Ejercicio 1

Actores humanos

- Administrador

Otros objetos físicos

- Sensor infrarrojo (dispositivo físico)

Sistemas de software

- Cámara Web
- Sistema de control de calidad

Tipo de comunicaciones

- Son mayoritariamente **síncronas** (llamadas con espera de respuesta), ya que cada paso depende del resultado del anterior.
- No se observa ninguna comunicación claramente asíncrona en el dibujo.

Loops o alternativas

- No se ve un fragmento UML explícito de tipo loop/alt. Es una secuencia lineal.

Numeración de acciones

1. Administrador inicia la inspección al sensor infrarrojo.
2. El sensor detecta botellas y avisa a la cámara web.
3. La cámara captura imagen y la envía al sistema de control de calidad.
4. El sistema procesa la imagen.
5. El sistema revisa el nivel de llenado.
6. El sistema detecta impurezas.
7. El sistema guarda los resultados.

8. El sistema devuelve los resultados hacia atrás hasta el administrador.

Ejercicio 2

Actores humanos

- No se ve claramente un actor humano explícito; parece un flujo entre componentes del sistema.

Clases / paquetes

- Sensor infrarrojo
- Cámara Web
- Sistema de control de calidad
  Cada uno puede considerarse una clase o componente dentro de un paquete de control de calidad/visión artificial.

Comunicaciones síncronas y asíncronas

- Síncronas: llamadas como “captura imagen”, “procesamiento de imágenes”, “revisa nivel de llenado”, “detecta impurezas”, “guarda los resultados”, porque hay retorno de control.
- No se aprecia una comunicación asíncrona explícita con flecha abierta o sin retorno.

Parámetros

- En “captura imagen” implícitamente se pasa la imagen capturada.
- En “procesamiento de imágenes” se pasa la imagen como parámetro.
- En “guarda los resultados” se pasan los resultados del análisis.

Ejercicio 3

Actores

- Person (cliente/usuario)

Objetos / clases

- GUI
- aProduct / Product
- ShoppingCart
- aCartItem / CartItem
- Order
- ShippingInfo

Loops / alternativas

- Hay un comportamiento repetitivo al añadir productos al carrito (implícito, aunque no esté marcado con un fragmento loop).
- Hay una alternativa clara: si el cliente no está logueado, se ejecuta el login.

¿Falta algún loop?

- Sí, podría añadirse explícitamente un fragmento `loop` para "añadir productos al carrito varias veces".

¿Cómo cambiaría si el login se limita a 3 intentos?

- Se añadiría un fragmento `loop (max 3)` alrededor del mensaje de login, con una alternativa `alt` para éxito o bloqueo tras 3 fallos.

Numeración de acciones (resumen)

1. El usuario selecciona producto.
2. La GUI pide detalles del producto.
3. Se añaden ítems al carrito.
4. Se calcula el precio del ítem.

5. Se comprueba el carrito.
6. Si no está logueado, se solicita login.
7. Se actualizan datos de envío.
8. Se calcula el total.
9. Se crea el pedido.
10. Se notifica confirmación del pedido.

Por qué algunas flechas hacia atrás son continuas y otras no

- Las continuas suelen representar retornos síncronos normales.
- Las discontinuas suelen representar respuestas o mensajes de retorno simplificados, no llamadas nuevas.

Ejercicio 4

Boundary (interfaz)

- IU Menu
- IU Proveedor

Control

- Proveedor controller

Entity

- Proveedor
- Vehículo

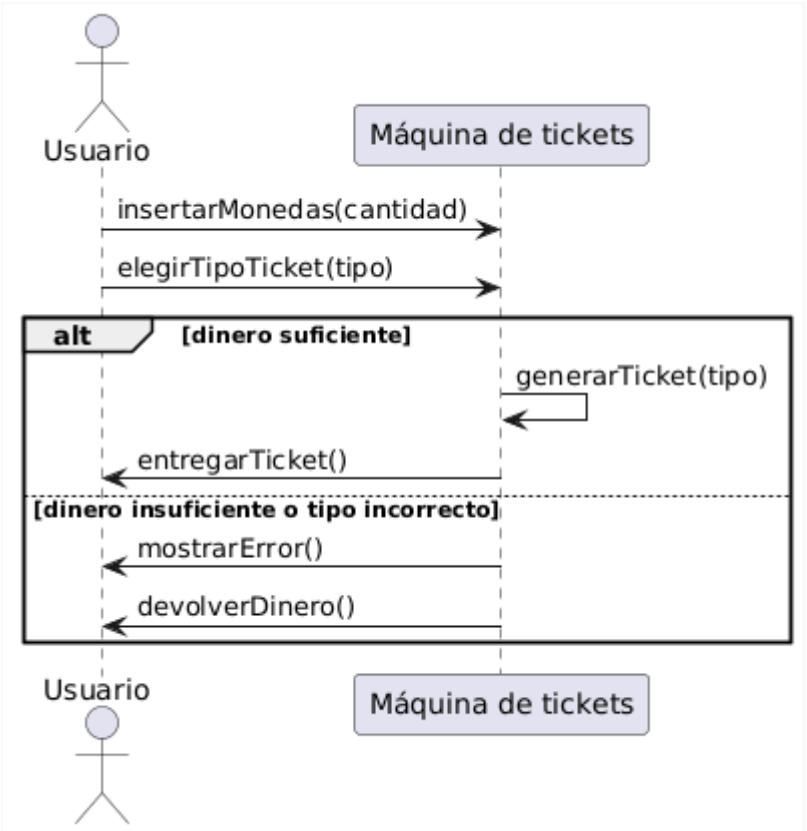
El actor “Administrador de Vehículos” es el usuario externo al sistema.

Ejercicio 5

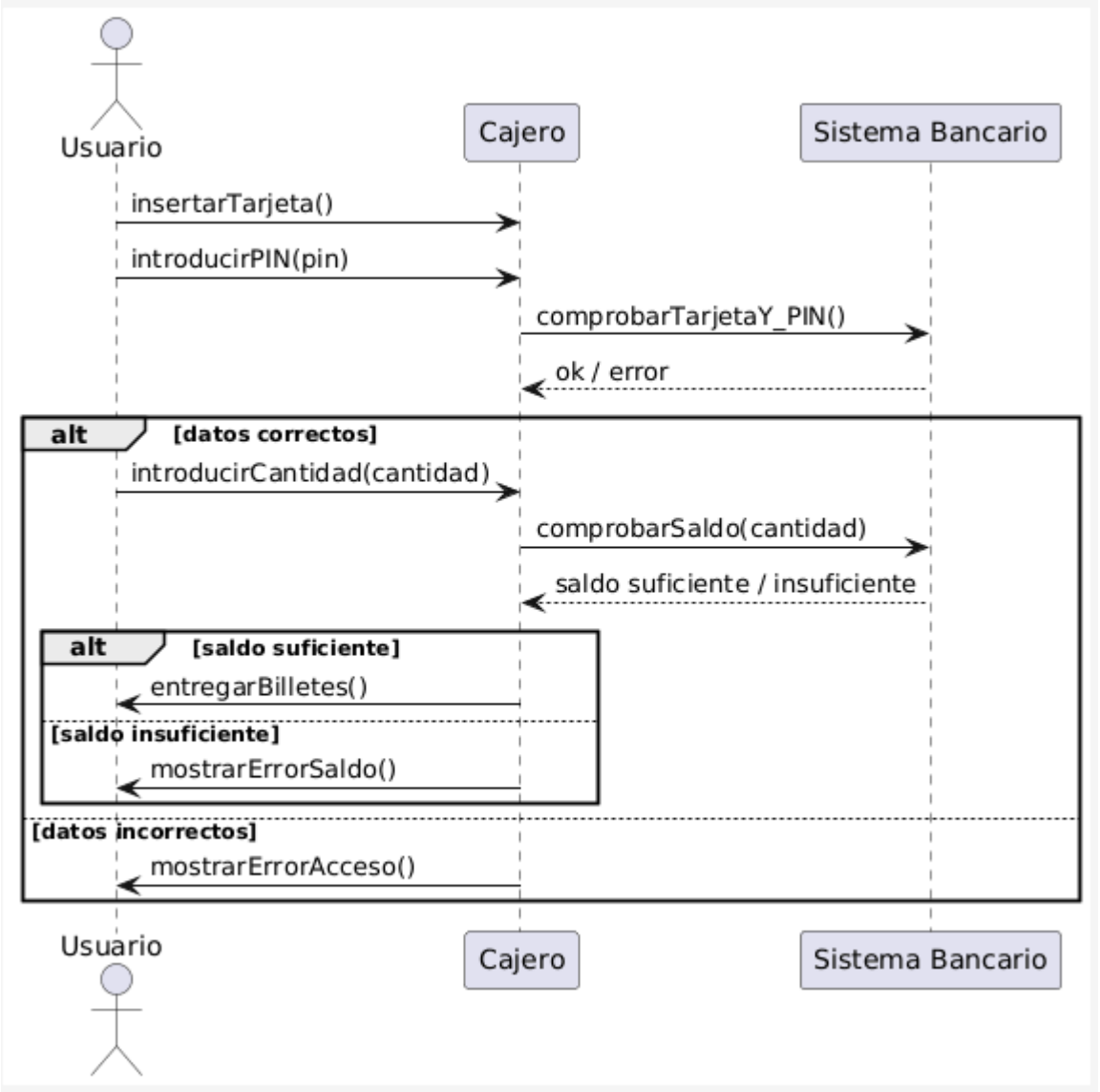
Elementos peculiares

- Uso de **múltiples escenarios en un mismo diagrama** (List a rental, Lease a NFT, Redeem a NFT).
- Aparición de **contratos inteligentes** como participantes (Rental Contract, NFT Contract).
- Mensajes numerados por escenarios (1.x, 3.x, 4.x).
- Uso de **mensajes cruzados entre contratos**, no solo entre usuario y sistema.
- Separación clara en bloques de contexto, algo no habitual en diagramas simples de secuencia.

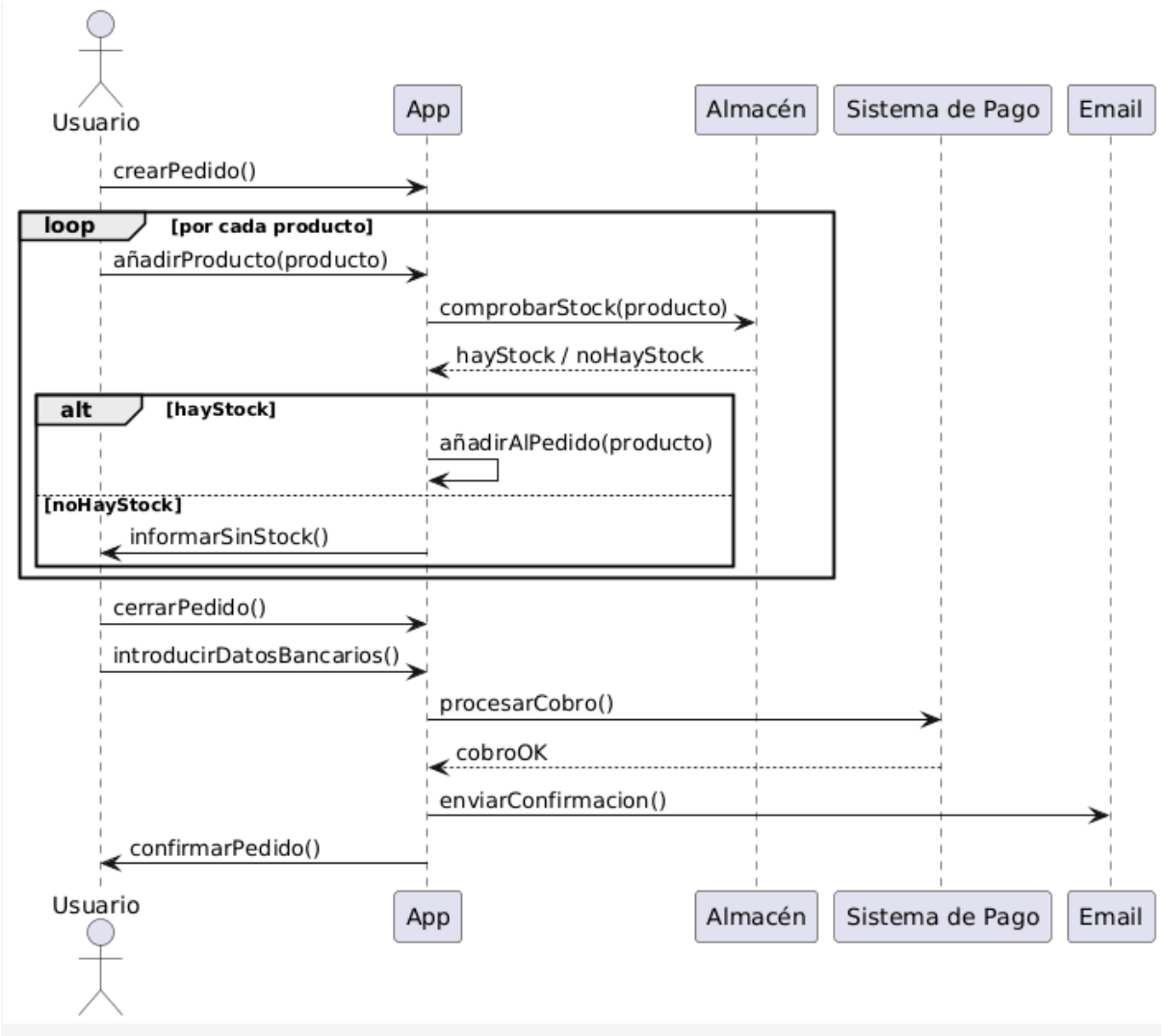
Ejercicio 6



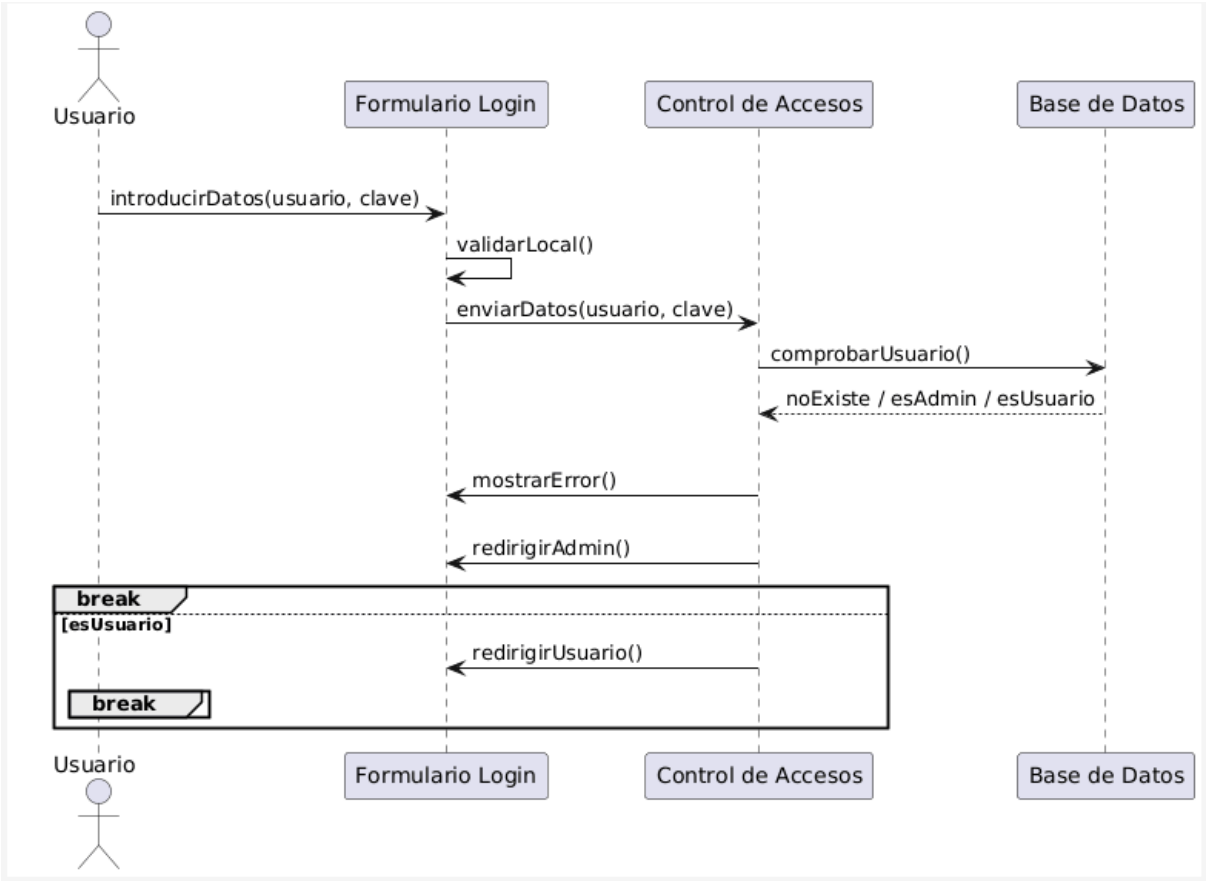
Ejercicio 7


Usuario
Cajero
Sistema Bancario
insertarTarjeta()
introducirPIN(pin)
comprobarTarjetaY_PIN()
ok / error
alt
[datos correctos]
introducirCantidad(cantidad)
comprobarSaldo(cantidad)
saldo suficiente / insuficiente
alt
[saldo suficiente]
entregarBilletes()
[saldo insuficiente]
mostrarErrorSaldo()
[datos incorrectos]
mostrarErrorAcceso()
Usuario
Cajero
Sistema Bancario

Ejercicio 8


Usuario
App
Almacén
Sistema de Pago
Email
crearPedido()
loop
[por cada producto]
añadirProducto(producto)
comprobarStock(producto)
hayStock / noHayStock
alt
[hayStock]
añadirAlPedido(producto)
[noHayStock]
informarSinStock()
cerrarPedido()
introducirDatosBancarios()
procesarCobro()
cobroOK
enviarConfirmacion()
confirmarPedido()
Usuario
App
Almacén
Sistema de Pago
Email

Ejercicio 9


Usuario
Formulario Login
Control de Accesos
Base de Datos
introducirDatos(usuario, clave)
validarLocal()
enviarDatos(usuario, clave)
comprobarUsuario()
noExiste / esAdmin / esUsuario
mostrarError()
redirigirAdmin()
break
[esUsuario]
redirigirUsuario()
break
Usuario
Formulario Login
Control de Accesos
Base de Datos

Ejercicio 10