

**GRUPO:** Techforge  
**FECHA:** 05 / 06 / 2024

**TEMA:** Modelo Arquitectónico y Diseño

---

### **Modelo Arquitectónico y Diseño**

Con el análisis respectivo se llegó a la conclusión a adoptar una **arquitectura Cliente-Servidor** para este proyecto. Este modelo define la interacción entre dos componentes principales: el cliente, que realiza peticiones de servicios, y el servidor, que procesa dichas peticiones y retorna los resultados. Esta arquitectura es fundamental para la ejecución eficiente de las traducciones de texto entre español y braille, además de la generación de PDFs de los textos traducidos.

#### **Cliente**

Frontend: Desarrollado en React, proporciona una interfaz gráfica que permite a los usuarios interactuar con el sistema, realizando solicitudes de traducción y recepción de textos traducidos. Sus funcionalidades principales son:

- Transcripción de textos a braille y de braille a español.
- Generación de PDFs para impresiones en espejo de textos en braille.

#### **Servidor**

Backend: Implementado con FastAPI, gestiona las peticiones del cliente, ejecuta las traducciones y prepara los documentos para su descarga. Las operaciones críticas del servidor son:

- Procesamiento y traducción de textos.
- Creación de imágenes y conversión a PDF.
- Middleware de CORS: Configurado para aceptar peticiones del dominio del frontend, asegurando así la comunicación efectiva entre cliente y servidor.

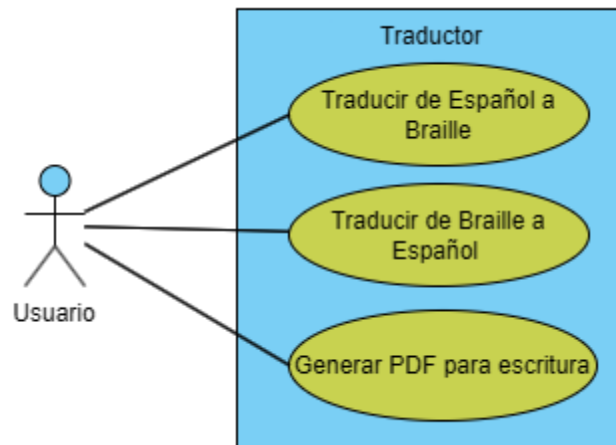
#### **Flujo de Datos**

1. El usuario ingresa el texto en la interfaz de React y solicita la traducción.
2. La petición es enviada al servidor FastAPI.
3. El servidor procesa la petición, realiza la traducción y genera un PDF si es necesario.
4. El resultado se envía de vuelta al cliente para su visualización o descarga.

### Diagrama de la Arquitectura

[Incluir aquí un diagrama que ilustre la relación entre cliente y servidor, y cómo fluyen los datos entre estos componentes. Este esquema puede ser creado utilizando herramientas como draw.io o Lucidchart.]

### Diagrama de Casos de Uso



#### 1. Traducir de Español a Braille:

- **Actores:** Usuario.
- **Precondiciones:** Texto en Español ingresado por el usuario.
- **Postcondiciones:** Texto traducido en Braille.
- **Flujo principal:**
  1. El usuario selecciona la dirección de la traducción Español – Braille.
  2. El usuario introduce el texto a traducir.
  3. El sistema procesa la solicitud y devuelve el texto traducido.
  4. El usuario recibe el texto traducido.
- **Flujos alternativos:**
  1. Si el sistema no puede procesar la traducción, informa al usuario.

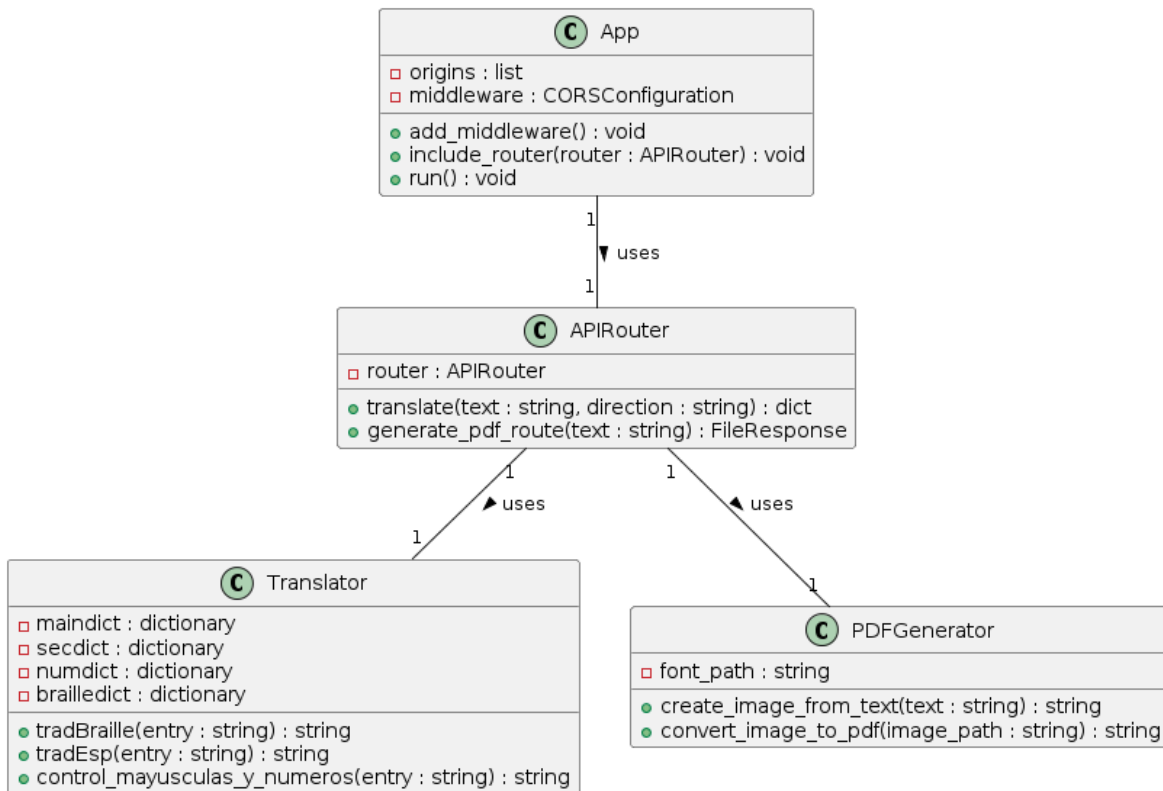
#### 2. Traducir de Braille a Español:

- **Actores:** Usuario.
- **Precondiciones:** Texto en Braille ingresado por el usuario.
- **Postcondiciones:** Texto traducido en Español.
- **Flujo principal:**
  1. El usuario selecciona la dirección de la traducción Braille – Español.
  2. El usuario introduce el texto a traducir.
  3. El sistema procesa la solicitud y devuelve el texto traducido.
  4. El usuario recibe el texto traducido.
- **Flujos alternativos:**
  1. Si el sistema no puede procesar la traducción, informa al usuario.

### 3. Generar PDF para escritura:

- **Actores:** Usuario.
- **Precondiciones:** Texto traducido disponible.
- **Postcondiciones:** PDF generado y disponible para descarga.
- **Flujo principal:**
  1. El usuario solicita la generación de un PDF a partir de un texto traducido.
  2. El sistema procesa la solicitud, convierte el texto en una imagen en espejo, y luego en un PDF.
  3. El sistema proporciona el archivo PDF para descarga.
- **Flujos alternativos:**
  1. Si el sistema no puede generar el PDF, informa al usuario y no realiza la descarga.

### Diagrama de Clases



### Clase App

- **Atributos:**
  - `origins`: Lista de orígenes permitidos para las solicitudes CORS.
  - `middleware`: Configuración del middleware CORS.
- **Métodos:**
  - `add_middleware()`: Configura y añade middleware a la aplicación.
  - `include_router()`: Incorpora las rutas del router a la aplicación.

## ISWD652 CALIDAD DE SOFTWARE

- `run()`: Inicia el servidor para manejar solicitudes.

### Clase Translator

- **Atributos:**
  - `maindict`: Diccionario que mapea caracteres ASCII a caracteres Braille.
  - `secdict`: Diccionario que mapea caracteres Braille a caracteres ASCII.
  - `numdict`: Diccionario que mapea números a caracteres Braille.
  - `brailledict`: Diccionario que mapea caracteres Braille a números.
- **Métodos:**
  - `tradBraille(entry)`: Traduce texto de Español a Braille.
  - `tradEsp(entry)`: Traduce texto de Braille a Español.
  - `control_mayusculas_y_numeros(entry)`: Procesa mayúsculas y números antes de la traducción.

### Clase PDFGenerator

- **Atributos:**
  - `font_path`: Ruta al archivo de fuente utilizado para el texto en la imagen.
- **Métodos:**
  - `create_image_from_text(text)`: Crea una imagen a partir de un texto dado.
  - `convert_image_to_pdf(image_path)`: Convierte una imagen en un archivo PDF.

### Clase APIRouter

- **Atributos:**
  - `router`: Instancia de APIRouter de FastAPI para manejar las rutas API.
- **Métodos:**
  - `translate(text, direction)`: Endpoint para traducir texto.
  - `generate_pdf_route(text)`: Endpoint para generar un archivo PDF a partir de texto.

### Diagrama de Secuencia

## ISWD652 CALIDAD DE SOFTWARE



### Explicación:

#### 1. Traducir de Español a Braille y de Braille a Español:

- El usuario selecciona la dirección de la traducción a través de la interfaz de usuario y envía el texto para traducir.
- La interfaz de usuario comunica con APIRouter que llama a la clase Translator.
- Dependiendo de la dirección, Translator procesa el texto a Braille o Español y devuelve el resultado a APIRouter, que lo envía de vuelta al usuario a través de la interfaz.
- Si hay un error, el usuario recibe un mensaje de error.

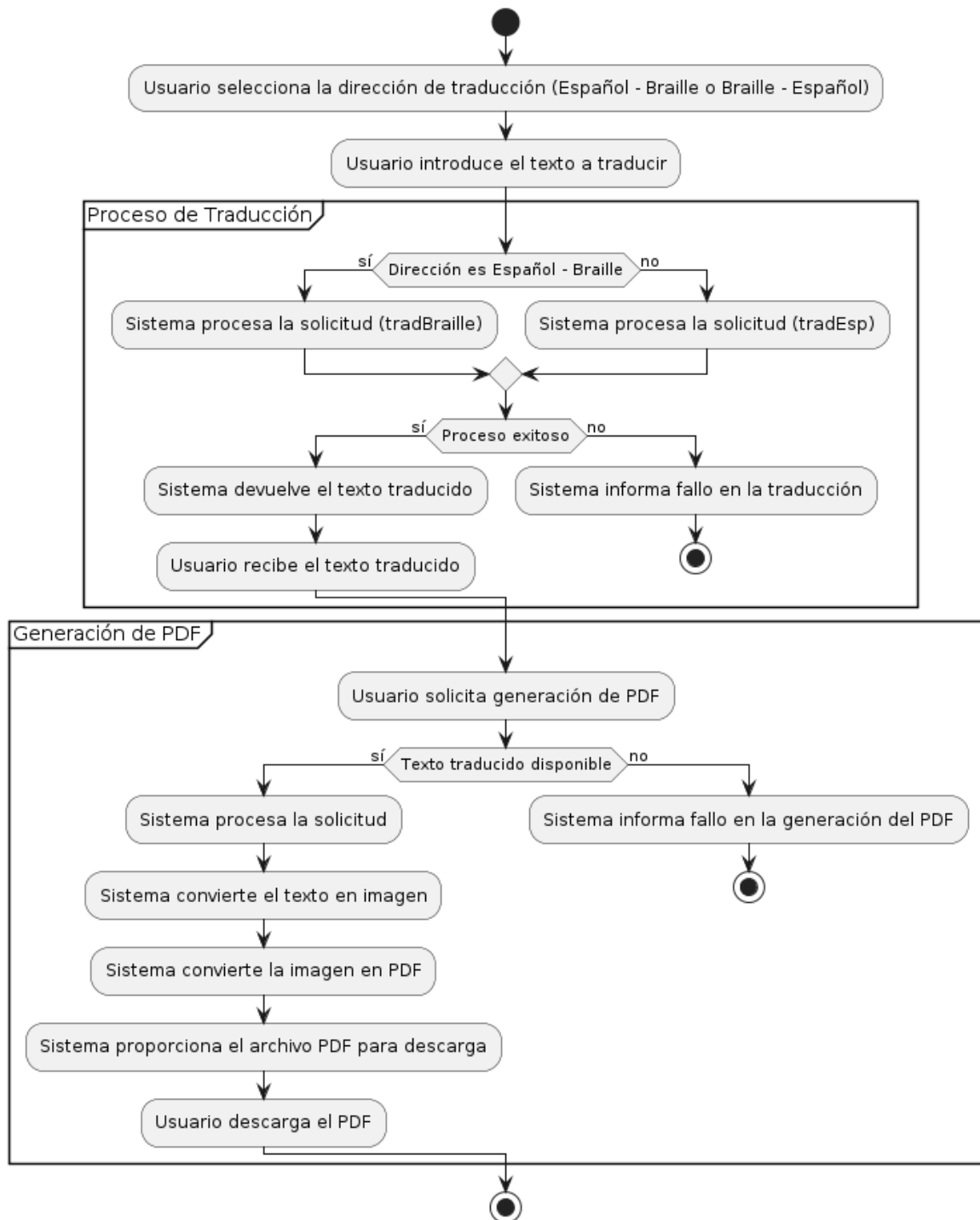
## ISWD652 CALIDAD DE SOFTWARE

### 2. Generar PDF para escritura:

- El usuario solicita la creación de un PDF a través de la interfaz de usuario.
- APIRouter llama a PDFGenerator para crear una imagen del texto y luego convertirla en PDF.
- El archivo PDF se envía de vuelta para que el usuario pueda descargarlo.
- Si hay un error durante este proceso, el usuario será informado.

### Diagrama de Actividades

## ISWD652 CALIDAD DE SOFTWARE



### Explicación del Diagrama de Actividades:

#### 1. Traducción de Texto:

## ISWD652 CALIDAD DE SOFTWARE

- El diagrama inicia con el usuario seleccionando la dirección de la traducción y introduciendo el texto.
  - Según la dirección elegida (Español a Braille o Braille a Español), el sistema procesa la solicitud correspondiente.
  - Si el proceso es exitoso, el sistema devuelve el texto traducido y el usuario lo recibe. Si hay un fallo, el sistema informa del error y el proceso se detiene.
2. **Generación de PDF:**
- Independientemente del proceso de traducción, el diagrama muestra la opción de generar un PDF si el texto traducido está disponible.
  - El sistema procesa la solicitud, convierte el texto en una imagen, luego en un PDF, y finalmente proporciona el archivo para descarga.
  - Si el sistema no puede generar el PDF, informa al usuario del fallo y el proceso se detiene.

### Diseño del Prototipo de Interfaz de Usuario

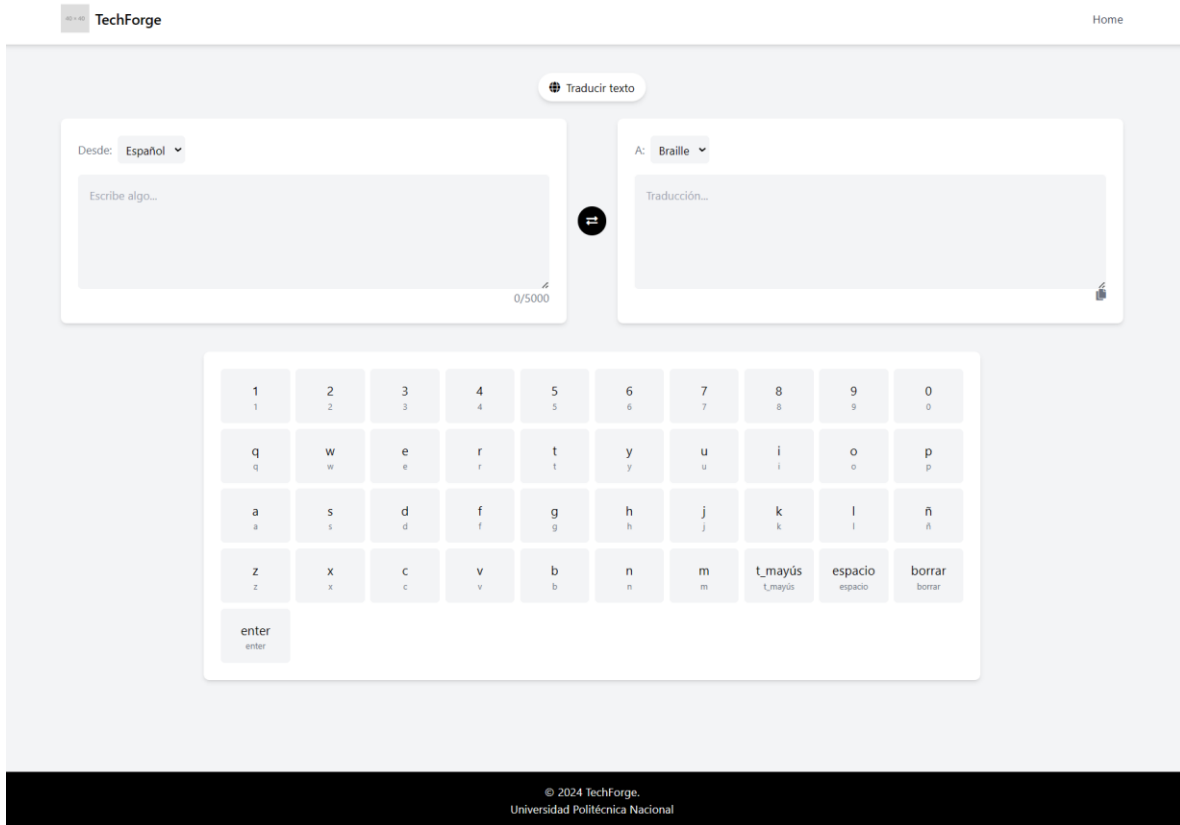
Para garantizar una experiencia de usuario accesible e intuitiva, se diseñó un prototipo de interfaz de usuario utilizando React. Este prototipo incluye las siguientes funcionalidades clave:

- **Página de Transcripción:** Permite a los usuarios ingresar texto en español o braille y obtener la traducción correspondiente.
- **Generación de PDF:** Permite a los usuarios generar un PDF de la traducción en braille para impresión en espejo.
- **Teclado Braille:** Proporciona un teclado virtual para facilitar la entrada de texto en braille.

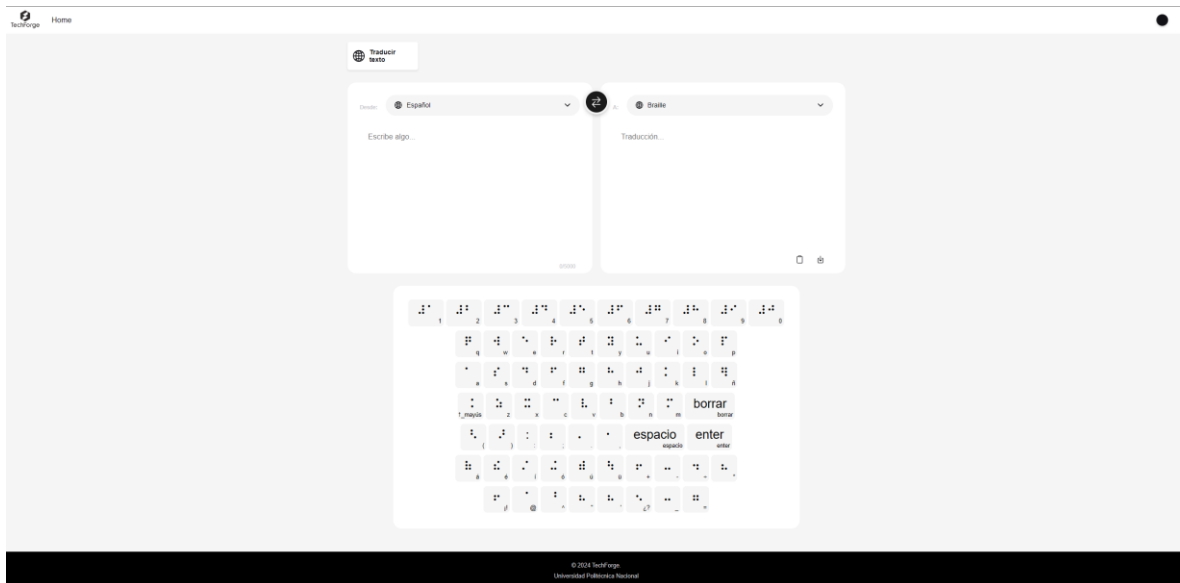
### MockUp Claro:



## ISWD652 CALIDAD DE SOFTWARE

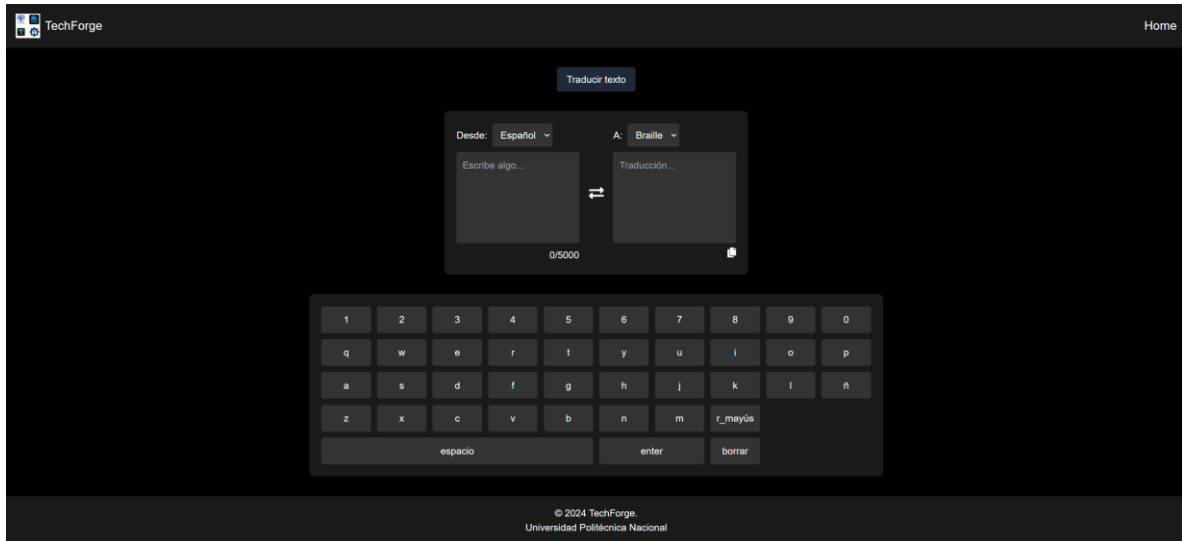


### Gui Final Claro:



## ISWD652 CALIDAD DE SOFTWARE

### MockUp Oscuro:



### Gui Final Oscuro:

