Siempre hay varias maneras de programar una solución

En el área de la programación nos encontramos con un problema al cual hay que encontrarle una solución por medio de los algoritmos representados con código.

En el área de la programación nos encontramos con un problema al cual hay que encontrarle una solución por medio de los algoritmos representados con código.

Pero no solo en esta parte radica hacer la solución, también nos encontramos en armar una arquitectura, un diseño de base de datos, o un balance de componentes de tecnologías distintas que convivan entre si y juntos sean la solución.

El entender que el realizar un software de una manera la cual funcione bien, es decir, que cumpla su trabajo, dando la solución y esta solución sea segura, balanceada, escalar y todas esas características de calidad, **implica que es una de las tantas maneras de haber creado la solución**, más no es ni la mejor, ni la peor, pero si una de tantas.

A inicios del siglo 20 por allá de 1900 se reunieron un grupo de matemáticos, una asamblea donde a uno de estos matemáticos llamado **David Hilbert**, se le ocurrió plantear algunos problemas a resolver (desde entonces nos gustaba complicarnos), uno de esos problemas es el

llamado **Entscheidungsproblem** (problema de decisión) el cual propone lo siguiente:

¿Existe un algoritmo el cual pueda decirnos si un cálculo de primer orden es un teorema valido?

En palabras más simples, existe un algoritmo que pueda decirnos **si un enunciado lógico es demostrable.**

Es aquí un punto crítico en la historia de la computación, ya que gracias a este problema alguien llamado **Alan Turing** y alguien llamado **Alonzo Church** demostraron que era imposible tal algoritmo pero también demostraron que en la computación se puede encontrar una solución de distintas formas.

Alan Turing creo la máquina de Turing la base de la programación imperativa y por su parte **Alonzo Church creo el cálculo lambda** la base de la programación declarativa.

La programación imperativa se ha convertido en la base de paradigmas de programación como la programación Orientada a Objetos, en cambio, la **programación declarativa** lo vemos en paradigmas funcional o lógico (prolog), de hecho, hoy en día el combinar los paradigmas es el pan de cada día.

Al final, nosotros creamos algoritmos que darán con la solución a un problema, pero siempre hay formas distintas de hacerlo, con menor o mayor abstracción, con menos líneas o más líneas de código, como sea, siempre habrá maneras distintas de tener una solución.

El objetivo de este texto es entender que si estamos programando y no damos con la solución, es bueno comenzar de nuevo esa parte y pensar cómo se haría de una manera distinta, no cerrarnos al primer camino en el cual nos quedamos atorados, sino entender que siempre hay otra forma de hacerlo.

Esto aplica tanto para programar, para diseñar una base de datos, para la arquitectura de software, bueno, aplica para toda etapa en el desarrollo de sistemas.

Si has llegado hasta aquí te agradezco, puedes compartir el texto si ha sido de tu agrado.

25 marzo, 2021 / Blog / Alan Turing, Alonzo Church, calculo lambda, maquina de turing, paradigmas, programación, programación funcional, programación imperativa / Deja un comentario

¿Cómo tener motivación para Programar?

Esta pregunta la he escuchado bastante desde hace años, ¿Cómo obtengo motivación para programar?

Esta pregunta la he escuchado bastante desde hace años, **¿Cómo obtengo motivación para programar?**

La motivación es importante en el área de la programación, ya que programar sin motivación nos puede llevar a tener mejor o peor eficiencia.

Sin motivación podemos pasar horas en algo que puede ser sencillo de realizar, y con motivación podemos acortar el tiempo de algo que es difícil de realizar.

En el área de la psicología la motivación es un tema de debate, ya que la motivación puede influir en el comportamiento.

Puedes tener motivación por diversos factores que van desde el dinero, metas, o reconocimiento, el problema es cuando se pierde la motivación y aquí es cuando es bueno detenernos y pensar que es lo que nos motivaba en un principio y porque ya no nos motiva.

Un caso real de cuando perdemos la motivación viene cuando hemos comenzado un proyecto en el cual debemos aprender una **tecnología nueva**, el reto puede ser factor para tener motivación, el problema radica cuando al reto le echas problemáticas o reglas como el tiempo de entrega, cambios de requerimientos, o simplemente que la tecnología no es lo que uno esperaba.

El desmotivarnos es un problema para llevar el proyecto con eficiencia, ya que no seremos igual de productivos, y es aquí cuando comienzan los problemas. El perder el control del tiempo por no tener motivación de realizar las cosas es una bola de nieve que va creciendo hasta que se convierte en una avalancha.

En lo personal yo he pasado por muchas fases de desmotivación a través de mi vida laboral, y analizándolo ahora lo que me ha servido es lo siguiente: **pensar si vale la pena seguir con la tarea, si lo vale, ya habremos encontrado un factor de motivación, pero si no lo vale es momento de actuar, de hacer un cambio.**

Si la motivación se ha perdido porque no se siente que se gana el dinero suficiente, es momento de negociar y encontrar una solución, pero como todo, la negociación siempre debe ser yo te

ofrezco y tú me ofreces, un ganar-ganar para los 2 lados.

Si la falta de motivación es por cuestión de **carga de trabajo**, es momento de jerarquizar y organizar, no quedarnos callados, pedir ayuda, ya sea de un recurso extra, ya sea de flexibilidad en el tiempo de entrega, más vale hacer algo bien que algo a la carrera, esto ahorrara mucho tiempo a futuro.

Si pierdes la motivación te recomiendo analizar qué es lo que te motiva en realidad, y ver como eso que te motiva puede ser obtenido con la tarea que realizas, o como llegar a realizar un cambio para que puedas estar en camino a la meta anhelada.

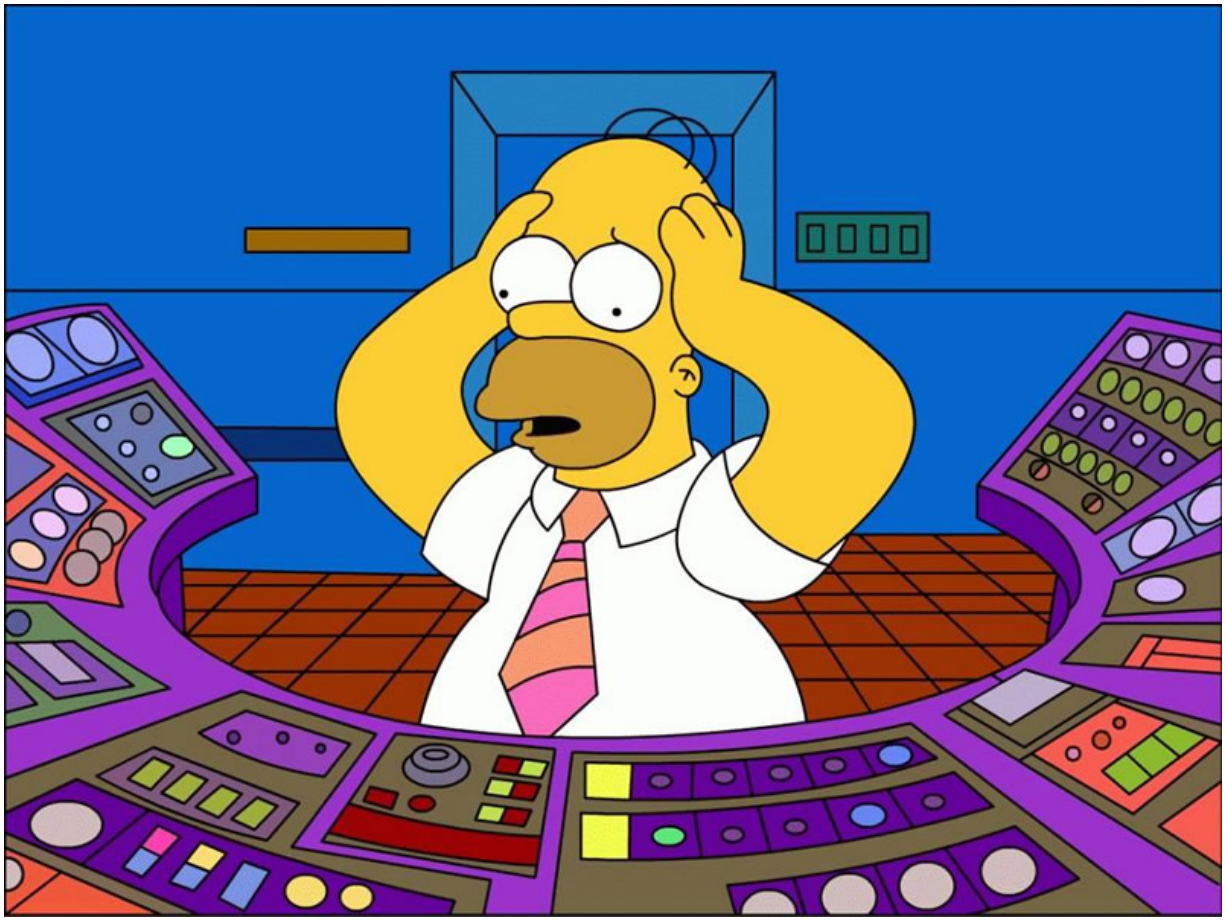
El motivarte es algo tan importante, que puede ser el recurso necesario en hacer algo bien, y no es malo apoyarnos también de sentimientos como el hacerlo por la familia o seres cercanos, ahí también se encuentra un factor que puede ser motivante, ya sea para compartir lo ganado con ellos como el tiempo o el dinero (el cochino dinero).

Cuando sientas que no tienes motivación, busca hacer un cambio, no solo te quedes estancado, hay que analizar que debemos cambiar para que regrese esa motivación con la cual un día estuvimos de buenas, **se vale cambiar de trabajo, de tecnología, de retos, todo esto se vale**.

Si llegaste hasta aquí te agradezco, que pases un buen día.

23 marzo, 2021 / Blog / estrés, familia, motivación, paz, programación, programar, trabajo / Deja un comentario

Lo que se aprende al terminar un proyecto de software



A través de los años he terminado bastantes proyectos como programador líder o arquitecto de software, y he encontrado un patrón común en todos ellos, y el patrón común es el siguiente pensamiento: *si uno comenzara nuevamente el proyecto, lo haría mejor.*

Esto nos trae cierta confusión, ya que nos hace pensar que lo hicimos de una manera no correcta, y nos hace plantearnos si es que realmente no somos buenos.

En sí, esta situación es de lo más normal, siendo parte del aprendizaje, la experiencia es eso, aprender a través de errar, el errar no es malo, el errar siempre ayuda, y sobre todo, ayuda más cuando ya no erramos donde hemos errado.

Con la experiencia que vamos ganando, vamos mejorando en tomar decisiones correctas, que harán reflejo en nuestros nuevos software que haremos en el futuro.

También es importante la retroalimentación del equipo de trabajo, y porque no, del usuario final, ya que el usuario final es el que vestirá nuestra creación.

Con el tiempo nos daremos cuenta que nos costara menos trabajo hacer ciertas tareas que en su momento fueron nuestro martirio, encontraremos soluciones de manera intuitiva, y pareceremos artesanos esculpiendo código como maestros.

Al finalizar un proyecto, es de mucha ayuda analizar y pensar de que otra forma lo hubiéramos hecho, y así, ver huecos donde nos los habíamos visto, y veras que esos huecos ahí están, esas oportunidades de ahorro de tiempo al trabajar (Una abstracción mejor, una implementación de clases más apta).

Al mirar atrás (al ver mis códigos viejos), hace algunos años me sentía asustado, en como un sistema parecido a un espagueti está ahí funcionando, en la espera a que alguien tumbe esa torre de **jenga**, pero ahora, al mirar esos códigos, los veo como fuertes cimientos los cuales han forjado la experiencia que tengo ahora, en lo mismo de párrafos arriba, en el errar y mejorar.

El consejo que le doy a toda persona que comienza a programar es: *siempre hay distintas formas de llegar a la solución, y siempre hay unas mejores que otras, con los años encontraremos esos fósiles de abstracción, pero por ahora diviértete experimentando, escarbando, y pronto miraras atrás y dirás: ¡Mira ese jenga! Cuando lo he armado lo he dejado tan perfectamente balanceado que sigue de pie.*

Si llegaste hasta aquí, te agradezco bastante.

25 febrero, 2021 / Blog / aprendizaje, espagueti, experiencia, jenga, software / Deja un comentario

Los 10 mejores lenguajes de programación en 2021

Estos 10 lenguajes de programación tendrán una gran protagonismo en el 2021 y he hecho un resumen de lo que tratan.

Estos 10 lenguajes de programación tendrán una gran protagonismo en el 2021 y he hecho un resumen de lo que tratan.

Esta entrada está basada en encuestas de popularidad, salarios en estados unidos y aclaro, un ordenamiento personal, subjetivo, sobre estas fuentes, hecho por mí.

Por lo cual tomarlo como 10 lenguajes que están en estos listados, pero, tienen mi juicio personal.

10. Swift

Este lenguaje de programación es utilizado para crear aplicaciones móviles para Iphone y aplicaciones para macOs, por lo cual su aceptación laboral va a seguir siendo clave.

El lenguaje de programación fue creado por **Chris Lattner en Apple** en el año 2014, y desde entonces ha sido un lenguaje de programación que ha ido madurando versión a versión.

9. R

Lenguaje de programación creado en el año de 1995 por **Robert Gentleman y Ross Ihaka** del departamento de estadística de la Universidad de Auckland. Aunque le sirvió como referencia un lenguaje de programación más antiguo llamado lenguaje S.

Este lenguaje de programación está orientado al análisis estadístico, minería de datos, machine learning e investigación científica, e incluye un numero amplio de herramientas estadísticas las cuales se pueden extender en funcionalidad gracias a este lenguaje.

R también permite conectarse a distintos gestores de base de datos, así como al lenguaje de programación **Python.**

8. Go

El lenguaje de programación Go nace en el año 2009 de la mano de Robert Griesemer, Rob Pike y **Ken Thompson** (creador junto a Dennis Ritchie de Unix) en la empresa google.

El lenguaje de programación es concurrente y compilado y es simular al lenguaje C y Python.

Algunas de sus particularidades son su facilidad para aprenderlo, el punto y coma es opcional y no existen las Excepciones, ya que si hay una excepción debería ser eso, un caso excepcional.

Este lenguaje es utilizado en procesos que necesiten cargar información masiva, software que tenga poca interacción con usuarios y procesos que necesiten concurrencia.

Algunas empresas que utilizan el lenguaje a parte del mismo Google son: **Netflix, Paypal, Twitch y Uber**.

7. C#

El lenguaje C# nace en el año 2000 de la mano de **Anders Hejlsberg** (creador de Typescript).

Desde entonces este lenguaje de programación ha tenido un crecimiento de madurez constante.

C# es un lenguaje de programación multiparadigma y multiplataforma, siendo el lenguaje de programación principal de .Net, por lo cual lo podemos encontrar en aplicaciones web, Iot, videojuegos, aplicaciones móviles y escritorio.

6. Php

Php es un lenguaje de programación amado por muchos, y odiado por otros, y esto quizá se deba al gran número de proyectos existentes. **El 78.9 % de sitios web están hechos con php (2020)**, es decir, 8 de 10 sitios están hechos en php.

El lenguaje ha tenido un empuje en rendimiento a partir de su versión 7 y es utilizado también para hacer **Web Scraping** gracias a la facilidad que nos proporciona por su naturaleza de lenguaje interpretado.

Si tú quieres hacer una aplicación web, deberías investigar antes si ya existe un Opensource hecho en php, ya que es una de las grandes ventajas de este lenguaje, que hay una gran variedad de soluciones ya hechas gratuitas que nos podrían resolver alguna problemática.

5. C++

C++ nacio como un lenguaje de programación que extendia al lenguaje C.

Fue llamado "**lenguaje C con clases**" al inicio, pero termino llamándose C++ refiriéndose a la instrucción de incremento de C.

Este lenguaje sigue actualizándose cada año, y ya incluye funciones lambda.

El lenguaje de programación C++ tiene este lugar ya que lo encontramos detrás de muchísimas aplicaciones Iot, la creación de compiladores, sistemas operativos, gestores de base de datos, videojuegos, navegadores o motores de código como V8 el cual corre javascript.

4. Java

Java es un lenguaje de programación que fue creado en 1996 por el equipo Gren Team en Sun Microsystem.

El lenguaje es multiparadigma y multiplataforma y tuvo una gran aceptación en los años 90s, lo cual lo hizo un lenguaje clave a aprender, ya que mucho software de bancos y empresariales están hechos con este lenguaje.

También lo seguimos encontrando en aplicaciones móviles en Android, aplicaciones embebidas, escritorio, videojuegos (Minecraft fue creado con este lenguaje en un inicio) y desarrollos web.

3. C

Muchos se sorprenderán de ver al lenguaje de programación C en este lugar.

El lenguaje C fue creado en los años 70 de la mano de Dennis Ritchie en los laboratorios Bell.

Nació como evolución del lenguaje B para resolver problemáticas que daba el estar programando en lenguaje ensamblador.

Este lenguaje es la base de muchos sistemas operativos hoy en día, el mismo Windows 10 tiene mucha funcionalidad base creada con este lenguaje.

En pocas palabras, el lenguaje de programación C es el pilar de muchos lenguajes que utilizamos hoy en día, y aunque no lo creas, sigue siendo utilizado bastante en los cimientos de lo que utilizamos todos los días: **los sistemas operativos.**

2. Javascript

Javascript es un lenguaje que nació en 1995 de la mano de Brendan Eich en Netscape.

Este lenguaje lo encontramos en todos lados, siendo el lenguaje de programación base para crear aplicaciones web del lado del frontend.

Pero javascript va más allá, gracias a que la mayoría de programadores saben programar poco o mucho en este lenguaje, ha sido llevado a ser lenguaje con el cual podemos crear videojuegos, aplicaciones de escritorio, aplicaciones móviles, en pocas palabras, javascript está en casi todos lados.

Su gran aceptación da un gran potencial de encontrar trabajo ya que es de los lenguajes mas solicitados.

1. Python

En el año de 1991 nace este lenguaje de programación de la mano de Guido Van Rossum en el centro para las matemáticas y la informática en Paises Bajos.

El lenguaje ha tenido un crecimiento enorme en los últimos años gracias a la gran facilidad que

nos aporta para crear aplicaciones de Inteligencia Artificial y para crear **Machine Learning.**

Con la repercusión que ha ido teniendo la IA en los últimos años, Python es de los lenguajes más utilizados, por lo cual tenemos ya muchísima funcionalidad creada para solucionar estas problemáticas.

También es utilizado para desarrollo web de la mano de frameworks como **Django o Flask.**

Conclusión

Este top fue creado para darte a conocer los lenguajes de programación más utilizados o con mayor aceptación en el mercado laboral.

Aquí te dejo un podcast donde estuvo de invitado coderos y hablamos de este top.

Fuentes:

1. https://www.tiobe.com/tiobe-index/
2. https://www.northeastern.edu/graduate/blog/most-popular-programming-languages/
3. https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm
4. https://www.jetbrains.com/es-es/lp/devecosystem-2020/
5. https://insights.stackoverflow.com/survey/2020

19 diciembre, 2020 / Blog / lenguajes de programación, top 10 / Deja un comentario

El Roadmap de C#

El objetivo de esta entrada es darte un camino que pueda servirte en tu preparación al adentrarte en el lenguaje de programación C#.

El objetivo de esta entrada es darte un camino que pueda servirte en tu preparación al adentrarte en el **lenguaje de programación C#**.

Esta entrada ira en actualización constante (solo pondré los cursos completos), aquí iré poniendo el camino a seguir para aprender C# con el contenido ya existente en mi canal.

Fundamentos de C#

Los fundamentos son importantes ya que te daran las bases solidas antes de entrar a crear aplicaciones reales.

Muchas veces pasa que entramos de lleno a un framework sin comprender las bases, y es cuando vienen los problemas

1.- Tipos de datos y var | Curso de fundamentos de C#

1.- Tipos de datos y var | Curso de fundamentos de C#

2.- Clases, objetos y constructores | Curso de fundamentos de C#

3.- Arreglos y listas | Curso de fundamentos de C#

1 of 5 Next

Programación orientada a objetos

La programación orientada a objetos es fundamental para llevar buenas prácticas, es por ello que una vez que has visto el curso de fundamentos, el siguiente paso sería la siguiente lista de videos.

Curso de programación orientada a objetos en C# .Net #1 | Clases y objetos

1 of 3 Next

Curso de programación orientada a objetos en C# .Net #1 | Clases y objetos

Curso de programación orientada a objetos en C# .Net #2 | Herencia

Curso de programación orientada a objetos en C# .Net #3 | Sobrecarga

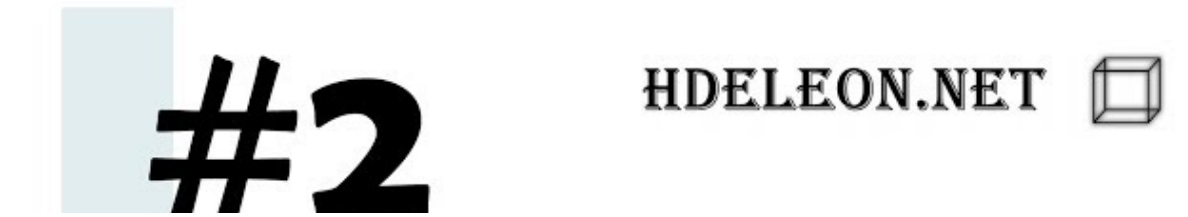
Una vez que se comprende la programación orientada a objetos, es momento de entrar a un framework para desarrollo de cosas reales.

La lista de desarrollo en MVC .Net es el punto a seguir, donde aprenderás a crear sistemas web en el patrón arquitectónico MVC (***Pronto se actualizara este curso por MVC .Net 5***)

Curso de MVC .Net C#| Introducción, controladores y vistas | #1

1 of 3 Next

Curso de MVC .Net C#| Introducción, controladores y vistas | #1



Curso de MVC .Net C#| Entity Framework, autentificación | #2

Curso de MVC .Net C#| Filtros, seguridad | #3

Conforme sigan terminándose de grabar más cursos irán apareciendo en esta entrada, por lo pronto puedes suscribirte a mi canal dando clic aquí, que tiene ya más de 600 videos.

1 diciembre, 2020 / C#, Roadmap, Sin categoría / .net, .net 5, csharp, dotnet, programación, Programación Orientada a Objetos, roadmap / Deja un comentario

¿Cómo superar la frustración al programar?

No es nuevo que yo te diga que **la frustración es parte de cuando uno está programando**, ya sea por no dar con la solución, ya sea por un bug con la tecnología, o por un sinfín de cosas más, esto es parte de la vida de un programador y espero te sirva lo que a mí me ha servido a través de los años para romper con esta maldición.

Creo que me di cuenta de cómo ir superando con esto gracias a una situación que me paso hace algunos años y te contare la anécdota, quizá te identifiques.

Cuando parecía no haber solución

Hace algunos ayeres me encontraba resolviendo una problemática: hacer un administrador de hilos.

Tenía que administrar un pool de hilos (grupo de hilos limitado disponibles para realizar una tarea) el cual estuviera al tanto para hacer la solicitud de un servicio el cual me retornaría una gran cantidad de información la cual debía ser base para una base de datos local.

El proceso necesitaba de algunos minutos por cada solicitud, por lo cual se optó por hacer un pool de hilos, esto yo lo comencé a crear de cero (era un chaval que no se le ocurrió que ya existía en .Net una implementación, aquí te muestro un video).

Cuando manejamos hilos tenemos que entender un conjunto de prácticas, estos hilos que trabajaran a la par, **¿Compartirán recursos?, ¿Compartirán variables?** Hay que considerar ciertas cuestiones que yo en mi novata vida de programador era nuevo.

Como todo en la vida, cuando uno es nuevo en un terreno la frustración llegó, gracias a que la administración de hilos soltaba excepciones por el hecho de no entender que al compartir recursos, algunos de estos recursos solo pueden ser leídos 1 a la vez, cosa que desconocía y me llevó a la frustración.

Intente varios mecanismos pensando que todo era cuestión de mala planeación del algoritmo, pero en realidad el algoritmo iba bien, lo que no iba era esa parte de compartir recursos, algo que seguía yo ignorando, así que después de días sin dormir bien llegue a la conclusión de

dejarlo por un tiempo y hacer otro tipo de cosas, como salir a correr.

Y es aquí cuando **mi cerebro de manera inconsciente llegó a esa cuestión** ¿No será que los hilos al compartir recursos de lectura causen este estrago?, y si, eso era, el compartir la lectura de un archivo de configuración de la vieja escuela hacia que esto no fuera para adelante.

En mi caso me sirvió dejar la codificada un tiempo, darle relajamiento al asunto, salir a correr, y la solución llegó, esta solución a partir de ese momento me sirvió, ahora cada que no veo solución después de intentar por horas, lo que hago es cerrar la laptop y hacer algo ajeno a programar y creerme, la mayoría de veces llegara la solución.

El subconsciente es muy trabajador

Esto no es mágico, en el libro llamado "La mente nueva del emperador de Roger Penrose" se habla sobre cómo es que llegan las ideas, en un apartado llamado "Inspiración, perspicacia y originalidad" se dan algunos ejemplos recopilados por el matemático francés Jacques Hadamard, y aquí transcribo un ejemplo de su colega Henri Poincaré que cita después de un momento de frustración intenso en sus propias palabras:

> *"...Dejé Caen, en donde vivía, para participar en una excursión geológica organizada por la Escuela de Minas. Las peripecias del viaje me hicieron olvidar mi trabajo matemático. Al llegar a Coutances abordamos un autobús para ir a algún lugar. En ese momento, cuando puse mi pie en el estribo, me vino la idea, sin que nada en mis pensamientos anteriores pareciera haber preparado el camino para ello, de que las transformaciones que había utilizado para definir las funciones Fuchsianas eran idénticas a las de la geometría no euclidiana. No verifiqué la idea; no hubiera tenido tiempo, ya que en cuanto tomé asiento en el autobús continué una conversación ya comenzada, pero tenía la certidumbre absoluta. A mi vuelta a Caen, y para quedarme tranquilo, verifiqué detenidamente el resultado."*

Esto parece algo típico de programadores, pero vemos que se da en todos lados (**Newton y la manzana**), la mente subconsciente sigue trabajando en el material ya recopilado, pero a veces solo falta darle un descanso al consciente para que el subconsciente se encargue de buscar y brindar esa idea que nosotros estando buscando no dimos.

Es normal que te frustres, pero hay que darle un respiro al cerebro realizando otra actividad, ¿O por qué no? Dormir, dudo ser el único que ha resuelto un problema de programación dormido.

30 noviembre, 2020 / Blog / frustración, la mente nueva del emperador, roger penrose, subconsciente / Deja un comentario

¿Qué diablos son los paradigmas de programación?

Comencemos por las palabras separadas, la palabra paradigma significa: ejemplo o modelo de algo, es decir, un paradigma es una base con la cual puedes partir para obtener una respuesta a algo.

Comencemos por las palabras separadas, **la palabra paradigma significa**: ejemplo o modelo de algo, es decir, un paradigma es una base con la cual puedes partir para obtener una respuesta a algo.

Ahora, combinándolo con el concepto de **programación** tenemos que ***un paradigma de programación es un modelo para crear programación***, y ¿Qué es lo que hace la programación? Pues resolver problemas, creando soluciones.

Entonces tenemos que **un paradigma de programación son modelos para partir en la creación de estas soluciones**, y aquí es donde, ya entendiendo lo que es un paradigma, ya podemos entender lo siguiente: en la programación hay **distintos modelos** con los cuales podemos llegar a crear soluciones, por ejemplo tenemos el paradigma funcional, el paradigma orientado a objetos, el paradigma lógico y algunos otros más, y aquí vamos a darle una revisión a algunos.

Paradigma orientado a objetos

Este paradigma se centra en caracterizar el universo en el que vivimos, si observamos nuestro entorno, **todo está compuesto por objetos**, objetos que interactúan con otros objetos, objetos que son parte de otros objetos, todo es un objeto, pero para diferenciar un objeto de otro tenemos propiedades y acciones que hacen estos objetos.

Por ejemplo, sabemos que una cerveza es una cerveza porque tiene ciertas características y propósito: es líquido, tiene alcohol, tiene un porcentaje de malta, lúpulo y algunas otras cosas, estas cosas las llamaremos atributos o propiedades, y juntas hacen que una cerveza sea un objeto el cual podemos identificar por las mismas propiedades.

Pero una cerveza no solo tiene propiedades, también tiene un propósito o funcionalidad, y su funcionalidad es que se puede beber, y aquí lo bonito del ejemplo ¿Quién se bebe la cerveza? Pues un humano como yo, que viene siendo otro objeto con sus características y funcionalidades propias.

En este paradigma nos basamos en el mundo real para representar la problemática a resolver, como estos objetos interaccionan unos con otro, y gracias al concepto de Clase podemos crear estos objetos.

La clase en este paradigma es un molde que sirve para crear los objetos, como el molde para

crear un auto por ejemplo, un molde que sabe que debe tener puertas, llantas, motor, ventanas y un proceso secuencial para crearlo, eso es una clase.

Este paradigma tiene más características de las cuales hablare en una entrada propia para adentrarnos de lleno.

Si deseas aprender programación orientada a objetos te recomiendo esta lista de videos.

Paradigma funcional

Cuando escuchamos hablar de este paradigma, creemos que es algo nuevo, que es el siguiente paso de la programación, y bueno, en realidad este paradigma es mucho más viejo que el paradigma orientado a objetos.

Este paradigma se basa en el concepto matemático de **cálculo lambda**, un sistema formal que fue creado en la década de 1930 gracias a **Alonzo Church**, este paradigma se basa en funciones que invocan otras funciones las cuales invocan a otras funciones.

Con funciones podemos hacer lo mismo que con la programación imperativa (programación que se basa en cómo hacerlo), en este paradigma nos preocupamos más en el ¿Qué vamos a hacer?, una forma de resolver los problemas de manera más humana.

Hay que resaltar que el paradigma funcional puede convivir con el paradigma orientado a objetos, es algo que estamos viendo últimamente en los lenguajes de programación más utilizados, pero también el paradigma funcional no solo es la creación de funciones, sino también llevar ciertas prácticas para cumplir con este objetivo

No me adentrare mucho, solo te mencionare que una función en este paradigma debería siempre regresar el mismo resultado si siempre es invocado con los mismos parámetros, por ejemplo una función suma que recibe 1 y 2 siempre debe devolver 3, y no depender de efectos colaterales, esto nos ayuda a que podamos remplazar suma por otra función que haga lo mismo sin que afecte a otras partes de nuestro sistema.

Si quieres darte una idea de este paradigma ya a nivel práctico te recomiendo este video.

Paradigma lógico

El paradigma funciona y el paradigma lógico son parte de la **programación declarativa**, aquella que se centra en **que vamos a hacer**, y no el cómo vamos a hacerlo.

El paradigma lógico se centra en el concepto de **lógica matemática**, todo son predicados y hechos.

Teniendo una base de hechos como "**Todos los que toman cerveza son borrachos**" y consultando **¿Quiénes son borrachos?** Este paradigma nos regresaría "**Todos los que toman cerveza**".

Este paradigma resulta útil en **sistemas expertos** gracias a su naturalidad de hechos y predicados, imaginar lo siguiente, cuando uno va al doctor, va con un experto en una rama, este experto tiene una base de hechos en su experiencia, y dependiendo de nuestras preguntas el utilizara sus predicados para darnos una respuesta, tal cual funciona este paradigma.

También es utilizado para el **reconocimiento del lenguaje natural y demostración de teoremas.**

En este video te explico el lenguaje lógico prolog por si deseas darte una idea de a que va este paradigma.

Paradigma reactivo

Este paradigma es lo de moda de hoy en día, pero en realidad tiene ya sus años de existir.

El paradigma reactivo se **centra en los flujos de datos y como entrelazarlo con objetos.**

Por ejemplo, todos hemos utilizado Excel, sabemos que Excel se compone de celdas (los observadores), y sabemos que en Excel podemos hacer formulas, como una fórmula que sume un conjunto de celdas, la celda donde aparecerá el total será un observador de algo, este algo es la sumatoria de el conjunto de celdas que representan la suma, por lo cual si una de las celdas dentro de la formula tiene un cambio, la celda que observadora representara este cambio de manera automática, esto es el paradigma reactivo.

Tenemos muchas ventajas con este paradigma que se basa en un conjunto de **patrones de diseño**, sobre todo en el **patrón observador**, el cual es muy sencillo de explicar: teniendo un **sujeto**, siendo el sujeto cualquier cosa que tenga información, existirá una **colección de observadores** interesados cada que cambie el sujeto sus valores, estos observadores estarán a la espera de cuando el sujeto cambie algo y serán notificados de ese cambio, cada sujeto sabrá que hacer dependiendo ese cambio, esto en pocas palabras es el patrón observador.

En el siguiente video te muestro como puedes programar el patrón observador en javascript

Conclusión

Como te diste cuenta hay distintas formas de crear programación (hay más paradigmas no mencionados), y un punto que quiero plantear aquí, es que con cualquiera de las formas anteriores mencionadas puedes llegar al mismo resultado, la diferencia radica que unas serán mas aptas que otras para cierta problemática.

Los paradigmas no son enemigos, últimamente los lenguajes de programación más utilizados son multiparadigma, esto con el propósito de darnos flexibilidad de poder trabajar con un estilo de programación dependiendo la circunstancia.

Así que ya sabes que es un paradigma, y espero esta entrada te haya dado su propósito, gracias por llegar hasta aquí.

Te dejo una charla que tuve con **Nicolás Schürmann del canal HolaMundo** sobre paradigmas de programación.

28 noviembre, 2020 / Blog / Alonzo Church, paradigma lógico, paradigmas, programación funcional, Programación Orientada a Objetos, prolog / Deja un comentario

¿Cómo aprender un nuevo lenguaje de programación?

Adentrarnos en un **nuevo lenguaje de programación** es un proceso que más de una vez va a pasarnos en nuestra carrera como programadores.

En mi historia he trabajado con más de 10 lenguajes de programación a lo largo de 20 años programando, y puedo decir que es ya un proceso común para mi aprender (o re-aprender) nuevos lenguajes de programación.

En mi experiencia puedo darte algunos consejos de cómo es que yo aprendo un nuevo lenguaje (**no framework**) de programación y es lo siguiente.

Fundamentos del lenguaje

Siempre, lo primero que hago es ir a ver los fundamentos del lenguaje, es decir, las cosas básicas del mismo, sin importar que tenga experiencia en otros lenguajes, es bueno no suponer, la suposición de creer que es igual una palabra reservada en un lenguaje que en otro es un error al cual no debemos caer (*por ejemplo la palabra reservada **var de javascript** que es muy distinta a la misma palabra **var en c#** por ejemplo*).

Los fundamentos van desde como declarar una variable, que paradigmas soporta el lenguaje (orientado a objetos, funcional, estructurado), como declarar una clase, si el lenguaje soporta interfaces o no, las cualidades únicas del lenguaje, las debilidades y ventajas del lenguaje son importantes también, pero sobre todo ir a este lenguaje nuevo como una persona que se adentrara sin suposiciones traídas de otros lenguajes de programación.

Documentación oficial

Para leer sobre los fundamentos, cuando aprendo un lenguaje de programación que tiene gran cantidad de apoyo, la documentación oficial siempre nos dará un camino sobre las cualidades de este.

La documentación oficial, sobre todo en inglés, ayuda dándonos lo correcto, lo oficial, y de primera mano la información correcta de cómo utilizar el lenguaje.

Antes de tomar un curso, te recomiendo ir a la documentación oficial.

Cursos

Una vez que se han leído los fundamentos, que se ha ido a la documentación oficial, ya podemos tener una base sólida para tomar un curso, ¿Por qué no antes? Pues porque necesitamos la base para poder tener un criterio del curso que tomaremos, hay cursos buenos y malos, pero para saber si estamos perdiendo el tiempo o no, necesitamos este criterio.

Los cursos son un complemento en el aprendizaje, pero es importante tener conocimiento por lo menos base antes de tomarlo, ya que algunos cursos podrían mal educarnos, o no darnos las bases que la documentación oficial ya a este punto nos habrá brindado.

Al tomar un curso ya sea de paga o gratuito hay que leer lo que comenta la gente, la ventaja de youtube es que podemos ir a los comentarios y leer sobre que va el curso, la ventaja en algunas plataformas de paga es que podemos ver lo que la gente dice del curso, es bueno antes de pagar saber si por lo que se va a pagar vale la pena.

Programar

Suena como algo obvio, pero lo he puesto ya que no se gana nada con solo leer, hay que darle al código, no hay buen corredor que sea buen corredor solo leyendo, hay que correr, hay que errar, hay que aprender de los errores, y un consejo es ir más allá del típico "**Get Started**" de la documentación, y puedes hacerlo vislumbrando algo que puedas hacer que sea utilizado por alguien, una aplicación básica que tenga la solución a algo básico.

Por ejemplo, si es un lenguaje para aplicaciones móviles, pues crear una app que resuelva alguna tarea de la vida diaria ya sea personal o de algún conocido, esto nos ira dando pautas de que ir investigando para ir solucionando las distintas partes de la aplicación. Esto ira forjando un camino orgánico de investigación, ya que tendrás un fin y objetivo por el cual puedes ir investigando, aprendiendo y sobre todo programando.

Comunidades

Las comunidades son parte esencial en el aprendizaje, ya que brindan un apoyo de compañerismo, apoyándonos en momento cuando caigamos en bugs endemoniados, o dudas sobre el lenguaje que estamos adentrándonos.

Por suerte, hoy en día las comunidades las podemos encontrar desde foro, hasta en slack, telegram o discord.

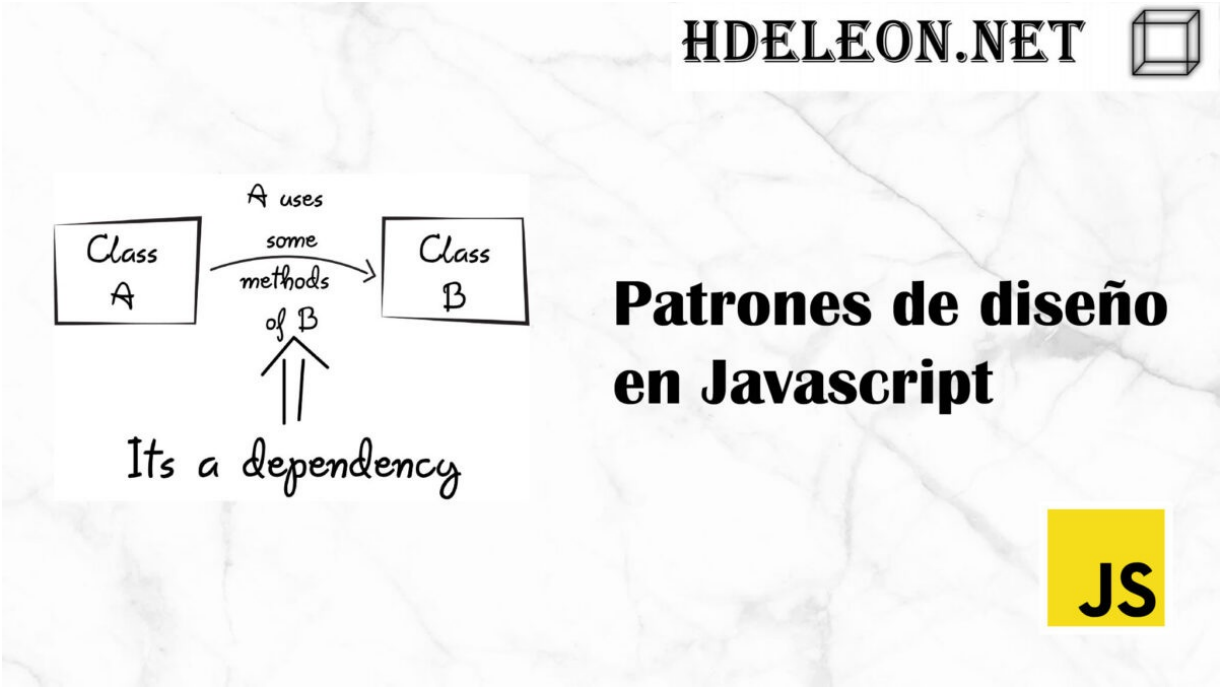
Es bueno involucrarnos con comunidades, ya que podemos tener varios puntos de vista extras, que nos podrán dar más objetividad a nuestro conocimiento.

Conclusión

Espero estos consejos te sirvan, y sobre todo, entender que para aprender un nuevo lenguaje de programación, se debe tener un propósito por el cual deseamos aprenderlo, ya sea laboral, ya sea por gusto, pero nunca que sea porque lo escuchamos como el lenguaje de moda, no caigamos en fanatismos.

27 noviembre, 2020 / Blog / aprendizaje, lenguaje de programación / Deja un comentario

Patrones de diseño en javascript



Los patrones de diseño son técnicas ya probadas que resuelven problemáticas en concreto.

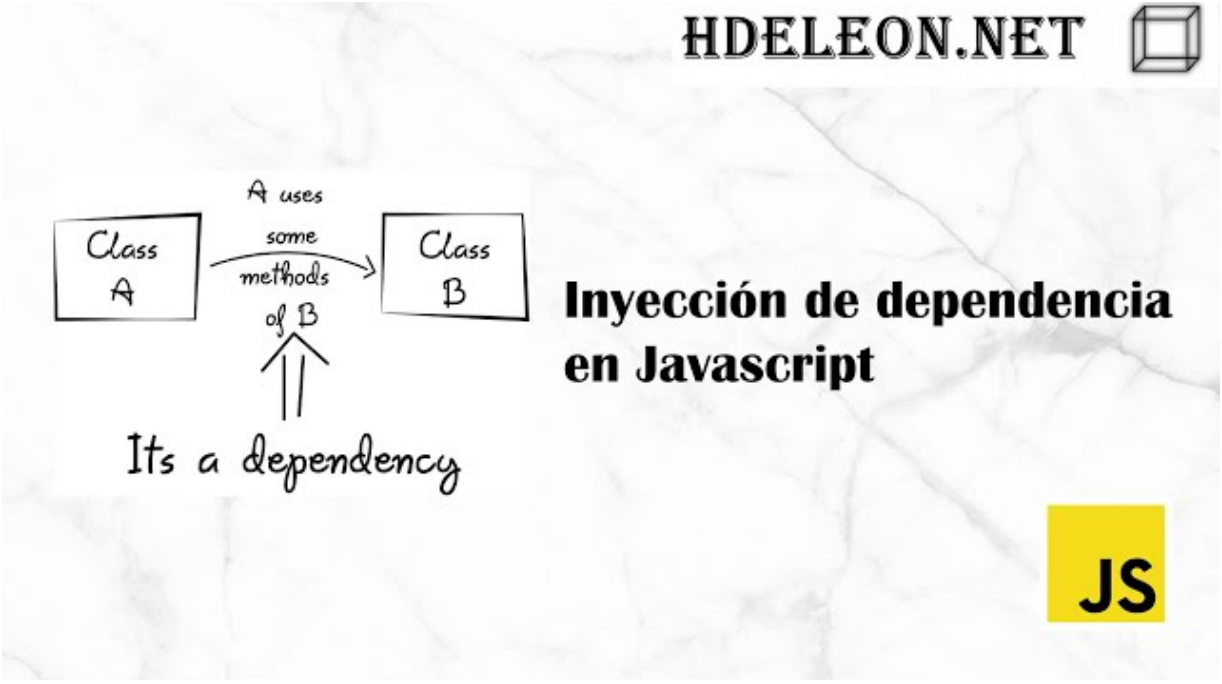
Imagina que tienes **soluciones** que ya han sido utilizadas por muchos programadores, y estas soluciones fueron documentadas como soluciones comunes para un tipo de problemas comunes.

Aprender el porqué de los patrones de diseño te ayudara a poder tener una mente más organizada al momento de programar.

En esta entrada tienes videos de varios patrones de diseño explicados en javascript.

El curso es publicado en mi canal de Youtube y para seguir el rastro te invito a que te suscribas dando clic aquí.

Inyección de dependencia en Javascript



Inyección de dependencia en Javascript 🔥

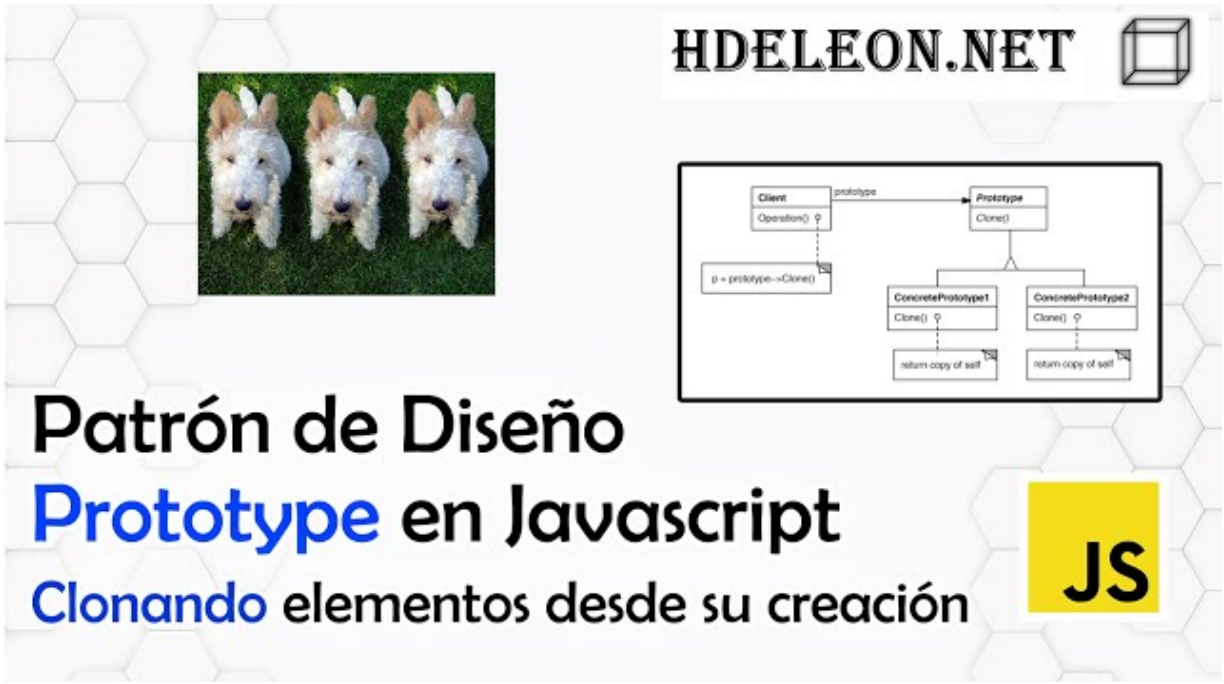
En este video te daré un acercamiento a la inyección de dependencia programada en javascript.

Patrón observador en Javascript 👀

En este video te daré una explicación del patrón de diseño observador implementándolo en javascript.

Patrón Iterador en javascript, a recorrer elementos de forma elegante

En este video te daré una explicación del patrón Iterador en javascript, que te servirá para recorrer listas de elementos de una manera elegante.



Patrón de diseño Prototype en javascript, creando objetos a partir de otros ya existentes

En este video te daré una explicación del patrón Prototype en javascript, que te servirá para crear objetos a partir de otros objetos ya iniciados.

19 noviembre, 2020 / Cursos, Javascript, Sin categoría / inyección de dependencia, iterador, javascript, js, observador, Patrones de diseño / Deja un comentario

Curso de fundamentos de C# .Net Core

En este curso aprenderas las caracteristicas fundamentales del **lenguaje de programación C#**.

Al finalizar el curso aprenderas:

- Variables y tipos de datos
- Programación orientada a objetos: herencia, polimorfismo, control de acceso
- Generics
- Deserialización y serialización de objetos
- Json
- Conexión a base de datos
- Interfaces en Programación orientada a objetos
- Solicitudes a servicios web
- Delegados
- Excepciones
- Creación, lectura y escritura de archivos

El curso esta en desarrollo y para que no pierdas ningún video puedes suscribirte a mi canal aquí.

1.- Tipos de datos y var | Curso de fundamentos de C#

1.- Tipos de datos y var | Curso de fundamentos de C#

#Csharp #core #var

2.- Clases, objetos y constructores | Curso de fundamentos de C#

3.- Arreglos y listas | Curso de fundamentos de C#

#Csharp #core #listas

4.- Interfaces | Curso de fundamentos de C#

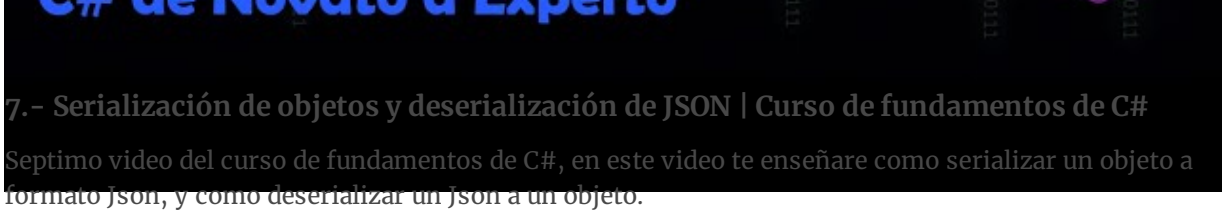
#Csharp #core #interfaces

5.- Conexión a base de datos, obtener información | Curso de fundamentos de C#

Quinto video del curso de fundamentos de C#, en este video te enseñare como puedes conectarte a una base de datos para obtener información guardada.

6.- Conexión a base de datos, crear, editar y eliminar contenido | Curso de fundamento...

Sexto video del curso de fundamentos de C#, en este video te enseñare como puedes crear, editar y eliminar contenido guardado en una base de datos.



7.- Serialización de objetos y deserialización de JSON | Curso de fundamentos de C#

Septimo video del curso de fundamentos de C#, en este video te enseñare como serializar un objeto a formato json, y como deserializar un json a un objeto.

8.- Solicitudes a Servicios Web por HTTP GET | Curso de fundamentos de C#

Octavo video del curso de fundamentos de C#, en este video te enseñare como solicitar por HTTP GET información a un servicio web.

9.- Envío de información a Servicios Web por HTTP POST, PUT y DELETE | Curso de fun...

Noveno video del curso de fundamentos de C#, en este video te enseñare como enviar información por solicitudes HTTP POST, PUT y DELETE.

11.- LINQ | Curso de fundamentos de C#

Onceavo video del curso de fundamentos de C#, en este video veremos LINQ, una herramienta que nos ayudara en el manejo de las colecciones.

10.- Generics | Curso de fundamentos de C#

Decimo video del curso de fundamentos de C#, en este video te enseñare para que sirven los Generics y como pueden hacer que reutilices código de manera sencilla.

12.- LINQ para manipular y obtener objetos complejos con subconsultas | Curso de fun...

Doceavo video del curso de fundamentos de C#, en este video veremos como crear objetos complejos y subconsultas en Linq.

20 julio, 2020 / Cursos / .net core, c# .net, curso de c#, curso gratis, deserializar, dotnet, dotnetcore, excepciones, fundamentos c#, generics, herencia, json, polimorfismo, POO, Programación Orientada a Objetos, serializar, sobrecarga, sobreescritura, visual studio / Deja un comentario

PÁGINA 1

¿ME DAS PARA UNA CERVEZA?

Donar

CATEGORÍAS

- .Net
- .Net Core
- 8 curiosidades
- AForge
- Algoritmos
- Angular
- API Google Maps
- Aprende a programar desde cero (SIN CENSURA)
- Arduino
- Arduino
- Arquitectura de software
- ASP .NET MVC
- Autómata celular
- Base de datos de sistemas reales
- Bases de Datos
- Blade
- Blog
- C
- C
- C#

- Chrome
- Contribuciones
- Crear la recuperación de contraseña por correo electrónico en C# MVC .Net
- Crud
- CSS3
- Curso Básico de Javascript
- Curso Básico de Jquery
- Curso cerradura eléctrica con .Net y Arduino
- Curso complemento para recepción de pagos 1.0 en C# .Net
- Curso de arquitectura de software
- Curso de C#
- Curso de facturació electrónica Colombia en C# .Net
- Curso de programación orientada a objetos en C# .Net
- Curso de python básico
- Curso de SQL Server nivel Intermedio
- Curso de web scraping en C# .Net
- Curso gratis de Angular y C# .Net Core
- Curso gratis de desarrollo de aplicaciones con IONIC
- Curso gratuito de C# 8 .Net
- Curso para crear sistema para facturación electrónica web gratis en C# .Net
- Curso para programar el videojuego Snake en C# .Net
- Curso para realizar pagos en línea con php
- Curso Programación Orientada a Objetos POO
- Cursos
- Descargas
- Diagramas
- DIAN
- Digital Persona
- Documentales
- Drivers
- Ejemplos en sistemas reales de patrones de diseño
- Eloquent
- English posts
- Entity Framework
- EntityFramework.Extended
- Errores
- Estructura de datos
- Estructura de datos en C#
- Facturación electrónica
- Fractales
- Frameworks – APIs
- gratis
- Hacking
- Hardware
- HTML
- HTML5

- IDE
- Inteligencia Artificial
- IONIC
- ITextSharp
- Java
- Javascript
- Jquery
- Kendo
- La opinión de un desarrollador de software
- Laravel 5
- Lenguajes
- Lenguajes de Programación
- Libros
- LINQ
- Magick.Net
- Mis productos
- Mis programas
- MVC .Net
- MVC Api .Net
- MvcSiteMapProvider
- mysql
- MySqlBackup.Net
- Navegadores
- Open Source
- Patrones de diseño en C#
- Patrones de diseño en C# .Net
- Php
- PHP
- PHPMailer
- Podcasts
- Preguntas y respuestas de programación
- Programación de videojuegos
- Programación del videojuego Snake en Javascript
- Programando un sistema de ventas real
- Promociones
- Python
- Razor
- Razor
- RazorEngine
- Redes Neuronales
- Relatos de terror de programadores
- Roadmap
- Rotativa
- SAT
- SautinSoft.PdfFocus
- ScrapySharp

- Sheets API Google
- SignalR
- Sin categoría
- Sistema para reclutamiento en Línea
- SOAP
- Software escolar
- Software para gimnasio hdeleon
- Spire.Pdf
- Spreadsheet light
- SpreadSheetLight
- SQL
- Sql Server
- SQLite
- Telerik
- tesseract
- Transact SQL
- Transmisión en vivo
- Typescript
- Videotutoriales
- Visual Basic .Net
- Visual Basic 6.0
- Visual Studio
- Vue
- vue
- WCF
- Windows form
- Windows Forms
- Windows Presentation Foundation
- Winows Forms
- wkhtmltopdf
- WPF
- XML
- XSD

Sobre mí

Youtube

Contacto

Inglés