14-12-2017

WEB BUSCAR DESTINO

Proyecto de Fin de Curso

PATRICIA MARTÍNEZ LUCENA
2º DAW

Contenido

IMPLEMENTAR UNA WEB-BUSCA-DESTINO

ALUMNO: PATRICIA MARTÍNEZ LUCENA

CURSO: 2º DAW

EMAIL: patrimartinez31@hotmail.com

OBJETIVOS

El objetivo principal de la aplicación consiste en que un usuario puede realizar **búsquedas de trayectos** en coche entre dos lugares de **España**, sin incluir Portugal (que no es de España), Ceuta, Melilla ni las islas Baleares y Canarias, ya que el trayecto solo se realiza en coche.

Los **lugares de búsqueda** pueden ser comunidades autónomas, ciudades, pueblos, calles (no se permiten números) o algún edificio o tienda, como realiza Google Maps.

Existen varios mecanismos de **control de datos:**

- La aplicación intentará coger el valor del **código postal** del lugar de llegada y mostrará la trayectoria en el mapa y los datos meteorológicos.
- Si no se muestra el código postal, la aplicación cogerá el **valor** de dicha **ciudad** o pueblo y mostrará la trayectoria en el mapa y los datos meteorológicos.
- Si el lugar de **salida** y el lugar de **llegada es el mismo**, se mostrará el punto en el mapa y los datos meteorológicos del lugar, pero sin distancia ni duración.
- Si la aplicación **no encuentra la ruta**, mostrará diferentes mensajes de error dependiendo del tipo de error que aparezca (no encontrado, sin acceso a la URL, si la búsqueda se realiza fuera de España…).

La aplicación también dispone de una opción de **“evitar autopistas”**, para calcular la ruta sin tener que pagar peaje y el camino será distinto y más largo.

El usuario puede realizar las búsquedas ya sea **registrándose** para almacenarlas y poder verlas en otro momento, **o sin registrarse**.

En ambos casos, la aplicación mostrará la misma **tabla informativa**: lugar de salida, lugar de llegada, fecha y hora de realización de la búsqueda, duración del trayecto en coche, distancia entre los dos lugares y tiempo atmosférico que hará en el lugar de llegada y a la hora de llegada (temperatura, nubes, probabilidad de precipitación, probabilidad de lluvia, porcentaje de humedad, velocidad del viento y una pequeña descripción del tiempo atmosférico, que se mostrará con una imagen).

La aplicación también mostrará un **mapa** en el cual se dibuja el **trayecto** que se realizaría.

En la aplicación se podrá **editar** la **información del usuario** o incluso **eliminarlo**. Esta información podrá ser actualizada en la página de “Ver Perfil”.

Las **búsquedas** también podrán ser **eliminadas** por el usuario en el historial de búsqueda una a una con un botón de borrado, o todas a la vez con el botón “Limpiar historial”.

CATÁLOGO DE REQUISITOS

Usuario:

La aplicación guardará la información de los usuarios que se registren en una base de datos.

Un usuario no registrado podrá:

- Registrarse.
- Realizar búsquedas sin registrarse.

Un usuario registrado podrá:

- Editar su perfil.
- Eliminar su cuenta.

Acciones que se pueden realizar con un usuario:

- Crear usuario
- Validar que el usuario y la contraseña coincidan con el de la base de datos.
- Buscar un usuario por el código de usuairo.
- Editar un usuario.
- Eliminar un usuario.

Usaremos el código del usuario (usuario en la aplicación) y la contraseña para controlar el acceso a la aplicación en el apartado login.

También utilizaremos el código del usuario y la contraseña para poder cambiar los datos del usuario en el perfil del usuario.

Datos del Usuario:

- Código del Usuario (codUsuario)
- Nombre del Usuario (nomUsuario)
- Email (email)
- Contraseña (password)

Búsqueda:

La aplicación guarda las búsquedas que realiza un usuario registrado automáticamente.

Un usuario registrado podrá:

- Almacenar las búsquedas que realice en la aplicación (proceso automático).
- Visualizar el historial de búsquedas que realice en la aplicación.
- Eliminar las búsquedas una a una.
- Eliminar todo el historial de una vez.

Acciones que se pueden realizar con las búsquedas:

- Insertar una búsqueda en la base de datos.
- Listar todas las búsquedas realizadas por un usuario.
- Contar el número de búsquedas que hay en el historial.
- Eliminar una búsqueda.
- Eliminar todas las búsquedas.

Un usuario no registrado también podrá realizar búsquedas, pero no serán almacenadas en la base de datos.

Datos de la Búsqueda:

- Código de la Búsqueda (codBusqueda)
- Código del Usuario que realizó la búsqueda (codUsuario)
- Fecha de realización de la búsqueda (fecha)
- Lugar de salida (salida)
- Lugar de llegada (llegada)
- Distancia entre los dos lugares (distancia)
- Duración del trayecto en coche (duracion)
- Temperatura en el lugar de llegada (temperatura)
- Probabilidad de nubes en el lugar de llegada (nubes)
- Probabilidad de precipitacion en el lugar de llegada (precipitacion)
- Humedad relativa en el lugar de llegada (humedad)

- Velocidad del viento en el lugar de llegada (viento)
- Descripción del tiempo que hará (una imagen) (frase)

INFORMACIÓN RELATIVA A CADA PÁGINA

INICIO/BÚSQUEDA:

Página principal en la que se puede **realizar la búsqueda** sin importar si se ha iniciado sesión o no.

Si no se ha iniciado sesión, únicamente realiza la búsqueda y muestra una tabla con los datos correspondientes a la búsqueda y un mapa en el que se dibuja la ruta del trayecto. Si por el contrario se ha **iniciado sesión**, se mostrará la tabla y el mapa y dichos datos de la búsqueda serán **almacenados** en la base de datos del usuario correspondiente.

La aplicación sólo permite realizar búsquedas **en España** (sin contar Portugal) y el trayecto se realiza **en coche** siempre, por lo cual, Ceuta, Melilla, las islas Baleares y Canarias quedarán excluidas de la búsqueda.

Elementos de la página:

- Un **formulario** de dos campos: lugar de salida y lugar de llegada. Ambos son de tipo cadena o string. También dispone de un checkbox en el cual el usuario podrá elegir si desea evitar las autopistas o no.

- Un **botón de enviar** que recoge los valores escritos en los campos anteriores y redirige a la página del controlador de la página de búsqueda.

- Si no hay ninguna sesión iniciada, aparecerá un **botón de iniciar sesión**, por si el usuario quiere iniciar sesión para guardar sus búsquedas. El botón redirige a la página de login.

- Si no se ha iniciado sesión, también aparecerá un **botón de registro**, por si el usuario quiere guardar sus búsquedas y aún no tiene una cuenta. El botón redirige a la página de registro.

- Si por el contrario se ha iniciado sesión, aparecerá un **botón de Ver Perfil**, a través del cual se puede acceder a la página del perfil de usuario para hacer los cambios correspondientes o bien sólo visualizar su perfil o historial de búsquedas.

- Si se ha iniciado sesión, también aparecerá un **botón de Ver Historial** para poder acceder al historial de búsquedas del usuario.

- Si se ha iniciado sesión, también aparecerá un botón de Cerrar Sesión, que redirige a la página de Logoff (no tiene layout), que cierra la sesión del usuario y redirige de nuevo a la página de inicio/búsquedas.

Control de errores que efectúa la página:

- Si la búsqueda se realiza en Ceuta, Melilla o las islas, la aplicación mostrará el siguiente mensaje: “El trayecto solo se realiza en coche”.

- Si existe algún problema con el servicio REST, la aplicación mostrará el siguiente mensaje: “Hay un problema con el servicio REST”.

- Si existe algún problema que no se conoce, la aplicación mostrará el siguiente mensaje: “Hay un problema desconocido”.

- Si la solicitud no es válida, la aplicación mostrará el siguiente mensaje: “La solicitud no es válida”.

- Si no se ha encontrado ningún resultado, la aplicación mostrará el siguiente mensaje: “No se ha encontrado la búsqueda”.

- Si la búsqueda se ha realizado fuera de España, la aplicación mostrará el siguiente mensaje: “El trayecto solo se realiza en España”.

Trozo de código para extraer información JSON de una URL:

```
if($entradaOK){
    //URL DEL SERVICIO REST DE LA API DE GOOGLE (DISTANCIA Y DURACIÓN).
    if(isset($_POST['autopista'])){
        $gmUrl="https://maps.googleapis.com/maps/api/distancematrix/json
        ?region=es?country=spain?language=es?units=imperial&avoid=tolls
        &origins=".urlencode($campos['salida']).urlencode(", españa")."
        &destinations=".urlencode($campos['llegada']).urlencode(", españa")."
        &key=AIzaSyBOTmJGCrDskI_CD6DePfvmP-SCsKLRRT0";
    }else{
        $gmUrl="https://maps.googleapis.com/maps/api/distancematrix/json
        ?region=es?country=spain?language=es?units=imperial
        &origins=".urlencode($campos['salida']).urlencode(", españa")."
        &destinations=".urlencode($campos['llegada']).urlencode(", españa")."
        &key=AIzaSyBOTmJGCrDskI_CD6DePfvmP-SCsKLRRT0";
    }
    //ARCHIVO JSON QUE RECIBE LA URL.
    $gmJson = @file_get_contents($gmUrl);
    if ($gmJson === false) {
       print("<br/><span class='errorphp'>NO SE PUEDE ACCEDER A LA URL GMAPS</span>
    }else{
        //ARRAY QUE DEVUELVE EL JSON DE LA API.
    $gmArray=json_decode($gmJson,true);
```

La aplicación dispone de varios mecanismos de **control de errores**, tanto en métodos de entrada de **texto** como de **peticiones** que se realizan al **servidor**.

En cuanto a validación de entrada, existe **validación de entrada** para los dos campos (lugar de salida y lugar de llegada), tanto en el lado del **cliente** como en el lado del **servidor**.

En el lado del **cliente** he utilizado **Javascript**, mientras que en el lado del **servidor** he utilizado **PHP** para validar el formulario.

Código Javascript:

vInicio.php

```
var patron=/^[a-zA-ZàèìòùÀÈÌÒÙáéíóúýÁÉÍÓÚÝâêîôûÂÊÎÔÛãñõÃÑÕäëïöüÿÄËÏÖÜŸçÇ\s,\/]+$/;
function validacion(){
    var camposalida = document.getElementById("desde");
    var campollegada = document.getElementById("address");
    //SI EL CAMPO ESTÁ VACÍO
    if(camposalida.value.length < 1 || campollegada.value.length < 1){
        sessionStorage.setItem("errorcampos","Campo vacío");
        return false;
    }
    //SI NO COINCIDE CON EL PATRÓN
    else if(!camposalida.value.match(patron) || !campollegada.value.match(patron)){
        sessionStorage.setItem("errorcampos","Solo se permiten letras y los caracteres
        return false;
    }
    //SI COINCIDE CON EL PATRÓN
    else{
        sessionStorage.setItem("errorcampos","");
        return true;
    }
}
```

cInicio.php

```
//CREAR ALMACENAMIENTO LOCAL PARA LOS DOS CAMPOS.
var autopista = sessionStorage.getItem('autopista');
```

```
function mostrarerrores(){
    if(errorcampos.length>0){
        document.getElementById("errorcampos").innerHTML=errorcampos;
        document.getElementById("div1").style.marginTop="10px";
        document.getElementById("contError").style.display="flex";
        sessionStorage.setItem("errorcampos","");
    }
    if(errorcampos.length<1 && errorcampos.length<1){
        calculaRuta();
    }

}
```

Código PHP:

cInicio.php

```php
//ACCIÓN SI SE PULSA EL BOTÓN ACEPTAR.
if(isset($_POST['aceptar'])){
    $entradaOK=true;
    $fraseimg="";
    //RECOGER EN UN ARRAY LOS VALORES DE LOS CAMPOS DEL FORMULARIO.
    $campos['salida']=$_POST['salida'];
    $campos['llegada']=$_POST['llegada'];

    //INCLUIR LA LIBRERÍA DE FUNCIONES
    include "core/funciones.php";
    if(validartextocoma($campos['salida'])){
        $entradaOK=false;
        //ALMACENAR EN EL ARRAY EL ERROR RECIBIDO.
        $errores['salida']=validartextocoma($campos['salida']);
        $campos['salida']="";
    }
    if(validartextocoma($campos['llegada'])){
        $entradaOK=false;
        //ALMACENAR EN EL ARRAY EL ERROR RECIBIDO.
        $errores['llegada']=validartextocoma($campos['llegada']);
        $campos['llegada']="";
    }
}
```

funciones.php

```php
function validardireccion($direccion){
    $patrontexto="/^[a-z A-ZàèìòùÀÈÌÒÙáéíóúýÁÉÍÓÚÝâêîôûÂÊÎÔÛãñõÃÑÕäëïöüÿÄËÏÖÜŸçÇ,\/]+$/";
    $error="";
    if(empty(trim($direccion))){ //Se genera error si el campo está vacío
        $error = "Campo vacío";
    }else{
        if(!preg_match($patrontexto, $direccion)){
            $error="Introduce solo caracteres de direcciones";
        }
    }
    return $error;
}
```

También existe **validación** en cuanto a peticiones realizadas al **servicio Rest**.

Se controlan los errores de acceso a los datos:

- Si no se reconoce la URL.
- Si no ha devuelto ningún dato / si no se ha encontrado la búsqueda.
- Si no funciona alguno de los servicios Rest.
- Si se han realizado demasiadas peticiones al servidor.

REGISTRO:

Página donde se puede dar de **alta a un usuario**. Contiene mecanismo de **verificación** de los campos. La aplicación no permitirá la creación de un usuario con el código del usuario (Usuario en la aplicación) igual a otro ya creado. Los demás campos serán irrelevantes.

Elementos de la página:

- Un **formulario** donde se insertan los datos del usuario (código de usuario, nombre, email y password).

- Un **botón de aceptar**, que añade el usuario a la base de datos si no ha ocurrido ningún error y redirige a la página de inicio con la sesión iniciada automáticamente.

  Si el usuario ya existe, la aplicación mostrará un mensaje conforme el usuario ya existe.

- Un **botón de cancelar**, que redirige otra vez a la página de inicio.

LOGIN:

Página de **inicio de sesión** de un usuario (control de acceso). La **contraseña** será **cifrada** al crear el usuario con la función **hash()** y cifrado **sha256**.

Elementos de la página:

- Un **formulario** donde se insertan los datos del usuario (código y password). La aplicación realiza la validación del usuario accediendo a la base de datos del servidor.

- Un **botón de aceptar**, que inicia la sesión del usuario si se ha validado correctamente y redirige a la página de inicio con la sesión iniciada. Si no se ha encontrado el usuario o no coincide la contraseña, la aplicación mostrará por pantalla un mensaje informativo.

- Un **enlace a la página de registro**, que redirige a la página de registro por si no el usuario no se había dado de alta.

- Un **botón de cancelar**, que redirige otra vez a la página de inicio.

En esta página sólo utilizo validación en el lado del servidor. Compruebo si el usuario coincide con el usuario guardado en la base de datos del servidor.

```php
//COMPROBAR SI EXISTE EL USUARIO LLAMANDO A LA FUNCIÓN DE LA CLASE USUARIO Y ALMACENARLO EN LA VARIABLE
$usuario=Usuario::validarUsuario($_POST['usuario'],hash('sha256', $_POST['password']));
//ACCIÓN SI NO SE HA ENCONTRADO EL USUARIO
if($usuario==null){
    //LA ENTRADA SERÁ FALSA
    $entradaOK=false;
    //INTRODUCIR VALORES EN LAS VARIABLES DE ERROR
    $error = "Debes introducir un nombre de usuario y una contraseña válidos";
    $span="<span class='error' name='error'>".$error."</span>";
}
```

PERFIL:

Página con **información del usuario** que inició sesión donde se puede **editar** o **eliminar** su cuenta. Se muestran los datos del usuario en un formulario visual.

Elementos de la página:

- Un **formulario visual** con los datos del usuario (código de usuario, nombre y email). En esta sección no se podrán editar los campos del formulario.

- Un **botón de Editar usuario**, que redirige la información de la sesión del usuario a la página de editar usuario.

- Un **botón de Borrar usuario**, que redirige la información de la sesión del usuario a la página de borrar usuario.

- Un **botón de volver**, que redirige a la página de inicio.

EDITAR USUARIO:

Página de **edición del usuario** que inició sesión.

Elementos de la página:

- Un **formulario** donde muestra los datos del usuario en cada campo correspondiente. El **código** del usuario (Usuario) **no se podrá editar**. El resto de los campos sí se pueden editar (nombre de usuario, email y contraseña). Si se cambia algún campo, **se deberá introducir la contraseña** actual del usuario o no se podrá editar la información. Los campos de password (password antiguo y nuevo password) estarán vacíos para introducir la contraseña, bien sea sólo la antigua o las dos si se quiere cambiar.

- Un **botón de aceptar**, que actualiza la información del usuario en la base de datos y en la sesión si no existe ningún error al insertar los datos y redirige a la página de inicio. Si existe algún error, simplemente muestra un mensaje por pantalla advirtiendo de que no se ha podido editar el usuario.

- Un **botón de cancelar**, que redirige otra vez a la página de inicio.

BORRAR USUARIO:

Página que **pregunta al usuario** si realmente quiere **borrar** su perfil.

Elementos de la página:

- Un t**exto informativo** que advierte de que se está intentando eliminar una cuenta.

- Un **formulario** donde se visualizan los datos del usuario (código, nombre, email y password). No se podrán editar los campos.

- Un **botón de aceptar**, que elimina la cuenta de usuario y la sesión correspondiente y redirige a la página de inicio, si ha podido eliminar el usuario. Si no ha podido eliminar el usuario, redirige otra vez a la página de perfil del usuario.

- Un **botón de cancelar**, que redirige otra vez a la página de perfil del usuario.

BÚSQUEDAS/HISTORIAL:

Página que **lista** todas las **búsquedas** que ha realizado ese usuario, con opción a poder **borrar una a una o todo el historial de una vez.**

Elementos de la página:

- Una **tabla** con los datos de la búsqueda del usuario (Usuario que realizó la búsqueda y datos de la búsqueda).

- En la última celda del encabezado de la tabla existe **un botón de limpiar historial**, el cual elimina todas las búsquedas realizadas por el usuario que inició sesión.

- En la última celda de cada fila existe **un botón de borrar**, el cual redirige los datos de la búsqueda a la página de borrar búsqueda.

- Un **botón de volver**, que redirige a la página de inicio.

BORRAR BÚSQUEDA:

Página que **pregunta al usuario** si realmente quiere **borrar la búsqueda** seleccionada en el historial de búsquedas.

Elementos de la página:

-Un **formulario** donde se visualizan algunos datos de la búsqueda (código del usuario, fecha, lugar de salida y lugar de llegada). Estos datos no se podrán editar.

-Un **texto informativo** que advierte de que se está intentando eliminar una

búsqueda.

-Un **botón de aceptar**, que elimina la búsqueda y redirige a la página del historial si no ha ocurrido ningún error. Si no se ha podido eliminar la búsqueda muestra por pantalla un mensaje informativo y redirige de nuevo a la página del historial.

-Un **botón de cancelar**, que redirige otra vez a la página del historial.

FUNCIONES JAVASCRIPT

- **sesion()** Almacena en SessionStorage los valores de los campos de la búsqueda.

```
//FUNCIÓN PARA ALMACENAR EN SESSION STORAGE LOS VALORES DE LOS CAMPOS.
function sesion(){
    salida=document.getElementById('desde').value;
    llegada=document.getElementById('address').value;
    autopista=document.getElementById('autopista').checked;
    if(document.getElementById('desde').value.length !=0 &&
    document.getElementById('address').value.length !=0){
        sessionStorage.setItem("salida",salida+",spain");
        sessionStorage.setItem("llegada",llegada+",spain");
        sessionStorage.setItem("autopista",autopista);
        return true;
    }
}
```

- **validación()** Valida el texto introducido en los campos de la búsqueda. Sólo permite caracteres alfabéticos y acentuados.

```
//CREAR PATRÓN DE VALIDACIÓN.
var patron=/^[a-zA-ZàèìòùÀÈÌÒÙáéíóúýÁÉÍÓÚÝâêîôûÂÊÎÔÛãñõÃÑÕäëïöüÿÄËÏÖÜŸçÇ\s,\/]+$/;
//FUNCIÓN DE VALIDACIÓN DE LOS CAMPOS DE BÚSQUEDA.
function validacion(){
    var camposalida = document.getElementById("desde");
    var campollegada = document.getElementById("address");
    //SI EL CAMPO ESTÁ VACÍO
    if(camposalida.value.length < 1 || campollegada.value.length < 1){
        sessionStorage.setItem("errorcampos","Campo vacío");
        return false;
    }
    //SI NO COINCIDE CON EL PATRÓN
    else if(!camposalida.value.match(patron) || !campollegada.value.match(patron)){
        sessionStorage.setItem("errorcampos","Solo se permiten letras y los caracteres / y ,");
        return false;
    }
    //SI COINCIDE CON EL PATRÓN
    else{
        sessionStorage.setItem("errorcampos","");
        return true;
    }
}
```

- **mostrarerrores()** Muestra los errores obtenidos de la validación de los campos.

```
//FUNCIÓN PARA MOSTRAR LOS ERRORES DE LOS CAMPOS DE BÚSQUEDA.
function mostrarerrores(){
    if(errorcampos.length>0){
        document.getElementById("errorcampos").innerHTML=errorcampos;
        document.getElementById("div1").style.marginTop="10px";
        document.getElementById("contError").style.display="flex";
        sessionStorage.setItem("errorcampos","");
    }
    if(errorcampos.length<1 && errorcampos.length<1){
        calculaRuta();
    }
}
```

- **calculaRuta()** Calcula y dibuja en el mapa la ruta con los datos de la búsqueda.

```
//FUNCIÓN PARA CALCULAR EL WAYPOINT CON LOS DATOS ALMACENADOS.
function calculaRuta(){
    //ALMACENAR LOS PARÁMETROS QUE NECESITA PARA CALCULAR LA RUTA.
    var datos={
        origin: salida,
        destination: llegada,
        travelMode:'DRIVING',
        avoidTolls: autopista,
        region:'es'
    };
    //INTENTAR CALCULAR LA RUTA CON LOS PARÁMETROS ANTERIORES.
    directionsService.route(datos, function(result,status){
        //SI EL ESTADO ES CORRECTO SIGNIFICA QUE HA PODIDO CALCULAR LA RUTA.
        if(status === "OK"){
            //INSERTAR LA RUTA CALCULADA EN EL MAPA.
            directionsDisplay.setDirections(result);
            return true;
        }else{
            return false;
        }
    });
}
```

- **mapa()** Si el tamaño del navegador o dispositivo es mayor que 1000px, mostrará un mapa con un zoom. Si es más pequeño, mostrará un mapa con más zoom.

```
function mapa(){
    //SI EL TAMAÑO DE LA VENTANA DEL NAVEGADOR O DISPOSITIVO ES MAYOR, SE MOSTRARÁ UN MAPA CON MÁS ZOOM
    if (window.outerWidth>1000 || screen.width>1000){
        //DEFINIR EL MAPA EN EL DIV CON ID=MAP, CON ZOOM 6 Y CENTRADO EN ESPAÑA.
        var map = new google.maps.Map(document.getElementById('map'), {
          zoom: 6,
          center: {lat:39.6693985 , lng:-4.0645625 }
        });
        directionsDisplay.setMap(map);
    }
    //SI EL TAMAÑO DE LA VENTANA DEL NAVEGADOR O DISPOSITIVO ES MENOR, SE MOSTRARÁ UN MAPA CON MENOS ZOOM
    if (window.outerWidth<=1000 || screen.width<=1000){
        //DEFINIR EL MAPA EN EL DIV CON ID=MAP, CON ZOOM 5 Y CENTRADO EN ESPAÑA.
        var map2 = new google.maps.Map(document.getElementById('map'), {
          zoom: 5,
          center: {lat: 39.6693985, lng: -4.0645625}
        });
        directionsDisplay.setMap(map2);
    }
}
```

FUNCIONES JQUERY

```
//SI EXISTEN ERRORES, LOS MUESTRA.
$(document).ready(function(){
    //SI EXISTEN ERRORES, LOS MUESTRA.
    if($("#error").html().length!=0 || $("#errorcampos").html().length!=0){
        $("#div1").css("marginTop","10px");
        $("#contError").css("display","flex");
    }
});
```

```
<script>
$(document).ready(function(){
    //RECOGER EL ERROR CAPTURADO CON PHP.
    var vararerror=$("#varError").html();
    if($("#varError").html().length!=0){
        $("#error").text(vararerror);
    }
    //SI EXISTEN ERRORES, LOS MUESTRA.
    if($("#error").html().length!=0 || $("#errorcampos").html().length!=0){
        $("#div1").css("marginTop","10px");
        $("#contError").css("display","flex");
    }
});
</script>
```

TECNOLOGÍAS Y HERRAMIENTAS

Entorno de desarrollo

Windows 7 Enterprise x64

Entorno de explotación

Ubuntu Server x64

Servidor HTTP

Apache 2

Sistema Gestor de Base de Datos

MySql-Server

Lenguajes en el lado del servidor

Lenguajes en el lado del cliente

Lenguajes de marcas

Editor de texto

Sublime Text 3

MODELOS Y DIAGRAMAS

MODELO DE CASOS DE USO

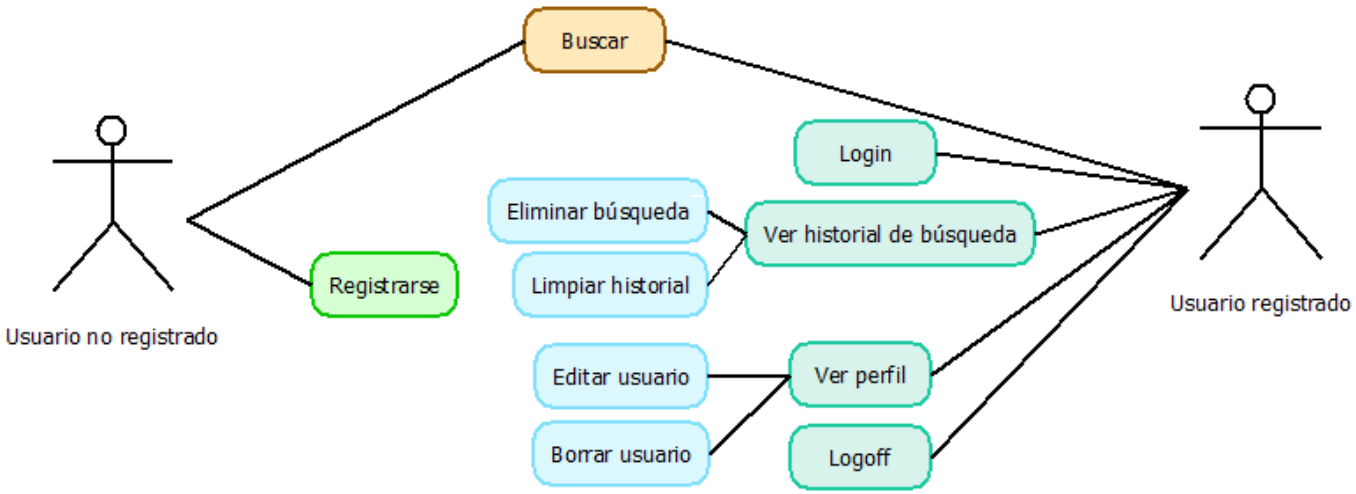
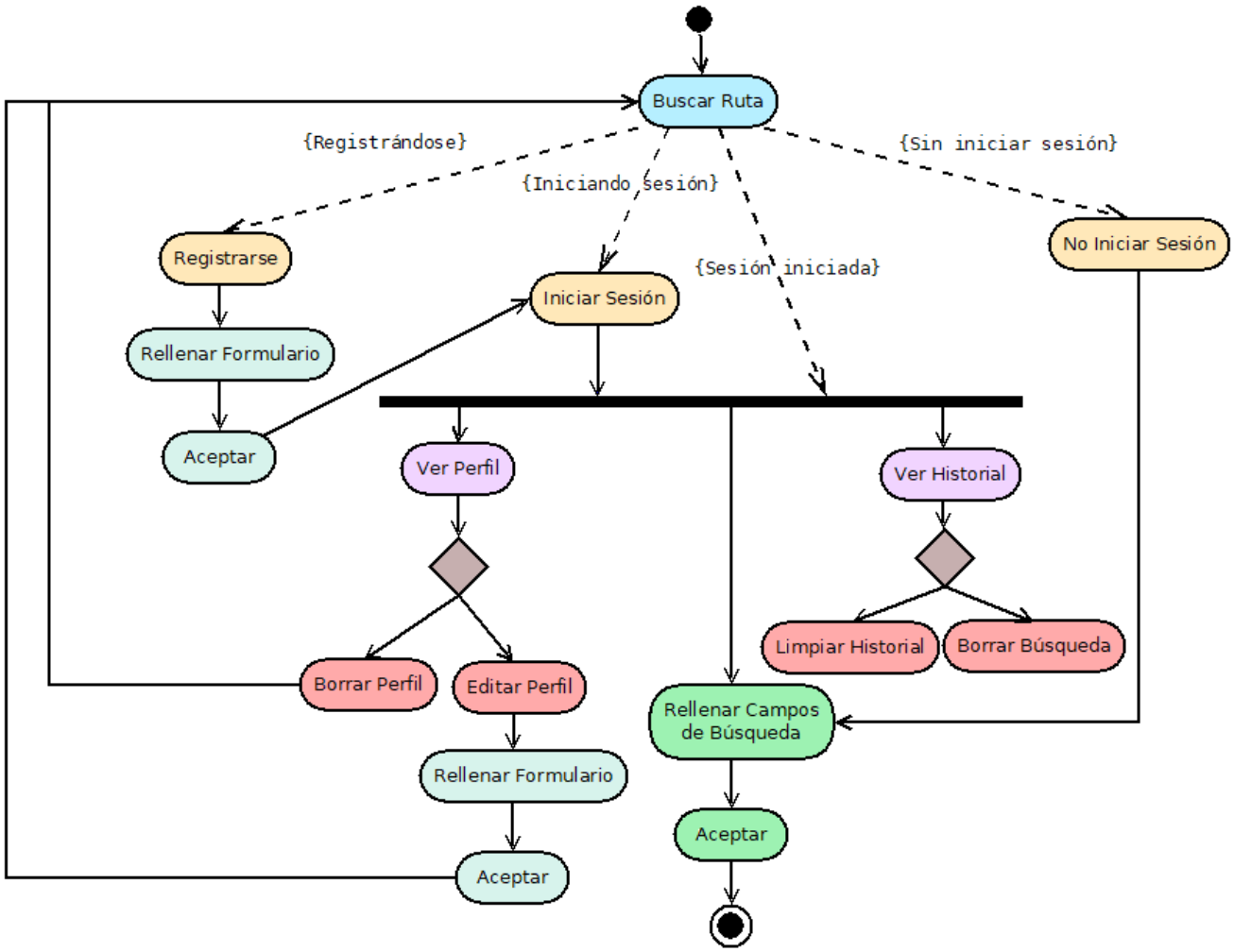
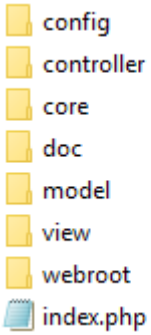
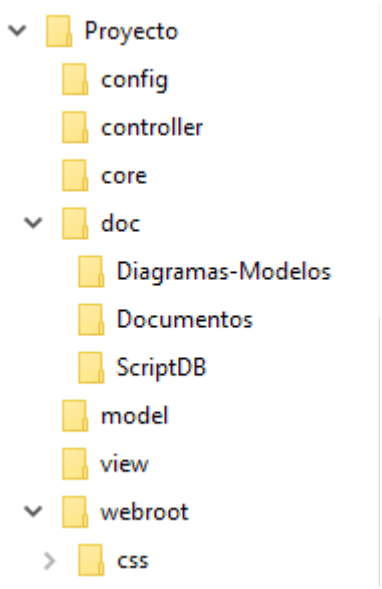


DIAGRAMA DE ACTIVIDADES

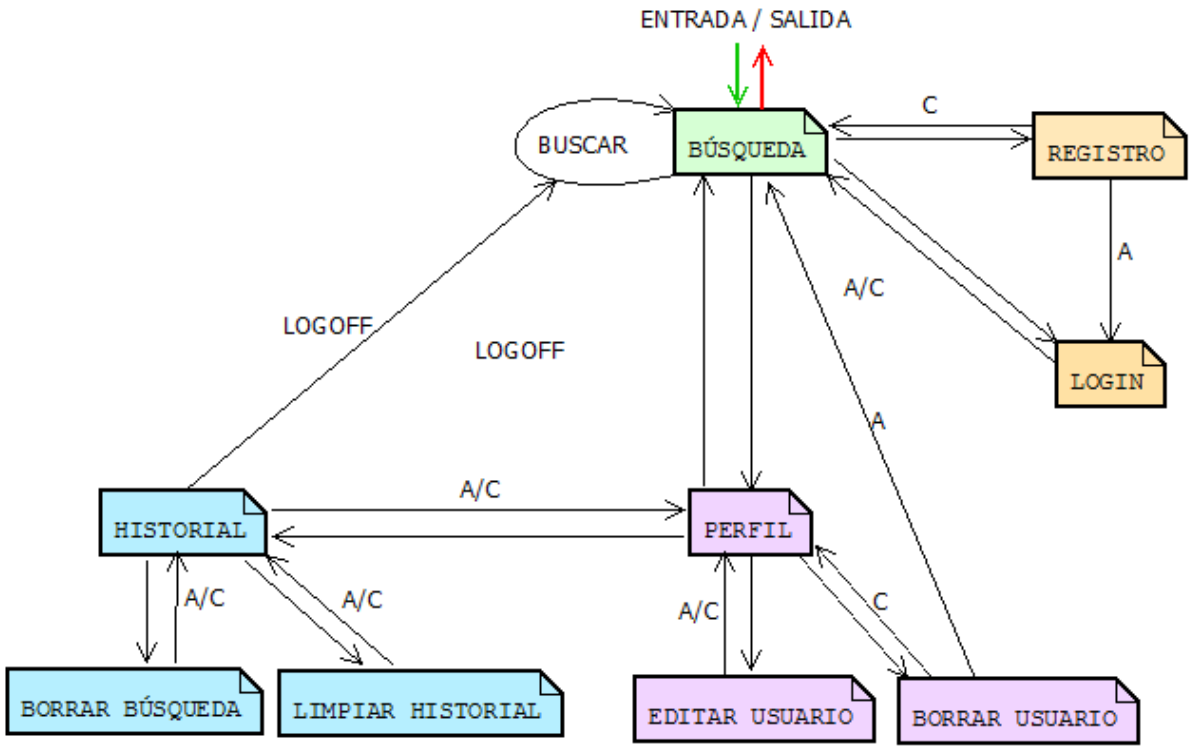


ALMACENAMIENTO





ÁRBOL DE NAVEGACIÓN



ÁRBOL DE NAVEGACIÓN DE FICHEROS

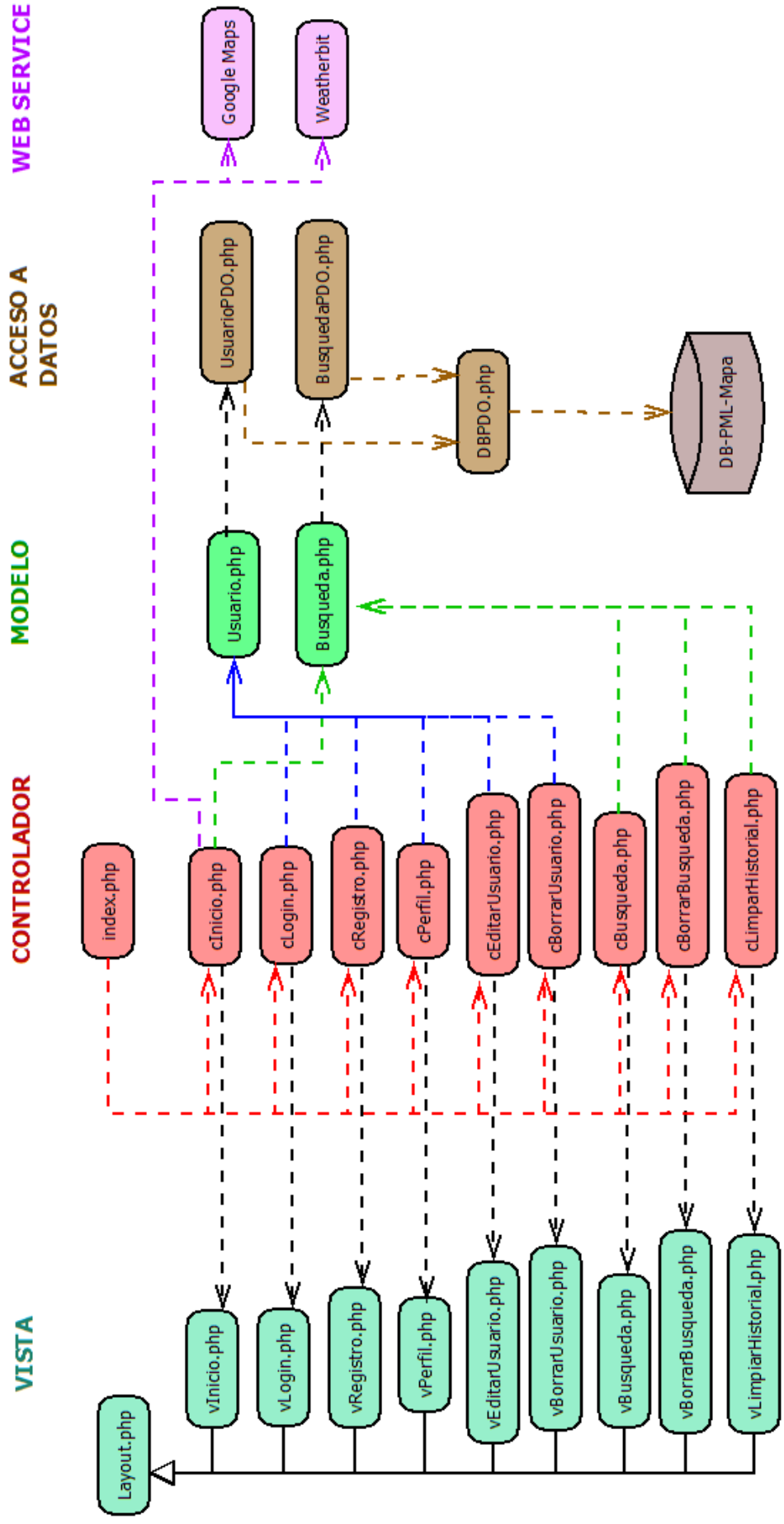


DIAGRAMA DE COLABORACIONES

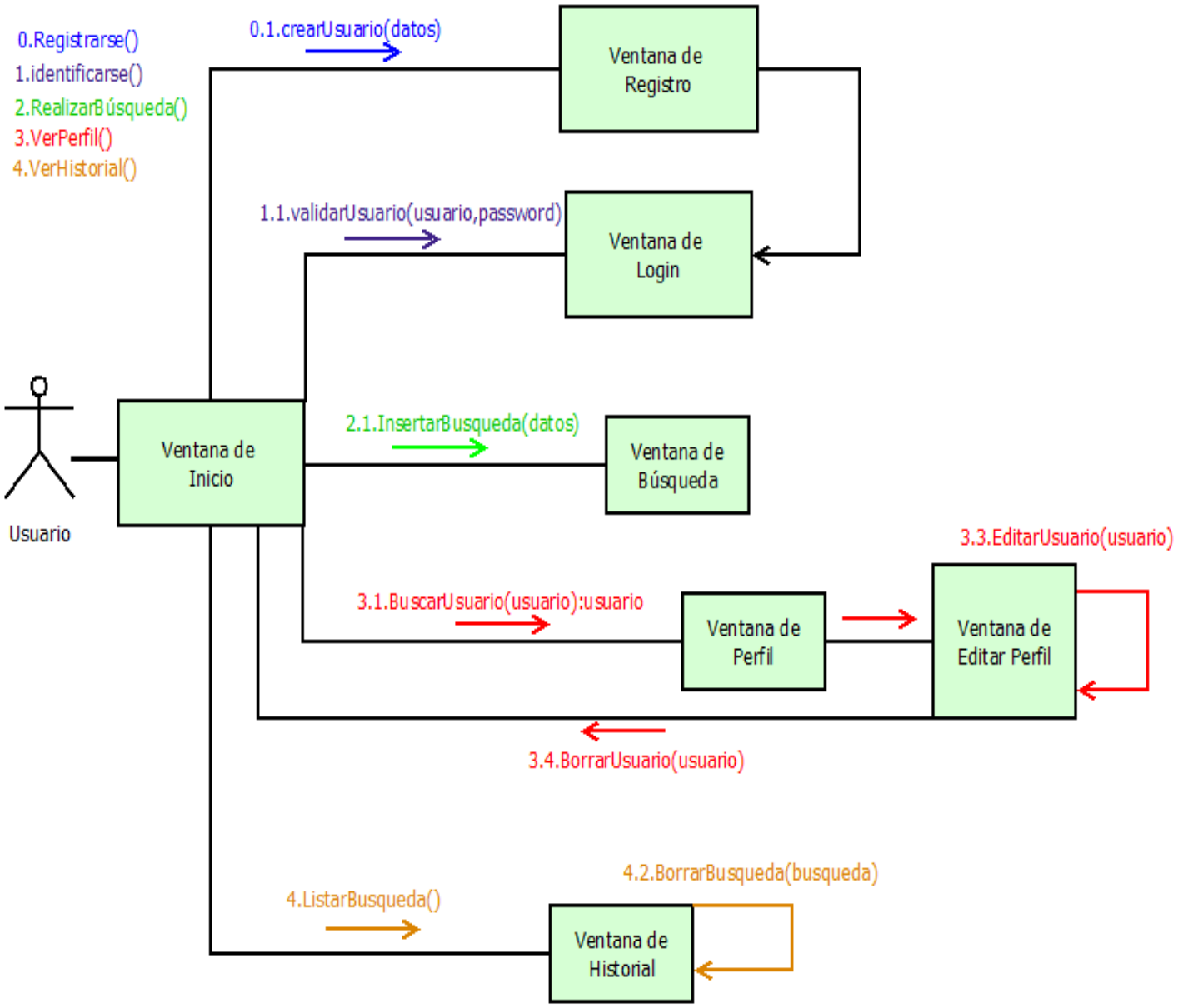
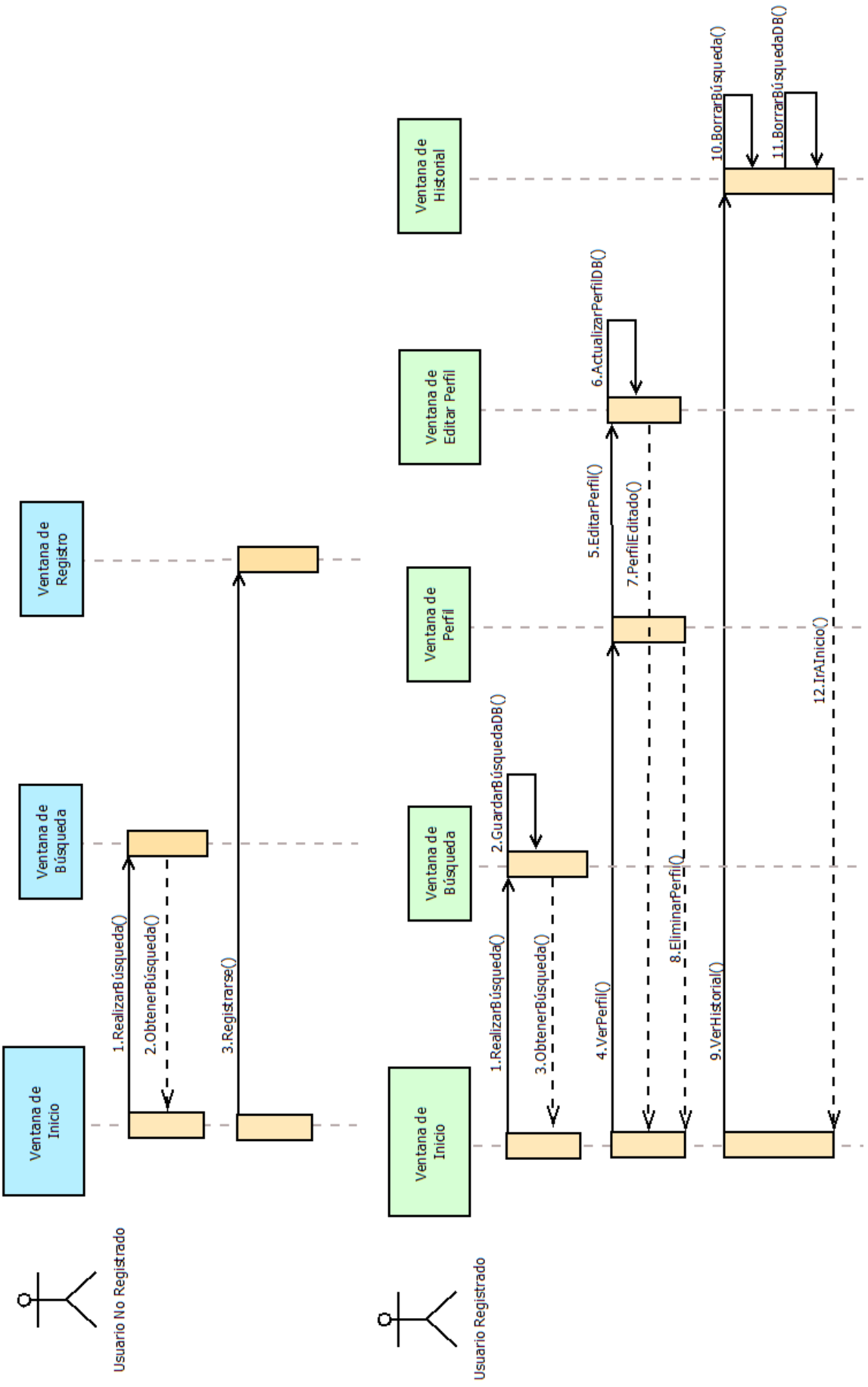


DIAGRAMA DE SECUENCIA



MODELO ENTIDAD RELACIÓN

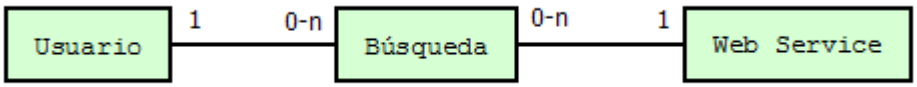
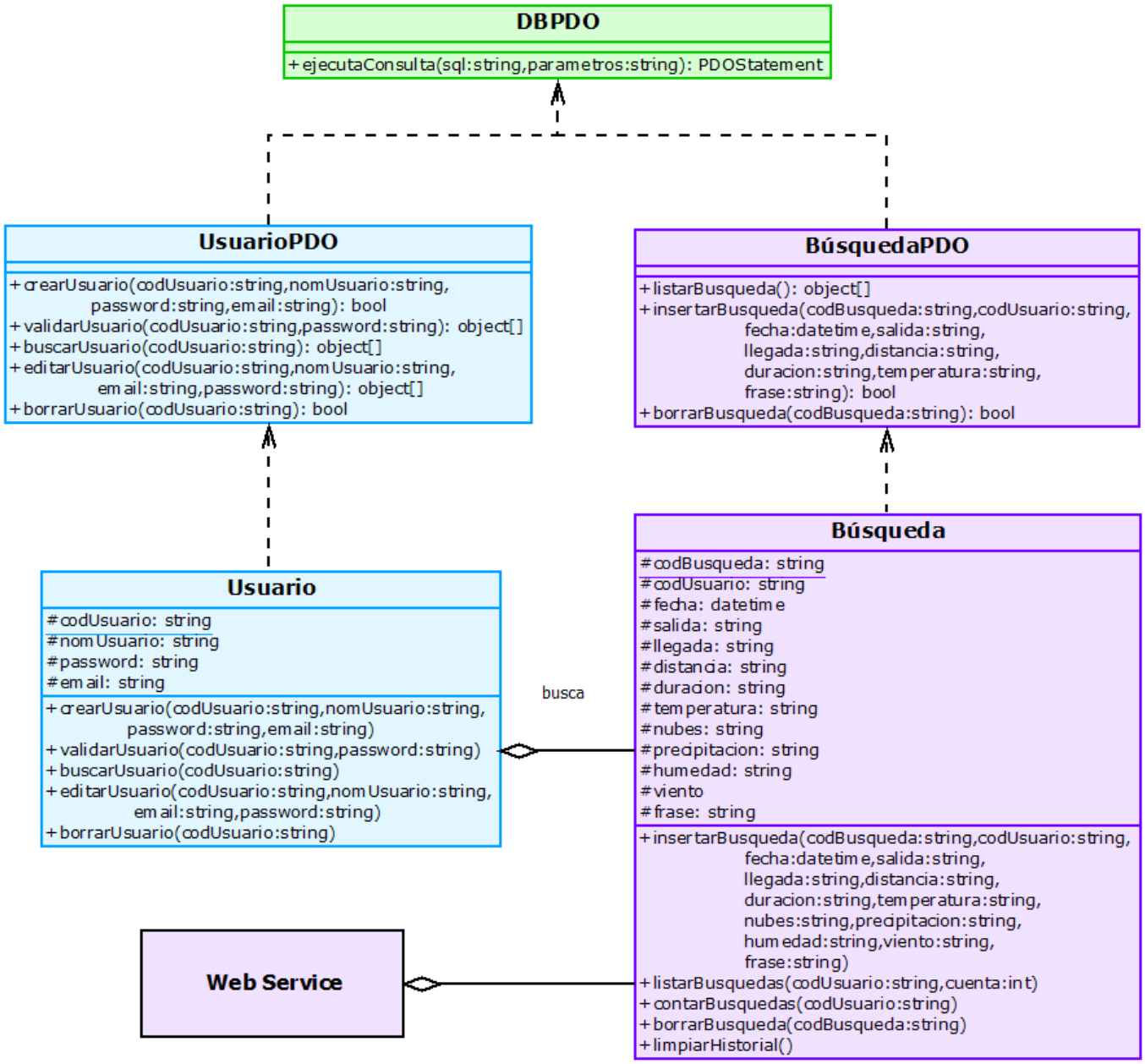
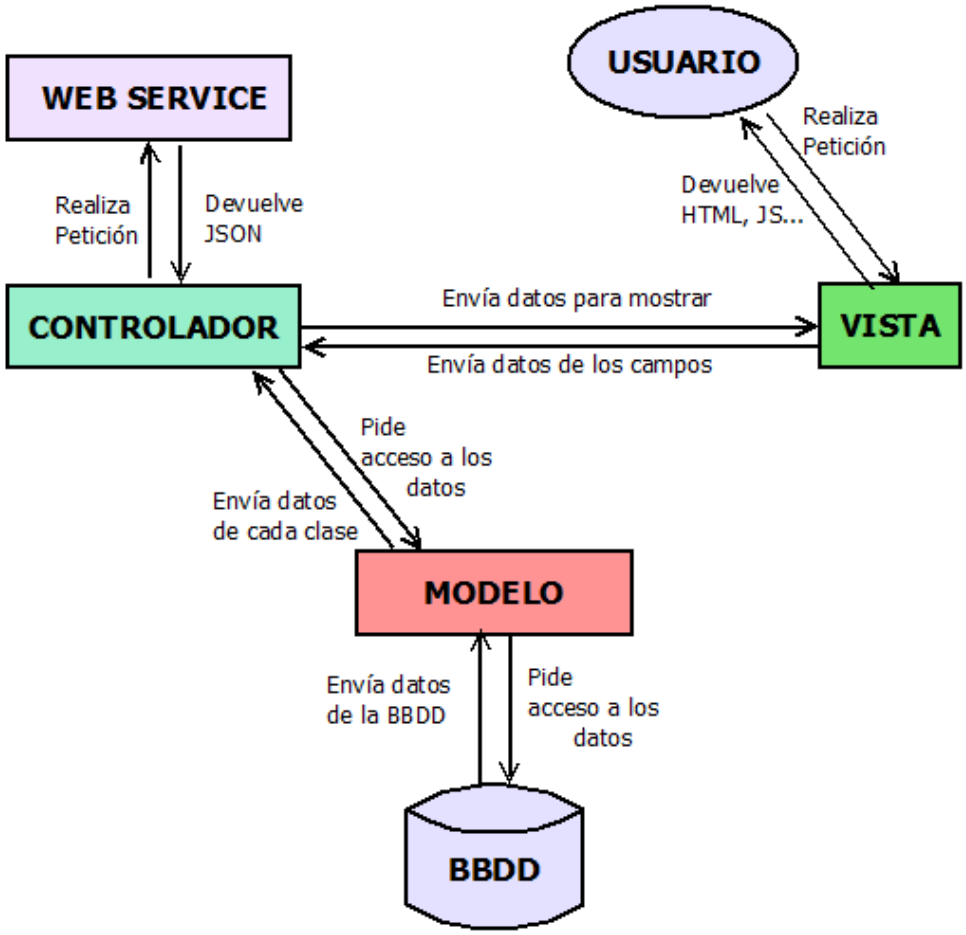


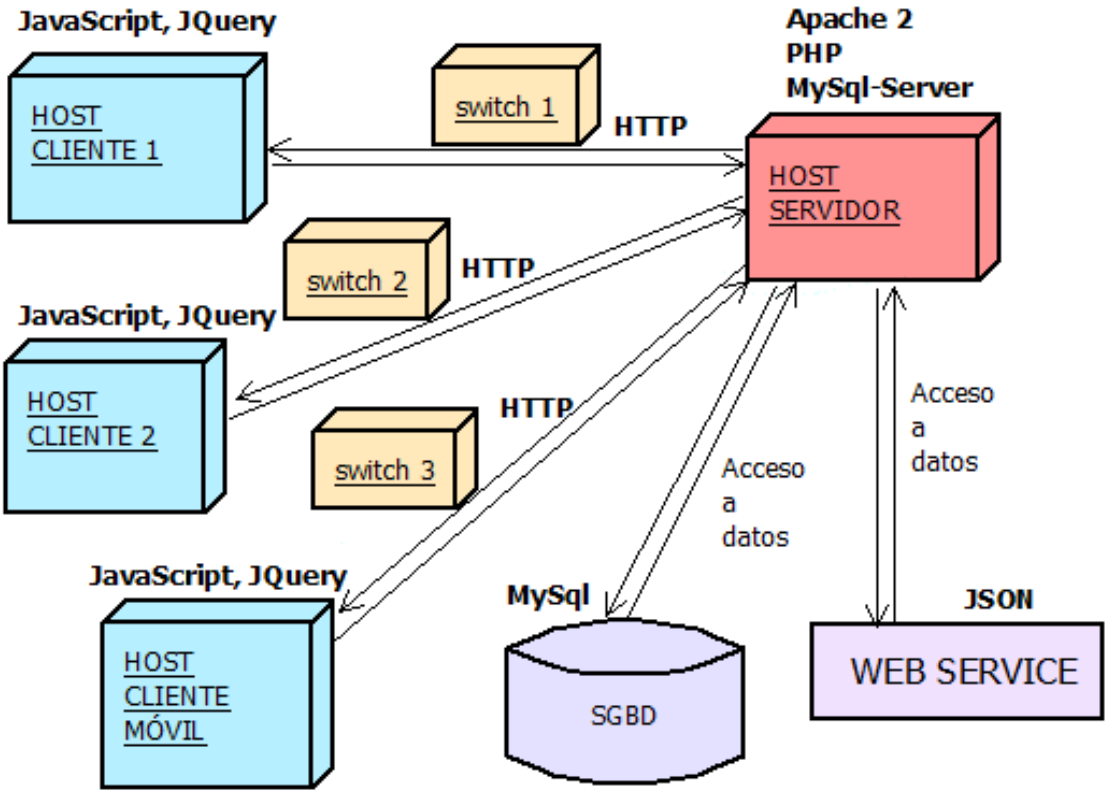
DIAGRAMA DE CLASES



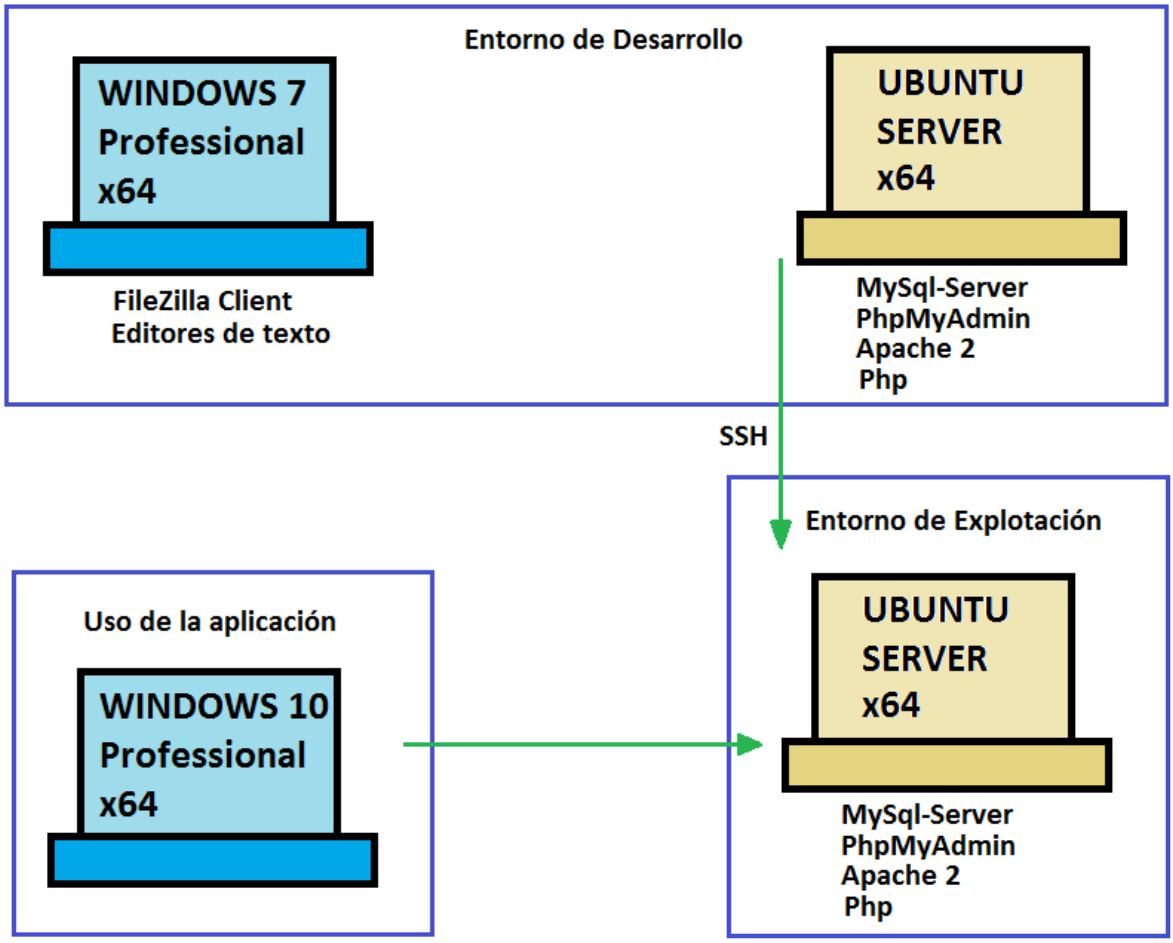
VISTA DESPLEGADA DE LA ARQUITECTURA



ARQUITECTURA



DESPLIEGUE DE LA APLICACIÓN



SERVICIOS REST UTILIZADOS

GOOGLE MAPS API

Google Maps Distance Matrix: De esta API saco los datos relacionados a la trayectoria entre dos lugares, como la distancia entre estos y el tiempo de viaje, entre otros.

URL:

https://maps.googleapis.com/maps/api/distancematrix/json&key=AIzaSyBOTmJGCrDskICD6DePfvmP-SCsKLRRT0

Google Maps Servicio de Indicaciones (Waypoints): Obtiene los datos de los lugares y calcula la ruta más corta para ir, en este caso, en coche. Genera el mapa con la trayectoria correspondiente.

URL:

Coge por defecto la URL anterior, ya que es la API de Google Maps. Este servicio lo único que necesita es llamar a un objeto DirectionsService y a otro objeto DirectionsRenderer e insertar los parámetros correspondientes para calcular la ruta.

WEATHERBIT.IO

De esta API saco los datos relacionados al tiempo en 48 horas de un lugar en concreto, en este caso del lugar de llegada.

URL:

https://api.weatherbit.io/v2.0/forecast/hourly&lang=es&key=9d1b4156f1cb4c7295d5d203d68969d9

TIEMPO EMPLEADO EN HACER LA APLICACIÓN:

He empleado aproximadamente unas 70 horas mínimo en hacer la aplicación.

ESTIMACIÓN DE COSTES EN BASE AL TIEMPO EMPLEADO:

Para calcularlo, he optado por sumar los costes a la hora, que serían unos 7 euros, a 70 horas serían unos 490 euros por hacer la aplicación.

MI APLICACIÓN Y POSIBLES MEJORAS QUE REALIZAR

En la aplicación se pueden hacer muchas mejoras, ya que el servicio Rest que he utilizado de Google Maps tiene muchísimas posibilidades y muchos parámetros que se pueden configurar.

Para elaborar mi aplicación he optado por hacerlo bastante sencillo en el aspecto de que he utilizado parámetros bastante fijos para extraer la información que deseaba.

Al utilizar más de un servicio rest y combinarlos entre sí habría que controlar muchísimos aspectos, tales como el país que quieres visitar, el modo de viaje y el tipo de vehículo.

Para mi aplicación, estas opciones las he considerado irrelevantes ya que añadiéndolas o sin añadirlas, mi aplicación realiza los mismos pasos, es decir, realiza la búsqueda que se le pide y muestra datos del trayecto, del tiempo atmosférico y muestra el mapa con la ruta correspondiente. Añadir estas opciones sería sólo pérdida de tiempo ya que, como ya he mencionado, la aplicación realizaría la misma consulta con o sin esos datos.

Como posibles mejoras podría añadir algún campo que pidiera datos del país, del modo de viaje (a pie, en vehículo) o del tipo de vehículo que se quiere utilizar. Así, mi aplicación podría utilizarse a nivel mundial.

También podría añadir la opción de introducir números en la búsqueda y controlar que la aplicación no lo interprete como si fueran coordenadas geográficas.

Otra opción más utilizada actualmente sería añadir la ubicación para que sólo haya que aportar el lugar de llegada.

ACCESO A GITHUB

https://github.com/PatriiML11/ProyectoWeb/tree/ProyectoWeb-VersionFinal

TUTOR:

Baldomero Sanchez

bsanchezpe@educa.jcyl.es