en tiempos donde la tecnología se apoderó del mundo, viendo una infinidad de lenguajes de programación, enfatizando en el informe el lenguaje de Perl, que nos permitirá la elaboración de una calculadora web, acompañado de un HTML. Sin dejar a lado herramientas de contenedores como Docker permite desarrollar y desplegar aplicaciones de manera más eficiente. Este informe se enmarca en la tendencia actual de utilizar tecnologías modernas para simplificar y mejorar los procesos de desarrollo web.

IV. METODOLOGÍA

Durante el desarrollo del laboratorio se llevó a cabo una diversa cantidad de “pasos” que nos llevó a la culminación con éxito del trabajo planificado:

1. **Planificación y Diseño:** Se inició el proyecto definiendo los requisitos funcionales de la calculadora, estableciendo las operaciones matemáticas que debía soportar. A partir de ahí, se diseñó la interfaz en HTML para asegurar que fuera simple e intuitiva para el usuario.
2. **Desarrollo del Código Perl:** Se escribió el script en Perl encargado de recibir las entradas del usuario desde la interfaz HTML, procesar los cálculos solicitados y devolver los resultados. Se prestó especial atención a la validación de entradas para manejar posibles errores y evitar resultados inesperados.
3. **Configuración del Entorno Docker:** Para facilitar la ejecución y despliegue de la aplicación, se creó un Dockerfile que define el entorno necesario para alojar la calculadora. El Dockerfile incluía la instalación de un servidor web Apache configurado para soportar CGI, permitiendo la ejecución de scripts Perl.
4. **Archivo de Configuración [httpd-cgi.config](#):** Se configuró el archivo [httpd-cgi.config](#) para asegurar que el servidor web permitiera la ejecución de scripts CGI, estableciendo rutas correctas y permisos necesarios para el funcionamiento sin interrupciones de la

calculadora.

5. **Implementación de Constructores y Ejecutores:** Se añadieron pasos en el Dockerfile que actúan como constructores y ejecutores, permitiendo que el entorno se construya y ejecute de manera automatizada. Esta configuración garantiza que la aplicación se pueda desplegar de forma sencilla y repetible, sin necesidad de configurar manualmente el entorno cada vez.
6. **Pruebas y Ajustes:** Se realizaron pruebas exhaustivas para garantizar que la calculadora funcionara correctamente bajo diversas condiciones. Durante esta etapa, se identificaron y solucionaron problemas, optimizando el código y la configuración para mejorar la eficiencia y estabilidad del sistema.

V. DESARROLLO

1. Desarrollo del HTML(index.html)

- a) **Título y Estilos:** Un título claro y estilos CSS que brindan un aspecto limpio y atractivo, con colores suaves y bordes redondeados.
- b) **Formulario de Entrada:** Dos campos de texto para introducir números, un menú desplegable para seleccionar la operación matemática (suma, resta, multiplicación, división), y un botón para enviar los datos al script Perl en el servidor.
- c) **Sección de Resultados:** Un área que muestra el resultado del cálculo de forma destacada y centrada.

2. Implementacion del codigo Perl Código:

```
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4  use CGI qw(:standard);
5
6  print header('text/html');
7  print start_html('Calculadora');
8
9  my $num1 = param('num1');
10 my $num2 = param('num2');
11 my $operacion = param('operacion');
12
13 my $resultado = '';
14
15 if (defined $num1 && defined $num2 && defined $operacion) {
16     if ($operacion eq 'suma') {
17         $resultado = $num1 + $num2;
18     } elsif ($operacion eq 'resta') {
19         $resultado = $num1 - $num2;
20     } elsif ($operacion eq 'multiplicacion') {
21         $resultado = $num1 * $num2;
22     } elsif ($operacion eq 'division') {
23         $resultado = $num2 != 0 ? $num1 / $num2 : 'Error: División por cero';
24     }
25 } else {
26     $resultado = 'Error: Faltan parámetros.';
27 }
28
29 print <<HTML;
30 <h2>Resultado:</h2>
31 <p>$resultado</p>
32 <a href="/">Volver a la calculadora</a>
33 HTML
34
35 print end_html;
```

Explicación:

`use strict;` y `use warnings;` se emplean para mejorar la calidad del código, ayudando a identificar errores y evitar posibles problemas durante la ejecución.

`use CGI qw(:standard);` carga el módulo CGI, que facilita la creación de scripts web interactivos y permiten manejar entradas y salidas de formularios HTML.

De la línea 9 a 11: El script utiliza la función `param` del módulo CGI para recibir los valores enviados desde el formulario HTML. Los parámetros `num1`, `num2`, y `operacion` corresponden a los números ingresados por el usuario y la operación matemática seleccionada.

Línea 13: variable de resultado

```
14
15  if (defined $num1 && defined $num2 && defined $operacion) {
16      if ($operacion eq 'suma') {
17          $resultado = $num1 + $num2;
18      } elsif ($operacion eq 'resta') {
19          $resultado = $num1 - $num2;
20      } elsif ($operacion eq 'multiplicacion') {
21          $resultado = $num1 * $num2;
22      } elsif ($operacion eq 'division') {
23          $resultado = $num2 != 0 ? $num1 / $num2 : 'Error: División por cero';
24      }
25  } else {
26      $resultado = 'Error: Faltan parámetros.';
27  }
28
```

Método: Primero, verifica que los tres parámetros (**num1**, **num2**, **operacion**) estén definidos. Si faltan parámetros, muestra un mensaje de error.

Dependiendo del valor de **operacion**, se realizan diferentes cálculos:

- **Suma:** Se suman los valores de **num1** y **num2**.
- **Resta:** Se resta **num2** de **num1**.
- **Multiplicación:** Se multiplican ambos números.
- **División:** Si **num2** no es cero, se realiza la división. Si **num2** es cero, se muestra un mensaje de error para evitar una operación no válida.

Finalmente genera el resultado HTML.

3. Creación del Dockerfile

```
1 FROM httpd:2.4
2 RUN apt-get update && \
3     apt-get install -y perl libcgi-pm-perl && \
4     cpan CGI && \
5     rm -rf /var/lib/apt/lists/*
6 # Copiar el script y configuración CGI
7 COPY ./calculadora-html/ /usr/local/apache2/htdocs/
8 COPY ./calculadora-scripts/ /usr/local/apache2/cgi-bin/
9 COPY ./calculadora-conf/httpd-cgi.conf /usr/local/apache2/conf/extra/
10
11 # Incluir la configuración CGI en httpd.conf
12 RUN echo "Include conf/extra/httpd-cgi.conf" >> /usr/local/apache2/conf/httpd.conf
13
14 # Dar permisos ejecutables al script Perl
15 RUN chmod -R +x /usr/local/apache2/cgi-bin/
16
17 # Exponer el puerto 80
18 EXPOSE 80
19
20
21
22
```

Instalación de Dependencias: Instala Perl y los módulos necesarios para ejecutar scripts CGI, asegurando que el entorno sea capaz de procesar las operaciones matemáticas definidas en el código Perl.

Configuración del Servidor Apache: Copia los archivos HTML, scripts Perl y configuraciones adicionales al contenedor, permitiendo que Apache sirva la interfaz web y ejecute los scripts CGI. También ajusta la configuración del servidor para incluir soporte CGI.

Permisos y Exposición del Puerto: Otorga permisos de ejecución a los scripts Perl y expone el puerto 80, lo que permite el acceso externo a la aplicación desde un navegador.

4. Configuración del Archivo `httpd-cgi.config`

```
1 LoadModule cgi_module modules/mod_cgi.so
2
3 <Directory "/usr/local/apache2/cgi-bin">
4     AllowOverride None
5     Options +ExecCGI
6     Require all granted
7 </Directory>
8
9 # Habilitar CGI para archivos .perl
10 AddHandler cgi-script .cgi .pl .perl
11
12
```

LÍNEA 1: Esta línea habilita el módulo `mod_cgi`, que permite al servidor Apache ejecutar scripts CGI. Este módulo es esencial para que Apache pueda manejar scripts escritos en lenguajes como Perl, ejecutándolos en respuesta a solicitudes HTTP.

LÍNEA 3-7: Esta sección define cómo Apache debe tratar el directorio `/usr/local/apache2/cgi-bin`, que es donde se almacenan los scripts CGI.

Las directivas son: `AllowOverride None`: Impide que las configuraciones `.htaccess` modifiquen las opciones de este directorio, asegurando que las reglas definidas aquí prevalezcan.

Options `+ExecCGI`: Permite la ejecución de scripts CGI en este directorio, lo que es crucial para que los scripts Perl funcionen correctamente.

`Require all granted`: Da acceso completo a todos los usuarios, permitiendo que cualquier solicitud pueda acceder y ejecutar scripts dentro del directorio.

LÍNEA 10: Configura Apache para tratar archivos con las extensiones `.cgi`, `.pl`, y `.perl` como scripts CGI, habilitando su ejecución.

5. Implementación de Constructores y Ejecutores en el Dockerfile

CONSTRUCTOR:

`docker build -t calculadora-magica .`

```
PS C:\TAREAS\calculadora-main\calculadora-main> docker build -t calculadora-magica .
[+] Building 4.6s (13/13) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 648B
=> [internal] load metadata for docker.io/library/httpd:2.4
=> [auth] library/httpd:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/7] FROM docker.io/library/httpd:2.4@sha256:bbea29057f25d9543e6a96a8e3cc7c7c937206d20eab2323f478fdb2469d536d
=> [internal] load build context
=> => transferring context: 267B
=> CACHED [2/7] RUN apt-get update && apt-get install -y perl libcgi-pm-perl && cpan CGI && rm -rf /var/lib/apt/lists/*
=> CACHED [3/7] COPY ./calculadora-html/ /usr/local/apache2/htdocs/
=> CACHED [4/7] COPY ./calculadora-scripts/ /usr/local/apache2/cgi-bin/
=> CACHED [5/7] COPY ./calculadora-conf/httpd-cgi.conf /usr/local/apache2/conf/extra/
=> CACHED [6/7] RUN echo "Include conf/extra/httpd-cgi.conf" >> /usr/local/apache2/conf/httpd.conf
=> CACHED [7/7] RUN chmod -R +x /usr/local/apache2/cgi-bin/
=> exporting to image
=> => exporting layers
=> => writing image sha256:e6aaa3ed2e2724ce3de5c67f189f4c973efa3bce98f8aa61c7747b8a854366c4
=> => naming to docker.io/library/calculadora-magica
```

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/6hnfz65q8e0ggyo8baxljfy51

What's next:

View a summary of image vulnerabilities and recommendations → `docker scout quickview`

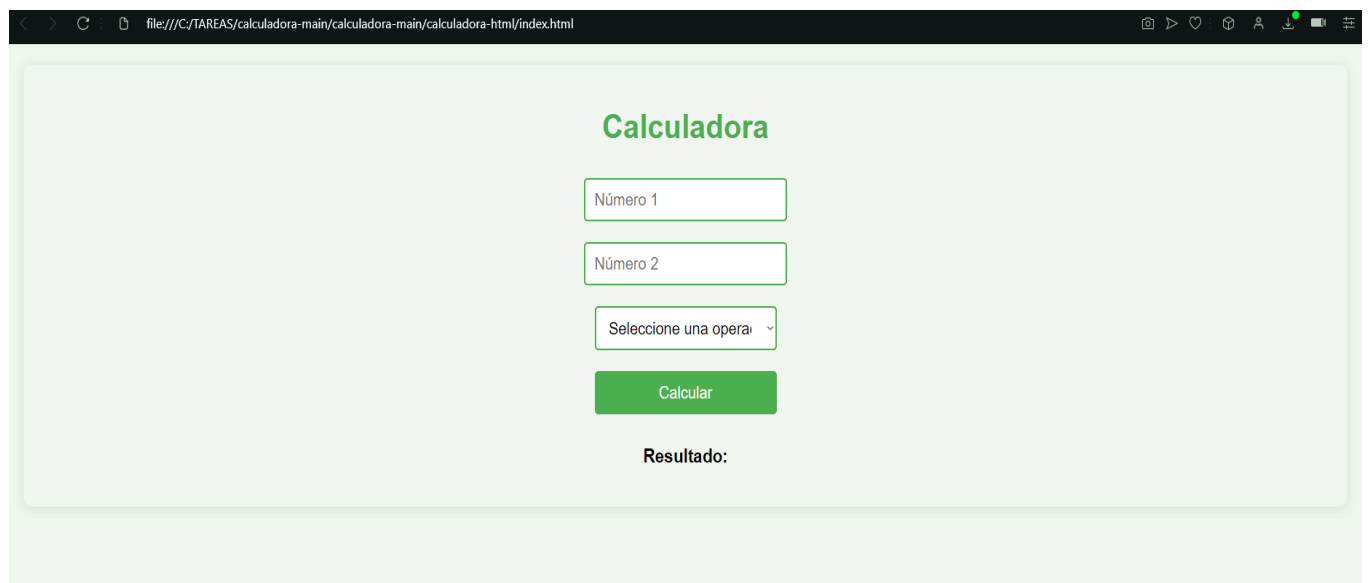
```
PS C:\TAREAS\calculadora-main\calculadora-main>
```

EJECUTOR:

`docker run -dit --name calculadora-magica -p 8090:80 calculadora-magica`

```
PS C:\TAREAS\calculadora-main\calculadora-main> docker run -dit --name calculadora-magica -p 8090:80 calculadora-magica
a682985841b724d4b1242e780c38b00a0d33a1c3eaadeb60dd70e4c4e679247b
```

6. FUNCIONAMIENTO:



file:///C:/TAREAS/calculadora-main/calculadora-main/calculadora-html/index.html

Calculadora

Número 1

Número 2

Seleccione una operación

Calcular

Resultado:

Calculadora

Resultado:

VI. CONCLUSIONES

Integración de Tecnologías: El proyecto ha logrado integrar con éxito diferentes tecnologías, combinando HTML, Perl y Docker para crear una aplicación web funcional. Esta integración permite aprovechar las fortalezas de cada tecnología, ofreciendo una interfaz de usuario intuitiva junto con la lógica de cálculo robusta proporcionada por Perl.

VII. RECOMENDACIONES

Implementar Validaciones de Entrada: Para mejorar la seguridad y la robustez de la aplicación, es recomendable implementar validaciones más estrictas de los datos ingresados por el usuario. Esto podría incluir la verificación de que solo se ingresen números y que se eviten entradas no válidas.