Registro Diario de Avances – Sprint 7 Braille Express

**Fecha:** 30/05/2025

**Autor:** Ylia Jamile Ochoa Gutierrez

**Versión:** 1.0

# 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 2.3 — 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 las **8 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.