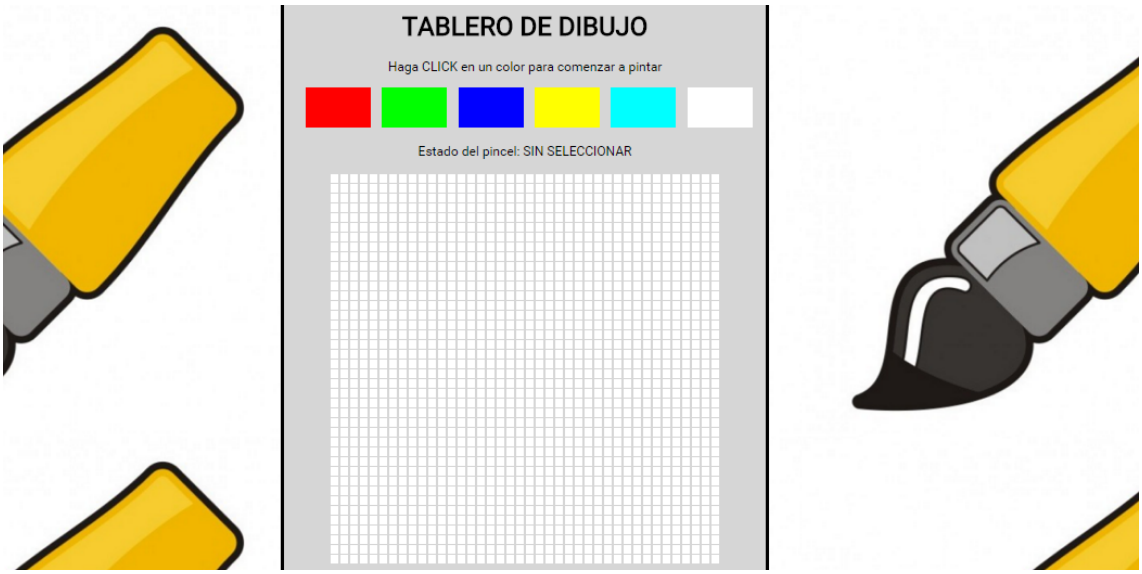
TAREA UT7

PROGRAMACIÓN AJAX EN JAVASCRIPT

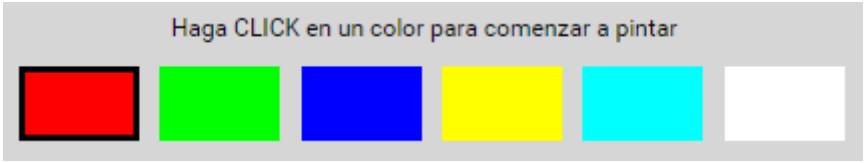
La tarea consiste en la implementación de varios scripts en lenguaje Javascript.

1. [2 puntos] El ejercicio consiste en la realización de un tablero de dibujo compuesto por una paleta de colores y una cuadrícula de 40 x 40 puntos. Ambas se generarán dinámicamente al iniciar la aplicación.
   Se implementarán dos versiones:
   - **Una utilizando las funciones estándar para manejo del DOM y eventos.**
   - **Otra utilizando las funciones que proporciona jQuery para tratamiento del DOM y manejo de eventos.**

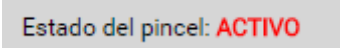


La forma de funcionamiento de la aplicación es la siguiente:

- Al pulsar en un botón de la paleta, se mostrará un recuadro alrededor del mismo.

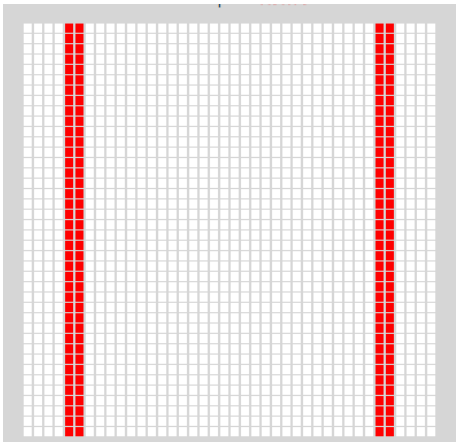


- Una vez seleccionado un color, si pulsamos en una celda, se pintará del color activo en la paleta, en cuyo momento activaremos el panel indicativo especificando que el pincel está ACTIVO:



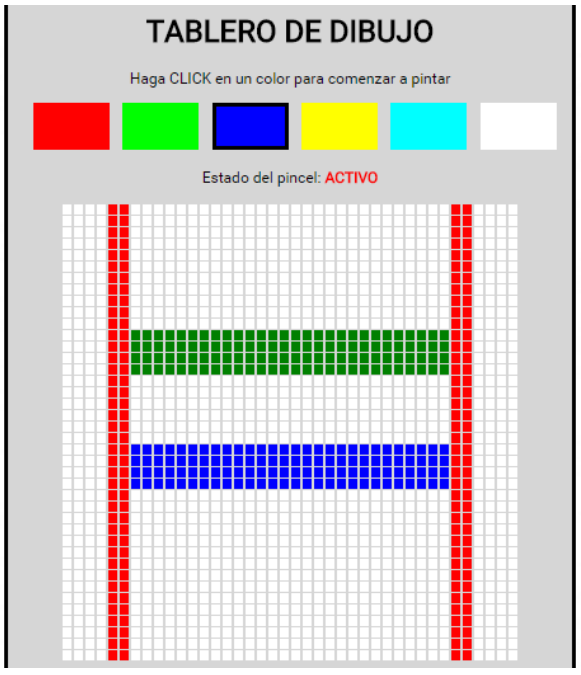
  Además, el cursor sobre las celdas cambiará su forma para indicar que estamos dibujando.

- A partir de ese momento, al mover el ratón por el tablero pintaremos del color activo todas las celdas por las que vayamos pasando el ratón.



- En el momento en que pulsemos en otra celda, dejaremos de pintar, pasando a estar el pincel en estado INACTIVO:

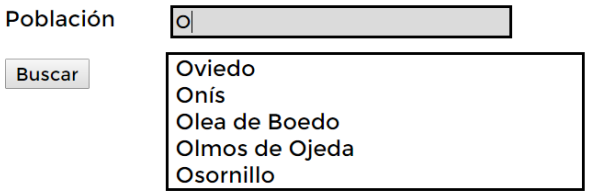
- Podemos escoger un color diferente y repetir el proceso, incluso para celdas ya pintadas. Para borrar el contenido de una celda escogeremos el color blanco:



Se proporciona la página HTML y la hoja de estilos que hay que usar obligatoriamente. Ninguna puede ser modificada, excepto en lo relativo a las llamadas a scripts Javascript. Esto implica que **todos los elementos** de la página deben ser generados dinámicamente desde Javascript.

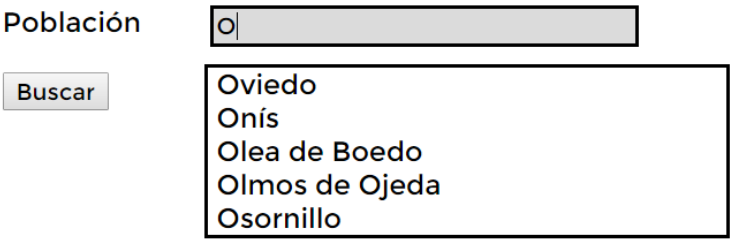
Para gestionar la lógica del programa, crea una clase `Tablero` (por el método que prefieras), que contendrá las variables necesarias para configurar el programa: Número de colores, filas y columnas del tablero, color actual, o la información que consideres oportuna.

2. [4 puntos] Se realizará una Web que nos permite consultar información acerca de pueblos de España mediante un sencillo formulario. Se proporciona:
   - Los scripts PHP que vas a necesitar
   - El código SQL que necesitas para importar el usuario de la base de datos (ddwec07_ejercicio2/ddwec07_ejercicio2), la base de datos (ddwec07_ejercicio2) y las tablas (población y provincias).
   - Una sugerencia de hoja de estilos CSS (**que puedes modificar a tu gusto**).



En la capa con id `ubicacion-actual` se mostrará sobre un mapa de Google Maps[1] **la ubicación actual del usuario**, obtenida a partir de la API de Geolocalización.

Bajo el mapa se mostrará un formulario con una caja de texto. En el momento en que el usuario escribe un carácter (evento de teclado sobre la caja de texto) la aplicación nos va a sugerir 5 nombres de poblaciones que aparecerán en una lista. Estos nombres se pedirán mediante una petición AJAX al script **`sugerencia_poblaciones.php`.**



[La captura no se corresponde con el listado real de sugerencias a partir de la BD proporcionada]

[1] Para mostrar las coordenadas en el mapa habrá que hacer uso de la API de Google Maps, cuyo uso no se explica en el tema pero se puede consultar en el siguiente enlace.

Una vez que ha aparecido la lista de sugerencias, tenemos la opción de seguir introduciendo caracteres, en cuyo caso se actualiza el listado, o escoger una población del listado pulsado sobre ella. Una vez seleccionada, al pulsar sobre el botón *Buscar* podemos obtener información sobre la población en un panel que se mostrará tras el formulario (inicialmente oculto):

- Provincia.
- Código postal
- Ubicación de la población en Google Maps.

La información se obtendrá mediante AJAX a partir del script **`info_poblacion.php`**

A la hora de implementar la aplicación habrá que tener en cuenta lo siguiente:

- Las operaciones de manipulación del DOM/manejo de eventos se implementarán usando jQuery.
- Para obtener la información de **`sugerencia_poblaciones.php`** se usará una petición AJAX asíncrona a partir de un objeto XMLHttpRequest (si bien a modo de ejercicio se recomienda al alumno que intente implementar el ejercicio usando una petición AJAX con funciones jQuery). Ten en cuenta que el script acepta un parámetro enviado mediante GET y devuelve las sugerencias como un array de objetos representados en JSON.
- Para obtener la información de **`info_poblacion.php`** se usará una petición AJAX apoyándonos en las funciones de jQuery (si bien a modo de ejercicio se recomienda al alumno que intente implementar el ejercicio usando una petición AJAX con funciones nativas). Ten en cuenta que el script acepta parámetros enviados mediante POST y devuelve la información en formato XML.

Script PHP	Recibe	Devuelve
`sugerencia_poblaciones.php`	[GET] Parámetro `sugerencia`. Contiene la cadena a partir de la cual deseo buscar sugerencias.	Array de objetos JSON que representan las poblaciones que comienzan con sugerencia.
`info_poblacion.php`	[POST] Parámetro `poblacion`. Contiene la población para la cual queremos información adicional.	Documento XML que contiene provincia, código postal y coordenadas.

3. [4 puntos] El Ayuntamiento de Santander necesita realizar una aplicación que **monitorice el estado de los parkings** mostrando el porcentaje de ocupación de plazas libres. Para ello se necesita un sistema de representación gráfica del estado de los parkings basado en iconos:

Porcentaje de plazas libres	Icono
>= 75 %	Verde
>= 50% y < 75 %	Azul
>= 25% y < 50%	Naranja
< 25%	Rojo

Se requiere que sobre el mapa con la imagen de la ciudad se superpongan los iconos relativos al estado de ocupación de los parkings sobre los que hay información. Para obtener información desde un servidor de forma centralizada, se ha implementado el script **`estado_parkings.php`**:

- Si no recibe nada, devuelve un listado en formato JSON con el listado de los parkings. Para cada parking se proporciona un identificador, las coordenadas x e y del parking sobre el mapa, el nombre del parking y su porcentaje de ocupación.

  Ejemplo de respuesta JSON:

```
[ {id:0, coordenadas: [120, 280], porcentaje: 85, nombre: "Valdecilla Norte" },
  {id:1, coordenadas: [150, 360], porcentaje: 45, nombre: "Valdecilla Sur" },
  {id:2, coordenadas: [250, 220], porcentaje: 72, nombre: "Mercado de Méjico" },
  {id:3, coordenadas: [380, 180], porcentaje: 53, nombre: "Numancia" },
  {id:4, coordenadas: [470, 160], porcentaje: 85, nombre: "Jesús de Monasterio" },
  {id:5, coordenadas: [625, 200], porcentaje: 64, nombre: "Ferry" },
  {id:6, coordenadas: [875, 35], porcentaje: 51, nombre: "Tetuán" }]
```

- Si recibe por GET un parámetro denominado id con el identificador del parking sobre el cual queremos conocer el estado, y devuelve en formato JSON la información relativa a dicho parking.

  Ejemplo de respuesta JSON (enviamos por GET el parámetro id con valor 2):

```
{ id:2, coordenadas: [250, 220], porcentaje: 63, nombre: "Mercado de Méjico" }
```

Teniendo en cuenta lo anterior, habrá que implementar los siguientes puntos:

Se usará usando **objetos literales** una clase **`Parking`** que tiene las siguientes **propiedades públicas**

Propiedad	Descripción
`id: entero`	Identificador del parking
`porcentajeOcupacion: entero`	Porcentaje de ocupación, del 1 al 100.
`coordenadas:Array(2)`	Array de dos elementos que representa las coordenadas del parking, siendo la esquina superior izquierda el origen de coordenadas.
`nombre: cadena`	Nombre del parking.

- Implementar usando prototipos la clase **`ListadoParkings`** que encapsula el acceso a un conjunto de Parkings.

Propiedad	Descripción
`parkings: Array`	Listado de objetos **`Parking`**

Método	Descripción
`ListadoParkings()`	Constructor sin parámetro.
`addParking(parking)`	Añade un nuevo objeto de tipo Parking.
`getNumeroParkings():entero`	Devuelve el número de parkings.
`getParkingFromId(id):Parking`	Devuelve el parking que tiene la id que se pasa como parámetro
`actualizaPorcentaje(id,porcentaje)`	Actualiza el porcentaje del parking que tiene la id que se pasa como parámetro.

Se valorará positivamente la realización de alguna prueba unitaria con la clase.

Además, se podrán implementar aquellos métodos que se vea necesario para realizar el ejercicio, teniendo en cuenta que no se podrá acceder directamente a la propiedad privada **parkings** (habrá que hacerlo a través de métodos).

Se instanciará un objeto de esta clase al cargar la página HTML, [llamaremos a dicha instancia **`listado`**]. Esta variable será la única variable global que está permitida en el script asociado a la página HTML.

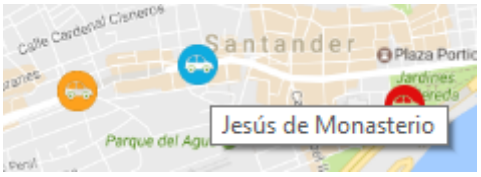
- Al pulsar el botón *Actualizar*, se mostrará información sobre los parking. Dicho proceso se realizará en tres pasos:
  a. En primer lugar se obtendrá la información de los parkings a través de una petición AJAX al fichero **`estado_parkings.php`**. La petición se hará obligatoriamente usando funciones de jQuery para AJAX.

  b. En segundo lugar se pasará la información obtenida mediante la petición AJAX a **`listado`**.
     i. Si es la primera vez que mostramos el mapa, habrá que crear nuevos objetos de tipo parking, mientras que si ya lo hemos mostrado anteriormente, solo habrá que actualizar los porcentajes de ocupación

  c. En tercer lugar, se recorrerá el objeto listado. Para cada elemento del mismo dibujaremos una representación del parking sobre el mapa que se proporciona.
     i. Si es la primera vez que mostramos el mapa, habrá que crear los puntos del mapa, mientras que si ya lo hemos mostrado anteriormente, solo habrá que actualizar los iconos en función del porcentaje de ocupación de cada punto.

Para mostrar cada uno de los parkings se usará una capa de clase **`info-parking`** donde se establecen los siguientes elementos:

- Se asignará la clase rojo, verde, naranja o azul en función del porcentaje de ocupación del parking.
- Se establecerán las propiedades de estilo **`top`** y **`left`** en función de la posición del parking.
- Se establecerá la propiedad **`title`** del parking para que al pasar el ratón sobre el mismo se muestre un mensaje con su nombre:



- Si pulsamos sobre el icono de un parking, se actualizará la información relativa a dicho parking. Para ello:
  a. En primer lugar se obtendrá la información de los parkings a través de una petición AJAX al fichero **`estado_parkings.php`**, pasándole únicamente el id del parking que queremos actualizar como parámetro. La petición se hará obligatoriamente usando funciones AJAX nativas.
  b. Se actualizará la información relativa al parking (porcentaje de ocupación) en el objeto **`listado`**.
  c. Se actualizará el icono del parking en función del porcentaje de ocupación del parking.

- Cada vez que pulsemos el botón *Guardar en local,* se guardará la información de todos los parkings en localStorage.

  - Si pulsamos el botón *Recuperar desde local¸* se leerá la información desde localStorage y se utilizará para actualizar los porcentajes de ocupación de todos los parkings

  - El botón *Guardar en local* se habilitará la primera vez que pulsemos *Actualizar.*
  - El botón *Recuperar desde local* se habilitará la primera vez que pulsemos *Guardar en local.*

En este ejercicio se utilizan funciones nativas para el manejo del DOM y eventos.