

TAREA OPCIONAL 1

MENUS.PHP

Primero, en Menus.php, haremos que, si es el usuario es profesor, le aparezca la opción de “Obtener datos pregunta”. Para ello, como no sabemos el rol del usuario, haremos una llamada a la base de datos y guardaremos el tipo:

```
<?php
//Obtener el tipo de estudiante llamando a la BD:
$link = new mysqli($server, $user, $pass, $basededatos);
$sql="SELECT tipo FROM usuario where usuario.email = '". $_GET['email']."'";
$respuesta = mysqli_query($link , $sql);
$row = mysqli_fetch_array($respuesta);
$tipo = $row["tipo"];
?>
```

Ahora, tenemos el tipo de usuario en \$tipo, con lo que, simplemente, hacemos un if y, si el tipo es igual a ‘profe’, mostramos la opción, si no, no.

```
<?php
if ($tipo == "profe"){
    echo "<span><a href='ClientGetQuestion.php?email=$email&img=$foto'>Obtener datos pregunta</a></span>";
}
?>
```

Ahora procedemos con la parte del servidor del servicio.

GETQUESTIONWS.PHP

La parte principal sería casi igual que el ejercicio obligatorio 2, con lo que nos centraremos en explicar las diferencias.

El principal cambio es que la devolución será de un objeto **complejo**, un objeto tipo struct al que llamaremos *questiondetails*. El objeto tendrá tres campos, uno donde rellenaremos el autor, otro donde rellenaremos el enunciado y otro donde rellenaremos la respuesta correcta. Los tres serán de tipo String:

```
$server->wsdl->addComplexType(
    'questiondetails',
    'complexType',
    'struct',
    'all',
    '',
    array(
        'autor' => array('name' => 'autor', 'type' => 'xsd:string'),
        'enunc' => array('name' => 'enunc', 'type' => 'xsd:string'),
        'resco' => array('name' => 'resco', 'type' => 'xsd:string')
    )
);
```

Ahora, hacemos que el tipo devuelto sea el tipo “questiondetails”

```
$server->register('getPregunta', // r
    array('id' => 'xsd:int'), // param
    array('return' => 'tns:questiondetails'),
    'urn:WSCourse',
```

Ahora, definimos la función a la que hemos llamado *getpregunta*.

Primero, cogeremos, si existiera, la pregunta de la base de datos:

```
function getPregunta($x) {  
    include '../php/DbConfig.php';  
    $link = new mysqli($server, $user, $pass, $basededatos);  
    $sql = "SELECT * FROM preguntasconimagen WHERE preguntasconimagen.numpre='$x'";  
    $pregunta = mysqli_query($link, $sql);
```

Si la respuesta tiene 0 filas, significará que no existe, con lo que tal y como dice el enunciado, crearemos el objeto y llenaremos los campos con "":

```
if (mysqli_num_rows($pregunta) == 0) { //Caso id no existe  
    $res = array(  
        'autor'=>'',  
        'enunc'=>'',  
        'resco'=>''  
    );  
    return $res;
```

Si no, los rellenaremos con los valores que obtengamos de la base de datos:

```
}else{  
    $row = mysqli_fetch_array($pregunta);  
    $res = array(  
        'autor'=>$row["email"],  
        'enunc'=>$row["enunc"],  
        'resco'=>$row["resco"],  
    );  
    return $res;  
}
```

Pasamos ahora al cliente.

ClientGetQuestions.php

Crearemos un formulario vacío donde preparamos tres campos <text> donde se almacenarán las respuestas:

```
<form id="fquestion" name="fquestion" action="ClientGetQuestion.php"><?php $email=$_GET["email"]; echo "email=$email"; $img=$_GET["img"]; echo "img=$img"; ?> method="POST">  
<label for="email">Id:</label>  
<input type="text" id="id" name="id" size="30"><br><br>  
<text id="resautor" style="color:blue; font-weight: bold"></text><br><br>  
<text id="resenunc" style="color:blue; font-weight: bold"></text><br><br>  
<text id="resresco" style="color:blue; font-weight: bold"></text><br><br>  
<input type="submit" id="boton" value="Buscar"><br><br>  
</form>
```

Hacemos la llamada al servidor, como la dirección del servidor del servicio dependerá si está en local o en la nube, creamos un *servicesConfig.php* que dependiendo si estamos en local o en nube hará que la llamada tenga un valor u otro (como hacemos con dbConfig):

```
<?php  
$estado = 0; //1 para la nube  
if ($estado ==0) {  
    $verifypass='http://localhost/LABS/Proyecto1SW/php/VerifyPassWS.php';  
    $getQuestion='http://localhost/LABS/Proyecto1SW/php/GetQuestionWS.php';  
    $ip = 'check';  
    $ip2 = 'check';  
}else{  
    $verifypass='https://swmikel-iturria.000webhostapp.com/Proyecto1SW/php/VerifyPassWS.php';  
    $getQuestion='https://swmikel-iturria.000webhostapp.com/Proyecto1SW/php/GetQuestionWS.php';  
    $ip = $_SERVER['HTTP_X_REAL_IP'];  
    $ip2 = 'check';
```

Creamos el objeto cliente con la variable de *servicesConfig.php* y si el id esta "set" (se ha clicado al botón), hacemos la llamada.

```
$soapclient = new nusoap_client($getQuestion);  
if (isset($_POST['id'])) {  
    $pregunta = $soapclient->call('getPregunta', array('x'=>$_POST['id']));
```

Si al campo de autor en la respuesta es igual a vacío, significa que no existe ninguna pregunta con ese id, con lo que se lo haremos saber al usuario mediante un mensaje de error:

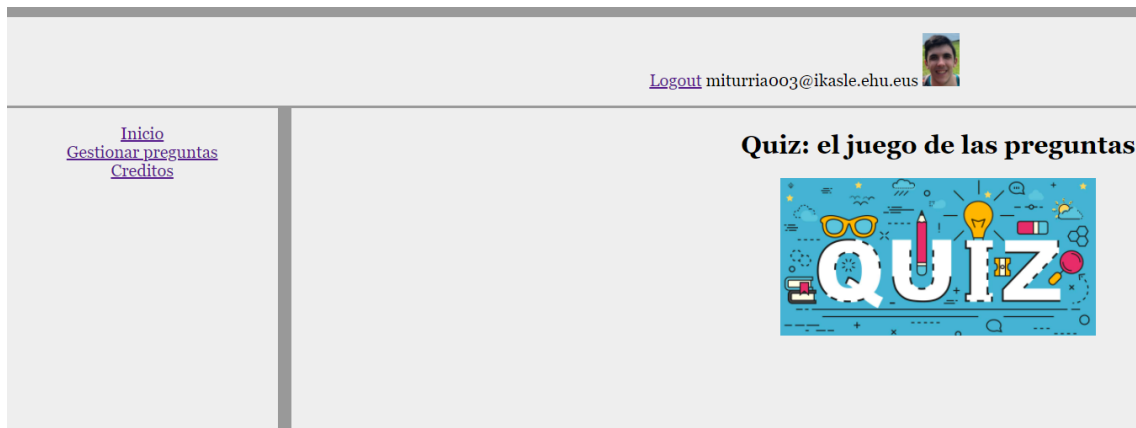
```
$pregunta = $soapclient->call('getPregunta', array('x'=>$_POST['id']));  
if ($pregunta['autor'] == "") {  
    echo "<script>$('#resautor').text('ERROR, no hay pregunta con ese id'); $('#resautor').css({'color':'red','font-weight':'bold'}); </script>";
```

En cambio, si no es igual a vacío, significa que sí existe una pregunta, con lo que imprimimos los valores donde corresponde usando JavaScript:

```
$autor=$pregunta['autor'];  
$enunc=$pregunta['enunc'];  
$resco=$pregunta['resco'];  
echo "<script>$('#resautor').text('Autor: $autor'); </script>";  
echo "<script>$('#resenunc').text('Enunciado: $enunc'); </script>";  
echo "<script>$('#resresco').text('Respuesta correcta: $resco'); </script>";
```

Pruebas de funcionamiento:

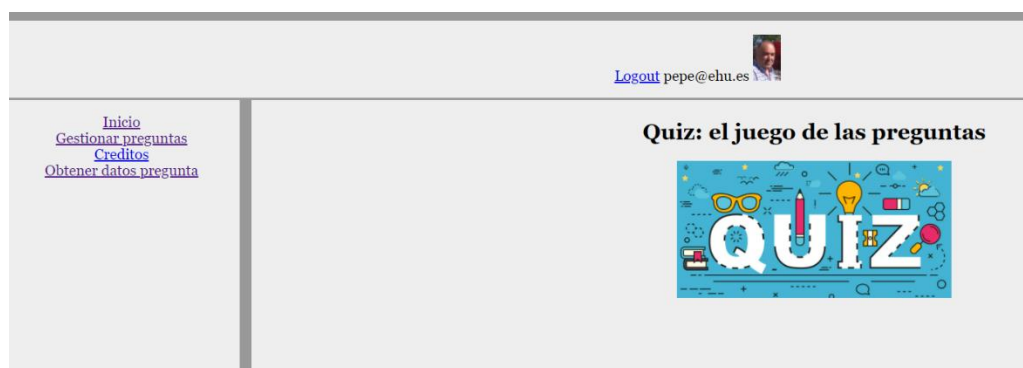
Caso Alumno:



Se puede observar que, en el caso de un alumno no aparece la opción de ver preguntas.

Caso Profesor;

Para las pruebas de profesor hemos usado tu usuario, pepe@ehu.es, ya que es profesor y al tener hecho lo de registro es más complicado crear un profesor (email VIP etc...)

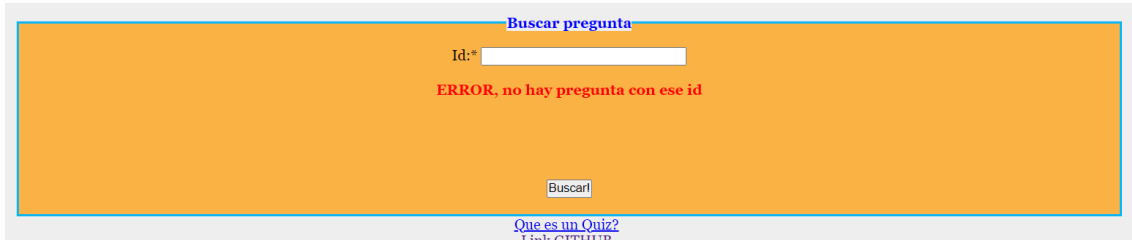


Como se puede observar, aparece el “obtener datos pregunta”.

Probamos las distintas respuestas que puede dar la API

Id inexistente:

Probamos con el id 500:

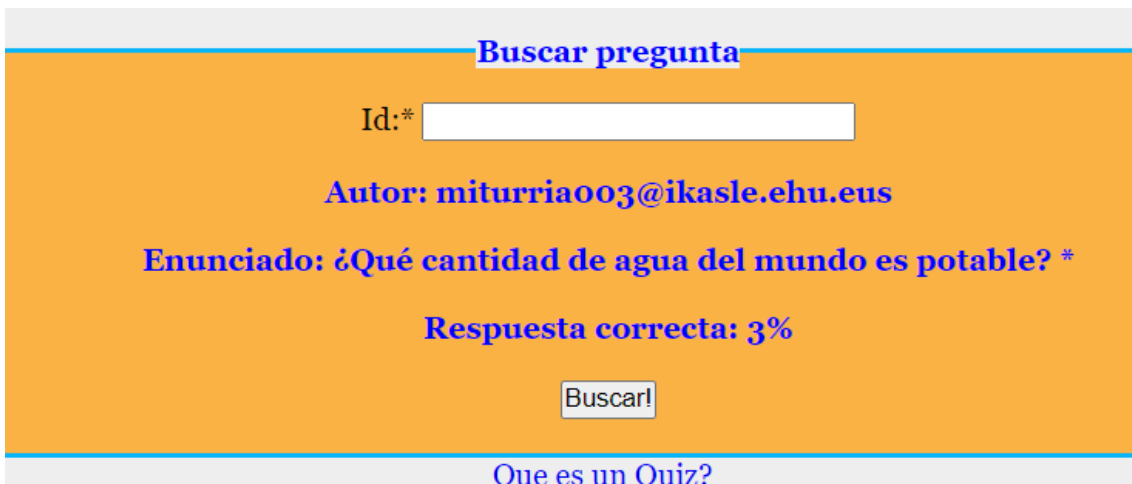


The screenshot shows a web form titled "Buscar pregunta" with a blue header. Below the title is a text input field labeled "Id: *" and a "Buscar!" button. A red error message "ERROR, no hay pregunta con ese id" is displayed in the center. At the bottom, there is a link "Que es un Quiz?" and a "Link GITHUB" below it.

Efectivamente no existe.

Id existente:

Probamos con el id 1:



The screenshot shows the same "Buscar pregunta" form, but now it displays the details for id 1. The "Id: *" field is empty. The "Autor:" field shows "miturria003@ikasle.ehu.eus". The "Enunciado:" field shows "¿Qué cantidad de agua del mundo es potable? *". The "Respuesta correcta:" field shows "3%". The "Buscar!" button is still present. At the bottom, there is a link "Que es un Quiz?".

Ha funcionado correctamente.

NOTAS:

- Después de las pruebas eliminamos el usuario pepe@ehu.es de la base de datos
- Se pueden probar las preguntas desde el id 1 al 5.