|  |  |
| --- | --- |
| **Nombre Corto de la Asignatura** | **Fundamentos Ingeniería de SW** |
| **Nombre Largo de la Asignatura** | **Fundamentos de ingeniería de software** |
| **Código de la asignatura** | **34806** |
| **Grado** | Pregrado |
| **Descripción** | El objetivo de este curso es presentar los elementos esenciales de la ingeniería de software, así como las metodologías y herramientas de colaboración asociadas a esta disciplina. El curso busca desarrollar las habilidades y competencias necesarias para ser parte de un equipo colaborativo y afrontar un proyecto de construcción de software, independiente de la plataforma particular de implementación. La metodología del curso se basa en clases magistrales, talleres prácticos, y el desarrollo de un proyecto de desarrollo de software en grupo durante todo el semestre. |
| **Número de Créditos** | 3 |
| **Condiciones Académicas de Inscripción (Pre-requisitos)** | Requisitos de inscripción: Análisis y diseño de Software |
| **Período Académico de Vigencia** | 2430 |

|  |
| --- |
| **Objetivos de Formación** |
| 1. Presentar las metodologías y herramientas para la ingeniería de software, incluyendo las fases de planeación, desarrollo, implantación y puesta en marcha de un proyecto de software.  2. Presentar los diferentes roles y estrategias para el trabajo colaborativo dentro de un grupo de desarrollo de software.  3. Identificar estrategias, herramientas y metodologías para la generación de alternativas de solución a problemas de construcción de software.  4. Ilustrar la utilización de las herramientas para garantizar la calidad de un producto de software  5. Brindar literatura y textos en inglés relacionados con la ingeniería de software. |

|  |
| --- |
| **Resultados de Aprendizaje Esperados (RAE)** |
| Al finalizar el curso el estudiante estará en capacidad de:  1. Conocer las diferentes¿metodologías¿de proyectos de desarrollo de software y¿las responsabilidades y roles de los miembros de equipos establecidos en los proyectos (CDIO 4.5 y 3.1) (Disciplinar 1).  2. Conocer los mecanismos para la captura y definición de los requisitos de un producto de software. (CDIO 4.3 y 2.4) (Disciplinar 2)  3. Utilizar los lenguajes y herramientas de modelado, para diseñar la arquitectura de un producto de software (CDIO 4.3, 4.4) (Disciplinar 3)  4. Conocer los procesos para la verificación de los requisitos funcionales de un producto de software (CDIO 4.5 y 2.4) (Disciplinar 4) |

|  |
| --- |
| **Contenidos temáticos** |
| 1. Metodologías de desarrollo de software y ciclos de vida  2. Ingeniería de requisitos  3. Diseño de software  4. DevOps  5. Calidad y pruebas de software (Producto) |

|  |
| --- |
| **Estrategias Pedagógicas** |
| Las siguientes son las estrategias pedagógicas, que van a ser utilizadas dentro del curso, para apoyar el aprendizaje de los temas, por parte de los estudiantes.  1. Para cada tema, se desarrollarán exposiciones y talleres prácticos, en los cuales el estudiante debe ir aplicando los conocimientos adquiridos.  2. Se efectuarán sesiones de clase, en las cuales los grupos van a trabajar directamente con el profesor, para aclarar las dudas que vayan surgiendo con relación a la aplicación de las fases, metodologías y herramientas expuestas.  3. Se desarrollará un proyecto de curso, el cual ira evolucionando en su construcción, a partir de las distintas fases vistas en clase.  4. Para las secciones teóricas, se suministrará una bibliografía clara y concreta de los temas a ser vistos, que el estudiante debe leer y preparar previamente. |

|  |
| --- |
| **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 de la asignatura son:  1. Parcial 1: 20%  2. Parcial 2: 20%  3. Quices y tareas: 10%  4. Laboratorios 20%  5. Proyecto: 30%  5.1 Primera entrega: 8%  5.2 Segunda entrega: 10%  5.3 Entrega final: 12% |

|  |
| --- |
| **Recursos Bibliográficos** |
| Bibliografia - Textos Guia  1. Ambler, M., Lines, S. Disciplined Agile delivery. A practitioner¿s guide to agile software delivery in the enterprise. IBM Press., Ed. Pearson, 2013.  2. Larman, C. (2003). Agile and Iterative Development: A Manager¿s Guide. Addison-Wesley Professional.  Bibliografía - Textos complementarios  1. Bruegge, B., Dutoit. A. Object-Oriented Software Engineering, Prentice Hall, 3rd Edition, 2010. IEEE Computer Society.  2. SWEBOK: Guide to the Software Engineering Body of Knowledge, 2014. http://www.swebok.org [Enero, 2016].  3. Pressman, R. Ingeniería de Software: Un enfoque práctico. Quinta Edición. McGraw-Hill, 2002.  4. Sommerville, I. Software Engineering, 9th Edition, Addison Wesley, 2011.  5. Kim, G., Behr, K., & Spafford, G. (2018). The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win. IT Revolution.  6. Kim, G., Humble, J., Debois, P., & Willis, J. (2016). The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution.  7. Thorpe, E. (2019). Devops: A Comprehensive Beginners Guide to Learn Devops Step by Step. Independently Published.  8. H ttermann, M. (2012). DevOps for Developers. Apress.  9. Basili V., Caldiera, G., Rombach, H. D. Goal Question Metric Approach. Encyclopedia of Software Engineering, pp. 528-532, John Wiley & Sons, Inc., 1994.  10. Brooks, F. No Silver Bullet ¿ Essence and Accident in Software Engineering. Proceedings of the IFIP Tenth World Computing Conference: 1069-1076. 1986.  11. Cockburn, A. Basic Use Case Template. http://www.cs.colorado.edu/~kena/classes/6448/s01/examples/edmonb3.pdf [Enero, 2016]  12. Jacobson, I., Ng, P., McMahon, P., Spence, I., Lidman, S. The Essence of Software Engineering: Applying the SEMAT Kernel. Addison-Wesley Professional; 1 edition (January 26, 2013)  13. IEEE 610.12:1990. Standard Glossary of Software Engineering Terminology  14. IEEE 730:2014. Standard for Software Quality Assurance Processes  15. IEEE 828:2012. Standard for Software Configuration Management in Systems and Software Plans  16. IEEE 829:2008. Standard for Software and System Test Documentation  17. IEEE 1008:1987. Standard for Software Unit Testing  18. IEEE 1012:2004. Software Verification and Validation  19. IEEE 1016:2009. Standard for Information Technology - Systems Design - Software Design Descriptions  20. IEEE 1028:2008. Standard for Software Reviews and Audits  21. IEEE 1042:1987. Guide to Software Configuration Management  22. IEEE 1045:1992. Standard for Software Productivity Metrics  23. IEEE 1061:1998. Standard for a Software Quality Metrics Methodology  24. IEEE 1074:2006. Standard for Developing a Software Project Life Cycle Process  25. ISO/IEC/IEEE 24765:2010. International Standard for Systems and Software Engineering - Vocabulary  26. ISO/IEC/IEEE 29148:2011. International Standard for Systems and Software Engineering - Life cycle processes - Requirements Engineering  27. ISO/IEC/IEEE 42010:2011. Systems and software engineering - Architecture description  28. Larman, C. Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design, 2nd Edition, Prentice Hall, 2001  29. Naur, P., Randell, B. (Eds.). Software engineering: report of a conference sponsored by the NATO Science Commite, Garmish, Germany, 7-11 oct. http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF [Enero, 2017]  30. The Standish Group. The CHAOS Report. 1995. https://www.standishgroup.com/sample\_research\_files/chaos\_report\_1994.pdf [Diciembre, 2016]  31. Wells, D. Extreme Programming: a Gentle Introduction. http://www.extremeprogramming.org/ [Enero, 2017] |