RÚBRICA DE EVALUACIÓN DE ENTREGAS DEL PROYECTO FINAL

Una **rúbrica** es un conjunto de criterios y normas que sirven para evaluar el nivel de desempeño en una tarea. Es una herramienta de calificación que simplifica la tarea de corrección y permite lograr una evaluación más objetiva y transparente tanto para el evaluador como el evaluado.

Adicionalmente, la rúbrica ayuda a igualar y alinear los criterios de evaluación entre los diferentes docentes y tutores.

En resumen, la rúbrica:

1. Define los criterios de evaluación de cada entrega de proyecto final.
2. Describe y detalla los puntos clave que debe tener cada entrega.
3. Facilita y transparenta la corrección de cada trabajo.

Introducción a las Rúbricas

Las rúbricas son un esfuerzo por crear un método unificado y homogéneo de evaluar los desafíos, brindando transparencia a las correcciones. Sin embargo es importante aclarar que es un material de referencia y no tanto una guía, ya que en el mismo se sientan las bases de lo que se espera de los desafíos. Hay que tener en cuenta que cada estudiante hace un recorrido único y es por eso que es fundamental que tomen este material como referencia para cada situación.

Dicho eso, se plantean tres estados para clasificar la calidad y el desempeño de los desafío: **bajo, correcto y óptimo.** Se entiende que cada columna, itera sobre la base de la otra, y se asume que por ejemplo, errores de la columna **baja**, no van a ser encontrados en un trabajo **correcto** ni **óptimo**. A su vez, se espera que el estudiante mejore su desafío con cada entrega, por lo que algunas correcciones **óptimas** son *degradadas* a solo **correctas** o quizás **bajas**, para momentos más avanzados de las entregas, denotando el aumento de exigencia con cada nueva clase.

También es importante notar que no todos los estudiantes van a llegar a entregas **óptimas** ya que en esta columna suelen encontrarse características de un trabajo superador. Un trabajo **óptimo** es al que esperamos que todos aspiren, ya que es donde encontramos la mayor cantidad de exigencia. Lo importante es notar que esto no equivale a un trabajo **correcto** sea considerado como erróneo, sino donde hay mucho espacio para la mejora y el feedback.

1RA ENTREGA DEL PROYECTO FINAL

Componentes:

1. Prototipo de la web
2. Estructura Inicial de la Web
3. Estilo inicial de la web

Prototipo de la web

- **Formato:** Archivo PDF o de Imagen
- **Objetivo del desafío:** El estudiante deberá prototipar la web para tener una idea clara del resultado al que quiere llegar.

Incluir:

- **Wireframe/Prototipo:** crear una versión que muestre cómo se verá el sitio cuando esté productivo. No es necesario entrar en detalle ni realizar un mockup completamente desarrollado; puede elaborar un boceto a mano o con Sketch.
- **Estructura de la web prototipada:** crear una estructura donde se organizan los elementos que van a estar en su web. El nivel de detalle no es importante, sino más bien las posiciones que los elementos van a tener y su tamaño aproximado.

Estructura inicial de la web

- **Formato:** Archivos HTML
- **Objetivos del desafío:**

- **Maquetar la web:** El estudiante deberá utilizar los tags, en especial los semánticos, para desde el código describir la estructura de la web.
- **Estructuración del contenido:** El estudiante deberá llevar su contenido a la estructura HTML haciendo uso de los tags que corresponden para el contenido a insertar.

Incluir:

- **Etiquetas semánticas:** utilizar los tags semánticos de HTML5.
- **Contenido:** agregar etiquetas que van a servir para denotar dónde va a haber contenido como imágenes, párrafos y titulares.
- **Páginas:** incluir las secciones de su sitio ya maquetadas con la estructura propia de cada página.

Estilo inicial de la web

- **Formato:** Archivo CSS
- **Objetivo del desafío:**
  - El estudiante deberá comenzar a darle estilo básico a su web definiendo colores. El documento debe estar linkeado en las páginas del proyecto.

Incluir:

- **Estilo básico:** agregar al CSS colores pertinentes al diseño inicial que está planteando el estudiante.

🔗 **Desafío de ejemplo**

RÚBRICAS DE EVALUACIÓN
1RA ENTREGA DEL PROYECTO FINAL

Desafío: Prototipo de la web			
Criterios	**Bajo** Falta más profundización. Es confuso.	**Correcto** Acorde pero con errores puntuales.	**Óptimo** Es claro y pertinente.
Estructura clara	• No condice con lo que **podría** ser una web. -El diseño del prototipo es difuso o no hay una navegación clara. • No se notan las diferentes secciones o elementos.	• La estructura del sitio condice con lo que será el sitio vivo y las secciones que podría contener.	• El prototipo representa el modelo vivo y se intuye una navegación clara de la que espera un usuario que navega un sitio web.
Diseño Simple	• No se intuyen secciones marcadas y definidas, sino que todo parece ser un gran torrente de información. • Se detiene demasiado en los detalles, tratando de emular lo que va a ser la web, cuando debería ser más representativo.	• Las secciones se identifican claramente y son diferenciables, pero tienen muy poco o nulo contenido. • El diseño del prototipo utiliza un formato del que luego se puede partir a un sitio web funcional.	• Las secciones son identificables y tienen una cantidad de contenido apropiada para esta etapa del diseño. • El diseño es simple y da espacio a que el diseño futuro pueda crecer a partir de él.

Desafío: Estructura inicial de la web			
Criterios	**Bajo** Falta más profundización. Es confuso.	**Correcto** Acorde pero con errores puntuales.	**Óptimo** Es claro y pertinente.
Código prolijo	• Hace poco uso de tabulaciones, de manera errática, sin diferencias entre “padres” e “hijos”. • Los elementos comienzan en una columna (ya sea por la tabulación o por falta de nueva línea) y terminan en otra. • Uso de nuevas líneas erráticas.	• El uso de tabulaciones y/o nuevas líneas marca de forma correcta la jerarquía de padre/hijo pero hay inconsistencias; ya sea que es es errático, usa diferentes niveles de tabulaciones, usa diferentes tamaños de tabulaciones, o los elementos no cierran en el mismo nivel en el que empiezan.	• El uso de tabulaciones es óptimo para declarar las jerarquías padre/hijo. • Los elementos padre comienzan y terminan en la misma columna. • No hay inconsistencias en el uso de nuevas líneas o HTML.
Comentarios <!-- -->	• Errores de tipeo a la hora de escribir correctamente el tag de comentario, o los nestea. • Se usan (en demasía) para ocultar código que el estudiante no quiere que el explorador muestre.	• Se usan demasiados comentarios (seguramente por la falta de tags semánticos).	• El uso de comentarios es correcto: se usan de forma pragmática para saber dónde comienzan y/o terminan secciones.
Tags HTML	• Tags mal escritos o deprecados en HTML5 (center, font, blink, etc). • Atributos mal escritos o	• Nesting acorde en general. • Errores mínimos a la hora de escribir tags. • Tags semánticos usados pero	• Buen nesting, entiende claramente el uso del mismo para darle jerarquía y estructura a los elementos.

	deprecados (align, border, etc). ● Repetición de atributos en un mismo tag. ● Mal nesting. ● Uso nulo de tags semánticos. ● Tags hijo o padre que no están permitidos (li dentro de p, p dentro de ul, etc). ● Repite IDs.	con problemas para comprender cuál sirve para cada caso.	● Los tags y atributos están bien escritos. ● Usa los tags semánticos de forma correcta para estructurar la página desde el HTML, entendiendo el uso de cada uno.
Estilo en el HTML	● Mal linkeo de hojas de estilo o fuera del head. ● Usa el tag style para agregar estilo a la página. ● Usa el atributo style para darle estilo al elemento. ● Usa nombres para las clases de forma inconsistente (en algunos elementos kebab-case en otros camelCase y en otros todojunto) ● Utiliza clases que no son descriptivas para lo que hace en el estilo. ● Usa caracteres prohibidos en el atributo clase. -Usa IDs como si fuesen clases.	● Hojas de estilo bien linkeadas y en las mismas se encuentra todo el CSS. ● Usa demasiadas clases innecesariamente, no las re-utiliza ni organiza correctamente. ● Declara clases de CSS en el HTML pero éstas no aparecen en las hojas de estilo.	● Hojas de estilo bien linkeadas y en las mismas se encuentra todo el CSS. ● Reutiliza clases para no re-definir reglas de CSS para tags con estilos iguales (como colores, fondos, etc). ● 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 cómo 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 que representan secciones aún no desarrolladas, por lo que estos no llevan al lugar correspondiente. • Las fotos están bien cargadas pero son de relleno (no tienen un propósito o funcionan como “placeholder”). • Hay enlaces que llevan a diferentes páginas y permiten retornar a la home. • Se usan rutas relativas para los archivos de la web.	• Los enlaces a las secciones funcionan. • Las páginas están correctamente interconectadas y el usuario puede navegar entre ellas. • Las imágenes están bien cargadas y el contenido es pertinente a la web (no simulado o placeholder). • El enlace a la home es el logo. • Las rutas relativas son correctas.

Desafío: Estilo inicial 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	• El uso de tabulaciones es poco o es errático. • Usa diferentes formas de escribir las reglas de CSS: mezclando algunos la forma en la que escribe los	• Es consistente el uso de tabulaciones y las declaraciones de reglas. • Métodos de tabulaciones poco convencionales o de una forma poco predecible	• Consistente el uso de tabulaciones, declaraciones de reglas y de espacios entre los elementos.

	selectores, algunos poniendo todas las líneas a sola línea.	para otro que mire el código.	

Entendimiento del CSS	• Usa solo IDs para dar estilo a los elementos.
	• Errores de CSS usando reglas que no corresponden al elemento, como por ejemplo un p con list-style-none.
	• Errores en los selectores:
	  ○ # en vez de .
	  ○ Errores al hacer selectores de tags
	  ○ Nesting de CSS
	  ○ Nula selección de etiquetas por anteponer un símbolo
	• Nombres de clases con poca consistencia o no son predecibles.
		• Muestra algunos errores a la hora de hacer selectores o equivoca los estilos.
		• Le da estilo a elementos a través de selectores e IDs según lo requiera.
		• Las selecciones son correctas, pero poco óptimas, usando demasiados elementos para una regla de CSS.
		• Usa solo IDs para armar los selectores de CSS por desconocimiento de las clases.
		• No recicla código.
		• Hace uso de demasiados elementos.
		• Mal uso de comentarios.
			• Recicla de forma óptima.
			• Usa clases para armar
			• Las reglas de CSS que utiliza son correctas para el elemento en el que las utiliza.
			• El código está dividido en secciones o porciones de código predecibles.
			• Le da estilo a elementos a través de selectores e IDs según lo requiera.

Diseño de la web	• Estructura demasiado desordenada y/o errática. (Aclaración: Dada la fase de aprendizaje del estudiante, aún no hay herramientas de CSS para estructurar el contenido de forma correcta).	• Uso de elementos para reflejar el diseño de la web prototipado originalmente desordenados por el tipo de tags elegidos.	• Uso correcto de los elementos básicos para reflejar el diseño web prototipado originalmente.

2DA ENTREGA DEL PROYECTO FINAL

Componentes:

1. Estructura básica de la Web
2. Estilo básico de la Web

Estructura de la web

- **Formato:** Archivos HTML
- **Objetivo del desafío:**
  - **Maquetar la web:** El estudiante deberá, en base a la estructura visual que desea plantear con las nuevas herramientas de Flexbox y Grids, agregar y/o reestructurar los elementos de su HTML.
  - **Estructuración del contenido:** El estudiante deberá llevar su contenido a la estructura HTML haciendo uso de los tags que corresponden para el contenido a insertar.

Incluir:

- **Maquetado de la web:** Incluir etiquetas en base al contenido que incluirá y su futura estructuración visual desde el estilo.
- **Páginas:** agregar más contenido a sus páginas utilizando las etiquetas correspondientes.

Estilo básico de la web

- **Formato:** Archivo CSS

- **Objetivo del desafío:**
    - El estudiante deberá seleccionar la paleta de color y respetarla consistentemente en los nuevos elementos creados.
    - El estudiante deberá definir tamaños tipográficos que reflejan lo planteado en el prototipo.
    - El estudiante deberá usar reglas de CSS para darle estilo y forma a los elementos web.
    - El estudiante deberá aplicar lo aprendido en box-modelling, flex, y grids para estructurar el contenido de su web armando layouts en las que distribuye el contenido.

Incluir:

- **Estructuración visual del contenido:** usar reglas de CSS para crear layouts y distribuir los elementos de su sitio web en base al diseño planteado originalmente. Debe valerse de flex o box-modeling para estructuras unidimensionales, o grid para bidimensionales.
- **Estilo básico de la web:** crear paletas para darle un estilo consistente a su sitio web. Ésto acompañado de la utilización de otros elementos como las tipografías, el uso de sombras, el radio de los bordes o el espaciado entre los elementos entre otros, para forjar el estilo que la web va a mantener página a página.

🔗 **Desafío de ejemplo**

RÚBRICAS DE EVALUACIÓN

2DA ENTREGA DEL PROYECTO FINAL

Desafío: Estructura básica 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; ya sea que es es errático, usa diferentes niveles de tabulaciones, usa diferentes tamaños de tabulaciones, o los elementos no cierran en el mismo nivel en el que empiezan.	• Tabulaciones correctas y ordenadas, denotando jerarquía entre los elementos. • No hay saltos erráticos y hay un tamaño consistente de tabulación.	• La estructura es clara por el uso de tabulaciones y líneas nuevas consistentes en tamaño y cantidad. • Las etiquetas comienzan y terminan en la misma columna.
Comentarios <!-- -->	• Se usan demasiados comentarios (seguramente por la falta de tags semánticos).	• Los comentarios se usan de forma pragmática para saber dónde comienzan y/o terminan secciones. • Comenta demasiado código para uso futuro o porque ya no lo va a usar.	• Usa los comentarios de forma pragmática y pertinente para documentar secciones de su HTML/CSS.
Tags HTML	• Nesting acorde en general. • Errores mínimos a la hora de escribir tags. • Tags semánticos usados pero con problemas para comprender	• Buen nesting, usándolo con el fin de jerarquizar y estructurar los elementos. • Los tags y atributos están bien escritos.	• El HTML no contiene errores en los atributos y tags. • El nesting es óptimo. Uso correcto de tags semánticos y estructuración de la página

	cual sirve para cada caso.	• Crea tags que envuelven a otros innecesariamente, ya sea porque no cumplen ninguna función o no se usan.	desde el HTML.
Estilo en el HTML	• Hojas de estilo bien linkeadas y con todo el CSS. • Usa demasiadas clases innecesariamente por aún no saber cómo reutilizarlas u organizarlas. • Clases de CSS declaradas en el HTML pero no aparecen en las hojas de estilo.	• Hojas de estilo bien linkeadas. • Las hojas de estilo tienen todo el CSS. • Uso de clases CSS eficiente y óptimo. Re-utilizan clases para no re-definir reglas de CSS para tags con estilos iguales (como colores, fondos, etc).	• Uso de clases eficiente en los atributos. • 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, que representan secciones que van a desarrollarse. • Las fotos están bien cargadas pero son de relleno (no tienen un propósito o funcionan como “placeholder”). • Hay enlaces que llevan a diferentes páginas y tiene como volver a la home. • Se usan rutas relativas para los archivos de la web.	• Los enlaces a las secciones funcionan correctamente. • Las páginas están correctamente interconectadas y el usuario puede navegar entre ellas. • Las imágenes están bien cargadas y el contenido es pertinente a la web (no simulado o placeholder). • El enlace a la home es el logo. • Las rutas relativas son correctas.
Contenido	• El contenido tiene errores ortográficos. • Hay contenido que no corresponde a la página donde está ubicado.	• El contenido está bien escrito y no presenta faltas ortográficas. • Hay poco contenido o está incompleto en algunas de las páginas.	• Las páginas tienen una cantidad contenido apropiado y está bien distribuida. • El contenido no tiene faltas ortográficas o gramaticales.

	• El contenido está desorganizado.	• La organización del contenido es equitativa pero mal distribuida a lo largo de la página.	• El contenido es apropiado a la sección en la que está ubicado.

Desafío: Estilo básico 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	• Es consistente el uso de tabulaciones y las declaraciones de reglas. • Métodos de tabulaciones poco convencionales. • Tabulaciones erráticas y poco predecibles.	• Uso de tabulaciones consistente. • Declaraciones de reglas y de espacios entre los elementos estructuradas correctamente.	• Uso de código óptimo. • Tabulaciones correctas para estructurar el selector con sus reglas en el CSS. • Las secciones o porciones de código en la que está estructurado el código se marcan con comentarios.
Entendimiento del CSS	• Errores a la hora de hacer selectores. • Usa reglas de CSS no pertinentes al elemento seleccionado. No recicla código y repite mucho. • Usa porcentajes o unidades de viewport (vh o vw) para demasiados elementos, volviendo impredecible el	• Recicla código de forma poco óptima, Usa demasiados elementos en su selector. • Las reglas de CSS que utiliza son correctas para el elemento en el que las utiliza. • Los selectores se ordenan desde el CSS tiene sentido a lo largo del archivo, agrupando diferentes selectores de acuerdo ya sea a la página a la que pertenecen o a la	• Selectores de CSS correctos que corresponden a elementos HTML presentes en los archivos. • Uso de clases para evitar repetir código. • Usa reglas de CSS pertinentes al selector. • Usa unidades correctas para el tipo de elemento seleccionada.

	tamaño de los elementos.	función que cumplen o ambas.	
Código de la estructura visual o layout	• Se usa posición absoluta o relativa para armar el layout visual de los contenidos. • Se usa float para armar columnas de contenido. • Se usan tablas para armar layouts. • Uso de br para separar elementos en vez de usar reglas de spacing.	• Logra el layout de web haciendo uso de reglas CSS ineficientes. • Usa flex para layouts multidimensionales complejos que se favorecen del uso de grids. • Usa grids para layouts unidimensionales simples que se favorecen del uso de flex. •	• Usa reglas apropiadas para armar los diferentes tipos de layout. • El uso de tablas es solo para mostrar contenido estructurado. • El uso de flex y grid es apropiado para el tipo de layout armado. • Uso de br apropiado, separando los párrafos del texto.
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. • El estilo visual de los elementos web cambia mucho página a página. • La ubicación de elementos de navegación cambia de lugar. • El layout de la web no es intuitivo o de fácil navegación • El estilo visual de algunos elementos web es el default.	• El diseño del layout de la web es consistente a lo largo de algunas páginas pero no en todas. • El layout del sitio web es navegable. • Los elementos de la misma jerarquía, son consistentes a lo largo de las diferentes páginas. • El estilo visual de los elementos web se mantiene consistente a lo largo de las páginas.	• El diseño del layout es consistente página a página. • Los estilos definidos para los elementos se mantienen consistentes a lo largo de las páginas. • La interfaz web planteada por el layout es intuitiva y navegable.
Diseño web atractivo	• Elementos fuera que no pertenecen a sección sueltos a lo largo de la página. • Uso de colores chillantes o	• Uso correcto de colores pero el texto no es legible. • La paleta de colores varía a lo largo de las páginas.	• El contraste entre los colores es apropiado. • Hay una paleta de colores y se respeta a lo largo de las páginas

	con mucho contraste entre sí. • El texto no es legible. • La página ocupa todo el ancho del navegador lo que dificulta la lectura. • No hay una paleta de colores.	• Hay un elemento contenedor pero es demasiado grande o demasiado chico. • El elemento contenedor no está centrado o alineado a nada. • Los elementos respetan la paleta pero varían en diseño página a página.	del sitio web. • El texto es legible. • Hay un elemento contenedor que evita que el contenido vaya hasta los bordes de la pantalla y está alineado.

3RA ENTREGA DEL PROYECTO FINAL

Componentes:

1. Estructura avanzada de la web
2. Estilo avanzado de la web

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
- **Objetivo del desafío:**
  - El estudiante deberá crear archivos de CSS para darle estilo a su web.
  - El estudiante deberá agregar transformaciones, animaciones y/o transiciones para otorgarle dinamismo a la web en elementos que tengan interacción con el usuario.
  - El estudiante deberá hacer uso de selectores de CSS para poder darle su estilo a los elementos que ya vienen con su propia identidad del framework.

Incluir:

- **Estilo avanzado:** Se le mejorarán los elementos interactivos con variaciones en sus diferentes estados, ya sea de la mano de transformaciones, transiciones y/o animaciones.
- **Estilo del Framework:** No todos los elementos del framework van a tener una estética que condice con el sitio en el que son implementados, por lo que se usará CSS para darles un estilo acorde.
- **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.

🔗 **Desafío de ejemplo**

RÚBRICAS DE EVALUACIÓN
3RA ENTREGA DEL PROYECTO FINAL

Desafío: 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.	• 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. •

	• Comentan demasiado código para uso futuro o porque ya no lo va a usar.		
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.

Desafío: 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.	• 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.

	• No recicla código y lo repite. • Los selectores son innecesariamente precisos, previniendo una óptima reutilización del código.		
Código de la estructura visual o layout	• Logra el layout de web haciendo uso de reglas CSS ineficientes como floats, o position. • Usa grids o flex para estructuras que se resuelven fácilmente con box-modelling.	• Uso de br apropiado, separando los párrafos de texto. • 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. • Utilización de flex y grid pertinente al tipo de layout a generar.
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 o fácil de navegar.	• El diseño del layout de la web es consistente a lo largo de algunas páginas pero no en todas. • El layout del sitio web es navegable. • Los elementos de la misma jerarquía, son consistentes a lo largo de las diferentes páginas.	• El diseño del layout es consistente página a página. • Los estilos definidos para los elementos se mantienen consistentes a lo largo de las páginas. • La interfaz web planteada por el layout es intuitiva y navegable.
Diseño web atractivo	• Uso de transiciones muy lentas. • Uso de transiciones exageradas. • Elementos fuera que no	• Transiciones apropiadas y decorativas. • Solo los elementos interactivos tienen transiciones que les dan estilo.	• Transiciones y animaciones apropiadas y decorativas moderadamente. • El contraste entre los colores es apropiado. Hay una paleta de

	pertenecen a sección sueltos a lo largo de la página. ● Animaciones muy largas y molestas para la navegación. ● Elementos no interactivos con transiciones que dan a entender lo contrario. ● Uso de colores chillantes o con mucho contraste entre sí. ● El texto no es legible. ● La página ocupa todo el ancho del navegador lo que dificulta la lectura. ● No hay una paleta de colores.	● Uso de colores correcto pero el texto no es legible. ● La paleta de colores varía a lo largo de las páginas. ● Hay un elemento contenedor pero es demasiado grande o demasiado chico. ● El elemento contenedor no está centrado o alineado a nada. ● Los elementos respetan la paleta pero tienen un diseño diferente página a página.	colores y se respeta a lo largo de las páginas del sitio web. ● El texto es legible. ● Hay un elemento que evita que el contenido vaya hasta los bordes de la pantalla y está alineado.
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.	● 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.	● 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.

	• Utiliza un theme descargado y realiza cambios mínimos.		
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 unidad rem a lo largo de los tamaños tipográficos para hacer luego cambiar el tamaño de forma pareja desde :root. • 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.

4TA 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**

RÚBRICAS DE EVALUACIÓN
4TA 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.	• 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 en kebab-case. • 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.	• 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,	• 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

	• Se usan rutas absolutas para archivos de la web, incluso haciendo uso del protocolo file:// • Fotos de relleno o placeholders.	sino relativas.	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 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. • Uso del srcset para usar imágenes diferentes en los diferentes viewport/queries.

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	• 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 consistente. • Declaraciones de reglas y de espacios entre los elementos estructuradas 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	• 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	• 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.

	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.	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.	
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.	• Personaliza el framework haciendo uso de las variables del mismo sin superponer las clases con cascada. Utiliza un framework no visto en clase de forma óptima. • Inserta sólo los módulos que desea de su framework. • Inserta la librería desde SASS.
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	• 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	• Uso de unidad rem a lo largo de los tamaños tipográficos para hacer luego cambiar el tamaño de forma pareja desde :root. • El sitio web cuenta con una buena navegación en numerosos tamaños, en particular en mobile, laptop y desktop.

	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.	para los cambios de tamaño de los elementos, como pueden ser las tipografías.	• Elementos irrelevantes para los tamaños más chicos son removidos en favor de favorecer la lectura del resto.

Desafío: 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	• 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	• 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.

	SASS, CSS, o JS. • Instala más de un módulo para la misma tarea (como frameworks).	modificaciones para adaptarlos a su propia estructura de archivos. • Módulos que parecen estar siendo usados, pero no aparecen en las dependencias.	

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.

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	• Consistencia entre la estructura HTML de diferentes páginas. • Tabulaciones correctas y consistentes.

		HTML/CSS.	
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 en kebab-case. • 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	• 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	• Las páginas tienen enlaces funcionales. • Las imágenes tienen rutas relativas y correctas. • Las páginas están interconectadas correctamente. • Todas las imágenes son

	falta de enlaces. • Se usan rutas absolutas para archivos de la web, incluso haciendo uso del protocolo file:// • Fotos de relleno o placeholders.	para los archivos de la web, sino relativas.	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. • Uso del srcset para usar imágenes diferentes en los diferentes viewport/queries.

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	• Recrea innecesariamente clases que ya trae el framework elegido. • Usa framework para hacer uso de pocos elementos.	• Personaliza el framework haciendo uso de las variables del mismo sin superponer las clases con cascada. • Utiliza un framework no visto en clase de forma óptima. • Inserta sólo los módulos que desea de su framework. • Inserta la librería desde SASS.

	descargado y realiza cambios mínimos.		
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	• 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 rem a lo largo de los tamaños tipográficos para hacer luego cambiar el tamaño de forma pareja desde :root. • 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.

	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.		

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.