Diseño de Arquitectura del Sistema – Braille Express

**Fecha:** 16/04/2025

**Autor: Ylia Jamile Ochoa Gutierrez**

**Versión:** 1.0

# 1. Introducción

Este documento describe la arquitectura del sistema Braille Express, una solución tecnológica que busca mejorar el acceso a la lectura de personas con discapacidad visual mediante un dispositivo físico y una aplicación móvil que traduce texto y voz a Braille.  
El sistema está compuesto por una app Android y un microcontrolador Arduino Nano, comunicados vía Bluetooth. Se define la estructura general del sistema, los componentes que lo conforman, sus interacciones y las tecnologías empleadas.

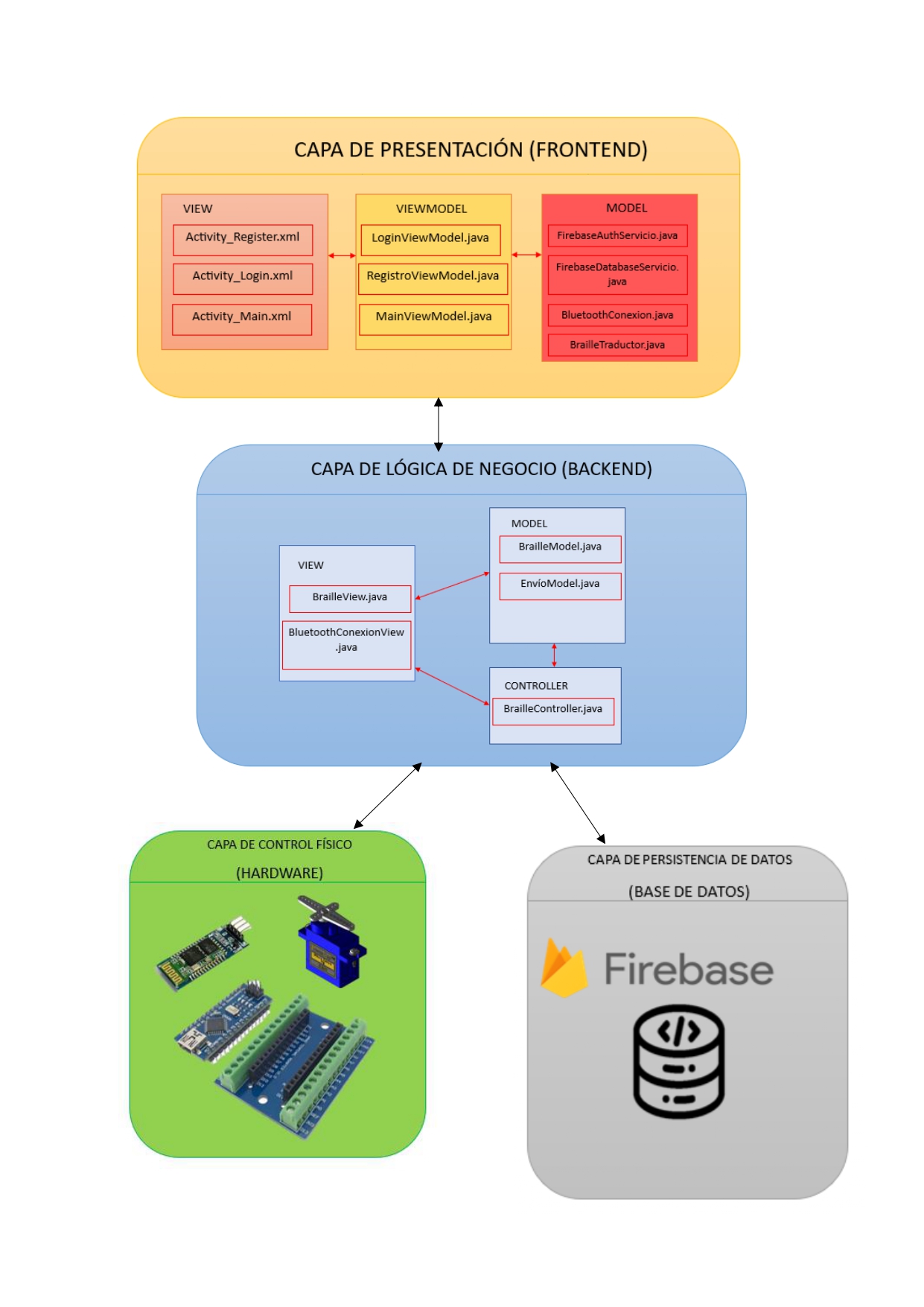
# 2. Visón General de la Arquitectura

## 2.1. Estilo Arquitectónico

El sistema adopta una arquitectura en capas basada en el modelo cliente-servidor, donde el cliente concentra la mayor parte de la lógica del sistema. Se identifican cuatro capas principales:

* **Capa de Presentación (Frontend)**: Interfaz con la que interactúa el usuario.
* **Capa de Lógica de Negocio** **(Backend)**: Lógica de traducción a Braille y conexión con servicios.
* **Capa de Persistencia de Datos (Base de datos)**: Manejo de usuarios y sincronización en la nube.
* **Capa de Control Físico**: Representación física de los caracteres Braille usando servomotores.

## 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.

* Modelo: MVVM
* Lenguaje: Java
* IDE: Android Studio
* Estilo visual: XML Layouts
* View:
* Activity\_Login.xml: Formulario para el inicio de sesión, con campos para DNI, botón de acceso, y mensaje de error.
* Activity\_Registro.xml: Formulario de registro con campos de entrada para nombres, apellidos, edad, género, DNI.
* Activity\_Main.xml:
* TextView, para mostrar el estado de conexión Bluetooth.
* EditText, para ingresar texto a traducir.
* TextView, para mostrar el resultado traducido a Braille.
* Botones, para traducir, enviar y desconectar.
* ViewModel:
* LoginViewModel.java: Valida entradas del usuario, maneja la lógica de autenticación y expone estados para la UI.
* RegistroViewModel.java: Procesa los datos del formulario de registro y coordina la creación de usuario con el backend.
* MainViewModel.java: Gestiona la lógica de traducción de texto a Braille, y la conexión/envío al módulo Bluetooth.
* Model:
* FirebaseAuthServicio.java: Se conecta con Firebase Authentication para iniciar sesión o registrar usuarios.
* FirebaseDatabaseServicio.java: Accede o guarda datos en Realtime Database.
* BluetoothConexion.java: Gestiona la conexión Bluetooth con el Arduino Nano.
* BrailleTraductor.java: Traduce texto a la codificación Braille (lista de patrones de 6 bits).

## 3.2. Backend (Lógica de negocio)

* Modelo: MVC
* Lenguaje: Java
* IDE: Android Studio
* View:
* BrailleView.java: Muestra el resultado de la conversión a Braille y el estado del envío al Arduino.
* BluetoothConexionView.java: Muestra el estado de conexión con el arduino
* Model:
* BrailleModel.java: Contiene la lógica para convertir texto en caracteres Braille.
* EnvíoModel.java: Gestiona el envío de los caracteres Braille al Arduino Nano mediante Bluetooth.
* Controller:
* BrailleController.java: Recibe las solicitudes desde el frontend, coordina el flujo entre modelo y vista. Llama al modelo para traducir texto a Braille o enviar datos al Arduino.

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

* Sistema Gestor: Firebase
* Responsabilidad:
  + Registro y login de usuarios.
  + Almacenamiento de configuraciones del usuario.
  + Sincronización de datos entre sesiones y dispositivos.
  + Gestión segura mediante reglas de acceso y encriptación de datos.

## 3.4. Capa de Control Físico (Hardware)

* Componente Principal: Arduino Nano con módulo Bluetooth HC-06
* Elementos clave:
* 6 servomotores SG90 conectados a Shield Nano IO V1.0
* Comunicación inalámbrica vía Bluetooth
* **Responsabilidades:**
* Recepción de caracteres desde la app.
* Activación de servomotores para representar los puntos Braille.
* Reinicio de conexión en caso de errores.

# 4. Integraciones Externas (Opcionales)

* **Firebase:** Autenticación y almacenamiento remoto.
* **API de Reconocimiento de Voz:** Google Speech-to-Text (para entrada por voz).
* **Bluetooth HC-06:** Comunicación inalámbrica entre la app y el Arduino.

# 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

Posibles problemas que pueden surgir durante el desarrollo.

* Aplicación Móvil:
* Compatible con Android 7.0 o superior.
* Desplegable mediante Google Play Store.
* Backend Firebase:
* Escalable automáticamente según la demanda.
* Dispositivo Físico:
* Modular y replicable con componentes accesibles.
* Escalable en instituciones educativas mediante múltiples kits Braille.
* Posibles desafíos:
* Conectividad Bluetooth limitada a 10 metros.
* Capacidad limitada del Arduino en procesamiento.

# 7. Conclusiones

La arquitectura propuesta para Braille Express busca un equilibrio entre simplicidad técnica, accesibilidad y escalabilidad. Aprovecha tecnologías conocidas y confiables como Android, Firebase y Arduino, y se enfoca en ofrecer una solución tangible y funcional para la comunidad con discapacidad visual. La estructura modular permite su mejora futura mediante nuevas funcionalidades, sin comprometer su rendimiento ni accesibilidad.