

Escuela Politecnica Nacional

Facultad de Ingeniería de Sistemas

**Construcción y Evolución de Software**

---

<b>Versión:</b>	1.0
<b>Grupo:</b>	5
<b>Fecha:</b>	Noviembre 2025

---

# Casos de Prueba General

---

Este documento propone un conjunto de pruebas (unitarias, de integración y manuales) para verificar el correcto funcionamiento del proyecto.

## 1) Pruebas unitarias (función `texto_a_braille`)

Recomendado usar `pytest`. AGREGAR IMAGENES

- Caso: letras minúsculas
  - Entrada: `"abcxyz"`
  - Esperado: cada caracter mapeado correctamente según `BRaille_MAP`.
- Caso: números
  - Entrada: `"0123456789"`
  - Esperado: mapeo correcto (ver `BRaille_MAP`).
- Caso: puntuación y espacio
  - Entrada: `"Hola, mundo."`
  - Esperado: comas y puntos convertidos y espacios preservados.
- Caso: caracteres no soportados
  - Entrada: `"@#€"`
  - Esperado: los caracteres no mapeados se devuelven tal cual.
- Caso: cadena vacía
  - Entrada: `""`
  - Esperado: `""` (cadena vacía) o manejo definido.

## Consideraciones

Ejemplo de test (esqueleto): SI ES OPCIONAL TDD

```
from app import texto_a_braille

def test_letras():
    assert texto_a_braille('abc') == '⠠⠠⠠'

def test_vacio():
    assert texto_a_braille('') == ''
```


## 3) Pruebas End-to-End (UI)

Manual o automatizado (Selenium / Playwright):



### Caso E2E-001: Conversión básica de texto

- Abrir <http://localhost:5000>.
- Introducir texto en el [textarea](#).
- **Verificar:** La traducción a texto Braille es instantánea a la par que se introduce texto.
- **Resultado esperado:** El texto Braille aparece en tiempo real.



### Caso E2E-002: Funcionalidad de copiar

- Introducir texto: "Hola mundo".
- Esperar a que se muestre el resultado en Braille.
- Hacer clic en el botón "  Copiar".
- **Verificar:**
  - El botón cambia a " ☒ ¡Copiado!" temporalmente.
  - El botón cambia de color a verde.
  - Después de 2 segundos vuelve al estado original.
- Pegar (Ctrl+V) en un editor de texto.
- **Resultado esperado:** El texto en Braille se pega correctamente.

### Caso E2E-003: Exportar como PNG

- Introducir texto: "Python 2024".
- Esperar a que se muestre el resultado en Braille.
- Hacer clic en el botón "  Exportar PNG".
- **Verificar:**
  - El botón muestra "  Generando..." temporalmente.
  - Se descarga un archivo PNG con nombre formato: [braille-traduccion-YYYY-MM-DD-HH-MM-SS.png](#).
  - El botón cambia a " ☒ ¡Exportado!" en verde.
  - Después de 2 segundos vuelve al estado original.
- Abrir el archivo PNG descargado.
- **Resultado esperado:**
  - La imagen contiene el título "Traducción a Braille".
  - Muestra el texto original.
  - Muestra el texto en Braille con fuente grande.
  - Tiene el footer "Generado por Brailator".
  - Fondo blanco profesional.

### Caso E2E-004: Validación de campos vacíos

- No introducir texto (dejar vacío).
- Intentar hacer clic en "Copiar".
- **Resultado esperado:** Alerta "  No hay texto en Braille para copiar".
- Intentar hacer clic en "Exportar PNG".
- **Resultado esperado:** Alerta "  No hay texto en Braille para exportar".

### Caso E2E-005: Responsividad

- Abrir la aplicación en diferentes tamaños de pantalla (desktop, tablet, mobile).
- **Verificar:**
  - Los botones se reorganizan correctamente en pantallas pequeñas.
  - Los botones ocupan el ancho completo en móviles.
  - Todas las funcionalidades siguen operativas.

## 4) Pruebas de accesibilidad

- Probar con lector de pantalla (NVDA, VoiceOver) y comprobar que las secciones son legibles.
- Comprobar contraste de colores y tamaño de fuente adaptables.

## 5) Pruebas de rendimiento

- Medir latencia para textos de diferentes tamaños (por ejemplo: 1KB, 10KB, 100KB).
- Determinar throughput concurrente con herramientas como `ab` o `wrk` si se despliega detrás de un servidor WSGI.

## 6) Pruebas de seguridad

- Limitar tamaño de payload para prevenir DoS.
- Validar que la API no ejecuta código o realiza operaciones peligrosas con la entrada.

### Checklist de aceptación

#### Funcionalidad Core

- Conversión correcta para letras básicas.
- API devuelve 400 para entradas vacías.
- Conversión en tiempo real funciona correctamente.
- Aplicación arranca en entorno virtual con `pip install -r requirements.txt`.
- Pruebas unitarias y de integración pasan en CI.

#### Funcionalidad de Copiar

- Botón "Copiar" copia el texto Braille al portapapeles.
- Feedback visual correcto (cambio a verde y texto "¡Copiado!").
- Validación de campo vacío funciona (muestra alerta).
- Funciona en Chrome, Firefox, Edge.

#### Funcionalidad de Exportar PNG

- Botón "Exportar PNG" genera la imagen correctamente.
- La imagen contiene título, texto original y traducción Braille.
- El nombre del archivo incluye timestamp.
- La imagen tiene alta calidad (scale: 2).
- Feedback visual correcto durante la generación.
- Validación de campo vacío funciona (muestra alerta).
- La librería `html2canvas` se carga correctamente desde CDN.

## Interfaz de Usuario

- UI muestra y copia correctamente el resultado.
- Botones tienen estados hover correctos.
- Diseño responsivo funciona en móviles y tablets.
- Botones se apilan verticalmente en pantallas pequeñas.
- Iconos y textos de botones son claros y descriptivos.