



Relaciones

Sesión Conceptual 2

{desafío}
latam_







Inicio

{desafío}
latam_



15 minutos

- Estructurar vista bootstrap en un proyecto jsp.
- Añadir funcionalidad a login.jsp para login.
- Generar un Servlet procesaLogin.
- Entender el contexto de relaciones.
- Entender el concepto de pool de conexiones.

Objetivo



Desarrollo

{desafío}
latam_



150 minutos

/* Pool de conexiones */

Concepto de pool de conexiones

- Los usuarios no esperan a que otro usuario libere la conexión.
- Permite que la aplicación tenga un funcionamiento constante y disminuyendo la posibilidad de caídas.



Funcionamiento

- Servidor de aplicaciones que tiene N conexiones previamente creadas.
- El servidor web solo toma las que están en espera y las disponibiliza.
- Cuando la aplicación termina de usar la conexión, no es cerrada, es devuelta al pool de conexiones para ser usada por otra petición.

Configuración

Para utilizar un pool de conexiones, utilizaremos el proyecto anterior.

```
20 protected Connection generaConexion() {  
21     String usr = "sys as sysdba";  
22     String pwd = "admin";  
23     String driver = "oracle.jdbc.driver.OracleDriver";  
24     String url = "jdbc:oracle:thin:@//localhost:1521/desafio_ejemplo01";  
25  
26     try {  
27         Class.forName(driver);  
28         conn = DriverManager.getConnection(url,usr,pwd);  
29     } catch (Exception ex) {  
30         ex.printStackTrace();  
31     }  
32     return conn;  
33 }
```

Generar conexion.

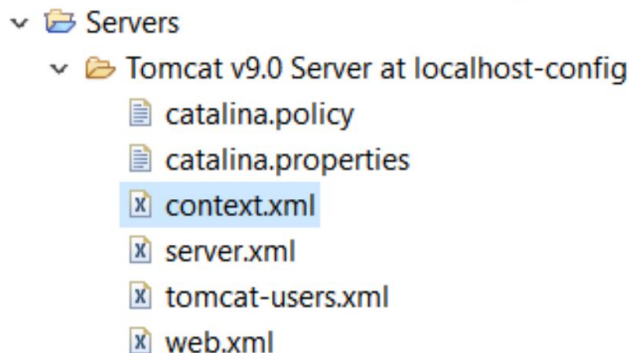
Generar pool de conexiones.

```
35= protected Connection generaPoolConexion() {  
36     Context initContext;  
37     try {  
38         initContext = new InitialContext();  
39         DataSource ds = (DataSource) initContext.lookup("java:/comp/env/jdbc/ConexionOracle");  
40         try {  
41             conn = ds.getConnection();  
42         } catch (SQLException e) {  
43             e.printStackTrace();  
44         }  
45     } catch (NamingException e) {  
46         e.printStackTrace();  
47     }  
48     return conn;  
49 }
```

¿Qué es un lookup?

Se hace una búsqueda del recurso en la ruta dispuesta como parámetro.

Archivo context.xml



Archivo de configuración propio del servidor tomcat.
Se definen las credenciales y datos de conexión a la base de datos.

```
<Resource name="jdbc/ConexionOracle" auth="Container" type="javax.sql.DataSource"  
    maxActive="20" maxIdle="10" maxWait="5000"  
    username="sys as sysdba" password="admin" driverClassName="oracle.jdbc.driver.OracleDriver"  
    url="jdbc:oracle:thin:@//localhost:1521/desafio_ejemplo01"/>
```

Con esas líneas lo que se está configurando es un DataSource.

1. **name:** Nombre del Datasource o fuente de datos.
2. **type:** tipo de Datasource en este caso Oracle.
3. **url:** Dirección del servidor y la base de datos.
4. **driverClassName:** Tipo de driver.
5. **username:** Usuario de conexión.
6. **password:** Clave de conexión.

Obteniendo el DataSource

```
35 protected Connection generaPoolConexion() {  
36     Context initContext;  
37     try {  
38         initContext = new InitialContext();  
39         DataSource ds = (DataSource) initContext.lookup("java:/comp/env/jdbc/ConexionOracle");  
40         try {  
41             conn = ds.getConnection();  
42         } catch (SQLException e) {  
43             e.printStackTrace();  
44         }  
45     } catch (NamingException e) {  
46         e.printStackTrace();  
47     }  
48     return conn;  
49 }
```

Archivo web.xml

- ▼ ControlaDeptosEmpresa
 - > JAX-WS Web Services
 - ▼ Java Resources
 - > src
 - > Libraries
 - > JavaScript Resources
 - > Referenced Libraries
 - > build
 - ▼ WebContent
 - > META-INF
 - ▼ WEB-INF
 - > lib
 - web.xml
 - index.jsp
 - resultado.jsp
- ▼ Servers
 - ▼ Tomcat v9.0 Server at localhost-config
 - catalina.policy
 - catalina.properties
 - context.xml
 - server.xml
 - tomcat-users.xml
 - web.xml

Agregando las referencias al datasource.

```
1 <resource-ref>
2     <description>Pool conexiones</description>
3     <res-ref-name>jdbc/ConexionOracle</res-ref-name>
4     <res-type>javax.sql.DataSource</res-type>
5     <res-auth>Container</res-auth>
6 </resource-ref>
```

Utilizar el DataSource.

```
12 public class DepartamentoDaoImp extends AdministradorConexion implements DepartamentoDao{
13
14     public DepartamentoDaoImp() {
15         Connection conn = generaPoolConexion();
16     }
17 }
```

/* Relaciones */

Un pequeño contexto de las relaciones

Relación empleado - departamento.



Es un contexto ideal para representar una relación simple, tiene el potencial de ser mejorado para aplicar problemáticas más complejas.

Relación = Cardinalidad

Relación 1:1

- Un registro de la tabla A está asociado solo con un registro de la tabla B.
- Se hace mediante las claves primarias (PK) y claves foráneas (FK).

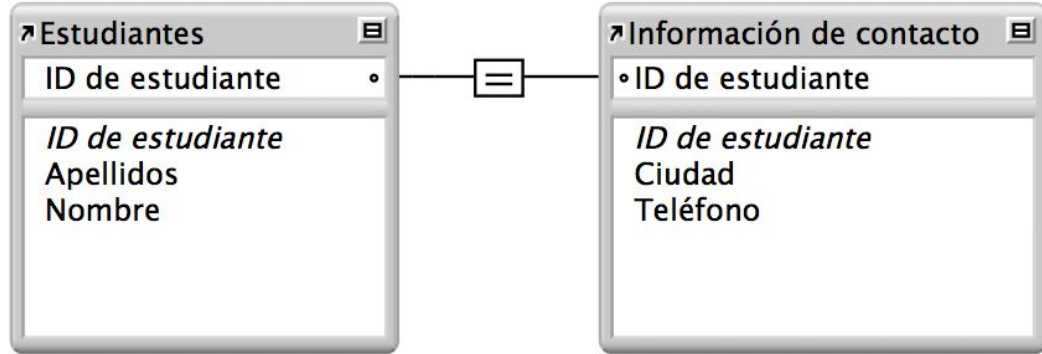


Tabla Alumnos

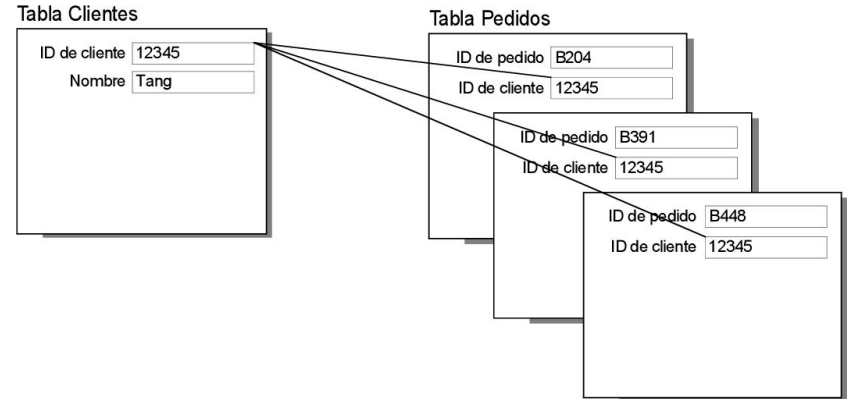
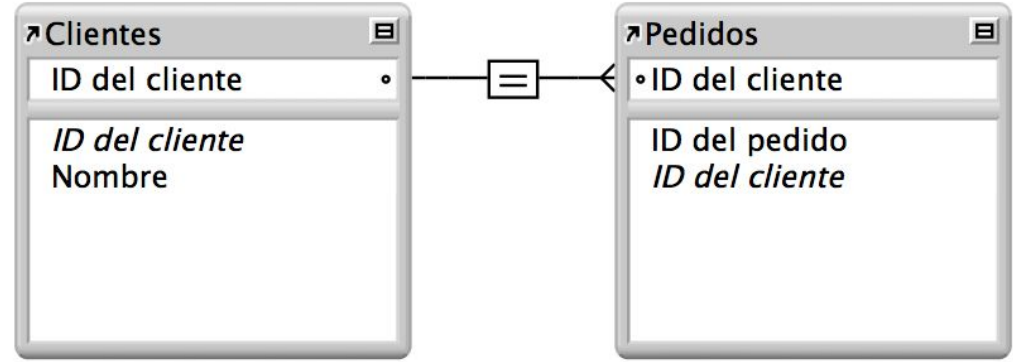
ID de estudiante	12345
Apellidos	Tang
Nombre	Sophie

Tabla Información de contacto

ID de estudiante	12345
Ciudad	Nueva York
Teléfono	408-555-3456

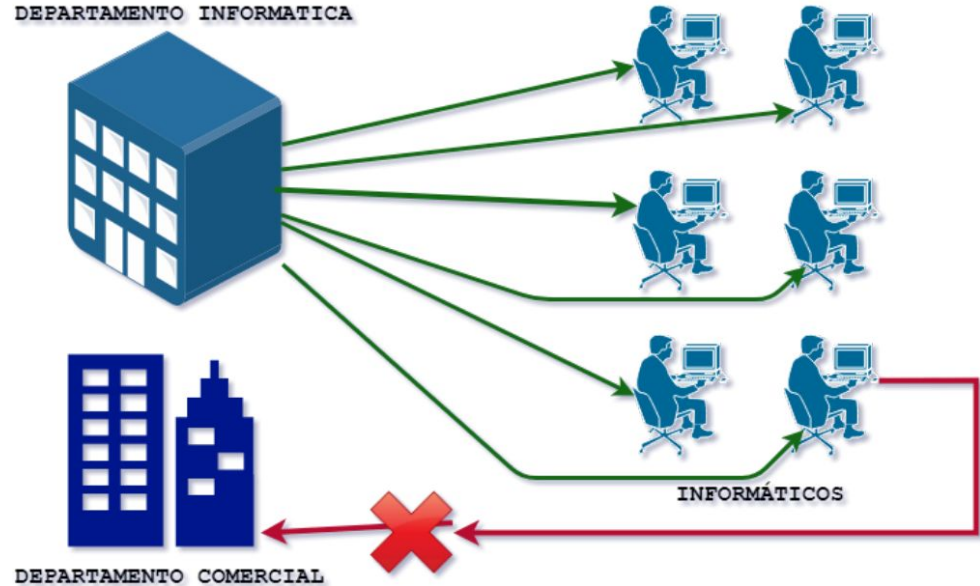
Relación 1:N

- Un registro de la tabla padre puede estar asociado a 1 o muchos registros de la tabla hija.
- Una clave primaria de la tabla cabecera pasa a ser una clave foránea en la tabla hija.



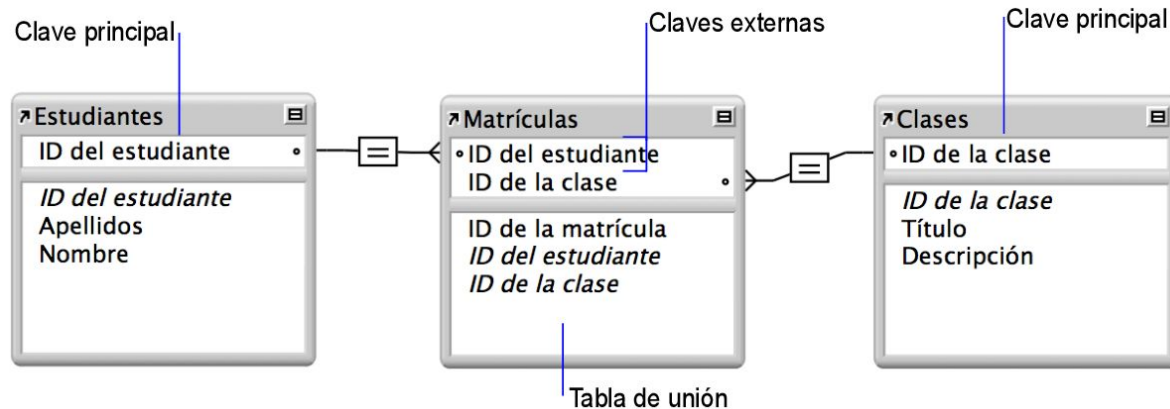
Ejemplo relación 1 a N

No puede existir en la tabla empleados más de una vez ya que tiene una clave única.



Relación N:N

- Un registro de la tabla A puede estar asociado a 1 o muchos registros de la tabla B y viceversa.
- Crear una tabla intermedia que almacena la clave primaria de la tabla A y la clave primaria de la tabla B.

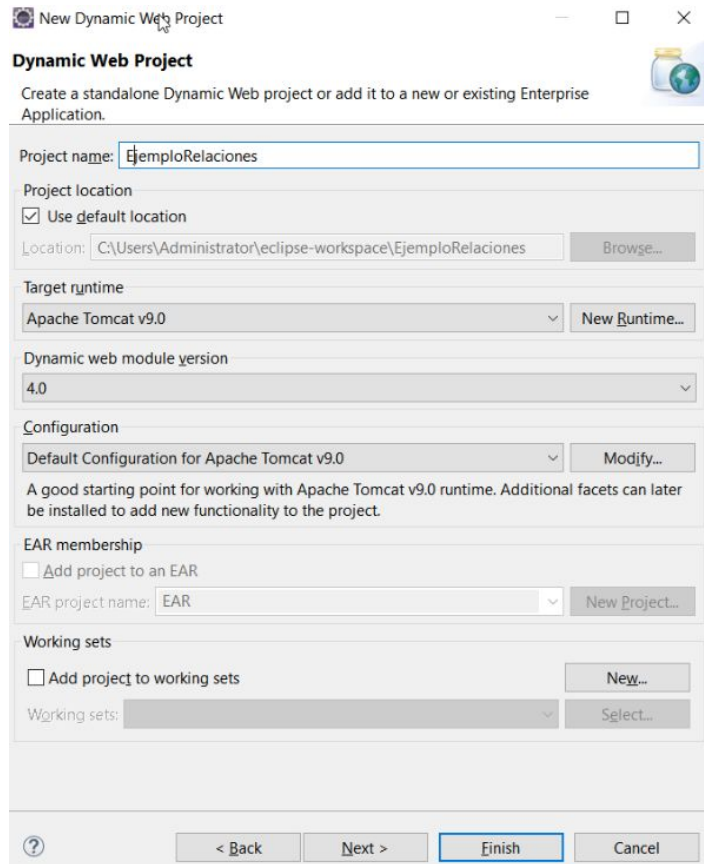


Trabajar con una relación 1-N en una aplicación JEE

Se implementará un pequeño proyecto con servlet, jsp y patrón dao para trabajar con una relación 1 a muchos.

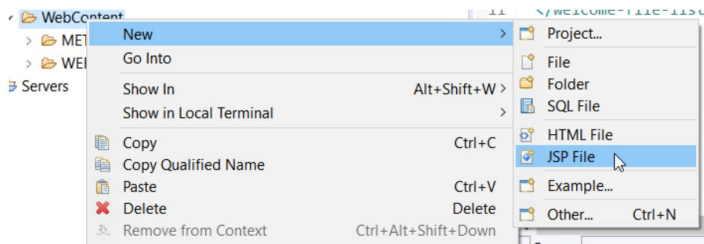
Paso 1.

Creando el proyecto.



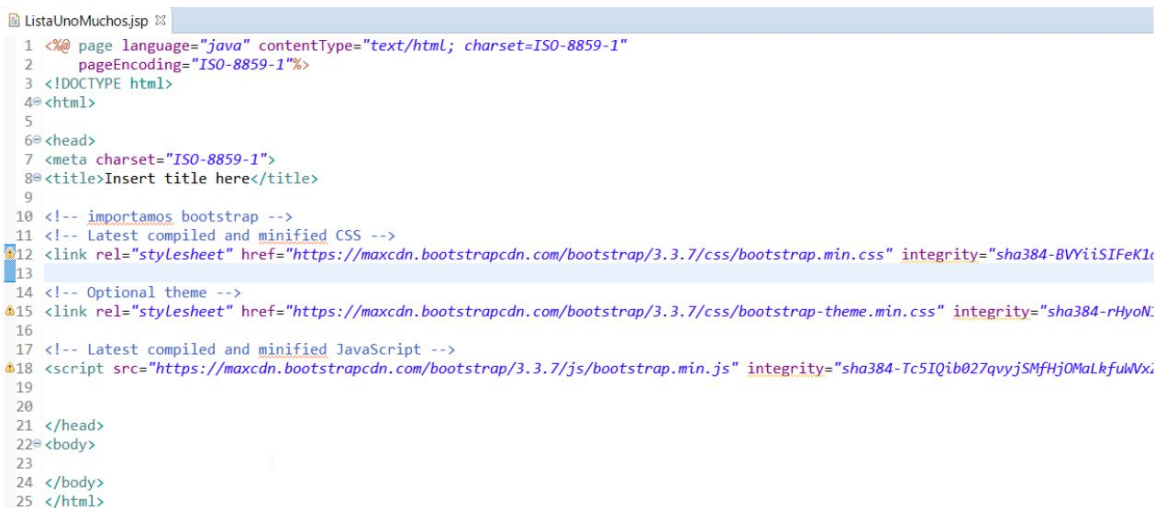
Paso 2

Crear un JSP.



Paso 3

Importando Bootstrap.



Paso 4

Creando la vista.

```
23<body>
24
25<div class="jumbotron text-center" style="margin-bottom: 0">
26  <h1>Ejemplo relaciones con JSP</h1>
27  <p>Para el curso de JEE</p>
28</div>
29
30<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
31  <a class="navbar-brand" href="#">Navegacion</a>
32  <button class="navbar-toggler" type="button" data-toggle="collapse"
33    data-target="#collapsibleNavbar">
34    <span class="navbar-toggler-icon"></span>
35  </button>
36  <div class="collapse navbar-collapse" id="collapsibleNavbar">
37    <ul class="navbar-nav">
38      <li class="nav-item"><a class="nav-link" href="#">Uno a
39        muchos</a></li>
40      <li class="nav-item"><a class="nav-link" href="#">En
41        construcción</a></li>
42      <li class="nav-item"><a class="nav-link" href="#">En
43        construcción</a></li>
44    </ul>
45  </div>
46</nav>
47
48<div class="container" style="margin-top: 30px">
49
50</div>
51
52</body>
```

Vista implementada.



Tenemos una tabla de departamentos que puede tener muchos o un empleado, y una tabla de empleados en donde un mismo empleado no puede pertenecer a más de un departamento.

```
SELECT EMP.NUMEMPLEADO, EMP.NOMBRE, DEP.NOMDEPTO
FROM EMPLEADO EMP INNER JOIN DEPARTAMENTO DEP ON DEP.NUMDEPTO = EMP.NUMDEPTO
WHERE DEP.NOMDEPTO = 'INFORMATICA';
```

Consulta INNER JOIN entre empleado y departamento.

```

48@ <div class="container" style="margin-top: 30px">
49@   <div class="row">
50@     <div class="col-sm-3">
51@       <h2>departamento</h2>
52@     </div>
53@
54@     <div class="col-sm-9">
55@       <h2>resultados</h2>
56@     </div>
57@   </div>
58@   <br>
59@   <div class="row">
60@     <form action="procesaBusquedaEmplDept" method="post">
61@       <div class="col-sm-3">
62@         <label for="NOMBRE DEPARTAMENTO">Nombre Departamento:</label>
63@         <input type="text" class="form-control" id="nomDepto" name="nomDepto">
64@         <br>
65@         <button type="button" class="btn btn-primary">Buscar</button>
66@       </div>
67@       <div class="col-sm-9">
68@         <div id="tabla">
69@           <br>
70@           <table class="table table-sm table-dark">
71@             <thead>
72@               <tr>
73@                 <th scope="col">Numero Empleado</th>
74@                 <th scope="col">Nombre Empleado</th>
75@                 <th scope="col">Nombre Departamento</th>
76@               </tr>
77@             </thead>
78@             <tbody>
79@               <td>Marco Zaron</td>
80@               <td>0458</td>
81@               <td>CHILE</td>
82@             </tbody>
83@           </table>
84@         </div>
85@       </div>
86@     </form>
87@   </div>
88@ </div>

```

{desafío}
latam_

@
Mejorando la vista
de búsqueda

Ejemplo relaciones con JSP

Para el curso de JEE

Navegación Uno a muchos En construcción En construcción

departamento

Nombre Departamento:

Buscar

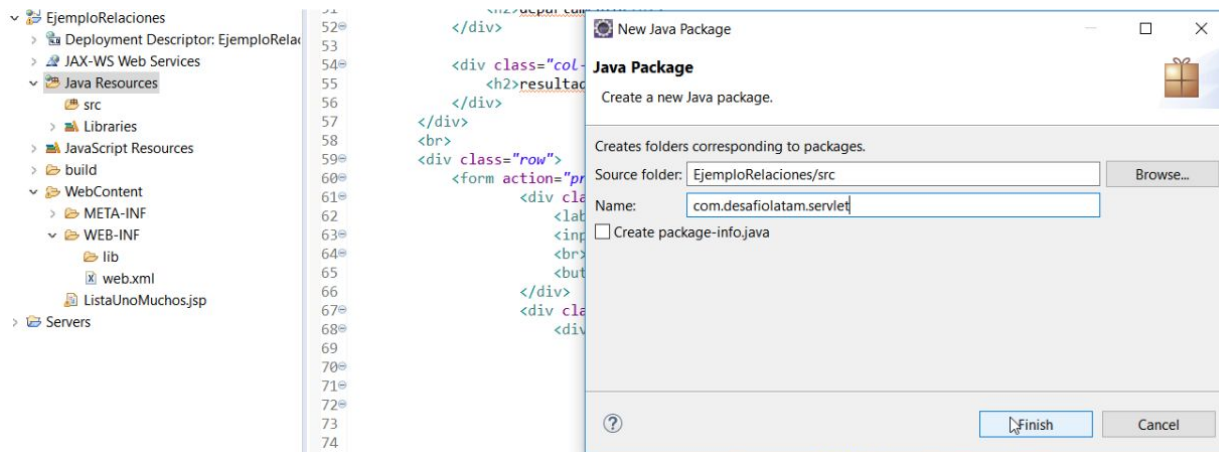
resultados

Numero Empleado	Nombre Empleado	Nombre Departamento
Marco Zaror	0458	CHILE

@ Resultado de la
implementación
de la vista

El botón que quiera procesar una petición debe ser de tipo submit.

Creando Servlet.



Manteniendo el método doPost.

```
1 package com.desafiolatam.servlet;
2
3 import java.io.IOException;
4
5
6
7
8
9
10
11 @WebServlet("/procesaBusquedaEmplDept")
12 public class ProcesaBusquedaEmplDept extends HttpServlet {
13     private static final long serialVersionUID = 1L;
14
15     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
16     }
17 }
18
19 }
```

Creando el modelo de Departamento

```
Departamento.java
1 package com.desafiolatam.modelo;
2
3 public class Departamento {
4     private int numDepto;
5     private String nombreDepto;
6     private String ubicacionDepto;
7
8     public Departamento(int numDepto, String nombreDepto, String ubicacionDepto) {
9         super();
10        this.numDepto = numDepto;
11        this.nombreDepto = nombreDepto;
12        this.ubicacionDepto = ubicacionDepto;
13    }
14
15    //getters y setters
16
17    public int getNumDepto() {
18        return numDepto;
19    }
20
21    public void setNumDepto(int numDepto) {
22        this.numDepto = numDepto;
23    }
}
```

Creando modelo de Empleado

```
Empleado.java
1 package com.desafiolatam.modelo;
2
3 public class Empleado {
4     private int numEmpleado;
5     private String nombreEmpleado;
6     private int numDepto;
7
8     public Empleado(int numEmpleado, String nombreEmpleado, int numDepto) {
9         super();
10        this.numEmpleado = numEmpleado;
11        this.nombreEmpleado = nombreEmpleado;
12        this.numDepto = numDepto;
13    }
14
15    //getters y setters
16
17    public int getNumEmpleado() {
18        return numEmpleado;
19    }
20
21    public void setNumEmpleado(int numEmpleado) {
22        this.numEmpleado = numEmpleado;
23    }
}
```

(El usuario quiere ver todos los empleados que pertenecen a un departamento)

Estructura de la clase DepartamentoEmpleado.

```
1 package com.desafiolatam.modelo;
2
3 public class DepartamentoEmpleado {
4     private Departamento departamento;
5     private Empleado empleado;
6
7     public DepartamentoEmpleado(Departamento departamento, Empleado empleado) {
8         super();
9         this.departamento = departamento;
10        this.empleado = empleado;
11    }
12
13    public Departamento getDepartamento() {
14        return departamento;
15    }
16    public void setDepartamento(Departamento departamento) {
17        this.departamento = departamento;
18    }
19    public Empleado getEmpleado() {
20        return empleado;
21    }
22    public void setEmpleado(Empleado empleado) {
23        this.empleado = empleado;
24    }
25
26
27 }
```

Crearemos la clase que se encargará de contactar con la base de datos.

- Utilizaremos la creación de un pool de conexiones y la estructura de clases del patrón:
 - com.desafiolatam.procesaconexion
 - com.desafiolatam.dao

```
1 package com.desafiolatam.procesaconexion;
2
3 import java.sql.Connection;
4
5
6
7
8
9
10
11
12
13
14 public class AdministradorConexion {
15
16     protected Connection conn = null;
17     protected PreparedStatement pstmt = null;
18     protected ResultSet rs = null;
19
20     protected Connection generaConexion() {
21         String usr = "sys as sysdba";
22         String pwd = "admin";
23         String driver = "oracle.jdbc.driver.OracleDriver";
24         String url = "jdbc:oracle:thin:@//localhost:1521/desafio_ejemplo01";
25
26         try {
27             Class.forName(driver);
28             conn = DriverManager.getConnection(url,usr,pwd);
29         } catch (Exception ex) {
30             ex.printStackTrace();
31         }
32         return conn;
33     }
34
35     protected Connection generaPoolConexion() {
36         Context initContext;
37         try {
38             initContext = new InitialContext();
39             DataSource ds = (DataSource) initContext.lookup("java:/comp/env/jdbc/ConexionOracle");
40             try {
41                 conn = ds.getConnection();
42             } catch (SQLException e) {
43                 e.printStackTrace();
44             }
45         } catch (NamingException e) {
46             e.printStackTrace();
47         }
48         return conn;
49     }
50
51 }
```

Creando interfaz DepartamentoEmpleadoDAO.

```
1 package com.desafiolatam.dao;
2
3 import java.util.List;
4
5
6
7
8 public interface DepartamentoEmpleadoDao {
9     public List<DepartamentoEmpleado> obtieneDepartamento(String nomDepto);
10 }
11
```

Implementando el patrón DAO.

```
10 import com.desafiolatam.modelo.Empleado;
11 import com.desafiolatam.procesaconexion.*;
12
13 public class DepartamentoEmpleadoDaoImpl extends AdministradorConexion implements DepartamentoEmpleadoDao {
14
15     public DepartamentoEmpleadoDaoImpl() {
16         Connection conn = generaPoolConexion();
17     }
18
19     @Override
20     public List<DepartamentoEmpleado> obtieneDepartamento(String nomDepto) {
21         List<DepartamentoEmpleado> deptosEmpleados = new ArrayList<DepartamentoEmpleado>();
22         String query = "SELECT * FROM EMPLEADO EMP INNER JOIN DEPARTAMENTO DEP ON DEP.NUMDEPTO = EMP.NUMDEPTO\r\n" +
23             "WHERE ";
24
25         if((nomDepto.isEmpty()) && nomDepto.isEmpty()) {
26             query = "SELECT * FROM EMPLEADO EMP INNER JOIN DEPARTAMENTO DEP ON DEP.NUMDEPTO = EMP.NUMDEPTO";
27         } else {
28             query += "DEP.NUMDEPTO = '"+nomDepto+"'";
29         }
30
31         try {
32             pstmt = conn.prepareStatement(query);
33             rs = pstmt.executeQuery();
34             while(rs.next()) {
35                 Departamento depto = new Departamento(rs.getInt("NUMDEPTO"), rs.getString("NOMDEPTO"), rs.getString("UBICACIONDPTO"));
36                 Empleado empleado = new Empleado(rs.getInt("NUMEMPLEADO"), rs.getString("NOMBRE"), rs.getInt("NUMDEPTO"));
37                 DepartamentoEmpleado deptoEmpl = new DepartamentoEmpleado(depto, empleado);
38                 deptosEmpleados.add(deptoEmpl);
39             }
40         } catch (SQLException e) {
41             e.printStackTrace();
42         }
43         return deptosEmpleados;
44     }
45
46 }
```

código de la implementación	
Línea 21	Se genera una lista de tipo DepartamentoEmpleado.
Línea 22	Contiene la query con el inner join.
Línea 34	Recorremos el resultset para rescatar los valores.

Procesando el resultado para mostrarlo en la vista.

```

17 @WebServlet("/procesaBusquedaEmplDept")
18 public class ProcesaBusquedaEmpDept extends HttpServlet {
19     private static final long serialVersionUID = 1L;
20
21     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
22         String nomDepartamento = (String) request.getParameter("nomDepto");
23
24         List<DepartamentoEmpleado> deptosEmpleadosList = new ArrayList<DepartamentoEmpleado>();
25
26         DepartamentoEmpleadoDaoImpl obtieneDeptoEmpleado = new DepartamentoEmpleadoDaoImpl();
27
28         deptosEmpleadosList = obtieneDeptoEmpleado.obtieneDepartamento(nomDepartamento);
29
30         request.setAttribute("departamentoEmpleado", deptosEmpleadosList);
31
32         request.getRequestDispatcher("ListaUnoMuchos.jsp").forward(request, response);
33     }
34
35 }

```

El servlet.

No maneja lógica de negocio, hace las llamadas correspondientes al dao y devuelve los valores a la vista.

- En la línea 24 se crea una lista de tipo DepartamentoEmpleado y a continuación crea una instancia del dao DepartamentoEmpleadoDaoImpl.
- Luego de obtener los valores y guardarlos, el objeto request añade la lista obtenida y luego redirige a la vista pasando tal request.

Instanciando la clase DepartamentoEmpleado.

```
23= <%  
24     List<DepartamentoEmpleado> deptos = new ArrayList<DepartamentoEmpleado>();  
25     deptos = (List)request.getAttribute("departamentoEmpleado");  
26 %>  
27 </head>
```

Mostrando los elementos de la lista

```
<table class="table table-sm table-dark">
  <thead>
    <tr>
      <th scope="col">Numero Empleado</th>
      <th scope="col">Nombre Empleado</th>
      <th scope="col">Nombre Departamento</th>
    </tr>
  </thead>
  <tbody>
    <%
      if(deptos != null){
        for(DepartamentoEmpleado depto: deptos){
          %>
          <tr>
            <td><%=depto.getEmpleado().getNumEmpleado()%></td>
            <td><%=depto.getEmpleado().getNombreEmpleado()%></td>
            <td><%=depto.getDepartamento().getNombreDepto()%></td>
          </tr>
          <%%>
        }
      }
    </tbody>
  </table>
```

{desafío}
latam_

Resultado de la consulta

Navegacion Uno a muchos En construcción En construcción

departamento resultados

Nombre Departamento:

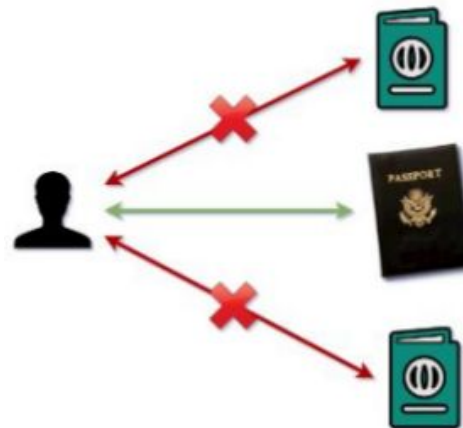
CONTABILIDAD

Buscar

Numero Empleado	Nombre Empleado	Nombre Departamento
31	Kilos Demetrio	CONTABILIDAD
32	Astorga Daniel	CONTABILIDAD
33	Yeny Marlen	CONTABILIDAD
34	Julia Zapata	CONTABILIDAD
35	Esteban Cerrado	CONTABILIDAD
41	Ricardo Abrilar	CONTABILIDAD
42	Rocio Vicencio	CONTABILIDAD
43	Bianca Vicencio	CONTABILIDAD

Relación 1:1

Por cada registro de la tabla principal solo existe un registro en su tabla relacionada.



Creando tabla DOC_EXTRANJERIA.

```
CREATE TABLE DOC_EXTRANJERIA (ID_PASAPORTE NUMBER PRIMARY KEY, PAIS_ORIGEN VARCHAR(100), DESCRIPCION VARCHAR(100));  
ALTER TABLE DOC_EXTRANJERIA ADD CONSTRAINT DOC_EXTRANJERIA_EMPL FOREIGN KEY (ID_PASAPORTE) REFERENCES EMPLEADO (NUMEMPLEADO) ON DELETE CASCADE;
```

Relación mucho a muchos

Cuando muchos elementos de una tabla se relacionan con muchos elementos de otra tabla.

Ejemplo:

Un Profesor enseña muchas unidades de un curso y a su vez una Unidad puede tener muchos profesores que la imparten.

Clase Profesor

```
@Entity
@Table(name="Profesor")
public class Profesor implements Serializable {
    @Id
    @Column(name="Id")
    private int id;
    @Column(name="nombre")
    private String nombre;
    @Column(name="email")
    private String email;
    @ManyToMany(cascade = {CascadeType.ALL})
    @JoinTable(name="ProfesorUnidad",
joinColumns={@JoinColumn(name="IdProfesor")},
inverseJoinColumns={@JoinColumn(name="IdUnidad")})
    private Set<Unidad> unidades=new HashSet();
    public Profesor(){
    }
    public Profesor(int id, String nombre, String email)
    {
        this.id = id;
        this.nombre = nombre;
        this.email = email;
    }
    //agregar Getter y Setter
}
```

Clase Unidad

```
@Entity
@Table(name="Unidad")
public class Unidad implements Serializable {
    @Id
    @Column(name="IdUnidad")
    private int IdUnidad;
    @Column(name="nombre")
    private String nombre;

    @ManyToMany(cascade =
{CascadeType.ALL},mappedBy="unidades")
    private Set<Profesor> profesores=new HashSet();
    public Unidad() {
    }
    public Unidad(int IdUnidad, String nombre) {
        this.IdUnidad = IdUnidad;
        this.nombre = nombre;
    }
    //agregar Getter y Setter
}
```

Relación Recursiva

Indica que si una tabla se relaciona consigo misma estamos en presencia de una relación recursiva o asociada a si misma.

Ejemplo:

Un directorio puede tener archivos y a su vez otros directorios que contendrán otros archivos y directorios (carpetas).

Clase Directorio

```
public abstract class Directorio {  
    private String nombre;  
    public String getNombre() {  
        return nombre;  
    }  
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
    public Directorio(String nombre) {  
        super();  
        this.nombre = nombre;  
    }  
    public abstract boolean esCarpeta() ;  
    public abstract List<Directorio>  
getDirectorios();  
}
```

{desafío}
latam_

Clase Archivo

```
public class Archivo extends Directorio {  
    public Archivo(String nombre) {  
        super(nombre);  
    }  
    @Override  
    public boolean esCarpeta() {  
        return false;  
    }  
    @Override  
    public List<Directorio> getDirectorios() {  
        throw new RuntimeException(" un  
archivo no contiene directorios");  
    }  
}
```

Clase Carpeta.

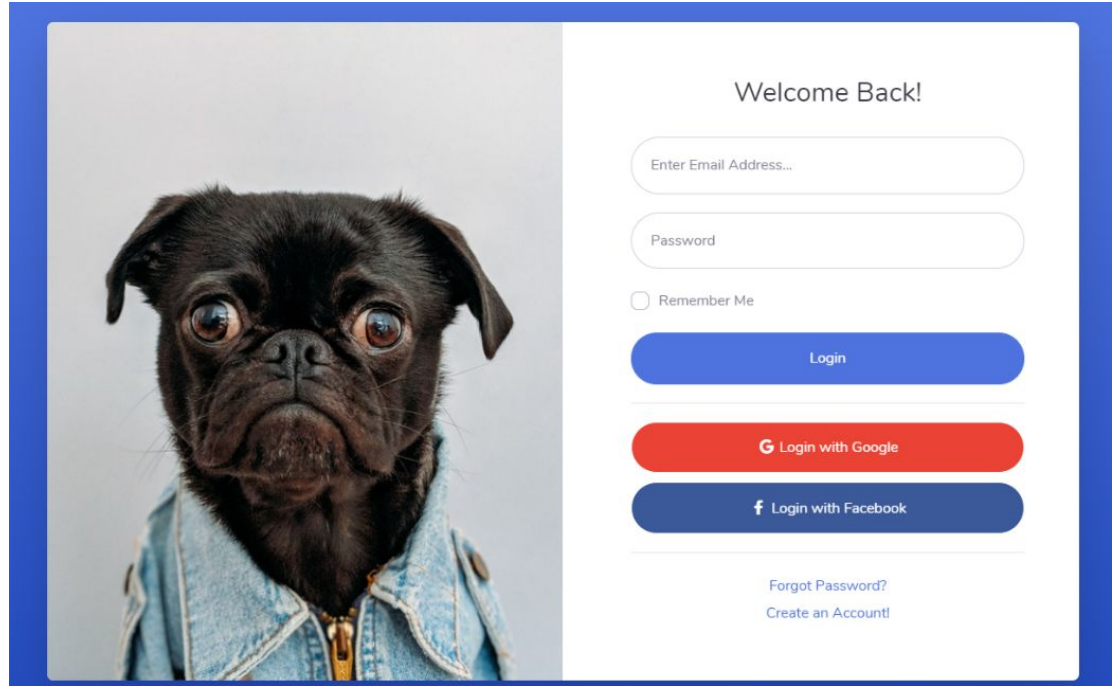
```
public class Carpeta extends Directorio{
    public Carpeta(String nombre) {
        super(nombre);
    }
    private List<Directorio> directorios= new ArrayList<>();
    public List<Directorio> getDirectorios() {
        return directorios;
    }
    public void setDirectorios(List<Directorio> directorios) {
        this.directorios = directorios;
    }
    public void addDirectorio (Directorio d) {
        directorios.add(d);
    }
    @Override
    public boolean esDirectorio() {
        return true;
    }
}
```

**/* Aplicación web dinámica con patrón DAO y
manipulación de datos */**

Estructurar vista bootstrap a un proyecto JSP

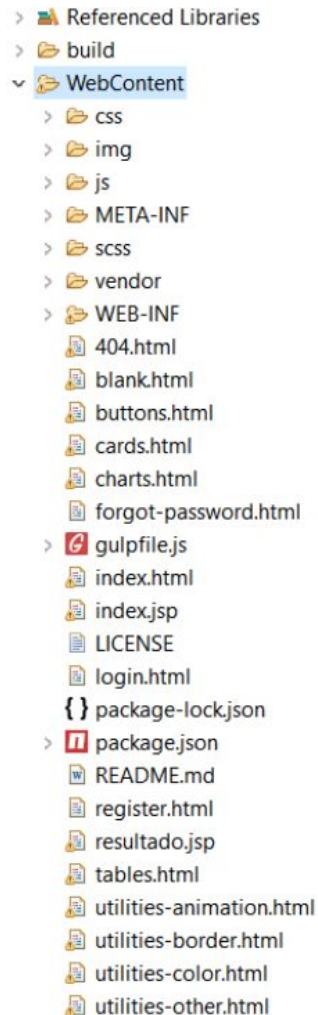
La imagen es referencial.

Plantilla Bootstrap.



Para utilizar la plantilla seguir los siguientes pasos:

1. Abrir la carpeta WebContent y en ella copiar todo el contenido de la carpeta de la plantilla que se acaba de instalar.



Archivo web.xml

Configurando la página de bienvenida



```
1  <web-app>
2  <welcome-file-list>
3    <welcome-file>login.html</welcome-file>
4  </welcome-file-list>
5  <resource-ref>
6    <description>Pool conexiones</description>
7    <res-ref-name>jdbc/ConexionOracle</res-ref-name>
8    <res-type>javax.sql.DataSource</res-type>
9    <res-auth>Container</res-auth>
10 </resource-ref>
11 </web-app>
```

En la línea 4.

Indicamos que la página de bienvenida será la login.html.

Estas páginas están en formato html y para efectos del curso debemos convertirlas en páginas JSP.

```
<%@ page language="java" contentType="text/html;  
charset=ISO-8859-1"  
    pageEncoding="ISO-8859-1"%>
```

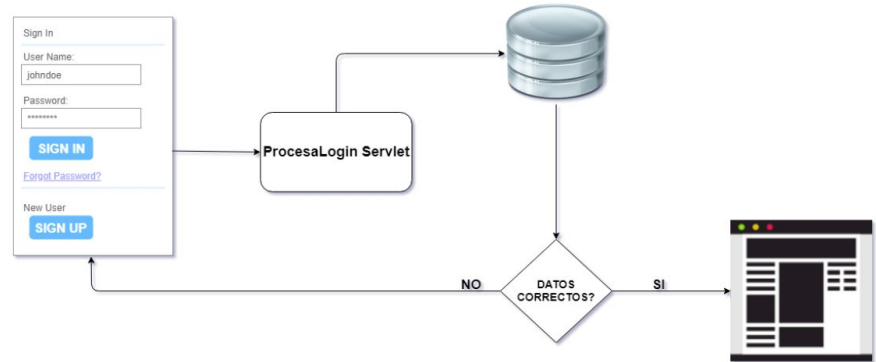
Creando la tabla de usuarios

Crear una nueva tabla en la base de datos que almacene los valores de los usuarios administradores.

```
CREATE TABLE USUARIOS_ADMIN (ID_USUARIO  
NUMBER PRIMARY KEY, CORREO  
VARCHAR(100), PASS VARCHAR(100),  
PRIMER_NOMBRE VARCHAR(100),  
SEGUNDO_NOMBRE VARCHAR(100));
```

Añadiendo funcionalidad a login.jsp para login

Deseamos que la aplicación permite iniciar sesión mediante un correo y una contraseña que el usuario ingresará en la página login.jsp.



- **Iniciar el login.**

- Primero se trabaja sobre la vista.
- Configurar las etiquetas form con:
 - Etiqueta action="procesaLogin"
 - Etiqueta method="get"

Agregando login al JSP.

```
<form class="user" action="procesaLogin" method="GET">
  <div class="form-group">
    <input type="email" name="email" class="form-control form-control-user" id="exampleInputEmail" aria-describedby="emailHelp" placeholder="Enter email">
  </div>
  <div class="form-group">
    <input type="password" name="password" class="form-control form-control-user" id="exampleInputPassword" placeholder="Password">
  </div>
  <div class="form-group">
    <div class="custom-control custom-checkbox small">
      <input type="checkbox" class="custom-control-input" id="customCheck">
      <label class="custom-control-label" for="customCheck">Remember Me</label>
    </div>
  </div>
  <input type="submit" value="Login" class="btn btn-primary btn-user btn-block">
</form>
```

Servlet procesaLogin

Luego de modificar la capa de vista, se modifica la capa controladora, representada en este caso por el servlet ProcesaLogin.

```
30 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
31     boolean usuarioExiste = false;
32     String correo = request.getParameter("email");
33     String password = request.getParameter("password");
34     LoginDaoImp loginDao = new LoginDaoImp();
35     usuarioExiste = loginDao.usuarioRegistrado(correo,password);
36     request.setAttribute("usuarioExiste", usuarioExiste);
37     if(usuarioExiste) {
38         HttpSession sesionUsuario = request.getSession(true);
39         sesionUsuario.setAttribute("correo", correo);
40         request.setAttribute("correo", correo);
41         request.getRequestDispatcher("index.jsp").forward(request, response);
42         return;
43     }
44     //request.setAttribute("deptoDao", listaDeptos);
45     request.getRequestDispatcher("login.jsp").forward(request, response);
46 }
```

Modificando el controlador ProcesaLogin.

Método getParameter.

Se obtienen los valores enviados desde el formulario para luego enviarlos al metodo usuario Registrado correspondiente a la clase LoginDao.

LoginDao.

```
1 package com.desafiolatam.dao;
2
3 public interface LoginDao {
4     public boolean usuarioRegistrado(String correo, String password);
5 }
```

Implementando LoginDao.

```
18@ Override
19 public boolean usuarioRegistrado(String correo, String password) {
20     boolean usuarioExiste = false;
21     String sql = "SELECT * FROM USUARIOS_ADMIN WHERE CORREO = '" + correo + "'" + " AND PASS = '" + password + "'";
22     try {
23         pstmt = conn.prepareStatement(sql);
24         rs = pstmt.executeQuery();
25         if(rs.next()) {
26             usuarioExiste = true;
27         }
28     } catch (SQLException e) {
29         // TODO Auto-generated catch block
30         e.printStackTrace();
31     }
32     return usuarioExiste;
33 }
34 }
```


El método retorna un booleano es comprobada desde el servlet, el cual dependiendo del resultado redirige al index del sistema o a la pantalla de inicio de sesión para que la gente la ingrese nuevamente.

```
/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    boolean usuarioExiste = false;
    String correo = request.getParameter("email");
    String password = request.getParameter("password");
    LoginDaoImp loginDao = new LoginDaoImp();
    usuarioExiste = loginDao.usuarioRegistrado(correo,password);
    request.setAttribute("usuarioExiste", usuarioExiste);
    if(usuarioExiste) {
        HttpSession sesionUsuario = request.getSession(true);
        sesionUsuario.setAttribute("correo", correo);
        request.setAttribute("correo", correo);
        request.getRequestDispatcher("index.jsp").forward(request, response);
        return;
    }
    //request.setAttribute("deptoDao", listaDeptos);
    request.getRequestDispatcher("login.jsp").forward(request, response);
}
```

El servlet también crea una variable de sesión la cual registra el correo para luego validar la presencia de un usuario registrado en el sistema.

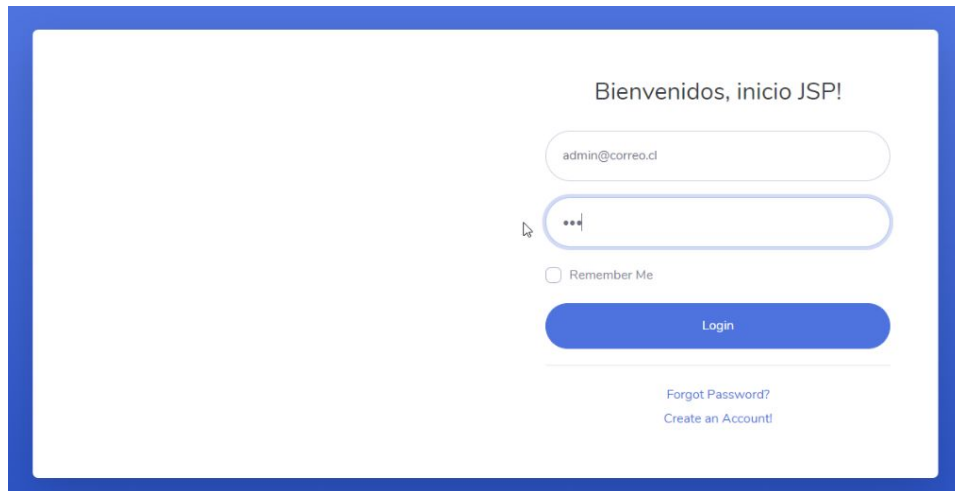
```
35 <body id="page-top">
36 <%
37     String correo = "no existe usuario";
38     HttpSession misesion = request.getSession();
39     if (mision != null) {
40         correo = (String) misesion.getAttribute("correo");
41     } else {
42         request.getRequestDispatcher("login.jsp").forward(request, response);
43     }
44 %>
```

La línea 40 obtiene el valor 'correo' que se asignó en el servlet para guardarla en la variable del mismo nombre.

USUARIOS_ADMIN

Result				
SELECT * FROM USUARIOS_ADMIN FOR update Enter a SQL expression to				
Grid	123	ID_USUARIO	ABC CORREO	ABC PASS
1	2		admin@correo.cl	123
2	21		jorgecarmona1986@gmail.com	123
3	41		bianquita@correo.cl	1234

Iniciando sesión



Bienvenidos, inicio JSP!

admin@correo.cl

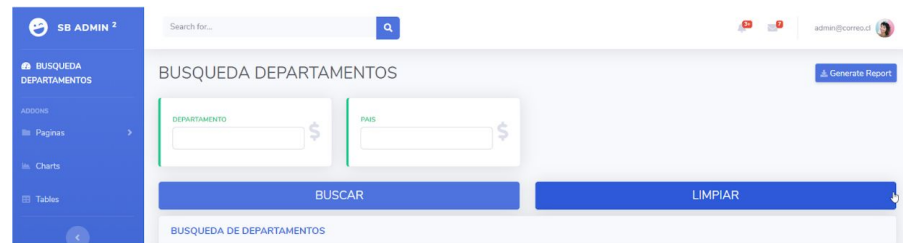
☐ Remember Me

Login

[Forgot Password?](#)

[Create an Account!](#)

Inicio correcto



SB ADMIN ?

Search for...

admin@correo.cl

BUSQUEDA DEPARTAMENTOS

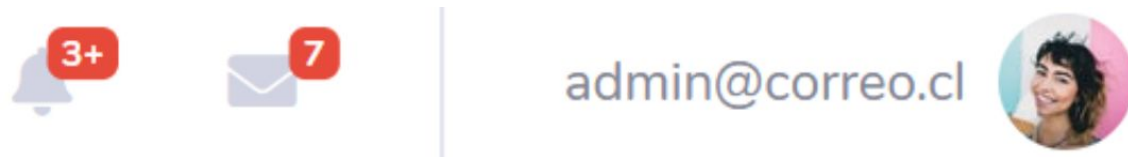
Generate Report

DEPARTAMENTO \$ PAIS \$

BUSCAR LIMPIAR

BUSQUEDA DE DEPARTAMENTOS

Usuario conectado.



En la imagen a continuación se grafica la porción de código que hace tal tarea con la variable `<%=correo%>`:

```
292 <!-- Nav Item - User Information -->
293 <li class="nav-item dropdown no-arrow"><a
294     class="nav-link dropdown-toggle" href="#" id="userDropdown"
295     role="button" data-toggle="dropdown" aria-haspopup="true"
296     aria-expanded="false">
297     <span>class="mr-2 d-none d-lg-inline text-gray-600 small"><%=correo%></span>
298     
```

Variable correo.

Construcción de filtros de departamentos

- En un sistema web enfocado a la empresa.
- Los mantenedores dan mantenimiento a los datos alojados en las tablas de base de datos.
- Los filtros son parámetros de las tablas en la cual restringimos los resultados a lo que el usuario desee ver.

SELECT * FROM EMPLEADO | Enter a SQL expression to filter results (use Ctrl+Space)

	123 NUMEMPLEADO	ABC NOMBRE	123 NUMDEPTO	
1	101	Matias Zarate	11	
2	16	Matias Zarate	15	
3	17	Felipe Perez	15	
4	18	Bob Marley	15	
5	19	Pedro Polanco	15	
6	20	Eduardo Azul	15	
7	21	Oscar Fernandez	14	

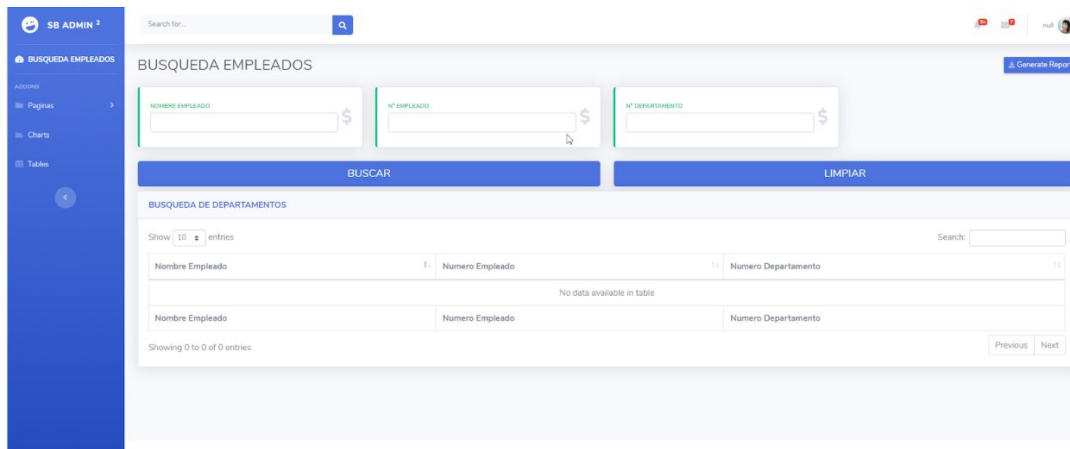
Si queremos ver a todos los empleados que pertenezcan al departamento número 14.

```
select * from empleados where numdepto = 14;
```

Se desea obtener a todos los empleados del departamento 14 y que se llame Pedro Polanco la query aumenta algo de complejidad:

```
select * from empleados where numdepto = 14 and  
nombre = 'Pedro Polanco' ;
```

Modificar el archivo index.jsp y en él vamos a agregar tres input text y dos botones (buscar y limpiar).



Creando campo de texto.

{desafío}
latam_

```
343 <!-- Earnings (Monthly) Card Example -->
344 <div class="col-xl-3 col-md-6 mb-4">
345   <div class="card border-left-success shadow h-100 py-2">
346     <div class="card-body">
347       <div class="row no-gutters align-items-center">
348         <div class="col mr-2">
349           <div
350             class="text-xs font-weight-bold text-success text-uppercase mb-1">Nombre Empleado</div>
351           <input type="text" class="form-control" id="nombre" name="nombre">
352         </div>
353         <div class="col-auto">
354           <i class="fas fa-dollar-sign fa-2x text-gray-300"></i>
355         </div>
356       </div>
357     </div>
358   </div>
359 </div>
```

Creando campos nombre de empleado, número de empleado y número de departamento.

SB ADMIN

Search for...

BUSQUEDA EMPLEADOS

Generate Report

NOMBRE EMPLEADO \$

Nº EMPLEADO \$

Nº DEPARTAMENTO \$

BUSCAR

LIMPIAR

BUSQUEDA DE DEPARTAMENTOS

Show 10 entries

Search:

Nombre Empleado	T1	Numero Empleado	T2	Numero Departamento	T3
No data available in table					
Nombre Empleado		Numero Empleado		Numero Departamento	

Showing 0 to 0 of 0 entries

Previous Next

ProcesaBúsquedaEmpleado.

325

```
<form action="procesaBusquedaEmpleado" method="get">
```

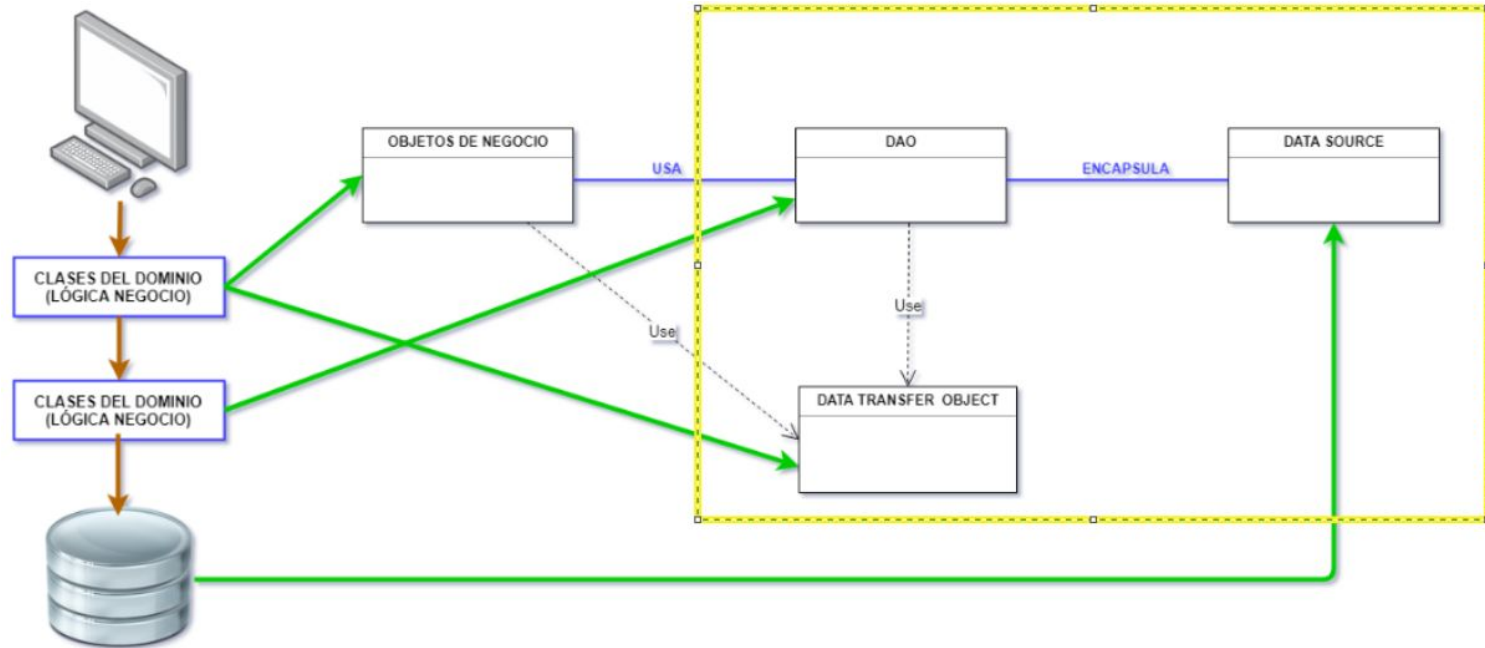

Creando botón Buscar.

```
<button type="submit" class="btn btn-primary btn-lg btn-block">BUSCAR</button>
```

Servlet ProcesaBusquedaEmpleado.java.

```
ProcesaBusquedaEmpleado.java
1 package com.desafiolatam.servlet;
2
3 import java.io.IOException;
4
5 /**
6  * Servlet implementation class ProcesaLogin
7  */
8 @WebServlet("/procesaBusquedaEmpleado")
9 public class ProcesaBusquedaEmpleado extends HttpServlet {
10     private static final long serialVersionUID = 1L;
11
12     /**
13      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
14      */
15     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
16         List<Empleado> listaEmpleados = new ArrayList<Empleado>();
17         String nombre = request.getParameter("nombre");
18         String numEmpleado = request.getParameter("numEmpleado");
19         String numDepartamento = request.getParameter("numDepto");
20
21         if(numEmpleado.isEmpty()) {
22             numEmpleado = "0";
23         }
24
25         if(numDepartamento.isEmpty()) {
26             numDepartamento = "0";
27         }
28
29         BusquedaEmpleadoDaoImp busqueda = new BusquedaEmpleadoDaoImp();
30         listaEmpleados = busqueda.busquedaDepto(nombre, Integer.parseInt(numEmpleado), Integer.parseInt(numDepartamento));
31         request.setAttribute("empleados", listaEmpleados);
32         request.getRequestDispatcher("busquedaEmpleados.jsp").forward(request, response);
33     }
34 }
```

Diagrama patrón DAO.



Creación de clases DAO

Dentro del package com.desafiolatam.dao crear una interface y una clase de nombre:

- BusquedaEmpleadoDao.java
- BusquedaEmpleadoDaoImp.java

Interface BusquedaEmpleadoDao.

```
1 package com.desafiolatam.dao;
2
3 import java.util.List;
4
5
6
7
8 public interface BusquedaEmpleadoDao {
9     public List<Empleado> busquedaEmpleado(String nombre, int numEmpleado, int numDepartamento);
10 }
```

BusquedaEmpleadoDao.

Implementación de la clase BusquedaEmpleadoDaoImp.java.

```
3*import java.sql.Connection;
11 public class BusquedaEmpleadoDaoImp extends AdministradorConexion implements BusquedaEmpleadoDao {
12     public BusquedaEmpleadoDaoImp() {
13         Connection conn = generaPoolConexion();
14     }
15     @Override
16     public List<Empleado> busquedaEmpleado(String nombre, int numEmpleado, int numDepartamento) {
17         String query = "SELECT * FROM EMPLEADO WHERE ";
18         List<Empleado> empleados = new ArrayList<Empleado>();
19         if((nombre.isEmpty()) && (numEmpleado == 0) && (numDepartamento == 0)) {
20             query = "SELECT * FROM EMPLEADO";
21         } else {
22             if(!nombre.isEmpty() && (numEmpleado > 0) && (numDepartamento > 0)) {
23                 query.concat("NOMBRE = '"+nombre+"' " + "AND" + "NUMEMPLEADO = " + numEmpleado + " AND " + "NUMDEPTO = " + numDepartamento);
24             } else {
25                 if(!nombre.isEmpty()) {
26                     query += "NOMBRE = '"+nombre+"'";
27                 }
28                 if(numEmpleado > 0){
29                     query += "NUMEMPLEADO = " + numEmpleado;
30                 }
31                 if(numDepartamento > 0){
32                     query += "NUMDEPTO = " + numDepartamento;
33                 }
34             }
35         }
36     }
37     try {
38         pstmt = conn.prepareStatement(query);
39         rs = pstmt.executeQuery();
40         while(rs.next()) {
41             Empleado empleado = new Empleado(rs.getInt("NUMEMPLEADO"),rs.getString("NOMBRE"),rs.getInt("NUMDEPTO"));
42             empleados.add(empleado);
43         }
44     } catch (SQLException e) {
45         // TODO Auto-generated catch block
46         e.printStackTrace();
47     }
48     return empleados;
49 }
50 }
51 }
```

Generación de tabla normal.

```
444Ⓜ
445Ⓜ
446
447Ⓜ
448Ⓜ
449
450
451
452
453
454Ⓜ
455Ⓜ
456
457
458
459
460
461Ⓜ
462Ⓜ
463
464
465
466
467
468
469Ⓜ
470
471
472
473
474Ⓜ
475
476
477
478
479
```

```
<div class="table-responsive">
  <table class="table table-bordered" id="dataTable"
    width="100%" cellspacing="0">
    <thead>
      <tr>
        <th>Nombre Empleado</th>
        <th>Numero Empleado</th>
        <th>Numero Departamento</th>
      </tr>
    </thead>
    <tfoot>
      <tr>
        <th>Nombre Empleado</th>
        <th>Numero Empleado</th>
        <th>Numero Departamento</th>
      </tr>
    </tfoot>
    <tbody>
      <%
        if (request.getAttribute("empleadoDao") != null) {
          List<Empleado> empleados;
          empleados = (List) request.getAttribute("empleadoDao");

          for (Empleado empleado : empleados) {
            %>
            <tr>
              <td><%=empleado.getNombreEmpleado()%></td>
              <td><%=empleado.getNumEmpleado()%></td>
              <td><%=empleado.getNumDepto()%></td>
            </tr>
            <%
              }
            %>
          </tbody>
        </table>
```

Búsqueda de tablas con filtros.

SB ADMIN 2

BUSQUEDA EMPLEADOS

ADDONS

Paginas

Charts

Tables

Search for...

Q

7

8

Full

Generata Report

NOMBRE EMPLEADO

admin

N° EMPLEADO

N° DEPARTAMENTO

BUSCAR

LIMPIAR

BUSQUEDA DE DEPARTAMENTOS

Show 10 entries

Search:

Nombre Empleado	Numero Empleado	Numero Departamento
Adamiro Rojas	62	6
Alex Citrico	58	7
Alfonso Mastil	60	7
Arturo Prat	70	5
Aston Gonzalez	79	3
Astorga Daniel	32	12
Bianca Vicencio	43	10
Bob Marley	18	15

Activate Windows



Quiz



{desafío}
latam_

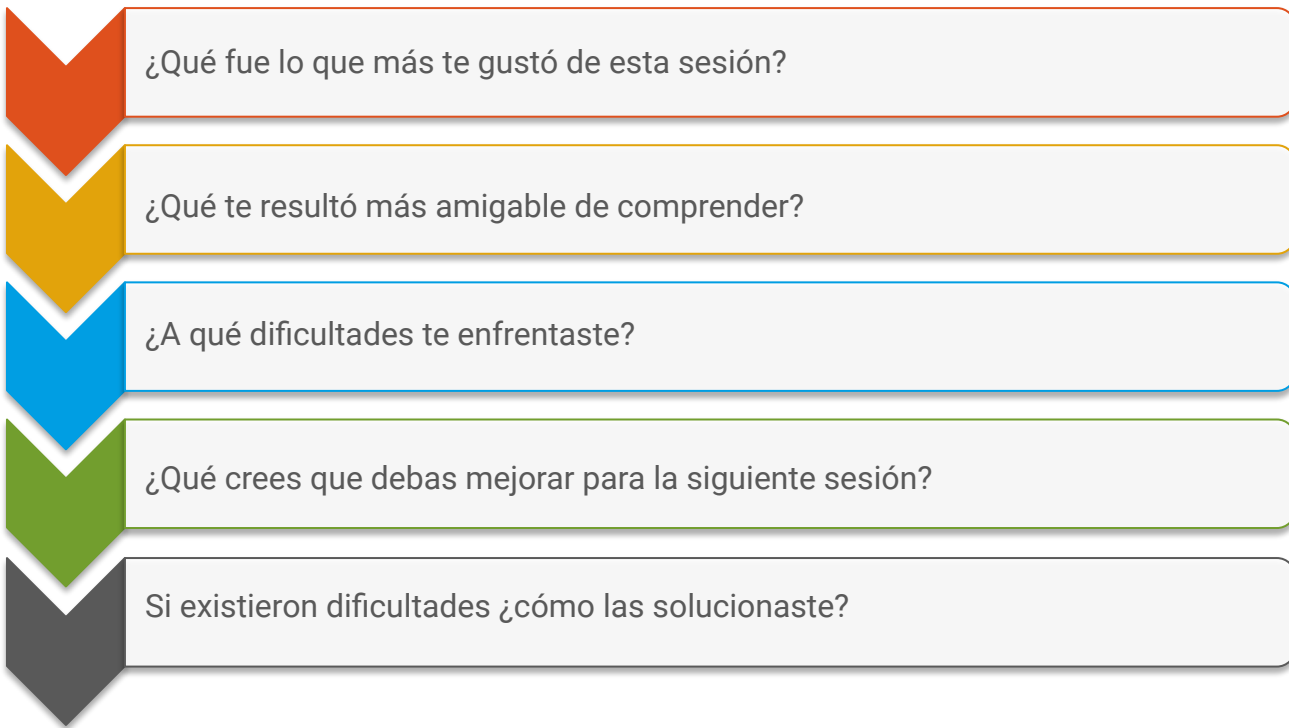


Cierre

{desafío}
latam_



15 minutos



¿Qué fue lo que más te gustó de esta sesión?

¿Qué te resultó más amigable de comprender?

¿A qué dificultades te enfrentaste?

¿Qué crees que debas mejorar para la siguiente sesión?

Si existieron dificultades ¿cómo las solucionaste?



*Academia de
talentos digitales*

www.desafiolatam.com



/DesafioLatam



/DesafioLatam



/DesafioLatam



/DesafioLatam