# Tarea 2: Desarrollar el modelo de comportamiento del sistema

## 2.1 Objetivos

### 2.1.1 Objetivo General

Modelar el comportamiento dinámico del sistema Drakon Store Web a través de la definición de operaciones, estados y transiciones de los objetos del sistema, para profundizar en el conocimiento de los requisitos funcionales y detectar posibles conflictos en su especificación.

### 2.1.2 Objetivos Específicos

• Analizar los requisitos funcionales definidos para el e-commerce, con el fin de detectar ambigüedades, contradicciones o vacíos que puedan afectar la lógica de compra, registro y administración.

• Especificar las operaciones del sistema necesarias para representar el comportamiento esperado frente a las acciones de los clientes (registro, login, exploración, compra) y del administrador (validación de pagos, generación de facturas).

• Identificar los tipos de objetos con comportamiento significativo (Usuario, Carrito, Pedido, Factura, Producto) y definir sus estados y transiciones, reflejando su ciclo de vida dentro de la tienda.

• Representar gráficamente el comportamiento del sistema utilizando diagramas de secuencia, de estados y de actividad, facilitando la comprensión del flujo de ejecución y las interacciones entre cliente, servidor y base de datos.

## 2.2 Descripción

En esta tarea modelamos cómo reacciona Drakon Store Web frente a las acciones de los usuarios (clientes y administradores) y a los eventos del entorno (validaciones, errores de pago, actualizaciones de inventario). A partir de los casos de uso y requisitos obtenidos en la fase de elicitación, se definen las operaciones esenciales del sistema y se identifican los objetos con comportamiento dinámico.  
  
El objetivo es asegurar que:  
- El cliente pueda registrarse, iniciar sesión, navegar el catálogo, agregar al carrito y completar la compra con feedback claro (éxito o error).  
- El sistema valide la sesión, procese los pagos y genere facturas de forma fiable.  
- El administrador reciba y valide los pagos, gestione errores y confirme facturación.  
  
Además, este análisis ayuda a descubrir conflictos en los flujos de interacción (por ejemplo, conciliación de inventario, errores de transacción o sesiones expiradas) antes de comenzar el diseño detallado y la implementación.

## 2.4 Productos entregables

Modelo de comportamiento como parte del Documento de Análisis del Sistema (DAS), que incluye:

### Operaciones del sistema

• RegistrarUsuario(nombre, email, contraseña)

• IniciarSesion(email, contraseña)

• AgregarAlCarrito(idProducto, cantidad, opciones)

• RevisarCarrito()

• SeleccionarMetodoPago(metodo)

• RealizarPago(detallesPago)

• ConsultarEstadoPedido(pedidoId)

• AdministrarInventario(productoId, ajusteCantidad)

### Objetos con comportamiento significativo

• Usuario: Registrado → Autenticado → Desconectado

• Carrito: Vacío → Con ítems → Listo para pago → Pagado

• Pedido: Creado → En procesamiento → Enviado → Entregado

• Producto: Disponible → Agotado → En reposición

• Factura: Generada → Pagada → Archivada

### Diagramas desarrollados

• Diagrama de Secuencia – Registro de Usuario.

• Diagrama de Secuencia – Inicio de Sesión.

• Diagrama de Secuencia – Proceso de Compra.

• Diagrama de Estados – Carrito de Compras.

• Diagrama de Actividad – Flujo de Compra.

• Diagrama de Estados – Pedido.

## Conflictos detectados durante el análisis

• Ambigüedad en métodos de pago: No se define prioridad ni manejo cuando un método falla (tarjeta vs. PayPal vs. transferencia).

• Sesiones expiran sin notificación: Falta especificar comportamiento y mensaje al usuario si su sesión caduca durante checkout.

• Sincronización de inventario: No hay flujo detallado para ajustar stock tras un fallo de pago o rollback de transacción.

• Errores de pasarela de pago no cubiertos: No se documentan rutas alternativas ante timeout o rechazo de la pasarela.

• Roles y permisos incompletos: No se precisa qué puede hacer un usuario autenticado vs. administrador en cada operación.

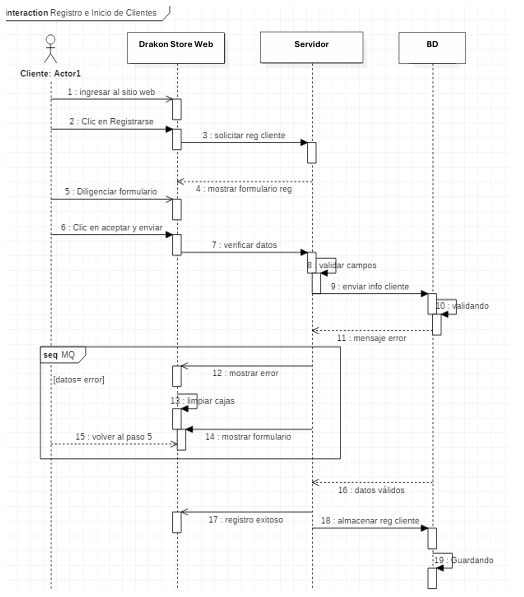
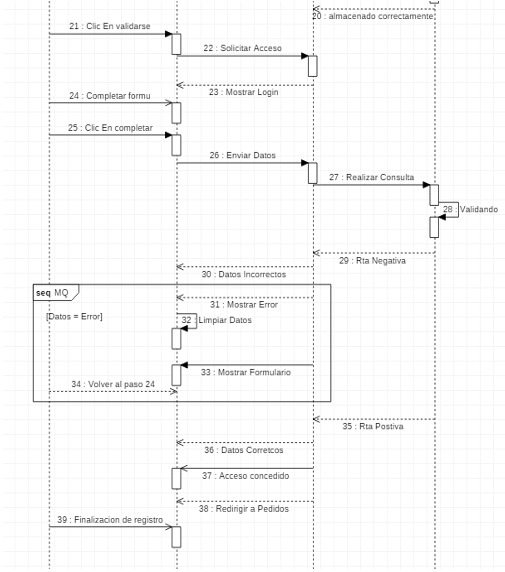
## 2.5 Técnicas

### 2.5.1 Especificación de operaciones

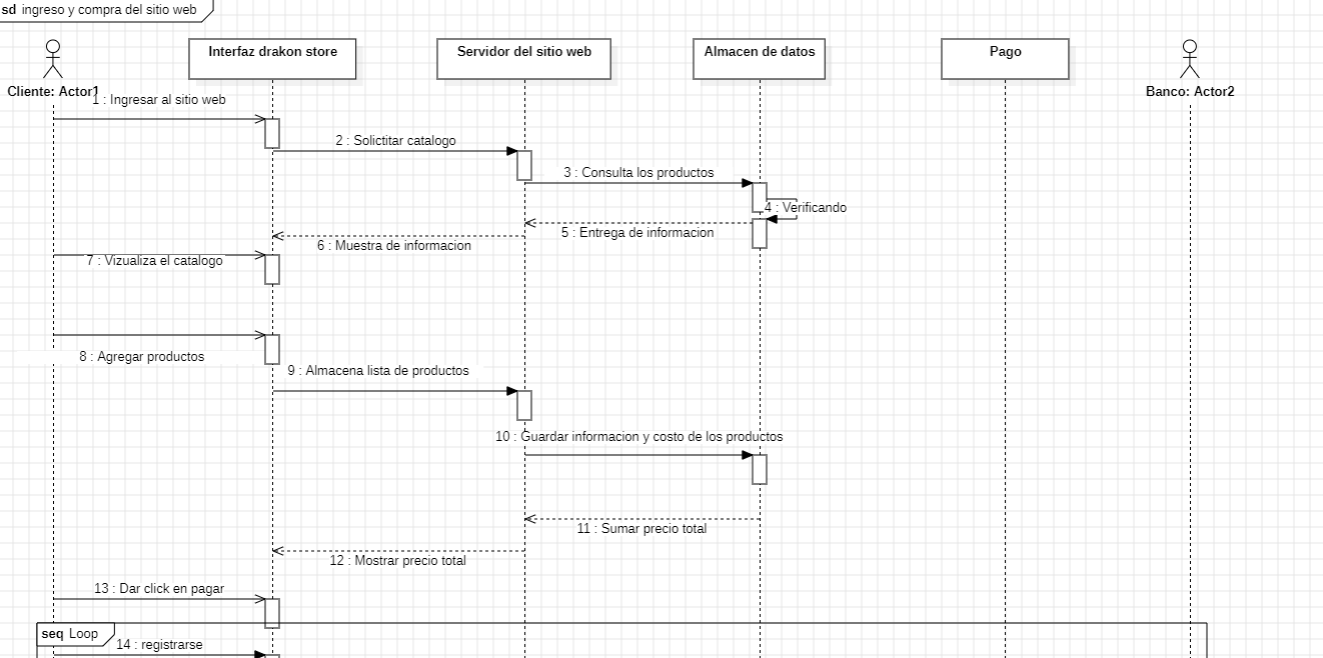
Plantillas formales con precondiciones, postcondiciones y manejo de excepciones.  
Ejemplo:  
acción RealizarPago(pedidoId: ID, método: String, datosPago: InfoPago)  
pre: pedido existe ∧ estadoPedido = “Listo para pago” ∧ método ∈ {“tarjeta”, “paypal”, “transferencia”}  
post:  
- pago exitoso ∧ estadoPedido = “En procesamiento” ∧ factura generada  
- pago denegado ∧ respuesta = “Error: pago rechazado”  
- timeout pasarela ∧ respuesta = “Error: tiempo de espera agotado”

### 2.5.2 Diagramas de traza de eventos (secuecia)

* Registro e inico de cliente.



* Registro e inico de cliente.



### 2.5.3 Diagramas de estados

