



**Facultad Regional Rosario**  
Universidad Tecnológica Nacional

## **Trabajo práctico integrador:**

### **SignLearn**

**Grupo 14**

**Juego Interactivo para Aprender Lengua de Señas**

#### **Alumnos:**

Nombre y Apellido	Legajo
Pecoraro, Lucio	50239
Cordoba, Lucas	45091
Genoud, Franco	48992
Battistoni, Maria Paz	49641

**Fecha de presentación: 11-11-2025**

## 1. Narrativa del Proyecto

El aprendizaje de la Lengua de Señas Argentina (LSA) representa un desafío significativo, especialmente para los niños. Las herramientas de aprendizaje tradicionales, como libros o videos, carecen de la interactividad y la retroalimentación inmediata que son cruciales para dominar los gestos físicos. Esta falta de práctica dinámica puede llevar a la frustración y a una curva de aprendizaje lenta.

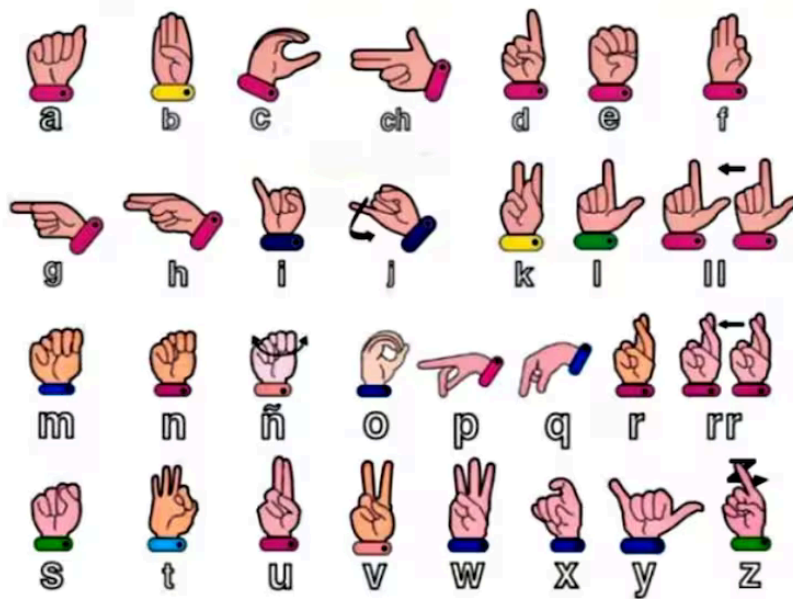
Sign Learn nace como una solución a este problema, proponiendo un puente entre la tecnología de visión por computadora y la educación inclusiva. El proyecto busca transformar el aprendizaje de la LSA en una experiencia lúdica y atractiva, eliminando las barreras de la práctica estática.

## 2. Abstract

SignLearn es un juego educativo interactivo programado en Python, diseñado para enseñar a niños el abecedario de la lengua de señas. El sistema utiliza la cámara web del usuario para capturar los gestos de la mano en tiempo real. Mediante el uso de librerías como OpenCV para el procesamiento de video y MediaPipe para el reconocimiento de landmarks de la mano, el juego analiza la seña realizada por el usuario y la compara con la seña objetivo.

El proyecto implementa una arquitectura de 3 capas (Presentación, Negocio y Datos) y sigue un modelo de dominio centrado en el Usuario y sus Sesiones de Juego. El sistema incorpora elementos de gamificación, como puntuaciones, combos y límites de tiempo, para motivar al jugador. La lógica de negocio valida la precisión de las señas y gestiona la progresión del juego, mientras que la capa de datos (diseñada para SQLite con SQLAlchemy) permite el almacenamiento del progreso y el desempeño histórico.

Las letras que se tienen que imitar por parte del jugador



### 3. Requerimientos Funcionales (Casos de Uso).

#### **CU01: Inicio de Sesión**

Permite que el usuario acceda al juego sin necesidad de registro.

#### **CU02: Iniciar Juego / Sesión**

El usuario inicia una nueva sesión de práctica. Se activa la cámara para el reconocimiento

#### **CU03: Detección y Evaluación de Señales**

El sistema utiliza OpenCV y MediaPipe

#### **CU04: Cálculo de Puntuación y Nivel**

El sistema calcula los puntos obtenidos según precisión y tiempo, y determina si el jugador avanza de nivel.

#### **CU05: Almacenamiento de Resultados**

Al finalizar la sesión, se guarda el nombre, el puntaje y la fecha de finalización en la base de datos.

#### **CU06: Visualización de Progreso**

El usuario puede consultar un resumen de sus sesiones pasadas.

## 4. Requerimientos No Funcionales

**Rendimiento:** El sistema debe procesar el video y realizar el reconocimiento de señas en tiempo real (mínimo 20 FPS) para no afectar la jugabilidad.

**Usabilidad:** La interfaz debe ser simple, intuitiva y visualmente atractiva para niños, con retroalimentación clara e inmediata (visual y auditiva).

**Portabilidad:** El juego debe poder ejecutarse en diferentes sistemas operativos (Windows, macOS, Linux) que soporten Python y las librerías requeridas.

## 5. Stack Tecnológico

- ☐ Flask
- ☐ OpenCV
- ☐ MediaPipe (Hands)
- ☐ Python 3.9+
- ☐ SQLite
- ☐ SQLAlchemy

## 6. Reglas de Negocio

### **RN01: Registro de puntajes válidos**

- Solo se podrán registrar puntajes cuando el jugador haya completado una partida.
- Cada registro debe incluir: nombre del jugador, puntaje obtenido y fecha y hora exacta de finalización.
- Los nombres se limitan a 30 caracteres y no pueden quedar vacíos.

### **RN02: Orden y visualización del ranking**

- El sistema debe mostrar los puntajes en orden descendente por puntaje y, en caso de empate, por fecha ascendente
- El usuario puede elegir la cantidad de resultados visibles (Top 10, 20 o 50) y filtrar por nombre.

### **RN03: Integridad del juego en ejecución**

- Solo puede haber una instancia activa del juego en un mismo dispositivo al mismo tiempo.

- Si el juego ya está en ejecución, el botón “Jugar” no debe lanzar una nueva partida hasta que la actual finalice o se detenga.

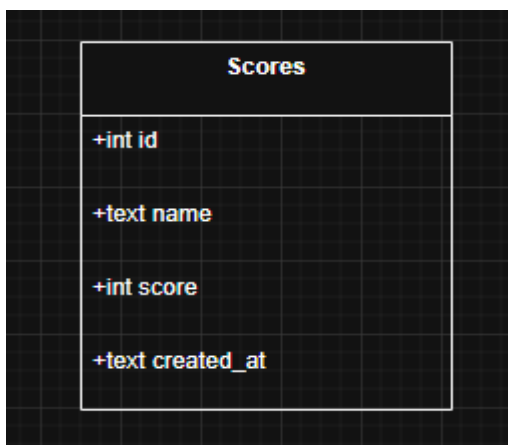
## 7. Modelo de Dominio

### Entidades Principales:

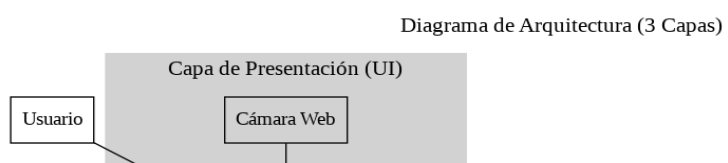
#### Scores:

- Registra cada sesión o ronda de práctica del usuario.
- Nombre
- Puntaje total obtenido.
- Fecha y hora de la sesión (created at).

### Diagrama de Clases (Modelo de Dominio)



### Diagrama de Arquitectura



## 10. Bibliografía y Documentación de Librerías

OpenCV: Open Source Computer Vision Library. Utilizada para la captura de video, procesamiento de imágenes y renderizado de la interfaz de usuario.

Documentación: <https://docs.opencv.org/>

MediaPipe: Framework de Google para la construcción de pipelines de percepción multimodal. Se utiliza específicamente la solución "Hands" para la detección de landmarks de manos.

Documentación: [https://developers.google.com/mediapipe/solutions/vision/hand\\_landmarker](https://developers.google.com/mediapipe/solutions/vision/hand_landmarker)

Python: Python es potente y rápido; se integra bien con otros lenguajes; funciona en cualquier lugar; es amigable y fácil de aprender; es de código abierto.

Su mayor fortaleza es El Índice de Paquetes de Python (PyPI - <https://pypi.org/>) ya que alberga miles de módulos de terceros para Python. Tanto la biblioteca estándar de Python como los módulos aportados por la comunidad ofrecen un sinfín de posibilidades.

Documentación: <https://docs.python.org/3/>

SQLite: SQLite es una biblioteca integrada que implementa un motor de base de datos SQL transaccional, autónomo, sin servidor y sin configuración. El código de SQLite es de dominio público, y por lo tanto, su uso es gratuito.

SQLite es un motor de base de datos SQL integrado. A diferencia de la mayoría de las bases de datos SQL, SQLite no requiere un proceso de servidor independiente. SQLite lee y escribe directamente en archivos de disco comunes. Una base de datos SQL completa, con múltiples tablas, índices, disparadores y vistas, se encuentra en un único archivo de disco.

El módulo sqlite3 de Python proporciona una interfaz estándar para interactuar con bases de datos SQLite, además se incluye en la biblioteca estándar de Python a partir de la versión 2.5, por lo que no es necesaria ninguna instalación adicional.

Documentación: <https://sqlite.org/docs.html> - <https://docs.python.org/es/3/library/sqlite3.html>

SQLAlchemy: SQLAlchemy es el kit de herramientas SQL de Python y un mapeador objeto-relacional que brinda a los desarrolladores de aplicaciones todo el poder y la flexibilidad de SQL.

Proporciona un conjunto completo de patrones de persistencia de nivel empresarial bien conocidos, diseñados para un acceso a bases de datos eficiente y de alto rendimiento, adaptados a un lenguaje de dominio simple y propio de Python.

Documentación: <https://docs.sqlalchemy.org/en/20/>

## 11. Link al Código Fuente

[https://github.com/lcordoba844/frro-python-2025-14/tree/master/TPI\\_nuestro](https://github.com/lcordoba844/frro-python-2025-14/tree/master/TPI_nuestro)