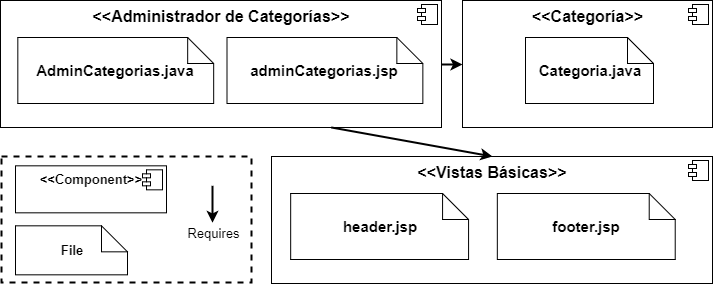
### Ejemplo base

Para trabajar las diferentes metodologías de implementación de componentes según los diferentes paradigmas de programación, utilizaremos el siguiente ejemplo.

**“Administración de categorías” y todas sus dependencias**.

Se basa en que, en una línea de tiendas virtuales, los productos deben pertenecer a categorías (“ropa hombre”, “electrodomesticos”, “televisores”, “etc”). Ahora, veamos el diagrama de componentes de esta “parte” de la línea de tiendas virtuales que presentamos en la Figura 1.



**Figura 1.** Parte del diagrama de componentes, correspondiente al componente

“Administrador de categorías” y sus dependencias.

### A continuación, describiremos el proceso de implementación de estos componentes, utilizando una arquitectura MVC, o en su defecto, una por capas que se acople al paradigma de programación FragOP.

### Programación orientada a fragmentos (FragOP)

FragOP es un nuevo paradigma utilizado para diseñar e implementar componentes de dominio de una línea de productos de software. FragOP es una mezcla entre enfoques compositivos y anotativos, y se basa en la definición de: (i) componentes de dominio, (ii) puntos de fragmentación, que son anotaciones sobre el código de los componentes de dominio; y (iii) fragmentos, un nuevo tipo de archivo que altera el código de los componentes de dominio.

**A) Implementación de componentes de dominio.** Los componentes están formados por diferentes tipos de archivos, tales como: HTML, CSS, JavaScript, Java, PHP, Python y JSP. Para este caso vamos a suponer que debemos crear la cabecera de una aplicación (header.php) su ID será BasicViewsHtml-Header, tal como se ve en el siguiente código. Este código está escrito en HTML y PHP, y representa el encabezado de una aplicación. Este código contiene un menú, que corresponde a una lista desordenada con un solo elemento (es decir, "Home") que está vinculado a la sección de inicio de la aplicación.

|  |
| --- |
| <!DOCTYPE html><html><head><title><?php echo($title); ?></title></head>  <body>    <ul class="nav navbar-nav">  <li><a href="<?php echo base\_url(); ?>index.php/Home/">Home</a></li>    </ul> |

**B) Puntos de fragmentación.** El archivo anterior podría refinarse con la adición de un **punto de fragmentación**. Un punto de fragmentación es una anotación (una marca muy simple) que especifica un "punto" en el que se puede modificar un archivo. Por ejemplo: otros componentes como Login o Productos pueden requerir la modificación del archivo header.php; específicamente para agregar nuevos elementos al menú anterior. El siguiente código muestra la estructura que debe seguir un punto de fragmentación.

|  |
| --- |
| LanguageCommentBlock<B|E>-<PointID>LanguageCommentBlock |

FragOP sugiere crear el punto de fragmentación comenzando con un bloque de comentarios (LanguageCommentBlock) basado en el tipo de lenguaje del archivo actual. Por ejemplo: para un archivo escrito en PHP, el punto de fragmentación debe comenzar con /\* y debe terminar con \*/. Para un archivo escrito en HTML, el punto de fragmentación debe comenzar con <!-- y debe terminar con -->. De esta forma, el código fuente de un archivo no se altera al agregar los puntos de fragmentación, lo que garantiza la coherencia del código y la capacidad de mantenimiento del código. Si un lenguaje de un archivo no proporciona un bloque de comentarios (como: archivos txt), entonces, sugerimos crear una expresión regular, como: [FragAnnot] [/FragAnnot].

Después de la sección de apertura de LanguageCommentBlock, el punto de fragmentación continúa con <B|E>-<PointID>. <B|E> corresponde a una sección de comienzo de punto de fragmentación (B) o sección final (E). En la primera aparición de un punto de fragmentación, se debe colocar la letra B. La sección final es opcional porque se utiliza para delimitar dónde termina un punto de fragmentación, que solo es necesario para acciones de *replace* and *hide* que describiremos más adelante. El punto de fragmentación continúa con un símbolo menos (-) y un ID de punto, que es un texto personalizado que se utiliza para identificar el punto de fragmentación. Finalmente, se debe agregar la sección de cierre de LanguageCommentBlock. El siguiente Código muestra un punto de fragmentación.

|  |
| --- |
| <!--B-menu-elements-modificator--> |

Basado en los conceptos anteriores, el archivo header.php se refina como se muestra en el siguiente código. Como se mencionó anteriormente, la adición del punto de fragmentación permitirá incluir nuevos elementos de menú en el futuro.

|  |
| --- |
| <!DOCTYPE html><html><head><title><?php echo($title); ?></title></head>  <body>   <ul class="nav navbar-nav">  <li><a href="<?php echo base\_url(); ?>index.php/Home/">Home</a></li>  **<!--B-menu-elements-modificator-->**   </ul> |

**C) Implementación de Fragmentos.** Un fragmento es un tipo especial de archivo que permite a los desarrolladores especificar alteraciones de código en los archivos de componentes. En general, un fragmento se debe definir tal cual se presenta en el siguiente código

|  |
| --- |
| Fragment <ID> {  Action: <add|replace|hide>  FragmentationPoint: <pointID>  PointBracketsLan: <language>  Destination: <fileID>  SourceCode: [ALTERCODE-FRAG]<code>[/ALTERCODE-FRAG]  } |

**Fragmento <ID>** ID sirve como un identificador para el fragmento.

**• Action:** <add | replace | hide> especifica el tipo de alteración.

* add permite inyectar una pieza de código sobre un PointID específico.
* replace permite reemplazar una pieza de código sobre un PointID específico.
* hide permite ocultar un fragmento de código sobre un PointID específico (el fragmento de código se coloca dentro de un bloque de comentario).

**• FragmentationPoint:** <PointID> PointID es un texto único que sirve para identificar un punto de fragmentación.

**• PointBracketsLan:** <language> language especifica el idioma en el que se definió el bloque de comentarios del punto de fragmentación. Por ejemplo: PHP, HTML y Java.

**• Destination:** <fileID> fileID representa el archivo de componente de dominio que se va a modificar.

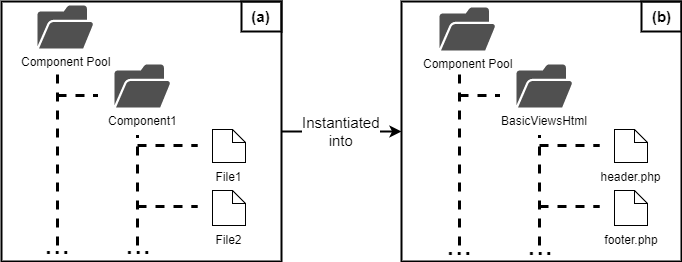
**• SourceCode:** <code> (Opcional) contiene el código que se inyectará.

Para una mejor comprensión sobre cómo funcionan los fragmentos, consideremos el siguiente ejemplo: El siguiente código muestra el código del fragmento ListProducts-AlterHeader, este código especifica que el archivo BasicViewsHtml-Header (Destination) será alterado en el menu-elements-modificador (FragmentationPoint) En este caso, el fragmento agregará (Action) un nuevo elemento de menú (SourceCode) dentro del archivo.

|  |
| --- |
| Fragment ListProducts-AlterHeader {  Action: add  FragmentationPoint: menu-elements-modificator  PointBracketsLan: html  Destination: BasicViewsHtml-Header  SourceCode: [ALTERCODE-FRAG]<li><a href="<?php echo base\_url(); ?>in-dex.php/Prod/">Products</a></li>[/ALTERCODE-FRAG]  } |

#### Proceso

En FragOP se debe crear un directorio donde se almacenen todos los componente y fragmentos. Se sugiere utilizar la estructura definida en la Figura 2.



**Figura 2.** Estructura del pool de componentes sugerida para FragOP.

Finalmente, en este caso y para tener un panorama más detallado de estos tipos de aplicaciones, implementaremos a continuación todos los componentes de la Figura 1.

**Ejemplo:**

#### Implementación de componentes en FragOP

**Herramientas.** Para implementar componentes en FragOP requerimos:

* **Eclipse:** entorno de desarrollo de software Web.
* **El componente *JSTL Library*** de Java EE; este componente extiende la funcionalidad de las JavaServer Pages (JSP) permitiendo utilizar unas etiquetas específicas para el desarrollo de páginas web dinámicas.
* Un servidor web escrito en Java que soporte el uso de servlets y JSPs, por ejemplo **Apache Tomcat.** El servidor web debe brindar soporte para aplicaciones JEE, y será usado dentro del proyecto para desplegar los componentes desarrollados.
* **VariaMos:** herramienta que permite modelar y desarrollar componentes bajo un paradigma como FragOP.

**Desarrollo:**

1. Creamos los siguientes directorios idealmente en C://

|  |
| --- |
| assets/  assets/Categoria/  assets/AdminCategorias/  assets/VistasBasicas/ |

**Iniciamos la implementación del archivo “admincategorias.jsp”.**

1. Creamos la vista admincategorias.jsp (en la ruta assets/AdminCategorias/) con el siguiente contenido:

|  |
| --- |
| <%@ include file="header.jsp" %>  <div class="main">  <div class="container" style="margin-top: 60px;">         <h1>Listado de Categorias</h1>  </div>  </div>  <div class="container">     <table>     <tr>         <th width="80">ID</th>         <th width="120">Nombre</th>         <th width="120">Descripcion</th>     </tr>     <c:forEach items="${categorias}" var="cat">         <tr>             <td>${cat.getId()}</td>             <td>${cat.getNombre()}</td>             <td>${cat.getDescripcion()}</td>         </tr>     </c:forEach>  </table>  </div>  <%@ include file="footer.jsp" %> |

**Iniciamos la implementación del archivo “header.jsp”.**

1. Creamos la vista header.jsp (en la ruta assets/VistasBasicas/) con el siguiente contenido:

|  |
| --- |
| <%@ page contentType="text/html" pageEncoding="UTF-8"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <html>  <head>  <meta charset="utf-8">  <title>${title}</title>  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">  </head>  <body>  <nav class="navbar navbar-inverse navbar-fixed-top">      <div class="container">        <div id="navbar">          <ul class="nav navbar-nav">          <!--B-menu-elements-modificator-->          </ul>        </div>      </div>    </nav> |

**Iniciamos la implementación del archivo “footer.jsp”.**

1. Creamos la vista footer.jsp (en la ruta assets/VistasBasicas/) con el siguiente contenido:

|  |
| --- |
| <div class="container">  <hr />  <footer>  <p>© By MyCompany</p>  </footer>  </div>  </body> |

**Iniciamos la implementación del archivo “Categoria.java”.**

1. Creamos la clase Categoria.java (en la ruta assets/Categoria/) con el siguiente contenido:

|  |
| --- |
| package org.pl.eshopping.models;  import java.util.ArrayList;  import java.util.List;  public class Categoria {  public Integer id;  public String nombre;  public String descripcion;    public Categoria(Integer i, String n, String d){  this.id=i;  this.nombre=n;  this.descripcion=d;  }    public static List<Categoria> categorias = new ArrayList<Categoria>() {  {  add(new Categoria(102, "Ropa", "ropa en general"));  add(new Categoria(103, "Gorras", "gorras en general"));  add(new Categoria(104, "Televisores", "Tvs en general"));  }  };    public String getNombre() { return nombre; }  public void setNombre(String nombre) { this.nombre = nombre; }  public Integer getId() { return id; }  public void setId(Integer id) { this.id = id; }  public String getDescripcion() { return descripcion; }  public void setDescripcion(String descripcion) { this.descripcion = descripcion; }  } |

**Iniciamos la implementación del archivo “AlterAdminCat.java”.**

1. Creamos el archivo AlterAdminCat.java (en la ruta assets/Categoria/) con el siguiente contenido:

|  |
| --- |
| Fragment Categoria-AlterAdminCat {    Action: replace    FragmentationPoint: category-data-collection    PointBracketsLan: java    Destination: AdminCategorias-AdminCategorias    SourceCode: [ALTERCODE-FRAG]List<Categoria> categorias = Categoria.categorias;[/ALTERCODE-FRAG]  }  Fragment Categoria-AlterAdminCat2 {    Action: add    FragmentationPoint: importing-zone    PointBracketsLan: java    Destination: AdminCategorias-AdminCategorias    SourceCode: [ALTERCODE-FRAG]import org.pl.eshopping.models.Categoria;[/ALTERCODE-FRAG]  } |

**Iniciamos la implementación del archivo “AdminCategorias.java”.**

1. Creamos la clase AdminCategorias.java (en la ruta assets/AdminCategorias/) con el siguiente contenido:

|  |
| --- |
| package org.pl.eshopping.controllers;  import java.io.IOException;  import java.util.List;  import javax.servlet.RequestDispatcher;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  /\*B-importing-zone\*/  @WebServlet("/AdminCategorias")  public class AdminCategorias extends HttpServlet {        /\*B-category-data-collection\*/        List<Categoria> categorias = null;        /\*E-category-data-collection\*/    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {            RequestDispatcher view = request.getRequestDispatcher("admincategorias.jsp");          request.setAttribute("categorias", categorias);          view.forward(request, response);    }  } |

**Iniciamos la implementación del archivo “AlterHeader.frag”.**

1. Creamos el archivo AlterHeader.frag (en la ruta assets/AdminCategorias/) con el siguiente contenido:

|  |
| --- |
| Fragment AdminCategorias-AlterHeader {    Action: add    FragmentationPoint: menu-elements-modificator    PointBracketsLan: html    Destination: VistasBasicas-Header    SourceCode: [ALTERCODE-FRAG]<li><a href="<c:url value='AdminCategorias'/>">Admin Categorias</a></li>[/ALTERCODE-FRAG]  } |

**Modificación:**

En este caso no necesitamos modificar ningún componente.

**Integración:**

Para ver los componentes integrados debemos:

1. Iniciamos VariaMos, seleccionado la opción “Component-based project”.
2. Cargamos el archivo TiendaVirtualModel.vmum (desde File -> Load File)
3. Luego damos clic en “Domain implementation” -> “Configure integration” y configuramos tanto la ruta raíz donde almacenamos todos los componentes, como la ruta donde queremos que queden ensamblados.
4. Damos clic en “Domain implementation” -> “Execute integration”. Lo cual integra todos los archivos.
5. Utilizando Eclipse creamos un nuevo proyecto tipo “Dynamic Web Project” llamado “FragOPEShopping” y utilizamos Apache Tomcat como ejecutor.
6. Importamos el archivo .jar de la librería JSTL en la ruta WebContent/WEB-INF/lib/
7. Copiamos y pegamos los archivos generados por VariaMos en el nuevo proyecto.

**Finalmente:**

Corremos Apache Tomcat y nos debería salir algo como lo siguiente:

