Universidad Nacional de Luján
Departamento de
Tecnologia

DISPOSICION PRESIDENTE/A DEL CONSEJO DIRECTIVO DEPARTAMENTO DE TECNOLOGÍA DISPPCD-T : 108 / 2025

LUJÁN, BUENOS AIRES

VISTO: La presentación del programa de la asignatura Sistemas Distribuidos y Programación Paralela (41429) correspondiente a la Carrera de Licenciatura en Sistemas de Información efectuada por el Profesor Responsable; y

CONSIDERANDO:

Que el referido programa se presentó ante la Comisión Plan de Estudios de la Carrera de Licenciatura en Sistemas de Información, la que aconseja su aprobación.

Que corresponde al Consejo Directivo la aprobación de los programas de las asignaturas de las distintas carreras a las que presta servicios académicos este Departamento, conforme el artículo 64, inciso d) del Estatuto de esta Universidad.

Que el Consejo Directivo Departamental, mediante Disposición DISPCD-TLUJ: 0000357/14, delegó en su Presidente la emisión de actos administrativos de aprobación de programas de asignaturas, que cuenten con el informe favorable de la Comisión Plan de Estudios correspondiente.

Por ello,

LA PRESIDENTA DEL CONSEJO DIRECTIVO

DEL DEPARTAMENTO DE TECNOLOGÍA

D I S P O N E:

ARTÍCULO 1°.- APROBAR el programa de la asignatura Sistemas Distribuidos y Programación Paralela (41429): 2025-2026 - Plan 17.14,

correspondiente a la Carrera de Licenciatura en Sistemas de Información, que como Anexo forma parte de la presente Disposición.-

ARTÍCULO 2°.- Regístrese, comuníquese, remítase a la Dirección General de Asuntos Académicos. Cumplido, archívese.-

Mgter. Jimena O. MAZIERES - Presidenta Consejo Directivo - Departamento de Tecnología

UNIVERSIDAD NACIONAL DE LUJÁN
DEPARTAMENTO DE TECNOLOGÍA

PROGRAMA OFICIAL

DENOMINACIÓN DE LA ACTIVIDAD: 41429 – Sistemas Distribuidos y Programación Paralela
TIPO DE ACTIVIDAD ACADÉMICA: Asignatura

CARRERA: Licenciatura en Sistemas de Información
PLAN DE ESTUDIOS: 17.14

DOCENTE RESPONSABLE:
Dr. Petrocelli, David Marcelo – Profesor Adjunto

EQUIPO DOCENTE:
An. Univ. Mariano Rapaport - Ayudante de primera
An. Univ. Nehuen Prados - Ayudante de primera

ACTIVIDADES CORRELATIVAS PRECEDENTES:
PARA CURSAR: 14029 - Administración de Redes ; 11286 - Programación en Ambiente Web

PARA APROBAR: 14029 - Administración de Redes ; 11286 - Programación en Ambiente Web

CARGA HORARIA TOTAL:
HORAS SEMANALES: 6 - HORAS TOTALES 96

DISTRIBUCIÓN INTERNA DE LA CARGA HORARIA:
TEÓRICO: 35% - 2 horas semanales
PRÁCTICO: 65% - 4 horas semanales

PERÍODO DE VIGENCIA DEL PRESENTE PROGRAMA: **2025 -2026**

CONTENIDOS MÍNIMOS O DESCRIPTORES

Sistemas distribuidos y Sistemas Operativos Distribuidos. Comunicación y sincronización. Algoritmos secuenciales, concurrentes, distribuidos y paralelos. Conceptos de Cluster, Grid y Cloud Computing. Concurrencia y Paralelismo. Comunicación por mensaje y memoria compartida en arquitecturas multiprocesador. Computación de alta performance (HPC). Máquinas paralelas. Programación Paralela CPU y GPU. Algoritmos paralelos fundamentales.

FUNDAMENTACIÓN, OBJETIVOS, COMPETENCIAS

FUNDAMENTACIÓN

El mundo se encuentra en un proceso de transformación continua con el uso de nuevas tecnologías, y en particular de aquellas que provienen del campo informático. La demanda en capacidad de cómputo, almacenamiento y conectividad por parte de usuarios a lo largo del planeta, junto con la necesidad de tráfico de datos de formatos y fuentes diversas, hacen que los conceptos, nociones y requerimientos de la infraestructura y el software asociado a estos nuevos desafíos se vuelvan cada vez más complejos y diversos. Los dispositivos y equipos informáticos evolucionan constantemente y se vuelven más potentes mientras que las redes de interconexión mejoran, son más confiables y otorgan mayor velocidad de respuesta, lo que posibilita atender ese crecimiento en demanda de recursos a través de los sistemas en "Sistemas Distribuidos", los cuales presentan ventajas de escalabilidad, confiabilidad, disponibilidad, performance y distribución inherente respecto de los sistemas centralizados. No obstante, las dificultades que presentan estos y la complejidad de los mismos requieren que el profesional conozca adecuadamente las ventajas y desventajas de adoptarlos, como así también que comprenda las alternativas de diseño en dichos sistemas.

El eje principal de esta asignatura es, por ello, presentar un análisis detallado de aquellos conceptos, características y habilidades prácticas vinculadas a los sistemas informáticos distribuidos, junto con una visión integral de aquellos requerimientos de infraestructura y el manejo de técnicas y actividades de programación asociadas al desarrollo de software bajo este tipo de paradigma. A su vez, se incorporan los fundamentos de virtualización, el uso de contenedores como herramienta esencial para el empaquetado y despliegue de aplicaciones, y la orquestación con Kubernetes para la gestión escalable de servicios distribuidos. Estas tecnologías permiten aprovechar de manera eficiente los recursos en entornos modernos, ya sea en la nube o en implementaciones locales.

Asimismo, desde la otra vertiente de la asignatura, los estudiantes abordan conocimientos generales acerca de la programación concurrente y paralela, y el análisis de sistemas distribuidos constituidos por nodos teniendo en cuenta la transparencia requerida en la programación de los mismos. Esto permite que los alumnos sean capaces de describir los modelos de cómputo paralelo y los paradigmas de programación paralela. De esta forma, se complementa la visión con la computación de alta performance (HPC), abordando arquitecturas multiprocesadoras y los esquemas de comunicación y sincronización necesarios para implementar soluciones de gran envergadura. Como resultado, los estudiantes podrán caracterizar los problemas de procesamiento paralelo desde dos puntos de vista: la arquitectura física y los lenguajes de programación, poniendo énfasis en la transformación de algoritmos secuenciales en paralelos.

Por último, estudiarán las métricas de performance asociadas al paralelismo, lo cual permitirá un adecuado análisis y optimización de los sistemas distribuidos y de las soluciones basadas en virtualización, contenedores y computación de alta performance.

OBJETIVOS GENERALES

Desarrollar en el estudiante capacidades suficientes para comprender y manejar los conceptos asociados a los Sistemas Informáticos Distribuidos, integrando fundamentos de virtualización y contenedores para el despliegue y la gestión de aplicaciones.

Generar competencias básicas para entender los requerimientos de componentes físicos y funcionales de los Sistemas Distribuidos, abarcando desde arquitecturas locales hasta entornos en la nube y de alta performance (HPC).

Potenciar las habilidades del estudiante en la comprensión y empleo de herramientas para el desarrollo de software soportado por sistemas distribuidos, valorando la importancia de la orquestación con Kubernetes y las prácticas de automatización (DevOps).

Comprender la implicancia de los Sistemas Distribuidos en el ambiente real actual, reconociendo su relevancia en términos de escalabilidad, disponibilidad y rendimiento para diversas áreas de aplicación.

Conocer la importancia de las distintas tecnologías y arquitecturas de los Sistemas Distribuidos, relacionándolas con los entornos paralelos y de altas prestaciones para maximizar su impacto en el ámbito profesional y académico.

OBJETIVOS ESPECÍFICOS

Conocer los conceptos asociados a los Sistemas Distribuidos, incluyendo la problemática de comunicación y sincronización de procesos, así como las bases de virtualización y contenedores.

Comprender los desafíos de seguridad y administración en entornos distribuidos, considerando la autenticación, autorización y cifrado de datos en diferentes niveles de implementación.

Aplicar los conceptos adquiridos en el desarrollo de soluciones para casos reales, empleando herramientas modernas de virtualización y contenedorización que permitan el despliegue eficiente de servicios y aplicaciones distribuidas.

Familiarizar al estudiante con criterios y prácticas de medición y mejora de desempeño en sistemas distribuidos, integrando métricas de escalabilidad, latencia, throughput y disponibilidad.

Analizar plataformas de Cloud Computing para comprender su implicancia en el ámbito universitario y profesional, reconociendo los distintos modelos de negocio (IaaS, PaaS, SaaS) y su impacto en el diseño de soluciones.

Aprender a desplegar arquitecturas elásticas y escalables, aprovechando estrategias DevOps para la automatización de tareas y la integración y entrega continua (CI/CD) de aplicaciones distribuidas.

Mostrar la importancia y la innovación de la Computación de Altas Prestaciones (HPC), abarcando arquitecturas paralelas y técnicas de programación de alto rendimiento.

Entender los conceptos de la Computación Paralela y Distribuida, explorando paradigmas y estándares de programación adecuados para optimizar el uso de recursos en sistemas multiprocesadores y multinúcleo.

Presentar técnicas y criterios para desarrollar software paralelo, enfatizando la transformación de algoritmos secuenciales en versiones altamente escalables y eficientes.

Destacar el empleo de estrategias en diferentes plataformas de ejecución paralela y distribuida, para que el estudiante pueda seleccionar las tecnologías y metodologías más adecuadas según los requerimientos de cada proyecto.

CONTENIDOS

Unidad 1. Introducción y comunicación en Sistemas Distribuidos .

1.1 Conceptos. Características y objetivos. Paradigmas de aplicaciones distribuidas. Un modelo de computación distribuida

1.2 Ventajas y Desventajas de los Sistemas Distribuidos frente a los Centralizados y a los de Computadoras independientes. Sistemas Totalmente Distribuidos. Sistemas de Multiprocesador con Tiempo Compartido.

1.3 Hardware de Sistemas Distribuidos. Multiprocesadores con Base en Buses. Multiprocesadores con Conmutador. Multicomputadoras con Base en Buses Multicomputadoras con Conmutador. Memoria compartida distribuida.

1.4 Conceptos de Software de base asociado a un Sistema Distribuido. Sistemas de Redes. NFS: Network File System

1.5 Características del Diseño. Transparencia. Flexibilidad. Confiabilidad. Desempeño. Escalabilidad.

1.6 Introducción a la Comunicación en los Sistemas Distribuidos. Protocolos con Capas. Modelo Cliente - Servidor. Direccionamiento. Características de Bloqueo en C/S

1.7 Migración de procesos. Uso de Buffer en C/S. Primitivas Confiables y No Confiables

1.8 Llamado a un Procedimiento Remoto (RPC). Operación. Transferencia de Parámetros. Conexión Dinámica. Casos de RPC en Presencia de Fallos. Pérdida del Servidor, pérdida de mensajes, fallo del Servidor, fallo del cliente. Protocolos RPC. Reconocimientos. Ruta Crítica. Copiado. Uso de Cronómetro

Unidad 2. Sincronización y Procesos en Sistemas Distribuidos.

2.1 Introducción a la Sincronización en Sistemas Distribuidos. Sincronización de Relojes (Lógicos - Físicos). Algoritmos para la Sincronización de Relojes.

2.2 Exclusión Mutua Algoritmos Centralizado, Distribuido, Token Ring. Transacciones Atómicas. El Modelo de Transacción, Control de Concurrencia en el Modelo de Transacción

2.3 Bloqueos en Sistemas Distribuidos. Detección Distribuida de Bloqueos. Prevención Distribuida de bloqueos.

2.4 Introducción a los Hilos (Threads). Empleo y diseño de un Paquete de Hilos. Hilos y RPC. Modelos de Sistemas. El Modelo de Estación de Trabajo. Arreglo de Procesadores. Asignación de Procesadores.

2.5 Modelos de Asignación. Conceptos de Diseño de Algoritmos de Asignación de Procesadores y su implementación. Ejemplos. Planificación en Sistemas Distribuidos

Unidad 3. Paradigmas de Sistemas Distribuidos.

3.1 Nociones de Clustering, Grid, Peer-to-peer (P2P), Cloud Computing. Principios, comparación y aplicaciones bajo los modelos mencionados.

3.2 Grid Computing (Computación de Grilla o Malla). Objetivos y características. Arquitectura computacional del Grid. Beneficios. Planificación. Gestores locales de Recursos. Middleware Grid. Grid con capas (Única vs. Múltiples). Monitoreo. Gestores de recursos/colas.

3.3 Computación P2P. Definición de P2P. Historia y evolución. Estado actual. Red Estructurada/Desestructurada. Principios de arquitectura de red, diseño e implementación. Características principales: autoorganización, comportamiento simultáneo como clientes y servidores, administración distribuida, compartimiento de recursos, balanceo de carga y replicación, etc.

3.4 Casos de estudio en los diferentes paradigmas.

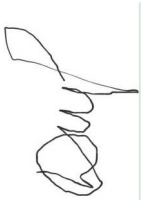
Unidad 4. Virtualización Avanzada, Contenedores y Orquestación / Modelo de Cloud Computing.

4.1 Introducción. Definiciones y conceptos relacionados con la Computación en la Nube. Historia y evolución. Análisis FODA de la implantación del Cloud Computing.

4.2 Virtualización como modelo de evolución tecnológica: Evolución del diseño del Datacenter. Elementos virtualizados (Servidores, Almacenamiento, Redes). Administración de recursos. Hipervisores. Consideraciones para proveedores de servicio Cloud. Implementación de virtualización a diferentes niveles (hardware, escritorio, SO, aplicación). Clusters virtuales y optimización de recursos.

4.3 Contenedores vs. Máquinas Virtuales. Conceptos de "sandboxing" y aislamiento de recursos a nivel de sistema operativo. Fundamentos de Docker: Imágenes y contenedores. Dockerfile y sus instrucciones básicas. Repositorios de contenedores (Docker Hub, GitHub Container Registry, etc.). Uso de docker-compose para la orquestación de servicios simples. Ventajas y casos de uso de contenedores frente a la virtualización tradicional. Escenarios de despliegue en nubes públicas y privadas.

4.4 Orquestación de contenedores con Kubernetes: Arquitectura básica (Master, Workers, Pods, Services, Deployments, ReplicaSets). Gestión de volúmenes, configuraciones (ConfigMaps, Secrets) y políticas de red. Implementación de escalabilidad (autoscaling) y tolerancia a fallos (self-healing). Balanceo de cargas y actualizaciones continuas (rolling updates). Integración con entornos de nube (AWS EKS, Google GKE, Azure AKS). Buenas prácticas de seguridad en contenedores y Kubernetes (control de acceso, namespaces, políticas de red, escaneo de vulnerabilidades).



4.5 Características de la Nube. Tipos de nube (Privada, Pública, Híbrida). Servicios de Cloud Computing (IaaS, PaaS, SaaS). Elasticidad, flexibilidad, multi-tenant, pedido en base a demanda, acceso ubicuo, métricas de uso dinámicas, auto organización, etc.

4.6 Modelo de negocio Cloud Computing. Servicios y Contratos. Ley de Protección de Datos (LOPD). Seguridad, Privacidad e Interoperabilidad entre servicios de nube. Seguridad en Cloud Computing. Recomendaciones para una arquitectura segura. Autenticación, autorización y accounting. Protección del sistema host.

4.7 Plataformas más conocidas: Google App Engine (GAE), Amazon Web Services (AWS) y Amazon EC2, Microsoft Azure. Almacenamiento de objetos en la nube (Blobs, tablas, colas, archivos). Bases de datos SQL/NoSQL. Uso avanzado de JSON y REST.

Unidad 5. Despliegue Continuo (CI/CD) y Prácticas DevOps

5.1 Principios de DevOps. Cultura de colaboración y automatización.

5.2 Flujos de trabajo con Git. Estrategias de branching y versionado de código.

5.3 Herramientas de CI/CD (Jenkins, GitLab CI, GitHub Actions, etc.).

5.4 Pipelines de despliegue para aplicaciones contenedorizadas. Automatización de pruebas y construcción de imágenes Docker. Integración con repositorios de contenedores. Despliegues automatizados sobre Kubernetes (helm, kustomize, etc.).

5.5 Observabilidad: logging, métricas y monitoreo en entornos distribuidos (Prometheus, Grafana, ELK).

5.6 Infraestructura como Código (IaC): definición y manejo de entornos con Ansible, Terraform u otras herramientas.

5.7 Mejores prácticas para la entrega continua (Continuous Delivery) y la liberación continua (Continuous Deployment).

Unidad 6. Introducción Computación Paralela.

6.1 Fundamentos para el empleo de Máquinas paralelas. Limitaciones de las computadoras secuenciales. Necesidades de memoria o de velocidad de cómputo. Razones económicas.

6.2 Dificultades físicas: integración de componentes, disipación de calor, cuello de botella en acceso a datos. Dificultades lógicas: compiladores y entornos de programación para sistemas paralelos.
6.3 Paralelismo en computadores secuenciales. Segmentación encauzada, Tubería (Pipeline). Paralelismo a nivel de instrucción (ILP). Ejecución fuera de orden, especulación. Procesadores vectoriales. Coprocesadores de E/S. Ley de Moore en procesadores Intel.
6.4 Requerimientos de Sistemas paralelos. Modelo MIMD. Modelo SPMD (Single Program Multiple Data).Sistemas multinúcleo. Sistemas de memoria compartida y programación con Threads.
6.5 Soluciones basadas en varios procesadores simples del tipo FPGA (Field Programmable Gate Array) o DSP (Digital Signal Processors). Procesadores gráficos GPGPU (General Processing Graphical Processing Unit).
6.6 Supercomputación. Supercomputadores con multicores o clusters de ordenadores. Top500.

Unidad 7: Programación Paralela.
7.1 Paradigmas de programación: Modelo de Memoria Compartida (Shared Memory Model). Sistemas de memoria virtual compartida. Modelo de Paso de Mensajes. Entornos de programación de Paso de Mensajes (PVM, BSP) y el estándar MPI.
7.2 Programación OpenMP: programación paralela en entornos multiprocesador con memoria compartida. Conceptos teóricos de la programación en OpenMP, y su entorno computacional de desarrollo de aplicaciones.
7.3 Programación MPI (Message Passing Interface): una interface de paso de mensajes para entornos paralelos con memoria distribuida. Llamadas a rutinas de biblioteca. Conceptos teóricos de la programación en MPI, y su entorno computacional para el desarrollo de aplicaciones.
7.4 Programación Híbrida: comunicación de información común por medio de memoria compartida y con memoria distribuida, fusión de MPI con OpenMP. Ejemplos de programación MPI+OpenMP en sistemas híbridos.
7.5 Esquemas Algorítmicos Paralelos: Paralelismo, Particionado de Datos, Esquemas Paralelos en Árbol, Computación Pipeline, Esquema Maestro/Esclavo, Granja de Trabajadores, Bolsa de tareas, Computación Síncrona.
7.6 Programación con Shaders. Conceptos básicos de shaders (Pixel Shaders, Vertex Shaders). Pipeline de renderizado en GPU y su relación con la programación paralela. Uso de frameworks (p. ej., WebGL, GLSL) para tareas de procesamiento gráfico y postprocesamiento.
7.7 CUDA Compute Unified Device Architecture (Arquitectura Unificada de Dispositivos de Cómputo); compilador y herramientas de desarrollo de Nvidia. Algoritmos en GPU (G8X y siguientes) de Nvidia. Ventajas de las GPU frente a las CPU de propósito general utilizando el paralelismo y lanzamiento de hilos simultáneos.

METODOLOGÍA

Clases teóricas:

Las clases teóricas se imparten de forma expositiva, utilizando presentaciones de computadora, proyector y pizarra. En cada sesión se introducen y analizan los conceptos que se van a desarrollar, se explican los fundamentos teóricos y se concluye con una síntesis de los puntos clave. Además, se indica cuáles serán las actividades prácticas relacionadas, de manera que el estudiantado pueda comprender cómo se aplicarán los contenidos abordados.

Clases prácticas:
A lo largo de la asignatura, se proponen cinco trabajos prácticos diseñados para la ejercitación y la comprensión aplicada de los temas vistos en clase. Estos trabajos, enfocados en la implementación de programas y la resolución de problemas, presentan un nivel de dificultad creciente que permite afianzar progresivamente los conceptos estudiados. Se organizan en grupos para fomentar la colaboración y el aprendizaje colectivo, y en aquellos puntos más complejos se llevan a cabo puestas en común con todo el grupo, con el fin de acompañar y guiar en la resolución de dificultades. A continuación, se describen brevemente las actividades prácticas.

Actividades Prácticas (Trabajos Prácticos):

Trabajo Práctico Nº1: "Introducción a los Sistemas Distribuidos"
Requisito de entrega: Sí, obligatorio
El objetivo de este trabajo es llevar a la práctica los conceptos básicos de los sistemas distribuidos, la comunicación asincrónica y el modelo cliente-servidor.
Se desarrollarán ejercicios para implementar un servidor TCP y un cliente que intercambien mensajes de forma robusta, considerando reconexiones y manejo de errores. Se avanzará hacia un nodo híbrido capaz de actuar simultáneamente como cliente y servidor, utilizando JSON para la comunicación. Finalmente, se desarrollará un sistema de inscripción de nodos en una red distribuida, coordinado por un nodo central que regula la ventana de tiempo para la inscripción y el intercambio de mensajes. Se trabajará con contenedores Docker y pipelines CI/CD para el despliegue automatizado en la nube

Trabajo Práctico Nº 2: "Sistemas Distribuidos y Concurrencia"
Requisito de entrega: Sí, obligatorio
En este trabajo se profundiza en la arquitectura y comunicación en sistemas distribuidos, abarcando desde el uso de contenedores Docker hasta la implementación de un servidor HTTP que permita ejecutar tareas remotas a través de un servicio contenerizado. Se introducen conceptos avanzados de concurrencia y manejo de tareas genéricas mediante contenedores Docker, con un enfoque en la automatización del despliegue a través de pipelines CI/CD (GitHub Actions). Además, se desarrolla un sistema de endpoints públicos accesibles desde Internet, proporcionando métricas de estado y logs detallados tanto en memoria como en disco.

Trabajo Práctico Nº 3: "Computación en la Nube (Kubernetes / RabbitMQ)"
Requisito de entrega: Sí, obligatorio
Este trabajo práctico aborda la construcción de una arquitectura híbrida y elástica en la nube utilizando Kubernetes y RabbitMQ. Se parte del desarrollo de un filtro de imagen distribuido (operador de Sobel) que procesa imágenes de forma paralela, asignando tareas a nodos de trabajo distribuidos. Posteriormente, se extiende la solución con una infraestructura escalable mediante Kubernetes (GKE) y Terraform, implementando pipelines CI/CD que automatizan la construcción, despliegue y escalabilidad de los servicios. Se analizan métricas de rendimiento bajo diferentes cargas de trabajo, evaluando la capacidad de adaptación y eficiencia de la arquitectura propuesta.

Trabajo Práctico Nº 4: "Programación Paralela (Shaders / CUDA) "
Requisito de entrega: No, optativo
El objetivo de este trabajo es llevar a la práctica los conceptos de programación paralela utilizando tanto shaders gráficos (a través de Shadertoy) como GPU CUDA para tareas de computación intensiva. Se

desarrollarán ejemplos prácticos de filtros gráficos, efectos visuales interactivos y procesamiento paralelo de imágenes.
Esta práctica busca integrar conceptos de programación gráfica y paralela con un enfoque en el uso efectivo del hardware disponible**.**
Trabajo Práctico Final: "Blockchain Distribuida y Programación CUDA"
Requisito de entrega: Sí, integrador y obligatorio
El proyecto integrador propone el desarrollo de una blockchain distribuida con una arquitectura basada en Kubernetes y servicios distribuidos. Los estudiantes implementarán un sistema de minería basado en el algoritmo Proof of Work (PoW) utilizando GPU CUDA para el cálculo criptográfico. La arquitectura se complementa con el uso de RabbitMQ para la distribución de tareas y Redis para la persistencia de datos. Se debe desplegar la solución en un entorno escalable en la nube, aplicando pipelines CI/CD para la automatización del despliegue y pruebas de rendimiento bajo diferentes escenarios de carga. El trabajo finaliza con un análisis detallado del rendimiento y la escalabilidad de la red blockchain desarrollada.

REQUISITOS DE APROBACIÓN Y CRITERIOS DE CALIFICACIÓN:

CONDICIONES PARA PROMOVER (SIN EL REQUISITO DE EXAMEN FINAL) DE ACUERDO AL ART.23 DEL RÉGIMEN GENERAL DE ESTUDIOS RESHCS 261-21 y su ANEXO PARA CARRERAS CON MODALIDAD PEDAGÓGICA A DISTANCIA

a) Tener aprobadas las actividades correlativas al finalizar el turno de examen extraordinario de ese cuatrimestre.

b) Cumplir con un mínimo del 80 % de asistencia para las actividades teóricas y prácticas. c) Aprobar las evaluaciones previstas en este programa, en carácter de trabajos prácticos aplicados, con una nota superior a seis (6) puntos sin haber recuperado ninguno de ellos. d) Aprobar una evaluación integradora de la asignatura con calificación no inferior a siete (7) puntos.

CONDICIONES PARA APROBAR COMO REGULAR (CON REQUISITO DE EXAMEN FINAL) DE ACUERDO AL ART.24 DEL RÉGIMEN GENERAL DE ESTUDIOS RESHCS 261-21 y su ANEXO PARA CARRERAS CON MODALIDAD PEDAGÓGICA A DISTANCIA

a) estar en condición de regular en las actividades correlativas al momento de su inscripción al cursado de la asignatura.

b) Cumplir con un mínimo del 80 % de asistencia para las actividades teóricas y prácticas. c) Aprobar las evaluaciones previstas en este programa, en carácter de trabajos prácticos aplicados, con una nota **no inferior** a cuatro (4) puntos con posibilidad de recuperar hasta un 40% de los mismos por ausencias o aplazos.

EXÁMENES PARA ESTUDIANTES EN CONDICIÓN DE LIBRES

1. Para aquellos estudiantes que, habiéndose inscriptos oportunamente en la presente actividad hayan quedado en condición de libres por aplicación de los artículos 22, 25, 27, 29 o 32 del Régimen General de Estudios, SI podrán rendir en tal condición la presente actividad.
2. Para aquellos estudiantes que no cursaron la asignatura y se presenten en condición de alumnos libres en la Carrera, por aplicación de los artículos 10 o 19 del Régimen General de Estudios, SI podrán rendir en tal condición la presente actividad.

Para ambos casos, los estudiantes deberán presentarse con los trabajos prácticos aprobados y con el trabajo final integrador, que deberá ser aprobado con nota de 4 o superior.

BIBLIOGRAFÍA

BIBLIOGRAFÍA OBLIGATORIA

Aspectos generales: (Arquitectura, Sistemas Operativos, Bases de Datos)

● *Patterson, D. A., & Hennessy, J. L. (2020). Computer Organization and Design RISC-V Edition: The Hardware/Software Interface (2nd ed.). Morgan Kaufmann.*

● *Stallings, W. (2020). Computer Organization and Architecture (11th ed.). Pearson.* ● *Silberschatz, A., Gagne, G., & Galvin, P. (2021). Operating System Concepts (11th ed.). Wiley.* ● *Elmasri, R., & Navathe, S. (2021). Fundamentals of Database Systems (8th ed.). Pearson.* ● *Chang, K. (2020). The future of RISC-V: A new generation in open hardware. IEEE Micro, 40(4), 85–88.*

Sistemas Distribuidos:

● *Van Steen, M., & Tanenbaum, A. S. (2020). Distributed Systems (3rd ed.). CreateSpace.* ● *Ghosh, S. (2022). Distributed Systems: An Algorithmic Approach (3rd ed.). CRC Press.* ● *Kshemkalyani, A. D., & Singhal, M. (2020). Commentary on 25 Years of the Ricart–Agrawala Algorithm. Distributed Computing, 33(4), 257–259.*

● *Verma, R., & Shukla, V. (2021). Edge-Fog-Cloud computing in IoT: Architecture, technologies, and insights into distributed systems. International Journal of Communication Systems, 34(5), e4737.*

● *Al-Jaroodi, J., Mohamed, N., & Eid, M. (2021). A Survey on Data Management in Fog Computing. IEEE Internet of Things Journal, 8(12), 9478–9496.*

Cloud Computing:

● *Baun, C., & Kunze, M. (2020). Cloud Computing Basics: A Non-Technical Introduction. Springer.* ● *Adhikari, M., & Amgoth, T. (2021). Adaptive compute resource allocation in the cloud computing environment: A survey. Journal of Network and Computer Applications, 174, 102906.*

● *Mustafa, A. S., & Al-Dubai, A. (2021). A comprehensive review of cloud computing: challenges and opportunities. Journal of Cloud Computing, 10(1), 1–29.*

● *Villari, M., Fazio, M., Dustdar, S., Rana, O. F., & Ranjan, R. (Eds.). (2020). OCC: Open Challenges in Cloud Computing. Springer.*

● *Xiang, T., & Tzeng, W. G. (2022). Privacy-preserving computations in cloud computing: An overview. IEEE Transactions on Cloud Computing, 10(3), 1589–1602.*

Virtualización Avanzada, Contenedores y Orquestación / Modelo de Cloud Computing.

● *Hightower, K., Burns, B., & Beda, J. (2022). Kubernetes: Up and Running (3rd ed.). O'Reilly Media.*

● *Turnbull, J. (2021). The Docker Book: Containerization is the New Virtualization (3rd ed.). Turnbull Press.*

● *Burns, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2020). Design Patterns for Container-Based Distributed Systems. Communications of the ACM, 63(8), 62–69.*

● *Pahl, C., & Banerjee, P. (2021). Containerization and the PaaS cloud. Future Generation Computer Systems, 117, 185–195.*

● *Wettinger, J., Breitenbücher, U., et al. (2021). Declarative orchestration of containerized microservices using TOSCA-based model-driven patterns. Software: Practice and Experience, 51(6), 1301–1322.*

Despliegue Continuo (CI/CD) y Prácticas DevOps

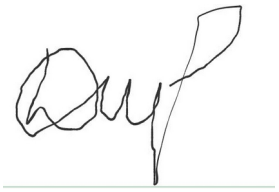
● *Kim, G., Willis, J., Debois, P., & Humble, J. (2021). The DevOps Handbook (2nd ed.). IT Revolution Press.*

● *Gruver, G. (2020). Starting and Scaling DevOps in the Enterprise. IT Revolution Press.* ● *Forsgren, N., Humble, J., & Kim, G. (2021). Accelerate: The Science of Lean Software and DevOps. IT Revolution Press (Expanded Edition).*

● *Rajasekar, K., Morshedlou, N., & Westerlund, M. (2022). Continuous Integration, Delivery, and Deployment: A Systematic Mapping Study. Journal of Systems and Software, 183, 111116.* ● *Thönes, J.*

(2021). Microservices and DevOps in practice: A systematic mapping study. Software: Practice and Experience, 51(3), 411–429.
Programación Paralela, Shaders y CUDA:
● *Kirk, D. B., & Wen-mei, W. H. (2022). Programming Massively Parallel Processors: A Hands-on Approach (4th ed.). Morgan Kaufmann.*
● *Mittal, S. (2020). A Survey of Techniques for Managing and Leveraging Caches in GPUs. ACM Computing Surveys (CSUR), 53(3), 1–37.*
● *Xavier, D. M., & de Azevedo, R. M. (2021). Evaluation of CUDA-Aware MPI Performance on HPC Systems. IEEE Transactions on Parallel and Distributed Systems, 32(7), 1703–1713.* ● *Sellers, G., & Wright, R. S. (2021). OpenGL SuperBible (8th ed.). Addison-Wesley.* ● *Lichtenbelt, D., Sellers, G., & Kessenich, J. (2020). Evolution of shading languages for GPUs. Computer Graphics Forum, 39(2), 677–694.*

DISPOSICIÓN DE APROBACIÓN: PCDD



Dr. David PETROCELLI
Prof. Responsable

Hoja de firmas

Sistema: sudocu
Fecha: 08/08/2025 12:46:13
Cargado por: ENZO EMMANUEL FERRARI

Sistema: sudocu
Fecha: 08/08/2025 13:52:19
Autorizado por: JIMENA MAZIERES