Trabajo Práctico Taller “Social Commerce”

Documento de Arquitectura de Software

Versión <1.0>

Revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 27/05/18 | 1.0 | Primer entrega del documento de arquitectura de software. | Grupo Gauss |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contenidos

[**Introducción**](#_30j0zll) **4**

[Definiciones, Siglas y Abreviaturas](#_1fob9te) 4

[Referencias](#_3znysh7) 4

[**Representación Arquitectónica**](#_2et92p0) **4**

[**Objetivos de la Arquitectura y Restricciones**](#_tyjcwt) **5**

[**Vista de Casos de Uso**](#_3dy6vkm) **5**

[Realizaciones de los Casos de Uso](#_1t3h5sf) 17

[**Vista Lógica**](#_4d34og8) **18**

[Visión General](#_2s8eyo1) 19

[Paquetes de Diseño Significativos](#_17dp8vu) 19

[**Vista de Procesos**](#_3rdcrjn) **19**

[**Vista de Despliegue**](#_26in1rg) **20**

[**Vista de Implementación**](#_lnxbz9) **20**

[Visión General](#_35nkun2) 21

[Capas](#_1ksv4uv) 21

[**Vista de Datos (opcional)**](#_44sinio) **21**

[**Tamaños y Performance**](#_2jxsxqh) **21**

[**Calidad**](#_z337ya) **22**

Documento de Arquitectura de Software

# Introducción

En este Documento de Arquitectura de Software buscamos ofrecer una visión detallada a través de

diferentes vistas, de la arquitectura de una aplicación de Compra/Venta del estilo e-commerce orientado a un funcionamiento social. Cada una de las vistas exhibe un aspecto en particular del software a desarrollar. El objetivo de este documento es darle al lector una visión general y comprensible del diseño de dicha aplicación.

## Definiciones, Siglas y Abreviaturas

Actores: Especifica un rol de un usuario o cualquier otro sistema que interactúa con un Caso de Uso.

Caso de Uso: Acciones que deben realizarse para llevar a cabo un proceso.

UML: Lenguaje Unificado de Modelado (proveniente de su nombre en inglés “Unified Modeling Language”)

Vista: Representación gráfica de un aspecto del sistema en alto nivel.

Cliente: Usuario del sistema que hace el rol de comprador. Crea/administra grupos, etc.

Owner: Usuario del sistema que hace el rol de vendedor. Crea publicaciones de artículos, ofertas, etc.

Equipo de compra: Grupo de clientes que se juntan para comprar artículos a menor precio.

Círculo de compra: Grupo de clientes que aportan un monto de plata. La suma de esta plata se sortea un usuario (no repetido) al azar por mes.

Grupo de compra/venta: Grupo de clientes que puede vender y comprar artículos usados.

## Referencias

# Representación Arquitectónica

La arquitectura de la aplicación Social Commerce será expuesta a través de las siguientes vistas y sus correspondientes diagramas UML:

* Vista de Casos de Uso: Muestra en uno o más Diagramas de Casos de Uso los casos de uso principales del modelo de casos de uso que sean funcionalidades centrales del sistema final. En este diagrama los Actores están relacionados a Casos de Uso que representan funcionalidades que le son relevantes.
* Vista Lógica: Brinda una visión estática (no cambiante con el tiempo) de las clases de negocio y clases del Sistema Software a través de Diagramas de Clases. Puede o no presentar una descomposición en paquetes para simplificar un diagrama al costo de ser más abstracto. El contenido del paquete debe ser detallado en otro diagrama, si éste es el caso.
* Vista de Procesos: describe a través de un Diagrama de Actividades funcionalidades clave del sistema, exponiendo que acciones deben suceder para que los distintos módulos del proyecto se comuniquen entre sí; y a través de un Diagrama de Estados representa la transición entre múltiples estados en los que el sistema se encontrará para un caso de uso determinado a través de eventos y condiciones que disparen el cambio entre estados.
* Vista de Despliegue: A través de un Diagrama de Despliegue, esta vista describe nodos los cuales contienen algunos componentes del Diagrama de Componentes y establece de qué forma se comunicarán (protocolo de comunicación, ejemplo: TCP/IP, HTTPS, etc.) y/o en qué clase de soporte físico se realizará la implementación de dicho nodo (ejemplo: qué clase de base de datos se utilizará).
* Vista de Implementación: describe el Diagrama de Componentes, en el cual dichos componentes representan conjuntos de servicios o funcionalidades con una finalidad común. Los componentes deben ser suficientemente genéricos como para poder ser intercambiables y reutilizables. El hecho que sean reutilizables también implica que dichos componentes pueden ya existir y pueden simplemente ser una API externa a nuestro sistema que implementaremos en lugar de desarrollar la funcionalidad entera.

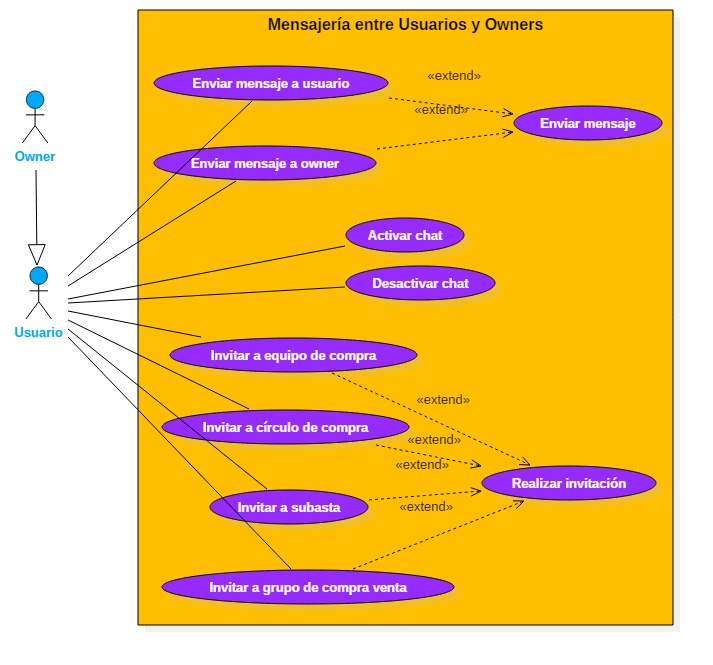
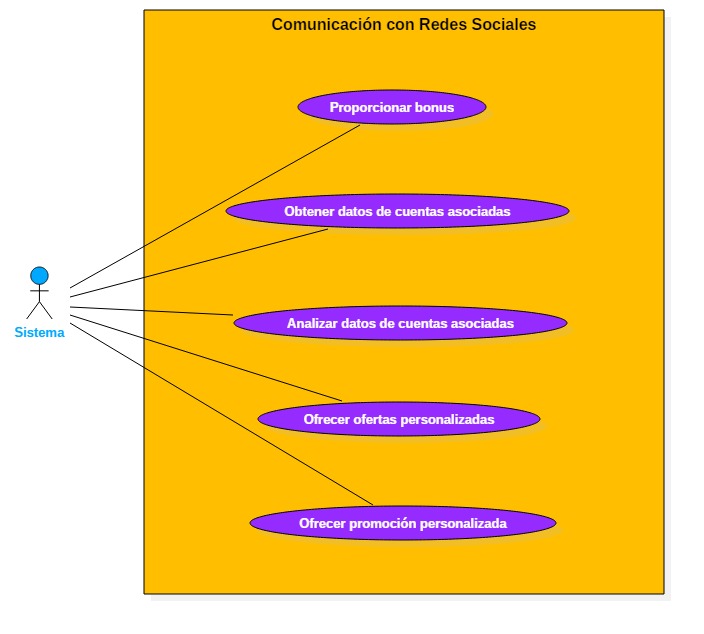
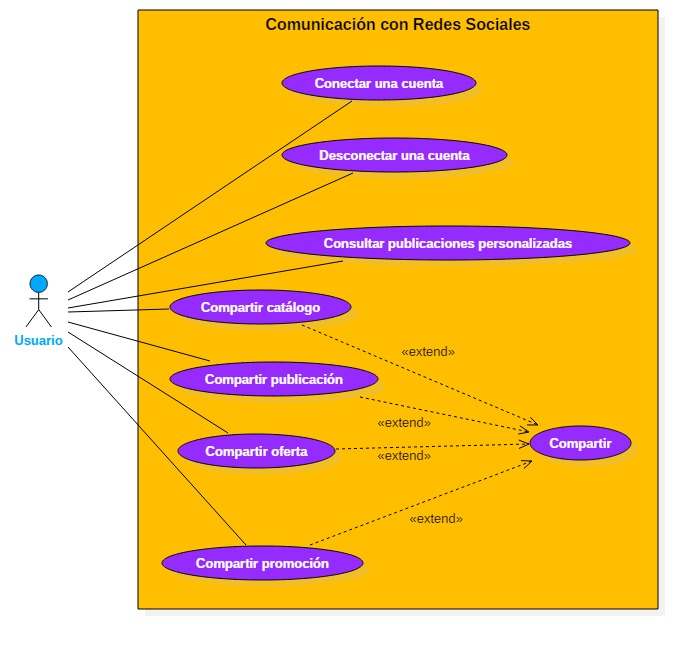
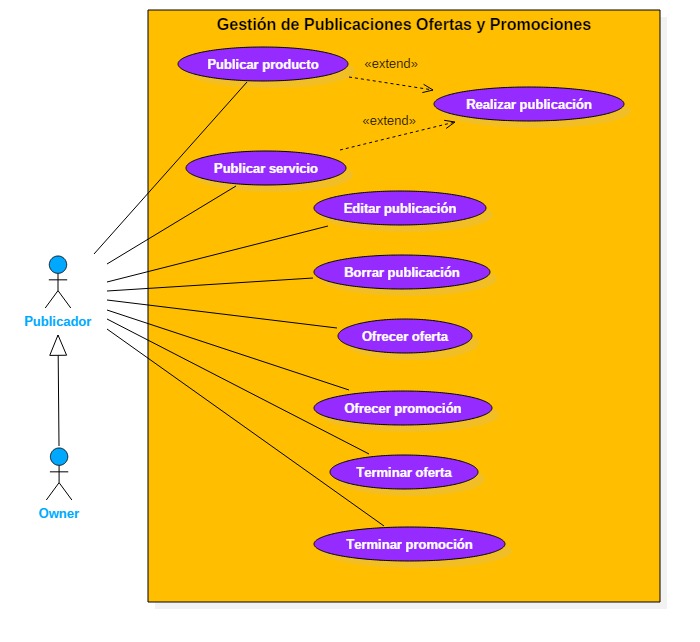
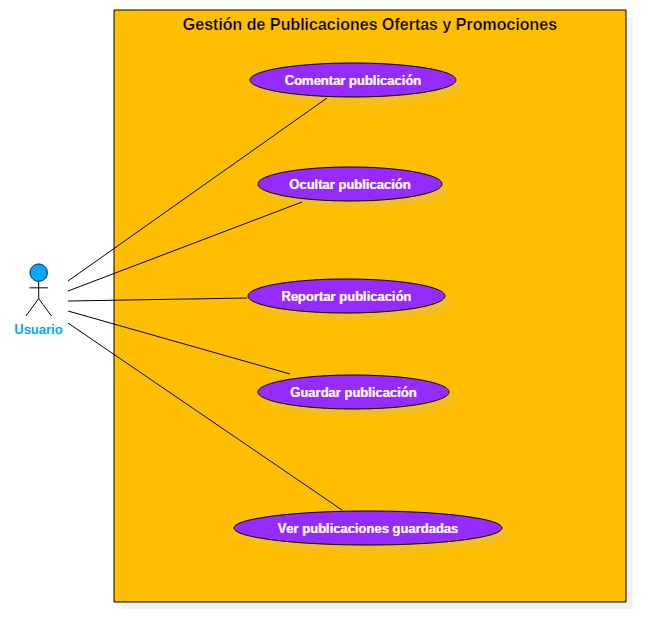
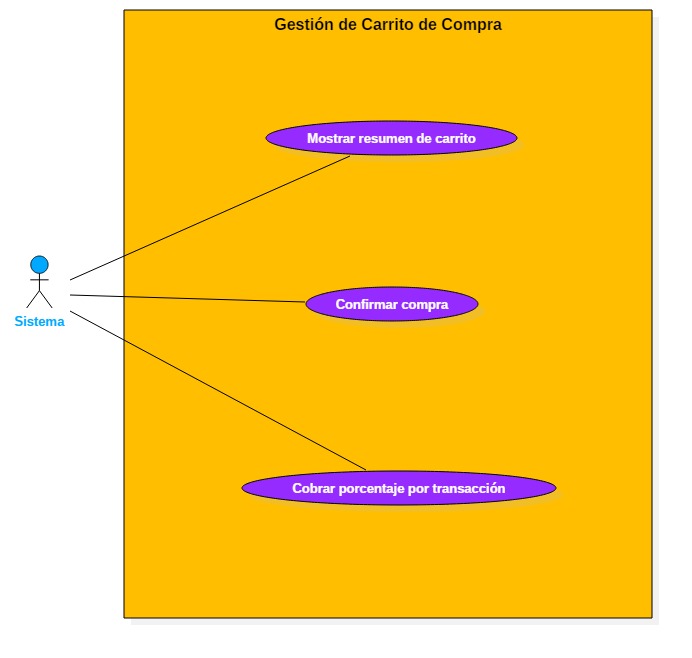
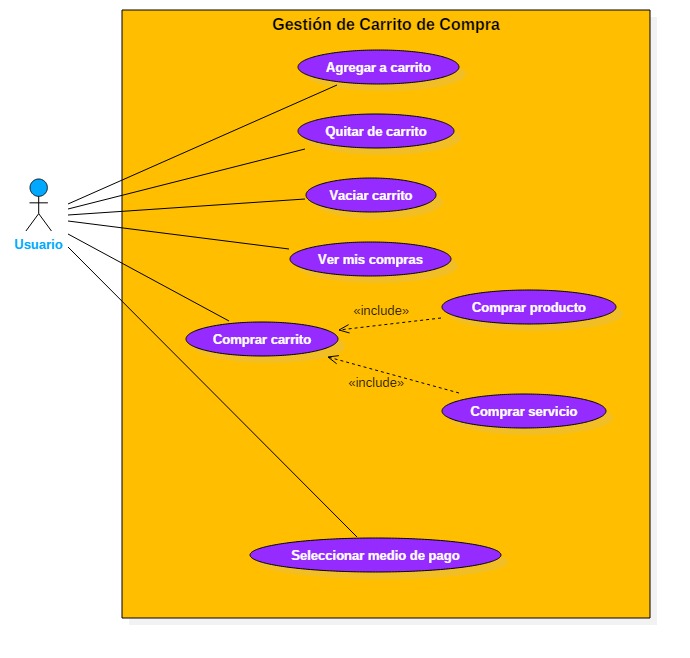
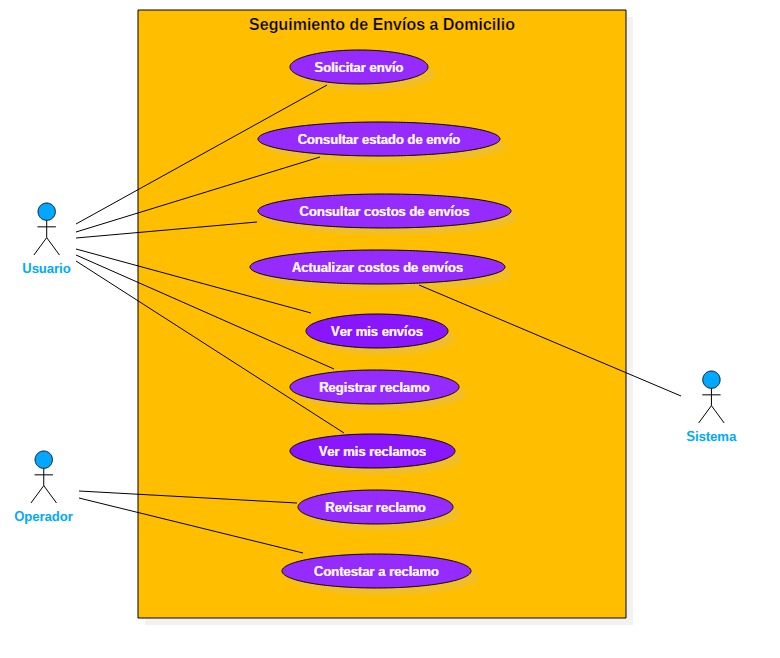
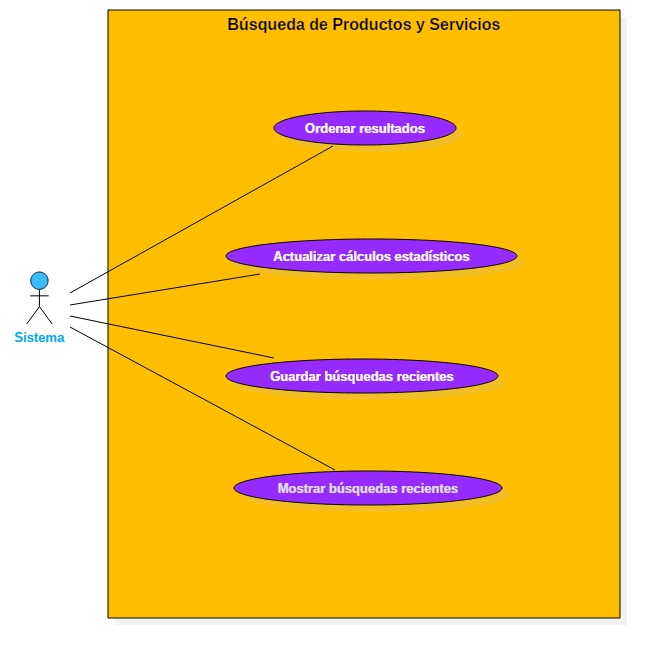
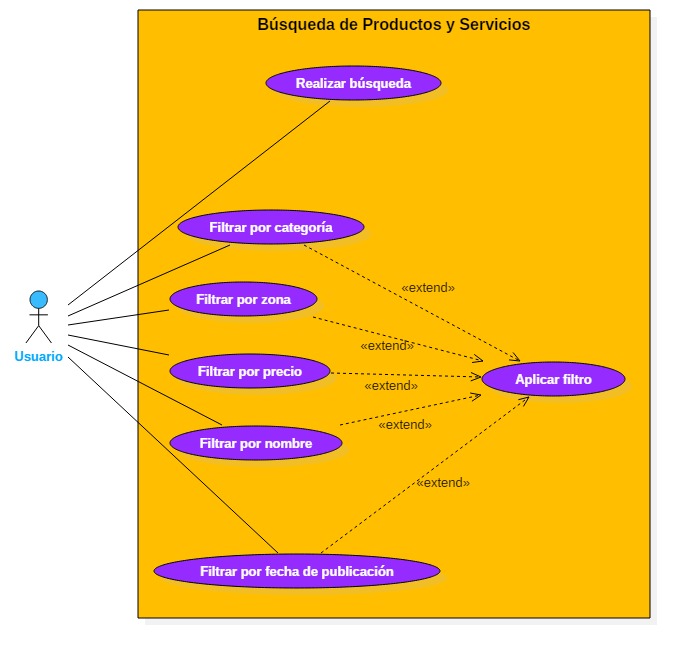
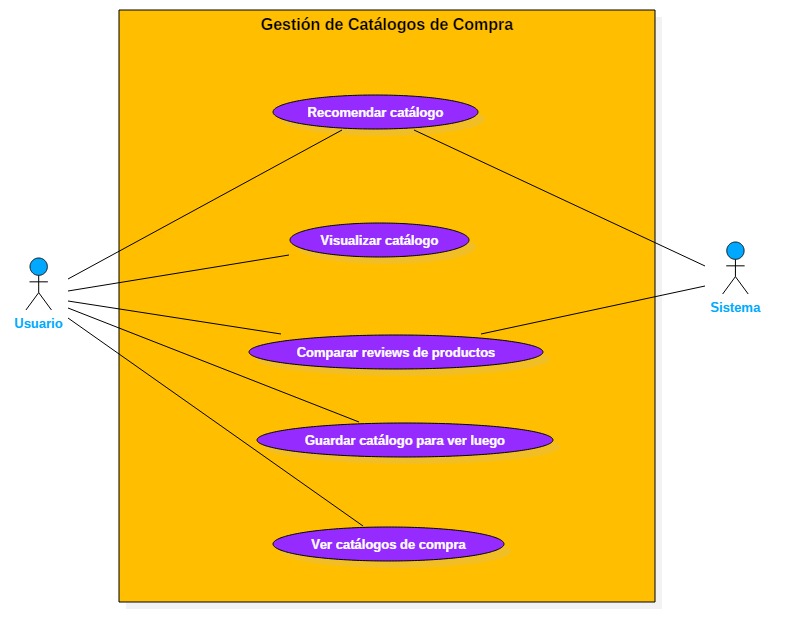
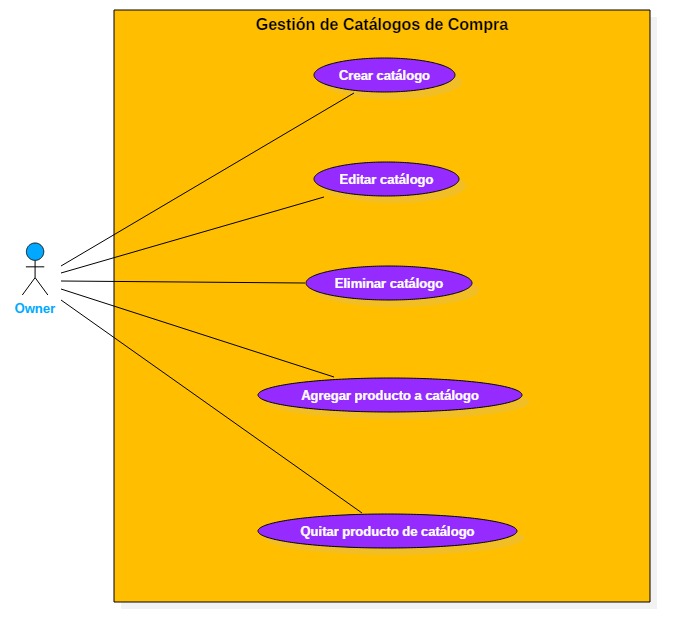
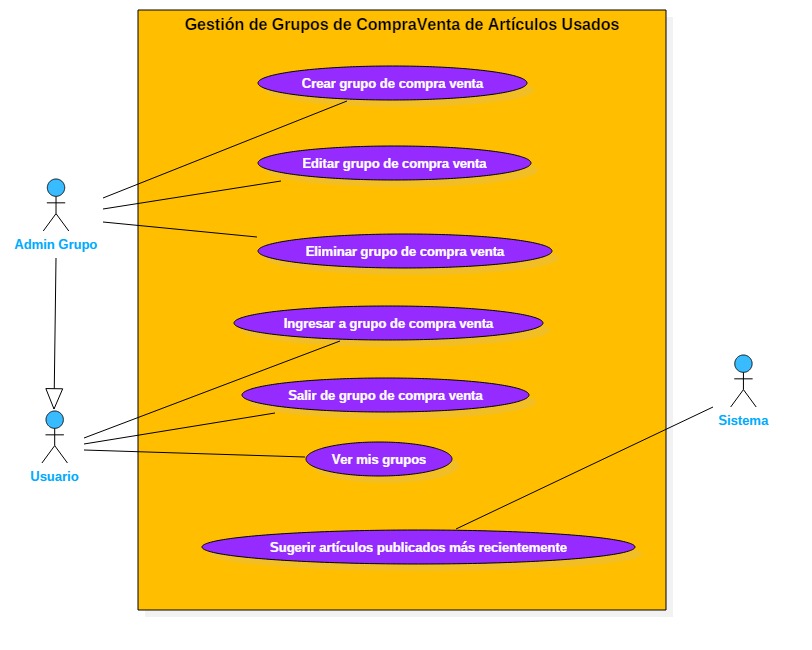
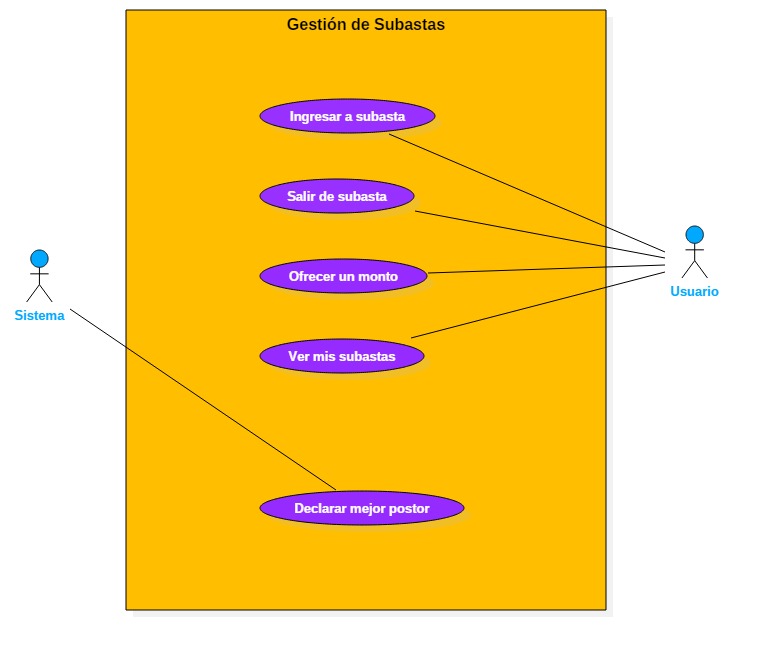
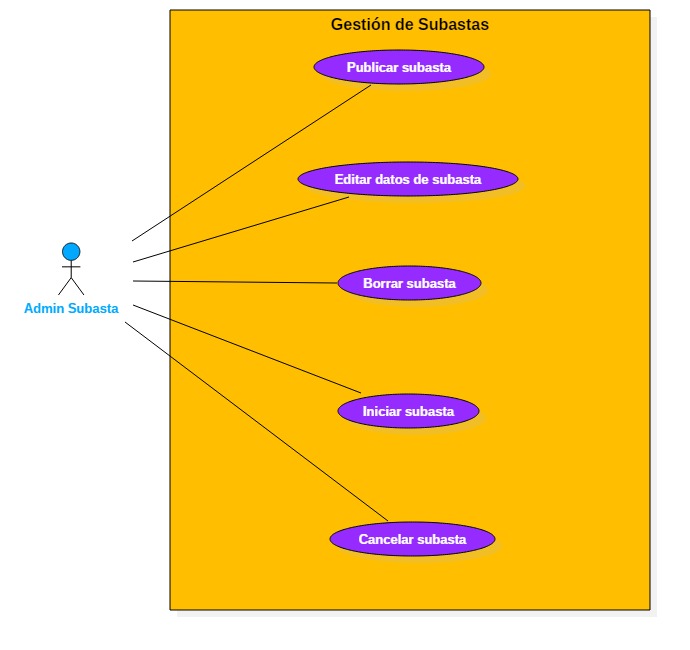
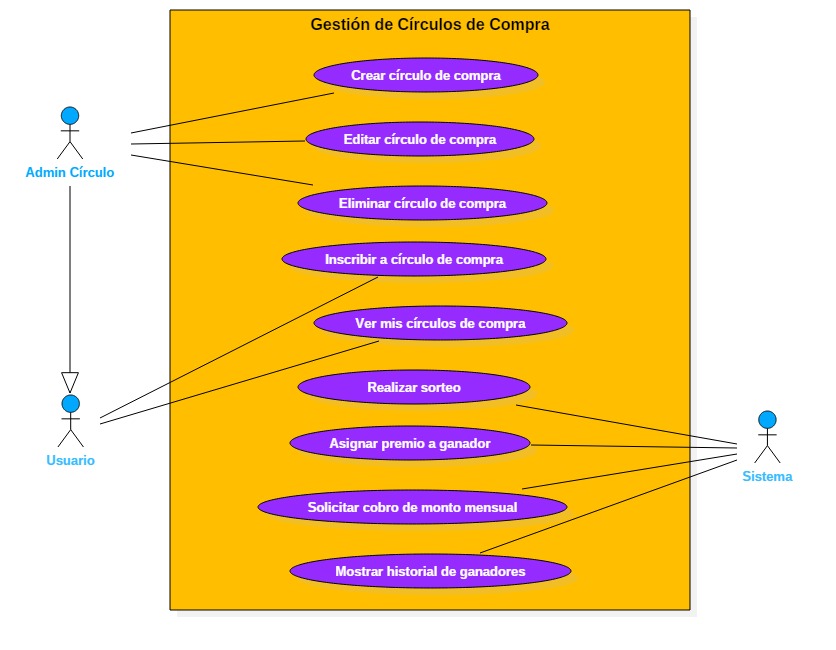
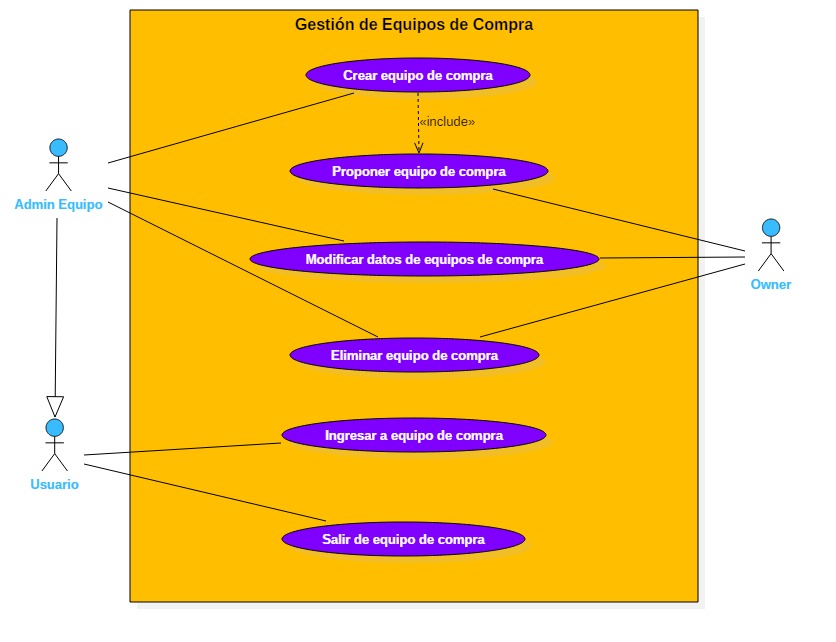
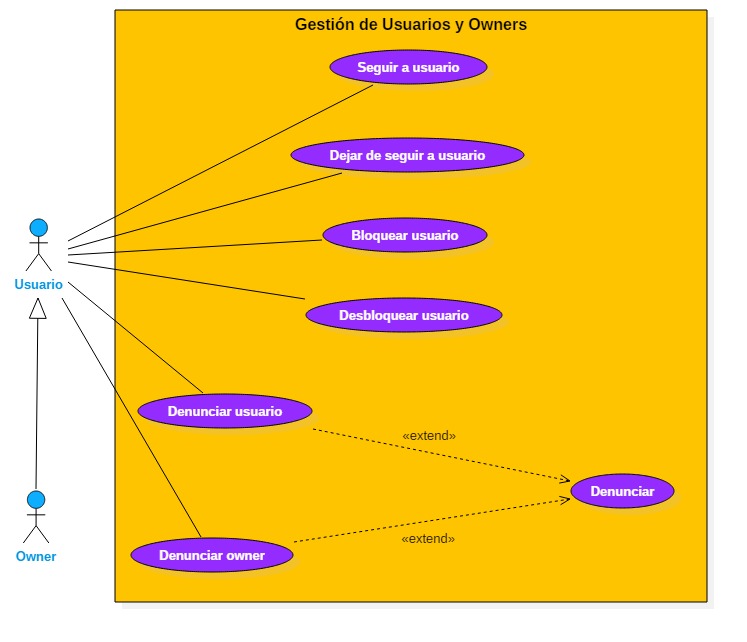
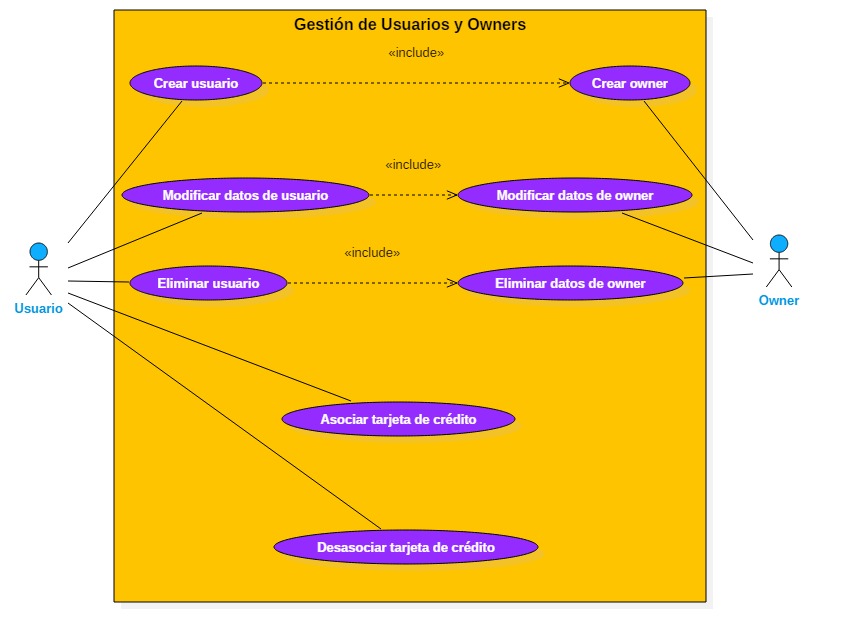
# Objetivos de la Arquitectura y Restricciones

El objetivo de un documento de arquitectura es el actuar como medio de comunicación entre un arquitecto de software y los miembros de otros equipos que participan del proyecto, dejándoles conocer aspectos significativos de la arquitectura que se desarrollará en el proyecto. Esto es especialmente cierto cuando el documento de arquitectura se presenta a los Stakeholders, los cuales a menudo no conocen detalles técnicos sobre software, pero desean saber que sus inquietudes e intereses están siendo tomados en cuenta.

# Vista de Casos de Uso

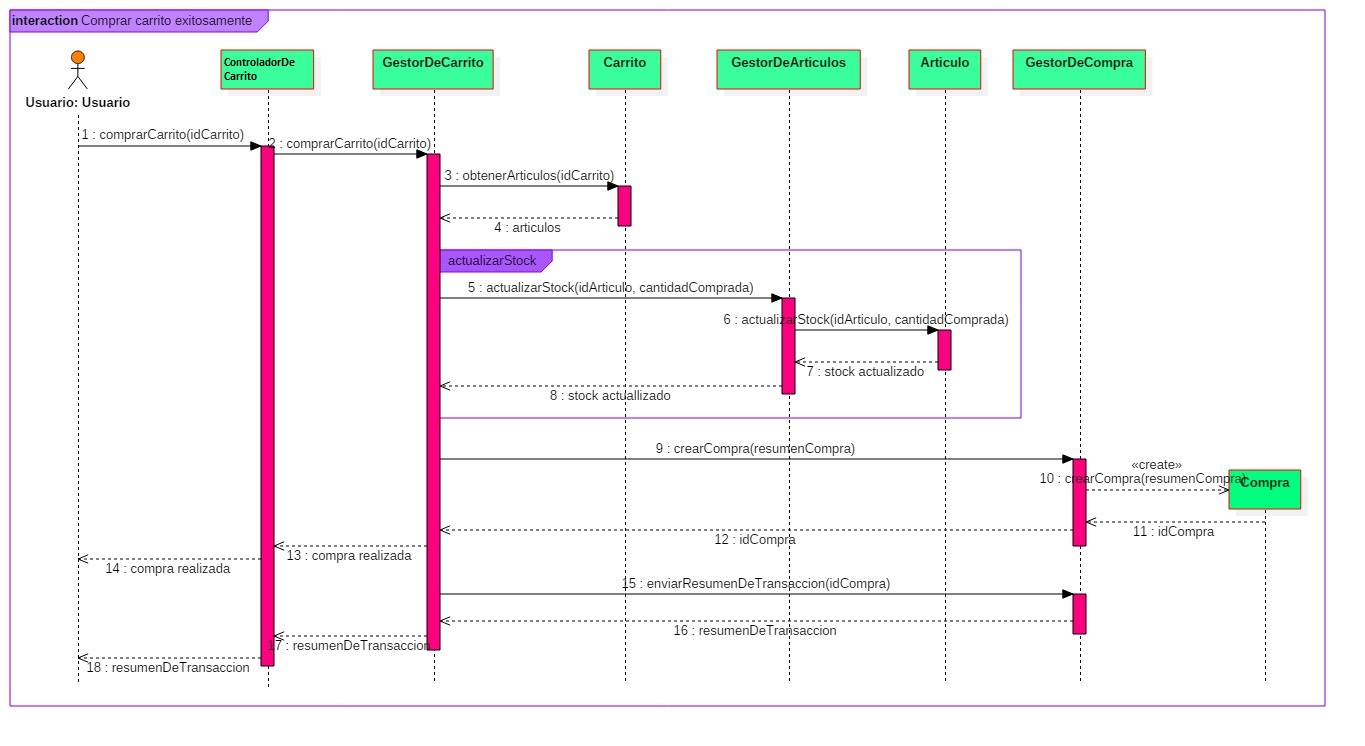
En esta sección del documento se expondrán los casos de uso relevantes a nuestra aplicación, los cuales fueron convenientemente separados por actor, para permitir una visualización más sencilla de los diagramas, así como para hacer más fácil identificar las responsabilidades de cada uno de ellos.

Aclaración: Los diagramas de casos de uso están divididos por funcionalidad, no por subsistemas.



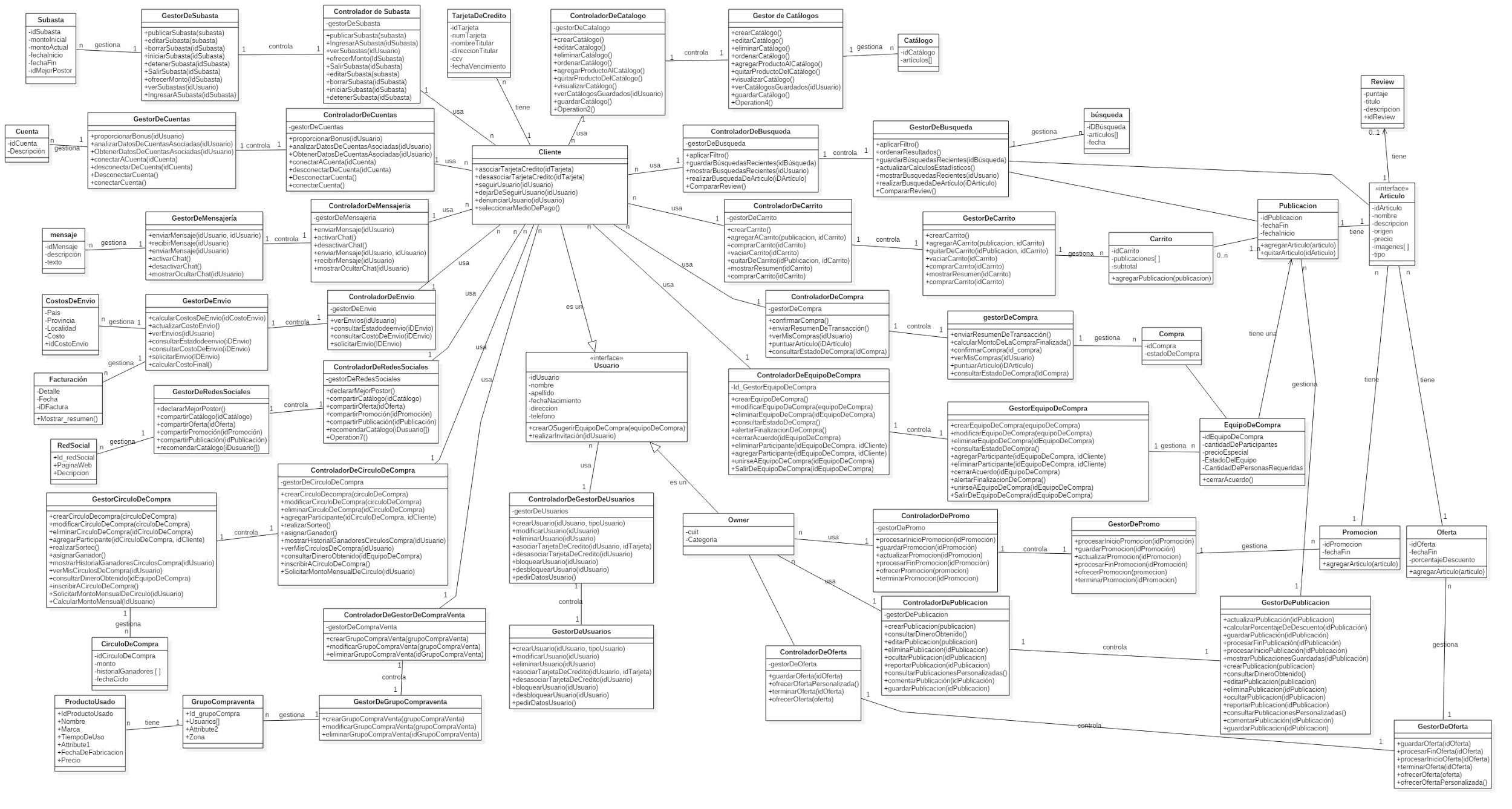
## Realizaciones de los Casos de Uso

Para ejemplificar un par de casos de uso relevantes al sistema, se han seleccionado el caso de uso “Comprar carrito” para el Diagrama de Secuencia y el escenario de “Comprar carrito exitosamente”.



# Vista Lógica

En la vista lógica se representan los requerimientos funcionales del sistema. Esta sección del documento debe describir las partes del diseño del modelo significativas para la arquitectura, tales como subsistemas y paquetes. En nuestro caso, presentaremos una visión estática del sistema a través de un Diagrama de Clases, detallando las características (atributos) y métodos (responsabilidades) de cada clase del negocio y de la aplicación.



## Visión General

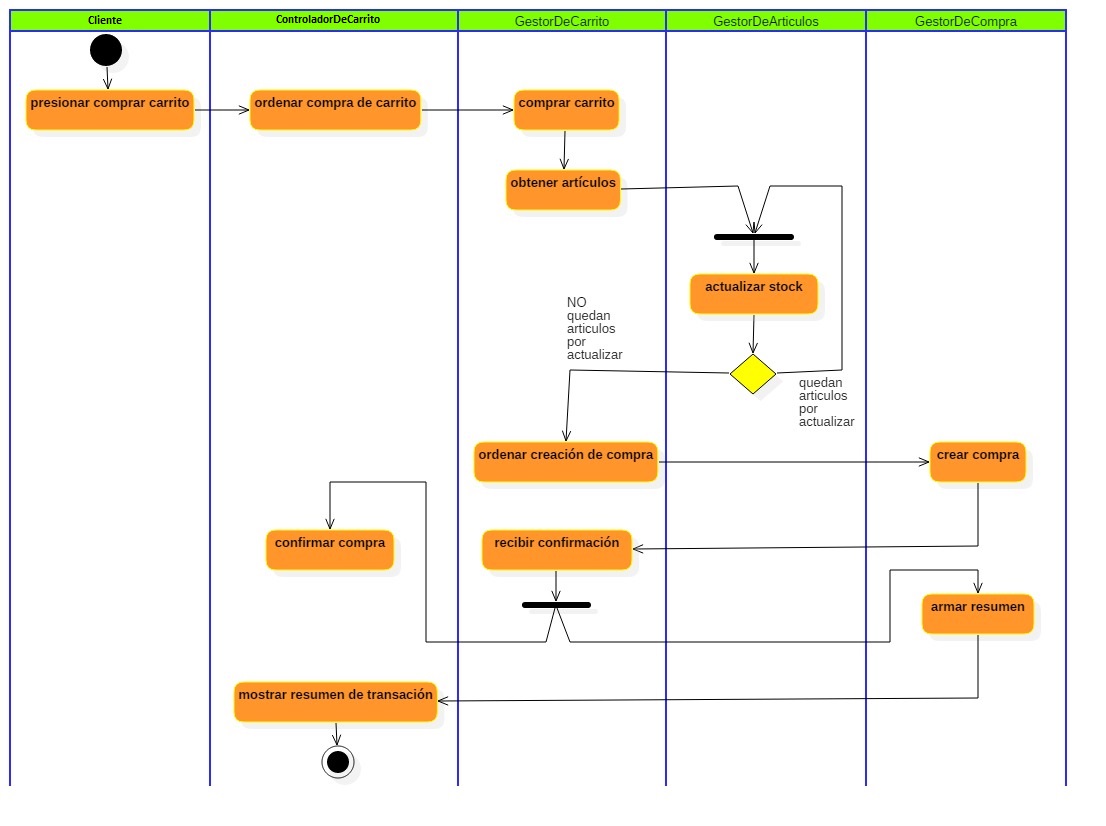
No aplicable.

## Paquetes de Diseño Significativos

No aplicable.

# Vista de Procesos

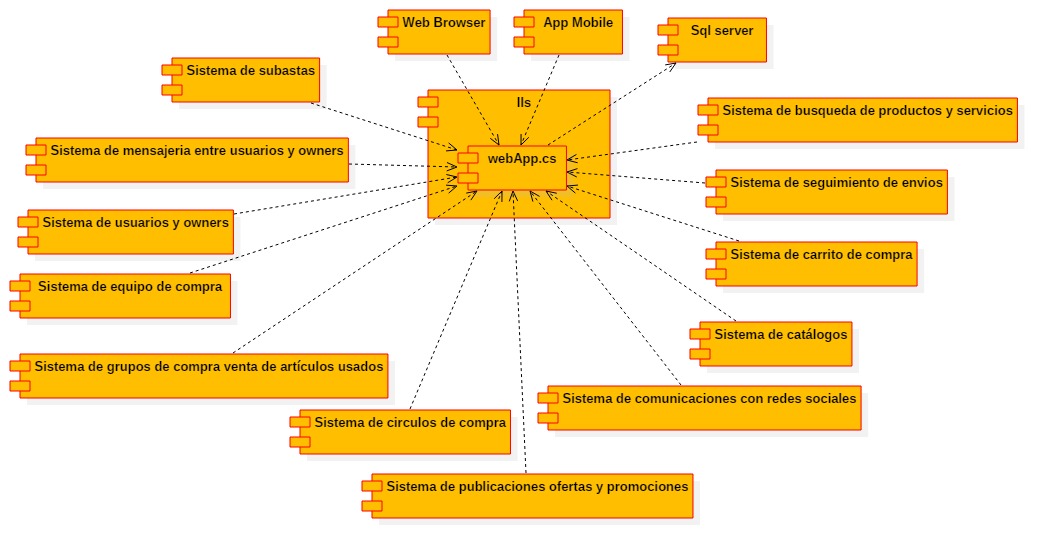
La vista de procesos permite ver el detalle de las acciones que componen un proceso a través de los Diagramas de Actividad. En nuestro caso, el siguiente diagrama representa el Caso de Uso “Comprar Carrito Exitosamente”.



# Vista de Despliegue

# Vista de Implementación

En esta vista, se observa el Diagrama de Componentes el cual indica la dependencia entre dichos componentes. Los componentes pueden ser archivos, bibliotecas compartidas, módulos, o paquetes.



## Visión General

No aplicable.

## Capas

No aplicable.

# Vista de Datos (opcional)

No aplicable.

# Tamaños y Performance

En esta sección se describen las características principales de la arquitectura de este producto software:

* La aplicación deberá permitir acceso simultáneo a 250.000 usuarios.
* El tiempo de respuesta promedio para que el usuario inicie sesión no superará los 5 segundos, sin contemplar el Hardware y Software de la máquina destino.
* El tiempo de procesamiento para responder, ante los eventos que involucren la tarjeta de crédito del cliente, no superará los 7 segundos.

# Calidad

* La aplicación debe ser compatible con cualquier dispositivo móvil con Chrome v35 y Firefox v57 o superiores.
* La aplicación debe ser compatible con cualquier sistema operativo de escritorio que use Chrome v35 y Firefox v57 o superiores.
* Debe existir una capa de seguridad que administre los permisos de los distintos tipos de usuario mostrando las interfaces adecuadas a cada uno.
* La misma capa de seguridad también asegura que usuarios no existentes accedan a las funcionalidades de los usuarios registrados al verificar los usuarios y contraseñas.
* Los datos sensibles de los usuarios deben estar encriptados con un algoritmo de cifrado en flujo.