



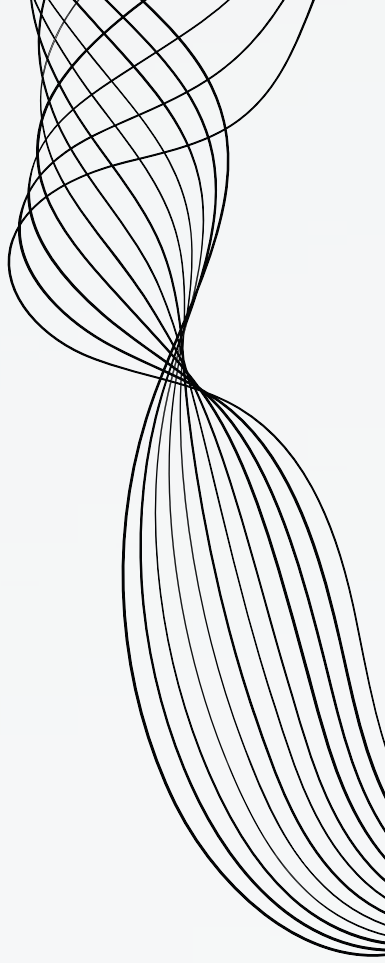
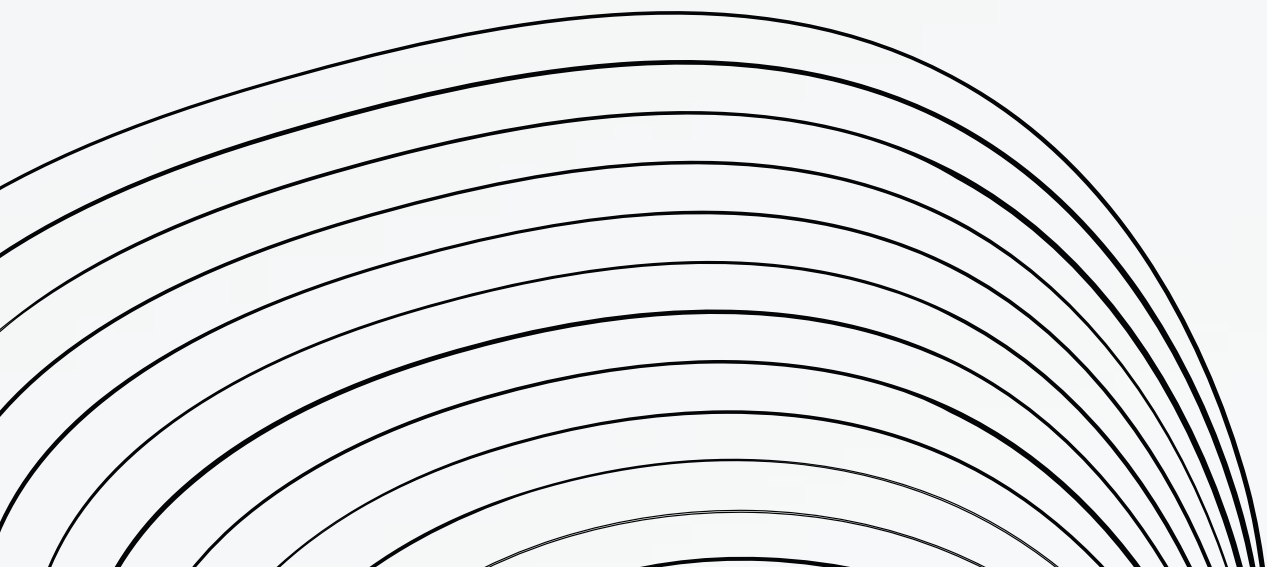
**Universidad
Continental**

**DESARROLLO DE UN DISPOSITIVO
ELECTRÓNICO PARA LA TRADUCCIÓN
DE TEXTO DIGITAL A BRAILLE QUE**

AUTOR:

YLIA JAMILE OCHOA GUTIERREZ

VISIÓN DEL PROYECTO



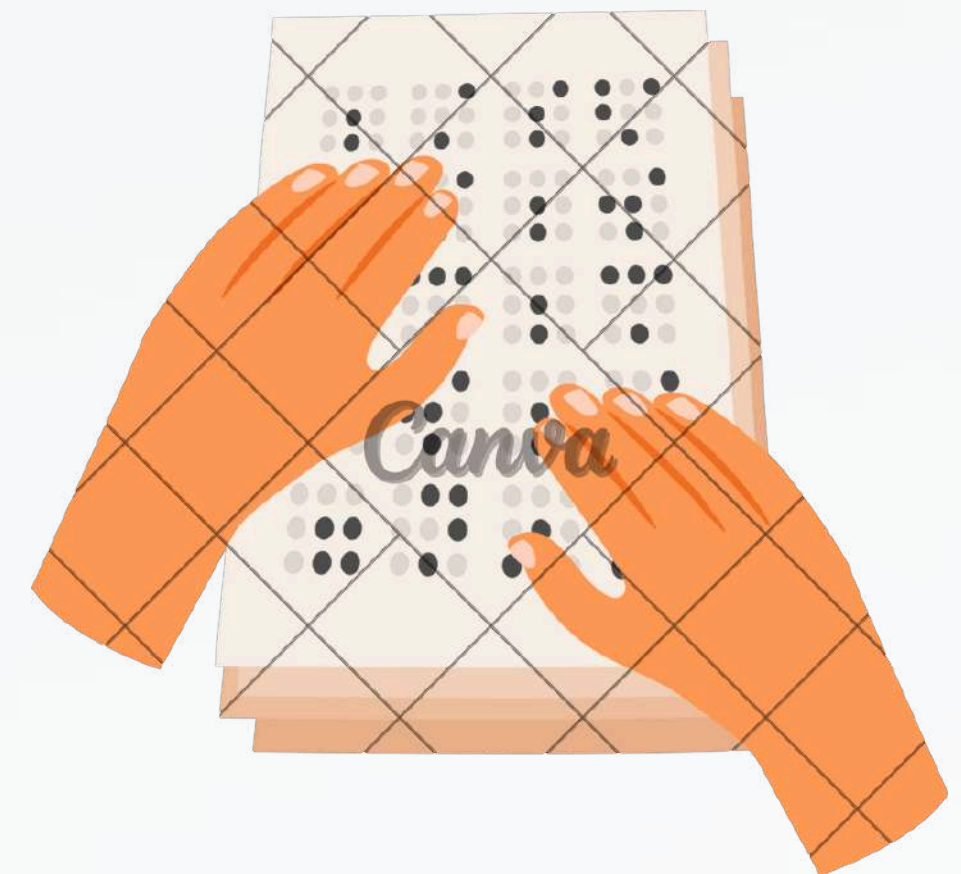
INTRODUCCIÓN

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. Uno de los mayores retos que enfrentan los estudiantes con discapacidad visual es el acceso a materiales educativos accesibles. El desarrollo de un dispositivo electrónico que traduce texto digital a Braille ofrece una solución innovadora para abordar este problema.



PROBLEMA

A pesar de los avances tecnológicos, la mayoría de los recursos educativos digitales no están adaptados para el uso de personas con discapacidad visual, lo que limita su participación en el proceso de aprendizaje y afecta su rendimiento académico.



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

OE 1

Desarrollar una interfaz intuitiva y accesible que permita a los usuarios cargar y traducir textos digitales de manera sencilla y rápida.



OE 2

Diseñar y construir el prototipo de un dispositivo electrónico que permita la traducción de texto digital a Braille de manera rápida y efectiva.

OE 3

Validar el rendimiento y la precisión del dispositivo mediante pruebas experimentales en entornos educativos y sociales.

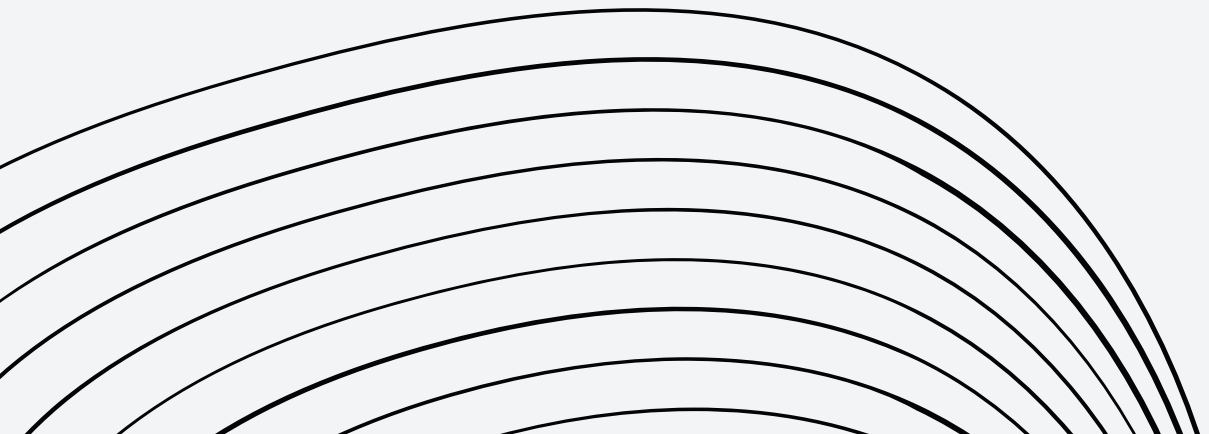
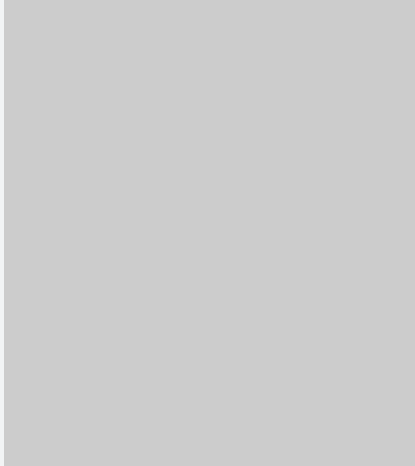


USUARIOS PRINCIPALES

- Personas con discapacidad visual.
- Docentes especializados.
- Centros educativos especializados.
- Bibliotecas para personas con discapacidad visual.
- Organizaciones de personas con discapacidad visual.



BENEFICIOS ESPERADOS

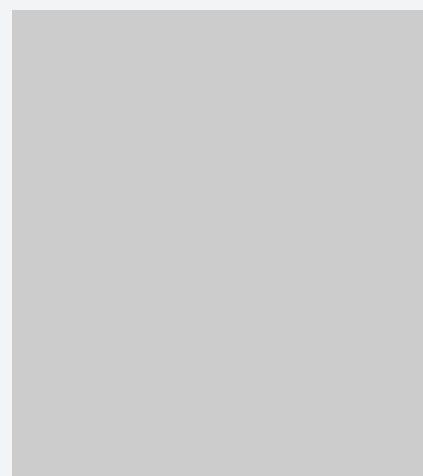
- Acceso a contenidos digitales.
 - Facilidad de uso y adaptación.
 - Aumento de oportunidades educativas y laborales.
 - Inclusión en el ámbito educativo mediante la tecnología.
 - Mejora en la comprensión lectora.
- 
- 

FUNCIONALIDADES ESENCIALES

- Ingreso manual de texto para traducción a Braille.
- Ingreso de texto por voz para traducción a Braille.
- Conectividad Bluetooth con Arduino Nano.
- Control de servomotores para las impresiones de caracteres.
- Panel de control con acceso mediante login.
- Interfaz accesible.
- Salida de los caracteres uno por uno.
- Notificaciones y alertas.
- Actualización del software.
- Autenticación y almacenamiento de configuraciones en Firebase.



FUNCIONALIDADES FUTURAS

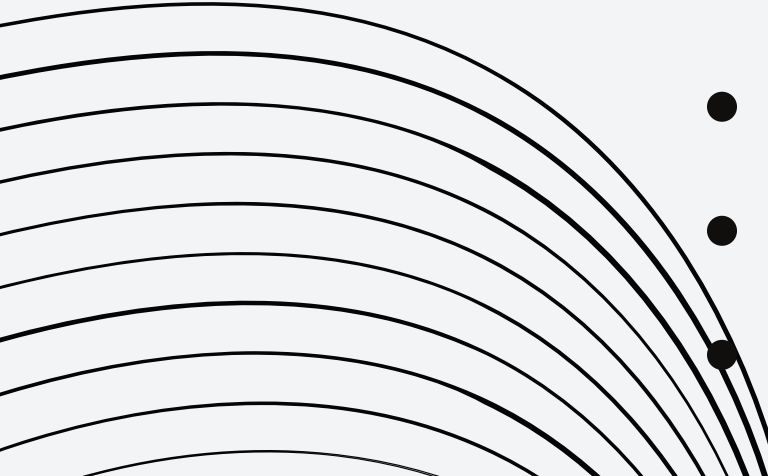
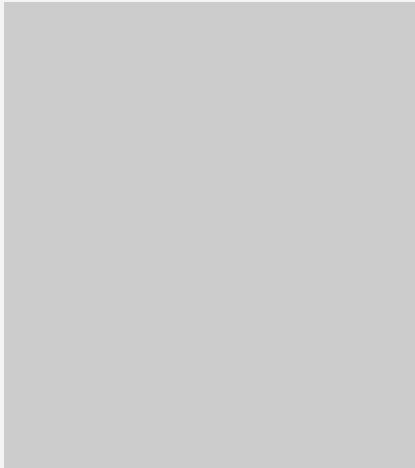
- Implementación de un módulo para traducir documentos completos
 - Traducción bidireccional (texto a Braille y Braille a texto)
 - Implementación de OCR
 - Compatibilidad con dispositivos de impresión Braille.
 - Acceso a bibliotecas digitales
- 



LENGUAJES Y FRAMEWORKS

- Android Studio (Java).
- Arduino IDE (C).
- Comunicación Bluetooth mediante módulo HC-06.
- Firebase.
- Google Text-to-Speech (TTS)

COMPATIBILIDAD

- 
- Compatible con dispositivos Android.
 - Compatible con Arduino Uno.
 - Módulo Bluetooth HC-006.
 - Servomotores SG90.
 - Dispositivos de entrada de voz.
 - Firebase.
- 

RIESGOS

- Problemas de conectividad bluetooth.
- Errores en la precisión de la traducción de voz a texto.
- Desfase en la sincronización entre texto y braille.
- Fallas en los servomotores.
- Problemas de compatibilidad de hardware.
- Errores de programación o bugs.
- Latencia en la impresión de los caracteres.
- Dificultades en la integración de APIs.
- Consumo de batería excesiva.
- Vulnerabilidades de seguridad.
- Problemas de actualización de software.
- Accesibilidad de la interfaz.
- Errores en la gestión de la base de datos en Firebase.

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.

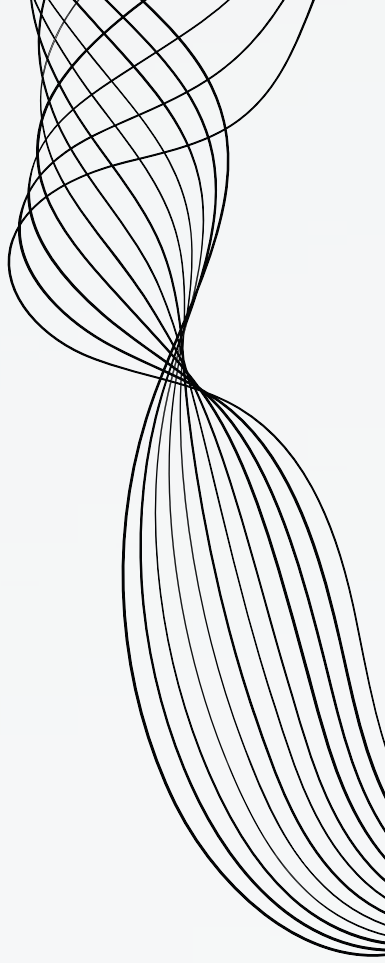
LO QUE INCLUIRÁ

- Desarrollo de la aplicación móvil con traducción de texto digital y de voz.
- Implementación del dispositivo Arduino Nano con servomotores.
- Conectividad Bluetooth entre la aplicación y el dispositivo.
- Interfaz de usuario accesible.
- Sistema de actualizaciones automáticas.
- Personalización del aplicativo.

LO QUE NO INCLUIRÁ

- Traducción de imágenes mediante OCR.
- Integración con otros dispositivos que no sean Android.
- Compatibilidad con otros módulos de conectividad (Wi-Fi, NFC, etc.).
- Soporte multilingüe.
- Reconocimiento automático de textos complejos o fórmulas matemáticas.
- Almacenamiento de textos traducidos en la nube.
- Personalización avanzada del aplicativo.
- Integración con asistentes virtuales (como Google Assistant, Gemini o Alexa).
- Manejo de múltiples dispositivos conectados simultáneamente.

ESPECIFICACIÓN DE REQUISITOS DEL SOFTWARE



REQUISITOS FUNCIONALES

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

REQUISITOS NO FUNCIONALES

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

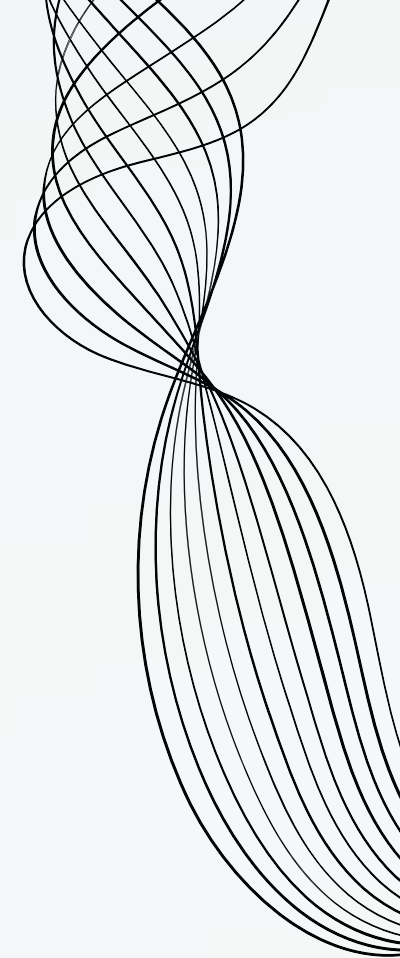
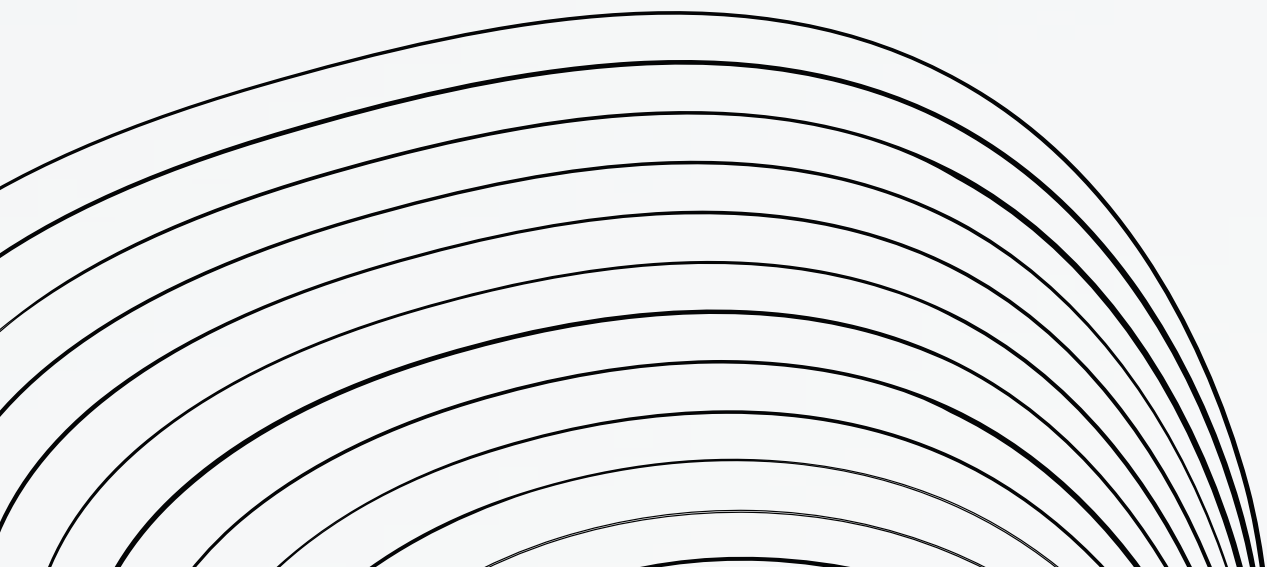
REQUISITOS DE UI

1. Interfaz accesible y minimalista.
2. Uso de colores contrastantes.
3. Fuentes grandes y legibles.
4. Botones grandes y bien espaciados.
5. Opción para ingresar texto manualmente.
6. Botón para activar la entrada de voz.
7. Área donde se muestra el texto original.
8. Traducción automática del texto digital a Braille.
9. Representación gráfica de la traducción en Braille.
10. Indicador de estado de conexión con el Módulo de Bluetooth.
11. Botón “Enviar” para enviar la señal al Arduino.
12. Switch para emparejar con el módulo HC-006.
13. Boton de ingreso por voz.
14. Notificaciones y sonidos.
15. Modo horizontal y vertical.

REQUISITOS DE DE HARDWARE Y SOFTWARE

1. Android Studio 7.0 o superior.
 2. SDK de Android.
 3. Firebase.
 4. Librerías de Procesamiento de Voz.
 5. IDE Arduino.
 6. Librerías para Arduino:
 - o Servo.h – Para el control de los servomotores.
 - o SoftwareSerial.h – Para la comunicación Bluetooth HC-006.
-
- 1.· Arduino Nano.
 - 2.· Módulo Bluetooth HC-06.
 - 3.· 6 Servos SG90.
 - 4.· Shield Nano IO V1.0.
 - 5.· Batería recargable o fuente de alimentación.
 - 6.· Celda Braille mecánica.
 - 7.· Dispositivo Android.

BACKLOG DEL PRODUCTO



ÉPICAS E HISTORIAS DE USUARIO

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

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

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, 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.

Épica 3: Conectividad con el Dispositivo Braille

Historia de Usuario 3.1: Conexión Bluetooth

Como usuario, 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 usuario, 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 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: Notificaciones de acciones y errores

Como usuario, 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 4.2: Notificación de actualizaciones

Como usuario, 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

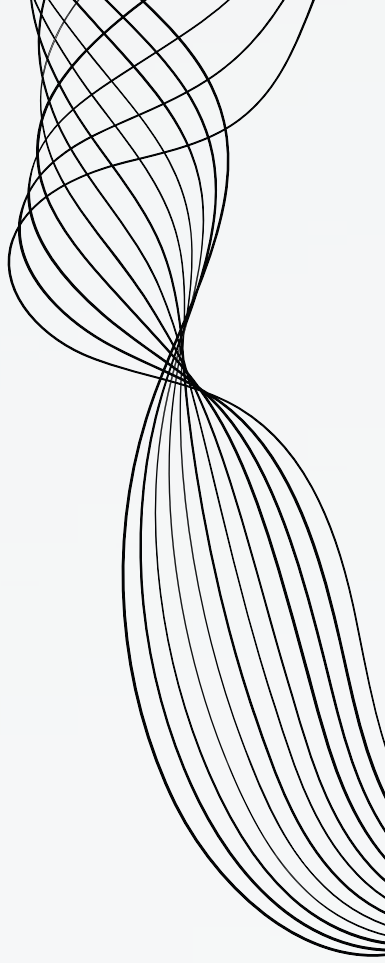
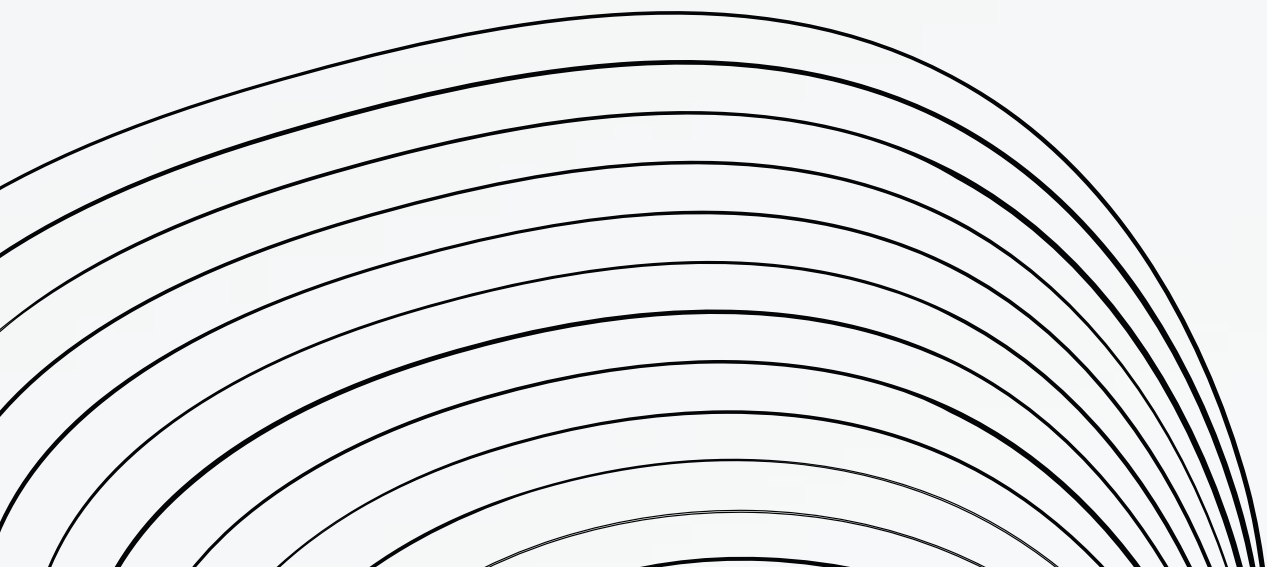
Definición de "Hecho":

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

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	Conexión Bluetooth	Pendiente	10
Alta	Envío de caracteres a Arduino	Pendiente	10
Media	Reinicio manual de Bluetooth	Pendiente	5
Media	Notificaciones de acciones y errores	Pendiente	3
Baja	Notificaciones de actualizaciones	Pendiente	3

SPRINT BACKLOG CON TAREAS Y ESTIMACIONES



SPRINT 1

Se ha considerado realizar las 6 tareas del HI 1.1 y las 2 primeras tareas del HI 1.2, 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 1 durará 64 horas.

ID	Tarea	Responsable	Estimación (Horas)	Estado
1.1.1	Diseñar estructura de la base de datos para almacenar usuarios	Backend Dev	8	Pendiente
1.1.2	Implementar validación de formato de DNI	Backend Dev	8	Pendiente
1.1.3	Implementar almacenamiento seguro de datos	Backend Dev	10	Pendiente
1.1.4	Crear interfaz de registro	Frontend Dev	8	Pendiente
1.1.5	Diseñar flujo de registro	UX/UI Designer	6	Pendiente
1.1.6	Pruebas unitarias y de integración	QA Engineer	8	Pendiente
1.2.1	Diseñar experiencia de usuario para login/logout	UX/UI Designer	6	Pendiente
1.2.2	Diseñar sistema de autenticación	Backend Dev	10	Pendiente

SPRINT 2

Se ha considerado realizar las 4 tareas restantes del HI 1.2 y las 3 primeras tareas del HI 2.1, 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 2 durará 67 horas.

ID	Tarea	Responsable	Estimación (Horas)	Estado
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	7	Pendiente
2.1.1	Diseñar experiencia de usuario para ingreso manual	UX/UI Designer	6	Pendiente
2.1.2	Crear campo de entrada de texto en la interfaz	Frontend Dev	6	Pendiente
2.1.3	Implementar validación de caracteres y signos de puntuación	Backend Dev	12	Pendiente

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 60 horas.

ID	Tarea	Responsable	Estimación (Horas)	Estado
2.1.4	Integrar conversión a Braille	Backend Dev	12	Pendiente
2.1.5	Pruebas funcionales del ingreso de texto	QA Engineer	7	Pendiente
2.2.1	Diseñar experiencia de usuario para ingreso por voz	UX/UI Designer	6	Pendiente
2.2.2	Integrar API de reconocimiento de voz	Backend Dev	12	Pendiente
2.2.3	Implementar edición del texto transcrito	Frontend Dev	8	Pendiente
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	7	Pendiente

SPRINT 4

Se ha considerado realizar las 4 tareas del HI 2.3 y las 2 primeras tareas del HI 3.1, con un total de 6 tareas porque se desea cumplir con la cantidad de horas que se necesita para un Sprint, en este caso el Sprint 4 durará 63 horas.

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	8	Pendiente
2.3.3	Mostrar traducción en la interfaz de usuario	Frontend Dev	7	Pendiente
2.3.4	Pruebas de conversión automática	QA Engineer	8	Pendiente
3.1.1	Detectar dispositivos Bluetooth disponibles	Backend Dev	16	Pendiente
3.1.2	Implementar conexión y gestión de estado	Backend Dev	14	Pendiente

SPRINT 5

Se ha considerado realizar las 2 tareas restantes del HI 3.1, 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.

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
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	8	Pendiente
3.3.1	Implementar opción de reinicio manual	Backend Dev	8	Pendiente

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 51 horas.

ID	Tarea	Responsable	Estimación (Horas)	Estado
3.3.2	Diseñar notificación del estado de reinicio	Frontend Dev	6	Pendiente
3.3.3	Pruebas del reinicio manual	QA Engineer	7	Pendiente
4.1.1	Implementar mensajes emergentes para errores	Frontend Dev	6	Pendiente
4.1.2	Implementar mensajes de confirmación de acciones	Frontend Dev	6	Pendiente
4.1.3	Pruebas de visualización de notificaciones	QA Engineer	5	Pendiente
4.2.1	Implementar verificación de actualizaciones en la nube	Backend Dev	10	Pendiente
4.2.2	Diseñar e implementar notificación de nueva versión	Frontend Dev	6	Pendiente
4.2.4	Pruebas de notificación de actualización	QA Engineer	5	Pendiente

ESTIMACIÓN DE ESFUERZO

Función	Responsable	Horas Totales
UX/UI Design	UX/UI Designer	24
Desarrollo Frontend	Frontend Dev	77
Desarrollo Backend	Backend Dev	190
Pruebas	QA Engineer	72
Total General		363

GRACIAS!

