

TECHNICAL WEIGHT

CAMILO CASTRO - CAMILO@NINJAS.CL

TEMARIO

TEMAS PRINCIPALES

- ¿Qué es el Technical Weight?
- Estrategias
- Ejemplos



TECHNICAL DEBT

TECHNICAL DEBT

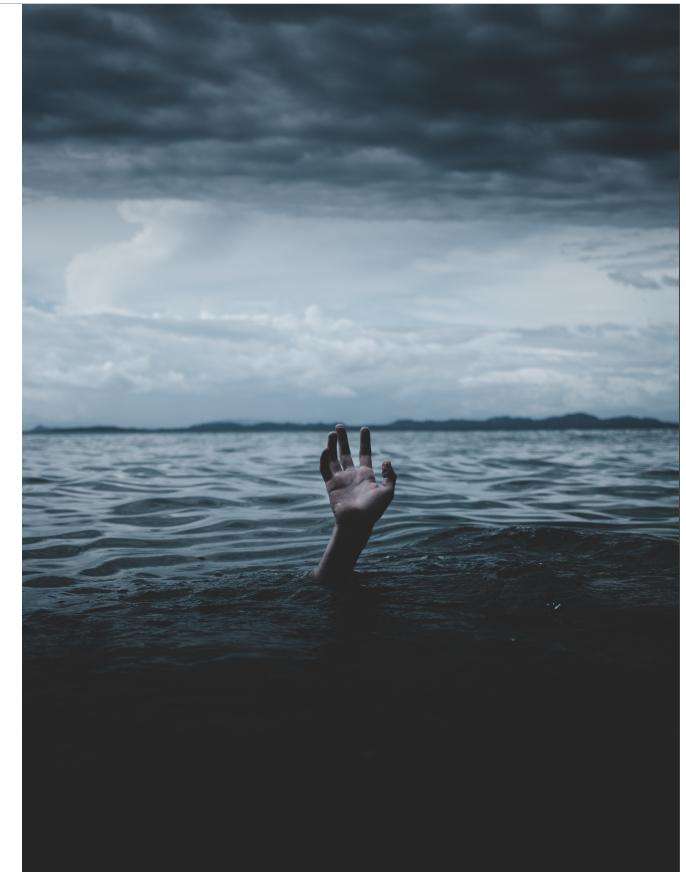
ACUÑADO POR WARD CUNNINGHAM EN 1992 (1)

Javier Garzas: “La deuda técnica es el coste y los intereses a pagar por hacer mal las cosas. El sobre esfuerzo a pagar para mantener un producto software mal hecho, y lo que conlleva, como el coste de la mala imagen frente a los clientes, etc.”. (2)

[1] <http://c2.com/doc/oopsia92.html>

[2] <https://www.javiergarzas.com/2012/11/deuda-tecnica-2.html>

CAMILO CASTRO - NINJAS.CL





TECHNICAL WEIGHT

TECHNICAL WEIGHT

ACUÑADO POR BART WRONSKI EN 2016 (1)

El peso técnico es una característica de una solución que la puede tornar costosa y "pesada" de mantener a mediano y largo plazo. Incluso en ambientes limpios y apropiadamente diseñados (con baja deuda técnica).

[1] <https://bartwronski.com/2016/06/26/technical-weight/>

CAMILO CASTRO - NINJAS.CL



ANALOGÍAS

FERRARI

- Costo circulación elevado (impuesto al lujo).
- Costo de reparación y repuestos elevado (Difíciles de encontrar).

CAMILO CASTRO - NINJAS.CL



ANALOGÍAS

EQUIPAJE

- Si realizas un viaje. ¿Cuánto peso puedes llevar en tu mochila?.
- ¿Usarás todo lo guardado en tu equipaje?.
- Puedes deshacerte del peso extra en el camino, pero ¿Se justifica la inversión inicial?.

CAMILO CASTRO - NINJAS.CL

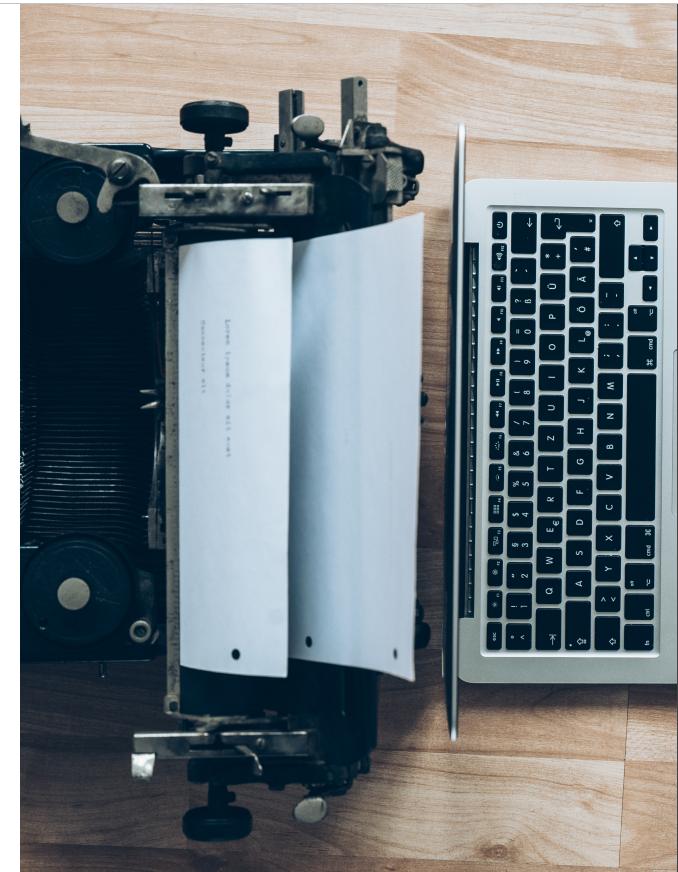


T. WEIGHT VS T. DEBT

RELACIÓN

- Un sistema con alto peso técnico puede tener alta deuda técnica.
- Un sistema con baja deuda técnica puede tener alto peso técnico.
- Toda decisión técnica tiene un peso.

CAMILO CASTRO - NINJAS.CL



1. Camilo Castro

12 de julio de 2020, 02:10:36

Mano de obra = Mientras más peso tenga el sistema, más personas son necesarias para levantar su peso.

Sesgo de confirmación = Solamente se ven las cosas positivas. No se encuentra nada negativo.

Aumento del compromiso = Como ya he invertido "X" dinero o "Y" tiempo, conviene seguir invirtiendo recursos a pesar de que no represente ningún beneficio hacerlo.

Trampa del progreso = Solamente hay progreso si el sistema tiene nuevas características. No se ve valor en "ordenar" y mejorar lo ya existente sin agregar nuevas características.

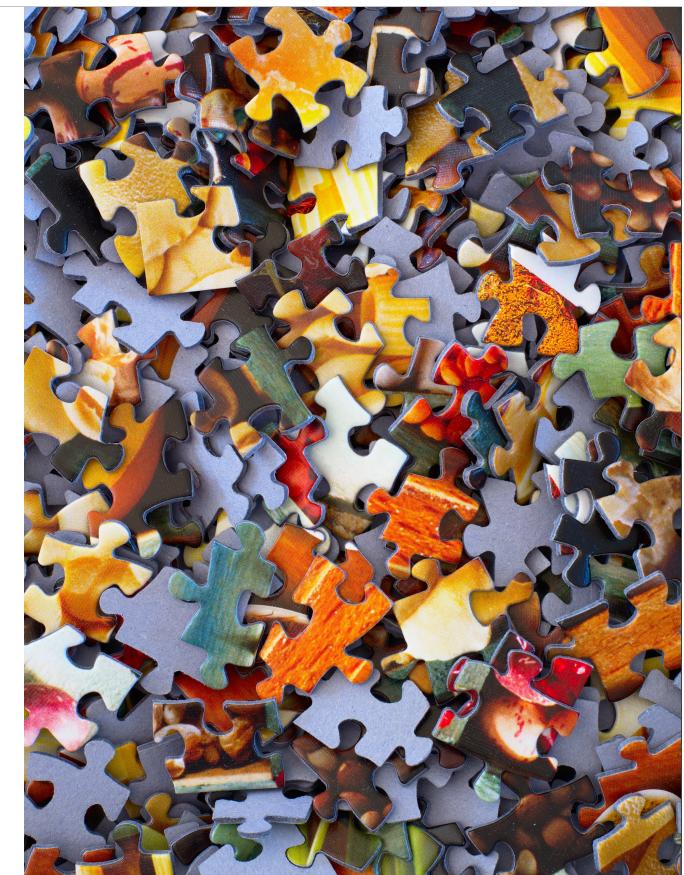
Aversión a la pérdida = Se concentra en evitar las fallas. Sin la evaluar si el sistema es el problema y no las causas externas. Normalmente pasa en sistemas "Legacy" que podrían ser eliminados o reemplazados por sistemas más simples.

¿POR QUÉ ES BUENO CONTROLARLO?

1

- Mano de obra necesaria para mantener el sistema.
- Dificultad de encontrar nuevos devs.
- Costos de reparación de bugs.
- Refactorización complicada.
- Extensión complicada.
- Prejuicios:
 - Sesgo de confirmación
 - Aumento del compromiso
 - Trampa del progreso
 - Aversión a la pérdida

CAMILO CASTRO - NINJAS.CL



¿POR QUÉ ES BUENO CONTROLARLO?

1

- Douglas Hubbard, en su libro “How to measure anything”, define el riesgo como “un estado de incertidumbre donde algunas de las posibilidades pueden implicar pérdidas, catástrofes u otros resultados no deseados”.

CAMILO CASTRO - NINJAS.CL



1. Camilo Castro

12 de julio de 2020, 01:28:12

Usando la composición de Kleisli simplificamos el código, evitando redundancia. Sin embargo requiere de los desarrolladores aprender sobre composición monádica para entender bien. Lo que aumenta el peso técnico a pesar de que disminuye la deuda técnica.

EJEMPLO

[HTTPS://WWW.NICOLAS-SCHURMANN.COM/KLEISLI/](https://www.nicolas-schurmann.com/kleisli/)

```
1 const composeMap = (...ms) =>
2   ms.reduce((f, g) => x => g(x).map(f))
3
4 const comp = composeMap(
5   funcion1,
6   funcion2,
7 )
8
9 const comp2 = x => x.map(funcion1).map(funcion2)
10
11 comp === comp2 // en este caso, el resultado es el mismo
```



EJEMPLO

¿POR QUÉ NO SIEMPRE ES BUENO USAR ALGO QUE CLARAMENTE ES SUPERIOR?

- Las soluciones adoptadas dependerán de condiciones de cada proyecto. Lo que pudo servir para uno, puede ser muy “pesado” de utilizar en otro.
- Se debe considerar no solo la implementación inicial, también los costos de mantenimiento y extensión.
- ¿Cuántas personas pueden entender cómo funciona el sistema?, ¿Qué tan simple es de aprender?.



EJEMPLO: SISTEMA DE TURNOS

<https://github.com/ninjascl/screensharer>

1. Camilo Castro

12 de julio de 2020, 01:30:32

La mayoría de los sistemas de turno requieren de soluciones complejas o dependencia de actores externos que ofrecen servicios. La idea de esta solución es concentrarse en el desafío focal (Root Cause Analysis).

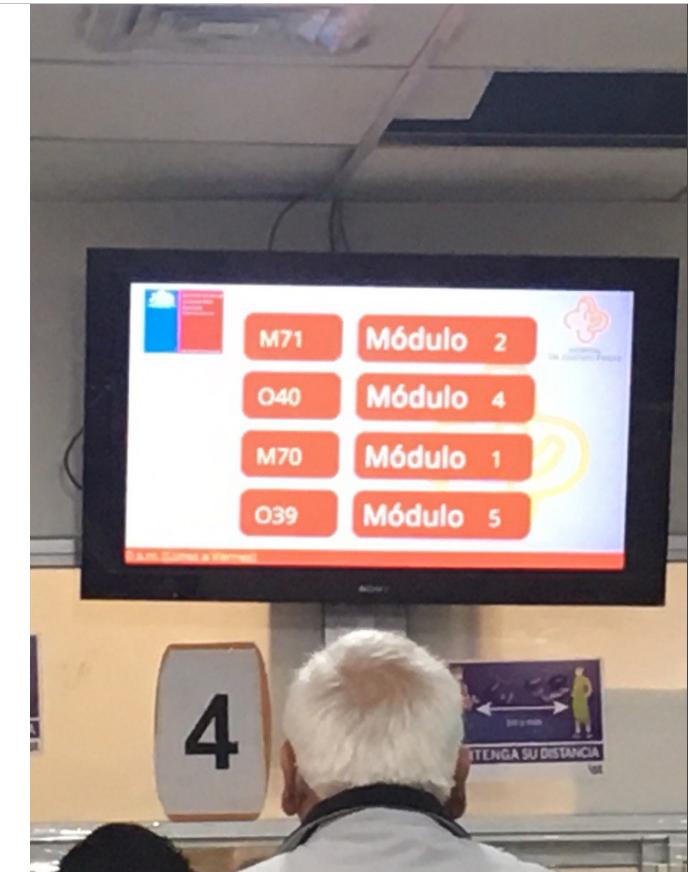
Implementando un sistema utilizando los componentes ya disponibles. (Computadores y Asistentes de Atención).

SISTEMA DE TURNOS

PROBLEMAS

- 1 ● Información offline.
- Necesidad de asistir físicamente y esperar por largos períodos.
- Riesgo de contagio de enfermedades.

CAMILO CASTRO - NINJAS.CL

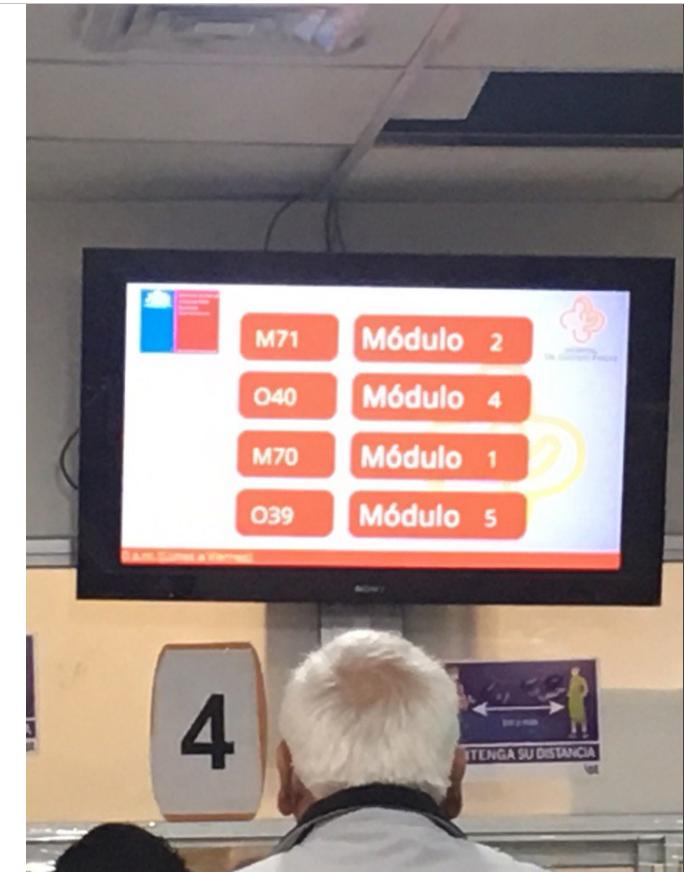


SISTEMA DE TURNOS

DESAFÍO FOCAL

- Personas obligadas a esperar en el sitio para saber su turno.

CAMILO CASTRO - NINJAS.CL



SISTEMA DE TURNOS

POSIBLES SOLUCIONES (CON POCO PESO TÉCNICO)

- Compartir pantalla de turnos en página web.
- Persona encargada de llamar e informar por teléfono.

CAMILO CASTRO - NINJAS.CL



SISTEMA DE TURNOS

COMPARTIR PANTALLA EN PÁGINA WEB

- Se saca un screenshot cada x segundos del computador usado en la sala de espera.
- El archivo se sube a un servidor web o almacenamiento externo (Firebase, S3, FTP).
- La página web se refresca automáticamente cada x segundos mostrando el último screenshot disponible.

CAMILO CASTRO - NINJAS.CL



SISTEMA DE TURNOS

PERSONA ENCARGADA DE LLAMAR E INFORMAR POR TELÉFONO.

- Persona encargada puede ser contactada por SMS/Whatsapp/Llamada.
- Saca turno manualmente con los datos del paciente.
- Llama o envía mensaje al paciente cuando queden 20 números para llegar al indicado.
- Pensado para personas sin acceso a internet o dificultades técnicas (abuelitos).

CAMILO CASTRO - NINJAS.CL



SISTEMA DE TURNOS

RESULTADO DE LAS SOLUCIONES

- La información está disponible online.
- Se reduce el tiempo necesario de estar presentes. (Solamente se requiere un tiempo mínimo para realizar el trámite mismo).
- Al requerir menor tiempo presencial, se reduce el riesgo de enfermarse (menor aglomeración de gente).

CAMILO CASTRO - NINJAS.CL



1. Camilo Castro

12 de julio de 2020, 01:22:52

Problema difícil = Existen problemas cuya dificultad entregará siempre un peso técnico considerable. (Problemas matemáticos, sistemas con muy pocos expertos disponibles).

Acotación del Alcance = Las expectativas del cliente no han sido correctamente gestionadas y constantemente se ingresan nuevas características o "mejoras".

Funcionalidad Futura = Los desarrolladores piensan en dar solución a problemas que aún no se han presentado.

Implementando características innecesarias o abstracciones muy complejas.

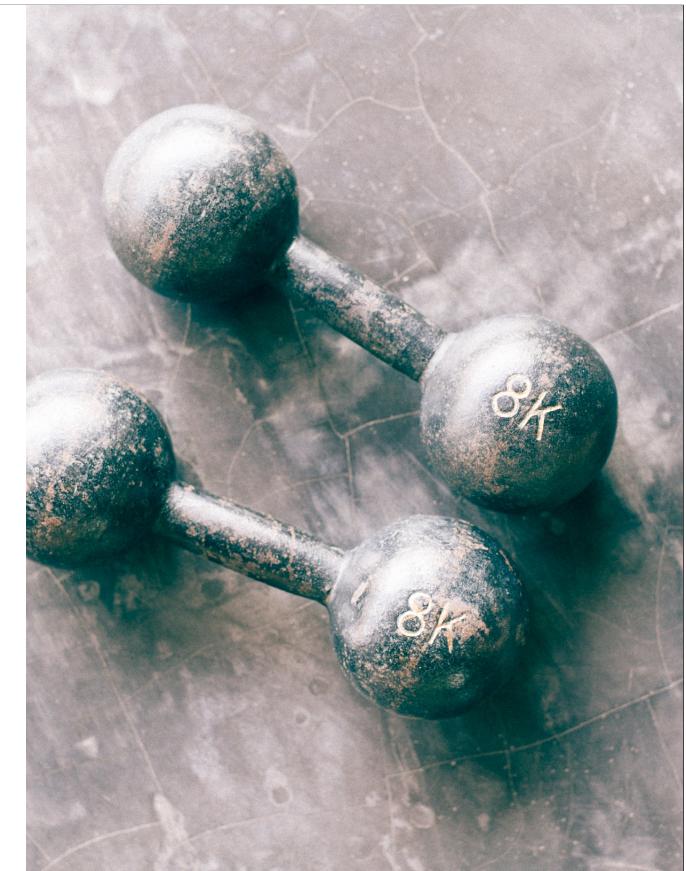
Comprometerse = Al usar tecnología "X" ahorramos un 5% de "Y". Pero significa aumentar la complejidad del sistema en un "Z"%. ¿Qué es prioridad?, ¿Vale la pena?. A veces es mejor tomar una decisión aunque existan ciertas desventajas. ¿Cómo podemos afrontarlas?.

¿POR QUÉ SE GENERA?

1

- Problema difícil.
- Sobre dimensionar un problema.
- Poca acotación del alcance.
- Funcionalidad futura (ANTI KISS).
- No dispuesto a comprometerse.
- Poca experiencia.

CAMILO CASTRO - NINJAS.CL



1. Camilo Castro

12 de julio de 2020, 01:18:36

Dogmatismo Técnico = Creer que una solución, práctica, recomendación, herramienta, lenguaje de programación, framework, patrón de diseño o similar. Es siempre la mejor alternativa. Evaluar si las opiniones y decisiones están basadas en creencias o evidencias. Priorizar simpleza y comprensión por sobre cumplimiento ciego.

Ser Ingenioso = Usar soluciones complejas o poco comprensibles solo para aparentar o "apropiarse" de un área.

Carrera Individual = Es recomendable probar cosas nuevas para avanzar en la carrera, pero siempre tener en cuenta cómo afectará al proyecto y a los demás colaboradores.

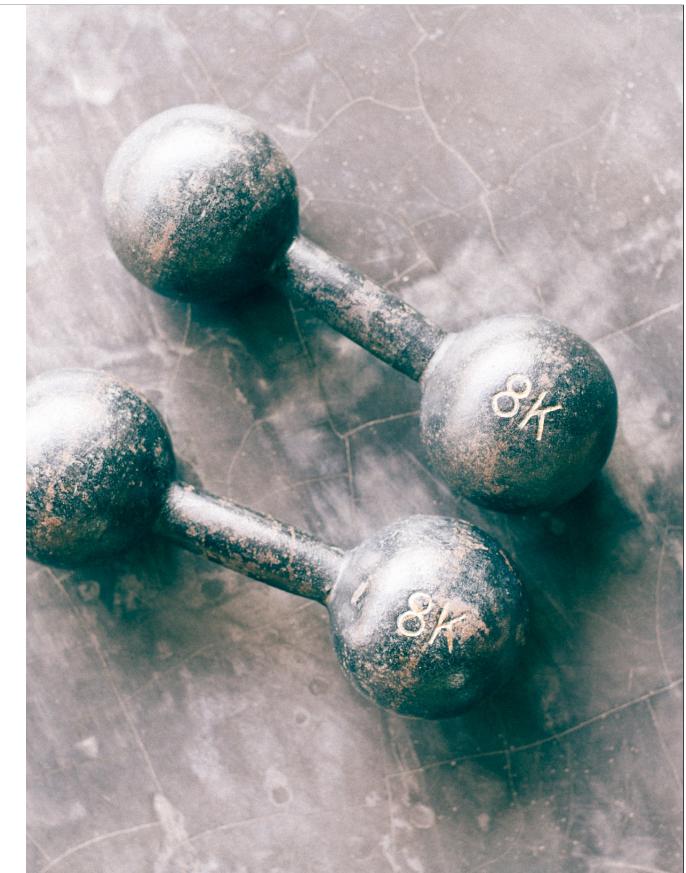
Carrera Organizacional = Una organización que solo promueve a personas que siempre crean nuevas "features" o "proyectos" y son "dueños exclusivos" de un producto o conocimiento. Poca prioridad al mantenimiento a largo plazo. Se fomenta el "aserruchar" el piso.

¿POR QUÉ SE GENERA?

1

- Dogmatismo técnico.
- Priorizar lo novedoso o "divertido" por sobre lo conocido y estable.
- Ser "ingenioso".
- Progreso de carrera individual.
- Progreso de carrera organizacional.

CAMILO CASTRO - NINJAS.CL



ESTRATEGIAS

1. Camilo Castro

12 de julio de 2020, 12:19:19

La primera estrategia es saber ¿Dónde estamos?. ¿Cuál es la situación actual?. Para esto tenemos las herramientas para calcular la dificultad usadas en ciencias de la computación. Como Big Omicron (Big O). La cual mide tiempo de ejecución (*runtime*) y espacio o memoria (*space*) utilizada por un algoritmo. Podría también ser usada para medir peso (*Weight*).

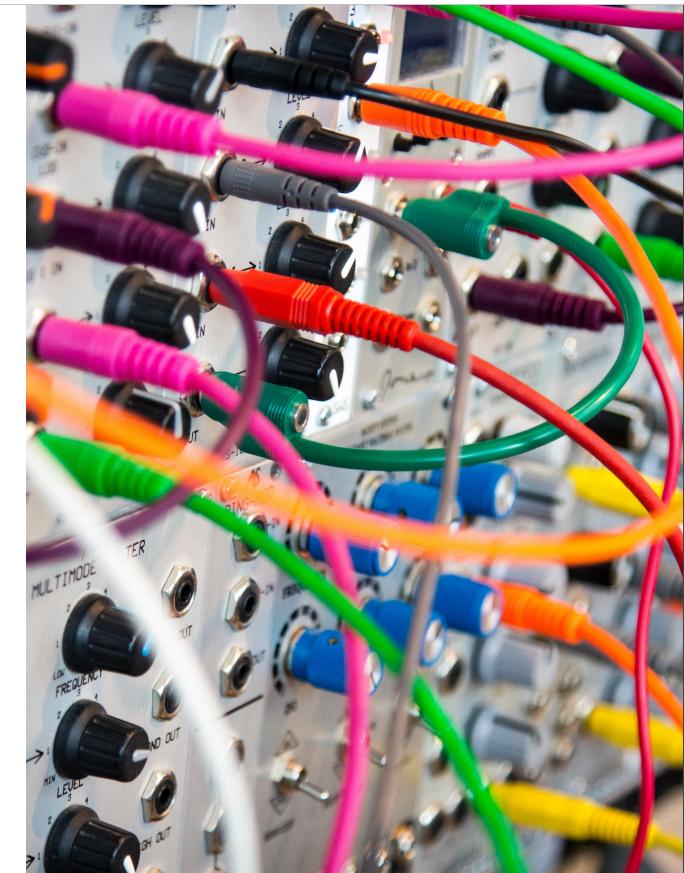
DEFINIR PESO (COMPLEJIDAD)

DONALD E. KNUTH: [HTTPS://DL.ACM.ORG/DOI/10.1145/1008328.1008329](https://dl.acm.org/doi/10.1145/1008328.1008329)

- 1 ● **Big Omicron $O(f(n))$** : Como máximo $f(n)$.
- **Big Omega $\Omega(f(n))$** : Al menos $f(n)$.
- **Big Theta $\Theta(f(n))$** : Exactamente $f(n)$.

- Mide complejidad de ejecución (*runtime*).
- Mide complejidad de espacio (*space*).
- Podría medir complejidad técnica (*weight*).
- **¿Cuál es tu $O(1)$?**

CAMILO CASTRO - NINJAS.CL



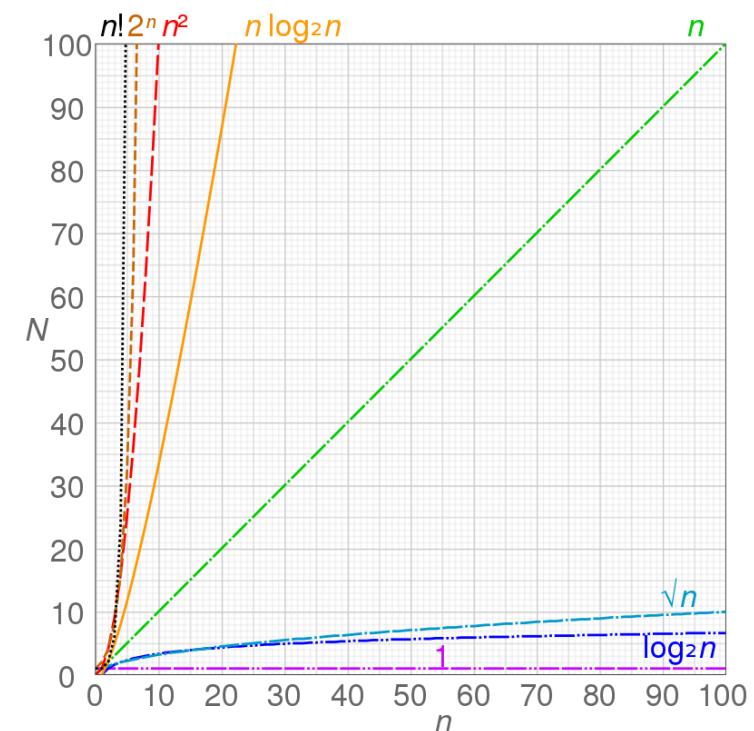
DEFINIR PESO (COMPLEJIDAD)

[HTTPS://EN.WIKIPEDIA.ORG/WIKI/BIG_O_NOTATION](https://en.wikipedia.org/wiki/BIG_O_NOTATION)

1

- $O(1)$ = El mejor escenario.
- $O(\log^2 n)$
- $O(\sqrt{n})$
- $O(n)$ = Escenario promedio.
- $O(n \log^2 n)$
- $O(n^2)$
- $O(2^n)$
- $O(n!)$ = El peor escenario.

CAMILO CASTRO - NINJAS.CL



1. Camilo Castro

12 de julio de 2020, 12:22:57

Para calcular el peso, también puede ser útil saber ¿Qué tanta libertad existe en el sistema?. Mientras más libertad, menor orden. Libertad se entiende cómo la capacidad de realizar cambios (agregar nuevos componentes, lenguajes, personas, estructuras, etc.) y desarrollar comportamientos o propiedades no contemplados inicialmente.

¿Cuándo es mejor tener mayor libertad?, ¿En qué contexto es mejor tener mayor orden?.

Existen estructuras para gestionar la tendencia del sistema hacia la entropía?.

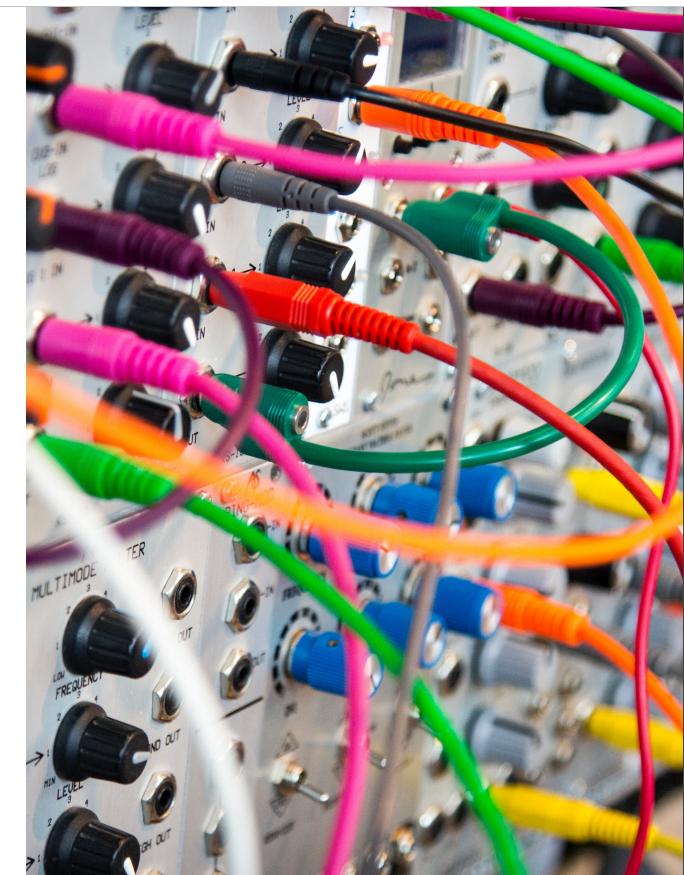
DEFINIR PESO (COMPLEJIDAD)

[HTTPS://ES.WIKIPEDIA.ORG/WIKI/ENTROP%C3%ADA](https://es.wikipedia.org/wiki/Entrop%C3%ADa)

1

- Considerar la Entropía. (Nivel de libertad). Mayor orden = menor libertad.
- La cantidad de entropía de un sistema es proporcional al logaritmo natural del número de microestados posibles ($S = k \ln \Omega$).
- El tiempo pasa y la entropía crece hasta alcanzar el punto de máxima entropía del sistema.

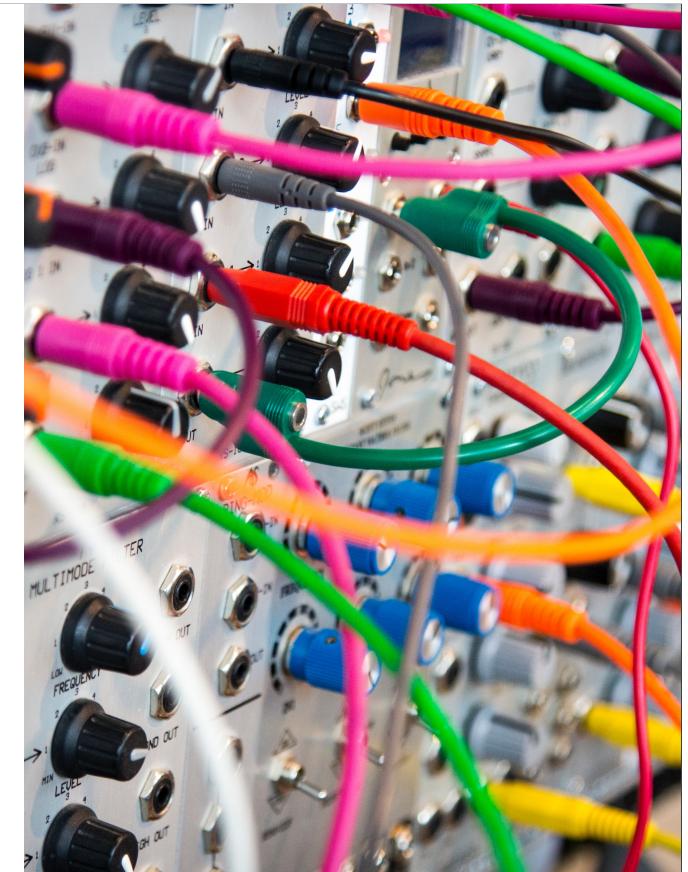
CAMILO CASTRO - NINJAS.CL



DEFINIR PESO (COMPLEJIDAD)

- **¿Cuánto peso técnico tienes?.**
- **¿Por cuánto tiempo lo piensas sostener?.**
- **¿Podrás continuar sosteniéndolo si las personas se van del equipo?.**
- **¿Realmente se aprovechan todas las características del producto?.**

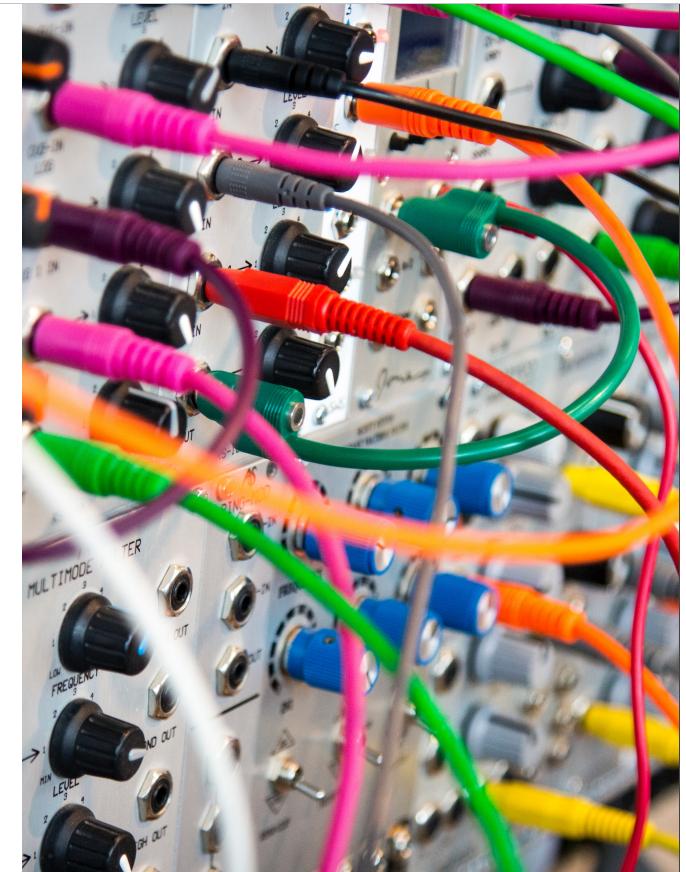
CAMILO CASTRO - NINJAS.CL



DEFINIR PESO (COMPLEJIDAD)

- ¿Tratas de ser “ingenioso” e impresionar?
- ¿Qué desventajas hay en usar soluciones más ligeras?. ¿Existe algún prejuicio?.
- ¿Qué alternativas existen?.

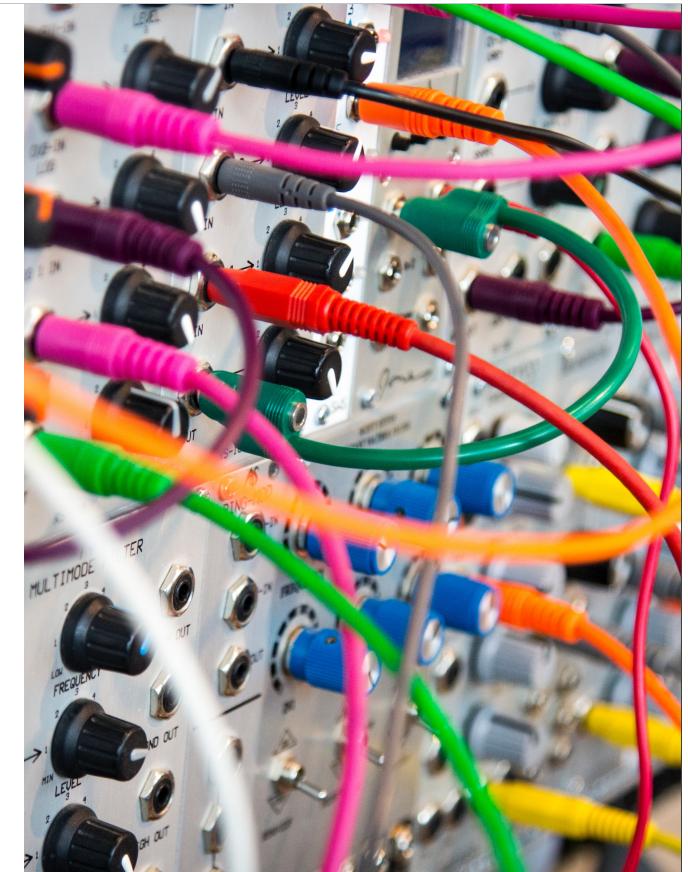
CAMILO CASTRO - NINJAS.CL



DEFINIR PESO (COMPLEJIDAD)

- Medir apoyándose en personas con experiencia y comparar estimaciones.
- Medir la cantidad de personas disponibles en el mercado laboral. ¿Qué tan fácil o costoso es encontrar personas que deseen trabajar con el stack tecnológico seleccionado?, ¿Qué tan fácil o costosa es su capacitación?.

CAMILO CASTRO - NINJAS.CL



1. Camilo Castro

12 de julio de 2020, 12:31:38

Elegir herramientas solo por su capacidad de “ser mejor que X para resolver Y”. Puede causar mayor dispersión y complejidad. Hay que analizar los “trade off” y determinar si realmente se necesita esa herramienta, o se puede utilizar algo más conocido y con menor riesgo (se sabe los pro y contra, es más manejable).

Por ejemplo tener un sistema que realiza tareas similares. Uno escrito en Python, otro en Ruby y otro en JavaScript. Lo ideal es estandarizar el lenguaje y utilizarlo apropiadamente en casos de uso similares.

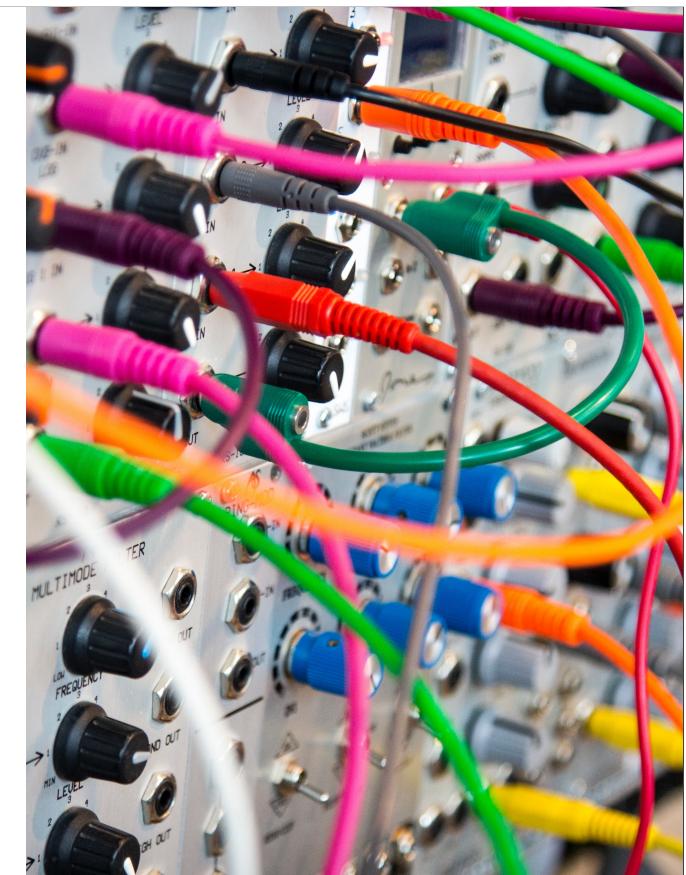
No por que una herramienta pueda resolver un problema quiere decir que sea la idónea para el contexto del proyecto. ¿Cómo evolucionará la complejidad de esa nueva herramienta?. ¿Cómo podemos solucionar los problemas usando las herramientas existentes?.

DEFINIR PESO (COMPLEJIDAD)

1

- Distintas organizaciones pueden soportar distinto peso.
- Lo que funcionó para una organización pequeña, puede no servir para una organización más grande y viceversa.
- “La herramienta adecuada, para el problema adecuado” es una falacia.
- Una ecuación general para definir peso no es posible debido a la naturaleza cambiante de proyectos y organizaciones. Solo es posible una aproximación.

CAMILO CASTRO - NINJAS.CL



1. Camilo Castro

12 de julio de 2020, 12:47:12

Un sistema Legacy se puede mejorar con mayor documentación y personas que saben cómo funciona (tal vez mejorando la calidad de código, refactorización, uso de herramientas de apoyo).

También se puede simplificar eliminando características poco usadas (Usar telemetría y pruebas A/B para determinar esto).

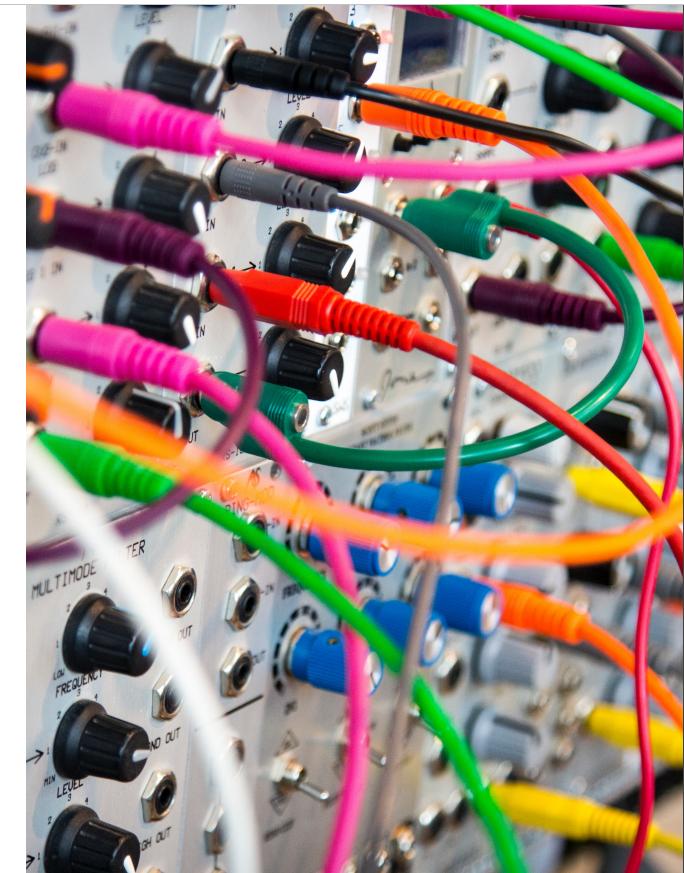
Desecharlos debe ser una opción si existe una solución más simple o si el problema inicial ha mutado y el sistema Legacy no lo está resolviendo adecuadamente.

DEFINIR PESO (COMPLEJIDAD)

1

- Evaluar sistemas “legacy”. Mejorarlos, simplificarlos o desecharlos.
- Todo cambio debe ser estudiado, ser gradual y ser aprobado por el equipo involucrado y Stakeholders.
- «Apenas se puede negar que el objetivo supremo de toda teoría es hacer los elementos básicos tan simples y tan poco numerosos como sea posible, sin renunciar a la representación adecuada de un simple dato» - Albert Einstein

CAMILO CASTRO - NINJAS.CL

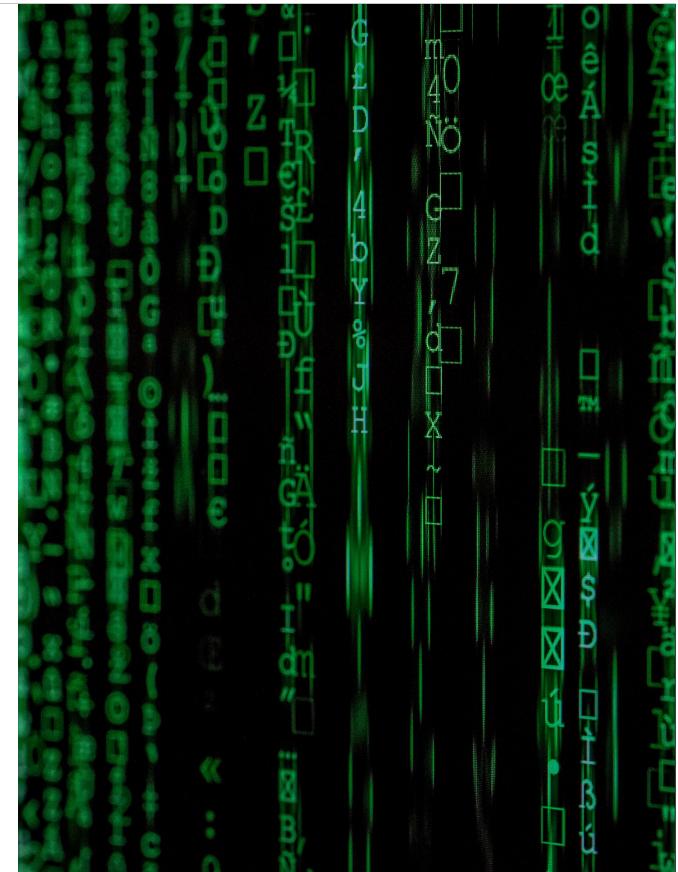


USAR TECNOLOGÍA CONOCIDA

DAN MCKINLEY: [HTTP://BORINGTECHNOLOGY.CLUB/](http://BORINGTECHNOLOGY.CLUB/)

- Permite conocer fortalezas y debilidades de cada solución.
- Mayor cantidad de acervo documental.
- Usar pruebas de concepto, prototipos y MVP para tecnologías más experimentales.
- Usar tecnologías conocidas y probadas para proyectos que requieran mayor control de riesgos. (Se puede usar tecnologías más nuevas, siempre que se conozcan bien).
- Buscar utilizar la menor cantidad de elementos posibles.
¿Cómo puedo lograrlo utilizando lo que ya tengo disponible, sin agregar nuevos componentes?.

CAMILO CASTRO - NINJAS.CL



ENCONTRAR EL DESAFÍO FOCAL

HBR: [HTTPS://HBR.ORG/2015/12/FIND-INNOVATION-WHERE-YOU-LEAST-EXPECT-IT](https://hbr.org/2015/12/find-innovation-where-you-least-expect-it)

- ¿Dónde está el problema raíz?.
- ¿Puede un problema ser usado como su propia solución?.
- Brainstorming “mudo”. Utilizando tarjetas y texto en vez de voz. Da más oportunidades a personas tímidas.
- Root Cause Analysis.

CAMILO CASTRO - NINJAS.CL



1. Camilo Castro

12 de julio de 2020, 13:02:06

Gracias a Raúl de BeerJS Valdivia,
Chile por la oportunidad de ser
expositor.

camilo@ninjas.cl

<https://ninjacl>

CRÉDITOS

FOTOGRAFÍAS USADAS

1 <https://unsplash.com/photos/rX12B5uX7QM> Photo by Ian Espinosa on Unsplash

<https://unsplash.com/photos/Yuv-iwByVRQ> Photo by Victor Freitas on Unsplash

<https://unsplash.com/photos/SPbcqTVoYqE> Photo by Matt Lamers on Unsplash

<https://unsplash.com/photos/oNDRCGrqaYc> Photo by Karim MANJRA on Unsplash

<https://unsplash.com/photos/1F4MukOOUNg> Photo by Glenn Carstens-Peters on Unsplash

<https://unsplash.com/photos/3y1zF4hIPCg> Photo by Hans-Peter Gauster on Unsplash

<https://unsplash.com/photos/TsVN31Dzyv4> Photo by Cyril Saulnier on Unsplash

<https://unsplash.com/photos/I090uFWoPal> Photo by John Barkiple on Unsplash

https://unsplash.com/photos/DX3_dXuHVi8 Photo by Immo Wegmann on Unsplash

<https://unsplash.com/photos/m2TU2gfqSeE> Photo by You X Ventures on Unsplash

CAMILO CASTRO - NINJAS.CL