**Datos de identificación del programa de formación**

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

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | 220501099 – Probar la solución del *software* de acuerdo con parámetros técnicos y modelos de referencia. | RESULTADOS DE APRENDIZAJE | 220501099-03 – Evaluar las pruebas de *software* de acuerdo a los requerimientos técnicos y modelos de referencia. |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | 20 |
| NOMBRE DEL COMPONENTE FORMATIVO | Evaluación de pruebas. |
| BREVE DESCRIPCIÓN | Se abordarán tanto los temas relacionados con pruebas, sus criterios de aceptación y finalización, como algunos conceptos claves, por ejemplo, tipos y clasificación. Dada su importancia en el desarrollo de *software*, el componente permitirá un entendimiento global de cómo poder manejar la temática en un proyecto de *software*. Igualmente, se enfatiza en la necesidad de que, antes de lanzar un producto, se deben cumplir estándares de calidad que garanticen un correcto funcionamiento. |
| PALABRAS CLAVE | Hallazgo, artefacto, requerimiento, modelos. |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | 6 - VENTAS Y SERVICIOS |
| IDIOMA | Español |

# **TABLA DE CONTENIDOS**

# **1. Alcance de pruebas 2. Evaluación de programas mediante pruebas**

**INTRODUCCIÓN**

|  |
| --- |
| Cuadro de texto |
| Aprendiz, bienvenido a este componente formativo, a continuación, aprenderá sobre las pruebas, tanto sus criterios de aceptación y finalización, como algunos conceptos claves sobre tipos y clasificación. Esto le permitirá tener un entendimiento global sobre cómo manejar este tema en un 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 motion | | | |
| **NOTA** |  | | | |
| **Título** | Introducción | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración** | **Texto** |
| **1** | Pruebas de rendimiento: Pruebas de referencia y de referencia - LoadView |  | A continuación conocerá, de forma general, la temática que se estudiará en este componente formativo. Aprenderá sobre las pruebas, tanto sus criterios de aceptación y finalización, como algunos conceptos claves, que le permitirán tener un entendimiento global sobre cómo manejar esta temática en un proyecto de *software*. | Material  Pruebas  Criterios  *Software* |
| **2** | Benchmark testing abstract concept vector illustration Stock Vector Image &  Art - Alamy |  | También, se brindarán los conocimientos fundamentales sobre los diferentes conceptos de pruebas, cuál es su importancia, hasta dónde se debe llegar con una prueba, qué se debe probar y cuáles son sus alcances y objetivos al momento de plantearse en un proyecto. Este tema es de gran relevancia en el desarrollo de *software,* ya que, antes de lanzar un producto, se deben cumplir estándares de calidad que garanticen un correcto funcionamiento. | Conocimientos  Objetivos  Proyecto  *Software* |
| **3** | Los 8 beneficios de las pruebas de Rendimiento - El mínimo viable |  | En la industria del *software* las pruebas son un proceso fundamental y, por eso, existen diferentes herramientas para aplicarlas. Por ejemplo, en Java, se emplea la herramienta *SonarQube*, que permite detectar *bugs*, falencias, vacíos de seguridad, redundancias y muchos otros defectos dentro de un código escrito. Igualmente, en la industria, existe el área de aseguramiento de la calidad y pruebas, la cual es la encargada de montar las pruebas y garantizar una calidad mínima en todos los productos. | Pruebas  Proceso  Java  Industria  Productos |
| **4** | Adolescente harto cascada satisfacción Sucio formar tipos de testing -  angelsde-signs.com  Qué implica un buen proceso de QA testing - Syntonize |  | Así mismo, la industria provee a las personas encargadas de la construcción de pruebas, una amplia variedad de librerías creadas por diferentes proyectos desarrollados para empresas, que pueden utilizar. Pero antes de ejecutar una prueba, es necesario tener claro su alcance y cuándo usarla.  En un proceso de construcción de pruebas, aparte de identificar los objetos o módulos a los que testear, es importante conocer los tipos de pruebas, cómo construir un plan, cómo se declaran los objetivos, el alcance y la finalidad de la prueba, como también, entender qué es un criterio de finalización, cómo se crean las pruebas y cómo se declaran los escenarios.  En este componente, por medio de un ejemplo, se detallan los modelos de pruebas, su definición, sus características y cómo se podría usar en un proyecto. | Construcción de pruebas  Librería  Empresa  Módulo  Escenario  Proyecto |
| **5** | Home of Crowdtesting: UX & Usability Testing | QA & Functional Testing |  | El material que conforma este componente, tiene un carácter teórico-práctico y, por este motivo, se entiende que llevará una secuencia de teorías importantes para la conceptualización de los conceptos sobre la construcción de pruebas de *software*, sus elementos, sus objetos y sus diferentes cualidades. Posteriormente, encontrará dos ejemplos prácticos.  Lo anterior es necesario, ya que, al tener claros los conceptos, se irán aplicando fácilmente, de forma práctica, para construir las pruebas. | Material  Teoría  *Software*  Cualidad  Pruebas |
| **6** | How Much Time Does it Take to Test an Application? |  | Teniendo en cuenta lo anterior, se profundizará en las diferentes cualidades para la construcción de pruebas (de acuerdo a los requerimientos establecidos), las características y las formas específicas de aplicarlas.  Así mismo, se busca desarrollar conocimientos claros sobre su objetivo y cuáles son las principales particularidades que impactan la forma en la cual, el usuario, experimenta la interacción con la aplicación web desarrollada. | Construcción de pruebas  Requerimientos  Objetivo  Usuario |
| **7** | testing monitoring |  | Una vez culminada la exploración temática, consultado el material complementario y realizadas las actividades evaluativas, contará con herramientas valiosas para el ejercicio profesional. Muchos éxitos en su aprendizaje. | Temática  Material Actividad |
| **Nombre del archivo** | 228125\_v1 | | |  |

**DESARROLLO DE CONTENIDO**

1. **Alcance de pruebas**

|  |
| --- |
| Cuadro de texto |
| Primero, es necesario definir qué es una prueba de *software* y cuál es su alcance. Observe el siguiente video para adentrarse en estas preguntas. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tipo de recurso** | Video motion | | | |
| **NOTA** |  | | | |
| **Título** | **Alcance de pruebas de *software*** | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración** | **Texto** |
| **1** | Proceso de Testing - Tareas de Análisis y Diseño - YouTube |  | La prueba de *software* se refiere al proceso de evaluar y verificar que un producto, o aplicación de *software,* realiza lo que se supone que debe realizar. Los beneficios de las pruebas incluyen la prevención de errores, la reducción de los costos de desarrollo y el mejoramiento del rendimiento. Su propósito principal es asegurar que la aplicación desarrollada cumpla con altos estándares, es decir, que llegue al cliente como un producto de calidad.  Los encargados de probar los *softwares* son conocidos como *testers*, y planifican y realizan pruebas de *software* para comprobar si todo funciona correctamente. | *Software*  Proceso  Aplicación  Costos  Planificar |
| **2** | No hay ninguna descripción de la foto disponible. |  | Para resumir, una prueba de *software* es un proceso en el cual se revisa un sistema que se quiere probar, siguiendo unas condiciones definidas y aplicando una serie de estímulos que, previamente, se crearon (casos de prueba) de forma sistemática, con diferentes metodologías, técnicas y herramientas, para detectar el nivel de calidad del producto final. | Proceso  Sistema  Metodología  Técnica  Herramientas |
| **3** | Top 5 Software Testing Tools - EduinPro |  | Este proceso debe ser estricto, riguroso y llevarse a cabo de manera ordenada. Igualmente, se debe respaldar en métricas y modelos estandarizados y bien definidos.  Todas estas tareas de pruebas ejecutadas, serán documentadas, especialmente, si llegaran a fallar o a encontrar un error.  Para tener un ejemplo, si se hace referencia a una prueba de caja negra, se tendrá una secuencia de acciones que se llevan a cabo para probar el sistema. | Proceso  Métricas  Modelos  Secuencia |
| **4** | Software Testing Services – Software QA Company |  | Este proceso conlleva las siguientes tareas:   * Establecer el alcance, entregables y criterios de éxito. * Estimar el esfuerzo de la prueba. * Planear el proyecto. * Diseñar el caso de prueba. * Aplicar el caso de prueba. * Reportar métricas y hacer seguimiento. * Reportar análisis de resultados. | Proceso  Proyecto  Prueba  Métricas |
| **5** |  |  | Con referencia a los casos de prueba, es necesario que cumplan los siguientes atributos:   * Que no incluya segmentos de otros casos (ortogonalidad). * Que detecte las fallas (efectividad). * Que con poco se pruebe mucho (eficiencia). * Que muestre las fallas de una forma clara, sin ambigüedad, de acuerdo con los requerimientos funcionales y no funcionales. | Atributos  Efectividad  Eficiencia  Requerimiento |
| **6** | Pruebas de software desarrollo de software proceso codificación an - vector  de stock 2717386 | Crushpixel |  | Posteriormente, para revisar el alcance de las pruebas, es importante indicar que las pruebas de *software* tienen limitaciones, tanto teóricas como prácticas.  En cuanto a lo teórico, la prueba se conoce como un problema designado, no decidible, es decir, que el programa que se crea no es autogestionable, por lo cual, no se puede probar a sí mismo y siempre debe contar con intervención de un humano.  Desde la perspectiva de lo práctico, la cantidad de posibilidades con las que se puede probar minuciosamente un sistema, son variadas e incalculables. Por esta razón, se hace necesario revisar y seleccionar las técnicas adecuadas para optimizar y encontrar la mayor cantidad posible de fallas en los recursos explorados. | Pruebas  *Software*  Teórico  Programa  Sistema |
| **7** | Concepto De Programación, De Software De Desarrollo Y De Codificación  Ilustración del Vector - Ilustración de editable, informe: 109371534 |  | La prueba de *software* implica la aplicación de técnicas y herramientas apropiadas en el marco de un proceso bien definido. Se debe tener claridad en cuanto a la presión para cumplir tiempos, la inexperiencia, la complejidad y el buen juicio, para no cometer errores en la construcción del *software*. Si esos errores no se detectan, el producto llegará al cliente defectuoso, sin satisfacer sus necesidades. | Técnicas  Herramientas  Producto  Cliente |
| **Nombre del archivo** | 228125\_v2 | | |  |

|  |
| --- |
| Cuadro de texto |
| En una prueba de *software* se buscan varios objetivos. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Carrusel de tarjetas | |
| **Introducción** | A continuación, se presenta cada uno de los objetivos. | |
| Conocer la calidad de los productos, no solo de los productos finales, también, de los productos intermedios que se van obteniendo en cada *Sprint,* o avance, según la metodología de desarrollo empleada. | | calidad icono gratis  **Imagen:** 225125\_i1 |
| Evitar el pago de un producto que no cumpla con criterios de calidad mínimos. Esto debe estar pactado desde el inicio del proyecto en los acuerdos de nivel de servicio (*SLA*) registrados. | | método de pago icono gratis  **Imagen:** 225125\_i2 |
| Disminuir el evento de soporte excesivo, es decir, que al entregarse el producto, este, constantemente, de soporte a los usuarios por fallas o malos funcionamientos. Las pruebas, precisamente, sirven para minimizar los riesgos y los eventos de falla cuando se pasa a producción. | | falla icono gratis  **Imagen:** 225125\_i3 |
| Tener un registro completo y bien documentado de las fallas, lo que permite un continuo mejoramiento en el proceso, además de conocer, de antemano, cómo solucionar el problema presentado. | | Registro - Iconos gratis de archivos y carpetas  **Imagen:** 225125\_i4 |

|  |
| --- |
| Cuadro de texto |
| Al hablar de condiciones y criterios de finalización de las pruebas, se refiere a aquellos tópicos, o requerimientos establecidos en las pruebas, que, al cumplirse, ayuden a tomar la decisión de finalizar las pruebas. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | | Pestañas o tabs Verticales |
| **Introducción** | | Es importante tener presente los siguientes conceptos. |
| Pequeñas personas probando la garantía de calidad en software aislado de ilustración de vector plano. Errores de corrección de caracteres de dibujos animados en el dispositivo de hardware. Prueba de aplicaciones y concepto de servicio de TI  **Imagen:** 225125\_i5 | | |
| **Fallas** | Varias falencias o imprevistos pueden detenerse al encontrar todas las fallas. Así, es posible conocer cuáles serán todas las posibles fallas o cuándo se encuentran.  No obstante, la mayor parte del *software* es complejo y tiene un alcance de prueba enorme. No es imposible encontrar todos los defectos en el *software*, pero llevará una eternidad. Incluso, después de encontrar muchos errores en el *software*, nadie puede garantizar que esté libre de defectos, es decir, no existe una situación en la que se pueda afirmar que se han completado las pruebas, que se han encontrado todos los defectos en el *software*, o que no tenga más errores. | |
| **Propósito de las pruebas** | El propósito de las pruebas no es encontrar todos y cada uno de los defectos del *software*, su intención es demostrar que el programa funciona según lo previsto, bien al romperlo, o al encontrar una desviación entre su comportamiento actual y el comportamiento esperado.  Hay defectos ilimitados en el *software* y, por tanto, no es práctico probarlo hasta que se encuentren todos los defectos, aunque nunca se puede saber cuál es el último. La verdad es que no se puede depender de encontrar todos los defectos en el *software* para concluir las pruebas. | |
| **Ciclos de pruebas** | Los ciclos de pruebas continuarán hasta que se tome una decisión sobre cuándo y dónde detenerse. Sin embargo, es complicado tomar la decisión de dejar de realizar pruebas, o de definir si detenerse cuando se encuentran todos los defectos es el criterio para pararlas pruebas, o sobre qué otra base se debe definir. El enfoque más común para detenerse es cuando se agota el tiempo o el presupuesto, o cuando se ejecutan todos los escenarios de prueba.  Así, se deben tener criterios de salida para saber cuándo detener las pruebas, pero ¿cuáles serán los factores importantes que se deben tener en cuenta para concluir las pruebas? Esta decisión se puede tomar, principalmente, por tiempo, presupuesto o porque ya se cumplió el alcance de las pruebas. | |

1. **Evaluación de programas mediante pruebas**

|  |
| --- |
| Cuadro de texto |
| Antes de continuar, es importante recordar qué es una prueba de *software*.   * Es una actividad realizada para evaluar la calidad del producto y mejorarla, identificando sus defectos y problemas. * Es la verificación dinámica del comportamiento de un programa contra el comportamiento esperado, empleando un conjunto finito de casos de prueba que han sido seleccionados de manera adecuada. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Tarjetas Avatar | |
| **Introducción** | Ahora, según la norma ISO/IEC/IEEE 24765:2010, se debe tener en cuenta lo siguiente: | |
| **Verificación**  Proceso de evaluación de un sistema o componente para determinar si el producto, de una determinada fase de desarrollo, satisface las condiciones impuestas al inicio de la fase. | | lista de verificación icono gratis  **Imagen** 225125\_i6 |
| **Validación**  Proceso de evaluación de un sistema o componente, durante o al final del proceso de desarrollo, para determinar cuándo se satisfacen los requerimientos especificados. | | **Imagen:** 225125\_i7 |

|  |
| --- |
| Cuadro de texto |
| Siguiendo las anteriores definiciones, reflexione sobre la siguiente pregunta: ¿Qué es un proceso de pruebas y para qué implementarlo? |

|  |
| --- |
| Cuadro de texto |
| Muchas empresas de desarrollo de *software* tienen una tendencia a pensar que el proceso de las pruebas de *software* se debe realizar en la última etapa para consolidar la calidad de su producto. Esa tendencia debe dejarse atrás, las pruebas tienen que estar alineadas al proceso de desarrollo y ser implementadas en diferentes fases. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Carrusel de tarjetas | |
| **Introducción** | Resulta clave recordar la importancia de realizar un proceso de pruebas que incluya: | |
| Revisión de los requerimientos. | | Icono de Requerimiento de cliente. Icono de desarrollo web simple monocromo para plantillas, diseño web e infografías  **Imagen:** 225125\_i8 |
| Realización de análisis documentales. | | Icono de línea vectorial de investigación de texto. Análisis de la información y examen cuidadoso de los errores documentales. Hoja de papel con lupa estudiando texto. Verificando la legalidad del contrato comercial.  **Imagen:** 225125\_i9 |
| Identificación de defectos. | | los servicios de análisis de fallos y defectos están diseñados para investigar a fondo las complejas formas en que se rompen los dispositivos médicos  **Imagen:** 225125\_i10 |
| Pruebas funcionales y no funcionales. | | metodología de proceso de desarrollo de software de prueba funcional  **Imagen:** 225125\_i11 |
| Pruebas dinámicas y estáticas. | | hackeo de crecimiento maneras en que la empresa de tecnología empresarial se propone mejorar el número de usuarios e ingresos  **Imagen:** 225125\_i12 |
| Pruebas de integración. | | integración prueba arquitectura software programación programación programación programación programación revisión de aplicaciones  **Imagen:** 225125\_i13 |
| Informes de confianza en el nivel de calidad. | | **Imagen:** 225125\_i14 |
| Información para la toma de decisiones. | | Icono de concepto turquesa de toma de decisiones informada. Planificar una estrategia de oportunidad. Solución de problemas. Ilustración de línea delgada de la idea de análisis de datos. Dibujo de color RGB con contorno aislado del vector. Trazo editable  **Imagen:** 225125\_i15 |
| Planes de mejora continua. | | Ciclo automático electrónico ágil. Desarrollo y mejora. Proceso de integración rápido. Icono lineal vectorial aislado en fondo blanco.  **Imagen:** 225125\_i16 |

|  |
| --- |
| Cuadro de texto |
| Scrum concepto de negocio, Agile ilustración con equipo de negocios. Ilustración vectorial para banner web, infografía, móvil.  Las actividades de un proceso de pruebas deben cumplir con las siguientes características: planificar y controlar, seleccionar las condiciones de la prueba, diseñar y ejecutar casos de prueba, comprobar los resultados, evaluar los criterios de resultados, elaborar informes del proceso, y de la aplicación objeto de las pruebas, incluyendo bitácoras de experiencia.  Es importante que, en el proceso, se elabore un plan y una matriz de pruebas con las cuales, al momento de ejecutarlas, se pueda dictaminar si el caso funciona adecuadamente, estableciéndolo, así, una ‘conformidad’. En el evento que un caso de prueba, al ejecutar el producto *software,* no funcione de forma adecuada, se relacionará como una ‘no conformidad’.  A estas métricas se le define un nivel, o puntuación, según la clasificación, las técnicas y la matriz utilizada. Las métricas ayudan a estimar los tiempos, esfuerzos y las asignaciones, además, determinan, también, los niveles de riesgo, lo que permite que el equipo de desarrollo ajuste los flujos de actividades y optimice los recursos y el talento de los equipos de desarrollo.  Siguiendo todo lo anterior, la implementación de un proceso de pruebas brinda las pautas tanto para definir objetivos, analizar y viabilizar los requerimientos, como para diseñar, detallar, programar, implementar y asegurar la calidad de un producto de desarrollo *software*. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Infografía interactiva Punto caliente | |
| **Texto introductorio** | En los últimos años, se han propuesto una serie de principios que permiten establecer unas pautas comunes para que las empresas de desarrollo de *software* conozcan y se adapten a sus procesos de pruebas. Estos principios son: | |
| Los textos editables y enlaces se encuentran en la carpeta Anexos con el nombre Infografia1.pptx | | |
| **Código de la imagen** | 225125\_i18 | |
| **Punto caliente 1** | Las pruebas son herramientas que permiten identificar la presencia de defectos, sin embargo, no garantizan que no se encuentren defectos ocultos en el *software,* y, el hecho que no se identifiquen, no es evidencia de que el *software* esté totalmente correcto. | Principio 1  Las pruebas demuestran la presencia de defectos |
| **Punto caliente 2** | Probar todo el aplicativo de extremo a extremo, con todas las entradas de datos y condiciones, es algo imposible. Así, en lugar de tratar de conseguir ese tipo de pruebas, se debe realizar un análisis de los riesgos para establecer prioridades y tomar decisiones adecuadas en la utilización del talento humano y de los recursos, centralizando los esfuerzos en las pruebas. | Principio 2  Las pruebas exhaustivas no existen |
| **Punto caliente 3** | Identificar los defectos en etapas tempranas ahorrará a la empresa todo tipo de recursos. Cuanto más rápido se identifiquen los defectos, más aumentará la eficiencia. | Principio 3  Pruebas tempranas |
| **Punto caliente 4** | Las pruebas deben concentrarse de manera proporcional. Por lo general, la mayoría de los fallos operativos, se concentran en un número reducido de módulos. | Principio 4  Agrupación de defectos |
| **Punto caliente 5** | Si se repite la misma prueba una y otra vez, la misma serie de casos de prueba dejará de encontrar nuevos defectos. Es importante tanto revisar los casos de prueba periódicamente como escribir nuevos casos de prueba con el objetivo de encontrar más defectos. | Principio 5  Paradoja del pesticida |
| **Punto caliente 6** | Las pruebas dependerán del contexto en el cual se ejecuten, por ejemplo, aquella prueba creada para un sistema crítico de seguridad, como el financiero, requerirá un mayor número de pruebas en comparación con otras de un nivel de complejidad menor, o menos críticos. | Principio 6  Las pruebas dependen del contexto |
| **Punto caliente 7** | La detección y corrección de los defectos no sirven de nada si el sistema no cumple con los requerimientos o con las necesidades del usuario. Estos principios, nombrados en diferentes textos y artículos, son una guía para tener en cuenta todo el proceso de pruebas de calidad de *software*. El equipo de trabajo debe procurar que se cumplan y debe tenerlos presentes en cada actuación y ejecución de herramientas, o técnicas, para garantizar la calidad de un producto *software*. | Principio 7  Falacia de ausencia de errores |

|  |
| --- |
| Cuadro de texto |
| Por otra parte, existe un modelo V para la ejecución de pruebas de *software*, el cual muestra la serie de pruebas que se pueden desarrollar. |

|  |  |
| --- | --- |
| **Tipo de recurso** | Acordeón tipo 1 |
| **Introducción** | Entre esas pruebas se encuentran: |
| Vector illustration of a Software engineer gathering requirements from the business users over phone and developing an integration system by connecting database, application and web servers  **Imagen:** 225125\_i19 | |
| **Pruebas de componentes**  Tienen por objeto localizar defectos y comprobar el funcionamiento de módulos *software*, programas, objetos, clases, etc., que puedan probarse por separado, es decir, de aquellos elementos a los que pueda realizarse una comprobación independiente al resto del sistema en función del contexto. El diseño de casos de pruebas es requisito esencial de los componentes, del diseño de detalle en los casos de uso y del código del módulo o del componente. | |
| **Pruebas de integración**  Se encargan de probar las interfaces entre los componentes o módulos, por ejemplo, el componente de validación de usuario con el sistema operativo, el sistema de archivos en integración con el *hardware*, etc. En el diseño de casos de prueba, es el diseño de *software*, arquitectura, flujos de trabajo, casos de uso y se deben tener en cuenta los objetos de prueba típicos:  1. Base de datos de subsistemas.  2. Infraestructura.  3. Interfaces.  4. Configuración del sistema.  5. Datos de configuración. | |
| **Pruebas de sistema**  Hacen referencia al sistema como un todo, elaborando un plan de pruebas de forma clara y bien estructurada. En el diseño de casos de prueba son los requisitos del usuario, requisitos del sistema, casos de uso, procesos de negocio, informes de análisis de riesgo, y se deben tener en cuenta los objetos de prueba típicos:  1. Procesos de negocio en sistema completamente integrado.  2. Procesos operativos y de mantenimiento.  3. Procedimientos de usuario.  4. Formularios.  5. Informes.  6. Datos de configuración. | |
| **Pruebas de aceptación**  Se enfocan en la aceptación de los criterios previstos en un contrato de desarrollo de *software*, acordado entre la empresa de *software* y el cliente. | |
| **Pruebas de *backup*/restauración**  Se incluye la recuperación de desastres, la gestión de usuarios, la tarea de mantenimiento, la carga de datos, las tareas de migración, las comprobaciones periódicas de vulnerabilidades de seguridad, las pruebas de aceptación contractual y normativa y las pruebas alfa y beta. Es importante, para aplicar el modelo V, tener claro, por parte de los probadores o *tester,* las definiciones de equivocación, defecto o falta y falla.  Siguiendo la norma ISO/ IEC/ IEEE 24765: 2010, estos conceptos se definen así:   * Equivocación (*mistake*): acción del ser humano que produce un resultado incorrecto. * Defecto o falta (*fault*): paso, proceso o definición de dato incorrecto en un programa de computadora. El resultado de una equivocación origina, potencialmente, una falla. * Falla (*failure*): resultado incorrecto. | |

|  |
| --- |
| Cuadro de texto |
| Las pruebas tienen una clasificación. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Infografía interactiva Punto caliente | |
| **Texto introductorio** | A continuación, se presenta cada una de las clases. | |
| Los textos editables y enlaces se encuentran en la carpeta Anexos con el nombre Infografia2.pptx | | |
| **Código de la imagen** | 225125\_i20 | |
| **Punto caliente 1** | Pueden llevarse a cabo en todos los niveles de prueba, se basan en funciones, prestaciones e interoperabilidad con sistemas específicos. Es importante mencionar que se orientan, en el comportamiento externo de un producto o aplicativo *software*, a las pruebas de caja negra. | Pruebas funcionales |
| **Punto caliente 2** | Se refiere a las pruebas necesarias para medir las características de los sistemas y *softwares* que puedan cuantificarse según una escala variable. Se debe tener en cuenta que se orientan hacia el comportamiento externo del *software* y, en la mayoría de los casos, utilizan técnicas de diseño de pruebas de caja negra. | Pruebas no funcionales |
| **Punto caliente 3** | Pueden realizarse en todos los niveles de prueba. Son las más idóneas, después de las técnicas basadas en la especificación, para medir la exhaustividad de las pruebas a través de la evaluación de la cobertura de un tipo de estructura.  Para todo proceso de pruebas, se debe tener clara la diferencia al clasificar los tipos de pruebas, esto contribuye a un análisis sólido del plan de pruebas y a estructurar los casos de pruebas y la creación de la respectiva matriz. Así mismo, se tributa, además, a la eficiencia en el proceso de calidad del producto *software*. | Pruebas estructurales |

**Escenario de prueba**

|  |
| --- |
| Cuadro de texto |
| Ahora, para entender cómo se realiza una evaluación de un *software* o un programa mediante pruebas, se va a plantear, a continuación, un escenario de pruebas hipotético.  Aprendiz, es la única persona encargada de hacer las pruebas. Se va a probar un módulo de *software* y se tiene asignado un presupuesto para cumplir con dicho proceso. Los plazos que se tienen para el periodo de pruebas del proyecto son cuatro semanas, un mes, y los escenarios totales de prueba son 200. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Pestañas o tabs horizontales | |
| **Introducción** | **Escenario 1:** | |
| **Semana 1** | En el desarrollo se detecta la falencia *showstopper*/*severity* 1 el día 1. Las pruebas se detienen por 3 días. De esta manera, ninguna otra prueba se puede ejecutar porque se encontró un error de gravedad 1, que debe solucionarse antes de continuar.  Después de perder 3 días, el bloqueador se resuelve y continúa con su ejecución. Al terminar la semana, se completan 20 escenarios de pruebas con hitos importantes encontrados en los defectos con alta prioridad. | Ilustración del icono vectorial del tramo de tiempo / 1 semana, semanal  Colocar resaltada la primera semana del calendario, no la segunda como está la imagen.  Imagen: 225125\_i21 |
| **Semana 2** | Continúa la prueba del módulo de *software* con más énfasis en el hallazgo de defectos. Se encuentran algunos defectos más de ‘Severidad 1’, ‘Severidad 2’ y ‘Severidad 3’ durante la segunda semana. Al final de la semana, se pueden cubrir 70 escenarios. | Ilustración del icono vectorial del tramo de tiempo / 1 semana, semanal  Imagen: 225125\_i22 |
| **Semana 3** | Al comienzo de la tercera semana, todos los defectos de alta prioridad ya se han resuelto, por tanto, aparte de los escenarios pendientes, se deben probar todos los ajustes realizados que se han podido modificar gracias a las pruebas.  Continuando con un buen progreso, se cubren 120 escenarios con defectos adicionales. En este momento, todos los defectos de alta prioridad ya se encuentran y se notifican, por lo que, ahora, solo le quedan defectos de ‘Severidad 3’ por encontrar. | Ilustración del icono vectorial del tramo de tiempo / 1 semana, semanal  Colocar resaltada la tercera semana del calendario, no la segunda como está la imagen.  Imagen: 225125\_i23 |
| **Semana 4** | En esta semana, es necesario volver a probar la mayoría de los defectos abiertos y los 80 escenarios restantes. Con esto, al final de la semana 4, se pueden completar hasta 180 escenarios con todos los defectos de prioridad alta y media, corregidos y vueltos a probar. | Ilustración del icono vectorial del tramo de tiempo / 1 semana, semanal  Colocar resaltada la cuarta semana del calendario, no la segunda como está la imagen.  Imagen: 225125\_i24 |

|  |
| --- |
| Cuadro de texto |
| La siguiente tabla presenta las actividades de prueba realizadas y el resultado final por semana del escenario 1. |

**Tabla 1**

*Actividades de prueba y resultado final escenario 1*

|  |  |  |
| --- | --- | --- |
| **Semanas** | **Actividades de prueba realizadas** | **Resultado al final de la semana** |
| Semana 1 | • Día 1 - Mostrar defecto de tapón encontrado. • La prueba está bloqueada debido a un defecto de gravedad 1 encontrado el día 1. • Defecto del bloqueador resuelto el día 4. • La ejecución de la prueba continúa hasta el final de la semana 1. | • Defectos críticos de alta prioridad abiertos. • 20 escenarios completados. |
| Semana 2 | • Más enfoque en la búsqueda de defectos. • Ejecución de los escenarios de prueba restantes. • Nueva prueba de defectos reparados. | • Se abrieron defectos más de ‘Severidad 1’, ‘Severidad 2’ y ‘Severidad 3’. • Cobertura total de 70 escenarios cubiertos. |
| Semana 3 | • Nueva prueba de todos los defectos de alta prioridad. • Ejecución de los escenarios de prueba restantes. • Solo quedan por encontrar defectos de ‘Severidad 3’. | • Se abrieron pocos defectos más de ‘Severidad 1’, ‘Severidad 2’ y ‘Severidad 3’. • Cobertura total 120 Escenarios cubiertos. |
| Semana 4 | • Nueva prueba de todos los defectos de prioridad alta y media. • Ejecución de los escenarios de prueba restantes. | • Se abrieron pocos defectos más de ‘Severidad 3’. • Cobertura total 180 Escenarios cubiertos. |

|  |
| --- |
| Cuadro de texto |
| ¿Se debería finalizar o parar en este punto? En estos momentos, se han agotado varios factores como el tiempo de prueba, los informes de defectos y sus soluciones. Sin embargo, ¿es seguro terminar las pruebas? Realmente no, dadas las siguientes razones:   * Los escenarios no se ejecutan por completo. * Pocos flujos que, ni siquiera, se prueban una vez. * Todos los escenarios cubiertos se ejecutan una sola vez. * El software todavía tiene defectos. * La regresión no está cubierta. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Pestañas o tabs horizontales | |
| **Introducción** | **Escenario 2:** | |
| **Semana 1** | Encuentra un defecto de ‘Severidad 1’ el día 1. La prueba completa se bloquea durante 3 días. Por este motivo, no se podrá ejecutar ninguno de los escenarios hasta que se resuelva el defecto de gravedad 1. Luego de perder 3 días, el bloqueador se resuelve y continúa la ejecución.  Al final de la semana, se completan 20 escenarios con algunos defectos más. Esta semana sigue siendo la misma que el escenario 1. | Ilustración del icono vectorial del tramo de tiempo / 1 semana, semanal  Colocar resaltada la primera semana del calendario, no la segunda como está la imagen.  Imagen: 225125\_i25 |
| **Semana 2** | Se encuentran algunos defectos más de ‘Severidad 1’, ‘Severidad 2’ y ‘Severidad 3’ durante la segunda semana. No obstante, el enfoque es cubrir más escenarios para suplir el retraso de la semana 1. Al final de la semana, se cubren 120 escenarios. | Ilustración del icono vectorial del tramo de tiempo / 1 semana, semanal  Imagen: 225125\_i26 |
| **Semana 3** | Al comienzo de la semana 3, se resuelven todos los defectos abiertos, por lo que, junto con la ejecución de los escenarios pendientes, se deben volver a probar todos los defectos que se encuentran en un depósito de prueba. Continuando con un buen progreso hasta el final de la semana, el número de escenarios completados se convierte en 200, con defectos adicionales. Ahora, solo debe informarse los defectos de ‘Severidad 2’ y ‘Severidad 3’. | Ilustración del icono vectorial del tramo de tiempo / 1 semana, semanal  Colocar resaltada la tercera semana del calendario, no la segunda como está la imagen.  Imagen: 225125\_i27 |

|  |
| --- |
| Cuadro de texto |
| La siguiente tabla presenta las actividades de prueba realizadas y el resultado final por semana del escenario 2. |

**Tabla 2**

*Actividades de prueba y resultado final escenario 2*

|  |  |  |
| --- | --- | --- |
| **Semanas** | **Actividades de prueba realizadas** | **Resultado al final de la semana** |
| Semana 1 | • Día 1: muestra el defecto de tapón encontrado. • La prueba está bloqueada debido a un defecto de gravedad 1 encontrado el día 1. • Defecto de bloqueador resuelto el día 4. • La ejecución de la prueba continuó hasta el final de la semana 1. | • Defectos críticos, de alta prioridad, abiertos. • 20 escenarios completados. |
| Semana 2 | • El foco está en ejecutar más escenarios para cubrir el *Backlog* de la semana anterior. • Nueva prueba de defectos reparados. | • Se abrieron pocos defectos más de ‘Severidad 1’, ‘Severidad 2’ y ‘Severidad 3’. • Cobertura total 120 escenarios cubiertos. |
| Semana 3 | • Nueva prueba de todos los defectos de alta prioridad. • Ejecución de los escenarios de prueba restantes. • Solo quedan por encontrar defectos de ‘Severidad 3’ y pocos defectos de ‘Severidad 2’. | • Se abrieron pocos defectos más de ‘Severidad 1’. ‘Severidad 2’ y ‘Severidad 3’. • Todos los escenarios cubiertos. |

|  |
| --- |
| Cuadro de texto |
| Aprendiz, en este momento se debe preguntar, ¿en este punto se debe detener la prueba? Se cubrieron todos los escenarios en su totalidad, incluso, se cubrieron algunos defectos abiertos. No obstante, realmente no es viable detenerse debido a las siguientes razones:   * Todos los escenarios se ejecutan solo una vez. * El *software* todavía tiene muchos defectos importantes. * La regresión no está cubierta.   Se puede ver que la calidad se ve comprometida en ambos escenarios. El mejor enfoque sería encontrar un punto donde todos los factores del escenario 1 y el escenario 2 estén cubiertos, sin que la calidad se vea comprometida. Para lograrlo, se tendrán que definir ciertos criterios al inicio de las pruebas. Estos criterios se denominan criterios de finalización o de salida, siendo la respuesta a cuándo dejar de realizar pruebas. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Tarjetas Avatar | |
| **Introducción** | Los criterios de finalización o de salida, son el conjunto de condiciones o actividades que se deben cumplir para concluir las pruebas. | |
| Se evalúan al final del ciclo de prueba y se definen en el plan de prueba. | | Iconos vectores DevOps. Trazo editable. El desarrollo de software y las operaciones de TI establecen símbolos. Monitor de versión de prueba operar implementar compilación de código de plan  **Imagen** 225125\_i28 |
| Definen cuántas pruebas son suficientes y cuándo se pueden declarar completas las actividades de prueba. | | Iconos vectores DevOps. Trazo editable. El desarrollo de software y las operaciones de TI establecen símbolos. Monitor de versión de prueba operar implementar compilación de código de plan  **Imagen:** 225125\_i29 |

|  |
| --- |
| Cuadro de texto |
| Idealmente, los criterios de salida se definen combinando varios factores y, por lo tanto, son únicos en todos los proyectos. Dependen de los requisitos de cada proyecto y, así, deben definirse durante la planificación de pruebas al inicio del proyecto. Los parámetros definidos en él deben cuantificarse tanto como sea posible. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | | Pestañas o tabs Verticales |
| **Introducción** | | A continuación, se presentan algunos consejos que se deben considerar al definir los criterios de salida en caso de pruebas funcionales o de un sistema que combine algunos o todos los factores, mientras se decide dónde dejar de probar según las necesidades de cada proyecto.  Es así que las pruebas se pueden detener cuando: |
| Ordenador portátil y software que ayuda en el proceso de pruebas, pequeños testers de personas. Prueba automatizada, prueba de ejecución automática, concepto de prueba automática de software. ilustración moderna de vector plano  **Imagen:** 225125\_i30 | | |
| **Requisitos** | Se alcanza el 100 % de cobertura de requisitos. | |
| **Defectos** | * Se alcanza el recuento de defectos definidos o deseados. * Todos los defectos o bloqueadores de *showstopper* están corregidos y ningún defecto conocido crítico, con ‘Severidad 1’ está en estado abierto. * Todos los defectos de alta prioridad se identifican y solucionan. * La tasa de defectos cae por debajo de la tasa aceptable definida. * Muy pocos defectos de prioridad media están abiertos y tienen una solución alternativa. * Muy pocos defectos abiertos de baja prioridad no afectan el uso del *software*. * Todos los defectos de alta prioridad se vuelven a probar, se cierran y los escenarios de regresión correspondientes se ejecutan con éxito. | |
| **Cobertura de prueba** | * La cobertura de la prueba debe alcanzarse en un 95%. * La tasa de aprobación del caso de prueba debe ser del 95 %. Esto se puede calcular por fórmula:   (No total de TC aprobadas / Número total de TC) \* 100.   * Se pasan todos los casos de prueba críticos. * 5% de los casos de prueba pueden fallar, pero los casos de prueba fallidos son de baja prioridad. * Se logra una cobertura funcional completa. * Todos los flujos principales, funcionales/comerciales, se ejecutan con éxito con varias entradas y funcionan bien. | |
| **Plazos** | Se alcanza la fecha límite del proyecto o la fecha límite de finalización de la prueba. | |
| **Documentos de prueba** | Todos los documentos de prueba, o entregables, se preparan, revisan y publican. | |
| **Presupuesto** | El presupuesto de prueba completo está agotado. | |
| **Reuniones *Go/No Go*** | Se ha realizado la reunión con las partes interesadas y se decide si el proyecto debe pasar a producción o no. | |

|  |
| --- |
| Cuadro de texto |
| Para concluir, es importante dejar un resumen o unos *tips* sobre la ejecución de pruebas y cuándo detenerse, para ello se plantearán algunas preguntas de respuesta SÍ - NO.  Si obtiene la mayoría de las respuestas como SI, significa que puede dejar de realizar la prueba en este punto. Si la mayoría de las respuestas son NO, significa que debe verificar lo que falta en las pruebas. Esto puede ayudar en encontrar un defecto de producción que se escapa.   * ¿Se ejecutan todos los casos de prueba al menos una vez? * ¿La tasa de aprobación de casos de prueba es la definida? * ¿Se alcanza una cobertura de prueba completa? * ¿Se ejecutan todos los flujos funcionales/comerciales al menos una vez? * ¿Se alcanza el recuento de defectos decidido? * ¿Se corrigen y cierran todos los defectos importantes de alta prioridad? * ¿Se han vuelto a probar y se han cerrado todos los defectos? * ¿Se ha realizado una regresión para todos los defectos abiertos? * ¿Ha agotado el presupuesto de prueba? * ¿Ha llegado la hora de finalización de la prueba? * ¿Se revisan y publican todos los entregables de prueba? |

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

|  |  |
| --- | --- |
| **Tipo de recurso** | Síntesis |
| Desarrollo de aplicaciones web full stack  Síntesis: Evaluación de pruebas | |
| **Introducción** | El siguiente mapa integra los criterios y las especificidades de los conocimientos expuestos en este componente formativo. |
| Análisis de pruebas  Alcance  Objetivos  Medibles  Alcanzables  Escenarios  Caja negra  Caja blanca  Otros  Escenarios  Descripción  Condiciones  Criterios de finalización  Evaluación mediante pruebas  Plan de pruebas  225125\_i31 | |

**ACTIVIDAD DIDÁCTICA**

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Actividad didáctica. Arrastrar y soltar | |
| Esta actividad le permitirá determinar el grado de apropiación de los contenidos del componente formativo.  De acuerdo con la definición planteada en la columna izquierda, arrastre cada término al lugar que considere correcto de la columna derecha. | | Conjunto de ilustraciones vectoriales de concepto abstracto de implementación de software. Análisis de comparaciones y pruebas de automatización, sistema de colas electrónicas, rendimiento de productos, generador de entradas, metáfora abstracta de soluciones de TI.  **Imagen:** 225125\_i32 |
| Se refiere al proceso de evaluar y verificar que un producto de *software* realiza lo que se supone que debe hacer. | | Prueba de *software*. |
| Encargados de probar los *software*. | | *Testers.* |
| Conocer la calidad de los productos. | | Objetivo prueba de *software.* |
| Debe estar incluido en el proceso de pruebas. | | Identificación de defectos. |
| Tienen por objeto localizar defectos y comprobar el funcionamiento de módulos *software*, programas, objetos, clases, etc., que puedan probarse por separado. | | Pruebas de componentes. |
| Se encargan de probar las interfaces entre los componentes o módulos. | | Pruebas de integración. |

**Retroalimentación general positiva:**

¡Felicitaciones! Ha logrado una óptima aprehensión de los conocimientos relacionados con evaluación de pruebas.

**Retroalimentación general negativa:**

¡Inténtelo de nuevo! Se sugiere revisar, nuevamente, el material de estudio para afianzar los conocimientos presentados.

**MATERIAL COMPLEMENTARIO**

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo de recurso | Material complementario | | |
| Tema | Referencia APA del material | tipo | Enlace |
| Alcance de pruebas | Quality-Stream. (2021). *Tipos y Niveles de Pruebas de Software* [Video]. YouTube. <https://www.youtube.com/watch?v=hSxXuRxA9mo> | Video | <https://www.youtube.com/watch?v=hSxXuRxA9mo> |
| Alcance de pruebas | Quality-Stream. (2021). *Cómo crear un Plan de Pruebas de Software – Norma ISO 29119* [Video]. YouTube. <https://www.youtube.com/watch?v=0anZpU5W0Z8> | Video | <https://www.youtube.com/watch?v=0anZpU5W0Z8> |
| Evaluación de programas mediante pruebas | OpenWebinars. (2019). *Tipos de Pruebas de Software* [Video]. YouTube. <https://www.youtube.com/watch?v=dJTcfuSMwFE> | Video | <https://www.youtube.com/watch?v=dJTcfuSMwFE> |
| Evaluación de programas mediante pruebas | Quality-Stream. (2021). *Cómo reportar defectos – Paso a paso* [Video]. YouTube. <https://www.youtube.com/watch?v=kYeiqPbG5Vg> | Video | <https://www.youtube.com/watch?v=kYeiqPbG5Vg> |

**GLOSARIO**

|  |  |
| --- | --- |
| **Tipo de recurso** | Glosario |
| *Block*: | indica que los elementos estarán organizados en una línea vertical uno abajo del otro. De esa forma los interpreta el navegador. |
| *CSS*: | siglas en inglés que, en español, significan hoja de estilos en cascada. Complementa las páginas web dando estilo y forma. |
| *Inline*: | indica que los elementos estarán organizados en una línea horizontal uno al lado del otro. De esa forma los interpreta el navegador. |
| Parámetro: | especifica el tipo de dato que recibirá un método. |
| Pruebas de aceptación: | se enfocan en la aceptación de los criterios previstos en un contrato de desarrollo de *software* acordado, previamente, entre la empresa de *software* y el cliente. |
| Pruebas de componentes: | tienen por objeto localizar defectos y comprobar el funcionamiento de módulos *software*, programas, objetos, clases, etc., que puedan probarse por separado. |
| Pruebas de integración: | se encargan de probar las interfaces entre los componentes o módulos. |
| Prueba de *software:* | se refiere al proceso de evaluar y verificar que un producto, o aplicación de *software*, realiza lo que se supone que debe hacer. |
| *Responsive:* | propiedad que indica que la página se adapta a todas las pantallas de los diferentes dispositivos donde se ejecute. |
| *Testers:* | encargados de probar los *software*. |

**REFERENCIAS BIBLIOGRÁFICAS:**

|  |  |
| --- | --- |
| **Tipo de recurso** | Bibliografía |
| International Standard. (2015). *ISO/IEC/IEEE24765:2010 Systems and software engineering – Vocabulary.* <https://www.cse.msu.edu/~cse435/Handouts/Standards/IEEE24765.pdf> | |
| Manrique, J. (s.f.). *Pruebas de Software*. Cibertec. <https://www.academia.edu/8813345/Pruebas_de_Software> | |
| Zuluaga, L. & Bedoya, Y. (2018). *Desarrollo de un manual sobre pruebas de software durante el ciclo de desarrollo e implementación* [Tesis de pregrado, Universidad Tecnológica de Pereira]. Repositorio Institucional UTP. <https://repositorio.utp.edu.co/server/api/core/bitstreams/0e705442-9ab0-41a3-aa02-80c241bbf9a3/content> | |