4133 - LENGUAJE DE PROGRAMACIÓN II

I - Datos de identificación de la asignatura

Carrera:	Licenciatura en Análisis de Sistemas		
Código:	4133	**Plan:**	2024
Denominación:	Lenguaje de programación II		
Área:	Tecnologías aplicadas		
Año:	Tercero		
Horas con acompañamiento docente (HTD), semanal			4
Horas de Trabajo Independiente del estudiante (HTI), semanal			8
Horas semanales (HS)			12
Cantidad de sesiones			32
Total Horas de Trabajo con el docente (THTD)			128
THD teóricas	0	**THD prácticas**	128
Total de Horas de Trabajo Independiente del estudiante (THTI)			256
Total Horas Académicas (THA)			384
Crédito académico (CA)			15,4
Pre-requisito:	Lenguaje de programación I		

II - Fundamentación

La asignatura tiene como objetivo principal brindar a los estudiantes un conjunto avanzado de conocimientos y habilidades en programación, centrándose en diferentes situaciones problemáticas de la vida real.

En esta asignatura, los estudiantes se enfrentarán a diversos problemas reales que requieren soluciones de calidad, eficiencia y corrección. A través del análisis y diseño de algoritmos, los estudiantes aprenderán a aplicar estrategias de solución que cumplan con los requisitos y especificaciones planteadas. La implementación de estas soluciones se realizará utilizando lenguajes de programación apropiados y considerando los diferentes paradigmas de programación.

El enfoque práctico de la asignatura permitirá a los estudiantes familiarizarse con los aspectos fundamentales de la programación y desarrollar habilidades para abordar problemas complejos. A medida que trabajen en diferentes proyectos y ejercicios prácticos, los estudiantes adquirirán experiencia en el desarrollo de soluciones eficientes y correctas, aplicando conceptos como estructuras de datos, algoritmos avanzados, programación orientada a objetos u otros paradigmas de programación relevantes para su formación.

Además, la asignatura fomentará el desarrollo de habilidades de diseño de software, como modularidad, la reutilización de código y la creación de interfaces claras y coherentes. Los estudiantes aprenderán a realizar pruebas y depuración de sus programas, así como a documentar adecuadamente el código desarrollado.

La naturaleza teórica-práctica de la asignatura garantiza que los estudiantes adquieran tanto los fundamentos teóricos como las habilidades prácticas necesarias para enfrentar los desafíos reales en el campo de la programación. A través de la resolución de problemas y la implementación de soluciones, los estudiantes fortalecerán su capacidad

para abordar situaciones complejas y adquirirán una comprensión profunda de los conceptos básicos de la disciplina.

La asignatura busca proporcionar a los estudiantes un conjunto avanzado de conocimientos y habilidades en programación. A través del trabajo con situaciones problemáticas de la vida real y la aplicación de estrategias de solución, los estudiantes desarrollarán su capacidad para crear soluciones de calidad, eficiencia y corrección. La asignatura se enfoca en diferentes paradigmas de programación y fomenta el desarrollo de habilidades de diseño de software. Su naturaleza teórica-práctica garantiza una formación integral que prepara a los estudiantes para enfrentar los desafíos reales en el campo de la programación.

III - Competencias a desarrollar

Competencias genéricas

1. Trabajar en equipos multidisciplinarios.
2. Identificar, plantear y resolver problemas.
3. Identificar, analizar, abstraer, formular y resolver problemas relacionados con sus áreas de competencia.
4. Diseñar, programar, ejecutar, analizar e interpretar resultados de pruebas realizadas en sus áreas de competencia.

Competencias específicas

1. Aplicar adecuadamente herramientas y técnicas de mapeo objeto-relacional, para representar y vincular objetos de software con estructuras de bases de datos relacionales, facilitando la persistencia y recuperación de datos en aplicaciones modernas.
2. Definir y comunicar con claridad la arquitectura de un sistema de software, identificando sus componentes principales e interdependencias, para facilitar su comprensión, documentación y desarrollo colaborativo.
3. Utilizar eficazmente herramientas de control de versiones, para gestionar el código fuente y otros recursos del proyecto, promoviendo el trabajo colaborativo y la trazabilidad en entornos de desarrollo compartido.
4. Diseñar y generar con precisión informes y reportes funcionales, empleando herramientas especializadas, para visualizar y comunicar datos de forma efectiva en el contexto del desarrollo de software.
5. Analizar con pensamiento crítico problemas complejos del desarrollo de sistemas, aplicando técnicas de resolución y estrategias de diseño, para proponer soluciones eficientes y sostenibles en diferentes contextos.
6. Planificar y gestionar con buenas prácticas proyectos de desarrollo de software, siguiendo las etapas del ciclo de vida y controlando recursos, tiempos y alcance, para garantizar el cumplimiento de los objetivos del proyecto.
7. Implementar con eficiencia sistemas de software, aplicando metodologías de desarrollo orientadas a sistemas de gestión, para desarrollar soluciones funcionales, reutilizables y mantenibles, aplicando buenas prácticas de codificación.
8. Diseñar e implementar de forma modular sistemas utilizando principios de la programación orientada a objetos, aplicando técnicas como encapsulamiento, herencia y polimorfismo, para construir software robusto, escalable y flexible.

9. Elaborar documentación técnica clara y estructurada, centrada en aspectos de codificación, instalación, configuración y operación del sistema desarrollado, para facilitar su comprensión, uso y mantenimiento por parte de usuarios y desarrolladores.

IV - Cuerpo de conocimientos

Unidad 1: Mapeo relacional de objetos.
Contenidos:

- Estrategias de persistencia, como RDBMS y su papel en la ejecución del sistema.
- Las estructuras de datos y objetos en tiempo de ejecución y su almacenamiento en la memoria volátil
- Herramientas O/RM como puente entre contextos de ejecución volátiles y contextos de persistencia
- Uso y familiaridad con al menos una implementación O/RM
- Integración y configuración de herramientas O/RM.

Unidad 2: Principios de la orientación a objetos en el diseño e implementación de sistemas/software.
Contenidos:

- Diseño para modularidad para reutilización
- Aplicación de los principios SOLID

Unidad 3: Arquitectura del sistema de software con enfoque en la visualización y comprensión de sus principales componentes y dependencias.
Contenidos:

- Seleccionar y aplicar patrones de diseño básicos.
- Seleccionar y aplicar patrones arquitectónicos.
- Aplicar los principios SOLID para la gestión de complejidad.

Unidad 4: Herramientas para el control de versiones de código y recursos.
Contenidos:

- Sistemas de control de versiones (SCV).
- Ramificación y fusión para la evolución de características, gestión de defectos y coordinación de equipos.
- Uso de etiquetas para alinear los cambios de código con las características.
- El control de versiones como un sistema de transacciones con propiedades ACID
- Uso y familiaridad con al menos un SCV como Git, SVN, Mercurial, CVS, etc.

Unidad 5: Herramientas para el diseño de reportes.
Contenidos:

- Herramientas para diseño de reportes.
- Estrategias para generación de reportes.
- Reportes complejos: reportes maestros y subreportes, tablas, listas y otros.
- Integración de reportes.

Unidad 6: Identificación del problema y planteamiento de soluciones desarrollando sistemas.
Contenidos:

- Identificación de la problemática
- Justificación del sistema

Unidad 7: Planificación, gestión, control y administración del proyecto basado en el ciclo de vida del software y con énfasis en la programación.
Contenidos:

- Definición y selección del Proyecto
- Presentación del anteproyecto de sistema
- Aprobación del anteproyecto

Unidad 8: Codificación de un sistema de acuerdo con las técnicas y metodologías de desarrollo de sistema de gestión.

Contenidos:

- Interpretación e implementación de los diagramas del sistema
- Definición de la arquitectura del sistema
- Diseño de las interfaces gráficas del sistema
- Codificación de algoritmos, procesos y reportes
- Test de software de los códigos
- Empaquetado y distribución de software

Unidad 9: Documentaciones técnicas del proyecto.

Contenidos:

- Elaboración de la documentación técnica del proyecto de software, fundamentalmente las referidas a la programación y operación.

V - Estrategias didácticas a ser implementadas en el proceso de enseñanza aprendizaje. (abarcando actividades de formación e investigación)

En las clases se articula fuertemente la teoría y la práctica con un abordaje integrador del contenido.

La teoría consta de un repaso de los contenidos ya tratados, luego se presenta el nuevo tema con una explicación inicial o utilizando algún material didáctico que abra la discusión, donde el docente procura la participación activa de los alumnos. Se trabaja sobre un material que sirve para el análisis y reflexión del contenido. Es un disparador para la lectura y análisis de problemáticas y casos relacionados. Se trabaja sobre la participación en los foros y materiales de investigación en la plataforma.

Al final se hace una sistematización y se registran los puntos centrales que se retomarán en la siguiente clase.

La práctica acompaña la teoría, están íntimamente relacionadas. El sentido de esta actividad es la de probar, aplicar, testear lo que se vio teóricamente y proyectar posibles diseños y soluciones al uso de esa tecnología.

En la plataforma virtual los alumnos participan en foros, encuestas, suben información solicitada de casos, de investigaciones o desarrollos a realizar y sus producciones. La cátedra hace disponible las teorías, prácticos, materiales pedagógicos necesarios para el desarrollo de las distintas actividades, como también artículos científicos, casos de uso didáctico, fichas y guías de recomendación de diseños. Además, se utilizan producciones que los mismos alumnos han desarrollado.

Para poder desarrollar las habilidades y capacidades que requiere esta competencia, a los estudiantes se le presentan actividades prácticas en la que deben analizar sistemas de software y proponer su desarrollo teniendo en cuenta la incorporación de tecnologías contemporáneas. Para esto deben realizar todo el proceso ingenieril, analizando qué contenido, funciones y servicios son adecuados para el entorno de desarrollo. Realizan el proceso de selección, filtrado de la información más relevante, adaptación de las funciones e incorporación de funcionalidad nueva, haciendo uso de las potencialidades que brinda el entorno de desarrollo.

Las clases se desarrollarán por medio de tutorías y asesorías por parte del profesor. Exposición y presentación de trabajo individual y evaluaciones periódicas de los avances en el proyecto. Deben considerar qué tecnologías utilizar de acuerdo a las ventajas y desventajas de cada uno y del caso planteado. Los estudiantes deben exponer sus

soluciones, demostrar la calidad de diseño de la interfaz propuesta e indicar las decisiones tomadas en todo el proceso de desarrollo, cuya justificación y criterios empleados para su defensa, serán analizados y evaluados.

VI - Estrategias de evaluación.

A modo de ejercitación y evaluación se plantean, a lo largo de la cursada, entregas de ejercicios que los estudiantes deben desarrollar y entregar en las prácticas y teorías.

Se entrega un trabajo en distintas etapas. En cada etapa se presenta una producción que los estudiantes defienden en forma de coloquio oral. En esa instancia, además, se indaga sobre los conceptos teóricos vistos en esta etapa del desarrollo. Esto es un requisito para la aprobación de la cursada. Estas entregas son de seguimiento y de evaluación con calificación.

Al finalizar existe una instancia de evaluación final donde los estudiantes exponen, en forma completa, el trabajo realizado. Esto se realiza en coloquios en los cuales deben exponer la tarea realizada en forma individual y donde el docente evalúa no sólo los conocimientos sino la claridad de la presentación, su organización y la forma de expresión.

Toda evaluación realizada a los estudiantes queda plasmada en una planilla muy bien detallada, donde se indican los resultados de las diferentes evaluaciones realizada a los mismos: capacidad del estudiante para desarrollar su aprendizaje, claridad de las presentaciones realizadas, forma de organización y expresión en las diferentes instancias de evaluación oral, formulación de la solución de los diferentes desafíos en forma autónoma, entre otros.

Para la obtención de calificaciones parciales y finales se tendrá en cuenta el Reglamento Académico de la universidad.

VII - Actividades de extensión y de responsabilidad social universitaria.

Rige de acuerdo al reglamento de la Universidad y el reglamento interno de la facultad.

VIII - Fuentes bibliográficas

Básica

- Jacobson, Ivar, “El Proceso Unificado de Desarrollo de Software” / Ivar Jacobson, Grady Booh, James Rumbaugh. - - España: Addison Wesley, 2000.
- Pressman, Roger. “Ingeniería de Software. Un enfoque practico” / Roger Pressman. - - España: McGraw-Hill Interamericana, 2002.
- Somerville, Ian “Ingeniería de Software” / Ian Somerville. - México: Pearson Educación, 2005.
- Hernández, Jordi. “Ingeniería de Software en entornos de Software Libre” / Jordi Hernández, David Mejias, David Aycart Perez. - - 2da. Ed. - - España: Eureka Media SL Catalunya, 2009
- Goodrich, M. T., Tamassia, M. H. (2014). Data structures and algorithms in Java - 4th Edition. John Wiley & Sons.
- Sznajdleder, P. (2018). Java a fondo:-estudio del lenguaje y desarrollo de aplicaciones. Alfaomega Grupo Editor.

Complementaria

- Jaramillo Valbuena, S. et.al. Programación Avanzada en Java. Ediciones Elizcom: Armendia.
- Eckstein, R. (2017). Java SE Application Design With MVC. Recuperado de https://www.oracle.com/technical-resources/articles/javase/mvc.html
- Grenyer, P. (2008). Model View Controller with Java Swing. Recuperado de https://accu.org/journals/overload/16/88/grenyer_1524/
- Martin, Robert C. (2018). Arquitectura limpia : guía para especialistas en la estructura y el diseño de software. Anaya Multimedia.