

Módulos integrados. GA8-220501096-AA1-EV02.

Por

Oscar Danilo Sanchez Aguilar

Sena

Tecnología en Analisis y Desarrollo de Software

Ficha 2721519

Instructor: Milton Ivan Barbosa

## Tabla de contenido

Introducción .....	¡Error! Marcador no definido.
<b>Justificación .....</b>	<b>4</b>
<b>Objetivo General .....</b>	<b>5</b>
<b>DESARROLLO DE LA ACTIVIDAD .....</b>	<b>6</b>
<b>Determinar los frameworks a en cada capa de la aplicación. ....</b>	<b>36</b>
<b>Librerías necesarias en cada etapa.....</b>	<b>37</b>
<b>Frameworks para cada etapa .....</b>	<b>38</b>
<b>Conclusiones .....</b>	<b>40</b>
<b>Bibliografía .....</b>	<b>41</b>

## Introduction

La presentación en esta evidencia consiste en el desarrollo de una página web para la venta de gorros hechos a mano en materia de lana, la información y en resumen es similar al proyecto presentando en donde explicaba desarrollar el software a partir de la integración de sus módulos componentes.

La idea de crear este software surge como una oportunidad para aplicar los conocimientos adquiridos en la carrera de Analista y Desarrollador de Software, y al mismo tiempo, apoyar el emprendimiento familiar, facilitando la comercialización de estos productos a través de una plataforma en línea.

## **Justificación**

Hoy en día el mundo ha avanzado a pasos agigantados el tema de la tecnología el contar con una presencia en línea es esencial para cualquier negocio, incluso para aquellos que se dedican a la producción artesanal. La creación de una página web no solo permitirá ampliar el alcance del mercado para estos gorros, sino que también brindará una experiencia de compra más conveniente para los clientes. Además, este proyecto representa una excelente oportunidad para aplicar mis conocimientos y habilidades técnicas en un contexto real, lo que contribuirá al desarrollo profesional y laboral en el área de análisis y desarrollo de software.

## **Objetivo General**

Desarrollar una página web de ventas de gorros funcional y atractiva que permita la venta de gorros hechos a mano de lana, facilitando la gestión de productos, el proceso de compra para los usuarios, y el apoyo al emprendimiento familiar.

## **Objetivos Específicos**

1. Planificar una interfaz de usuario intuitiva y visualmente atractiva que represente adecuadamente la naturaleza artesanal de los productos ofrecidos.
2. Hacer una página visualmente atractiva página para cada tipo de usuario.
3. Ejecutar o poner en marcha la página, el cual funcione en cualquier dispositivo, que le sea de fácil acceso el usuario.

## DESARROLLO DE LA ACTIVIDAD

### Los requerimientos del sistema.

Tipo de Requerimiento	Funcionalidad	Descripción	Prioridad
Requerimiento Funcional	Registro de Usuario	El sistema debe permitir a los usuarios crear una cuenta proporcionando información básica como nombre, correo, y contraseña.	Alta
Requerimiento Funcional	Inicio de Sesión	El sistema debe permitir a los usuarios iniciar sesión utilizando su correo electrónico y contraseña.	Alta
Requerimiento Funcional	Carrito de Compras	El sistema debe permitir a los usuarios agregar productos al carrito, modificar cantidades y eliminar productos.	Alta
Requerimiento Funcional	Navegabilidad	El sistema debe ser fácil de navegar, permitiendo acceso a la página principal, catálogo, carrito y perfil de usuario.	Alta
Requerimiento No Funcional	Usabilidad	La interfaz de usuario debe ser intuitiva, facilitando la realización de tareas sin asistencia.	Alta
Requerimiento No Funcional	Mantenibilidad	El código debe estar bien documentado y estructurado, facilitando el mantenimiento y futuras mejoras.	Alta

**Presentación de la URL:**

Los módulos de mi proyecto de software de ventas de gorros realizados de manera manual son:

Iniciar sesión

[Inicio de Sesión](#)

Agregar al carro

[Tienda de Gorritos](#)

Registrarse

[Bienvenido a Tienda de Gorritos CAMIPIA](#)

Manejar los documentos de casos de uso o historias de usuario.

## CASOS DE USOS

### Caso de Uso 1: Registro de Usuario

- **Actor Primario:** Usuario no registrado
- **Precondiciones:** El usuario no debe tener una cuenta existente.
- **Flujo Principal:**
  1. El usuario accede a la página de registro.
  2. El usuario introduce su nombre, correo electrónico y contraseña.
  3. El sistema valida la información y crea una nueva cuenta.
  4. El sistema envía una confirmación al correo electrónico del usuario.



- **Flujo Alternativo:**

Si el correo electrónico ya está registrado, el sistema muestra un mensaje de error.

## **Caso de Uso 2: Inicio de Sesión**

**Actor Primario:** Usuario registrado

**Precondiciones:** El usuario debe tener una cuenta registrada.

**Flujo Principal:**

1. El usuario accede a la página de inicio de sesión.
2. El usuario introduce su correo electrónico y contraseña.
3. El sistema valida las credenciales.
4. El usuario accede a su perfil.

**Flujo Alternativo:**

Si las credenciales son incorrectas, el sistema muestra un mensaje de error.

## **Caso de Uso 3: Navegación por el Catálogo**

**Actor Primario:** Usuario

**Precondiciones:** El usuario ha iniciado sesión o es un visitante.

**Flujo Principal:**

1. El usuario accede al catálogo de productos.
2. El usuario navega por la lista de productos disponibles.
3. El usuario selecciona un producto para ver más detalles.

**Flujo Alternativo:**

- Ninguno.

Caso de Uso 4: Agregar Productos al Carrito

- **Actor Primario:** Usuario
- **Precondiciones:** El usuario ha iniciado sesión o es un visitante.
- **Flujo Principal:**
  1. El usuario selecciona un producto en el catálogo.
  2. El usuario hace clic en "Agregar al carrito".
  3. El sistema añade el producto al carrito de compras.
- **Flujo Alternativo:**
  - Ninguno.

Caso de Uso 5: Administración de Productos

- **Actor Primario:** Administrador
- **Precondiciones:** El administrador ha iniciado sesión.
- **Flujo Principal:**
  1. El administrador accede a la página de gestión de productos.
  2. El administrador selecciona "Agregar nuevo producto" o "Editar" o "Eliminar" en un producto existente.
  3. El sistema guarda los cambios y actualiza el catálogo de productos.
- **Flujo Alternativo:**
  - Si hay un error al guardar los cambios, el sistema muestra un mensaje de error.

### **Conocer el funcionamiento del IDE de desarrollo.**

Para realizar el proyecto de CAMIPIA, elegí **NetBeans 21**, porque es una herramienta robusta y ampliamente utilizada en el desarrollo de aplicaciones Java, lo que se alinea perfectamente con los objetivos de mi proyecto. Además, fue la herramienta recomendada por la mayoría de las personas que trabajamos en este medio de desarrollo software, lo que asegura que estoy utilizando un entorno compatible con los requerimientos de mi curso y con el soporte adecuado.

**NetBeans 21** es la última versión del entorno de desarrollo integrado (IDE) de Apache NetBeans, una herramienta poderosa y versátil que permite desarrollar aplicaciones en varios lenguajes de programación, como Java, PHP, JavaScript, y más. NetBeans ofrece un conjunto completo de herramientas para el desarrollo de software, que incluye un editor de código avanzado, depuración integrada, soporte

para proyectos Maven y Gradle, y herramientas de diseño de interfaces gráficas (GUI).

Aunque está optimizado para Java, NetBeans soporta múltiples lenguajes, lo que lo hace versátil para proyectos diversos.

Se debe conocer el diagrama de clases.

Diagrama de Clases

## **Clases Principales:**

### **1. Usuario**

- **Atributos:**
  - idUsuario: int
  - nombre: String
  - email: String
  - contraseña: String
- **Métodos:**
  - registrarse ()
  - iniciar Sesión()

## 2. Producto

- **Atributos:**
  - id Producto: int
  - nombre: String
  - descripción: String
  - precio: double
  - Cantidad: int
- **Métodos:**
  - obtener Detalle Producto ()

## 3. Carrito De Compras

- **Atributos:**
  - Id Carrito: int
  - productos: List<Producto>
  - total: double
- **Métodos:**
  - Agregar Producto (Producto)
  - Eliminar Producto (Producto)
  - Calcular Total()

## 4. Navegación

- **Atributos:**
  - secciones: List<String>
- **Métodos:**
  - Navegar A (sección: String)

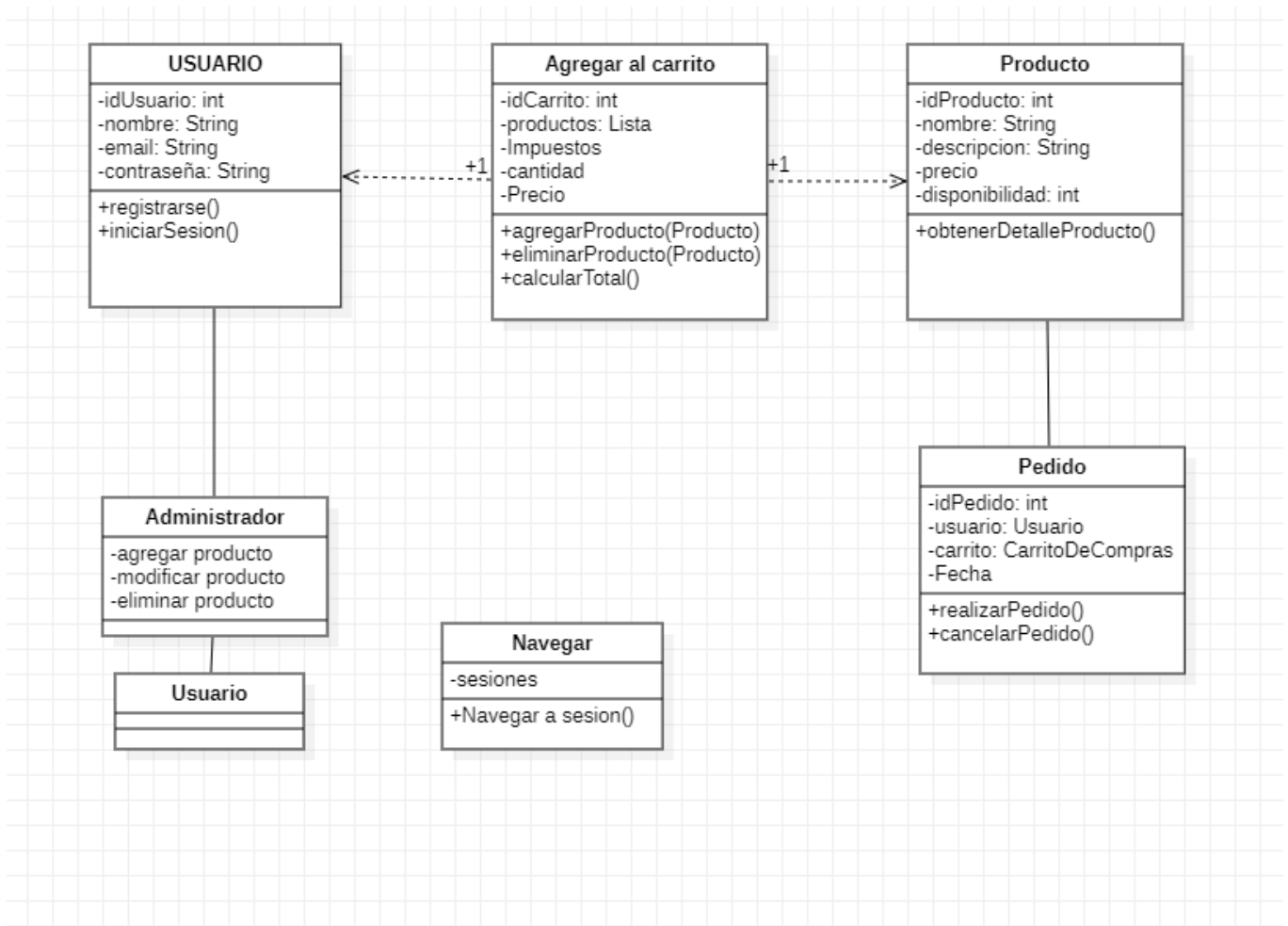
## 5. Administrador (Extiende de Usuario)

### ○ Métodos:

- Agregar Producto ()
- Modificar Producto ()
- Eliminar Producto ()

Figura de diagrama de clases

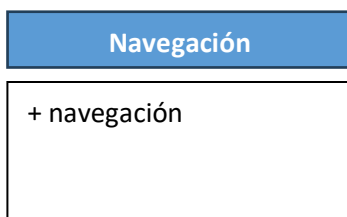
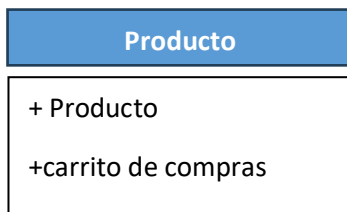
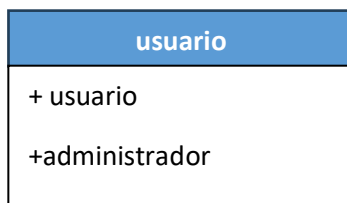
Ilustración 1 Diagrama de Clases

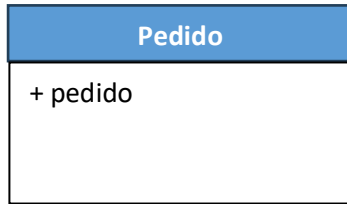


- **Se debe conocer el diagrama de paquetes.**

Representación del Diagrama de Paquetes:

A continuación, te muestro cómo se organizarían las clases dentro de los paquetes.





Una breve explicación del diagrama de paquetes anteriormente ilustrado

#### **Paquete Usuario:**

- Agrupa las clases que manejan la autenticación, registro, y permisos de los usuarios.
- Usuario: Maneja las operaciones básicas de un usuario normal.
- Administrador: Extiende Usuario y agrega funcionalidades adicionales para la gestión de productos.

#### **Paquete Producto:**

- Contiene las clases que se encargan de la gestión de los productos y la interacción del usuario con ellos.
- Producto: Representa los gorros en la tienda.
- Carrito de Compras: Maneja la adición y gestión de productos seleccionados para la compra.

#### **Paquete Pedido:**

- Incluye las clases que gestionan el ciclo de vida de un pedido, desde la selección de productos hasta la confirmación de la compra.



- Pedido: Clase que representa un pedido realizado por un usuario.

**Paquete Navegación:**

- Contiene las clases que permiten la navegabilidad en la página web.
- Navegación: Facilita la interacción del usuario con diferentes secciones del sitio.

Se debe conocer el diagrama de componentes.

Para el proyecto de la página web de ventas de gorros "CAMIPIA", el diagrama de componentes podría incluir componentes como la interfaz de usuario, el backend, la base de datos, y los diferentes servicios que interactúan entre sí.

**Componentes Principales:****Interfaz de Usuario (UI)**

Representa la capa frontal del sistema, donde los usuarios interactúan con la aplicación.

**Subcomponentes:**

- Página de registro, inicio de sesión, y navegación por el catálogo.
- Gestión del carrito.

**Controlador Web (Servlets)**

Se maneja las solicitudes del usuario desde la UI y coordina con otros componentes para procesar esas solicitudes.

**Subcomponentes:**

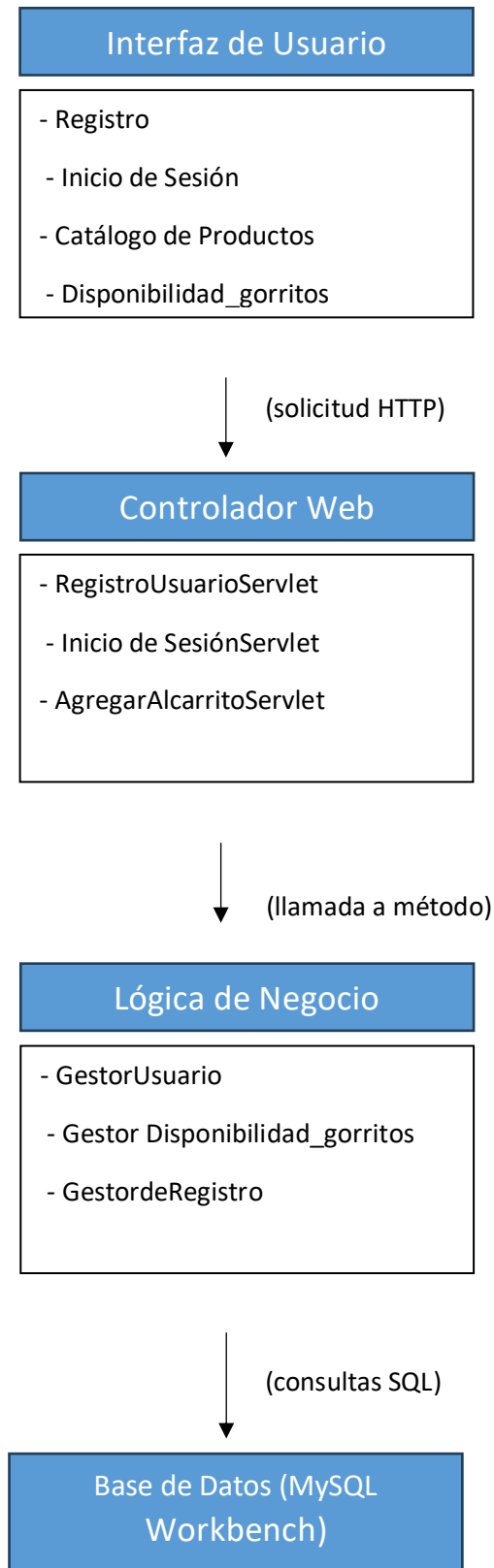
- loginServlet
- AgregarAlCarritoServlet
- RegistrameServlet

**Lógica de Negocio**

- Contiene las reglas de negocio y la lógica que define cómo se manejan los productos, usuarios, y pedidos.
- **Subcomponentes:**
  - GestorUsuario
  - GestorRegistrame
  - GestorAgregarAlCarrito

**Base de Datos (MySQL Workbench)**

- Almacena la información persistente del sistema, como los datos de los iniciar sesión usuarios, productos, y registros.
- **Subcomponentes:**
  - Iniciar sesión
  - Productos
  - Registro

**Representación del Diagrama de Componentes:**

- Iniciar sesión
  - Disponibilidad\_gorritos
  - Registro usuario

### **Los mecanismos de seguridad que requiere la aplicación.**

En la aplicación desarrollada para la venta de gorros **CAMIPIA**, se han implementado algunos mecanismos de seguridad para garantizar la protección de los datos del usuario y la integridad del sistema:

#### **1. Uso de API Seguras:**

- La aplicación utiliza API para gestionar la comunicación entre el frontend y el backend, asegurando que los datos se transfieran de manera segura y controlada.

#### **2. Restricciones en el Proceso de Inicio de Sesión:**

- **Validación de Correo Electrónico:** Se implementa una validación estricta del correo electrónico, asegurando que el mismo no se repita entre usuarios y que contenga el símbolo “@”, evitando así errores comunes y asegurando la unicidad del usuario.
- **Validación de Nombre:** La casilla de nombre no acepta números, garantizando que los nombres de usuario sean coherentes y libres de caracteres inválidos.
- **Redirección en Caso de Error:** Si un usuario ingresa datos incorrectos durante el inicio de sesión, se le impide el acceso y se lo redirige de vuelta al formulario de inicio, manteniendo la seguridad del acceso al sistema.

### 3. Control de Inventario en la Cesta de Compras:

- Al intentar agregar un producto al carrito, si la base de datos indica que la cantidad está en cero, la aplicación notifica al usuario que el producto está agotado, evitando así que se intente comprar un producto no disponible.

Identificar las capas en donde se ubican los componentes.

Estos son algunos de los mecanismos que se han implementado en el proyecto **CAMIPIA**, aseguran que solo los usuarios autorizados puedan acceder a sus cuentas, que los datos de usuario sean válidos y únicos

En una arquitectura de software típica para aplicaciones web y más que todo el proyecto que estoy presentando, los componentes se organizan en capas que separan las responsabilidades y facilitan el mantenimiento y la escalabilidad del sistema. Para la aplicación de venta de gorros CAMIPIA, podemos identificar las siguientes capas y ubicar en ellas los componentes previamente mencionados:

ORDEN DE CAPAS	COMPONENTES	DESCRIPCION
1. Capa de Presentación (Front-end)	Interfaz de Usuario (UI): Registro, inicio de sesión, catálogo de productos, carrito de compras.	Esta capa es responsable de la interacción con el usuario. Aquí se manejan todos los aspectos visuales y la experiencia del usuario. Incluye las páginas web, formularios, y cualquier otro elemento con el que los usuarios interactúan directamente.

2. Capa de Controlador (Controlador Web)	Controlador Web (Servlets): loginServlet, AgregarCarritoServlet, CarritoServlet, RegsitrarClienteServlet.	La capa de controlador actúa como intermediaria entre la capa de presentación y la capa de lógica de negocio. Recibe las solicitudes del usuario desde la interfaz de usuario y las procesa, invocando los servicios y componentes necesarios en la lógica de negocio para manejar la solicitud.
3. Capa de Lógica de Negocio (Back-end)	GestorLoginUsuario, GestorRegistro, GestorDisponibilidad_Producto	Esta capa contiene las reglas de negocio y la lógica central de la aplicación. Aquí se define cómo se gestionan los usuarios, productos, y pedidos, y se aplican las políticas de negocio y validaciones necesarias.
4. Capa de Persistencia (Base de Datos)	Base de Datos (MySQL Workbench): Usuarios, Registro, Agregar al carrito	La capa de persistencia se encarga de almacenar y recuperar los datos persistentes del sistema. Interactúa con la base de datos para guardar la información de usuarios, productos, y pedidos, y para recuperarla cuando sea necesario.

Conocer la metodología de desarrollo de *software*.

Características del Desarrollo de Software CAMIPIA:

Modularidad:

La página web se organiza en módulos como el registro de usuario, inicio de sesión, catálogo de productos, agregar al carrito. Cada módulo tiene una responsabilidad específica, lo que facilita la gestión y la posibilidad de hacer cambios sin afectar el resto del sistema.

#### Reusabilidad:

El código y los componentes, como la validación de formularios, el manejo de la base de datos MYSQL Workbench, y la lógica de negocio, se diseñan para ser reutilizables en diferentes partes de la aplicación. Por ejemplo, las validaciones de entrada de datos (como el formato del correo electrónico y el nombre) se pueden aplicar en varios formularios-

#### Mantenibilidad:

La página se desarrolla con un enfoque en la fácil actualización y mantenimiento. El uso de buenas prácticas como la separación en capas (presentación, lógica de negocio, y persistencia) ayuda a realizar mejoras o corregir errores de manera eficiente.

#### Escalabilidad:

La estructura del software permite agregar nuevas funcionalidades, como la integración con un sistema de pago en línea o la ampliación del catálogo de productos, sin necesidad de realizar grandes cambios en la estructura existente aún están en proceso de elaboración.

#### Seguridad:

La aplicación implementa mecanismos de seguridad importantes: uso de APIs para manejo de datos, validaciones estrictas para el registro de usuarios (como la unicidad del correo y la exigencia del símbolo @), y restricciones que evitan la compra de productos agotados. Además, la aplicación impide el acceso si se ingresan credenciales incorrectas.

### Metodología de Desarrollo de Software:

Para la página web de ventas de gorros CAMIPIA, se ha adoptado una metodología ágil y sencilla, que permite una rápida adaptación a los cambios en los requisitos del cliente o del mercado. La metodología ágil es ideal para proyectos como este, donde las necesidades pueden evolucionar a medida que se desarrolla el negocio en línea.

#### Características Clave:

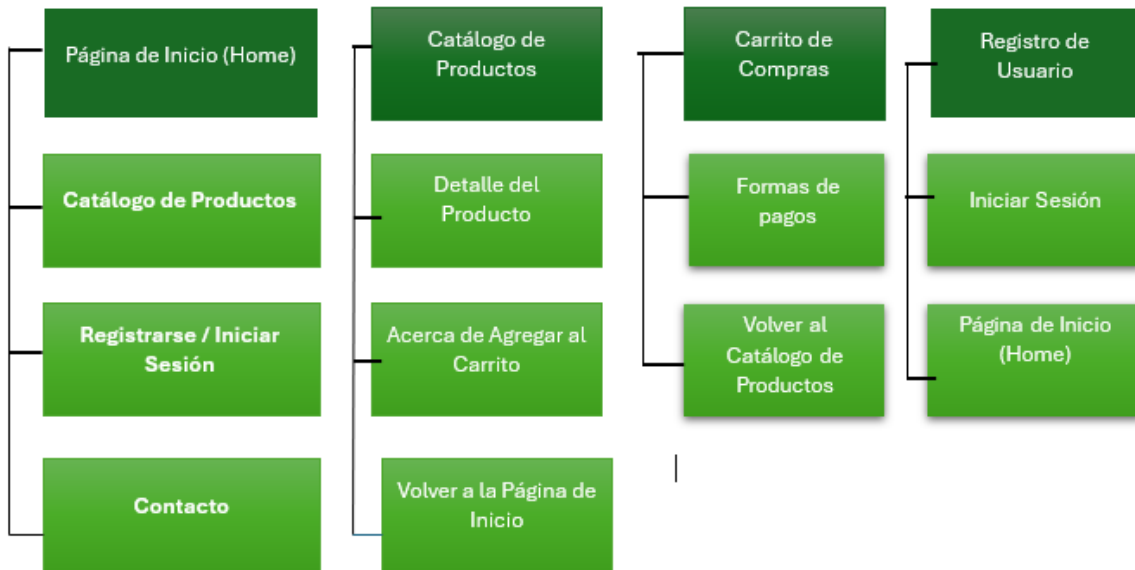
**Iteraciones Cortas:** Se trabaja en ciclos cortos (sprints), donde se desarrollan y prueban funcionalidades específicas, como el módulo de registro de usuario o la integración del carrito de compras.

**Retroalimentación Continua:** Se obtiene retroalimentación del cliente (en este caso, la persona que realiza los gorros) para asegurarse de que el producto final cumple con sus expectativas y necesidades.

**Adaptabilidad:** Si se detectan nuevas necesidades, como agregar un nuevo método de pago o cambiar el diseño de la interfaz, se pueden incorporar rápidamente en las siguientes iteraciones.



Conocer el mapa de navegación de la aplicación.



**Codificación de cada módulo en el lenguaje seleccionado.**

**Módulo de iniciar sesión:** para asegurar que el usuario ha proporcionado la información necesaria, se implementan condicionales que verifican si todos los campos requeridos (como "nombre de usuario" y "contraseña") están completos antes de permitir que el usuario proceda.

Si el campo de "nombre de usuario" o "contraseña" está vacío, el sistema muestra un mensaje diciendo que los campos son obligatorios y no permite continuar hasta que se completen.

Esta validación es crucial para asegurar que el proceso de inicio de sesión sea seguro y funcional, ya que evita que el sistema intente procesar un inicio de sesión con información incompleta.

**Iniciar Sesión**

**Ingresar**

### Código en Apache netbeans y el servlet

```

34         pst.setString(1, usuario);
35         pst.setString(2, contraseña);
36
37         ResultSet rs = pst.executeQuery();
38         if (rs.next()) {
39
40             // Si existe el usuario,
41
42             response.sendRedirect("panel.jsp");
43
44
45             // Si no existe, redirige a login.jsp con mensaje de error
46
47         } else {
48
49             conexion con = new conexion();
50             Connection cn = con.getConnection();
51
52             // la consulta para verificar usuario y contraseña
53
54             try {
55
56                 String sql = "SELECT * FROM usuarios WHERE usuario = ? AND contraseña = ?";
57                 PreparedStatement pst = cn.prepareStatement(sql);

```

```

49         response.sendRedirect("loginfallido.jsp?error=Usuario o contraseña incorrectos");
50     }
51
52     } catch (SQLException e) {
53         e.printStackTrace();
54         response.sendRedirect("loginfallido.jsp?error=Error de conexión");
55     }
56 }
57
58 @Override
59 protected void doGet(HttpServletRequest request, HttpServletResponse response)
60     throws ServletException, IOException {
61     response.sendRedirect("login.jsp");
62 }
63 }

```

com.mycompany.login.servlet.LoginServlet > doPost > con >

## El módulo de registro

Es la primera interacción que un usuario tiene con una aplicación que requiere autenticación. Su propósito es permitir que nuevos usuarios creen una cuenta proporcionando la información necesaria.

Durante el registro, se solicitan datos esenciales como:

Nombre de usuario,

correo electrónico: Que servirá como identificador único.

Contraseña: Que el usuario utilizará para acceder a su cuenta de forma segura.

Este quedara archivado en la base de datos que se esta trabajando MySQL Workbench.

La primera Capa de Presentación (Front-end) que el cliente ve al registrarse en la página de CAMIPIA y al hacer el click sale otra ventana en donde se le solicitan al cliente o usuario Campos como el nombre, correo y contraseña.

The diagram illustrates the registration process flow. It starts with a welcome screen titled "Bienvenido a Tienda de Gorritos CAMIPIA" with the subtitle "Crea tu cuenta y disfruta de nuestros productos." and a green "Registrate" button. An arrow points from this button to a registration form titled "Regístrate". The form contains three input fields labeled "Nombre:", "Correo:", and "Contraseña:", each followed by a light blue input box. At the bottom of the form is a light blue "Registrar" button.

**Bienvenido a Tienda de Gorritos CAMIPIA**

Crea tu cuenta y disfruta de nuestros productos.

**Registrate**

**Regístrate**

Nombre:

Correo:

Contraseña:

**Registrar**

Este módulo le indica al usuario debe diligenciar todos los campos sino no podrá seguir en le proceso.



**Regístrate**

**Nombre:**

**Correo:**

**Contraseña:**

! Rellena este campo.

Registrar

Adicional también indica que si no usan el símbolo @ en el campo del correo no dejara continuar con el proceso, saldrá un mensaje como que se muestra a continuación:



**Regístrate**

**Nombre:**

**Correo:**

! Incluye "@" en la dirección de correo electrónico. En "anagmail.com" falta un símbolo "@".

\*\*\*\*\*

Registrar



Cuando se  
registra

correctamente el usuario quedara registrado en la base de datos de MySQL Workbench la cual fue la base datos que se eligió para este proyecto y te dirige a la pagina principal

A screenshot of a registration form titled 'Regístrate' (Sign up). The form is enclosed in a light blue border. It contains three input fields: 'Nombre:' (Name) with the value 'Amanda', 'Correo:' (Email) with the value 'ama@hotmail.com', and 'Contraseña:' (Password) with four dots. Below the fields is a blue button labeled 'Registrar' (Register).

1 • `SELECT * FROM tienda_gorritos.clientes;`

	id	nombre	email	password
1	dani	oscar@sanchez	1234	
2	dani	oscar@sanchez	1234	
3	Aminta Castro	sanchezdan@gmail...	1234	
4	dani	sanchez86@gmail....	1234	
5	amin	amin@gmail	5678	
23	Amanda	ama@hotmail.com	amnada	
*	NULL	NULL	NULL	NULL

Código en Apache netbeans y el servlet del formulario

```

16  @WebServlet(name = "RegistrarClienteServlet", urlPatterns = {"/RegistrarClienteServlet"})
17  public class RegistrarClienteServlet extends HttpServlet {
18
19      @Override
20      protected void doGet(HttpServletRequest request, HttpServletResponse response)
21          throws ServletException, IOException {
22          response.sendRedirect("registro.jsp");
23      }
24
25      @Override
26      protected void doPost(HttpServletRequest request, HttpServletResponse response)
27          throws ServletException, IOException {
28          try {
29              processRequest(request, response);
30          } catch (SQLException ex) {
31              Logger.getLogger(RegistrarClienteServlet.class.getName()).log(Level.SEVERE, null, ex);
32              response.sendRedirect("registroFallido.jsp");
33          }
34      }
35
36      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
37          throws ServletException, IOException, SQLException {
38
39          String nombre = request.getParameter("nombre");
40          String email = request.getParameter("email");
41          String password = request.getParameter("password");
42
43          // Validación de campos obligatorios
44          if (nombre == null || nombre.isEmpty() || email == null || email.isEmpty() || password == null || p
45              response.sendRedirect("registro.jsp?error=Todos los campos son obligatorios");
46          return;
47      }
48
49
50      // Validación del nombre (solo letras)
51
52
53      if (!nombre.matches("[a-zA-Z]+")) {
54          response.sendRedirect("registro.jsp?error=El nombre solo debe contener letras");
55          return;
56      }
57
58
59      // Validación del formato del email
60
61      if (!email.matches("^([\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$)")) {
62          response.sendRedirect("registro.jsp?error=Formato de email incorrecto");
63          return;
64      }
65
66      // Conexión a la base de datos
67
68
69      conexion con = new conexion();
70      try {Connection cn = con.getConnection()} {

```

```

71 String sql = "INSERT INTO clientes (nombre, email, password) VALUES ('"+nombre+"','"+email+"','"+
72 try (PreparedStatement pst = cn.prepareStatement(sql)) {
73     pst.setString(1, nombre);
74     pst.setString(2, email);
75     pst.setString(3, password);
76
77     int rowCount = pst.executeUpdate();
78
79     if (rowCount > 0) {
80         // Registro exitoso
81         response.sendRedirect("registroFallido.jsp");
82     } else {
83         // Error en el registro
84         response.sendRedirect("registroFallido.jsp");
85     }
86 }
87 } catch (SQLException ex) {
88     Logger.getLogger(RegistrarClienteServlet.class.getName()).log(Level.SEVERE, null, ex);
89     response.sendRedirect("registroFallido.jsp");
90 }
91 }
92 }

```


**El módulo de "Agregar al carrito":** permite a los usuarios seleccionar productos que desean comprar. Si el producto está disponible o la cantidad disponible en la base de datos es mayor de uno te deja seguir comprando, como lo muestra las siguientes imágenes:

 Importar favoritos
 Para acceder rápidamente, coloca los favoritos en la barra de favoritos. ¿Buscas tus favoritos? [Comprobar los perfiles](#)

## Bienvenido a la tienda de gorritos

ID del Producto:

Muestra de la base de datos con algunos productos

Result Grid			
		Filter Rows:	<input type="text"/>
		Edit:	
	Id	nombre	cantidad
▶	1	gorrox	5
	2	hilos	5
	3	gorra	0
*	NULL	NULL	NULL

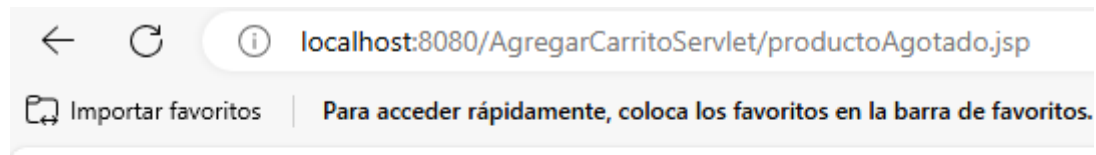


Si se digita en el ID del producto y cantidad es 0 le indica que

## Bienvenido a la tienda de gorritos

ID del Producto:

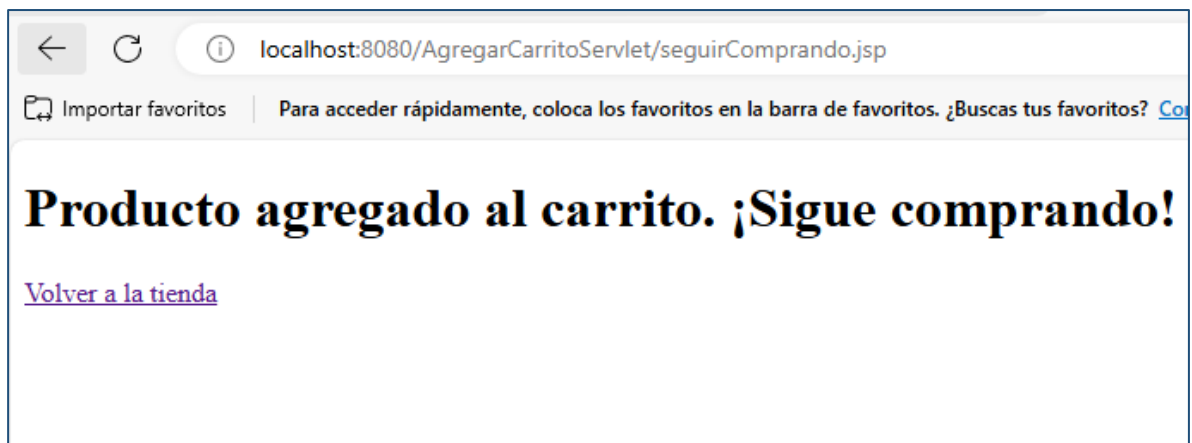
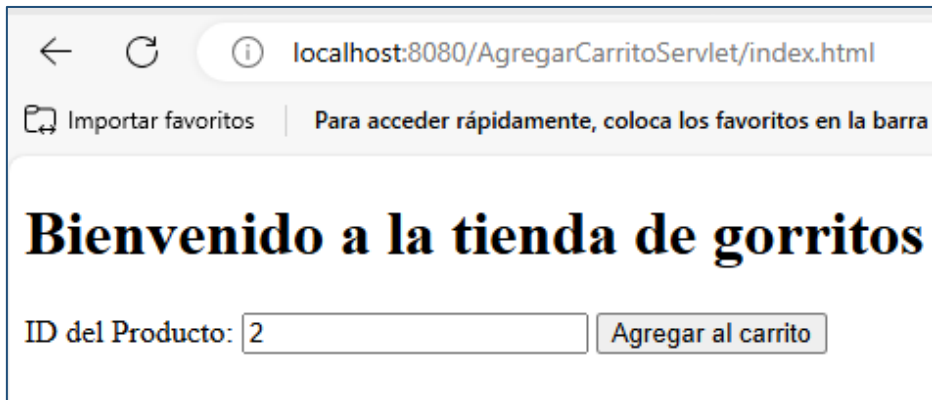
Pasa al API, con el mensaje que se muestra a continuación.



## Lo sentimos, el producto está agotado.

[Volver a la tienda](#)

El otro ejemplo se muestra con cantidades disponibles, basados en la imagen de la base de datos antes mencionada.



Código en Apache netbeans y el servlet del Agregar al carrito

```

19  @WebServlet(name = "AgregarCarritoServlet", urlPatterns = {"/agregarCarrito"})
20  public class AgregarCarritoServlet extends HttpServlet {
21
22      @Override
23      protected void doPost(HttpServletRequest request, HttpServletResponse response)
24          throws ServletException, IOException {
25
26          String Id = request.getParameter("Id");
27          conexion con = new conexion();
28          Connection cn = con.getConnection();
29
30          try {
31              // Consulta SQL para verificar la cantidad del producto
32              String sql = "SELECT cantidad FROM productos WHERE Id = ?";
33              PreparedStatement pst = cn.prepareStatement(sql);
34              pst.setString(1, Id);
35              ResultSet rs = pst.executeQuery();
36
37              if (rs.next()) {
38                  int cantidad = rs.getInt("cantidad");
39
40                  if (cantidad > 0) {
41                      // Redirige a la página para seguir comprando
42                      response.sendRedirect("seguirComprando.jsp");
43                  } else {
44                      // Redirige a la página que indica que el producto está agotado
45                      response.sendRedirect("productoAgotado.jsp");
46                  }
47              } else {
48                  // Si no encuentra el producto, redirige a una página de error o manejo adecuado
49                  response.sendRedirect("productoAgotado.jsp");
50              }
51
52          } catch (SQLException e) {
53              e.printStackTrace();
54              response.sendRedirect("errorConexion.jsp");
55          } finally {
56              try {

```

```

57              if (cn != null) {
58                  cn.close();
59              }
60          } catch (SQLException e) {
61              e.printStackTrace();
62          }
63      }
64  }
65
66  @Override
67  protected void doGet(HttpServletRequest request, HttpServletResponse response)
68      throws ServletException, IOException {
69      response.sendRedirect("index.html");
70  }
71  }

```

Control de versiones (GIT)

Característica	Git Local	Git Remoto
Almacenamiento	Repositorio local en el disco duro del usuario	Repositorio remoto en un servidor o servicio web
Acceso	Acceso local únicamente	Acceso remoto a través de Internet
Colaboración	Menos adecuado para colaboración en equipo	Diseñado para facilitar la colaboración entre equipos
Backup	Mayor riesgo de pérdida de datos sin respaldo	Respaldo seguro y accesible en caso de pérdida de datos
Historial de cambios	Mantiene historial de cambios localmente	Historial de cambios compartido entre colaboradores
Sincronización de código	No es posible sincronizar con otros repositorios locales	Permite sincronizar con múltiples repositorios remotos
Funcionalidades adicionales	Limitado en funcionalidades de colaboración	Ofrece funcionalidades avanzadas como solicitudes de extracción, ramas remotas, etc.
Requiere conexión a Internet	No requiere conexión a Internet	Requiere conexión a Internet para sincronizar cambios

**Determinar los frameworks a en cada capa de la aplicación.**

Generalmente, las capas de una aplicación se dividen en:

1. **Capa de Presentación (Frontend)**
2. **Capa de Lógica de Negocio (Backend)**
3. **Capa de Persistencia (Base de Datos)**

### **Librerías necesarias en cada etapa**

#### **1. Capa de Presentación (Frontend)**

HTML/CSS/JavaScript:

HTML5: Para la estructura del contenido.

CSS3: Para el diseño y la presentación visual.

JavaScript: Para la interactividad en la página.

Frameworks y Librerías de JavaScript:

React.js / Vue.js / Angular: Para construir interfaces de usuario dinámicas.

Bootstrap / Tailwind CSS: Para un diseño responsive y predefinido.

Axios / Fetch API: Para hacer peticiones HTTP desde el frontend.

#### **2. Capa de Lógica de Negocio (Backend)**

JDBC: Para conectar Java con la base de datos.

Hibernate: Un ORM (Object-Relational Mapping) que simplifica la interacción con la base de datos.

### **3. Capa de Acceso a Datos (Persistencia)**

Base de Datos Relacional:

MySQL / PostgreSQL: Bases de datos comunes para aplicaciones web.

Librerías de JDBC (Java Database Connectivity): Para la conexión y ejecución de consultas SQL desde Java.

ORM (Object-Relational Mapping)

### **4. Capa de Seguridad**

Java:

JWT (JSON Web Token): Para la autenticación basada en tokens.

Maven/Gradle: Para la gestión de dependencias y la automatización del build.

## **Frameworks para cada etapa**

### **1. Capa de Presentación (Frontend)**

React.js: Popular por su facilidad para crear interfaces de usuario dinámicas y su gran ecosistema.

### **2. Capa de Lógica de Negocio (Backend)**

Spring MVC: Para estructurar la aplicación utilizando el patrón Modelo-Vista-Controlador.

Express.js (Node.js): Minimalista y rápido, ideal para construir APIs RESTful con Node.js.

### **3. Capa de Acceso a Datos (Persistencia)**

Hibernate (Java): Framework ORM (Object-Relational Mapping) que simplifica la interacción con bases de datos relacionales.

TypeORM (Node.js): Un ORM para Node.js que trabaja bien con bases de datos SQL y NoSQL.

### **4. Capa de Seguridad**

Spring Security (Java): Framework completo para manejar autenticación y autorización en aplicaciones Java.

JWT (JavaScript/Java/Python): Utilizado para la autenticación basada en tokens, es común en aplicaciones con APIs RESTful.

## Conclusiones

**1. Mejora de la eficiencia y cohesión del sistema:** Los módulos integrados permiten que diferentes funciones dentro de un software operen de manera más eficiente y coherente. Al trabajar juntos, estos módulos comparten datos y funcionalidades en tiempo real, lo que reduce duplicidades y mejora la integración de información en todo el sistema.

**2. Facilidad en el mantenimiento y actualización:** La arquitectura modular facilita la actualización y mantenimiento del software, ya que los módulos pueden ser modificados o reemplazados sin afectar el sistema en su totalidad. Esto reduce el tiempo y costo de mantenimiento, y permite adaptarse a las nuevas necesidades del usuario o del entorno tecnológico.

**3. Escalabilidad del software:** Al estar compuesto por módulos independientes, el software es fácilmente escalable. Nuevas funcionalidades pueden añadirse sin necesidad de una reestructuración completa del sistema. Esto resulta crucial para proyectos que requieren crecimiento progresivo o expansión a nuevas áreas.



### **Bibliografía**

1. Aprendiendo JavaScript: Diseño y Programación Web Ilya Kantor  
(2007.2014) Moderna <https://es.javascript.info/>
2. tuatara (17 de Agosto de 2023) <https://tuatara.co/blog/software/seguridad-en-desarrollo-web/>
3. Somosmpt, Sergio Ariel Guzman (07 de Abril de 2020)  
<https://somospnt.com/blog/158-securizando-mi-api-con-spring-security-y-oauth-2-0>