*6to desafío entregable*

*Configurar nuestro proyecto para que soporte SASS. Aplicar los conceptos de partials y vars a nuestro proyecto. Cargar todo en github.*

Componentes:

1. Estructura avanzada de la web
2. Estilo avanzado de la web
3. Implementación de módulos de Node
4. Repositorio en Github

*Estructura avanzada de la web*

* **Formato:** Archivos HTML
* **Objetivo del desafío:** 
  + El estudiante deberá realizar una estructura del HTML prolija, limpia, fácil de leer y que no tenga errores en sus atributos o en sus valores.
  + El estudiante deberá agregar elementos HTML según su necesidad en armar contenedores o elementos web determinados, en base al framework elegido y la documentación del mismo.

**Incluir**:

* **Maquetado de la web:** Las estructuras maquetan a la web en base al framework elegido, haciendo usos de clases utilitarias para armar grillas, elementos web y estilos propios del framework, además del HTML de contenido. En caso de no elegir framework, los elementos deben respetar una cierta maqueta propia.
* **Páginas:** Todas las páginas tienen el contenido estructurado y el estilo linkeado. En caso de elegir un framework también tiene que tener agregadas las diferentes librerías de Javascript y CSS pertinentes al framework.

*Estilo avanzado de la web*

* **Formato:** Archivos CSS, Archivos SCSS o SASS
* **Objetivo del desafío:**
  + El estudiante deberá crear archivos de SCSS o SASS para darle estilo a su web.
  + El estudiante deberá utilizar trasladar los estilos creados en CSS a SCSS, haciendo uso correcto del nesting, las variables y los operadores de lenguaje Sass.
  + El estudiante formará un archivo de SCSS con una sintaxis correcta, dónde el código no tiene errores ya sea de CSS cómo de compilación.

**Incluir:**

* **Estilo avanzado:** Transforma lo que originalmente eran estilos de CSS en SCSS. Aprovechándose de las características de SCSS para armar estilos de CSS de forma dinámica, además del uso de nesting para estructurarlo de forma legible y evitando repetir código.
* **Estructura de la web:** Usa etiquetas no sólo para armar contenido, sino para armar los elementos que van a conformar el layout de la web, los contenedores, etc.

*Implementación de módulos de Node*

* **Formato:** Archivo package.json y package-lock.json
* **Objetivos del desafío:**
  + El estudiante deberá inicializar npm en el proyecto y configurar su package con los datos pertinentes al mismo.
  + El estudiante deberá instalar y agregar las dependencias nodemon y node-sass a su proyecto, además de los scripts necesarios para la compilación de archivos de Sass.
  + El estudiante deberá, en caso de utilizar otro módulo de npm, agregarlo como dependencia al proyecto.

**Incluir**:

* **Metadatos del proyecto:** Archivo package.json con información relevante del proyecto como el nombre y una mínima descripción.
* **Scripts de npm:** Para poder generar los archivos de CSS que va a necesitar luego el sitio web.
* **Dependencias del proyecto:** Además de nodemon y node-sass, que deben estar detalladas en el package.json como dependencias, también debe estar cualquier librería o módulo que el estudiante agregue, como puede ser el caso de un framework CSS.

*Repositorio en Github*

* **Formato:** Link al repositorio en Github donde está hosteado el proyecto
* **Objetivo del desafío:**
  + El estudiante deberá utilizar git de forma correcta para versionar su proyecto.
  + El estudiante hará uso de Github para brindar acceso al proyecto versionado.

**Incluir:**

* Se envían en el repositorio todos los archivos necesarios para visualizar correctamente la web.
* En el repositorio se muestran los commit que el estudiante usó para actualizar/versionar su código.

*RÚBRICAS DE EVALUACIÓN*

*6to desafío entregable*

| **Estructura avanzada de la web** | | | |
| --- | --- | --- | --- |
| **Criterios** | **Bajo**  Falta más profundización. Es confuso. | **Correcto**  Acorde pero con errores puntuales. | **Óptimo**  Es claro y pertinente. |
| Código prolijo | * El uso de tabulaciones y/o nuevas líneas marca de forma correcta la jerarquía de padre/hijo pero hay inconsistencias. * Se usan demasiados comentarios. * Comentan demasiado código para uso futuro o porque ya no lo va a usar. | * Uso de nuevas líneas y tabulaciones de manera prolija y consistente. * Usa los comentarios para documentar secciones de su HTML/CSS. | * Tabulaciones correctas y ordenadas, denotando jerarquía entre los elementos. |
| Tags HTML | * Errores mínimos a la hora de escribir tags. * Tags semánticos usados pero con problemas para comprender cual sirve para cada caso. * Falta de h1 en algunas páginas. * Más de un h1 en algunas páginas. * Las imágenes no tienen alt. | * El HTML no contiene errores en los atributos y tags. * Crea tags que envuelven a otros innecesariamente, ya sea porque no cumplen ninguna función o no se usan. * Todas las imágenes tienen alt. | * El nesting es óptimo, usando la menor cantidad de tags posibles. Uso de tags semánticos correcto y estructuración de la página desde el HTML. * El alt de las imágenes es pertinente y descriptivo. |
| Estilo en el HTML | * Usa nombres poco legibles para las clases. | * Clases correctas pero redundantes o irrelevantes. | * Nombres de clases consistentes, ya sea camelCase o kebab-case. |
| Funcionalidad | * La web tiene enlaces rotos. * Las fotos no se cargan por errores en la ruta. * No hay enlaces para navegar por las diferentes páginas. * El usuario queda atrapado al no tener como volver a la home por falta de enlaces. * Se usan rutas absolutas para archivos de la web, incluso haciendo uso del protocolo file:// | * La web tiene enlaces a todas las secciones en su navegación. * Las fotos están bien cargadas pero algunas son de relleno. * Hay enlaces que llevan a diferentes páginas y tiene como volver a la home. * No se usan rutas absolutas para los archivos de la web, sino relativas. | * Las páginas tienen enlaces funcionales. * Las imágenes tienen rutas relativas y correctas. * Las páginas están interconectadas correctamente. * Todas las imágenes son pertinentes al sitio y no hay de relleno. |
| Contenido | El contenido tiene errores ortográficos.  Hay contenido que no corresponde a la página donde está ubicado.  El contenido está desorganizado.  Hay poco contenido o está incompleto en algunas de las páginas.  La organización del contenido es equitativa pero mal distribuida a lo largo de la página. | El contenido no tiene faltas ortográficas o gramaticales.  Las páginas tienen una cantidad contenido apropiado y está bien distribuida.  El contenido es apropiado a la sección en la que está ubicado.  Imágenes demasiado pesadas o mucho más grandes del tamaño que corresponde. | La información está correctamente estructurada, usando los tags correctos para cada tipo de contenido, ya sean tablas, listas, titulares, párrafos o imágenes.  El contenido no está distribuido monótonamente y tiene varios niveles de lectura.  Las imágenes tienen un tamaño apropiado al que ocupan en el contenido. |

| **Estilo avanzado de la web** | | | |
| --- | --- | --- | --- |
| **Criterios** | **Bajo**  Falta más profundización. Es confuso. | **Correcto**  Acorde pero con errores puntuales. | **Óptimo**  Es claro y pertinente. |
| Código limpio y prolijo | * Tabulaciones erráticas y poco predecibles. | * Uso de tabulaciones consistente. * Declaraciones de reglas y de espacios entre los elementos estructuradas correctamente. * Métodos de tabular poco convencionales. | * Uso de tabulaciones y correcto. * Hace buen uso de los selectores para evitar repetir código. |
| Entendimiento del CSS | * Errores a la hora de hacer selectores. * Usa reglas de CSS no pertinentes al elemento seleccionado. * No recicla código y lo repite. * Los selectores son innecesariamente precisos, previniendo una óptima reutilización del código. | * Recicla código de forma poco óptima. * Usa demasiados elementos en su selector. * Usa reglas de CSS pertinentes al selector. | * Expande sobre elementos que ya había creado con clases que los modifican. * Genera estilos que son fáciles de cambiar o transformar para diferentes tamaños de dispositivo. |
| Utilización de Frameworks | * Trae la librería en su proyecto y no hace uso de las herramientas. * Arruina el box-modelling o el sistema de grillas del framework con estilos desde su hoja de estilo. * No usa las clases de responsive ya dadas por la librería. * Usa los estilos por defecto de la librería en toda la web, no hay identidad propia a lo largo de las páginas. * Utiliza un theme descargado y realiza cambios mínimos. | * Recrea las clases que ya trae su librería innecesariamente. * Utiliza el framework pero hace uso de pocos elementos. * Superpone los estilos de su librería para darle su propio estilo. | * Maqueta usando el framework. * Personaliza el framework haciendo uso de las variables del mismo y no de superponer las clases previamente definidas por el framework haciendo uso de la cascada. * Inserta sólo los módulos que desea de su framework. |
| Media queries & Responsive | * El sitio web no es usable en dispositivos más pequeños que desktop. * Contiene elementos completamente fuera de cuadro que no se adaptan a los cambios de tamaño. * Contiene imágenes o bloques que superan el ancho del padre. * El texto es ilegible, ya sea porque es demasiado grande o muy chico. * El elemento contenedor no se adapta a diferentes tamaños. * Usando la librería de bootstrap no usa img-fluid. | * Hace uso de las columnas de bootstrap o de media queries propios para lograr responsividad. * Hay demasiados queries porque no usa los breakpoints provistos por bootstrap o el framework que eligió. * No usa un framework y hay demasiados queries por no definir sus propios breakpoint. * Hace selectores muy específicos para los cambios de tamaño de los elementos, como pueden ser las tipografías | * Uso de unidades relativas. * El sitio web cuenta con una buena navegación en numerosos tamaños, en particular en mobile, laptop y desktop. * Usa los breakpoint de su framework para generar media query para los diferentes tamaños con los que trabaja. * Ante la falta de un framework, genera sus propios breakpoint y limita sus media query a esos breakpoint. |

| Entendimiento de SCSS/SASS | * No utiliza ninguna propiedad de SASS, solo se limita a poner el CSS en los archivos de SCSS. * Hay errores en los operadores de SASS. * Hay errores en las funciones propias de SASS. * Hay errores cuando crea variables con listas o mapas. * Errores de nesting o en el uso de ampersand. * Cuando usa más de una vez un color importante para la paleta, no lo almacena en una variable. * No hay nesting de elementos. | * Usa mixins pero para hacer repeticiones de código poco relevantes o erróneas. * Usa correctamente los operadores pero algunos están forzados o *hardcodeados*. * El nesting de los elementos es correcto pero desordenado. Falta de orden con los elementos en general. * Valores que usa de forma repetitiva como border-radius que podrían estar en variables no lo están. * El estudiante utiliza SCSS pero hay oportunidades de mejora en el uso del lenguaje para simplificar código que previamente era CSS. | * Utiliza operadores como each para armar clases de forma dinámica. * Utiliza variables para no tener que repetir una y otra vez valores como colores y tamaños de tipografías. * Utiliza nesting con un nivel de profundidad claro y al leer las líneas de código, se entiende a primera vista, que está trabajando. |
| --- | --- | --- | --- |

| **Implementación de módulos de Node** | | | |
| --- | --- | --- | --- |
| **Criterios** | **Bajo**  Falta más profundización. Es confuso. | **Correcto**  Acorde pero con errores puntuales. | **Óptimo**  Es claro y pertinente. |
| Utilización de módulos | * No hay package.json. * No hay librerías agregadas a las dependencias del package.json. * Se instalan más módulos de los que se usan. * El estudiante envía package.json con scripts y dependencias a pesar de no hacer uso de ningún módulo de NPM. * No hay scripts para crear los archivos necesarios para que el proyecto funcione, ya sea de SASS, CSS, o JS. * Instala más de un módulo para la misma tarea (como frameworks). | * Algunos scripts son los entregados en forma de prueba en las clases y no fueron editados por el alumno. * Se entrega (innecesariamente) del node\_modules, no entendiendo que su uso es solo local. * El package.json no tiene un nombre pertinente al proyecto. * Usa los scripts tal cual fueron entregados en clase, sin hacer modificaciones para adaptarlos a su propia estructura de archivos. * Módulos que parecen estar siendo usados, pero no aparecen en las dependencias. | * Los módulos de las librerías de las que depende el desafío están agregadas al package.json. * Los scripts reflejan la estructura de carpetas y archivos del alumno, y no son copiados/pegados de las clases. |

| **Repositorio en Github** | | | |
| --- | --- | --- | --- |
| **Criterios** | **Bajo**  Falta más profundización. Es confuso. | **Correcto**  Acorde pero con errores puntuales. | **Óptimo**  Es claro y pertinente. |
| Utilización de git | * No hay commits excepto uno solo donde inicia el repositorio. * Los mensajes de commit no son pertinentes a las actualizaciones hechas. | * Commits con demasiados cambios. * Commits con una cantidad muy chica de cambios. | * Realiza cambios pertinentes a un grupo de mejoras y las *commitea*. * Genera branches en caso de tener que testear algo experimental y luego hace el merge a master. * Utiliza .gitignore para no versionar los archivos o directorios que no son requeridos como node\_modules |
| Uso de Github | * El trabajo no está subido a Github. * Hay más de un repositorio para su proyecto en github. | * No creó la clave SSH pero se conectó a su repositorio haciendo uso del asistente de inicio de sesión de Github. * Entrega un repositorio del que podemos clonar y recibir todo el trabajo. | * Hay un readme.md con toda la información pertinente al proyecto y al estudiante. * Utiliza Github Pages para su trabajo. |