Examen E2 – Full Stack Developer: Debug + Git

En este examen vamos a integrar el control de versiones con depuración y testing en un flujo de trabajo realista.

Este documento debe recoger lo siguiente:

- **Índice**
- **Introducción:** Breve resumen de la factorización realizada
- **Configuración:** Uso y configuración de GitHub
- **Descripción de proceso:** Justificar tanto de forma argumentativa como de forma visual la realización de las distintas partes:
  - **Parte 1: Análisis y Planning**
  - **Parte 2: Depuración y Tests**
    - **Reproducir con Debugger**
    - **Crear Test Unitario**
    - **Commit con mensaje específico**
  - **Parte 3: Corrección Sistematizada**
  - **Parte 4: Integración y Review**
  -
- **Conclusiones:** Realiza un pequeño análisis crítico d

Sistema de Gestión de Biblioteca con Bugs

Se entrega un proyecto de biblioteca con varios bugs intencionados. Los alumnos deben:

1. **Identificar bugs mediante debugging**
2. **Crear tests para reproducirlos**
3. **Corregirlos usando control de versiones apropiado**.

Trabajos a realizar:

Fase 1: Análisis y Planning

1. **Clonar repositorio** del proyecto
2. **Crear rama** `bugfix/library-issues`
3. **Ejecutar aplicación** y observar comportamientos erróneos
4. **Crear issues en GitHub** para cada bug identificado
   - Issue #1: "Libros duplicados permitidos"
   - Issue #2: "Búsqueda sensible a mayúsculas"
   - etc.

Fase 2: Depuración y Tests

Para cada bug encontrado:

Paso A - Reproducir con Debugger:

- Poner breakpoints estratégicos
- Usar Step Into/Over para seguir el flujo
- Inspeccionar variables en puntos clave
- Capturar screenshots como evidencia

Paso B - Crear Test Unitario:

```
// Ejemplo: Test para bug de duplicados
@Test
public void testAddDuplicateBook() {
    Library lib = new Library();
    Book b1 = new Book("Titulo", "Autor", "ISBN-123");
    Book b2 = new Book("Titulo", "Autor", "ISBN-123");

    lib.addBook(b1);
    lib.addBook(b2);

    // ¿Cómo verificar que no se permiten duplicados?
    // Debug aquí para ver el estado interno
}
```

Paso C - Commit con mensaje específico:

```
fix: prevenir libros duplicados por ISBN
- Añadir validación en addBook()
- Crear método findBookByIsbn()
- Añadir tests unitarios
- Close #1
```

Fase 3: Corrección Sistematizada

Estructura de commits REQUERIDA:

1. `test: añadir pruebas para detectar libro duplicado`
2. `fix: implementar validación de ISBN único en addBook()`
3. `test: añadir pruebas para búsqueda case-insensitive`
4. `fix: implementar búsqueda case-insensitive`

5. `refactor: extraer lógica de validación a método separado`
6. `fix: añadir verificación en borrow() y returnBook()`
7. `feat: añadir método removeBook()`
8. `docs: actualizar README con nuevas funcionalidades`

Fase 4: Integración y Review

1. **Merge conflict simulation** (profesor añade cambio concurrente)
2. **Crear Pull Request** con:
    - Resumen de bugs encontrados
    - Técnicas de debugging usadas
    - Screenshots de breakpoints y watch expressions
    - Enlace a issues resueltos
3. **Code Review** cruzado entre compañeros

Criterios de Evaluación

Categoría	Subcategoría	Puntos
Control de Versiones	Commits atómicos y semánticos	1,5
	Uso correcto de branches	1
	Issues y PR bien documentados	1
Depuración	Breakpoints efectivos	1
	Uso de Step Into/Over/Out	1
	Inspección de variables y watch	1
Testing	Tests que reproducen bugs	1
	Tests después de corrección	0,5
Calidad de Código	Soluciones correctas	1
	Comentarios y documentación	0,5
Colaboración	Code Review útil	0,5