|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | Tecnólogo en implementación y gestión de bases de datos |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | 220501092 - Establecer requisitos de la solución de *software* de acuerdo con estándares y procedimiento técnico | RESULTADOS DE APRENDIZAJE | 220501092-02 Construir el informe de requisitos de la solución conforme a la identificación de las necesidades del negocio |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | CF18 |
| NOMBRE DEL COMPONENTE FORMATIVO | Construcción del informe de requisitos |
| BREVE DESCRIPCIÓN | En los proyectos de desarrollo de *software*, resulta determinante la fase de levantamiento de información, razón por la cual se debe estudiar a fondo el tema, dado que se trata de la base fundamental del *software* que se planea producir. Ahora bien, para afinar las competencias en estos aspectos, se requiere el estudio de las técnicas que permiten el mejor aprovechamiento de la mencionada información. |
| PALABRAS CLAVE | Requerimientos, Levantamiento, entrevistas, ingeniería de requisitos, casos de uso |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | Ciencias naturales, aplicadas y relacionadas |
| IDIOMA | Español |

# **Tabla de contenidos**

1. **Requisitos de *software***

1.1. Requisitos funcionales

1.2. Requisitos no funcionales

1.3. Estándares

2. **Técnicas de recolección de requisitos**

2.1. Técnicas tradicionales

2.2. Técnicas ágiles

2.3. Técnicas de Priorización de Requisitos

2.4. Ventajas y desventajas

3**. Viewpoint**

3.1. Concepto y aplicación

3.2. Modelos

3.3. Stakeholders

**Introducción**

|  |
| --- |
| Cuadro de texto |
| En el presente componente se desarrollan contenidos relacionados con la construcción del informe de requisitos de software que va de la mano con la recolección de información, pues no se puede priorizar una de la otra, simplemente se complementan, puesto que mientras una permite obtener la información necesaria para iniciar el proyecto, también nos permiten entender y construir todo el comportamiento del nuevo sistema que se planea desarrollar. Se invita a revisar el siguiente video para identificar el contexto de aprendizaje. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tipo de recurso** | Video spot animado | | | |
| **NOTA** | **La totalidad del texto locutado para el video no debe superar las 500 palabras aproximadamente** | | | |
| **Título** | Introducción | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración (voz en off)** | **Texto** |
| **1** | Se sugiere que salga una persona presentando, a medida que se hable aparecen imágenes que refuercen la idea transmitida.  Juego De Caracteres Del Hombre De Negocios Carácter Animado ...  Imagen de referencia (o similar)  <https://acortar.link/5CO4c5>  228130\_i1  Isometric Vector plano 3D Ilustración conceptual del análisis de requisitos, desarrollo de productos de software  Imagen de referencia (o similar)  <https://acortar.link/gDDgD0>  228130\_i2  Administración ágil, los principios del desarrollo ágil de software y la administración ágil a diversos procesos de administración, ciclo de vida del desarrollo de productos y administración de proyectos. Cambio de concepto.  Imagen de referencia (o similar)  <https://acortar.link/b2OKqv>  228130\_i3 | N/A | La idea central es que los requisitos de *software* se deben identificar, plasmar y priorizar; para ello, es muy importante dejar claro que este proceso es la base fundamental en los proyectos de desarrollo de *software*. | Requisitos de *software*  Identificar  Plasmar  Priorizar |
| **2** | A medida que se hable aparecen imágenes que refuercen la idea transmitida.  Refuerzo preparado para verter una base de hormigón reforzado o una plancha. Preparación de la base para verter hormigón. Trabajos monolíticos de hormigón reforzado en una construcción industrial  Imagen de referencia (o similar)  <https://acortar.link/Bspbaf>  228130\_i4  Joven poniendo losas de hormigón gris en el patio de la casa sobre la base de la base de grava. El maestro pone piedras de pavimento. Calzada de ladrillo de jardín pavimentada por un trabajador profesional de pavimentación. Reparando la acera.  Imagen de referencia (o similar)  <https://acortar.link/eB0EuI>  228130\_i5  La crisis de deuda inmobiliaria o inmobiliaria que causa efecto dominó, el concepto de caída del mercado de la vivienda y el mercado de valores o de la inversión, el pánico del inversionista de negocios se aleja del dominó inmobiliario que colapsa.  Imagen de referencia (o similar)  <https://acortar.link/okrCx6>  228130\_i6 | N/A | Se puede hacer la analogía con el concepto de los cimientos de las edificaciones, que son las que soportan el peso, repartiendo equitativamente la carga a lo largo del terreno, con el único propósito de servir como una base sólida; pero, si por alguna razón técnica o de cualquier otra índole, llegara a fallar este cimiento o en su defecto, no se armara bien, desembocaría en el derrumbe de la construcción. | Edificaciones  Peso  derrumbe |
| **3** | A medida que se hable aparecen imágenes que refuercen la idea transmitida.    Imagen de referencia (o similar)  <https://acortar.link/axRxIE>  228130\_i7  Requerimientos no funcionales: Ejemplos - La Oficina de Proyectos de  Informática  Imagen de referencia (o similar)  <https://acortar.link/SrmBQQ>  228130\_i8  Pantalla portátil foto de primer plano estándares de codificación de sitios web de tecnología de la información para el diseño web pantalla de desarrollador del programador  Imagen de referencia (o similar)  <https://acortar.link/KollFJ>  228130\_i9 | N/A | Por otro lado, luego de sé que recolecta o se extraen los requisitos, se deben clasificar en funcionales y no funcionales, aplicando los estándares que la industria del *software* tiene para este menester. Se necesita, ahondar en el tema de las metodologías para el acopio de los mencionados requisitos, dado que este aspecto brinda un factor diferenciador para que la anhelada consecución de información, logre los retos planteados. | Requisitos  Funcionales  No funcionales  Estándares  Metodologías |
| **4** | A medida que se hable aparecen imágenes que refuercen la idea transmitida.  Requerimientos funcionales: Los niveles de granularidad - La Oficina de  Proyectos de Informática  Imagen de referencia (o similar)  <https://acortar.link/C2FRxn>  228130\_i10  7 Ejemplos de requerimientos no funcionales  Imagen de referencia (o similar)  <https://acortar.link/m8cHpy>  228130\_i11  Requerimientos Funcionales y No Funcionales – Porqueria  Imagen de referencia (o similar)  <https://acortar.link/1FBjfU>  228130\_i12 | N/A | Por supuesto, sin perder de vista la importancia de los requerimientos funcionales, que una vez identificados, aclaran el panorama en torno a las funcionalidades más importantes del *software* que se planea construir; no se puede obviar los no funcionales, que, si bien es cierto, se pueden tomar como secundarios, no tenerlos en cuenta es un error; es decir, los requerimientos se deben tratar como un todo, que se necesita satisfacer a cabalidad y por completo. | Identificados  Panorama  Secundarios  Error  Todo |
| **5** | A medida que se hable aparecen imágenes que refuercen la idea transmitida.  a business woman presses an index finger on the virtual button, vector illustration  Imagen de referencia (o similar)  <https://acortar.link/Weo7Fi> 228130\_i13  Una empresaria sonriente señala notas pegajosas con tareas de planificación de números. Las mujeres felices priorizan hacer lista, compartir horario de trabajo. Concepto de gestión del tiempo. Ilustración vectorial.  Imagen de referencia (o similar)  <https://acortar.link/ZeOES4> 228130\_i14  man user smartphone bubble speech social media  Imagen de referencia (o similar)  <https://acortar.link/kbMQzO>  228130\_i15 | N/A | Ahora bien, otro detalle no menor, corre por cuenta de la priorización de los mencionados requerimientos, se debe tener claro el orden o, dicho de una forma más coloquial, el lugar o puesto de cada uno de los requisitos que ocupa en la “cola” para su construcción. En este mismo sentido, ese puesto o lugar, se debe definir aplicando las técnicas existentes, desde las más sencillas, como, por ejemplo, una simple votación entre usuarios, bajo la supervisión y asesoría de los desarrolladores, hasta las más avanzadas y modernas. | Priorización  Construcción  Técnicas existentes |
| **6** | A medida que se hable aparecen imágenes que refuercen la idea transmitida.  Ilustración del concepto de método scrum  Imagen de referencia (o similar)  <https://acortar.link/xiZbpa>  228130\_i16  Desarrollo ágil de software con desarrollador usando la metodología de marco de placa Kanban en computadora. Herramienta de administración de proyectos más flexible para cambios rápidos, orientados al cliente, trabajo incremental, proceso iterativo.  Imagen de referencia (o similar)  <https://acortar.link/ymk1H8>  228130\_i17  Ilustración conceptual de engranajes de trabajo en equipo. Idea de cooperación, unión y colaboración.  Imagen de referencia (o similar)  <https://acortar.link/3aLcXd>  228130\_i18 | N/A | De tal manera que se aterricen en las metodologías ágiles, que, sin menospreciar las tradicionales, ubican más en el contexto actual de la ingeniería de requisitos del siglo XXI, puesto que se trabaja de forma más eficiente y rápida , lo cual permite, mostrar mejores y más efectivos resultados, integrando un equipo de trabajo de manera compenetrado y eficiente, donde fluyen los procesos en total armonía, lo que augura éxito rotundo; dada la vinculación al equipo de desarrollo y los usuarios, respetando y aprovechando de los roles, experiencias, conocimiento y sabiduría, lo que se traduce en una motivación optima, eficaz y productiva; que redunda en forma positiva y directa en los resultados del proyecto. | Metodologías ágiles  Equipo de trabajo  Roles  Proyecto |
| **7** | A medida que se hable aparecen imágenes que refuercen la idea transmitida.  Concepto de seguridad informática e internet  Imagen de referencia (o similar)  <https://acortar.link/eyGO5S>  228130\_i19  Tecnología de inteligencia de negocios y análisis de big data medios mixtos que muestran el concepto de informe de información futurista usando software de computadora para analizar el asesoramiento estratégico de inversión para la toma de decisiones.  Imagen de referencia (o similar)  <https://acortar.link/WOPQHl>  228130\_i20 |  | No obstante, no todo es “color de rosa”, es menester recordar que existen peligros y/o riesgos en todo este proceso, donde existen usuarios que por distintas razones ocultan información e incluso pueden ser hasta involuntarias, pero que al ser mal manejadas, pueden destruir el proyecto, razón por la cual, es necesario el entrenamiento efectivo y completo del desarrollador. | Peligros  Usuarios  Desarrollador |
| **Nombre del archivo** | **228130\_v1** | | | |

Desarrollo de contenidos

1. **Requisitos de *software***

|  |  |
| --- | --- |
| **Tipo de recurso** | Tarjetas Conectadas |
| **Introducción** | A continuación, encontrará la definición y elementos a tener en cuenta de los requisitos de *software*. |
| Imagen de referencia (o similar)  <https://acortar.link/SqOBOg>  228130\_i21 | |
| Conversación con los bots de charla. Persona que utiliza el servicio de atención al cliente en línea con chat bot para obtener ayuda. Inteligencia artificial y tecnología de automatización de software CRM. Asistente virtual en internet.  Imagen de referencia (o similar)  <https://acortar.link/N0RyOs>  228130\_i22 | **Definición**  Los requisitos *software* son la definición de las características y funciones del sistema planteado. Dichos requerimientos anuncian lo que los clientes necesitan o pretenden, dejan claro sus expectativas; no obstante, en algunas ocasiones, los mencionados requisitos pueden quedan ocultos a la vista de los interesados, pero en la retina de los usuarios, contrario sensu, la mayoría de los requisitos son obvios, incluso para el mismo cliente. Algunas lecturas cambian la palabra “requisitos” por “servicios”, pero realmente se refieren al mismo concepto. |
| Descripción del requisito de software descripción ilustración abstracta del vector de concepto. Descripción del sistema de software, herramienta ágil, análisis de negocios, especificaciones de desarrollo de proyectos, metáfora abstracta de documentos.  Imagen de referencia (o similar)  <https://acortar.link/O1d9pz>  228130\_i23 | **Para tener en cuenta**  Haciendo una síntesis de las características de los requerimientos, se puede acotar lo siguiente :   * Deben ser específicos y concretos, no puede existir ambigüedad alguna. * Deben ser realizables y aterrizados; puesto que se construyen servicios que son irrealizables. * Ser pertinentes, dado a que los requisitos deben centrarse en el caso de estudio actual y no tratar de solucionar otras situaciones que no aplican al caso. * Priorizar los requerimientos, es decir, no se puede medir de la misma forma todas las peticiones, lo normal es que para cumplir algunos, ya se tengan otros ejecutados . |

1.1 **Requisitos funcionales**

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Slider Presentación | |
| **Introducción** | En el siguiente ítem, encontrará información sobre los requisitos funcionales de un proyecto de desarrollo de *software*. | |
| **Definición**  Los requerimientos funcionales de un proyecto de desarrollo de *software* son los que describen todas las actividades que el sistema debe realizar, es decir, definen los comportamientos o funciones, particulares o generales del mencionado procesador en determinados momentos y condiciones específicas. Generalmente, estos requerimientos funcionales, deben incluir procesos cumplidos por formularios, interfaces, descripciones de los flujos de trabajo, cálculos, rutinas, etc., y en general todas las exigencias del negocio que el sistema debe cumplir.  En otras palabras, los servicios funcionales se definen como declaraciones del trabajo que realizará el *software* o los resultados que debe brindarnos el mismo ante el ingreso de determinados datos; aclarando que el término “datos de entradas”, no necesariamente se refiere exclusivamente a la información digitada por los usuarios, también puede ser respuestas automáticas de otros desarrollos e incluso procesos predefinidos. Lo anterior, sin mencionar que, en muchas ocasiones, los requisitos funcionales establecen clara y explícitamente lo que el sistema no debe o no puede hacer. Esto no se concretar como definición alternativa, dejando claro el concepto de requerimiento funcional, mediante una declaración negativa de lo que no puede pasar; lo cual lo convierte en algo correcto y necesario determinar cuándo se presente. | | Análisis de requisitos en el desarrollo de negocios o sistemas, creación de requisitos de software y especificación que describe la tarea de usuario en el documento con equipo  Imagen de referencia (o similar)  <https://image.shutterstock.com/image-vector/requirement-analysis-business-system-development-600w-1805229358.jpg>  228130\_i24 |
| **Falencias o fallas**  Un elevado número de proyectos de desarrollo, presentan falencias o fallas en su construcción por especificaciones de requisitos inexactas; los encargados de construir y/o determinar los requisitos funcionales, pueden tomar decisiones y hasta suposiciones basados en la propia experiencia o de situaciones anteriores o generales o, en el peor de los casos, de por sentado algún comportamiento que no corresponde al caso de estudio actual: aún más, puede pasar que el profesional encargado, defina de forma ambigua o demasiado simple un requerimiento, pensando más en una solución fácil o rutinaria, en aras de simplificar su implementación. | | Problemas de software aislaron ilustraciones de vectores de dibujos animados. Un hombre enojado tiene problemas con el software de computadora, la tecnología de TI, el mantenimiento de hardware, la línea de soporte necesaria, la caricatura del vector de error del sistema.  Imagen de referencia (o similar)  [https://image.shutterstock.com/image-vector/*software*-problems-isolated-cartoon-vector-600w-2177600167.jpg](https://image.shutterstock.com/image-vector/software-problems-isolated-cartoon-vector-600w-2177600167.jpg)  228130\_i25 |
| **Posibilidades para construir requerimientos funcionales**  Definir los datos de entrada completos y bien validados.  Determinar todas y cada una de las operaciones a realizar en cada formulario, interface, etc.  Resolver de manera clara y concisa los reportes requeridos.  Precisar roles de usuario, especialmente para determinar quiénes y cuando pueden digitar datos al *software*.  Establecer otros datos de salida (diferente a reportes) requeridos por el usuario.  Detallar la forma de cumplir con estándares internaciones y/o de diversos tipos. | | Smartwatches aplicaciones móviles desarrollo ilustración abstracta del vector concepto. Software para dispositivos viables, equipo de desarrollo de aplicaciones, requerimiento técnico de aplicaciones, metáfora abstracta de interacción entre plataformas.  Imagen de referencia (o similar)  <https://image.shutterstock.com/shutterstock/photos/2113315622/display_1500/stock-vector-smartwatches-mobile-apps-development-abstract-concept-vector-illustration-wearable-devices-2113315622.jpg>  228130\_i26 |
| **Ventajas**  Ayuda en la verificación, es decir que la aplicación realmente está proporcionando todas las funciones requeridas y prometidas.  Ayuda en la definición de las funcionalidades de un sistema o en su defecto de sus subsistemas.  Permite completar las necesidades planteadas y posiblemente faltantes.  Accede a depurar errores cometidos en la obtención de requisitos, en aras de afinar los resultados finales.  Posibilita cumplir con los objetivos propuestos, así como las actividades planteadas y las tareas correspondientes .  Adhiere en forma clara y precisa los procesos a sistematizar y con ello a determinar los comportamientos planeados y esperado del sistema. | | Smartwatches aplicaciones móviles desarrollo ilustración abstracta del vector concepto. Software para dispositivos viables, equipo de desarrollo de aplicaciones, requerimiento técnico de aplicaciones, metáfora abstracta de interacción entre plataformas.  Imagen de referencia (o similar)  [https://image.shutterstock.com/image-vector/*software*-tester-concept-application-website-600w-2185160713.jpg](https://image.shutterstock.com/image-vector/software-tester-concept-application-website-600w-2185160713.jpg)  228130\_i27 |
| **Clasificación**  **P**odemos clasificar los requerimientos funcionales en los siguientes tipos:   * Auditoría * Interfaces internas y externas * Histórico de datos * Transaccionales * Certificación * Reglas del negocio * Roles y permisos de usuarios * Gestión legal | | Banca digital, pago por internet, compras en línea, concepto de tecnología financiera. Mujer que usa teléfono móvil y tarjeta de crédito que paga a través de una aplicación de banca móvil para comprar en línea con iconos de tecnología  Imagen de referencia (o similar)  <https://image.shutterstock.com/image-photo/digital-banking-internet-payment-online-600w-2037588638.jpg>  228130\_i28 |
| **Ejemplos**  **Con base** en lo anterior, es posible construir ejemplos de requerimientos funcionales:  El sistema emitirá un mensaje de texto y/o correo electrónico cuando se realice alguna de las siguientes transacciones: pedido de un cliente, despacho de mercancía, emisión de factura y registro de pago .  El sistema de ventas virtual permitirá a los usuarios, adicionar, modificar y consultar las ventas de los clientes activos.  El sistema de información tendrá un botón de pagos, conectado con PSE que permitirá realizar pagos electrónicos.  El sistema enviará automáticamente un correo electrónico a los interesados cuando se cree una nueva requisición.  El sistema le asignará un número consecutivo, único e irrepetible a cada pedido, para identificarlo en todos los procesos del inventario.  El campo ‘Neto a Pagar' únicamente acepta valores decimales con dos posiciones.  El campo ‘fecha de nacimiento’ únicamente acepta una fecha que indique que el trabajador tiene entre 18 y 65 años, rango de edad determinada por políticas de la empresa.  Los clientes tendrán disponible únicamente los productos con *stock* suficiente.  El administrador del sistema podrá ver las ganancias, perdidas y principales balances contables en tiempo real. | | Concepto de tecnología de cadena de bloqueo con diagrama de cadena e interfaz de bloques cifrados. Empresario trabajando en tableta digital, ordenador portátil, desarrollo de software con tecnología moderna en pantalla virtual  Imagen de referencia (o similar)  <https://image.shutterstock.com/image-photo/blockchain-technology-concept-chain-diagram-600w-1176021937.jpg>  228130\_i29 |
| **Para tener en cuenta**  Posibles errores en la construcción de requisitos funcionales:   * Evite copiar necesidades externas o generales de artículos, libros, etc., puesto que estos deben ser personalizados en cada sistema. * Evite adicionar información, a menos que sea estrictamente necesario, ya que esto puede generar dudas o confusiones. * Evite omitir detalles técnicos en los requisitos funcionales. | | Una molesta mujer de negocios que tiene problemas con la computadora, una mujer loca indignada por una falla en una laptop, perdió datos importantes luego de un error fatal crítico, mensaje de caída de PC, pérdida de información no guardada  <https://image.shutterstock.com/image-photo/angry-annoyed-businesswoman-having-problem-600w-692204311.jpg>  228130\_i30 |

* 1. **Requisitos no funcionales**

|  |
| --- |
| Cuadro de texto |
| Conozca qué son y cómo funcionan los requisitos no funcionales de un proyecto de desarrollo de *software*. |

|  |  |
| --- | --- |
| **Tipo de recurso** | Acordeón tipo 1 |
| **Introducción** | Los requisitos no funcionales son características, cualidades o restricciones del *software* que no determinan una funcionalidad del sistema que se construye; es decir, no son una especificación del usuario, ni un pedido del cliente, sin embargo, influyen en el funcionamiento del *software*. En otras palabras, los ‘requisitos no funcionales’ no definen lo que el proyecto *debe* hacer, sino *cómo* lo debe hacer, por lo tanto, son cuantificables, medibles y verificables. |
| Requerimientos no funcionales: Ejemplos - La Oficina de Proyectos de  Informática  Imagen de referencia (o similar)  <http://www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html>  228130\_i31 | |
| **Elementos para construir los ‘requisitos no funcionales’**  **Seguridad:** el activo más preciado de las empresas es la información que debe protegerse realizando acciones como:   * + Control de intentos en la digitación de claves de usuario: después de determinado número de intentos fallidos al ingresar una contraseña, se debe bloquear la cuenta por razones de seguridad y estipular un proceso seguro y controlado para su desbloqueo.   + Control de Cuentas de Usuario: Todos los usuarios del *software* deben tener cuentas personalizadas, con el fin de realizar procesos de auditoría y control de la información efectivos. Con esta acción es posible garantizar la seguridad de la información a través de la creación de roles de usuario.   + Control de contraseñas: es necesario exigir la utilización de contraseñas fuertes que contengan características como un mínimo de 8 caracteres, obligatoriedad de combinar mayúsculas, minúsculas, números y caracteres especiales. Incluso, puede indicarle al algoritmo que genere dichas contraseñas al usuario, si este lo desea. También es importante agregar preguntas de seguridad previamente establecidas por los usuarios en los procesos de recuperación de contraseñas   **Compatibilidad: el mundo está cada vez más conectado, lo cual** obliga a construir aplicaciones compatibles con la mayor cantidad de dispositivos posibles . En consecuencia, se deben diseñar aplicaciones que funcionen en cualquier navegador y dispositivo.  **Usabilidad:** depende de elementos como:   * + Capacitación: Un *software* debe facilitar la vida de sus usuarios, no complicarla, es decir, con entrenamientos cortos y sencillos debe ser posible aprender a manejar el aplicativo.   + Rendimiento: la optimización de algoritmos, script de bases de datos y demás aspectos medibles y comprobables inherentes al desarrollo de *software*, deben mejorar el rendimiento de las aplicaciones .   + Estandarizaciones: utilizar símbolos universalmente conocidos, como el + para indicar adición o la lupa para denotar búsquedas; facilita el uso de las aplicaciones de manera intuitiva.   + **Almacenamiento:** esta característica define un aspecto importante, especialmente para los usuarios, ya que hay aplicaciones que tienen un alto crecimiento en cuanto a registros. Algunas aplicaciones permiten personalizar la configuración de usuario, lo cual es un factor relevante para el almacenamiento.   **Eficiencia:** los servidores que alojan las aplicaciones web requieren gran capacidad de almacenamiento y procesamiento de datos, ya que deben responder muchas solicitudes en cuestión de pocos segundos . | |
| **Ejemplos de requerimientos funcionales**  Cada transacción (clic) debe tener una respuesta del *software* de máximo 5 segundos.  El *software* debe poder trabajar sin problemas hasta con 80 mil usuarios conectados simultáneamente.  Los roles de usuario solo pueden ser asignados por el administrador del sistema.  Realizar copias de seguridad automáticas cada 24 horas y almacenarlas en la nube y en un disco físico fuera de las instalaciones de la empresa .  Todas las claves de usuario deben estar encriptadas.  El sistema debe contar con manuales de usuario ubicados estratégicamente mediante atajos especiales o repositorios comunes .  Las soluciones web deben construirse con la característica de diseño ‘*Responsive’* (adaptable), que implica que el sitio web se adapte al tamaño de pantalla de cualquier dispositivo. | |
| **Para tener en cuenta**  Aunque los requisitos no funcionales son estandarizados, es posible que se presenten situaciones como:  El desarrollo de *software* se debe realizar utilizando una de las herramientas CASE específicas.  La invención del *software* se debe realizar cumpliendo con la norma ISO 9000 que establece los estándares al respecto.  El *software* no puede registrar más de dos fallas de distinta índole cada mes.  La aplicación web debe funcionar para cualquier sistema operativo y en cualquier navegador. | |

**1.3 Estándares**

|  |
| --- |
| Cuadro de texto |
| Un estándar, es un modelo, una referencia o un patrón a seguir. En este sentido, un estándar de calidad, se refiere a las normas mínimas que se deben cumplir para obtener el más alto nivel de calidad de un producto.  Concepto de estandarización y certificación del control de calidad de los estándares.  Imagen de referencia (o similar)  <https://image.shutterstock.com/image-photo/standards-quality-assurance-control-standardisation-600w-1408800641.jpg>  228130\_i32  En 1946, se creó la Organización Internacional de Estandarización, más conocida como ISO por sus iniciales en inglés. Esta institución nació de la unión de dos organismos que existían anteriormente, el primero de ellos el *International Federation of National Standarzing (ISA*); y el segundo, la *United Nations Standards Coordinating Committee (UNSCC*), cuya reunión inicial se realizó en el Instituto de Ingenieros Civiles de Londres, con una participación de 64 delegados representantes de 25 países.  Todo lo anterior dio como resultado la creación de las normas ISO, una serie de estándares con reconocimiento mundial e internacional, cuyo objetivo principal es ayudar a las empresas a establecer determinados niveles de igualdad y homogeneidad en lo relacionado directamente con su gestión, desarrollo de productos de diversas industrias e incluso, prestación de servicios. Han creado más de 23.000 estándares que cubren diversidad de áreas del saber y de gestión, tecnologías y procesos industriales y de producción. La ISO es una organización no gubernamental, presente en 164 países, con 781 comités y subcomités técnicos dedicados a el desarrollo de estándares, con sede principal en Ginebra, Suiza. |

|  |
| --- |
| Cuadro de texto |
| A continuación, se dan a conocer las normas aplicadas al desarrollo de *software* más comunes |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Infografía interactiva Punto caliente | |
| **Texto introductorio** | Uno de los temas más importantes de recolección de requisitos son los estándares que se trabajan en esta industria y mediante esta herramienta, se presentan las más usadas. | |
| **Imagen**  Garantizar la calidad del software y los datos  ISO 9001-3  ISO 9004-2  ISO/IEC 33000  ISO 9001  Imagen de referencia  [https://revista.aenor.com/370/garantizar-la-calidad-del-*software*-y-los-datos.html](https://revista.aenor.com/370/garantizar-la-calidad-del-software-y-los-datos.html) | | |
| **Código de la imagen** | 228130\_i33 | |
| **Punto caliente 1** | ISO 9001: Dedicado al diseño en el desarrollo de un producto | ISO 9001 |
| **Punto caliente 2** | ISO 9000-3: Documento específico que aplica el ISO 9001 al desarrollador de *software*. | ISO 9001-3 |
| **Punto caliente 3** | ISO 9004-2: Registro concreto que nombra las directrices a tener en cuenta para brindar todas las facilidades en el *software*, específicamente en campos como el soporte de usuarios. | ISO 9004-2 |
| **Punto caliente 4** | ISO/IEC 33000: Calidad de los procesos de desarrollo de *software* | ISO/IEC 33000 |

|  |
| --- |
| Cuadro de texto |
| En este capítulo se profundizará en la norma ISO/IEC 33000 |

|  |
| --- |
| Cajón de texto de color |
| La norma ISO/IEC 33000 hace referencia a la "Calidad de los procesos de desarrollo de *software*" que representa a un conjunto de normas internacionales que reemplaza a la ISO 15504 "Evaluación y Mejora de la capacidad y madurez de procesos"; SPICE ISO/IEC 33000 es una actualización de la serie ISO 15504. La norma ISO 33000 es un proyecto de *Software* llamado *Process Improvement and Capability Determination* (SPICE) dirigido fundamentalmente a empresas de desarrollo de *software* y a organizaciones que subcontratan esos servicios.  ISO 33000  Imagen de referencia  <https://www.iso33000.es/>  228130\_i34  La norma ISO 33000 aporta líneas de trabajo coherentes para la evaluación de procesos *software*, *Grupo ACMS Consultores*, (2019)  ¿Qué establece la Norma ISO 33000 “Calidad de los procesos de desarrollo de *software*”?  Proporciona la base para realizar evaluaciones de capacidad en los procesos de *software* y permite medir los resultados obtenidos, con dicha información se puede: comprobar la evolución de una empresa a lo largo del tiempo, el seguimiento de productos: oferta y/o demanda en una organización respecto a la competencia y determinar las estrategias de mejora, Grupo ACMS Consultores, (2019). |

**2. Técnicas de recolección de requisitos**

|  |
| --- |
| Cuadro de texto |
| Consisten en estudiar la forma de conseguir información de los clientes o usuarios, para el desarrollo un proyecto de *software*. Es decir, se enfoca en la técnica que extrae los insumos para dar inicio al esquema. Para lograrlo es indispensable tener competencias sólidas para recolectar la información base, lo que abarca interpretar lo que desea los interesados, contactar al equipo de desarrollo, este proceso se conoce como elicitación.  7 Técnicas de levantamiento de requerimientos software - La Oficina de  Proyectos de Informática  Imagen de referencia (o similar)  <http://www.pmoinformatica.com/2016/08/tecnicas-levantamiento-requerimientos.html>  228130\_i35  Es importante tener claridad que los requisitos se pueden obtener de diversas formas, siendo las más comunes trabajar directamente con los involucrados, realizando investigaciones, pruebas y corrigiendo errores. Estos dos últimos aspectos sirven mucho, pero se debe hacer énfasis que la comunicación directa con los actores claves, eslabón fundamental en el proceso.  Las técnicas de recolección de requisitos varían mucho y en algunos puntos se torna subjetivo. los puntos clave que deben tenerse en cuenta se describen a continuación:   * Reuniones grupales * Prototipos * Entrevistas * Cuestionarios * Observación y/o seguimiento |

**2.1 Técnicas tradicionales**

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Pestañas o tabs horizontales | |
| **Introducción** | Dentro de las técnicas de recolección de datos existen las tradicionales, las cuales se describen a continuación: | |
| **Lluvia de ideas** | Consiste en una reunión de trabajo, planificada y estructurada, el objetivo primordial es recolectar todas las ideas acerca de un tema específico. Debe ser abierta con reglas claras que van desde el tiempo, tener un moderador, respetar el uso de la palabra, entre otras pautas establecidas al inicio y de estricto cumplimiento. Se deben fijar los criterios para valorar ideas y establecer un puntaje; al igual que el desarrollo de los temas agendados.  **Para tener en cuenta**  Para que una sesión sea exitosa se debe realizar de la siguiente manera:   * Hacer una introducción obligatoria, bienvenida, contexto y exposición del caso de estudio, conceptos teóricos, planteamiento del problema y ejemplos de casos similares. * La organización y estructuras de las sesiones son fundamentales para el producto final, lo cual obliga a una planeación de todos los aspectos a tratar y de qué forma abordarlos. * Como se ha mencionado el manejo del tiempo debe ser suficiente para analizar y asimilar el tema central; seguidamente se debe dar espacio para la generación y preparación de ideas; luego un debate y finalmente escoger los planteamientos más destacados, este último en su mayoría se hace por consenso o votación. | Lluvia de ideas o toma de decisiones o ilustración conceptual de ideas con carácter masculino con proceso de pensamiento. Ilustración del vector  Imagen de referencia (o similar)  <https://image.shutterstock.com/image-vector/brainstorming-decision-making-idea-conceptual-600w-2180557871.jpg> 228130\_i36 |
| **Talleres de trabajo (*workshop*)** | Se trata de un espacio de trabajo, donde confluyen determinadas personas, que aportan conocimientos, conceptos, opiniones, pensamientos, material de estudio e incluso dudas e inquietudes del tema proyectado para el desarrollo de *software*.  **Pautas a seguir**  Un *workshop* exitoso, debe tener mínimo los siguientes aspectos:   * Material óptimo: Las ayudas audiovisuales, son una herramienta que facilita captar la atención de los asistentes, el material a proyectar se debe seleccionar de forma técnica, sin olvidar que los grupos son heterogéneos por naturaleza, en cuanto a nivel de preparación y asimilación de los conceptos. * Sitio adecuando: Es una experiencia que requiere de mucha concentración, lo cual implica contar un sitio optimo en cuanto a privacidad, comodidad, confianza, no se puede olvidar que la misión a cumplir es de gran importancia y por ende se deben proporcionar los espacios y elementos ídoneos. * Ritmo adecuado: El manejo del tiempo es fundamental, así como realizar pausas activas de manera organizada; la metodología debe ser planeada con antelación y socializada, por ejemplo, cambiando de actividad o de tema-subtema. Sumado a los descansos programados de 15 y hasta 20 minutos, luego de jornadas entre 3 o 4 horas. | Ver el hombro de una mujer de negocios sentado en el escritorio del trabajo mirar la pantalla de la computadora donde un collage de muchas personas diversas involucradas en la actividad de las negociaciones de videoconferencia, concepto moderno de uso de la tecnología de aplicaciones  Imagen de referencia (o similar)  <https://image.shutterstock.com/image-photo/view-over-businesslady-shoulder-seated-600w-1704165919.jpg> 228130\_i37 |
| **Listas de Verificación** | Denominado “*checklist*” o lista de verificación se utiliza para analizar una o más propiedades de un tema en concreto que se somete a consideración de un grupo de observadores. Generalmente esta herramienta se cumple mediante una lista de preguntas con respuestas cerradas. Algunos autores las denominan como “respuestas binarias” ya que se niega (f*alse*) o aprueba (*true)*. Este carácter cerrado de las respuestas, aunque muy criticado y con muchas desventajas, ofrece un nivel de precisión y objetividad muy alto, que se le debe sacar provecho.  **Consideraciones**  Una lista de verificación bien construida debe tener en cuenta los aspectos descritos a continuación:   * Crear el registro respetando detalles como categoría, la escala determinada en cada pregunta y un detalle que parece obvio, pero es necesario mencionar, dibujar las cuadriculas que guíen al usuario en el manejo de la misma. * Determinar los periodos de verificación en forma clara y concisa. * Clarificar el área, tema, subtema, etc., para que al final sea más fácil recoger los objetivos específicos y hacer la verificación * Realizar los cálculos estadísticos y el análisis para llegar a una conclusión efectiva. | **I** Concepto de negocio del servicio de coches y lista de comprobación inteligente del servicio de coches en pantalla virtual, cierre la mano de un hombre que lleve el signo de lápiz en la lista de comprobación y utilice el portátil en la oficina del centro de servicio de coches y la parte automática en la mesa  Imagen de referencia (o similar)  <https://image.shutterstock.com/image-photo/car-service-business-concept-smart-600w-2164099863.jpg> 228130\_i38 |

**2.2 Técnicas ágiles**

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| En la historia de la humanidad, la necesidad de mejora ha sido una constante en cada proceso, lo que ha llevado a la evolución en todos los ámbitos. Fue así como en el año 2001, se comienza a mencionar el término “*ágil*” en el mundo del desarrollo de *software*, dado que existían proyectos que contaban con todas las herramientas modernas para la época: infraestructura, sin embargo, los resultados, aunque eran los esperados, requerían demasiado tiempo, especialmente cuando surgían cambios. Fue así como un grupo de 17 personas, que venían trabajando en desarrollos de *software*, con técnicas tradicionales, se dieron a la tarea de agilizarlos, dando al inicio a otra forma de trabajo, lo cual se conoce hoy como *The Agil Alliance*, una organización que nació específicamente para promover aspectos referentes a la agilidad, surgiendo de esta forma lo que se conoce como el *Manifiesto Ágil.* Dicho tratado se centra en unos postulados, donde sobresalen:   * La comunión permanente con el cliente. * Prima el funcionamiento del *software*. * Es prioridad la respuesta a los cambios. * Priman los individuos involucrados. | |

|  |  |
| --- | --- |
| **Tipo de recurso** | Acordeón tipo 2 |
| **Introducción** | Actualmente hay una gran variedad de tecnologías agiles, siendo las más utilizadas: |
| **SCRUM**  Es una metodología ágil, amplia, efectiva y ordenada que se establece para que varios equipos trabajen mancomunadamente y lograr soluciones optimas en proyectos de alto grado de complejidad. Lo anterior, se basa en el manejo de buenas prácticas para el trabajo en equipo con alta productividad. En el desarrollo de este método colaborativo, se hace una serie de entregables, parciales, constantes y ascendentes, dejando clara la meta o producto final, dándole prioridad a las necesidades más inmediatas que identifique el cliente.    Imagen de referencia o similar  <https://cdnb.ganttpro.com/uploads/2022/01/agil-metodologia-scrum-1024x665.png.webp>  228130\_i39  Entre otras muchas razones, este punto es la razón fundamental para sugerir *Scrum*, herramienta empleada para proyectos complejos, cambiantes o con escasa claridad en su definición, que requieren resultados a la mayor brevedad posible. *Scrum* hace entregas parciales y regulares del producto final, priorizadas por el beneficio que aporta al receptor.  Otro nicho de trabajo para *Scrum*, son los casos en donde el equipo y los clientes, no logran concretar los entregables, por ende, los plazos pactados se alargan, aumentando los costos sin mayor justificación. También se deben tener en cuenta escenarios muy complejos donde la competencia lleva la delantera y requiere una respuesta inmediata ante la situación.  **¿Como funciona *Scrum*?**  Para poder entender el funcionamiento de esta metodología, es necesario conocer la forma de trabajo, basado en lo siguiente:   * *Sprint:* Es el corazón del *Scrum*, se refiere a cualquier interacción realizada por el equipo. Los desarrolladores completarán muchos *sprints* a lo largo del mismo. Se inicia con el compromiso de un hacer determinada tarea y debe finalizar con la prueba real de un entregable. Cada proceso tiene una duración mínima de una semana y máximo 4. * Roles Básicos: Por el funcionamiento mancomunado que se requiere, esta práctica se asocia con el Rugby, teniendo en cuenta el trabajo en equipo que se debe respetar para cumplir con los objetivos planteados. Para lograr este propósito, se requiere personas que desempeñen los siguientes roles:   + - * + *Product Owner*: el propietario del proyecto, comúnmente llamado usuario, es quien vigila que realmente se están cumpliendo con los objetivos.         + Equipo de desarrollo: son los encargados de plasmar el *product backlog* (un listado de todas aquellas tareas que se van a ejecutar durante el proyecto) (*IEBS Business School, 2022*) es el código fuente, con sus consecuentes pruebas, para finalmente convertirlo en un entregable.         + *Scrum Master*: es el líder, quien tiene un contacto permanente con el *product owner*, pero al mismo tiempo es quien protege al equipo de desarrollo de presiones externas, cuenta con la experiencia en la creación de *software*, proyectos, etc., incluso con conocimientos empíricos en administración de personal.         + Iteraciones: ciclo repetitivo, donde a medida que avanza se incrementa el valor del producto, es decir, cada iteración implica un avance y un paso firme y seguro a la entrega final.   **Para tener en cuenta**  En la historia de *Scrum,* existen casos de éxitos, pero también de fracasos.  *Shopify:* Comprendió y le otorgó a esta metodología la importancia para alcanzar el éxito del proyecto, en otras palabras, cumpliendo con todos los preceptos de *Scrum*, dándole libertad de acción al *Scrum Master*, contrataron personas externas para que realizara el trabajo, entendieron con lujo de detalles el rol de *Project Owner* y como complemento ideal, asignaron en forma equitativa las partes del *software* en el equipo, en forma exclusiva. Peralta, (2021)  *Healthcare:* No obstante, no todos los casos han llegado a buen producto, tal fue el caso de *Healthcare.gov*, que formó parte de un proyecto del gobierno americano, que buscaba darle aire de total de transparencia en los temas de seguros sanitarios. No obstante, la implementación de la metodología *Scrum* fue un verdadero fracaso, por múltiples factores, no respetaron pasos obligatorios e imprescindibles, como hacer pruebas semanales de los pequeños resultados con entregas parciales, para minimizar y asimilar errores, a pesar de contratar una cantidad excesiva de consultores, no asignaron el liderazgo del proyecto; fue lanzado por fases, como lo exige Scrum, por su entorno altamente cambiante, lo cual requiere organización semanalmente; sin mencionar que las pruebas finales, se realizaron en un periodo de tiempo demasiado corto, Peralta, (2021) | |
| **Extreme Programming (XP)**  Esta técnica fue la precursora de las metodologías actuales; sin lugar a dudas, uno de los pilares de su éxito consiste en que cada iteración establece el alcance de la próxima, tiene un manejo avanzado de entregas frecuentes y continuas, advirtiendo que cada versión liberada debe ir a producción inmediatamente.    Imagen de referencia  <https://www.digite.com/wp-content/uploads/2019/09/Extreme-Programming-XP-1024x548.jpg>  228130\_i40  **Consideraciones**  Se deben establecer cuatro variables fundamentales para un óptimo funcionamiento de la metodología XP en todos los proyectos de *software*: calidad, tiempo, alcance y costo. Tres de las mencionadas variables, las pueden fijar los clientes o los responsables del proyecto y la cuarta, la establece la división que adelanta el programa. | |
| **Crystal**  En la metodología sobresale por la estimación grupal, necesidades de la empresa, lo cual la hace ligera y versátil; hace énfasis en la eficiencia en el desarrollo de *software*, emplea fórmulas para que sean más rentables, priorizando la seguridad de los entregables; se tiene una comunicación fluida, efectúa constantes entregas, generando completa armonía con el usuario, partiendo del respeto por los puntos de vista.  Metodología Agil: Crystal – Ingenieria de Software I  Imagen de referencia  <https://isi2018.wordpress.com/2018/04/09/metodologia-agil-crystal/>  228130\_i41  **La metodología Crystal, funciona con los siguientes aspectos:**   * Interacción con los usuarios: dependiendo de la complejidad, profundidad y tamaño del proyecto, se recomienda más de dos usuarios opinando acerca de los entregables. * Planificación actual y futura: el equipo planifica el entregable presente y el que sigue. * Holística: Se subdividen equipos netamente funcionales en multifuncionales. * Revisión continua: Los incrementos se componen de iteraciones y de actividades que permiten construir, revisar, probar y controlar los objetivos de cada incremento*. Ingeniería de Software* I, (2018). | |

**2.3 Técnicas de priorización de requisitos**

|  |
| --- |
| Cuadro de texto |
| Como se ha mencionado en capítulos anteriores, en la fase de consecución de información, se debe efectuar un correcto y efectivo levantamiento de requisitos, también es necesario priorizar dichas exigencias; tarea de alta complejidad, por las implicaciones, dado que en su mayoría genera fuertes controversias, incluso, posiciones encontradas, por ende, es de recalcar que se deben tomar decisiones así no todos los actores estén de acuerdo.  Establece la prioridad del trabajo, organiza la lista de tareas que hay que hacer antes y después, concepto de gestión de tareas, joven empresario se las arregla para priorizar la nota pegajosa con el número primero, segundo y tercero.  Imagen de referencia  <https://image.shutterstock.com/image-vector/set-work-priority-arrange-do-600w-1851861454.jpg>  228130\_i42  Las técnicas más utilizadas son: |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Infografía interactiva Punto caliente | |
| **Texto introductorio** | *MosCow*, señala que todos los requisitos son importantes, no se puede demeritar ninguno, algunos son vitales, debido al funcionamiento de los procesos, tornándose obligatorios, lo cual indica, que, sin ellos, el proceso, no se puede ejecutar. Consiste en calificar los requisitos dentro de las categorías: *Must, Should, Could or Would*. | |
| **Imagen**  Priorización de requisitos - Técnica MoSCoW  Must  Should  Could  Would  Imagen de referencia tomada de:[**https://www.overti.es/tecnologia/275-priorizacion-de-requisitos-tecnica-moscow**](https://www.overti.es/tecnologia/275-priorizacion-de-requisitos-tecnica-moscow) | | |
| **Código de la imagen** | 228130\_i43 | |
| **Punto caliente 1** | *Must* – Imperativo, obligatorio, forzoso | Must |
| **Punto caliente 2** | *Should* – Muy alta prioridad | Should |
| **Punto caliente 3** | *Could* – Importante, pero realmente no necesario | Could |
| **Punto caliente 4** | *Would* – Se puede aplazar para futuras ejecuciones | Would |

|  |
| --- |
| Cuadro de texto |
| En el siguiente video se puede conocer más sobre la técnica de *MosCoW,* es importante prestar mucha atención a las explicaciones que se detallan en allí:  **Video Técnica MosCoX.**  **CF18\_** **Construcción del informe de requisitos** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tipo de recurso** | Video clase/tutorial | | | |
| **NOTA** | **La totalidad del texto locutado para el video no debe superar las 500 palabras aproximadamente** | | | |
| **Título** | Técnica de *MosCoW* | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración** | **Texto** |
|  |  | N/A |  | Ya está grabado por el Experto, ubicado en la carpeta anexos, archivo: 228130\_v2.mp4 |
| **Nombre del archivo** | 228130\_v2.mp4 | | | |
| **Datos del narrador** | Armando Edilberto de Andreis Mattos | | | |

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| **Votación**  Cuando se tienen una cantidad muy grande de requisitos, combinado con posiciones encontradas, una de las formas tradicionales y sencillas es la priorización a votación, dando un puntaje determinado.  **Técnica de clasificación de lista**  Muchos expertos prefieren esta técnica, dado que realmente no requiere mayor preparación ni esfuerzo, simplemente consiste en hacer un análisis simple, es decir, primero hago esto y después aquello. Por ejemplo, vamos a recorrer Colombia desde el norte hasta el sur, entonces comenzamos: Primero por Riohacha, luego Santa Marta, surgen el interrogante ¿puedo ir a Cartagena sin pasar por Barranquilla? si, pero no tiene sentido, primero voy a Barraquilla y luego a Cartagena. Así de simple es la técnica. | |

**2.4 Ventajas y desventajas**

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| **Ventajas**   * Permite establecer una comunicación fluida entre usuarios y desarrolladores, facilitando la comprensión entre sí. * Se eliminan los intermediarios entre las partes, por lo tanto, la información no tiene peligro de distorsión. * Es posible captar incluso hasta el lenguaje corporal de los clientes, lo cual nos facilita la compresión de gestos, tonos de voz, etc. * Facilita la determinación de funciones, las utilidades que requiere el sistema y la posibilidad de atender las necesidades, de forma temprana, evitando reprocesos. * La cercanía con los usuarios conlleva a un entendimiento optimo, que es fundamental en el modelo de desarrollo, incluso a la hora de realizar todo tipo de pruebas. * Los requerimientos que se obtienen son más confiables, estables y veraces. * Al mejorar la calidad de los requerimientos, se eleva la calidad del producto final.   **Desventajas**   * Algunos usuarios tienen la tendencia a la mentira. * Posibles problemas de comunicación y expresión con los clientes. * La mala interpretación de las ordenes entre los involucrados puede entorpecer el proyecto. * En ocasiones, se requiere un reentrenamiento a los desarrolladores, en aspectos como el levantamiento de requisitos. * Algunas veces las entrevistas generan un numeroso volumen de información y requiere de mucha habilidad para organizarla y no olvidar detalles importantes. * Se puede obtener información repetida o excesivamente redundante, cuya revisión requiere demasiado tiempo. * Se requiere excelente entendimiento con los usuarios, independiente de factores externos. * Puede resultar tedioso y muy difícil vincular a patrocinadores y/o dueños, que se requieran, por la dificultad de sus agendas. | |

**3. *Viewpoint***

|  |
| --- |
| Cuadro de texto |
| Se presenta a continuación el concepto de *Viewpoint* y su campo de aplicación, así como los diferentes términos que se pueden emplear y determinan los privilegios de las antes mencionadas funciones, basadas principalmente en el concepto de los *stakeholders.* Medianero, (2015) |

**3.1 Concepto y aplicación**

|  |
| --- |
| Cuadro de texto |
| El modelo se cimienta en la integración de enfoque con puntos de vista, lo cual indica que la base fundamental es la metodología de *AORE*. Inicialmente, se enfoca en las perspectivas de los actores, según las funcionalidades que desempeñan, haciendo posible un detallado y óptimo tratamiento. *El Viewpoint*, facilita funciones a partir de los requerimientos del sistema, posteriormente, se clasifica las visibilidades por estas funcionalidades, se sigue realizando diagrama de casos y escenarios, así como también, la individualización de los requerimientos no funcionales.    Imagen de referencia  <https://slideplayer.es/slide/1675310/>  228130\_i44 |

**3.2 Modelos**

|  |  |
| --- | --- |
| **Tipo de recurso** | Acordeón tipo 1 |
| **Introducción** | El concepto de modelo nace de la necesidad de mirar desde distintas perspectivas al *software* que se planea desarrollar, es decir, cuando se realiza el levantamiento de requisitos aplicando las técnicas que se están estudiando, dos usuarios del mismo proyecto pueden mirar y dar una visión distinta. Luego se debe presentar la perspectiva que realmente sea del interés del usuario y lleva a la correcta construcción del *software*. Fue así como surgió el modelado de sistemas de información.  Existen en el mercado distintas herramientas de modelado, sin embargo, es obligatorio centrarse en la herramienta universal, es decir, la que fue creada específicamente para estos menesteres: UML (lenguaje unificado de modelado, por sus siglas en inglés, *Unified Modeling Language*), es el lenguaje de modelado más utilizado.  Para conocer más sobre el UML se le invita a visualizar el video anexo. |
| Sistema de administración de documentos o DMS para abreviar. Un consultor de TI está utilizando un portátil. Modelo teórico de Internet. Software de automatización de archivos y administración de archivos de información.  Imagen de referencia o similar  <https://image.shutterstock.com/image-photo/document-management-system-dms-short-600w-2135798499.jpg>  228130\_i45 | |
| **Lenguaje de modelado**  Consiste en emplear una imagen, símbolo o una figura; ya aterrizando en el mundo de la ingeniería de requisitos, es la representación de un objeto, sistema o idea, desde diversos ángulos, pero siempre refiriéndose a la misma entidad. Luego entonces, surge la pregunta, ¿por qué se utiliza un modelo? Se emplea para tratar de entender, y explicar, cualquier sistema de información; es decir para una demostración exacta de un sistema, e incluso una abstracción de todas y cada una de las propiedades. Esta forma de trabajar, es una tendencia que se utiliza desde tiempos remotos, ha ido mejorando y tiene como punto de partida, la representación de ideas nuevas, que rompen paradigmas. Lo determinante en un modelo es que describa el proceso a estructurar o ya sistematizado, en forma completa y suficiente, que represente el comportamiento del sistema, lo que implica que las características del mismo coincidan totalmente con las características del desarrollo.  En un lenguaje de modelado está compuesto por vistas, símbolos, diagramas, etc., y, por supuesto, las directrices específicas que explican la forma de utilización; sin perjuicio de la lógica normal que cada lógica de modelado requiera. Existen procedimientos explícitos para construir el pensamiento y el accionar de todos y cada uno de los individuos, recordando que este método es el que realmente guarda las acciones que se debe realizar y el momento de ejecutarlas. | |
| **Herramientas de modelado**  La cantidad de información que se produce diariamente en el mundo es realmente asombrosa, llegando al punto que hablar de un crecimiento exponencial, es poco. Por tal razón, se requieren formas que permitan el manejo de la información, disponible y de especial importancia para las organizaciones, que, por ejemplo, admiten tomar decisiones que mejoren y faciliten la vida a sus usuarios. Todo lo anterior desemboca en la necesidad de aprovechar el reporte, de la manera más adecuada, para lo cual se requiere modelar dichos datos en forma correcta. Las principales herramientas son:   * Erwin Data Modeler: El *software* de modelado de datos número 1 del mundo. Es una herramienta galardonada y utilizada para encontrar, visualizar, diseñar, desplegar y estandarizar activos de datos empresariales de alta calidad. reducen la complejidad, facilitando el diseño, el despliegue y la comprensión de las fuentes de información para satisfacer las necesidades del negocio, *Apiumhub,* (2020). * Moon Modeler: Herramienta de modelado de datos para MongoDB, PostgreSQL, MySQL, MariaDB, SQLite y GraphQL. Diseñada para dibujar diagramas ER, *Apiumhub,* (2020). * Toad Data Modeler: Con este instrumento podrá visualizar las estructuras de las bases de datos y comprenderlas mejor, al igual que las relaciones existentes; también instaura modelos de datos lógicos y físicos de alta calidad, se pueden conectar múltiples bases de datos de forma nativa y simultánea, incluyendo *Oracle, SAP, MySQL, SQL Server, PostgreSQL, DB2, Ingres y Microsoft Access*, Apiumhub, (2020). * UML: fue creado en octubre de 1994, gracias a una reunión de entes mundiales que inventaron una estandarización que necesitaba del desarrollo de *software*, está compuesto por una serie de diagramas:   Estructurales:   * + Diagrama de clases   + Diagrama de componentes   + Diagrama de despliegue   + Diagrama de objetos   + Diagrama de paquetes   + Diagrama de perfiles   + Diagrama de estructura compuesta * Diagramas de comportamiento   + Diagrama de actividades   + Diagrama de casos de uso   + Diagrama de máquina de estados * Diagramas de interacción: Realmente son una subdivisión de los anteriores, compuestos por:   + De secuencia   + De comunicación   + De tiempos   + Diagrama global de interacciones | |
| **Control de versiones**  Al igual que en la vida cotidiana, en los proyectos de desarrollo de *software* se busca la mejora continua, optimizar los productos; lo anterior es una tarea cotidiana y habitual. Esto aplica para el código fuente, también para los artefactos que se construyen en el ciclo de vida de fabricación del producto. En consecuencia, se requiere el llamado versionamiento o control de versiones, que consiste en identificar y controlar los cambios en todas las partes durante el ciclo de vida del *software*, incluido el código fuente. La forma más común de realizar esta actividad es asignándoles tres números seguidos de puntos, por ejemplo A.B.C. Donde:   * El primer número, ejemplarizado con la letra A, apunta a la versión mayor, es decir, la principal del *software*. Por ejemplo: 1.0.0. * El segundo número, simbolizado con la letra B, perfila a la versión menor, es decir, que en esta existen nuevas funcionalidades del *software*, puede ser que se construya un módulo nuevo. Por ejemplo: 1.1.0. * El tercer número, representado con la letra C, señala una revisión del código por algún error o fallo. Por ejemplo: 1.1.1.   En otras ocasiones, se adicionan otros aspectos al versionamiento, puede ser por parche, lo que traduce a agregar un digito adicional, es decir: A.B.C.D., donde D es el número del parche, por ejemplo: 1.1.1.1.  Finalmente, se utiliza la versión por fecha, que consiste en agregarle los datos exactos en que se libera la versión, en formato año, mes y día. Por ejemplo: 1.1.1.22830, donde 22 es el año 2022, 8 es el mes 8 (agosto) y 30 es el día, es decir, que esta versión fue liberada el 30 de agosto del año 2022, *EDteam*, (2017) | |

**3.3 Stakeholders**

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| Este concepto data desde 1984, cuando el filósofo Robert Edward Freeman, lo definió como los grupos de interés en las empresas, en otras palabras, son personas que de una u otra forma, hacen parte de los procesos de las organizaciones; en términos de la ingeniería de requisitos, es un individuo que será parte integral del *software* que se planea construir. Existen dos categorías principales de *Stakeholders*:   * Primarios: Se trata de sujetos con el rotulo de imprescindibles para el normal funcionamiento de la empresa, específicamente en el ámbito económico como los accionistas, dueños, clientes, trabajadores. * Secundarios: No tienen nada tienen que ver con la organización, pero que de una u otra forma, se ven afectados o beneficiados, aplica para la competencia, sindicatos, partidos políticos y en general toda la comunidad.   Icono de línea de vectores de partes interesadas  Imagen de referencia  <https://image.shutterstock.com/image-vector/stakeholder-vector-line-icon-600w-596464301.jpg>  228130\_i46  ¿La pregunta que surge es, como se identifican los Stakeholders?   * El primer paso es reconocer a los interesados en el proyecto, bien sean actores internos o externos a la empresa. * Seguidamente establecer de manera clara los intereses de la organización, en cuanto al proyecto planteado. * Posteriormente se fija el interés de cada uno de los Stakeholders equiparando el primer paso, con relación a lo planteado por la entidad, definiendo la influencia, peso específico y preponderancia. * Para finalizar se clasifican por niveles.   La traducción literal al español de *Stakeholders* es inexacta, debido a que varía la interpretación dependiendo del área del estudio o contexto, puede ser un interesado, pero también puede ser un afectado por el proyecto. | |

|  |  |
| --- | --- |
| **Tipo de recurso** | Síntesis |
| Tecnólogo en implementación y gestión de bases de datos  Síntesis: construcción del informe de requisitos | |
| **Introducción** | La construcción del informe indica la forma de realizarlo, bajo los mejores y más eficientes estándares,  como se muestra en el siguiente mapa conceptual |
| **Figura 1**  *Construcción del informe de requisitos*  Diagrama  Descripción generada automáticamente  Imagen anexa 228130\_i47 | |

**Actividad didáctica**

|  |
| --- |
| Cuadro de texto |
| El archivo de la actividad didáctica se encuentra en la carpeta Anexos / Actividad interactiva CF18. |

**Material complementario**

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo de recurso | Material complementario | | |
| Tema | Referencia APA del material | tipo | Enlace |
| Ingeniería de Requerimientos | Pressman, R. (2010). *Séptima Edición.* <https://www-ebooks7-24-com.bdigital.sena.edu.co/stage.aspx?il=&pg=&ed> | Libro | https://www-ebooks7-24-com.bdigital.sena.edu.co/stage.aspx?il=&pg=&ed= |
| Ingeniería de Requerimientos | *Sommerville, I (2011. Novena Edición.*  [*https://www-ebooks7-24-com.bdigital.sena.edu.co/?il=3313*](https://www-ebooks7-24-com.bdigital.sena.edu.co/?il=3313) | Libro | https://www-ebooks7-24-com.bdigital.sena.edu.co/?il=3313 |
| Requerimientos | *Cervantes Maceda, H. (2016). Primera Edición.* [*https://www-ebooks7-24-com.bdigital.sena.edu.co/?il=1983*](https://www-ebooks7-24-com.bdigital.sena.edu.co/?il=1983) | Libro | https://www-ebooks7-24-com.bdigital.sena.edu.co/?il=1983 |
| Ingeniería de Requerimientos | Pressman, R. (2010).  *Séptima Edición.* <https://www-ebooks7-24-com.bdigital.sena.edu.co/stage.aspx?il=&pg=&ed>= | Libro | https://www-ebooks7-24-com.bdigital.sena.edu.co/stage.aspx?il=&pg=&ed= |

**Glosario**

|  |  |
| --- | --- |
| **Tipo de recurso** | Glosario |
| MoSCoW: | técnica de priorización de requisitos que consiste en calificar dichos requisitos dentro de las categorías: Must, Should, Could or Would. |
| Ingeniería del *software*: | es una parte importante de las ciencias de la computación, o una ramificación de ella, que estudia la construcción de *software* seguro y de alta calidad, basándose en métodos y técnicas de ingeniería, además, brinda un efectivo soporte operacional y de otra índole, donde sobresale el mantenimiento. |
| Scrum: | es una metodología ágil, muy amplia, en la que se establece una forma efectiva y ordenada para que varios equipos trabajen mancomunadamente para lograr soluciones óptimas en proyectos muy complejos. |
| Investigación cualitativa: | es el método científico de observación directa para extraer o recepcionar datos no numéricos. ​ |
| ViewPoint | se cimienta en la integración de enfoque con puntos de vista. |
| Prototipo: | consiste en una representación supuesta pero concreta y muy aterrizada de una parte específica o la totalidad de un proyecto, idea de negocio, e incluso, un producto o servicio. |
| Requerimiento: | la funcionalidad que se espera tenga un determinado *software*. |
| Roles: | funciones que un individuo desempeña en un proceso, solución, proyecto, situación, etc. |
| Stakeholders: | se trata del participante, integrante, parte interesada, puede ser un individuo, una empresa u organización, que tiene cualquier tipo de participación directa, de forma interna o externa en un proceso que estudiamos y pretendemos sistematizar. |
| Técnicas: | hace referencia a todos los procedimientos que se usan para lograr un resultado determinado. |

**Referentes bibliográficos**

|  |  |
| --- | --- |
| **Tipo de recurso** | **Bibliografía** |
| **Página web** | *Apiumhub*, (2016), *Técnicas de priorización de requerimientos de software que debes saber*. [https://apiumhub.com/es/tech-blog-barcelona/priorizacion-requerimientos-*software*/](https://apiumhub.com/es/tech-blog-barcelona/priorizacion-requerimientos-software/) |
| **Libro** | Debrauwer, L. & Van Der, F. (2020), *UML 2.5 Iniciación, ejemplos y ejercicios corregidos* (5ª edición).<https://www-eni-training-com.bdigital.sena.edu.co/portal/client/mediabook/home> |
| **Página web** | *Edteam*, (2017), *¿Cómo se deciden las versiones del software?.*[https://ed.team/blog/como-se-deciden-las-versiones-del-*software*](https://ed.team/blog/como-se-deciden-las-versiones-del-software) |
| **Página web** | Grupo ACMS Consultores, (2019), *Norma ISO 33000 Calidad de procesos de desarrollo software*. <https://www.grupoacms.com/norma-iso-33000> |
| **Página web** | IEBS Business School, (2022), Qué es un product backlog y cómo hacer uno.<https://www.iebschool.com/blog/que-es-un-product-backlog-y-como-hacer-uno-guia-scrum-agile-scrum/#:~:text=Un%20product%20backlog%20es%20una,visibles%20para%20todo%20el%20equipo> |
| **Página web** | Ingeniería de *Software* I, (2018), *Ingeniería de software I*.<https://isi2018.wordpress.com/2018/04/09/metodologia-agil-crystal/> |
| **Libro** | Lis, G, & Pantaleo, L. (2018), *Ingeniería de Software. (1ª Ed.),* [https://www-alphaeditorialcloud-com.bdigital.sena.edu.co/reader/ingenieria-de-*software*?location=155](https://www-alphaeditorialcloud-com.bdigital.sena.edu.co/reader/ingenieria-de-software?location=155) fuente SENA, No se puede comprobar, requiere contraseña y usuario |
| Página web | Peralta, E. (2021), *Cómo Funciona la Metodología Scrum*. <https://www.genwords.com/blog/metodologia-scrum> |
| **Libro** | Pressman, R. & Maxim, B. (2021). *Ingeniería de software – Un enfoque práctico. Novena Edición.* <https://www-ebooks7-24-com.bdigital.sena.edu.co/?il=16414> fuente SENA, No se puede comprobar, requiere contraseña y usuario. |
| **Libro** | *Sommerville, I (2011), Ingeniería del software. Novena Edición. México:* Pearson Educación |