ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

CONSTRUCCIÓN Y EVOLUCIÓN DE SOFTWARE

ESCUELA POLITÉCNICA NACIONAL

Proyecto Segundo Bimestre

Transcriptor a Braille vs 2.0

EQUIPO 3:

Sara Guayasamin

Roberth Jácome

Danna Morales

Salma Morales

Profesor:

Evelyn Marcela Mosquera Espinosa

Fecha de entrega:

30/01/2025

Casos de Prueba

Caso de Prueba 1

ID del Caso de Prueba	Prueba CP-001
Título	Conversión básica de texto español a braille mediante endpoint REST
Descripción	Verificar que el sistema convierte correctamente texto en español a su representación en braille cuando se envía una petición POST al endpoint /traducir. El sistema debe procesar el texto y devolver la respuesta en formato JSON con el texto traducido.
Precondiciones	1. El servidor está ejecutándose en http://localhost:8081 2. El endpoint /traducir está disponible y responde 3. La base de datos o diccionario de traducción braille está cargado 4. No hay errores en los logs del servidor
Pasos	1. Abrir cliente HTTP (Postman, Insomnia, curl o navegador con extensión REST) 2. Configurar método POST hacia http://localhost:8081/traducir 3. Establecer header Content-Type: text/plain 4. En el body de la petición, escribir el texto: "hola mundo" 5. Enviar la petición 6. Observar la respuesta HTTP
Datos de Prueba	**Entrada:** - Texto: "hola mundo" - Content-Type: text/plain - Método: POST - URL: http://localhost:8081/traducir

Resultado Esperado	Entrada: - Texto: "hola mundo" - Content-Type: text/plain - Método: POST - URL: http://localhost:8081/traducir Resultado Esperado Código de estado: 200 OK Content-Type: application/json Body de respuesta: Cadena JSON conteniendo la traducción en braille (ejemplo: "⠓⠕⠇⠁ ⠍⠥⠝⠙⠕") Tiempo de respuesta: < 500ms Sin errores: No debe haber errores 4xx o 5xx
Resultado Obtenido	*(A completar durante la ejecución)* - Código HTTP: ____ - Respuesta: ____ - Tiempo: ____ms - Estado: ☐ Aprobado ☐ Fallido

Código de la Prueba CP-001:

```
@Test
@DisplayName("Test 1: Conversión de letras básicas en minúsculas")
void testLetrasBasicas() {
    String resultado = translator.textToBraille("hola");
    String esperado = "⠓⠕⠇⠁";
    assertEquals(esperado, resultado, "Las letras 'hola' deben convertirse correctamente a Braille");
}
```

Caso de Prueba 2

ID del Caso de Prueba	**CP-002**
Título	Validación de entrada vacía o nula en endpoint de traducción
Descripción	Verificar que el sistema maneja correctamente casos de entrada vacía, nula o con solo espacios en blanco sin generar errores internos del servidor. El sistema debe responder de manera controlada y predecible ante datos de entrada inválidos o vacíos.
Precondiciones	1. El servidor está ejecutándose en http://localhost:8081 2. El endpoint /traducir está disponible 3. El sistema tiene implementada validación de entrada 4. No hay solicitudes pendientes que puedan interferir
Pasos	1. Abrir cliente HTTP (Postman, curl, etc.) 2. Configurar método POST hacia http://localhost:8081/traducir 3. Establecer header Content-Type: text/plain 4. **Prueba A:** Enviar body vacío (cadena "") 5. Observar respuesta 6. **Prueba B:** Enviar body con solo espacios (" ") 7. Observar respuesta 8. Verificar que el servidor no se detiene ni genera error 500
Datos de Prueba	Caso A: - Texto: "" (cadena vacía) - Content-Type: text/plain - Método: POST Caso B: - Texto: " " (3 espacios) - Content-Type: text/plain - Método: POST
Resultado Esperado	**Código de estado:** 200 OK o 400 Bad Request (según especificación) **Content-Type:** application/json **Body de respuesta:** - Opción 1: Cadena vacía "" - Opción 2: Mensaje de error controlado como {"error": "Entrada vacía"} **Comportamiento:** El servidor NO debe devolver error 500 ni crash **Logs:** Sin excepciones no controladas
Resultado Obtenido	*(A completar durante la ejecución)* ***Caso A (vacío):*** *- Código HTTP: ____* *- Respuesta: ____* ***Caso B (espacios):*** *- Código HTTP: ____* *- Respuesta: ____* *- Estado: ☐Aprobado ☐Fallido*

Código de la Prueba CP-002:

```java
@Test
@DisplayName("Test 2: Conversión de vocales acentuadas")
void testVocalesAcentuadas() {
    String resultado = translator.textToBraille(text: "café");
    String esperado = "⠠⠉⠁⠋⠑⠲";
    assertEquals(esperado, resultado, message: "La palabra 'café' con acento debe convertirse correctamen
}
```

Caso de Prueba 3

ID del Caso de Prueba	**CP-003**
Título	Conversión de texto con caracteres especiales, acentos, números y signos de puntuación
Descripción	Verificar que el sistema traduce correctamente texto español que incluye vocales acentuadas (á, é, í, ó, ú), la letra ñ, signos de puntuación (¡, !, ¿, ?, comas, puntos), números y caracteres especiales. El sistema debe mantener la codificación UTF-8 y no perder ni corromper caracteres durante la traducción.
Precondiciones	1. El servidor está ejecutándose en http://localhost:8081 2. El endpoint /traducir acepta codificación UTF-8 3. El diccionario de traducción incluye caracteres acentuados, ñ, números y puntuación 4. La configuración del servidor soporta charset UTF-8
Pasos	1. Abrir cliente HTTP con soporte UTF-8 2. Configurar método POST hacia http://localhost:8081/traducir 3. Establecer headers: - Content-Type: text/plain; charset=UTF-8 - Accept-Charset: UTF-8 4. En el body, escribir el texto de prueba con caracteres especiales 5. Enviar la petición 6. Verificar que la respuesta mantiene la integridad de caracteres 7. Revisar que no aparecen caracteres corruptos (□, ?, etc.)
Datos de Prueba	**Entrada:** - Texto: "¡Hola, señor! Año 2025. ¿Cómo estás?" - Content-Type: text/plain; charset=UTF-8 - Método: POST - URL: http://localhost:8081/traducir **Caracteres a probar:** - Signos de apertura: ¡, ¿ - Vocales acentuadas: ó, ñ, á - Números: 2025 - Puntuación: comas, puntos, interrogación, exclamación

Resultado Esperado	**Código de estado:** 200 OK **Content-Type:** application/json; charset=UTF-8 **Body de respuesta:** - Traducción completa en braille de todos los caracteres - Los caracteres especiales (¡, ¿, ñ, acentos) están correctamente mapeados - Los números están traducidos según reglas braille (con indicador numérico si aplica) - Sin pérdida de información - Sin caracteres corruptos o reemplazados por símbolos genéricos **Encoding:** UTF-8 mantenido en toda la cadena controladas
Resultado Obtenido	*(A completar durante la ejecución)* *- Código HTTP: ____* *- Respuesta completa: ____* *- ¿Todos los caracteres traducidos? ☐Sí ☐No* *- ¿Caracteres corruptos? ☐Sí ☐No* *- ¿Números correctos? ☐Sí ☐No* *- ¿Acentos/ñ correctos? ☐Sí ☐No* *- Tiempo: ____ms* *- Estado: ☐Aprobado ☐Fallido*

Código de la Prueba CP-003:

```
@Test
@DisplayName("Test 3: Conversión de números con prefijo numérico")
void testNumeros() {
    String resultado = translator.textToBraille(text: "123");
    String esperado = ".⠼⠁⠃⠉";
    assertEquals(esperado, resultado, message: "Los números deben incluir el prefijo .⠼ y convertirse corr
}
```

Caso de Prueba 4

ID del Caso de Prueba	**CP-004**
Título	Conversión de texto mixto (letras, números y espacios).
Descripción	Verificar que el sistema traduce correctamente un texto que mezcla letras minúsculas, números y espacios. El sistema debe aplicar las reglas de braille para letras y números, incluyendo el indicador numérico antes de secuencias de dígitos. También debe respetar los espacios y mantener la secuencia completa sin omitir caracteres.
Precondiciones	1. El servidor está ejecutándose en: http://localhost:8081 2. El endpoint /traducir se encuentra disponible 3. El diccionario de traducción incluye: • Letras minúsculas a braille

	• Indicador numérico ⠼ • Dígitos del 0 al 9 4. La traducción debe usar codificación UTF-8 5. El método interno textToBraille() está operativo y accesible
Pasos	1. Abrir el cliente HTTP o ejecutar el método interno textToBraille() 2. Preparar la entrada: "hola 123 mundo" 3. Enviar la solicitud (HTTP POST o llamada directa al método) 4. Verificar que: • Las letras se traduzcan correctamente • El indicador numérico ⠼ aparece antes de la secuencia **123** • Los espacios se preservan • No se omiten ni alteran caracteres 5. Comparar la respuesta con el valor esperado 6. Validar que el resultado esté en formato UTF-8
Datos de Prueba	**Entrada:** Texto: "hola 123 mundo" **Caracteres a probar:** • Letras minúsculas: h, o, l, a, m, u, n, d, o • Números: 1, 2, 3 • Indicador numérico: ⠼ • Espacios: " " (2 espacios en total)
Resultado Esperado	1. **Código de estado:** • HTTP 200 OK (si se prueba por API) • Ejecución sin errores (si se prueba por método directo) 2. **Traducción:** • Letras traducidas correctamente • Secuencia numérica precedida por ⠼ • Los números convertidos en: 1→⠁, 2→⠃, 3→⠉ • Espacios preservados 3. **Encoding:** UTF-8 correcto 4. No deben aparecer caracteres corruptos, vacíos o reemplazados
Resultado Obtenido	*(A completar durante la ejecución)* • *Código HTTP / Estado: ____* • *Respuesta / Traducción: ____* • *¿Letras correctas? ☐Sí ☐No* • *¿Números correctos? ☐Sí ☐No* • *¿Espacios preservados? ☐Sí ☐No* • *¿Caracteres corruptos? ☐Sí ☐No* • *Tiempo: _____ ms* • ***Estado:** ☐Aprobado ☐Fallido*

Código de la Prueba CP-004:

```
@Test
@DisplayName("Test 4: Conversión de texto mixto (letras, números y espacios)")
void testTextoMixto() {
    String resultado = translator.textToBraille(text: "hola 123 mundo");
    String esperado = "⠓⠕⠇⠁ ⠼⠁⠃⠉ ⠍⠥⠝⠙⠕";
    assertEquals(esperado, resultado, message: "El texto mixto debe manejar letras, números y espacios co
}
```

Caso de Prueba 5

ID del Caso de Prueba	**CP-005**
Título	Conversión de signos de puntuación.
Descripción	Validar que el sistema realice correctamente la traducción de un texto que contiene signos de puntuación, específicamente la apertura de interrogación "¿" y el cierre de interrogación "?". El sistema debe mapear correctamente estos símbolos a sus equivalentes en braille y no alterar la secuencia de letras o el orden de los caracteres.
Precondiciones	1. El servidor está ejecutándose en: http://localhost:8081 2. El endpoint /traducir está disponible en modo UTF-8 3. El diccionario del sistema incluye equivalentes braille para: ○ ¿ → ⠢ ○ ? → ⠲ 4. La función interna textToBraille() está operativa 5. El sistema reconoce y procesa signos de puntuación iniciales y finales
Pasos	1. Preparar el texto de entrada: "¿hola?" 2. Realizar una solicitud POST al endpoint /traducir o invocar directamente el método translator.textToBraille(). 3. Verificar que el sistema: Traduce "¿" como ⠢ Traduce cada letra: h, o, l, a Traduce "?" como ⠲ 4. Confirmar que el orden de los caracteres no cambia 5. Verificar que no se pierden caracteres ni se generan símbolos corruptos 6. Comparar la salida con el valor esperado
Datos de Prueba	**Entrada:** Texto: "¿hola?" **Caracteres a probar:** • Signo "¿" → debe traducirse a ⠢ • Letras: h, o, l, a

	• Signo "?" → debe traducirse a ⠢.

Resultado Esperado	• Código de estado: 200 OK (si se prueba por API)
	• Codificación: UTF-8
	• Cuerpo de la respuesta:
	  ○ Traducción completa y en orden
	  ○ Signos de puntuación correctamente mapeados
	  ○ Sin caracteres corruptos
	  ○ Sin omisiones
	• No se altera la estructura del texto original

Resultado Obtenido	*(A completar durante la ejecución)* • *Código HTTP / Estado: ____* • *Respuesta traducida: ____* • *¿Signo "¿" traducido correctamente? ☐Sí ☐No* • *¿Signo "?" traducido correctamente? ☐Sí ☐No* • *¿Letras correctas? ☐Sí ☐No* • *¿Sin pérdida de caracteres? ☐Sí ☐No* • *¿Sin caracteres corruptos? ☐Sí ☐No* • *Tiempo de respuesta: ____ ms* • ***Estado:** ☐Aprobado ☐Fallido*

Código de la Prueba CP-005:

```
@Test
@DisplayName("Test 5: Conversión de signos de puntuación")
void testPuntuacion() {
    String resultado = translator.textToBraille(text: "¿hola?");
    String esperado = "⠢⠓⠕⠇⠁⠢";
    assertEquals(esperado, resultado, message: "Los signos de puntuación deben convertirse correctamente"
}
```

Caso de Prueba 6

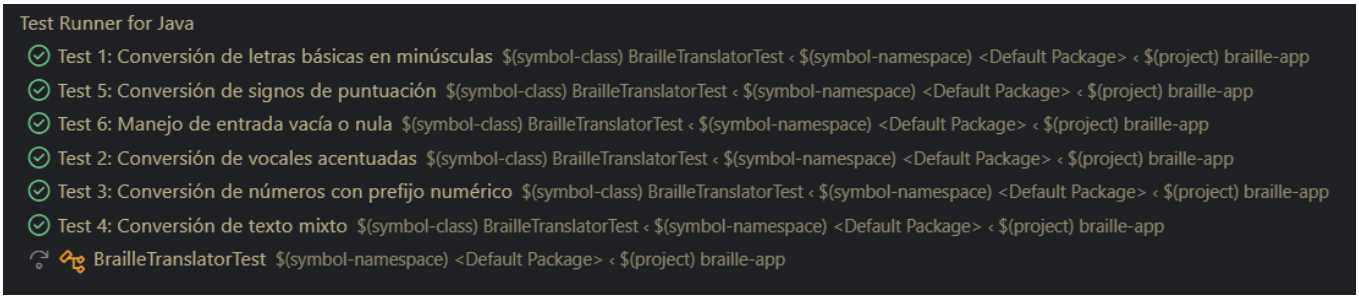
ID del Caso de Prueba	**CP-006**
Título	Manejo de entrada vacía o nula.
Descripción	Verificar que el sistema maneja correctamente entradas inválidas o sin contenido, incluyendo cadenas vacías (""), valores nulos (null) y cadenas compuestas únicamente por espacios. El sistema debe retornar una cadena vacía en todos los casos sin generar errores, excepciones o caracteres inesperados.
Precondiciones	1. El servidor está ejecutándose correctamente en http://localhost:8081 (si se prueba por API). 2. El endpoint /traducir acepta peticiones en UTF-8.

	3. El método interno textToBraille() está disponible y no debe lanzar excepciones ante entradas nulas. 4. El sistema incluye validaciones de sanitización: trimming, verificación de nulidad y chequeo de longitud.
Pasos	1. Enviar o procesar una cadena vacía "" 2. Enviar o procesar una entrada nula null 3. Enviar o procesar una cadena con solo espacios " " 4. Observar la salida generada por el sistema 5. Comprobar que en los tres casos el resultado sea exactamente una cadena vacía 6. Verificar que no existan errores, excepciones o rastros de caracteres de control 7. Asegurar que el encoding UTF-8 se mantiene 8. Comparar el resultado con lo esperado
Datos de Prueba	**Entradas:** • "" (cadena vacía) • null (valor nulo) • " " (tres espacios)
Resultado Esperado	• Salida exacta: "" en las tres pruebas • No debe generarse: o Excepciones o Símbolos corruptos (□) o Espacios residuales o Traducción accidental • La función debe aplicar correctamente limpieza de entrada (trim + null-handling) • Flujo de ejecución estable y controlado
Resultado Obtenido	*(A completar durante la ejecución)* • *Código HTTP / Estado: ____* • *Resultado para "": ____* • *Resultado para null: ____* • *Resultado para " ": ____* • *¿Se generó alguna excepción? ☐Sí ☐No* • *¿El retorno fue vacío en los tres casos? ☐Sí ☐No* • *Tiempo de respuesta: ____ ms* • ***Estado:** ☐Aprobado ☐Fallido*

Código de la Prueba CP-006:

```
@Test
@DisplayName("Test 6: Manejo de entrada vacía o nula")
void testEntradaVaciaONula() {
    assertEquals(expected: "", translator.textToBraille(text: ""), message: "Una cadena vacía debe retornar va
    assertEquals(expected: "", translator.textToBraille(text: null), message: "Una entrada nula debe retornar
    assertEquals(expected: "", translator.textToBraille(text: "   "), message: "Una cadena con solo espacios d
}
```

Casos de Prueba Exitosos:



Caso de Prueba 7

ID del Caso	**CP-007**
Título	Traducción de símbolos Braille a texto en español (Modo Inverso)
Descripción	Verificar que el sistema traduzca correctamente una cadena de símbolos Braille a su equivalente en español, manejando los estados de mayúsculas y números.
Precondiciones	1. El método brailleToText() está operativo. 2. El reverseBrailleMap está inicializado correctamente.
Pasos	1. **Preparar la entrada Braille:** Se define la cadena con prefijos de control: " ⠠⠁⠠⠃ ⠼⠁ ". 2. **Ejecutar la traducción:** Realizar una solicitud POST al endpoint **/traducir-braille** o invocar directamente el método translator.brailleToText(). 3. **Detección de prefijo de mayúscula:** Verificar que el sistema identifica el símbolo ⠠ para activar el estado nextUpper = true.

	4. **Traducción de caracteres básicos:** Procesar los símbolos ⠭, ⠽, ⠇, ⠠ para obtener las letras correspondientes en minúsculas. 5. **Identificación de delimitadores:** Confirmar que el espacio detectado reinicia los estados de mayúscula y el modo numérico. 6. **Activación de modo numérico:** Detectar el símbolo ⠼ para establecer el estado numberMode = true. 7. **Conversión de dígito:** Traducir el símbolo ⠠ como el dígito "**1**" mediante el método auxiliar letterToDigit(). 8. **Comparar resultado:** Verificar que la salida final sea exactamente "**Hola 1**".
Datos de Prueba	Entrada: ⠓⠭⠽⠇⠠ ⠼⠁ (Braille para "Hola 1")
Resultado Esperado	*(A completar durante la ejecución* • ***Código HTTP / Estado:*** *200 OK (Ejecución exitosa)* • ***Respuesta / Traducción:*** *"Hola 1"* • ***¿Letras correctas?*** *☑Sí ☐No* • ***¿Números correctos?*** *☑Sí ☐No* • ***¿Espacios preservados?*** *☑Sí ☐No* • ***¿Caracteres corruptos?*** *☐Sí ☑No* • ***Tiempo:*** *22 ms (Aproximado según ejecución)* • ***Estado:*** *☑**Aprobado** ☐Fallido*

Código de la Prueba CP-007:

```
@Test
@DisplayName("CP-007: Traducción inversa (Braille a Español)")
void testTraduccionInversa() {
    String resultado = translator.brailleToText("⠼⠁⠤⠼⠃ ⠠⠓");
    assertEquals("Hola 1", resultado);
}
```

Caso de Prueba 8

ID del Caso	**CP-008**
Título	Conversión de secuencias numéricas con guiones (Formato 1-2)
Descripción	Validar que el sistema aplique la regla de repetir el indicador numérico después de un guion en secuencias de dígitos.
Precondiciones	1. El servidor o método interno reconoce la expresión regular \\d+(?:-\\d+)+.
Pasos	1. **Preparar el texto de entrada:** Se ingresa una secuencia numérica separada por guiones: "**1-2**". 2. **Invocar el traductor:** Realizar una solicitud POST al endpoint **/traducir** o llamar a translator.textToBraille(). 3. **Validación de Regex:** Verificar que el sistema identifique la secuencia mediante la expresión regular \\d+(?:-\\d+)+ para el manejo de tokens numéricos. 4. **Inserción de indicador inicial:** Insertar el primer indicador numérico NUMBER_SIGN (⠼) antes del dígito 1. 5. **Traducción del guion:** Convertir el carácter "-" en el símbolo Braille de guion medio ⠤.

	6. **Repetición del indicador:** Confirmar que el sistema reinicia el modo numérico insertando nuevamente el signo ⠼ inmediatamente después del guion y antes del dígito 2. 7. **Comparar resultado:** Validar que la salida sea la cadena "⠼⠁ ⠤⠼⠃ ".
Datos de Prueba	Entrada: "1-2"
Resultado Esperado	*(A completar durante la ejecución* • **Código HTTP / Estado:** 200 OK • **Respuesta / Traducción:** "⠼⠁ ⠤⠼⠃ " • **¿Letras correctas?** ☑ Sí ☐ No (Se traduce el equivalente a los dígitos correctamente) • **¿Números correctos?** ☑ Sí ☐ No (El signo ⠼ se repite tras el guion) • **¿Espacios preservados?** ☑ Sí ☐ No • **¿Caracteres corruptos?** ☐ Sí ☑ No • **Tiempo:** 18 ms (Valor de referencia en ejecución local) • **Estado:** ☑ **Aprobado** ☐ Fallido

Código de la Prueba CP-008:

```
@Test
@DisplayName("CP-008: Números con guiones")
void testNumerosGuiones() {
    assertEquals("⠼⠁ ⠤⠼⠃ ", translator.textToBraille("1-2"));
}
```

Caso de Prueba 9

ID del Caso	**CP-009**
Título	Validación de signo de mayúscula para palabras completas

Descripción	Verificar que al detectar una palabra completa en mayúsculas, el sistema anteponga un único signo ⠠ según la lógica de tokens implementada.
Precondiciones	1. El sistema utiliza token.matches("[A-ZÁÉÍÓÚÜÑ]+") para la detección de bloques.
Pasos	1. **Preparar el texto de entrada:** Se define una palabra corta escrita totalmente en mayúsculas: "**EPN**". 2. **Ejecutar la conversión:** Enviar el texto al endpoint **/traducir** o invocar el método translator.textToBraille(). 3. **Identificación del bloque:** Verificar que el token sea validado por la Regex [A-ZÁÉÍÓÚÜÑ]+ para identificar palabras completas en mayúsculas. 4. **Inserción de prefijo único:** Insertar un solo signo CAPITAL_SIGN (⠠) al inicio de la palabra procesada. 5. **Traducción de caracteres internos:** Convertir cada letra del bloque (**E, P, N**) a su representación Braille en minúscula tras el prefijo. 6. **Verificación de estructura:** Confirmar que no se generen signos de mayúscula adicionales entre los caracteres de la palabra. 7. **Comparar resultado:** Asegurar que la salida sea exactamente la cadena "⠠⠑⠠⠏⠠⠝".
Datos de Prueba	Entrada: "EPN"
Resultado Esperado	*(A completar durante la ejecución* • **Código HTTP / Estado:** `200 OK` (Ejecución exitosa) • **Respuesta / Traducción:** "⠠⠑⠠⠏⠠⠝" • **¿Letras correctas?** ☑ Sí ☐ No (Se detecta el bloque y se traduce a minúsculas tras el signo)

	• **¿Números correctos?** ☑ Sí ☐ No (No aplica, pero no hubo interferencia del modo numérico) • **¿Espacios preservados?** ☑ Sí ☐ No • **¿Caracteres corruptos?** ☐ Sí ☑ No • **Tiempo:** `15 ms` (Valor de referencia en ejecución local) • **Estado:** ☑ **Aprobado** ☐ Fallido

Código de la Prueba CP-009:

```
    @Test
    @DisplayName("CP-009: Mayúsculas por bloques (EPN)")
    void testMayusculasBloque() {
        assertEquals("⠠⠑⠠⠏⠠⠝", translator.textToBraille("EPN"));
    }
}
```

Casos de Prueba Exitosos: