**Prueba- Implementación de un pipeline de integración continua**

Alumnos:

* César Guerra
* Ernesto Díaz
* Neil Aular
* Omer Solís

1. **Configuración inicial del repositorio (2 puntos)**

* **Inicializa un repositorio Git local y realiza un commit inicial con el archivo db.json y el código proporcionado.**

Se carga la carpeta con los archivos facilitados y ejecuta la acción **git init.** Posteriormente se ejecuta **git add .** Cabe mencionar que aparecen otros archivos, debido a que este informe se comenzó a escribir luego de haber realizado varias pruebas

Imagen que contiene exterior, computadora, tabla, teclado

El contenido generado por IA puede ser incorrecto.

Luego se ejecuta la instrucción **git status** para saber en qué estado están los archivos.

Texto

El contenido generado por IA puede ser incorrecto.

Los archivos están en una zona temporal, previa a dejarlos definitivamente en el repositorio. Por ello, se ejecuta **git commit -m “…”** y de esa forma, quedan subidos al repositorio

Texto

El contenido generado por IA puede ser incorrecto.

* **Crea un repositorio remoto en GitHub y configura el enlace remoto en tu máquina local.**

En el github personal de Omer Solís se creó el repositorio Pruebafinalv1, y su URL es la siguiente: <https://github.com/OmerSolis/Pruebafinalv1.git>

Se ejecutan los pasos para enlazar el repo remoto en forma local.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Y en el repositorio se pueden ver los archivos subidos:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

**2. Configuración básica de la API (2 puntos)**

* **Verifica que la API funciona correctamente utilizando los datos en db.json**

Hubo que modificar el archivo app.js que se entregó en el código para poder para efectuar la lectura del archivo db.json. Se destaca en negrita lo agregado, previo a la creación de los endpoint.

const express = require('express');

**const fs = require('fs');**

**const path = require('path');**

const app = express();

app.use(express.json());

// Leer usuarios desde db.json

**const dbPath = path.join(\_\_dirname, 'db.json');**

**let users = [];**

**try {**

**const data = fs.readFileSync(dbPath, 'utf8');**

**users = JSON.parse(data).users || [];**

**} catch (err) {**

**console.error('Error reading db.json:', err);**

**}**

// Endpoints …

* **Ejecuta la API con json-server o incluye código para servir el archivo como base de datos local**

Se ejecuta la app.js

Texto

El contenido generado por IA puede ser incorrecto.

Y se prueba en localhost.

Se ejecuta el endpoint **/users**, y muestra el contenido del archivo db.json

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Se ejecuta el endpoint **/users/1** y devuelve los datos del user con id = 1.

Interfaz de usuario gráfica, Texto, Aplicación, Word, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Se ejecuta el endpoint **/users/5**, y muestra mensaje de usuario no encontrado.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**3. Automatización básica con Jenkins (3 puntos)**

**Configura un pipeline en Jenkins utilizando un Jenkinsfile que:**

* **Clone el código desde el repositorio remoto.**
* **Ejecute npm install para instalar las dependencias.**
* **Inicie el servidor para validar que la API responde en el puerto configurado.**

El archivo **Jenkinsfile** se debiera ver más o menos así (pensado inicialmente para uso en Linux):

pipeline {

agent any

stages {

stage('Clonar repositorio') {

steps {

git '**https://github.com/OmerSolis/Pruebafinalv1.git**'

}

}

stage('Instalar dependencias') {

steps {

sh 'npm install'

}

}

stage('Iniciar servidor') {

            steps {

                bat 'start /B npm start'

                bat 'ping 127.0.0.1 -n 6 > nul'

                bat 'curl http://localhost:3000/users'

            }

}

}

post {

success {

echo '🚀 API levantada correctamente y responde en /users.'

}

failure {

echo '❌ Algo falló al levantar la API.'

}

}

}

Para probar este código, es necesario levantar Jenkins. La instrucción es la siguiente:

docker run -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts

Y se va generando el siguiente log.

Texto

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Luego, se debiera ejecutar Jenkins desde **http://localhost:8080**

Se solicita una clave, y luego se accede a esta página en la que se selecciona **Install suggested plugins.** Y se comienzan a instalar los plugins.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación, Tabla

El contenido generado por IA puede ser incorrecto.

Luego solicita crear un usuario administrador. Se ingresan los datos y se de clic en botón **Save and Continue.**

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Y se llega a la siguiente página donde se debe indicar desde qué URL se debe ejecutar Jenkins. En este caso, **http://localhost:8080**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Y se llega a la siguiente página:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Se crea el pipeline dando clic en la opción **+ Nueva Tarea**. Se leda un nombre a la tarea y se selecciona la opción **Pipeline.** Luego, clic en botón **OK**.

Interfaz de usuario gráfica, Aplicación, Teams

El contenido generado por IA puede ser incorrecto.

Y se ve más o menos de la siguiente forma, donde en números correlativos (1, 2 y 3) se indican los valores que se deben completar. Cabe mencionar que, por defecto, el pipeline asume que en el repositorio existe un archivo llamado **Jenkinsfile**. Y se da clic en botón **Save**.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Entonces se da clic en la opción **Construir ahora**, y el pipeline se comienza a ejecutar.

Y luego de varias iteraciones, se logra una salida exitosa.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

En la 7ma iteración se logró respuesta exitosa. Cabe mencionar que en cada iteración hubo que realizar algunos cambios en el archivo Jenkins hasta lograr el definitivo.

Este fue el resultado:

Lanzada por el usuario [Omer Solís](http://localhost:8080/user/omer)

Obtained Jenkinsfile from git <https://github.com/OmerSolis/Pruebafinalv1.git>

[Pipeline] Start of Pipeline

[Pipeline] node

Running on [Jenkins](http://localhost:8080/computer/(built-in)/)  in C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Declarative: Checkout SCM)

[Pipeline] checkout

Selected Git installation does not exist. Using Default

The recommended git tool is: NONE

No credentials specified

> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal\.git # timeout=10

Fetching changes from the remote Git repository

> git.exe config remote.origin.url <https://github.com/OmerSolis/Pruebafinalv1.git> # timeout=10

Fetching upstream changes from <https://github.com/OmerSolis/Pruebafinalv1.git>

> git.exe --version # timeout=10

> git --version # 'git version 2.48.1.windows.1'

> git.exe fetch --tags --force --progress -- <https://github.com/OmerSolis/Pruebafinalv1.git> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10

Checking out Revision 0e6dc945ebde4badadbf17c76314fefa0a5e99d6 (refs/remotes/origin/main)

> git.exe config core.sparsecheckout # timeout=10

> git.exe checkout -f 0e6dc945ebde4badadbf17c76314fefa0a5e99d6 # timeout=10

Commit message: "Se reemplaza timeout por ping"

> git.exe rev-list --no-walk f0234b3c2af08ab18bcb9cd634f2b42ec0a2eab6 # timeout=10

[Pipeline] }

[Pipeline] // stage

[Pipeline] withEnv

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Clonar repositorio)

[Pipeline] git

Selected Git installation does not exist. Using Default

The recommended git tool is: NONE

No credentials specified

> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal\.git # timeout=10

Fetching changes from the remote Git repository

> git.exe config remote.origin.url <https://github.com/OmerSolis/Pruebafinalv1.git> # timeout=10

Fetching upstream changes from <https://github.com/OmerSolis/Pruebafinalv1.git>

> git.exe --version # timeout=10

> git --version # 'git version 2.48.1.windows.1'

> git.exe fetch --tags --force --progress -- <https://github.com/OmerSolis/Pruebafinalv1.git> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10

Checking out Revision 0e6dc945ebde4badadbf17c76314fefa0a5e99d6 (refs/remotes/origin/main)

> git.exe config core.sparsecheckout # timeout=10

> git.exe checkout -f 0e6dc945ebde4badadbf17c76314fefa0a5e99d6 # timeout=10

> git.exe branch -a -v --no-abbrev # timeout=10

> git.exe branch -D main # timeout=10

> git.exe checkout -b main 0e6dc945ebde4badadbf17c76314fefa0a5e99d6 # timeout=10

Commit message: "Se reemplaza timeout por ping"

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Instalar dependencias)

[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal>npm install

up to date, audited 356 packages in 2s

48 packages are looking for funding

run `npm fund` for details

found 0 vulnerabilities

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Iniciar servidor)

[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal>start /B npm start

[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal>ping 127.0.0.1 -n 6 1>nul

[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal>curl <http://localhost:3000/users>

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0

100 101 100 101 0 0 6310 0 --:--:-- --:--:-- --:--:-- 6733

[{"id":1,"name":"Alice","email":"alice@example.com"},{"id":2,"name":"Bob","email":"bob@example.com"}]

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Declarative: Post Actions)

[Pipeline] echo

🚀 El pipeline se ejecutó correctamente. La API está funcionando.

[Pipeline] }

[Pipeline] // stage

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // node

[Pipeline] End of Pipeline

Finished: SUCCESS

**4. Ejecución de pruebas automatizadas (2 puntos)**

* **Modifica el pipeline para ejecutar las pruebas preescritas (npm test) automáticamente.**
* **Asegúrate de que el pipeline falle si alguna prueba no pasa**

Antes de cambiar el archivo Jenkins, se ejecutó al test directamente desde la terminal para chequear su funcionamiento, y funcionó correctamente.

Texto

El contenido generado por IA puede ser incorrecto.

En el Jenkins se agrega lo siguiente:

stage('Ejecutar pruebas') {

steps {

bat 'npm test'

}

}

Y la ejecución del Jenkins fue exitosa (iteración 8):

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

y esta es la respuesta. Se destaca en color verde la salida correspondiente al test realizado.

Lanzada por el usuario [Omer Solís](http://localhost:8080/user/omer)

Obtained Jenkinsfile from git <https://github.com/OmerSolis/Pruebafinalv1.git>

[Pipeline] Start of Pipeline

[Pipeline] node

Running on [Jenkins](http://localhost:8080/computer/(built-in)/)  in C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Declarative: Checkout SCM)

[Pipeline] checkout

Selected Git installation does not exist. Using Default

The recommended git tool is: NONE

No credentials specified

> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal\.git # timeout=10

Fetching changes from the remote Git repository

> git.exe config remote.origin.url <https://github.com/OmerSolis/Pruebafinalv1.git> # timeout=10

Fetching upstream changes from <https://github.com/OmerSolis/Pruebafinalv1.git>

> git.exe --version # timeout=10

> git --version # 'git version 2.48.1.windows.1'

> git.exe fetch --tags --force --progress -- <https://github.com/OmerSolis/Pruebafinalv1.git> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10

Checking out Revision 22c474776a46089e3fcc62de0ff2f08d57012e47 (refs/remotes/origin/main)

> git.exe config core.sparsecheckout # timeout=10

> git.exe checkout -f 22c474776a46089e3fcc62de0ff2f08d57012e47 # timeout=10

Commit message: "Se agrega stage para ejecutar pruebas"

> git.exe rev-list --no-walk 0e6dc945ebde4badadbf17c76314fefa0a5e99d6 # timeout=10

[Pipeline] }

[Pipeline] // stage

[Pipeline] withEnv

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Clonar repositorio)

[Pipeline] git

Selected Git installation does not exist. Using Default

The recommended git tool is: NONE

No credentials specified

> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal\.git # timeout=10

Fetching changes from the remote Git repository

> git.exe config remote.origin.url <https://github.com/OmerSolis/Pruebafinalv1.git> # timeout=10

Fetching upstream changes from <https://github.com/OmerSolis/Pruebafinalv1.git>

> git.exe --version # timeout=10

> git --version # 'git version 2.48.1.windows.1'

> git.exe fetch --tags --force --progress -- <https://github.com/OmerSolis/Pruebafinalv1.git> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10

Checking out Revision 22c474776a46089e3fcc62de0ff2f08d57012e47 (refs/remotes/origin/main)

> git.exe config core.sparsecheckout # timeout=10

> git.exe checkout -f 22c474776a46089e3fcc62de0ff2f08d57012e47 # timeout=10

> git.exe branch -a -v --no-abbrev # timeout=10

> git.exe branch -D main # timeout=10

> git.exe checkout -b main 22c474776a46089e3fcc62de0ff2f08d57012e47 # timeout=10

Commit message: "Se agrega stage para ejecutar pruebas"

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Instalar dependencias)

[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal>npm install

up to date, audited 356 packages in 2s

48 packages are looking for funding

run `npm fund` for details

found 0 vulnerabilities

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Iniciar servidor)

[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal>start /B npm start

[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal>ping 127.0.0.1 -n 6 1>nul

[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal>curl <http://localhost:3000/users>

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0

100 101 100 101 0 0 4083 0 --:--:-- --:--:-- --:--:-- 4208

[{"id":1,"name":"Alice","email":"alice@example.com"},{"id":2,"name":"Bob","email":"bob@example.com"}]

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Ejecutar pruebas)

[Pipeline] bat

**C:\ProgramData\Jenkins\.jenkins\workspace\pipeline1\_pruebafinal>npm test**

**> proyecto\_ci@1.0.0 test**

**> jest**

**PASS tests/app.test.js**

**API Tests**

**√ should return a list of users (58 ms)**

**√ should return a single user (7 ms)**

**Test Suites: 1 passed, 1 total**

**Tests: 2 passed, 2 total**

**Snapshots: 0 total**

**Time: 1.438 s**

**Ran all test suites.**

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Declarative: Post Actions)

[Pipeline] echo

🚀 El pipeline se ejecutó correctamente. La API está funcionando.

[Pipeline] }

[Pipeline] // stage

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // node

[Pipeline] End of Pipeline

Finished: SUCCESS

**5. Reporte y retroalimentación (1 punto)**

* **Configura un paso en el pipeline que genere un reporte con los resultados de las pruebas.**

En el archivo Jenkins se agrega lo siguiente, después del stage de ejecución de pruebas:

stage('Generar reporte') {

steps {

// Guarda salida de tests en archivo, incluso si fallan

bat 'npm test > test-report.txt || exit 0'

archiveArtifacts artifacts: 'test-report.txt', allowEmptyArchive: true

}

}

La novena iteración fue exitosa:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Para acceder al reporte, se debe clickear la última ejecución, y se visualiza lo siguiente:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

En la sección “Artefactos generados” está el reporte:

Interfaz de usuario gráfica, Aplicación, Word

El contenido generado por IA puede ser incorrecto.

Sin embargo, el contenido del archivo dice:

> proyecto\_ci@1.0.0 test

> jest

Tal como se muestra a continuación:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Para que se vea el reporte completo de las pruebas, en el Jenkins se debe cambiar la línea:

bat 'npm test > test-report.txt || exit 0'

por

bat 'npm test > test-report.txt 2>&1 || exit 0'

La décima ejecución resulta exitosa:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Se selecciona, y se accede al test-report.txt generado:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Y el contenido ahora sí muestra el resultado de la ejecución:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

* **Incluye un archivo REPORT.md explicando los pasos realizados y los problemas encontrados.**

En el repositorio se dejó el archivo REPORT.MD donde se indican los pasos realizados y los problemas encontrados.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.