

# Agregar Cliente con PrimeFaces y Spring Boot



## Agregar un Nuevo Cliente en la Aplicación



En esta guía, implementaremos la funcionalidad para **agregar un nuevo cliente** en nuestra aplicación. Agregaremos un botón en el menú para abrir un formulario modal donde se capturarán los datos del nuevo cliente.

### 1 Conceptos Clave



Antes de empezar, repasemos algunos conceptos importantes que usaremos en esta implementación:

- **p:menuItem**: Es un elemento de PrimeFaces que permite agregar opciones dentro de un menú.
- **actionListener**: Atributo en JSF que permite ejecutar un método del controlador cuando se presiona un botón.
- **update**: Permite actualizar parcialmente la vista sin recargar toda la página.
- **PF('ventanaModalCliente').show()**: Método de PrimeFaces que se usa para mostrar un diálogo modal.
- **clienteSeleccionado**: Atributo que se usará para almacenar los datos del cliente a agregar.

## 2 Agregar el Botón "Nuevo Cliente" en el Menú

El primer paso es agregar un nuevo botón en nuestro menú de navegación. Este botón permitirá abrir el formulario para agregar un nuevo cliente.

### Código:

```
<p:menuitem value="Nuevo Cliente" icon="pi pi-fw pi-plus"
    actionListener="#{indexControlador.agregarCliente}"
    update=":forma-modal:cliente-ventana"
    oncomplete="PF('ventanaModalCliente').show()"/>
```

### Explicación:

1. `value="Nuevo Cliente"` → Define el texto del botón en el menú.
2. `icon="pi pi-fw pi-plus"` → Agrega un ícono de "más" (+) al botón.
3. `actionListener="#{indexControlador.agregarCliente}"` → Llama al método `agregarCliente()` en el controlador cuando se presiona el botón.
4. `update=":forma-modal:cliente-ventana"` → Actualiza la ventana modal que contendrá el formulario.
5. `oncomplete="PF('ventanaModalCliente').show()"` → Muestra el formulario modal después de ejecutar el `actionListener`. Este formulario se creará en la próxima lección.

## 3 Definir el Método y Atributo en el Controlador

Ahora agregaremos el método `agregarCliente()` en el controlador para inicializar un nuevo objeto `Cliente` y declararemos el atributo `clienteSeleccionado`.

### Código:

```
public class IndexControlador {

    private Cliente clienteSeleccionado;

    public void agregarCliente() {
        this.clienteSeleccionado = new Cliente();
    }
}
```

### Explicación:

- **private Cliente clienteSeleccionado;** → Declara el atributo para almacenar los datos del nuevo cliente.
  - **@Data** → Genera automáticamente los métodos *getter* y *setter* para `clienteSeleccionado` y `clientes`.
  - **agregarCliente()** → Crea un nuevo objeto `Cliente`, que será utilizado en el formulario.
- 

## 4 Código Final

Aquí está el código completo después de los cambios:

 `index.xhtml`

```
<!DOCTYPE html>
<h:html xmlns:h="http://xmlns.jcp.org/jsf/html"
         xmlns:f="http://xmlns.jcp.org/jsf/core"
         xmlns:p="http://primefaces.org/ui">
    <h:head>
        <title>Zona Fit (GYM)</title>
        <link rel="stylesheet" href="https://unpkg.com/primeflex@latest/primeflex.css"/>
        <link rel="stylesheet" href="https://unpkg.com/primeflex@latest/themes/primeone-dark.css"/>
    </h:head>
    <h:body>
        <div class="card">
            <h:form id="forma-clientes">
                <p:growl id="mensajes" showDetails="true"/>
                <!-- Menubar -->
                <div class="card">
                    <p:menubar>
                        <p:menuItem value="Inicio" icon="pi pi-fw pi-user"
                                   update=":forma-clientes:clientes-tabla"
                                   actionListener="#{indexControlador.cargarDatos}"/>
                        <p:menuItem value="Nuevo Cliente" icon="pi pi-fw pi-plus"
                                   actionListener="#{indexControlador.agregarCliente}"
                                   update=":forma-modal:cliente-ventana"
                                   oncomplete="PF('ventanaModalCliente').show()"/>
                    </p:menubar>
                </div>
                <!-- DataTable -->
                <div class="card">
                    <p:dataTable value="#{indexControlador.clientes}" var="cliente"
                               id="clientes-tabla" widgetVar="clientesTabla">
                        <f:facet name="header">
```

```
<div class="flex justify-content-center flex-wrap
    card-container yellow-container">
    <div class="flex align-items-center
    justify-content-center
    w-20rem h-4rem bg-yellow-500 font-bold
    text-gray-900 border-round m-2">
        <h3>Zona Fit (GYM)</h3>
    </div>
</div>
</f:facet>

<p:column headerText="Id">
    <h:outputText value="#{cliente.id}" />
</p:column>

<p:column headerText="Nombre">
    <h:outputText value="#{cliente.nombre}" />
</p:column>

<p:column headerText="Apellido">
    <h:outputText value="#{cliente.apellido}" />
</p:column>

<p:column headerText="Membresía">
    <h:outputText value="#{cliente.membresia}" />
</p:column>

</p:dataTable>
</div>
</h:form>
</div>
</h:body>
</h:html>
```

#### IndexControlador.java

```
package gm.zona_fit.controlador;

import gm.zona_fit.modelo.Cliente;
import gm.zona_fit.servicio.IClienteServicio;
import jakarta.annotation.PostConstruct;
import jakarta.faces.view.ViewScoped;
import lombok.Data;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Component;
import java.util.List;
import org.slf4j.Logger;

@Component
@Data
@ViewScoped
public class IndexControlador {

    @Autowired
   IClienteServicio clienteServicio;
    private List<Cliente> clientes;
    private Cliente clienteSeleccionado;
    private static final Logger logger = LoggerFactory.getLogger(IndexControlador.class);

    @PostConstruct
    public void init(){
        cargarDatos();
    }

    public void cargarDatos(){
        this.clientes = this.clienteServicio.listarClientes();
        this.clientes.forEach(cliente -> logger.info(cliente.toString()));
    }

    public void agregarCliente(){
        this.clienteSeleccionado = new Cliente();
    }
}
```

---

## 5 Aún NO deben ejecutar el proyecto

Aún no podemos ejecutar el proyecto, ya que falta agregar la ventana modal para agregar un nuevo cliente. Esto lo haremos en la próxima lección.

---

## 6 Conclusión

En esta guía, hemos agregado la funcionalidad para **crear un nuevo cliente** mediante un botón en el menú. Ahora, cuando el usuario presiona "Nuevo Cliente", se ejecuta un método en el servidor para inicializar un nuevo objeto `Cliente`, que será utilizado en un formulario modal.

En la siguiente lección, completaremos este formulario y aprenderemos a guardar los datos en la base de datos. 

---

**Nota: Recuerda que tienes el archivo .zip en la sección de recursos de este video con todos los archivos del proyecto actualizado según la lección estudiada, el cual puedes utilizar para cualquier problema.**

Sigue adelante con tu aprendizaje  , ¡el esfuerzo vale la pena!

**¡Saludos! **

**Ing. Ubaldo Acosta**

Fundador de [GlobalMentoring.com.mx](https://www.globalmentoring.com.mx)