HOJA CALENDARIO– PLANIFICACIÓN DE LAS SESIONES DE CLASE

MODALIDAD PRESENCIAL

Nombre de la asignatura	Construcción de Software	**Resultado de aprendizaje de la asignatura:**	Al finalizar la asignatura, el estudiante será capaz de aplicar los principios fundamentales de la construcción de software empleando metodologías y herramientas de construcción de software.
Ciclo	7	**EAP**	Ingeniería de Sistemas e Informática.

COMPETENCIA	CRITERIOS	NIVEL	ESPECIFICACIÓN
Gestión de proyectos	**C. 1 Diseño del proyecto**	Intermedio	Prepara una propuesta preliminar del proyecto para atender las necesidades identificadas.
	C. 2 Planificación de la gestión	Intermedio	Categoriza los recursos necesarios para llevar a cabo las actividades identificadas.
	C. 3 Ejecución del proyecto	Intermedio	Coordina equipos de trabajo para cumplir con las actividades planeadas, estableciendo responsabilidades.
Experimentación	**C. 1 Desarrollo de experimentos**	Intermedio	Realiza experimentos o pruebas de ensayo identificando los objetivos, principios, procedimientos y recursos necesarios.
	C. 2 Análisis e interpretación de resultados	Intermedio	Clasifica información relevante de los experimentos o pruebas de ensayo que realiza, validando los resultados obtenidos.
Diseño y desarrollo de soluciones	**C. 1 Definición de requerimientos y restricciones**	Intermedio	Identifica y define requerimientos y restricciones de forma clara sin llegar a validarlos.
	C. 2 Diseño y desarrollo de sistemas, componentes o procesos	Intermedio	Aplica los procedimientos necesarios para el diseño preliminar de un sistema, considerando los requerimientos y restricciones.
Uso de herramientas modernas	**C. 1 Uso de técnicas y metodologías**	Intermedio	Compara técnicas o metodologías apropiadas para la solución de un problema.
	C. 2 Uso de herramientas	Intermedio	Compara herramientas apropiadas para la solución de un problema.

Unidad 1	**Nombre de la unidad:**	Construcción sólida desde el inicio	**Resultado de aprendizaje de la unidad:**	**Al finalizar la unidad, el estudiante será capaz de desarrollar aplicaciones sencillas integrando prácticas ágiles, empleando herramientas modernas como la inteligencia artificial generativa para la creación de código, y aplicando principios de código limpio, estilo de código y refactorización, garantizando así la calidad, la mantenibilidad y la legibilidad del software.**	**Duración en horas**	24

Semana	**Horas / Tipo de sesión**	**Temas y subtemas**	**Propósito**	**Metodología / Estrategias**	**Actividades para la enseñanza aprendizaje (Docente - Estudiante)**	**Recursos**	**Actividades de aprendizaje autónomo Asíncronas (Estudiante – Aula virtual)**
1	**2T**	- Presentación de la asignatura y el sílabo - Presentación del docente y estudiante **Desarrollo rápido de aplicaciones sencillas.** - Conceptos básicos de aplicaciones sencillas: estructura, componentes y flujo de trabajo. - Introducción a las prácticas ágiles (Scrum, Kanban) y su aplicación en el desarrollo de software. - Herramientas modernas para desarrollo ágil.	- Al finalizar la sesión, cada estudiante desarrolla aplicaciones sencillas aplicando prácticas ágiles y utilizando herramientas modernas de desarrollo, en el contexto de un entorno configurado y un flujo de trabajo ágil.	**Clase expositiva / lección magistral,**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente introduce el tema de las metodologías ágiles (Scrum, Kanban) y su importancia en el desarrollo de software. - Se realiza una lluvia de ideas sobre problemas cotidianos que podrían resolverse con enfoque ágil. ➢ **Desarrollo**: - El docente introduce el tema de las metodologías ágiles (Scrum, Kanban) y su importancia en el desarrollo de software. - Se realiza una lluvia de ideas sobre problemas cotidianos que podrían resolverse con enfoque ágil. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Se realiza una discusión grupal sobre los beneficios de las metodologías ágiles y cómo pueden aplicarse en proyectos reales. El docente responde a las preguntas de los estudiantes.	- Proyector multimedia - Herramientas informáticas para la educación	- Lectura: Leer el post sobre 10 diferencias entre Scrum y Kanban (https://openwebinars.net/blog/diferencias-scrum-kanban/). - Video: Ver el video tutorial sobre cómo crear prototipos funcionales con Figma (https://www.youtube.com/watch?v=AdwaVcpBjmQ). - Foro: Participar en un foro de discusión sobre los beneficios de las metodologías ágiles.
	4P	- Configuración del entorno de desarrollo (IDE, frameworks, etc.). - Desarrollo de una aplicación sencilla (ejemplo: calculadora, lista de tareas) aplicando prácticas ágiles. - Uso de herramientas de control de versiones (Git) para gestionar el código.		**Aprendizaje Colaborativo,**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente presenta un prototipo básico de una aplicación sencilla y explica los pasos para su creación. - Se realiza una lluvia de ideas sobre los pasos para elaborar un prototipo. ➢ **Desarrollo**: - Los estudiantes, guiados por el docente, realizan la planificación de un proyecto pequeño. - Los estudiantes, guiados por el docente, comienzan a crear su propio prototipo funcional utilizando un IDE. Se fomenta la colaboración entre estudiantes. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Los estudiantes presentan sus prototipos funcionales y reciben retroalimentación del docente y sus compañeros.	- Proyector multimedia - Herramientas de desarrollo	

Las actividades de aprendizaje autónomo en el aula virtual son las realizadas por el estudiante. Cada semana, el docente tiene el rol de monitorear, supervisar, evaluar y retroalimentar estas actividades, además de atender los foros y las comunicaciones generadas en el aula virtual.

Universidad Continental

HOJA CALENDARIO– PLANIFICACIÓN DE LAS SESIONES DE CLASE

MODALIDAD PRESENCIAL

2	**2T**	**Uso práctico de inteligencia artificial generativa en generación de código.** - Introducción a la inteligencia artificial generativa (IA generativa) y su aplicación en la generación de código. - Herramientas de IA generativa para desarrollo de software (ejemplos: GitHub Copilot, ChatGPT, Codex). - Ventajas y limitaciones de la IA generativa en la creación de código. - Integración de la IA generativa en el flujo de trabajo de desarrollo.	- Al finalizar la sesión, cada estudiante integra herramientas de inteligencia artificial generativa para generar y ajustar código automáticamente, en el contexto de un proyecto de desarrollo de software	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente introduce el concepto de inteligencia artificial generativa y su aplicación en el desarrollo de software. - Se realiza una lluvia de ideas sobre el futuro de los desarrolladores de software ➢ **Desarrollo**: - El docente explica cómo funcionan herramientas como GitHub Copilot y ChatGPT, y cómo pueden generar código automáticamente. Se discuten las ventajas y limitaciones de estas herramientas. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Se realiza una discusión grupal sobre el impacto de la IA generativa en el desarrollo de software y cómo puede mejorar la productividad.	- Proyector multimedia - herramientas informáticas para la educación	- Lectura: Leer el post sobre el Impacto de la Inteligencia Artificial en el Desarrollo de Software (https://sg.com.mx/revista/56/inteligencia-artificial-desarrollo-software). - Video: Ver el video tutorial sobre Como usar Chat GPT para crear un CÓDIGO de Python (https://www.youtube.com/watch?v=RfI_xuD0W5c). - Foro: Participar en un foro de discusión sobre las implicaciones éticas del uso de IA generativa.
	4P	- Configuración y uso de una herramienta de IA generativa (ejemplo: GitHub Copilot). - Generación de código automático para una aplicación sencilla usando IA generativa. - Análisis y ajuste del código generado para cumplir con los requisitos del proyecto.		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente muestra un ejemplo de código generado automáticamente utilizando una herramienta de IA generativa. - Se realiza una lluvia de ideas sobre la calidad de código y como mejorarlo. ➢ **Desarrollo**: - Los estudiantes integran una herramienta de IA generativa en su flujo de trabajo para desarrollar una funcionalidad específica de su aplicación. El docente supervisa y guía el proceso. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Los estudiantes presentan el código generado y discuten cómo la IA les ayudó a mejorar su productividad.	- Presentación en PowerPoint. - Herramientas de IA generativa. - Presentación en PowerPoint.	
3	**2T**	Manejo de excepciones para robustez en aplicaciones. - Conceptos básicos de manejo de excepciones: qué son, cuándo y cómo usarlas. - Tipos de excepciones y buenas prácticas para su manejo. - Diseño de aplicaciones robustas: prevención de errores y gestión de fallos. - Pruebas unitarias y de integración para validar la robustez del software.	- Al finalizar la sesión, cada estudiante implementa manejo de excepciones para garantizar la robustez de una aplicación, en el contexto de un código existente y pruebas unitarias validadas.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente introduce el concepto de manejo de excepciones y su importancia para la robustez de las aplicaciones. - Se realiza una lluvia de ideas sobre la importancia de las pruebas unitarias ➢ **Desarrollo**: - El docente explicará los fundamentos del manejo de excepciones, los tipos de errores comunes y las buenas prácticas para manejar excepciones en aplicaciones. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Los estudiantes implementarán bloques de excepciones en su aplicación y personalizarán excepciones para mejorar la robustez. El docente supervisará y corregirá los errores comunes.	- Proyector multimedia - herramientas informáticas para la educación	- Lectura: Leer el post sobre Manejo de Excepciones y Errores en Python (https://www.datacamp.com/es/tutorial/exception-handling-python). - Video: Ver el video How to write unit tests in Python using pytest and PyCharm (https://www.youtube.com/watch?v=Z0f00BdJ3yw). - Foro: Participar en un foro de discusión sobre la importancia de la validación de entradas.
	4P	- Implementación de manejo de excepciones en una aplicación existente. - Desarrollo de pruebas unitarias para validar el manejo de excepciones. - Refactorización de código para mejorar la robustez y legibilidad.		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente presenta un ejemplo de código con manejo de excepciones y explica cómo implementar bloques de excepciones. - Se realiza una lluvia de ideas sobre el manejo de excepciones y errores. ➢ **Desarrollo**: - Los estudiantes implementan manejo de excepciones en su aplicación y personalizan excepciones para mejorar la robustez. El docente supervisa y corrige errores comunes. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Los estudiantes presentan su código con manejo de excepciones y reciben retroalimentación del docente.	- Proyector multimedia - Herramientas de desarrollo	

Las actividades de aprendizaje autónomo en el aula virtual son las realizadas por el estudiante. Cada semana, el docente tiene el rol de monitorear, supervisar, evaluar y retroalimentar estas actividades, además de atender los foros y las comunicaciones generadas en el aula virtual.

Universidad Continental

HOJA CALENDARIO– PLANIFICACIÓN DE LAS SESIONES DE CLASE

MODALIDAD PRESENCIAL

4	**2T**	Manejo de excepciones para robustez en aplicaciones. - Principios de código limpio (Clean Code): legibilidad, simplicidad y mantenibilidad. - Estilo de código: convenciones y estándares (ejemplo: PEP 8 para Python, convenciones de Java). - Refactorización: técnicas y herramientas para mejorar la calidad del código. - Principios de diseño SOLID y su aplicación en el desarrollo de software.	- Al finalizar la sesión, cada estudiante aplica principios de código limpio y refactorización para mejorar la mantenibilidad y legibilidad del software, en el contexto de un proyecto existente y usando herramientas de análisis estático.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente introduce los principios de código limpio (SOLID) y su importancia para la mantenibilidad del software. - Se realiza una lluvia de ideas sobre ¿por qué se debe construir sobre mantenible? ➢ **Desarrollo**: - El docente explica las técnicas de refactorización más comunes, como la extracción de métodos y el renombrado, y cómo identificar "code smells". ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Se realiza una discusión grupal sobre cómo la refactorización puede mejorar la calidad del código y facilitar su mantenimiento.	- Proyector multimedia - herramientas informáticas para la educación	- Lectura: Leer el post sobre Principios SOLID en Python: Una Guía para un Código más Limpio y Mantenible (https://programacion365.com/principios-solid/). - Video: Ver el video Formatters & Linters in Python (https://www.youtube.com/watch?v=906PWxtteFQ). - Foro: Participar en un foro de discusión sobre la importancia de la refactorización en el ciclo de vida del desarrollo.
	4P	- Aplicación de principios de código limpio en un proyecto existente. - Refactorización de código para mejorar la mantenibilidad y legibilidad. - Uso de herramientas de análisis estático de código para evaluar la calidad del software.		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente presenta un fragmento de código con varios "code smells" y explica cómo refactorizarlo. - Se realiza una lluvia de ideas sobre que técnicas se refactorización se aplicó y donde. ➢ **Desarrollo**: - Los estudiantes refactorizan el código de su aplicación, aplicando las convenciones de estilo y las técnicas de refactorización aprendidas. El docente supervisa y proporciona retroalimentación. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Los estudiantes presentan su código refactorizado y discuten los cambios realizados.	- Proyector multimedia - Herramientas de desarrollo	

-

Las actividades de aprendizaje autónomo en el aula virtual son las realizadas por el estudiante. Cada semana, el docente tiene el rol de monitorear, supervisar, evaluar y retroalimentar estas actividades, además de atender los foros y las comunicaciones generadas en el aula virtual.

Universidad Continental

HOJA CALENDARIO– PLANIFICACIÓN DE LAS SESIONES DE CLASE

MODALIDAD PRESENCIAL

Unidad 2	Nombre de la unidad:	Colaboración eficiente en entornos distribuidos	Resultado de aprendizaje de la unidad:	Al finalizar la unidad, el estudiante será capaz de implementar control de versiones con Git y GitHub, adoptando flujos de trabajo como GitFlow para la organización colaborativa y automatizando procesos de construcción, garantizando entregas iterativas y consistentes.	Duración en horas	24

Semana	Horas / Tipo de sesión	Temas y subtemas	Propósito	Metodología /Estrategias	Actividades para la enseñanza aprendizaje (Docente - Estudiante)	Recursos	Actividades de aprendizaje autónomo Asíncronas (Estudiante – Aula virtual)
5	2T	Control de versiones efectivo con Git y GitHub. - Introducción al control de versiones - Fundamentos de Git. - Introducción a GitHub	- Al finalizar la sesión, cada estudiante utiliza Git y GitHub para gestionar un repositorio de código, realizando commits, clonando repositorios y manejando ramas básicas, en un entorno local y remoto.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente realiza una actividad de motivación mostrando un caso real donde el control de versiones evitó un desastre en un proyecto de software. Los estudiantes interactúan respondiendo preguntas como "¿Qué hubieran hecho en esa situación?". ➢ **Desarrollo**: - El docente explica los conceptos básicos de Git y GitHub mediante una exposición dialogada, complementada con una demostración en vivo de comandos básicos (commit, clone, push, pull). ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Se realiza una síntesis de los conceptos clave y se abre un espacio para preguntas y retroalimentación.	- Proyector multimedia - herramientas informáticas para la educación	- Leer el capítulo 1 del libro "Pro Git" (https://git-scm.com/book/es/v2) sobre los fundamentos de Git. - Ver el video Guía instalación y manejo básico de Git con pycharm (https://www.youtube.com/watch?v=HNni9ehnSvA).
	4P	- Instalación y configuración de Git en el entorno local. - Creación de un repositorio local y realización de commits básicos. - Clonación de un repositorio desde GitHub. - Ejercicios básicos de manejo de ramas y merge. - Subida de cambios a un repositorio remoto en GitHub.		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente motiva a los estudiantes con un ejercicio rápido donde deben adivinar qué hace cada comando básico de Git. ➢ **Desarrollo**: - Los estudiantes siguen un taller guiado para instalar Git, crear un repositorio local, realizar commits básicos y clonar un repositorio desde GitHub. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Los estudiantes comparten sus resultados y reflexionan sobre lo aprendido.	- Proyector multimedia - Herramientas de desarrollo	
6	2T	Control de versiones efectivo con Git y GitHub con ramas. - Trabajo avanzado con ramas - Uso de herramientas complementarias - Buenas prácticas en el uso de Git	- Al finalizar la sesión, cada estudiante aplica técnicas avanzadas de Git, como la resolución de conflictos, el uso de git stash y git rebase, y colabora en proyectos mediante GitHub, siguiendo buenas prácticas de mensajes de commit y exclusión de archivos innecesarios.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente motiva a los estudiantes con un caso de estudio donde se resolvió un conflicto complejo en un proyecto colaborativo. Los estudiantes proponen alternativas para evitar conflictos. ➢ **Desarrollo**: - El docente explica técnicas avanzadas de Git (resolución de conflictos, git stash, git rebase) mediante una exposición interactiva y casos de estudio. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Se realiza una síntesis de las técnicas y se discuten las mejores prácticas.	- Proyector multimedia - herramientas informáticas para la educación	- Leer el capítulo 3 del libro "Pro Git" (https://git-scm.com/book/es/v2) sobre ramas y resolución de conflictos. - Ver el video Resuelve conflictos en Git - El poder de rebase y stash en git (https://www.youtube.com/watch?v=1tEORD7FTPI&t=810s).
	4P	- Ejercicios de creación y fusión de ramas con y sin conflictos. - Uso de git stash para guardar cambios temporales. - Práctica de git rebase para reorganizar el historial. - Configuración de un archivo gitignore para excluir archivos innecesarios. - Ejercicios de colaboración en GitHub: fork, pull request y revisión de código.		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente motiva a los estudiantes con un ejercicio donde deben identificar posibles conflictos en un escenario simulado. ➢ **Desarrollo**: - Los estudiantes trabajan en equipos para resolver conflictos, usar git stash y aplicar git rebase. También configuran un archivo .gitignore. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Cada equipo presenta su solución y recibe retroalimentación.	- Proyector multimedia - Herramientas de desarrollo	

Las actividades de aprendizaje autónomo en el aula virtual son las realizadas por el estudiante. Cada semana, el docente tiene el rol de monitorear, supervisar, evaluar y retroalimentar estas actividades, además de atender los foros y las comunicaciones generadas en el aula virtual.

Universidad Continental

HOJA CALENDARIO– PLANIFICACIÓN DE LAS SESIONES DE CLASE

MODALIDAD PRESENCIAL

7	**2T**	Implementación del flujo de trabajo GitFlow para equipos. - Introducción a GitFlow - Ventajas y desventajas de GitFlow - Herramientas para facilitar GitFlow	- Al finalizar la sesión, cada estudiante implementa el flujo de trabajo GitFlow en un proyecto colaborativo, manejando ramas como feature, release y hotfix, y resolviendo conflictos en un entorno de desarrollo organizado.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente motiva a los estudiantes con un caso real donde GitFlow mejoró la organización de un proyecto. Los estudiantes dan su apreciación crítica. ➢ **Desarrollo**: - El docente explica el flujo de trabajo GitFlow mediante una exposición conceptual y ejemplos prácticos. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Se realiza una síntesis de las etapas de GitFlow y se discuten sus ventajas y desventajas.	- Proyector multimedia - herramientas informáticas para la educación	- Leer el post GitFlow: ¿Qué és y cómo aplicarlo sin morir en el intento? (https://gfourmis.co/gitflow-sin-morir-en-el-intento/). - Ver el video Domina Git y GitFlow con un ejemplo real paso a paso (https://www.youtube.com/watch?v=oFi8Pr3Gsjc).
	4P	- Configuración de un repositorio con GitFlow. - Creación y manejo de ramas feature, release y hotfix. - Simulación de un flujo de trabajo colaborativo con GitFlow. - Resolución de conflictos en un entorno GitFlow. - Ejercicios de integración continua con GitHub Actions (configuración básica).		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente motiva a los estudiantes con un ejercicio donde deben identificar las ramas necesarias para un proyecto simulado. ➢ **Desarrollo**: - Los estudiantes implementan GitFlow en un proyecto, creando y manejando ramas (feature, release, hotfix) y resolviendo conflictos. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Cada equipo presenta su flujo de trabajo y recibe retroalimentación.	- Proyector multimedia - Herramientas de desarrollo	
8	**2T**	Gestión y automatización en la construcción del software. - Planificación de las iteraciones. - Automatización de procesos en el desarrollo de software - Configuración de pipelines básicos.	- Al finalizar la sesión, cada estudiante configura un pipeline de integración y entrega continua (CI/CD) utilizando GitHub Actions, automatizando procesos de compilación, pruebas y despliegue, garantizando entregas iterativas y consistentes en un proyecto de software.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente motiva a los estudiantes con un caso real donde la automatización mejoró la eficiencia de un proyecto. Los estudiantes comentan sobre las ventajas de automatizar. ➢ **Desarrollo**: - El docente explica los conceptos de planificación de las iteraciones, .CI/CD y GitHub Actions mediante una exposición interactiva y análisis de ejemplos. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Se realiza una síntesis de los conceptos clave y se discuten las mejores prácticas.	- Proyector multimedia - herramientas informáticas para la educación	- Leer el post GitHub Actions Tutorial – Getting Started & Examples (https://spacelift.io/blog/github-actions-tutorial) . - Ver el video Introducción a Github Actions (https://www.youtube.com/watch?v=ePXwmNXG_nA&t=86s).
	4P	- Configuración de un pipeline básico con GitHub Actions - Ejercicios de integración de pruebas unitarias en el pipeline. - Simulación de un proceso de entrega continua (CD) con despliegue en un entorno de prueba. - Creación de un flujo de trabajo completo: desde el desarrollo hasta la entrega. - Revisión y optimización del pipeline para garantizar entregas iterativas y consistentes.		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente motiva a los estudiantes con un ejercicio rápido donde deben identificar los pasos necesarios para automatizar un proceso. ➢ **Desarrollo**: - Los estudiantes configuran un pipeline básico en GitHub Actions, integran pruebas unitarias y simulan un proceso de entrega continua. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Cada estudiante o equipo presenta su pipeline y recibe retroalimentación.	- Proyector multimedia - Herramientas de desarrollo	

Las actividades de aprendizaje autónomo en el aula virtual son las realizadas por el estudiante. Cada semana, el docente tiene el rol de monitorear, supervisar, evaluar y retroalimentar estas actividades, además de atender los foros y las comunicaciones generadas en el aula virtual.

Universidad Continental

HOJA CALENDARIO– PLANIFICACIÓN DE LAS SESIONES DE CLASE

MODALIDAD PRESENCIAL

Unidad 3	Nombre de la unidad:	Pruebas automatizadas para iteraciones exitosas	Resultado de aprendizaje de la unidad:	Al finalizar la unidad, el estudiante será capaz de diseñar pruebas unitarias, aplicando el enfoque de Desarrollo Guiado por Pruebas (TDD), para la resolución de problemas con ejercicios prácticos como Kata TDD y empleo de técnicas de mapeo Objeto-Relacional (ORM) para el manejo eficiente de datos.	Duración en horas	

Semana	Horas / Tipo de sesión	Temas y subtemas	Propósito	Metodología /Estrategias	Actividades para la enseñanza aprendizaje (Docente - Estudiante)	Recursos	Actividades de aprendizaje autónomo Asíncronas (Estudiante – Aula virtual)
9	2T	Introducción a pruebas unitarias y su integración en ciclos ágiles. - Introducción a las pruebas unitarias - Integración de pruebas unitarias en ciclos ágiles - Herramientas y frameworks para pruebas unitaria	- Al finalizar la sesión, cada estudiante identifica los conceptos básicos de las pruebas unitarias y su importancia, configura un entorno de desarrollo con un framework de pruebas unitarias, y crear pruebas unitarias básicas, en el contexto de un proyecto ágil.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - Motivación mediante una lluvia de ideas sobre la importancia de las pruebas unitarias en el desarrollo de software, permitiendo la interacción entre docente y estudiantes. ➢ **Desarrollo**: - Exposición dialogada sobre los conceptos básicos de pruebas unitarias y su integración en ciclos ágiles. Discusión guiada sobre herramientas y frameworks. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Síntesis y retroalimentación mediante preguntas reflexivas sobre lo aprendido.	- Proyector multimedia - herramientas informáticas para la educación	- Lectura del post Tutorial de PyTest (https://misovirtual.virtual.uniandes.edu.co/codelabs/tutorial-PyTest/index.html?index=..%2F..index#0). - Ver el vídeo Cómo realizar pruebas Unitarias y de Itegración con PyTest (https://www.youtube.com/watch?v=BRaRrQWv29A).
	4P	- Configuración de un entorno de desarrollo con un framework de pruebas unitarias. - Creación de pruebas unitarias básicas para funciones simples. - Ejecución de pruebas y análisis de resultados. - Integración de pruebas unitarias en un proyecto existente.		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - Motivación mediante la presentación de un caso práctico donde se muestre el impacto de las pruebas unitarias en un proyecto real. Los estudiantes comentan sobre la cobertura de la prueba. ➢ **Desarrollo**: - Taller guiado para configurar un entorno de desarrollo con un framework de pruebas unitarias. Ejercicio guiado para crear pruebas unitarias básicas. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Retroalimentación y discusión sobre los resultados obtenidos en las pruebas.	- Proyector multimedia - Herramientas de desarrollo	
10	2T	Práctica de Desarrollo Guiado por Pruebas (TDD). - Introducción al Desarrollo Guiado por Pruebas (TDD) - Buenas prácticas en TDD - Ejemplos de TDD en diferentes lenguajes de programación.	- Al finalizar la sesión, cada estudiante aplica el ciclo de Desarrollo Guiado por Pruebas (TDD) (Red, Green, Refactor), escribiendo pruebas significativas y refactorizando código, en el contexto de un ejercicio práctico simple.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - Motivación mediante una demostración en vivo de un ciclo de TDD (Red, Green, Refactor) en un ejemplo simple. Los estudiantes responden ¿por qué es importante que la primera prueba falle? ➢ **Desarrollo**: - Exposición interactiva sobre el ciclo de TDD y buenas prácticas. Discusión guiada sobre ejemplos de TDD en diferentes lenguajes. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Síntesis y retroalimentación mediante preguntas sobre los beneficios de TDD.	- Proyector multimedia - herramientas informáticas para la educación	- Leer el post ¿Qué es y para qué sirve un TDD o Test Driven Development? (https://intelequia.com/es/blog/post/qu%C3%A9-es-y-para-qu%C3%A9-sirve-un-tdd-o-test-driven-development#:~:text=%C2%BFQu%C3%A9%20es%20Test%20Driven%20Development,antes%20de%20escribir%20el%20c%C3%B3digo.). - Ver el videoAprende a TRIANGULAR en TDD para pasar de rojo a verde (https://www.youtube.com/watch?v=1gttkO9JKtU).
	4P	- Implementación del ciclo de TDD en un ejercicio simple (por ejemplo, una calculadora). - Escritura de pruebas para una funcionalidad específica antes de implementar el código. - Refactorización del código después de pasar las pruebas. - Discusión en grupo sobre los desafíos y beneficios de TDD.		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - Motivación mediante la presentación de un problema práctico que se resolverá aplicando TDD. Los estudiantes responden ¿De dónde se originan los casos de prueba? ➢ **Desarrollo**: - Taller práctico para implementar el ciclo de TDD en un ejercicio simple. Ejercicio guiado para escribir pruebas antes de implementar el código. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Retroalimentación y discusión sobre los desafíos encontrados durante la práctica.	- Proyector multimedia - Herramientas de desarrollo	
11	2T	Resolución de problemas con Katas TDD. - Introducción a las Katas TDD - Estrategias para resolver Katas TDD	- Al finalizar la sesión, cada estudiante resuelve problemas de programación utilizando Katas TDD, aplicando el ciclo de TDD y mejorando continuamente su solución, en un entorno colaborativo y de retroalimentación.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - Motivación mediante la presentación de una Kata TDD resuelta, mostrando el proceso paso a paso. Los estudiantes responden ¿Cuál es la importancia del Kata TDD? ➢ **Desarrollo**: - Exposición conceptual sobre las Katas TDD y estrategias para resolverlas. Discusión participativa sobre cómo abordar problemas con TDD. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Síntesis y retroalimentación mediante preguntas sobre las estrategias aprendidas.	- Proyector multimedia - herramientas informáticas para la educación	- Leer el post Como hacer una kata de código (https://franiglesias.github.io/resolving-code-katas/). - Ver el video Python - TDD - Kata Juego de Cartas (https://www.youtube.com/watch?v=4_fFz4UyHDU).

Las actividades de aprendizaje autónomo en el aula virtual son las realizadas por el estudiante. Cada semana, el docente tiene el rol de monitorear, supervisar, evaluar y retroalimentar estas actividades, además de atender los foros y las comunicaciones generadas en el aula virtual.

Universidad Continental

HOJA CALENDARIO– PLANIFICACIÓN DE LAS SESIONES DE CLASE

MODALIDAD PRESENCIAL

	4P	- Resolución de una Kata TDD en parejas o pequeños grupos. - Aplicación del ciclo de TDD en la resolución de la Kata. - Presentación de soluciones y discusión sobre diferentes enfoques. - Retroalimentación y mejora de las soluciones propuestas.		**Aprendizaje colaborativo**.	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - Motivación mediante la asignación de una Kata TDD para resolver en parejas o pequeños grupos. ➢ **Desarrollo**: - Taller de resolución de problemas aplicando el ciclo de TDD. Trabajo en equipo para presentar soluciones y recibir retroalimentación. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Discusión guiada sobre los diferentes enfoques utilizados y mejora de las soluciones propuestas.	- Proyector multimedia - Herramientas de desarrollo	
12	**2T**	Implementación ágil de mapeo Objeto-Relacional (ORM). - Introducción al mapeo Objeto-Relacional (ORM) - Herramientas de ORM - Integración de ORM en un proyecto ágil	- Al finalizar la sesión, cada estudiante implementa un mapeo Objeto-Relacional (ORM) en un proyecto, creando models y realizando operaciones CRUD, en el contexto de un proyecto ágil con pruebas unitarias.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - Motivación mediante la presentación de un caso práctico donde se utilice ORM en un proyecto real. Los estudiantes responden ¿Por qué usar ORM? ➢ **Desarrollo**: - Exposición interactiva sobre el mapeo Objeto-Relacional (ORM) y herramientas disponibles. Análisis de ejemplos de integración de ORM en proyectos ágiles. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Síntesis y retroalimentación mediante preguntas sobre los beneficios y desafíos del uso de ORM.	- Proyector multimedia - herramientas informáticas para la educación	- Leer el post ¿Qué ORM elegir? ¿Peewee o SQLAlchemy? (https://codigofacilito.com/articulos/peewee-or-sqlalchemy). - Ver el video Cómo conectar a una base de datos con un ORM y Python (https://www.youtube.com/watch?v=5zb5vwRv_oM).
	4P	- Configuración de un proyecto con un framework ORM. - Creación de modelos y mapeo a una base de datos. - Implementación de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) utilizando ORM. - Integración de ORM en un proyecto existente con pruebas unitarias.		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - Motivación mediante la presentación de un proyecto donde se integrará ORM. Los estudiantes responden ¿Qué modificaciones se tiene que realizar para cambiar de gestor de base de datos? ➢ **Desarrollo**: - Taller práctico para configurar un proyecto con un framework ORM. Ejercicio guiado para crear modelos y realizar operaciones CRUD. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Retroalimentación y discusión sobre la integración de ORM en un proyecto existente con pruebas unitarias.	- Proyector multimedia - Herramientas de desarrollo	

Las actividades de aprendizaje autónomo en el aula virtual son las realizadas por el estudiante. Cada semana, el docente tiene el rol de monitorear, supervisar, evaluar y retroalimentar estas actividades, además de atender los foros y las comunicaciones generadas en el aula virtual.

Universidad Continental

HOJA CALENDARIO– PLANIFICACIÓN DE LAS SESIONES DE CLASE

MODALIDAD PRESENCIAL

Unidad 4	Nombre de la unidad:	Implementación y despliegue ágil	Resultado de aprendizaje de la unidad:	Al finalizar la unidad, el estudiante será capaz de diseñar interfaces gráficas centradas en el usuario, almacenando la aplicación en un contenedor, facilitando su despliegue y garantizando su portabilidad y adaptabilidad a entornos productivos.	Duración en horas	24

Semana	Horas / Tipo de sesión	Temas y subtemas	Propósito	Metodología /Estrategias	Actividades para la enseñanza aprendizaje (Docente - Estudiante)	Recursos	Actividades de aprendizaje autónomo Asíncronas (Estudiante – Aula virtual)
13	2T	Diseño de interfaces gráficas centradas en el usuario. - Introducción a las interfaces gráficas de usuario (GUI) - Herramientas y frameworks para diseño de interface	- Al finalizar la sesión, cada estudiante diseña interfaces gráficas centradas en el usuario, aplicando principios de usabilidad y accesibilidad, y utilizando herramientas de prototipado para crear wireframes y mockups.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente presenta un caso real de una aplicación con una interfaz gráfica exitosa y pregunta a los estudiantes: "¿Qué elementos de la interfaz consideran que hacen que esta aplicación sea fácil de usar?". ➢ **Desarrollo**: - El docente realiza una exposición dialogada sobre los conceptos básicos de diseño de interfaces gráficas, incluyendo usabilidad, accesibilidad y experiencia de usuario (UX). - Se presentan ejemplos de herramientas de prototipado y frameworks para desarrollo de GUI. - El docente muestra un ejemplo de wireframe y mockup, explicando su importancia en el diseño centrado en el usuario. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Se realiza una discusión guiada sobre cómo los principios de diseño pueden aplicarse en proyectos reales.	- Proyector multimedia - herramientas informáticas para la educación	- Leer el post GUI programming with Python (https://dev.to/amigosmaker/gui-programming-with-python-spanish-2889). - Ver el video Use a Drag & Drop Editor to Make Tkinter Python GUI Applications (https://www.youtube.com/watch?v=oLxFqpUbaAE).
	4P	- Diseño de un prototipo de interfaz gráfica - Implementación básica de una interfaz		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente presenta un prototipo básico de una aplicación sencilla y pregunta: "¿Qué pasos seguirían para diseñar una interfaz similar?". ➢ **Desarrollo**: - El docente guía a los estudiantes en un taller guiado para crear un prototipo de interfaz gráfica utilizando herramientas de diseño. - Los estudiantes trabajan en equipos para diseñar wireframes y mockups, aplicando los principios de usabilidad y accesibilidad. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Los estudiantes presentan sus prototipos y reciben retroalimentación del docente y sus compañeros.	- Proyector multimedia - Herramientas de desarrollo	
14	2T	Uso de Docker para despliegue y portabilidad ágil de aplicaciones. - Introducción a Docker y contenedores - Configuración de entornos con Docker	- Al finalizar la sesión, cada estudiante configura y gestiona contenedores Docker, creando imágenes personalizadas y utilizando Docker Compose para facilitar el despliegue y garantizar la portabilidad de aplicaciones en diferentes entornos.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente presenta un caso real donde Docker resolvió un problema de portabilidad en un proyecto de software y pregunta: "¿Qué ventajas creen que ofrece Docker en el desarrollo de software?". ➢ **Desarrollo**: - El docente realiza una exposición interactiva sobre los conceptos básicos de Docker: imágenes, contenedores, Dockerfile y Docker Compose. - Se presentan ejemplos de cómo Docker facilita el despliegue y la portabilidad de aplicaciones. - El docente muestra una demostración en vivo de cómo crear y ejecutar un contenedor Docker. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Se realiza una discusión guiada sobre las ventajas y limitaciones de Docker.	- Proyector multimedia - herramientas informáticas para la educación	- Leer el postPython y Docker () . - Ver el video Guía de Docker y Python 2025 (https://www.youtube.com/watch?v=Q2BldKGQo34).

Las actividades de aprendizaje autónomo en el aula virtual son las realizadas por el estudiante. Cada semana, el docente tiene el rol de monitorear, supervisar, evaluar y retroalimentar estas actividades, además de atender los foros y las comunicaciones generadas en el aula virtual.

Universidad Continental

HOJA CALENDARIO– PLANIFICACIÓN DE LAS SESIONES DE CLASE

MODALIDAD PRESENCIAL

	4P	- Configuración de un contenedor Docker - Despliegue de una aplicación en un contenedor		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente presenta un ejemplo de una aplicación empaquetada en un contenedor Docker y pregunta: "¿Qué pasos seguirían para desplegar esta aplicación en otro entorno?". ➢ **Desarrollo**: - El docente guía a los estudiantes en un ejercicio guiado para configurar un contenedor Docker y desplegar una aplicación sencilla. - Los estudiantes trabajan en equipos para crear un Dockerfile, construir una imagen y ejecutar un contenedor. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Los estudiantes presentan sus contenedores y reciben retroalimentación del docente y sus compañeros.	- Proyector multimedia - Herramientas de desarrollo	
15	**2T**	Desarrollo del proyecto integrador - Planificación del proyecto integrador - Buenas prácticas de desarrollo en el proyecto integrador	- Al finalizar la sesión, cada estudiante desarrolla una aplicación integradora, combinando el diseño de interfaces gráficas con la lógica de negocio, y configurando un entorno de despliegue utilizando Docker para garantizar su portabilidad y adaptabilidad.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente presenta un ejemplo de un proyecto integrador exitoso y pregunta: "¿Qué elementos consideran clave para el éxito de un proyecto de software?". ➢ **Desarrollo**: - Los estudiantes trabajan de forma colaborativa en la elaboración de su proyecto integrador. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Se realiza una discusión guiada sobre cómo aplicar las buenas prácticas en el proyecto integrador.	- Proyector multimedia - herramientas informáticas para la educación	- Leer el post Creación de proyectos en Python con PyScaffold (https://misovirtual.virtual.uniandes.edu.co/codelabs/creacion-de-proyectos-PyScaffold/#0). - Ver el video Introducción a la documentación (https://www.youtube.com/watch?v=aZEjSzlMhsE).
	4P	- Desarrollo de la interfaz gráfica del proyecto integrador - Configuración del entorno de despliegue del proyecto integrador		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente presenta un prototipo del proyecto integrador y pregunta: "¿Qué pasos seguirían para completar este proyecto?". ➢ **Desarrollo**: - Los estudiantes trabajan de forma colaborativa en la elaboración de su proyecto integrador. ➢ **Cierre**: Metacognición, síntesis y retroalimentación - Se realiza una discusión guiada sobre cómo aplicar las buenas prácticas en el proyecto integrador.	- Proyector multimedia - Herramientas de desarrollo	
16	**2T**	Demostración del proyecto integrador - Presentación del proyecto integrador	- Al finalizar la sesión, cada estudiante presenta y demuestra una aplicación integradora, explicando el proceso de diseño, desarrollo y despliegue, y defendiendo las decisiones técnicas tomadas para garantizar la usabilidad, portabilidad y adaptabilidad del proyecto.	**Clase expositiva / lección magistral.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente presenta un prototipo del proyecto integrador y pregunta: "¿Qué pasos seguirían para completar este proyecto?". ➢ **Desarrollo**: - Presentación del proyecto integrador ➢ **Cierre**: Metacognición, síntesis y retroalimentación - - Los estudiantes reciben retroalimentación del docente y sus compañeros.	- Proyector multimedia - herramientas informáticas para la educación	- Leer el post Cómo crear un proceso de demostración de software que capte clientes (https://www.ipanemacomunicacion.com/blog/como-crear-un-proceso-de-demostracion-de-software-que-capte-clienteshttps://www.ipanemacomunicacion.com/blog/como-crear-un-proceso-de-demostracion-de-software-que-capte-clientes). - Ver el videoVídeos de demostración de software: ¡consejos! (https://www.youtube.com/watch?v=CWrIvL4mBtY) .
	4P	- Presentación del proyecto integrador		**Aprendizaje colaborativo.**	➢ **Inicio**: Motivación, se presenta el propósito de la sesión - El docente presenta un prototipo del proyecto integrador y pregunta: "¿Qué pasos seguirían para completar este proyecto?". ➢ **Desarrollo**: - Presentación del proyecto integrador ➢ **Cierre**: Metacognición, síntesis y retroalimentación - - Los estudiantes reciben retroalimentación del docente y sus compañeros.	- Proyector multimedia - Herramientas de desarrollo	