Revista RITI
Investigación en Tecnologías de la Información

Revista de Investigación en Tecnologías de la Información
ISSN: 2387-0893
revista.riti@gmail.com
Universitat Politècnica de Catalunya
España

Arellano Pimentel, J. Jesús; Solar González, Rocío;
Nieva García, Omar S.; Canedo Ibarra, Sabrina Patricia

Compilador e intérprete en línea de diagramas de flujo con fines didácticos

Revista de Investigación en Tecnologías de la Información,
vol. 10, núm. 20, 2022, Enero-Junio, pp. 80-94
Universitat Politècnica de Catalunya
España

DOI: https://doi.org/10.36825/RITI.10.20.007

Compilador e intérprete en línea de diagramas de flujo con fines didácticos

Online flowchart compiler and interpreter for educational purposes

J. Jesús Arellano Pimentel
Universidad del Istmo, campus Tehuantepec, Sto. Domingo Tehuantepec, México
jjap@sandunga.unistmo.edu.mx
ORCID: 0000-0003-0609-9470

Rocío Solar González
Universidad del Istmo, campus Tehuantepec, Sto. Domingo Tehuantepec, México
solgr@sandunga.unistmo.edu.mx

Omar S. Nieva García
Universidad del Istmo, campus Tehuantepec, Sto. Domingo Tehuantepec, México
omarg2017@mail.com
ORCID: 0000-0003-0577-2838

Sabrina Patricia Canedo Ibarra
Universidad Virtual del Estado de Michoacán, Morelia, Michoacán, México
spcanedo@univim.edu.mx
ORCID: 0000-0002-6570-4979

doi: https://doi.org/10.36825/RITI.10.20.007

Recibido: Enero 04, 2022
Aceptado: Abril 02, 2022

Resumen: En este trabajo se aborda el desarrollo, puesta a punto y pruebas de un compilador e intérprete en línea de diagramas de flujo. La metodología de desarrollo toma como marco de referencia el modelo de proceso de reingeniería de software, en conjunto con el método de diseño de hipermedios orientado a objetos (MDHOO). El producto de software obtenido es una aplicación web con fines didácticos que se distingue de otras herramientas principalmente por tres aspectos: dar soporte para la heurística de resolución de problemas de Polya, crear diagramas de flujo atendiendo las recomendaciones para la simbología gráfica del American National Standards Institute (ANSI), y estar disponible en internet de forma libre. La puesta a punto del compilador e intérprete en línea se ejecutó en dos navegadores web compatibles con cuadros de diálogo modales, uno para computadoras personales y otro para dispositivos móviles. Además, se realizaron pruebas de cumplimiento de estándares de la W3C, pruebas de velocidad de carga, y se aplicó una prueba de percepción de experiencia de usuario a 22 estudiantes de un curso propedéutico de algoritmos. Los resultados obtenidos en todas las pruebas realizadas se consideran satisfactorios y acordes al contexto actual de los estudiantes.

Palabras clave: *Aplicación Web, Compiladores e Intérpretes, Desarrollo de Software, Enseñanza de Algoritmos, Heurística de Polya.*

Abstract: This work deals with the development, fine-tuning and testing of an online flowchart compiler and interpreter. The development methodology takes as a frame of reference the software reengineering process model, in conjunction with the Object-Oriented Hypermedia Design Method (OOHDM). The software product obtained is a web application for educational purposes that differs from other tools mainly in three aspects: supporting the Polya problem-solving heuristics, create flowcharts following the recommendations for graphic symbology of the American National Standards Institute (ANSI), and be freely available on the internet. The online compiler and interpreter fine-tuning ran on two web browsers that support modal dialog boxes, one for personal computers and one for mobile devices. In addition, compliance tests with the W3C standards, loading speed tests, and a user experience perception test were applied to 22 students of a preparatory course on algorithms. The results obtained in all the tests carried out are considered satisfactory and in accordance with the current context of the students.

Keywords: *Web Application, Compilers and Interpreters, Software Development, Algorithm Teaching, Polya Heuristics.*

1. Introducción

La construcción de herramientas visuales orientadas a la enseñanza de la programación tiene uno de sus primeros registros en el trabajo de Göktepe, Özgüc y Baray [1], desde entonces a la fecha los enormes avances tecnológicos de hardware, software y conectividad han generado un nuevo perfil de usuario-alumno habituados a interactuar con múltiples dispositivos con interfaces gráficas muy intuitivas, consumidores habidos de información visual disponible en línea a la que ingresan a través de sus teléfonos inteligentes o sus equipos de cómputo personal. De acuerdo con el INEGI [2] en México, durante el año 2020, el 72.0% de la población de seis años o más son usuarios de internet, siendo los grupos de edad de 12 a 17 años y de 18 a 24 años donde el uso del internet está más generalizado. Y aunque las cohortes generacionales no son una ciencia exacta [3], se podría decir que ahora los estudiantes universitarios, en el rango de edad de 17 hasta los 22 años, pertenecen a la generación denominada como Centennials, I-GEN, D-GEN o Generación Z [4].

Un rasgo distintivo de la Generación Z es haber nacido en la era del internet, con un ecosistema repleto de dispositivos electrónicos donde el teléfono inteligente sobresale al permitir una conectividad permanente y ubicua, es decir, en cualquier momento y en cualquier lugar, esto tiene una implicación directa en cómo y con qué aprenden los alumnos actualmente [5]. Pero cuando se trata de la formación profesional el teléfono inteligente no es el único dispositivo al que recurren los estudiantes, también se apoyan fuertemente en equipos de cómputo personal. Para aquellos que estudian en línea, o se vieron obligados a continuar en la modalidad a distancia a causa de la pandemia por la COVID-19, utilizan más la computadora portátil que el teléfono inteligente, esto según los datos reportados por la Asociación Mexicana de Internet [6], donde previo a la pandemia el 74% utilizaba la computadora portátil frente al 63% del uso del teléfono inteligente; durante la pandemia los porcentajes crecieron a 77% y 64% respectivamente. Además, según el INEGI [2] los equipos de conexión más utilizados en México durante el 2020 fueron los teléfonos inteligentes (96.0%) y las computadoras portátiles (33.7%).

Dado lo anterior resulta indudable pensar en las aplicaciones en línea o web de diseño adaptable o responsivo (se ajusta a las dimensiones del dispositivo de despliegue) como una excelente alternativa de herramientas didácticas, capaces de dar respuesta al contexto actual de los estudiantes que requieren ejecutar los entornos de aprendizaje tanto en sus equipos de cómputo personal como en sus teléfonos inteligentes. Sin embargo, en su gran mayoría las herramientas de programación visual con soporte para diagramas de flujo abordadas en publicaciones científicas son aplicaciones de escritorio [7]. Dichas herramientas se suelen utilizar en cursos introductorios de programación a nivel universitario o preuniversitario de manera frecuente [8-13]. En cuanto a herramientas en línea con soporte para diagramas de flujo orientados a la lógica de programación son muy escasas, y máxime en el idioma español, en este trabajo se logró localizar la propuesta, en español, de Vázquez y Jaimez [12]; en inglés están las propuestas de Allen y Vahid [13] y Supaartagorn [14].

A decir de Hooshyar [7], el uso de herramientas didácticas con soporte para diagramas de flujo en cursos introductorios de programación permite a los estudiantes: a) incrementar su habilidad de resolver problemas computables, b) mejorar significativamente su lógica de programación, y c) escribir código. En parte esto se debe a que los diagramas de flujo proveen un mejor modelo para la representación del proceso de un algoritmo que una representación textual, lo cual los convierte en un andamiaje efectivo para aprender un lenguaje de programación formal [15]. Además, desde la perspectiva pedagógica del andamiaje progresivo, los diagramas de flujo son una

excelente ayuda tanto para los profesores como para los estudiantes, ya que permiten una rápida retroalimentación para detectar y eliminar impedimentos cognitivos en el desarrollo de habilidades para la solución de problemas y el aprendizaje de la programación [16].

Debido a la relevancia del uso de los diagramas de flujo como apoyo al aprendizaje de la lógica de programación, así como las escasas de herramientas didácticas en línea con soporte para crear, interpretar y traducir diagramas de flujo, en este trabajo se presenta la metodología y resultados de un compilador e intérprete en línea de diagramas de flujo llamado Entorno para el Aprendizaje de Algoritmos (EpAA). Básicamente se trata de una aplicación en línea construida a partir de un proceso de reingeniería de software [17] respecto de una aplicación de escritorio llamada Software para la Asistencia en el Aprendizaje de Algoritmos, abreviada SAAA [18], pero tomando como marco de referencia en la fase de ingeniería hacia adelante el método de diseño de hipermedios orientado a objetos, abreviado MDHOO [17]. Un aspecto importante retomado del SAAA es el soporte brindado para las cuatro etapas de la heurística de resolución de problemas de G. Polya [19], además de estar diseñada para estudiantes que hablan español como lengua materna.

2. Metodología

2.1. Proceso de reingeniería y MDHOO

La reingeniería es un proceso de reconstrucción, en este caso, se aplicó para reconstruir una aplicación de escritorio en una aplicación en línea. Pressman [17] presenta un modelo de proceso de reingeniería de software como un modelo cíclico que involucra seis actividades: análisis de inventarios, reestructuración de documentos, ingeniería inversa, reestructuración de código, reestructuración de datos, e ingeniería hacia adelante (Figura 1). Es posible que estas seis actividades ocurran de forma secuencial, pero también, puede ser que para alguna iteración del ciclo ciertas actividades no sean necesarias, esto dependerá de los elementos que se estén reconstruyendo. Dado que la aplicación a reconstruir tiene soporte para las cuatro etapas de la heurística de resolución de problemas de Polya [19], también fueron necesarias cuatro iteraciones sobre el proceso de reingeniería, una por cada etapa, además de una iteración adicional para las funcionalidades básicas de toda aplicación: nuevo, abrir y guardar.

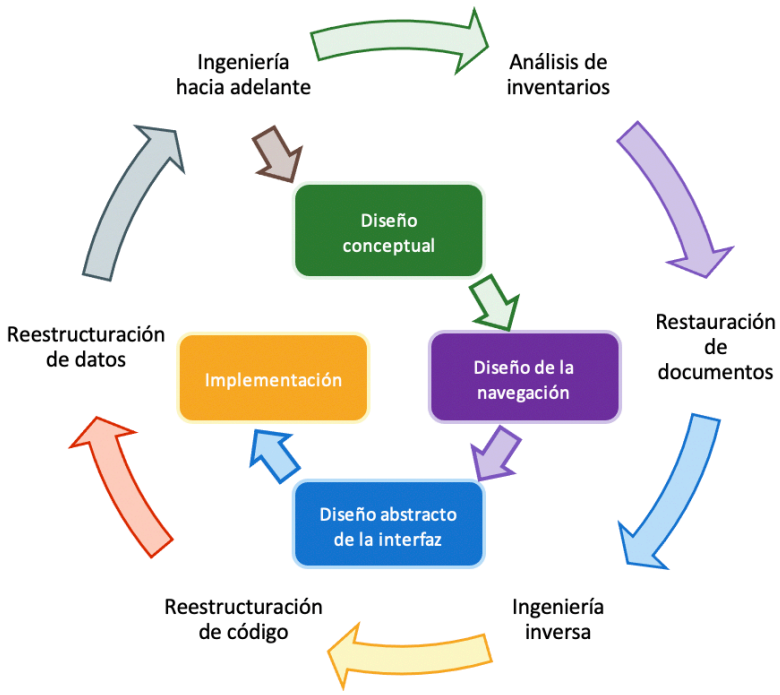


Figura 1. Modelo de proceso de reingeniería de software y MDHOO.
Fuente: Adaptado de Pressman [17].

En la primera iteración del modelo, durante la actividad de análisis de inventarios, se identificó la aplicación de escritorio SAAA como un candidato idóneo para reconstruir como una aplicación en línea o web. El resto de las iteraciones solo consideran las otras cinco actividades. La actividad de reestructuración de documentos requirió, por un lado, hacer cambios puntuales en documentos existentes, por otro lado, reconstruir con base en la nueva tecnología empleada. Durante la actividad de ingeniería inversa se analizaron algunos bloques de código no documentados a profundidad a fin de comprender por completo qué hacían, con qué lo hacían, y cómo lo hacían. En cuanto a la actividad de reestructuración de código, inherentemente necesaria, se reescribió la codificación escrita originalmente en Java a JavaScript, pero conservando prácticamente intacta la arquitectura de software original con los principales módulos y técnicas asociadas a cada etapa de la heurística de Polya (Figura 2). El módulo del compilador aplica la técnica de análisis por descenso recursivo [20] con una gramática para cada elemento del diagrama de flujo. El módulo del intérprete usa un esquema de traducción dirigida por la sintaxis [21] con gramáticas atribuidas en cada elemento del diagrama de flujo.

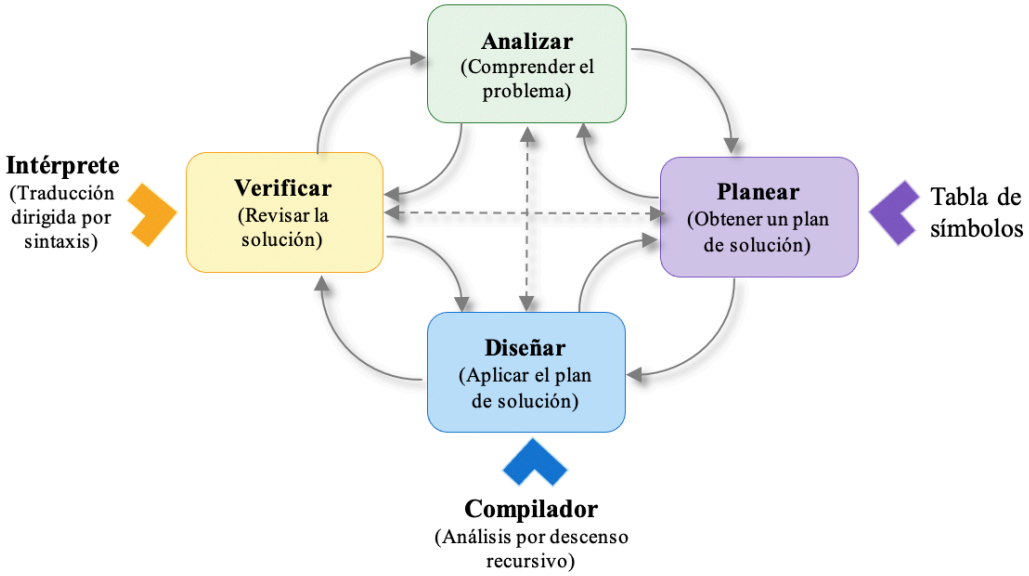


Figura 2. Módulos y técnicas en la arquitectura del software asociadas con las etapas de la heurística de Polya.

Durante la actividad de reestructuración de datos se migró de las estructuras de datos dinámicas de Java a las propias de JavaScript conservando en lo general el modelo abstracto de los datos; uno de los principales módulos reestructurados fue la tabla de símbolos. Finalmente, en la actividad de ingeniería hacia adelante es donde se aplicó el método de diseño de hipermedios orientado a objetos (Figura 1), que de acuerdo con Pressman [17] se compone de cuatro actividades: diseño conceptual, diseño de navegación, diseño abstracto de la interfaz, e implementación. En este caso, el diseño conceptual parte de los productos derivados de las actividades del proceso de reingeniería de software, considerando la semántica de modelado del dominio de la aplicación, las clases, subsistemas, relaciones, composiciones, agregaciones, atributos, etc. En el diseño de la navegación se tuvo como eje principal la heurística de resolución de problemas de Polya con sus cuatro etapas, así es posible navegar de forma secuencial entre las etapas, pero también retroceder y pasar hacia cualquiera de ellas en cualquier momento (Figura 2). La navegación también cuenta con vínculos hacía plataformas de apoyo, por ejemplo, un canal de *YouTube* y un *Blogger* con información sobre la temática de algoritmos y el uso de la aplicación en línea.

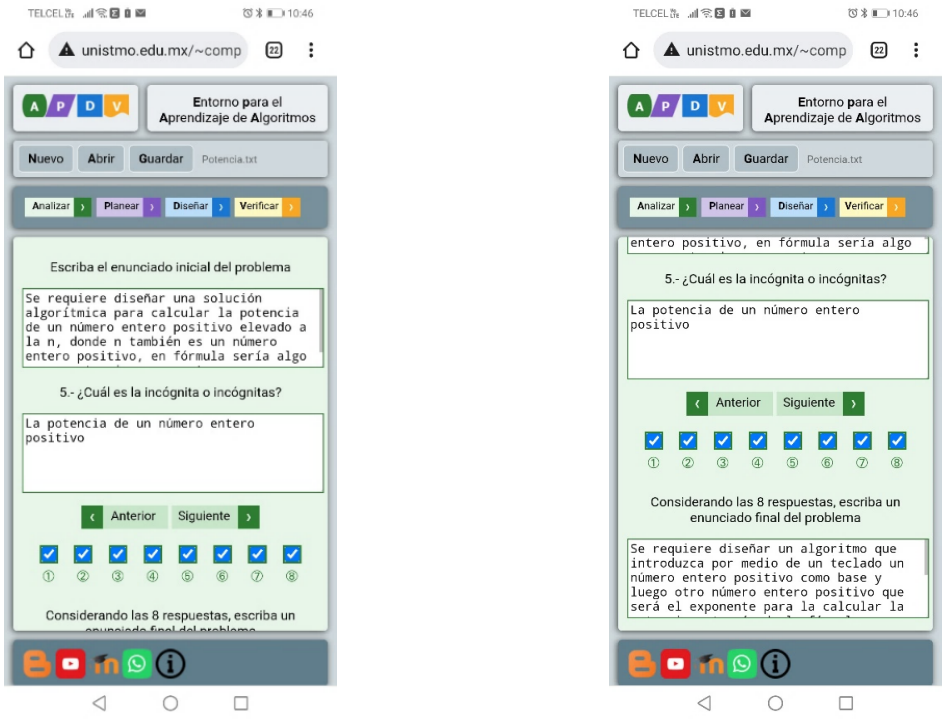
Respecto al diseño abstracto de la interfaz, se tomó en cuenta el despliegue de cada etapa de la heurística de Polya tanto en dispositivos móviles como en computadoras de escritorio, manteniendo visible en todo momento las etiquetas atribuidas a cada etapa: Analizar, Planear, Diseñar y Verificar. El producto de software resultante de la actividad de implementación cumple con buena parte de las características presentes en herramientas de escritorio similares [7], [22], entre las que están: 1) generación automática de código a partir del diagrama de flujo, 2) visualización interactiva de la ejecución del diagrama de flujo, 3) notificación de errores y advertencias, 4) soporte para tipos primitivos de variables, 5) soporte para estructuras secuenciales, 6) soporte para estructuras de selección simple y compuesta, 7) soporte para estructuras iterativas, 8) reglas estructurales al construir el diagrama

de flujo, 9) uso de color para diferenciar el tipo de elemento estructural en el diagrama de flujo, y 10) versión en el idioma español. Todas estas características soportadas permiten hacer frente a las dificultades comúnmente identificadas en cursos introductorios o básicos de programación [23], [24], las cuales a su vez se han abordado empleando aplicaciones con soporte para diagramas de flujo [8-13].

2.2. Puesta a punto de la aplicación en línea propuesta

El compilador e intérprete de diagramas de fujo se desarrolló empleando HTML5 (Lenguaje de Marcación de Hipertexto, del inglés *HyperText Markup Language*), CSS3 (Hojas de Estilo en Cascada, del inglés *Cascading Style Sheets*) y JavaScript [25], dando como resultado una aplicación en línea que se ejecuta completamente del lado del cliente, es decir, en el navegador web que accede al enlace donde se aloja la aplicación. Los ajustes y depuración para la puesta a punto se realizaron utilizando las herramientas de desarrollo del navegador web Edge, para posteriormente probar en el navegador Chrome en dispositivos móviles con Android (tableta electrónica y teléfono inteligente). A manera de ejemplo de la puesta a punto de la herramienta se presentan la solución al problema de calcular la potencia de un número entero positivo elevado a la *n*, donde *n* también es un entero positivo.

Para una mejor visualización y demostración de la propiedad responsiva de la aplicación en línea, las etapas Analizar y Diseñar se presentan desde un teléfono inteligente, mientras que las etapas Planear y Verificar se presentan desde la pantalla de una computadora personal. Durante la etapa Analizar (Figura 3) se debe escribir el enunciado inicial del problema, y después resolver una serie de 8 preguntas para lograr una comprensión más profunda del mismo, dichas preguntas son: 1) ¿Cuáles son las fórmulas o datos iniciales?, 2) ¿Cuáles son los datos requeridos o a solicitar?, 3) ¿A través de qué medio se proporcionan los datos?, 4) ¿Cuáles son los supuestos?, 5) ¿Cuál es la incógnita o incógnitas?, 6) ¿Qué es lo que se quiere resolver o calcular?, 7) ¿Qué información ha de presentarse como resultado?, 8) ¿A través de qué medio se presentarán los resultados?. Esta etapa concluye reescribiendo el enunciado final del problema tomando en cuenta las 8 respuestas a las preguntas.



(**a**) (**b**)

Figura 3. Pantalla Analizar ejecutándose en el navegador de un dispositivo móvil.: (a) área para la edición de enunciado inicial del problema; (b) área para la edición del enunciado final del problema considerando las 8 respuestas a las preguntas previas.

En la etapa Planear (Figura 4) se crea la lista de elementos que habrán de intervenir en la solución algorítmica, estos elementos surgen principalmente de las respuestas las preguntas 1, 2, 5 y 6 de la etapa Analizar. A cada elemento se le asigna un rol de entrada, salida o auxiliar, los cuales permiten hacer validaciones semánticas durante

la compilación. Además, también se les asigna un tipo de dato, un nombre de identificador, un tipo de identificador (variable o constante), y opcionalmente un valor inicial.

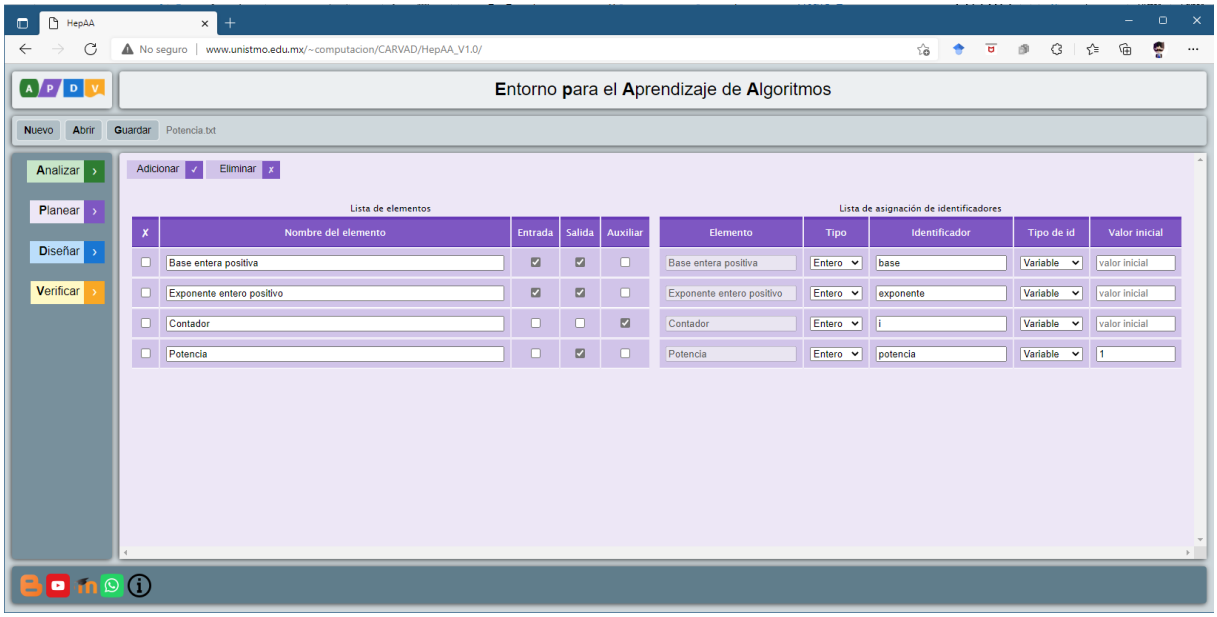
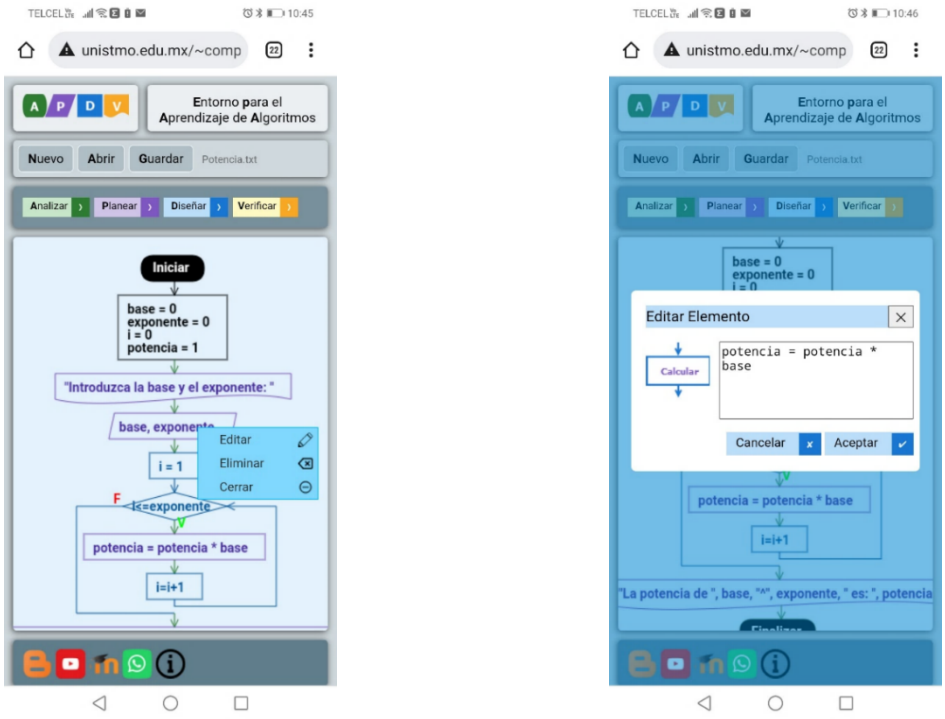


Figura 4. Pantalla Planear ejecutándose en el navegador de una computadora personal.



(a) **(b)**

Figura 5. Pantalla Diseñar ejecutándose en el navegador de un dispositivo móvil: (a) despliegue de menú flotante para la edición de los elementos del diagrama; (b) cuadro de diálogo modal para la edición de las sentencias asociadas al elemento.

En la etapa Diseñar (Figura 5) se crea un diagrama de flujo inicial con tres elementos: Iniciar, Inicializar, y Finalizar. El elemento inicializar contiene los identificadores definidos con sus valores iniciales por omisión, o los asignados por el usuario de acuerdo con el tipo de dato definido. Posterior al elemento inicializar es posible insertar elementos secuenciales (escribir, leer o calcular), condicionales o cíclicos (repetir, mientras y hacer hasta), para esto se debe dar clic derecho sobre una de las flechas del diagrama y seleccionar del menú flotante un tipo de

elemento. Una vez elegido el tipo de elemento se abrirá un cuadro de diálogo modal para editar sus sentencias. Para editar o eliminar un elemento ya insertado se debe dar clic sobre éste y elegir del menú flotante la opción deseada. La edición de un elemento provoca el proceso de compilación sobre sus sentencias, en caso de detectar algún error se notificará al usuario el mensaje correspondiente y el color del texto del elemento será rojo.

Durante la etapa Verificar (Figura 6) se realiza la interpretación del diagrama de flujo elemento a elemento, generándose en el proceso una corrida de escritorio donde se muestra cómo van cambiando los valores de las variables, el valor de la evaluación de las expresiones lógicas, así como los mensajes mostrados en "pantalla". También se hace uso de cajas de diálogo para introducir valores en la ejecución de un elemento de tipo Leer.

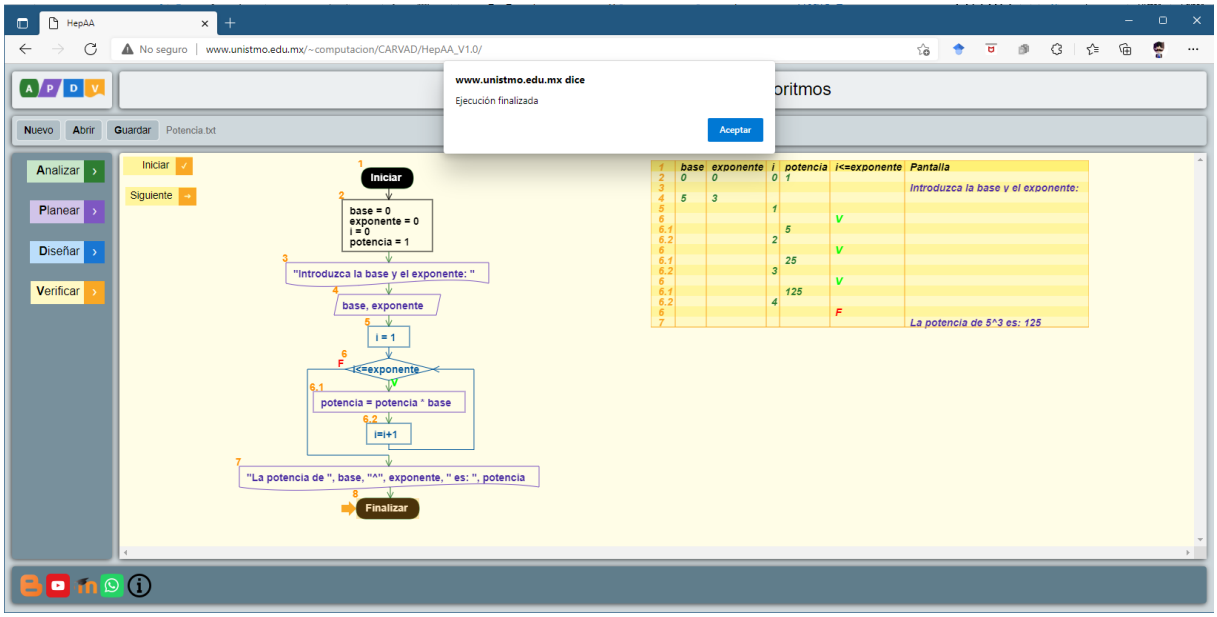


Figura 6. Pantalla Verificar ejecutándose en el navegador de una computadora personal.

La Figura 7 presenta la traducción del diagrama de flujo a código fuente en el lenguaje C, éste se obtiene dando clic sobre el elemento Iniciar y seleccionar la opción Traducir a C; también es posible traducir a pseudocódigo. Todo el código generado utiliza un sangrado de acuerdo con el nivel de anidamiento de las sentencias.

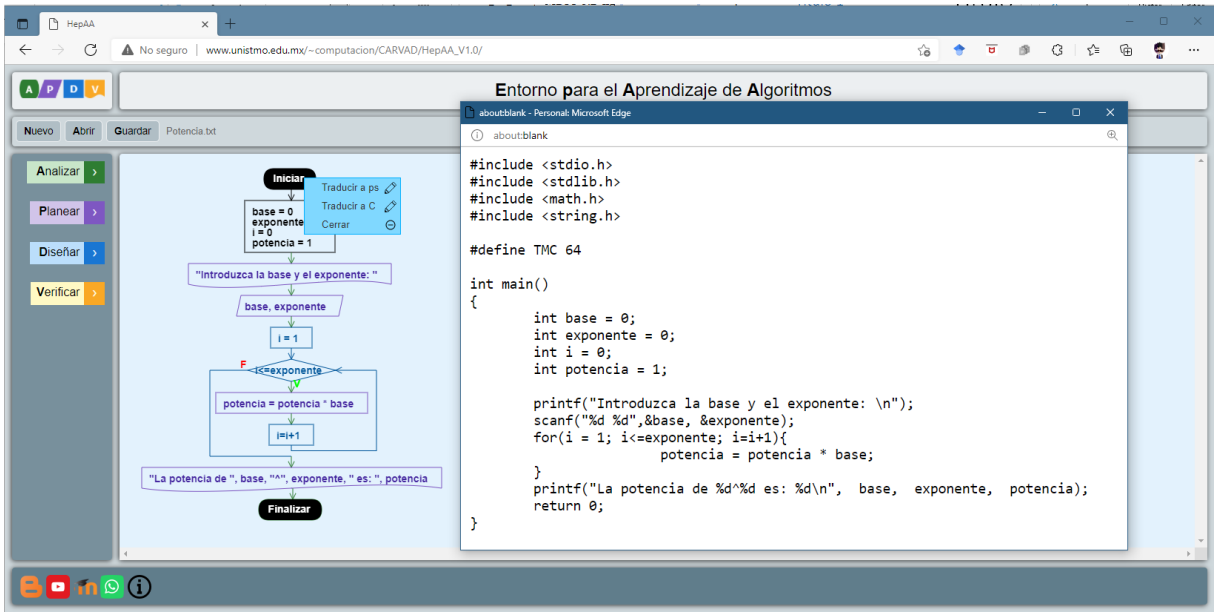


Figura 7. Traducción del diagrama de flujo a código en el lenguaje C.

Una característica más de la aplicación en línea EpAA es la posibilidad de copiar como imagen, desde el navegador pulsando el clic derecho del ratón, el diagrama de flujo durante las etapas de Diseñar y Verificar, así como la tabla correspondiente a la corrida de escritorio. Las Figuras 8 y 9 presentan imágenes copiadas durante la etapa Verificar de otros dos ejemplos de soluciones algorítmicas desarrolladas con EpAA. En la Figura 8 muestra la corrida de escritorio del diagrama de flujo correspondiente al problema de determinar si un número entero positivo mayor a uno es un número primo o no, mientras que la Figura 9 muestra la corrida de escritorio del diagrama de flujo para, dados tres números distintos entre sí, imprimirlos siempre en orden creciente.

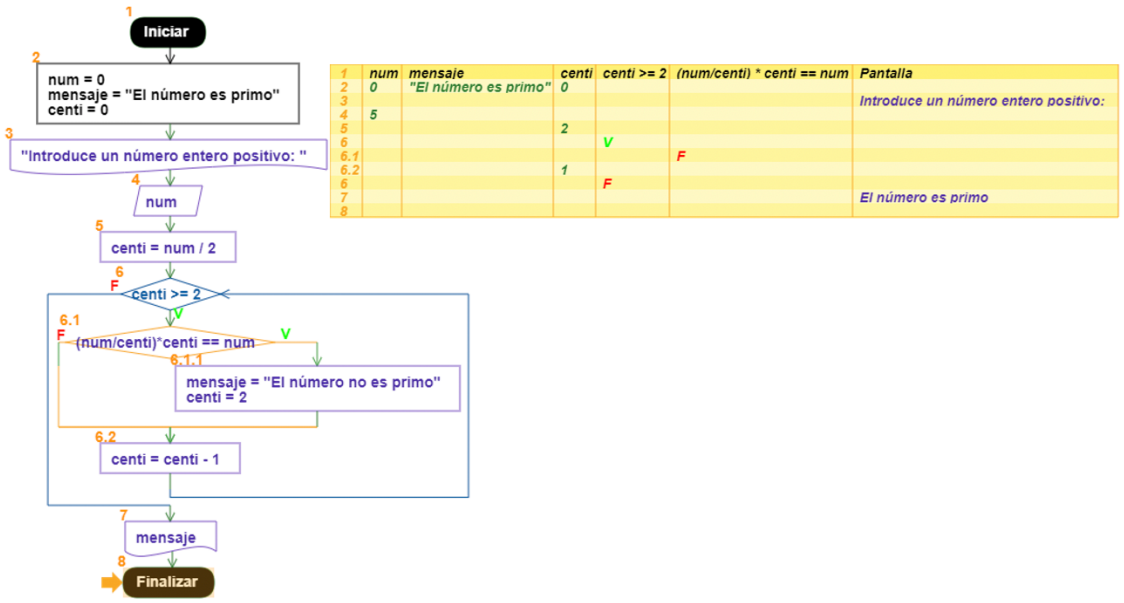


Figura 8. Corrida de escritorio para determinar si un número entero mayor a uno es primo o no.

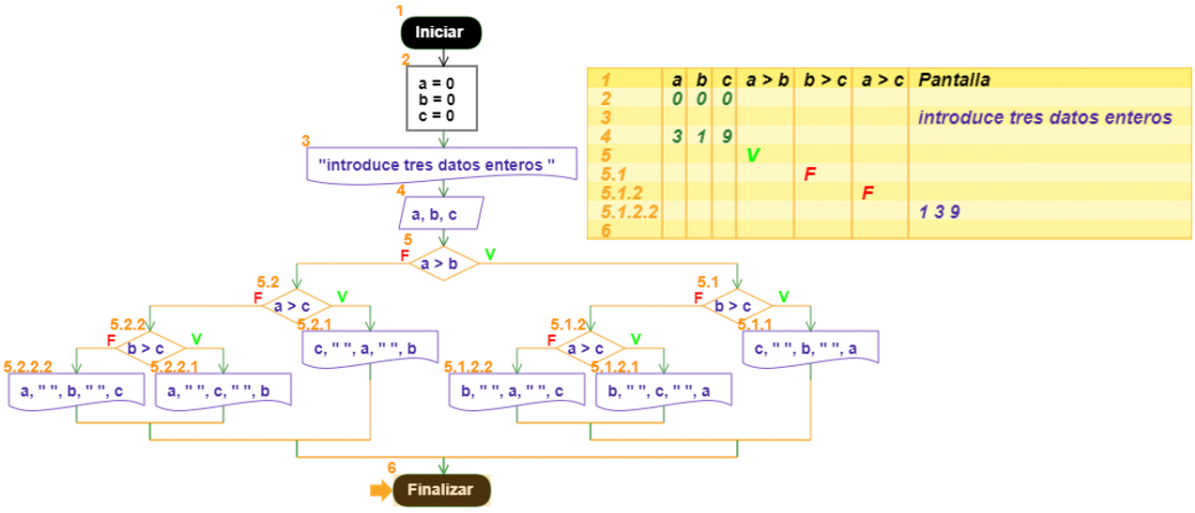


Figura 9. Corrida de escritorio para determinar imprimir en orden creciente tres números enteros diferentes entre sí.

El lector puede desarrollar y verificar las soluciones algorítmicas aquí ejemplificadas accediendo, de forma libre y gratuita, a la aplicación en línea EpAA a través del siguiente enlace: http://www.unistmo.edu.mx/~computacion/CARVAD/HepAA_V1.0/

3. Resultados y discusión

3.1. Evaluación de características con aplicaciones similares

En la Tabla 1 se contrastan las características referidas en la literatura [7], [22] y que están presentes en aplicaciones de escritorio o en línea recientes, se incluye también la aplicación en línea aquí propuesta, de nombre EpAA. En la primera columna está la aplicación AlgoBuild, creada como un software educativo diseñado para el estudio de la programación y los algoritmos con base en el paradigma imperativo [26]. En la segunda columna se tiene la aplicación Raptor, diseñada específicamente para visualizar algoritmos representados en diagramas de flujo [27]. En la tercera columna está la aplicación Flowgorithm, diseñada como un lenguaje de programación que se basa en diagramas de flujo simples [28]. En la cuarta columna está la aplicación PSeInt, diseñada para asistir en la construcción de programas o algoritmos computacionales usando pseudocódigo y diagramas de flujo [29].

Tabla 1. Características evaluadas en herramientas didácticas con soporte para diagramas de flujo.

	AlgoBuild	**Raptor**	**Flowgorithm**	**PSeInt**	**WebApp1**	**WebApp2**	**Coral**	**EpAA**
Genera código	×	✓	✓	✓	×	✓	✓	✓
Ejecución interactiva	✓	✓	✓	✓	×	×	✓	✓
Notificación de errores	✓	✓	✓	✓	×	✓	✓	✓
Tipos primitivos	✓	✓	✓	✓	×	×	✓	✓
Estructuras secuenciales	✓	✓	✓	✓	✓	✓	✓	✓
Estructuras selectivas	✓	✓	✓	✓	✓	✓	✓	✓
Estructuras iterativas	✓	✓	✓	✓	✓	✓	✓	✓
Reglas estructurales	✓	✓	✓	✓	✓	✓	×	✓
Coloreo de estructuras	×	×	✓	✓	×	✓	×	✓
Versión en español	×	×	✓	✓	✓	×	×	✓
Soporte de heurística	×	×	×	×	×	×	×	✓
Simbología ANSI	✓	×	×	×	✓	✓	×	✓
Ejecución en línea	×	×	×	×	✓	✓	✓	✓
Interfaz responsiva	×	×	×	×	×	×	✓	✓
Soporte de arreglos	✓	✓	✓	✓	×	✓	✓	×
Soporte de funciones	✓	✓	✓	✓	×	×	✓	×
Soporte de recursión	✓	✓	✓	✓	×	×	✓	×

Fuente: Elaboración propia.

Las aplicaciones en línea o Webs App contrastadas en la Tabla 1 son tres, en la columna etiquetada como Web App 1 se hace alusión al trabajo publicado por Vázquez y Jaimez [12], estos autores describen una aplicación web

para crear diagramas de flujo como apoyo al proceso de enseñanza aprendizaje de cursos de programación estructurada. En la columna etiquetada como Web App 2 se considera el trabajo de Supaartagorn [14], este autor propone una aplicación web que permite generar código de forma automática a partir de la construcción de un diagrama de flujo, de esta forma se pretende que los estudiantes se concentren más en la lógica del programa que en la sintaxis del código. En la penúltima columna está el lenguaje Coral [13], definido como un pseudo lenguaje de sintaxis simplificada que permite generar el diagrama de flujo a partir de sentencias textuales y, ya sea con una visualización de sentencias en Coral, o usando su representación en diagrama de flujo, e incluso con código en C++, permite la ejecución interactiva de una corrida de escritorio de forma visual.

En el caso de EpAA, además de cumplir con la mayoría de las características disponibles en sus similares de escritorio o en línea, se distingue por algunas características que pocas aplicaciones cumplen, por ejemplo: 1) soporte para la heurística de resolución de problemas de Polya, 2) usar simbología gráfica ANSI, y 3) estar disponible para ejecutarse en línea de forma responsiva y en el idioma español. La primera característica permite abordar un par de dificultades de aprendizaje identificadas en la literatura [23]: la no comprensión de enunciados a resolver, y la debilidad para la resolución de problemas. La segunda característica permite emplear símbolos estándar en la diagramación acordes con la bibliografía de algoritmos y cursos introductorios a la lógica de programación [30], [31]. Por último, la tercera característica al parecer no tiene un precedente que aglutine todas las características soportadas por EpAA como una aplicación en línea, ya que, por ejemplo, puede ser que solo permitan la edición de los diagramas de flujo sin llegar a su ejecución [12], o que resultan ser solo para traducir del diagrama de flujo al código fuente de un lenguaje de programación en particular [14], o que se dependa de sentencias textuales en lugar de inserción de símbolos gráficos para construir el diagrama de flujo [13]. No obstante, también es importante mencionar que EpAA tiene limitaciones respecto a otras aplicaciones contrastadas en la Tabla 1, por ejemplo, aún carece de soporte para arreglos, funciones y recursividad (técnica de programación que permite invocar funciones que se llaman a sí mismas en forma repetida, hasta que se cumple una condición de paro).

3.2. Cumplimiento de estándares W3C

La aplicación en línea EpAA fue verificada utilizando el NU Html Checker [32] de la *World Wide Web Consortium* (W3C). El resultado de la verificación arroja solo una advertencia y un error (Figura 10), ambos relacionados con el uso de cuadros de diálogo modales ya que no son soportados por todos los navegadores. Sin embargo, los navegadores que sí soportan este tipo de elementos de interacción con el usuario son el Chrome, navegador por omisión en dispositivos móviles Android, y el Edge, navegador por omisión en computadoras con sistema operativo Windows 10 y posteriores. De hecho, la cuota de mercado mensual a nivel mundial del Chrome fue del 63.59% en enero de 2021, mientras que para el mismo periodo Edge tenía una cuota del 3.24% [33]. Dada la disponibilidad y popularidad de los navegadores compatibles con cuadros de diálogo modales no se considera un inconveniente mayor la advertencia y error reportados.

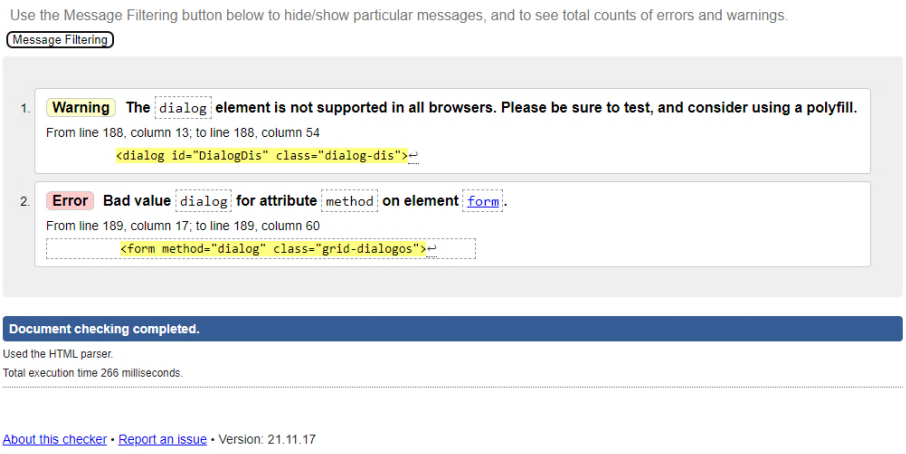


Figura 10. Resultados de la validación del cumplimento de estándares de la W3C.

3.3. Desempeño y velocidad de carga

La estimación de la velocidad de carga se realizó con la herramienta *solarwinds pingdom* [34] la cual estima el grado de rendimiento de un sitio web a través del monitoreo de ciertos atributos como: encabezados *expires* (líneas de código que indican a un navegador el tiempo de vencimiento para mantener archivos temporales en un sitio), solicitudes HTTP (Protocolo de Transferencia de Hipertexto, del inglés *Hypertext Transfer Protocol*, su función es comunicar al servidor lo que se quiere realizar), imágenes, *scripts* (instrucciones escritas en código para realizar una función específica que son ejecutadas en el navegador), elementos DOM (Modelo de Objetos del Documento, del inglés *Document Object Model*, permiten recorrer o manipular el contenido HTML), favicon en caché (facilitan la identificación visual de los recursos web), y errores 404 (producidos por recursos no localizados). Los resultados muestran un grado de desempeño de 83 en la escala de 0 a 100 (Figura 11), lo que corresponde con una nota tipo B en la escala de A hasta F, siendo A la nota más alta, por lo tanto, este resultado se considera muy positivo. El tiempo de carga reportado es de 1.42 segundos con un tamaño de página de 63.2 KB, distribuidos en 38.7 KB (61.20%) de scripts, 15.9 KB (25.20%) de imágenes, 4.2 KB (6.63%) de código HTML (para la estructura y despliegue del contenido de las páginas web), y 3.7 KB (5.83%) de CSS (códigos que se encargan de estilizar la representación visual de las páginas web).

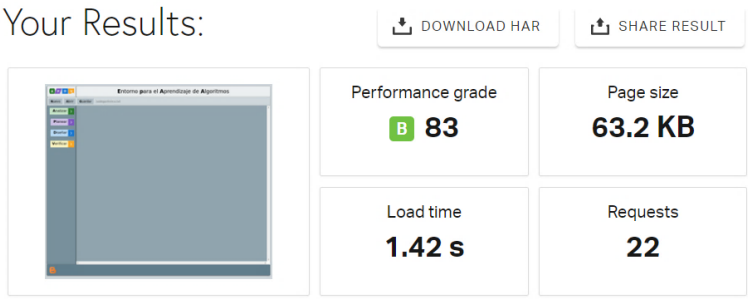


Figura 11. Resultados del desempeño y la velocidad de carga.

3.4. Percepción de la experiencia de usuario

Para la percepción de la experiencia de usuario se aplicó la prueba estandarizada SUPR-Q [35] ya que está diseñada específicamente para medir de forma integral la calidad de la experiencia del usuario de un sitio web. Dicha prueba está conformada con ocho ítems y respuestas en escala de Likert para valorar cuatro aspectos: usabilidad, confianza, lealtad y apariencia. Sin embargo, debido a que el aspecto de confianza hace alusión a compras y negocios en línea no aplica en el caso de la evaluación de EpAA. La prueba se aplicó a 22 estudiantes de los cuales el 77 % (17/22) son hombres y el 23% (5/22) son mujeres, con edades que oscilan entre los 17 y 21 años y una edad promedio de 17.8 años, de acuerdo con Bertuzzi [4] estas edades están en el rango de la generación Z. Además, todos ellos disponen de su propio teléfono inteligente, y un 91% (20/22) disponen de una computadora personal.

Los 22 estudiantes que respondieron la prueba SUPR-Q concluyeron un curso propedéutico de algoritmos, como requisito para ingresar a la universidad, durante el periodo de agosto – septiembre de 2021, cabe mencionar que la denominación de propedéutico se debe a que se trata de un curso preparatorio para el estudio de la programación de sistemas de cómputo. Este curso fue impartido a distancia utilizando la aplicación en línea EpAA como principal recurso didáctico. Para corroborar la fiabilidad de la prueba se calculó el alfa de Cronbach y la omega de McDonald utilizando el software Jamovi [36]. Los valores obtenidos fueron de 0.656 y 0.684 respectivamente, sin embargo, para alcanzar un valor de al menos 7.0 para la omega de McDonald, la cual a diferencia del alfa de Cronbach no se ve afectada por el número reducido de ítems [37], el análisis de correlación arrojado por Jamovi sugirió eliminar uno de los ítems relacionados con el aspecto de apariencia. Una vez eliminado el ítem el alfa de Cronbach quedó en 0.667 y la omega de McDonald alcanzó el valor de 0.701, valores clasificados como débil y aceptable, respectivamente [38], [37].

La Tabla 2 presenta una síntesis de los resultados del análisis de fiabilidad de la prueba SUPR-Q, es de notarse que la correlación de cada ítem con el resto de los ítems muestra que todos aportan positivamente a la valoración global, de igual forma, la correlación entre los ítems de cada aspecto también es positiva, lo cual es

evidencia de la consistencia interna de la prueba. En dicha tabla también se presentan las medias de cada ítem, en todos los casos la media está más cercana a los valores máximos que al valor medio de la escala. Lo anterior sugiere que la percepción de la experiencia de los usuarios, en su mayoría, considera al compilador e intérprete en línea de diagramas de flujo EpAA como una aplicación web que es fácil de usar y fácil de navegar dentro de ella. También, es en extremo probable que recomienden la aplicación a un amigo o colega, así como que ellos regresen a usarla en un futuro. Finalmente, opinan que la aplicación tiene una presentación limpia y clara.

Tabla 2. Estadísticas del análisis de confiabilidad de los ítems del SUPR-Q para N = 22.

Aspectos	Ítem	Media	Desviación estándar	Correlación del ítem con el resto de los ítems	Correlación entre ítems del aspecto
Usabilidad	**U1**.- La aplicación web EpAA es fácil de usar	4.64	0.492	0.561	0.588
	U2.- Es fácil de navegar dentro de la aplicación web EpAA	4.68	0.568	0.249	
Lealtad	**L1**.- ¿Qué tan probable es que recomiende la aplicación web EpAA a un amigo o colega?	9.18	1.006	0.548	0.334
	L2.- Es probable que regrese a utilizar la aplicación web EpAA en un futuro	4.55	0.800	0.367	
Apariencia	**A1**.- La aplicación web EpAA tiene una presentación limpia y simple	4.68	0.568	0.520	N/A

Fuente: Elaboración propia.

Cabe aclarar que el rango de valores para los ítems U1, U2, L2 y A1 va del 1 al 5, siendo 1 “Muy en desacuerdo” y 5 “Muy de acuerdo”, mientras que para el ítem L1 la escala va desde 0 hasta 10, donde 0 representa “No en absoluto probable” y 10 “Si es extremadamente probable”. De forma más puntual, la Figura 12 muestra de forma gráfica las frecuencias asociadas a los ítems U1, U2, L2 y A1. Es de notarse que es realmente bajo el número de estudiantes que consideran estar ni de acuerdo ni en desacuerdo respecto los ítems U2, L2 y A1; el ítem L2 también tiene un número reducido de estudiantes que manifestaron estar en desacuerdo. En lo que respecta al ítem L1 el valor más bajo fue de 7, en la escala de 0 a 10, para este ítem la moda fue 10, es decir el valor más alto de la escala.

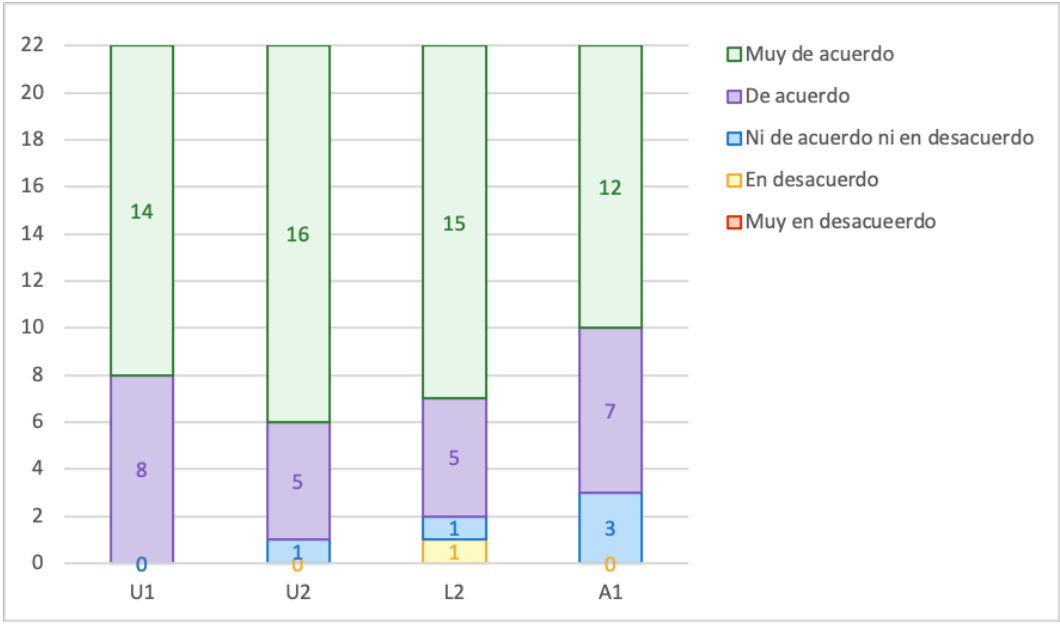


Figura 12. Frecuencias de los ítems U1, U2, L2, y A1 de la prueba SUPR-Q.

5. Conclusiones

El compilador e intérprete en línea de diagramas de flujo, denominado EpAA, reúne un conjunto de características suficientes para el abordaje de un primer curso de programación imperativa a nivel universitario o preuniversitario. En este sentido, se puede concluir que está a la par de aplicaciones de escritorio similares ampliamente utilizadas en cursos introductorios a la lógica de programación [27], [28], [29]. Pero también presenta una característica única respecto a las herramientas didácticas con soporte para diagramas de flujo, ya sean de escritorio o en línea, el soporte para la heurística de resolución de problemas de Polya [19]. Además de otras características no tan comunes que pueden destacarse son el atender las recomendaciones de la simbología gráfica ANSI, y la disponibilidad en el idioma español en internet de forma libre, con la posibilidad de ejecutarse de forma adaptable tanto en equipos de cómputo personal Windows como en dispositivos móviles Android. No obstante, también presenta algunas limitantes como carecer de soporte para arreglos, funciones y recursividad.

Los resultados de las pruebas del cumplimiento de estándares W3C fueron satisfactorios ya que solo se detectaron dos problemas relacionados con el uso de cuadros de diálogo modales, los cuales se utilizan durante la interacción con el usuario para garantizar el cumplimiento de las reglas estructurales y sintácticas del diagrama de flujo. Afortunadamente, los navegadores compatibles con este tipo de interacción son los preinstalados en la inmensa mayoría de los dispositivos móviles con Android, y en las computadoras personales con Windows, también es importante señalar que estos dos navegadores son de los más utilizados [33]. En cuanto a las pruebas de velocidad de carga y desempeño se consideran satisfactorias puesto que el tiempo de carga es menor a dos segundos y el tamaño es relativamente pequeño, menor a 100 KB.

Respecto a la percepción de la experiencia de usuario se aplicó la prueba estándar SUPR-Q [35] a 22 estudiantes con edades acordes a la generación Z [3], [4]. Se validó el grado de confiabilidad interna de dicha prueba con los índices alfa de Cronbach y omega de McDonald, siendo este último aceptable con un valor de 0.701 [37]. Los resultados de la prueba respecto a la aplicación en línea EpAA señalan que los estudiantes la valoran con un muy buen nivel de usabilidad, lealtad y apariencia. Es decir, no tuvieron mayores complicaciones para aprender a usarla y navegar en ella, consideran volverla a usar y en extremo probable recomendarla con amigos y colegas, además de que la perciben con una presentación limpia y simple. Cabe mencionar que, de los 22 estudiantes 2 de ellos (9%) utilizaron consistentemente su teléfono inteligente para realizar sus tareas en equipo, ello se debe a que no disponían de equipo de cómputo personal. Quienes si disponían de equipo de cómputo personal usaban en menor medida el teléfono inteligente.

Aunque al momento se obtuvieron buenos resultados a partir del análisis comparativo de características entre aplicaciones didácticas con soporte para diagramas de flujo, así como de las pruebas realizadas a la aplicación web EpAA, este es un primer esfuerzo por construir una aplicación más completa y robusta, por lo que se consideran como trabajos a futuro el soporte para arreglos, funciones y recursividad, así como la traducción a otros lenguajes de alto nivel. Además, si bien se tiene la limitante de un número reducido de usuarios encuestados, los resultados de la percepción de la experiencia de usuario permiten establecer una primer línea base para contrastar con futuras adecuaciones o mejoras en la interfaz; como trabajo a futuro también se prevé encuestar a un mayor número de usuarios para obtener resultados estadísticos más concluyentes.

6. Referencias

[1] Göktepe, M., Özgüc, B., Baraym M. (1989). Design and implementation of a tool for teaching programming. *Computers & Education*, 13 (2), 167-178. doi: https://doi.org/10.1016/0360-1315(89)90009-2

[2] INEGI. (2021). *Comunicado de prensa Núm. 352/21*. Recuperado de: https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2021/OtrTemEcon/ENDUTIH_2020.pdf

[3] Dimok, M. (2019). *Defining generations: Where Millennials end and Generation Z begins*. Pew Research Center. Recuperado de: http://www.pewresearch.org/fact-tank/2019/01/17/where-millennials-end-and-generation-z-begins/

[4] Bertuzzi, M. F. (2021). Centenials en la universidad: prosumidores de contenido en el aula. En M. Veneziani, P. de la Sotta (Coord.), *Cuadernos del Centro de Estudios en Diseño y Comunicación Nº 134* (pp. 161-173). Buenos Aires, Argentina: Universidad de Palermo. Recuperado de: https://dspace.palermo.edu/ojs/index.php/cdc/article/view/5020/6682

[5] Manzanares Triquet, J. C. (2020). Generación Z y gamificación: el dibujo pedagógico de una nueva sociedad educativa. *Tejuelo*, 32, 263-298. doi: https://doi.org/10.17398/1988-8430.32.263

[6] Asociación Mexicana de Internet. (2021). *17º Estudio sobre los Hábitos de los Usuarios de Internet en México 2021*. Recuperado de: https://irp.cdn-website.com/81280eda/files/uploaded/17%C2%B0%20Estudio%20sobre%20los%20Ha%CC%81bitos%20de%20los%20Usuarios%20de%20Internet%20en%20Me%CC%81xico%202021%20v16%20Publica.pdf

[7] Hooshyar, D., Ahmad, R. B., Nasir, M. H. N. M., Shamshirband, S., Horng, S. J. (2015). Flowchart-based programming environments for improving comprehension and problem-solving skill of novice programmers: a survey. *International Journal of Advanced Intelligence Paradigms*, 7 (1), 24-56. doi: https://doi.org/10.1504/ijaip.2015.070343

[8] Rahman, M. M., Sharker, M. H., Paudel, R. (2020). An Effective Approach to Teach an Introductory Computer Science Course with Computational Thinking and Flow-Chart Based Visual Programming. Trabajo presentado en *IEEE Frontiers in Education Conference (FIE)*. Uppsala, Sweden. doi: https://doi.org/10.1109/FIE44824.2020.9273930

[9] Sánchez, M., Valderrama Bahamondez, E., de Clunie, G. T. (2020). Use of PSeInt in teaching programming: a case study. Trabajo presentado en *10th Euro-American Conference on Telematics and Information Systems (EATIS)*. Aveiro, Portugal. doi: https://doi.org/10.1145/3401895.3402083

[10] Shivacheva, G. I., Ruseva, N. R. (2021). Training in Programming using Innovative Means. Trabajo presentado en *International Conference on Technics, Technologies and Education.* Yambol, Bulgaria. Recuperado de: https://iopscience.iop.org/article/10.1088/1757-899X/1031/1/012124

[11] Zaretska, I., Zholtkevych, G., Radchenko, A., Minayev, A. (2019). Algorithms Constructor. En V. Ermolayev, F. Mallet, V. Yakovyna, H. Mayr, A. Spivakovsky (Eds.). *ICT in Education, Research and Industrial Applications 2019* (pp. 501-506). Kherson, Ukranine: CEUR-WS. Recuperado de: http://ceur-ws.org/Vol-2387/20190501.pdf

[12] Vázquez-Peñaloza, F., Jaimez-González, C. R. (2019). Towards a Web Application to Create Flowcharts for Supporting the Teaching-Learning Process of Structured Programming Courses. *American Journal of Educational Research*, 7 (12), 976-982. Recuperado de: http://pubs.sciepub.com/education/7/12/12/

[13] Allen, J. M., Vahid, F. (2020). Teaching Coral before C++ in a CS1 Course. Trabajo presentado en *American Society for Engineering Education (ASEE), Virtual Annual Conference Content Access*. Virtual On line. Recuperado de: https://peer.asee.org/35273

[14] Supaartagorn, C. (2017). Web Application for Automatic Code Generator Using a Structured FlowChart. Trabajo presentado en *8th IEEE International Conference on Software Engineering and Service Science*. Beijing, China. doi: https://doi.org/10.1109/ICSESS.2017.8342876

[15] Cabo, C. (2018). Effectiveness of Flowcharting as Scaffolding Tool to Learn Python. Trabajo presentado en *IEEE Frontiers in Education Conference (FIE)*. San Jose, CA, USA. doi: https://doi.org/10.1109/FIE.2018.8658891

[16] Zhang, J., Meng, B., Zou, L., Zhu, Y., Hwang, G. (2021). Progressive flowchart development scaffolding to improve university students' computational thinking and programming self-efficacy. *Interactive Learning Environments*, 1-18. doi: https://doi.org/10.1080/10494820.2021.1943687

[17] Pressman, R. S. (2010). *Ingeniería de Software. Un enfoque práctico* (7ma Ed.). México D. F.: McGraw Hill Educación.

[18] Arellano Pimentel, J. J., Nieva García, O. S., Solar González, R., Arista López, G. (2012). Software para la enseñanza-aprendizaje de algoritmos estructurados. *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología*, (8), 23-33. Recuperado de: https://teyet-revista.info.unlp.edu.ar/TEyET/article/view/253

[19] Polya, G. (2005). *Cómo plantear y resolver problemas* (1era Ed.). México: Trillas.

[20] Louden, K. (2004). *Construcción de compiladores principio y práctica*. México: Parainfo.

[21] Aho, A., Sethi, R., Ullman, J. (1998). *Compiladores, principios, técnicas y herramientas*. Naucalpan de Juárez, Estado de México: Addison Wesley Longman.

[22] Gajewski, R. R. (2018). Algorithms, Programming, Flowcharts and Flowgorithm. En E. Smyrnova-Trybulska (Ed.). *E-learning and smart learning environment for the preparation of new generation specialist* (pp. 393–408). Katowice, Polonia: University of Silesia.

[23] Jiménez-Toledo, J. A., Collazos, C., Revelo-Sánchez, O. (2019). Consideraciones en los procesos de enseñanza-aprendizaje para un primer curso de programación de computadores: una revisión sistemática de la literatura. *TecnoLógicas*, 22, 83-117. doi: https://doi.org/10.22430/22565337.1520

[24] Santimateo, D., Nuñez, G., González, E. (2018). Estudio de dificultades en la enseñanza y aprendizaje en los cursos básicos de programación de computadoras en Panamá. *Revista de Investigación en Tecnologías de la Información (RITI)*, 6 (11), 13-18. Recuperado de: https://www.riti.es/ojs2018/inicio/index.php/riti/article/view/81

[25] Gauchat, J. D. (2012). *El gran libro de HTML5, CSS3 y Javascript*. Barcelona, España: MARCOMBO.

[26] Santi, P. (2020). *AlgoBuild*. (Versión 0.85) [Software de computadora]. Recuperado de: https://algobuild.com/en/index.html

[27] Wilson, T. A., Carlisle, M. C., Humphries, J. W., Moore, J. A. (2019). *Raptor* (Versión 4.1.0.0001) [Software de computadora]. Recuperado de: https://raptor.martincarlisle.com/

[28] Cook, D. (2021). *Flowgorithm* (Versión 2.30.3) [Software de computadora]. Recuperado de: http://www.flowgorithm.org/

[29] Novara, P. (2021). *PSeInt* (Versión 20210609) [Software de computadora]. Recuperado de: http://pseint.sourceforge.net/

[30] Cairó, O. (2005). *Metodología de la programación. Algoritmos, diagramas de flujo y programas* (3era Ed.). México, D.F.: Alfaomega.

[31] Joyanes, L. (2008). *Fundamentos de programación. Algoritmos, estructuras de datos y objetos* (4ta Ed.). Madrid, España: McGraw-Hill.

[32] World Wide Web Consotium. (2021). *Nu Html Checker* (Versión 21.11.17) [Software de computadora]. Recuperado de: https://validator.w3.org/nu/

[33] Statista. (2021). *Ranking de los navegadores de internet con mayor cuota del mercado mensual entre enero de 2016 y enero de 2021*. Recuperado de: https://es.statista.com/estadisticas/600249/cuota-de-mercado-mensual-de-los-principales-navegadores-de-internet/

[34] SolarWinds Pingdom. (2021). *Pingdom Website Speed Test*. Recuperado de: https://tools.pingdom.com/

[35] Sauro, J. (2015). SUPR-Q: A Comprehensive Measure of the Quality of the Website User Experience. *Journal of Usability Studies*, 10 (2), 68-86. Recuperado de: https://uxpajournal.org/wp-content/uploads/sites/7/pdf/JUS_Sauro_Feb2015.pdf

[36] The jamovi Project. (2021). *Jamovi* (Version 1.6) [Computer Software]. Recuperado de: https://www.jamovi.org.

[37] Ventura-León, J. L., Caycho-Rodríguez, T. (2017). El coeficiente Omega: un método alternativo para la estimación de la confiabilidad. *Revista Latinoamericana de Ciencias Sociales, Niñez y Juventud*, 15 (1), 625-627. Recuperado de: https://www.redalyc.org/pdf/773/77349627039.pdf

[38] Gliem, J. A, Gliem, R. R. (2003). Calculating, Interpreting, And Reporting Cronbach's Alpha Reliability Coefficient For Likert-Type Scales. Trabajo presentado en *Midwest Research-to-Practice Conference in Adult, Continuing, and Community Education*. Columbus, Ohio. Recuperado de: https://hdl.handle.net/1805/344

