ESCUELA POLITÉCNICA NACIONAL

ISWD652 CALIDAD DE SOFTWARE

INTEGRANTES: Eddy Arias
Fredviner Bailón
Max Carrión
Milton Pástor
Jonathan Poaquiza

FECHA: 30-06-2024

TEMA: Historias de usuario

Historias de Usuario

Nro: HU-001	**Título:** Configuración del Entorno de Desarrollo con Node.js y NPM	**Prioridad:** (Alta) **Estimación:** (4 horas)

Historia de usuario:

Como **desarrollador** quiero **configurar el entorno de desarrollo con Node.js y NPM**, para **asegurarme de que todas las dependencias necesarias están instaladas y el entorno de desarrollo está listo para trabajar**.

Sabré que he terminado cuando se cumplan todos los criterios de aceptación:

- **Instalación de Node.js y NPM:** Dado que el entorno de desarrollo está configurado, cuando se ejecuta node -v y npm -v, entonces se deben mostrar las versiones correctas de Node.js y NPM. [ESCENARIO IDEAL]
- **Instalación de dependencias del proyecto:** Dado que se ejecuta npm install, cuando se completa la instalación, entonces todas las dependencias listadas en package.json deben estar instaladas correctamente. [ESCENARIO IDEAL]

Tareas de implementación:

1. **Descargar e instalar Node.js y NPM en el entorno de desarrollo:**
    - **Estimación:** 1 hora

- **Descripción:** Visitar la página oficial de Node.js, descargar el instalador y seguir las instrucciones para instalar Node.js y NPM.

2. **Crear o actualizar el archivo package.json con las dependencias necesarias:**
   - **Estimación:** 1 hora
   - **Descripción:** Inicializar un nuevo proyecto Node.js si es necesario (npm init), y añadir las dependencias necesarias en package.json (por ejemplo, jest, webpack, babel, etc.).
3. **Ejecutar npm install para instalar todas las dependencias del proyecto:**
   - **Estimación:** 1 hora
   - **Descripción:** Ejecutar el comando npm install en la terminal para instalar todas las dependencias listadas en package.json.
4. **Verificar que todas las dependencias se instalen correctamente y que no haya errores:**
   - **Estimación:** 1 hora
   - **Descripción:** Verificar que la instalación de las dependencias se complete sin errores revisando los mensajes en la terminal. Ejecutar node -v y npm -v para asegurar que Node.js y NPM están instalados correctamente.

Nro: HU-002	**Título:** Configuración de Webpack para el Empaquetado del Proyecto	**Prioridad:** (Alta) **Estimación:** (4 horas)

Historia de usuario:

Como **desarrollador** quiero **configurar Webpack para el empaquetado del proyecto**, para **asegurarme de que los archivos se compilen y se empaqueten correctamente para la distribución**.

Sabré que he terminado cuando se cumplan todos los criterios de aceptación:

- **Configuración básica de Webpack**: Dada una configuración básica de Webpack, cuando se ejecute npm run build, entonces el proyecto debe compilarse sin errores y generarse en la carpeta de distribución. **[ESCENARIO IDEAL]**

- **Integración de loaders y plugins:** Dado que se han añadido loaders y plugins necesarios, cuando se ejecute npm run build, entonces los archivos CSS, JS y otros recursos deben procesarse correctamente. **[ESCENARIO IDEAL]**

Tareas de implementación:

1. **Crear el archivo de configuración webpack.config.js:**
   - **Estimación:** 1 hora
   - **Descripción:** Crear el archivo webpack.config.js en la raíz del proyecto y definir la configuración básica incluyendo los entry y output.
2. **Configurar los entry points y output en Webpack:**
   - **Estimación:** 1 hora
   - **Descripción:** Definir el entry point de la aplicación (por ejemplo, src/index.js) y configurar la salida (output) para que los archivos compilados se generen en la carpeta dist.
3. **Añadir loaders para CSS y JavaScript:**
   - **Estimación:** 1 hora
   - **Descripción:** Configurar Webpack para utilizar babel-loader para archivos JavaScript y css-loader junto con style-loader para archivos CSS.
4. **Configurar plugins como HtmlWebpackPlugin para la generación del HTML final:**
   - **Estimación:** 1 hora
   - **Descripción:** Añadir e integrar el plugin HtmlWebpackPlugin para generar el archivo HTML final y enlazar automáticamente el archivo JavaScript empaquetado.
5. **Ejecutar npm run build y verificar que los archivos se empaqueten correctamente:**
   - **Estimación:** 1 hora
   - **Descripción:** Ejecutar el comando npm run build para iniciar el proceso de empaquetado y verificar que todos los archivos se generen correctamente en la carpeta de distribución sin errores.

Nro: HU-003	**Título:** Estructuración del Proyecto por Capas o Módulos	**Prioridad:** (Media) **Estimación:** (6 horas)
Historia de usuario: Como **desarrollador** quiero **estructurar el proyecto por capas o módulos**, para **mantener el código organizado y facilitar el mantenimiento y la escalabilidad del proyecto**.		
Sabré que he terminado cuando se cumplan todos los criterios de aceptación:		

- **Creación de una estructura modular:** Dada la necesidad de una estructura modular, cuando se revisen los archivos del proyecto, entonces cada módulo debe estar organizado en su propio directorio con responsabilidades claras. [ESCENARIO IDEAL]

- **Implementación de patrones de diseño adecuados:** Dado que se implementen patrones de diseño, cuando se revisen los módulos, entonces deben seguir principios de separación de preocupaciones y reutilización de código. [ESCENARIO IDEAL]

Tareas de implementación:

1. **Dividir el proyecto en directorios como src, tests, config, etc.**
   - **Estimación:** 1 hora
   - **Descripción:** Crear la estructura de directorios principal del proyecto y mover los archivos correspondientes a cada uno.
2. **Crear módulos específicos para funcionalidades como conversión de texto (braille.js, brailleMap.js), manejo de la interfaz (index.js), y estilos (style.css).**
   - **Estimación:** 1.5 horas
   - **Descripción:** Organizar los archivos en subdirectorios bajo src (por ejemplo, src/components, src/data, src/handlers, src/utils) y asegurarse de que cada módulo tenga una responsabilidad clara.
3. **Implementar patrones de diseño como MVC (Modelo-Vista-Controlador) o similar para separar las responsabilidades.**
   - **Estimación:** 2 horas
   - **Descripción:** Revisar y refactorizar el código existente para asegurarse de que sigue un patrón de diseño adecuado, separando la lógica de la aplicación en modelos, vistas y controladores.
4. **Documentar la estructura del proyecto para facilitar la comprensión y colaboración.**
   - **Estimación:** 1.5 horas
   - **Descripción:** Crear documentación que explique la estructura del proyecto, incluyendo la finalidad de cada directorio y archivo, así como las responsabilidades de cada módulo.

Nro: HU-004	**Título:** Configuración de Pruebas Unitarias con Jest	**Prioridad:** (Alta) **Estimación:** (10 horas)
Historia de usuario:		

Como **tester** quiero **configurar pruebas unitarias con Jest**, para **asegurarme de que las funciones clave del proyecto se prueben automáticamente y se mantenga la calidad del código**.

Sabré que he terminado cuando se cumplan todos los criterios de aceptación:

- **Configuración inicial de Jest.**
  Dada una configuración inicial de Jest, cuando se ejecute npm run test, entonces las pruebas unitarias deben ejecutarse correctamente y mostrar los resultados. [ESCENARIO IDEAL]
- **Implementación de pruebas para funciones clave.**
  Dado que se implementen pruebas para las funciones de conversión, cuando se ejecuten las pruebas, entonces deben validar correctamente el comportamiento de las funciones toBraille y toText. [ESCENARIO IDEAL]

Tareas de implementación:

1. **Instalar Jest como una dependencia de desarrollo en el proyecto.**
   - **Estimación:** 1 hora
   - **Descripción:** Ejecutar npm install --save-dev jest para agregar Jest como una dependencia de desarrollo en el proyecto.
2. **Crear el archivo de configuración jest.config.js para definir las opciones de Jest.**
   - **Estimación:** 2 horas
   - **Descripción:** Configurar Jest según las necesidades del proyecto, incluyendo la especificación de los directorios de pruebas y las opciones de cobertura.
3. **Escribir pruebas unitarias para las funciones en braille.js.**
   - **Estimación:** 4 horas
   - **Descripción:** Desarrollar pruebas unitarias que validen las funciones toBraille y toText, asegurando que se comporten como se espera con diferentes entradas.
4. **Escribir pruebas unitarias para las funciones en translator.js.**
   - **Estimación:** 1.5 horas
   - **Descripción:** Desarrollar pruebas unitarias que validen las funciones de conversión de texto en translator.js.
5. **Ejecutar las pruebas y verificar que todas pasen correctamente.**
   - **Estimación:** 1.5 horas
   - **Descripción:** Ejecutar npm run test y revisar los resultados para asegurarse de que todas las pruebas pasen y los resultados se muestren correctamente.

Nro: HU-005	**Título:** Transcripción de Español a Braille	**Prioridad:** (Alta) **Estimación:** (10 horas)

Historia de usuario:

Como **usuario** quiero **transcribir texto de español a Braille incluyendo números, abecedario, vocales acentuadas y signos básicos**, para **poder comunicarme con personas con discapacidad visual**.

Sabré que he terminado cuando se cumplan todos los criterios de aceptación:

- **Conversión exitosa de texto español a Braille:** Dada una cadena de texto en español, cuando se presione el botón "Traducir", entonces el texto debe ser convertido correctamente a Braille. [ESCENARIO IDEAL]
- **Manejo de caracteres no válidos en la conversión a Braille:** Dada una cadena de texto con caracteres no válidos, cuando se intente realizar la conversión, entonces debe mostrarse un mensaje de error indicando los caracteres inválidos. [ESCENARIO EXCEPCIONAL]

Tareas de implementación:

1. **Diseñar el algoritmo de conversión:**
   - **Estimación:** 2 horas
   - **Descripción:** Analizar y definir la lógica necesaria para mapear caracteres de español a Braille según el estándar.
2. **Implementar el algoritmo de conversión:**
   - **Estimación:** 3 horas
   - **Descripción:** Codificar la función toBraille en braille.js y asegurarse de que maneje correctamente todos los caracteres especificados.
3. **Actualizar el archivo de mapeo:**
   - **Estimación:** 1 hora
   - **Descripción:** Asegurarse de que brailleMap.js contenga todos los caracteres necesarios y realizar pruebas unitarias para verificar su precisión.
4. **Validar entradas de texto:**
   - **Estimación:** 2 horas
   - **Descripción:** Incluir validaciones en eventHandlers.js para verificar que el texto en español contiene solo caracteres válidos antes de la conversión.
5. **Integrar la funcionalidad en la interfaz:**

- **Estimación:** 2 horas
- **Descripción:** Conectar la funcionalidad de conversión con la interfaz gráfica (index.html y eventHandlers.js), incluyendo la gestión de eventos del botón "Traducir".

Nro: HU-006	Título: Transcripción de Braille a español	**Prioridad:** Alta **Estimación:** 10 horas

Historia de usuario:

Como **usuario** quiero **transcribir texto de Braille a español**, para **poder entender los documentos escritos en Braille**.

Sabré que he terminado cuando se cumplan todos los criterios de aceptación a continuación:

- **Conversión exitosa de Braille a texto español:** Dada una cadena de texto en Braille, cuando se presione el botón "Traducir", entonces el texto debe ser convertido correctamente a español. [ESCENARIO IDEAL]
- **Manejo de caracteres no válidos en la conversión a español:** Dada una cadena de texto en Braille con caracteres no válidos, cuando se intente realizar la conversión, entonces debe mostrarse un mensaje de error indicando los caracteres inválidos. [ESCENARIO EXCEPCIONAL]

Tareas de implementación:

1. **Diseñar el algoritmo de conversión:**
   - **Estimación:** 1 horas
   - **Descripción:** Analizar y definir la lógica necesaria para mapear caracteres de Braille a español.
2. **Implementar el algoritmo de conversión:**
   - **Estimación:** 2 horas
   - **Descripción:** Codificar la función toText en braille.js y asegurarse de que maneje correctamente todos los caracteres especificados.
3. **Actualizar el archivo de mapeo:**
   - **Estimación:** 1 hora

- **Descripción:** Verificar que brailleMap.js contenga todas las combinaciones de caracteres necesarios y realizar pruebas unitarias para asegurar su precisión.

4. **Validar entradas de texto:**
   - **Estimación:** 2 horas
   - **Descripción:** Incluir validaciones en eventHandlers.js para verificar que el texto en Braille contiene solo caracteres válidos antes de la conversión.
5. **Integrar la funcionalidad en la interfaz:**
   - **Estimación:** 2 horas
   - **Descripción:** Conectar la funcionalidad de conversión con la interfaz gráfica (index.html y eventHandlers.js), incluyendo la gestión de eventos del botón "Traducir".
6. **Integrar la funcionalidad del teclado Virtual:**
   - **Estimación:** 2 horas
   - **Descripción:** Conectar la funcionalidad para que se puede ingresar la simbología braille a través del teclado virtual en el index.html y el componente VirtualKeyboard.js

Nro: HU-007	Título: Generación de Señalética Braille	**Prioridad:** Media **Estimación:** 8 horas

Historia de usuario:

Como **usuario** quiero **generar señaléticas Braille a partir de textos en español**, para **crear etiquetas accesibles**.

Sabré que he terminado cuando se cumplan todos los criterios de aceptación a continuación:

- **Generación exitosa de señalética Braille:** Dado un texto en español, cuando se seleccione la opción "Descargar Señalética", entonces se debe generar y descargar una imagen con el texto en Braille. [ESCENARIO IDEAL]
- **Manejo de caracteres no válidos en la señalética**: Dado un texto vacío o con caracteres no válidos, cuando se intente generar la señalética, entonces debe mostrarse un mensaje de error. [ESCENARIO EXCEPCIONAL]

Tareas de implementación:

1. **Implementar funcionalidad de generación de señalética:**
   - **Estimación:** 3 horas

- **Descripción:** Codificar la función createSignageElement en utils.js para generar el elemento HTML correspondiente.

2. **Implementar descarga de imagen:**
   - **Estimación:** 2 horas
   - **Descripción:** Utilizar html2canvas para convertir el elemento HTML a imagen y permitir su descarga.
3. **Validar entradas de texto:**
   - **Estimación:** 2 horas
   - **Descripción:** Incluir validaciones en eventHandlers.js para asegurar que el texto en español contiene solo caracteres válidos antes de la generación de señalética.
4. **Integrar la funcionalidad en la interfaz:**
   - **Estimación:** 1 hora
   - **Descripción:** Conectar la funcionalidad de generación de señalética con la interfaz gráfica (index.html y eventHandlers.js), incluyendo la gestión de eventos del botón "Descargar Señalética".

Nro: HU-008	Título: Generación de Señalética Braille	**Prioridad:** Media **Estimación:** 8 horas

Historia de usuario:

Como **usuario** quiero **generar impresión en espejo de textos Braille para escritura manual**, para **poder perforar papel con una regleta y un punzón o utilizar una impresora 3D**

Sabré que he terminado cuando se cumplan todos los criterios de aceptación a continuación:

- **Generación exitosa de impresión en espejo:** Dado un texto en español, cuando se seleccione la opción "Descargar Formato Espejo", entonces se debe generar y descargar un PDF con el texto en Braille en formato espejo. [ESCENARIO IDEAL]
- **Manejo de caracteres no válidos en la impresión en espejo:** Dado un texto vacío o con caracteres no válidos, cuando se intente generar la impresión en espejo, entonces debe mostrarse un mensaje de error. [ESCENARIO EXCEPCIONAL]

Tareas de implementación:

1. **Implementar funcionalidad de generación de impresión en espejo:**

- **Estimación:** 3 horas
- **Descripción:** Codificar la función createMirrorElement en utils.js para generar el elemento HTML correspondiente.

2. **Implementar descarga de PDF:**
   - **Estimación:** 2 horas
   - **Descripción:** Utilizar jsPDF para convertir el elemento HTML a PDF y permitir su descarga.
3. **Validar entradas de texto:**
   - **Estimación:** 2 horas
   - **Descripción:** Incluir validaciones en eventHandlers.js para asegurar que el texto en español contiene solo caracteres válidos antes de la generación de impresión en espejo.
4. **Integrar la funcionalidad en la interfaz:**
   - **Estimación:** 1 hora
   - **Descripción:** Conectar la funcionalidad de generación de impresión en espejo con la interfaz gráfica (index.html y eventHandlers.js), incluyendo la gestión de eventos del botón "Descargar Formato Espejo".

REFERENCIAS

[1] M. Pástor, E. Arias, M. Carrion, J. Poaquiza y F. Bailón, «BrailleTech,» [En línea]. Available: https://github.com/MaxCar31/BrailleTech.