Metodologías y Estándares

En una cadena de trabajo, donde la información viaja entre distintos actores, hay un riesgo serio de que la información sufra distorsiones, que al final del camino generen un desvío extraordinario.

Algunas fallas que se pueden evitar o disminuir son:

- Falta de coordinación de recursos y actividades.
- Productos finales que no representan lo que el cliente esperaba.
- Proyectos que terminan fuera de tiempo y consumen más dinero del planificado.
- Planificación inadecuada de recursos y actividades.
- Falta de conocimiento del estado real del proyecto.
- Alcance del proyecto poco definido y falta de gestión de los cambios al mismo.

Tales fallas se pueden tratar mediante distintas metodologías de gestión (estándares), como la PMI (de EEUU) o la PRINCE2 (de Europa; más adecuada para proyectos grandes y complejos).

PROYECTO → esfuerzo temporal comprometido con la creación de un producto/servicio de resultado único.

Características de un Proyecto:

- Son temporales → tienen un principio y un fin definidos.
    - El proyecto termina cuando: (1) ha alcanzado sus objetivos; o bien (2) resulta claro que no pueden alcanzarse; o bien (3) la necesidad del proyecto ya no existe.
    - Las actividades de mantenimiento realizadas una vez entregado el producto/servicio pueden ser tareas pequeñas o grandes, estas últimas son a su vez otro proyecto.
- Son de elaboración progresiva → avanzan en pasos e incrementos continuos.

Fases del Proyecto:

1. Pre-Proyecto → se revisa la idea a fin de proveer la información necesaria para decidir si se comienza el proyecto, determinando factibilidades técnicas y económicas.
2. Inicio → se trata de que todos los involucrados entiendan qué producirá el proyecto, cuándo, con qué costos, con qué calidad, elaborando un plan que permita realizarlo.
3. Ejecución → se realiza el trabajo *per se*, se controla el flujo de trabajo de los equipos, se gestionan riesgos y problemas y se informan los avances del proyecto.
4. Cierre → se comprueba que todo se haya realizado y se informa cómo finalizó el proyecto.

Ciclo PDCA (*Plan-Do-Check-Act*):

- *Plan* → identificar y analizar el problema.
- *Do* → elaborar e implementar una solución.
- *Check* → evaluar los resultados.
- *Act* → aplicar acciones correctivas si es necesario. Finalmente, si el resultado fue exitoso, estandarizar la solución y capitalizarla en nuevas oportunidades.

Gestión del Proyecto → Talle Único vs. Talle A Medida:

En términos generales, siempre conviene manejarse con algunos aspectos estándar generales y, después, como todo proyecto es único, cada uno tendrá sus particularidades:

	TALLE ÚNICO	**TALLE A MEDIDA**
Objetivo	Triple restricción: alcance-costo-tiempo, manteniendo la calidad.	Resultados al negocio, cumpliendo múltiples criterios.
Planificación	Se planifica una vez al inicio.	Plan inicial y replanificación cuando es necesario.
Enfoque de Dirección	Rígido, enfocado en el plan inicial.	Flexible, cambiante, adaptativo.
Trabajo del Proyecto	Predecible, conocido, lineal, simple.	Impredecible, incierto, no lineal, complejo.
Control del Proyecto	Busca desvíos respecto al plan y toma acciones para alinearlo.	Identifica cambios en el entorno y ajusta el plan de acuerdo al entorno.
Metodología	Todos los proyectos siguen la misma metodología.	Adaptada a la complejidad e incertidumbre del proyecto.

RIESGO → evento posible que, si sucede, tiene un efecto sobre los objetivos del proyecto.

- Están presentes en todos los proyectos.
- Pueden ser negativos o positivos.
  - Si son positivos, a los riesgos se los conoce como **oportunidades**.
- Los riesgos pueden ser conocidos o desconocidos:
  - Riesgos conocidos → aquellos que identificamos y analizamos y para los cuales podemos planificar respuestas.
  - Riesgos desconocidos → aquellos que no se pueden gestionar de manera proactiva, pero para lo cual debemos crear un plan de contingencia.

Atributos del Riesgo:

- Probabilidad de ocurrencia → posibilidad de que el riesgo se materialice.
- Impacto → resultado de la materialización del riesgo.
- Severidad → producto de la probabilidad por el impacto.
  - Una vez determinada la severidad de cada riesgo, se debe priorizar (de acuerdo a cierto criterio) cuáles deben gestionarse primero

Gestión del Riesgo → si no es posible evitar su aparición, será posible gestionarlo mediante el siguiente proceso:

1. Identificación → reconocimiento de las fuentes de riesgo y sus consecuencias potenciales.
2. Análisis → determinación de la necesidad de tratamiento del riesgo y la prioridad de su implementación.
3. Tratamiento o Respuesta → selección de opciones para actuar:
    - Evitar → la amenaza se elimina por completo, asegurando que no podrá ocurrir o que no tendrá efecto en el proyecto.
    - Transferir → se traslada el impacto negativo de una amenaza a un tercero, confiriéndole la responsabilidad de su gestión.
    - Mitigar → reducir la probabilidad de ocurrencia y/o el impacto del riesgo a un umbral aceptable.
    - Aceptar o Asumir → se asume que el riesgo se manifestará y se debe estar en constante monitoreo de la situación, aunque sin tomar acción.
4. Monitoreo y Revisión → evaluación del progreso en la implementación del tratamiento. En esta instancia pueden aparecer nuevos riesgos.

PROBLEMA → evento, esperado o no, que afecta negativamente los objetivos de un proyecto.

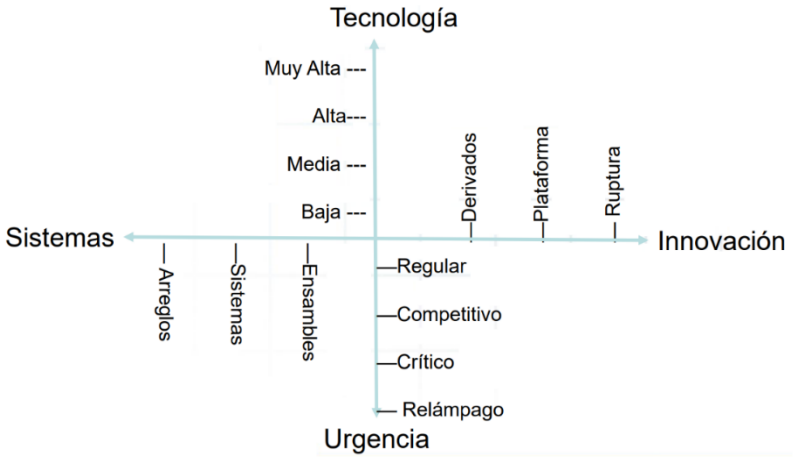
- Un **riesgo** es algo posible → un **problema** es un hecho que requiere acción.
- Un problema esperado puede ser un riesgo que se materializa.

Seguimiento y Control → "*no se puede controlar lo que no se puede medir*":

- Un proyecto es exitoso cuando contribuye al éxito de la organización.
- Es necesario realizar mediciones para determinar su grado de avance. Comparando el grado de avance medido con el avance esperado, verificaremos si se observan desvíos significativos respecto de lo planificado. Si ése resultara el caso, deben tomarse acciones correctivas.

Éxito y Complejidad de Proyectos – Enfoque de Diamante → para evaluar la complejidad de los proyectos se definen 4 dimensiones y cada factor se representa en un eje:

- Innovación → mide cuán nuevo es el producto/servicio para los clientes, para los usuarios o para el mercado en general.
- Tecnología → mide la cantidad requerida de nuevas tecnologías.
- Sistemas Involucrados → mide la complejidad del producto/servicio, la tarea y la organización del proyecto.
- Urgencia → mide cuánto tiempo hay para completar el trabajo.



ALCANCE → definición exacta y unívoca de todo lo que estará (y lo que no) comprendido dentro del proyecto a ejecutar, proporcionando un entendimiento común entre los interesados del mismo.

- Alcance del Producto → funciones y características que describen un producto/servicio.
- Alcance del Proyecto → trabajo que debe realizarse para entregar un producto de acuerdo de acuerdo con el alcance del producto.

REQUERIMIENTOS → cuestiones que los proyectos van a satisfacer.

- Un requerimiento es válido si cumple con las siguientes características [IEEE-830]:
  - Necesario → es necesario si su omisión provoca una deficiencia en el producto.
  - Conciso → fácil de leer y entender, de redacción simple y clara.
  - Completo → proporciona la información suficiente para ser comprendido.
  - Consistente → no es contradictorio con otro requerimiento.
  - No ambiguo → tiene una sola interpretación, sin causar confusiones.
  - Verificable → puede ser cuantificado a través de inspección, pruebas y/o análisis.
    - Es fundamental para saber si el requerimiento en cuestión se cumple o no.
- Tipos de Requerimientos:
  - Requerimientos Funcionales → describen qué es lo que el sistema debe hacer, estableciendo las funciones que el producto de software debe incluir.
  - Requerimientos NO Funcionales → restricciones a las que está sometido el producto de software a desarrollar y que influyen sobre el funcionamiento o sobre el proceso de desarrollo de software.

EDT (WDS) · Estructura de Desglose de Trabajo → forma de planificación que consiste en la descomposición jerárquica del trabajo, con el fin de organizar y definir el alcance total del proyecto.

Pasos para desarrollar una planificación:

1. Definir actividades → identificar las acciones específicas a realizar.
2. Secuenciar las actividades → definir su orden.
3. Estimar recursos de las actividades.
4. Estimar la duración de las actividades.
5. Desarrollar el cronograma:
   - Diagrama de Gantt → representación gráfica del cronograma de esas actividades.
   - Enfoques de los avances del proyecto:
     - % Trabajo Completado → avance asociado al tiempo incurrido.
     - % Físico Completado → avance asociado al trabajo real ejecutado.
   - Línea de Base → luego de planificar, se debe definir una línea de base del proyecto como una fotografía del cronograma para que, cuando el proyecto se vaya ejecutando, se pueda comparar el desempeño de la situación actual con la fotografía inicial.

METODOLOGÍA	**TRADICIONAL**	**ÁGIL**
Foco	Focalizada en el cumplimiento de la ejecución de las tareas.	Focalizada en el esfuerzo de planificación y ejecución en aquellos objetivos de corto plazo.
Planificación	A largo plazo (pasos más largos). Las tareas a definir en el plan cubren toda la vida del proyecto. Puede haber entregas intermedias.	A corto plazo (pasos más cortos). Planificación intensa entre 2 y 4 semanas. Hay demostraciones al cliente de los incrementos del producto/servicio.
Cambios en la Planificación	Difíciles de realizar.	Fáciles de realizar.
Retroalimentación y Retrospectivas	Menos frecuente. Única, al final del proyecto.	Más frecuente. Durante todo el proyecto.
Control sobre el producto/servicio	Predictivo.	Empírico, se puede probar.

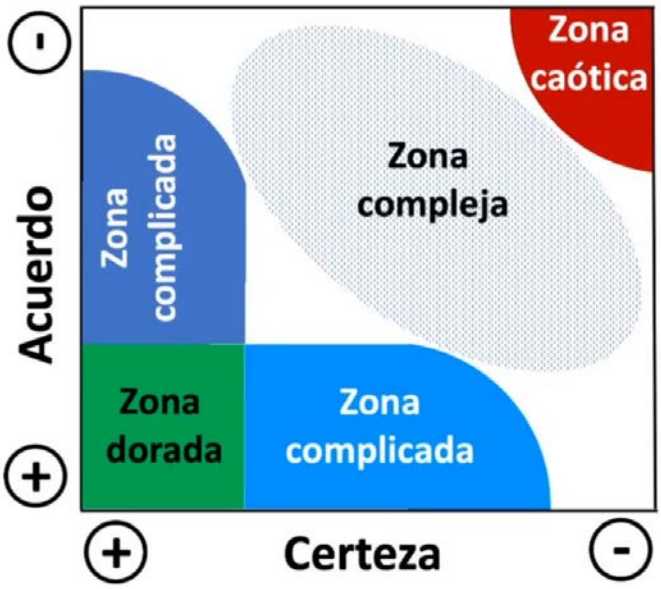
Niveles de Planificación de la Metodología Ágil → se asume un horizonte de incertidumbre a partir del cual no tiene sentido planificar tareas detalladas:

- Nivel Estratégico → planificación de los grandes objetivos del producto.
- Nivel Táctico → planificación referida a las tareas en curso.
- Nivel Operativo → replanificaciones diarias.

Un proyecto puede ser complejo por aspectos tecnológicos, por falta de recursos, por razones de tiempo, etcétera…

Diagrama de Stacey → enfoque de alto nivel que permite analizar la complejidad de los proyectos:

- Plantea 2 dimensiones:
  - Certeza → grado de conocimiento sobre lo que hay que hacer.
  - Acuerdo → grado de acuerdo político entre las partes sobre cómo hay que resolver.
- Con tales dimensiones, aparecen varias zonas:
  - Zona Dorada → zona ideal, donde uno quiere estar siempre.
    - Es recomendable trabajar con una metodología tradicional.
  - Zonas Complicadas → situaciones en las que: (1) hay acuerdo sobre qué hacer pero se sabe menos sobre qué hacer, o bien, (2) se sabe qué hacer pero no hay acuerdo sobre cómo hacerlo.
    - Para mejorar la situación, se puede elegir entre un enfoque ágil o tradicional.
  - Zona Compleja → hay menores certezas y acuerdos.
    - La solución es vía metodología ágil → aplicar pasos cortos y recalcular.
  - Zona Caótica → no hay certezas ni acuerdos.
    - No se puede hacer nada.



***BUSINESS PLAN* · PLAN DE NEGOCIOS** → evaluación económica y financiera sobre cómo va a funcionar una organización, del presente a 1 año o a 5 años.

- Aspecto Económico → se refiere al resultado del ejercicio, es decir, refiere a si una organización gana o pierde dinero.
    - Los ingresos/egresos se contabilizan al comprar/vender independientemente del instante en que se realicen los pagos/cobros de dinero (→ devengado).
- Aspecto Financiero → se relaciona con el momento en que entra/sale dinero de la empresa, es decir, al instante en que se realizan los cobros/pagos de dinero.
    - Los ingresos/egresos se contabilizan cuando se realizan los cobros/pagos de dinero.

Impuestos → pueden afectar a los proyectos de una organización, trasladando costos.

- Impuestos Directos → gravan un conjunto de operaciones → gravan el sujeto.
    - *Ejemplo:* Ganancias.
- Impuestos Indirectos → gravan la transacción comercial → gravan el bien/servicio.
    - *Ejemplos:* IVA, IIBB.

Existen diferentes tipos de impuestos:

1. IVA · Impuesto al Valor Agregado → se grava las compras/ventas devengadas con total independencia del pago/cobro percibido.
    - Lo recauda el Estado Nacional → la tasa es del 0%, 21% o 27%, según el bien/servicio.
    - Es trasladable → no afecta al flujo de fondos o el aspecto económico.
    - Se paga todos los meses → afecta el aspecto financiero.
2. IIBB · Ingresos Brutos → grava los ingresos provenientes de la explotación del negocio.
    - Lo recauda el Estado Provincial → la tasa está entre 0% y 15%, según la provincia.
    - No es trasladable → el fabricante está exento → como se aplica sobre el total de la operación, puede afectar significativamente la rentabilidad.
3. IDCB · Impuestos a los Débitos y Créditos Bancarios · "Impuesto al Cheque":
    - Grava todos los débitos y créditos bancarios → la tasa es del 0,6%.
4. Impuesto a las Ganancias:
    - Grava las ganancias, según definición del fisco (considerando amortizaciones[1]).
    - La tasa es de 35%.

---

[1] Las amortizaciones se pueden registrar como pérdidas, pero es el Estado Nacional quien regula cómo se registran esas pérdidas → los bienes de uso no se consumen en su totalidad en el primer año, sino que deben amortizarse a lo largo de su vida útil (cada bien de uso tiene su propia vida útil).

Ejemplo de primer año de actividad: ventas por \$50M, inversión en bienes de uso por \$33M y otros costos por \$2M. Según el contribuyente, la ganancia sería de \$15M… En realidad, para el 1er año de actividad, sólo se toman \$3,3M (\$33M repartidos en 10 años son \$3,3M por año) en concepto de costo de amortización (y \$3,3M para cada año de los siguientes), por lo que la ganancia será de \$44,7M. Entonces, el impuesto a las ganancias será el 35% de dicha suma.

Amortización → depreciación que sufren los bienes por su uso, obsolescencia o paso del tiempo.

- Como el bien pierde valor, se contabiliza como una pérdida.
- Se asocia al concepto de inversión, el cual es distinto al concepto de gasto:
    - La **inversión** permite aumentar el valor productivo, restando capital de la empresa.
        - Está asociada a un bien/servicio NO consumible a corto plazo → se amortiza.
    - El **gasto** NO permite aumentar el valor productivo, restando capital de la empresa.
        - Está asociado a un bien/servicio consumible a corto plazo → NO se amortiza.

Leasing → contrato de alquiler de un bien (se paga por su uso) con opción de compra del mismo al finalizar el período de alquiler.

- Terminado el período de alquiler, si quien alquiló el bien quisiera venderlo, quien tuvo el *leasing* tiene prioridad si desea comprarlo.

Costo Laboral → *referido al "costo de un empleado" desde el punto de vista del empleador*.

- Está compuesto por:
    - Sueldo Bruto, que está compuesto por Sueldo Neto y por Aportes y Deducciones.
    - Cargas Sociales.
    - SAC (Sueldo Anual Complementario) o Aguinaldo.
    - Vacaciones.
    - Licencias por examen, licencias por enfermedad, etc.
    - Regalos.
- Horas Efectivas de Trabajo → la cantidad de horas en que realmente se produce.
    - Tienen una implicancia directa en la planificación → la duración de las tareas resulta del esfuerzo diario efectivo que puede entregar cada persona.

Infraestructura IT → conjunto de hardware, software, redes e instalaciones que se requieren para desarrollar, probar, entregar, monitorear, administrar y dar soporte a los servicios de IT.

Plan de Contingencia → determinación precisa del *quién*, *qué*, *cómo*, *cuándo* y *dónde* realizar acciones en caso de producirse una anomalía en el sistema de información.

- **Plan de Prevención** → contempla las contramedidas preventivas antes de que se materialice una amenaza.
- **Plan de Emergencia** → contempla las contramedidas necesarias durante la materialización de una amenaza.
- **Plan de Recuperación** → contempla las medidas necesarias después de materializada y controlada la amenaza.

Capacidad → rendimiento máximo que puede ofrecer un servicio de IT o un componente del mismo en un marco temporal.

Performance → rendimiento que se obtiene de un servicio o componente del mismo en un marco temporal → parte de la capacidad que se puede utilizar en un instante dado.

Demanda → lo que se nos pide.

- La comprensión de los modelos y patrones de capacidad y rendimiento ayuda a predecir la demanda y a hacer frente a incidentes.

On-Premise → instalación llevada a cabo dentro de la infraestructura de la empresa.

- La propia empresa es la responsable de la seguridad, disponibilidad y gestión del software.

Cloud Computing → infraestructura IT externa a la empresa, contratada bajo demanda.

Virtualización → el hecho de imitar características de hardware, por medio de software, para crear un sistema informático virtual → se pueden ejecutar múltiples sistemas virtuales, sistemas operativos y aplicaciones en un mismo servidor.

- *Ejemplo:* varios almacenamientos físicos que, mediante un software de virtualización, se muestran como uno solo.

Hiperconvergencia → infraestructura definida por software que virtualiza todos los elementos de los sistemas convencionales definidos por hardware.

Cluster → virtualización en la que se administran recursos de función similar.

Balanceo de Carga → técnica en la que las solicitudes de Internet se distribuyen entre un *cluster* de servidores.

Caché → copia, de alcance rápido, de un determinado recurso.

- Hay mecanismos que permiten mantener actualizada la copia respecto del original.
- Hay *cachés* orientadas a cierto tipo de soluciones.

***TCO (Total Cost of Ownership* · Costo Total de Propiedad)** → costo total de un producto a lo largo de su ciclo de vida completo, considerando todos los costos directos, indirectos y recurrentes.

CapEx → gastos de capital → gastos e inversiones asociados con los bienes.

- *Ejemplo:* la compra de un automóvil.

OpEx → gastos operativos → costos relacionados con las operaciones y servicios.

- *Ejemplo:* la compra del combustible para dicho automóvil.

SLA (*Service Level Agreement* · Acuerdo de Nivel de Servicio) → contrato documentado entre un proveedor y el cliente que identifica tanto los requerimientos como el nivel esperado de servicio.

- "SLA" también puede referirse a lo que dice dicho contrato, por ejemplo: "la disponibilidad[2] será de X horas al año".

Alta Disponibilidad → la disponibilidad mide el grado con el que los recursos están disponibles para su utilización por el usuario final a lo largo de cierto período → cuando se habla de alta disponibilidad se habla de un porcentaje de disponibilidad muy alto y cercano al 100%.

- Todo sistema debe tener establecido un SLA que defina cuánto tiempo y en qué horarios los recursos deben estar disponibles.
- La idea es que las disrupciones, en caso de existir, sean mínimas.

Tolerancia a Fallas → capacidad del servicio para seguir funcionando y sin interrumpirse ante la falla o rotura de algún componente.

- La idea es que no haya disrupciones.

Redundancia → refiere a la (al menos) duplicación de componentes que realizan un trabajo crítico y cuya caída provocaría una disrupción del sistema.

RTO (*Recovery Time Objective* · Tiempo Objetivo de Reparación) → tiempo máximo de recuperación → tiempo que uno se compromete a cumplir para recuperarse de un incidente de gravedad.

- Por lo general, está especificado en un SLA.

RPO (*Recovery Point Objective* · Punto Objetivo de Recuperación) → tiempo que transcurre entre el momento del desastre y el último punto de restauración de los datos → cantidad de datos que la organización va a perder en caso de que se produzca un incidente de gravedad.

- Por lo general, está especificado en un SLA.

UPS → dispositivo que permite mantener el suministro de energía eléctrica por un tiempo limitado a todos los dispositivos que tenga conectados independientemente de la continuidad de la tensión de la red eléctrica.

- La UPS complementa al grupo electrógeno → la UPS da un tiempo de gracia para bajar correctamente los equipos y para arrancar el grupo electrógeno, el cual será fuente de electricidad por un período de tiempo mayor.

[2] La disponibilidad es una variable con la que se puede medir el nivel de servicio.

GESTIÓN DEL VALOR GANADO → método utilizado para conocer el estado del proyecto y, así, poder hacerle un seguimiento y un control. Integra alcance, cronograma y costos para medir el rendimiento y el avance del proyecto en forma objetiva.

Variables Principales (PV, EV y AC):

[PV] Valor Planeado:	**[EV] Valor Ganado:**	**[AC] Costo Real:**
Lo que dijimos que iba a costar aquello que dijimos que íbamos a hacer.	Lo que dijimos que iba a costar aquello que realmente hicimos.	Lo que realmente costó aquello que realmente hicimos.

- Al inicio del proyecto, $\mathrm{PV} = 0$.
- Al finalizar el proyecto, PV equivale al presupuesto del proyecto.

El estado del proyecto se puede medir en **avance** y en **rendimiento**:

Para medir **el avance (según el cronograma)**, se comparan EV y PV:

Variación de Cronograma:

$$\boxed{\mathrm{SV} = \mathrm{EV} - \mathrm{PV}}$$

Índice de Performance de Cronograma:

$$\boxed{\mathrm{SPI} = \frac{\mathrm{EV}}{\mathrm{PV}}}$$

$\mathrm{Si\ SV} > 0 \Leftrightarrow \mathrm{SPI} > 1,$
estamos adelantados respecto del cronograma (hicimos más de lo esperado).

$\mathrm{Si\ SV} = 0 \Leftrightarrow \mathrm{SPI} = 1,$
estamos al día respecto del cronograma (hicimos exactamente lo esperado).

$\mathrm{Si\ SV} < 0 \Leftrightarrow 0 < \mathrm{SPI} < 1,$
estamos atrasados respecto del cronograma (hicimos menos de lo esperado).

Para medir **rendimiento** y **costo (según el presupuesto)**, se comparan EV y AC:

Variación de Costos:

$$\boxed{\mathrm{CV} = \mathrm{EV} - \mathrm{AC}}$$

Índice de Performance de Costos:

$$\boxed{\mathrm{CPI} = \frac{\mathrm{EV}}{\mathrm{AC}}}$$

$\mathrm{Si\ CV} > 0 \Leftrightarrow \mathrm{CPI} > 1,$
estamos por debajo del presupuesto (gastamos menos de lo esperado).

$\mathrm{Si\ CV} = 0 \Leftrightarrow \mathrm{CPI} = 1,$
estamos dentro del presupuesto (gastamos exactamente lo esperado).

$\mathrm{Si\ CV} < 0 \Leftrightarrow 0 < \mathrm{CPI} < 1,$
estamos por encima del presupuesto (gastamos más de lo esperado).

Otros indicadores:

- **[BAC] *Budget At Completion*** → costo presupuestado inicialmente para todo el proyecto.
  - *Lo que dijimos en un primer momento que iba a costar todo el proyecto.*

- **[EAC] *Estimate at Completion*** → costo proyectado al final del proyecto → la re-estimación del costo del proyecto durante su ejecución, la cual se espera que resulte más certera que la inicial (**BAC**), ya que estima un período más corto.

  El valor de **EAC** depende de cómo imaginemos el futuro del proyecto.
  Para eso, se plantean 4 posibles escenarios:

  1. Desempeño Típico → la **CPI** observada hasta el momento se mantendrá igual.

  $$\boxed{\text{EAC} = \frac{\text{BAC}}{\text{CPI}}}$$

  2. Desempeño Atípico → la **CPI** observada hasta el momento ha sido excepcional y, además, de aquí en adelante la **CPI** corresponderá a lo planificado:

  $$\boxed{\text{EAC} = \text{AC} + (\text{BAC} - \text{EV})}$$

  3. Cambio a Desempeño Diferente → la **CPI** observada hasta el momento no se mantendrá, sino que de aquí a adelante habrá otra **CPI** (pudiendo o no ser lo planificado):

  $$\boxed{\text{EAC} = \text{AC} + \frac{\text{BAC} - \text{EV}}{\text{CPI}_{\text{nuevo}}}}$$

  El valor del $\text{CPI}_{\text{nuevo}}$ no siempre es el mismo:

  - Si el $\text{CPI}_{\text{nuevo}}$ debe permitir concluir el proyecto dentro del **BAC**, entonces:

    $$\boxed{\text{CPI}_{\text{nuevo}} = \frac{\text{BAC} - \text{EV}}{\text{BAC} - \text{AC}}}$$

  - Si el $\text{CPI}_{\text{nuevo}}$ estará afectado por la **SPI** observada, entonces:

    $$\boxed{\text{CPI}_{\text{nuevo}} = \text{CPI} \cdot \text{SPI}}$$

  4. Nueva Estimación Detallada → los supuestos de la estimación original no resultan válidos (ni siquiera modificando la estimación original) porque los desvíos en la performance son atribuibles a una mala estimación o bien porque las condiciones del proyecto han cambiado significativamente.

  Para lo que resta del proyecto se debe realizar una nueva estimación detallada:

  $$\boxed{\text{EAC} = \text{AC} + \text{Nueva Estimación}}$$

- **[VAC] *Variance at Completion*** → desvío en el costo total del proyecto.

  $$\boxed{\text{VAC} = \text{BAC} - \text{EAC}}$$

- **[ETC] *Estimate to Complete*** → costo estimado para completar el proyecto.

  $$\boxed{\text{ETC} = \text{EAC} - \text{AC}}$$

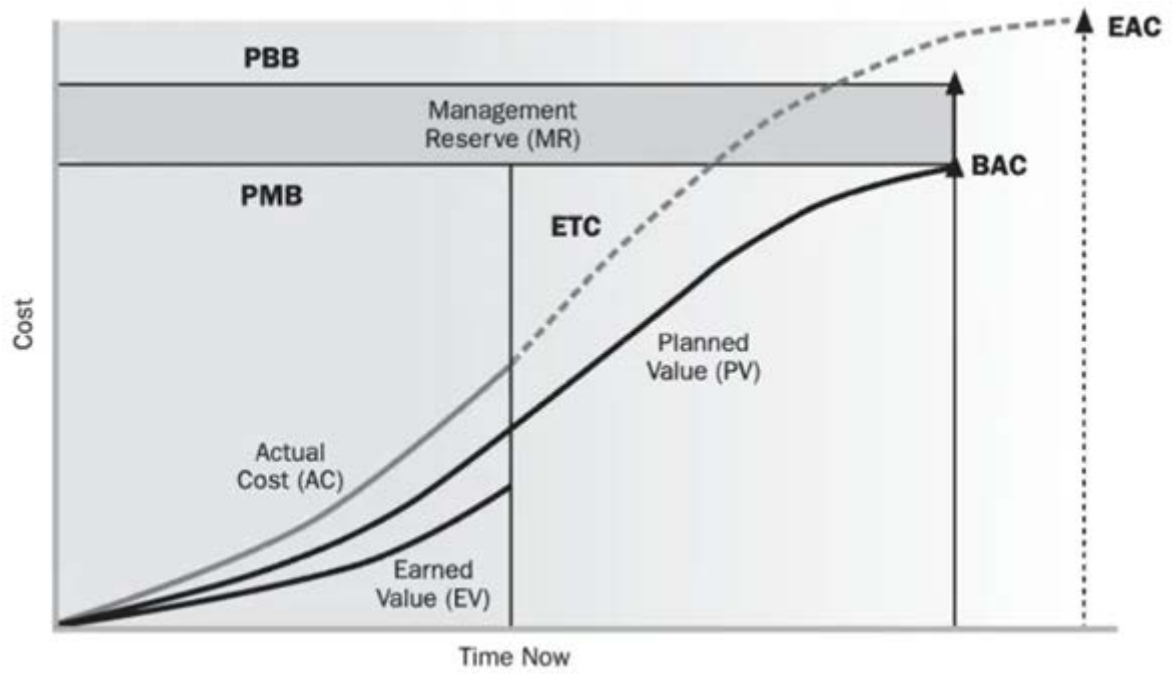
- **[TCPI] Desempeño de Costos Requerido para Finalizar dentro de BAC o EAC:**

  $$\boxed{\text{TCPI} = \frac{\text{BAC} - \text{EV}}{\text{BAC} - \text{AC}}}$$

  No obstante, si hay un nuevo presupuesto, entonces:

  $$\boxed{\text{TCPI} = \frac{\text{BAC} - \text{EV}}{\text{EAC} - \text{AC}}}$$

Ejemplo:



Para medir **el <u>avance</u> (según <u>cronograma</u>)**, se comparan EV y PV:

$$\mathrm{EV} < \mathrm{PV}$$
$$\mathrm{SV} < 0$$
$$0 < \mathrm{SPI} < 1$$

*Estamos **<u>atrasados</u** respecto del cronograma.*

Es decir, hicimos <u>menos</u> de lo esperado.

Para medir **<u>rendimiento</u>** y **<u>costo</u> (según el <u>presupuesto</u>)**, se comparan EV y AC:

$$\mathrm{EV} < \mathrm{AC}$$
$$\mathrm{CV} < 0$$
$$0 < \mathrm{CPI} < 1$$

*Estamos **<u>por encima del presupuesto</u>**.*

Es decir, gastamos <u>más</u> de lo esperado.

<u>Enfoque EVM Ágil:</u>

- Se toman marcos temporales reducidos → la incertidumbre avanza conforme pasa el tiempo, por lo que no tiene sentido hacer un plan hasta el final del proyecto.
    - Los marcos temporales no deben ser muy largos ni muy cortos.
- El tamaño del *backlog* del proyecto se mide en valores relativos → el tamaño estimado para el proyecto será la suma de todos los ítems agrupados por *sprints*.
    - Al proyecto se lo va entregando en *sprints*.
    - Cada *sprint* tendrá cierta cantidad de puntos.
    - Para hacer la conversión costos-puntos, se divide el presupuesto entre la cantidad de puntos totales → así se determina cuánto cuesta cada punto.
        - Conociendo cuánto cuesta 1 *release* y conociendo cuántos puntos va a entregar, puedo determinar cuánto cuesta 1 punto (según el presupuesto, claro). Sabiendo cuánto cuesta 1 punto, puedo determinar cuánto cuesta cualquier cantidad de puntos.
    - La relación entre punto y costo debe ser constante para todos los ítems del *backlog* incluidos en el *release*.

EVM Tradicional vs EVM Ágil:

CONCEPTO	**EVM TRADICIONAL**	**EVM ÁGIL**
Unidad de Medida	$.	Puntos y $.
BAC	Presupuesto del proyecto.	Presupuesto del *release*.
Baseline (Línea de Base)	PV para cada período del proyecto.	Cantidad de puntos (y su equivalente presupuestario) que deben completarse en cada *sprint*.
PV	Costo presupuestario del trabajo que espera realizarse para un momento del proyecto.	Cantidad de puntos (y su equivalente presupuestario) que deben completarse al finalizar un *sprint*.
EV	Costo presupuestario del trabajo realizado para un momento del proyecto.	Cantidad de puntos (y su equivalente presupuestario) que realmente se completaron al finalizar un *sprint*.
AC	Costo real del trabajo realizado acumulado para un momento del proyecto.	Costo real acumulado de los puntos completados al finalizar un *sprint*.
SPI	Tasa de avance lograda en comparación con el cronograma original. $\boxed{\text{SPI} = \frac{\text{EV}}{\text{PV}}}$	Tasa de avance lograda en comparación con el cronograma original. $\boxed{\text{SPI} = \frac{\text{Cant. Puntos Entregados}}{\text{Cant. Planeada de Puntos}}}$
CPI	Cuánto se obtiene por unidad de costo comparado con el estimado originalmente. $\boxed{\text{CPI} = \frac{\text{EV}}{\text{AC}}}$	Cuánto se obtiene por unidad de costo comparado con el estimado originalmente. $\boxed{\text{CPI} = \frac{\text{Costo Planeado por Punto}}{\text{Costo Real por Punto}}}$

BIG DATA 9/5

BIG DATA → conjuntos extensos de datos, principalmente en las características de volumen, variedad, velocidad y/o variabilidad, que requieren una arquitectura escalable para su almacenamiento, manipulación y análisis eficientes.

→ conjunto de datos cuyo tamaño supera la capacidad de las herramientas típicas de software de base de datos para capturar, almacenar, administrar y analizar.

Las 5 V de *Big Data*:

- Volumen → grandes volúmenes de datos.
- Variedad → distintos formatos de datos provenientes de diferentes fuentes de información.
- Velocidad → altas velocidades de acumulación y procesamiento de datos.
- Veracidad → precisión e integridad en la generación y procesamiento de los datos.
- Valor → valor propio de los datos recolectados y analizados para el negocio.

Usos:

- UX (Experiencia de Usuario).
- Desarrollo de productos.
- Fraude y conformidad.
- Eficiencia operacional.
- Mantenimiento predictivo.

Desafíos:

- Lidiar con el crecimiento de los datos.
- Generar conocimiento en forma oportuna.
- Reclutar y retener talento de *Big Data*.
- Integrar diferentes fuentes de datos.
- Validación de datos.
- Seguridad.

Tecnologías y Productos

- Hadoop.
- Spark.
- R.
- Python.
- Data Lakes.
- Bases de Datos No-SQL.
- Predictive Analytics.
- In-Memory Databases.
- *Big Data* Governance Solutions.
- *Big Data* Security Solutions.
- Self-Service Capabilities.
- Inteligencia Artificial.
- Streaming Analytics.
- Edge Computing.
- Blockchain.
- Prescriptive Analytics.

Seguridad en *Big Data* → conjunto de acciones de protección de datos y de procesos de análisis frente a factores que podrían comprometer su confidencialidad e integridad.

- Requiere combinaciones de herramientas de seguridad tanto tradicionales como recientes y procesos inteligentes para monitorear la seguridad a lo largo de la vida de la plataforma.
- Se busca que los datos se enruten a través de un circuito establecido no vulnerable.
- Opera sobre los datos de entrada (lo que ingresa), los datos almacenados (lo que se guarda) y los datos de salida (lo que se envía a otras aplicaciones y reportes).
- Tecnologías:
    - Cifrado → protección de datos en tránsito y en reposo.
    - Control de Acceso a Usuarios → configuración de acceso basada en roles y usuarios.
    - Detección y Prevención de Intrusiones.
    - Seguridad Física → debe considerarse siempre, sea nuestro *data center* o en *cloud*.
    - Gestión Centralizada de Claves → registro de utilización, entrega de claves bajo demanda, abstracción de la administración de claves respecto de su uso, …
- Responsables de la Seguridad:
    - DBAs.
    - Programadores.
    - Áreas de calidad.
    - Seguridad de la información.
    - Áreas de *compliance*.

Conceptos Relacionados:

- ***Data Engineering*** → ingeniería que se dedica a superar los cuellos de botella en el procesamiento de datos y los problemas de manejo de datos para aplicaciones que usan *Big Data*.
- ***Data Science*** → campo multidisciplinario centrado en encontrar información procesable a partir de grandes conjuntos de datos tanto sin procesar como estructurados.
    - Focalizada en encontrar respuestas a las cosas que no sabemos que no sabemos.
    - Se usan varias técnicas diferentes para obtener respuestas, incorporando ciencias de la computación, análisis predictivo, estadísticas y ***machine learning*** para analizar conjuntos masivos de datos en un esfuerzo por establecer soluciones a problemas que aún no fueron pensados.
- ***Data Analytics*** → centrado en procesar y realizar análisis estadísticos en conjuntos de datos existentes, con el objetivo de mostrar dicha información con una herramienta adecuada.
    - Se busca crear métodos para capturar, procesar y organizar datos que permitan tanto descubrir información procesable sobre problemas actuales como establecer la mejor manera de presentar estos datos.
    - Está dirigido a resolver problemas disparados por preguntas cuyas respuestas aún no conocemos → busca producir resultados que pueden conducir a mejoras inmediatas.

Data Engineering nutre a ***Data Science*** y ***Data Analytics***, quienes permiten la toma de decisiones.

GESTIÓN DE ABASTECIMIENTO/COMPRAS 23/5

Abastecimiento → proceso a través del cual una organización puede adquirir bienes o contratar servicios, provistos/prestados por terceros, y que son necesarios para poder cumplir con sus operaciones propias de la organización.

GESTIÓN DE ABASTECIMIENTO/COMPRAS → acción de utilizar los recursos que disponemos de manera efectiva y eficaz para poder mejorar el proceso de compra de los bienes/servicios que necesita la organización para su funcionamiento.

El **proceso de abastecimiento/compra** comprende varios pasos:

1. **Definición de Requerimientos de Compra:**
   - Es una etapa crítica → es necesario ser muy preciso, porque corremos el riesgo de terminar comprando algo que no satisfaga nuestras necesidades.
   - Se trata de traducir la necesidad de un usuario en un requerimiento para los proveedores: se define la necesidad, se determinan las características más importantes del bien o servicio necesario y se establecen sus condiciones de compra y entrega.
2. **Selección del Mecanismo de Compra:**
   - Se determina de qué forma adquiriremos dicho bien o servicio.
   - Los mecanismos de compra están definidos por las leyes de compras públicas de cada jurisdicción y/o por reglamentos de compras internos de cada organización.
   - Los mecanismos de compra más comúnmente utilizados consisten en:
     - Convenios/Acuerdos Marco → pensado para compras habituales donde se especifica un *convenio/acuerdo* con todos los aspectos generales de una compra y, después, en cada instancia, se agregan sólo aspectos particulares de esa compra.
     - Licitación Pública → se llama a una serie de proveedores (en donde en principio puede participar cualquiera) para que oferten un producto/servicio.
     - Licitación Privada → se llama a una serie de proveedores seleccionados (no cualquiera puede participar) para que oferten un producto/servicio.
     - Trato Directo → contacto directo con el proveedor.
3. **Llamado y Recepción de Ofertas** → contacto con los proveedores, donde les pedimos que nos oferten para recibir sus propuestas.
4. **Evaluación de las Ofertas Recibidas** → se analizan y se elige la mejor → **MEP**.
5. **Adjudicación de Ofertas:**
   - Se cierra la evaluación y se decide a quién se comprará.
   - Se deben formalizar, documentar y comunicar los acuerdos administrativos.
6. **Recepción del Producto/Servicio** → se controla que cumpla con todo lo pautado.
   - Es una etapa importante → la conformidad del comprador hace que el proveedor facture.
7. **Seguimiento y Monitoreo de la Compra** → se evalúan los proveedores, se revisan periódicamente fechas de término y renovación de contratos, se aclaran los mecanismos de garantías de la compra, se ordena información relevante para futuras compras, etc.

BENCHMARK → evaluación de desempeño (manifestada con métricas) de algo que nos interesa.

- Un *benchmark* implica siempre una comparación.

Sirve para:

- Comprar elementos a través de características claves para la solución.
- Obtener un resultado objetivo, independientemente de quien realice el *benchmark*.
- Obtener la mejor relación costo/beneficio.
- Comprobar si los elementos estudiados se adecúan a las necesidades.

Ventaja → permite obtener muy buena información sobre algo que nos interesa, que nos puede llevar a comprar ese algo o a recomendarlo.

Desventaja → es costoso. Para las grandes decisiones, las organizaciones no realizan *benchmarks* sino que compran uno publicado por profesionales y, en base a eso, aproximan (si bien no es lo mismo, es más barato).

Proceso de *Benchmarking*:

1. Determinar el elemento de estudio.
   1.1. Determinar qué se someterá a estudio.
   1.2. Elegir factores y variables claves.
   1.3. Seleccionar las opciones disponibles que ofrece el mercado.
2. Preparar el entorno de prueba.
   2.1. Recopilar requerimientos del tipo de *benchmark*.
   2.2. Realizar tareas previas a la etapa de ejecución.
3. Realizar el *benchmark*.
   3.1. Someter el elemento a pruebas.
   3.2. Tomar muestras de las respuestas de las distintas variables analizadas.
   3.3. Realizar comparaciones y obtener resultados.
4. Analizar resultados de la medición.
   4.1. Descartar elementos que no cumplen con las necesidades.
   4.2. Informar resultados.
   4.3. Determinar si se requiere recalibrar el *benchmark*.
   4.4. Desarrollar planes de acción.

[TPC] *Transaction Processing Performance Council* → organización que define *benchmarks* de medición de procesamiento de transacciones en bases de datos con un alto grado de sofisticación (reduciendo al mínimo la interferencia subjetiva de medición).

- Para una determinada situación y problemática de bases de datos, se eligen un producto, un sistema operativo y un hardware. Luego, con eso, se corre una simulación de ejecución de un *benchmark* que permita medir una variedad de métricas (como la cantidad de transacciones por segundo o la cantidad de usuarios concurrentes, por ejemplo).

1. **Armar la TABLA DE REQUERIMIENTOS** → se analiza la información de quien quiere comprar para armar una tabla con dos columnas críticas **[C]** y dos columnas no críticas **[NC]**:
   - **Columna "Indispensable/Obligatorio" [C]** → se listan requerimientos obligatorios; aquellos que deben cumplirse sí o sí para que una propuesta pueda ser evaluada.
     - Una propuesta será evaluada si y sólo si cumple con todos estos requerimientos.
     - Si una propuesta no tiene un requerimiento de esta columna, no será evaluada.
   - **Columna "Preferido/Deseable" [C]** → se listan requerimientos deseados; aquellos que marcan diferencias con el resto de las propuestas, aquellos que *suman puntos*.
     - Hay requerimientos que pueden en estas dos columnas críticas, pero de diferentes maneras → no sólo son obligatorios, sino que marcan una diferencia.
       - *Ejemplo:* un SLA para incidentes de cierta prioridad → no es lo mismo que sea solamente menor o igual a 5 horas (obligatorio) a que sea menor a 4 horas o menor a 3 horas o menor a 2 horas, por ejemplo (deseable).
       - *Otro ejemplo:* el costo total → no es lo mismo que sea solamente menor a $1.000.000 (obligatorio) a que sea de $800.000 o a que sea de $600.00 o a que sea de $400.000, por ejemplo (deseable).
   - **Columna "No Deseado" [NC]** → se listan características no deseadas; aquellas que no sólo que no *suman,* sino que *restan* (lo cual es peor que *no sumar*).
   - **Columna "No Considerado" [NC]** → se listan características que, de cumplirse, no le suman absolutamente nada a la propuesta; son irrelevantes.

2. **Armar la TABLA DE PESOS RELATIVOS** → se consideran sólo los requerimientos de la **columna "Preferido/Deseable"** para luego agrupar por características (físicas, técnicas, de funcionamiento, comerciales, etcétera) agregando una característica *Costo* al final de todo.
   - Esta tabla muestra cuánto pesa cada ítem de acuerdo a nuestro nivel de satisfacción.

   La tabla consta de 4 columnas:
   - **Columna "Ítem"** → una fila por cada ítem, incluida las características que los agrupan.
     - El ítem *Costo* debe estar sí o sí, y sin desagregar.
   - **Columna Nivel 1 "N1"** → porcentaje de peso relativo (sobre un 100%) asignado, en forma arbitraria, a cada característica (no a cada ítem).
     - La columna siempre suma 100%.
     - El ítem *Costo* debe ser alrededor del 20% (siempre menor del 25%) → si el costo pesa mucho, se lleva puesto al resto de las características e ítems, por lo que es mejor utilizar otro método de evaluación de propuestas.
   - **Columna Nivel 2 "N2"** → porcentaje de peso relativo (sobre un 100%) asignado, en forma arbitraria, a cada ítem perteneciente a una misma característica.
     - Para una misma característica, la columna siempre suma 100%.
   - **Columna Nivel General "NG"** → % de peso relativo (sobre el 100%) de cada ítem.
     - La columna siempre suma 100%

3. **Armar la TABLA DE VALORACIÓN DE ATRIBUTOS** → los atributos son las alternativas ofrecidas por el mercado para los ítems a evaluar.
   - La tabla tiene una columna *Ítem* (con sus respectivos agrupamientos introducidos por la TABLA DE PESOS RELATIVOS), donde cada ítem "se abre" con todos los valores posibles.
   - La asignación de valores deberá estar entre 0 y 100, siendo:
     - 0 → para el atributo que cumpla mínimamente con el requerimiento.
     - 100 → para lo que más satisfacción nos daría.
   - Siempre debe haber un atributo que nos de 100%, a excepción de los atributos relativos.

   Existen varios tipos de atributos:

   - **Atributos Mutuamente Excluyentes** → atributos que se excluyen entre sí.
     - Dada una lista de atributos posibles, una alternativa sólo puede tener 1.
     - Siempre es 1 opción sobre N opciones → símil *RadioButton*.
   - **Atributos Binarios** → atributos en donde se tienen únicamente los extremos.
     - Son un caso particular de atributos mutuamente excluyentes.
     - Es todo o nada: 100 o 0.
   - **Atributos Aditivos** → atributos que no se excluyen entre sí.
     - Dada una lista de atributos posibles, una alternativa puede tener cuantos sean.
     - Puede ser 0…N opciones entre N opciones → símil *CheckButton*.
     - Sumados todas las opciones, el resultado es 100.
   - **Atributos definidos con una Función Lineal** → atributos en los que la satisfacción varía de forma continua y no escalonada. Las funciones lineales permiten una variación continua de la preferencia en función del atributo.
     - Un ejemplo es el ítem *Costo*, que varía de forma continua en donde a mayor costo, menor satisfacción (y viceversa). La función lineal que lo represente, con dominio e imagen definidos, tendrá pendiente negativa (donde CM es el *costo máximo con margen de seguridad* y Cm es el *costo mínimo con margen de seguridad*).

       Para armar la función lineal del *Costo*, se pueden dar dos situaciones:
       - Situación 1: el presupuesto máximo está definido (es dato de la consigna), el cual será el CM y, luego, se estima cierto Cm razonable.
       - Situación 2: el presupuesto máximo no está definido (no es dato de la consigna). En este caso, es necesario realizar una investigación de mercado[3] para establecer precios máximo y mínimo de mercado. Al precio máximo, se le agrega un 5% para obtener el CM y, luego, al precio mínimo se le resta un 5% para obtener el Cm.

       Conociendo los valores CM y Cm, es posible armar la función lineal:

       $$F(x) = ax + b \quad \Rightarrow \quad \left\{ \begin{matrix} F(\mathrm{CM}) = 0 \\ F(\mathrm{Cm}) = 100 \end{matrix} \right\} \quad \Rightarrow \quad \left\{ \begin{matrix} a \cdot \mathrm{CM} + b = 0 \\ a \cdot \mathrm{Cm} + b = 100 \end{matrix} \right\} \quad \Rightarrow \quad \ldots$$

     - Otros ejemplos pueden ser algunas magnitudes físicas, como alturas, volúmenes o superficies.

[3] Para hallar los costos totales máximo y mínimo durante la investigación de mercado, se deben considerar, además del *precio de compra*, tanto la *garantía* como el *valor residual*.

4. **Armar la TABLA DE PONDERACIÓN DE PROPUESTAS** → se aplica cada oferta recibida a la TABLA DE VALORACIÓN DE ATRIBUTOS, obteniéndose ponderaciones para cada ítem para cada propuesta.

   Para cada oferta/propuesta/alternativa recibida, se arman 3 columnas:

   - Las columnas ***Atributo*** y ***Valor*** son similares a la TABLA DE VALORACIÓN DE ATRIBUTOS, pero ahora asignamos el valor correspondiente de acuerdo a cada atributo de dicha propuesta en particular.
   - La columna ***Ponderación***, donde se obtiene el valor ponderado de atributo de cada ítem.

   Finalmente:

   - La sumatoria de los valores de cada ponderación corresponde al puntaje que obtiene cada alternativa, siendo la mejor propuesta (es decir, la que mayor satisfacción brindará) aquella que obtenga mayor puntaje.
   - Si la diferencia de puntaje final entre dos alternativas es estrecha (una diferencia de 2 puntos, por ejemplo), no significa que una sea mejor que la otra → esa diferencia está sujeta tanto a la subjetividad de quien realizó la evaluación como al error del método. Considerando esa diferencia, ambas propuestas son prácticamente equivalentes.

[VPP] Valor de Punto de Ponderación → cantidad de dinero que uno debería estar dispuesto a pagar por un incremento de un punto de ponderación.

$$\text{VPP} = \frac{\text{CM} - \text{Cm}}{\text{Peso Relativo del Costo en la TABLA DE PESOS RELATIVOS}(\sim 20\%)}$$

Si el costo tiene un peso relativo del, supongamos, 20%, el costo puede variar entre 0 y 20 puntos de ponderación y además le corresponde una variación entre Cm y CM, por lo que se podrá determinar cuánta plata representa 1 punto de ponderación.

Si se quiere mejorar un ítem de una propuesta recibida o bien agregar algo que la propuesta recibida no lo tiene, podemos calcular cuánta plata estaríamos dispuestos a pagar por esa mejora.

- Para eso, se debe obtener la cantidad de puntos de ponderación que representa esa mejora (ese "salto" entre el valor ofertado y el valor al que se aspira llegar) y multiplicar dicha cantidad por el **VPP**.

BLOCKCHAIN → red P2P que no depende de entidades centralizadas para llegar a un consenso.
→ tecnología que permite tener un registro distribuido donde cada par tiene su propia copia de la información, cuya validez y veracidad se establece por consenso entre los pares.

Conceptos técnicos:

- **P2P** → protocolo de red de comunicación entre pares.
- **Algoritmo de *Hash*** → función que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija.
- **Criptografía Asimétrica** → sistema que utiliza dos claves (una clave pública para cifrar y una clave privada para descifrar) para el envío de datos.
- ***Proof of Work* o Prueba de Trabajo** → algoritmo de consenso basado en operaciones matemáticas complejas (donde se utiliza **fuerza bruta**) y utilizado para la confirmación de las transacciones y la generación de nuevos bloques.
    - Al requerir una gran cantidad de esfuerzo, sirve también para disuadir atacantes maliciosos que quieran lanzar ataques del 51%.
- **Consenso** → que toda la red esté de acuerdo con el resultado de una prueba.

Principios de *Blockchain*:

- Integridad en la Red → la integridad está cifrada y distribuida en todas las etapas del proceso y no depende de cada individuo.
- Poder Distribuido → el sistema distribuye el poder por una red de iguales sin que haya ningún punto de control → las partes no pueden apagar el sistema por sí solas.
- El Valor como Incentivo → el sistema alinea los incentivos de los *stakeholders* y sus intereses.
- Seguridad → se garantiza la confidencialidad y la autenticidad de todas las actividades.
- Privacidad → se elimina la necesidad de conocer la verdadera identidad de los pares.
- Preservación de Derechos → los derechos de propiedad están garantizados.
- Inclusión → la economía funciona mejor cuando funciona para todos.

Secuencia de Pasos:

1. Un usuario solicita una transacción.
2. Se crea un bloque que representa esa transacción.
3. El bloque se difunde a todos los nodos de la red.
4. Dichos nodos validan el bloque y, así, también validan la transacción.
5. El bloque se añade a la cadena.
6. La transacción se verifica y se ejecuta → la transacción ya no se podrá alterar ni modificar.

Usos → cualquiera donde se quiera asegurar que la información no se altere:

- Contratos.
- Libros Contables y Libros de Registros.
- *Tokenización* de Activos.
- Registros Automotores.
- Historias Clínicas.
- Escrituras de Propiedades.

Ventajas:

- Naturaleza Distribuida → no hay ningún nodo privilegiado, ningún nodo puede apagar la red, no tiene un punto único de falla.
- Estabilidad → es muy poco probable que los bloques confirmados sean revertidos.
- Sistema *Trustless* → no requiere de confianza entre terceros.
- Incorruptible → es casi imposible alterar la información contenida en los bloques.
- Transparencia → cualquiera puede consultar las transacciones en el registro y verificarlas.
- Trazabilidad → la información se puede rastrear de manera sencilla y su historial se comprueba constantemente.
- Libre de Errores → los resultados siempre son comprobados y correctos.

Desventajas:

- Ataques del 51% → si un número suficiente de nodos logra hacerse con el control de más del 50%, pasa a ser la mayoría, pudiendo así crear "nuevas verdades" y "una nueva realidad".
- Claves Privadas → perder la clave privada implica no tener acceso a la información.
- Ineficiente → las *blockchains*, en especial las que usan *proof of work*, usan mucha energía debido al gran procesamiento por la fuerza bruta empleada.
- Almacenamiento → la cantidad de información de cada copia de la *blockchain* es inmensa.
- Apertura → cualquiera puede consultar datos presentes e históricos de cualquier otro.

Posibles Riesgos o Ataques[4]:

- Doble Gasto → una misma moneda digital puede gastarse más de una vez.
- Redes Fantasma → red impostora donde se realicen operaciones falsas.

BaaS · *Blockchain as a Service* → *blockchain* pensado como un servicio → redes basadas en la nube por parte de terceros para empresas en el negocio de la creación de aplicaciones *blockchain*.

- Permiten la integración con contratos inteligentes.
- Permiten la integración con plataformas que aseguran identidad.
- Permiten poder trabajar con distintas implementaciones de *blockchain*.
- Ofrecen mecanismos de consenso basados en la identidad.

BFA · *Blockchain* Federal Argentina → plataforma multiservicios abierta y participativa pensada para integrar servicios y aplicaciones sobre *blockchain*.

- Sin criptomoneda → la plataforma no está pensada para criptomonedas.
- Modelo liviano → no usa *fuerza bruta* (el minado) para obtener recompensas.
- Permisionada → el consenso se logra porque los integrantes se conocen (no son anónimos).
- Transacciones gratuitas → las transferencias no tienen costo.
- Almacenamiento *off-chain* → no se almacena información *per se* dentro de la *blockchain*, sino los *hashes* de esa información.
- Software libre → el software se basa en una implementación abierta.

[4] El protocolo de *blockchain* soluciona ambos ataques.

Gestión de RRHH → conjunto de actividades que ponen en funcionamiento, desarrollan y movilizan a las personas para que una organización alcance sus objetivos.

- Durante el proceso de gestión de RRHH intervienen todas las personas de la organización.
- Se necesitan métodos para captar/atraer, conservar/retener y desarrollar los RRHH.

Objetivos generales:

- Crear, mantener y desarrollar un conjunto de personas con habilidades, motivación y satisfacción suficientes para conseguir los objetivos de la organización.
- Crear, mantener y desarrollar condiciones organizacionales que permitan la aplicación, el desarrollo y la satisfacción plena de las personas y el logro de los objetivos individuales.

Liderazgo → capacidad de influir en otros y apoyarlos para que trabajen con entusiasmo en el logro de objetivos comunes.

- Características de líderes efectivos:
  - Son buenos comunicadores.
  - Son flexibles → adaptan su estilo de liderazgo a las necesidades de sus subordinados.
  - Te quitan presión.
  - Saben cómo administrar y resolver los conflictos del grupo.
  - Saben planificar y conocen con precisión los roles de cada miembro del equipo.
  - Delegan la autoridad entre sus subordinados.
  - Generan sinergia entre los miembros del equipo.
  - Definen objetivos y roles claros, compartidos por todos los integrantes del equipo.
- *Liderar* implica *administrar*, pero *administrar* no implica *liderar*.
  - ***Administrar*** → refiere al planeamiento, al control y la ejecución respecto de la asignación eficiente de los recursos/personas a las tareas.
  - ***Liderar*** → no solamente se asignan tareas, se controla y gestiona, sino que también se influye en el comportamiento de las personas.

Gestión del Cambio Organizacional → proceso que busca tanto mitigar los efectos no deseados del cambio en cuestión (sea externo o interno) como aumentar las posibilidades de crear futuro en la organización, su gente y contexto.

- Quiebre → ruptura o cambio brusco en las recurrencias, transparencias, "pilotos automáticos" en los que funcionan ciertos comportamientos, procesos, metodologías o prácticas.
- Transformación → proceso que surge por hechos externos a la organización en pos de un futuro mejor e implican estructuras profundas de los sistemas.
- Cambio → proceso que surge por hechos internos a la organización, respondiendo a una demanda de adaptación dentro del sistema.
- Fuerzas Impulsoras del Cambio:
  - Motivación → proporcionar motivos para una acción.
  - Persuasión → convencer con argumentos (con o sin promesas) a alguien de algo.
- Fuerzas Restrictivas del Cambio → resistencia individual y resistencia organizacional.

Negociación → proceso de comunicación que tiene por finalidad influir en el comportamiento de los demás donde ambas partes lleguen a un acuerdo *win-win* (acuerdo donde ambos ganan).

→ proceso por el cual las partes interesadas resuelven conflictos, acuerdan líneas de conducta, buscan ventajas individuales/colectivas, procuran obtener resultados que sirvan a sus intereses mutuos.

- Buen líder es aquel quien gana al menos una negociación.
- Una buena herramienta para ganar una negociación es usar métricas para medir desempeños. Así, se logra disminuir la subjetividad, siendo un argumento más difícil de contrarrestar.

Pirámide Motivacional de Maslow → jerarquía de necesidades humanas, en donde los humanos desarrollamos necesidades/deseos más elevados conforme se van satisfaciendo las necesidades más básicas.

- Conforme se satisfacen las necesidades más básicas, los humanos desarrollamos necesidades o deseos más elevados.
- Las necesidades superiores ocupan nuestra atención únicamente cuando se han satisfecho las necesidades inferiores de la pirámide.

→ encontrarle un sentido válido a la vida mediante el desarrollo potencial de una actividad.

→ respeto a uno mismo y a las demás personas.

→ relacionadas con los afectos del individuo.

→ seguridad física: tener empleo, casa, etc.

→ necesidades básicas referentes a la supervivencia.

- Son elementos de tecnología que se relacionan entre sí para que los productos, los sistemas de información y los servicios de tecnología funcionen de la mejor forma posible para que el usuario final los use.
- Se ocupa de las interfaces (que dividen lo privado de lo público) centrándose en la comunicación e interacción entre esos elementos.
- Se abstrae de los detalles internos y de cómo funcionan internamente esos elementos.

Interesados en la ARQDS

- Usuario Final → quien usará los sistemas de información o productos software.
    - Interesado en que los sistemas/productos funcionen de la mejor manera posible (se consideran criterios como rapidez, disponibilidad y confiabilidad) para que el usuario final justamente los quiera usar.
- Cliente → quien nos contrató, no necesariamente son el usuario final.
    - Interesado en que se implemente una arquitectura que respete calendario y presupuesto previamente seleccionados.
- *Proyect Manager* (PM) → quien define cómo se lleva adelante un proyecto.
    - Interesado en que los equipos trabajen en forma independiente interactuando con disciplina.
- Arquitecto → quien interactúa con el usuario final, el cliente y el PM.
    - Encargado de comunicar qué arquitectura se implementará.
    - Interesado en dejar conforme a los tres protagonistas mencionados.
- Administradores de Bases de Datos (DBA), desarrolladores, *testers*, gente de seguridad.

Decisiones de Diseño a considerar al elegir la ARQDS

- ¿Procesamiento distribuido o no?
- ¿Software dividido en capas? ¿Cuántas capas? ¿Qué patrones se usan?
- ¿Comunicación sincrónica o asincrónica?
- ¿Dependemos del sistema operativo que tenemos?
- ¿Dependemos del hardware que tenemos?

Contextos/Aspectos a analizar al elegir la ARQDS

- Técnico → conocimiento.
- Negocio → imposiciones de reglas de negocio que tiene la organización (convenios para usar ciertas tecnologías, por ejemplo).
- Ciclo de Vida del Proyecto → saber si el ciclo de vida de un producto encaja con la ARQDS que se desea implementar.
- Profesional → conocer si las características del equipo de trabajo encajan con la ARQDS que se desea implementar.

Atributos de Calidad → propiedades o medidas de testeo que permiten indicar qué tan bien funciona un sistema y cómo satisfacer las necesidades de los interesados.

- Atributos de Calidad referidos a los requerimientos:
    - Requerimientos Funcionales → qué hará el sistema de información o producto software.
    - Requerimientos NO Funcionales → caracterizar las funcionalidades.
    - Restricciones de negocio → reglas de negocio y convenios que tiene la organización.
- Disponibilidad → minimizar las interrupciones del servicio y mitigar posibles fallas que puedan ocurrir.
    - Algunas tácticas pueden ser: detección, recuperación y prevención de fallas.
- Interoperabilidad → capacidad que tienen dos elementos dentro de la ARQDS para poder relacionarse entre sí vía interfaces.
- Adaptabilidad → capacidad de no sentir resultados de cambios (de plataforma, de SO, por ej.), costos o riesgos.
- *Performance* → referido al tiempo y a la habilidad.
- Seguridad → referido a la detección de ataques y a cómo resistirlos.
    - Detección de intrusos, denegación de servicios, verificación de integridad, autenticación de actores, límites de acceso, encriptación de datos, etc.
- Usabilidad → cuán fácil es para el usuario ejecutar una tarea deseada.
- Testeabilidad → buena parte del costo de una buena ingeniería en el desarrollo de los sistemas es absorbida por las pruebas.
- Variabilidad → adaptación al contexto.
- Portabilidad → capacidad para adaptarse a los cambios de plataforma.
- Desarrollo Distribuido → diseño del software.
- Escalabilidad → capacidad de agregar más recursos.
- Monitoreo → controlar e investigar el sistema mientras trabaja.
- Comerciabilidad → si se adaptó lo que terminamos construyendo para lo que quería el negocio.

Patrones de Arquitectura → conjunto de soluciones de diseño que se dan bajo contextos y problemas similares.

Requerimientos de Arquitectura Significativos (ASR)

- Revisar documentación previa sobre qué arquitecturas se implementaron.
- Entrevistar a los interesados en el negocio por qué se eligieron esas arquitecturas.
- Interactuar con el usuario final para entender los objetivos del negocio.

La ARQDS en los Proyectos Ágiles – Características Importantes

- División del proyecto en intervalos de tiempo relativamente cortos (*sprints*).
- Entregas de software entre semanas y meses, tiempos relativamente cortos.
- Interacción continua y fluidez de la comunicación entre el equipo de trabajo y el cliente.
- Alta satisfacción del cliente cuando se entrega una versión.

Existe una combinación de arquitecturas que se basan en proyectos ágiles y arquitecturas de paradigmas estructurados → no siempre hay que caer en lo que ofrece el mercado como solución.

Gestión y Gobierno de la ARQDS

- Evaluación:
  - El arquitecto debe interesarse por la gestión de proyectos.
  - El PM y el arquitecto deben trabajar en conjunto por la perspectiva de la organización.
  - A mayor complejidad de proyectos, más útil es la implementación de una arquitectura.
- Planificación:
  - Si bien la planificación sucede constantemente, existe un plan inicial para convencer a la dirección de construir el sistema y dar una idea de costo y agenda.
  - El PM debe educar a otros *managers* para que puedan corregir desvíos en el desarrollo del software.
- Organización:
  - *Team Leader* → gestiona las tareas del equipo.
  - *Developer* → diseñan e implementan los subsistemas de código.
  - *Configuration Manager* → ejecutan y construyen tests de integración.
  - *System Test Manager* → testeo de Sistema y testing de aceptación.
  - *Product Manager* → representan el marketing

Arquitectura *Cloud* → brindan mayor flexibilidad que otras arquitecturas.

- Servicios a demanda → se contratan servicios a medida que se van necesitando.
- Acceso único de red → varios sistemas pueden acceder al sistema mediante una única URL.
- *Pool* de recursos → los proveedores dan la sensación de que brindan servicios casi ilimitados.
- Independencia de ubicación → se desconocen las ubicaciones de los proveedores.
- Elasticidad rápida → al requerir escalar ciertos recursos es posible hacerlo relativamente rápido.
- Servicios medidos → los proveedores nos cobran únicamente por lo que se utilizamos.
- ***Cloud* Privado vs *Cloud* Público:**
  - *Cloud* Privado → arquitectura privada a nuestra organización, que no se vea para afuera.
  - *Cloud* Público → arquitectura pública, para que sea compartida con el mundo.
  - Esquema Híbrido → combinación entre arquitecturas *on-premise* y *cloud*.
- ***Cloud* vs *On-Premise*** → la responsabilidad de mantener la calidad, en tanto seguridad, performance y disponibilidad, recaen sobre distintos actores:
  - Si la arquitectura es *cloud*, la responsabilidad recae sobre el proveedor.
  - Si la arquitectura es *on-premise*, la responsabilidad es nuestra.

Arquitecturas Monolíticas

- La aplicación se construye como una unidad, un todo → siempre una sola pieza, un monolito.
- Todos los módulos ejecutan dentro de un mismo proceso y sobre un mismo hardware.
- Forma típica que adquieren:
  - Interfaz de Usuario del lado del cliente → formada por páginas HTML y código *JavaScript* que se ejecutan en el navegador web del cliente.
  - Base de Datos → generalmente es una BD relacional, pero puede ser otra.
  - Aplicación del lado del servidor → un único monolito que recibe solicitudes HTTP del cliente, ejecuta una lógica de negocio, recupera y actualiza datos de la BD y arma las vistas HTML que serán enviadas al navegador web.
- Los cambios en la aplicación están muy acoplados:
  - Cualquier cambio implica construir y desplegar una nueva versión de toda la aplicación.
  - Cualquier escalado requiere que escale toda la aplicación.
- Todo cambio condiciona la frecuencia de las entregas → los despliegues son complejos.
- Si bien pueden ser muy exitosas estas arquitecturas, suelen generar cierta frustración en los responsables, sobre todo a medida que la aplicación crece.
- Ventajas:
  - Fáciles de desarrollar.
  - Fáciles de testear → no significa que sean fáciles de corregir.
  - Fáciles de desplegar → no significa que las tareas previas al despliegue sean sencillas, porque el armado de dichas tareas puede llevar bastante tiempo.
  - Fáciles de escalar horizontalmente, replicando en distintos servidores.
- Desventajas:
  - A medida que la aplicación crece:
    - Aumenta la complejidad del monolito → sigue siendo una sola pieza.
    - Aumenta la dificultad de mantener la modularidad inicial.
    - Aumenta la dificultad de incorporar nuevas tecnologías → es muy difícil cambiar la tecnología de un sector sin cambiar todo.
    - Disminuye la fiabilidad → un error en cualquier módulo puede potencialmente afectar toda la aplicación.
  - El escalado es completo → se replica toda la aplicación.

Arquitectura de Microservicios (MSA)	**Arquitectura Monolítica**
Coloca cada elemento de funcionalidad en un servicio separado.	Pone toda su funcionalidad en un solo proceso.
Escala distribuyendo estos servicios entre servidores, replicando según sea necesario.	Escala replicando el monolito en múltiples servidores.

Microservicios (MSA) → estilo de arquitectura en el que una aplicación se desarrolla como un conjunto de pequeños servicios que:

- Ejecutan en su propio proceso y se comunican con otros microservicios vía mecanismos ligeros.
- Se construyen en torno a capacidades de negocio.
- Pueden desplegarse de forma independiente mediante procesos automatizados.
- Poseen una mínima gestión centralizada.
- Pueden escribirse en diferentes lenguajes de programación y utilizar diferentes tecnologías de almacenamiento de datos.

MSA – Características Principales

- **Componentización a través de servicios** → las MSA usan bibliotecas, pero su forma principal de crear componentes de su propio software es segmentarlo en servicios.
    - Componente → unidad de software independiente, se puede reemplazar y actualizar.
    - Bibliotecas → componentes que corren dentro del mismo proceso → están vinculadas a un determinado software que se invocan por medio de llamadas a funciones en memoria.
    - Servicios → componentes que corren fuera del proceso → se comunican por medio de mecanismos como peticiones a *web services* o RPC.
        - Que el servicio corra en un proceso aparte lo dota de independencia → si se cae el servicio, no se cae todo el monolito → no tiene un único punto de fallo.

- **Organizada en torno a funcionalidades de negocio** → el hecho de partir las funcionalidades en servicios puede generar dudas.
    - Los cambios simples pueden llevar a que un proyecto entre equipos tome tiempo y requiera aprobación presupuestaria.
    - Los equipos suelen intentar optimizar su rendimiento introduciendo lógica en la parte sobre la que tienen control directo.

- **Productos, no proyectos**
    - Enfoque de Proyecto:
        - Tiene un objetivo determinado.
        - Se desarrollará en un marco temporal (tendrá fechas de inicio y fin planificadas).
        - El objetivo es entregar una pieza de software que se considerará terminada en algún momento.
        - Al finalizar, el software se asignará a una organización de mantenimiento y el equipo que lo creó se disuelve.
    - Enfoque de Producto → "*somos los padres de la criatura y hay que hacerse cargo*"
        - Un equipo debe poseer un producto durante toda su vida útil.
        - Si nosotros lo construimos, entonces nosotros nos encargamos de él.
        - El equipo de desarrollo tiene un conocimiento profundo del producto → hay una relación más intensa con el negocio, que en general da mejores soluciones.
        - Se concibe al software en una relación continua en la que se busca cómo este puede ayudar a sus usuarios a mejorar sus capacidades en el negocio.
        - Mientras la solución está disponible no termina → termina cuando la retiremos.

- ***Smart endpoints and dumb pipes***
  - La lógica de la aplicación está en los servicios y no en los mecanismos de comunicación.
  - Los mensajes son coreografiados utilizando protocolos simples.

- **Gobierno descentralizado**
  - Los microservicios no están obligados a ninguna estandarización de plataformas.
  - La segmentación de una aplicación en servicios permite elegir para cada uno de ellos cuál es la tecnología que mejor se adapta a sus necesidades.

- **Gestión de Datos descentralizada**
  - Descentralización del modelo conceptual → se segmenta un dominio complejo en múltiples dominios delimitados y mapea las relaciones entre ellos.
  - Descentralización de las decisiones de persistencia de datos → cada servicio administra su propia DB.
  - **Teorema de CAP** → se aplica a sistemas distribuidos, sistemas que tienen partes que se tienen que comunicar entre sí → es imposible para un sistema distribuido de persistencia de datos asegurar permanentemente las 3 características a la vez (sólo 2, máximo):
    - Consistencia → cada lectura recibe la escritura más reciente o bien un error, nunca se lee un dato viejo.
    - Disponibilidad → cada solicitud de lectura recibe siempre una respuesta, aunque esa respuesta puede que no sea la escritura más reciente.
    - Tolerancia al Particionamiento → el sistema sigue funcionando aun si un número arbitrario de mensajes son descartados/retrasados entre nodos de la red.

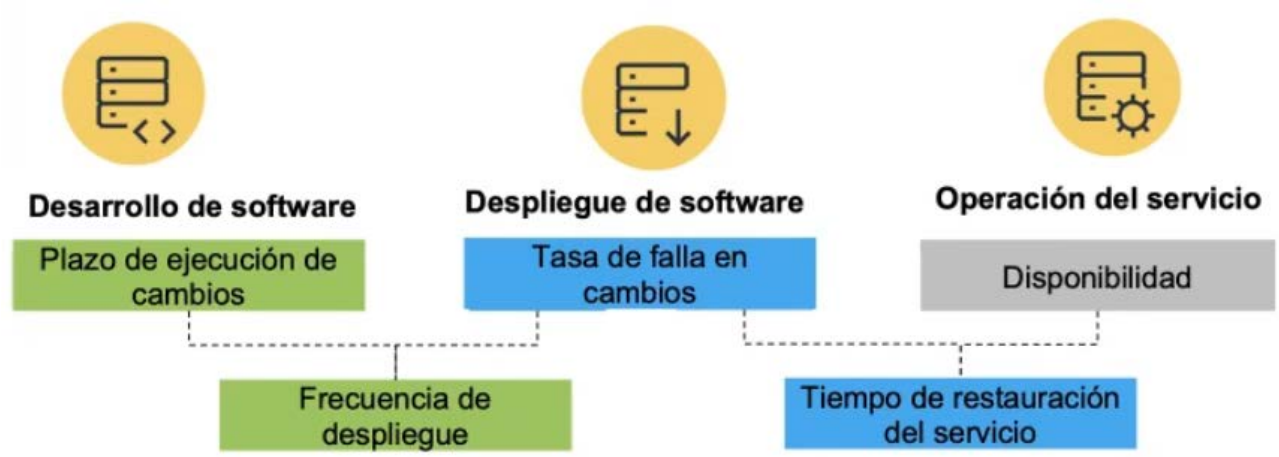
    Una opción es asegurarse C-A y no asegurarse P, pero prepararse para cuando eso suceda:
    - Pueden generarse particiones, ya que son inevitables.
    - Se tienen mecanismos para detectar tempranamente las particiones y, una vez recuperada la conectividad, se trabaja para corregir rápidamente las consecuencias de las particiones (un *merge* entre las dos particiones).

- **Automatización de infraestructura**.

- **Diseño tolerante a fallos** → se preparan soluciones que contemplan escenarios de fallas para lidiar con ellas y mantenerse operativo.
  - Las comunicaciones sobre redes son, por naturaleza, poco confiables.
  - Las aplicaciones se diseñan para ser resilientes y manejar errores, no sólo prevenirlos.
  - Monitoreo en tiempo real de la aplicación → tanto de elementos de arquitectura como de métricas relevantes del negocio.

- **Diseño evolutivo**.
  - Los servicios evolucionan buscando reducir al mínimo el impacto de los cambios en sus consumidores.
  - Los servicios se diseñan buscando el mínimo acoplamiento posible al contrato de sus proveedores.
  - El uso de servicios como componentes posibilita planeamientos de despliegues más granulares.

MSA – Métricas



- Plazo de Ejecución de Cambios → mide el tiempo que va desde que empieza el análisis de los requerimientos hasta que todo se despliega y quede disponible para su uso.
- Tasa de Falla de Cambios → mide cuántos despliegues son exitosos respecto del total de despliegues realizados.
- Disponibilidad → mide qué tan disponible está un servicio.
- Frecuencia de Despliegue → lo ideal es que se las entregas se realicen muy seguido (varias veces por día), con una muy baja tasa de errores.
    - Es la combinación entre el plazo de ejecución de cambios y la tasa de falla de cambios.
- Tiempo de Restauración del Servicio → mide el tiempo que se tarda en recuperar el servicio, en volverlo a disponibilizar.
    - Es la combinación entre la tasa de falla de cambios y la disponibilidad.

Aspecto de la Performance de la Entrega de Software	**Élite**	**Alto**	**Medio**	**Bajo**
Frecuencia de Despliegue	Bajo Demanda. Varios despliegues por día.	Entre 1 vez por día y 1 vez por semana.	Entre 1 vez por semana y 1 vez por mes.	Entre 1 vez por mes y 1 vez por semestre.
Plazo de Ejecución de Cambios	Menos de 1 día.	Entre 1 día y 1 semana.	Entre 1 semana y 1 mes.	Entre 1 y 6 meses.
Tiempo de Restauración de Servicio	Menos de 1 hora.	Menos de 1 día.	Menos de 1 día.	Entre 1 semana y 1 mes.
Tasa de Falla en Cambios	0% – 15%.	0% – 15%.	0% – 15%.	46% – 60%.

MSA – Ventajas

- Facilita el despliegue continuo de aplicaciones grandes y complejas → la aplicación grande y compleja en realidad son aplicaciones chicas combinadas entre sí.
- Facilita el mantenimiento.
- Facilita la incorporación de nuevas tecnologías.
- Permite el despliegue y escalado independiente.
- Permite trabajar con equipos de desarrollo autónomos.

MSA – Desventajas

- Aumento significativo de la complejidad propia de un sistema distribuido.
- Requiere implementar mecanismos de comunicación entre servicios y el manejo de fallos.
- El *testing* de interacción es, en general, más complejo.
- Aumenta la complejidad de implementación, gestión y monitoreo.
- Se dificulta la detección de errores en tiempo de ejecución.
- Complejidad de la arquitectura de persistencia de datos particionada → son muy comunes las transacciones de negocio que requieren actualizaciones en repositorios pertenecientes a múltiples servicios.

¿Cuándo utilizar una MSA?

- Cuando el aprovisionamiento de infraestructura es rápido y, en lo posible, automatizado.
- Cuando hay herramientas adecuadas para el monitoreo correcto.
- Cuando hay mecanismos que permitan un despliegue rápido.

¿Cuándo NO utilizar una MSA?

- Si la aplicación es pequeña y no se espera que escale, microservicios puede resultar una solución complicada y cara → una arquitectura monolítica es una solución mejor.

Hay mejores y peores marcos de trabajo.
No existe la mejor solución → cada una tiene sus debilidades y fortalezas.
A veces se evalúa muy ligeramente la calidad de una solución por la tecnología o la complejidad de la respuesta que damos.

- *Si la necesidad a cubrir es básica, una solución básica puede ser la mejor solución de todas.*
- *A veces, un desarrollo para una solución puntual no suma mucho más que una planilla de Excel adecuadamente concebida y trabajada y cuesta. Además, cuesta mucho más.*

Es importante plantear cuál es la necesidad a cubrir…

En IT, darle al cliente lo que pide no siempre es lo mejor.
Lo mejor es lo que necesita, más allá de si es o no lo que pide.

Es muy común que no nos digan lo que necesitan, sino lo que quieren → piden algo que ellos entienden que va a satisfacer su necesidad, pero si uno escarba un poco más, puede ser que se confirme esa presunción inicial o no: puede ser que la solución para satisfacer esas necesidades sea otra.

Infraestructura de Procesamiento de Datos – Lo que se espera de ella

- Confiabilidad → dada por la estabilidad del sistema.
  - Refiere a que, al ponerla en funcionamiento, responda de manera exitosa.
- Rendimiento → predictibilidad del tiempo de obtención del resultado.
- Sustentabilidad Económica → que se tenga presupuesto suficiente.
  - Nuestra astucia será cuánto más podemos hacer con el presupuesto que hay → también es válido solicitar aumentos de presupuesto (eso también es parte de la buena gestión).

Unidades de Procesamiento de Datos – Lo que esperamos de ellas

- Confiabilidad.
- Disponibilidad → que pueda funcionar de manera continua y regular.
  - Un sistema no sirve de nada que no está en condiciones de ser utilizado.
- Tolerancia a Fallas → que, ante fallas, el servicio siga operando y no se vea afectado.
- Escalabilidad → capacidad de agregar/reducir recursos ante un cambio en la demanda.
  - Hay que estar preparados para cuando aumenta la demanda.
  - La idea es gastar en función de lo que se necesita.
- Compatibilidad entre el hardware y los controladores disponibles.
- Administración Remota.
- Mantenimiento en Caliente → que, ante mantenimientos, el sistema siga funcionando.
  - Los sistemas de más alta disponibilidad siguen funcionando cuando se mantienen, aun si esos mantenimientos no son planificados.

***Mainframe* vs Supercomputadora**

	Mainframe	**Supercomputadora**

Descripción	• Servidor centralizado.
	• De propósito general:
	  o almacena grandes DB.
	  o atiende miles de usuarios en forma simultánea.
	• Ataca problemas limitados por la confiabilidad.
	• Focalizada en la performance de las DB masivas.
		• Equipo cuya fortaleza reside en la capacidad de cálculos complejos en punto flotante, que es posible por la gran cantidad de núcleos que posee.
		• Para procesar semejante cantidad de datos, se requieren suficientes memoria y almacenamiento.
		• Ataca problemas limitados por la capacidad de cálculo.
		• Son trajes a medida.

Funciones Básicas		
Principio de Trabajo		
Velocidad	Millones de instrucciones por segundo.	Miles de millones de operaciones en punto flotante por segundo.
Usos		Ciencia, Industria y Defensa.

Virtualización → recursos hardware y software que se muestran como software, brindándole al usuario una vista distinta de la realidad subyacente.

- Aplicaciones → servidores virtuales, aplicaciones de seguridad, almacenamiento distribuido y/o remoto, hardware de redes, redes virtuales, sistemas de *cluster* de servicios.
- ¿Por qué es importante?
    - Optimiza el uso de recursos.
    - Aumenta la velocidad de despliegue muy significativamente.
    - Aumenta la disponibilidad de los servicios.
    - Disminuye tiempos de parada por mantenimiento de hardware.
    - Permite delegar la gestión de los recursos.

Hiperconvergencia → servidores que vienen con una capa de software que permiten administrar los recursos hardware.

Máquinas Virtuales (VMs) → sobre la infraestructura corre el SO, y sobre él se monta el software de virtualización (un hipervisor), que permite administrar las distintas VMs con sus respectivos SO y aplicaciones.

- Cada VM es una computadora → cada VM tiene su propio SO y sus propias aplicaciones.
- La infraestructura de la VM está virtualizada.

Contenedores → sobre la infraestructura corre el SO, y sobre él se monta el software de virtualización (un contenedor), que permite administrar pequeñas aplicaciones dentro del SO sobre el cual está montado dicho contenedor.

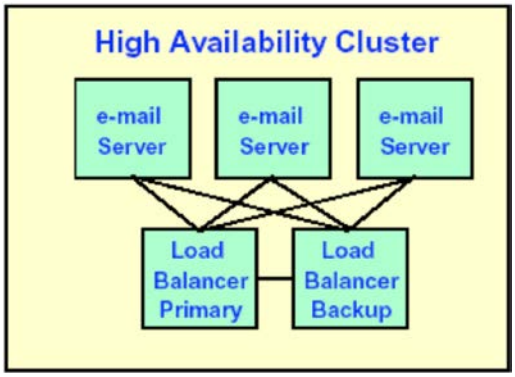
- No siempre se necesitan → en general, complementan las VMs.

Cluster → forma de virtualización → conjunto de recursos (cada recurso es un nodo).

- ***Cluster* de Alta Disponibilidad (HA-C)** → sistema que aumenta la disponibilidad del servicio ofrecido a través de la redundancia de nodos agrupados.
    - Garantiza la disponibilidad del servicio mientras exista al menos 1 nodo operativo.
        - Los nodos hacen exactamente lo mismo.
        - Hay un software de virtualización que orquesta los nodos.
    - Si un nodo del sistema falla, el sistema de gestión del *cluster* transfiere el servicio activo a otro nodo.
    - El nivel de redundancia (dada por la cantidad de nodos) determinará la cantidad de fallas simultáneas admisibles sin pérdida de servicio.
    - La implementación es compleja.
- ***Cluster* de Balanceo de Carga (LB-C)** → provee escalabilidad basada en la distribución de la carga de trabajo entre los nodos activos del sistema.
    - El balanceador de carga distribuye el trabajo entre los distintos servidores de aplicación.
        - Los servidores no hacen lo mismo → no atienden las mismas peticiones.
    - Se busca gestionar la capacidad → la idea es que los servidores reciban una carga pareja, más allá de la capacidad de cada servidor.
    - Si un servidor de aplicación falla, el resto de los servidores debe absorber el trabajo.
    - Si en un momento crítico (servidores trabajando al 100% habiendo peticiones en espera) se agrega otro servidor, aumenta la *performance*.
    - Si se trabaja con 1 único LB-C, hay un punto único de falla.
        - La falla en ese componente desencadena la falla del sistema completo (aunque las otras partes del sistema estén intactas) → el sistema no puede funcionar sin ese componente.
        - Si en algún momento ese LB-C se cae, todo el sistema también se cae.
        - Los puntos únicos de falla pueden pasar desapercibidos.
        - Se soluciona agregando un LB-C secundario.
- ***Cluster* de Alta *Performance* (HP-C)** → pensado específicamente para explotar el potencial del procesamiento en paralelo entre múltiples computadoras.
    - Parte cada petición en sub-peticiones y las reparte entre los servidores.
    - Los servidores trabajan en paralelo cada uno atendiendo su sub-petición.

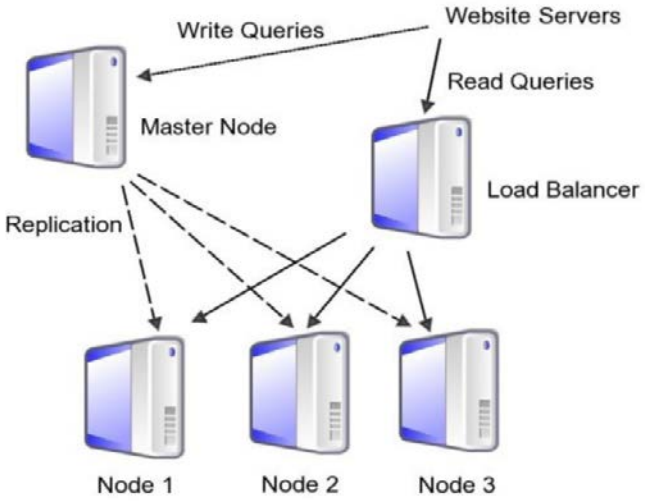
***Clusters* Combinados – Ejemplos**

- **LB-C y HA-C sin punto único de falla**



  - Para evitar que el balanceador de carga sea un punto único de falla, se agrega un segundo balanceador de carga → si se cae uno, el sistema sigue funcionando (pero que no le pase nada a ese porque ahí sí, si ése se cae, el sistema deja de andar).
  - Los balanceadores de carga distribuyen el trabajo entre 3 servidores de mail.

- **Ejemplo más sofisticado → tipos de peticiones**



  - Una granja de servidores recibe peticiones y la clasifican en peticiones de escritura en la DB (INSERTs, UPDATEs, DELETEs) y peticiones de lectura (SELECTs).
    - Se dividen para evitar desbalanceos de carga entre las cantidades de una y las de otra, de manera que no se llegue a un cuello de botella.
  - Si se reciben solicitudes de escritura (*write queries*), se actualiza la DB. Luego, el motor de DB (*master node*) replica en otros 3 servidores (*node 1*, *node 2*, *node 3*).
  - Si se reciben solicitudes de lectura (*read queries*), las toma el balanceador de carga y éste distribuye el trabajo en los 3 servidores (*node 1*, *node 2*, *node 3*).

Grid Computing → cluster de uso general donde los recursos utilizados son ociosos (por no decir residuales) y distantes (no están en el mismo lugar, sino que están distribuidos geográficamente), y son administrados por un software de virtualización.

- Se parece a la hiperconvergencia → se tienen recursos en equipos distintos y se busca que sean vistos de otra manera, no necesariamente "como uno solo".

CLOUD COMPUTING → modelo que permite el acceso ubicuo (es decir, desde cualquier lugar y en diferentes condiciones), conveniente y bajo demanda a través de la red a un conjunto compartido de recursos informáticos configurables (redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden aprovisionar y lanzar rápidamente con un mínimo esfuerzo de gestión o interacción del proveedor de servicios.

Características Esenciales

- Autoservicio Bajo Demanda → el servicio se presta de acuerdo a lo solicitado.
    - No solamente se puede solicitar más o menos recursos en cualquier momento, sino que además, una vez hecho el pedido, se los puede obtener inmediatamente.
    - Que el servicio sea bajo demanda es una novedad → antes el gran problema era que nos exigieran algo que no podíamos dar, generando así otros problemas (como la necesidad de ampliar el presupuesto o diferentes demoras producto de esas necesidades).
    - No se es propietario de lo que se necesita, sino que es una suerte de alquiler → se usa algo brindado por alguien cuya capacidad es prácticamente infinita comparado con una organización individual.
- Amplio Acceso a través de la Red → al servicio se accede a través de la red.
    - La conectividad es esencial → la capacidad de red debe ser muy buena, ya que, si no lo es, no se podrá consumir el servicio como se desea → sin conectividad, no hay servicio.
- Conjunto Compartido de Recursos → los recursos se consumen de una manera distinta a cuando uno tiene contratado un *datacenter*.
    - La cantidad de recursos a compartir es extraordinariamente grande.
    - En un gran *datacenter*, los recursos están perfectamente identificados de manera física.
    - En *cloud computing*, los recursos están virtualizados y el cliente obtiene el equivalente a esos recursos → no se sabe dónde están ubicados físicamente esos recursos, ni interesa.
- Rápida Elasticidad → las capacidades pueden ser aprovisionadas y desplegadas casi de manera automática, permitiendo escalar rápidamente aumentando/disminuyendo según la demanda.
- Servicio Medido → se paga por lo que se consume, siempre de acuerdo a lo contratado.

Modelos de Despliegue

- ***Private Cloud*** → la infraestructura se proporciona para el uso de una sola organización.
- ***Community Cloud*** → la infraestructura se proporciona para la organización, proveedores y socios comerciales que forman una comunidad en su conjunto.
- ***Public Cloud*** → la infraestructura se proporciona para uso abierto, público en general.
- ***Hybrid Cloud*** → combinaciones de los modelos anteriores.

Modelos de Servicio Ofrecidos → *On-Premise* no.

- ***SaaS · Software as a Service*** → el software está instalado en la infraestructura del proveedor del servicio (no en las dependencias del cliente) y el cliente usa dicho software desde afuera.
- ***PaaS · Platform as a Service*** → el cliente ya no contrata algo tan encapsulado como el software a usar sino toda una plataforma (para desarrollo, despliegue y mantenimiento).
- ***IaaS · Infrastructure as a Service*** → el cliente contrata la infraestructura (sea procesamiento, almacenamiento, redes u otros recursos informáticos).
- ***CaaS · Container as a Service*** → el proveedor ofrece herramientas para construir y desplegar aplicaciones basadas en contenedores que resulten seguras y escalables.
- ***BPaaS · Business Process as a Service*** → el proveedor ofrece procesos de negocio, tales como gestión bancaria, publicidad, marketing, administración y finanzas y soporte a clientes.
- ***DBaaS · Data Base as a Service*** → el proveedor ofrece la utilización de DB, desentendiendo al usuario de detalles como la infraestructura subyacente, la instalación y las actualizaciones.
- ***FaaS · Function as a Service*** → el proveedor ofrece la ejecución de código en respuesta a eventos sin la infraestructura compleja típica de la construcción y despliegue de aplicaciones basadas en microservicios.
- ***BaaS · Blockchain as a Service*** → el proveedor ofrece infraestructura y herramientas al usuario para crear y mantener aplicaciones *blockchain*.

<table>
<tr><th></th><th>On-Premise</th><th>IaaS</th><th>CaaS</th><th>PaaS</th><th>SaaS</th><th></th></tr>
<tr><td rowspan="9">Responsabilidad del Usuario</td><td colspan="5">Uso</td><td rowspan="9">Responsabilidad del Proveedor</td></tr>
<tr><td colspan="4">Aplicaciones</td><td>Aplicaciones</td></tr>
<tr><td colspan="4">Datos</td><td>Datos</td></tr>
<tr><td colspan="3">Ejecución</td><td colspan="2">Ejecución</td></tr>
<tr><td colspan="2">SO</td><td colspan="3">SO</td></tr>
<tr><td>Virtualización</td><td colspan="4">Virtualización</td></tr>
<tr><td>Servidores</td><td colspan="4">Servidores</td></tr>
<tr><td>Almacenamiento</td><td colspan="4">Almacenamiento</td></tr>
<tr><td>Redes</td><td colspan="4">Redes</td></tr>
</table>

Crecimiento en *IaaS*, *Paas*, *SaaS* y *Serverless*

- En una época, *cloud* era marginal → muy pocas organizaciones lo utilizaban.
- Existía un miedo respecto a la confidencialidad de los datos, sobre todo en organismos estatales.
- Con el correr del tiempo ha habido un gran acercamiento a *cloud* → es un acercamiento que sigue creciendo día a día.

Planificación Estratégica: *DataCenter On-Premise* vs *Cloud*

	DataCenter On-Premise	***Cloud***
VENTAJAS	• Si el acceso es local, total independencia de la conexión a Internet. • Manejo propio de la seguridad. • Gestión de todos los recursos, excepto el acceso a Internet (que es resuelto por el proveedor). • Manejo propio de las actualizaciones de software → se evitan inconvenientes de compatibilidades entre softwares.	• *CapEx* casi nulo. • *OpEx* dinámico, ajustado a la demanda. • Alta flexibilidad para ampliar/disminuir las capacidades. • Usa estándares/normas internacionales que regulan los servicios brindados. • El software está siempre actualizado, siempre se tiene la última versión.
DESVENTAJAS	• *CapEx* muy elevado. • *OpEx* atado a la capacidad instalada y no a la demanda. • Baja flexibilidad para ampliar/disminuir capacidades → cualquier cambio implica altos costos y demoras en el tiempo.	• Para la misma necesidad, el *OpEx* es más elevado que en *DataCenter*. • Es imposible gestionar la totalidad de la seguridad. • La infraestructura física se comparte. • Las actualizaciones de software pueden generar problemas de compatibilidad con otros softwares también utilizados en la organización.

- Cuando la disponibilidad de los recursos es casi infinita, el enfoque para brindar soluciones (de quienes deciden en el área de IT) cambia drásticamente.
    - Hay soluciones *cloud* que son impensables si se trabaja *on-premise*.
- No es fácil determinar los servicios *cloud* que se contratan → la sobrecontratación es un riesgo, ya que se puede terminar contratando servicios que no se necesitan.
- Para aquellos negocios que tienen perfectamente identificadas situaciones de concurrencia muy por encima de la media, los servicios de *cloud computing* son ideales.
- Desde el punto de vista del área de IT, es necesario no sólo mantenerse actualizado respecto de nuevas ofertas de servicios *cloud* sino también de los enfoques respecto de la utilización de esos recursos y del diseño de soluciones.

PERSISTENCIA DE DATOS → capacidad de almacenar cualquier tipo de información de manera que perdure en el tiempo.

- Persistencia Volátil → los datos no necesitan ser almacenados más allá de su procesamiento.
- Persistencia NO Volátil → los datos deben perdurar más allá de su procesamiento.

Transacción (DB) → conjunto de instrucciones que se ejecutan como una unidad de trabajo, es decir, en forma atómica, de manera indivisible → debe cumplir con las propiedades ACID.

Propiedades ACID → atomicidad-consistencia-aislamiento-durabilidad.

Teorema de CAP → en un sistema distribuido (donde las partes deben comunicarse entre sí), existen tres características cuyo cumplimiento no se puede asegurar siempre: puede ser que las 3 se cumplan la mayor parte del tiempo, pero no siempre; siempre, solamente 2.

- (C) Consistencia → si distintos usuarios se conectan a distintas partes del sistema, deben obtener la misma lectura.
- (A) Disponibilidad → cuando un usuario se conecta al sistema siempre obtiene una respuesta, aunque no se garantiza que esa respuesta sea la última (la más actualizada).
- (P) Tolerancia a la Partición → que el sistema pueda seguir funcionando correctamente aun habiendo sufrido particiones (producto de una catástrofe).
- Una buena estrategia (y muy utilizada), aprovechando que las particiones son muy poco frecuentes, es asegurar la consistencia y la disponibilidad. Respecto de las particiones, podemos prepararnos para, si sucede (no las podemos evitar), recuperarnos rápidamente.

Sistemas de Persistencia Volátil:

- **Sistema de *Caché*** → memoria para el almacenamiento de información de rápido acceso.
  - Necesita inteligencia para determinar, en base a información estadística, cuáles son aquellas cosas que más utilizarán en el futuro, para que luego se puedan guardar allí.
  - Si algo se usa muy seguido, se trae a *caché* y la próxima vez que se lo requiera, no se lo irá a buscar a disco sino a la memoria *caché*.
- **Sistema *MemCaché*** → ubicado detrás de los servidores, a la par de DB y discos.
  - Auxiliar del código que tiene que almacenar información.
  - Las decisiones de uso de caché son tomadas por el servidor.
- **Sistema *Varnish*** → ubicado entre los servidores y el balanceador de carga.
  - Pensado para servidores web cuyo objetivo es cachear contenido de uso frecuente.
  - Las decisiones de uso de caché son tomadas por el propio sistema *Varnish*.
- **Sistema REDIS (*REmote DIctionary Server*)** → esquema de almacenamiento volátil que tiene la opción de persistir en forma no volátil, usando un esquema clave-valor.
  - Puede tener un esquema tolerante a fallas, con un *cluster* de esquema *master-slave* (donde la replicación de los nodos *masters* en los nodos *slave* es asincrónica).
  - Trabaja con un modelo no relacional.
  - Es extraordinariamente veloz → el tiempo de respuesta es muy bajo.
  - Puede escalar bastante → puede atender millones de solicitudes por segundo.

Sistemas de Persistencia NO Volátil:

- **Datos Estructurados** → guardan o respetan una estructura de datos que permite, más allá del posicionamiento físico, almacenarlos o recuperarlos de manera predefinida.
    - Lenguaje SQL → lenguaje estructurado de tratamiento de datos para interactuar con motores de DB relacionales.

- **DB SQL** → DB que implementan modelos relacionales estrictos con el objetivo de garantizar la consistencia de los datos a partir de relaciones.
    - Son las DB más maduras, dada su antigüedad y extensa utilización.

- **DB NO-SQL** → DB cuyo modelo no busca garantizar la consistencia de los datos a partir de relaciones, sino que tienen por objetivo soportar modelos flexibles que no requieran estructuras rígidas propias del modelo relacional.
    - Hay varios esquemas: Clave-Valor (*Key-Value*), familia de columnas, basadas en documentos, basadas en grafos, etcétera.
    - Muchas de ellas están preparadas para sistemas distribuidas → están relacionadas con el Teorema de CAP.
    - *Elastic Search* → motor de búsqueda y análisis de documentos (a partir de relaciones de términos) y, a la vez, DB (ya que recibe información, la almacena y la indexa).
        - Monitorea (*logs*, infraestructura en esquemas *cloud*) en tiempo real.
        - Es sumamente rápido.

- **Persistencia de Objetos** → modelo de persistencia distribuida de archivos implementado generalmente en servicios de *cloud* pública.
    - Tiene gran escalabilidad.
    - Ejemplo → Amazon S3 (*Amazon Simple Storage Service*).

- **Persistencia de Archivos Distribuidos** → sistema distribuido de archivos para manejo de grandes volúmenes de datos.
    - Son de rápido acceso.
    - Tienen alta disponibilidad → tienen un alto nivel de redundancia.
    - Ejemplo → Apache Hadoop. Dicha herramienta es un *framework* de procesamiento y almacenamiento de información.

- **CDN (*Content Delivery Network*)** → sistema distribuido y escalable de entrega de contenidos basado en minimizar el costo de red entre el punto de distribución y el usuario.
    - En Internet, dependiendo de la ubicación del usuario, la *performance* de las aplicaciones suele ser pobre (sobre todo aquellas que demandan un gran ancho de banda) → el usuario recibe un servicio que no es bueno.
    - Una solución a eso son los CDN → suerte de memoria caché ubicada entre los usuarios e Internet que almacena la información que más frecuentemente utilizan los mencionados usuarios.
    - Cuando un usuario realiza una solicitud al servidor original, esa petición se delega a otro servidor más cercano que puede ofrecer mejor *performance* que el original.

Sistemas de Persistencia Políglota → varias formas de persistencia en una misma solución.

- Distintas funcionalidades dentro de una misma solución se persisten de diferentes maneras:
  - *Key-Value* para funcionalidades de carrito de compras e inicio de sesión.
  - DB relacionales para funcionalidades que requieren transacciones.
  - DB con grafos para necesidades de navegación entre distintos conceptos.
- Aplica a microservicios, porque puede suceder que …
  - … varios microservicios usen un mismo tipo de persistencia, compartiendo una misma fuente de datos.
  - … un mismo microservicio use varios tipos de persistencia, requiriendo múltiples fuentes de datos.
  - … varios microservicios utilicen varios tipos de persistencia, requiriendo múltiples fuentes de datos → es común que se requiera una capa de persistencia políglota.

Equipos y Sistemas de Almacenamiento

- **DAS (*Direct Attached Storage*)** → método tradicional donde se conecta el dispositivo de almacenamiento directamente al servidor (PC), como si fuera un disco duro externo.
  - Las aplicaciones hacen las peticiones de datos directamente al sistema de archivos.
  - El acceso a los archivos es a bajo nivel → a nivel de bloque de datos.
  - VENTAJA → es la menos costosa.
    VENTAJA → es la que tiene las mayores velocidades de transmisión de información.
  - DESVENTAJA → escalabilidad limitada → sólo se puede acceder al dispositivo de almacenamiento desde la PC al que está físicamente conectado → al ser un recurso compartido, quienes quieran usarlo deben ponerse de acuerdo.
- **SAN (*Storage Area Network*)** → conjunto de dispositivos que crean una red enfocada en el intercambio de datos a través de una red de alta velocidad, equipos de interconexión (como *switches*) y discos duros donde almacenar los datos.
  - Las aplicaciones hacen las peticiones de datos directamente al sistema de archivos.
  - El acceso a los archivos es a bajo nivel → a nivel de bloque.
  - VENTAJA → muy altas velocidades de transmisión de información.
    VENTAJA → muy buen rendimiento.
  - DESVENTAJA → más difíciles de administrar.
    DESVENTAJA → altos costos.
- **NAS (*Network Attached Storage*)** → método de almacenamiento dedicado para compartir información entre servidores o PCs dentro de una red local o Internet (vía HTTP).
  - Se puede trabajar con la información desde varios equipos en forma simultánea.
  - Las aplicaciones hacen las peticiones de datos a los sistemas de archivos de manera remota y el almacenamiento es local al sistema de archivos.
  - El acceso a los archivos es a alto nivel → a nivel de archivo.
  - VENTAJA → de uso sencillo y bajo costo.
  - DESVENTAJA → no tiene la *performance* de una SAN.

Disponibilidad de los Datos

- **Disponibilidad** → que la información esté accesible para quien la quiera utilizar. Un dispositivo ofrece disponibilidad si los usuarios pueden acceder al mismo y el dispositivo funciona de acuerdo a lo esperado. Cualquier acción que altere eso afecta la disponibilidad.

  Elementos que aumentan la disponibilidad:
  - Redundancia de fuentes.
  - Redundancia de controladoras.
  - Redundancia de *switches*.
  - Redundancia de discos → sistema RAID.

- ***Performance*** → métrica usada para medir la velocidad de un sistema de almacenamiento, como *cantidad de información transmitida/recibida por unidad de tiempo* (la más común) o el *tiempo de respuesta*.

- **Dispositivos de Almacenamiento *Offline*** → muy útiles para la restauración.

  Características:
  - El aspecto esencial es su integridad → reemplazan la información que sí está *online*.
  - La idea es mantener a estos dispositivos en un lugar distinto de donde están los entornos productivos → si sucede un desastre masivo, estarán a salvo.

  Un ejemplo son las cintas magnéticas:
  - La lectura y escritura es secuencial → si sólo nos preocupa grabar (escribir), es muy rápido. Ahora, si necesitamos leer, no.
  - Pueden ser de operación manual, semi-automática o automática.
  - Hay "cintas magnéticas virtuales", que en realidad son discos.

Firewall → dispositivo de seguridad que regula el tráfico entrante y saliente. En base a ciertas reglas, decide si conceder/denegar cierto tráfico específico.

- Permite tener redes internas seguras, controladas y con cierto grado de confianza.
- Puede ser un dispositivo que viene con un software embebido o bien un servicio (SaaS) dentro de una nube pública o privada.

IDS (Sistema de Detección de Intrusos) → solamente avisa; otro sistema tomará acción.

- Analiza lo que atravesó el firewall y, en base a las reglas que tiene almacenada, analiza los paquetes y eventualmente levanta alertas.
- Ubicado detrás del firewall, conectado al *switch* de entrada/salida de la red.

IPS (Sistema de Protección de Intrusos) → sí toma acciones, pudiendo filtrar paquetes.

- Ubicado entre el *firewall* y el *switch* de entrada/salida de la red.

WAF (*Web Application Firewall*)

- Similar al firewall, pero con información específica de la capa de aplicación (del modelo OSI).
- VENTAJA → aplicado al tráfico de Internet, puede aplicar más información más específica respecto a HTTP.

VPN (Red Privada Virtual) → red privada dentro de la red pública.

- Se disfrazan las direcciones IP de los paquetes enviados → la comunicación es segura.
- Un software cliente VPN evita fallas de seguridad en los accesos e impide utilizar ciertas herramientas que pueden poner en riesgo la seguridad de toda una organización.
- Se usa en trabajo remoto, ya que el acceso a redes internas de una organización es seguro.

SIEM → sistemas que reúnen gestión de seguridad en la información con gestión de eventos.

- Una parte analiza la ocurrencia de ciertos eventos y la otra parte analiza cuáles de esos eventos pueden representar algo significativo respecto de la seguridad en la información.
- Monitorea (detecta) y, de acuerdo a lo que examina, actúa (protege).

Actualizaciones de Seguridad → cualquier sistema (firmwares, SO, DB, aplicaciones, etc.) puede presentar fallos de seguridad, por lo que para evitar que cualquier atacante explote esas vulnerabilidades resulta clave mantener actualizados a todos los sistemas involucrados.

¿Por qué la seguridad en la información es importante?

Toda solución deberá contener en, al menos alguno de sus componentes, las características de:

- **Confiabilidad** → que la información la pueda ver quien corresponda.
- **Integridad** → que la información se mantenga igual si no hay cambios.
- **Disponibilidad** → si bien los dispositivos de almacenamiento pueden funcionar bien, la información ahí persistida puede que haya sufrido un daño fruto de algún tipo de ataque malicioso, pudiendo generarse disrupciones en el funcionamiento del sistema.

¿Cuál es su importancia?

- Nos permiten acceder a sistemas remotos, pudiendo intercambiar datos.
- La disponibilidad de acceso a los sistemas informáticos remotos depende de la conectividad. Es decir, si no se puede acceder al servicio es como si el servicio no estuviera.
- La disponibilidad de las redes depende de medios físicos (cables, antenas, microondas y activos de red) y lógicos (protocolos y accesos).

Servicios Disponibles de Conectividad

- **Enlaces de acceso a Internet**.
- **Enlaces punto-a-punto (P2P)**, como LAN o VPN.
- **Enlaces punto-a-multipunto**.
- **MPLS (Multi-Protocol Label Switching)** → sistema que permite mejorar la eficiencia de las redes acelerando el encaminamiento de los paquetes, combinando ventajes tanto del control de enrutamiento como de la conmutación rápida → ideal para conectar sucursales.
- **Enlaces por Fibra Óptica** → el más eficiente, con altísimas velocidades de transmisión.
    - No está libre de incidentes → para evitar cortes de cables, en lugar de hacer un cableado subterráneo, se hacen tendidos.
- **Enlace por Radiofrecuencia** → permite unir dos puntos con línea de vista distantes a muchos kilómetros sin necesidad de cables, aunque utilizando torres y antenas.
    - No está libre de inclemencias climáticas.
- **Enlace Satelital** → llega prácticamente a cualquier locación, aunque son altos los tiempos de respuesta.
    - No está libre de inclemencias climáticas.

Alta Disponibilidad → *cluster* con varios nodos, en donde ante una caída del nodo principal, un nodo secundario lo reemplaza y realiza la misma tarea que hacía el nodo principal. La disponibilidad prácticamente no se ve afectada porque se evita el punto único de falla.

- Se aplica en servidores, balanceadores de carga y dispositivos de red (como *switches*).

CONTINUIDAD EN LAS OPERACIONES DE NEGOCIO 19/9

Desastre → incidente o evento dramático mayúsculo que debe ser definido.

La **continuidad en las operaciones de negocio** es una forma de hacer gestión de riesgos, ya que uno se prepara para situaciones que no sabemos si van a ocurrir, pero, si ocurren, queremos que la organización siga funcionando luego de recuperarse lo más rápido posible.

- Hay que definir qué riesgos hay.
- Hay que definir sobre qué riesgos debemos actuar → para medir la severidad del riesgo se consideran probabilidades de ocurrencia e impacto.

Riesgos que pueden afectar significativamente las operaciones de IT:

- Errores humanos.
- Fallas de hardware/software.
- Pérdida del suministro eléctrico.
- Desastres naturales.

Debemos considerar **planes de contingencia y planes de recuperación** ante desastres.

Un **plan de contingencia** incluye las medidas técnicas, humanas y organizativas necesarias para garantizar la continuidad del negocio y las operaciones de una compañía ante un desastre.

Hay dos métricas relacionadas con la capacidad de recuperación (pero independientes entre sí):

- **RTO (*Recovery Time Objective*)** → tiempo máximo tolerado desde que ocurrió el desastre hasta que el servicio se recupera y se vuelve a operar con normalidad.
    - Si lo más crítico es perder tiempo, el RTO debe ser pequeño.
    - Soluciones para RTO (ordenadas de menor a mayor pérdida de tiempo):
        - Esquema *clusterizado*.
        - Migración manual de datos.
        - Restauración con cintas magnéticas.
- **RPO (*Recovery Point Objective*)** → cantidad de datos que se pueden perder antes de que ocurra el desastre, desde la última copia de seguridad hasta el instante del desastre.
    - Si lo más crítico es perder datos, el RPO debe ser pequeño.
    - Soluciones para RPO (ordenadas de menor a mayor pérdida de datos):
        - Replicación sincrónica.
        - Replicación asincrónica.
        - Replicación periódica.
        - Resguardo con cintas magnéticas.

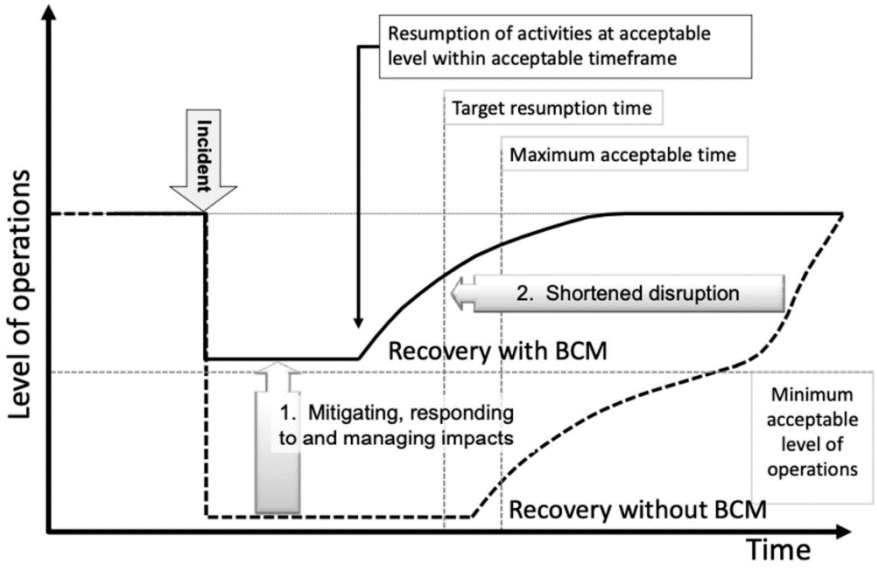
La **gestión de la continuidad del negocio (BCM)** es una parte integral de un proceso de gestión de riesgos que salvaguarda los intereses de las partes interesadas clave, la reputación, la marca y las actividades de creación de valor de una organización. Esto se hace:

- Mediante identificación de amenazas potenciales que pueden causar impactos adversos en las operaciones del negocio de una organización y los riesgos asociados.
- Proporcionando un marco para desarrollar resiliencia para las operaciones del negocio.
- Proporcionando capacidades, instalaciones, procesos y listas de tareas de acción para respuestas efectivas a desastres y fallas.

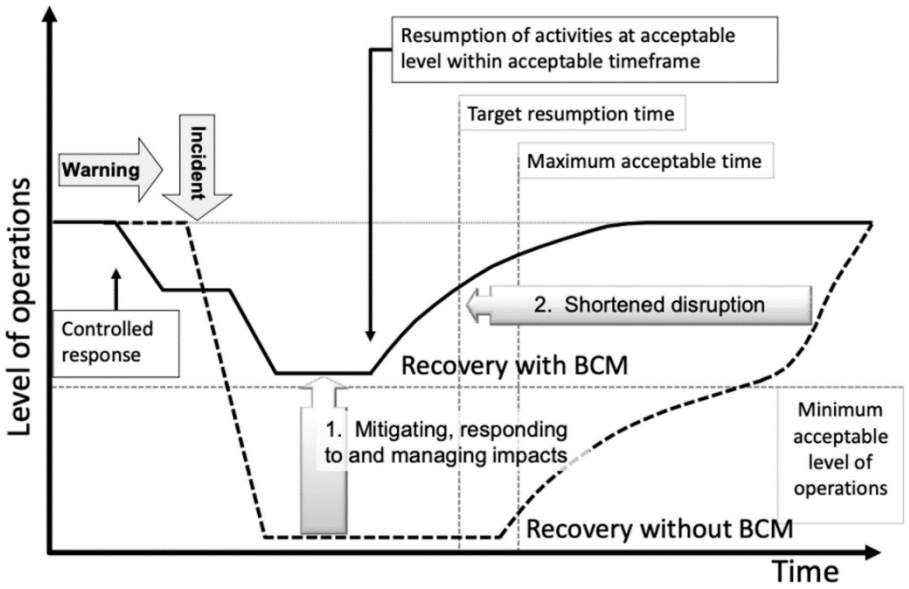
El impacto es mucho menor con BCM.

La **continuidad del negocio** para aquellos eventos considerados como desastre no evitan que ocurran, pero sí puede en algunos casos bajar la probabilidad de ocurrencia. Ante la ocurrencia, la diferencia está en la respuesta, en estar preparados para lo que puede pasar: sabemos exactamente lo que hay que hacer, lo hemos ensayado y lo haremos en forma ordenada.

Situación: Mitigación de Impacto mediante BCM efectiva – Disrupción Brusca



Situación: Mitigación de Impacto mediante BCM efectiva – Disrupción Gradual



En las instalaciones *on-premise*, los datos son una parte importante del asunto, es por eso que existen algunas **estrategias de protección de datos** para mitigar impactos de un desastre:

- Copias de resguardo en discos locales y externos.
- Copias de resguardo periódicas en cinta, sin y con almacenamiento de manera externa en discos locales y externos.
- Replicación de datos en un sitio externo, fuera del ámbito donde se usan.
- Replicación de datos en un *datacenter* externo implementado como sitio de contingencia.

Ahora bien, los datos deben montarse sobre una infraestructura, la cual también es importante, ya que si la infraestructura fue afectada por el desastre, los datos no se podrán usar. Por esa razón, también tiene que haber **estrategias de protección de infraestructura**.

Por otro lado, las **medidas para la recuperación** ante desastres pueden ser:

- Medidas de Prevención → acciones para evitar que los desastres ocurran.
- Medidas de Detección → controles para la detección de desastres e inmediatos avisos.
- Medidas de Corrección → acciones para recuperar la operatoria de los sistemas.
    - Una vez producido el daño, es importante que se frene y que no se dañe nada más.
    - Frenado el daño, se debe hacer un diagnóstico de los daños.
    - Luego, se debe trabajar para recuperar el nivel operativo mínimo y aceptable.
    - Finalmente, trabajar para volver a la normalidad.

La información es un activo, algo valioso para una organización → debe protegerse adecuadamente.

Seguridad de la Información → refiere a las medidas preventivas y reactivas de los sistemas tecnológicos y las organizaciones que tienen como objetivo mantener 3 pilares:

- **Confidencialidad** → no todos pueden acceder a toda la información que posee la organización y, de los que pueden, no todos pueden acceder de la misma manera.
- **Integridad** → toda modificación de información debe ser realizada aplicando ciertas reglas.
- **Disponibilidad** → la información debe estar disponible (según lo establecido en los SLA) para aquellas personas que están autorizadas a acceder a esa información.

Algunas definiciones relacionadas con la seguridad de la información:

- **Evento** → ocurrencia identificada que indica una posible violación de la política de seguridad, pudiendo ser relevante para la seguridad de la información.
- **Incidente** → evento o serie de eventos de seguridad de la información inesperados o no deseados que tiene/n una probabilidad significativa de afectar al negocio.
- **Identidad** → refiere a asegurar que una persona es quien dice ser que es.
- **Autenticación** → refiere a los permisos que tiene una persona (a sabiendas de quién es).

La **gestión de la seguridad de la información (SGSI)** es un proceso continuo que busca establecer y mantener (a lo largo del tiempo) programas, controles y políticas que tengan como finalidad conservar la confidencialidad, integridad y disponibilidad de la información.

Plan de Respuesta a Incidentes de Seguridad

- Fases:
  - Acción inmediata para detener o minimizar su impacto.
  - Investigación temprana del incidente.
  - Restauración de los recursos afectados.
  - Reporte del incidente por los canales apropiados.
- Componentes:
  - Equipo de expertos en seguridad.
  - Estrategia legal revisada y aprobada, referida a obligaciones.
  - Soporte financiero de la organización.
  - Soporte ejecutivo de la gerencia superior de la compañía o áreas afectadas.
  - Recursos físicos.

Seguridad de Datos – Acciones y Herramientas

- Análisis de vulnerabilidades en códigos fuente y aplicaciones.
- Separación de ambientes de desarrollo, de *testing*, de preproducción y de producción.
- *Tests* → *tests* unitarios, *tests* de integración, *tests* de regresión, etcétera.
- Control y auditoría de acceso → mediante registros ocultos, para que nadie los altere.

Seguridad Lógica

- Restringir el acceso a los programas y archivos.
- Asegurar que los usuarios tengan todo lo que necesitan para trabajar y sólo lo que necesitan.
- Que la información transmitida sea recibida solamente por el destinatario deseado.
- Que la información recibida sea la misma que ha sido enviada.
- Que existan canales alternativos secundarios de transmisión entre diferentes puntos.
- NO Repudio:
    - NO Repudio de Origen → asegurarse de que el mensaje fue enviado por quien dice ser el emisor.
    - NO Repudio de Destino → asegurarse de que el mensaje fue recibido por quien dice ser el receptor.
- Tipos de Usuario:
    - Propietario.
    - Administrador.
    - Usuario principal.
    - Usuario de explotación.
    - Usuario de auditoría.

Seguridad Física

- *Backups* → administración de respaldos de información.
- Disponibilidad en sí misma.
- Gestión de centros de cómputos principales y secundarios.

ISO/IEC 27000 → familia de normas orientadas a seguridad de la información.

- ISO/IEC 27001 → contiene los requisitos del sistema de gestión de seguridad de la información y es la norma que se certifica por auditores externos.
- ISO/IEC 27002 → guía de buenas prácticas que describe los objetivos de control y controles recomendables en cuanto a seguridad de la información.
- ISO/IEC 27005 → guía de gestión de riesgos específica para seguridad de la información.
- En otros puntos se hace referencia a:
    - Requisitos de la empresa para el control de acceso.
    - Controles criptográficos.
    - Áreas seguras → controles de acceso físico.
    - Procedimientos y responsabilidades operativas, protección contra *malware*, respaldo, registros y monitoreo, control del software en producción, gestión de vulnerabilidades técnicas.

ISO/IEC 31000 → familia de normas que aplican un enfoque de gestión de la seguridad de la información basado en riesgos:

- **Amenaza** → posible causa de un incidente no deseado que puede dañar un sistema.
- **Riesgo** → exposición a la amenaza → efecto de la incertidumbre sobre los objetivos.
- **Activo** (no definido por la norma) → información, infraestructura y personal que contienen información.
- **Control** → medidas que pueden tomarse para atenuar el riesgo o directamente eliminarlo.
- **Vulnerabilidad** → debilidad de un activo o de un control que puede ser explotada por una o más amenazas.
- **Impacto** → resultado de la materialización del riesgo.

Cuando las vulnerabilidades de activos o controles quedan expuestas a una o más amenazas, se genera un riesgo el cual genera un impacto si se materializa.

Ejemplos de amenazas, vulnerabilidades y activos:

	Ejemplo 1	***Ejemplo 2***	***Ejemplo 3***
Amenaza	Falla del sistema por sobrecalentamiento de la sala de servidores. (ALTA)	Interferencia humana maliciosa mediante denegación de servicio (DDoS). (ALTA)	Interferencia humana accidental consistente en el borrado de archivos. (ALTA)
Vulnerabilidad	El sistema de aire acondicionado tiene 10 años de antigüedad. (ALTA).	El firewall tiene una configuración adecuada y buena mitigación de DDoS. (BAJA)	Los permisos están adecuadamente confirmados, se realizan regularmente auditorías de software y respaldo de datos. (BAJA)
Activo	Servidores ubicados en la sala. (CRÍTICO)	Sitio web. (CRÍTICO)	Archivos en un repositorio compartido. (MEDIO)
Impacto	Todos los servicios no estarán disponibles durante, al menos, 3 horas. (CRÍTICO)	Los recursos del sitio web no estarán disponibles. (ALTO)	Los datos podrían perderse, pero casi con seguridad se podrían recuperar de un respaldo. (BAJO)
Probabilidad	La última falla del sistema ocurrió el mes pasado. (ALTA)	Se detectó una sola DDoS en los últimos 2 años. (MEDIA)	(MEDIA)
Impacto	Pérdida de $30.000.000. (ALTO)	Pérdida de $3.000.000 por cada hora caída. (MEDIO)	(BAJO)
Recomendación de Control	Comprar un nuevo sistema de aire acondicionado por un costo de $1.000.000.	Monitorear el *firewall*.	Continuar el monitoreo de cambios de permisos de usuarios y respaldos.

COBIT 5 → normativa más general, centrada en la gestión de servicios de IT que incluye aspectos de seguridad de información.

- Norma centrada en los objetivos del negocio.
- Alinea la seguridad de la información con los objetivos de la organización.

ITIL → normativa más general, centrada en la gestión de servicios de IT que incluye aspectos de seguridad de información.

- La seguridad de información no es su elemento más fuerte.

PCI → se preocupa de proteger los datos sensibles del poseedor de la tarjeta de crédito, principalmente la confidencialidad y la integridad

Ciberseguridad – ¿Por qué es importante? → los ciberataques son considerados uno de los principales riesgos dentro del mundo de los negocios digitales. El aumento de las comunicaciones hace más probable la ocurrencia de ciberataques.

Herramientas y debilidades:

- Inteligencia Artificial → manipulación tendenciosa a través de *fake news* y *deepfakes*.
- Tecnología móvil de 5ta generación (5G) → deberá modernizarse la infraestructura tecnológica, para poder combatir el déficit de cobertura, seguridad y confiabilidad.
- Computación cuántica → podría reducir drásticamente el tiempo necesario para resolver problemas matemáticos en los que actualmente se apoyan las técnicas de cifrado.
- Computación en la nube → el hecho de que las empresas vuelquen en *cloud* cada vez más información personal crea potenciales riesgos a la privacidad y la seguridad de los datos.

Firma Digital → resultado de aplicar a un documento digital un procedimiento matemático (vía criptografía) que requiere información de exclusivo conocimiento del firmante.

- Se basa en esquemas de clave pública y clave privada.
- Todas las firmas digitales son electrónicas, pero no todas las electrónicas son digitales.

Firma Electrónica → concepto legal que apunta a asegurar la identidad de alguien.

- Una forma de implementarla es con una firma digital.

Servicios de Autenticación → se le delega la responsabilidad de autenticación a otros servicios.