PASOS PA NO TROLEAR MI FINAL

BACK-END

- En el back-end vamos a tener los siguientes proyectos especificando cada una de sus referencias:
  - DBManager: *driver, commons-codec*
  - Model:
  - DA: *model, dbmanager*
  - BO: *model, DA*
  - RMI-Servidor (*servidor*): *dbmanger, model, da, bo, driver, commons-codec*
  - WS (*cliente)*: *model, servidorRMI, jakarta*
- Teniendo en cuenta que lo más probable es que tengas que renombrar tanto **paquetes** como **proyectos,** vamos a tener que tener en cuenta lo siguiente dentro de cada uno de ellos
- **En los datos conexion.txt fijarnos de colocar nuestras credenciales correspondientes**

Proyecto DBManager

- Dentro de *leerArchivoYCrearCadena,* en caso renombremos el paquete ***softplantilla.db***, modificamos

```
String archivoConfiguracion = "/pe/edu/pucp/softplantilla/config/" + ARCHIVO_CONFIGURACION;
```

> 💡 Este *config,* hace referencia a los proyectos donde necesitemos establecer la conexión. Especificamente el *test* y *ServidorRMI*

Proyecto Model

- Como solamente nos darán los scripts y .aspx, nos vamos a fijar en las tablas creadas dentro de los scripts. Reconocemos **clases y atributos de cada uno de ellos.**
- Definimos constructor (*por defecto y con parámetros*), getters y setters.
- Recordar que tenemos que utillizar ***implements Serializable*** para poder utilizar el RMI.

Proyecto DA

- Dentro de este proyecto vamos a tener tanto los paquetes ***DAO y DAOImp***

Paquete DAO

- Simplemente vamos a definir todos los métodos que nos soliciten

```
public interface PersonaDAO {
    Integer insertar(Persona persona);
    ArrayList<Persona> listarTodos();
}
```

Paquete DAOImp

- Vamos a tener 2 tipos de consultas sql. Aquellas que utilicen ***ejecutarProcedimiento*** (*insertar-modificar-eliminar*) y aquellas que utilizan ***ejecutarProcedimientoLectura*** (*select*)
- Para saber qué colocar, nos vamos a fijar en **los scripts que nos proporcionen.**
  - Por ejemplo para un insertar tenemos que pasar como parámetros al ***ejecutarProcedimiento*** el *nombreProcedimiento, arreglo objects de parámetros, parametroSalida*

```
@Override
    public Integer insertar(Persona persona) {
        int resultado = 0;

        Object[] parametros = new Object[3];
        parametros[0] = persona.getIdPersona();
        parametros[1] = persona.getNombre();
        parametros[2] = persona.getApellido();
        // Si tienes un enum
        // parametros[3] = persona.getTipo().ToString();

        persona.setIdPersona(dbManager.EjecutarProcedimiento("INSERTAR_PERSONA", parame
tros, true));

        resultado = persona.getIdPersona();

        return resultado;
    }
```

- Por ejemplo para un listarTodos tenemos que pasar como parámetros al ***ejecutarProcedimientoLectura*** el *nombreProcedimiento, arreglo objects de parámetros*

```
@Override
    public ArrayList<Persona> listarTodos() {
        ArrayList<Persona> personas = new ArrayList<>();

        rs = dbManager.EjecutarProcedimientoLectura("LISTAR_PERSONAS_TODOS", null);

        try {
            while(rs.next()) {
                Persona persona = new Persona();

                persona.setIdPersona(rs.getInt("idPersona"));
                persona.setNombre(rs.getString("nombre"));
                persona.setApellido(rs.getString("apellido"));

                personas.add(persona);
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
        finally{
            try {
                dbManager.cerrarConexion();
            } catch (SQLException ex) {
                System.out.println(ex.getMessage());
            }
        }

        return personas;
    }
```

Proyecto BO

- Vamos a definir métodos que hagan referencia a aquellos implementados en el *proyectoDA*.

- Como lo vamos a trabajar es pasar parámetro por parámetro, no una instancia de la clase. Con esos parámetros, vamos a instanciarlo dentro y lo pasaremos al *ejemploDA* correspondiente

```
public class PersonaBO {
    private PersonaDAO personaDAO;

    public PersonaBO() {
        personaDAO = new PersonaDAOImp();
    }

    public Integer insertar(String nombre, String apellido) {
        Persona persona = new Persona(nombre, apellido);
        return personaDAO.insertar(persona);
    }

    public ArrayList<Persona> listarTodos() {
        return personaDAO.listarTodos();
    }
}
```

Proyecto ServidorRMI

- Dentro de este proyecto vamos a tener tanto los paquetes ***Interfaces e InterfacesImp***

Paquete Interfaces

- Vamos a definir el mismo nombre de métodos que nuestro BO pero a un nivel de abstracción superior. Manejamos con un ***RemoteException***

```
public interface PersonaBO extends Remote {
    Integer insertar(String nombre, String apellido) throws RemoteException;

    ArrayList<Persona> listarTodos() throws RemoteException;
}
```

Paquete InterfacesImp

- Como estamos creando una capa de abstracción por encima de nuestro BO original, vamos a tener como atributo dicho BO inicial. Aquí solamente lo instanciamos y llamamos a sus métodos.

```
public class PersonaBOImp extends UnicastRemoteObject implements PersonaBO {
    private pe.edu.pucp.softplantilla.bo.PersonaBO personaBO;

    public PersonaBOImp(Integer puerto) throws RemoteException {
        super(puerto);
        personaBO = new pe.edu.pucp.softplantilla.bo.PersonaBO();
    }

    @Override
    public Integer insertar(String nombre, String apellido) throws RemoteException {
        return personaBO.insertar(nombre, apellido);
    }

    @Override
    public ArrayList<Persona> listarTodos() throws RemoteException {
        return personaBO.listarTodos();
```

```
    }
}
```

💡 En este proyecto se establece una conexión, es por ello que tenemos que tener cuidado si renombramos la carpeta *.config.* Debe coincidir con lo planteado el en *dbManager*

Paquete servidor

- Dentro de *leerArchivoYCrearCadena,* en caso renombremos el paquete ***softplantilla.db***, modificamos

```
String archivoConfiguracion = "/pe/edu/pucp/softplantilla/config/" + ARCHIVO_CONFIGURACION;
```

```
public class ServidorRMI {
    private static final String ARCHIVO_CONFIGURACION = "datosConexionRMI.txt";
    private static String IP;
    private static Integer puerto;

    public static void main(String[] args) {
        leerArchivoYCrearCadena();
        System.out.println("Datos del servidor RMI: IP = " + IP + " - Puerto = " + puerto);

        try {
            System.setProperty("java.rmi.server.hostname", IP);

            LocateRegistry.createRegistry(puerto);

            // Agregar los BOs necesarios
            PersonaBO personaBO = new PersonaBOImp(puerto);

            // Agregar los nombres de servicios necesarios
            String nombreServicio = retornarNombreDelServicio("personaBO");
            Naming.rebind(nombreServicio, personaBO);

            System.out.println("Servidor RMI registrado correctamente...");
        } catch (RemoteException | MalformedURLException ex) {
            System.out.println(ex.getMessage());
        }
    }

    public static String retornarNombreDelServicio(String servicio) {
        return "//" + IP + ":" + puerto + "/" + servicio;
    }

    public static void leerArchivoYCrearCadena() {
        Map<String, String> config = new HashMap<>();
        String archivoConfiguracion = "/pe/edu/pucp/softplantilla/config/" + ARCHIVO_CONFIGUR
ACION;

        try {
            // Obtener el archivo como InputStream desde el JAR
            InputStream inputStream = ServidorRMI.class.getResourceAsStream(archivoConfigurac
ion);

            // Leer el contenido del archivo
            try (BufferedReader br = new BufferedReader(new InputStreamReader(inputStream)))
```

```
{
                String linea;
                while ((linea = br.readLine()) != null) {
                    String[] partes = linea.split("=");
                    if (partes.length == 2) {
                        config.put(partes[0].trim(), partes[1].trim());
                    }
                }
            }

            // Asignar los valores al final
            IP = config.get("IP");
            puerto = Integer.valueOf(config.get("puerto"));

        } catch (IOException | NumberFormatException e) {
            System.out.println("Error leyendo archivo de conexion: " + e.getMessage());
        }
    }
}
```

Proyecto WebService

- Vamos a tener en cuenta el **archivo disco.** Aquí debemos hacer referencia a tantos servicios como definamos o nos soliciten. Asimismo en caso nos pidan renombrar, hacer que coincida la referencia: *ref="http://localhost:8080/SoftPlantillaWS/PersonaWS?WSDL"*

```
<discovery xmlns="http://schemas.xmlsoap.org/disco/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
    <contractRef xmlns="http://schemas.xmlsoap.org/disco/scl/" ref="http://localhost:8080/SoftPlantillaWS/PersonaWS?WSDL"/>
</discovery>
```

- En caso renombremos la carpeta de ***softplantilla.services,*** vamos a tener que hacer que coincida para cada uno de los servicios web.
- El *"BO"* que estamos referenciando en el web service es el **objeto remoto de nuestro BO original.** Esto porque hemos establecido una mayor abstracción a nivel de *back*

```
@WebService(serviceName = "PersonaWS", targetNamespace = "softplantilla.services")
public class PersonaWS {
    private static final String ARCHIVO_CONFIGURACION = "datosConexionRMI.txt";
    private static String IP;
    private static Integer puerto;

    private PersonaBO personaBO;

    public PersonaWS() {
        ServidorRMI.leerArchivoYCrearCadena();
        String nombreServicio = ServidorRMI.retornarNombreDelServicio("personaBO");
        try {
            this.personaBO = (PersonaBO) Naming.lookup(nombreServicio);
        } catch (NotBoundException | MalformedURLException | RemoteException ex) {
            System.out.println(ex.getMessage());
            this.personaBO = null;
```

```
        }
    }

    @WebMethod(operationName = "insertarPersona")
    public Integer insertarPersona(String nombre, String apellido) {
        Integer resultado = -1;

        try {
            resultado = personaBO.insertar(nombre, apellido);
        } catch (RemoteException ex) {
            System.out.println(ex.getMessage());
        }

        return resultado;
    }

    @WebMethod(operationName = "listarPersonasTodos")
    public ArrayList<Persona> listarPersonasTodos() {
        ArrayList<Persona> resultado = new ArrayList<>();

        try {
            resultado = personaBO.listarTodos();
        } catch (RemoteException ex) {
            System.out.println(ex.getMessage());
        }

        return resultado;
    }
}
```

💡 Recordar que tenemos que ejecutar **primero** el servidorRMI y posteriormente el **WebService**

FRONT-END

SoftPlantillaBaseBO

- Creamos la clase *BaseBO* y colocamos **public** en vez de **internal.**
- Damos click derecho-Agregar referencia de servicio.
- Pegamos la ruta del archivo disco

```
namespace SoftPlantillaBaseBO
{
    public class BaseBO
    {
        // Todas las referencias de servicios
        private PersonaWSClient personaWS;

        // Constructor
        public BaseBO()
        {
            // Agregamos todas las referencias de servicios
            personaWS = new PersonaWSClient();
        }
```

```
        public PersonaWSClient PersonaWS { get => personaWS; set => personaWS = value; }
    }
}
```

SoftPlantillaBO

- Agregamos la referencia al proyecto ***SoftPlantillaBaseBO***
- Creamos **una claseBO por cada servicio que traigamos de Java.** De la misma manera, colocamos **public** en vez de **internal.**
- **Para ahorrar tiempo, podemos copiar lo que hemos definido en las clases BO de Java al proyecto de C#.**

```
public class PersonaBO : BaseBO
{
    public int insertar(string nombre, string apellido)
    {
        return PersonaWS.insertarPersona(nombre, apellido);
    }

    public BindingList<persona> listarTodos()
    {
        persona[] personas = PersonaWS.listarPersonasTodos();
        if (personas == null)
        {
            return null;
        } else
        {
            return new BindingList<persona>(personas);
        }
    }
}
```

SoftPlantillaWS

- Agregamos la referencia a los 2 proyectos anteriores: *SoftPlantillaBaseBO y SoftPlantillaBO*
- Luego, copiamos la información del **app.config** del proyecto *SoftPlantillaBaseBO* al archivo **web.config**

```
<configuration>
    <system.web>
        <compilation debug="true" targetFramework="4.8.1" />
        <httpRuntime targetFramework="4.8.1" />
    </system.web>

    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="PersonaWSPortBinding" receiveTimeout="00:1:00"
                sendTimeout="00:1:00" maxBufferSize="2147483647"
                maxReceivedMessageSize="2147483647" />
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://localhost:8080/SoftPlantillaWS/PersonaWS"
                binding="basicHttpBinding" bindingConfiguration="PersonaWSPortBinding"
                contract="ServicioWS.PersonaWS" name="PersonaWSPort" />
        </client>
```

```
    </system.serviceModel>
</configuration>
```

💡 Para soportar las imágenes que queramos insertar, le agregamos las líneas de **rosado**

- Uso del ***ddl***

```
ddlNombre.DataSource = nombreDelBindingList/Función que me liste;
ddlNombre.DataTextField = "atributoMostrar";
ddlNombre.DataValueField = "atributoSeleccionadoAlClickear";
ddlNombre.DataBind();
```

💡 Generalmente la carga de datos se coloca dentro del *page Load()*

- Uso del ***modal***

```
// Modal
function showModalPlantilla() {
    modalPlantilla = new bootstrap.Modal(document.getElementById('busca la variable id para s
aber que colocar aca'));
    modalPlantilla.show();
}

// Como ejecutarlo
// Opcion 1
string script = "nombreDelaFunción();";
ScriptManager.RegisterStartupScript(this, GetType(), "nombreDelaFunción", script, true);

// Opcion 2
string script = "window.onload = function() { nombreDelaFunción() };";
ClientScript.RegisterStartupScript(GetType(), "", script, true);
```

- Uso del ***eval***

```
// boton para eliminar
<asp:LinkButton runat="server" Text="<i class='fa-solid fa-trash'></i>" CssClass="btn btn-dan
ger" OnClick="btnEliminar_Click" CommandArgument='<%# Eval("idIngrediente") %>'/>
```

```
// aspx.cs
protected void btnEliminar_Click(object sender, EventArgs e)
{
    int idIngrediente = Int32.Parse(((LinkButton)sender).CommandArgument);
    ingredientesPlato = (BindingList<ingrediente>)ViewState["ingredientesPlato"];
    ingrediente ingrediente = ingredientesPlato.SingleOrDefault(x => x.idIngrediente == idIng
rediente);
    ingredientesPlato.Remove(ingrediente);
    ViewState["ingredientesPlato"] = ingredientesPlato;
    dgvIngredientes.DataSource = ingredientesPlato;
    dgvIngredientes.DataBind();
}
```

- Uso del ***On Row DataBound***

```
<div class="row">
<asp:GridView ID="gvAdministradores" runat="server" AllowPaging="true" PageSize="5" OnRowData
Bound="gvAdministradores_RowDataBound" OnPageIndexChanging="gvAdministradores_PageIndexChangi
ng" AutoGenerateColumns="false" CssClass="table table-hover table-responsive table-striped" S
howHeaderWhenEmpty="true">
            <Columns>
                <asp:BoundField HeaderText="ID" />
                <asp:BoundField HeaderText="DNI" />
                <asp:BoundField HeaderText="Nombre completo" />
                <asp:BoundField HeaderText="Correo" />
                <asp:BoundField HeaderText="Fecha de creación" />
                <asp:TemplateField>
                    <ItemTemplate>
                        <div style="display: inline-flex; align-items: center;">
                            <asp:LinkButton ID="lbModificar" runat="server" Text="<i class
='fa-solid fa-edit ps-2'></i>" CommandArgument='<%# Eval("idUsuario") %>' OnClick="lbModifica
r_Click" />
                            <asp:LinkButton ID="lbEliminar" runat="server" Text="<i class
='fa-solid fa-trash ps-2'></i>" CommandArgument='<%# Eval("idUsuario") %>' OnClick="lbElimina
r_Click" />
                        </div>
                    </ItemTemplate>
                </asp:TemplateField>
            </Columns>
</asp:GridView>
</div>
```

```
protected void gvAdministradores_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        e.Row.Cells[0].Text = DataBinder.Eval(e.Row.DataItem, "idUsuario").ToString();
        e.Row.Cells[1].Text = DataBinder.Eval(e.Row.DataItem, "dni").ToString();
        e.Row.Cells[2].Text = DataBinder.Eval(e.Row.DataItem, "nombres").ToString() + "   " +
DataBinder.Eval(e.Row.DataItem, "apellidos").ToString();
        e.Row.Cells[3].Text = DataBinder.Eval(e.Row.DataItem, "correo").ToString();
        e.Row.Cells[4].Text = DateTime.Parse(DataBinder.Eval(e.Row.DataItem, "fechaCreacio
n").ToString()).ToString("dd-MM-yyyy");
    }
}
```

- Uso del ***on page Index***

```
protected void gvAdministradores_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    gvAdministradores.PageIndex = e.NewPageIndex;
    gvAdministradores.DataBind();
}
```

```
listaAdministradores = administradorBO.listarTodos();
gvAdministradores.DataSource = listaAdministradores;
gvAdministradores.DataBind();
```

💡 Generalmente la carga de datos se coloca dentro del *page Load()*

- Uso de la función ***cargarFoto()***

```
protected void Cargar_Foto(object sender, EventArgs e)
{
  if (IsPostBack && fileUploadFotoPlato.PostedFile != null && fileUploadFotoPlato.HasFile)
  {
      string extension = System.IO.Path.GetExtension(fileUploadFotoPlato.FileName);
      if (extension.ToLower() == ".jpg"  extension.ToLower() == ".jpeg"  extension.ToLower()
== ".png" || extension.ToLower() == ".gif")
      {
          string filename = Guid.NewGuid().ToString() + extension;
          string filePath = Server.MapPath("~/Uploads/") + filename;
          fileUploadFotoPlato.SaveAs(Server.MapPath("~/Uploads/") + filename);
          imgFotoPlato.ImageUrl = "~/Uploads/" + filename;
          imgFotoPlato.Visible = true;
          FileStream fs = new FileStream(filePath, FileMode.Open, FileAccess.Read);
          BinaryReader br = new BinaryReader(fs);
          Session["foto"] = br.ReadBytes((int)fs.Length);
          fs.Close();
      }
      else
      {
          Response.Write("Por favor, selecciona un archivo de imagen válido.");
      }
  }
}
```

💡 Si nos damos cuenta, la función *cargarFoto()* almacena las imágenes que seleccionemos dentro de una carpeta **Uploads.** Por eso, tenemos que crear una.

- Uso del ***casteo de enum***

```
// Castear Enum
productora.TipoEspecializacion = (TipoEspecializacion)Enum.Parse(typeof(TipoEspecializacion),
dr.GetString("tipo_especializacion"));
```

💡 Cuando traemos del java al C# una clase que tiene **ENUM** se crea un “atributo” llamado *nombreSpecified.* Al momento que queramos *insertar, modificar* necesitamos darle a este “campo” *nombreSpecified*=true;