

Duoc UC



SignBridge

Conectando Mundos



Equipo de trabajo



Francisco Egenau (SM)

- Planificacion de sprints
- Coordinacion de equipo
- Gestion de entregables
- Documentacion de proyectos



Sebastián Medina (DEV)

- Desarrollo React Native / Wep App
- Implementacion de UI/UX
- Integracion de modelos IA
- Optimizacion de performance
- Testing



Matías Machuca (TL)

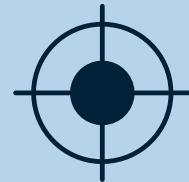
- Arquitectura modelo IA
- Entrenamiento de Modelos
- Optimización TensorFlow
- Testing de precisión
- Integración MediaPipe

Descripción del proyecto



Objetivos del proyecto

Objetivo General



Romper barreras de comunicación entre personas sordas y oyentes mediante una solución accesible, portable y gratuita.

Objetivos Específicos

1 Reconocer alfabeto, números y 31 frases comunes en lengua de señas chilenas

2 Implementar un modelo de machine learning optimizado para capturar señas

3 Escalar el diccionario de señas progresivamente para ampliar cobertura

Alcance y limitaciones del proyecto



Alcance

Cobertura funcional inicial:

- Reconocimiento del alfabeto completo (A-Z)
- Traducción de números (1-9)
- Vocabulario básico de 31 frases comunes

Compatibilidad y despliegue:

- Aplicación Web desplegable en navegador.

Características principales:

- Traducción en tiempo real de señas a texto
- Opción en salida en voz sintetizada
- Uso de MediaPipe y TensorFlow para clasificación



Limitaciones

Alcance inicial :

- Solo incluye alfabeto, números y 31 frases.
- No permite tener conversaciones fluidas.
- Dataset Limitado, puede afectar precisión en condiciones reales.

Dependencia técnicas:

- Requiere buena iluminación y cámara bien posicionada
- Diferencias en ejecución de señas pueden reducir exactitud

Restricciones de dependencias

- El versionado de las librerías, limitó el desarrollo mobile, en esta primera versión.

Metodología de trabajo



Enfoque general

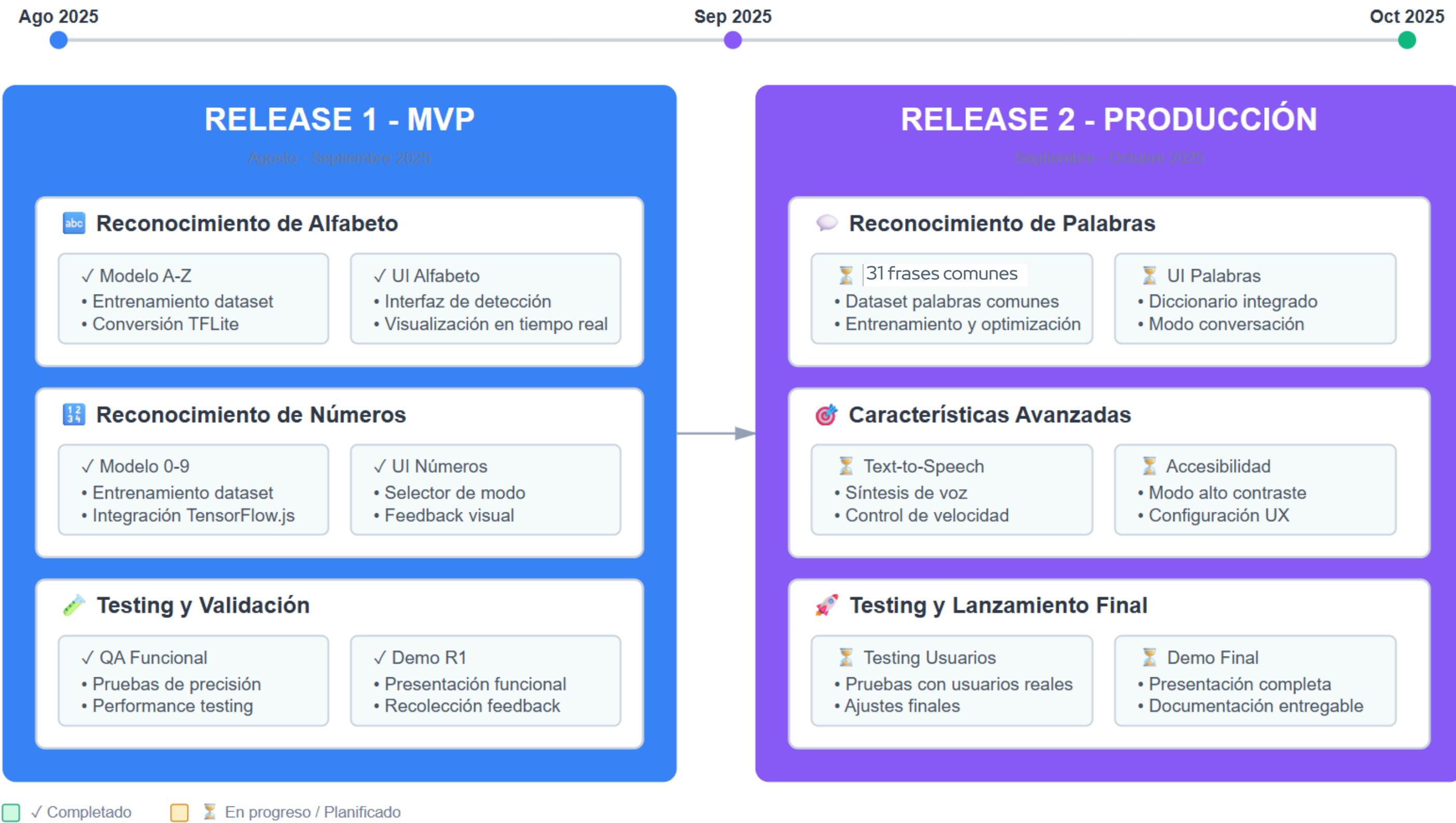
- Se aplicara una metodología ágil basada en scrum
- Cada sprint incluye:
Planing-Development-Review-Retro
- Daily/weekly, para coordinación y alinear avances



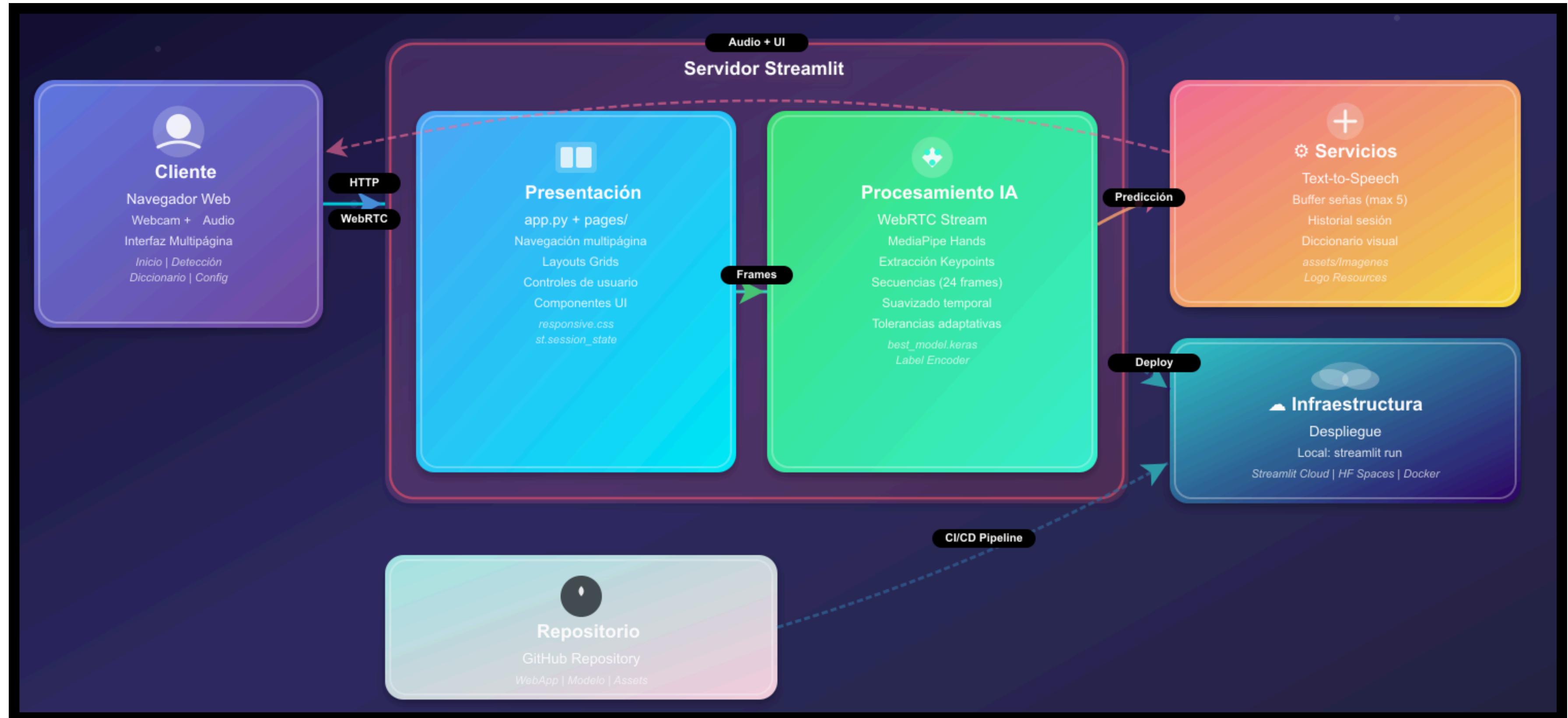
Herramientas de Apoyo

- **Gestión Ágil:** Jira
- **Comunicacion :** Discord / WhatsApp
- **Control de Versiones:** GitHub

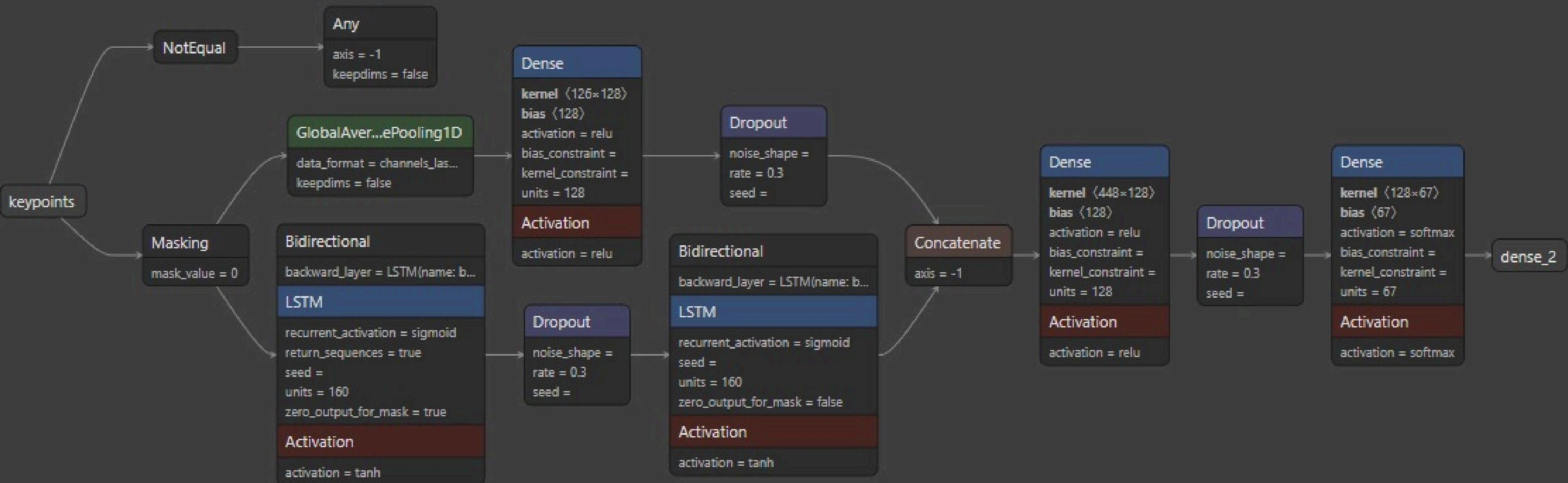
Released Plan de proyecto



Arquitectura del software



Flujo de datos



Tecnologías utilizadas

Backend & ML



Python

v3.10

Lenguaje principal para todo el backend y training del modelo



TensorFlow/Keras

2.13+

Framework de deep learning para LSTM bidireccional (92.8% accuracy)



MediaPipe

0.10+

Extracción de landmarks de manos en tiempo real



NumPy

1.24+

Operaciones numéricas y manipulación de arrays



scikit-learn

Latest

Preprocesamiento de datos y normalización

Tecnologías utilizadas

Frontend & Web



Streamlit

1.28+

Framework web para interfaz de demostración rápida



streamlit-webrtc

0.47+

Streaming de video en tiempo real desde navegador



OpenCV

4.8+

Procesamiento avanzado de imágenes y frames



av (PyAV)

10.0+

Manejo de codecs y procesamiento de audio/video

Herramientas & DevOps



Git

Control de versiones y colaboración



VS Code

Editor de código principal con extensiones para Python



PowerShell

Scripts de automatización y configuración



YAML

Archivos de configuración y orquestación

Demostración del resultado del proyecto

Resultados obtenidos

Rendimiento del Modelo

EXACTITUD FINAL

97.42% Accuracy

PÉRDIDA DE ENTRENAMIENTO

0.0983 Loss

LATENCIA EN TIEMPO REAL

<30ms MediaPipe + LSTM

Casos de Uso Exitosos

Detección de Números

Precisión casi perfecta en reconocimiento numérico

Saludos Básicos

"Hola", "Gracias", "Adiós" detectados correctamente

Preguntas

"¿Por qué?", "¿Quién?", "¿Cómo?" funcionan correctamente

Direcciones Espaciales

Señas de ubicación detectadas con precisión

Obstáculos presentados durante el desarrollo

1

Conversión a TFLite

Problema:

Pérdida severa: 97.3% → 60% accuracy (-37%)

Causa:

Cuantización float32 → int8 degradaba modelo híbrido

Solución:

Pivatar a aplicación web con Streamlit manteniendo Keras original

✓ Revertir a modelo Keras

2

Desbalance Estático-Dinámico

Problema:

Señas estáticas: 85% vs Dinámicas: 95%

Causa:

LSTM capturaba bien movimiento pero no poses fijas

Solución:

Arquitectura híbrida (LSTM + MLP) + data augmentation 5x

✓ 85% → 98% en estáticas

3

Predicciones Inestables

Problema:

Flickering de predicciones entre frames

Causa:

Confianza variable sin estabilización

Solución:

Ventana deslizante 8 frames + votación mayoritaria + cooldown

✓ Predicciones estables

4

Pérdida de Manos

Problema:

Predicción se reseteaba al ocultar manos brevemente

Causa:

Reset inmediato de buffer sin tolerancia

Solución:

Tolerancia 5 frames + reutilización keypoints + reset a 30 frames

✓ Detección robusta

Conclusiones

Logros Principales

- ✓ Sistema Funcional
Aplicación web completa con detección en tiempo real
- ✓ Alta Precisión
97.3% accuracy en validación
- ✓ Interfaz Completa
4 páginas: Inicio, Detección, Diccionario, Configuración
- ✓ Robustez
Sistema tolerante a occlusiones y condiciones variables
- ✓ Documentación
README, guías y walkthrough completos

Aprendizajes Clave

- 1 Flexibilidad en la Planificación
El cambio de Android a Web fue crucial para el éxito del proyecto
- 2 Importancia de la Arquitectura
El diseño híbrido permitió detectar ambos tipos de señas efectivamente
- 3 Data Augmentation
Fundamental con datasets pequeños para evitar overfitting
- 4 Experiencia de Usuario
Suavizado y tolerancia mejoran significativamente la usabilidad

Trabajo Futuro

- Mejoras Técnicas
 - Expandir dataset > 100 muestras/clase
 - Agregar 150+ señas nuevas
 - Detección de expresiones faciales
 - Múltiples usuarios simultáneos
- Mejoras de Producto
 - Imágenes reales en diccionario
 - Sistema de aprendizaje (feedback)
 - Soporte multi-idioma
- Despliegue
 - Publicar en Streamlit Cloud
 - Versión Docker
 - Optimización responsive

Preguntas de la comision

Duoc UC



Muchas Gracias