Práctica Web-Components:

El objetivo de la práctica desarrollar un prototipo de página de inicio de un sitio web basado en micro frontend. En concreto se desarrollará un Web Component que exponga al usuario información sobre bienes patrimoniales de la ciudad de La Laguna. El componente mostrará información relativa a los antecedentes, el tipo de edificio, una imagen y la ubicación del edificio. También se agregará un segundo componente que presente información sobre valoraciones de los usuarios en visitas al mismo.

Preparación del entorno de pruebas:

En la fase de desarrollo la información respecto al bien la puedes obtener a través de un servicio de simulación de API REST que facilite la creación de un entorno de pruebas en el desarrollo del prototipo. Existen varias opciones para esto: MockAPI, Beeceptor, Mockable, etc. Un ejemplo con Mockable: https://www.mockable.io/a/#/space/demo7708722

Se ha elaborado con un número reducido de bienes (3), del tipo Arquitectura Civil y Doméstica anterior al s. XX. cuya información se ha obtenido en el Gestor del Patrimonio Cultural del CICOP.

El conjunto de datos sobre los bienes usados en el ejemplo está disponible en el Campus Virtual de la asignatura. Un ejemplo de bien:


```
{
 "nombre": "CASA ALVARADO BRACAMONTE",
 "antecedentes": "El Capitán General de Tenerife don Diego de Alvarado Bracamonte
edificó esta casa entre 1624 y 1631",
 "tipo": {
     "arquitectura": "CIVIL Y DOMÉSICA",
     "época":"Anterior al s. XX"
 },
 "img":
"http://gestorpatrimoniocultural.cicop.com/imgFicha/17_1_2021_0_49_40.png",
 "localizacion":{
     "lat": "28°29'21.2 N",
     "long": "16°18'51.5 W"
     }
}
```


Esta información la debes recuperar mediante llamadas a la API para inyectar la información al componente. En la página de inicio se distribuirán varias cartas, por ejemplo, podrían ser los 3 componentes con mejores valoraciones, o con cualquier otro criterio que quieras considerar.

Estructura del componente

La estructura del componente se debe especificar mediante una plantilla que permitirá generar el árbol DOM del nodo que genera el componente. Para ello se debe utilizar la etiqueta `<template>`. Se recomienda añadirle un identificador para poder manejarla en el código del componente. La plantilla puede tener sus estilos definidos. Esta estructura estará disponible en el HTML para que el Web Component pueda clonarlo y generar fragmentos del DOM donde se necesite.
En una primera versión del prototipo se creará sólo la información relacionada con el bien, obtenida a través de la API anterior, para lo cual se pide que en la tarjeta se muestre:

```
<template id="card-bien">
    <style>
      p {
        text-align: center;
        font-weight: normal;
        }
    </style>
    <p></p>
    <p></p>
    <p></p>
   </template>
```

En la plantilla, si fuese necesario puedes insertar `<slot>` para dejar la estructura preparada para insertar contenido marcado adicional.

Creación del Web Component

El componente se define en javascript a partir de la clase `HTMLElement`. Podemos generar los nodos en el constructor inicialmente, o en alguna función callback. Se crean elementos en memoria a partir de la plantilla, y se añade a la página en el punto que se elija.

```
var template = document.getElementById('bienTemplate').content;
    var clone = template.cloneNode(true);
    document.body.appendChild(clone);
```

Encapsulamiento:

Se puede utilizar el Shadow DOM para encapsular el elemento y evitar conflictos con el resto del DOM. Podemos elegir el modo: closed u open, según queramos que sólo sea accesible por él mismo, o que se pueda acceder mediante JavaScript usando `Element.shadowRoot`

```
shadow = this.attachShadow({ mode: 'closed' })
template = document.getElementById('bienTemplate').content.cloneNode(true)
shadow.append( template )
```

Atributos:

Podemos usar atributos, que actuarán como parámetros de entrada en el elemento. Puedes utilizar el constructor:

```
class BienPatrimonial extends HTMLElement {

    constructor() {
      super();
      this.name = 'Nombre del bien';
    }
}
```

Se puede declarar un array de atributos, para ello debes usar **`static observedAttributes().`** Estos atributos son propiedades observadas, existe una función de callback para responder ante los cambios: **`attributeChandgedCallback()`**

```
class BienPatrimonial extends HTMLElement {

    static get observedAttributes() {
        return ['name'];
}
```

El web component pasa por diferentes estados en su ciclo de vida, cada no de ellos tiene asociada una función de respuesta. Se pueden usar eventos de la clase HTMLElement, o bien crear eventos para la etiqueta. Un evento se puede manejar en objetos del Shadow y se pude activar, enviando mensajes a todo el que esté a la escucha del evento. También se pueden definir parámetros.

```
const event = new Event("build");

// Listen for the event.
elem.addEventListener(
  "build",
  (e) => {
    /* ... */
```

```
  },
  false,
);

// Dispatch the event.
elem.dispatchEvent(event);
```

https://developer.mozilla.org/en-US/docs/Web/API/Web_components