

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERIA**

**Escuela Profesional de Ingeniería de Sistemas**

**Proyecto de desarrollo de una aplicación móvil para la enseñanza del lenguaje de señas y su traducción automática en tiempo real**

Curso: SI – 983 Construcción de Software I

Docente: Mag. Alberto Johnatan Flor Rodríguez

Integrantes:

***Espinoza Caso, Lisbeth Isabel (código universitario)***

***Quispe Levano, Cristian Aldair (2018000590)***

**Tacna – Perú**

***2025***

Desarrollo de una aplicación móvil para la enseñanza del lenguaje de señas y su traducción automática en tiempo real

Documento de Arquitectura de Software

Versión *1.0*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CONTROL DE VERSIONES | | | | | |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | LEC | LEC | LEC | 20/08/2025 | Versión Original |

**INDICE GENERAL**

1. Introducción [4](#_heading=h.iz5jwp4xhi6h)

**1.1 Propósito [4](#_heading=h.y78o7zq0kpff)**

**1.2 Alcance [4](#_heading=h.h6d5nnsrve2u)**

**1.3 Definición, siglas y abreviaturas [4](#_heading=h.mk9fi98aiku0)**

**1.4 Referencias [4](#_heading=h.4l1z32i8gj6j)**

**1.5 Visión General [5](#_heading=h.xwqq3wtrpsuz)**

2. Representación Arquitectónica [5](#_heading=h.ynht39ac7grb)

**2.1 Escenarios [5](#_heading=h.d9bq1f87onyk)**

**2.2 Vista Lógica [6](#_heading=h.59ujnnirf1hz)**

**2.3 Vista del Proceso [6](#_heading=h.ubil7pyzacni)**

**2.4 Vista del desarrollo [6](#_heading=h.aobz8d6328gr)**

**2.5 Vista Física [6](#_heading=h.3a7vilc2xsrq)**

3. Objetivos y limitaciones arquitectónicas [7](#_heading=h.41bmx5232e3o)

**3.1 Disponibilidad [7](#_heading=h.5g9h4yzdkluo)**

**3.2 Seguridad [7](#_heading=h.fxd3xllguwv5)**

**3.3 Adaptabilidad [7](#_heading=h.c12ca9m9a34p)**

**3.4 Rendimiento [7](#_heading=h.ed27r8rc9qky)**

4. Análisis de Requerimientos [8](#_heading=h.9ruqms93q4i4)

**4.1 Requerimientos funcionales [7](#_heading=h.5g9h4yzdkluo)**

**4.2 Requerimientos no funcionales [7](#_heading=h.fxd3xllguwv5)**

5. Vistas de Caso de Uso [9](#_heading=h.w2i34xk4vn65)

6. Vista Lógica [17](#_heading=h.aqpj3ct2nxn)

**6.1 Diagrama Contextual [17](#_heading=h.wqzn09y4qzwo)**

7. Vista de Procesos [18](#_heading=h.cndjm24yio7x)

**7.1 Diagrama de Proceso Actual [18](#_heading=h.6cqx1ods3dz6)**

**7.2 Diagrama de Proceso Propuesto [18](#_heading=h.vznordnlry7j)**

8. Vista de Despliegue [19](#_heading=h.vb2c45ygkgv5)

**8.1 Diagrama de Contenedor [19](#_heading=h.ri1305z2ve8e)**

9. Vista de Implementación [20](#_heading=h.5h2pro5nf097)

**9.1 Diagrama de Componentes [20](#_heading=h.tekcnq6ody)**

10. Vista de Datos [22](#_heading=h.v3fww7ae8xlf)

**10.1 Diagrama Entidad Relación [22](#_heading=h.upf6yoktjqlk)**

11. Calidad [24](#_heading=h.yb7nm41hojzg)

**11.1 Escenario de Seguridad 25**

**11.2 Escenario de Usabilidad 26**

**11.3 Escenario de Adaptabilidad 27**

**11.4 Escenario de Disponibilidad 28**

**11.5 Otro Escenario 28**

**Documento de Arquitectura de Software**

**1. Introducción**

**1.1 Propósito**

*Este documento describe la arquitectura del sistema SignaPerú, una aplicación móvil inclusiva que traduce señas en tiempo real y ofrece módulos educativos para personas sordas. El propósito es definir las vistas arquitectónicas, componentes, procesos y criterios de calidad que guían su desarrollo.*

**1.2 Alcance**

*SignaPerú está orientado a usuarios sordos, docentes y validadores lingüísticos. La app permite traducción bidireccional entre señas y texto/voz, validación colaborativa de nuevas señas, y seguimiento educativo. Se implementa en Flutter, con backend en Node.js y procesamiento IA en Python.*

**1.3 Definición, siglas y abreviaturas**

* *IA: Inteligencia Artificial*
* *TTS: Text-to-Speech*
* *STT: Speech-to-Text*
* *API: Application Programming Interface*
* *gRPC: Google Remote Procedure Call*
* *WCAG: Web Content Accessibility Guidelines*
* *CU: Caso de Uso*
* *DAS: Documento de Arquitectura de Software*

**1.4 Referencias**

* IEEE 1471: Recommended Practice for Architectural Description
* WCAG 2.1 – Accesibilidad Web
* Flutter Documentation
* TensorFlow Lite API
* Firebase Authentication & Firestore

**1.5 Visión General**

*SignaPerú es una aplicación móvil inclusiva diseñada para facilitar la comunicación entre personas sordas y oyentes mediante la traducción automática de señas peruanas a texto o voz, y viceversa. Además, incorpora un módulo educativo que permite el aprendizaje estructurado de señas, el seguimiento del progreso de los estudiantes, y la validación colaborativa de nuevas señas por parte de expertos lingüísticos.*

*El sistema se apoya en tecnologías de inteligencia artificial, visión por computadora, procesamiento de lenguaje natural y arquitectura modular para garantizar escalabilidad, accesibilidad y trazabilidad. Está orientado a impactar positivamente en la inclusión social, la educación bilingüe y la preservación de la lengua de señas peruana.*

***Objetivos Clave***

* *Traducir señas en tiempo real usando IA entrenada con datasets locales.*
* *Permitir traducción inversa de texto/voz a señas animadas.*
* *Ofrecer un módulo educativo con lecciones, evaluaciones y seguimiento.*
* *Validar nuevas señas propuestas por la comunidad.*
* *Gestionar usuarios con roles diferenciados (estudiante, docente, validador).*
* *Garantizar accesibilidad conforme a WCAG 2.1.*

***Arquitectura General***

* ***Frontend móvil:*** *desarrollado en Flutter, con interfaces accesibles y captura de cámara.*
* ***Backend cloud:*** *construido en Node.js/Firebase, maneja autenticación, lógica de negocio y persistencia.*
* ***Servidor IA:*** *implementado en Python con TensorFlow, realiza inferencias de señas y conversión semántica.*
* ***Panel web:*** *para docentes y validadores, permite supervisar progreso y validar contenido.*

***Público objetivo***

* *Personas sordas y oyentes en entornos educativos, familiares y laborales.*
* *Docentes especializados en educación inclusiva.*
* *Validadores lingüísticos y expertos en lengua de señas peruana.*

***Impacto esperado***

* *Reducción de barreras comunicativas en tiempo real.*
* *Mejora del acceso a contenidos educativos en lengua de señas.*
* *Fortalecimiento de la comunidad sorda mediante participación activa en la validación del lenguaje.*
* *Generación de datos trazables para investigación lingüística y social.*

**2. Representación Arquitectónica**

**2.1 Escenarios**

*Los siguientes escenarios representan los principales flujos funcionales que el sistema debe soportar:*

*🔹* ***ESCENARIO 1: TRADUCCIÓN DE SEÑAS EN TIEMPO REAL***

***Actor:*** *Usuario sordo*

***Flujo:***

* *El usuario activa la cámara desde la app móvil.*
* *Se capturan los gestos en video.*
* *El motor de IA procesa los frames y devuelve el texto o voz correspondiente.*
* *El resultado se muestra en pantalla y se reproduce por voz si está habilitado.*

*🔹* ***ESCENARIO 2: TRADUCCIÓN DE TEXTO O VOZ A SEÑAS***

***Actor:*** *Usuario oyente*

***Flujo:***

* *El usuario ingresa texto o habla al micrófono.*
* *El sistema convierte la entrada en señas animadas.*
* *Se muestra la seña correspondiente en pantalla.*

***🔹 ESCENARIO 3: VALIDACIÓN DE NUEVAS SEÑAS***

***Actor:*** *Docente*

***Flujo:***

* *Se recibe una propuesta de seña por parte de la comunidad.*
* *El validador revisa el contenido, agrega comentarios y aprueba o rechaza.*
* *La seña validada se integra al sistema.*

*🔹* ***ESCENARIO 4: ACCESO Y SEGUIMIENTO EDUCATIVO***

***Actor:*** *Estudiante y docente*

***Flujo:***

* *El estudiante accede a lecciones interactivas.*
* *Realiza prácticas y evaluaciones.*
* *El sistema registra el progreso.*
* *El docente supervisa el avance desde un panel web.*

**2.2 Vista Lógica**

*La vista lógica representa la estructura del sistema desde el punto de vista de clases y objetos:*

* ***Entidades clave:*** *Usuario, Seña, Traducción, Lección, Progreso, Comentario, Notificación*
* ***Controladores:*** *Autenticador, MotorIA, Validador, GestorProgreso, Traductor*
* ***Interfaces:*** *FrmLogin, FrmTraductor, FrmResultado, FrmValidador, ModuloEducativo*

*Esta vista se modela mediante diagramas de clases y análisis de objetos, ya generados.*

**2.3 Vista del Proceso**

*Describe cómo se ejecutan los procesos en tiempo real:*

***Procesos asincrónicos:***

* *Captura de video desde cámara*
* *Inferencia IA en servidor Python*
* *Comunicación gRPC entre backend y servidor IA*
* *Notificaciones en tiempo real vía WebSocket*

***Procesos síncronos:***

* *Autenticación de usuario*
* *Consulta de progreso educativo*
* *Validación de señas*

**2.4 Vista del desarrollo**

*Describe cómo está organizado el código fuente:*

***Frontend (Flutter):***

* *lib/screens/: interfaces por módulo*
* *lib/models/: clases de datos*
* *lib/services/: lógica de negocio y API*
* *lib/providers/: gestión de estado*

***Backend (Python + Node.js):***

* *model.py, evaluate\_model.py, process\_video.py: IA*
* *server.py: API REST*
* *convert\_to\_tflite.py: exportación de modelos*
* *text\_to\_speech.py: salida vocal*

**2.5 Vista Física**

*Describe cómo se despliega el sistema en infraestructura real:*

***Dispositivo móvil:***

* *App Flutter instalada en Android/iOS*
* *Acceso a cámara, micrófono y almacenamiento local*

***Servidor en la nube (Firebase / Node.js):***

* *Autenticación, base de datos, API REST*
* *Comunicación con servidor IA vía gRPC*

***Servidor de IA (Jetson Nano / Cloud GPU):***

* *Motor de inferencia con TensorFlow*
* *Procesamiento de video y generación de señas*

**3. Objetivos y limitaciones arquitectónicas**

*Esta sección define los atributos de calidad que la arquitectura debe cumplir, así como las restricciones que condicionan su diseño. Se abordan los pilares clave: disponibilidad, seguridad, adaptabilidad y rendimiento.*

**3.1 Disponibilidad**

***Objetivo:***

*Garantizar que el sistema esté disponible para los usuarios en todo momento, especialmente durante sesiones educativas o traducciones en tiempo real.*

***Estrategias arquitectónicas:***

* *Despliegue en Firebase Hosting con alta disponibilidad.*
* *Persistencia local en la app móvil para módulos educativos.*
* *Uso de WebSocket para mantener sesiones activas sin reconexión constante.*
* *Monitoreo de servicios y alertas ante fallos.*

***Limitaciones:***

* *Dependencia de conexión a internet para traducción en tiempo real.*
* *El servidor IA puede requerir reinicio manual si se ejecuta en hardware local (Jetson Nano).*

**3.2 Seguridad**

***Objetivo:***

*Proteger los datos personales, credenciales y contenidos educativos, garantizando autenticación segura y control de acceso por roles.*

***Estrategias arquitectónicas:***

* *Autenticación con Firebase Auth y gestión de roles (estudiante, docente, validador).*
* *Cifrado de contraseñas y tokens.*
* *Validación de entrada en formularios y endpoints.*
* *Trazabilidad en procesos de validación de señas.*

***Limitaciones:***

* *La validación colaborativa requiere balance entre trazabilidad y privacidad.*
* *WebSocket exige protección contra secuestro de sesión.*

**3.3 Adaptabilidad**

***Objetivo:***

*Permitir que el sistema evolucione fácilmente ante cambios en el lenguaje de señas, incorporación de nuevos módulos o expansión regional.*

***Estrategias arquitectónicas:***

* *Diseño modular con componentes desacoplados (MotorIA, ModuloEducativo, Validador).*
* *Uso de interfaces gRPC definidas en .proto, fácilmente extensibles.*
* *Separación entre lógica de negocio y presentación (MVC).*
* *Base de datos flexible para nuevos tipos de señas y contenidos.*

***Limitaciones:***

* *La animación de señas en 3D puede requerir integración con motores gráficos externos.*
* *La regionalización lingüística exige curaduría experta y validación cultural.*

**3.4 Rendimiento**

***Objetivo:***

*Asegurar tiempos de respuesta óptimos en traducción, validación y navegación, incluso en dispositivos móviles de gama media.*

***Estrategias arquitectónicas:***

* *Uso de TensorFlow Lite para inferencia rápida en dispositivos.*
* *Comunicación eficiente entre backend e IA mediante gRPC.*
* *Carga diferida de módulos educativos y multimedia.*
* *Indexación de datos y caché local para historial de traducciones.*

***Limitaciones:***

* *Procesamiento de video en tiempo real puede saturar dispositivos con poca RAM.*
* *El entrenamiento de modelos IA requiere recursos computacionales elevados.*

**4. Análisis de Requerimientos**

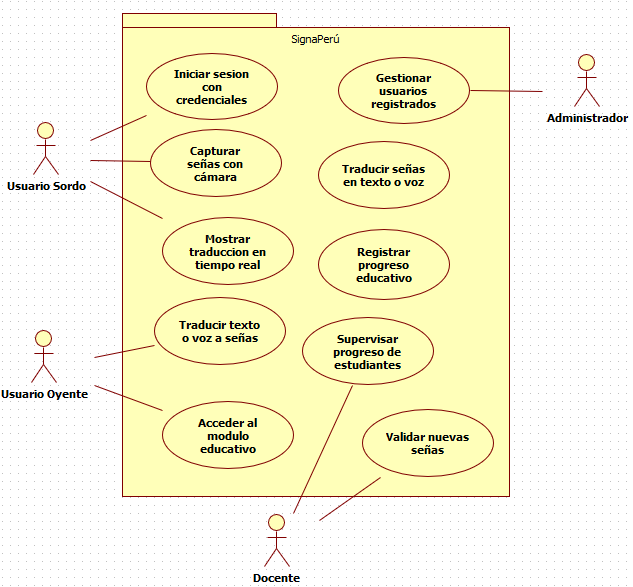
**4.1 Requerimientos funcionales**

|  |  |  |  |
| --- | --- | --- | --- |
| COD | Requerimiento Funcional | Descripción | Prioridad |
| RF01 | Traducir señas en texto o voz | El sistema debe permitir al usuario sordo traducir sus señas en texto o voz mediante reconocimiento visual. | Alta |
| RF02 | Traducir texto o voz a señas | El sistema debe permitir al usuario oyente ingresar texto o voz para recibir la traducción en LSPerú. | Alta |
| RF03 | Capturar señas con cámara | El sistema debe capturar señas mediante la cámara y procesarlas con modelos de IA entrenados. | Alta |
| RF04 | Mostrar traducción en tiempo real | El sistema debe mostrar la traducción de señas o texto/voz de forma inmediata en pantalla. | Alta |
| RF05 | Acceder al módulo educativo | El sistema debe permitir a cualquier usuario acceder a contenidos para aprender LSPerú. | Media |
| RF06 | Registrar progreso educativo | El sistema debe registrar el avance del usuario en el módulo educativo, incluyendo prácticas y evaluaciones. | Media |
| RF07 | Supervisar progreso de estudiantes | El sistema debe permitir al docente visualizar el progreso de sus estudiantes registrados. | Media |
| RF08 | Validar nuevas señas | El sistema debe permitir al administrador revisar y aprobar señas antes de que se publiquen en el sistema. | Alta |
| RF09 | Gestionar usuarios registrados | El sistema debe permitir al administrador crear, editar, eliminar y asignar roles a los usuarios. | Alta |
| RF10 | Iniciar sesión con credenciales | El sistema debe permitir a cualquier usuario iniciar sesión con usuario y contraseña válidos. | Alta |

**4.2 Requerimientos no funcionales**

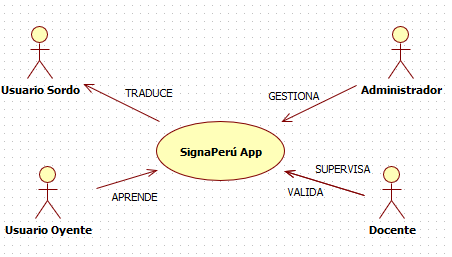
|  |  |  |  |
| --- | --- | --- | --- |
| COD | Requerimiento No Funcional | Descripción | Categoría |
| RNF01 | Respuesta rápida en traducción | La app debe responder en menos de 2 segundos ante solicitudes de traducción. | Rendimiento |
| RNF02 | Carga inicial eficiente | La app debe cargar completamente en menos de 5 segundos en dispositivos móviles. | Rendimiento |
| RNF03 | Interfaz intuitiva | La interfaz debe ser clara, simple y adaptada a usuarios sordos y oyentes. | Usabilidad |
| RNF04 | Accesibilidad universal | La app debe cumplir con estándares WCAG 2.1 nivel AA para accesibilidad visual y táctil. | Usabilidad |
| RNF05 | Protección de datos | La app debe cifrar los datos personales y aplicar autenticación segura. | Seguridad |
| RNF06 | Escalabilidad modular | La app debe permitir agregar nuevas funciones sin afectar las existentes. | Mantenibilidad |
| RNF07 | Preparación para integración externa | La app debe estar preparada para conectarse con sistemas externos en futuras versiones. | Escalabilidad |
| RNF08 | Compatibilidad | Funciona en Android, iOS y navegadores modernos. | Compatibilidad |

**5. Vistas de Caso de Uso**



**6. Vista Lógica**

**6.1 Diagrama Contextual**



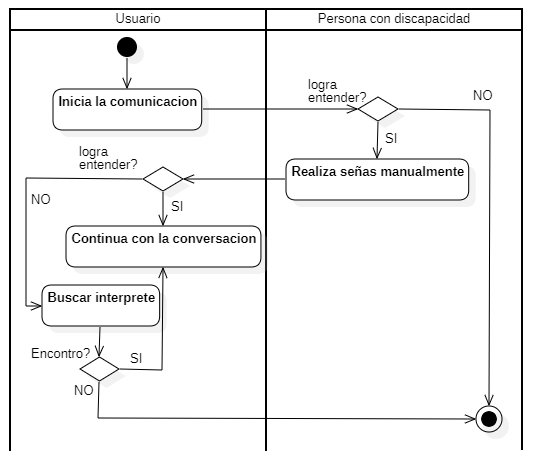
*Imagina que el sistema es una caja llamada SignaPerú App. Alrededor de esa caja hay personas que interactúan con ella:*

* ***Usuario Sordo*** *usa la app para traducir señas a texto o voz.*
* ***Usuario Oyente*** *la usa para aprender señas desde el módulo educativo.*
* ***Docente*** *puede supervisar el progreso de los estudiantes y también validar contenido.*
* ***Administrador*** *se encarga de gestionar usuarios y mantener el sistema funcionando correctamente.*

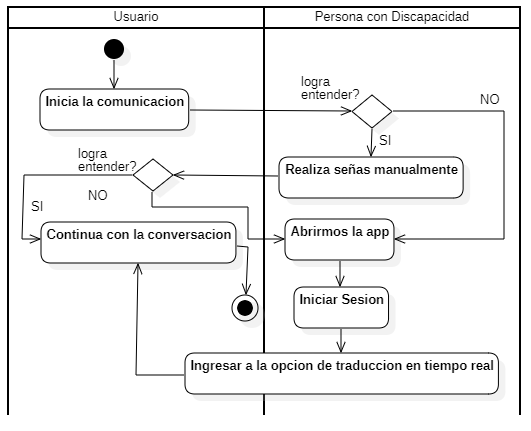
*Cada flecha indica qué acción realiza cada tipo de usuario con la app. Es como un mapa de quién hace qué.*

**7. Vista de Procesos**

**7.1 Diagrama de Proceso Actual**

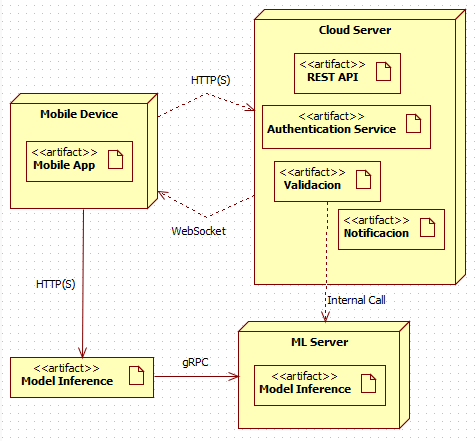
****

**7.2 Diagrama de Proceso Propuesto**

****

**8. Vista de Despliegue**

**8.1 Diagrama de Contenedor**

****

*Este diagrama explica dónde vive cada parte del sistema (en qué dispositivo o servidor) y cómo se comunican entre sí.*

***1. El usuario abre la app en su celular***

* *En el dispositivo móvil, está instalada la app (Mobile App).*
* *Esta app tiene pantallas para iniciar sesión, traducir señas, acceder a contenido educativo, etc.*

***2. La app se conecta al servidor en la nube***

* *Usa HTTP(S) para enviar datos (como credenciales, señas capturadas, texto).*
* *Usa WebSocket para recibir respuestas en tiempo real (como traducciones o notificaciones).*

***3. El servidor en la nube procesa la solicitud***

* *Tiene varios servicios:*
  + *REST API: recibe las peticiones de la app.*
  + *Authentication Service: verifica si el usuario está autorizado.*
  + *Validación: revisa si una seña es válida.*
  + *Notificación: envía mensajes al usuario (por ejemplo, “seña aprobada”).*

***4. El servidor en la nube se comunica con el servidor de IA***

* *Usa gRPC, una forma rápida y eficiente de enviar datos.*
* *Le pide al motor de IA (Model Inference) que reconozca la seña o traduzca el texto.*

***5. El servidor de IA responde***

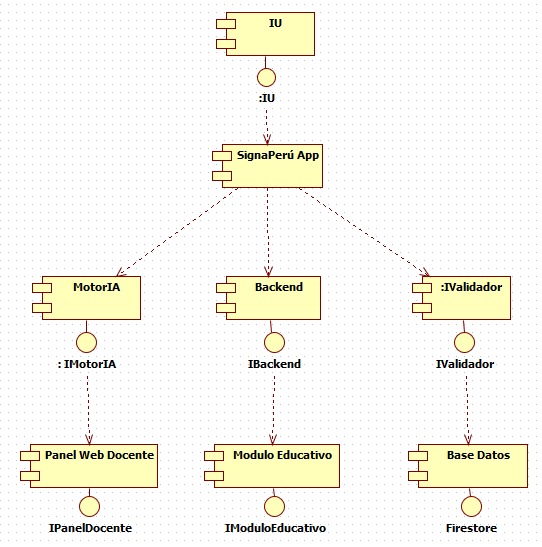
* *Devuelve el resultado (por ejemplo: “Hola” o “Gracias”).*
* *También puede enviar información a los módulos de validación o notificación.*

***6. La app muestra el resultado al usuario***

* *El usuario ve el texto, escucha la voz, o recibe una animación de la seña.*

**9. Vista de Implementación**

**9.1 Diagrama de Componentes**



*Este diagrama muestra cómo está dividido el sistema en módulos independientes (llamados componentes) y cómo se comunican entre sí. Cada componente tiene una función específica dentro de la app, y las flechas indican qué módulo necesita usar otro para funcionar.*

*Es como ver el plano de una máquina: cada parte tiene su tarea, y todas trabajan juntas para que el sistema funcione correctamente.*

***1. IU (Interfaz de Usuario)***

* *Es lo que el usuario ve y toca: botones, pantallas, formularios.*
* *Se conecta directamente con el componente central: SignaPerúApp.*

***2. SignaPerúApp (App principal)***

* *Recibe las acciones del usuario desde la IU.*
* *Decide qué módulo debe activarse según lo que el usuario quiere hacer:*
  + *Traducir señas*
  + *Aprender LSPerú*
  + *Validar señas*
  + *Supervisar estudiantes*

***3. MotorIA***

* *Si el usuario quiere traducir señas, la app llama al MotorIA.*
* *Este módulo usa inteligencia artificial para reconocer gestos y convertirlos en texto o voz.*

***4. Backend***

* *Si el usuario inicia sesión, consulta datos o guarda progreso, la app se comunica con el Backend.*
* *Este módulo gestiona usuarios, roles, autenticación y lógica interna.*

***5. Validador***

* *Si se propone una nueva seña, la app envía esa información al Validador.*
* *Este módulo permite que expertos revisen y aprueben señas nuevas.*

***6. Modulo Educativo***

* *Si el usuario accede a lecciones o prácticas, la app usa el ModuloEducativo.*
* *Este módulo muestra contenido de aprendizaje y registra el progreso.*

***7. Panel Web Docente***

* *Los docentes acceden desde navegador para supervisar estudiantes.*
* *Este panel se conecta al ModuloEducativo y a la BaseDatos.*

***8. BaseDatos (Firestore)***

* *Todos los módulos guardan y consultan información aquí.*
* *Usuarios, señas, progreso, comentarios… todo se almacena en esta base de datos.*

**10. Vista de Datos**

**10.1 Diagrama Entidad Relación**

**11. Calidad**

*Este apartado describe los atributos de calidad del sistema SignaPerú, junto con escenarios concretos que validan su comportamiento ante condiciones específicas.*

**11.1 Escenario de Seguridad**

***Objetivo:*** *Proteger los datos personales y evitar accesos no autorizados.*

|  |  |
| --- | --- |
| *Escenario* | *Un usuario intenta acceder sin credenciales válidas.* |
| *Respuesta esperada* | *El sistema bloquea el acceso y muestra un mensaje de error. Se registra el intento en el backend.* |
| *Mecanismos aplicados* | *Autenticación con Firebase Auth, cifrado de contraseñas, validación de tokens.* |

**11.2 Escenario de Usabilidad**

***Objetivo:*** *Garantizar que cualquier usuario (sordo, oyente, docente) pueda usar la app sin dificultad.*

|  |  |
| --- | --- |
| *Escenario* | *Un usuario sordo accede por primera vez a la app.* |
| *Respuesta esperada* | *La app muestra una interfaz accesible, con íconos claros, navegación táctil y soporte visual.* |
| *Mecanismos aplicados* | *Diseño conforme a WCAG 2.1 AA, pruebas de accesibilidad, tutorial inicial.* |

**11.3 Escenario de Adaptabilidad**

***Objetivo:*** *Permitir que el sistema se adapte a nuevos módulos o tecnologías sin romper lo existente.*

|  |  |
| --- | --- |
| *Escenario* | *Se desea agregar un nuevo módulo para reconocimiento facial.* |
| *Respuesta esperada* | *El sistema permite integrar el nuevo componente sin afectar los módulos actuales.* |
| *Mecanismos aplicados* | *Arquitectura modular, interfaces desacopladas, uso de servicios independientes.* |

**11.4 Escenario de Disponibilidad**

***Objetivo:*** *Asegurar que el sistema esté disponible para los usuarios en todo momento.*

|  |  |
| --- | --- |
| *Escenario* | *Un usuario accede a la app durante una hora pico.* |
| *Respuesta esperada* | *La app responde sin demoras y mantiene la sesión activa.* |
| *Mecanismos aplicados* | *Infraestructura en Firebase, escalabilidad automática, monitoreo de rendimiento.* |

**11.5 Otros Escenarios**

|  |  |  |
| --- | --- | --- |
| *Escenario* | *Objetivo* | *Un usuario intenta acceder sin credenciales válidas.* |
| *Traducción en tiempo real* | *Rendimiento* | *La app traduce señas en menos de 2 segundos.* |
| *Validación colaborativa* | *Trazabilidad* | *Cada seña validada queda registrada con fecha, usuario y estado.* |
| *Aprendizaje progresivo* | *Mantenibilidad* | *El módulo educativo permite agregar nuevas lecciones sin modificar el código base.* |