**Guía de Laboratorio #1**

Julian Ruiz Guzmán

Keiry Lucia Olaya

Julian Aya Orozco

Olga Lucía Roa Bohórquez

Universidad Manuela Beltrán

Taller de Programación

Ingeniería de Software

2024

Contenido

[Preguntas Orientadoras 2](#_Toc1984724531)

[Requerimientos 3](#_Toc246260877)

[Requerimiento Funcional 2: Actualización de Datos de Cliente 4](#_Toc2123855977)

[Requerimiento Funcional 3: Eliminar de Cliente 5](#_Toc308965879)

[Requerimiento Funcional 4: Registro de Producto 5](#_Toc968922977)

[Requerimiento Funcional 5: Actualización de Producto 5](#_Toc779621968)

[Requerimiento Funcional 6: Registro de Venta 6](#_Toc932156441)

[Requerimiento Funcional 7: Consulta de Inventario 6](#_Toc1109063386)

[Requerimiento Funcional 8: Reporte de Ventas Diarias 7](#_Toc377515565)

[Requerimiento Funcional 9: Generación de Factura 7](#_Toc1782180271)

[Requerimiento Funcional 10: Autenticación de Usuario 7](#_Toc1409514863)

[Mockups 8](#_Toc1481382824)

[Requerimiento Funcional 4: Registro de Producto 10](#_Toc784646420)

[Requerimiento Funcional 5: Actualización de Producto 10](#_Toc1030012002)

[Requerimiento Funcional 6: Registro de Venta 11](#_Toc933229990)

[Requerimiento Funcional 7: Consulta de Inventario 12](#_Toc1838613832)

[Requerimiento Funcional 8: Reporte de Ventas Diarias 12](#_Toc505405423)

[Requerimiento Funcional 9: Generación de Factura 12](#_Toc1103583076)

[Requerimiento Funcional 10: Autenticación de Usuario 13](#_Toc1847622206)

[Manejo de situaciones anormales 14](#_Toc1820452445)

[Criterios de Aceptación 15](#_Toc1039364833)

[Diagrama de clases aplicando UML, Tienda de Ropa y Moda. 18](#_Toc1567429417)

[Actividad de Trabajo Autónomo 18](#_Toc844622968)

[Pantallazos SoloLearn 21](#_Toc180900636)

[Referencias Bibliográficas 22](#_Toc847014402)

# Preguntas Orientadoras

1. ¿Cuáles fueron los aprendizajes obtenidos al realizar esta guía? Liste como mínimo 3 aprendizajes y relaciónelos con su futuro quehacer profesional.

Aprendizajes:

* Estructuración de Requerimientos Funcionales: Aprendí a estructurar los requerimientos funcionales de una manera clara y organizada, lo que es crucial para el desarrollo de software efectivo en mi futura carrera profesional.
* Importancia de la Documentación Completa: Aprendí la importancia de mantener una documentación detallada y completa, lo cual facilita la comunicación entre los miembros del equipo y asegura que todos comprendan los objetivos y restricciones del proyecto.
* Manejo de Situaciones Anormales: Aprendí a identificar y manejar situaciones anormales en el software, lo cual es esencial para asegurar la robustez y fiabilidad del software que desarrollé en el futuro.
  + 1. ¿Dónde presento mayor dificultad resolviendo la guía? ¿Cómo lo resolvieron? ¿Cuáles fueron las estrategias de solución?

Dificultades y Soluciones:

* Dificultad: Identificar todos los posibles escenarios de entrada de datos y situaciones anormales.
* Solución: Se realizó una lluvia de ideas en equipo para listar todas las posibles entradas y situaciones anormales.

Estrategias: Utilizamos ejemplos prácticos y revisamos documentación de proyectos similares para asegurarnos de no omitir ningún detalle crítico.

# Requerimientos

***Requerimiento Funcional 1: Registro de Usuario***

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **SALIDA** | **DESCRIPCIÓN** |
| código: String (15) | Registro creado correctamente en la base de datos | Validación: El código debe tener exactamente 15 caracteres. |
| nombre: String (max 50) | Retorno del ID del registro creado | Se verifica que el nombre no esté vacío. |
| apellido: String (max 50) |  | Se realiza la concatenación de nombre y apellido para formar el campo completo de nombre. |

### 

### Requerimiento Funcional 2: Actualización de Datos de Cliente

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **SALIDA** | **DESCRIPCIÓN** |
| id\_cliente: Integer | Registro actualizado correctamente | Se verifica que el id\_clien te exista en la base de datos. |
| nombre: String (max 50) | Confirmación de la actualización de datos | Validación: El nuevo nombre no debe ser nulo o vacío. |
| apellido: String (max 50) |  | Solo se actualizan los campos que contienen datos. |

### Requerimiento Funcional 3: Eliminar de Cliente

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **SALIDA** | **DESCRIPCIÓN** |
| id\_cliente: Integer | Cliente eliminado correctamente | Validación: Se verifica que el id\_cliente exista antes de eliminarlo. |
|  |  | - Se eliminará el registro del cliente de la base de datos. |

### Requerimiento Funcional 4: Registro de Producto

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **SALIDA** | **DESCRIPCIÓN** |
| codigo\_producto: String (10) | Producto registrado en la base de datos | Validación: El código del producto debe ser único y tener un máximo de 10 caracteres. |
| nombre\_producto: String (max 50) | Retorno del ID del producto creado | Se verifica que el nombre del producto no esté vacío. |

### Requerimiento Funcional 5: Actualización de Producto

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **SALIDA** | **DESCRIPCIÓN** |
| id\_producto: Integer | Producto actualizado correctamente | Validación: El id\_producto debe existir en la base de datos. |
| precio: Decimal | Confirmación de la actualización de precio | Validación: El precio debe ser mayor a 0. |

### Requerimiento Funcional 6: Registro de Venta

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **SALIDA** | **DESCRIPCIÓN** |
| id\_cliente: Integer | Venta registrada exitosamente | Validación: El id\_cliente debe existir en la base de datos. |
| id\_producto: Integer | Retorno del ID de la venta creada | Se verifica la existencia del id\_producto antes de registrar la venta. |
| cantidad: Integer |  | Validación: La cantidad debe ser mayor a 0. |

### Requerimiento Funcional 7: Consulta de Inventario

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **SALIDA** | **DESCRIPCIÓN** |
| id\_producto: Integer | Información del inventario del producto | Validación: Se verifica la existencia del id\_producto antes de consultar. |
|  | Cantidad disponible en inventario | Se retorna la cantidad disponible y el detalle del producto. |

### Requerimiento Funcional 8: Reporte de Ventas Diarias

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **SALIDA** | **DESCRIPCIÓN** |
| fecha: Date | Reporte de ventas generado | Validación: La fecha debe estar en un formato válido. |
|  | Resumen de ventas del día | Se genera un reporte con el total de ventas y los productos vendidos en la fecha especificada. |

### Requerimiento Funcional 9: Generación de Factura

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **SALIDA** | **DESCRIPCIÓN** |
| id\_venta: Integer | Factura generada en formato PDF | Validación: El id\_venta debe existir en la base de datos. |
|  | Detalle de la venta con impuestos aplicados | Se calcula el total incluyendo impuestos y se genera el documento. |

### Requerimiento Funcional 10: Autenticación de Usuario

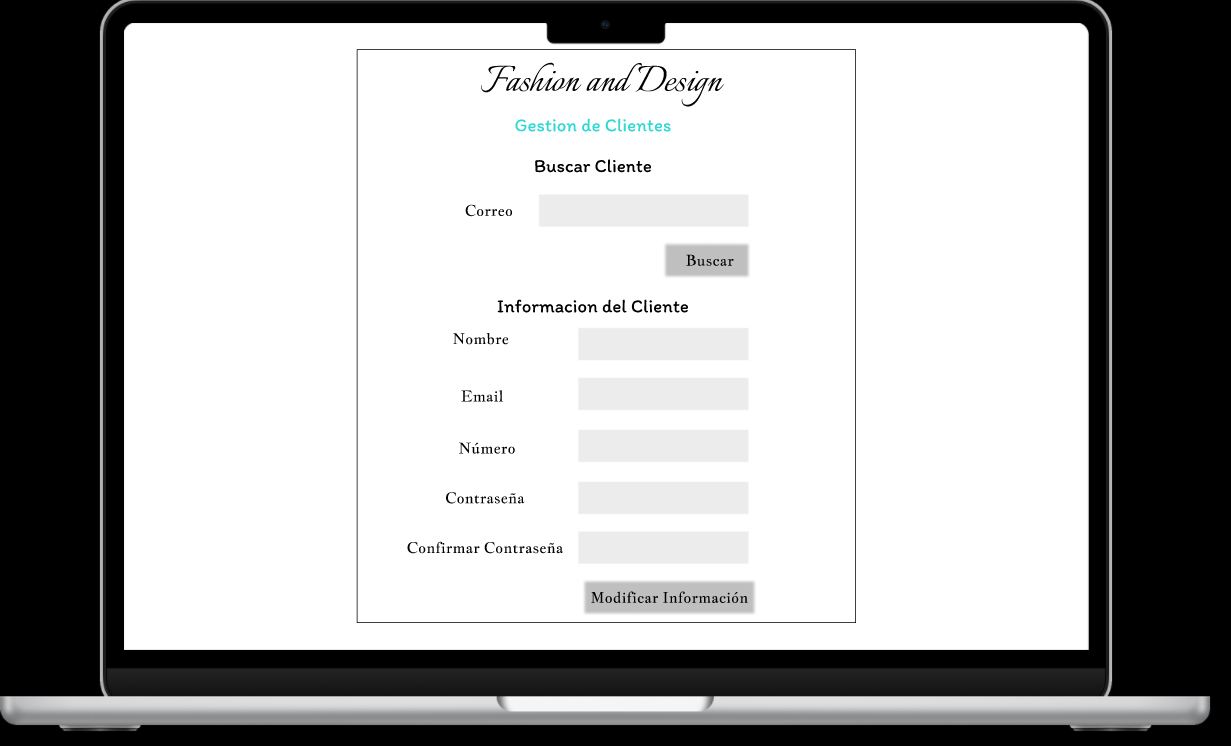
|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **SALIDA** | **DESCRIPCIÓN** |
| usuario: String (max 30) | Acceso concedido o denegado | Validación: El nombre de usuario y la contraseña deben coincidir con un registro existente. |
| contraseña: String (max 20) | Generación de un token de sesión | Se encripta la contraseña antes de comparar. |

# Mockups

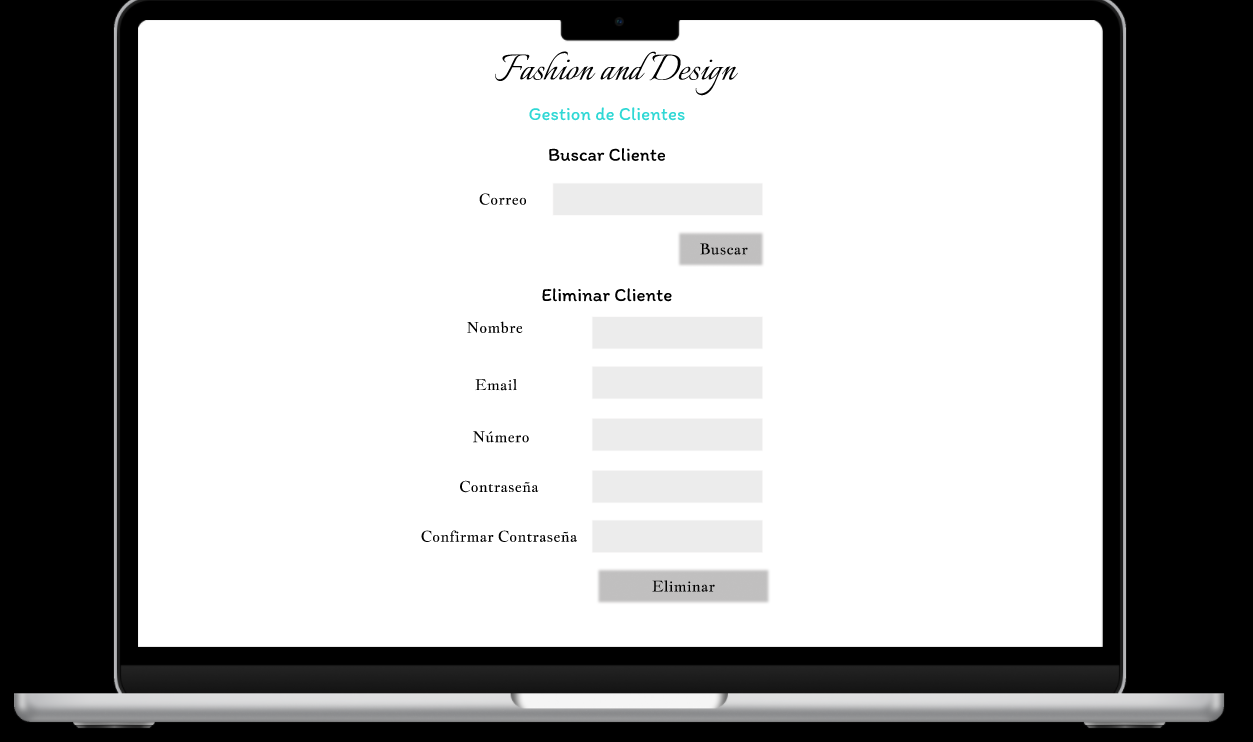
***Requerimiento Funcional 1: Registro de Usuario***



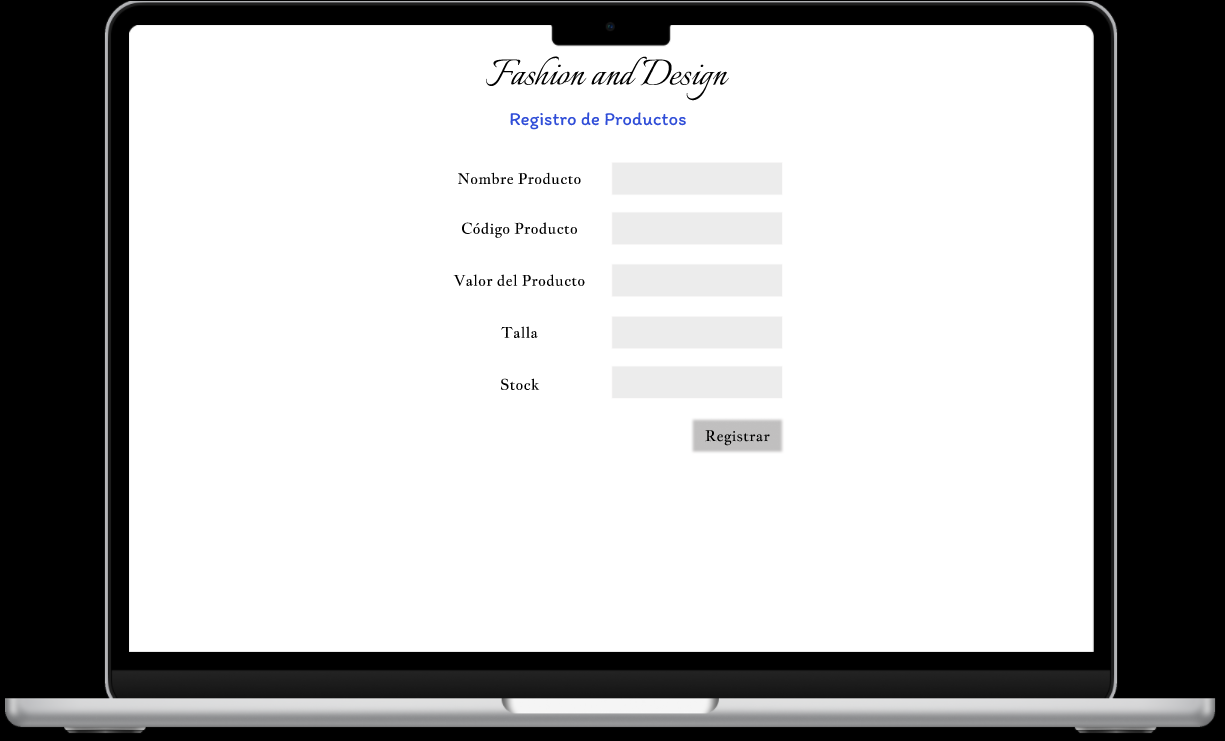
**Requerimiento Funcional 2: Actualización de Datos de Cliente**



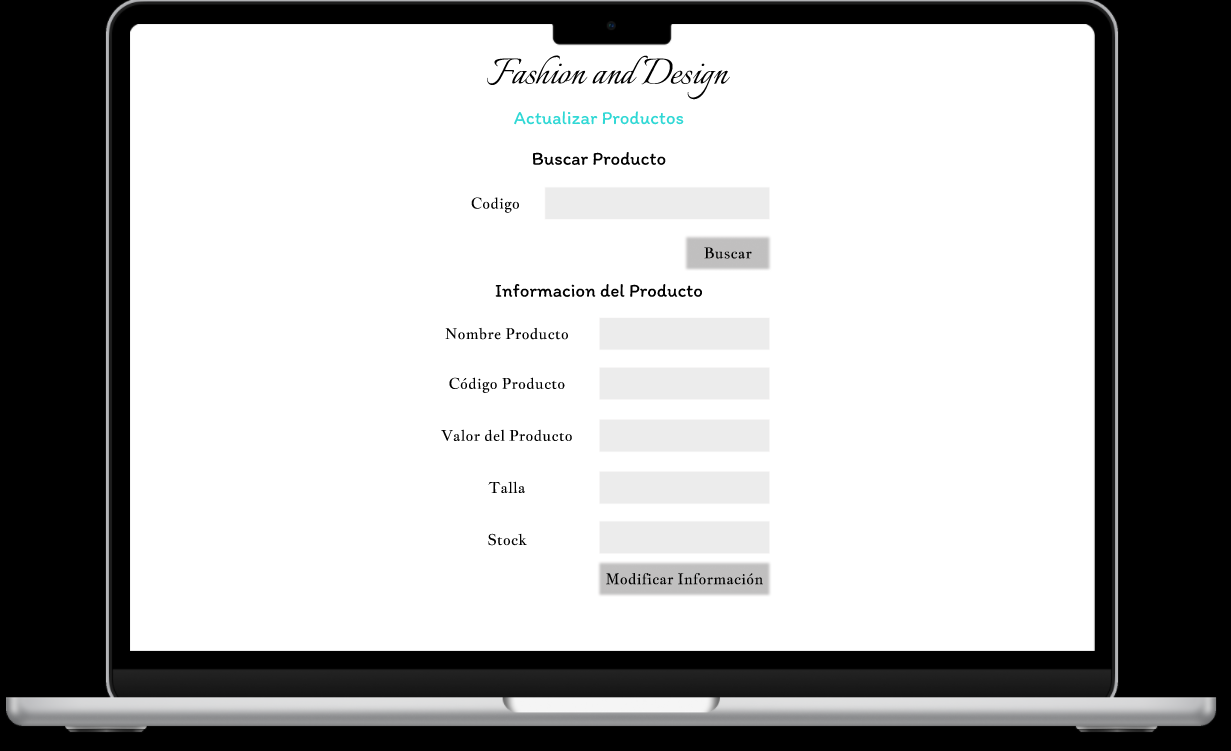
**Requerimiento Funcional 3: Eliminar de Cliente**



### Requerimiento Funcional 4: Registro de Producto



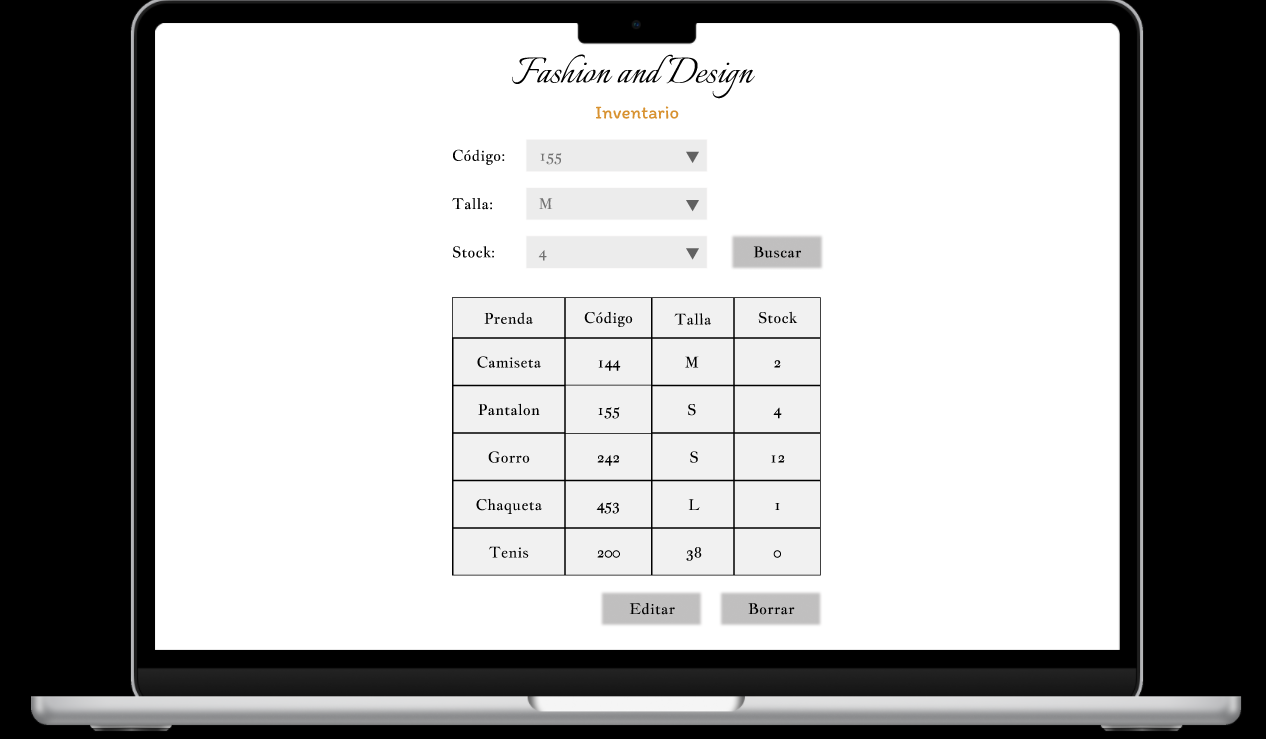
### Requerimiento Funcional 5: Actualización de Producto



### Requerimiento Funcional 6: Registro de Venta



### Requerimiento Funcional 7: Consulta de Inventario



### Requerimiento Funcional 8: Reporte de Ventas Diarias



### Requerimiento Funcional 9: Generación de Factura



### Requerimiento Funcional 10: Autenticación de Usuario



# Manejo de situaciones anormales

1. **Violaciones a la Precondición**

* **Compra sin productos en el carrito:** El sistema muestra un mensaje de error indicando que el carrito está vacío.
* **Acceso no autorizado:** Se deniega el acceso y se muestra un mensaje indicando la falta de permisos.
* **Datos de registro inválidos:** El sistema valida los datos y muestra errores específicos hasta que sean correctos.
* **Producto sin stock:** Se informa al usuario que el producto no está disponible.

1. **Situaciones Anormales con Recursos**

* **Espacio en disco insuficiente:** El sistema alerta al administrador, limpia archivos temporales y limita funciones que requieren almacenamiento.
* **Memoria insuficiente:** El sistema libera memoria, balancea la carga y limita funciones pesadas.
* **Sobrecarga por alta concurrencia:** El sistema escala recursos, usa caché y gestiona solicitudes en cola para evitar sobrecarga.

# Criterios de Aceptación

Criterios de Aceptación Generales:

**Entradas Válidas y Salidas Válidas:**

* + **Precondición:** Cuando un usuario realiza una acción válida (e.g., agregar un producto, registrar un usuario), la entrada debe cumplir con las reglas de negocio (e.g., campos obligatorios, formato correcto).
  + **Postcondición:** El sistema debe generar la salida esperada. Por ejemplo, si se agrega un producto, este debe aparecer en el inventario con la información correcta.

**Manejo de Situaciones Anormales:**

* + **Precondición:** Si se intenta realizar una acción con datos inválidos (e.g., omitir un campo obligatorio), el sistema debe manejarlo apropiadamente.
  + **Postcondición:** El sistema debe mostrar un mensaje de error claro y no debe realizar ninguna acción incorrecta (e.g., no agregar un producto con datos incompletos).

***Criterios de Aceptación Específicos:***

**Persistencia de Datos:**

* + **Criterio:** Toda la información que se guarda (e.g., productos, usuarios, transacciones) debe ser almacenada de manera persistente en la base de datos.
  + **Verificación:** Después de realizar una operación que afecta los datos (e.g., agregar, modificar, eliminar un producto), se debe verificar que la información se ha guardado correctamente y puede recuperarse en posteriores consultas.
  + **Prueba:** Ejecutar una operación y luego verificar que los datos están en la base de datos, y se pueden consultar sin pérdida de información.

**Desempeño y Utilización de Recursos:**

* + **Criterio:** El sistema debe responder dentro de un tiempo aceptable para todas las operaciones principales (e.g., menos de 2 segundos para agregar un producto al inventario).
  + **Verificación:** Realizar pruebas de carga y desempeño para asegurar que el sistema funcione adecuadamente bajo diferentes condiciones (e.g., con 100 usuarios simultáneos).
  + **Prueba:** Medir el tiempo de respuesta de las operaciones clave y comparar con los tiempos establecidos en los requisitos.

**Seguridad:**

* + **Criterio:** El sistema debe proteger la información sensible de los usuarios (e.g., contraseñas encriptadas, roles de usuario).
  + **Verificación:** Implementar pruebas de seguridad para verificar que se cumplen las políticas de acceso y protección de datos.
  + **Prueba:** Intentar acceder a funciones restringidas con un usuario no autorizado y verificar que el acceso es denegado.

# Diagrama de clases aplicando UML, Tienda de Ropa y Moda.

Diagrama

Descripción generada automáticamente

# Actividad de Trabajo Autónomo

|  |  |  |  |
| --- | --- | --- | --- |
| ***Paradigma de Programación*** | ***Ventajas*** | ***Desventajas*** | ***Lenguaje de Programación*** |
| Programación Imperativa | Control detallado sobre el flujo del programa.  Fácil de entender para tareas simples. | Puede volverse complejo y difícil de mantener en programas grandes.  Mayor riesgo de errores. | C. Pascal |
| Programación  Declarativa | Código más conciso y fácil de leer.  Reduce errores al no necesitar detallar pasos. | Menos control sobre cómo se ejecutan los procesos.  Puede ser difícil de entender para novatos. | Haskell  Prolog |
| Programación **Orientada a Objetos** | Reutilización de código a través de la herencia.  Facilita el modelado de sistemas complejos. | Puede introducir sobrecarga y complejidad innecesaria.  Dificultad para gestionar la herencia múltiple. | Java  Python  C++ |
| Programación Orientada a Aspectos | Modula aspectos transversales como la seguridad.  Mejora la separación de preocupaciones. | Puede ser difícil de entender y depurar.  No es ampliamente soportado por muchos lenguajes. | AspectJ |
| Programación Orientada a Aspectos | Evita efectos secundarios y estados mutables.  Facilita la concurrencia y paralelismo. | Puede ser menos intuitivo para programadores acostumbrados a paradigmas imperativos.  Requiere un cambio de mentalidad. | Lisp  Haskell  Erlang |
| Programación Lógica | Adecuada para problemas que pueden expresarse en términos de reglas lógicas.  Facilita la creación de programas que "piensan". | Limitada en cuanto a eficiencia y aplicación  práctica en comparación con otros paradigmas. | Prolog |
| Programación Reactiva | Adecuada para aplicaciones que responden a eventos en tiempo real.  Simplifica el manejo de eventos asíncronos. | Puede ser difícil de depurar y razonar sobre su comportamiento.  Mayor complejidad en la implementación. | RxJava  ReactiveX |
| Programación Basada en Eventos | Ideal para aplicaciones interactivas y con interfaces gráficas.  Facilita el diseño de sistemas reactivos. | Puede volverse difícil de gestionar en aplicaciones complejas.  Mayor riesgo de errores en la gestión de eventos. | JavaScript |

# Pantallazos SoloLearn

# Referencias Bibliográficas

Bustos, J. L. (2024b, abril 25). ¿Qué son los paradigmas de programación y qué tipos hay? *KeepCoding Bootcamps*. <https://keepcoding.io/blog/paradigmas-de-programacion/>

*¿Qué son los paradigmas de programación? | TripleTen México*. (2024, 17 enero). TripleTen. <https://tripleten.mx/blog/que-son-los-paradigmas-de-programacion/>