Diseño de Arquitectura del Sistema – Braille Express

**Fecha:** 16/04/2025

**Autor:** Ylia Jamile Ochoa Gutierrez

**Versión:** 1.0

# 1. Introducción

Este Este documento detalla la arquitectura del sistema para la aplicación móvil Braille Express, cuyo propósito es permitir a los usuarios convertir texto a Braille y gestionar dispositivos Bluetooth para su comunicación con hardware externo. Se define la estructura general del sistema, los componentes que lo conforman, sus interacciones y las tecnologías empleadas, todo ello bajo un enfoque moderno basado en el patrón de arquitectura Model-View-ViewModel (MVVM).

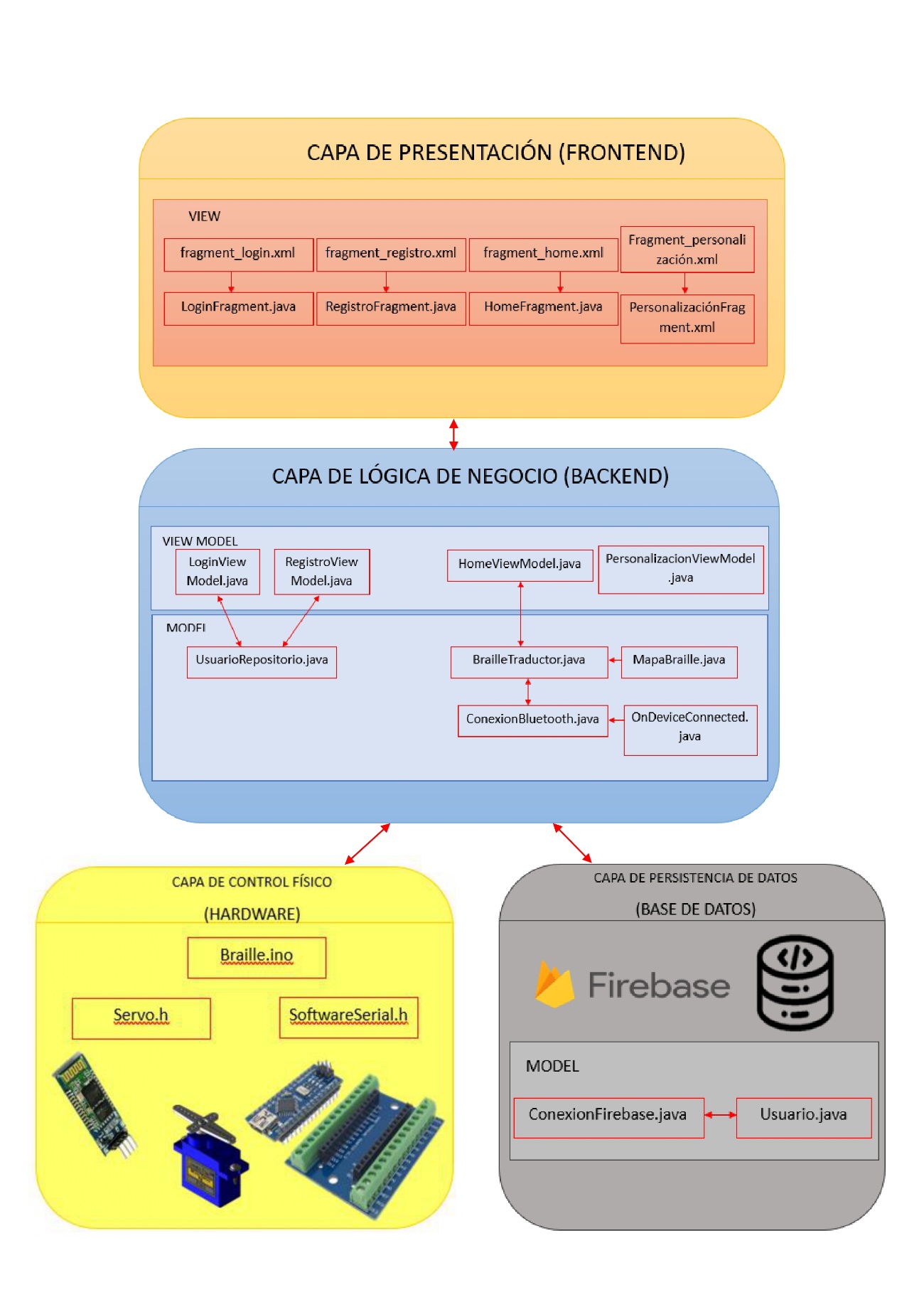
# 2. Visón General de la Arquitectura

## 2.1. Estilo Arquitectónico

El sistema se basa en una arquitectura en capas, donde cada una se corresponde con una responsabilidad claramente definida dentro del patrón MVVM. Se identifican cuatro capas principales:

* **Capa de Presentación (Frontend):** Interfaz gráfica e interacción directa con el usuario.
* **Capa de Lógica de Negocio (Backend):** Procesamiento de datos, reglas del sistema y conexión con fuentes de datos.
* **Capa de Persistencia de Datos:** Acceso y almacenamiento remoto de información (Firebase).
* **Capa de Control Físico:** Comunicación con hardware externo, como el Arduino Nano mediante Bluetooth.

## 2.2. Diagrama General de Arquitectura:



# 3. Descripción de Componentes Principales

## 3.1. Frontend (Presentación)

Esta capa corresponde a la interfaz gráfica con la que interactúa el usuario. Está diseñada bajo el patrón MVVM, permitiendo una separación clara entre los datos, la lógica de presentación y la vista.

* **Plataforma:** Android
* **Lenguaje:** Java
* **Modelo:** MVVM
* **Estilo visual:** XML Layouts
* **Responsabilidades:**
* Mostrar interfaces amigables.
* Capturar y validar entradas del usuario.
* Notificar errores, resultados y estados del sistema.
* Navegar entre pantallas mediante Navigation Component.
* **Componentes:**
* MainActivity.java: Actividad principal que contiene el menú de navegación lateral (Drawer) y gestiona los fragments.
* HomeFragment.java: Vista para ingresar texto y visualizar su traducción en braille.
* PersonalizaciónFragment.java: Interfaz para realizar personalizaciones como temas, y color de botones.
* LoginFragment.java: Pantalla de inicio de sesión con validación de usuario.
* RegistroFragment.java: Formulario para el registro de nuevos usuarios.
* fragment\_home.xml: Diseño visual del módulo de traducción.
* fragment\_personalizacion.xml: Vista de elección de temas y color de botones.
* fragment\_login.xml: Vista del formulario de login.
* fragment\_registro.xml: Vista del formulario de registro.

## 3.2. Backend (Lógica de negocio)

* **Plataforma:** Android
* **Lenguaje:** Java
* **Modelo:** MVVM
* **Responsabilidades:**
* Autenticar usuarios.
* Almacenar y recuperar datos del usuario.
* Asegurar el acceso remoto a la información.
* **Componentes:**
* HomeViewModel.java: Coordina la lógica del módulo de traducción y notifica cambios a la vista.
* PersonalizacionViewModel.java: Encargado de gestionar la lógica relacionada con la personalización de la interfaz de usuario en la aplicación.
* LoginViewModel.java: Maneja la validación de credenciales y la sesión con Firebase.
* RegistroViewModel.java: Procesa los datos del formulario y coordina el alta de usuario.
* UserRepository.java: Abstracción del acceso a Firebase; gestiona usuarios y datos.
* BrailleTraductor.java: Modelo encargado de ejecutar la conversión de texto a braille.
* MapaBraille.java: Diccionario interno que mapea caracteres a su representación braille.
* ConexionBluetooth.java: Encapsula la lógica de conexión, búsqueda y emparejamiento Bluetooth.
* OnDeviceConnected.java: Interfaz para escuchar eventos relacionados con dispositivos conectados.

## 3.3. Capa de Persistencia de Datos (Firebase)

* **Sistema Gestor:** Firebase
* **Componentes:**
* Usuario.java: Modelo de usuario que contiene datos personales y preferencias.
* ConexionFirebase.java: Clase que gestiona la gestión de usuarios para validar el login y almacena las personalizaciones del usuario.

## 3.4. Capa de Control Físico (Hardware)

* **Plataforma:** Arduino IDE
* **Responsabilidades:**
* Ejecutar el código que traduce señales recibidas vía Bluetooth en acciones físicas.
* Controlar dispositivos físicos como actuadores, servomotores o pines que representan caracteres braille en relieve.
* Interpretar los datos enviados por la app Android y convertirlos en salidas táctiles o visuales para el usuario.
* **Componentes:**
* Braille.ino: Archivo principal que contiene todo el código de control del hardware. Internamente modularizado en funciones específicas para conexión, traducción y control de servos.
* Servo.h: Librería utilizada para controlar los movimientos de los servomotores que simulan los puntos del sistema Braille.
* SoftwareSerial.h: Librería utilizada para establecer la comunicación serie con el módulo Bluetooth HC-06 sin interferir con el puerto serie principal del Arduino Nano.

# 4. Integraciones Externas (Opcionales)

* **Firebase:**
* Se utiliza para autenticación de usuarios mediante email/DNI y para almacenar de forma segura los datos del perfil.
* Usa Firebase Authentication y Firestore
* **API de Reconocimiento de Voz (**Google Speech-to-Text)**:** Permite la entrada por voz del usuario que luego es traducida a texto y posteriormente a Braille.
* Bluetooth Android API: Permite establecer conexión con dispositivos emparejados (HC-06) usando sockets seguros.
* **Bluetooth HC-06:** Dispositivo externo que recibe los datos Braille vía UART desde el Arduino Nano.

# 5. Seguridad

* Autenticación segura: Firebase con validación de DNI.
* Protección de datos: Almacenamiento cifrado y seguro de la información en la nube.
* Comunicación Bluetooth: Conexión limitada a dispositivos emparejados.
* Validación de entradas: Validar de texto para evitar errores o comandos maliciosos.
* Privacidad: No se almacena contenido de texto traducido en la nube (solo configuración de usuario).

# 6. Escalabilidad y Despliegue

**Aplicación Móvil:**

* Compatibilidad: Diseñada para funcionar de forma estable en dispositivos con Android 7.0 (Nougat) o superior, garantizando acceso a una amplia base de usuarios sin requerir versiones recientes del sistema operativo.
* Distribución: La app puede ser desplegada como APK directamente o publicada en Google Play Store para facilitar su instalación y futuras actualizaciones automáticas.
* Mantenibilidad: Gracias a la arquitectura MVVM, los componentes están desacoplados, lo cual facilita la incorporación de mejoras, corrección de errores y escalabilidad funcional sin afectar la estabilidad general.

**Backend Firebase**

* Escalabilidad automática: Firebase maneja automáticamente el crecimiento de usuarios sin necesidad de configurar o mantener servidores, lo que permite enfocarse en el desarrollo funcional.
* Gestión centralizada: La base de datos y la autenticación están centralizadas en Firebase, lo que permite actualizaciones y sincronización en tiempo real para múltiples usuarios.

**Dispositivo Físico (Arduino + Servos)**

* Modularidad: El sistema electrónico está diseñado con componentes de fácil adquisición (Arduino Nano, HC-06, servos, Shield IO), permitiendo su réplica o reparación sin dificultad.
* Escalabilidad educativa: La solución puede ser implementada en múltiples espacios educativos o de rehabilitación, facilitando el acceso a la lectura en Braille mediante kits replicables.

Posibles problemas que pueden surgir durante el desarrollo.

* Rango limitado del HC-06: Bluetooth clásico tiene un alcance de ~10 metros. Se recomienda su uso en escritorio o áreas controladas.
* Recursos limitados del Arduino: El Arduino Nano es adecuado para tareas simples. Se debe mantener la lógica central en la app para evitar sobrecargas.
* Latencia en la transmisión: El envío caracter por caracter puede producir pequeñas demoras, solucionables con buffers o codificación optimizada.
* Gestión de permisos: Android 7 gestiona permisos en tiempo de instalación o por solicitud directa (no dinámica como en Android 12+), lo cual facilita su implementación.

# 7. Conclusiones

La arquitectura propuesta para Braille Express establece una solución clara, modular y enfocada en la accesibilidad. Al utilizar el patrón MVVM en todo el proyecto (frontend, backend, Firebase y comunicación), se logra una alta separación de responsabilidades, mantenibilidad y facilidad para extender el sistema en el futuro.

El uso combinado de Firebase, Bluetooth clásico (HC-06) y Arduino Nano permite crear una aplicación móvil eficiente con un dispositivo físico complementario, funcional incluso en entornos educativos o de bajos recursos.

Esta solución:

* Cumple con los requerimientos de compatibilidad para Android 7+
* No depende de hardware costoso
* Garantiza independencia de servidores físicos gracias a Firebase
* Puede ser expandida fácilmente a nuevos dispositivos o entornos

BrailleExpress representa una herramienta inclusiva, replicable y con alto impacto social, orientada a mejorar el acceso a la lectura para personas con discapacidad visual mediante el uso de tecnología móvil accesible y hardware asequible.