GUÍA Nº 3 – ARQUITECTURA EN CAPAS

FACULTAD	CURSO	AMBIENTE
INGENIERÍA	DISEÑO Y ARQUITECTURA DE SOFTWARE	LABORATORIO DE DISEÑO Y ARQUITECTURA DE SOFTWARE 77C0206

ELABORADO POR	ANÍBAL SARDÓN PANIAGUA	**APROBADO POR**	ARTURO RIVERA
VERSIÓN	001	**FECHA DE APROBACIÓN**	01/03/2022

1. LOGRO GENERAL DE UNIDAD DE APRENDIZAJE

Al finalizar la segunda unidad, el estudiante diseña la arquitectura de un producto de software, haciendo uso de patrones arquitectónicos, elementos del modelo de diseño, clases de diseño, subsistemas, interfaces en la herramienta IBM RSA; cumpliendo con los requerimientos fundamentales del software a construir.

2. OBJETIVOS ESPECÍFICOS DE LA PRÁCTICA

- Diseñar una arquitectura en capas con herramientas de diseño
- Entender el funcionamiento de la arquitectura en capas.
- Conocer la arquitectura en capas con JEE.

3. MATERIALES Y EQUIPOS

- Computadoras Personales.
- Sistema Operativo Windows.
  Star UML
  Netbeans
  Wildfly
- Pizarra
- Plumón
- Mota

4. PAUTAS DE SEGURIDAD

Las computadoras y laptops deben de estar prendidas mientras se usan. Pero al terminar el laboratorio estas deben dejarse apagadas.

- En el laboratorio debe estar prendido el aire acondicionado para evitar sobrecalentamientos y averías, especialmente en épocas de altas temperaturas.
- Los estudiantes no pueden llevar alimentos que puedan derramar sobre los computadores.
- Computadoras, router, switch, puntos de acceso (caídas).
- Eléctricos, por contacto directo o indirecto, electricidad estática y por fenómeno térmico. Puede producir: electrocuciones y quemaduras.
- Procedimiento ante Corte de Energía Eléctrica
- No tocar el equipo eléctrico en el que se encuentra trabajando, puede que retorne la energía.
- Comunicarse con el Asistente de Operaciones de turno quien se comunicará con el Técnico.

5. FUNDAMENTO

La asignatura de Diseño y Arquitectura de Software es de carácter teórico-práctico y tiene el propósito de potenciar en el estudiante sus habilidades para analizar y diseñar una arquitectura de software. Se desarrolla los siguientes contenidos: Introducción a la arquitectura de software, vistas y estilos de la arquitectura, requisitos de calidad de un software, diagramación UML orientada al diseño arquitectónico de software, patrones de arquitectura, arquitectura orientada a servicios (SOA), Arquitecturas en Cloud Computing , Arquitecturas para software en dispositivos móviles y documentación de una arquitectura de software.

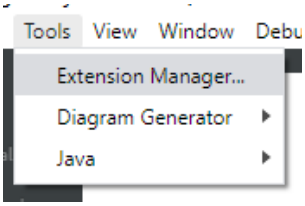
6. PROCEDIMIENTO (DESARROLLO DE LA PRÁCTICA)

Ejemplo 01: Diseño del Modelo 4 + 1

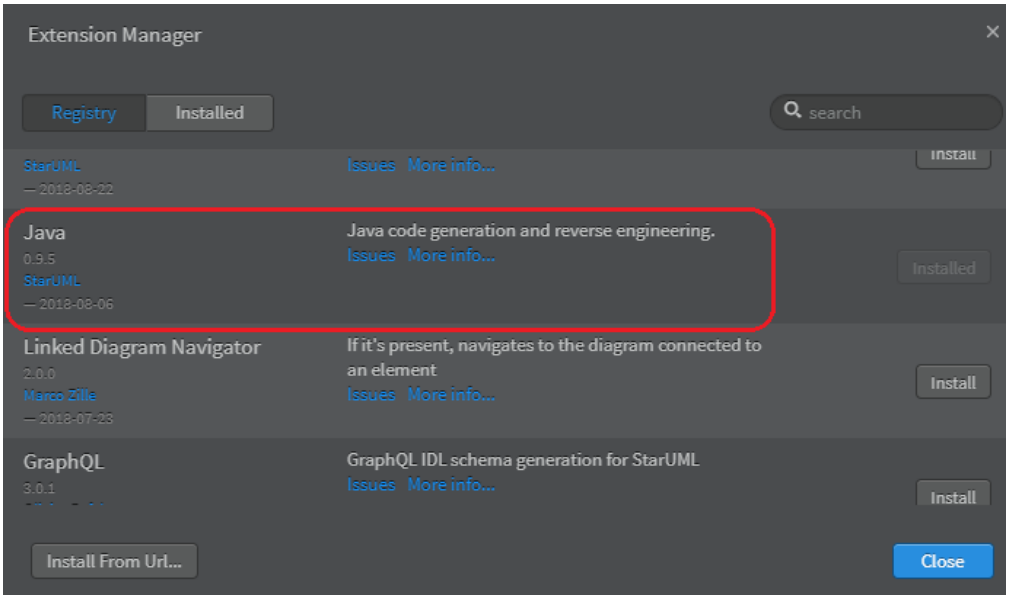
1. Agregar extensión para generar código Java.

Vamos a ingresar al Extension Manager para agregar la extensión que genera código en Java.

Hacemos clic en Tools **Extension Manager**.

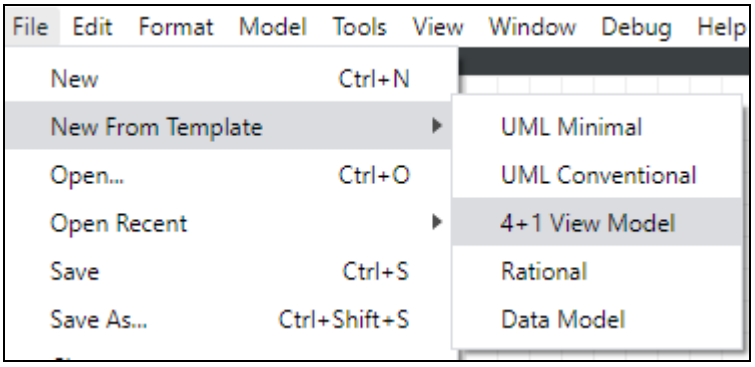


Buscamos y seleccionamos la librería Java y hacemos clic en el botón **Install**.

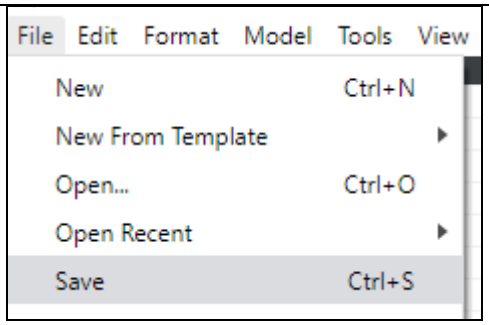


2. Creación del Proyecto Modelo con Star UML

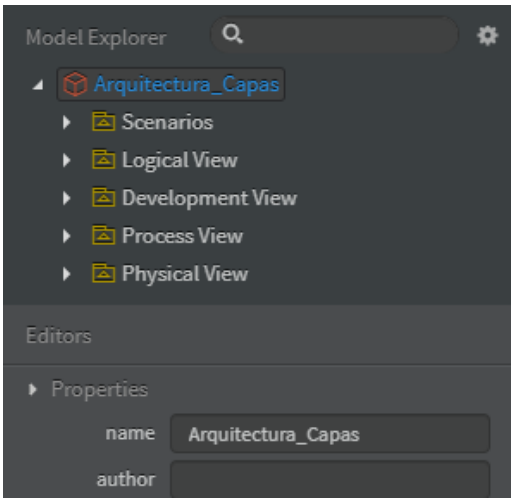
Creamos un proyecto modelo (**Arquitectura_Capas**) de tipo **4+1 View Model**.



Grabamos el proyecto **Arquitectura_Capas**



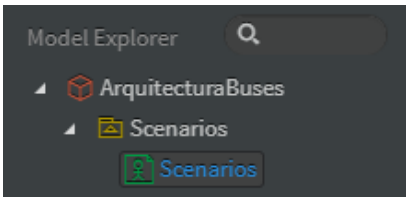
Ponemos nombre al proyecto con el **Model Explorer**



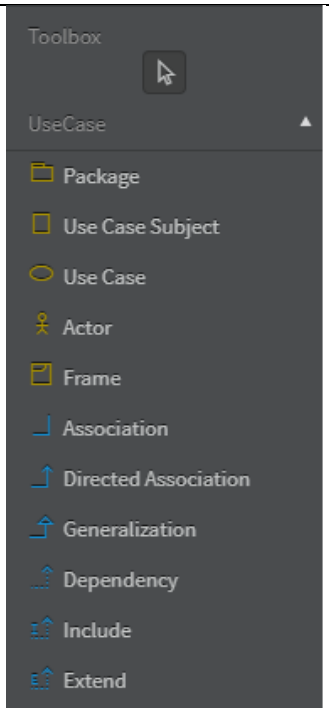
3. Creación de las Vistas

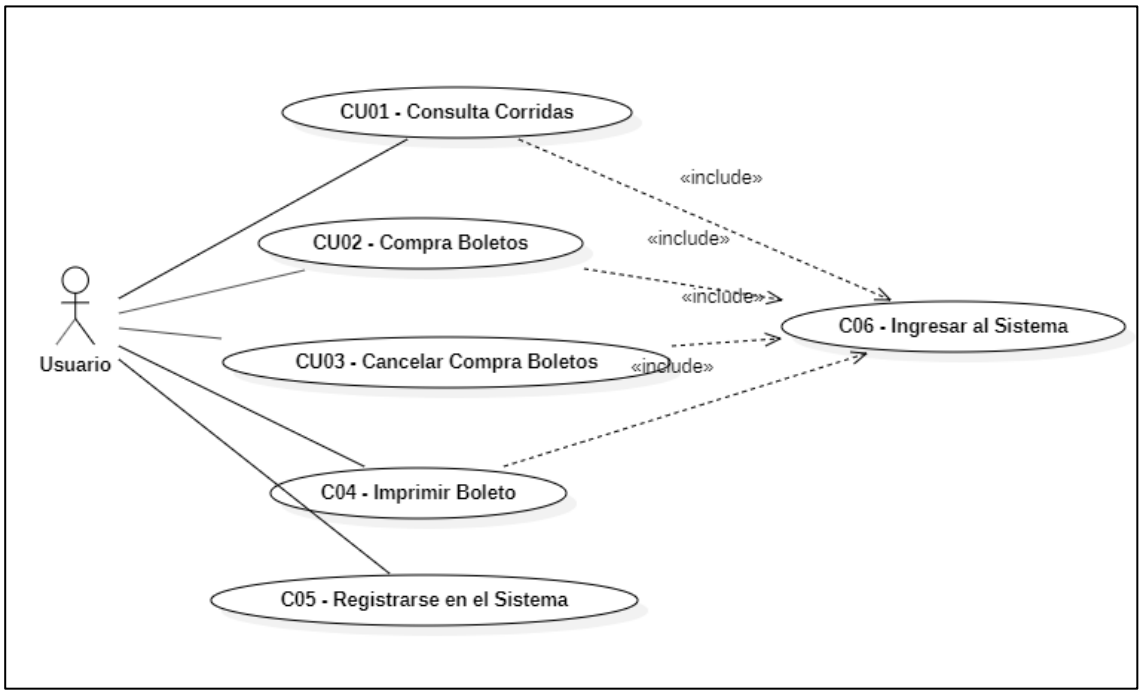
3.1 Creación de la Vista de Casos de Uso

Seleccionamos la **carpeta Scenarios** y hacemos doble clic sobre el **diagrama Scenarios.**

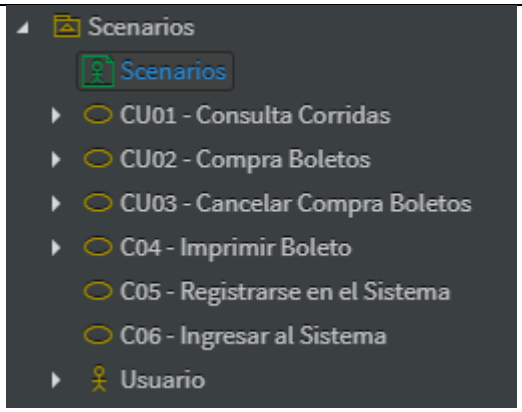


Vamos a seleccionar desde el **ToolBox** los elementos y agregaremos *los Actores, Casos de Uso y Relaciones* al lienzo, para crear el siguiente diagrama de Casos de Uso:



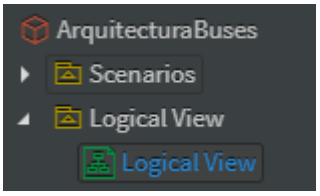


Al final en el **Model Explorer** se agregaran los siguientes elementos:

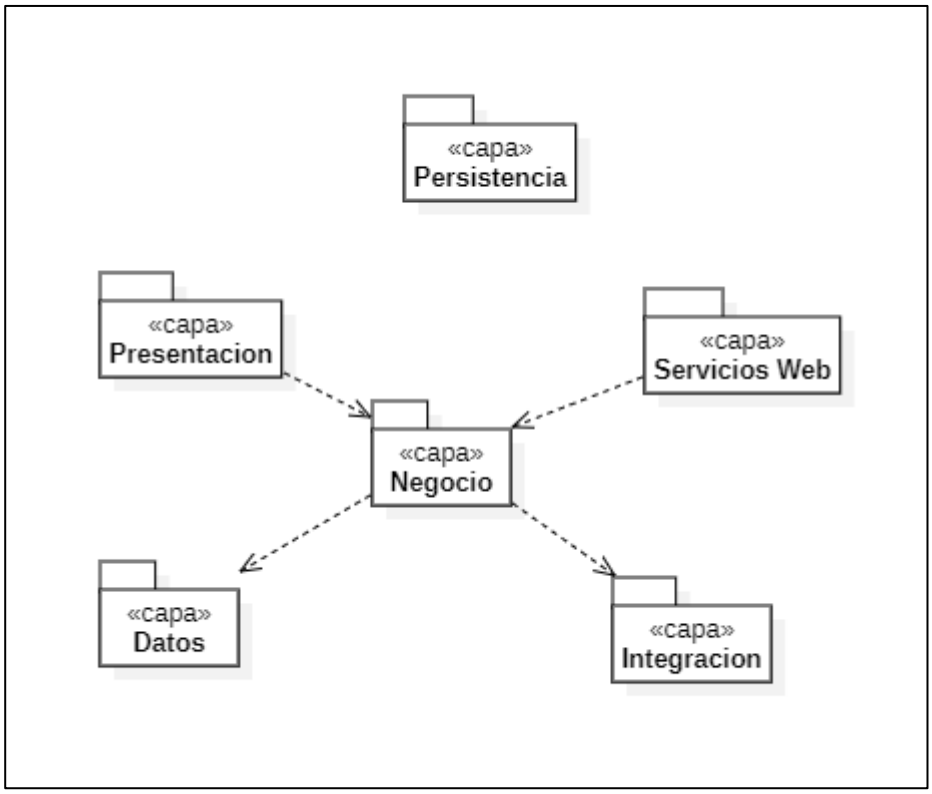


3.2 Creación de la Vista de Lógica

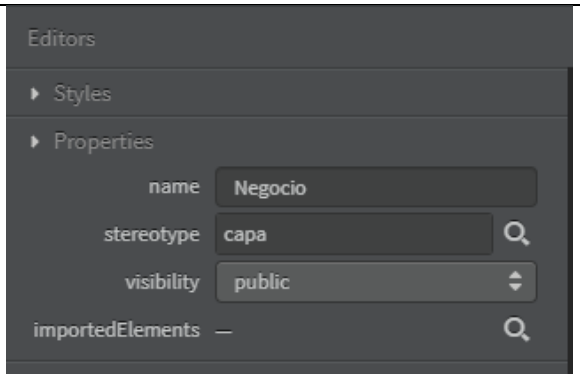
Seleccionamos la **carpeta Logical View** y hacemos doble clic sobre el **diagrama Logical View.**



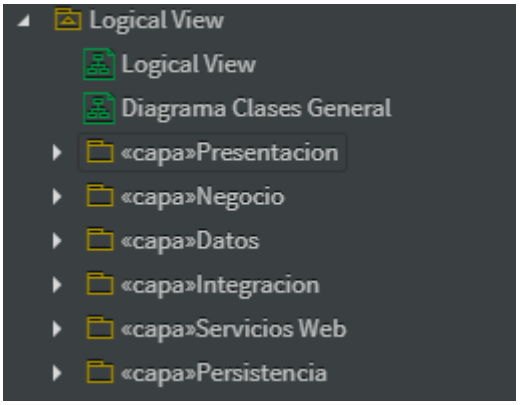
Vamos a seleccionar desde el **ToolBox** los elementos y agregaremos *los Paquetes, y Relaciones de dependencia* al lienzo, para crear el siguiente diagrama de paquetes:



En la **ventana Editors** ingresamos a las **properties** para asignarle los **name** y **stereotype** de cada paquete.



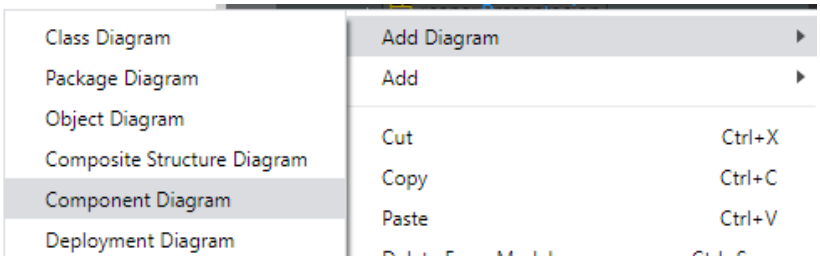
Al final en el **Model Explorer** se agregaran los siguientes elementos:



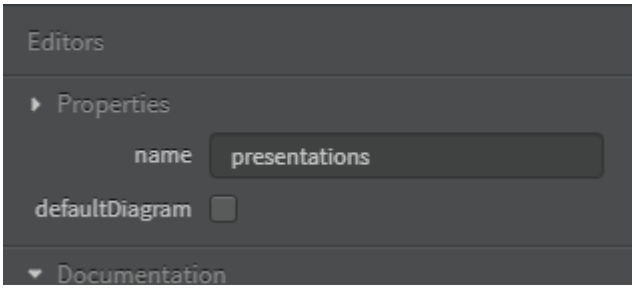
3.2.1 Diseño de la Capa de Presentación

Vamos a agregar un ***Diagrama de Componentes:***

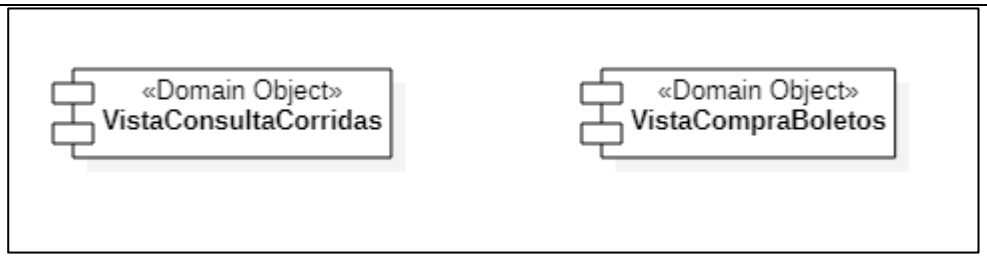
Seleccionamos la **carpeta <<capa>> Presentacion** y hacemos clic derecho **Add Diagram | Component Diagram.**



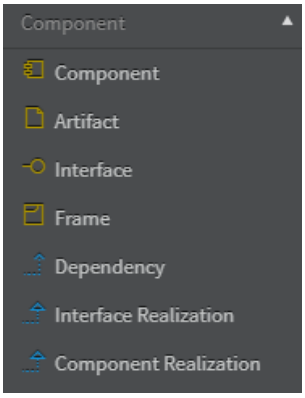
Le colocamos un nombre en la ventana **Editors**: **presentations**



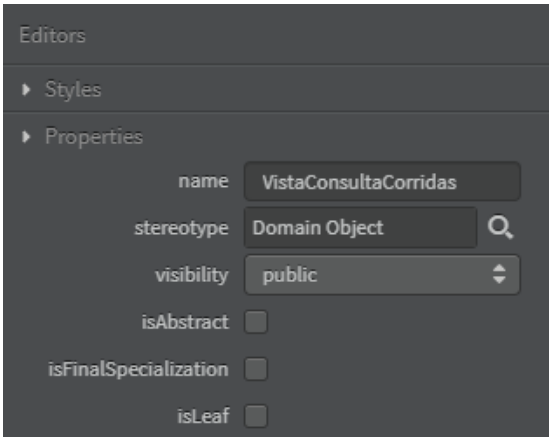
Agregamos los siguientes componentes:



Arrastramos los componentes desde el **ToolBox**

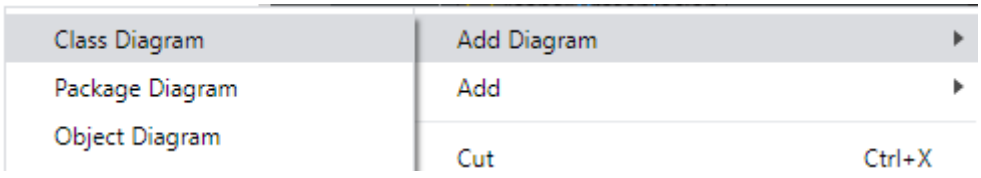


En la ventana **Editors** colocamos los nombres y propiedades de cada **componente**:

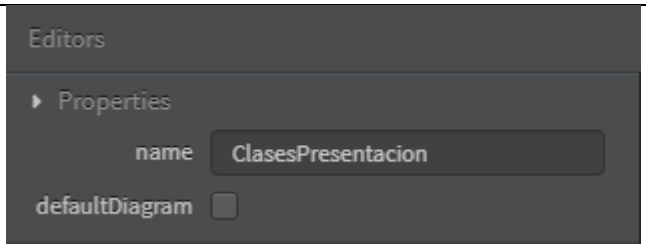


Vamos a agregar un ***Diagrama de Clases:***

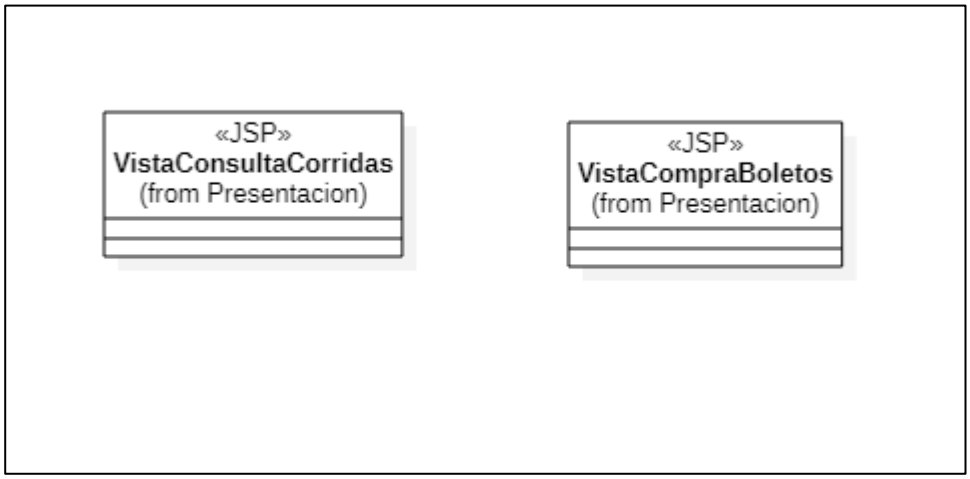
Seleccionamos la **carpeta <<capa>> Presentacion** y hacemos clic derecho **Add Diagram | Class Diagram.**



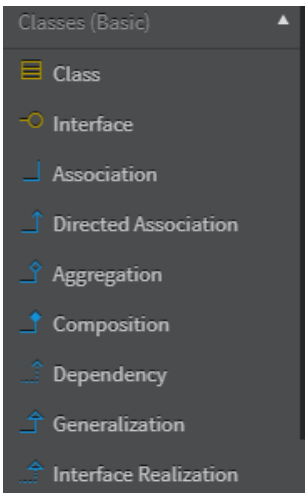
Le colocamos un nombre en la ventana **Editors**: **ClasesPresentation**



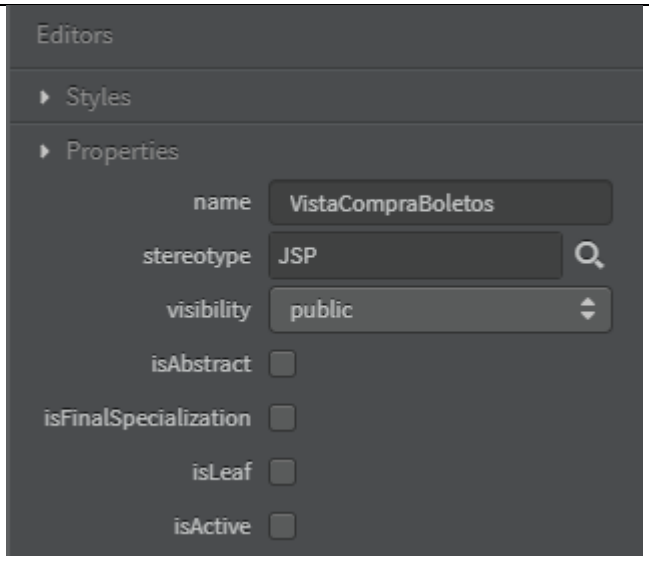
Agregamos las siguientes clases:



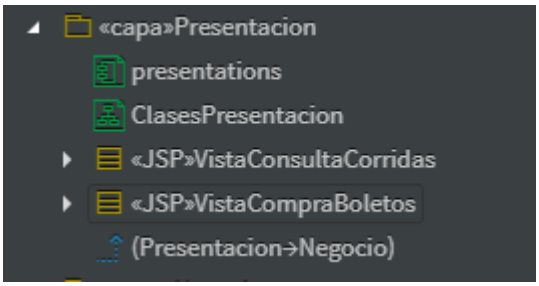
Arrastramos los componentes desde el **ToolBox**



En la ventana **Editors** colocamos los nombres y propiedades de cada **clase**:



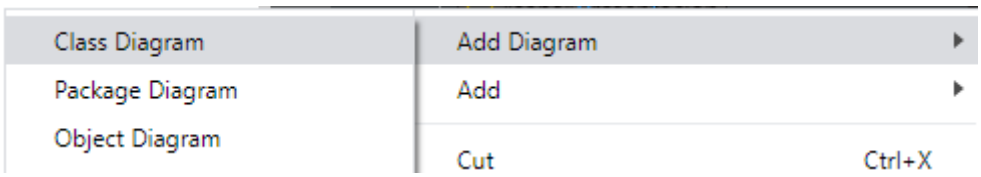
Al final en el **Model Explorer** se agregaran los siguientes elementos en el **Paquete Presentacion**:



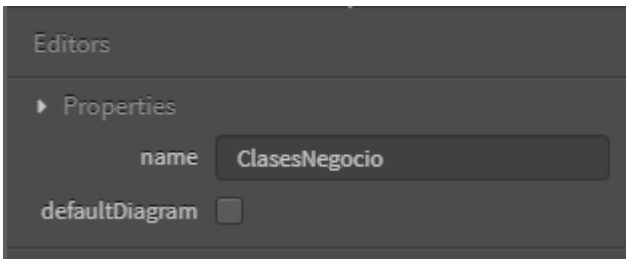
3.2.2 Diseño de la Capa de Negocio

Vamos a agregar un ***Diagrama de Clases:***

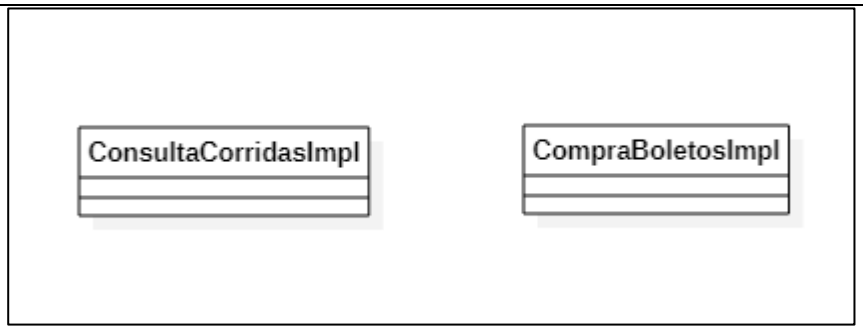
Seleccionamos la **carpeta <<capa>> Negocio** y hacemos clic derecho **Add Diagram | Class Diagram.**



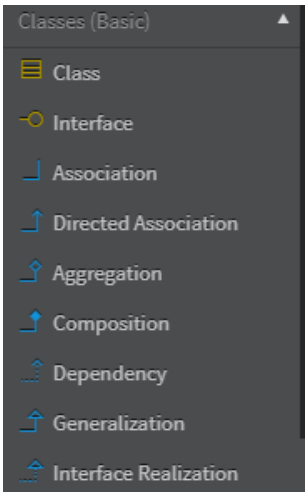
Le colocamos un nombre en la ventana **Editors**: **ClasesNegocio**



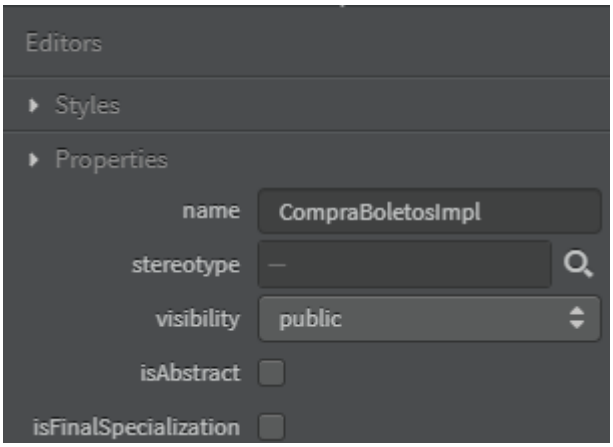
Agregamos las siguientes clases:



Arrastramos los componentes desde el **ToolBox**

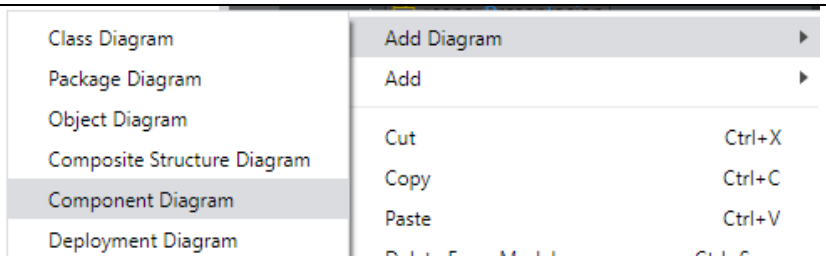


En la ventana **Editors** colocamos los nombres y propiedades de cada **clase**:

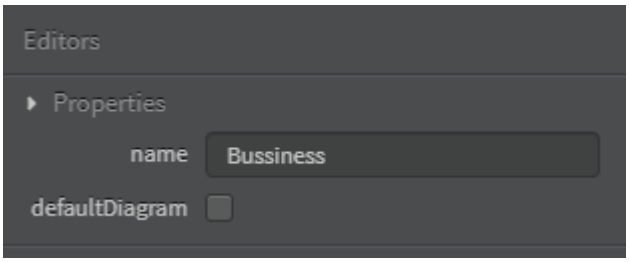


Vamos a agregar un ***Diagrama de Componentes:***

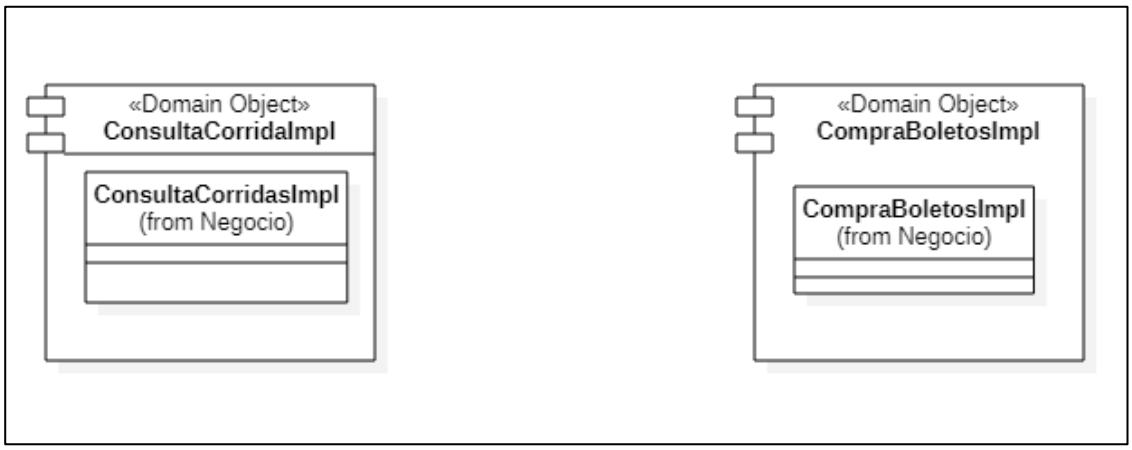
Seleccionamos la **carpeta <<capa>> Negocio** y hacemos clic derecho **Add Diagram | Component Diagram.**



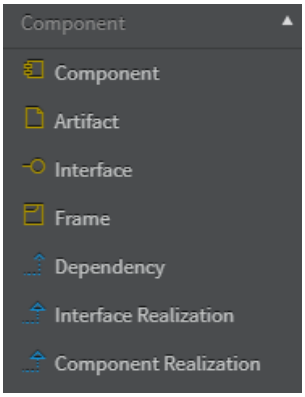
Le colocamos un nombre en la ventana **Editors**: **Bussiness**



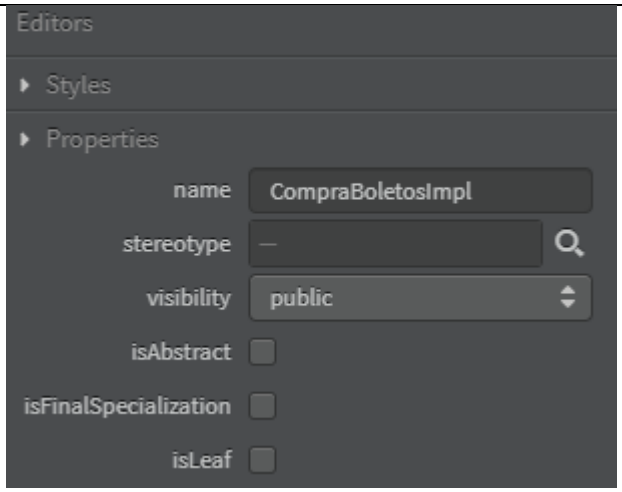
Agregamos los siguientes componentes:



Arrastramos los componentes desde el **ToolBox**



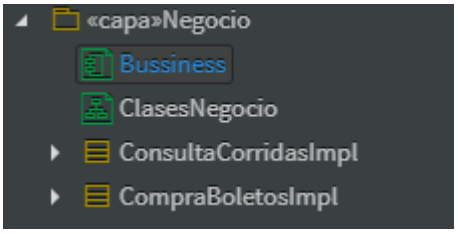
En la ventana **Editors** colocamos los nombres y propiedades de cada **componente**:



NOTA:

Las clases **ConsultaCorridasImpl** y **CompraBoletosImpl** los arrastramos desde la ventana **Model Explorer** y los colocamos dentro de los Componentes.

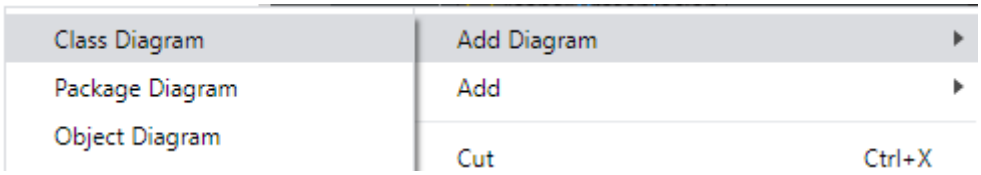
Al final en el **Model Explorer** se agregaran los siguientes elementos en el **Paquete Negocio**:



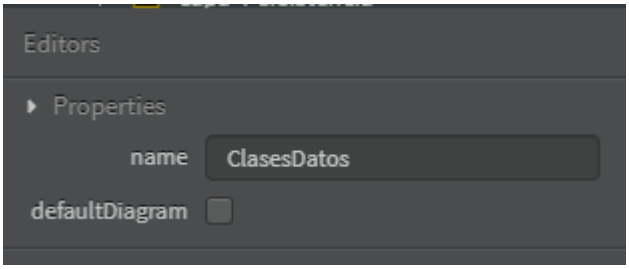
3.2.3 Diseño de la Capa de Datos

Vamos a agregar un ***Diagrama de Clases:***

Seleccionamos la **carpeta <<capa>> Datos** y hacemos clic derecho **Add Diagram | Class Diagram.**

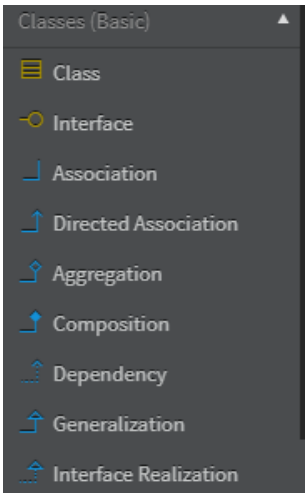


Le colocamos un nombre en la ventana **Editors**: **ClasesDatos**

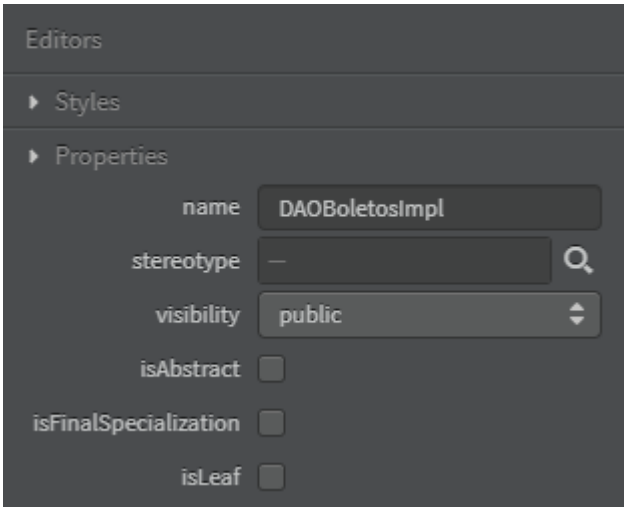


Agregamos las siguientes clases:

Arrastramos los componentes desde el **ToolBox**

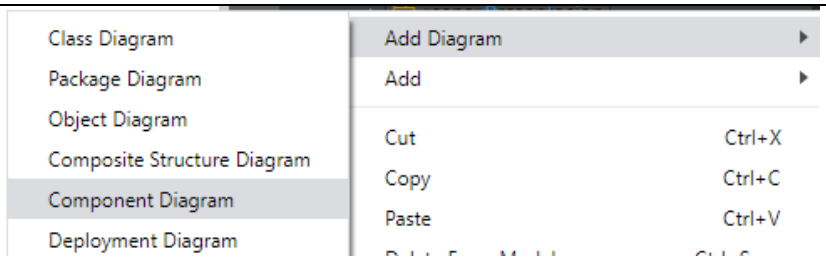


En la ventana **Editors** colocamos los nombres y propiedades de cada **clase**:

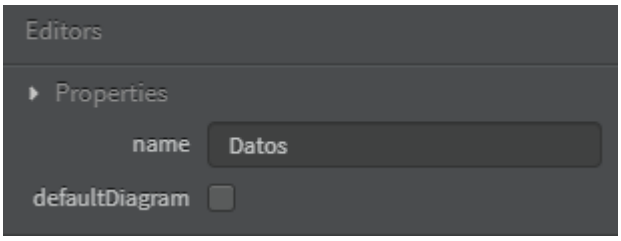


Vamos a agregar un ***Diagrama de Componentes:***

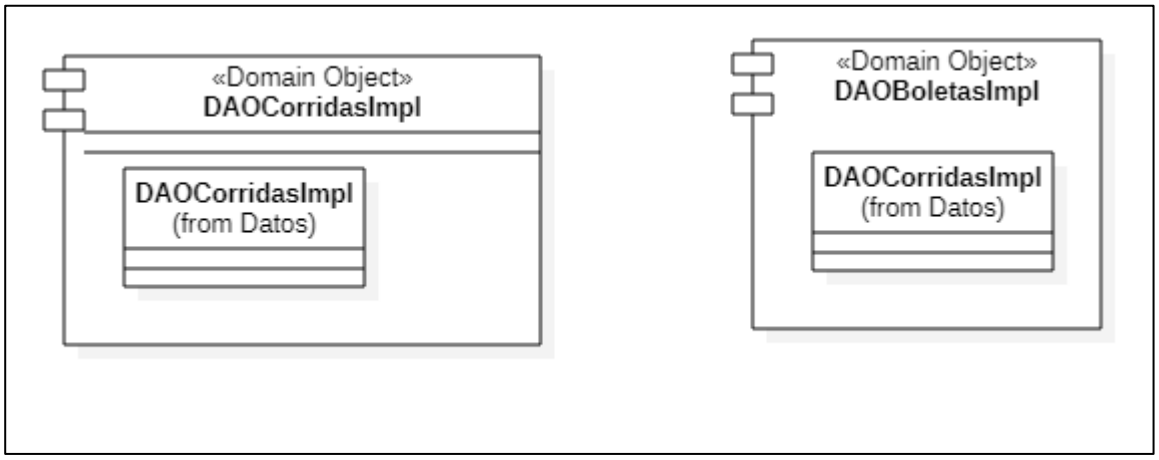
Seleccionamos la **carpeta <<capa>> Datos** y hacemos clic derecho **Add Diagram | Component Diagram.**



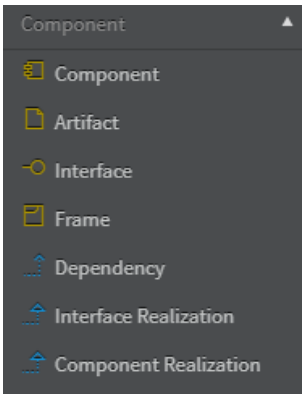
Le colocamos un nombre en la ventana **Editors**: **Datos**



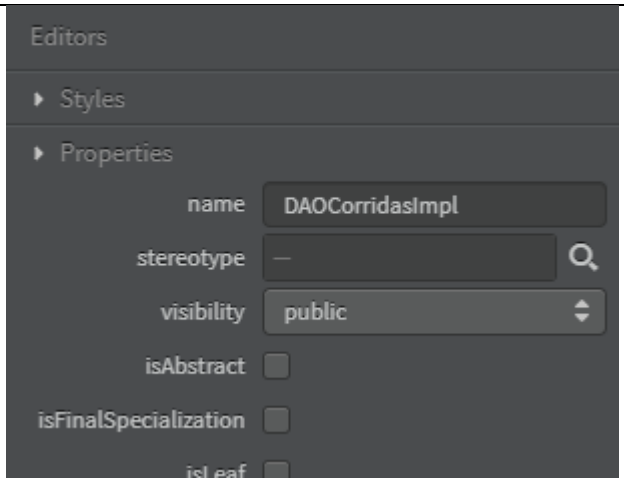
Agregamos los siguientes componentes:



Arrastramos los componentes desde el **ToolBox**



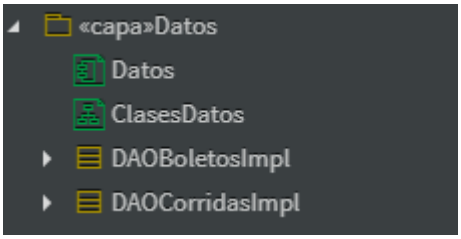
En la ventana **Editors** colocamos los nombres y propiedades de cada **componente**:



NOTA:

Las clases **DAOCorridasImpl** y **DAOBoletosImpl** los arrastramos desde la ventana **Model Explorer** y los colocamos dentro de los Componentes.

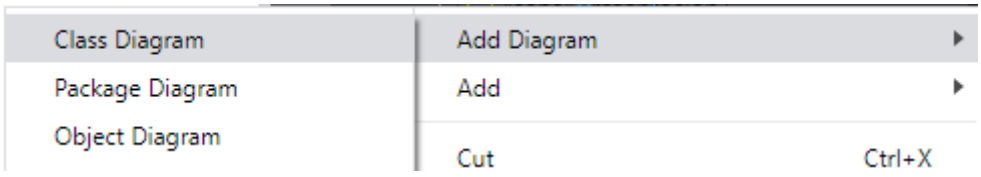
Al final en el **Model Explorer** se agregaran los siguientes elementos en el **Paquete Datos**:



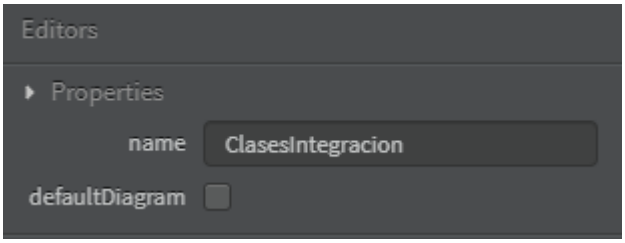
3.2.4 Diseño de la Capa de Integracion

Vamos a agregar un ***Diagrama de Clases:***

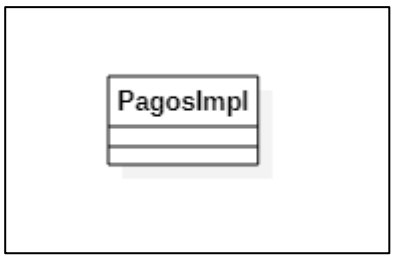
Seleccionamos la **carpeta <<capa>> Integracion** y hacemos clic derecho **Add Diagram | Class Diagram.**



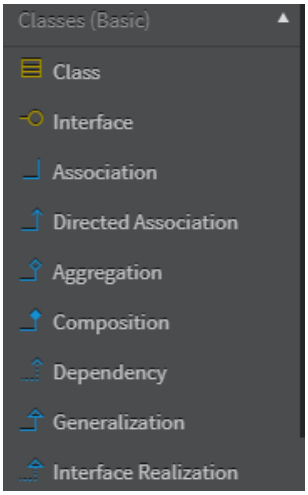
Le colocamos un nombre en la ventana **Editors**: **ClasesIntegracion**



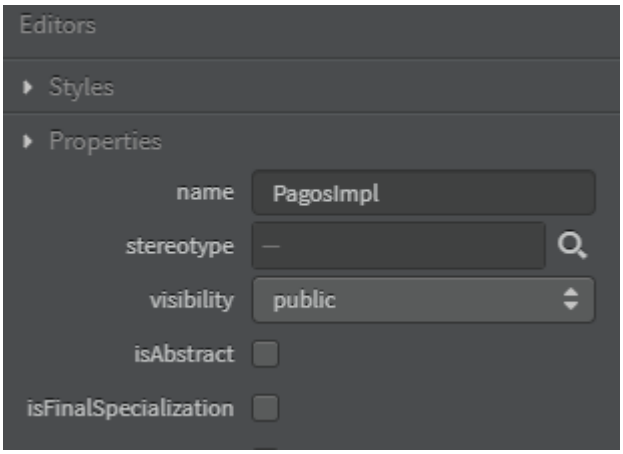
Agregamos las siguientes clases:



Arrastramos los componentes desde el **ToolBox**

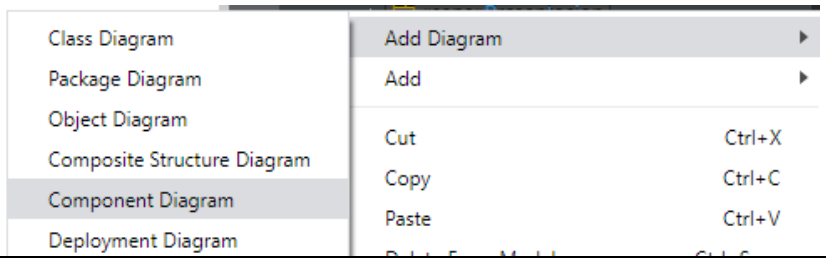


En la ventana **Editors** colocamos los nombres y propiedades de cada **clase**:

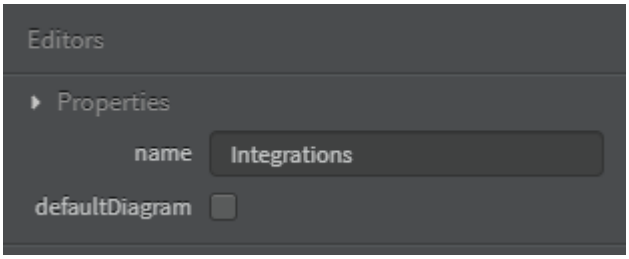


Vamos a agregar un ***Diagrama de Componentes:***

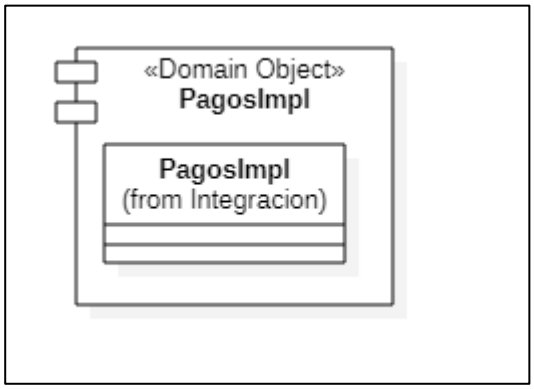
Seleccionamos la **carpeta <<capa>> Integracion** y hacemos clic derecho **Add Diagram | Component Diagram.**



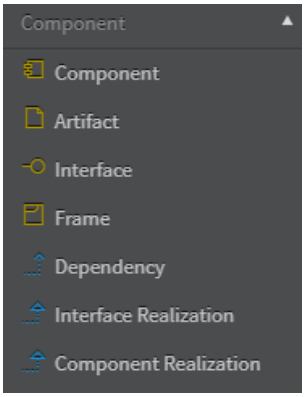
Le colocamos un nombre en la ventana **Editors**: **Integrations**



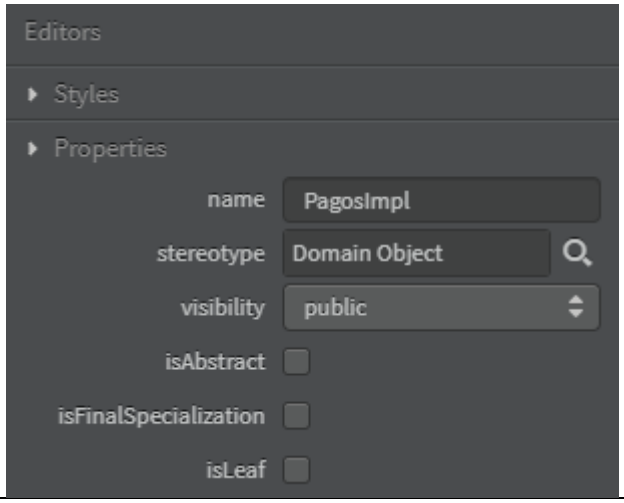
Agregamos los siguientes componentes:



Arrastramos los componentes desde el **ToolBox**



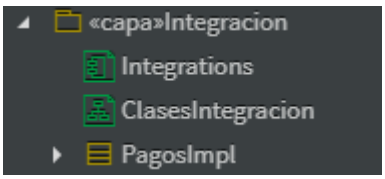
En la ventana **Editors** colocamos los nombres y propiedades de cada **componente**:



NOTA:

Las clases **PagossImpl** los arrastramos desde la ventana **Model Explorer** y los colocamos dentro del Componente.

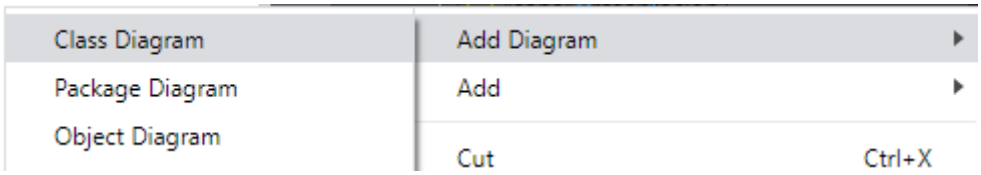
Al final en el **Model Explorer** se agregaran los siguientes elementos en el **Paquete Integracion**:



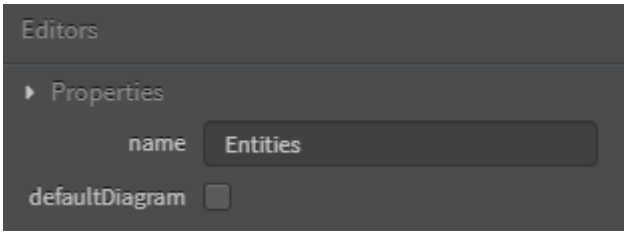
3.2.5 Diseño de la Capa de Persistencia

Vamos a agregar un ***Diagrama de Clases:***

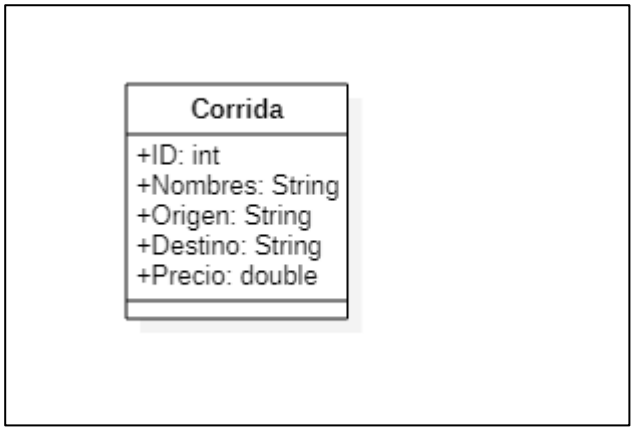
Seleccionamos la **carpeta <<capa>> Persistencia** y hacemos clic derecho **Add Diagram | Class Diagram.**



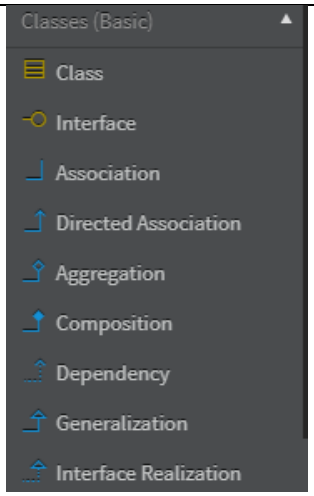
Le colocamos un nombre en la ventana **Editors**: **Entities**



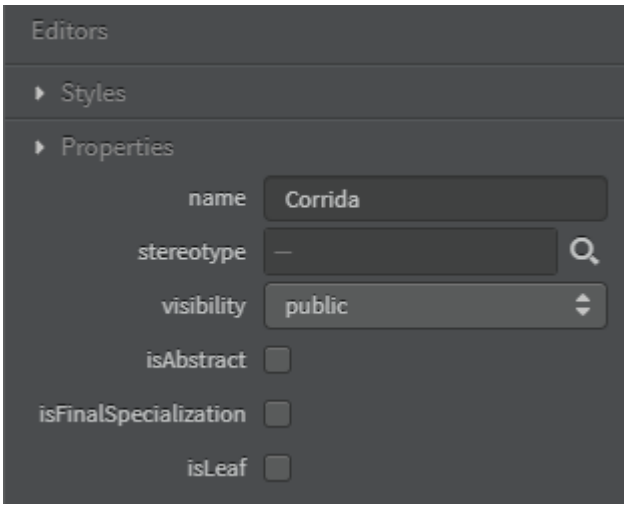
Agregamos las siguientes clases:



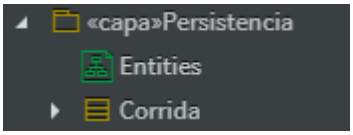
Arrastramos los componentes desde el **ToolBox**



En la ventana **Editors** colocamos los nombres y propiedades de cada **clase**:



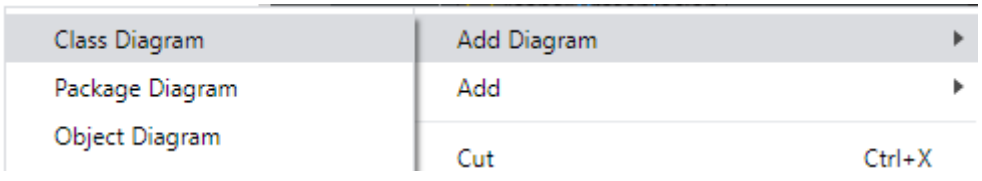
Al final en el **Model Explorer** se agregaran los siguientes elementos en el **Paquete Integracion**:



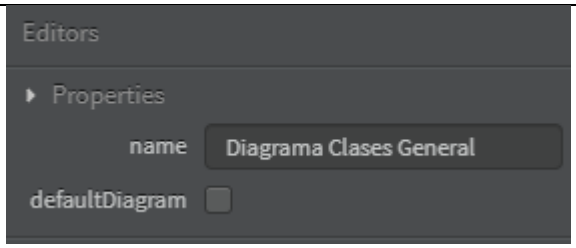
3.2.6 Diagrama de Clases General

Vamos a agregar un ***Diagrama de Clases:*** a la Vista **Logical View.**

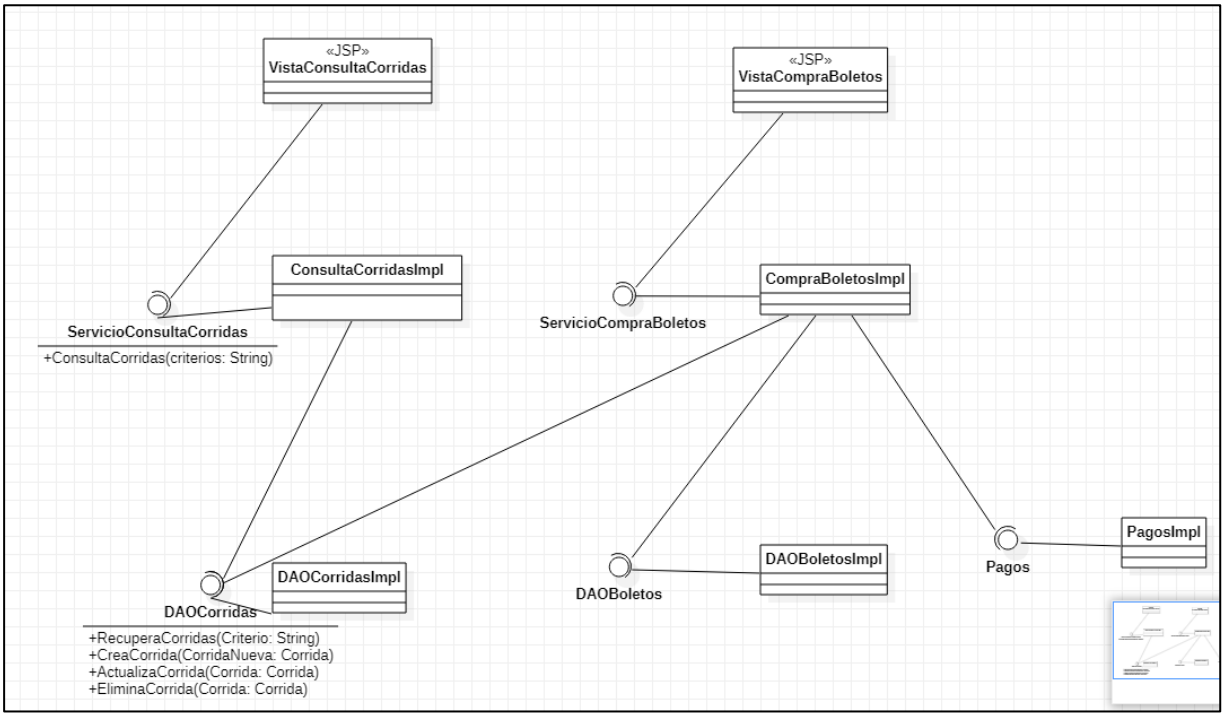
Seleccionamos la **carpeta Logical View** y hacemos clic derecho **Add Diagram | Class Diagram.**



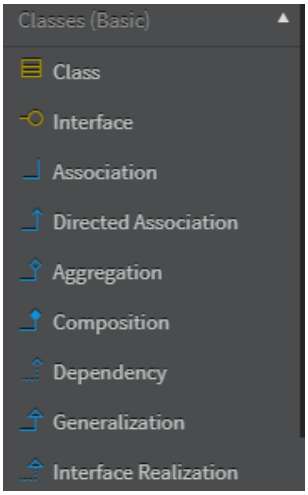
Le colocamos un nombre en la ventana **Editors**: **Diagrama Clases General**



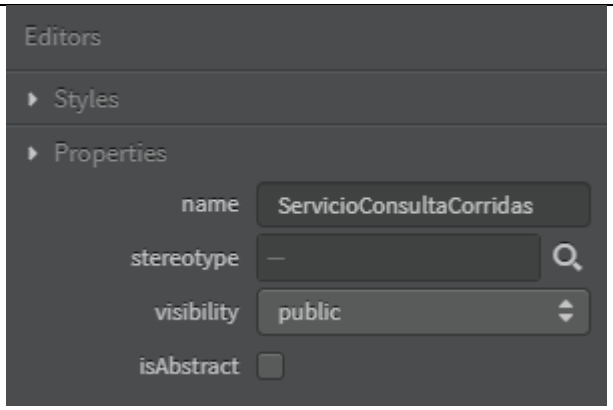
Vamos a arrastrar las Clases desde los **paquetes** de la Logical View desde la Ventana **Model Explorer**:



Vamos a agregar las Interfaces desde el **ToolBox**

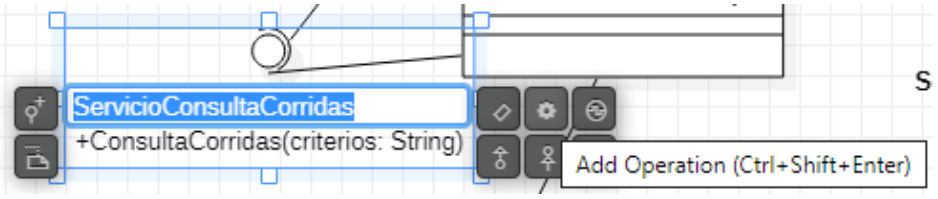


En la ventana **Editors** colocamos los nombres y propiedades de cada **Interface**:

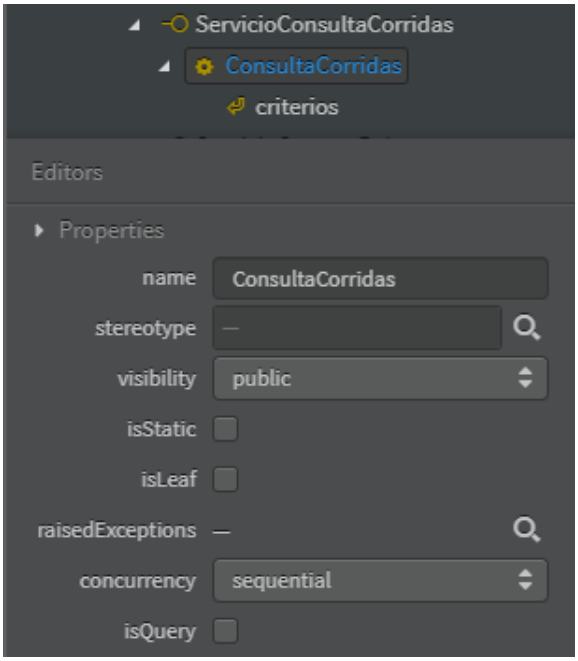


Vamos a agregar **métodos** a las interfaces:

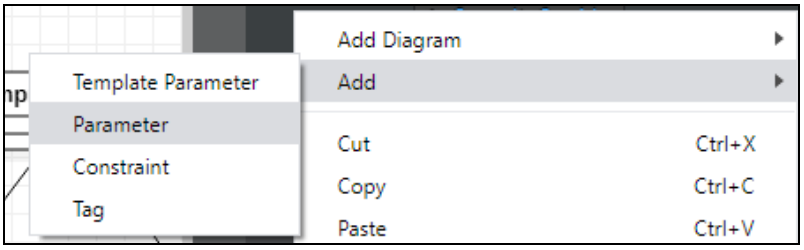
Hacemos doble clic sobre la **interface** hacemos **doble clic** sobre el botón **Add Operation**

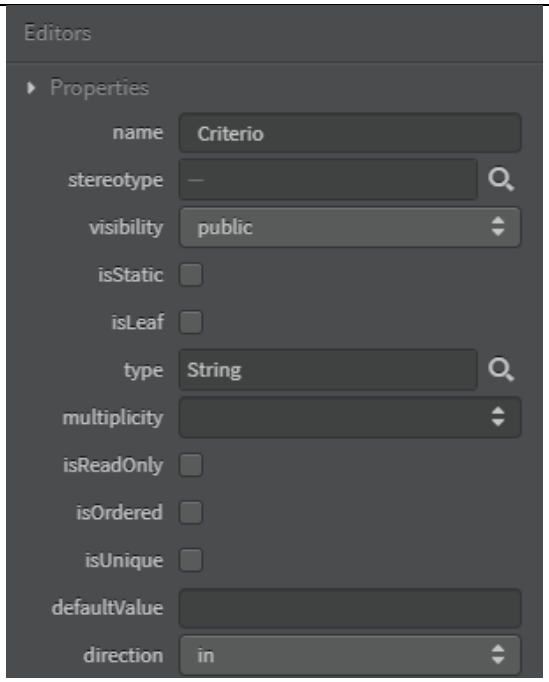


En la Ventana **Model Explorer** podemos asignarle el nombre al **método** y sus propiedades:

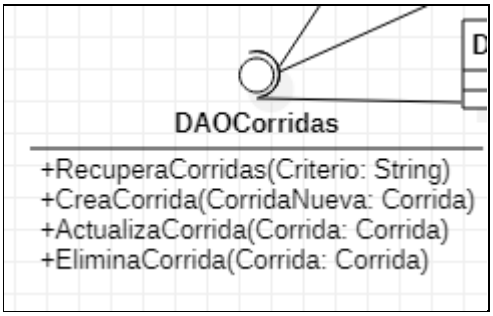


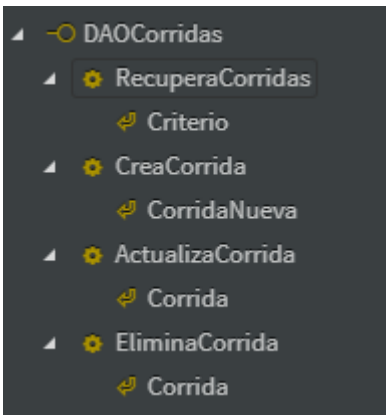
Para agregarle parámetros hacemos **clic derecho** sobre el **método** y elegimos **Add | Parameter**



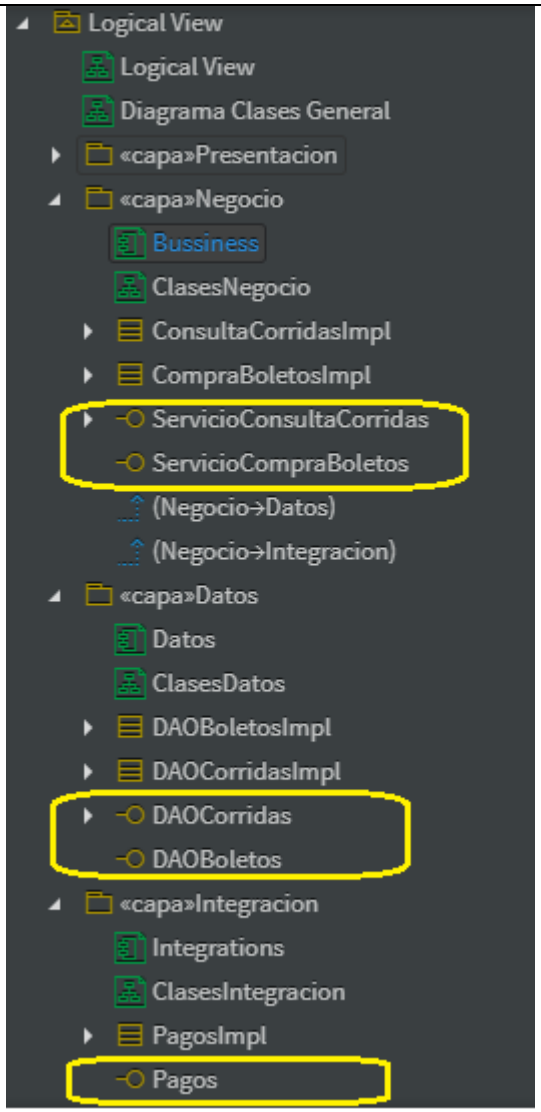


Repetimos los pasos para el resto de métodos:



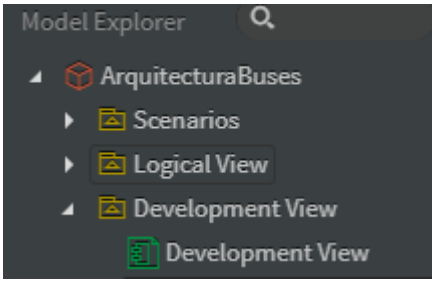


Al finalizar, mediante la ventana **Model Explorer**, vamos a mover las **interfaces** a sus **paquetes** correspondientes como se ve en la siguiente figura:



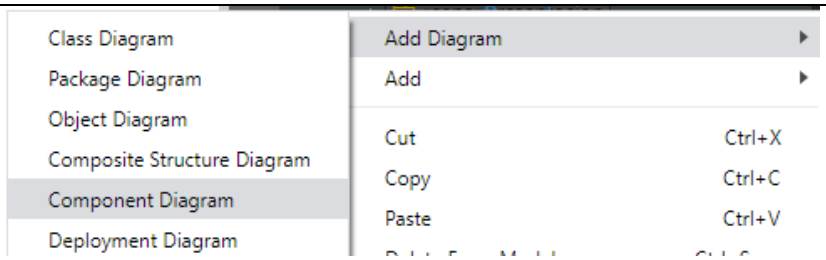
3.3 Creación de la Vista de Desarrollo

Seleccionamos la **carpeta Development View.**

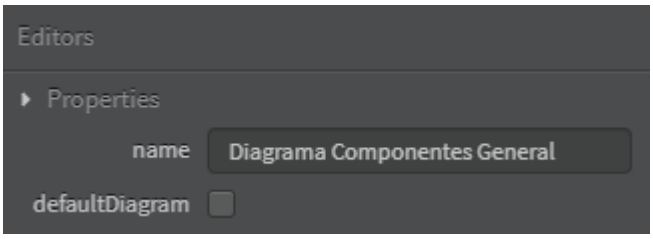


Vamos a agregar un ***Diagrama de Componentes:***

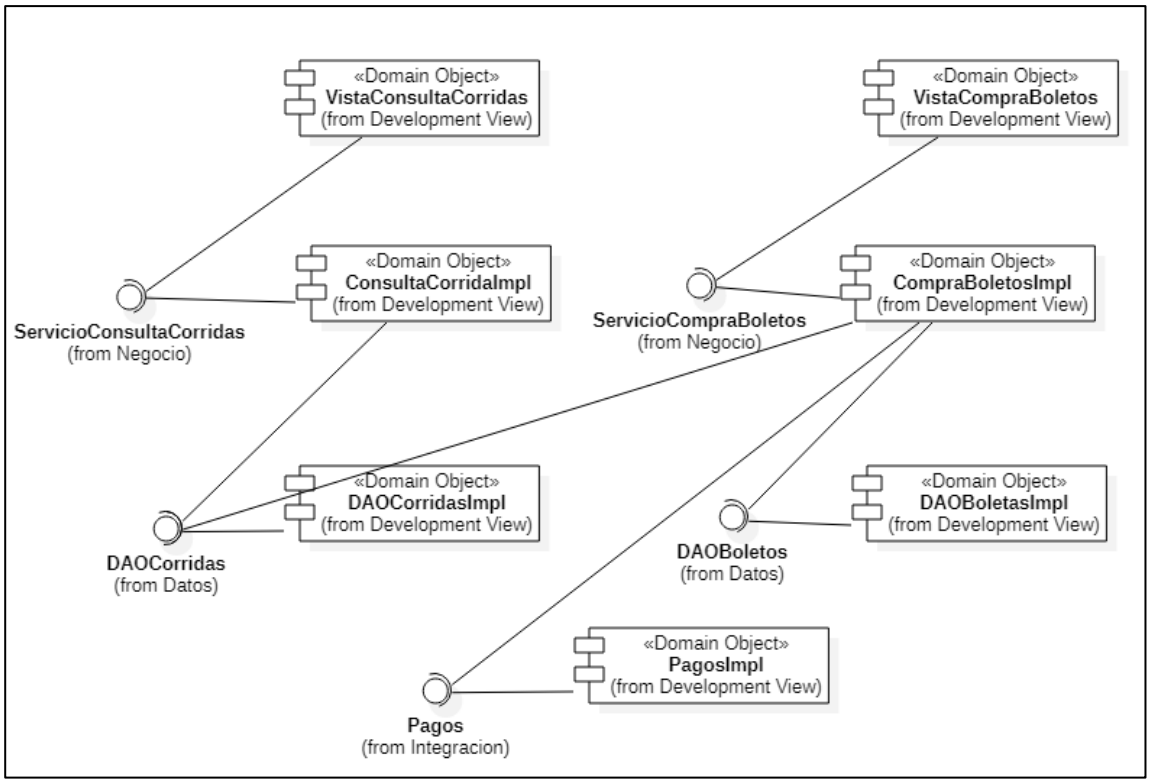
Seleccionamos la **carpeta Development View** y hacemos clic derecho **Add Diagram | Component Diagram.**



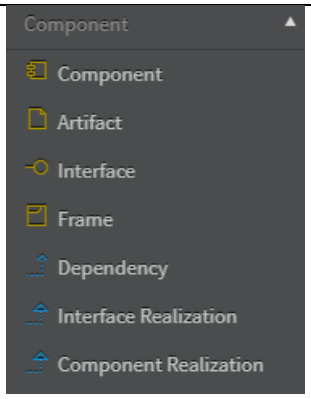
Le colocamos un nombre en la ventana **Editors**: **Diagrama Componentes General**



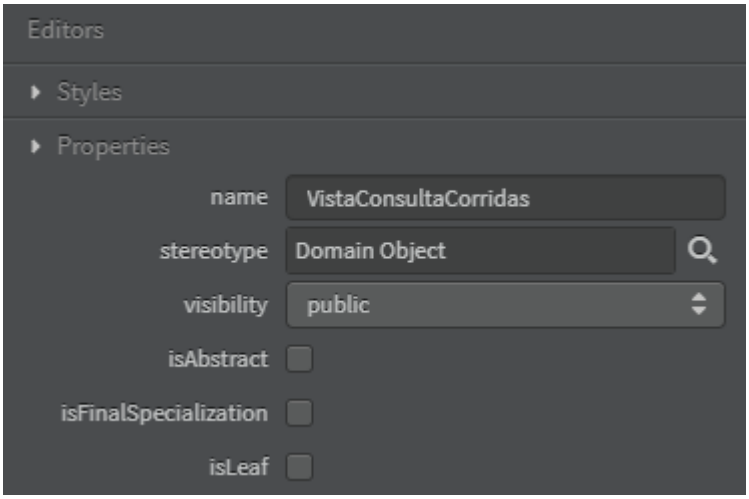
Vamos a seleccionar desde el **ToolBox** los elementos y agregaremos *los Component, Interface y Relaciones de dependencia* al lienzo, para crear el siguiente diagrama de componentes:



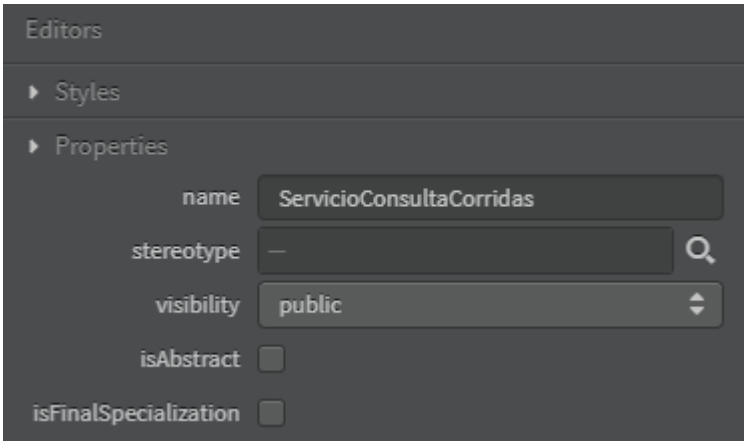
Arrastramos los componentes desde el **ToolBox**



En la ventana **Editors** colocamos los nombres y propiedades de cada **componente**:



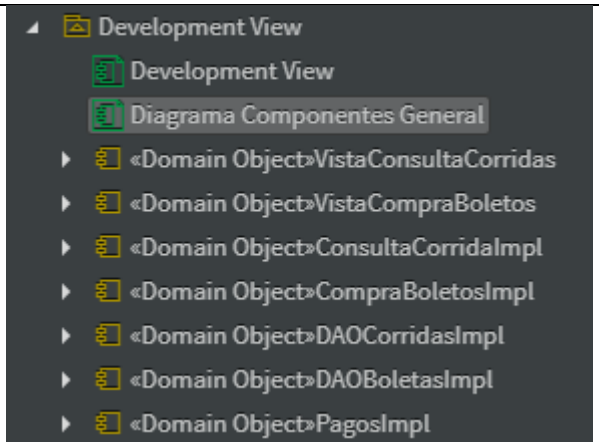
En la ventana **Editors** colocamos los nombres y propiedades de cada **interface**:



NOTA:

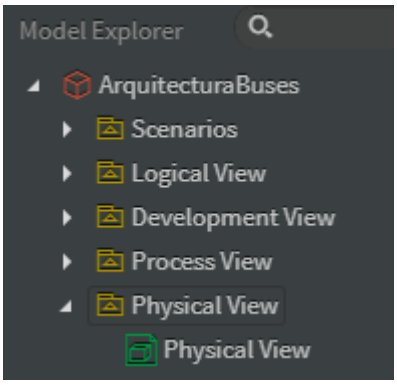
Las interfaces las arrastramos desde la ventana **Model Explorer** de la vista **Logical View** al lienzo.

Al final en el **Model Explorer** se agregaran los siguientes elementos en la **Vista Development**:

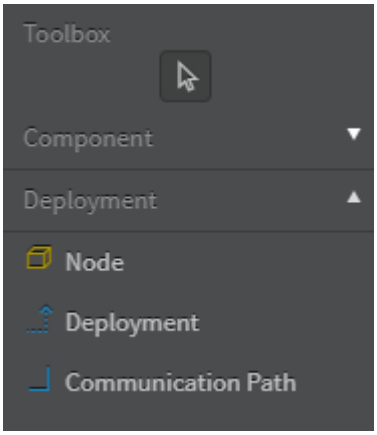


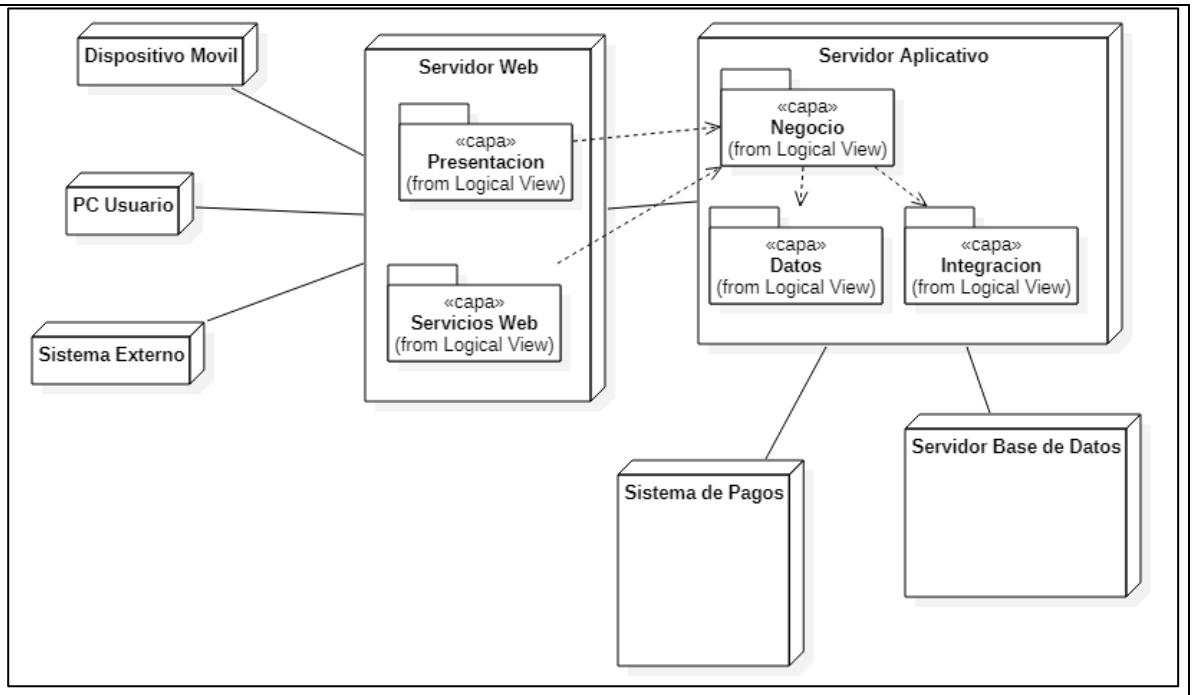
3.4 Creación de la Vista Física

Seleccionamos la **carpeta Physical View** y hacemos doble clic sobre el **diagrama Physical View.**



Vamos a seleccionar desde el **ToolBox** los elementos y agregaremos *los Nodos, y Relaciones Comunication Path* al lienzo, para crear el siguiente diagrama de despliegue:

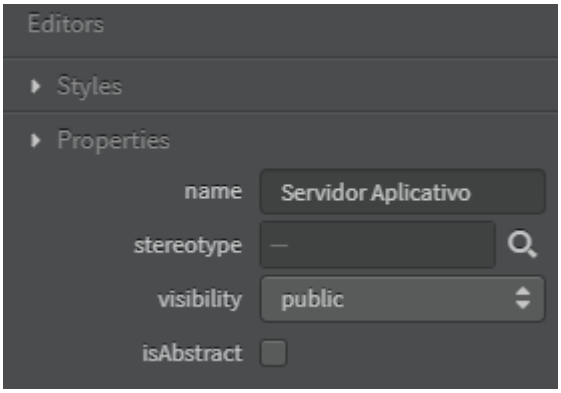




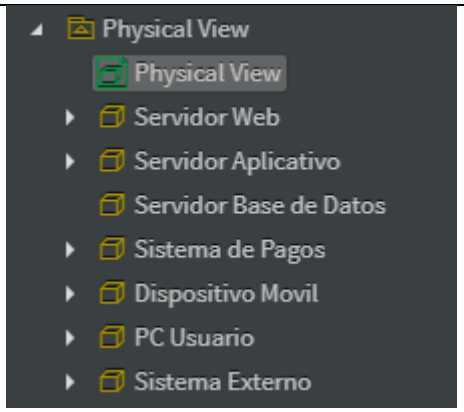
NOTA:

Los paquetes **Presentacion, Negocio, Datos, Integracion, Servicios Web** los arrastramos desde la ventana **Model Explorer** y los colocamos dentro de los Nodos.

En la **ventana Editors** ingresamos a las **properties** para asignarle los **name** y **stereotype** de cada Nodo.

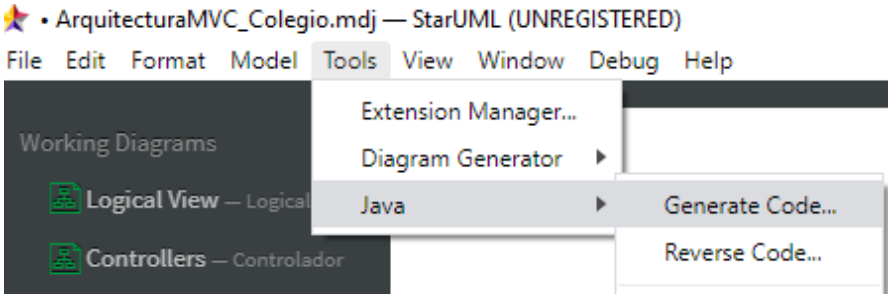


Al final en el **Model Explorer** se agregaran los siguientes elementos:

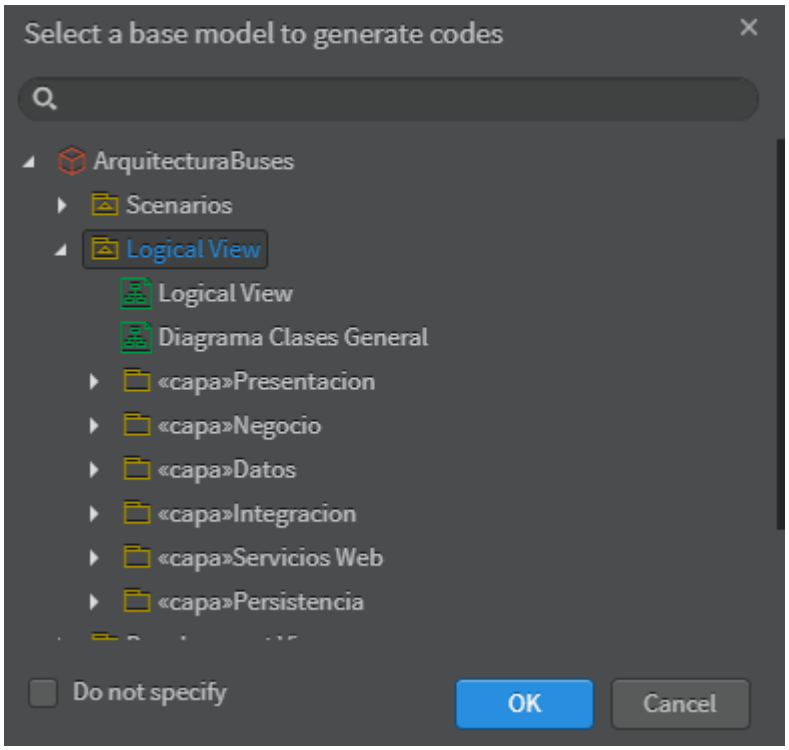


4. Generación de Código Fuente en JAVA para los elementos del MVC

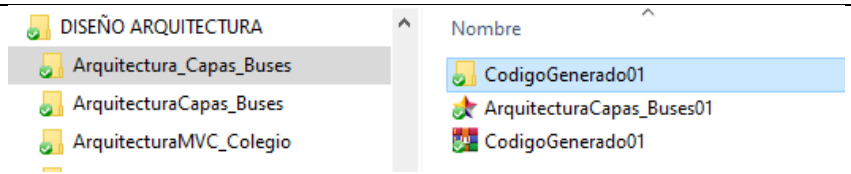
a) Hacemos Clic en el menú Tools – Java – Generate Code.



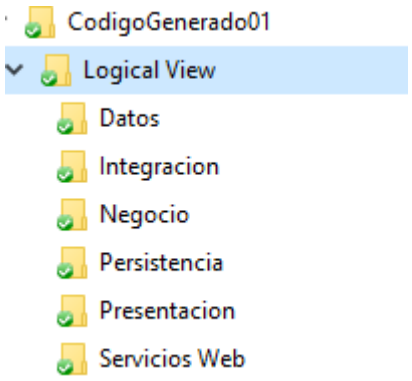
b) Seleccionamos la carpeta donde se encuentra el modelo "**Logical View"** para que transforme a código JAVA. Presionamos el **botón OK**



c) Seleccionamos la carpeta donde se va a grabar el Proyecto con el Código Generado en Java.



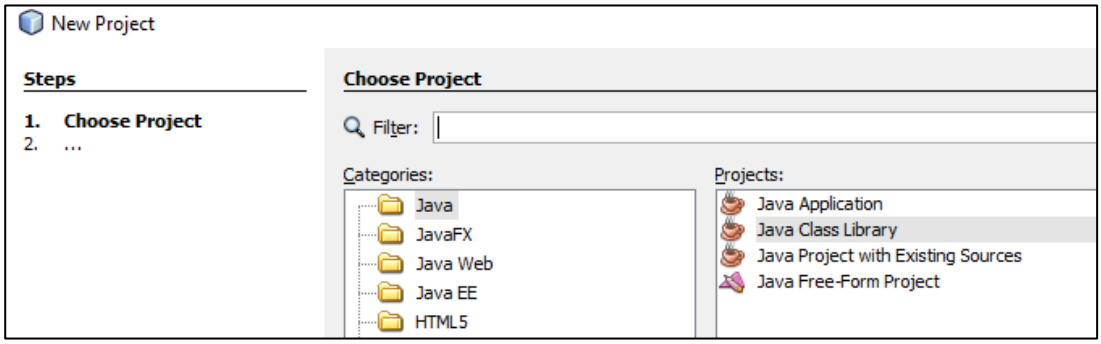
d) Luego se generarán las capas



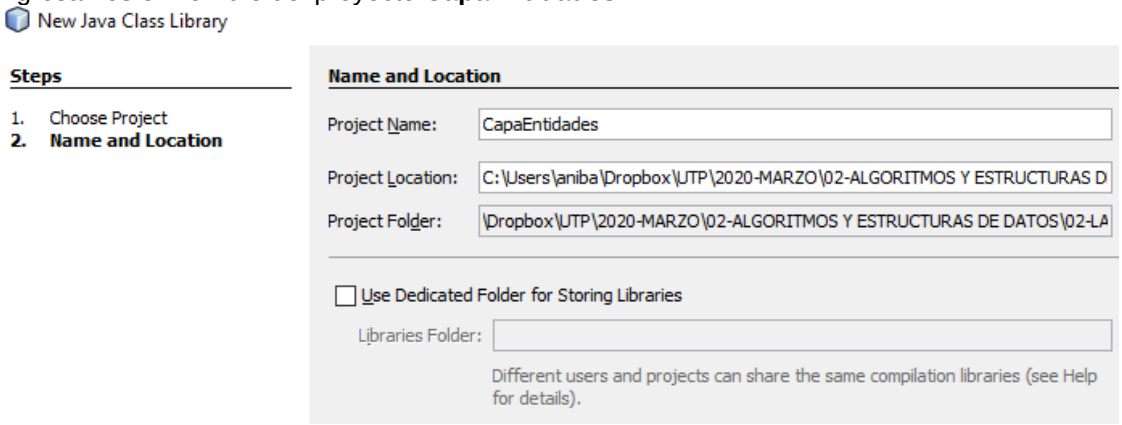
Ejemplo 02: Diseño de la Aplicación en Capas

1. Creación de la Capa de Entidades.

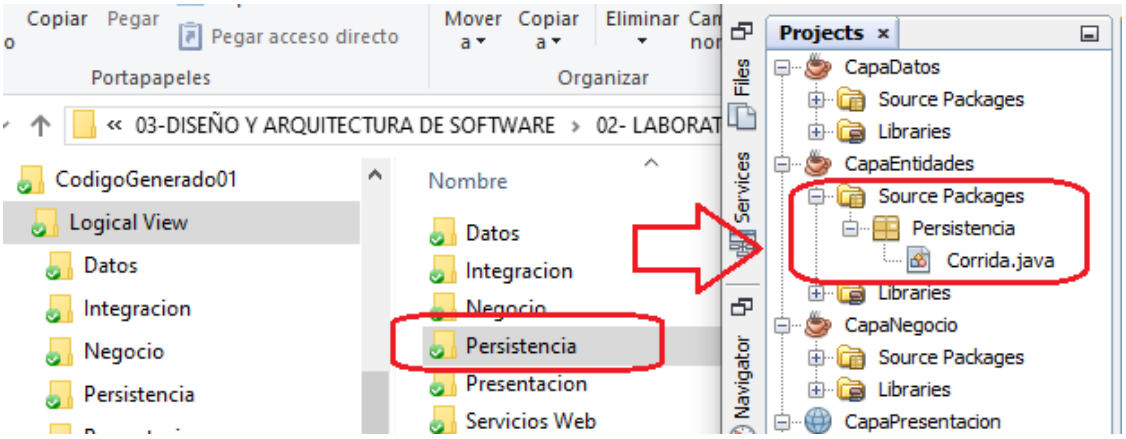
Ingresamos a NetBeans, y creamos un nuevo proyecto, hacemos Clic en el menú Archivo -> Nuevo Proyecto -> Java - > Java Class Library.



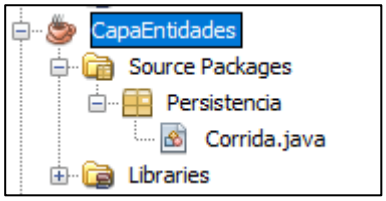
Ingresamos el Nombre del proyecto **CapaEntidades**



Abrimos el Explorador de Windows seleccionamos la carpeta donde se han generado los paquetes y clases **(Logical View).**
Vamos a arrastrar la carpeta **Persistencia** hacia el proyecto **CapaEntidades** y lo soltamos en **Source Packages.**



El proyecto **CapaEntidades** contendrá los siguientes elementos.



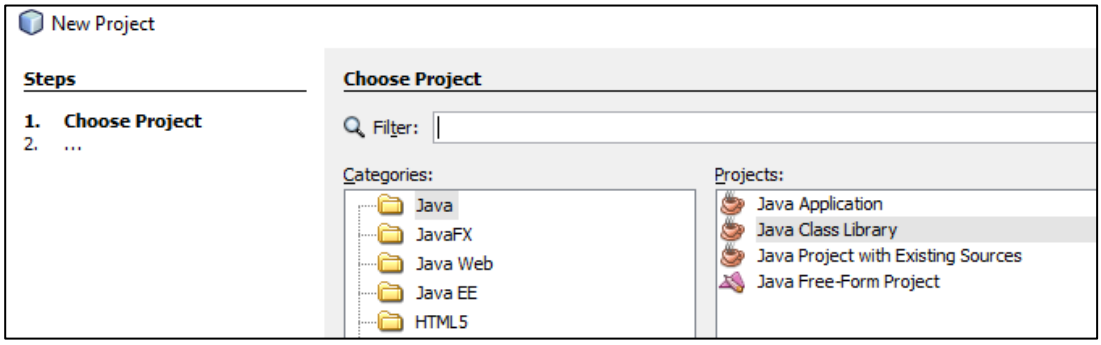
El código fuente de la clase **Corrida.java** es el siguiente:

```
1   package Persistencia;
2   import java.util.*;
3
4   public class Corrida {
5
6       public Corrida() {
7       }
8       public int ID;
9       public String Nombres;
10      public String Origen;
11      public String Destino;
12      public double Precio;
13
14      public int getID() {
15          return ID;
16      }
17      public void setID(int ID) {
18          this.ID = ID;
19      }
20      public String getNombres() {
21          return Nombres;
22      }
```

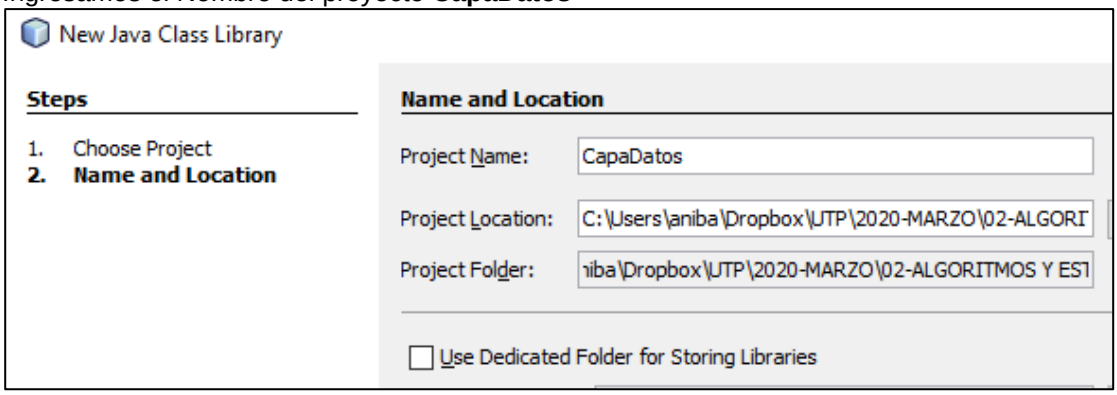
```
23      public void setNombres(String Nombres) {
24          this.Nombres = Nombres;
25      }
26      public String getOrigen() {
27          return Origen;
28      }
29      public void setOrigen(String Origen) {
30          this.Origen = Origen;
31      }
32      public String getDestino() {
33          return Destino;
34      }
35      public void setDestino(String Destino) {
36          this.Destino = Destino;
37      }
38      public double getPrecio() {
39          return Precio;
40      }
41      public void setPrecio(double Precio) {
42          this.Precio = Precio;
43      }
44  }
```

2. Creación de la Capa de Datos.

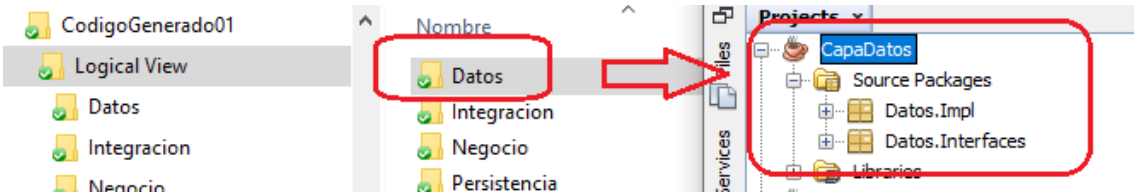
Ingresamos a NetBeans, y creamos un nuevo proyecto, hacemos Clic en el menú Archivo -> Nuevo Proyecto -> Java - > Java Class Library.



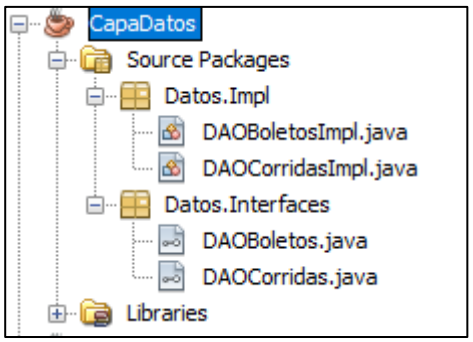
Ingresamos el Nombre del proyecto **CapaDatos**



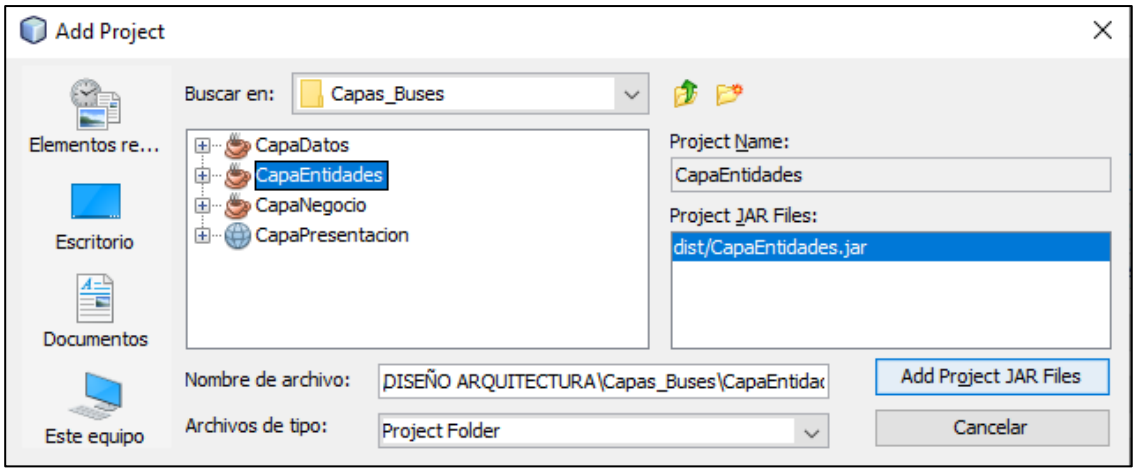
Abrimos el Explorador de Windows seleccionamos la carpeta donde se han generado los paquetes y clases **(Logical View).**
Vamos a arrastrar la carpeta **Datos** hacia el proyecto **CapaDatos** y lo soltamos en **Source Packages.**

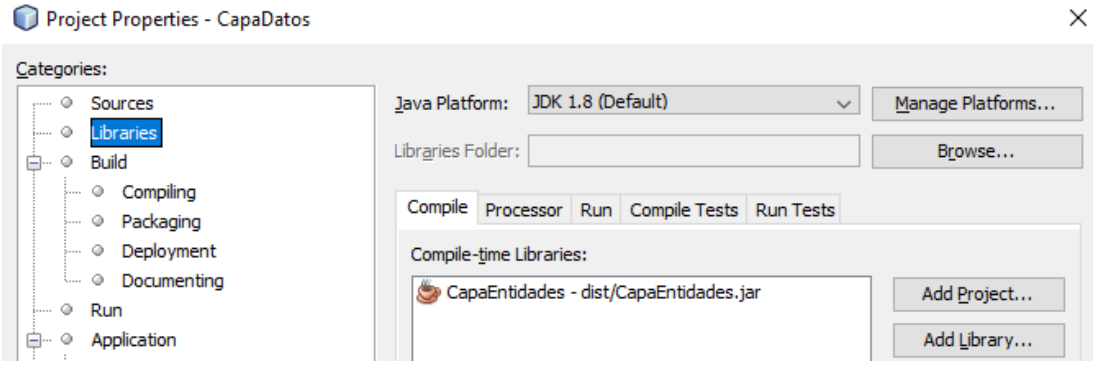


El proyecto **CapaDatos** contendrá los siguientes elementos.

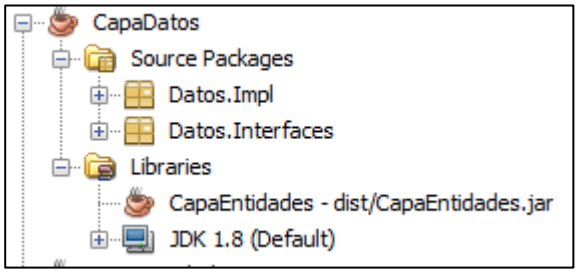


Vamos a crear la dependencia con el proyecto **CapaEntidades**.
Hacemos Clic derecho sobre el proyecto | **Properties** , en **Categories** seleccionamos **Libraries** y presionamos el botón **Add Project.**
Seleccionamos la **CapaEntidades** y presionamos el botón **Add Project JAR Files.**

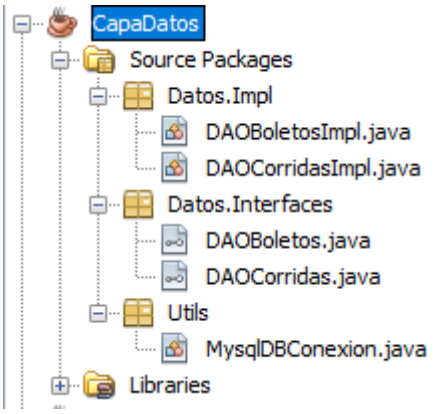




Luego en el proyecto **CapaDatos** en la carpeta **Libraries** se agregará la dependencia **CapaEntidades – dist/CapaEntidades.jar.**



Vamos a agregar el **paquete Utils** y la clase **MysqlDBConexion.java** Ingresamos a NetBeans, y creamos un nuevo proyecto, hacemos Clic en el menú Archivo -> Nuevo Proyecto -> Java - > Java Class Library.



El código fuente de la clase **MysqlDBConexion.java** es el siguiente:

```
1   package utils;
2
3   import java.sql.Connection;
4   import java.sql.DriverManager;
5
6   public class MysqlDBConexion
7   {
8       static{
9               try {
10                      Class.forName("com.mysql.jdbc.Driver");
11              }catch (ClassNotFoundException e) {
                        e.printStackTrace();
13              }
14      }
15      public static Connection getConexion()
16      {
17          Connection con=null;
18          try
19          {
20              con=DriverManager.getConnection("jdbc:mysql://localhost/buses","root","");
21          }
22          catch (Exception e)
23          {
                e.printStackTrace();
25          }
26          return con;
27      }
28  }
29
```

El código fuente de la interface **DAOCorridas.java** es el siguiente:

```
1   package Datos.Interfaces;
2   import java.util.*;
3   import Persistencia.Corrida;
4
    public interface DAOCorridas {
6       /**
7        * @param Criterio
8        */
        public List<Corrida> RecuperaCorridas(String Criterio);
10      /**
11       * @param CorridaNueva
12       */
        public boolean CreaCorrida(Corrida CorridaNueva);
14      /**
15       * @param Corrida
16       */
        public boolean ActualizaCorrida(Corrida Corrida);
18      /**
19       * @param Corrida
20       */
        public boolean EliminaCorrida(Corrida Corrida);
22  }
```

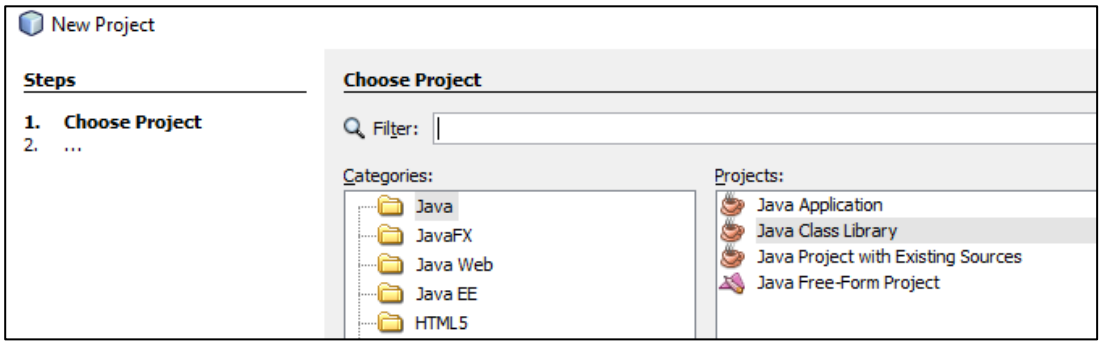
El código fuente de la interface **DAOCorridasImpl.java** es el siguiente:

```
1   package Datos.Impl;
2
3   import java.sql.Connection;
4   import java.sql.PreparedStatement;
5   import java.sql.ResultSet;
6   import java.util.ArrayList;
7   import java.util.List;
8
9   import Datos.Interfaces.DAOCorridas;
    import java.util.*;
11  import Persistencia.Corrida;
12  import Utils.*;
13
14  public class DAOCorridasImpl implements DAOCorridas
15  {   //CONSTRUCTOR
16      public DAOCorridasImpl() {
17      }
```

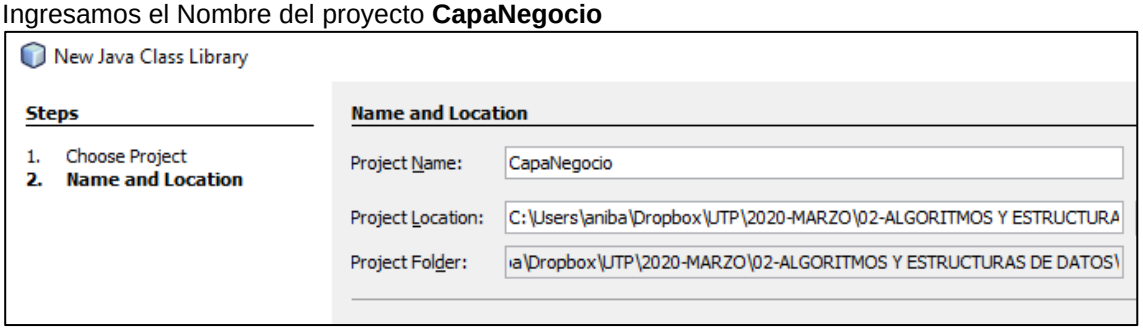
```
    public List<Corrida>  RecuperaCorridas(String Criterio)
    {
        List<Corrida> lista = new ArrayList<Corrida>();

        Connection conn= null;
            PreparedStatement pstm = null;
            ResultSet rs = null;
            try
            {   //LLAMADA A METODO para conectarse a la BD
                conn = MysqlDBConexion.getConexion();
                // CONSULTA CON FILTRO like
                String sql ="select * from corrida ";
                pstm = conn.prepareStatement(sql);
                //
                rs = pstm.executeQuery();
                Corrida obj = null;
                while(rs.next())
                {
                        obj = new Corrida();
                        obj.setID(rs.getInt("id"));
                        obj.setNombres(rs.getString("nombres"));
                        obj.setOrigen(rs.getString("origen"));
                        obj.setDestino(rs.getString("destino"));
                        obj.setPrecio(rs.getInt("precio"));
                        lista.add(obj);
                }
            }
            catch (Exception e)
            {
                    e.printStackTrace();
            }
            finally
            {       try
                    {       if(rs!= null) rs.close();
                            if(pstm!= null) pstm.close();
                            if(conn!= null) conn.close();
                    }
                    catch (Exception e2)
                    {
                    }
            }
        return lista;// TODO implement here
    }
    public boolean CreaCorrida(Corrida CorridaNueva) {
        return false;// TODO implement here
    }
    public boolean ActualizaCorrida(Corrida Corrida) {
        return false;// TODO implement here
    }
    public boolean EliminaCorrida(Corrida Corrida) {
        return false;// TODO implement here
    }
}
```

3. Creación de la Capa de Negocio.

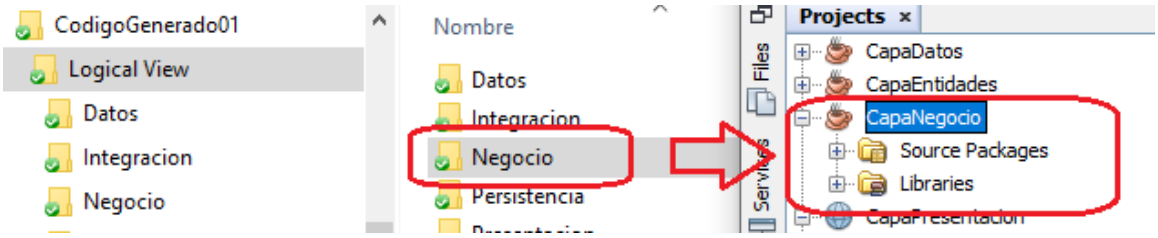
Ingresamos a NetBeans, y creamos un nuevo proyecto, hacemos Clic en el menú Archivo -> Nuevo Proyecto -> Java - > Java Class Library.



Ingresamos el Nombre del proyecto **CapaNegocio**



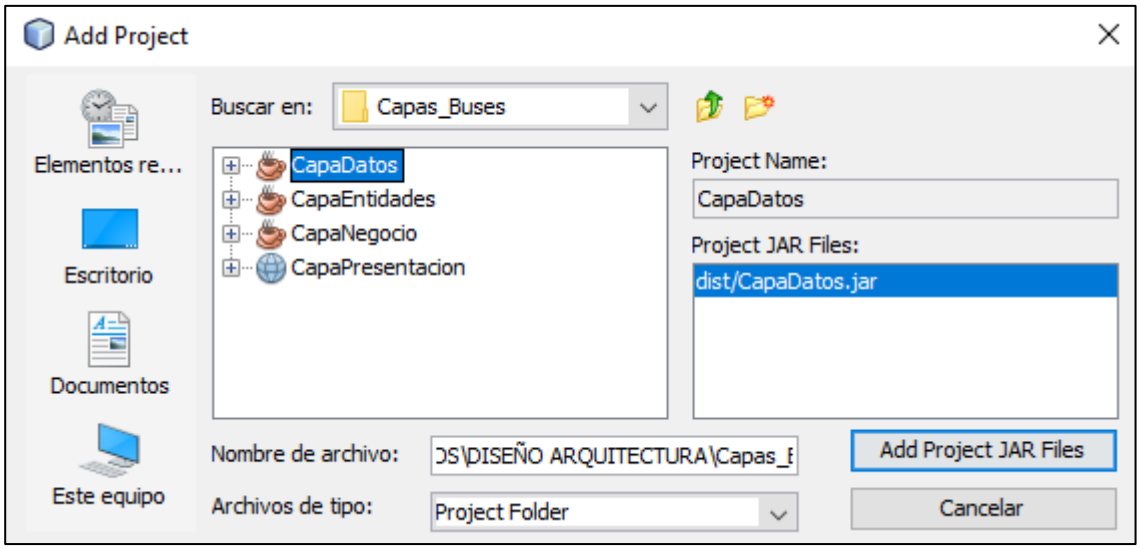
Abrimos el Explorador de Windows seleccionamos la carpeta donde se han generado los paquetes y clases **(Logical View).**
Vamos a arrastrar la carpeta **Datos** hacia el proyecto **CapaNegocio** y lo soltamos en **Source Packages.**

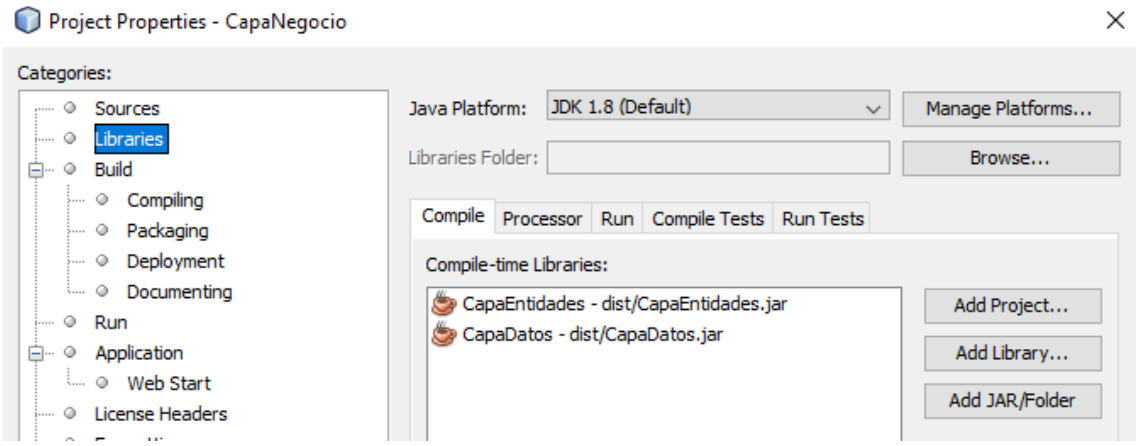


El proyecto **CapaNegocio** contendrá los siguientes elementos.

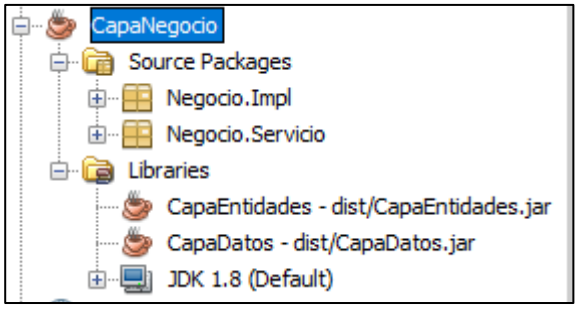
Vamos a crear la dependencia con el proyecto **CapaEntidades y CapaDatos**.
Hacemos Clic derecho sobre el proyecto | **Properties** , en **Categories** seleccionamos **Libraries** y presionamos el botón **Add Project.**

Seleccionamos la **CapaEntidades** y presionamos el botón **Add Project JAR Files.**
Seleccionamos la **CapaDatos** y presionamos el botón **Add Project JAR Files.**





Luego en el proyecto **CapaNegocio** en la carpeta **Libraries** se agregará la dependencia **CapaEntidades – dist/CapaEntidades.jar.** y **CapaDatos – dist/CapaDatos.jar**



El código fuente de la interface **ServicioConsultaCorrida.java** es el siguiente:

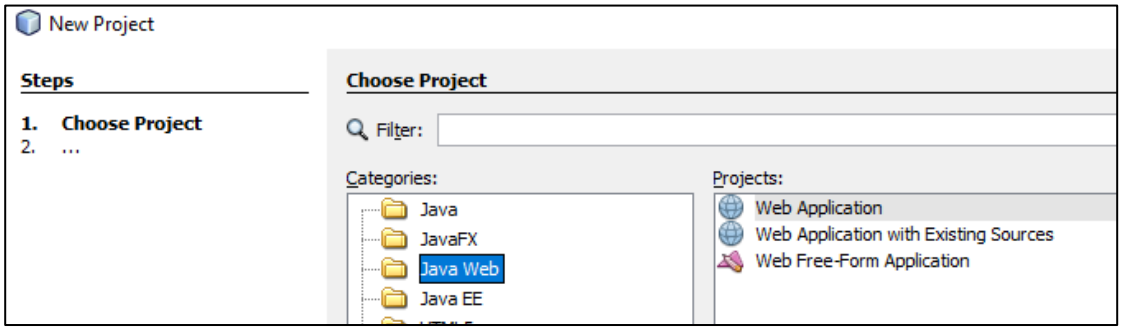
```java
package Negocio.Servicio;
import Persistencia.Corrida;
import java.util.*;


public interface ServicioConsultaCorridas {


    public List<Corrida> ConsultaCorridas(String criterios);

}
```

La clase **ConsultaCorridasImpl** llama a la clase de la **Capa de Datos CorridasDAO** el cual a travez de su método **RecuperaCorridas(),** devuelve una lista de objetos.
El código fuente de la clase **ConsultaCorridasImpl.java** es el siguiente:

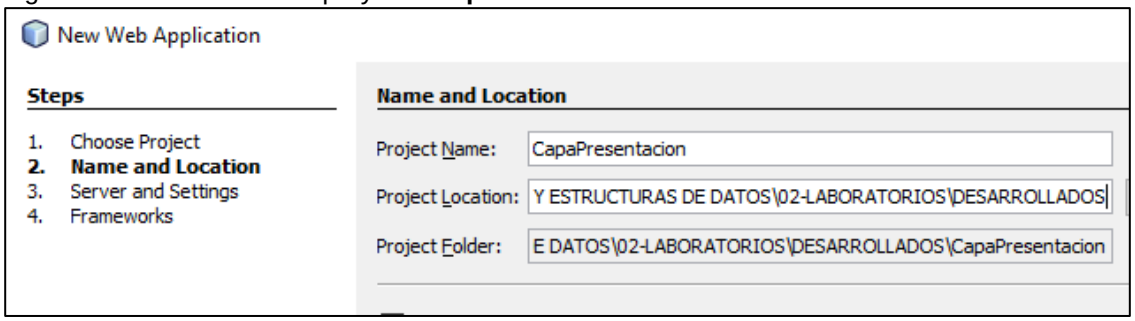
```java
package Negocio.Impl;

import java.util.*;
import Negocio.Servicio.ServicioConsultaCorridas;
//IMPORTA CAPA DATOS
import Datos.Interfaces.*;
import Datos.Impl.*;
//IMPORTA CAPA PERSISTENCIA
import Persistencia.Corrida;

public class ConsultaCorridasImpl implements ServicioConsultaCorridas {
    //CONSTRUCTOR
    public ConsultaCorridasImpl() {
    }

    public List<Corrida>  ConsultaCorridas(String criterios)
    {
        DAOCorridas CorridasDao = new DAOCorridasImpl();
        List<Corrida> lista = CorridasDao.RecuperaCorridas(criterios);
        return lista;// TODO implement here
    }
}
```

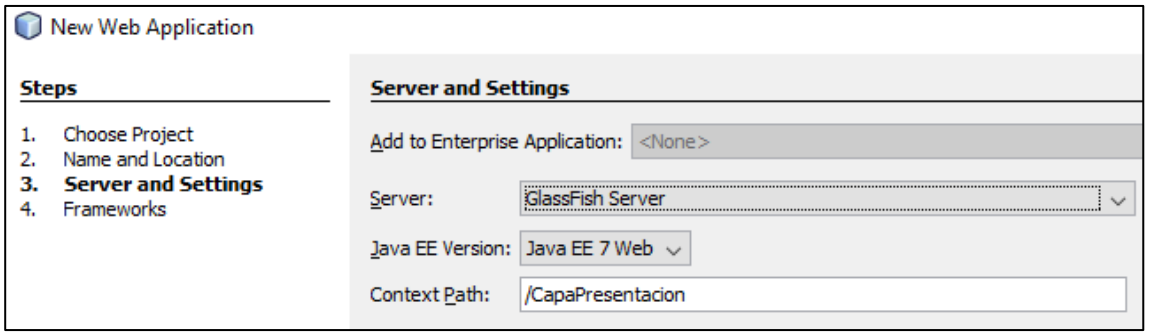
4. Creación de la Capa de Presentación.

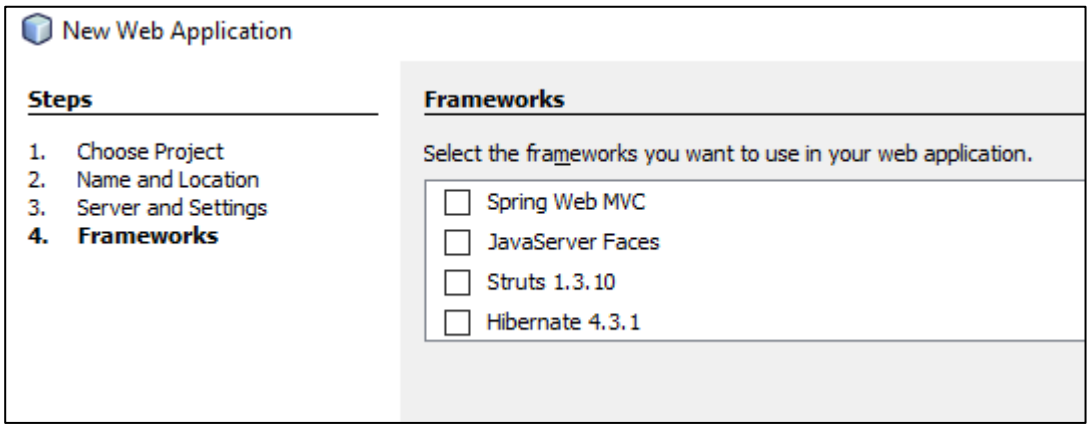
Ingresamos a NetBeans, y creamos un nuevo proyecto, hacemos Clic en el menú Archivo -> Nuevo Proyecto -> Java Web - > Web Application.



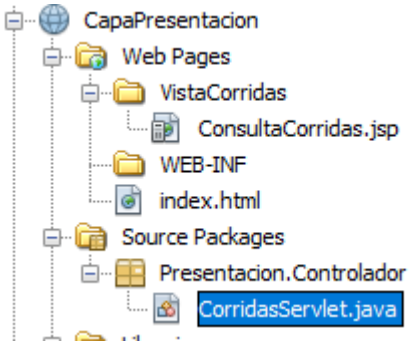
Ingresamos el Nombre del proyecto **CapaPresentacion**







Vamos a crear el **servlet CorridasServlet.java** el cual llama a la clase **ServicioConsultaCorridas** de la **Capa de Negocio.** El método **ConsultaCorridas()** devuelve una lista de objetos que se envía a la página JSP **consultaCorridas.jsp** para mostrar la lista.



El código fuente del servlet **CorridasServlet** es el siguiente:

.

```
1   ...5 lines
6   package Presentacion.Controlador;
7
8   import java.io.IOException;
9   import java.io.PrintWriter;
10  import javax.servlet.ServletException;
11  import javax.servlet.annotation.WebServlet;
12  import javax.servlet.http.HttpServlet;
13  import javax.servlet.http.HttpServletRequest;
14  import javax.servlet.http.HttpServletResponse;
15  //IMPORTA CAPA NEGOCIO
16  import Negocio.Servicio.*;
    import Negocio.Impl.CompraBoletosImpl.*;
18  import Negocio.Impl.ConsultaCorridasImpl;
19  //IMPORTA CAPA ENTIDADES
20  import Persistencia.*;
21  import java.util.List;
22
```

```
23  @WebServlet(name = "CorridasServlet", urlPatterns = {"/CorridasServlet"})
24  public class CorridasServlet extends HttpServlet {
25
26      /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
35      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
36              throws ServletException, IOException {...15 lines }
51
52      HttpServlet methods. Click on the + sign on the left to edit the code.
90
```

```
92      protected void service(HttpServletRequest request, HttpServletResponse response)
93              throws ServletException, IOException
94      {
95          //1 Se obtiene una lista
96          // de la base de datos
97          String prmCriterios;
98          prmCriterios = request.getParameter("criterios");
99          ServicioConsultaCorridas ConsultaService = new ConsultaCorridasImpl();
100         List<Corrida> lista = ConsultaService.ConsultaCorridas(prmCriterios);
101
102         //2 Se asigna el list en request con el alias "corridas"
103         //  El request es una memoria temporal
104         request.setAttribute("corridas", lista);
105
106         //3 Se reenvia el request al formulario
107         request.getRequestDispatcher("VistaCorridas/ConsultaCorridas.jsp").
108                 forward(request, response);
109     }
    }
```

Vamos a agregar la página JSP **consultaCorridas.jsp.** La página muestra una lista de objetos enviada por el servlet: **CorridasServlet**

```jsp
<%@page import="Persistencia.Corrida"%>
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Consulta de Corridas</title>
    </head>
<body>
    <h1>Listado de Corridas</h1>

    <ul>
        <li><a href="index.html">Volver</a></li>

    </ul>

    <table>
        <tr class="grilla_cabecera">
           <th>Id</th><th>Nombre</th><th>Origen</th><th>Destino</th><th>Precio</th>
        </tr>
        <%
            List<Corrida> a = (List<Corrida>)request.getAttribute("corridas");
            if(a != null)
            {
                for(Corrida aux :a)
                {
        %>
                    <tr class="grilla_campo">
                        <td><%= aux.getID() %></td>
                        <td><%= aux.getNombres()  %></td>
                        <td><%= aux.getOrigen() %></td>
                        <td><%= aux.getDestino() %></td>
                        <td><%= aux.getPrecio() %></td>
                    </tr>

        <%      }
            } %>

      </table>
</body>
</html>
```

7. ENTREGABLES

Luego de culminar el proyecto y hacer que funcione la página JSP, realice los siguientes entregables:

1. Crear en la Base de Datos **Transporte**, la tabla **Boletos** con datos.
2. Agregar las clases y métodos en las diferentes capas creadas, para que funcione la página **mantenimientoCorridas.jsp** *(implementar las operaciones de **Inserción**, **Actualización**, y **Eliminación**)*
3. Agregar las Clases y métodos en las diferentes capas creadas, para que funcione las páginas JSP **mantenimientoBoletos.jsp** *(implementar las operaciones de **Inserción**, **Actualización**, y*

Eliminación) y la página **consultasBoletos.jsp** *(que permita **Consultar boletos por Fechas y Consultar Boletos por Rutas**)*

8. FUENTES DE INFORMACIÓN COMPLEMENTARIA

- Cervantes, H & Velasco,P (2016). Arquitectura de Software, Conceptos y ciclo de Desarrollo. Mexico D.F. Cengage Learning Editores.
- Somerville, I.(2015). Ingeniería de Software. Madrid, España. Pearson. 7ma. Edición.
- Pressman, R.(2015). Ingeniería de Software, un enfoque práctico. Mexico DF. Mc Graw Hill. 7ma. Edición.