REQUISITOS NO FUNCIONALES

Equipo de Trabajo

Berserkers

Docente
Robinson Coronado Garcia

Responsables
Alejandro Mesa Gómez
Jhon Janer Torres Restrepo
Hernán Javier Aguilar Cruz
Cristhian Javier González Rodríguez

Arquitectura de Software
Universidad de Antioquia
Medellín
2020

Índice

Integridad de datos

En esta época, donde el big data ha tomado un lugar importante dentro de la industria, se trata y almacena más información que nunca, cada vez es más importante poner en práctica medidas que conserven la integridad de los datos recabados. El primer paso para conservar la seguridad de los datos es entender los conceptos básicos de la integridad y saber cómo funciona.

Definición

La integridad de datos se refiere a la precisión, completitud y coherencia general de los datos. La integridad de datos también guarda relación con los datos en su faceta de conformidad normativa, como sucede con el cumplimiento del RGPD y en materia de seguridad. Se mantiene gracias a un conjunto de procesos, reglas y normas que se ponen en práctica durante la fase de diseño.

Cuando la integridad de los datos es segura, la información almacenada en una base de datos seguirá siendo completa, precisa y fiable por mucho tiempo que pase almacenada o por muchas veces que acceda uno a ella. La integridad de los datos también garantiza que sus datos estarán a salvo de fuerzas externas.

Medición de la integridad

La medición de la integridad de datos requiere de la asunción de responsabilidades, partiendo de un compromiso previo que distribuya las competencias de cada encargado de los datos, en base a sus roles. En esta etapa de medición, la calidad del dato durante el ciclo de vida juega un papel fundamental.

La integridad de los datos en un proyecto implica verificar periódicamente las siguientes métricas:

- **Precisión de los datos:** Cada dato es fiel representante de lo que la función que se le atribuye requiere, haciéndolo de la forma establecida.

- **Confiabilidad de los datos:** Garantiza coherencia y estabilidad de la información.

- **Completitud de los datos:** Garantiza que ni en los propios datos ni en los registros o tablas donde se almacenan no falten campos ni valores, que todo esté completo.

- **Conformidad de los datos:** Respetar un formato a la hora de ingresar el dato y cuyas condiciones se han dispuesto de manera específica y predeterminada.

- **Consistencia de los datos:** Relaciona los datos con las las reglas de negocio existentes, garantizando tanto que el dato es correcto en cuanto a sus atributos como también que no vulnera ninguna de dichas reglas.

Tipos de integridad

Existen dos tipos principales que se derivan de la integridad. Ambos se complementan con sus respectivos métodos para hacer esa integridad en los datos de la empresa.

- Integridad física

Es la protección de la integridad y la precisión de los datos cuando sucede una catástrofe natural, se produce un apagón o realizan algún ataque que altera las funciones de la base de datos.

El error humano, del almacenamiento, entre otras incidencias también pueden hacer que los gestores de tratamiento de datos, programadores de sistemas o aplicaciones obtengan datos erróneos.

- Integridad lógica

Conserva los datos y los protege del error humano, pero de forma distinta a la integridad física, por esto, la integridad lógica se divide a su vez en 4 tipos.

- Integridad de la entidad, se basa en el manejo y creación de claves primarias o valores únicos para asegurar que no existan repeticiones y que ningún campo se encuentre nulo.

- Integridad referencial, aseguran que los datos se almacenen y se utilicen uniformemente. Las reglas integradas en la estructura de la base de datos sobre cómo se utilizan claves foráneas para garantizar que tan solo se produzcan cambios, incorporaciones o supresiones de datos adecuados.

- Integridad de dominio, es el conjunto de procesos que garantizan la veracidad de cada dato de un dominio. Dónde el dominio corresponde al conjunto de valores aceptables que una columna puede tener.

- Integridad definida por el usuario, se refiere a las reglas y restricciones creadas por el usuario para adaptarse a sus necesidades particulares.

La integridad de datos no es...

No es seguridad de datos
Si bien la integridad de los datos se ocupa de mantener la información intacta y veraz en el transcurso de su existencia, el objetivo de la seguridad es el de proteger la información de ataques externos.

No es calidad de datos
La calidad de datos determina si la base de datos cumple los estándares definidos por la empresa y las necesidad del negocio. La calidad al igual que la seguridad es tan solo una parte de la integridad de datos.

Riesgos a la integridad

Debemos tener en cuenta también los múltiples factores que pueden afectar a la integridad de los datos.

- **Error humano:** Cuando se introduce información de forma incorrecta, duplican o borran datos o no siguen el protocolo adecuado, se está poniendo en juego la integridad de nuestros datos.

- **Errores de transferencia:** Cuando los datos no alcanzan a llegar a su destino y no se pueden transferir correctamente.

- **Virus y errores de programación:** Estos son capaces de invadir un ordenador y alterar, borrar o incluso apropiarse de los datos.

- **Hardware comprometido:** Los problemas de funcionamiento de un dispositivo resulta ser un claro ejemplo de fallo y esto puede ocasionar que al momento de transferir datos, no se pueda realizar de forma correcta. También al momento de su acceso.

De lo anterior podemos definir algunas estrategias para dar solución o prevenir estas situaciones de riesgo.

- Limitar el acceso a los datos, es decir, cambiar los permisos para restringir los cambios a la información por parte de usuarios no autorizados.
- Validar los datos para corroborar su veracidad al momento de usarlos.
- Crear copias de seguridad.
- Hacer seguimiento mediante registros de cualquier tipo de modificación de los datos.

Este RNF se implementa directamente desde el diseño de la estructura del DBMS, ya sea con métodos tradicionales. Pero en la actualidad la mejor opción es usar un proveedor de servicios desde el cual se ponen las aplicaciones a disposición de los usuarios a través de

internet. Gracias a herramientas de integración de datos en la nube, se pueden conectar múltiples aplicaciones y acceder a todos los datos de su empresa desde una única ubicación.

Elasticidad

La elasticidad se define como el grado en que un sistema es capaz de adaptarse a los cambios de la carga de trabajo mediante el aprovisionamiento y desaprovisionamiento de recursos de manera autónoma, de modo que en cada momento los recursos disponibles se ajusten lo más posible a la demanda actual.
La elasticidad tiene como objetivo hacer coincidir la cantidad de recursos asignados a un servicio con la cantidad de recursos que realmente requiere, evitando un aprovisionamiento excesivo o insuficiente. El sobreaprovisionamiento, es decir, la asignación de más recursos de los necesarios, debe evitarse, ya que el proveedor de servicios a menudo tiene que pagar por los recursos que se asignan al servicio. Por ejemplo, una instancia extragrande de Amazon EC2 M4 cuesta 0,239 USD / hora. Si un servicio ha asignado dos máquinas virtuales cuando solo se requiere una, el proveedor de servicios desperdicia $2095 cada año. Por tanto, los gastos del proveedor de servicios son superiores a los óptimos y su beneficio se reduce.

Debe evitarse el aprovisionamiento insuficiente, es decir, la asignación de menos recursos de los necesarios, de lo contrario, el servicio no puede servir a sus usuarios con un buen servicio. En el ejemplo anterior, el aprovisionamiento insuficiente del sitio web puede hacer que parezca lento o inalcanzable. Los usuarios de la web eventualmente renuncian a acceder a ella, por lo que el proveedor de servicios pierde clientes. A largo plazo, los ingresos del proveedor disminuirán, lo que también reducirá sus ganancias.

La elasticidad a pesar de todo presenta algunos problemas, los cuales se describirán a continuación.

Tiempo de aprovisionamiento de recursos

Un problema potencial es que la elasticidad requiere tiempo. El usuario puede adquirir una máquina virtual (VM) en la nube en cualquier momento, sin embargo, la VM adquirida puede tardar varios minutos en estar lista para usarse. El tiempo de inicio de la máquina virtual depende de factores como el tamaño de la imagen, el tipo de máquina virtual, la ubicación del centro de datos, la cantidad de máquinas virtuales, etc. Los proveedores de nube tienen un rendimiento de inicio de máquina virtual diferente. Esto implica que cualquier mecanismo de control diseñado para aplicaciones elásticas debe considerar en su proceso de decisión el tiempo necesario para que las acciones de elasticidad surtan efecto, como el aprovisionamiento de otra VM para un componente de aplicación específico.

Monitoreo de aplicaciones elásticas

Las aplicaciones elásticas pueden asignar y desasignar recursos (como VM) bajo demanda para componentes de aplicación específicos. Esto hace que los recursos de la nube sean volátiles y las herramientas de monitoreo tradicionales que asocian los datos de monitoreo con un recurso en particular (es decir, VM), como Ganglia o Nagios, ya no son adecuadas para monitorear el comportamiento de aplicaciones elásticas. Por ejemplo, durante su vida útil, un nivel de almacenamiento de datos de una aplicación elástica puede agregar y eliminar VM de almacenamiento de datos debido a los requisitos de costo y rendimiento, variando la cantidad de VM utilizadas. Por lo tanto, se necesita información adicional para monitorear aplicaciones elásticas, como asociar la estructura lógica de la aplicación sobre la infraestructura virtual subyacente. Esto, a su vez, genera otros problemas, como cómo agregar datos de múltiples VM para extraer el comportamiento del componente de la aplicación que se ejecuta en la parte superior de esas VM, ya que es posible que sea necesario agregar diferentes métricas de manera diferente (por ejemplo, el uso de la CPU podría la transferencia podría resumirse).

Requisitos de elasticidad

Al implementar aplicaciones en infraestructuras en la nube, se deben considerar los requisitos de las partes interesadas para garantizar un comportamiento de elasticidad adecuado. Aunque tradicionalmente se intentaría encontrar el equilibrio óptimo entre costo y calidad o rendimiento, para los usuarios de la nube del mundo real los requisitos con respecto al comportamiento son más complejos y apuntan a múltiples dimensiones de elasticidad.

Múltiples niveles de control

Las aplicaciones en la nube pueden ser de diversos tipos y complejidades, con múltiples niveles de artefactos implementados en capas. El control de tales estructuras debe tener en cuenta una variedad de cuestiones. Para el control de niveles múltiples, los sistemas de control deben considerar el impacto que tiene el control de nivel inferior sobre los de nivel superior y viceversa (por ejemplo, controlar máquinas virtuales, contenedores web o servicios web al mismo tiempo), así como los conflictos que pueden aparecer. entre varias estrategias de control de varios niveles. Las estrategias elásticas en las nubes pueden aprovechar los métodos teóricos del control (por ejemplo, el control predictivo se ha experimentado en escenarios de la nube mostrando ventajas considerables con respecto a los métodos reactivos).

Internationalization and localization

En informática, la internacionalización y la localización, son medios para adaptar software informático a diferentes idiomas, peculiaridades regionales y requisitos técnicos de un lugar de destino. La internacionalización es el proceso de diseñar una aplicación de software para que pueda adaptarse a varios idiomas y regiones sin cambios de ingeniería. La localización es el proceso de adaptar software internacionalizado para una región o idioma específico mediante la traducción de texto y la adición de componentes específicos de la configuración regional. La localización (que potencialmente se realiza varias veces, para diferentes lugares) utiliza la infraestructura o la flexibilidad proporcionada por la internacionalización (que idealmente se realiza solo una vez antes de la localización, o como parte integral del desarrollo continuo).

Según Software sin fronteras [5], los aspectos de diseño a considerar al internacionalizar un producto son "codificación de datos, datos y documentación, construcción de software, soporte de dispositivos de hardware, interacción del usuario"; mientras que las áreas de diseño clave a considerar al hacer un producto totalmente internacionalizado desde cero son "interacción del usuario, diseño de algoritmos y formatos de datos, servicios de software, documentación". Los programas informáticos pueden encontrar diferencias más allá de la simple traducción de palabras y frases, porque los programas informáticos pueden generar contenido de forma dinámica. Es posible que el proceso de internacionalización deba tener en cuenta estas diferencias como preparación para la traducción. Muchas de estas diferencias son tan regulares que la conversión entre idiomas se puede automatizar fácilmente. El repositorio de datos de configuración regional común de Unicode proporciona una colección de tales diferencias. Sus datos son utilizados por los principales sistemas operativos, incluidos Microsoft Windows, macOS y Debian, y por las principales empresas o proyectos de Internet como Google y la Fundación Wikimedia. Ejemplos de tales diferencias incluyen:

Diferentes "escrituras" en diferentes sistemas de escritura usan diferentes caracteres: un conjunto diferente de letras, ideogramas, logogramas o símbolos. Los sistemas modernos utilizan el estándar Unicode para representar muchos idiomas diferentes con una única codificación de caracteres. La dirección de escritura es de izquierda a derecha en la mayoría de los idiomas europeos, de derecha a izquierda en hebreo y árabe, o ambos en Bustrófedon, y opcionalmente vertical en algunos idiomas asiáticos. Las mayúsculas existen en algunos lenguajes y no en otros. Los diferentes idiomas y sistemas de escritura tienen diferentes reglas de clasificación de texto. Los diferentes idiomas tienen diferentes sistemas numéricos, que podrían necesitar ser compatibles si no se utilizan números arábigos occidentales. Los diferentes idiomas tienen diferentes reglas de pluralización, lo que puede complicar los programas que muestran dinámicamente contenido numérico. Otras reglas gramaticales también pueden variar, por ejemplo el genitivo. Los diferentes idiomas usan una puntuación diferente (p. Ej., Citar texto con comillas dobles (" ") como en inglés, o guillemets (« ») como en francés) Los atajos de teclado solo pueden hacer uso de botones

en la distribución del teclado para la que se está localizando. Si un atajo corresponde a una palabra en un idioma en particular (por ejemplo, Ctrl-s significa "guardar" en inglés pero en español se usa Ctrl-g), es posible que deba cambiarse.

User friendliness

User friendliness, o por su definición en español “Facilidad de uso” es un requisito no funcional encargado de darle importancia a la capacidad que tiene un software para ser aprendido, utilizado y así mismo ser amigable para un usuario. Esto, bajo ciertos criterios a tener en cuenta, y con diferentes condiciones. Normalmente, se presenta este requisito para cumplir con ciertas necesidades, la más común es la de llegar a más personas, y así poder obtener un mercado más amplio y llegar a más gente con una aplicación. [13]

Cómo se puede lograr

Para cumplir con este requisito, se presentan unas pautas a seguir, tal que:

- Atracción: Nos habla de la capacidad que tiene el producto para ser amigable. Se puede presentar con los atributos, ya sean de la parte visual o del funcionamiento para que sea más atractivo.
- Facilidad de comprensión: Es, como su nombre lo dice, la capacidad que presenta el software para permitirle entender la aplicación de manera más simple. Entonces, hace parte de ver si un software es conveniente para el usuario, y cómo se dan las tareas particulares, etc.
- Facilidad cognoscitiva: La capacidad del producto del software para permitirle al usuario aprender su aplicación.
- Operabilidad: Es la capacidad que tiene una aplicación para prestarse a un usuario de tal manera que lo pueda operar y controlar. Entonces, corresponde a la capacidad de ser controlado y tolerancia ante errores, así mismo una conformidad.

Requerimientos para llevarlo a cabo

Así mismo, para lograr comprender estos requisitos, se necesitan los siguientes requerimientos:

- Control de errores: Presentar tipos de errores que se puedan dar, y una solución a estos para manejar de manera más fácil la aplicación.
- Interfaz amigable e intuitiva: Hacer del diseño algo entendible para todos, reduciendo la complejidad del programa y haciéndolo más visible.

- Información: Hacer uso de manuales que permitan entender la aplicación, como puede ser una documentación.
- Control de procesos: Identificar las tareas que hace la aplicación, para así poder terminarlas de manera satisfactoria sin que ocurran errores.

¿Por qué?

Este, se suele ver aplicado en muchos tipos de proyectos, en general, podemos verlos en aplicaciones para niños o bebés, aplicaciones para gente mayor, o también para aplicaciones con un público grande que necesita manejar una sencillez adecuada. [11]

La definición del requisito de facilidad de uso es fundamental, y parte esencial de un proyecto, ya que este puede no funcionar debido a que sus usuarios no encuentran una manera sencilla de utilizarlo. Así mismo, puede darse que no sea una interfaz agradable, sencilla o atractiva, o porque no exista una manera de entender el funcionamiento del sistema para un usuario. [10]

Explotabilidad

Es la capacidad de que tiene un software para poder contrarrestar el uso o beneficio de su aplicación de tal manera que sea vulnerable

Se debe poder identificar debilidades en el sistema. Así mismo, saber si estas pueden ser explotables y que no exista una ruta por la cual esto se pueda hacer. Si se realiza que existe alguna dificultad como esa, se deberá tomar medidas de seguridad adecuadas para que no exista algún riesgo. Vamos entonces, con la explotabilidad haciendo referencia a esta parte en la cual se podrían ejecutar los procesos vulnerables en nuestra aplicación, y así mismo saber cómo actúan para poder evaluarlos y atacarlos.

Vulnerabilidad vs Explotabilidad

La explotabilidad es la encargada de encontrar estas vulnerabilidades. Así mismo, es capaz de evaluar su potencial. “La vulnerabilidad se ocupa de lo teórico, la explotabilidad se ocupa de los datos reales”.[14]

¿Por qué?

Una vez se tiene presente que pueden existir riesgos de seguridad en la aplicación, es evidente que se deben plantear soluciones para atacar estos mismos. Lo que presenta la explotabilidad es la manera en la cual actúan estas partes vulnerables del sistema, por lo que es importante tener presente este requisito.

Operabilidad (Operability)

Según la norma **ISO 9126** [4] (llamadas hoy ISO 25000)la operabilidad es uno de los 5 criterios de la usabilidad y se define como *"la capacidad de un producto software para permitir al usuario manejarlo y controlarlo."* sin embargo actualmente se considera un RNF cuya definición es la siguiente [5] *"Capacidad del producto software de ser entendido, aprendido, usado y atractivo al usuario, cuando es utilizado bajo ciertas condiciones específicas. Considera las subcaracterísticas de: Reconocimiento de adecuación, Capacidad de aprendizaje, Facilidad de uso, Útil, Atractivo, Accesible técnicamente, Adherencia a Normas."*

Métricas

Como la 'Operabilidad' está contenida dentro de la 'Usabilidad' puede medirse con las 3 métricas fundamentales de la Usabilidad: [9]

- **Medir la efectividad**
  - El "porcentaje de éxito" (o porcentaje de conclusión) hace referencia a los participantes que alcanzan de forma correcta cada objetivo. De forma que, idealmente, antes de que empieces a realizar pruebas, tendrás que haber identificado varios escenarios a testear. El test no debe contar con la ayuda de un moderador.

- **Medir la eficiencia**
  - Las métricas de la eficiencia hacen referencia al tiempo de media que es necesario invertir para completar cada tarea. Junto a esto, también puedes calcular el rango y la desviación estándar. Esta es la métrica principal pero existen otras muchas métricas que puedes recoger:

    - Tiempo invertido en el primer intento.
    - Tiempo requerido para completar una tarea comparado con el que necesitaría un experto.
    - Tiempo invertido en subsanar errores cometidos.

Esta lista no enumera todas las métricas y deberías elegir tareas que tengan sentido para ti. Por ejemplo, si estuvieses probando un proceso muy breve no valdría la pena que calculases algo como el tiempo invertido en corregir los errores.

Los fallos de los usuarios son habituales, entre ellos podríamos incluir las acciones, los olvidos o las equivocaciones. Yo normalmente agrego a cada uno una descripción breve, una valoración estricta y los clasifico bajo su respectiva sección. Basándome en indicadores estándar del sector intento no superar la cifra de 0.7 errores por tarea.

Las métricas de eficiencia también pueden ser buenas cuando comparamos los diferentes tipos de usuarios. Comparar, por ejemplo, los principiantes con los expertos.

- **Medir la satisfacción**
    - El nivel de satisfacción puede medirse y calcularse por medio de la “Escala de Sistemas de Usabilidad”. La escala estándar contiene diez preguntas que miden la opinión general del usuario sobre la usabilidad del software.

Robustez (Robustness)

Un sistema robusto es un sistema fuerte, sin debilidades ni vacíos de seguridad. La robustez en un programa informático hace referencia a su capacidad para hacer frente a errores mientras se está ejecutando.

Un programa es robusto si se comporta en forma razonable aún en circunstancias que no fueron anticipadas en la especificación de requerimientos; por ejemplo cuando encuentra datos de entrada incorrectos o algún malfuncionamiento del hardware como rotura de disco. Un programa que genere un error no recuperable en tiempo de ejecución tan pronto como el usuario ingrese inadvertidamente un comando incorrecto no será robusto, aunque podría ser correcto si en la especificación de requerimientos no se establece la acción a tomar si se ingresa un comando incorrecto.

La robustez es una cualidad difícil de definir, ya que si se pudiera establecer en forma precisa lo que se debiera hacer para obtener una aplicación robusta, se podría especificar completamente el comportamiento “razonable”, con lo cual sería equivalente a la correctitud, o a la confiabilidad en el caso ideal mencionado en la definición anterior.

Se puede observar que la robustez y la correctitud están fuertemente relacionadas: si se incluye un requerimiento en la especificación será un tema de correctitud, si no se incluye podría ser un tema de robustez. La línea divisoria entre ambos es la especificación del sistema. La relación con la confiabilidad surge del hecho de que no todos los

comportamientos incorrectos significan problemas igualmente serios, algunos comportamientos incorrectos pueden ser tolerados.

Observación:

La correctitud, confiabilidad y robustez también se aplican al proceso de producción del software, por ejemplo un proceso es robusto si puede adaptarse a cambios no previstos en el entorno como ser una nueva liberación del sistema operativo o una transferencia repentina de la mitad de los empleados a otra sección, un proceso es confiable si lleva consistentemente a la producción de productos de alta calidad.

Métricas

En este caso existen técnicas especializadas para medir la 'Robustez' de un sistema, las cuales se conocen como Robustness testing, una de las que más se destaca es el **Fuzzing** (o Fuzz Testing) que consiste en proporcionar datos inválidos, inesperados o aleatorios a las entradas del programa.

La **inyección de fallas** es una técnica para mejorar la cobertura de una prueba mediante la introducción de fallas en las rutas del código de prueba.

Referencias

[1] Mell, P., & Grance, T. (2011, September 28). The NIST Definition of Cloud Computing. Retrieved September 15, 2020, from https://csrc.nist.gov/publications/detail/sp/800-145/final

[2] Herbst, Nikolas; Samuel Kounev; Ralf Reussner (2013). "Elasticity in Cloud Computing: What It Is, and What It Is Not" (PDF). Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013), San Jose, CA, Retrieved September 15, 2020, from https://sdqweb.ipd.kit.edu/publications/pdfs/HeKoRe2013-ICAC-Elasticity.pdf

[3] Hudson, R., & Hall, P. A. (1997). *Software without frontiers: A multi-platform, multi-cultural, multi-nation approach*. Chichester: Wiley.

[4] Organización Internacional de Normalización. (1991). *Calidad del Software (ISO/IEC 9126).* https://iso25000.com/index.php/normas-iso-25000

[5] Francisco Valdés Souto. (2013). *Definición De Categorías Que Determinan La Calidad De Software (Std. ISO/IEC 25000-Square).* https://franciscovaldessouto.wordpress.com/2013/04/26/definicion-de-categorias-que-determinan-la-calidad-de-software-std-isoiec-25000-square/#:~:text=Operabilidad%3A%20Capacidad%20del%20producto%20software,utilizado%20bajo%20ciertas%20condiciones%20espec%C3%ADficas.&text=Mantenibilidad%3A%20Capacidad%20del%20producto%20software%20para%20ser%20modificado

[6] SG Buzz.(2013). *Midiendo La Calidad Del Software.* https://sg.com.mx/revista/40/midiendo-la-calidad-del-software
[7] Web Design Envato Tuts+.(2017). *3 Métricas Para Medir Y Cuantificar La Usabilidad.* https://webdesign.tutsplus.com/es/tutorials/3-metrics-for-quantifying-usability--cms-29150
[8] Informática para tu negocio. (2017). *¿Qué Entendemos Por La Robustez Del Dato En Informática?.* https://www.informaticaparatunegocio.com/blog/entendemos-la-robustez-del-dato-informatica/#:~:text=Un%20sistema%20robusto%20es%20un,errores%20mientras%20se%20est%C3%A1%20ejecutando.
[9] RUMBAUGH, I.J.G.B.J.. *El Proceso Unificado de Desarrollo de Software*. 2000: Addison Wesley. Capítulos 7, 8 páginas 125-163, 187-202.
[10] Giraldo, O.P. (2007). *Ingeniería de Requisitos*. Volumen, 13.
[11] Thayer, M.D.a.R. (1990). *"Standards, Guidelines and Examples on System and Software Requirements Engineering"*.
[12] Desarrollo de Software (2011). *Determinación de los requisitos no funcionales de la aplicación.* Recuperado de: https://rzamurianos.wordpress.com/2011/06/29/determinacin-de-los-requisitos-no-funcionales-de-la-aplicacin/
[13] Sec-Tec (2016) *Vulnerable vs explotaible.* Recuperado de: https://www.sec-tec.co.uk/blog/vulnerable-vs-exploitable-what-vulnerabilities-count
[14] CloudTweaks. (2017). *Vulnerability VS. Exploitability: Why they're different.* Recuperado de: https://cloudtweaks.com/2017/07/vulnerability-vs-exploitability/
[15] Talend Real-Time Open Source Data Integration Software. (2020). *¿En Qué Consiste La Integridad De Los Datos Y Por Qué Es Importante?.* Recuperado de: https://www.talend.com/es/resources/what-is-data-integrity/
[16] PowerData, R., (2013). Qué Se Entiende Por Integridad De Los Datos. Recuperado de: https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/348870/qu-se-entiende-por-integridad-de-los-datos
[17] PowerData, R., (2014). Medición De La Integridad De Datos. Recuperado de: https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/349180/medici-n-de-la-integridad-de-
[18] Instituto de Computación - Universidad de la República. (2002). *Cualidades representativas de productos y procesos de software.* *https://www.fing.edu.uy/tecnoinf/mvd/cursos/ingsoft/material/teorico/CualidadesSoftware.pdf*