UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

PROTOTIPO DE UN SISTEMA DE APOYO PARA APRENDER A PROGRAMAR

Reporte Técnico de Investigación presentado por:

Héctor Miguel Melero Román 141017

Requisito para la obtención del título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

M.A.T.I. René Noriega Armendáriz

Ciudad Juárez, Chihuahua 25 de noviembre de 2019

Ciudad Juárez, Chihuahua, a 11 de Noviembre de 2019

Asunto: Liberación de Asesoría

Mtro. Ismael Canales Valdiviezo
Jefe del Departamento de Ingeniería
Eléctrica y Computación
Presente.-

Por medio de la presente me permito comunicarle que, después de haber realizado las asesorías correspondientes al reporte técnico Prototipo de un sistema de apoyo para aprender a programar, del) alumno Héctor Miguel Melero Román, de la Licenciatura en Ingeniería en Sistemas Computacionales, considero que lo ha concluido satisfactoriamente, por lo que puede continuar con los trámites de titulación intracurricular.

Sin más por el momento, reciba un cordial saludo.

Atentamente

René Noriega Ármendáriz
Profesor Investigador IIT

UACJ
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y COMPUTACIÓN

Cep:
Coordinador del Programa de Sistemas Computacionales
Héctor Miguel Melero Román
Archivo

Ciudad Juárez, Chihuahua, a 11 de Noviembre de 2019

Asunto: Autorización de publicación

C. Héctor Miguel Melero Román

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la publicación del documento de Prototipo de un sistema de apoyo para aprender a programar, para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Licenciado en Ingeniería en Sistemas Computacionales.

Sin otro particular, reciba un cordial saludo.

Mtra. Ivonne Haydee Robledo Portillo

Profesor Titular de Seminario de Titulación II

Declaración de Originalidad

Yo, Héctor Miguel Melero Román, declaramos que el material contenido en esta publicación fue elaborado con la revisión de los documentos que se mencionan en el capítulo de Bibliografía, y que la solución obtenida es original y no ha sido copiada de ninguna otra fuente, ni ha sido usada para obtener otro título o reconocimiento en otra institución de educación superior.

Héctor Miguel Melero Román

Agradecimientos

Este proyecto fue posible gracias a mi asesor, el maestro René Noriega, quien siempre mostró una gran paciencia, ánimo y apoyo constante durante todo este proyecto. Muchas gracias.

Le agradezco a mi maestra de Seminario II, la Dr. Ivonne Robledo por su paciencia a lo largo del proyecto.

A mi compañero Cesar Alonso, quien me brindó su ayuda y conocimientos en el desarrollo del presente proyecto.

A mis compañeros del trabajo, quienes siempre mostraron su ánimo, apoyo y disponibilidad para concluir con el proyecto.

Por último, pero no menos importante, agradezco a mis padres y a mi novia, quienes siempre me otorgarón su apoyo y comprensión para poder seguir adelante y concluir este proyecto.

Gracias.

Dedicatoria

Para mis padres, hermanas y amigos.

Índice general

Índice de figuras

Índice de tablas

Resumen

El progreso de la tecnología hoy en día está presente en cualquier lugar, desde el hogar y trabajo, hasta en la educación, entre otras cosas. Esto significa que la sociedad se encuentra en un cambio llamado "la cuarta revolución industrial", donde grandes avances como la inteligencia artificial, redes neuronales y el procesamiento de millones de datos son parte de la vida diaria de muchas personas que se dedican a estos campos.

Debido a esto, aprender el arte de la programación es una habilidad altamente valorada hoy en día. Lamentablemente, algunas veces puede parecer intimidante, debido a que antes de resolver un problema mediante la programación, es necesario tener un mínimo de conocimientos sobre pasos, reglas y conceptos fundamentales antes de comenzar a programar.

El presente documento abarca el desarrolló de un prototipo de sistema web para aprender a programar. Se desarrolló un proyecto web que permite seleccionar al usuario temas relacionados con el contenido utilizado en la asignatura de fundamentos de programación, este contenido se compone de seis secciones. La primer sección, corresponde a la introducción donde se explica el concepto de cada tema. Luego, en el siguiente apartado se muestra un problema de ejemplo con el propósito de mostrar al usuario como se analiza, como se desarrolla un algoritmo (diagrama de flujo) y un pseudocódigo, con la finalidad de llegar a la codificación.

Con base en lo anterior, se desarrolló una evaluación utilizando la herramienta llamada formularios de Google, la cual ofrece la opción de crear auto evaluaciones, es decir que se

puede definir la respuesta correcta de cada pregunta y al final de cada evaluación se realizará una revisión de manera automática. Además de asignar un valor a cada pregunta para obtener un puntaje final y así obtener un resultado.

El proyecto se desarrolló utilizando Laravel, que es un framework de código abierto basado en PHP, entre otras herramientas como Vue.js para darle reactividad. Además, se usó la herramienta Laragon (traje de desarrollo PHP que funciona como un administrador Mysql).

Finalmente, el proyecto fue presentado a 32 estudiantes y se aplicó como instrumento de evaluación una encuesta, con el propósito de medir la funcionalidad del sistema.

Palabras clave: Fundamentos de programación, resolución de problemas, sistema web, educación, objetos de aprendizaje.

Abstract

Today, the progress of technology is present anywhere, from home and work, even in education, among other areas. This means that the society is in a change called "the fourth industrial revolution, where big advances such as artificial intelligence, neural networks and the processing of millions of data are part of the daily life of people who are dedicated to these fields.

For this reason, learn programming is a highly valued ability today. Unfortunately, sometimes it may seem intimidating, because solving a problem through programming, it is necessary to have a minimum of knowledge about fundamental steps, rules and concepts before starting to program.

This document include the development of a web system prototype to learn to program. A web project was developed that allows the user to select topics related to the content used in the subject of programming fundamentals, this content consists of six sections. The first section corresponds to the introduction where the concept of each topic is explained. Then, in the following section an example problem is shown in order to show the user how it is

analyzed, how an algorithm (flow chart) and a pseudocode are developed, in order to arrive at the coding.

Based on the above, an evaluation was developed using the tool called Google forms, which offers the option of creating self-evaluations, in other words, the correct answer to each question can be defined and at the end of each evaluation a review will be carried out automatically. In addition to assigning a value to each question to obtain a final score and thus obtain a result.

The project was developed using Laravel, which is an open source framework based on PHP, among other tools such as Vue.js to give it reactivity. In addition, the Laragon tool (PHP development suit that works as a Mysql manager) was used.

Finally, the project was presented to 32 students and a survey was applied as an evaluation instrument, with the purpose of measuring the functionality of the system.

Keywords: Programming Fundamentals, problem Solving, web system, education, learning objects.

Introducción

El presente proyecto abarca el proceso de desarrollo de un prototipo de sistema web para aprender a programar, con base en la carta descriptiva de fundamentos de programación (formato modelo educativo UACJ VISIÓN 2020).

En el primer capítulo se da comienzo a los antecedentes que fundamentan este proyecto, donde se abordan problemas relacionados a con la educación acerca de la programación. Además, se describen las aplicaciones más relevantes, así como sus objetivos y visión respecto a la enseñanza de la programación.

Posteriormente, en el segundo capítulo se plantean los temas y conceptos para el desarrollo de este proyecto. Se menciona la técnica de resolución de problemas utilizada, así como los temas más relevantes acerca de los fundamentos de la programación. Por último, se revisa el apartado de usabilidad, con el fin de desarrollar un proyecto amigable para el usuario.

Después, en el capítulo se muestran todas las herramientas y métodos que fueron fundamentales para desarrollar el presente proyecto. Además, se explica el producto y se propone una solución, posteriormente se describe y desarrolla las etapas según la metodología seleccionada (modelo RUP).

Enseguida, en el cuarto capítulo se exponen los resultados obtenidos mediante encuestas acerca del funcionamiento y efectividad del proyecto. Finalmente, en capítulo cinco se muestran las conclusiones adquiridas al finalizar con el desarrollo del proyecto, donde se analiza si se alcanzó el objetivo y se exponen recomendaciones para futuras investigaciones.

Capítulo 1
Planteamiento del Problema

El presente capítulo da comienzo a los antecedentes primordiales del proyecto, donde se abordan problemas relacionados con la educación acerca de la programación. También se mencionan las aplicaciones más importantes, sus objetivos y visión con la enseñanza de la programación. De manera subsecuente, se abordará la definición del problema donde se establecen los inconvenientes que tiene un estudiante al momento de intentar resolver un problema mediante la codificación. Finalmente, se plantea el objetivo general del proyecto, que es la elaboración de un prototipo de aplicación para ofrecer una herramienta más de apoyo en el aprendizaje de los conceptos básicos de la solución de problemas utilizando programación.

1.1. Antecedentes

A través de los años, la humanidad ha tenido el menester de realizar diversas tareas con el fin de conseguir un sustento. Con el pasar de los años y el avance de la tecnología, las tareas implicaban un mayor grado de productividad, manteniéndose así hasta llegar a el periodo conocido como la revolución industrial, el cual, trajo cambios importantes en la productividad de las fábricas, donde rápidamente los trabajos manuales fueron reemplazados por máquinas.

Con la integración de las computadoras en la industria el costo y el tiempo se fueron

reduciendo, con esto la capacidad de generar y manipular información fue incrementando hasta llegar a la actualidad donde se tienen cifras enormes de datos. por ello, la programación surge para poder procesar y manipular grandes cantidades de datos con mucha facilidad [1]. Debido a la alta importancia de la programación hoy en día, se tiene previsto que, dentro de unos años, la programación será fundamental en el día a día de las personas. Por esto mismo se puede observar que escuelas de países con un desarrollo de alto nivel, fomentan de mayor manera iniciativas para enseñar a los niños a programar desde pequeños.

Hadi Partovi quien es el cofundador de la famosa organización "Code.org", afirma que cualquier niño debería aprender a dominar al menos todos los aspectos básicos de la programación, sin importar si llegan a ser programadores o no, dado a que la programación será una habilidad elemental [2].

En la web existen múltiples sitios que apoyan la enseñanza de la programación, ofrecen explicaciones en forma de "cursos en línea" donde, ya sea de forma gratuita o por medio de una suscripción de pago para tener acceso a su contenido de modo completo y el usuario pueda aprender desde casa.

A nivel global se cuenta con las herramientas siguientes:

En [3], se desarrolló un proyecto web sin ánimo de lucro de origen español, el cual, ofrece una gran variedad de cursos en diferentes áreas de la programación, por ejemplo, realidad aumentada, Arduino, robótica, entre otras cosas.

En [4], se puede ver un proyecto web gratuito desarrollado por la organización code, compañía que se dedica a ofrecer acceso a las ciencias de la computación enfocado a escuelas, esta organización tiene el apoyo de grandes compañías de tecnología como *Microsoft*, *Google*, *Amazon*.

En el sitio web [5], enfocado a entusiastas de la programación fundado en 2011 por Rajesh Moorjani, permite a los usuarios aprender a programar en distintos lenguajes de manera

gratuita. Ofrece un amplio catálogo de ejemplos y ejercicios que ayudan a reforzar las clases vistas. Además, ofrece la oportunidad de codificar y compilar desde el navegador.

La plataforma web española [6], creada por Rubén Bernárdez, ofrece retos de programación y un espacio para compartir su código y resolver dudas mediante el apoyo de la comunidad.

En [7], se llevó a cabo un sistema web fundado en el 2013, con el objetivo de ayudar a personas en el estudio de la programación, haciendo uso de material de didáctico como teoría y ejercicios prácticos con el fin de reforzar los conceptos aprendidos. Actualmente también cuenta con soporte para móviles.

En [8], se encuentra una aplicación web desarrollada para ayudar a los usuarios a practicar la programación y mejorar sus habilidades de codificación. Esta aplicación ofrece a los usuarios una colección de desafíos con diferentes dificultades. El sitio web también ofrece la oportunidad de completar dichos retos desde su compilador de código en línea.

A nivel institucional se tienen los proyectos:

En [9], se desarrolló un proyecto el cual utiliza los objetos de aprendizaje (OA) para apoyar a los estudiantes con el fin de introducir a los alumnos hacia un pensamiento lógico algorítmico para la resolución de problemas mediante computadora.

En [10], se presenta un juego serio con el propósito de reforzar habilidades cognitivas necesarias en la programación, enfocado a estudiantes de ingeniería.

En [11], se desarrolló una aplicación en el 2013 enfocada a la enseñanza y aprendizaje de la codificación estructurada utilizando animaciones generadas a partir de código fuente, con el fin de apoyar a profesores y estudiantes.

1.2. Definición del problema

Hoy en día el progreso tecnológico está presente en todos lados: hogar, trabajo, educación, entre otras cosas. La sociedad se encuentra cruzando por la llamada "cuarta revolución industrial", en donde invenciones tales como grandes volúmenes de datos, inteligencia artificial o el Internet de las cosas son tecnologías en constante crecimiento y son parte del día a día de muchas personas y para poder lograr todo esto se necesita de personas expertas en estos entornos [12].

Aprender el arte de la programación puede parecer intimidante, debido a que se deben de conocer al menos cierta cantidad de conceptos y reglas básicas antes de poder resolver un problema. En la actualidad, la programación tiene mucha importancia en el campo laboral, incluso hoy en día puede llegar a considerarse una habilidad tan apreciada como la de aprender un segundo idioma, por ejemplo, inglés.

Otro de los problemas más comunes cometido por estudiantes que comienzan a avanzar en la programación está en la técnica de resolución de un problema, donde se ignoran pasos muy importantes en este proceso, como lo es el análisis y comprensión del problema, el diseño de un método para resolverlo, diagramas de flujos, pseudocódigo, entre otras cosas. Por estos malos hábitos los estudiantes suelen estancarse en un problema y a largo plazo pueden llegar a abandonar el curso.

1.3. Objetivo de la investigación

Desarrollar un prototipo de aplicación web que ofrezca contenido didáctico acerca de los principios básicos de programación para las personas interesadas en aprender los fundamentos de la programación.

1.4. Preguntas de investigación

¿De que manera se va a realizar, mostrar y evaluar el contenido del proyecto?

¿Cómo desarrollar una herramienta web que haga gestión del contenido desarrollado?

1.5. Justificación

El desarrollo de este proyecto es importante puesto que en la actualidad se tiene previsto a la programación como una habilidad fundamental en la vida cotidiana de las personas. Por lo tanto, es crucial que las personas cuenten con las herramientas necesarias para aprender a programar.

A nivel educativo se espera que los estudiantes puedan contar con un prototipo de aplicación web, que les ofrezca explicaciones sobre temas y problemas de programación vistos en clase, donde cada problema cuente con un análisis, un algoritmo, una representación en un diagrama de flujo y la codificación del problema en diferentes lenguajes de programación.

A nivel local, específicamente en el Instituto de ingeniería y Tecnología según el último anuario estadístico de la UACJ (2017-2018) [13], se espera beneficiar a 5,035 estudiantes, (Figura 1.1), es decir que, se incluyen todos los estudiantes de ingeniería independientemente del programa al que pertenezcan, debido a que aprender el arte de programar es una habilidad de gran importancia en todas las ramas de la ingeniería.

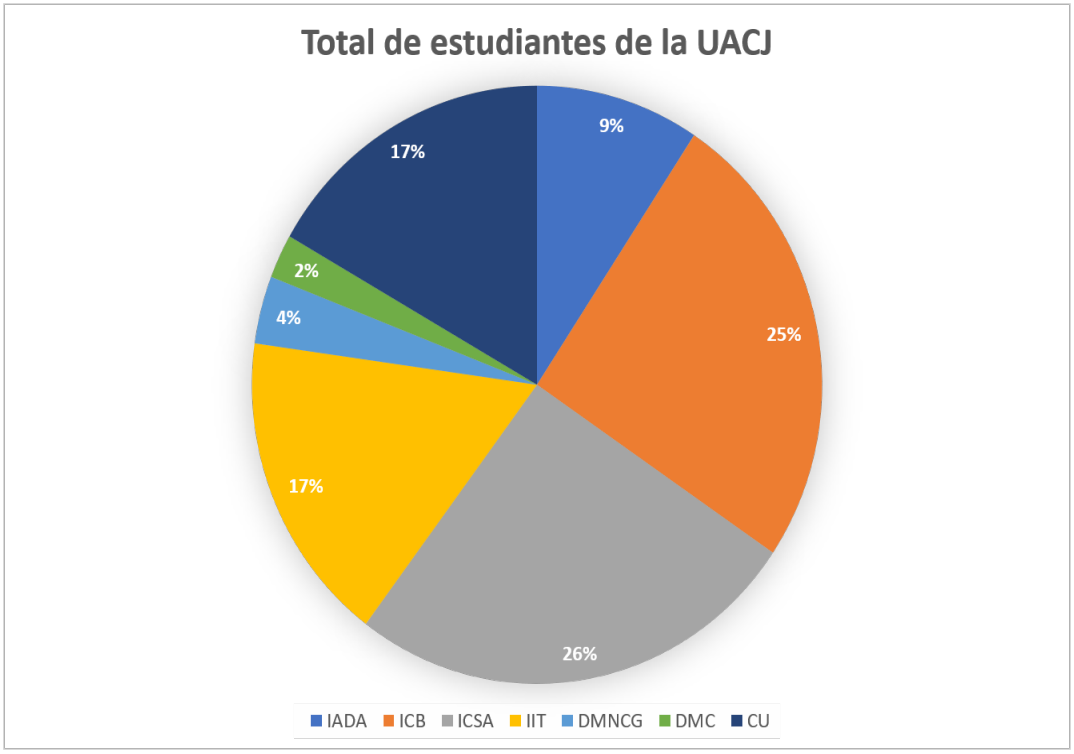


Figura 1.1: Gráfica de estudiantes beneficiados.

Capítulo 2
Marco teórico

El capítulo siguiente aborda los temas y conceptos esenciales para el desarrollo del proyecto. En primer lugar, se menciona una de las técnicas más reconocidas para la solución de problemas. Después, se presentan los temas mas relevantes de los fundamentos de programación, donde se conocerán aspectos importantes para el proyecto como lo son las herramientas de diseño de soluciones, entre otros cosas. Por último, se revisa la parte de usabilidad, la cual permitirá desarrollar un proyecto amigable para el usuario.

2.1. Metodología para la solución de un problema

George Polya fue un matemático nacido en Budapest, Hungría. quien fue el autor de uno de los estudios más notables acerca de los métodos de solución de problemas entre muchas otras publicaciones las cuales plasmó en su libro "cómo plantear y resolver problemas", donde planteó una metodología que ayudara a resolver problemas matemáticos e incluso ser aplicada para resolver problemas cotidianos [14, 15].

En [16], la técnica actualmente es conocida como método de Polya y puede ser aplicada para resolver un problema mediante la programación, consta de las siguientes cuatro fases, (Figura 2.1).

- I. Entender el problema. Una persona no podrá resolver un problema sin antes comprender que es lo necesita calcular. Para esto primero se debe leer y examinar el problema de manera cuidadosa, una vez hecho este análisis minucioso es momento de preguntarse ¿cuál es la incógnita?, ¿qué condiciones debo considerar? ¿qué datos hay?

- II. Trazar un plan. Una vez que se comprendió por completo el problema, es momento de preguntarse ¿qué tipo de cálculos debo hacer?, se debe de trazar el plan adecuado y específico a el problema que se desea resolver.

- III. Ejecutar el plan. En este paso se debe poner en marcha el plan para solucionar el problema, es importante estar atento a los resultados y ser paciente.

- IV. Revisar. Esta fase consiste en examinar la solución obtenida, el resultado debe ser razonable y puede realizar preguntas tales como ¿se cumplió con las necesidades del problema?, ¿es posible dar la misma solución al problema con otro plan? para estar satisfecho con el resultado

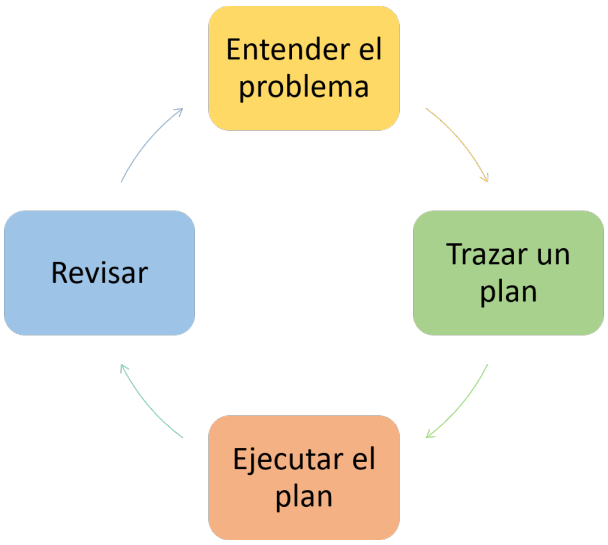


Figura 2.1: Esquema de las cuatro fases de Polya.

2.2. Resolución de problemas por computadora

Para dar solución a un problema utilizando la computadora es importante fijar cierto número de pasos para resolver un problema, esta serie de pasos es lo que se llama algoritmo, el cual necesita llevar como propiedad final la probabilidad de ser transcrito a un lenguaje de programación [17, 18].

Básicamente se pueden identificar como:

- Etapa de solución del problema.
- Etapa de implementación con un lenguaje de programación.

2.2.1. Análisis del problema

Este es la primera fase para dar solución a un problema, donde se debe de hacer un análisis, es decir, es necesario inspeccionar atentamente el problema para lograr obtener un concepto preciso acerca de lo que se necesita [18].

2.2.2. Diseño del algoritmo

En [17], se define un algoritmo como una serie ordenada de pasos, sin llegar a ser ambiguo, para poder llegar a la solución del problema previamente analizado, el cual puede ser mencionado en un lenguaje natural, por ejemplo, el español (ver Figura 2.2).

Un algoritmo, además de tener la propiedad de ser transcrito, también necesita ser:

- Preciso. Necesita llevar un orden en el se debe realizar cada paso para llegar a la solución.

- Definido. El resultado siempre debe ser el mismo, es decir, el resultado no puede cambiar si se encuentra bajo las mismas circunstancias del problema.
- Finito. El algoritmo debe tener un fin, no puede tener repeticiones sin sentido.

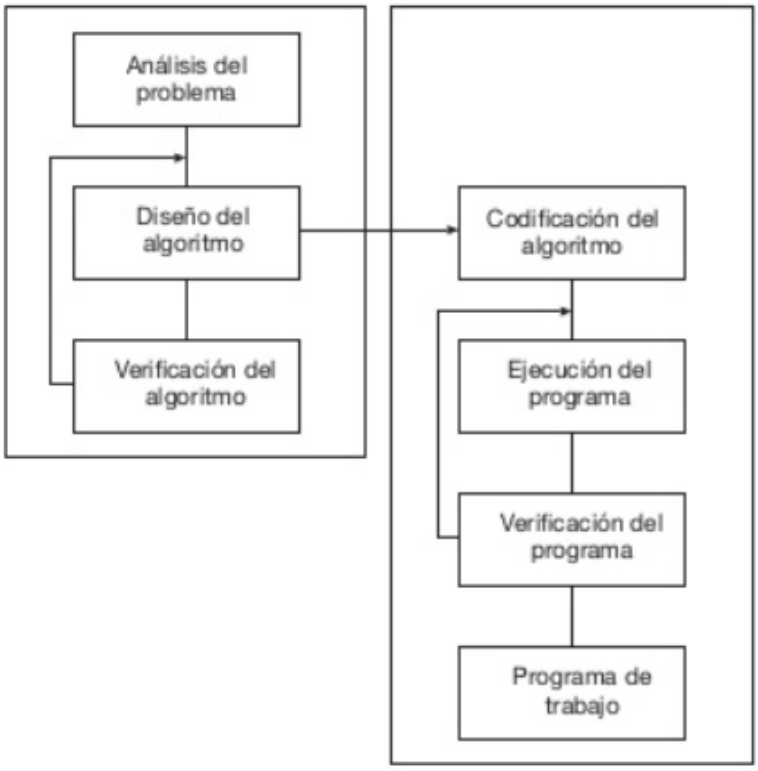


Figura 2.2: Solución de problemas mediante computadora.

2.2.3. Implementación

Cuando el algoritmo esté completado, debe ser representado gráficamente con un diagrama de flujo o pseudocódigo (véase en Herramientas de diseño de soluciones). Una vez comprobada está etapa se pasará a la codificación, es decir, se transcribe el algoritmo a un

lenguaje de programación y se concluirá con la ejecución y verificación de el programa en una computadora [17].

2.3. Fundamentos de programación

Se puede definir a los fundamentos de programación como las bases primordiales para aprender a programar.

2.3.1. Tipos de datos

En [19], los tipos de datos se definen como la propiedad de un valor que determina su dominio, es decir, los valores que puede tomar y como es representado internamente por la computadora.

En (Figura 2.3), se muestra un ejemplo de los tipos de datos en lenguaje de programación "C" y su rango de valor.

Tipo de datos en C	*Descripción*	*Rango*
`int`	Enteros	-32,768 a +32,767
`float`	Reales	3.4×10^{-38} a 3.4×10^{38}
`long`	Enteros de largo alcance	-2'147,483,648 a 2'147,483,647
`double`	Reales de doble precisión	1.7×10^{-308} a 1.7×10^{308}
`char`	caracter	Símbolos del abecedario, números o símbolos especiales, que van encerrados entre comillas.

Figura 2.3: Tipos de datos según el lenguaje de programación C.

2.3.2. Constantes y variables

En programación, una constante se define como una variable cuyo valor siempre es el mismo, es decir su valor no puede cambiar mientras se ejecuta el programa. Por esta razón se puede diferenciar de una variable [20, 21].

Por otra parte, las variables en programación son consideradas como una de las propiedades más potentes. Una variable se puede definir como un contenedor que se utiliza para almacenar datos en un programa. En realidad, una variable hace referencia a un espacio de memoria en la RAM de la computadora [20, 21].

2.3.3. Operadores y expresiones

En [22], Un programa está basado en la ejecución de varios cálculos con el fin de producir ciertos resultados, dichos resultados son almacenados en variables que a su vez son usados para nuevos cálculos, hasta obtener un resultado final. Los cálculos son procesados mediante expresiones, (Figura 2.4), compuestas de ciertos símbolos.

Los operadores se pueden categorizar en partes, aritméticos (Figura 2.5) y relacionales/-comparativos (Figura 2.6).

2.3.4. Herramientas de diseño de soluciones

Hoy en día la humanidad se enfrente a diversas situaciones de aprendizaje o retroalimentación, pero en repetidas ocasiones estos obstáculos pueden superarse ya sea por experiencia o por que se utilizó una alternativa adecuada. En este tema se tratará acerca de diseños de soluciones.

Operador	Tipo de Operador	Asociatividad
[] -> .	Binarios	Izq. a Dch.
! ~ - *	Unarios	Dch. a Izq.
* / %	Binarios	Izq. a Dch.
+ -	Binarios	Izq. a Dch.
<< >>	Binarios	Izq. a Dch.
< <= > >=	Binarios	Izq. a Dch.
== !=	Binarios	Izq. a Dch.
&	Binario	Izq. a Dch.
^	Binario	Izq. a Dch.
|	Binario	Izq. a Dch.
&&	Binario	Izq. a Dch.
||	Binario	Izq. a Dch.
?:	Ternario	Dch. a Izq.

Figura 2.4: Tipos de expresiones.

- valor	Menos unario
valor * valor	Producto (multiplicación)
valor / valor	División (entera o real según el tipo de operandos)
valor % valor	Módulo (resto de la división) (sólo tipos enteros)
valor + valor	Suma
valor - valor	Resta

Figura 2.5: Significado expresiones aritméticas.

valor < valor	Comparación menor
valor <= valor	Comparación menor o igual
valor > valor	Comparación mayor
valor >= valor	Comparación mayor o igual
valor == valor	Comparación de igualdad
valor != valor	Comparación de desigualdad

Figura 2.6: Significado expresiones relacionales/comparativas.

Diagrama de flujo

En [19], define al diagrama de flujo como una esquematización visual de un algoritmo, es decir que en muestra de manera gráfica los pasos que se deben seguir para lograr una solución a un problema. Para construir un diagrama de flujo es de suma importancia utilizar la simbología correcta, (Figura 2.7), dado que se va a partir de esta representación para realizar un programa mediante la codificación.

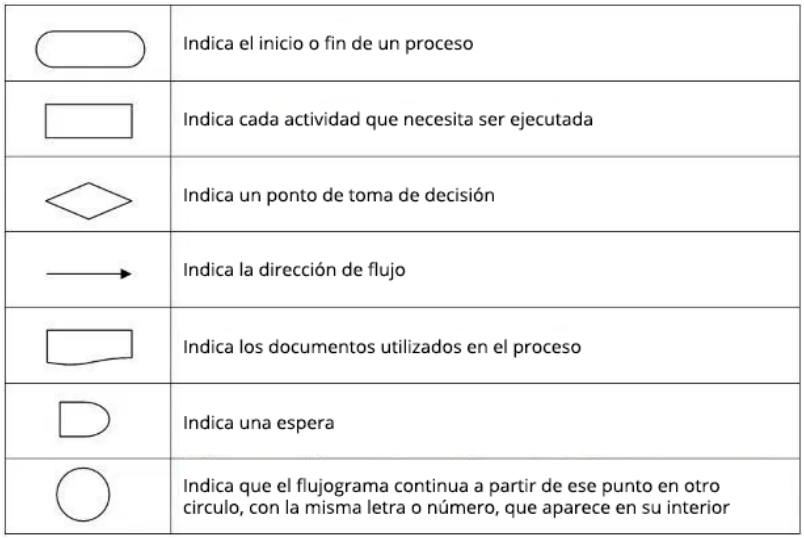


Figura 2.7: Simbología perteneciente a los diagramas de flujo.

Pseudocódigo

Otra de las herramientas usuales para representar un algoritmo es el pseudocódigo, puesto que es un instrumento de programación en donde los procesos son escritos utilizando palabras parecidas a un lenguaje (español o inglés) para facilitar la escritura y lectura de un programa informático [23].

En un pseudocódigo no hay reglas definidas para una escrituras, se pueden utilizar palabras simples reservadas para palabras reservadas de algún lenguaje, por ejemplo C, Java,

entre otros.

Por ejemplo, un programa solicita un número cualquiera a un usuario, y debe mostrar el resultado por pantalla (Figura 2.8).

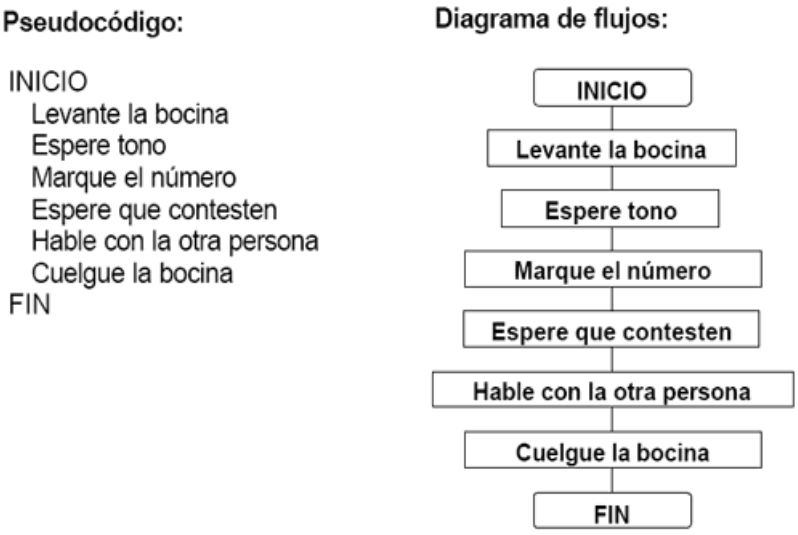


Figura 2.8: Representación en pseudocódigo del problema de ejemplo.

2.3.5. Estructuras algorítmicas repetitivas

Las estructuras algorítmicas repetitivas son de alta importancia al momento de resolver problemas, dado que estas ayudan a ejecutar determinado número de veces alguna operación para la solución de un problema. En programación existen tres tipos de estructuras algorítmicas repetitivas, las cuales se presentan a continuación.

Estructura repetitivas "for"

Es una estructura que permite asignar un número conocido de iteraciones, por lo tanto, no es necesario tener una condición de salida para detener el ciclo. Cuando el contador cumple con el número de iteraciones establecidas, el ciclo termina (Figura 2.9).

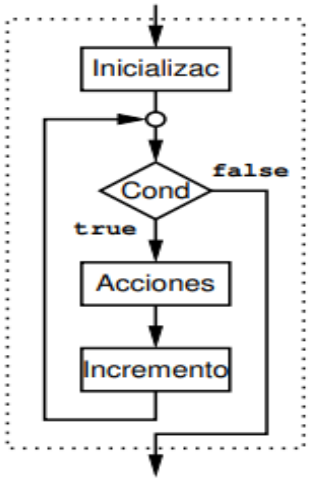


Figura 2.9: Representación de la estructura for.

Estructura repetitivas "while"

En [19], estructura "while" es una estructura repetitiva que es ejecutada mientras la condición predefinida sea completada. Cada que un nuevo ciclo inicia la condición es evaluada, si la condición se cumple se detiene, en caso contrario la condición será evaluada de nuevo (Figura 2.10).

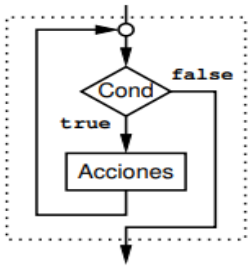


Figura 2.10: Representación de la estructura while.

Estructura repetitivas "do while"

En [22], define a la estructura como una sentencia que cuenta con una construcción específica para ciertas situaciones, donde se permite una solución más clara. Las diferencias con las estructuras anteriores son que las condiciones son evaluadas desde el inicio del ciclo, mientras que en ésta las evalúa al final (Figura 2.11).

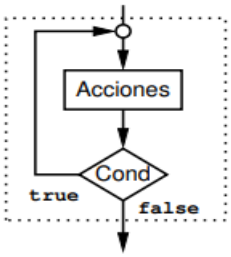


Figura 2.11: Representación de la estructura do while.

2.4. Objetos de aprendizaje

En [24], un objeto de aprendizaje (OA), se define como cualquier elemento que pueda ayudar el procedimiento de enseñanza a través de una tecnología. Los OA son la respuesta a una técnica educativa con base en la programación orientada a objetos, donde instrucciones independientes puedes ser utilizadas varias veces, pero en diferentes contextos, satisfaciendo así los fundamentos del encapsulado, herencia y abstracción (ver Figura 2.12).

En la actualidad los OA son considerados como un recurso barato por su propiedad de ser reutilizados, es decir son adaptables a las necesidades del usuario. Los objetos de aprendizaje pueden componerse de diferentes elementos como imágenes, audio, vídeos, texto.

Figura 2.12: Estructura de un objeto de aprendizaje.

Metadatos

En [25], define a los metadatos como un conjunto grupo de elementos inevitables para describir un recurso. Son específicamente convenientes en recursos no textuales, los metadatos clasifican como:

- Metadatos descriptivos: descubren, identifican y seleccionan, es decir los metadatos descriptivos ayudan a formar colecciones de recursos similares.
- Metadatos administrativos: son la información que es facilitada a la administración de recursos. Contienen información sobre el recurso acerca de su creación y su forma de ser creado y además incluye datos técnicos de software o hardware para ser ejecutado.
- Metadatos estructurales: su función es identificar las partes que componen un recurso, como es su estructura y que le da forma. Son usados específicamente para el procesamiento de la máquina.

2.5. Usabilidad

La usabilidad puede definirse como una disciplina encargada de estudiar la manera de diseñar un sitio web fácil de usar, intuitivo y amigable hacia el usuario. En [26], plantea que el término usabilidad se refiere al nivel de facilidad para usar una herramienta, aplicación o producto.

Importancia de la usabilidad

En [27, 28], se describe la importancia de la usabilidad para la supervivencia, es decir si no es aplicada en un sistema web, este podría tornarse en un sitio muy complicado de manipular, por lo tanto, las personas evitarán usarlo. Si una página web no es clara con su propósito, las personas no mostrarán interés en ella. Otra de las razones por la no se tiene éxito es por que la información puede ser muy difícil de leer o comprender. Incluso si una persona se pierde al navegar por un sitio web, probablemente lo abandone y no vuelva,(Figura 2.11).

2.5.1. Métodos para evaluación de usabilidad

Método de sondeo

Esta técnica según [29], se basa en la observación de el usuario, es decir ser atento a la forma de trabajar y de las respuestas a ciertas preguntas. Este método se puede realizar con entrevistas independientes a los usuarios, con el objetivo de conocer el nivel de satisfacción que sienten con el sitio web a desarrollar. Este método puede también llevarse acabo en forma de grupos de discusión de no más de ocho personas y luego tomar las opiniones de los usuarios acerca del sitio web.

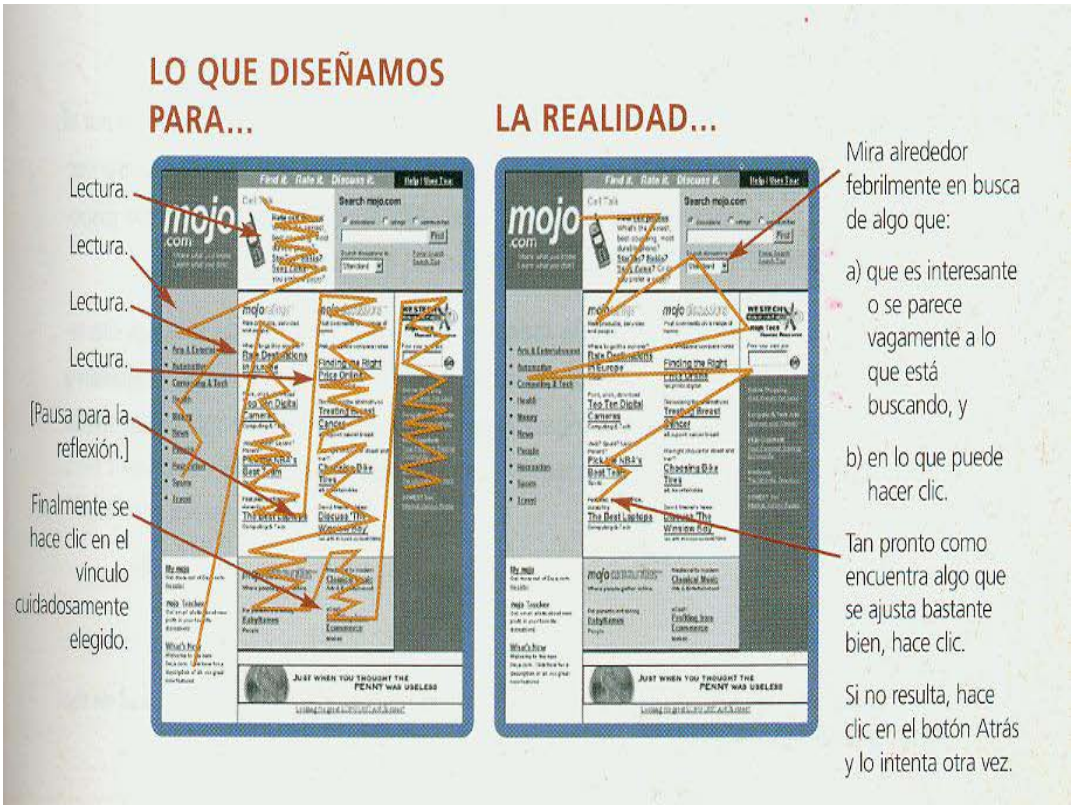


Figura 2.13: Ejemplo planteado en el libro "No me hagas pensar"de Steve Krug.

Método de inspección

En [30], está fundamentado en el análisis de profesionales directamente sobre el prototipo, donde una de las técnicas más comunes para dicha evaluación es la llamada “heurística” dado que es un método rápido y barato pues es basado en los principios establecidos de la usabilidad. Otra forma consiste en realizar una verificación de interfaz para ser comparada con los estándares establecidos por la "W3C"

2.5.2. Accesibilidad web

En [31], se define la accesibilidad web como el talento de asegurar que un sitio web sea concurrido y usado de una forma eficaz por la mayor cantidad de personas posibles, sin importar el tipo de limitaciones que tenga el usuario o su entorno.

La mayoría de las veces se identifica web como la accesibilidad enfocada a personas con una discapacidad, cuando en realidad se refiere a ofrecer un “acceso universal a la web, independientemente del tipo de hardware, software, red, idioma, cultura y capacidades del usuario” [32].

- Limitaciones personales: no siempre se refieren a una discapacidad del cuerpo. Se pueden derivad a partir de la edad. El nivel de experiencia con la tecnología, el idioma
- Limitaciones tecnológicas: Pueden ser conexión a internet deficiente, ambiente con ruido, sin compatibilidad a imágenes, falta del ratón de la computadora, entre otras cosas.

Para hacer un sitio web accesible es necesario hacerlo sencillo o complicado, es decir, varía según los factores, por ejemplo, la clase de contenido, el tamaño del sitio, el entorno de desarrollo, entre otras cosas. La mayoría de las propiedades que hacen accesible a un sitio web son agregadas de manera sencilla, siempre y cuando sean planeadas desde un principio del proyecto o en su rediseño [32].

En [33], se establecen las reglas para implementar accesibilidad en un contenido web, las cuales han sido implementadas por la Web Accessibility Initiative (WAI), quien es un organización independientes de W3C y que definen como implementar contenido web accesible. Además, son un estándar de clase internacional en muchos países, incluso desde 2012 son parte de un estándar ISO.

Capítulo 3
Desarrollo del Proyecto

En el presente capítulo se abordan las herramientas y métodos necesarios para concluir con el desarrollo de este proyecto. En la primera sección se explica el producto que se propone como solución, la siguiente sección muestra la metodología seleccionada para desplegar el proyecto. Posteriormente, se describen las fases de la metodología aplicada durante el proyecto, donde se observa la evolución y avance en cada etapa de la metodología, la cual se compone de cuatro fases: inicio, elaboración, construcción y transición.

3.1. Producto propuesto

Se propuso desarrollar un prototipo de aplicación web que ofrezca contenido didáctico sobre los principios básicos de la programación, que permita a los estudiantes reforzar, aprender y aplicar métodos o técnicas para la resolución de problemas mediante la codificación.

El prototipo fue diseñado para abarcar los conocimientos sobre la asignatura de fundamentos de programación, a través de pasos ya establecidos para resolver un problema mediante la programación, los cuales son análisis del problema, desarrollo del algoritmo, implementación del pseudocódigo y codificación.

Como parte del contenido se ofrece la opción de realizar una evaluación general de la

materia, por unidad y por tema, con el fin de medir el nivel de progreso del estudiante. Además, el proyecto fue desarrollado con la intención de poder enriquecer su contenido, es decir, es posible agregar nuevas asignaturas mediante el rol de administrador, con el fin de ofrecer al usuario una experiencia completa.

3.2. Descripción de la metodología

La metodología seleccionada para desarrollar el proyecto es conocida como "Rational Unified Process" (RUP). Este método (Ver figura 3.1), es un proceso de ingeniería de software formulado por Philippe Kruchten en 1996. El objetivo de este método es el de producir software de alta calidad, es decir, que satisfaga los requerimientos de los usuarios dentro de una planificación y presupuesto establecidos, la cual cumple con el ciclo de vida de desarrollo de software.

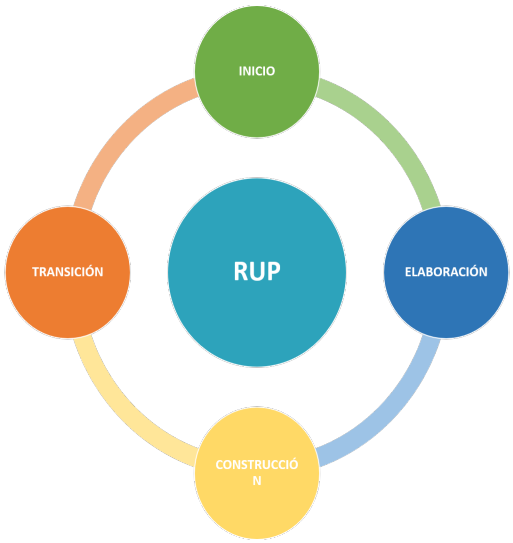


Figura 3.1: Diagrama del modelo RUP.

El motivo por el cual se seleccionó esta metodología es debido a que se adapta bien al proceso del desarrollo de prototipo de aplicación web.

3.2.1. Fase de inicio: alcance del proyecto

Para la elaboración del proyecto se definieron los siguientes parámetros de alcance, el primero de ellos fue determinar el tipo de usuario interesado en utilizar el programa, los cuales fueron ofrecer una herramienta alternativa para aprender sobre los fundamentos de programación enfocado a estudiantes, profesores de nivel de educación básico, media superior, universitario cualquier y persona interesada.

3.2.2. Fase de elaboración: análisis y diseño

En esta sección se muestran las diversas herramientas de software utilizadas para desarrollar el proyecto, (ver la Tabla 3.1). Además, se especificarán las características del equipo de hardware utilizado para realizar y soportar el prototipo de un sistema de apoyo para aprender a programar, (ver la Tabla 3.2).

Herramientas utilizadas

Tabla 3.1: Tabla de herramientas de software para el desarrollo del proyecto.

Aplicación	Descripción
Visual Studio Code	Editor de texto y de código fuente
Google Chrome versión 76.0.3809.132 (64 bits)	Navegador web
HTML5	Quinta revisón del lenguaje básico html
CSS3	Lenguaje para definir estilos de html
PHP v7.3.9	Lenguaje prosesador de hipertexto
Laravel	Framework de desarrollo para PHP
Composer	Administrador de paquetes de PHP
Laragon	Entorno de desarrollo para PHP
MySql	Gestor de base de datos relacional
Justinmind	Herramienta para diseño de prototipos
Vue	Marco de JavaScript

Tabla 3.2: Tabla de características de hardware del equipo de cómputo.

Hardware	Equipo 1
RAM	2x4GB DDR3
Prosesador	I5 3570k
Arquitectura	64-bit
Disco duro	WD 1tb
Unidad de estado solido	Raid 0 2x120gbPNY

Análisis del contenido a abordar

Se realizó el análisis de los capítulos a abordar, basados en la carta descriptiva (modelo educativo visión 2020), de la materia de fundamentos de programación. por lo tanto la estructura principal quedo de la siguiente manera, (ver, Tabla 3.3) :

Tabla 3.3: Tabla de contenido a abordar.

Materia	Tema	Unidad
Fundamentos de programación	El proceso de programación	tipos de datos
		Constates y variables
		Algoritmo
		Diagrama de flujo
		Pseudocódigo
	Control de flujo de datos	Estructura secuencial
		IF
		IF-ELSE
		CASE
	Ciclos iterativos	Contadores
		Estructura FOR
		Estructura WHILE
		Estructura DO UNTIL
	Estructura de datos	Arreglos unidimencionales

Diseño de interfaz de usuario

Se efectuó el diseño de la interfaz de usuario del proyecto utilizando la herramienta Justinmind, la cual es una herramienta para diseñar y generar prototipos de proyectos web y aplicaciones. Básicamente se desea tener un área de trabajo en el centro que permita mostrar las cuatro fases para resolver un problema, (ver Figura 3.2).

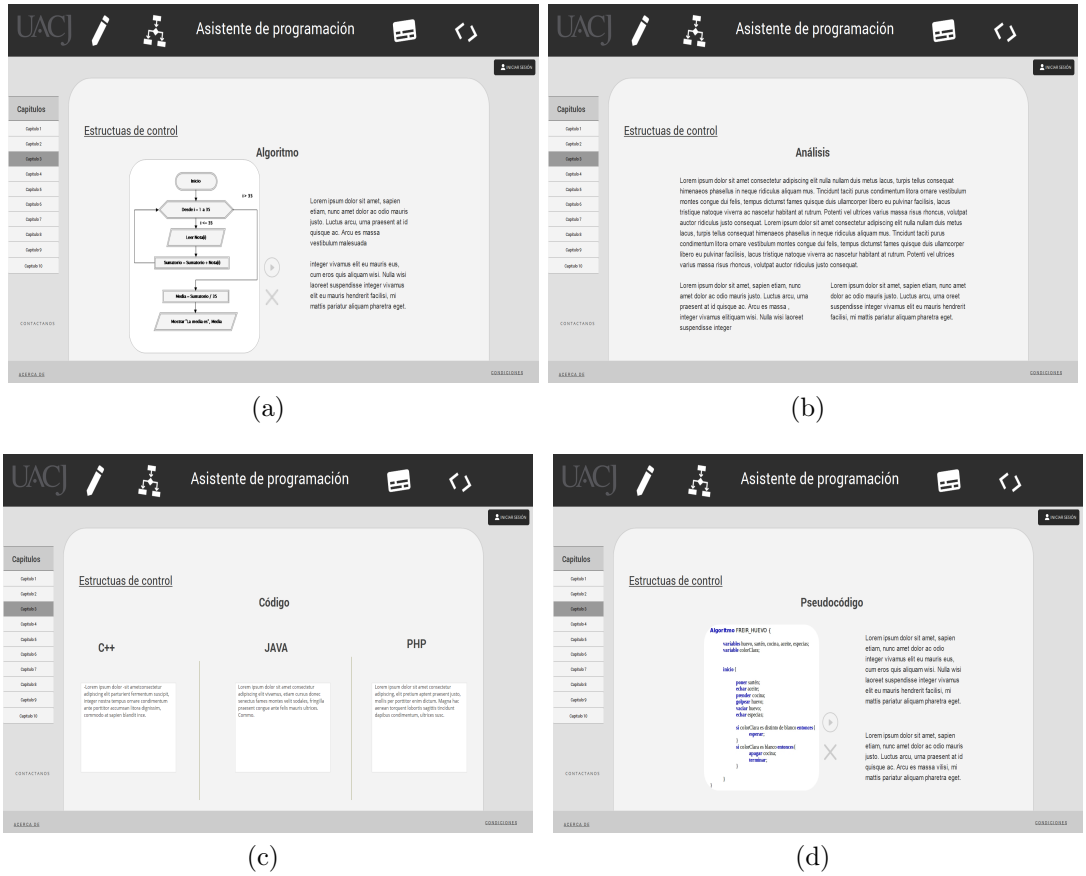


Figura 3.2: Diferentes pantallas de la aplicación

Diseño de la base de datos

Se concluyó el análisis y diseño de la base de datos según las necesidades del prototipo, a continuación se muestra el desarrollo de los puntos fundamentales para diseñar la base de datos, es decir, la implementación de un universo del discurso, personajes implicados, requerimientos, entre otras cosas.

Se llevó a cabo un análisis sobre las entidades implicadas que existentes en el proyecto. A continuación, se muestran los personajes que conforman el universo del discurso, los cuales se dividen en estudiante y administrador.

- Administrador. Es el encargado del desarrollo y mantenimiento de el prototipo. Sus actividades incluyen agregar nueva información, problemas y/o contenido.
- Estudiante. Es el usuario principal del prototipo, sus actividades implican interactuar con el contenido del sistema, de manera que puede consultar y/o resolver problemas establecidos.

El prototipo requiere el manejo de una base de datos para almacenar los temas, unidades, temas y el nombre de los documentos pdf, los cuales muestran el contenido de cada tema. Cada documento se compone de una introducción al tema, análisis, algoritmo, pseudocódigo y código en diferentes lenguajes establecidos. Se debe tomar en cuenta que existe el caso de restringir el acceso a los datos como medida de seguridad (modificar el contenido).

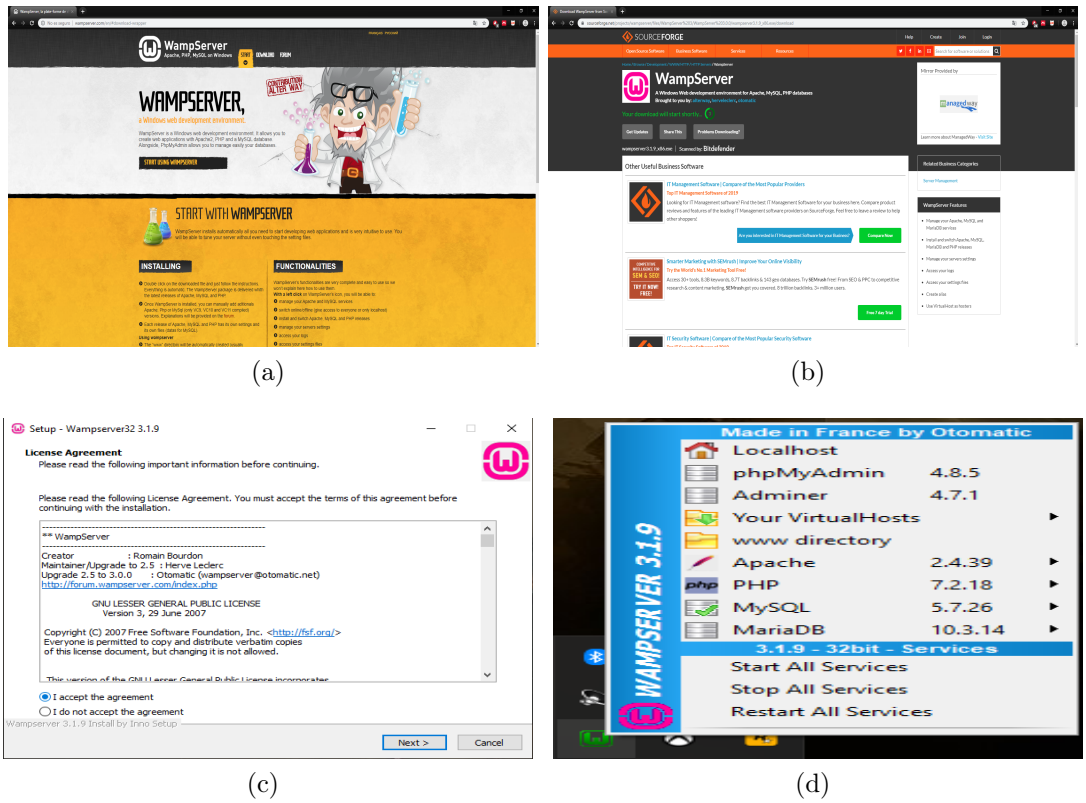
- Existe una materia.
- Cada materia tiene varias unidades.
- Cada unidad tiene varios temas.
- Hay dos tipos de rol de usuario, estudiante y administrador.

3.2.3. Fase de desarrollo: implementación

Antes de iniciar con el desarrollo de la interfaz de usuario, es importante comentar que hay varias maneras de crear un servidor que almacene y muestre el proyecto, en este caso, se explicarán dos métodos distintos con las herramientas WAMP Y Laragon. Ambos métodos cumplen con su función de servidor (Apache), cuentan con un gestor de base de datos (Mysql) y el lenguaje de programación de script (Php).

Metodo 1 (Instalación del servidor con WAMP)

Primero se descargó la herramienta WAMP desde la página oficial WampServer.com, después se comenzó la instalación, como se muestra en la Figura 3.3.



(a) (b)

(c) (d)

Figura 3.3: Proceso de instalación de la herramienta WAMP.

Una vez instalado y abierta la herramienta WAMP, se procedió a crear el proyecto dentro de la carpeta del servidor, la cual por defecto se encuentra en la dirección c/wamp/www. Dentro de esta carpeta se creó el arhivo index.php el cual contendrá la interfaz de usuario, después se generó otro documento para almacenar los estilos de la interfaz, (ver Figura 3.4).

Metodo 2 (Instalación del servidor con Laragon)

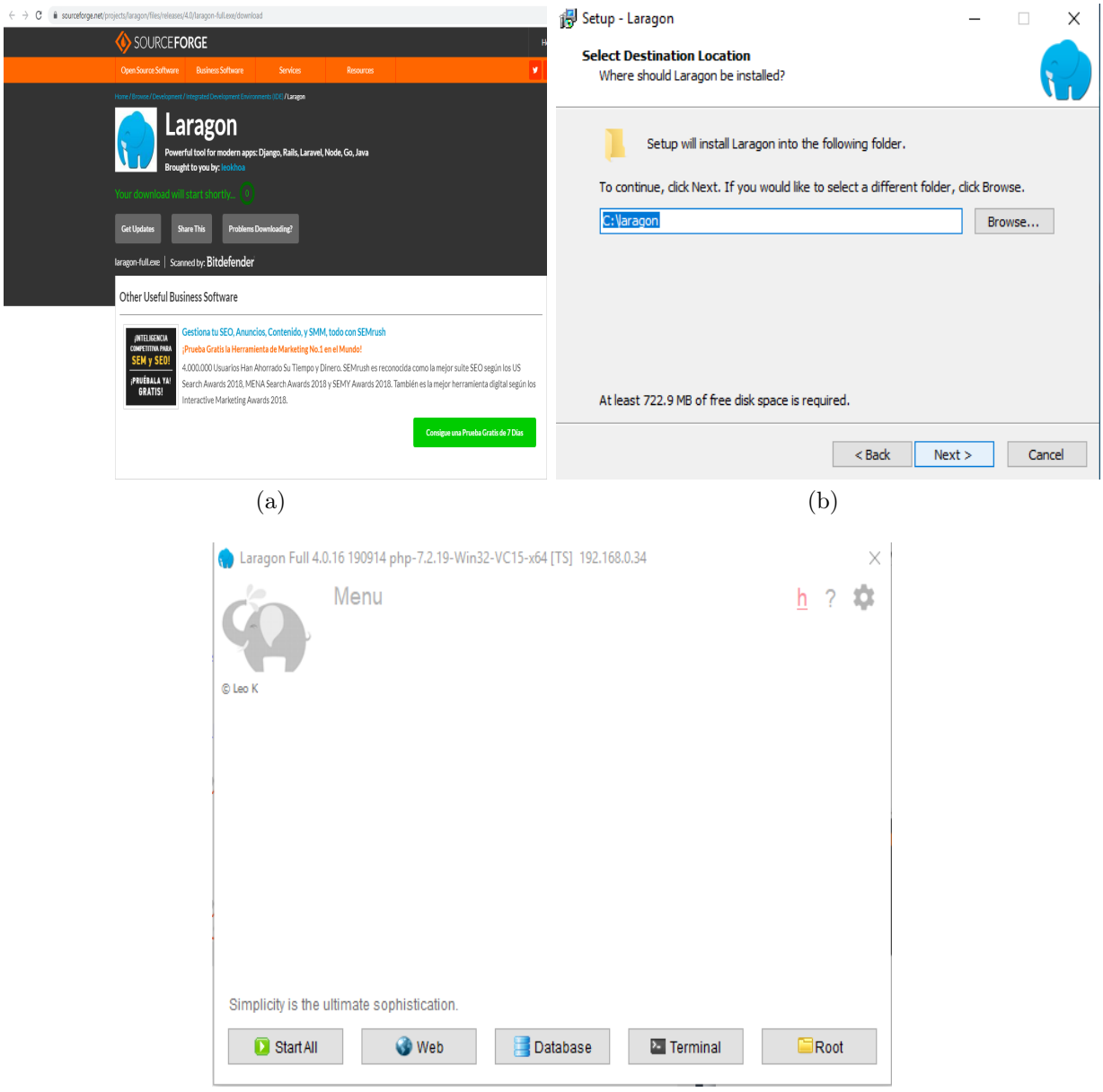
En muchas ocasiones el método 1 (WAMP), suele presentar problemas posteriores a la instalación, es decir que requiere de ciertas configuraciones adicionales o archivos de biblioteca de enlace dinámico, mejor conocidas por sus siglas en inglés "DLL".

Debido a esto, otra forma más segura y fácil de crear el servidor fue utilizando la herramienta Laragon, dado que este entorno de desarrollo PHP, fue creado específicamente para trabajar con el framework Laravel. Además, Laragon permite utilizar otras herramientas de desarrollo extras a las ofrecidas por WAMP y necesarias para el desarrollo de este proyecto, las cuales son:

Tabla 3.4: Tabla de herramientas compatibles con Laragon.

Herramienta	Descripción
Cmder	Consola de comandos
Node.js	Entorno JavaScript para servidor
Npm	gestor de paquetes para Node.js
PHP 7	Lenguaje procesador de hipertexto
Vue	Marco de JavaScriptpara crear aplicaciones

Para comenzar con la instalación, primero se descargó el ejecutable de Laragon desde su página oficial (laragon.com/download). Durante el proceso de descarga e instalación, se puede apreciar que se tiene gran similitud con la herramienta WAMP. Con la diferencia de que una vez concluida la instalación, no se generan errores o requiere configuraciones adicionales para su funcionamiento, (ver Figura 3.4 a).



(a) (b)

(c)

Figura 3.4: Proceso de instalación de la herramienta Laragon.

Instalación de PHP

Una vez instalada la herramienta de servidor, se procedió a instalar en la computadora el preprocesador de hipertexto (PHP). Para esto fue necesario descargar la última versión más estable, en este caso la versión 7.3.9.

Se realizaron los siguiente pasos:

- Ingresar a la página oficial de PHP y descargar.
- Descomprimir el archivo en la carpeta raíz del disco local C.
- Renombrar la carpeta con "PHP".
- Abrir el editor de variables de windows.
- Pegar la ruta *C:/php* dentro de PATH

Una vez instalado, se puede comprobar que se tuvo éxito en la instalación ejecutando el siguiente comando en la consola de windows (cmd), ver Figura 3.5.

```
C:\>php -version
```

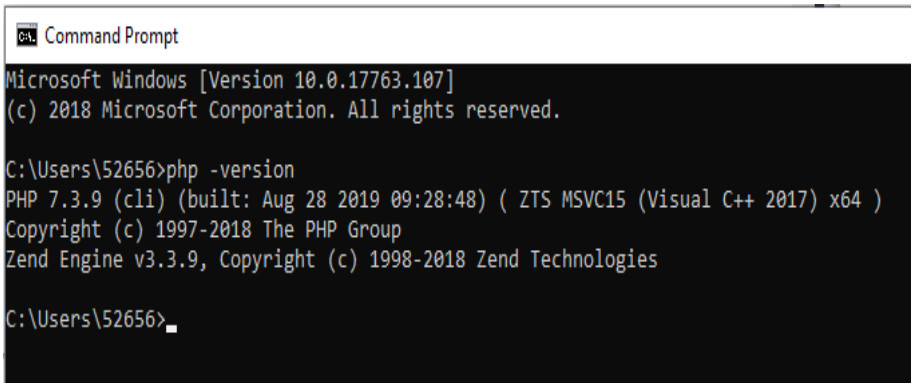
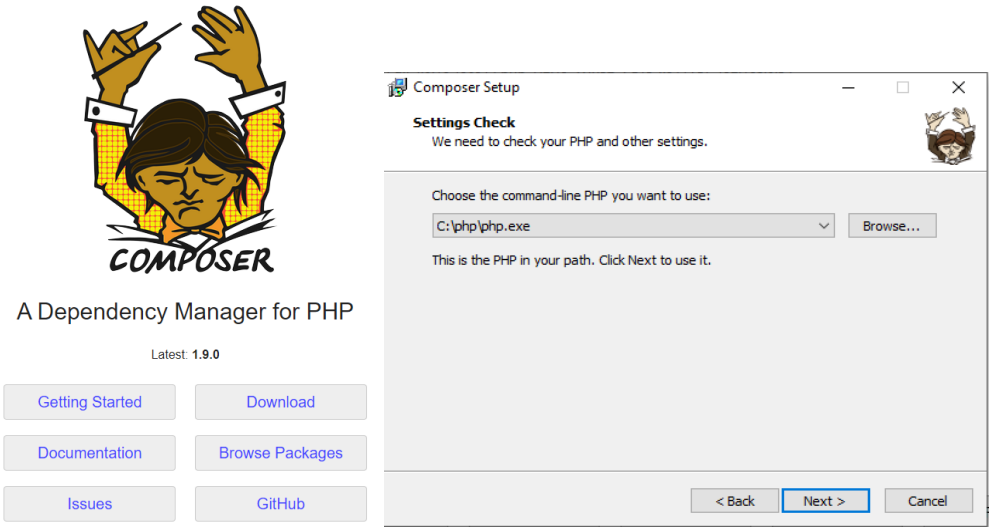


Figura 3.5: Revisión de la versión instalada de php

Instalación de Composer

Con PHP instalado correctamente en la computadora, se procedió a instalar el gestor de paquetes de PHP, para hacerlo es necesario descargar ejecutable desde su página oficial, nuevamente el proceso de instalación fue muy parecido a las anteriores hasta este punto, ver Figura 3.6.



(a) (b)

Figura 3.6: Proceso de instalación de la herramienta Composer.

También se pudo verificar su correcta instalación con el comando:

```
C:\>composer --version
```

Creación de un proyecto con el framework Laravel

La descarga de Laravel fue posible mediante el gestor de paquetes Composer. Como se planeó utilizar otras herramientas de desarrollo semejantes a Vue, es indispensable crear un proyecto de Laravel con una versión superior a 5.8, para hacerlo, es necesario ejecutar el siguiente comando en la consola de Windows, ver Figura 3.7.

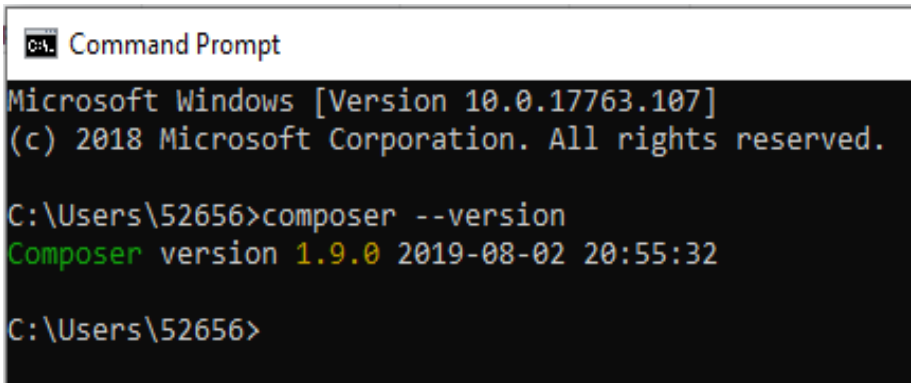


Figura 3.7: Revisión de la versión instalada de Composer.

```
C:\>composer create-project laravel/laravel proyecto "5.8.35"
```

Al ejecutar el comando anterior, básicamente se comenzó a descargar y crear un proyecto de Laravel en el disco local C, dentro de una carpeta de nombre "proyecto", con la versión específica 5.8.35. En la Figura 3.8, se puede observar el proceso de descarga de el proyecto, es posible que esta tarea dure de 3 a 5 minutos en completarse, por lo que se recomienda ser paciente y no interrumpir el proceso.

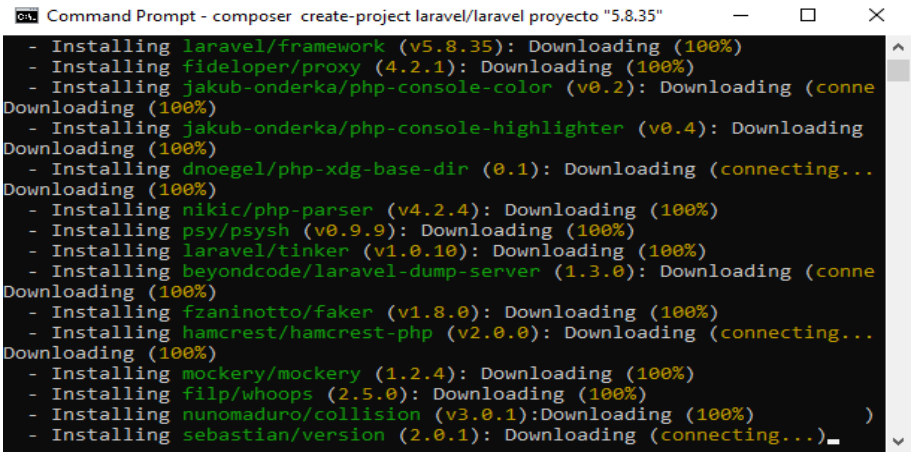


Figura 3.8: Proceso de descarga y creación de un proyecto de Laravel.

En este punto ya se pudo empezar a trabajar en el proyecto generado con el framework Laravel, para hacerlo, fue necesario utilizar un editor de código. En este caso se seleccionó

Visual Studio Code debido a que es cómodo y agradable de utilizar con proyectos de este estilo, dado que cuenta con una consola de comandos propia .

Se debe asegurar de iniciar la herramienta Laragon, luego abrir proyecto con Visual Studio Code y levantar el servidor con el comando:

```
C:\proyecto>php artisan serve
```

Como la aplicación no tiene problemas de compilación, la terminal o consola de comandos devolvió el siguiente mensaje:

```
Laravel development server started: <http://127.0.0.1:8000>
```

Esto significa que se levantó el servidor con éxito y ahora es posible ver el proyecto, ingresando en el navegador la url que muestra http://127.0.0.1:8000, (ver Figura 3.9).

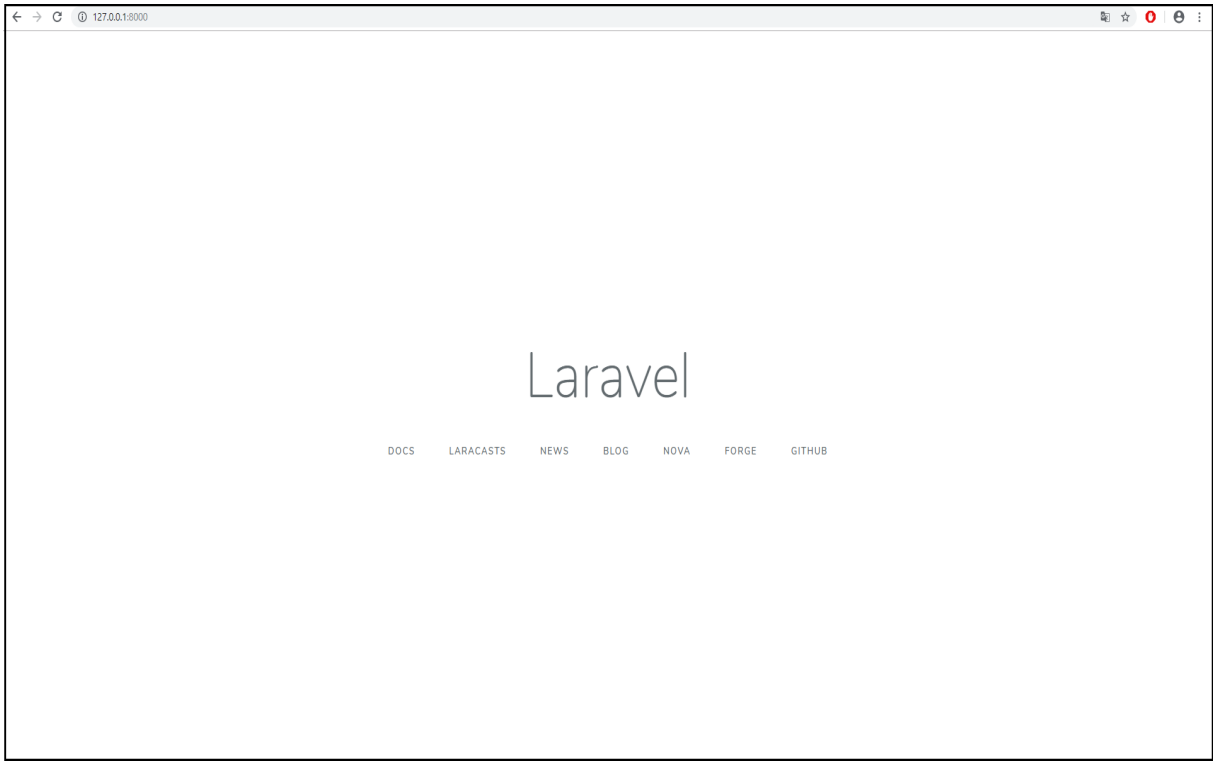
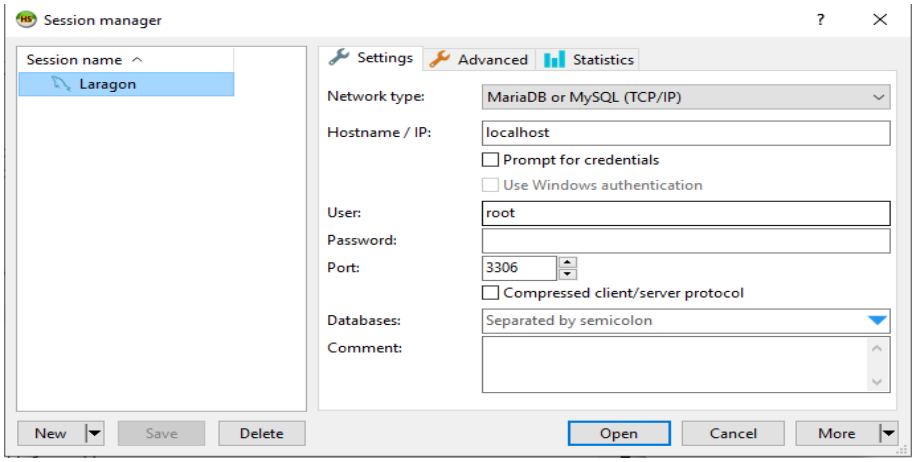


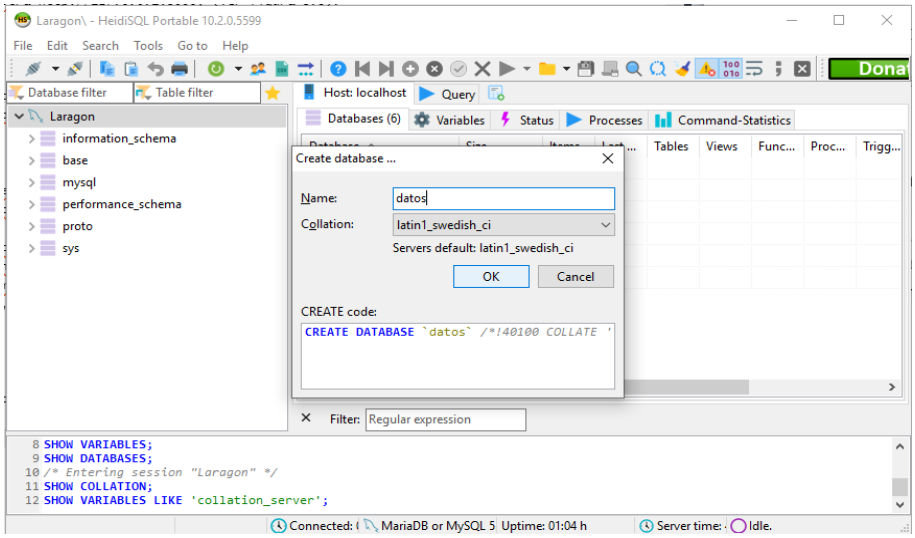
Figura 3.9: Página de ejemplo "Welcome".

Autenticación de usuarios

Una vez verificado el funcionamiento del proyecto, fue necesario generar un sistema de autenticación de usuarios. Para esto, primero se creó una base de datos y luego se realizó la conexión de la base con el proyecto de Laravel. En la Figura 3.10, se puede observar la creación de la base de datos en Laragon.



(a) Configuraciónes de el manejador de la base de datos.



(b) Creación de la base de datos de nombre base.

Figura 3.10: Creación de la base de datos.

Posteriormente se realizó la conexión de la base de datos, esto se hizo modificando los parámetros del archivo de nombre .env, en la raíz de la carpeta del proyecto. En este caso solo es necesario modificar el nombre de la base de datos (base) en DBDATABASE, el usuario (root) en DBUSERNAME.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=base
DB_USERNAME=root
DB_PASSWORD=
```

Con esta configuración, el proyecto quedó enlazado a la base de datos en la herramienta Laragon. Después, es posible crear la autenticación de manera fácil y rápida con Laravel, solo se debe ejecutar el siguiente comando en la terminal:

```
C:\proyecto>php artisan make:auth
```

Posterior a este comando, se generaron archivos con etiquetas y estilos de Boostrap. Luego, se hicieron las migraciones a la base de datos, las cuales crean las tablas necesarias para que la autenticación funcione de forma correcta. El comando utilizado es el siguiente:

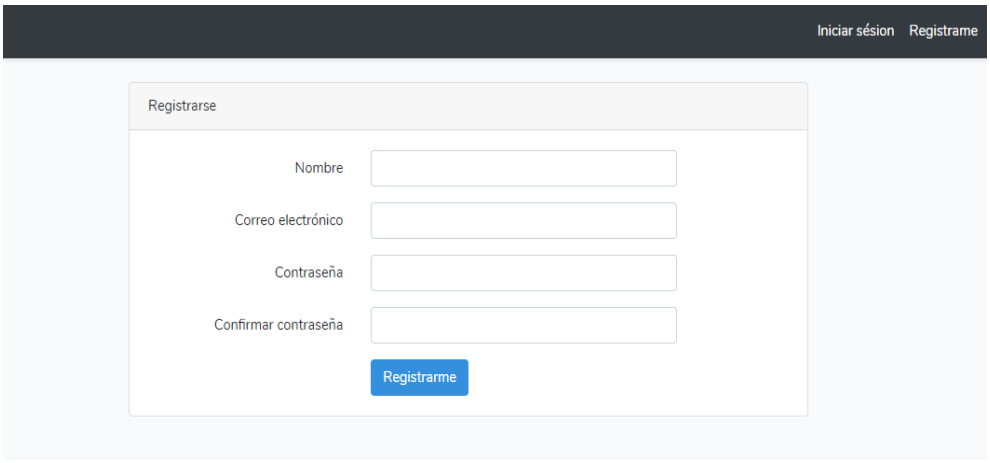
```
C:\proyecto>php artisan make:migration
```

Seguidamente, al compilar el proyecto aparecieron en la esquina superior derecha botones para iniciar sesión o registrar una cuenta de correo, (ver Figura 3.11).

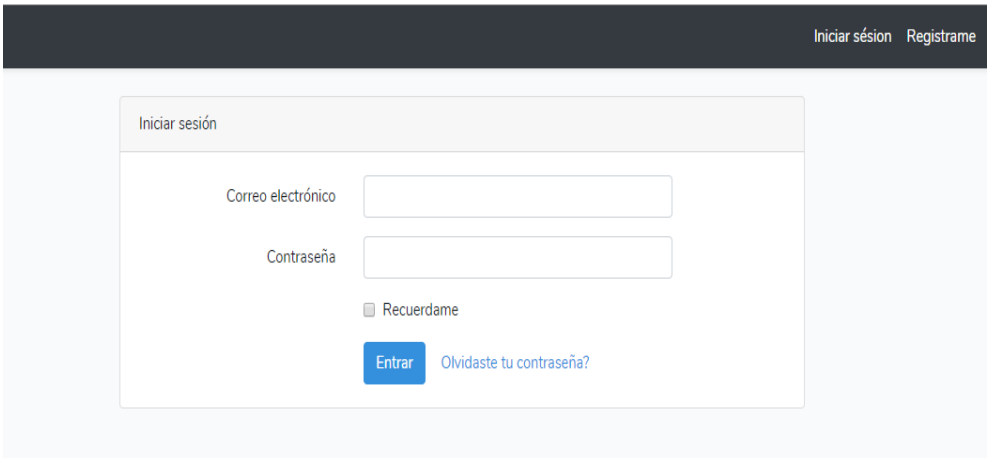
Figura 3.11: Botones generados para la autenticación.

Adicionalmente se crearon dos formularios, el primero corresponde al registro de una nueva cuenta de correo para obtener acceso, (ver figura 3.12a). El segundo formulario pertenece al inicio de sesión, el cual permite entrar a otras secciones del proyecto, (ver Figura 3.12b).

Ambos formularios fueron modificados y adaptados al estilo de clases y colores definidos en el diseño del prototipo.



(a) Registro de cuenta de correo



(b) Formulario de inicio de sesión.

Figura 3.12: Formularios de autenticación.

Pantalla de presentación

En esta sección se comenzó a desarrollar una pantalla principal, cuya función fuese mostrar una breve pero atractiva descripción del proyecto. Gracias a las propiedades y facilidades que ofrece Laravel, es posible extender o reutilizar fragmentos de código ya escritos.

```
@extends('layouts.app')
```

En este caso, se exportó el código que se generó con la autenticación, manteniendo todas sus clases, estilos y la funcionalidad propia de autenticación. El resultado es el que se muestra en la Figura 3.13.

Figura 3.13: Página de bienvenida.

Solo fue necesario agregar al código fuente un pie de página, con esto se terminó la estructura básica que fue utilizada en todas aquellas pantallas nuevas.

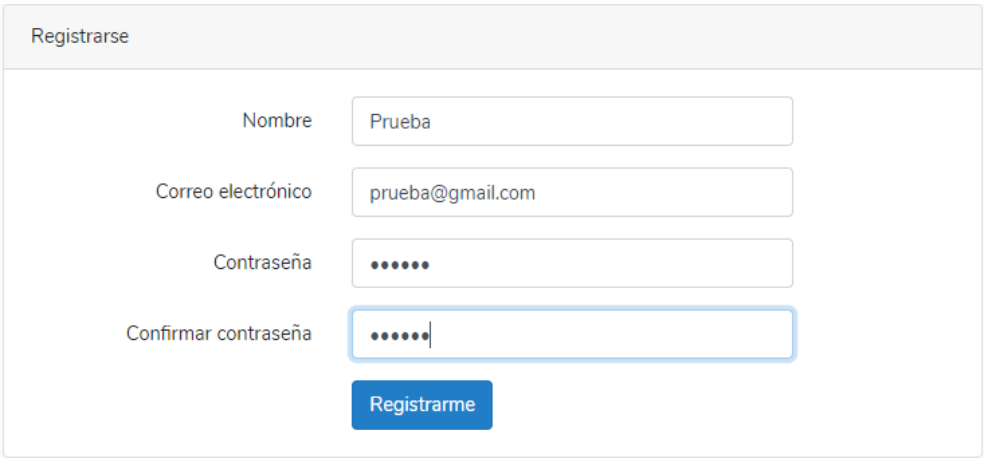
Más tarde fue necesario desarrollar una estructura con tablas en Html y Boostrap para terminar la pantalla de inicio. También, se desarrolló e implementó un logotipo para el proyecto, el cual puede apreciarse en la figura 3.14.

Figura 3.14: Página de bienvenida terminada

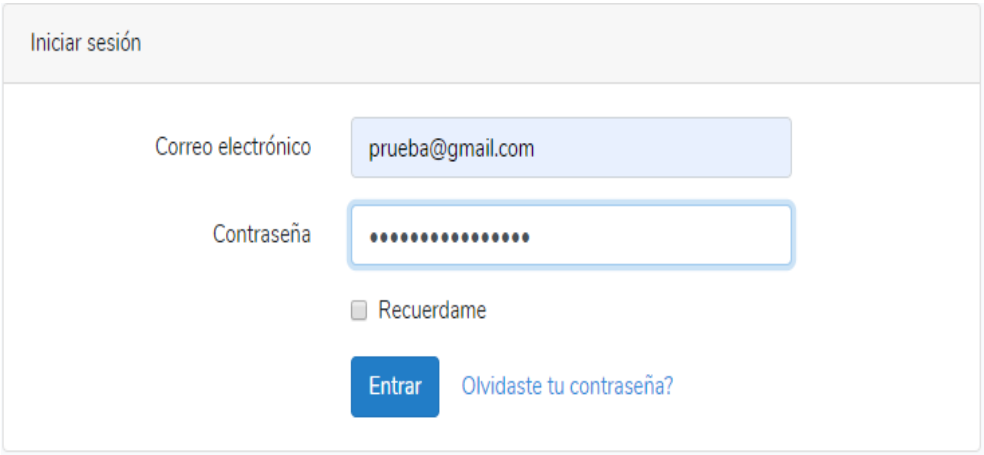
Debido a que la pantalla de inicio será accesible a todo público, es decir que no requiere autenticación para ser visualizada.

Pantalla de trabajo

Para desarrollar la pantalla de trabajo, se creó de la misma manera que en la sección anterior una nueva pantalla que importe el código fuente básico. La única diferencia es que esta pantalla requiere de autenticación, por lo tanto, si se desea visualizarla primero, se debe crear una cuenta y después iniciar sesión, (ver Figura 3.15).



(a) Registro de cuenta de correo de prueba.



(b) Formulario de inicio de sesión.

Figura 3.15: Autenticación de prueba.

Por lo tanto, obtuvo nuevamente una pantalla base, solo que ahora se cuenta con los permisos para acceder a la página. Ahora se implementará la funcionalidad que se explicará más adelante.

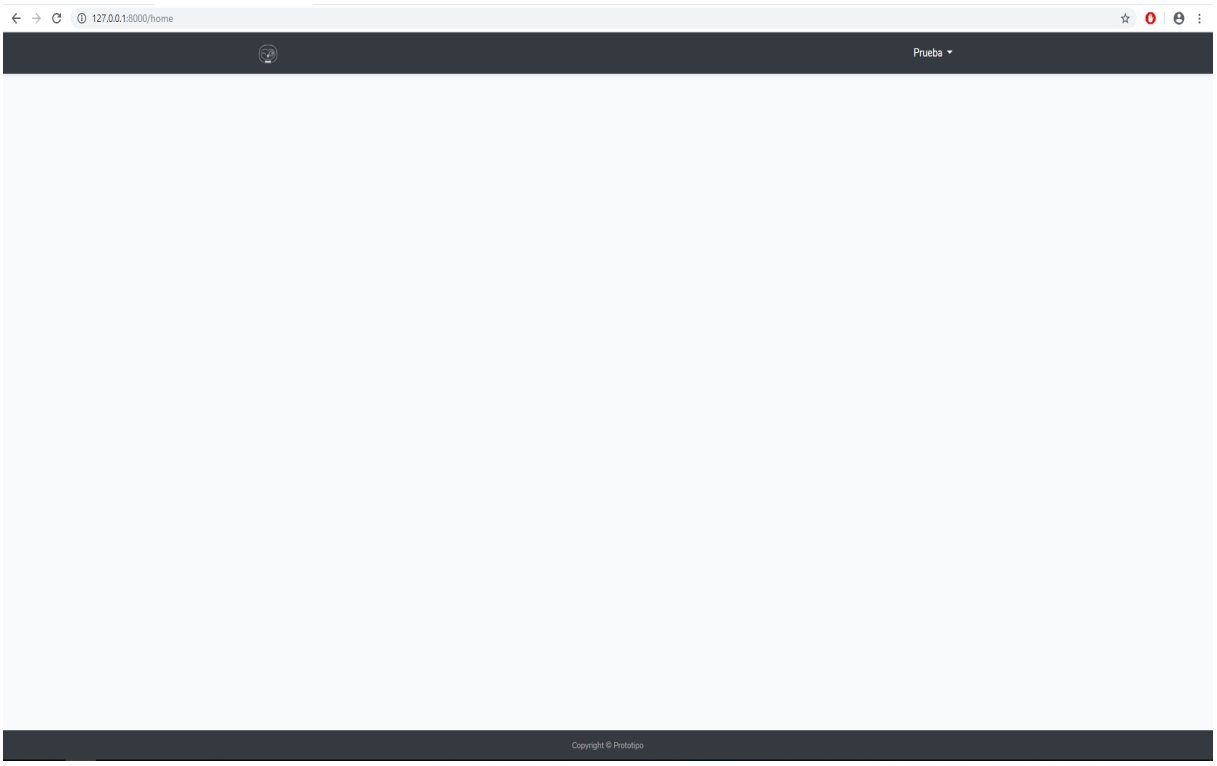


Figura 3.16: Página de trabajo.

Anteriormente, en la fase de diseño del prototipo se mostró que se utilizarían listas de selección dinámicas para seleccionar las materias, unidades y temas. Antes de implementarlas es importante instalar la herramienta Vue, puesto que tiene la propiedad de crear componentes, los cuales son necesarios para dar funcionalidad y reactividad a las listas.

Por lo tanto, fue requerido interrumpir el desarrollo del proyecto por unos breves momentos. Primero se descargó e instaló Node.js, el cual es un entorno de ejecución de JavaScript. Nuevamente el proceso de instalación fue similar al de las herramientas anteriores. Además no debería presentarse ningún problema (ver Figura 3.17).

(a) (b)

Figura 3.17: Proceso de instalación de la herramienta Node.js.

Después se debe instalar el siguiente comando dentro de la terminal o consola de comandos y se puede proseguir con el desarrollo de la pantalla de trabajo.

```
C:\proyecto>npm i install
C:\proyecto>npm install vue
```

Una vez adquirido Vue, se procedió a crear un componente para cada lista, donde el primer paso es agregar al archivo de rutas dentro del proyecto llamado web.php las rutas correspondientes a cada componente. Las rutas son importantes dado a que es el medio por donde se transportarán los datos desde la base a los componentes.

```
Route::get('/materias', 'DynamicDependent@index');
Route::get('/unidad/{id_materia}', 'DynamicDependent@unidad');
Route::get('/tema/{id_unidad}', 'DynamicDependent@tema');
```

En seguida, se creó un controlador para realizar las consultas de MySql para extraer los datos de la base, a continuación se muestra un ejemplo de una consulta correspondiente a la lista de materias.

```
function index()
    {
     $materia= DB::table('t_materia')
     ->select('id_materia','nombre_mat')
     ->get();
     return ($materia);
    }
```

De esta manera se logra que el componente obtenga la información de la tabla materias y la muestre dentro de una lista, ver figura 3.18.

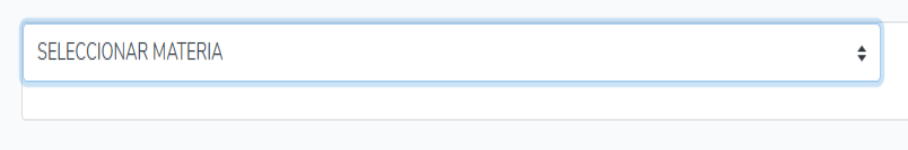


Figura 3.18: Lista de materias.

Por lo tanto, se crearon componentes hijos adentro de este componente, para que fuera posible enviar los datos relevantes como el id de la materia y con esto posteriormente el componente hijo pudiera obtener los datos de la tabla unidad. Este proceso fue repetido hasta obtener lo siguiente, ver Figura 3.19.

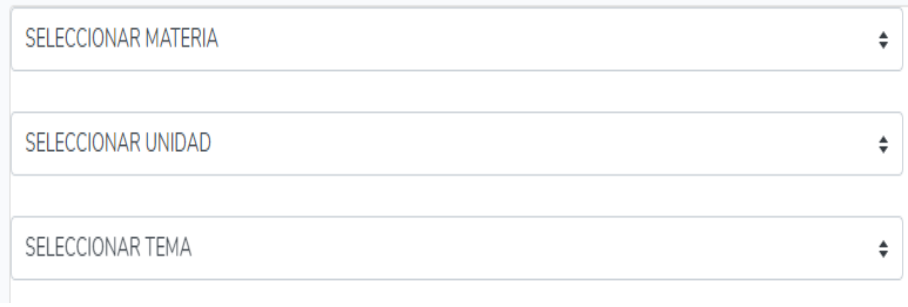


Figura 3.19: Lista dinámica terminada.

Mostrar el contenido didáctico

Aprovechando la estructura lógica de los componentes que almacenan la lista dinámica, se optó por meter dentro del último componente hijo (lista que muestra el tema), una etiqueta de Html que permitiera mostrar documentos pdf.

```
<embed id=mypdf :src=jsvar type="application/pdf" />
```

Por lo tanto, también fue necesario agregar en la base de datos una columna a la tabla de temas, con el fin de guardar los nombres del documento pdf perteneciente a cada tema. Dado a que ya se tenía una función que sacará en nombre del tema de la tabla temas, se tomó y modificó para crear una nueva que recogiera el nombre del pdf.

id_tema	id_unidad	nombre_tema	nombre_pdf
1	1	Tipos de datos	1
2	1	Constantes y variables	2
3	1	Algoritmo	3
4	1	Diagrama de flujo	4
5	1	Pseudocódigo	5
6	2	Estructura secuencial	6
7	2	Estructura de selección	7
8	2	Estructura IF	8
9	2	Estructura IF ELSE	9
10	2	CASE	10
11	3	Contadores	11
12	3	Estructura FOR	12
13	3	Estructura WHILE	13
14	3	Estructura DO UNTIL	14
15	4	Estructura Arreglos unidimencionales	15

Figura 3.20: Tabla tema.

Una vez obtenido el nombre del pdf, fue cuestión de concatenarlo a la dirección (src) donde están almacenados todos los documentos públicos, de esta manera cada vez que se seleccione un tema diferente, el pdf también cambiará.

```
methods:
  {
  muestraPdf(par){
      let ruta='pdf/'.concat(par)
      axios.get(ruta).then((response) => {
      this.jsvar = response.data[0].nombre_pdf.concat(".pdf");

  })
```

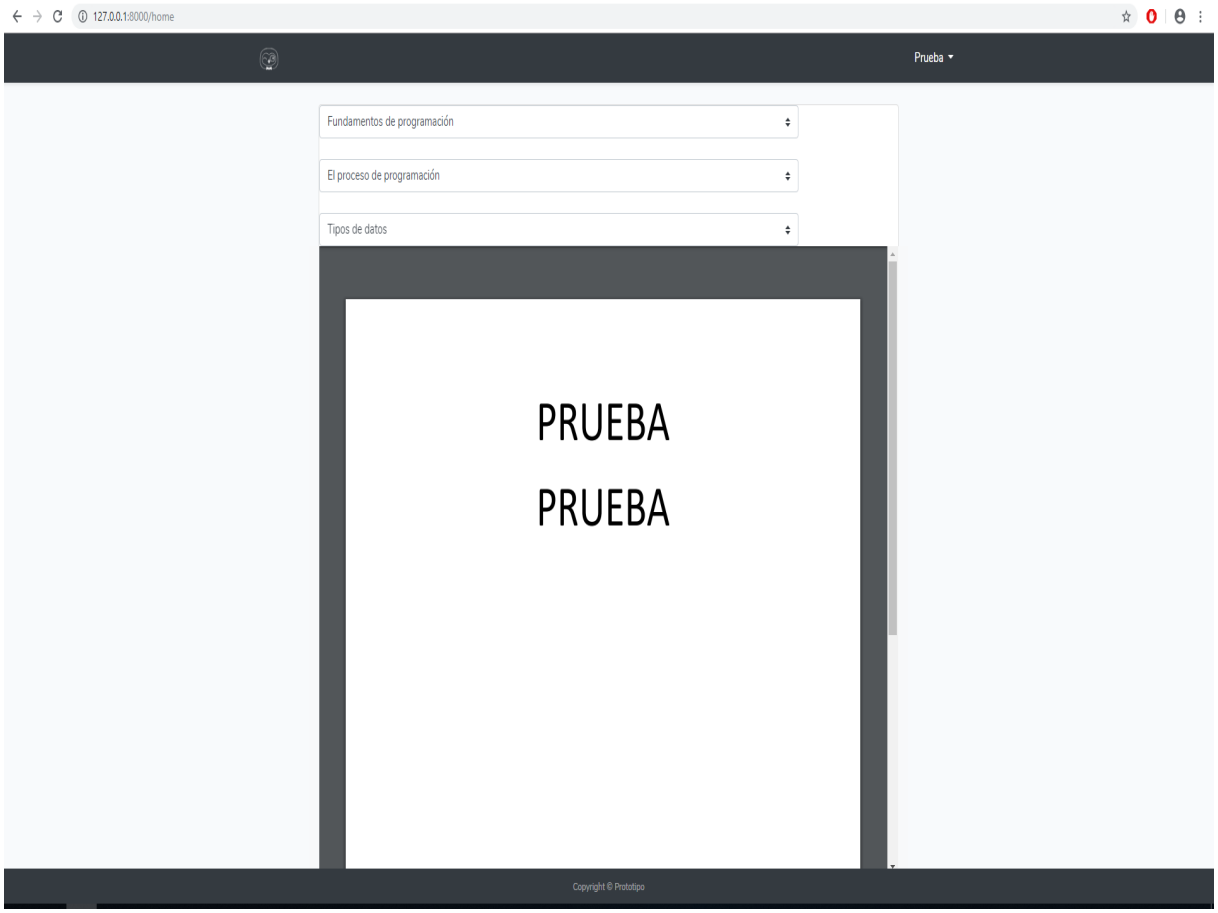


Figura 3.21: Página de trabajo con lista dinámica y pdf.

Contenido didáctico

Se diseñó un plantilla específica para vaciar la información de los temas que previamente fueron seleccionados, el formato de la plantilla seleccionado fue PDF, por sus siglas en inglés (Portable Document Format,«formato de documento portátil»). Cada tema cuenta con una introducción, un problema de ejemplo, un análisis, algoritmo, pseudocódigo y código, (ver Figura 3.22 y 3.23).

El contenido de cada tema esta basado en problemas y ejemplos de libros de Fundamentos de programación de autores reconocidos como Osvaldo Cairó y Luis Joyanes Aguilar.

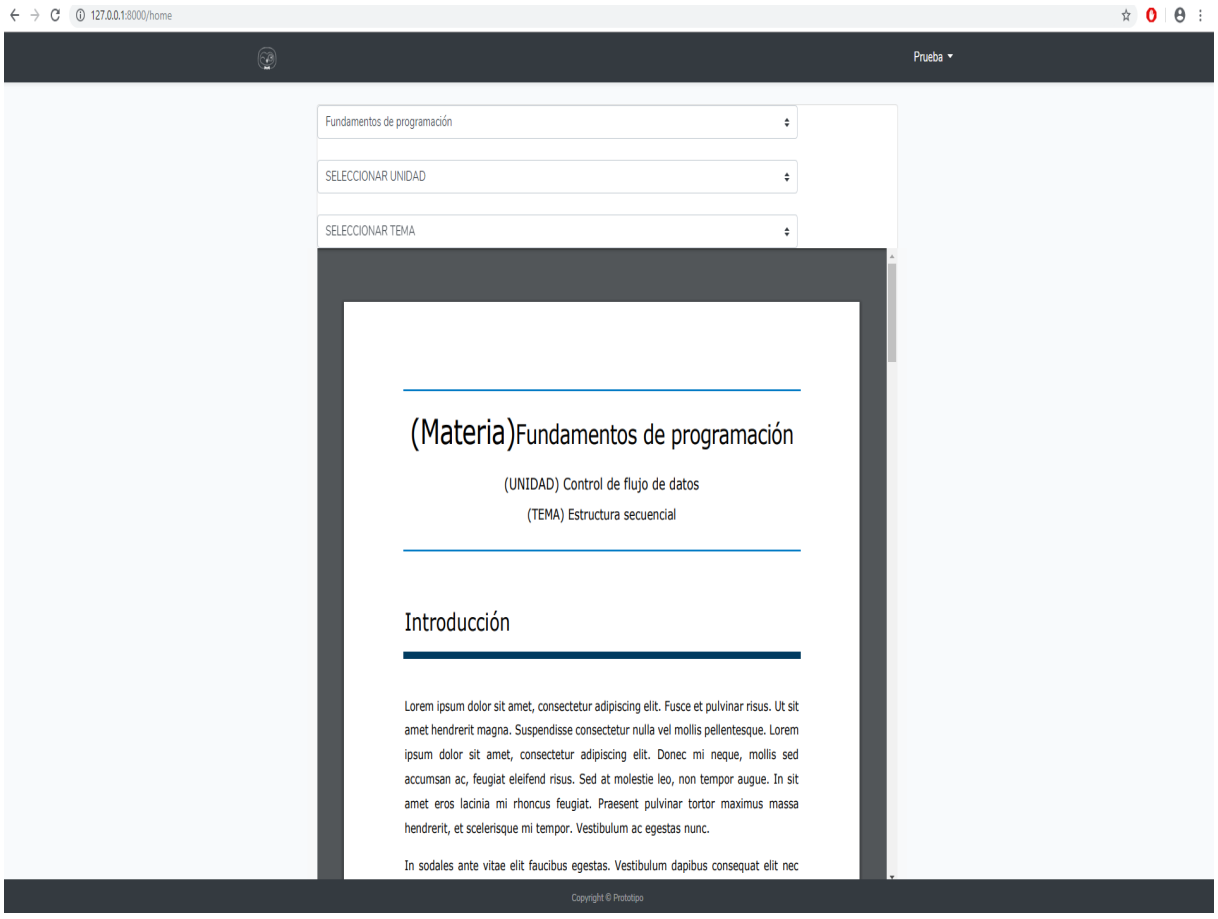
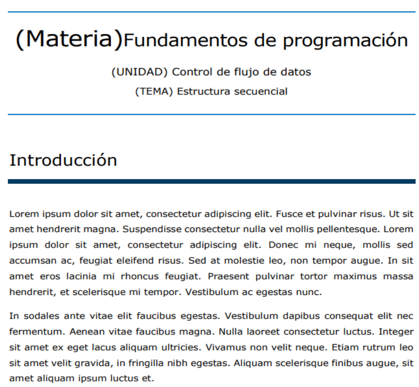


Figura 3.22: Página de trabajo con lista dinámica y pdf.

El encabezado del documento tiene la información relacionada con el tema, es decir, muestra la materia a la que pertenece, la unidad y el tema seleccionado. Después, está la sección de introducción, donde se almacena el concepto y explicación de cada tema, (ver Figura 3.23a). Seguidamente aparece la sección de ejemplo, donde se agregaron problemas de ejemplo, los cuales son serán analizados y desarrollados en secciones posteriores, (ver Figura 3.23b).



(Materia)Fundamentos de programación

(UNIDAD) Control de flujo de datos

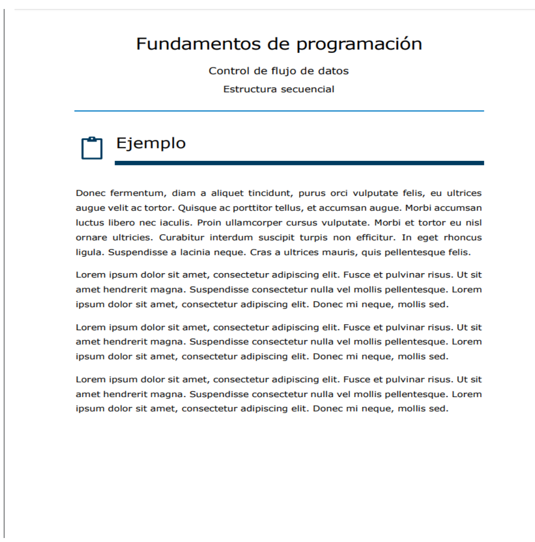
(TEMA) Estructura secuencial

Introducción

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce et pulvinar risus. Ut sit amet hendrerit magna. Suspendisse consectetur nulla vel mollis pellentesque. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec mi neque, mollis sed accumsan ac, feugiat eleifend risus. Sed at molestie leo, non tempor augue. In sit amet eros lacinia mi rhoncus feugiat. Praesent pulvinar tortor maximus massa hendrerit, et scelerisque mi tempor. Vestibulum ac egestas nunc.

In sodales ante vitae elit faucibus egestas. Vestibulum dapibus consequat elit nec fermentum. Aenean vitae faucibus magna. Nulla laoreet consectetur luctus. Integer sit amet ex eget lacus aliquam ultricies. Vivamus non velit neque. Etiam rutrum leo sit amet velit gravida, in fringilla nibh egestas. Aliquam scelerisque finibus augue, sit amet aliquam ipsum luctus et.

(a) Encabezado e introducción.



Fundamentos de programación

Control de flujo de datos

Estructura secuencial

Ejemplo

Donec fermentum, diam a aliquet tincidunt, purus orci vulputate felis, eu ultrices augue velit ac tortor. Quisque ac porttitor tellus, et accumsan augue. Morbi accumsan luctus libero nec iaculis. Proin ullamcorper cursus vulputate. Morbi et tortor eu nisl ornare ultricies. Curabitur interdum suscipit turpis non efficitur. In eget rhoncus ligula. Suspendisse a lacinia neque. Cras a ultrices mauris, quis pellentesque felis.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce et pulvinar risus. Ut sit amet hendrerit magna. Suspendisse consectetur nulla vel mollis pellentesque. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec mi neque, mollis sed.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce et pulvinar risus. Ut sit amet hendrerit magna. Suspendisse consectetur nulla vel mollis pellentesque. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec mi neque, mollis sed.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce et pulvinar risus. Ut sit amet hendrerit magna. Suspendisse consectetur nulla vel mollis pellentesque. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec mi neque, mollis sed.

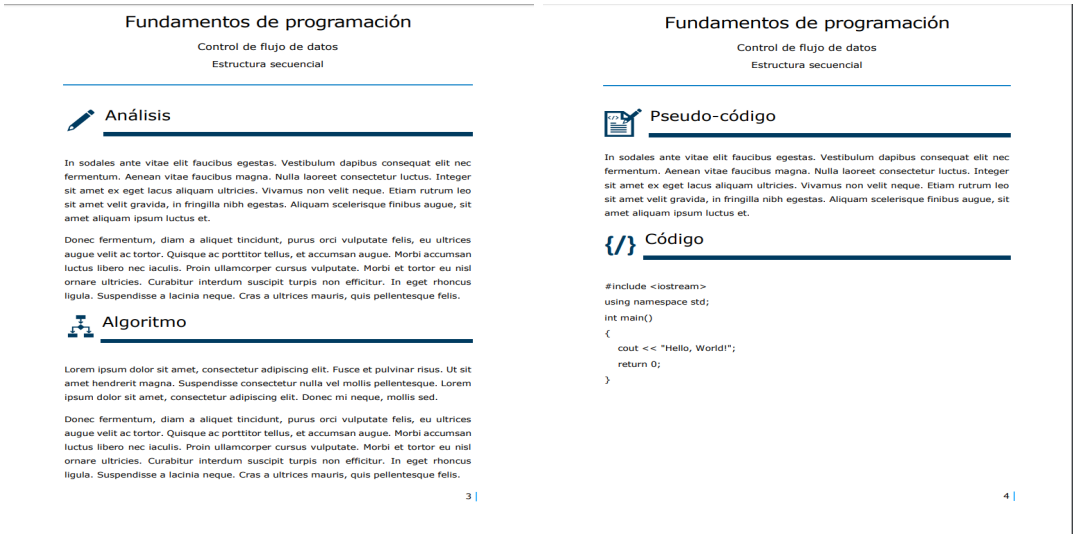
(b) Ejemplo.

Figura 3.23: Contenido didáctico.

Enseguida, se encuentran las secciones de análisis y algoritmo, donde se desglosa el problema de ejemplo y se identifican las variables y constantes que se requieres para el desarrollo de un algoritmo, el cual se encuentra en la sección algoritmo y está representado de manera gráfica, es decir, en diagrama de flujo, (ver Figura 3.24a).

Después que se analizó y desarrolló un algoritmo para el problema de ejemplo, sigue la sección pseudo-código, donde se desarrolla una representación textual que describe el algoritmo previamente realizado. Esta parte es altamente importante debido a que es el último paso antes de pasar a la codificación, (ver Figura 3.24b).

Luego se encuentra el apartado de código, donde muestra el código alcanzado gracias a los pasos de las secciones anteriores. El código se muestra en tres diferentes lenguajes de programación, con el fin de ayudar al usuario a conocer y familiarizarse con estos lenguajes, (ver Figura 3.24b). Finalmente de agregó un apartado para incluir las referencias utilizadas en el desarrollo de cada tema, (para ver el contenido didáctico desarrollado, ir al Apéndice A).



(a) Análisis y algoritmo. (b) pseudo-código y código.

Figura 3.24: Contenido didáctico.

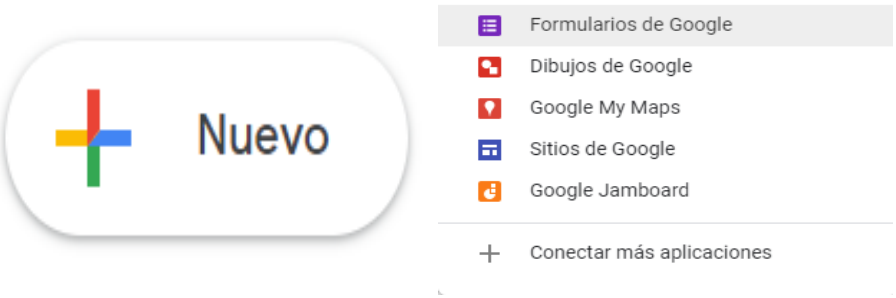
Evaluación

Para la evaluación se utilizó una herramienta que ofrece Google Drive de nombre formularios de Google. Esta herramienta permite realizar exámenes y cuestionarios con la ventaja de crear preguntas de diferentes formas, es decir, se pueden aplicar preguntar de tipo selección múltiple, con casillas de verificación, desplegable, de respuesta corta, entre otros tipos.

Los formularios de Google ofrecen la opción de crear auto evaluaciones, es decir que se puede definir la respuesta correcta de cada pregunta y al final de cada evaluación se realizará

una revisión de manera automática. Además, se puede agregar un puntaje a cada pregunta. Estas evaluaciones son acceso público, opción que fue de gran ventaja para este proyecto.

Para realizar cada evaluación fue necesario crear y entrar a una cuenta de Gmail, una vez hecho esto, se debe abrir el botón "Nuevo", (ver, Figura 3.25a). Luego seleccionar la opción "Formularios de Google", (ver, Figura 3.25b).



(a) botón nuevo (b) formulario de Google.

Figura 3.25: Crear formulario de Google.

Los pasos anteriores generaron una evaluación vacía, donde se pueden apreciar una barra de herramientas para la personalización de cada pregunta (ver Figura 3.26).

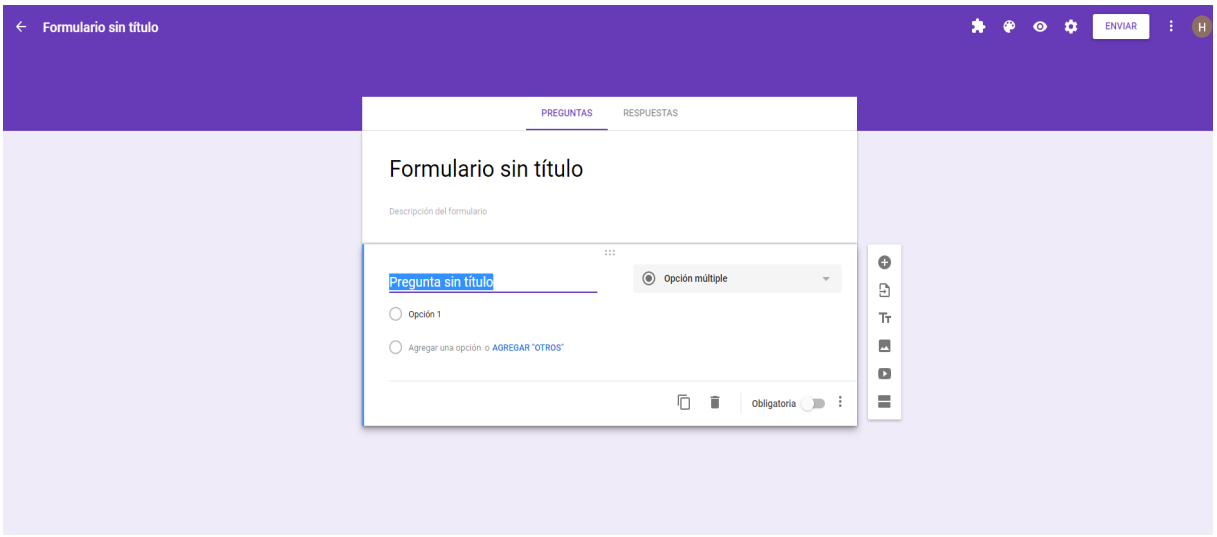


Figura 3.26: Formulario de Google vacío.

Antes de empezar a llenar la evaluación, fue necesario activar la opción de autoevaluación, la cual permite asignar un valor (puntos) a cada pregunta, (ver Figura 3.27). El puntaje asignado a cada pregunta varía según su dificultad y va desde los 10 a 30 puntos.

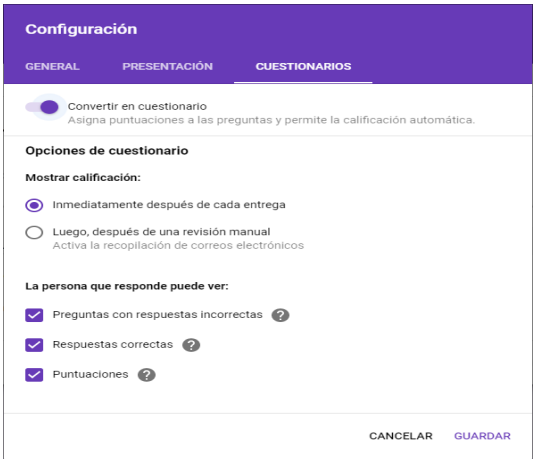


Figura 3.27: Configuración del formulario.

Después, se empezó a llenar con el contenido respectivo cada evaluación, (ver Figura 3.28). Para ver el contenido de evaluación desarrollado para cada tema ver el Apéndice B.

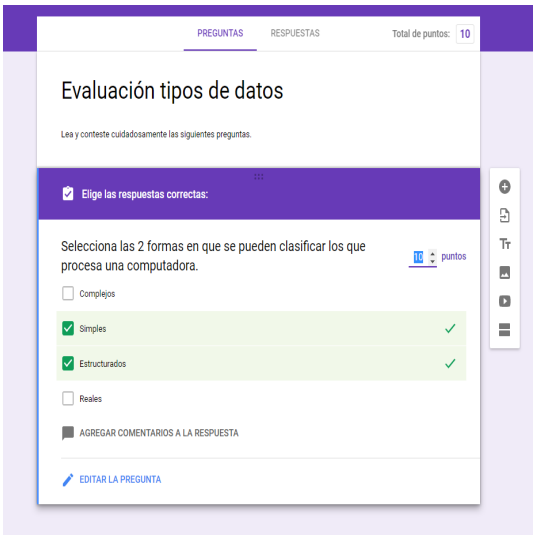


Figura 3.28: Ejemplo de una pregunta con autoevaluación y puntaje.

Enseguida, se creó un botón en la aplicación para acceder a la evaluación, (ver Figura

3.29).

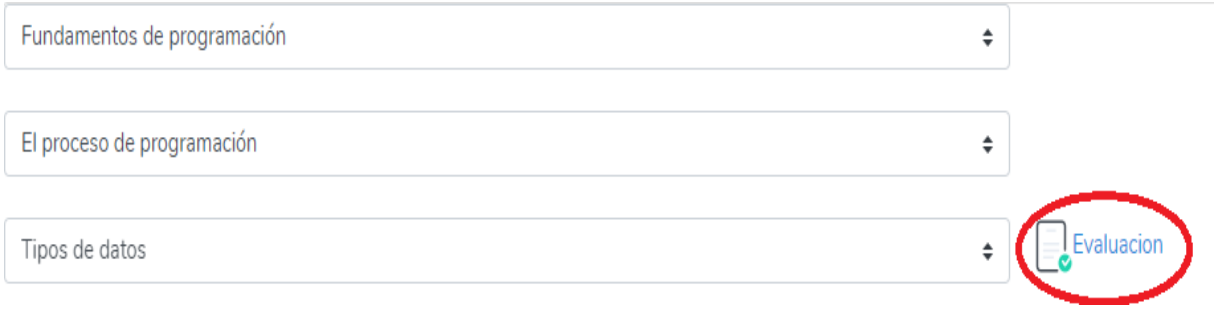


Figura 3.29: Botón de evaluación.

Luego, para mostrar la evaluación que corresponde a cada tema, se realizó una función que es llamada por el botón anterior que almacena una estructura selectiva switch, donde se evalúa e identifica el tema seleccionado para dar la instrucción de mostrar una evaluación en específico, (ver Figura 3.30).

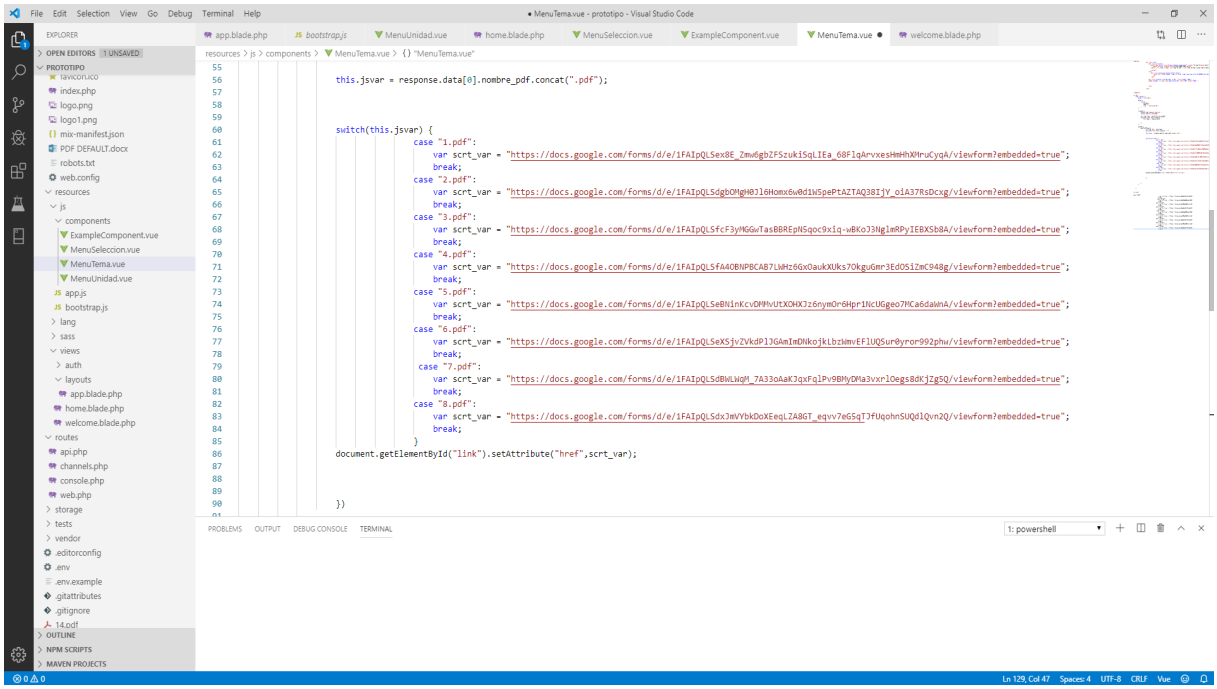


Figura 3.30: Estructura switch con las evaluaciones.

Finalmente, al seleccionar un tema y hacer clic al botón evaluación, se abrirá una nueva pestaña con la evaluación correspondiente al tema, (ver Figuras 3.31 y 3.32).

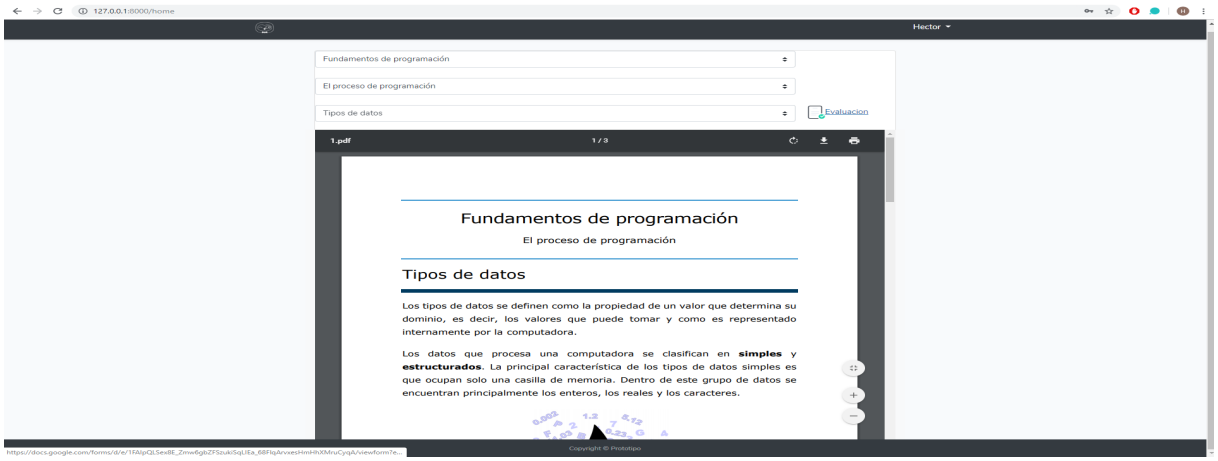


Figura 3.31: Funcionalidad del botón evaluación.

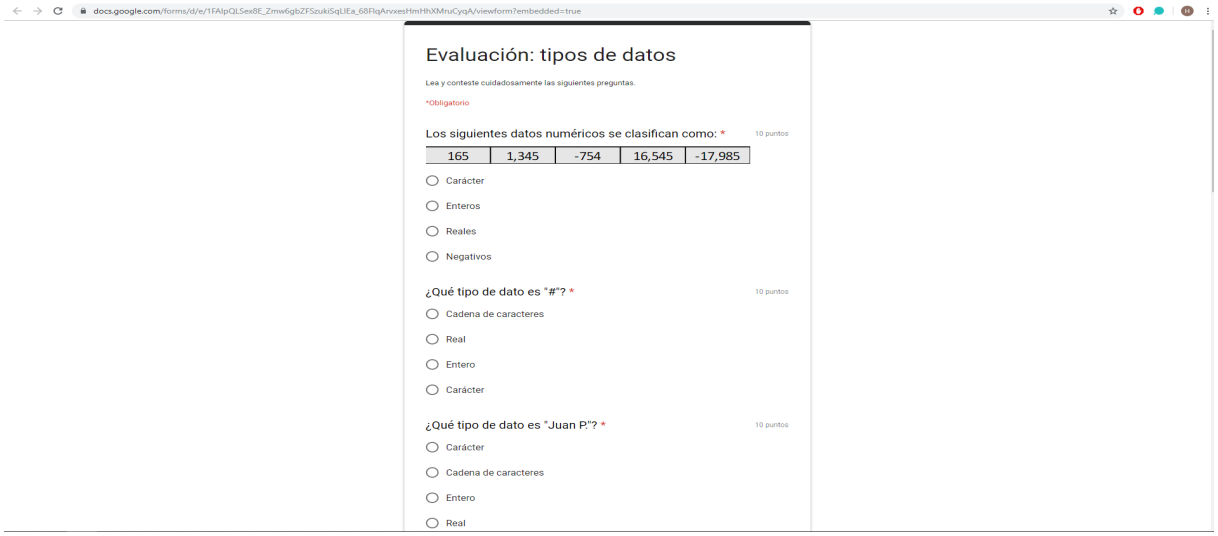


Figura 3.32: Evaluación.

Capítulo 4
Resultados y Discusiones

El presente capítulo muestra los resultados producto de la investigación, acerca del funcionamiento y efectividad del proyecto con el usuario. Posteriormente se comentarán las conclusiones obtenidas, donde se abordan los puntos fuertes y débiles que presenta el prototipo de sistema web según los resultados. Al final, se mostrará las respuestas obtenidas en forma de gráficas para su análisis.

4.1. Presentación de resultados

Con la intención de analizar, medir y comparar los resultados acerca del funcionamiento que tiene el proyecto, se preparó como instrumento de evaluación, una encuesta. La ubicación seleccionada para realizar las encuestas fue el Instituto de Ingeniería y Tecnología de la UACJ, con el objetivo de aplicar la encuesta a estudiantes de distintas carreras.

La encuesta se aplicó a 32 estudiantes de diversas carreras, cada encuesta fue compuesta por dos secciones, la primera sección abarca todo el contenido relacionado con el diseño del entorno web, como lo son los colores, el tamaño de letra, entre otros. La segunda parte engloba el apartado de la evaluación del contenido que ofrece el prototipo, con el propósito de evaluar la calidad de cada apartado que ofrece cada tema, (ver Apéndice C).

4.1.1. Resultados de los cuestionarios

Como datos preliminares a la encuesta, se solicitó a cada estudiante especificar el programa académico al que pertenecen, con fin de identificar y clasificar los datos obtenidos. En la Figura 4.1, se puede observar que del total estudiantes que aceptaron realizar la encuesta, los estudiantes de la carrera que mostró más interés fue Ingeniería en sistemas computacionales.

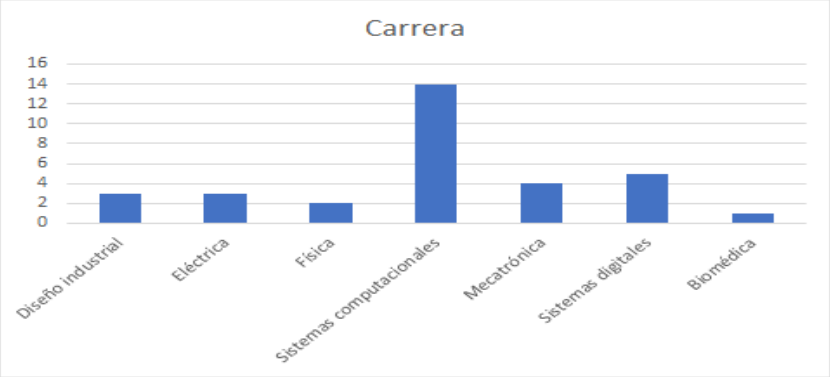


Figura 4.1: Programas académicos.

De todos los estudiantes encuestados, se puede apreciar en la Figura 4.2, que de los treinta y dos estudiantes, solo la tercera parte corresponde a estudiantes mujeres.

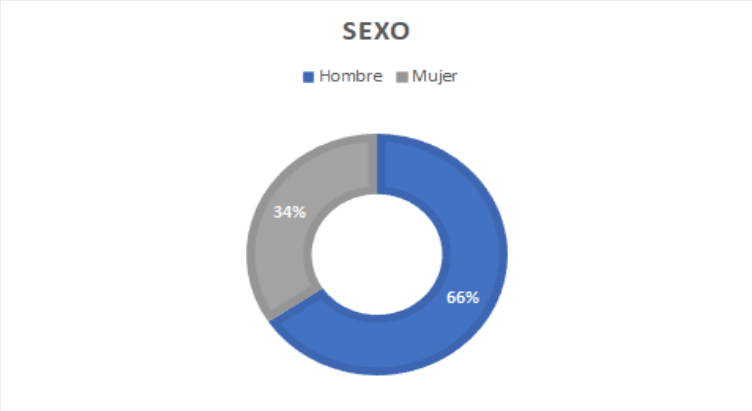


Figura 4.2: Gráfica de género.

En la Figura 4.3, se puede observar que todas las personas están de acuerdo con el diseño de la página de bienvenida, (estructura, usabilidad, entre otros).

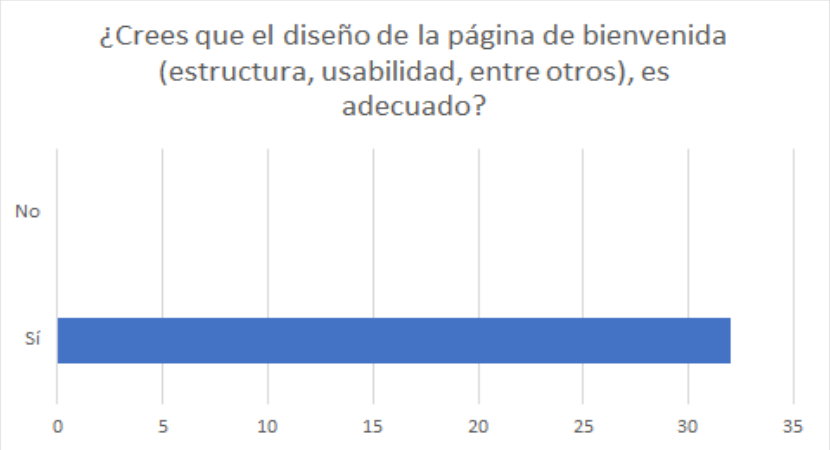


Figura 4.3: Pregunta uno, acerca del diseño de la página de bienvenida.

En la siguiente gráfica, se puede ver que la mayoría de los estudiantes piensan que la forma de presentar el contenido es adecuado, no obstante, una pequeña parte de los estudiantes no están de acuerdo, esta opinión es compartida por hombres y mujeres, (ver Figura 4.4).

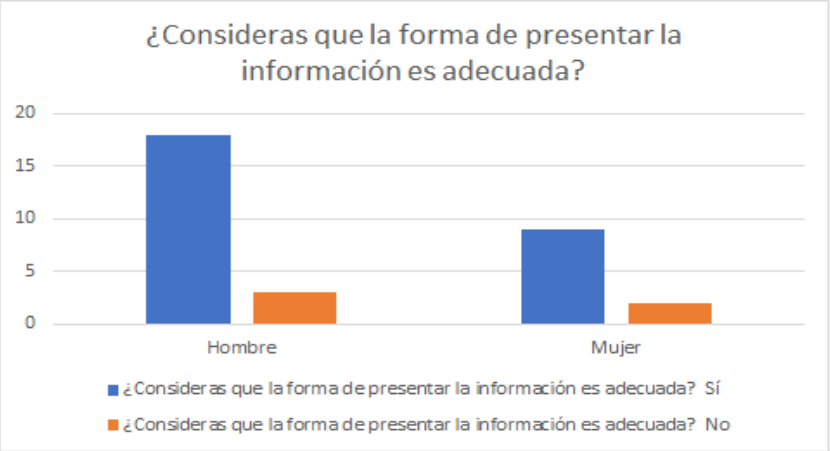


Figura 4.4: Pregunta dos, forma adecuada de presentar el contenido.

En la pregunta tres, treinta y un estudiantes indicaron que el diseño de la página de interfaz de usuario, les pareció adecuado. Por otra parte solo un estudiante indicó estar en desacuerdo, (ver Figura 4.5).

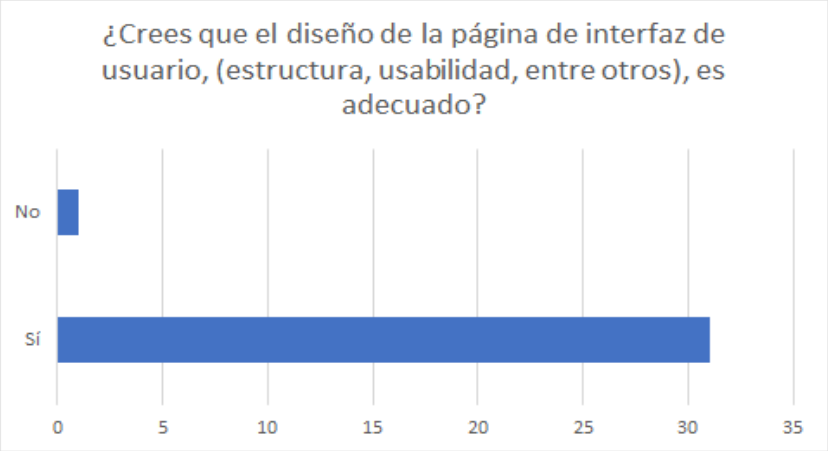


Figura 4.5: Pregunta tres, diseño de la interfaz de usuario.

La pregunta cuatro es con respecto a la consideración sobre exceso de elementos en pantalla, (ver Figura 4.6), se puede observar que todos los estudiantes concuerdan con que no hay un exceso de estos.

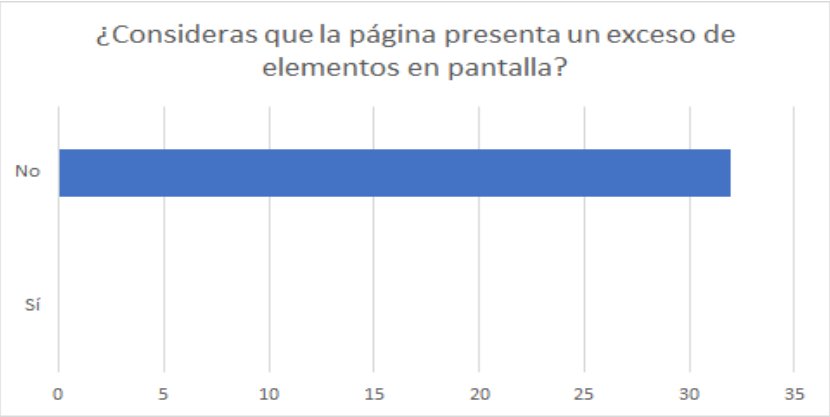


Figura 4.6: Pregunta cuatro, exceso de información.

Luego, en la pregunta cinco se puede apreciar que se obtuvieron respuestas diversas, donde más de la mitad de los estudiantes consideran que se tiene una redacción buena-regular, mostrando así, que se recomienda mejorar más el contenido redactado, (ver Figura 4.7).

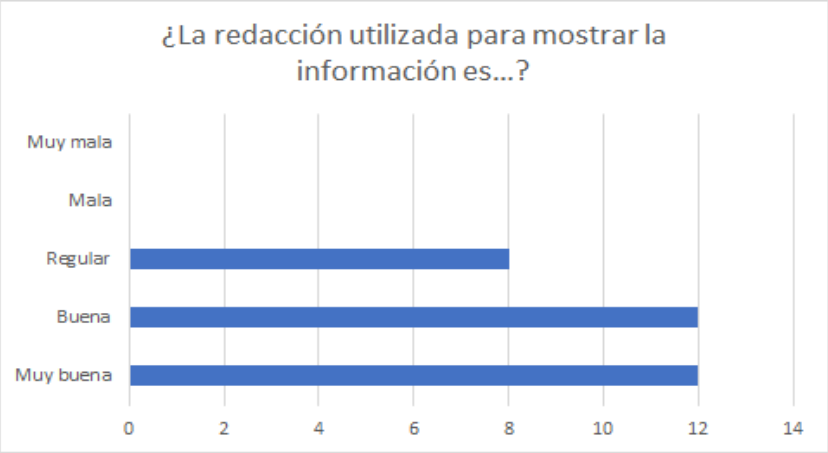


Figura 4.7: Pregunta cinco, sobre la redacción.

La pregunta seis arrojó resultados que demuestran que, los estudiantes no tienen problema con el tamaño y tipo de letra utilizado en el contenido, (ver Figura 4.8).

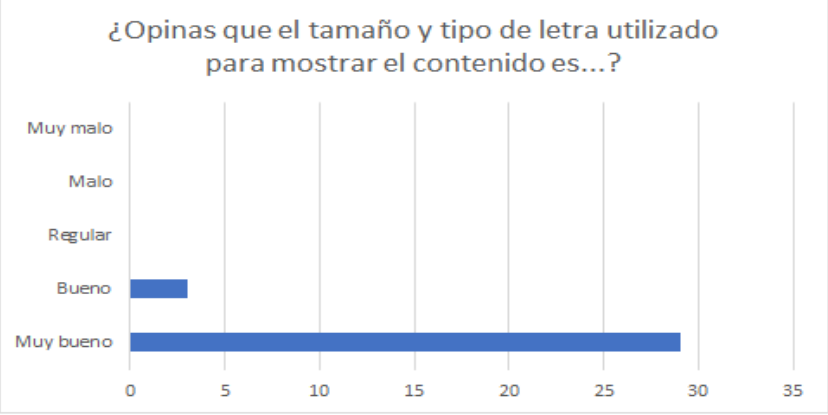


Figura 4.8: Pregunta seis, tamaño y tipo de letra.

Después, en la pregunta número siete, se obtuvieron resultados positivos que indican la aceptación en la facilidad de uso del proyecto, (ver Figura 4.9).

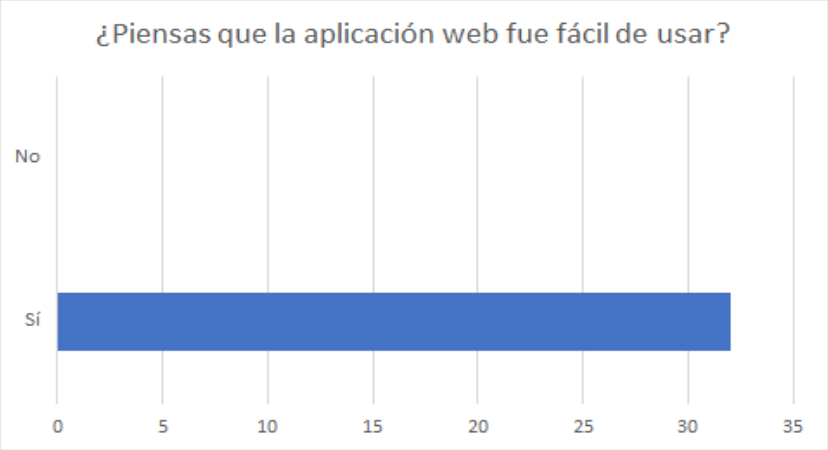


Figura 4.9: Pregunta siete, aplicación fácil de usar.

En la pregunta ocho, se obtuvieron respuestas muy interesantes, pues más de la mitad de estudiantes consideró que el inicio de sesión obligatorio para acceder al proyecto, puede parecer complicado o innecesario, (ver Figura 4.10).

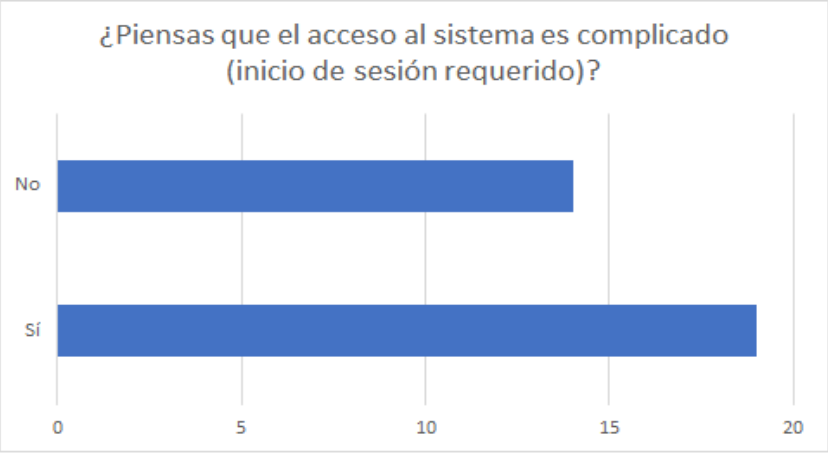


Figura 4.10: Pregunta ocho, aplicación fácil de usar.

En la pregunta número nueve, es posible apreciar que todos los estudiantes encuestados concuerdan que la programación es importante, (ver Figura 4.11).

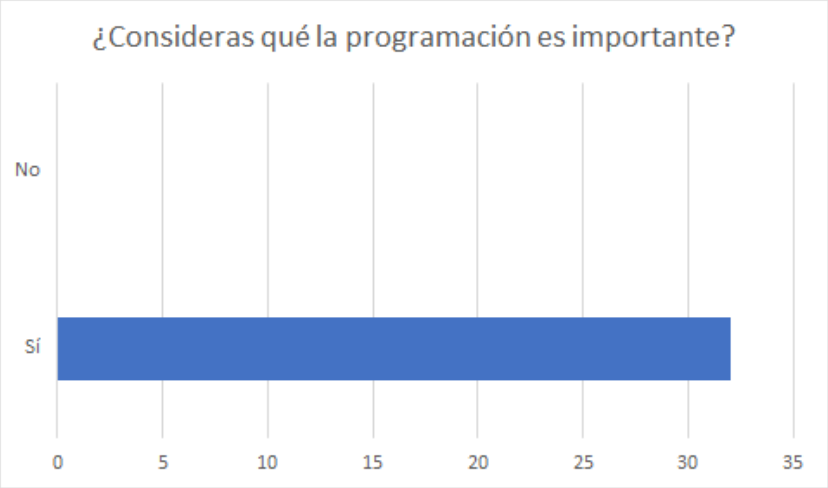


Figura 4.11: Pregunta nueve, acerca de la programación.

En la siguiente gráfica se indica que a todos los estudiantes encuestados les pareció interesante el objetivo de este proyecto, (ver Figura 4.12).

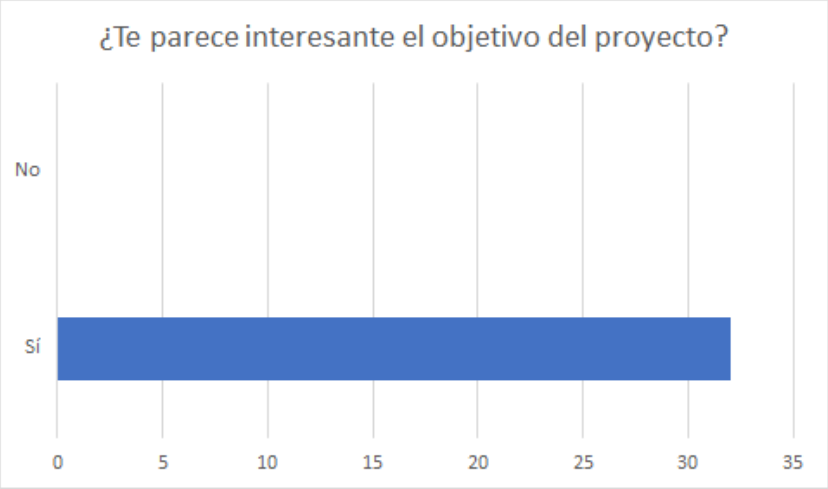


Figura 4.12: Pregunta diez, objetivo del proyecto.

En la pregunta número once, se muestra que de los 32 estudiantes encuestados, 4 indican que el contenido les pareció difícil de comprender, por lo tanto, se puede decir que se recomienda revisar el nivel de complejidad del contenido, (ver Figura 4.13).

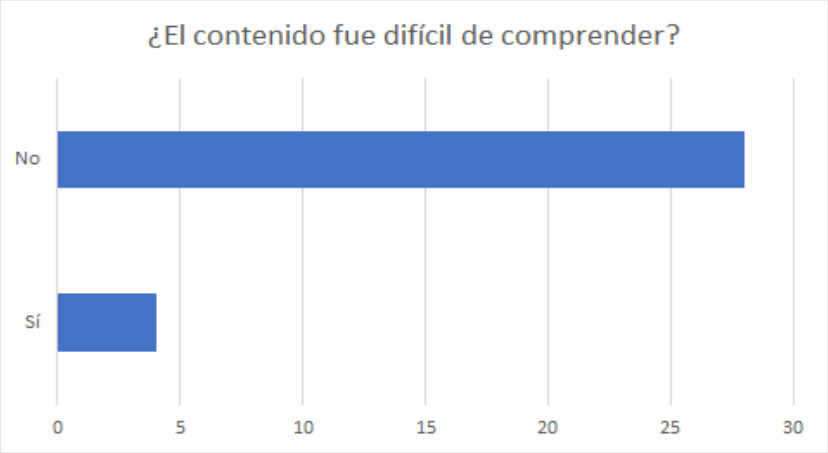


Figura 4.13: Pregunta once, complejidad del contenido.

En la siguiente gráfica se indica que a la gran mayoría les pareció que el contenido del proyecto es bueno o muy bueno, solo una persona indicó que el contenido le pareció regular, (ver Figura 4.14).

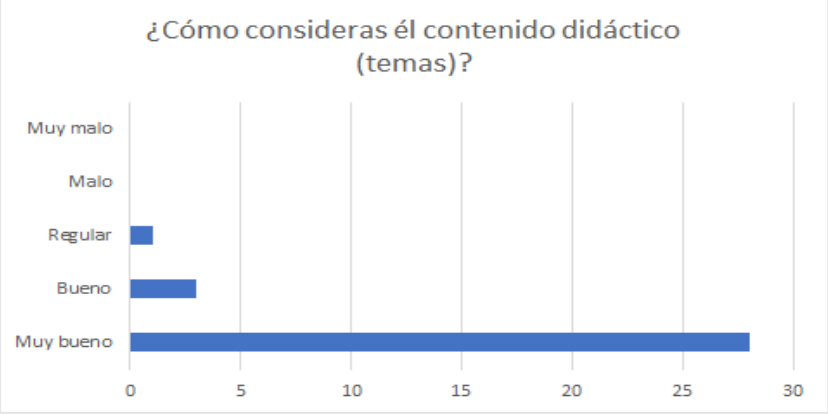


Figura 4.14: Pregunta doce, evaluación del contenido.

En la Figura 4.15, se muestra la frecuencia indicada por los estudiantes después de probar el prototipo, donde la gran mayoría (26/32 estudiantes) seleccionó *siempre*, luego *regular* por tres estudiantes, *aveces* por dos estudiantes y por último una persona indicó que no lo usaría *nunca*.

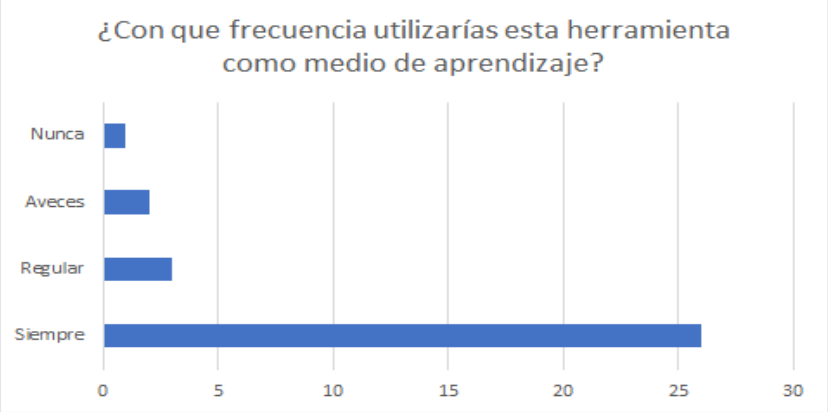


Figura 4.15: Pregunta trece, frecuencia de uso

En la pregunta catorce, se muestra que 31 estudiantes señalaron que recomendarían esta herramienta a sus amigos, solo un estudiante contestó de manera negativa, (ver Figura 4.16).

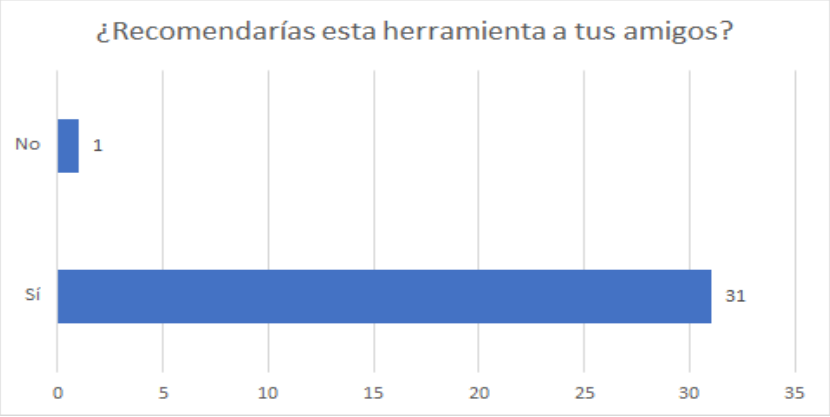


Figura 4.16: Pregunta catorce, recomendar el proyecto.

Por último, en la pregunta quince, se aprecia que la mayoría de los encuestados (24 estudiantes) indicaron que les gustaría se implementará problemas al contenido, solo ocho estudiantes indicaron que no es necesario, (ver Figura 4.17).

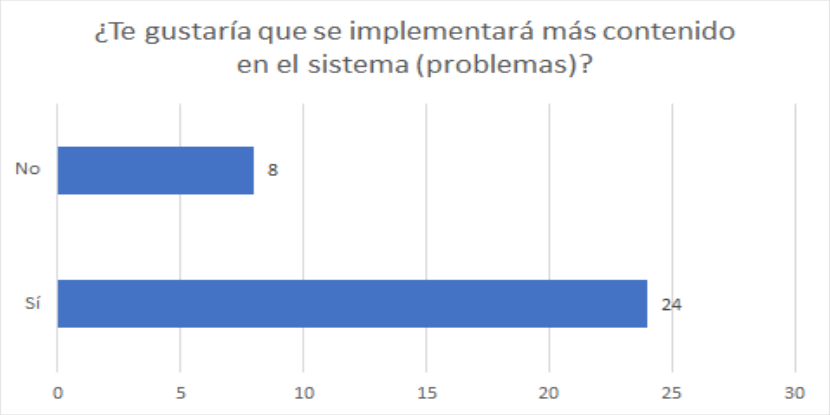


Figura 4.17: Pregunta quince, implementación de más contenido.

Capítulo 5
Conclusiones

Para finalizar, en este capítulo se abordarán las conclusiones adquiridas durante el desarrollo del proyecto, con el fin de analizar si el objetivo se logró y así mismo darle continuidad al proyecto con recomendaciones para futuras investigaciones.

5.1. Con respecto a las preguntas de investigación

1) ¿De qué manera se va a realizar, mostrar y evaluar el contenido del proyecto?

Antes de comenzar a desarrollar el contenido, se definió una estructura estándar para todos los temas, con el objetivo de mantener una secuencia lógica a largo de cada uno de los mismos. La estructura fue diseñada con base a la metodología de resolución de problemas de George Polya, con la diferencia de que esta metodología se modificó para enfocarla a la resolución de problemas por computadora mediante la programación.

Por lo tanto, cada tema comienza con una explicación clara y concisa del tema, luego se desarrolló un problema de ejemplo, el cual fue divido en cuatro etapas las cuales fueron: análisis del problema, desarrollo gráfico de un algoritmo, pseudocódigo y código. La finalidad por el cual se seleccionó esta estructura fue la de facilitar el entendimiento al usuario sobre los pasos fundamentales de resolver un problema por computadora mediante la programación.

El formato seleccionado para mostrar el documento en pantalla fue "formato de documento portátil", más conocido como PDF. Por otra parte, el contenido de la evaluación fue generado en su totalidad a partir del tema correspondiente elaborado y se utilizó la herramienta Google Forms para desarrollarla.

2) ¿Cómo desarrollar una herramienta web que haga gestión del contenido desarrollado?

Una vez desarrollado todo el contenido con base a la carta la descriptiva (formato modelo educativo UACJ Visión 2020), de la asignatura Fundamentos de programación, se diseñó y desarrolló un proyecto web que permitiera gestionar todo el contenido en su formato (PDF).

Por consiguiente, fue necesario utilizar diversas herramientas enfocadas al desarrollo web, donde una de las más importantes fue el uso de componentes con la herramienta Vue.js, la cual permitió gestionar y mostrar el contenido respectivo a los parámetros unidad y tema.

5.2. Con respecto al objetivo de la investigación

Al terminar este proyecto, es posible concluir que el objetivo de la investigación realizada fue cumplido con éxito, gracias a que los resultados indican que obtuvo la aprobación de la gran mayoría del público a la que se le presentó.

En consecuencia, se puede decir que el desarrollo de este proyecto cuenta con una buena probabilidad de ayudar a reforzar y aprender los conocimientos sobre los fundamentos de programación, así como llegar influir en la mejora de las notas de los estudiantes y disminuir el índice de reprobados de esta asignatura.

5.3. Recomendaciones para futuras investigaciones

Durante el desarrollo de este proyecto existieron muchas buenas ideas sobre funcionalidades a implementar que por motivo del tiempo no pudieron ser completadas o incluidas en el proyecto final, a continuación se muestran algunas recomendaciones para todas aquellas personas interesadas en realizar mejoras a proyecto.

Se recomienda desarrollar un banco de preguntas más grande para cada una de las evaluaciones. Con esto se pretende seleccionar de manera aleatoria problemas y preguntas cada que se genere una evaluación.

Desarrollar un sistema de gestión de contenido, es decir, permitir a un usuario con rol de administrador subir, eliminar o modificar el contenido del prototipo de aplicación. Esta funcionalidad daría paso a crear un proyecto escalable, que permita agregar más materias y así mismo desarrollar su respectivo contenido con el fin de ayudar a otras personas.

Se recomienda crear un sistema de progreso por usuario, es decir, el usuario pueda ver su nivel de avance en cada una de las unidades evaluadas, incluso desarrollar un sistema de niveles o puntuación con el fin de alentar al usuario a concluir el contenido de la materia.

Posteriormente, contar con otras formas de accesibilidad sería muy útil para aquellas personas que tengan ciertas dificultades para acceder al contenido, alguna de las recomendaciones sería desarrollar el contenido de manera auditiva, para facilitar el uso de este prototipo de aplicación.

Finalmente, siempre está la posibilidad de desarrollar o adaptar este proyecto a una aplicación móvil, con el objetivo de tener acceso al proyecto desde otros dispositivos. Esto debido a que hoy en día es más probable que las personas cuenten con un teléfono móvil y no con una computadora.

Bibliografía

[1] J. Iruela, "La importancia de la programación en la formación", mar 2015.

[2] universia.edu.ve, "Programación, el lenguaje del futuro", jun 2015.

[3] M. Á. Abellán, "programoergosum", mar 2019.

[4] code.org, "code", jun 2013.

[5] J. Atwood, "programmr.com", mar 2011.

[6] R. Bernárdez, "sololearn", mar 2014.

[7] D. Kolb, "sololearn", oct 2011.

[8] V. Spraul, "solveet", jan 2012.

[9] C. Tinoco T. Anchondo, "objetos de aprendizaje que apoyen el proceso de la enseñanza-aprendizaje de la programación de computadoras a nivel medio superior y superior", may 2009.

[10] E. Valdivia J. Gaytán, "Juego serio para desarrollo de habilidades cognitivas de programación de estructuras de datos.", nov 2015.

[11] L. Valles, "Runaround: Aplicación orientada a la enseñanza y aprendizaje de la programación estructurada a través de animaciones generadas a partir de código fuente.", may 2013.

[12] A. Gonzales, "La importancia en la vida diaria de aprender a programar", oct 2017.

[13] UACJ, "Anuario estadístico 2017-2018", jan 2017.

[14] J. Hornsby C.Miller, V.Heeren, *Matemática Razonamiento y aplicaciones*, Pearson educación de México, S.A. de C.V., Naucalpan de Juárez, Edo.México, 2013.

[15] I. de Jesús M. Cen, "Cómo plantear y resolver problemas [título original: How to solve]", *Entreciencias: diálogos en la Sociedad del Conocimiento*, vol. 3, pp. 419–420, 2015.

[16] G. Polya, *Cómo plantear y resolver problemas*, Editorial Trillas S.A de C.V., Mexíco D.F., 1965.

[17] L. Aguilar, *FUNDAMENTOS DE PROGRAMACIÓN: Libro de problemas*, mcgraw hill /interamericana de España S. A. U., second edition, 2003.

[18] C. Velázquez F.Pinales, "Problemario de algoritmos resueltos con diagramas de flujo y pseudocÓdigo", jun 2014.

[19] O. Cairó, *Fundamentos de programación. Piensa en C*, Pearson Educación de México, S.A de C.V., 2006.

[20] A. Becerra, *Introducción a la programación*, Ignacio Murgueitio, Santiago de Cali, 2009.

[21] J.Sanchez, "https://openlibra.com", 2009.

[22] M. Roldán V. Benjumea, "Fundamentos de programación con el lenguaje de programación c++", mar 2016.

[23] J. Vázquez Gómez, *Análisis y diseño de algoritmos*, RED TERCER MILENIO S.C, Tlalnepantla, 2012.

[24] David A. Wiley, "Learning object design and sequencing theory", jun 2002.

[25] L. Guzmán, "Los repositorios de objetos de aprendizaje como soporte para los entornos e- learning", sep 2005.

[26] O. Santamaría H. Montero, *Informe APEI sobre Usabilidad*, Asociación Profesional de Especialistas en Información, 2009.

[27] J. Dimuro, "Guía de usabilidad web", may 2014.

[28] S. Krug, *No me hagas pensar: Una aproximación a la usabilidad en la Web*, pearson prentice hall, second edition, 2006.

[29] Alejandro Floría Cortés, *Métodos de Indagación*, Centro Politécnico Superior | Universidad de Zaragoza., feb 2000.

[30] J. Sierra, *Métodos de Evaluación de Usabilidad para Sistemas de Información Web: Una revisión*, Universidad Nacional de Colombia, jun 2014.

[31] O. Montoto, "Accesibilidad web y seo", jun 2012.

[32] W3C, "Introducción a la accesibilidad web", aug 2005.

[33] W3C, "Understanding wcag 2.0 a guide to understanding and implementing web content accessibility guidelines 2.0", oct 2016.

Apéndice A
Contenido didáctico

Fundamentos de programación

El proceso de programación

Tipos de datos

Los tipos de datos se definen como la propiedad de un valor que determina su dominio, es decir, los valores que puede tomar y como es representado internamente por la computadora.

Los datos que procesa una computadora se clasifican en **simples** y **estructurados**. La principal característica de los tipos de datos simples es que ocupan solo una casilla de memoria. Dentro de este grupo de datos se encuentran principalmente los enteros, los reales y los caracteres.

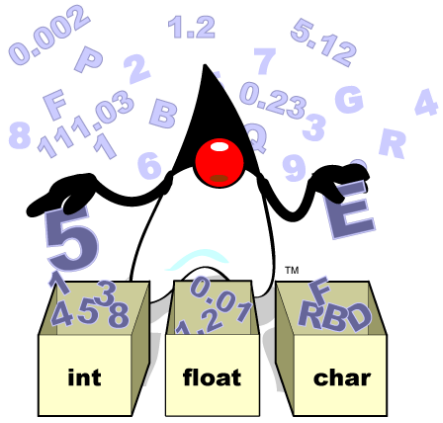


Ilustración 1, tipos de datos

Por otra parte, los datos estructurados se caracterizan por el hecho de que con un nombre se hace referencia a un grupo de casillas de memoria. Es decir, un dato estructurado tiene varios componentes. Los arreglos, cadena de caracteres y registros representan los datos estructurados más conocidos.

Fundamentos de programación

El proceso de programación

Ejemplo

Tabla 1, ejemplo de tipos de datos numéricos

Datos numéricos					
Enteros	165	1,345	-754	16,545	-17,985
Reales	8.5	165.2	-46.821	164.8	-16.3

Tabla 2, ejemplo de tipos de datos alfanuméricos

Datos alfanuméricos					
Carácter simple	a	F	@	&	^
Cadena de caracteres	"abc"	"Juan"	"165"	"Juan P."	"@#^"

Tabla 3, tipos de datos en lenguaje C

Tipos de datos en C	Descripción	Rango
int	Entero	-32,758 A +32,767
float	Reales	$3.4X10^{-38}$ A $3.4X10^{38}$
long	Enteros de largo alcance	-2,147,483,648 A 2,147,483,467
double	Reales de doble precisión	$1.7X10^{-308}$ A $1.7X10^{308}$
char	Caracter	Símbolos del abecedario, números o símbolos espaciales

Referencias

[1] O. Cairó, Metología de la programación, ciudad de México: Alfaomega Grupo editor S.A de C.V., 2005.

[2] L. J. Aguilar, Fundamentos de programación, Algoritmos, estructura de datos y objetos, México: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A .U., 2008.

[3] M. R. Vicente Benjumea, «Fundamentos de Programación con el Lenguaje de Programación C++,» 11 Marzo 2016. [En línea]. Available: https://openlibra.com/es/book/download/fundamentos-de-programacion-con-el-lenguaje-de-programacion-c. [Último acceso: 14 Junio 2019].

Fundamentos de programación

El proceso de programación

Constantes y variables

En programación, una ***constante*** es un dato cuyo valor siempre es el mismo, es decir su valor no puede cambiar mientras se ejecuta el programa.

Por otra parte, las variables en programación son consideradas como una de las propiedades más potentes. Una ***variable*** se puede definir como un contenedor que se utiliza para almacenar datos en un programa y su valor puede cambiar durante el desarrollo o ejecución del programa.

Ilustración 1, constantes y variables.

Ejemplo

En la tabla 1, se muestra la declaración de constantes en distintos lenguajes de programación:

Tabla 1, Ejemplos de cómo declarar una constante en los lenguaje C, Java y PHP.

Lenguaje	Declaración de constantes

C	`const <tipo_de_dato> <nombre_constante> = <valor>;`
	`#define <nombre_constante> = <valor>;`
	`const int PI = 3.1416;`
	`#define PI = 3.1416;`

Java	`static final <tipo_de_dato> <nombre_constante> = <valor>;` `static final float PI = 3.1416;`
PHP	`define("nombre_constante", "valor")` `define("PI", "3.1416")`

De la misma manera, se muestra la sintaxis para declarar variables se muestra en la tabla 2:

Tabla 2, Ejemplos de cómo declarar una variable en los lenguajes C, Java y PHP.

Lenguaje	Declaración de variables

C	`<tipo_de_dato> <nombre_variable>`
	`int x;`
	`int y = 0;`

Java	`<tipo_de_dato> <nombre_variable>`
	`int x;`
	`int y = 0;`

PHP	`$nombre_variable = valor;`
	`$x = 25;`
	`$var = 'ejemplo';`

Referencias

[1] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

[2] A. B. Sandoval, Introducción a la programación, Santiago de Cali: Ignacio Murgueitio, 2009.

[3] M. R. Vicente Benjumea, «Fundamentos de Programación con el Lenguaje de Programación C++,» 11 Marzo 2016. [En línea]. Available: https://openlibra.com/es/book/download/fundamentos-de-programacion-con-el-lenguaje-de-programacion-c. [Último acceso: 14 Junio 2019].

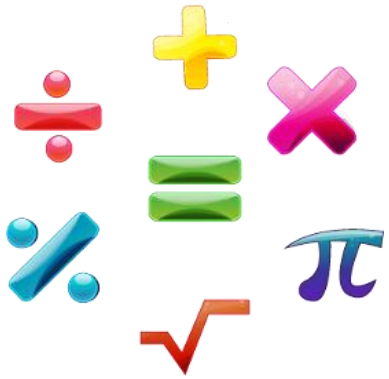
Fundamentos de programación

El proceso de programación

Operadores y expreciones

Los **operadores** son símbolos que expresan operaciones que se aplican a los operandos.

Las **expresiones** son operaciones que dan como resultado un valor a excepción de expresiones *void*.



Operadores

Los distintos tipos de operadores son: aritméticos, relacionales, lógicos, aritméticos simplificados, operadores de incremento y decremento, y el operador coma.

Fundamentos de programación

El proceso de programación

Operadores aritméticos

Los operadores aritméticos nos permiten realizar operaciones entre operandos: números, constantes o variables. El resultado de una operación aritmética siempre es un número.

En la siguiente tabla se muestran los operadores aritméticos con varios ejemplos de su uso y el resultado correspondiente a cada caso. Considere que "x" es una variable de tipo entero (int) y la variable "v" como tipo real (float).

Operador aritmético	Operación	Ejemplos	Resultados

+	Suma	x = 4.5 + 3;
		
		v = 4.5+3;
			x = 7
			
			v = 7.5

-	Resta	x = 4.5 - 3;
		
		v = 4.5 - 3;
			x = 1
			
			v =1.5

*****	Multiplicación	x = 4.5 * 3;
		
		v = 4.5 * 3;
			x = 13
			
			v = 13.5

/	División	x = 4 / 3;
		
		v = 4 / 3;
			x = 1
			
			v = 1.0

%	Módulo(residuo)	x = 15 % 2;
		
		v = (15 % 2) / 2;
			x = 1
			
			v = 0.0

Un aspecto del lenguaje C es la forma como se puede simplificar el uso de los operadores aritméticos.

Fundamentos de programación

El proceso de programación

En la siguiente tabla se muestran los operadores aritméticos y sus formas simplificadas. Considere que "x" y "y" son variables de tipo entero (int).

Operador aritmético	Forma simplificada de uso	Ejemplos	Equivalencia	Resultados

+	+=	x = 6;
		x += 5;
		x += y;
			x = 6;
			x = x + 5;
			x = x + y;
				x = 6
				x = 11
				x = 15

-	-=	x = 10;
		y = 5;
		x -= 3;
			x = 10;
			y = 5;
			x = x – 3;
				x = 10
				y = 5
				x = 7

*****	*=	x = 5;
		y = 3;
		x *= 4;
			x = 5;
			y = 3;
			x = x * 4;
				x = 5
				y = 3
				x = 20

/	/=	x = 25;
		y = 3;
		x /= 3;
			x = 25;
			y = 3;
			x = x / y;
				x = 25
				y = 3
				x = 8

%	%=	x = 20;
		y = 3;
		x %= 12;
			x = 20;
			y = 3;
			x = x % 12;
				x = 20
				y = 3
				x = 8

Operadores de incremento y decremento

Los operadores de incremento (++) y decremento (--) suman o restan 1 cada vez que se aplican a una variable. Se pueden utilizar antes o después de la variable. Los resultados son diferentes, como se puede observar en los ejemplos de la siguiente tabla. Considere que "x" y "y" son variables de tipos entero (int).

Operador	Operación	Ejemplos	Resultados
++	Incremento	x = 7; y = x++;	x = 7 y = 7 x = 8
		x = 7; y = ++x;	x = 7 y = 8 x = 8
--	Decremento	x = 6; y = x--;	x = 6 y = 6 x = 5
		x = 6; y = --x;	x = 6 y = 5 x = 5

Referencias

[1] L. J. Aguilar, Fundamentos de programación, Algoritmos, estructura de datos y objetos, México: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A .U., 2008.

[2] O. Cairó, Metología de la programación, Algoritmos, diagramas de flujo y programas, México : Alfaomega Grupo Editor, S.A de C.V., 2014.

Fundamentos de programación

El proceso de programación

Funciones internas

Las operaciones que se requieren en los programas exigen en numerosas ocasiones, además de las operaciones aritméticas básicas, ya tratadas, un número determinado de operadores especiales que se denominan funciones internas, incorporadas o estándar.

Por ejemplo, la función "ln(x)" se puede utilizar para determinar el logaritmo neperiano de un número y la función "sqrt(x)" calcula la raíz cuadrada de un número positivo. Existen otras funciones que se utilizan para determinar las funciones trigonométricas. En la Tabla 1, se observan las funciones más usuales, donde x es el argumento de la función.

Tabla 1: Ejemplo de funciones internas

Función	Significado	Tipo de operandos	Tipo de resultado
abs (x)	Valor absoluto de x	Entero o real	Entero o real
arctg (x)	Arco tangente de x	Entero o real	Real
cos(x)	Coseno de x	Entero o real	Real
ent(x)	Parte entera de x	Entero o real	Real
exp(x)	e^x	Entero o real	Real
ln(x)	Logaritmo natural de x	Entero o real	Real
log(x)	Logaritmo decimal de x	Entero o real	Real
sqrt(x)	Raiz cuadrada de x	Entero o real	Real
sqr(x)	Cuadrado de x	Entero o real	Entero o real
round(x)	Redondea x al entero más cercano	Entero o real	Entero
sin(x)	Seno de x	Entero o real	Real
tan(x)	Tangente de x	Entero o real	Real
trunc(x)	Parte entera de x	Real	Entero

Ejemplo

Tabla 2, ejemplo de funciónes internas con valor

FUNCIÓN	VALOR DE X	RESULTADO
sqrt (x)	25	5
sqr (x)	6	36
trunc (x)	6.58	6
ent (x)	6.58	6
abs (x)	-78	78
round (x)	7.65	8
round (x)	9.68	9

Tabla 3, función resuelta

FUNCIÓN	VALORES	SOLUCIÓN
X = sqr(A) + sqrt (B) + C	A=5	X =sqr(5) + sqrt(4) + 3
	B=4	X=25 + 2 + 3
	C=3	X= 30

Referencias

[1] L. J. Aguilar, Fundamentos de programación, Algoritmos, estructura de datos y objetos, México: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A .U., 2008.

[2] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

[3] O. Cairó, Metología de la programación, Algoritmos, diagramas de flujo y programas, México : Alfaomega Grupo Editor, S.A de C.V., 2014.

Fundamentos de programación

El proceso de la programación

Lenguajes de programación

Un programa se escribe en un lenguaje de programación y las operaciones que conducen a expresar un algoritmo en forma de programa se llaman programación. Así pues, los lenguajes utilizados para escribir programas de computadoras son los lenguajes de programación y programadores son los escritores y diseñadores de programas. El proceso de traducir un algoritmo en pseudocódigo a un lenguaje de programación se denomina codificación, y el algoritmo escrito en un lenguaje de programación se denomina código fuente.

Hoy en día, la mayoría de los programadores emplean lenguajes de programación como C++, C, C#, Java, Visual Basic, XML, HTML, Perl, PHP, JavaScript..., aunque todavía se utilizan, sobre todo profesionalmente, los clásicos COBOL, FORTRAN, Pascal o el mítico BASIC. Estos lenguajes se denominan lenguajes de alto nivel y permiten a los profesionales resolver problemas convirtiendo sus algoritmos en programas escritos en alguno de estos lenguajes de programación.

Los lenguajes de programación se utilizan para escribir programas. Los programas de las computadoras modernas constan de secuencias de instrucciones que se codifican como secuencias de dígitos numéricos que podrán entender dichas computadoras.

Fundamentos de programación

El proceso de la programación

El sistema de codificación se conoce como lenguaje máquina que es el lenguaje nativo de una computadora. Desgraciadamente la escritura de programas en lenguaje máquina es una tarea tediosa y difícil ya que sus instrucciones son secuencias de 0 y 1 (patrones de bit, tales como 11110000, 01110011...) que son muy difíciles de recordar y manipular por las personas.

En consecuencia, se necesitan lenguajes de programación "amigables con el programador" que permitan escribir los programas para poder "charlar" con facilidad con las computadoras.

Ilustración 1, ejemplo de los diferentes lenguajes de programación que existen.

Referencias

[1] L. J. Aguilar, Fundamentos de programación, Algoritmos, estructura de datos y objetos, México: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A .U., 2008.

[2] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

[3] O. Cairó, Metología de la programación, Algoritmos, diagramas de flujo y programas, México : Alfaomega Grupo Editor, S.A de C.V., 2014.

Fundamentos de programación

El proceso de programación

Algoritmo

Los humanos efectuamos cotidianamente series de pasos, procedimientos o acciones que nos permiten alcanzar algún resultado o resolver algún problema. Esta serie de pasos, procedimientos o acciones, comenzamos a aplicarlas desde que empieza el día, cuando, por ejemplo, decidimos bañarnos.

Posteriormente, cuando tenemos que ingerir alimentos también seguimos una serie de pasos que nos permiten alcanzar un resultado específico: tomar el desayuno. La historia se repite innumerables veces durante el día. En realidad, todo el tiempo estamos aplicando algoritmos para resolver problemas.

Muchas veces aplicamos el algoritmo de manera invertida, inconscientemente o automática. Esto ocurre generalmente cuando el problema al que nos enfrentamos lo hemos resuelto con anterioridad con gran número de veces.

Supongamos, por ejemplo, qué tenemos que abrir una puerta. Lo hemos hecho tantas veces que difícilmente nos tomamos la molestia de enumerar los pasos para alcanzar este objetivo. Lo hacemos de manera automática.

Lo mismo ocurre cuando nos subimos a un automóvil, lustramos nuestros zapatos, hablamos por teléfono, nos vestimos, cambiamos la llanta de un automóvil o simplemente cuando tomamos un vaso con agua.

Ejemplo

	Ejemplo 1: Receta de helado de fresa
Ingredientes	▪ 500 gr. de fresas. ▪ 100 gr. de azúcar. ▪ 25 ml. de azúcar invertido. ▪ 200 ml. de leche entera. ▪ 225 ml. de nata para montar. ▪ 1 limón.
Preparación del helado de fresa	1. Cortar las fresas en trocitos. Y mezclarlas con azúcar. Dejar que repose durante hora y media. 2. Añadir la leche, unas gotitas de limón y el azúcar invertido, triturar todo con la batidora. Colar para eliminar las semillas. 3. Poner la nata hasta que se formen puntas firmes. Añadir el molido de fresas lentamente, moviendo ampliamente, pero con cuidado sin que se baje. 4. Colocar la mezcla en el refrigerador y sacarla cuando transcurra una hora.

Fundamentos de programación

El proceso de programación

	Ejemplo 2: Receta de pechuga de pollo en salsa de elote y chile poblano
Ingredientes **(6 personas)**	▪ 3 PECHUCHAS SIN HUESO Y SIN PIEL, PARTIDAS A LA MITAD. ▪ 1 DIENTE DE AJO. ▪ 3 GRAMOS DE PIMIENTA NEGRA. ▪ SAL AL GUSTO.
Preparación	1. Moler el ajo, pimienta y un poco de sal, luego untarlo a las pechugas. 2. Calentar el aceite y dore las pechugas. 3. Licuar los chiles con la leche y la crema, mezclar con la crema de elote. 4. En una fuente colocar las pechugas y bañarlas con la mezcla anterior. 5. Cubrir el platón con papel aluminio y hornear a 200C, durante 15 min.

Referencias

[1] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

[2] O. Cairó, Metología de la programación, Algoritmos, diagramas de flujo y programas, México : Alfaomega Grupo Editor, S.A de C.V., 2014.

[3] G. Polya, Cómo plantear y resolver problemas, Decimoquinta ed., Mexíco D.F.: Editorial Trillas S.A de C.V., 1965.

Fundamentos de programación

El proceso de programación

Diagrama de flujo

Se define al diagrama de flujo como una esquematización visual de un algoritmo, es decir que en muestra de manera gráfica los pasos que se deben seguir para lograr una solución a un problema.

Para construir un diagrama de flujo es de suma importancia utilizarla simbología correcta, (ver Ilustración 1), dado que se va a partir de esta representación para realizar un programa mediante la codificación.

	Indica el inicio o fin de un proceso
	Indica cada actividad que necesita ser ejecutada
	Indica un ponto de toma de decisión
	Indica la dirección de flujo
	Indica los documentos utilizados en el proceso
	Indica una espera
	Indica que el flujograma continua a partir de ese punto en otro circulo, con la misma letra o número, que aparece en su interior

Ilustración 1: Simbología de un diagrama de flujo.

Fundamentos de programación

El proceso de programación

Reglas para la construcción de un diagrama de flujo

1. Cada diagrama de flujo debe contar con un **INICIO** y un **FIN**.
2. Se deben utilizar líneas para señalar la dirección del flujo del diagrama, las cuales pueden ser rectas, horizontales o verticales.
3. Todas las líneas que indican la dirección de flujo del diagrama deben estar conectadas con un símbolo.
4. El diagrama de flujo deberá estar construido de arriba hacia abajo y de izquierda a derecha.
5. La notación utilizada en cada diagrama de flujo debe ser independiente del lenguaje de programación.
6. Se recomienda utilizar comentarios que ayuden a entender los procedimientos realizados.
7. En caso de que el diagrama de flujo requiera más de una hoja para la construcción, debemos utilizar los conectores adecuados y enumerar las páginas.
8. No se pueden utilizar más de una línea para conectar un símbolo.

Un ejemplo básico de las etapas de un diagrama de flujo se puede apreciar en la Ilustración 2, donde también se pueden observar que se cumplen las reglas básicas para la construcción de un diagrama de flujo.

Ilustración 2: etapas en la construcción de un diagrama de flujo.

Ejemplo

A continuación, se muestra un ejemplo donde se pretende cocinar un huevo para otra persona.

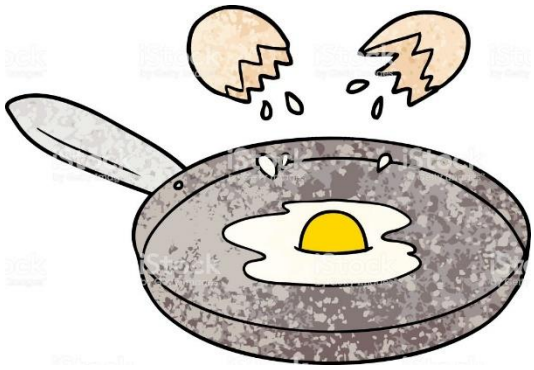


Ilustración 3, pasos para cocinar un huevo

1. Preguntar si quiere el huevo frito.
2. Si dice que si, se fríe, si dice que no, se hierve.
3. Una vez cocinado preguntar si quiere sal en el huevo.
4. Si la respuesta es no, servir el huevo en el plato, en el caso contrario, ponerle sal y después servir en el plato.

Analizando los pasos, se puede apreciar que los pasos no pueden cambiar su posición. Sería imposible preguntar si se quiere un huevo frito después de haberlo hervido, por ejemplo. Es muy importante que los pasos sean una secuencia **lógica** y **ordenada**.

Ahora que se analizaron todos los pasos, mediante el algoritmo, se puede hacer un esquema con estos pasos a seguir, (ver Ilustración 4).

Diagrama de flujo

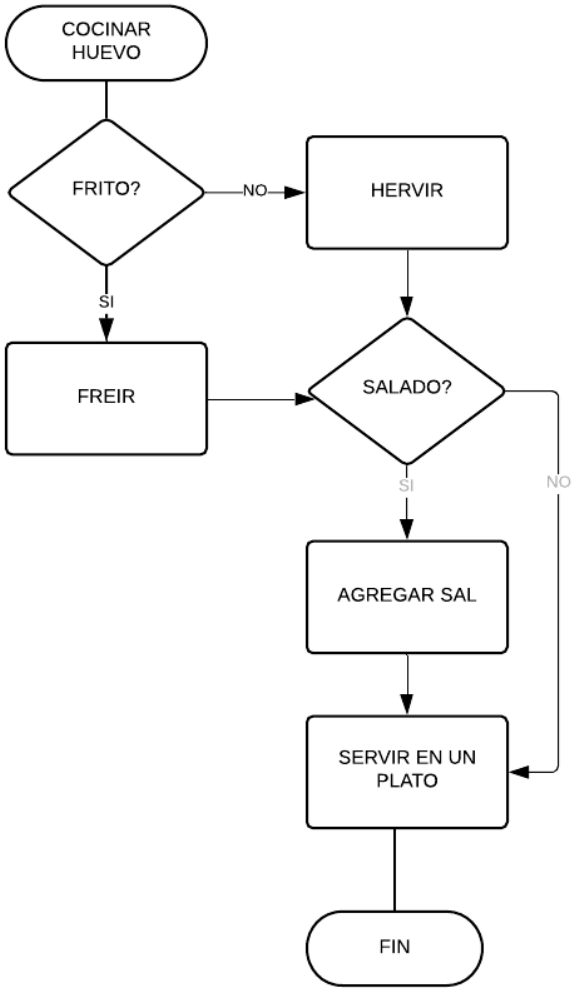


Ilustración 4: Ejemplo de diagrama de flujo para cocinar un huevo.

Referencias

[1] L. J. Aguilar, Fundamentos de programación, Algoritmos, estructura de datos y objetos, México: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A .U., 2008.

[2] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

[3] O. Cairó, Metología de la programación, Algoritmos, diagramas de flujo y programas, México : Alfaomega Grupo Editor, S.A de C.V., 2014.

[4] J. B. V. GOMEZ, Análisis y diseño de algoritmos, Tlalnepantla: RED TERCER MILENIO S.C, 2012.

Fundamentos de programación

El proceso de programación

Pseudocódigo

El pseudocódigo es un lenguaje de especificación (descripción) de algoritmos. El uso de tal lenguaje hace el paso de codificación final (esto es, la traducción a un lenguaje de programación) relativamente fácil. Los lenguajes APL Pascal y Ada se utilizan a veces como lenguajes de especificación de algoritmos.

El pseudocódigo nació como un lenguaje similar al inglés y era un medio de representar básicamente las estructuras de control de programación estructurada. Se considera un primer borrador, dado que el pseudocódigo tiene que traducirse posteriormente a un lenguaje de programación.

El pseudocódigo **no puede ser ejecutado** por una computadora. La ventaja del pseudocódigo es que, en su uso, en la planificación de un programa, el programador se puede concentrar en la lógica y en las estructuras de control y no preocuparse de las reglas de un lenguaje específico.

Es también fácil modificar el pseudocódigo si se descubren errores o anomalías en la lógica del programa, mientras que en muchas ocasiones suele ser difícil el cambio en la lógica, una vez que está codificado en un lenguaje de programación. Otra ventaja del pseudocódigo es que puede ser traducido fácilmente a lenguajes estructurados como Pascal, C, FORTRAN 77/90, C++, Java, C#, etc.

Fundamentos de programación

El proceso de programación

El pseudocódigo original utiliza para representar las acciones sucesivas palabras reservadas en inglés —similares a sus homónimas en los lenguajes de programación, tales como **start**, **end**, **stop**, **if-then-else, while-end**, **repeat-until**, etc.

La escritura de pseudocódigo exige normalmente la indentación (sangría en el margen izquierdo) de diferentes líneas.

Una representación en pseudocódigo de un problema de cálculo del salario neto de un trabajador se muestra en la siguiente tabla:

Tabla 1, pseudocódigo de cálculo de salario neto de un trabajador.

	Ejemplo 1: cálculo de impuesto y salarios
Pseudocódigo en ingles	**start** **read** nombre, horas, precio salario <- horas * precio tasas <- 0,25 * salario salario_neto <- salario -tasas **write** nombre, salario, tasas, salario **end**

Ejemplo

Construya un diagrama de flujo que, dadas la base y altura de un triángulo, **calcule** e **imprima** su **área**.

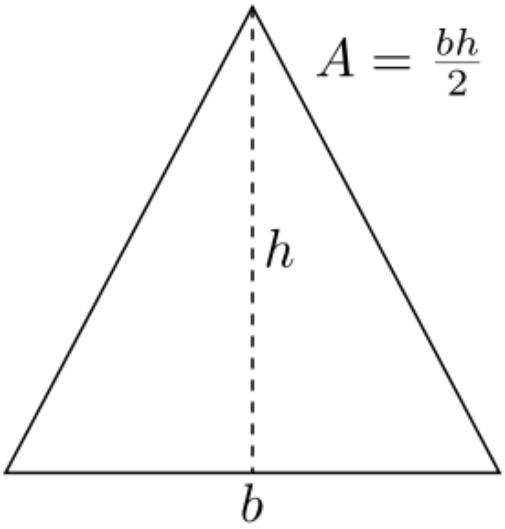


Ilustración 1, fórmula para calcular el área de un triángulo

Análisis

Para calcular el área de un triángulo, es necesario identificar los valores de la base y la altura, estas variables son suficientes como se muestra en la ilustración 2.

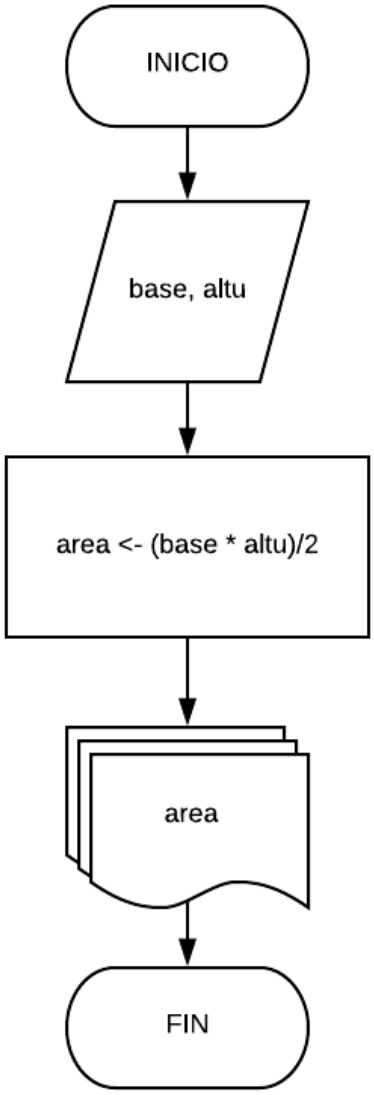
$$A = \frac{bh}{2}$$

Ilustración 2, Formula para calcular el área de un triángulo.

Se crearán las variables **base**, **altu** y **area** donde:

- **Base**: es la variable de tipo real que indica la base del triángulo.
- **Altu**: es una variable de tipo real que representa la altura del triángulo.

Diagrama de flujo



Pseudo-código

	Ejemplo 1: cálculo de impuesto y salarios

Pseudocódigo	**Inicio**
	
	  Leer base y altura
	
	  area <- (base * altura) /2
	
	  Mostrar area
	
	Fin

{/} Código

Lenguaje	Código

C	`float altura, area, base;`
	`  printf( "Introduzca base: " );`
	`  scanf( "%f", &base );`
	`  printf( "Introduzca altura: " );`
	`  scanf( "%f", &altura );`
	`  area = base * altura / 2;`
	`  printf( "El area del triangulo es: %f", area );`
	`return 0;`

Java	`Scanner sc = new Scanner(System.in);`
	`  double b,h;`
	`  System.out.println("Ingresa base");`
	`  b=sc.nextDouble();`
	`  System.out.println("Ingresa altura");`
	`  h=sc.nextDouble();`
	`  double area;`
	`  area=b*h/2;`
	`  System.out.print(area);`

C++	`double b,h;`
	`  cout<<"Ingresa base"<<endl;`
	`  cin>>b;`
	`  cout<<"Ingresa altura"<<endl;`
	`  cin>>h;`
	`  double area;`
	`  area=b*h/2;`
	`cout<<area;`
	`return 0;`

Referencias

[1] L. J. Aguilar, Fundamentos de programación, Algoritmos, estructura de datos y objetos, México: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A .U., 2008.

[2] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

[3] O. Cairó, Metología de la programación, Algoritmos, diagramas de flujo y programas, México : Alfaomega Grupo Editor, S.A de C.V., 2014.

Fundamentos de programación

Control de flujo de datos

Estructura secuencial

La estructura secuencial es aquella en donde cada instrucción sigue a otra en secuencia. Tienen una entrada y una salida, es decir, las instrucciones son ejecutadas de tal manera que la salida de una es la entrada de la siguiente, sucesivamente hasta el fin del programa.

La asignación consiste en pasar los valores o resultados a una zona de memoria, en donde será identificado con el nombre de la variable que recibe el valor. Dicha asignación se puede clasificar de la siguiente forma:

Simples: se pasa un valor constante a una variable, (b←15).

Contador: se usa como un verificador del número de veces que se realiza un proceso (a←a+1)

Acumulador: consiste en utilizarla como sumador en un proceso, (a←a+b).

De trabajo: se recibe un resultado de una operación matemática que involucre diversas variables, (a←c+b*2/4).

Fundamentos de programación

Control de flujo de datos

Ejemplo

Crear un programa que calcule e imprima el número de **segundos** que hay en un determinado número de **días**.

Ilustración 1, reloj de manecillas.

Análisis

Para empezar, se deben identificar las variables necesarias para calcular los segundos que tienen un cierto número de días. Por lo tanto, se puede observar que el enunciado del problema ya menciona dos variables: **segundos** y **días**.

Donde:

- **Días:** Es una variable de tipo entero y expresa el total de días que se desea calcular.
- **Seg:** Es una variable de tipo entero, que guardará la cantidad de segundos que hay en un número determinado de días.

Fundamentos de programación

Control de flujo de datos

Para calcular la cantidad de segundos que tiene un día, se debe tener en cuenta lo siguiente:

- Horas que tiene un día (24 horas)
- Minutos que tiene una hora (60 minutos)
- Segundos que tiene una hora (60 segundos)

Por lo tanto, se puede definir la siguiente fórmula para calcular los segundos de un día:

Segundos de un día =24 horas * 60 minutos * 60 segundos

Y si se desea calcular los segundos en una cantidad de días en específico, solo se debe multiplicar la fórmula anterior por el número de días.

Segundos = nDías * 24 horas * 60 minutos * 60 segundos

Diagrama de flujo

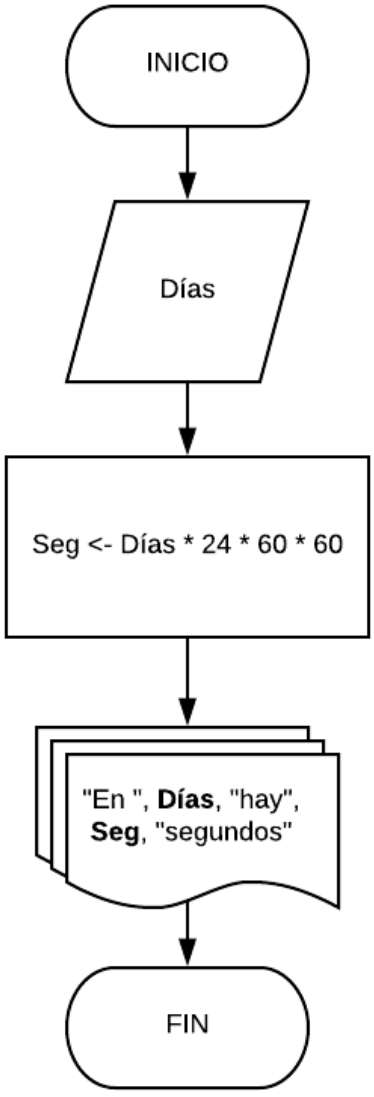


Ilustración 2, Diagrama de flujo que calcula los segundos que tiene un número de días.

Pseudo-código

	Ejemplo: calcular los segundos que tiene un número de días
Pseudocódigo	**Inicio** **Leer** los días que se desean convertir a segundos Calcular los segundos <-Días *24 horas* 60 minutos* 60 seg. **Mostrar** los segundos **Fin**

Fundamentos de programación

Control de flujo de datos

{/} Código

Lenguaje	Código

C	`int Días,Seg;`
	`printf("ingrese los días que desea calcular");`
	`scanf("%d", &Días);`
	`Seg= Días * (24*60*60);`
	`printf("En %d hay %d segundos",Días,Seg);`
	`return 0;`

Java	`Scanner sc = new Scanner(System.in);`
	`int Días,Seg;`
	`printf("ingrese los días que desea calcular");`
	`scanf(Días);`
	`Seg= Días * (24*60*60);`
	`printf("En "+Días+" hay "+Seg+"segundos");`

C++	`int Días,Seg;`
	`cout<<"ingrese los días que desea calcular"<<endl;`
	`cin>>Días;`
	`Seg= Días * (24*60*60);`
	`cout<<"En "<< Días<" hay ","<< Seg<<" segundos";`
	`return 0;`

Referencias

[1] L. J. Aguilar, Fundamentos de programación, Algoritmos, estructura de datos y objetos, México: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A .U., 2008.

[2] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

[3] O. Cairó, Metología de la programación, Algoritmos, diagramas de flujo y programas, México : Alfaomega Grupo Editor, S.A de C.V., 2014.

Fundamentos de programación

Control de flujo de datos

Selección IF

La estructura selectiva **IF** permite que el flujo del diagrama sigue por un camino específico si se cumple una condición determinada. Si al evaluar una condición del resultado es verdadero, entonces se sigue por un camino específico – hacia bajo – y se ejecuta una operación o acción o un conjunto de ella.

Por otra parte, si el resultado de la evaluación es falso, entonces se pasa (n) por alto esa (s) operación (es). En ambos casos se continúa con la secuencia normal del diagrama de flujo, (Ver Ilustración 1).

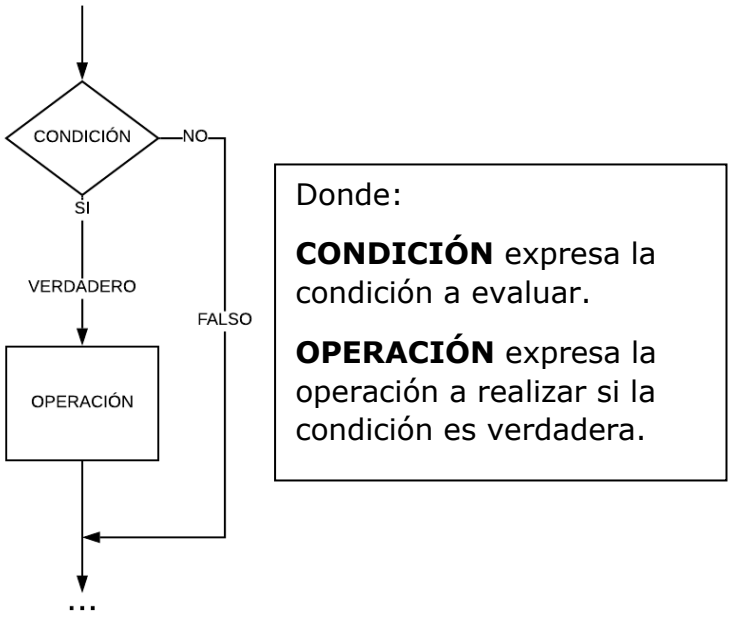


Ilustración 1, Diagrama de flujo de estructura IF.

Fundamentos de programación

Control de flujo de datos

Ejemplo

Conociendo como dato el sueldo de un trabajador, aplique un **aumento del 15%** si su sueldo es inferior a **$1,300**. Muestre en este caso el nuevo sueldo del trabajador.

Ilustración 2, sueldo de un trabajador.

Análisis

En este problema, se puede observar que se tienen dos constantes: **$1,300** y **15%**, las cuales nos ayudaran a encontrar el valor que se pide (Nuevo sueldo). Por lo tanto, se tienen tres variables: **sueldo** (sueldo base), **aumento** (Sueldo + 15%) y **nsueldo,** que corresponde al nuevo sueldo (sueldo + aumento).

Además, en esta ocasión se puede observar que existe una **condición**, la cual es que se aplique el 15% al sueldo base **si** el este es inferior a **$1,300**, (ver tabla 1).

Tabla 1, Datos del problema.

Variables	Constantes	Condición
sueldo	15%	Sueldo < 1,300
aumento	$1,300	
nsueldo		

Diagrama de flujo

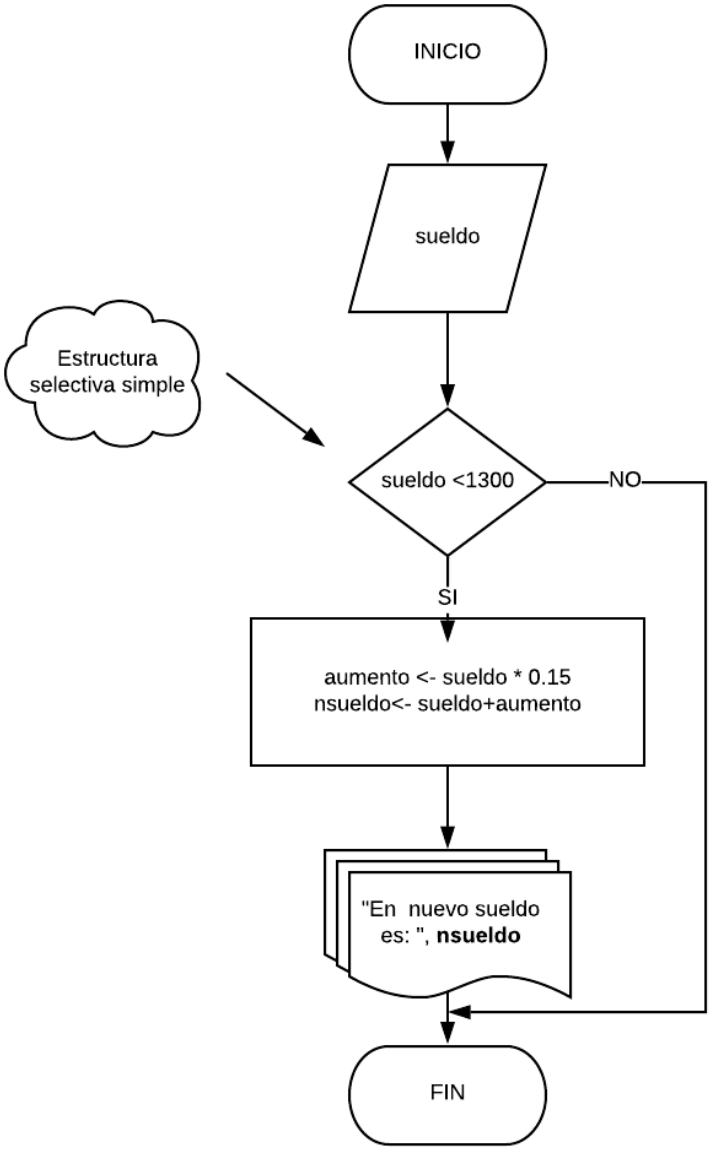


Ilustración 3, Diagrama de flujo del ejemplo.

Fundamentos de programación

Control de flujo de datos

Pseudo-código

	Ejemplo: calcular los segundos que tiene un número de días
Pseudocódigo	**Inicio** **Leer** el sueldo base del trabajador Evaluar si el sueldo es menor < 1,300 **Si** es mayor entonces, calcular el aumento y sumarlo al sueldo **Mostrar** el nuevo sueldo **Fin**

Fundamentos de programación

Control de flujo de datos

{/} Código

Lenguaje	Código

C	`float sueldo, aumento, nsueldo;`
	`printf("ingrese el sueldo: ");`
	`scanf("%f", &sueldo);`
	`if(sueldo < 1300){`
	`aumento = sueldo * 0.15;`
	`nsueldo = sueldo + aumento;`
	`}`
	`printf("EL nuevo sueldo es: %f", nsueldo );`
	`return 0;`

Java	`Scanner sc=new Scanner(System.in);`
	`float sueldo, aumento, nsueldo;`
	`System.out.println("ingrese el sueldo: ");`
	`sueldo=sc.nextInt();`
	`if(sueldo < 1300){`
	`aumento = sueldo * 0.15;`
	`nsueldo = sueldo + aumento;`
	`}`
	`System.out.println("\nEL nuevo sueldo es: " + nsueldo);`

C++	`float sueldo, aumento, nsueldo;`
	`cout<<"ingrese el sueldo: "<<endl;`
	`cin>>sueldo;`
	`if(sueldo < 1300){`
	`aumento = sueldo * 0.15;`
	`nsueldo = sueldo + aumento;`
	`}`
	`cout<<"EL nuevo sueldo es: "<<nsueldo<< endl;`
	`return 0;`

Referencias

[1] L. J. Aguilar, Fundamentos de programación, Algoritmos, estructura de datos y objetos, México: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A .U., 2008.

[2] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

[3] O. Cairó, Metología de la programación, Algoritmos, diagramas de flujo y programas, México : Alfaomega Grupo Editor, S.A de C.V., 2014.

Fundamentos de programación

Control de flujo de datos

IF-ELSE

La estructura doble **if – else** permiten que el flujo el diagrama se bifurque por dos ramas diferentes en el punto de la toma de decisión. Si al evaluar la condición El resultado es verdadero, entonces se sigue por un camino específico – el de la izquierda – y se ejecutan la acción determinada o un conjunto de ellas.

Por otra parte, si el resultado de la evaluación es falso, entonces se sigue por otro camino – el de la derecha – y se realiza (n) otra (s) acción (es). En ambos casos, luego de ejecutar las acciones correspondientes, se continúa con la secuencia normal del diagrama de flujo, (ver Ilustración 1).

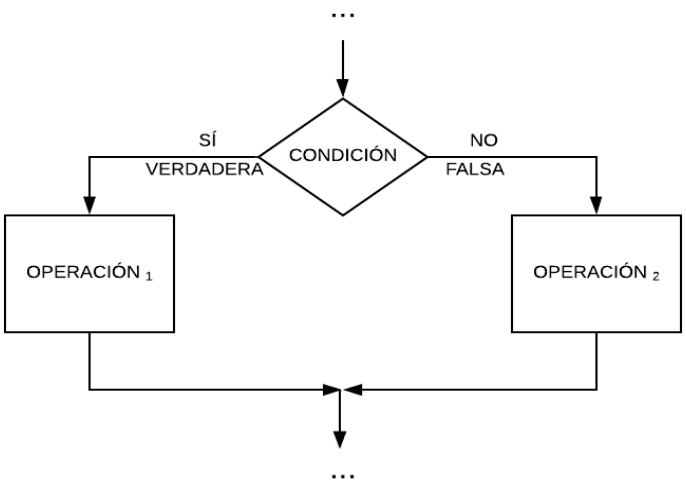


Ilustración 1, estructura if-else

Fundamentos de programación

Control de flujo de datos

Ejemplo

Desarrollar un programa el cual, dependiendo de la edad de una persona, se desea mostrar un mensaje de que es o no **mayor de edad**, suponiendo que una persona mayor de edad tiene por lo menos **18 años**.

Ilustración 2, Persona mayor de edad

Análisis

Como primer paso, analizando el problema se anterior, se puede observar que es necesario solicitar la entrada de la **edad** (variable de tipo entero). Además, el problema menciona una condición, la cual es mostrar un mensaje de **si es** o **no** mayor de 18 años, es decir, mayor de edad, (ver Tabla 1).

Tabla 1, Datos del problema.

Variables	Constantes	Condición
edad	18 años	**edad > 18**

Algoritmo

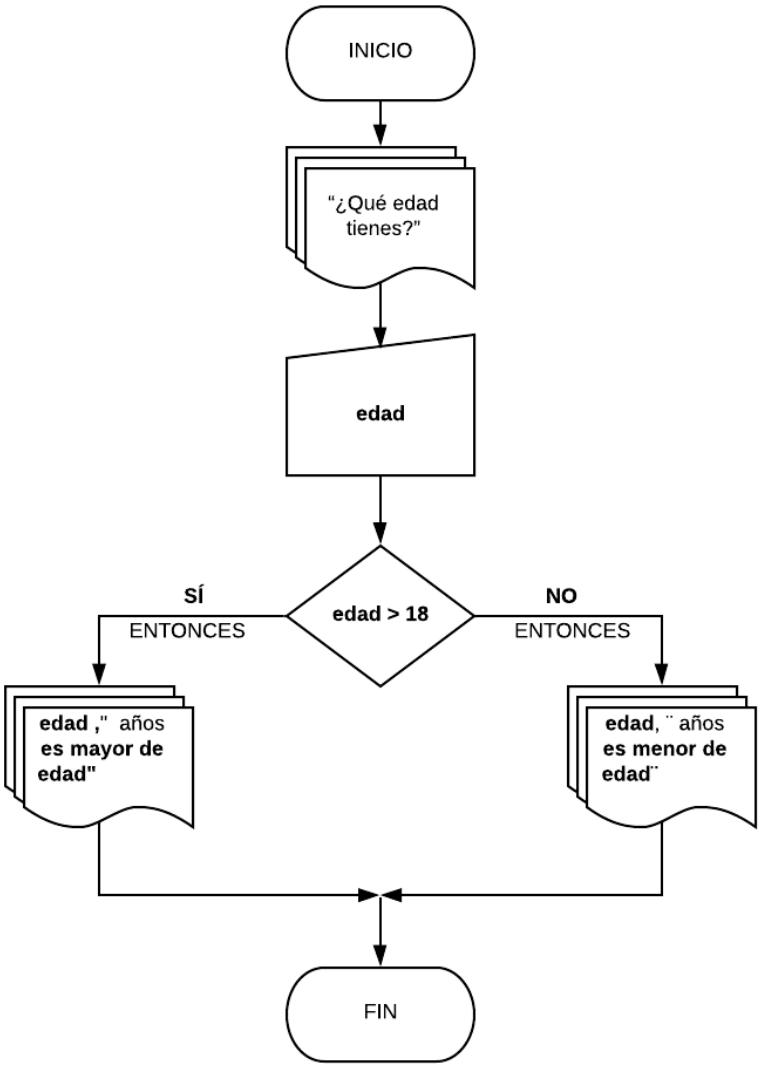


Ilustración 3, estructura if-else para saber si una persona es o no mayor de edad.

Fundamentos de programación

Control de flujo de datos

Pseudo-código

	Ejemplo: Programa que muestra si una persona es o no mayor de edad
Pseudocódigo	**Inicio** **Mostrar** “¿Qué edad tienes?” **Leer** edad **Si Edad >20 Entonces** **Mostrar** “Eres mayor de edad” **Si No** **Mostrar** “Eres menor de edad” **Fin**

Fundamentos de programación

Control de flujo de datos

{/} Código

Lenguaje	Código

C	`int edad;`
	`printf("¿Qué edad tienes? ");`
	`scanf("%d", &edad);`
	`if(edad > 18){`
	`printf("%d años, es mayor de edad", edad);`
	`}else{`
	`printf"%d años, es menor de edad", edad);`
	`}`
	`return 0;`

Java	`Scanner sc=new Scanner(System.in);`
	`int edad;`
	`System.out.println("¿Qué edad tienes? ");`
	`edad=sc.nextInt();`
	`if(edad > 18){`
	`System.out.println(edad+ " años, es mayor de edad);`
	`} else{`
	`System.out.println(edad+ " años, es menor de edad);`
	`}`

C++	`int edad;`
	`cout<<"¿Qué edad tienes?"<<endl;`
	`cin>>edad;`
	`if(edad > 18){`
	`cout<<edad<<" años, es mayor de edad"<< endl;`
	`} else {`
	`cout<<edad<<" años, es menor de edad"<< endl; }`
	`return 0;`

Referencias

[1] L. J. Aguilar, Fundamentos de programación, Algoritmos, estructura de datos y objetos, México: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A .U., 2008.

[2] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

[3] O. Cairó, Metología de la programación, Algoritmos, diagramas de flujo y programas, México : Alfaomega Grupo Editor, S.A de C.V., 2014.

Fundamentos de programación

Control de flujo de datos

SWITCH

La estructura selectiva **switch** permite que el diagrama de flujo se bifurque por varias ramas en el punto de la toma de decisión. La elección del camino a seguir depende del contenido de la variable conocida como selector, la cual puede tomar valores de un conjunto previamente establecidos.

El camino elegido, entonces dependerá del valor que tome el selector. Así, si el selector toma el valor 1, se ejecutará la acción 1; si toma el valor 2, se ejecutará la acción 2, y si toma el valor n coma se realizará la acción en punto a continuación se presenta la figura que ilustra la estructura selectiva, (ver Ilustración 1).

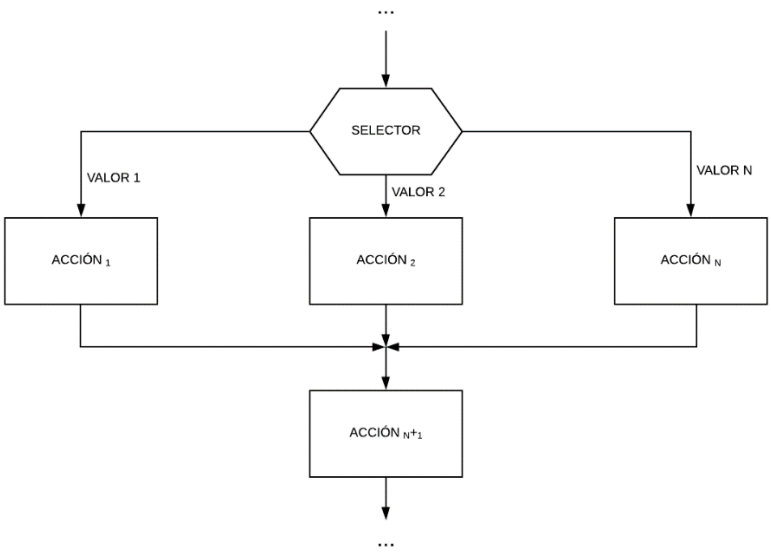


Ilustración 1, diagrama de flujo de la estructura case

Fundamentos de programación

Control de flujo de datos

Ejemplo

Desarrollar un programa que, dado como datos la categoría y el sueldo de un trabajador, se **calcule y muestre** en pantalla el aumento correspondiente, tome en cuenta la Tabla 1.

Tabla1, relación de aumento en los sueldos

CATEGORÍA	AUMENTO
1	18%
2	12%
3	10%
4	8%

Análisis

Obsérvese que en el enunciado y tabla anterior se puede identificar 2 variables, el **sueldo** (suel, variable tipo real), **nuevo sueldo** (nsueldo, variable tipo real) y **categoría** (categ, variable d etipo entero), ver la tabla 1.

Tabla 2, Datos del problema.

Variables	Constantes	Calcular nuevo sueldo
categ	18%	Nsueldo=suel * 1.18
sueldo	12%	Nsueldo=suel * 1.12
nsueldo	10%	Nsueldo=suel * 1.10
	8%	Nsueldo=suel * 1. 08

Fundamentos de programación

Control de flujo de datos

Algoritmo

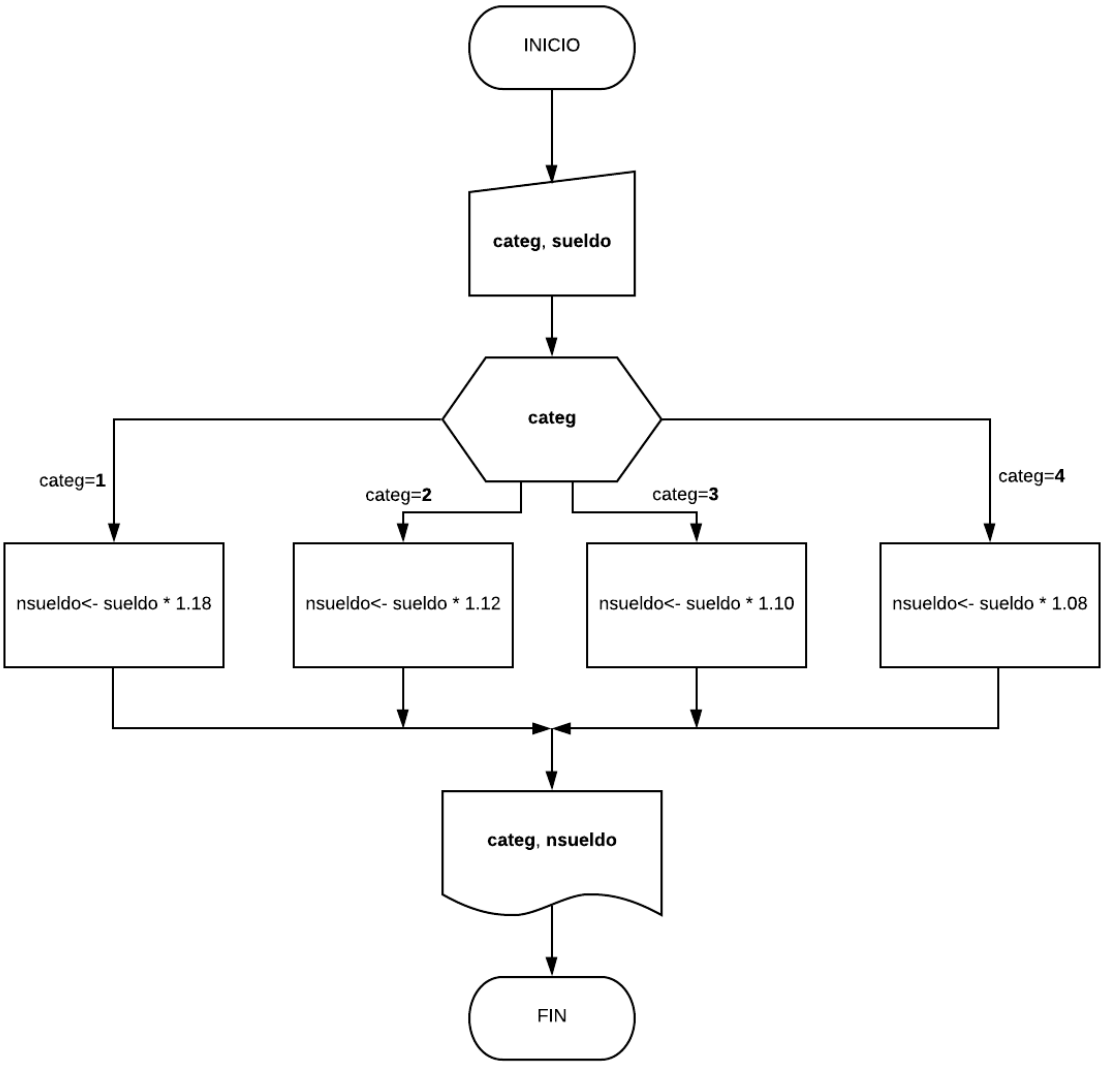


Ilustración 2, diagrama de flujo del problema

Fundamentos de programación

Control de flujo de datos

Pseudo-código

	Ejemplo: Incremento de sueldos
Pseudocódigo	**Inicio** **Mostrar** "Ingrese la categoría y sueldo del trabajador: " **Leer** categoría, sueldo **Si categoría es igual a** **1: hacer** nuevo sueldo← sueldo * 1.18 **2: hacer** nuevo sueldo← sueldo * 1.12 **3: hacer** nuevo sueldo← sueldo * 1.10 **4: hacer** nuevo sueldo← sueldo * 1.08 **Fin de la estructura switch** **Mostrar** categoría y nuevo sueldo **Fin**

Fundamentos de programación

Control de flujo de datos

{/} Código

Lenguaje	Código

C	`float sueldo, nsueldo;`
	`int categ;`
	`printf("Ingrese la categoría del trabajador 1[18%] 2[12%] 3[10%] 4[8%] \n");`
	`scanf("%d", &categ);`
	`printf("Ingrese el sueldo trabajador: ");`
	`scanf("%f", &sueldo);`
	`switch (categ)`
	`case 1: nsueldo=sueldo * 1.18`
	`break;`
	`case 2: nsueldo=sueldo * 1.12`
	`break;`
	`case 3: nsueldo=sueldo * 1.10`
	`break;`
	`case 4: nsueldo=sueldo * 1.08`
	`break;`
	`default printf("Ingrese una categoría correcta: ");`
	`printf("Categoría: %d \n Nuevo sueldo: %f",categ,nsueldo);`
	`return 0;`

Java	`Scanner sc=new Scanner(System.in);`
	`float sueldo, nsueldo;`
	`int categ;`
	`System.out.println("Ingrese la categoría del trabajador 1[18%] 2[12%] 3[10%] 4[8%] \n ");`
	`categ=sc.nextInt();`
	`System.out.println("Ingrese el sueldo trabajador: ");`
	`sueldo=sc.nextInt();`
	`switch (categ)`
	`case 1: nsueldo=sueldo * 1.18 break;`
	`case 2: nsueldo=sueldo * 1.12 break;`
	`case 3: nsueldo=sueldo * 1.10 break;`
	`case 4: nsueldo=sueldo * 1.08 break;`
	`default System.out.println("Ingrese una categoría correcta: ");`
	
	`System.out.println("Categoría: "+categ);`
	`System.out.println("Nuevo sueldo: "+nsueldo);`

C++

```
float sueldo, nsueldo;
int categ;
 cout<<Ingrese la categoría del trabajador 1[18%] 2[12%] 3[10%] 4[8%]
\n"<<endl;
    cin>>categ;
    cout<<Ingrese el sueldo trabajador: "<<endl;
    cin>>sueldo;
    switch (categ)
           case 1: nsueldo=sueldo * 1.18
                   break;
           case 2: nsueldo=sueldo * 1.12
                   break;
           case 3: nsueldo=sueldo * 1.10
                   break;
           case 4: nsueldo=sueldo * 1.08
                   break;
     default printf("Ingrese una categoría correcta: ");
     cout<<"Categoría: "<<categ<<endl;
     cout<<"Nuevo sueldo: "<<nsueldo<<endl;
return 0;
```

Referencias

[1] O. Cairó, Metología de la programación, Algoritmos, diagramas de flujo y programas, México : Alfaomega Grupo Editor, S.A de C.V., 2014.

[2] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

Fundamentos de programación

Ciclos iterativos

FOR

Eso es la estructura algorítmica utilizada para repetir un conjunto de instrucciones un número definido de veces. Este tipo de estructura se encuentra prácticamente en todos los lenguajes de programación. Es similar a la estructura do de Fortran y for de Pascal, (ver Ilustración 2).

Es muy importante destacar que hay dos tipos de variables que se utilizan frecuentemente en los ciclos. Éstas se conocen como **contadores** y **acumuladores**. Los contadores, como su nombre lo indica, sirven para contar, y los acumulado-res, para acumular. Ambas variables se inicializan generalmente en cero antes de iniciar el ciclo, aunque este valor puede ser diferente dependiendo del problema que se vaya a resolver.

Ilustración 1, representación de un ciclo.

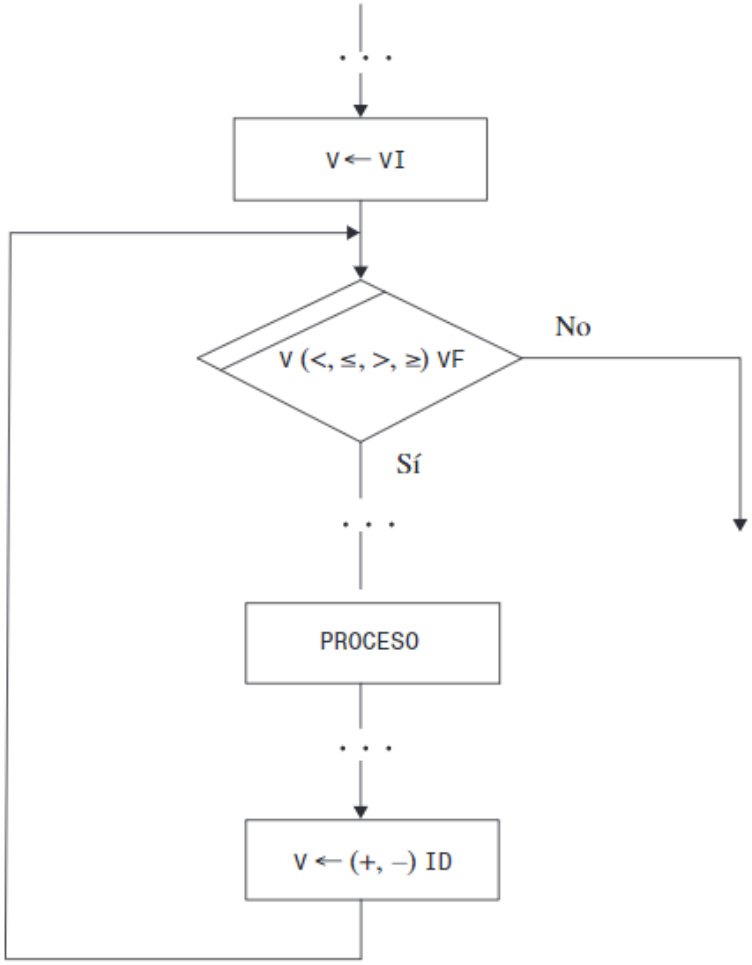


Ilustración 2, representación gráfica del ciclo for.

Donde:

- **V** representa la variable de control del ciclo
- **VI** expresa el valor inicial
- **VF** representa al valor final
- **ID** representa el incremento o decremento de la variable de control

Ejemplo

Construye un programa que, al recibir como datos los **salarios de 15 profesores** de una universidad, obtenga **el total de la nómina**.

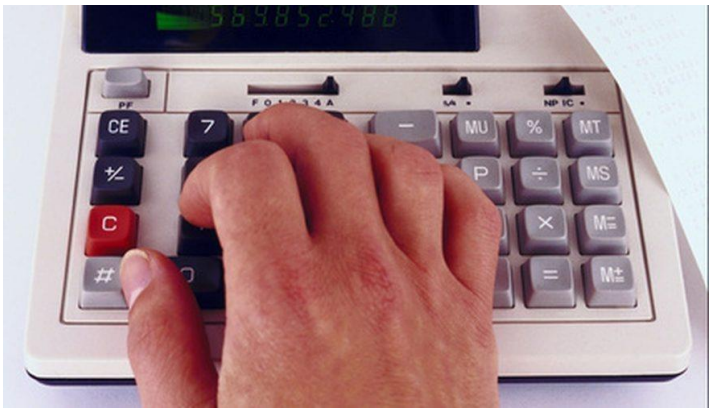


Ilustración 3, calculadora.

Análisis

Obsérvese que, es necesario solicitar 15 salarios de profesores (Sal_1, Sal_2, … Sal_{15}), los cuales se pueden representar como **Sal_i ($1 \leq i \leq 15$),** Donde **i** es el salario de cada trabajador.

Tabla 1, Datos del problema.

Variables	Descripción
I	Variable de **tipo entero** que representa la **variable de control del ciclo.**
NOM	Es una variable de **tipo real** que **acumula** los salarios.
SAL	Variable de **tipo real** que representa el **salario**.

Algoritmo

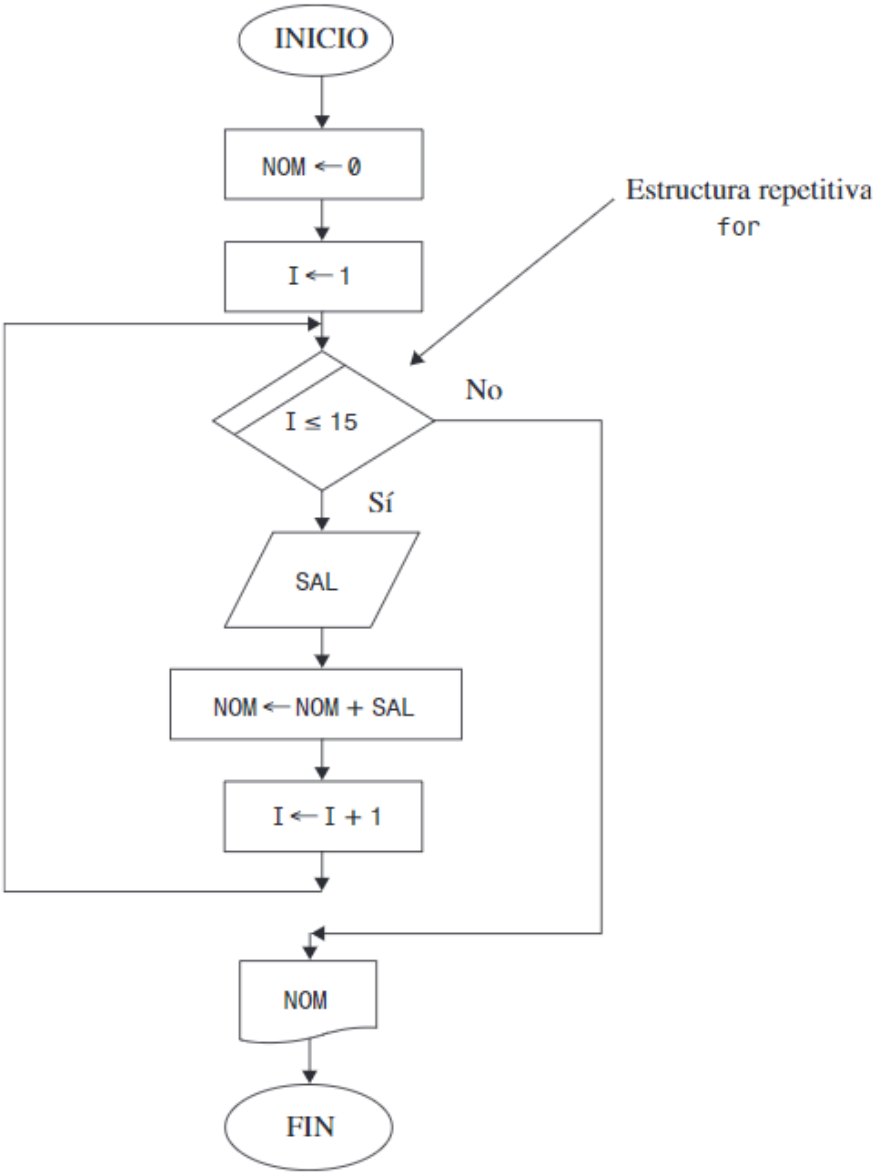


Ilustración 4, Diagrama de flujo del algoritmo

Fundamentos de programación

Ciclos iterativos

Pseudo-código

	Ejemplo: Calcula la nómina
Pseudocódigo	**Inicio** **Inicializar en 0 el acumulador nómina** **Inicializar en 1 el contador i** **Repetir** desde **i ≤ 15** **Mostrar** "Ingrese el sueldo del trabajador: " **Leer** sueldo **Calcular** nomina **NOM ← NOM + SAL** **Incrementar** contador **I←I+1** **Fin del ciclo for** **Mostrar** NOM **Fin**

Fundamentos de programación

Ciclos iterativos

{/} Código

Lenguaje	Código

C	`float SAL, NOM; // Variable salario y nómina`
	`int i;// Contador`
	`NOM=0; // Se inicializa en 0 la nómina`
	`for (i=1; i≤15; i++) // Ciclo for`
	`{`
	`printf("\Ingrese el salario del profesor %d:", i);`
	`scanf("%f", &SAL);`
	`NOM = NOM + SAL;`
	`}`
	`printf("\nEl total de la nómina es: %.2f", NOM); // Muestra la nómina total.`
	`return 0;`

Java	`Scanner sc=new Scanner(System.in);`
	`float SAL, NOM=0;`
	`int i;`
	`for (i=1; i≤15; i++) // Ciclo for`
	`{`
	`System.out.println("\nIngrese el salario del profesor "+i);`
	`SAL=sc.nextInt();`
	`NOM = NOM + SAL;`
	`}`
	`System.out.println("\nEl total de la nómina es: "+NOM);`

C++	`float SAL, NOM;`
	`int i;`
	`NOM=0;`
	`for (i=1; i≤15; i++)`
	`{`
	`cout<<"\nIngrese el salario del profesor "<<i<<endl;`
	`cin>>SAL;`
	`NOM = NOM + SAL;`
	`}`
	`Cout<<"\nEl total de la nómina es:"<<NOM<<endl;`
	`return 0;`

Referencias

[1] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

[2] A. B. Sandoval, Introducción a la programación, Santiago de Cali: Ignacio Murgueitio, 2009.

Fundamentos de programación

Ciclos iterativos

WHILE

La estructura algorítmica repetitiva **while** permite repetir un conjunto de instrucciones. Sin embargo, el número de veces que se debe repetir depende de las proposiciones que contenga el ciclo. Cada vez que corresponde iniciar el ciclo se evalúa una condición, si ésta es verdadera (diferente es 0) así se continúa con la ejecución, de otra forma se detiene.

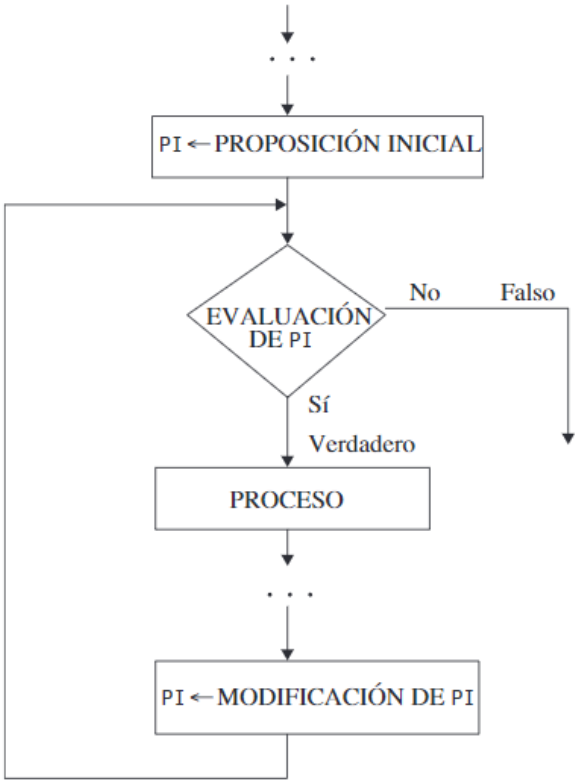


Ilustración 1, representación del ciclo while

Fundamentos de programación

Ciclos iterativos

Ejemplo

Construya un programa que, cuando reciba como datos los pagos efectuados en el último mes, permita obtener la suma de estos.

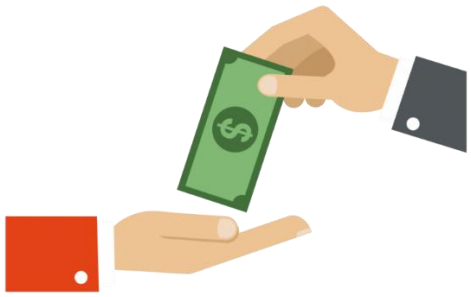


Ilustración 2, pagos.

Análisis

Revisando el problema se puede observar que es necesario solicitar todos los pagos realizados en el último mes, los cuales se guardaran en **PAG**. Además, calcular la suma total de todos los pagos, que se guardaran en **SPA**, (ver Tabla 1).

Para salir del ciclo, se tendrá la condición **PAG ≠ 0**, es decir, cuando se ingrese un pago de valor 0, el programa terminará y mostrara en pantalla el total de los pagos.

Tabla 1, Datos del problema.

Variables	Descripción
PAG	Variable de **tipo real** que representa los **pagos realizados el último mes.**
SPA	Es una variable de **tipo real** que **acumula** los pagos.

Algoritmo

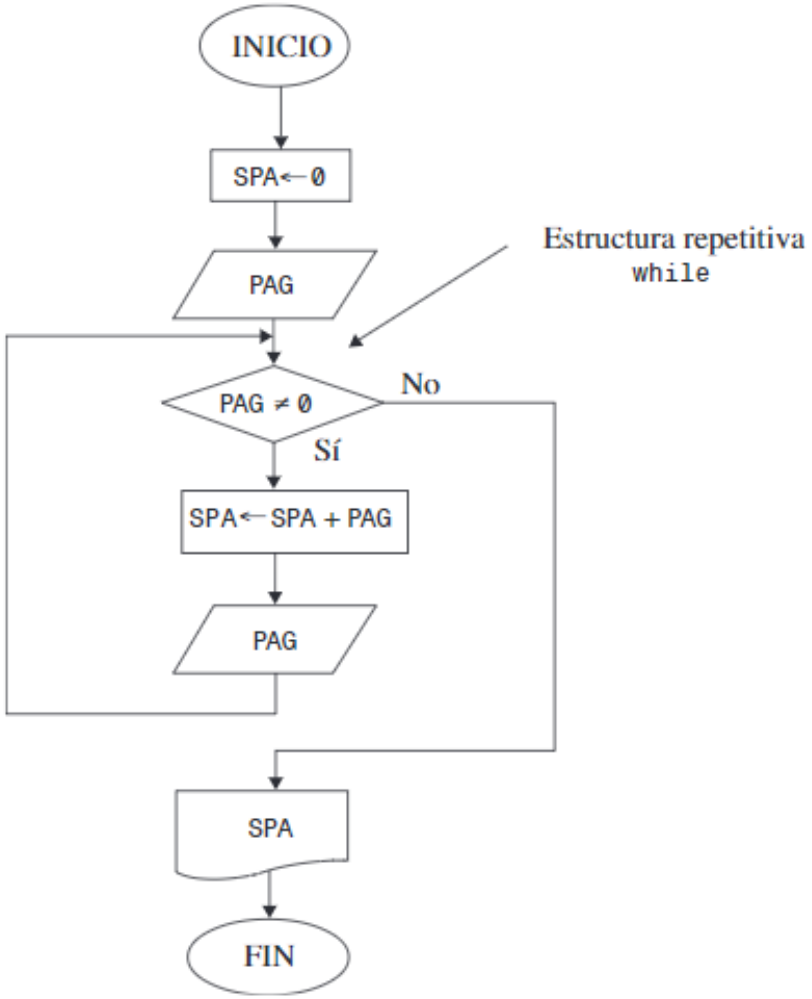


Ilustración 3,representación gráfica del algoritmo.

Fundamentos de programación

Ciclos iterativos

Pseudo-código

	Ejemplo: Calcula la suma de los pagos
Pseudocódigo	**Inicio** **Inicializar en 0 el acumulador de pagos SPA** **Mostrar** "Ingrese los pagos: " **Leer PAG** **Repetir Mientras PAG ≠ 0** **Calcular** el total de los pagos **SPA ← SPA + PAG** **Mostrar** "Ingrese los pagos: " **Leer PAG** **Fin del ciclo While** **Mostrar PAG** **Fin**

Fundamentos de programación

Ciclos iterativos

{/} Código

Lenguaje	Código

C	`float PAG, SPA; // Variable pagos y suma de pagos`
	`SPA=0; // Se inicializa en 0 la nómina`
	`printf("\Ingrese los últimos pagos del mes");`
	`scanf("%f", &PAG);`
	`while (PAG ≠ 0) // Ciclo while`
	`{`
	`SPA = SPA + PAG;`
	`printf("\Ingrese los últimos pagos del mes");`
	`scanf("%f", &PAG);`
	`}`
	`printf("\nEl total de los pagos de este mes es: %.2f", NOM); // Muestra la nómina total.`
	`return 0;`

Java	`Scanner sc=new Scanner(System.in);`
	`float SAL, NOM=0;`
	`while (PAG ≠ 0)`
	`{`
	`SPA = SPA + PAG;`
	`System.out.println("\n Ingrese los últimos pagos del mes ");`
	`PAG=sc.nextInt();`
	`}`
	`System.out.println("\nEl total de la nómina es: "+NOM);`

C++	`float PAG, SPA=0;`
	`cout<<"\Ingrese los últimos pagos del mes"<<endl;`
	`cin>>PAG;`
	`while (PAG ≠ 0)`
	`{`
	`SPA = SPA + PAG;`
	`cout<<"\Ingrese los últimos pagos del mes";`
	`cin>>PAG;`
	`}`
	`cout<<"\nEl total de los pagos de este mes es: "<<SPA<<endl;`
	`return 0;`

Referencias

[1] M. R. Vicente Benjumea, «Fundamentos de Programación con el Lenguaje de Programación C++,» 11 Marzo 2016. [En línea]. Available: https://openlibra.com/es/book/download/fundamentos-de-programacion-con-el-lenguaje-de-programacion-c. [Último acceso: 14 Junio 2019].

[2] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

Fundamentos de programación

Ciclos iterativos

DO UNTIL

Es una estructura que se encuentra prácticamente en cualquier lenguaje de programación de alto nivel, similar al repeat de los lenguajes Pascal y Fortran. A **Diferencia de las estructuras for y while**, en las cuales las condiciones evalúan al principio del ciclo, en ésta se valúan al final. Esto implica que el ciclo **se debe Ejecutar por lo menos una vez**.

La estructura es adecuada cuando no sabemos el número de veces que se debe repetir un ciclo, pero conocemos que se debe ejecutar por lo menos una vez. Es decir, se ejecuta el conjunto de instrucciones una vez, y luego cada vez que corresponde iniciar nuevamente el ciclo se evalúan las condiciones, siempre al final del conjunto de instrucciones. Si el resultado es verdadero (diferente de 0) se continúa con la ejecución, de otra forma se detiene.

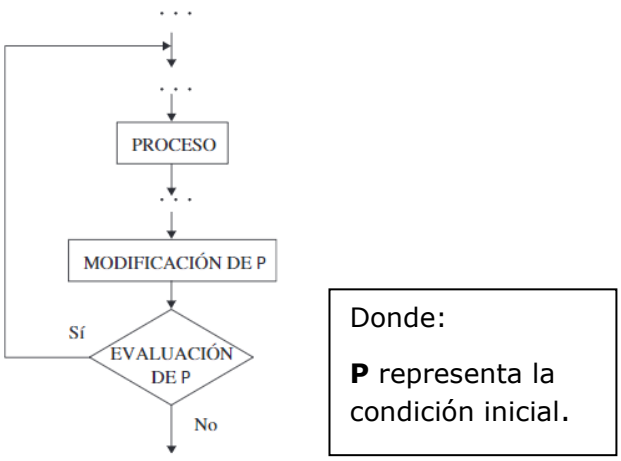


Ilustración 1, Diagrama de flujo de la estructura DO UNTIL

Fundamentos de programación

Ciclos iterativos

Ejemplo

En este ejemplo se resolverá el problema del tema anterior (estructura FOR), el cual es **construir** un programa que, cuando reciba como datos los **pagos efectuados en el último mes**, permita obtener la **suma** de estos.

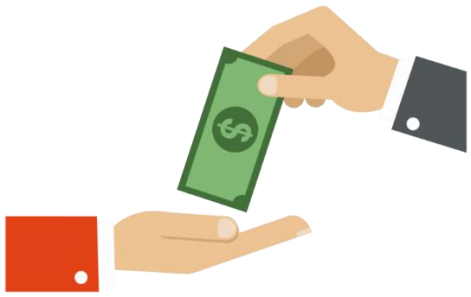


Ilustración 2, pagos.

Análisis

Revisando el problema se puede observar que es necesario solicitar todos los pagos realizados en el último mes, los cuales se guardaran en **PAG**. Además, calcular la suma total de todos los pagos, que se guardaran en **SPA**, (ver Tabla 1).

Para salir del ciclo, se tendrá la condición **PAG ≠ 0**, es decir, cuando se ingrese un pago de valor 0, el programa terminará y mostrara en pantalla el total de los pagos.

Tabla 1, Datos del problema.

Variables	Descripción
PAG	Variable de **tipo real** que representa los **pagos realizados el último mes.**
SPA	Es una variable de **tipo real** que **acumula** los pagos.

Algoritmo

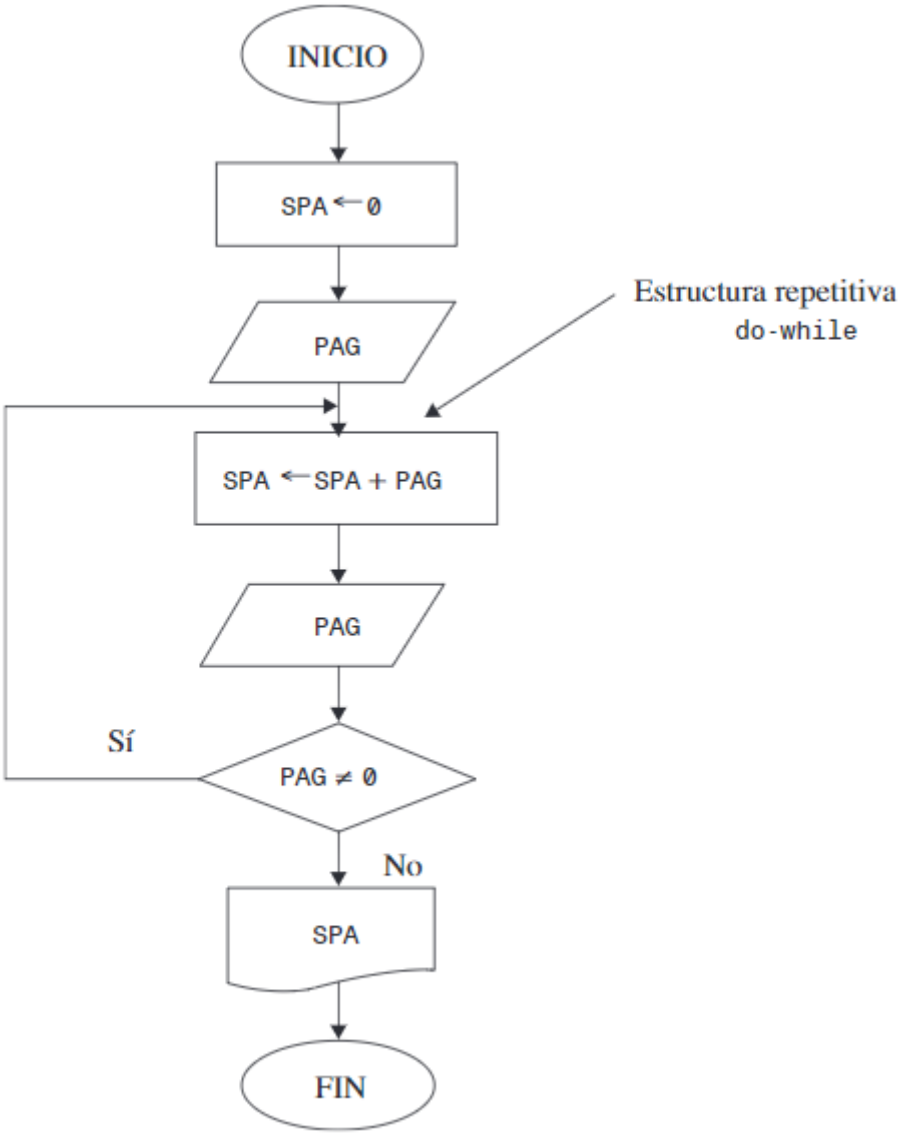


Ilustración 3, diagrama de flujo del problema.

Fundamentos de programación

Ciclos iterativos

Pseudo-código

	Ejemplo: Calcula el total de los pagos
Pseudocódigo	**Inicio** **Inicializar en 0 el acumulador de pagos SPA** **Mostrar** "Ingrese los pagos: " **Leer PAG** **hacer** **Calcular** el total de los pagos **SPA ← SPA + PAG** **Mostrar** "Ingrese los pagos: " **Leer PAG** **Mientras PAG ≠ 0** **Mostrar PAG** **Fin**

Fundamentos de programación

Ciclos iterativos

{/} Código

Lenguaje	Código

C	`float PAG, SPA; // Variable pagos y suma de pagos`
	`SPA=0; // Se inicializa en 0 la nómina`
	`printf("\Ingrese los últimos pagos del mes");`
	`scanf("%f", &PAG);`
	`do// Ciclo while`
	`{`
	`SPA = SPA + PAG;`
	`printf("\Ingrese los últimos pagos del mes");`
	`scanf("%f", &PAG);`
	`} while (PAG ≠ 0);`
	`printf("\nEl total de los pagos de este mes es: %.2f", NOM); // Muestra la nómina total.`
	`return 0;`

Java	`Scanner sc=new Scanner(System.in);`
	`float SAL, NOM=0;`
	`do`
	`{`
	`SPA = SPA + PAG;`
	`System.out.println("\n Ingrese los últimos pagos del mes ");`
	`PAG=sc.nextInt();`
	`} while (PAG ≠ 0);`
	`System.out.println("\nEl total de la nómina es: "+NOM);`

C++	`float PAG, SPA=0;`
	`cout<<"\Ingrese los últimos pagos del mes"<<endl;`
	`cin>>PAG;`
	`do`
	`{`
	`SPA = SPA + PAG;`
	`cout<<"\Ingrese los últimos pagos del mes";`
	`cin>>PAG;`
	`} while (PAG ≠ 0);`
	`cout<<"\nEl total de los pagos de este mes es: "<<SPA<<endl;`
	`return 0;`

Referencias

[1] L. J. Aguilar, Fundamentos de programación, Algoritmos, estructura de datos y objetos, México: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A .U., 2008.

[2] O. C. Battistutti, Fundamentos de programación. Piensa en C, Naucalpan de júarez: Pearson Educación de México, S.A de C.V., 2006.

[3] O. Cairó, Metología de la programación, Algoritmos, diagramas de flujo y programas, México : Alfaomega Grupo Editor, S.A de C.V., 2014.

Apéndice B
Evaluación

Evaluación: tipos de datos

Lea y conteste cuidadosamente las siguientes preguntas.

*Obligatorio

1. **Los siguientes datos numéricos se clasifican como:** *

165	1,345	-754	16,545	-17,985

Marca solo un óvalo.

- Negativos
- Carácter
- Enteros
- Reales

2. **¿Qué tipo de datos es "#$%"?** *

Marca solo un óvalo.

- Real
- Cadena de caracteres
- Carácter
- Entero

3. **Si se desea declarar una variable que almacene el valor 3.1416 en el lenguaje C, que almacene ¿Qué tipo de dato le corresponde?** *

Marca solo un óvalo.

- int
- float
- Real
- Char

4. **¿Qué tipo de dato es "Juan P."?** *

Marca solo un óvalo.

- Carácter
- Cadena de caracteres
- Entero
- Real

5. **¿Cuál de las siguientes respuestas se denomina como un tipo de dato estructurado?** *
Marca solo un óvalo.

- Reales
- Cadena de caracteres
- Positivos
- Datos simples

6. **Los siguientes datos numéricos se clasifican como:** *

8.5	165.2	-46.821	164.8	-16.3

Marca solo un óvalo.

- Reales
- Enteros
- Negativos
- Carácter

7. **Los datos _____ se caracterizan por el hecho de que con un nombre se hace referencia a un grupo de casillas de memoria.** *

_____

8. **Los tipos de datos simples ocupan solo una casilla de memoria.** *
Marca solo un óvalo.

- Falso
- Cierto

9. **¿Qué tipo de dato es "#"?** *
Marca solo un óvalo.

- Real
- Cadena de caracteres
- Entero
- Carácter

10. **Selecciona las 2 formas en que se pueden clasificar los que procesa una computadora.** *
Selecciona todas las opciones que correspondan.

- [ ] Simples
- [ ] Estructurados
- [ ] Positivos
- [ ] Reales

Evaluación: Constantes y variables

Lea y conteste cuidadosamente las siguientes preguntas.

*Obligatorio

1. **Es un dato cuyo valor siempre es el mismo durante la ejecución de un programa.** *
   Marca solo un óvalo.
   - Datos numéricos
   - Constante
   - Variable
   - Tipo de datos

2. **Es un dato cuyo valor puede cambiar durante la ejecución de un programa.** *
   Marca solo un óvalo.
   - Variable
   - Datos numéricos
   - Tipos de datos
   - Constante

3. **Son consideras como una de las propiedades más potentes en programación.** *
   Marca solo un óvalo.
   - Constantes
   - Variables
   - Tipos de datos
   - Datos númericos

4. **¿A que lenguaje de programación pertenece la siguiente sintaxis para declarar una constante? #define<nombre_constante>=<valor>;** *
   Marca solo un óvalo.
   - C
   - C++
   - PHP
   - Java

5. **¿A que lenguaje de programación pertenece la siguiente sintaxis para declarar una constante? staticfinal<tipo_de_dato><nombre_constante>=<valor>;** *

Marca solo un óvalo.

- C
- Java
- PHP
- C++

6. **¿A que lenguaje de programación pertenece la siguiente sintaxis para declarar una variable? <tipo_de_dato> <nombre_variable>** *

Marca solo un óvalo.

- C++
- Java
- C
- PHP

7. **Seleccione la opción que corresponda a una variable en el lenguaje C** *

Marca solo un óvalo.

- int x;
- Números reales
- entero x;
- #define Pi=3.1416;

8. **Seleccione la opción que corresponda a una constante en el lenguaje C** *

Marca solo un óvalo.

- Números reales
- entero x;
- int x;
- #define Pi=3.1416;

9. **¿Cuál de estas variables esta bien definidas en lenguaje C?** *

Marca solo un óvalo.

- int numero_de_hijos;
- int #_de_hijos;
- int numero de hijos;
- Ninguna es correcta

10. **Si se desea declarar una variable para almacenar un número entero y que, inicialmente, contenga el valor 35, se debe escribir:** *

Marca solo un óvalo.

- numero=35;
- Ninguna es correcta.
- int numero 35;
- int numero=35;

Evaluación: Operadores y expreciones

Lea y conteste cuidadosamente las siguientes preguntas.

*Obligatorio

1. **Se definen como operaciones que dan como resultado un valor a excepción de expresiones void.** *
   Marca solo un óvalo.
   - Variables
   - Constantes
   - Expreciones
   - Operadores

2. **Se definen como símbolos que expresan operaciones que aplican a operandos.** *
   Marca solo un óvalo.
   - Operadores
   - Variables
   - Constantes
   - Expreciones

3. **Son conocidos como operadores de incremento.** *
   Marca solo un óvalo.
   - x
   - + +
   - - -
   - /

4. **Selecciona las respuestas que son clasificadas como operadores aritméticos.** *
   Selecciona todas las opciones que correspondan.
   - /
   - -
   - +
   - #

5. **Seleccione el valor de "y" según la siguiente expresión.** *

```
x = 6;

y = --x;
```

Marca solo un óvalo.

- x=5
- x=6
- y=5
- y=6

6. **Son conocidos como operadores de decremento.** *

Marca solo un óvalo.

- + +
- /
- - -
- x

7. **Seleccione el valor de "y" según la siguiente expresión.** *

```
x = 7;

y = ++x;
```

Marca solo un óvalo.

- y=7
- y=8
- x=7
- x=8

Funciones internas

Lea y conteste cuidadosamente las siguientes preguntas.

*Obligatorio

1. **¿Qué calcula la función round(x)?** *
   Marca solo un óvalo.
   - Coseno de x
   - Seno de x
   - Valor absoluto de x
   - Redondeo de x

2. **¿Cuál es el resultado de la operación round (89.65)?** *
   Marca solo un óvalo.
   - 80
   - 89
   - 90
   - 91

3. **¿Qué calcula la función abs(x)?** *
   Marca solo un óvalo.
   - Coseno de x
   - Seno de x
   - Valor absoluto de x
   - Redondeo de x

4. **Las funciones internas pueden ahorran tiempo y facilitar algunos calculos al programador.** *
   Marca solo un óvalo.
   - Verdadero
   - Falso

5. **Los lengujes de programación cuentan con funciones de operaciones aritméticas incorporadas.** *
   Marca solo un óvalo.
   - Verdadero
   - Falso

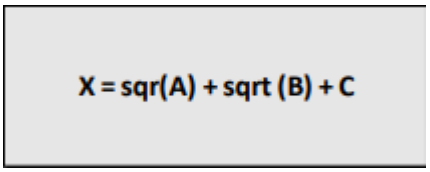
6. **¿Cuál es el resultado de la operación sqrt (25)?** *
Marca solo un óvalo.

- x
- 5
- 25
- 0

7. **¿Qué calcula la función cos(x)?** *
Marca solo un óvalo.

- Coseno de x
- Seno de x
- Valor absoluto de x
- Redondeo de x

8. **Tomando como datos A=5, B=25 y C=10, seleccione el resultado correcto.** *



Marca solo un óvalo.

- 40
- 25
- 45
- 30

9. **¿Cuál es el resultado de la operación abs (-250)?** *
Marca solo un óvalo.

- -250
- 250
- 25
- 0

Lenguajes de programación

Lea y conteste cuidadosamente las siguientes preguntas.

*Obligatorio

1. **Se utilizan para escribir programas de computadora.** *
   Marca solo un óvalo.
   - Funciones código fuente
   - Algoritmos
   - Ninguna de las anteriores
   - Lenguajes de programación

2. **Un programa se escribe en un lenguaje de programación y las operaciones que conducen a expresar un algoritmo en forma de programa se llama:** *
   Marca solo un óvalo.
   - Funciones internas
   - Tipos de datos
   - Programación
   - Ninguna de las anteriores

3. **El algoritmo escrito en un lenguaje de programación se denomina:** *
   Marca solo un óvalo.
   - Funciones código fuente
   - Algoritmos
   - Lenguajes de programación
   - Código fuente

4. **Es el proceso de traducir un algoritmo en pseudocódigo a un lenguaje de programación se le denomina:** *
   Marca solo un óvalo.
   - Algoritmos
   - Código fuente
   - Codificación
   - Lenguajes de programación

5. **Los lenguajes de programación NO son necesarios para escribir programas.** *
   Marca solo un óvalo.
   - Falso
   - Verdadero

6. **Son lenguajes de programación denominados de alto nivel.** *
Selecciona todas las opciones que correspondan.

- [ ] Ensamblador
- [ ] C++
- [ ] C
- [ ] Java

7. **Se le denomina el lenguaje nativo de una computadora.** *
Marca solo un óvalo.

- ◯ Ensamblador
- ◯ Lenguajes de programación
- ◯ Binario
- ◯ Lenguaje máquina

Algoritmo

Lea y conteste cuidadosamente las siguientes preguntas.

*Obligatorio

1. **A partir de los siguientes datos, desarrolla un algoritmo para llenar un vaso de agua: un vaso, un grifo.** *

a) Abrir el grifo

b) Esperar que el vaso se llene

c) Cerrar el grifo

d) Retirar el vaso

e) Tomar el vaso

f) Colocar el vaso bajo el grifo

Marca solo un óvalo.

- ◯ c,d,f,a,b,e
- ◯ b,f,c,a,d,e
- ◯ a,b,c,d,e,f
- ◯ e,a,f,b,d,c

2. **Un algoritmo es:** *

Marca solo un óvalo.

- ◯ Un esquema para resolver cierto tipo de problema
- ◯ Un lenguaje humano que se traduce en un programa que se ejecuta en un computador
- ◯ Todas las anteriores
- ◯ Ninguna de las anteriores

3. **No es necesario que un algoritmo sea preciso, es decir que tenga un orden estricto y expresiones precisas.** *

Marca solo un óvalo.

- ◯ Verdadero
- ◯ Falso

4. **Teniendo como datos los siguientes ingredientes, ordene la receta para preparar helado de fresa.** *

Ingredientes	▪ 500 gr. de fresas. ▪ 100 gr. de azúcar. ▪ 25 ml. de azúcar invertido. ▪ 200 ml. de leche entera. ▪ 225 ml. de nata para montar. ▪ 1 limón.
Preparación del helado de fresa	a) Colocar la mezcla en el refrigerador y sacarla cuando transcurra una hora. b) Añadir la leche, unas gotitas de limón y el azúcar invertido, triturar todo con la batidora. Colar para eliminar las semillas. c) Cortar las fresas en trocitos. Y mezclarlas con azúcar. Dejar que repose durante hora y media. d) Poner la nata hasta que se formen puntas firmes. e) Añadir el molido de fresas lentamente, moviendo ampliamente, pero con cuidado sin que se baje.

Marca solo un óvalo.

- ◯ e,b,c,d,a
- ◯ c,b,d,e,a
- ◯ a,b,c,d,e
- ◯ b,c,a,d,e

5. **Un algoritmo se define como una serie de pasos, procedimientos o acciones que nos permiten alcanzar algún resultado o resolver algún problema. Esta definición es:** *

Marca solo un óvalo.

- ◯ Falso
- ◯ Verdadero

Diagrama de flujo

Lea y conteste cuidadosamente las siguientes preguntas.

*Obligatorio

1. **Un diagrama de flujo se define una esquematización visual de un algoritmo.** *
   Marca solo un óvalo.
   - Falso
   - Verdadero

2. **Un diagrama de flujo se define como una series de pasos, procedimientos o acciones que nos permiten alcanzar algún resultado o resolver algún problema.** *
   Marca solo un óvalo.
   - Falso
   - Verdadero

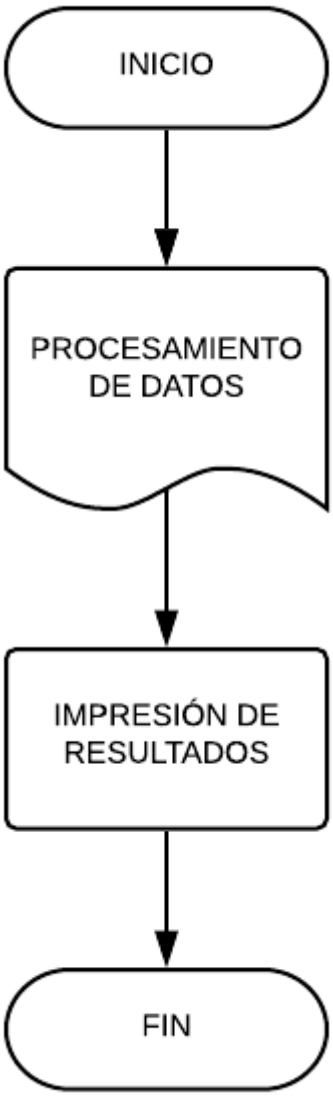
3. **Qué indica el siguiente símbolo?** *

   Marca solo un óvalo.
   - Toma de decisión
   - Dirección de flujo
   - inicio o fin de un proceso
   - Actividad que debe ser ejecutada

4. **Qué indica el siguiente símbolo?** *

   Marca solo un óvalo.
   - Actividad que debe ser ejecutada
   - Toma de decisión
   - inicio o fin de un proceso
   - Dirección de flujo

5. **Señala el error en el siguiente diagrama de flujo.** *



Selecciona todas las opciones que correspondan.

- [ ] El flujo de los datos es incorrecto
- [ ] Se intenta realizar el procesamiento de datos en el símbolo de imprimir
- [ ] Se intenta imprimir con el símbolo de procesamiento de datos
- [ ] El símbolo de inicio y fin es incorrecto

6. **Qué indica el siguiente símbolo?** *

Marca solo un óvalo.

- ◯ inicio o fin de un proceso
- ◯ Dirección de flujo
- ◯ Actividad que debe ser ejecutada
- ◯ Toma de decisión

7. **Qué indica el siguiente símbolo?** *

Marca solo un óvalo.

- ◯ Dirección de flujo
- ◯ Toma de decisión
- ◯ Actividad que debe ser ejecutada
- ◯ inicio o fin de un proceso

Pseudocódigo

Lea y conteste cuidadosamente las siguientes preguntas.

*Obligatorio

1. **El pseudocódigo puede facilitar de gran manera la conversión de un algoritmo a codificación.** *

   Marca solo un óvalo.

   - Falso
   - Verdadero

2. **El pseudocódigo puede ser ejecutado por una computadora.** *

   Marca solo un óvalo.

   - Verdadero
   - Falso

3. **Cuáles de las siguientes ventajas pertenecen al pseudocódigo?** *

   Selecciona todas las opciones que correspondan.

   - [ ] Ayuda a concentrar la lógica y estructuras de control
   - [ ] Es complicado de traducir a lenguajes de programación
   - [ ] Se puede ejecutar en una computadora
   - [ ] No es necesario preocuparse por reglas de un lenguaje de programación

4. **Nació como un lenguaje similar al inglés y era un medio de representar básicamente las estructuras de control de programación estructurada.** *

   Marca solo un óvalo.

   - Pseudocódigo
   - Código
   - C++
   - Lenguaje de programación

5. **En el siguiente pseudocódigo se calcula el salario neto de un trabajador. Encuentre el error.** *

```
start

   read nombre

   salario <- horas * precio

   tasas <- 0,25 * salario

   salario_neto <- salario -tasas

   write nombre, salario, tasas, salario

end
```

Marca solo un óvalo.

- La tasa es calculada mal
- El salario es calculado mal
- No se piden algunos datos
- Se imprimen demasiados datos

6. **La escritura de pseudocódigo exige normalmente la indentación (sangría en el margen izquierdo).** *

Marca solo un óvalo.

- Veradero
- Falso

7. **Cuál de las siguientes palabras reservadas en ingles no pertenecen a la representación de una acción en un pseudocódigo?** *

- **start**
- **end**
- **stop**
- **if-then-else**
- **Int**
- **while-end**
- **Float**
- **repeat-until**

Marca solo un óvalo.

- int y float
- start y end
- float y repeat-until
- int y repeat-until

Google Forms

Apéndice C
Encuesta

Encuesta sobre el proyecto Prototipo de un sistema de apoyo para aprender a programar

Preguntas generales

Carrera: _____

Edad: _____ :Sexo _____

Diseño del entorno web

1.- ¿Crees que el diseño de la página de bienvenida (estructura, usabilidad, entre otros), es adecuado?

☐Sí.
☐No.

2.- ¿Consideras que la forma de presentar la información es adecuada?

☐Sí.
☐No.

3.- ¿Crees que el diseño de la página de interfaz de usuario, (estructura, usabilidad, entre otros), es adecuado?

☐Sí.
☐No.

4.- ¿Consideras que la página presenta un exceso de elementos en pantalla?

☐Sí.
☐No.

5.- ¿La redacción utilizada para mostrar la información es…?

☐ Muy buena. ☐ Buena. ☐ Regular. ☐ Mala. ☐ Muy mala.

6.- ¿Opinas que el tamaño y tipo de letra utilizado para mostrar el contenido es...?

☐ Muy buena. ☐ Buena. ☐ Regular. ☐ Mala. ☐ Muy mala.

7.- ¿Piensas que la aplicación web fue fácil de usar?

☐Sí.
☐No.

8.- ¿Piensas que el acceso al sistema es complicado (inicio de sesión requerido)?

☐Sí.
☐No.

Evaluación del contenido

9.- ¿Consideras qué la programación es importante?

☐Sí.
☐No.

10.- ¿Te parece interesante el objetivo del proyecto?

☐Sí.
☐No.

11.- ¿El contenido fue difícil de comprender?

☐Sí.
☐No.

12.- ¿Cómo consideras él contenido didáctico (temas)?

☐ Muy buena. ☐ Buena. ☐ Regular. ☐ Mala. ☐ Muy mala.

13.- ¿Con que frecuencia utilizarías esta herramienta como medio de aprendizaje?

☐Siempre.
☐Regular.
☐A veces
☐Nunca.

14.- ¿Recomendarías esta herramienta a tus amigos?

☐Si.
☐No.

15.- ¿Te gustaría que se implementará más contenido en el sistema (problemas)?

☐Sí.
☐No.