

**FACULTAD DE INGENIERÍA**

**ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**

**“Desarrollo de un Dispositivo Electrónico y App Móvil para Traducir Texto Digital a Braille”**

**Autor:**

**OCHOA GUTIERREZ YLIA JAMILE**

**Cusco – Perú**

**2025**

**Introducción**

En el mundo, la situación de las personas con discapacidad visual refleja los desafíos que existen en los ámbitos educativo y social respecto a la inclusión y accesibilidad. De acuerdo con la Organización Mundial de la Salud (2020), aproximadamente 2,200 millones de personas en el mundo tienen algún tipo de discapacidad visual, de las cuales 1,000 millones podrían haber sido prevenidas con tratamiento adecuado. En el contexto peruano, el Instituto Nacional de Estadística e Informática (2017) reporta que el 48.3% de las personas con discapacidad visual enfrentan serias dificultades en su vida diaria, especialmente en lo que respecta a la educación.

Uno de los mayores retos que enfrentan los estudiantes con discapacidad visual es el acceso a materiales educativos accesibles. La escasez de textos en Braille y el limitado uso de tecnologías asistidas en las aulas son barreras que afectan negativamente su proceso de aprendizaje. Este contexto es especialmente evidente en la comprensión lectora, una habilidad esencial para el desarrollo académico. A pesar de los avances tecnológicos, las herramientas disponibles para mejorar la comprensión lectora de los estudiantes con discapacidad visual en muchos casos siguen siendo insuficientes o inadecuadas para sus necesidades específicas.

El desarrollo de un dispositivo electrónico que traduce texto digital a Braille ofrece una solución innovadora para abordar este problema. Este tipo de tecnología no solo permitiría que los estudiantes accedan a una mayor cantidad de material educativo en formato Braille, sino que también contribuiría al desarrollo de su comprensión lectora. Sin embargo, el éxito de tal dispositivo dependerá de su capacidad para responder a las dificultades que actualmente enfrentan los estudiantes con discapacidad visual al utilizar textos en formato digital, así como de su efectividad en comparación con los métodos tradicionales de enseñanza.

**Definiciones, Acrónimos y Abreviaturas**

* **UI:** Interfaz de Usuario
* **API:** Interfaz de Programación de Aplicaciones
* **DB:** Bases de Datos Remota
* **TTS:** Text-to-Speech (Texto a Voz)
* **Módulo HC-06:** Módulo de comunicación inalámbrica utilizado con Arduino
* **Firebase:** Plataforma de base de datos en la nube para autenticación y almacenamiento.
* **Servo SG90:** Motor que permite el control preciso de la posición angular de piezas.
* **Arduino Nano:** Una placa electrónica de desarrollo que permite programar microcontroladores.
* **Android Studio:** Entorno de desarrollo integrado (IDE) oficial para crear aplicaciones Android.
* **Arduino IDE:** Entorno de programación que permite escribir, compilar y cargar código en placas Arduino mediante el lenguaje de programación basado en C/C++.
* Shield Nano IO V1.0**:** Placa de control de servos

**Referencias**

* Documentación oficial de Android Studio, Firebase, Arduino IDE y Google Play.
* ISO 17049:2013 Evaluación de conformidad de la transcripción Braille.
* IEEE 802.15.1 Standard Bluetooth Comunication
* UNESCO: Directrices para la producción de documentos en Braille

**Problema a Resolver**

En el contexto educativo actual, los estudiantes con discapacidad visual enfrentan múltiples barreras para acceder a materiales de lectura en formatos accesibles. A pesar de los avances tecnológicos, la mayoría de los recursos educativos digitales no están adaptados para su uso por personas con discapacidad visual, lo que limita su participación en el proceso de aprendizaje y afecta su rendimiento académico. La falta de dispositivos accesibles que traduzcan texto digital a Braille de manera rápida y efectiva representa un obstáculo significativo para garantizar la inclusión educativa.

**Objetivos del Proyecto**

El proyecto busca abordar esta problemática mediante el desarrollo de un dispositivo electrónico que traduzca texto digital a Braille, permitiendo a los estudiantes acceder de forma autónoma a materiales educativos digitales. Al facilitar la traducción de texto y voz a Braille, promoviendo la inclusión y garantizando que los personas con discapacidad visual tengan las mismas oportunidades educativas.

**Objetivo General:**

Desarrollar un dispositivo electrónico para traducir texto digital a Braille para mejorar la experiencia de lectura y comprensión de las personas con discapacidad visual, optimizando su acceso a la información de manera eficiente y práctica.

**Objetivos Específicos:**

* Desarrollar una interfaz intuitiva y accesible que permita a los usuarios cargar, visualizar y traducir textos digitales de manera sencilla y rápida.
* Diseñar y construir el prototipo del dispositivo electrónico que permita la traducción de texto digital a Braille de manera rápida y efectiva.
* Validar el rendimiento y la precisión del dispositivo mediante pruebas experimentales en entornos educativos y sociales.

**Público Objetivo (Usuarios Finales)**

**Características de los Usuarios**

* **Usuarios principales:**
* Personas con discapacidad visual.
* **Beneficios esperados:**
* Pueden convertir texto digital en Braille de forma rápida.
* No dependen de terceros para acceder a material escrito.
* La traducción precisa a Braille facilita el aprendizaje.
* La interfaz accesible permite un uso sencillo y eficiente.
* **Usuarios principales:**
* Docentes especializados.
* **Beneficios esperados:**
* Facilita la enseñanza del Braille a los estudiantes.
* Permite convertir cualquier texto en material accesible.
* Reduce el tiempo de preparación de material en Braille.
* **Usuarios principales:**
* Centros educativos especializados.
* **Beneficios esperados:**
* Integración de tecnología en la enseñanza del Braille.
* Posibilita una educación equitativa para estudiantes con discapacidad visual.
* Alternativa más económica frente a impresoras Braille tradicionales.
* Pueden adaptar cualquier documento digital a Braille fácilmente.
* **Usuarios principales:**
* Bibliotecas para personas con discapacidad visual.
* **Beneficios esperados:**
* Traducción de textos digitales a Braille en tiempo real.
* Pueden acceder a libros y documentos sin intermediarios.
* Menos necesidad de almacenar libros físicos en Braille.
* Incrementa el acceso a la información y la cultura.
* **Usuarios principales:**
* Organizaciones de personas con discapacidad visual.
* **Beneficios esperados:**
* Facilita el acceso a información sin depender de terceros.
* Herramienta útil en formación académica y profesional.
* Adopción de tecnologías accesibles para la comunidad.

## **Restricciones**

* Capacidad del Arduino Nano es limitada en memoria y procesamiento, lo que puede afectar la velocidad de respuesta.
* Solo se pueden manejar 6 servos para representar los puntos Braille, lo que restringe la velocidad de impresión.
* La comunicación del módulo Bluetooth HC-06 con el dispositivo Android está limitada a una distancia de aproximadamente 10 metros.
* La aplicación solo funcionará en dispositivos con Android 7.0 o superior.
* El procesamiento de voz dependerá de la calidad del micrófono del dispositivo.
* La aplicación deberá ser optimizada para no consumir demasiados recursos del dispositivo móvil.
* La interfaz debe cumplir con estándares de accesibilidad para personas con discapacidad visual.
* Si el Bluetooth no está activado o hay interferencias, el sistema no podrá comunicarse con el Arduino.
* Requiere aprendizaje inicial, los usuarios deben familiarizarse con la aplicación y el dispositivo Braille.

## **Suposiciones y Dependencias**

* El Arduino Nano será suficiente para manejar la traducción a Braille, a pesar de sus limitaciones.
* Los 6 servos serán suficientes para representar correctamente los caracteres en Braille.
* El usuario estará dentro del rango de 10 metros para que la conexión Bluetooth funcione correctamente.
* Los dispositivos donde se instale la aplicación tendrán Android 7.0 o superior y serán compatibles.
* El micrófono del dispositivo será adecuado para el reconocimiento de voz sin problemas.
* Los usuarios estarán dispuestos a aprender a utilizar la aplicación y el dispositivo Braille.
* Se requieren librerías de terceros para la conversión de voz a texto.
* El sistema depende del acceso a hardware específico: Arduino Nano, servos y Shield Nano IO V1.0 y módulo Bluetooth HC06.
* La conexión Bluetooth es esencial para la comunicación entre el dispositivo móvil y el Arduino.
* El acceso a smartphones con Android y conexión a internet para actualizaciones podría ser un desafío en algunas instituciones.
* Los servomotores SG90 pueden desgastarse con el tiempo y necesitar calibración o reemplazo.

**Funcionalidades Principales**

Lista de características clave de la primera versión del software.

**Funcionalidades esenciales:**

* Ingreso manual de texto para traducción a Braille:
* Permite al usuario escribir directamente en la interfaz para su traducción a Braille.
* Ingreso de texto por voz para traducción a Braille:
* Reconocimiento de voz en español para convertir el dictado en texto y posteriormente en Braille.
* Conectividad Bluetooth con Arduino Nano:
* Comunicación en tiempo real con el Arduino para el control preciso de los 6 servomotores.
* Control de servomotores para las impresiones de caracteres.
* Activación y desactivación de cada servo para representar los puntos del sistema Braille.
* Panel de control con acceso mediante login:
* Control de usuarios para el uso del aplicativo y configuraciones preferidas.
* Interfaz accesible:
* Diseño intuitivo y compatible con lectores de pantalla.
* Personalización de temática en la aplicación.
* Salida de los caracteres uno por uno:
* Impresión secuencial de cada carácter para asegurar la precisión en la lectura en Braille.
* Notificaciones y alertas:
* Alerta en caso de pérdida de conexión bluetooth.
* Indicador de batería baja o problemas en los servomotores.
* Actualización del software.
* Descarga e instalación de nuevas versiones para mejorar la precisión y añadir funcionalidades.
* Notificación al usuario sobre actualizaciones disponibles.
* Autenticación y almacenamiento de configuraciones en Firebase**.**
* Gestiona los usuarios que están registrados y los que se registrarán.
* Almacena las configuraciones del dispositivo.

**Funcionalidades futuras (Opcionales):**

* Implementación de un módulo para traducir documentos completos.
* Posibilidad de cargar archivos en formatos como PDF, DOCX y TXT.
* Traducción automática de todo el contenido del documento a texto Braille.
* Traducción bidireccional (texto a Braille y Braille a texto):
* Capacidad de ingresar texto en Braille (a través de un teclado especializado o interfaz táctil) y traducirlo a texto digital.
* Implementación de OCR:
* Captura de imágenes de documentos impresos o manuscritos.
* Extracción del texto mediante técnicas de reconocimiento óptico.
* Compatibilidad con dispositivos de impresión Braille:
* Conexión con impresoras Braille para generar copias físicas de los textos traducidos.
* Acceso a bibliotecas digitales:
* Integración con plataformas de libros y recursos educativos accesibles.
* Descarga directa de libros en formatos compatibles para su traducción a Braille.

**Requisitos Técnicos**

Define el stack tecnológico del proyecto.

**Lenguajes y Frameworks:**

* Android Studio (Java)
* Creación de la aplicación móvil.
* Arduino IDE (C )
* Control de servomotores.
* Comunicación Bluetooth mediante módulo HC-06.
* Envío y recepción de datos.
* Firebase:
* Base remota para el Login.
* Google Text-to-Speech (TTS)
* Reconocimiento de texto por voz.

**Compatibilidad:**

* Compatible con dispositivos Android.
* Compatibilidad con dispositivos que trabajen con Android 7.0 a superior.
* Compatible con Arduino Nano:
* Compatibilidad con el modelo Arduino Nano.
* Módulo Bluetooth HC-006:
* Envío y recepción de datos
* Servomotores SG90:
* Conexión con el Arduino Nano y para la impresión de los caracteres.
* Dispositivos de entrada de voz:
* Compatibilidad con micrófonos integrados en los dispositivos Android para el reconocimiento de voz.
* Firebase:
* Integración con la base de datos remota Firebase para el ingreso de usuarios.

**Requisitos Específicos**

**Requisitos Funcionales**

* RF01: Registro de usuarios con DNI.
* RF02: Inicio y cierre de sesión en cualquier dispositivo móvil con sincronización en la nube.
* RF03: Campo de texto para ingreso manual.
* RF04: Ingreso de texto mediante comandos de voz.
* RF05: Traducción automáticamente el texto ingresado a caracteres Braille.
* RF06: Conexión de la aplicación vía Bluetooth con el Arduino Nano.
* RF07: El Arduino enviará órdenes a los servomotores para moverse de acuerdo al caracter
* RF08: Se mostrarán notificaciones en pantalla para confirmar acciones y alertas para errores.
* RF09: Se notificará a los usuarios cuando haya actualizaciones disponibles.
* RF10: Se podrá reiniciar la conexión Bluetooth manualmente si se detecta un error de comunicación.

**Requisitos No Funcionales**

* RNF1: El dispositivo electrónico debe responder en menos de 5 segundos para garantizar una experiencia fluida una vez enviada la información.
* RNF2: La aplicación debe cumplir con los estándares de accesibilidad, asegurando compatibilidad con lectores de pantalla.
* RNF3: La aplicación mostrará si está conectado o desconectado.
* RNF4: La comunicación Bluetooth debe ser estable y con bajo consumo energético para optimizar la autonomía del dispositivo.
* RNF5: La aplicación debe ser escalable para permitir futuras mejoras y nuevas funcionalidades.
* RNF6: La autenticación y el almacenamiento en Firebase deben garantizar la seguridad y privacidad de los datos del usuario.
* RNF7: La interfaz debe ser intuitiva y permitir la navegación sencilla para usuarios con discapacidad visual.
* RNF8: El consumo de batería del dispositivo debe ser optimizado para un uso prolongado.
* RNF9: El sistema debe permitir actualizaciones sin afectar la configuración del usuario.
* RNF10: El hardware debe ser resistente y garantizar una operación confiable a largo plazo.
* RNF11: La aplicación debe ser compatible con dispositivos Android 7.0 o superior.

**Requisitos de Interfaz de Usuario**

* Interfaz accesible y minimalista – Diseñada para ser intuitiva y fácil de usar.
* Uso de colores contrastantes – Para mejorar la visibilidad de los elementos en pantalla.
* Fuentes grandes y legibles – Para facilitar la lectura a personas con baja visión.
* Botones grandes y bien espaciados – Para facilitar la interacción táctil.
* Opción para ingresar texto manualmente.
* Botón para activar la entrada de voz.
* Botón “Traducir” para convertir el texto en Braille.
* Área donde se muestra el texto original.
* Representación gráfica de la traducción en Braille.
* Indicador de estado de conexión con Arduino (Bluetooth).
* Botón “Enviar” para enviar la señal al Arduino.
* Lista de dispositivos disponibles.
* Switch para emparejar con el módulo HC-006.
* Indicador de estado de conexión.
* Control por voz – Para ingresar texto sin necesidad de escribir.
* Notificaciones y sonidos – Para confirmar acciones como conexión Bluetooth o traducción exitosa.
* Modo horizontal y vertical – Adaptabilidad a diferentes orientaciones de pantalla.

**Requisitos de Hardware y Software**

* Android Studio 7.0 o superior – Para el desarrollo de la aplicación móvil.
* SDK de Android – Con soporte para Bluetooth y procesamiento de texto.
* Firebase – Para almacenamiento de datos.
* Librerías de Procesamiento de Voz – Para convertir voz en texto.
* IDE Arduino – Para programar el Arduino Nano y controlar los servomotores.
* Librerías para Arduino:
* Servo.h – Para el control de los servomotores.
* SoftwareSerial.h – Para la comunicación Bluetooth HC-006.
* Arduino Nano – Microcontrolador para gestionar la activación de los pines Braille.
* Módulo Bluetooth HC-06 – Para la comunicación inalámbrica entre la app y el Arduino.
* 6 Servos SG90 – Para representar los 6 puntos del sistema Braille.
* Shield Nano IO V1.0 – Para administrar la alimentación y el control de los motores.
* Batería recargable o fuente de alimentación – Para alimentar el Arduino y los servos.
* Celda Braille mecánica – Superficie donde los servos activarán los puntos Braille.
* Dispositivo Android – Compatible con la app de traducción.

**Riesgos y Limitaciones**

Posibles problemas que pueden surgir durante el desarrollo.

**Riesgos:**

* Problemas de conectividad bluetooth:
* Interferencias o pérdida de señal entre el dispositivo móvil y el módulo Arduino.
* Errores en la precisión de la traducción de voz a texto.
* Reconocimiento incorrecto de palabras o frases debido al ruido, variación en la pronunciación, problemas con los micrófonos, etc.
* Desfase en la sincronización entre texto y braille:
* Retrasos en la respuesta del dispositivo al traducir el texto.
* Fallas en los servomotores:
* Desgaste o falta de calibración que afecte la precisión de la impresión de los caracteres.
* Problemas de compatibilidad de hardware:
* Incompatibilidad entre versiones de Arduino o módulos Bluetooth con la aplicación.
* Errores de programación o bugs:
* Fallos en el código que puedan provocar cierres inesperados o errores en la traducción.
* Latencia en la impresión de los caracteres:
* Tiempo de respuesta lento al enviar comandos desde la app al dispositivo.
* Dificultades en la integración de APIs:
* Problemas al integrar el reconocimiento de voz y la traducción a Braille en una sola aplicación.
* Consumo de batería excesiva:
* La comunicación bluetooth y los servomotores pueden agotar rápidamente la batería del dispositivo.
* Vulnerabilidades de seguridad:
* Posibles accesos no autorizados al sistema a través de bluetooth.
* Problemas de actualización de software:
* Fallos al actualizar el firmware del dispositivo o la aplicación móvil.
* Accesibilidad de la interfaz:
* La interfaz podría no ser completamente accesible para personas con discapacidad visual.
* Errores en la gestión de la base de datos en Firebase:
* Fallos en la sincronización o pérdida de datos almacenados en la nube.

**Limitaciones:**

* El dispositivo no soportará traducción mediante OCR.
* Dependencia de dispositivos Android para el uso de la aplicación.
* Requiere conexión Bluetooth estable para el funcionamiento continuo.
* Limitación en el número de caracteres traducidos de manera simultánea, lo que puede afectar la lectura de textos extensos.
* Capacidad limitada de los servomotores para representar caracteres complejos o símbolos especiales.
* No incluye traducción de idiomas adicionales, solo trabaja con texto en castellano.
* Dependencia de energía eléctrica o baterías para el funcionamiento del dispositivo, lo que puede limitar su uso en zonas sin acceso a electricidad.
* No incluye retroalimentación táctil para indicar caracteres traducidos de forma incorrecta.
* La aplicación no permitirá la traducción de archivos en formatos PDF, DOCX u otros documentos digitales directamente.
* El sistema de reconocimiento de voz puede presentar limitaciones ante acentos o pronunciaciones diferentes.
* El mantenimiento de los componentes físicos, como los servomotores y la conexión Bluetooth, puede requerir asistencia técnica especializada.

**Alcance del Proyecto**

Define qué incluirá y qué quedará fuera en la primera versión.

**Lo que incluirá:**

* Desarrollo de la aplicación móvil con traducción de texto manual y de voz:
* Creación de una interfaz intuitiva y accesible para el ingreso manual de texto y comandos por voz.
* Implementación de algoritmos de procesamiento de texto para traducir caracteres al sistema Braille.
* Integración de un módulo de reconocimiento de voz para capturar comandos y convertirlos en texto.
* Visualización de los caracteres traducidos en pantalla antes de enviarlos al dispositivo.
* Sistema de login para el control de usuarios.
* Implementación del dispositivo Arduino con servomotores.
* Configuración de la placa Arduino Uno R3 para controlar los 6 servomotores que representan los puntos Braille.
* Programación de algoritmos de posicionamiento de servomotores para generar los patrones Braille según la traducción recibida.
* Montaje estructural para asegurar la estabilidad y precisión en el movimiento de los servomotores.
* Inclusión de indicadores visuales o auditivos que confirmen el funcionamiento correcto del dispositivo.
* Implementación de un sistema de alertas en caso de fallas en los servomotores o problemas de conectividad.
* Conectividad Bluetooth entre la aplicación y el dispositivo.
* Integración del módulo Bluetooth HC-06 con el Arduino para la comunicación inalámbrica.
* Implementación de mensajes de confirmación de conexión y estado funcional desde la aplicación.
* Optimización del consumo energético para prolongar la duración de la batería durante el uso continuo.
* Interfaz de usuario accesible:
* Diseño de una interfaz optimizada para usuarios con discapacidad visual, incluyendo opciones de accesibilidad como lectura en voz alta de los menús.
* Implementación de botones grandes y fácilmente identificables para facilitar la navegación.
* Personalización de la interfaz según las preferencias del usuario.
* Sistema de actualizaciones automáticas:
* Notificación al usuario cuando haya actualizaciones disponibles.
* Personalización del aplicativo:
* Permitirá en cambio de temática de la aplicación blanco y oscuro.

**Lo que NO incluirá (por ahora):**

* Traducción de imágenes mediante OCR:
* No se implementará la captura y reconocimiento de texto a partir de imágenes.
* Integración con otros dispositivos que no sean Android:
  + La aplicación será exclusiva para sistemas Android y no se desarrollarán versiones para iOS u otros sistemas operativos.
* Compatibilidad con otros módulos de conectividad (Wi-Fi, NFC, etc.):
* Solo se utilizará Bluetooth como medio de comunicación.
* Soporte multilingüe:
  + La traducción estará limitada al idioma castellano y no se contemplará la inclusión de otros idiomas.
* Reconocimiento automático de textos complejos o fórmulas matemáticas:
  + La aplicación no soportará la traducción de expresiones matemáticas o símbolos especiales a Braille.
* Almacenamiento de textos traducidos en la nube:
  + Los textos ingresados y traducidos no se guardarán en servidores externos ni en bases de datos en línea.
* Personalización avanzada del aplicativo:
  + No se incluirán opciones para cambiar la configuración de los servomotores por medio de la aplicación.
* Integración con asistentes virtuales (como Google Assistant, Gemini o Alexa):
  + No habrá la posibilidad de utilizar los asistentes virtuales para el uso de la aplicación.
* Manejo de múltiples dispositivos conectados simultáneamente:
  + La aplicación estará limitada a la conexión con un solo dispositivo Arduino a la vez.

**Épicas e Historias de Usuario**

**Épica 1: Gestión de Usuarios y Sesiones**

**Historia de Usuario 1.1: Registro de usuarios**

Como usuario, quiero registrarme en la aplicación utilizando mi DNI para asegurar la autenticidad de mi cuenta.

**Criterios de Aceptación:**

* El sistema debe validar el formato del DNI como usuario.
* Se debe almacenar la información de manera segura.

**Estimación:** 5 puntos

**Definición de "Hecho":**

* El usuario puede registrarse exitosamente.
* La información se almacena de forma segura en la base de datos.

**Historia de Usuario 1.2: Inicio y cierre de sesión en la nube**

Como usuario, centros educativos especializados y organizaciones de personas con discapacidad visual, quiero iniciar y cerrar sesión en cualquier dispositivo móvil con sincronización en la nube para acceder a mis datos de manera segura.

**Criterios de Aceptación:**

* La aplicación debe permitir autenticación desde múltiples dispositivos.
* La información debe estar sincronizada en la nube.

**Estimación:** 8 puntos

**Definición de "Hecho":**

* El usuario puede iniciar y cerrar sesión sin problemas.
* Los datos del usuario se mantienen sincronizados en la nube.

**Épica 2: Ingreso y Traducción de Texto a Braille**

**Historia de Usuario 2.1: Ingreso manual de texto**

Como docente especializado, quiero ingresar texto manualmente en la aplicación para que pueda ser traducido a Braille y mis alumnos puedan leer.

**Criterios de Aceptación:**

* La aplicación debe permitir el ingreso de caracteres alfanuméricos y signos de puntuación básicos.
* Debe validar que el texto ingresado sea compatible con la conversión a Braille.

**Estimación:** 5 puntos

**Definición de "Hecho":**

* El usuario puede ingresar texto manualmente.
* La conversión a Braille es precisa.

**Historia de Usuario 2.2: Ingreso de texto por voz**

Como docente especializado, quiero ingresar texto mediante comandos de voz para convertirlo a Braille y mis alumnos puedan leer.

**Criterios de Aceptación:**

* La aplicación debe transcribir la voz a texto utilizando una API de reconocimiento de voz.
* El usuario podrá editar el texto antes de su conversión.

**Estimación:** 8 puntos

**Definición de "Hecho":**

* La transcripción de voz a texto es precisa.
* El usuario puede editar y convertir el texto sin problemas.

**Historia de Usuario 2.3: Traducción automáticamente el texto ingresado**

Como usuario con discapacidad visual, quiero que el texto ingresado en la aplicación se traduzca automáticamente a Braille sin necesidad de acciones adicionales, para mejorar la fluidez y rapidez en la conversión.

**Criterios de Aceptación:**

* La aplicación debe detectar automáticamente cuando el usuario finaliza la escritura.
* Debe iniciar la conversión a Braille sin necesidad de presionar un botón adicional.
* El usuario debe poder visualizar la traducción en un área específica de la interfaz.
* La traducción debe ser precisa y compatible con el sistema Braille.

**Estimación:** 5 puntos

**Definición de "Hecho":**

* La traducción se activa automáticamente al finalizar la escritura.
* El texto traducido se muestra correctamente en la interfaz de usuario.
* No se requiere acción adicional por parte del usuario para iniciar la traducción.
* La conversión es precisa y se valida con pruebas funcionales.

**Historia de Usuario 2.4: Historial de traducciones**

Como docente especializado y bibliotecas inclusivas, quiero que las traducciones a Braille se guarden en la nube y se pueda volver a abrirlas con mi cuenta, para acceder a las conversiones anteriores sin necesidad de volver a escribirlas.

**Criterios de Aceptación:**

* La aplicación debe guardar automáticamente cada traducción realizada (texto original + Braille) asociada a la cuenta del usuario.
* El usuario puede ver una lista de sus traducciones guardadas.
* Al seleccionar una entrada del historial, el sistema debe mostrar el texto original y su equivalente en Braille.

**Estimación:** 10 puntos

**Definición de "Hecho":**

* Límite de almacenamiento: 100 traducciones por usuario (mostrar advertencia al llegar al 90%).
* El usuario puede ver el historial desde cualquier dispositivo con su cuenta.
* La traducción seleccionada se muestra correctamente en la interfaz de usuario.

**Épica 3: Conectividad con el Dispositivo Braille**

**Historia de Usuario 3.1: Conexión Bluetooth**

Como personas con discapacidad visual, quiero que la aplicación se conecte al dispositivo Braille mediante Bluetooth para transmitir los caracteres.

**Criterios de Aceptación:**

* La aplicación debe detectar dispositivos Bluetooth disponibles.
* Debe establecer y mostrar el estado de la conexión (conectado/desconectado).

**Estimación:** 10 puntos

**Definición de "Hecho":**

* La aplicación se conecta correctamente al dispositivo Bluetooth.
* El estado de la conexión es visible y actualizable.

**Historia de Usuario 3.2:** **Envío de caracteres a Arduino**

Como persona con discapacidad visual, quiero que la aplicación envíe los caracteres al Arduino para que los represente en Braille.

**Criterios de Aceptación:**

* La aplicación debe transmitir los caracteres al dispositivo electrónico.
* El Arduino debe activar los servomotores correspondientes para representar cada carácter en Braille.

**Estimación:** 10 puntos

**Definición de "Hecho":**

* Los caracteres se transmiten correctamente y el Arduino los representa en Braille.

**Historia de Usuario 3.3: Reinicio manual de bluetooth**

Como docente especializado o usuario, quiero poder reiniciar la conexión Bluetooth manualmente en caso de errores de comunicación.

**Criterios de Aceptación:**

* La aplicación debe permitir restablecer la conexión manualmente.
* Debe notificar cuando la conexión se restablezca correctamente.

**Estimación:** 5 puntos

**Definición de "Hecho":**

* La opción de reinicio manual funciona correctamente.
* La aplicación notifica al usuario el estado del reinicio.

**Épica 4: Interfaz de Usuario y Notificaciones**

**Historia de Usuario 4.1: Personalización de tema y color de botones**

Como centro educativo especializado, quiero personalizar el tema de la aplicación (claro/oscuro) y cambiar los colores de los botones para adaptar la interfaz a las preferencias y necesidades.

**Criterios de Aceptación:**

* La aplicación debe ofrecer al menos dos opciones de tema predefinidas: claro (light) y oscuro (dark).
* El cambio debe aplicarse inmediatamente y guardarse en la configuración del usuario.
* Permitir al usuario cambiar el color principal de los botones (ej: azul, verde, rojo) desde una paleta predefinida.
* Los colores deben cumplir con estándares de contraste WCAG 2.1 (> 4.5:1 para accesibilidad).
* La aplicación debe cargar automáticamente la última configuración al iniciar.
* Notificación al usuario cuando se apliquen los cambios ("Tema actualizado correctamente").

**Estimación:** 5 puntos

**Definición de "Hecho":**

* La aplicación debe cargar automáticamente la última configuración al iniciar.
* Los cambios de color en botones son visibles y guardados tras reiniciar la app.
* Se cumplen las ratios de contraste WCAG 2.1

**Historia de Usuario 4.2: Notificaciones de acciones y errores**

Como persona con discapacidad visual y docente, quiero recibir notificaciones cuando haya errores o confirmaciones de acciones dentro de la aplicación.

**Criterios de Aceptación:**

* Se deben mostrar mensajes emergentes para errores como "Bluetooth no disponible".
* Deben existir confirmaciones para acciones como "Texto enviado correctamente".

**Estimación:** 3 puntos

**Definición de "Hecho":**

* Los mensajes emergentes se muestran correctamente en la interfaz.
* Se pueden diferenciar entre errores y confirmaciones exitosas.

**Historia de Usuario 4.3: Notificación de actualizaciones**

Como usuario y organizaciones de personas con discapacidad visual, quiero recibir notificaciones cuando haya actualizaciones disponibles para la aplicación.

**Criterios de Aceptación:**

* La aplicación debe verificar actualizaciones en la nube.
* Debe notificar a los usuarios cuando haya una nueva versión disponible.

**Estimación:** 3 puntos

**Definición de "Hecho":**

* La aplicación verifica actualizaciones correctamente.
* Los usuarios reciben notificaciones de nuevas versiones.

**4. Priorización del Backlog**

|  |  |  |  |
| --- | --- | --- | --- |
| Prioridad | Historia de Usuario | Estado | Estimación (Puntos) |
| Alta | Registro seguro | Pendiente | 5 |
| Alta | Inicio y cierre de sesión en la nube | Pendiente | 8 |
| Alta | Ingreso manual de texto | Pendiente | 5 |
| Media | Ingreso de texto por voz | Pendiente | 8 |
| Alta | Traducción automática del texto ingresado | Pendiente | 5 |
| Alta | Historial de traducciones | Pendiente | 10 |
| Alta | Conexión Bluetooth | Pendiente | 10 |
| Alta | Envío de caracteres a Arduino | Pendiente | 10 |
| Media | Reinicio manual de Bluetooth | Pendiente | 5 |
| Media | Personalización de tema y color de botones | Pendiente | 5 |
| Media | Notificaciones de acciones y errores | Pendiente | 3 |
| Baja | Notificaciones de actualizaciones | Pendiente | 3 |

**Historias de Usuario Seleccionadas para el Sprint 1**

Se ha considerado realizar las 6 tareas del HI 1.1 y la primera tarea del HI 1.2, con un total de 6 tareas porque se desea cumplir con la cantidad de horas que se necesita para un Sprint, en este caso el Sprint 1 durará 66 horas.

**Historia de Usuario 1.1: Registro de usuarios**

**Descripción:** Como usuario, quiero registrarme en la aplicación utilizando mi DNI para asegurar la autenticidad de mi cuenta.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 1.1.1 | Diseñar estructura de la base de datos para almacenar usuarios | Backend Dev | 10 | Pendiente |
| 1.1.2 | Implementar validación de DNI | Backend Dev | 10 | Pendiente |
| 1.1.3 | Implementar almacenamiento seguro de datos | Backend Dev | 12 | Pendiente |
| 1.1.4 | Crear interfaz de registro | Frontend Dev | 8 | Pendiente |
| 1.1.5 | Diseñar flujo de registro | UX/UI Designer | 8 | Pendiente |
| 1.1.6 | Pruebas unitarias y de integración | QA Engineer | 10 | Pendiente |

**Historia de Usuario 1.2: Inicio y cierre de sesión en la nube**

**Descripción:** Como usuario, quiero iniciar y cerrar sesión en cualquier dispositivo móvil con sincronización en la nube para acceder a mis datos de manera segura.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 1.2.1 | Diseñar experiencia de usuario para login/logout | UX/UI Designer | 8 | Pendiente |

**Total de Esfuerzo Estimado**

|  |  |  |
| --- | --- | --- |
| Función | Responsable | Horas Totales |
| UX/UI Design | UX/UI Designer | 16 |
| Desarrollo Frontend | Frontend Dev | 8 |
| Desarrollo Backend | Backend Dev | 32 |
| Pruebas | QA Engineer | 10 |
| Total General |  | 66 |

**Historias de Usuario Seleccionadas para el Sprint 2**

Se ha considerado realizar las 5 tareas restantes del HI 1.2 y las 2 primeras tareas del HI 2.1, con un total de 6 tareas porque se desea cumplir con la cantidad de horas que se necesita para un Sprint, en este caso del Sprint 2 durará 68 horas.

**Historia de Usuario 1.2: Inicio y cierre de sesión en la nube**

**Descripción:** Como usuario, quiero iniciar y cerrar sesión en cualquier dispositivo móvil con sincronización en la nube para acceder a mis datos de manera segura.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 1.2.2 | Diseñar sistema de autenticación | Backend Dev | 10 | Pendiente |
| 1.2.3 | Implementar autenticación desde múltiples dispositivos | Backend Dev | 12 | Pendiente |
| 1.2.4 | Desarrollar sincronización de datos en la nube | Backend Dev | 10 | Pendiente |
| 1.2.5 | Diseñar pantalla de inicio de sesión | Frontend Dev | 8 | Pendiente |
| 1.2.6 | Pruebas de inicio y cierre de sesión | QA Engineer | 10 | Pendiente |

**Historia de Usuario 2.1: Ingreso manual de texto**

**Descripción:** Como usuario con discapacidad visual, quiero ingresar texto manualmente en la aplicación para que pueda ser traducido a Braille.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 2.1.1 | Diseñar experiencia de usuario para ingreso manual | UX/UI Designer | 8 | Pendiente |
| 2.1.2 | Crear campo de entrada de texto en la interfaz | Frontend Dev | 8 | Pendiente |

**Total de Esfuerzo Estimado**

|  |  |  |
| --- | --- | --- |
| Función | Responsable | Horas Totales |
| UX/UI Design | UX/UI Designer | 8 |
| Desarrollo Frontend | Frontend Dev | 16 |
| Desarrollo Backend | Backend Dev | 32 |
| Pruebas | QA Engineer | 10 |
| Total General |  | 66 |

**Historias de Usuario Seleccionadas para el Sprint 3**

Se ha considerado realizar las 2 tareas restantes del HI 2.1 y las 5 tareas del HI 2.2, con un total de 7 tareas porque se desea cumplir con la cantidad de horas que se necesita para un Sprint, en este caso del Sprint 3 durará 68 horas.

**Historia de Usuario 2.1: Ingreso manual de texto**

**Descripción:** Como usuario con discapacidad visual, quiero ingresar texto manualmente en la aplicación para que pueda ser traducido a Braille.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 2.1.3 | Implementar validación de caracteres y signos de puntuación | Backend Dev | 16 | Pendiente |
| 2.1.4 | Integrar conversión a Braille | Backend Dev | 12 | Pendiente |
| 2.1.5 | Pruebas funcionales del ingreso de texto | QA Engineer | 10 | Pendiente |

**Historia de Usuario 2.2: Ingreso de texto por voz**

**Descripción:** Como usuario con discapacidad visual, quiero ingresar texto mediante comandos de voz para convertirlo a Braille.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 2.2.1 | Diseñar experiencia de usuario para ingreso por voz | UX/UI Designer | 8 | Pendiente |
| 2.2.2 | Integrar API de reconocimiento de voz | Backend Dev | 12 | Pendiente |
| 2.2.3 | Implementar edición del texto transcrito | Frontend Dev | 10 | Pendiente |

**Total de Esfuerzo Estimado**

|  |  |  |
| --- | --- | --- |
| Función | Responsable | Horas Totales |
| UX/UI Design | UX/UI Designer | 8 |
| Desarrollo Frontend | Frontend Dev | 10 |
| Desarrollo Backend | Backend Dev | 40 |
| Pruebas | QA Engineer | 10 |
| Total General |  | 68 |

**Historias de Usuario Seleccionadas para el Sprint 4**

Se ha considerado realizar las 2 últimas tareas del HI 2.2, las 3 tareas del HI 2.3 y la primera tarea del HI 2.4, con un total de 6 tareas porque se desea cumplir con la cantidad de horas que se necesita para un Sprint, en este caso el Sprint 4 durará 65 horas.

**Historia de Usuario 2.2: Ingreso de texto por voz**

**Descripción:** Como usuario con discapacidad visual, quiero ingresar texto mediante comandos de voz para convertirlo a Braille.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 2.2.4 | Desarrollar interfaz para mostrar y editar el texto transcrito | Frontend Dev | 8 | Pendiente |
| 2.2.5 | Pruebas de transcripción de voz a texto | QA Engineer | 10 | Pendiente |

**Historia de Usuario 2.3: Traducción automática a Braille**

**Descripción:** Como usuario, quiero que el texto ingresado en la aplicación se traduzca automáticamente a Braille sin necesidad de acciones adicionales, para mejorar la fluidez y rapidez en la conversión.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 2.3.1 | Implementar detección de finalización de entrada de texto | Backend Dev | 10 | Pendiente |
| 2.3.2 | Integrar conversión automática a Braille | Backend Dev | 10 | Pendiente |
| 2.3.3 | Mostrar traducción en la interfaz de usuario | Frontend Dev | 7 | Pendiente |
| 2.3.4 | Pruebas de conversión automática | QA Engineer | 10 | Pendiente |

**Historia de Usuario 2.4: Historial de Traducciones**

**Descripción:** Como usuario, quiero que las traducciones a Braille se guarden en la nube y pueda volver a abrirlas con mi cuenta, para acceder a mis conversiones anteriores sin necesidad de volver a escribirlas.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 2.4.1 | Diseñar esquema de base de datos para almacenar traducciones | Backend Dev | 10 | Pendiente |

**Total de Esfuerzo Estimado**

|  |  |  |
| --- | --- | --- |
| Función | Responsable | Horas Totales |
| UX/UI Design | UX/UI Designer | 0 |
| Desarrollo Frontend | Frontend Dev | 15 |
| Desarrollo Backend | Backend Dev | 30 |
| Pruebas | QA Engineer | 20 |
| Total General |  | 65 |

# **Historias de Usuario Seleccionadas para el Sprint 5**

Se ha considerado realizar las 4 tareas restantes del HI 2.4, las 3 tareas del HI 3.2 y la primera tarea del HI 3.3, con un total de 6 tareas porque se desea cumplir con la cantidad de horas que se necesita para un Sprint, en este caso el Sprint 5 durará 64 horas.

**Historia de Usuario 2.4: Historial de Traducciones**

**Descripción:** Como usuario, quiero que las traducciones a Braille se guarden en la nube y pueda volver a abrirlas con mi cuenta, para acceder a mis conversiones anteriores sin necesidad de volver a escribirlas.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 2.4.2 | Diseñar interfaz de historial | UI/UX Designer | 8 | Pendiente |
| 2.4.3 | Implementar frontend del historial (Vista de lista) | Frontend Dev | 8 | Pendiente |
| 2.4.4 | Implementar sincronización multi-dispositivo | Backend Dev | 10 | Pendiente |
| 2.4.5 | Pruebas de carga | QA Engineer | 8 | Pendiente |

## **Historia de Usuario 3.1: Conexión Bluetooth**

**Descripción:** Como usuario, quiero que la aplicación se conecte al dispositivo Braille mediante Bluetooth para transmitir los caracteres.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 3.1.1 | Detectar dispositivos Bluetooth disponibles | Backend Dev | 16 | Pendiente |
| 3.1.2 | Implementar conexión y gestión de estado | Backend Dev | 14 | Pendiente |

# **Total de Esfuerzo Estimado**

|  |  |  |
| --- | --- | --- |
| Función | Responsable | Horas Totales |
| UX/UI Design | UX/UI Designer | 8 |
| Desarrollo Frontend | Frontend Dev | 8 |
| Desarrollo Backend | Backend Dev | 40 |
| Pruebas | QA Engineer | 8 |
| Total General |  | 64 |

# **Historias de Usuario Seleccionadas para el Sprint 6**

Se ha considerado realizar las 2 tareas restantes del HI 3.3, las 3 tareas del HI 4.1 y las 3 tareas del HI 4.2, con un total de 8 tareas porque se desea cumplir con la cantidad de horas que se necesita para un Sprint, en este caso el Sprint 6 durará 68 horas.

**Historia de Usuario 3.1: Conexión Bluetooth**

**Descripción:** Como usuario, quiero que la aplicación se conecte al dispositivo Braille mediante Bluetooth para transmitir los caracteres.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 3.1.3 | Diseñar interfaz de conexión Bluetooth | Frontend Dev | 8 | Pendiente |
| 3.1.4 | Pruebas de conexión y estabilidad | QA Engineer | 10 | Pendiente |

**Historia de Usuario 3.2: Envío de caracteres a Arduino**

**Descripción:** Como usuario, quiero que la aplicación envíe los caracteres al Arduino para que los represente en Braille.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 3.2.1 | Implementar transmisión de caracteres a Arduino | Backend Dev | 16 | Pendiente |
| 3.2.2 | Configurar Arduino para representar caracteres en Braille | Backend Dev | 14 | Pendiente |
| 3.2.3 | Pruebas de transmisión y representación en Braille | QA Engineer | 10 | Pendiente |

**Historia de Usuario 3.3: Reinicio manual de Bluetooth**

**Descripción:** Como usuario, quiero poder reiniciar la conexión Bluetooth manualmente en caso de errores de comunicación.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 3.3.1 | Implementar opción de reinicio manual | Backend Dev | 10 | Pendiente |

# **Total de Esfuerzo Estimado**

|  |  |  |
| --- | --- | --- |
| Función | Responsable | Horas Totales |
| UX/UI Design | UX/UI Designer | 0 |
| Desarrollo Frontend | Frontend Dev | 8 |
| Desarrollo Backend | Backend Dev | 40 |
| Pruebas | QA Engineer | 20 |
| Total General |  | 68 |

# **Historias de Usuario Seleccionadas para el Sprint 7**

Se ha considerado realizar las 2 tareas restantes del HI 3.3 y las 6 tareas del HI 4.1, con un total de 8 tareas porque se desea cumplir con la cantidad de horas que se necesita para un Sprint, en este caso el Sprint 7 durará 69 horas.

**Historia de Usuario 3.3: Reinicio manual de Bluetooth**

**Descripción:** Como usuario, quiero poder reiniciar la conexión Bluetooth manualmente en caso de errores de comunicación.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 3.3.2 | Diseñar notificación del estado de reinicio | Frontend Dev | 8 | Pendiente |
| 3.3.3 | Pruebas del reinicio manual | QA Engineer | 10 | Pendiente |

**Historia de Usuario 4.1: Personalización de Tema y Color de Botones**

**Descripción:** Como usuario, quiero personalizar el tema de la aplicación (claro/oscuro) y cambiar los colores de los botones para adaptar la interfaz a mis preferencias y necesidades.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 4.1.1 | Diseñar componentes UI para selector de temas (claro/oscuro) y color de botones | UI/UX Designer | 7 | Pendiente |
| 4.1.2 | Implementar lógica de cambio de tema global | Frontend Dev | 8 | Pendiente |
| 4.1.3 | Implementar selector de color para botones | Frontend Dev | 8 | Pendiente |
| 4.1.4 | Implementar notificaciones de cambio de tema/color | Frontend Dev | 8 | Pendiente |
| 4.1.5 | Configurar persistencia de preferencias | Backend Dev | 10 | Pendiente |
| 4.1.6 | Pruebas de cambio de temas y color de botones | QA Engineer | 10 | Pendiente |

# **Total de Esfuerzo Estimado**

|  |  |  |
| --- | --- | --- |
| Función | Responsable | Horas Totales |
| UX/UI Design | UX/UI Designer | 7 |
| Desarrollo Frontend | Frontend Dev | 32 |
| Desarrollo Backend | Backend Dev | 20 |
| Pruebas | QA Engineer | 10 |
| Total General |  | 69 |

**Historias de Usuario Seleccionadas para el Sprint 8**

Se ha considerado realizar las 3 tareas restantes del HI 4.2 y las 3 tareas del HI 4.3, con un total de 6 tareas porque se desea cumplir con la cantidad de horas que se necesita para un Sprint, en este caso el Sprint 8 durará 54 horas.

**Historia de Usuario 4.2: Notificaciones de acciones y errores**

**Descripción:** Como usuario, quiero recibir notificaciones cuando haya errores o confirmaciones de acciones dentro de la aplicación.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 4.2.1 | Implementar mensajes emergentes para errores | Frontend Dev | 8 | Pendiente |
| 4.2.2 | Implementar mensajes de confirmación de acciones | Frontend Dev | 8 | Pendiente |
| 4.2.3 | Pruebas de visualización de notificaciones | QA Engineer | 10 | Pendiente |

**Historia de Usuario 4.3: Notificación de actualizaciones**

**Descripción:** Como usuario, quiero recibir notificaciones cuando haya actualizaciones disponibles para la aplicación.

Tareas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| 4.3.1 | Implementar verificación de actualizaciones en la nube | Backend Dev | 10 | Pendiente |
| 4.3.2 | Diseñar e implementar notificación de nueva versión | Frontend Dev | 8 | Pendiente |
| 4.3.4 | Pruebas de notificación de actualización | QA Engineer | 10 | Pendiente |

**Total de Esfuerzo Estimado**

|  |  |  |
| --- | --- | --- |
| Función | Responsable | Horas Totales |
| UX/UI Design | UX/UI Designer | 0 |
| Desarrollo Frontend | Frontend Dev | 24 |
| Desarrollo Backend | Backend Dev | 10 |
| Pruebas | QA Engineer | 20 |
| Total General |  | 54 |

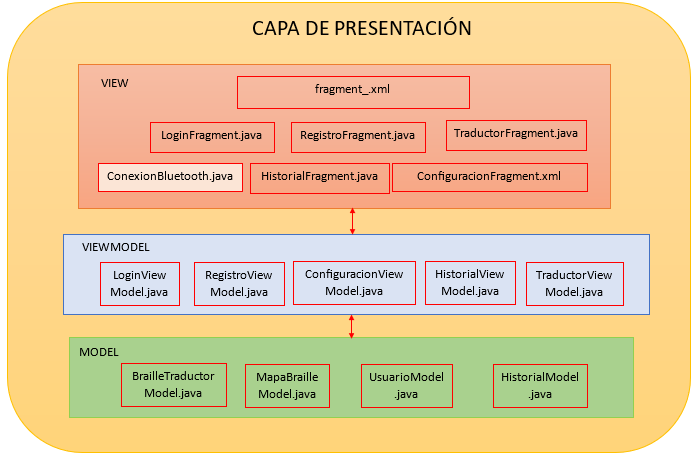
**Visión General de la Arquitectura**

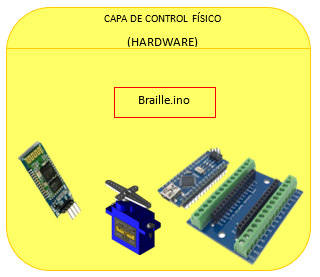
**Estilo Arquitectónico**

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 tres capas principales:

* **Capa de Presentación:** Interfaz gráfica e interacción directa con el usuario.
* **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.

## 

**Diagrama General de Arquitectura:**





**Descripción de Componentes Principales**

**Capa de 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.
* **Views:**
* MainActivity.java: Actividad principal que contiene el menú de navegación lateral (Drawer) y gestiona los fragments.
* TraductorFragment.java/fragment\_traductor.xml: Vista para ingresar texto y visualizar su traducción en braille.
* ConfiguracionFragment.java/fragment\_configuracion.xml: Interfaz para realizar personalizaciones como temas, y color de botones.
* LoginFragment.java/fragment\_login.xml: Pantalla de inicio de sesión con validación de usuario.
* RegistroFragment.java/fragment\_registro.xml: Formulario para el registro de nuevos usuarios.
* HistorialFragment.java/ fragment\_historial.xml: Muestra las traducciones realizadas recientemente.
* fragment\_splash.xml: Diseño de presentación inicial muestra el logo y el slogan.
* fragment\_diccionario.xml: Interfaz donde muestra el alfabeto Braille por imágenes.
* ConexiónBluetooth.java: Servicio de comunicación con Arduino a través de Bluetooth.
* **ViewModel:**
* TraductorViewModel.java: Se encarga de la conversión de texto a Braille mediante el MapaBraille.java.
* ConfiguracionViewModel.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.
* HistorialViewModel.java: Obtiene los datos desde la base de datos y los presenta al usuario.
* **Model:**
* BrailleTraductorModel.java: Modelo encargado de ejecutar la conversión de texto a braille.
* MapaBrailleModel.java: Diccionario interno que mapea caracteres a su representación braille.
* UsuarioModel.java: Clase entidad para el usuario (nombre, apellidos, DNI, etc.).
* HistorialModel.java: Clase entidad para elementos del historial.

**Capa de Persistencia de Datos (Firebase)**

* **Sistema Gestor:** Firebase
* **Componentes:**
* UsuarioRepositorio.java: Gestiona el acceso a Firebase para operaciones de creación, autenticación y recuperación de usuarios.
* HistorialRepositorio.java: Guarda y obtiene el historial de traducciones por usuario.
* ConfiguracionRepositorio.java: Administra las preferencias de usuario y el estado de sesión localmente utilizando SharedPreferences.

**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.

**Integraciones Externas (Opcionales)**

* Implementación de un módulo para traducir documentos completos.
* Traducción bidireccional (texto a Braille y Braille a texto)
* Implementación de OCR
* Compatibilidad con dispositivos de impresión Braille
* Acceso a bibliotecas digitales

**Seguridad**

La aplicación BrailleExpress incorpora mecanismos fundamentales para proteger tanto la integridad como la privacidad de los datos de los usuarios y el correcto funcionamiento del sistema:

* Protección de datos: Los datos del usuario (como credenciales y configuraciones) se almacenan de forma segura en la nube mediante Firebase.
* Privacidad de contenido: Por principios de privacidad, el contenido textual traducido no se almacena en servidores; únicamente se guardan configuraciones de usuario y registros mínimos del historial local.
* Comunicación Bluetooth segura: La conexión con el módulo HC-06 está restringida a dispositivos previamente emparejados, evitando accesos no autorizados. Además, se implementa un control de flujo que impide el envío de datos fuera de sesiones activas.
* Validación de entradas: Todos los campos de texto e interacciones del usuario son validados previamente, previniendo errores operativos o posibles intentos de inyección de comandos maliciosos.
* Gestión de permisos: La aplicación solicita únicamente los permisos estrictamente necesarios, en conformidad con la versión del sistema operativo, y ofrece explicaciones contextuales al usuario para mayor transparencia.

**Escalabilidad y Despliegue**

Aplicación móvil:

* Compatibilidad: Desarrollada para dispositivos con Android 7.0 (Nougat) o superior, asegurando compatibilidad con una amplia gama de terminales, incluso aquellos de gama media o baja.
* Despliegue flexible: Puede distribuirse en formato APK o a través de la Google Play Store, permitiendo actualizaciones automáticas y control de versiones.
* Mantenibilidad: La arquitectura MVVM garantiza una alta cohesión y bajo acoplamiento entre componentes, lo que facilita la implementación de mejoras, nuevas funcionalidades y la corrección de errores sin comprometer la estabilidad general de la aplicación.

Dispositivo físico (Arduino + Servos):

* Modularidad del hardware: El dispositivo está compuesto por componentes ampliamente disponibles (Arduino Nano, módulo Bluetooth HC-06, servos SG90 y Shield IO), lo que permite su fácil reemplazo, réplica y mantenimiento.
* Replicabilidad educativa: Esta solución puede adaptarse a diferentes contextos escolares, centros de rehabilitación o instituciones inclusivas, al estar pensada como un kit de bajo costo y sencilla implementación.

Posibles fallas técnicas:

* Alcance limitado del HC-06: El módulo Bluetooth clásico ofrece un alcance de aproximadamente 10 metros, lo que lo hace ideal para ambientes controlados o uso de escritorio.
* Limitaciones del Arduino Nano: Dada su capacidad limitada, se delega la lógica principal a la aplicación móvil, reservando al Arduino funciones simples de control físico.
* Latencia de transmisión: El envío carácter por carácter puede introducir demoras perceptibles, mitigables mediante la implementación de buffers o protocolos de codificación más eficientes.
* Compatibilidad con permisos: Android 7 permite gestionar permisos en la instalación o mediante solicitudes en tiempo real, lo que facilita la implementación de funcionalidades sin complejas dependencias del sistema.

**Diagrama Entidad-Relación (ERD)**

**Entidades y sus Atributos**

1. **TbUsuarios**

* Id\_DNI (PK)
* Nombre
* Apellidos
* Edad
* Genero
* Contraseña
* Fecha\_Registro

1. **TbHistorial**

* id\_historial (PK)
* Texto\_Original
* Traduccion\_Braille
* Fecha\_Hora
* id\_DNI (FK, referencia a Usuarios)

1. **TbConfiguracion**

* id\_DNI (FK, referencia a Usuarios)
* mantener\_sesion
* Tema
* Color\_boton

**Relaciones**

* Un usuario puede tener múltiples registros de traducciones, lo que establece una relación uno a muchos (1:N) entre TbUsuarios e TbHistorial.
* Un usuario puede tener una única configuración visual personalizada. Por lo tanto, existe una relación uno a uno (1:1) entre TbUsuarios y TbConfiguracion.

**Reglas y Restricciones**

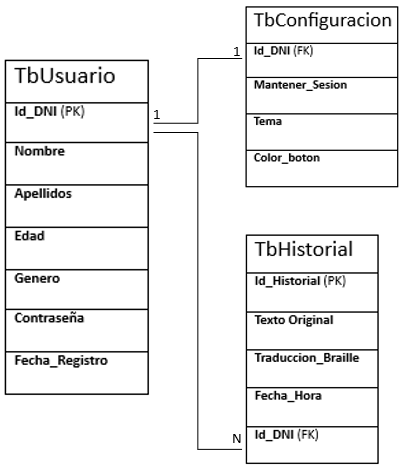
Para garantizar la integridad y seguridad de los datos, se establecen las siguientes reglas:

* Id\_DNI en Usuarios debe ser único, tener exactamente 8 dígitos numéricos y no repetirse.
* Contraseña debe ser almacenada en forma cifrada.
* En TbConfiguracion, no se permite insertar más de una configuración por usuario.
* Un historial de traducción no puede registrarse si no existe el usuario correspondiente.
* No se puede eliminar un usuario si tiene historial asociado (a menos que se elimine primero el historial).
* Texto\_Original en Historial no puede estar vacío.
* Traduccion\_Braille debe ser generada por el motor de traducción.
* Mantener\_Sesion acepta solo valores booleanos (true o false).
* Fecha\_Registro debe asignarse automáticamente al crear un usuario.

**Diagrama de Base de Datos (E/R)**

* + Relación (1) → (1) entre Usuarios y Configuracion, por Id\_DNI.
  + Relación (1) → (N) entre Usuarios e Historial, por Id\_DNI.

**Diagrama Relacional**



**Diccionario de Datos**

**TbUsuarios**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de Dato | Descripción | Restricciones |
| id\_DNI | CHAR(8) | DNI identificador único del usuario. | PRIMARY KEY, NOT NULL, UNIQUE |
| Nombre | VARCHAR(50) | Nombre del usuario. | NOT NULL |
| Apellidos | VARCHAR(50) | Apellido del usuario. | NOT NULL |
| Edad | INT | Edad del usuario. | MAYOR O IGUAL A 0 |
| Género | ENUM | Género seleccionado por usuario. | VALORES POSIBLES: MASCULINO, FEMENINO |
| Contraseña | VARCHAR(100) | Contraseña cifrada del usuario. | NOT NULL |
| Fecha\_Registro | DATE | Fecha y hora en que el usuario se registró. | DEFECT |

**TbHistorial**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de Dato | Descripción | Restricciones |
| id\_Historial | INT (20) | Identificador único de la personalización. | PRIMARY KEY, AUTO\_INCREMENT |
| Texto\_Original | STRING | |  | | --- | |  |  |  | | --- | | Texto ingresado por el usuario | | NOT NULL |
| Traduccion\_Braille | STRING | Resultado de la traducción en sistema Braille | NOT NULL |
| Fecha\_Hora | DATE | Fecha y hora de traducción | DEFECT |
| id\_DNI | CHAR(8) | DNI identificador del usuario propietario. | FOREIGN KEY (Usuarios) |

**TbConfiguracion**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de Dato | Descripción | Restricciones |
| Mantener\_Sesion | BOOLEAN | Indica si desea mantener la sesión iniciada | TRUE, FALSE |
| Tema | ENUM | Tema seleccionado para la interfaz | VALORES POSIBLES: CLARO, OSCURO |
| Color\_Boton | ENUM | Color seleccionado para los botones. | VALORES POSIBLES: ROJO, AZUL, VERDE, MORADO, ROSADO, AMARILLO, CELESTE |
| id\_DNI | CHAR(8) | DNI identificador del usuario propietario. | FOREIGN KEY (Usuarios) |

**Relaciones y Restricciones:**

**Relaciones:**

* Usuarios – Configuración (1:1)

Cada usuario tiene una única configuración de personalización. Esta relación se representa mediante el atributo Id\_DNI como clave foránea en la TbConfiguracion.

* Un usuario no puede tener más de una configuración.
* Cada configuración pertenece exclusivamente a un usuario.
* Usuarios – Historial (1:N)

Un usuario puede realizar múltiples traducciones, generando varios registros en la tabla Historial. La relación se establece a través del campo Id\_DNI como clave foránea.

* Cada historial debe pertenecer a un usuario existente.
* La eliminación de un usuario requiere, previamente, la eliminación de sus historiales asociados.

**Restricciones:**

* Integridad de claves:
* Id\_DNI en Usuarios es clave primaria y no debe repetirse.
* Id\_DNI en Configuracion e Historial debe coincidir con un Id\_DNI existente en Usuarios (clave foránea).
* Restricciones de unicidad:
* No se permite más de un registro de configuración por usuario.
* Cada id\_historial debe ser único.
* Restricciones de integridad semántica:
* Edad debe ser mayor o igual a 0.
* Texto\_Original no puede estar vacío.
* Mantener\_Sesion solo acepta valores booleanos (true o false).
* Restricciones de seguridad:
* Contraseña debe ser almacenada de forma cifrada.
* No se permite el registro de usuarios sin un DNI válido (8 dígitos).

**Consideraciones de Escalabilidad**

La base de datos de BrailleExpress ha sido diseñada con un enfoque modular y flexible, permitiendo una futura ampliación funcional o de volumen de datos sin afectar la estructura existente.

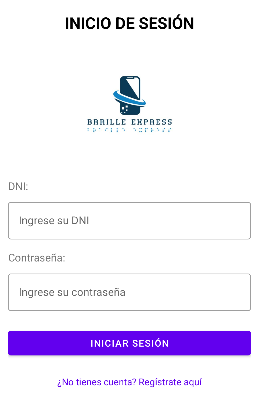
* **Escalabilidad funcional**
* Modularidad: La separación de entidades permite agregar nuevas funcionalidades sin modificar las tablas existentes. Por ejemplo:
* Se puede incluir una nueva tabla PreferenciasAvanzadas si se añaden más opciones visuales.
* Se puede agregar una tabla DispositivosEmparejados para manejar múltiples conexiones Bluetooth por usuario.
* **Escalabilidad en volumen**
* La tabla Historial puede almacenar un número creciente de registros por usuario sin impactar el rendimiento, ya que cada historial está relacionado de forma simple mediante una clave foránea (Id\_DNI).
* La estructura permite indexar campos como Fecha\_Hora para mejorar las consultas de historial por periodo.
* **Escalabilidad en infraestructura**
* Al implementarse en Firebase (plataforma en la nube), se garantiza:
* Escalado automático de la base de datos.
* Alta disponibilidad sin necesidad de servidores físicos.
* Sincronización en tiempo real entre la app y los datos.

**Lineamientos de Diseño**

* **Diseño centrado en el usuario**: Interfaz limpia, accesible y fácil de navegar.
* **Responsive**: Compatible con dispositivos móviles, tabletas y escritorios.
* **Consistencia visual**: Uso de estilos unificados (colores, tipografías, botones).
* **Elementos reutilizables**: Componentes UI modulares para mayor mantenibilidad.

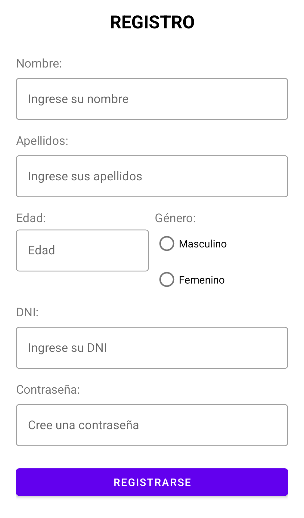
**Wireframes por Pantalla**

**Pantalla de Inicio de Sesión**

* Campos: DNI, Contraseña
* Botón: Iniciar sesión
* Enlace: "¿No tienes cuenta? Regístrate aquí”

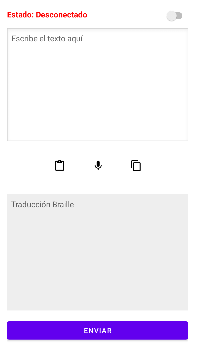
**Pantalla de Registro**

* Campos: Nombre, Apellidos, Edad, Genero, DNI, Contraseña.
* Botón: Registrarse



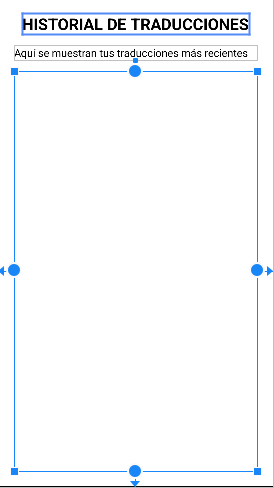
**Traductor**

* Switch: Conexión Bluetooth
* Campos: Escribir texto, salida de texto
* Botón: Pegar, Micrófono, Pegar y enviar



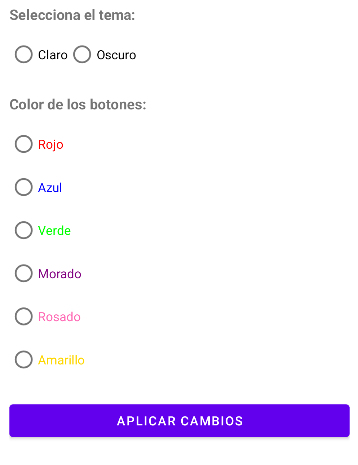
**Historial**

* RecyclerView: Lista de traducciones anteriores y se muestra en lista.



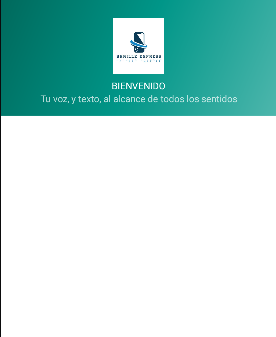
**Configuración**

* RadioGroup: Selección de tema, y color de botones
* RadioButton: Claro y Oscuro, Rojo, Azul, Verde, Morado, Rosado y Amarillo determinadamente.
* Botón: Aplicar cambios



**Ventana de Navegación**

* Sidebar: Navegación (Traductor, Historial y Configuración)
* Button: Cerrar Sesión
* Header: Logo, slogan de la app.



**Prototipo de Navegación**

1. Usuario accede a pantalla de inicio de sesión.
2. Si no tiene cuenta, puede registrarse.
3. Tras iniciar sesión, accede al panel principal que viene a ser el Traductor.
4. Desde el panel puede realizar la conexión a bluetooth y realizar traducciones a Braille estas pueden ser enviadas al Arduino Nano.
5. Accede al diccionario de imágenes de Braille.
6. Accede al historial para revisar las traducciones anteriores.
7. Accede a configuración para personalizar su experiencia como cambio de tema y cambio de botones.

**Guía de Estilo UI (Diseño Visual)**

**Tipografía**

* Títulos principales: 24sp, bold ("INICIO DE SESIÓN")
* Etiquetas de campos: 16sp regular ("DNI:", "Contraseña:")
* Texto en botones: 14-16sp medium
* Texto en inputs: 14-16sp regular
* Enlaces/Texto secundario: 14-16sp regular ("¿No tienes cuenta? Regístrate aquí")

**Paleta de Colores**

Basado en los colores usados (@color/purple\_500, @color/black, @color/white):

* Color primario: purple\_500 (usado en botones)
* Color de texto principal: black
* Color de texto en botones: white
* Color de enlaces/accent: purple\_500
* Fondo: white

**Botones**

* Forma: Rectangular con bordes redondeados (8dp de radio).
* Altura: 48dp (tamaño estándar para accesibilidad).
* Color de fondo: Púrpura primario.
* Color de texto: Blanco.
* Efectos: Sombra ligera (opcional) y cambio de color al presionar.

**Inputs (EditText)**

1. Etiqueta (ej. "DNI:"):
   * Encima del campo, en negro/gris oscuro.
2. Contenedor (TextInputLayout):
   * Borde delgado con esquinas redondeadas (8dp).
   * Color del borde: Gris en estado normal, púrpura al estar activo.
3. Campo de texto (TextInputEditText):
   * Texto en negro.
   * Hint (texto de ejemplo) en gris.

**Diseño General de los Fragments**

* Padding general: 24dp (como en el ConstraintLayout)
* Margen entre elementos:
  + Entre título y logo: 16dp
  + Entre logo y primer campo: 32dp
  + Entre campos: 16dp
  + Entre último campo y botón: 24dp
  + Entre botón y enlace: 24dp
* Alineación: Centrado horizontalmente (constraintStart/End\_toStart/EndOf="parent")
* Imágenes/Logo: Tamaño fijo (150x165dp) con margen superior de 88dp.

**Componentes Frontend (HTML/CSS base)**

**Button**

<Button

        android:id="@+id/btnLogin"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:layout\_marginTop="24dp"

        android:backgroundTint="@color/purple\_500"

        android:text="INICIAR SESIÓN"

        app:layout\_constraintTop\_toBottomOf="@+id/passwordLayout"

        tools:layout\_editor\_absoluteX="24dp"

        android:textColor="@color/white" />

**TextField**

 <com.google.android.material.textfield.TextInputEditText

        android:id="@+id/inputDni"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:hint="Ingrese su DNI"

        android:inputType="number"

        android:maxLength="8"/>

**EditText**

<EditText  
 android:id="@+id/txtEntrada"  
 android:layout\_width="367dp"  
 android:layout\_height="232dp"  
 android:layout\_marginTop="16dp"  
 android:background="@android:drawable/edit\_text"  
 android:gravity="start|top"  
 android:hint="Escribe el texto aquí"  
 android:inputType="textMultiLine"  
 android:padding="10dp"  
 android:scrollbars="vertical"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/txtEstadoBluetooth" />

**TextView**

<TextView  
 android:id="@+id/txtSalida"  
 android:layout\_width="368dp"  
 android:layout\_height="234dp"  
 android:layout\_marginTop="12dp"  
 android:background="#EEEEEE"  
 android:gravity="start|top"  
 android:hint="Traducción Braille"  
 android:padding="10dp"  
 android:scrollbars="vertical"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/layoutTraductores" />

**Consideraciones Finales**

* Accesibilidad como prioridad: La interfaz fue diseñada considerando la inclusión de personas con discapacidad visual, permitiendo una experiencia más accesible y personalizada.
* Diseño modular y claro: Se utilizaron fragments para estructurar la app, facilitando una navegación intuitiva y coherente entre secciones como Home y Personalización.
* Opciones de personalización: El usuario puede seleccionar entre modo claro y oscuro, así como elegir el color de botones, mejorando la usabilidad y adaptabilidad a diferentes preferencias visuales.
* Estilo visual consistente: Se aplicó una guía de estilo con tipografía legible, colores contrastantes y botones accesibles, respetando principios de diseño inclusivo y moderno.
* Interacción centrada en el usuario: Los elementos de entrada, como el campo de texto y los botones de acción (copiar, pegar, micrófono), se colocaron de forma estratégica para facilitar la interacción.
* Base para mejoras futuras: El diseño actual sienta las bases para seguir implementando funciones adicionales, como lectores de pantalla o ajustes avanzados de accesibilidad.
* Fomento de la autonomía: La app busca empoderar al usuario, especialmente a personas con discapacidad visual, promoviendo la comprensión lectora en Braille de forma práctica y tecnológica

**Lineamientos de Diseño Sprint 1**

Desarrollar una primera versión funcional de la aplicación Braille Express que permita a los usuarios:

* Diseño de interfaces de registro y login.
* Registro seguro de usuarios (con almacenamiento en Firebase Realtime Database).
* Validación de DNI.

**Historias de Usuario y Tareas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| HU-1 | Registro de usuario | 5 pts | 21/04/2025 | 29/04/2025 |
| HU-2 | Inicio y cierre de sesión en la nube | 8 pts | 29/04/2025 | 30/04/2025 |

**Evidencias de Código**

**Historia de Usuario 1 — Registro de Usuario**

Archivo: UsuarioModel.java

public class UsuarioModel {

// Campos del modelo

private String nombre;

private String apellido;

private int edad;

private String genero;

private String dni;

private String password;

public UsuarioModel() {

}

//Firebase Realtime Database necesita getters y setters para mapear automáticamente los objetos Java ↔ JSON

public UsuarioModel(String nombre, String apellido, int edad, String genero, String dni, String password) {

this.nombre = nombre;

this.apellido = apellido;

this.edad = edad;

this.genero = genero;

this.dni = dni;

this.password = password;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getApellido() {

return apellido;

}

public void setApellido(String apellido) {

this.apellido = apellido;

}

public int getEdad() {

return edad;

}

public void setEdad(int edad) {

this.edad = edad;

}

public String getGenero() {

return genero;

}

public void setGenero(String genero) {

this.genero = genero;

}

public String getDni() {

return dni;

}

public void setDni(String dni) {

this.dni = dni;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

Archivo: UsuarioRepositorio.java

public class UsuarioRepositorio {

// Referencia a la colección "usuarios" en Firebase

private final DatabaseReference usuariosRef;

public UsuarioRepositorio() {

FirebaseDatabase database = FirebaseDatabase.getInstance();

usuariosRef = database.getReference("usuarios");

}

//Registra un nuevo usuario en Firebase.

public void registrarUsuario(UsuarioModel usuario, String dni,

String password, OnUsuarioRegistradoListener listener) {

usuario.setPassword(password);

usuariosRef.child(dni).setValue(usuario)

.addOnSuccessListener(aVoid -> listener.onRegistrado())

.addOnFailureListener(listener::onError);

}

//Valida las credenciales de login comparando con la base de datos.

public void validarLogin(String dni, String password, OnLoginValidadoListener listener) {

usuariosRef.child(dni).get().addOnCompleteListener(task -> {

if (task.isSuccessful() && task.getResult().exists()) {

UsuarioModel usuario = task.getResult().getValue(UsuarioModel.class);

if (usuario != null && usuario.getPassword().equals(password)) {

listener.onLoginExitoso(usuario);

} else {

listener.onLoginFallido("Credenciales incorrectas");

}

} else {

listener.onLoginFallido("Usuario no encontrado");

}

}).addOnFailureListener(listener::onError);

}

//Listener para eventos de registro.

public interface OnUsuarioRegistradoListener {

void onRegistrado();

void onError(@NonNull Exception e);

}

//Listener para eventos de login.

public interface OnLoginValidadoListener {

void onLoginExitoso(UsuarioModel usuario);

void onLoginFallido(String mensajeError);

void onError(@NonNull Exception e);

}

}

Archivo: RegistroFragment.java

public class RegistroFragment extends Fragment {

private RegistroViewModel registroViewModel;

private FragmentRegistroBinding binding;

private TextInputEditText inputNombre, inputApellido, inputEdad, inputDNI, inputPassword;

private RadioGroup radioGroupGenero;

private Button btnRegistrar;

@Override

public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

// Configuración del View Binding

binding = FragmentRegistroBinding.inflate(inflater, container, false);

return binding.getRoot();

}

@Override

public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {

super.onViewCreated(view, savedInstanceState);

// Inicialización del ViewModel

registroViewModel = new ViewModelProvider(this).get(RegistroViewModel.class);

// Vinculación de vistas (alternativa con binding)

inputNombre = binding.Nombre;

inputApellido = binding.Apellidos;

inputEdad = binding.Edad;

inputDNI = binding.Dni;

inputPassword = binding.password;

radioGroupGenero = binding.radioGroupGenero;

btnRegistrar = binding.btnRegistrar;

// Evento de clic para registro

btnRegistrar.setOnClickListener(v -> registrarUsuario());

// Observadores del ViewModel

configurarObservadores();

}

//Recopila datos del formulario y solicita registro mediante

private void registrarUsuario() {

// Extracción de datos con validación básica

String nombre = inputNombre.getText().toString().trim();

String apellido = inputApellido.getText().toString().trim();

String edad = inputEdad.getText().toString().trim();

String dni = inputDNI.getText().toString().trim();

String password = inputPassword.getText().toString().trim();

// Validación de género seleccionado

int selectedGeneroId = radioGroupGenero.getCheckedRadioButtonId();

if (selectedGeneroId == -1) {

Toast.makeText(getContext(), "Seleccione un género", Toast.LENGTH\_SHORT).show();

return;

}

String genero = ((RadioButton) getView().findViewById(selectedGeneroId)).getText().toString();

// Envío de datos al ViewModel

registroViewModel.registrarUsuario(nombre, apellido, edad, genero, dni, password);

}

//Configura observadores para eventos del ViewModel:

private void configurarObservadores() {

registroViewModel.getRegistroExitoso().observe(getViewLifecycleOwn r(), exito -> {

if (exito != null && exito) {

Toast.makeText(getContext(), "Registro exitoso", Toast.LENGTH\_SHORT).show();

Navigation.findNavController(requireView()).navigate(R.id.action\_RegistroFragment\_to\_LoginFragment);

}

});

registroViewModel.getMensajeError().observe(getViewLifecycleOwner(), error -> {

if (!TextUtils.isEmpty(error)) {

Toast.makeText(getContext(), error, Toast.LENGTH\_SHORT).show();

}

});

}

@Override

public void onDestroyView() {

super.onDestroyView();

binding = null; // Limpieza del View Binding

}

}

Archivo: RegistroViewModel.java

public class RegistroViewModel extends ViewModel {

// LiveData para comunicación con la UI

private final MutableLiveData<Boolean> registroExitoso = new MutableLiveData<>();

private final MutableLiveData<String> mensajeError = new MutableLiveData<>();

// Provee el estado del registro a la UI

public LiveData<Boolean> getRegistroExitoso() {

return registroExitoso;

}

//Manda mensajes de error a la UI

public LiveData<String> getMensajeError() {

return mensajeError;

}

//Registra un nuevo usuario con validaciones y seguridad básica

public void registrarUsuario(String nombre, String apellido, String edad, String genero, String dni, String password) {

// Validación de campos obligatorios

if (TextUtils.isEmpty(nombre) || TextUtils.isEmpty(apellido) ||

TextUtils.isEmpty(edad) || TextUtils.isEmpty(genero) || TextUtils.isEmpty(dni)) {

mensajeError.setValue("Todos los campos son obligatorios.");

return;

}

// Validación adicional recomendada para producción:

if (password.length() < 6) {

mensajeError.setValue("La contraseña debe tener al menos 6 caracteres.");

return;

}

// Hash de la contraseña con BCrypt

String hashedPassword = BCrypt.withDefaults().hashToString(12, password.toCharArray());

// Configuración de Firebase

DatabaseReference database = FirebaseDatabase.getInstance().getReference("usuarios");

// Estructura de datos del usuario

HashMap<String, Object> datosUsuario = new HashMap<>();

datosUsuario.put("Nombre", nombre);

datosUsuario.put("Apellidos", apellido);

datosUsuario.put("Edad", edad);

datosUsuario.put("Genero", genero);

datosUsuario.put("Dni", dni);

datosUsuario.put("Password", hashedPassword); // Almacena solo el hash

// Operación de escritura en Firebase

database.child(dni).setValue(datosUsuario)

.addOnSuccessListener(unused -> registroExitoso.setValue(true))

.addOnFailureListener(e -> mensajeError.setValue("Error al registrar: " + e.getMessage()));

}

}

Archivo: fragment\_registro.xml

//Nombre

<TextView

android:id="@+id/titleRegister"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="4dp"

android:text="REGISTRO"

android:textColor="@color/black"

android:textSize="24sp"

android:textStyle="bold"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

<TextView

android:id="@+id/labelNombre"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="28dp"

android:text="Nombre:"

android:textSize="16sp"

app:layout\_constraintTop\_toBottomOf="@+id/titleRegister"

tools:layout\_editor\_absoluteX="24dp" />

<com.google.android.material.textfield.TextInputLayout

android:id="@+id/nombreLayout"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toBottomOf="@id/labelNombre"

android:layout\_marginTop="4dp">

<com.google.android.material.textfield.TextInputEditText

android:id="@+id/Nombre"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Ingrese su nombre" />

</com.google.android.material.textfield.TextInputLayout>

//Apellidos

<TextView

android:id="@+id/labelApellidos"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Apellidos:"

android:textSize="16sp"

android:layout\_marginTop="16dp"

app:layout\_constraintTop\_toBottomOf="@id/nombreLayout" />

<com.google.android.material.textfield.TextInputLayout

android:id="@+id/apellidosLayout"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toBottomOf="@id/labelApellidos"

android:layout\_marginTop="4dp">

<com.google.android.material.textfield.TextInputEditText

android:id="@+id/Apellidos"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Ingrese sus apellidos" />

</com.google.android.material.textfield.TextInputLayout>

//Edad y Género

<LinearLayout

android:id="@+id/edadGeneroLayout"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:layout\_marginTop="16dp"

app:layout\_constraintTop\_toBottomOf="@id/apellidosLayout">

<!-- Edad -->

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="start|center\_vertical"

android:layout\_marginEnd="8dp"

android:text="Edad:"

android:textSize="16sp" />

<com.google.android.material.textfield.TextInputEditText

android:id="@+id/Edad"

android:layout\_width="95dp"

android:layout\_height="wrap\_content"

android:hint="Edad"

android:inputType="number" />

</LinearLayout>

<!-- Género -->

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginTop="8dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Género:"

android:textSize="16sp"

android:layout\_gravity="start|center\_vertical"

android:layout\_marginEnd="8dp"/>

<RadioGroup

android:id="@+id/radioGroupGenero"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<RadioButton

android:id="@+id/radioMasculino"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Masculino" />

<RadioButton

android:id="@+id/radioFemenino"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Femenino" />

</RadioGroup>

</LinearLayout>

</LinearLayout>

//DNI

<TextView

android:id="@+id/labelDniReg"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="8dp"

android:text="DNI:"

android:textSize="16sp"

app:layout\_constraintTop\_toBottomOf="@id/edadGeneroLayout"

tools:layout\_editor\_absoluteX="24dp" />

<com.google.android.material.textfield.TextInputLayout

android:id="@+id/dniRegLayout"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toBottomOf="@id/labelDniReg"

android:layout\_marginTop="4dp">

<com.google.android.material.textfield.TextInputEditText

android:id="@+id/Dni"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:inputType="number"

android:hint="Ingrese su DNI"

android:maxLength="8"/>

</com.google.android.material.textfield.TextInputLayout>

//Contraseña

<TextView

android:id="@+id/labelPasswordReg"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Contraseña:"

android:textSize="16sp"

android:layout\_marginTop="16dp"

app:layout\_constraintTop\_toBottomOf="@id/dniRegLayout" />

<com.google.android.material.textfield.TextInputLayout

android:id="@+id/passwordRegLayout"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toBottomOf="@id/labelPasswordReg"

android:layout\_marginTop="4dp">

<com.google.android.material.textfield.TextInputEditText

android:id="@+id/password"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Cree una contraseña"

android:inputType="textPassword"/>

</com.google.android.material.textfield.TextInputLayout>

//Botón Registro

<Button

android:id="@+id/btnRegistrar"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="REGISTRARSE"

android:textColor="@color/white"

android:backgroundTint="@color/purple\_500"

android:layout\_marginTop="24dp"

app:layout\_constraintTop\_toBottomOf="@id/passwordRegLayout" />

//Enlace a login

<TextView

android:id="@+id/linkLogin"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="20dp"

android:text="¿Ya tienes cuenta? Inicia sesión"

android:textColor="@color/purple\_500"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@id/btnRegistrar" />

</androidx.constraintlayout.widget.ConstraintLayout>

**Historia de Usuario 2 — Inicio y cierre de sesión en la nube**

Archivo: fragment\_login.xml

//Título "INICIO DE SESIÓN"

<TextView

android:id="@+id/titleLogin"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="16dp"

android:text="INICIO DE SESIÓN"

android:textColor="@color/black"

android:textSize="24sp"

android:textStyle="bold"

app:layout\_constraintBottom\_toTopOf="@+id/logoApp"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

//Logo de la aplicación

<ImageView

android:id="@+id/logoApp"

android:layout\_width="150dp"

android:layout\_height="165dp"

android:layout\_marginTop="88dp"

android:src="@drawable/logo"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.497"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintTop\_toBottomOf="@id/titleLogin"

android:padding="2dp" />

//TextView DNI y campo para DNI

<TextView

android:id="@+id/labelDni"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="32dp"

android:text="DNI:"

android:textSize="16sp"

app:layout\_constraintTop\_toBottomOf="@id/logoApp"

tools:layout\_editor\_absoluteX="24dp" />

<com.google.android.material.textfield.TextInputLayout

android:id="@+id/dniLayout"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toBottomOf="@id/labelDni"

android:layout\_marginTop="8dp">

<com.google.android.material.textfield.TextInputEditText

android:id="@+id/Dni"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Ingrese su DNI"

android:inputType="number"

android:maxLength="8"/> <!-- Validación de 8 dígitos -->

</com.google.android.material.textfield.TextInputLayout>

//TextView Contraseña y campo para Contraseña

<TextView

android:id="@+id/labelPassword"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="16dp"

android:text="Contraseña:"

android:textSize="16sp"

app:layout\_constraintTop\_toBottomOf="@id/dniLayout"

tools:layout\_editor\_absoluteX="24dp" />

<com.google.android.material.textfield.TextInputLayout

android:id="@+id/passwordLayout"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toBottomOf="@id/labelPassword"

android:layout\_marginTop="8dp">

<com.google.android.material.textfield.TextInputEditText

android:id="@+id/password"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Ingrese su contraseña"

android:inputType="textPassword"/>

</com.google.android.material.textfield.TextInputLayout>

//Botón de Inicio de Sesión

<Button

android:id="@+id/btnLogin"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="24dp"

android:backgroundTint="@color/purple\_500"

android:text="INICIAR SESIÓN"

app:layout\_constraintTop\_toBottomOf="@+id/passwordLayout"

tools:layout\_editor\_absoluteX="24dp"

android:textColor="@color/white" />

//Link de Registro

<TextView

android:id="@+id/linkRegistro"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="¿No tienes cuenta? Regístrate aquí"

android:textColor="@color/purple\_500"

app:layout\_constraintTop\_toBottomOf="@id/btnLogin"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

android:layout\_marginTop="24dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>

**Burndown Chart**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Backlog ID | Historia de Usuario | Tiempo Estimado | 21/04/2025 | 22/04/2025 | 23/04/2025 | 24/04/2025 | 25/04/2025 | 26/04/2025 |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** |
| HU-1 | Registro de usuario | 54 | 10 | 6 | 11 | 8 | 9 | 10 |
| HU-2 | Inicio y cierre de sesión en la nube | 7 |  |  |  | 4 | 2 | 1 |
| Tiempo de trabajo | | 61 | 10 | 6 | 11 | 12 | 11 | 11 |

**Análisis del Sprint**

* Se cumplió con todas las tareas estimadas.
* El progreso fue constante y sin bloqueos mayores.
* Las decisiones técnicas sobre persistencia con Firebase resultaron efectivas para este primer Sprint.
* Se logró modularizar el código y documentarlo en comentarios.
* Las estimaciones casi precisas y se logró un buen ritmo de trabajo.

**Lineamientos de Diseño Sprint 2**

Desarrollar una primera versión funcional de la aplicación Braille Express que permita a los usuarios:

* Diseño de login.
* Inicio de Sesión seguro de usuarios (con almacenamiento en Firebase Realtime Database).
* Validación de usuarios y contraseñas

**Historias de Usuario y Tareas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| HU-1.2 | Inicio y cierre de sesión en la nube | 8 pts | 01/05/2025 | 09/05/2025 |
| HU-2.1 | Ingreso manual de texto | 5 pts | 09/05/2025 | 10/05/2025 |

**Evidencias de Código**

**4.1 Historia de Usuario 1.2 — Inicio y cierre de sesión en la nube**

Archivo: LoginFragment.java

public class LoginFragment extends Fragment {  
  
 private FragmentLoginBinding binding;  
 private LoginViewModel loginViewModel;  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
 binding = FragmentLoginBinding.*inflate*(inflater, container, false);  
 loginViewModel = new ViewModelProvider(this).get(LoginViewModel.class);  
 // Configuración del botón de iniciar sesión  
 binding.btnLogin.setOnClickListener(view -> {  
 String dni = binding.Dni.getText().toString().trim();  
 String password = binding.password.getText().toString().trim();  
 // Verificamos si algún campo está vacío  
 if (dni.isEmpty() || password.isEmpty()) {  
 // Mostramos un mensaje de advertencia si los campos no están completos  
 Toast.*makeText*(getContext(), "Por favor, completa todos los campos", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
 // Llamamos al metodo iniciarSesion del ViewModel para autenticar al usuario  
 loginViewModel.iniciarSesion(dni, password, isExitoso -> {  
 if (isExitoso) {  
 // Llamamos al metodo iniciarSesion del ViewModel para autenticar al usuario  
 NavHostFragment.*findNavController*(LoginFragment.this)  
 .navigate(R.id.*nav\_Traductor*);  
 } else {  
 // Si las credenciales son incorrectas, mostramos un mensaje de error  
 Toast.*makeText*(getContext(), "DNI o contraseña incorrectos", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 });  
 // Configuración del link para ir al fragmento de registro  
 binding.linkRegistro.setOnClickListener(v -> {  
 NavHostFragment.*findNavController*(LoginFragment.this)  
 .navigate(R.id.*nav\_Registro*);  
 });  
 // Retornamos la vista raíz del fragment para que se muestre en la interfaz  
 return binding.getRoot();  
  
 }  
}

Archivo: LoginViewModel.java

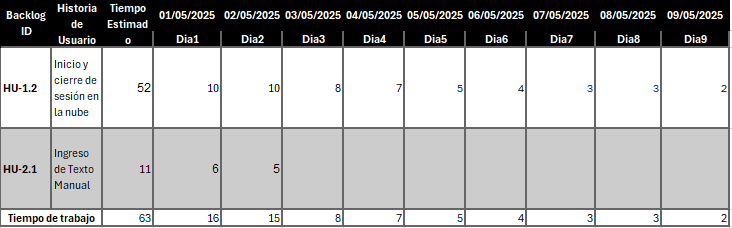
public class LoginViewModel extends ViewModel {  
  
 public interface LoginCallback {  
 void onResult(boolean isExitoso);  
 }  
 // Metodo para iniciar sesión usando el DNI y la contraseña  
 public void iniciarSesion(String dni, String password, LoginCallback callback) {  
 // Usamos Firebase para obtener los datos del usuario desde la base de datos  
 FirebaseDatabase.*getInstance*().getReference("usuarios").child(dni)  
 .addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(DataSnapshot snapshot) {  
 // Verificamos si el usuario existe en la base de datos  
 if (snapshot.exists()) {  
 // Obtenemos la contraseña almacenada en la base de datos para el usuario con el DNI proporcionado  
 String storedPassword = snapshot.child("Password").getValue(String.class);  
 // Si la contraseña almacenada no es nula y coincide con la contraseña ingresada  
 if (storedPassword != null && storedPassword.equals(password)) {  
 callback.onResult(true);  
 } else {  
 callback.onResult(false);  
 }  
 } else {  
 callback.onResult(false);  
 }  
 }  
 // En caso de error al intentar acceder a la base de datos  
 @Override  
 public void onCancelled(DatabaseError error) {  
 callback.onResult(false);  
 }  
 });  
 }  
}

**Historia de Usuario 2.1 — Ingreso manual de texto**

Archivo: fragment\_traductor.xml

//<!-- Estado Bluetooth + Switch -->  
 <TextView  
 android:id="@+id/txtEstadoBluetooth"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="Estado: Desconectado"  
 android:textColor="#FF0303"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/swBluetooth"  
 app:layout\_constraintHorizontal\_bias="0"/>  
  
 <Switch  
 android:id="@+id/swBluetooth"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:onClick="abrirConfiguracionBluetooth"  
 app:layout\_constraintTop\_toTopOf="@id/txtEstadoBluetooth"  
 app:layout\_constraintEnd\_toEndOf="parent" />  
  
 <!-- Caja de texto de entrada -->  
 <EditText  
 android:id="@+id/txtEntrada"  
 android:layout\_width="367dp"  
 android:layout\_height="232dp"  
 android:layout\_marginTop="16dp"  
 android:background="@android:drawable/edit\_text"  
 android:gravity="start|top"  
 android:hint="Escribe el texto aquí"  
 android:inputType="textMultiLine"  
 android:padding="10dp"  
 android:scrollbars="vertical"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/txtEstadoBluetooth" />  
  
 <!-- Botones: Pegar - Micrófono - Copiar -->  
 <LinearLayout  
 android:id="@+id/layoutBotonesTexto"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center"  
 app:layout\_constraintTop\_toBottomOf="@id/txtEntrada"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:layout\_marginTop="12dp">  
  
 <ImageButton  
 android:id="@+id/btnPegar"  
 android:layout\_width="70dp"  
 android:layout\_height="64dp"  
 android:background="@null"  
 android:contentDescription="Pegar texto"  
 android:src="@drawable/ic\_pegar" />  
  
 <ImageButton  
 android:id="@+id/btnHablar"  
 android:layout\_width="85dp"  
 android:layout\_height="match\_parent"  
 android:background="@null"  
 android:contentDescription="Hablar"  
 android:src="@drawable/ic\_micro" />  
  
 <ImageButton  
 android:id="@+id/btnCopiar"  
 android:layout\_width="67dp"  
 android:layout\_height="64dp"  
 android:background="@null"  
 android:contentDescription="Copiar texto"  
 android:src="@drawable/ic\_copiar" />  
 </LinearLayout>  
  
 <!-- Castellano - Traducir - Braille -->  
 <LinearLayout  
 android:id="@+id/layoutTraductores"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center"  
 android:layout\_marginTop="12dp"  
 app:layout\_constraintTop\_toBottomOf="@id/layoutBotonesTexto"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent">  
  
 </LinearLayout>  
  
 <!-- Resultado en Braille -->  
 <TextView  
 android:id="@+id/txtSalida"  
 android:layout\_width="368dp"  
 android:layout\_height="234dp"  
 android:layout\_marginTop="12dp"  
 android:background="#EEEEEE"  
 android:gravity="start|top"  
 android:hint="Traducción Braille"  
 android:padding="10dp"  
 android:scrollbars="vertical"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/layoutTraductores" />  
  
 <!-- Botón Enviar -->  
 <Button  
 android:id="@+id/btnEnviar"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="Enviar"  
 android:backgroundTint="@color/purple\_500"  
 android:textColor="@color/white"  
 app:layout\_constraintTop\_toBottomOf="@id/txtSalida"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:layout\_marginTop="16dp" />  
</androidx.constraintlayout.widget.ConstraintLayout>

**Burndown Chart**



**Análisis del Sprint 2**

* Se logró implementar correctamente la autenticación en la nube con Firebase, incluyendo inicio y cierre de sesión desde múltiples dispositivos.
* Aunque el sprint estaba estimado en 68 horas, se completaron todas las tareas en 63 horas, lo que refleja eficiencia y una mejora en la velocidad del equipo.
* La integración con la nube permite mantener los datos del usuario disponibles y seguros, cumpliendo con los criterios de aceptación de la HU 1.2.
* Se logró diseñar la interfaz de ingreso de texto y dar los primeros pasos técnicos hacia la traducción Braille, anticipando posibles dificultades.
* Tomó tiempo establecer y validar correctamente los permisos y rutas para la autenticación, pero se resolvió satisfactoriamente.
* Al desarrollar la interfaz para usuarios con discapacidad visual, fue necesario realizar ajustes para mejorar la usabilidad y accesibilidad, lo que exigió pruebas adicionales.

**2. Lineamientos de Diseño Sprint 3**

En este Sprint se priorizó desarrollar una primera versión funcional de las siguientes características en Braille Express:

* Validación de caracteres y signos de puntuación en el ingreso manual de texto.
* Integración de la conversión de texto a Braille para el ingreso manual.
* Diseño de la experiencia de usuario para el ingreso de texto por voz.
* Integración de la API de reconocimiento de voz para transcribir el texto hablado.
* Implementación de edición sobre el texto transcrito por voz.
* Realización de pruebas funcionales para asegurar la calidad en ambos métodos de ingreso.

**Historias de Usuario y Tareas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| HU-2.1 | Ingreso manual de texto | 5 pts | 12/05/2025 | 29/04/2025 |
| HU-2.2 | Ingreso de texto por voz | 8 pts | 29/04/2025 | 19/05/2025 |

**Evidencias de Código**

**Historia de Usuario 2.1 — Ingreso manual de texto**

Archivo: MapaBrailleModel.java

public class MapaBrailleModel {  
  
 private final Map<Character, String> mapaBraille;  
  
 public MapaBrailleModel() {  
 mapaBraille = new HashMap<>();  
  
 // Letras (usando representación Unicode Braille ⠁ = U+2801)  
 mapaBraille.put('a', "⠁");  
 mapaBraille.put('b', "⠃");  
 mapaBraille.put('c', "⠉");  
 mapaBraille.put('d', "⠙");  
 mapaBraille.put('e', "⠑");  
 mapaBraille.put('f', "⠋");  
 mapaBraille.put('g', "⠛");  
 mapaBraille.put('h', "⠓");  
 mapaBraille.put('i', "⠊");  
 mapaBraille.put('j', "⠚");  
 mapaBraille.put('k', "⠅");  
 mapaBraille.put('l', "⠇");  
 mapaBraille.put('m', "⠍");  
 mapaBraille.put('n', "⠝");  
 mapaBraille.put('o', "⠕");  
 mapaBraille.put('p', "⠏");  
 mapaBraille.put('q', "⠟");  
 mapaBraille.put('r', "⠗");  
 mapaBraille.put('s', "⠎");  
 mapaBraille.put('t', "⠞");  
 mapaBraille.put('u', "⠥");  
 mapaBraille.put('v', "⠧");  
 mapaBraille.put('w', "⠺");  
 mapaBraille.put('x', "⠭");  
 mapaBraille.put('y', "⠽");  
 mapaBraille.put('z', "⠵");  
  
 // Números (Braille literario usual usa prefijo de número)  
 mapaBraille.put('1', "⠼⠁");  
 mapaBraille.put('2', "⠼⠃");  
 mapaBraille.put('3', "⠼⠉");  
 mapaBraille.put('4', "⠼⠙");  
 mapaBraille.put('5', "⠼⠑");  
 mapaBraille.put('6', "⠼⠋");  
 mapaBraille.put('7', "⠼⠛");  
 mapaBraille.put('8', "⠼⠓");  
 mapaBraille.put('9', "⠼⠊");  
 mapaBraille.put('0', "⠼⠚");  
  
 // Signos de puntuación comunes  
 mapaBraille.put('.', "⠲");  
 mapaBraille.put(',', "⠂");  
 mapaBraille.put(';', "⠆");  
 mapaBraille.put(':', "⠒");  
 mapaBraille.put('?', "⠦");  
 mapaBraille.put('!', "⠖");  
 mapaBraille.put('(', "⠶");  
 mapaBraille.put(')', "⠶");  
 mapaBraille.put('"', "⠦");  
 mapaBraille.put('-', "⠤");  
 mapaBraille.put('\'', "⠄");  
 mapaBraille.put('/', "⠌");  
 mapaBraille.put('@', "⠈");  
 mapaBraille.put(' ', " "); // Espacio  
 }  
  
 */\*\*  
 \* Devuelve la representación en braille de un carácter dado.  
 \*/* public String obtenerBraille(char c) {  
 c = Character.*toLowerCase*(c); // Para que sea case-insensitive  
 return mapaBraille.getOrDefault(c, "⍰"); // Caracter desconocido  
 }

Archivo: BrailleTraductorModel.java

public public class BrailleTraductorModel {  
  
 private MapaBrailleModel mapaBrailleModel;  
  
 public BrailleTraductorModel() {  
 mapaBrailleModel = new MapaBrailleModel();  
 }  
  
 *//Convierte un texto completo a su representación en Braille* public String convertirTextoABraille(String texto) {  
 StringBuilder resultado = new StringBuilder();  
  
 for (char c : texto.toCharArray()) {  
 if (Character.*isUpperCase*(c)) {  
 resultado.append("⠠"); // Prefijo de mayúscula  
 c = Character.*toLowerCase*(c); // Convertir a minúscula para obtener su símbolo Braille  
 }  
  
 String simboloBraille = mapaBrailleModel.obtenerBraille(c);  
 resultado.append(simboloBraille);  
 }  
  
 return resultado.toString();  
 }  
}

Archivo: TraductorFragment.java

public public class TraductorFragment extends Fragment {  
  
 private FragmentTraductorBinding binding;  
 private TraductorViewModel traductorViewModel;  
  
 private static final int *REQUEST\_CODE\_SPEECH\_INPUT* = 1;  
  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater,  
 ViewGroup container, Bundle savedInstanceState) {  
  
 traductorViewModel = new ViewModelProvider(this).get(TraductorViewModel.class);  
 binding = FragmentTraductorBinding.*inflate*(inflater, container, false);  
 View root = binding.getRoot();  
  
 // Traducción automática al escribir  
 binding.txtEntrada.addTextChangedListener(new TextWatcher() {  
 @Override public void beforeTextChanged(CharSequence s, int start, int count, int after) {}  
 @Override public void onTextChanged(CharSequence s, int start, int before, int count) {  
 traductorViewModel.setTextoOriginal(s.toString());  
 }  
 @Override public void afterTextChanged(Editable s) {}  
 });  
  
 // Observa el resultado traducido  
 traductorViewModel.getTextoBraille().observe(getViewLifecycleOwner(), braille -> {  
 binding.txtSalida.setText(braille);  
 });  
  
 // 🎤 Botón de micrófono  
 binding.btnHablar.setOnClickListener(v -> iniciarReconocimientoVoz());  
  
 return root;  
 }  
  
 private void iniciarReconocimientoVoz() {  
 Intent intent = new Intent(RecognizerIntent.*ACTION\_RECOGNIZE\_SPEECH*);  
 intent.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE\_MODEL*,  
 RecognizerIntent.*LANGUAGE\_MODEL\_FREE\_FORM*);  
 intent.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE*, Locale.*getDefault*());  
 intent.putExtra(RecognizerIntent.*EXTRA\_PROMPT*, "Habla ahora...");  
  
 try {  
 startActivityForResult(intent, *REQUEST\_CODE\_SPEECH\_INPUT*);  
 } catch (Exception e) {  
 Toast.*makeText*(getContext(), "Error al iniciar el reconocimiento de voz", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 @Override  
 public void onActivityResult(int requestCode, int resultCode,  
 @Nullable Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 if (requestCode == *REQUEST\_CODE\_SPEECH\_INPUT* && resultCode == Activity.*RESULT\_OK* && data != null) {  
 ArrayList<String> resultados = data.getStringArrayListExtra(RecognizerIntent.*EXTRA\_RESULTS*);  
 if (resultados != null && resultados.size() > 0) {  
 String textoReconocido = resultados.get(0);  
 binding.txtEntrada.setText(textoReconocido); // Esto activa automáticamente la traducción  
 }  
 }  
 }  
  
 @Override  
 public void onDestroyView() {  
 super.onDestroyView();  
 binding = null;  
 }  
}

Archivo: TraductorViewModel.java

public class TraductorViewModel extends ViewModel {  
  
 // LiveData para almacenar y observar el texto original ingresado por el usuario.  
 private final MutableLiveData<String> textoOriginal = new MutableLiveData<>();  
  
 // LiveData para almacenar y observar el texto traducido a Braille.  
 private final MutableLiveData<String> textoBraille = new MutableLiveData<>();  
  
 // Instancia del modelo que contiene la lógica de conversión a Braille.  
 private final BrailleTraductorModel brailleTraductorModel = new BrailleTraductorModel();  
  
 // Metodo público para establecer el texto original desde la vista.  
 public void setTextoOriginal(String texto) {  
 textoOriginal.setValue(texto); // Actualiza el texto original  
 traducirTextoABraille(texto); // Traduce el texto al formato Braille  
 }  
  
 // Metodo para exponer el texto original como LiveData para ser observado por la UI.  
 public LiveData<String> getTextoOriginal() {  
 return textoOriginal;  
 }  
  
 // Metodo para exponer el texto traducido a Braille como LiveData.  
 public LiveData<String> getTextoBraille() {  
 return textoBraille;  
 }  
  
 // Metodo privado que realiza la conversión del texto original a Braille.  
 private void traducirTextoABraille(String texto) {  
 String resultado = brailleTraductorModel.convertirTextoABraille(texto); // Llama al modelo para convertir  
 textoBraille.setValue(resultado); // Actualiza el LiveData con el resultado traducido  
 }  
}

}

Archivo: fragment\_traductor.xml

<!-- Estado Bluetooth + Switch -->  
 <TextView  
 android:id="@+id/txtEstadoBluetooth"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="Estado: Desconectado"  
 android:textColor="#FF0303"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/swBluetooth"  
 app:layout\_constraintHorizontal\_bias="0"/>  
  
 <Switch  
 android:id="@+id/swBluetooth"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:onClick="abrirConfiguracionBluetooth"  
 app:layout\_constraintTop\_toTopOf="@id/txtEstadoBluetooth"  
 app:layout\_constraintEnd\_toEndOf="parent" />  
  
 <!-- Caja de texto de entrada -->  
 <EditText  
 android:id="@+id/txtEntrada"  
 android:layout\_width="367dp"  
 android:layout\_height="232dp"  
 android:layout\_marginTop="16dp"  
 android:background="@android:drawable/edit\_text"  
 android:gravity="start|top"  
 android:hint="Escribe el texto aquí"  
 android:inputType="textMultiLine"  
 android:padding="10dp"  
 android:scrollbars="vertical"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/txtEstadoBluetooth" />  
  
 <!-- Botones: Pegar - Micrófono - Copiar -->  
 <LinearLayout  
 android:id="@+id/layoutBotonesTexto"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center"  
 app:layout\_constraintTop\_toBottomOf="@id/txtEntrada"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:layout\_marginTop="12dp">  
  
 <ImageButton  
 android:id="@+id/btnPegar"  
 android:layout\_width="70dp"  
 android:layout\_height="64dp"  
 android:background="@null"  
 android:contentDescription="Pegar texto"  
 android:src="@drawable/ic\_pegar" />  
  
 <ImageButton  
 android:id="@+id/btnHablar"  
 android:layout\_width="85dp"  
 android:layout\_height="match\_parent"  
 android:background="@null"  
 android:contentDescription="Hablar"  
 android:src="@drawable/ic\_micro" />  
  
 <ImageButton  
 android:id="@+id/btnCopiar"  
 android:layout\_width="67dp"  
 android:layout\_height="64dp"  
 android:background="@null"  
 android:contentDescription="Copiar texto"  
 android:src="@drawable/ic\_copiar" />  
 </LinearLayout>  
  
 <!-- Castellano - Traducir - Braille -->  
 <LinearLayout  
 android:id="@+id/layoutTraductores"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center"  
 android:layout\_marginTop="12dp"  
 app:layout\_constraintTop\_toBottomOf="@id/layoutBotonesTexto"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent">  
  
 </LinearLayout>  
  
 <!-- Resultado en Braille -->  
 <TextView  
 android:id="@+id/txtSalida"  
 android:layout\_width="368dp"  
 android:layout\_height="234dp"  
 android:layout\_marginTop="12dp"  
 android:background="#EEEEEE"  
 android:gravity="start|top"  
 android:hint="Traducción Braille"  
 android:padding="10dp"  
 android:scrollbars="vertical"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/layoutTraductores" />  
  
 <!-- Botón Enviar -->  
 <Button  
 android:id="@+id/btnEnviar"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="Enviar"  
 android:backgroundTint="@color/purple\_500"  
 android:textColor="@color/white"  
 app:layout\_constraintTop\_toBottomOf="@id/txtSalida"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:layout\_marginTop="16dp" />  
</androidx.constraintlayout.widget.ConstraintLayout>

**Historia de Usuario 2.2 — Ingreso de Texto por voz**

Archivo: TraductorFragment.xml

// Botón de micrófono  
 binding.btnHablar.setOnClickListener(v -> iniciarReconocimientoVoz());  
  
 return root;  
}  
  
private void iniciarReconocimientoVoz() {  
 Intent intent = new Intent(RecognizerIntent.*ACTION\_RECOGNIZE\_SPEECH*);  
 intent.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE\_MODEL*,  
 RecognizerIntent.*LANGUAGE\_MODEL\_FREE\_FORM*);  
 intent.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE*, Locale.*getDefault*());  
 intent.putExtra(RecognizerIntent.*EXTRA\_PROMPT*, "Habla ahora...");  
  
 try {  
 startActivityForResult(intent, *REQUEST\_CODE\_SPEECH\_INPUT*);  
 } catch (Exception e) {  
 Toast.*makeText*(getContext(), "Error al iniciar el reconocimiento de voz", Toast.*LENGTH\_SHORT*).show();  
 }  
}

**Burndown Chart**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Backlog ID** | **Historia de Usuario** | | **Tiempo Estimado** | | **12/05/2025** | | **13/05/2025** | | **14/05/2025** | | **15/05/2025** | | **16/05/2025** | | **17/05/2025** | | **19/05/2025** | |
| **Dia1** | | **Dia2** | | **Dia3** | | **Dia4** | | **Dia5** | | **Dia6** | | **Dia7** | |
| **HU-2.1** | Ingreso manual de texto | | 36 | | 10 | | 7 | | 6 | | 5 | | 6 | | 2 | | 0 | |
| **HU-2.2** | Ingreso de texto por voz | | 21 | | 1 | | 2 | | 3 | | 5 | | 2 | | 5 | | 3 | |
| **Tiempo de trabajo** | | 57 | | 11 | | 9 | | 9 | | 10 | | 8 | | 7 | | 3 | |

**Análisis del Sprint 3**

* Se completaron todas las tareas planificadas para este Sprint 3, cumpliendo el objetivo de 68 horas de trabajo.
* El equipo mantuvo un ritmo constante y eficiente, sin bloqueos significativos.
* La integración de la API de reconocimiento de voz se realizó con éxito, mejorando notablemente la interacción del usuario.
* La validación de texto ingresado manualmente fue robusta, incluyendo soporte para caracteres y signos de puntuación esenciales.
* Las pruebas funcionales demostraron la estabilidad de las nuevas funcionalidades, con ajustes menores realizados durante el Sprint.

La modularización del código se mantuvo, facilitando la escalabilidad y mantenibilidad.

**Lineamientos de Diseño**

En este Sprint se definieron criterios específicos para asegurar una experiencia de usuario fluida, accesible y coherente con los módulos anteriores. Se priorizó lo siguiente:

* **Automatización del flujo de trabajo**:   
  • El sistema detecta de manera automática el final de la entrada de texto para iniciar la traducción a Braille sin requerir acciones adicionales.   
  • La conversión se muestra en la interfaz de manera inmediata tras la entrada del usuario.
* **Escalabilidad y estructura de almacenamiento**:   
  •Se diseñó un esquema de base de datos en la nube orientado a guardar traducciones por usuario, preparando el sistema para la gestión de historiales personalizados.

**3. Historias de Usuario y Tareas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| HU-2.2 | Ingreso de texto por voz | 8 pts | 29/04/2025 | 19/05/2025 |
| HU-2.3 | Traducción automática a Braille | 8 pts | 29/04/2025 | 19/05/2025 |
| HU-2.4 | Historial de Traducciones | 8 pts | 29/04/2025 | 19/05/2025 |

**4. Evidencias de Código**

**4.1 Historia de Usuario 2.2: Ingreso de texto por voz**

Archivo: fragment\_traductor.xml

<ImageButton  
 android:id="@+id/btnHablar"  
 android:layout\_width="85dp"  
 android:layout\_height="match\_parent"  
 android:background="@null"  
 android:contentDescription="Hablar"  
 android:src="@drawable/ic\_micro" />

**4.2 Historia de Usuario 2.3 — Traducción automática a Braille**

Archivo: TraductorFragment.java

private void iniciarReconocimientoVoz() {  
 Intent intent = new Intent(RecognizerIntent.*ACTION\_RECOGNIZE\_SPEECH*);  
 intent.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE\_MODEL*,  
 RecognizerIntent.*LANGUAGE\_MODEL\_FREE\_FORM*);  
 intent.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE*, Locale.*getDefault*());  
 intent.putExtra(RecognizerIntent.*EXTRA\_PROMPT*, "Habla ahora...");  
  
 try {  
 startActivityForResult(intent, *REQUEST\_CODE\_SPEECH\_INPUT*);  
 } catch (Exception e) {  
 Toast.*makeText*(getContext(), "Error al iniciar el reconocimiento de voz", Toast.*LENGTH\_SHORT*).show();  
 }  
}

**4.3 Historia de Usuario 2.4 — Historial de Traducciones**

Archivo: HistorialModel.java

public HistorialModel() {  
 }  
  
 public HistorialModel(String textoOriginal, String traduccionBraille, String fechaHora, String idDNI) {  
 this.textoOriginal = textoOriginal;  
 this.traduccionBraille = traduccionBraille;  
 this.fechaHora = fechaHora;  
 this.idDNI = idDNI;  
 }  
  
 public String getTextoOriginal() {  
 return textoOriginal;  
 }  
  
 public void setTextoOriginal(String textoOriginal) {  
 this.textoOriginal = textoOriginal;  
 }  
  
 public String getTraduccionBraille() {  
 return traduccionBraille;  
 }  
  
 public void setTraduccionBraille(String traduccionBraille) {  
 this.traduccionBraille = traduccionBraille;  
 }  
  
 public String getFechaHora() {  
 return fechaHora;  
 }  
  
 public void setFechaHora(String fechaHora) {  
 this.fechaHora = fechaHora;  
 }  
  
 public String getIdDNI() {  
 return idDNI;  
 }  
  
 public void setIdDNI(String idDNI) {  
 this.idDNI = idDNI;  
 }  
}

**5. Burndown Chart**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Backlog ID** | **Historia de Usuario** | **Tiempo Estimado** | **20/05/2025** | **21/05/2025** | **22/05/2025** | **23/05/2025** | **24/05/2025** | **25/05/2025** |  | **26/05/2025** | **27/05/2025** |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** |  | **Dia7** | **Dia8** |
| **HU-2.2** | Ingreso de texto por voz | 12 | 7 | 5 |  |  |  |  |  |  |  |
| **HU-2.3** | Traducción automática a Braille | 35 | 2 | 3 | 5 | 6 | 6 | 6 |  | 5 | 2 |
| **HU-2.4** | Historial de Traducciones | 8 |  |  |  |  |  |  |  |  | 8 |
| **Tiempo de trabajo** | | 55 | 9 | 8 | 5 | 6 | 6 | 6 |  | 5 | 10 |

**6. Análisis del Sprint**

* Se completaron las 6 tareas planificadas dentro del Sprint 4, con un esfuerzo estimado de 65 horas y una ejecución real de 55 horas, lo que demuestra una eficiencia destacable por parte del equipo.
* Las tareas cubrieron los siguientes aspectos:   
  -Diseño de interfaz y edición del texto transcrito por voz (HI 2.2).   
  -Implementación de la conversión automática a Braille, incluyendo la detección del fin de entrada, el procesamiento en segundo plano y la presentación inmediata en la interfaz (HI 2.3).   
  -Diseño inicial del esquema de base de datos para guardar traducciones en la nube (HI 2.4).
* La integración de la conversión automática se realizó con éxito, reduciendo los pasos para el usuario y mejorando la experiencia general.
* Las pruebas iniciales revelaron algunas variaciones en la precisión del reconocimiento de voz, las cuales fueron corregidas mediante mejoras en la normalización del texto transcrito.
* La arquitectura modular del proyecto permitió realizar estas integraciones sin afectar las funciones ya existentes, manteniendo la estabilidad del sistema.

**7. Conclusiones**

El **Sprint 4 de Braille Express** concluyó con éxito al integrar funcionalidades clave para la automatización de la traducción a Braille y al comenzar el diseño del historial de traducciones en la nube. Se logró avanzar significativamente en la experiencia del usuario, quien ahora puede ingresar texto por voz y recibir la conversión a Braille de forma automática, sin necesidad de interacción adicional.   
El equipo técnico mostró un desempeño sólido, completando las tareas en menos tiempo del previsto y adaptándose a los retos que surgieron, especialmente en la sincronización del sistema de reconocimiento de voz.   
La adopción de lineamientos de diseño centrados en la accesibilidad y escalabilidad permitió una implementación coherente y robusta. Las bases sentadas para el almacenamiento en la nube abren paso a futuras mejoras como la gestión personalizada de traducciones y la recuperación de contenido.   
Este Sprint refleja el compromiso del equipo con una experiencia digital inclusiva, funcional y orientada a las necesidades reales del usuario con discapacidad visual, consolidando el avance continuo del proyecto.

Registro Diario de Avances – Sprint 5 Braille Express

**1. Introducción**

El presente documento detalla el seguimiento y desarrollo del **Sprint 5** del proyecto **Braille Express**, centrado en la implementación de funcionalidades esenciales para mejorar la accesibilidad y la conectividad del sistema de traducción digital a Braille. En este Sprint se abordaron seis tareas pertenecientes a dos historias de usuario: HI 2.4 (Historial de traducciones), HI 3.1 (Conexión Bluetooth)

El principal objetivo fue avanzar en la funcionalidad de almacenamiento en la nube para el historial de traducciones, habilitar la detección y conexión Bluetooth con el dispositivo Braille, y establecer la base para la sincronización de datos entre dispositivos. Estas características son fundamentales para garantizar una experiencia continua, personalizada y accesible, especialmente para usuarios con discapacidad visual.

**2. Lineamientos de Diseño**

* **Accesibilidad**: Se priorizaron contrastes altos, botones grandes y compatibilidad con lectores de pantalla para usuarios con discapacidad visual.
* **Historial personalizado**: Las traducciones se almacenan en la nube por usuario (vía Firebase), permitiendo acceso desde distintos dispositivos.
* **Sincronización multi-dispositivo**: Se diseñó la estructura para recuperar automáticamente el historial al iniciar sesión desde cualquier equipo.
* **Bluetooth intuitivo**: Se desarrolló una interfaz para detectar dispositivos cercanos, gestionar el estado de conexión y reconectar automáticamente.
* **Consistencia visual**: Se respetaron las pautas gráficas establecidas en sprints anteriores, manteniendo una experiencia coherente.

**3. Historias de Usuario y Tareas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| HU-2.4 | Hitorial de traducciones | 10 pts | 29/04/2025 | 19/05/2025 |
| HU-3.1 | Conexión Bluetooth | 10 pts | 29/04/2025 | 19/05/2025 |

**4. Evidencias de Código**

**4.1 Historia de Usuario 2.2: Historial de Traducciones**

Archivo: fragment\_historial.xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:padding="24dp"  
 android:background="?android:windowBackground"  
 tools:context=".ui.Historial.HistorialFragment">  
  
 <!-- Título del historial -->  
 <TextView  
 android:id="@+id/titleHistorial"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="HISTORIAL DE TRADUCCIONES"  
 android:textSize="24sp"  
 android:textStyle="bold"  
 android:textColor="?android:textColorPrimary"  
 android:gravity="center"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent" />  
  
 <!-- Descripción -->  
 <TextView  
 android:id="@+id/textDescription"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="12dp"  
 android:text="Aquí se muestran tus traducciones más recientes"  
 android:textSize="16sp"  
 android:textColor="?android:textColorSecondary"  
 app:layout\_constraintTop\_toBottomOf="@id/titleHistorial"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent" />  
  
 <!-- RecyclerView del historial -->  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/recyclerHistorial"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_marginTop="16dp"  
 app:layout\_constraintTop\_toBottomOf="@id/textDescription"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 tools:listitem="@layout/item\_historial" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

Archivo: HistorialFragment.java

public class HistorialFragment extends Fragment {  
  
 private FragmentHistorialBinding binding;  
 private HistorialAdapter historialAdapter;  
 private HistorialViewModel historialViewModel;  
 private TraductorViewModel traductorViewModel;  
  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater,  
 ViewGroup container, Bundle savedInstanceState) {  
  
 binding = FragmentHistorialBinding.*inflate*(inflater, container, false);  
 View root = binding.getRoot();  
  
 // Inicializar ViewModels  
 historialViewModel = new ViewModelProvider(this).get(HistorialViewModel.class);  
 traductorViewModel = new ViewModelProvider(requireActivity()).get(TraductorViewModel.class);  
  
 // Configurar RecyclerView  
 binding.recyclerHistorial.setLayoutManager(new LinearLayoutManager(getContext()));  
  
 historialAdapter = new HistorialAdapter(new ArrayList<>(), new HistorialAdapter.OnItemClickListener() {  
 @Override  
 public void onItemClick(HistorialModel item) {  
 traductorViewModel.setTextoOriginal(item.getTextoOriginal());  
 }  
  
 @Override  
 public void onEliminarClick(HistorialModel item) {  
 historialViewModel.eliminarElemento(item.getKey());  
 Toast.*makeText*(getContext(), "Elemento eliminado", Toast.*LENGTH\_SHORT*).show();  
 // Actualizar la lista después de eliminar  
 SharedPreferences prefs = requireActivity().getSharedPreferences("user\_session", Context.*MODE\_PRIVATE*);  
 String idDNI = prefs.getString("dni\_usuario", "");  
 historialViewModel.cargarHistorial(idDNI);  
 }  
  
 @Override  
 public void onCopiarClick(HistorialModel item) {  
 ClipboardManager clipboard = (ClipboardManager) requireContext().getSystemService(Context.*CLIPBOARD\_SERVICE*);  
 ClipData clip = ClipData.*newPlainText*("texto", item.getTextoOriginal());  
 clipboard.setPrimaryClip(clip);  
 Toast.*makeText*(getContext(), "Texto copiado", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 @Override  
 public void onItemDelete(HistorialModel item) {  
  
 }  
 });  
  
 binding.recyclerHistorial.setAdapter(historialAdapter);  
  
 // Obtener ID del usuario desde SharedPreferences  
 SharedPreferences prefs = requireActivity().getSharedPreferences("user\_session", Context.*MODE\_PRIVATE*);  
 String idDNI = prefs.getString("dni\_usuario", "sin\_dni");  
  
 if (!idDNI.isEmpty()) {  
 historialViewModel.cargarHistorial(idDNI);  
 } else {  
 Toast.*makeText*(getContext(), "⚠️ No se encontró ID del usuario", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 // Observar cambios en LiveData  
 historialViewModel.getHistorial().observe(getViewLifecycleOwner(), historialList -> {  
 historialAdapter.setLista(historialList);  
 });  
  
 return root;  
 }  
  
 @Override  
 public void onDestroyView() {  
 super.onDestroyView();  
 binding = null;  
 }  
}

Archivo: HistorialRepositorio.java

public class HistorialRepositorio {  
  
 private final DatabaseReference historialRef;  
  
 public HistorialRepositorio() {  
 FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
 historialRef = database.getReference("TbHistorial");  
 }  
  
 // Metodo para guardar una nueva traducción  
 public void guardarTraduccion(String textoOriginal, String traduccionBraille, String idDNI) {  
 String idHistorial = historialRef.push().getKey(); // genera ID único  
 String fechaHora = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.*getDefault*()).format(new Date());  
  
 HistorialModel nuevoHistorial = new HistorialModel(  
 textoOriginal,  
 traduccionBraille,  
 fechaHora,  
 idDNI  
 );  
  
 if (idHistorial != null) {  
 historialRef.child(idHistorial).setValue(nuevoHistorial);  
 }  
 }  
  
 // Metodo para obtener el historial de un usuario  
 public void obtenerHistorial(String idDNI, ValueEventListener listener) {  
 historialRef.orderByChild("idDNI").equalTo(idDNI).addListenerForSingleValueEvent(listener);  
 }  
  
 // Nuevo metodo: eliminar historial por clave  
 public void eliminarHistorial(String key) {  
 historialRef.child(key).removeValue();  
 }  
 public void eliminarPorKey(String key) {  
 historialRef.child(key).removeValue();  
 }  
  
}

Archivo: HistorialModel.java

public class HistorialModel {  
 private String textoOriginal;  
 private String traduccionBraille;  
 private String fechaHora;  
 private String idDNI;  
 private String key;  
  
 public HistorialModel() {  
 // 🔧 Firebase necesita este constructor vacío  
 }  
  
 public HistorialModel(String textoOriginal, String traduccionBraille, String fechaHora, String idDNI) {  
 this.textoOriginal = textoOriginal;  
 this.traduccionBraille = traduccionBraille;  
 this.fechaHora = fechaHora;  
 this.idDNI = idDNI;  
 }  
  
 public String getTextoOriginal() {  
 return textoOriginal;  
 }  
  
 public void setTextoOriginal(String textoOriginal) {  
 this.textoOriginal = textoOriginal;  
 }  
  
 public String getTraduccionBraille() {  
 return traduccionBraille;  
 }  
  
 public void setTraduccionBraille(String traduccionBraille) {  
 this.traduccionBraille = traduccionBraille;  
 }  
  
 public String getFechaHora() {  
 return fechaHora;  
 }  
  
 public void setFechaHora(String fechaHora) {  
 this.fechaHora = fechaHora;  
 }  
  
 public String getIdDNI() {  
 return idDNI;  
 }  
  
 public void setIdDNI(String idDNI) {  
 this.idDNI = idDNI;  
 }  
  
 public String getKey() {  
 return key;  
 }  
  
 public void setKey(String key) {  
 this.key = key;  
 }  
}

**4.2 Historia de Usuario 3.1 — Conexión Bluetooth**

Archivo: ConexionBluetooth.java

public class ConexionBluetooth {  
  
 private static final UUID *UUID\_SERIAL* = UUID.*fromString*("00001101-0000-1000-8000-00805F9B34FB");  
 private final BluetoothAdapter bluetoothAdapter;  
 private BluetoothSocket bluetoothSocket;  
 private OutputStream outputStream;  
  
 public ConexionBluetooth() {  
 bluetoothAdapter = BluetoothAdapter.*getDefaultAdapter*();  
 }  
  
 @RequiresPermission(allOf = {Manifest.permission.*BLUETOOTH\_CONNECT*, Manifest.permission.*BLUETOOTH\_SCAN*})  
 public boolean conectar(String macAddress, Context context) {  
 if (bluetoothAdapter == null || !bluetoothAdapter.isEnabled()) {  
 Log.*e*("Bluetooth", "Bluetooth no disponible o no habilitado.");  
 return false;  
 }  
  
 try {  
 BluetoothDevice dispositivo = bluetoothAdapter.getRemoteDevice(macAddress);  
 bluetoothSocket = dispositivo.createRfcommSocketToServiceRecord(*UUID\_SERIAL*);  
 bluetoothAdapter.cancelDiscovery();  
 bluetoothSocket.connect();  
 outputStream = bluetoothSocket.getOutputStream();  
  
 Log.*i*("Bluetooth", "Conexión establecida con: " + macAddress);  
 return true;  
  
 } catch (IOException e) {  
 Log.*e*("Bluetooth", "Error al conectar: " + e.getMessage());  
 desconectar();  
 return false;  
 }  
 }  
  
 public void desconectar() {  
 try {  
 if (outputStream != null) outputStream.close();  
 if (bluetoothSocket != null) bluetoothSocket.close();  
 Log.*i*("Bluetooth", "Conexión Bluetooth cerrada.");  
 } catch (IOException e) {  
 Log.*e*("Bluetooth", "Error al cerrar conexión: " + e.getMessage());  
 }  
 }  
  
 public boolean enviarDatos(String datos) {  
 if (outputStream != null) {  
 try {  
 outputStream.write(datos.trim().getBytes());  
 return true;  
 } catch (IOException e) {  
 Log.*e*("Bluetooth", "Error al enviar datos: " + e.getMessage());  
 }  
 }  
 return false;  
 }  
  
 public boolean estaConectado() {  
 return bluetoothSocket != null && bluetoothSocket.isConnected();  
 }  
}

Archivo: fragment\_traductor.java

<TextView  
 android:id="@+id/txtEstadoBluetooth"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="Estado: Desconectado"  
 android:textColor="?android:textColorPrimary"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/swBluetooth"  
 app:layout\_constraintHorizontal\_bias="0"/>  
  
<Switch  
 android:id="@+id/swBluetooth"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="@id/txtEstadoBluetooth" />

**5. Burndown Chart**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Backlog ID** | **Historia de Usuario** | **Tiempo Estimado** | **20/05/2025** | **21/05/2025** | **22/05/2025** | **23/05/2025** | **24/05/2025** | **25/05/2025** | **25/05/2025** |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** | **Dia7** |
| **HU-2.4** | Historial de Traducciones | 30 | 12 | 10 | 8 | 0 | 0 | 0 | 0 |
| **HU-3.1** | Conexión Bluetooth | 42 | 0 | 0 | 1 | 10 | 10 | 12 | 9 |
| **Tiempo de trabajo** | | 72 | 12 | 10 | 9 | 10 | 10 | 12 | 9 |

**6. Análisis del Sprint**

* Se completaron las 6 tareas planificadas dentro del Sprint 5, con un esfuerzo
* La arquitectura modular del proyecto permitió realizar estas integraciones sin afectar las funciones ya existentes, manteniendo la estabilidad del sistema.
* Se planificaron 6 tareas con una duración total de 72 horas, abarcando el historial de traducciones (HI 2.4) y la conexión Bluetooth (HI 3.1).
* Se avanzó en eldiseño y vista del historial**,** lasincronización de datos en Firebase**, y** ladetección/conexión Bluetoothcon el dispositivo Braille.
* El equipo cumplió con las tareas asignadas dentro del tiempo estimado, manteniendo una buena eficiencia y coordinación entre roles.
* Las funcionalidades desarrolladas permiten una experiencia más personalizada,continua y conectada**,** sentando la base para futuras integraciones.

**7. Conclusiones**

El Sprint 5 permitió fortalecer dos pilares fundamentales del proyecto Braille Express: la personalización del historial de traducciones y la interacción directa con el dispositivo físico a través de Bluetooth. Se lograron importantes avances en la experiencia de usuario y la arquitectura técnica del sistema, facilitando la continuidad del uso entre sesiones y dispositivos.

El equipo cumplió con el total de horas asignadas al Sprint, manteniendo una planificación adecuada y un enfoque colaborativo entre las áreas de diseño, desarrollo y pruebas. Las funcionalidades implementadas representan un gran paso hacia una solución integral, inclusiva y tecnológicamente robusta.

Con la base de estas implementaciones, el proyecto se encuentra preparado para cerrar los desarrollos pendientes del historial y avanzar en la gestión completa del hardware, consolidando el ecosistema de traducción digital a Braille pensado para estudiantes con discapacidad visual.

Registro Diario de Avances – Sprint 6 Braille Express

**1. Introducción**

El presente documento describe las actividades ejecutadas durante el **Sprint 6** del proyecto **Braille Express**, centrado en la integración completa entre la aplicación móvil y el dispositivo físico mediante Bluetooth. Este Sprint tuvo como objetivo principal habilitar el envío de caracteres Braille desde la aplicación al dispositivo Arduino, asegurar la estabilidad de la conexión y brindar al usuario control sobre la reconexión manual en caso de fallos.

Se planificaron un total de **8 tareas**, correspondientes a tres historias de usuario clave:

* HI 3.1 (Conexión Bluetooth)
* HI 3.2 (Envío de caracteres a Arduino)
* HI 3.3 (Reinicio manual de Bluetooth)

El Sprint tuvo una duración estimada de **68 horas**, distribuidas entre desarrollo frontend, backend y pruebas de calidad.

**2. Lineamientos de Diseño**

* **Interfaz clara de Bluetooth**: Indicadores visuales de conexión y botón de reconexión accesible.
* **Compatibilidad con Arduino**: Protocolos simples para enviar caracteres sin errores.
* **Estabilidad**: Se priorizó una conexión fluida y control de fallos.
* **Consistencia visual**: Se respetó la estética general de la app, garantizando accesibilidad y claridad.

**3. Historias de Usuario y Tareas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| HU-3.1 | Conexión Bluetooth | 10 pts | 29/04/2025 | 19/05/2025 |
| HU-3.2 | Envío de Caracteres a Arduino | 10 pts | 29/04/2025 | 19/05/2025 |
| HU-3.3 | Reinicio manual de Bluetooth | 5 pts | 29/04/2025 | 19/05/2025 |

**4. Evidencias de Código**

**4.1 Historia de Usuario 3.1: Conexión Bluetooth**

Archivo: fragment\_traductor.java

<TextView  
 android:id="@+id/txtEstadoBluetooth"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="Estado: Desconectado"  
 android:textColor="?android:textColorPrimary"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/swBluetooth"  
 app:layout\_constraintHorizontal\_bias="0"/>  
  
<Switch  
 android:id="@+id/swBluetooth"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="@id/txtEstadoBluetooth" />

Archivo: ConexionBluetooth.java

public class ConexionBluetooth {  
  
 private static final UUID *UUID\_SERIAL* = UUID.*fromString*("00001101-0000-1000-8000-00805F9B34FB");  
 private final BluetoothAdapter bluetoothAdapter;  
 private BluetoothSocket bluetoothSocket;  
 private OutputStream outputStream;  
  
 public ConexionBluetooth() {  
 bluetoothAdapter = BluetoothAdapter.*getDefaultAdapter*();  
 }  
  
 @RequiresPermission(allOf = {Manifest.permission.*BLUETOOTH\_CONNECT*, Manifest.permission.*BLUETOOTH\_SCAN*})  
 public boolean conectar(String macAddress, Context context) {  
 if (bluetoothAdapter == null || !bluetoothAdapter.isEnabled()) {  
 Log.*e*("Bluetooth", "Bluetooth no disponible o no habilitado.");  
 return false;  
 }  
  
 try {  
 BluetoothDevice dispositivo = bluetoothAdapter.getRemoteDevice(macAddress);  
 bluetoothSocket = dispositivo.createRfcommSocketToServiceRecord(*UUID\_SERIAL*);  
 bluetoothAdapter.cancelDiscovery();  
 bluetoothSocket.connect();  
 outputStream = bluetoothSocket.getOutputStream();  
  
 Log.*i*("Bluetooth", "Conexión establecida con: " + macAddress);  
 return true;  
  
 } catch (IOException e) {  
 Log.*e*("Bluetooth", "Error al conectar: " + e.getMessage());  
 desconectar();  
 return false;  
 }  
 }  
  
 public void desconectar() {  
 try {  
 if (outputStream != null) outputStream.close();  
 if (bluetoothSocket != null) bluetoothSocket.close();  
 Log.*i*("Bluetooth", "Conexión Bluetooth cerrada.");  
 } catch (IOException e) {  
 Log.*e*("Bluetooth", "Error al cerrar conexión: " + e.getMessage());  
 }  
 }  
  
 public boolean enviarDatos(String datos) {  
 if (outputStream != null) {  
 try {  
 outputStream.write(datos.trim().getBytes());  
 return true;  
 } catch (IOException e) {  
 Log.*e*("Bluetooth", "Error al enviar datos: " + e.getMessage());  
 }  
 }  
 return false;  
 }  
  
 public boolean estaConectado() {  
 return bluetoothSocket != null && bluetoothSocket.isConnected();  
 }  
}

**4.2 Historia de Usuario 3.2 — Envío de caracteres a Arduino**

Archivo: TraductorFragment.java

// Botón enviar al Arduino  
 btnEnviar.setOnClickListener(v -> {  
 String textoOriginal = txtEntrada.getText().toString();  
 String traduccionBraille = txtSalida.getText().toString();  
  
 if (textoOriginal.isEmpty()) {  
 Toast.*makeText*(requireContext(), "No hay datos para enviar", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 if (conexionBluetooth.estaConectado()) {  
 if (conexionBluetooth.enviarDatos(textoOriginal)) {  
 Toast.*makeText*(requireContext(), "Datos enviados al Arduino", Toast.*LENGTH\_SHORT*).show();  
  
 // Obtener ID del usuario (DNI) desde SharedPreferences  
 String idDNI = requireActivity().getSharedPreferences("user\_session", Context.*MODE\_PRIVATE*)  
 .getString("dni\_usuario", "sin\_dni");  
  
  
 // Guardar en Firebase  
 historialRepositorio.guardarTraduccion(textoOriginal, traduccionBraille, idDNI);  
 Toast.*makeText*(requireContext(), "✅ Guardado en historial", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 Toast.*makeText*(requireContext(), "Error al enviar datos", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 Toast.*makeText*(requireContext(), "Bluetooth no conectado", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 SharedPreferences prefs = requireActivity().getSharedPreferences("user\_session", Context.*MODE\_PRIVATE*);  
 String colorBoton = prefs.getString("color\_boton", "#FF6200EE");  
 try {  
 btnEnviar.setBackgroundColor(Color.*parseColor*(colorBoton));  
 } catch (IllegalArgumentException e) {  
 Log.*e*("ColorError", "Color inválido en configuración: " + colorBoton);  
 }  
  
}

Archivo: BRAILLEEXPRESS.ino

< #include <Servo.h>

#include <SoftwareSerial.h>

SoftwareSerial bluetooth(13, 12); // RX, TX (Conecta TX del HC-06 al pin 13, RX al 12)

Servo servo1;

Servo servo2;

Servo servo3;

Servo servo4;

Servo servo5;

Servo servo6;

const int pinServo1 = 3;

const int pinServo2 = 5;

const int pinServo3 = 6;

const int pinServo4 = 8;

const int pinServo5 = 9;

const int pinServo6 = 10;

const int anguloActivo = 20;

const int anguloInactivo = 0;

void setup() {

  Serial.begin(9600);         // Inicia el Monitor Serial para ver los datos

  bluetooth.begin(9600);      // Inicia el Bluetooth

  servo1.attach(pinServo1);

  servo2.attach(pinServo2);

  servo3.attach(pinServo3);

  servo4.attach(pinServo4);

  servo5.attach(pinServo5);

  servo6.attach(pinServo6);

  resetearServos();

  Serial.println("Sistema Braille listo. Esperando datos por Bluetooth...");

}

void loop() {

  if (bluetooth.available()) {

    char letra = bluetooth.read();

    letra = tolower(letra);

    Serial.print("Letra recibida: ");

    Serial.println(letra);  // Muestra la letra en el monitor serial

    mostrarLetraBraille(letra);

  }

}

void resetearServos() {

  servo1.write(anguloInactivo);

  servo2.write(anguloInactivo);

  servo3.write(anguloInactivo);

  servo4.write(anguloInactivo);

  servo5.write(anguloInactivo);

  servo6.write(anguloInactivo);

}

void mostrarLetraBraille(char letra) {

  resetearServos();

  switch (letra) {

    case 'a': servo1.write(anguloActivo); break;

    case 'b': servo1.write(anguloActivo); servo2.write(anguloActivo); break;

    case 'c': servo1.write(anguloActivo); servo4.write(anguloActivo); break;

    case 'd': servo1.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

    case 'e': servo1.write(anguloActivo); servo5.write(anguloActivo); break;

    case 'f': servo1.write(anguloActivo); servo2.write(anguloActivo); servo4.write(anguloActivo); break;

    case 'g': servo1.write(anguloActivo); servo2.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

    case 'h': servo1.write(anguloActivo); servo2.write(anguloActivo); servo5.write(anguloActivo); break;

    case 'i': servo2.write(anguloActivo); servo4.write(anguloActivo); break;

    case 'j': servo2.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

    case 'k': servo1.write(anguloActivo); servo3.write(anguloActivo); break;         // 1,3

    case 'l': servo1.write(anguloActivo); servo2.write(anguloActivo); servo3.write(anguloActivo); break; // 1,2,3

    case 'm': servo1.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); break; // 1,3,4

    case 'n': servo1.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break; // 1,3,4,5

    case 'o': servo1.write(anguloActivo); servo3.write(anguloActivo); servo5.write(anguloActivo); break; // 1,3,5

    case 'p': servo1.write(anguloActivo); servo2.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); break; // 1,2,3,4

    case 'q': servo1.write(anguloActivo); servo2.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break; // 1,2,3,4,5

    case 'r': servo1.write(anguloActivo); servo2.write(anguloActivo); servo3.write(anguloActivo); servo5.write(anguloActivo); break; // 1,2,3,5

    case 's': servo2.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); break; // 2,3,4

    case 't': servo2.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break; // 2,3,4,5

    case 'u': servo1.write(anguloActivo); servo3.write(anguloActivo); servo6.write(anguloActivo); break; // 1,3,6

    case 'v': servo1.write(anguloActivo); servo2.write(anguloActivo); servo3.write(anguloActivo); servo6.write(anguloActivo); break; // 1,2,3,6

    case 'w': servo2.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); servo6.write(anguloActivo); break; // 2,4,5,6

    case 'x': servo1.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); servo6.write(anguloActivo); break; // 1,3,4,6

    case 'y': servo1.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); servo6.write(anguloActivo); break; // 1,3,4,5,6

    case 'z': servo1.write(anguloActivo); servo3.write(anguloActivo); servo5.write(anguloActivo); servo6.write(anguloActivo); break; // 1,3,5,6

    case ' ': break; // 1,3,5,6

    // Números 1–0 representados como letras a–j con el signo numeral (no incluido aquí)

    case '1': servo1.write(anguloActivo); break;

    case '2': servo1.write(anguloActivo); servo2.write(anguloActivo); break;

    case '3': servo1.write(anguloActivo); servo4.write(anguloActivo); break;

    case '4': servo1.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

    case '5': servo1.write(anguloActivo); servo5.write(anguloActivo); break;

    case '6': servo1.write(anguloActivo); servo2.write(anguloActivo); servo4.write(anguloActivo); break;

    case '7': servo1.write(anguloActivo); servo2.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

    case '8': servo1.write(anguloActivo); servo2.write(anguloActivo); servo5.write(anguloActivo); break;

    case '9': servo2.write(anguloActivo); servo4.write(anguloActivo); break;

    case '0': servo2.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

     // Letras acentuadas (adaptación común en español)

    case 'á': servo1.write(anguloActivo); servo2.write(anguloActivo); servo6.write(anguloActivo); break;

    case 'é': servo1.write(anguloActivo); servo5.write(anguloActivo); servo6.write(anguloActivo); break;

    case 'í': servo2.write(anguloActivo); servo4.write(anguloActivo); servo6.write(anguloActivo); break;

    case 'ó': servo1.write(anguloActivo); servo4.write(anguloActivo); servo6.write(anguloActivo); break;

    case 'ú': servo1.write(anguloActivo); servo3.write(anguloActivo); servo6.write(anguloActivo); break;

    case 'ñ': servo1.write(anguloActivo); servo2.write(anguloActivo); servo5.write(anguloActivo); servo6.write(anguloActivo); break;

    // Signos de puntuación

    case '.': servo2.write(anguloActivo); servo5.write(anguloActivo); break;

    case ',': servo2.write(anguloActivo); break;

    case ';': servo2.write(anguloActivo); servo3.write(anguloActivo); break;

    case ':': servo2.write(anguloActivo); servo4.write(anguloActivo); break;

    case '?': servo2.write(anguloActivo); servo6.write(anguloActivo); break;

    case '!': servo2.write(anguloActivo); servo3.write(anguloActivo); servo5.write(anguloActivo); break;

    case '(': case ')': servo2.write(anguloActivo); servo3.write(anguloActivo); servo6.write(anguloActivo); break;

    case '"': servo2.write(anguloActivo); servo6.write(anguloActivo); break;

    case '-': servo3.write(anguloActivo); servo6.write(anguloActivo); break;

    case '\'': servo3.write(anguloActivo); break;

    case '/': servo2.write(anguloActivo); servo4.write(anguloActivo); servo6.write(anguloActivo); break;

    case '@': servo1.write(anguloActivo); servo6.write(anguloActivo); break;

    default:

      Serial.println("Letra no válida o sin asignación Braille.");

      resetearServos();

      return;

  }

  delay(1000);

  resetearServos();

}

**4.3 Historia de Usuario 3.3 — Reinicio manual de Bluetooth**

Archivo: TraductorFragment.java

swBluetooth.setOnCheckedChangeListener((buttonView, isChecked) -> {  
 if (isChecked) {  
 if (conexionBluetooth.conectar(*HC06\_MAC*, requireContext())) {  
 txtEstadoBluetooth.setText("Conectado");  
 Toast.*makeText*(requireContext(), "Bluetooth conectado", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 txtEstadoBluetooth.setText("Error de conexión");  
 Toast.*makeText*(requireContext(), "No se pudo conectar", Toast.*LENGTH\_SHORT*).show();  
 swBluetooth.setChecked(false);  
 }  
 } else {  
 conexionBluetooth.desconectar();  
 txtEstadoBluetooth.setText("Desconectado");  
 Toast.*makeText*(requireContext(), "Bluetooth desconectado", Toast.*LENGTH\_SHORT*).show();  
 }  
});

**5. Burndown Chart**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Backlog ID** | **Historia de Usuario** | **Tiempo Estimado** | **20/05/2025** | **21/05/2025** | **22/05/2025** | **23/05/2025** | **24/05/2025** | **25/05/2025** | **26/05/2025** | **27/05/2025** |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** | **Dia7** | **Dia8** |
| **HU-3.1** | Conexión Bluetooth | 22 | 12 | 10 |  |  |  |  |  |  |
| **HU-3.2** | Envío de caracteres a Arduino | 47 |  | 2 | 10 | 10 | 8 | 7 | 5 | 5 |
| **HU-3.3** | Reinicio manual de Bluetooth | 6 |  |  |  |  |  |  | 3 | 3 |
| **Tiempo de trabajo** | | 72 | 12 | 12 | 10 | 10 | 8 | 7 | 5 | 8 |

**6. Análisis del Sprint**

* La integración de la conversión automática se realizó con éxito, reduciendo los pasos para el usuario y mejorando la experiencia general.
* Las pruebas iniciales revelaron algunas variaciones en la precisión del reconocimiento de voz, las cuales fueron corregidas mediante mejoras en la normalización del texto transcrito.
* La arquitectura modular del proyecto permitió realizar estas integraciones sin afectar las funciones ya existentes, manteniendo la estabilidad del sistema.
* Se ejecutaron las 8 tareas planificadas dentro de las 72 horas estimadas**.**
* Se completó la interfaz de conexión Bluetooth y se implementó la lógica de envío de caracteres a Arduino.
* Las pruebas confirmaron una representación física correcta del Brailley unaconexión estable**.**
* La opción de reinicio manual funcionó correctamente ante desconexiones simuladas.
* Se resolvieron pequeñas demoras al enviar texto extenso, agregando pausas entre caracteres.

**7. Conclusiones**

El Sprint 6 representó un hito técnico en el desarrollo del proyecto *Braille Express*, al consolidar la conexión entre la aplicación y el hardware, permitiendo la traducción real y tangible del texto a Braille mediante el Arduino.

Las funcionalidades desarrolladas permiten al usuario:

* Visualizar y gestionar la conexión Bluetooth desde la interfaz.
* Enviar texto desde la app y verlo representado físicamente en Braille.
* Reiniciar manualmente la conexión en caso de errores, garantizando continuidad en el uso.

El equipo demostró un alto nivel de coordinación entre desarrollo y pruebas, cumpliendo con los objetivos planteados en tiempo y forma. La integración del hardware con el software fue exitosa y sienta las bases para las últimas fases del proyecto, que buscarán optimizar el rendimiento general y cerrar el ciclo completo de traducción, almacenamiento y representación física.

Este Sprint reafirma el compromiso del equipo con la inclusión digital y tecnológica de personas con discapacidad visual, acercando cada vez más el producto a su versión final funcional.

Registro Diario de Avances – Sprint 7 Braille Express

**1. Introducción**

El Sprint 7 del proyecto *Braille Express* tuvo como objetivo principal fortalecer la experiencia personalizada del usuario mediante la implementación de opciones de personalización visual y la mejora del control de conectividad Bluetooth. Para ello, se desarrollaron funcionalidades que permiten cambiar el tema global de la aplicación (modo claro u oscuro), seleccionar el color de los botones principales, y notificar el estado del reinicio manual de la conexión Bluetooth, ofreciendo una experiencia más adaptable, accesible y controlada.

En total, se ejecutaron 8 tareas correspondientes a las historias de usuario HI 3.3 (Reinicio manual de Bluetooth) y HI 4.1 (Personalización de tema y botones), con una duración estimada de 69 horas, distribuidas entre diseño de interfaces, desarrollo frontend y backend, así como pruebas de funcionamiento. Este sprint marcó un paso importante hacia la inclusión visual, brindando mayor autonomía a los usuarios con discapacidad visual o sensibilidad a determinados estilos visuales.

**2. Lineamientos de Diseño**

En este Sprint se definieron criterios específicos para asegurar una experiencia de usuario fluida, accesible y coherente con los módulos anteriores. Se priorizó lo siguiente:

* **Interfaz accesible**: Se diseñaron selectores de tema y color con contraste alto y controles grandes.
* **Consistencia visual**: El cambio de tema respeta la línea estética general de la app.
* **Persistencia en la nube**: Las preferencias del usuario se guardan y sincronizan automáticamente.
* **Retroalimentación clara**: El sistema notifica al usuario cuando cambia el tema o hay un reinicio de Bluetooth.

**3. Historias de Usuario y Tareas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| HU-3.3 | Reinicio manual de Bluetooth | 8 pts | 29/04/2025 | 19/05/2025 |
| HU-4.1 | Personalización de Tema y Color de Botones | 8 pts | 29/04/2025 | 19/05/2025 |

**4. Evidencias de Código**

**4.1 Historia de Usuario 3.4: Reinicio de Bluetooth**

Archivo: TraductorFragment.java

< swBluetooth.setOnCheckedChangeListener((buttonView, isChecked) -> {  
 if (isChecked) {  
 if (conexionBluetooth.conectar(*HC06\_MAC*, requireContext())) {  
 txtEstadoBluetooth.setText("Conectado");  
 Toast.*makeText*(requireContext(), "Bluetooth conectado", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 txtEstadoBluetooth.setText("Error de conexión");  
 Toast.*makeText*(requireContext(), "No se pudo conectar", Toast.*LENGTH\_SHORT*).show();  
 swBluetooth.setChecked(false);  
 }  
 } else {  
 conexionBluetooth.desconectar();  
 txtEstadoBluetooth.setText("Desconectado");  
 Toast.*makeText*(requireContext(), "Bluetooth desconectado", Toast.*LENGTH\_SHORT*).show();  
 }  
});

**4.2 Historia de Usuario 4.3 — Personalización de Tema y Color de Botones**

Archivo: ConfiguracionModel.java

public class ConfiguracionModel {  
 private String idDNI;  
 private String tema; // "claro" o "oscuro"  
 private String colorBoton; // Ej: "#FF0000"  
  
 public ConfiguracionModel() {} // Constructor vacío requerido por Firebase  
  
 public ConfiguracionModel(String idDNI, String tema, String colorBoton) {  
 this.idDNI = idDNI;  
 this.tema = tema;  
 this.colorBoton = colorBoton;  
 }  
  
 public String getIdDNI() {  
 return idDNI;  
 }  
  
 public void setIdDNI(String idDNI) {  
 this.idDNI = idDNI;  
 }  
  
 public String getTema() {  
 return tema;  
 }  
  
 public void setTema(String tema) {  
 this.tema = tema;  
 }  
  
 public String getColorBoton() {  
 return colorBoton;  
 }  
  
 public void setColorBoton(String colorBoton) {  
 this.colorBoton = colorBoton;  
 }  
}

Archivo: ConfiguracionRepositorio.java

public class ConfiguracionRepositorio {  
 private final DatabaseReference ref;  
  
 public ConfiguracionRepositorio() {  
 ref = FirebaseDatabase.*getInstance*().getReference("TBConfiguracion");  
 }  
  
 public void guardarConfiguracion(ConfiguracionModel config) {  
 ref.child(config.getIdDNI()).setValue(config);  
 }  
  
 public void obtenerConfiguracion(String idDNI, ValueEventListener listener) {  
 ref.child(idDNI).addListenerForSingleValueEvent(listener);  
 }  
}

Archivo: ConfiguracionFragment.java

public class ConfiguracionFragment extends Fragment {  
  
 private RadioGroup radioGroupTema, radioGroupColor;  
 private Button btnAplicarCambios;  
 private ConfiguracionViewModel configViewModel;  
 private String idDNI;  
  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
 return inflater.inflate(R.layout.*fragment\_configuracion*, container, false);  
 }  
  
 @Override  
 public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {  
 super.onViewCreated(view, savedInstanceState);  
  
 radioGroupTema = view.findViewById(R.id.*radioGroupTema*);  
 radioGroupColor = view.findViewById(R.id.*radioGroupColor*);  
 btnAplicarCambios = view.findViewById(R.id.*btnAplicarCambios*);  
 configViewModel = new ViewModelProvider(this).get(ConfiguracionViewModel.class);  
  
 // Obtener ID del usuario  
 SharedPreferences prefs = requireActivity().getSharedPreferences("user\_session", Context.*MODE\_PRIVATE*);  
 idDNI = prefs.getString("dni\_usuario", "");  
  
 if (!idDNI.isEmpty()) {  
 configViewModel.cargarConfiguracion(idDNI);  
 }  
  
 configViewModel.getConfiguracion().observe(getViewLifecycleOwner(), config -> {  
 if (config != null) {  
 // Cargar valores existentes  
 if ("claro".equals(config.getTema())) {  
 radioGroupTema.check(R.id.*radioClaro*);  
 } else {  
 radioGroupTema.check(R.id.*radioOscuro*);  
 }  
  
 switch (config.getColorBoton()) {  
 case "#FF0000": radioGroupColor.check(R.id.*colorRojo*); break;  
 case "#0000FF": radioGroupColor.check(R.id.*colorAzul*); break;  
 case "#00FF00": radioGroupColor.check(R.id.*colorVerde*); break;  
 case "#800080": radioGroupColor.check(R.id.*colorMorado*); break;  
 case "#FF69B4": radioGroupColor.check(R.id.*colorRosado*); break;  
 case "#FFD700": radioGroupColor.check(R.id.*colorAmarillo*); break;  
 }  
 }  
 });  
  
 btnAplicarCambios.setOnClickListener(v -> {  
 String tema = radioGroupTema.getCheckedRadioButtonId() == R.id.*radioClaro* ? "claro" : "oscuro";  
 String color = obtenerColorSeleccionado();  
  
 ConfiguracionModel nuevaConfig = new ConfiguracionModel(idDNI, tema, color);  
 configViewModel.guardarConfiguracion(nuevaConfig);  
  
 // Guardar en SharedPreferences  
 SharedPreferences.Editor editor = prefs.edit();  
 editor.putString("tema", tema);  
 editor.putString("color\_boton", color);  
 editor.apply();  
  
 // Aplicar tema inmediatamente  
 if (tema.equals("oscuro")) {  
 AppCompatDelegate.*setDefaultNightMode*(AppCompatDelegate.*MODE\_NIGHT\_YES*);  
 } else {  
 AppCompatDelegate.*setDefaultNightMode*(AppCompatDelegate.*MODE\_NIGHT\_NO*);  
 }  
 });  
 }  
  
 private String obtenerColorSeleccionado() {  
 if (radioGroupColor == null) return "#000000";  
  
 int checkedId = radioGroupColor.getCheckedRadioButtonId();  
  
 if (checkedId == -1) return "#000000";  
  
 if (checkedId == R.id.*colorRojo*) return "#FF0000";  
 if (checkedId == R.id.*colorAzul*) return "#0000FF";  
 if (checkedId == R.id.*colorVerde*) return "#00FF00";  
 if (checkedId == R.id.*colorMorado*) return "#800080";  
 if (checkedId == R.id.*colorRosado*) return "#FF69B4";  
 if (checkedId == R.id.*colorAmarillo*) return "#FFD700";  
  
 return "#000000";  
 }  
  
}

Archivo: ConfiguracionViewModel.java

public class ConfiguracionViewModel extends ViewModel {  
 private final ConfiguracionRepositorio repositorio = new ConfiguracionRepositorio();  
 private final MutableLiveData<ConfiguracionModel> configuracionLiveData = new MutableLiveData<>();  
  
 public LiveData<ConfiguracionModel> getConfiguracion() {  
 return configuracionLiveData;  
 }  
  
 public void cargarConfiguracion(String idDNI) {  
 repositorio.obtenerConfiguracion(idDNI, new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 ConfiguracionModel config = snapshot.getValue(ConfiguracionModel.class);  
 if (config != null) {  
 configuracionLiveData.setValue(config);  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 // Manejo de errores si lo deseas  
 }  
 });  
 }  
  
 public void guardarConfiguracion(ConfiguracionModel config) {  
 repositorio.guardarConfiguracion(config);  
 }  
}

Archivo: ConfiguracionViewModel.java

public class ConfiguracionViewModel extends ViewModel {  
 private final ConfiguracionRepositorio repositorio = new ConfiguracionRepositorio();  
 private final MutableLiveData<ConfiguracionModel> configuracionLiveData = new MutableLiveData<>();  
  
 public LiveData<ConfiguracionModel> getConfiguracion() {  
 return configuracionLiveData;  
 }  
  
 public void cargarConfiguracion(String idDNI) {  
 repositorio.obtenerConfiguracion(idDNI, new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 ConfiguracionModel config = snapshot.getValue(ConfiguracionModel.class);  
 if (config != null) {  
 configuracionLiveData.setValue(config);  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
 // Manejo de errores si lo deseas  
 }  
 });  
 }  
  
 public void guardarConfiguracion(ConfiguracionModel config) {  
 repositorio.guardarConfiguracion(config);  
 }  
}

Archivo: fragment\_configuracion.xml

?xml version="1.0" encoding="utf-8"?>  
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:padding="16dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <!-- Sección para Tema Claro / Oscuro -->  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Selecciona el tema:"  
 android:textStyle="bold"  
 android:textSize="16sp"  
 android:layout\_marginBottom="10dp"/>  
  
 <RadioGroup  
 android:id="@+id/radioGroupTema"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <RadioButton  
 android:id="@+id/radioClaro"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Claro" />  
  
 <RadioButton  
 android:id="@+id/radioOscuro"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Oscuro" />  
 </RadioGroup>  
  
 <!-- Espacio -->  
 <View  
 android:layout\_width="match\_parent"  
 android:layout\_height="20dp" />  
  
 <!-- Sección para selección de color de botón -->  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Color de los botones:"  
 android:textStyle="bold"  
 android:textSize="16sp"  
 android:layout\_marginBottom="10dp"/>  
  
 <RadioGroup  
 android:id="@+id/radioGroupColor"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <RadioButton  
 android:id="@+id/colorRojo"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Rojo"  
 android:textColor="#FF0000" />  
  
 <RadioButton  
 android:id="@+id/colorAzul"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Azul"  
 android:textColor="#0000FF" />  
  
 <RadioButton  
 android:id="@+id/colorVerde"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Verde"  
 android:textColor="#00FF00" />  
  
 <RadioButton  
 android:id="@+id/colorMorado"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Morado"  
 android:textColor="#800080" />  
  
 <RadioButton  
 android:id="@+id/colorRosado"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Rosado"  
 android:textColor="#FF69B4" />  
  
 <RadioButton  
 android:id="@+id/colorAmarillo"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Amarillo"  
 android:textColor="#FFD700" />  
 </RadioGroup>  
  
 <!-- Botón para aplicar los cambios -->  
 <Button  
 android:id="@+id/btnAplicarCambios"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Aplicar Cambios"  
 android:layout\_marginTop="20dp"/>  
 </LinearLayout>  
</ScrollView>

**5. Burndown Chart**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Backlog ID** | **Historia de Usuario** | **Tiempo Estimado** | **20/06/2025** | **21/06/2025** | **22/06/2025** | **23/06/2025** | **24/06/2025** | **25/06/2025** | **25/06/2025** |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** | **Dia7** |
| **HU-3.3** | Reinicio manual de Bluetooth | 16 | 8 | 8 |  |  |  |  |  |
| **HU-4.1** | Personalización de temas | 29 |  |  | 8 | 8 | 7 | 4 | 2 |
| **Tiempo de trabajo** | | 45 | 8 | 8 | 8 | 8 | 7 | 4 | 2 |

**6. Análisis del Sprint**

* Se cumplieron las8 tareas planificadas dentro del tiempo establecido (69 horas).
* Se diseñaron e implementaron los selectores de tema (claro/oscuro) y color de botones.
* Se configuró la persistencia de preferencias usando Firebase, permitiendo sincronización multi-dispositivo.
* Se integraron notificaciones visuales para cambios de tema y reinicio Bluetooth.
* Las pruebas confirmaron que los cambios se aplican correctamente y que el sistema conserva la configuración entre sesiones.

**7. Conclusiones**

El Sprint 7 concluyó con la implementación exitosa de una de las funcionalidades más valoradas en aplicaciones accesibles: la personalización visual, lo cual mejora significativamente la comodidad y adaptabilidad para distintos perfiles de usuario. Ahora los usuarios pueden elegir entre tema claro u oscuro y seleccionar colores de botones que se ajusten a su preferencia o necesidad visual, lo cual representa un avance concreto en términos de usabilidad e inclusión.

Además, se mejoró el manejo de errores relacionados con la conexión Bluetooth al diseñar una notificación clara del estado de reinicio, permitiendo que los usuarios sepan si su acción fue efectiva o si deben intentar nuevamente.

Las tareas se completaron en el plazo estimado, mostrando una buena coordinación entre diseño, desarrollo y pruebas. La persistencia de configuraciones en la nube mediante Firebase asegura que las personalizaciones elegidas se mantengan incluso si el usuario cambia de dispositivo, fortaleciendo la continuidad de la experiencia.

Este sprint sienta las bases para una aplicación más flexible, intuitiva y centrada en el usuario, consolidando a *Braille Express* como una solución tecnológica accesible y confiable para personas con discapacidad visual.

Manual Técnico — Braille Express

**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

Manual de Instalación y Configuración — Braille Express

**1. Introducción**

Este documento proporciona instrucciones detalladas para la instalación, configuración y despliegue del sistema Braille Express**,** compuesto por una aplicación Android, un backend basado en Firebasey un sistema físico controlado mediante Arduino Nano yBluetooth HC-06**.** Está dirigido a desarrolladores, técnicos o responsables de implementación en contextos educativos.

**2. Requisitos del Sistema**

**2.1 Requisitos de Software**

* Android Studio 7.0 o superior
* Firebase Console con:
* Autenticación habilitada (DNI/Password)
* Realtime Database activa
* Arduino IDE para cargar el código al Arduino Nano
* Librerías Arduino: Servo.h, SoftwareSerial
* APK firmado o emulador para ejecución

**2.2 Requisitos de Hardware (mínimos)**

* Smartphone o Tablet con Android 8.0+
* Arduino Nano + Shield de expansión
* 6 servos (uno por punto Braille)
* Módulo Bluetooth HC-06
* Cable USB para programación del Arduino
* Fuente de alimentación 5V-3A

**3. Estructura del Proyecto**

/

/braille-express/

├── app/

│ ├── java/com/example/brailleexpress/

│ │ ├── ui/

│ │ ├── data/

│ │ ├── model/

│ │ ├── ConexionBluetooth

│ ├── res/layout/

└── arduino/

└── braille.ino

**4. Instalación del Entorno de Desarrollo**

**4.1 Clonación del Repositorio**

git clone https://github.com/amestrada/taskmanager.git

cd taskmanager

**4.2 Configuración del Proyecto Android**

1. Abrir Android Studio
2. Seleccionar “Import Project” y elegir la carpeta braille-express/
3. Esperar a que Gradle sincronice el proyecto
4. Verificar dependencias y habilitar permisos de Bluetooth en AndroidManifest.xml

**4.3 Configuración de Firebase**

1. Crear un proyecto en Firebase Console: BrailleExpress\_db
2. Habilitar:

* Authentication (DNI/Password)
* Realtime Database

1. Descargar google-services.json y colocarlo en /app
2. Configurar reglas de seguridad en firebase/rules/

**4.4 Instalación del APK**

1. Compilar desde Android Studio en un emulador o dispositivo real
2. Alternativamente, instalar el APK firmado:

adb install BrailleExpress.apk

**4.5 Configuración del Arduino**

1. Abrir braille.ino en Arduino IDE
2. Seleccionar placa **Arduino Nano** y puerto correspondiente
3. Cargar el programa
4. Verificar conexión del HC-06 en pines RX/TX adecuados
5. Verificar los pines de conexión de los 6 servomotores.
6. Alimentar servos desde fuente externa (opcional, pero recomendado)

**5. Despliegue en Producción**

**5.1 Aplicación Android**

* Firmar la APK para distribución
* Habilitar el uso de permisos requeridos en el manifiesto
* Publicar en Google Play para las descargas de los usuarios.

**5.2 Firebase**

* Migrar a plan Blaze una vez que el uso supera las cuotas gratuitas.
* Reforzar reglas de seguridad en producción respecto a la base de datos.

**5.3 Dispositivo Electrónico**

* Asegurar montaje mecánico de los servos
* Verificar emparejamiento HC-06 con el dispositivo Android
* Revisar nivel de señal para evitar interferencias en entornos escolares

**6. Configuración Adicional**

**6.1 Seguridad**

* Uso de autenticación segura con Firebase Authentication
* Sincronización de datos personales (historial, configuración) por ID único
* Los datos de Firebase están protegidos por reglas específicas por usuario
* Protección del archivo google-services.json y claves API
* Recomendación de uso solo en redes WiFi seguras

**6.2 Logs y Diagnóstico**

* Logcat en Android Studio para depuración
* Validación de logs del módulo Bluetooth (en consola serial Arduino)
* Opcional: integración con Firebase Crashlytics para monitoreo de errores

**7. Verificación de Instalación**

Verifica que:

* Se puede registrar e iniciar sesión con una cuenta nueva
* Se traduce texto a Braille correctamente
* La configuración (tema y color de botones) se guarda y recupera
* El historial muestra correctamente las traducciones anteriores
* El Arduino responde correctamente a las letras enviadas desde la app

**8. Contacto Técnico**

Para soporte o colaboración técnica, contactar al correo: [soporteBrailleExpress@gmail.com](mailto:soporteBrailleExpress@gmail.com).

**9. Anexos**

**9.1 Enlaces de descarga**

* Android Studio (IDE para desarrollo Android): <https://developer.android.com/studio>
* Arduino IDE (programación del hardware):

<https://www.arduino.cc/en/software/>

* Firebase Console:

<https://console.firebase.google.com>

Manual del Administrador del Sistema — Braille Express

**1. Introducción**

Este manual está dirigido a los administradores del sistema Braille Express. Su objetivo es proporcionar lineamientos para la gestión de usuarios, monitoreo de datos almacenados en Firebase, mantenimiento del sistema de traducción Braille (software y hardware) y respaldo de información clave. Incluye procedimientos técnicos tanto para la aplicación Android como para el dispositivo Arduino.

**2. Acceso al Panel de Administración**

Actualmente, Braille Express no cuenta con un dashboard visual de administración. Todas las acciones administrativas se realizan mediante:

* Firebase Console (https://console.firebase.google.com)
* El entorno de desarrollo Android Studio
* Arduino IDE para mantenimiento del firmware
* Herramientas externas de respaldo manual y monitoreo

Se planea incorporar un panel de administración accesible desde la aplicación para facilitar la gestión en futuras versiones.

**3. Gestión de Usuarios**

**3.1 Visualizar Usuarios**

* **Ubicación:** Los datos de los usuarios están almacenados en la tabla TbUsuario dentro de Firebase Realtime Database.
* **Ruta típica:** /TbUsuario/{idDNI}/
* **Campos comunes:**
* DNI (PK)
* Nombre
* Apellidos
* Edad
* Género
* Contraseña
* **Acciones:**
* Revisar los registros desde Firebase Console.
* Exportar datos para auditoría (se puede hacer mediante Firebase Admin SDK).

**3.2 Eliminar Usuario**

* Se puede realizar desde la consola de Firebase.
* **Consecuencias esperadas:**
* Se deben eliminar también los datos relacionados:
  + /TbHistorial/{idDNI}
  + /TbConfiguracion/{idDNI}
* **Recomendación:** Implementar una función que realice la eliminación en cascada desde la app o con Firebase Functions para evitar inconsistencias.

**3.3 Actualización de Datos del Usuario**

* Firebase no tiene editor masivo, por lo que los cambios deben hacerse manualmente:
* Ir a /TbUsuarios/{idDNI}
* Editar campos directamente desde la consola.
* Recomendado: validar que el ID (DNI) no cambie, ya que es la clave primaria del resto de tablas.

**4. Gestión de Historial**

**4.1 Consultar Historial de Traducciones**

* Ubicación: TbHistorial/
* Filtrar por campo idDNI para obtener todas las traducciones asociadas a un usuario.

**4.2 Eliminar Ítems del Historial**

* Los administradores pueden eliminar entradas específicas del historial de traducciones asociadas a un usuario desde Firebase.
* Eliminación Manual desde Firebase Console:
* Ir a Realtime Database → TbHistorial/{idDNI}/
* Identificar el nodo del historial que deseas eliminar (por su ID único o contenido).
* Hacer clic derecho sobre el nodo → *Eliminar*

**5. Respaldo y Recuperación de Base de Datos**

**5.1 Respaldo de Firebase Realtime Database**

* Ir a Firebase Console → Realtime Database
* Menú de ⋮ → *Export JSON*
* Guardar respaldo local

**5.2 Restauración**

1. Desde la consola, usar *Import JSON*
2. Seleccionar el archivo previamente exportado

**Nota:** Este método es manual. Para automatizar se recomienda Firebase Admin SDK con cron jobs.

**6. Monitoreo y Logs**

**6.1 Actividad de la Aplicación**

* Usar Logcat en Android Studio para monitorear el funcionamiento.
* Registrar eventos de sesión, conexión Bluetooth, errores de Firebase, etc.

**6.2 Logs del Hardware**

* Monitorear la conexión y recepción de caracteres desde el Monitor Serial del Arduino IDE.
* Incluir mensajes de depuración como: Serial.println("Letra recibida: a");

**7. Seguridad Administrativa**

* Protege la consola de Firebase: solo usuarios autorizados deben tener permisos de lectura/escritura.
* Evita subir archivos sensibles (google-services.json) a repositorios públicos.
* Restringe el acceso a la app física (Arduino): mediante autenticación desde la app Android en futuras versiones.
* Cambia regularmente los tokens de API si usas Firebase Admin SDK.

**8. Automatización Recomendada**

* Respaldo periódico de Realtime Database: mediante script o herramientas de terceros.
* Depuración automática de historial antiguo: por fecha en Firebase.
* Notificaciones automáticas: uso de Firebase Cloud Messaging para alertar a usuarios sobre cambios de configuración.

**9. Contacto para Escalamiento**

En caso de fallos graves en hardware o software, comunicarse con el equipo técnico

soporteBrailleExpress@gmail.com

**10. Cambios Previstos para el Rol de Administrador**

* Implementación de un panel de control visual dentro de la app.
* Gestión de estadísticas por alumno y frecuencia de uso.
* Sistema de recuperación de cuentas por DNI y verificación vía correo.
* Sincronización avanzada y cifrado de datos sensibles.

**Braille Express** Aplicación es una app móvil en crecimiento. Este manual será actualizado conforme se integren nuevas funciones administrativas.

Manual de Mantenimiento — Braille Express

**1. Introducción**

Este manual describe de manera detallada los procedimientos de mantenimiento preventivo y correctivo del sistema Braille Express, una solución tecnológica compuesta por tres componentes principales: la aplicación móvil Android, la base de datos en Firebase y el módulo de hardware físico controlado por Arduino Nano. El objetivo principal de este documento es proporcionar una guía estructurada al equipo técnico encargado del soporte y operación del sistema, con el fin de garantizar la continuidad del servicio, la integridad de los datos y el correcto funcionamiento del hardware electromecánico.

El mantenimiento adecuado de Braille Express es esencial para asegurar que la plataforma siga cumpliendo con su propósito educativo: facilitar la comprensión lectora de estudiantes con discapacidad visual mediante la traducción automática de texto digital a Braille. Dado que el sistema involucra tanto software como componentes físicos, el enfoque del mantenimiento es integral y considera aspectos como:

* Prevención de fallos en la aplicación móvil, conectividad Bluetooth y módulo Arduino.
* Monitoreo de rendimiento, errores y estabilidad en el entorno de Firebase.
* Limpieza periódica de datos innecesarios o temporales.
* Actualización de firmware, bibliotecas y librerías utilizadas en el sistema.
* Verificación del estado de los servos, la comunicación HC-06 y las respuestas del hardware.

Este manual está dirigido a desarrolladores, técnicos de soporte, personal de mantenimiento escolar o cualquier miembro del equipo responsable de mantener operativa y segura la solución Braille Express. Su correcta aplicación garantiza una experiencia de uso continua, confiable y accesible para los estudiantes y docentes beneficiarios.

**2. Objetivos del Mantenimiento**

* Prevenir fallos antes de que afecten la experiencia del usuario.
* Corregir errores en la app, base de datos o hardware de manera eficiente.
* Asegurar la integridad de la información almacenada en Firebase.
* Mantener la conectividad Bluetooth y la funcionalidad del dispositivo Braille.

**3. Mantenimiento Preventivo**

**3.1 Tareas Semanales**

* Verificar registros de errores y mensajes Logcat en Android Studio.
* Confirmar conectividad y respuesta del módulo Bluetooth (HC-06).
* Validar sincronización correcta entre Firebase y la app.
* Revisar funcionamiento físico de los 6 servomotores del sistema Braille.

**3.2 Tareas Mensuales**

* Limpieza manual de entradas duplicadas o erróneas en TbHistorial o TbConfiguracion.
* Probar funcionalidades críticas: login, registro, traducción y envío al Arduino.
* Revisar el estado de almacenamiento local (SharedPreferences) y sincronización por ID de usuario.
* Realizar pruebas en múltiples dispositivos Android con diferentes versiones del sistema operativo.

**3.3 Tareas Trimestrales**

* Verificar que el firmware cargado en el Arduino Nano sigue siendo compatible con la versión de la app.
* Realizar pruebas de desempeño del Bluetooth bajo diferentes distancias y obstáculos.
* Evaluar la estructura de datos en Firebase y optimizar reglas de seguridad si es necesario.
* Auditar el comportamiento de los ViewModels y su interacción con Firebase para evitar fugas de memoria.

**4. Mantenimiento Correctivo**

**4.1 Gestión de Errores Comunes**

* **La app no se conecta por Bluetooth:** Confirmar que el módulo HC-06 esté emparejado y encendido.
* **Firebase no guarda datos:** Revisar conexión a internet, reglas de acceso o posibles errores de permisos.
* **Traducciones no se registran en historial:** Verificar que el método de envío también invoque la función de almacenamiento.
* **Preferencias no se aplican:** Corregir posibles problemas de lectura desde SharedPreferences o de sincronización con TbConfiguracion.

**4.2 Procedimiento de Corrección**

1. Identificar el error (pantalla afectada, logs, dispositivo).
2. Replicar el error en un entorno de prueba.
3. Corregir el código o revisar la configuración (Bluetooth/Firebase).
4. Validar la solución en varios dispositivos.
5. Documentar la causa raíz y la solución aplicada.

**5. Respaldo de Información**

**5.1 Respaldo Manual**

* Desde la consola de Firebase, exportar los datos de:
* /TbUsuarios
* /TbHistorial
* /TbConfiguracion
* Descargar archivo .json y almacenarlo en una unidad segura.

**5.2 Respaldo Programado (CRON)**

* Usar funciones de exportación automática desde Firebase CLI.
* Programar tareas (CRON o scripts) en entorno de servidor para guardar copias de seguridad en Google Drive o almacenamiento local.

**6. Actualización del Sistema**

**6.1 Aplicación Android**

* Compilar nueva versión APK desde Android Studio.
* Firmar digitalmente el APK para distribución.
* Instalar la nueva versión en los dispositivos o distribuir vía Google Play o enlaces directos.

**6.2 Configuración de Firebase**

* Revisar y ajustar reglas de seguridad.
* Confirmar integridad de estructuras JSON después de actualizaciones.
* Validar conectividad con Realtime Database y autenticación.
* Usar Arduino IDE para cargar el archivo actualizado Braille.ino.
* Confirmar asignación correcta de pines y lógica de impresión Braille.

**6.3 Firmware del Arduino**

* Usar Arduino IDE para cargar el archivo actualizado Braille.ino.
* Confirmar asignación correcta de pines y lógica de impresión Braille.

**7. Documentación de Cambios**

Se registrará cada cambio en el archivo Cambios.doc, que estará en un archivo Drive Braille Express, este estará presentado bajo el siguiente formato:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Fecha | Tipo de Cambio | Descripción Detallada | Responsable |
| 1.0.0 | 15/06/2025 | Publicación inicial | Primera versión estable de la aplicación Braille Express. Incluye traducción de texto a Braille y Bluetooth. | Equipo de desarrollo |
|  |  | Agregado | - Funcionalidad de traducción texto → Braille con vista en pantalla. - Conexión a Arduino Nano vía HC-06. |  |
|  |  | Agregado | - Sistema de historial persistente. - Configuración de tema y color de botones con sincronización en Firebase. |  |
| 1.1.0 | 01/07/2025 | Mejora | Menú contextual en historial: opciones para copiar texto y eliminar entradas. | Y. Ochoa / DevTeam |
|  |  | Corrección | Ajustes en el cambio dinámico de tema claro/oscuro y aplicación del color de botones. |  |
|  |  | Mejora | Separación de lógica con MVVM y mejoras en la accesibilidad visual de la interfaz. |  |
| 1.2.0 | En desarrollo | Planificado | Implementación futura de OCR para imágenes, soporte de múltiples idiomas y panel estadístico para docentes. |  |

Leyenda:

* **Agregado**: Nueva funcionalidad o componente incorporado al sistema.
* **Corrección**: Solución aplicada a errores o problemas detectados.
* **Mejora**: Optimización de funcionalidades existentes o del rendimiento.
* **Eliminado**: Funcionalidad retirada por obsolescencia o rediseño.
* **Planificado**: Cambios en curso o pendientes de implementación.

**8. Herramientas Recomendadas**

* Desarrollo y depuración: Android Studio, Arduino IDE, Firebase Console.
* Monitoreo de errores: Logcat, Firebase Crashlytics.
* Control de versiones: Git + GitHub.
* Respaldo de datos: Firebase CLI, Google Drive, JSON export.
* Documentación técnica: Google Docs, Notion o GitHub Wiki.

**9. Contacto de Soporte**

Para incidencias mayores o errores de funcionamiento, contactar al equipo técnico en:

[soporteBrailleExpress@gmail.com](mailto:soporteBrailleExpress@gmail.com)

**10. Conclusión**

El mantenimiento de Braille Express no solo implica la corrección de fallos técnicos o la actualización de componentes, sino que representa un compromiso continuo con la calidad, la accesibilidad y la inclusión educativa. Este sistema, que integra una aplicación móvil Android, una plataforma de almacenamiento en la nube mediante Firebase, y un módulo físico con Arduino y servomotores, requiere un enfoque integral de soporte técnico para asegurar que funcione de manera óptima en entornos reales con usuarios con discapacidad visual.

La aplicación de buenas prácticas en el mantenimiento preventivo, como la verificación de logs, la sincronización de datos y las pruebas de conectividad con hardware, permite anticipar problemas antes de que afecten a los usuarios finales. Asimismo, el mantenimiento correctivo documentado y estructurado garantiza que, ante cualquier eventualidad, se pueda responder de manera rápida, segura y eficiente.

Además, la creación de respaldos periódicos, la documentación sistemática de cambios, y la revisión periódica de las configuraciones y preferencias del usuario, aseguran la continuidad del servicio, incluso en casos de fallos inesperados o cambios de dispositivo.

En el marco de un entorno educativo, donde estudiantes con discapacidad visual dependen de Braille Express para desarrollar habilidades de lectura y comprensión a través del sistema Braille, la responsabilidad del mantenimiento técnico adquiere una dimensión ética y social. Por eso, este manual debe ser entendido como una guía viva, que evoluciona junto con la tecnología, las necesidades del usuario y las mejoras del sistema.

Finalmente, es importante que el equipo técnico encargado de Braille Express mantenga una actitud proactiva, colaborativa y centrada en el usuario, promoviendo la accesibilidad, la mejora continua y la sostenibilidad del proyecto a largo plazo. Solo así se garantizará que esta solución siga siendo una herramienta efectiva, confiable y transformadora para las personas que más la necesitan.

Manual de Pruebas — Braille Express

**1. Introducción**

Este manual describe el proceso de verificación y validación del sistema Braille Express, que incluye la aplicación móvil Android, la base de datos en Firebase Realtime Database y el módulo físico basado en Arduino Nano con servomotores. Está dirigido al equipo de control de calidad (QA), desarrolladores y personal técnico, con el fin de garantizar que la solución cumpla con los requisitos funcionales, sea estable, segura y accesible para personas con discapacidad visual.

**2. Objetivos**

* Verificar que las funcionalidades de traducción, historial, configuración y conexión física funcionen correctamente.
* Detectar errores antes de la entrega final o su uso en el entorno educativo.
* Asegurar la usabilidad y accesibilidad del sistema.
* Validar que los datos se guarden y sincronicen correctamente por usuario.

**3. Tipos de Pruebas Aplicadas**

**3.1 Pruebas Unitarias**

* Validan clases Java como MapaBrailleModel, BrailleTraductorModel y los ViewModels.
* Herramientas: JUnit, Mockito.

**3.2 Pruebas de Integración**

* Verifican la interacción entre Firebase y la app (ej. guardar historial, obtener configuración).
* Ejemplo: Traducción → Envío a Arduino → Registro en Firebase.

**3.3 Pruebas Funcionales**

* Validan funciones como: traducción de texto, selección de tema, conexión Bluetooth, y recuperación del historial.
* Herramientas: TestLab (Firebase), Espresso (Android).

**3.5 Pruebas Manuales (Caja Negra)**

* Pruebas desde el punto de vista del usuario final (estudiante o docente).
* Simulan el uso en dispositivos reales y evalúan respuesta visual y física.

**4. Ambiente de Pruebas**

|  |  |
| --- | --- |
| Componente | Versión / Condición |
| Android Studio | 7.0 o superior |
| Dispositivo | Android físico con Bluetooth |
| Firebase | Proyecto con Realtime Database |
| Arduino IDE | 1.8+ (con librería Servo + SoftwareSerial) |
| Hardware | Arduino Nano + módulo HC-06 + 6 servos |

**5. Casos de Prueba**

**5.1 Traducción de Texto a Braille**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Descripción | Entrada | Resultado Esperado | Estado |
| TC01 | Traducción simple | "hola" | Muestra braille ⠓⠕⠇⠁ en pantalla y hardware | Aceptado |
| TC02 | Traducción con signos y mayúsculas | "¡Hola!" | Muestra signos y representación en Braille | Aceptado |

**5.2 Conexión con Arduino vía Bluetooth**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Descripción | Entrada | Resultado Esperado | Estado |
| TC03 | Activar Bluetooth | Toggle activado | Conexión establecida con HC-06 | Aceptado |
| TC04 | Enviar letra traducida | Letra "a" | Servo 1 se mueve según mapa de Braille | Aceptado |

**5.3 Historial y Configuración**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Descripción | Entrada | Resultado Esperado | Estado |
| TC05 | Guardar traducción | "luz" | Se guarda con fecha y color elegido | Aceptado |
| TC06 | Cargar configuración | ID de usuario | Carga tema y color desde Firebase | Aceptado |

**6. Resultados**

Todas las pruebas funcionales fueron exitosas durante los ciclos de prueba del Sprint 1. Las funcionalidades críticas (traducción, historial, personalización y Bluetooth) se comportan conforme a lo esperado, tanto a nivel de interfaz como en hardware físico.

**7. Criterios de Aceptación**

* El sistema debe traducir texto digital a Braille correctamente (visual + físico).
* La configuración debe ser persistente por usuario.
* El historial debe almacenarse correctamente y permitir acciones como copiar y eliminar.
* La conexión Bluetooth debe ser estable y funcional.
* La app debe responder sin errores en dispositivos con Android 7 o superior.

**8. Herramientas Utilizadas**

* **Android Studio:** Emulador + pruebas unitarias/instrumentadas.
* **Firebase Test Lab:** Pruebas automáticas en distintos dispositivos.
* **JUnit / Mockito:** Validación de lógica de negocio.
* **Hardware físico:** Arduino Nano + servos + módulo Bluetooth.
* **Espresso:** Automatización de pruebas UI.

**9. Recomendaciones Futuras**

* Incluir pruebas automatizadas con Espresso para flujos completos.
* Ejecutar pruebas de rendimiento en múltiples dispositivos Android.
* Probar uso del sistema por usuarios con baja visión o sin visión para validar accesibilidad.
* Simular desconexiones de Bluetooth para validar comportamiento.
* Agregar pruebas E2E conectando app, Firebase y Arduino en tiempo real.

**10. Contacto QA**

Para informes de errores o coordinación de nuevas pruebas: [qaBrailleExpress@gmail.com](mailto:qaBrailleExpress@gmail.com)

Registro de Mantenimiento y Plan de Mejoras Futuras — Braille Express

**1. Introducción**

Este documento consolida el historial de mantenimiento del sistema Braille Express, que incluye la aplicación Android, la base de datos en Firebase y el módulo físico con Arduino. Además, presenta un plan de mejoras futuras para asegurar la sostenibilidad, accesibilidad, escalabilidad y eficiencia del sistema. Está dirigido a los responsables del soporte técnico, desarrollo, control de calidad y toma de decisiones del proyecto.

**2. Registro de Mantenimiento**

**2.1 Historial de Mantenimientos Ejecutados**

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Tipo | Descripción | Responsable |
| 2025-06-10 | Preventivo | Limpieza de datos antiguos en el historial de Firebase | Equipo Técnico |
| 2025-06-11 | Correctivo | Solución a error de desconexión Bluetooth entre app y módulo HC-06 | Dev Android |
| 2025-06-12 | Preventivo | Actualización de librerías Android y revisión de permisos de Bluetooth | QA / DevOps |
| 2025-06-13 | Correctivo | Restauración de configuración corrupta en SharedPreferences | Android Dev |
| 2025-06-14 | Preventivo | Verificación de integridad del sistema, logs de conexión y actualización de temas | Equipo de Soporte |

**3. Diagnóstico General**

Tras la primera iteración del sistema, se han identificado fortalezas y áreas de oportunidad:

**3.1 Fortalezas**

* Interfaz accesible para usuarios con discapacidad visual.
* Arquitectura en capas (MVVM) modular y organizada.
* Configuración personalizada por usuario (temas, colores).
* Sincronización en Firebase en tiempo real.

**3.2 Oportunidades de Mejora**

* Aún no se implementa reconocimiento OCR.
* No existe panel administrativo para monitoreo centralizado.
* El sistema no notifica errores de conexión Bluetooth en tiempo real.
* Falta soporte para idiomas adicionales y accesibilidad ampliada.

**4. Plan de Mejoras Futuras**

**4.1 Funcionalidades Prioritarias**

|  |  |  |  |
| --- | --- | --- | --- |
| Mejora | Descripción | Prioridad | Versión Estimada |
| Integración de OCR | Permitir traducir texto desde imágenes capturadas | Alta | v1.1 |
| Panel de administración | Dashboard para gestión de usuarios, configuraciones y monitoreo de uso | Alta | v1.2 |
| Estadísticas de uso | Registro de métricas por estudiante o sesión | Media | v1.2 |
| Multilenguaje | Soporte para traducción desde varios idiomas al Braille | Alta | v1.3 |
| Mejoras de accesibilidad | Interfaz con texto ampliado y navegación por voz | Alta | v1.3 |
| Modo sin conexión | Funcionalidad parcial en ausencia de internet | Media | v1.4 |
| Registro de errores | Sistema de captura de errores en el módulo Arduino y app | Alta | v1.4 |

**4.2 Acciones de Mantenimiento Programado**

* Backups automáticos semanales de datos en Firebase (mediante Cloud Functions).
* Revisión mensual del rendimiento de la aplicación en dispositivos reales.
* Pruebas de compatibilidad con nuevos modelos Android cada trimestre.
* Auditoría del hardware Arduino y calibración de servomotores cada 3 meses.
* Actualización semestral del código Braille.ino con mejoras de precisión.

**5. Seguimiento y Actualización**

* Este documento debe revisarse al final de cada ciclo de desarrollo o Sprint.
* Todas las mejoras aplicadas deben ser documentadas en el registro de cambios en Google Docs.
* Se recomienda establecer una reunión técnica bimensual para evaluar la evolución del sistema.

**6. Contacto Responsable**

[mantenimientoBrailleExpress@gmail.com](mailto:mantenimientoBrailleExpress@gmail.com)

**7. Conclusión**

El mantenimiento regular y la evolución estratégica del sistema Braille Express son esenciales para garantizar que los estudiantes con discapacidad visual puedan acceder a una experiencia educativa más equitativa. A través de este plan se busca asegurar que la aplicación siga siendo confiable, segura, y adaptada a las necesidades reales del entorno educativo. Su constante revisión permitirá al equipo técnico responder de forma proactiva a nuevas exigencias tecnológicas y pedagógicas, garantizando la continuidad y expansión del impacto social del proyecto..