

Reto 2 - Agencia de Publicidad Tom Nook



Alaitz Candela
Daniel Tamargo
Raúl Melgosa

ÍNDICE

- 1.- RWD
- 2.- CSS Minify
- 3.- Documentación código JavaScript y PHP
- 4.- Metodología GIT
- 5.- Librerías
- 6.- Bibliografía

Responsive web design (RWD)

El diseño de nuestras páginas (layout, maquetación, enfoque) está basado en la filosofía RWD (responsive web design).

Esto significa que este proyecto está enfocado de tal forma que dé igual desde qué dispositivo se acceda a la web, dado que la página conservará las mismas funcionalidades en todas las resoluciones.

Para lograr este objetivo hemos hecho uso de un diseño web fluido basado en porcentajes y en las tecnologías **css grid** y **css flexbox** además de utilizar tamaños de fuente elásticos, medidos en **em**.

Debido a todo esto la disposición de la página varía dependiendo del ancho de la pantalla en la que se esté mostrando, pensando durante el desarrollo en las pantallas más pequeñas, las de los dispositivos móviles que hoy en día son de los más utilizados.

CSS minify

Uno de los principales problemas al desarrollar páginas web es el tamaño que terminan ocupando.

Para ganar espacio, hemos reducido el tamaño de nuestro fichero CSS quitándole caracteres inútiles como espacios, tabulaciones y saltos de línea que al fin y al cabo, también ocupan espacio.

[Para ello hemos utilizado la página CSS Minifier.](#)

Nota: en el repositorio del proyecto mantenemos el fichero css antes de minimizarlo para que su contenido sea legible.

Documentación código JavaScript y PHP

Para seguir una de las buenas prácticas de la programación, se han documentado los métodos (o la mayoría) donde se ha proporcionado una pequeña descripción del objetivo, parámetros de entrada y valor de respuesta (si fueran necesarios).

Ejemplos JavaScript:

```
// Recibe otra tanda de usuarios a cargar (simulando lazy load, pero cada vez que clicas)
function mostrarMasUsuarios(){
    |   cogerUsuarios();
}

// Carga la siguiente página de usuarios
function siguientePagUsuarios() {
    |   paginaActual++;
    |   numVecesCargado = 0;
    |   vaciarYCogerUsuarios();
    |   $("#cargar-mas-usuarios").prop('disabled', false);
    |   $("#anterior-pag-usuarios").prop('disabled', false);
    |   $("#siguiente-pag-usuarios").addClass('display-none');
}

// Carga la página anterior de usuarios
function anteriorPagUsuarios() {
    |   paginaActual--;
    |   numVecesCargado = 0;
    |   vaciarYCogerUsuarios();
    |   if (paginaActual <= 1) {
    |       |   $("#anterior-pag-usuarios").prop('disabled', true);
    |       |   $("#anterior-pag-usuarios").addClass('display-none');
    |   }
    |   $("#siguiente-pag-usuarios").prop('disabled', false);
    |   $("#cargar-mas-usuarios").prop('disabled', false);
}

function vaciarYCogerUsuarios() {
    |   // Limpiamos el contenido actual de la lista para volcar lo siguiente
    |   $("#usuarios-registrados").html('');
    |
    |   // Volcamos lo siguiente
    |   cogerUsuarios();
}
```

```
function clickFavoritos(id) {
    // Obtenemos la posición del id en la cookie, si no existe, devuelve -1
    let posicion = anuncios_favoritos.indexOf(anuncios_favoritos.find(elm => elm == id));
    if (posicion < 0) { // Si no existía, lo está añadiendo como favoritos
        anuncios_favoritos.push(id);
    } else { // Si existe, lo está eliminando de favoritos
        anuncios_favoritos.splice(posicion, 1);
    }

    if (anuncios_favoritos.length <= 0) { // Si ha eliminado el último de favoritos, eliminamos la cookie en si
        document.cookie = "favoritos=;max-age=0";
    } else { // Si existen valores en favoritos, simplemente guardamos o reemplazamos la cookie favoritos
        document.cookie = "favoritos=" + anuncios_favoritos;
    }
}
```

Ejemplos PHP:

```
// Vincular un anuncio a una categoría
function insertarCategoriaAnuncios($dbh,$data){
    $stmt = $dbh -> prepare("INSERT INTO categoriasAnuncios VALUES (:id_anuncio,:id_categoria)");
    return $stmt->execute($data);
}
```

```
// Comprueba si el usuario loggeado existe y sigue habilitado
function usuarioLoggeadoYHabilitado() {
    $usuarioHabilitado = false;
    $habilitado = -1;

    if (isset($_COOKIE["id_usuario"])) {
        $dbh=connect();
        $habilitado = obtenerHabilitadoUsuario($dbh,$_COOKIE["id_usuario"]);
        if ($habilitado == 1) {
            $usuarioHabilitado = true;
        }
        $dbh=close($dbh);
    }

    // Si no hay usuario loggeado o el usuario ha sido deshabilitado, redirigimos al login
    if (!$usuarioHabilitado) {
        // Forma limpia de borrar una cookie (server y browser)
        // https://stackoverflow.com/questions/686155/remove-a-cookie#:~:text=A%20clean%20way%20to%20delete%20a%20cookie
        limpiarCookiesUsuario();
        if ($habilitado == 0) setcookie("usuario-deshabilitado", "true", time() + 3600 * 24 * 7 * 4 * 12);
        header ("location: ../login.php");
        die();
    }

    return $habilitado;
}
```

Al comentar los métodos no solo nos ayudamos a nosotros mismos de cara al futuro, si no también a otros desarrolladores o compañeros que puedan acabar modificando y/o necesitando nuestro código.

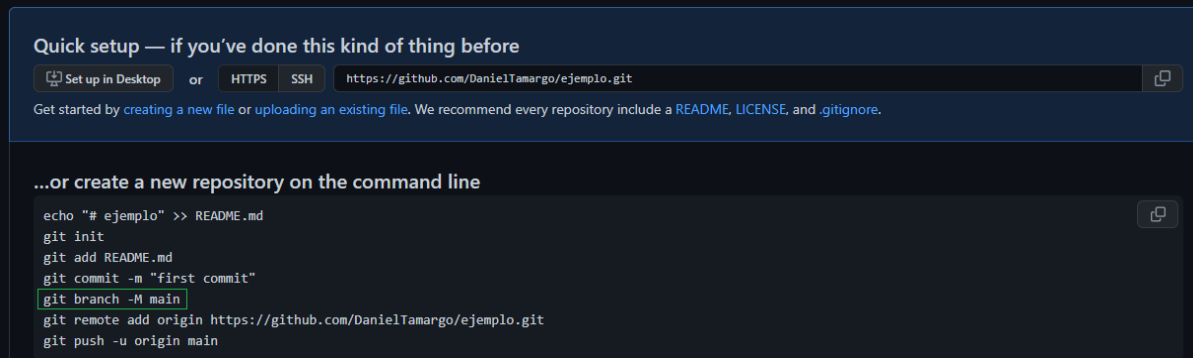
Metodología GIT

Hemos basado nuestro flujo de trabajo conjunto con git en la metodología Main-Develop.

¿Por qué Main y no Master?

El concepto de Master surge del sistema de nombres master/slaves (maestro/exclavos), en 2020 estos términos llamaron la atención de gran parte del mundo y empresas como GitHub, empatizando con el motivo, comenzaron a sugerir el renombrar la rama master a main nada más crear el repositorio.

Esto se puede observar en los propios comandos sugeridos por GitHub al crear un nuevo repositorio:



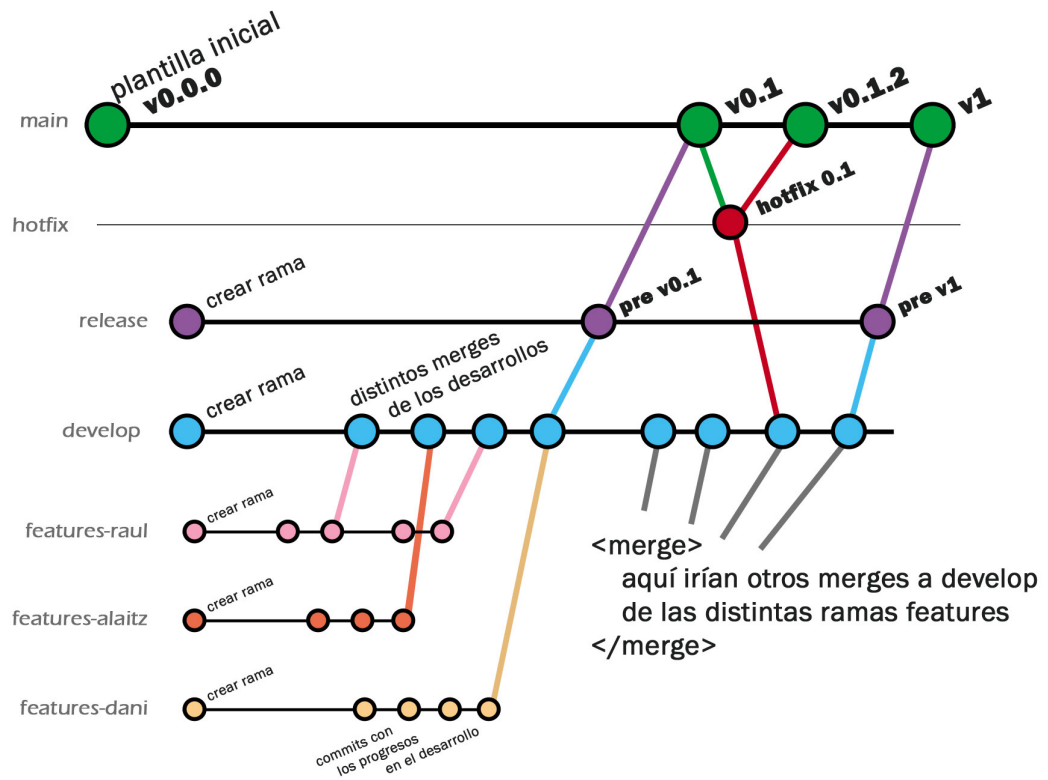
```
Quick setup — if you've done this kind of thing before
[icon] Set up in Desktop or HTTPS SSH https://github.com/DanielTamargo/ejemplo.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line
echo "# ejemplo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/DanielTamargo/ejemplo.git
git push -u origin main
```

[Enlace a post de Genbeta que habla al respecto.](#)

Diagrama de nuestra metodología Git:

Metodología git: Main-Develop



Librerías

Para este proyecto hemos utilizado 3 librerías de JavaScript; JQuery, SweetAlert 2 y Chart.js.

JQuery

Es una librería muy básica que nos permite hacer lo mismo que con JavaScript plano, pero escribiendo menos cantidad de código. Además simplifica la comunicación con el servidor mediante métodos como Ajax, o los Web Services.

Sweet Alert 2

Esta librería permite embellecer los mensajes que se muestran por pantalla de forma muy simple que de otra forma llevaría mucho más tiempo realizar.

Chart.js

Esta librería permite integrar gráficas de forma bastante simple con un aspecto mucho mejor de lo que lograríamos conseguir sin utilizarla.

Ionicons

Es una librería que permite importar iconos en formato svg para que se escalen sin perder calidad

Bibliografía

Git:

- [Por qué renombrar de master a main](#)
- [Documentación Git](#)

Documentación JavaScript

- [Funciones arrays](#)
- [Funciones sets/maps](#)
- [jQuery](#)
- [jQuery – Ajax](#)
- [Chart.js \(gráficas\)](#)

Documentación PHP

- [Manual PHP](#)

Iconos:

- [Ionicons](#)

Salvación:

- [StackOverFlow](#) (infinitos posts)

Además, se han consultado los apuntes de JavaScript, PHP, CSS y Git proporcionados en las distintas asignaturas.