Explicación del Flujo MVC (Modelo–Vista–Controlador)

Aplicación: WebProfilePage – Java Servlets + JSP + MongoDB

La aplicación implementa el patrón **MVC (Modelo–Vista–Controlador)** para separar responsabilidades, mejorar la mantenibilidad del código y permitir una evolución sencilla del proyecto. A continuación se describe cómo fluye la información dentro de cada capa.

1. Modelo (M – Model)

El **Modelo** encapsula las estructuras de datos que representan la información principal del sistema.

En este proyecto, el modelo está compuesto por:

Profile

Contiene los datos del perfil del usuario:

- nombre.
- bio.
- experiencia.
- contacto.
- foto (ruta).
- color del banner.

Skill

Representa una habilidad técnica:

- id (ObjectId generado por MongoDB).
- nombre.
- nivel (porcentaje).

Repositorios (Capa de acceso a datos)

Se implementan interfaces para abstraer la persistencia:

- **IProfileRepository / ProfileRepositoryMongo.**
- **ISkillRepository / SkillRepositoryMongo.**

Responsabilidades:

- Guardar y recuperar datos desde MongoDB.
- Mapear documentos BSON a Objetos Java.
- Proveer operaciones CRUD para profile y skills.

Esto permite que los controladores puedan trabajar sin conocer detalles de la base de datos.

2. Vista (V – View)

Las **vistas** son páginas JSP que presentan la información al usuario. El proyecto utiliza dos vistas principales:

profile.jsp

- Muestra los datos del perfil: nombre, bio, experiencia, contacto, foto.
- Muestra la lista de habilidades.
- No contiene formularios.
- Sirve únicamente como vista principal del usuario.
- Tiene un botón que dirige a edit.jsp.

edit.jsp

- Muestra las habilidades de programación en formato editable.
- Contiene un formulario para actualizar el perfil.
- Permite agregar nuevas habilidades.
- Permite modificar o eliminar habilidades existentes.

Las vistas no contienen lógica de negocio. Únicamente muestran datos enviados por los controladores.

3. Controlador (C – Controller)

La capa de control está formada por dos servlets:

- **ProfileController.**
- **SkillController.**

Cada uno recibe las peticiones del usuario, procesa la lógica correspondiente y decide qué vista mostrar.

ProfileController – Control del perfil

Flujo GET “/” o “/profile”

- Obtiene perfil y habilidades.
- Los envía a profile.jsp para mostrarlos.

Flujo POST "/profile"

Procesa el formulario enviado desde **edit.jsp**:

1. Actualiza los datos personales:
    a. nombre.
    b. biografía.
    c. experiencia.
    d. contacto.
2. Procesa y valida el color del banner (**bannerColor**).
3. Procesa la foto de perfil:
    a. recibe el archivo.
    b. lo guarda en **/uploads.**
    c. lo redimensiona a 180×180 px.
    d. Guarda los cambios en MongoDB.
4. Se redirige a **profile.jsp**.

SkillController – Control de habilidades

Flujo GET "/edit"

Cuando el usuario accede a la ruta /edit, el controlador procesa la solicitud siguiendo estos pasos:

1. **Carga de información desde MongoDB:**
   El controlador obtiene el perfil almacenado y la lista completa de habilidades.
    - Esto se realiza mediante **profileRepo.getProfile()** y **skillRepo.getAllSkills()**.
    - Ambos objetos se agregan al **request** como atributos para que puedan ser utilizados por la vista.
2. **Detección de modo edición (opcional):**
    - Si la URL incluye los parámetros **action=edit** e **id**, el controlador interpreta que el usuario desea modificar una habilidad existente.
    - En ese caso, busca la habilidad por su identificador y, si la encuentra, la almacena en el atributo **skillToEdit**.
    - Esto permite que el formulario de habilidades en **edit.jsp** se rellene automáticamente con los datos de la habilidad seleccionada.
3. **Envío de datos a la vista:**
    - Finalmente, la solicitud se envía a edit.jsp, donde se muestra:
        - El formulario de edición del perfil.
        - El formulario para agregar o actualizar habilidades.
        - La tabla/listado de habilidades existentes.

Flujo POST – múltiples acciones

Cuando el usuario envía un formulario de habilidades desde **edit.jsp**, el controlador recibe una solicitud POST y decide qué operación realizar basado en el parámetro action.

1. Agregar habilidad (action=add)

El controlador crea una nueva habilidad cuando el formulario envía **action=add**. El proceso incluye:

- Validar que los campos obligatorios (name y level) estén completos.
- Asegurar que el nivel esté dentro del rango permitido (1–100).
- Verificar que no exista otra habilidad con el mismo nombre.
- Crear la nueva habilidad y guardarla en la base de datos.
- Finalmente, dirigir nuevamente a **/edit** para evitar reenvíos duplicados y actualizar la interfaz.
- Si ocurre algún error, el controlador devuelve la vista con un mensaje explicativo.

2. Actualizar habilidad (action=update)

Si el usuario edita una habilidad existente:

- Se recupera el identificador enviado por el formulario.
- Se valida que la habilidad exista, que los datos sean correctos y que no se produzca un duplicado si el nombre fue modificado.
- La habilidad es actualizada en MongoDB.
- Después, se redirige nuevamente a **/edit** para mostrar los cambios actualizados.
- Si hay problemas de validación, el controlador reenvía la vista indicando el error.

3. Eliminar habilidad (action=delete)

Cuando el formulario envía action=delete:

- El controlador obtiene el id de la habilidad seleccionada.
- Intenta eliminar la habilidad desde el repositorio.
- Redirige nuevamente a **/edit** para reflejar el estado actualizado de la lista.
- Si ocurre un error durante la eliminación, este se muestra en la vista.

4. Flujo completo de una petición (resumen general)

A nivel global, el flujo MVC de la aplicación funciona así:

1. El usuario envía una solicitud (GET o POST).
2. El controlador (Servlet) correspondiente procesa la petición.
3. El controlador consulta o modifica el modelo mediante los repositorios.
4. Los datos del modelo se inyectan en la vista usando request.setAttribute.

5. La vista JSP muestra los datos y devuelve el HTML al navegador.