Laboratorio Guiado

Del Código a Historias de Usuario con GitHub Copilot

Proyecto: `dotnet-architecture/eShopOnWeb`

https://github.com/NimblePros/eShopOnWeb.git

Objetivo del laboratorio

Usar GitHub Copilot Chat para:

1. Traducir código técnico a lenguaje funcional.
2. Extraer comportamiento real desde el código.
3. Redactar historias de usuario alineadas a lo que el sistema realmente hace.
4. Mejorar criterios de aceptación basados en lógica existente.
5. Identificar edge cases implícitos en la implementación.

Requisitos

- Visual Studio / VS Code
- GitHub Copilot activo

Proyecto sugerido para análisis

Trabajar sobre:

- `/src/Web/Controllers/BasketController.cs`
- `/src/Core/Services/BasketService.cs`
- `/src/Web/Endpoints/CatalogItemListPagedEndpoint.cs`
- `/src/Core/Entities/Order.cs`

BLOQUE 1 – Comprensión funcional desde código

Ejercicio 1: Entender un Endpoint real

Abrir:

- `CatalogItemListPagedEndpoint.cs`

Prompt 1 (Comprensión funcional)

```
Actúa como Product Owner técnico.
Explícame en lenguaje funcional qué hace este endpoint.
Evita términos técnicos y enfócate en comportamiento de
negocio.
```

Resultado esperado del alumno

- Entiende que:
    - Lista productos del catálogo
    - Permite paginación
    - Permite filtrar por marca y tipo
    - Devuelve resultado paginado

Ejercicio 2: Traducir código técnico a lenguaje negocio

Prompt 2

```
Traduce la lógica de este endpoint a lenguaje entendible por
negocio.
Describe qué problema resuelve y para quién.
```

Reflexión guiada

El alumno debe detectar:

- No es “un endpoint”
- Es “consulta de productos del catálogo con filtros”

- El usuario final es comprador

BLOQUE 2 – Generación de Historia de Usuario

Ejercicio 3: Generar Historia desde código

Prompt clave

```
Genera una historia de usuario basada en este endpoint.
Incluye:
- Rol
- Necesidad
- Beneficio
- Criterios de aceptación en formato Given/When/Then
```

Resultado esperado (ejemplo correcto)

Historia:

Como comprador
Quiero ver una lista paginada de productos
Para poder navegar fácilmente el catálogo

Criterios de aceptación:

- Given existen productos
- When consulto la página 1
- Then obtengo máximo N productos
- Given filtro por marca
- When consulto productos
- Then solo veo productos de esa marca

PARTE 3 – Extraer criterios desde lógica real

Ir a:

- `BasketService.cs`

Identificar métodos como:

- AddItemToBasket
- SetQuantities
- DeleteBasketAsync

Ejercicio 4: Extraer criterios reales

Prompt

```
Extrae criterios de aceptación basados únicamente en la lógica
real implementada en este servicio.
No inventes comportamiento que no exista en el código.
```

Objetivo didáctico

El alumno debe notar:

- Si el producto ya existe → aumenta cantidad
- Si no existe → lo agrega
- Si cantidad es 0 → se elimina
- No hay validación de stock (importante detectar lo que NO hace)

PARTE 4 – Identificación de Edge Cases

Ejercicio 5: Analizar escenarios límite

Sobre el método:

- `AddItemToBasket`

Prompt clave

```
¿Qué escenarios edge cases cubre este método?
¿Cuáles no cubre?
Clasifícalos en:
- Cubiertos explícitamente
```

```
- Implícitos
- No cubiertos
```

Resultado esperado

Copilot puede identificar:

Cubiertos

- Producto repetido
- Cantidad acumulada

No cubiertos

- Stock insuficiente
- Producto inexistente
- Usuario no autenticado
- Cantidades negativas

Aquí el instructor debe reforzar:

> Copilot no inventa reglas; analiza lo que el código realmente hace.

PARTE 5 – Validación crítica de Copilot

Ejercicio 6: Detectar errores de interpretación

Prompt

```
¿Hay alguna suposición en tu respuesta que no esté
directamente respaldada por el código?
```

Esto entrena:

- Pensamiento crítico
- No confiar ciegamente en la IA
- Validar contra código real

PARTE 6 – Alinear Desarrollo y Negocio

Ejercicio 7: Gap Analysis

Prompt

```
Compara esta historia de usuario con el comportamiento real
del código.
¿Qué diferencias encuentras?
```

Objetivo:

Detectar:

- Historias demasiado generales
- Criterios inventados
- Falta de validaciones reales
- Reglas que negocio cree que existen pero no están implementadas

PARTE 7 – Refactor de Historia Mejorada

Ejercicio 8: Mejorar criterios

Prompt

```
Reescribe los criterios de aceptación alineándolos
estrictamente a la lógica existente.
Evita agregar comportamiento no implementado.
```

PARTE 8 – Ejercicio avanzado (Arquitectura + negocio)

Abrir:

- `Order.cs`

Prompt

```
Extrae el modelo de negocio implícito en esta entidad.
¿Qué reglas del dominio puedes identificar?
```

Objetivo:

- Identificar agregados
- Entender encapsulamiento
- Detectar invariantes

PARTE 9 – Caso Integrador

El alumno debe:

1. Elegir un endpoint

2. Generar:
    - Historia de usuario
    - 5 criterios de aceptación
    - 3 edge cases
    - 2 riesgos funcionales

3. Validarlo contra el código