**PONTIFICIA‌ ‌UNIVERSIDAD‌ ‌CATÓLICA‌ ‌DEL‌ ‌PERÚ**

**FACULTAD DE CIENCIAS E INGENIERÍA**

**Logotipo

Descripción generada automáticamente**

**TRABAJO GRUPAL**

Curso : [1INF50] Diseño de Software

Título : Informe de laboratorio 06

Número de grupo : 19

Integrantes :

* 20203057 - Diego Alonso La Torre Salas
* 20213732 - Frank Jiménez Valencia
* 20212471 - Ricardo Suarez Sánchez
* 20191520 - Adrian Jorge Arce Cohello
* 20220666 - Angel David Sanchez Enriquez
* 20210943 - Luis Eduardo Catalán Otero

Jefe de Práctica : Medina, M

SEMESTRE 2025-1

1. **Tabla de contenido**

[2 Introducción 4](#_heading=h.smvwyvm3p5f5)

[3 Catálogo de Requerimientos 5](#_heading=h.hssxbqwnx7om)

[3.1 Requerimientos 5](#_heading=h.zc05kyd73tgm)

[3.2 Restricciones 7](#_heading=h.o005uq4594ps)

[3.2.1 Organizacionales 7](#_heading=h.4haz7w6t2orw)

[3.2.2 Técnicas 7](#_heading=h.h1iq831uzouz)

[3.2.3 Legales o Normativas 7](#_heading=h.s7bs1uybaj2)

[4 Arquitectura de la Solución 8](#_heading=h.lhev1ijwr1im)

[4.1 Vista de Contexto 8](#_heading=h.xomy7vxho7uk)

[4.2 Vista de Contenedores 9](#_heading=h.iphdd9vyqgce)

[4.3 Vista de Componentes 9](#_heading=h.ixu850648hnk)

[4.3.1 Componentes del Contenedor <Aplicación Móvil> 10](#_heading=h.pqxlhwquoaii)

[4.3.2 Componentes del Contenedor <Aplicación Web> 11](#_heading=h.wnhf2hmma5b6)

[4.3.3 Componentes del Contenedor <Aplicación API> 14](#_heading=h.hradq3786ekr)

[4.4 Vista de Código 16](#_heading=h.8dvb6052qw22)

[4.4.1 Vista de Código del Componente: Sistema de Publicaciones 16](#_heading=h.jwougz3p31c)

[4.4.2 Vista de Código del Componente: Gestión de transacciones 17](#_heading=h.2np2mn8j84vt)

[4.4.3 Vista de Código del Componente: Autenticación de Usuario 18](#_heading=h.57ref07e6mtv)

[Vista de Código del Componente: Gestión de Usuario 18](#_heading=h.r0w9aa6h4fjb)

[4.4.4 Vista de Código del Componente: Generación de Informes 20](#_heading=h.9pp4sx63gxhr)

[4.4.5 Vista de Código del Componente: Sistema de Reseñas y Comentarios 21](#_heading=h.s5hl9zjk95t7)

[4.4.6 Vista de Código del Componente: Historial de Transacciones 22](#_heading=h.41tlkniunq2p)

[4.5 Patrones de Diseño 23](#_heading=h.3n7fnysqq7lu)

[4.6 Patrones de Persistencia 29](#_heading=h.6sedgp8ef4r3)

[4.6.1 Identity Field 29](#_heading=h.88498ux5tp1l)

[4.6.2 Foreign Key Mapping 30](#_heading=h.vzz13x9bscx6)

[4.6.3 Class Table Inheritance 31](#_heading=h.v8xbvr59e4ii)

[4.7 Patrones de Interacción con la Base de Datos 32](#_heading=h.48gga9tj47ku)

[4.7.1 Data Mapper 32](#_heading=h.cagjnfxq2rwk)

[4.7.2 Objeto Grande (LOB) Serializado 34](#_heading=h.1rokulmurx63)

[4.7.3 Dependent Mapping 35](#_heading=h.xbz2kicfhfnx)

[4.8 Diagramas de Interacción 37](#_heading=h.pij7pfywidte)

[4.8.1 Diagrama de interacción de gestión de transacciones 37](#_heading=h.w6wvb6lrgcn5)

[4.8.2 Diagrama de interacción de historial de transacciones 38](#_heading=h.v0m42lit4rd3)

[4.8.3 Diagrama de interacción de generación de informes 41](#_heading=h.3s3n6vp6ely8)

[4.8.4 Diagrama de interacción para la gestión de reseñas y comentarios 42](#_heading=h.ccude9s2lnqb)

[4.8.5 Diagrama de interacción de registro de pedidos 44](#_heading=h.tnplldwvi9xm)

[4.8.6 Diagrama de interacción de Gestión de Usuarios 46](#_heading=h.wgtl51up8zza)

[5 Plan de Pruebas 48](#_heading=h.kxmzdadvcjiu)

[5.1 Contexto de las Pruebas 48](#_heading=h.vqb6d71qmqoz)

[5.1.1 Uso del Sistema 48](#_heading=h.vr2yv4id3lb9)

[5.1.2 Elementos a Probar 48](#_heading=h.ptxvjuj36ns0)

[5.1.3 Alcance de las Pruebas 48](#_heading=h.v5ni3i4qb4ed)

[5.1.4 Asunciones 49](#_heading=h.qe0gqmjvlfz3)

[5.1.5 Restricciones 49](#_heading=h.f0k3fkij25x3)

[5.2 Enfoque de Pruebas 50](#_heading=h.49bsprtyvn9r)

[5.3 Comunicación del Plan 50](#_heading=h.9e2vxbbxw97m)

[5.4 Tipos de Pruebas 51](#_heading=h.k7sc1asefpgy)

[5.5 Riesgos en el Desarrollo de Pruebas 52](#_heading=h.e0lzvrxmz06z)

[5.6 Cronograma 53](#_heading=h.dsh0oeax8bb7)

[5.7 Casos de Prueba 55](#_heading=h.4jej44npp0lo)

[5.7.1 Pantalla de Gestión de Transacciones por Pagar 55](#_heading=h.ox1z7kuv8wf)

[5.7.2 Pantalla de Confirmación de Pedido 57](#_heading=h.sfkgg5xusgt2)

[5.7.3 Pantalla de Agregar Método de Pago 60](#_heading=h.o2xjzivgncdi)

[5.7.4 Pantalla de Gestión de Historial de Transacciones 63](#_heading=h.ne4j3e5588mq)

[5.7.5 Pantalla de Registrar reseñas por Producto : 66](#_heading=h.pfa3k9nt9q1x)

[5.7.6 Pantalla de Nueva Publicación 69](#_heading=h.dnnfm4rf3y6v)

[5.7.7 Pantalla Registro de Usuarios 72](#_heading=h.m3y6u8qqidt1)

# Introducción

Squirlearn es una empresa que ha identificado un nicho de negocio en la intermediación comercial de materiales educativos para estudiantes universitarios, con un enfoque inicial en los miembros de la Pontificia Universidad Católica del Perú. Este enfoque se basa en la necesidad de facilitar el acceso a recursos académicos de calidad para mejorar el aprendizaje y desempeño de los estudiantes, promoviendo la creación y distribución de contenido especializado.

La problemática que enfrenta Squirlearn radica en la falta de plataformas centralizadas que permitan a los estudiantes encontrar, publicar y adquirir recursos educativos de manera eficiente. Si bien existen otras plataformas de intercambio de conocimiento, como las comunidades en línea y los repositorios de documentos, estas a menudo carecen de organización, seguridad y escalabilidad, lo que dificulta tanto a los creadores como a los usuarios finales en sus procesos de búsqueda y adquisición.

Como se mencionó anteriormente, Squirlearn será una aplicación que será utilizado por los miembros de la comunidad educativa, por lo que para que sea más rápido de usar, para los compradores y vendedores será una aplicación móvil para que puedan realizar sus compras más fácilmente. Por parte del administrador, este será una aplicación web para que pueda gestionar el sistema y generar reportes.

El objetivo de Squirlearn es crear un espacio seguro y práctico donde los estudiantes puedan generar ingresos adicionales a través de la comercialización de sus materiales académicos, mientras complementan su proceso de aprendizaje. Además, se busca asegurar la integridad de las transacciones realizadas durante el proceso de adquisición de material, promoviendo una experiencia fluida y confiable para todos los usuarios.

# Catálogo de Requerimientos

## Requerimientos

| ID | Descripción | Tipo (Funcional / Calidad) | Prioridad (Alta / Media / Baja) |
| --- | --- | --- | --- |
| R1 | El sistema debe mostrar los productos y sus detalles (nombre, precio, descripción, tipo, imágenes). | Funcional | Alta |
| R2 | El sistema debe permitir al comprador realizar la búsqueda de productos disponibles por el nombre. | Funcional | Alta |
| R3 | El sistema debe permitir la comunicación entre comprador y vendedor mediante un chat de la aplicación. | Funcional | Alta |
| R4 | El sistema debe permitir a los compradores acceder a un historial detallado de sus compras y alquileres. | Funcional | Alta |
| R5 | Garantizar que la plataforma esté disponible de forma continua para los usuarios, operando los 7 días de la semana y por lo menos 23 horas del día, para brindar una experiencia de uso constante y confiable. | Calidad | Alta |
| R6 | El sistema debe permitir a los usuarios dejar reseñas y asignar una valoración al producto. | Funcional | Media |
| R7 | El sistema debe mostrar un diferenciador entre productos en venta y productos en alquiler. | Funcional | Media |
| R8 | El sistema debe calcular y mostrar la valoración del vendedor, como el promedio de las valoraciones de sus productos. | Funcional | Media |
| R9 | El sistema deberá ser capaz de soportar usuarios en simultáneo y ofrecer tiempos de respuesta rápidos para garantizar una experiencia fluida incluso en condiciones de alta demanda. | Calidad | Media |
| R10 | El sistema debe permitir al administrador suspender cuentas de usuarios. | Funcional | Baja |
| R11 | El sistema debe permitir al administrador visualizar el historial de transacciones de usuarios. | Funcional | Baja |
| R12 | El sistema debe permitir al administrador responder a los reportes. | Funcional | Baja |
| R13 | El sistema debe permitir a los usuarios ingresar empleando credenciales personales | Funcional | Alta |
| R14 | El sistema debe validar que los usuarios registrados pertenezcan a un centro académico | Funcional | Alta |
| R15 | El sistema debe permitir a los usuarios crear, editar y gestionar publicaciones | Funcional | Alta |
| R16 | El sistema debe permitir que un administrador pueda aprobar o rechazar publicaciones | Funcional | Alta |
| R17 | El sistema debe permitir que los usuarios puedan realizar pagos empleando diferentes métodos. | Funcional | Alta |
| R18 | El sistema debe permitir que el administrador pueda generar reportes de las transacciones realizadas | Funcional | Alta |
| R19 | El sistema debe proteger los datos de los usuarios mediante encriptación, autenticación multifactor (MFA) y control de acceso basado en roles (RBAC) para asegurar la confidencialidad, integridad y disponibilidad de la información. | Calidad | Alta |
| R20 | El sistema debe ofrecer una interfaz intuitiva y fácil de usar, asegurando que los usuarios puedan navegar y completar tareas sin dificultad, con tiempos de aprendizaje mínimos. | Calidad | Media |
| R21 | El sistema debe calcular y mostrar la información sin errores, garantizando que los datos presentados a los usuarios sean precisos y consistentes en todos los módulos. | Calidad | Media |

## 

## Restricciones

### Organizacionales

* Capacidades y Conocimientos del Equipo : El equipo de desarrollo debe trabajar dentro de sus especialidades y conocimientos.
* Guías y Políticas Internas : El desarrollo debe seguir las políticas de la empresa, como el uso de herramientas específicas.
* Recursos y Presupuesto : El proyecto debe ajustarse a un presupuesto y recursos disponibles.

### Técnicas

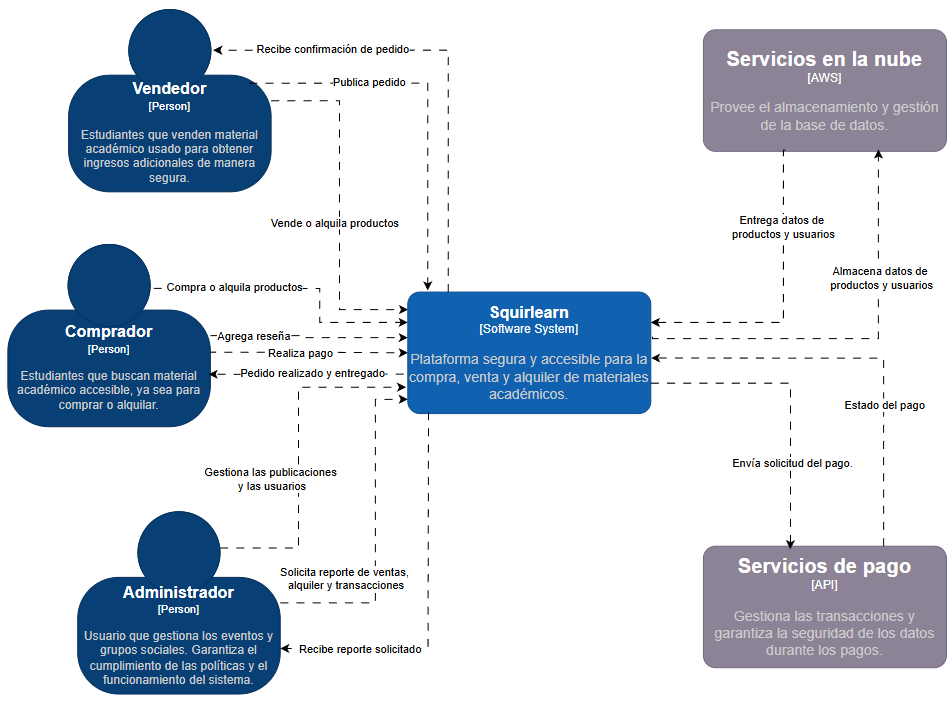
* Plataforma y Entorno: El sistema debe ejecutarse en una infraestructura como AWS debido a las siguientes razones técnicas:
* **Escalabilidad y Elasticidad:** AWS permite escalar recursos automáticamente según la demanda, garantizando que la plataforma pueda manejar tanto cargas bajas como picos de tráfico sin interrupciones.
* **Alta Disponibilidad y Redundancia:** Con múltiples zonas de disponibilidad y opciones de recuperación ante desastres, AWS asegura que el sistema esté siempre disponible, incluso ante fallos de infraestructura.
* Integración de Librerías y Frameworks: Solo se pueden utilizar ciertas librerías o frameworks previamente aprobados

### Legales o Normativas

* Protección de Datos: Cumplimiento con regulaciones como GDPR o CCPA para la protección de datos personales.
* Propiedad Intelectual y Licencias : El uso de software de terceros debe cumplir con las licencias y derechos de propiedad intelectual.

# Arquitectura de la Solución

## Vista de Contexto

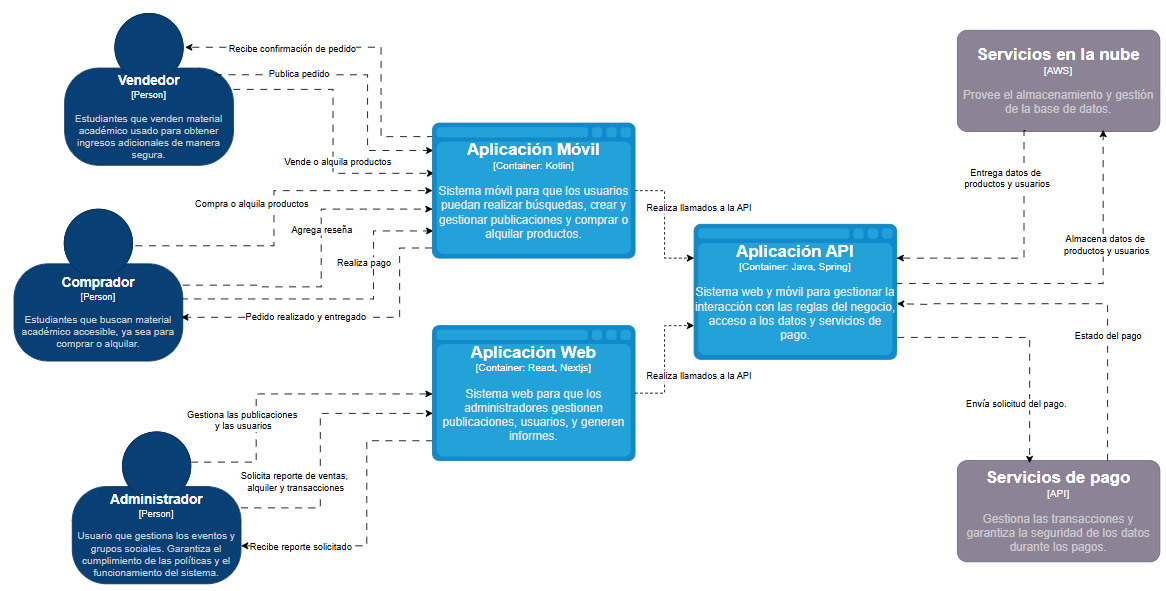


*Imagen 1: Vista de Contexto de Squirlearn en el modelo C4.*

| Actor | Descripción |
| --- | --- |
| Vendedor | Usuarios que venden material académico usado. |
| Comprador | Usuarios que buscan material académico para comprar. |
| Administrador | Usuario responsable de gestionar las publicaciones y garantizar el cumplimiento de las políticas de la plataforma. |

| Sistema Externo | Descripción |
| --- | --- |
| Servidores en la nube (AWS) | Proporciona almacenamiento, gestión de bases de datos y soporte de caché. |
| Servicio de pago(Mercado Pago) | Se encarga de procesar transacciones y garantizar la seguridad de los datos durante los pagos. |

## Vista de Contenedores



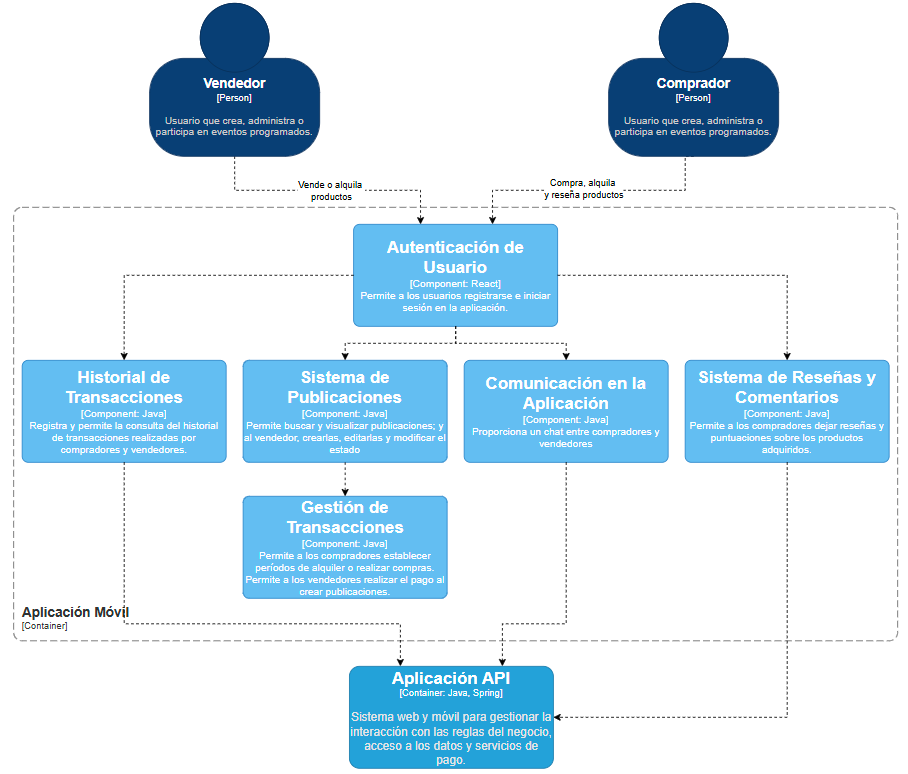
*Imagen 2: Vista de Contenedores de Squirlearn en el modelo C4.*

| Contenedor | Descripción |
| --- | --- |
| Aplicación Móvil (Container : Frontend) | Sistema móvil para que los vendedores y compradores accedan desde sus dispositivos.  Permite realizar búsquedas, crear y gestionar publicaciones, comunicarse y efectuar pagos. |
| Aplicación Web (Container: Java y Spring MVC) | Sistema web para que los administradores gestionen publicaciones, usuarios y generen informes. |
| Aplicación API (Container: Java, Spring Boot) | Provee APIs vía JSON/HTTPS para gestionar la interacción entre la aplicación móvil, la aplicación web y la base de datos. También permite la integración con los servicios de pago. |

## 

## Vista de Componentes

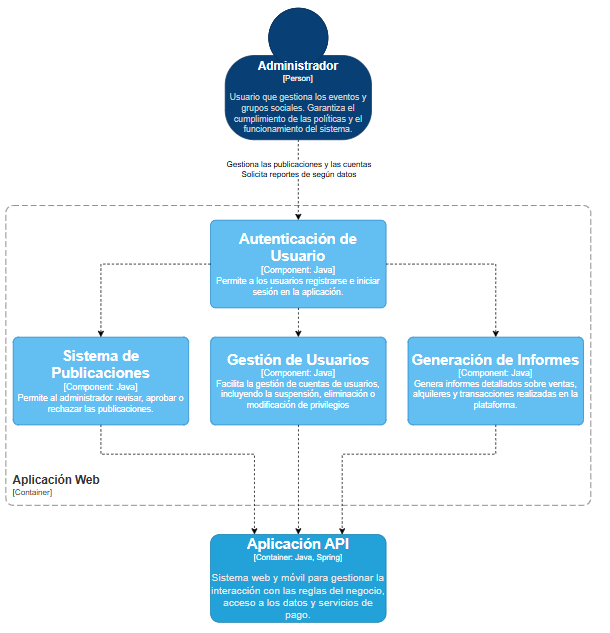
### Componentes del Contenedor <Aplicación Móvil>



*Imagen 3: Vista del Componente de Aplicación Móvil de Squirlearn en el modelo C4.*

| Componente | Descripción | Roles | Req. Funcionales | Req. Calidad |
| --- | --- | --- | --- | --- |
| Autenticación de Usuario | Permite a los usuarios registrarse e iniciar sesión en la aplicación. | Vendedor, Comprador y Administrador. | R13 y R14 | R19 |
| Sistema de Publicaciones | Permite buscar y visualizar publicaciones; y al vendedor crearlas. para su próxima revisión. | Comprador, Vendedor | R1, R2, R6, R7, R15 y R16 | R20 y R21 |
| Historial de transacciones | Registra y permite la consulta del historial de transacciones. | Comprador, Vendedor | R4 y R11 | R19 |
| Comunicación en la Aplicación | Proporciona un chat entre compradores y vendedores. | Comprador Vendedor | R3 | R20 y R5 |
| Sistema de Reseñas y Comentarios | Permite a los compradores dejar reseñas y puntuaciones sobre los productos adquiridos. | Comprador | R6 y R8 | R20 |
| Gestión de transacciones | Permite a los compradores establecer períodos de alquiler o realizar compras, y a los vendedores realizar el pago al crear publicaciones. Incluye las normas de alquiler y el proceso de pago. | Comprador, Vendedor | R1 y R17 | R20 y R21 |

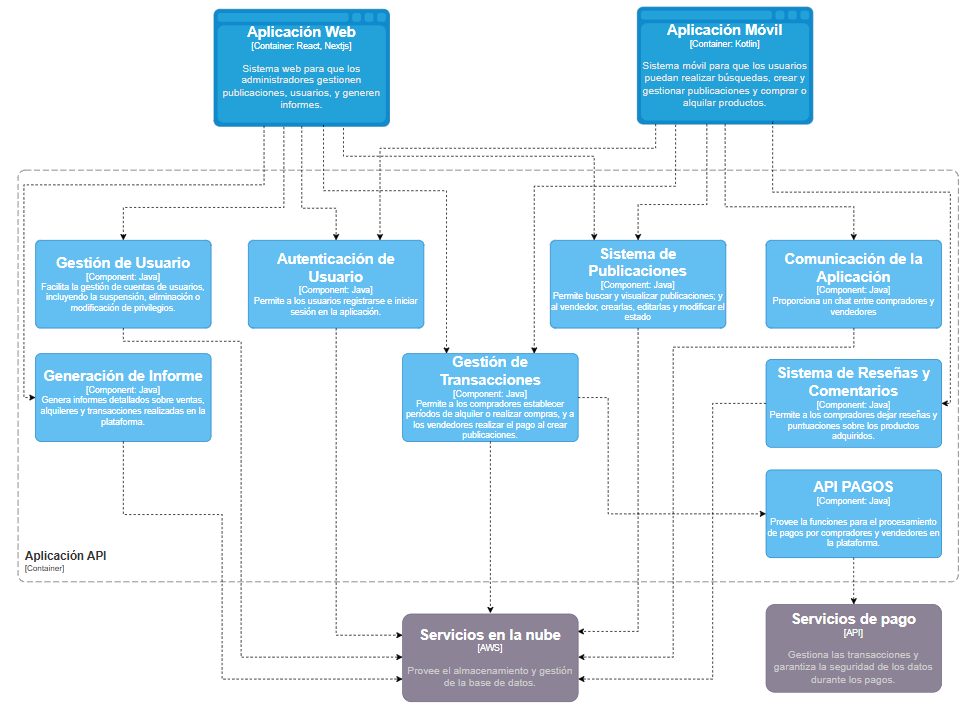
### Componentes del Contenedor <Aplicación Web>



*Imagen 4: Vista del Componente de Aplicación Web de Squirlearn en el modelo C4.*

| Componente | Descripción | Roles | Req. Funcionales | Req. Calidad |
| --- | --- | --- | --- | --- |
| Autenticación de Usuario | Permite a los usuarios registrarse e iniciar sesión en la aplicación. | Administrador, Vendedor, Comprador | R13 y R14 | R19 |
| Sistema de Publicaciones | Permite al administrador gestionar las publicaciones y modificar el estado de la publicación. | Administrador | R1, R2, R6, R7, R15 y R16 | R20 y R21 |
| Gestión de usuarios | Realiza una gestión de cuentas de usuarios, incluyendo la suspensión, eliminación o modificación de privilegios. | Administrador | R10 | R19 |
| Generación de informes | Genera informes detallados sobre ventas, alquileres y transacciones realizadas en la plataforma. | Administrador | R18 | R21 |

### Componentes del Contenedor <Aplicación API>

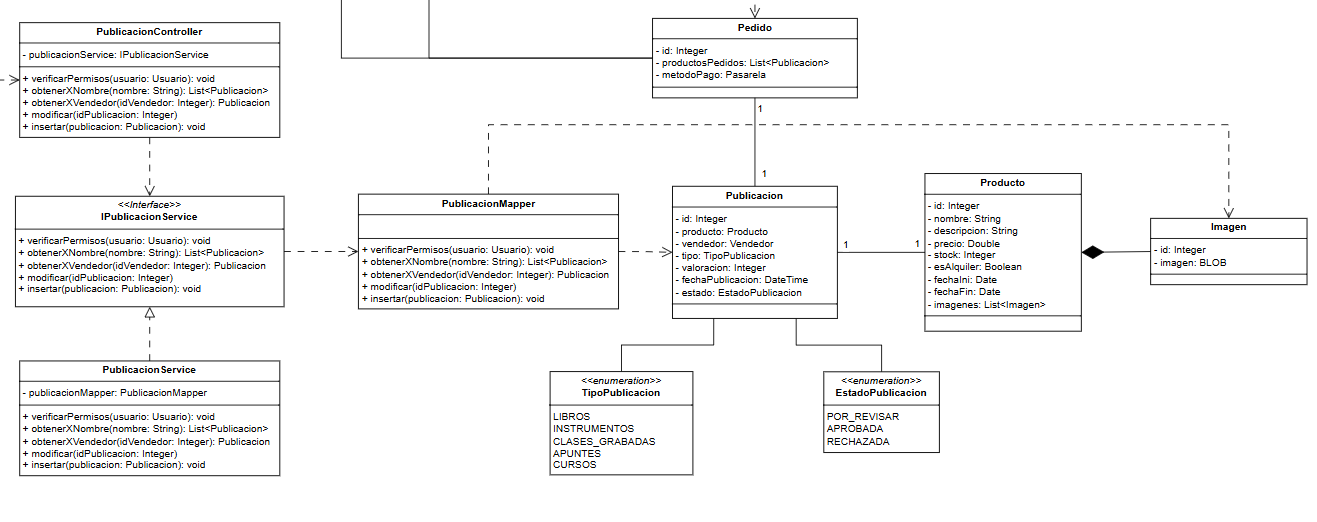
**

*Imagen 4: Vista del Componente de Aplicación API de Squirlearn en el modelo C4.*

| Componente | Descripción | Roles | Req. Funcionales | Req. Calidad |
| --- | --- | --- | --- | --- |
| Gestión de usuarios | Realiza una gestión de cuentas de usuarios, incluyendo la suspensión, eliminación o modificación de privilegios. | Administrador | R10 | R19 |
| Generación de informes | Genera informes detallados sobre ventas, alquileres y transacciones realizadas en la plataforma. | Administrador | R18 | R21 |
| Autenticación de Usuario | Permite a los usuarios registrarse e iniciar sesión en la aplicación. | Vendedor, Comprador, Administrador | R13 y R14 | R19 |
| Sistema de Publicaciones | Permite buscar y visualizar publicaciones; y al vendedor y administrador, crearlas, editarlas y modificar el estado | Comprador Vendedor Administrador | R1, R2, R6, R7, R15 y R16 | R20 y R21 |
| Sistema de Reseñas y Comentarios | Permite a los compradores dejar reseñas y puntuaciones sobre los productos adquiridos. | Comprador | R6 y R8 | R20 |
| Comunicación en la Aplicación | Proporciona un chat entre compradores y vendedores | Comprador Vendedor | R3 | R20 y R5 |
| Gestión de transacciones | Permite a los compradores establecer períodos de alquiler o realizar compras, y a los vendedores realizar el pago al crear publicaciones. Incluye las normas de alquiler y el proceso de pago. | Compradores Vendedores | R1 y R17 | R20 y R21 |

## Vista de Código

### Vista de Código del Componente: Sistema de Publicaciones

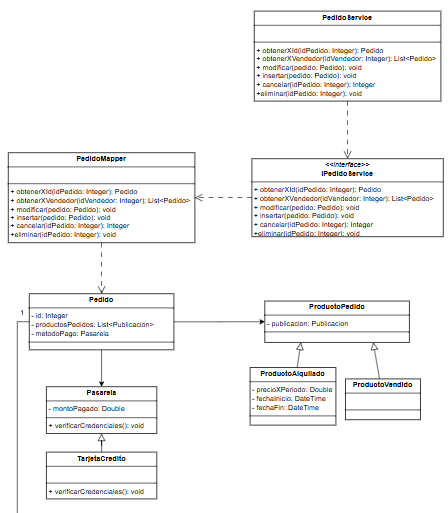


*Imagen 5: Vista de Código del Componente de Sistema de Publicaciones*

**Sustento de decisiones de diseño:**

* **Persistencia de datos:** Se ha implementado una clase PublicaciónMapper que se encarga de almacenar y administrar la información en la base de datos del sistema.
* **Lógica de negocio:** La clase PublicacionService para la lógica de negocio utilizada en la gestión de las publicaciones. Se cuenta con un producto, este producto cuenta con una lista de imágenes almacenadas en un gestor de imágenes. La publicación es una clase que envuelve la clase producto, pero define más parámetros importantes para su gestión. La publicación tiene estados (aceptada, rechazada, en proceso) y el tipo de publicación que especifica a qué sector va dirigido.
* **Comunicación externa:** Para la comunicación externa a través de solicitudes HTTP es necesario la clase de PublicaciónController.
* **Gestión de publicaciones:** El producto cuenta con un atributo de “esAlquiler” de tipo boolean para diferenciar entre un producto que se ofrece para alquiler o para venta. Una publicación cuenta con un producto y un vendedor, es la publicación la que es gestionada y es el producto el que tiene el atributo.

### Vista de Código del Componente: Gestión de transacciones

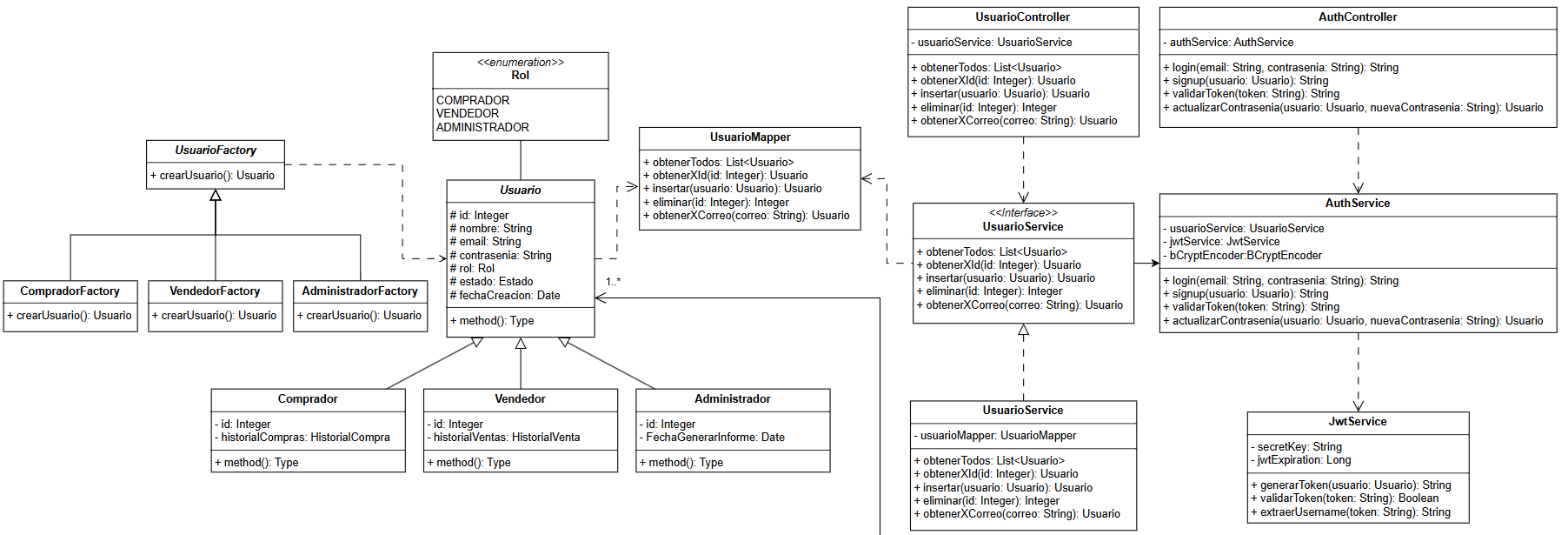


*Imagen 6: Vista de Código del Componente de Gestión de transacciones*

**Sustento de decisiones de diseño:**

* **Lógica de negocio:** La interfaz PedidoService se utiliza para asegurar una comunicación entre componentes que no sea dependiente de la estructura interna de este (encapsulamiento). Se encargan de las lógica de negocio.
* **Persistencia de datos:** Se ha implementado la clase PedidoMapper para almacenar la información en la base de datos. Es el encargado de generar las consultas SQL.
* **Abstracción de clases:** La subclase TarjetaCredito es accedida desde una clase abstracta Pasarela, con el objetivo de hacer espacio para otros posibles métodos de pago.
* **Definición de pedidos:** La clase Pedido guarda varios objetos de ProductoPedido, que representan las condiciones adicionales alrededor de la compra o el alquiler de un producto.

### Vista de Código del Componente: Autenticación de Usuario



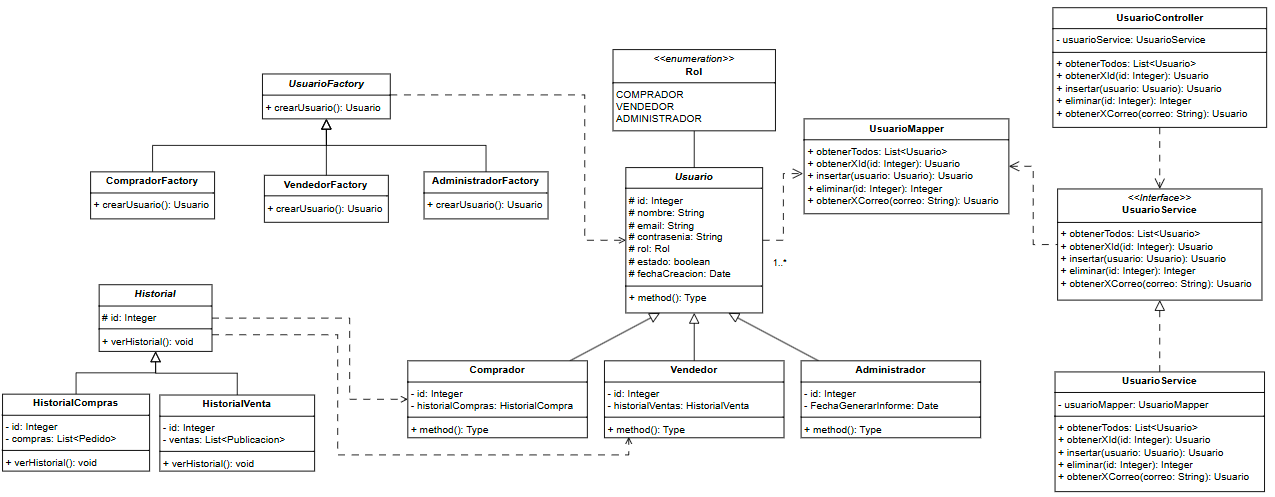
*Imagen 7: Vista de Código del Componente de Autenticación de Usuario*

**Sustento de decisiones de diseño:**

* **Lógica de negocio:** La interfaz UsuarioService, AuthService y JwtService se utiliza para asegurar una comunicación entre componentes que no sea dependiente de la estructura interna de este (encapsulamiento). Se generó una clase servicio para delegar la tarea de generación y validación de tokens a una clase específica. Se encarga de la lógica de negocio.
* **Declaración de usuarios:** Las herencia en las clases Comprador, Vendedor y Administrador permiten que cada rol tenga un tipo específico de comportamientos al realizar la validación.
* **Validación y gestión de sesión de usuarios:** La clase AuthService delega y encapsula la lógica de validación del código de usuario a la clase PucpValidator. Esta última clase mantiene una única responsabilidad.
* **Comunicación con las vistas:** La clase AuthController actuará de intermediario entre la clase AuthView y la clase AuthService. De este modo se puede gestionar las peticiones entre la vista y la lógica del sistema.

### 

### Vista de Código del Componente: Gestión de Usuario

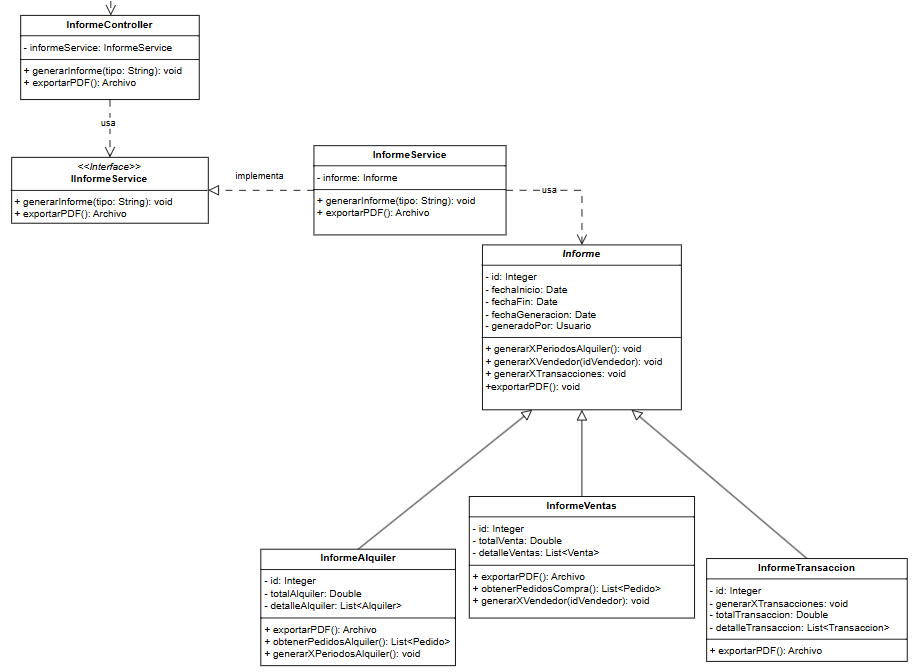


*Imagen 8: Vista de Código del Componente de Gestión de Usuario*

**Sustento de decisiones de diseño:**

* **Manejo de usuarios:** Se ha decidido aplicar herencia a la clase abstracta Usuario para que haya una definición clara para las clases hijas Comprador, Vendedor y Administrador. Esto con la finalidad de separar sus funcionalidades.
* **Persistencia de datos:** Utilizamos Mapper para la comunicación entre la entidad y la base de datos.
* **Lógica del negocio:** La clase UsuarioService se encarga de la lógica de negocio necesaria para los métodos.

### Vista de Código del Componente: Generación de Informes

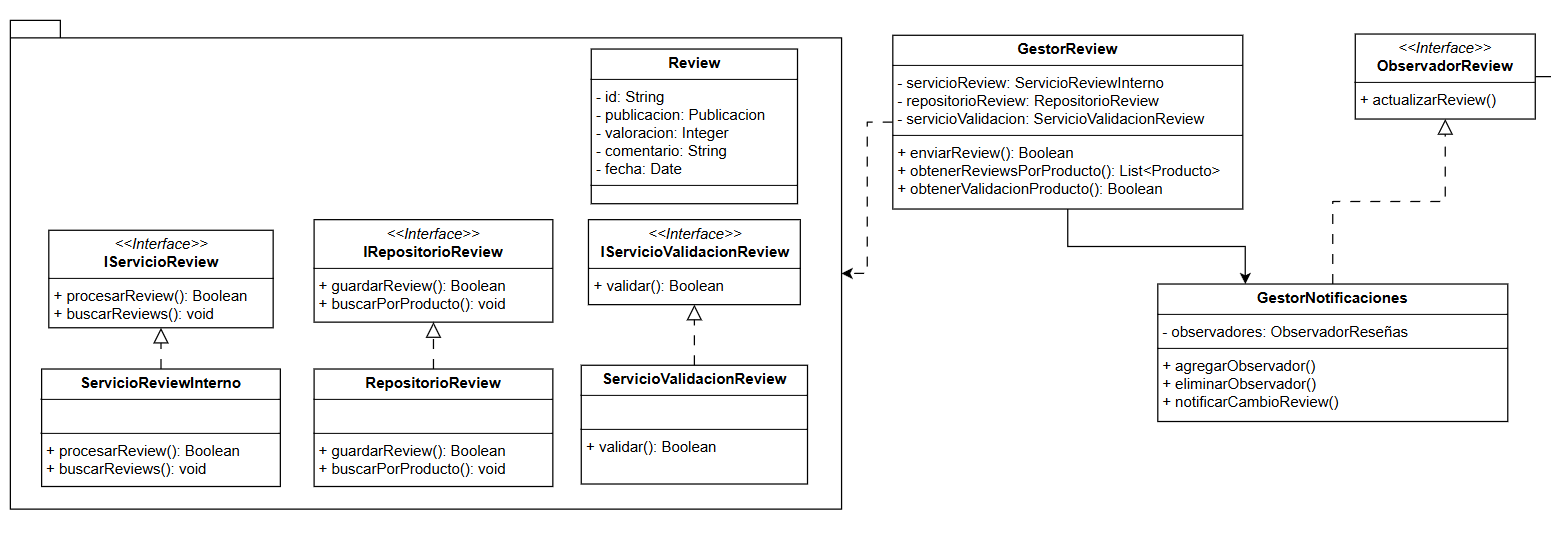


*Imagen 9: Vista de Código del Componente de Generación de Informes*

**Sustento de decisiones de diseño:**

* **Persistencia de datos:** Debido a que los informes se generan cada vez que se solicita, no son almacenados en la base de datos del sistema. Sin embargo, el usuario puede descargarlo como PDF una vez se haya generado.
* **Lógica de negocio:** El encargado de la lógica de negocio para esta generación es la clase de InformeService.
* **Generación de informes:** Para la generación de informes, se ha utilizado el patrón Template Method para definir un esqueleto y que cada clase pueda ejecutar una secuencia de métodos según el tipo de informe.
* Método para generar informe: Se ha decidido utilizar un método unificado por parte del controlador y la vista para que se envíe el tipo de informe a generar, luego la clase InformeService lo divide según el tipo de informe y luego es generado según el tipo pasado como parámetro.

### Vista de Código del Componente: Sistema de Reseñas y Comentarios

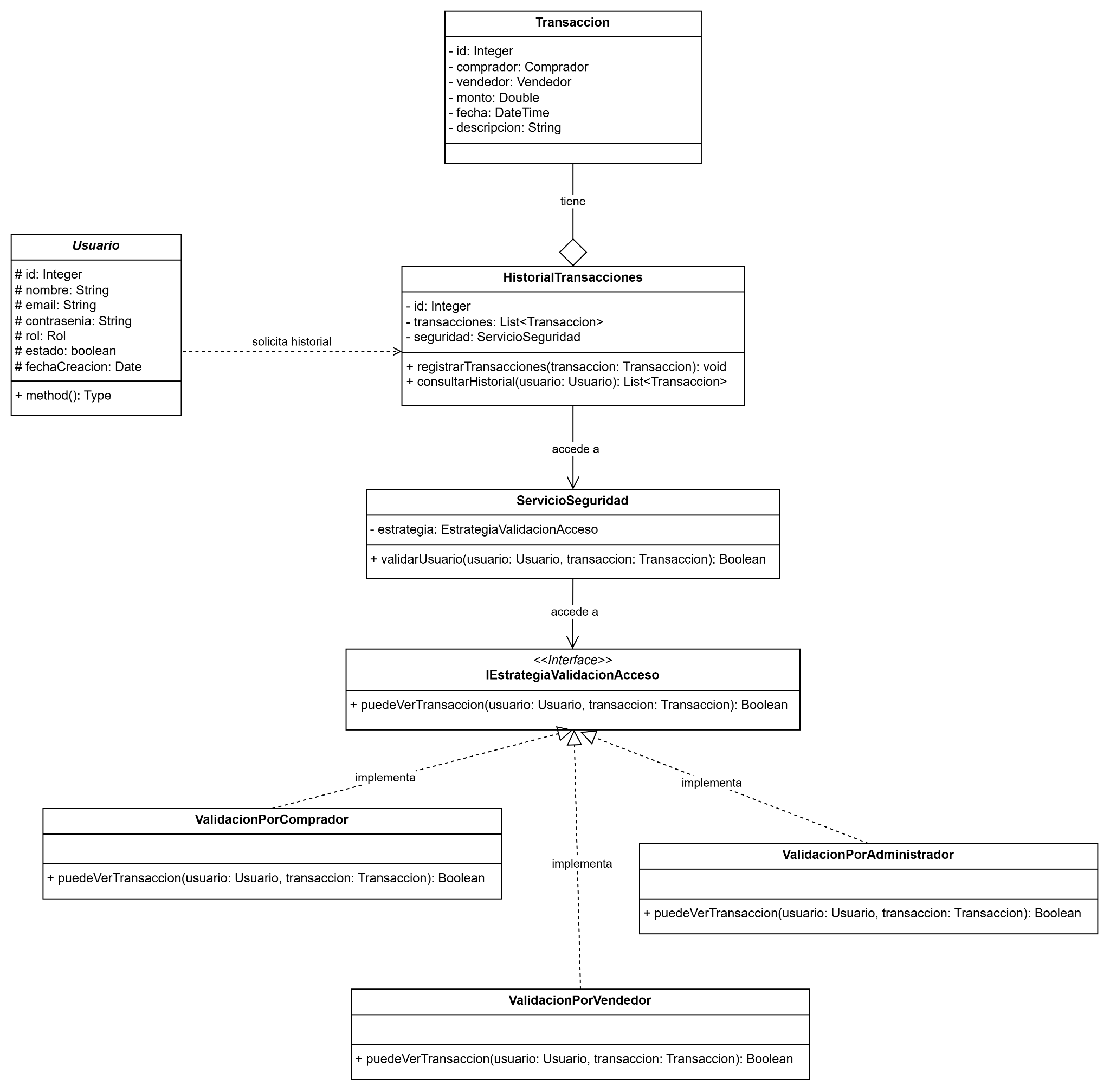


*Imagen 10: Vista de Código del Componente de Sistema de Reseñas y Comentarios*

**Sustento de decisiones de diseño:**

* **Simplificación del acceso a funcionalidades complejas:** La clase GestorReview actúa como fachada, permitiendo al cliente interactuar con métodos como enviarReview(), obtenerReviewsPorProducto() y obtenerValoracionProducto(). Así, el cliente no necesita conocer cómo se validan, almacenan o procesan las reseñas internamente, lo que simplifica el uso del sistema y mejora la legibilidad del código.
* **Desacoplamiento de responsabilidades :** Al combinar los patrones Facade y Observer, se separan las responsabilidades entre los componentes. GestorReview se encarga de manejar la lógica relacionada con las reseñas, mientras que GestorNotificaciones administra la lista de vendedores registrados y les envía notificaciones cuando ocurre un cambio. De esta forma, cada clase se concentra en una tarea específica, lo que facilita la organización y el mantenimiento.
* **Flexibilidad y extensibilidad:** Los patrones implementados permiten modificar o reemplazar componentes sin afectar otras partes del sistema. Las interfaces IServicioReview, IRepositorioReview e IServicioValidacion permiten intercambiar las implementaciones de validación, almacenamiento o lógica sin tocar la fachada GestorReview. También, gracias a la interfaz ObservadorReview, los vendedores pueden recibir notificaciones sin estar directamente acoplados al resto del sistema, lo que facilita ajustes futuros en la forma de notificar sin afectar el núcleo de la aplicación.
* **Modularidad:** GestorReview actúa como sujeto, pero delega la notificación a GestorNotificaciones, que gestiona una lista de observadores (vendedores que implementan ObservadorReview) y los actualiza mediante notificarCambioReview(). Esta estructura modular permite una comunicación desacoplada y un sistema organizado.

### Vista de Código del Componente: Historial de Transacciones



*Imagen 11: Vista de Código del Componente de Historial de Transacciones*

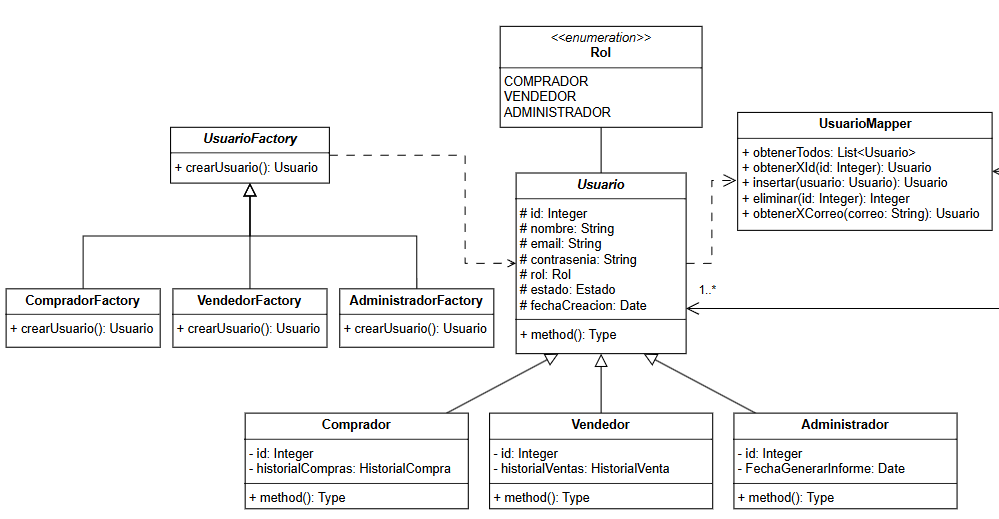
**Sustento de decisiones de diseño:**

* **Implementación de diversas estrategias de validación:** El presente diseño adopta el patrón de diseño Strategy con el propósito de delegar la lógica de validación a distintas implementaciones. La clase ServicioSeguridad no centraliza el proceso de validación, sino que lo delega a clases concretas como ValidacionPorComprador,ValidacionPorVendedor y ValidacionPorAdministrador, las cuales implementan la interfaz EstrategiaValidacionAcceso. Esta interfaz define el método puedeVerTransaccion, permitiendo encapsular las reglas de acceso según el rol del usuario. Esta arquitectura no solo mejora la cohesión del sistema, sino que también facilita la incorporación futura de nuevos perfiles o roles sin afectar la estructura existente.
* **Separación de responsabilidades en la consulta de historiales:** Con el fin de promover la modularidad y el principio de única responsabilidad, se establece una separación entre la generación de reportes y el control de acceso a los registros. La clase HistorialTransacciones se encarga de la gestión de transacciones asociadas a un usuario, permitiendo su registro, eliminación y visualización. Dicha clase mantiene una lista de objetos Transaccion y, a través de su atributo seguridad, establece una relación con una instancia de ServicioSeguridad. Este último se encarga de aplicar una estrategia específica de validación de acuerdo con el rol del usuario, asegurando así un control de acceso flexible y extensible.

### Patrones de Diseño

#### Factory Method

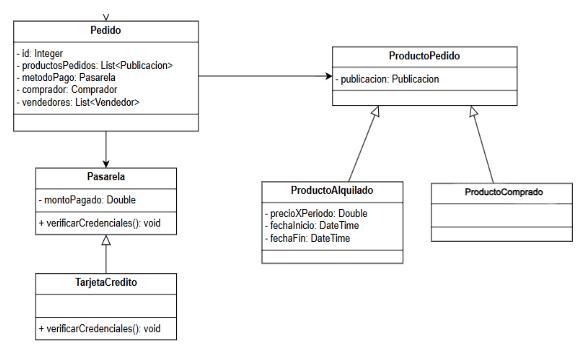
#### El Factory Method se utiliza para la creación de usuarios en este sistema porque permite una gestión más flexible y escalable de los distintos tipos de usuarios (Comprador, Vendedor, Administrador). Al centralizar la lógica de creación en fábricas especializadas (CompradorFactory, VendedorFactory, AdministradorFactory), se reduce el acoplamiento entre las clases y se facilita el mantenimiento, permitiendo agregar nuevos tipos de usuarios sin modificar el código existente. Esto sigue el principio de responsabilidad única (SRP) y promueve una arquitectura más limpia y organizada, adecuada para aplicaciones en crecimiento.



*Imagen: Patrón Factory Method para Gestión de Usuarios.*

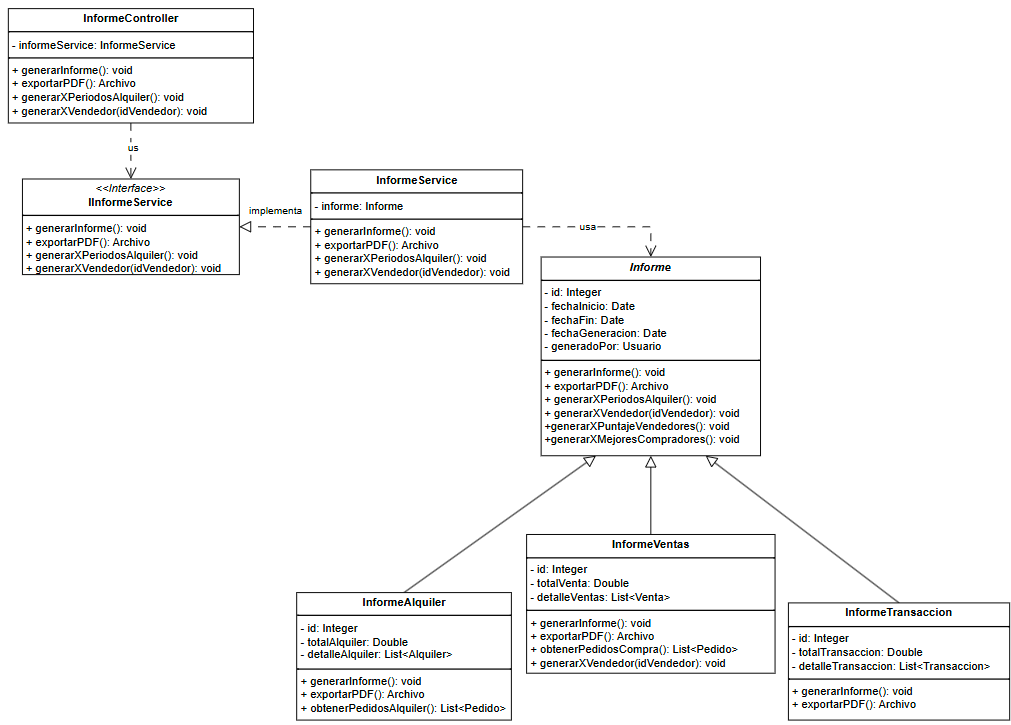
#### Template Method

#### Se aplicó el patrón Template Method para estructurar de forma controlada y reutilizable el proceso de generación de pedidos. En el caso de ProductoPedido, esta clase abstracta define una estructura base para productos dentro de un pedido (mediante su relación con Stock), estas estructuras, luego podrían ser usadas para manejar un producto dentro de un pedido. Por otro lado, ProductoAlquilado y ProductoComprado implementan detalles concretos, como pueden ser fechas de alquiler o duración, precio, etc. En Pasarela, se define el método abstracto verificarCredenciales(), que debe ser implementado por sus subclases como TarjetaCredito, permitiendo distintas formas de autenticación según el medio de pago, sin alterar la lógica general del proceso de pago.



*Imagen: Patrón Template Method para Pedidos*

De la misma manera, se aplicó el patrón **Template Method** para estructurar de forma controlada y reutilizable el proceso de generación de informes. La clase Informe actúa como plantilla, definiendo el flujo general con métodos como generarInforme(), exportarPDF() y otros relacionados a la generación por vendedor o por usuario. Este enfoque permite que las subclases (InformeAlquiler, InformeVentas e InformeTransaccion) implementen únicamente los pasos específicos que varían según el tipo de informe, como los detalles de alquileres, ventas o transacciones.

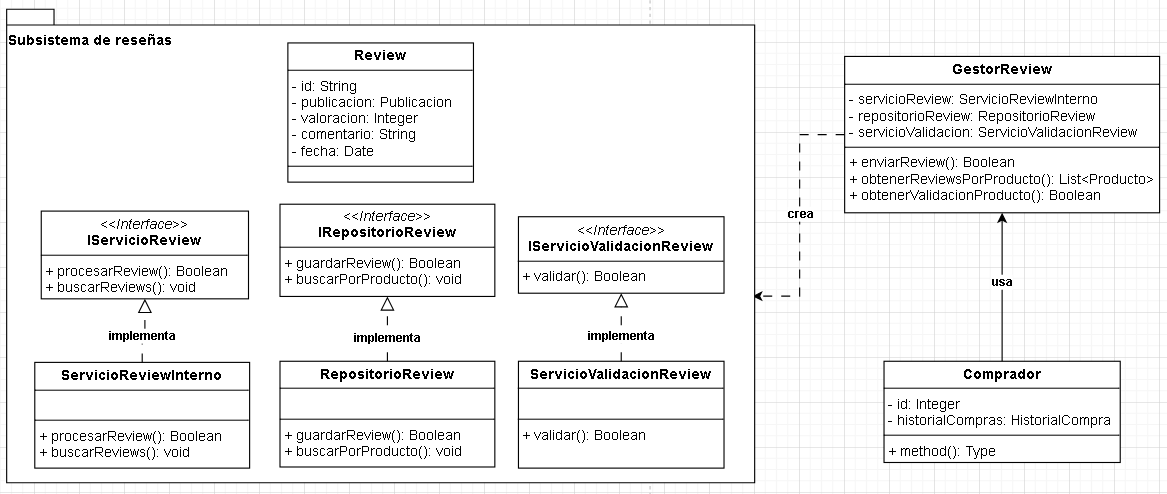
*Imagen: Patrón Template Method para Generación de Informes*

Gracias a esto, se **evita la duplicación de código**, se **mantiene la coherencia en la estructura de los informes** y se facilita la **extensión del sistema** ante nuevos requerimientos. Si en el futuro se desea agregar otro tipo de informe, basta con crear una nueva subclase que implemente los pasos particulares, sin alterar la lógica base ya establecida en Informe. Esto respeta principios de diseño como el de **abierto/cerrado**, promoviendo un sistema más mantenible y escalable.

#### Facade

#### El patrón Facade cumple esta característica al implementar la clase fachada con la finalidad de centralizar y reducir la complejidad en el uso de un subsistema.. En esta clase se encapsula la lógica de un subsistema de reseñas mediante la clase GestorReview, la cual será invocada por Comprador.

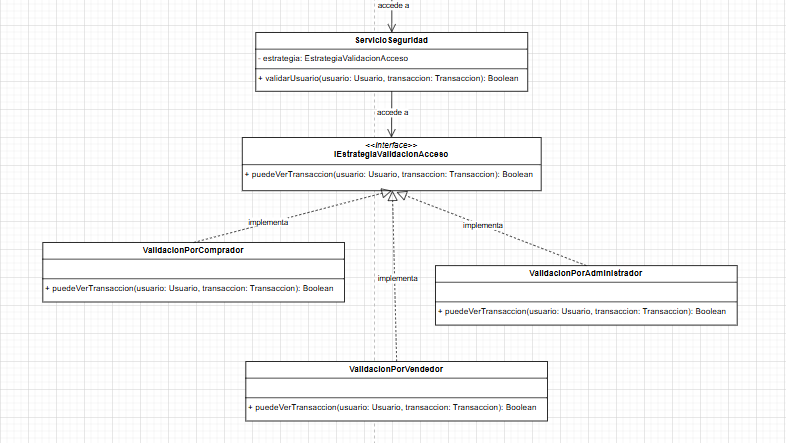
#### 



*Imagen: Patrón Facade para Gestión de Reseñas y Comentarios*

#### Strategy

#### El patrón Strategy es el adecuado para personalizar las medidas de seguridad al gestionar el historial de transacciones. Permite elegir la estrategia adecuada de acuerdo al contexto del llamado a la clase Servicio

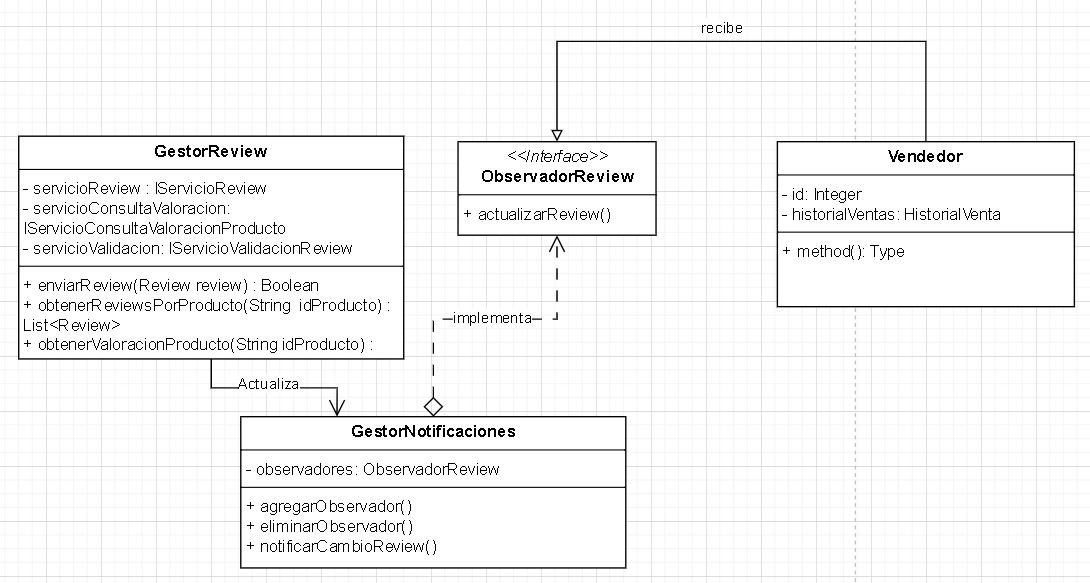


*Imagen: Patrón Strategy para Gestión de Historial de Transacciones*

#### 

#### Observer

#### El patrón Observer se implementa para permitir la notificación automática a los vendedores cuando se producen cambios en las reseñas de productos o cuando se registra una nueva reseña.



*Imagen: Patrón Observer para Notificaciones.*

#### 

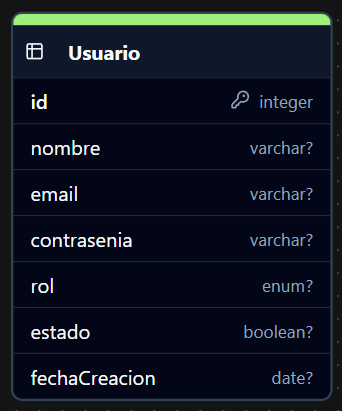
## 

## Patrones de Persistencia

### Identity Field

Se ha utilizado el patrón Identity Field para asignar un identificador único a cada objeto persistente en la base de datos. Los beneficios que se han encontrado son:

* Identificación Unívoca: Cada objeto tiene un identificador que permite diferenciarlo de otros objetos, lo que facilita las consultas SQL.
* Simplificación: Simplificar las consultas SQL ya que el identificador es utilizado para las operaciones CRUD.



*Imagen: Patrón Identity Field para los objetos*

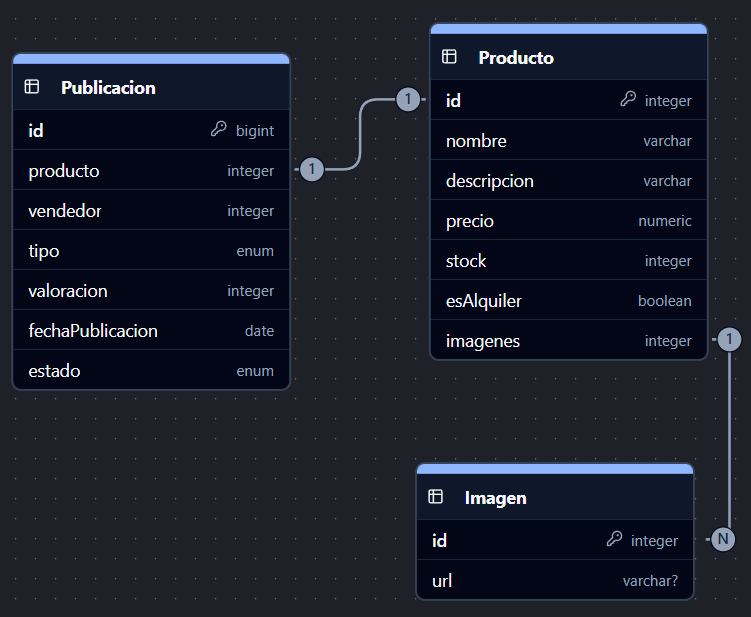
### 

### Foreign Key Mapping

Este es otro recurso recurrente que se puede evidenciar en el trabajo, debido a que de esta forma es posible trabajar con mayor comodidad las relaciones de datos de 1 a muchos. Fácilmente evidenciables en casos como el siguiente:

La relación entre el objeto review y comprador presenta una cardinalidad de 1 a muchos una reseña debe haber sido realizada por un único comprador, y un comprador puede hacer muchas o ninguna reseña.

La relación entre objeto review y vendedor presenta una cardinalidad de 1 a muchos porque una reseña se notifica a un único vendedor, y un vendedor es notificado por muchas o ninguna reseñas.



*Imagen: Patrón Foreign Key Mapping para publicaciones*

### 

### Class Table Inheritance

Una de las razones por la que se escogió este patrón de persistencia es debido a que es un modelo de datos entendible y no hay desperdicio de espacio, pues se utiliza una sola llave primaria para todas las clases heredadas que parten de la clase padre. Por otro lado, permite una relación directa entre clases y tablas.

Usado para llevar una distinción más adecuada de algunos datos como en el siguiente caso dentro el proyecto:

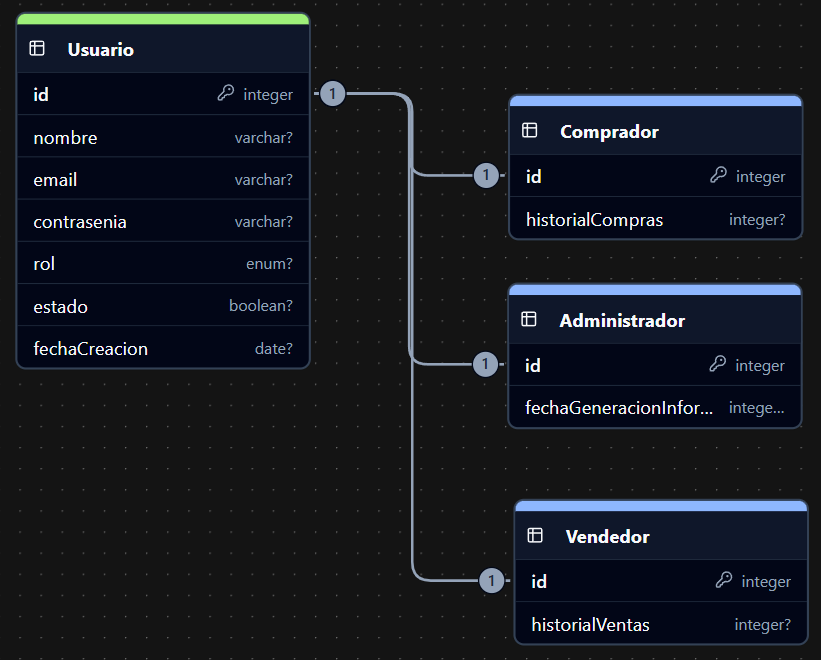


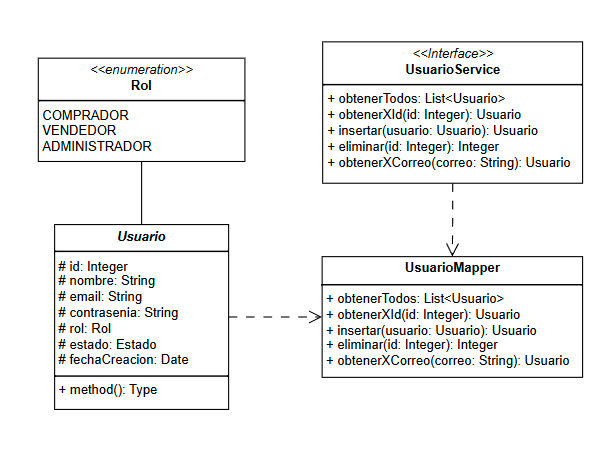
Imagen: Patrón Class Table Inheritance para las herencias.

## 

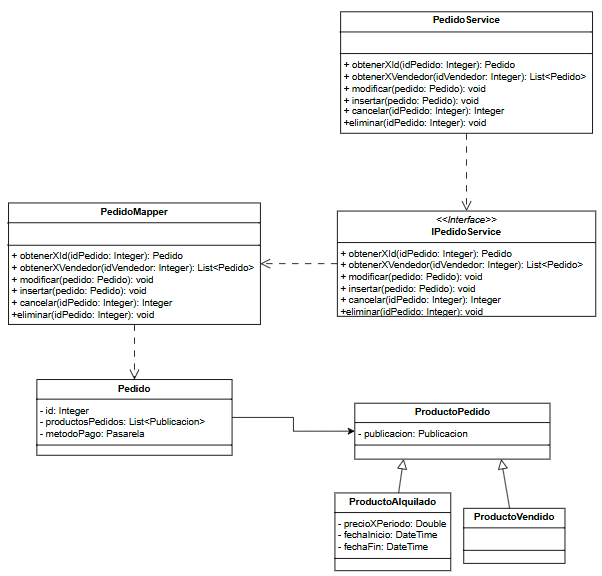
## Patrones de Interacción con la Base de Datos

### Data Mapper

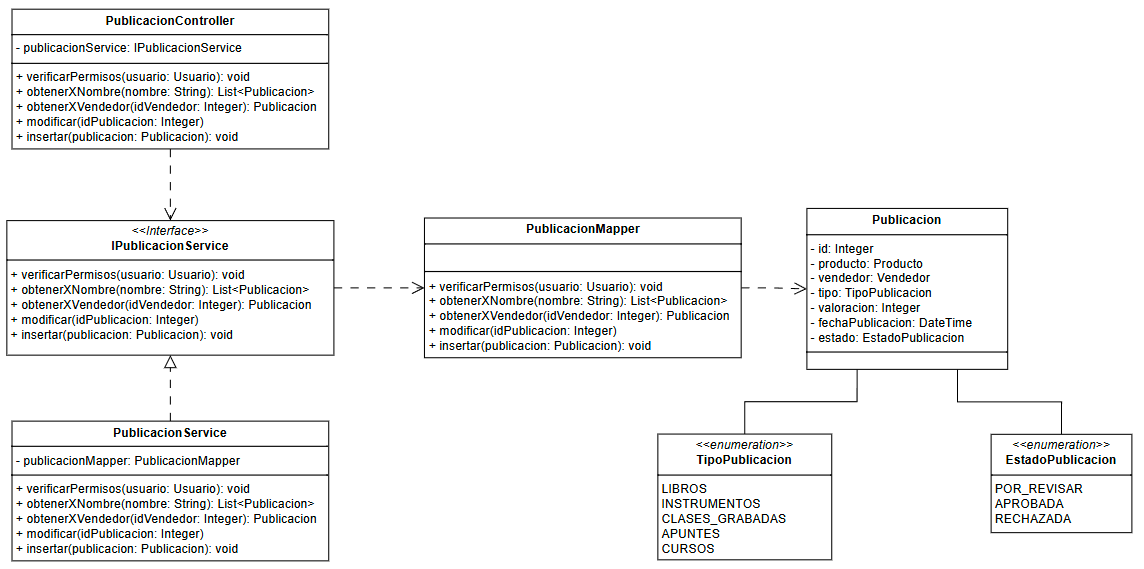
Se ha utilizado en el diseño de la presente aplicación, el patrón de Data Mapper para transferir datos entre los objetos entidad y el repositorio de datos. El uso de este permite soportar la lógica compleja como herencias e independiza la estructura de la entidad de la estructura de la tabla. Además, es la clase encargada de contener las consultas SQL que pueden devolver tanto una instancia del objeto como una lista de instancias.



*Imagen: Patrón Mapper en la gestión de usuarios.*

**

*Imagen: Patrón Mapper en la gestión de productos.*

**

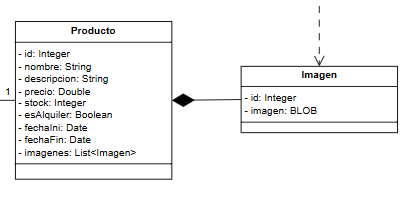
*Imagen: Patrón Mapper en la gestión de publicaciones*

### Objeto Grande (LOB) Serializado

Se implementará para la clase de imagen debido a que, a pesar que las imágenes de los productos son elementos esenciales para atraer a los clientes, almacenarlas directamente en un sistema de archivos puede dificultar su gestión y recuperación.

Al utilizar LOB Serializado, las imágenes se almacenan directamente en la base de datos lo que ofrece las ventajas de:

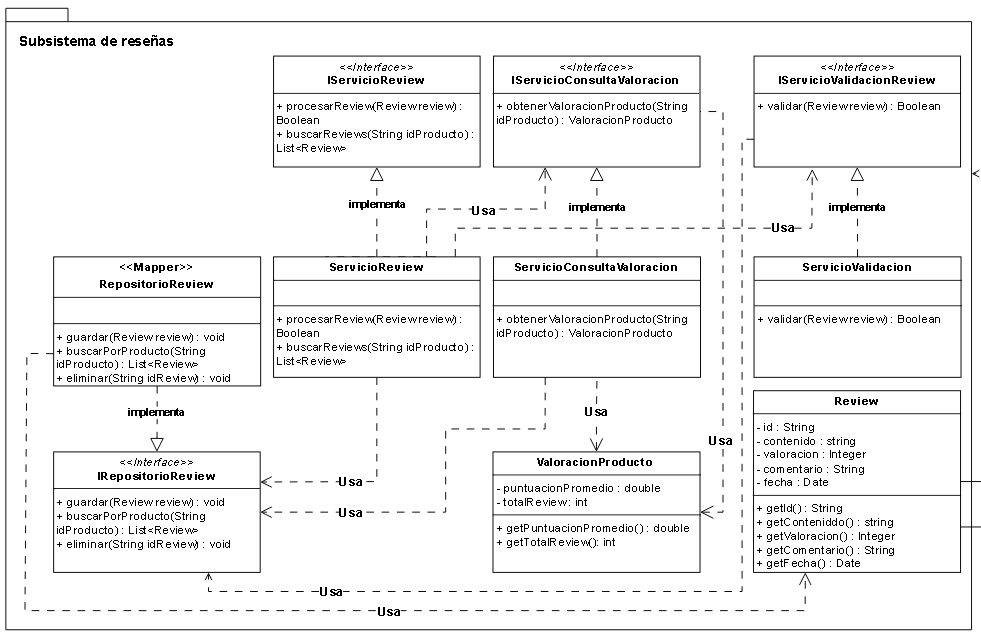
* Consistencia Transaccional: Las imágenes y datos asociados se mantienen en la misma transacción.
* Facilidad de Consultas: Las consultas que combinan datos y BLOBs pueden ser realizadas de manera más eficiente permitiendo buscar productos basados en ciertos atributos y retornas imágenes correspondientes en una única consulta.

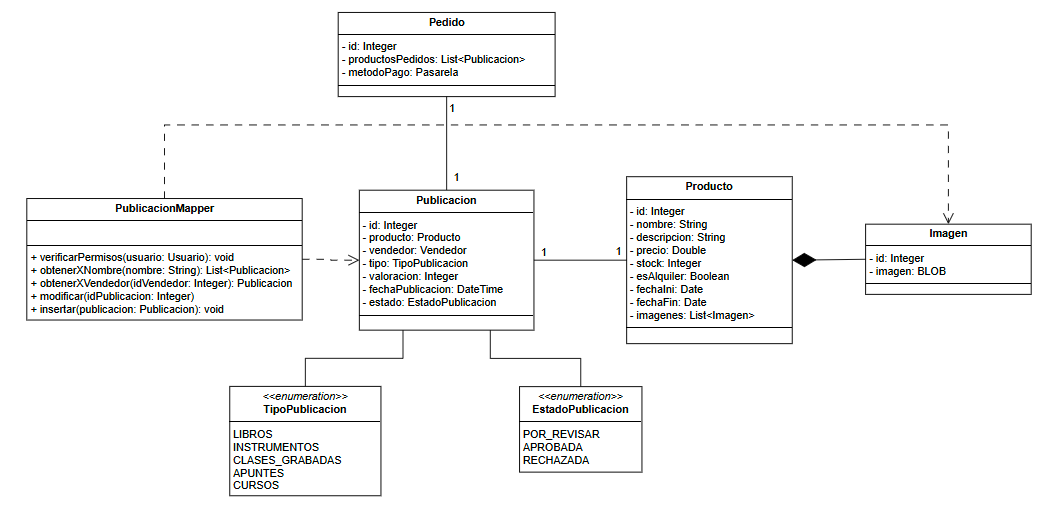


*Imagen: Objeto Grande (LOB) Serializado*

### Dependent Mapping

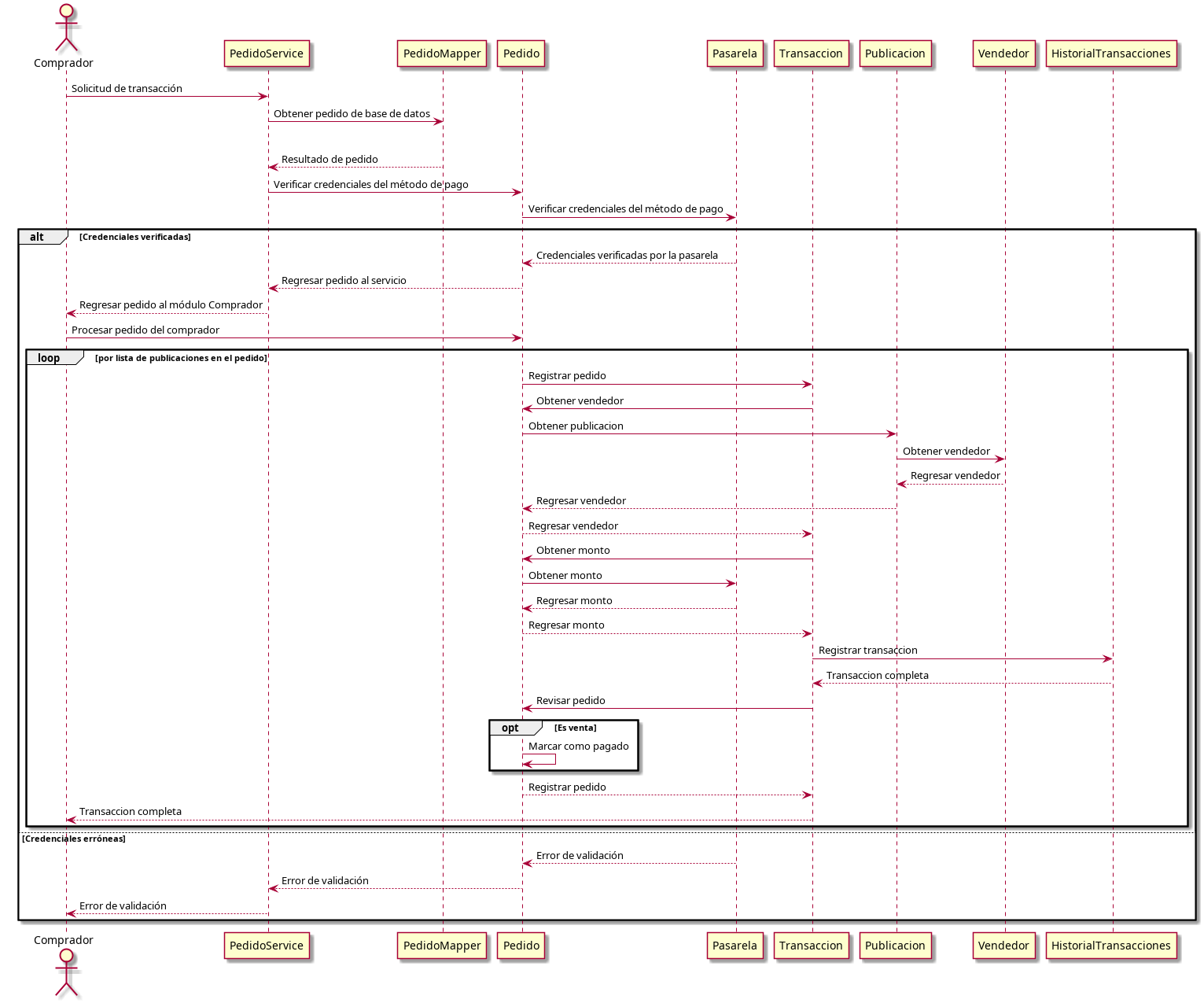
Se utiliza el patrón Dependent Mapping con el fin de mantener una clara separación entre la lógica de dominio y la lógica de persistencia. El mapeo de los objetos del dominio de reseñas se delega a una clase mapper, que actúa como repositorio de reseñas, evitando así que el modelo de dominio se vea afectado por detalles de acceso a datos o de integración con servicios externos al subsistema. Asimismo, esta clase Dependent Mapping gestiona la interacción con los distintos servicios del subsistema de reseñas como la validación, el almacenamiento y la consulta de la valoración del producto mediante interfaces que abstraen su implementación concreta, promoviendo un bajo acoplamiento y una arquitectura más mantenible y escalable.

**

*Imagen: Dependent mapping*

## Diagramas de Interacción

### Diagrama de interacción de gestión de transacciones

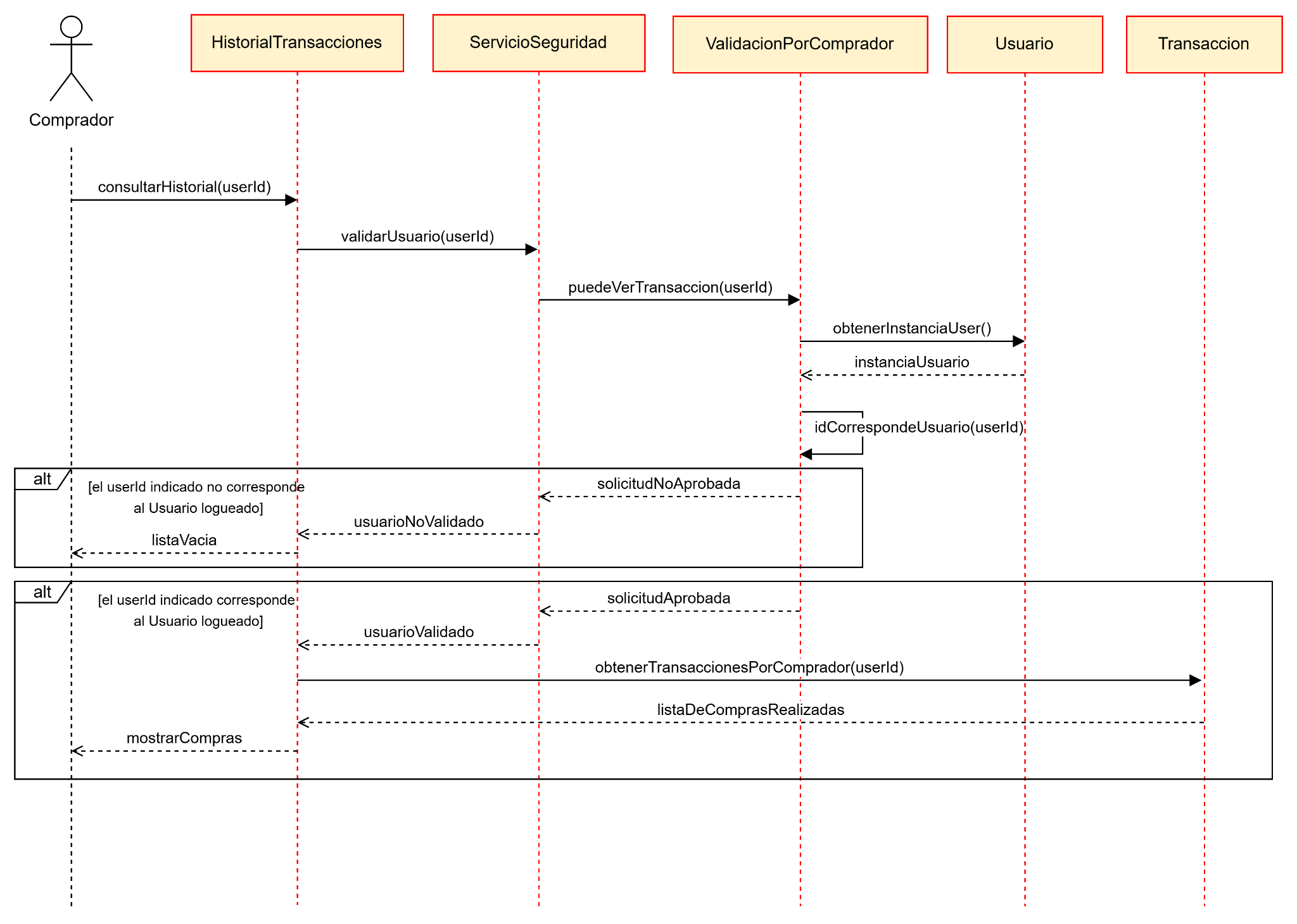


*Imagen: Diagrama de secuencia de Gestión de Transacciones.*

En este diagrama de interacción, con el pedido ya registrado, el usuario realiza una transacción, sea de venta o por alquiler.

1. El comprador inicia el proceso enviando una solicitud al PedidoService.
2. El servicio busca el pedido por ID usando PedidoMapper.
3. Si el pedido existe, el sistema valida las credenciales del cliente con una pasarela de pago.
4. Si las credenciales son válidas, se recorren las publicaciones asociadas al pedido y se generan transacciones por cada una.
5. Si las credenciales son incorrectas, se devuelve un error al cliente.

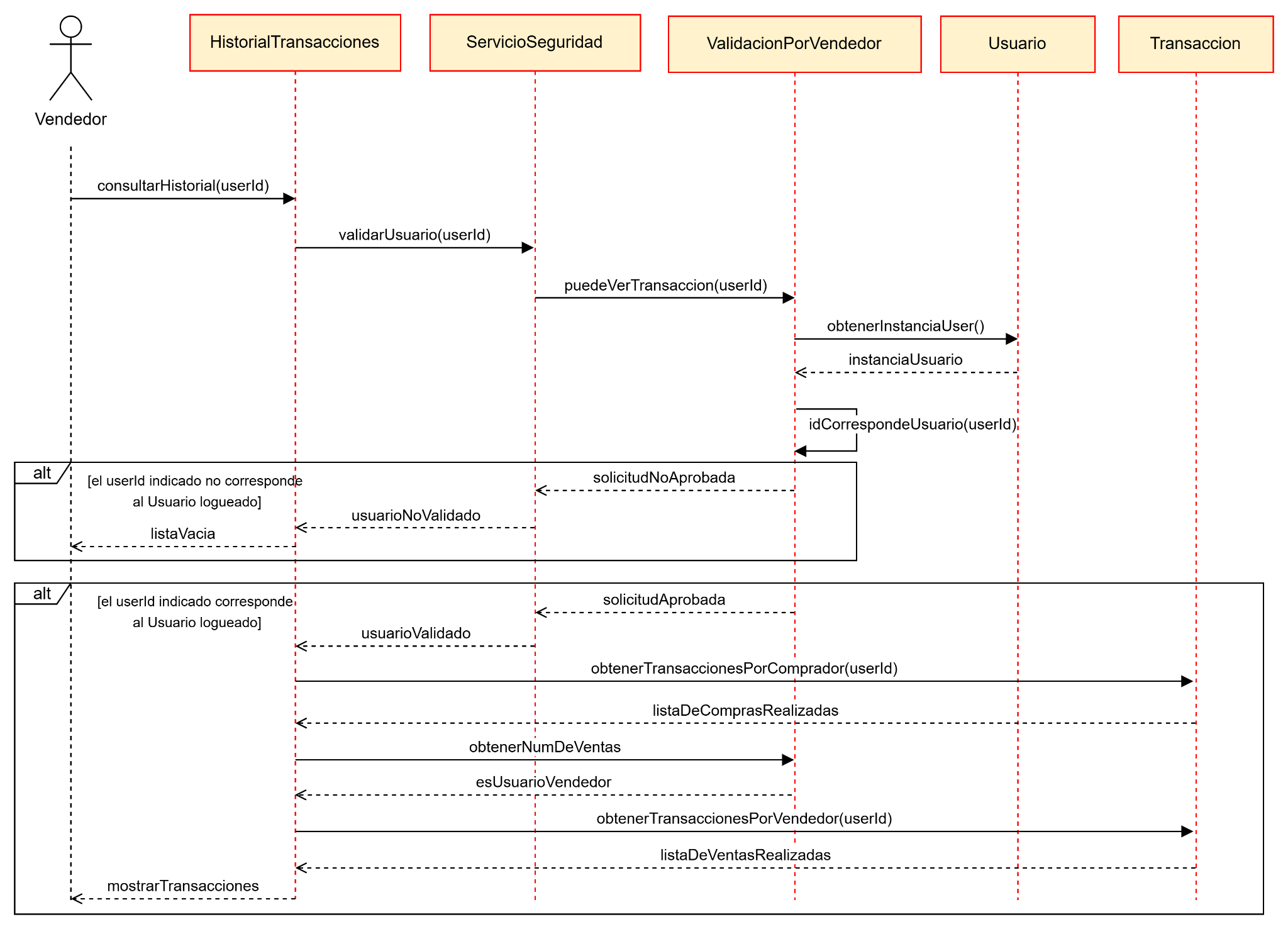
### Diagrama de interacción de historial de transacciones



*Imagen: Diagrama de secuencia de Historial de Transacciones para Compradores*

En este diagrama de interacción, el comprador solicita el historial de transacciones. Previo a la visualización de los resultados, es importante que se restrinjan resultados mediante validaciones por roles y que se visualicen las transacciones correspondientes.

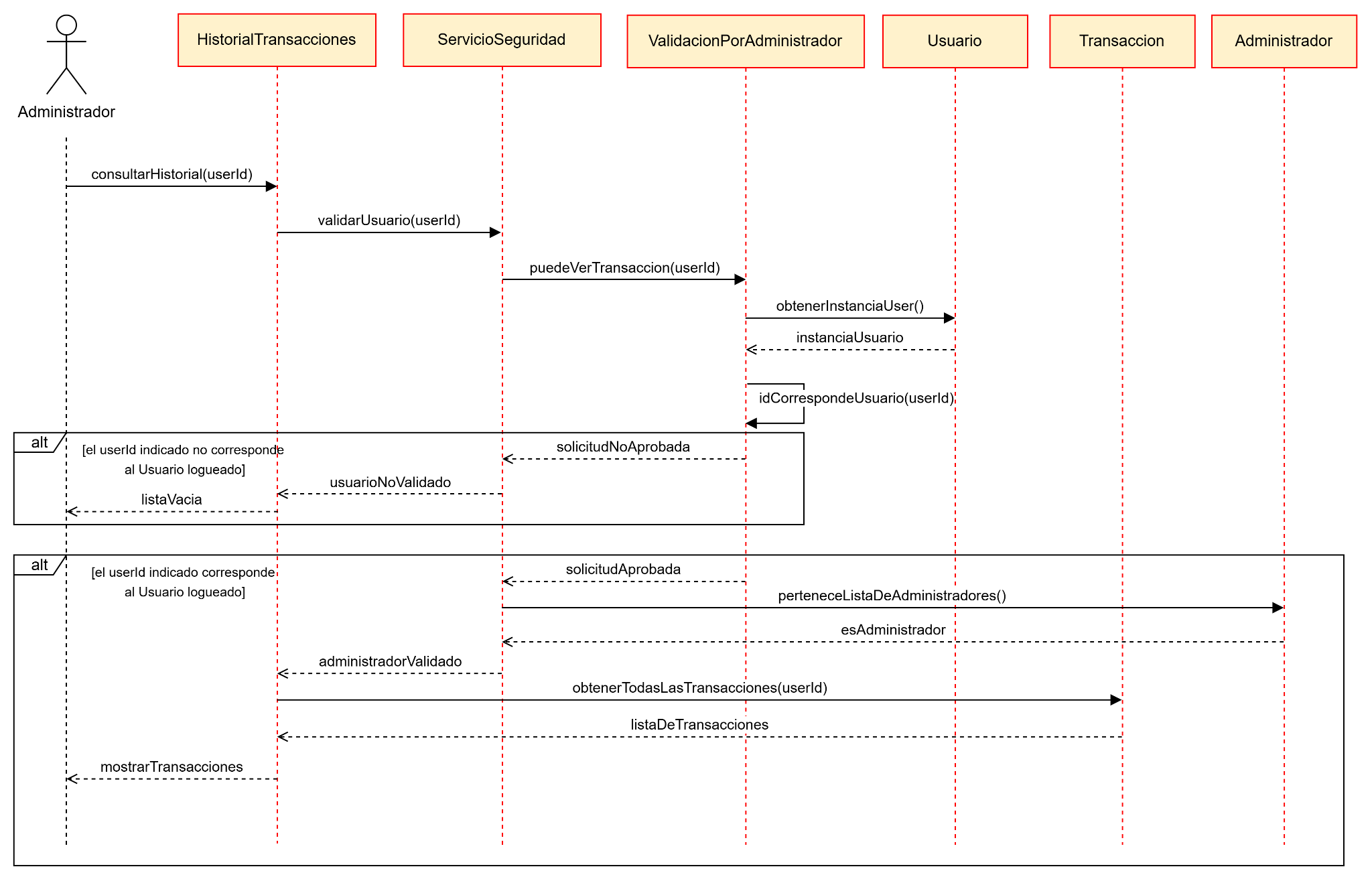
1. El Comprador inicia el proceso llamando a la clase HistorialTransacciones con el método consultarHistorial y envía su userId como parámetro del método.
2. La clase HistorialTransacciones valida el usuario que solicita las transacciones con el atributo seguridad con la clase ServicioSeguridad y también envía el userId como parámetro..
3. El ServicioSeguridad cumple el rol de contexto y delega la validación a IEstrategiaValidaciónAcceso. Esta interfaz, dependiendo del tipo de Usuario, accede a una de las 3 estrategias de validación. Para este caso, el Usuario es un Comprador y se aplica ValidaciónPorComprador.
4. La validación para un Comprador consiste en obtener el usuario logueado actualmente y verificar si el userId de ese usuario y la userId como parámetro coinciden. Esto se realiza para evitar que un Comprador pueda acceder al histórico de otros usuarios.
5. Posterior a la validación del usuario, se retorna una confirmación hacia la clase HistorialTransacciones
6. Luego de haber obtenido la validación, HistorialTransacciones recien puede buscar y recopilar en una lista los resultados correspondientes.
7. Finalmente, HistorialTransacciones retorna la lista de transacciones al Comprador solicitante.



*Imagen: Diagrama de secuencia de Historial de Transacciones para Vendedores*

En este diagrama de interacción, el vendedor solicita el historial de transacciones. Previo a la visualización de los resultados, es importante que se restrinjan resultados mediante validaciones por roles y que se visualicen las transacciones correspondientes.

1. El Vendedor inicia el proceso llamando a la clase HistorialTransacciones con el método consultarHistorial y envía su userId como parámetro del método.
2. La clase HistorialTransacciones valida el usuario que solicita las transacciones con el atributo seguridad con la clase ServicioSeguridad y también envía el userId como parámetro..
3. El ServicioSeguridad cumple el rol de contexto y delega la validación a IEstrategiaValidaciónAcceso. Esta interfaz, dependiendo del tipo de Usuario, accede a una de las 3 estrategias de validación. Para este caso, el Usuario es un Vendedor y se aplica ValidacionPorVendedor.
4. La validación para un vendedor consiste en obtener el usuario logueado actualmente y verificar si el userId de ese usuario y la userId como parámetro coinciden. Además, se verifica el número de ventas realizadas por el usuario. Esto se realiza para evitar que un vendedor pueda acceder al histórico de compras de otros usuarios pero sí pueda acceder a las transacciones en las cuales él fue el vendedor.
5. Posterior a la validación del usuario, se retorna una confirmación hacia la clase HistorialTransacciones
6. Luego de haber obtenido la validación, HistorialTransacciones recien puede buscar y recopilar en una lista los resultados correspondientes.
7. Finalmente, HistorialTransacciones retorna la lista de transacciones al Vendedor solicitante.

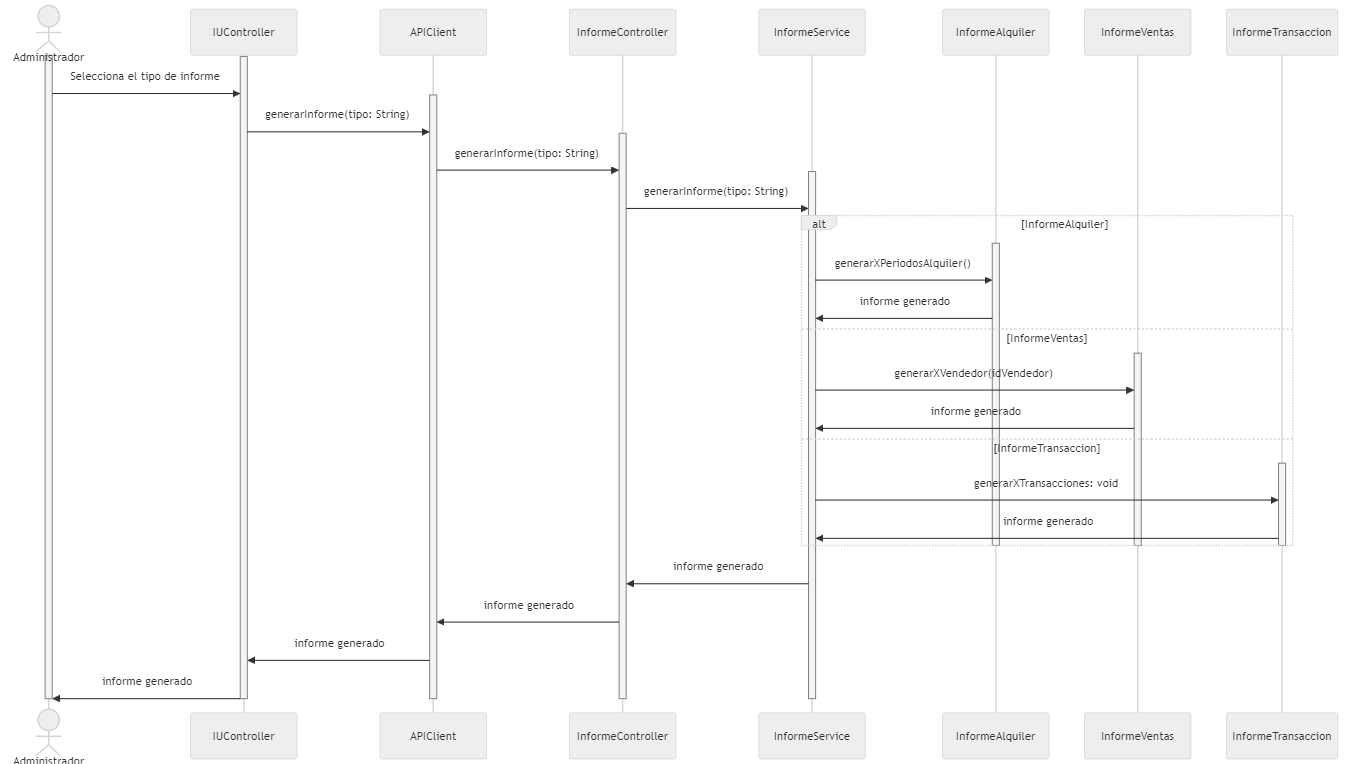


*Imagen: Diagrama de secuencia de Historial de Transacciones para Administrador*

En este diagrama de interacción, el administrador solicita el historial de transacciones. Previo a la visualización de los resultados, es importante que se restrinjan resultados mediante validaciones por roles y que se visualicen las transacciones correspondientes.

1. El Administrador inicia el proceso llamando a la clase HistorialTransacciones con el método consultarHistorial y envía su userId como parámetro del método.
2. La clase HistorialTransacciones valida el usuario que solicita las transacciones con el atributo seguridad con la clase ServicioSeguridad y también envía el userId como parámetro..
3. El ServicioSeguridad cumple el rol de contexto y delega la validación a IEstrategiaValidaciónAcceso. Esta interfaz, dependiendo del tipo de Usuario, accede a una de las 3 estrategias de validación. Para este caso, el Usuario es un Administrador y se aplica ValidacionPorAdministrador.
4. La validación para un administrador consiste en obtener el usuario logueado actualmente y verificar si el userId de ese usuario y la userId como parámetro coinciden. Además, se verifica que el userId del administrador coincida con los administrador registrados en la base de datos. Esto se realiza para evitar que un usuario no autorizado pueda acceder a los privilegios que sí cuenta un administrador.
5. Posterior a la validación del usuario, se retorna una confirmación hacia la clase HistorialTransacciones
6. Luego de haber obtenido la validación, HistorialTransacciones recien puede buscar y recopilar en una lista los resultados correspondientes.
7. Finalmente, HistorialTransacciones retorna toda la lista de transacciones al administrador solicitante.

### Diagrama de interacción de generación de informes



*Imagen: Diagrama de secuencia para la generación de informes.*

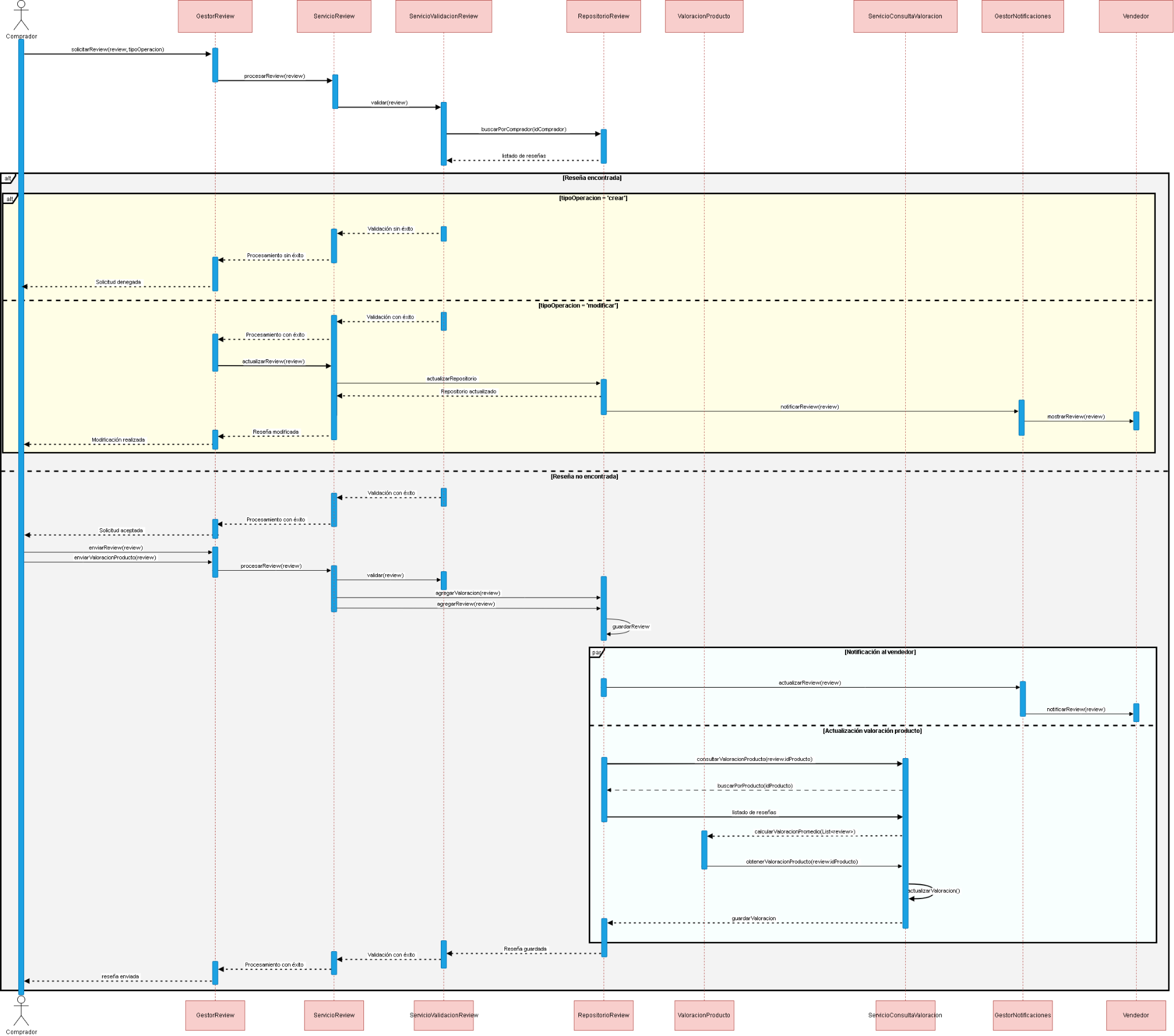
En el diagrama de secuencia presentado a continuación, el usuario Administrador mediante la interfaz visual, solicita la generación de un informe, luego, el APICliente, el encargo de generar las solicitudes HTTP, envía el método con el tipo de informe como parámetro del método,

En este diagrama de interacción, el administrador solicita el historial de transacciones. Previo a la visualización de los resultados, es importante que se restrinjan resultados mediante validaciones por roles y que se visualicen las transacciones correspondientes.

La generación de cada informe y los parámetros enviados a los métodos va a depender del tipo de informe que sea: Alquiler, Vendedores y Transacciones.

* Alquiler: Sin parámetros (se devuelven todos los alquileres de los últimos 30 días).
* Vendedores: IdVendedor (se devuelve las ventas realizadas por el vendedor).
* Transacciones: Sin parámetros (se devuelven todas las transacciones de los últimos 30 días).

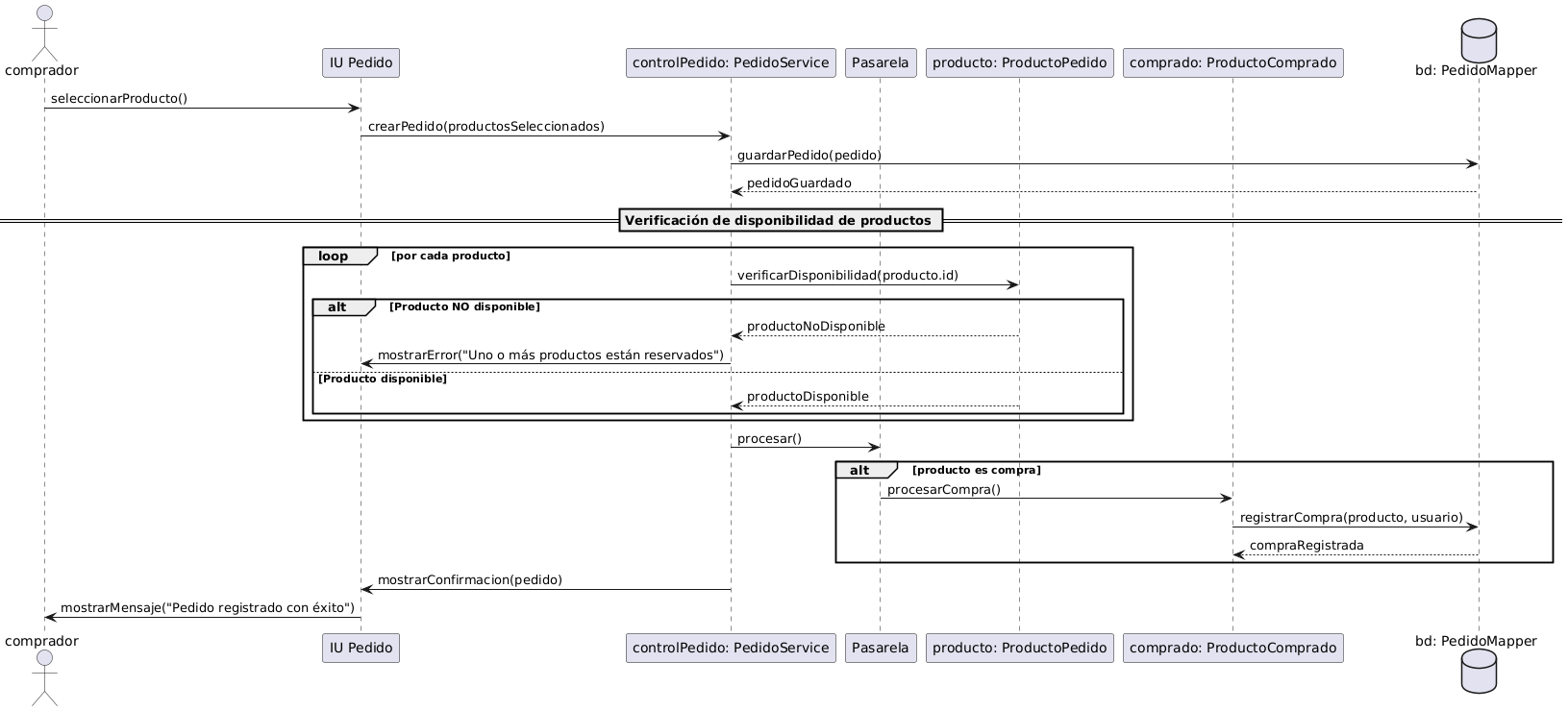
### Diagrama de interacción para la gestión de reseñas y comentarios



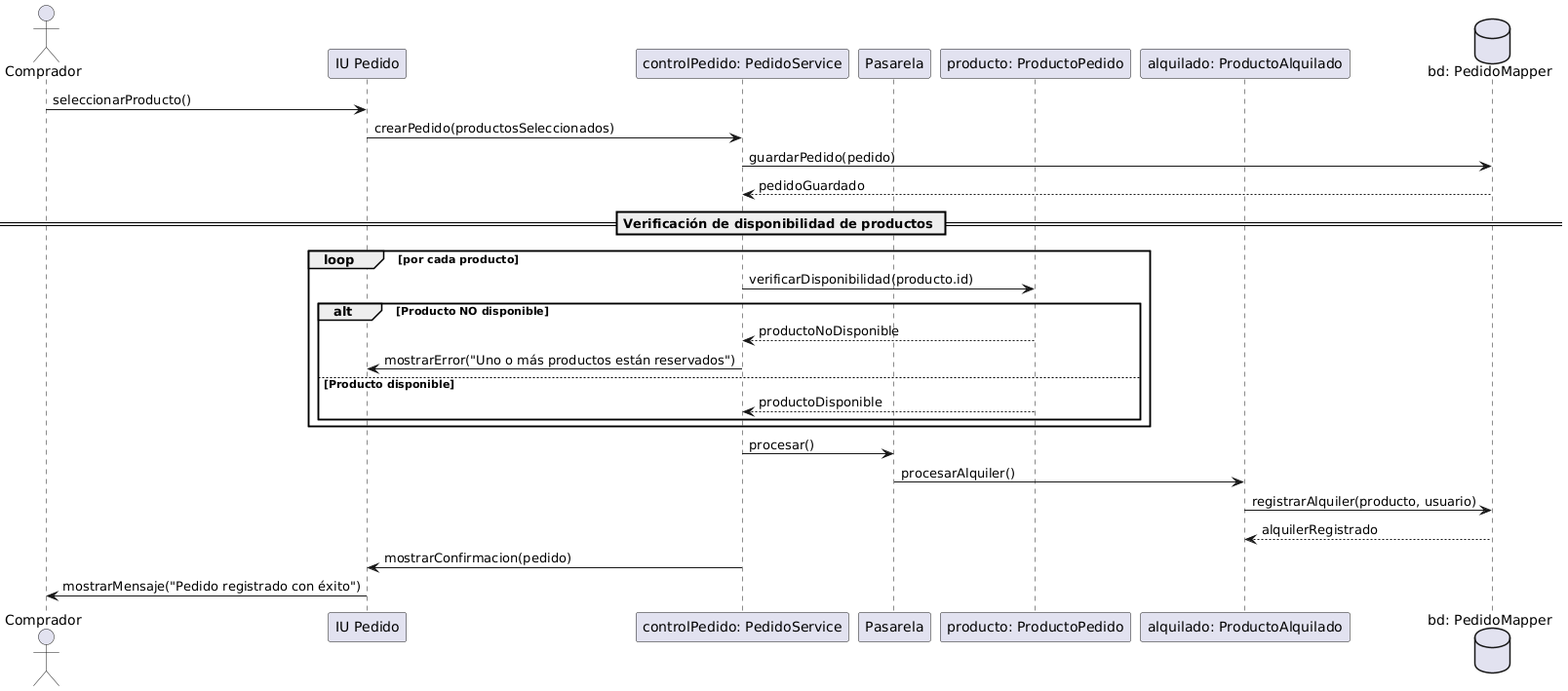
*Imagen: Diagrama de interacción para gestión de reseñas y comentarios.*

1. En el diagrama de interacción de reseñas y comentarios, una instancia de la clase Comprador inicia el proceso enviando un mensaje a la clase GestorReview mediante el método solicitarReview, que recibe como parámetros la reseña y el tipo de operación (crear o modificar). Luego, GestorReview envía la solicitud a ServicioReview a través del método procesarReview. Este servicio solicita la validación correspondiente mediante el método validarReview, el cual es gestionado por la clase ServicioValidacionReview. Para realizar esta validación, se consulta al RepositorioReview utilizando el método buscarPorComprador, que recibe como parámetro el idComprador y devuelve un listado de reseñas asociadas.
2. Si la reseña ya existe, se evalúa el tipo de operación. Si se intenta crear una nueva reseña para un producto ya comentado, se retorna un mensaje de validación fallida, lo cual provoca que ServicioReview y luego GestorReview devuelvan mensajes de procesamiento y solicitud denegada al comprador. En cambio, si la operación es de modificación, la validación es exitosa, lo que permite que GestorReview envíe una solicitud de actualización a ServicioReview mediante el método actualizarReview. A su vez, este actualiza la información en RepositorioReview y responde con la confirmación de la modificación. Finalmente, GestorReview informa al comprador sobre la modificación exitosa. Paralelamente, se inicia el proceso de notificación: RepositorioReview invoca el método notificarReview en GestorNotificaciones, que notifica al Vendedor mediante el método mostrarReview, permitiéndole visualizar la reseña actualizada.
3. En caso de que la reseña no exista, ServicioValidacionReview retorna una validación exitosa, permitiendo que la solicitud avance. Posteriormente, el comprador envía la reseña con su valoración mediante los métodos enviarReview y enviarValoracion. Esta información es procesada y validada nuevamente por ServicioReview, el cual agrega la reseña al RepositorioReview (agregarReview) y registra la valoración en ServicioConsultaValoracion (agregarValoracion). Para actualizar el promedio de valoraciones del producto, ServicioReview invoca el método consultarValoracionProducto. ServicioConsultaValoracion consulta las reseñas existentes a través de buscarPorProducto, calcula el nuevo promedio con calcularValoracionPromedio, obtiene el valor actual (obtenerValoracionProducto), lo actualiza (actualizarValoracion) y luego se guarda el nuevo promedio mediante guardarValoracion, cerrando así el ciclo de valoración. RepositorioReview retorna un mensaje de reseña guardada a la clase ServicioValidacionReview.
4. Finalmente, RepositorioReview retorna un mensaje de reseña guardada a ServicioValidacionReview, quien valida nuevamente la información, y ServicioReview notifica al comprador que la reseña fue enviada correctamente.

### Diagrama de interacción de registro de pedidos



*Imagen: Diagrama de Interacción de Registro de Pedidos para compra.*

******

*Imagen: Diagrama de Interacción de Registro de Pedidos para alquiler.*

El diagrama de interacción inicia con la selección de productos por parte del comprador para luego ser validados con el sistema, si el procesamiento del producto es para compra o alquiler, y la confirmación del pedido.

1. El comprador selecciona productos a través de la interfaz (Pedido). La interfaz envía la solicitud al servicio de pedidos (controlPedido) para crear el pedido.

2. El controlPedido guarda el nuevo pedido en la base de datos mediante PedidoMapper. La base de datos confirma que el pedido ha sido registrado exitosamente.

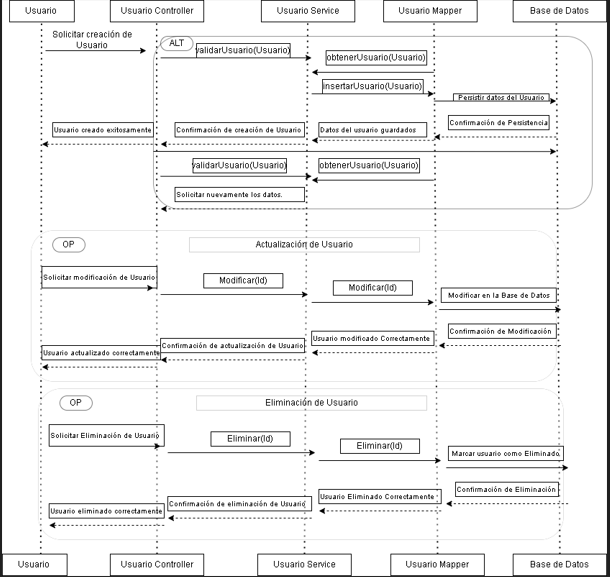
3. Por cada producto en el pedido, el sistema verifica su disponibilidad en la base de datos. Si algún producto no está disponible, se muestra un mensaje de error al comprador y el proceso se detiene inmediatamente. Caso contrario, el proceso continúa.

4.Por cada producto disponible, se ejecuta la operación procesar() sobre el producto. y se diferencia el gráfico si es de compra o si es de alquiler, si es compra, se llama a procesarCompra(),  
por otro lado si es alquiler, se llama a procesarAlquiler(). Finalmente se registra en la base de datos.

5. Una vez procesado todo el pedido, el sistema muestra una confirmación a través de la interfaz. El comprador recibe el mensaje: “Pedido registrado con éxito.”

### Diagrama de interacción de Gestión de Usuarios

El diagrama de interacción para la gestión de usuarios comienza con el registro del usuario, seguido por la actualización de datos y la eliminación de cuentas, incluyendo la validación de privilegios.



#### 1. Creación de Usuario (Registro)

* El **Usuario** solicita el registro al UsuarioController.
* El UsuarioController y UsuarioService colaboran para **validar los datos** y **verificar que el usuario no exista** (buscando en la Base de Datos vía UsuarioMapper).
* **Si la validación es exitosa y el usuario no existe**, el UsuarioService instruye al UsuarioMapper para Insertar los datos en la Base de Datos.
* La Base de Datos confirma la persistencia, y esta confirmación se propaga de vuelta al UsuarioController, quien informa al **Usuario** del registro exitoso.
* **Si la validación falla o el usuario ya existe**, se notifica al **Usuario** para que intente de nuevo con datos válidos.

#### 2. Actualización de Usuario

* El **Usuario** solicita modificar sus datos al UsuarioController.
* El UsuarioController llama a Modificar(id) en el UsuarioService.
* El UsuarioService (tras posibles validaciones internas no explícitas de privilegios o datos) indica al UsuarioMapper que Modificar los datos en la Base de Datos.
* La Base de Datos confirma la modificación, y esta confirmación se propaga de vuelta al UsuarioController, quien informa al **Usuario** de la actualización exitosa.

#### 3. Eliminación de Usuario

* El **Usuario** solicita eliminar su cuenta al UsuarioController.
* El UsuarioController invoca Eliminar(id) en el UsuarioService.
* El UsuarioService instruye al UsuarioMapper para Marcar usuario como Eliminado en la Base de Datos (eliminación lógica).
* La Base de Datos confirma la operación, y esta confirmación se propaga de vuelta al UsuarioController, quien informa al **Usuario** de la eliminación exitosa.

# Plan de Pruebas

## Contexto de las Pruebas

### Uso del Sistema

El sistema se utilizará para la compra, venta y alquiler online de materiales estudiantiles dirigido a los miembros de la PUCP. Asimismo, por la parte administrativa se cuenta con una interfaz para gestionar las publicaciones y las ventas y alquileres realizadas. Algunas de las funciones principales son:

1. Venta online de materiales estudiantiles
2. Compra online de materiales estudiantiles
3. Alquiler online de materiales estudiantiles por un periodo de tiempo definido por el vendedor
4. Gestión de las publicaciones para la venta y alquiler de materiales estudiantiles

Es necesario que el sistema pueda realizar las funciones mencionadas anteriormente para que pueda cumplir con los requerimientos propuestos por Squirt Learn.

### Elementos a Probar

#### A nivel de Usuario (General)

##### Login para Comprador, Vendedor

##### Login para Administrador

#### A nivel de Comprador

Sistema de Publicaciones

Comunicación en la Aplicación

#### A nivel de Vendedor

* + - * 1. Sistema de Publicaciones
        2. Comunicación en la Aplicación

#### A nivel de Administrador

* + - * 1. Sistema de Publicaciones
        2. Gestión de Usuarios

### Alcance de las Pruebas

#### Pruebas de Caja Negra

Se ha escogido la prueba de caja negra debido a que se evalúa la funcionalidad del sistema sin conocer el código interno. Esto permite probar desde la perspectiva del usuario final y es útil para validar requerimientos funcionales. Además, es posible detectar errores en la lógica de negocio y funciones visibles para el usuario.

#### Pruebas de Integración

Verifican que diferentes módulos del sistema funcionan bien en conjunto. Esto es útil para detectar errores en la comunicación entre componentes (APIS), así como identificar fallos que no aparecen en pruebas unitarias.

#### Pruebas de Rendimiento

Las pruebas de estrés evalúan la capacidad del sistema para continuar con sus operaciones bajo diferentes cargas de trabajo. Este tipo de pruebas es fundamental para un ecommerce, que es un software con alta demanda debido a la cantidad de transacciones en simultáneo. Además, permite identificar cuellos de botella, problemas de escalabilidad o consumo excesivo de recursos. Por otro lado, ayudan a planificar necesidades de infraestructura orientado al escalado horizontal o vertical.

#### Pruebas de Usabilidad

Estas pruebas evalúan la facilidad de uso e interacción del usuario con el sistema, lo cual es importante para detectar barreras en la navegación, compresión o interacción de los usuarios con la interfaz. Además, están orientadas a mejorar la experiencia de usuario y su satisfacción.

### Asunciones

#### Datos a disposición

Para realizar y aproximar las pruebas a un contexto cercano a lo real, se utilizarán datos que permitan analizar el flujo completo durante el funcionamiento del software. Estos datos serán almacenados en un esquema de la base de datos dedicado para pruebas y puestos a disposición del equipo encargado de las pruebas.

#### Ambiente de Pruebas

Se asumirá un ambiente de pruebas dedicado con una instancia del sistema y un conjunto de datos cargados específicamente para la ejecución de pruebas funcionales. Este ambiente contará con las funciones activas para cubrir los casos de prueba planificados.

#### Disponibilidad del Software

Se asumirá que el software está disponible para realizar las pruebas dentro de periodos de mantenimiento definidos que no afecten el funcionamiento del sistema.

### Restricciones

* El entorno de pruebas no debe tener conexión directa con el sistema de producción.
* Las realización de las pruebas deberán ser en horarios previamente definidos.
* El uso de los datos puede llegar a ser limitado debido a la sobrecarga del sistema.

## Enfoque de Pruebas

* + 1. **Analítico**: Se priorizaron los módulos vitales como pagos y autenticación, usando métricas como historial de fallos, complejidad técnica y frecuencia de uso.
    2. **Dinámico**: Las pruebas dinámicas permitirán detectar errores reales durante la ejecución (bugs, cuellos de botella, errores de interfaz).
    3. **Dirigido**: Se asegurará que cada funcionalidad pueda ser ejecutada correctamente por el rol correspondiente.
    4. **Regresionista**: Las pruebas se enfocarán en validar que nuevas funcionalidades no interfieran con las ya existentes.

## Comunicación del Plan

El objetivo de la siguiente tabla es que todos los interesados estén informados del alcance, avances y resultados del plan de pruebas y que puedan tomar decisiones oportunas según su rol.

| ID | Interesado | Medio de Comunicación | Frecuencia |
| --- | --- | --- | --- |
| P1 | Administrador | Reuniones virtuales o presenciales. Reportes visuales. | Semanal |
| P2 | Equipo de Pruebas | Reuniones virtuales o presenciales. | Diario |
| P3 | Gerente General | Reuniones presenciales. Reportes del avance. | Mensual |
| P4 | Representante de MercadoPago | Reuniones virtuales. Datos estadísticos. | Mensual |
| P5 | Gerente de Ventas | Reuniones virtuales o presenciales. Reportes de avance. | Mensual |
| P6 | Equipo de Desarrollo | Reuniones virtuales. Reportes de pruebas con retroalimentación. | Diario |
| P7 | Equipo de QA | Reuniones virtuales. | Diario |
| P8 | Soporte | Reuniones virtuales informativas. Cambios funcionales importantes | Al final del ciclo de pruebas |

## Tipos de Pruebas

#### Pruebas Basadas en Requerimientos

Estas pruebas aseguran que cada requerimiento funcional o no funcional se haya implementado correctamente. Además, permiten validar que el desarrollo cumple con lo establecido en la documentación base. Por lo tanto, se reducen las implementaciones no solicitadas en el proyecto y se facilitan las demás pruebas. Asimismo, se enfocan en cubrir todos los casos definidos por el negocio independientemente de la interfaz. Para este contexto, se debe aplicar estas pruebas a los módulos core de la plataforma con la finalidad de poder validar el cumplimiento de los matriz de requerimientos y los módulos seleccionados.

Módulos de aplicación:

* Módulo de Ventas/Alquiler
* Módulo de Gestión de Publicaciones
* Módulo de Transacciones
* Módulo de Comunicación entre usuarios

#### Prueba de Escenarios

Esta prueba simula flujos complejos del usuario desde el inicio hasta el fin de una tarea (compra, venta o alquiler). Permite validar que la aplicación responde correctamente en secuencias reales de uso considerando combinaciones de acciones. Además, analiza los escenarios según roles, importante para evaluar la perspectiva de cada tipo de usuario que interactúa con el sistema. Para este contexto, es necesario evaluar las respuestas de la plataforma ante las diversas interacciones realizadas por los tipos de usuarios que emplea y las respuestas de seguridad esperadas. Finalmente, cada usuario tendrá permisos y funciones específicas.

Módulos de aplicación:

* Módulo de Logueo
* Módulo de Venta/Alquiler
* Módulo de Generación de Informes
* Módulo de Gestión de Publicaciones

#### Pruebas de Humo

Son pruebas rápidas son útiles para confirmar que las funciones principales del sistema funcionan después de un despliegue o cambio. Su objetivo es detectar fallas críticas de inmediato para evitar perder tiempo en probar algo que no funciona. Para este contexto, son vitales para el enfoque regresionista, ya que se busca que los nuevos cambios no afecten o perjudiquen los módulos básicos y primordiales como:

Módulos de aplicación:

* Módulo de Ventas/Alquiler
* Módulo de Gestión de Publicaciones

#### Pruebas de aceptación

Esta prueba permite decidir si el sistema pasa a producción. Es necesario que cumpla con los requerimientos del cliente. Además, se realizan desde la perspectiva del usuario final, validando si el software está listo para continuar con la siguiente etapa del ciclo de vida de un sistema. Para este contexto, esta prueba es importante ya que ayuda a concluir si la plataforma satisface de manera efectiva las necesidades del negocio y el planteamiento central del requerimiento, el cual es: “Generar una experiencia agradable y segura para comercializar materiales académicos”.

* Módulos de aplicación
* Módulo de Ventas/Alquiler
* Módulo de Gestión de Publicaciones

## Riesgos en el Desarrollo de Pruebas

| ID | Riesgo | Justificación |
| --- | --- | --- |
| R1 | Fallos en servicios externos | APIs como MercadoPago o validadores académicos son claves y podrían fallar, ya que su indisponibilidad, generaría errores en sus APIs o cambios inesperados en su comportamiento que pueden afectar directamente la ejecución de pruebas y la estabilidad del sistema. |
| R2 | Cobertura insuficiente | El tiempo disponible podría limitar la verificación de todos los módulos, en especial aquellos de menor prioridad o uso menos frecuente. Esto puede dejar defectos sin detectar que se manifiesten en producción. |
| R3 | Escenarios de prueba poco realistas | El uso de datos simulados podría no reflejar completamente el uso real. Esto puede generar que el sistema se comporte de manera diferente cuando se enfrente a datos reales con variaciones inesperadas, errores humanos, formatos incorrectos o casos extremos. |

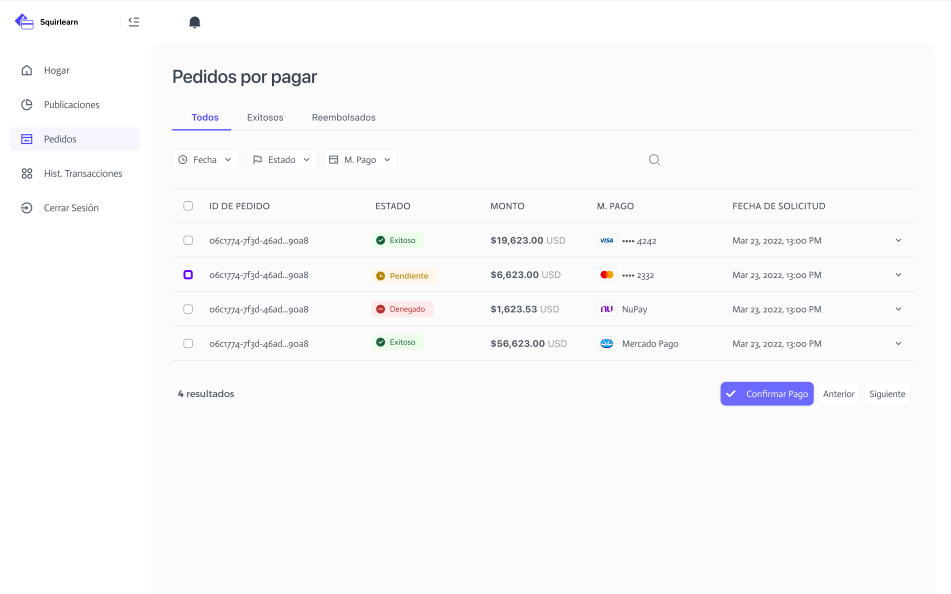
## Cronograma

| Semana | Fecha | Actividad | Responsable / Rol |
| --- | --- | --- | --- |
| Semana 1 | 23/05/25 | Reunión de Inicio de Pruebas: Inicio formal del ciclo de pruebas, definición de responsabilidades y revisión del calendario estimado. | Administrador y Gerente General |
|  | 27/05/25 | Revisión del plan inicial de pruebas: Verificación conjunta del plan diseñado para asegurar que cubre todos los escenarios requeridos. | Administrador y Gerente de Ventas |
|  | 29/05/25 | Informe del avance: Análisis de progreso, revisión de incidencias técnicas. | Equipo de Desarrollo y Equipo de Pruebas |
| Semana 2 | 02/06/25 | Reporte del estado del módulo de ventas: Revisión del comportamiento funcional del módulo y ajustes basados en resultados preliminares. | Gerente de Ventas |
|  | 04/06/25 | Evaluación Intermedia del Proceso de Pruebas:Revisión de avances parciales para definir si se requieren ajustes en la ejecución. | Equipo de Pruebas |
| Semana 3 | 09/06/25 | Valoración de integración de componentes : Prueba del funcionamiento conjunto de los módulos internos y conexión con APIs externas. | Equipo de Desarrollo |
|  | 11/06/25 | Revisión de compatibilidad y Conectividad : Pruebas de funcionamiento estable entre el sistema y servicios externos. | Representante de MercadoPago |
| Semana 4 | 16/06/25 | Análisis del rendimiento del sistema : Pruebas de respuesta del sistema con distintos niveles de carga o escenarios. | Equipo de Pruebas |
|  | 18/06/25 | Informe de ajustes de rendimiento : Reporte técnico con sugerencias para optimizar la eficiencia del sistema. | Equipo de QA |
| Semana 5 | 23/06/25 | Verificación de Sincronización con APIs externas: Confirmación de la correcta interacción con servicios de terceros. | Representante de MercadoPago |
|  | 25/06/25 | Informe sobre estado de APIs integradas: Revisión de desempeño y propuestas para mejorar las integraciones. | Equipo de Desarrollo |
| Semana 6 | 27/06/25 | Corrección de errores | Todo el equipo de desarrollo |
|  | 01/07/25 | Elaboración del informe final de pruebas | Líder del grupo |
| Semana 6 | 04/07/25 | Presentación y entrega final | Todo el equipo |

## Casos de Prueba

### Pantalla de Gestión de Transacciones por Pagar

La pantalla de gestión de transacciones por pagar es vista por el administrador en el que se puede observar las transacciones, su estado, monto y fecha de solicitud para cada pedido.



*Imagen: Pantalla de transacciones por pagar*

**Tabla de Equivalencia:**

| Condición de entrada | Clases válidas | ID | Clases no válidas | ID |
| --- | --- | --- | --- | --- |
| Estado | Pendiente | 1 |  |  |
|  | Exitoso |  |  |  |
|  | Denegado |  |  |  |
|  | Sin especificar |  |  |  |
| Fecha | Formato DD/MM/YYYY (8 números, 10 caracteres), que represente un día existente y que no sea de un día futuro al actual | 2 | * != 10 caracteres. | 6 |
| * Formato incorrecto | 7 |
| * Caracteres inválidos (letras, otros símbolos) | 8 |
| * Día futuro al actual | 9 |
|  | Sin especificar |  |  |  |
| M. Pago | VISA | 3 |  |  |
|  | MasterCard |  |  |  |
|  | NuPay |  |  |  |
|  | MercadoPago |  |  |  |
|  | Sin especificar |  |  |  |
| Id Pedido | Cadena de máximo 36 caracteres (guiones o cifras hexadecimales) | 4 | * Caracteres inválidos | 10 |
| * > 36 caracteres. | 11 |
|  | Sin especificar |  |  |  |
| Transacción seleccionada (casilla) | Transacción seleccionada, de estado igual a “Pendiente” | 5 | * Sin selección | 12 |
| * Estado != “Pendiente” | 13 |

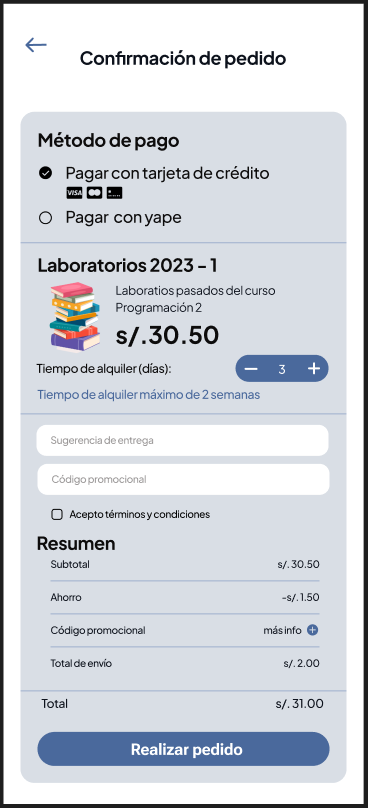
**Casos de Prueba**

| Caso de prueba válido | | |
| --- | --- | --- |
| Prueba R001 | | |
| Objetivo | Confirmación exitosa de una transacción | |
| Precondición | El usuario está autenticado y requiere confirmar la transacción relacionada a un pedido | |
| Descripción de la prueba | 1. Abrir la pantalla de registro.  2. Ingresar a la sección de Transacciones  3. Ingresar:   * Filtro de “pendiente” en estado de transacción * Ningún filtro en método de pago * Filtro 04/07/2025 en fecha * Campo de Id de transacción vacío * Primera o segunda transacción seleccionada | |
| Resultados esperados | Se confirma la transacción, y se emite el mensaje de exito apropiado | |

| Caso de prueba inválido | | |
| --- | --- | --- |
| Prueba R002 | | |
| Objetivo | Verificación del manejo de una confirmación de transacción ya confirmada | |
| Precondición | El usuario está autenticado y requiere confirmar la transacción relacionada a un pedido | |
| Descripción de la prueba | 1. Abrir la pantalla de registro.  2. Ingresar a la sección de Transacciones  3. Ingresar:   * Ningún filtro en estado de transacción * Ningún filtro en método de pago * Filtro de 04/07/2025 en fecha * Campo de Id de transacción vacío * Transacción en estado “Exitoso” seleccionada | |
| Resultados esperados | – Sistema muestra el mensaje “la transacción ya fue confirmada” | |

### Pantalla de Confirmación de Pedido

La pantalla para confirmar un pedido, en este caso de alquiler, permite al usuario comprador seleccionar los parámetros correspondientes a su pedido, los cuales son el método de pago, periodo de alquiler, sugerencia para la entrega y, de haber, código promocional. La finalidad es que el usuario pueda seleccionar los datos que necesita para realizar su pedido.



*Imagen: Pantalla de confirmación de pedido (alquiler).*

**Tabla de equivalencia:**

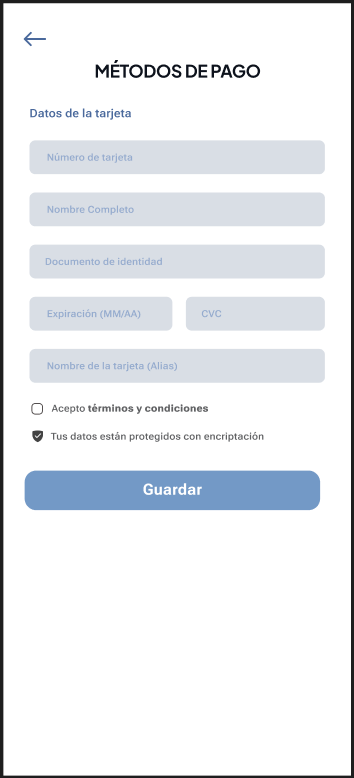
| Condición de entrada | Clases válidas | ID | Clases no válidas | ID |
| --- | --- | --- | --- | --- |
| Método de pago | Booleano (BOOLEAN) según el método de pago escogido. | 1 | - |  |
| Tiempo de alquiler | - |  | - |  |
| Sugerencia de entrega | Cadena de caracteres (VARCHAR). | 2 | Vacío | 5 |
| Caracteres != (a-z,A-Z) | 6 |
| Código promocional | Valor numérico (NUMBER) de 8(ocho) dígitos. | 3 | Vacío | 7 |
| No numérico | 8 |
| Menor a 0, Mayor a 99999999 | 9 |
| Acepto términos y condiciones | Booleano (BOOLEAN) | 4 | - |  |

**Casos de prueba:**

| Caso de prueba válido | | |
| --- | --- | --- |
| Prueba P001 | | |
| Objetivo | Confirmar pedido (alquiler) | |
| Precondición | Ingresar a la aplicación como un usuario autorizado (comprador) | |
| Descripción de la prueba | En la interfaz del sistema deberá introducir:   * Selección de método de pago * Tiempo de alquiler (menor al máximo) * Sugerencia * Código promocional * Términos y condiciones   Una vez estén todos los campos completados y que el usuario haya aceptado los términos y condiciones, podrá continuar con el proceso de alquiler. | |
| Resultados esperados | Se continua con el proceso de alquiler. | |
| Caso de prueba inválido | | |
| Prueba P002 | | |
| Objetivo | No permitir la confirmación del pedido (alquiler) | |
| Precondición | Ingresar a la aplicación como un usuario autorizado (comprador) | |
| Descripción de la prueba | En la interfaz del sistema deberá introducir:   * Selección de método de pago * Tiempo de alquiler (mayor al máximo) * Sugerencia * Código promocional * Términos y condiciones | |
| Resultados esperados | Se muestra un mensaje de “El periodo seleccionado es superior al máximo permitido. Intente de nuevo.” | |

### Pantalla de Agregar Método de Pago

Las pantallas vinculadas a agregar un método de pago para realizar sus pedidos de forma automática permiten vincular una a más tarjetas de crédito a débito diferenciadas por el número de tarjeta y nombre único, el cuál es un alias puesto por el usuario. Su implementación hace posible un flujo más fluido para el usuario que compra o alquila productos. Como parámetros de entrada se consideran los necesarios para establecer comunicación con la pasarela de pagos de MercadoPago. Además, para dar como finalizado el guardado de su tarjeta, es indispensable que el usuario acepte los términos y condiciones puestos por la organización, documento que es accesible al público.



*Imagen: Pantalla para agregar métodos de pago (tarjeta bancaria)*

**Tabla de equivalencia:**

| Condición de entrada | Clases válidas | ID | Clases no válidas | ID |
| --- | --- | --- | --- | --- |
| Número de tarjeta | Valor numérico (NUMBER) de 16(dieciséis) dígitos. | 1 | Vacío | 8 |
|
| Dato no numérico | 9 |
|
| Menor a 0 | 10 |
|
| Nombre Completo | Cadena de caracteres (VARCHAR). | 2 | Vacío | 11 |
| Documento de identidad | Valor numérico (NUMBER) de 8(ocho) dígitos | 3 | Vacío | 12 |
| Dato no numérico | 13 |
| Menor a 0 y mayor a 8 | 14 |
| Expiración | Valor numérico (NUMBER) en formato MM/AA para la fecha de expiración. | 4 | Vacío | 15 |
| Dato no numérico | 16 |
| Menor a 0 | 17 |
| CVC | Valor numérico (NUMBER) de 3(tres) a 4(cuatro) dígitos. | 5 | Vacío | 18 |
| Dato no numérico | 19 |
| Fuera del rango | 20 |
| Nombre de la tarjeta (Alias) | Cadena de caracteres (VARCHAR). | 6 | Vacío | 21 |
| Acepta términos y condiciones | Booleano (BOOLEAN). | 7 | - |  |

**Casos de prueba:**

| Caso de prueba válido | | |
| --- | --- | --- |
| Prueba P001 | | |
| Objetivo | Agregar una tarjeta como método de pago | |
| Precondición | Ingresar a la aplicación como un usuario autorizado (comprador o vendedor) | |
| Descripción de la prueba | En la interfaz del sistema deberá introducir:   * Número de tarjeta * Nombre del usuario * Documento de identidad del usuario * Expiración de la tarjeta * CVC de la tarjeta * Nombre de la tarjeta o Alias * Términos y condiciones   Una vez estén todos los campos completados y que el usuario haya aceptado los términos y condiciones que aseguran la integridad de los datos ingresados, se podrá guardar la tarjeta como método de pago. | |
| Resultados esperados | Se añade la tarjeta como método de pago, por lo que el usuario podrá utilizarla en sus transacciones directamente. | |
| Caso de prueba inválido | | |
| Prueba P002 | | |
| Objetivo | No permitir agregar una tarjeta como método de pago | |
| Precondición | Ingresar a la aplicación como un usuario autorizado (comprador o vendedor) | |
| Descripción de la prueba | En la interfaz del sistema deberá introducir:   * Número de tarjeta * Nombre del usuario * Documento de identidad del usuario * Expiración de la tarjeta (no coincide con la tarjeta) * CVC de la tarjeta (no coincide con la tarjeta) * Nombre de la tarjeta o Alias * Términos y condiciones | |
| Resultados esperados | Se muestra un mensaje de “Los datos de la tarjeta no son válidos. Intente de nuevo.” | |

### Pantalla de Gestión de Historial de Transacciones

La vista representada en el mockup corresponde a la sección "Mi Historial" dentro de una aplicación móvil, destinada a la compra, venta y/o alquiler de materiales académicos. A continuación, se describe la estructura y funcionalidad de esta pantalla:

* Encabezado: El encabezado de la sección consta de 5 campos seleccionables que sirven de filtro para personalizar los resultados obtenidos en el historial de transacciones.
* Sección Principal: La sección principal de la vista tiene la función de visualizar los resultados relacionados a las transacciones realizadas por el usuario.
* Pie de página: La sección inferior de la vista tiene la función de mostrar atajos a las otras vistas principales y facilitar la navegación y usabilidad.



**Tabla de equivalencia :**

| Condición de entrada | Clases válidas | ID | Clases no válidas | ID |
| --- | --- | --- | --- | --- |
| Fecha Específica | Fecha en formato DD/MM/YYYY | 1 | Fecha incompleta. P.e : ‘dd/mm’ | 13 |
|
| Fecha con un formato externo. P.e : ‘MM/DD/YYYY’ | 14 |
| Selección de fecha en una ventana modal de calendario. | 2 |
| Caracteres alfabéticos a excepción de ‘/’ | 15 |
|
| Palabra Clave | Cadena vacía o de valores alfanuméricos con una longitud de hasta 20 | 3 | Carácteres especiales como ‘#’, ‘@’, ‘<’ , ‘>’, ‘?’ | 16 |
| Categoría | Selección: Compra | 4 | Ningún campo seleccionado | 17 |
| Selección: Alquiler | 5 |
| Tipo | Selección: Compras | 6 | Ningún campo seleccionado | 18 |
| Selección: Ventas | 7 |
| Material | Selección: Resúmenes | 8 |  |  |
| Selección: Evaluaciones | 9 |
| Selección: Libros | 10 |
| Selección: Electrónicos | 11 |
| Selección: Videos | 12 |

**Casos de prueba:**

| Caso de prueba válido | | |
| --- | --- | --- |
| Prueba P001 | | |
| Objetivo | Búsqueda por palabra clave | |
| Precondición | Ingresar a la aplicación como un usuario autorizado (comprador, vendedor o administrador) | |
| Descripción de la prueba | Verificar que el sistema permita realizar una búsqueda de transacciones a través de una palabra clave, recuperando correctamente todos los registros donde el autor o la descripción contengan dicha palabra. | |
| Resultados esperados | El sistema debe mostrar todas las transacciones cuyo campo autor o descripción contenga la palabra clave ingresada. | |
| Caso de prueba inválido | | |
| Prueba P002 | | |
| Objetivo | Filtrado por fecha | |
| Precondición | Ingresar a la aplicación como un usuario autorizado (comprador, vendedor o administrador) | |
| Descripción de la prueba | Verificar que el sistema detecte y rechace fechas ingresadas en un formato inválido, tanto si fueron seleccionadas desde una ventana modal de calendario como si fueron escritas manualmente por el usuario. Entrada:   * "31/13/2024” * "99/99/9999" * "12-05-2024" * "2024/04/31" * "15/abril/2024" | |
| Resultados esperados | El sistema no debe ejecutar la búsqueda si la fecha ingresada está en un formato inválido o representa una fecha imposible (por ejemplo, 31/02/2024).  Salida esperada:   * "Formato de fecha inválido. Use dd/mm/aaaa" * "La fecha ingresada no existe" | |

### Pantalla de Registrar reseñas por Producto :

**Añadir Reseña :** Esta pestaña está diseñada para facilitar el registro de reseñas y comentarios por parte del usuario, específicamente después de la adquisición de materiales académicos. Su objetivo es permitir a los usuarios compartir su experiencia y opinión sobre los productos comprados.



*Imagen: Registrar reseña de producto*

**Tabla de equivalencia :**

| Condición de entrada | Clases válidas | | Clases no válidas | |
| --- | --- | --- | --- | --- |
| Título | Cadena de valores alfanuméricos con una longitud máxima de hasta 20 carácteres | 1 | Vacío | 13 |
| Carácteres especiales como ‘#’, ‘@’, ‘<’ , ‘>’, ‘?’ | 14 |
| Carácteres con longitudes mayores a 20 | 15 |
| Comentario | Cadena de valores alfanuméricos con una longitud máxima de hasta 100 carácteres. | 2 | Vacío | 16 |
| Carácteres con longitudes mayores a 100 | 17 |
| Calificación | 0 estrellas (Sin calificar) | 3 |  | |
| 1 estrella | 4 |
| 2 estrellas | 5 |
| 3 estrellas | 6 |
| 4 estrellas | 7 |
| 5 estrellas | 8 |
| Categoría | Selección : Sugerencia | 9 | Ningún campo seleccionado | 18 |
| Selección : Opinión | 10 |
| Selección : Queja | 11 |
| Enviar reseña como usuario anónimo | Vacío  Seleccionado | 12 |  | |

**Casos de prueba :**

| Caso de prueba válido | | |
| --- | --- | --- |
| Prueba P003 | | |
| Objetivo | Permitir el registro de una reseña | |
| Precondición | Ingresar a la aplicación como un usuario autorizado (comprador o vendedor). Se presionó el botón de “agregar reseña”. | |
| Descripción de la prueba | En la interfaz se deberá completar los siguientes campos :   * Título : “El mejor material” * Comentario : “Me fue útil para practicar” * Calificación : 4 estrellas * Categoría : “Opinión” * Enviar reseña como usuario anónimo : False | |
| Resultados esperados | El sistema muestra el mensaje : “Reseña enviada con éxito” | |
| Caso de prueba inválido | | |
| Prueba P004 | | |
| Objetivo | No permitir el registro de una reseña por incumplir con los campos. | |
| Precondición | Ingresar a la aplicación como un usuario autorizado (comprador o vendedor). Se presionó el botón de “agregar reseña”. | |
| Descripción de la prueba | En la interfaz se deberá completar los siguientes campos :   * Título : “” * Comentario : “” * Calificación : 4 estrellas * Categoría : “Opinión” * Enviar reseña como usuario anónimo : False | |
| Resultados esperados | El sistema muestra el mensaje : “No se admiten campos vacíos” | |

### Pantalla de Nueva Publicación

Esta pantalla permite al usuario vendedor agregar los datos de una nueva publicación relacionado a un producto específico, esta puede ser para venta o alquiler.



*Imagen: Pantalla de nueva publicación*

**Tabla de equivalencia :**

| Condición de entrada | Clases válidas | ID | Clases no válidas | ID |
| --- | --- | --- | --- | --- |
| Nombre del producto | Cadena de entre 1 y 60 caracteres alfanuméricos | 1 | Vacío | 11 |
|
| Cadena de +60 caracteres | 12 |
|
| Cadena con caracteres no alfanuméricos | 13 |
|
| Descripción del producto | Cadena de entre 1 y 255 caracteres alfanuméricos | 2 | Cadena de +255 caracteres | 14 |
| Vacío | 15 |
| Cadena con caracteres no alfanuméricos | 16 |
| Archivo | Archivo JPG o PNG de menos de 15 MB | 3 | Vacío | 17 |
|
| Archivo no JPG ni PNG | 18 |
|
| Archivo mayor a 15 MB | 19 |
|
| Categoría | Selección: Libros | 4 | Vacío | 20 |
|
| Selección: Exámenes | 5 |
|
| Selección: Resúmenes | 6 |
| Categoría inválida | 21 |
| Selección: Electrónicos | 7 |
|
| Selección: Videos | 8 |
|
| Tipo de transacción | Selección: Alquiler | 9 |  |  |
| Selección: Vender | 10 |

**Casos de prueba:**

| Caso de prueba válido | | |
| --- | --- | --- |
| Prueba R001 | | |
| Objetivo | Publicar un artículo | |
| Precondición | Ingresar a la aplicación como un usuario autorizado (comprador, vendedor o administrador) | |
| Descripción de la prueba | 1. Abrir la pantalla de registro de nueva publicación. 2. Ingresar datos, por ejemplo:  * Nombre del producto: “Calculadora Casio XM-213” * Descripción del producto: “Calculadora gráfica, casi nueva, color azul y blanco.“ * Archivo: [Foto JPG de 3 MB]. * Categoría: “Electronicos” * Tipo de transacción: “Venta” | |
| Resultados esperados | El sistema debe permitir continuar con el proceso de nueva publicación con los datos establecidos por el usuario., mostrando así que los datos ingresados son correctos. | |
| Caso de prueba inválido | | |
| Prueba R002 | | |
| Objetivo | Publicar un artículo | |
| Precondición | Ingresar a la aplicación como un usuario autorizado (comprador, vendedor o administrador) | |
| Descripción de la prueba | 1. Abrir la pantalla de registro de nueva publicación. 2. Ingresar datos, por ejemplo:  * Nombre del producto: [] (Vacío) * Descripción del producto: “Calculadora gráfica, casi nueva, color azul y blanco.“ * Archivo: [Foto JPG de 3 MB]. * Categoría: [] (Vacío) * Tipo de transacción: “Venta” | |
| Resultados esperados | El sistema no debe permitir que el usuario progrese con el proceso de publicación de un artículo. Y debe solicitar al usuario corregir el o los campos correspondientes (nombre del producto y categoría) mediante un mensaje en la pantalla. | |

### 

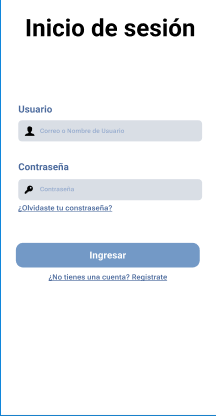
### Pantalla Registro de Usuarios

Estas pantallas de abajo hacen posible que un usuario nuevo se registre o que, si ya tenía cuenta, simplemente inicie sesión. Cuando hablamos de “usuario” nos referimos tanto a compradores como a vendedores, así que todos entran por el mismo lugar.

**Inicio de sesión**: aquí metes tu usuario (puede ser correo o tu nickname) y tu contraseña. Si la olvidaste, está el link “¿Olvidaste tu contraseña?” para que pidas un reseteo, o si aún no te registraste, un enlace que te lleva al formulario de registro.

**Registro de cuenta**: básicamente te pedimos tu Nombre y Apellido, tu Especialidad y en qué Ciclo Universitario vas, además de marcar tus Cursos de interés y Servicios de interés (normalmente escoges entre 1 y 5). Al darle “Ingresar” validamos todo y ¡listo!, ya quedas dentro con tu rol (comprador o vendedor) según elijas.

**Inicio de Sesión Registro de la Cuenta**



Estas pantallas permiten al usuario recuperar su contraseña de forma segura:

* **Restablecer contraseña**: introduce tu correo y pulsa el botón para que el sistema envíe un enlace temporal.
* **Verifica tu bandeja**: indica revisar tu email (inbox o spam) y usar ese enlace para definir una nueva clave.

**Para restablecer la contraseña**

****

**Tabla de Equivalencia:**

| Condición de entrada | Clases válidas | ID | Clases no válidas | ID |
| --- | --- | --- | --- | --- |
| Usuario (login) | Correo local@dominio.ext **o** usuario alfanumérico (5–15 car.). | 1 | * Vacío. | 8 |
| * Con símbolos especiales. | 9 |
| * < 5 carácteres. | 10 |
| * > 15 carácteres. | 11 |
| Contraseña (login) | 8–20 car., al menos una letra y un número. | 2 | * < 8 caracteres. | 12 |
| * Solo letras. | 13 |
| * Solo números. | 14 |
| Nombre y Apellido | 3–50 car., solo letras y espacios. | 3 | * < 3 caracteres. | 15 |
| * > 50 caracteres. | 16 |
| * Con números o símbolos. | 17 |
| Especialidad | 3–30 car., solo letras y espacios. | 4 | * Vacío. | 18 |
| * > 30 caracteres. | 19 |
| * Con números o símbolos. | 20 |
| Ciclo universitario | Entero entre 1 y 12. | 5 | * 0 ó mayor a 12. | 21 |
| * Número con decimales. | 22 |
| Cursos de interés | Selección de cursos del listado predefinido. | 6 | * Selección nula ( < 1). | 23 |
| Correo (reset) | Correo local@dominio.ext. | 7 | * Vacío. | 24 |
| * Sin arroba. | 25 |
| * Sin dominio. | 26 |

**Casos de Prueba**

| Caso de prueba válido | | |
| --- | --- | --- |
| Prueba R001 | | |
| Objetivo | Registro satisfactorio de un nuevo usuario. | |
| Precondición | El usuario no está autenticado ni existe en el sistema. | |
| Descripción de la prueba | 1. Abrir la pantalla de registro.  2. Ingresar por ejemplo:   * Nombre y Apellido: “María López”. * Especialidad: “Ingeniería”. * Ciclo Universitario: “5”. * Cursos de interés: [“Programación”, “Matemáticas”]. * Servicios de interés: [“Chat interno”, “Alertas”]. | |
| Resultados esperados | Se muestra notificación de “Registro exitoso”. | |

| Caso de prueba inválido | | |
| --- | --- | --- |
| Prueba R002 | | |
| Objetivo | Verificar manejo de datos inválidos en el formulario de registro. | |
| Precondición | El usuario no está autenticado ni existe en el sistema. | |
| Descripción de la prueba | 1. Abrir la pantalla de registro.  2. Ingresar:   * Nombre y Apellido: “” (vacío). * Especialidad: “Ing3niería!” (números y símbolo). * Ciclo Universitario: “0” (fuera de rango). * Cursos de interés: [] (ninguno). * Servicios de interés: [] (ninguno).   3. Pulsar **Ingresar**. | |
| Resultados esperados | – El formulario no se envía ni crea la cuenta. | |