# 4145 - ANÁLISIS Y DISEÑO DE SISTEMAS

## I - Datos de identificación de la asignatura

| | | | |
|---|---|---|---|
| **Carrera:** | Licenciatura en Análisis de Sistemas | | |
| **Código:** | 4145 | **Plan:** | 2024 |
| **Denominación:** | Análisis y diseño de sistemas | | |
| **Área:** | Fundamentos de la informática | | |
| **Año:** | Cuarto | | |
| **Horas con acompañamiento docente (HTD), semanal** | | | 4 |
| **Horas de Trabajo Independiente del estudiante (HTI), semanal** | | | 6 |
| **Horas semanales (HS)** | | | 10 |
| **Cantidad de sesiones** | | | 32 |
| **Total Horas de Trabajo con el docente (THTD)** | | | 128 |
| **THD teóricas** | 32 | **THD prácticas** | 96 |
| **Total de Horas de Trabajo Independiente del estudiante (THTI)** | | | 192 |
| **Total Horas Académicas (THA)** | | | 320 |
| **Crédito académico (CA)** | | | 12,8 |
| **Pre-requisito:** | Lenguaje de programación II, Ingeniería de software | | |



## II - Fundamentación

La asignatura de Análisis y Diseño de Sistemas es una etapa avanzada que sigue a la asignatura de Lenguaje de Programación II. En esta fase, los estudiantes aplicarán los conocimientos adquiridos anteriormente para resolver problemas de programación y llevar a cabo el desarrollo completo de un proyecto seleccionado.

El objetivo principal de esta asignatura es permitir a los estudiantes aplicar de manera práctica sus habilidades de análisis y diseño de sistemas. Se espera que los alumnos alcancen un alto nivel de calidad en la solución de problemas de programación y sean capaces de ofrecer interfaces amigables e intuitivas a los usuarios.

Durante el curso, los estudiantes aprenderán técnicas avanzadas de análisis y diseño de sistemas, centrándose en la resolución de problemas complejos. Se explorarán diferentes metodologías y enfoques para el análisis y diseño de sistemas, incluyendo la identificación de requisitos, el modelado de datos, el diseño de arquitectura y la creación de interfaces de usuario.

La asignatura se desarrolla en un entorno teórico-práctico, lo que significa que los estudiantes tendrán la oportunidad de aplicar los conceptos aprendidos en la teoría a través de ejercicios y proyectos prácticos. Trabajarán en la implementación completa de un proyecto seleccionado, lo que les permitirá consolidar su comprensión de los procesos de análisis y diseño de sistemas.

Al finalizar la asignatura, los estudiantes habrán adquirido habilidades sólidas en el análisis y diseño de sistemas, así como en la implementación de soluciones de alta calidad. Estarán capacitados para abordar problemas complejos de programación y podrán diseñar interfaces amigables para los usuarios. Además, habrán adquirido experiencia práctica en la implementación de proyectos de desarrollo de software, lo que les permitirá enfrentar desafíos reales en su carrera profesional.

## III - Competencias a desarrollar

**Competencias genéricas**

1. Liderar proyectos en el campo de la informática.
2. Integrar equipos multidisciplinarios y realizar trabajos colaborativos.
3. Analizar, delinear, gestionar, desarrollar, implementar y evaluar proyectos con innovación y creatividad.
4. Trabajar en equipos multidisciplinarios.
5. Conocer y saber aplicar técnicas y herramientas actualizadas en sus áreas de competencia.
6. Diseñar, programar, ejecutar, analizar e interpretar resultados de pruebas realizadas en sus áreas de competencia.

**Competencias específicas**

1. Realizar de manera sistemática el análisis de requisitos, identificando y organizando hechos esenciales del problema o necesidad, para orientar el diseño funcional y guiar el desarrollo de soluciones efectivas de software.
2. Seleccionar y aplicar adecuadamente una metodología de desarrollo de sistemas, adaptándola a las características y necesidades del proyecto, para asegurar la coherencia, utilidad y calidad del sistema de software.
3. Formalizar y comunicar con claridad los requisitos del sistema, utilizando lenguajes y formatos comprensibles para todas las partes interesadas, con el fin de facilitar la toma de decisiones informadas y el éxito del proyecto.
4. Especificar con precisión la arquitectura del sistema, definiendo sus componentes, interdependencias y estructura lógica, para facilitar el diseño detallado, la programación colaborativa y la escalabilidad del sistema.
5. Identificar y documentar los componentes externos, bibliotecas y dependencias utilizadas en el sistema, reconociendo su impacto en el rendimiento, la interoperabilidad y la seguridad del software.
6. Desarrollar la implementación del código fuente según la arquitectura y diseño establecidos, respetando buenas prácticas de codificación y asegurando la calidad y funcionalidad del software.
7. Trabajar de manera colaborativa en equipos de desarrollo, demostrando habilidades interpersonales, comunicación efectiva y resolución conjunta de problemas, para fomentar un entorno de desarrollo ágil y eficiente.
8. Aplicar técnicas de prueba exhaustiva a los componentes del software, verificando su funcionalidad, coherencia e integridad, para garantizar la calidad, fiabilidad y ajuste a los requisitos especificados.
9. Realizar tareas de mantenimiento evolutivo y correctivo del software, modificando y ajustando el sistema según nuevos requisitos o errores detectados, para asegurar su funcionamiento continuo y adaptabilidad.
10. Establecer una comunicación efectiva y constante con las partes interesadas del sistema, promoviendo la transparencia, el diálogo técnico y la documentación oportuna, para mantener la viabilidad, aceptación y éxito del proyecto en todas sus etapas.

## IV - Cuerpo de conocimientos

**Unidad 1:** Análisis de requisitos sistemáticos.
**Contenidos:**

- Aplicación de los principios de elicitación y documentación del proceso.
- Utilización de herramientas para los aspectos estructurales del problema.
- Utilización de herramientas para el comportamiento de los aspectos del problema.

**Unidad 2:** Metodología de proceso de sistemas de software.
**Contenidos:**

- Metodologías de proceso de software disponibles según sea apropiado para el dominio del problema y el contexto.
- Revisión y ajustes periódicos de los modelos de procesos de software en respuesta a las condiciones cambiantes.

**Unidad 3:** Formalización y comunicación de requisitos de software.
**Contenidos:**

- Trabajo dentro de un paradigma de diseño conocido para expresar requisitos.
- Mantenimiento iterativo de una especificación.
- Mejora de objetivos de incremento para el desarrollo y la validación.

**Unidad 4:** Arquitectura del sistema de software.
**Contenidos:**

- Selección y aplicación de patrones de diseño básicos
- Selección y aplicación de patrones arquitectónicos
- Aplicación los principios SOLID para la gestión de la complejidad.

**Unidad 5:** Diálogo adecuado entre las partes interesadas que asegure un grado de transparencia en la comunicación y la información para mantener la viabilidad del sistema de software.
**Contenidos:**

- Utilización de estructuras para la interacción de las partes interesadas especificadas con un modelo de proceso de software seleccionado.
- Establecimiento de metas y criterios de aceptación para cada incremento de producto.

**Unidad 6:** Trabajo en un entorno de equipo.
**Contenidos:**

- Herramientas y habilidades del entorno de equipo.
- Habilidades grupales interpersonales y desarrollo de software en equipo.
- Comunicación del equipo.
- Personalidades del equipo
- Responsabilidades del equipo
- Uso de herramientas de comunicación en equipo.
- Comunicación adecuada con los miembros del equipo.
- Software de control de versiones en entorno independiente
- Software de control de versiones en un proyecto de desarrollo en equipo

**Unidad 7:** Componentes y bibliotecas del sistema.
**Contenidos:**

- Identificación e implementación de aspectos relevantes de las bibliotecas que son endógenas al lenguaje de programación o plataforma de desarrollo.
- Identificación e implementación de aspectos relevantes de las bibliotecas que son exógenas al lenguaje de programación o plataforma de desarrollo.
- Utilización del control de versiones semántico para la especificación de dependencia.

**Unidad 8:** Codificación.
**Contenidos:**

- Selección, desarrollo y utilización de interfaces de programación de aplicaciones apropiadas
- Mantención de la integridad del código mediante la gestión del código fuente.

**Unidad 9:** Prueba de componentes del código de programación.
**Contenidos:**

- Selección de las estrategias apropiadas de prueba de unidad, aceptación e integración.
- Explicación de cómo se acumula y paga la deuda técnica.

**Unidad 10:** Mantenimiento del software a lo largo de la implementación.
**Contenidos:**

- Utilización de estrategias de implementación e integración continua.
- Utilización de un paradigma de proceso de software que sea adaptable e iterativo.

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

En las clases se articula en torno a un proyecto integrador.

Para poder desarrollar las habilidades y capacidades que requiere las competencias, 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 y codificación 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.

En cada etapa se presenta una producción que los estudiantes defienden en forma de coloquio oral. 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/