|  |  |
| --- | --- |
| **Nombre Corto de la Asignatura** | **Teoría de la computación** |
| **Nombre Largo de la Asignatura** | **Teoría de la computación** |
| **Código de la asignatura** | **34801** |
| **Grado** | Pregrado |
| **Descripción** | Las tecnologías modernas de desarrollo de aplicaciones se basan en una serie de conocimientos teóricos asociados a modelos de ejecución y paradigmas de programación. Se abordan conceptos básicos que serán aplicados en cursos posteriores de la carrera, a saber: a) Programación reactiva/basada en eventos, la cual se aplica en implementación de interfaces gráficas; b) programación orientada a aspectos y mixins, usados en el desarrollo de servicios web; c) programación de flujo de datos y funcional, que se utiliza para procesamiento de datos; d) máquinas de estado, que se usa en diseño de ciertos tipos software; e) expresiones regulares, que se usa en muchos problemas de reconocimiento de "strings" con cierta estructura; f) Redes de Petri, que son el modelo de ejecución de BPMN; y, g) teoría de la computabilidad, que es la base a todo lo anterior. Este curso enseña dichos temas a través de una clases expositivas y talleres sobre dichos temas. |
| **Número de Créditos** | 2 |
| **Condiciones Académicas de Inscripción (Pre-requisitos)** | Requisitos de inscripción: Programación avanzada /y/ (Lógica y Matemáticas Discretas /o/  Matemáticas Discretas Sistemas /o/ Fundamentos de matemáticas II /o/ Matemáticas II) |
| **Período Académico de Vigencia** | 2430 |

|  |
| --- |
| **Objetivos de Formación** |
| * Presentar los conocimientos básicos sobre programación reactiva/basada en eventos. * Presentar los conocimientos básicos sobre programación orientada a aspectos y mixins * Presentar los conocimientos básicos sobre programación de flujo de datos y funcional * Presentar los conocimientos básicos sobre máquinas de estado * Presentar los conocimientos básicos sobre expresiones regulares * Presentar los conocimientos básicos sobre redes de Petri * Presentar los conocimientos básicos sobre teoría de la computabilidad, que es la base a todo lo anterior |

|  |
| --- |
| **Resultados de Aprendizaje Esperados (RAE)** |
| Al finalizar el curso el estudiante estará en capacidad de:   * Explicar los conceptos esenciales de la teoría de la computabilidad a partir de documentación en inglés (disciplinar 2, CDIO 3.3) * Desarrollar programas usando diferentes paradigmas de programación, basado en documentación en inglés (disciplinar 3, CDIO 3.3) * Desarrollar modelos para problemas que podrían ser resueltos con máquinas de estado y autómatas (disciplinar 1, CDIO 2.1) |

|  |
| --- |
| **Contenidos temáticos** |
| 1. Máquinas de estado y aplicaciones 2. Teoría de la computabilidad 3. Paradigmas de programación |

|  |
| --- |
| **Estrategias Pedagógicas** |
| Esta asignatura se centrará en la apropiación del conocimiento para la resolución de problemas relacionados con máquinas de estado, teoría de computabilidad, y paradigmas de programación. Se centra en dos estrategias: clase magistral expositiva, centrada en explicar la teoría detrás de los contenidos del curso y aprendizaje por proyectos, donde los estudiantes aplicarán dichos conceptos teóricos. |

|  |
| --- |
| **Evaluación** |
| Las estrategias de evaluación están centradas en la valoración de los resultados de aprendizaje esperado de la asignatura; las cuales pueden ser formativas que suscitan la comprensión y construcción de conocimiento, y sumativas que incluyen porcentajes de evaluación con el fin de corroborar el logro de los aprendizajes y el desarrollo de las competencias en los estudiantes.  Las estrategias de evaluación sumativas de la asignatura son:  Parcial 1. Máquinas de estado y aplicaciones. Teoría de la computabilidad (30%)  Parcial 2. Paradigmas de programación (30%)  Taller 1. Autómatas y Expresiones regulares (10%)  Taller 2. Programación funcional y lenguajes de flujo de datos (10%)  Taller 3. Programación reactiva/basada en eventos (10%)  Taller 4. Programación orientada a aspectos (10%) |

|  |
| --- |
| **Recursos Bibliográficos** |
| Básicos   * Bainomugisha, E., Carreton, A. L., Cutsem, T. van, Mostinckx, S., & Meuter, W. de. (2013). A survey on reactive programming. ACM Computing Surveys (CSUR), 45(4), 52. https://doi.org/10.1145/2501654.2501666 * Bracha, G., Cook, W., Bracha, G., & Cook, W. (1990). Mixin-based inheritance. ACM SIGPLAN Notices, 25, 303¿311. https://doi.org/10.1145/97946.97982 * Cutland, N. (1980). Computability: An Introduction to Recursive Function Theory (1 edition). Cambridge Eng.¿; New York: Cambridge University Press. * Filman, R., & Friedman, D. (2000). Aspect-Oriented Programming is Quantification and Obliviousness. Retrieved from citeseer.ist.psu.edu/filman00aspectoriented.html * Friedl, J. E. F. (2006). Mastering Regular Expressions (Third edition). Sebastapol, CA: O¿Reilly Media. * Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2006). Introduction to Automata Theory, Languages, and Computation (3 edition). Boston: Pearson. * Pierce, B. C. (2002). Types and programming languages. Cambridge, MA, USA: MIT Press. * Reisig, W. (2013). Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies. Retrieved from https://www.springer.com/gp/book/9783642332777   Complementarios   * Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., & Griswold, W. (2001). An Overview of AspectJ. Lecture Notes in Computer Science, 2072, 327¿355. * Kiczales, G., & Irwin, J. (1997). Aspect-Oriented Programming. Proc. of ECOOP 1997. Retrieved from http://www.cs.ubc.ca/~gregor/papers/kiczales-ECOOP1997-AOP.pdf * Kiczales, Gregor, Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., & Griswold, W. G. (2001). Getting started with ASPECTJ. Communications of the ACM, 44(10), 59¿65. * ReactiveX. (n.d.). Retrieved August 26, 2019, from http://reactivex.io/ |