DEPARTAMENTO:	**CIENCIAS DE LA COMPUTACIÓN**	**CARRERA:**	**INGENIERÍA DE SOFTWARE**		
ASIGNATURA:	Pruebas de Software	**PERÍODO LECTIVO:**	202550	**NIVEL:**	6to
DOCENTE:	Ing. Luis Castillo, Mgtr.	**NRC:**	23311	**PRÁCTICA N°:**	6
LABORATORIO DONDE SE DESARROLLARÁ LA PRÁCTICA		Laboratorio H-201			
TEMA DE LA PRÁCTICA:	CI/CD usando GitHub Actions				

INTRODUCCIÓN:

La integración continua (CI) es una práctica fundamental del desarrollo de software moderno. Este laboratorio tiene como propósito familiarizar con la automatización de tareas esenciales como la instalación de dependencias, la ejecución de pruebas unitarias y la verificación de calidad del código mediante ESLint, todo ello gestionado a través de GitHub Actions. A través de una aplicación sencilla en Node.js, se experimentará el poder de los flujos automatizados y se comprenderá la importancia de detectar errores temprano en el ciclo de vida del desarrollo.

OBJETIVOS:

- Configurar un flujo de integración continua (CI) en GitHub Actions que se active automáticamente con cada push o pull request a la rama principal del repositorio.
- Implementar pruebas unitarias usando Jest, garantizando que la lógica del sistema funcione correctamente en cada actualización del código.
- Aplicar análisis estático de código con ESLint, reforzando buenas prácticas de programación y detección temprana de errores o inconsistencias.
- Simular un proceso de despliegue automatizado, demostrando cómo se automatizan las etapas previas al paso final de entrega continua (CD), aún sin depender de un proveedor de hosting.

MATERIALES:

REACTIVOS:
No aplica

INSUMOS:

- Una PC con Windows/Linux
- NodeJS
- Cuenta de GitHub
- GitHub
- Acceso a Internet

EQUIPOS:

Windows 10 o superior, Procesador Intel® Core™ i7-6700T o superior, 12GB RAM o superior, 480GB SSD o superior, Intel HD Graphics 530, similar o superior.

MUESTRA:
No aplica

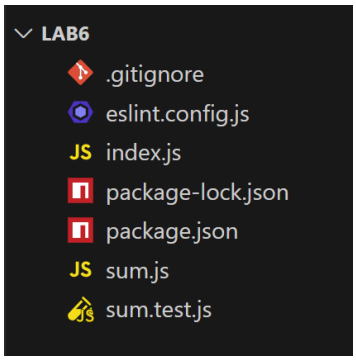
INSTRUCCIONES:

1. Utilizar como material principal de apoyo, aquel indicado en clase por el docente.
2. No olvide incluir capturas de pantallas de todas las actividades realizadas durante la práctica.
3. En los datos ingresados, por favor usar sus datos personales, con el fin de verificar la realización de este trabajo.
4. Se debe comentar el código como mejor práctica de programación.

ACTIVIDADES POR DESARROLLAR:

PARTE 1: Establecimiento de la estructura del proyecto base

Paso 1: Creación de la estructura básica.



Paso 2: Instalación de dependencias necesarias.

a. Creamos el archivo package.json para cargar las dependencias *npm init -y*
b. Instalamos la dependencia de Express *npm install express*
c. Instalamos las dependencias de Jest y ESLint *npm install --save-dev jest eslint* para que se puedan ejecutar en modo desarrollador

PARTE 2: Creación de archivos base

Paso 1: Crear archivo index.js.

a. Usar el servidor express
b. Implementar un endpoint sencillo que responda con un mensaje
c. Levantar el servidor en el puerto 3000

Paso 2: Crear archivo sum.js.

a. Crear una función que sume dos números pasados como parámetros
b. Exportar la función.

Paso 3: Crear archivo sum.test.js.

a. Usar el archivo con la función de suma
b. Crear una prueba para la función de suma.

Paso 4: Configurar package.json.

a. Agregar o editar los scripts para start, test y lint
b. Agregar la característica type para que ESLint funcione como módulo.

Paso 5: Crear el archivo ESLint.

a. Trabajar con reglas sencillas

Paso 6: Ignorar node_modules.

a. En el archivo .gitignore ignorar todos los archivos que puedan causar conflictos para un proyecto NodeJS

PARTE 3: Configuración de Git

Paso 1: Crear repositorio en la cuenta de Git.

a. Abrir la cuenta de Git en el navegador
b. Crear un nuevo repositorio vacío

Paso 2: Ejecución de comandos para clonar al repositorio.

a. git init
b. git add .
c. git commit -m "Proyecto base con CI"
d. git branch -M main
e. git remote add origin https://github.com/TU_USUARIO/nombreRepositorio.git
f. git push -u origin main

Paso 3: Crear el workflow de GitHub Actions.

a. Crear un archivo nuevo para el workflow *.github/workflows/ci.yml*.
b. Configurar los triggers.
c. Configurar los trabajos a realizar
d. Configurar dentro de los trabajos los pasos a ejecutarse.

Paso 4: Probar la CI.

a. Realizar un cambio al código.
b. Ejecutar de nuevo los comandos para realizar un nuevo push.
c. Revisar en GitHub dentro del repositorio, en la pestaña Actions, como se ejecutan los Worflows

SECCIÓN DE PREGUNTAS/ACTIVIDADES

- Agregar más pruebas unitarias
  - Agregar al menos 2 funciones nuevas (por ejemplo, factorial, fibonacci) en un archivo math.js.
  - Crear su correspondiente archivo math.test.js con pruebas Jest.
  - Asegurarse de que GitHub Actions ejecute todas las pruebas con éxito.
- Provocar un error intencional y corregirlo
  - Modificar cualquier función o el test de alguna de ellas para que falle intencionalmente.
  - Subir los cambios y verificar que el flujo CI falla.
  - Corregir el error y volver a subir.

- Adjuntar captura de ambas ejecuciones en GitHub Actions.

RESULTADOS OBTENIDOS:

a. Realizar el informe en el formato general de informes de laboratorio.
b. Evidencia con capturas las actividades prácticas realizadas.
c. Anexar el código fuente en el informe.

CONCLUSIONES:

- Escribir al menos dos conclusiones.

RECOMENDACIONES:

- Escribir al menos dos recomendaciones.

FIRMAS

F:	**F: .....**	**F: .....**
Nombre: Ing. Luis Castillo, Mgtr. **DOCENTE**	**Nombre: Ing. Juan Fernando Galarraga, Mgtr.** **COORDINADOR DE ÁREA DE CONOCIMIENTO**	**Nombre: Crnl (SP) Fidel Castro de la Cruz** **JEFE DE LABORATORIO**