Práctica 2

URL: https://sertel.shortnim.me/p2/index.html

Código: https://github.com/Albermonte/SerTel/tree/main/p2

Apartado 1

Para la comprobación de los valores introducidos en el formulario se ha usado una función que será llamada al hacer click en el botón “Submit”.

Primero se define globalmente los elementos correspondientes a cada campo para poder ser reusados más adelante. También se ha deshabilitado el comportamiento por defecto del formulario para que solo se envíe si todo es correcto.

La función encargada de comprobar los valores de los campos guarda dentro de constantes el valor de cada elemento, se comprueba que todos tienen un valor “truish”, es decir que son verdaderos, esto incluye no ser “undefined”.

Si se cumple estás reglas se envía el formulario y se informa al usuario, de lo contrario se le pide que rellene todos los campos

```
// Prevent form from reloading page
    const form = document.getElementById("form");
    const handleForm = (event) => {
      event.preventDefault();
    };
    form.addEventListener("submit", handleForm);

// Check if any input is empty
    const countryElement = document.getElementById("country");
    const cityElement = document.getElementById("city");
    const gpsElement = document.getElementById("gps");
    const mapUrlElement = document.getElementById("url");
    const pictureElement = document.getElementById("picture");
    const passElement = document.getElementById("pass");

    const submitData = () => {
      const countryValue = countryElement.value;
      const cityValue = cityElement.value;
      const gpsValue = gpsElement.value;
      const mapUrlValue = mapUrlElement.value;
      const pictureValue = pictureElement.value;
      const passValue = passElement.value;

      if (
        countryValue &&
        cityValue &&
        gpsValue &&
        mapUrlValue &&
```

```
        pictureValue &&
        passValue
      ) {
        // If all is correct submit the form
        form.submit();
        alert("Form submitted");
      } else alert("Rellena todos los campos");
    };
```

En el caso de la comprobación del campo “country” se ha optado por usar el evento “onblur” para llamar a la función una vez que el usuario abandone el campo.

A esta función se le pasa el valor introducido en el campo “country”, se almacena en un objeto el “datalist” con todas las posibles opciones y se comprueba si el valor introducido por el usuario concuerda con alguno de las posibles opciones. Si esto no es así se muestra una alerta al usuario.

```
<input list="countries"
       id="country"
       name="country"
       onblur="checkCountry(this.value)"
/>
```

```
    // Country
    const checkCountry = (value) => {
      // Save all options from datalist into an object
      const options = document.getElementById("countries").options;
      // Initialize as false to have only an if and not an else
      let ok = false;
      // Search on the object and check if the values from the options an
d the one from the user are the same
      for (let i = 0; i < options.length; i++) {
        // All uppercase and remove spaces to let user type "reino unido"
 or "ReinoUnido"
        if (
          options[i].value.toUpperCase().replace(/\s+/g, "") ===
          value.toUpperCase().replace(/\s+/g, "")
        )
          ok = true;
      }
      // If theres no coincidence show an alert
      if (!ok) alert("País Erroneo");
    };
```

Para el campo de temperatura se ha hecho la llamada a la función de verificación desde la función que calcula la variación para aprovechar los elementos usados anteriormente y el evento que llama a la función “calculateTempVar”.

Para empezar guardamos los valores de las temperaturas mínima y máxima en sus respectivas variables, se pone todo el texto en verde, asumiendo que es correcto lo introducido por el usuario, y después se comprueba por separado si las dos variables anteriores cumplen con los criterios. Si es así no pasa nada ya que el texto ya estaba en verde, si no es correcto alguno de los valores este se pone en rojo y se muestra un alert.

```
// Temperature Variation
    const calculateTempVar = () => {
      const tempMinElement = document.getElementById("tempMin");
      const tempMin = tempMinElement.value;
      const tempMaxElement = document.getElementById("tempMax");
      const tempMax = tempMaxElement.value;
      const tempVar = tempMax - tempMin;
      if (tempVar < 0) {
        alert("La temperatura mínima no puede ser mayor que la máxima");
        tempMinElement.value = tempMax - 1;
        tempMaxElement.value = 1;
      } else document.getElementById("tempVar").value = tempVar;
      // Call here the function to take advantage of already having tempM
inElement and tempMaxElement
      verifyTemp(tempMinElement, tempMaxElement);
    };

    // Temperature Between 22 and 26
    const verifyTemp = (tempMinElement, tempMaxElement) => {
      const tempMin = tempMinElement.value;
      const tempMax = tempMaxElement.value;
      // Ponemos todo el texto en verde
      tempMinElement.classList.remove("text-red");
      tempMaxElement.classList.remove("text-red");
      tempMinElement.classList.add("text-green");
      tempMaxElement.classList.add("text-green");
      // En esta ocasión asumimos que está correcto al principio y lo cam
biamos si no lo está
      let ok = true;
      // Si alguna de las temperaturas está mal se pone en rojo
      if (tempMin < 22 || tempMin > 26) {
        tempMinElement.classList.add("text-red");
        ok = false;
      }
      // No se usa else if porque pueden estar ambas mal
      if (tempMax < 22 || tempMax > 26) {
        tempMaxElement.classList.add("text-red");
        ok = false;
      }
      // No se pone alert dentro de los otros if para evitar que salga 2
veces si ambas temperaturas están mal
      if (!ok) alert("La temperatura tiene que estar entre 22 y 26 ºC");
    };
```

Para el campo de contraseña lo primero que se hace es añadir un “listener” para escuchar a todo input del usuario y comprobarlo al momento.

La comprobación de la seguridad se calcula con respecto al número de caracteres, si es menor o igual a 6 es insegura, si es mayor la seguridad depende del número de caracteres dividido entre 2, por lo que se puede tener contraseñas por debajo del 5 aunque sean “seguras”. Si la fuerza de la contraseña supera el valor máximo de 10 solo se mostrará 10, por lo tanto la mayor fuerza de una contraseña será 10 (20 caracteres o más).

Tanto para una contraseña insegura como para la segura se mostrará un texto avisando al usuario de la seguridad de su contraseña.

```
// Password verification
    passElement.addEventListener("input", (e) => {
      // Escuchar a cada input del teclado y comprobar
      verifyPass(e.target.value);
    });

    const verifyPass = (pass) => {
      const passLength = pass.length;
      const passStrengthElement = document.getElementById("passStrength")
;
      let ok = true;
      if (passLength <= 6) {
        ok = false;
        passStrengthElement.innerHTML = "Contraseña insegura";
      } else {
        // Seguridad = numero caracteres /2
        const strength = passLength / 2;
        // La seguridad no va a ser mayor de 10
        passStrengthElement.innerHTML = `Contraseña de calidad ${
          strength > 10 ? 10 : strength
        }`;
      }

      // Si todo esta ok texto en verde, si hay algun problema en rojo
      if (ok) {
        passStrengthElement.classList.remove("text-red");
        passStrengthElement.classList.add("text-green");
      } else {
        passStrengthElement.classList.remove("text-green");
        passStrengthElement.classList.add("text-red");
      }
    };
```

Apartado 2

Aquí también se ha optado por usar un “listener” para escuchar a todos los input del usuario, normalmente copiará y pegará las coordenadas por lo que es igual.

Para poder convertir lo introducido por el usuario a algo entendible por los mapas de Google se eliminan los espacios usando la función “replace” y usando RegEx para encontrar todos los espacios de la cadena de texto.

Se separa la cadena usando la coma como carácter de separación, así se obtiene la latitud y la longitud.

Si la latitud contiene la “S” de Sur (“O” para longitud) se reemplaza esta “S” por un signo negativo, si no es así se borra la “N” (“E” para longitud).

Si las coordenadas están en modo decimal no contendrán estas letras, por lo tanto no es necesario hacer ningún cambio.

Estas dos variables se añaden a la query de la API de Google Maps y se asigna esta URL al valor del campo “Mapa URL”.

```
// GPS Coordinates to URL
    gpsElement.addEventListener("input", (e) => {
      // Save value without spaces
      const value = e.target.value.replace(/\s+/g, "");
      // Separar latitud y longitud usando el caracter ,
      const separate = value.split(",");
      let latitude = separate[0];
      let longitude = separate[1];
      // Check if we need to add negative value, if not just remove the l
etter (doesn't work with letters)
      if (latitude.includes("S")) latitude = latitude.replace("S", "-");
      else latitude = latitude.replace("N", "");
      if (longitude.includes("O")) longitude = longitude.replace("O", "-
");
      else longitude = longitude.replace("E", "");
      // Assign result to Mapa URL input
      mapUrlElement.value = `https://www.google.com/maps/search/?api=1&qu
ery=${latitude},${longitude}`;
    });
```

Se ha incluido una mejora por la cual se pueden obtener estas coordenadas de forma automática usando la API de Geolocalización que traen los navegadores (será más precisa en móviles con el GPS activado).

Para esto, primero se comprueba si esta API está disponible, si no lo está se avisa al usuario y si lo está se procede con la geolocalización.

La función “getCurrentPosition” devuelve un objeto que contiene las coordenadas para latitud y longitud, solo queda asignar estos valores a los campos “Coordenadas GPS” y “Mapa URL”.

```
// Geolocation
    const geolocate = () => {
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(function (position) {
          const latitude = position.coords.latitude;
          const longitude = position.coords.longitude;
          gpsElement.value = `${latitude},${longitude}`;
          mapUrlElement.value = `https://www.google.com/maps/search/?api=
1&query=${latitude},${longitude}`;
        });
      } else {
        alert("Geolocalización no disponible");
      }
    };
```

Apartado 3

Para la verificación del correo electrónico se usa un “listener” que llamará la función “checkEmailRX” o “checkEmailJS” (dependiendo si se quiere usar JS o RegEx) en cada “input” del usuario para comprobar si el email es correcto.

```
// Email Listener
    const emailElement = document.getElementById("femail");
    emailElement.addEventListener("input", checkEmailRX);
```

Para el caso de JS se usa el carácter “@” para diferenciar el nombre del usuario del resto y el carácter “.” para separar el dominio del TLD.

Estos valores se almacenan en sus respectivas variables, si alguna de las variables está vacía se devolverá un mensaje indicando que el email no es válido.

Debido a que mientras el usuario está escribiendo puede no haber introducido aún el carácter “@” o “.” las variables “str[1]” y “domainTLD” puedes no estar definidas, lo que lanzaría un error que atraparemos con “try catch” e indicaremos que el email no es válido.

```
// Check email only JS
    const checkEmailJS = ({ target: { value } }) => {
      let ok = false;
      try {
        // Split using @ character
        const str = value.split("@");
        const username = str[0];
```

```
        // Split the second part using . character to separate domain fro
m TLD
        const domainTLD = str[1].split(".");
        const domain = domainTLD[0];
        const tld = domainTLD[1];
        // If nothing is undefined it's correct
        if (username && domain && tld) ok = true;
        else ok = false;
      } catch (e) {
        ok = false;
      }

      if (ok) {
        resultElement.classList.remove("text-
red"); // Removing a class that does not exist, does NOT throw an error
        resultElement.classList.add("text-green");
        resultElement.innerHTML = "Email válido";
      } else {
        resultElement.classList.add("text-
green"); // Removing a class that does not exist, does NOT throw an error
        resultElement.classList.add("text-red");
        resultElement.innerHTML = "Email inválido";
      }
    };
```

Para el caso en el que se usa RegEx directamente se usa la función “match” para comprobar si existe una coincidencia en la cadena de caracteres para la expresión regular usada.

Se buscan palabras incluidos puntos, seguidos de un @ y terminando con una palabra de longitud entre 2 y 4 caracteres.

```
// Check email RegEx
    const checkEmailRX = ({ target: { value } }) => {
      /**
       * ^      : String start
       * \w     : Match any word character
       * \.     : Match the . character
       * +      : Match all times posible
       * @      : Match the @ character
       * {2,4} : Match between 2 and 4 characters, TLD domains are .co, .
com, .tech for example
       */
      const ok = value.match(/^[\w\.]+@([\w]+\.)+[\w]{2,4}$/gm);

      if (ok) {
        resultElement.classList.remove("text-
red"); // Removing a class that does not exist, does NOT throw an error
        resultElement.classList.add("text-green");
        resultElement.innerHTML = "Email válido";
```

```
      } else {
        resultElement.classList.add("text-
green"); // Removing a class that does not exist, does NOT throw an error
        resultElement.classList.add("text-red");
        resultElement.innerHTML = "Email inválido";
      }
    };
```

Apartado 4

Las mejores añadidas han sido:

- La mencionada y explicada anteriormente, la cual consiste en detectar la localización automáticamente.
- Efecto Marquee en la página principal en el apartado de información.
- Corregido el color de contraste en la tabla para que sea más visible al hacer hover.
- Un efecto para el puntero usando CSS y detectando la posición del mouse (siguiendo el tutorial: https://www.youtube.com/watch?v=rfpRZ2t_BrQ).