

Universidad Distrital Francisco José de  
Caldas

FACULTAD DE INGENIERÍA

# DISEÑO DE SOFTWARE

*Proyecto Final Fundamentos Ingeniería de Software*

Autores:  
Christian Galindo y Guillermo Vélez

Febrero 2021



# Índice general

<b>I</b>	<b>Proyecto</b>	<b>7</b>
<b>1.</b>	<b>Proyecto</b>	<b>9</b>
1.1.	Introducción . . . . .	9
1.2.	Descripción . . . . .	10
1.3.	Objetivos . . . . .	11
1.4.	Alcance y Límites . . . . .	12
1.4.1.	Alcance . . . . .	12
1.4.2.	Límites . . . . .	12
1.5.	Metodología . . . . .	13
1.6.	Cronograma . . . . .	14
<b>2.</b>	<b>Lenguaje</b>	<b>15</b>
2.1.	UML . . . . .	15
2.1.1.	Introducción . . . . .	15
2.1.2.	Abstracción . . . . .	16
2.1.2.1.	Paradigma Orientado a objetos . . . . .	16
2.1.2.2.	Paradigma Estructurado . . . . .	17
2.1.3.	Programación Orientada a Objetos . . . . .	17
2.1.3.1.	Principios . . . . .	17
2.1.3.2.	Encapsulamiento . . . . .	17
2.1.3.3.	Ocultación de la información . . . . .	18
2.1.3.4.	Paso de mensajes . . . . .	18
2.1.3.5.	Delegación . . . . .	18
2.1.3.6.	Atado posterior . . . . .	18
2.1.3.7.	Instanciación . . . . .	18
2.1.3.8.	Generalización con o sin polimorfismo . . . . .	19
2.1.3.9.	Asociación . . . . .	19
2.1.4.	Casos de uso . . . . .	19
2.1.5.	Estructura . . . . .	21

<b>II</b>	<b>Diseño</b>	<b>25</b>
<b>3.</b>	<b>Casos</b>	<b>27</b>
3.1.	Diagrama . . . . .	27
3.2.	Descripción casos de uso . . . . .	28
<b>4.</b>	<b>Interacciones</b>	<b>41</b>
4.1.	Diagramas de secuencia . . . . .	41
4.2.	Diagramas de comunicación . . . . .	48
<b>5.</b>	<b>Clases</b>	<b>53</b>
5.1.	Relación de juegos con pedido . . . . .	53
5.2.	Relación de pedido con usuario . . . . .	54
<b>6.</b>	<b>Sistema</b>	<b>55</b>
6.1.	Introducción . . . . .	55
<b>7.</b>	<b>Estados</b>	<b>57</b>
7.1.	Estados de sesión . . . . .	57
7.2.	Estados de juego en carrito . . . . .	58
<b>8.</b>	<b>Componentes</b>	<b>61</b>
8.1.	Introducción . . . . .	61
<b>9.</b>	<b>Nodos</b>	<b>63</b>
9.1.	Introducción . . . . .	63
<b>III</b>	<b>Cierre</b>	<b>65</b>
<b>10.</b>	<b>Conclusiones</b>	<b>67</b>
<b>11.</b>	<b>Anexos</b>	<b>69</b>

# Índice de figuras

1.1. Comportamiento Sistema . . . . .	10
1.2. Ciclo de vida ASD[1] . . . . .	14
1.3. Cronograma . . . . .	14
3.1. Casos de uso ecommerce . . . . .	27
4.1. Añadir categoría . . . . .	41
4.2. Añadir producto . . . . .	42
4.3. Gestionar datos administrador . . . . .	42
4.4. Gestionar datos cliente . . . . .	43
4.5. Pagar . . . . .	43
4.6. Eliminar categoría . . . . .	44
4.7. Eliminar producto . . . . .	44
4.8. Comprar . . . . .	45
4.9. Gestión carrito de compra . . . . .	45
4.10. Consultar catálogo . . . . .	46
4.11. Dar baja . . . . .	46
4.12. Registrar . . . . .	47
4.13. Validar usuario . . . . .	47
4.14. Añadir categoría . . . . .	48
4.15. Añadir producto . . . . .	48
4.16. Gestionar datos administrador . . . . .	48
4.17. Gestionar datos cliente . . . . .	48
4.18. Pagar . . . . .	48
4.19. Eliminar categoría . . . . .	49
4.20. Eliminar producto . . . . .	49
4.21. Comprar . . . . .	50
4.22. Gestión carrito de compra . . . . .	50
4.23. Consultar catálogo . . . . .	50

4.24. Dar baja . . . . .	51
4.25. Registrar . . . . .	51
4.26. Validar usuario . . . . .	51
5.1. Diagrama de clases primera parte . . . . .	53
5.2. Diagrama de clases segunda parte . . . . .	54
6.1. Diagrama Sistemas . . . . .	55
7.1. Diagrama de estados sesión . . . . .	57
7.2. Ejemplo de clase representativa de los estados de sesión . . .	57
7.3. Diagrama de estados de juegos en carrito . . . . .	58
7.4. Ejemplo de clase representativa de los estados de juego en carrito . . . . .	59
8.1. Diagrama Componentes . . . . .	62
9.1. Diagrama Nodos . . . . .	63

# Parte I

## Proyecto





# Capítulo 1

## Proyecto

### 1.1. Introducción

El proyecto consiste en la elaboración de una página web que implementa el comercio electrónico o e-commerce como es mayormente conocido, en el ámbito del mercado de los videojuegos. Para el desarrollo backend se utiliza un lenguaje core Java haciendo uso del framework Spring Boot, y para el desarrollo frontend (HTML, CSS y Scripting) se utiliza el framework Angular.

Esto se realizará con el fin de generar una alternativa a las plataformas que existen actualmente como Steam, Epic Games Store, Origin, Uplay, entre otras. Además de ofrecer un catálogo diverso tanto en consolas como en géneros de videojuegos.

## 1.2. Descripción

El proyecto tiene tres grandes componentes: Una base de datos, una aplicación Spring Boot y la aplicación Angular. La base de datos guarda la información de la tienda online, como lo es la información de los clientes, pedidos y productos además de realizar las funciones básicas CRUD. El sistema de gestión de base de datos utilizado es MySQL.

La aplicación Spring Boot (framework para desarrollo de aplicaciones java), es la intermediaria entre el sistema gestor de base de datos y la aplicación Angular. Se encarga de consultar la base de datos para obtener la información que se necesita mostrar en el lado del cliente. Además de procesar las peticiones que la aplicación Angular realice, como las funciones básicas CRUD relacionadas con clientes, administradores o productos.

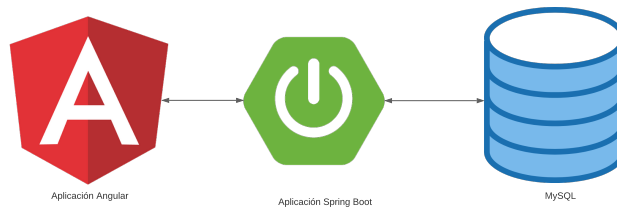


Figura 1.1: Comportamiento Sistema

Por último la aplicación Angular (framework para desarrollo de aplicaciones web, su principal lenguaje de programación es TypeScript), es la encargada de la interacción con el usuario, esta se encarga de desplegar la interfaz gráfica y atender las peticiones, comunicándolas a la aplicación Spring Boot. Luego de recibir una respuesta del lado servidor, la aplicación Angular procesa dicha respuesta para mostrarla en el lado cliente, obteniendo así una buena interacción con el usuario.

## 1.3. Objetivos

### Objetivo General

- Elaborar una página web que genere variabilidad en cuanto a las plataformas dedicadas al mercado de los videojuego, enfocado en basarse en las recomendaciones del usuario.

### Objetivos Específicos

- Generar un módulo de administración fácil de usar, donde se puedan agregar, modificar y eliminar, categorías, juegos y usuarios.
- Ofrecer la posibilidad de crear descuentos para los diferentes juegos en cualquier momento, donde el administrador decidirá el valor de este.
- Dar posibilidad al usuario de consultar sus pedidos a través de la plataforma luego de haber iniciado sesión.
- La plataforma no se limitara exclusividades en cuanto a los juegos, ofrecerá juegos pertenecientes a cualquier compañía.

## 1.4. Alcance y Límites

### 1.4.1. Alcance

- El trabajo se enfocara en la realización de la plataforma.
- Se podrán generar token's para el control de sesiones dentro de la plataforma.
- La plataforma permitirá concurrencia, por lo que se puede realizar cualquier transacción en simultáneo desde diferentes sesiones.
- Se generarán distintos roles para los usuarios, de modo que los usuarios no administradores no puedan acceder de ninguna manera al módulo de administración (al cual tendrán acceso los usuarios con rol de administrador).

### 1.4.2. Límites

- La plataforma no se implementara de manera comercial.
- Los juegos presentados no podrán ser descargados por problemas de derechos de autor.
- Los pagos Bancarios no podrán realizarse.
- Las categorías de juegos presentaran poca cantidad de juegos a la hora de ser presentadas.
- No se podrán realizar comentarios acerca del producto por parte de los usuarios.

## 1.5. Metodología

El modelo de implementación para el desarrollo del proyecto es la metodología ágil ASD (Adaptive Software Development). Con ella se busca la fácil adaptabilidad del equipo de manera rápida y eficaz ante los posibles requisitos cambiantes, mediante una planificación ligera y aprendizaje continuo.

Esta metodología se enfoca en un desarrollo iterativo, orientado principalmente a los componentes de software y su función, tolerante a los cambios y guiado por los riesgos. Por otra parte la revisión de los componentes permite aprender de los errores y volver a iniciar el ciclo de desarrollo.

El ciclo de vida se divide en 3 etapas:

### 1. Especulación

- a) Inicio para determinar la misión del proyecto.
- b) Fijación del marco temporal del proyecto.
- c) Determinación del n<sup>o</sup> de iteraciones y la duración de cada una.
- d) Definición del objetivo de cada iteración.
- e) Asignación de funcionalidad a cada iteración.

### 2. Colaboración

- Desarrollo concurrente del trabajo de construcción y gestión del producto

### 3. Aprendizaje

En cada iteración se revisa:

- Calidad, con criterios de cliente.
- Calidad, con criterios técnicos.
- Funcionalidad desarrollada.
- Estado del proyecto.

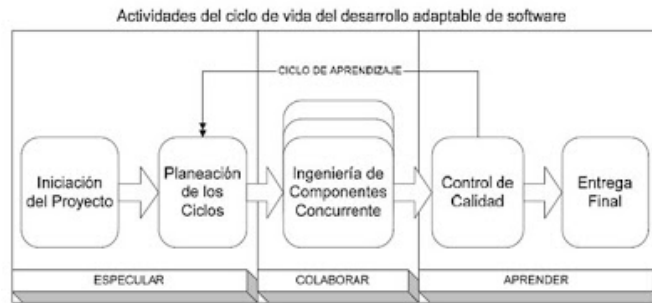


Figura 1.2: Ciclo de vida ASD[1]

## 1.6. Cronograma

CRONOGRAMA									
ETAPA/SEMANA	1	2	3	4	5	6	7	8	9
Especulación	X	X							
Colaboración			Modulo Admin	Modulo Registro Usuario		Modulo Catálogo	Modulo Compra		Entrega
Aprendizaje					X			X	

Figura 1.3: Cronograma

## Capítulo 2

# Lenguaje

### 2.1. UML

#### 2.1.1. Introducción

(UML: Unified Modeling Language) es una herramienta que permite modelar software orientado a objetos a través de un amplio vocabulario gráfico enfocado a la representación conceptual y física de los sistemas de software. Actualmente es un estándar adoptado por el grupo de desarrollo en objetos(OMG:Object Management Group) [2].

Este vocabulario gráfico está conformado por cosas, relaciones y diagramas. Dentro de las cosas encontramos de tipo estructural, de comportamiento, de agrupación y de anotación. Las relaciones se dividen en cliente/proveedor y generalización/implementación. Por últimos están los diagramas que pueden ser de comportamiento o estructurales.

Mediante los componentes anteriormente descritos se representan los límites, la estructura y el comportamiento del sistema y los objetos que contiene.

### 2.1.2. Abstracción

Consiste en buscar generalidades que ofrezcan una mejor perspectiva del programa. En la programación tenemos diferentes niveles de abstracción:

- Lenguaje de alto nivel: El compilador es el encargado de traducir el código a maquina.
- Lenguaje de ensamble: Un programa de ensamble traduce la sentencias a código de maquina.
- Control: Las instrucciones que se reciban son interpretadas para poder realizar las operaciones sobre los datos.
- Unidades Funcionales: Unidades como Registros o unidades de almacenamiento.
- Compuertas y Transistores: Las compuertas son encargadas de realizar operaciones lógicas fundamentales. Las unidades funcionales se construyen a partir de compuertas, y estas a su vez son hechas con transistores.

#### 2.1.2.1. Paradigma Orientado a objetos

El paradigma orientado a objetos OO define los programas en términos de comunidades de objetos. Los objetos con características comunes se agrupan en clases (un concepto similar al de tipo abstracto de dato (TAD)). Los objetos son entidades que combinan un estado (es decir, datos) y un comportamiento (esto es, procedimientos o métodos). Estos objetos se comunican entre ellos para realizar tareas. Es en este modo de ver un programa donde este paradigma difiere del paradigma imperativo o estructurado, en los que los datos y los métodos están separados y sin relación. El paradigma OO surge para solventar los problemas que planteaban otros paradigmas, como el imperativo, con el objeto de elaborar programas y módulos más fáciles de escribir, mantener y reutilizar. Entre los lenguajes que soportan el paradigma OO están Smalltalk, C++, Delphi (Object Pascal), Java y C



#### 2.1.2.2. Paradigma Estructurado

La programación estructurada es un paradigma de programación basado en utilizar funciones o subrutinas, y únicamente tres estructuras de control:

- secuencia: ejecución de una sentencia tras otra.
- selección o condicional: ejecución de una sentencia o conjunto de sentencias, según el valor de una variable booleana.
- iteración (ciclo o bucle): ejecución de una sentencia o conjunto de sentencias, mientras una variable booleana sea verdadera.

Este paradigma se fundamenta en el teorema correspondiente, que establece que toda función computable puede ser implementada en un lenguaje de programación que combine sólo estas tres estructuras lógicas o de control. además de permitir hacer un código mas fácil de entender y no hay necesidad de buscar los saltos de línea (Go to) porque estas se encuentran fuertemente ligadas o relacionadas así

#### 2.1.3. Programación Orientada a Objetos

El UML se estableció como un modelo estandarizado para describir un enfoque de programación orientada a objetos (POO). Es por ello que se hace una breve descripción de este paradigma de programación, estableciendo su lugar en los niveles de abstracción y sus principios.

##### 2.1.3.1. Principios

La programación orientada a objetos al ser un paradigma, tiene características que le permiten un manejo óptimo de las estructuras de programación, entre las cuales tenemos: encapsulamiento, ocultación de información, paso de mensajes, delegación, atado posterior, la instanciación, generalización y asociación

##### 2.1.3.2. Encapsulamiento

El encapsulamiento es la característica que busca el ordenamiento de los elementos obtenidos por medio de la abstracción, los cuales

constituyen su estructura. fundamentalmente este ordenamiento busca mantener los datos protegidos en una estructura mayor; la cual tiene como finalidad bloquear el acceso directo a sus datos internos, por medio de otro ente externo que se encuentre en el programa.

#### 2.1.3.3. Ocultación de la información

Consiste en aislar un objeto de tal modo que se puedan proteger sus propiedades, de tal forma que ninguno que no tenga acceso a ellas, no las pueda modificar. Esto implica que cada objeto debe ser independiente y no pueden generar cambios en los estados internos de otros objetos.

#### 2.1.3.4. Paso de mensajes

Es una característica también aplicada en la programación concurrente, su finalidad es invocar de forma abstracta el comportamiento concreto de un actor. Esto permite principalmente que se de un mecanismo que permite distinguir su función de su implementación, además de permitir que se de el encapsulamiento de manera correcta. lo que a su vez significa un código mas limpio.

Esto se da mas claro cuando se manejan sistemas sincrónicos, con el paso de mensajes, se facilita

#### 2.1.3.5. Delegación

Un objeto puede ser implementado de forma que delegue parte de su funcionalidad en otro objeto. Esto es utilizado para generar una mayor flexibilidad en el diseño del software

#### 2.1.3.6. Atado posterior

También se conoce como ligadura dinámica y consiste en un mecanismo que determina la versión del método a ejecutar según la forma dinámica que tenga el objeto.

#### 2.1.3.7. Instanciación

Es la característica que permite la creación de miembros procedentes de una clase, estos miembros son llamados instancias. Se

llama instancia a todo objeto que derive de algún otro. De esta forma, todos los objetos son instancias de algún otro, menos la clase Object que es la madre de todas.

#### 2.1.3.8. Generalización con o sin polimorfismo

Esta característica se puede dividir en dos partes la primera aclara que una clase de objetos puede definirse como subconjunto de alguna otra clase, este subconjunto debería constituir siempre un conjunto de objetos similares. Hablamos entonces de subclases de otras clases que, por tanto, constituyen especializaciones de esas otras clases.

La segunda parte es la generalización la cual es la relación inversa a la especialización. Si una clase es una especialización de otra clase, ésta última es una generalización de la primera. Es decir su superclase.

#### 2.1.3.9. Asociación

Es una relación de estructura entre clases, es decir, una entidad se construye a partir de otra u otras. Aunque este tipo de relación es mas fuerte que la Dependencia es más débil que la Agregación, ya que el tiempo de vida de un objeto no depende de otro.

#### 2.1.4. Casos de uso

El modelado de Casos de uso es la técnica más efectiva y a su vez la mas simple para el moldeamiento de los requisitos de un sistema, manejando la perspectiva del usuario. Los casos de usos se utilizan para describir el funcionamiento actual de un sistema, o en su defecto modela el el funcionamiento que el usuario desea que este tenga. No es realmente una aproximación a la orientación a objetos; es realmente una forma de modelar procesos. Es, sin embargo, una manera muy buena de dirigirse hacia el análisis de sistemas orientado a objetos. Los casos de uso son generalmente el punto de partida del análisis orientado a objetos con UML.

El modelo de casos de uso consiste en actores y casos de uso. Los actores representan usuarios y otros sistemas que interaccionan

con el sistema. Los casos de uso representan el comportamiento del sistema, los escenarios que el sistema atraviesa en respuesta a un estímulo desde un actor. Se dibujan como elipses cada una de las acciones y dentro de ellas se nombra la acción a realizar, estas elipses se unen con flechas las cuales indican su relación de dependencia o de asociación entre ellas.

**2.1.5. Estructura**














Cosas			
Tipo	Cosa	Descripción	Gráfico
Estructural	Clase	Plantilla para la creación de objetos.	
	Interface	Conjunto de métodos abstractos y de constantes cuya funcionalidad es la de determinar el funcionamiento de una clase.	
	Colaboración	diagrama que muestra interacciones organizadas alrededor de los roles	
	Caso de uso	Un caso de uso es la descripción de una acción o actividad	
	Componente	Parte modular de un sistema que encapsula su contenido y cuya manifestación se puede sustituir en su entorno.	
	Artefacto	Especificación de un componente físico de información.	
	Nodo	Representa los elementos básicos de software o hardware, en el sistema.	
Comportamiento	Transición	Relación entre dos o muchos estados que están unidos por una flecha.	
	Estado	Condición o situación en la vida de un objeto durante la cual satisface alguna condición.	
	Actividad	Comportamiento que supedita las unidades subordinadas (acciones y objetos) a una secuencia cronológica.	
Agrupación	Sistema	Encapsula los sistemas que intervienen en una acción	
	Frame	Encapsula una colección de instancias colaboradoras o se refiere a otra representación de las mismas.	
Anotación	Nota	Sirve para añadir cualquier tipo de comentario a un diagrama o a un elemento de un diagrama.	

Tabla 2.2: Tabla de cosas







Relaciones			
Tipo	Cosa	Descripción	Gráfico
Cliente/Proveedor	Dependencia	Relación de uso entre dos clases (una usa a la otra).	
	Asociación	Relación que describe un conjunto de vínculos entre clases.	
	Agregación	Tipo de asociación que indica que una clase es parte de otra clase (composición débil).	
	Composición	Forma fuerte de composición donde la vida de la clase contenida debe coincidir con la vida de la clase contenedor.	
Generalización/	Generalización	Es una relación entre dos clases de modo que una, la subclase o clase hija, se considera como forma especializada (refinada) de la otra, la superclase o clase padre. La superclase se considera como generalización de la subclase.	
Implementación	Realización	Es una relación entre dos clases de modo que una, la subclase o clase hija, se considera como forma especializada (refinada) de la otra, la superclase o clase padre. La subclase se considera una especialización de la superclase..	

Tabla 2.4: Tabla de relaciones

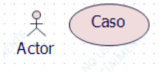
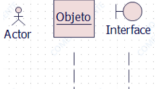












Diagramas			
Tipo	Cosa	Descripción	Gráfico
Comportamiento	Casos de uso	Relacionan un actor y los sistemas con los que se relaciona	
	Secuencia	Se centra específicamente en líneas de vida o en los procesos y objetos que coexisten simultáneamente.	
	Comunicación	Se utilizan para mostrar cómo interactúan los objetos para efectuar el comportamiento de un guión de uso concreto, o una parte de un guión de uso.	
	Temporización	Es utilizado para modelar el comportamiento del sistema dando especial importancia al tiempo.	
	Estados	Usado para especificar el comportamiento de una parte del sistema diseñado a través de transiciones de estados finitos.	
	Actividades	Muestra el flujo de control, con especial énfasis en la secuencia y las condiciones de este flujo.	
	Vista conjunta de interacción	muestra el flujo de control de la interacción a alto nivel	
Estructural	Sistemas	contiene clases, interfaces, paquetes y sus relaciones	
	Clases	Se utiliza para representar los elementos que componen un sistema de información desde un punto de vista estático.	
	Componentes	Se utilizan para modelar los componentes que ayudan a hacer esas funcionalidades.	
	Estructura Compuesta	Muestra la estructura interna (incluidas las partes y los conectores) de un clasificador estructurado.	
	Objeto	Representa una instancia de un diagrama de clase	
	Artefacto	Especificación de un componente físico de información que es usado o producido por un proceso	
	Emplazamiento	Se utiliza para modelar la disposición física de los artefactos software en nodos	

Tabla 2.5: Tabla de diagramas



Parte II

Diseño



# Capítulo 3

## Casos

### 3.1. Diagrama

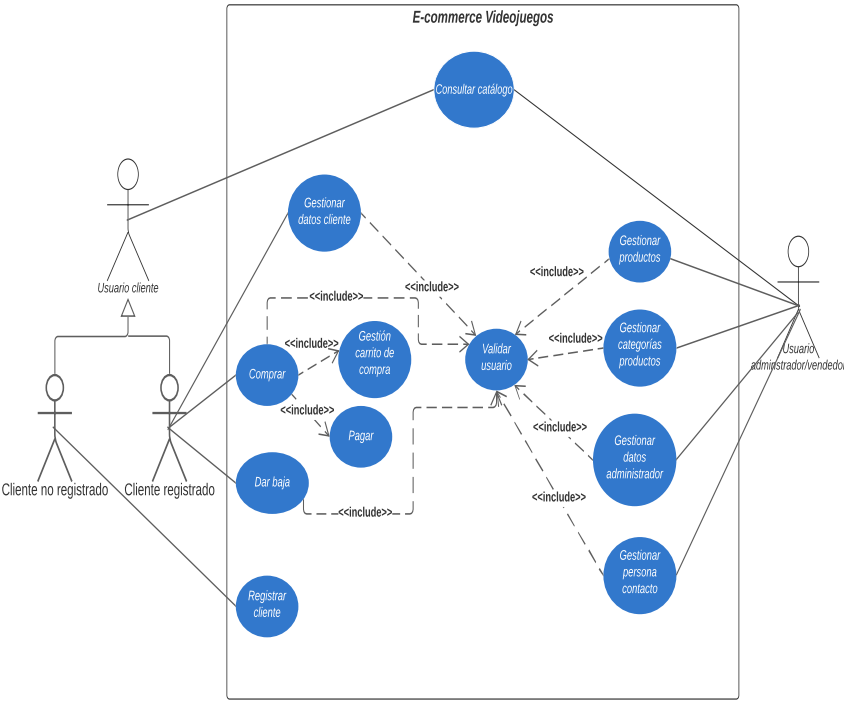


Figura 3.1: Casos de uso ecommerce

### 3.2. Descripción casos de uso

Caso de uso		
Nombre		Consultar catálogo
Autores		Luis Guillermo Vélez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Usuario/cliente-Administrador
Descripción		Se visualiza el catalogo de juegos correspondiente a una categoría seleccionada
Precondición		El usuario/cliente previamente debe tener conexión a internet
Secuencia Normal	Paso	Acción
	1	El usuario selecciona la categoría de juegos que desea consultar
	2	El sistema muestra a los usuarios el catálogo respectivo
Poscondición		Se guardan las preferencias de usuario si se encuentra registrado
Excepciones	Paso	Acción
	1	El sistema no muestra resultados por error de conexión
Importancia		Alta
Casos de uso de extensión		

Tabla 3.1: Consultar Catálogo

Caso de uso		
Nombre		Registrar Cliente
Autores		Luis Guillermo Vélez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Usuario no registrado
Descripción		El usuario no registrado Ingresa información básica la cual se validará y se guardará en la base de datos
Precondición		El usuario no registrado, no debe poseer alguna cuenta preexistente y debe ingresar información verídica
Secuencia Normal	Paso	Acción
	1	El usuario selecciona la pestaña de registro
	2	El sistema solicita al usuario información personal para ser almacenada en la base de datos
	3	El usuario ingresa la información solicitada
	4	El sistema verifica si la información es correcta o no pertenece a alguna otra cuenta
	5	El sistema envía la confirmación al usuario de la nueva asignación de cuenta
Poscondición		Se registra la información en la base de datos
Excepciones	Paso	Acción
	1	La información suministrada por el usuario es errónea
	2	La información suministrada por el usuario pertenece a alguna otra cuenta
Importancia		Media
Casos de uso de extensión		

Tabla 3.2: Registrar Cliente

Caso de uso		
Nombre		Dar baja
Autores		Luis Guillermo Velez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Cliente registrado
Descripción		El Cliente registrado decide el momento de retirar su información de la base de datos
Precondición		El usuario debe estar registrado
Secuencia Normal	Paso	Acción
	1	El usuario selecciona la pestaña de información personal
	2	El usuario confirma la eliminación de su información del sistema
	3	El sistema busca el registro de los datos perteneciente al cliente
Poscondición		Se borra la información relacionada al usuario de la base de datos
Excepciones	Paso	Acción
	1	La información de validación suministrada por el usuario es errónea
Importancia		Baja
Casos de uso de inclusión		Validar usuario

Tabla 3.3: Dar baja

Caso de uso		
Nombre		Comprar
Autores		Luis Guillermo Vélez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Cliente registrado
Descripción		El Cliente registrado realiza la adquisición de uno o varios productos
Precondición		El usuario debe estar registrado- el usuario debe tener saldo existente suficiente para costear el pedido
Secuencia Normal	Paso	Acción
	1	El usuario mediante el catálogo busca el juego que desea comprar
	2	El sistema muestra información básica para la compra como lo es el precio y la cantidad a comprar
	3	El sistema redirige al cliente a la confirmación de pago
	4	El cliente confirma el pago
Poscondición		Se guarda el pedido realizado por el cliente y sus detalles en la base de datos
Excepciones	Paso	Acción
	1	El cliente no cuenta con el dinero suficiente para realizar el pago
	2	Error durante la compra
Importancia		Alta
Casos de uso de inclusión		Gestion carrito de compra Pagar Validar usuario

Tabla 3.4: Comprar

Caso de uso		
Nombre		Gestionar datos cliente
Autores		Luis Guillermo Vélez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Cliente registrado
Descripción		El Cliente registrado realiza cambios en su información.
Precondición		El usuario debe tener una cuenta activa.
Secuencia Normal	Paso 1	Acción El usuario inicia sesión
	2	El usuario abre la pestaña de información personal.
	3	El usuario realiza la actualización o consulta de datos
Poscondición		Se realiza cambio de información o consulta de datos
Excepciones	Paso 1	Acción El cliente no cambia ningún dato
	2	Error de conexión con la base de datos
Importancia		Alta
Casos de uso de inclusión		Validar usuario

Tabla 3.5: Gestionar cliente



Caso de uso		
Nombre		Gestión Carro de Compra
Autores		Luis Guillermo Vélez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Cliente registrado
Descripción		El Cliente decide la cantidad de productos y cuales va a comprar en una transacción
Precondición		El usuario debe ingresar al carrito de compra algún producto
Secuencia Normal	Paso 1	Acción El usuario ingresa al carrito de compra
	2	El usuario cambia las cantidades deseadas o elimina algún producto del carrito
	3	El usuario finaliza la compra y se dirige al módulo de pago
Poscondición		Se realiza cambio de información de los productos a comprar
Excepciones	Paso 1	Acción El cliente no tiene ningún producto
Importancia		Alta
Casos de uso de inclusión		Validar usuario

Tabla 3.6: Gestión Carro de Compra

Caso de uso		
Nombre		Pagar
Autores		Luis Guillermo Vélez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Cliente registrado
Descripción		El Cliente confirma la cantidad de y valor de los productos
Precondición		El usuario tiene productos en el carrito de compra y cuenta con el saldo suficiente para adquirirlos.
Secuencia Normal	Paso	Acción
	1	El usuario visualiza cantidad y valor de los productos en el carrito de compra.
	2	El usuario ingresa la información correspondiente a la tarjeta
	3	el usuario confirma la cantidad a pagar
	4	El sistema valida la información
Poscondición		Se realiza el pago
Excepciones	Paso	Acción
	1	El cliente no cuenta con el valor suficiente de pago
	2	No se encuentran productos en el carrito
Importancia		Alta
Casos de uso de inclusión		Validar usuario

Tabla 3.7: Pagar

Caso de uso		
Nombre		Validar usuario
Autores		Luis Guillermo Vélez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Cliente registrado-administrador
Descripción		Se verifica que la información suministrada coincida con la información de la base de datos
Precondición		El usuario intento acceder a algún módulo de la página donde se requiere autenticación
Secuencia Normal	Paso 1	Acción El usuario ingresa datos
	2	El sistema valida que la información ingresada coincida con la que se encuentra en la base de datos
	3	El usuario es redirigido a donde deseaba ingresar
Poscondición		El sistema valida la información
Excepciones	Paso	Acción
	1	La información suministrada no coincide con la de la base de datos
Importancia		Alta
Casos de uso de extensión		

Tabla 3.8: Validar usuario

Caso de uso		
Nombre		Gestionar productos
Autores		Luis Guillermo Vélez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Administrador
Descripción		El administrador crea nuevas rutas y entradas del catálogo o las elimina
Precondición		Debe ser administrador y tener una sesión iniciada
Secuencia Normal	Paso	Acción
	1	El administrador selecciona la opción de ingresar o retirar contenido
	2	selecciona la categoría del el producto y sus especificaciones
	3	carga la información a la base de datos
	4	f
Poscondición		El sistema ingresa o retira un producto respetando las especificaciones del administrador
Excepciones	Paso	Acción
	1	Error en el almacenamiento de de información
Importancia		Alta
Casos de uso de inclusión		validar Usuario

Tabla 3.9: Gestionar productos

Caso de uso		
Nombre		Gestionar categorías productos
Autores		Luis Guillermo Vélez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Administrador
Descripción		El administrador crea nuevas rutas y entradas del catálogo o las elimina
Precondición		Debe ser administrador y tener una sesión iniciada
Secuencia Normal	Paso	Acción
	1	El administrador selecciona la opción de administrar categorías
	2	El administrador busca los juegos y les dicta las distintas etiquetas que puede tener por categorías
	3	Carga la información a la base de datos
Poscondición		El sistema guarda el juego en distintas categorías
Excepciones	Paso	Acción
	1	Error en el almacenamiento de de información
Importancia		Alta
Casos de uso de inclusión		Validar usuario

Tabla 3.10: Gestionar categorías productos

Caso de uso		
Nombre		Gestionar datos administrador
Autores		Luis Guillermo Vélez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Administrador
Descripción		Se agrega o actualiza información del administrador en la plataforma
Precondición		Debe ser administrador
Secuencia Normal	Paso	Acción
	1	El Administrador inicia sesión
	2	El administrador se dirige a información personal
	3	El administrador ingresa la información que desea actualizar
Poscondición		Se realiza actualización o inserción de información
Excepciones	Paso	Acción
	1	El Administrador no cambia ningún dato.
	2	Error de conexión con la base de datos
Importancia		Alta
Casos de uso de inclusión		Validar usuario

Tabla 3.11: Gestionar datos administrador

Caso de uso		
Nombre		Gestionar persona contacto
Autores		Luis Guillermo Vélez, Christian Yesid Galindo, Camilo Andrés Caimán
Fecha		18/11/2020
Actores		Administrador
Descripción		Se gestionan los datos relacionados a los clientes (funciones CRUD)
Precondición		Iniciar sesión
Secuencia Normal	Paso	Acción
	1	Ir a módulo de clientes
	2	Seleccionar el cliente sobre el cual se efectuará una operación
	3	Confirmar cambio de datos
Poscondición		Se actualizan los cambios realizados en la base de datos
Excepciones	Paso	Acción
	1	Error en el cambio de datos
	2	Error de conexión con la base de datos
Importancia		Alta
Casos de uso de inclusión		Validar usuario

Tabla 3.12: Gestionar persona contacto





## Capítulo 4

# Interacciones

### 4.1. Diagramas de secuencia

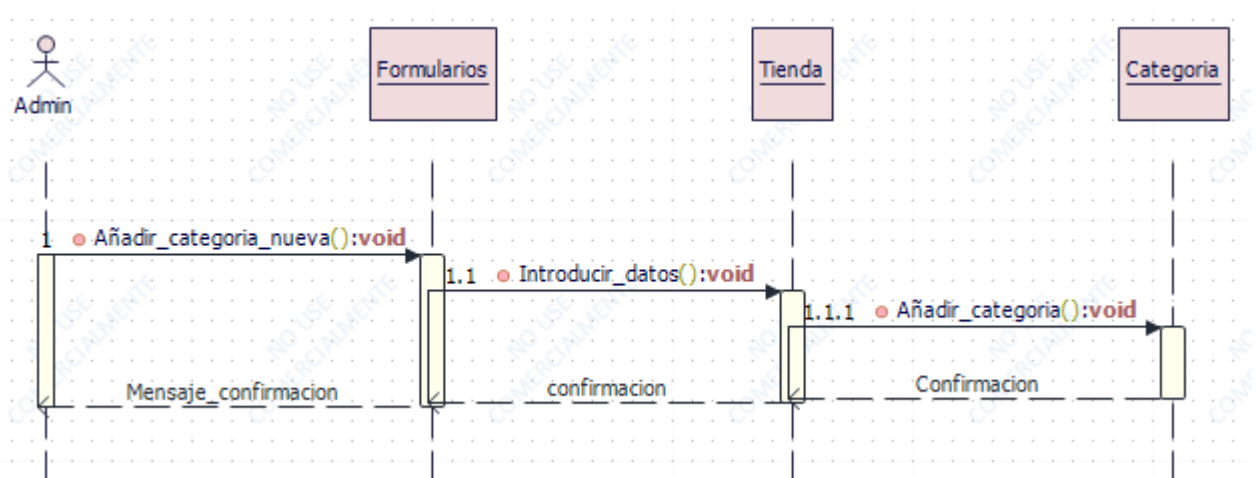


Figura 4.1: Añadir categoría

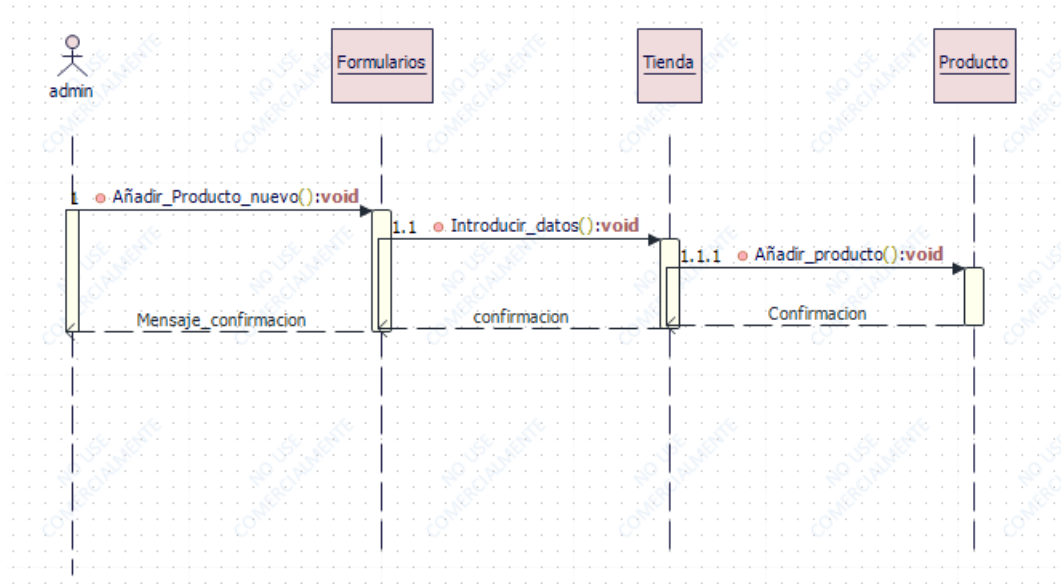


Figura 4.2: Añadir producto

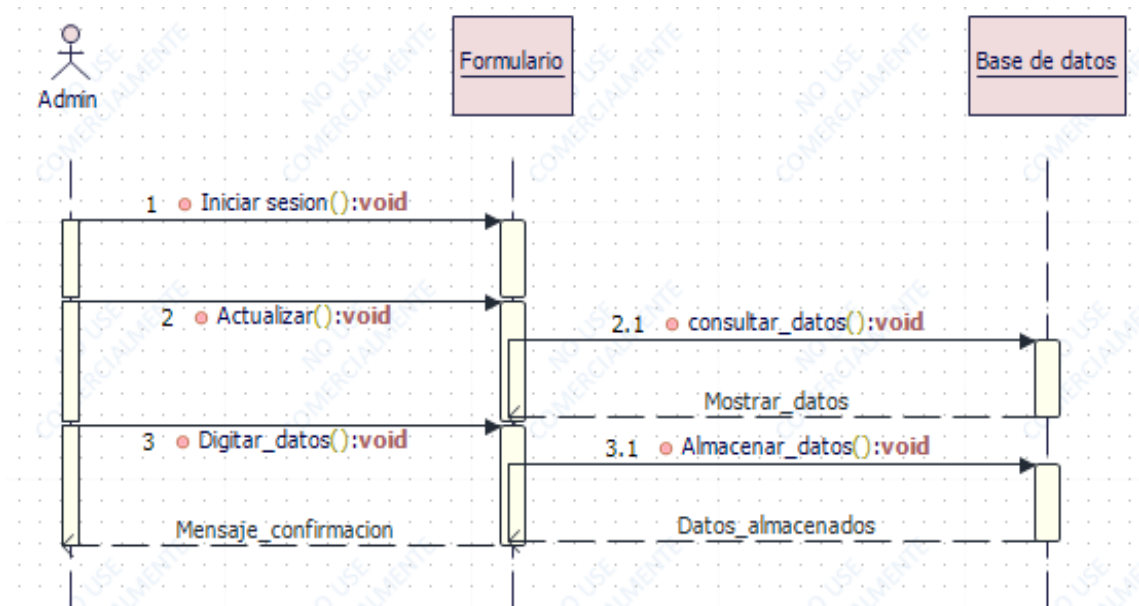


Figura 4.3: Gestionar datos administrador

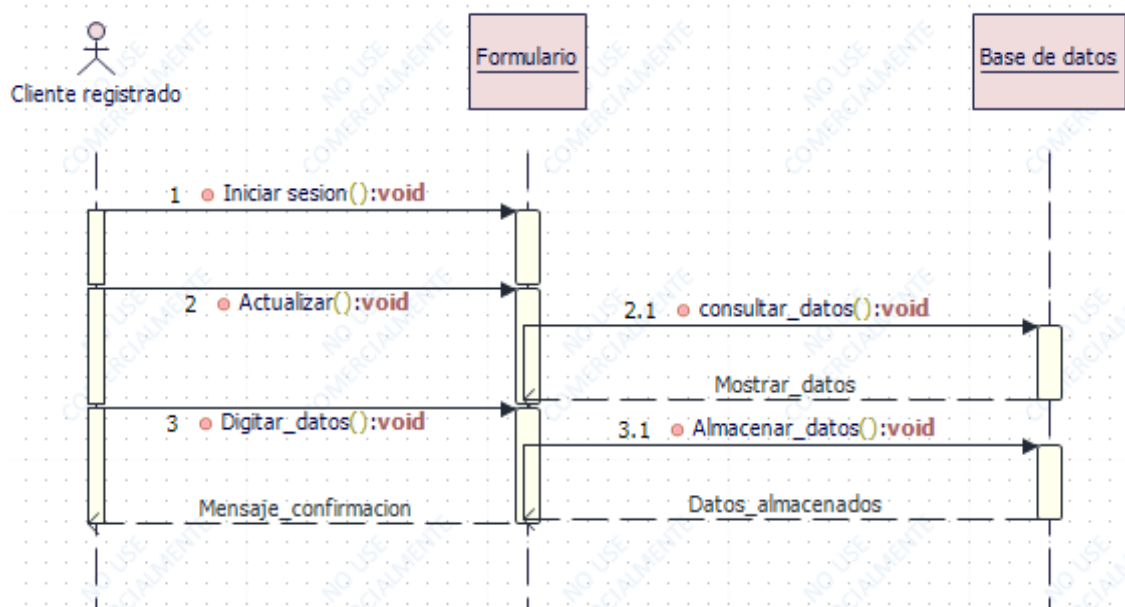


Figura 4.4: Gestionar datos cliente

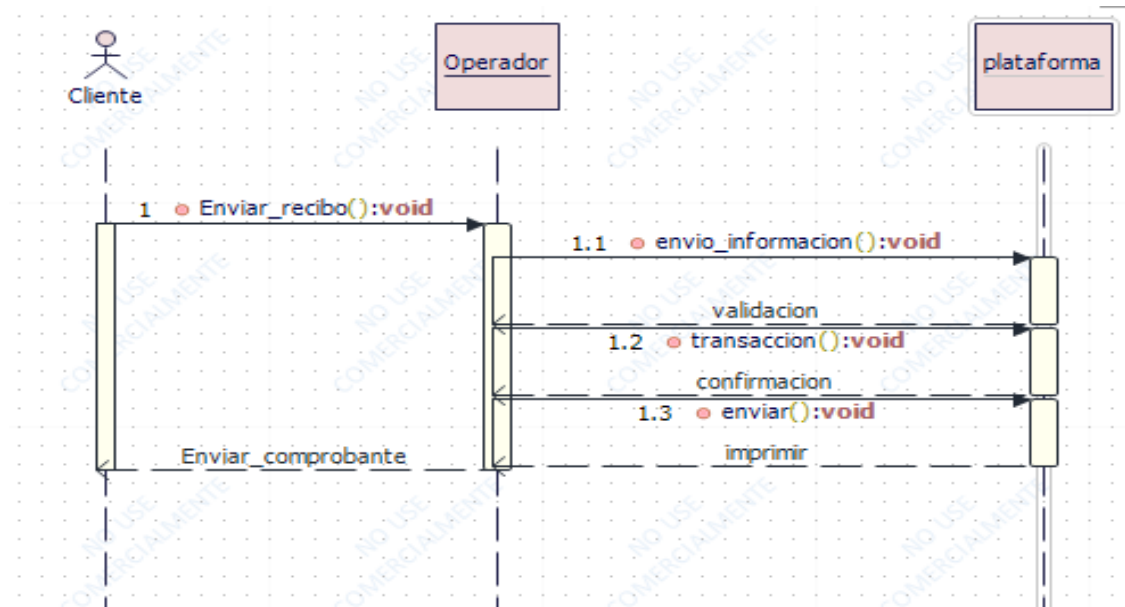


Figura 4.5: Pagar

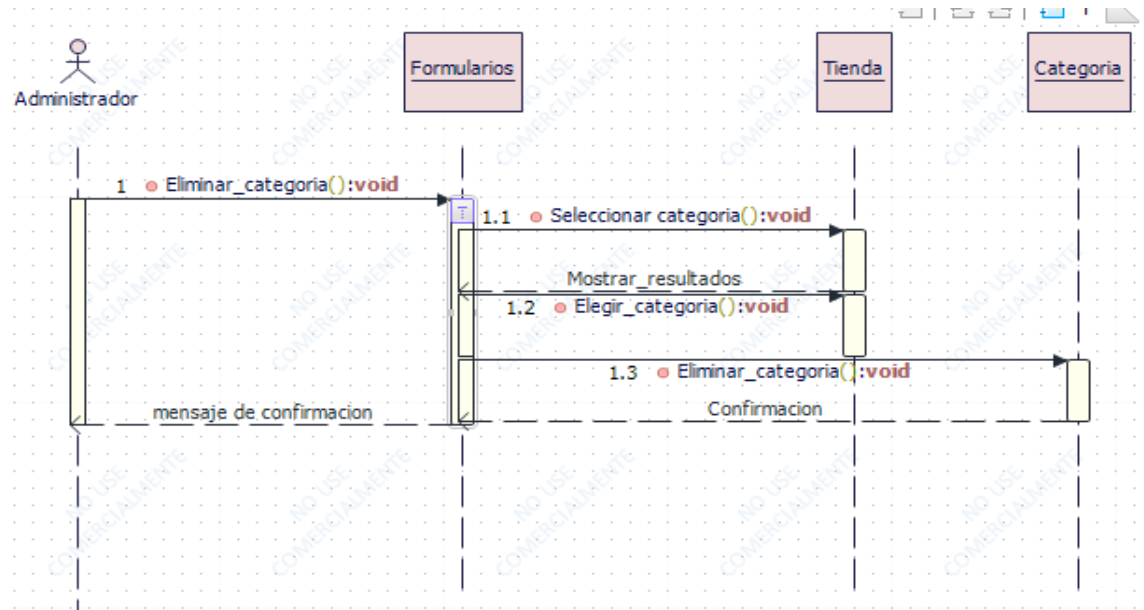


Figura 4.6: Eliminar categoría

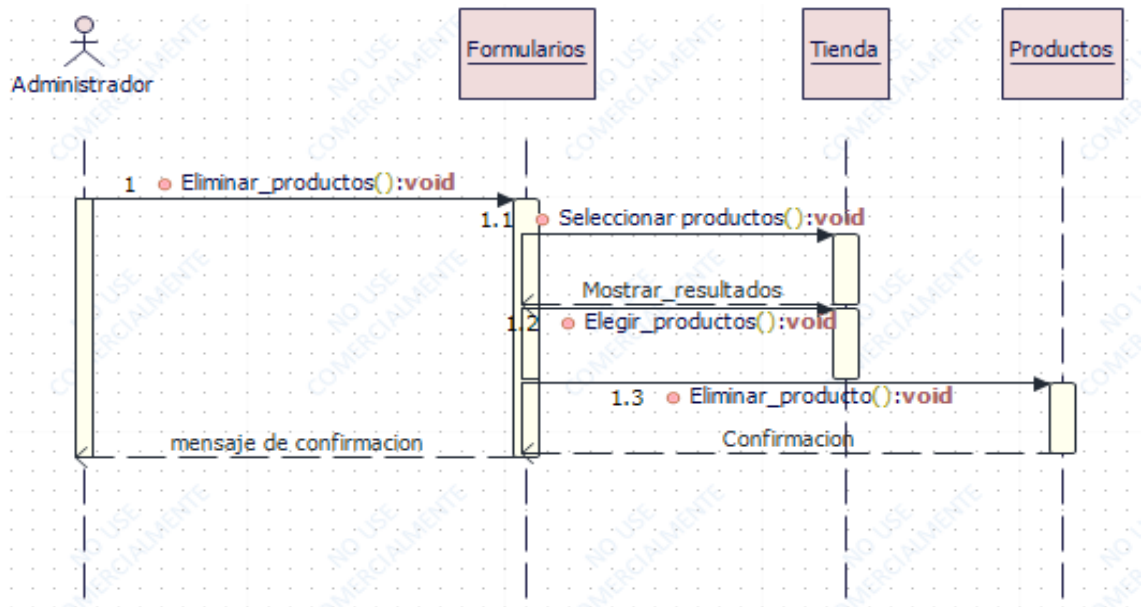


Figura 4.7: Eliminar producto

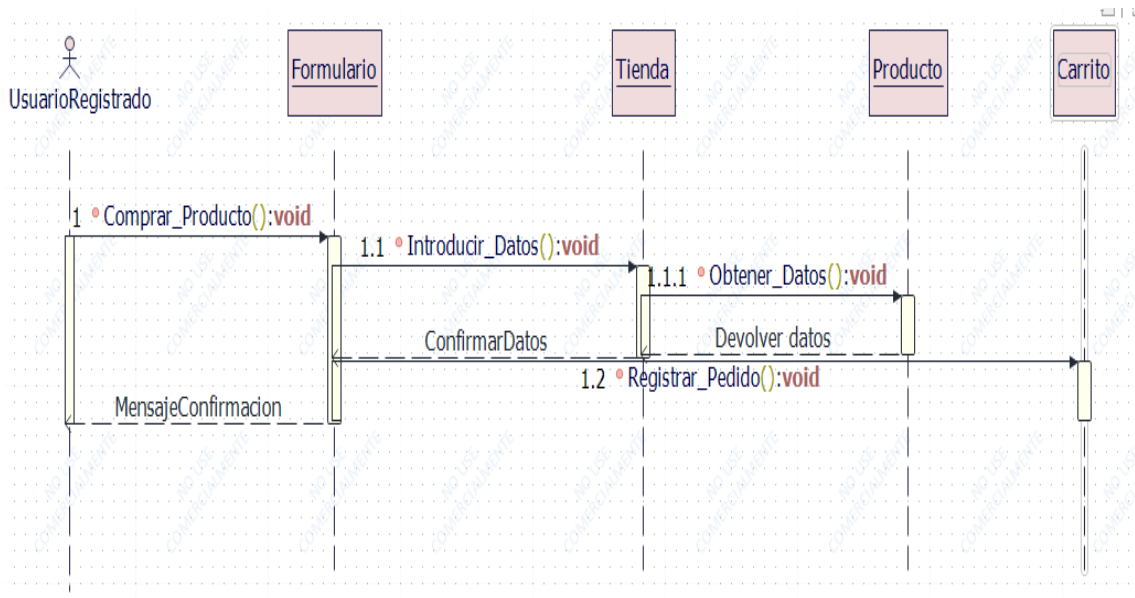


Figura 4.8: Comprar

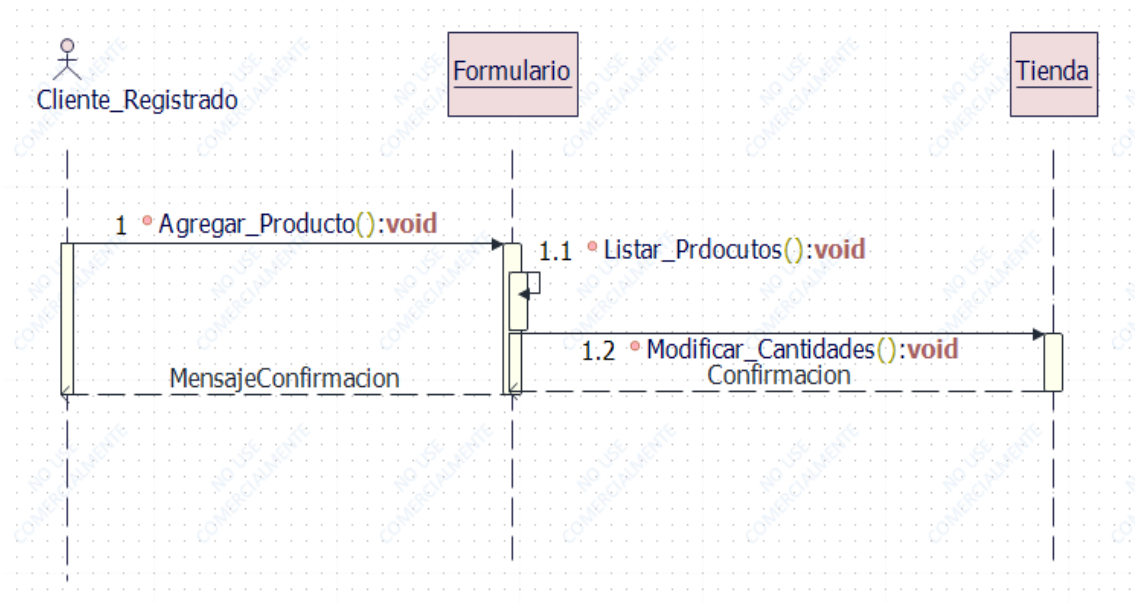


Figura 4.9: Gestión carrito de compra

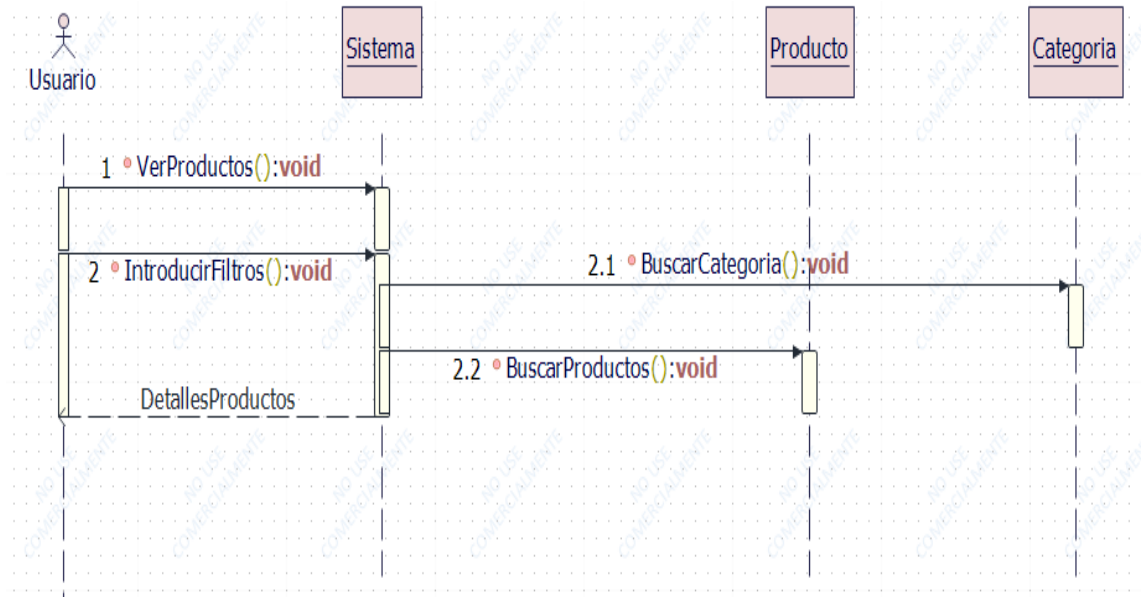


Figura 4.10: Consultar catálogo

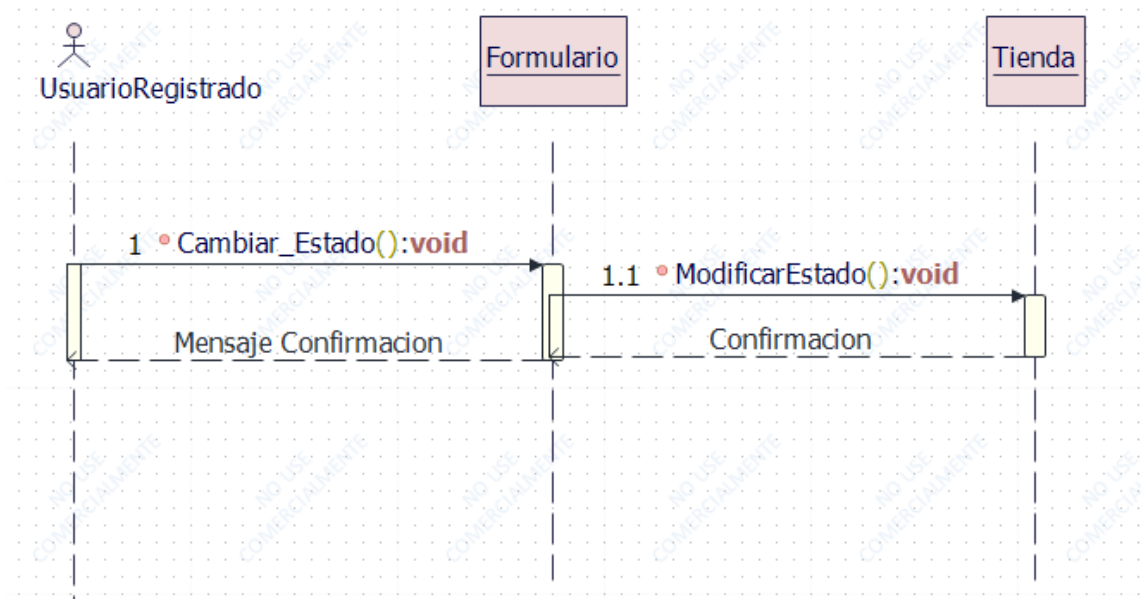


Figura 4.11: Dar baja

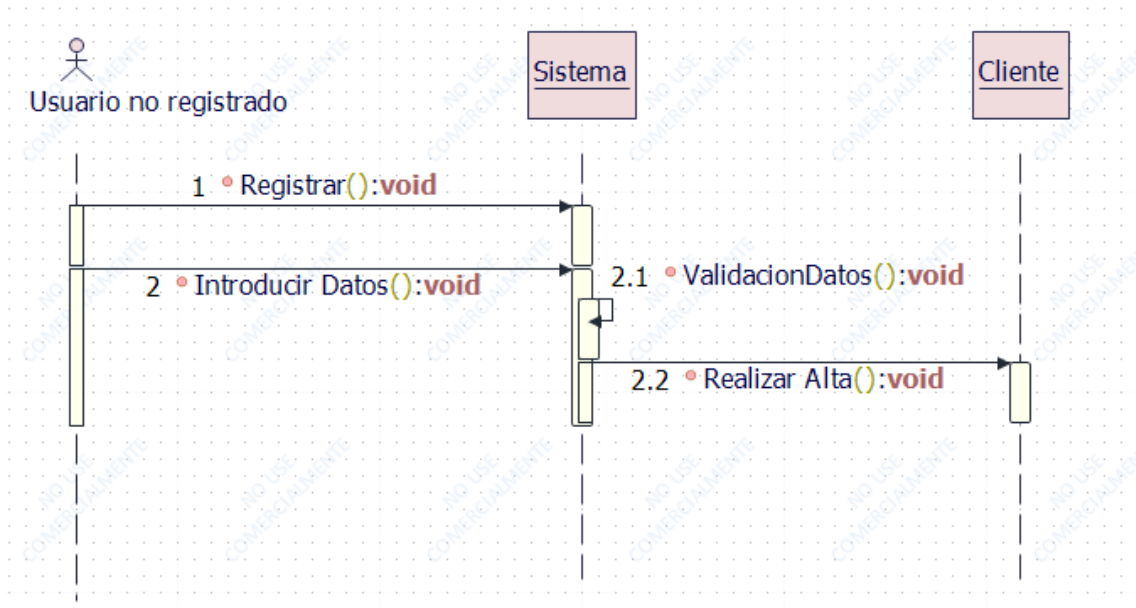


Figura 4.12: Registrar

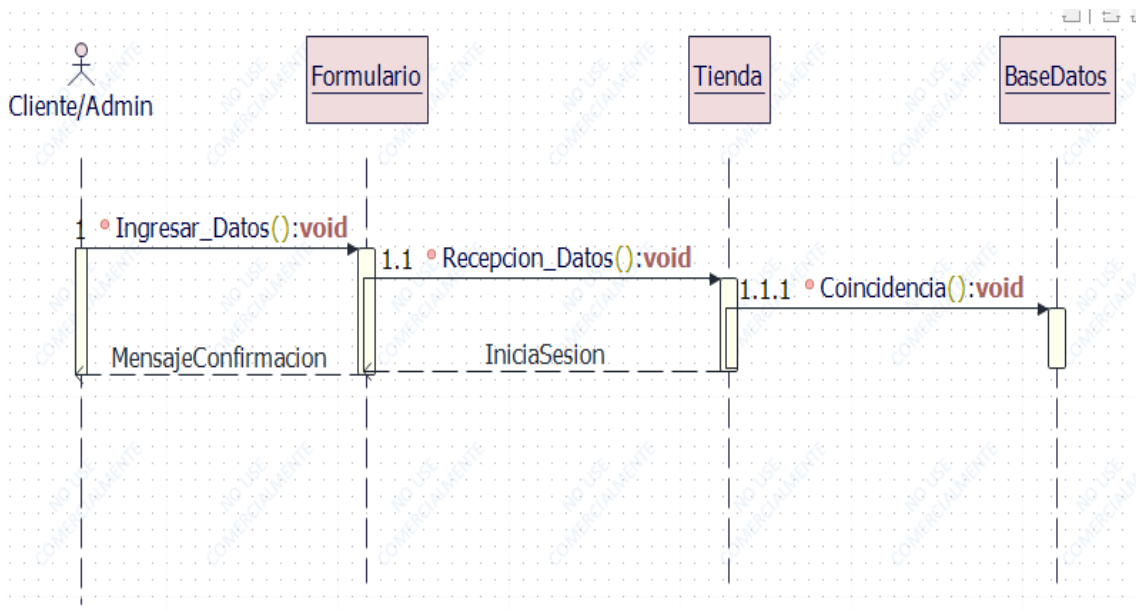


Figura 4.13: Validar usuario

## 4.2. Diagramas de comunicación

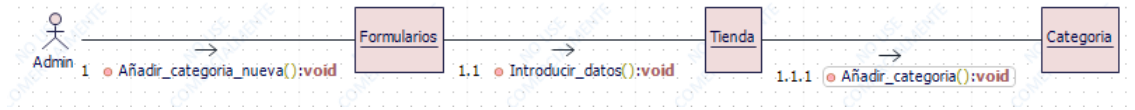


Figura 4.14: Añadir categoría

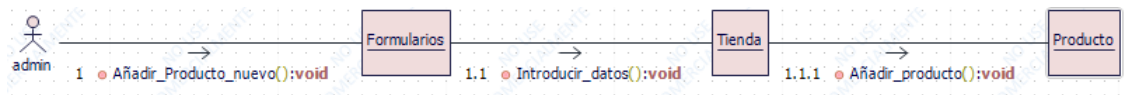


Figura 4.15: Añadir producto

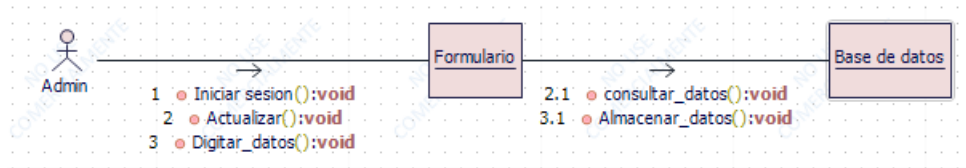


Figura 4.16: Gestionar datos administrador

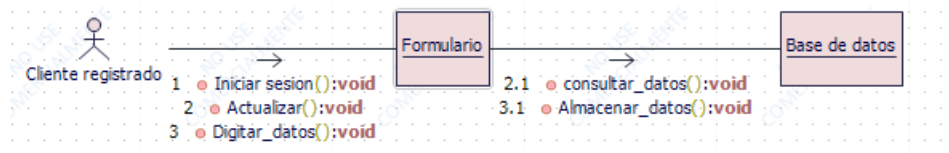


Figura 4.17: Gestionar datos cliente

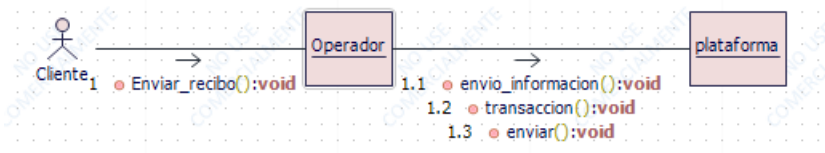


Figura 4.18: Pagar



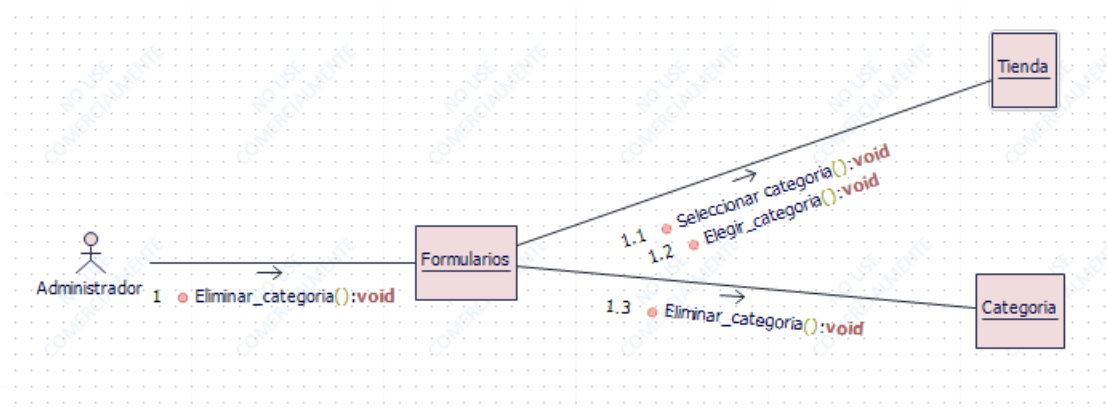


Figura 4.19: Eliminar categoría

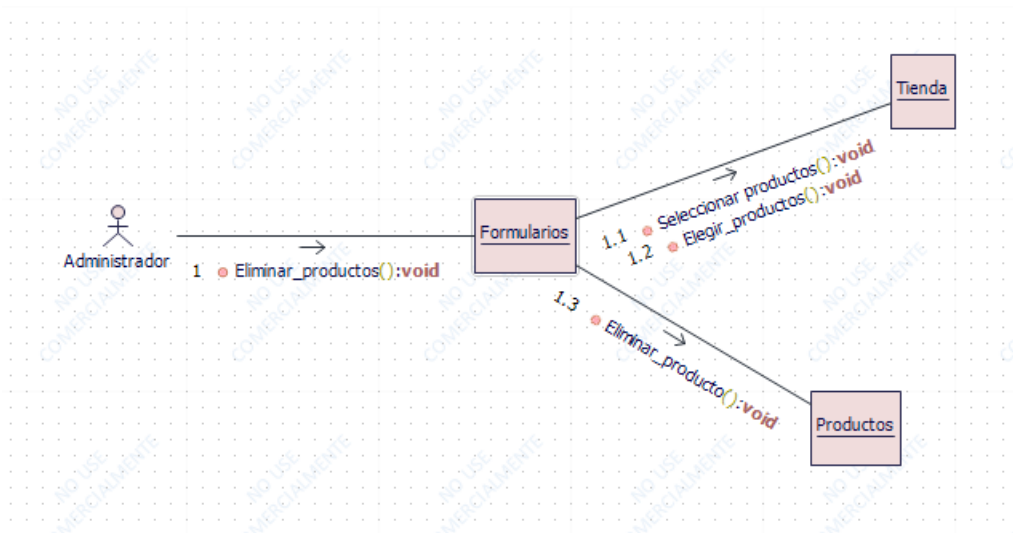


Figura 4.20: Eliminar producto

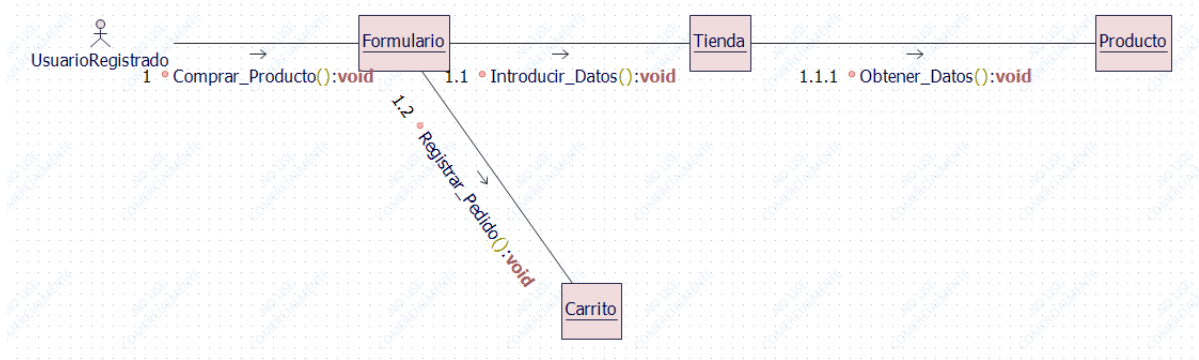


Figura 4.21: Comprar

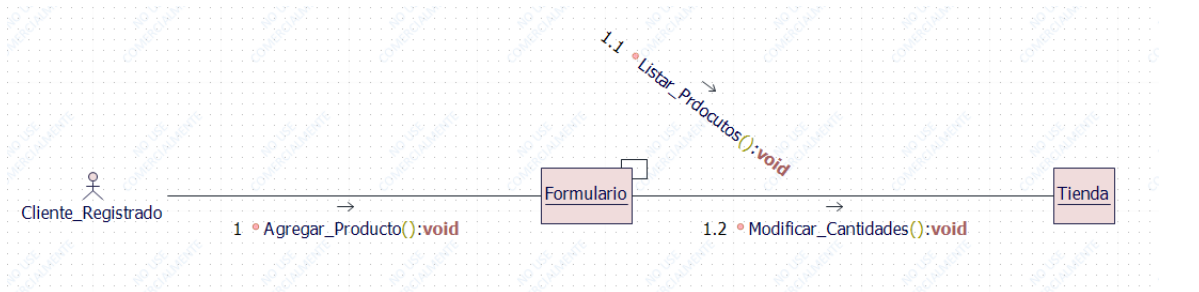


Figura 4.22: Gestión carrito de compra

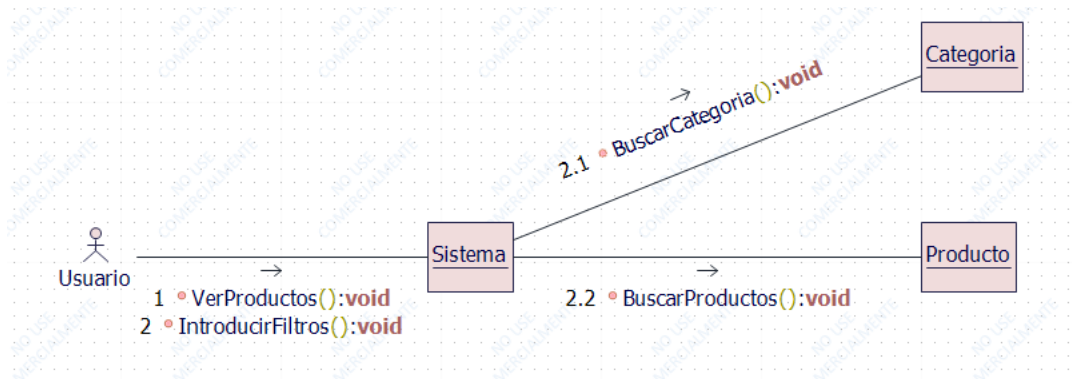


Figura 4.23: Consultar catálogo

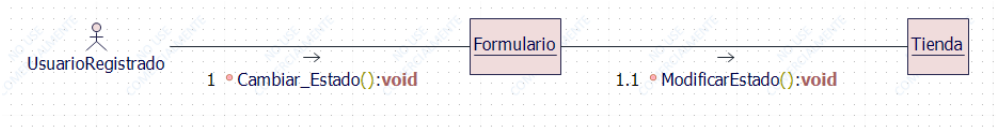


Figura 4.24: Dar baja

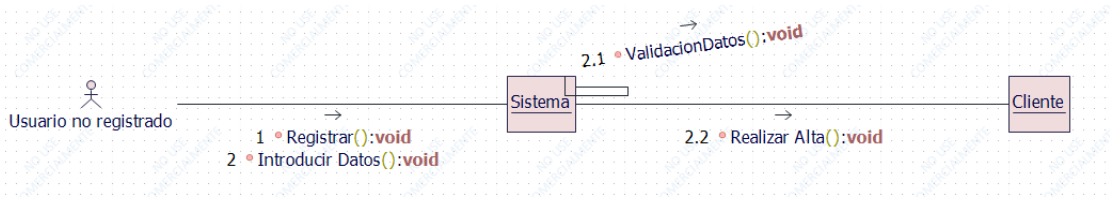


Figura 4.25: Registrar



Figura 4.26: Validar usuario





## 5.2. Relación de pedido con usuario

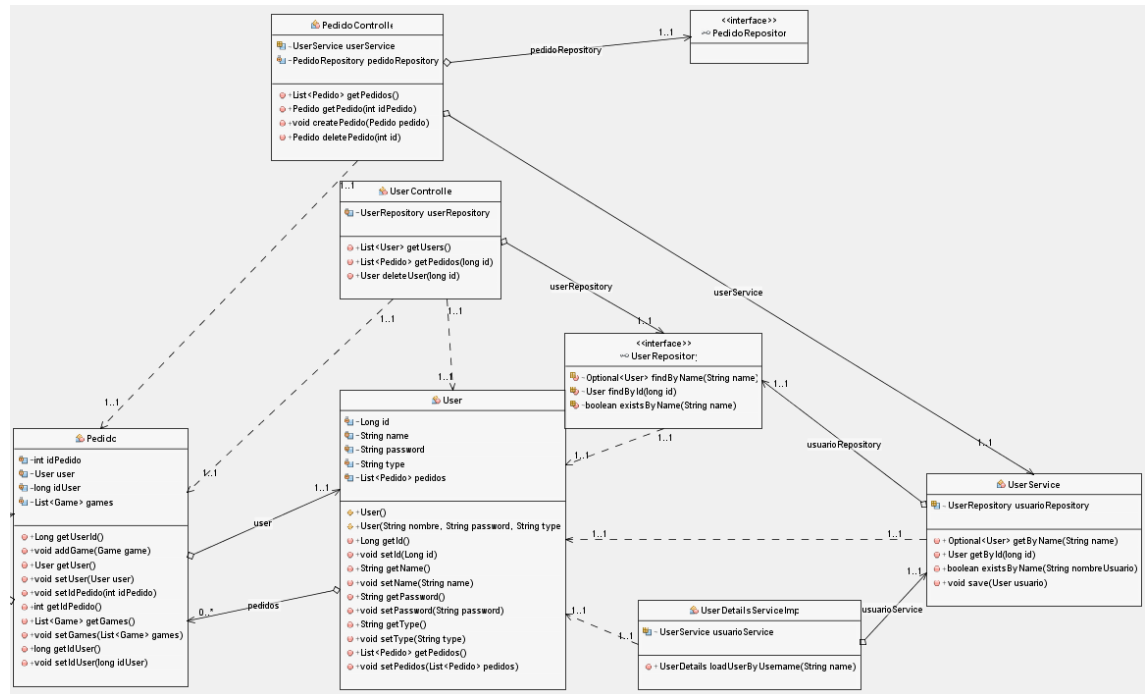


Figura 5.2: Diagrama de clases segunda parte

Para los diagramas de clase se tuvo en cuenta principalmente las clases relacionadas al modelo, como lo son usuario, categoría, juego y pedido y sus clases asociadas como lo son las clases de servicios, controladores y repositorios.

Estos diagramas pertenecen al modelo de aplicación realizado en Spring Boot de manera simplificada, ya que faltan algunas clases relacionadas a la seguridad de la aplicación y generación de token's para crear una sesión. Tampoco se incluye la representación de que los repositorios son repositorios JPA por lo que 'heredan' de estos, para realizar la comunicación con la base de datos MySQL .

## Capítulo 6

# Sistema

### 6.1. Introducción

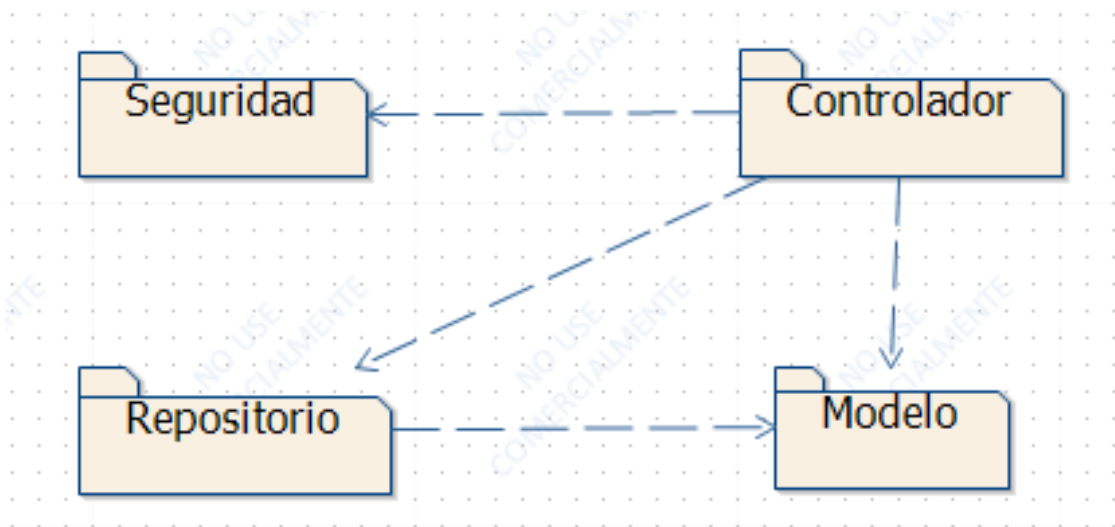


Figura 6.1: Diagrama Sistemas

Para representar nuestros sistemas y sus dependencias dentro del proyecto, elegimos los sistemas de seguridad, controlador, repositorio y modelo. El sistema controlador depende de los sistemas de seguridad, repositorio y modelo, mientras que el sistema repositorio depende únicamente del modelo.





## Capítulo 7

# Estados

### 7.1. Estados de sesión

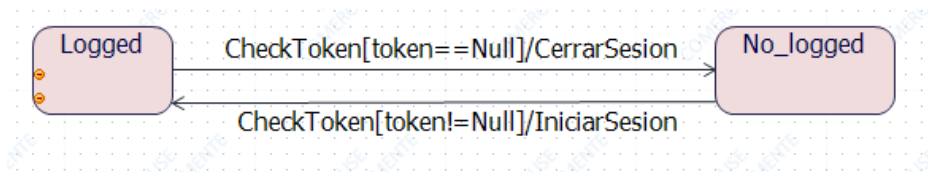


Figura 7.1: Diagrama de estados sesión

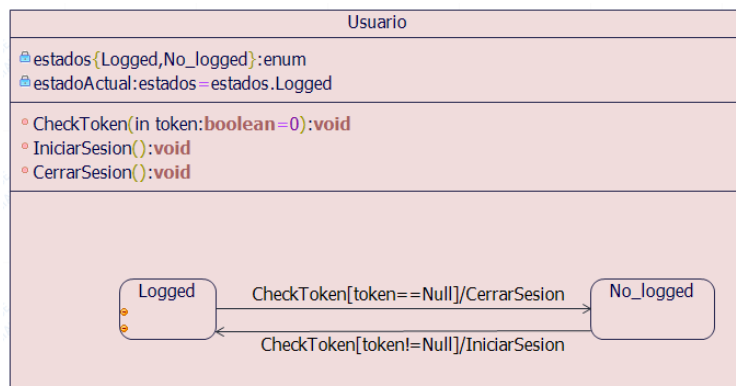


Figura 7.2: Ejemplo de clase representativa de los estados de sesión

Nuestro usuario poseerá dos estados que dependerá de un token. En un inicio nuestro usuario no tendrá iniciada la sesión, es

decir nuestro token sera igual a nulo, para iniciar sesión, ingresara un usuario y contraseña que ya deben estar registrado en la base de datos, así nuestro token pasara a ser diferente de nulo y el usuario ya habrá iniciado sesión en nuestra tienda.

## 7.2. Estados de juego en carrito

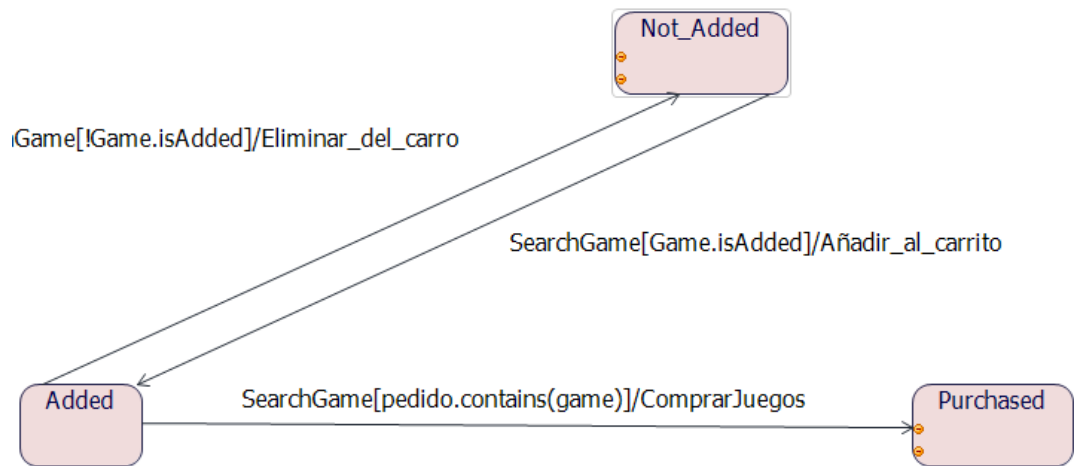


Figura 7.3: Diagrama de estados de juegos en carrito

Nuestro juego posee 3 estados, en un principio no va estar añadido a un carrito, es decir nuestra condición `Game.isAdded` va ser falsa. Ahora el usuario decide añadir el juego al carrito, por lo que la condición va pasar a ser verdadera. Si el usuario lo desea puede eliminar el juego del carrito y así nuestra condición pasara nuevamente a ser falsa. Ya por ultimo cuando el usuario compre todos los juegos que pertenecen al carrito, nuestra nueva condición `pedido.contains(game)` sera verdadera y los juegos dentro del carrito pasaran a ser juegos comprados por el usuario.

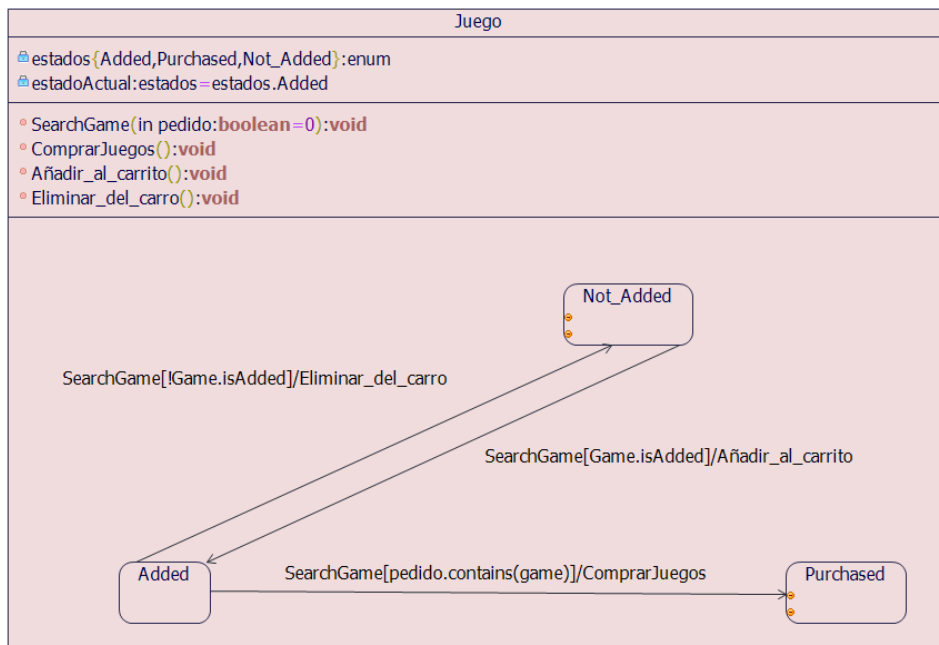


Figura 7.4: Ejemplo de clase representativa de los estados de juego en carrito



# Capítulo 8

## Componentes

### 8.1. Introducción

En nuestro proyecto detectamos tres componentes principales, los cuales son el Front end, Back end y un componente denominado orquestador. Estos se comunican mediante un API dentro de la cual se encuentran dos interfaces principales.

El componente orquestador contiene todas las clases 'lanzadoras' y que se encargan de 'orquestar' (de allí su nombre), el flujo de la plataforma (comunicación entre los demás componentes) a través de las interfaces.

Las dos interfaces las llamamos `httpClient` y `Controlador(es)`. La interfaz de `httpClient` es la que se encarga de suministrar los métodos necesarios para el componente Front. Dentro del componente Front se encuentra todo lo realizado con Angular, que se encarga de la interacción con el usuario. Por otra parte la interfaz de `Controlador(es)` se encarga de suministrar los métodos necesarios para el componente Back. Dentro de este componente se encuentra toda la lógica de negocio (Spring Boot) y la respectiva base de datos (MySQL). La representación de este razonamiento se encuentra en la figura 8.1 <sup>1</sup>

---

<sup>1</sup>Para ver una ejemplificación simple en código vaya al apartado de anexos

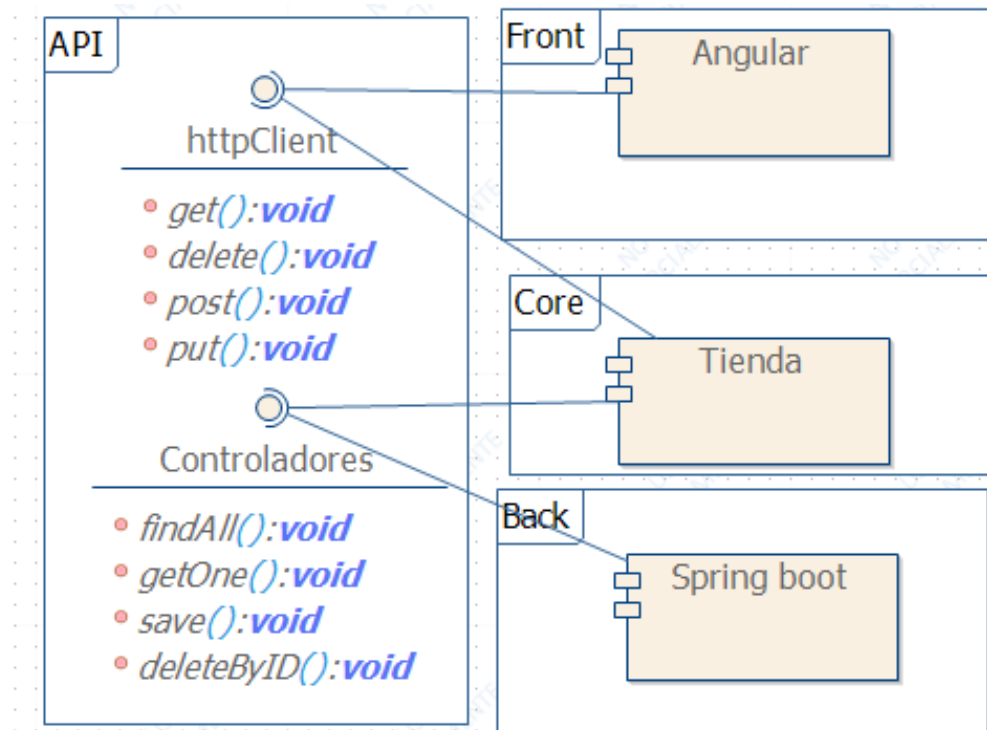


Figura 8.1: Diagrama Componentes

## Capítulo 9

# Nodos

### 9.1. Introducción

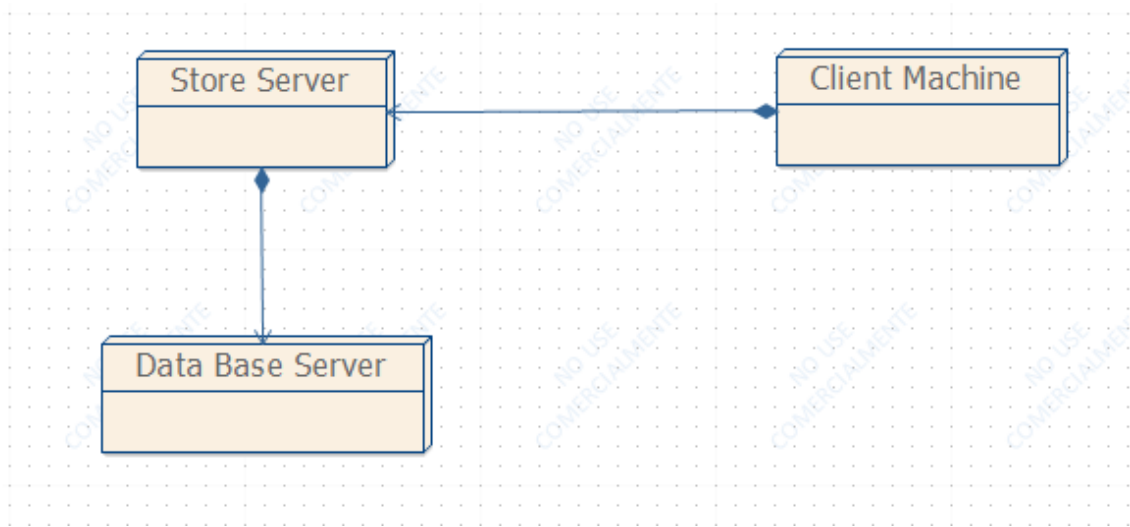


Figura 9.1: Diagrama Nodos

El montaje físico de la plataforma se compone de tres partes esenciales. El Store Server que se encarga de recibir las peticiones de usuario y realizar las respectivas consultas al Data Base Server, en el cual se encuentra el sistema gestor de base de datos. Por último el nodo de Client Machine representa todos los dispositivos des-

de los cuales se accede a la plataforma (computadores personales, teléfonos celulares, tabletas, etc)



## Parte III

### Cierre



## Capítulo 10

# Conclusiones

- La aplicación de la ingeniería de software durante la creación del proyecto, permite tener plena confianza en la calidad del producto final y asegurar un soporte de mantenimiento.
- En el desarrollo de productos de software las etapas de análisis de requerimientos y diseño son de gran importancia y por ende toman gran parte del tiempo del proyecto. Aunque para el desarrollo de este proyecto se trató de seguir una metodología ágil como lo es ASD, cabe aclarar que esta metodología se aplicó principalmente para las etapas de implementación y pruebas, puesto que como se mencionó, las etapas de análisis de requerimientos y diseño toman tiempo y de hacerse mal desde un principio puede generar retrasos en la entrega final, así se aplique una metodología ágil.
- La documentación específica de cada framework utilizado resultó vital para el desarrollo del proyecto. Como lo es la documentación de angular [3], de Spring Boot[4] y del sistema gestor de base de datos MySQL[5]. Por otra parte para el modulo de autenticación basado en token's se utilizó y adaptó el código realizado por Luigi Code en sus tutoriales sobre autenticación JWT [6].
- Durante la fase de diseño fue muy importante el desarrollo de distintos diagramas, para representar la estructura y comportamiento del sistema. Para realizar estos diagramas se utilizó la herramienta Coloso en su versión universitaria [7], lo cual facilitó esta labor puesto que contiene todo lo necesario para modelar un software.
- Con respecto al resultado final del proyecto, se cumplieron todos los

objetivos, lo restante se basa en mantenimiento e implementación de nuevas funciones que permitan la expansión, diversificación y escalamiento del proyecto.

# Capítulo 11

## Anexos

```
package edu.presentation;

import api.edu.cableado.Controladores;
import api.edu.cableado.httpClient;

public class Tienda {
    public static void main(String[] args) {
        httpClient Web = new Angular();
        Web.get();
        Web.delete();
        Web.post();
        Web.put();

        Controladores Con= new Spring_Boot();
        Con.findAll();
        Con.deleteByID();
        Con.getOne();
        Con.save();
    }
}
```

Tienda.java

```
package edu.presentacion;
import javax.swing.JOptionPane;

import api.edu.cableado.*;
public class Angular implements httpClient{

    @Override
    public void delete() {
        JOptionPane.showMessageDialog(null, "delete");
    }

    @Override
    public void get() {
        JOptionPane.showMessageDialog(null, "get");
    }

    @Override
    public void post() {
        JOptionPane.showMessageDialog(null, "post");
    }

    @Override
    public void put() {
        JOptionPane.showMessageDialog(null, "put");
    }
}
```

Angular.java

```
package api.edu.cableado;

public interface Controladores {
    void findAll();
    void getOne();
    void save();
    void deleteById();
}

Controladores.java
```

```
package edu.presentacion;
import javax.swing.JOptionPane;

import api.edu.cableado.Controladores;
public class Spring_Boot implements Controladores{

    @Override
    public void deleteByID() {
        JOptionPane.showMessageDialog(null, "deleteByID");
    }

    @Override
    public void findAll() {
        JOptionPane.showMessageDialog(null, "findAll");
    }

    @Override
    public void getOne() {
        JOptionPane.showMessageDialog(null, "getOne");
    }

    @Override
    public void save() {
        JOptionPane.showMessageDialog(null, "save");
    }
}
```

Spring Boot.java



```
package api.edu.cableado;

public interface httpClient {
    void get();
    void delete();
    void post();
    void put();
}

httpClient.java
```



# Referencias

- [1] L. Lizcano. “UML: Un Lenguaje de Modelo de Objetos”. En: *Universidad de La Rioja* (2002), págs. 1-6.
- [2] J. Rivero y K. Torres. “DESARROLLO ADAPTABLE DE SOFTWARE (ASD)”. En: *Universidad Nacional Experimental De los Llanos Occidentales* (2014), págs. 3-8.
- [3] Angular. *Angular Documentation*. URL: <https://angular.io/docs>. (accessed: 18.02.2021).
- [4] Spring Boot Developer Team. *Spring Boot Reference Documentation*. URL: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>. (accessed: 19.02.2021).
- [5] Oracle. *Oracle Cloud Infrastructure Documentation*. URL: <https://docs.oracle.com/en-us/iaas/mysql-database/doc/getting-started.html>. (accessed: 19.02.2021).
- [6] Luigi Code. *Autenticación JWT con Spring Boot y Angular*. URL: <https://www.youtube.com/watch?v=gKzEFSnWnk4>. (accessed: 19.02.2021).
- [7] Sandro Javier Bolaños Castro. *COLOSO Una herramienta de ensueño*. URL: <https://www.colosoft.com.co/>. (accessed: 19.02.2021).