**Datos de identificación del programa de formación**

|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | Desarrollo de aplicaciones web full stack |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | 220501094. Elaboración de la propuesta técnico-económica del producto o servicio. | RESULTADOS DE APRENDIZAJE | 220501094-01 - Plantear la propuesta técnico-económica de acuerdo a las necesidades del cliente. |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | 4. |
| NOMBRE DEL COMPONENTE FORMATIVO | Contrato y planeación de propuesta técnico - económica |
| BREVE DESCRIPCIÓN | En este recorrido temático, vamos a plantear una propuesta de manera técnico-económica, de acuerdo con las necesidades del cliente y respetando los requerimientos. Lo anterior se hará teniendo en cuenta las normativas, legislaciones y condiciones contractuales de este proceso. Se realiza también una estimación de desarrollo, costos y el talento humano requerido para el proyecto en curso, además del uso de las herramientas para calcular los costos y generar recursos económicos, monetizando un servicio o un producto. |
| PALABRAS CLAVE | Costo, esfuerzo, estimación, métodos, monetización |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | VENTAS Y SERVICIOS |
| IDIOMA | Español |

**TABLA DE CONTENIDOS**

**1. Contrato de *software***

**2. Técnicas de estimación de proyectos de *software***

**3. Licenciamiento**

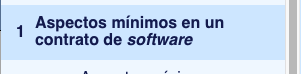
**INTRODUCCIÓN**

|  |
| --- |
| Cuadro de texto |
| Apreciado aprendiz, bienvenido a este componente formativo, donde se estudiará el punto de equilibrio entre la oferta del cliente, el pago al equipo de trabajo y las ganancias para la empresa, en el marco del desarrollo de un *software* web. Afiance sus conocimientos y capacidades para realizar una buena estimación de un equipo y del talento necesarios; sepa calcular, de manera óptima, el costo del proyecto y diseñe con efectividad y acierto, un plan de ventas y monetización adecuados, para el proyecto de *software.*  En el siguiente video conocerá, de forma general, la temática que se estudiará a lo largo del componente formativo. |

**GUION DE VIDEO INTRODUCTORIO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tipo de recurso** | Video spot animado | | | |
| **NOTA** | **La totalidad del texto locutado para el vídeo debe tener mínimo 490 y máximo 510 palabras** | | | |
| **Título** | Contrato y planeación de propuesta técnico económica | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración (voz en off)** | **Texto** |
| **Escena 1** |  | NA | Estimado aprendiz, bienvenido a este nuevo recorrido de conocimiento, a continuación tendremos la oportunidad de explorar una serie de temáticas muy relevantes, entre ellas, las normativas y legislaciones necesarias en el desarrollo de un proyecto de *software*, así mismo, vamos a revisar qué hay detrás del proceso de estimación en el equipo de desarrollo y cómo obtener el talento necesario para el proyecto de manera óptima. | Proyecto de *software* |
| **Escena 2** |  | NA | Lo anterior permitirá tener un soporte sólido a la hora de calcular el valor costo/beneficio de un proyecto de *software*, siguiendo los lineamientos de normativas vigentes establecidas actualmente para la industria del desarrollo de *software*. De esta manera, se asegura un conocimiento completo que, hoy en día, es primordial en la industria de las tecnologías y del *software*. | Industria del desarrollo de *software*  Tecnologías |
| **Escena 3** |  | NA | Se considera muy importante que los contenidos aquí mencionados sean comprendidos de manera clara y a profundidad, debido a que, en este proceso de aprendizaje, esta parte es una pieza fundamental en el desarrollo *full stack*. Los temas aquí relacionados son cruciales si se quiere emprender en estas áreas. En este orden de ideas, tener esta información clara nos va a permitir el desarrollo de un negocio óptimo, claro, integral, con una visión clara de los objetivos y sin ambigüedades, todo esto en busca de la generación de una monetización en proyectos que tengan que ver con la industria del *software*. |  |
| **Escena 4** |  | NA | La estimación de proyectos se puede considerar una parte crucial de la ingeniería de software, ya que hacer estimaciones precisas lleva a un desarrollo exitoso de proyectos de software. Además, en este componente formativo se examinan los principales modelos de estimación disponibles que pueden ser utilizados como base para elegir los modelos a utilizar en las estimaciones de esfuerzo, tiempo y costo en proyectos de software. |  |
|  |  |  | En esta misma línea, se trata de analizar los requisitos, prioridades, objetivos, estrategias del negocio y demás aspectos relevantes para la elaboración de un presupuesto viable tanto para el cliente como para la industria tecnológica que realiza el proyecto. Del mismo modo, se puede decir que esta temática es una parte orientada más al área de negocios, pero, aun así, es un tema primordial para que el estudiante tenga el conocimiento básico al momento de empezar a emprender con sus propios proyectos. |  |
|  |  |  | Por último, cabe anotar que para el buen desarrollo de esta experiencia de aprendizaje de manera satisfactoria, también, es necesario recalcar que, aparte de un presupuesto viable, es necesario tener en cuenta los licenciamientos de *software* a utilizar para ejecutar las actividades de desarrollo, revisar qué aspectos se adaptan mejor a la solución, además de conocer sus respectivos tipos, con el fin de reunir todos los puntos en el círculo de la legalidad de manera que no afecte la venta del producto desarrollado.  ¡Éxitos en esta nueva experiencia! |  |

1. **Contrato de *software***

****

|  |
| --- |
| Cuadro de texto |
| El contrato de *software* es un tipo de contratación que permite que el propietario del programa conceda un derecho de uso a una persona o empresa, a cambio de un intercambio económico. ~~De esta manera, puede tratarse de un contrato de desarrollo de~~ *~~software~~*~~, entre otros~~. |

* 1. **~~Aspectos mínimos en un contrato de~~ *~~software~~***

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Carrusel de tarjetas | |
| **Introducción** | De acuerdo con los intereses de las partes y la modalidad de contrato, las cláusulas que se deben tener en cuenta son las siguientes: | |
|  | | |
| Las partes deben tener la capacidad jurídica necesaria para suscribir o firmar el acuerdo. | |  |
| La exposición de los términos contractuales y, con mayor detalle, los datos técnicos. | |  |
| Debe especificar el objetivo del contrato: permiso para el uso interno o servicio de terceros. | |  |
| Los derechos que se establecen a través del contrato. | |  |
| Responsabilidades y deberes de cada una de las partes implicadas. | |  |
| ~~Responsabilidades y deberes del proveedor.~~ | |  |
| Duración del contrato. | |  |
| Claridad acerca de la exclusividad. | |  |
| Propiedad intelectual (marca, derechos de autor, etc.) e industrial. | |  |
| Formas de pago y costos. | |  |
| Carácter de confidencialidad. | |  |
| Entrega, instalación y compatibilidades. | |  |
| Entrega de documentos y manuales necesarios para el uso del *software.* | |  |
| Garantía y términos de uso. | |  |
| Fiscalidad. | |  |
| Causales de resolución del contrato. | |  |
| Jurisdicción competente y ley aplicable. | |  |
| Adicionales donde se detallen los requisitos, el producto y demás cláusulas adaptadas al contrato. | |  |

**~~1.2~~ Tipos de contratos**

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Slider pasos | |
| **Introducción** | Los tipos de contratos de *software,* se pueden clasificar de la siguiente manera: | |
| **Slide 1** | **Contrato de licencia de *software***  Puede incluir un *software* de código cerrado (licencias de *software* propietario o privativo) o de código abierto. El representante de los derechos (licenciante) autoriza a un tercero (licenciatario), el poder para reproducir o usar un programa de *software* dentro de unas normas. Estos contratos suponen el uso del *software* del propietario, protegido por los derechos de autor; además, no permiten la explotación y transmisión de los derechos del *software,* a menos que se indique en el contrato. Por lo anterior, es primordial especificar detalladamente en las cláusulas, cada uno de los puntos de conveniencia de las partes. Los *software* de código cerrado no incluyen la transmisión del código fuente, dado que no permiten cambios del mismo, mientras que los de código abierto sí lo tienen, permitiendo evolucionar el mismo, según la conveniencia de los usuarios. |  |
| **Slide 2** | **Contrato de desarrollo de *software***  Es un acuerdo legal entre dos o más partes que establece los términos y condiciones para el desarrollo y entrega de un *software*. Este contrato suele incluir detalles como la descripción del *software* que se va a desarrollar, las responsabilidades de cada parte, los plazos de entrega y pago, y cualquier otro aspecto relevante para el desarrollo del *software*. Los contratos de desarrollo de *software* son importantes, porque establecen un marco legal para el proyecto, y protegen los intereses de ambas partes involucradas. |  |
| **Slide 3** | **Contrato de soporte y mantenimiento**  Este tipo de contrato tiene un *software* de inicio (desarrollo o de licencia), y permite que el proveedor asegure el correcto funcionamiento del *software*, tanto en aspectos asociados al buen uso, como en la verificación del correcto funcionamiento. |  |
|  | **Contrato de *Software As A Service* (contrato SAAS)**  Este es el tipo de contrato que incluye una licencia de uso de un *software;* dicho de otro modo, es aquel que es conocido como “la nube” de forma común, hospedado en los servidores del proveedor de uso, de acuerdo con lo establecido en el contrato. |  |
|  | **Contrato de distribución de *software***  En este tipo de contrato, el propietario (licenciante), otorga los derechos de explotación a un tercero (distribuidor), en un espacio establecido contractualmente, para ser distribuido a los usuarios finales (licenciatarios). Este contrato requiere una remuneración inicial. |  |

**~~1.3~~ Normativa y legislaciones en la contratación de *software***

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | **Rutas /Pasos Horizontal** | |
| **Introducción** | El marco normativo y legislativo correspondiente al uso, protección y adquisición de herramientas de *software* es validado por las entidades compradoras para los procesos de contratación de estos bienes y servicios. Son los mismos que incorporan los decretos, leyes, circulares y resoluciones que se citan a continuación (Colombia Compra Eficiente, 2020), sin perjuicio de las disposiciones constitucionales a que haya lugar: | |
| **Ley 23 de 1982** | **Ley 23 de 1982**  La protección jurídica que se concede al *software,* tiene su fundamento en la Ley 23 de 1982, mediante la cual, se establece que: “los autores de obras literarias, científicas y artísticas, gozarán de protección para sus obras”. Ahora bien, la doctrina ha asociado el lenguaje de programación (código objeto y código fuente) a un idioma natural, por lo que el *software,* como herramienta, es asociado a una obra literaria que demanda protección como creación. No obstante, lo anterior desconoce componentes diferentes al mero lenguaje de programación, evitando proteger de manera integral el *software* como invención, o bajo el amparo de una patente. |  |
| **Decreto 1360 de 1989 Aunado a la Ley 23 de 1982** | **Decreto 1360 de 1989 Aunado a la Ley 23 de 1982**  A través de este decreto, se incorpora el concepto de *software* en la legislación colombiana; es ahí donde se define al *software* como una creación propia del dominio literario y, el Artículo 2 contempla que el *software* sujeto de protección, debe comprender alguno de estos 3 aspectos: (i) el programa de computador, (ii) la descripción de programa, y (iii) el material auxiliar. Sin embargo, al igual que la generalidad prevista en la Ley 23, el Artículo 7 del decreto, señala que: “la protección otorgada al soporte lógico no excluye otras formas de protección”, reiterando en la protección jurídica que puede predicarse de otros componentes, relacionados con el *software*. |  |
| **Ley 565 de 2000** | **Ley 565 de 2000**  Por medio de la cual se aprueba el Tratado de la OMPI - Organización Mundial de la Propiedad Intelectual - sobre Derechos de Autor (WCT), adoptado en Ginebra, el veinte de diciembre de 1996, en la cual se establece, expresamente, que los programas de ordenador están protegidos como obras literarias en el marco de lo dispuesto en el Artículo 2 del Convenio de Berna. Dicha protección se aplica a los programas de ordenador, cualquiera que sea su modo o forma de expresión. En el mismo sentido, se refiere a la protección que, por derecho de autor, debe darse a las compilaciones de bases de datos, sea cual sea su modalidad. |  |
| **Decisión 486 de 2000** | **Decisión 486 de 2000**  La Decisión de la Comunidad Andina de Naciones, refiriéndose a los requisitos de la patentabilidad establece, en el Artículo 15, el listado para no considerar como invenciones sujetas a patente, “(e) Los programas de ordenadores o el soporte lógico”, derivando así en la imposibilidad de proteger jurídicamente mediante patente, al *software* como creación. Así mismo, se evidencia una falencia en el desarrollo normativo que contraría acuerdos internacionales, como el Acuerdo sobre los Aspectos de los Derechos de Propiedad Intelectual, relacionados con el comercio y que establece, en el Artículo 27, como materia patentable, lo siguiente:  Sin perjuicio de lo dispuesto en los párrafos 2 y 3, las patentes podrán obtenerse por todas las invenciones, sean de productos o de procedimientos, en todos los campos de la tecnología, siempre que sean nuevas, entrañen una actividad inventiva y sean susceptibles de aplicación industrial.  Dicha limitación contraría la política de gobierno digital, establecida por el Ministerio de las Tecnologías de la Información y las Telecomunicaciones, pues impide la protección de nuevas creaciones en materia tecnológica. |  |
| **Ley 1450 de 2011** | **Ley 1450 de 2011**  En relación con la naturaleza de los contratos de cesión de derechos patrimoniales de autor, el Artículo 30 de la Ley 1450 de 2011, establece que:  Los actos o contratos por los cuales se transfieren, parcial o totalmente, los derechos patrimoniales de autor o conexos, deberán constar por escrito como condición de validez. Todo acto por el cual se enajene, transfiera, cambie o limite el dominio sobre el derecho de autor, o los derechos conexos, así como cualquier otro acto o contrato que implique exclusividad, deberá ser inscrito en el Registro Nacional del Derecho de Autor, para efectos de publicidad y oponibilidad ante terceros. Será inexistente toda estipulación, en virtud de la cual el autor transfiera, de modo general o indeterminable, la producción futura, o se obligue a restringir su producción intelectual o a no producir.  Estos contratos que implican enajenación total o parcial, según lo pactado entre las partes intervinientes en la cesión; del mismo modo, los contratos de cesión de derechos patrimoniales de autor, deben ser inscritos en el Registro Nacional de Derecho de Autor, a efectos de ser oponibles frente a terceros. |  |

**~~2.~~ Cálculo de costos de *software***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tipo de recurso** | Video spot animado | | | |
| **NOTA** | **La totalidad del texto locutado para el vídeo debe tener mínimo 490 y máximo 510 palabras** | | | |
| **Título** | En el siguiente video, se muestran aspectos claves para la el **cálculo de costos de *software***. Visualícelo atentamente: | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración (voz en off)** | **Texto** |
| **Escena 1** |  | NA | En el presente recorrido temático, también es necesario recalcar la importancia que tienen los costos en el proceso de desarrollo de un proyecto de *software* y cómo se ha generado su estimación a partir de diversas investigaciones con focos a mejorar el entendimiento del proceso de desarrollo de *software*, así como a constituir y evaluar las herramientas de estimación de costos del *software*. | Desarrollo de un proyecto de *software*  Estimación de costos del *software* |
| **Escena 2** |  | NA | En este sentido, se puede decir que la estimación de costos requiere la realización de predicciones acerca de la cantidad más probable de esfuerzo, tiempos y características del personal que se requiere para establecer un sistema de *software*. |  |
| **Escena 3** |  | NA | Estos análisis de coste se hacen a lo largo de todo el ciclo de vida y se pueden calcular de dos formas. Por un lado, encontramos las estimaciones preliminares y, por otro lado, las estimaciones detalladas. Las estimaciones preliminares se necesitan para diseñar una oferta o determinar la viabilidad de un proyecto, también, son las más complejas de hacer, las más vinculantes y las menos exactas, dado que se hacen estimaciones “gruesas” del esfuerzo y se divide el esfuerzo total por actividades. |  |
| **Escena4** |  | NA | En cuanto a las estimaciones detalladas, estas son necesarias cuando empieza el proyecto para realizar la planificación. El trabajo y duración de tareas individuales se contrasta con los valores estimados, desarrollando periódicamente reestimaciones a las que se destinan recursos en caso de ser necesario. Adicionalmente, también cabe recalcar que existen muchas formas de abordar la elaboración de las estimaciones, en este componente formativo se van a abarcar algunas de las principales como lo son: |  |
| **Escena5** |  | NA | La opinión de expertos: aquí el desarrollador plantea los criterios del proyecto y los expertos hacen estimaciones con base en su experiencia; así mismo, hacen uso de la Técnica Delphi que posibilita sistematizar y mejorar la opinión de los expertos consultados. |  |
| **Escena6** |  | NA | La analogía: este modelo se orienta, de manera más formal que el anterior. En este caso, los expertos establecen diferencias entre el proyecto propuesto con uno o más proyectos anteriores, buscando similitudes y diferencias específicas.  Las técnicas de descomposición: son las que se hacen sobre cada componente de un *software* o sobre actividades de mínimo nivel en que se dividen las actividades. Estas estimaciones de bajo nivel se juntan para crear la estimación completa del proyecto; es decir, el costo total del proyecto es el resultado de sumar las estimaciones de todos los componentes en los que se ha dividido el proyecto. |  |
|  |  |  | Los modelos empíricos: se puede definir como el conjunto de técnicas que caracterizan los elementos clave que generan una fórmula matemática que asocia estos factores con el esfuerzo. Los modelos se basan en experiencias pasadas.  Los simuladores: son los que están constituidos sobre modelos dinámicos. Permiten copiar el comportamiento del proyecto a lo largo del periodo de tiempo. |  |

1. **Técnicas de estimación de proyectos de *software***

|  |
| --- |
| Cuadro de texto |
| El primer elemento de un proyecto de *software,* es la planificación. En esta etapa del proceso, una de las tareas con mayor valor es la estimación, tanto de recursos, esfuerzos, costos y tiempos para el buen desarrollo del proyecto. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Slider pasos | |
| **Introducción** | Evaluar los elementos que tienen relación con un proyecto (cronograma, esfuerzo, costo, personal, etc.), requiere conocer su alcance para estimar las posibles acciones, contrastar opciones y obtener cálculos de los costos, antes de decidirse por un enfoque determinado. | |
| **Slide 1** | Dentro del entorno de un proyecto de *software,* existen diferentes factores a estimar; por ejemplo, el cálculo del tiempo de ejecución de cada una de las actividades a realizar dentro del proyecto (análisis, gestión, desarrollo, pruebas, etc.), o del proyecto en general, así como las validaciones de los casos de prueba por los riesgos, casos de uso, el esfuerzo a realizar, entre distintos índices estimables. |  |
| **Slide 2** | Cualquier esfuerzo por estimar los tiempos, costos o recursos a largo plazo, implica estimaciones. Cuando se realizan la planificación y la gestión de proyectos, lo primero que se hace es la estimación del tamaño del *software,* para luego calcular el esfuerzo necesario y los posibles cronogramas. |  |
| **Slide 3** | La tarea de estimar los costos suele ser una de las primeras cosas que se hacen en un proyecto, a menudo después de definir los requisitos. Sin embargo, esta tarea se suele realizar de forma regular a lo largo de la ejecución del proyecto con el fin de ajustar la precisión de la estimación inicial. |  |
| **Slide 4** | En otras palabras, estimar es difícil, y una forma adecuada de acercarse a esto puede ser desarrollando los requerimientos de manera incremental, detallando las estimaciones a medida que se obtiene más información. Estimar temprano puede ser peligroso, ya que al comienzo de un sistema, la cantidad de información que se tiene suele ser muy limitada, lo que dificulta el proceso. |  |
| **Slide 5** | Resulta útil tener muy claros los usos principales de la estimación de costos en la administración de proyectos:   * La etapa de planteamiento posibilita indicar la cantidad de personas que se requieren para la ejecución del proyecto y establecer cronogramas correctos. * Durante el desarrollo del proyecto, permite conocer su alcance y calcular si el proyecto se está desarrollando según lo indicado en el cronograma. De lo contrario, se deberá crear un plan de acción a tiempo. Para lo anterior, es necesario contar con métricas que permitan calcular el nivel de cumplimiento del desarrollo del *software*. |  |
| **Slide 6** | En el campo de la ingeniería de software, la estimación de costos se refiere a determinar el alcance del software y la cantidad de personas necesarias para desarrollar el producto. Existe una fuerte conexión entre calidad, costo y cronograma, ya que estos tres aspectos están estrechamente relacionados entre sí. |  |
| **Slide 7** | La complejidad de mejorar la calidad de un software sin aumentar su costo y/o su plazo de ejecución se señala en lo anterior. Como resultado, el plazo de desarrollo no puede reducirse significativamente sin afectar la calidad del producto y/o aumentar el costo de desarrollo. Los modelos de estimación tienen un papel importante en este sentido, ya que permiten equilibrar estos tres factores. |  |

|  |
| --- |
| Cuadro de texto |
| Una historia de usuario es una definición breve, informal y en lenguaje sencillo, de lo que un usuario quiere hacer dentro de un producto de *software,* para conseguir algo que le resulte valioso.  Es necesario resaltar que el proceso de estimación es impredecible dado que se intenta medir elementos como el esfuerzo, tamaño y costos de un proyecto que aún no ha iniciado y del cual es muy probable que se tenga poca información. Es muy importante que el representante de la gestión del proyecto posea buen conocimiento del tema, que tenga en cuenta trabajos pasados para no repetir los mismos problemas anteriores, y sea una persona con características que le permitan prever posibles riesgos. Además, hay que tener presente que la estimación en horas tempranas es de mayor riesgo, por lo que hay que realizar constantemente validaciones y ajustes a las estimaciones ya existentes. Las historias de usuario, suelen seguir el patrón rol – función – beneficio. |

**~~3.1~~ Técnicas de estimación del tamaño del *software***

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Slider Presentación | |
| **Introducción** | Hay diferentes métodos para evaluar el tamaño del software, como por ejemplo: | |
| **Medición de líneas de código fuente producidas (SLOC)**  Es una de las técnicas más conocidas, pero requiere disponer del código, lo que representa una restricción para realizar estimaciones tempranas, debido a que la información que se obtiene en esos momentos y el tiempo del que se dispone para las primeras estimaciones del proyecto, son normalmente escasos. De todas las etapas en las cuales puede realizarse la estimación, cuando menos datos/información se tiene para realizarla, es en el momento inicial, cuando todavía se está revisando la factibilidad del proyecto. Desde el punto de vista del desarrollo y comercialización del *software* específico para terceros, esa estimación temprana tiene gran importancia. Sin embargo, la tarea de estimación no se realiza una sola vez en el proyecto, sino que, a medida que se cuenta con más datos/información, se realizan excelentes estimaciones que permiten una mejor planificación de lo que falta del proyecto. | |  |
| **Puntos de Función (FP)**  El Análisis de Puntos de Función (FPA), provee un enfoque alternativo porque permite estimar el tamaño del *software,* basándose en los requerimientos y luego la cantidad de SLOC se puede evaluar desde el tamaño en Puntos Función (FP). En etapas iniciales del ciclo de vida, se identifican los “Actores” y los “Casos de Uso del Sistema” y se documenta cada uno de estos, mediante una pequeña descripción. Ejecutando el Análisis de Puntos de Función a estos Casos de Uso, se podrá conseguir una estimación de mayor tamaño y, a partir de ella, el esfuerzo ejecutado. Más adelante, cuando se obtengan más datos acerca del sistema, se podrá refinar el análisis. Posteriormente, se aumenta la documentación de cada Caso de Uso, definiendo los escenarios que se producen dentro de los mismos. Un escenario explica la secuencia de pasos que deben efectuar los actores y el sistema durante la reproducción del Caso de Uso. Si se ejecuta nuevamente el Análisis de Puntos de Función sobre estos Casos de Uso con mayor detalle, la estimación del esfuerzo y tamaño será más puntual que la anterior. El FPA calcula el tamaño del *software* determinando la funcionalidad provista al usuario, basándose solamente en las especificaciones funcionales y el diseño lógico. Este método consiste en analizar los puntos de función, teniendo en cuenta:   * Entradas al sistema. * Salidas del sistema. * Consultas. * Grupos de datos lógicos del sistema. * Grupos de datos lógicos que no pertenecen al sistema, pero que el sistema utiliza.   Luego, estos puntos de función se utilizan para estimar el tamaño y esfuerzo del sistema.  La fórmula de Albrecht para calcular los puntos función, es la siguiente:  **FP = UFP x TCF** donde:  UFP: son los Puntos Función No Ajustados  TCF: Factor de Complejidad Técnica.  Los UFP son deducidos mediante la sumatoria de los pesos de todos los parámetros identificados.  Existen varios tipos de metodologías que estiman el tamaño del *software*; por ejemplo:     * IFPUG-FPA *(Function Point Analysis).* * MKII *(Mark II).* * FFP *(Full Function Point).* * NESMA FPA *(Netherlands Software Metrics Users Association Function Point Analysis).*   Este método también puede ser difícil de aplicar al comienzo del desarrollo del software debido a la falta de información sobre el sistema en ese momento | |  |
| **Puntos de Casos de Usos (UCP):** el método UCP clasifica los casos de uso en simple, promedio y complejo con factores de peso 5, 10 y 15, respectivamente. La clasificación se realiza con base en el número de transacciones que contenga el caso de uso, 1 a 3 para simple, 4 a 7 para promedio y 8 o más para complejos. Una transacción es un evento que ocurre entre el sistema que se está modelando y un actor. Una vez que se han especificado los casos de uso del sistema con más detalle, se pueden utilizar las transacciones de puntos de función (consultas externas, entradas externas y salidas externas) resultantes de las secuencias de un escenario para definir la complejidad de los archivos lógicos internos y/o la interfaz externa y obtener una estimación más precisa de los puntos de función.  En resumen, para establecer Puntos Objeto (Casos de Uso) en un proyecto de software, se deben seguir los siguientes pasos:   1. Identificar la cantidad de objetos, estimar el número de reportes, pantallas y componentes que tendrá la aplicación. 2. Clasificar cada objeto según su nivel de complejidad (difícil, medio o simple). 3. Asignar un peso a cada objeto en función de su nivel de complejidad. Estos pesos reflejarán el esfuerzo relativo necesario para implementar cada instancia. 4. Determinar la cantidad total de Puntos Objeto, sumando los pesos de todas las instancias de los diferentes tipos de objetos.   Es importante tener en cuenta que los casos de uso no tienen un tamaño definido, por lo que un método basado en ellos debe tener en cuenta el número de escenarios o transacciones para determinar su peso. A medida que se completen las especificaciones de los casos de uso, se pueden ir ajustando las estimaciones de los puntos de función. | |  |

**~~3.2~~ Técnicas de estimación de esfuerzo (cantidad de personas necesarias)**

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | | Pestañas o tabs Verticales |
| **Introducción** | | Para evaluar el esfuerzo necesario en un proyecto de software, se utiliza la unidad de medida meses/persona (PM). Esta unidad representa el tiempo de trabajo de una persona en el equipo de desarrollo del proyecto. Existen varios métodos para estimar este esfuerzo, algunos de ellos son: |
|  | | |
| **Puntos de Función Ajustados y Coeficientes de Conversión:** | Los Puntos de Función Ajustados, se obtienen como el producto de los Puntos de Función sin ajustar por el Factor de Ajuste:  FP = UFP x AF donde:  UFP son los puntos de función sin ajustar.  AF es el factor de ajuste.  Para nivelar los puntos de función, previamente calculados, hay que estimar cada una de las características del sistema, las cuales se deben tener en cuenta con su factor de peso asociado, el cual indica el grado de importancia de la característica asociada que se está revisando. Luego de calculados los puntos de función ajustados, se pueden ejecutar los coeficientes que permitan estimar el esfuerzo del proyecto, tiempo y costo. | |
| **Estimación del esfuerzo a partir de los Casos de Uso:** | El esfuerzo en horas-persona viene dado por:  E = UCP x CF donde:  E: esfuerzo estimado en horas-persona.  UCP: Puntos de Casos de Uso ajustados.  CF: factor de conversión. | |
| **Modelo Composición de Aplicación:** | Para este modelo, la fórmula que se propone es la siguiente:  PM = NOP / PROD donde:  NOP (Nuevos Puntos Objeto): tamaño del nuevo *software* a desarrollar, expresado en Puntos Objeto, se calcula de la siguiente manera:  NOP = OP x (100 - % reúso)/100, donde:  OP (Puntos Objeto): es el tamaño del *software* a desarrollar, representado en Puntos Objeto.  % reúso: porcentaje de reúso que se espera lograr en el proyecto.  PROD: es la productividad promedio, definida a partir del análisis de datos de proyectos. | |
| **Modelo Diseño Temprano:** | Este modelo se utiliza, como su nombre lo indica, en etapas tempranas del proyecto de *software*, cuando se conoce muy poco del sistema a desarrollar. El mismo, emplea para su cálculo, Multiplicadores de Esfuerzo, los cuales se clasifican en categorías; estas son:   * Del producto:   + RCPX: confiabilidad y complejidad del producto.   + RUSE: reusabilidad requerida. * De la plataforma:   + PDIF: dificultad de la plataforma. * Del personal:   + PERS: aptitud del personal.   + PREX: experiencia del personal. * Del proyecto:   + FCIL: facilidades.   + SCED: cronograma de desarrollo requerido. | |
| **Modelo Post-Arquitectura:** | Este modelo utiliza:   * Puntos Función y/o Líneas de Código Fuente para realizar la estimación del tamaño, con modificadores que contemplan el reúso, con y sin traducción automática, y el "desperdicio" (*breakage*). * Un grupo de 17 atributos, llamados factores de costo, que permiten estimar características del proyecto, referentes al personal, plataforma de desarrollo, etc., que tienen intervención en los costos. * Cinco factores que establecen un exponente, según el modelo del concepto de economía de escala y deseconomía de escala. | |
| **COCOMO II:** | Cocomo II está compuesto por tres de los modelos previamente descritos: Composición de Aplicación, Diseño Temprano y Post-Arquitectura. El modelo Composición de Aplicación se usa en desarrollos de *software* que se encuentran en la etapa de prototipación; el modelo Diseño Temprano se utiliza en las primeras fases del desarrollo, en las cuales se estiman las alternativas de *hardware* y *software* de un proyecto; el modelo Post-Arquitectura se ejecuta en la etapa de desarrollo, después que se describe la arquitectura del sistema, y en la fase de mantenimiento. | |

**~~3.3.~~ Estimación de cronogramas (tiempo)**

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Pestañas o tabs horizontales | |
| **Introducción** | Estimar el cronograma de un proyecto implica determinar cuánto tiempo (semanas, días, horas) será necesario para completar cada actividad del proyecto. Es importante tener en cuenta que el cronograma puede variar a medida que avanza el proyecto. Hay diferentes técnicas para estimar la duración de un proyecto, algunas de ellas son: | |
| **Juicio de**  **expertos**: | **Juicio de expertos**  Utilizar la experiencia de las personas. |  |
| **Estimación por analogía:** | **Estimación por analogía**  Pueden basarse en la estimación hecha con anterioridad, para una actividad parecida (del mismo o de otro proyecto). |  |
| **Simulación:** | **Simulación**  Se basa en medir múltiples estimaciones, partiendo de diferentes grupos de asunciones. El método más general es el Análisis de Monte Carlo, el cual dice que, a partir de una distribución de resultados probables para cada actividad, se cuantifica una distribución de resultados probables para todo el proyecto. |  |
| **Métodos matemáticos:** | **Métodos matemáticos**  Proporcionan estimaciones directas, ejecutando técnicas estadísticas o fórmulas matemáticas (lineales o no lineales), sobre algunos datos empíricos; por ejemplo, Cocomo. |  |

1. **Licenciamiento**

|  |
| --- |
| Cuadro de texto |
| Las licencias de *software* son un acuerdo entre el autor o propietario del *software* y el usuario final que determina las condiciones y términos en los que el usuario puede utilizar el *software*.  Las licencias de *software* establecen las reglas para el uso del *software*, incluyendo cómo se puede distribuir, modificar y redistribuir el *software*. Algunas licencias de *software* son gratuitas y permiten al usuario utilizar el *software* de forma libre y sin restricciones, mientras que otras licencias de *software* son de pago y requieren que el usuario pague una tarifa para utilizar el *software*. Es importante leer y entender las condiciones de la licencia de *software* antes de utilizar cualquier *software*, ya que al hacerlo, se están aceptando los términos y condiciones de la licencia. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Carrusel de tarjetas | |
| **Introducción** | Desde la década de 1980, el número de tipos de licencias de *software* ha estado en constante evolución debido a que las empresas de *software* comenzaron a restringir el uso del *software* de acuerdo a sus intereses. Para entender el mundo de las licencias de *software*, es necesario diferenciar entre dos categorías principales. | |
| **El *software* libre no está protegido por derechos de autor**  No tiene por qué ser gratis, pero el usuario tiene libertad sobre ese programa para modificarlo, usarlo, copiarlo y distribuirlo a su manera. Eso permite a los millones de desarrolladores del mundo, aumentar sus posibilidades y expandir el conocimiento. | |  |
| **El *software* de propietario**  Es aquel que su creador/dueño legal, prohíbe la copia o delimita su redistribución y modificación, sin su autorización o sin un pago previo. | |  |

|  |
| --- |
| Cajón de texto de color |
| Además de las licencias de *software* libre y de propiedad, existen otras opciones de licencias, como el *software* gratuito o de dominio público, que no está protegido por derechos de autor y se puede utilizar libremente. Sin embargo, el autor del *software* puede imponer ciertas restricciones en cuanto a la distribución y uso del *software* y de cualquier trabajo generado a partir del mismo. También existe la licencia de *software* semilibre, que pertenece a un dueño pero se permite la distribución, uso, copia y modificación del *software* por parte de individuos sin fines de lucro. Es importante leer y comprender las condiciones de cada tipo de licencia de *software* antes de utilizar el *software*. |

|  |  |
| --- | --- |
| **Tipo de recurso** | Acordeón tipo 1 |
| **Introducción** | Cuando se toma un programa, sea bajo el tipo de licencia que sea, lo que realmente se adquiere es una licencia de uso, nunca el programa en sí y, como norma principal, se otorga una licencia por punto de uso o dispositivo. Veamos algunos tipos de licencias: |
|  | |
| **Licencias de *software* libre:** permiten a los usuarios utilizar el *software* de acuerdo a sus necesidades, incluyendo ejecutar el programa, estudiar su funcionamiento, modificarlo para adaptarlo a sus necesidades, copiar y redistribuir el *software* y mejorarlo y compartir esas mejoras con la comunidad. Estas licencias ofrecen una gran libertad y flexibilidad para el uso del *software* y son una buena opción para aquellos que deseen colaborar en el desarrollo y mejora del *software*. | |
| **Licencias de *software* libre no protegido con** *copyright***:** son aquellas que permiten la redistribución y modificación del *software*, pero también permiten añadir restricciones adicionales a algunas copias o versiones modificadas del *software*. Esto significa que algunas copias o versiones modificadas del *software* pueden no ser completamente libres y pueden requerir un *software* de pago para poder ejecutar ciertos elementos. Estas licencias pueden ser útiles para aquellos que deseen distribuir *software* libre, pero también deseen restringir ciertas formas de uso o modificación del *software*. | |
| **Licencias de *software* libre protegidas con** *copyright***:** son aquellas que no permiten a quienes modifican o redistribuyen el *software* agregar restricciones adicionales. Esto significa que cualquier copia o modificación del *software* debe ser libremente utilizable y distribuible. El *copyright* es una práctica legal y común que consiste en utilizar los derechos de autor para promover el uso y la distribución libres de una obra y exigir las mismas libertades al distribuir copias y derivados de la obra. Esto se utiliza para garantizar que el *software* libre siga siendo libre a medida que se modifica y se redistribuye. | |
| **Licencia de *software* de GPL (GNU LGPL):** es *software* libre y permite que se enlace con módulos no libres. Debe estar disponible y accesible para copias ilimitadas y a cualquier persona que lo solicite. | |
| **Licencia BSD:** es una licencia permisiva, pues casi no impone condiciones sobre lo que un usuario puede hacer. Es la menos restrictiva para los desarrolladores. El *software* puede ser vendido y no hay responsabilidad de incluir el código fuente. | |
| **Licencias de MPL:** son las de Mozilla Firefox, Mozilla Thunderbird y la mayoría de otros *software* de Mozilla. Son algo menos permisivas que las anteriores. | |
| **Licencia de *software* de Debian: Licencia de *software* de *Debian*:** su nombre completo es *Debian* Free *software Guidelines* (DFSG) y de ella deriva la licencia de *Open Source Initiative*. Decreta en sus criterios de distribución, la exigencia de la redistribución libre (también del trabajo derivado bajo la misma licencia del original), publicación del código fuente y que no puede excluir a grupo de personas o ninguna persona, ni el *software,* ni ninguna forma de su utilización. | |
| **Licencia *Open Source Initiative*:** una de las licencias más conocidas y utilizadas, llamada también de fuentes abiertas o código abierto, que permite la libre distribución (también del código fuente), la variación del mismo, los desarrollos derivados y su redistribución en las mismas condiciones que el *software* original. | |
| **Licencia de *software* de X.org (X *Windows System*):** distribuida por el Consorcio X (es la organización que controla su desarrollo), es usada en los sistemas operativos Linux y UNIX nacidos como una alternativa a la minoría de los sistemas gráficos como Microsoft. Es también *software* libre, aunque sin adherirse al *copyright*. | |
| **Licencia de *software* Freeware:** autoriza el uso del *software* de forma gratuita y libre, pero en general a particulares y no a empresas u organismos oficiales. | |
| **Licencia de *software* de Donationware:** es similar a la licencia Freeware, pero solicita a cambio una donación, aunque no sea obligatorio hacerlo para poder utilizarla. La licencia de *software* de Postcardware es similar, pero, en lugar de una donación, suele solicitar el envío de una postal como confirmación de su utilización. | |
| **Licencias de prueba o Shareware:** todos los usuarios se han encontrado alguna vez con este tipo de licencia. Suelen pedir un registro previo y autoriza el uso de un programa para que el usuario lo evalúe y, posteriormente, lo compre. La licencia tipo Demo es una parte de un programa para su evaluación. Son muy populares en el caso de *software* de juegos, etc. | |
| **Licencias de *software* de propietario:** con las licencias de *software* de código cerrado o *software* propietario o privativo (llamadas CLUFs o para usuario final), los propietarios establecen los derechos de modificación, uso, copia, distribución, redistribución, cesión y cualquier otra consideración necesaria. Los fabricantes de programas sometidos a este tipo de licencias ofrecen servicios de actualizaciones durante el tiempo de vida del producto y, además, soporte técnico. Este tipo de licencias no permiten que el *software* sea cambiado, copiado o distribuido, desensamblado; además, controlan el número de copias que pueden ser instaladas y los fines establecidos para los cuales puede ser utilizado. | |
| **Licencia de *software* de privativo:** su uso, redistribución o modificación está prohibida, o requiere de la solicitud de una autorización. | |
| **Licencia de *software* de comercial:** es aquel que está desarrollado por una entidad que tiene la intención de monetizarlo por su uso. | |
| **Licencia de *software* de OEM:** supedita su venta a que ese *software* debe formar parte de un equipo nuevo (por ejemplo, el caso del sistema operativo Microsoft cuando nos compramos un nuevo ordenador). | |
| **Licencia de *software* de Retail:** es *software* que se vende, el programa es cien por ciento propiedad del usuario nuevo y puede regalarlo o venderlo. | |
| **Licencia de *software* de volumen:** su destino son los grandes usuarios (empresas) y el contrato de adquisición estipula un determinado número de equipos que pueden utilizar el mismo código. | |

|  |
| --- |
| Cuadro de texto |
| Recuerde explorar los demás recursos que se encuentran disponibles en este componente formativo; para ello, diríjase al menú principal, donde encontrará la síntesis, una actividad didáctica para reforzar los conceptos estudiados, material complementario, entre otros. |

**SÍNTESIS**

|  |  |
| --- | --- |
| **Tipo de recurso** | Síntesis |
| Síntesis | |
| **Introducción** | El siguiente mapa integra los criterios y especificidades de los conocimientos expuestos en el presente componente formativo: |
|  | |

**ACTIVIDAD DIDÁCTICA**

|  |  |
| --- | --- |
| **Tipo de recurso** | Actividad didáctica. Verdadero y falso |
| Apreciado aprendiz, a continuación, encontrará una serie de preguntas que deberá resolver, con el objetivo de evaluar la aprehensión de los conocimientos expuestos en este componente formativo. |  |
| 1. **Los contratos de confidencialidad y los contratos de duración pertenecen a los tipos de contratos de *software*** |  |
| **Verdadero** | Falso (correcto) |
| Retroalimentación positiva: los tipos de contrato de *software* se pueden clasificar de la siguiente manera:   * Contratos de licencia de *software* * Contratos de desarrollo de *software* * Contrato de soporte y mantenimiento * Contrato de *Software As A Service* * Contrato de distribución de *software* | Retroalimentación negativa ¡Incorrecto! Recuerde seguir estudiando los diferentes tipos de contratos. |
| **Verdadero** (correcto) | Falso |
| 1. **¿La Técnica de Estimación del tamaño del *software* pertenece a un tipo de Técnicas de estimación de proyecto de *software*?** |  |
| **Verdadero** (correcto) | Falso |
| Retroalimentación positiva:  Las técnicas de estimación de proyecto son:     * Técnicas de estimación del tamaño del *software*. * Técnicas de estimación de esfuerzo * Estimación de cronogramas | Retroalimentación negativa ¡Incorrecto! Recuerde seguir estudiando las técnicas de estimación de proyecto. |
| **3. La estimación de cronogramas (calendario) consiste en estimar el número de períodos de trabajo (1 año, 2 años, 3 años) que serán necesarios para completar cada actividad dentro del proyecto.** |  |
| Retroalimentación positiva:La estimación de cronogramas (calendario) consiste en estimar el número de períodos de trabajo (horas, días, semanas) que serán necesarios para completar cada actividad dentro del proyecto. | Retroalimentación negativa ¡Incorrecto! Recuerde seguir estudiando la estimación de cronogramas. |
| **Verdadero** | Falso (correcto) |
| **4. Las licencias de *software* son una especie de acuerdo contractual que determina las normas para usarlas. Existen diferentes licencias en función de las condiciones para su uso o de su gratuidad.** |  |
| Retroalimentación positiva: Las licencias de *software* se pueden definir como un contrato entre el autor de un programa informático y los usuarios del mismo. En ellas se establecen los términos, condiciones y cláusulas que se deben cumplir para poder usar ese programa y cada usuario que se descarga, instala, copia o lo utiliza debe aceptar esas condiciones. | Retroalimentación negativa: ¡Incorrecto! Recuerde seguir estudiando los temas relacionados con las licencias de *software*. |
| **Verdadero**(correcto) | Falso |
| **5. Es aquel que su creador o dueño legal limita o prohíbe la copia, redistribución y modificación sin su permiso o sin un pago previo. Esta definición corresponde al tema de *software* libre carece de *copyright.*** |  |
| **Verdadero** | Falso (correcto) |
| Retroalimentación positiva: El *software* libre carece de *copyright*: No tiene por qué ser gratuito, pero el usuario tiene libertad sobre ese programa para usarlo, copiarlo, modificarlo y distribuirlo a su gusto. Eso permite a los millones de desarrolladores del mundo ampliar sus posibilidades y expandir su saber y conocimiento. | Retroalimentación negativa: ¡Incorrecto! Recuerde seguir estudiando los temas relacionados con el *software* libre. |
| **6. Para calcular los Puntos de Función hay que tener en cuenta lo siguiente:**     * **Entradas al sistema (entradas externas).** * **Salidas del sistema (salidas externas).** * **Consultas.** * **Grupos de datos lógicos del sistema.** * **Grupos de datos lógicos que no son del sistema, pero que el sistema usa.** |  |
| **Verdadero** (correcto) | Falso |
| (Retroalimentación positiva: El Análisis de Puntos Función (FPA) provee un enfoque alternativo pues permite estimar el tamaño del *software* basándose en los requerimientos, luego la cantidad de SLOC se puede estimar desde el tamaño en Puntos Función (FP). En etapas tempranas del ciclo de vida, se identifican los Actores y los Casos de Uso del sistema, y se documenta cada uno de ellos mediante una breve descripción. Aplicando el Análisis de Puntos de Función a estos Casos de Uso, se podrá obtener una estimación, *grosso* *modo,* del tamaño y, a partir de ella, del esfuerzo. Más adelante, cuando se tenga más información acerca del sistema se podrá refinar este análisis. Posteriormente, se amplía la documentación de cada Caso de Uso, describiendo los escenarios que se producen dentro del mismo. Un Escenario relata la secuencia de pasos que efectúan los actores y el sistema durante la ejecución del Caso de Uso. Si se aplica nuevamente el Análisis de Puntos de Función sobre estos Casos de Uso detallados, la estimación del tamaño y esfuerzo será más precisa que la anterior. El FPA mide el tamaño del *software* cuantificando la funcionalidad provista al usuario basándose solamente en el diseño lógico y las especificaciones funcionales. Este método consiste (brevemente) en calcular los puntos de función teniendo en cuenta:     * Entradas al sistema (entradas externas). * Salidas del sistema (salidas externas). * Consultas. * Grupos de datos lógicos del sistema. * Grupos de datos lógicos que no son del sistema, pero que el sistema usa. | Retroalimentación negativa: ¡Incorrecto! Recuerde seguir estudiando el proceso para calcular los Puntos de Función. |
| **7. La fórmula de Albretch para calcular los puntos función, es la siguiente:**  **FP = UFP x TCF**  **Donde UFP: Puntos Función no Ajustados y TCF: Factor de Complejidad Técnica.**  **Los UFP son calculados mediante la sumatoria de los pesos de todos los parámetros identificados.** |  |
| **Verdadero** (correcto) | Falso |
| Retroalimentación positiva: Este método consiste (brevemente) en calcular los Puntos de Función teniendo en cuenta:     * Entradas al sistema (entradas externas). * Salidas del sistema (salidas externas). * Consultas. * Grupos de datos lógicos del sistema. * Grupos de datos lógicos que no son del sistema, pero que el sistema usa.   Debe computarse cuántas ocurrencias de cada parámetro contiene un sistema, calificándolas según su complejidad en alta, media y baja. Cada parámetro para una complejidad dada tiene un determinado peso, ese peso son los puntos de función asignados a ese parámetro. Luego de realizado este proceso los puntos de función se ajustan a los requerimientos del sistema.  La fórmula de Albretch para calcular los puntos función, es la siguiente:  **FP = UFP x TCF**  Donde UFP: Puntos Función no Ajustados y TCF: Factor de Complejidad Técnica.  Los UFP son calculados mediante la sumatoria de los pesos de todos los parámetros identificados. | Retroalimentación negativa: ¡Incorrecto! Recuerde seguir estudiando el proceso para calcular los puntos de función. |
| **8. En las técnicas de estimación del tamaño del *software*, el método UCP clasifica los casos de uso en simple, promedio y complejo con factores de peso “s”, “m” y “c”, respectivamente.** |  |
| **Verdadero** | Falso (correcto) |
| Retroalimentación positiva: El método UCP clasifica los casos de uso en simple, promedio y complejo con factores de peso 5, 10 y 15, respectivamente. La clasificación se hace con base en el número de transacciones que contiene el caso de uso, 1 a 3 para simple, 4 a 7 para promedio y 8 o más para complejos | Retroalimentación negativa: ¡Incorrecto! Recuerde seguir estudiando las técnicas de estimación del tamaño del *software*. |
| **9. COCOMO II pertenece a las técnicas de estimación de esfuerzo** |  |
| **Verdadero** (correcto) | Falso |
| Retroalimentación positiva:  Existen varios métodos de estimación del esfuerzo como son:     * Puntos de Función Ajustados y Coeficientes de Conversión * Estimación del esfuerzo a partir de los Casos de Uso * Modelo Composición de Aplicación * Modelo Diseño Temprano * Modelo Post-Arquitectura * COCOMO II | Retroalimentación negativa: ¡Incorrecto! Recuerde seguir estudiando las técnicas de estimación de esfuerzo. |
| **10. En la estimación de cronogramas existen varias técnicas para la estimación de la duración de un proyecto como son:**   * **Juicio de expertos** * **Estimación por analogía** * **Simulación** * **Métodos matemáticos** |  |
| **Verdadero** (correcto) | Falso |
| Retroalimentación positiva: La estimación de cronogramas (calendario) consiste en estimar el número de períodos de trabajo (horas, días, semanas) que serán necesarios para completar cada actividad dentro del proyecto. Existen varias técnicas para la estimación de la duración de un proyecto:     * Juicio de expertos: utilizar la experiencia histórica de las personas. * Estimación por analogía: basarse en la estimación realizada con anterioridad para una actividad similar (del mismo o de otro proyecto). Es un caso especial de juicio de expertos. * Simulación: se basa en calcular múltiples estimaciones partiendo de diferentes conjuntos de asunciones. El método más común es el Análisis de Monte Carlo: a partir de una distribución de resultados probables para cada actividad, se calcula una distribución de resultados probables para todo el proyecto. * Métodos matemáticos: proporcionan estimaciones directas aplicando técnicas estadísticas o fórmulas matemáticas (lineales o no lineales) sobre ciertos datos empíricos; por ejemplo, COCOMO. | Retroalimentación negativa: ¡Incorrecto! Recuerde seguir estudiando las técnicas para la estimación de la duración de un proyecto. |

**MATERIAL COMPLEMENTARIO**

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo de recurso | Material complementario | | |
| Tema | Referencia APA del material | tipo | Enlace |
| **Técnicas de estimación de proyectos de *software*** | [Iacono](https://www.youtube.com/@MatiasIacono), M. (2021). *Técnicas de estimación de proyectos de software* [video].YouTube. | Video | https://www.youtube.com/watch?v=47DKIbfcVlY |
| **Contrato de *software*** | Código ISC. (2020). *Partes del contrato de software o contrato informático* [video]. YouTube. | Video | https://www.youtube.com/watch?v=SKWQSGjufg4 |
| **Licenciamiento** | OpenWebinars. (2019). *Licencias de Software (y sus tipos de licencia)* [video]. YouTube. | Video | https://www.youtube.com/watch?v=Sz07vfAlqYo |
| **Técnicas de estimación de proyectos de *software*** | Infortech. (2021). *Presupuesto y Estimación de Proyecto de software* [video]. YouTube. | Video | https://youtu.be/Scz9te0Ylfo |

**GLOSARIO**

|  |  |
| --- | --- |
| **Tipo de recurso** | Glosario |
| **Administración de proyectos:** | es una estrategia usada por empresas e instituciones para alcanzar objetivos en un tiempo determinado. También puede usarse para objetivos personales. |
| **Contrato:** | es un instrumento jurídico que regula la ejecución de una obra, prestación de un servicio o suministro de bienes, incluidas las órdenes de compras y órdenes de servicio, que contendrían, al menos, las siguientes condiciones: precio, cantidades, forma de pago, tiempo y forma de entrega y especificaciones contenidas en el pliego de condiciones, si fuere necesario. |
| ***Copyright:*** | es la forma de atribuir la autoría a alguien que ha creado una obra, además de proporcionarle una serie de derechos como autor. |
| **Estimación:** | es un cálculo que se realiza a partir de la evaluación estadística. Dicho estudio suele efectuarse sobre una muestra y no sobre toda la población objetivo. |
| **Gestión de costos del proyecto:** | proceso relacionado con planificación, estimación, determinación y control de los costos que busca completar la ejecución del proyecto dentro del presupuesto aprobado. |
| **Inflación:** | en economía, es el crecimiento generalizado y sostenido de los precios de bienes y servicios durante un periodo de tiempo determinado. |
| **Proyecto:** | es una planificación que consiste en un conjunto de actividades que se encuentran interrelacionadas y coordinadas entre sí y que cumplen los objetivos de calidad esperados. |
| **Licenciamiento:** | son un tipo de acuerdo contractual que establece las normas para el uso de un producto. Existen varias licencias en función de las condiciones para su uso o de su gratuidad. |
| ***Software*** **libre:** | es aquel que les da a sus usuarios la libertad de ejecutar, copiar, estudiar, modificar y distribuir el *software*. |
| **Viabilidad:** | es la expectativa en función del crecimiento económico, el bienestar social y la conservación del ambiente producto del logro exitoso de los objetivos del proyecto. |

**REFERENCIAS BIBLIOGRÁFICAS**

|  |  |
| --- | --- |
| **Tipo de recurso** | Bibliografía |
| Colombia Compra Eficiente. (2020). *Guía de buenas prácticas en la adquisición de Software y servicios asociados*. Departamento Nacional de Planeación. https://www.colombiacompra.gov.co/sites/cce\_public/files/cce\_documents/guia\_de\_buenas\_practicas\_software\_ver1\_logo01.pdf | |
| ~~Iteckne vol.17 no.2 Bucaramanga July/Dec. 2020  Epub Mar 01, 2021~~ | |
| Méndez, E. R. (2018). *Estimación de esfuerzo en proyectos de desarrollo de software con metodologías ágiles.* [Tesis de máster en Gestión y Dirección de Proyectos, Universitat Politècnica de València]. | |
| Pérez, I., López, P., Varona, E., Piñero, P. y García, R. (2018): Construcción de resúmenes lingüísticos a partir de rasgos de la personalidad y el desempeño en el desarrollo de *software*. *Revista Cubana de Ciencias Informáticas*, *12*, p. 135-150. | |
| Rodríguez, J. R. (2005). *Gestión de Proyectos Informáticos: Métodos, herramientas y casos.* Editorial UOC. | |

