ISWD633 CONSTRUCCIÓN Y EVOLUCIÓN DE SOFTWARE

TRADUCTOR VERSIÓN: 1.0

FECHA DE PUBLICACIÓN: 6/6/2024

AUTOR: BrailleTech

TEMA: FLUJO DE TRABAJO

WORK-FLOW										
# ITERACIÓN	**FASE**				**RESPONSABLE**	**ARTEFACTOS PRODUCIDOS**	**TAREAS**	**PRIORIDAD**	**ESTIMACIÓN**	**ESTADO**
Iteración 1	Planificación				Max Carrion - Analista	Historia de usuario 1, 2 y 4	1. Identificar y documentar las necesidades y requerimientos del usuario. 2. Elaborar las historias de usuario con criterios de aceptación claros. 3. Realizar reuniones de revisión con el equipo de desarrollo para asegurar el entendimiento de las historias de usuario. 4. Revisar y ajustar las historias de usuario basándose en la retroalimentación del equipo. 5. Aprobar las historias de usuario finales para su implementación.	Alta	4 horas (1 por cada HU)	COMPLETADO
		Diseño			Max Carrion - Analista	Configuración de entorno de desarrollo Archivo package.json (dependencias del proyecto)	1. Descargar e instalar Node.js y NPM en el entorno de desarrollo. 2. Crear o actualizar el archivo package.json con las dependencias necesarias. 3. Ejecutar npm install para instalar todas las dependencias del proyecto. 4. Verificar que todas las dependencias se instalen correctamente y que no haya errores	Alta	4 horas	COMPLETADO
			Implementación		Jonathan Poaquiza, Eddy Arias - Programadores	Configuración de Webpack para compilación y empaquetado del proyecto.	1. Configuración básica de Webpack. 2. Integración de loaders y plugins. 3. Crear el archivo de configuración webpack.config.js. 4. Configurar los entry points y output en Webpack. 5. Añadir loaders para CSS y JavaScrip. 6. Configurar plugins como HtmlWebpackPlugin para la generación del HTML final. 7. Ejecutar npm run build y verificar que los archivos se empaqueten correctamente.	Alta	4 horas	COMPLETADO

				Pruebas	Milton Pástor, Fredviner Bailon - Testers	Configuración de Jest para pruebas unitarias	1. Instalar Jest como una dependencia de desarrollo en el proyecto. 2. Crear el archivo de configuración jest.config.js para definir las opciones de Jest. 3. Escribir pruebas unitarias para las funciones en braille.js. 4. Escribir pruebas unitarias para las funciones en translator.js. 5. Ejecutar las pruebas y verificar que todas pasen correctamente.	Media	10 horas	COMPLETADO
Iteración 2	Planificación				Max Carrion - Analista	Historia de usuario 3, 5 y 6	1. Identificar y documentar las necesidades y requerimientos del usuario. 2. Elaborar las historias de usuario con criterios de aceptación claros. 3. Realizar reuniones de revisión con el equipo de desarrollo para asegurar el entendimiento de las historias de usuario. 4. Revisar y ajustar las historias de usuario basándose en la retroalimentación del equipo. 5. Aprobar las historias de usuario finales para su implementación.	Alta	3 horas (1 hora por HU)	COMPLETADO
		Diseño			Max Carrion - Analista	Estructura del proyecto por Capas o modulos	1. Dividir el proyecto en directorios como src, tests, config, etc. 2. Crear módulos específicos para funcionalidades como conversión de texto (braille.js, brailleMap.js), manejo de la interfaz (index.js), y estilos (style.css). 3. Implementar patrones de diseño como MVC (Modelo-Vista-Controlador) o similar para separar las responsabilidades. 4. Documentar la estructura del proyecto para facilitar la comprensión y colaboración. 5. Diseñar el algoritmo de conversión.	Media	6 horas	COMPLETADO
			Implementación		Jonathan Poaquiza, Eddy Arias - Programadores	Codigo fuente de los modulos braille.js (Funcion toBraille y funciones auxiliares) Modulo brailleMap para realizar el mapeo. Modulo translate que implenta los metodos del braille.js	1. Codificar la función toBraille en braille.js y asegurarse de que maneje correctamente todos los caracteres especificados. 2. Incluir validaciones en eventHandlers.js para verificar que el texto en español contiene solo caracteres válidos antes de la conversión. 3. Conectar la funcionalidad de conversión con la interfaz gráfica (index.html y eventHandlers.js), incluyendo la gestión de eventos del botón "Traducir".	Alta	15 horas	INCOMPLETO (FALTA IMPLEMENTAR MÁS CARACTERES ESPECIALES)
				Pruebas	Milton Pástor, Fredviner Bailon - Testers	Casos de prueba: CP001, CP002, CP03, CP008, CP009, CP010, CP011, CP012, CP013, CP018 Pruebas unitarias de braille.test.js: PU001, PU002, PU003, PU004, PU005, PU006, PU007, PU008, PU009, PU010, PU011, PU012, PU013, PU014, PU015, PU016, PU017, PU018, PU019, PU020, PU021, PU022	1. Verificar la correcta codificación de la función toBraille en braille.js, asegurándose de que maneje correctamente todos los caracteres especificados. 2. Revisar las validaciones en eventHandlers.js para confirmar que el texto en español contiene solo caracteres válidos antes de la conversión. 3. Validar la conexión de la funcionalidad de conversión a braille con la interfaz gráfica (index.html y eventHandlers.js), incluyendo la gestión de eventos del botón "Traducir".. 4. Estructurar las pruebas unitarias. 5. Implementar Casos de pruebas.	Alta	8 horas	INCOMPLETO (FALTA PASAR TODAS LAS PRUEBAS UNITARIAS)

Iteración 3	Planificación				Max Carrion - Analista	Historia de usuario 7 y 8	1. Identificar y documentar las necesidades y requerimientos del usuario en esta iteración se determino que es necesario la generación de señaletica Braille y generación de señaletica con espejo. 2. Elaborar las historias de usuario con criterios de aceptación claros. 3. Realizar reuniones de revisión con el equipo de desarrollo para asegurar el entendimiento de las historias de usuario. 4. Revisar y ajustar las historias de usuario basándose en la retroalimentación del equipo. 5. Aprobar las historias de usuario finales para su implementación.	Alta	2 horas	COMPLETADO
		Diseño			Max Carrion - Analista	Arquitectura del sistema Dependencias adicinales: html2canvas y jsPDF Flujo de Datos Definición de patrones de diseño	1. Diseño de una arquitectura por capas, en la que cada capa tiene responsabilidades específicas. 2. Análisis de herramientas utilizadas para generar imágenes y PDFs a partir de elementos HTML. 3. Creación del flujo de datos. 4. Definición del encapsulamiento y la separación de responsabilidades.	Alta	4 horas	COMPLETADO
			Implementación		Jonathan Poaquiza, Eddy Arias - Programadores	Modulo utils.js con funciones utilitarias, como la generación de señalética y la creación de elementos en espejo.	1. Codificar la función createSignageElement en utils.js para generar el elemento HTML correspondiente. 2. Utilizar html2canvas para convertir el elemento HTML a imagen y permitir su descarga. 3. Incluir validaciones en eventHandlers.js para asegurar que el texto en español contiene solo caracteres válidos antes de la generación de señalética. 4. 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". 5. Codificar la función createMirrorElement en utils.js para generar el elemento HTML correspondiente. 6. Utilizar jsPDF para convertir el elemento HTML a PDF y permitir su descarga. 7. Incluir validaciones en eventHandlers.js para asegurar que el texto en español contiene solo caracteres válidos antes de la generación de impresión en espejo. 8. 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".	Alta	8 horas	COMPLETADO
				Pruebas	Milton Pástor, Fredviner Bailon - Testers	Casos de prueba: CP004, CP005, CP006 y CP007	1. Verificar la correcta codificación de la función createSignageElement en utils.js para asegurar que genera el elemento HTML adecuado. 2. Probar la utilización de html2canvas para convertir el elemento HTML en una imagen y validar la funcionalidad de descarga. 3. Revisar las validaciones en eventHandlers.js para confirmar que el texto en español contiene solo caracteres válidos antes de la generación de señalética. 4. Validar la conexión de la funcionalidad de generación de señalética con la interfaz gráfica (index.html y eventHandlers.js), incluyendo la gestión de eventos del botón "Descargar Señalética". 5. Comprobar la correcta codificación de la función createMirrorElement en utils.js para asegurar que genera el elemento HTML adecuado. 6. Probar la utilización de jsPDF para convertir el elemento HTML en un PDF y validar la funcionalidad de descarga. 7. Revisar las validaciones en eventHandlers.js para confirmar que el texto en español contiene solo caracteres válidos antes de la generación de impresión en espejo. 8. Validar la conexión de 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".	Media	6 horas	COMPLETADO