Ingeniería de Software

Guía para el profesor

Clave LSTI2317

TECMILENIO

Contenido

Datos generales

Nombre del certificado: Ingeniería de Software

Nivel: Profesional

Modalidad: Presencial

Clave: LSTI2317

Competencia global

Analiza y evalúa las metodologías para diseñar, desarrollar e implementar soluciones de software innovadoras y eficientes, que se ajusten a las necesidades específicas de empresas y organizaciones.

Competencias esenciales

- Agilidad para el aprendizaje.
- Adaptabilidad.
- Solución de problemas.
- Colaboración.
- Comunicación efectiva.

Introducción

En la actualidad interactúas diariamente con el software en múltiples ámbitos: desde tus dispositivos personales, pasando por herramientas académicas y laborales, hasta las máquinas que construyen, transportan o ejecutan tareas en beneficio del ser humano.

Es esencial que comprendas las metodologías y modelos necesarios para su desarrollo. En este certificado estudiarás los fundamentos de la ingeniería de software mediante casos prácticos, principios de ética profesional y una revisión profunda del ciclo de vida del software y sus fases. Además, conocerás los principios del desarrollo ágil y cómo aplicarlos en productos de software.

Aprenderás a realizar el levantamiento de requerimientos, analizarlos, transformarlos en historias de usuario y escenarios, así como a explorar diversas arquitecturas de software adaptadas a distintos entornos. También abordarás el despliegue en servicios en la nube utilizando microservicios, garantizando en todo momento prácticas de seguridad, privacidad y aseguramiento de la calidad.

Finalmente, comprenderás cómo gestionar un proyecto desde su conceptualización hasta el cierre, y reflexionarás sobre el futuro del software a partir del análisis de tendencias actuales, inteligencia artificial y computación cuántica.

Recuerda que, como parte de esta experiencia, deberás obtener la certificación correspondiente en la plataforma LinkedIn Learning. Haz clic aquí para consultar el material asignado.

Información general

Metodología

El modelo académico **MAPS** se caracteriza por ser modular, apilable y personalizable con un enfoque flexible y centrado en el estudiante. Implementamos técnicas didácticas que buscan no solo la adquisición de conocimientos teóricos, sino también la aplicación práctica y el desarrollo de competencias profesionales altamente valoradas por los empleadores. A continuación, se detallan las técnicas didácticas y características principales de nuestro modelo académico.

Técnicas didácticas

Aprendizaje basado en retos. El alumno demuestra la adquisición de los conocimientos y los aplica por medio de retos propuestos.

Aprendizaje basado en proyectos. El alumno demuestra la adquisición de los conocimientos y los aplica en la práctica, por medio de proyectos que impacten de manera positiva a las organizaciones.

Aula invertida. Esta metodología promueve el autoestudio fuera de las clases, para que, una vez que los alumnos se encuentren en el aula virtual, se promueva la interacción, la construcción conjunta del conocimiento, la generación de ideas y el desarrollo de las competencias, gracias al acompañamiento de docentes expertos.

El **aprendizaje basado en retos** se implementa del primero al quinto semestre, el **aprendizaje basado en proyectos** se aplica del sexto semestre en adelante y la metodología de **aula invertida** está presente en todos los certificados.

En las Semanas de Desarrollo Integral (SeDI) y los certificados de idioma, solamente aplica la metodología de aula invertida.

Características

1. Certificados:
    a. El modelo está formado por certificados de especialidad, los cuales buscan el desarrollo y la adquisición de competencias requeridas por los principales empleadores de nuestro país a través del aprendizaje activo.
    b. Todos los certificados son creados en alianza y colaboración con empresas de prestigio nacional e internacional y/o con expertos que cuentan con conocimiento técnico actual y académico que se requiere en las distintas industrias, con lo que se garantiza el desarrollo de competencias profesionales.
    c. En cada período, el estudiante lleva un máximo de dos certificados simultáneos, con ello los estudiantes tienen la oportunidad de profundizar más en cada tema. Esto es especialmente valioso en cursos que requieren una comprensión detallada de teorías complejas, aplicaciones prácticas y habilidades analíticas avanzadas.

2. Duración:
    La duración de la licenciatura varía según el formato elegido. Los programas ejecutivos se cursan en 15 bimestres, mientras que los programas semestrales se cursan en 8 semestres. Ambos están compuestos por los mismos certificados en sus mapas curriculares, lo que permite transitar entre ambas modalidades dependiendo de las necesidades de los estudiantes.

3. Flexibilidad:
    Este modelo promueve la participación de los estudiantes al permitirles personalizar su experiencia de aprendizaje de acuerdo con sus intereses y necesidades individuales. Esta personalización no solo facilita

un mayor compromiso y motivación, sino que también prepara a los estudiantes para enfrentar retos específicos de su futuro campo profesional, aumentando así su empleabilidad y éxito académico.

4. Credenciales apilables:
   La idea atrás de estas credenciales es proveer un esquema de capacitación y aprendizaje para los aprendedores, de tal forma que puedan moverse rápidamente en el proceso educativo, aprendiendo habilidades que son aplicables en el trabajo. Las credenciales, por lo tanto, pueden ser apiladas para cumplir con el estándar de un programa de grado tradicional.

5. Insignias digitales:
   Las insignias digitales permiten documentar la educación de los estudiantes, así como sus logros. Una de las ventajas de las insignias digitales es que, a través de la metadata, se pueden obtener los detalles de las competencias adquiridas, la institución que otorga la insignia, así como un reconocimiento visual que puede ser compartido en redes sociales o redes profesionales.

6. Diferenciadores del modelo:
   a. Certificados de lengua extranjera: se cuenta con certificados para adquirir o reforzar el dominio de lengua extranjera y con certificados impartidos en una lengua extranjera específicos de la disciplina, todo con el objetivo de atender las demandas de los empleadores.
   b. Semanas de Desarrollo Integral: unidades de aprendizaje transversal, diseñadas para vivir una experiencia inmersiva, desarrollando las competencias humanas, profesionales y de bienestar.
   c. Períodos de Skilling: período complementario donde el alumno puede llevar a cabo actividades que suman a su formación académica. Son opcionales y personalizadas, ya que el estudiante las selecciona con base en sus intereses profesionales y personales.
   d. Estancia empresarial al final del programa de estudios: los estudiantes tendrán a su disposición tres opciones en función de la estancia empresarial que vayan a realizar, entre las cuales se encuentran: gestión de proyectos, emprendimiento y desarrollo sostenible.

Bibliografía y software

Bibliografía de apoyo:

- D'Andrade, B. (2020). *Software Engineering: Artificial Intelligence, Compliance, and Security.* Estados Unidos: Nova Science. ISBN: 9781536189896
- Wright, C. (2022). *Agile Project Management, Assurance and Auditing: A Practical Guide for Auditors, Reviewers and Project Teams*. Reino Unido: ITGP. ISBN: 9781787783553

Consulta gratuitamente la versión electrónica (e-book) de ambos libros en la Biblioteca Digital: https://biblioteca.tecmilenio.mx/

Software:

- Atlassian. (s.f.). *Los buenos resultados comienzan con Jira*. Recuperado de https://www.atlassian.com/es/software/jira
- GitHub. (2024). *AI that builds with you*. Recuperado de https://github.com/features/copilot
- Microsoft. (s.f.). *Te damos la bienvenida a la aplicación Microsoft 365 Copilot*. Recuperado de https://www.office.com/

Evaluación

La evaluación combina los siguientes elementos:

- Actividades que abordan el contenido conceptual de los temas.
- Reto mediante el cual el participante demostrará que ha adquirido las habilidades y los conocimientos necesarios para acreditar el certificado. Este reto se divide en dos fases.
- Presentación del reto.

A continuación, se presenta el detalle de la evaluación:

Semana	Evaluable	Ponderación
1	Actividad 1	6 %
2	Actividad 2	6 %
3	Avance del proyecto	25 %
4	Actividad 3	6 %
5	Actividad 4	6 %
6	Actividad 5	6 %
7	Entrega final del proyecto	35 %
8	Presentación del proyecto	10 %
Semana de Assessment		
	Total	**100 %**

Avance y entrega final del proyecto

El avance y la entrega final del reto se realizarán de manera individual.

Con el fin de fomentar el dinamismo y la interacción entre los participantes en diversos formatos, el profesor alternará, durante las sesiones, intervenciones individuales, plenarias y grupales. Estas actividades enriquecerán tus perspectivas y, al mismo tiempo, te ofrecerán la oportunidad de presentar tus ideas y posturas respecto a los temas de clase.

Los resultados del avance y la entrega final del reto deberán presentarse a través de la plataforma tecnológica para su revisión y evaluación por parte del docente. Es muy importante que revises el esquema de evaluación y los criterios que utilizará el docente para otorgarte una calificación. Lo anterior con la intención de que desde el inicio tengas claro el nivel de complejidad y esfuerzo que requieres para realizar las entregas semanales y garantizar tu éxito.

En caso de dudas sobre el avance, la entrega final del reto o el contenido, puedes contactar a tu docente a través de los medios que se te indiquen.

Calendario de entregas semestral

Semana	Evaluable
1	Actividad 1
2	Actividad 2
3	Avance del proyecto
4	Actividad 3
5	Actividad 4
6	Actividad 5
7	Entrega final del proyecto
8 Semana de Assessment	Presentación del proyecto

Temario

1. **Fundamentos de la ingeniería de software**
   1.1. Conceptos generales
   1.2. Ética en la ingeniería de software
   1.3. Casos prácticos de uso del software

2. **Ciclo de vida del software**
   2.1. Ciclo de vida del software
   2.2. Fases del ciclo de vida del software

3. **Proceso del software**
   3.1. Modelo cascada
   3.2. Modelo espiral
   3.3. Modelo incremental
   3.4. Modelo proceso evolutivo

4. **Ingeniería de software ágil**
   4.1. Métodos ágiles
   4.2. Programación extrema
   4.3. Scrum

5. **Proyectos de software vs. productos de software**
   5.1. La visión del producto
   5.2. Gestión de productos de software
   5.3. Prototipado de productos

6. **Ingeniería de requerimientos**
   6.1. Proceso de levantamiento de requerimientos
   6.2. Proceso de análisis de requerimientos
   6.3. Habilidades del ingeniero de requerimientos

7. **Definición de requerimientos características, escenarios e historias de usuario**
   7.1. Personas y escenarios
   7.2. Historias de usuario
   7.3. Identificación de características

8. **Arquitectura de software**
   8.1. Diseño y modelado de software
   8.2. Diseño arquitectónico
   8.3. Descomposición del sistema
   8.4. Arquitectura de distribución
   8.5. Modelos de arquitectura en software

9. **Software basado en la nube**
   9.1. Virtualización y contenedores
   9.2. Software como servicio

9.3. Arquitectura de software en la nube

10. Arquitectura de microservicios

10.1. Microservicios y su arquitectura
10.2. Servicios Restfull
10.3. Implementación del servicio

11. Seguridad y privacidad

11.1. Ataques y defensas
11.2. Autenticación y autorización
11.3. Cifrado y privacidad

12. Calidad de software

12.1. Definición de la calidad
12.2. Modelos de calidad de software
12.3. Métricas de calidad

13. Pruebas

13.1. Pruebas funcionales
13.2. Desarrollo basado en pruebas
13.3. Pruebas de seguridad
13.4. Revisiones de código

14. DevOps y gestión del código

14.1. Integración y entrega continua (CI/CD)
14.2. Contenedores y orquestación
14.3. Monitoreo y logging
14.4. Medición de DevOps

15. Despliegue y mantenimiento de software

15.1. Despliegue del software características
15.2. Mantenimiento de software
15.3. Gestión de versiones

16. Procesos de desarrollo de software

16.1. Modelos de desarrollo
16.2. Gestión del cambio en la ingeniería de software
16.3. Integración de modelos de madurez de capacidades (CMMI)

17. Gestión de proyectos - parte 1

17.1. Definición del proyecto
17.2. Planificación del proyecto
17.3. Administración de personas y recursos
17.4. Técnicas de estimación (recursos)
17.5. Cronograma del proyecto

18. Gestión de proyecto - parte 2

- 18.1. Técnicas de estimación (tiempos)
- 18.2. Ejecución y control
- 18.3. Administración de riesgos y calidad
- 18.4. Comunicación y liderazgo
- 18.5. Cierre y evaluación del proyecto

19. Ingeniería de software auxiliada por inteligencia artificial

- 19.1. Aplicaciones IA para ingeniería de software
- 19.2. Desarrollo de software con técnicas de IA
- 19.3. Desafíos y oportunidades en la integración de IA

20. Tendencias y retos futuros en la ingeniería de software

- 20.1. Ingeniería de software en la cuarta revolución industrial
- 20.2. Ingeniería de software en la era de la computación cuántica

Preguntas más frecuentes

¿En dónde o a quién le reporto un error detectado en el contenido?

Lo puedes reportar a través del botón “Mejora tu curso”, también puedes compartir sugerencias para el contenido y actividades del certificado.

¿Quién me informa de la cantidad de sesiones y el tiempo de cada sesión en las semanas?

El coordinador docente te debe proporcionar esta información.

¿Tengo que capturar las calificaciones en Banner y en la plataforma educativa?

Sí, es importante que captures las calificaciones en la plataforma para que los participantes estén informados de su avance y reciban retroalimentación de parte tuya de todo lo que realizan en esta experiencia educativa. En Banner es el registro oficial de las calificaciones de los participantes.

Recomendaciones para la explicación de temas, actividades y proyecto

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 1:

Al profesor impartidor, se le recomienda lo siguiente:

- Presenta la definición de software e ingeniería de software.
- Destaca la importancia del ciclo de vida del desarrollo de software (SDLC) y sus fases: planificación, diseño, implementación, pruebas, despliegue y mantenimiento.
- Aborda las categorías del software como sistemas de aplicación, de ingeniería y ciencias, incrustado, de línea de productos, web e inteligencia artificial.
- Explica los principios éticos en el desarrollo de software como la privacidad, seguridad, responsabilidad social, y código de conducta profesional.
- Destaca la escalabilidad como factor clave para la evolución y mantenimiento de sistemas.
- Presenta casos y solicita a los aprendedores proponer una metodología de desarrollo adecuada, identificando fases del SDLC involucradas.
- Propicia el diálogo para plantear dilemas éticos comunes en el desarrollo de software (ejemplo privacidad de datos en aplicaciones de pago) y debatir posibles soluciones.
- Muestra la clasificación de software proporcionando ejemplos reales de productos y pide a los estudiantes clasificarlos según las categorías de software vistas en clase.
- Referencia la importancia de la escalabilidad en aplicaciones populares como Netflix o WhatsApp.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 2:

Al profesor impartidor, se le recomienda lo siguiente:

- Presenta la definición del ciclo de vida del software (SDLC) como marco estructurado para gestionar el desarrollo de software mediante las fases:
    - Planificación (identificación de necesidades, expectativas, análisis de viabilidad, análisis de riesgos y creación de casos de uso).
    - Diseño (creación de arquitectura técnica, diagramas, UX/UI y documentación técnica).
    - Implementación programación basada en la arquitectura, buenas prácticas, integración continua).
    - Pruebas (verificación de requisitos con pruebas unitarias, de integración, funcionales y de aceptación).
    - Despliegue (preparación del entorno, migración de datos, estrategia de despliegue gradual).
    - Mantenimiento (corrección de errores, mejoras, actualizaciones, monitoreo, documentación continua y refactorización).
- Explica un caso de análisis colaborativo en equipos y asignándoles la tarea de diseñar el ciclo de vida de un sistema para una empresa y que identifiquen fases, tareas, tecnologías y posibles retos.
- Simula una reunión de planificación para levantar requerimientos, donde algunos estudiantes serán *stakeholders* y otros el equipo técnico.
- Solicita a los aprendedores construir un diagrama en grupo que conecte cada fase del SDLC con las actividades específicas que se realizan, incluyendo ejemplos concretos (ejemplo entrevistas, pruebas unitarias, despliegue controlado).
- Pregunta al grupo qué fase consideran más crítica en un proyecto de software y por qué, fomentando la argumentación basada en ejemplos y casos reales.
- Menciona proyectos reales donde la falta de una fase adecuada (ejemplo: pruebas) causó problemas serios, como fallas en lanzamientos de aplicaciones móviles o caídas de plataformas digitales.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 3:

Al profesor impartidor, se le recomienda lo siguiente:

- Organiza actividades donde los estudiantes puedan comparar y analizar los distintos modelos de desarrollo de software (cascada, espiral, incremental y evolutivo) mediante estudios de caso. Por ejemplo, asigna a cada equipo un escenario realista (como el desarrollo de un sistema hospitalario, una app fintech o una plataforma educativa) y pídeles justificar qué modelo sería el más adecuado y por qué, considerando factores como estabilidad de requisitos, participación del usuario y recursos disponibles.
- Promueve la simulación de fases de cada modelo. Puedes dividir la clase en equipos y asignarles roles para simular las etapas del modelo en cascada (requerimientos, diseño, implementación, pruebas, despliegue, mantenimiento), o bien, ciclos iterativos del modelo espiral, donde cada ciclo incluya planeación, análisis de riesgos, desarrollo de prototipo y evaluación. Utiliza herramientas colaborativas como Miro, Lucidchart o Google Jamboard para que los estudiantes diagramen y documenten cada fase.
- Incentiva el uso de herramientas de prototipado rápido (Figma, Balsamiq, Marvel) para que los estudiantes experimenten el enfoque evolutivo e incremental. Así podrán crear prototipos funcionales, recibir retroalimentación de sus compañeros y mejorar sus propuestas en iteraciones sucesivas, reforzando el aprendizaje sobre la importancia de la retroalimentación y la adaptación de requisitos.
- Fomenta debates guiados sobre las ventajas y desventajas de cada modelo, utilizando preguntas detonadoras como: ¿En qué tipo de proyectos sería más adecuado el modelo espiral en lugar de cascada? o ¿qué ventajas ofrece el modelo incremental frente al evolutivo? Esto ayudará a los estudiantes a reflexionar críticamente y a argumentar sus elecciones en función del contexto del proyecto.
- Propicia ejercicios de análisis de riesgos, especialmente al abordar el modelo espiral. Asigna a los estudiantes la tarea de identificar posibles riesgos en un proyecto ficticio y plantear estrategias para mitigarlos, documentando el ciclo de retroalimentación y mejora continua.
- Recomienda el uso de foros de discusión y sesiones de videoconferencia con funciones de compartir pantalla, para que los estudiantes presenten sus avances, reciban retroalimentación y resuelvan dudas en tiempo real, facilitando la colaboración y el aprendizaje activo.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 4:

Al profesor impartidor, se le recomienda lo siguiente:

- Organiza actividades prácticas donde los estudiantes simulen la aplicación de metodologías ágiles, como Scrum y Programación Extrema (XP), en proyectos reales o casos de estudio. Puedes dividir a los estudiantes en equipos y asignarles roles específicos (Product Owner, Scrum Master, desarrolladores), promoviendo la rotación de roles para que todos experimenten distintas responsabilidades.
- Promueve el análisis y discusión de los 12 principios del Manifiesto Ágil. Solicita a los estudiantes que identifiquen ejemplos concretos de cómo estos principios pueden aplicarse en proyectos con requisitos cambiantes. Utiliza foros de discusión o pizarras colaborativas (como Miro o Jamboard) para que los equipos compartan sus reflexiones y conclusiones.
- Propicia ejercicios de elaboración y priorización de un Product Backlog utilizando herramientas digitales como Trello, Jira o Asana. Pide a los estudiantes que definan ítems de backlog, los prioricen y simulen una sesión de Sprint Planning, seleccionando qué elementos abordarán en un sprint ficticio.
- Incentiva la práctica de ceremonias ágiles, tales como Daily Scrum, Sprint Review y Sprint Retrospective, a través de videollamadas programadas. Sugiere que los equipos documenten los acuerdos y aprendizajes de cada ceremonia, fomentando la mejora continua y la transparencia en el proceso.
- Fomenta la programación en parejas (pair programming) en actividades de codificación, utilizando plataformas como Visual Studio Live Share o CodeTogether. Invita a los estudiantes a reflexionar sobre las ventajas y retos de este enfoque, comparándolo con el desarrollo individual.
- Recomienda la automatización de pruebas unitarias en ejercicios de desarrollo, resaltando la importancia de la retroalimentación continua y la validación temprana de funcionalidades, como lo sugiere la metodología XP.

- Propicia la colaboración cercana con un “cliente” ficticio (puede ser un profesor o un estudiante designado), quien provea retroalimentación sobre los incrementos de producto desarrollados. Esto refuerza la importancia de la comunicación constante y la adaptación a los cambios de requisitos.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 5:

Al profesor impartidor, se le recomienda lo siguiente:

- Organiza actividades que permitan a los estudiantes distinguir claramente entre proyecto de software y producto de software. Propón casos prácticos, como el de “Adopta un amigo peludo”, para que los estudiantes identifiquen cuándo una iniciativa es un proyecto (con inicio y fin definidos) y cuándo se convierte en producto (requiere mantenimiento y evolución constante).
- Promueve ejercicios en los que los estudiantes desarrollen la visión de un producto. Pide que redacten una declaración de visión clara y concisa para un caso específico, asegurándose de que esté centrada en el problema a resolver y no en la solución técnica. Sugiere que compartan y discutan sus visiones en equipos para fomentar la aceptación y alineación entre todos los miembros.
- Propicia la elaboración y priorización de historias de usuario. Solicita que los estudiantes utilicen el formato recomendado (“Yo como [perfil], quiero [actividad] para [agregar valor]”) para documentar necesidades de usuarios reales o ficticios. Después, organiza sesiones de revisión cruzada en las que los equipos den retroalimentación sobre la claridad y el valor de las historias generadas.
- Incentiva la creación de un road map visual. Utiliza herramientas como Trello, Miro o Google Sheets para que los estudiantes organicen funcionalidades y características por prioridad y cronograma, incluyendo hitos y fechas de lanzamiento. Pide que presenten sus road maps y expliquen cómo estos contribuyen a la estrategia del producto.
- Recomienda el uso de herramientas de prototipado (por ejemplo, Figma, Balsamiq, Marvel) para que los estudiantes creen prototipos funcionales de sus productos. Sugiere que realicen pruebas rápidas con usuarios o compañeros y documenten la retroalimentación recibida, enfocándose en cómo los prototipos ayudan a clarificar requisitos y reducir errores antes del desarrollo completo.
- Fomenta la reflexión sobre la flexibilidad de la visión del producto, las ventajas de las historias de usuario frente a los requisitos tradicionales y el valor del prototipado en la comunicación con los clientes. Utiliza preguntas detonadoras para guiar debates y foros en línea, promoviendo el pensamiento crítico y la aplicación práctica de los conceptos.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 6:

Al profesor impartidor, se le recomienda lo siguiente:

- Organiza actividades en las que los estudiantes analicen diferentes estilos y arquitecturas de software (por ejemplo, arquitectura en capas, cliente-servidor, basada en componentes). Propón casos prácticos para que los equipos seleccionen y justifiquen la arquitectura más adecuada según los requisitos y restricciones del sistema a desarrollar.
- Promueve el uso de diagramas de diseño, como diagramas de clases, de componentes y de paquetes, utilizando herramientas como Lucidchart, draw.io o StarUML. Solicita que los estudiantes documenten tanto la estructura estática como el comportamiento dinámico del sistema, reforzando la importancia de la notación y la claridad en la comunicación técnica.
- Propicia ejercicios de identificación y aplicación de patrones de diseño (como Singleton, Observer, Factory, MVC). Asigna a los equipos la tarea de seleccionar un patrón apropiado para un problema específico y justificar su elección, implementando ejemplos prácticos en el lenguaje de programación de su preferencia.
- Incentiva el análisis de la calidad del diseño a través de la revisión entre pares. Pide a los estudiantes que evalúen los diseños de otros equipos considerando criterios como cohesión, acoplamiento, reutilización y mantenibilidad. Esto fomentará la reflexión crítica y la mejora continua en el proceso de diseño.

- Recomienda la integración de principios de diseño, como los principios SOLID, en las actividades de codificación y revisión. Sugiere ejercicios donde los estudiantes refactoricen fragmentos de código para mejorar su estructura y calidad, documentando los cambios realizados y el impacto en la mantenibilidad del sistema.
- Fomenta la discusión sobre la importancia de la documentación del diseño. Solicita que los estudiantes generen documentación técnica clara y concisa, incluyendo descripciones de la arquitectura, decisiones de diseño y justificaciones, utilizando plantillas estándar o herramientas colaborativas en línea.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 7:

Al profesor impartidor, se le recomienda lo siguiente:

- Organiza actividades en las que los estudiantes creen y documenten “personas” a partir de datos ficticios o reales sobre usuarios finales del sistema. Propón casos prácticos, como el desarrollo de una aplicación bancaria, para que los equipos diseñen al menos dos personas con características, necesidades y frustraciones bien diferenciadas, utilizando herramientas visuales como mapas de empatía para profundizar en la comprensión de los usuarios.
- Promueve la elaboración de escenarios detallados donde cada persona interactúe con el sistema en situaciones específicas. Solicita que los estudiantes describan paso a paso cómo sus personas enfrentarían tareas clave dentro de la aplicación, identificando posibles obstáculos y oportunidades de mejora en la experiencia de usuario.
- Propicia ejercicios de redacción de historias de usuario siguiendo el formato estándar (“Yo como [tipo de usuario] quiero [acción] para [beneficio]”). Asegúrate de que cada historia esté alineada con las necesidades identificadas en las personas y escenarios creados. Incluye la definición de criterios de aceptación claros y verificables para cada historia.
- Incentiva la aplicación del modelo INVEST en la evaluación de historias de usuario. Pide a los estudiantes que revisen y ajusten sus historias para que sean independientes, negociables, valiosas, estimables, pequeñas y testeables. Organiza sesiones de retroalimentación entre equipos para fortalecer la calidad de las historias y fomentar el aprendizaje colaborativo.
- Recomienda el uso de herramientas colaborativas como Miro, Google Jamboard o Trello para la documentación y presentación de personas, escenarios y mapas de empatía. Esto facilitará la visualización y discusión de los resultados en sesiones sincrónicas o asincrónicas.
- Fomenta la reflexión sobre la importancia de centrar el desarrollo en el usuario final. Utiliza preguntas detonadoras como: ¿Cómo ayuda la definición de personas y escenarios en el diseño de una mejor experiencia de usuario? y ¿por qué es importante que las historias de usuario sean pequeñas y testeables? para guiar debates y foros en línea.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 8:

Al profesor impartidor, se le recomienda lo siguiente:

- Organiza actividades donde los estudiantes analicen casos reales de empresas que han migrado de arquitecturas obsoletas a arquitecturas modernas. Propón escenarios como el de “TechLegacy” para que los equipos identifiquen los problemas de escalabilidad, mantenibilidad y actualización, y propongan soluciones arquitectónicas adecuadas, justificando sus decisiones.
- Promueve ejercicios de modelado utilizando diagramas UML (clases, componentes, secuencia) para representar la estructura y el comportamiento de sistemas complejos. Utiliza plataformas colaborativas como Lucidchart, draw.io o StarUML para que los estudiantes trabajen en equipos y documenten sus modelos, facilitando la discusión y retroalimentación entre pares.
- Propicia la comparación entre ventajas y desventajas del diseño y modelado de software. Solicita que los equipos elaboren listas colaborativas o mapas conceptuales donde identifiquen cómo el modelado

contribuye a la claridad, detección temprana de errores y planificación eficiente, así como los riesgos de sobreingeniería o rigidez excesiva.
- Incentiva la descomposición de sistemas en componentes o módulos. Asigna ejercicios donde los estudiantes tomen un sistema (por ejemplo, una biblioteca o una tienda en línea) y lo dividan en funciones, subfunciones y componentes, documentando el proceso y justificando la asignación de responsabilidades.
- Recomienda la simulación de arquitecturas de distribución. Pide a los estudiantes que diseñen la arquitectura distribuida de un sistema, identificando los componentes que se ubicarían en diferentes servidores o ubicaciones geográficas, y definan los mecanismos de comunicación entre ellos. Sugiere el uso de diagramas de despliegue para visualizar la distribución y los flujos de datos.
- Fomenta el análisis comparativo de modelos arquitectónicos (por ejemplo, arquitectura en capas, microservicios, cliente-servidor). Organiza debates o presentaciones donde los equipos expongan las características, ventajas y limitaciones de cada modelo, y propongan cuál sería más adecuado para distintos tipos de aplicaciones (banca en línea, redes sociales, comercio electrónico, etc.).
- Propicia la reflexión sobre la importancia de la arquitectura en la escalabilidad, flexibilidad y seguridad de los sistemas. Utiliza preguntas detonadoras como: ¿Cómo puede una buena arquitectura prevenir problemas futuros? y ¿qué impacto tiene la descomposición de sistemas en la eficiencia del desarrollo y mantenimiento?.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 9:

Al profesor impartidor, se le recomienda lo siguiente:

- Plantea este tema como una experiencia evolutiva de la arquitectura de software. Haz énfasis en lo que cambia al desarrollar “para la nube” y cómo las decisiones técnicas impactan el negocio.
- Diseña un caso práctico guiado, por ejemplo, migrar una app monolítica a contenedores:
    - Muestra cómo convertir una aplicación básica a microservicios usando Docker.
    - Integra Copilot para sugerencias de Dockerfiles, archivos docker-compose, e incluso scripts de CI/CD.
- Simula una situación de alta demanda:
    - Usa una app web simple y simula picos de tráfico.
- Discute cómo respondería una arquitectura cloud escalable vs. on-premise.
- Explica Terraform o Bicep asistidos por Copilot:
    - Diseña una práctica de despliegue en Azure con infraestructura como código.
    - Observa cómo Copilot puede ayudar a escribir código declarativo en archivos .bicep o .tf.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 10:

Al profesor impartidor, se le recomienda lo siguiente:

- Presenta la definición de arquitectura de microservicios como un sistema fragmentado en componentes pequeños, autónomos, con comunicación a través de API.
- Compara la arquitectura monolítica, SOA y microservicios: diferencias clave en flexibilidad, escalabilidad y mantenimiento.
- Explica las ventajas de los microservicios como la escalabilidad individual, despliegue independiente, resiliencia ante fallos, modularidad.
- Aborda los desafíos como la complejidad en la comunicación, autenticación, consistencia de datos, pruebas y gestión de múltiples servicios.
- Explica las características de un sistema de microservicios robusto:
    - Servicios RESTful definición, principios (cliente-servidor, sin estado, caché, capas, URI, JSON/XML).
    - Implementación de un servicio RESTful con el uso de tecnologías (Node.js, Express, Python, Django, Docker, API Gateway, AWS Cloud Map).

- Presenta un caso de análisis en equipos presentado casos de una migración de monolito a microservicios, pedir a cada equipo que diseñe una propuesta de arquitectura de microservicios identificando los módulos, tecnologías, estrategias de despliegue y pruebas.
- Simula un API RESTful diseñado en equipos un servicio para un sistema de inventario definir los endpoints, definir métodos HTTP, URI, formato de datos y seguridad básica.
- Aborda las herramientas clave como Docker, Kubernetes, AWS Cloud Map, API Gateway, CloudWatch, GitLab CI/CD, Logstash.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 11:

Al profesor impartidor, se le recomienda lo siguiente:

- Organiza actividades donde los estudiantes analicen casos reales de brechas de seguridad (como Equifax o Facebook) para identificar el impacto de las vulnerabilidades en la reputación, confianza y viabilidad de las organizaciones. Propón debates sobre cómo una mala gestión de seguridad puede afectar tanto a usuarios como a empresas, y pide a los equipos que propongan estrategias de mitigación basadas en los ejemplos analizados.
- Promueve ejercicios prácticos de identificación y defensa contra ataques comunes como inyección SQL, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) y ataques de fuerza bruta. Utiliza simuladores o entornos de laboratorio donde los estudiantes puedan experimentar con ejemplos controlados y aplicar defensas como consultas preparadas, sanitización de entradas, uso de tokens CSRF y políticas de bloqueo tras intentos fallidos.
- Propicia la diferenciación clara entre autenticación y autorización. Solicita que los estudiantes implementen mecanismos de autenticación (usuarios, contraseñas, tokens, MFA) y autorización (roles, permisos granulares) en pequeños proyectos, utilizando frameworks o librerías actuales. Fomenta la discusión sobre modelos RBAC y ABAC, y su aplicación en escenarios como plataformas educativas o bancarias.
- Incentiva la aplicación de buenas prácticas de seguridad en el ciclo de vida del software: desde el diseño seguro, la codificación defensiva y la gestión de sesiones, hasta la implementación de políticas de menor privilegio y la expiración de sesiones inactivas. Pide a los estudiantes que documenten y justifiquen cada medida aplicada en sus proyectos.
- Recomienda ejercicios de cifrado y privacidad. Asigna tareas donde los estudiantes cifren información tanto en tránsito (usando HTTPS/TLS) como en reposo (bases de datos, archivos), implementando algoritmos como AES o RSA. Propón reflexiones sobre la importancia de la privacidad, la minimización de datos y el cumplimiento normativo (GDPR, HIPAA), así como la gestión segura de claves y la transparencia con los usuarios.
- Fomenta la reflexión sobre el equilibrio entre seguridad y usabilidad. Utiliza preguntas detonadoras como: ¿Qué consecuencias puede tener no aplicar mecanismos adecuados de autenticación y autorización?, ¿cómo cambia tu estrategia de desarrollo si tu aplicación almacena datos sensibles? y ¿qué medidas aplicarías para defender una API pública de ataques de inyección o fuerza bruta?

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 12:

Al profesor impartidor, se le recomienda lo siguiente:

- Organiza actividades donde los estudiantes analicen casos reales de fallos de software y sus consecuencias, como el colapso de aplicaciones bancarias o sistemas críticos. Propón debates sobre cómo la falta de pruebas adecuadas, una arquitectura deficiente o la ausencia de métricas de calidad pueden afectar la reputación, seguridad y viabilidad de una organización. Invita a los equipos a proponer acciones preventivas y correctivas fundamentadas en modelos de calidad reconocidos.
- Promueve ejercicios de identificación y comparación de los principales modelos de calidad de software (ISO/IEC 25010, CMMI, McCall). Asigna a los equipos el análisis de un sistema o aplicación y solicita que

evalúen su calidad utilizando al menos dos modelos distintos, justificando cuál consideran más adecuado para ese contexto y por qué.

- Propicia la aplicación práctica de métricas de calidad. Solicita a los estudiantes que seleccionen un módulo de software (propio o de código abierto) y calculen métricas clave como complejidad ciclomática, líneas de código, cobertura de pruebas, tasa de defectos y mantenibilidad. Pide que interpreten los resultados y propongan mejoras, fomentando la reflexión sobre el impacto de estas métricas en la calidad global del software.
- Incentiva la elaboración de presentaciones colaborativas donde los equipos expliquen los atributos de calidad definidos por la norma ISO/IEC 25010 (funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad, seguridad y calidad en uso), ilustrando cada atributo con ejemplos prácticos y casos de estudio.
- Recomienda el uso de herramientas de análisis estático y dinámico de código (como SonarQube, CodeClimate o PyLint) para automatizar la medición de métricas de calidad. Sugiere que los estudiantes documenten el proceso y los hallazgos, y discutan en foros virtuales la importancia de la medición continua y la mejora basada en datos objetivos.
- Fomenta la reflexión sobre la integración de la calidad en todas las etapas del ciclo de vida del software, desde el levantamiento de requerimientos hasta el mantenimiento. Utiliza preguntas detonadoras como: ¿Por qué invertir en calidad reduce costos a largo plazo? o ¿cómo influyen las métricas en la toma de decisiones técnicas y de negocio?, para guiar discusiones y foros en línea.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 13:

Al profesor impartidor, se le recomienda lo siguiente:

- Este tema debe impartirse con un enfoque de práctica continua. Fomenta una cultura de calidad en la que la validación está integrada desde el inicio.
- Recomendaciones para la clase:
    - Comienza con una prueba fallida → implementa → refactoriza. Que los estudiantes lo vivan.
    - Muestra cómo Copilot puede sugerir funciones directamente desde la redacción de la prueba.
    - Presenta un sistema mal probado (como el caso de HealthCare.gov).
    - Pide a los estudiantes que propongan un plan de pruebas correctivo (unitarias, integración, sistema, aceptación).
    - Integra OWASP ZAP o Burp Suite en prácticas reales.
    - Usa Copilot para generar scripts de pruebas de seguridad (por ejemplo, XSS o SQL injection simples).

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 14:

Al profesor impartidor, se le recomienda lo siguiente:

- Comienza la lección contrastando dos casos prácticos de la industria: Netflix y Knight Capital. Estos ejemplos ilustran cómo la adopción o la ausencia de prácticas de DevOps y gestión de código impactaron significativamente sus negocios. Netflix, con pipelines automatizados y una cultura colaborativa, logró un despliegue eficiente y resiliente. Por otro lado, Knight Capital sufrió una pérdida masiva debido a la falta de integración continua y pruebas automatizadas. Utiliza estos relatos para enfatizar la importancia de la integración continua (CI), la entrega continua (CD), y la orquestación de contenedores en la ingeniería de software moderna.
- Explica qué es CI/CD y su impacto en el ciclo de vida del desarrollo de software.
- Resalta cómo CI implica integrar y validar cambios de código continuamente, mientras que CD abarca la automatización del despliegue en entornos de prueba o producción.
- Analiza los casos de Netflix y Knight Capital para mostrar el contraste de prácticas exitosas y fallidas.

- Discute los principios históricos desde los métodos artesanales hasta la modernización con CI/CD.
- Enfatiza en pipelines pequeños y frecuentes, validaciones de seguridad, pruebas automatizadas, y el uso de métricas clave.
- Menciona herramientas populares de CI/CD y ejemplos de implementación.
- Define qué son los contenedores y cómo difieren de las máquinas virtuales.
- Explica la importancia de la portabilidad y eficiencia de recursos.
- Ofrece recomendaciones para construir imágenes ligeras, un proceso por contenedor, y gestión segura de secretos.
- Detalla la diferencia entre logs estructurados y no estructurados.
- Discute herramientas y ejemplos de monitoreo de métricas y trazabilidad distribuida.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 15:

Al profesor impartidor, se le recomienda lo siguiente:

- Explica el despliegue de software, qué es, actividades principales (lanzamiento, instalación, actualización, etc.) y estrategias.
- Muestra las herramientas de automatización para despliegue con sus ventajas y desventajas.
- Simula un despliegue en equipo de estudiantes para simular un despliegue de software, asignando roles (DevOps, QA, usuario final) y resolviendo problemas hipotéticos (por ejemplo, error en producción o falla de compatibilidad).
- Analiza casos reales y solicita a los estudiantes proponer un plan de mantenimiento adecuado.
- Propicia un debate reflexivo sobre Hasta qué punto innovar puede afectar la estabilidad de un software.
- Presenta un caso de mantenimiento adaptativo frente a una actualización crítica de sistema.
- Explica un ejemplo de uso de Git simulando la creación de ramas, fusión de cambios y reversión en un repositorio sencillo.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 16:

Al profesor impartidor, se le recomienda lo siguiente:

- Explica los modelos de desarrollo de software (Tradicionales (cascada, iterativo/incremental, espiral) y ágiles (Scrum, Kanban, XP)) y sus diferencias, ventajas, desventajas y aplicaciones.
- Aborda los factores para elegir un modelo de desarrollo bajo la claridad de requisitos, complejidad, tamaño del proyecto, participación del cliente, presupuesto y plazos.
- Presenta herramientas para la gestión del cambio mediante sistemas de control de versiones (Git, TFS, SVN), gestión de tickets (Jira, Trello, Azure DevOps).
- Muestra el modelo CMMI identificando los niveles de madurez (1 a 5), áreas de proceso (gestión de procesos, proyectos, ingeniería, soporte) y beneficios de su implementación.
- Compara de modelos de desarrollo en grupos pequeños analizan un caso real (ejemplo un startup de aplicaciones móviles) y discutan qué modelo sería más adecuado, justificando su decisión.
- Simula la gestión de cambios al asignar un cambio ficticio en un proyecto (ejemplo una nueva funcionalidad) y pedir a los estudiantes seguir el proceso de gestión de cambios, documentando cada paso.
- Expón un caso práctico de CMMI como ejemplo de empresa que alcanza el nivel 5 de madurez para cumplir requisitos de clientes grandes.
- Explica aplicación ágil de un desarrollo de una aplicación en un entorno de startup usando Scrum, donde los requisitos cambian constantemente.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 17:

Al profesor impartidor, se le recomienda lo siguiente:

- Explica la definición clara de proyectos (Objetivos SMART, alcance, restricciones, identificación de riesgos y stakeholders).
- Destaca la importancia de la gestión proactiva revisando y actualizar el cronograma, gestionar retrasos y ajustes.
- Analiza un caso de proyecto real presentando un escenario en grupo en equipos y solicitar:
    - Definir objetivos, alcance, restricciones, riesgos y stakeholders.
    - Diseñar una EDT básica.
    - Elaborar una matriz RACI para algunas actividades clave.
    - Crear un cronograma simplificado en Gantt.
- Simula la estimación (mediante un análisis hipotético).
- Propicia un debate sobre qué retos han enfrentado en proyectos escolares o personales al estimar tiempos y recursos.
- Presenta un ejemplo de EDT de un proyecto de tienda en línea, con desglose en diseño, desarrollo frontend/backend, pruebas y despliegue.
- Solicita el uso de diagramas Gantt/Pert mostrando ejemplos de cronogramas simples y cómo representan las relaciones y dependencias.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 18:

Al profesor impartidor, se le recomienda lo siguiente:

- Explica las técnicas de estimación de costos con juicio de expertos, analogía, paramétrica, tres puntos.
- Presenta la ejecución y control del proyecto con diagrama de Gantt, ruta crítica (CPM), PERT, indicadores (SPI, CPI), monitoreo de calidad y gestión de cambios.
- Demuestra la gestión de riesgos, identificación, clasificación (internos, externos, técnicos, financieros, legales, conocidos, desconocidos), análisis cualitativo y cuantitativo, planificación de respuestas (evitar, mitigar, transferir, aceptar).
- Presenta un caso simulando un proyecto identificando los costos, riesgos y planes de mitigación.
- Haz un breve taller de estimación aplicando la técnica de tres puntos para un proyecto (optimista, probable, pesimista) y calcular costos esperados.
- Solicita un diseño de matriz de riesgos analizando posibles amenazas de un proyecto tecnológico y clasificarlas en una matriz de probabilidad e impacto.
- Simula un conflicto de proyecto y aplicar estilos de liderazgo y técnicas de negociación, así como el cierre de proyecto, incluyendo checklist de tareas, auditoría, lecciones aprendidas y generación del informe final.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 19:

Al profesor impartidor, se le recomienda lo siguiente:

- Este tema debe abordarse como una transformación de paradigma en la práctica del desarrollo de software. El docente debe guiar a los estudiantes no solo en el uso de herramientas, sino en la comprensión crítica de sus implicaciones técnicas, organizacionales y éticas.
- Recomendaciones para la clase
    - Abre Visual Studio Code y muestra cómo escribir un comentario y dejar que Copilot genere código completo.
    - Pide al grupo observar y analizar si la propuesta sigue buenas prácticas.
    - Propón mini-retos con IA, por ejemplo: “Implementar una función de validación de formularios con sugerencias de Copilot”.

- Haz que el estudiante justifique cuándo y por qué acepta o modifica una sugerencia.
- Asigna a un grupo que programe sin asistencia y otro con IA. Luego, comparen velocidad, calidad y comprensión del código.
- Propicia la discusión ética y crítica a través de las siguientes preguntas:
  - ¿Debe usarse Copilot para tareas de seguridad?
  - ¿Qué ocurre si sugiere código que tiene vulnerabilidades?

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 20:

Al profesor impartidor, se le recomienda lo siguiente:

- Presenta las principales tecnologías emergentes:
  - Internet de las cosas (IoT) con uso de sensores, conectividad, procesamiento, interfaces; aplicaciones en hogares inteligentes, salud, agricultura, ciudades inteligentes.
  - Inteligencia Artificial (IA) con Machine learning, redes neuronales, PNL, visión artificial; impactos en industria, salud, educación.
  - Robótica avanzada con sensores, actuadores, inteligencia; aplicaciones en industria, medicina, exploración.
  - Big Data y su infraestructura, análisis, visualización; retos de almacenamiento, procesamiento y seguridad.
  - Computación en la nube: IaaS, PaaS, SaaS; seguridad, accesibilidad, escalabilidad.
  - Computación cuántica y Qubits, superposición, entrelazamiento; retos y potencial para resolver problemas complejos.
- Destaca los retos clave como la ética, privacidad, seguridad de datos, impacto social, habilidades futuras.
- Menciona las habilidades para el futuro aprendizaje continuo, conocimiento en IA, ciberseguridad, manejo de datos, programación cuántica.
- Propicia un debate sobre el impacto social y ético tiene el uso masivo de IA y cómo proteger la privacidad en un entorno dominado por IoT.
- Solicita una reflexión sobre las habilidades necesarias para adaptarme a los cambios de la cuarta revolución industrial.
- Presenta las aplicaciones cotidianas de IA y IoT (asistentes virtuales, domótica, autos autónomos, monitoreo de salud).
- Ejemplifica casos de uso de robótica avanzada en cirugías asistidas, prótesis inteligentes, exploración en ambientes hostiles.

Notas para el profesor impartidor. Estas corresponden a la explicación de la actividad 1:

- Presenta los conceptos fundamentales de CI/CD.
- Explica la importancia de la integración continua y la entrega continua utilizando los ejemplos de Netflix y Knight Capital para ilustrar el impacto positivo y negativo de su implementación.
- Instruye a los estudiantes para que configuren un pipeline de CI utilizando una herramienta como Jenkins, GitLab CI o CircleCI.
- Incluye pasos para la fusión de cambios de código, ejecución de pruebas unitarias, y validaciones básicas.
- Revisa y discute los resultados de las actividades prácticas, enfocándote en los desafíos encontrados y las soluciones aplicadas.
- Fomenta que los estudiantes compartan sus experiencias y aprendizajes.

Notas para el profesor impartidor. Estas corresponden a la explicación de la actividad 2:

- Organiza debates comparativos entre metodologías ágiles (Scrum, XP) y tradicionales (Cascada, Incremental). Propón casos concretos (ejemplo: startups vs. proyectos gubernamentales) para que los estudiantes analicen ventajas, desventajas y contextos de aplicación. Sugiere el uso de tablas comparativas y discusiones grupales para sintetizar aprendizajes.
- Facilita talleres prácticos con herramientas ágiles (Trello, Jira, Asana) para simular la planificación de un Sprint. Guía a los estudiantes en la creación de tableros Kanban, priorización de backlog y asignación de tareas, incentivando la captura de evidencias (screenshots) que reflejen su comprensión del flujo de trabajo ágil.
- Orienta la creación del Product Vision Board mediante ejemplos reales (ejemplo: plataformas como Trello o Notion). Proporciona plantillas y realiza sesiones de retroalimentación para asegurar que los estudiantes definan claramente público objetivo, métricas de éxito y propuesta de valor, vinculándolos al caso de PlanificaTech.
- Simula escenarios de levantamiento de requerimientos con roles asignados (ejemplo: "cliente", "equipo ágil"). Diseña ejercicios donde los estudiantes practiquen entrevistas con preguntas abiertas y elaboren historias de usuario bien estructuradas. Revisa que diferencien requerimientos funcionales (ejemplo: "asignar tareas") y no funcionales (ejemplo: "tiempo de respuesta <2 segundos").
- Promueve la reflexión crítica sobre la diferencia entre proyecto y producto de software. Utiliza preguntas detonadoras como: ¿Cómo cambia la gestión si PlanificaTech quiere escalar su plataforma a otros mercados? o ¿qué métricas indicarían el éxito continuo del producto?.
- Incorpora sesiones de peer review donde los estudiantes presenten sus entregables (historias de usuario, visión de producto) y reciban feedback constructivo. Fomenta el uso de foros o videollamadas para discusiones sincrónicas, enfatizando la mejora iterativa típica de los enfoques ágiles.

Notas para el profesor impartidor. Estas corresponden a la explicación de la actividad 3:

- Facilita talleres prácticos sobre OWASP Top 10 mediante ejemplos reales de aplicaciones web vulnerables (ejemplo: plataformas de gestión de usuarios con fallos de inyección SQL o XSS). Guía a los estudiantes en la simulación de ataques básicos (usando herramientas como OWASP ZAP o Burp Suite en entornos controlados) y en la elaboración de estrategias de mitigación (ejemplo: sanitización de inputs, uso de prepared statements).
- Promueve el análisis crítico del modelo ISO/IEC 25010 asignando casos de estudio donde los estudiantes relacionen características de calidad (como usabilidad, seguridad y mantenibilidad) con problemas comunes en desarrollo (ejemplo: baja escalabilidad, vulnerabilidades). Sugiere comparar aplicaciones existentes (como Trello vs. Jira) para identificar diferencias en la implementación de estas características.
- Diseña ejercicios de métricas de calidad con herramientas como SonarQube o ESLint. Pide a los estudiantes que analicen fragmentos de código predefinidos (con errores deliberados) para calcular métricas como cobertura de pruebas o complejidad ciclomática, y que propongan planes de mejora basados en los resultados.
- Organiza la creación de protocolos de pruebas mediante roles simulados (ejemplo: tester, desarrollador, auditor). Proporciona plantillas para documentar casos de prueba funcionales (ejemplo: "Validar inicio de sesión con credenciales incorrectas") y de seguridad (ejemplo: "Pruebas de fuzzing en campos de formulario"). Incentiva el uso de herramientas como Postman (para API) o Selenium (para UI).
- Fomenta debates sobre "shift-left" (integración temprana de seguridad y calidad) con preguntas como: ¿Cómo evitar costosos reworks si se detecta una vulnerabilidad en fase de producción? o ¿qué métricas priorizarías en un proyecto con plazos ajustados? Relaciona estas discusiones con la reflexión final de la actividad.
- Incorpora revisiones por pares de los protocolos de pruebas, donde los estudiantes evalúen la exhaustividad y realismos de los casos propuestos por sus compañeros. Utiliza rúbricas claras para calificar la profundidad del análisis de vulnerabilidades y la viabilidad de las estrategias de mitigación.

Notas para el profesor impartidor. Estas corresponden a la explicación de la actividad 4:

- Explica la importancia de la medición en DevOps.
- Presenta las métricas clave del equipo de investigación DORA y el marco SPACE.
- Utiliza ejemplos prácticos para ilustrar cómo Netflix y otras empresas líderes utilizan estas métricas para mejorar continuamente.
- Introduce herramientas como Prometheus para la recopilación y Grafana para la visualización.
- Fomenta que los estudiantes comparen sus datos y discutan posibles cuellos de botella y puntos de mejora.
- Guía a los estudiantes a identificar patrones y discutir planes de acción para optimizar sus pipelines y procesos DevOps.

Notas para el profesor impartidor. Estas corresponden a la explicación de la actividad 5:

- Recuerda al aprendedor que debe comenzar a realizar los cursos desde el primer día para optimizar tiempos.
- Discute los principios básicos de la gestión de proyectos y su relevancia en el mundo profesional actual.
- Diseña un plan de proyecto detallado utilizando herramientas como Microsoft Project. Incluye los objetivos, alcance, cronograma y recursos necesarios.
- Crea un seguimiento y reporte del avance del proyecto. Utiliza dashboards y Gantt charts para visualizar el progreso.
- Desarrolla un informe de cierre de proyecto que incluya lecciones aprendidas, análisis de rendimiento y recomendaciones para futuros proyectos.
- Revisa los proyectos desarrollados por los estudiantes, discute los desafíos enfrentados y proporciona retroalimentación constructiva.
- Explica los principios de Agile, Scrum y Kanban. Discute sus diferencias y aplicaciones prácticas.
- Utiliza herramientas como Jira para diseñar y priorizar un backlog de producto, incluyendo historias de usuario y criterios de aceptación.
- Planifica y ejecuta sprints en un entorno ágil, utilizando herramientas de seguimiento y retrospectivas.
- Desarrolla un incremento de producto y presenta los resultados al equipo. Incluye revisiones y ajustes basados en retroalimentación.
- Revisa los resultados de los sprints, discute los desafíos y proporciona retroalimentación constructiva para la mejora continua.
- Presenta GitHub Copilot, sus características y cómo puede aumentar la productividad del desarrollo de software.
- Configura GitHub Copilot en un entorno de desarrollo integrado (IDE) y practica con ejemplos sencillos.
- Utiliza GitHub Copilot para desarrollar un pequeño proyecto, aprovechando su capacidad para sugerir y autocompletar código.
- Revisa el proyecto desarrollado por los estudiantes, discutiendo las sugerencias de Copilot y proporcionando retroalimentación sobre el uso y optimización del código.

Notas para el profesor impartidor. Estas corresponden a la explicación del proyecto (fase I):

- Guía la elaboración del acta de constitución mediante ejemplos reales de proyectos sociales (ejemplo: plataformas como Banco de Alimentos). Asegúrate que los estudiantes definan claramente el impacto social (métricas cuantificables) y alineen el alcance con los ODS (Objetivos de Desarrollo Sostenible). Proporcione plantillas estandarizadas para este documento.
- Facilita talleres de identificación de stakeholders con dinámicas de roles. Pide a los equipos que simulen entrevistas con actores clave (ejemplo: representantes de ONG) para priorizar necesidades en la matriz de poder/interés. Usa herramientas colaborativas como Miro para construir la matriz visualmente.

- Orienta el diseño de arquitectura con casos comparativos (monolitos vs. microservicios en proyectos sociales). Sugiere evaluar factores como costos de implementación y escalabilidad. Recomienda herramientas como Draw.io para diagramas y exija justificaciones técnicas basadas en los requisitos identificados.
- Promueve la planificación ágil/híbrida mediante simulaciones con herramientas como Jira. Asigna a cada equipo un escenario distinto (ejemplo: "startup con recursos limitados" vs. "gobierno local con plazos fijos") para que adapten su metodología y cronograma. Revisa que incluyan buffers para riesgos críticos.
- Diseña ejercicios de priorización donde los estudiantes clasifiquen historias de usuario con técnicas como MoSCoW (Must-have, Should-have, Could-have). Fomenta debates sobre trade-offs (ejemplo: "¿Invertir en geolocalización o en notificaciones push primero?").

Notas para el profesor impartidor. Estas corresponden a la explicación del proyecto (fase II):

- Guía la implementación segura mediante talleres prácticos con JWT (ejemplo: diferenciar permisos de admin/usuario en endpoints clave) y pruebas unitarias (usar Jest/Pytest con ejemplos de casos límite). Proporcione repositorios base con vulnerabilidades intencionales para que corrijan.
- Supervisa la creación del pipeline CI/CD con GitHub Actions/GitLab, enfatizando buenas prácticas como:
    - Automatización de pruebas en cada commit.
    - Despliegue en entornos stage antes de producción.
    - Uso de secrets management para credenciales.
- Facilita auditorías de seguridad con OWASP ZAP, asignando vulnerabilidades específicas a buscar (XSS, SQLi). Compare resultados entre equipos y discuta mitigaciones (ejemplo: sanitización de inputs, CSP headers).
- Promueve el análisis de calidad con SonarQube, pidiendo que documenten:
    - Deuda técnica crítica por resolver.
    - "Code smells" recurrentes en su implementación.
    - Comparativas de métricas pre/post refactorización.
- Orienta el informe de cierre con plantillas que incluyan:
    - Gráficos de desviaciones (tiempos, funcionalidades).
    - Lecciones técnicas y de gestión.
    - Propuestas de mejora con fundamentos (ejemplo: IA para predicción basada en datos históricos).
- Organiza una demo day donde los equipos presenten su pipeline funcionando y expliquen cómo resolverían un escenario de fallo en producción (rollback, hotfixes), evaluando su capacidad de respuesta ante incidentes.

Anexo 1. Rúbrica del avance del proyecto (fase I)

	Nivel de desempeño			
Criterios de evaluación	Altamente competente 100%-86%	Competente 85%-70%	Aún sin desarrollar la competencia 69%-0%	%
1. Definición del proyecto y análisis de *stakeholders*	20 – 18 puntos Crea un acta de constitución completa con objetivos, alcance y justificación claros. Desarrolla una matriz de al menos cuatro stakeholders con un análisis preciso de poder contra interés.	17 – 15 puntos Crea un acta de constitución con la mayoría de los elementos requeridos. Desarrolla una matriz de tres stakeholders con omisiones o imprecisiones menores.	14 – 0 puntos Crea un acta de constitución incompleta que carece de elementos clave. El análisis de los stakeholders es superficial u omite actores importantes.	20
2. Ingeniería de requerimientos	25 – 23 puntos Entrega al menos cinco historias de usuario bien redactadas siguiendo el formato correcto, y al menos tres requisitos no funcionales claramente definidos y alineados con las necesidades del proyecto.	22 – 20 puntos Entrega entre tres y cuatro historias de usuario con problemas menores de formato. Asimismo, define entre dos y tres requisitos no funcionales con algunos problemas de claridad.	19 – 0 puntos Entrega menos de tres historias de usuario con problemas mayores de formato, o menos de dos requisitos no funcionales.	25
3. Diseño arquitectónico y selección de metodología	25 – 23 puntos Crea un diagrama de arquitectura detallado con todos los componentes e interfaces claramente identificados. La selección de metodología está bien justificada con consideraciones específicas del proyecto.	22 – 20 puntos Crea un diagrama de arquitectura con la mayoría de los componentes identificados. La selección de metodología está justificada, pero carece de algunas consideraciones específicas del proyecto.	19 – 0 puntos Crea un diagrama de arquitectura incompleto con componentes clave faltantes. La selección de metodología carece de justificación.	25
4. Gestión de riesgos	15 – 13 puntos Desarrolla una matriz de riesgos exhaustiva con al menos cinco riesgos, incluyendo probabilidad, impacto y estrategias de mitigación para cada uno.	12 – 10 puntos Desarrolla una matriz de riesgos con tres o cuatro riesgos, con algunas brechas en probabilidad, impacto o estrategias de mitigación.	9 – 0 puntos Desarrolla una matriz de riesgos incompleta con menos de tres riesgos o falta de elementos clave.	15
5. Cronograma del proyecto	15 – 13 puntos Crea un diagrama de Gantt detallado con todas las fases del proyecto, tareas, dependencias e hitos claramente identificados.	12 – 10 puntos Crea un diagrama de Gantt con la mayoría de las fases y tareas identificadas, pero con algunas dependencias o hitos faltantes.	9 – 0 puntos Crea un diagrama de Gantt incompleto que carece de componentes esenciales o coherencia en la línea de tiempo.	15
			Total	**100 %**

Anexo 2. Rúbrica de la entrega final del proyecto (fase II)

	Nivel de desempeño			
Criterios de evaluación	Altamente competente 100%-86%	Competente 85%-70%	Aún sin desarrollar la competencia 69%-0%	%
1. Implementación del módulo y seguridad	20 – 18 puntos Desarrolla un módulo completamente funcional con un sistema de autenticación completo usando JWT y roles. Implementa pruebas unitarias exhaustivas con cobertura mayor o igual a 80 %.	17 – 15 puntos Desarrolla un módulo funcional con un sistema de autenticación, pero con problemas menores de implementación. Cobertura de pruebas unitarias entre 60-79 %.	14 – 0 puntos Desarrolla un módulo incompleto, con problemas graves de autenticación o roles faltantes. Cobertura de pruebas unitarias por debajo del 60 %.	20
2. Implementación de *pipeline* CI/CD	25 – 23 puntos Implementa un pipeline CI/CD completo con etapas automatizadas de pruebas, construcción y despliegue que funcionan correctamente.	22 – 20 puntos Implementa un pipeline CI/CD con la mayoría de las etapas requeridas, pero con problemas menores de configuración.	19 – 0 puntos Implementa un pipeline CI/CD incompleto que carece de etapas esenciales o con problemas graves de configuración.	25
3. Pruebas de seguridad y análisis de calidad de código	25 – 23 puntos Realiza pruebas de seguridad exhaustivas con identificación y corrección de vulnerabilidades. Análisis completo de calidad de código con todas las métricas documentadas.	22 – 20 puntos Realiza pruebas de seguridad con alguna identificación de vulnerabilidades. Análisis de calidad de código con algunas de las métricas documentadas.	19 – 0 puntos No realiza pruebas de seguridad. El análisis de calidad de código está incompleto.	25
4. Cierre del proyecto y análisis	15 – 13 puntos Crea un informe de cierre exhaustivo con comparación detallada de líneas de tiempo planificadas vs. reales y lecciones aprendidas completas.	12 – 10 puntos Crea un informe de cierre con la mayoría de los elementos requeridos, pero con falta la comparación o las lecciones aprendidas.	9 – 0 puntos Crea un informe de cierre incompleto que carece de componentes esenciales o análisis.	15
5. Plan de mejora continua	15 – 13 puntos Desarrolla un plan de mejora detallado con acciones específicas y medibles, asimismo genera propuestas de innovación.	12 – 10 puntos Desarrolla un plan de mejora con algunas acciones, pero carece de criterios de medición o elementos de innovación.	9 – 0 puntos Desarrolla un plan de mejora incompleto que carece de acciones específicas o relevancia para el proyecto.	15
			Total	**100 %**

Anexo 3. Prácticas de bienestar

Práctica 01

Nombre de la práctica	Un momento para respirar.
Descripción de la práctica	Aprender a respirar por la nariz y a tranquilizar tu mente.
Palabras clave	Fortalezas de carácter, autorregulación.
Instrucciones para el aprendedor	La autorregulación, también percibida como control, es una fortaleza de carácter muy importante dentro de la psicología positiva. Este concepto implica regular lo que uno siente y hace, ser disciplinado, así como mantener un control sobre los apetitos y, especialmente, sobre las emociones. En la actualidad vivimos situaciones muy estresantes que provocan que nuestra reacción instintiva y natural ante ellas sea estallar en ira. Pero, las consecuencias de este comportamiento no solo se quedan en nosotros, sino que también pueden llegar a afectar a terceros. A continuación, se presenta un ejercicio que te ayudará a cultivar la fortaleza de autorregulación: 1. Toma 2 minutos de tu tiempo, siéntate en un lugar cómodo, donde no haya mucho ruido que te pueda distraer. 2. Escucha música de relajación (crea tu propio ambiente de meditación). 3. Comienza a respirar y exhalar por nariz. 4. Trata de que tu respiración y exhalación dure el mismo tiempo. 5. Fija tu mente en tu respiración, en cómo entra y sale el aire de tu cuerpo. Así durante 2 minutos. Te recomendamos que si durante este periodo algún pensamiento (olvidé algo en la oficina, más tarde tengo que hacer tal actividad, etc.) llega a tu mente, solo déjalo pasar y regresa a la concentración en tu respiración. Al finalizar los dos minutos sentirás paz en tu ser. Comienza a hacer este ejercicio de respiración y meditación todos los días y poco a poco vas aumentando los minutos de este.
Fuente	Conferencia Rosalinda Ballesteros.

Práctica 02

Nombre de la práctica	Fomentando la atención plena.
Descripción de la práctica	Llevarás a cabo breves ejercicios de meditación para fomentar la atención plena en tus actividades diarias.
Palabras clave	Atención plena, fortalezas de carácter, autorregulación.
Instrucciones para el aprendedor	La meditación es una herramienta que ayuda a mejorar el desempeño de cualquier persona, ya que fomenta el desarrollo de la atención plena en una sola actividad. Para fomentar la atención plena y lograr cada vez más estar en una zona de concentración mientras realizas tus actividades cotidianas, puedes llevar a cabo los siguientes ejercicios de meditación: Encuentra en algún momento del día cinco minutos para ti, siéntate en un lugar cómodo, donde no tengas distracciones. 1. Haz tres respiraciones profundas, inhala y exhala por la nariz. 2. Comienza a hacer un repaso de tu día, de lo que más te acuerdes, por ejemplo, te levantaste, ¿qué hiciste?, ¿desayunaste?, ¿te bañaste?, ¿diste los buenos días?, etcétera. Si desayunaste, ¿qué fue lo que desayunaste?, ¿te gustó?, ¿tomaste tu alimento despacio o apurado? Si estabas apurado, ¿qué era lo que te tenía en esa situación? 3. Sigue meditando en lo que te acuerdes: ¿te molestase con alguien?, ¿por qué?, ¿qué fue lo que pasó?, ¿crees que era posible haber reaccionado de alguna manera más pacífica? Con este ejercicio te darás cuenta de que reaccionamos o hacemos cosas de manera automática. Algunas veces si estamos más conscientes y presentes, podemos tener otra actitud sin que alguna situación nos afecte demasiado.
Fuente	Eby, D. (s.f.). *Creativity and Flow Psychology*. Recuperado de http://talentdevelop.com/articles/Page8.html

Práctica 03

Nombre de la práctica	Experiencias difíciles.
Descripción de la práctica	En esta práctica podrás analizar las estrategias que seguiste para afrontar problemáticas y cómo aprendiste de tales sucesos.
Palabras clave	Resiliencia.
Instrucciones para el aprendedor	Todos hemos pasado por situaciones complejas, no solo en lo laboral, sino también en el ámbito familiar y personal. La manera en que enfrentamos dichos obstáculos es muy diferente, algunas personas continúan con su vida sin problema alguno, a otras tantas se les complica esa transición, también hay quienes no pueden sobreponerse a las experiencias difíciles. La resiliencia es la capacidad de reponerse tras la adversidad, de recuperarse después de vivir experiencias difíciles, dolorosas o traumáticas. Para algunos la resiliencia implica no solo salir adelante después de una situación muy dura, sino incluso crecer o ser mejor a raíz de esta experiencia. (Tarragona, 2012) La siguiente práctica te ayudará a fomentar esta importante cualidad: 1. Crea una tabla con tres columnas y cinco filas. 2. En la primera columna escribe un evento difícil o desagradable al que te hayas enfrentado en tu vida. 3. En la segunda columna menciona cuáles son tus creencias sobre esa adversidad. 4. En la tercera columna describe las consecuencias que tiene esa creencia. 5. Cuando termines, lee toda la tabla y reflexiona sobre cómo te ha cambiado cada evento y cómo lo enfrentaste. 6. Escribe al final cómo enfrentarías cada evento hoy en día.
Fuente	• Metodología ABC. • Fundamentos de psicología positiva.

Práctica 04

Nombre de la práctica	Concentrarse en lo positivo.
Descripción de la práctica	Analizarás sucesos que te hayan ocurrido recientemente, buscando orientar el análisis hacia las consecuencias positivas.
Palabras clave	Resiliencia y esperanza.
Instrucciones para el aprendedor	¿Qué es lo primero que piensas cuando recibes una noticia inesperada?, o bien, ¿qué te imaginas cuando un acontecimiento complejo se presenta ante ti? La mayoría de las personas automáticamente se concentra en el peor de los escenarios independientemente del tipo de noticia que reciban. Martin Seligman sugiere hacer un breve ejercicio para fomentar la resiliencia y la esperanza con base en la premisa antes señalada: 1. Piensa en una noticia reciente que hayas recibido y que creas que es negativa para ti. 2. Luego de analizarla, haz una tabla con tres columnas. En la primera, señala cuál sería el peor de los escenarios posibles que pudieran resultar de esa noticia; en la segunda columna señala cuál sería el mejor de los escenarios posibles, y en la última, cuál es el escenario que realmente tiene mayor probabilidad de ocurrir. 3. Reflexiona sobre los tres escenarios, ¿cómo enfrentarías cada uno de ellos? Procura repetir este ejercicio cada vez que sientas que te enfrentas a una situación complicada. Hacerlo te dará perspectiva y te ayudará a cultivar tu resiliencia.
Fuente	Seligman, M. (2011). *Building Resilience*. Recuperado de https://hbr.org/2011/04/building-resilience

Práctica 05

Nombre de la práctica	Crecimiento postraumático.
Descripción de la práctica	En esta práctica harás un recuento de las situaciones difíciles a las que te has enfrentado y reflexionarás sobre lo positivo que surgió de ellas.
Palabras clave	Resiliencia.
Instrucciones para el aprendedor	La resiliencia es la capacidad de reponerse tras la adversidad, de recuperarse después de vivir experiencias difíciles, dolorosas o traumáticas. Para algunos la resiliencia implica no solo salir adelante después de una situación muy dura, sino incluso crecer o ser mejor a raíz de esta experiencia. (Tarragona, 2012) La siguiente práctica te ayudará a fomentar esta importante cualidad: 1. Escribe acerca de un momento en el que enfrentaste una adversidad significativa o pérdida. 2. Primero escribe acerca de las puertas que se te cerraron debido a esa adversidad o pérdida, ¿qué perdiste? 3. Después escribe acerca de las puertas que se abrieron al termino o como secuela de esa adversidad o pérdida. 4. ¿Hay nuevas maneras de actuar, pensar o relacionarse que son más probables de suceder ahora?
Fuente	• Ejercicio contribuido por Taylor Kreiss de University of Pennsylvania Positive Psychology Center, y basado en el libro: *A Primer in Positive Psychology* de Christopher Peterson.

Práctica 06

Nombre de la práctica	La mejor versión de ti mismo.
Descripción de la práctica	Escribe acerca de la mejor versión posible de ti mismo durante al menos 20 minutos.
Palabras clave	Emociones positivas, fortalezas de carácter, autorregulación y esperanza.
Instrucciones para el aprendedor	Imagina que dentro de 20 años has crecido en todas las áreas o maneras que te gustaría crecer y las cosas te han salido tan bien como te las imaginaste. • ¿Cómo es esa mejor versión de ti mismo? • ¿Qué hace él o ella cotidianamente? • ¿Qué dicen los demás acerca de él o ella? No es necesario que compartas este escrito, ya que el objetivo de esta reflexión es enfocarse en la experiencia que viviste mientras reflexionabas en esa mejor versión posible de ti mismo.
Fuente	• Ejercicio contribuido por Taylor Kreiss de University of Pennsylvania Positive Psychology Center, y basado en el libro *A Primer in Positive Psychology* de Christopher Peterson.

Práctica 07

Nombre de la práctica	Obtener lo que quieres.
Descripción de la práctica	Reflexionarás sobre alguna meta que desees alcanzar y propondrás una forma de conseguirla.
Palabras clave	Logro, involucramiento, fortalezas de carácter, esperanza, autorregulación, metas y objetivos a largo plazo.
Instrucciones para el aprendedor	Tener una idea clara de lo que deseas lograr a corto, mediano y largo plazo es de suma importancia, pues te ayuda a seguir un camino trazado previamente. Para que puedas generar esta guía, responde las siguientes preguntas: 1. ¿Qué quieres lograr? Al trazar tu meta, procura que esta sea específica, medible, alineada, realista, retadora y con una fecha para lograrla. Piensa en algo y utiliza el método SMART para definirla. 2. ¿Qué te impide que lo tengas en este momento? 3. ¿Qué sufrimiento estás experimentando en tu vida por no tenerlo en este momento? 4. ¿Qué placer, involucramiento, relación, significado o logro tendrías en tu vida si tuvieras eso en este momento? 5. ¿Qué hábitos te detienen o no te dejan avanzar hacia eso que quieres? 6. ¿Qué nuevos hábitos podrías generar para ayudarte a obtener lo que quieres? 7. ¿Qué dos cosas podrías hacer para romper con los hábitos que no te permiten avanzar hacia lo que quieres y generar hábitos nuevos? 8. ¿Te comprometes a hacer esas dos cosas? Si es así, ¿cuándo las harás? Escribe tus resultados en un sitio donde puedas verlos constantemente.
Fuente	• Ejercicio contribuido por Taylor Kreiss de University of Pennsylvania Positive Psychology Center, y basado en el libro *A Primer in Positive Psychology* de Christopher Peterson.

Práctica 08

Nombre de la práctica	Felicidad en el trabajo.
Descripción de la práctica	Reflexionarás sobre las distintas dimensiones de tu vida cotidiana, enfocando el análisis a cómo fomentar un estado de ánimo y relaciones positivos en el ámbito laboral.
Palabras clave	Involucramiento, emociones positivas, relaciones positivas.
Instrucciones para el aprendedor	Elegir conscientemente maneras de incrementar la felicidad en el trabajo puede hacer la diferencia en cómo nosotros nos sentimos y qué tan bien nos desempeñamos. En lugar de quejarnos del trabajo, ¿por qué no pensar en cómo podemos obtener mayor felicidad de lo que hacemos? Estar más involucrados en lo que hacemos contribuye a nuestra felicidad y bienestar, y nos lleva a un mejor desempeño y productividad. A manera de reflexión, responde las siguientes preguntas que están enfocadas en distintas dimensiones de tu vida: • **Dar:** ¿cómo estoy apoyando a mis colaboradores, compañeros, líderes, proveedores y clientes? • **Relaciones:** ¿cómo puedo mejorar mis relaciones en el trabajo?, ¿cómo logro un balance entre la vida laboral y familiar? • **Ejercicio:** ¿cómo puedo integrar la actividad física dentro de mis actividades diarias?, ¿cómo aseguro que estoy comiendo bien y descansando lo suficiente? • **Conciencia:** ¿cómo puedo construir momentos de atención plena en mi día laboral? • **Ensayo:** ¿qué habilidades estoy construyendo?, ¿qué cosas nuevas he experimentado? • **Dirección:** ¿cuáles son mis metas laborales hoy, esta semana, este año?, ¿cómo caben y contribuyen estas con mis metas de vida y me ayudan a desarrollar mis competencias en la construcción de mis relaciones y cómo contribuyo con lo anterior a ayudar a otros?, ¿cómo se pueden alinear mis metas laborales con las de mi equipo y la organización? • **Resiliencia:** ¿cuáles son mis tácticas para lidiar con los retos difíciles en el trabajo?, ¿me estoy enfocando en lo que puedo controlar?, ¿necesito pedir ayuda a otros?, ¿hay alguien a mi alrededor que requiere de mi ayuda? • **Emoción:** ¿qué cosas, aunque sean pequeñas, puedo encontrar que me pueden hacer sentir bien en mi trabajo hoy?, ¿qué me ha hecho sonreír?
Fuente	Tomado del Catálogo de actividades para profesores.

Práctica 09

Nombre de la práctica	Interacciones positivas.
Descripción de la práctica	Reflexionarás sobre las cualidades positivas que aprecias de las personas con las que interactúas diariamente.
Palabras clave	Relaciones positivas.
Instrucciones para el aprendedor	Puedes obtener mayor gozo de los momentos que compartes con tus colegas si te tomas el tiempo para pensar en lo que valoras y aprecias de ellos. Diversas investigaciones muestran que enfocarse en lo positivo que sucede diariamente ayuda a incrementar nuestra felicidad y lo mismo aplica a todas nuestras relaciones cercanas. El psicólogo John Gottman sugiere que, para tener relaciones felices con alguna persona, es necesario aspirar a tener cinco interacciones positivas por cada interacción negativa que se tenga con ella. Enfócate en tus compañeros y/o colegas y piensa en las siguientes preguntas. En cada caso, anota ejemplos específicos. 1. ¿Qué te atrajo de tus compañeros cuando se conocieron? 2. ¿Qué cosas han disfrutado al hacerlas juntos? 3. ¿Qué cosas realmente aprecias de ellos en este momento? 4. ¿Cuáles son sus fortalezas? Ahora, lo más importante es que cuando estés con tus compañeros te tomes el tiempo para darte cuenta y reconocer estas cualidades, sus fortalezas y las cosas que ellos hacen que realmente aprecies, así como los momentos agradables que han compartido. Piensa en estas declaraciones: • “Realmente me encanta cuando ellos...”. • “Son tan buenos para…”. • “Viéndolos hacer…, me recuerda ese fantástico día cuando nosotros...”. Aunque realizar dicho análisis con todas las personas que conoces resulta poco práctico, puedes usar los mismos principios para mejorar tus relaciones en general. Por ejemplo, antes de pasar tiempo con alguien tómate un momento para pensar en aquellas cosas que te gustan, aprecias o admiras de esa persona o cómo te hacen sentir bien. Asimismo, después de pasar tiempo con esa persona, piensa en las cosas que apreciaste o lo que disfrutaste del tiempo que pasaron juntos.
Fuente	Basado en el Catálogo de actividades para profesores.

Práctica 10

Nombre de la práctica	Las fortalezas se muestran en nuestras historias.
Descripción de la práctica	Reflexionarás sobre las fortalezas de carácter que aplicaste en una situación.
Palabras clave	Fortalezas de carácter.
Instrucciones para el aprendedor	Antes de comenzar el ejercicio, ¿sabes cuáles son las fortalezas de carácter? Consulta la descripción de las 24 fortalezas de carácter en la siguiente liga: **El siguiente enlace es externo a la Universidad Tecmilenio, al acceder a este considera que debes apegarte a sus términos y condiciones.** http://www.viacharacter.org/www/Character-Strengths/VIA-Classification Luego de que leas cuáles son las fortalezas de carácter, realiza lo que se pide a continuación: 1. Describe detalladamente, mediante un texto, una anécdota en la que hayas llevado a cabo alguna acción de la mejor manera posible, o bien, que hayas actuado por encima de lo ordinario. Procura enfocarlo al entorno laboral. 2. Puede ser cualquier suceso que te haya marcado por la manera en que te desenvolviste. 3. Señala en tu descripción: ¿qué ocurrió?, ¿qué papel jugaste en el suceso?, ¿qué acciones llevaste a cabo que fueron de utilidad para ti y para los demás? 4. Luego de que hayas terminado de escribir, lee tu texto y subraya las palabras y oraciones que te den una idea sobre cómo usaste cualquiera de las 24 fortalezas de carácter. 5. Observa y clasifica cuáles son las fortalezas que usaste en tu anécdota. Reflexiona sobre el impacto que estas pueden tener en tu desempeño cotidiano.
Fuente	Niemiec, R. (2016). *How to Assess Your Strengths: 5 Tactics for Self-Growth*. Recuperado de https://www.psychologytoday.com/us/blog/what-matters-most/201603/how-assess-your-strengths-5-tactics-self-growth

Práctica 11

Nombre de la práctica	Tus fortalezas en los ojos del otro.
Descripción de la práctica	En la práctica podrás reflexionar sobre la percepción que otros tienen sobre tus fortalezas de carácter.
Palabras clave	Fortalezas de carácter.
Instrucciones para el aprendedor	¿Recuerdas alguna ocasión en la que hablaste con algún colega y este te reveló algo positivo que piensa de ti? Cuando esto ocurre, usualmente deja huella en nuestros comportamientos y acciones, pues nos damos cuenta de que las personas tienen percepciones sobre nuestras fortalezas que nosotros mismos no vislumbramos. Haz lo siguiente: 1. Piensa sobre alguna vez que algún compañero de trabajo te compartió lo que piensa de ti y que te haya sorprendido. 2. Piensa en lo siguiente: ¿qué fue lo que te llamó más la atención?, ¿qué fortalezas vio en ti que pensaste que no tenías tan desarrolladas? 3. Por último, señala en un texto por qué consideras que esta revelación te causó tanto impacto, así como la manera en que te ayudó a cultivar tus fortalezas de carácter.
Fuente	Niemiec, R. (2016). *How to Assess Your Strengths: 5 Tactics for Self-Growth*. Recuperado de https://www.psychologytoday.com/us/blog/what-matters-most/201603/how-assess-your-strengths-5-tactics-self-growth

Práctica 12

Nombre de la práctica	Plantea tus objetivos como metas de aproximación y replantea tus metas de evitación.
Descripción de la práctica	Con base en lo que plantea Grenville (2012), en la práctica podrás definir diferentes tipos de metas y encontrar la mejor manera de conseguirlas.
Palabras clave	Objetivos, metas y planes.
Instrucciones para el aprendedor	La autora Bridget Grenville-Cleave (2012) comenta que en el establecimiento de metas es importante distinguir los tipos de metas que hay y menciona dos: 1. Metas de aproximación (*approach*): son las metas con resultados positivos (deseables, placenteros, benéficos o que nos gustaría tener) y hacia las cuales trabajamos. 2. Metas de evitación (*avoidance*): son las metas con resultados negativos (indeseables, dolorosos, dañinos, o nos disgustan) y en las cuales trabajamos para evitarlas. Ejemplo: **Meta de aproximación:** • Ser más eficiente. • Ser amigable y extrovertido en reuniones. • Asumir el rol de líder en el trabajo. **Meta de evitación:** • Dejar de aplazar. • Dejar de ser tan tímido en las reuniones. • No pasar desapercibido en el trabajo. Las investigaciones que se han realizado respecto a estos tipos de metas muestran que perseguir metas de evitación resulta en un detrimento del bienestar. Estos descubrimientos sugieren que el establecer metas de aproximación o replantear las metas de evitación es benéfico. Reflexiona lo siguiente: • ¿Qué tipo de metas te has planteado tú? • ¿Hay algunas metas que puedas replantear en una forma más positiva? • ¿Cuándo las tendrás listas?
Fuente	Grenville, B. (2012). *GOAL-SETTING SECRETS.* Recuperado de http://positivepsychologynews.com/news/bridget-grenville-cleave/2012013120696