



IT Academy  
by KIBERNUM



# Sass: Organización y Modularización de Estilos

# Propósito de la clase

En esta sesión, exploraremos metodologías avanzadas para la organización y modularización de estilos en SASS. Aprenderás a estructurar y escribir código SASS de manera eficiente, escalable y mantenible, aprovechando técnicas como anidación, parciales, mixins, extends y el uso de variables. Al finalizar esta clase, serás capaz de:

- Comprender los conceptos fundamentales de SASS, incluyendo variables, anidación, mixins, extends y parciales.
- Diferenciar el uso de @extend, @mixin y @use, identificando cuándo y cómo aplicar cada uno en un proyecto real.
- Implementar un sistema de estilos escalable mediante SASS, organizando el código en parciales y estructurando los estilos de manera modular.
- Optimizar el código CSS reduciendo la redundancia mediante @extend, @mixin y @use, mejorando la mantenibilidad y la escalabilidad del proyecto.

# Herencia uso de @extends y @use



# 📌 Herencia en SASS

Para entender la herencia imagina que estás diseñando camisetas para un equipo de fútbol. Todas las camisetas tienen características comunes: **mangas cortas, cuello redondo y un logo del equipo**. Pero cada jugador puede personalizar la suya con su número y nombre.

En **SASS**, la **herencia** funciona igual. Te permite crear un estilo base (la camiseta común) y luego hacer que otros elementos (jugadores) hereden esos estilos, sin necesidad de copiarlos una y otra vez.



```
1 $primary-color: #bbbfc0;
2 $margin: 16px;
3
4 .content-navigation {
5   border-color: $primary-color;
6   color: darken($primary-color, 10%);
7 }
8
9 .border{
10   padding: $margin / 4;
11   margin: $margin / 2;
12   border-color: $primary-color;
13 }
14
15 $itemCount: 4;
16 @for $i from 1 through $itemCount {
17   #item-#{$i} {
18     background-color: red;
19     width: 50px * $i;
20     height: 120px / $i;
21   }
22 }
23
24 .error{
25   border: 1px #f00;
26   background: #fdd;
27 }
28
29 .error .intrusion{
30   font-size: 1.3em;
31   font-weight: bold;
32 }
```

# 📌 @extend Heredar Estilos sin Duplicar Código

## ✗ Problema sin @extend (Código Repetitivo)

Si tenemos varios botones con estilos similares, en **CSS normal** escribiríamos:

```
.btn {  
    display: inline-block;  
    padding: 10px 20px;  
    border-radius: 5px;  
    font-size: 16px;  
    text-align: center;  
}  
  
.btn-primary {  
    display: inline-block;  
    padding: 10px 20px;  
    border-radius: 5px;  
    font-size: 16px;  
    background: blue;  
    color: white;  
}  
  
.btn-secondary {  
    display: inline-block;  
    padding: 10px 20px;  
    border-radius: 5px;  
    font-size: 16px;  
    background: gray;  
    color: black;  
}
```

### ✓ Solución con @extend (Reutilizando Estilos)

```
.btn {  
    display: inline-block;  
    padding: 10px 20px;  
    border-radius: 5px;  
    font-size: 16px;  
    text-align: center;  
}  
  
.btn-primary {  
    @extend .btn;  
    background: blue;  
    color: white;  
}  
  
.btn-secondary {  
    @extend .btn;  
    background: gray;  
    color: black;  
}
```

### 📌 Se compila en CSS como:

```
.btn, .btn-primary, .btn-secondary {  
    display: inline-block;  
    padding: 10px 20px;  
    border-radius: 5px;  
    font-size: 16px;  
    text-align: center;  
}  
  
.btn-primary {  
    background: blue;  
    color: white;  
}  
  
.btn-secondary {  
    background: gray;  
    color: black;  
}
```

### 🎯 ¿Qué hizo @extend?

- **.btn-primary** y **.btn-secondary** heredaron los estilos de **.btn**.
- Ahora **.btn-primary** y **.btn-secondary** usan menos código.
- SASS automáticamente los combinó en un solo bloque de CSS.



En SASS, podemos **heredar los estilos** de **.btn** en **.btn-primary** y **.btn-secondary**

# ⚠ Cuándo NO usar @extend

## ✗ Cuando los estilos son muy diferentes.

Si dos clases tienen **pocos estilos en común**, usa **Mixins (@mixin)** en lugar de **@extend**.

- ◆ Ejemplo con Mixins en vez de **@extend**:

```
@mixin button($bg, $color) {  
  display: inline-block;  
  padding: 10px 20px;  
  border-radius: 5px;  
  font-size: 16px;  
  text-align: center;  
  background: $bg;  
  color: $color;  
}  
  
.btn-primary {  
  @include button(blue, white);  
}  
  
.btn-secondary {  
  @include button(gray, black);  
}
```



📌 Usa **@extend** para herencias completas, pero usa **@mixin** cuando quieras personalizar valores.

# @use: La Mejor Manera de Importar Archivos en SASS

## ✗ Problema con @import (Antigua Forma)

Antes usábamos @import para dividir el código en varios archivos:

### 📌 Problemas de @import:

- Carga múltiples veces los mismos archivos (Código duplicado).
- No permite modularidad real (puedes acceder a variables sin importar de dónde vienen).
- Hace más lento el compilado de SASS.

```
@import 'variables';
@import 'buttons';
@import 'header';
```

## ✓ Solución: Usar @use en Lugar de @import

@use es la nueva forma recomendada para importar archivos.

### 📌 Diferencias clave entre @use y @import:

Característica	@import (Viejo)	@use (Nuevo)
Múltiples cargas	✗ Puede importar el mismo archivo más de una vez	<input checked="" type="checkbox"/> Solo se importa una vez
Variables y mixins	✗ Todo es global	<input checked="" type="checkbox"/> Usa namespaces para evitar conflictos
Rendimiento	✗ Más lento	<input checked="" type="checkbox"/> Más rápido

# Cómo usar @use correctamente

- 1 Crea archivos parciales (sin @use todavía).

📌 \_variables.scss

```
$primary-color: blue;  
$secondary-color: gray;
```

📌 \_buttons.scss

```
@use 'variables' as v; // Importa 'variables' con un alias 'v'  
  
.btn {  
  background: v.$primary-color;  
  color: white;  
}
```

- 2 Importa todo en main.scss usando @use

📌 main.scss

```
@use 'variables' as v;  
@use 'buttons';
```

◆ Explicación:

- **@use variables as v;** → Importa variables.scss con el alias **v**.
- En **\_buttons.scss**, accedemos a variables como **v.\$primary-color**.
- No se contamina el código global (cada archivo usa solo lo que necesita).

🎯 ¿Por qué @use es mejor?

- ✓ Evita duplicaciones.
- ✓ Mantiene el código modular y limpio.
- ✓ Mejora el rendimiento.

# Patrón 7-1 en SASS



# ¿Qué es el patrón 7-1 en SASS?

El patrón 7-1 es una forma de organizar los archivos SASS en 7 carpetas y 1 archivo principal.

## 📁 Estructura del patrón 7-1

```
/scss
  ├── abstracts/      # Funciones, mixins, variables, placeholders
  ├── base/           # Reset, estilos globales, tipografías
  ├── components/     # Botones, tarjetas, formularios
  ├── layout/          # Grid, header, footer, sidebar
  ├── pages/          # Estilos específicos de cada página
  ├── themes/          # Variaciones de color o estilos temáticos
  ├── vendors/         # Librerías externas (Bootstrap, normalize.css)
  └── main.scss        # Archivo principal que importa todo
```

## 📌 Explicación de cada carpeta

Carpeta	Descripción
abstracts/	Variables, mixins y funciones reutilizables.
base/	Reset, tipografías, estilos generales.
components/	Botones, tarjetas, formularios.
layout/	Header, footer, grid, sidebar.
pages/	Estilos específicos para cada página (ej. home.scss, contacto.scss).
themes/	Temas de color (ej. modo oscuro, versión corporativa).
vendors/	Librerías externas como Bootstrap o Normalize.css.
main.scss	Archivo que importa todo lo demás.

# Ejercicio Patrón 7-1



# Ejercicio Implementación del Patrón 7-1 en Sass

A continuación, el facilitador creará un proyecto demostrando la implementación del patrón 7-1 en Sass, utilizando la sintaxis moderna `@use`. Este ejercicio permitirá comprender la organización modular de los archivos Sass y aplicar buenas prácticas en la estructuración del código.

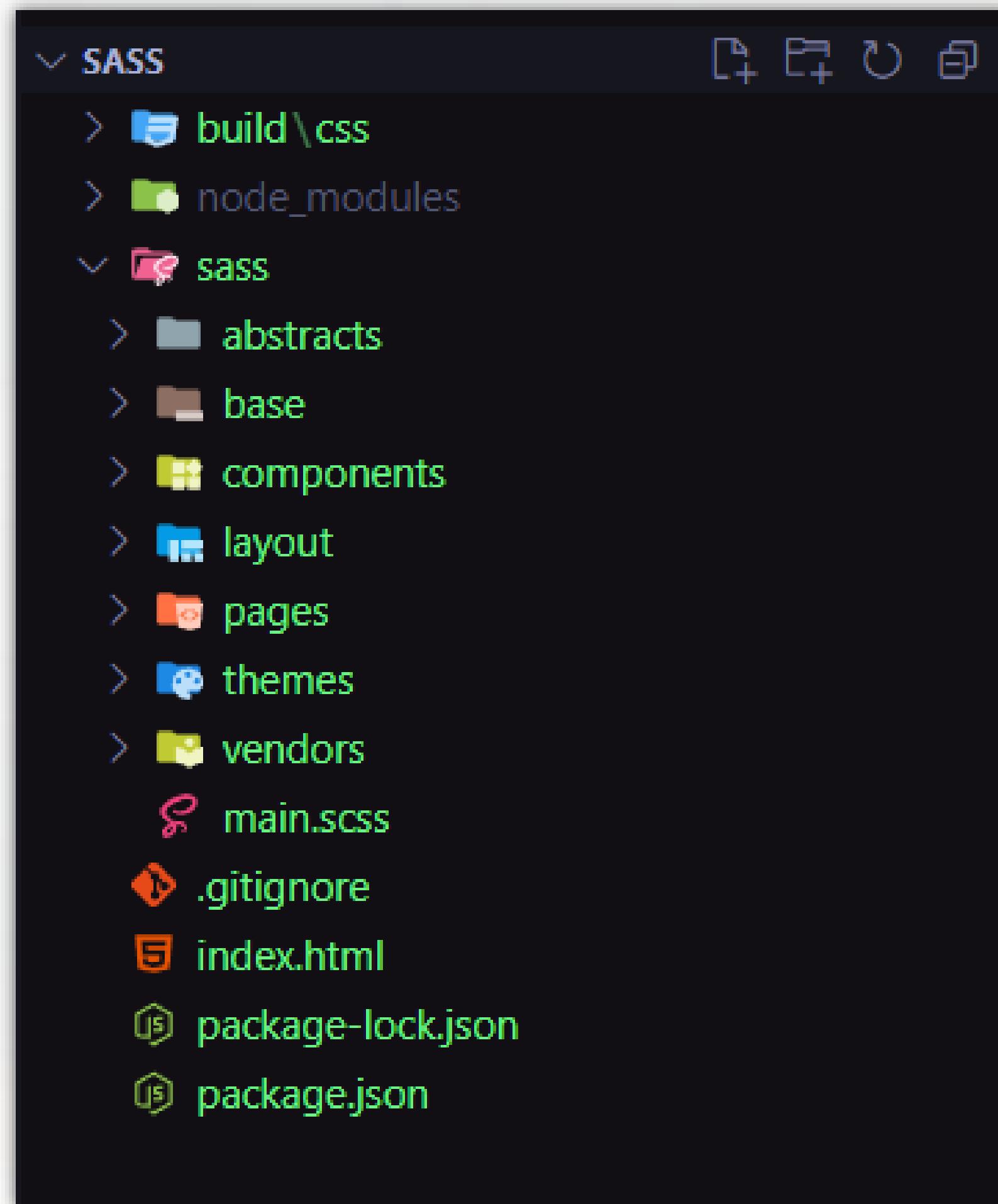
## Objetivo del Ejercicio

El facilitador guiará a los estudiantes en la creación de un proyecto de estilos donde:

- Se utilice Sass siguiendo la estructura del patrón 7-1.
- Se implementen variables para una paleta de colores pastel.
- Se utilicen mixins para reutilizar estilos en componentes.
- Se aplique Flexbox para un sticky footer que se mantenga siempre en la parte inferior de la página.
- Se integre una fuente de Google Fonts en la tipografía.
- Se compile Sass a CSS y se estructuren correctamente los archivos.

# Apariencia esperada del Proyecto

## 📌 Estructura del Proyecto

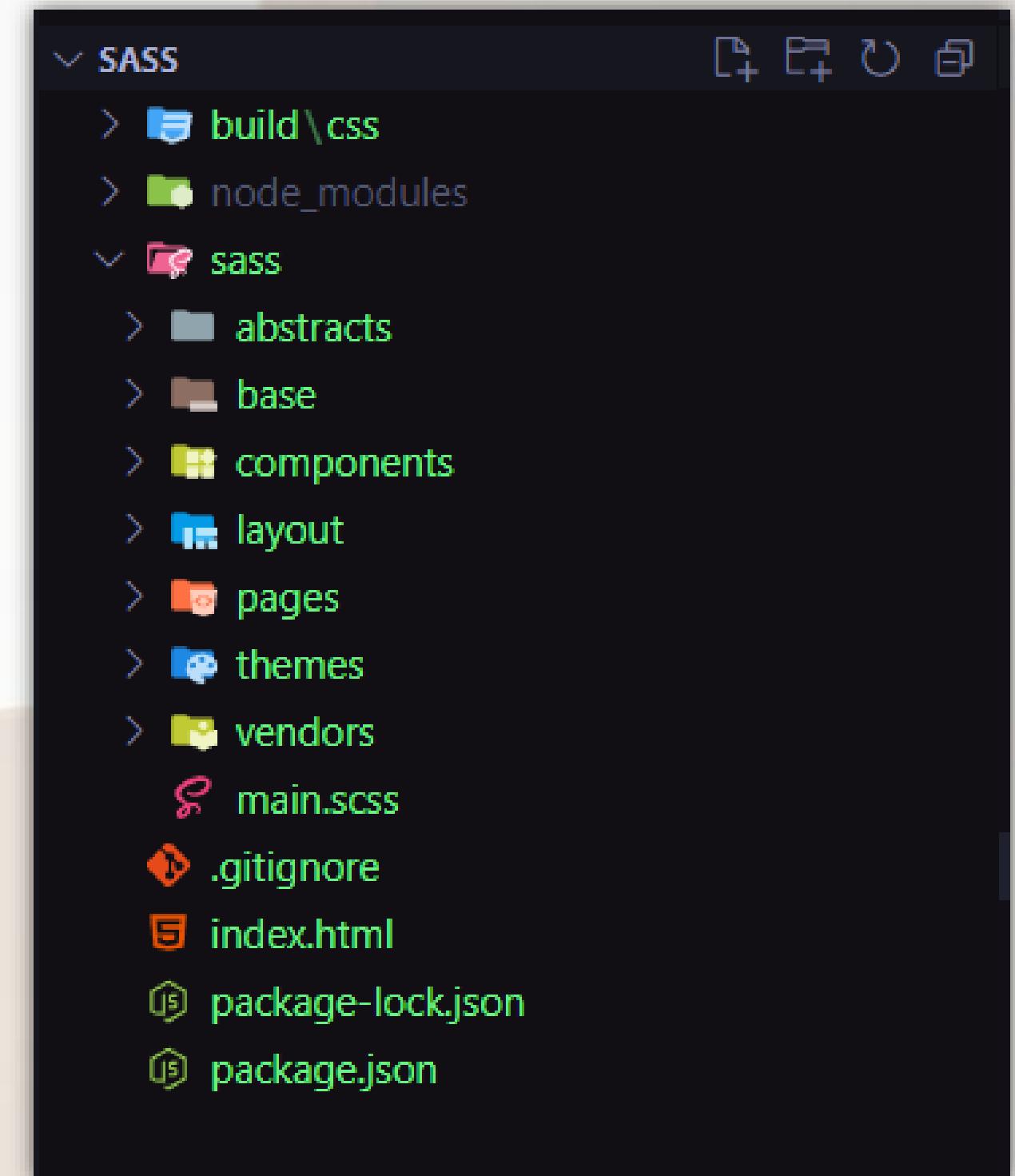


## 📌 Resultado del Proyecto

The screenshot shows a web browser window with the title 'Proyecto Sass 7-1 con Sticky Fo'. The URL in the address bar is '127.0.0.1:5500/index.html'. The page has a header with three navigation links: 'Inicio', 'Acerca', and 'Contacto'. Below the header is a green section containing the text 'Bienvenido a mi proyecto con SASS 7-1' and two buttons: 'Botón Principal' (blue) and 'Botón Secundario' (pink). The main content area is titled 'Acerca de Nosotros' and contains a card with the heading 'Título de la Tarjeta' and the text 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.' At the bottom of the page is a pink footer bar with the copyright notice '© 2025 Mi Proyecto Sass'.

# Explicación

- abstracts/: Archivos globales y reutilizables (`variables.scss`, `mixins.scss`, `functions.scss`).
- base/: Estilos iniciales y globales del proyecto (`reset.scss`, `typography.scss`).
- components/: Pequeños bloques reutilizables (botones, tarjetas, formularios, etc.).
- layout/: Estructura general de la página (header, footer, grid, etc.).
- pages/: Estilos específicos por cada página (home, about, contacto, etc.).
- themes/: Variaciones de tema o estilos de color (light, pastel, dark, corporate, etc.).
- vendors/: Librerías externas (normalize.css, Bootstrap, etc.).
- main.scss: Punto de entrada principal donde se incluyen los demás archivos con @use.



# Plugin para Sass con VsCode



# Plugin Live Sass Compiler: Automatización de la Compilación de Sass en VSCode

Para automatizar la compilación de archivos Sass/SCSS en Visual Studio Code (VSCode), puedes utilizar la extensión Live Sass Compiler. A continuación, veamos el paso a paso para instalar y utilizar esta herramienta:

## Instalación de Live Sass Compiler

Abre Visual Studio Code.

Accede al Marketplace de Extensiones:

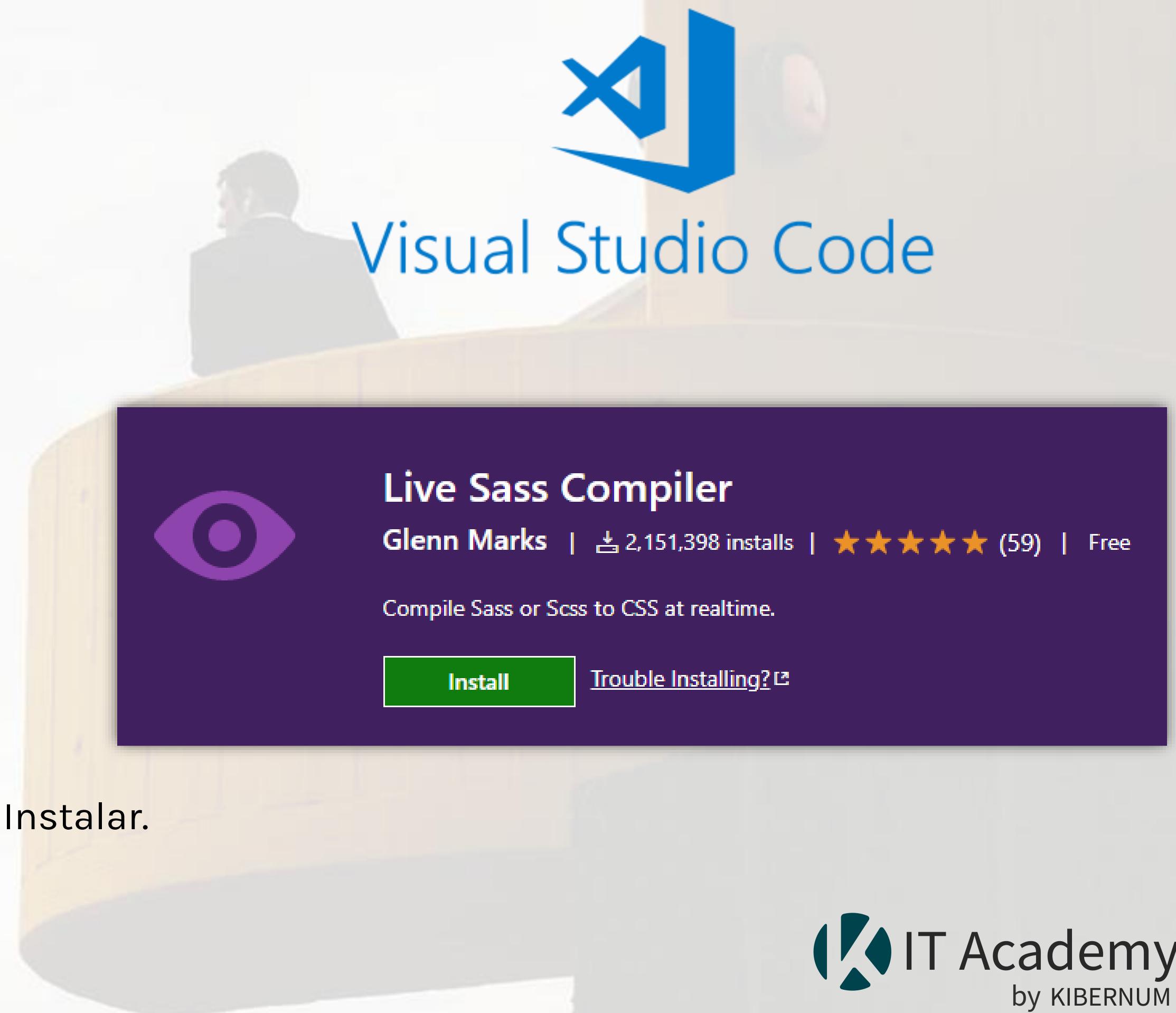
- Haz clic en el ícono de extensiones en la barra lateral izquierda o presiona Ctrl+Shift+X.

### Busca la extensión:

- Escribe Live Sass Compiler en la barra de búsqueda.

### Instala la extensión:

- Selecciona la extensión desarrollada por Ritwick Dey y haz clic en el botón Instalar.



# Uso de Live Sass Compiler

Abre tu proyecto:

- Asegúrate de tener tus archivos .scss o .sass listos en tu proyecto.

Inicia la compilación en tiempo real:

- Haz clic en el botón Watch Sass en la barra de estado en la parte inferior de VSCode.
- Alternativamente, presiona F1 o Ctrl+Shift+P, escribe Live Sass: Watch Sass y selecciona la opción.

Generación de archivos CSS:

- La extensión compilará automáticamente tus archivos Sass/SCSS y generará los archivos CSS correspondientes en tiempo real.

Para una guía visual y más detallada sobre cómo utilizar esta extensión, puedes consultar el siguiente video:

<https://www.youtube.com/watch?v=9ghnGxHS8ZA&t=33s>



# Utilizando Sass desde línea de comandos



# Utilizando Sass desde línea de comandos

Compilar un archivo Sass a CSS:

```
sass styles.scss styles.css
```

Compilación automática con watch:

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

Compilar una carpeta completa:

```
sass --watch scss/:css/
```

# Participemos Lluvia de Ideas



# Lluvia de Ideas sobre Sass y el Patrón 7-1

Hemos explorado diferentes conceptos clave en Sass y su uso en la organización de estilos, incluyendo el patrón 7-1 y la sintaxis moderna con @use. También hemos trabajado con variables, mixins, anidación, parciales, imports.

Ahora, queremos reflexionar sobre lo aprendido y compartir nuestras opiniones para reforzar los conceptos y descubrir qué técnicas nos parecen más útiles o interesantes en el desarrollo con Sass.

## Instrucciones:

- No hay respuestas incorrectas, la idea es generar discusión y reflexión.
- Comparte tu opinión y escucha la de tus compañeros para enriquecer el aprendizaje.

## 💡 Preguntas para la Lluvia de Ideas:

- ◆ ¿Qué concepto de Sass te pareció más claro o fácil de entender?
- ◆ ¿Cuál crees que será más útil en tus proyectos actuales o futuros?
- ◆ ¿Crees que el patrón 7-1 ayuda a mejorar la organización del código? ¿Por qué?
- ◆ ¿Te gustaría conocer cómo se usa Sass en proyectos grandes dentro de la industria?





# Conclusión: Reflexionando sobre Sass y el Patrón 7-1

Después de explorar y discutir Sass, @use y el patrón 7-1, hemos visto cómo estas herramientas nos ayudan a organizar y optimizar nuestros estilos de manera eficiente. No hay una única forma de trabajar con Sass, sino que cada técnica tiene sus ventajas y aplicaciones dependiendo del tipo de proyecto.

📌 Lo que hemos aprendido:

- ✓ El patrón 7-1 nos ayuda a estructurar y modularizar mejor nuestros archivos Sass.
- ✓ El uso de @use mejora la organización del código y evita problemas de globalización.
- ✓ Las variables y mixins permiten reutilizar código y mejorar la mantenibilidad del CSS.

📌 Reflexión Final:

- ◆ Cada estudiante tiene su propia experiencia y necesidades.
- ◆ Lo importante no es solo aprender la teoría, sino saber cuándo aplicar cada técnica en un proyecto real.
- ◆ Muchas empresas ya usan Sass con @use y estructuras como el patrón 7-1, por lo que conocer estas herramientas nos ayuda a mejorar como desarrolladores.

🚀 ¡Felicidades por este aprendizaje! Ahora pongámoslo en práctica. 🎯



## Pregunta:

¿Cómo crees que el uso de Sass y sus herramientas (variables, mixins, imports, patrón 7-1) pueden mejorar tu flujo de trabajo en la construcción de sitios web y qué estrategias podrías aplicar para optimizar su uso en proyectos reales?



# IT Academy

by KIBERNUM