

Plan de Estudios

Licenciatura en Desarrollo de Software

Índice

1. Identificación de la carrera	3
1.1. Fundamentación	3
1.2. Denominación	3
1.3. Nivel	3
1.4. Ubicación en la estructura institucional	3
2. Horizontes de la carrera	4
2.1. Objetivos	4
2.2. Perfil del egresado	4
2.3. Alcances	4
3. Diseño curricular de la carrera	5
3.1. Requisitos de ingreso	5
3.2. Estructura curricular	5
3.2.1. Diplomatura en Programación Informática	5
3.2.2. Licenciatura en Desarrollo de Software	7
3.3. Diplomatura en Programación Informática – Asignaturas del Núcleo Básico – Contenidos mínimos	8
3.3.1. Matemática I	8
3.3.2. Introducción a la Programación	9
3.3.3. Organización de Computadoras	9
3.3.4. Estructuras de Datos	9
3.3.5. Programación con Objetos I	10
3.3.6. Bases de Datos	10
3.3.7. Matemática II	10
3.3.8. Programación con Objetos II	11
3.3.9. Redes de Computadoras	11
3.3.10. Sistemas Operativos	12
3.3.11. Programación Funcional	12
3.3.12. Construcción de Interfaces de Usuario	13
3.3.13. Estrategias de Persistencia	14
3.3.14. Laboratorio de Sistemas Operativos y Redes	14
3.4. Licenciatura en Desarrollo de Software – Asignaturas del Núcleo Básico – Contenidos mínimos	15
3.4.1. Análisis Matemático	15
3.4.2. Lógica y Programación	15
3.4.3. Elementos de Ingeniería de Software	15
3.4.4. Seguridad de la Información	16
3.4.5. Matemática III	16
3.4.6. Programación Concurrente	17

3.4.7.	Gestión de Proyectos de Desarrollo de Software	17
3.4.8.	Práctica del Desarrollo de Software	17
3.4.9.	Probabilidad y Estadística	18
3.4.10.	Lenguajes Formales y Autómatas	19
3.4.11.	Algoritmos	19
3.4.12.	Ingeniería de Requerimientos	19
3.5.	Licenciatura en Desarrollo de Software – Asignaturas del Núcleo Avanzado	
	– Contenidos mínimos	20
3.5.1.	Teoría de la Computación	20
3.5.2.	Programación con objetos III	20
3.5.3.	Arquitectura de Software I	20
3.5.4.	Sistemas Distribuidos	22
3.5.5.	Características de Lenguajes de Programación	22
3.5.6.	Arquitectura de Software II	22
3.5.7.	Arquitectura de Computadoras	23
3.5.8.	Parseo y generación de código	23
3.5.9.	Aspectos Legales y Sociales	24
3.6.	Licenciatura en Desarrollo de Software – Asignaturas del Núcleo de Ori-	
	entación – Contenidos mínimos	25
3.6.1.	Bases de Datos II	25
3.6.2.	Participación y Gestión en Proyectos de Software Libre	25
3.6.3.	Introducción a la Bioinformática	26
3.6.4.	Políticas Públicas en la Sociedad de la Información y la Era Digital	26
3.6.5.	Sistemas de Información Geográfica	26
3.6.6.	Herramientas Declarativas en Programación	27
3.6.7.	Introducción al Desarrollo de Videojuegos	27
3.6.8.	Derechos de Autor y Derecho de Copia en la Era Digital	27
3.6.9.	Análisis Estático de Programas y Herramientas Asociadas	28
3.6.10.	Semántica de Lenguajes de Programación	28
3.6.11.	Seminarios	28
3.6.12.	Seminarios sobre Herramientas o Técnicas Puntuales	28
3.7.	Talleres de formación humanística – Contenidos mínimos	29
3.7.1.	Taller de Trabajo Intelectual	29
3.7.2.	Taller de Trabajo Universitario	29
3.8.	Contenidos mínimos de los niveles de idioma Inglés	29
3.8.1.	Inglés I	29
3.8.2.	Inglés II	29

1. Identificación de la carrera

1.1. Fundamentación

La Argentina muestra una actividad económica robusta en el área de desarrollo de software, originada en y retroalimentándose con una cultura informática temprana y ampliamente extendida, al menos en los principales centros urbanos. El aumento sostenido que se espera en la demanda global de servicios asociados a las tecnologías de la información y las comunicaciones (TICs) augura para el área un amplio potencial de crecimiento.

El país cuenta con varios de los factores necesarios para aprovechar este potencial en particular respecto del desarrollo de software, entre ellos una amplia base de empresas del sector de distintas características y tamaños que trabajan tanto en el mercado local como en el internacional, una cantidad interesante de profesionales con capacidades competitivas a nivel global, y un fuerte y consistente apoyo estatal al sector.

Por otro lado, Argentina no es el único país que ha detectado la posibilidad de generación de empleos de calidad y de desarrollo económico que brindan las TICs en general y el desarrollo de software en particular. Hay varios países que vienen desarrollando estrategias que les han permitido una inserción importante en los mercados mundiales dentro de estas actividades.

Creemos que el desarrollo del sector en la Argentina puede beneficiarse de un posicionamiento global que destaque la capacidad de proveer servicios de alta calidad. Esta visión motiva la orientación del plan que proponemos, que aspira a conjugar práctica extensiva en habilidades directamente relacionadas con las necesidades que percibimos en el mercado laboral con una sólida formación en los conceptos de base de la programación y con el énfasis en el cuidado de distintos criterios de calidad de los productos de software construidos.

Otro factor que estimamos importante destacar es el entorno sociohumano de la UNQ. La Universidad está inserta en un área urbana densamente poblada con una gran dispersión en la calidad de la educación recibida por los jóvenes que ingresan en la Universidad.

La propuesta que presentamos tiene en cuenta la realidad de la población de estudiantes con que contamos. El plan está pensado para una transmisión gradual y progresiva de los conceptos principales que deben ser incorporados. De esta forma se hace énfasis en el acompañamiento a cada estudiante en su incorporación a una currícula de nivel universitario.

La propuesta que presentamos aspira a fortalecer la capacidad de los egresados/as de ocupar empleos de calificación alta en un sector que cuenta con una amplia oferta laboral y buenas perspectivas de crecimiento.

1.2. Denominación

Carrera: Licenciatura en Desarrollo de Software

Título: Licenciado en Desarrollo de Software

1.3. Nivel

Grado.

1.4. Ubicación en la estructura institucional

La formación de grado del Licenciado en Desarrollo de Software incluye un ciclo inicial, que se acredita como Diplomatura en Programación Informática, y un ciclo superior, que conduce a la Licenciatura en Desarrollo de Software. La carrera se desarrollará en modalidad presencial y tendrá una duración total estimada de 5 años.

2. Horizontes de la carrera

2.1. Objetivos

Formar profesionales con pensamiento crítico y conciencia social, capaces de concebir soluciones a un amplio espectro de problemas asociados a las tareas de análisis, diseño, programación e implantación de software. Los mismos deben ser capaces de aprovechar los conceptos aprendidos en la carrera para pensar y resolver situaciones concretas ya sea individualmente o en equipos, y basados en una amplia experiencia práctica obtenida durante el recorrido de la carrera.

En particular se persiguen los siguientes objetivos para el egresado:

- Que tenga pensamiento crítico.
- Que tenga conciencia social.
- Que sea creativo e innovador.
- Que tenga capacidad de liderazgo y dirección de proyectos.
- Que adquiera conocimientos robustos sobre los procesos de análisis, diseño, programación e implantación de software.
- Que adquiera experiencia práctica en la aplicación de estos conocimientos.

2.2. Perfil del egresado

El egresado es un profesional universitario cuya área de acción principal es la problemática de la construcción de software, incluyendo todas las fases involucradas en el proceso.

El egresado será capaz de diseñar soluciones a problemas informáticos como así también implementar dichas soluciones describiendo los conceptos que fundamentan las decisiones que tomó y velando por los parámetros de calidad del producto. Podrá conformar y liderar equipos de trabajo que aborden estas problemáticas y que privilegien la colaboración por sobre la competencia, teniendo en cuenta elementos que faciliten el trabajo en grupo, tanto en lo actitudinal (intercambio de conocimientos, organización tareas) como en lo técnico (conocimientos de herramientas y entornos).

2.3. Alcances

El egresado deberá poder realizar las siguientes acciones:

- Diseñar soluciones a problemas algorítmicos de variada envergadura contemplando múltiples requerimientos.
- Conformar y liderar equipos de análisis, diseño, desarrollo e implantación, fijando objetivos y realizando un adecuado seguimiento.
- Realizar o dirigir proyectos de software teniendo en cuenta parámetros básicos de calidad incluyendo grado de test, claridad, mantenibilidad, robustez frente a fallos, uso eficiente de recursos y extensibilidad.
- Evaluar proyectos para determinar su factibilidad técnica y estimar los recursos necesarios para su compleción.
- Incorporar a su práctica nuevas herramientas que surjan en el ámbito profesional.
- Manejar con fluidez el entorno que necesita un profesional para trabajar: sistema operativo, entornos de desarrollo, entornos de ejecución.

- Comprender las implicancias y responsabilidades sociales asociadas a la concepción, construcción y uso del software.

3. Diseño curricular de la carrera

La formación de grado de la Licenciatura en Desarrollo de Software incluye un ciclo inicial, que se acredita como Diplomatura en Programación Informática, y un ciclo superior que conduce al título de Licenciado en Desarrollo de Software. El plan se elabora sobre la base de períodos medidos en cuatrimestres y se organiza en seis áreas, siguiendo los lineamientos de la Resolución 786/09 del Ministerio de Educación. La siguiente tabla muestra las seis áreas en conjunto con las cargas horarias mínimas de este plan para cada una de ellas:

Área	Horas totales mínimas
Ciencias Básicas	504
Teoría de la Computación	396
Algoritmos y Lenguajes	900
Arquitectura, Sistemas Operativos y Redes	576
Ingeniería de Software, Base de Datos y Sistemas de Información	936
Aspectos Profesionales y Sociales	72

Por otra parte, el conjunto de las asignaturas a dictar están presentadas en tres núcleos: Núcleo Básico, Núcleo Avanzado y Núcleo de Orientación.

		Horas	Créditos	Cuatr.
Diplomatura en Programación Informática	Núcleo Básico	1512	168	4
	Taller de Formación Humanística	36	4	
	Otros requerimientos (Inglés)	108	-	
Ciclo superior de la Licenciatura en Desarrollo de Software	Núcleo Básico	1152	128	3
	Núcleo Avanzado	720	80	3
	Núcleo Orientación	216	24	
	Seminario Final	-	20	
Total Diplomatura en Programación Informática		1656	172	4
Total Licenciatura en Desarrollo de Software		3744	424	10

3.1. Requisitos de ingreso

Los establecidos por la Ley 24521 de Educación Superior, o las leyes que eventualmente la reemplacen.

3.2. Estructura curricular

3.2.1. Diplomatura en Programación Informática

El ciclo inicial, llamado Diplomatura en Programación Informática, está orientado a ofrecer formación sólida en los conocimientos de base relacionados con la Programación Informática. Al finalizar los estudios correspondientes al primer ciclo, los alumnos obtendrán un certificado de Diplomado en Programación Informática.

Para acceder al certificado de Diplomatura en Programación Informática, el estudiante deberá:

1. acreditar conocimientos de Inglés análogos a dos niveles cuatrimestrales de 54 horas cada uno;
2. aprobar al menos un Taller de Formación Humanística; y
3. aprobar las asignaturas del Núcleo Básico reuniendo la cantidad de horas indicadas arriba.

A continuación se listan las asignaturas de la Diplomatura en Programación Informática y la organización cuatrimestral, comenzando con los talleres de formación humanística e Inglés:

Núcleo	Asignatura	Horas se-ma-nales	Carga ho-raria total	Crédi-tos	Cont. Míni-mos (pág.)
	Taller de Trabajo Intelectual	2	36	4	29
	Taller de Trabajo Universitario	2	36	4	29
	Inglés I	3	54	-	29
	Inglés II	3	54	-	29

A continuación se listan el resto de las asignaturas de la Diplomatura en Programación Informática, incluyendo la división sugerida en cuatrimestres:

Núcleo	Asignatura	Horas se-ma-nales	Carga ho-raria total	Crédi-tos	Cont. Míni-mos (pág.)
Primer Cuatrimestre					
Básico	Matemática I	8	144	16	8
Básico	Introducción a la Programación	8	144	16	9
Básico	Organización de Computadoras	6	108	12	9
Segundo Cuatrimestre					
Básico	Estructuras de Datos	8	144	16	9
Básico	Programación con Objetos I	8	144	16	10
Básico	Bases de Datos	6	108	12	10
Tercer Cuatrimestre					
Básico	Matemática II	4	72	8	10
Básico	Programación con Objetos II	6	108	12	11
Básico	Redes de Computadoras	4	72	8	11
Básico	Sistemas Operativos	6	108	12	12
Cuarto Cuatrimestre					
Básico	Programación Funcional	4	72	8	12
Básico	Construcción de Interfaces de Usuario	6	108	12	13
Básico	Estrategias de Persistencia	6	108	12	14
Básico	Laboratorio de Sistemas Operativos y Redes	4	72	8	14
Totales			1512	168	

3.2.2. Licenciatura en Desarrollo de Software

El ciclo superior de la Licenciatura en Desarrollo de Software completa la Diplomatura en Programación Informática con extensa formación en temas específicos de la disciplina con el objetivo de que el egresado pueda ejercer un rol protagónico en la misma. Para acceder al título de Licenciado en Desarrollo de Software, el estudiante deberá:

1. acreditar la posesión del título de Diplomatura en Programación Informática;
2. haber aprobado las asignaturas del Núcleo Básico, Avanzado y de Orientación, reuniendo la cantidad de horas indicadas arriba; y
3. realizar el *Seminario Final* (que se computa como una asignatura de 20 créditos).

A continuación se listan las asignaturas del Núcleo Básico, Avanzado y de Orientación correspondientes al ciclo superior de la Licenciatura en Desarrollo de Software.

Núcleo	Asignatura	Horas se-ma-nales	Carga ho-raria total	Crédi-tos	Cont. Míni-mos (pág.)
Quinto Cuatrimestre					
Básico	Análisis Matemático	6	108	12	15
Básico	Lógica y Programación	6	108	12	15
Básico	Elementos de Ingeniería de Software	6	108	12	15
Básico	Seguridad de la Información	4	72	8	16
Sexto Cuatrimestre					
Básico	Matemática III	4	72	8	16
Básico	Programación Concurrente	6	108	12	17
Básico	Gestión de Proyectos de Desarrollo de Software	4	72	8	17
Básico	Práctica del Desarrollo de Software	8	144	16	17
Séptimo Cuatrimestre					
Básico	Probabilidad y Estadística	6	108	12	18
Básico	Lenguajes Formales y Autómatas	4	72	8	19
Básico	Algoritmos	6	108	12	19
Básico	Ingeniería de Requerimientos	4	72	8	19
Octavo Cuatrimestre					
Orient.	Complementaria	4	72	8	NA
Avanz.	Teoría de la Computación	4	72	8	20
Avanz.	Programación con objetos III	4	72	8	20
Avanz.	Arquitectura de Software I	6	108	12	20
Avanz.	Sistemas Distribuidos	4	72	8	22
Noveno Cuatrimestre					
Orient.	Complementaria	4	72	8	NA
Avanz.	Características de Lenguajes de Programación	4	72	8	22
Avanz.	Arquitectura de Software II	6	108	12	22
Avanz.	Arquitectura de Computadoras	4	72	8	23
Décimo Cuatrimestre					
Orient.	Complementaria	4	72	8	NA
Avanz.	Parseo y generación de código	4	72	8	23
Avanz.	Aspectos Legales y Sociales	4	72	8	24
Totales			2088	232	

Para cursar materias del Núcleo Básico del ciclo superior de la Licenciatura en Desarrollo de Software, el estudiante deberá acreditar como mínimo 75 % de los créditos del Núcleo Básico de la Diplomatura en Programación Informática. Para cursar materias del Núcleo Avanzado del ciclo superior de la Licenciatura en Desarrollo de Software, el estudiante deberá acreditar el 100 % de los créditos del Núcleo Básico de la Diplomatura en Programación Informática, y como mínimo 50 % de los créditos del Núcleo Básico del ciclo superior de la Licenciatura en Desarrollo de Software. Para cursar materias del Núcleo de Orientación del ciclo superior de la Licenciatura en Desarrollo de Software, el estudiante deberá acreditar el 100 % de los créditos del Núcleo Básico del ciclo superior de la Licenciatura en Desarrollo de Software.

Los cursos del Núcleo de Orientación son:

Núcleo	Asignatura	Horas semanales	Carga horaria total	Créditos	Cont. Mínimos (pág.)
Orient.	Bases de Datos II	4	72	8	25
Orient.	Participación y Gestión en Proyectos de Software Libre	4	72	8	25
Orient.	Introducción a la Bioinformática	4	72	8	26
Orient.	Políticas Públicas en la Sociedad de la Información y la Era Digital	4	72	8	26
Orient.	Sistemas de Información Geográfica	4	72	8	26
Orient.	Herramientas Declarativas en Programación	4	72	8	27
Orient.	Introducción al Desarrollo de Videojuegos	4	72	8	27
Orient.	Derechos de Autor y Derecho de Copia en la Era Digital	4	72	8	27
Orient.	Análisis Estático de Programas y Herramientas Asociadas	4	72	8	28
Orient.	Semántica de Lenguajes de Programación	4	72	8	28
Orient.	Seminarios	4	72	8	28
Orient.	Seminarios sobre Herramientas o Técnicas Puntuales		32	4	28

3.3. Diplomatura en Programación Informática – Asignaturas del Núcleo Básico – Contenidos mínimos

3.3.1. Matemática I

- Lógica proposicional y de primer orden. Técnicas de prueba.
- Teoría básica de conjuntos.
- Inducción matemática sobre números naturales.
- Relaciones binarias: relaciones de orden, relaciones de equivalencia, relaciones funcionales.
- Elementos básicos de análisis combinatorio.

3.3.2. 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.

3.3.3. 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.

3.3.4. Estructuras 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: sk/heap, asignación 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.

3.3.5. 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.
- Nociones básicas sobre manejo de errores: distinción entre error y valor de retorno, acciones posibles al detectar una situación de error. Interrupción del flujo de ejecución: modelado mediante estructuras de control, concepto de excepción.

3.3.6. Bases de Datos

- Qué es un modelo de datos, modelos conceptuales, lógicos y físicos.
- Modelo de entidad-relación: conceptos básicos.
- Modelo relacional: tabla, atributo, dominio, valor, fila; restricciones de integridad; operaciones que se pueden hacer.
- SQL: concepto de lenguaje de consulta, sintaxis, concepto de join, agrupamientos, subqueries, joins parciales.
- Transacción: concepto, demarcación de transacciones.

3.3.7. Matemática II

- Matrices, determinantes y sistemas de ecuaciones lineales.
- Estructuras algebraicas: monoides, semigrupos y grupos.
- Espacios vectoriales de dimensión finita.
- Aritmética entera y modular.
- Introducción a la probabilidad discreta.

3.3.8. 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.
- Profundización del trabajo sobre testeo unitario y automático.
- Profundización del trabajo sobre manejo de errores, impacto del manejo de errores en el diseño.

3.3.9. 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, topologías, 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.

3.3.10. Sistemas Operativos

- Introducción a los sistemas operativos: función de abstracción del hardware; organización, estructura y servicios de los SO. Tipos de sistemas (Sistemas batch / Multi-programación / Sistemas de tiempo real / Sistemas distribuidos / Sistemas paralelos / Sistemas embebidos).
- Procesos y threads: Conceptos de proceso, thread y planificación. Comunicación y cooperación entre procesos. Deadlocks.
- Planificación: Algoritmos, criterios. Multiprocesamiento.
- Manejo de memoria: Espacio lógico vs físico, swapping, alocaión contigua, paginación, segmentación.
- Memoria virtual: Paginación bajo demanda, algoritmos de reemplazo de página, thrashing.
- Sistemas de archivos: Manejo de archivos, manejo de directorios.
- Protección: objetivos, dominio de protección, matriz de acceso y sus implementaciones.
- Prácticas, trabajos incluyendo uso de shell scripting e instalaciones en distintos sistemas operativos, en particular del estilo Unix: GNU/Linux, etc..

3.3.11. 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.

3.3.12. Construcción de Interfaces 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 asignatura; 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.

3.3.13. Estrategias de Persistencia

- Nociones sobre los problemas que derivan del acceso concurrente a una base de datos. Algunas estrategias para mitigarlos, en particular lockeo y manejo adecuado de transacciones.
- Nociones sobre la problemática de performance en el acceso a una base de datos, relación con la escala, otros factores que influyen. Estrategias de acceso a los datos ante una consulta, concepto de índice.
- Conceptos de usuario y permiso en una base de datos, esquemas típicos de definición de usuarios y permisos.
- Bases de objetos: concepto, panorama, experimentación práctica, comparación con bases de datos relacionales.
- Bases de datos distribuidas para grandes volúmenes de datos, acceso a datos como un servicio, herramientas de programación asociadas. Transacciones distribuidas.
- Interacción entre un programa y un mecanismo de persistencia: nociones básicas, problemáticas generales.
- Mecanismos de acceso y recuperación de objetos persistidos en bases de datos relacionales: mecanismos de recuperación de objetos (uso de lenguajes de consulta relacionales, lenguajes de consulta orientados a objetos, interfaz en objetos orientada al acceso, interfaces en términos del modelo de dominio). Actualización del estado persistente: reachability, cascada.
- ORM, conceptos básicos, alcances, cuestiones que resuelven, enfoque que toma respecto de la transformación de objetos. Problemas de mapeo: herencia, relaciones n-m, estrategias no standard.
- Transacciones a nivel aplicación, transacciones de negocio, reflejo de la transaccionalidad al acceder a un mecanismo de persistencia, concepto de unit of work.
- Reflejo de cuestiones de performance y concurrencia al acceder a un mecanismo de persistencia desde un programa, lazyness, cache, versionado, lockeo optimista y pesimista.

3.3.14. Laboratorio de Sistemas Operativos y Redes

- 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.
- Computación orientada a redes. Sistemas colaborativos.

3.4. Licenciatura en Desarrollo de Software – Asignaturas del Núcleo Básico – Contenidos mínimos

3.4.1. Análisis Matemático

- Funciones.
- Límite.
- Continuidad.
- Derivada.
- Aplicaciones del teorema del valor medio.
- Integral definida.
- Métodos de integración.
- Regla de L'Hôpital.
- Aplicaciones de la integral en una variable.

3.4.2. Lógica y Programación

- Lógica Proposicional: Lenguaje, Semántica, Mecanismo Deductivo, Metateoremas, Lógica trivaluada.
- Lógica de Primer Orden: Lenguaje, Semántica, Sistema axiomático, Metateoremas, Indecidibilidad.
- Programación lógica: Resolución en lógica de primer orden, PROLOG.
- Fundamentos de inteligencia artificial simbólica y no simbólica.
- Especificación de Programas: Especificación e implementación de programas, Lógica de Hoare, Corrección de programas. Verificación de algoritmos.

3.4.3. Elementos de Ingeniería de Software

- Teoría general de sistemas. Sistemas de información.
- 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, heurísticas 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.
- Ingeniería de Software de sistemas de tiempo real.

3.4.4. Seguridad de la Información

- Introducción a la Seguridad de la Información. Conceptos fundamentales y objetivos. Gestión de la Seguridad de la Información. Riesgo: análisis y tratamiento.
- Conceptos de Criptografía. Criptografía Simétrica y Asimétrica. Algoritmos de Hash. Infraestructura de Clave Pública. Certificados digitales.
- Seguridad en Redes. Objetivos. Ataques, Servicios y Mecanismos de Seguridad. Seguridad en Redes Inalámbricas. Control de Acceso Lógico. Controles físicos de seguridad: seguridad en el centro de cómputos.
- Seguridad en las operaciones. Gestión de usuarios. Control de cambios. Métodos de Evaluación de seguridad: Auditorías, Evaluaciones funcionales, Vulnerability Assessment y Penetration Test. Gestión de Incidentes.
- Seguridad en Aplicaciones. Vulnerabilidades. Software malicioso. Problemática de las aplicaciones WEB.
- Leyes, Regulaciones y Estándares. Marcos legales nacional e internacional.

3.4.5. Matemática III

- Polinomios.
- Números complejos.
- Polinomio de Taylor para funciones de una variable.
- Conceptos de cálculo diferencial e integral en varias variables: límite doble, continuidad, derivada parcial y direccional, integrales dobles.
- Fórmula de Taylor en dos variables.

3.4.6. Programación Concurrente

- Los porqués de la concurrencia. Concurrencia vs paralelismo.
- Modelo de memoria compartida, atomicidad e independencia.
- Secciones críticas, locks y barriers, semáforos, monitores y condition variables, Rendezvous.
- Problemas de la concurrencia: Starvation, Deadlocks, Liveness y Progress, Safety, Race conditions, Fairness.
- Modelo de pasaje de mensajes: Comunicación sincrónica vs comunicación asincrónica, Modelo de transacciones.
- Modelos de interacción: Cliente/Servidor, Productor/Consumidor.
- Aplicación de los conceptos estudiados en lenguajes de programación concretos, mecanismos de sincronización.

3.4.7. Gestión de Proyectos de Desarrollo de Software

- Planificación y estimación de proyectos de software. Diferencias entre las estimaciones de esfuerzo, tiempo y costo. Presupuestos.
- Definición y documentación de las actividades. Priorización de actividades: por valor asociado, por dificultad, por nivel de riesgo. Secuenciación de actividades: secuenciación por dependencias, diagramas de Gantt; secuenciación en iteraciones, conceptos de *sprint* y *backlog*. Asignación de recursos.
- Monitoreo y seguimiento de proyectos de software. Estrategias para detección y corrección de desvíos.
- Nociones de aseguramiento de calidad.
- Gestión del equipo de trabajo. Selección de los miembros del equipo. Asignación de tareas. Liderazgo. Resolución de conflictos. Capacitación.
- Comunicación. Herramientas de colaboración y comunicación interna entre los miembros del equipo. Gestión del conocimiento compartido.
- Gestión de riesgos. Identificación, dimensionamiento. Planificación de la respuesta a los riesgos: mitigación, planes de contingencia. Seguimiento y Control de Riesgos
- Gestión de adquisiciones y subcontrataciones.
- Gestión de la relación con el cliente. Modelos de contratos más comunes: contratos de precio fijo, por tiempo y asignaturales. Modelos de participación del cliente al proceso de desarrollo. Comunicación y resolución de conflictos. Control de cambios.

3.4.8. Práctica del Desarrollo de Software

- Validación y testing como un proceso continuo que se lleva a cabo durante todo el ciclo de vida del software, desde que se comienza a programar hasta que, luego de ser implementado y utilizado, el sistema se vuelve obsoleto.
- Tests de integración. Problemática específica para la automatización de tests de integración, persistencia, interfaz de usuario.

- Técnicas para diagnóstico de problemas: stacktraces, breakpoints, watchpoints. Manejo de excepciones. Relación con unit testing.
- Reingeniería de software. Técnicas de refactorización sobre un proyecto funcionando. Migraciones y actualizaciones a los modelos de datos. Compatibilidad hacia atrás.
- Herramientas metodológicas y conceptuales para trabajo en grupo. División de tareas planeando reunión de los resultados. Aprovechamiento de conceptos de objetos, citamos como posibles ejemplos: interfaces como forma de coordinar la tarea de distintas personas o grupos, *mock objects* para simular los objetos de otros grupos, etc..
- Versionado y compartición de programas fuente. Repositorios de código centralizados y distribuidos. Técnicas para la modificación de una misma base de código por múltiples desarrolladores en forma concurrente. Resolución de conflictos.
- Versionado y compartición de bibliotecas y ejecutables. Administración de entregables y dependencias. Repositorios de bibliotecas.
- Integración continua. Automatización de procesos en desarrollos de envergadura, como integración, compilación, verificación, versionado, despliegue, entre otros.
- Control de cambios. Trazabilidad de requerimientos, errores y cambios de funcionalidad. Herramientas para la administración integral de cambios y correcciones.
- Aplicación e integración de las técnicas, prácticas y herramientas aprendidas en un proyecto mediano de desarrollo de software.

3.4.9. Probabilidad y Estadística

- Estadística descriptiva.
- Modelos determinísticos y estocásticos.
- Distribución de probabilidades sobre un espacio muestral.
- Variables aleatorias discretas y continuas.
- Distintos tipos de distribuciones.
- Inferencia estadística.
- Intervalos de confianza.
- Varianza.
- Regresión lineal.
- Coeficientes de correlación.
- Ensayos de hipótesis.
- Números aleatorios.
- Método Montecarlo.

3.4.10. Lenguajes Formales y Autómatas

- Lenguajes y gramáticas
- Clasificación de Chomsky
- Lenguajes regulares. Autómatas.
- Expresiones regulares.
- Minimización de autómatas.
- Analizadores lexicográficos.
- Lenguajes independientes de contexto.
- Árboles de derivación.
- Autómatas de pila.
- Lenguajes determinísticos.
- Lenguajes tipo 1 y tipo 0. Máquinas asociadas.

3.4.11. Algoritmos

- Noción de algoritmo, ejemplos de algoritmos (criba de Eratóstenes, mcd, etc). Criterios de selección de un algoritmo.
- Notación O y W. Análisis teórico del tiempo de ejecución de un algoritmo Análisis práctico del tiempo de ejecución de un algoritmo.
- Algoritmos Divide y Vencerás. Análisis de procedimientos recursivos.
- Algoritmos Basados en Programación Dinámica.
- Algoritmos Greedy.
- Algoritmos de Precondicionamiento y Transformación del Dominio.
- Algoritmos de programación matemática, heurísticas. Algoritmos numéricos y propagación de errores.
- Casos: algoritmo de Huffman, encriptación, compresión, búsqueda, actualización, ordenamiento, estructuras de datos y algoritmos, árboles estrella, matrices.
- Algoritmos sobre grafos (DFS, BFD, Prim, Kruskal, Dijkstra, Floyd, sort topológico, etc).
- Algoritmos básicos sobre cadenas: matching, alineamiento, sufijos.

3.4.12. Ingeniería de Requerimientos

- Estrategias para la extracción de requerimientos.
- Herramientas conceptuales para la organización de requerimientos en modelos. Análisis basado en casos de uso. Análisis orientado a objetos. Estructuración mediante reglas de negocio, invariantes de clase, workflows, entre otros. Nociones sobre métodos formales.
- Estrategias de análisis en metodologías ágiles.

- Validación de requerimientos, relación con testing. Herramientas para la automatización de tests de aceptación.
- Definición de requerimientos no funcionales: performance, escalabilidad, flexibilidad, usabilidad, testeabilidad, robustez, seguridad, etc. Variación del comportamiento del sistema a lo largo del tiempo, diferentes formas de distribución. Comportamiento típico y picos de utilización. Métricas utilizadas.

3.5. Licenciatura en Desarrollo de Software – Asignaturas del Núcleo Avanzado – Contenidos mínimos

3.5.1. Teoría de la Computación

- Máquinas de Turing. Máquinas Algorítmicas.
- Problemas computables y no computables.
- Problema de la parada.
- Problemas tratables e intratables.
- Conjuntos decidibles, conjuntos r.e., reducciones many-one.
- Clases L, P, PSPACE, NP, NP - completitud.

3.5.2. Programación con objetos III

- Introducción a los sistemas de tipos y chequeo de tipos en un lenguaje de programación con objetos: tipos nominales y estructurales, tipado explícito e implícito. Duck typing. Inferencia de tipos. Esquemas de binding, early / late binding.
- Variantes del paradigma de objetos. Bloques y closures. Non-local returns. Herencia simple y múltiple; mixins y traits. Programación orientada a objetos basada en prototipos. Introducción a la programación orientada a aspectos. Open classes. Extensiones al paradigma de objetos mediante la introducción de conceptos provenientes del paradigma funcional.
- Desarrollo de aplicaciones sencillas utilizando las variantes del paradigma de objetos. Construcción de programas multilenguaje y multiparadigma. Implicancias en el diseño, patrones de diseño en las diferentes variantes del paradigma, behavioral completeness.
- Metaprogramación, programación reflexiva, introspección, self-modification. *Mirrors*.
- Lenguajes específicos de dominio (DSL). Clasificación de los DSLs: compilados, interpretados; traductores; embebidos. Creación de DSLs. Programación declarativa.

3.5.3. Arquitectura de Software I

- Arquitectura de software y arquitectura de sistemas. Definición, objetivos y clasificación.
- Actividades en un proyecto de software relacionadas con la arquitectura. Proceso de definición y evolución de una arquitectura en diferentes metodologías de desarrollo. Arquitectura en metodologías ágiles. Construcción de prototipos como herramienta de verificación y documentación de una arquitectura.

- Insumos para la definición de arquitectura: requerimientos funcionales y no funcionales, restricciones, influencias, entorno social y técnico, estándares, herramientas disponibles. Objetivos de una arquitectura: no intrusividad, no duplicación, separación de responsabilidades, garantía de atributos de calidad, robustez.
- Estilos arquitectónicos. Arquitecturas en capas, arquitecturas orientadas a servicios, arquitecturas orientadas a objetos, arquitecturas orientadas a procesos (BPM). Patrones arquitecturales. Limitaciones del paradigma de objetos y las arquitecturas tradicionales. Programación declarativa.
- Arquitectura de dominio. Modelado de la lógica de dominio. Patrones. Diseño guiado por el dominio (DDD). Motores de reglas. Workflows.
- Integración de los componentes de una arquitectura: lógica de dominio, interfaz de usuario, persistencia, seguridad, etc. Integración basada en aspectos (AOP). Acoplamiento y comunicación entre los componentes de la arquitectura. Transporte de la información. Manejo de transacciones. Tácticas para garantizar robustez y modificabilidad.
- Patrones arquitecturales para la interfaz de usuario. Integración con el dominio. Internacionalización.
- Arquitectura de persistencia. Impacto de la persistencia sobre un diseño orientado a objetos. Bases de datos multidimensionales, data warehouse, data mining.
- Integración de aplicaciones. Clasificación de los mecanismos de integración: base de datos, dominio, servicios, interfaz de usuario. Estrategias de integración apropiadas para ambientes compatibles e incompatibles entre sí. Integración sincrónica y asincrónica. Colas de mensajes. *Callbacks*. Arquitecturas orientadas a servicios. *Web services*. Integración con aplicaciones *legacy*. Patrones para la integración: punto a punto, middleware, *Enterprise Service Bus*. Definición de interfaces y conectores. Definición de procesos de negocio. Coreografías y orquestación. Manejo de transacciones y compensaciones. Servicios de directorio (JNDI, UDDI, etc).
- Configuración. Reemplazo de configuración por convenciones. Inyección de dependencias. Inversión de control. Contenedores y microcontenedores. Arquitecturas extensibles. Arquitecturas basadas en *plugins*. Lenguajes de scripting.
- Arquitecturas de seguridad Integración de métodos de autenticación y autorización en una aplicación. Patrones. *Single Sign-on*. Acceso basado en roles. Perspectivas de seguridad de una aplicación: seguridad web, sistema operativo, base de datos, *middleware*.
- Diseño de APIs y Frameworks. *Fluent interfaces*.
- Estrategias de verificación de arquitecturas. Procesos formales de evaluación de la arquitectura y de los requerimientos no funcionales. Aseguramiento de la adecuación de un sistema a la arquitectura definida, automatización de aseguramiento. Herramientas arquitecturales para la automatización de pruebas de dominio.
- Arquitecturas concurrentes y distribuidas. Objetos distribuidos. Máquinas virtuales distribuidas. Programación orientada a agentes.
- Herramientas tecnológicas para soportar las decisiones arquitectónicas.
- Cuestiones organizacionales, humanas y sociales relativas a la arquitectura de software. Relación entre la arquitectura y el grupo de desarrollo. Comunicación de la arquitectura: modelos, vistas y perspectivas. Herramientas y prácticas que complementan una arquitectura para poder llevar adelante un desarrollo grande y/o complejo.

3.5.4. Sistemas Distribuidos

- Introducción a los sistemas de procesamiento distribuido y su terminología.
- Comunicación en sistemas distribuidos, pasaje de mensaje y llamadas a procedimiento remoto (rpc)
- Tiempo, Sincronización y Coordinación Distribuida.
- Memoria compartida distribuida, asignación de tareas y balance de cargas (Algoritmos básicos)
- Manejo de archivos distribuidos

3.5.5. Características de Lenguajes de Programación

- Lenguajes según su modelo de cómputo:
 - Modelos de cómputo imperativo, funcional, objetos, lógico
 - Realización de estructuras de datos en los diferentes paradigmas
- Lenguajes según sus características:
 - Lenguajes tipados y no tipados. Sistemas de tipos
 - Mecanismos de binding (estático y dinámico)
 - Mecanismos de pasaje de parámetros (valor, referencia, nombre, otros)
 - Formas de llevar a cabo la ejecución (compilación, interpretación, máquinas virtuales)
 - Formas de administración de memoria (explícita y garbage collection)
- Lenguajes según su propósito:
 - Lenguajes de propósitos generales
 - Lenguajes de dominio específico
 - Lenguajes de scripting
 - Aptitudes de diferentes lenguajes para diferentes tareas (claridad, eficiencia, modificabilidad, etc.)
- Lenguajes según la forma de asignarles significado
 - Herramientas de asignación de significado (semánticas operacional, denotacional, axiomática)
 - Casos específicos de semántica operacional, ilustrando modelos de cómputo y características.

3.5.6. Arquitectura de Software II

- Escalabilidad, eficiencia y efectividad.
- Técnicas para dimensionar los requerimientos de hardware de un sistema: capacidad de procesamiento, espacio en memoria, almacenamiento, etc.
- Dimensionamiento de las necesidades de red de un sistema de software: ancho de banda promedio, picos de utilización, cantidades de usuarios totales, sesiones simultáneas, pedidos concurrentes.

- Técnicas para escalamiento vertical y horizontal. *Clustering*, balanceo de carga, afinidad, *sharding*. Estrategias de particionamiento de bases de datos.
- Tolerancia a fallos. Replicación de estado global y por pares. Comparación de las técnicas utilizadas para obtener tolerancia y *performance*.
- Estrategias de *cache* de datos: cacheo en el cliente, contenido estático, contenido precalculado, distribución de contenido (CDN).
- Hardware específico para sistemas de gran envergadura. Virtualización. Granjas de servidores. *Cloud computing*. Nubes privadas y públicas. Software y hardware como servicios.
- Verificación del cumplimiento de los requerimientos no funcionales: performance, tolerancia a fallos, carga. Automatización. Evaluación del comportamiento del sistema más allá de las condiciones normales de funcionamiento.
- Operación y monitoreo de sistemas. Estrategias de *logging* para sistemas de gran envergadura. Herramientas para medición de *performance*. *Profiling*. Información caliente e información de ciclo de vida largo. Análisis de servicios en red, análisis de tráfico. Herramientas de monitoreo de fallas.

3.5.7. Arquitectura de Computadoras

- Jerarquias de memoria: Memoria segmentada, Memoria virtual.
- Interrupciones: Concepto y definición. Tipos de interrupción. Definición de entorno y contexto de un programa. Detección de interrupción: cambio de contexto. Atención de interrupciones.
- Subsistema de Entrada y salida.
- Coprocesadores (aritméticos, de video, etc).
- Procesadores de alta prestación.
- Nivel de Microarquitectura: Unidad de control, Memoria de control, Microprograma, Microinstrucciones, Cronología de microinstrucciones, Secuenciamiento de microinstrucciones
- Tipos de arquitecturas: Arquitectura RISC, arquitectura en paralelo, Pentium, arquitecturas GRID, Arquitecturas multiprocesadores.

3.5.8. Parseo y generación de código

- Estructura de compiladores. Compilación vs interpretación.
- Análisis léxico y sintáctico.
- Árboles de parsing y árboles de sintaxis abstracta
- Análisis semántico
- Generación de código

3.5.9. Aspectos Legales y Sociales

- Consideraciones generales sobre el ordenamiento jurídico.
- Elementos de las relaciones económicas-jurídicas.
- La propiedad intelectual.
- Contratos: conceptos generales.
- Contratos Informáticos en particular.
- Documento digital, firma digital y derecho de Internet.
- Régimen legal de las bases de datos
- Responsabilidad penal: delitos informáticos
- Actuación judicial del licenciado en informática. Nociones de auditoría y peritaje.

3.6. Licenciatura en Desarrollo de Software – Asignaturas del Núcleo de Orientación – Contenidos mínimos

Se consignan los contenidos mínimos de las materias del núcleo de orientación.

3.6.1. Bases de Datos II

- Cuestiones de eficiencia en el acceso a bases de datos, entre otras: transformación de consultas, *hints* al motor, trabajo sobre índices.
- Configuraciones de nivel físico en un motor de base de datos relacional, p.ej. tablespaces y replicación.
- Tipos de datos no-standard en bases de datos, como ser blobs o XML.
- Implementación física de bases de datos relacionales, en particular: manejo eficiente de archivos, implementación de índices usando árboles B y variantes.
- Conceptos básicos de Data Mining y Data Warehousing.

3.6.2. Participación y Gestión en Proyectos de Software Libre

- Cibercultura y cultura hacker. Nuevos modos de relacionarse en internet: cultura abierta, distribuida, libre, producción colaborativa en red.
- Idea de software libre, movimiento de software libre, principios, principales productos y logros.
- Participación en proyectos de software libre: fuentes de información, formas que puede asumir la participación.
- Creación de proyectos de software libre: de la idea a la formulación
- El sitio de la comunidad del proyecto: forjas de software libre y otros espacios de trabajo colaborativo.
- Herramientas para el desarrollo de un proyecto de software libre, en particular: herramientas de comunicación del proyecto, de análisis y diseño y desarrollo de aplicaciones, de gestión de código y control de versiones, de gestión de la documentación
- Gestión de la admisión de contribuciones, requerimientos, errores y parches.
- Etiqueta en la comunicaciones electrónicas en el marco de los proyectos.
- Motivaciones de los desarrolladores y de los grupos de software libre.
- Roles usados más frecuentemente, mecanismos de decisión dentro del proyecto.
- Bifurcaciones de proyectos, conexiones entre proyectos, cierre de proyectos.
- Licencias para obras intelectuales, en particular para software y para su documentación técnica asociada. Licencias de software libre. BSD. GNU. Mozilla.
- Experiencia concreta de participación en al menos un proyecto existente

3.6.3. Introducción a la Bioinformática

- Conceptos básicos de la genética molecular: leyes de la herencia, genética de poblaciones, genética evolutiva, replicación del ADN, mutación y reparación.
- Acceso remoto a bancos de datos, bancos genéticos.
- Análisis de secuencias biológicas, algoritmos asociados.
- Homologías secuenciales y estructurales.

3.6.4. Políticas Públicas en la Sociedad de la Información y la Era Digital

- Estado y políticas públicas.
- Cultura abierta, distribuida, libre, producción colaborativa en red.
- Derechos en la sociedad de la información.
- Diferentes iniciativas públicas referentes a los estándares abiertos y al software libre.
- Diversidad e identidad culturales, diversidad lingüística y contenidos locales.
- Sociedad de la información y el conocimiento.
- Proyectos de infraestructura y accesibilidad TICs.
- Acceso y usos: de la red, de los contenidos.
- Datos abiertos, gobierno electrónico, gobierno abierto, democracia electrónica. Planteo y eventual desarrollo de algún software relacionado con esta temática.
- Neutralidad en la red.

3.6.5. Sistemas de Información Geográfica

- Introducción a los Sistemas de Información Geográfica (GIS): objetivos, principales tecnologías utilizadas.
- Posicionamiento: coordenadas, sistemas de referencia, proyecciones, datums, precisión.
- Modelos de datos: vectorial, raster, interpolación, implementaciones de formatos (SHP, GeoTIFF, KML, otros).
- Bases de datos espaciales: tipos de datos, consultas, índices.
- Servidores de Mapas: protocolos, en particular WMS y WFS; tecnologías. Clientes de Mapas: protocolos y tecnologías.
- Sistemas de Información Geográfica de Escritorio.
- Implementación de GIS con tecnologías OpenSource: servidor de Bases de Datos, servidor de Mapas, clientes Desktop y Web.

3.6.6. Herramientas Declarativas en Programación

- Enfoques imperativo y declarativo de la programación informática, sus diferencias, consecuencias de adoptar un enfoque declarativo.
- Bases del paradigma de programación lógico: describir un programa definiendo relaciones, concepto de cláusula, inversibilidad, principio de universo cerrado.
- Posibilidad de utilizar conceptos de la programación lógica en entornos de objetos o procedurales, programación de motores de reglas.
- Posibilidad de combinar características de los paradigmas funcional y de objetos: manejo de la estrategia de evaluación, objetos que representan funciones.
- Aplicación de un enfoque declarativo en la construcción de interfaces de usuario: separación de los detalles de visualización, generación de la interfaz a partir de un modelo de objetos a renderizar.
- Lenguajes de dominio específico (DSL): concepto, separación entre especificación de dominio y código común, modelo semántico, experimentación con herramientas concretas.

3.6.7. Introducción al Desarrollo de Videojuegos

- Panorama de la historia y estado corriente de la industria de videojuegos.
- Diversidad de videojuegos, géneros mejor establecidos.
- Concepto de game design, relevancia del relato al pensar el concepto de un juego.
- Aspectos generales en la concepción de videojuegos: estilos visuales, relevancia de la experiencia interactiva del usuario, necesidad de testeo subjetivo, pertinencia de conceptos de modelado físico.
- Cuestiones de arquitectura de software y hardware pertinentes para el dominio de videojuegos: game loop, arquitecturas P2P o cliente-servidor para juegos multiplayer, necesidad de sincronización de estados en distintas terminales.
- El proceso de desarrollo de videojuegos, pertinencia de aplicar conceptos ágiles.
- Características y bondades del modelado de un juego utilizando los conceptos de la programación con objetos: modelado del dominio en función del game design, modelado del comportamiento aprovechando el polimorfismo, modelado del flujo interactivo usando estados.
- Relevancia del procesamiento de eventos en varios géneros de juegos.
- Cuestiones ligadas al tratamiento de gráficos: uso extensivo de bibliotecas gráficas y buenas prácticas para su integración en una arquitectura de software, sprites, meshes, frustum, cálculo de colisiones.

3.6.8. Derechos de Autor y Derecho de Copia en la Era Digital

- La arquitectura jurídico-política del derecho de autor y derecho de copia.
- El derecho de autor y derecho de copia y su relación con el cambio tecnológico.
- Los derechos personales/morales y los derechos patrimoniales de autor.
- Propiedad intelectual. Patentes, marcas y logotipos.

- Las relaciones laborales y las presunciones legales sobre la titularidad de las obras.
- Las obras intelectuales, sus formas de expresión en soportes y la duplicidad de sus regulaciones
- El derecho de copia como construcción jurídico-política.
- Las licencias abiertas / libres, recíprocas/permisivas/mixtas, el concepto del copyleft, el sistema de licencias abiertas / libre de Creative Commons, otras licencias.
- El software libre, el software de fuente abierta (open source), software privativo y software privado o no publicado.
- Dominio público.

3.6.9. Análisis Estático de Programas y Herramientas Asociadas

- Análisis de flujo de datos. Análisis intraprocedural. Análisis interprocedural. Análisis de forma.
- Análisis basado en restricciones. Análisis abstracto 0-CFA y 0-CFA dirigido por sintaxis.
- Sistemas de tipos y efectos. Análisis de flujo de control. Inferencia de tipos. Efectos. Comportamiento.

3.6.10. Semántica de Lenguajes de Programación

- Definiciones inductivas. Principios de inducción.
- Semántica operacional y denotacional de lenguajes imperativos. Órdenes parciales completos. Equivalencia.
- Semántica axiomática de lenguajes imperativos. Aserciones. Corrección. Reglas de Hoare.
- Conceptos básicos de teoría de dominios. CPOs, productos, espacio de funciones, lifting, sumas.
- Semántica operacional y denotacional de lenguajes funcionales. Estrategias call-by-value, call-by-name, equivalencia.

3.6.11. Seminarios

Se trata de cursos sobre temáticas específicas correspondientes a las características dinámicas del ámbito de la programación, relacionadas con

- temas avanzados de programación.
- dominios o tipos específicos de proyectos de software.
- herramientas que cuenten con un real interés para la complementación de la formación de los estudiantes.

3.6.12. Seminarios sobre Herramientas o Técnicas Puntuales

Se trata de cursos que brindan al estudiante la posibilidad de conocer y experimentar con herramientas o técnicas de programación de especial interés para determinados dominios de aplicación.

3.7. Talleres de formación humanística – Contenidos mínimos

3.7.1. Taller de Trabajo Intelectual

- Sistematización de la información científico-técnica, económica y cultural.
- Bancos de datos. Acceso y métodos de búsqueda. Métodos de indexación y archivo de la información de interés.
- Técnicas de trabajo intelectual. Técnicas de comunicación oral y escrita (estilo y redacción de revisiones e informes, edición, audiovisuales).

3.7.2. Taller de Trabajo Universitario

- Sistemas de cogobierno universitario.
- Ley de Educación Superior.
- Estatuto. Organigrama de la Universidad.
- Centros de Estudiantes.
- Reglamentaciones.
- Problemáticas universitarias.
- Sistema de becas y pasantías.

3.8. Contenidos mínimos de los niveles de idioma Inglés

3.8.1. Inglés I

- Comunicación oral y escrita sobre la base de temáticas profesionales preferentemente, comprensión de textos y producción de textos orales y escritos.
- Formación del vocabulario técnico.
- Práctica intensiva de traducción con referencia especial a obras profesionales.

3.8.2. Inglés II

- Presentación de textos profesionales considerando con el Inglés como idioma vivo, como útil de trabajo; para ello se utilizarán textos tomados de las asignaturas de la carrera así como tomados de la red Internet. El contenido gramatical no recorre todo el abanico de estructuras posibles de Inglés general sino que se restringe a las esenciales y típicas en un contexto científico.
- Formación del vocabulario técnico, de traducción con referencia especial a obras profesionales. Comprensión de discursos orales vinculados con la vida profesional.