Apartado 1.1

jQuery UI

Librería basada en jQuery centrada en la interfaz de usuario, recopila efectos, widgets y temas para facilitar la experiencia de usuario ahorrándote la programación que va detrás de los componentes.

Kendo UI

Misma idea que jQuery UI pero con una cantidad de componentes mucho mayor y con soporte para móviles.

Bootstrap

Librería basada en jQuery y Popper.js, diseñada para ser “mobile-first” con multitud de componentes, plugins y utilidades. Capacidad de usar Sass, con iconos y temas propios.

La librería front-end más completa y más al día.

Apartado 1.2

URL: https://sertel.shortnim.me/p3/galeria.html

Código: https://github.com/Albermonte/SerTel/blob/main/p3/galeria.html

Se ha optado por poner botones en los laterales del contenedor, tapando parcialmente las imágenes pero dejándolas ver usando una opacidad baja. Independientemente del número de imágenes siempre habrá 3 visibles.

Si se llega al principio o al final de la galería, está volverá al final o al principio respectivamente.

Al hacer hover sobre una imagen aparecerá un botón que permitirá abrir la imagen.

Apartado 1.3

URL: https://sertel.shortnim.me/p3/tablas.html

Código: https://github.com/Albermonte/SerTel/blob/main/p3/tablas.html

Para evitar tener los datos “hardcoded” y hacer el archivo muy grande se ha elegido generarlos aleatoriamente al cargar la página, así también da la sensación de que los datos son obtenidos desde el servidor.

Código del generador de datos aleatorios:
https://github.com/Albermonte/SerTel/blob/3f2972e4ffecc7708c964e4841b98cf3fbd910a6/p3/tablas.html#L200-L210

Las 3 primeras páginas son estáticas, mientras la página 4 y 5 se obtienen usando AJAX, explicado en el apartado 2.2.

Se ha usado jQuery para eliminar las filas de datos actuales

```
$("table").find("tr").last().remove();
```

E intercambiarlas por las nuevas final con los nuevos datos, cada uno dependiente de en qué página estemos

```
for (
let i = actualPage * tableRowsNumber;
i < actualPage * tableRowsNumber + tableRowsNumber;
i++
) {
      const element = `
      <tr>
       <td>${cities[i].name}</td>
       <td>${cities[i].temp}</td>
       <td>${cities[i].hum}</td>
       <td>${cities[i].noise}</td>
       <td>${cities[i].light}</td>
       <td>${cities[i].color}</td>
      </tr>
      `;
      $("table").append(element);
}
```

Apartado 1.4

Para este apartado se podría haber usar librerías como https://datatables.net/ para la tabla o similares para la galería, las cuales ya tienen todo programado y solo hay que añadir pocas cosas.

Se ha implementado una animación al cambiar de página en las tablas para que se note el cambio de los datos y sea más suave.

También se ha añadido la opción de elegir el número de filas que contiene la tabla y qué página mostrar desde el principio, opciones que podrían ser cambiadas por el usuario a través de un input y guardadas usando localStorage para sesiones posteriores.

También se han incluido mejoras menores como el puntero en la página de la galería, la marquesina en la página principal y el botón de “Auto Locate” en la página de configuración.

Apartado 2.1

URL: https://sertel.shortnim.me/p3/config.html

Código: https://github.com/Albermonte/SerTel/blob/main/p3/config.html

Se ha usado la función “getJSON” de jQuery para facilitar la extracción de los datos de las ciudades desde un archivo JSON guardado en el servidor.

Los datos se encuentran en el archivo “data/cities.json” con varias ciudades por cada país. Dependiendo de la opción de país elegida por el usuario se guardan las ciudades de este país en un array para posteriormente sustituir las opciones del “datalist” anterior (vacío si es la primera vez) por las nuevas opciones (una por cada ciudad) del nuevo país elegido.

```
        // Get cities from server
        $.getJSON("data/cities.json", (cities) => {
          let options = "";
          const citiesArray = eval(`cities.${country}`);
          // Translate array to options for datalist
          citiesArray.forEach((element) => {
            options += `<option value="${element}" />`;
          });
          const citiesElement = document.getElementById("cities");
          // Change every option for new ones
          citiesElement.innerHTML = options;
        });
```

Apartado 2.2

URL: https://sertel.shortnim.me/p3/tablas.html

Código: https://github.com/Albermonte/SerTel/blob/main/p3/tablas.html

La página 4 es la encargada de obtener los datos para la tabla desde el archivo “data/citiesData.xml” alojado en el servidor. Usando la función “ajax” de jQuery para obtener este fichero e indicando que es un archivo xml en el “dataType”.

Al igual que para el resto de páginas se eliminan las filas anteriores y se sustituyen por las nuevas. En esta ocasión se ha usado jQuery para obtener los elementos “Name”, “Temp”, etc. de cada ciudad debido a que es mucho más rápido y sencillo que con JS vanilla.

```
$.ajax({
        type: "GET",
        url: "data/citiesData.xml",
        dataType: "xml",
      }).done((xmlDoc) => {
        for (let i = 0; i < tableRowsNumber; i++) {
          $("table").find("tr").last().remove();
        }
```

```
    for (let i = 0; i < tableRowsNumber; i++) {
      const city = xmlDoc.getElementsByTagName("City")[i];
      const element = `
        <tr>
          <td>${$(city).find("Name").text()}</td>
          <td>${$(city).find("Temp").text()}</td>
          <td>${$(city).find("Hum").text()}</td>
          <td>${$(city).find("Noise").text()}</td>
          <td>${$(city).find("Light").text()}</td>
          <td>${$(city).find("Color").text()}</td>
          </tr>
          `;
      $("table").append(element);
    }
```

La página 5 es la elegida para obtener los datos a través de un archivo JSON que se encuentra en el servidor (“data/citiesData.json”).

Funciona de la misma forma que para xml, cambiando “dataType” por “json” y la url por la propia del archivo JSON. Al ser un archivo JSON no necesita ser “parseado” para obtener las propiedades “name”, “temp”, etc. de cada elemento.

El archivo JSON alojado en el servidor es un array que contiene a todas las ciudades para que sea más sencillo de usar.

```
  $.ajax({
    type: "GET",
    url: "data/citiesData.json",
    dataType: "json",
  }).done((jsonDoc) => {
    for (let i = 0; i < tableRowsNumber; i++) {
      $("table").find("tr").last().remove();
    }
    for (let i = 0; i < tableRowsNumber; i++) {
      const element = `
        <tr>
          <td>${jsonDoc[i].name}</td>
          <td>${jsonDoc[i].temp}</td>
          <td>${jsonDoc[i].hum}</td>
          <td>${jsonDoc[i].noise}</td>
          <td>${jsonDoc[i].light}</td>
          <td>${jsonDoc[i].color}</td>
          </tr>
          `;
      $("table").append(element);
    }

    $("table").fadeToggle(750);
  });
```

Apartado 2.3

Se podría haber usado librerías como Axios y el uso de promesas para hacer el código más rápido y mostrar datos antes al usuario, es decir, cargar la primera página en la tabla mientras se cargan el resto más tarde ya que el usuario verá primero la primera página y más adelante el resto, ahorrando tiempo para el primer dibujado de los datos.

También se podían haber usado librerías como Bootstrap , Bulma, Material Design, etc. que ya traen componentes prefabricados para dar formato a estas tablas.