DEPARTAMENTO:	**CIENCIAS DE LA COMPUTACIÓN**	**CARRERA:**	**INGENIERÍA DE SOFTWARE**		
ASIGNATURA:	Pruebas de Software	**PERÍODO LECTIVO:**	202550	**NIVEL:**	6to
DOCENTE:	Ing. Luis Castillo, Mgtr.	**NRC:**	22431	**PRÁCTICA N°:**	5
LABORATORIO DONDE SE DESARROLLARÁ LA PRÁCTICA		Laboratorio H-204			
TEMA DE LA PRÁCTICA:	Pruebas de carga y rendimiento				

INTRODUCCIÓN:

Este laboratorio tiene como objetivo aplicar pruebas de carga y rendimiento a dos niveles de complejidad: primero, a una API REST sencilla que responde con un mensaje básico, y luego a un backend completo que incluye autenticación de usuarios con JWT y acceso a base de datos MongoDB. Para ello, se utilizará la herramienta k6, que permite simular múltiples usuarios concurrentes enviando peticiones de manera controlada. A través de estas pruebas, se podrá observar cómo se comportan distintos tipos de servicios ante escenarios de uso intensivo, y se evaluarán métricas como latencia, tasa de errores y capacidad de respuesta del sistema.

OBJETIVOS:

- Realizar pruebas de carga a una API REST con k6.
- Evaluar el rendimiento del servidor bajo diferentes escenarios de estrés.
- Interpretar métricas como tiempo de respuesta, tasa de errores, y throughput.
- Comparar resultados en diferentes configuraciones.

MATERIALES:

REACTIVOS:
No aplica

INSUMOS:

- Una PC con Windows/Linux
- NodeJS
- MongoDB
- Chocolatey (Windows)
- 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 del ambiente de pruebas

Paso 1: Creación de API sencilla.

a. Crear un archivo server.js
b. Importamos el módulo express para crear el servidor
c. Ruta GET para simular una respuesta simple
d. Simulamos un retardo aleatorio de hasta 500 ms (para pruebas de rendimiento)
e. Ruta POST para recibir datos y responder con lo recibido
f. Iniciamos el servidor escuchando en el puerto especificado

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. Ejecutamos el servidor con *node server.js*

Paso 3: Instalación de k6.

a. Verificar si se tiene instalado Chocolatey en Windows ejecutando el comando *choco* en una terminal
b. Si no se tiene instalado instalarlo como indica su página oficial
c. Abrir una terminal con permisos de Administrador e instalar k6 con el comando *choco install k6*
    - Para Ubuntu usar el comando sudo *apt install k6*
    - Si se usa Docker usar el comando *docker run -i grafana/k6 run - <script.js*

Paso 4: Creación de Script de prueba con k6.

a. Crear el archivo carga-y-rendimiento.js en la raíz del proyecto
b. Importar los módulos necesarios de la librería k6
c. Configurar la prueba que define cómo se comporta la prueba de carga
d. Configurar umbrales de rendimiento
e. Crear una función principal que se ejecutará por cada "usuario virtual" durante la prueba
f. Enviar una solicitud HTTP GET al endpoint de prueba
g. Validar la respuesta usando "check"
h. Esperar 1 segundo antes de que el mismo usuario haga otra solicitud

PARTE 2: Realizar pruebas con k6

Paso 1: Ejecución de las pruebas de carga y rendimiento.

a. Ejecutar la prueba con el comando *k6 run carga-y-rendimiento.js*

Paso 2: Interpretación de métricas.

a. http_req_duration: muestra la latencia (rendimiento).
b. http_req_failed: evidencia errores bajo carga (robustez).
c. http_reqs: número de solicitudes realizadas.
d. vus: virtual users simulados (carga).

Paso 3: Cambio en la configuración del test.

a. Ejecutar la prueba con diferentes parámetros (100, 150, 200, 300).

PARTE 3: Pruebas de carga y rendimiento a un backend completo

Paso 1: Ejecución del backend completo.

a. Ejecutar el backend asegurándose que se ejecute sin problemas
b. Realizar pruebas a los endpoint asegurándose que no haya errores de ejecución

Paso 2: Crear el script para pruebas de carga y rendimiento del backend completo.

a. Crear un archivo nuevo para el script.
b. Configurar la prueba.
c. Generar datos únicos para enviar, se puede usar el número de usuario virtual (VU) e iteración.
d. Crear una función que se ejecuta por cada usuario virtual en cada iteración
e. Verificar que las respuestas sean OK (200)
f. Esperar 1 segundo antes de repetir (simula comportamiento realista)

Paso 4: Ejecución del script y revisión de los resultados arrojados.

a. Ejecutar con el comando *k6 run nombre-archivo.js*.
b. Interpretar resultados arrojados.

SECCIÓN DE PREGUNTAS/ACTIVIDADES

- Probar con POST /api/data enviando JSON con k6.
- Ejecutar pruebas concurrentes GET y POST.
- Crear una versión del script para pruebas de soak testing (larga duración).
- Simular una prueba de spike testing (pico súbito de usuarios).

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**