Carrera: Ingeniería en Sistemas de Información

Asignatura: Desarrollo y Operaciones DevOps

Planificación a partir del Ciclo Lectivo 2025

1. Datos administrativos de la asignatura

Nivel en la carrera	4	Duración	Cuatrimestral
Plan	2023		
Bloque curricular:	Asignatura Electiva		
Carga horaria presencial semanal (hs. cátedra):	6 horas cátedra	Carga Horaria total (hs. reloj):	72 horas
Carga horaria no presencial semanal (hs. reloj) (si correspondiese)	No corresponde	% horas no presenciales (hs. reloj) (si correspondiese)	No corresponde

2. Presentación, Fundamentación

En orden de complementar la formación de Ingenieros en Sistemas de Información, es necesario aplicar conocimientos y herramientas para el desarrollo y despliegue de software. Partiendo de los conocimientos adquiridos a lo largo de la carrera con respecto al diseño, desarrollo y sistemas operativos, es necesario la incorporación del estudio y aplicación práctica de operaciones tales como la Integración Continua, Despliegue Continuo, verificación automatizada de seguridad de desarrollo y el despliegue y mantenimiento de aplicaciones en el Cloud.

Esta asignatura contribuye al perfil del egresado:

- Interpretación y solución de problemas utilizando técnicas y herramientas para el despliegue de sistemas distribuidos.
- Toma de decisiones de infraestructura, escalabilidad y costos de sistemas de software contemplando requerimientos funcionales y no funcionales.
- Aplicación de las mejores prácticas en el desarrollo de software seguro.
- Control de sistemas de información mediante la automatización y monitoreo de ejecución y infraestructura.

Relación de la asignatura con los alcances del título:

- Formar un criterio común para la toma de decisiones estratégicas de una organización para el desarrollo de software.

- Evaluar herramientas existentes y mejorar su respuesta ante el cambio de condiciones de ejecución.
- Evaluar y seleccionar las herramientas de automatización e infraestructura más adecuadas para problemáticas concretas de sistemas distribuidos.

3. Relación de la asignatura con las competencias de egreso de la carrera

En la tabla siguiente se establece la relación de la asignatura con las competencias de egreso: Específicas, Genéricas Tecnológicas y Genéricas Sociales, Políticas y Actitudinales de la carrera. Se incluyen las competencias de egreso a las que tributa, aportes reales y significativos de la asignatura, y en qué nivel (no aporta, bajo, medio, alto).

Competencias	Nivel
Competencias genéricas tecnológicas (CG):	
CG.1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.	No aporta
CG.2. Concepción, diseño y desarrollo de proyectos de Ingeniería en Sistemas de Información/Informática	No aporta
CG.3. Gestión, planificación, ejecución y control de proyectos de Ingeniería en Sistemas de Información/Informática.	No aporta
CG.4. Utilización de técnicas y herramientas de aplicación de Ingeniería en Sistemas de Información/Informática.	No aporta
CG.5. Generación de desarrollos tecnológicos y/o innovaciones tecnológicas.	No aporta
Competencias genéricas sociales, políticas y actitudinales (CG)	
CG.6. Fundamentos para el desempeño en equipos de trabajo.	No aporta
CG.7. Fundamentos para una comunicación efectiva.	No aporta
CG.8. Fundamentos para una actuación profesional ética y responsable.	No aporta
CG.9. Fundamentos para evaluar y actuar en relación con el impacto social de su actividad profesional en el contexto global y local.	No aporta
CG.10. Aprender en forma continua y autónoma.	Medio
CG.11. Fundamentos para el desarrollo de una actitud profesional emprendedora	No aporta
Competencias Específicas de la carrera	

CE1.1. Especificar, proyectar y desarrollar sistemas de información para concebir soluciones tecnológicas que permitan resolver situaciones en las organizaciones mediante el empleo de metodologías de sistemas y tecnologías asociadas a los sistemas de información.	No aporta
CE1.2. Especificar, proyectar y desarrollar sistemas de comunicación de datos, evaluando posibles soluciones tecnológicas disponibles para dar soporte a los sistemas de información en lo referido al procesamiento y comunicación de datos.	No aporta
CE1.3. Especificar, proyectar y desarrollar software para la elaboración de soluciones informáticas con el propósito de resolver problemas estratégicos y operativos, así como de servicios y de negocios, en el marco de una actividad económica que sea social y ambientalmente sustentable.	Bajo
CE2.1. Proyectar y dirigir lo referido a seguridad informática para seleccionar y aplicar técnicas, herramientas, métodos y normas, garantizando la seguridad y privacidad de la información procesada y generada por los sistemas de información.	No aporta
CE.3.1. Establecer métricas y normas de calidad de software para medir, evaluar, controlar y monitorear el rendimiento, impulsando mejoras de acuerdo a técnicas y normas vigentes definidas por los organismos de estandarización.	No aporta
CE.4.1. Certificar el funcionamiento, condición de uso o estado de sistemas de información, sistemas de comunicación de datos, software, seguridad informática y calidad de software para asegurar la generación de los resultados deseados en función de restricciones de tiempo y recursos establecidos.	No aporta
CE.5.1. Dirigir y controlar la implementación, operación y mantenimiento de sistemas de información, sistemas de comunicación de datos, software, seguridad informática y calidad de software, a los fines de alcanzar los objetivos fijados por la organización.	Alto
CE.6.1. Asesorar y capacitar a organizaciones, empresas, organismos públicos o privados en la adquisición, instalación y uso, en lo que respecta a sistemas de información, sistemas de comunicación de datos, software, seguridad informática y calidad de software, a los fines de un uso correcto de los sistemas intervinientes.	No aporta
CE.7.1. Realizar pericias, tasaciones y arbitrajes relacionados con su actividad profesional, respetando marcos normativos y jurídicos con el objeto de asesorar a las partes o a los tribunales de Justicia.	No aporta

4. Contenidos Mínimos

No corresponde

5. Objetivos establecidos en el DC

- No corresponde

6. Resultados de aprendizaje

Los siguientes resultados de aprendizaje se promueven en el desarrollo de la asignatura

Identificador de RA	Redacción
RA1	Construir un proceso de despliegue automatizado de un sistema de información, para asegurar la efectividad, uso de recursos y la generación de métricas de validación, contemplando las mejores prácticas de la industria y mantenibilidad de infraestructura como código.
RA2	Comprobar la importancia del monitoreo de logs de una aplicación para asegurar la calidad de servicio operacional, contrastando una aplicación con monitoreo de logs vs una sin monitoreo.
	Escriba el RA.

7. Relación de los RA y las competencias

En la tabla siguiente se indica con X la tributación de cada Resultado de Aprendizaje con las competencias de egreso: específicas, genéricas tecnológicas, sociales, políticas y actitudinales de la carrera.

RA	CE1.1	CE1.2	CE1.3	CE2.1	CE3.1	CE4.1	CE5.1	CE6.1	CE7.1	CG1	CG2	CG3	CG4	CG5	CG6	CG7	CG8	CG9	CG10	CG11
RA1							X												X	
RA2			X				X												X	

8. Asignaturas correlativas previas

Para cursar y rendir debe tener cursadas:

- Asignatura/s:

  Backend

  Desarrollo de Software

Para cursar y rendir debe tener aprobada:

- Asignatura/s:

  Paradigmas de Programación

  Sintaxis y Semántica de los Lenguajes

9. Asignaturas correlativas posteriores

Indicar las asignaturas correlativas posteriores:

- Asignatura/s:

  No corresponde

10. Programa analítico

Este programa analítico contempla los contenidos mínimos, previstos en el DC vigente, y aquellos que se consideran necesarios para desarrollar los resultados de aprendizaje propuestos.

Unidad Nº: 1

Título: Introducción a DevOps, Automatización y Scripting

Contenidos:

- Concepto de Devops y su relación con el desarrollo de software.
- Responsabilidades del Devops.
- Automatización con scripting en Linux y Python.
- Concepto de DevSecOps y SRE.

Carga horaria por Unidad: 10

Unidad Nº: 2

Título: Contenedores y aplicaciones dockerizadas

Contenidos:

- ¿Qué es un contenedor?
- Contenedores vs Máquinas virtuales.
- Herramientas de contenedorización. Por ejemplo Docker, Linux Container.

- Componentes/Arquitectura de una herramienta de contenedorización, por ejemplo Docker. Daemon; CLI. API. Registry. Hub de imágenes.
- Comandos de contenedores. Distribución de recursos CPU, memoria, etc. al ejecutar contenedores.
- Contenedorización de aplicaciones. Buenas prácticas. Contenedores livianos.
- Seguridad de contenedores Docker. OWASP. Docker Benchmark.
- Automatización de creación de imágenes.

Carga horaria por Unidad: 10

Unidad Nº: 3

Título: Aplicaciones con múltiples contenedores

Contenidos:

- Docker compose. Instalación. Sintaxis. YAML.
- Desplegando una aplicación localmente con docker compose.
- Desplegando una aplicación en un servidor remoto.
- Automatización del deploy en un servidor remoto con un script (Linux o python)

Carga horaria por Unidad: 10

Unidad Nº: 4

Título: CI/CD. Automatización de ciclos de despliegue

Contenidos:

- CI/CD: Integración Continua/ Entrega Continua.
- Pipelines. Stages. Jobs.
- Gitlab CI/CD. Pipelines. Componentes. Variables predefinidas. gitlab-ci.yml. Debug de pipelines.
- Jenkins. Pipelines. Componentes. Variables predefinidas. jenkinsfile. Debug de pipelines.
- Análisis estáticos del código.
- Análisis automatizados de seguridad a una aplicación. SAST.
- Creación de pipelines.

Carga horaria por Unidad: 12

Unidad Nº: 5

Título: Orquestación de Contenedores

Contenidos:

- Orquestación de contenedores.
- Docker Swarm vs Docker compose vs Kubernetes.
- Kubernetes. K8s. K3s. Arquitectura. Cluster. Nodo Maestro. Kubelet. Pod. Contenedor. kubectl.

- Kubernetes Security. OWASP Kubernetes Top Ten.
- Desplegando una aplicación dada en Kubernetes localmente.
- Automatizar el deploy sobre Kubernetes con un script (Linux o python).

Carga horaria por Unidad: 10

Unidad Nº: 6

Título: Monitoreo y Logging

Contenidos:

- Monitoreo de aplicaciones
- Herramientas de monitoreo de código abierto
- Implementación de una arquitectura de monitoreo con herramientas de monitoreo de código abierto de una aplicación dada.

Carga horaria por Unidad: 10

Unidad Nº: 7

Título: Infraestructura como servicio (IaaS) – Infraestructura como Código (IaC)

Contenidos:

- Infraestructura como servicio (IaaS).
- Proveedores de IaaS más conocidos. Por ejemplo Amazon Web Services (AWS). Google Cloud Platform (GCP). Microsoft Azure. Openstack.
- Herramientas para crear recursos de Infraestructura como Código (IaC). Ej.: Terraform
- Caso práctico: IaC + IaaS. Ej.: Terraform + AWS

Carga horaria por Unidad: 10

Carga horaria por tipo de formación práctica de toda la asignatura

Tipo de formación práctica	**Horas reloj**
Formación experimental	22
Análisis y resolución de problemas de ingeniería y estudios de casos	30
Formulación, análisis y desarrollo de proyectos.	20

Bibliografía Obligatoria:

Mikael Krief (2022). Learning DevOps, 2nd Edition. Packt

John Culkin, Mike Zazon (2021) AWS Cookbook: Recipes for Success on AWS. Oreilly

Martyn Coupland (2021). DevOps Adoption Strategies. Packt

Bibliografía optativa y otros materiales a utilizar en la asignatura:

Gene Kim y otros (2021). The DevOps Handbook. IT Revolution.

11. Metodología de enseñanza

- Lección magistral participativa de conceptos involucrados en la filosofía de DevOps.
- Clases prácticas tipo taller para que el alumno afiance los conceptos involucrados.
- Resolución de problemas. Se presentarán desafíos, la mayoría no evaluables, para incentivar al alumno al autoaprendizaje y autodescubrimiento de conceptos relacionados.
- Estudio de casos con conferencias donde se incluirán experiencias concretas de sistemas reales, con disertantes invitados.

12. Recomendaciones para el estudio

Se recomienda que los alumnos repasen los conceptos teóricos de la clase anterior para lograr un aprendizaje iterativo.

También se sugiere remitirse a la documentación oficial de las herramientas utilizadas durante el cursado, ya que según la versión utlizada en cada computadora puede o no haber diferencias en alguna solución aplicada en clases.

Es recomendable tener a mano la bibliografía utilizada por la cátedra para quitarse dudas rápidamente de conceptos teóricos.

Se sugiere al estudiante llevar a cabo los desafíos prácticos no evaluables para afianzar conocimientos.

13. Metodología de evaluación

El modelo de enseñanza basado en competencias implica la aplicación de metodologías e instrumentos de evaluación que permiten conocer, a docentes y estudiantes, el nivel de desarrollo de las competencias que aborda la asignatura.

- Instancia de evaluación individual tipo cuestionario
- Trabajo práctico integrador que se desarrollará a lo largo de toda la materia.
- Instancias de revisión para el Trabajo Práctico Integrador. Cada presentación será evaluada desde la fundamentación de las decisiones tomadas por el grupo de alumnos y por el avance de su proyecto programático.

- Un desafío practico evaluable.

Es requisito obligatorio la presentación del trabajo práctico integrador con las siguientes condiciones:

• Código fuente y documentos involucrados accesible en un repositorio remoto.

• Incluir documento de Threat Model de la aplicación.

• Presentación de avances y demo en clase.

A continuación, se detallan todos los Resultados de Aprendizajes con sus contenidos a desarrollar para alcanzarlos, la mediación pedagógica, metodologías y estrategias de evaluación, tiempo en horas reloj.

Resultados de Aprendizaje	Contenidos según programa	Mediación Pedagógica	Metodología y Estrategias de Evaluación	Tiempos en hora reloj
RA 1	- Concepto de Devops y su relación con el desarrollo de software. - Responsabilidades del Devops. - Automatización con scripting en Linux y Python. - Concepto de DevSecOps y SRE. - ¿Qué es un contenedor? - Contenedores vs Máquinas virtuales. - Herramientas de contenedorización. Por ejemplo Docker, Linux Container. - Componentes/Arquitectura de una herramienta de contenedorización, por ejemplo Docker. Daemon; CLI. API. Registry. Hub de imágenes. - Comandos de contenedores. Distribución de recursos CPU,	Estrategia del docente: - Presentación conceptual de elementos de automatización involucrados en el despliegue de un sistema y estudio de casos. - Alcance comparativo entre diseños de despliegue. - Presentación de problemas que la automatización de infraestructura resuelve. Actividades del estudiante: - Desafios de scripting enfocados a la	Instrumentos: Exposición del trabajo práctico grupal. Instancia de evaluación. Resolución de un desafío práctico de diseño de despliegue. Criterios: Participa en el trabajo integrador grupal y las presentaciones de avance. Construye un proceso de despliegue mediante el diseño de despliegue, herramientas de CI/CD y usa monitoreo de logs como parámetros de medición.	Total hrs presenciales teóricas/prácticas: 15 Total hrs de práctica áulica: 15 Total hrs extra áulicas: 5

	memoria, etc. al ejecutar contenedores. - Contenedorización de aplicaciones. Buenas prácticas. Contenedores livianos. - Seguridad de contenedores Docker. OWASP. Docker Benchmark. - Automatización de creación de imágenes. - Docker compose. Instalación. Sintaxis. YAML. - Desplegando una aplicación localmente con docker compose. - Desplegando una aplicación en un servidor remoto. Automatización del deploy en un servidor remoto con un script (Linux o python) - CI/CD: Integración Continua/ Entrega Continua.- Pipelines. Stages. Jobs. - Gitlab CI/CD. Pipelines. Componentes. Variables	automatización de tareas. - Construcción de un proceso de despliegue automatizado usando la práctica sobre scripting y herramientas de CI/CD. - Clases práticas tipo taller asociando conceptos teóricos. - Los estudiantes presentarán avances de sus respectivos TPIs en fechas acordadas. - En todas las clases se dará espacio para dudas/consultas relacionados a los TPIs para permitir el avance de los mismos.		

	predefinidas. gitlab-ci.yml. Debug de pipelines. - Jenkins. Pipelines. Componentes. Variables predefinidas. jenkinsfile. Debug de pipelines. - Análisis estáticos del código. - Análisis automatizados de seguridad a una aplicación. SAST. - Creación de pipelines.			
RA 1	- Orquestación de contenedores. - Docker Swarm vs Docker compose vs Kubernetes. - Kubernetes. K8s. K3s. Arquitectura. Cluster. Nodo Maestro. Kubelet. Pod. Contenedor. kubectl.	Estrategia del docente: - Presentación conceptual de elementos constitutivos de orquestración de contenedores. - Clases prácticas de orquestación de contenedores, tipo taller, asociando conceptos teóricos. - Caracterización de evolución de procesamiento y escalabilidad en la	Instrumentos: Exposición del trabajo práctico grupal. Instancia de evaluación. Resolución de un desafío práctico de diseño de despliegue. Criterios: Participa en el trabajo integrador grupal y las presentaciones de avance, demostrando la capacidad de crear recursos escalables mediante exposición de	Total hrs presenciales teóricas/prácticas: 5 Total hrs de práctica áulica: 5 Total hrs extra áulicas: 5

	- Kubernetes Security. OWASP Kubernetes Top Ten. - Desplegando una aplicación dada en Kubernetes localmente. - Automatizar el deploy sobre Kubernetes con un script (Linux o python). - Infraestructura como servicio (IaaS). - Proveedores de IaaS más conocidos. Por ejemplo Amazon Web Services (AWS). Google Cloud Platform (GCP). Microsoft Azure. Openstack. - Herramientas para crear recursos de Infraestructura como Código (IaC). Ej.: Terraform	orquestación de contenedores. - Presentación de conceptos de IaaS + IaC Actividades del alumno: - Desafío de creación de recursos escalables usando infraestructura como código y k8s. - Taller práctico de IaaS + IaC: que integre la creación y el versionado de al menos un recurso básico en una IaaS con infraestructura como código (IaC). Este recurso puede o no contener el orquestador de contenedores. - Desafío sobre orquestación de contenedores.	oportunidades de mejora en sus TPs.	

	- Caso práctico: IaC + IaaS. Ej.: Terraform + Proxmox.			
RA 2	- Monitoreo de aplicaciones - Herramientas de monitoreo de código abierto - Implementación de una arquitectura de monitoreo con herramientas de monitoreo de código abierto de una aplicación dada.	Estrategia del docente: - Presentación de la importancia de monitoreo en aplicaciones mediante el estudio de casos. - Clases práticas tipo taller. Actividades del alumno: - Los estudiantes presentarán avances de sus respectivos TPIs en fechas acordadas. - Práctica sobre herramientas de monitoreo con Elastic y Kibana, contrastando la misma aplicación con y sin monitoreo de logs.	Instrumentos: Exposición del trabajo práctico grupal. Instancia de evaluación. Resolución de un desafío práctico de diseño de despliegue. Criterios: Participa en el trabajo integrador grupal y las presentaciones de avance. Justifica las decisiones del diseño de despliegue mediante el uso de análisis de logs, comprobando la importancia de la incorporación de monitoreo de aplicaciones y la creación de dashboards.	Total hrs presenciales teóricas/prácticas: 10 Total hrs de práctica áulica: 10 Total hrs extra áulicas: 2

14. Condiciones de aprobación

Las condiciones mínimas para acceder a la regularización de la materia serán:

- Aprobar trabajo práctico integral con nota no menor a 4.
- Aprobar desafío práctico con nota no menor a 4.
- Aprobar parcial teórico/práctico con nota no menor a 4, con posibilidad de recuperar en caso de aplazo.

El estudiante en condición de regular, deberá rendir un coloquio final de la materia. Durante el periodo de un ciclo lectivo, podrá rendir sin control de correlativas aprobadas. La siguiente tabla muestra los porcentajes necesarios para las calificaciones.

Aprobación Directa:

Todo aquel estudiante que apruebe el desafío práctico, el parcial y el trabajo integrador al momento de regularizar, con una nota promedio de 7 (siete) o más, no deberá rendir coloquio final oral. La calificación será la nota registrada como Nota Final en Autogestión. El estudiante en esta condición, puede registrar su nota en examen en el plazo de un ciclo lectivo, sin control de correlativas aprobadas, y después de ello se le exigirán correlativas aprobadas.

15. Modalidad de examen

Examen Final

El alumno que haya alcanzado sólo la condición de regular, deberá rendir un coloquio final oral. El objetivo del mismo es repasar el temario visto en clase y conocer los criterios del alumno a la hora de resolver despliegues de softwares.

Todo aquel estudiante que apruebe el parcial y el trabajo integrador al momento de regularizar, con una nota promedio de 7 (siete) o más, no deberá rendir coloquio final oral.

16. Recursos necesarios

Para el correcto dictado de la materia es necesario:

- Aula de laboratorio (LABSIS) con capacidad para 15 personas.
- Máquina virtual con Ubuntu con acceso root. Ubuntu ultima versión disponible.
- Acceso a internet con ancho de banda suficiente para una Conferencia Zoom.
- Proyector y Pantalla. Si no se cuenta con Pantalla, una pared pintada de blanco.

- UV Moodle
- IDE: Visual Studio Code.
- Cuenta por alumno en Gitlab LabSis.
- Repositorios en Gitlab LabSis.
- Servidor remoto Jenkins.
- Gitlab CI/CD con runners disponibles
- Servidores remotos con acceso ssh por grupos para TPI.
- Servidor remoto (o local) con Docker Daemon.
- Docker CLI instalado que tenga comunicación con el Docker Daemon.
- Docker-compose instalado.
- Servidor remoto con Docker Registry.
- Terraform instalado.
- K3s instalado.