Manual Técnico — Braille Express

**Fecha**: 26 /06/2025

**Autor**: Ylia Jamile Ochoa Gutierrez

**Versión**: 1.0

# 1. Introducción

Este manual técnico documenta los aspectos internos del desarrollo del sistema **Braille Express**, una solución compuesta por una aplicación móvil Android y un dispositivo físico con conectividad Bluetooth, orientado a la traducción de texto digital a Braille para mejorar la comprensión lectora de personas con discapacidad visual. El documento está dirigido a desarrolladores, técnicos de soporte y profesionales interesados en entender la arquitectura, tecnologías y lógica funcional del sistema.

# 2. Tecnologías Utilizadas

|  |  |
| --- | --- |
| Componente | Tecnología |
| Aplicación Móvil | Android Studio (Java + XML) |
| Comunicación | Bluetooth HC – 06 |
| Microcontrolador | Arduino Nano |
| Lógica de Traducción | Traductor ViewModel Firebase Realtime |
| Base de Datos Remota | Firebase Realtime Database |
| Persistencia Local | SharedPreferences |
| Librerías Adicionales | Firebase SDK, Arduino Servo Library, Arduino Bluetooth Library |

# 3. Estructura del Proyecto

/brailleexpress

│

├── data/

│ ├── model/

│ │ ├── BrailleTraductorModel.java

│ │ ├── ConfiguracionModel.java

│ │ ├── HistorialModel.java

│ │ ├── MapaBrailleModel.java

│ │ ├── UsuarioModel.java

│ ├── repositorio/

│ │ ├── ConfiguracionRepositorio.java

│ │ ├── HistorialRepositorio.java

│ │ ├── UsuarioRepositorio.java

├── ui

│ ├── Configuracion

│ │ ├── ConfiguracionViewModel

│ │ ├── ConfiguracionFragment

│ ├── Diccionario

│ │ ├── ConfiguracionViewModel

│ │ ├── ConfiguracionFragment

│ ├──Historial

│ │ ├── HistorialViewModel

│ │ ├── HistorialFragment

│ ├── Login

│ │ ├── LoginViewModel

│ │ ├── LoginFragment

│ ├── Registro

│ │ ├── RegistroViewModel

│ │ ├── RegistroFragment

│ ├── Splash

│ │ ├── SplashFragment

│ ├── Traductor

│ │ ├── TraductorViewModel

│ │ ├── TraductorFragment

├── ConexionBluetooth

├── MainActivity

res

├── layout

│ ├── activity\_main.xml

│ ├── fragment\_configuracion.xml

│ ├── fragment\_diccionario.xml

│ ├── fragment\_historial.xml

│ ├── fragment\_login.xml

│ ├── fragment\_registro.xml

│ ├── fragment\_splash.xml

│ ├── fragment\_traductor.xml

│ ├── item\_historial.xml

# 4. Arquitectura General

Arquitectura en **capas**:

* **Capa de Presentación:** Java + XML layouts
* **Capa de Persistencia de datos:** Firebase Realtime Database
* **Capa de Control de Datos:** Arduino Nano, Bluetooth HC-06, servomotores

# 5. Descripción de Componentes

## 5.1 Capa de Presentación

* **Plataforma:** Android
* **Lenguaje:** Java
* **Modelo Arquitectónico:** MVVM
* **Diseño Visual:** XML Layouts

Esta capa permite la interacción directa del usuario con la aplicación móvil. Está organizada mediante fragments y actividades, respetando el principio de separación de responsabilidades del patrón MVVM.

* **Responsabilidades:**
  + Mostrar interfaces visuales accesibles.
  + Capturar y validar las entradas del usuario.
  + Navegar entre pantallas utilizando el NavigationComponent.
  + Notificar errores, confirmaciones y estados del sistema.
* **Componentes Principales:**

View:

|  |  |  |
| --- | --- | --- |
| Archivo Java/XML | Función clave | |
| MainActivity.java | | Contiene el menú lateral (DrawerLayout) y gestiona la navegación. | |
| TraductorFragment.java / fragment\_traductor.xml | | Permite ingresar texto y visualizar su traducción a Braille. | |
| ConfiguracionFragment.java / fragment\_configuracion.xml | | Permite seleccionar tema claro/oscuro y color de botones. | |
| LoginFragment.java / fragment\_login.xml | | Autenticación del usuario por DNI. | |
| RegistroFragment.java / fragment\_registro.xml | | Registro de nuevos usuarios con validación. | |
| HistorialFragment.java / fragment\_historial.xml | | Visualiza el historial de traducciones realizadas. | |
| fragment\_splash.xml | | Muestra el logo y eslogan durante el arranque de la app. | |
| fragment\_diccionario.xml | | Muestra imágenes del alfabeto Braille como material de apoyo. | |
| ConexionBluetooth.java | | Gestiona la conexión Bluetooth con el módulo HC-06 del Arduino. | |

View Model:

|  |  |
| --- | --- |
| Archivo Java | Responsabilidad principal |
| TraductorViewModel.java | Controla la lógica de traducción a Braille. |
| ConfiguracionViewModel.java | Administra preferencias de tema y color de botones. |
| LoginViewModel.java | Maneja el inicio y cierre de sesión mediante Firebase. |
| RegistroViewModel.java | Valida y registra los datos de nuevos usuarios. |
| HistorialViewModel.java | Recupera y actualiza datos del historial desde Firebase. |

Model:

|  |  |
| --- | --- |
| Clase Java | Contenido / Estructura |
| UsuarioModel.java | Datos del usuario: DNI, nombres, apellidos, correo, contraseña. |
| HistorialModel.java | Texto original, traducción, fecha, ID de usuario, clave Firebase. |
| ConfiguracionModel.java | Preferencias del usuario: tema y color de botones. |
| BrailleTraductorModel.java | Lógica de conversión letra a Braille. |
| MapaBrailleModel.java | Diccionario de correspondencia entre caracteres y puntos Braille. |

## 5.2 Capa de Persistencia de Datos (Firebase)

* **Base de Datos:** Firebase Realtime Database
* **Almacenamiento local complementario:** SharedPreferences

La capa de persistencia permite guardar los datos del usuario (cuenta, historial, configuración) en la nube, lo que permite acceder a la misma configuración desde cualquier dispositivo si el usuario se identifica correctamente.

* **Responsabilidades:**
  + Sincronizar historial de traducciones entre sesiones y dispositivos.
  + Almacenar configuraciones personalizadas de interfaz (tema, color).
  + Verificar credenciales de acceso en tiempo real.
  + Mantener la sesión de usuario activa entre usos.
* **Componentes Clave:**

|  |  |
| --- | --- |
| Archivo Java | Función |
| UsuarioRepositorio.java | Registro e inicio de sesión de usuarios en Firebase. |
| HistorialRepositorio.java | Guarda las traducciones por usuario en la tabla TbHistorial. |
| ConfiguracionRepositorio.java | Administra preferencias del usuario en TBConfiguracion. |

## 5.3 Capa de Control Físico

* **Plataforma:** Arduino Nano
* **Lenguaje:** C++ (Arduino IDE)
* **Módulo de Comunicación:** HC-06 (Bluetooth)
* **Actuadores:** 6 servomotores conectados a shield de expansión.

Esta capa representa el componente físico del sistema. El dispositivo recibe datos en tiempo real desde la app Android vía Bluetooth, decodifica los caracteres y activa los servos correspondientes a los puntos del Braille.

* **Responsabilidades:**
* Escuchar vía Bluetooth los caracteres enviados por la app.
* Activar los servomotores de acuerdo con el patrón Braille de cada letra.
* Apagar o reposicionar los servos según el flujo de impresión.
* **Componente Clave:**

|  |  |
| --- | --- |
| Archivo | Descripción |
| braille.ino | Contiene toda la lógica de conexión Bluetooth, lectura de caracteres, mapeo Braille y activación de servos. |

Funciones Internas:

* + setup(): Inicializa el Bluetooth y servos.
  + loop(): Escucha los caracteres recibidos.
  + activarBraille(char letra): Activa servos según la letra recibida.

# 6. Instalación y Ejecución

## 6.1 Requisitos

* Dispositivo Android 7.0 o superior
* APK de Braille Express generado desde Android Studio
* Cuenta Firebase con proyecto y Realtime Database habilitada
* Módulo Bluetooth HC-06 emparejado con el dispositivo Android
* Arduino Nano con shield de expansión para servomotores
* Arduino IDE para cargar el programa al Nano

## 6.2 Aplicación Braille Express

1. Transferir el archivo BrailleExpress.apk al dispositivo Android.

2. En el dispositivo, habilitar "Instalar apps de fuentes desconocidas" en ajustes.

3. Abrir el archivo APK y seguir el asistente de instalación.

## 6.3 Base de Datos

* Crear base de datos: BrailleExpres\_db
* Agregar el archivo google-services.json en la carpeta de Braille Express en AndroidStudio.
* Ejecutar la sincronización de Gradle (app).
* Habilitar Realtime Database
* Establecer reglas de lectura/escritura

## 6.4 Dispositivo Electrónico

* Conectar el Arduino Nano al PC mediante USB
* Seleccionar los puertos de conexión.
* Conectar el módulo HC-06:
* Cargar el sketch `braille.ino`
* Cargar el código al Arduino

# 7. Seguridad

* Se utiliza autenticación de usuarios mediante Firebase, lo que restringe el acceso solo a personas registradas con DNI y contraseña válidos.
* La aplicación implementa validaciones en los formularios de entrada para prevenir campos vacíos o datos malformados.
* La sesión de usuario se mantiene de forma segura utilizando SharedPreferences, almacenando el DNI de forma privada para personalización y acceso a datos.
* Se establecen reglas de seguridad en Firebase Realtime Database, permitiendo leer y escribir datos solo a usuarios autenticados (auth != null).
* Cada usuario accede únicamente a sus datos mediante el identificador único idDNI, aislando la información por cuenta.
* Las contraseñas de los usuarios están protegidas mediante cifrado en los servidores de Firebase, sin posibilidad de acceso a texto plano.
* Toda la comunicación entre la aplicación y los servicios de Firebase se realiza mediante conexiones seguras (HTTPS).
* El módulo Bluetooth HC-06 solo permite conexión desde dispositivos emparejados, lo que evita interferencias externas o accesos no autorizados.
* El código cargado en el Arduino filtra comandos inválidos y ejecuta únicamente las instrucciones esperadas, evitando respuestas erróneas del hardware.
* No se almacenan credenciales sensibles en texto plano ni se transmiten datos críticos sin cifrado.

# 8. Buenas Prácticas

* Separación de lógica por fragmento y ViewModel: La aplicación sigue el patrón arquitectónico MVVM, permitiendo una clara separación entre la lógica de negocio (ViewModel), la visualización (Fragmentos XML) y los datos (Modelos). Esto facilita el mantenimiento, la escalabilidad y las pruebas del sistema.
* Actualización automática del historial al enviar traducción: Cada vez que el usuario realiza una nueva traducción de texto a Braille, esta se registra automáticamente en la base de datos en tiempo real sin necesidad de intervención manual, garantizando una trazabilidad completa de las acciones del usuario.
* Configuración persistente sincronizada por ID de usuario: Las preferencias de tema (claro/oscuro) y color de botones se almacenan en Firebase bajo el identificador único del usuario (idDNI). Esto permite recuperar la configuración personalizada desde cualquier dispositivo donde el usuario inicie sesión.
* Código documentado y modular:  
  El sistema está organizado en clases y archivos específicos para cada funcionalidad (por ejemplo: TraductorViewModel, ConfiguracionRepositorio, Braille.ino). Cada clase está comentada y estructurada para facilitar la comprensión del código por parte de otros desarrolladores.
* Uso eficiente de recursos: Se implementa una comunicación eficiente con Firebase y el módulo Bluetooth, minimizando el consumo de batería y procesamiento en dispositivos móviles.
* Control de errores y validaciones robustas: Las entradas de usuario son validadas antes de ser procesadas, evitando errores por campos vacíos o formatos inválidos, y se implementan capturas de excepciones para prevenir cierres inesperados de la app.

# 9. Futuras Mejoras

* Soporte para lectura de imágenes (OCR): Se planea implementar un sistema de reconocimiento óptico de caracteres (OCR) que permita convertir texto impreso en imágenes a formato Braille, ampliando las fuentes de entrada para usuarios.
* Mejora del diseño accesible para baja visión: Se incorporarán opciones adicionales como aumento de tamaño de fuente, alto contraste, narración por voz y navegación por gestos, dirigidas a usuarios con visión reducida parcial.
* Estadísticas de uso por estudiante: Se desarrollará un módulo que permita generar reportes de actividad del usuario (cantidad de traducciones, progreso, uso por día), útil para docentes y padres de familia en contextos educativos.
* Compatibilidad con múltiples idiomas: Se buscará soportar la traducción desde distintos idiomas (inglés, francés, quechua, etc.) al sistema Braille, permitiendo un uso más inclusivo en contextos multilingües.
* Modo sin conexión con almacenamiento local: El sistema podría almacenar traducciones localmente cuando no hay acceso a Internet, y sincronizarlas con Firebase automáticamente cuando se restablezca la conexión.
* Interfaz web para administración remota: Se plantea el desarrollo de un panel web para docentes o cuidadores que les permita revisar y administrar el historial, configuración o rendimiento del estudiante de forma remota.

# 10. Contacto

Para soporte técnico o colaboración: devBrailleExpress@gmail.com