**GUÍA DE DESARROLLO**

Detector de Señas LSC - Para Desarrolladores

**🤟 Detector de Señas LSC**

Lenguaje de Señas Colombiano

Versión: 1.0  
Fecha: 29/09/2025

# Estructura del Código

Esta guía está dirigida a desarrolladores que deseen:  
• Extender el sistema con nuevas señas  
• Modificar algoritmos de detección   
• Integrar con otros sistemas  
• Optimizar el rendimiento  
• Añadir nuevas funcionalidades

# Funciones Principales

## get\_finger\_states(landmarks)

Propósito: Determina el estado de cada dedo (extendido/doblado)

Parámetros: landmarks: Lista de 21 puntos MediaPipe

Retorna: Lista de 5 booleanos [pulgar, índice, medio, anular, meñique]

Algoritmo: Compara posiciones Y de articulaciones (X para pulgar)

## get\_finger\_directions(landmarks)

Propósito: Determina dirección de dedos (up/down/folded)

Parámetros: landmarks: Lista de 21 puntos MediaPipe

Retorna: Lista de 5 strings con direcciones

Algoritmo: Extiende get\_finger\_states con análisis direccional

## reconocer\_sena(landmarks)

Propósito: Reconoce qué seña se está realizando

Parámetros: landmarks: Lista de 21 puntos MediaPipe

Retorna: String con letra detectada o None

Algoritmo: Árbol de decisiones basado en estados y direcciones

# Cómo Agregar Nuevas Señas

Para agregar una nueva seña al sistema:  
  
1. Analizar la configuración de dedos requerida  
2. Determinar estados necesarios (extendido/doblado/curvado/dirección)  
3. Agregar lógica en la función reconocer\_sena()  
4. Probar con múltiples intentos para asegurar precisión  
5. Documentar la nueva seña en este manual  
  
Ejemplo - Agregar seña 'Z':

def reconocer\_sena(landmarks):  
 states = get\_finger\_states(landmarks)  
 directions = get\_finger\_directions(landmarks)  
   
 pulgar, indice, medio, anular, menique = states  
   
 # Seña Z - Solo índice extendido moviéndose  
 if (indice and not medio and not anular and not menique):  
 # Verificar movimiento adicional si es necesario  
 return 'Z'  
   
 # ... resto de señas existentes