*3ER ENTREGA DEL PROYECTO FINAL*

**Componentes:**

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

*Estructura final de la web*

* **Formato:** Archivos HTML
* **Objetivos del desafío:**
  + **Estructura prolija y limpia:** El estudiante entrega su estructura de HTML con buena indentación, espacios prolijos, y comentarios siendo usados para comentar secciones y no para ocultar código.
  + **Maqueta de la web en base al framework elegido:** El estudiante deberá agregar elementos HTML según sus necesidades en armar contenedores o elementos web determinados, a través del código del 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 respetan 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 final 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, los mixins, 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.
* El estudiante utilizará .gitignore para evitar enviar archivos irrelevantes para la presentación como node\_modules.
* En el repositorio se muestran los commit que el estudiante usó para actualizar/versionar su código.

[**🔗 Desafío de ejemplo**](https://drive.google.com/file/d/1Zxh9z8RAIyceO0IcII69koTOy72tAfC7/view?usp=sharing)

*RÚBRICAS DE EVALUACIÓN*

*3ER ENTREGA DEL PROYECTO FINAL*

| **Estructura Final 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 muchas inconsistencias. * Mucho código deprecado comentado para ocultarlo. | * Uso de nuevas líneas y tabulaciones de manera prolija y consistente. * Tabulaciones correctas y ordenadas, denotando jerarquía entre los elementos. * Usa los comentarios para documentar secciones. | * Consistencia entre la estructura HTML de diferentes páginas. * Tabulaciones correctas y consistentes. |
| Tags HTML | * 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. * No todas o ninguna de las imágenes tienen alt. * Los alt están vacíos * Crea tags que envuelven a otros innecesariamente, ya sea porque no cumplen ninguna función o no se usan. | * Todas las imágenes tienen alt, pero el tag no es pertinente a la imágen o está mal escrito. * Nesting de elementos que no es aconsejable pero igualmente válido, como p dentro de p. | * 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. * Uso de etiquetas semánticas. * Uso óptimo de etiquetas, evitando crear por demás. |
| Estilo en el HTML | * Usa nombres poco legibles para las clases o que no hacen referencia su función. * Linkea al archivo de SCSS en su head en vez del CSS. * Algunas clases son redundantes o irrelevantes. * Uso indiscriminado de IDs. | * Linkea correctamente a el/los archivos de CSS que son generados por el SCSS. * Uso de IDs consistente y en camelCase o kebab-case. * Linkea correctamente a el/los archivos de CSS que son generados por el SCSS. * Clases redundantes como footer en el elemento footer. | * Nombres de clases consistentes y con BEM aplicado. * Linkea correctamente a el/los archivos de CSS que son generados por el SCSS. * El nombre del archivo generado por el SCSS es apropiado y no de prueba. |
| 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:// * Fotos de relleno o placeholders. | * La web tiene enlaces a todas las secciones en su navegación. * 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 contenido y no hay placeholders. |
| 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. * Imágenes demasiado pesadas y no están comprimidas u optimizadas para web. * Imágenes más grandes del tamaño que corresponde. | * El contenido no tiene faltas ortográficas o gramaticales. * Las páginas tienen una cantidad de contenido apropiado y está bien distribuida. * El contenido es apropiado a la sección en la que está ubicado. * Imágenes con un tamaño apropiado a lo que ocupan en el contenido. | * 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, lo que genera diferentes centros de interés visual y peso. * Las imágenes no se pixelan y están escaladas de manera proporcional. |

| **Estilo Final 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 | * Métodos de tabluar poco convencionales. * Tabulaciones erráticas y poco predecibles. * Nesting de selectores SCSS y sus respectivas clases desordenadas y con mala tabulación. * No hace uso de los &, para reutilizar el selector del padre. | * Uso de tabulaciones consistentes. * Declaraciones de reglas y de espacios entre los elementos estructurados correctamente. * Tabulaciones correctas para estructurar el selector. * El nesting de los selectores de SCSS es correcto pero no hace uso de los &, para reutilizar el selector del padre. | * Uso de tabulaciones y nesting bien estructurado en el SCSS. * Uso de & para realizar selectores óptimos con pocas repeticiones. |
| Entendimiento del CSS | * Usa reglas de CSS no pertinentes al elemento seleccionado. No recicla código y lo repite. * Usa demasiados elementos en su selector. | * Usa reglas de CSS pertinentes al selector. * Hay mucho código CSS que no usa y es de legado o deprecado. * Uso de variables de CSS correcto para evitar repetir algunos valores como colores y tamaños tipográficos. | * 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. |
| 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. | * Crea mixins y evita repetir código que usa a lo largo de su web. * 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. |
| Código de la estructura visual o layout | * Logra el layout de web haciendo uso de reglas CSS ineficientes como floats, o position. * Los layouts son correctos pero repite el código incluso aunque las estructuras sean iguales. | * Uso de br apropiado, separando los párrafos de texto. * A través del uso de clases especiales o *helper*, logra layouts diferentes reutilizando código y no reinventando el layouts similares. * Usa grid en situaciones en las que hubiera sido mejor flex y viceversa. * Fuerza flex en situaciones donde la estructura de bloques por sí sola sería la solución. | * Utilización de flex y grid pertinente al tipo de layout a generar. * No fuerza flex o grid para elementos que no lo necesitan y se resuelven con box-modelling |
| Diseño de la estructura visual o layout | * El diseño del layout de la web no es consistente a lo largo de las páginas. * Elementos de misma jerarquía son inconsistentes página a página. * La ubicación de elementos de navegación cambia de lugar. * El layout de la web no es intuitivo ni fácil de navegar. | * La interfaz web planteada por el layout es intuitiva y navegable * Los elementos de la misma jerarquía, son consistentes a lo largo de las diferentes páginas. * Los estilos definidos para los elementos se mantienen consistentes a lo largo de las páginas. | * No se conforma con los layouts clásicos y genera una estructura propia o poco convencional para el diseño web, pero aún así es navegable e intuitiva. |
| Diseño web atractivo | * Elementos no interactivos con transiciones que dan a entender lo contrario. * La paleta de colores elegida es problemática. * El texto no es legible. * No hay una paleta de colores. * El elemento contenedor presenta problemas de tamaño, alineamiento o posicionamiento. | * Transiciones decorativas. * Solo los elementos interactivos tienen transiciones que les dan estilo. * La paleta de colores varía a lo largo de las páginas. * Los elementos respetan la paleta pero tienen un diseño diferente página a página. * El texto es legible, pero con pocos niveles de lectura. | * El contraste entre los colores elegidos es apropiado. * Hay una paleta de colores y se respeta a lo largo de las páginas del sitio web. * Transiciones apropiadas y decorativas, no se hace un uso porque sí del recurso ni se lo abusa. * Las transiciones se usan con elementos que merecen la atención del usuario. |
| Utilización de Frameworks | * No usa las clases de responsive ya dadas por la librería. * No hay identidad propia a lo largo de las páginas, solo la del framework. * Pisa los estilos de su librería para darle su propio estilo. * Utiliza un theme descargado y realiza cambios mínimos. | * Recrea innecesariamente clases que ya trae el framework elegido. * Usa framework para hacer uso de pocos elementos. | * Utiliza el framework para maquetar y no para traerse solamente componentes (carousel, menu hamb, modal, etc). |
| Media queries & Responsive | * El sitio web no es usable en dispositivos más pequeños que desktop. * Elementos completamente fuera de cuadro que no se adaptan a los cambios de tamaño. * 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. * Hay demasiados queries porque no usa los breakpoints provistos por bootstrap o el framework que eligió. | * Hace uso de las columnas de bootstrap o de media queries propios para lograr responsividad. * Hay demasiados queries que no usa un framework y no define 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. * Elementos irrelevantes para los tamaños más chicos son removidos en favor de favorecer la lectura del resto. |

| **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. |