

# Metodologías para la organización y el preprocesador Sass

Sass y el patrón 7-1

***Implementar una interfaz  
de usuario web utilizando  
buenas prácticas  
en el manejo de estilos  
para brindar un aspecto  
visual e interacciones  
acordes a lo requerido***

- Unidad 1:  
Metodologías para la organización  
y el preprocesador Sass
- Unidad 2:  
El modelo de cajas y el Layout
- Unidad 3:  
Utilizando Bootstrap como  
Framework CSS



Te encuentras aquí



## ¿Qué aprenderás en esta sesión?

- *Implementa estilos en una interfaz web de baja complejidad utilizando preprocesador SASS.*

¿Cuáles son algunas de  
las ventajas de trabajar  
con Sass?



***/\* Sass y el patrón 7-1 \*/***

# Patrón 7-1



- Estructurar un proyecto con Sass es la base para crear estilos consistentes, mantenibles y escalables.
- El patrón 7-1 es una estructura creada por Hugo (Kitty) Giraudel, la cual se basa en un principio muy simple: **7 directorios, 1 archivo**.
- Esta técnica separa los parciales en 7 directorios diferentes y un archivo que funciona como manifiesto.

# ¿Cómo implementar el patrón 7-1?

- Para implementar el patrón 7-1 en nuestro proyecto deberemos, en primer lugar, conocer más sobre su estructura.
- Ingreseemos a [sass guidelines](#) para conocer más sobre el patrón 7-1.
- La estructura base de esta técnica consta de 7 directorios que se encuentran dentro de una **carpeta raíz**.

**{desafío}**  
**latam\_**

```
sass/
|
|- abstracts/
|   |- _variables.scss    # Sass Variables
|   |- _functions.scss    # Sass Functions
|   ...                  # etc..
|
|- base/
|   |- _reset.scss        # Reset/normalize
|   |- _typography.scss   # Typography rules
|   ...                  # Etc.
|
|- components/
|   |- _buttons.scss      # Buttons
|   |- _carousel.scss     # Carousel
|   ...                  # Etc.
|
|- layout/
|   |- _navigation.scss   # Navigation
|   |- _grid.scss         # Grid system
|   ...                  # Etc.
|
|- pages/
|   |- _home.scss         # Home specific styles
|   |- _contact.scss      # Contact specific styles
|   ...                  # Etc.
|
|- themes/
|   |- _theme.scss        # Default theme
|   |- _admin.scss        # Admin theme
|   ...                  # Etc.
|
|- vendors/
|   |- _bootstrap.scss    # Bootstrap
|   |- _jquery-ui.scss    # jQuery UI
|   ...                  # Etc.
|
|- main.scss              # Main Sass file
```

# ¿Cómo implementar el patrón 7-1?

Esta carpeta raíz aloja en su interior un **manifiesto** que contendrá a su vez los archivos parciales de nuestro proyecto Sass.

```
sass/  
  base/  
  components/  
  layout/  
  pages/  
  themes/  
  abstracts/  
  vendors/  
  main.scss
```



# Archivos parciales de un proyecto Sass

## *Abstracts*

La carpeta **abstracts** contiene todos los parciales relacionados con todas las herramientas y helpers de Sass usados en nuestro proyecto. Aquí encontraremos las variables globales, funciones y mixin que deseemos guardar.

**La regla de oro de este directorio es que ninguno de los elementos incluidos dentro de este se deben compilar a CSS.**

```
- abstracts/  
|   |- _variables.scss      # Sass Variables  
|   |- _functions.scss     # Sass Functions  
|   |- _mixins.scss        # Sass Mixins  
|   |- _placeholders.scss  # Sass Placeholders
```

# Archivos parciales de un proyecto Sass

## Base

El directorio **base** será, como dice su nombre, la base de nuestro proyecto. En él podremos guardar todos los estilos principales como las tipografías, reset y estilos comúnmente usados en HTML.

```
| - base/  
|   | - _reset.scss          # Reset/normalize  
|   | - _typography.scss    # Typography rules  
|   | - _base.scss          # Base styles  
|   ...                     # Etc.
```

# Archivos parciales de un proyecto Sass

## *Components*

En **components** irán todos los parciales que representan un pequeño componente de nuestra interfaz. Elementos que cumplan una función de manera modular como carruseles, dropdown, thumbnails serán bienvenidos en este directorio.

```
| - components/  
|   | - _buttons.scss      # Buttons  
|   | - _carousel.scss    # Carousel  
|   ...                   # Etc.
```

# Archivos parciales de un proyecto Sass

## Layout

**Layout** contiene todo lo que tenga que ver con el layout o diseño de la interfaz. Esto incluye los elementos típicos del layout como los header, footer, sidebar, navegación, además de las grillas que le dan forma a este diseño.

```
| - layout/  
|   | - _navigation.scss    # Navigation  
|   | - _grid.scss         # Grid system  
|   ...                    # Etc.
```

# Archivos parciales de un proyecto Sass

## *Pages*

En el caso del directorio **pages**, si tuviéramos distintos tipos de diseño en las páginas deberíamos agregar este archivo.

En el caso de no tener distintos tipos de páginas, este puede quedar sin contenido.

```
| - pages/  
|   | - _home.scss           # Home specific styles  
|   | - _contact.scss       # Contact specific styles  
|   ...
```

# Archivos parciales de un proyecto Sass

## *Themes*

En interfaces grandes, es común encontrar este tipo de estilos, los cuales tienen diferentes formas de definir un tema dependiendo de factores, como el tipo de usuario que la usará o si la marca usa diferentes colores dependiendo del lugar geográfico.

Esta carpeta en proyectos pequeños o específicos es probable que no sea necesario usarla, de modo que si ese es el caso, lo mejor que podemos hacer es dejarla vacía.

```
| - themes/  
|   | - _theme.scss      # Default theme  
|   | - _admin.scss     # Admin theme  
|   ...                  # Etc.
```

# Archivos parciales de un proyecto Sass

## *Vendor*

Finalmente, encontraremos el directorio **vendor**. En esta carpeta se encuentran todos los archivos CSS de bibliotecas, plugins y frameworks externos como: Normalize, Bootstrap, entre otras.

El usar esta carpeta es una buena manera de separar el código que es nuestro con el código de otras personas. En algunos proyectos puede que no sea necesario crear todos los directorios, ya que por ejemplo un proyecto que no usa librerías externas no será necesario crear la carpeta vendors.

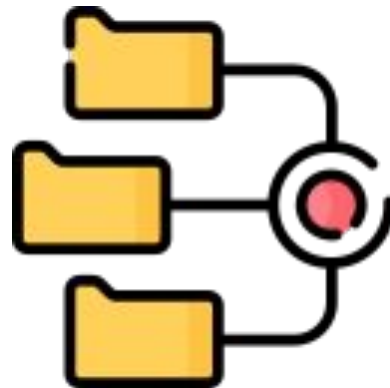
```
| - vendors/  
|   | - _bootstrap.scss      # Bootstrap  
|   | - _jquery-ui.scss     # jQuery UI  
|   ...                     # Etc.
```

# Archivos parciales de un proyecto Sass

*main.scss*

Un último elemento crucial de esta estructura es el archivo **main.scss**.

Este como sabemos es el **manifiesto** que contendrá todo el código que será compilado a CSS.





# Orden de carga patrón 7-1

La carga dentro de este archivo deberá tener la siguiente estructura a modo de evitar errores de carga o cascada.

```
abstracts/  
vendors/  
base/  
layout/  
components/  
pages/  
themes/
```

Esto significa que al ingresar los archivos dentro del manifiesto, estos deberán ser ingresados comenzando con parciales de abstracts y terminando con themes.

# Orden de carga patrón 7-1

Según nos especifica Sass guideline: Para mantener la legibilidad en el manifiesto, el archivo deberá:

- Tener un archivo por @import.
- Haber un @import por línea.
- No hay que dejar una línea después de importar archivos de la misma carpeta.
- La extensión del archivo y guión bajo deberá ser omitido de la ruta.

Siguiendo estas recomendaciones obtendremos el siguiente resultado:

**{desafío}**  
**latam\_**

```
// main.scss # Manifest File
@import 'abstracts/variables';
@import 'abstracts/functions';
@import 'abstracts/mixins';
@import 'abstracts/placeholders';
@import 'vendors/bootstrap';
@import 'vendors/jquery-ui';
@import 'base/reset';
@import 'base/typography';
@import 'layout/navigation';
@import 'layout/grid';
@import 'layout/header';
@import 'layout/footer';
@import 'layout/sidebar';
@import 'layout/forms';
@import 'components/buttons';
@import 'components/carousel';
@import 'components/cover';
@import 'components/dropdown';
@import 'pages/home';
@import 'pages/contact';
@import 'themes/theme';
@import 'themes/admin';
```

# Sintaxis Sass

Un punto importante sobre el uso de Sass, es que este presenta dos tipos de sintaxis para escribir su código.

- **Sintaxis Sass**

Esta sintaxis es un poco diferente de la sintaxis de CSS estándar.

Por ejemplo, te evita colocar puntos y coma al final de los valores de propiedades. Además, las llaves no se usan y en su lugar se realizan indentados.

- **Sintaxis SCSS**

Es una sintaxis bastante similar a la sintaxis del propio CSS. El código CSS es código SCSS válido. Podríamos decir que SCSS es código CSS con algunas cosas extra.

Al usar **SCSS** tienes la ventaja de que el código CSS de toda la vida será válido para el preprocesador, pudiendo reutilizar los conocimientos básicos de CSS y códigos de estilo con los que ya se ha trabajado en los proyectos anteriores.

# Sintaxis Sass

## Sass vs SCSS

Acá vemos un ejemplo de las principales diferencias:

### SCSS

```
1 .button {
2   background: cornflowerblue;
3   border-radius: 5px;
4   padding: 10px 20px;
5
6   &:hover {
7     cursor: pointer;
8   }
9
10  &:disabled {
11    cursor: default;
12    background: gray;
13    pointer-events: none;
14  }
15 }
```

### Sass

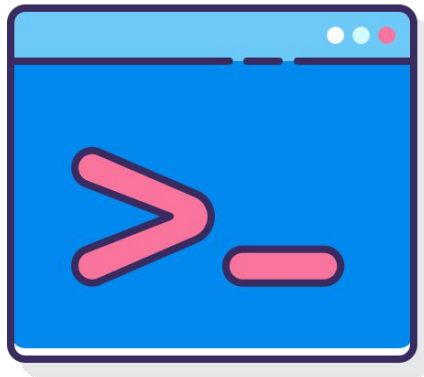
```
1 .button
2   background: cornflowerblue;
3   border-radius: 5px;
4   padding: 10px 20px;
5
6   &:hover
7     cursor: pointer;
8
9   &:disabled
10     cursor: default;
11     background: gray;
12     pointer-events: none;
```

# Flujo de trabajo Sass

Como mencionamos anteriormente, necesitamos compilar nuestro código de Sass a CSS. Esto podemos hacerlo de varias formas:

- **Desde la terminal o línea de comandos.**
- **Usando extensiones de nuestro editor de código.**
- Usando compiladores online.
- Entre otras.

Para efectos de esta clase, revisaremos sólo las dos primeras opciones.



# Utilización de Sass desde línea de comandos

Para compilar Sass debemos ejecutar un comando para que **nuestro archivo .scss se transforme en un archivo con extensión .css**, es decir: Generar un archivo style.css a partir de nuestro style.scss

Esto lo hacemos utilizando el comando **sass --watch** seguido de la ruta y el nombre del archivo SCSS (input o entrada), luego : para separar los archivos, y finalmente la ruta y el nombre del archivo CSS (output o saludo)

```
sass --watch style.scss:style.css
```



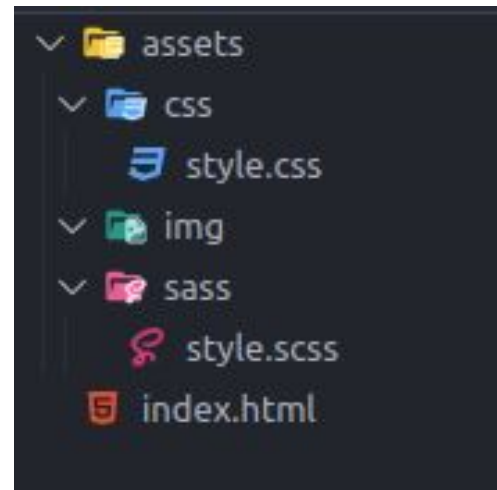
En este ejemplo sólo utilizamos el nombre de los archivos, ya que todos se encuentran en la misma ruta, pero esto puede variar dependiendo de la estructura de carpetas

# Utilización de Sass desde línea de comandos

## Ejemplo

Si consideramos esta estructura de carpetas, el comando para compilar quedaría de la siguiente manera:

```
sass --watch assets/sass/style.scss:assets/css/style.css
```



En el archivo index.html se debe referenciar su estilo al archivo .css

```
<link rel="stylesheet" href="assets/css/style.css">
```



# Utilización de Sass desde línea de comandos

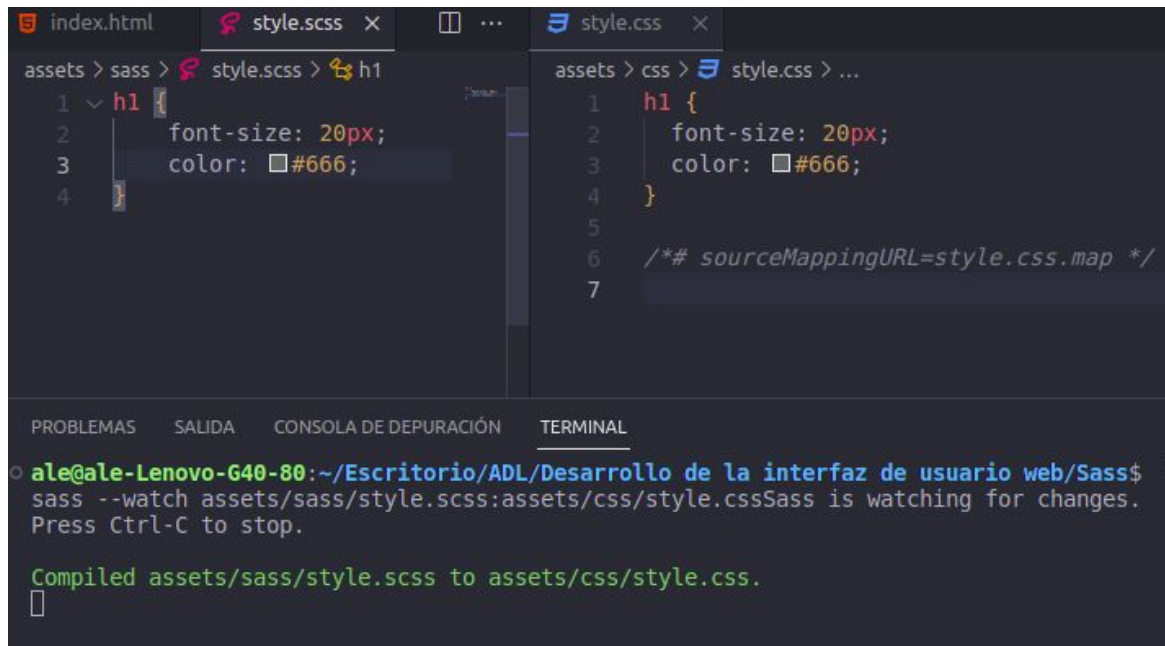
Si la compilación se ejecutó correctamente, veremos el siguiente mensaje

```
Sass is watching for changes. Press Ctrl-C to stop
```

Esto quiere decir que cualquier cambio que hagamos en nuestro archivo de entrada (.scss) se irá aplicando en nuestro archivo de salida (.css) al momento de guardar **ctrl + s**

# Utilización de Sass desde línea de comandos

Si nuestro código .scss no tiene errores, la terminal nos arrojaría un mensaje que fue compilado, y los estilos descritos se reflejan en el archivo .css.



The screenshot shows a code editor with two panels. The left panel displays the `style.scss` file with the following content:

```
1 h1 {
2   font-size: 20px;
3   color: #666;
4 }
```

The right panel displays the `style.css` file with the following content:

```
1 h1 {
2   font-size: 20px;
3   color: #666;
4 }
5
6 /*# sourceMappingURL=style.css.map */
7
```

At the bottom of the editor is a terminal window with the following output:

```
ale@ale-Lenovo-G40-80:~/Escritorio/ADL/Desarrollo de la interfaz de usuario web/Sass$
sass --watch assets/sass/style.scss:assets/css/style.css
Sass is watching for changes.
Press Ctrl-C to stop.

Compiled assets/sass/style.scss to assets/css/style.css.
```

# Utilización de Sass desde línea de comandos



Ya estamos listos para trabajar en cualquier proyecto usando Sass desde nuestra terminal, pero también podemos instalar un plugin o extensión para automatizar el proceso.

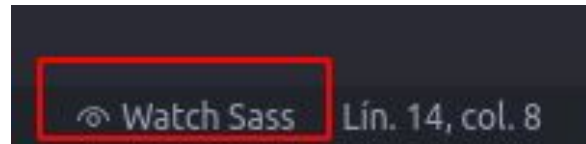
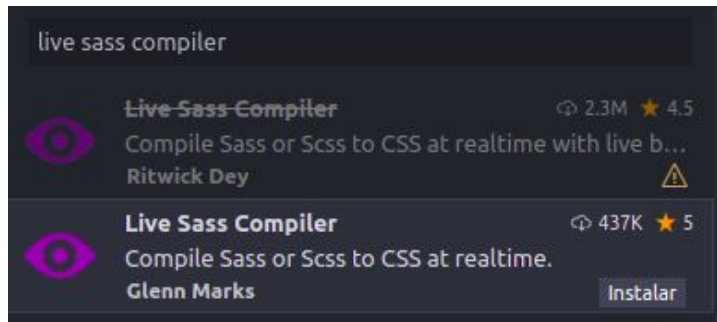
**/\* Instalación de plugins de Sass \*/**

# Instalación de plugins de Sass

## en editores de texto para automatización de tareas

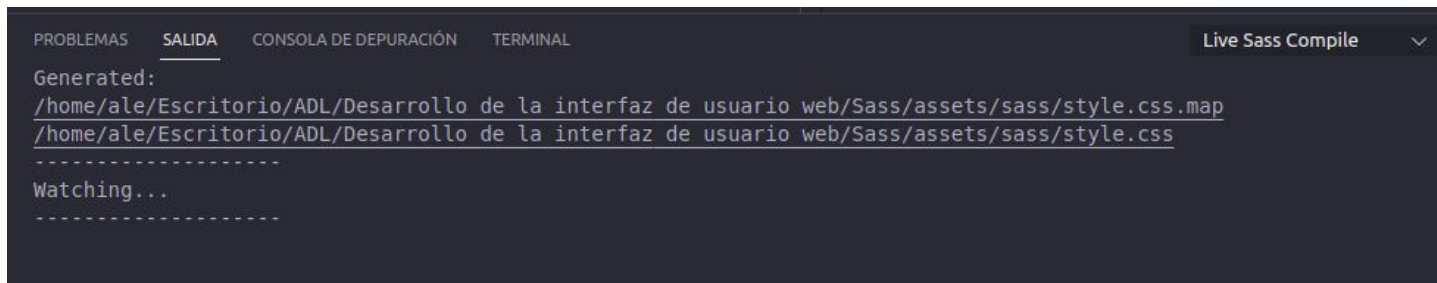
Para automatizar parte del proceso instalaremos una extensión de Sass llamada Live Sass Compiler, al buscarla tenemos que fijarnos en no utilizar la versión deprecada.

Al instalarla nos aparecerá esta opción en la barra de estado de Visual Studio Code:



# Live Sass Compiler

Si hacemos clic en Watch Sass de la barra de estado, nos abrirá una consola indicando lo siguiente:

A screenshot of a code editor's console window. The console has tabs for 'PROBLEMAS', 'SALIDA', 'CONSOLA DE DEPURACIÓN', and 'TERMINAL'. The 'SALIDA' tab is selected. In the top right corner of the console, there is a button labeled 'Live Sass Compile' with a dropdown arrow. The console output shows the following text: 'Generated:' followed by two file paths on separate lines, both underlined: '/home/ale/Escritorio/ADL/Desarrollo de la interfaz de usuario web/Sass/assets/sass/style.css.map' and '/home/ale/Escritorio/ADL/Desarrollo de la interfaz de usuario web/Sass/assets/sass/style.css'. Below these paths are two lines of dashes. Then, the text 'Watching...' appears, followed by another line of dashes.

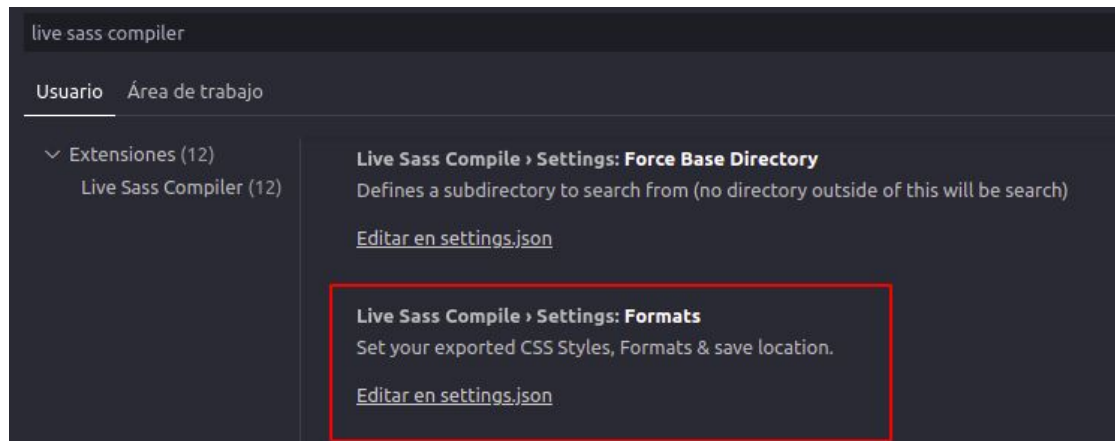
```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  Live Sass Compile v
Generated:
/home/ale/Escritorio/ADL/Desarrollo de la interfaz de usuario web/Sass/assets/sass/style.css.map
/home/ale/Escritorio/ADL/Desarrollo de la interfaz de usuario web/Sass/assets/sass/style.css
-----
Watching...
-----
```

Lo que hace esta extensión, es crear automáticamente un archivo .css **dentro de la misma ruta del archivo compilado**.

# Live Sass Compiler

Si queremos mantener la misma estructura de carpetas que vimos anteriormente, debemos **modificar la configuración de la extensión** para que no nos genere el archivo de salida en la misma ruta *assets/sass*, sino que lo cree en la ruta *assets/css* o en la ruta que queramos.

Debemos entrar a configuración, luego en el buscador escribir **live sass compiler** y ubicar el apartado **Formats**, y hacemos clic en **Editar en settings.json**.



# Modificando la ruta de Live Sass Compiler

Ahora que estamos en el archivo de configuración, modificaremos el valor de `savePath` por nuestra ruta. Si no existe esta carpeta en nuestro proyecto, la extensión la crea automáticamente, y en el caso de que el archivo `.css` ya existe, los estilos se aplicarán a este archivo. En este caso la configuración quedaría de la siguiente manera:

```
"liveSassCompile.settings.formats": [  
  {  
    "format": "expanded",  
    "extensionName": ".css",  
    "savePath": "assets/css",  
    "savePathReplacementPairs": null  
  }  
]
```



El nombre del archivo css de salida, será el mismo que el archivo `.scss` de entrada. En este caso nuestro archivo se llama `style.scss`, por lo que el archivo a crear en la carpeta `css` será `style.css`

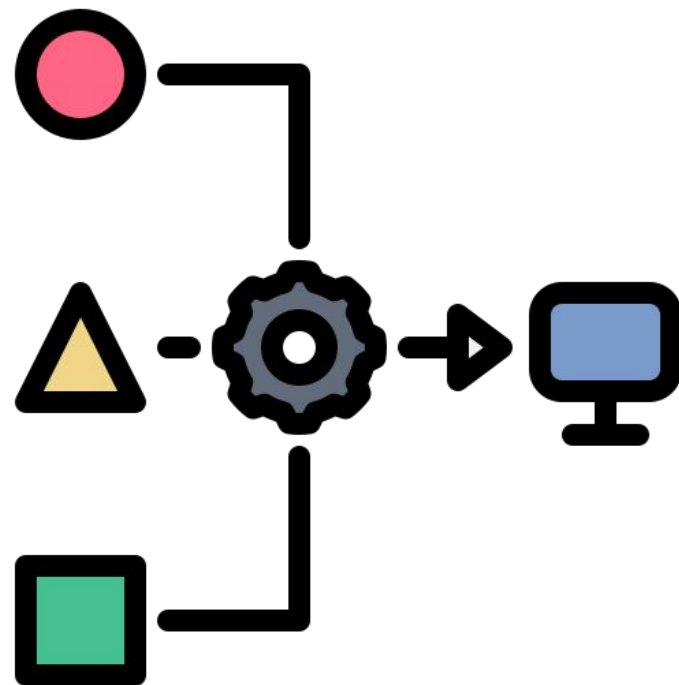


# Live Sass Compiler

Una vez configurado esto, sólo debemos hacer clic en **watch sass** en la barra de estado para comenzar a compilar automáticamente.

Si queremos detener o poner en “pausa” la compilación, presionamos la misma opción.

Con esto ya estamos preparados para trabajar en un proyecto de Sass.



# Ejercicio guiado - Login Blog FrontEnd



# Ejercicio guiado

## *Login Blog FrontEnd*

- Ahora crearemos la vista de login del Blog FrontEnd, en dónde deberán separar los archivos de estilos e incluirlos en cada carpeta que corresponde según el patrón 7-1.
- Se recomienda crear un nuevo proyecto en lugar de trabajar sobre el mismo proyecto, ya que podemos generar problemas de cascada.
- En sesiones siguientes llevaremos el blog completo a Sass y el patrón 7-1



# Ejercicio guiado

## Vista Login Blog FrontEnd

 **Blog FrontEnd**

Bienvenido

☐ Recordarme    ¿Olvidaste tu contraseña?

**Iniciar sesión**

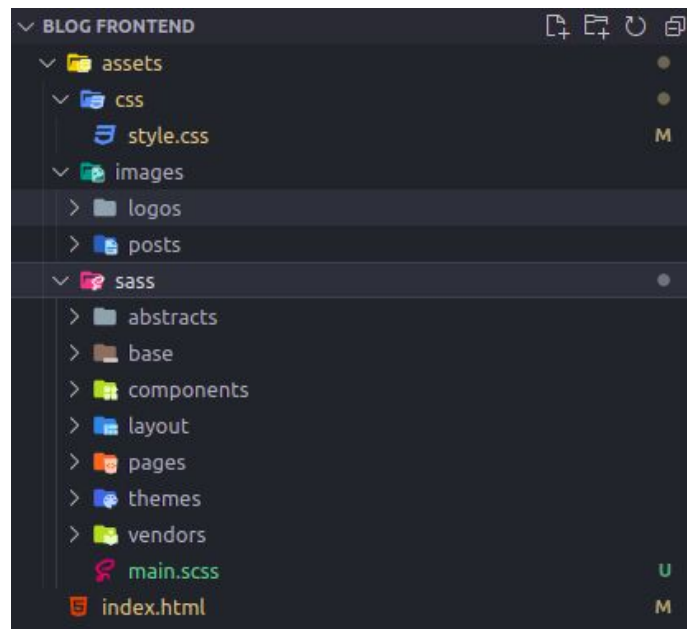
 **Inicia sesión con Google**

¿No tienes una cuenta? **Regístrate**

# Ejercicio guiado

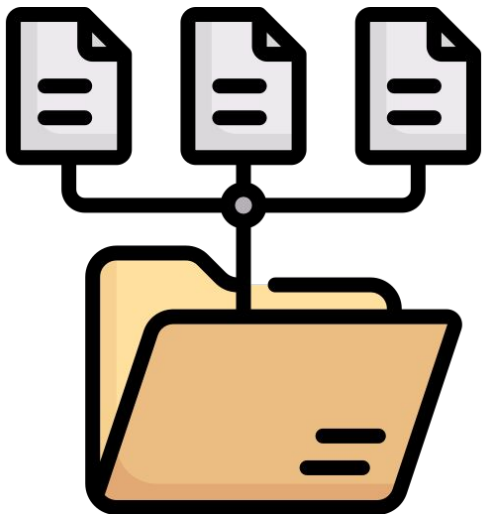
## Estructura de carpetas (Patrón 7-1)

Comenzaremos creando la estructura de carpetas propuesta por el patrón 7 - 1



## Ejercicio guiado

### Organizando parciales del login



Para organizar los archivos parciales que utilizaremos en nuestro proyecto es necesario identificar en qué directorio irá cada uno de los elementos que componen nuestra interfaz.

Primero revisemos qué elementos componen esta interfaz.



# Ejercicio guiado

## Layout

Lo primero es definir las partes que componen el diseño del sitio a maquetar.

Este sitio lo podemos separar de la siguiente manera:



**Blog FrontEnd**

La primera sección que podemos identificar es el header o cabecera del sitio.

# Ejercicio guiado

## Layout

La siguiente parte del diseño que podemos identificar es el formulario de inicio de sesión, y podemos agrupar todos estos elementos dentro de este layout.

Ahora que identificamos los elementos de la página a construir.

**¿Dónde integramos estos layout dentro del patrón 7-1?**

**{desafío}**  
**latam\_**

### Bienvenido

Email

Contraseña

☐ Recordarme      ¿Olvidaste tu contraseña?

Iniciar sesión

 Inicia sesión con Google

¿No tienes una cuenta? **Registrate**



# Ejercicio guiado

## Integrar layouts al patrón 7-1

Integrar estos layout dentro del directorio de Sass es fácil.

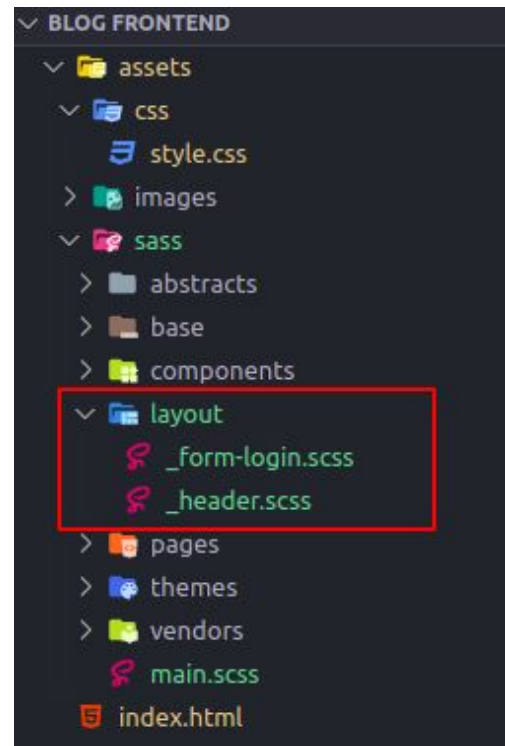
- Primero debemos ingresar a la carpeta especializada para estos estilos llamada **Layout** y luego crear dos archivos con nombres que representen su tipo.
- Con base en lo visto anteriormente, sus nombres deberán ser: `_header.scss`, y `_forms.scss`.
- Con el archivo `_form.scss` es mejor usar un nombre más específico, debido a que podemos integrar más formularios como el de registro o contacto, por lo tanto, un buen nombre para este archivo es `_form-login.scss`.



# Ejercicio guiado

## Integrar layouts al patrón 7-1

Ingresemos a la carpeta layout y creamos los archivos `_header.scss` y `_form-login.scss`.



# Ejercicio guiado

## Integrar Componentes al patrón 7-1

Ahora es el turno de separar los **componentes** que presenta la vista de login, en este encontraremos 2 clases de componentes:

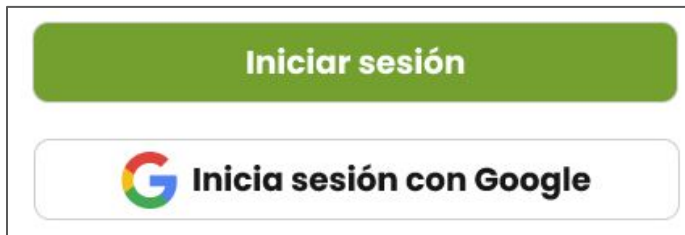
- Dos inputs
  - Email
  - Contraseña



Formulario de login con dos campos de entrada:

- Input de Email
- Input de Contraseña

- Dos botones
  - Iniciar sesión
  - Iniciar sesión con Google



Botones de login:

- Botón Iniciar sesión (verde)
- Botón Inicia sesión con Google (con logo de Google)

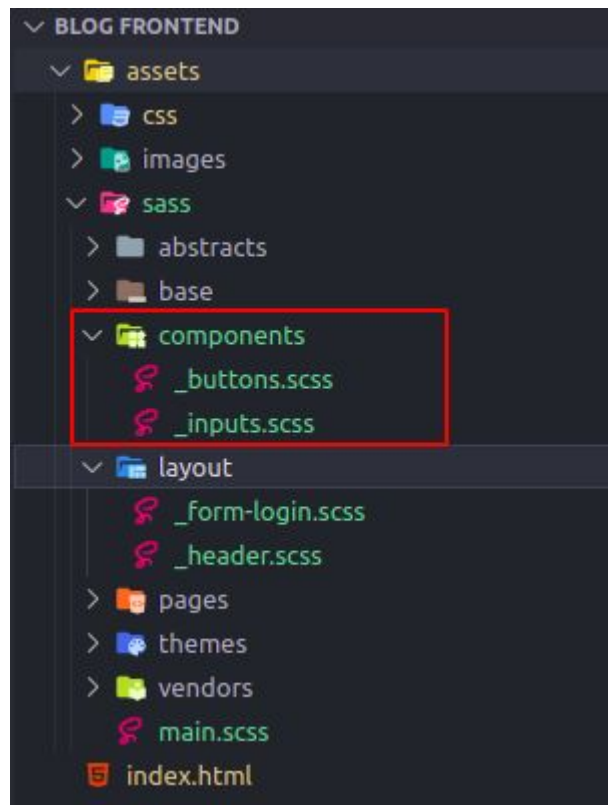


# Ejercicio guiado

## Integrar Componentes al patrón 7-1

Para integrar los componentes dentro del patrón 7-1 crearemos dos archivos dentro de la carpeta **components**:

- `_inputs.scss`
- `_buttons.scss`



# Ejercicio guiado

## Base

- Ahora que ya tenemos definido el diseño y los componentes que podremos utilizar en nuestro login, el siguiente paso es agregar los archivos que serán la base de nuestro nuestros estilos.
- Por el momento sólo utilizaremos el parcial `_typography` dentro del directorio base, aquí agregaremos las reglas de estilo relacionadas a la tipografía.



En las próximas sesiones migraremos la vista home y sus reglas de estilos a Sass y el patrón 7-1.



# Ejercicio guiado

## Orden de carga en el manifiesto

Ahora solo nos queda importar nuestros archivos creados recientemente dentro del manifiesto main.scss.

Pero esto se hace manteniendo un orden específico:

```
abstracts/  
vendors/  
base/  
layout/  
components/  
pages/  
themes/
```



# Ejercicio guiado

## Orden de carga en el manifiesto

Siguiendo el orden visto anteriormente, nuestro manifiesto quedaría de la siguiente manera:

```
1 // main.scss # Manifest File
2
3 // abstracts
4
5 // Vendors
6
7 // Base
8 @import 'base/typography';
9
10 // Layout
11 @import 'layout/header';
12 @import 'layout/form-login';
13
14 // Components
15 @import 'components/buttons';
16 @import 'components/inputs';
17
18 // Pages
19
20 // Theme
```



# Ejercicio guiado

## Compilar Sass

Probemos si no tenemos algún error en nuestra implementación ejecutando este comando en la terminal o usando la extensión Live Sass Compilar previamente configurada para mantener nuestra estructura de carpetas.

```
sass --watch assets/sass/main.scss:assets/css/style.css  
Sass is watching for changes. Press Ctrl-C to stop.
```





## Ejercicio guiado

¡Con esto terminamos de crear la estructura para nuestra vista y configurar parte de nuestro proyecto con Sass!

Ahora los invitamos a crear la interfaz del login en los archivos creados usando Sass.



¿Cómo nos beneficia el uso  
del patrón 7-1 en futuros  
proyectos?



# Resumen

A

El **patrón 7-1** separa los parciales en 7 directorios diferentes y un archivo que funciona como manifiesto.

B

El preprocesador Sass tiene **dos sintaxis diferentes**:

- sass
- scss

Recomendamos trabajar con SCSS, ya que es la más similar a Css.

C

Para **compilar nuestros archivos sass o scss** podemos hacerlo:

- Desde la terminal
- Usando extensiones
- Usando compiladores online

D

Si utilizamos la extensión **Live Sass Compiler** debemos configurar la ruta de salida de los archivos css para mantener siempre organizado nuestros proyectos.



## Próxima sesión...

- *Utiliza variables para la reutilización de código CSS.*

**{desafío}**  
**latam\_**

*Academia de  
talentos digitales*

