PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE

1. IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE

- Denominación del Programa de Formación: **Técnico en Programación de Software.**
- Código del Programa de Formación: **233104 v.2**
- Nombre del Proyecto: **DESARROLLO Y PROGRAMACIÓN DE SOFTWARE A LA MEDIDA EN LAS INSTITUCIONES EDUCATIVAS EN EL DEPARTAMENTO DE CÓRDOBA.**
- Fase del Proyecto: **Fase I - ANÁLISIS**
- Actividad de Proyecto: **DETERMINAR LAS ESPECIFICACIONES Y NECESIDADES PARA EL DESARROLLO DEL SOFTWARE.**
- Competencia: **220501092 - ESTABLECER REQUISITOS DE LA SOLUCIÓN DE SOFTWARE DE ACUERDO CON ESTÁNDARES Y PROCEDIMIENTO TÉCNICO.**
- Resultados de Aprendizaje Alcanzar: **RAP 1. CARACTERIZAR LA INFORMACIÓN A RECOLECTAR DE ACUERDO CON LAS NECESIDADES DEL CLIENTE.**
- Duración de la Guía: **48 horas**

2. PRESENTACIÓN

Lectura Introductoria:

Establecer requisitos es una habilidad crucial para cualquier persona que esté interesada en el desarrollo de soluciones de software. Con esta competencia, podrás identificar y especificar los requisitos necesarios para crear soluciones de software que cumplan con las necesidades y expectativas de los usuarios finales. Además, aprenderás a verificar y validar estos requisitos de acuerdo con los estándares y procedimientos técnicos establecidos.

Al dominar esta competencia, te convertirás en un profesional de tecnología altamente valorado, capaz de crear soluciones de software que sean efectivas,

eficientes y que cumplan con las expectativas del cliente. Además, desarrollarás habilidades importantes de comunicación y trabajo en equipo que te permitirán trabajar eficazmente en colaboración con otros profesionales de tecnología y stakeholders del proyecto.

Recuerda que la tecnología es un campo en constante evolución y demanda profesionales capacitados y actualizados. Aprender y dominar la competencia "Establecer requisitos de la solución de software de acuerdo con estándares y procedimientos técnicos" es una inversión valiosa en tu futuro profesional y te permitirá mantener una ventaja competitiva en el mercado laboral.

3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

3.1 Actividades de reflexión inicial:

Descripción de la actividad:

Las metodologías de desarrollo de software se refieren a los procesos y técnicas utilizados para diseñar, desarrollar y mantener soluciones de software. Existen dos tipos principales de metodologías de desarrollo de software: ágiles y tradicionales. Cada tipo tiene sus propias características, ventajas y desventajas, y es importante que los estudiantes de tecnología comprendan las diferencias entre ellas.

Ambiente requerido: Ambiente de aprendizaje con Equipos de Cómputo y Software Especializado.

Estrategias o técnicas didácticas activas: Trabajo colaborativo, mesas redondas, lluvia de ideas.

Materiales de formación: Marcadores de diferente color, Video beam o TV, bafles, Conexión a Internet.

Material de apoyo:

Duración de la actividad: 3 horas.

3.2 Actividades de contextualización e identificación de conocimientos necesarios para el aprendizaje:

Descripción de la actividad:

3.2.1 Foro temático: Metodologías de desarrollo de software: ¿Qué factores influyen en la elección de una metodología y cómo se pueden eliciar los requisitos de manera efectiva?

Metodologías de desarrollo de software: ¿Qué factores influyen en la elección de una metodología y cómo se pueden eliciar los requisitos de manera efectiva?

- ¿Cuáles son las principales metodologías de desarrollo de software ágiles y tradicionales y qué ventajas y desventajas ofrecen?
- ¿Qué factores deberíamos considerar para elegir una metodología de desarrollo de software?
- ¿Cómo se pueden eliciar los requisitos de manera efectiva y cuáles son las técnicas de recolección de información más comunes?
- ¿Cómo afecta la elección de la metodología de desarrollo de software a la elicitación de requisitos?
- ¿Qué importancia tiene la colaboración entre los miembros del equipo y los stakeholders en la elicitación de requisitos?
- ¿Cómo se pueden medir y evaluar los requisitos y la satisfacción del cliente en un proyecto de desarrollo de software?

Ambiente requerido: Ambiente de aprendizaje con Equipos de Cómputo y Software Especializado.

Estrategias o técnicas didácticas activas: Trabajo colaborativo, mesas redondas, lluvia de ideas.

Materiales de formación: Marcadores de diferente color, Video beam o TV, bafles, Conexión a Internet.

Material de apoyo:

Duración de la actividad: 3 horas.

3.3 Actividades de apropiación:

Descripción de la actividad:

METODOLOGÍAS DE DESARROLLO DE SOFTWARE ÁGILES Y TRADICIONALES: CONCEPTOS, TIPOS, CARACTERÍSTICAS, VENTAJAS Y DESVENTAJAS.

Las metodologías de desarrollo de software se refieren a los procesos y técnicas utilizados para diseñar, desarrollar y mantener soluciones de software. Existen dos tipos principales de metodologías de desarrollo de software: ágiles y tradicionales. Cada tipo tiene sus propias características, ventajas y desventajas, y es importante que los estudiantes de tecnología comprendan las diferencias entre ellas.

3.3.1 Metodologías Ágiles.

En este contenido se profundiza en las metodologías de desarrollo de software ágiles, definiéndose como un conjunto de procesos iterativos e incrementales que se enfocan en la colaboración entre los miembros del equipo, la adaptación al cambio y la entrega temprana de software funcional. Se describen las características principales de las metodologías ágiles y se presentan los diferentes tipos de metodologías ágiles, como Scrum, Kanban, XP, Lean, entre otras. También se analizan las ventajas y desventajas de las metodologías ágiles.

1. **Características**
   - Enfocadas en la entrega de funcionalidades de manera rápida e incremental.
   - Procesos iterativos y colaborativos que permiten adaptarse a los cambios.
   - Comunicación y colaboración estrecha entre miembros del equipo y stakeholders del proyecto.
   - Enfoque en la calidad del software y en la satisfacción del cliente.
2. **Ventajas:**
   - Mayor flexibilidad para adaptarse a los cambios en los requisitos del proyecto.
   - Entrega rápida de funcionalidades y mayor visibilidad del progreso del proyecto.
   - Mayor interacción y colaboración entre miembros del equipo y stakeholders del proyecto.
   - Mayor enfoque en la satisfacción del cliente y en la calidad del software.
3. **Desventajas:**
   - Puede ser difícil para equipos grandes o distribuidos.

- Requiere un mayor nivel de comunicación y coordinación entre los miembros del Equipo.
- Puede haber una falta de documentación y planificación debido a la naturaleza iterativa del proceso.

3.3.2 Metodologías Tradicionales:

En este contenido se profundiza en las metodologías de desarrollo de software tradicionales, definiéndose como un conjunto de procesos secuenciales que se enfocan en la planificación detallada del proyecto y la documentación exhaustiva de los requerimientos. Se describen las características principales de las metodologías tradicionales y se presentan los diferentes tipos de metodologías tradicionales, como el Modelo en Cascada y el Modelo en Espiral. También se analizan las ventajas y desventajas de las metodologías tradicionales.

1. Características:

- Enfocadas en la planificación y documentación exhaustiva de los requisitos, procesos y entregables del proyecto.
- Procesos secuenciales y estructurados que siguen un ciclo de vida definido.
- Enfoque en la gestión del proyecto y en la minimización de los riesgos

2. Ventajas:

- Mayor nivel de planificación y documentación que facilita la gestión del proyecto.
- Mayor control sobre el proceso de desarrollo y los entregables del proyecto.
- Mayor adecuación para proyectos grandes o complejos.

3. **Desventajas**:

- Menor flexibilidad para adaptarse a los cambios en los requisitos del proyecto.
- Entrega tardía de funcionalidades y menor visibilidad del progreso del proyecto.
- Menor interacción y colaboración entre miembros del equipo y stakeholders del proyecto.

3.3.3 ELICITACIÓN DE REQUISITOS: CONCEPTO, ANÁLISIS DE DOCUMENTOS, TÉCNICAS DE RECOLECCIÓN DE INFORMACIÓN Y CARACTERÍSTICAS.

En este módulo el estudiante debe ser capaz de identificar, analizar, validar y documentar los requisitos de un sistema de software. Esta actividad es esencial para el éxito de cualquier proyecto de software, ya que los requisitos son la base de la planificación, diseño, desarrollo y pruebas del software. La elicitación de requisitos implica una serie de técnicas y herramientas para recopilar información de los usuarios, los stakeholders y otros participantes del proyecto.

Concepto:

En este contenido se explica el concepto de la elicicación de requisitos, definido como el proceso de identificar, comprender y documentar los requerimientos del sistema. Se presentan los diferentes tipos de requerimientos, como los funcionales, no funcionales, explícitos e implícitos. Se puede utilizar una presentación audiovisual o una lluvia de ideas para generar interés en el tema y motivar a los estudiantes a participar en la discusión.

Análisis de Documentos:

En este contenido se describe el análisis de documentos como una técnica utilizada para identificar los requerimientos del sistema a partir de la revisión de documentos existentes, como especificaciones técnicas, manuales de usuario, informes, entre otros. Se presentan las ventajas y desventajas de esta técnica y se realizan ejercicios prácticos en los que los estudiantes analizan diferentes documentos y extraen los requerimientos. Se puede utilizar una dinámica de trabajo en grupo para fomentar la participación y el intercambio de ideas.

Técnicas de Recolección de Información:

En este contenido se presentan las diferentes técnicas de recolección de información utilizadas en la elicicación de requisitos, como entrevistas, cuestionarios, observación, prototipado, entre otras. Se describen las características, ventajas y desventajas de cada técnica y se realizan ejercicios prácticos en los que los estudiantes aplican diferentes técnicas para recopilar los requerimientos. Se puede utilizar una simulación de entrevista o una dinámica de role-playing para que los estudiantes experimenten con diferentes técnicas.

1. **Entrevistas:** las entrevistas son una técnica común de recolección de información. Permiten a los analistas de requisitos obtener información directamente de los stakeholders del proyecto. Las entrevistas pueden ser individuales o en grupo, y pueden ser estructuradas, semiestructuradas o no estructuradas.

2. **Talleres**: los talleres son sesiones de trabajo en grupo que involucran a múltiples stakeholders. Los talleres son una técnica útil para recopilar información de manera colaborativa y permiten a los stakeholders intercambiar ideas y discutir posibles soluciones.

3. **Cuestionarios**: los cuestionarios son una técnica de recolección de información que se utilizan para obtener información de un gran número de stakeholders de manera rápida y eficiente. Los cuestionarios pueden ser diseñados para ser autoadministrados o pueden ser administrados por un entrevistador.

4. **Observación**: la observación es una técnica de recolección de información que implica la observación directa del comportamiento del usuario en su entorno natural. Esta técnica es útil para identificar requisitos que pueden no ser evidentes a través de otras técnicas de elicitación de requisitos.

Características:

En este contenido se describen las características que deben cumplir los requisitos, como ser precisos, claros, consistentes, completos, verificables, entre otros. Se presentan las técnicas utilizadas para validar los requisitos y se realizan ejercicios prácticos en los que los estudiantes validan los requisitos generados en las técnicas anteriores. Se puede utilizar una sesión de preguntas y respuestas para evaluar el aprendizaje de los estudiantes y reforzar los conceptos clave:

- La elicitación de requisitos es un proceso iterativo y continuo.
- La elicitación de requisitos es un proceso colaborativo que involucra a múltiples stakeholders.
- La elicitación de requisitos debe ser enfocada en los objetivos del proyecto y los requisitos del usuario final.
- La elicitación de requisitos debe ser documentada y validada por los stakeholders.

En resumen, la elicitación de requisitos es un proceso crucial en el desarrollo de software y permite a los analistas de requisitos identificar, analizar y documentar los requisitos del software.

Ambiente requerido: Ambiente de aprendizaje con Equipos de Cómputo y Software Especializado.

Estrategias o técnicas didácticas activas: Trabajo colaborativo, mesas redondas, lluvia de ideas.

Materiales de formación: Marcadores de diferente color, Video beam o TV, bafles, Conexión a Internet.

Material de apoyo:

Duración de la actividad: 20 horas.

3.4 Actividades de Transferencia el Conocimiento:

Descripción de la actividad:

3.4.1. Explorando las metodologías de desarrollo de software ágiles y tradicionales: Debatiendo, analizando y simulando procesos

- **Debate en grupo:** Divida a los estudiantes en grupos y pídales que debatan sobre las ventajas y desventajas de las metodologías de desarrollo de software ágiles y tradicionales. Deben presentar argumentos sólidos y sustentar sus posturas en la literatura revisada.
- **Estudio de casos:** Presente a los estudiantes diferentes estudios de casos sobre el uso de metodologías de desarrollo de software ágiles y tradicionales en empresas de diferentes tamaños y sectores. Los estudiantes deben analizar cada caso y discutir en grupo las ventajas y desventajas de cada metodología.
- **Simulación de procesos:** Organice una simulación de procesos de desarrollo de software utilizando diferentes metodologías (por ejemplo, scrum, waterfall). Los estudiantes deben participar en el proceso y luego comparar los resultados de cada metodología.
- **Investigación en equipo:** Divida a los estudiantes en grupos y pídales que investiguen sobre las características de las metodologías de desarrollo de software ágiles y tradicionales. Cada grupo debe presentar su investigación al resto de la clase y promover una discusión abierta sobre las metodologías.

- **Análisis de documentos:** Proporcione a los estudiantes diferentes documentos que describen metodologías de desarrollo de software ágiles y tradicionales, como manuales o guías de usuario. Los estudiantes deben analizar los documentos y presentar las similitudes y diferencias entre cada metodología.

Es importante que el profesor adapte estas actividades de acuerdo al ritmo de la clase y a los objetivos de aprendizaje que se deseen alcanzar.

3.4.2. Aprendiendo a escuchar: técnicas y estrategias para la elicitación de requisitos.

- **Simulación de una sesión de recolección de requisitos**: En esta actividad, los estudiantes formarán grupos y actuarán como miembros de un equipo de desarrollo de software. Se les proporcionará un escenario y un conjunto de preguntas para recolectar requisitos de los usuarios. Los estudiantes practicarán el uso de diferentes técnicas de recolección de información y aprenderán a trabajar en equipo para identificar y priorizar los requisitos.
- **Análisis de documentos**: En esta actividad, los estudiantes revisarán diferentestipos de documentos de requisitos, tales como casos de uso, especificacionesfuncionales y no funcionales, y prototipos. Se les pedirá que identifiquen y analicen la información importante, las posibles lagunas y contradicciones en los requisitos, y propongan posibles soluciones.
- **Juego de roles**: En esta actividad, los estudiantes actuarán como diferentes interesados (stakeholders) involucrados en el proceso de recolección de requisitos, como usuarios, analistas, líderes de proyecto, etc. Se les proporcionará un conjunto de requisitos y un escenario en el que deberán negociar y priorizar los requisitos según las necesidades y expectativas de cada interesado.
- **Presentación y discusión de casos de estudio**: En esta actividad, los estudiantes investigarán y presentarán casos de estudio sobre proyectos de software exitosos y

fracasados relacionados con el proceso de recolección de requisitos. Se fomentará la discusión y el análisis crítico de los procesos y técnicas de recolección de requisitos utilizados en cada caso.

- **Desarrollo de una herramienta de gestión de requisitos**: En esta actividad, los estudiantes trabajarán en grupos para diseñar y desarrollar una herramienta de gestión de requisitos. Deberán considerar diferentes características y funcionalidades que deben tener para asegurarse de que sean útiles para el proceso de recolección y gestión de requisitos.

Al diseñar estas actividades, se fomentará el aprendizaje activo, el trabajo en equipo, la investigación y la reflexión crítica sobre los procesos de recolección de requisitos en el desarrollo de software.

Ambiente requerido: Ambiente de aprendizaje con Equipos de Cómputo y Software Especializado.

Estrategias o técnicas didácticas activas: Trabajo colaborativo, mesas redondas, lluvia de ideas.

Materiales de formación: Marcadores de diferente color, Video beam o TV, bafles, Conexión a Internet.

Material de apoyo:

Duración de la actividad: 22 horas.

4. PLANTEAMIENTO DE EVIDENCIAS DE APRENDIZAJE PARA LA EVALUACIÓN EN EL PROCESO FORMATIVO.

Fase del proyecto formativo	Actividad del proyecto formativo	Actividad de Aprendizaje	Evidencias de Aprendizaje	Criterios de Evaluación	Técnicas e Instrumentos de Evaluación
ANALISIS	DETERMINAR LAS ESPECIFICACIONES Y NECESIDADES PARA EL DESARROLLO DEL SOFTWARE	Determinar las especificaciones y necesidades para el desarrollo del software	informe De requerimientos del sistema de información necesarios para iniciar el proceso de construcción del software.	Determina las técnicas de elicitación de acuerdo con las necesidades del cliente. Aplica los instrumentos de recolección de información según las técnicas seleccionadas	Lista de chequeo

5. GLOSARIO DE TÉRMINOS

Metodologías de desarrollo de software: Conjunto de procesos y prácticas utilizados para planificar, diseñar, desarrollar, probar y mantener software.

Metodologías ágiles: Conjunto de metodologías de desarrollo de software que se enfocan en la colaboración y la entrega incremental de funcionalidades.

Metodologías tradicionales: Conjunto de metodologías de desarrollo de software que se enfocan en la planificación detallada y la entrega en una sola versión.

Proceso incremental: Proceso en el que el producto final se construye a través de la entrega de funcionalidades en incrementos.

Colaboración: Trabajo en equipo en el que se comparten responsabilidades, conocimientos y habilidades.

Requisitos de software: Conjunto de características y funcionalidades que debe tener un software.

Elicitación de requisitos: Proceso de identificar, obtener y documentar los requisitos de software de un cliente o usuario.

Análisis de documentos: Proceso de examinar documentos para identificar requisitos de software.

Técnicas de recolección de información: Métodos utilizados para recopilar información sobre los requisitos de software, como entrevistas, cuestionarios y observación.

Stakeholders: Personas o grupos interesados en el proyecto de software, como los usuarios, el cliente y los patrocinadores.
Cliente líder: Persona responsable de proporcionar una visión clara del proyecto de software y de la gestión de los requisitos.

Dueño del producto: Persona responsable de definir y priorizar los requisitos de software.

Equipo de desarrollo: Grupo de personas encargadas de desarrollar el software.

Analista de requisitos: Persona encargada de recopilar, analizar y documentar los requisitos de software.

Usuarios: Personas que utilizarán el software.

Actores: Personas que interactúan con el software pero no lo utilizan directamente.

Diagrama de casos de uso: Diagrama utilizado para representar los requisitos funcionales del software.

Historias de usuario: Descripción breve y sencilla de una funcionalidad o requisito del software.

Prototipado: Creación de un modelo o demostración del software para validar los requisitos.

Ventajas de las metodologías ágiles: Flexibilidad, adaptabilidad a cambios, entrega rápida de valor, involucramiento del cliente y colaboración.

Desventajas de las metodologías ágiles: Falta de documentación, dificultad para estimar el tiempo y los costos, dependencia de la comunicación.

Ventajas de las metodologías tradicionales: Planificación detallada, control de calidad, documentación completa.

Desventajas de las metodologías tradicionales: Falta de flexibilidad, difícil adaptación a cambios, entrega tardía de valor, falta de colaboración.

Caso de negocio: Documento que describe las razones comerciales y financieras para el desarrollo de un proyecto de software.
Ciclo de vida de software: Fases por las que pasa un software, como la planificación.

Documento de requisitos: Es un documento formal que describe los requisitos del software a desarrollar, su propósito es establecer una guía para los desarrolladores y los clientes.

Sprint: Un sprint es un periodo de tiempo fijo durante el cual se desarrolla una funcionalidad del software, generalmente de 1 a 4 semanas.
Refactorización: Es el proceso de modificar el código fuente de un software para mejorar su estructura interna sin cambiar su comportamiento externo.

Pruebas unitarias: Son pruebas automatizadas que se ejecutan en pequeñas secciones del código, conocidas como unidades, para verificar que cada unidad funciona correctamente.

Ciclo de vida del software: Es el proceso que sigue el software desde su concepción hasta su retiro, incluyendo el análisis de requisitos, diseño, implementación, pruebas, mantenimiento y retirada.

Diagrama de casos de uso: Es un diagrama que describe cómo un usuario interactúa con un sistema para lograr un objetivo específico.

Prototipo: Es una versión inicial del software, que se utiliza para probar y validar los requisitos.

Historias de usuario: Son descripciones breves y simples de las características o funcionalidades que el software debe proporcionar al usuario.

Verificación: Es el proceso de asegurarse de que el software cumple con los requisitos especificados.

Validación: Es el proceso de asegurarse de que el software cumple con las expectativas del usuario y los requisitos del negocio.

Diagrama de Gantt: Es una herramienta gráfica que se utiliza para planificar y programar tareas en un proyecto.

Interfaz de usuario: Es la parte del software que permite la interacción del usuario con el sistema.

Integración continua: Es una práctica de desarrollo de software que implica integrar el código fuente en un repositorio común varias veces al día, para detectar y resolver errores rápidamente.

Backlog del producto: Es una lista de todas las características o funcionalidades que deben incluirse en el software.

Estimación de costos: Es el proceso de determinar los costos necesarios para desarrollar un software.

Modelo de proceso de software: Es un conjunto de actividades que se llevan a cabo para desarrollar software de manera sistemática y controlada.

Diagrama de flujo de datos: Es un diagrama que describe cómo fluyen los datos a través del sistema.

Integración de sistemas: Es el proceso de combinar diferentes sistemas o componentes paracrear un sistema más grande y complejo.

Métricas de software: Son medidas utilizadas para evaluar la calidad, eficiencia y eficacia del software.

Pruebas de aceptación: Son pruebas realizadas para asegurarse de que el software cumple con las expectativas del usuario y los requisitos del negocio.

6. REFERENTES BILBIOGRÁFICOS

- Agile Estimating and Planning, de Mike Cohn
- User Stories Applied: For Agile Software Development, de Mike Cohn
- Extreme Programming Explained: Embrace Change, de Kent Beck
- Agile Software Development: Principles, Patterns, and Practices, de Robert C. Martin
- Scrum: The Art of Doing Twice the Work in Half the Time, de Jeff Sutherland
- The Mythical Man-Month: Essays on Software Engineering, de Frederick P. Brooks Jr.
- Requirements Engineering: From System Goals to UML Models to Software Specifications, de Axel van Lamsweerde
- Software Requirements, de Karl Wiegers y Joy Beatty
- Writing Effective Use Cases, de Alistair Cockburn

- Agile Testing: A Practical Guide for Testers and Agile Teams, de Lisa Crispin y Janet Gregory
- The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, de Eric Ries
- Clean Code: A Handbook of Agile Software Craftsmanship, de Robert C. Martin
- Refactoring: Improving the Design of Existing Code, de Martin Fowler
- Design Patterns: Elements of Reusable Object-Oriented Software, de Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides
- Domain-Driven Design: Tackling Complexity in the Heart of Software, de Eric Evans
- Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, de Jez Humble y David Farley
- The Art of Agile Development, de James Shore y Shane Warden
- Succeeding with Agile: Software Development Using Scrum, de Mike Cohn
- Agile Project Management with Scrum, de Ken Schwaber
- Agile Retrospectives: Making Good Teams Great, de Esther Derby y Diana Larsen

7. CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	**SNEIDER DAVID PINEDA SANCHEZ**	**INSTRUCTOR**	**CCIT**	**11/02/2026**

8. CONTROL DE CAMBIOS (diligenciar únicamente si realiza ajustes a la guía)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					