Universidad Nacional de Quilmes

Quilmes, 6 de agosto de 2012

VISTO el Expediente Nº 827-0963/12, y

CONSIDERANDO:

Que por el mismo se tramita la apertura del llamado a concurso propuesto por el Departamento de Ciencia y Tecnología, para el año 2012.

Que los concursos docentes son la base democrática del funcionamiento universitario y dan la oportunidad de constituir una masa crítica que posibilite el adecuado funcionamiento de las Carreras.

Que se deben formalizar los cargos docentes obtenido por convenios específicos firmados con la Secretaría de Políticas Universitarias a través de los programas FOMENI I y FOMENI II.

Que los perfiles académicos requeridos en dicha propuesta y que figuran en el Anexo I se encuadran en las áreas de vacancia.

Que la Comisión de Asuntos Académicos Posgrados y Extensión ha emitido dictamen.

Por ello,

EL CONSEJO DEPARTAMENTAL DE CIENCIA Y TECNOLOGIA
R E S U E L V E:

ARTICULO 1º: Proponer al Consejo Superior la aprobación del llamado a concursos docentes del Departamento de Ciencia y Tecnología, según se detalla en el Anexo I de la presente Resolución. Se adjuntan los contenidos mínimos en el anexo II de la presente resolución.

ARTICULO 2º: Regístrese, practíquense las comunicaciones de estilo

Res. CD CyT Nº 091/12
FIRMADA POR: Dr. Pablo Daniel Ghiringhelli Director Dpto. Ciencia y Tecnología

ANEXO I

AREA DISCIPLINAR/ CARRERA	CAMPO CURRICULAR/ ASIGNATURA	CATEGORIA	Nº DE CARGOS	DEDICACION	PERFIL	FUNCION	MODALIDAD	LINEA DE INVESTIGACION/ DESARROLLO PROFESIONAL	REQUISITOS ACADEMICOS Y PROFESIONALES
Programación (TPI)	Estructura de Datos	Prof. Adjunto	1	Parcial	Docencia e Investigación	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05
Programación (TPI)	Introducción a la Programación	Prof. Adjunto	1	Parcial	Docencia e Investigación	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05
Programación (TPI)	Programación con Objetos I	Prof. Adjunto	1	Parcial	Docencia e Investigación	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05
Programación (TPI)	Programación Funcional	Prof. Instructor	1	Parcial	Docencia e Investigación	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05
Programación (TPI)	Introducción a la Programación	Prof. Instructor	1	Parcial	Docencia e Investigación	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05
Programación (TPI)	Construcción de Interfases de Usuario	Prof. Instructor	1	Parcial	Docencia y Desarrollo Profesional	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05

Programación (TPI)	Introducción a la Programación	Prof. Instructor	1	Parcial	Docencia y Desarrollo Profesional	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05
Programación (TPI)	Programación con Objetos II	Prof. Instructor	1	Parcial	Docencia y Desarrollo Profesional	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05
Sistemas Informáticos (TPI)	Organización de computadoras	Prof. Instructor	1	Parcial	Docencia y Desarrollo Profesional	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05
Sistemas Informáticos (TPI)	Redes de Computadoras	Prof. Instructor	1	Parcial	Docencia	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05
Sistemas Informáticos (TPI)	Laboratorio de Redes y Sistemas Operativos	Prof. Instructor	1	Parcial	Docencia y Desarrollo Profesional	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05
Sistemas Informáticos (TPI)	Seguridad Informática	Prof. Instructor	1	Parcial	Docencia y Desarrollo Profesional	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05
Sistemas Informáticos (TPI)	Elementos de Ingeniería de Software	Prof. Instructor	1	Parcial	Docencia y Desarrollo Profesional	Docente	Presencial	-	Lo que corresponde a la categoría y perfil según Resolución (CS) Nº 324/05

ANEXO II

AREA DISCIPLINAR/ CARRERA	CAMPO CURRICULAR/ ASIGNATURA	CONTENIDOS MÍNIMOS
Programación (TPI)	Estructura de Datos	Recursión sobre listas y árboles. Programas recursivos.Tipos algebraicos: maybe, either, enumerativos, listas, árboles binarios, árboles generales. Estructuras contenedoras: pilas, colas, diccionarios, heaps, árboles balanceados, contenedores basados en representaciones numéricas. Nociones de representación e invariante de representación y su utilidad en el diseño e implementación de estructuras de datos. Uso imperativo de estructuras de datos. Iteración en listas y árboles. Modelo de memoria imperativo: stack/heap, alocación de memoria. Punteros. Variables por referencia. Listas encadenadas y sus variantes.Árboles implementados con punteros. Binary heaps implementadas con arrays. Hashing. Análisis de eficiencia e implementación. Algoritmos de ordenamiento. Clasificación e implementación. Nociones básicas de algoritmos sobre grafos.
Programación (TPI)	Introducción a la Programación	Qué es un programa.Las herramientas del programador: entornos de ejecución y de desarrollo. Principios de la programación imperativa: acciones y comandos, valores y expresiones, tipos, estado. Terminación y parcialidad. Precondiciones como metodología para desarrollo de software robusto. Principios de la programación estructurada: funciones y procedimientos. Necesidad de darle una estructura a un programa no trivial. Resolución de pequeños problemas mediante programas. Estructuras de datos básicas: listas y registros.
Programación (TPI)	Programación con Objetos I	Conceptos fundantes del paradigma: objeto y mensaje. Visión externa del objeto: dispositivo computacional capaz de recibir mensajes y otorgar respuestas adecuadas. Relevancia de estos conceptos (con qué objetos cuento, qué mensajes le puedo enviar a cada uno) en el desarrollo de software. Concepto de polimorfismo en objetos, comprensión de las ventajas de aprovecharlo. Protocolo/interfaz, concepto de tipo en objetos, comprensión de que un objeto puede asumir distintos tipos. La interfaz como contrato al que se comprometen ciertos objetos, posibilidad de reforzar ese contrato. Estado en el paradigma de objetos: referencias, conocimiento, estado interno. Métodos, clases, herencia, method lookup. Conceptos de responsabilidad y delegación, su rol al definir una trama de objetos que responde a requerimientos determinados. Colecciones: conceptualización como objetos, caracterización a partir de los conceptos de protocolo y responsabilidad, protocolo, acceso a sus elementos. Testeo automático y repetible, test como comprobación tanto del correcto funcionamiento como de que los objetos definidos son efectivamente usables.
Programación (TPI)	Programación Funcional	Nociones generales del paradigma funcional. Valores y expresiones. Las funciones como valores. Mecanismos de definición de expresiones y valores. Ecuaciones orientadas para definir funciones. Sintaxis. Sistema de Tipos Hindley-Milner. Tipos básicos. Constructores de tipos. Polimorfismo. Sintaxis para valores de cada tipo (caracteres, tuplas, listas, strings, funciones). Funciones de alto orden. Currificación. Inducción/Recursión. Definición inductiva de conjuntos. Definición recursiva de funciones sobre esos conjuntos. Demostraciones inductivas sobre dichas funciones. Inducción estructural. Ejemplos: programas, expresiones aritméticas, listas. Listas. Listas como tipo inductivo. Funciones básicas sobre listas (append, head, tail, take, drop, reverse, sort, elem, etc.). Funciones de alto orden sobre listas. Patrón de recorrido: map. Patrón de selección: filter. Patrón de recursión: foldr. Demostración de propiedades sobre listas y funciones sobre listas. Sistemas de Tipos. Nociones básicas. Sistemas de tipado fuerte. Ventajas y limitaciones de los lenguajes de programación con tipos. Lenguaje de tipos. Asignación de tipos a expresiones. Propiedades interesantes de esta asignación. Algoritmo de inferencia. Mecanismos de definición de tipos nuevos y

		de funciones sobre ellos. Tipos algebraicos recursivos. Ejemplos: enumeraciones, listas, árboles binarios, árboles generales. Transformación de Programas. Motivación. Obtención de programas a partir de especificaciones. Mejoramiento de eficiencia, con corrección por construcción. Técnicas particulares de transformación: tupling, eliminación de recursión, fusión.
Programación (TPI)	Introducción a la Programación	Qué es un programa.Las herramientas del programador: entornos de ejecución y de desarrollo. Principios de la programación imperativa: acciones y comandos, valores y expresiones, tipos, estado. Terminación y parcialidad. Precondiciones como metodología para desarrollo de software robusto. Principios de la programación estructurada: funciones y procedimientos. Necesidad de darle una estructura a un programa no trivial. Resolución de pequeños problemas mediante programas. Estructuras de datos básicas: listas y registros.
Programación (TPI)	Construcción de Interfases de Usuario	Variantes en arquitecturas de sistema respecto de la interfaz de usuario: aplicación centralizada, cliente-servidor o distribuida; ejecución en un cliente de aplicación (browser, flash, otros) o mediante un programa específico; concepto de RIA. Arquitecturas web, protocolos y tecnologías asociados. Modelos de interacción de la interfaz de usuario con su entorno: interfaces orientadas a eventos, pedido-respuesta, basadas en continuations. Aplicaciones client-initiative y application-initiative. Componentes gráficos usuales en interfaces de usuario. Diferentes estrategias para describir una vista, sus componentes y la distribución espacial de los mismos: HTML estático, CSS, generación programática de HTML, server pages, templates, descripción basada en componentes, descripciones declarativas. Problemas característicos de cada estrategia; herramientas que las soportan. Vinculación entre la interfaz de usuario y el modelo de dominio subyacente. Problemática asociada a transformaciones, validaciones, manejo de errores, excepciones, transacciones e identidad. Distintos enfoques: generación automática de la interfaz de usuario a partir del modelo, vínculos explícitos entre elementos de interfaz de usuario y de modelo, DAOs, servicios. Adaptaciones de un modelo de dominio a las necesidades de dinamismo, navegación y distintos niveles de discriminación/agregación de la interfaz de usuario. Objetos de nivel de aplicación, casos de uso, concepto de modelo de la vista. Patrones de interacción, mvc. Análisis de tecnologías de presentación de acuerdo a los conceptos presentados en esta materia; evaluación de características, selección de opciones tecnológicas teniendo en cuenta el proyecto de desarrollo a realizar. Nociones sobre desarrollo propio de complementos a tecnologías desarrolladas por otros. Impacto de la distribución de aplicaciones en la interfaz de usuario, comunicación sincrónica y asincrónica. Navegación y manejo del estado conversacional. REST, estado en sesión. Nociones de usabilidad: concepto, pertinencia, conveniencia de definir y mantener standards.
Programación (TPI)	Introducción a la Programación	Qué es un programa.Las herramientas del programador: entornos de ejecución y de desarrollo. Principios de la programación imperativa: acciones y comandos, valores y expresiones, tipos, estado. Terminación y parcialidad. Precondiciones como metodología para desarrollo de software robusto. Principios de la programación estructurada: funciones y procedimientos. Necesidad de darle una estructura a un programa no trivial. Resolución de pequeños problemas mediante programas. Estructuras de datos básicas: listas y registros.
Programación (TPI)	Programación con Objetos II	Aproximación al diseño de software: en qué dimensiones puede crecer un proyecto de software, problemáticas que devienen de este crecimiento, necesidad de pensar en la organización de un sistema como elementos relacionados, pensando en la funcionalidad de cada elemento y de qué relaciones se establecen. Noción de decisión de diseño, el diseño es un proceso de toma de decisiones. Conceptos de acoplamiento y cohesión. Problemas que derivan de un grado de acoplamiento inadecuado. Vinculación entre las ideas básicas de diseño y el paradigma de objetos: objetos como elementos, conocimiento como relación, responsabilidades como funcionalidad de cada elemento, tipo y polimorfismo para comprender que ciertos elementos son intercambiables a efectos de un diseño. Características deseadas en un diseño de objetos: no repetición de implementación de lógica, capacidad de separar entre grupos de objetos cohesivos con responsabilidades aplicables al grupo. Patrones de diseño: idea de patrón, consecuencias del uso de algunos patrones respecto de las características del diseño y de las cualidades

		pretendidas del producto. Nociones sobre proceso de diseño: foco en las responsabilidades, pensar los problemas desde las características básicas del paradigma, pertinencia de iterar entre diseño en papel, codificación y test, relevancia de los diagramas de objetos y de clases. Metaprogramación, características reflexivas de un lenguaje de programación. Uso de un entorno integrado de software del estilo de los utilizados ampliamente en la industria, funcionalidades que provee, aprovechamiento de sus facilidades. Notación UML de los diagramas de clases, de objetos y de secuencia.
Sistemas Informáticos (TPI)	Organización de computadoras	Representación de la información: alfanumérico, numérico, punto fijo y flotante, ASCII. Sistema de numeración binario. Aritmética de las computadoras: Unidades. Funcionamiento y organización (modelo de Von Neumann). Unidades funcionales: Unidad Central de Proceso, Unidad de Control, memorias, ciclo de instrucciones, direccionamiento, subsistema de Memoria. Periféricos: conceptos y principio de funcionamiento. Procesadores de Entrada/Salida. Lógica digital: tablas de verdad, equivalencia de fórmulas proposicionales, circuitos combinatorios, circuitos secuenciales Arquitectura del computador: Componentes de la CPU, memoria principal y secundaria, jerarquía de memorias. Subsistema de Entrada/Salida. Lenguaje Máquina. Código fuente y código objeto.
Sistemas Informáticos (TPI)	Redes de Computadoras	Concepto de red de computadoras, redes y comunicación. Modelos en capas, modelo OSI, modelo de la Internet. Conceptos de protocolo y de servicio. Nivel físico: dispositivos, cableado estructurado. Nivel de enlace: concepto de enlace, tramas, puentes, enlaces inalámbricos. Nivel de red: concepto de ruteo, topolog´ıas, algoritmos de ruteo, protocolos IP, resolución de direcciones. Nivel de transporte: funciones, protocolos UDP y TCP, multiplexación, concepto de socket, control de congestión. Modelo general de Internet: integración de niveles y protocolos, servicios de red (http, dhcp, dns, smtp, etc.), su utilización en el funcionamiento de la Web. Estándares utilizados en Internet, concepto de RFC. Concepto e implementación de las VPN. Administración de redes: servicios, firewalls. Sistemas cliente/servidor.
Sistemas Informáticos (TPI)	Laboratorio de Redes y Sistemas Operativos	Instalación, configuración y operación de distintos servicios relacionados con Internet: servidores de aplicaciones, servidor y cliente de mail, servidor y cliente FTP, firewalls, etc. Servicios de directorio, servidores LDAP, uso desde aplicaciones. Gestión de usuarios y control de accesos en un entorno operativo, impacto en la instalación de aplicaciones, posibilidad de compartir recursos. Sistemas de backup automatizados, políticas de criticidad. Instalación, configuración y operación de repositorios de código. Monitoreo de redes, protocolo SNMP.
Sistemas Informáticos (TPI)	Seguridad Informática	Conceptos de seguridad informática: ataques, amenazas, servicios, herramientas, estándares. Firma Digital. Nociones básicas de criptografía, criterios de calidad, algoritmos. Seguridad en redes: protocolos asociados a la seguridad, autenticación de equipos en redes de computadora.
Sistemas Informáticos (TPI)	Elementos de Ingeniería de Software	Metodologías ágiles: actividades, productos, formas de articulación, roles. Ejemplos: Scrum. Metodologías estructuradas: actividades, productos, formas de articulación, roles. Ejemplos: UP. Debate sobre similitudes y diferencias entre metodologías ágiles y estructuradas. Concepto de ciclo de vida, relación con distintas metodologías. Métricas: qué son, qué miden, para qué sirven, cuándo sirven. Ejemplos de métricas asociadas a desarrollo de software en general y actividades de programación en particular. Estimación de esfuerzos: relevancia de la experiencia previa para estimar, heurísitcas utilizadas. Pertinencia de estimaciones relativas. Técnicas de estimación asociadas a metodologías ágiles. Conceptos de requerimiento funcional y no funcional, pertinencia de definiciones comprensibles y adecuadas. Comprensión de requerimientos funcionales, detección de inconsistencias. Implementación en código de requerimientos funcionales, verificación de que el código construido cumple los requerimientos. Problemas asociados a requerimientos no funcionales: pertinencia de definiciones medibles, nociones sobre técnicas de verificación, posibilidad de garantizarlos por construcción. Nociones sobre distintos tipos de testing: de unidad, funcional, de sistema, de stress, de carga. Cualidades deseadas y técnicas para lograrlas: regresión, automatización, independencia. Ejemplos concretos de test de unidad y de test funcional. Noción de coverage. Prácticas asociadas a extreme

		programming: peer programming, relevancia de tests automáticos, integración continua, interacción de las actividades de coding y refactor. Noción de TDD. Nociones de riesgo y plan de contingencia.