Módulo 8
Fundamentos de integración continua

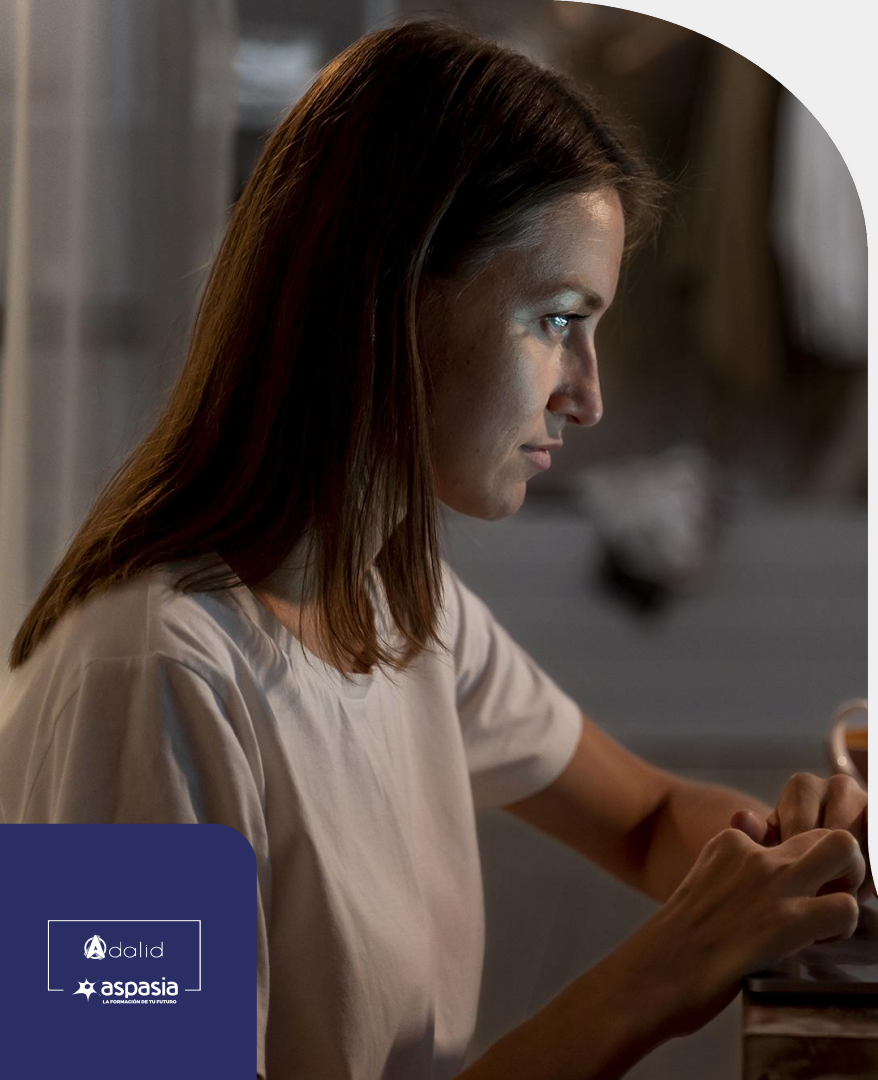
Sistemas de Control de Versiones (SCV)

Módulo 8

AE 3.2

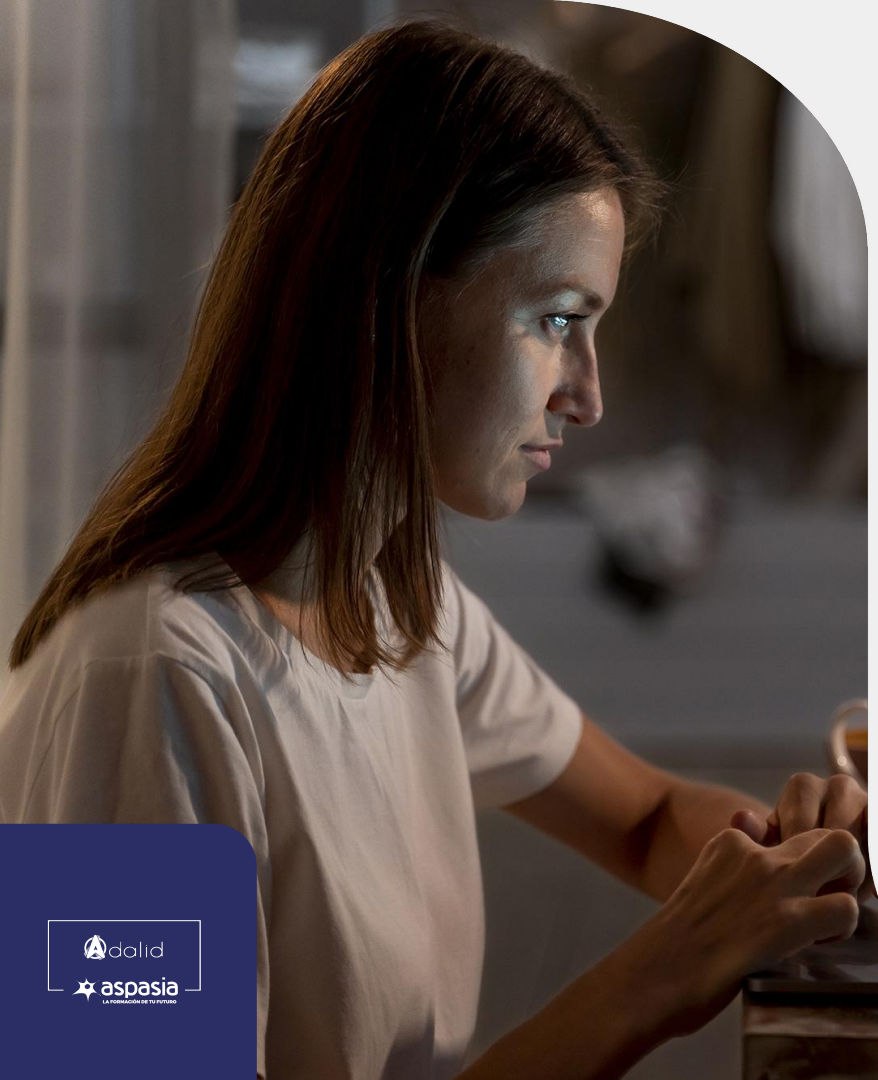
OBJETIVOS

Entender qué es un SCV y cómo resuelve problemas en la gestión del código fuente. Conocer los conceptos clave de Git y su uso en proyectos de desarrollo. Aprender a gestionar repositorios locales y remotos con GitHub, GitLab y Bitbucket. Explorar flujos de trabajo eficientes con ramas, merge, stash y rebase.



¿QUÉ VAMOS A VER?

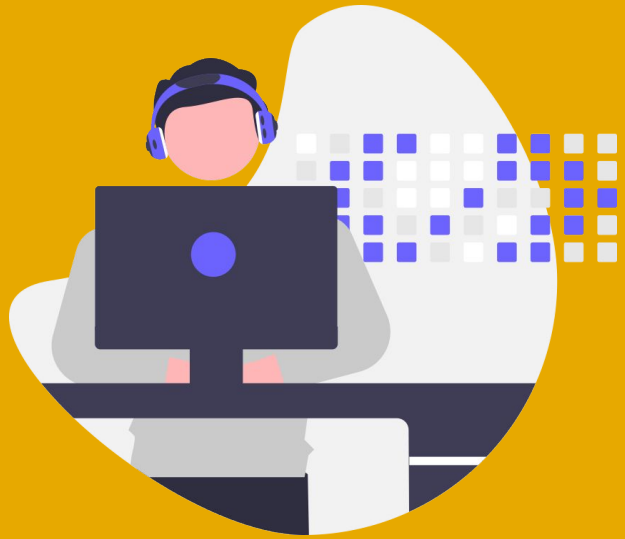
- Sistemas de Control de Versiones (SCV).
- Qué es un SCV.
- Problema que resuelve un SCV.
- Principales conceptos de un SCV. (Repositorio, Diff, Commit, Branch, Merge,
- Clone, Fork)
- Tipos de SCV y alternativas.
- Centralizados (SVN, CVS).
- Distribuidos (Git, Mercurial).
- Git como sistema de control de versiones.
- Instalación, configuración y comandos básicos.



¿QUÉ VAMOS A VER?

- Utilización de Git en repositorios locales
- Commits y restauración de archivos.
- Ignorando archivos.
- Ramas,uniones, conflictos y tags.
- Stash y rebase.
- Centralización de repositorios.
- Servicios de centralización de repositorios (Gitlab, Github, Bitbucket)
- Repositorios remotos, push y pull.
- Fetch vs Pull.
- Clone y Fork de un repositorio.
- Flujos de trabajo.

Pongamos a prueba lo aprendido 😊!!!



Adalid aspasia

Ejercicio Guiado: Creación de un Proyecto con React, Vite, Docker, Git y Testing

Paso 1: Crear el Proyecto con React y Vite

1. Inicializa el Proyecto

```
npm create vite@latest mi-proyecto --template react
```

2. Ingresa a la carpeta del proyecto e instala las dependencias:

```
cd mi-proyecto
npm install
```

3. Verifica que el proyecto funciona correctamente:

```
npm run dev
```

Debe abrirse http://localhost:5173 **con la plantilla de Vite.**

Ejercicio Guiado: Creación de un Proyecto con React, Vite, Docker, Git y Testing

Paso 2: Inicializar un Repositorio Git y Subirlo a GitHub

1. Inicializa Git en el proyecto:

```
git init
```

2. Crea un repositorio en GitHub con el nombre mi-proyecto y agrega el repositorio remoto:

```
git remote add origin
https://github.com/tu-usuario/mi-proyecto.git
```

3. Sube el código al repositorio:

```
git add .
git commit -m "Primer commit - Proyecto con Vite"
git branch -M main
git push -u origin main
```

Ejercicio Guiado: Creación de un Proyecto con React, Vite, Docker, Git y Testing

Paso 3: Crear y Gestionar Ramas en Git

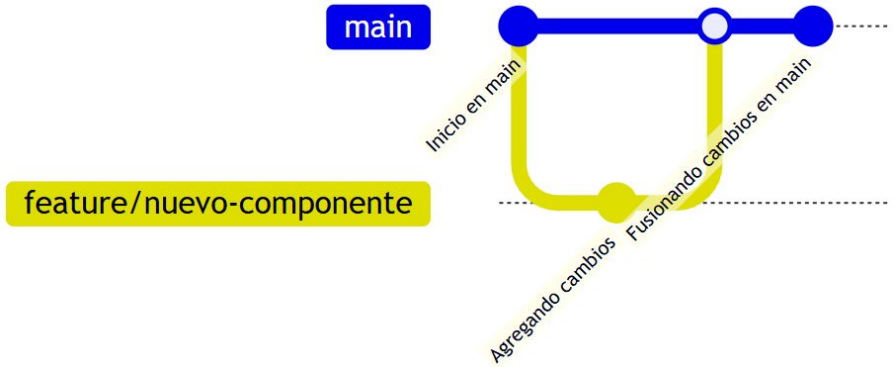
1. Crea una nueva rama para una funcionalidad:

```
git checkout -b feature/nuevo-componente
```

2. Confirma y sube cambios en la nueva rama:

```
git add .
git commit -m "Agregando nuevo componente"
git push origin feature/nuevo-componente
```

Ejemplo de Flujo con Branching:



Ejercicio Guiado: Creación de un Proyecto con React, Vite, Docker, Git y Testing

Paso 4: Crear y Contenerizar la Aplicación con Docker

1. Crea un Dockerfile en **mi-proyecto/Dockerfile**

```
# Usar una imagen base de Node.js
FROM node:16

# Crear y establecer el directorio de trabajo
WORKDIR /app

# Copiar archivos y descargar dependencias
COPY package.json package-lock.json ./
RUN npm install

# Copiar el resto del código
COPY . .

# Exponer el puerto 5173
EXPOSE 5173

# Comando para ejecutar el servidor de desarrollo
CMD ["npm", "run", "dev", "--", "--host"]
```

2. Ignora archivos innecesarios **mi-proyecto/.dockerignore**

```
node_modules
dist
.vite
```

3. Construir y correr el contenedor:

```
docker build -t mi-react-app .
docker run -p 5173:5173 mi-react-app
```

Ejercicio Guiado: Creación de un Proyecto con React, Vite, Docker, Git y Testing

Paso 5: Crear un Nuevo Componente en React

1. Ubicación: **src/components/Saludo.jsx**

```
const Saludo = ({ nombre }) => {
  return <h1>¡Hola, {nombre}! 👋</h1>;
};

export default Saludo;
```

2. Modifica App.jsx para usar el nuevo componente:

```
import Saludo from "./components/Saludo";

function App() {
  return (
    <div>
      <Saludo nombre="Desarrollador" />
    </div>
  );
}

export default App;
```

Ejercicio Guiado: Creación de un Proyecto con React, Vite, Docker, Git y Testing

Paso 6: Implementar Testing en React con Jest y React Testing Library

1. Instala Dependencias para Testing:

```
npm install --save-dev jest @testing-library/react
@testing-library/jest-dom
```

2. Configurar Jest en **package.json**:

```
"scripts": {
  "test": "jest"
}
```

2. Crea una prueba para el componente Saludo **src/components/Saludo.test.jsx**

```
import { render, screen } from "@testing-library/react";
import Saludo from "./Saludo";

test("Muestra el mensaje de saludo con el nombre proporcionado",
() => {
  render(<Saludo nombre="Juan" />);
  expect(screen.getByText("¡Hola, Juan!
👋")).toBeInTheDocument();
});
```

3. Ejecuta las pruebas con el comando:

```
npm run test
```

Ejercicio Guiado: Creación de un Proyecto con React, Vite, Docker, Git y Testing

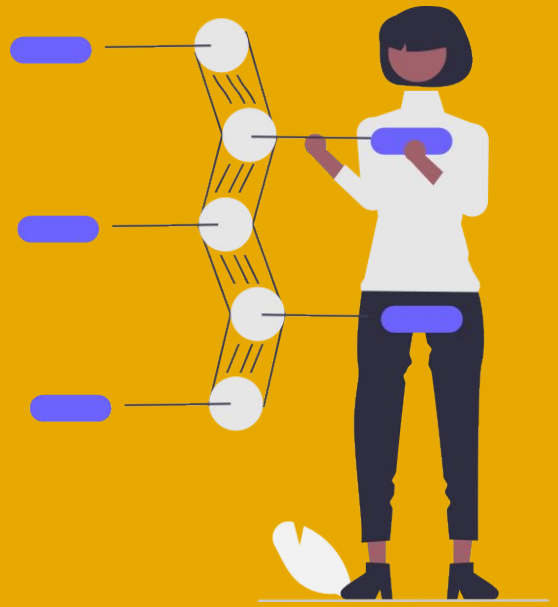
Paso 7: Integrar Testing en un CI/CD con GitHub Actions

1. Crea un archivo para integración continua **.github/workflows/ci.yml.**

Este workflow ejecutará pruebas en cada push y pull request.

```yaml
name: CI/CD Pipeline

on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - name: Clonar el código
        uses: actions/checkout@v2

      - name: Instalar dependencias
        run: npm install

      - name: Ejecutar pruebas unitarias
        run: npm test
```

Resumen de lo aprendido 🧑‍🎓



Adalid aspasia

Resumen de lo aprendido

- Los SCV permiten rastrear cambios en el código, mejorar la colaboración y evitar conflictos.
- Git es un sistema distribuido que facilita la gestión de versiones mediante commits y ramas.
- Plataformas como GitHub centralizan repositorios y optimizan el trabajo en equipo.
- Flujos de trabajo eficientes con Git incluyen merge, rebase, stash y uso estratégico de branches.

GRACIAS POR TU ATENCIÓN

Nos vemos en la próxima clase

Chile
Adalid
aspasia
LA FORMACIÓN DE TU FUTURO