República Dominicana

Pontificia Universidad Católica Madre y Maestra

Facultad de ingenierías

Escuela de computación y telecomunicaciones

Practica 6: Ajax WebSockets - Tienda Productos

Presentado por:

Lenin E. Gutiérrez S.

Matrícula:

2019-0010

Materia:

ISC-415

Profesor:

Carlos Camacho

Santiago de los caballeros

Noviembre 2024

Introducción

El presente proyecto, fue desarrollado para proporcionar una plataforma interactiva en la que los usuarios pueden navegar y comprar productos, además de permitir a los administradores visualizar métricas en tiempo real sobre la actividad en la tienda. Este proyecto se basa en la tecnología AJAX y WebSockets, que aportan mejoras significativas en la experiencia de usuario mediante la actualización dinámica del contenido sin necesidad de recargar la página. Este enfoque es particularmente útil en aplicaciones de comercio electrónico, donde la interacción en tiempo real y la respuesta inmediata a las acciones del usuario son cruciales para una experiencia de compra fluida.

El objetivo principal del proyecto es permitir una interacción en tiempo real tanto para usuarios como para administradores. Los usuarios pueden ver la cantidad de personas conectadas en la plataforma, agregar comentarios en los productos y recibir notificaciones inmediatas cuando un comentario es eliminado. Para los administradores, el sistema proporciona un "Dashboard" que muestra datos de ventas y estadísticas de productos a través de gráficos en tiempo real, implementados con Google Charts, y que se actualizan automáticamente cada vez que ocurre una venta.

El proyecto utiliza Javalin como framework en el backend, complementado con tecnologías de frontend como HTML, JavaScript y jQuery, que facilitan la comunicación en tiempo real a través de WebSockets. Además, la persistencia de datos se maneja mediante JDBC y JPA, con Hibernate como proveedor, para asegurar una interacción eficiente y organizada con la base de datos.

Esta plataforma, al integrar AJAX y WebSockets, proporciona una experiencia enriquecida y moderna para el usuario final, además de herramientas efectivas de monitoreo y control para los administradores de la tienda.

Desarrollo del proyecto

El proyecto "Ajax WebSocket - Tienda de Productos" integra una serie de tecnologías y componentes que interactúan entre sí para crear una experiencia de usuario dinámica y en tiempo real. A continuación, se presenta un desglose de los aspectos técnicos y funcionalidades principales, junto con una descripción detallada de los archivos y códigos utilizados.

1. Integración de AJAX para Interactividad Dinámica

La tecnología AJAX (Asynchronous JavaScript and XML) permite que la aplicación web actualice contenido sin requerir recargas de página completas. En este proyecto, AJAX se emplea para diversas interacciones entre el cliente y el servidor, como la actualización de comentarios en los productos y la actualización en tiempo real del conteo de usuarios conectados.

Un ejemplo claro de esta implementación se puede observar en el archivo producto_detalle.html, donde los usuarios pueden ver la lista de comentarios asociados a cada producto y eliminar un comentario sin recargar la página. La siguiente porción de código maneja esta funcionalidad:

```
$("#boton").click(function() {
    var idComentario = $("#idComentario").val().toString();
    if (webSocket.readyState === WebSocket.OPEN) {
        webSocket.send(idComentario);
    } else {
        console.log("El WebSocket no está conectado.");
    }
});
```

Esta sección de código utiliza AJAX junto con WebSocket para enviar el ID del comentario al servidor cuando se hace clic en el botón "Eliminar". La conexión en tiempo real permite que este

cambio se refleje instantáneamente en los navegadores de otros usuarios que están viendo el mismo producto.

2. Uso de WebSockets para Comunicación en Tiempo Real

La implementación de WebSockets es un pilar en este proyecto, ya que permite mantener una comunicación bidireccional y en tiempo real entre el cliente y el servidor. En este caso, WebSockets facilita la actualización instantánea del conteo de usuarios conectados y la eliminación de comentarios de productos para todos los usuarios en tiempo real.

Por ejemplo, en el archivo userCount.html, el conteo de usuarios conectados se actualiza automáticamente cada vez que un usuario entra o sale de la tienda. El siguiente fragmento muestra cómo se configura la conexión de WebSocket para recibir datos en tiempo real:

```
const socket = new WebSocket('ws://localhost:7000/user-count');

socket.onmessage = function(event) {

    const data = JSON.parse(event.data);

    if (data.type === 'userCount') {

        userCountDiv.textContent = `Usuarios conectados:
${data.count}`;

    }

};
```

Este código crea una conexión WebSocket al servidor y recibe mensajes JSON que contienen el número actual de usuarios conectados, actualizando el contenido de la interfaz de usuario de manera continua y sin intervención del usuario.

3. Dashboard para Administradores con Google Charts

El proyecto incluye un "Dashboard" o tablero de control diseñado exclusivamente para los administradores, donde se muestran las métricas de ventas y productos en gráficos generados en tiempo real. Este componente, implementado en dashboard.html, permite a los administradores ver de inmediato el impacto de las ventas y ajustar sus estrategias en consecuencia.

El código JavaScript de este archivo define la carga de gráficos de pastel y de barras utilizando Google Charts. A continuación, se presenta un extracto del código para generar el gráfico de pastel que muestra la cantidad de productos vendidos:

```
function drawProductosPieChart(datos) {
    const data = new google.visualization.DataTable();
    data.addColumn('string', 'Producto');
    data.addColumn('number', 'Cantidad');
    datos.forEach(item => {
        data.addRow([item.nombre, item.cantidad]);
    });
    const options = { title: 'Cantidad de Productos Vendidos', pieHole: 0.4 };
    const chart = new google.visualization.PieChart(document.getElementById('productos-pie-chart'));
    chart.draw(data, options);
}
```

Este fragmento configura el gráfico de pastel utilizando los datos de productos vendidos y actualiza el gráfico cada vez que el servidor envía nuevos datos a través de WebSocket. Además, el tablero incluye un histograma de ventas a lo largo del tiempo, lo cual proporciona una visión detallada del rendimiento de la tienda.

4. Implementación del Backend con Javalin y Persistencia de Datos

En el backend, el proyecto utiliza el framework Javalin para gestionar las solicitudes HTTP y los endpoints necesarios para las interacciones del frontend. Javalin facilita la creación de controladores RESTful, permitiendo una integración eficiente con el frontend de la aplicación.

La persistencia de datos se maneja a través de JDBC y JPA, utilizando Hibernate como proveedor. El archivo persistence.xml configura la conexión con la base de datos H2, la cual almacena la información sobre productos, usuarios, y transacciones. A continuación, se presenta una porción de la configuración de este archivo:

```
<persistence-unit name="MiUnidadPersistencia" transaction-type="RESOURCE_LOCAL">

<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <class>clases.Carrito</class>
    <class>clases.CarritoItem</class>
    <class>clases.Producto</class>
    <class>clases.Usuario</class>
    <class>clases.Comentario</class>
    <properties>
        <property name="jakarta.persistence.jdbc.url" value="jdbc:h2:tcp://localhost/~/mibasededatos" />
        <property name="jakarta.persistence.jdbc.driver" value="org.h2.Driver" />
        <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>
```

```
        <property                    name="jakarta.persistence.schema-
generation.database.action" value="drop-and-create"/>
    </properties>
</persistence-unit>
```

Este archivo define las clases de entidad, como Producto, Usuario y Comentario, que representan las tablas en la base de datos. Al almacenar y gestionar estos datos, el proyecto garantiza una experiencia de usuario coherente y accesible.

5. Controladores y Servicios para la Gestión de Funcionalidades

El backend cuenta con varios controladores y servicios que administran las funcionalidades del sistema. Algunos de los archivos clave incluyen ControladorCarrito.java, ControladorCRUD.java y ComentarioService.java, los cuales manejan la lógica detrás del carrito de compras, el CRUD (Crear, Leer, Actualizar, Eliminar) de productos y la gestión de comentarios, respectivamente.

Por ejemplo, el controlador ComentarioService.java permite a los usuarios agregar y eliminar comentarios en productos específicos. A través de WebSocket, este servicio también permite que todos los usuarios conectados vean inmediatamente cuando un comentario es eliminado.

6. Front-End de la Tienda

El frontend de la tienda está diseñado para ser visualmente atractivo e intuitivo. En index.html, la página principal presenta una estructura moderna con secciones de promoción de productos y un sistema de búsqueda. También cuenta con una barra de navegación y un sistema de carrusel para mostrar los diferentes productos, resaltando las categorías de frutas y vegetales.

El código en producto_detalle.html maneja las páginas de detalle de cada producto, donde los usuarios pueden ver imágenes, descripciones y comentarios, así como agregar nuevos comentarios mediante un formulario. Esta página también está configurada para conectarse a WebSocket y reflejar en tiempo real los cambios en los comentarios, eliminando automáticamente aquellos que han sido borrados por otros usuarios.

7. Manejo del Carrito de Compras

El carrito de compras es una característica esencial en este proyecto de tienda, permitiendo a los usuarios agregar productos que desean adquirir y gestionar su compra. La clase Carrito.java y CarritoService.java son las encargadas de manejar las operaciones relacionadas con el carrito. CarritoService.java implementa la lógica para agregar y quitar productos del carrito, actualizar cantidades, y calcular el total del carrito.

En el archivo ControladorCarrito.java, se definen los endpoints que permiten la interacción del frontend con el carrito. Un ejemplo de código en ControladorCarrito.java muestra cómo se gestiona la adición de un producto:

```
post("/carrito/agregar", ctx -> {
    String idProducto = ctx.formParam("idProducto");
    int cantidad = Integer.parseInt(ctx.formParam("cantidad"));
    carritoService.agregarProducto(idProducto, cantidad);
    ctx.redirect("/carrito");
});
```

Este código recibe los datos del producto y la cantidad desde el formulario en el frontend y los pasa al servicio del carrito para que el producto se agregue al carrito actual.

8. Gestión de Usuarios y Sesiones

La gestión de usuarios es otro componente clave del proyecto, donde UsuarioService.java y ControladorLogin.java desempeñan roles importantes. UsuarioService.java maneja las operaciones de autenticación y verificación de usuarios, incluyendo el inicio y cierre de sesión, mientras que ControladorLogin.java facilita el control de las rutas de autenticación.

En ControladorLogin.java, el método de inicio de sesión verifica las credenciales del usuario, estableciendo la sesión del usuario autenticado y redirigiéndolo a la tienda. A continuación, se muestra un fragmento del código que se encarga de esta autenticación:

```
post("/login", ctx -> {
    String username = ctx.formParam("username");
    String password = ctx.formParam("password");
    Usuario usuario = usuarioService.autenticar(username, password);
    if (usuario != null) {
        ctx.sessionAttribute("usuario", usuario);
        ctx.redirect("/tienda");
    } else {
        ctx.redirect("/login?error");
    }
});
```

Este método recibe el nombre de usuario y la contraseña desde el formulario de inicio de sesión en el frontend. Si las credenciales son correctas, el usuario se redirige a la página de la tienda, y se establece un atributo de sesión para mantener el estado autenticado del usuario.

9. Gestión de Productos y CRUD

La gestión de productos en la tienda incluye las operaciones CRUD, que son fundamentales para que los administradores puedan actualizar el catálogo. Las operaciones CRUD están gestionadas principalmente por ProductoService.java y ControladorCRUD.java.

ProductoService.java maneja la lógica de negocio de los productos, mientras que ControladorCRUD.java define los endpoints para realizar operaciones como crear, leer, actualizar y eliminar productos. A continuación, un ejemplo de cómo se gestiona la creación de un nuevo producto:

```
post("/productos/crear", ctx -> {
    Producto nuevoProducto = new Producto();
    nuevoProducto.setNombre(ctx.formParam("nombre"));

nuevoProducto.setPrecio(Double.parseDouble(ctx.formParam("precio")));
    nuevoProducto.setDescripcion(ctx.formParam("descripcion"));
    productoService.crearProducto(nuevoProducto);
    ctx.redirect("/productos");
});
```

Este código toma los parámetros del formulario en el frontend y crea un nuevo producto en la base de datos, permitiendo a los administradores agregar nuevos artículos a la tienda en cualquier momento.

10. Implementación de Seguridad y Manejo de Permisos

La seguridad es un aspecto crucial en este proyecto, especialmente para controlar el acceso a las funcionalidades administrativas. El acceso al "Dashboard" y las funciones de administración de productos está restringido solo a usuarios con el rol de administrador. Este control de acceso se maneja en el frontend y el backend.

En el frontend, se utilizan condiciones para mostrar u ocultar elementos según el rol del usuario. Por ejemplo, en producto_detalle.html, solo los administradores pueden ver el botón para eliminar comentarios:

```
<div th:if="${rol}">
```

```
    <input type="hidden" id="idComentario" name="idComentario" th:value="${comentario.id}">
    <button type="submit" class="btn btn-danger btn-sm" id="boton">Eliminar</button>
</div>
```

11. Conexión a la Base de Datos y Configuración de Persistencia

Como se mencionó anteriormente, el proyecto utiliza JDBC y JPA con Hibernate para la persistencia de datos. La configuración de la base de datos se define en persistence.xml, conectando el sistema a una base de datos H2. Esto permite que todas las operaciones de datos se realicen de manera persistente y organizada, con la capacidad de manejar transacciones y consultas SQL complejas.

Las entidades mapeadas en persistence.xml incluyen Producto, Usuario, Comentario, Carrito, y CarritoItem. Estas clases representan las tablas de la base de datos y contienen las propiedades y métodos necesarios para reflejar las estructuras de datos de la tienda.

12. Comprobaciones de Validación y Manejo de Errores

Finalmente, el proyecto también incluye comprobaciones para asegurar la integridad de las interacciones del usuario y manejar errores. Tanto en el frontend como en el backend se implementan validaciones que aseguran, por ejemplo, que los usuarios no puedan agregar productos sin haber iniciado sesión, o que los formularios estén completos antes de enviarse.

En ControladorCarrito.java, se verifican los parámetros enviados por el usuario para evitar errores al agregar productos al carrito. Además, se manejan errores de conexión en el WebSocket para reconectarse en caso de una interrupción.

```
function verificarConexion(){
    if(!webSocket || webSocket.readyState == 3){
```

```
        conectar();
    }
}
setInterval(verificarConexion, tiempoReconectar);
```

Este fragmento de código en producto_detalle.html establece una función para verificar y reconectar el WebSocket automáticamente si la conexión se pierde, manteniendo así la comunicación en tiempo real.

Conclusión

El proyecto "Ajax WebSocket - Tienda de Productos" logra una integración eficiente de diversas tecnologías para ofrecer una plataforma de comercio electrónico dinámica y orientada a la experiencia del usuario. Mediante el uso de AJAX y WebSockets, la aplicación permite una interacción en tiempo real tanto para los usuarios como para los administradores, proporcionando funciones como la actualización instantánea de comentarios, la visualización en tiempo real de usuarios conectados y un tablero con métricas de ventas actualizado al instante. La implementación de Javalin en el backend, junto con Hibernate para la persistencia de datos, asegura una gestión de datos sólida y una comunicación fluida entre el cliente y el servidor.

La estructura del proyecto se divide en componentes específicos: el manejo del carrito de compras, la gestión de usuarios y permisos, el sistema CRUD para productos, y las funcionalidades de notificaciones en tiempo real, que se ven complementadas por una interfaz de usuario intuitiva y estéticamente atractiva. Cada uno de estos componentes se conecta y sincroniza a través de WebSockets y AJAX, lo que permite mantener una experiencia de usuario fluida, sin recargas innecesarias, y que responde a los cambios de estado en el sistema de manera eficiente.

Desde una perspectiva de reflexión, el uso de AJAX y WebSockets en este proyecto demuestra cómo estas tecnologías pueden mejorar significativamente la interactividad en aplicaciones web. AJAX, al permitir solicitudes asíncronas al servidor, minimiza los tiempos de espera y mejora la velocidad de respuesta, mientras que WebSockets aporta una ventaja al mantener una conexión continua, ideal para funciones que requieren actualizaciones en tiempo real, como el conteo de usuarios conectados y la gestión de comentarios en productos. Para maximizar el aprovechamiento de estas tecnologías en futuros proyectos, es recomendable:

Utilizar AJAX en Interacciones Puntuales: AJAX es idóneo para operaciones puntuales, como cargar detalles específicos de un producto o actualizar pequeñas secciones de la interfaz sin recargar toda la página. Su implementación eficiente reduce la carga en el servidor y mejora la experiencia del usuario.

Aprovechar WebSockets para Actualizaciones en Tiempo Real: Los WebSockets son esenciales cuando se requieren notificaciones instantáneas, como las vistas en el dashboard de ventas y el

sistema de comentarios de este proyecto. Integrarlos adecuadamente garantiza que los cambios sean reflejados de inmediato, mejorando la fluidez y la transparencia en la interacción del usuario.

Estructurar el Backend para Soportar Alta Concurrencia: A medida que la aplicación escale, es crucial que el backend, en este caso implementado con Javalin, esté optimizado para manejar múltiples conexiones simultáneas. Esto incluye configuraciones de WebSocket y optimización de consultas a la base de datos.

Garantizar la Seguridad en la Comunicación y el Acceso: Es recomendable aplicar controles de seguridad más robustos, especialmente en el acceso a funciones administrativas y en el manejo de datos sensibles. La implementación de autenticación y autorización bien definidas asegura que solo los usuarios adecuados puedan realizar ciertas operaciones, como la eliminación de comentarios o el acceso al dashboard.