

2018

UNIVERSIDAD POLITECNICA SALESIANA

Byron Calle, Rommel Inga, Ricardo Pozo y
Edwin Quishpe

Mejora



	Lo que Tenemos	Bien
1. Plantilla (Formato de página)	Implementa estilos pero no plantilla	Implementa correctamente Plantilla con Facelets
	1puntos	3puntos
2. Código fuente (Arquitectura JEE)	Tiene la estructura básica de JEE pero implementa algunas reglas de negocio en vista (ManagedBean)	Si implementa arquitectura JEE
	1puntos	
		3puntos
3. Código fuente (documentación de código)	No implementa documentación de código fuente	Implementa documentación en todo el proyecto
	0puntos	3puntos
4. Funcionalidad - Cruds relacionados	Implementa relaciones a nivel de entidad pero no en funcionalidad o vista	Implementa relaciones a nivel de entidades y también en vista de manera adecuada
	3puntos	10puntos
5. Funcionalidad - Validaciones	validaciones a nivel de entidades y de vista en todos los mantenimientos presentados pero sin cumplir los parámetros de evaluación anteriores	validaciones a nivel de entidades y de vista en todos los mantenimientos presentados
	3puntos	5puntos
6. Diagrama de clases	Diagrama de clases implementa solo el paquete modelo pero no esta adecuadamente definido en relaciones (aosciación, cardinalidad, etc.)	Diagrama de clases implementado completamente (paquetes modelo, controlador, negocio, etc) y adecuadamente definido en relaciones (asociación, cardinalidad, etc.)
	3puntos	6puntos
7. Funcionalidad - control de excepciones	No implementa control de duplicados pero no se produce el error pues al intentar ingresar un duplicado lo toma como edición	Implementa control de excepciones en todos los mantenimientos, mostrando un mensaje en el caso de que se ingrese un registro duplicado
	2puntos	

1. Implementación de manera correcta las plantillas

Definición de la plantilla

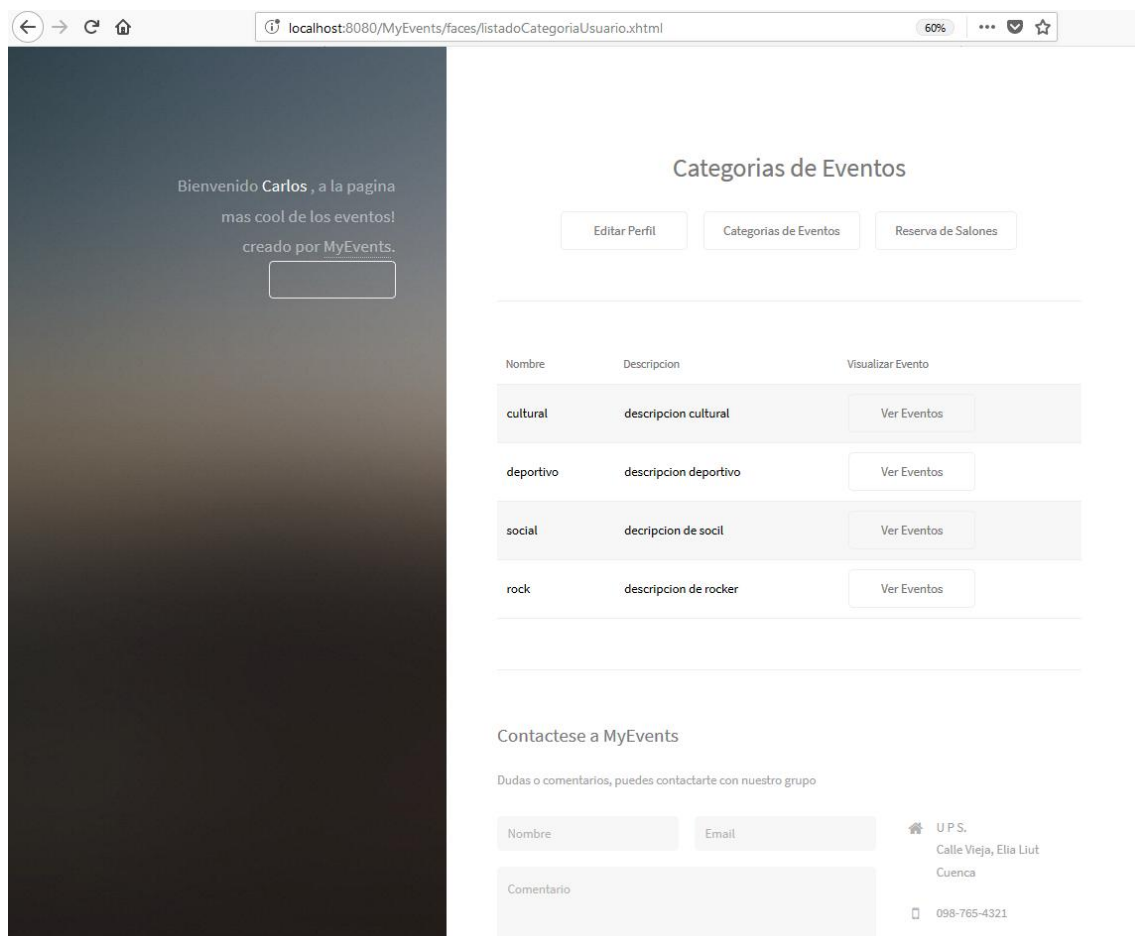
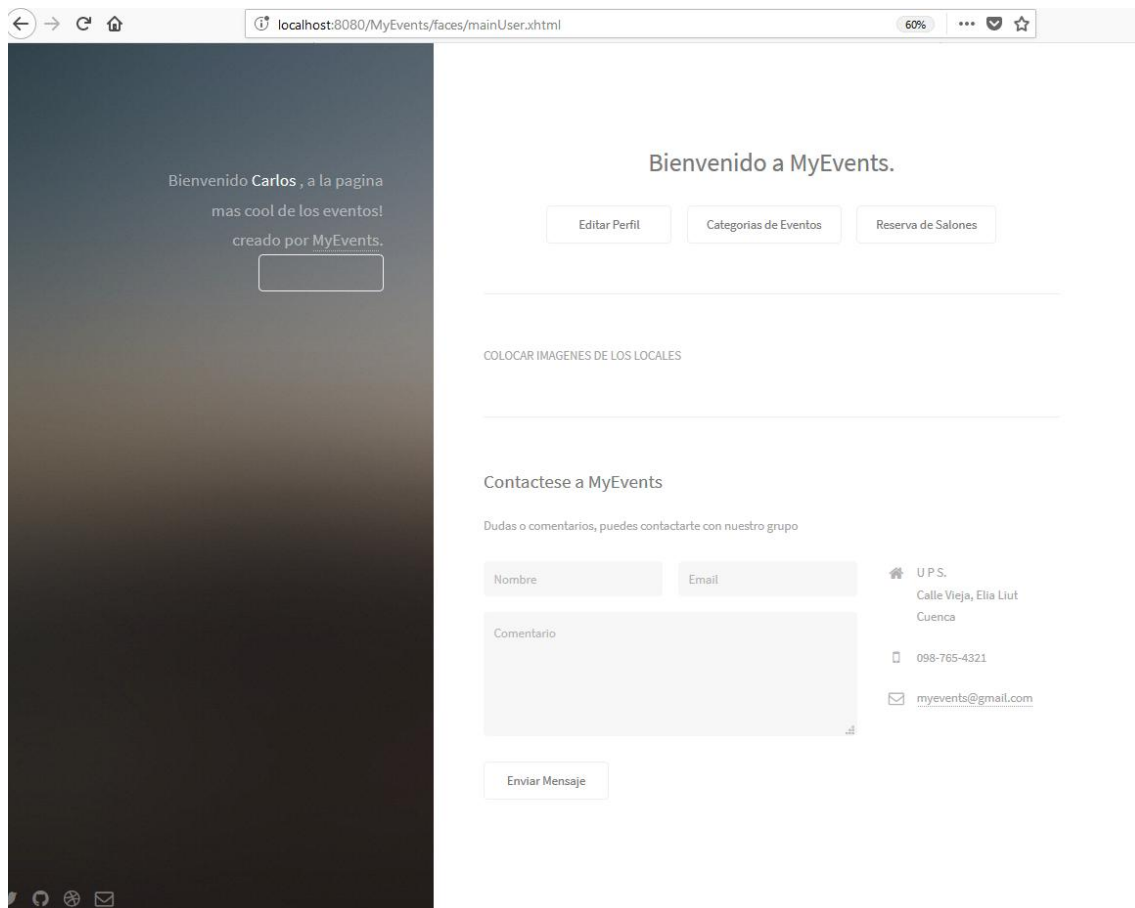
```
Red Hat Central  mainAdmin.xhtml  mainUser.xhtml  plantillaUsuario.xhtml  ⌵
1 <!DOCTYPE HTML>
2<!--
3  Strata by HTML5 UP
4  html5up.net | @ajlkn
5  Free for personal and commercial use under the CCA 3.0 license (html5up.net/license)
6 -->
7<html xmlns="http://www.w3.org/1999/xhtml"
8      xmlns:ui="http://java.sun.com/jsf/facelets"
9      xmlns:f="http://java.sun.com/jsf/core"
10     xmlns:h="http://java.sun.com/jsf/html"
11     xmlns:hs="http://xmlns.jcp.org/jsf/html"
12     xmlns:pt="http://xmlns.jcp.org/jsf/passthrough">
13<h:head>
14     #{personaController.cargarDatosUsuario()}
15
16     <!-- Define un titulo -->
17     <title><ui:insert name="titulo">title</ui:insert></title>
18
19     <meta charset="utf-8" />
20     <meta name="viewport" content="width=device-width, initial-scale=1" />
21     <!--[if lte IE 8]><script src="resources/user-true/assets/js/ie/html5shiv.js"></scrip
22     <link rel="stylesheet" href="resources/user-true/assets/css/main.css" />
23     <!--[if lte IE 8]><link rel="stylesheet" href="assets/css/ie8.css" /><![endif]-->
24 </h:head>
25<body id="top">
26
27     <!-- Header -->
28     <header id="header">
29         <div class="inner">
30             <a href="#" class="image avatar"></a>
32             <h1>
33                 Bienvenido <strong> #{personaController.myUser.nombre} </strong>, a
34                 la pagina<br /> mas cool de los eventos!<br /> creado por <a
35                     href="mainUser.xhtml">MyEvents</a>.
36             </h1>
37             <h:form>
38                 <h:commandButton action="#{personaController.cerrarSesion()}"
39                     value="Salir" class="btn danger"
40                     style="font-size: 100%; color:white;" />
41             </h:form>
42         </div>
```

Reutilizando la plantilla

```
Red Hat Central  mainAdmin.xhtml  mainUser.xhtml  editUser.xhtml  ⌵
1 <ui:composition xmlns="http://www.w3.org/1999/xhtml"
2   xmlns:ui="http://java.sun.com/jsf/facelets"
3   xmlns:f="http://java.sun.com/jsf/core"
4   xmlns:h="http://java.sun.com/jsf/html"
5   xmlns:hs="http://xmlns.jcp.org/jsf/html"
6   xmlns:pt="http://xmlns.jcp.org/jsf/passthrough"
7   template="plantillaUsuario.xhtml">
8
9   <!-- Definicion del titulo respecto a la plantilla -->
10  <ui:define name="titulo">MyEvents - Usuario</ui:define>
11
12  <!-- Definicion del encabezado respecto a la plantilla -->
13  <ui:define name="encabezado">Edicion de Usuario.</ui:define>
14
15  <!-- Definicion del contenido respecto a la plantilla -->
16  <ui:define name="contenido">
17    <f:metadata>
18      <f:viewParam name="id" value="#{personaController.id}"></f:viewParam>
19    </f:metadata>
20
21    <!-- Definicion del formulario para la respectiva edicion del objeto persona -->
22    <h:form>
23
24
25      <h:outputLabel value="Nombre: " for="nombre"/>
26      <h:inputText value="#{personaController.personas.nombre}"
27        id="nombre" pt:placeholder="Nombre"
28        style="color: black" />
29      <h:message for="nombre" errorStyle="color:red; display:block" />
30
31
32      <h:outputLabel value="Apellido: " for="apellido"/>
33      <h:inputText value="#{personaController.personas.apellido}"
34        id="apellido"
35        pt:placeholder="Apellido" style="color: black" />
36      <h:message for="apellido" errorStyle="color:red; display:block" />
37      <br />
38
39      <h:outputLabel value="Cedula: " for="cedula"/>
40      <h:inputText value="#{personaController.personas.cedula}"
41        id="cedula"
42        pt:placeholder="#####" style="color: black" />
43      <h:message for="cedula" errorStyle="color:red; display:block" />
44      <br />
45  </h:form>
23
```

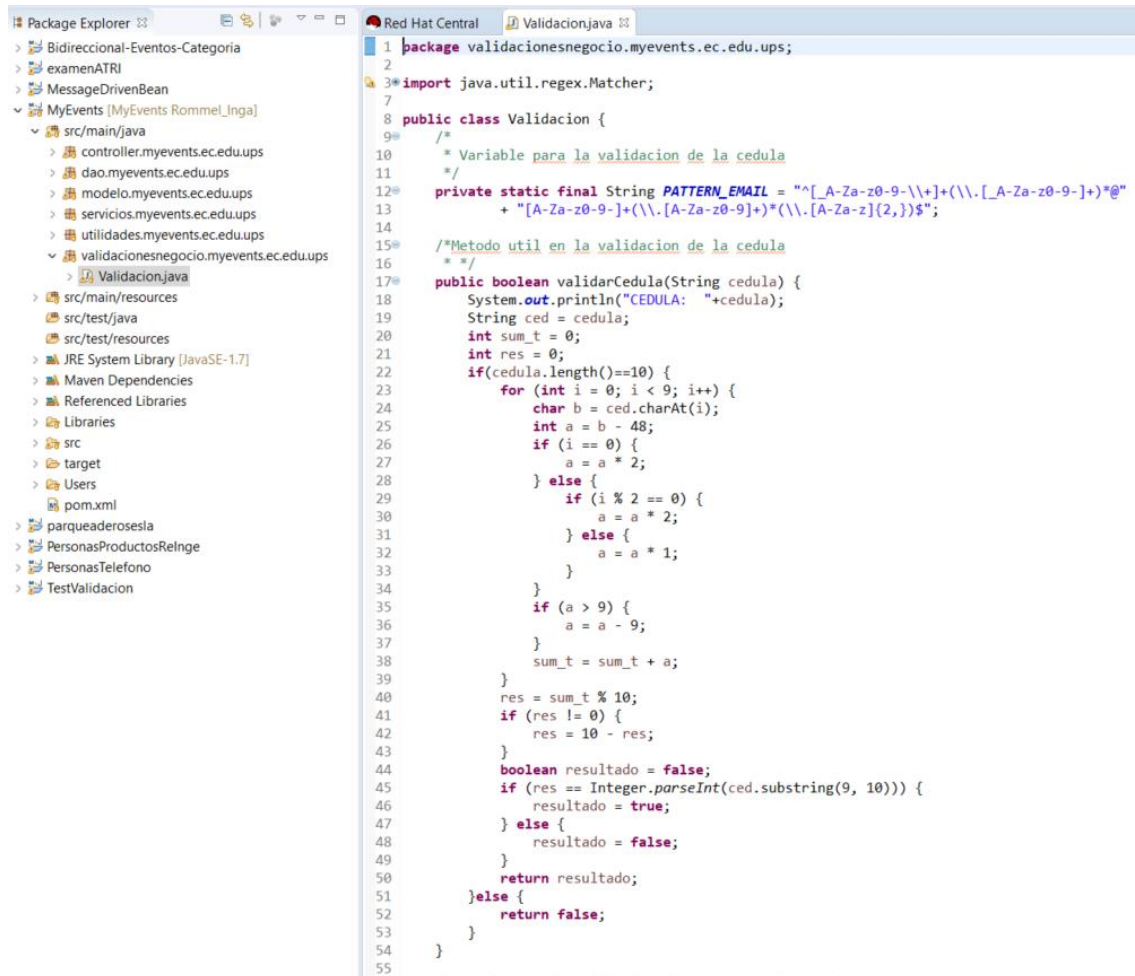
```
Red Hat Central  mainAdmin.xhtml  mainUser.xhtml  consulEventLocPers.xhtml  ⌵
1 <ui:composition xmlns="http://www.w3.org/1999/xhtml"
2   xmlns:ui="http://java.sun.com/jsf/facelets"
3   xmlns:f="http://java.sun.com/jsf/core"
4   xmlns:h="http://java.sun.com/jsf/html"
5   xmlns:hs="http://xmlns.jcp.org/jsf/html"
6   xmlns:pt="http://xmlns.jcp.org/jsf/passthrough"
7   template="PlantillaAdmin.xhtml">
8
9   <!-- Colocar el titulo en la pestana -->
10  <ui:define name="titulo">Salones - Eventos</ui:define>
11
12  <!-- Colocar el Breadcrumbs o seccion donde se encuentra -->
13  <ui:define name="breadcrumbs">Consultas Salones - Eventos</ui:define>
14
15  <!-- Titulo del Contenido -->
16  <ui:define name="bienvenida">Locales - Eventos</ui:define>
17
18  <ui:define name="contenido">
19    <h:form>
20
21      <f:metadata>
22        <f:viewParam name="id" value="#{personaController.idrecuperar}" />
23      </f:metadata>
24
25      <h:outputText value="El registro no contiene datos"
26        rendered="#{empty personaController.plelist}"></h:outputText>
27      <h:dataTable value="#{personaController.plelist}" var="persona"
28        class="table" rendered="#{not empty personaController.plelist}">
29
30        <h:column>
31          <f:facet name="header">Nombre del Salon</f:facet>
32          #{persona.l_nombre}
33        </h:column>
34
35        <h:column>
36          <f:facet name="header">Capacidad</f:facet>
37          #{persona.l_capacidad}
38        </h:column>
39
40        <h:column>
41          <f:facet name="header">Costo</f:facet>
42          #{persona.l_costo}
43        </h:column>
44      </h:dataTable>
45    </h:form>
46  </ui:define>
47</ui:composition>
```

JSF



2. Implementación de la Arquitectura Java EE

Colocación de las validaciones en el paquete Validaciones Negocio



```
1 package validacionesnegocio.myevents.ec.edu.ups;
2
3 import java.util.regex.Matcher;
4
5 public class Validacion {
6     /*
7      * Variable para la validacion de la cedula
8      */
9     private static final String PATTERN_EMAIL = "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
10         + "[A-Za-z0-9-]+(\\.[A-Za-z0-9-]+)*\\.([A-Za-z]{2,})$";
11
12     /*Metodo util en la validacion de la cedula
13     */
14     public boolean validarCedula(String cedula) {
15         System.out.println("CEDULA: "+cedula);
16         String ced = cedula;
17         int sum_t = 0;
18         int res = 0;
19         if(cedula.length()==10) {
20             for (int i = 0; i < 9; i++) {
21                 char b = ced.charAt(i);
22                 int a = b - 48;
23                 if (i == 0) {
24                     a = a * 2;
25                 } else {
26                     if (i % 2 == 0) {
27                         a = a * 2;
28                     } else {
29                         a = a * 1;
30                     }
31                 }
32             }
33             if (a > 9) {
34                 a = a - 9;
35             }
36             sum_t = sum_t + a;
37         }
38         res = sum_t % 10;
39         if (res != 0) {
40             res = 10 - res;
41         }
42         boolean resultado = false;
43         if (res == Integer.parseInt(ced.substring(9, 10))) {
44             resultado = true;
45         } else {
46             resultado = false;
47         }
48         return resultado;
49     }
50 }
51
52
53
54
55
```


3. Código Fuente Documentación.

```
1 package modelo.myevents.ec.edu.ups;
2
3 import java.util.ArrayList;
4
5 // TODO: Auto-generated Javadoc
6 /**
7  * The Class Persona.
8  */
9
10 @Entity
11 @Table(name = "PERSONA")
12 public class Persona {
13
14     /** The id. */
15     @Id
16     @Column(name = "per_id")
17     @GeneratedValue(strategy = GenerationType.SEQUENCE)
18     private int id;
19
20     /** The nombre. */
21     @Column(name = "per_nombre")
22     @NotBlank(message = "Por favor ingrese el nombre")
23     private String nombre;
24
25     /** The apellido. */
26     @Column(name = "per_apellido")
27     @NotBlank(message = "Por favor ingrese el apellido")
28     private String apellido;
29
30     /** The cedula. */
31     @Column(name = "per_cedula")
32     @Pattern(regexp = "[\\s]*[0-9]*[1-9]*", message = "Solo debe ingresar numeros")
33     @NotBlank(message = "Por favor ingrese la cedula")
34     private String cedula;
35
36     /** The correo. */
37     @Column(name = "per_correo")
38     @NotBlank(message = "Por favor ingrese el correo")
39     private String correo;
40
41     /** The perfil. */
42     @Column(name = "per_perfil")
43     private String perfil;
44
45     /** The contrasenia. */
46     @Column(name = "per_contrasenia")
47     @Size(min = 4, message = "Debe ingresar un minimo de 4 caracteres")
48     private String contrasenia;
49
50     /** The estado. */
51     @Column(name = "per_estado")
52     private String estado;
53
54     /** The aeventos. */
55     /**Listado Tipo Asistencia Eventos, unidireccional
56      * */
57     @OneToMany(cascade={javax.persistence.CascadeType.ALL}, fetch=FetchType.LAZY)
58     @JoinColumn(name="per_aev_fk", referencedColumnName="per_id")
59     private List<AsistenciaEvento> aeventos =new ArrayList<>();
60
61     /**Listado Tipo Locales, unidireccional
62      * */
63
64     /** The locales. */
65     /**Listado Tipo Locales, unidireccional
66      * */
67     @OneToMany(cascade={javax.persistence.CascadeType.ALL}, fetch=FetchType.EAGER)
68     @JoinColumn(name="per_loc_fk", referencedColumnName="per_id")
69     private List<Local> locales=new ArrayList<>();
70
71     /** The srecepciones. */
72     /**Listado Tipo Salon Recepciones, unidireccional
73      * */
74     @OneToMany(cascade={javax.persistence.CascadeType.ALL}, fetch=FetchType.LAZY)
75     @JoinColumn(name="per_sal_fk", referencedColumnName="per_id")
76     private List<SalonRecepcion> srecepciones=new ArrayList<>();
77
78     /**
79      * Gets the id.
80      *
81      * @return the id
82      */
83     public int getId() {
84         return id;
85     }
86 }
```

4 📁 > modelo.myevents.ec.edu.ups

- ▶ 📄 AsistenciaEvento.java
- ▶ 📄 > Categoria.java
- ▶ 📄 CategoriaEventos.java
- ▶ 📄 > Evento.java
- ▶ 📄 Extra.java
- ▶ 📄 > Local.java
- ▶ 📄 Persona.java
- ▶ 📄 PersonaLocalEvento.java
- ▶ 📄 SalonRecepcion.java


```

4 📁 > controller.myevents.ec.edu.ups
  ▶ 📄 AsistenciaEventoController.java
  ▶ 📄 > CategoriaController.java
  ▶ 📄 > EventoController.java
  ▶ 📄 ExtraController.java
  ▶ 📄 > LocalController.java
  ▶ 📄 > PersonaController.java
  ▶ 📄 SalonRecepcionController.java

```

```

1 package controller.myevents.ec.edu.ups;
2
3 import java.io.IOException;
4
5
6
7 // TODO: Auto-generated Javadoc
8 /**
9  * The Class PersonaController.
10  */
11 @ManagedBean
12 @SessionScoped
13 public class PersonaController {
14
15     /** The log. */
16     @Inject
17     private Logger log;
18
19     /** The personas. */
20     private Persona personas = null;
21
22     /** The v. */
23     private Validacion v;
24
25     /** The id. */
26     private int id;
27
28     /** The pactual. */
29     private String pactual;
30
31     /** The id edit user. */
32     private int idEditUser;
33
34     /** The contrasenia. */
35     @NotBlank(message = "Ingrese las contrasenas")
36     private String contrasenia;
37
38     /** The conincidencia. */
39     private String conincidencia;
40
41     /** The Loginexiste. */
42     private String Loginexiste;
43
44     /**
45      * The nusuario. Variables donde se almacena los valores de la consulta
46      * maestro-detalles
47      */
48     private String nusuario;
49
50     /** The nlocal. */
51     private String nlocal;
52
53     /** The ndescripcion. */
54     private String ndescripcion;
55
56     /** The ncapacidad. */
57     private String ncapacidad;
58
59     /** The ncosto. */
60     private String ncosto;
61
62     /** The idrecuprerar. */
63     private int idrecuprerar;
64
65     /**PARA OBTENER EL LISTADO PERSONA y el listado de todos los objetos definidos en PERSONA, LOCAL, EVENTO
66     */
67     /** The List per ID. */
68     private List<Persona> ListPerID;
69
70     /** The. */
71     private List<PersonaLocalEvento> plelist;
72
73     /** The pdao. */
74     @Inject
75     private PersonaDAO pdao;
76
77
78
79
80
81
82
83
84
85
86 /**
87  * Metodo para inicializar los metodos e instancias.
88  */
89 @PostConstruct
90 public void init() {
91     personas = new Persona();
92     lpersonas = listaPersonas();
93     ListPerID = new ArrayList<Persona>();
94     plelist = new ArrayList<PersonaLocalEvento>();
95     v = new Validacion();
96     consultaLocalEventos();
97 }
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113 /**
114  *
115  */
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130

```

```

488  /**
489   * Coincidir contrasenia. Comparacion de los 2 campos referentes a la
490   * contrasenia, devolucion(true/false), segun sea la cedula valida o no valida
491   * respectivamente.
492   *
493   * @return true, if successful
494   */
495
496  public boolean coincidirContrasenia() {
497      if (personas.getContrasenia().equals(this.contrasenia)) {
498          return true;
499      } else {
500          return false;
501      }
502  }
503
504  /**
505   * Inicializar. Setea las variable como vacias, ocupado al momento de haber
506   * creado el usuario y dejar los h:inputText del JSF en blanco
507   */
508
509  public void inicializar() {
510      personas.setCedula("");
511      personas.setApellido("");
512      personas.setNombre("");
513      personas.setContrasenia("");
514  }
515

```

```

450
451  /**
452   * /**
453   * Crear. Creacion del
454   * validacion
455   */
456
457  public void crear() {
458      if (coincidirContra
459          if (v.validarCe
460              if (v.valid
461                  // SI E
462                  if (pda
463                      if
464
465
466
467
468
469
470
471
472
473
474      } else {
475          thi
476      }
477      } else {
478          thi.co
479      }
480      } else {
481          System.out.
482          this.coninc
483      }
484      } else {
485          this.coninciden
486      }
487

```

```

516  /**
517   * Modificar. Modificacion de los objetos de tipo Persona(USUARIO/ADMIN)
518   *
519   * @return URL para la navegabilidad a traves de JSF
520   * mainUser: pagina para el usuario
521   * mainAdmin: pagina para el adminstrador del local
522   * pages-black: pagina para el superUsuario
523   */
524
525  public String modificar() {
526      try {
527          System.out.println(personas.getPerfil());
528          if (myUser.getPerfil().equals("USUARIO")) {
529              personas.setContrasenia(pactual);
530              pdao.updatePersona(personas);
531              return "mainUser";
532          } else if (myUser.getPerfil().equals("ADMIN")) {
533              personas.setContrasenia(pactual);
534              System.out.println("ACTUALIZAR ADMIN : " + personas.getCedula());
535              System.out.println("ELSE IF ADMIN");
536              pdao.updatePersona(personas);
537              return "mainAdmin";
538
539          } else if (myUser.getPerfil().equals("ADMIN-SUPER")) {
540              personas.setContrasenia(pactual);
541              System.out.println("ACTUALIZAR ADMIN : " + personas.getCedula());
542              System.out.println("ELSE IF ADMIN");
543              pdao.updatePersona(personas);
544              return "pages-black";
545
546          }
547          return null;
548      } catch (Exception e) {
549          // TODO: handle exception
550          e.printStackTrace();
551          return null;
552      }
553  }
554
555  /**
556   * Metodo Leer. Dirije a el archivo crearPersona, dado como parametro un Id
557   *
558   * @param id int id
559   *
560   * @return URL de navegabilidad para crearPersona en JSF
561   */
562
563  public String leer(int id) {
564      personas = pdao.selectPersona(id);
565      return "crearPersona";
566  }
567
568  /**
569   * Eliminar. Metodo eliminar, llama al metodo Delete de PersonaDAO, parametro
570   * Id, para eliminar un registro especifico
571   *
572   * @param id int id
573   *
574   * @return URL de direccionalidad en JSF
575   */
576
577  public String eliminar(int id) {
578      pdao.deletePersona(id);
579      System.out.println("Eliminado Usuario .." + personas);
580      return "actualizar";
581  }
582
583  /**
584   * Lista personas, devuelve un objeto Listado de tipo Persona(Devuelve todas las
585   * personas).
586   *
587   * @return list de personas que existen en esa entidad
588   */
589
590  public List<Persona> listaPersonas() {
591      lpersonas = pdao.listPersonas();
592      return lpersonas;
593  }

```

```

595  /**
596   * Load datos editar.
597   *
598   * @param id int id
599   * Obtener las personas segun su rol y cargar la pagina.
600   *
601   * @return URL de navegabilidad
602   */
603  public String loadDatosEditar(int id) {
604      System.out.println("Cargando...Persona a Editar" + id);
605      personas = pdao.selectPersona(id);
606      pactual = personas.getContrasenia();
607      if (personas.getPerfil().equals("USUARIO")) {
608          return "recuperaPersona";
609      } else if (personas.getPerfil().equals("ADMIN")) {
610          System.out.println("LOAD DATOS ");
611          return "editAdmin";
612      }
613      return null;
614  }

615  /**
616   * IniciarSesion inicializar una Sesion HTTP y establecimiento de parametros en
617   * session, FacesContext acceso tanto al contexto de JSF como HTTP.
618   */
619  public void iniciarSesion() {
620      if (pdao.login(personas.getCorreo(), personas.getContrasenia()).size() != 0) {
621          HttpSession session = SessionUtils.getSession();
622          session.setAttribute("username",
623              pdao.login(personas.getCorreo(), personas.getContrasenia()).get(0).getCorreo());
624          session.setAttribute("perfil",
625              pdao.login(personas.getCorreo(), personas.getContrasenia()).get(0).getPerfil());
626          session.setAttribute("estado",
627              pdao.login(personas.getCorreo(), personas.getContrasenia()).get(0).getEstado());
628          this.Loginexiste = " ";
629          FacesContext context = FacesContext.getCurrentInstance();
630          if (pdao.login(personas.getCorreo(), personas.getContrasenia()).get(0).getPerfil().equals("USUARIO")) {
631              System.out.println("CONTEXTO USER");
632              try {
633                  context.getExternalContext().redirect("mainUser.xhtml");
634              } catch (IOException e) {
635                  // TODO Auto-generated catch block
636                  e.printStackTrace();
637              }
638          } else if (pdao.login(personas.getCorreo(), personas.getContrasenia()).get(0).getPerfil().
639              equals("ADMIN-SUPER")) {
640              // FacesContext contextAS= FacesContext.getCurrentInstance();
641              try {
642                  context.getExternalContext().redirect("pages-blank.xhtml");
643              } catch (IOException e) {
644                  // TODO Auto-generated catch block
645                  e.printStackTrace();
646              }
647          } else if (pdao.login(personas.getCorreo(), personas.getContrasenia()).get(0).getPerfil().equals("ADMIN")) {
648              // FacesContext contextAS= FacesContext.getCurrentInstance();
649              System.out.println("CONTEXTO ADMINNN");
650              try {
651                  context.getExternalContext().redirect("mainAdmin.xhtml");
652              } catch (IOException e) {
653                  // TODO Auto-generated catch block
654                  e.printStackTrace();
655              }
656          }
657      }
658  }

659  }

660  }

661  personas.setCorreo("");
662  personas.setContrasenia("");
663  this.Loginexiste = "El usuario o la contrasenia son incorrectos";
664  }

665  /**
666   * Cargar datos usuario obtenidos en session(HTTP) y su respectiva validacion.
667   */
668  public void cargarDatosUsuario() {
669      myUser = new Persona();
670      HttpSession session = SessionUtils.getSession();
671      String nus = (String) session.getAttribute("username");
672      System.out.println("NUS " + nus);
673      try {
674          if (pdao.verificaCorreo(nus).size() != 0) {
675              List<Persona> lusuario = new ArrayList<Persona>();
676              lusuario = pdao.verificaCorreo(nus);
677              myUser = lusuario.get(0);
678              System.out.println("MYUSER EMAIL: " + myUser.getCorreo());
679          } else {
680              FacesContext context = FacesContext.getCurrentInstance();
681              try {
682                  context.getExternalContext().redirect("index.xhtml");
683              } catch (IOException e) {
684                  // TODO Auto-generated catch block
685                  e.printStackTrace();
686              }
687          }
688      } catch (Exception e) {
689          System.out.println("Error al cargar");
690          e.printStackTrace();
691      }
692  }
693  }
694  }
695  }

```

```

697@ /**
698 * Cerrar sesion. Metodo Utilizado para la eliminacion de una sesion HTTP, con
699 * su respectiva navegacion
700 *
701 * @return URL de direccionalidad para cerrar la sesion
702 */
703
704@ public String cerrarSesion() {
705     HttpSession session = SessionUtils.getSession();
706     session.invalidate();
707     return "index.xhtml";
708 }
709
710@ /**
711 * Verifica sesion.
712 * Metodo para comparar la sesion entre los diferentes usuarios en la aplicacion web
713 */
714
715@ public void verificaSesion() {
716     HttpSession session = SessionUtils.getSession();
717     String nusv = (String) session.getAttribute("username");
718     if (nusv != null) {
719         System.out.println("si tiene sesion");
720         FacesContext contex = FacesContext.getCurrentInstance();
721         try {
722             if (myUser.getPerfil().equals("USUARIO")) {
723                 contex.getExternalContext().redirect("mainUser.html");
724             } else if (myUser.getPerfil().equals("ADMIN")) {
725                 contex.getExternalContext().redirect("mainAdmin.html");
726             }
727         } catch (IOException e) {
728             // TODO Auto-generated catch block
729             e.printStackTrace();
730         }
731     }
732 }
733
734
735@ /**
736 * Consulta local eventos.
737 * Recorre las entidades persona, locales, eventos, para recuperar el local con el evento
738 * de la persona que ingresa al sistema
739 *
740 * @return URL null para cargar los datos en la misma pagina en JSF
741 */
742
743@ public String consultaLocalEventos() {
744     plelist.clear();
745     System.out.println("CONSULTA LOCAL EVENTO ID: " + idrecuperar + " " + "ENTRA");
746     ListPerID = pdao.listPersonaID(idrecuperar);
747
748     /** PRIMER FOR: Recupero todo el Objeto Persona, Local, Evento */
749     for (Persona p : ListPerID) {
750         /** Segundo FOR: a partir de 'p', obtengo el objeto evento */
751         for (Local l : p.getLocales()) {
752             PersonaLocalEvento ple = new PersonaLocalEvento();
753             ple.setL_nombre(l.getNombre());
754             ple.setL_descripcion(l.getDescripcion());
755             ple.setL_capacidad(l.getCapacidad());
756             ple.setL_costo(l.getCosto());
757             /** Condicionamiento cuando no hay ningun evento, agrego campos pro default */
758             if (l.getEvento().isEmpty()) {
759                 System.out.println("LOCAL SIN EVENTOS");
760                 ple.setE_nombre("Ningun Registro");
761                 ple.setE_descripcion(" ");
762                 ple.setE_fecha(" ");
763             } else {
764                 /** Tercer FOR: a partir de 'l' obtengo el objeto Evento */
765                 for (Evento ev : l.getEvento()) {
766                     /**
767                      * Declaro un objeto para setear cada uno de los objeto y finalmente agregarlos
768                      * en la lista respectiva
769                      */
770
771                     ple.setE_nombre(ev.getNombre());
772                     ple.setE_descripcion(ev.getDescripcion());
773                     ple.setE_fecha(ev.getFechaEvento().toString());
774                 }
775             }
776             /** Anadiendo cada uno de los objetos a la lista */
777             plelist.add(ple);
778         }
779     }
780     return null;
781 }
782
783@ }

```

4  > dao.myevents.ec.edu.ups
 ▷  AsistenciaEventoDAO.java
 ▷  > CategoriaDAO.java
 ▷  EventoDAO.java
 ▷  ExtraDAO.java
 ▷  > LocalDAO.java
 ▷  > PersonaDAO.java
 ▷  SalonRecepcionDAO.java

```

17 @Stateless
18 public class PersonaDAO {
19
20     /** The em. */
21     @Inject
22     private EntityManager em;
23
24
25     /**
26      * Insert persona.
27      * Persiste la entidad persona haciendo uso del EntityManager
28      * @param p de persona
29      */
30
31     public void insertPersona(Persona p) {
32         em.persist(p);
33     }
34
35     /**
36      * Update persona.
37      * Actualiza la entidad persona haciendo uso del EntityManager
38      * @param p the persona
39      */
40
41     public void updatePersona(Persona p) {
42         System.out.println("Updating..." + p.getId() + p.getCorreo() + p.getContrasenia());
43         em.merge(p);
44     }
45
46     /**
47      * Select persona.
48      *
49      * @param id the id
50      *
51      * @return the persona
52      */
53     public Persona selectPersona(int id) {
54         Persona p = em.find(Persona.class, id);
55         return p;
56     }
57
58     /**
59      * Delete persona.
60      *
61      * @param id the id
62      */
63
64     public void deletePersona(int id) {
65         Persona p = selectPersona(id);
66         em.remove(p);
67     }
68
69     /**
70      * List personas.
71      *
72      * @return una coleccion del listado de personas
73      */
74
75     public List<Persona> listPersonas() {
76         String sql = "select p from Persona p";
77         TypedQuery<Persona> query = em.createQuery(sql, Persona.class);
78         List<Persona> lpersonas = query.getResultList();
79         for (Persona p : lpersonas) {
80             p.getAeventos().size();
81             p.getSrecepciones().size();
82         }
83         return lpersonas;
84     }
85
86     /**
87      * Guardar.
88      *
89      * @param p the p
90      */
91     public void guardar (Persona p) {
92         Persona aux = selectPersona(p.getId());
93         System.out.println("ID GUARDAR:" + p.getId());
94         if (aux != null) {
95             updatePersona(p);
96         } else {
97             System.out.println("Grabando!");
98             insertPersona(p);
99         }
100     }

```



```

102  /**
103   * Login.
104   *
105   * @param user the user
106   * @param pass the pass
107   * @return una coleccion personas para el logeo a traves del @param 1 y @param2
108   * @param1 user the user
109   * @param2 pass the pass
110   */
111  public List<Persona> login(String user, String pass) {
112      String sql = "Select p from Persona p WHERE p.correo = '"+user+"' AND p.contrasenia='"+pass+"'";
113      TypedQuery<Persona> query = em.createQuery(sql, Persona.class);
114      List<Persona> personas = query.getResultList();
115      for(Persona p : personas) {
116          p.getAeventos().size();
117          p.getSrecepciones().size();
118      }
119      return personas;
120  }
121
122  /**
123   * Verifica correo.
124   *
125   * @param user the user
126   * @return una coleccion de personas para verificar el correo
127   */
128  public List<Persona> verificaCorreo(String user)
129  {
130      String sql="Select p from Persona p WHERE p.correo = '"+user+"'";
131      TypedQuery<Persona> query=em.createQuery(sql,Persona.class);
132      List<Persona>personas=query.getResultList();
133      for(Persona p : personas) {
134          p.getAeventos().size();
135          p.getSrecepciones().size();
136      }
137      return personas;
138  }
139
140  /**
141   * Existe cedula.
142   *
143   * @param cedula the cedula
144   * @return una lista de las personas con la cedula
145   */
146  public List<Persona> existeCedula(String cedula) {
147      String jpql = "Select p from Persona p WHERE p.cedula = '"+cedula+"'";
148      TypedQuery<Persona> query = em.createQuery(jpql, Persona.class);
149      List<Persona> personas = query.getResultList();
150      for(Persona p : personas) {
151          p.getAeventos().size();
152          p.getSrecepciones().size();
153      }
154      return personas;
155  }
156
157
158  /**
159   * List persona ID (Recuperar lista de personas por el id enviado).
160   *
161   * @param id the id
162   * @return una coleccion de listado de personas
163   */
164  public List<Persona> listPersonaID(int id){
165      String jpql = "Select p from Persona p WHERE p.id = '"+id+"' ";
166      TypedQuery<Persona> query = em.createQuery(jpql, Persona.class);
167      List<Persona>personas= query.getResultList();
168      System.out.println("LISTPERSID DAO");
169      for(Persona p : personas) {
170          p.getLocales().size();
171          p.getAeventos().size();
172          /*Comprobar para los eventos que se dan en dicho local*/
173          if(!p.getLocales().isEmpty()) {
174              p.getLocales().get(0).getEvento().size();
175          }
176      }
177      return personas;
178  }

```



```

> servicios.myevents.ec.edu.ups
  > Respuesta.java
  > RestApplication.java
  > UsuariosWSRest.java

```

```

1 package servicios.myevents.ec.edu.ups;
2
3 import java.util.List;
4
16
17 @Path("usuarios")
18 public class UsuariosWSRest {
19
20     /** The pdao. */
21     @Inject
22     private PersonDAO pdao;
23
24     /** The locdao. */
25     @Inject
26     private LocalDAO locdao;
27
28     /**
29      * WS Lista persona.
30      * Mostrar el listado de usuarios
31      * @return the list
32      * @path llamdo desde la url
33      * @Produces tipo de datos JSON ha ser enviados al cliente
34      */
35
36     @GET
37     @Path("listado-users")
38     @Produces("application/json")
39     public List<Persona> listaPersona() {
40         return pdao.listPersonas();
41     }
42
43
44     /**
45      * WS Grabar persona.
46      *
47      * @param p the p
48      * @return the respuesta que contiene codigos para enviar mensajes
49      *         definido en la clase Respuesta.java
50      * @Produces tipo de datos JSON ha ser enviados al cliente
51      * @Consumes tipo de datos JSON que puede ser aceptada o consumida por el usuario
52      */
53
54     @POST
55     @Path("crear-usuarios")
56     @Produces("application/json")
57     @Consumes("application/json")
58     public Respuesta GrabarPersona(Persona p) {
59         Respuesta r = new Respuesta();
60         try {
61             p.setPerfil("USUARIO");
62             p.setEstado("A");
63             pdao.guardar(p);
64             r.setCodigo(1);
65             r.setMensaje("Grabado Exitosamente");
66             return r;
67         } catch (Exception e) {
68             e.printStackTrace();
69             r.setCodigo(99);
70             r.setMensaje("Error al grabar");
71             return r;
72         }
73     }
74
75
76     /**
77      * WS Login.
78      *
79      * @param p the p
80      * @return una el logeo de la persona enviando el correo y contrasenia para
81      *         comprobarla
82      */
83
84     @POST
85     @Path("login")
86     @Produces("application/json")
87     @Consumes("application/json")
88     public List<Persona> login(Persona p) {
89         return pdao.login(p.getCorreo(), p.getContrasenia());
90     }
91
92     /**
93      * Listlocal.
94      *
95      * @return the list de todos los locales en formato JSON
96      */
97     @GET
98     @Path("listado-locales")
99     @Produces("application/json")
100     public List<Local> listlocal() {
101         return locdao.listlocal();
102     }
103
104 }

```

4.

5. Implementación de las relaciones a nivel de Entidades y también en vista de manera adecuada

Visualización de las categorías disponibles (Usuario)

localhost:8080/MyEvents/faces/listadoCategoriaUsuario.xhtml

80%

...

☆

Bienvenido Carlos , a la pagina
mas cool de los eventos!
creado por MyEvents.

Categorías de Eventos

Editar Perfil

Categorías de Eventos

Reserva de Salones

Nombre	Descripcion	Visualizar Evento
cultural	descripcion cultural	<div>Ver Eventos</div>
deportivo	descripcion deportivo	<div>Ver Eventos</div>
social	decripcion de socil	<div>Ver Eventos</div>
rock	descripcion de rocker	<div>Ver Eventos</div>

Visualización de los Eventos según la categoría (Usuario)

localhost:8080/MyEvents/faces/listadoCategoriaUsuario.xhtml

80%

...

☆

Bienvenido Carlos , a la pagina
mas cool de los eventos!
creado por MyEvents.

Eventos cultural

Editar Perfil

Categorías de Eventos

Reserva de Salones

Evento	Costo	Fecha
La descripcion para el cumple del sobrinito	878.0	1992-01-08
EVENTO 5	0.03423	1992-01-08
EVENTO 6	0.03423	2017-01-07
CODIGO 2	738732.0	2017-01-08

Visualización de los locales con sus respectivos Eventos que se festejaron
(Administrador)

←→↻🏠

localhost:8080/MyEvents/faces/consulEventLocPers.xhtml?id=1

80%⋮🐦☆

🔍📄☰

Administrador ana guillen

Salir

Inicio

Editar Perfil
ana@gmail.com

Creacion Salones

Crear Eventos

Detalle de Locales-Evento

MyEvents / Consultas Salones - Eventos

Locales - Eventos

Nombre del Salon	Capacidad	Costo	Descripcion	Evento	Fecha	Descripcion
CABALLO CAMPANA	677	899	caba csao oasd caj	Ningun Registro		
El rosario	45	478	DESCRIPCION EL ROSARIO	ksjdffhsjdj	2017-01-07	EVENTO 6
La salsoteca	45	345	Descripcion de la salsoteca	Las bodas de mi abue	2017-01-08	EVENTO 7
El arenal	98	800	Descripcion del arenal	cumple de mi abue	2017-01-08	CODIGO 2

Principal

6. Funcionalidad y Validaciones.
- Validaciones a nivel de Entidad

```

Persona.java
26 @Entity
27 @Table(name = "PERSONA")
28 public class Persona {
29     |
30     /** The id. */
31     @Id
32     @Column(name = "per_id")
33     @GeneratedValue(strategy = GenerationType.SEQUENCE)
34     private int id;
35
36     /** The nombre. */
37     @Column(name = "per_nombre")
38     @NotBlank(message = "Por favor ingrese el nombre")
39     private String nombre;
40
41     /** The apellido. */
42     @Column(name = "per_apellido")
43     @NotBlank(message = "Por favor ingrese el apellido")
44     private String apellido;
45
46     /** The cedula. */
47     @Column(name = "per_cedula")
48     @Pattern(regexp = "[\\s]*[0-9]*[1-9]+",message="Solo debe ingresar numeros")
49     @NotBlank(message = "Por favor ingrese la cedula")
50     private String cedula;
51
52     /** The correo. */
53     @Column(name = "per_correo")
54     @NotBlank(message = "Por favor ingrese el correo")
55     private String correo;
56
57     /** The perfil. */
58     @Column(name = "per_perfil")
59     private String perfil;
60
61     /** The contrasenia. */
62     @Column(name = "per_contrasenia")
63     @Size(min = 4, message = "Debe ingresar un minimo de 4 caracteres")
64     private String contrasenia;
65
66     /** The estado. */
67     @Column(name = "per_estado")
68     private String estado;

```

- Validaciones a nivel de vista

REGISTRO DE USUARIO

Debera proporcionar los siguientes datos:

Al crear una cuenta usted esta aceptando todos los terminos y condiciones.

La cedula es incorrecta

REGISTRO DE USUARIO

Debera proporcionar los siguientes datos:

Al crear una cuenta usted esta aceptando todos los terminos y condiciones.

estefania

villacis

1900773381

estefany@gmail.com

.....

.....

Ingrese las mismas contrasenas

Registrar

REGISTRO DE USUARIO

Debera proporcionar los siguientes datos:

Al crear una cuenta usted esta aceptando todos los terminos y condiciones.

estefania

villacis

1900773381

estefany@gmail.com

.....

.....

La cedula ya se encuentra registrada

Registrar

Administrador ricardoADMIN pozo Salir

[Inicio](#)

[Editar Perfil](#)
pozoAdmin@gmail.com

[Creacion Salones](#)

[Crear Eventos](#)

[Detalle de Locales-Evento](#)

[MyEvents](#) / [Agregar Evento](#)

Ingreso del Evento

Info! Los campos que contienen * son obligatorios.

* Nombre del Evento: Ingrese el nombre del evento!

* Fecha del Evento:

* Costo del Evento Ingrese el costo del evento!

* Seleccione una categoria

* Descripcion:

Ingrese una descripcion!

Guardar

Cancelar

Uso de PrimeFaces para el calendario

Administrador ricardoADMIN pozo Salir

MyEvents / Agregar Evento

Ingreso del Evento

Info! Los campos que contienen * son obligatorios.

* Nombre del Evento:

* Fecha del Evento:

* Costo del Evento

* Seleccione una categoria

* Descri

Copyright © Your Website 2017

7. Diagrama.

Si se encuentra bien, y nos menciono que le recordemos para ponernos bien la calificación.
Gracias.

8. Funcionalidad Control de Excepciones.

```
/**
 * Verifica correo.
 *
 * @param user the user
 * @return the list
 */
public List<Persona> verificaCorreo(String user)
{
    try {
        String sql="Select p from Persona p WHERE p.correo = '"+user+"'";
        TypedQuery<Persona> query=em.createQuery(sql,Persona.class);
        List<Persona>personas=query.getResultList();
        for(Persona p : personas) {
            p.getAeventos().size();
            p.getSrecepciones().size();
        }
        return personas;
    } catch (Exception e) {
        System.out.println("correo ya se encuentra registrado");
    }
    return null;
}
```

```

public void guardar (Persona p) {
    try {
        Persona aux = selectPersona(p.getId());
        System.out.println("ID GUARDAR:" +p.getId());
        if(aux!=null) {
            updatePersona(p);
        }else {
            System.out.println("Grabando!");
            insertPersona(p);
        }
    } catch (Exception e) {
        // TODO: handle exception
        System.out.println("coincidencia de datos" +e.getMessage());
    }
}

```

Funcionamiento.

WildFly 10.x [JBoss Application Server Startup Configuration] C:\Program Files\Java\jdk1.8.0_60\bin\javaw.exe (2 ene. 2018 22:53:02)

```

22:58:57,919 INFO [stdout] (default task-49) locales0_.local_costo as local_co4_4_1_,
22:58:57,919 INFO [stdout] (default task-49) locales0_.local_descripcion as local_de5_4_1_,
22:58:57,919 INFO [stdout] (default task-49) locales0_.even_fotografia as even_fot6_4_1_,
22:58:57,919 INFO [stdout] (default task-49) locales0_.latitud as latitud7_4_1_,
22:58:57,919 INFO [stdout] (default task-49) locales0_.longitud as longitud8_4_1_,
22:58:57,919 INFO [stdout] (default task-49) locales0_.local_nombre as local_no9_4_1_,
22:58:57,919 INFO [stdout] (default task-49) locales0_.local_puntuacion as local_p10_4_1_
22:58:57,919 INFO [stdout] (default task-49) from
22:58:57,919 INFO [stdout] (default task-49) LOCAL locales0_
22:58:57,919 INFO [stdout] (default task-49) where
22:58:57,919 INFO [stdout] (default task-49) locales0_.per_loc_fk=?

22:58:57,922 INFO [stdout] (default task-49) Hibernate:
22:58:57,922 INFO [stdout] (default task-49) select
22:58:57,922 INFO [stdout] (default task-49) aeventos0_.per_aev_fk as per_aev_3_0_0_,
22:58:57,922 INFO [stdout] (default task-49) aeventos0_.aev_codigo as aev_codi1_0_0_,
22:58:57,922 INFO [stdout] (default task-49) aeventos0_.aev_codigo as aev_codi1_0_1_,
22:58:57,922 INFO [stdout] (default task-49) aeventos0_.aev_estado as aev_esta2_0_1_
22:58:57,922 INFO [stdout] (default task-49) from
22:58:57,923 INFO [stdout] (default task-49) ASISTENCIAEVENTO aeventos0_
22:58:57,923 INFO [stdout] (default task-49) where
22:58:57,923 INFO [stdout] (default task-49) aeventos0_.per_aev_fk=?

22:58:57,926 INFO [stdout] (default task-49) Hibernate:
22:58:57,926 INFO [stdout] (default task-49) select
22:58:57,926 INFO [stdout] (default task-49) srecepcion0_.per_sal_fk as per_sal_3_6_0_,
22:58:57,926 INFO [stdout] (default task-49) srecepcion0_.sare_id as sare_id1_6_0_,
22:58:57,926 INFO [stdout] (default task-49) srecepcion0_.sare_id as sare_id1_6_1_,
22:58:57,926 INFO [stdout] (default task-49) srecepcion0_.sare_estado as sare_est2_6_1_
22:58:57,926 INFO [stdout] (default task-49) from
22:58:57,926 INFO [stdout] (default task-49) SALONRECEPCION srecepcion0_
22:58:57,926 INFO [stdout] (default task-49) where
22:58:57,926 INFO [stdout] (default task-49) srecepcion0_.per_sal_fk=?

22:58:57,929 INFO [stdout] (default task-49) correo ya se encuentra registrado

```