Registro Diario de Avances – Sprint 5 Braille Express

**Fecha:** 30/05/2025

**Autor:** Ylia Jamile Ochoa Gutierrez

**Versión:** 1.0

# 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 el **diseño y vista del historial,** la **sincronización de datos en Firebase, y** la **detección/conexión Bluetooth** con 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.