**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA DE SISTEMAS



**Proyecto Final de Laboratorio**

**Sistemas Distribuidos**

**Autores**

CAUSHI CUEVA, YORDY DENIS (16200110)

RAMOS VARGAS, ROLANDO JAVIER (16200182)

TRUJILLO CRUZ, JOEL ANGEL (16200031)

VELASQUEZ IZQUIERDO, MIGUEL EDUARDO (16200101)

**Docente**

ERWIN MAC DOWALL REYNOSO

Lima – Perú

2020

Tabla de contenido

[Índice de Figuras 3](#_heading=h.30j0zll)

[**1.**](#_heading=h.3znysh7) **Introducción** 4

[**2.**](#_heading=h.2et92p0) **Marco Teórico** 5

[**3.**](#_heading=h.tyjcwt) **Especificación funcional** 7

[**3.1.**](#_heading=h.3dy6vkm) **Descripción del negocio** 8

[**3.2.**](#_heading=h.1t3h5sf) **Procesos del negocio** 9

[**3.3.**](#_heading=h.4d34og8) **Modelo de dominio** 9

[**3.4.**](#_heading=h.17dp8vu) **Actores** 10

[**3.5.**](#_heading=h.26in1rg) **Diagrama de casos de uso del sistema** 11

[**3.6.**](#_heading=h.lnxbz9) **Descripción de los casos de uso del sistema** 11

[6.6.1 Procesamiento de Órdenes 11](#_heading=h.35nkun2)

[6.6.1.1 Descripción 11](#_heading=h.1ksv4uv)

[6.6.1.2 Interfaz 12](#_heading=h.44sinio)

[6.6.2 Administración de inventario 13](#_heading=h.2jxsxqh)

[6.6.2.1 Descripción 13](#_heading=h.z337ya)

[6.6.3 Reserva 14](#_heading=h.3j2qqm3)

[6.6.3.1 Descripción 14](#_heading=h.1y810tw)

[6.6.4 Facturación 14](#_heading=h.4i7ojhp)

[6.6.4.1 Descripción 14](#_heading=h.2xcytpi)

[6.6.5 Cuentas por cobrar 15](#_heading=h.1ci93xb)

[6.6.5.1 Descripción 15](#_heading=h.3whwml4)

[6.6.5.2 Interfaz 16](#_heading=h.2bn6wsx)

[**4.**](#_heading=h.qsh70q) **Diseño técnico de la solución** 16

[**4.1.**](#_heading=h.3as4poj) **Vista Lógica** 16

[4.1.1.](#_heading=h.1pxezwc) Estilo Arquitectónico 16

[**4.2.**](#_heading=h.49x2ik5) **Vista de implementación** 18

[4.2.1.](#_heading=h.2p2csry) Descripción 18

[4.2.2.](#_heading=h.147n2zr) Diagrama de Componentes 18

[4.2.3.](#_heading=h.3o7alnk) Modelo de Datos 18

[**4.3.**](#_heading=h.23ckvvd) **Vista física** 19

# Índice de Figuras

7

[Figura 1: Modelo de servicio de cloud computing](#bookmark=id.2s8eyo1) 5

Figura 2: Esquema funcional 6

[Figura 3: Modelado de procesos](#_heading=h.ihv636)

7

Figura 4: Modelo de dominio

Figura 5: Actor del sistema

Figura 6: Diagrama de casos de uso

Figura 7: Estilo Arquitectónico del sistema

Figura 8: Diagrama de componente

Figura 9: Modelo de datos

Figura 10: Modelo de despliegue

# **Introducción**

A través del tiempo la tecnología ha estado evolucionando para dar solución a los diversos problemas que ocurren en la sociedad de la información, con el transcurso del tiempo algunas tecnologías quedan en desuso y aparecen nuevas que permiten solucionar de mejor manera los problemas que se tiene.

Uno de los grandes problemas que ocurre en la actualidad es la conectividad de sistemas heterogéneos ya sea a nivel de sistema operativos o lenguajes de programación, esto sucede debido a que las organizaciones no tienen un estándar general ya que cada organización implementa sus sistemas de acuerdo a lo que le es conveniente, esto acarrea un problema de comunicación entre sistemas, para solucionar esto aparecieron los middlewares que dan la facilidad de permitir la comunicación entre diferentes sistemas sin importar la tecnología que se ha utilizado para desarrollarlas.

En este presente informe se dará a conocer sobre el uso del middleware ZeroMQ en la cual se verá como este middleware logra que módulos desarrollados en diferentes lenguajes de comunicación y en diferentes sistemas operativos logran comunicarse entre sí a pesar de que los módulos se encuentran implementados en máquinas ubicadas en diferentes puntos de red.

* 1. **Antecedentes**

En la actualidad hay muchas organizaciones que tienen diferentes sistemas informáticos desarrollados para optimizar los diferentes procesos que se realizan dentro de ello. Por causas de que las organizaciones necesitan comunicarse con otras por el comercio electrónico o tramites que se necesitas, además de ello por la situación actual de la pandemia se necesita comunicar diferentes sistemas, debido a que estos sistemas son heterogéneos hay dificultad para que se logre hacer la comunicación entre estos, pero gracias al avance de la tecnología se puede realizar la comunicación mediante middleware que solucionan el problema de heterogeneidad tecnológica.

* 1. **Descripción del Problema**

Al tener implementados aplicaciones o módulos en diferentes tecnologías ocurre el problema de comunicación entre estas, ya que por las diferentes tecnologías que se utilizan no se pueden comunicar y eso es un gran problema esto ocurre porque las diferentes organizaciones desarrollan sus aplicaciones de acuerdo a las herramientas o facilidades que tienen a disposición para desarrollar sus aplicaciones.

* 1. **Objetivos**

Para el presente trabajo se ha considerado algunos objetivos, que deben de ser alcanzados a la culminación de este proyecto.

* + 1. **Objetivo General**

Demostrar la aplicabilidad de la utilización de la Tecnología de Middleware Orientado a Mensajes para soportar la ejecución sincronizada de procesos de negocio.

* + 1. **Objetivos Específicos**

Implementación de una herramienta de Middleware Orientado a Mensajes.

* 1. **Alcance del Proyecto**

En el presente documento abordaremos los casos de uso más importantes para el Sistema de Ventas de la empresa “Sistemas Distribuidos” en base al modelo de vistas 4+1 utilizando el estilo arquitectónico de 3 capas, además de mencionar los procesos de negocio de la empresa, modelo de datos e información adicional.

# **Marco Teórico**

**2.1 Middleware**

Es un software que se sitúa entre un sistema operativo y las aplicaciones que se ejecutan en él. Básicamente, funciona como una capa de traducción oculta para permitir la comunicación y la administración de datos en aplicaciones distribuidas.

A veces, se le denomina “plumbing” (tuberías), porque conecta dos aplicaciones para que se puedan pasar fácilmente datos y bases de datos por una “canalización”. El uso de middleware permite a los usuarios hacer solicitudes como el envío de formularios en un explorador web o permitir que un servidor web devuelve páginas web dinámicas en función del perfil de un usuario.

Algunos ejemplos comunes de middleware son el middleware de base de datos, el middleware de servidor de aplicaciones, el middleware orientado a mensajes, el middleware web y los monitores de procesamiento de transacciones. Cada programa suele proporcionar servicios de mensajería para que aplicaciones diferentes puedan comunicarse usando marcos de mensajería como el Protocolo simple de acceso a objetos (SOAP), servicios web, transferencia de estado representacional (REST) y notación de objetos JavaScript (JSON). Si bien todo el middleware desempeña funciones de comunicación, el tipo que elige una compañía depende del servicio que se va a usar y del tipo de información que debe comunicarse. Puede tratarse de autenticación de seguridad, administración de transacciones, colas de mensajes, servidores de aplicaciones, servidores web y directorios. El middleware se puede usar también para procesamiento distribuido con acciones que ocurren en tiempo real en lugar de enviar los datos para allá y para acá.

**2.2 ZeroMQ**

ZeroMQ (también escrito ØMQ, 0MQ o ZMQ) es una biblioteca de mensajería asíncrona de alto rendimiento, destinada a su uso en aplicaciones distribuidas o concurrentes. Proporciona una cola de mensajes, pero a diferencia del middleware orientado a mensajes, un sistema ZeroMQ puede ejecutarse sin un intermediario de mensajes dedicado.

ZeroMQ admite patrones de mensajería comunes (pub / sub, solicitud / respuesta, cliente / servidor y otros) a través de una variedad de transportes (TCP, en proceso, entre procesos, multidifusión, WebSocket y más), lo que hace que la mensajería entre procesos sea tan simple como mensajería entre hilos. Esto mantiene su código claro, modular y extremadamente fácil de escalar.

**2.3 Cloud Computing**

Conocida también como servicios en la nube es un paradigma que permite ofrecer y acceder a servicios de computación a través de una red, que usualmente es internet.

La computación en la nube es la disponibilidad a pedido de los recursos del sistema informático, especialmente el almacenamiento de datos y la capacidad de cómputo, sin una gestión activa directa por parte del usuario. El término se usa generalmente para describir los centros de datos disponibles desde cualquier lugar para muchos usuarios a través de Internet desde cualquier dispositivo móvil o fijo.

A menudo, el término «computación en la nube» se lo relaciona con una reducción de costos, disminución de vulnerabilidades y garantía de disponibilidad. Asimismo, la computación en la nube se la relaciona con un modelo de pago por uso. No obstante, el concepto de pago no puede ser solo relacionado con la erogación económica dado que solo aplica en caso de proveedores externos, y en muchos casos hace referencia a poder medir el consumo aplicándose a centros de costos de la propia empresa.

La computación en la nube es un nuevo modelo de prestación de servicios tecnológicos que impacta sin lugar a dudas en diversos negocios. Este modelo se apoya en infraestructuras tecnológicas dinámicas, caracterizadas por la virtualización de recursos, un alto grado de automatización, una elevada capacidad de adaptación para atender demandas variables.

Los modelos de servicios que brinda son tres: el software como servicio (SaaS, por sus siglas en inglés), plataforma como servicio (PaaS) e infraestructura como servicio (IaaS).

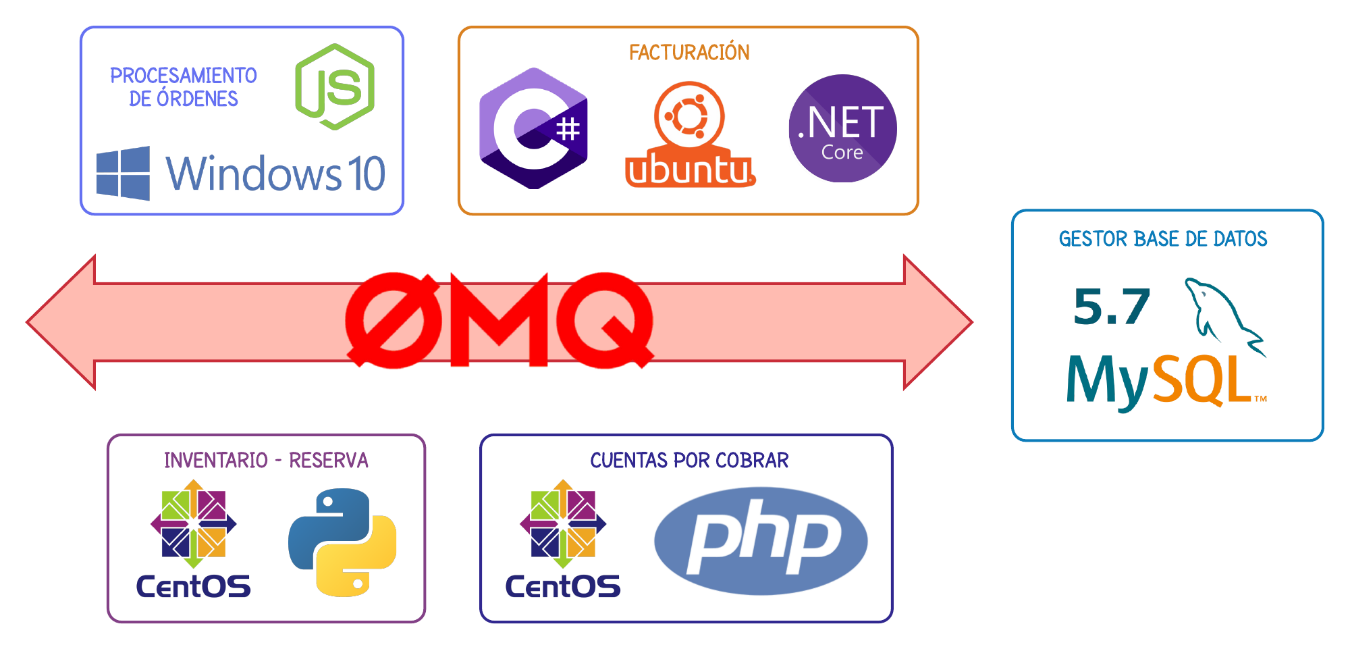
El software como servicio es un modelo de distribución de software en el que las aplicaciones están alojadas por una compañía o proveedor de servicio y puestas a disposición de los usuarios a través de una red, generalmente internet. La plataforma como servicio es un conjunto de utilitarios para abastecer al usuario de sistemas operativos y servicios asociados a través de Internet sin necesidad de descargas o instalación alguna. Infraestructura como servicio se refiere a la tercerización de los equipos utilizados para apoyar las operaciones, incluido el almacenamiento, hardware, servidores y componentes de red.

En el presente proyecto se ha utilizado el modelo de Infraestructura como servicio.



**Figura 1. Modelo de servicios de cloud computing**

# **Especificación funcional**



**Figura 2. Esquema funcional**

**Descripción de los módulos de software**

A continuación, se describe el funcionamiento básico de los módulos de software a desarrollar:

**1:Procesamiento de órdenes:**

Se ejecutan las siguientes acciones:

**i.** El cliente solicita una lista de ítems en la interfaz gráfica definida para tal efecto, que contiene los siguientes datos:

- imagen de los ítem.

- nombre de ítem.

- cantidad requerida.

**ii.** Se “empaqueta” los datos en un mensaje y se coloca este en el bus de mensajes.

**2:Inventario.**

Se ejecutan las siguientes acciones:

**i.** El módulo recibe el mensaje enviado por el módulo de Procesamiento de Órdenes y “desempaqueta” el mismo para extraer los datos definidos en el punto anterior.

**ii.** Se valida contra la Base de Datos la existencia de los ítems solicitados y la cantidad requerida.

**iii.** En caso no exista el ítem o no se disponga de la cantidad requerida se envía un mensaje de error al módulo de Procesamiento de Órdenes para ser entregado al cliente. Fin del módulo.

**iv.** En caso de que existan el ítem y la cantidad requerida, se envía un mensaje al módulo de Reserva para “reservar” cantidad requerida de los ítems. Fin de módulo.

**3:Reserva.**

Se ejecutan las siguientes acciones:

**i.** El módulo recibe el mensaje enviado por el módulo de Inventario y “desempaqueta” el mismo para extraer los datos definidos en el punto anterior.

**ii.** Ejecuta la “reserva”, esto es disminuye la cantidad de ítems existente por cada ítem. Actualiza la Base de Datos la cantidad existente de los ítems solicitados.

**iii.** Se envía un mensaje al módulo de Facturación para generar la respectiva factura para su envío posterior al cliente. Fin de módulo.

**4:Facturación.**

Se ejecutan las siguientes acciones:

i. El módulo recibe el mensaje enviado por el módulo de Reserva y “desempaqueta” el mismo para extraer los datos.

ii. Genera la factura y actualiza la Base de Datos de Facturas emitidas. Fin de módulo.

**5:Cuentas x Cobrar.**

Se ejecutan las siguientes acciones:

**i.** El módulo recibe el mensaje enviado por el módulo de Procesamiento de órdenes y “desempaqueta” el mismo para extraer los datos.

**ii.** Genera el registro de Cuenta x Cobrar y actualiza la Base de Datos de Facturas.Fin de módulo.

## **Descripción del negocio**

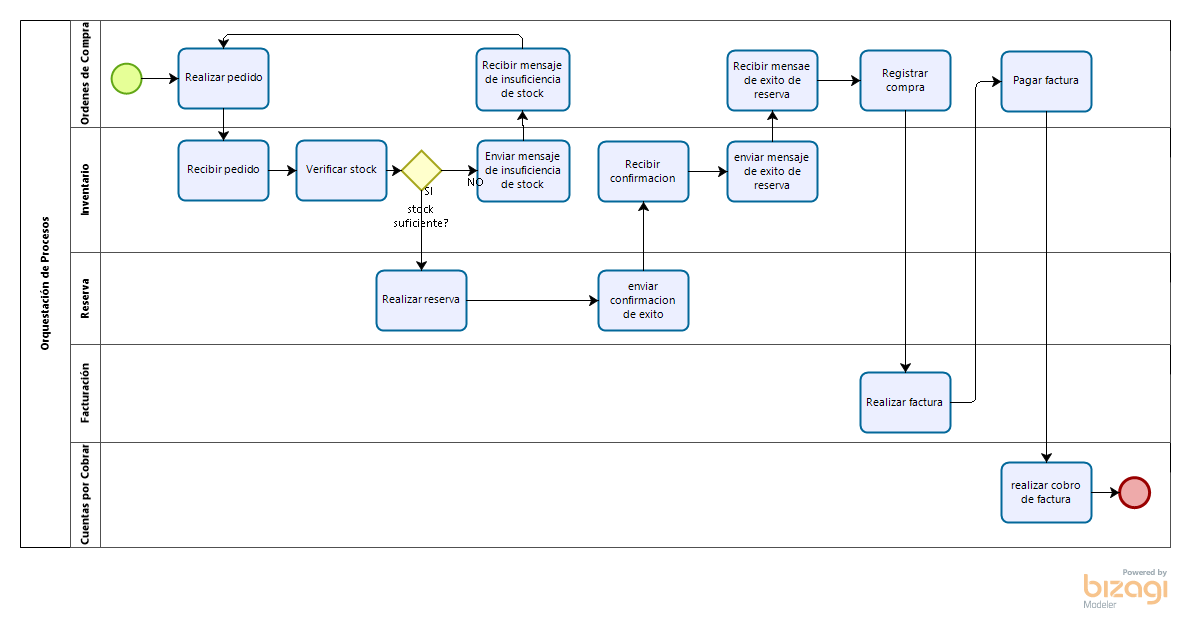
La empresa “Sistemas Distribuidos” está brindando un servicio de ventas de productos para todas las personas interesadas en realizar compras a un bajo costo. La persona interesada se acerca al local para poder realizar su orden de compra.

La orden de compra es un documento que acredita que el cliente ha aceptado los términos y condiciones que impone la empresa para la venta de sus productos; la estructura de la orden de compra consta de los datos del cliente (Dni, nombre), los productos elegidos, cantidad, precios unitarios y precio total.

Una vez generada la orden de compra, el módulo de inventarios verifica el estock de los productos que se solicitan en la orden de compra; si el inventario verifica que todos los productos, que se indican en la orden de compra, están en stock el módulo de reserva pasa a separar la cantidad de productos solicitada, y manda un mensaje de confirmación al módulo de inventario, luego el módulo de factura registra la factura de los productos que se pidió con su respectivo monto. En aquella factura queda registrada la deuda sino fue pagada totalmente y el cliente tiene la posibilidad de pagarla otro día para eso en el módulo de procesamiento de orden tiene la opción de pagar la deuda con el monto que va a aportar y digitando su DNI. El módulo de cuentas por cobrar capta esos datos y actualiza la base de datos con aquella información.

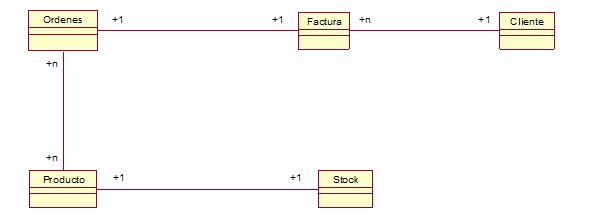
## **Procesos del negocio**

En la figura 2 se muestra el diagrama de los procesos que sigue la empresa “Sistemas Distribuidos” para la ejecución de sus órdenes de compra.



**Figura 3. Modelado de proceso**

## **Modelo de dominio**



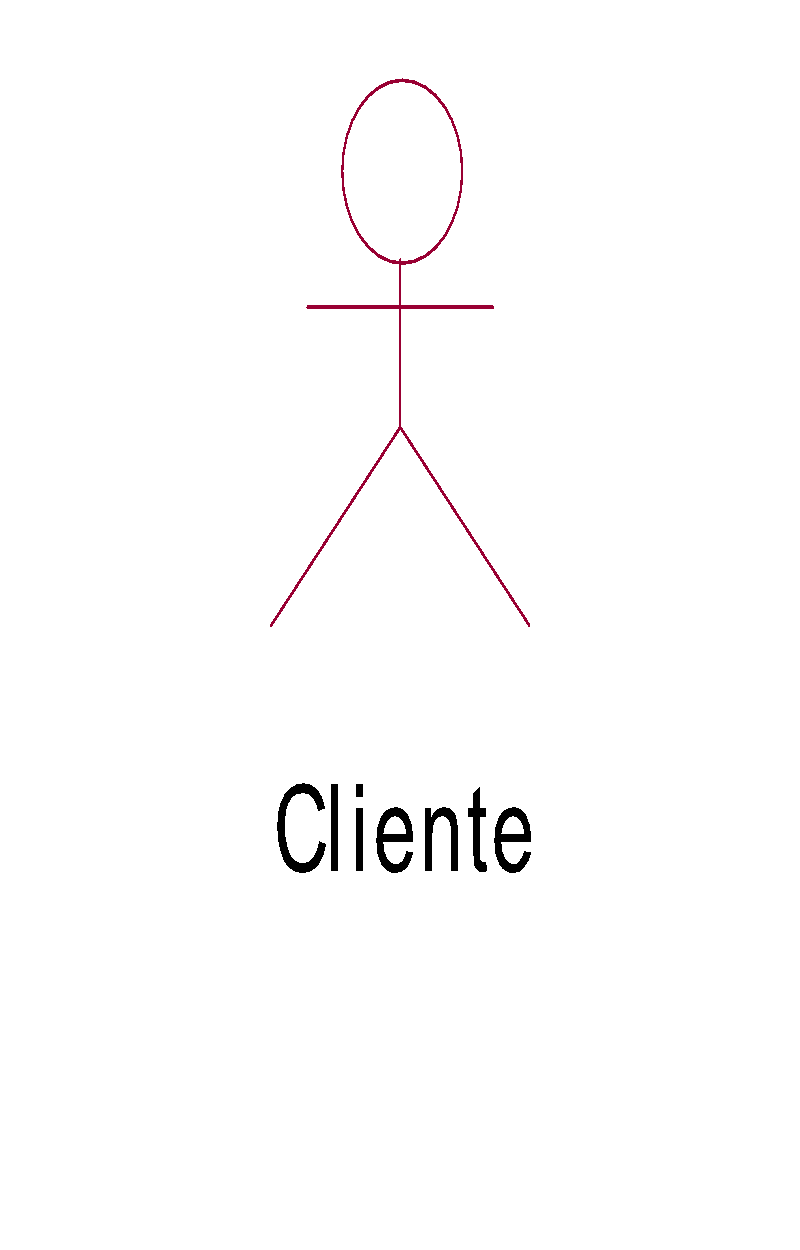
**Figura 4. Modelo de Dominio**

**Descripción:**

* La orden de compra es generada por el cliente; y puede contener uno o más productos que el cliente desea comprar.
* El cliente puede tener una o varias órdenes de compra.
* La factura es el comprobante de pago que está asignada a una sola orden de compra.
* El inventario es el único que se encarga de gestionar el stock de los productos y a su vez de reservar uno o más productos según la orden de compra.

## **Actores**

En esta sección, se describirán los actores que interactúan con el sistema propuesto como parte del proyecto para solucionar el problema descrito anteriormente. Los actores se pueden apreciar en la siguiente figura:



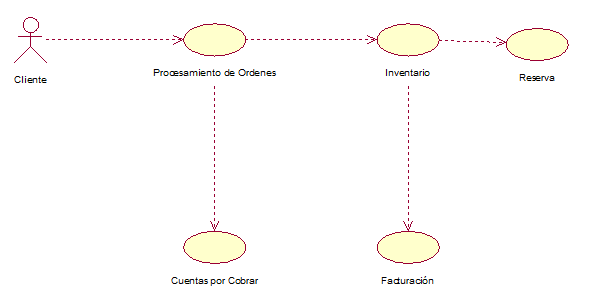
**Figura 5. Actor del sistema**

En la figura 5, se muestran el actor que interactúan con el sistema propuesto y su respectiva descripción se realizará a continuación:

1. Cliente: Es cualquier persona que desee comprar productos de la empresa, para esto podrá ingresar los productos y la cantidad que desee para luego poder enviar su orden de compra y después de que la orden de compra pase por el proceso de reserva y de facturación se procederá a realizar el pago respectivo.

## **Diagrama de casos de uso del sistema**

A continuación en la figura 6 se muestra el diagrama de casos de uso del sistema, se observa que el usuario cliente tendrá acceso de vista con el caso de uso de Procesamiento de órdenes, este caso de uso internamente tiene comunicación con el caso de uso de administración de inventario para validar el stock disponible en la tienda, automáticamente se realizará la reserva de los productos en caso no pase en stock, luego el cliente deberá aprobar la factura y esta pasará al caso de uso de cuentas por cobrar, el encargado de cuentas por cobrar tendrá será el que tenga la vista de este caso de uso.

**

**Figura 6. Diagrama de casos de Uso**

## **Descripción de los casos de uso del sistema**

### 6.6.1 Procesamiento de Órdenes

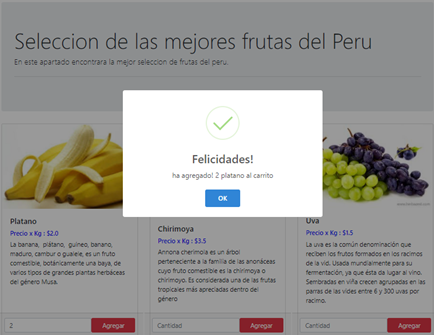
#### 6.6.1.1 Descripción

|  |  |
| --- | --- |
| **Caso de Uso** | Procesamiento de órdenes |
| **Objetivo** | Generar orden de compra |
| **Actores** | Cliente |
| **Precondiciones** | - El cliente debe encontrarse logueado en el sistema.  - La base de datos del sistema está en línea |
| **Flujo Principal** | |
| 1. El CUS inicia cuando el cliente ingresa al sistema.  2. El sistema muestra un catálogo en la cual muestra los siguiente: “nombre del producto”, “precio”, “descripción”, además los botones: “Agregar” y “Cotizar”, un campo para ingresar la “cantidad” en la cual el cliente realiza el pedido de la cantidad que desea comprar.  3. El sistema enviará la solicitud de pedido de los productos seleccionados.  4. El sistema validará los datos ingresados mostrando una ventana en la cual muestra la cantidad que ha agregado.  5.El cliente cierra la ventana seleccionando el botón “OK”.  6.El cliente selecciona el botón “cotizar” para ver los productos que ah pedido.  7. El CUS finaliza. | |
| **Post-condición** | Orden de compra registrada |

#### 6.6.1.2 Interfaz

- Flujo principal







### 6.6.2 Administración de inventario

#### 6.6.2.1 Descripción

|  |  |
| --- | --- |
| **Caso de Uso** | Administración de inventario |
| **Objetivo** | Verificar si la cantidad solicitada por el comprador se encuentra disponible en inventario. |
| **Actores** |  |
| **Precondiciones** | El comprador debe seleccionar los productos que desea ordenar. |
| **Flujo Principal** | |
| 1. El CUS inicia cuando el sistema recibe la cantidad solicitada por el comprador  2. El sistema consulta la base de datos y verifica si la cantidad solicitada se encuentra disponible.  3. El sistema envía mensaje de aprobación.  4. El CUS finaliza. | |
| **Flujo Alternativo** | |
| En el paso 2:  1. En caso la cantidad solicitada no se encuentre disponible, el sistema enviará un mensaje de rechazo a la orden. | |
| **Post-condición** | Enviar mensaje de confirmación o rechazo de la cantidad solicitada. |

### 6.6.3 Reserva

#### 6.6.3.1 Descripción

|  |  |
| --- | --- |
| **Caso de Uso** | Reserva |
| **Objetivo** | Mantener en reserva la cantidad de productos solicitados por el comprador |
| **Actores** |  |
| **Precondiciones** | Los productos seleccionados por el comprador deben encontrarse disponibles. |
| **Flujo Principal** | |
| 1. El CUS inicia cuando el sistema recibe el mensaje de aprobación a la cantidad solicitada por el comprador.  2. El sistema reservará la cantidad solicitada.  3. El sistema enviará la información del orden de compra, en caso el comprador haya realizado la confirmación de la orden.  4. El CUS finaliza. | |
| **Post-condición** | Reserva del producto solicitado y actualización de inventario. |

### 6.6.4 Facturación

#### 6.6.4.1 Descripción

|  |  |
| --- | --- |
| **Caso de Uso** | Facturación |
| **Objetivo** | Registro de la compra |
| **Actores** |  |
| **Precondiciones** | Confirmación de la orden de compra. |
| **Flujo Principal** | |
| 1. El CUS inicia cuando el sistema obtiene el mensaje de confirmación de la orden de compra.  2. El sistema valida el pago realizado por el comprador.  3. El sistema realiza el registro de la compra.  4. El sistema envía mensaje de confirmación de la compra realizada.  5. El sistema envía la información de la compra.  6. El CUS finaliza. | |
| **Post-condición** | Registro de la compra |

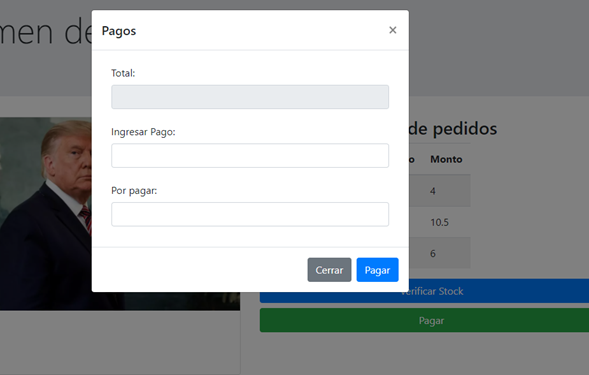
### 6.6.5 Cuentas por cobrar

#### 6.6.5.1 Descripción

|  |  |
| --- | --- |
| **Caso de Uso** | Cuentas por cobrar |
| **Objetivo** | Visualizar la relación de ventas realizadas. |
| **Actores** | cliente |
| **Precondiciones** | El cliente debe encontrarse loggeado en el sistema. |
| **Flujo Principal** | |
| 1. El CUS inicia cuando el cliente ingresa al sistema.  2. El sistema muestra la opción para pagar factura.  3. El cliente ingresa su DNI y el monto que va a pagar.  4. El sistema coge los datos y actualiza la base de datos.  5. El CUS finaliza. | |
| **Post-condición** | - El cliente logra realizar el pago de su deuda. |

#### 6.6.5.2 Interfaz

- Flujo principal



# **Diseño técnico de la solución**

## **Vista Lógica**

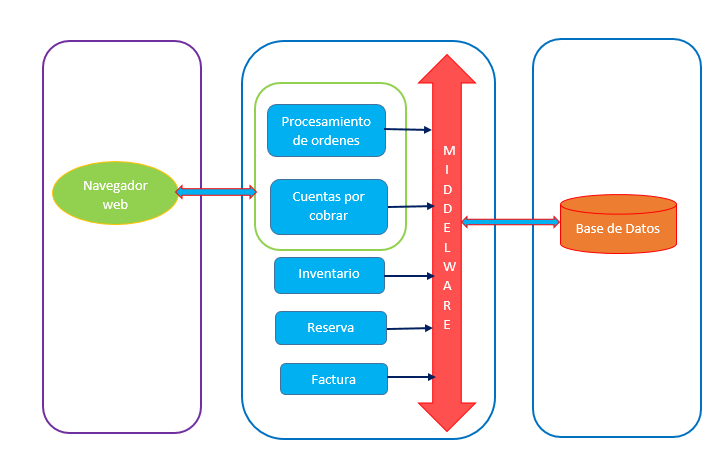
### Estilo Arquitectónico

El estilo arquitectónico del sistema se presenta en la figura 4 es el modelo de 3 capas (capa de presentación, lógica de negocios, capa de datos).

La capa de presentación de nuestro sistema va a ser una interfaz web, que va a presentar la vista de orden de compra.

En la capa de lógica de negocios será implementada usando el middleware que a través de paso de mensajes permitirá la conexión entre los distintos módulos del sistema.

En la capa de datos estarán implementadas las bases de datos necesarios para el desarrollo del sistema, que son la base de datos de productos, donde se guarda el catálogo de productos y su stock, la base de datos de órdenes de compra se guardaran las órdenes de compra, y la base de datos de factura donde se guardaran las órdenes de compra del sistema.

**Figura 7. Estilo Arquitectónico del sistema**

**Capa de presentación**

La que ve el usuario (también se la denomina «capa de usuario»), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser «amigable» (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

**Capa de Negocio o Lógica de la aplicación**

Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

**Capa de Persistencia o Datos**

Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

## **Vista de implementación**

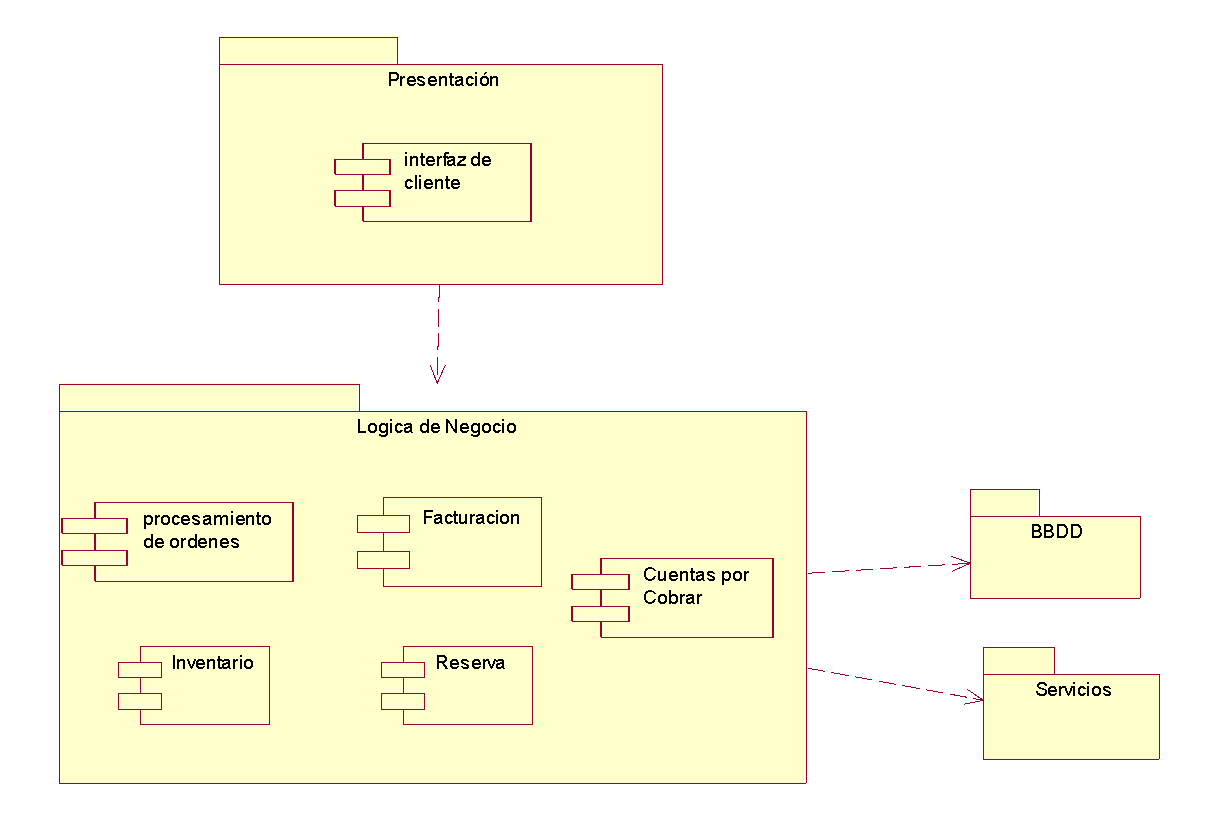
### Descripción

En esta vista de implementación se presenta el sistema en términos de componentes, es decir ficheros de código fuente. Nos enfocaremos en la organización de los módulos de software.

Se ha decidido separar en cinco módulos: Procesamiento de Órdenes (abarca cada una de las actividades del paquete Procesamiento de Órdenes), Administración de Inventario (abarca cada una de las actividades del paquete de Administración de Inventario), Reserva (abarca cada una de las actividades del paquete Reserva), Facturación (abarca cada una de las actividades del paquete Facturación) y Cuentas por Cobrar (abarca cada una de las actividades del paquete Cuentas por Cobrar).

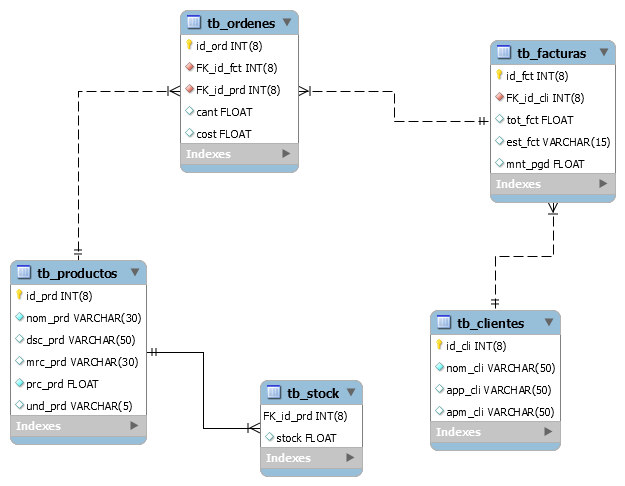
En esta vista también se explicará cual es la relación entre los componentes y la clase de diseño de cada módulo.

### Diagrama de Componentes



**Figura 8. Diagrama de componente**

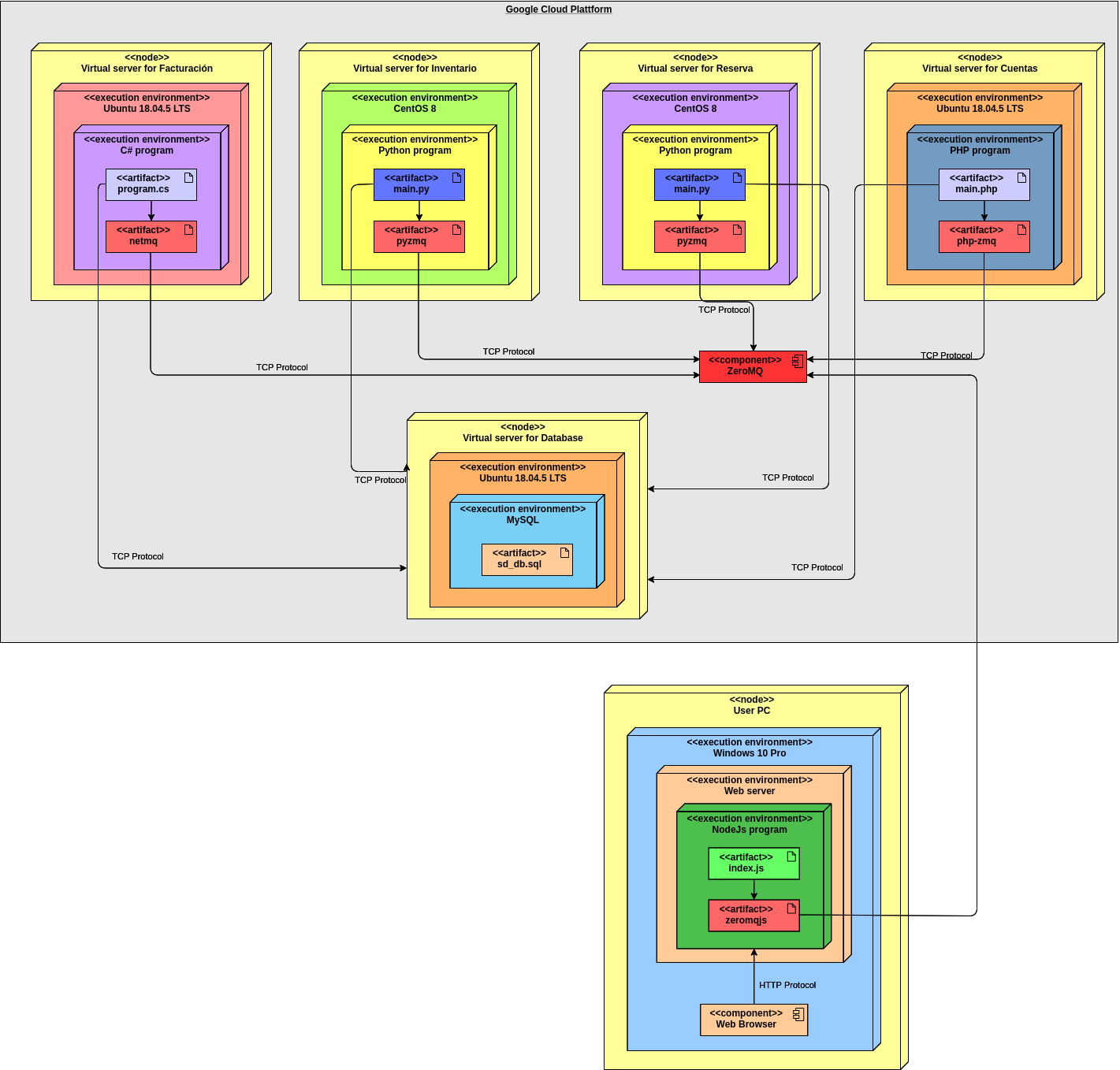
### Modelo de Datos



**Figura 9. Modelo de Datos**

## **Vista física**

En esta sección, se describe la vista física del sistema de gestión de órdenes de compra permitiendo tener una visión general de cómo estará ubicado cada componente del sistema y también mostrar las interacciones existentes entre los componentes, la vista física del sistema se muestra en la siguiente figura:



**Figura 10. Modelo de Despliegue**

En la figura10, se muestra el diagrama de despliegue del sistema de gestión de órdenes de compra que a su vez está compuesto de los componentes respectivos de las interacciones existentes; a continuación, se describe cada componente del diagrama de despliegue:

* Computadora del comprador: Es una computadora de escritorio o laptop que tenga conexión a Internet y tenga un navegador web de preferencia Google Chrome.
* Middleware ZeroMQ: Es un servidor que contiene el middleware orientado a mensajes “ZeroMQ”, que es el encargado de actuar como un intermediario para poder solucionar el problema de heterogeneidad de los sistemas permitiendo la interoperabilidad.
* Servidor de base de datos: Es un servidor que contiene la base de datos que usarán los demás componentes para leer, escribir y modificar; este servidor tiene el sistema operativo Ubuntu y debe tener instalado MySQL 5.7.
* Servidor de procesamiento de órdenes de compra: Es un servidor que contiene el módulo de procesamiento de órdenes de compra, este es el servidor debe mostrar en la computadora del comprador una interfaz que le muestre los campos que ingresara los productos y su cantidad y le permita generar la orden de compra para que sea verificada en el servidor de administración de inventario si se tiene todos los productos de la orden de compra. Este servidor debe tener el sistema operativo Windows 10 y debe instalar JavaScript.
* Servidor de administración de inventario: Es un servidor que contiene el módulo de administración de inventario, este servidor tiene la funcionalidad de verificar el stock para saber si la orden de compra procede o se debe modificar o cancelar si el comprador lo desea; además el servidor debe interactuar con el servidor de reserva para que se reserven los productos de la orden de compra si ha procedido. Este servidor debe tener el sistema operativo Centos y debe tener instalado Python.
* Servidor de reserva: Es un servidor que contiene el módulo de reserva, este servidor tiene funcionalidad de reservar la cantidad de un producto respectivo según una orden de compra; además el servidor debe interactuar con el servidor de facturación para que se realice la facturación de la respectiva orden de compra. Este servidor debe tener el sistema operativo Centos y debe tener instalado Python.
* Servidor de facturación: Es un servidor que contiene el módulo de facturación, este servidor tiene funcionalidad de realizar la facturación de una orden de compra según las compras que realizan los compradores y también debe de enviar la factura al comprador. Este servidor debe tener el sistema operativo Ubuntu y debe tener instalado C# de Microsoft .NET.
* Servidor de cuentas por cobrar: Es un servidor que contiene el módulo de cuentas por cobrar, este servidor tiene la funcionalidad de mostrar las cuentas que están pendientes por cobrar de las órdenes de compra que realizaron los compradores; además tiene la funcionalidad de realizar el pago respectivo. Este servidor debe tener el sistema operativo Centos y debe tener instalado PHP.
* Computadora del módulo de inventario y reserva: Es una computadora de escritorio o laptop que tenga conexión a Internet.

1. **Conclusiones**

Gracias a la tecnología middleware se puede realizar la comunicación entre diferentes aplicaciones heterogéneas sin importar el sistema operativo o el lenguaje de programación donde estén implementados, gracias a esto se puede probar que es una gran solución que puede ser aplicado en la vida real ya que ya organizaciones que no tienen comunicadas sus distintas aplicaciones.