1. ¿Puedes identificar pruebas de unidad y de integración en la práctica que se realizó?

Si para el caso de pruebas de unidad seria para el caso del método de eliminar un producto y para agregar un producto, mientras que una prueba de integración se ve dentro del test del método de buscado, pues primero tenemos que agregar un producto a nuestra tienda para luego a través de ese producto agregado verifiquemos si el método realiza la búsqueda de manera correcta o no

1. ¿Podría haber escrito las pruebas primero antes de modificar el código de la aplicación? ¿Cómo sería el proceso de escribir primero los tests? Describe el proceso con tus palabras.

Sí, este enfoque es conocido como

**Desarrollo Guiado por Pruebas** (TDD, por sus siglas en inglés: Test-Driven Development). La filosofía de TDD establece que primero se escriben las pruebas, luego el código que hace que esas pruebas pasen, y finalmente se refactoriza el código.

2. ¿Cómo sería el proceso de escribir primero los tests?

El proceso de TDD se puede describir en tres pasos cíclicos:

1. **Rojo (Red) 🔴:** Escribes una prueba para una nueva funcionalidad. La prueba falla de inmediato porque la funcionalidad aún no existe.
2. **Verde (Green) 🟢:** Escribes la cantidad mínima de código de producción para que la prueba que acabas de escribir pase. En esta etapa, el código puede no ser el más limpio o eficiente.
3. **Refactor (Refactor) 🟡:** Una vez que la prueba pasa, refactorizas el código para mejorar su diseño y estructura, sin cambiar su comportamiento. Si alguna prueba falla en esta etapa, sabes que has introducido un error, por lo que reviertes los cambios y corriges el problema. Luego, se repite el ciclo para la siguiente funcionalidad.
4. En lo que va del trabajo práctico, ¿puedes identificar 'Controladores' y 'Resguardos'? ¿Qué es un “test double”? ¿Hay otros nombres para los objetos/funciones simulados?

En el contexto de las pruebas de software, estos términos se refieren a componentes que ayudan a aislar y ejecutar la unidad bajo prueba (UUT):

* **Controladores (Drivers/Controladores):** Es el código que **ejecuta** el test y llama a la unidad bajo prueba. Actúa como el motor del test.
  + **En tu código:** Cada método marcado con **[Test]** (por ejemplo, eliminarProductoDeberiaEliminarlo o AplicarDescuento\_DeberiaLlamarACambiarPrecio\_ConElPrecioCalculado) es el **Controlador**. Es la pieza de código que orquesta el Arrange, llama al método Act (tienda.eliminarProducto) y realiza el Assert.
* **Resguardos (Stubs/Fakes/Dummies):** Son dobles de prueba que **simulan** el comportamiento de los componentes de los que depende la UUT. Se utilizan para *devolver* valores fijos o evitar que se ejecuten partes complejas del código.
  + **En tu código:** El **MockProducto** que creaste es un tipo de resguardo (más específicamente, un Mock que también actúa como Stub) porque, mediante mockProducto.SetupGet(p => p.Precio).Returns(precioOriginal), lo estás configurando para **devolver un valor fijo** (el precio simulado) cuando la clase Tienda lo necesita.

Un **Test Double** (Doble de Prueba) es un término general que engloba cualquier objeto o componente que **reemplaza un componente real** con fines de prueba. Se usa para aislar la unidad bajo prueba, evitar efectos secundarios (como llamadas a una base de datos o servicios externos) y controlar el entorno de la prueba.

Sí, existen varios tipos de dobles de prueba, cada uno con un propósito ligeramente distinto:

* **Dummies:** Objetos que se pasan como argumentos, pero nunca se usan.
* **Stubs (Resguardos):** Proporcionan respuestas preprogramadas a las llamadas de métodos durante la prueba (como tu mockProducto.SetupGet).
* **Spies:** Objetos reales que registran información sobre cómo fueron llamados.
* **Fakes:** Objetos con implementaciones funcionales, pero más simples y no aptas para producción (ej. usar una base de datos en memoria en lugar de SQL Server).
* **Mocks:** Objetos que, además de simular el comportamiento (Stubbing), establecen **expectativas** sobre las llamadas que deberían recibir. La prueba falla si estas expectativas (verificaciones con Verify) no se cumplen. (Tu MockProducto es este tipo).