DAM 1 DAW 1 / Entornos de desarrollo

Control de versiones con Git y GitHub

3.3 Uso experto de Git

Autor: Lisa ERIKSEN
Fecha: 2025 / 2026

1. Objetivos de Aprendizaje

En este capítulo aprenderás técnicas avanzadas que te permitirán manejar Git con un nivel experto. Estas habilidades son esenciales para mantener un historial de commits limpio, gestionar ramas de manera eficiente y solucionar problemas complejos en tus proyectos.

Comando	Descripción
git **rebase**	Mantener historial limpio
git **git cherry-pick**	Aplicar commits específicos
git **git rebase -i**	Combinar commits pequeños
git **stash**	Guardar cambios temporales

2. Preparación del proyecto

Actividad: vas a crear un programa en Java para adivinar un numero aleatorio.

1. Crea una carpeta **AdivinaNumero**
2. Crea el archivo **AdivinaNumero.java** con el siguiente código inicial:

```
import java.util.Scanner;
import java.util.Random;

public class AdivinaNumero {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Random rand = new Random();
        int numeroSecreto = rand.nextInt(100) + 1;
        int intento;
        System.out.println("¡Adivina el número entre 1 y 100!");

        do {
            System.out.print("Introduce tu intento: ");
            intento = sc.nextInt();

            if(intento < numeroSecreto) System.out.println("Demasiado bajo");
            else if(intento > numeroSecreto) System.out.println("Demasiado alto");
            else System.out.println("¡Correcto!");
        } while(intento != numeroSecreto);
    }
}
```

3. Inicializa un repositorio Git:

```
Terminal
git init
git add AdivinaNumero.java
git commit -m "Versión inicial del juego"
```

3. Crear una rama para añadir mejoras

Actividad: Vas a crear una rama, para añadir una nueva funcionalidad al juego.

1. Crea una rama **mejoras**:

```
Terminal
git branch mejoras
git switch mejoras
```

2. Modifica el juego para que el jugador reciba **una pista extra**: si el número secreto es par o impar.
   Pista: puedes usar el operador %

3. Guarda tus cambios y haz commit:

```
Terminal
git add .
git commit -m "Añadida pista par/impar"
```

4. Rebase

El comando git rebase reescribe el historial de commits de una rama. Es una alternativa a git merge, pero en lugar de fusionar, aplica los commits de una rama sobre otra como si se hubieran hecho en secuencia.

Actividad: Vas a crear un pequeño cambio en la rama master, y incorpar este cambio en tu rama mejoras.

1. Cambia a la rama **master** y incorpora a los cambios de mejoras con un rebase:

```
Terminal
git rebase mejoras
```

2. Haz un pequeño cambio, por ejemplo, cambia el mensaje de bienvenida:

```
System.out.println("¡Bienvenido al juego de adivinanza!");
```

2. Guarda y haz commit en master:

```
Terminal
git add .
git commit -m "Actualizar mensaje de bienvenida"
```

3. Vuelve a la rama **mejoras** y aplica rebase:

```
Terminal
git checkout mejoras
git rebase master
```

4. Resuelve cualquier conflicto aceptando los cambios entrantes (incoming changes).

👉 Comprueba que los cambios de master se hayan incorporado correctamente en tu rama mejoras y que tu juego funcione como antes.

5. Cherry-pick

El comando git cherry-pick te permite aplicar un commit específico de una rama a otra, sin necesidad de fusionar toda la rama.

5.1 Primer commit: intentos máximos

Actividad:

1. En la rama mejoras, modifica el juego para que el jugador tenga un número máximo de intentos. Si no adivina el número en ese límite, el juego termina mostrando la respuesta. *Pista: usa una variable que se incremente cada vez que el jugador intenta adivinar.*
2. Guarda tus cambios: **git add .**
3. Haz un commit: **git commit -m "intentos maximos"**

5.1 Segundo commit: mensajes personalizados

Actividad:

1. Haz que los mensajes que recibe el jugador cambien según en qué rango se encuentre el número secreto (por ejemplo: "muy bajo", "bajo", "medio", "alto", "muy alto").
2. Guarda tus cambios: **git add .**
3. Haz un commit: **git commit -m "mensajes personalizados"**
4. Revisa el historial de commits para copiar el hash del commit que cambió el mensaje de bienvenida:

Verás algo como:

```
commit 87ed67fc51f7032113af724d75472dd30cc30714 (HEAD -> master)
Author: Alumno
Date:   Thu Nov 25 …

    mensajes personalizados

commit 420a57448022d4abe49ead6f3e9696a6dfbbadf7
Author: Alumno…
Date:   Thu Nov 25 …

    intentos maximos
```

2. Copia el **numero** del primer commit: intentos maximos
3. Aplica ese commit a la rama master con los comandos siguiente:

```
Terminal
git checkout master
git cherry-pick numero-del-commit
```

4. Resuelve los conflictos si aparecen, aceptando los cambios llegando.
5. Comprueba que el cambio se aplicó correctamente en la rama master:

```
Terminal
git log
```

👉 Ahora la rama master tiene el primer cambio, sin fusionar todos los cambios de mejoras.

6. Stash (guardar cambios temporales)

El comando git stash te permite guardar cambios que todavía no quieres commitear, para poder cambiar de rama sin perderlos.

Actividad:

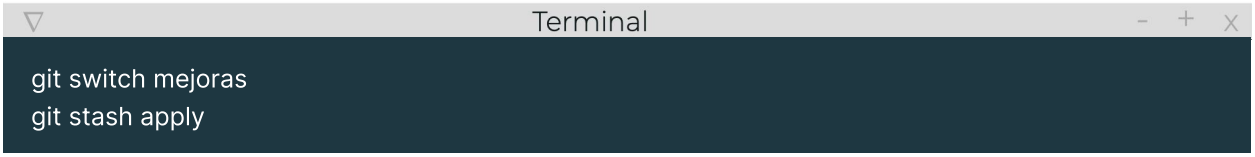
1. En la rama mejoras, **agrega un contador de intentos al juego**.

   Pista: *Cada vez que el jugador introduce un número, aumenta el contador y muéstralo al final cuando adivine correctamente.*

2. Guarda temporalmente tus cambios sin hacer commit:

3. Cambia a la rama master para revisar otra parte del juego:

4. Vuelve a la rama mejoras y recupera tus cambios guardados:



👉 Verifica que tu contador de intentos reaparezca y funcione correctamente.

7. Squash (Agrupar commits)

Objetivo: Combinar varios commits pequeños en uno solo para mantener el historial limpio.

Actividad:

1. Asegúrate de estar en la rama **mejoras** donde quieres agrupar los commits.

2. Revisa los últimos commits:

```
Terminal
git log --oneline
```

Por ejemplo, verás algo como:

```
def456 mensajes personalizados
abc123 intentos maximos
```

3. Decide cuántos commits quieres combinar. Por ejemplo, si quieres agrupar los últimos 2 commits, ejecuta:

```
Terminal
git rebase -i HEAD~2
```

4. Se abrirá un editor con algo así:

```
pick abc123 mensajes personalizados
pick def456 intentos maximos
```

5. Cambia la palabra **pick** en los commits que quieras unir a squash:

```
pick def456 mensajes personalizados
squash abc123 intentos maximos
```

6. Guarda y cierra el editor: **esc** y **:wq**
7. Git pedirá un mensaje final para el commit combinado. *Por ejemplo, escribes: "cambios unidos"*
8. Guarda y cierra el edito: **esc** y **:wq**

👉 Ahora los commits seleccionados están combinados en uno solo. Puedes verificarlo con el comando **git log**