Escuela Politécnica Nacional

Facultad de Ingeniería de Sistemas

Construcción y evolución de Software

Proyecto Braille

Integrantes:

Gabriel Del Valle.

Andreina Pallo.

Luis Guerrero.

AMBIENTE DE DESARROLLO Y ESTRATEGIA DE RAMIFICACIÓN

Versión 1.0

1. Introducción

Este documento describe la configuración necesaria para trabajar con el proyecto, así como las herramientas utilizadas tanto para el frontend como para el backend.

2. Tecnologías Utilizadas

2.1. Frontend

Framework: React con next.js

Lenguaje: TypeScript

Estilos: CSS

Paquetería: npm

Despliegue: Vercel

2.2. Backend

Lenguaje: Java

Framework: Spring Boot

ORM: Spring Data JPA

Base de Datos: PostgreSQL

Build Tool: Maven

Despliegue: Render

3. Requisitos previos

- Node.js 18+
- Npm
- Java 17
- Maven

4. Configuración del entorno local

4.1. Clonar el repositorio

git clone https://github.com/DV-Gabriel/ProyectoConstruccionBraille

4.2. Backend (Spring Boot + Maven)

1. Entrar a la carpeta
2. Configurar Supabase
3. Configurar archivo application.properties

```
spring.datasource.url=jdbc:postgresql://<host>:<puerto>/<base>
spring.datasource.username=<usuario>
spring.datasource.password=<contraseña>

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect

server.port=8080
```

4. Ejecutar Spring Boot
   En bash:
   mvn spring-boot:run

4.3. Frontend

Instalar dependencias:

npm install

Crear .env

```
VITE_API_URL=http://localhost:8080
```

Ejecutar frontend

```
e>npm run dev
```

5. Flujo de trabajo

Este flujo define cómo se desarrollan nuevas funcionalidades en el proyecto usando una estrategia Git Flow Simplificada.

5.1. Actualizar la rama principal

Asegura que trabajas sobre la última versión estable.

5.2. Crear una rama de feature

Cada funcionalidad se desarrolla en una rama independiente:

```
feature/<número-nombre>
```

5.3. Desarrollo del feature

Dentro de esta rama puedes modificar lo necesario:

- frontend
- backend

5.4. Commits claros y push

Realizar commits pequeños y descriptivos y subir a la rama

5.5. Pull Request hacia main

Desde GitHub:

- **Crear un PR**
- **Describir los cambios**

5.6. Merge a main

Se fusiona en main

Vercel (frontend) y Render (backend) despliegan la nueva versión

6. Despliegue

Frontend → Vercel

- Construcción automática desde frontend/
- Detecta Vite
- Variables de entorno desde la UI de Vercel

Backend → Render

- Servicio Web
- Build Command: npm install
- Start Command: npm start
- Variables de entorno configuradas en Render

7. ESTRATEGIA DE RAMIFICACIÓN

La estrategia utilizada para gestionar las ramas del proyecto, basada en un modelo Git Flow Simplificado, ideal para equipos pequeños.

7.1. Ramas Principales

- **main**
  - Contiene la versión estable del proyecto
  - Siempre funcional
  - Desde aquí se generan ramas de desarrollo
  - Base para los despliegues en producción

7.2. Ramas de Trabajo

- **Ramas de Feature**

Cada nueva funcionalidad se desarrolla en una rama independiente con la siguiente convención:

feature/<numero-descripcion-corta>

Ejemplo:

feature/01-españolBraille

- **Rama de Documentación**

Se usa únicamente para:

- Manuales técnicos
- Requisitos
- Documentación de despliegue

7.3. Flujo

1. git checkout main
2. git pull
3. git checkout -b feature/xx-nueva-feature
4. Trabajas en:
   /frontend
   /backend
5. Commit + Push
6. PR hacia main