Objetivos generales

Con esta práctica comenzamos la transición del primer bloque del bootcamp al segundo, utilizando lo aprendido en los módulos iniciales (HTML, CSS, JavaScript y PHP) para dar forma a una aplicación web plenamente funcional.

Que nadie se asuste, sabemos que algunos lleváis poco más de tres semanas en el mundo del desarrollo de software, no vamos a pedir que se os ocurran genialidades para optimizar el funcionamiento de la aplicación, ni siquiera vamos a pedir que hagáis nada que no tengáis ya en los ejemplos de clase: digamos que será como resolver uno de esos crucigramas en los que solo hay que unir los puntos, pero, eso sí, veréis que aunque sea para copiar y pegar tendremos que entender mínimamente lo que estamos escribiendo o quizá nuestra aplicación no funcione.

Con esta práctica queremos que...

- ganéis familiaridad con la sintaxis de los distintos lenguajes;
- veáis cómo se conectan entre sí;
- os acostumbréis a escribir un código limpio y ordenado, tratando de usar funciones para los grupos de sentencias que se repitan;
- ganéis autonomía a la hora de investigar posibles soluciones.

Ojo, que estáis aquí para aprender, así que preferimos ver una sola funcionalidad que le podrías enseñar a un desconocido sin pasar apuro y en la que entendéis perfectamente lo que habéis implementado, que diez funcionalidades plagadas de errores y que no sabríais explicar cómo hacen lo que hacen.

Lo más importante de esta práctica es perderle el miedo a equivocarse, a lo que no sé todavía, lanzarse a la piscina sin tener claro si sé nadar, porque no se puede aprender a nadar desde el vestuario, hay que mojarse.

Requisitos técnicos

Puesto que en esta práctica vamos a usar todas las tecnologías que hemos visto hasta ahora, hay unos mínimos que esperamos que se incluyan de lo que habéis visto en cada uno de los módulos. En la siguiente lista marcamos algunos requisitos como EXTRA, indicando qué objetivos podríamos marcarnos después de haber conseguido cumplir los requisitos mínimos.

HTML

- La estructura de las páginas creadas dinámicamente tendrá que tener sentido a nivel semántico
- EXTRA: utilizaremos un formulario correctamente validado para añadir datos

CSS

- Usaremos Flex para posicionar los elementos (alternativamente podremos usar otro esquema de posicionamiento como Grid o la Grid de Bootstrap)
- EXTRA: se buscará un sentido y orden a las clases definidas en el CSS (usando por ejemplo el sistema de nombrado BEM u otro similar)
- EXTRA: la web será responsive, contemplando el tamaño móvil, tablet y desktop

JavaScript

- Tenemos que implementar nuestra web como una SPA (single page application), lo que quiere decir que...
    - Los enlaces no deberán cargar un nuevo documento, dispararán la modificación del documento actual por medio de JS
    - Los formularios no deberán enviarse, los datos se recogerán desde JS y, si fuese necesario, se modificará el documento actual por medio de JS
- Los datos se obtendrán de una API REST por medio de la API Fetch
- Aparte de los elementos <script> necesarios para cargar nuestra aplicación, en el <body> de nuestro fichero HTML solo tendremos un elemento en el que inyectaremos los demás elementos que iremos creando dinámicamente mediante JS usando...
    - Métodos del objeto document como createElement y otros de propios de los nodos como appendChild
    - Manejadores de evento registrados con addEventListener
- La web tendrá elementos comunes a todas sus páginas, por ejemplo, una barra de navegación y/o un footer con información
- Se espera la web tenga al menos tres vistas...
    - Una home con un menú general (por ejemplo, mostrando distintas categorías)
    - Un listado resultado de pulsar en una de las opciones de la home (por ejemplo, al pulsar en una categoría se muestran en forma de lista todos los elementos que coinciden con esa clasificación)
    - El detalle de un elemento (por ejemplo, al pinchar en un elemento de la anterior lista nos llevaría a una página con todos los detalles de este)
- EXTRA: configurar el proyecto para que utilice ESLint para corregir el código y formatearlo al estilo de Google

- EXTRA: utilizaremos funciones para no repetir una y otra vez las mismas sentencias, funciones que tendrán resposabilidades lo más concretas posible
- EXTRA: utilizaremos ficheros distintos que cargaremos desde el HTML con la etiqueta <script>, agrupando las funciones por resposabilidad
- EXTRA: los listados se cargarán con lazy loading, o sea, la primera vez cargarán algunos resultados y hasta que no hagamos scroll no cargarán los siguientes
- EXTRA: en la barra de navegación habrá un buscador que nos permite obtener elementos que coincidan con la cadena de búsqueda introducida
- EXTRA: el buscador sugerirá al empezar a escribir algunas cadenas de búsqueda en las que el usuario podría estar interesado: al pulsar en una sugerencia se lanzará la búsqueda en cuestión
- EXTRA: la web almacenará localmente ciertas preferencias del usuario (podría ser que por ejemplo hubiese una página de configuración en la elegir ciertas preferencias, o que se almacenen elementos de los listados en las búsquedas en una lista de favoritos, o que se puedan añadir notas ligadas a los elementos, o comentarios de los lectores, o cualquier otra idea que trabaje con la persistencia local del navegador)
- EXTRA: en los listados podría haber un filtro que modificase la vista según los elementos coincidan o no con el texto de filtrado
- EXTRA: replicar el sistema de pruebas Jest que hemos usado para corregir automáticamente baterías de ejercicios y crear pruebas específicas para nuestras funciones, de forma que si al hacer un cambio algo deja de funcionar, nos enteremos rápidamente

PHP

- EXTRA: obtener algunos o todos los datos que vamos a mostrar en nuestra web de un fichero PHP

Por dónde empezar

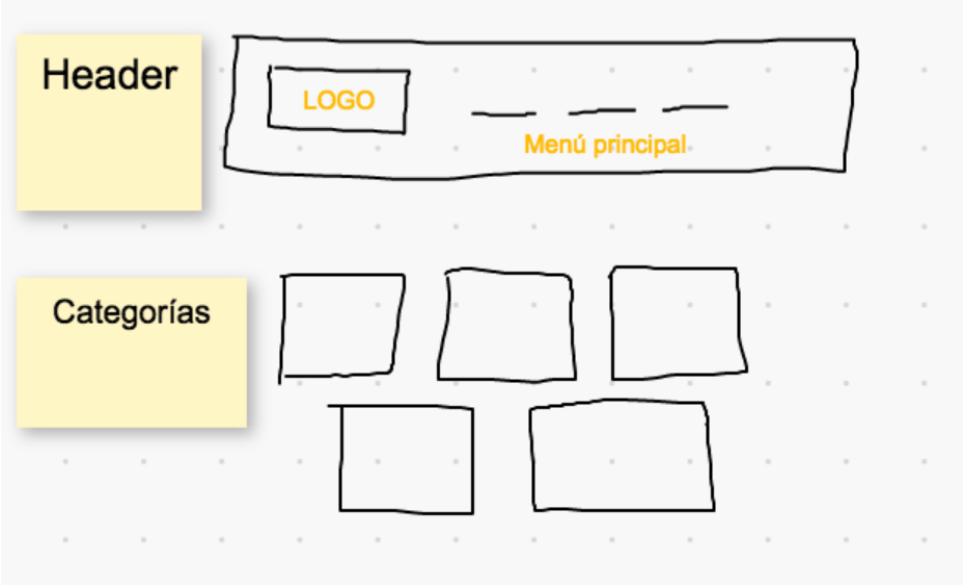
Lo primero que debemos hacer es encontrar una fuente de datos. Tenemos que buscar una API REST que nos permita hacer consultas de forma gratuita y que nos ponga los menores problemas a la hora de registrarnos. Hay infinidad de APIs pensadas para hacer pruebas o creadas por aficionados a las películas, series, libros, deportes, etc.

Nos registraremos y averiguaremos de qué forma tenemos que hacer las peticiones HTTP para que el sistema nos identifique correctamente. Algunas de estas APIs serán totalmente abiertas y no tendremos que hacer nada de especial, pero habrá otras en las que tengamos que registrar la aplicación que queremos que se conecte al sistema y obtener un identificador que habrá que

incluir en cada consulta (es lo que se suele llamar API key o, según la tecnología, token).

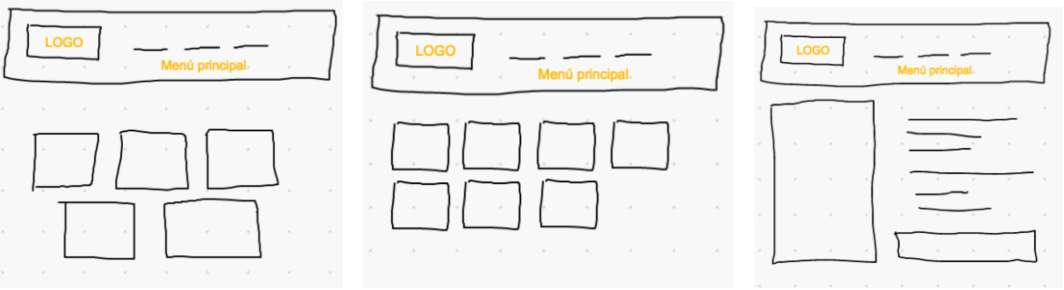
Una vez que hemos revisado la documentación de la API nos habremos hecho una idea de qué datos nos puede proporcionar. Por ejemplo, podríamos usar una base de datos de películas que nos proporciona no solo detalles sobre estas (título, dirección, año, etc.) si no también una imagen con el póster promocional. Supongamos que esta API nos permite clasificar además las películas por género. Visto lo visto, estamos preparados para plantear el diseño de la web.

Para plantear el diseño se puede empezar con unos simples garabatos sobre papel, lo único que necesitamos es hacernos una idea de la estructura, de las piezas o componentes que forman cada vista y de la navegación entre estas.



Dibujamos el layout, o sea, la parte común a todas las vistas: en el anterior ejemplo sería la barra de navegación con el logo y el menú principal.

Luego podríamos hacer un pequeño esquema de la estructura de cada una de las vistas y la relación entre estas.



Una vez que tenemos claros los elementos que vamos a tener que maquetar llega el momento de escribir el HTML y el CSS como si se tratase de una página estática (nos servirá como plantilla para luego reproducirlo desde JS). En este momento nos pueden venir muy bien librerías como Bootstrap o Material para

acelerar el proceso de maquetación, e incluso podríamos usar sus directamente sus componentes, que solo tendríamos que adaptar a la estructura de los datos que nos llega de la API.

El siguiente paso será codificar una serie de funciones en JavaScript para poder crear dinámicamente el HTML y CSS que hemos creado en el paso anterior de forma estática.

Después tendremos que crear funciones distintas para cada llamada a la API (por ejemplo, obtener categoría, listar elementos, obtener detalles de un elemento, etc.).

Y el paso final sería hilar toda la funcionalidad que hemos codificado. Las funciones de DOM serán alimentadas con los datos de la API y se añadirán a la vista siguiendo las pautas de la navegación que hayamos decidido.

Entrega de la práctica y presentación

Podemos hacer cambios hasta el día de la presentación. Al final de la presentación, como tarde, tendremos que hacer un merge request a la rama master del proyecto poniendo al mentor como asignado (asiggnee).

La presentación consistirá en explicar frente a las demás personas del grupo y uno o más mentores los retos a los que te has enfrentado, cómo los has resuelto, lo que te habías propuesto y hasta dónde has llegado. Mientras haces la demo de tu web los mentores y otras personas del grupo te harán preguntas para entender bien lo que has hecho, por qué y cómo.

Tras la presentación el mentor repasará el código de las prácticas y pondrá los comentarios pertinentes en el hilo de comentarios del merge request, pidiendo en ocasiones que se hagan ciertos cambios. Una vez hechas las correciones, tras una última revisión, el mentor dará por cerrado el merge request y con eso habrá terminado tu trabajo.