Manual Técnico

Rafael Retamal Donoso

EasyInventory

Tecnologías Usadas

JavaServe Faces(Jsf) y Primeface

JSF es un framework MVC (Modelo-Vista-Controlador) basado en el API de Servlets que proporciona un conjunto de componentes en forma de etiquetas definidas en páginas XHTML mediante el framework Facelets. Facelets se define en la especificación 2 de JSF como un elemento fundamental de JSF que proporciona características de plantillas y de creación de componentes compuestos. Antes de la especificación actual se utilizaba JSP para componer las páginas JSF.

Uso JSF para hacer el registro y el login, tiro de dos beans Usuario y Prestatario , usuario guardia el bean de Prestatario dentro de si mismo todos los datos. En usuario ahí dos métodos add donde registro el usuario y hago la comprobaciones de si existe usuario y llamo a login , este sirve para dos cosas para después del registro añadir a usuario a la sesión o para entrar de nuevo a la aplicación.

Primeface básicamente lo uso para dar estilo para la comprobación de dos contraseñas, dar estilos un poco más moderno a jsf dentro de lo que se pudo. También compruebo si la contraseña es débil, media o fuerte.

Java,Jsp,Servlet, Gson

La aplicación en servidor uso el lenguaje Java , para que se pueda ejecutar en varios dispositivos por su maquina virtual.
Uso JavaServerPages para hacer la páginas web dinámicas todas mis vistas son jsp menos el index porque uso Jsf, me ayudo también de angular para darle más dinamismo.
Servlet es una clase de Java de la cual me ayudo para ampliar las capacidades del servidor, estos responden a las solicitudes desde el cliente donde hago la llamada con ajax , jquery y angular lo explicare más abajo.

A los servlet siempre me llegaran un parámetro que será un json, que después con la librería Gson lo transformaremos directamente a uno de los bean, también responderemos siempre json o persistiremos en la base de datos.

Por ejemplo en el controlador de equipo:

```
 Gson gson = new Gson();
        DAOFactory daof = DAOFactory.getDAOFactory();
        IGenericoDAO gdao = daof.getGenericoDAO();
        //En este apartado añadiremos nuevos equipos
        if (request.getParameter("equipo") != null) {
            //Traaformamos el parametro en un bean equipo y lo añadimos a la bbdd
            String json = request.getParameter("equipo");
            Equipo equipo = gson.fromJson(json, Equipo.class);
            gdao.add(equipo);

        }
        //En este parametro nos traeremos todos los equipo de la bbdd y lo devolveremos en
un json
        else if (request.getParameter("getEquipos") != null) {
            ArrayList<Equipo> lista = new ArrayList();
            lista = (ArrayList<Equipo>) gdao.get("Equipo");
            String representacionJSON = gson.toJson(lista);
            response.getWriter().write(representacionJSON);

        }
        // En este apartado le pasamos un equipo y lo borramos de la base de datos
        else if (request.getParameter("borrarEquipo") != null) {
            String json = request.getParameter("borrarEquipo");
            Equipo equipo = gson.fromJson(json, Equipo.class);
            equipo.setMarca(null);
            equipo.setCategoria(null);
            gdao.delete(equipo);
        }
        //Aqui devolvemos todos los equipos que esten libres
        else if (request.getParameter("getEquiposLibres") != null) {
            ArrayList<Equipo> lista = new ArrayList();
            lista = (ArrayList<Equipo>) gdao.get("Equipo where idPrestatario=null");
            String representacionJSON = gson.toJson(lista);
            response.getWriter().write(representacionJSON);

        }
```

Jquery,Angular,Ajax.

Uso jquery para que me sea mucho más fácil acceder al árbol del DOM y la funcionalidad que trae de base para usar Ajax, uso ajax para no tener que recargar la página por ejemplo:

```
$.ajax({
            data: {"getMiUsuario": "toda"},
            async: false,
            url: '../../ControlUsuarioAdmin',
            type: 'post',
            dataType: "json",
```

```
        beforeSend: function () {

        },
        success: function (response) {
            // alert("Inicialidad:"+$scope.marcas);
            $scope.miUsuario = response;
            // alert(response);
            Command: toastr["success"]("Se han cargado tu usuario correctamente!");

        }, error: function (jqXHR, textStatus, errorThrown) {
            Command: toastr["error"]("No se ha podido cargar tu usuario!");

        }
    });
```

Uso angular porque es una solución completa que incluye prácticamente todos los aspectos que puedes necesitar para crear una aplicación cliente en javascript. Esto incluye la generación de vistas,el uso de databinding, las rutas, la organización de modulo y la comunicación con el servidor.

Ejemplo de rooting :

```
app.config(function ($routeProvider) {
    $routeProvider
        .when("/", {
            templateUrl: "vista/main.jsp"
        })
        .when("/usuario", {
            templateUrl: "vista/usuario.jsp",
            controller: "usuarioCtrl"
        })
        .when("/equipo", {
            templateUrl: "vista/equipo.jsp",
            controller: "equipoCtrl"
        })
        .when("/incidencia", {
            templateUrl: "vista/incidencias.jsp",
            controller: "incidenciasCtrl"
        })
        .when("/categoria", {
            templateUrl: "vista/categoria.jsp",
            controller: "categoriaCtrl"
        })
        .when("/marca", {
            templateUrl: "vista/marca.jsp",
            controller: "marcaCtrl"
        });
});
```

Ejemplo de organización de modulos:

```
var app = angular.module("myApp", ["ngRoute"]);
```

Ejemplo de comunicación con el servidor:

```
 $scope.getMiUsuario = function () {
        $.ajax({
            data: {"getMiUsuario": "toda"},
            async: false,
            url: '../../ControlUsuarioAdmin',
            type: 'post',
            dataType: "json",
            beforeSend: function () {

            },
            success: function (response) {
                // alert("Inicialidad:"+$scope.marcas);
                $scope.miUsuario = response;
                // alert(response);
                Command: toastr["success"]("Se han cargado tu usuario correctamente!");

            }, error: function (jqXHR, textStatus, errorThrown) {
                Command: toastr["error"]("No se ha podido cargar tu usuario!");

            }
        });
    };
```

JsPDF,autotable,DataTable,Toastr

Con jsPDF creo los pdf solo tirando de la parte de cliente por ejemplo con ayuda de jquery hago el filtrado de la tabla que se este mostrando ahora mismo y la imprimo en un pdf por ejemplo:

```
    var doc = new jsPDF('landscape');
        var elem = $('#example')[0];
        var res = doc.autoTableHtmlToJson(elem);
        doc.autoTable(res.columns, res.data);
        doc.output("dataurlnewwindow");
```

Uso DataTable para crear tablas dinámicas con filtrados y ordenación con varias posibilidades investigué por Internet para ponerlo en español tal que así:

```
        $('#example').DataTable({
            "language": {
                "sProcessing": "Procesando...",
                "sLengthMenu": "Mostrar _MENU_ registros",
                "sZeroRecords": "No se encontraron resultados",
                "sEmptyTable": "Ningún dato disponible en esta tabla",
                  "sInfo": "Mostrando registros del _START_ al _END_ de un total de _TOTAL_
registros",
                "sInfoEmpty": "Mostrando registros del 0 al 0 de un total de 0 registros",
                "sInfoFiltered": "(filtrado de un total de _MAX_ registros)",
```

```
            "sInfoPostFix": "",
            "sSearch": "Buscar:",
            "sUrl": "",
            "sInfoThousands": ",",
            "sLoadingRecords": "Cargando...",
            "oPaginate": {
                "sFirst": "Primero",
                "sLast": "Ãšltimo",
                "sNext": "Siguiente",
                "sPrevious": "Anterior"
            },
            "oAria": {
                "sSortAscending": ": Activar para ordenar la columna de manera ascendente",
                        "sSortDescending": ": Activar para ordenar la columna de manera descendente"
            }

        }});
```

(Lo suyo viese sido poner el json fuera y cargarlo siempre con un par de lineas pero 40 horas no dan para mucho).

Toastr para hacer alertas dinámicas con varias configuraciones yo uso esta:

```
toastr.options = {
    "closeButton": false,
    "debug": false,
    "newestOnTop": false,
    "progressBar": true,
    "positionClass": "toast-top-right",
    "preventDuplicates": false,
    "onclick": null,
    "showDuration": "300",
    "hideDuration": "1000",
    "timeOut": "5000",
    "extendedTimeOut": "1000",
    "showEasing": "swing",
    "hideEasing": "linear",
    "showMethod": "fadeIn",
    "hideMethod": "fadeOut"
};
```

Hibernate

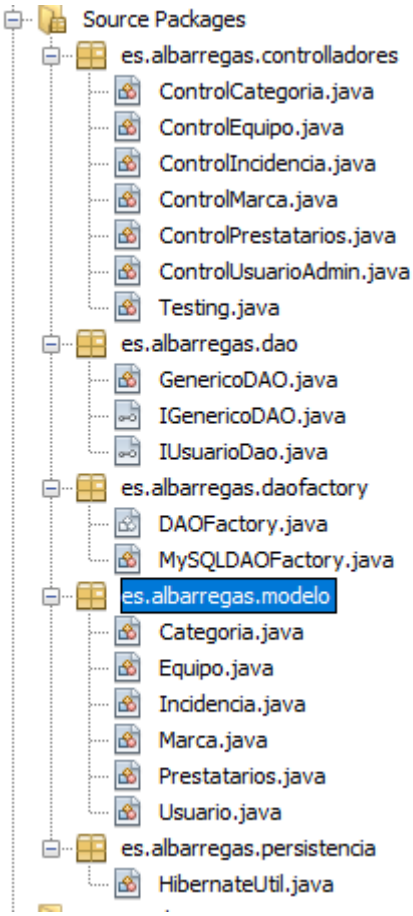
Esta es la tecnología que me viese gustado ampliarla y estudiarla más pero por falta de tiempo solo he hecho un par de consulta en xql y unos cuantos filtrados . Pero me ayudado mucho por que mi aplicación tiene siete crud y gracias al dao generico no tenido que usar otros daos.
Ejemplo de sentencia xql:

gdao.get("Equipo where idPrestatario=null");

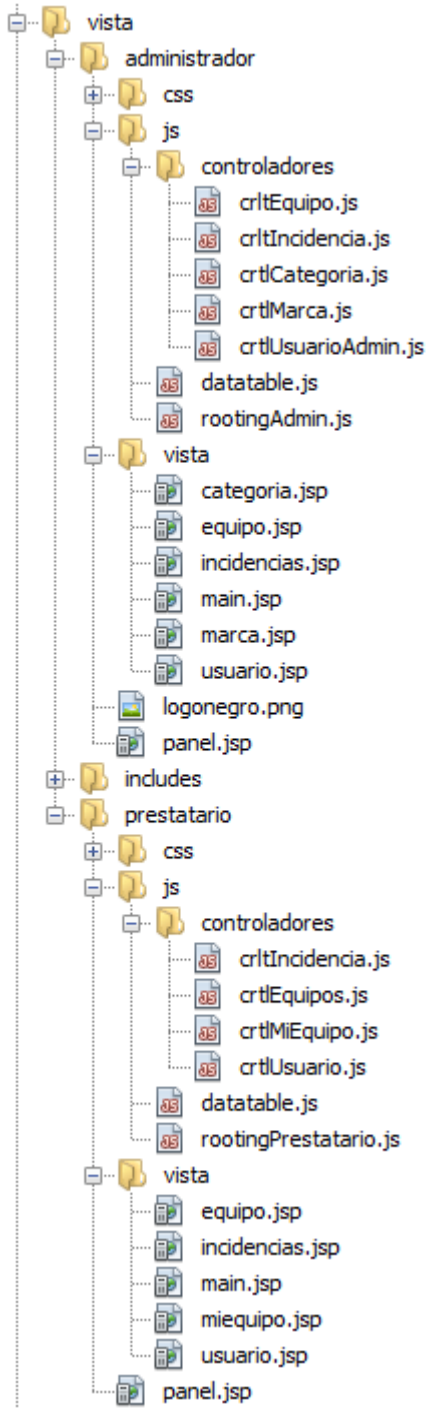
También he hecho configuraciones varias para añadir poder borrar o no dependiendo de las relaciones o de lo que quería como por ejemplo no borrar una categoría que tuviera un equipo , pero poder añadir un usuario y un prestatario a vez por ejemplo:

@OneToOne(cascade = {CascadeType.PERSIST})
@JoinColumn(name = "idCategoria")
Para poder persistir dos objeto a la vez y borrarlo también tirando de usuario las dos.
 @OneToOne(cascade = {CascadeType.ALL})
  @JoinColumn(name = "idPrestatarios")

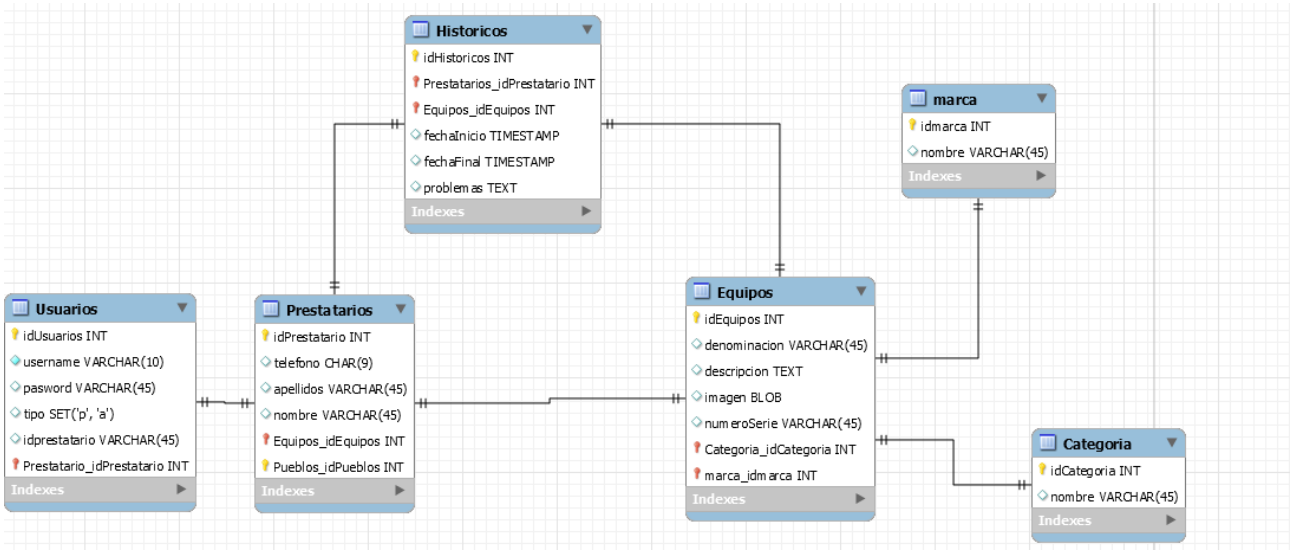
Estructura de Java del Proyecto



Estructura del cliente del Proyecto



Estructura de la Base de datos



Despliegue de la aplicación.

Copiamos el archivo al directorio webapps

```
sudo cp rafaelretamaldonoso-1.0shanpsot.war $CATALINA_BASE/webapps
```

En caso de ser necesario, añadimos usuarios y roles para la aplicación

```
sudo nano $CATALINA_BASE/conf/tomcat-users.xml
```

Reiniciamos Tomcat

```
sudo /etc/init.d/tomcat restart
```

Guías de estilos de Rafael Retamal Donoso

1.Fondos

Cabecera: El color de la cabecera que he cogido es el siguiente código hexadecimal #F2F1F1 y código RGB R(242)G(241)B(241). El color es un poco grisacio ya que concuerda bien con el logo del dj de música que cogido ya que el logo es negro. No lo pongo en blanco por que es un contraste muy grande respecto el negro y puede afectar a la vista.

Cuerpo: El color del cuerpo que he cogido es el siguiente código hexadecimal #120150 y código RGB R(18)G(1)B(80). El color es un azul oscuro, e cogido este por que da una sensacion de juventud a la pagina y de moderna ya que el el grupo de músico que e cogido es un dj de música electrónica.

Pie: El color del cuerpo que he cogido es el siguiente código hexadecimal #0E013C y código RGB R(14)G(1)B(60). El color de fondo de pie que he elegido es un azul algo mas oscuro que el cuerpo para que se distinga el pie del cuerpo y siga en concordancia de los colores básicos que escogí al principio.

2.Barra de navegación

La barra de navegación va a tener el mismo color que la cabecera para que parezca que esta integrada en ella a si que el colo elegido es el siguiente código hexadecimal #F2F1F1 y código RGB R(242)G(241)B(241) . Lo que vamos a cambiar del estilo de la cabecera es que la tipografía va a cambiar a un color grisacio oscuro tal que código hexadecimal #9A98A1 y código RGB R(154)G(152)B(161) este color lo va a tener solo cuando el ratón no este por encima o no este elegido esa página, cuando el ratón este hover o la página esta elegida sera blanco código hexadecimal #FFFFFF y código RGB R(255)G(255)B(255).

3.Tipografía

<h1>,<h2>,<h3>: El color que le vamos a dar a las etiquetas <h> es el blanco normal código hexadecimal #FFFFFF y código RGB R(255)G(255)B(255) los demás estilo estarán por defecto.

<p>: La tipografía que he elegido para la etiqueta <p> es la de por defecto cambiando que el color que cogido para esta etiqueta en el cuerpo es tal que código hexadecimal #F2F1F1 y código RGB R(242)G(241)B(241). Line heigth el espacio entre lineas y lineas le puesto de 1.6 para que no quede como un libro. La etiqueta <p> solo cambia de estilo en el pie que le puesto un color mas oscuro como el de la barra de navegación código hexadecimal #9A98A1 y código RGB R(154)G(152)B(161)