



Universidad Politécnica de Madrid
**Escuela Técnica Superior de
Ingenieros Informáticos**



Grado en Ingeniería Informática

Trabajo Fin de Grado

**Desarrollo del Módulo de Control de un
Entorno Virtual de Ensamblador Web y
Diseño de la Interfaz Accesible**

Autora: Nicole García Langa

Tutores: José Luis Fuertes Castro y Aurora Pérez Pérez

Madrid, enero de 2026

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Título: Desarrollo del Módulo de Control de un Entorno Virtual de
Ensamblador Web y Diseño de la Interfaz Accesible

Enero 2026

Autor: Nicole García Langa

Tutores:

José Luis Fuertes Castro

Aurora Pérez Pérez

Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de
Software

ETSI Informáticos

Universidad Politécnica de Madrid

Resumen

Este Trabajo de Fin de Grado (TFG) presenta el diseño y desarrollo de la interfaz web para un nuevo simulador de lenguaje ensamblador, basado en la herramienta ENS2001 y orientado a la asignatura "Traductores de Lenguajes" de la Universidad Politécnica de Madrid. La arquitectura de este simulador ha sido dividida en tres módulos fundamentales: el módulo de ensamblado, el módulo de ejecución y la interfaz de usuario. Estos módulos han sido implementados de forma paralela por un equipo de tres alumnos. El objeto de este TFG es el desarrollo y la integración de la interfaz de usuario.

La solución desarrollada consiste en una aplicación web que elimina las barreras de instalación y garantiza la compatibilidad multiplataforma. Ofrece control completo sobre la ejecución de la simulación y sobre el estado de la máquina simulada. Entre las mejoras introducidas respecto a su predecesor destacan un editor de código integrado, una gestión de depuración más intuitiva y opciones de personalización de tema e idioma. El desarrollo se ha llevado a cabo bajo un enfoque de Diseño Centrado en el Usuario (DCU), validado mediante tests de usabilidad, y con un compromiso estricto con la accesibilidad web siguiendo las pautas WCAG 2.2 (Nivel AA).

Desde el punto de vista técnico, tras un análisis previo de las posibles herramientas a utilizar y sus consideraciones de accesibilidad, la interfaz ha sido implementada con React. A lo largo del proyecto se ha hecho uso de sus herramientas avanzadas para solventar eficientemente los desafíos técnicos de los componentes más complejos. En la fase de pruebas, se han combinado pruebas automatizadas utilizando React Testing Library con pruebas manuales de sistema y de accesibilidad.

El resultado final es una herramienta educativa moderna y accesible que, gracias a la integración de los tres módulos de desarrollo, ofrece una experiencia de simulación completa y adaptada a las necesidades actuales de los alumnos de "Traductores de Lenguajes", aunque su uso puede resultar útil para cualquier persona interesada en un simulador simple y didáctico de ensamblador.

Abstract

This Final Degree Project (TFG) presents the design and development of the web interface for a new assembly language simulator, based on the ENS2001 tool and aimed at the “Language Translators” course at the Universidad Politécnica de Madrid. The architecture of this simulator has been divided into three fundamental modules: the assembly module, the execution module, and the user interface. These modules have been implemented concurrently by a team of three students. Within this collaborative framework, the objective of this TFG is the development and integration of the user interface.

The solution developed consists of a web application that eliminates installation barriers and guarantees multi-platform compatibility. It offers complete control over the execution of the simulation and the status of the simulated machine. Among the improvements introduced in relation to its predecessor are an integrated code editor, more intuitive debugging management, and theme and language customization options. The development was carried out using a User-Centered Design (UCD) approach, validated through usability tests, and with a strict commitment to web accessibility following WCAG 2.2 (Level AA) guidelines.

From a technical standpoint, after a preliminary analysis of the possible tools to be used and their accessibility considerations, the interface was implemented with React. Throughout the project, its advanced tools were used to efficiently solve the technical challenges of the most complex components. In the testing phase, automated tests with React Testing Library were combined with manual system and accessibility tests.

The result is a modern and accessible educational tool that, thanks to the integration of the three development modules, offers a complete simulation experience adapted to the current needs of “Language Translators” students, although its use may be useful for anyone interested in a simple and educational assembler simulator.

Tabla de contenidos

1	Introducción.....	1
2	Estado de la cuestión.....	4
2.1	Conceptos relacionados con el ensamblador.....	4
2.1.1	El lenguaje ensamblador y simuladores de ensamblador	4
2.1.2	El ensamblador ENS2001	4
2.2	Conceptos fundamentales del desarrollo web	6
2.2.1	Aplicaciones de página única (<i>Single Page Applications</i>)	6
2.2.2	Accesibilidad web.....	7
2.3	Lenguajes web.....	9
2.3.1	Lenguaje de marcado (HTML).....	9
2.3.2	Hojas de estilo en cascada (CSS).....	9
2.3.3	Lenguaje de programación JavaScript.....	10
2.4	Bibliotecas y <i>frameworks</i> de JavaScript	11
2.4.1	Introducción a las bibliotecas y <i>frameworks</i> de JavaScript	11
2.4.2	Consideraciones de accesibilidad en el desarrollo con <i>frameworks</i> 12	
2.4.3	Bibliotecas y <i>frameworks</i> principales	13
2.4.3.1	React.js.....	13
2.4.3.2	Vue.js	14
2.4.3.3	Svelte.....	15
2.4.3.4	Angular.....	16
2.5	Herramientas de desarrollo	17
2.5.1	Control de versiones: Git y GitLab.....	17
2.5.2	Entorno de desarrollo integrado: Visual Studio Code	18
2.5.3	Prototipado y diseño de interfaces: Figma	19
3	Planteamiento del problema	20
3.1	Introducción.....	20
3.1.1	Propósito	20
3.1.2	Ámbito del sistema	20
3.1.3	Definiciones, acrónimos y abreviaturas.....	20
3.1.4	Referencias	21
3.1.5	Visión general de la ERS.....	21
3.2	Descripción general.....	21
3.2.1	Perspectiva del producto	21
3.2.2	Funciones del producto	21
3.2.3	Características de los usuarios	22
3.2.4	Restricciones	22

3.2.5	Suposiciones y dependencias	22
3.2.6	Requisitos futuros	22
3.3	Requisitos específicos.....	22
3.3.1	Interfaces externas	22
3.3.1.1	Comunicación con el módulo de ensamblado.....	22
3.3.1.2	Comunicación con el módulo de ejecución.....	23
3.3.2	Funciones.....	24
3.3.2.1	Aspectos generales.....	24
3.3.2.2	Ventana de editor	24
3.3.2.3	Ventana de memoria.....	25
3.3.2.4	Ventana de registros	26
3.3.2.5	Ventana de desensamblado	26
3.3.2.6	Ventana de consola.....	26
3.3.2.7	Ventana de logs	27
3.3.2.8	Configuración	27
3.3.3	Requisitos no funcionales	28
4	Desarrollo	29
4.1	Diseño.....	29
4.1.1	Casos de uso	29
4.1.2	Diagrama de secuencia.....	34
4.1.3	Diseño de la interfaz	35
4.1.3.1	Diseño de baja fidelidad	35
4.1.3.2	Diseño de alta fidelidad.....	36
4.1.4	Test de usabilidad.....	38
4.1.4.1	Planteamiento.....	38
4.1.4.2	Resultados.....	42
4.1.4.3	Mejoras aplicadas	49
4.2	Implementación	49
4.2.1	Herramientas utilizadas.....	49
4.2.2	Organización del proyecto.....	50
4.2.3	Principales características de implementación	50
4.2.3.1	Manejo de la vista de las ventanas	50
4.2.3.2	Manejo de la ventana de memoria.....	51
4.2.3.3	Manejo de la ventana de desensamblado	53
4.2.3.4	Manejo de la ventana del editor	53
4.2.3.5	Consistencia del estado de la máquina	54
4.2.3.6	Configuración de la interfaz	56
4.2.3.7	Comunicación con otros módulos	56
4.2.3.8	Implementación de la interfaz bilingüe.....	58

4.2.4	Implementación de la accesibilidad.....	58
4.2.4.1	HTML semántico y etiquetas WAI-ARIA.....	58
4.2.4.2	Navegación por teclado	60
4.2.4.3	Otros mecanismos	60
4.3	Casos de prueba	61
4.3.1	Pruebas unitarias	61
4.3.2	Pruebas de sistema.....	62
4.3.3	Pruebas de accesibilidad.....	62
4.3.3.1	Pruebas automáticas	62
4.3.3.2	Pruebas manuales	63
5	Resultados y conclusiones	66
6	Análisis de Impacto	69
6.1	Impacto Personal.....	69
6.2	Impacto Empresarial.....	69
6.3	Impacto Social.....	70
6.4	Impacto Económico	70
6.5	Impacto Medioambiental	70
6.6	Relación con los Objetivos de Desarrollo Sostenible	70
6.7	Decisiones Basadas en el Impacto	70
7	Futuras líneas de trabajo	72
7.1	Siguientes pasos a seguir	72
7.2	Funcionalidades a añadir.....	72
8	Bibliografía	75
9	Anexos.....	83
9.1	Manual de usuario	83
9.1.1	Selección de tema e idioma.....	84
9.1.2	Menú de la aplicación	84
9.1.2.1	Submenú ‘Archivo’.....	84
9.1.2.2	Submenú ‘Simulación’.....	85
9.1.2.3	Submenú ‘Configuración’.....	86
9.1.2.4	Submenú ‘Vista’.....	87
9.1.2.5	Submenú ‘Ayuda’.....	88
9.1.3	Barra de botones	88
9.1.4	Ventanas de la aplicación	89
9.1.4.1	Ventana de instrucciones.....	89
9.1.4.2	Editor	89
9.1.4.3	Ventana de <i>logs</i>	90
9.1.4.4	Consola.....	91
9.1.4.5	Ventana de registros	92

9.1.4.6	Ventana de memoria.....	92
9.1.4.7	Menú de navegación de tablas	94
9.1.5	Navegación por teclado	94

1 Introducción

La idea de este proyecto surge a partir de la intención de mejora de la asignatura Traductores de Lenguajes, cursada en el Grado en Ingeniería Informática de la Escuela Técnica Superior de Ingenieros Informáticos (ETSIInf) de la Universidad Politécnica de Madrid. En esta asignatura, los alumnos amplían su conocimiento de su asignatura predecesora, Procesadores de Lenguajes, aprendiendo el diseño y funcionamiento interno de los traductores de lenguajes.

Para aplicar los conocimientos adquiridos, una parte esencial de la asignatura es la realización de una práctica en equipo, en la que se desarrolla un traductor de un lenguaje a partir de un procesador, implementando desde cero los módulos generador de código intermedio y generador de código objeto. Este traductor convierte un fragmento de código de un lenguaje fuente descrito en el enunciado de la práctica a su equivalente en un lenguaje ensamblador simple, el ENS2001.

El ENS2001 fue diseñado en 2001 a partir del estándar IEEE 694 para su uso en esta asignatura. Una aplicación de escritorio con su mismo nombre (ENS2001) fue desarrollada para poder ejecutar este lenguaje de ensamblador. A través de esta herramienta, los alumnos comprueban el funcionamiento de su traductor y depuran su código. El ENS2001 les permite recibir el resultado del ensamblado y los posibles errores del programa. Después, pueden observar el desensamblado, la memoria y los registros de la máquina, ejecutar en lote o instrucción a instrucción e interactuar con la entrada y la salida.

Este simulador manifiesta ciertos inconvenientes con los que tienen que lidiar los alumnos de cada promoción. A continuación, se presenta una lista de los problemas más importantes:

- Problemas con las actualizaciones de los sistemas operativos. ENS2001 es una aplicación de escritorio que está disponible en varias plataformas, por lo que es susceptible a problemas de incompatibilidad causados por las actualizaciones de los sistemas operativos. A lo largo de los años, el sistema ha sufrido este problema y ha requerido de mantenimiento para que los alumnos pudieran continuar utilizándolo correctamente.
- Limitaciones en las instrucciones del lenguaje. El uso de ensamblador debe ser lo más simple posible, al no ser el foco de la asignatura. Sin embargo, algunas características del actual ENS2001 dificultan su entendimiento. Por ejemplo, sólo dos registros (IX e IY) se pueden utilizar para realizar el modo de direccionamiento relativo a registro índice. Además, este salto admite un entero de 8 bits, cuyo valor máximo es 127. Este valor ha demostrado ser insuficiente para muchos saltos habituales en la práctica.
- Problemas de experiencia de usuario. Aunque la interfaz es funcional y accesible, podría beneficiarse de actualizaciones que mejoren su usabilidad. Actualmente, no permite la ejecución de un archivo varias veces, siendo necesario volver a cargar y ensamblar el programa cada vez que se quiera ejecutar. Por otro lado, el sistema es flexible en la vista, la colocación y el tamaño de cada ventana, pero no ofrece ninguna opción de personalización respecto a aspecto o idioma.

A partir de esta situación, se tomó la decisión de desarrollar una nueva versión del simulador ENS2001, que arreglase todos los problemas de su predecesor. Para facilitar la instalación y asegurar su compatibilidad con todos los sistemas operativos, el nuevo producto toma la forma de sistema web en lugar de

aplicación de escritorio, con una interfaz amigable y moderna. El conjunto de instrucciones del ensamblador aceptado por este sistema también está actualizado, con nuevas instrucciones, permitiendo ahora el direccionamiento por registro índice con el puntero de pila y ampliando el tamaño de este direccionamiento a 16 bits.

Aunque el proyecto esté dirigido a los alumnos de Traductores de Lenguajes, la nueva solución web puede ser beneficiosa para cualquier usuario que busque un simulador simple con el que aprender ensamblador. Actualmente, el número de simuladores de ensamblador en línea disponibles es reducido y suelen ofrecer menos funcionalidades que las abordadas en este sistema.

Dada la gran dimensión del proyecto, se ha dividido en varios Trabajos de Fin de Grado (TFG), de forma que el trabajo a realizar en este TFG se enfocará específicamente en el diseño y el desarrollo de la interfaz web, y será posteriormente integrado con el resto de TFG, encargados del ensamblado y la ejecución del código.

La interfaz de este sistema tiene como objetivo ofrecer una experiencia completa centrada en el usuario. Contiene todas las funcionalidades y la flexibilidad de pantallas del ENS2001 con mejoras para ensalzar la usabilidad. Las modificaciones principales respecto a su predecesor son las siguientes:

- Inclusión de un editor. Se proporciona un editor integrado, eliminando la necesidad de un programa externo para la modificación del código.
- Botones de depuración mejorados. La nueva barra de botones contiene botones para ejecución en lote y ejecución paso a paso; previamente se debía acceder a un cambio en la configuración para alternar entre ambos modos de ejecución. También se incluye un nuevo botón de reinicio que permite volver al estado inicial de la máquina y ejecutar más de una vez sin necesidad de reensamblado.
- Personalización del tema y el idioma. La interfaz ofrece soporte bilingüe para los idiomas español e inglés (ampliable a más idiomas) y elección entre tema claro y oscuro.

El sistema debe poder ser utilizado por el mayor número posible de personas, independientemente de sus capacidades o limitaciones. Por ello, uno de los principales focos de este TFG es la accesibilidad web. Durante el desarrollo del proyecto se seguirán las pautas WCAG 2.2 como referencia, garantizando el cumplimiento de un nivel de conformidad AA. El seguimiento de estas pautas se verificará a través de pruebas manuales que permitan detectar y corregir los errores de accesibilidad del programa.

Para la realización de este proyecto de forma estructurada y organizada, se han seguido las bases de ingeniería del software y se ha dividido el proyecto en las siguientes etapas:

1. Investigación tecnológica. Esta fase inicial supone el estudio de las herramientas más aptas para este sistema, con foco en la accesibilidad.
2. Diseño. Se presenta la especificación de requisitos software y los casos de uso del sistema. Después, se realiza un prototipo de alta fidelidad del sistema. El prototipo se prueba a través de test de usabilidad con usuarios. Por último, se analizan los resultados de las pruebas de usabilidad y se rediseñan las pantallas del sistema.
3. Implementación. Se comparten los detalles más importantes de la implementación del sistema a partir de su diseño.

4. Pruebas. Se detalla un conjunto de pruebas unitarias, de integración y de usabilidad y se analizan los resultados.
5. Conclusiones y futuras líneas de trabajo. Para finalizar, se comentan las principales conclusiones del trabajo realizado y las futuras líneas de trabajo para mejorar el proyecto en años venideros.

A nivel personal, este proyecto representa una oportunidad valiosa para profundizar en el desarrollo accesible de software y aplicar de forma práctica los conocimientos adquiridos durante la titulación. La creación de software de calidad siempre ha sido uno de mis principales objetivos como ingeniera informática, y este trabajo supone un paso importante en esa dirección, al contribuir a resolver un problema real y mejorar la experiencia de los estudiantes de Traductores de Lenguajes y de otros posibles usuarios del sistema.

2 Estado de la cuestión

2.1 Conceptos relacionados con el ensamblador

2.1.1 El lenguaje ensamblador y simuladores de ensamblador

El lenguaje ensamblador es un lenguaje de programación a bajo nivel muy cercano al lenguaje máquina [1]. Proporciona una manera de enviar instrucciones al procesador a través de una sintaxis sencilla basada en mnemónicos y abreviaturas. De esta manera, ahorra al desarrollador la tarea de recordar los códigos numéricos de las operaciones y el cálculo de las direcciones de memoria. Para ejecutarlo, se utiliza un programa ensamblador, que convierte el código ensamblador en lenguaje máquina que puede ejecutar directamente el procesador.

El lenguaje ensamblador es dependiente de la arquitectura de la máquina; cada procesador tiene un conjunto de instrucciones y una sintaxis diferente. Sin embargo, todas las instrucciones en ensamblador suelen seguir una estructura similar: un mnemónico, que indica el código de operación, seguido de unos operandos, parámetros o datos que serán utilizados para llevar a cabo la instrucción. Un ejemplo de instrucción en el ensamblador x86 [2], creado por Intel, sería: `mov rax, 4` (asigna el valor 4 al registro rax). Además de las instrucciones, los lenguajes de ensamblador suelen contener sentencias de datos, utilizadas para manejar datos y variables en la memoria de la máquina, y directivas al ensamblador, como las etiquetas, que se resuelven durante el ensamblado del programa.

Este lenguaje tuvo su primer uso en la década de 1950 y fue evolucionando en los siguientes años. Posteriormente, la aparición de lenguajes como Fortran en 1957 comenzó una tendencia hacia lenguajes de alto nivel y el uso de ensamblador se vio progresivamente relegado [3]. Sin embargo, en la actualidad sigue habiendo varios sectores donde es esencial el uso del ensamblador: en el desarrollo de sistemas operativos, en los sistemas embebidos o en la programación de código crítico [4].

Para comprender de manera visual cómo afectan las instrucciones al estado de la máquina, se puede utilizar un simulador de ensamblador. Un simulador reproduce la ejecución de un código de ensamblador y muestra el estado que tendría la máquina en cada momento, incluyendo el contenido de los registros, de la memoria y de los módulos de entrada y salida. Una ventaja de un simulador es su independencia del hardware físico, lo que permite observar el comportamiento de cualquier programa sin poseer el dispositivo real.

2.1.2 El ensamblador ENS2001

ENS2001 [5] es una aplicación que integra un ensamblador de un subconjunto de instrucciones del estándar IEEE 694 y un simulador de este lenguaje, junto a una máquina virtual. La herramienta tiene fines didácticos, siendo su uso principal la ayuda a la depuración de la práctica a los alumnos de Compiladores, que es la asignatura predecesora a la asignatura Traductores de Lenguajes del actual Grado en Ingeniería Informática de la ETSIInf. Sin embargo, puede ser de interés para cualquier persona que busque un simulador sencillo con el que aprender las bases de ensamblador. El ENS2001 fue basado en dos herramientas anteriores, ASS y ENS96, así como en el ensamblador del procesador Z-80. La principal mejora respecto a sus predecesores fue la

inclusión de una nueva interfaz gráfica, que se puede observar en la Ilustración 1.

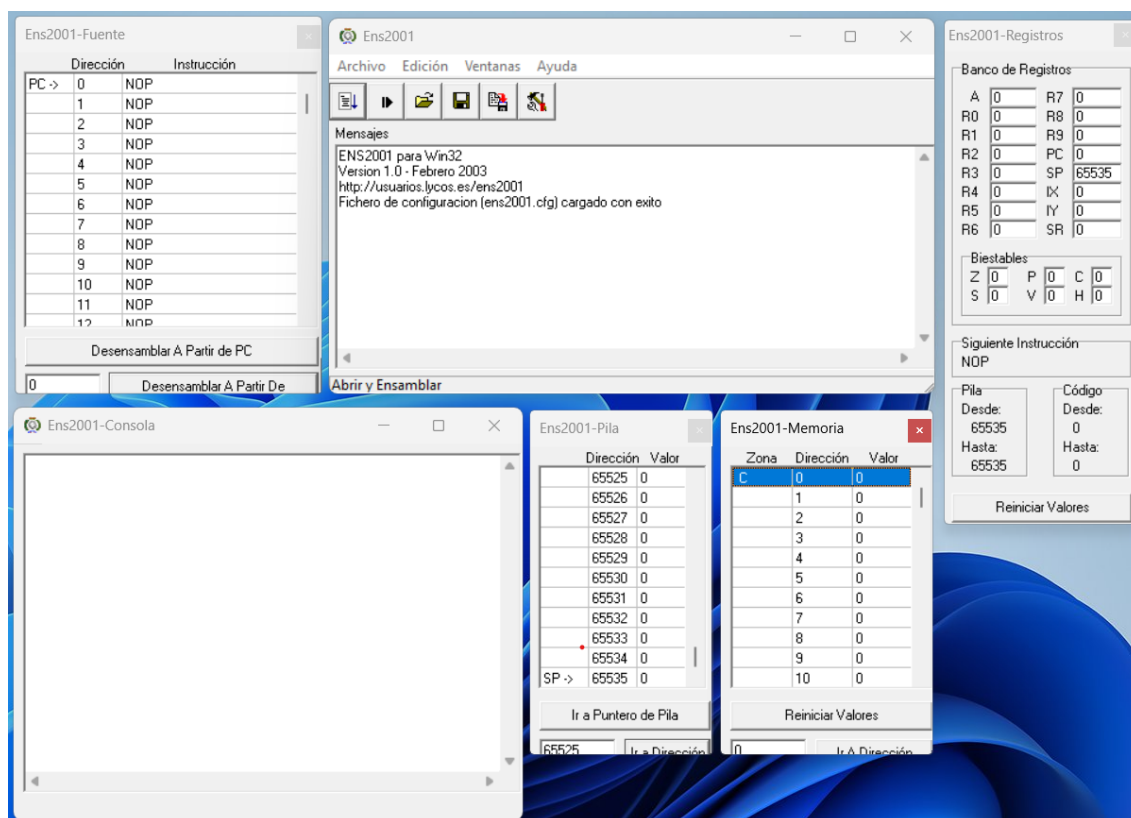


Ilustración 1. ENS 2001 en Windows

ENS2001 permite cargar y ensamblar un fichero de código fuente e informa en caso de error en el ensamblado. Tras este proceso, el usuario puede observar y modificar la memoria, los registros y los biestables, ver el desensamblado y comunicarse con el programa a través de la consola de entrada y salida. A continuación, se explican en más detalle las opciones de cada ventana:

- **Ventana del procesador.** Esta es la ventana principal. Permite controlar la simulación a través de un menú con las siguientes opciones: abrir y ensamblar un archivo, ejecutar el programa (en lote o instrucción a instrucción dependiendo de la configuración), cargar una imagen de memoria del disco, guardar la imagen de memoria actual y, por último, configurar el sistema. Todos los resultados de las operaciones se ven reflejados en la sección de mensajes.
- **Ventana de desensamblado.** Muestra el desensamblado del programa. Si el contenido de una sección de memoria no coincide con el de ninguna instrucción, se muestra en esa casilla el texto "***INSTRUCCIÓN NO IMPLEMENTADA**". También indica la posición del PC. Con los botones de *Desensamblar a partir de PC* y *Desensamblar a partir de Dirección*, se puede actualizar el desensamblado con el fin de descubrir si el programa ha modificado el código máquina. Además, se puede añadir un punto de ruptura en una dirección de memoria pulsando dos veces sobre su casilla. Este punto de ruptura hará que se interrumpa la ejecución en lote en esa dirección, solo si el modo de depuración está activado en la configuración.
- **Banco de registros.** Muestra el estado de todos los registros y biestables y permite su modificación. También cuenta con un botón para reiniciar sus valores. Tiene alguna información adicional como el desensamblado

de la siguiente instrucción y la dirección inicial y final de la zona de código y de la pila.

- **Ventana de memoria.** Muestra el contenido de la memoria y permite su modificación. Además de la columna de dirección y contenido, una columna adicional indica la región en la que se encuentra esa dirección, código o pila. Asimismo, cuenta con un botón de reiniciar su valor y de ir a una dirección.
- **Ventana de pila.** Igual que la ventana de la memoria, excepto que cuenta con un botón para ir a la dirección a la que apunta el puntero de pila.
- **Consola.** Muestra la entrada y salida del programa. En caso de pedir el programa una entrada, esta ventana muestra un pequeño mensaje en la parte inferior indicando al usuario qué tipo de entrada se debe introducir.
- **Ventana de configuración.** A través de este menú, accesible desde la ventana del procesador, se puede configurar el formato de los números, la dirección de la pila, la activación del modo de depuración, la ejecución en lote o paso a paso y varias opciones de comprobación de fallos. Se pueden ver con más detalle estas opciones en la Ilustración 2.

Ens2001-Configuración

Base de Representación Numérica

☐ Base 10 (decimal)

☒ Base 16 (hexadecimal)

Dirección de Crecimiento de la Pila

☒ Direcciones Ascendentes

☐ Direcciones Descendentes

Modo de Depuración

☒ Sí ☐ No

Ejecución Paso a Paso

☒ Sí ☐ No

Comprobar si PC invade el espacio de Pila

☐ Sí ☒ No

Comprobar si SP invade el espacio de Código

☐ Sí ☒ No

Inicializar los Registros al Ejecutar

☒ Sí ☐ No

Aceptar Cancelar

Ilustración 2. Ventana de configuración de ENS2001

Esta aplicación está disponible en cuatro versiones: para MSDOS, Windows, Linux y macOS Intel. No requiere de ningún proceso de instalación; para usarla solo se deben descomprimir sus archivos en cualquier directorio.

2.2 Conceptos fundamentales del desarrollo web

2.2.1 Aplicaciones de página única (*Single Page Applications*)

Una aplicación de página única (SPA) es un patrón de arquitectura web utilizado extensivamente en la actualidad. Consiste en la creación de una página o aplicación web que está contenida en un único documento. Cuando es necesario actualizar su contenido, se modifica de manera dinámica sólo la parte afectada sin necesidad de recargar la página entera [6].

Las SPA a menudo son implementadas mediante el uso de peticiones AJAX (*Asynchronous JavaScript and XML*), que permiten al navegador realizar peticiones y recibir datos en segundo plano y de forma transparente al usuario [7].

Este patrón se contrapone al concepto tradicional de aplicación multipágina (MPA), según el cual las aplicaciones se componen de varias páginas HTML, cada una de ellas representando una funcionalidad específica. En estas aplicaciones, cuando se quiere actualizar su contenido es necesario realizar una llamada al servidor, que envía una página HTML nueva [8]. En la Ilustración 3 [9] se muestra el funcionamiento de las MPA frente a las SPA.

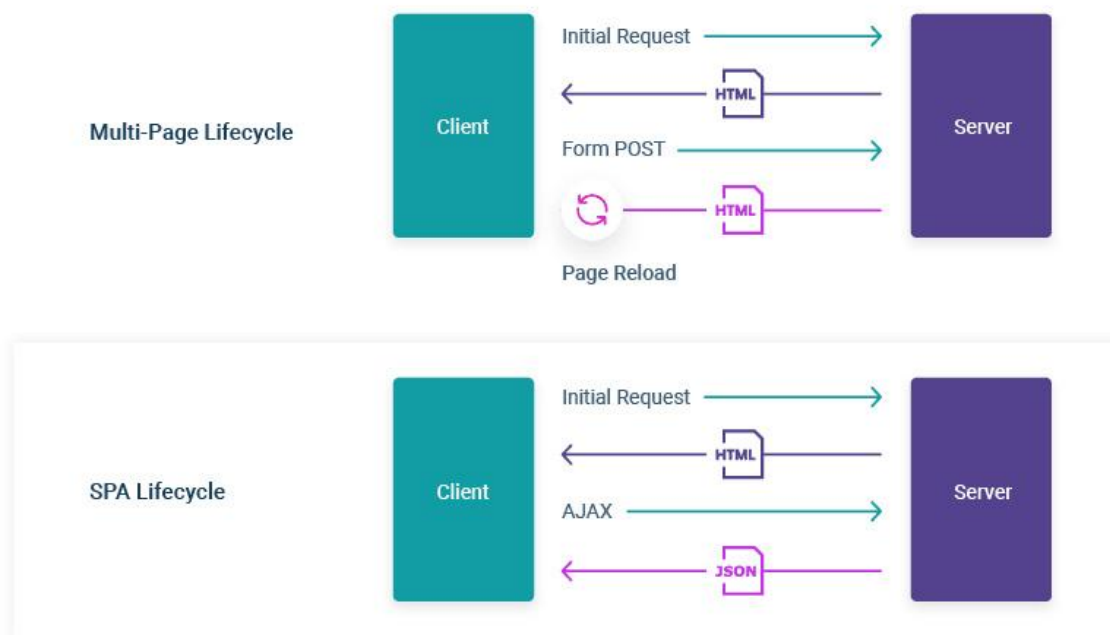


Ilustración 3. Ciclo de vida de arquitectura MPA versus SPA

El funcionamiento de las SPA les otorga ciertas ventajas respecto a las MPA. Las SPA ofrecen tiempos de respuesta significativamente más cortos y consumen menos ancho de banda, al no tener que recargar la página entera. Esta sensación de instantaneidad mejora la experiencia del usuario. En muchas ocasiones también es posible utilizarlas sin conexión o con una conexión a Internet más lenta [10]. Por otro lado, el desarrollo de las SPA da lugar a una independencia completa del *frontend* y el *backend* y permite un desarrollo modular [11].

Sin embargo, en páginas con mucho contenido la primera carga puede ser más larga, resultando en un peor tiempo de carga inicial. Otras desventajas son un peor rendimiento en la optimización de motores de búsqueda (SEO), peor funcionamiento del historial del navegador y una mayor vulnerabilidad a ataques como *cross-site scripting* (XSS) [10].

Dadas estas características, las SPA se recomiendan para aplicaciones web con pocos datos que quieran ofrecer una experiencia fluida al usuario.

2.2.2 Accesibilidad web

La accesibilidad web se refiere a que los sitios web puedan ser utilizados por el mayor número de personas posible, independientemente de sus capacidades. Aunque este término es mayoritariamente usado para la inclusión de personas

con discapacidad de cualquier tipo, también está vinculado a aquellas con limitaciones tecnológicas o ambientales, como el tipo de dispositivo que se esté utilizando o la velocidad de la conexión a Internet [12].

El World Wide Web Consortium (W3C), que es una organización internacional fundada en 1994 dedicada a la creación de estándares y directrices para el desarrollo web [13], creó la Iniciativa de Accesibilidad Web (WAI) con el fin de promover y estandarizar la accesibilidad. En 1999, esta iniciativa publicó los Pautas de Accesibilidad al Contenido Web (WCAG 1.0) [14], una serie de comprobaciones a seguir para garantizar la accesibilidad. Este documento se basa en dos temas principales: la necesidad de las páginas web de adaptarse correctamente a la situación del usuario y la necesidad de tener contenido entendible y navegable [15].

En 2008, se publicó una versión actualizada de estas pautas, las WCAG 2.0 [16]. Este documento tiene en cuenta más tipos de discapacidad y es menos dependiente de las tecnologías a utilizar. Se basa en cuatro principios que debe cumplir toda página web [17]:

- Perceptibilidad: todos los usuarios deben poder percibir con alguno de sus sentidos la información presentada.
- Operabilidad: todos los usuarios deben poder manejar los componentes de la web y navegar por ellos.
- Comprensibilidad: la información y el funcionamiento de la interfaz deben ser comprensibles.
- Robustez: la web debe ser funcional a través de todas las tecnologías actuales, incluyendo productos de apoyo, y debe ser robusta a cambios futuros en estas tecnologías.

Estos principios se desglosan en pautas generales, que a su vez contienen criterios de conformidad verificables con diferentes niveles de conformidad (A, AA y AAA). Las WCAG 2.0 han sido actualizadas varias veces desde su publicación, siendo la versión más moderna las WCAG 2.2 [18], publicadas en 2023. En esta actualización se añaden nuevos criterios de conformidad. En la Tabla 1 se muestran las pautas detalladas en las WCAG 2.2.

Pauta 1	Proporcionar alternativas textuales a cualquier contenido no textual.
Pauta 2	Proporcionar subtítulos y otras alternativas al contenido dependiente del tiempo.
Pauta 3	Crear contenido que se pueda presentar de diferentes formas sin perder información o estructura.
Pauta 4	Facilitar que los usuarios puedan ver y oír el contenido.
Pauta 5	Hacer toda la funcionalidad accesible por teclado.
Pauta 6	Proporcionar al usuario suficiente tiempo para leer y usar todo el contenido.
Pauta 7	Evitar contenido que pudiera causar convulsiones o reacciones físicas.
Pauta 8	Ayudar al usuario a navegar por el contenido.
Pauta 9	Facilitar métodos de entrada diferentes al teclado.
Pauta 10	Proporcionar texto legible y comprensible.
Pauta 11	Proporcionar contenido predecible en apariencia y uso.
Pauta 12	Ayudar a los usuarios a evitar y corregir errores.
Pauta 13	Maximizar la compatibilidad con agentes de usuario actuales y futuros.

Tabla 1. Pautas del WCAG 2.2

En el caso de España, la accesibilidad web no solo constituye una buena práctica de diseño, sino también una obligación legal. Por ejemplo, el Real Decreto 1112/2018 establece que todos los portales públicos deben cumplir, como mínimo, con el nivel AA de conformidad de las WCAG 2.1 [19].

A través de herramientas como las etiquetas semánticas de HTML o la extensión WAI-ARIA [20] se puede asegurar el cumplimiento de estos criterios. Además, existen validadores de accesibilidad, como WAVE (*Web Accessibility Evaluation Tools*) [21], que permiten encontrar algunos errores de accesibilidad durante el desarrollo.

2.3 Lenguajes web

2.3.1 Lenguaje de marcado (HTML)

HTML (*HyperText Markup Language*) es un lenguaje de marcado utilizado en la creación de páginas web. Un lenguaje de marcado es una manera de especificar la estructura de un documento y su contenido semántico, añadiéndole información adicional en la forma de etiquetas o marcas [22]. El término *hypertext* hace referencia a que el contenido no es lineal, sino que permite saltos o enlaces hacia otros puntos relacionados mediante hipervínculos.

HTML fue creado en 1991 por Tim Beners-Lee, padre de la World Wide Web y fundador del W3C. Esta primera versión consistía en 18 etiquetas, formadas por un código de etiqueta colocado entre los caracteres “<” y “>”. Estas etiquetas representan secciones de documentos como el título (etiqueta `title`), párrafos (etiqueta `p`) o listas (etiqueta `ul`) [23]. La simplicidad de este sistema hizo que se extendiese rápidamente y se utilizase como estándar de facto en la creación de sitios web.

A lo largo de los años, se ha actualizado su contenido para dar soporte a estructuras más complejas, como formularios, *scripts*, tablas, hojas de estilo externas... [24]. La versión más actual es HTML5, lanzada en 2014. Un cambio significativo de esta versión es la adición de etiquetas semánticas que especifican el contenido de etiquetas más genéricas, como `div`, que se puede usar para cualquier división del documento. Por ejemplo, se podría diferenciar entre un artículo, una sección o un encabezado. También añade soporte multimedia y mejora la compatibilidad en todos los navegadores [25].

Por otro lado, es esencial detallar la importancia de HTML en el desarrollo de páginas web accesibles [26]. El uso correcto de su sintaxis y su semántica permite un manejo y una navegación correcta de la página web con productos de apoyo. A través de atributos HTML se puede añadir también texto alternativo al contenido visual. Además, muchos elementos, como botones o hipervínculos, tienen por defecto comportamientos enfocados a la accesibilidad como navegación por teclado.

2.3.2 Hojas de estilo en cascada (CSS)

Las Hojas de Estilo en Cascada (CSS) son un lenguaje utilizado para dar estilo y mejorar la presentación de un documento escrito en un lenguaje de marcado como HTML o XML [27]. Permite definir la disposición, el color y la forma de los elementos, así como la tipografía y el tamaño del texto. Además, facilita la adaptación del contenido a diferentes dispositivos y tamaños de pantalla.

Estas hojas de estilo se definen a través de la creación de reglas. Estas reglas están formadas por un selector, que indica qué elementos van a ser afectados

por esta regla, y una declaración envuelta en llaves. La declaración contiene cualquier número de pares propiedad-valor con los que se definen las características de esos elementos [28]. En la Ilustración 4 se muestra un ejemplo de regla CSS y sus componentes. Una misma regla puede afectar a elementos de varios documentos, ahorrando tiempo a los desarrolladores.

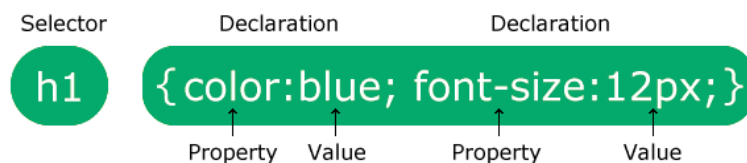


Ilustración 4. Sintaxis de una regla CSS

CSS1 surgió en 1996 al volverse más importante la apariencia de los sitios web. Fue desarrollado por Håkon Wium Lie y Bert Bos, siendo su objetivo separar el contenido de la presentación. Permitía estilos básicos, como color, márgenes, fuentes y color de fondo. Sin embargo, estaba limitado en términos de disposición de elementos. La publicación en 1998 de CSS2 añadió características como soporte para otros medios y posicionamiento avanzado de elementos, con propiedades como `position`, `top` y `left` [29].

La versión actual de CSS es CSS3, que está dividida en módulos publicados paulatinamente desde 1998. Un ejemplo es el Módulo de Caja Flexible, conocido como *flexbox* [30], que facilita la definición de alineaciones flexibles de los elementos y el espacio entre ellos a través de propiedades como `flex-direction`, que controla la dirección de los ítems contenidos en un elemento, o `align-items`, para establecer la alineación de los mismos. Otro módulo similar es el *Grid Layout* [31], con el que se divide un área en una cuadrícula flexible y se colocan libremente los elementos en su interior. Estas funcionalidades son esenciales para la creación de páginas que se adapten a diferentes resoluciones de pantalla, dispositivos o tamaños de letra, permitiendo el cumplimiento de algunos criterios de accesibilidad web.

En la actualidad, CSS continúa evolucionando de forma paralela a las necesidades de los desarrolladores web, siendo actualizado con más selectores, propiedades y funciones avanzadas. Además, la combinación de CSS con entornos de desarrollo modernos, como los *frameworks* de JavaScript (React, Vue, Svelte), ha impulsado la aparición de soluciones como *CSS Modules* o *CSS-in-JS* [32], que facilitan la modularidad y el mantenimiento de los estilos en aplicaciones web grandes.

2.3.3 Lenguaje de programación JavaScript

JavaScript es un lenguaje de programación de alto nivel utilizado junto a HTML y CSS para el desarrollo de páginas web. Es un lenguaje interpretado y fue diseñado para ser ejecutado en los navegadores web sin necesidad de ningún programa externo. Aporta dinamismo y comportamientos más complejos a las páginas web estáticas.

JavaScript fue creado por Brendan Eich, contratado por Netscape, en 1995. Fue integrado en el navegador de Netscape. Se utilizaba únicamente para tareas sencillas en el lado del cliente. Durante la primera década desde su nacimiento, JavaScript fue relativamente poco utilizado, hasta la introducción de tecnologías como AJAX en 2005 y JQuery en 2006, que permitieron interacciones con el usuario más rápidas y complejas [33].

El punto de cambio definitivo fue la salida de Google Chrome en 2008 y su motor V8, que aumentó notablemente el rendimiento del lenguaje. A partir de esta fecha, JavaScript se convirtió en el lenguaje por defecto para desarrollar aplicaciones web, integrado en muchos navegadores. La creación del *framework* Node.js poco después permitió el uso de JavaScript en el lado del servidor además de en el del cliente y fomentó aún más su popularidad [34].

JavaScript sigue un estándar llamado ECMAScript (ES), mantenido por ECMA International, que sirve de base para el lenguaje y define su sintaxis. Este estándar ha sido actualizado con los años y ha incorporado una gran variedad de funcionalidades. Entre las versiones más importantes de este documento se encuentran ES3, que añadió expresiones regulares y bloques *try-catch*, ES5, con soporte para JSON y manejo mejorado de listas, y sobre todo ES6, incluyendo clases, promesas y *array functions* [35].

En la actualidad, JavaScript es un lenguaje multiparadigma muy completo con un ecosistema rico y con una de las comunidades de desarrollo más grandes del mundo. Además de ser esencial en el desarrollo web, su uso se ha extendido a otros ámbitos como sistemas inteligentes, *WebAssembly* o creación de aplicaciones multiplataforma.

2.4 Bibliotecas y *frameworks* de JavaScript

2.4.1 Introducción a las bibliotecas y *frameworks* de JavaScript

El lenguaje JavaScript es imprescindible para el desarrollo de aplicaciones web y contiene todo lo necesario para la implementación de cualquier funcionalidad, desde las más simples hasta las más avanzadas. Sin embargo, según ha ido aumentando la complejidad de las páginas web, el desarrollo con JavaScript puro se ha vuelto más difícil de mantener y escalar. Para abordar esta complejidad surgieron las bibliotecas y *frameworks* de JavaScript, cuyo propósito es optimizar el trabajo de los desarrolladores otorgando herramientas para la realización de tareas comunes o repetitivas y ofreciendo una estructura más organizada.

Un *framework* es una plantilla o esquema conceptual que proporciona una base para el software [36]. Suele contener funciones y estructuras ya definidas que automatizan los procesos más comunes. Por lo general, es más rígido que una librería y guía al desarrollador a desenvolverse de una determinada manera. Su uso es recomendado para evitar la repetición de código, incrementar la seguridad del sistema y aumentar la eficiencia. Algunos *frameworks* de JavaScript utilizados para el desarrollo de páginas SPA son Angular, Vue.js o Svelte.

En cambio, una biblioteca es una colección de funciones o código de terceros que ofrece un conjunto de funcionalidades [37]. A diferencia de los *frameworks*, el desarrollador decide cuándo y cómo utilizar la biblioteca dentro de su propio flujo de ejecución. Tienen como beneficio la evasión de errores comunes y la optimización del tiempo de desarrollo. Las bibliotecas de Javascript más utilizadas son JQuery y React.js.

Tanto los *frameworks* como las bibliotecas se benefician del uso de herramientas externas que se encarguen del manejo de dependencias y de la configuración del entorno.

Para el manejo de dependencias, la herramienta más utilizada es npm (*Node Package Manager*) [38]. Con npm se puede compartir, descargar y ejecutar de

forma sencilla paquetes de JavaScript, además de controlar sus versiones y mantenerlos actualizados. Por otro lado, npm facilita la automatización de tareas comunes mediante *scripts* definidos en el archivo `package.json`, como la ejecución de pruebas, la compilación o el despliegue del proyecto.

Respecto a la configuración del entorno, se puede manejar con tecnologías como Vite [39]. Vite es una herramienta de construcción de software que actúa como compilador y servidor de desarrollo, permitiendo ver el estado actual del proyecto y realizar cambios inmediatos mediante el uso de HMR (*Hot Module Replacement*). Vite es compatible con los cuatro *frameworks* y librerías estudiadas en este trabajo: React.js, Vue.js, Svelte y Angular.

2.4.2 Consideraciones de accesibilidad en el desarrollo con *frameworks*

La mayoría de *frameworks* de desarrollo web consideran la accesibilidad como un aspecto fundamental. Incorporan medidas y herramientas para favorecerla y a menudo contienen guías para asegurar su correcto uso por parte de los desarrolladores. Sin embargo, algunos de sus comportamientos por defecto, como el uso de un DOM virtual o la gestión de eventos, pueden generar situaciones no accesibles si no se aplican las prácticas adecuadas. Por ello, es importante conocer las limitaciones de estos *frameworks* y corregir manualmente los comportamientos no deseados.

Algunos *frameworks*, como React.js o Vue.js, hacen uso de un DOM virtual para mejorar la eficiencia de la página web. El DOM (*Document Object Model*) es el árbol de nodos que representa la estructura del documento, mientras que el DOM virtual es una copia ligera de este en memoria que actúa de intermediario entre el DOM real y el desarrollador [40]. De esta manera, se pueden realizar cambios rápidos en nodos concretos del DOM virtual sin necesidad de actualizar el DOM real completo. Este proceso se realiza de la siguiente manera: cuando se produce un cambio en el estado, el DOM virtual crea un nuevo árbol, que es comparado con su versión anterior. Se calculan las diferencias y se actualizan sólo los nodos que presenten cambios [41].

El uso del DOM virtual puede dar lugar a problemas de accesibilidad como la pérdida del foco del teclado [42], que muestra el elemento que está seleccionado en cada momento y recibiría la entrada desde el teclado. Esta pérdida del foco se produce en ocasiones en las que el DOM virtual no reconcilia correctamente los nodos e interpreta que un nodo existente es nuevo, por lo que se elimina el nodo original y se recrea. Por ejemplo, esto puede ocurrir cuando no se gestiona adecuadamente el foco en interacciones en las que el contenido afectado no sigue directamente al elemento que las desencadena [43], en casos de *render* condicional que creen o eliminen nodos, o si se actualiza un elemento padre y se fuerza el *render* de los nodos hijo [44].

Los *frameworks* que realizan las operaciones directamente en el DOM, como Svelte o Angular, también pueden sufrir este problema en la situación de *render* condicional [45]. En ambos casos, se puede solucionar accediendo a una referencia al elemento afectado del DOM y programando manualmente el foco cuando sea necesario.

Por otro lado, estos sistemas implementan el patrón SPA. Esto proporciona eficiencia a la página y mejora la experiencia del usuario; sin embargo, poseen un problema de accesibilidad en referencia a los títulos de página. En las páginas MPA, el paso de una página a la siguiente requiere una recarga completa de la página. Como consecuencia, los lectores de pantalla vuelven a

leer el título, indicando el cambio al usuario. En las páginas SPA esto ocurre de forma transparente para el usuario. Por ello, los *frameworks* modernos ofrecen mecanismos para cambiar de forma manual los títulos de las páginas [46].

Estas tecnologías son compatibles con las etiquetas semánticas de HTML. En caso de querer crear un componente que no se adapte a ninguna etiqueta, se debe asegurar la accesibilidad mediante el uso de atributos WAI-ARIA y de una correcta gestión de eventos.

Dentro de esta gestión, destaca la implementación de la navegación por teclado. Los elementos por defecto no pueden ser seleccionados con el foco. En caso de que sea necesario, se debe establecer este comportamiento manualmente con atributos como `tab-index`. En este sentido, los atributos WAI-ARIA resultan especialmente útiles para complementar la semántica y describir el comportamiento de los componentes.

WAI-ARIA es una especificación escrita por el W3C que describe roles y atributos que aportan información adicional a las tecnologías de asistencia [47]. Los roles, como `menuitem`, `searchbox` o `checkbox`, proporcionan una breve descripción y definen las propiedades que puede tener el elemento. Los atributos especifican el estado del componente (ej. `aria-checked` o `aria-expanded`). Los marcos de desarrollo de JavaScript soportan y recomiendan el uso de estas etiquetas.

Por último, WAI-ARIA también se debe usar para anunciar los cambios dinámicos que ocurren en la aplicación, a través de las *ARIA Live Regions* [48]. Las regiones marcadas con este atributo son observadas por los lectores de pantalla. Cuando se detecta un cambio, se vuelve a leer su contenido. Esta funcionalidad mejora en gran medida la experiencia de los usuarios con este producto de apoyo, puesto que no disponen de otra manera de detectar este tipo de cambios.

2.4.3 Bibliotecas y *frameworks* principales

2.4.3.1 React.js

React.js es una librería de JavaScript de código abierto mantenida por Meta y utilizada para la creación de interfaces que siguen el patrón SPA. Fue publicada en 2013 por Jordan Walke, un desarrollador de Meta. Desde ese momento, experimentó un gran crecimiento, hasta convertirse en la librería de *frontend* más utilizada en la actualidad [49]. Cuenta con una comunidad muy activa de contribuyentes y desarrolladores.

A diferencia de JavaScript o de otras librerías como JQuery, React.js no es imperativo, sino que sigue el paradigma de la programación declarativa. Esto significa que el programador describe la lógica del programa sin detallar el control del flujo, y es el sistema el que se encarga de determinar cómo ejecutar la tarea. Además, hace uso del VDOM o DOM virtual, que permite actualizar solo aquellos nodos con cambios en lugar del DOM entero [50].

Un aspecto muy importante de React.js es la creación de componentes reutilizables [51]. Las interfaces se dividen en componentes independientes que almacenan su propio estado. Estos componentes son funciones de JavaScript que devuelven marcado llamado JSX [52], que tiene una sintaxis muy similar a la de HTML. Así, se mantienen en el mismo archivo la lógica del funcionamiento y el marcado de cada componente. El JSX tiene la ventaja de que permite la inserción de código de JavaScript directamente en el marcado, envuelto en llaves. En el Fragmento de Código 1 se muestra un ejemplo de un componente muy simple en el que se devuelve un encabezado con el texto “Hola,” seguido

del valor de la variable declarada previamente llamada `nombre`, cuyo contenido en este caso es “María”.

```
function Saludo() {
  const nombre = "María";

  return (
    <div>
      <h1>Hola, {nombre}</h1>
    </div>
  );
}
```

Fragmento de Código 1. Ejemplo de un componente en React.

Para desarrollar componentes más complejos, es necesario conocer el concepto de *hooks* [53]. Estos son funciones que permiten usar funcionalidades de React típicamente disponibles solo en clases en componentes funcionales. A través de ellos, se puede guardar y modificar el estado del componente o manejar su ciclo de vida. Existen *hooks* incorporados como `useState`, que permite declarar variables de estado, `useEffect`, para manejar efectos secundarios (como peticiones o suscripciones), y `useContext`, para acceder al contexto global de la aplicación. Además, los desarrolladores pueden crear *custom hooks* para encapsular lógica reutilizable entre distintos componentes.

Aparte de estas características nativas de React, existen numerosos paquetes externos que amplían sus capacidades. Para este proyecto, algunos de ellos pueden suponer una gran ayuda al desarrollador:

- React Redux [54]: este paquete se utiliza para gestionar el estado global de la aplicación de manera segura y organizada. Esto se realiza a través de un *reducer*, que recibe el estado actual de la aplicación y una acción, transforma el estado adecuadamente y devuelve una copia actualizada. De esta manera, se puede gestionar en una sola función cómo tratar cada acción, minimizando errores y permitiendo acceder a este estado global desde cualquier componente independientemente de su posición en el árbol de nodos.
- React-resizable-panels [55]: este proyecto facilita la creación de pantallas formadas por ventanas redimensionables. Permite agrupar las ventanas libremente de forma vertical u horizontal y anidar grupos. Además, se puede guardar la configuración del usuario en almacenamiento local o almacenamiento externo persistente. La dimensión de los paneles se puede controlar por completo con el teclado, un factor muy importante para garantizar la accesibilidad.
- React-i18next [56]: este paquete ayuda en la internacionalización de una interfaz, es decir, en la adición de varios idiomas entre los que pueda elegir el usuario. Esto se realiza a través de un componente configurable que maneja las opciones de cada idioma así como el idioma por defecto.

2.4.3.2 Vue.js

Vue.js es un *framework* de JavaScript de código abierto desarrollado por Evan You. Su objetivo era diseñar un *framework* para el desarrollo de interfaces SPA, similar a Angular pero más ligero y amigable [57]. Su curva de aprendizaje accesible le hizo adquirir popularidad desde sus inicios en 2014. Es el tercer *framework* de *frontend* de JavaScript más utilizado, por detrás de React y Angular [49].

Vue.js destaca por su pequeño tamaño y facilidad de uso [58]. Es muy flexible, por lo que existen varias posibilidades en la creación de un mismo funcionamiento. Aunque aporta libertad, esta característica también supone un riesgo porque permite el desarrollo de código de baja calidad. Por otro lado, aunque posee una comunidad dedicada en crecimiento, no tiene la misma cantidad de apoyo y literatura sobre él como otros *frameworks* o librerías más populares.

Al igual que otros *frameworks*, Vue.js permite la declaración de componentes web reutilizables [59]. Se declaran en un archivo propio, donde se escribe en tres secciones separadas el marcado HTML (*template*), el funcionamiento en JavaScript (*script*) y el estilo (*style*), como se puede ver en el ejemplo de componente en el Fragmento de Código 2. Este sistema aumenta la legibilidad, la modularidad del código y la simplicidad del proyecto.

```
<template>
  <div>
    <h1>Hola, {{ nombre }}</h1>
  </div>
</template>

<script>
export default {
  name: "Saludo",
  data() {
    return {
      nombre: "María"
    };
  }
};
</script>

<style scoped>
h1 {
  font-size: 2rem;
}
</style>
```

Fragmento de Código 2. Ejemplo de un componente en Vue.js

Otras características de Vue.js son el uso del DOM virtual, la reactividad, que establece una vinculación bidireccional entre las variables de un componente y partes de la plantilla asegurando actualizaciones automáticas de su valor, un sencillo sistema de eventos y transiciones, y un *router* integrado para permitir cambios de pantalla en webs SPA [60].

2.4.3.3 Svelte

Svelte es un *framework* para desarrollar interfaces web. Fue creado en 2016 por Rich Harris. A pesar de ser menos conocido que otros *frameworks* contemporáneos, es muy admirado por los desarrolladores que lo conocen [49].

A diferencia de otros marcos de trabajo, Svelte no hace uso de un DOM virtual. En su lugar, compila los componentes a HTML, CSS y JavaScript de forma óptima en tiempo de construcción. Al ahorrarse las transformaciones en tiempo de ejecución, aporta una gran eficiencia y rapidez a las aplicaciones. Además,

como la sintaxis de Svelte es muy parecida a la de HTML, CSS y JavaScript, tiene una curva de aprendizaje muy pequeña [61].

Al igual que en Vue.js, se puede declarar un componente en un único archivo dividido en tres secciones: lógica, estilo y marcado. Todas las variables son reactivas por defecto; su valor se actualiza automáticamente en la interfaz. En el Fragmento de Código 3 se puede observar la sintaxis y estructura de los componentes de Svelte.

```
<!-- Archivo: Saludo.svelte -->

<script>
  let nombre = "María";
</script>

<style>
  h1 {
    color: #2b7a78;
    font-family: Arial, sans-serif;
  }
</style>

<!-- Marcado HTML con interpolación de variable -->
<h1>Hola, {nombre}!</h1>
<p>Bienvenida a la aplicación.</p>
```

Fragmento de Código 3. Ejemplo de un componente en Svelte.

En 2021, se publicó SvelteKit, una herramienta complementaria a Svelte que proporciona funcionalidades necesarias para construir una aplicación completa. SvelteKit combina el *render* en el servidor y en el cliente. Al cargar una página por primera vez, el servidor genera el HTML y lo envía al navegador, lo que mejora la velocidad de carga inicial y el posicionamiento en buscadores. Después, cuando el usuario navega por la aplicación, el código JavaScript del cliente toma el control mediante un proceso llamado hidratación, que permite cambiar de página sin recargar todo el sitio. De esta forma, la experiencia de uso es más fluida y rápida. Además, SvelteKit permite configurar este comportamiento según las necesidades del proyecto [62]. También contiene integración con Vite para poder usar su entorno de desarrollo con HMR [63].

Aunque Svelte tiene una comunidad de contribuyentes más pequeña, cuenta con algunos paquetes que pueden ayudar al desarrollo de este TFG. Por ejemplo, *svelte-splitpanes* [64] ofrece paneles redimensionables y flexibles con funcionalidades avanzadas como tamaño máximo y mínimo, *render* condicional de paneles y soporte para dispositivos móviles. Sin embargo, este paquete no cuenta con guardado de la configuración de paneles del usuario ni con control por teclado.

2.4.3.4 Angular

Angular es un *framework* de código abierto desarrollado por Google y publicado en 2016. Al igual que el resto, sirve para crear aplicaciones web siguiendo el patrón SPA. Este proyecto surgió de una reescritura completa de un *framework* anterior, AngularJS. Esta versión previa apareció en 2010 y tuvo mucha popularidad; sin embargo, las funcionalidades cada vez más avanzadas de JavaScript y las necesidades crecientes de los marcos de trabajo no podían ser adaptadas a él, por lo que fue necesario empezar de cero. De ahí surgió Angular

2.0, o Angular a secas, uno de los *frameworks* más utilizados en la actualidad [65].

Angular recomienda el uso de TypeScript [66], un lenguaje de programación basado en JavaScript con tipado fuerte y sintaxis adicional, que es posteriormente convertido a JavaScript para tener compatibilidad con los navegadores. TypeScript es preferido por muchos desarrolladores por su inteligibilidad, limpieza del código y reducción de errores en ejecución [67].

Angular es un marco de trabajo muy completo con una amplia gama de servicios. Por ejemplo, cuenta con una interfaz de línea de comandos para crear, desarrollar y testear las aplicaciones [68]. Además, amplía las funcionalidades de HTML con directivas, como el añadido o la eliminación de elementos del DOM según una condición, y permite la creación de plantillas dinámicas. Tiene vinculación bidireccional de variables entre el HTML y los objetos creados con TypeScript [69].

Para la creación de componentes, utiliza el patrón MVC (modelo-vista-controlador). Cada componente se genera con cuatro archivos: una plantilla HTML, una hoja de estilos CSS, un archivo de pruebas unitarias y un archivo de código TypeScript, que contendrá los datos del componente (el modelo) y métodos para responder a eventos y modificar el componente (el controlador) [70].

La complejidad de las funcionalidades de Angular implica una curva de aprendizaje empinada en comparación con otras herramientas. Este marco es recomendable para proyectos grandes y de larga duración, donde se quiera priorizar la escalabilidad y mantenibilidad.

2.5 Herramientas de desarrollo

2.5.1 Control de versiones: Git y GitLab

Git es un sistema de control de versiones distribuido de código abierto. Fue creado en 2005 por Linus Torvalds, con la intención de crear un programa gratuito mejor que el sistema de control de versiones BitKeeper, empleado en ese momento para su proyecto del kernel de Linux. Su objetivo era crear un sistema sencillo, rápido y completamente distribuido con soporte para proyectos grandes y no lineales [71].

Para usar Git se usa un repositorio local, que al ser distribuido puede manejar todas las versiones del proyecto. En este repositorio, tras modificar o añadir un archivo, el desarrollador puede añadirlo a un área de preparación con el comando `git add <archivo>`. Después, se pueden confirmar todos los cambios de los archivos en el área de preparación con el comando `git commit`. El repositorio local se puede comunicar con una copia remota del repositorio mediante comandos como `git push` y `git pull`. En la Ilustración 5 se puede observar este comportamiento gráficamente. Cada confirmación tiene un código *hash* mediante el cual se puede regresar a puntos anteriores y revertir los cambios si es necesario [72].

Además, es posible colaborar de manera no lineal en el proyecto a través de la creación de ramas. Una rama encapsula los cambios que se producen en ella, aislándolos del resto de ramas, hasta que se decida fusionar ramas y comparar los cambios.

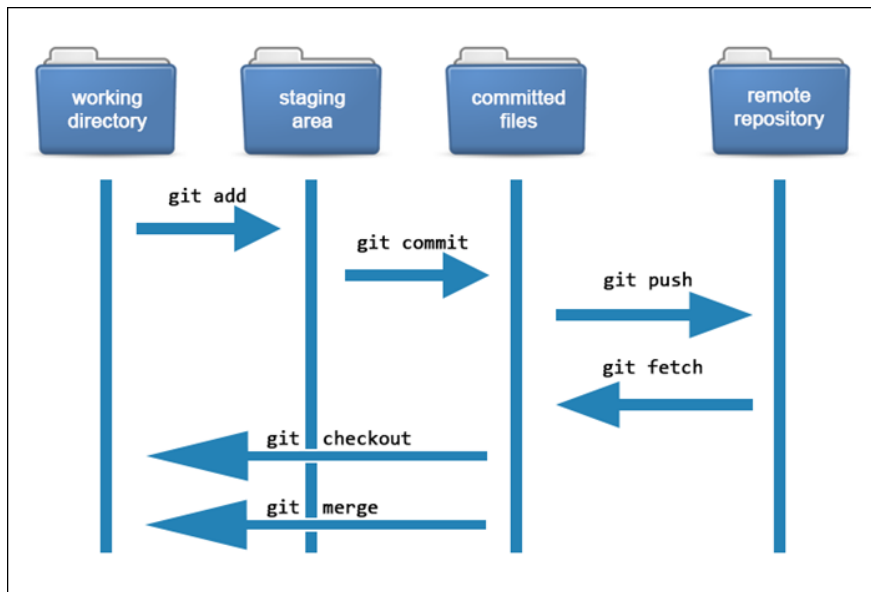


Ilustración 5. Flujo de Git.

El uso de Git se puede complementar con una plataforma que aloje repositorios de Git de forma remota, como GitLab. GitLab [73] es una página web que ofrece una gran variedad de herramientas para la gestión de proyectos *software*. Permite el manejo completo de repositorios Git, a través del cual se pueden proteger ramas, conceder o denegar permisos a los contribuyentes del proyecto y añadir procesos de integración continua. Además, posee otros servicios orientados a la organización del proyecto, como la creación y gestión de tareas, el seguimiento y la resolución de problemas del código o la inclusión de documentación del proyecto [74].

En la actualidad, el uso de Git y sus herramientas complementarias se considera esencial para cualquier proyecto. Supone una gran ayuda para la gestión del proyecto y facilita el proceso de despliegado y mantenimiento una vez se finalice el desarrollo. De esta manera, el desarrollador puede seguir un proceso más seguro y organizado y enfocarse en el desarrollo del código.

2.5.2 Entorno de desarrollo integrado: Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft. Su facilidad de uso, alto nivel de personalización e integración con otras herramientas de desarrollo lo han convertido en el IDE (entorno de desarrollo integrado) con más usuarios en la actualidad [75].

Esta herramienta proporciona soporte integrado para la mayoría de lenguajes de programación. Ofrece resaltado de sintaxis y detección automática de errores y avisos. Además, con su sistema de Intellisense [76] o autocompletado inteligente, basándose en el estado y en el lenguaje utilizado, agiliza la escritura del código y reduce la cantidad de errores. Este sistema también permite obtener descripciones de funciones de librería, reduciendo la necesidad de consultar la documentación externa.

Otra ventaja de Visual Studio Code es el sistema de depuración integrada. Este depurador avanzado permite la adición de puntos de ruptura y la visualización del estado de las variables. Es compatible con una gran variedad de lenguajes. Gracias a esta herramienta se allana considerablemente la búsqueda y corrección de errores en el código [77].

Además de las herramientas nativas, Visual Studio Code contiene un mercado de extensiones donde el usuario se puede instalar muy fácilmente extensiones creadas por otros usuarios. Estas pueden tratar temas estéticos, como aquellas que formatean automáticamente el código, o temas técnicos, como el Intellisense y la integración con un lenguaje específico de programación. Gracias a su gran comunidad de desarrolladores, existen extensiones que integran el programa con muchas otras herramientas de programación; entre ellas, plataformas de contenedores como Docker o sistemas de controles de versiones como Git [78].

Estas funcionalidades y su interfaz altamente configurable convierten a Visual Studio Code en un IDE adaptable a todos los proyectos y a los gustos de cada desarrollador. Además, la centralización de muchas acciones necesarias, como el control de versiones, el despliegue de un entorno de desarrollo o la consulta de documentación, supone un aumento de la eficiencia del programador.

2.5.3 Prototipado y diseño de interfaces: Figma

Figma es una herramienta de edición gráfica y prototipado que permite esbozar diseños de interfaces para aplicaciones web o móviles. También se puede utilizar para la creación de presentaciones o la edición de imágenes. Es una tecnología colaborativa en línea basada en la nube, de forma que varias personas pueden editar el mismo archivo simultáneamente y los cambios se guardan al instante [79].

Figma se utiliza a menudo para el desarrollo de prototipos de alta fidelidad de una interfaz. Un prototipo de alta fidelidad es una representación del sistema final que sirve para aclarar la idea del proyecto, hacer pruebas a usuarios y facilitar posteriormente la tarea a los desarrolladores [80]. La creación del prototipo se divide en dos partes: el diseño de la interfaz y el prototipado.

En el diseño de la interfaz se crean todas las pantallas del proyecto que se necesiten, colocando manualmente formas, cajas de texto, imágenes... Figma ofrece varias herramientas para facilitar este proceso. Por ejemplo, se pueden crear componentes reutilizables, con variables para todas sus versiones. De esta forma, si es necesario hacer un cambio en el componente, se puede modificar únicamente en la declaración del componente y el cambio se actualiza en todas sus instancias. Otra opción muy útil es el *autolayout*, que hace más eficiente la colocación de elementos dentro de otros y mantiene su posición en caso de redimensionamiento.

El prototipado consiste en la definición de las interacciones entre pantallas. Con un simple sistema de conexión de elementos, se puede planificar un evento cuando el usuario realice una acción, como hacer clic a un elemento o pulsar una tecla, o cuando pase una determinada cantidad de tiempo. De esta manera se simula el comportamiento del sistema real. Se pueden establecer varios flujos de pantalla dedicados a diferentes funcionalidades.

Una vez se termina la fase de diseño, Figma ayuda a los desarrolladores en la implementación del prototipo con *Dev Mode* [81]. Este modo permite inspeccionar los elementos y genera fragmentos de CSS con los datos que conoce, como el color de fondo, el borde, el tamaño... Este modo de Figma no está incluido en su plan gratuito.

Por último, Figma cuenta con una gran comunidad de usuarios que comparte librerías de componentes y *plugins*, agilizando aún más el trabajo. Gracias a todas estas funciones, Figma se ha consolidado como una de las herramientas más utilizadas en el diseño de interfaces centradas en el usuario.

3 Planteamiento del problema

Como paso previo al desarrollo del proyecto, se presenta a continuación la Especificación de Requisitos Software (ERS), para cuya redacción se ha seguido el estándar de IEEE 830 [82].

3.1 Introducción

3.1.1 Propósito

El objetivo de la ERS es la definición de unas directrices a las que se debe adherir el proyecto a desarrollar, además de la aclaración de las características generales del producto, como son su público objetivo o el contexto de su desarrollo. Estas pautas sirven de apoyo tanto en la fase de diseño como en la de pruebas, puesto que proporcionan una lista de objetivos verificables a comprobar.

3.1.2 Ámbito del sistema

El sistema completo a desarrollar ha sido nombrado ENS2025, en consonancia con su predecesor, el ENS2001, actualizando el año de desarrollo del mismo.

Este TFG solo engloba el desarrollo de la interfaz gráfica del ENS2025. Como consecuencia, los requisitos están detallados desde la perspectiva de este módulo y no del sistema en su totalidad.

La interfaz ofrece la posibilidad de cargar un archivo de ensamblador ENS2025, ensamblarlo y obtener los resultados. Posteriormente, se puede simular la ejecución del programa y observar el estado interno de la máquina. Para ello se comunica con otros dos módulos: un módulo de ensamblado y otro de ejecución.

Para más información sobre su alcance se puede consultar el capítulo de Análisis de Impacto de esta memoria.

3.1.3 Definiciones, acrónimos y abreviaturas

ENS2025. Se refiere tanto al nombre del sistema completo como al lenguaje ensamblador que acepta este sistema. Es el sucesor del sistema ENS2001.

ERS. Especificación de Requisitos Software.

TFG. Trabajo de Fin de Grado.

WCAG 2.2. Pautas de Accesibilidad para el Contenido Web, estándar internacional para asegurar que la web sea accesible a personas con discapacidad.

Registros. Unidades de almacenamiento de alta velocidad internas de la CPU. El sistema ENS2025 cuenta con los siguientes: registros de propósito general (R0-R9), registros índice (IX e IY), contador de programa (PC), puntero de pila (SP) y registro acumulador (A).

Biestables de estado. Indicadores binarios (0 o 1) que reflejan condiciones específicas tras la ejecución de una instrucción.

Ventanas del sistema. Componentes modulares de la interfaz gráfica que organizan la información. La interfaz tiene las siguientes:

- **Editor:** Ventana para la creación y modificación del código fuente en ensamblador. Se alterna con la ventana de desensamblado.

- **Desensamblado:** Vista que muestra la traducción del código máquina (binario) de la memoria a mnemónicos de ensamblador. Se alterna con la ventana del editor.
- **Memoria:** Tabla que permite visualizar y editar el contenido de la memoria principal.
- **Registros:** Panel donde se monitoriza y modifica el valor de los registros internos y biestables.
- **Consola:** Ventana para la introducción de datos (entrada) y visualización de resultados (salida) del programa en ejecución.
- **Logs:** Registro de eventos que muestra mensajes informativos, avisos y errores del sistema.

3.1.4 Referencias

Todas las referencias de la memoria se encuentran en el capítulo de Bibliografía.

3.1.5 Visión general de la ERS

Esta ERS, integrada en la memoria del TFG, consta de tres secciones claramente diferenciadas: una introducción, donde se pone en contexto el sistema, una descripción general, en la que se detallan algunos datos sobre el futuro uso del sistema y las circunstancias de su desarrollo, y la parte principal del documento, los requisitos específicos, divididos por categorías.

En ocasiones se suele incluir una última sección de anexos; sin embargo, debido a la ausencia de anexos relevantes, no se ha visto útil incorporarla.

3.2 Descripción general

3.2.1 Perspectiva del producto

El sistema estará comunicado con otros dos módulos:

- Módulo de ensamblado: permite enviar el contenido de un programa de ensamblador ENS2025 y recibir los errores de sintaxis del mismo o su mapa de memoria si el archivo es correcto.
- Módulo de ejecución: permite la ejecución en lote o paso a paso del programa ya ensamblado.

Esta división en módulos tiene como consecuencia que los requisitos de la interfaz no coincidan exactamente con las funcionalidades del ENS2025. Por ejemplo, un requisito del ENS2025 como el ensamblado de archivos se desglosa en otros más específicos para la interfaz gráfica: la subida de archivos del usuario, la llamada al módulo de ensamblado con los datos del archivo y la visualización de los resultados recibidos. No se ha redactado un ERS del ENS2025 completo; sus requisitos se pueden derivar de la unión de los requisitos de los tres módulos que lo componen, que forman parte de tres Trabajos de Fin de Grado diferentes.

3.2.2 Funciones del producto

Las funciones del producto se pueden dividir en tres categorías: el ensamblado y la edición de un archivo de ensamblador ENS2025, el control de la simulación y la configuración de la interfaz. A continuación, se comenta cada una:

- Ensamblado y edición de archivos: se permite subir un archivo de la máquina del usuario, editarlo, ordenar su ensamblado y observar los resultados.

- Control de la simulación: el usuario puede controlar la ejecución de un programa y comunicarse con el módulo de entrada y salida a través de una consola. También puede observar y modificar el estado la memoria y los registros de la máquina.
- Configuración de la interfaz: existirán aspectos configurables como el formato de las direcciones de memoria y de su contenido, la dirección de la pila de memoria o el idioma y tema de la interfaz.

3.2.3 Características de los usuarios

El sistema está destinado a los alumnos de Traductores de Lenguajes. Este público posee un perfil técnico y conocimiento previo de ensamblador, al ser estudiantes de tercer curso de Ingeniería Informática. Sin embargo, se pretende que el sistema sea útil a cualquier persona que busque un simulador sencillo de ensamblador con fines didácticos.

Por otro lado, el sistema debe adaptarse a todos los posibles usuarios independientemente de sus capacidades, por lo que la accesibilidad deberá ser una prioridad en su desarrollo.

3.2.4 Restricciones

Una restricción importante es la limitación temporal del desarrollo del producto. Al tratarse de un TFG, se dispone de 297 horas totales de trabajo distribuidas a lo largo de cinco meses, de las cuales un porcentaje significativo está dedicado a tareas no directamente relacionadas con la implementación del producto, como el desarrollo de esta memoria.

3.2.5 Suposiciones y dependencias

Los requisitos del sistema podrían verse modificados si cambian las necesidades de la asignatura de Traductores de Lenguajes. Unos ejemplos de posibles cambios son la modificación del lenguaje ensamblador que acepta o la necesidad de alguna funcionalidad extra.

Al depender de otros módulos, el sistema también se deberá modificar si se produce un cambio en alguno de ellos, sobre todo si afecta a la comunicación entre ellos.

Por último, también sería necesario adaptar el sistema si recibe un tráfico muy superior al estimado.

3.2.6 Requisitos futuros

Las posibles mejoras al sistema se pueden observar en el capítulo de Futuras líneas de trabajo.

3.3 Requisitos específicos

3.3.1 Interfaces externas

3.3.1.1 Comunicación con el módulo de ensamblado

RI-01. Relación cliente-servidor.

La interfaz establecerá una relación cliente-servidor, actuando como cliente, con el módulo de ensamblado.

Prioridad: alta.

RI-02. Envío del contenido del programa.

La interfaz debe enviar el contenido del archivo subido por el usuario al módulo de ensamblado.

Prioridad: alta.

RI-03. Respuesta del módulo de ensamblado.

La interfaz debe recibir la respuesta del módulo de ensamblado, que debe contener una lista de errores de ensamblado, que puede estar vacía, el contenido de la memoria ensamblada, la dirección inicial y final de la región de código y una lista de etiquetas junto a sus traducciones a direcciones. Los errores se deben representar con un código de error, y una lista de parámetros si aplica.

Prioridad: alta.

3.3.1.2 Comunicación con el módulo de ejecución

RI-04. Compartición de la memoria.

El sistema debe compartir la memoria con el módulo de ejecución. La memoria debe ser del tipo `Uint16Array` y contener 65.536 direcciones de memoria.

Prioridad: alta.

RI-05. Solicitud de ejecución del programa.

Cuando el usuario lo requiera, la interfaz debe solicitar al módulo de ejecución la ejecución en lote o de una instrucción. El módulo de ejecución debe comunicar el éxito o fracaso de la operación, y un código de error con sus posibles parámetros si se produce una excepción de la máquina.

Prioridad: alta.

RI-06. Consistencia del contenido.

El sistema debe mantener consistencia en el contenido de los registros, los biestables, la salida y la memoria entre el módulo de ejecución y la interfaz en todo momento. Esto requiere comunicar los cambios realizados por el usuario al módulo de ejecución, y actualizar el contenido de la interfaz cuando se ejecute el programa.

Prioridad: alta.

RI-07. Paso de entrada.

Cuando el programa requiera una entrada, el sistema debe detectarlo y mostrar la sección de entrada al usuario. Cuando el usuario introduzca una entrada válida, el sistema debe pasar su contenido al módulo de ejecución.

Prioridad: alta.

RI-08. Puntos de ruptura.

El sistema debe comunicar el establecimiento y la eliminación de puntos de ruptura al módulo de ejecución, para que pueda detener su ejecución en dichos puntos.

Prioridad: alta.

3.3.2 Funciones

3.3.2.1 Aspectos generales

RF-01. Ventanas del sistema.

El sistema deberá contener 6 ventanas diferenciadas: desensamblado, editor, *logs*, registros, memoria y consola. El usuario podrá alternar entre la ventana del desensamblado y del editor, de forma que no se podrán ver las dos simultáneamente. El sistema permitirá controlar la visibilidad y el tamaño de las ventanas, de forma que el usuario pueda elegir qué ventanas tener abiertas y cuánto deben ocupar.

Prioridad: alta.

RF-02. Subida de un archivo.

El sistema permitirá cargar un archivo de la máquina personal del usuario. Este archivo deberá ser un archivo de texto o poseer la extensión *.ens*. El tamaño máximo del archivo será de 1MB. En cuanto se cargue el archivo, el sistema deberá ordenar su ensamblado, mostrar el resultado del mismo y, en caso de éxito, actualizar el estado de la máquina con los datos del programa.

Prioridad: alta.

RF-03. Ejecución en lote.

El sistema permitirá la ejecución en lote de un programa ya ensamblado; el programa se ejecutará hasta el final o hasta que llegue a un punto de ruptura. Tras la ejecución, deberá mostrar el estado actualizado de la máquina.

Prioridad: alta.

RF-04. Ejecución instrucción a instrucción.

El sistema permitirá la ejecución de la siguiente instrucción. Tras la ejecución, deberá mostrar el estado actualizado de la máquina.

Prioridad: alta.

RF-05. Reiniciar estado de la máquina.

Tras el ensamblado del programa cargado, el sistema permitirá reiniciar el estado de la máquina, recuperando el valor que tenían la memoria y los registros antes del ensamblado.

Prioridad: media.

3.3.2.2 Ventana de editor

RF-06. Editar archivo.

El sistema permitirá la edición del archivo subido. Estos cambios no modificarán el archivo original. Si el usuario modifica el archivo, el programa será reensamblado la próxima vez que el usuario ordene una ejecución.

Prioridad: media.

RF-07. Crear archivo vacío.

El sistema permitirá crear un archivo nuevo vacío.

Prioridad: baja.

RF-08. Descargar archivo.

Se permitirá descargar el archivo subido al sistema en ese momento, con los cambios que se hayan realizado desde su subida o creación. El archivo descargado mantendrá el nombre del archivo original si lo posee, o tendrá un nombre genérico en caso contrario.

Prioridad: baja.

RF-09. Abrir caso de ejemplo.

El sistema permitirá abrir programas de ejemplo de ensamblador. Estos programas deben ser ensamblados según se abren, al igual que los archivos subidos.

Prioridad: baja.

3.3.2.3 Ventana de memoria

RF-10. Mostrar memoria.

El sistema mostrará el valor de las celdas de memoria, acompañado de su dirección y un indicador de su región, que puede ser código, pila o ninguna. Este indicador también señalará la dirección a la que apuntan los registros IX, IY, PC y SP. Las direcciones y los valores se deberán adaptar al formato elegido en la configuración.

Prioridad: alta.

RF-11. Modificación de la memoria.

El sistema permitirá la modificación de un valor de memoria. Si el formato del contenido de la memoria es ASCII, el usuario podrá introducir un carácter imprimible y el valor de esa celda pasará a ser el código de ese carácter. Si el formato es decimal o hexadecimal, se considerará que el valor introducido es decimal si no tiene ningún prefijo y hexadecimal si comienza por el prefijo '0x'.

Prioridad: media.

RF-12. Ir a una dirección.

El sistema permitirá cargar la tabla de memoria a partir de una dirección introducida por el usuario, del valor de un registro (PC, SP, IX o IY) o del valor de una etiqueta.

Prioridad: alta.

RF-13. Duplicar tabla de memoria.

El sistema ofrecerá la opción de ver dos zonas independientes de la memoria. La navegación por una de las tablas de memoria no implica ningún cambio en la otra. Los cambios en la memoria se deberán ver inmediatamente actualizados en ambas instancias.

Prioridad: media.

RF-14. Mostrar las delimitaciones de las regiones de memoria.

El sistema mostrará la dirección inicial y final de las regiones de código y pila.

Prioridad: baja.

3.3.2.4 Ventana de registros

RF-15. Mostrar registros y biestables.

El sistema mostrará el valor de los registros y de los biestables de estado. Los valores de los registros seguirán el formato decimal o hexadecimal según el seleccionado en la configuración. Si el formato seleccionado es ASCII, entonces los registros se adaptarán al formato que tengan las instrucciones.

Prioridad: alta.

RF-16. Modificación de registros y biestables.

El sistema permitirá modificar el valor de los registros y biestables.

Independientemente del formato seleccionado en la configuración, si el usuario introduce un número con el prefijo '0x' será tratado como un número hexadecimal, y en caso contrario será tratado como decimal.

Los biestables solo aceptarán los valores 0 o 1.

Prioridad: media.

3.3.2.5 Ventana de desensamblado

RF-17. Mostrar desensamblado.

El sistema mostrará el desensamblado de la memoria y la dirección de cada instrucción. La dirección deberá adaptarse al formato elegido en la configuración.

Prioridad: alta.

RF-18. Modificación de instrucciones desensambladas.

El sistema permitirá modificar instrucciones sin necesidad de volver a ensamblar el programa, alterando directamente la memoria, siempre que la modificación no cambie el tamaño de la instrucción original.

Prioridad: baja.

RF-19. Ir a una instrucción.

El sistema permitirá ver el desensamblado a partir de una dirección de memoria introducida por el usuario, del valor de un registro (PC, SP, IX o IY) o del valor de una etiqueta.

Prioridad: media.

RF-20. Establecer y eliminar puntos de ruptura.

El sistema permitirá establecer y eliminar puntos de ruptura en cualquier instrucción.

Prioridad: alta.

3.3.2.6 Ventana de consola

RF-21. Introducción de una entrada.

Cuando el programa lo requiera, el sistema pedirá una entrada al usuario del tipo adecuado: número entero, cadena o carácter. El usuario podrá entonces introducir la entrada pedida. El sistema procesará la entrada y, si es correcta, ordenará la continuación de la ejecución.

Prioridad: alta.

RF-22. Mostrar salida del programa.

El sistema mostrará la salida que produzca el programa.

Prioridad: alta.

RF-23. Limpiar la consola.

El sistema permitirá eliminar todo el contenido actual de la consola.

Prioridad: baja.

3.3.2.7 Ventana de logs

RF-24. Mostrar *logs*.

El sistema mostrará información sobre la ejecución, avisos y excepciones en las siguientes ocasiones:

- Se mostrará un mensaje informativo si se realizan con éxito acciones iniciadas por el usuario, específicamente: si se carga correctamente un archivo, si se ensambla correctamente, al ejecutar en lote, al ejecutar una instrucción o al reiniciar el estado de la máquina.
- Se mostrará un aviso si durante la ejecución el SP invade la región de código, el PC invade la región de pila o si se sobrescribe el código (se escribe en la región de código).
- Se mostrará un mensaje de error si ocurre un problema con la lectura de un archivo, si se reciben errores del ensamblado del programa o si se produce una excepción durante la ejecución del programa.

Los mensajes informativos, avisos y errores deben poder ser diferenciados por el usuario y deben explicar claramente la causa del mensaje.

Prioridad: alta.

RF-25. Limpiar *logs*.

El sistema permitirá eliminar todos los *logs* visibles en ese momento.

Prioridad: baja.

3.3.2.8 Configuración

RF-26. Seleccionar formato de las instrucciones.

El sistema permitirá configurar el formato de las instrucciones entre hexadecimal y decimal. En el formato hexadecimal, las direcciones deberán comenzar por el prefijo '0x' seguido del valor hexadecimal de cada *byte* que forme la dirección (por ejemplo: 0x001A). En el formato decimal, se mostrarán solo las cifras significativas del valor decimal de la dirección, sin ningún prefijo (por ejemplo: 53). Este formato afectará a las instrucciones mostradas en la ventana de desensamblado y en la ventana de memoria.

El formato por defecto será decimal.

Prioridad: baja.

RF-27. Seleccionar el formato del contenido.

El sistema permitirá configurar el formato del contenido de la memoria y de los registros.

En el formato hexadecimal, las direcciones deberán comenzar por el prefijo '0x' seguido del valor hexadecimal de cada *byte* que forme la dirección (por ejemplo: 0x001A). En el formato decimal, se mostrarán solo las cifras significativas del valor decimal de la dirección, sin ningún prefijo (por ejemplo: 53). En el formato

ASCII, si el valor corresponde al código ASCII de un carácter imprimible, se mostrará dicho carácter, en caso contrario, se mostrará la cadena ‘**’, que representa cualquier carácter no imprimible.

El formato por defecto será decimal.

Prioridad: baja.

RF-28. Cambiar la dirección de la pila.

El sistema permitirá configurar la dirección de la pila. Puede ser ascendente (crece hacia posiciones menores de memoria) o descendente (crece hacia direcciones mayores de memoria). Esta información será comunicada con el módulo de ejecución.

Prioridad: baja.

3.3.3 Requisitos no funcionales

RNF-01. Sistema web con soporte en navegadores principales.

El sistema deberá ser un sistema web que se ejecute completamente en el navegador del usuario. Deberá ser funcional en los navegadores principales: Google Chrome, Mozilla Firefox, Microsoft Edge y Safari.

Prioridad: alta.

RNF-02. Funcional en principales sistemas operativos.

El sistema deberá ser funcional en los tres sistemas operativos principales: Windows, macOS y Linux.

Prioridad: alta.

RNF-03. Seguimiento de pautas WCAG 2.2 nivel AA

El sistema deberá seguir las pautas WCAG 2.2, al menos con conformidad AA.

Prioridad: alta.

RNF-04. Interfaz flexible.

La interfaz se ajustará dinámicamente a las resoluciones de pantalla comunes para ordenadores de escritorio y portátiles (excluyendo dispositivos móviles) y a distintas configuraciones de tamaños de texto.

Prioridad: alta.

RNF-05. Manejo eficiente de la memoria.

Se debe asegurar una visualización fluida de la memoria y del desensamblado.

Prioridad: media.

RNF-06. Interfaz bilingüe.

El sistema permitirá configurar el idioma de la interfaz entre dos opciones: español e inglés. Este cambio de idioma afectará a todos los elementos de la interfaz y a los mensajes de error o avisos. El sistema deberá estar preparado para que un desarrollador pueda añadir otros idiomas con facilidad.

Prioridad: baja.

RNF-07. Modo claro y oscuro.

El sistema permitirá elegir entre el modo claro u oscuro de la interfaz.

Prioridad: baja.

4 Desarrollo

4.1 Diseño

4.1.1 Casos de uso

Los casos de uso expresan los pasos que realizan el usuario y el sistema para llevar a cabo una acción en una situación concreta. Sirven para entender con más claridad el contexto en el que se utiliza un sistema y la comunicación entre el usuario, la interfaz y los módulos externos, aspecto esencial de este proyecto.

Se presentan a continuación diez casos de uso donde se muestra un uso completo del sistema, ligado a los requisitos establecidos en el apartado anterior.

CU-01: Cargar un archivo.

Descripción - Permite al usuario subir un archivo con extensión .ens de su ordenador. El sistema ensambla su contenido y muestra el estado inicial de la máquina simulada.

Precondiciones - El usuario posee en su ordenador un archivo con extensión .ens que contiene el código a ensamblar.

Flujo principal -

1. El usuario pulsa el botón de “Abrir archivo” y selecciona el archivo a ensamblar de su ordenador.
2. El sistema envía el contenido del archivo al servidor para su ensamblado.
3. El sistema recibe el resultado del ensamblado y el contenido inicial de la memoria.
4. La interfaz muestra el contenido de la memoria y la posición de las regiones de código y de pila en la ventana de memoria, y las instrucciones en la ventana del editor y de desensamblado.

Flujos alternativos -

1.1. Si el archivo no tiene la extensión correcta o no se puede leer, se muestra un mensaje de error en la pantalla de *logs*.

1.2. Si en el sistema se encontraba previamente un archivo abierto y editado, el sistema muestra una alerta con tres opciones: descargar el archivo, no guardarlo o cancelar la apertura del archivo nuevo.

1.2.1. El usuario elige descargar el archivo - El sistema descarga el archivo.

1.2.2. El usuario elige no guardar el archivo - Se descartan los cambios y se abre el nuevo archivo.

1.2.3. El usuario elige cancelar la apertura - Se interrumpe el proceso de abrir el archivo.

2.1. Si el sistema no es capaz de comunicarse con el servidor, muestra un mensaje de error en la ventana de *logs* comunicando que el usuario vuelva a intentar la acción.

4.1. Si el resultado del ensamblado no es correcto, el sistema imprime en la ventana de *logs* todos los errores de ensamblado recibidos.

Postcondiciones - El sistema posee el estado interno de la máquina a partir del resultado del ensamblado del archivo.

Prioridad - Alta.

Requisitos asociados - RF-02, RF-10, RF-14, RF-17, RF-24.

CU-02: Ejecutar una instrucción.

Descripción - Permite al usuario ejecutar una instrucción y observar los cambios.

Precondiciones - Se ha abierto un archivo y se ha ensamblado el código correctamente.

Flujo principal -

1. El usuario pulsa el botón de “Ejecutar paso a paso”.
2. El sistema ordena la ejecución de una sola instrucción.
3. La interfaz actualiza el contenido de la memoria, los registros, la consola y la posición del PC.

Flujos alternativos -

2.1. Si el sistema recibe una excepción, el sistema imprime un mensaje de error en la ventana de *logs*.

2.2. Si el SP invade la zona de código o se sobrescribe parte del contenido del código, el sistema imprime un mensaje de aviso en la ventana de *logs*.

2.3. Si la instrucción ejecutada pide un valor por entrada estándar, el sistema habilita la entrada de datos del tipo correspondiente en la sección de consola. El usuario introduce el dato pedido y continúa la ejecución.

Postcondiciones - El sistema muestra el resultado de la ejecución de una instrucción.

Prioridad - Alta.

Requisitos asociados – RF-4, RF-10, RF-15, RF-17, RF-21, RF-22, RF-24.

CU-03: Ejecutar en lote.

Descripción - Permite al usuario ejecutar todo el código y observar el resultado.

Precondiciones - Se ha abierto un archivo y se ha ensamblado el código correctamente.

Flujo principal -

1. El usuario pulsa el botón de “Ejecutar”.
2. El sistema ordena la ejecución en lote.
3. La interfaz actualiza el contenido de la memoria, los registros, la consola y la posición del PC.

Flujos alternativos -

1.1 Si el archivo ha sido modificado desde que se ensambló, se debe volver a ensamblar.

2.1. Si el sistema recibe una excepción, el sistema imprime un mensaje de error en la ventana de *logs* y se para la ejecución.

2.2. Si el SP invade la zona de código o se sobrescribe parte del contenido del código, el sistema imprime un mensaje de aviso en la ventana de *logs*.

2.3. Si el código ejecutado pide algún valor por entrada estándar, el sistema habilita la entrada de datos de los tipos correspondientes en la sección de consola. El usuario introduce cada dato pedido y continúa la ejecución.

2.3.1. Si el usuario introduce un dato no válido, el sistema mostrará un mensaje de error y pedirá otro dato. No continuará la ejecución hasta que se introduzca un dato válido.

2.4. Si el usuario tiene establecido un punto de ruptura, el programa ejecutará hasta el punto de ruptura en lugar de hasta el final del programa.

Postcondiciones - El sistema muestra el resultado de la ejecución en lote.

Prioridad - Alta.

Requisitos asociados - RF-4, RF-10, RF-15, RF-17, RF-21, RF-22, RF-24.

CU-04: Editar un archivo y guardar los cambios.

Descripción - Permite al usuario editar el archivo abierto y descargar el archivo.

Precondiciones - Se ha abierto un archivo.

Flujo principal -

1. El usuario pulsa el botón de “Editar”.
2. La interfaz intercambia la ventana de instrucciones desensambladas por la ventana del editor.
3. El usuario modifica libremente el código del archivo.
4. El usuario abre el submenú “Archivo” y pulsa el botón de “Descargar”.

Flujos alternativos -

3.1. El usuario puede tomar acciones alternativas como deshacer y rehacer los cambios, copiar y pegar, etc., mediante atajos de teclado o a través del menú flotante que aparece al hacer clic derecho sobre el editor.

Postcondiciones - El archivo es modificado.

Prioridad - Media.

Requisitos asociados - RF-06, RF-08.

CU-05: Buscar, visualizar y editar una dirección de memoria.

Descripción - Permite al usuario navegar al contenido de una dirección de memoria concreta, visualizar y editar su contenido.

Precondiciones - Se ha abierto un archivo y se ha ensamblado el código correctamente.

Flujo principal -

1. El usuario escribe una dirección, en decimal o en hexadecimal con el prefijo ‘0x’, y pulsa el botón de “Ir a dirección”.
2. La interfaz actualiza la tabla de la memoria para mostrar esa dirección y su contenido.
3. El usuario pulsa sobre el contenido de la celda.
4. La interfaz habilita la edición de esa dirección.
5. El usuario escribe el nuevo valor y pulsa ‘Enter’ o pulsa en otro lado de la pantalla. Se aceptan valores en decimal y en hexadecimal con el prefijo

‘0x’ si el formato seleccionado es decimal o hexadecimal, o un carácter imprimible si el formato seleccionado es ASCII.

6. El sistema actualiza el valor.

Flujos alternativos -

1.1. El usuario puede navegar a la dirección de maneras alternativas: a través de los botones de ir a un registro o a una etiqueta o a través de la barra de desplazamiento de la tabla de memoria.

2.1. El usuario puede elegir cambiar el formato del contenido a través de las opciones de ‘Formato’ situadas en la ventana de memoria o a través del menú principal.

5.1. Si el nuevo valor no es posible, al pulsar ‘Enter’ el valor no se actualiza y se vuelve a mostrar el valor anterior. Además, en la ventana de *logs* se muestra un mensaje de error con el valor introducido.

Postcondiciones - Se actualiza el valor de la memoria.

Prioridad - Media.

Requisitos asociados - RF-10, RF-11, RF-12.

CU-06: Editar el valor de un registro.

Descripción - Permite al usuario visualizar y editar el contenido de un registro.

Precondiciones - Se ha abierto un archivo y se ha ensamblado el código correctamente.

Flujo principal -

1. El usuario pulsa sobre el contenido del registro.
2. La interfaz habilita la edición de ese registro.
3. El usuario escribe el nuevo valor y pulsa ‘Enter’ o pulsa en otro lado de la pantalla. Se aceptan valores en decimal o en hexadecimal con el prefijo ‘0x’.
4. El sistema actualiza el valor.

Flujos alternativos -

3.1. Si el nuevo valor no es posible, al pulsar ‘Enter’ el valor no se actualiza y se vuelve a mostrar el valor anterior. Además, en la ventana de *logs* se muestra un mensaje de error con el valor introducido.

Postcondiciones - Se actualiza el valor del registro.

Prioridad - Media.

Requisitos asociados – RF-15, RF-16.

CU-07: Reiniciar los valores de la memoria y de los registros.

Descripción - Permite restaurar los valores iniciales de la memoria y de los registros, tal y como estaban tras el ensamblado.

Precondiciones - Se ha abierto un archivo y se ha ensamblado el código correctamente.

Flujo principal -

1. El usuario pulsa el botón de ‘Reiniciar’.

2. El sistema actualiza la memoria con el valor que tenía tras el ensamblado e inicializa todos los registros y biestables a 0.
3. La interfaz actualiza la memoria y los registros.

Flujos alternativos - N/A.

Postcondiciones - Se restauran los valores iniciales de la memoria y de los registros.

Prioridad - Media.

Requisitos asociados – RF-05.

CU-08: Modificar una instrucción desensamblada.

Descripción - Permite al usuario modificar una instrucción sin volver a ensamblar todo el código, siempre y cuando la nueva instrucción tenga el mismo tamaño en memoria que la anterior.

Precondiciones - Se ha abierto un archivo y se ha ensamblado el código correctamente.

Flujo principal -

1. El usuario pulsa dos veces en una instrucción en la sección de instrucciones desensambladas.
2. La interfaz habilita la edición de esa instrucción.
3. El usuario escribe la nueva instrucción y pulsa 'Enter' o pulsa en otro lado de la pantalla.
4. El sistema actualiza el valor en memoria.

Flujos alternativos -

3.1. Si la nueva instrucción no es válida, la interfaz muestra un mensaje de error y no se actualiza su contenido.

Postcondiciones - Se cambia el valor de la instrucción en memoria.

Prioridad - Baja.

Requisitos asociados – RF-17, RF-18.

CU-09: Configurar el entorno.

Descripción - Permite configurar el entorno: la dirección de la pila, el formato de las instrucciones (hexadecimal o decimal), el formato del contenido (hexadecimal o decimal) y el modo de depuración.

Precondiciones - N/A.

Flujo principal -

1. El usuario pulsa el submenú de configuración.
2. La interfaz despliega todas las opciones de configuración.
3. El usuario se sitúa sobre una de las opciones.
4. La interfaz despliega las posibilidades de esa opción.
5. El usuario elige una de las posibilidades.
6. El sistema actualiza la configuración y sus cambios consecuentes.

Flujos alternativos -

6.1. La opción de modificar la dirección de la pila solo estará disponible cuando no esté siendo ejecutado un programa. En ese caso, la interfaz actualizará el SP y la región de la pila.

6.2. Si el usuario elige cambiar el formato de las instrucciones o del contenido, se actualiza en la sección de memoria, de instrucciones desensambladas y de registros. No se produce ningún cambio en el contenido interno de estas regiones, sólo en su presentación.

Postcondiciones - Cambia la configuración del entorno.

Prioridad - Media.

Requisitos asociados – RF-26, RF-27, RF-28.

CU-10: Configurar el tema y el idioma.

Descripción - Permite cambiar el tema de la interfaz (modo claro u oscuro) y el idioma (español o inglés).

Precondiciones - N/A.

Flujo principal -

1. El usuario pulsa el botón del tema, que alterna entre tema claro y oscuro.
2. El sistema actualiza todos los estilos de la interfaz.
3. El usuario pulsa el botón de idioma, que alterna entre idioma español e inglés.
4. El sistema actualiza la configuración del idioma y la interfaz cambia el idioma de todos los elementos de la pantalla.

Flujos alternativos - N/A.

Postcondiciones - Se cambia el tema e idioma de la interfaz.

Prioridad - Baja.

Requisitos asociados – RNF-06, RNF-07.

4.1.2 Diagrama de secuencia

Para aclarar la comunicación entre el usuario y los tres módulos del sistema global, se proporciona un diagrama de uso de los dos casos de uso más complejos y prioritarios del sistema: subir un archivo (CU-01) y ejecutar en lote (CU-03). Además, el diagrama de secuencia del CU-03 es igual que el de ejecutar una sola instrucción (CU-02). Estos diagramas se muestran en la Ilustración 6.

Para entender plenamente el diagrama se debe explicar en más detalle la relación entre el módulo de ejecución y la interfaz. Ambos módulos comparten el acceso al estado interno de la máquina (memoria, registros, biestables y salida). La memoria se gestiona mediante una referencia compartida creada inicialmente por la interfaz, mientras que es el módulo de ejecución el que mantiene los datos de los registros y la consola y permite a la interfaz acceder a ellos cuando es necesario. Sin embargo, aunque el estado sea compartido, en el diagrama de secuencia se ha indicado explícitamente el envío de estos datos para representar el flujo de la información.

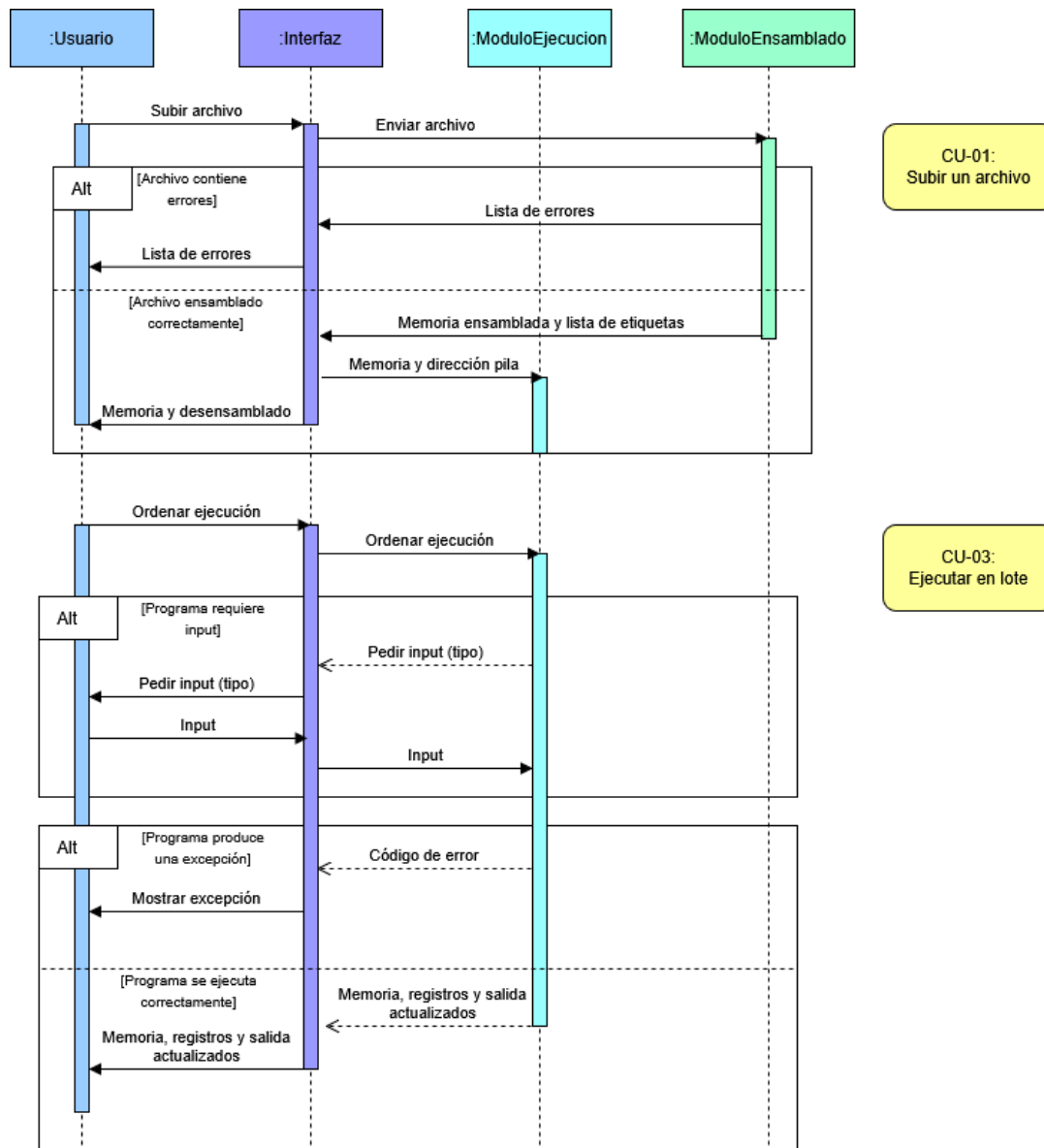


Ilustración 6. Diagrama de secuencia de CU-01 y CU-03.

4.1.3 Diseño de la interfaz

En este proyecto, una de las fases de diseño más importantes es el diseño de la interfaz. Este proceso incluye la decisión de la disposición de los elementos, la apariencia de cada uno, la gama de colores y la definición de las interacciones entre elementos.

El prototipo resultante de este diseño es utilizado en las pruebas de usabilidad para recibir retroalimentación antes de comenzar la codificación. Posteriormente, sirve de guía al desarrollador, que debe implementar la interfaz tan cercana al diseño obtenido como sea posible.

El diseño de la interfaz se ha realizado en dos etapas: la creación de un diseño de baja fidelidad y un diseño de alta fidelidad. Ambos prototipos se han creado utilizando la herramienta Figma.

4.1.3.1 Diseño de baja fidelidad

Los prototipos de baja fidelidad son réplicas simples del sistema, sin muchos detalles, que se puedan iterar rápidamente hasta acercarse a un diseño inicial

satisfactorio. Se pueden hacer en papel o en herramientas de diseño más sofisticadas. En este caso, se decidió hacer en Figma para poder compartirlo fácilmente y para que sirva de base para el diseño de alta fidelidad.

En este diseño, el reto principal era encontrar la mejor disposición de los elementos, ya que una misma pantalla debe poder mostrar las seis ventanas de la aplicación: el desensamblado, el editor, la ventana de *logs*, la consola, los registros y la memoria. Como tanto el desensamblado como el editor muestran las instrucciones del programa, estas ventanas no se presentan a la vez, sino que se pueden alternar entre ellas.

En la Ilustración 7 se observa la disposición decidida. A la izquierda, se han colocado los elementos relacionados con el control de la ejecución: una barra de botones de acceso rápido para las opciones más importantes (abrir un archivo, ejecutar en lote, ejecutar una instrucción...), la ventana de desensamblado, donde se establecen los puntos de ruptura y se muestra la instrucción actual de la ejecución; y la ventana de *logs*. A la derecha se encuentran las ventanas relacionadas con el estado interno de la máquina simulada: la consola, los registros y la memoria. La consola se ha colocado en la parte central de la pantalla porque es con la que más interactúan los usuarios, al ser el módulo de entrada y salida. La configuración del idioma y del tema se pueden realizar a través de los botones del encabezado. Además, todas las acciones y opciones de configuración se pueden realizar a través del menú.

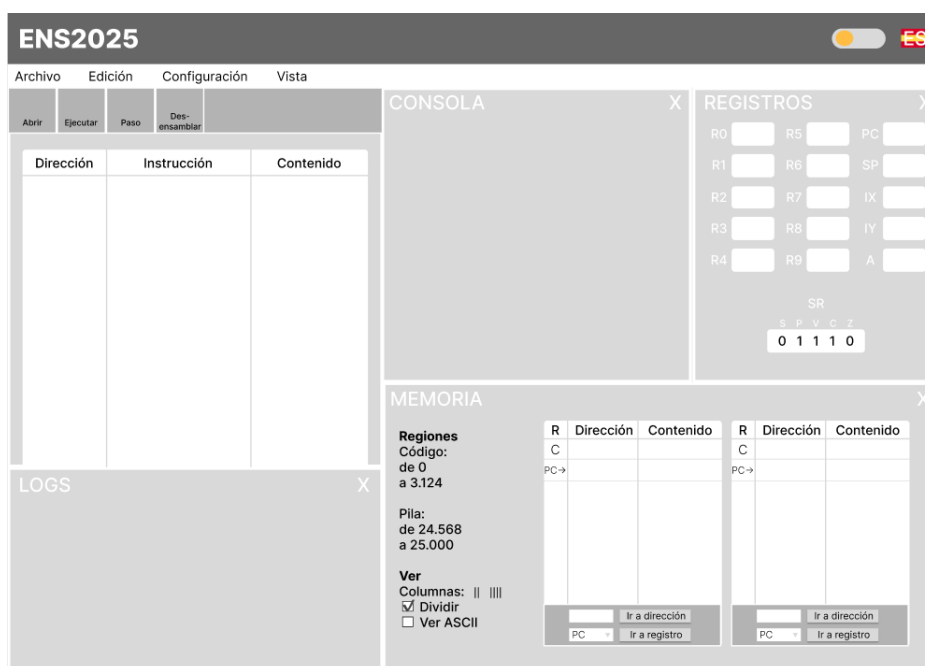


Ilustración 7. Diseño de baja fidelidad.

4.1.3.2 Diseño de alta fidelidad

En el diseño de alta fidelidad se pulen la interfaz y se añaden todos los detalles omitidos en el primer diseño, con el objetivo de acercarse lo máximo posible al producto real. Además, se añaden las interacciones entre componentes y entre pantallas.

En el caso de la interfaz del proyecto, los cambios más notables entre ambos prototipos fueron el ajuste de los colores, el diseño de todos los iconos y el relleno de todas las tablas con datos reales de programas. Además, se incluyó la opción de cambiar el formato (entre decimal, hexadecimal y ASCII) desde la

ventana de la memoria, para acceder de forma más directa en lugar de abrir la configuración. Las opciones presentes en la tabla de memoria también sufrieron modificaciones: en lugar de tener un botón para cada opción de navegación (ir a una dirección, al valor de un registro o al valor de una etiqueta) se centralizó todo, con un primer desplegable en el que se puede elegir el tipo de navegación. El diseño de alta fidelidad resultante se puede observar en la Ilustración 8.



Ilustración 8. Diseño de alta fidelidad.

El siguiente paso es la definición de interacciones de la interfaz. Como este prototipo es el utilizado para la evaluación de usabilidad, es necesario crear todas las pantallas necesarias para los escenarios y definir sus conexiones. Este proceso se puede realizar a través del modo de prototipado de Figma, indicando cuatro factores de la interacción: el elemento que desencadena la acción, el tipo de evento, el resultado de la acción, y, si aplica, el destino de la acción (por ejemplo, la pantalla a la que se cambia). En la Ilustración 9 se muestra la definición de una interacción entre la pantalla principal y la pantalla de subida de archivos, activada cuando se selecciona el botón de “Subir Archivo”.

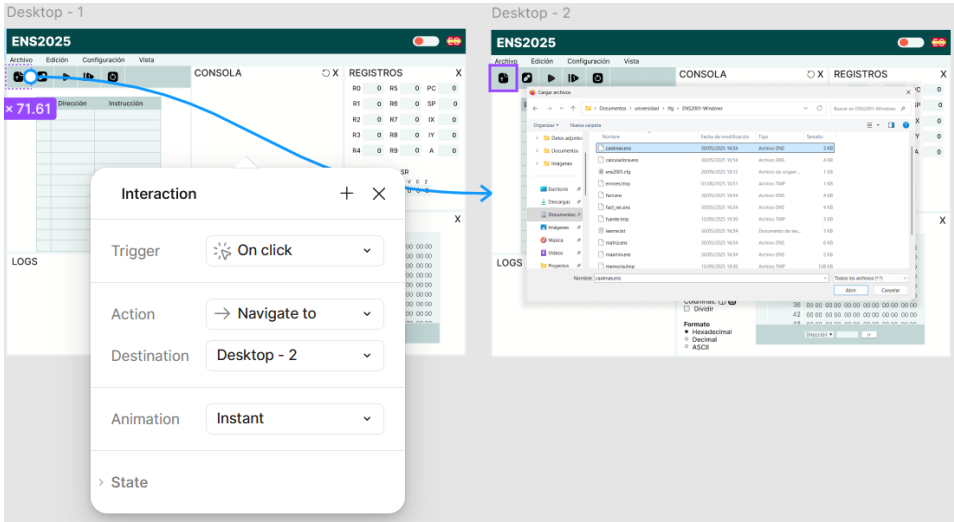


Ilustración 9. Interacción de cambio de pantalla en Figma.

Asimismo, existen conexiones entre elementos de la interfaz que no suponen un cambio de pantalla y se pueden realizar en cualquier momento, como la alternancia entre la ventana de desensamblado y el editor. Para estas conexiones se han utilizado componentes reutilizables de Figma. De esta forma, se declara en el componente su comportamiento una sola vez y se comporta de esa forma en todas las pantallas donde se utilice, ahorrando muchas interacciones adicionales. Además, si se desea modificar algún aspecto del componente, se puede modificar en su declaración y el cambio se ve automáticamente reflejado en todas sus instancias. Un ejemplo de componente es la ventana de memoria mostrada en la Ilustración 10, donde se definieron las opciones de vista de la tabla de la memoria.



Ilustración 10. Conexiones en el componente de la ventana de la memoria.

4.1.4 Test de usabilidad

Los test de usabilidad son una serie de pruebas que se realizan con usuarios representativos del público objetivo de la aplicación con el objetivo de analizar su experiencia con el sistema. Para ello, se diseñan unas tareas a realizar con el prototipo y se pide a los usuarios que interactúen con la aplicación o prototipo para completarlas, mientras se monitorea su reacción y sus acciones. Estas pruebas son esenciales para encontrar problemas de usabilidad durante la fase de diseño, mientras es sencillo corregirlos.

4.1.4.1 Planteamiento

El objetivo era la realización de 8 test de usabilidad presenciales con alumnos de la universidad, tratando que aproximadamente la mitad haya usado con anterioridad el ENS2001. Previamente a su realización, se hizo un proceso de planificación de las pruebas. Esta planificación incluye la redacción de los siguientes apartados:

- Preguntas iniciales. Sirven para identificar el perfil que tiene el usuario, que puede influenciar sus resultados.

- Texto introductorio. Se lee al usuario antes de la realización de las pruebas para explicar el objetivo de la evaluación, cómo va a funcionar y qué debe hacer.
- Lista de tareas. Aquí se redactan las tareas que deben realizar los usuarios a través del prototipo. Cada tarea consta de una situación inicial y un conjunto detallado de instrucciones. En este caso, las situaciones iniciales se basan en escenarios con los que se puede encontrar un alumno de Traductores de Lenguajes.
- Cuestionario de observación de las tareas. Este cuestionario debe ser rellenado por el organizador del test mientras el usuario realiza las pruebas. En él, se apuntan las acciones, los errores cometidos y la reacción del usuario.
- Cuestionario de impresiones generales. Se hacen preguntas abiertas para valorar la primera impresión del usuario.
- Cuestionario SUS (*System Usability Scale*) [83]. Es un test estandarizado para valorar la satisfacción del usuario según su nivel de aprobación de ciertos enunciados.
- Cuestionario UEQ (*User Experience Questionnaire*) [84]. Este es un cuestionario muy utilizado para valorar la impresión subjetiva del usuario. Mide esto a través de seis criterios: atractivo, eficiencia, claridad, fiabilidad, novedad y estimulación.

A continuación, se presenta la plantilla para estos apartados utilizada en todas las pruebas.

Preguntas iniciales

- Edad del participante
- ¿Has usado un simulador de ensamblador previamente?
- ¿Has usado el simulador ENS2001 previamente?
- ¿Tienes alguna necesidad especial en la realización de estas pruebas o en el uso de una página web?

Texto introductorio

Este proyecto consiste en el diseño e implementación de un simulador de ensamblador para la asignatura de Traductores de Lenguajes. Este simulador será usado por los alumnos para depurar la práctica de la asignatura. A través de él, se puede cargar un archivo ensamblador, editarlo y ejecutarlo, además de observar o modificar la memoria y los registros o gestionar la entrada y la salida. Se pretende que sea fácil de usar e intuitivo.

Para garantizar esto, estamos realizando estas pruebas de usabilidad. Consisten en la realización de tareas en un prototipo que simula el sistema final. Para cada tarea, se te dirá una situación inicial y unas instrucciones y deberás usar el prototipo para realizarla igual que harías en una página real. Puedes hacer clic en los elementos y escribir con el teclado. A la hora de escribir, en algunos momentos el prototipo puede autocompletar toda la entrada para simplificar la prueba. El prototipo tiene implementado todo lo necesario para realizar las tareas, pero puede actuar de forma inesperada. Si en algún momento ocurre esto, te informaré de cómo proceder.

Recuerda que esta es una evaluación del diseño, y cualquier error cometido es un fallo del sistema y no del usuario. Observando las dificultades y las dudas de los usuarios podremos mejorar el sistema para que no ocurran en el futuro.

Para evaluar mejor el prototipo, agradecería que pensases en voz alta las dudas que surgen o las acciones que esperabas que sucediesen en la realización de las tareas. No podré resolver dudas acerca de la solución de la tarea, pero puedes preguntar lo que desees acerca de las instrucciones o la prueba en sí. Cuando termine la prueba, haremos unos cuestionarios para capturar tu primera impresión del sistema. No repares en hacer críticas constructivas al sistema.

Agradezco mucho tu participación.

Tareas

Tarea 1. Abrir archivo y ejecutar en lote

Situación de partida:

Antes de empezar la práctica de Traductores de Lenguajes decides escribir un programa básico en el lenguaje ensamblador de la asignatura para probar su sintaxis y el uso del simulador.

Instrucciones:

Abre en el simulador el archivo llamado ‘cadenas.ens’.

Ejecuta el programa en lote.

Si en algún momento el programa te pide introducir un *input*, introduce la palabra ‘mundo’.

Observa el resultado de la ejecución.

Tarea 2. Depuración y edición de la memoria

Situación de partida:

Tu compañero de la práctica no entiende el funcionamiento del direccionamiento directo a memoria. Para explicárselo, decides depurar en el simulador el programa en el que estabais trabajando, ‘multiplicacion.ens’, y mostrarle los resultados.

Instrucciones:

En el simulador ya se encuentra abierto el programa ‘multiplicacion.ens’.

Coloca un punto de ruptura (*breakpoint*) en la instrucción ‘WRINT / 1000’.

Ejecuta hasta ese punto.

Cuando se interrumpa la ejecución, modifica la dirección de memoria 1000 del valor actual al valor 5.

Ejecuta hasta el final y observa el resultado de la ejecución.

Tarea 3. Depuración y edición de un registro

Situación de partida:

Tus compañeros y tú no estáis seguros de que el programa funcione siempre de manera correcta. Tu compañero te sugiere comprobar si en este programa el valor inicial de un registro afecta a su valor posterior.

Instrucciones:

En el simulador ya se encuentra abierto el programa ‘multiplicacion.ens’.

Cambia el valor del registro acumulador (A) de 0 a 4.

Ejecuta el programa instrucción a instrucción hasta que cambie el valor del registro.

Tarea 4. Edición de una instrucción

Situación de partida:

Has avanzado con la práctica hasta la creación de la zona de variables globales. Quieres comprobar su correcto funcionamiento imprimiendo el primer número que aparece en esta zona de memoria, que comienza en la dirección 500, y comparando este resultado con el resultado normal de la ejecución.

Instrucciones:

En el simulador ya se encuentra abierto el programa 'multiplicacion.ens'.

Ejecuta el programa en lote y observa el resultado de la ejecución.

Reinicia el programa.

Modifica la instrucción 'WRINT /1000' a 'WRINT /500'.

Vuelve a ejecutar en lote.

Observa el nuevo resultado de la ejecución.

Observación de las tareas

Nº usuario	
Nº tarea	
¿Ha completado la tarea con éxito?	
Número de errores	
¿Dónde ha cometido los errores?	
Lugares donde ha dudado	
Número de acciones	
Tiempo	
Comentarios del usuario	
Observaciones	

Cuestionario impresiones generales

- ¿Con qué facilidad has podido realizar las tareas?
- ¿Ha habido algún punto durante la realización de las tareas en el que no supieras cómo continuar?
- ¿Qué parte del prototipo te ha resultado más confusa? ¿Por qué?
- ¿Qué es lo que más te ha gustado del prototipo? ¿Por qué?
- ¿Cambiarías o añadirías algo a la interfaz?
- ¿Quieres añadir alguna observación o comentario?

Cuestionario SUS

Rellena el siguiente cuestionario. Para cada frase, marca con una X la casilla que consideres adecuada, siendo 5 "estoy muy de acuerdo" y 1 "no estoy nada de acuerdo".

	1	2	3	4	5
Creo que me gustaría usar este sistema con frecuencia					
El sistema es innecesariamente complejo					
Me pareció que el sistema era muy fácil de usar					
Creo que voy a necesitar ayuda de un técnico para poder utilizar el sistema					
Encontré diversas funciones en el sistema que están bien integradas					
Pienso que hay demasiada inconsistencia en este sistema					
Me imagino que la mayoría de la gente aprenderá a utilizar este sistema rápido					
Me pareció un sistema muy complicado de usar					
Me sentí muy seguro con el sistema					
Necesito aprender muchas cosas antes de poder usar bien el sistema					

Cuestionario UEQ

Para evaluar el producto, rellena el siguiente cuestionario. El cuestionario consta de pares de atributos contrastantes que pueden aplicarse al producto. Los números entre los atributos representan escalas entre los opuestos. Puedes expresar tu acuerdo con los atributos marcando con una X la casilla que mejor refleje tu impresión. Decide de forma espontánea. No pienses demasiado en tu decisión para asegurarte de transmitir tu impresión original.

	1	2	3	4	5	6	7		
Obstrutivo								Impulsor de apoyo	1
Complicado								Fácil	2
Ineficiente								Eficiente	3
Confuso								Claro	4
Aburrido								Emocionante	5
Ordenado								Sobrecargado	6
Convencional								Novedoso	7
Incómodo								Cómodo	8
Impredecible								Predecible	9
Cubre expectativas								No cubre expectativas	10
Atractivo								Feo	11

4.1.4.2 Resultados

Se han realizado finalmente 7 test de usabilidad presenciales. Tras recopilar estos datos, se han analizado las respuestas a cada sección del cuestionario, con el objetivo de recopilar una lista definitiva de problemas de usabilidad encontrados.

4.1.4.2.1 Análisis del perfil de los usuarios

Se ha realizado el test de usabilidad con 7 usuarios externos. Todos los usuarios pertenecen al grupo demográfico que por lo general va a utilizar este simulador en la asignatura de Traductores de Lenguajes. Se trata de personas jóvenes de 20 o 21 años con experiencia en el uso de simuladores de ensamblador.

Traductores de Lenguajes es una asignatura de tercer curso, por lo que los alumnos que la atienden ya han cursado otras asignaturas como Estructura de Computadores, donde se explica en detalle el funcionamiento de ensamblador y se realiza una práctica utilizando un simulador. Este es el motivo por el que, aunque se pretende la creación de un sistema muy simple, se presupone cierto conocimiento de ensamblador para utilizarlo.

Respecto al uso previo de los usuarios específicamente del ENS2001, se ha elegido a 3 usuarios que lo han utilizado previamente y 4 que no, como se representa en la Ilustración 11. Aunque el público objetivo de esta interfaz no va a haber usado ENS2001 antes, se quería tener en cuenta la opinión de aquellos que sí que lo han utilizado para averiguar si ven alguna ventaja o desventaja importante de ENS2025 respecto a su predecesor.

Ningún usuario entrevistado tiene ninguna necesidad especial respecto al uso de páginas web. Aunque habría sido interesante la inclusión de usuarios con diferentes tipos de necesidades, no ha sido posible debido a la falta de contactos con este perfil y al tiempo limitado disponible para la realización de las pruebas, que se plantearon como un test exploratorio de pequeña escala.

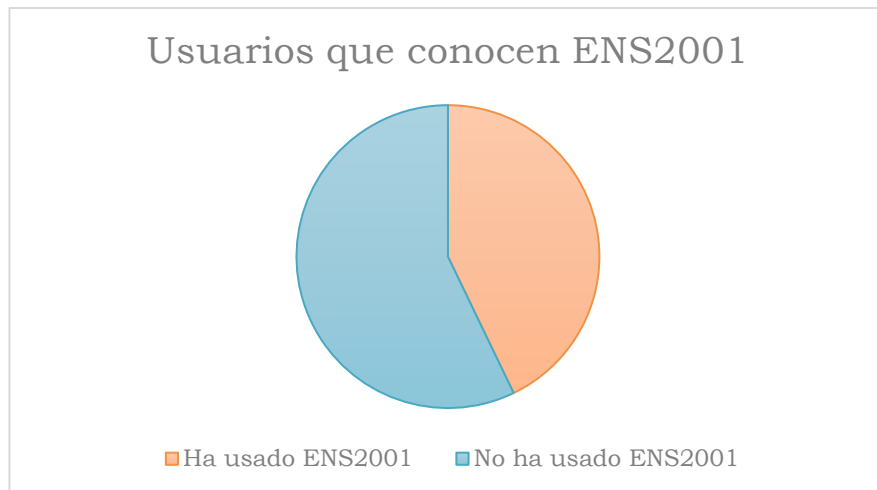


Ilustración 11. Experiencia con el ENS2001 de los participantes.

4.1.4.2.2 Análisis de efectividad

En la Tabla 2 se presentan los resultados de efectividad de la interfaz, que se refiere al número de errores y al porcentaje de compleción de las tareas.

	Media de errores	Porcentaje de éxito
Tarea 1	0,57	100%
Tarea 2	0,71	100%
Tarea 3	0,43	100%
Tarea 4	0,00	100%

Tabla 2. Media de errores y porcentaje de éxito por tarea.

A continuación, se presenta en forma de gráfico el número de errores cometidos por cada usuario en cada tarea.

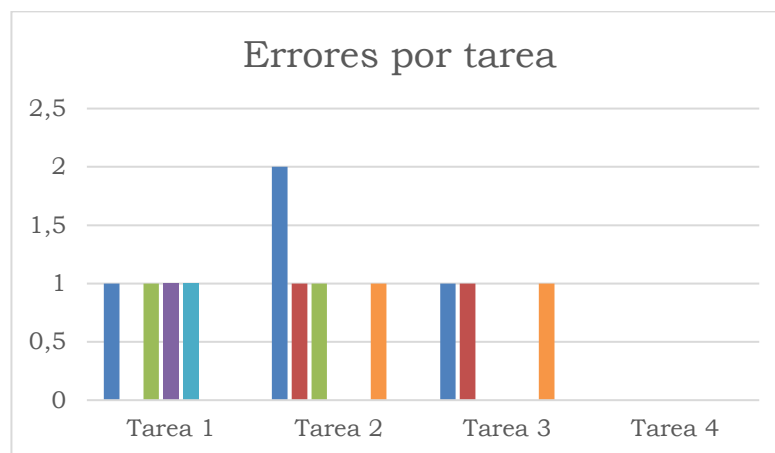


Ilustración 12. Número de errores por tarea.

Se puede hacer un cierto análisis a partir de esta información. Para comenzar, la mayor cantidad de errores se han cometido en las dos primeras tareas. Esto es razonable, puesto que los usuarios se están acostumbrando al prototipo y tienen más posibilidades de equivocarse. Además, la tarea 2 es la tarea más larga, lo que también explica su mayor tasa de errores. Por otro lado, ningún usuario cometió ningún error en la tarea 4, porque era una tarea sencilla donde solo se introducía un nuevo botón y el resto de las acciones ya eran conocidas de tareas anteriores.

Lugares donde se han cometido los errores

- Tarea 1: el error principal con diferencia, sufrido por 4 de los 7 participantes, es no darse cuenta de que el programa le estaba pidiendo un *input* o no saber si el programa se había ejecutado. Como consecuencia, le daban al botón de ejecución varias veces.
- Tarea 2: en esta tarea se cometen dos errores principales. El primero es pulsar la instrucción entera para poner un *breakpoint* en lugar de la casilla reservada para ello. El segundo es confusión sobre qué botón de ejecución es el correcto, puesto que algunos usuarios creen que el botón de ejecución paso a paso tiene que ver con la ejecución hasta un *breakpoint*.
- Tarea 3: los errores cometidos en esta tarea son menores, como no darse cuenta de que el valor del registro ya ha cambiado y seguir ejecutando.

4.1.4.2.3 Análisis de eficiencia

Análisis de tiempo

En la Tabla 3 se puede ver el tiempo promedio que han dedicado los usuarios a cada tarea, comparado con el tiempo mínimo. Este tiempo mínimo se ha estimado realizando las tareas a un ritmo normal, pero con total certeza de la siguiente acción a realizar. La ratio es la división del tiempo medio entre el tiempo mínimo.

	Tiempo mínimo	Tiempo medio	Ratio
Tarea 1	10	65,57	6,56
Tarea 2	14	65,57	4,68
Tarea 3	7	29,43	4,20
Tarea 4	10	18,00	1,80

Tabla 3. Resultados de tiempo de los participantes.

En la Ilustración 13 se puede ver el tiempo empleado por cada usuario en cada tarea y la línea base que indica el tiempo mínimo.

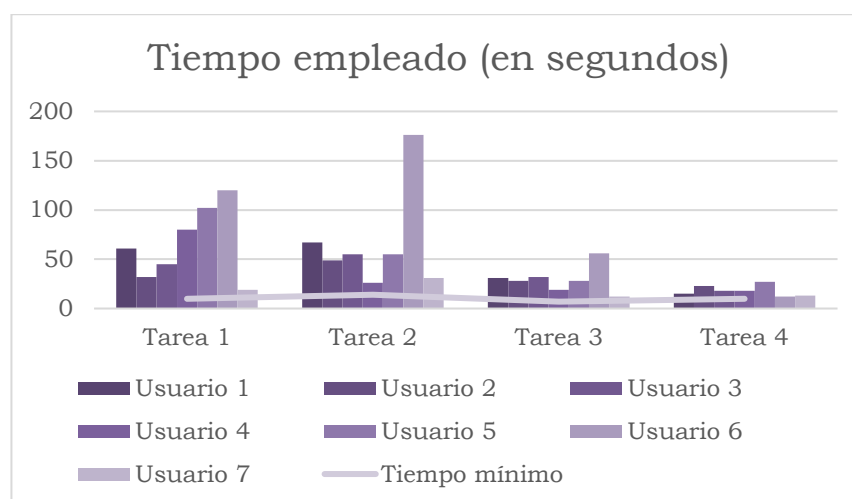


Ilustración 13. Tiempo empleado por cada participante.

Análisis de acciones

La Tabla 4 es una tabla equivalente a la anterior con el número de acciones medias realizadas por los usuarios. Una acción se considera cualquier otra interacción con el ordenador más allá de mover el ratón, como hacer clic o pulsar una tecla.

	Acciones mínimas	Acciones medias	Ratio
Tarea 1	9	12,00	1,33
Tarea 2	8	12,29	1,54
Tarea 3	5	5,71	1,14
Tarea 4	6	6,14	1,02

Tabla 4. Resultado de acciones de los participantes.

La Ilustración 14 es un gráfico que muestra el número de acciones por usuario en cada tarea, también comparado con el número mínimo de acciones.

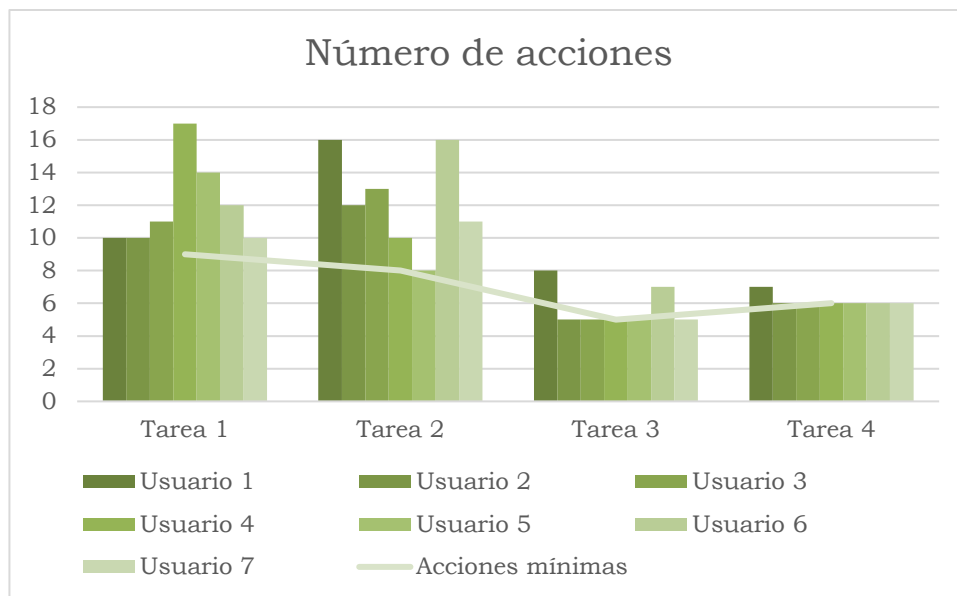


Ilustración 14. Número de acciones por cada participante.

Hallazgos respecto a la eficiencia

En general, el tiempo empleado en las tareas ha ido disminuyendo según avanzaba la prueba y los usuarios se acostumbraban a la interfaz. Esta tendencia se observa sobre todo en los ratios, que disminuyen de forma continua. La mayoría de los picos de tiempo se deben a los errores comentados en el apartado de efectividad, que confundían a los usuarios y retrasaban la compleción de la tarea.

Respecto a las acciones, el número de acciones de los usuarios se aproxima considerablemente al número mínimo. Muchas de las acciones adicionales se deben a una falta de confianza sobre si se ha realizado una acción, que los llevaba a pulsar el mismo botón o botones alternativos varias veces.

Dado este análisis, se considera que el tiempo y las acciones son congruentes con los problemas de usabilidad encontrados en la sección de efectividad, y no indicativos de ningún problema adicional de eficiencia.

4.1.4.2.4 Cuestionario SUS

La puntuación del SUS es de un 79,65. Esto es un resultado satisfactorio, ya que se considera un buen resultado cualquier valor superior a 68, y excelente cualquier valor superior a 80,3 [83]. En la Tabla 5 se presentan los resultados por pregunta del cuestionario. En cada casilla del 1 al 5, se muestra el número de usuarios que han respondido con ese número a esa pregunta. Como recordatorio, un 5 significa conformidad con el enunciado, y un 1 significa total discrepancia.

	1	2	3	4	5
Creo que me gustaría usar este sistema con frecuencia		1	1	4	1
El sistema es innecesariamente complejo	4	3			
Me pareció que el sistema era muy fácil de usar			1	2	4
Creo que voy a necesitar ayuda de un técnico para poder utilizar el sistema	6	1			
Encontré diversas funciones en el sistema que están bien integradas			1	5	1
Pienso que hay demasiada inconsistencia en este sistema	6	1			
Me imagino que la mayoría de la gente aprenderá a utilizar este sistema rápido			2	2	3
Me pareció un sistema muy complicado de usar	2	4	1		
Me sentí muy seguro con el sistema		1	2	3	1
Necesito aprender muchas cosas antes de poder usar bien el sistema	2	4	1		

Tabla 5. Resultados por pregunta del cuestionario SUS.

Hallazgos del cuestionario SUS

- Por lo general, los usuarios opinan que este es un sistema fácil de usar y consistente.
- En la pregunta de “*Me gustaría usar este sistema con frecuencia*”, hay cierta indecisión porque este no es un sistema de uso frecuente. Muchos de los usuarios no manejan frecuentemente simuladores de ensamblador, ni siquiera en el contexto de la práctica de Traductores de Lenguajes.
- Se han obtenido respuestas más bajas de lo deseado en la pregunta “*Me sentí muy seguro con el sistema*”. Esta falta de seguridad puede provenir de la confusión de algunos botones y sobre la falta de confirmación de las acciones del usuario. En bastantes ocasiones, los usuarios no estaban seguros de haber realizado o no una acción. También se ha comentado la necesidad de añadir texto a los botones para aclarar su uso.
- Respecto a las preguntas de “*Me imagino que la mayoría de la gente aprenderá a utilizar este sistema rápido*” y “*Necesito aprender muchas cosas antes de poder usar bien el sistema*”, las respuestas son más neutras porque el sistema no es de uso general, sino para gente con conocimientos de ensamblador. Muchos usuarios mencionaron esto al rellenar estas preguntas.

4.1.4.2.5 Cuestionario UEQ

Los resultados se han calculado utilizando el Excel que se ofrece en la página oficial del UEQ [84] y siguiendo sus pautas para analizar los resultados.

Criterios UEQ (Media y varianza)		
Atracción	1,929	0,54
Transparencia	2,000	1,08
Eficiencia	1,786	0,57
Controlabilidad	1,810	0,51
Estimulación	0,571	1,62
Novedad	0,143	3,81

Tabla 6. Puntuación de cada escala del UEQ.

A partir de las respuestas de los usuarios, se calcula un valor promedio en cada uno de los seis criterios que mide este formulario. Estos valores se pueden ver en la Tabla 6. Cada criterio se mide con un rango de -3 a 3, aunque dada la subjetividad de la prueba y la tendencia a la neutralidad de algunos usuarios,

es difícil encontrar valores superiores a 2 o menores que -2. Los resultados entre -0,8 y 0,8 se consideran neutrales, mientras que un resultado menor a -0,8 sería negativo y uno superior a 0,8 sería positivo. En la Ilustración 15 se muestran estos mismos valores de forma más visual.

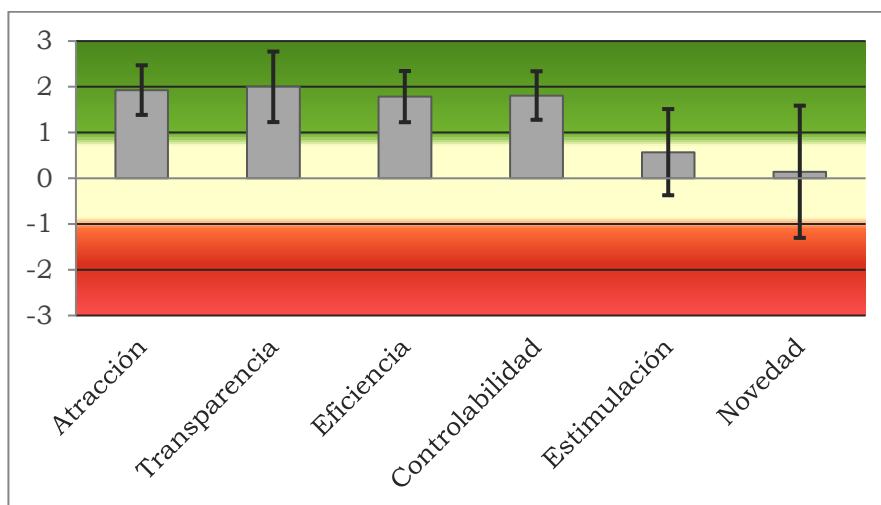


Ilustración 15. Diagrama de cajas de cada escala del UEQ.

Hallazgos del UEQ

Las primeras cuatro categorías han obtenido resultados muy satisfactorios. En cambio, estimulación y novedad tienen resultados más neutrales; sobre todo, la novedad ha resultado ser una categoría controvertida, como indica su elevada varianza.

Los criterios se suelen agrupar en dos conjuntos: la “calidad pragmática”, formada por las categorías de transparencia, eficiencia y controlabilidad, y la “calidad hedónica”, formada por estimulación y novedad. La “calidad hedónica” se refiere a los aspectos no estrictamente relacionados con la realización de tareas y coincide con las categorías que han obtenido peores resultados. Dada la naturaleza del proyecto, un simulador de ensamblador para uso académico, es coherente con que no se considere una interfaz particularmente estimulante o novedosa y obtenga resultados neutrales.

4.1.4.2.6 Cuestionario de impresiones generales

Se realiza a continuación un análisis pregunta a pregunta del cuestionario de impresiones generales.

1. ¿Con qué facilidad has podido realizar las tareas?
Los usuarios respondieron que las tareas les han parecido fácil (5/7 usuarios) o de dificultad media (2/7 usuarios).
2. ¿Ha habido algún punto durante la realización de las tareas en el que no supieras cómo continuar?
El mayor punto de confusión ha sido tras la ejecución en la primera tarea, porque no sabían que tenían que introducir la entrada y por ello no sabían si lo habían ejecutado. La segunda tarea también ha supuesto problemas a aquellos que se equivocaron de botón de ejecución y tampoco sabían si lo habían ejecutado.
3. ¿Qué parte del prototipo te ha resultado más confusa? ¿Por qué?
La respuesta más común se refería a los botones de ejecución, al ser demasiado parecidos. También se ha respondido que la entrada o la modificación de los registros (al no ser común en otros simuladores).

4. ¿Qué es lo que más te ha gustado del prototipo? ¿Por qué?
Tres usuarios han contestado que les gusta la distribución de pantallas, al poder ver toda la información a la vez. Otros usuarios comentan que la pantalla que más les gusta es la de desensamblado, por ser familiar y visualmente atractiva.
5. ¿Cambiarías o añadirías algo a la interfaz?
Se muestra una lista con todas las sugerencias de los usuarios:
- Modificar el botón de instrucción a instrucción.
 - Añadir instrucciones sobre cómo usar el sistema.
 - Añadir texto cuando pasas por encima de los botones.
 - Indicación sobre si hay que meter texto en la pantalla de la consola.
 - Añadir un botón de volver hacia atrás una instrucción.
 - Cambiar la etiqueta BR (de *breakpoint*) a BRK para que no se confunda con *branch*.
 - Añadir un botón para ir a una etiqueta o a una dirección en el desensamblado.
 - Ver contenido real en desensamblado.
 - Opciones para personalizar el fondo o la fuente del texto.
 - Opción de ponerlo en inglés.
6. ¿Quieres añadir alguna observación o comentario?
Todos los comentarios eran más sugerencias para el sistema que ya se han incluido en la respuesta anterior.

4.1.4.2.7 Resumen de problemas de usabilidad

Problemas y posibles soluciones

En la Tabla 7 se presentan los problemas de usabilidad de la interfaz, junto a su severidad y una posible solución a cada uno.

Problema	Severidad	Solución
Botón de ejecución instrucción a instrucción confuso	Alta	Cambiar a un botón triangular con un 1 en el centro. Añadir en todos los botones texto con su título al pasar el ratón por encima.
Poca visibilidad de la entrada	Alta	Dos soluciones conjuntas: 1. Marcar con rojo la instrucción de input en la que está esperando el programa. 2. Crear una caja de entrada en la consola con un texto encima que pida al usuario la entrada (ej.: "introduzca un entero").
Confusión sobre el estado de la ejecución	Alta	Marcar con rojo la instrucción a la que apunta el PC al final de la ejecución.

Tabla 7. Problemas de usabilidad.

Otras mejoras

Mejora	Prioridad
Cambiar etiqueta BR a BRK	Media
Añadir botón para ir a dirección o etiqueta en la ventana de desensamblado	Alta
Permitir volver hacia atrás una instrucción	Baja
Ver contenido real en la ventana de desensamblado	Baja
Personalizar más los colores y la fuente	Baja

Tabla 8. Posibles mejoras a la interfaz.

En la Tabla 8 se muestran posibles mejoras a la interfaz que no provienen de un problema de usabilidad encontrado, pero ensalzarían la experiencia del usuario.

4.1.4.3 Mejoras aplicadas

A partir de los resultados de las pruebas de usabilidad, se ha actualizado el prototipo de alta fidelidad para reflejar las mejoras aplicadas. En la Ilustración 16 se pueden apreciar los cambios:

- El botón de ‘Ejecución de una instrucción’ ha sido rediseñado.
- El botón de ‘Abrir archivo’ muestra el aspecto que tendrían todos los botones de la barra al recibir el foco del teclado o al pasar el ratón por encima, con un texto debajo indicando su función.
- La tabla de navegación de memoria se ha replicado en la tabla de desensamblado para facilitar su navegación.
- En la ventana de la consola se muestra la caja de entrada que aparecería cuando el programa pide una entrada al usuario.
- La etiqueta de los puntos de ruptura ha sido modificada de BR a BRK.

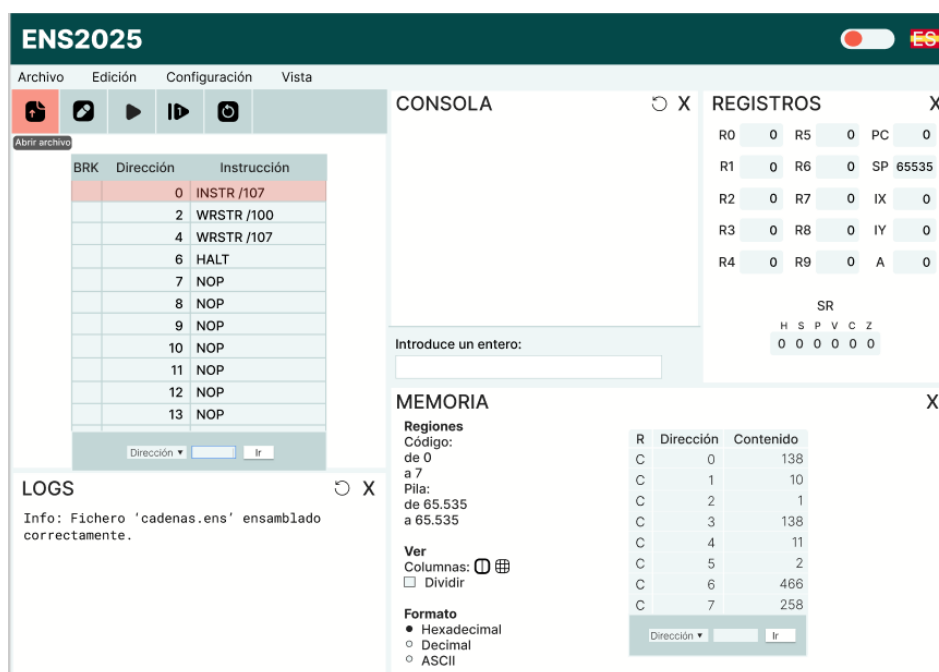


Ilustración 16. Prototipo de alta fidelidad con mejoras aplicadas.

Además, la instrucción a la que apunte el PC estará siempre resaltada para indicar el flujo de la ejecución. El resto de posibles mejoras se han considerado demasiado costosas para el tiempo de que se dispone y se plantean como futuras líneas de trabajo.

4.2 Implementación

4.2.1 Herramientas utilizadas

En el Estado de la cuestión se explican diversos marcos de trabajo y librerías de JavaScript que facilitan la creación de páginas web con la arquitectura SPA. Esta arquitectura es la más adecuada al proyecto, al ser una página con poco contenido que se beneficiaría de una experiencia fluida sin esperas al servidor.

Entre las herramientas presentadas, se ha elegido React como librería principal debido a sus paquetes asociados y a previa familiaridad con su uso. Entre los paquetes más útiles para este proyecto destaca `react-resizable-panels`, que contiene componentes avanzados para manejar ventanas flexibles. Otros marcos de trabajo contienen herramientas similares; sin embargo, `react-resizable-panels` es el único que permite controlar el tamaño de las ventanas mediante manejo por teclado, funcionalidad necesaria para la accesibilidad. Las funcionalidades de React utilizadas en el proyecto se comentarán según se exploran las características de implementación más relevantes.

4.2.2 Organización del proyecto

El proyecto sigue una estructura similar a la mayoría de los proyectos donde se emplea la librería React. Dentro de la carpeta `src`, donde se encuentra todo el código del proyecto, se han creado las siguientes subcarpetas para mantener el código modularizado y fácil de encontrar:

- *Assets*: contiene todas las imágenes e iconos utilizados en la interfaz.
- *Sections*: contiene los componentes que dan lugar a las seis ventanas principales de la interfaz. Cada ventana tiene una carpeta especializada con un archivo de código JSX y un archivo CSS.
- *Components*: contiene otros componentes reutilizables que no están incluidos en la carpeta anterior, como la cabecera de la página, o componentes utilizados en varias ventanas como el menú de navegación de tablas. Cada componente tiene una carpeta especializada con un archivo de código JSX y un archivo CSS.
- *Constants*: contiene constantes del programa y el contenido de los programas de ejemplo.
- *Contexts*: contiene contextos de React. Para más información sobre cómo se usan se puede consultar el apartado Consistencia del estado de la máquina.
- *Controller*: contiene el archivo `simulationController.js`, que maneja la simulación y la comunicación entre módulos.
- *Translations*: contiene archivos JSON con la traducción de las cadenas clave en cada idioma (ver Implementación de la interfaz bilingüe).
- *Utils*: contiene funciones y *hooks* de React utilizados en más de un componente.

4.2.3 Principales características de implementación

La tarea de implementación requiere una gran cantidad de decisiones técnicas. Debido a la imposibilidad de comentar todas las decisiones involucradas en el proceso, se resumen en esta sección las más importantes o aquellas con detalles de implementación más interesantes.

4.2.3.1 Manejo de la vista de las ventanas

El primer requisito del sistema (RF-01) es la implementación de ventanas cuyo tamaño y visibilidad sean ajustables por el usuario. Para crearlas, se ha utilizado el módulo `react-resizable-panels`, que ofrece tres componentes configurables: `PanelGroup`, un contenedor de paneles donde se puede definir la orientación del grupo; `Panel`, donde se coloca el contenido de cada ventana y se define el comportamiento del panel, como su tamaño por defecto, mínimo o máximo; y `PanelResizeHandle`, que actúa de separador y permite al usuario modificar el tamaño de los paneles. En el Fragmento de Código 4 se muestra el

código utilizado para implementar las ventanas de consola, registros y memoria en la disposición diseñada en el prototipo de alta fidelidad.

```
/* Grupo de paneles con consola, registros y memoria */
<PanelGroup direction="vertical" autoSaveId="persistence">

  /* Grupo de paneles con consola y registros */
  {(state.windows.console || state.windows.registers) && (
    <Panel order={1}>
      <PanelGroup direction="horizontal" autoSaveId="persistence">

        /* Panel de consola */
        {state.windows.console && (
          <Panel order={1}>
            <Console />
          </Panel>
        )}

        /* Separador entre consola y registros */
        {state.windows.console && state.windows.registers && (
          <PanelResizeHandle />
        )}

        /* Panel de registros */
        {state.windows.registers && (
          <Panel defaultSize={40} order={2} >
            <Registers />
          </Panel>
        )}
      </PanelGroup>
    </Panel>
  )}

  /* Separador entre consola, registros y memoria */
  {(state.windows.console || state.windows.registers) &&
    state.windows.memory && (
      <PanelResizeHandle className="panel-resize-handle-vertical" />
    )}

  /* Panel de memoria */
  {state.windows.memory && (
    <Panel order={2}>
      <Memory />
    </Panel>
  )}
</PanelGroup>
```

Fragmento de Código 4. Paneles de consola, registros y memoria.

La lógica de redimensión de los paneles está integrada en los componentes del paquete. En cambio, la visibilidad de los paneles se controla mediante *render* condicional a partir del valor verdadero o falso de unas variables globales de control guardadas en un *reducer*. Estas variables se explican con más detalle en el apartado Configuración de la interfaz.

4.2.3.2 Manejo de la ventana de memoria

Dada la longitud de la memoria, el *render* simultáneo de todas las celdas provoca grandes problemas de rendimiento, sobre todo en la edición de las celdas. Para solucionar esto, se ha implementado paginación en la tabla de

memoria. Inicialmente, se cargan las primeras mil celdas. Si el usuario se desplaza por la tabla, al alcanzar el 10% inferior o superior, se cargan 500 celdas más en la dirección correspondiente. Si el usuario busca una dirección, se cargan las 1000 celdas más cercanas a esa dirección y posteriormente se desplaza la tabla para colocar la celda deseada en la primera posición.

El formato de las direcciones y del contenido de la memoria es configurable por el usuario. Los números hexadecimales se representan con el prefijo '0x' y cuatro cifras hexadecimales para mostrar el valor de cada *byte*. En cambio, los números decimales muestran el mínimo número de cifras significativas y no contienen prefijo. El formato ASCII solo muestra los caracteres imprimibles, mientras que la cadena '**' se usa para los caracteres no representables.

Para la edición de las celdas, en los formatos hexadecimal y decimal se permiten números decimales y hexadecimales distinguidos por su prefijo, mientras que en el formato ASCII sólo se permite la inserción de un carácter. Si el usuario introduce una entrada no válida, el usuario recibe un mensaje de error en la ventana de *logs* y el cambio de la memoria no se llevará a cabo.

La ventana de memoria también ofrece dos opciones de vista de la tabla de memoria:

- Duplicar la tabla de memoria: permite que se puedan observar dos zonas independientes de la memoria. Para implementarlo, solo es necesario añadir una instancia del componente `MemoryTable` cuando se activa esta opción.
- Expandir la tabla de memoria: permite mostrar más de una celda de memoria por cada fila de la tabla. El número de celdas por fila depende del ancho de la ventana de memoria y del formato del contenido (los números hexadecimales ocupan más que los decimales). El número de celdas puede ser igual a uno si no hay espacio para más. La columna de dirección indica la dirección de la primera celda de esa fila. De la misma manera, el indicador de región de código aparece si la primera dirección pertenece a esa región. Por otro lado, los indicadores de los registros IX, IY, PC y SP están situados en la fila que contiene la dirección a la que apuntan.

MEMORIA

×

R	Dirección	Contenido			
PC ▶	0	0	0	0	0
C	3	0	0	0	0
C	6	0	0	0	0
C	9	0	0	0	0
C	12	0	0	0	0
C	15	0	0	0	0
C	18	0	0	0	0
C	21	0	0	0	0
C	24	0	0	0	0
C	27	0	0	0	0
C	30	0	0	0	0

Dirección ▼
Ir

R	Dirección	Contenido			
	19968	0	0	0	0
	19971	0	0	0	0
	19974	0	0	0	0
	19977	0	0	0	0
	19980	0	0	0	0
	19983	0	0	0	0
	19986	0	0	0	0
	19989	0	0	0	0
	19992	0	0	0	0
	19995	0	0	0	0
SP ▶	19998	0	0	0	0

Dirección ▼
Ir

Formato
☐ Hexadecimal
☒ Decimal
☐ ASCII

Vista
Agrupar: ☐ ☒
☒ Duplicar

Regiones
Código: de 0 a 3124.

Ilustración 17. Ventana de memoria con tabla duplicada y expandida.

Estas dos opciones se pueden combinar; si ambas están activas, se mostrarían dos tablas y cada tabla tendría el número máximo de columnas, como en la Ilustración 17. Se pueden elegir a través del menú lateral de la ventana de memoria o desde el submenú 'Vista' del menú principal.

4.2.3.3 Manejo de la ventana de desensamblado

La ventana de desensamblado comparte muchas cualidades con la ventana de memoria. Al tener un gran tamaño, no se pueden mostrar todas las celdas simultáneamente, por lo que se usa la misma técnica de paginación, leyendo una sección de 1000 elementos de memoria y convirtiéndolos a instrucciones desensambladas.

La traducción de binario a cadenas se realiza en un archivo separado del componente llamado *decodeInstructions.js*. Para realizar esta traducción, dada una sección de memoria, se lee la primera dirección y se supone que es una instrucción válida, por lo que se intenta decodificar. La primera palabra contiene el código de operación y los modos de direccionamiento de los argumentos, que se encuentran en las palabras posteriores. Si la instrucción completa resulta ser válida, se convierte en texto y se avanza el número de palabras correspondiente; en caso contrario, se imprime el mensaje `'**INSTRUCCIÓN NO IMPLEMENTADA**'` y se avanza sólo una posición.

La tabla de desensamblado también es afectada por el formato de direcciones y contenido elegidos por el usuario. Durante el proceso de traducción de la memoria, los números se convierten a cadena en el formato seleccionado (hexadecimal o decimal; si el formato del contenido es ASCII entonces se coge el formato de las direcciones). Si el usuario cambia el formato del contenido, se vuelve a traducir la sección actual con el nuevo formato.

4.2.3.4 Manejo de la ventana del editor

Para la ventana del editor se ha utilizado el paquete de React llamado *react-monaco-editor* [85] que ofrece una implementación en React del Monaco Editor, un editor creado por Microsoft para su uso en Visual Studio Code. Se ha utilizado este editor porque muchos usuarios ya tendrán familiaridad con su funcionamiento y sus atajos de teclado.

Para utilizarlo, se incluye el componente `Editor` con una serie de atributos para configurarlo: la altura, el tema (claro u oscuro), el contenido inicial, la función a ejecutar en caso de cambio del contenido y una serie de subopciones como la aparición o ausencia de los números de línea o el minimapa de código.

Además, se ha decidido añadir resaltado de sintaxis del lenguaje ensamblador. De esta forma, los usuarios pueden encontrar más fácilmente errores de sintaxis. La configuración de este resaltado se ha realizado en el archivo *monacoConfiguration.js* con los siguientes pasos: primero, se registra el lenguaje con un identificador. Posteriormente, se crean reglas con expresiones regulares para la definición de *tokens*. Si un conjunto de caracteres cumple con la expresión regular correspondiente, se le asigna ese tipo de *token*. En el Fragmento de Código 5 se puede ver el proveedor de *tokens* creado para este lenguaje, con las expresiones regulares de todos sus elementos. El orden de las reglas es importante: en este caso, la primera regla localiza todos los comentarios para que no se busque ningún *token* dentro de un comentario en las siguientes reglas. Como último paso, se declaran los temas claro y oscuro asignando un color a cada tipo de *token* y al fondo del editor.

El componente del editor se encarga de detectar cambios en el contenido del archivo, activando la variable `fileContentChanged` en el `simulationContext` (ver apartado Consistencia del estado de la máquina para más información). Si se produce un cambio y posteriormente se ejecuta, se vuelve a reensamblar el contenido antes de comenzar la simulación.

```

tokenizer: {
  root: [
    // COMENTARIOS
    [/;.*$/, "comentario"],

    // ETIQUETAS DE DEFINICIÓN (Ej: LECTURA:)
    [/^[a-zA-Z_]\w*/, "definicionEtiqueta"],

    // REGISTROS
    [
      /\.[a-zA-Z_]\w*/,
      {
        cases: {
          "@registers": "registro",
          "@default": "registroNoValido",
        },
      },
    ],

    // DIRECCIONAMIENTO INMEDIATO (#100, #0xAF, #ETIQUETA)
    [/#(0x[0-9a-fA-F]+|[0-9]+|[a-zA-Z_]\w*)/, "numero"],

    // DIRECCIONAMIENTO RELATIVO ($200, $0x10, $ETIQUETA)
    [/\$ (0x[0-9a-fA-F]+|[0-9]+|[a-zA-Z_]\w*)/,
      "direccionamientoaPC"],

    // DIRECCIONAMIENTO DIRECTO (/200, /0x10, /PROMPT)
    [/\/(0x[0-9a-fA-F]+|[0-9]+|[a-zA-Z_]\w*)/, "direccion"],

    // INSTRUCCIONES Y MNEMONICOS
    [
      /[a-zA-Z_]\w*/,
      {
        cases: {
          "@instructions": "instruccion",
          "@default": "palabraNoReservada",
        },
      },
    ],

    // OTROS
    ["/", "palabraNoReservada", "@string"],
    [/[\[\]\,]/, "delimitador"],
    [/[0-9]+/, "numero"], // Números sueltos
  ],

  string: [
    [/^[^\\"]+/, "palabraNoReservada"],
    [/"/, "palabraNoReservada", "@pop"],
  ],
}

```

Fragmento de Código 5. Proveedor de tokens de ENS2026.

4.2.3.5 Consistencia del estado de la máquina

Los valores del estado interno de la máquina (memoria, registros y consola) se deben mantener consistentes entre todos los elementos de la interfaz y con el módulo de ejecución. El mejor ejemplo es la memoria, que es utilizada en la tabla de desensamblado y en la tabla de memoria, que al ser duplicable puede

tener una o dos instancias. Los cambios en una celda de memoria tanto hechos por el usuario como por la ejecución de un programa se deben mostrar en las tres tablas simultáneamente.

Para comunicar esta información, se ha optado por utilizar dos conceptos de React: un *reducer* y un contexto.

Un *reducer* es un tipo de función que permite guardar un objeto y cambiar su valor de forma ordenada a través de acciones predefinidas [86]. Contiene un estado, en este caso `simState`, y la función de cambio, llamada `simDispatch`. Se puede llamar a la función de cambio con una acción que contiene un tipo y unos argumentos secundarios. Dependiendo del tipo de acción, la función modifica el estado y devuelve su nuevo valor.

Se ha utilizado un *reducer* para proporcionar una forma estandarizada de controlar el estado de la simulación y de la máquina. El estado del *reducer* contiene un objeto de la clase `SimulationController`, que se encarga de la comunicación con el resto de los módulos y contiene funciones como la ejecución en lote, la ejecución de una instrucción, el ensamblado de un archivo o la modificación de la memoria. Aparte de este objeto, el estado contiene números de versión de todos los elementos que necesitan mantener la consistencia (memoria, registros, salida y contenido del archivo). Estos números son incrementados cada vez que se produce un cambio en su respectivo elemento.

La función de cambio del *reducer* llama a la función adecuada del `SimulationController` e incrementa los números de versión de los elementos involucrados. Por ejemplo, la acción `RESET_SIMULATION` llama a la función `reiniciarSimulacion` del `SimulationController` e incrementa la versión de la memoria y de los registros.

Por otro lado, el contexto de React se utiliza para comunicar el *reducer* con todos los elementos que lo necesiten. La manera habitual de compartir un estado entre varios componentes es pasándolo de componentes padre a componentes hijo. Este método no es eficiente cuando el estado es utilizado por muchos componentes diferentes, como en el caso del estado de la máquina. Por ello, se ha utilizado un contexto, que permite acceder a una variable desde cualquier componente a través de la función `useContext` [87].

Con estos elementos combinados, desde cualquier elemento se puede modificar el estado de la simulación y detectar cambios en sus elementos. La detección de un cambio de versión se suele realizar con la función `useEffect`, a la que se le proporciona una función y una lista de dependencias. Cuando cambia una de las dependencias, se llama a la función proporcionada. Un ejemplo de este comportamiento es el Fragmento de Código 6, situado en la tabla de memoria. Así, cuando se produce un cambio en el número de la versión o en los límites de la sección de memoria activa (se ha buscado una dirección diferente o el usuario se ha acercado al límite superior o inferior), se accede a la memoria principal y se consigue una copia de la sección adecuada.

```
useEffect(() => {
  /* Actualiza el slice de memoria mostrado cuando cambian los
    límites o la memoria */
  const memory = simState.simController.getMemory();
  setMemorySliced(Array.from(memory.slice(memoryBounds[0],
    memoryBounds[1])));
}, [memoryBounds, simState.memoryVersion]);
```

Fragmento de Código 6. Actualización de la sección de memoria.

4.2.3.6 Configuración de la interfaz

La configuración del tema de la interfaz, la visibilidad de las ventanas o los ajustes de vista de la tabla de memoria también deben ser consistentes y accesibles desde varios componentes. Por ejemplo, el evento de cerrar una ventana se debe transmitir entre tres componentes: el botón de cerrar la ventana, el submenú de vista de navegación y la pantalla principal.

Por ello, se ha utilizado la misma técnica que en el apartado anterior, combinando un *reducer* con un contexto para controlar la configuración de la interfaz. Al ser funcionalidades completamente diferentes a las de control de la simulación, se ha creado un *reducer* diferente, que contiene la visibilidad de cada ventana, el formato del contenido y de las instrucciones, la dirección de la pila, la duplicidad de la tabla de memoria, el formato de sus columnas y el tema e idioma de la interfaz.

4.2.3.7 Comunicación con otros módulos

Toda la comunicación con los módulos externos se realiza mediante la clase `SimulationController`.

La interfaz sólo se comunica con el módulo de ensamblado para el ensamblado de archivos. Mantienen una relación cliente-servidor. La interfaz realiza una petición POST, incluyendo en el cuerpo de la petición un objeto con tres elementos: `sourceMode`, que indica el formato del contenido, en este caso se iguala a "REQ_STRING" para indicar que se va a proporcionar una cadena; `source`, donde se pasa todo el contenido del archivo en una cadena, y `group`, donde se puede incorporar opcionalmente el grupo de prácticas de los alumnos, pero en esta comunicación no se incluye. La respuesta a esta petición incluye una lista de errores de ensamblado, el contenido de la memoria, los límites de la región de código y la lista de etiquetas con sus valores correspondientes.

La interfaz mantiene una comunicación más estrecha con el módulo de ejecución. Este módulo proporciona una clase con los métodos públicos necesarios para controlar la ejecución. La interfaz crea un objeto de esta clase, le proporciona la referencia a la memoria y la dirección de la pila y accede a él para llamar a estas funciones. La comunicación entre estos módulos se puede categorizar en tres tipos:

- Funciones simples. Estas funciones se realizan de forma síncrona, como la modificación de un registro o el establecimiento de un punto de ruptura.
- Funciones asíncronas. Las funciones que tardan más tiempo en ejecutar (ejecución en lote y ejecución de una instrucción) son asíncronas, por lo que devuelven una promesa. A través de una cláusula `then`, se establece el código que se desea realizar cuando la promesa haya sido completada, y una cláusula `catch` si se produce una excepción. En el caso de la ejecución, en caso de éxito se debe distinguir entre el fin del programa o la parada por punto de ruptura para enviar un mensaje adecuado, y en caso de error se distingue entre tres excepciones: `ExcepcionMaquina`, `ErrorInesperadoException` y `DemasiadasInstruccionesException`. Este código de esta función se muestra en el Fragmento de Código 7.
- Comunicación mediante eventos. A pesar de que por lo general es la interfaz quien inicia los procesos del módulo de ejecución, hay una situación donde se requiere la comunicación inversa: en el caso de que el programa requiera una entrada del usuario. Para realizar esta

comunicación inversa, el módulo de ejecución envía un evento que es capturado por el elemento de la consola. Este evento contiene la información sobre el tipo de entrada para mostrar en la interfaz el mensaje correcto.

```
function runProgram() {
  simState.simController
    .ejecutarEnLote()
    .then(() => {
      simDispatch({ type: "UPDATE_MEMORY" });
      simDispatch({ type: "UPDATE_REGISTERS" });
      simDispatch({ type: "UPDATE_OUTPUT" });
      if (simState.simController.getBreakpointReached()) {
        simDispatch({
          type: "ADD_LOG_ENTRY",
          payload: {
            logEntry: { type: "info", message: "breakpointReached" },
          },
        });
      } else {
        simDispatch({
          type: "ADD_LOG_ENTRY",
          payload: {
            logEntry: { type: "info", message:
              "programRunSuccessfully" },
          },
        });
      }
    })
    .catch((e) => {
      if (e instanceof ExcepcionMaquina) {
        simDispatch({
          type: "ADD_LOG_ENTRY",
          payload: {
            logEntry: {
              type: "error", message: `${e.message}`,
            },
          },
        });
      } else if (e instanceof ErrorInesperadoException) {
        simDispatch({
          type: "ADD_LOG_ENTRY",
          payload: {
            logEntry: { type: "error", message:
              "unexpectedExecutionError" },
          },
        });
      } else if (e instanceof DemasiadasInstruccionesException) {
        simDispatch({
          type: "ADD_LOG_ENTRY",
          payload: {
            logEntry: { type: "error", message:
              "tooManyInstructions" },
          },
        });
      }
    });
}
```

Fragmento de Código 7. Función de ejecución en lote.

4.2.3.8 Implementación de la interfaz bilingüe

Para la implementación de la interfaz en dos idiomas, español e inglés, se ha utilizado el paquete de internacionalización más popular, `il8next`. Este paquete ofrece un gestor de idiomas centralizado y cambio de idiomas en tiempo real sin necesidad de recargar la página. Para inicializarlo solo es necesario llamar a una función del paquete con la configuración deseada, como el idioma por defecto, el idioma actual y la ruta de los archivos de traducción.

Estos archivos JSON, uno por cada idioma a incluir, contienen traducciones clave-valor. Por ejemplo, la clave `"fileLoadedSuccessfully"` está asociada a la frase `"File loaded successfully"` en el archivo inglés y `"Archivo cargado con éxito"` en el archivo español.

En cada componente en el que haya que mostrar texto por pantalla, se utiliza la función `t()` contenida en el paquete, que sustituye la clave proporcionada por la traducción en el idioma seleccionado. Esta función se utiliza en todos los botones, desplegables, títulos de secciones, tablas e incluso en el texto alternativo de las imágenes o en las etiquetas de los elementos.

La ventana de *logs* requiere un comportamiento especial para implementar correctamente la internacionalización. En lugar de acumular en una variable el texto a mostrar por pantalla, se guardan en una lista las claves de los mensajes. De esta forma, se traducen los mensajes dinámicamente si se produce un cambio de idioma, lo que no sería posible si solo se guardase el contenido del mensaje.

Además, ciertos mensajes como las excepciones de la máquina contienen información con parámetros (por ejemplo, el mensaje `"División por cero en la instrucción 0x100"`, donde la dirección puede cambiar en cada ejecución). Para estos casos, se utilizan cadenas formateadas, que permiten la interpolación de variables. Este tipo de cadenas se declaran en el archivo JSON como en el siguiente ejemplo:

```
"invalidAddress": "Dirección inválida '{{value}}'."
```

Después, el valor deseado se pasa como un argumento más a la función `t()`. Además, se puede declarar el tipo de la variable y cómo se desea formatear (por ejemplo, definir el número de decimales a imprimir).

4.2.4 Implementación de la accesibilidad

Se han utilizado diversos mecanismos para crear un software accesible. A continuación, se comentan las decisiones más relevantes respecto a la accesibilidad.

4.2.4.1 HTML semántico y etiquetas WAI-ARIA

El primer mecanismo utilizado es el uso de HTML semántico y de etiquetas WAI-ARIA en su defecto. Es esencial para garantizar un correcto funcionamiento de la interfaz con productos de apoyo, como lectores de pantalla. Además, las etiquetas de HTML, en ocasiones, incluyen comportamientos accesibles de navegación por teclado.

Algunos ejemplos de uso de HTML semántico son la etiqueta `header` para el encabezado de la página, la etiqueta `nav` para el menú principal, o la etiqueta `table` y sus elementos relacionados (`th`, `tr`, `td`) para las tablas de desensamblado y memoria.

Las etiquetas WAI-ARIA se han utilizado cuando las etiquetas de HTML semántico no aportan suficiente información. Proporcionan dos tipos de información sobre un elemento: su función y su estado.

La función de un elemento se puede declarar con el atributo `role` o el atributo `aria-label`. El atributo `role` designa qué es un elemento, sustituyendo a la etiqueta HTML. Sólo se deben utilizar roles definidos previamente por el W3C [88]. Un ejemplo de uso es la asignación del rol `tooltip` a las etiquetas de los botones de la barra de botones. En cambio, el atributo `label` es más flexible y sirve para etiquetar al elemento. Se ha utilizado mayoritariamente para definir la función de botones sin texto, como los de cerrar una ventana, de vaciar su contenido, o el botón de tema e idioma. Si un elemento posee la descripción de otro, este último debe recibir la etiqueta `labelled-by` seguida del identificador del primer elemento.

Las etiquetas WAI-ARIA también definen el estado de los elementos. Su uso más significativo ha sido en el menú principal, donde se han utilizado los siguientes atributos:

- `aria-has-popup`: indica que este botón despliega contenido al pulsarlo, como en el caso de los menús desplegables.
- `aria-expanded`: indica si un elemento desplegable está expandido o contraído.
- `aria-controls`: seguido de un identificador, explica que el primer elemento controla la visibilidad del segundo.
- `aria-disabled`: indica que un elemento está deshabilitado. Se utiliza en la opción de cambiar la dirección de la pila, que está deshabilitada mientras se está ejecutando un programa.

Combinando estas etiquetas y actualizando su contenido dinámicamente, se consigue definir programáticamente la estructura y el estado actual del menú principal.

Por último, los atributos WAI-ARIA también deben ser utilizados para establecer *live regions*, cuyo contenido es actualizado dinámicamente y debe ser comunicado al usuario.

```
<div
  className={styles["log-output"]}
  ref={logOutputRef}
  role="log"
  aria-live="polite"
  aria-relevant="additions"
  aria-atomic="false"
  aria-label={t("simulationLogs")}
>
  {simState.simLogs.map((entry, index) => {
    return (
      <p key={index} className={styles[entry.type]}
        role={entry.type === "error" ? "alert" : undefined}>
        {t(entry.type).toUpperCase(): {t(entry.message)}}
      </p>
    );
  })}
</div>
```

Fragmento de Código 8. JSX de la ventana de logs.

En la ventana de *logs*, cada vez que aparece un nuevo mensaje, los productos de apoyo debe leer ese *log*, pero no toda la sección. Esto se consigue con la combinación de etiquetas utilizada en el Fragmento de Código 8. La etiqueta *aria-live* con valor “*polite*” indica que se debe leer el nuevo contenido, pero sin interrumpir las acciones del usuario sino cuando haya una pausa en la lectura. La etiqueta *aria-atomic* indica que no es necesario leer todo el contenido del bloque, y *aria-relevant* especifica qué contenido se debe leer: las entradas nuevas. Además, si se trata de un mensaje de error, se le añade el rol *alert*.

Los mismos atributos han sido utilizados en la sección de salida de programa. La entrada también ha sido declarada como región activa, pero con el rol *status*, que indica que se ha actualizado contenido, pero tampoco interrumpe al usuario.

4.2.4.2 Navegación por teclado

Durante la implementación de la navegación por teclado, se alcanzaron tres puntos críticos en el editor, en la ventana de registros y en las tablas de memoria y de desensamblado.

El editor representó un desafío técnico al comportarse como una trampa del teclado (*keyboard trap*, criterio 2.1.2 de WCAG). Una vez que el foco alcanzaba esta sección, la tecla Tabulador dejaba de navegar entre los elementos de la página y se limitaba a insertar sangrías en el texto. Esto impedía al usuario avanzar hacia los elementos exteriores mediante el teclado, afectando gravemente a la accesibilidad. Para corregirlo, se ha desactivado el foco en el editor hasta que el usuario pulse la tecla ‘Enter’ en la ventana del editor. Cuando el usuario quiera volver a navegar por la página a través de la tabulación, puede quitar el foco del editor pulsando ‘Escape’.

Las tablas de memoria y desensamblado presentaban un problema similar. Ambas poseen casillas interactivas (las celdas de puntos de ruptura y las celdas con el contenido de la memoria, que son editables). Por lo tanto, una vez el foco alcanzaba la tabla, recorría todas las filas de la memoria antes de pasar al siguiente elemento. Debido al gran tamaño de la memoria, esto suponía un problema de accesibilidad grave. Para solucionarlo, se ha programado el mismo funcionamiento que en la ventana del editor: para acceder a las celdas de la tabla, el usuario debe poner el foco en la tabla y pulsar la tecla Enter, lo que llevará el foco al primer elemento visible de la tabla y permitirá su libre navegación por las filas. Si el usuario desea dejar de recorrer el interior de la tabla, puede pulsar la tecla Escape.

Este mismo comportamiento se ha implementado en la ventana de registros. Aunque no supone un problema tan grave como en los dos casos anteriores, recorrer todos los registros y biestables para acceder a secciones posteriores puede suponer una molestia para el usuario. Por ello, de la misma manera, el foco inicialmente se mostrará sobre la región entera de registros, mientras que los registros individuales serán accesibles al pulsar Enter.

4.2.4.3 Otros mecanismos

La elección de colores es esencial para la accesibilidad. Para garantizar que se cumple el contraste mínimo para cumplir las WCAG2.2 a nivel AA (criterio 1.4.3), se ha comprobado el contraste a través de comprobadores en línea [89].

La flexibilidad de la página a otros formatos de pantalla y tamaños de letra se ha conseguido a través de herramientas de CSS y siguiendo sus buenas prácticas. Para proporcionar flexibilidad a los elementos se han utilizado los

formatos *flexbox* y *flexgrid*, mientras que el ajuste a diferentes tamaños resulta del uso de medidas relativas como *em* y *rem*, que dependen del tamaño de letra seleccionado, en lugar de medidas absolutas como píxeles.

4.3 Casos de prueba

En esta sección se explica el método utilizado para realizar los tres tipos de pruebas realizados: pruebas unitarias, pruebas de sistema y pruebas de accesibilidad. No se pretende explicar en detalle cada prueba realizada y su resultado, sino presentar el procedimiento que se ha seguido y explicar los fallos más relevantes encontrados con cada tipo de pruebas.

4.3.1 Pruebas unitarias

Las pruebas unitarias tienen como objetivo probar meticulosamente cada elemento de la interfaz por separado, sin considerar las relaciones entre ellos. Se han realizado a través de la librería React Testing Library (RTL) [90]. Esta librería permite realizar pruebas simulando el comportamiento del usuario.

A diferencia de otras herramientas que comprueban el estado interno de los componentes, RTL analiza las pruebas puramente desde la perspectiva del usuario. Por ejemplo, para comprobar la existencia de un botón, en lugar de encontrarlo a través de su identificador o su clase, se busca a través de su rol y contenido. Esta filosofía de funcionamiento ayuda a la accesibilidad; por ejemplo, si no se localiza es probable que el botón no sea accesible, y un lector de pantallas tampoco lo encuentre o no anuncie correctamente su función.

El flujo de las pruebas se consigue a través de tres funcionalidades de la librería. La primera es la disposición de los elementos que se deseen probar (al tratarse de pruebas unitarias, típicamente sólo se prueba un elemento). La segunda es la búsqueda de los elementos involucrados en la prueba a través de funciones como `getByRole`, `getByText` o `getByLabelText`. La tercera es la simulación de acciones (clic del usuario, control del foco, eventos del teclado...) y la comprobación del resultado esperado. Además, si el elemento depende de o interactúa con otros elementos, se pueden crear *mocks* que simulen su comportamiento. El Fragmento de Código 9 contiene un test que comprueba el funcionamiento del formato hexadecimal en la tabla de memoria.

```
test('muestra valores en formato hexadecimal', () => {
  const mocks = setupDefaultMocks();
  mocks.mockMemory[0] = 255;

  render(<MemoryTable codeRange={[0, 100]}
    stackRange={[1000, 2000]} />);

  // Buscar el valor formateado
  expect(screen.getByText('0x00FF')).toBeInTheDocument();
});
```

Fragmento de Código 9. Ejemplo de prueba unitaria.

Las pruebas unitarias no han revelado ningún error significativo del sistema. La causa más probable de este resultado es la realización de pruebas manuales sobre cada elemento de la interfaz mientras se programa, que provoca una detección prácticamente inmediata de los errores.

4.3.2 Pruebas de sistema

Las pruebas de sistema verifican que el sistema completo e integrado cumpla con todos los requisitos funcionales y no funcionales en un entorno similar al real. Este proceso se ha realizado con una matriz de casos de prueba donde se anota el contenido de la prueba y su resultado siguiendo la plantilla de la Tabla 9.

Identificador de la prueba	PS-XX
Requisitos relacionados	[Cód. requisitos]
Descripción	Descripción de la prueba.
Precondiciones	Qué debe ocurrir antes de la realización de la prueba.
Pasos	Lista de pasos a realizar.
Resultado esperado	Qué debería ocurrir.
Resultado real	Qué ocurrió al realizar la prueba.
Estado	OK si el resultado esperado coincide con el real, FALLO en caso contrario.

Tabla 9. Plantilla para pruebas de sistema.

Los casos de prueba verifican todos los requisitos, tanto en circunstancias óptimas como en casos extremos, como la introducción de una dirección no válida o la ejecución de una instrucción cuando se alcanza el final de la memoria. Sin embargo, no se han incluido los requisitos que se ha decidido no implementar según ha progresado la realización del proyecto, explicados en la sección de Resultados y conclusiones.

Las pruebas realizadas han sido útiles para encontrar problemas en la integración de los componentes y en la realización de acciones diferentes al flujo habitual del usuario. Algunos de los problemas encontrados han sido los siguientes:

- Tras ejecutar un programa que solicita una entrada, si en lugar de introducir la entrada, se pide reiniciar la simulación, el programa se reiniciaba pero la ventana de entrada se quedaba visible.
- Si se establecen en un programa uno o varios puntos de ruptura y posteriormente se abre un nuevo archivo, los puntos de ruptura no se eliminaban y se mantenían del programa anterior.

Este tipo de errores se han solucionado controlando más de cerca el estado del simulador (en ejecución, solicitando una entrada, etc.) a través de variables situadas en el `simulationController`.

Además, se ha probado el funcionamiento correcto de la interfaz en los siguientes navegadores: Google Chrome v143.0, Mozilla Firefox v146.0 y Microsoft Edge v143.0. Las pruebas en otros sistemas operativos se realizarán tras el despliegue de la página web, antes de su uso oficial en la asignatura de Traductores de Lenguajes.

4.3.3 Pruebas de accesibilidad

4.3.3.1 Pruebas automáticas

La primera prueba de accesibilidad se ha realizado con Lighthouse [91], una herramienta de Google que analiza cualquier página web directamente desde el navegador y proporciona un informe con los problemas detectados. Los resultados tras la primera ejecución fueron los mostrados en la Ilustración 18.

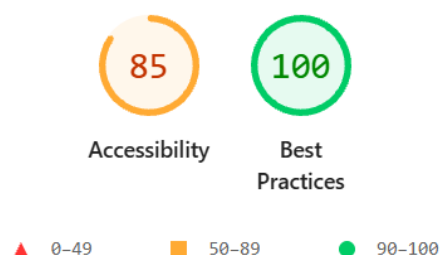


Ilustración 18. Resultados del primer informe de Lighthouse.

El informe proporcionado listaba problemas como la existencia de dos elementos `input` y `select` sin elementos `label` asociados o una sección con una diferencia de contraste insuficiente. Estos problemas fueron solucionados hasta alcanzar la puntuación máxima (Ilustración 19).

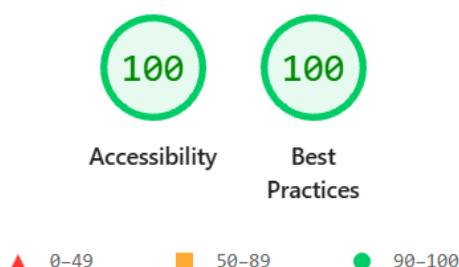


Ilustración 19. Resultados del último informe de Lighthouse.

Sin embargo, este informe no otorga una garantía completa de la accesibilidad de la página, puesto que los requisitos más complejos, como el orden lógico de los elementos en el DOM, las trampas para teclado o el uso de HTML semántico adecuado, deben ser comprobados manualmente.

4.3.3.2 Pruebas manuales

La comprobación del seguimiento de todas las pautas del WCAG 2.2 con nivel de conformidad AA se ha realizado manualmente siguiendo la plantilla de la Tabla 10.

Número de prueba y título de la pauta	Número y título traducido de las WCAG.
Descripción	Descripción de la pauta en las WCAG.
Criterios de éxito	Lista de criterios de éxito asociados a esta pauta (sólo su numeración y su título), hasta nivel de conformidad AA.
Análisis	Descripción de los elementos de la página para los que aplica esta pauta y del método utilizado para comprobar los criterios de éxito.
Resultado	OK si se cumplen todos los criterios de éxito, FALLO en caso contrario.
Cambios realizados (si aplica)	En caso de fallo en el apartado anterior, descripción de los cambios realizados para su cumplimiento.

Tabla 10. Plantilla para pruebas de accesibilidad.

Como muestra del contenido de las pruebas de accesibilidad, se presenta a continuación la comprobación de las cuatro primeras pautas, aquellas relacionadas con el principio de perceptibilidad.

Pauta 1.1. Alternativas textuales.

Número de prueba y título de la pauta	Pauta 1.1. Alternativas textuales.
Descripción	Proporcione alternativas de texto para cualquier contenido que no sea texto, de modo que pueda convertirse en otros formatos que las personas necesiten, como letra grande, Braille, voz, símbolos o lenguaje más sencillo.
Criterios de éxito	1.1.1. Contenido no textual.
Análisis	Se ha comprobado que los siguientes elementos tengan texto alternativo o un atributo aria-label : - Botones sin texto (botones de la barra de botones, botones de control de tema e idioma, botones de cerrar o limpiar ventanas). - Elementos input (entrada para ir a una dirección, opciones del menú de memoria...). Además, se ha comprobado que los elementos decorativos puedan ser ignorados por productos de apoyo, como los iconos de los botones (es el propio botón el que otorga la información sobre su uso).
Resultado	FALLO.
Cambios realizados	Los botones para añadir y eliminar un <i>breakpoint</i> no indicaban su propósito. Se les ha añadido el rol <i>checkbox</i> , un atributo aria-label con el texto “Alternar punto de ruptura en dirección X” y un atributo aria-checked que indica si están activados o desactivados.

Pauta 1.2. Contenidos multimedia basados en el tiempo.

Número de prueba y título de la pauta	Pauta 1.2. Contenidos multimedia basados en el tiempo.
Descripción	Proporcionar alternativas para los contenidos multimedia basados en el tiempo.
Criterios de éxito	1.2.1. Solo audio y solo video (pregrabado). 1.2.2. Subtítulos (pregrabados). 1.2.3. Descripción del audio o contenido alternativo. 1.2.4. Subtítulos (en directo).
Análisis	La aplicación no contiene ningún contenido multimedia dependiente del tiempo.
Resultado	OK.
Cambios realizados	N/A.

Pauta 1.3. Adaptable.

Número de prueba y título de la pauta	Pauta 1.3. Adaptable.
Descripción	Cree contenido que pueda presentarse de diferentes maneras (por ejemplo, con un diseño más sencillo) <u>sin perder información ni estructura</u> .
Criterios de éxito	1.3.1. Información y relaciones. 1.3.2. Secuencia significativa. 1.3.3. Características sensoriales. 1.3.4. Orientación. 1.3.5. Identificar el propósito de las entradas.

Análisis	Esta pauta afecta a todos los elementos del sistema. Para comprobar los dos primeros criterios de éxito, se ha comprobado la estructura y la etiqueta de todos los elementos, revisando que tengan un comportamiento lógico. La orientación se ha comprobado utilizando el modo de diseño reactivo que ofrecen los navegadores para simular otros tamaños de pantalla.
Resultado	OK.
Cambios realizados	N/A.

Pauta 1.4. Distinguible.

Número de prueba y título de la pauta	Pauta 1.4. Distinguible.
Descripción	Facilita a los usuarios la visualización y audición del contenido, incluyendo la separación entre el primer plano y el fondo.
Criterios de éxito	1.4.1. Uso del color. 1.4.2. Control del audio. 1.4.3. Contraste. 1.4.4. Cambiar el tamaño del texto. 1.4.5. Imágenes de texto. 1.4.10. Reflujo. 1.4.11. Contraste de elementos no textuales. 1.4.12. Espaciado del texto. 1.4.13. Contenido con <i>hover</i> o foco.
Análisis	El único elemento que muestra información principalmente a través del color es la instrucción resaltada que indica la posición del PC; sin embargo, el criterio se cumple porque también tiene texto en negrita. Los requerimientos de contraste se han comprobado con páginas de medición de contraste, distinguiendo entre texto grande, texto normal y elementos no textuales. Los criterios de ampliación del tamaño de la letra o del espaciado entre líneas se han realizado utilizando herramientas del navegador.
Resultado	FALLO.
Cambios realizados	Los elementos de la interfaz no textuales no cumplían los requerimientos de contraste de 3:1 respecto al fondo. Se ha añadido un <i>border</i> con el contraste adecuado en los elementos interactivos tanto en el modo claro como en el modo oscuro. Además, el <i>tooltip</i> de los botones de la barra de botones, que aparece cuando se pone encima el ratón, se ha modificado para cumplir la pauta de poder acceder al nuevo contenido sin que desaparezca.

5 Resultados y conclusiones

El propósito de esta sección es presentar el producto final obtenido tras el proceso de desarrollo y reflexionar sobre el grado de cumplimiento de los objetivos planteados al inicio del proyecto. A través de este análisis, se contrastan las expectativas iniciales con la realidad de la implementación, evaluando el impacto que esta herramienta tendrá en el aprendizaje de los alumnos.

El resultado principal de este trabajo es una plataforma web que moderniza la experiencia de simulación del lenguaje ENS2001. La interfaz final se puede observar en la Ilustración 20. Para conocer en más detalle la funcionalidad completa del sistema, se puede consultar el Manual de usuario.

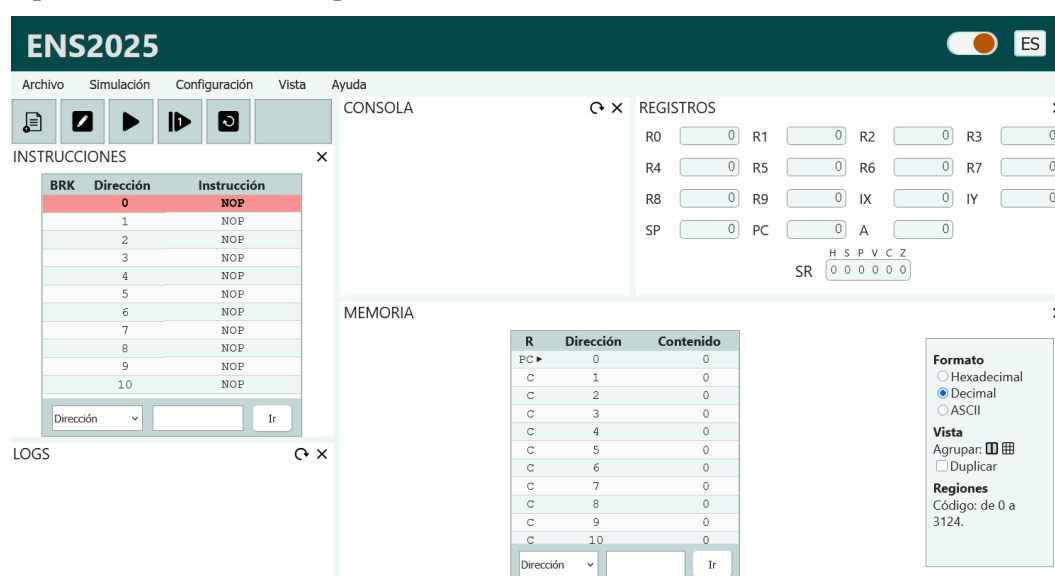


Ilustración 20. Interfaz final del ENS2025.

La herramienta ha sido diseñada pensando específicamente en las necesidades de los alumnos de Traductores de Lenguajes. Se ha priorizado un diseño centrado en el usuario al integrar los resultados de los test de usabilidad realizados durante la fase de diseño, lo que ha permitido refinar la experiencia del usuario desde las etapas iniciales. Las funcionalidades implementadas mantienen los beneficios de su predecesor y ofrecen ventajas respecto a los inconvenientes encontrados:

1. **Accesibilidad universal sin instalación:** Al ser una aplicación web, se elimina la barrera de la compatibilidad con sistemas operativos y la necesidad de actualizaciones manuales. El acceso es inmediato desde cualquier navegador moderno.
2. **Editor integrado:** Se ha eliminado la dependencia de herramientas externas. El alumno puede redactar, modificar y ensamblar su código en un único entorno de trabajo.
3. **Control directo y visual:** Las funciones críticas de depuración (ejecución paso a paso, ejecución en lote y el nuevo botón de reinicio) y la mayor parte de las funciones de configuración están disponibles mediante botones directos, evitando la navegación por menús secundarios que dificultaban el flujo de trabajo en la versión original.

4. Interfaz moderna, flexible y configurable: A diferencia del predecesor, el sistema permite la personalización del aspecto visual (tema claro y oscuro) y del idioma (español e inglés), adaptándose a las preferencias del estudiante.

Debido a la complejidad técnica de ciertos módulos y el límite de tiempo, no se ha conseguido la inclusión total de los requisitos secundarios. Se ha priorizado mantener un ritmo de desarrollo sostenible y asegurar una fase de pruebas de accesibilidad exhaustiva antes que comprometer la calidad del sistema por falta de tiempo.

Los requisitos que no han podido ser implementados son:

- RF-07: Editar una instrucción desensamblada. Actualmente, el desensamblado es de solo lectura; cualquier cambio debe realizarse desde el editor de código fuente.
- RF-18: Mostrar sección de la pila. Aunque el puntero de pila (SP) es funcional, no se ha habilitado una vista dedicada para visualizar el contenido específico de la pila en tiempo real.

Ambas funcionalidades han sido incluidas en la sección de Futuras líneas de trabajo para su implementación en iteraciones posteriores del proyecto.

Respecto a las conclusiones acerca del proyecto, la conclusión más relevante de este Trabajo de Fin de Grado es el valor de la colaboración entre alumnos para mejorar activamente la calidad educativa de la Universidad Politécnica de Madrid. Haber desarrollado una herramienta que contribuirá directamente a que futuros compañeros superen con éxito la asignatura de Traductores de Lenguajes es la mayor satisfacción del proyecto.

Este trabajo ha resultado ser un proyecto completo que ha requerido los conocimientos transmitidos en varias asignaturas de la carrera de Ingeniería Informática. Las más relevantes son Estructura de Computadores, donde se estudia el funcionamiento interno de los computadores y el lenguaje ensamblador; Interacción Persona-Ordenador, dedicada al diseño de interfaces accesibles y amigables, donde además se realizan pruebas de usabilidad y se analizan sus resultados; e Ingeniería del Software I, en la que se explica el ciclo de vida de los productos y el diseño de los requisitos, casos de uso o diagramas de secuencia de un sistema.

Para finalizar, cabe destacar los diversos desafíos técnicos que han surgido con su implementación. El principal desafío ha sido el uso avanzado de la librería de React. La lógica del simulador ha requerido herramientas avanzadas como uso de *hooks* e incluso creación de *hooks* propios, *reducers*, *contexts*, manejo del DOM virtual... Muchas de estas funciones resultaron una experiencia nueva y requirieron un proceso de investigación.

Además, ciertos componentes como la tabla de memoria debían contener comportamientos diversos y complejos, como el manejo por teclado, el control del foco, la edición de celdas, el control del *scroll* del usuario y de la paginación y la adaptación al formato de las instrucciones y el contenido. Esta complejidad supuso una dificultad añadida tanto en la codificación como en la integración con otros elementos y en la etapa de pruebas.

Por otro lado, el uso de React Testing Library también supuso una nueva experiencia. Aprender a probar el comportamiento desde la perspectiva del usuario, y no desde la implementación interna, requirió un cambio de pensamiento respecto a las pruebas tradicionales.

La verificación de las WCAG también ha supuesto un desafío técnico y un cambio de perspectiva. Mientras que antes de la realización del proyecto la accesibilidad parecía un requisito sencillo que solo englobaba el uso de buenas prácticas de programación, todas las diversas medidas y mecanismos de accesibilidad utilizados en el proyecto han revelado la verdadera complejidad de la tarea y el diseño meticuloso que implica en cada elemento de la interfaz.

6 Análisis de Impacto

Este capítulo analiza el impacto potencial del sistema desarrollado en diferentes contextos. Se destacan tanto los beneficios esperados como los posibles efectos perjudiciales, y se relacionan los resultados con los Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030. También se comentan las decisiones tomadas a lo largo del proyecto que tienen como base la consideración del impacto.

6.1 Impacto Personal

El desarrollo de este TFG ha supuesto un crecimiento significativo en múltiples áreas. A nivel técnico, he aprendido React y otras herramientas de desarrollo web, ampliando mi conocimiento en tecnologías que no había utilizado previamente. También he profundizado en la realización de pruebas de *frontend* mediante React Testing Library, una competencia esencial para el desarrollo software que nunca había realizado con anterioridad.

Más allá de las habilidades técnicas, este proyecto me ha permitido enfrentarme al diseño completo de un sistema, desde la conceptualización inicial hasta su implementación. He tenido que tomar decisiones de diseño y experiencia del usuario, además de decisiones más técnicas, lo que me ha dado una visión global del proceso de desarrollo de software.

La gestión del tiempo ha sido otro aspecto fundamental. Un proyecto de esta envergadura requiere planificación a largo plazo y la capacidad de seguir esta planificación de forma constante o cambiarla si ocurren imprevistos.

La comunicación también ha jugado un papel importante. He trabajado en coordinación con los tutores del proyecto y con los estudiantes responsables de los otros módulos del sistema, mejorando mis habilidades de comunicación en equipo en entornos profesionales.

Por último, este trabajo supone mi primera contribución a la comunidad académica. La puesta en producción de un producto propio para su uso real en una asignatura del Grado en Ingeniería Informática representa un gran logro personal que marcará mi carrera como informática.

Como posibles impactos negativos, es necesario mencionar la sobrecarga que ha supuesto compaginar este proyecto con otras responsabilidades académicas y personales. Además, la limitación temporal de la asignatura del TFG ha resultado en un alcance más reducido, al dejarse sin implementar algunas funcionalidades planteadas en el diseño inicial.

6.2 Impacto Empresarial

Aunque este proyecto está dirigido principalmente al ámbito educativo, tiene implicaciones empresariales indirectas. Los estudiantes que utilicen este sistema en la asignatura Traductores de Lenguajes acabarán incorporándose al mercado laboral, muchos de ellos en empresas tecnológicas. Una formación de mayor calidad se traduce en profesionales mejor preparados, lo que beneficia al sector tecnológico en su conjunto.

Por último, aunque el objetivo principal del sistema es educativo, es posible su uso en otros contextos, incluido el empresarial. Aunque este no sea el caso de uso esperado, existe la posibilidad de que el simulador encuentre aplicaciones

en entornos profesionales donde se requiera trabajar con un ensamblador sencillo de forma simple y rápida.

6.3 Impacto Social

El impacto social más importante de este proyecto es la normalización de la accesibilidad como requisito central. Este sistema contribuye a que más estudiantes, incluidos aquellos con limitaciones visuales, auditivas, motoras o cognitivas, puedan aprender ensamblador en igualdad de condiciones. La educación inclusiva es un derecho fundamental.

Además, al ser una herramienta web, el sistema está disponible para cualquier persona que desee aprender ensamblador, independientemente de su contexto económico o geográfico. Por ello, este proyecto amplía las oportunidades de formación y reduce las desigualdades en el ámbito de la educación tecnológica.

6.4 Impacto Económico

El impacto económico de este sistema es mínimo. Sin embargo, la sustitución de una aplicación de escritorio por una aplicación web tiene consecuencias económicas, debido a que este nuevo formato elimina la necesidad de mantener versiones separadas para diferentes sistemas operativos, reduciendo significativamente los costos de mantenimiento.

6.5 Impacto Medioambiental

El impacto medioambiental del sistema respecto a su predecesor no ha sido estudiado. Este proyecto proporciona una aplicación web centralizada para usarse en Traductores de Lenguajes durante años, por lo que el análisis medioambiental y la reducción de su huella de carbono puede resultar interesante como futura línea de trabajo.

6.6 Relación con los Objetivos de Desarrollo Sostenible

Este proyecto contribuye a varios Objetivos de Desarrollo Sostenible de la Agenda 2030:

ODS 4 – Educación de calidad. El desarrollo de este TFG supone una mejora directa en la calidad educativa de la asignatura Traductores de Lenguajes. Al proporcionar una herramienta más moderna con funcionalidades añadidas, se facilita el aprendizaje de conceptos fundamentales de la Ingeniería Informática. Además, mi propio crecimiento personal y profesional durante el proyecto también se alinea con este objetivo.

ODS 10 – Reducción de las desigualdades. Al priorizar la accesibilidad, el sistema contribuye a reducir las desigualdades en el acceso a la educación tecnológica. Estudiantes con diferentes capacidades, contextos económicos y ubicaciones geográficas pueden beneficiarse de la herramienta en igualdad de condiciones.

6.7 Decisiones Basadas en el Impacto

A lo largo del desarrollo del proyecto se han tomado decisiones específicas con el objetivo de maximizar el impacto positivo del sistema:

Diseño centrado en el usuario. Desde las primeras fases del proyecto, se priorizó un enfoque centrado en el usuario. Los test de usabilidad con usuarios reales permitieron identificar problemas y mejorar la experiencia antes de la implementación final. Esta decisión garantiza que el sistema realmente responde a las necesidades de sus usuarios.

Accesibilidad. La decisión de seguir las pautas WCAG 2.2 con nivel de conformidad AA tiene un impacto directo en la inclusividad del sistema.

Mejora de funcionalidades para potenciar la educación. La inclusión de funcionalidades mejoradas como un editor integrado, botones de depuración mejorados y opciones de personalización son decisiones orientadas a mejorar la experiencia de aprendizaje. Cada funcionalidad añadida se evaluó en función de su impacto educativo potencial.

Eficiencia. Decisiones como la paginación de las tablas de desensamblado y memoria han sido tomadas para mejorar la eficiencia del sistema y reducir su consumo energético.

7 Futuras líneas de trabajo

El análisis de las futuras líneas de trabajo se presenta dividido en dos secciones: los siguientes pasos a seguir, considerados esenciales para el despliegue correcto de la aplicación durante el siguiente semestre, y una lista de posibles funcionalidades a añadir.

7.1 Siguiendo pasos a seguir

Pruebas de usabilidad con el sistema final.

Sería necesario realizar otra ronda de pruebas de usabilidad con el sistema final para analizar la eficacia de las mejoras implementadas. Además, podría ser de gran valor incorporar a usuarios con diferentes discapacidades para comprobar la accesibilidad de la página más allá del cumplimiento de las pautas WCAG 2.2.

Seguimiento del despliegue de la aplicación.

Se pretende poner el sistema en uso en febrero de 2026, para ser utilizado por primera vez por los alumnos de Traductores de Lenguajes en el segundo semestre del curso 2025/2026. Durante los inicios, es fundamental realizar un seguimiento del despliegue del sistema para detectar posibles problemas no encontrados con anterioridad. Los errores pueden ser reportados por los alumnos y deben ser lidiados con la urgencia que corresponda dependiendo de su gravedad.

7.2 Funcionalidades a añadir

Editar una instrucción desensamblada.

Esta funcionalidad, inicialmente un requisito de la aplicación, se refiere a la edición de una instrucción desde la ventana de desensamblado sin necesidad de reensamblado del programa, siempre que la nueva instrucción ocupe el mismo número de palabras que la anterior. De esta forma, disminuye el número de comunicaciones entre el cliente y el servidor. Para implementarlo, sería necesaria la creación de un analizador léxico, sintáctico y semántico simple que convierta la nueva instrucción en binario y, si se cumplen las restricciones de tamaño, modifique directamente la memoria en la dirección correcta para reemplazar la instrucción antigua.

Deshacer una instrucción.

Esta funcionalidad fue sugerida en las pruebas de usabilidad, aunque no se llegó a realizar por la complejidad que supone el cambio, sobre todo por la eficiencia de memoria. Se podría permitir retroceder una instrucción hasta alcanzar el inicio del programa, o restringir a un número máximo de instrucciones (por ejemplo, sólo se podrían deshacer tres instrucciones). En ambos casos, para implementarlo sería necesario almacenar un historial con el estado de la máquina en cada instrucción a la que se pueda retroceder. Esto supone un gran uso de memoria, al tener que almacenar la matriz completa de memoria varias veces, por lo que convendría investigar alguna manera de almacenar solo las direcciones o secciones modificadas de una instrucción a la siguiente en lugar de la memoria entera.

Añadir más temas de color e idiomas.

Esta ampliación de la configuración de la página también se sugirió en las pruebas de usabilidad. Para añadir un tema de color, sólo sería necesario cambiar las variables de color situadas en el archivo `app.css` bajo un nuevo valor para la variable `data-theme`, que es la que controla el tema de la página. Como ejemplo, en el Fragmento de Código 10 se muestran las declaraciones de variables CSS del tema claro.

```
[data-theme="light"] {  
  --bg-darkest: #044949;  
  --bg-dark: #c2d6d6;  
  --bg: #eff6f6;  
  --bg-light: #ffffffe;  
  --bg-light-variation: #fafafa;  
  --text: #232323;  
  --hover: #fac5ba;  
  --pressed: #f89587;  
  --warning: #b85301;  
  --error: #ad0000;  
  --border: rgba(0, 0, 0, 0.5);  
  --desactivado: hsl(0, 0%, 75%);  
}
```

Fragmento de Código 10. Declaraciones de variables CSS asociadas al modo claro.

Es importante comprobar que los nuevos colores elegidos sigan manteniendo el contraste mínimo para cumplir las pautas del WCAG 2.2 a nivel AA, que recomiendan una ratio entre el color principal y el color de fondo de 4.5:1 para texto normal y 3:1 para texto grande. Además, el botón de selección del tema actual sólo permite dos opciones, por lo que sería necesario convertirlo en un desplegable para permitir más temas.

Si esta idea se llevase a su extremo, se podría crear un editor de configuración de colores, donde los usuarios puedan elegir libremente el color de cada elemento de la interfaz y guardar su configuración.

Respecto a la incorporación de nuevos idiomas si se considerase necesario, haría falta crear un nuevo archivo JSON que contenga las traducciones a todas las claves, pudiendo utilizar el archivo de español o de inglés como base, modificar el archivo `i18n.js` para añadir el nuevo idioma y modificar el botón de idioma en la interfaz.

Aumentar la flexibilidad de las ventanas.

Actualmente, la interfaz permite modificar el tamaño y la visibilidad de las ventanas. Para aumentar la flexibilidad, se podría permitir cambiar la disposición de las ventanas, pudiendo arrastrarlas a su nueva posición libremente.

Como esta funcionalidad no está incluida en el paquete usado actualmente para manejar las ventanas (`react-resizable-panels`), es probable que hiciese falta implementar su lógica desde cero, lo que implica manejar el cambio de tamaño, el cambio de posición, la visibilidad y la permanencia de los cambios (actualmente se guardan todos los datos acerca de la disposición de las ventanas en el almacenamiento local del navegador).

Detectar la sección de pila.

Al igual que se muestra en la ventana de memoria la sección del código en el menú lateral y en las casillas de región (R) de la tabla de memoria, se podría

detectar el tamaño y la ubicación de la sección de pila. Esto requeriría modificaciones en la comunicación con el módulo de ejecución, al poseer él el valor del registro SP en todo momento.

8 Bibliografía

- [1] MSMK, «El lenguaje Assembler,» 26 febrero 2025. [En línea]. Available: <https://msmk.university/el-lenguaje-assembler/>. [Último acceso: 25 octubre 2025].
- [2] M. Albert Orenge y G. Enrique Manonellas, «Programación en ensamblador (x86-64),» Universitat Oberta de Catalunya, 2011. [En línea]. Available: <https://openaccess.uoc.edu/server/api/core/bitstreams/7bb6c3ac-fd38-4b61-9b22-36a519e3d1bc/content>. [Último acceso: 25 octubre 2025].
- [3] S. Hierso, «Coding From 1849 to 2022: a Guide to The Timeline of Programming Languages,» 10 junio 2022. [En línea]. Available: https://www.computer.org/publications/tech-news/insider-membership-news/timeline-of-programming-languages?utm_source=chatgpt.com. [Último acceso: 5 noviembre 2025].
- [4] R. Maldonado, «¿Qué son los lenguajes ensambladores?,» KeepCoding, 14 octubre 2024. [En línea]. Available: <https://keepcoding.io/blog/que-son-los-lenguajes-ensambladores/>. [Último acceso: 25 octubre 2025].
- [5] F. J. Álvarez Valero, «ENS2001,» marzo 2003. [En línea]. Available: <https://ens2001.falvarez.es/>. [Último acceso: 30 octubre 2025].
- [6] MDN, «SPA (Single-page application),» [En línea]. Available: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>. [Último acceso: 21 octubre 2025].
- [7] R. Saquete, «AJAX,» 29 octubre 2019. [En línea]. Available: <https://www.humanlevel.com/diccionario-marketing-digital/ajax>. [Último acceso: 22 octubre 2025].
- [8] C. Álvarez Caules, «SPA vs MPA y las Arquitecturas Web,» 23 septiembre 2020. [En línea]. Available: <https://www.arquitecturajava.com/spa-vs-mpa-y-las-arquitecturas-web/>. [Último acceso: 23 octubre 2025].
- [9] Lvivity Team, «Single-Page Application vs Multi-Page Application: Pros, Cons, and Which is Better?,» [En línea]. Available: <https://lvivity.com/single-page-app-vs-multi-page-app>. [Último acceso: 23 octubre 2025].
- [10] D. Prasad Acharya, «¿Qué son las aplicaciones de página única? Ejemplos, marcos y más,» 17 mayo 2024. [En línea]. Available: <https://geekflare.com/es/single-page-applications/>. [Último acceso: 25 octubre 2025].
- [11] S. López Mora, «¿Qué son las Single-page application (SPA)? El desarrollo elegido por Gmail y LinkedIn,» 8 Junio 2021. [En línea]. Available: <https://digital55.com/blog/que-son-single-page-application-spa-desarrollo-elegido-por-gmail-linkedin/>. [Último acceso: 21 octubre 2025].

- [12] S. Lawton Henry, «Introducción a la Accesibilidad Web,» 7 marzo 2024. [En línea]. Available: <https://www.w3.org/WAI/fundamentals/accessibility-intro/es>. [Último acceso: 23 octubre 2025].
- [13] GoDaddy, «W3C: Qué es, estándares principales y su importancia en el desarrollo web,» 26 noviembre 2024. [En línea]. Available: <https://www.godaddy.com/resources/es/crearweb/w3c-world-wide-web-consortium-que-es>. [Último acceso: 23 octubre 2025].
- [14] Digital Education Strategies, The Chang School, «La evolución de la accesibilidad web,» [En línea]. Available: https://espanol.libretexts.org/Bookshelves/Vocacional/Vocacional/Aplicaciones_informaticas_y_tecnologia_de_la_informacion/Aplicaciones_informaticas/Auditoria_Profesional_de_Accesibilidad_Web_Facil/03%3A_Introducci%C3%B3n_a_WCAG/3.03%3A_La_evoluci%C3%B3n_d. [Último acceso: 23 octubre 2025].
- [15] W3C, «Web Content Accessibility Guidelines 1.0,» 5 mayo 1999. [En línea]. Available: <https://www.w3.org/TR/WAI-WEBCONTENT/>. [Último acceso: 23 octubre 2025].
- [16] W3C, «Web Content Accessibility Guidelines (WCAG) 2.0,» 8 diciembre 2008. [En línea]. Available: <https://www.w3.org/TR/WCAG20/>. [Último acceso: 25 octubre 2025].
- [17] W3C, «Introduction to Understanding WCAG 2.0,» [En línea]. Available: <https://www.w3.org/TR/UNDERSTANDING-WCAG20/intro.html#introduction-fourprincs-head>. [Último acceso: 25 octubre 2025].
- [18] W3C, «Web Content Accessibility Guidelines (WCAG) 2.2,» 12 diciembre 2024. [En línea]. Available: <https://www.w3.org/TR/WCAG22/>. [Último acceso: 25 octubre 2025].
- [19] Gobierno de España, «Real Decreto 1112/2018, de 7 de septiembre, sobre accesibilidad de los sitios web y aplicaciones para dispositivos móviles del sector público.,» Boletín Oficial de España, 19 septiembre 2018. [En línea]. Available: <https://www.boe.es/eli/es/rd/2018/09/07/1112>. [Último acceso: 25 10 2025].
- [20] mdn, «ARIA,» 2 octubre 2025. [En línea]. Available: <https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA>. [Último acceso: 25 octubre 2025].
- [21] Utah State University, «WAVE Web Accessibility Evaluation Tools,» [En línea]. Available: <https://wave.webaim.org/>. [Último acceso: 25 noviembre 2025].
- [22] Universidad Europea, «¿Qué es el lenguaje de marca o lenguaje de marcado?,» 24 enero 2023. [En línea]. Available: <https://universidadeuropea.com/blog/que-es-lenguaje-marca/>. [Último acceso: 25 octubre 2025].
- [23] W3C, «HTML Tags,» [En línea]. Available: <https://www.w3.org/History/19921103->

hypertext/hypertext/WWW/MarkUp/Tags.html. [Último acceso: 25 octubre 2025].

- [24] Inába, «Versiones de HTML: Desde HTML 1.0 Hasta HTML5,» [En línea]. Available: <https://www.inabaweb.com/versiones-de-html-desde-html-1-0-hasta-html5/>. [Último acceso: 25 octubre 2025].
- [25] ESIC University, «HTML5: qué es, características y cómo funciona,» mayo 2023. [En línea]. Available: <https://www.esic.edu/rethink/tecnologia/html5-que-es-caracteristicas-y-como-funciona-c>. [Último acceso: 25 octubre 2025].
- [26] MDN, «HTML: Una buena base para la accesibilidad,» 21 julio 2025. [En línea]. Available: https://developer.mozilla.org/es/docs/Learn_web_development/Core/Accessibility/HTML. [Último acceso: 25 octubre 2025].
- [27] geeksforgeeks, «CSS History and Versions,» 23 julio 2025. [En línea]. Available: <https://www.geeksforgeeks.org/css/css-history-versions/>. [Último acceso: 26 octubre 2025].
- [28] w3schools, «CSS Syntax,» [En línea]. Available: https://www.w3schools.com/css/css_syntax.ASP. [Último acceso: 26 octubre 2025].
- [29] A. Raj, «History of CSS: The Evolution of Web Design,» AlmaBetter, 4 febrero 2024. [En línea]. Available: <https://www.almabetter.com/bytes/articles/history-of-css>. [Último acceso: 26 octubre 2025].
- [30] C. Coyier, «CSS Flexbox Layout Guide,» CSS-tricks, 12 agosto 2024. [En línea]. Available: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>. [Último acceso: 26 octubre 2025].
- [31] MDN, «CSS Grid Layout,» 17 diciembre 2024. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/CSS/CSS_grid_layout. [Último acceso: 26 octubre 2025].
- [32] A. Casero, «Guía completa de CSS Modules: la clave para mejorar tu código,» 3 octubre 2024. [En línea]. Available: <https://keepcoding.io/blog/que-es-css-modules-y-para-que-sirve/>. [Último acceso: 26 octubre 2025].
- [33] geeksforgeeks, «History of JavaScript,» 27 julio 2025. [En línea]. Available: <https://www.geeksforgeeks.org/javascript/history-of-javascript/>. [Último acceso: 28 octubre 2025].
- [34] Z. Ali, «The History of JavaScript: From Birth to Present,» 29 junio 2025. [En línea]. Available: <https://zeeshan.p2pclouds.net/blogs/history-of-javascript/>. [Último acceso: 28 octubre 2025].
- [35] Web Reference, «A Brief History of ECMAScript Versions in JavaScript,» [En línea]. Available: <https://webreference.com/javascript/basics/versions/>. [Último acceso: 28 octubre 2025].

- [36] viewnext, «Diferencias entre Librerías y Frameworks,» [En línea]. Available: <https://www.viewnext.com/diferencias-entre-librerias-y-frameworks/>. [Último acceso: 27 octubre 2025].
- [37] Neubox Blog, «¿Qué son los Frameworks y las Bibliotecas en el Desarrollo Web?,» 28 octubre 2024. [En línea]. Available: <https://neubox.com/blog/que-son-frameworks-y-bibliotecas-desarrollo-web/>. [Último acceso: 27 octubre 2025].
- [38] L. Karrys, «About npm,» npm Docs, 23 octubre 2023. [En línea]. Available: <https://docs.npmjs.com/about-npm>. [Último acceso: 27 octubre 2025].
- [39] Vite, «Getting Started,» [En línea]. Available: <https://vite.dev/guide/>. [Último acceso: 27 octubre 2025].
- [40] J. Castro, «DOM vs. Virtual DOM | ¿Cuál es la mejor estrategia de render en el navegador?,» Platzi, 16 junio 2021. [En línea]. Available: <https://platzi.com/blog/virtual-dom/>. [Último acceso: 26 octubre 2025].
- [41] geeksforgeeks, «Difference Between Virtual DOM and Real DOM,» 9 enero 2025. [En línea]. Available: <https://www.geeksforgeeks.org/reactjs/difference-between-virtual-dom-and-real-dom/>. [Último acceso: 26 octubre 2025].
- [42] React Dev, «Gestionando programáticamente el foco.,» [En línea]. Available: <https://es.legacy.reactjs.org/docs/accessibility.html#programmatically-managing-focus>. [Último acceso: 26 octubre 2025].
- [43] M. Herrman, «Three Insights I Gained While Researching Vue.js Accessibility,» Smashing Magazine, 16 julio 2021. [En línea]. Available: <https://www.smashingmagazine.com/2021/07/three-insights-vuejs-accessibility/#focus-management>. [Último acceso: 26 octubre 2025].
- [44] D. Ford, «React Text Input Losing Focus After Each Keypress,» Medium, 25 marzo 2022. [En línea]. Available: <https://levelup.gitconnected.com/react-text-input-losing-focus-after-each-keypress-dd6eabfe2f7c>. [Último acceso: 26 octubre 2025].
- [45] C. Marcuta, «Svelte Accessibility Best Practices,» MoldStud, 27 octubre 2024. [En línea]. Available: <https://moldstud.com/articles/p-svelte-accessibility-best-practices>. [Último acceso: 26 octubre 2025].
- [46] T. Richards, «A ‘Vue’ of Accessibility: Tips and Tricks to make your application accessible,» deque, 17 junio 2021. [En línea]. Available: <https://www.deque.com/blog/a-vue-of-accessibility-tips-and-tricks-to-make-your-application-accessible/>. [Último acceso: 26 octubre 2025].
- [47] W3C, «Accessible Rich Internet Applications (WAI-ARIA) 1.2,» 6 junio 2023. [En línea]. Available: <https://www.w3.org/TR/wai-aria-1.2/>. [Último acceso: 26 octubre 2025].
- [48] F. Schroiff, «The Complete Guide to ARIA Live Regions for Developers,» A11y Collective, 5 diciembre 2024. [En línea]. Available: <https://www.a11y-collective.com/blog/aria-live/>. [Último acceso: 26 octubre 2025].

- [49] Stack Overflow, «Web frameworks and technologies,» 2025. [En línea]. Available: <https://survey.stackoverflow.co/2025/technology#1-web-frameworks-and-technologies>. [Último acceso: 26 octubre 2025].
- [50] F. García de Zúñiga, «Características de React: las más prácticas y útiles,» 8 mayo 2023. [En línea]. Available: <https://www.arsys.es/blog/caracteristicas-reactjs>. [Último acceso: 27 octubre 2025].
- [51] React Dev, «Tu primer componente,» [En línea]. Available: <https://es.react.dev/learn/your-first-component>. [Último acceso: 27 octubre 2025].
- [52] React Dev, «Escribir marcado con JSX,» [En línea]. Available: <https://es.react.dev/learn/writing-markup-with-jsx>. [Último acceso: 27 octubre 2025].
- [53] React Dev, «Built-in React Hooks,» [En línea]. Available: <https://react.dev/reference/react/hooks>. [Último acceso: 27 octubre 2025].
- [54] Redux, «Redux Essentials, Part 1: Redux Overview and Concepts,» 29 agosto 2024. [En línea]. Available: <https://redux.js.org/tutorials/essentials/part-1-overview-concepts>. [Último acceso: 27 octubre 2025].
- [55] B. Vaughn, «react resizable panels,» [En línea]. Available: <https://react-resizable-panels.vercel.app/>. [Último acceso: 27 octubre 2025].
- [56] npm, «react-i18next,» [En línea]. Available: <https://www.npmjs.com/package/react-i18next>. [Último acceso: 27 octubre 2025].
- [57] rootstack, «VueJS: Ventajas y desventajas de este framework,» [En línea]. Available: <https://rootstack.com/es/blog/vuejs-ventajas-desventajas>. [Último acceso: 27 octubre 2025].
- [58] F. García de Zúñiga, «Qué es Vue: por qué usarlo como framework JS de referencia,» 24 abril 2024. [En línea]. Available: <https://www.arsys.es/blog/vuejs>. [Último acceso: 27 octubre 2025].
- [59] S. Esemé, «10 cosas que debes saber sobre Vue.js Frontend Framework,» kinsta, . [En línea]. Available: <https://kinsta.com/es/blog/vue-js/#componente-de-archivo-nico>. [Último acceso: 27 octubre 2025].
- [60] CodingPotions, «Introducción a Vue JS >> Qué es y sus características,» 3 noviembre 2019. [En línea]. Available: <https://codingpotions.com/que-es-vue/>. [Último acceso: 27 octubre 2025].
- [61] A. Bhushan, «Svelte vs React: Características, Rendimiento y Más,» 17 enero 2025. [En línea]. Available: <https://kinsta.com/es/blog/svelte-vs-react/>. [Último acceso: 27 octubre 2025].
- [62] M. Seyedi, «What is SvelteKit: Review and Features,» 15 mayo 2025. [En línea]. Available: <https://bejamas.com/hub/web-frameworks/sveltekit>. [Último acceso: 27 octubre 2025].

- [63] Svelte Docs, «Introduction,» [En línea]. Available: <https://svelte.dev/docs/kit/introduction>. [Último acceso: 27 octubre 2025].
- [64] O. Refalo, «Svelte-Splitpanes,» [En línea]. Available: <https://www.npmjs.com/package/svelte-splitpanes>. [Último acceso: 27 octubre 2025].
- [65] D. Gavigan, «The History of Angular,» Medium, 3 abril 2018. [En línea]. Available: <https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7>. [Último acceso: 28 octubre 2025].
- [66] TypeScript, «What is TypeScript?,» [En línea]. Available: <https://www.typescriptlang.org/>. [Último acceso: 28 octubre 2025].
- [67] E. Novoseltseva , «Ventajas de usar Typescript,» ApiumHub, 5 diciembre 2020. [En línea]. Available: <https://apiumhub.com/es/tech-blog-barcelona/usar-typescript/>. [Último acceso: 28 octubre 2025].
- [68] Angular Dev, «The Angular CLI,» [En línea]. Available: <https://angular.dev/tools/cli>. [Último acceso: 28 octubre 2025].
- [69] M. Parada, «Qué es Angular,» 3 mayo 2021. [En línea]. Available: <https://openwebinars.net/blog/que-es-angular-2021/>. [Último acceso: 28 octubre 2025].
- [70] R. Bosarreyes, «¿Qué son los componentes en Angular y cómo se crean?,» Medium, 25 junio 2023. [En línea]. Available: <https://medium.com/@bosarreyesrodrigo/componentes-en-angular-i-63db68231067>. [Último acceso: 28 octubre 2025].
- [71] Git, «1.2 Inicio - Sobre el Control de Versiones - Una breve historia de Git,» [En línea]. Available: <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Una-breve-historia-de-Git>. [Último acceso: 30 octubre 2025].
- [72] Microsoft Learn, «¿Qué es Git?,» 27 octubre 2025. [En línea]. Available: <https://learn.microsoft.com/es-es/devops/develop/git/what-is-git>. [Último acceso: 30 octubre 2025].
- [73] GitLab, «Build software, not toolchains.,» [En línea]. Available: <https://about.gitlab.com/>. [Último acceso: 1 noviembre 2025].
- [74] Formadores IT, «¿Qué es Gitlab y para qué sirve?,» 29 enero 2024. [En línea]. Available: <https://formadoresit.es/que-es-gitlab-y-para-que-sirve/>. [Último acceso: 1 noviembre 2025].
- [75] Stack Overflow, «Dev IDEs,» 2025. [En línea]. Available: <https://survey.stackoverflow.co/2025/technology#1-dev-id-es>. [Último acceso: 1 noviembre 2025].
- [76] Visual Studio Code, «IntelliSense,» [En línea]. Available: <https://code.visualstudio.com/docs/editing/intellisense>. [Último acceso: 1 noviembre 2025].
- [77] F. García de Zúñiga, «¿Qué es Visual Studio Code y cuáles son sus ventajas?,» 17 julio 2025. [En línea]. Available:

<https://www.arsys.es/blog/que-es-visual-studio-code-y-cuales-son-sus-ventajas>. [Último acceso: 1 noviembre 2025].

- [78] G. Cimas Cuadrado, «Visual Studio Code: Editor de código para desarrolladores,» [En línea]. Available: <https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/>. [Último acceso: 1 noviembre 2025].
- [79] G. Blandino, «Figma: qué es y cómo funciona,» Pixartprinting, 4 enero 2023. [En línea]. Available: <https://www.pixartprinting.es/blog/figma-que-es/>. [Último acceso: 30 octubre 2025].
- [80] C. Busquets, «Prototipos de baja y alta fidelidad: diferencias y cuando usar qué,» [En línea]. Available: <https://www.uifrommars.com/prototipos-baja-alta-fidelidad/>. [Último acceso: 30 octubre 2025].
- [81] Figma, «Dev Mode,» [En línea]. Available: <https://www.figma.com/es-es/dev-mode/>. [Último acceso: 30 octubre 2025].
- [82] IEEE, «Especificación de Requisitos según el estándar de IEEE 830,» 22 octubre 2008. [En línea]. Available: <https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf>. [Último acceso: 25 diciembre 2025].
- [83] J. Sauro, «Measuring Usability with the System Usability Scale (SUS),» 3 febrero 2011. [En línea]. Available: <https://measuringu.com/sus/>. [Último acceso: 25 diciembre 2025].
- [84] The UEQ Team, «User Experience Questionnaire,» [En línea]. Available: <https://www.ueq-online.org/>. [Último acceso: 25 diciembre 2025].
- [85] Github, «react-monaco-editor,» [En línea]. Available: <https://github.com/react-monaco-editor/react-monaco-editor>. [Último acceso: 25 diciembre 2025].
- [86] React Dev, «Extracting State Logic into a Reducer,» [En línea]. Available: <https://react.dev/learn/extracting-state-logic-into-a-reducer>. [Último acceso: 25 diciembre 2025].
- [87] React Dev, «useContext,» [En línea]. Available: <https://react.dev/reference/react/useContext>. [Último acceso: 25 diciembre 2025].
- [88] W3C, «Accessible Rich Internet Applications (WAI-ARIA) 1.2,» 6 junio 2023. [En línea]. Available: https://www.w3.org/TR/wai-aria-1.2/#role_. [Último acceso: 25 diciembre 2025].
- [89] UserWay, «Comprobador de contraste de colores,» [En línea]. Available: <https://userway.org/es/comprobador-de-contraste/>. [Último acceso: 25 diciembre 2025].
- [90] M. Borenkraout, «React Testing Library,» 3 junio 2024. [En línea]. Available: <https://testing-library.com/docs/react-testing-library/intro/>. [Último acceso: 25 diciembre 2025].
- [91] chrome for developers, «Introducción a Lighthouse,» 2 junio 2025. [En línea]. Available:

<https://developer.chrome.com/docs/lighthouse/overview?hl=es-419#devtools>. [Último acceso: 25 diciembre 2025].

9 Anexos

9.1 Manual de usuario

La interfaz gráfica se muestra en la Ilustración 21.

La plataforma es compatible con cualquiera de los navegadores principales (Google Chrome, Mozilla Firefox, Microsoft Edge o Safari). En esta sección, se explica cómo interactuar con la interfaz y la función de cada elemento del sistema.

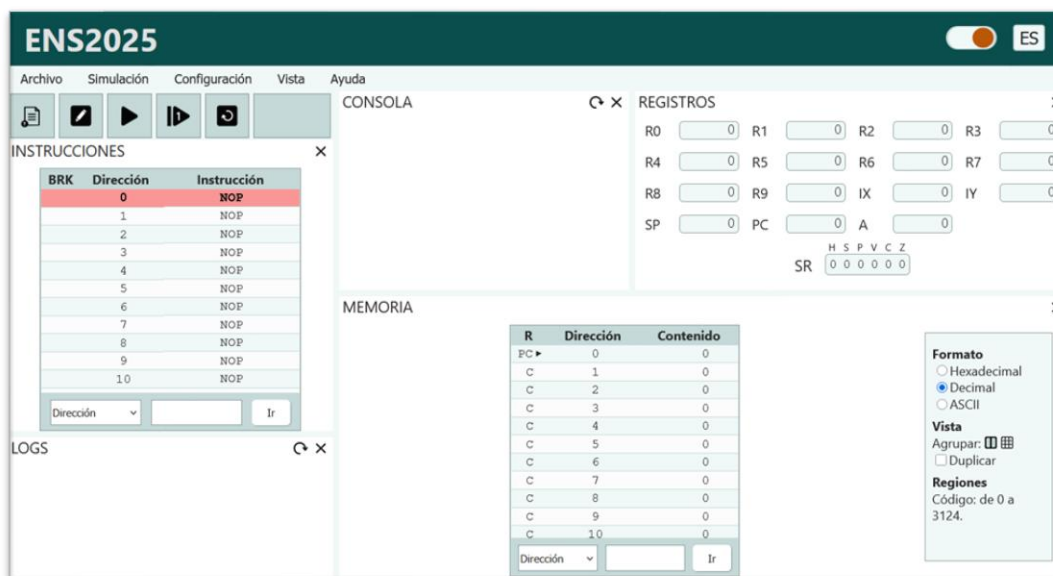


Ilustración 21. Interfaz gráfica del ENS2025.

En la interfaz se pueden distinguir varias secciones con funcionalidades diferentes:

- **Encabezado.** Muestra el título de la página a la izquierda y los botones de selección de tema e idioma a la derecha.
- **Menú de la aplicación.** A través de este menú se pueden realizar la mayoría de las acciones del sistema.
- **Barra de botones.** Contiene botones más directos para la realización de las acciones más comunes.
- **Ventanas de la aplicación:**
 - o Instrucciones. Muestra las instrucciones desensambladas del programa actual. Esta ventana se puede alternar con la ventana del editor.
 - o Editor. Permite observar y editar el código fuente del programa actual.
 - o Logs. Comunica al usuario confirmación de las acciones realizadas, avisos y errores del programa.
 - o Consola. Muestra la salida del programa ejecutado y permite introducir una entrada cuando sea necesario.
 - o Registros. Permite observar y editar el valor de los registros y biestables de la máquina simulada.

- Memoria. Permite observar y editar el contenido de la memoria de la máquina simulada.

A continuación, se explica en más detalle los procedimientos disponibles en cada sección.

9.1.1 Selección de tema e idioma

Los botones situados en el encabezado (Ilustración 22) permiten seleccionar el tema y el idioma de la aplicación.

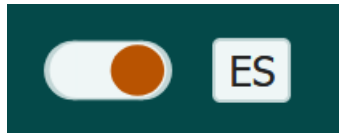


Ilustración 22. Botones de selección de tema e idioma.

El **botón de tema** (izquierda) permite alternar entre el modo claro y oscuro. Su apariencia se modifica dependiendo del modo elegido. Durante el modo claro, se muestra así:



Si se elige el modo oscuro, el botón adquiere esta apariencia:



El tema por defecto de la interfaz depende de las preferencias configuradas previamente en la máquina del usuario.

El **botón de idioma** (derecha) permite alternar entre el idioma español o inglés. El cambio de idioma afecta a todos los elementos de la interfaz: el menú, los botones, el título de las secciones e incluso la descripción interna de los elementos, por lo que también afecta a los usuarios que utilicen productos de apoyo. Por defecto, la aplicación está en español. Este botón también cambia de apariencia según la opción seleccionada, mostrando el texto 'ES' si la interfaz en español y 'EN' si está en inglés.

9.1.2 Menú de la aplicación

El menú de la aplicación (Ilustración 23) divide las funcionalidades en cinco submenús (Archivo, Simulación, Configuración, Vista y Ayuda).

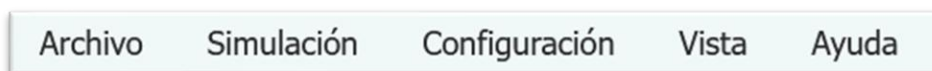


Ilustración 23. Menú de la aplicación.

9.1.2.1 Submenú 'Archivo'

El submenú 'Archivo' ofrece las siguientes funciones:

- **Crear archivo nuevo.** Abre un archivo vacío. Se reinicia el valor de los registros y la memoria.
- **Abrir.** Permite subir un archivo del ordenador. El contenido del archivo es ensamblado y preparado para la simulación. Además, el código fuente es cargado al editor.

- **Descargar archivo.** Descarga en un nuevo archivo el contenido actual del programa.
- **Abrir ejemplo.** Al seleccionar o pasar el ratón por encima de esta opción, se muestra un desplegable de opciones con los nombres de programas de ejemplo del ensamblador. Tras elegir uno, se abre y se ensambla ese ejemplo.



Ilustración 24. Submenú 'Archivo'.

Si se selecciona cualquiera de las opciones que suponen la apertura de un nuevo archivo (Crear archivo nuevo, Abrir, Abrir ejemplo) cuando ya había un archivo abierto y editado, se muestra el menú de la Ilustración 25 antes de realizar la acción correspondiente. Este menú presenta tres opciones:

- 'Descargar y continuar'. Descarga el archivo modificado antes de abrir el siguiente.
- 'Descartar cambios y continuar'. Descarta los cambios del archivo modificado y abre el siguiente.
- 'Cancelar'. Cierra este menú y no realiza ninguna acción.

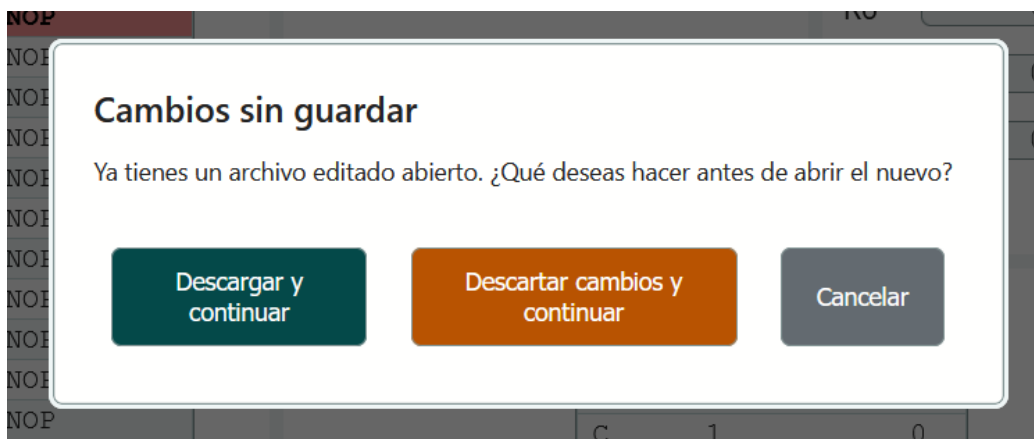


Ilustración 25. Menú de conflicto de archivos.

9.1.2.2 Submenú 'Simulación'

El submenú 'Simulación' (Ilustración 26) ofrece las siguientes funciones:

- **Ensamblar programa.** Reensambla el programa actual.
- **Ejecutar programa.** Ejecuta el programa hasta el final o hasta el siguiente punto de ruptura. Tras la finalización de la ejecución, se actualiza la memoria, los registros y la consola y se muestran en la ventana de *logs* los mensajes relacionados con esa ejecución. Si, durante la ejecución, el programa requiere de entrada del usuario, se interrumpe la ejecución y se muestra la ventana de entrada, hasta que el usuario

introduzca una entrada y continúe la ejecución. Por otro lado, si se ha modificado el programa desde el editor, éste se reensambla antes de ejecutar, a menos que la ejecución ya esté en marcha (se haya ejecutado ya al menos una instrucción y aún no se haya alcanzado el fin del programa), en cuyo caso se debe reiniciar la simulación si se desea reensamblar.

- **Ejecutar una instrucción.** Ejecuta la siguiente instrucción. El comportamiento es homólogo a la acción de 'Ejecutar programa'.
- **Reiniciar simulación.** Reinicia la memoria y los registros al estado que tenían tras el ensamblado del programa. **Es necesario reiniciar la simulación tras cada ejecución para volver a ejecutar.**
- **Abrir o cerrar editor.** Alterna entre las ventanas de 'Instrucciones' y 'Editor'.

Simulación	Confi
Ensamblar programa	
Ejecutar programa	
Ejecutar una instrucción	
Reiniciar simulación	
Abrir o cerrar editor	

Ilustración 26. Submenú 'Simulación'.

9.1.2.3 Submenú 'Configuración'

El submenú 'Configuración' permite configurar la interfaz a través de las siguientes opciones:

- **Formato de las instrucciones,** a elegir entre hexadecimal y decimal. Esta configuración afecta a las columnas de 'Dirección' en las tablas de instrucciones y memoria. El valor por defecto es decimal.
- **Formato del contenido,** a elegir entre hexadecimal, decimal y ASCII. Esta configuración afecta al contenido de los registros, la columna de 'Contenido' en la tabla de memoria y a los números y direcciones de las instrucciones desensambladas. Sin embargo, la opción de ASCII sólo es visible en la tabla de memoria, mientras que si está seleccionada el resto de regiones afectadas toman el valor del formato de las instrucciones. El valor por defecto es decimal.
- **Dirección de la pila,** a elegir entre ascendente o descendente. Si es ascendente, la pila crece hacia direcciones mayores, mientras que si es descendente crece hacia direcciones menores. El valor por defecto es descendente.
- **Tema,** a elegir entre tema claro y tema oscuro.
- **Idioma,** a elegir entre español e inglés.

Configuración	Vi
Formato de instrucciones	►
Formato del contenido	►
Dirección de la pila	►
Tema	►
Idioma	►

Ilustración 27. Submenú 'Configuración'.

Para cada opción, aparece un desplegable con sus subopciones, donde se marca con un punto la opción actualmente seleccionada, al igual que la opción 'Decimal' en la Ilustración 28.

Hexadecimal
● Decimal
ASCII

Ilustración 28. Subopciones de la opción 'Formato del contenido' en el submenú 'Configuración'.

Durante la ejecución de un programa no se puede modificar la dirección de la pila. Esta opción se mostrará desactivada y sin posibilidad de abrir su menú lateral, como en la Ilustración 29. Para volver a activarla hay que reiniciar la simulación.

Configuración	Vi
Formato de instrucciones	►
Formato del contenido	►
Dirección de la pila	►

Ilustración 29. Submenú 'Configuración' con 'Dirección de la pila' desactivado.

9.1.2.4 Submenú 'Vista'

El submenú 'Vista' (Ilustración 30) contiene las siguientes opciones:

- **Mostrar ventana.** Permite abrir una ventana cerrada o cerrar una ventana abierta. Las ventanas abiertas se marcan con un *tick*. La opción 'Instrucciones' representa tanto la ventana de instrucciones como la ventana del editor.

- **Duplicar tabla de memoria.** Muestra dos tablas de memoria independientes en la ventana de memoria (Ilustración 43).
- **Expandir columnas de la tabla de memoria.** Alterna entre dos formatos de tabla: una dirección por cada fila o tantas direcciones por fila como quepan cómodamente en la ventana.

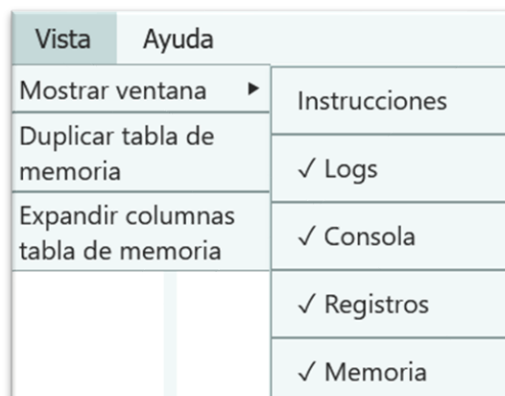


Ilustración 30. Submenú 'Vista' con opciones de 'Mostrar ventana' desplegadas.

9.1.2.5 Submenú 'Ayuda'

El submenú 'Ayuda' solo contiene una opción, **Abrir manual de usuario**, que descarga este documento.

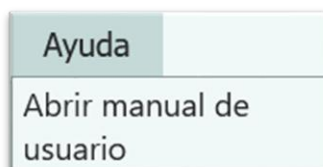


Ilustración 31. Submenú 'Ayuda'.

9.1.3 Barra de botones

La barra de botones (Ilustración 32) contiene botones directos a las acciones más comunes de la interfaz. Al hacer *hover* en cada botón se muestra su descripción.



Ilustración 32. Barra de botones.

De izquierda a derecha, se muestran los siguientes botones:

- **Abrir.** Equivalente al botón de abrir del submenú 'Archivo'.
- **Editar.** Alterna entre la ventana del editor y de instrucciones, equivalente al botón 'Abrir o cerrar editor' del submenú 'Simulación'.
- **Ejecutar programa.** Equivalente al botón de ejecutar programa del submenú 'Simulación'.
- **Ejecutar una instrucción.** Equivalente al botón de ejecutar una instrucción del submenú 'Simulación'.
- **Reiniciar simulación.** Equivalente al botón de reiniciar la simulación del submenú 'Simulación'.

9.1.4 Ventanas de la aplicación

Todas las ventanas se pueden cerrar seleccionando la cruz situada en la esquina superior derecha. Se pueden volver a abrir a través del submenú 'Vista'.

9.1.4.1 Ventana de instrucciones

La ventana de instrucciones (Ilustración 21) muestra la tabla de desensamblado de la memoria. Esta tabla tiene tres columnas: la columna de puntos de ruptura, de direcciones y de instrucciones desensambladas. Al no conocer el comienzo de las instrucciones, es posible que las primeras instrucciones desensambladas muestren datos incoherentes, al igual que el desensamblado de las secciones de datos. Si se encuentra un valor de memoria que no es una instrucción válida, en esa dirección aparece el texto ****INSTRUCCIÓN NO IMPLEMENTADA****.

INSTRUCCIONES		
BRK	Dirección	Instrucción
	0	INCHAR /4096
	2	WRCHAR /4096
	4	HALT
	5	NOP
	6	NOP
	7	NOP
	8	NOP
	9	NOP
	10	NOP
	11	NOP
	12	NOP

Ilustración 33. Ventana de instrucciones.

Para navegar por la tabla se puede subir y bajar con la barra de desplazamiento que aparece en el lateral al situarse en la tabla o a través del menú de navegación inferior, que se explica con más detalle en la sección ‘Menú de navegación de tablas’.

Para colocar o eliminar un punto de ruptura, se debe seleccionar la casilla de la columna 'BRK' de la dirección adecuada. Los puntos de ruptura se indican con un punto de color, como el que se muestra a continuación:

•	4	OR .R3, .R8
---	---	-------------

Además, la fila de la instrucción a la que apunta el PC aparece resaltada con un fondo de distinto color y con el texto en negrita.

9.1.4.2 Editor

La ventana del editor (Ilustración 34) muestra el código fuente del programa subido. Este código se puede editar, y será reensamblado automáticamente al ejecutar.

```

EDITOR
1  ;Lectura de dos cadenas, concatenar y dar la vue
2  ;El programa pide al usuario que introduzca dos
3  ;A continuacion, las concatena y muestra el resu
4  ;derecho y al revés
5
6  ;leer cadena1
7  WRSTR /mens1
8  INSTR /cadena1
9  ;leer cadena2
10 WRSTR /mens2
11 INSTR /cadena2
12 ;concatenar
13 MOVE #cadena1,.R1
14 MOVE #cadena2,.R2
15 MOVE #resu1,.R0
16 MOVE #resu2,.R3
17 cad1: CMP [.R1],#0 ; fin de cadena?
18 BZ cad2 ; pasamos a la siguiente

```

Ilustración 34. Ventana del editor.

El editor ofrece muchas acciones mediante atajos de teclado para agilizar la edición del código. Estos atajos son equivalentes a los atajos por defecto en el editor de código Visual Studio Code. Las acciones también se pueden realizar manualmente a través del menú flotante (Ilustración 36) que aparece al hacer clic derecho sobre el editor o pulsar ‘F1’ (Ilustración 35) y seleccionar ‘Paleta de comandos’.

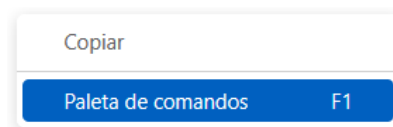


Ilustración 35. Menú flotante del editor.

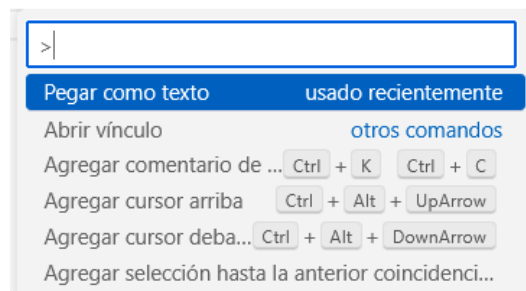


Ilustración 36. Paleta de comandos.

9.1.4.3 Ventana de logs

La ventana de logs (Ilustración 37) muestra tres tipos de mensajes:

- Mensajes de información: comienzan por la palabra INFO; confirman la correcta realización de las acciones del usuario.
- Mensajes de aviso: comienzan por la palabra AVISO y tienen color naranja; notifican situaciones que, sin ser errores críticos, requieren atención o podrían derivar en problemas con el programa o las acciones del usuario.
- Mensajes de error: comienzan por la palabra ERROR y tienen color rojo; comunican problemas del sistema, acciones que no se han podido realizar, excepciones de la máquina y errores de ensamblado.

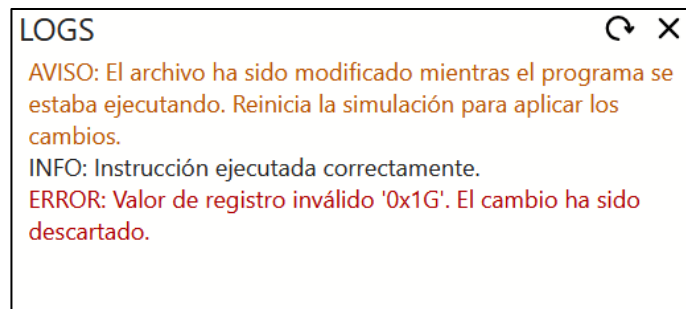


Ilustración 37. Ventana de logs.

Para limpiar la ventana de *logs* se puede activar el botón  de la esquina superior derecha.

9.1.4.4 Consola

La ventana de la consola (Ilustración 38) permite comunicarse con el módulo de entrada y salida de la máquina simulada. Tras cada ejecución, el contenido de las instrucciones de escritura aparecerá en la consola. Una línea llena de guiones ('-----') separa la salida de cada ejecución del programa. Para limpiar la salida de la consola se puede activar el botón  de la esquina superior derecha.

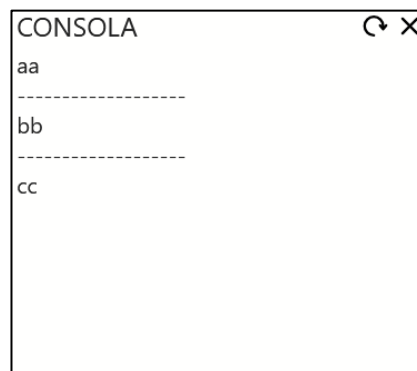


Ilustración 38. Ventana de consola.

Cuando el programa solicita una entrada, en la ventana de consola aparece una sección que indica el tipo de entrada a introducir (cadena, carácter o número) (Ilustración 39). Se debe escribir la entrada deseada en la caja dedicada para ello y pulsar la tecla 'Enter' o seleccionar el botón 'Introducir'. El programa quedará parado hasta que se introduzca la entrada, y retomará la ejecución automáticamente después.

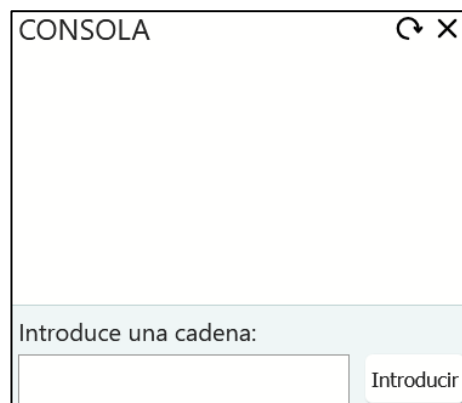


Ilustración 39. Ventana de consola con solicitud de entrada.

9.1.4.5 Ventana de registros

La ventana de registros (Ilustración 40) permite observar y editar el contenido de los registros y biestables de la máquina simulada.

Para editar un registro, se debe acceder a él, introducir su nuevo valor y salir del registro o pulsar 'Enter' para confirmar el cambio. Independientemente del formato seleccionado en ese momento, los registros aceptan números enteros decimales (sin prefijo) y hexadecimales (con el prefijo '0x') dentro del rango aceptable. Los biestables también son editables de la misma manera y aceptan los valores 0 o 1.

La ventana 'REGISTROS' contiene los siguientes campos:

Registro	Valor
R0	0
R1	0
R2	0
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
IX	0
IY	0
SP	0
PC	0
A	0

Los biestables SR (Status Register) tienen los siguientes valores:

Bit	Valor
H	0
S	0
P	0
V	0
C	0
Z	0

Ilustración 40. Ventana de registros.

9.1.4.6 Ventana de memoria

La ventana de memoria (Ilustración 41) está formada por la tabla de memoria y un menú lateral con opciones relevantes para esta ventana.

La ventana 'MEMORIA' contiene la siguiente tabla:

R	Dirección	Contenido
PC ▶	0	0
C	1	0
C	2	0
C	3	0
C	4	0
C	5	0
C	6	0
C	7	0
C	8	0

El menú lateral incluye:

- Formato:** ☐ Hexadecimal, ☒ Decimal, ☐ ASCII
- Vista:** Agrupar: ☒ ☐ Duplicar: ☐
- Regiones:** Código: de 0 a 3124.

Ilustración 41. Ventana de memoria.

La tabla de memoria tiene tres columnas: R (región), dirección y contenido. La región indica con una C aquellas direcciones que pertenezcan a la región de código. Además, contiene indicadores que apuntan a la dirección contenida en los registros IX, IY, PC o SP. El contenido de cada dirección es editable activando la celda correspondiente, introduciendo el valor nuevo y saliendo de la celda o pulsando 'Enter' para confirmar el cambio. Si el formato seleccionado es hexadecimal o decimal, las celdas aceptan números enteros decimales (sin prefijo) y hexadecimales (con el prefijo '0x') dentro del rango aceptable.

Si el formato seleccionado es ASCII, si el contenido de una dirección coincide con el código ASCII de un carácter imprimible, se mostrará ese carácter, y en caso contrario se mostrará la cadena '**', que indica que es un carácter no representable. Al editar una celda en este formato, se puede introducir

únicamente un carácter imprimible, y adquirirá el valor de su código ASCII correspondiente.

Para navegar por la tabla se puede subir y bajar con la barra de desplazamiento que aparece en el lateral al entrar en la tabla o a través del menú de navegación inferior, que se explica con más detalle en la sección ‘Menú de navegación de tablas’.

Respecto al menú lateral, está dividido en tres secciones:

- **Formato.** Permite cambiar el formato globalmente entre hexadecimal, decimal y ASCII. Cambia tanto el formato de las instrucciones como el del contenido, a menos que se escoja la opción ASCII, en cuyo caso sólo se modifica el formato del contenido.
- **Vista.** Contiene dos opciones de configuración:
 - o **Agrupar.** Alterna entre dos formatos de tabla: una dirección por cada fila o tantas direcciones por fila como quepan cómodamente en la ventana (Ilustración 42). Equivalente al botón ‘Expandir columnas de la tabla de memoria’ del submenú ‘Vista’.
 - o **Duplicar.** Muestra dos tablas de memoria independientes en lugar de una (Ilustración 43). Las tablas pueden mostrar regiones diferentes, pero los cambios de la memoria se sincronizarán instantáneamente en ambas tablas. Equivalente al botón ‘Duplicar tabla de memoria’ del submenú ‘Vista’.
- **Regiones.** Muestra la dirección inicial y final de la región de código.

R	Dirección	Contenido							
PC ▶	0	0	0	0	0	0	0	0	0
C	7	0	0	0	0	0	0	0	0
C	14	0	0	0	0	0	0	0	0
C	21	0	0	0	0	0	0	0	0
C	28	0	0	0	0	0	0	0	0
C	35	0	0	0	0	0	0	0	0
C	42	0	0	0	0	0	0	0	0
C	49	0	0	0	0	0	0	0	0
C	56	0	0	0	0	0	0	0	0

Ilustración 42. Ventana de la memoria con la tabla de memoria extendida.

R	Dirección	Contenido
PC ▶	0	0
C	1	0
C	2	0
C	3	0
C	4	0
C	5	0
C	6	0
C	7	0
C	8	0

R	Dirección	Contenido
C	2000	0
C	2001	0
C	2002	0
C	2003	0
C	2004	0
C	2005	0
C	2006	0
C	2007	0
C	2008	0

Ilustración 43. Ventana de memoria con tablas de memoria duplicadas.

9.1.4.7 Menú de navegación de tablas

El menú de navegación de tablas (Ilustración 44) se encuentra situado en la parte inferior de las tablas de instrucciones y de memoria y permite acceder a una dirección a partir de su número, del valor de un registro índice o del nombre de una etiqueta.

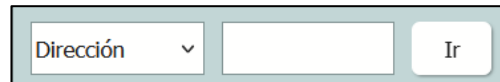


Ilustración 44. Menú de navegación de tablas.

Este menú de control consta de dos campos interactivos. El primero (Ilustración 45) permite seleccionar el Modo de navegación, el cual condiciona la función del segundo campo según la opción elegida:

- Dirección: Permite introducir manualmente una dirección en el segundo campo. Se admiten formatos decimal o hexadecimal (usando el prefijo '0x'). Para ejecutar la navegación, pulse la tecla 'Enter' o el botón 'Ir'.
- Registro: Navega hacia la dirección almacenada en un registro específico. Al seleccionar este modo, el segundo campo se convierte en un desplegable con cuatro opciones: SP, PC, IX e IY. Se debe seleccionar el registro deseado y pulsar 'Ir'.
- Etiqueta: Permite saltar a una dirección asociada a un nombre simbólico. Se debe seleccionar la etiqueta correspondiente en el desplegable del segundo campo y pulsar 'Ir' para desplazarse a la posición de memoria traducida.

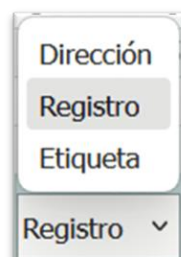


Ilustración 45. Primer campo del menú de navegación de tablas.

9.1.5 Navegación por teclado

Aunque la interfaz sigue los patrones de navegación estándar, se han implementado atajos específicos para agilizar el desplazamiento entre los bloques principales.

En las secciones de **Editor**, **Registros**, **Tabla de Memoria** y **Tabla de Instrucciones**, se ha modificado el comportamiento de la tecla 'Tab' para evitar recorridos innecesarios (como saltar celda por celda en la memoria o insertar tabulaciones en el editor). El funcionamiento es el siguiente:

- **Navegación Rápida (Saltar secciones):** Al pulsar 'Tab', el foco se situará sobre la sección completa. Si se pulsa 'Tab' nuevamente, saltará directamente al siguiente bloque de la interfaz, ignorando el contenido interno de la sección actual.

- **Navegación Detallada (Acceder al contenido):** Para interactuar con los elementos internos de una sección (un registro específico, una línea de código o una dirección de memoria):
 1. Sitúe el foco en la sección deseada.
 2. Pulse la tecla 'Enter' para entrar. A partir de este momento, podrá usar 'Tab' y 'Shift + Tab' para navegar por su interior.

Para salir del modo detallado y volver a la navegación general, pulse la tecla 'Esc'. El foco regresará al contenedor principal.

Este documento esta firmado por



Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
Fecha/Hora	Mon Jan 12 18:38:55 CET 2026
Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
Numero de Serie	561
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)