# Laboratorio Guiado

# Comprensión de Codebase con GitHub Copilot

**Proyecto:** `spring-projects/spring-petclinic`

https://github.com/spring-projects/spring-petclinic.git

**Objetivo del laboratorio**

Usar GitHub Copilot Chat como **asistente de lectura y análisis**, para:

- Comprender un codebase existente
- Identificar responsabilidades y flujos
- Entender relaciones arquitectónicas
- Detectar supuestos implícitos y ambigüedades

**Escenario**

Te incorporas a un equipo que mantiene *Spring Petclinic*.
No existe documentación funcional actualizada.

**Tu objetivo**

Entender cómo funciona el sistema antes de modificarlo.

**BLOQUE 1 — Comprensión básica de archivos, clases y funciones**

**Objetivo**

Aprender a usar Copilot para **leer código más rápido**, no para confiar ciegamente.

**Ejercicio 1.1 — ¿Qué hace este archivo?**

**Archivo sugerido**

- `PetController.java`

**Prompt básico**

> “Explícame qué hace este archivo y cuál es su propósito principal”

**Actividad**

1. Leer la respuesta de Copilot.
2. Revisar imports, anotaciones y métodos.
3. Marcar qué partes **sí** y **no** coinciden con el código.

**Resultado esperado**

- Comprensión general del rol del archivo.
- Primer contraste Copilot vs realidad.

**Ejercicio 1.2 — ¿Qué hace esta clase?**

**Clase**

- `OwnerController`

**Prompt**

> “Resume la responsabilidad principal de esta clase en una frase”

**Refinamiento**

> “Enumera las operaciones que expone esta clase”

**Aprendizaje clave**

- Copilot tiende a **sobre-simplificar** si el prompt es vago.

**Ejercicio 1.3 — ¿Qué hace este método?**

**Método**

- `processCreationForm(...)`

**Prompt**

> “Describe paso a paso qué hace este método”

**Validación**

- Comparar el flujo descrito vs el código real.

## BLOQUE 2 — Identificación de responsabilidades y flujos principales

**Objetivo**

Pasar de “qué hace” a **“cómo fluye el sistema”**.

### Ejercicio 2.1 — Responsabilidades por capa

**Archivo**

- `VisitController`
- `Visit`

**Prompt**

> “¿Qué responsabilidades cumple esta clase dentro de la arquitectura?”

**Refinamiento**

> “¿Esta clase contiene lógica de negocio? ¿Por qué?”

**Discusión guiada**

- Controller vs dominio
- Lógica permitida vs indebida

### Ejercicio 2.2 — Flujo funcional completo

**Escenario**

Registrar una nueva visita para una mascota.

**Prompt**

"Describe el flujo completo cuando se registra una nueva visita, desde la petición HTTP hasta la persistencia"

**Actividad**

1. Copilot describe el flujo.
2. El alumno navega por el código siguiendo el flujo.
3. Se anotan pasos implícitos.

**Ejercicio 2.3 — Flujos alternativos**

**Prompt avanzado**

"¿Qué ocurre si los datos enviados no son válidos en este flujo?"

**Objetivo**

- Identificar validaciones
- Detectar manejo de errores

## BLOQUE 3 — Relaciones entre módulos y capas

**Objetivo**

Entender **cómo se conectan las piezas**, no solo qué hacen.

**Ejercicio 3.1 — Relación Controller → Service → Repository**

**Prompt**

"Explica cómo se relaciona esta clase con la capa de persistencia"

**Refinamiento**

"¿Esta relación respeta una arquitectura por capas?"

**Ejercicio 3.2 — Dependencias implícitas**

**Archivo**

- Cualquier Controller

**Prompt**

> “¿De qué clases depende directamente esta clase y por qué?”

**Discusión**

- Dependencias explícitas vs implícitas
- Acoplamiento

**Ejercicio 3.3 — Mapa mental de arquitectura**

**Prompt**

> “Describe los principales módulos del sistema y cómo interactúan entre sí”

**Actividad**

- El alumno dibuja un diagrama simple basado en la respuesta.
- Se ajusta manualmente según el código real.

## BLOQUE 4 — Supuestos implícitos y zonas ambiguas

**Objetivo**

Usar Copilot para **pensar críticamente** sobre el código.

**Ejercicio 4.1 — Supuestos del dominio**

**Prompt**

> “¿Qué supuestos de negocio parecen estar implícitos en este código?”

**Ejemplos esperados**

- Existencia previa de Owner
- Reglas no validadas explícitamente

**Ejercicio 4.2 — Zonas ambiguas o frágiles**

**Prompt**

> “¿Qué partes de este código podrían generar confusión o errores futuros?”

**Validación**

- Revisar nombres
- Responsabilidades mezclada
- Falta de documentación

**Ejercicio 4.3 — Qué NO queda claro**

**Prompt avanzado**

> “¿Qué información no puede inferirse solo a partir de este código?”

**Aprendizaje clave**

- Copilot reconoce límites
- Identificación de deuda de conocimiento

**CIERRE DEL LABORATORIO**

**Reflexión guiada**

- ¿En qué ayudó más Copilot?
- ¿Dónde fue impreciso?
- ¿Qué preguntas humanas siguen siendo necesarias?

**Mensaje final (slide)**

> GitHub Copilot no entiende el sistema por ti,
> **te ayuda a hacer mejores preguntas más rápido**.