FUNCIONES PHP

- A través de esta función conseguimos conectarnos a la base de datos.

```php
//Conexion BBDD
function connect(){
    $dbname = "reto2";
    $host = "db";
    $username = "root";
    $pass = "db123";

    try {
        # MySQL
        $dbh= new PDO("mysql:host=$host;dbname=$dbname", $username, $pass, array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8"));
        $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $dbh;
    }
    catch(PDOException $e) {
        echo $e->getMessage();
        return null;
    }
}
```

- A través de introducir la conexión a la base de datos y la tabla de la que queremos conseguir toda la información, nos devolverá un fetchAll.

```php
function getAll($dbh,$tabla){
    $tabla = strtoupper($tabla);
    switch($tabla)
    {
        case 'CATEGORIA':
            $stmt = $dbh->prepare("SELECT * FROM CATEGORIA");
            break;
        case 'PREGUNTA':
            $stmt = $dbh->prepare("SELECT * FROM PREGUNTA");
            break;
        case 'PREGUNTAR':
            $stmt = $dbh->prepare("SELECT * FROM PREGUNTAR");
            break;
        case 'RESPONDER':
            $stmt = $dbh->prepare("SELECT * FROM RESPONDER");
            break;
        case 'RESPUESTA':
            $stmt = $dbh->prepare("SELECT * FROM RESPUESTA");
            break;
        case 'USUARIO':
            $stmt = $dbh->prepare("SELECT * FROM USUARIO");
            break;
        default:
            throw "Tabla no encontrada";
    }

    $stmt->setFetchMode(PDO::FETCH_OBJ);
    $stmt->execute();

    return $stmt->fetchAll();
}
```

- Funciones para conseguir los datos que necesitamos a través de las vistas creadas anteriormente.

```php
function getVistaPreguntas($dbh) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute();
    return $stmt->fetchAll();
}

function getVistaPregunta($dbh, $id_preg) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas WHERE id_preg = :id_preg");
    $data = array(
        "id_preg" => $id_preg
    );
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute($data);
    return $stmt->fetchAll();
}

function getVistaRespuestas($dbh, $id_preg) {
    $stmt = $dbh->prepare("SELECT * FROM vistaRespuestas WHERE id_preg = :id");
    $data = array(
        "id" => $id_preg
    );
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute($data);
    return $stmt->fetchAll();
}
```

- Ejemplos de funciones necesarias para la **búsqueda avanzada con filtros,** tenemos muchas más ya que hay varias formas de aplicar los filtros**:**

```php
// FUNCIONES PARA LA BUSQUEDA CON FILTROS
function getPreguntasCategoria($dbh, $categoria) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas WHERE id_cat = :id_cat");
    $data = array(
        "id_cat" => $categoria
    );
    $stmt->setFetchMode(PDO::FETCH_OBJ);
    $stmt->execute($data);
    return $stmt->fetchAll();
}

function getPreguntasMasVistas($dbh) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas ORDER BY vistos DESC");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute();
    return $stmt->fetchAll();
}

function getPreguntasMenosVistas($dbh) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas ORDER BY vistos");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute();
    return $stmt->fetchAll();
}

function getPreguntasMasLike($dbh) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas ORDER BY likes DESC");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute();
    return $stmt->fetchAll();
}

function getPreguntasMenosLike($dbh) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas ORDER BY likes");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute();
    return $stmt->fetchAll();
}
```

```php
function getPreguntasMasRespuestas($dbh) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas ORDER BY respuestas DESC");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute();
    return $stmt->fetchAll();
}

function getPreguntasMenosRespuestas($dbh) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas ORDER BY respuestas");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute();
    return $stmt->fetchAll();
}

function getPreguntasMasRecientes($dbh) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas ORDER BY fecha DESC");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute();
    return $stmt->fetchAll();
}

function getPreguntasMenosRecientes($dbh) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas ORDER BY fecha");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute();
    return $stmt->fetchAll();
}

function getPreguntasMasVistasCategoria($dbh, $categoria) {
    $stmt = $dbh->prepare("SELECT * FROM vistaPreguntas WHERE id_cat = :id_cat ORDER BY vistos DESC");
    $data = array(
        "id_cat" => $categoria
    );
    $stmt->setFetchMode(PDO::FETCH_OBJ);
    $stmt->execute($data);
    return $stmt->fetchAll();
}
```

- Funciones para contar Nº de respuestas, likes, votos:

```php
function countRespuestas($dbh,$id_preg) {
    $stmt = $dbh->prepare("SELECT * FROM countRespuestas WHERE id_preg = :id");
    $data = array(
        "id" => $id_preg
    );
    $stmt->setFetchMode(PDO::FETCH_OBJ);
    $stmt->execute($data);
    return $stmt->fetch();
}

function countLikes($dbh,$id_preg) {
    $stmt = $dbh->prepare("SELECT * FROM countLikes WHERE id_preg = :id");
    $data = array(
        "id" => $id_preg
    );
    $stmt->setFetchMode(PDO::FETCH_OBJ);
    $stmt->execute($data);
    return $stmt->fetch();
}

function countVotos($dbh,$id_res) {
    $stmt = $dbh->prepare("SELECT * FROM countVotos WHERE id_res = :id");
    $data = array(
        "id" => $id_res
    );
    $stmt->setFetchMode(PDO::FETCH_OBJ);
    $stmt->execute($data);
    return $stmt->fetch();
}
```

- **INSERTS:**

```php
function insertRespuesta($dbh,$datosRespuesta){
    try {
        //insertar respuesta
        $stmt = $dbh->prepare("INSERT INTO RESPUESTA(descripcion,id_preg)
                               VALUES (:descripcion,:id_preg)");

        $stmt->execute($datosRespuesta);

        //insertar en responder
        $stmt_ = $dbh->prepare("INSERT INTO RESPONDER(id_usu,id_res)
                                VALUES (:usuario, :respuesta)");
        $data = array (
            "usuario" => $_SESSION['id_usu'],
            "respuesta" => $dbh->lastInsertId()
        );
        $stmt_->execute($data);
    } catch(Exception $e) {
        echo 'Exception -> ';
        var_dump($e->getMessage());
    }
}

function insertPregunta($dbh, $datosPregunta){
    try {
        //insertar pregunta
        $stmt = $dbh->prepare("INSERT INTO PREGUNTA(titulo,detalle,id_cat)
                               VALUES (:titulo, :detalle, :categoria)");

        $stmt->execute($datosPregunta);

        //insertar en preguntar
        $stmt_ = $dbh->prepare("INSERT INTO PREGUNTAR(id_usu,id_preg)
                                VALUES (:usuario, :pregunta)");
        $data = array (
            "usuario" => $_SESSION['id_usu'],
            "pregunta" => $dbh->lastInsertId()
        );
        $stmt_->execute($data);
    } catch(Exception $e) {
        echo 'Exception -> ';
        var_dump($e->getMessage());
    }
}
```

- DELETE:

```php
function borrarVoto($dbh,$id_preg) {
    try {
        $stmt = $dbh->prepare("DELETE FROM VOTAR WHERE id_usu = :usuario AND id_res = :respuesta");
        $data = array (
            "usuario" => $_SESSION['id_usu'],
            "respuesta" => $id_preg
        );
        $stmt->execute($data);
    } catch(Exception $e) {
        echo 'Exception -> ';
        var_dump($e->getMessage());
    }
}
```

- UPDATE:

```php
function updateUsuario($dbh) { // UPDATE SIN LA CONTRASEÑA
    try {
        $stmt = $dbh->prepare("UPDATE USUARIO SET nombre = :nombre, apellidos = :apellidos, email = :email WHERE id_usu = :id_usu");
        $data = array (
            "nombre" => $_POST['pnombre'],
            "apellidos" => $_POST['papellidos'],
            "email" => $_POST['pemail'],
            "id_usu" => $_SESSION['id_usu']
        );
        $stmt->execute($data);
    } catch(Exception $e) {
        echo 'Exception -> ';
        var_dump($e->getMessage());
    }
}
```

- LOGIN:

```php
function userLogin($email,$pass){
    try{
        $db = connect();
        $stmt = $db->prepare("SELECT id_usu FROM USUARIO WHERE email=:email AND contrasenia=:pass");
        $data = array(
            "email" => $email,
            "pass" => $pass
        );
        $stmt->execute($data);
        $count = $stmt->rowCount();
        $datos = $stmt->fetch(PDO::FETCH_OBJ);
        $db = null;
        if($count){
            $_SESSION['id_usu']=$datos->id_usu; // Guardamos en sesión el id del usuario
            return true;
        }
        else {
            return false;
        }
    }
    catch(PDOException $e) {
        echo '{"error":{"text":'. $e->getMessage() .'}}';
    }
}
```

- REGISTRO:

```php
function userRegistration($nombre,$email,$pass){
    try{
        $db = connect();
        $stmt = $db->prepare("SELECT id_usu FROM USUARIO WHERE email=:email AND contrasenia=:pass");
        $data = array(
            "email" => $email,
            "pass" => $pass
        );
        $stmt->execute($data);
        $count=$stmt->rowCount();
        if($count<1) {
            $stmt = $db->prepare("INSERT INTO USUARIO(nombre,contrasenia,email) VALUES (:nombre,:pass,:email)");
            $data = array(
                "email" => $email,
                "pass" => $pass,
                "nombre" => $nombre
            );
            $stmt->execute($data);
            $uid=$db->lastInsertId(); // Ultimo id insertado
            $_SESSION['id_usu']=$uid;
            return true;
        }
        else {
            $db = null;
            return false;
        }
    }
    catch(PDOException $e) {
        echo '{"error":{"text":'. $e->getMessage() .'}}';
    }
}
```

- CERRAR SESIÓN:

```php
// CERRAR SESIÓN
function cerrarSesion() {
    unset($_SESSION[ "id_usu"]);
}
```

- Al entrar en nuestra página, a través de la sesiones comprobamos si el usuario ha iniciado sesión.

```php
<?php
    session_start();
    require ('./db_functions.php');

    // SI NO TIENE CREADA UNA SESIÓN...
    if (!isset($_SESSION['id_usu'])) {
        //require('VIEWS/login.view.php');

        // Inicio de sesión
        if (isset($_POST['email']) && isset($_POST['pswd'])) {
            $inicio = userLogin($_POST['email'],$_POST['pswd']); // Si es TRUE es correcto
            if ($inicio) {
                //echo '<p style="color:green">Inicio de sesión hecho correctamente</p>';
                header("Location: index.php?accion=preguntas", TRUE, 301);
                exit();
            } else echo '<p style="color:red">Contraseña o email no validos</p>';
        }

        // Registrarse
        else if (isset($_POST['remail']) && isset($_POST['rpswd']) && isset($_POST['rnombre'])) {
            $registro = userRegistration($_POST['rnombre'],$_POST['remail'],$_POST['rpswd']); // Si es TRUE es correcto
            if ($registro) {
                //echo '<p style="color:green">Registro hecho correctamente</p>';
                header("Location: index.php?accion=preguntas", TRUE, 301);
                exit();
            } else echo '<p style="color:red">Fallo al registrar</p>';
        }
        else {
            require('VIEWS/login.view.php');
        }
```

- Si ya tienes iniciada una sesión…

```php
    } else { // SI TIENE UNA SESIÓN INICIADA
        if (isset($_GET['accion'])) {
            switch ($_GET['accion']) {
                case 'preguntar':
                    $dbh = connect();
                    $infousuario = getUsuario($dbh);
                    require('VIEWS/publicarPregunta.view.php');
                    break;
                case 'detalles':
                    require('VIEWS/detalles.view.php');
                    break;
                case 'perfil':
                    $dbh = connect();
                    $infousuario = getUsuario($dbh);
                    require('VIEWS/perfil.view.php');
                    break;
                case 'cerrarsesion':
                    cerrarSesion();
                    require('VIEWS/login.view.php');
                    break;
                default:
                    require('VIEWS/visualizarPreguntas.view.php');
                    break;
            }
        }
        else {
            require('VIEWS/visualizarPreguntas.view.php');
        }
    }
?>
```

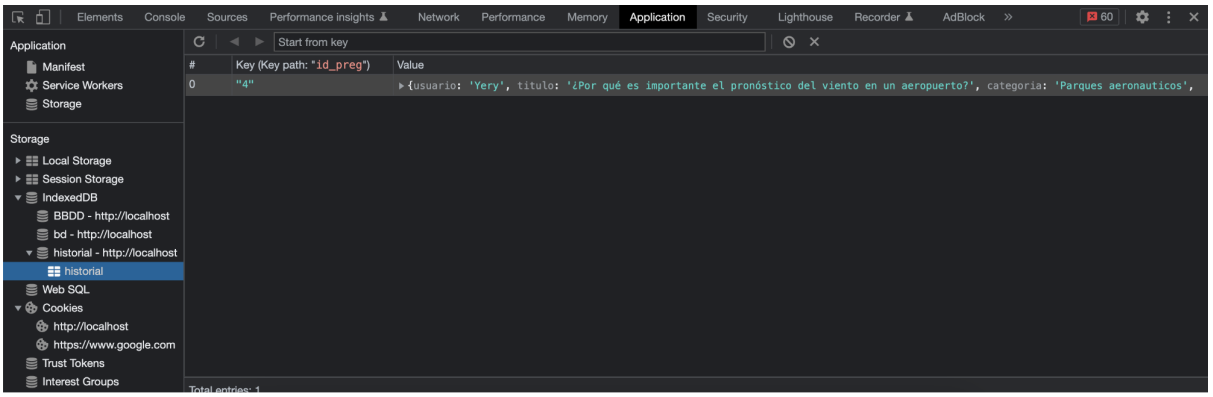
- Funciones API _GET:

```php
//Vamos a comprobar que lo que
$funcion = isset($_GET['funcion']) ? $_GET['funcion'] : null;

//Cada respuesta que nos va a dar la API
$respuesta = [];

switch ($funcion) {
    case 'getPreguntas':
        $respuesta = api_getPreguntas();
        break;
    case 'getDetallesPregunta':
        $respuesta = api_getPregunta();
        break;
    case 'getRespuestas':
        $respuesta = api_getRespuestas();
        break;
    case 'actualizarVisto':
        api_actualizarVisto();
        break;
    case 'insertarLike':
        api_actualizarLike();
        break;
    case 'borrarLike':
        api_borrarLike();
        break;
    case 'insertarVoto':
        api_actualizarVoto();
        break;
    case 'borrarVoto':
        api_borrarVoto();
        break;
    case 'getCategorias':
        $respuesta = api_getCategorias();
        break;
    case 'enviarPregunta':
        api_enviarPregunta();
        break;
    case 'enviarRespuesta':
        api_enviarRespuesta();
        break;
    default:
        break;
}
```

JAVASCRIPT

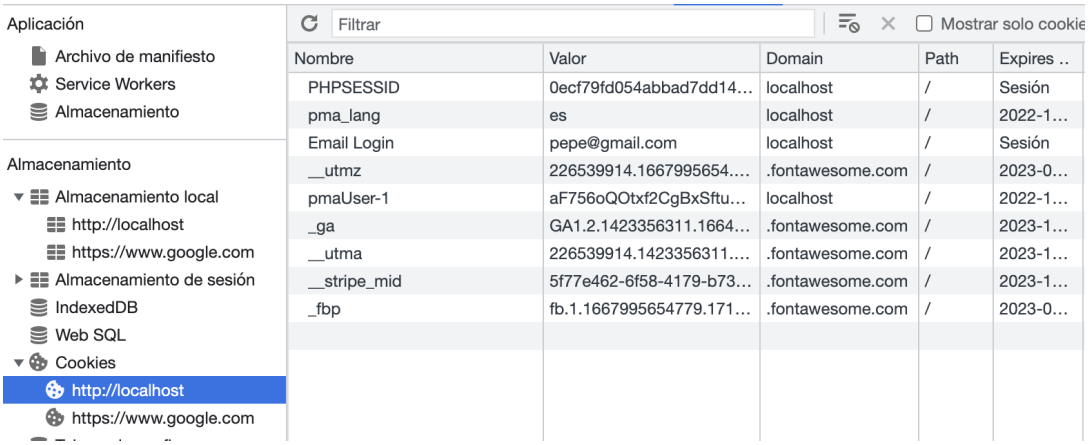
- IndexedDB:

A la hora de hacer una búsqueda por filtros, se crearía una base de datos IndexedDB con dicha información.



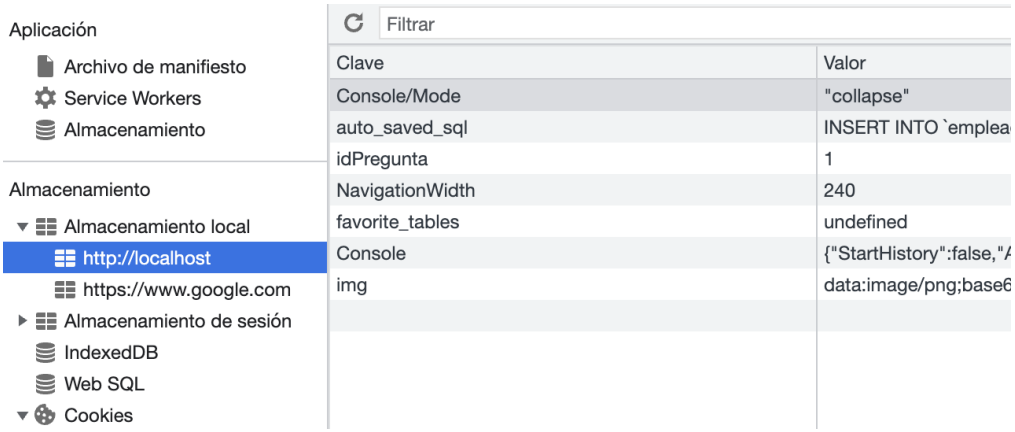
- Cookies:

Al iniciar sesión, guardamos en cookies la información de dicho usuario, asi como el correo electrónico, apellidos… (datos no sensibles). Para asi luego poder autocompletar campos automáticamente sin necesidad de acceder a la base de datos otra vez.



- LocalStorage:

Aqui guardaremos la imagen del usuario, id de la pregunta… (datos que solo le interesan al usuario).



- **JS para cargar pregunta:**

En la variable “datosPregunta” tenemos guardada toda la información que sacamos de una vista creada anteriormente. A través de DOM conseguimos crear a nuestro gusto el contenido para la pagina.

```
//Funcion cargar vista de los detalles de la pregunta:
function cargarLayout(datosPregunta) {
    let contenedorPregunta = document.getElementsByClassName("zonaPregunta")[0];
    let pregunta = document.createElement('div');
    pregunta.classList.add('detalles');
    pregunta.classList.add('recuadro');

    pregunta.innerHTML = `
            <div class='votacion'>
                <a class='like' onclick=\"insertarLike('${datosPregunta.id_preg}')\"><i class='fa-solid fa-sort-up'></i></a>
                <b id='likes' class='votos'>${datosPregunta.likes} LIKE</b>
                <a class='like'onclick=\"borrarLike('${datosPregunta.id_preg}')\"><i class='fa-solid fa-sort-down'></i></a>
            </div>
            <div class='user'>
                <h3 class='titulousuario' id='titulousuario'>${datosPregunta.usuario}</h3>
                <img class='perfil' src='../RECURSOS/IMAGES/user.png' alt='Foto de pergil'>
                <form>
                <p class='clasificacion'>
                    <input id='radio1' type='radio' name='estrellas' value='5' disabled ${setValoracion(datosPregunta.valoracion,5)}>
                    <label for='radio1'>★</label>
                    <input id='radio2' type='radio' name='estrellas' value='4' disabled ${setValoracion(datosPregunta.valoracion,4)}>
                    <label for='radio2'>★</label>
                    <input id='radio3' type='radio' name='estrellas' value='3' disabled ${setValoracion(datosPregunta.valoracion,3)}>
                    <label for='radio3'>★</label>
                    <input id='radio4' type='radio' name='estrellas' value='2' disabled ${setValoracion(datosPregunta.valoracion,2)}>
                    <label for='radio4'>★</label>
                    <input id='radio5' type='radio' name='estrellas' value='1' disabled ${setValoracion(datosPregunta.valoracion,1)}>
                    <label for='radio5'>★</label>
                </p>
            </form>
            </div>
            <div class='info'>
                <h1>${datosPregunta.titulo}</h1>
                <p><b>Usuario:</b> ${datosPregunta.usuario}</p>
                <p><b>Fecha de publicación:</b> ${datosPregunta.fecha}</p>
                <p><b>Departamento:</b> ${datosPregunta.categoria}</p>
                <div class='descripcion recuadro'>
                    ${datosPregunta.detalle}
                </div>
            </div>`;

    contenedorPregunta.appendChild(pregunta);
```

- **JS para cargar todas las preguntas:**

Es muy parecido al de arriba, pero con alguna función extra mas (para poder ordenarlos a gusto del usuario, buscar por nombre…). Esta es la función mas relevante: (es un bucle que hace que se creen las preguntas que ha solicitado el usuario)

```
//Funcion para cargar los detalles de la pregunta desde la vista creada en la BBDD
cargarPreguntas()
    .then( function(resultadoPromesa) {
        if (resultadoPromesa.mensaje) { // != undefined
            console.error(resultadoPromesa);
        } else {
            resultadoPromesa.forEach(datosPregunta => {
                cargarLayoutPregunta(datosPregunta);
            });
        }
    }
);
```

- **Validaciones de formularios:**

```
function validarPassword() {
    //Datos de entrada:
    let pass = iPassword.value;
    //Expresiones regulares:
    let exRegPass = new RegExp("^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#\$%\^&\*])(?=.{8,})");

    /* Explicación de la expresión regular de la contraseña:
        ->(?=.*[az])        La cadena debe contener al menos 1 carácter alfabético en minúscula.
        ->(?=.*[AZ])        La cadena debe contener al menos 1 carácter alfabético en mayúscula.
        ->(?=.*[0-9])       La cadena debe contener al menos 1 carácter numérico.
        ->(?=.*[!@#$%^&*])  La cadena debe contener al menos un carácter especial, pero estamos escapando de los caracteres RegEx reservados para evitar conflictos.
        ->(?=.{8,})         La cadena debe tener ocho caracteres o más. */

    if (exRegPass.test(pass)){
        passwordIncorrecto.hidden = true;
    } else {
        iPassword.focus();
        passwordIncorrecto.hidden = false;
        throw "Problemas contraseña";
    }
}
```

```
function matchPassword(){
    let pass1 = document.getElementById("contra1").value;
    let pass2 = document.getElementById("contra2").value;
    if(pass1 != pass2){
        document.getElementById("passIncorrecta").hidden = false;
    } else {
        document.getElementById("passIncorrecta").hidden = true;
    }
}

function validarNombre() {
    let nom = document.getElementById('nombre').value;
    let exRegName  =  new RegExp (/^[A-Z]{1}[a-z]+$/);
    if (!exRegName.test(nom)){
        document.getElementById("nombre").focus();
        document.getElementById("nombreIncorrecto").hidden = false;
    } else {
        document.getElementById("nombreIncorrecto").hidden = true;
    }
}

function validarApellidos() {
    let apellido = document.getElementById('apellidos').value;
    let exRegName  =  new RegExp (/^[A-Z]{1}[a-z]+$/);
    if (!exRegName.test(apellido)){
        document.getElementById("apellidos").focus();
        document.getElementById("apellidosIncorrectos").hidden = false;
    } else {
        document.getElementById("apellidosIncorrectos").hidden = true;
    }
}

function validarEmail() {
    let email = document.getElementById('email').value;
    let exRegEmail =  new RegExp ('^[_a-z0-9-]+(.[_a-z0-9-]+)*@[a-z0-9-]+(.[a-z0-9-]+)*(.[a-z]{2,4})$');
    if (!exRegEmail.test(email)){
        document.getElementById("email").focus();
        document.getElementById("emailIncorrecto").hidden = false;
    } else {
        document.getElementById("emailIncorrecto").hidden = true;
    }
}
```

- **Guardar foto de perfil en LocalStorage:**

```
/* GUARDAR LA IMAGEN EN LOCALSTORAGE */
function handleFileSelect(evt) {

  var files = evt.target.files; // FileList object
  localStorage.removeItem('img'); // Para que no hayan 2 imagenes

  // Loop through the FileList and render image files as thumbnails.
  for (var i = 0, f; f = files[i]; i++) {

    // Solo procesa imagenes.
    if (!f.type.match('image.*')) {
      continue;
    }

    var reader = new FileReader();

    // Closure to capture the file information.
    reader.onload = (function(theFile) {
      return function(e) {
        // Creamos un span con la imagen
        var span = document.createElement('span');
        span.innerHTML = ['<img id="fperfilnueva" class="perfil" src="', e.target.result,
          '"/>'
        ].join('');

        document.getElementById('list').insertBefore(span, null);
        localStorage.setItem('img', e.target.result);
      };
    })(f);

    // Read in the image file as a data URL.
    reader.readAsDataURL(f);
    document.getElementById('fotoperfil').hidden = true;
  }
}

// Si tenenemos una imagen en LocalStorage...
if (localStorage.img) {
  var span = document.createElement('span');
  span.innerHTML += ['<img class="perfil" src="', localStorage.img,
    '"/>'
  ].join('');

  document.getElementById('list').insertBefore(span, null);
} else {
  // Ocultamos la imagen por defecto del usuario
  document.getElementById('fotoperfil').hidden = false;
}
```