Informe de Laboratorio 06

Tema: Git y GitHub

Nota

Estudiante	Escuela	Asignatura
Auccacusi Conde Brayan Carlos bauccacusic@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web I Semestre: II Código: 20211196
Palma Apaza Santiago Enrique spalmaa@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web I Semestre: II Código: 20240689
Pamo Condori Benjamin Andre bpamoc@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web I Semestre: II Código: 20233480
Huaynacho Mango Jerry Anderson jhuaynacho@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web I Semestre: II Código: 20142322

Laboratorio	Tema	Duración
06	Git y GitHub	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - B	Del 12 Noviembre 2024	Al 17 Noviembre 2024

1. Introducción

Objetivo del Informe

Presentar el desarrollo de una aplicación web que permite realizar consultas dinámicas sobre universidades licenciadas en Perú, utilizando scripts CGI escritos en **Perl**, **HTML**, **CSS** y expresiones regulares. La aplicación fue desplegada y ejecutada en un contenedor **Docker** y se uso **GitHub** para trabajar de manera colaborativa.

Importancia del Proyecto

- Utilización de **Git** y **GitHub** para el control de versiones y colaboración en equipo.
- Introducción al uso de contenedores para implementar y ejecutar aplicaciones web de manera eficiente.
- Aplicación de expresiones regulares para el procesamiento automatizado de datos en un archivo CSV.

2. Equipos, Materiales y Temas Utilizados

Equipos

- Computadoras con capacidad para ejecutar Docker.
- Conexión a Internet.

Materiales

- **Archivos de datos proporcionados:**
  - `Data_Universidades_LAB06.ods` (ODS)
  - `Data_Universidades_LAB06.csv` (CSV)
  - `Licenciamiento Institucional - Diccionario_1.pdf` (PDF)
  - `Programas de Universidades - Diccionario.pdf` (PDF)
- **Software y herramientas:**
  - **Docker:** Para crear la imagen y ejecutar el contenedor.
  - **Git:** Control de versiones local.
  - **GitHub:** Repositorio remoto.
  - **Perl:** Para la lógica del servidor CGI.
  - **HTML y CSS:** Para la interfaz del usuario.
  - **Navegador web:** Para pruebas de la aplicación.
  - VIM 9.0.
  - OpenJDK 64-Bits 17.0.7.

Temas trabajados

- Uso de Git y GitHub.
- Desarrollo y ejecución de contenedores con Docker.
- Procesamiento de datos mediante expresiones regulares en Perl.
- Desarrollo de interfaces web interactivas con CGI, HTML y CSS.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JerryAndersonh/Universidades_CGI.git`

4. Actividades con el repositorio GitHub

4.1. Creando e inicializando repositorio GitHub

- Se creó el repositorio GitHub para gestionar el proyecto.
- Se realizaron los siguientes comandos en la computadora para crear el repositorio local y conectarlo al repositorio remoto:

Listing 1: Creando directorio de trabajo

```
$ mkdir -p $HOME/Universidades_CGI/
```

Listing 2: Dirigiéndonos al directorio de trabajo

```
$ cd $HOME/Universidades_CGI/
```

Listing 3: Creando directorio para el proyecto GitHub

```
$ mkdir -p $HOME/Universidades_CGI/proyecto
```

Listing 4: Inicializando directorio para el repositorio GitHub

```
$ cd $HOME/Universidades_CGI/proyecto
$ echo "# Universidades_CGI" >> README.md
$ git init
$ git config --global user.name "Jerry Anderson Huaynacho Mango"
$ git config --global user.email jerryandersonh@gmail.com
$ git add README.md
$ git commit -m "first commit"
$ git branch -M main
$ git remote add origin https://github.com/JerryAndersonh/Universidades_CGI.git
$ git push -u origin main
```

4.2. Commits

- Se creó el archivo **.gitignore** para no considerar los archivos innecesarios, como ***.log** y otros archivos temporales generados por el proyecto.

Listing 5: Creando .gitignore

```
$ vim .gitignore
```

Listing 6: Contenido de .gitignore

```
*.log
*.tmp
```

Listing 7: Commit: Creando .gitignore para archivos temporales

```
$ git add .
$ git commit -m "Creando .gitignore para archivos temporales"
$ git push -u origin main
```

- Para el siguiente commit, se implementó el backend con Perl para procesar las consultas sobre el archivo de universidades licenciadas.
- El archivo **consulta.pl** fue creado para recibir los parámetros de búsqueda desde el frontend y retornar los resultados adecuados.

Listing 8: Creando archivo de backend consulta.pl

```
$ touch consulta.pl
$ vim consulta.pl
```

Listing 9: Código en Perl para consulta de universidades licenciadas

```
#!/usr/bin/perl
use strict;
use warnings;
use CGI;

my $q = CGI->new;
print $q->header();
# Ejemplo de consulta sobre el archivo CSV usando expresiones regulares

my $archivo = 'Data_Universidades_LAB06.csv';
open my $fh, '<', $archivo or die "No se puede abrir el archivo: $!";

my $nombre = $q->param('nombre');
while (<$fh>) {
    if ($_ =~ /$nombre/i) {
        print "$_\n";
    }
}
close $fh;
```

Listing 10: Commit: Implementando backend con Perl para procesar consultas

```
$ git add .
$ git commit -m "Implementando backend con Perl para procesar consultas sobre
    universidades"
$ git push -u origin main
```

4.3. Estructura del proyecto

- El contenido del proyecto que se entregará en este laboratorio es el siguiente:

```
Universidades_CGI/
|--- consulta.pl
|--- index.html
|--- styles.css
|--- Dockerfile
|--- .gitignore
|--- README.md
|--- src
|--- Data_Universidades_LAB06.csv
```

```
|--- Data_Universidades_LAB06.ods
|--- Licenciamiento_Institucional.pdf
|--- Programas_Univeridades.pdf
```

4.4. Despliegue con Docker

- El despliegue de la aplicación se realiza en un contenedor Docker utilizando el archivo **Dockerfile**.
- El archivo **Dockerfile** configura el entorno con Perl y los módulos necesarios para ejecutar el CGI y manejar las consultas.

Listing 11: Dockerfile para el despliegue

```
FROM perl:latest

# Instalando dependencias
RUN cpan install CGI

# Copiando archivos al contenedor
COPY . /var/www/html

# Expone el puerto 80
EXPOSE 80

# Ejecuta el servidor CGI
CMD ["perl", "/var/www/html/consulta.pl"]
```

5. Pregunta: ¿Qué se aprendió del trabajo colaborativo en GitHub con cuatro integrantes en este proyecto?

En este proyecto, trabajamos de manera colaborativa con un equipo de cuatro integrantes utilizando GitHub como herramienta principal para gestionar el desarrollo y la coordinación del proyecto. A lo largo del proceso, aprendimos varias lecciones importantes sobre trabajo en equipo y control de versiones:

- **Importancia del control de versiones:** GitHub nos permitió gestionar los cambios en el código de manera eficiente. Cada miembro del equipo pudo trabajar de forma independiente en su parte del proyecto y, al final, realizar integraciones sin riesgo de sobrescribir el trabajo de los demás. La creación de ramas (branches) y la posterior fusión (merge) nos permitió trabajar simultáneamente en distintas funcionalidades sin conflictos.
- **Trabajo en equipo y comunicación:** El uso de GitHub facilitó la comunicación y la organización del trabajo entre los integrantes del equipo. Las issues y pull requests fueron esenciales para llevar un seguimiento de las tareas y para realizar revisiones de código antes de integrar las modificaciones al proyecto final. Esto mejoró la calidad del código y permitió que todos los miembros estuvieran al tanto de los avances y problemas.
- **Resolución de conflictos:** En ocasiones surgieron conflictos de código durante la fusión de ramas, lo que nos obligó a colaborar directamente para resolverlos. Este proceso nos ayudó a mejorar nuestras habilidades para manejar problemas técnicos en equipo y encontrar soluciones rápidamente.

- **Mejora de la organización:** Al crear una estructura de carpetas clara y un archivo `README.md` bien documentado, pudimos mantener el proyecto organizado y accesible para todos. Esto facilitó el onboarding de nuevos miembros y ayudó a que todos supieran cómo ejecutar y contribuir al proyecto.
- **Gestión de tareas y asignación de responsabilidades:** GitHub también nos permitió asignar tareas específicas mediante issues, lo que mejoró la asignación de responsabilidades y la planificación del trabajo. Esto nos permitió cumplir con los plazos establecidos y dividir el proyecto en tareas manejables para cada miembro.

En resumen, el trabajo colaborativo en GitHub nos permitió coordinar eficientemente nuestras tareas y mejorar la calidad del proyecto mediante una mejor organización, comunicación y manejo de versiones. Aprendimos a trabajar de forma más efectiva como equipo y a resolver problemas de manera conjunta, lo que fue esencial para el éxito del proyecto.

6. Rúbricas de calificaciones

Tabla 1: Estudiante **Auccacusi Conde Brayan Carlos**

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4			
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4			
Total		20		12	

Tabla 2: Estudiante **Palma Apaza Santiago Enrique**

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4			
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4			
Total		20		12	

Tabla 3: Estudiante **Pamo Condori Benjamin Andre**

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4			
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4			
Total		20		12	

Tabla 4: Estudiante **Huaynacho Mango Jerry Anderson**

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4			
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4			
Total		20		12	

7. Referencias

- https://www.w3schools.com/java/default.asp
- https://www.geeksforgeeks.org/insertion-sort/
- https://www.w3schools.com/java/default.asp
- https://www.geeksforgeeks.org/insertion-sort/
- https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/about-issues
- https://www.git-scm.com/book/en/v2/Getting-Started-About-Version-Control
- https://www.atlassian.com/git/tutorials/comparing-workflows
- https://www.educative.io/edpresso/what-are-pull-requests-in-git
- https://www.gitkraken.com/learn/git/tutorials/git-merge-conflict
- https://dev.to/rammyblog/how-to-merge-and-resolve-conflicts-in-git-2dgp
- https://www.turing.com/kb/what-is-git-and-why-is-it-important
- https://stackabuse.com/understanding-and-resolving-git-merge-conflicts/
- https://www.learngitbranching.js.org/
- https://www.geeksforgeeks.org/git-merge-conflict/
- https://opensource.com/article/19/11/why-use-git