ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

CONSTRUCCIÓN Y EVOLUCIÓN DE SOFTWARE

ESCUELA
POLITÉCNICA
NACIONAL

Proyecto Primer Bimestre

Transcriptor a Braille vs 1.0

EQUIPO 3:

Sara Guayasamin

Roberth Jacome

Danna Morales

Salma Morales

Profesor:

Evelyn Marcela Mosquera Espinosa

Fecha de entrega:

25/11/2025

Contenidos

Ambiente y Herramientas utilizadas

El proyecto utiliza un conjunto de herramientas modernas organizadas por su función dentro del ambiente de desarrollo.

Lenguaje: Java 17

Gestión del proyecto: Maven

Framework web: Spark Core

Manejo de datos: Librería Gson

Entorno de Desarrollo: Visual Studio Code

Recursos de UI: Google Fonts, Imágenes en línea.

El proyecto como tal sigue la estructura estándar de un proyecto Maven, lo cual facilita su gestión y portabilidad.

```
BRAILLE-APP
|— src
|   └— main
|      └— java
|         |— BrailleTranslator.java
|         |— Main.java
|         └— Server.java
|— target       <-- Contiene los archivos .class compilados y el JAR (generado por Maven)
|— web          <-- Contiene los recursos web/frontend
|   |— index.html
|   └— styles.css
└— pom.xml      <-- Define el proyecto, dependencias (Spark, Gson) y plugins de Maven
```

Ilustración 1 Estructura del Transcriptor braille

Flujo de trabajo y estrategia de ramificación.

Para la colaboración y la gestión de la estabilidad del código, se implementó una estrategia de ramificación basada en Git Flow Simplificado.

Git Flow Simplificado

Esta estrategia garantiza que el código de producción permanezca siempre funcional.

```
main
|—— develop  <-- Rama de Integración. Todo el trabajo nuevo se fusiona aquí.
|   |—— feat/  <-- Rama temporal para NUEVAS funcionalidades.
|   |
|   └—— fix/   <-- Rama temporal para CORRECCIÓN de errores.
```

Ilustración 2 Estructura de la estrategia de ramificación Git flow simplificado

Main: Es la rama estable que contiene únicamente el código aprobado y listo para ser desplegado (producción).

Develop: Es la rama de integración. Todo el trabajo de nuevas *features* y correcciones converge en esta rama antes de pasar a main.

Feat y fix: Son ramas temporales que se crean desde develop para aislar el desarrollo de nuevas funcionalidades o la corrección de errores. Una vez completado el trabajo y pasada la revisión, se fusionan de nuevo en develop.

Flujo de trabajo.

1. **Análisis de Requisitos:** Se inició con el análisis de los requisitos detallados en el PDF de especificación para definir los métodos y las estructuras de datos necesarias (especialmente en BrailleTranslator).

2. **Codificación y Bifurcación:** Cada desarrollador creaba una rama específica (feat/ o fix/) desde develop para codificar sus tareas.

3. **Integración Continua:** Los cambios individuales se fusionaban continuamente en la rama develop mediante *Pull Requests* para garantizar la integración.

4. **Lanzamiento:** Una vez que develop era estable y cumplía con los requisitos, se realizaba la fusión final a main para crear la versión de lanzamiento.