1. Datos Generales de la asignatura

Nombre de la asignatura:	Paradigmas de programación
Clave de la asignatura:	IAD-2422
SATCA[1]:	2-3-5
Carrera:	Ingeniería en Inteligencia Artificial

2. Presentación

Caracterización de la asignatura

La asignatura de paradigmas de programación en la carrera de Ingeniería en Inteligencia Artificial prepara a los estudiantes para entender, seleccionar y aplicar diferentes enfoques de programación en el desarrollo de sistemas de inteligencia artificial, brindándoles una base sólida para enfrentar los desafíos del campo en constante desarrollo y evolución.

La aportación de esta asignatura al perfil de egreso proporciona a los estudiantes un conocimiento sólido con una comprensión profunda para el desarrollo de programas informáticos empleando buenas prácticas y los conocimientos sobre paradigmas de programación utilizando diferentes enfoques, estilos y aplicaciones de programación que son relevantes para la disciplina de la inteligencia artificial.

La asignatura de paradigmas de programación es importante debido a que en la inteligencia artificial se utilizan diversas plataformas de programación donde se crean soluciones computacionales por lo que proporciona a los estudiantes las habilidades y la perspectiva necesarias para desarrollar Sistemas de IA eficientes, innovadores y adaptables en un campo en constante evolución.

Esta asignatura consiste en el estudio de diferentes enfoques o paradigmas utilizados en el desarrollo de programas y la programación de computadoras, utilizando distintas filosofías, estilos y técnicas para la escritura de programas.

En el contexto de la ingeniería en inteligencia artificial, la asignatura de paradigmas de programación se relaciona y complementa con varias otras asignaturas que son fundamentales para la formación en este campo, algunas de estas asignaturas incluyen: algoritmia y estructura de datos, programación para inteligencia artificial, programación orientada a objetos, Internet de las cosas, big data, analítica de datos, inteligencia artificial y programación lógica funcional.

[1] Sistema de Asignación y Transferencia de Créditos Académicos

Intención didáctica
La intención didáctica de la materia de paradigmas de programación es proporcionar a los estudiantes una comprensión sólida y profunda de los diferentes enfoques de programación que existen, así como desarrollar habilidades prácticas para aplicar estos paradigmas en la resolución de problemas relacionados a la programación de IA de manera efectiva. **Los contenidos** deben ser presentados de manera gradual y estructurada, comenzando con conceptos fundamentales y avanzando hacia temas más complejos. Se debe proporcionar una visión general de cada paradigma de programación, seguida de una exploración detallada de sus características, principios, técnicas asociadas y buenas prácticas de desarrollo. **La profundidad de los contenidos** debe ser suficiente para que los estudiantes adquieran una comprensión sólida de los diferentes paradigmas de programación. Se debe abordar cada paradigma en suficiente detalle para que los estudiantes puedan aplicarlo en la resolución de problemas prácticos con el uso de diferentes tecnologías y plataformas existentes. **Se deben diseñar actividades** que fomenten el pensamiento crítico, el ingenio, la resolución de problemas y la creatividad. Esto puede incluir la resolución de problemas prácticos utilizando diferentes paradigmas de programación, la participación en debates sobre las ventajas y desventajas de cada paradigma, y la creación de proyectos de software que integren múltiples enfoques con diferentes grados de complejidad. **El tratamiento de los contenidos** de la asignatura contribuye al desarrollo de competencias genéricas como el pensamiento analítico, la comunicación efectiva, el trabajo en equipo, la capacidad de aprendizaje autónomo y la toma de decisiones fundamentadas. Los estudiantes aprenden a evaluar críticamente diferentes enfoques de programación y a seleccionar el más adecuado para un problema dado, lo que les permita desarrollar habilidades transferibles a una variedad de contextos relacionados con la IA. **El docente debe** actuar como facilitador del aprendizaje, proporcionando orientación y apoyo a los estudiantes a lo largo del proceso de enseñanza-aprendizaje. Debe promover un ambiente de aprendizaje activo y participativo, fomentando la discusión y la colaboración entre los estudiantes. Además, el docente debe proporcionar retroalimentación constructiva sobre el desempeño de los estudiantes y guiarlos en el desarrollo de habilidades y competencias clave.

3. Participantes en el diseño y seguimiento curricular del programa

Lugar y fecha de elaboración o revisión	Participantes	Observaciones
Tecnológico Nacional de México del 4 al 6 de marzo de 2024	Representantes de los Institutos Tecnológicos de: Celaya, Chihuahua, Iztapalapa III, La Paz, Matehuala, Mérida, Minatitlán, Querétaro, Saltillo, Tijuana. Institutos Tecnológico Superior de Teziutlán. Tecnológico de Estudios Superiores de Ixtapaluca.	Propuesta sintética de la carrera de Ingeniería en Inteligencia Artificial
Tecnológico Nacional de México del 22 al 26 de abril de 2024	Representantes de los Institutos Tecnológicos de: Celaya, Chihuahua, Iztapalapa III, La Paz, Matehuala, Mérida, Minatitlán, Querétaro, Saltillo, Tijuana. Institutos Tecnológico Superior de Teziutlán, Tecnológico de Estudios Superiores de Ixtapaluca.	Diseño y/o desarrollo curricular de la carrera de Ingeniería en Inteligencia Artificial
Tecnológico Nacional de México del 27 al 31 de mayo del 2024	Representantes de los Institutos Tecnológicos de: Celaya, La Paz, Matehuala, Mérida, Minatitlán.	Consolidación curricular de la carrera de Ingeniería en Inteligencia Artificial

4. Competencia(s) a desarrollar

Competencia(s) específica(s) de la asignatura
El estudiante de la asignatura de Paradigmas de Programación adquiere un sólido conocimiento de los diferentes paradigmas de programación y es capaz de aplicarlos de manera efectiva para resolver una variedad de problemas de programación complejos. Además, tiene la capacidad de poder analizar críticamente los enfoques de programación y adaptarse a diferentes contextos y requerimientos de desarrollo de software con ingenio y creatividad mediante las siguientes competencias: ● Utiliza técnicas de programación orientada a objetos para modelar y resolver problemas de manera eficiente. ● Emplea conceptos de programación funcional para desarrollar soluciones robustas y escalables. ● Aplica principios de programación lógica para resolver problemas de inferencia y razonamiento. ● Implementa algoritmos de estructuras de datos y algoritmos clásicos en diversos lenguajes

de programación.
- Analiza y evalúa críticamente los diferentes paradigmas de programación en función de los requisitos específicos del proyecto.
- Adapta los paradigmas de programación a diferentes dominios de aplicación, demostrando flexibilidad y adaptabilidad.
- Diseña y desarrolla sistemas de ingeniería de software utilizando buenas prácticas con enfoques combinados de varios paradigmas de programación.

5. Competencias previas

El estudiante de la asignatura de paradigmas de programación debe tener una base sólida de conocimientos previos que incluyen:

- Comprende los fundamentos de la programación estructurada para seguir avanzando en la asignatura.
- Maneja conceptos básicos de algoritmos y estructuras de datos para asimilar los distintos paradigmas de programación.
- Aplica principios de lógica de programación para comprender la sintaxis y semántica de los diversos paradigmas.
- Conoce los conceptos básicos de la programación orientada a objetos para relacionarlos con otros paradigmas.
- Utiliza habilidades de resolución de problemas para enfrentar desafíos de implementación en diferentes paradigmas.
- Familiariza con la matemática discreta para entender los aspectos teóricos detrás de los paradigmas de programación. Tiene habilidades básicas en matemáticas aplicadas: El estudiante tiene habilidades básicas en matemáticas aplicadas, incluyendo cálculo diferencial e integral, lo que le permite comprender conceptos avanzados en algoritmos y técnicas de optimización utilizadas en Machine Learning.

6. Temario

No.	Temas	Subtemas
1	Fundamentos de los paradigmas de programación	1.1. Introducción a los paradigmas de programación. 1.1.1. Definición de paradigma de programación. 1.1.2. Evolución de los paradigmas de programación. 1.1.3. Clasificación y Características de los paradigmas de programación. 1.1.4. Importancia de los Tipos de Lenguajes de programación. 1.1.5. Análisis de los diferentes estilos de programación.

		1.1.6. Lenguajes y programación multiparadigma. 1.1.7. Aplicaciones de los Paradigmas en la Programación con base en los Criterios de desarrollo. 1.2. Programación imperativa y declarativa conceptos básicos y estructuras de control. 1.2.1. Clasificación de los Lenguajes de programación imperativa/declarativa. 1.2.2. Manejo de variables y constantes. 1.2.3. Ventajas – Desventajas de su uso. 1.2.4. Aplicación de la programación imperativa-declarativa.
2	Paradigmas avanzados de programación	2.1. Programación funcional. 2.1.1. Métodos de programación funcional. 2.1.2. Conceptos básicos de funciones y recursión. 2.1.3. Inmutabilidad y funciones de orden superior. 2.1.4. Ventajas y desventajas de la programación funcional. 2.2. Programación lógica. 2.2.1. Antecedentes de la programación lógica. 2.2.2. Lógica de primer y segundo orden. 2.2.3. Unificación y resolución de problemas mediante lógica. 2.2.4. Ventajas y desventajas de la programación lógica. 2.3. Programación concurrente y paralela. 2.3.1. Conceptos básicos de programación concurrente. 2.3.2. Técnicas y herramientas para la programación paralela. 2.3.3. Hilos, características, aplicaciones. 2.3.4. Procesos y colas de planificación. 2.3.5. Programación, diseño de algoritmos paralelos distribuidos.

3	Integración de paradigmas y aplicaciones prácticas	3.1. Evaluación y normas de paradigmas de programación. 3.1.1. Criterios de evaluación de paradigmas de programación. 3.1.2. Análisis comparativo de eficiencia y rendimiento. 3.1.3. Reglas y leyes de programación. 3.1.4. Buenas prácticas de programación. 3.1.5. ISO 25000:2005 evaluación del software. 3.1.6. ISO 9126-2001 calidad del software. 3.1.7. ISO/IEC 12207 Procesos del ciclo de vida del Software. 3.1.8. ISO/IEC 25012 calidad de datos. 3.2. Aplicaciones prácticas de paradigmas y multiparadigmas de programación. 3.2.1. Desarrollo de proyectos utilizando múltiples paradigmas y tecnologías actuales. 3.2.2. Implementación de Soluciones en diferentes dominios tanto en aplicaciones de TI como en IA.
4	Tendencias y futuros desarrollos en programación	4.1. Nuevos enfoques y paradigmas emergentes. 4.1.1. Exploración de tendencias actuales en programación centralizada y distribuida. 4.1.2. Servidores Web GPU/IA. 4.1.3. Paradigmas emergentes y su aplicabilidad centralizada-distribuida. 4.1.4. Programación de servidores en la nube para IA y automatización. 4.2. Ética y consideraciones sociales en programación. 4.2.1. Implicaciones éticas en el desarrollo de la ingeniería de software, límites y desafíos de la IA y robótica. 4.2.2. Responsabilidad del programador en IA elección de paradigmas y herramientas para el desarrollo de la IA y robótica. 4.2.3. Protección de desarrollo intelectual “IMPI”.

		4.2.4. Impacto Técnico, Social, Cultural, Medioambiental y Económico de la IA y Robótica.

7. Actividades de aprendizaje de los temas

1. Fundamentos de los paradigmas de programación	
Competencias	Actividades de aprendizaje
Específica(s): ● Comprender el concepto de paradigma de programación. ● Reconocer la importancia de los diferentes paradigmas en el desarrollo de software. *Genéricas:* ● Pensamiento crítico. ● Comunicación oral y escrita. ● Capacidad de expresión oral y escrita ● Facilidad para el trabajo en equipo.	● Lectura de textos introductorios sobre paradigmas de programación. ● Discusión en grupo sobre ejemplos de aplicaciones de diferentes paradigmas. ● Elaboración de mapas conceptuales sobre los diferentes paradigmas de programación.
2. Paradigmas avanzados de programación	
Competencias	Actividades de aprendizaje
Específica(s): ● Dominar los conceptos básicos y estructuras de control de la programación imperativa. ● Aplicar el enfoque imperativo en la resolución de problemas de programación. *Genéricas:* ● Resolución de problemas. ● Trabajo en equipo. ● Manejo de las TIC.	● Ejercicios prácticos de codificación utilizando estructuras de control. ● Desarrollo de programas simples para practicar el manejo de variables y constantes. ● Empleo de diferentes paradigmas en un mismo ejemplo. ● Empleo de Multiparadigmas en un caso práctico.

3. Integración de paradigmas y aplicaciones prácticas	
Competencias	Actividades de aprendizaje
Específica(s): ● Comprender los principios y conceptos fundamentales de la POO y Eventos. ● Diseñar y crear clases y objetos en un entorno de programación. *Genéricas:* ● Creatividad. ● Resolución de problemas. ● Gestión del tiempo.	● Implementar algoritmos de aprendizaje por refuerzo en entornos simulados, para comprender su funcionamiento y cómo se ajustan a diferentes problemas. ● Experimentar con diferentes configuraciones de algoritmos y parámetros para explorar su impacto en el rendimiento del agente de aprendizaje. ● Desarrollar un proyecto práctico que aborden desafíos del mundo real utilizando técnicas de aprendizaje por refuerzo. ● Adaptar algoritmos de aprendizaje por refuerzo. ● Participar en discusiones sobre las estrategias utilizadas, los desafíos encontrados y las soluciones propuestas.
4. Tendencias y futuros desarrollos en programación	
Competencias	Actividades de aprendizaje
Específica(s): ● Comprende los conceptos de funciones de orden superior y recursión. ● Aplica técnicas de programación funcional en la resolución de problemas. *Genéricas:* ● Pensamiento analítico. ● Aprendizaje autónomo. ● Utilización del ingenio. ● Solución de problemas.	● Investigación de normas y estándares para la ingeniería de Software. ● Investigación individual sobre paradigmas emergentes y su potencial impacto en la industria del software. ● Presentación de informes y discusión en grupo sobre los hallazgos. ● Debates y discusiones sobre dilemas éticos en el desarrollo de software. ● Análisis de casos de estudio que ilustren los impactos en diferentes ámbitos incluso el ambiental.

8. Práctica(s)

Tema 1

- Investigación de ejemplos de aplicaciones del mundo real que ilustren diferentes paradigmas de programación, como la web, la inteligencia artificial, la robótica, etc.
- Implementación de algoritmos simples (búsqueda, ordenación, etc.) utilizando estructuras de control y variables en un lenguaje de programación idóneo.
- Desarrollo de un pequeño proyecto orientado a objetos, como un sistema de gestión de biblioteca o un juego simple, utilizando clases y objetos.

Tema 2

- Implementación de funciones recursivas para resolver problemas comunes, como el cálculo de factorial, la generación de secuencias numéricas, etc., en un lenguaje funcional.
- Resolución de problemas de lógica y programación utilizando un lenguaje de programación lógica, como la implementación de un sistema experto simple o la resolución de problemas específicos.
- Desarrollo de una aplicación que utilice múltiples threads o procesos para realizar tareas concurrentes, como la descarga de archivos simultánea o la simulación de un sistema distribuido.

Tema 3

- Implementación de un proyecto que aborde un problema específico utilizando diferentes paradigmas de programación, comparando su eficiencia, mantenibilidad y legibilidad.
- Desarrollo de un proyecto multidisciplinario que involucre la integración de multiparadigmas de programación para resolver un problema complejo con planteamiento por equipo.

Tema 4

- Investigación y experimentación con un nuevo paradigma o enfoque de programación emergente, como la programación cuántica o la programación basada en redes neuronales, y la implementación de un proyecto simple para explorar su aplicabilidad.
- Debate y discusión en grupo sobre dilemas éticos en el desarrollo de software, como la privacidad de los datos, la equidad algorítmica, la responsabilidad del desarrollador, seguido de la redacción de un ensayo o artículo reflexivo sobre el tema y la protección intelectual informática.

9. Proyecto de asignatura

Desarrollar una aplicación informática que opere a través del cloud la cual mediante multiparadigmas de programación y aplicaciones que realicen una app para la búsqueda de imágenes a partir de muestras en internet.

1. Fundamentación:

Marco Teórico

- Procesamiento de Imágenes: Fundamentos teóricos sobre el procesamiento de imágenes digitales, incluyendo técnicas de análisis, reconocimiento de patrones y extracción de características.
- Inteligencia Artificial: Conceptos teóricos sobre inteligencia artificial, incluyendo aprendizaje automático, redes neuronales artificiales, y algoritmos de clasificación de imágenes.
- Computación en la Nube: Aspectos teóricos de la computación en la nube, como modelos de servicio (IaaS, PaaS, SaaS), virtualización, y escalabilidad.

Marco Conceptual

- Búsqueda de Imágenes: Conceptos relacionados con la búsqueda de imágenes en Internet, incluyendo motores de búsqueda, algoritmos de similitud, y técnicas de recuperación de información.
- Interfaz de Usuario: Principios conceptuales de diseño de interfaz de usuario para aplicaciones web, incluyendo usabilidad, accesibilidad, y diseño responsivo.
- Arquitectura de Software: Conceptos relacionados con la arquitectura de software, como arquitecturas de microservicios, contenedores, y sistemas distribuidos.

Marco Contextual

- Tecnologías Emergentes: Descripción del contexto tecnológico actual, destacando el papel de la inteligencia artificial y la computación en la nube en el desarrollo de aplicaciones modernas.
- Demanda del Mercado: Análisis del contexto de mercado, incluyendo la demanda de aplicaciones de búsqueda de imágenes y el crecimiento del uso de inteligencia artificial en diferentes industrias.
- Disponibilidad de Datos: Evaluación del contexto de disponibilidad de datos, incluyendo fuentes de imágenes disponibles en Internet y conjuntos de datos etiquetados para entrenar modelos de inteligencia artificial.

Marco Legal

- Privacidad y Protección de Datos: Consideraciones legales relacionadas con la privacidad y la protección de datos de los usuarios, incluyendo el cumplimiento de regulaciones como el GDPR y el CCPA.
- Derechos de Autor: Consideraciones legales relacionadas con los derechos de autor y la propiedad intelectual de las imágenes utilizadas en la aplicación, incluyendo licencias de uso y derechos de reproducción.
- Ética en Inteligencia Artificial: Discusión sobre consideraciones éticas en el desarrollo y uso de inteligencia artificial, incluyendo la equidad, la transparencia y el sesgo algorítmico.

2. Planeación de actividades

Definición del Alcance y Requerimientos

- Definir los objetivos del proyecto.
- Identificar los requisitos funcionales y no funcionales.
- Documentar los criterios de éxito y los entregables esperados.

Investigación y Selección de Tecnologías

- Investigar sobre tecnologías en la nube y multiparadigmas de programación.
- Evaluar y seleccionar las herramientas y servicios adecuados para el proyecto.

Diseño de la Arquitectura

- Diseñar la arquitectura de la aplicación en la nube.
- Definir la estructura de la base de datos y la interfaz de usuario para la búsqueda de imágenes.

Desarrollo de la Aplicación
- Implementar el backend de la aplicación en la nube.
- Desarrollar el frontend de la aplicación web o móvil.
- Integrar los servicios de búsqueda de imágenes y procesamiento en la nube.

Pruebas y Depuración
- Realizar pruebas unitarias y de integración.
- Depurar errores y optimizar el rendimiento de la aplicación

Implementación en la Nube y Despliegue
- Configurar el entorno de producción en la nube.
- Desplegar la aplicación en la infraestructura en la nube seleccionada.

Documentación y Entrega Final
- Elaborar la documentación técnica y de usuario.
- Preparar la presentación final del proyecto y demostrar la aplicación ante el cliente o usuarios finales.

Requerimientos

Tecnologías y Herramientas:
- Plataforma de computación en la nube, como AWS, Azure o Google Cloud Platform.
- Lenguajes de programación como Python, JavaScript (Node.js) u otros según los paradigmas seleccionados.
- Bibliotecas y frameworks para el desarrollo web o móvil, como React.js, Angular, Vue.js, etc.
- Bibliotecas de procesamiento de imágenes, como OpenCV o TensorFlow.
- Herramientas de gestión de versiones, como Git.
- Herramientas de gestión de proyectos, como Trello, Jira o Asana

Cronograma de actividades

Semanas	**Actividades**
1 a 2	Definición del Alcance y Requerimientos
3 a 4	Investigación y Selección de Tecnologías
5 a 6	Diseño de la Arquitectura
7 a 10	Desarrollo de la Aplicación
11 a 12	Pruebas y Depuración
13 a 14	Implementación en la Nube y Despliegue
15 a 16	Documentación y Entrega Final

3. Ejecución:
- Liderar y Organizar por parte del profesor módulos de programación, integración, secuencia y desarrollo del proyecto, aclarando dudas y o capacitando a el grupo
- Integrar en equipos al grupo de tal forma que estén en igualdad de integrantes
- Nombrar líderes en cada equipo en base a sus habilidades
- Asignar tareas a cada integrante donde se tenga control sobre los avances y cumplimiento
- Asignar tiempos y horarios de trabajo y resultados a cada equipo e integrante del mismo
- Verificar equipos de cómputo a utilizar y poner a tiempo
- Verificar acceso a internet y cloud con acceso gratuito o por cuenta institucional.

- Verificar equipo de red y espacios de trabajo.
- Revisar la secuencia de actividades asignadas a lideres.
- Revisar la percepción de lo que realmente se desea elaborar razonando y presentado por equipos diagramas de flujo de programación, así como las plataformas de trabajo en paradigmas o Multiparadigmas a emplear.
- Revisar constantemente el seguimiento e hitos y asesorar en el aprendizaje y evolución del avance del proyecto.
- Asegurarse de que la información esté compartida entre el grupo sobre los avances complicaciones a modo de apoyarse continuamente.
- Ensamble del producto entregable con sus respectivos informes escritos.

Evaluación:

1. Revisar la disponibilidad y seguridad de descarga del cloud.
2. Revisar su instalación en dispositivos móviles y fijos.
3. Revisar los tiempos de puesta a tiempo y ejecución.
4. De ser posible colocar en la App Store de Google.
5. Someter a la aplicación informática al objetivo primordial de búsqueda de imágenes de todo tipo similares en internet.
6. Analizar el porcentaje de éxito obtenido con diferentes tipos de imágenes y expresarlo en tablas de hojas de cálculo a modo de graficar los resultados para su análisis respectivo
7. Revisar mediante los ISOs revisados en esta cátedra si cumple para el aspecto de calidad global
8. Revisar y probar finalmente el proyecto en conjunto con los equipos para su posterior presentación en junta plenaria
9. Revisar si el producto entregable cumple las expectativas planteadas al inicio, de lo contrario revisar la organización y planificación a modo de corregir y optimizar los resultados
10. Analizar los aspectos sociales, culturales, económicos, ecológicos de la aplicación elaborando una tabla de ventajas - desventajas
11. Revisar en conjunto con el grupo documentos de actas de entrega y posible protección intelectual “IMPI”
12. Verificar la aplicación y ventajas del producto entregable a nivel empresa y social
13. Aplicaciones en el mercado laboral o de servicios con un posible plan de negocios
14. Cierre formal del proyecto
15. Respaldo del proyecto con sus manuales de programador y usuario
16. Análisis de resultados
17. Actualización del conocimiento y ciclo de vida posible del producto

10. Evaluación por competencias

Evaluación Continua

Tareas periódicas durante el semestre que evalúen diferentes aspectos de las competencias específicas y genéricas.

Ejercicios prácticos en clase y tareas para completar en casa que aborden diferentes paradigmas de programación.

Exámenes
Exámenes que incluyan preguntas teóricas y prácticas sobre los temas cubiertos en el curso, así como la aplicación de los diferentes paradigmas de programación en la resolución de problemas.

Pruebas cortas o quizzes para evaluar el conocimiento inmediato de los conceptos y habilidades fundamentales.

Proyectos y Trabajos Prácticos
Desarrollo de proyectos prácticos que integren múltiples paradigmas de programación, evaluando la capacidad de los estudiantes para aplicar los conceptos aprendidos en situaciones reales.

Entrega de trabajos escritos que analicen y comparen diferentes paradigmas de programación en función de criterios específicos, como eficiencia, legibilidad y mantenibilidad.

Evaluación de la colaboración y el trabajo en equipo en proyectos prácticos y discusiones grupales.

11. Fuentes de Información

1. Bratko, I. 2011 Prolog Programming for Artificial Intelligence Addison-WesleyLafore, R. (2002). Object-Oriented Programming in C++ (4a ed.). Sams Publishing.
2. Ford, N. 2014 Functional Thinking. Paradigm over sintax O'Reilly.
3. González, A. (2019). Comparación de Paradigmas de Programación en la Industria del Software. Revista de Informática Aplicada, 15(2), 123-135.
4. Lafore, R. (2002). Object-Oriented Programming in C++ (4a ed.). Sams Publishing.
5. Mertz, D. (2003). A Survey of Programming Paradigms. Software: Practice and Experience, 33(4), 369-395.
6. Sebesta, R. W. (2018). Concepts of Programming Languages (12a ed.). Pearson.
7. Warburton, R. 2016 Object-Oriented vs. Functional Programing. Bridging the Divide Between Opposing Paradigms O'Reilly.