MEDAC
Instituto Oficial de Formación Profesional

PROGRAMACIÓN

Introducción a la programación

1

ÍNDICE

OBJETIVOS

Conocer qué es un lenguaje de programación.

Conocer qué es un paradigma en programación.

Conocer qué tipos de lenguajes de programación existen.

Conocer e identificar las diferentes fases del ciclo de desarrollo de software.

Conocer los diferentes entornos de desarrollo que existen en el mercado.

Instalación de un entorno de desarrollo integrado.

/ 1. Introducción y contextualización práctica

En este tema, hablaremos sobre una serie de conceptos básicos a la hora de comenzar nuestra aventura en el mundo de la programación.

Muchos de estos conceptos los iremos viendo con mucha más profundidad en los siguientes temas, así que no te preocupes; tenemos mucho tiempo por delante.

Escucha el siguiente audio en el que planteamos el caso práctico que iremos resolviendo a lo largo de esta unidad.

Fig. 1. En la vida diaria programamos sin darnos ni cuenta

Audio Intro. "El comienzo de la aventura".
https://bit.ly/31qPrxq

/ 2. Los lenguajes de programación

Cuando dos personas se comunican utilizan un lenguaje o idioma para poder entenderse, tales como el español, el inglés, portugués, alemán... **Para que la comunicación sea posible, es necesario que las dos personas implicadas hablen el mismo idioma, o en su defecto, que haya una tercera** persona que **hable los dos idiomas** y pueda hacer de traductor.

Las personas se comunican entre sí utilizando el lenguaje natural, sin embargo, este tipo de lenguaje tiene ambigüedades, por lo que para programar un ordenador no sirve. Es por ello por lo que se hace necesario disponer de lenguajes más específicos, donde cada instrucción solo tenga una única interpretación y no haya lugar a errores.

En el caso de los lenguajes de programación, en lugar de tener a dos personas comunicándose tenemos a una persona (el programador) a un ordenador, y un lenguaje de programación con el que el primero le indica las instrucciones a ejecutar al segundo.

Un lenguaje de programación se puede definir como un **lenguaje formal, con sus reglas gramaticales bien definidas**, que les proporciona a las personas la capacidad de escribir una serie de algoritmos con instrucciones que la máquina es capaz de interpretar y que son capaces de controlar el comportamiento físico y/o lógico del ordenador. **Llamamos programa a la totalidad de las instrucciones y algoritmos escritos mediante un lenguaje de programación**.

Existen multitud de clasificaciones posibles para los lenguajes de programación: compilados, interpretador, de software libre, privativos...

Algunos lenguajes de programación son: C, C++, C#, Java, Python, Perl, PHP, Kotlin, Swift...

Como es lógico, no es humanamente posible aprender a programar en todos los lenguajes existentes, sino que **el paso lógico es aprender las bases en un lenguaje de programación inicial** y, una vez que esas bases estén bien cimentadas, el cambiar a otro lenguaje sea una tarea relativamente sencilla.

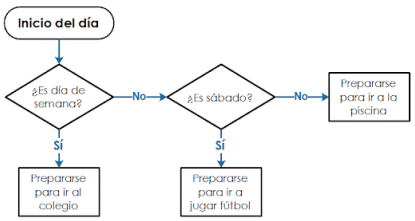

Fig. 2. Algoritmo escrito en un lenguaje de programación.

/ 3. Paradigmas de programación

Dentro del mundo de la programación, los paradigmas indican la forma en la que deben estructurarse y gestionarse todas las tareas que ejecutan los programas.

Existen diferentes tipos de paradigmas actualmente y, todos son diferentes los unos de los otros en sus estructuras y en las formas de gestionar las tareas que deberá realizar el programa que se esté ejecutando.

Hoy en día, casi la totalidad de los lenguajes de programación que existen trabajan sobre el paradigma de la orientación a objetos, lo que se conoce como un lenguaje orientado a objetos, aunque está empezando a imponerse en ciertos lenguajes de programación como Swift y Kotlin el paradigma de programación orientada a eventos.

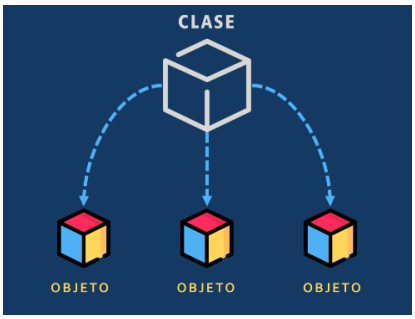

Fig.3. Cada paradigma tiene su propia filosofía.

Algunos de los paradigmas que se han utilizado o se utilizan en programación son:

- **Programación estructurada:** Este es el paradigma más simple. En este las instrucciones se ejecutan en el orden en el que se escriben y siempre se ejecutarán en el orden.

- **Programación funcional:** Es una mejora de la programación estructurada en la que el código se agrupa en bloques llamados funciones, que pueden ser invocadas tantas veces como sea necesario, lo cual permite reutilizar el código sin tener que volverlo a escribir. Reduce considerablemente la cantidad de líneas de código de los programas, es decir, aumenta la reusabilidad de código.

- **Programación orientada a objetos:** Este paradigma agregó el concepto de clase y objeto, así como las relaciones de herencia y el polimorfismo. Permite agrupar los datos en plantillas denominadas clases, de las que luego se crearán copias (los objetos) con los valores concretos. Ayuda también a la reutilización de código de los programas.

- **Programación orientada a eventos:** En este paradigma, la ejecución de código viene dada por los eventos que ocurren en el sistema.

/ 4. Caso práctico 1: "¿Qué primer lenguaje de programación elegir?"

Planteamiento. Pilar y José están investigando qué lenguaje de programación les convendría más utilizar para asentar bien sus bases. Como no tienen mucha idea se encuentran con que existen una infinidad de lenguajes, y todos distintos unos de otros. Unos necesitan una cantidad enorme de líneas de código para hacer un programa simple, otros están enfocados a las matemáticas (y eso no les gusta para nada), otros permiten con una línea hacer operaciones realmente poderosas. Otros son privativos y otros son libres, aunque no entienden muy bien que es eso.

Nudo. ¿Qué piensas sobre ello? ¿Crees que lo mejor para empezar a aprender a programar es un lenguaje de programación matemático? ¿Sería mejor un lenguaje privativo o uno libre?

Desenlace. Lo más adecuado para iniciarse en el mundo de la programación no es un lenguaje orientado a las matemáticas, ya que para entenderlos hace falta una base matemática muy dura, en cambio, un lenguaje de programación orientado a objetos puede darnos los conceptos básicos que vamos a necesitar, ya que los lenguajes orientados a objetos tienen la base estructural además de la orientación a objetos que tan básica va a ser a la hora de entrar en el mundo del desarrollo de software.

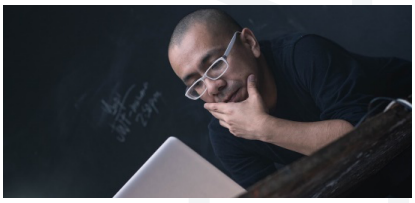

Fig. 4. Investiga bien antes de empezar.

Como último apunte, lo mejor es centrarse en un lenguaje no privativo, o libre, ya que la documentación que se encontrará disponible será mucho mayor que la de un lenguaje privativo. Éste pertenecerá a una empresa, y no dará tanta documentación al público.

Algunos buenos lenguajes de programación para iniciarse pueden ser C++, Java o Python.

/ 5. Datos, algoritmos y programas

En programación, hay tres conceptos fundamentales que son:

- El concepto de dato.
- El concepto de algoritmo.
- El concepto de programa.

Datos

Los datos son los elementos que van a proporcionar la entrada de información de los programas, es decir, en los datos se va a guardar información que se podrá almacenar y procesar, por ejemplo, un nombre, edad o dirección.

Hay diferentes tipos de datos. Los más utilizados en programación son:

- Datos de tipo numérico
  - Datos de tipo numérico enteros.
  - Datos de tipo numérico reales.
- Datos de tipo carácter
  - Datos que representan un carácter.
  - Datos que representan una cadena de caracteres.
- Datos de tipo booleano

Algoritmo

Un algoritmo es una secuencia de instrucciones bien implementadas y ordenadas que resuelve una tarea en concreto.

Como programadores, codificaremos algoritmos en el lenguaje en que estemos trabajando, de forma que indiquemos al ordenador cómo debe resolver cualquier el problema que se nos plantee.

Programa

Llamamos programa a la totalidad de las instrucciones y algoritmos escritos mediante un lenguaje de programación.

Audio 1. "Tipos de datos".
https://bit.ly/3dE3NNr

5.1. Datos de tipo numérico

Estos datos almacenarán información numérica y podrán tener signo negativo o positivo, es decir, podrán almacenar números tales como: -47, 963, 85,4, -96,745, 10284... Con este tipo de datos podremos representar cualquier cosa que sea un número: la edad de una persona, el número de alumnos de un aula, el sueldo de un trabajador, etc.

Una cosa muy importante a tener en cuenta es que jamás podremos representar infinitos números, esto se debe a que vamos a utilizar un ordenador para poder representarlos, el cual va a tener recursos limitados (memoria RAM limitada, disco duro limitado...) y esto nos va a condicionar la representación de la información, como veremos en los siguientes temas.

Fig. 5. No se pueden representar números infinitos ya que el espacio es limitado.

Podemos clasificar este tipo de datos de la siguiente forma:

- **Con signo positivo:** Estos serán los números positivos, concretamente, los que van del 0 al "infinito". Estos se pueden clasificar en:
  - Números enteros positivos: Estos números son los que no tienen una parte decimal, o lo que es lo mismo, sólo tienen parte entera. Ejemplos de ellos son: 0, 1, 2, 3, ..., 425, 426, ..., 85.412...
  - Números reales positivos: Estos números son los que poseen tanto parte entera como parte decimal. Ejemplos de ellos son: 1,54, 23,67, 67,12, 2.345,76...
- **Con signo negativo:** Estos serán los números negativos, o sea, los que van del -1 al "-infinito". Estos se pueden clasificar en:
  - Números enteros negativos: Estos números negativos son los que no tienen una parte decimal. Sólo tienen parte entera. Ejemplos de ellos son: -1, -2, -3, ..., -425, -426, ...,
  - Números reales negativos: Estos números negativos son los que poseen tanto parte entera como parte decimal. Ejemplos de ellos son: -1,0, -1,01, -1,02, -1,03, ..., -6,325...

5.2. Datos de tipo carácter

Estos datos almacenarán información alfanumérica. Estos pueden almacenar los siguientes caracteres: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z. Además, podrán almacenar caracteres especiales, como, por ejemplo: Ñ, ñ, ¡, ", #, $, _, espacios en blanco, las letras acentuadas tanto en minúscula como en mayúscula, letras de otros alfabetos distintos al latino (como el ruso), etcétera.

Estos datos pueden clasificarse de la siguiente forma:

- **Datos de tipo carácter que almacenan un único carácter:** Estos datos, como su descripción indica, podrán almacenar un único carácter. Ejemplos de ellos pueden ser: a, b, Y, ¿, 8...
- **Datos de tipo cadena de caracteres:** Estos datos pueden almacenar varios caracteres, es decir, vamos a representar con ellos lo que se conoce como una cadena de caracteres. Estos datos llevan asociado un tamaño, que representará el número de caracteres que posee la cadena. Ejemplos de ellos pueden ser: 'hola' (con un tamaño de 4), 'murciélago' (con un tamaño de 10), 'En un lugar de la mancha' (con un tamaño de 24). Asimismo, podemos tener cadenas de un solo carácter: 'A' (tamaño 1) e incluso vacías: '' (tamaño 0). Resumiendo, un carácter solo almacena un único elemento, mientras que una cadena puede tener cualquier número de ellos.

5.3. Datos de tipo booleano

Este tipo de datos puede ser el más complicado de entender. **Los datos de tipo booleano vienen heredados del Álgebra de Boole (la que les da su nombre)** y pueden representar únicamente **dos tipos de valores: verdadero y falso** (también conocidos como **0 y 1**).

Este tipo de datos no se utilizará para realizar operaciones, sino que su utilidad está en la evaluación de expresiones lógicas para el tratamiento de condiciones y comparaciones. Estas expresiones van a permitir a los programas tomar decisiones a la hora de escoger un camino u otro según si se cumplen unas condiciones u otras. Con ellos, podremos saber si un número es mayor que otro, si es menor, mayor o igual, menor o igual o igual. También **podremos hacer esas comprobaciones con datos de tipo carácter**, tanto de un único carácter como con cadenas de caracteres de cualquier longitud.

Cada vez que necesitemos hacer algo que pueda ser verdad o mentira (verdadero o falso) **tendremos que echar mano de los datos de tipo booleano.**

Más adelante, veremos que con este tipo de datos podremos realizar operaciones que no podremos realizar con los de tipo numérico o carácter.

Fig. 6. El camino que siguen los programas no tiene por qué ser siempre el mismo.

/ 6. Ciclo de vida del software

El ciclo de vida del *software* **describe el proceso que se ha de seguir para la creación o desarrollo de un producto software**, ya sea una aplicación de escritorio, una página web o una app móvil.

El ciclo de vida del software se divide en varias fases. Hay que ir pasándolas para poder validar el buen desarrollo de nuestro producto software. Es importante ir cumpliendo con todas las fases de desarrollo ya que un error en alguna de ellas seguramente nos hará retroceder a la anterior, y así constantemente.

La gran ventaja de seguir el ciclo de vida correctamente es que nos va a permitir detectar los errores que haya en nuestro desarrollo mucho más fácilmente.

El ciclo de vida de *software* se compone de las siguientes fases:

1. Definición de los objetivos del proyecto.
2. Análisis de los requisitos necesarios.
3. Diseño general del producto software.
4. Diseño de las partes del producto software.
5. Programación o implementación del producto.
6. Pruebas.
7. Integración de los módulos del software.
8. Pruebas de validación.
9. Documentación del producto.

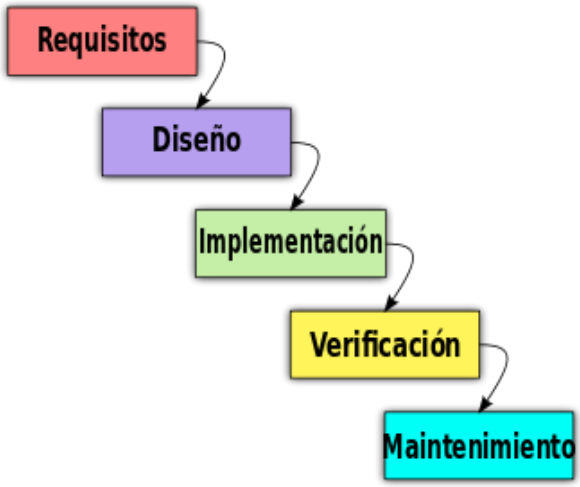

Fig. 7. Ciclo de vida del software en cascada.

/ 7. Entornos de Desarrollo Integrado

Cuando se empezó a desarrollar *software*, había que utilizar muchos programas para poder hacerlo, tales como un compilador, un enlazador, el depurador...

Los Entornos de Desarrollo Integrado, o IDE por sus siglas en inglés, son un programa que ya integra todas las herramientas necesarias para poder desarrollar software en ellos mismos, facilitando por tanto esta tarea, ya que en el mismo IDE tendremos todo lo necesario.

Los IDE incorporan los siguientes componentes:

- **Editores de texto:** Para poder escribir el código fuente.
- **Preprocesadores:** Este componente preprocesa el código fuente para poder facilitarle el trabajo al compilador.
- **Ensambladores:** Estos son los encargados de generar código ensamblador si es necesario.
- **Linkers:** Son los encargados de enlazar las bibliotecas externas utilizadas.
- **Depuradores:** Se encargan de la depuración del código fuente para la detección y eliminación de errores en tiempo de ejecución.

Hay muchos tipos de IDE, podemos clasificarlos en:

- IDE libres.
- IDE privativos.
- IDE multiplataforma.
- IDE monoplataforma.

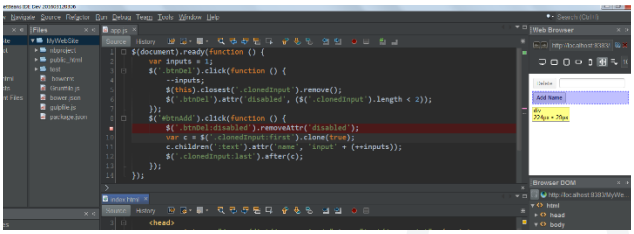

Fig. 8. Entorno de Desarrollo Integrado.

7.1. Instalación de NetBeans 10

Un paso importante a la hora de elegir el Entorno de Desarrollo Integrado que queramos usar es saber en qué lenguaje de programación vamos a programar, en nuestro caso, vamos a utilizar Java.

Dentro de la gran cantidad de Entornos de Desarrollo Integrado que existen actualmente vamos a elegir NetBeans en su última versión, por su fácil adaptación para programar en Java.

El primer paso será la instalación de NetBeans 10, para ello:

1. Primero debemos instalar el JDK[1] de Java, que será lo que nos va a permitir poder utilizar este lenguaje de programación. Vamos a instalar el JDK en su versión 8, que nos podremos descargar de la propia página de Oracle:

https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html

2. Una vez descargado e instalado el JDK deberemos instalar NetBeans, para ello lo descargamos de su propia página:

https://netbeans.apache.org/download/index.html

3. Una vez descargado, lo descomprimimos y dentro de la carpeta bin estará el ejecutable del IDE, el fichero llamado netbeans64.exe.

Vídeo 1. "Instalación de NetBeans".
https://bit.ly/3g8SGxv

1. El JDK es la herramienta de desarrollo que incluye el compilador de JAVA, entre otras muchas cosas.

/ 8. Caso práctico 2: "¿Por dónde empezamos?"

Planteamiento: Pilar y José van a empezar a desarrollar su primer programa software y están decidiendo sobre en qué etapa del ciclo de vida deberían empezar. José está intentando convencer a Pilar de que la mejor opción y más rápida es la de empezar directamente por la etapa de codificación, y que las demás etapas ya las irán haciendo sobre la marcha. Pilar insiste en que lo mejor es empezar por la etapa de definición de objetivos y análisis de requisitos, ya que para eso es la primera de las etapas.

Nudo: ¿Qué piensas al respecto? ¿Crees que José lleva razón y lo más fácil es empezar directamente programando y lo demás ir haciéndolo sobre la marcha? ¿O piensas que quien tiene razón es Pilar y lo más lógico es empezar por la primera etapa del ciclo de vida, la de definición de objetivos y análisis de requisitos?

Desenlace: Cuando vayamos a empezar un proyecto de software, lo más conveniente es seguir al pie de la letra las etapas del ciclo de vida del software. No es recomendable empezar programando y haciendo el resto de etapas sobre la marcha, porque lo más normal es que nos equivoquemos y cambiemos cosas muchas veces, ya que a la primera no vamos a acertar, y un fallo o cambio en una etapa del ciclo de vida, por muy pequeño que sea, hará que tengamos que cambiar todas las etapas anteriores. Es por esto que al ciclo de vida del software se le denomine "en cascada".

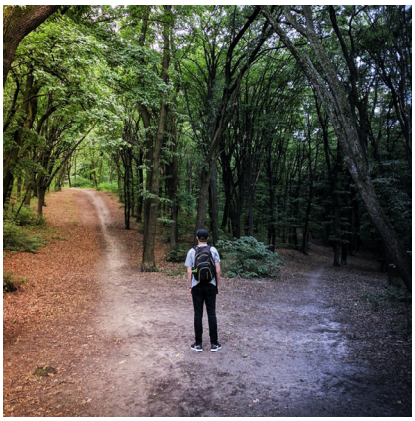

Fig. 9. La importancia de una buena decisión.

/ 9. Creación de un proyecto en NetBeans

Para crear un proyecto en NetBeans pulsaremos en Archivo -> Proyecto nuevo.

En Categorías, seleccionamos Java y en Proyectos seleccionamos Java with Ant -> Java Application y pulsamos Siguiente.

En la siguiente pantalla, elegiremos el nombre del proyecto. En Ubicación del proyecto, seleccionaremos dónde queremos que se guarde nuestro proyecto para cambiar la ubicación pulsamos en el botón Examinar. La opción carpeta del proyecto será la carpeta donde se almacenará nuestro proyecto nuevo (esta no la debemos cambiar). Por último, las demás opciones las dejaremos como están y pulsamos Terminar.

Con esto tendremos un proyecto en Java vacío, listo para poder empezar a escribir nuestro código.

Todos los proyectos en NetBeans se componen de las siguientes partes:

- **Pestaña de archivos abiertos:** Aquí aparecerán todos los archivos de código que tengamos abiertos.
- **Explorador de archivos:** Aquí aparecerán todos los archivos y paquetes de nuestro proyecto.
- **Editor de código Java:** Aquí podremos editar el código Java.

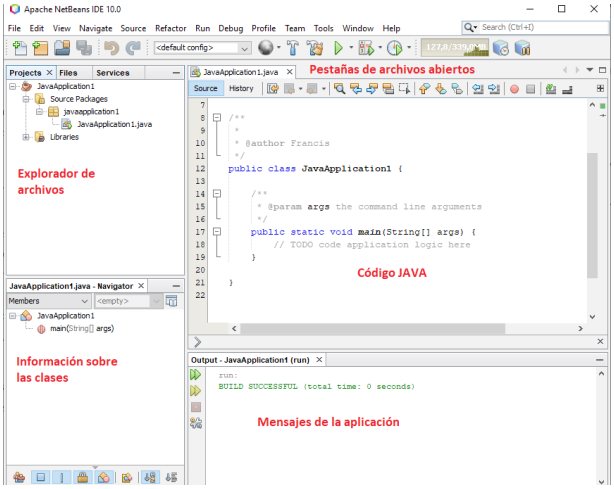

Fig. 10. Partes de un proyecto en NetBeans.

- **Información sobre las clases:** Aquí aparecerá cierta información sobre la clase que tengamos abierta, lo veremos más adelante.

- **Mensajes de la aplicación:** Aquí aparecerán todos los mensajes que nos muestre nuestro programa.

/ 10. Compilación de un proyecto en NetBeans

El proceso de compilación podemos definirlo como **el proceso por el cual partimos de un código fuente en un lenguaje de programación y lo transformamos a otro código equivalente en otro lenguaje de programación**.

El proceso de compilación pasa por las siguientes fases:

- Análisis léxico, sintáctico y semántico.
- Generación de código intermedio.
- Optimización de código.
- Generación de tabla de símbolos.

En nuestro caso, **el lenguaje fuente será Java** y el lenguaje al que lo transformaremos será el Bytecode. **El Bytecode es un lenguaje intermedio**, esto quiere decir que no es lenguaje máquina, sino que es un lenguaje que necesitará de un intérprete intermedio para poder ejecutarse. E**ste intérprete será la máquina virtual de Java, o Java Virtual Machine en inglés,** que es la que instalaremos con el JDK. En caso de que no tengamos esta máquina virtual, instalada no podremos ejecutar programas Java compilados al Bytecode.

El hecho de tener que instalar la máquina virtual hace que el lenguaje de programación Java sea multiplataforma, esto quiere decir, que indistintamente del sistema operativo que tengamos instalado (GNU/Linux, Windows o MacOS), siempre que la tengamos instalada podremos ejecutar programas en Java sin necesidad de volver a compilar el programa. Esto es una ventaja enorme frente a otros lenguajes dependientes de la plataforma, como por ejemplo C++, en los que si quieres ejecutar un programa en otro sistema operativo distinto al de la compilación inicial deberás volver a compilarlo forzosamente para esa plataforma.

Para ejecutar un programa en NetBeans deberemos pulsar el botón de compilado (icono de un martillo) o el de ejecución (con forma de triángulo verde). Una vez pulsado este botón, se iniciará automáticamente todo el proceso de compilado y en caso de haber algún error el compilador nos lo notificará en la parte de los mensajes de la aplicación, indicando el lugar y la causa del mismo. Si todo ha ido bien, se obtendrá el Bytecode y se ejecutará el programa si es el caso.

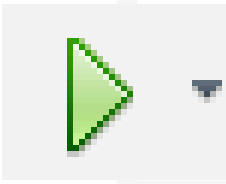

Fig. 11. Botón de ejecutar en NetBeans.

Vídeo 2. "Compilación de un proyecto de NetBeans".
https://bit.ly/2VsiZqK

/ 11. Resumen y resolución del caso práctico de la unidad

A lo largo de este tema, hemos visto un primer acercamiento al mundo de la programación. Se ha destacado la importancia del proceso de ciclo de vida del software y cómo hay que seguirlo para poder realizar una correcta implementación de un proyecto teniendo los menos problemas posibles.

Es importante entender la analogía de los lenguajes de programación con los lenguajes de comunicación de los humanos, ya que, al fin y al cabo, lo que estamos haciendo es un proceso de comunicación pero con una máquina, en lugar de con otro ser humano, y por ello, nos debemos adaptar a los lenguajes que éstas entienden.

Hemos visto que existen muchísimos lenguajes de programación en la actualidad y que vamos a utilizar el lenguaje Java. A su vez, también hemos visto que existen muchos Entornos de Desarrollo Integrado en el mercado, utilizando nosotros NetBeans 10, por tratarse de un IDE libre no propietario y multiplataforma.

Resolución del caso práctico de la unidad

Cuando echamos la vista al mercado laboral en los sectores de la informática, veremos que hay cientos de empresas que se dedican a ello, desde autónomos, pasando por pequeñas empresas y terminando en grandes multinacionales.

No hay que desanimarse con esto, ya que este es un mundo extensísimo y hay muchos sectores que necesitan especialistas de la programación en sus filas.

Una cosa que podemos hacer es especializarnos en lenguajes concretos en los que haya mucha demanda, en sistemas operativos concretos o en IDEs concretos, pero lo más importante sin duda alguna, es una buena base.

Fig. 12. Si se tiene una buena base en programación, aprender otros lenguajes resulta muy asequible.

/ 12. Bibliografía

Colaboradores de Wikipedia. (2020a, abril 8). *Sistema alfanumérico*. Recuperado 14 de mayo de 2020, de https://es.wikipedia.org/wiki/Sistema_alfanum%C3%A9rico

Colaboradores de Wikipedia. (2020, abril 22). *Desarrollo en cascada*. Recuperado 14 de mayo de 2020, de https://es.wikipedia.org/wiki/Desarrollo_en_cascada

Colaboradores de Wikipedia. (2020b, mayo 10). *Lenguaje de programación*. Recuperado 13 de mayo de 2020, de https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n

Gamboa, A. G. (2018, febrero 15). *¿Cuál es la relación entre los datos, algoritmos, y programas?* - Quora. Recuperado 14 de mayo de 2020, de https://es.quora.com/Cu%C3%A1l-es-la-relaci%C3%B3n-entre-los-datos-algoritmos-y-programas

Robledano, &. (2020, abril 14). *Qué es un algoritmo informático*. Recuperado 14 de mayo de 2020, de https://openwebinars.net/blog/que-es-un-algoritmo-informatico/

Tipos de Datos: Booleanos, Caracteres, Enteros, Fechas etc.. (s. f.). Recuperado 14 de mayo de 2020, de https://www.tecnologias-informacion.com/tipo-de-datos.html

Villagómez, C. V. (2017, marzo 8). *Ciclo de vida del software*. Recuperado 14 de mayo de 2020, de https://es.ccm.net/contents/223-ciclo-de-vida-del-software