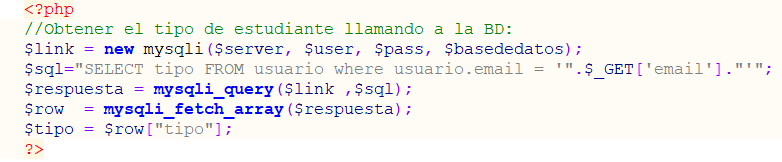
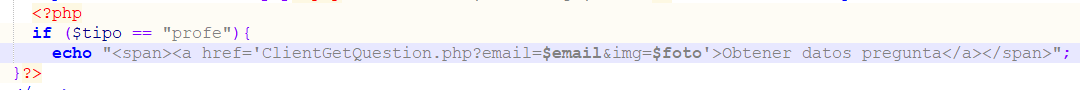
# TAREA OPCIONAL 1

## 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:



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.

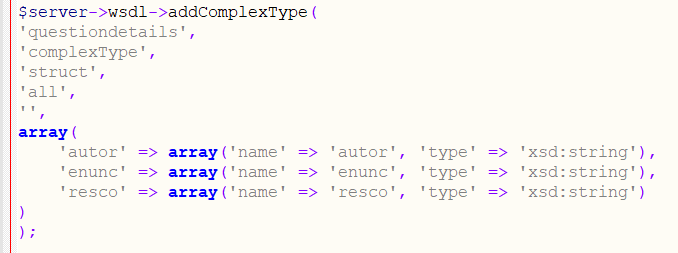


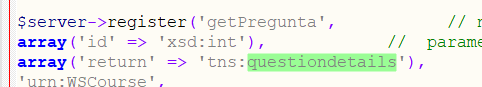
Ahora procedemos con la parte del servidor del servicio.

## GETQUESTIONWS.PHP

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

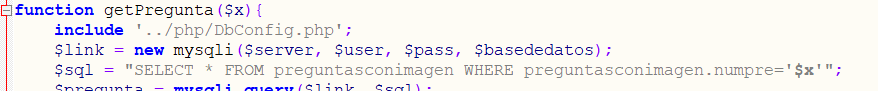
El principal cambio es que la devolución será de un objeto **complejo**¸ un objeto tipo struct al que llamaremos *questiondetails*. El objeto tendrá tres campos, uno donde rellenaremos el autor, otro donde rellenaremos el enunciado y otro donde rellenaremos la respuesta correcta. Los tres serán de tipo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



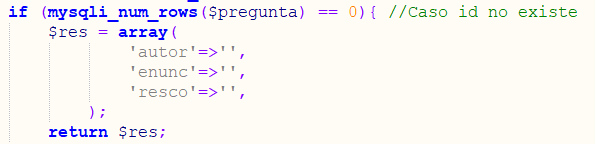
**Ahora, hacemos que el tipo devuelto sea el tipo *“questiondetails”*

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

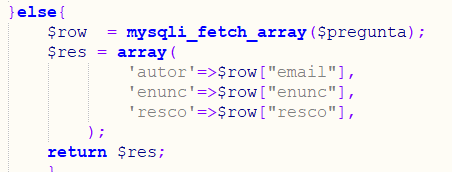
Primero, cogeremos, si existiera, la pregunta de la base de datos:



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 “”:



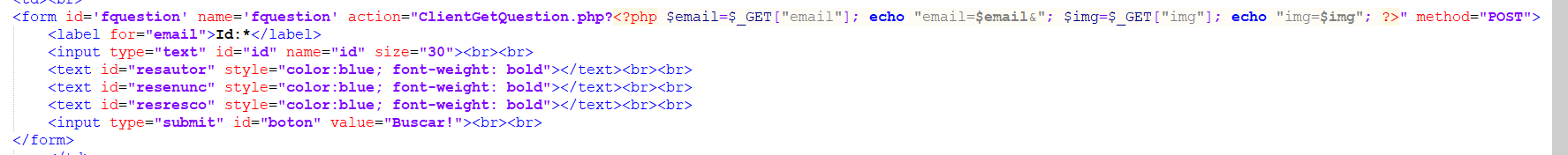
Si no, los rellenaremos con los valores que obtengamos de la base de datos:



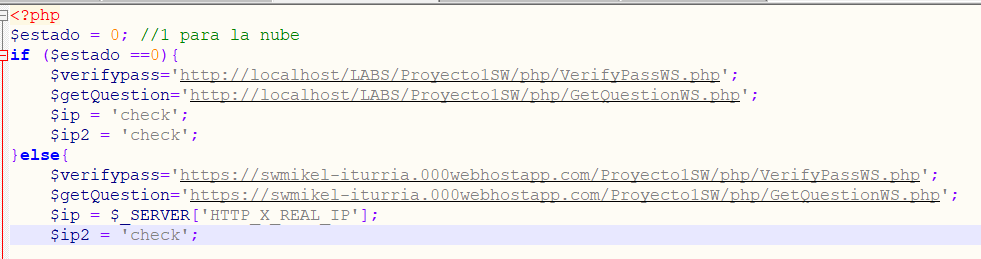
Pasamos ahora al cliente.

## ClientGetQuestions.php

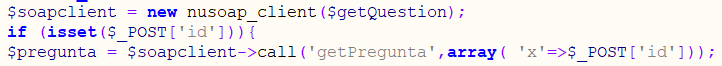
Crearemos un formulario vacío donde preparamos tres campos <text> donde se almacenarán las respuestas:



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):



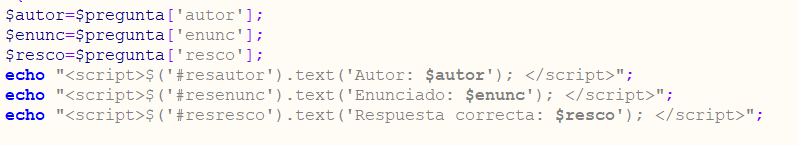
Creamos el objeto cliente con la variable de *servicesConfig.php* y si el id esta “set” (se ha clicado al botón), hacemos la llamada.



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:

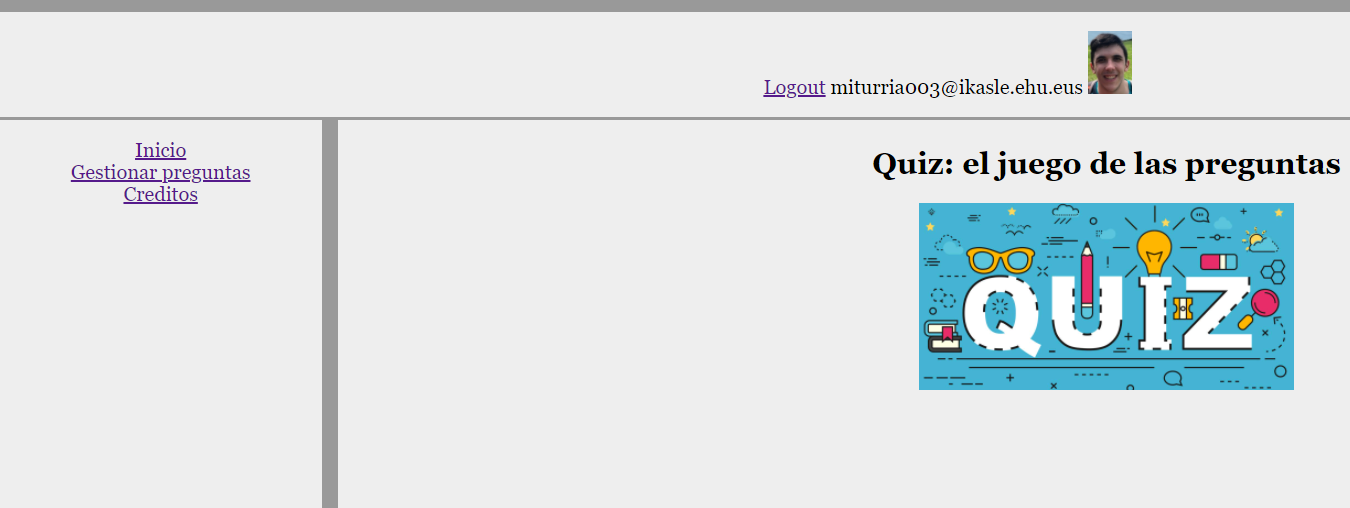


En cambio, si no es igual a vacío, significa que sí existe una pregunta, con lo que imprimimos los valores donde corresponde usando JavaScript:



## Pruebas de funcionamiento:

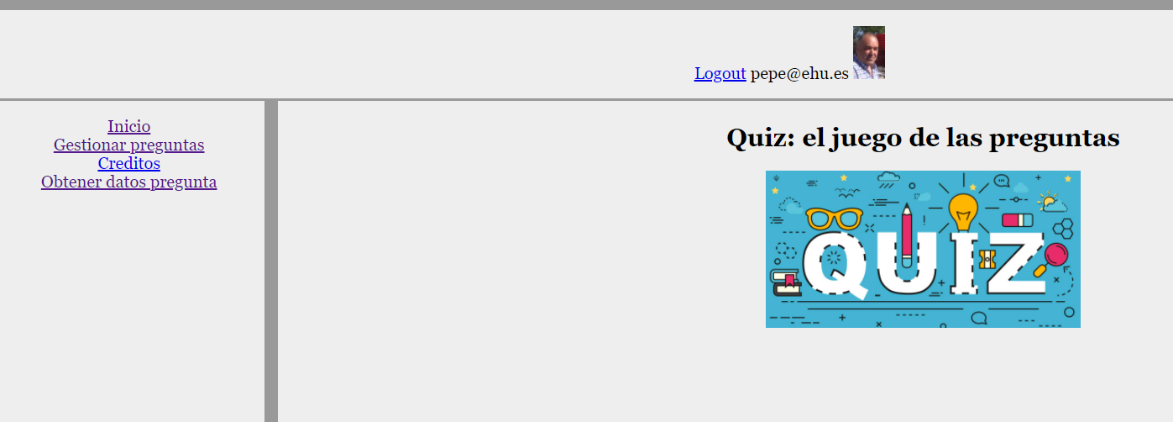
### Caso Alumno:



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

### Caso Profesor;

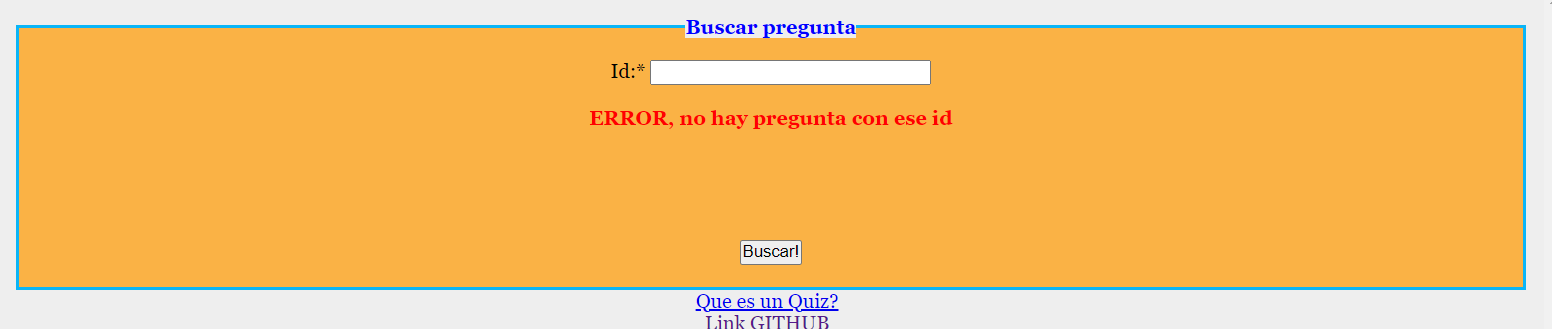
Para las pruebas de profesor hemos usado tu usuario, [pepe@ehu.es](mailto:pepe@ehu.es), ya que es profesor y al tener hecho lo de registro es más complicado crear un profesor (email VIP etc…)



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

Probamos las distintas respuestas que puede dar la API

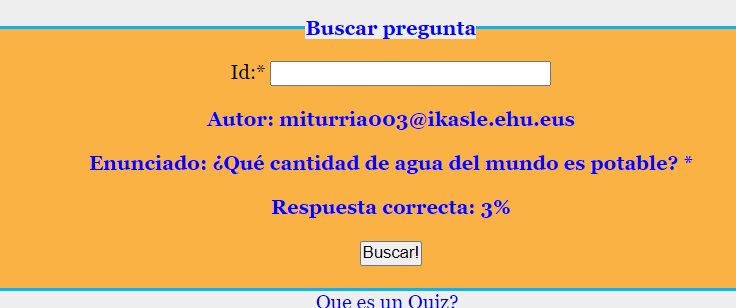
### Id inexistente:

Probamos con el id 500: 

Efectivamente no existe.

### Id existente:

Probamos con el id 1:



Ha funcionado correctamente.

**NOTAS:**

**- Después de las pruebas eliminamos el usuario** [**pepe@ehu.es**](mailto:pepe@ehu.es) **de la base de datos**

**- Se pueden probar las preguntas desde el id 1 al 5.**