*ENTREGA DEL PROYECTO FINAL*

**Componentes:**

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

*Estructura final de la web*

* **Formato:** Archivos HTML
* **Objetivos del desafío:**
  + **Código prolijo y limpio:** El código HTML a lo largo de las páginas cuenta con buena indentación, espacios prolijos y coherencia con el contenido a mostrar.
  + **Sitio web estructurado en base al framework elegido:** El estudiante estructura su contenido teniendo en cuenta la identidad y estilo que está armando y el framework elegido.

**Incluir:**

* **Estructura :** 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
* **Objetivos del desafío:**
  + El estudiante deberá crear archivos de SCSS o SASS para darle estilo a su web.
  + El estudiante deberá 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.
  + El estudiante utiliza SASS para personalizar el framework que está usando.

**Incluir:**

* **Estilo avanzado:** El estilo del sitio web está por completo desde el SCSS. Estructurando las paletas y valores más importantes del estilo.

*Implementación de módulos de Node*

* **Formato:** Archivo package.json y package-lock.json
* **Objetivo del desafío:**
  + El estudiante deberá, en caso de utilizar otro módulo de npm, agregarlo como dependencia al proyecto.
  + El estudiante deberá en caso de haber agregado una nueva dependencia al proyecto, usarla en su código.

**Incluir:**

* **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 debe estar detallado cualquier librería o módulo que el estudiante agregue, para la versión final de su proyecto.

*Subida al servidor*

* **Formato:** Link al sitio web donde está subido el proyecto
* **Objetivo del desafío:**
  + El estudiante deberá utilizar WebHost000 o cualquier servicio de hosting para poner su página web online.

*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.

*RÚBRICAS DE EVALUACIÓN*

*ENTREGA DEL PROYECTO FINAL*

| **Desafío: 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 de su HTML/CSS. | * Consistencia entre la estructura HTML de diferentes páginas. * Tabulaciones correctas y consistentes. |
| Tags HTML | * Falta de h1 en algunas páginas. * Más de un h1 en algunas páginas. * No todas o ninguna de las imágenes tienen alt. * Crea tags que envuelven a otros innecesariamente, ya sea porque no cumplen ninguna función o no se usan. * Nesting de elementos que no es aconsejable pero igualmente válido, como p dentro de p. | * Todas las imágenes tienen alt, pero el tag no es pertinente a la imágen o está mal escrito. | * 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 uso de BEM. * 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 | * 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 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. |

| **Desafío: 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 | * 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 consistente. * Declaraciones de reglas y de espacios entre los elementos estructuradas correctamente. | * Nesting bien estructurado en el SCSS. * Tabulaciones correctas para estructurar el selector. * 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. * Hay mucho código CSS que no usa y es de legado o deprecado. | * Usa reglas de CSS pertinentes al selector. * 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 | * Hay errores con los operadores, variables o funciones de SASS. * Usa mixins pero para hacer repeticiones de código poco relevantes o erróneas. * El nesting de los elementos es correcto pero desordenado. Falta de orden con los elementos en general. | * Usa correctamente los operadores pero algunos están forzados o *hardcodeados*. * 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 valores como colores y tamaños de tipografías. * Divide la lógica en diferentes archivos de SASS haciendo uso luego del @import en uno solo para unirlos. |
| 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. | * 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. |
| 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). * Inserta sólo los módulos que desea de su framework. * Inserta la librería desde SASS. |
| 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 apropiadas y 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. * Los párrafos de texto se presentan monótonos y demasiado extensos. | * Hay una paleta de colores y se respeta a lo largo de las páginas del sitio web. * Los textos tienen varios niveles de lectura, volviendo dinámica la lectura/escaneo visual del contenido. * Transiciones apropiadas y decorativas usándose con elementos que merecen la atención del usuario. |
| 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. * Hay demasiados queries que no usa un framework y no define sus propios breakpoint. * 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. * Hace selectores muy específicos para los cambios de tamaño de los elementos, como pueden ser las tipografías. | * Uso de unidad 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. |

| **Desafío: Subida al servidor** | | | |
| --- | --- | --- | --- |
| **Criterios** | **Bajo**  Falta más profundización. Es confuso. | **Correcto**  Acorde pero con errores puntuales. | **Óptimo**  Es claro y pertinente. |
| Entendimiento del sistema de subida de archivos | * No están los archivos correctamente subidos al servidor. * El sitio web no está subido al hosting por completo, solo algunos archivos. | * Existen algunas imágenes o recursos mal linkeados por uso de URLs absolutas que hacían referencia a un entorno de archivos local. | * El sitio web es una copia fiel a lo que el estudiante estaba trabajando de forma local. * Las URLs del sitio web online son amigables. * Página 404 personalizada y funcional. |
| Entendimiento del servicio de hosting | * No entrega el trabajo subido a un servidor. |  | * Se entrega el trabajo subido a la web designada y se hace uso de las herramientas que provee el servicio para subir los archivos. * El estudiante aplica lo aprendido para subir su sitio web a un servicio de hosting propio o que considero más apropiado para su trabajo. |

| **Desafío: 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. * Usa las clave SSH para conectarse con Github desde su computadora. * Utiliza Github Pages para su trabajo. |