Universidad Nacional de La Matanza
ESCUELA DE FORMACIÓN CONTINUA

1. NOMBRE DE LA CARRERA: Licenciatura en Gestión de la Tecnología

2. NOMBRE DE LA ASIGNATURA: Arquitectura de Software

Código de la asignatura: 1319

Ciclo anual: 2024

Ciclo cuatrimestral: Segundo

Modalidad de cursada: Semi Presencial

3. CUERPO DOCENTE: Lic. Rocco Guillermo Horacio, Lic. Jorge Alberto Vitale, Lic. Luciano Tomas Verni Gargallo

Profesor/es a cargo de la asignatura: Lic. Rocco Guillermo Horacio, Lic. Jorge Alberto Vitale, Lic. Luciano Tomas Verni Gargallo

4. ASPECTOS ESPECÍFICOS

A. **Carga horaria total**: 16

B. **Carga horaria semanal**: 4

C. **Carga horaria clases teóricas**: 11

D. **Carga horaria práctica disciplinar**:0

E. **Carga horaria práctica profesional**: 0

F. **Ubicación de la asignatura en el plan de estudios**: 1er año 2do cuatrimestre

G. **Correlatividades anteriores**: **N/A**

5. PROGRAMA

A. **Fundamentación del marco referencial del programa**.

La materia tiene por objetivo que los estudiantes aprendan a planificar y gestionar proyectos software.
Conozcan cómo realizar estimaciones de tamaño, esfuerzo y costos en proyectos de software y sepan identificar y gestionar riesgos en proyectos software, entre otras cosas.

B. **Objetivos generales**.

a. Comprender los fundamentos teóricos y prácticos de la arquitectura de software, incluyendo su importancia en el desarrollo de sistemas informáticos.

b. Analizar y aplicar diferentes patrones de diseño de software para la creación de arquitecturas robustas, flexibles y mantenibles.

c. Identificar y evaluar diferentes estilos arquitectónicos, comprendiendo sus ventajas, desventajas y casos de uso apropiados.

d. Desarrollar habilidades para evaluar la calidad de una arquitectura de software, considerando aspectos como la escalabilidad, la seguridad, el rendimiento y la mantenibilidad.

e. Utilizar herramientas y tecnologías relevantes para el diseño y modelado de arquitecturas de software, así como para el análisis y la evaluación de sistemas existentes.

f. Aplicar los conocimientos adquiridos en la asignatura a través de proyectos prácticos, donde se diseñen y evalúen arquitecturas de software en contextos reales.

g. Fomentar el pensamiento crítico y la capacidad de análisis en la resolución de problemas relacionados con la arquitectura de software.

h. Promover la comunicación efectiva y el trabajo en equipo en el desarrollo y análisis de arquitecturas de software.

C. Objetivos específicos.

a. Identificar y describir los principales roles y responsabilidades de un arquitecto de software en el proceso de desarrollo de sistemas informáticos.

b. Diferenciar entre arquitectura de software y diseño de software, comprendiendo cómo se complementan y difieren entre sí.

c. Aplicar los principios de modularidad, cohesión y acoplamiento para diseñar componentes de software reutilizables y mantenibles.

d. Analizar y aplicar patrones de diseño específicos, como MVC, en el diseño de sistemas software de diferentes complejidades.

e. Comparar y contrastar diferentes estilos arquitectónicos, como cliente-servidor y orientado a servicios, en términos de sus ventajas y desventajas.

f. Evaluar la calidad de una arquitectura de software mediante técnicas como revisión de código, análisis de dependencias y pruebas de rendimiento.

g. Utilizar herramientas de modelado arquitectónico, como UML, para representar y comunicar la estructura y el comportamiento de un sistema software.

h. Implementar arquitecturas de software utilizando frameworks y tecnologías modernas, como contenedores Docker y microservicios.

i. Diseñar y desarrollar soluciones software escalables, seguras y eficientes que cumplan con los requisitos funcionales y no funcionales del cliente.

j. Colaborar de manera efectiva en equipos multidisciplinarios para el diseño, implementación y evaluación de arquitecturas de software complejas.

k. Presentar y defender propuestas de arquitectura ante audiencias técnicas y no técnicas, comunicando de manera clara y persuasiva los aspectos clave del diseño.

l. Reflexionar críticamente sobre las decisiones de diseño tomadas durante el proceso de desarrollo, identificando lecciones aprendidas y áreas de mejora.

D. Unidades didácticas.

Unidad N°1.

Fundamentos de Arquitectura de Software

- Objetivos:
  - Comprender los conceptos fundamentales de la arquitectura de software.
  - Analizar la importancia de la arquitectura en el desarrollo de sistemas informáticos.
- Contenidos:
  - Introducción a la arquitectura de software.
  - Roles y responsabilidades del arquitecto de software.
  - Principios de diseño de arquitecturas de software.
- Actividades:
  - Conferencias magistrales sobre los conceptos fundamentales.
  - Debates en clase sobre la importancia de la arquitectura de software.
  - Ejercicios prácticos para aplicar los principios de diseño.

Unidad N°2

Business Model Canvas (BMC) en la Arquitectura de Software

- Objetivos:
  - Comprender los fundamentos del Business Model Canvas (BMC) y su aplicación en el contexto de la arquitectura de software.
  - Analizar cómo el BMC puede ser utilizado para definir y validar modelos de negocio en proyectos de desarrollo de software.

- Contenidos:
    - Introducción al Business Model Canvas (BMC): concepto, componentes y metodología de uso.
    - Integración del BMC en el proceso de diseño de arquitecturas de software.
    - Casos de estudio de empresas que han utilizado el BMC en el desarrollo de productos de software.
- Actividades:
    - Presentaciones sobre los conceptos básicos del BMC y su relación con la arquitectura de software.
    - Talleres prácticos de aplicación del BMC en proyectos de desarrollo de software.
    - Análisis de casos de estudio para comprender cómo el BMC puede guiar el diseño de arquitecturas de software centradas en el cliente y en el valor agregado.

Unidad N°3

Patrones de Diseño

- Objetivos:
    - Identificar y aplicar diferentes patrones de diseño de software.
    - Comprender cómo los patrones de diseño contribuyen a la creación de arquitecturas robustas.
- Contenidos:
    - Patrón Modelo-Vista-Controlador (MVC).
    - Patrón de Capas.
    - Patrón de Arquitectura de Microservicios.
- Actividades:
    - Presentaciones sobre los patrones de diseño.
    - Ejemplos prácticos de implementación de patrones en proyectos reales.
    - Desarrollo de ejercicios prácticos para aplicar los patrones de diseño.

Unidad N°4

Estilos Arquitectónicos

- **Objetivos:**
    - Identificar y comparar diferentes estilos arquitectónicos.

    - Comprender las ventajas y desventajas de cada estilo arquitectónico.
- **Contenidos:**
    - Arquitectura Cliente-Servidor.
    - Arquitectura Orientada a Servicios (SOA).
    - Arquitectura basada en Eventos.
- **Actividades:**
    - Conferencias sobre los diferentes estilos arquitectónicos.
    - Análisis de casos de estudio para entender la aplicación de cada estilo.

Debate en clase sobre los pros y los contras de cada estilo

Unidad N°5

Evaluación de la Arquitectura

- **Objetivos:**
    - Evaluar la calidad de una arquitectura de software.
    - Aplicar técnicas de evaluación para mejorar las arquitecturas existentes.
- **Contenidos:**
    - Calidad en la arquitectura de software.
    - Técnicas de evaluación de arquitecturas.
- **Actividades:**
    - Sesiones prácticas de revisión y análisis de arquitecturas existentes.
    - Desarrollo de proyectos de evaluación de la arquitectura.
    - Presentación de casos de estudio sobre la mejora de arquitecturas existentes.

Unidad N°6

Herramientas y Tecnologías

- **Objetivos:**
    - Utilizar herramientas y tecnologías para el diseño de arquitecturas de software.
    - Aplicar UML y otras herramientas de modelado arquitectónico.
- **Contenidos:**
    - Uso de UML en el modelado de arquitecturas.
    - Herramientas de modelado arquitectónico.

    - Frameworks y tecnologías modernas para el desarrollo de arquitecturas de software.
- **Actividades:**
    - Talleres prácticos sobre el uso de herramientas de modelado.
    - Demostraciones de frameworks y tecnologías modernas.
    - Desarrollo de proyectos utilizando las herramientas y tecnologías presentadas.

Unidad N°7

Proyecto Final

- **Objetivos:**
    - Aplicar los conocimientos adquiridos en el diseño y desarrollo de un proyecto práctico.
    - Presentar y defender el proyecto final ante el grupo.
- **Contenidos:**
    - Desarrollo de un proyecto práctico que integre los conceptos aprendidos en la asignatura.
- **Actividades:**
    - Desarrollo y seguimiento del proyecto final.
    - Sesiones de tutoría para resolver dudas y brindar orientación.
    - Presentación y defensa del proyecto final ante el grupo.

ACERCA DE LOS TRABAJOS PRACTICOS.

Para los trabajos prácticos se tomará, como referencia para la confección, el marco estructural del trabajo final de la carrera, confeccionando una empresa ficticia en el cual tendrá un organigrama con las funciones de cada uno de los integrantes del grupo. Los TP´s deberán incluir los ítems que se detallan debajo.

Desarrollo / Contenido de la carpeta:

1.- Tema propuesto

2.- Visión

3.- Misión

4.- Objetivos (generales/específicos)

5.- Planteo/justificación de la problemática (el porqué de la necesidad del sistema - vender la solución)

6.- Modelo de Negocio Canvas

7.- GANTT - cronograma (presentar tiempo del desarrollo)

8.- Delimitación del problema (Hasta donde contempla la solución del problema o la app)

9.- Factibilidad.

10.- Relevamiento

11.- Graficas a consideración (UML - Caso de uso, diagrama de funciones/relaciones, DER, etc)

12.- Arquitectura del sistema

13.- Contingencia/DR- Disaster Recovery Plan (si lo requiere)

14.- Matriz FODA

15.- Análisis de costos VAN/TIR (optativo si el proyecto lo amerita)

16.- Mockup (Una maqueta que sugiere cómo se verá el diseño final y, por lo general, se comparte con los clientes y las partes interesadas. Un prototipo)

17.- Conclusión

- Presentación de una carpeta en formato digital.
- Exposición del proyecto de forma oral.

E. Bibliografía general.

INGENIERÍA DEL SOFTWARE

UN ENFOQUE PRÁCTICO

Quinta edición

Roger S. Pressman.

6. METODOLOGÍA

A. Previsiones metodológicas y pedagógicas:

- Los mapas conceptuales
- La elaboración de estrategias de resolución de problemas
- La lluvia de ideas
- La construcción de gráficos, cuadros
- Los juegos de roles.
- Modelo de negocio CANVAS

1. **Clases Teóricas:**

- Presentación de los conceptos fundamentales de cada unidad.
- Explicación de principios, patrones y estilos arquitectónicos mediante ejemplos prácticos.
- Uso de presentaciones visuales, ejercicios interactivos y estudios de casos para facilitar la comprensión.
- Fomento del debate y la participación activa de los estudiantes mediante preguntas abiertas y discusiones en clase.

2. **Clases Prácticas:**
    - Desarrollo de ejercicios prácticos para aplicar los conceptos teóricos aprendidos.
    - Utilización de herramientas de modelado y frameworks de desarrollo para la implementación de soluciones prácticas.
    - Trabajo en proyectos grupales donde los estudiantes puedan colaborar y aplicar sus conocimientos en situaciones reales.
3. **Estudios de Casos:**
    - Análisis de casos reales de arquitecturas de software en diferentes industrias y contextos.
    - Discusión sobre los desafíos encontrados, las decisiones de diseño tomadas y las lecciones aprendidas.
4. **Tutorías y Asesoramiento:**
    - Sesiones de tutoría para resolver dudas, revisar progresos y brindar orientación individualizada a los estudiantes.
    - Disponibilidad para consultas fuera del horario de clases a través de correo electrónico u otras plataformas de comunicación.
5. **Evaluación Continua:**
    - Evaluación continua del progreso de los estudiantes a través de tareas, ejercicios prácticos y participación en clase.
    - Realización de pruebas de conocimiento para evaluar la comprensión de los conceptos teóricos.
    - Evaluación de proyectos prácticos que reflejen la aplicación de los conocimientos adquiridos en situaciones reales.
6. **Feedback Constructivo:**
    - Proporcionar retroalimentación regular y constructiva sobre el desempeño de los estudiantes en sus actividades y proyectos.

- Identificar áreas de mejora y sugerir recursos adicionales para el aprendizaje autodirigido.

Esta metodología combina tanto aspectos teóricos como prácticos para garantizar una experiencia de aprendizaje completa y efectiva en el campo de la arquitectura de software.

B. Actividades que se desarrollarán de acuerdo a la modalidad y articulación de las mismas en caso de corresponder:

Clase de 2 hs (aprox.) presencial y/o virtual en caso de aplicar cada una clase en particular.

Dependiendo del temario en curso, en cada clase (virtual o presencial), se dictará un tema nuevo o se continuará con el tema propuesto en la clase anterior dependiendo de la extensión de la unidad. En la mayor parte de la clase se hará una exposición de la teoría y se mostraran ejemplos. Dependiendo del contenido que se dicte en el día, éstas se complementarán con sesiones de práctica. Se realizarán también trabajos de investigación individuales y/o grupales sobre aspectos relacionados con los temas que abarca el curso.

Seguimiento offline y apoyo extra en casos particulares de al menos 2 horas semanales.

C. Implementación de herramientas digitales: (detalle de plataformas virtuales y modalidad de aplicación de las mismas)

N/A. En este caso al ser una materia presencial toda aplicación de la teoría será explicada y/o proyectada en el aula.

7. MECANISMOS DE SEGUIMIENTO, SUPERVISIÓN Y EVALUACIÓN DE LAS ACTIVIDADES, PRESENCIALES Y/O DE SEGUIMIENTO VIRTUAL

1. **Asistencia y Participación en Clase:**
   - Llevar un registro de la asistencia y participación de los estudiantes durante las clases presenciales y virtuales.
   - Fomentar la participación mediante preguntas, discusiones en grupo y actividades prácticas.
2. **Entrega de Tareas y Ejercicios:**
   - Establecer fechas límite para la entrega de tareas y ejercicios prácticos.
   - Utilizar plataformas virtuales de gestión de aprendizaje para recibir y evaluar las entregas de los estudiantes.
3. **Pruebas de Conocimiento:**
   - Realizar pruebas periódicas para evaluar la comprensión de los conceptos teóricos por parte de los estudiantes.

- Ofrecer pruebas en línea mediante herramientas de evaluación virtual para evaluar el aprendizaje a distancia.

4. **Revisiones de Proyectos Prácticos:**
    - Programar sesiones de revisión de proyectos prácticos donde los estudiantes puedan presentar y discutir sus avances.
    - Proporcionar retroalimentación detallada sobre el diseño y la implementación de los proyectos.
5. **Sesiones de Tutoría Individualizada:**
    - Ofrecer sesiones de tutoría individualizada para discutir el progreso académico y resolver dudas específicas.
    - Establecer horarios de consulta tanto presenciales como virtuales para garantizar la disponibilidad para todos los estudiantes.
6. **Encuestas de Retroalimentación:**
    - Realizar encuestas anónimas de retroalimentación para recopilar opiniones y sugerencias de los estudiantes sobre la calidad del curso y las actividades desarrolladas.
    - Utilizar los resultados de las encuestas para realizar ajustes y mejoras en el diseño y la implementación del curso.
7. **Seguimiento de Participación en Plataformas Virtuales:**
    - Monitorizar la participación de los estudiantes en las plataformas virtuales de aprendizaje, como foros de discusión y salas de chat.
    - Incentivar la interacción y el intercambio de ideas entre los estudiantes a través de estas plataformas.
8. **Evaluación de la Calidad de los Trabajos:**
    - Establecer criterios claros de evaluación para las tareas, proyectos y participaciones en clase.
    - Utilizar rúbricas y matrices de evaluación para proporcionar retroalimentación específica y transparente sobre el desempeño de los estudiantes.

Al combinar estos mecanismos de seguimiento, supervisión y evaluación, podrás obtener una visión completa del progreso y desempeño de tus estudiantes tanto en actividades presenciales como virtuales en el curso de Arquitectura de Software. Recuerda adaptar estos mecanismos según las necesidades y características específicas de tu programa académico y tus estudiantes.

8. CRONOGRAMA DE CLASES Y EXÁMENES

Contenidos/ Actividades/ Evaluaciones	Semanas															
N° de unidades / Parciales	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Unidad 1	X															
Unidad 2		X	X													
Unidad 3			X	X	X											
Unidad 4						X										
Parcial 1							X									
Unidad 4								X	X							
Unidad 5									X	X						
Unidad 6										X	X					
Unidad 7											X	X				
Parcial 2													X			
Recuperatorio 1 y 2														X		

9. CONDICIONES GENERALES PARA LA APROBACIÓN DE LA ASIGNATURA

A. **Asistencia:** Se requiere una asistencia a clases no menor al setenta y cinco (75%) sobre el total de la carga horaria de la asignatura.

B. **Evaluación**:

Se disponen de cuatro estados académicos posibles:

- Ausente: cuando el alumno no tenga calificación en alguno de sus exámenes (o su recuperatorio).
- Reprobada: cuando el alumno obtenga como calificación final de 1 a 3 puntos.
- Cursada: cuando el alumno obtenga entre 4 y 6 puntos como calificación final.
- Promocionada: cuando el alumno obtenga como calificación final entre 7 y 10 puntos.

Para las asignaturas cuatrimestrales habrá 2 instancias parciales y la posibilidad de 1 instancia recuperatoria. La calificación obtenida en el examen recuperatorio reemplaza y anula a todos los efectos, la obtenida en el examen parcial que se recupera.

A los fines de conformar la calificación final, los parciales no se promedian, salvo que ambas evaluaciones sean reprobadas, o ambas cursadas, o ambas promocionadas.

El alumno que culmine la materia en condición "cursada", deberá aprobar el examen final para tener la asignatura como aprobada.

Rocco, Guillermo

FIRMA Y ACLARACIÓN DEL DOCENTE/S A CARGO