Consejo Federal de Educación

Resolución CFE Nº 148/11

Buenos Aires, 31 de agosto de 2011

VISTO el artículo 38 de la Ley de Educación Nacional Nº 26.206, los artículos 33, 38, 39, 42 inciso d), 43 incisos b) y c), 45 inciso e), 46, 47 y 49 de la Ley de Educación Técnico Profesional Nº 26.058, la Resolución CFCyE Nº 261/06 y las Resoluciones CFE Nº 15/07, Nº 77/09, N° 91/09 y N° 107/10 y,

CONSIDERANDO:

Que conforme la Ley de Educación Nacional la Educación Técnico Profesional se rige por las disposiciones de la Ley Nº 26.058.

Que la Ley de Educación Técnico Profesional establece que el MINISTERIO DE EDUCACIÓN a través del Instituto Nacional de Educación Tecnológica (INET) y con participación jurisdiccional, garantizará el desarrollo de los marcos de referencia y el proceso de homologación para los diferentes títulos y/o certificaciones profesionales, para ser aprobados por el CONSEJO FEDERAL DE EDUCACIÓN.

Que el MINISTERIO DE EDUCACIÓN en acuerdo con el CONSEJO FEDERAL DE EDUCACIÓN, debe establecer las políticas, los criterios y parámetros para la homologación de los títulos de educación técnico profesional.

Que el INET ha llevado a cabo las acciones organizativas y técnicas necesarias en forma conjunta con la Comisión Federal de Educación Técnico Profesional, para la consulta y elaboración de los marcos de referencia para el proceso de homologación de títulos técnicos de nivel secundario y de nivel superior, donde se recuperan acuerdos federales previos y actualizaciones pertinentes, y que el Consejo Nacional de Educación, Trabajo y Producción ha tomado la intervención que le compete como órgano consultivo.

Que el documento que se presenta corresponde al marco de referencia que al momento se ha acordado en las instancias señaladas y ampliando los aprobados por el mediante Resoluciones CFE Nº 15/07, Nº 77/09, N° 107/10 y N° 129/11.

Que los marcos de referencia intervienen en el proceso de homologación con los propósitos de dar unidad nacional y organicidad a la educación técnico profesional, respetando la diversidad federal de las propuestas formativas, garantizar el derecho de

los alumnos y egresados a que sus estudios sean reconocidos en cualquier jurisdicción, promover la calidad, pertinencia y actualización permanente de las ofertas de educación técnico profesional, facilitar el reconocimiento de los estudios de los egresados por los respectivos Colegios, Consejos Profesionales y organismos de control del ejercicio profesional; y, como instrumentos para llevar a cabo las acciones de análisis y de evaluación comparativa de los títulos y sus correspondientes ofertas formativas que se presenten a homologar.

Que es necesario aclarar, entonces, que los marcos de referencia en tanto instrumentos para la homologación de títulos y certificados de la educación técnico profesional, no constituyen planes de estudio y deben operar en el ámbito de las carteras educativas jurisdiccionales.

Que la presente medida se adopta con el voto afirmativo de todos los miembros de esta Asamblea Federal, a excepción de las provincias de Corrientes, Chubut, Río Negro y San Juan, por ausencia de sus representantes.

Por ello,

LA XXXVI ASAMBLEA DEL CONSEJO FEDERAL DE EDUCACIÓN

RESUELVE:

ARTÍCULO 1º.- Aprobar el marco de referencia de "Técnico en Programación" de nivel secundario, que como anexo I forma parte de la presente medida.

ARTÍCULO 2º.- Establecer que las jurisdicciones tendrán un plazo de dos años para iniciar el proceso de homologación de títulos y planes de estudio correspondientes al marco de referencia aprobado, conforme el artículo 3º de la Resolución CFE Nº 91/09.

ARTÍCULO 3º.- Regístrese, comuníquese, notifíquese a los integrantes del CONSEJO FEDERAL DE EDUCACIÓN y cumplido, archívese.

Fdo:

Prof. Alberto Sileoni – Ministro de Educación de la Nación

Prof. Domingo de Cara – Secretario General del Consejo Federal de Educación

Res. CFE Nro. 148/11
Anexo I

Marco de referencia para procesos de homologación de títulos del nivel secundario

Sector Informático

Índice

1. Identificación del título

1.1. Sector/es de actividad socio productiva: Informática [1] (Software y servicios informáticos)

1.2. Denominación del perfil profesional: Técnico en Programación

1.3. Familia profesional: Informática

1.4. Denominación del título de referencia: Técnico en Programación

1.5. Nivel y ámbito de la trayectoria formativa: nivel Secundario y ámbito de la Educación Técnica de la modalidad de Educación Técnico profesional.

2. Referencial al Perfil Profesional

El perfil profesional del Técnico en Programación alude al conjunto de realizaciones profesionales que el técnico puede demostrar en las diversas situaciones de trabajo propias de su área ocupacional, una vez que ha completado el proceso formativo.

Este perfil involucra un conjunto de competencias que asegura un mayor nivel de especificidad y profundización en ámbitos contextualizados del saber, saber hacer y saber ser, dentro del sector profesional de la Informática. Se articula en torno a núcleos curriculares comunes y orientados, a partir de las demandas socio-productivas del sector y la realidad del medio industrial y de servicios.

2.1. Alcance del Perfil Profesional

> El Técnico en Programación estará capacitado para realizar programas o componentes de sistemas de computación – interpretar especificaciones de diseño, documentar los productos realizados, verificar los componentes programados, buscar causas de malfuncionamiento y corregir los programas o adaptarlos a cambios en las especificaciones – desarrollando las actividades descriptas en el perfil profesional y cumpliendo con los criterios de realización establecidos para las mismas en el marco de un equipo de trabajo organizado por proyecto.

Este Técnico en Programación participa en proyectos de desarrollo de software desempeñando roles que tienen por objeto producir programas, módulos o componentes de sistemas de computación. Estos módulos suelen integrarse en aplicaciones que interactúan con otras ya existentes desarrolladas con la misma o diferente tecnología.

Sus actividades profesionales cubren las siguientes áreas:

"Interpretar especificaciones de diseño o requisitos de las asignaciones a programar",

en el contexto del proyecto. Convalida su propia interpretación con quienes la hayan realizado o provisto.

"Planificar su trabajo en el contexto del equipo de desarrollo del proyecto y de la tecnología a utilizar".

Identifica aspectos de posible dificultad o riesgo, evalúa las características del entorno, tiempos y esfuerzos para lograr la solución del problema, considera la posibilidad de subdividir la asignación en pasos o componentes menores y establece un orden o secuencia de trabajo.
Analiza estrategias para desarrollar la asignación recibida e investiga para refinar aspectos de diseño, algoritmos o estructuras de datos, busca componentes disponibles y adecuados y plantea soluciones alternativas para ser evaluadas en el contexto del proyecto.

"Producir programas, módulos o componentes de sistemas de computación",

[1] Este técnico desempeña sus actividades en proyectos de desarrollo de software. La instrumentación del Catálogo Nacional de Títulos (Ley 26.058) determinará la denominación final del sector en el cual esta formación deba incluirse.

respondiendo a necesidades específicas en el contexto de la tecnología a utilizar. Para ello desarrolla algoritmos que den solución a los problemas a resolver y define estructuras de datos adecuadas a los mismos. También construye o modifica clases y objetos, reutiliza componentes existentes o diseña interfaces. Modifica códigos ya escritos para corregir errores o para cambiar funcionalidades o comportamientos de productos existentes.
Integra los componentes programados en aplicaciones que interactúan con otras ya existentes desarrollados con las mismas o diferentes tecnologías.

"Verificar y depurar el producto desarrollado ",

para asegurarse que cumple con las especificaciones recibidas. Implica la implementación de un conjunto de pruebas para detectar comportamientos o resultados no previstos y buscar sus causas. Comprende revisar códigos para encontrar las partes o instrucciones que provocan los malfuncionamientos y definir las acciones correctivas.

"Realizar revisiones cruzadas de código o de interfaces",

con otros programadores o con especialistas, para evaluar el uso eficiente de recursos y del ambiente de desarrollo, y aporten observaciones con propuestas de cambio con el objeto de mejorar la calidad, mantenibilidad y eficiencia del producto.

"Realizar la documentación técnica y de usuarios de acuerdo con los requerimientos funcionales y técnicos de las aplicaciones y sistemas".

Documenta su trabajo para que resulte interpretable y utilizable por otros. Esto incluye comentar el código, complementar documentos de diseño, confeccionar o completar reportes de incidentes, adjuntar resultados de pruebas o advertencias sobre posibles limitaciones de la solución. También incluye la identificación de las versiones producidas.

"Explotar las funcionalidades de los sistemas de información, hardware, software y redes",

para la realización de las actividades. Implica conocer y saber utilizar eficientemente recursos de hardware, software y redes para utilizar los ambientes que necesite para el desarrollo su trabajo.

La actividad del programador es no rutinaria a pesar de que muchas veces se reutilicen partes ya existentes. Cada asignación representa la necesidad de dar satisfacción a determinados requisitos. Ello requiere comprender el problema y la arquitectura en la que estará inserta la solución, idear estrategias de resolución y ser capaz de aplicar debidamente el lenguaje y ambiente de programación a emplear, así como aplicar buenas prácticas de programación, lo que incluye documentar decisiones significativas de diseño y las limitaciones que tendrá el artefacto construido.

Para poder desarrollar plenamente su profesionalidad, el técnico tiene que poseer ciertas capacidades que resultan transversales a todas sus funciones y tienen que ser desarrolladas durante el transcurso de su formación. Estas son:

Abstracción - Implica descartar o reducir detalles poco significativos de la información sobre un problema para concentrarse en pocos elementos por vez, lo que resulta en una reducción de la complejidad que permita conceptualizar de modo más simple un dominio de problemas para facilitar su comprensión y manejo en forma genérica de sus posibles soluciones.

Pensamiento combinatorio - Conduce a la consideración sistemática de un conjunto de alternativas, lo que incluye el manejo mental de muchas variables o detalles del problema sin perder nunca de vista el concepto o la estrategia general de resolución.

Autorregulación - Implica manejarse respetando reglas y limitaciones, tanto explícitas como implícitas, sean éstas propias o del grupo de trabajo; actuar ateniéndose a un orden propio que le facilite el acceso a lo que puede necesitar, reconocer y guardar; referenciar la información y registrarla de tal manera que le facilite acceder posteriormente en forma rápida para evaluarla y recuperarla.

Comunicación apropiada - Implica una disposición a reconocer que existen otros que pueden aportar información útil o a quienes puede interesarle lo que hace. Supone reconocer su rol y el de cada integrante del proyecto, transmitir la información necesaria en forma precisa y en un lenguaje apropiado para el entendimiento mutuo en interacciones individuales o grupales, o en forma escrita, utilizando, si es necesario para ello, el idioma inglés, que debe interpretar con propiedad a nivel técnico.

Trabajo en equipo - Implica adoptar una actitud abierta, estar dispuesto a compartir información y conocimientos, a tomar en cuenta a los usuarios del producto que está construyendo, a brindar, pedir y aceptar ayuda cuando ésta resulte necesaria para facilitar su propia labor o la de otro integrante del equipo. Comprende al equipo del proyecto, incluyendo a los usuarios que participan del mismo.

2.2. Funciones que ejerce el profesional

A continuación se presentan funciones y sub-funciones del perfil profesional del técnico de las cuales se pueden identificar las actividades profesionales:

Interpretar especificaciones de diseño o requisitos de las asignaciones a programar

Esto comprende:

Analizar el problema a resolver.

Determinar el alcance del problema.

Validar la coherencia e integridad de las especificaciones.

Convalidar su propia interpretación con quienes lo hayan realizado o provisto.

El Técnico en Programación de Computadores analiza el problema a resolver, que puede estar especificado formal o informalmente como instrucciones de diseño o requerimientos del usuario. Para ello resulta necesario interpretar críticamente el material recibido y validar si todo lo pedido resulta coherente entre sí o con otros aspectos que conozca del proyecto para clarificar eventuales malas interpretaciones o desacuerdos y convalidar su interpretación con el responsable del proyecto.

Esto implica que tiene que ser analítico y poseer una buena capacidad de abstracción para ser capaz de comprender lo especificado, observando reglas de los lenguajes en que está expresado (*storyboards*, casos de uso, *UML*, otros tipos de diagramas, diccionarios de datos), describir en sus propios términos el problema, identificar puntos ambiguos, aspectos faltantes o eventuales contradicciones entre distintos requisitos a cumplir o incoherencias entre estos y otros aspectos conocidos del proyecto. También debe ser capaz de comunicarse en un lenguaje preciso y adecuado con el líder o usuario con quien discuta su interpretación para convalidarla.

Planificar su trabajo y analizar estrategias para desarrollar la asignación recibida

Esto comprende:

Identificar aspectos críticos.

Dividir la asignación en subtareas o productos intermedios.

Establecer un orden o secuencia de trabajo.

Estimar tiempos de realización.

Establecer prioridades y necesidades de apoyo y consulta para refinar aspectos ambiguos o insuficientemente conocidos del diseño.

Utilizar metodologías de búsqueda de información de fuentes confiables.

Aplicar técnicas y metodologías para la resolución de problemas.

Para realizar esto el técnico tiene que contemplar requerimientos técnicos y funcionales, a cubrir, estimar dificultades y tiempos, imaginar y desarrollar alternativas de solución a fin de organizar su tarea y prever sus tiempos y posibles dificultades.

Esto implica que tiene que ser capaz de averiguar y completar detalles de diseño, considerar si existen bibliotecas con patrones, clases, rutinas o módulos que pueda utilizar, eventualmente construir prototipos y demos para visualizar la propuesta y comparar ventajas y desventajas de las distintas alternativas para seleccionar la que considera más adecuada para planificar su tarea, anticipando posibles riesgos a enfrentar en su asignación para solicitar la colaboración o asesoramiento que corresponda.

Al hacer esto utiliza su experiencia acumulada, consulta bibliotecas o listas de discusión en Internet y arma su propio repertorio de material a utilizar.

Producir programas, módulos o componentes de sistemas de computación en el contexto de la tecnología a utilizar

Esto comprende:

Desarrollar algoritmos que den solución a los problemas asignados.

Definir el código.

Definir estructuras de datos eficaces y explotarlos con eficiencia.

Definir, desarrollar instancias y completar clases y objetos apropiados para representar el problema a resolver.

Diseñar interfaces respetando el estilo del usuario y del contexto previsto.

Para realizar esto el técnico utiliza patrones, reutiliza código existente adaptándolo o complementándolo a su nueva función o redacta código nuevo aplicando sus conocimientos de programación, respetando buenas prácticas y las normas establecidas para asegurar la calidad del proyecto. Esto implica el dominio del lenguaje y ambiente de desarrollo utilizados en el proyecto, así como la tecnología en la cual va a ser implementada la solución, así como la aplicación de criterios de simplicidad y coherencia en la elaboración de interfases.

Verificar el producto desarrollado

Esto comprende:

Analizar y registrar todos los procesos alternativos importantes.

Procesar el producto obteniendo y registrando los resultados.

Para realizar esto el técnico determina las necesidades de cobertura en función de las características de su asignación y normas establecidas para asegurar la calidad del proyecto, identifica las clases de equivalencia de datos utilizados internamente o intercambiados y diseña los casos de prueba, tomando en cuenta la estructura del artefacto y las condiciones de borde, así como prepara el entorno de pruebas, incluyendo los *scripts* y datos necesarios. Esto implica el dominio de conceptos de *testing* y de herramientas utilizadas para establecer el ambiente de *testing* unitario. Realiza las pruebas correspondientes, registrando los datos y resultados alcanzados, así como las acciones correctivas realizadas para solucionar las fallas encontradas.

Depurar estructuras lógicas o códigos de programas

Esto comprende:

Relacionar resultados insatisfactorios con los datos o porciones de código que los originaron.

Analizar estos datos y/o partes del código que causaron el mal funcionamiento y determinar el tipo de corrección o reemplazo.

Verificar que la corrección y/o reemplazo solucionen el mal funcionamiento.

Para realizar esto el técnico tiene que relacionar resultados insatisfactorios con probables causas y recorrer la estructura y código del programa para identificar el origen del error en el código que origina el mal funcionamiento. Una vez identificado el error, corresponde razonar sobre el tipo de corrección o reemplazo y analizar que el nuevo código no introduzca otros problemas.

Esta actividad se aplica tanto a programas propios como ajenos, que agregan un nivel de dificultad al no tenerse presente su estructura o no conocerse el estilo del código. También consulta a pares y al líder del equipo de trabajo para reflexionar y recibir ayuda que le permita resolver problemas encontrados o aporta sus conocimientos y capacidad de reflexión a otros, y participa de foros y listas temáticas para encontrar soluciones o elementos reutilizables.

Realizar revisiones cruzadas de código o de interfaces

Esto comprende:

Revisar el cumplimiento de estándares y de especificaciones.

Revisar las interfaces desarrolladas con otros programadores o con especialistas para evaluar el uso eficiente de recursos y del ambiente.

Reportar observaciones sobre propuestas de cambio.

Para realizar esto el técnico revisa con otros programadores o especialistas si las interfaces resultan coherentes dentro del estilo del sistema, amigables para el usuario y para personas con capacidades diferentes; que los códigos producidos no demanden tiempos de proceso, asignaciones de memoria o almacenamiento excesivos para el contexto.

Esto implica la capacidad de reconocer estructuras y un dominio del lenguaje de programación, así como el conocimiento de buenas prácticas de programación y normas de documentación. También la capacidad de trabajar en equipo y de comunicación para informar las observaciones recibidas y presentar propuestas de cambio significativas en forma verbal o escrita.

Realizar la documentación técnica y de usuarios de acuerdo con los requerimientos funcionales y técnicos de las aplicaciones y sistemas.

Esto comprende:

Describir características, relaciones y limitaciones de nuevas clases utilizando diagramas u otros elementos.

Intercalar en el código descripciones de sus características y limitaciones.

Registrar decisiones de diseño, elementos utilizados y resultados de pruebas.

Plasmar incidentes, errores, soluciones y tiempos utilizados.

Identificar cada versión del producto de acuerdo a estándares.

El técnico realizará la documentación con claridad, consistencia y completitud. Describe que hace cada parte del código y por qué se incluye, datos, otros elementos o situación que lo originaron; registros y evidencias de las actividades realizadas y de los incidentes observados, identifica cada versión de acuerdo a estándares.

Para lograr un desempeño competente en sus actividades profesionales, el desarrollador de software, además de realizar las actividades previstas en su perfil profesional e incluidas aquí en la descripción de las funciones que realiza, tiene que conocer ciertos aspectos de la tecnología de la información que le sirven de base para poder desarrollar competentemente sus funciones profesionales. Al dominio de estos aspectos lo hemos denominado:

Desempeño de base – Esto implica conocer y saber utilizar con propiedad y en condiciones de seguridad recursos de hardware, software y redes para emplear los ambientes que necesite para el desarrollo y la verificación del software, mantener los repositorios de información que necesite utilizar y disponer de los productos de su trabajo en condiciones de confiabilidad.

Esto comprende:

Configurar lógicamente el sistema al entorno de trabajo para desarrollar y probar los programas.

Organizar y mantener componentes de software y datos de prueba en sistemas de archivos, utilizando las utilidades comunes al proyecto

Recuperar, presentar y distribuir información en su estación de trabajo o a través de la red.

Respetar procedimientos propios o de la organización que aseguren la integridad, disponibilidad y seguridad del sistema y de la información durante el desarrollo y verificación de programas.

Integrar la producción propia en el conjunto del proyecto identificándolas de acuerdo a los procedimientos de administración de versiones en uso por el proyecto.

Para realizar esto, el técnico tiene que poseer un dominio de la tecnología, tanto de hardware y redes, como de software de base, así como una disciplina de trabajo que le permita organizar y administrar sus propias herramientas y repositorios de información sin afectar a las actividades de otros y entregar los productos de su labor correctamente identificados de acuerdo a lo

establecido para el proyecto, manteniendo un adecuado seguimiento de su labor que permita responsabilizarse por lo realizado,

2.3 Área Ocupacional

Este técnico se ocupa en organizaciones de diversos tipos que tengan que desarrollar software. Empresas que realizan desarrollo de software por encargo de organizaciones locales o extranjeras, que proveen software junto con otros servicios de asesoramiento y consultoría, y, en menor número, que desarrollan sus propios productos de software para vender en el país o en el exterior. También en organizaciones dedicadas a otras actividades, pero que producen el software que necesitan para desarrollar sus propias actividades o que integran en productos que venden.

El software debe satisfacer especificaciones de requerimientos, ya sean estas formales o informales, las que pueden venir dadas por el cliente, algún consultor especializado en el tipo de problemas que aborda la aplicación o ser elaboradas por algún analista funcional integrante del equipo de trabajo del proyecto. El equipo de desarrollo suele estar encabezado por un gerente o líder responsables por el proyecto e integra diversos roles ocupacionales, como el de un arquitecto de software, que establece el diseño general del sistema y especificaciones de calidad de la solución, una serie de programadores, que son quienes lo construyen y un grupo de *testing*, que son los encargados de verificar que el software producido cumpla los requisitos, tanto funcionales como de comportamiento, oportunamente establecidos. Del equipo de trabajo pueden participar uno o más analistas técnicos que se ocupan de detalles relativos a aspectos de tecnología, seguridad, bases de datos o estándares de programación y asesoran y dan apoyo técnico a los desarrolladores. Eventualmente pueden participar diseñadores gráficos y especialistas en otros aspectos específicos.

La posición ocupacional de este técnico suele denominarse analista-programador o programador, aunque últimamente se está generalizando una denominación más abarcativa y menos categorizante de desarrollador de software. Integra equipos de proyecto dedicados al desarrollo o mantenimiento de software y recibe asignaciones específicas que tiene que resolver en lapsos que suelen medirse en términos de días o semanas, produciendo artefactos que satisfagan especificaciones y se integren al sistema objeto del proyecto.

A partir de especificaciones de diseño y del conocimiento de la arquitectura del sistema, los programadores (también denominados analistas programadores o simplemente desarrolladores) completan el diseño en detalle de la parte que les fuera asignada, la construyen, preferiblemente en base a artefactos de software ya existentes y adaptando o escribiendo lo que sea necesario, así como documentándola para facilitar su testeo y posterior mantenimiento por otros, verifican unitariamente lo producido y lo entregan para ser probado integralmente e integrado al resto. Habitualmente, los desarrolladores, que pueden estar especializados en una tecnología determinada, trabajan individualmente o de a pares dentro de un grupo más numeroso, brindándose mutuamente colaboración para resolver los problemas que deben enfrentar y los que tienen mayor experiencia suelen brindar orientación (*coaching*) a los más noveles.

Resuelve estas asignaciones individualmente o trabajando en pares, recibiendo la supervisión y asesoramiento de un líder de proyecto o de grupo, con quien consulta dudas y decisiones significativas o comunica inconvenientes. También recibe apoyo y brinda colaboración a otros miembros del grupo. Asimismo, puede desempeñarse en forma autónoma, asumiendo la mayor parte de las tareas propias del proceso, sobre todo trabajando en forma independiente resolviendo problemas de pequeñas organizaciones que requieren sistemas de baja complejidad y reducida dimensión. Por otra parte, Técnicos en Programación o profesionales equivalentes con capacidad emprendedora pueden y suelen asociarse entre ellos para generar sus propias empresas para brindar servicios de desarrollo y proveer software a terceros.

De lo anterior se desprende que el Técnico en Programación desarrolla su actividad en las siguientes áreas ocupacionales:

- Servicios informáticos para pequeñas y medianas empresas en áreas de análisis y programación de desarrollo y producción de software.

- Empresas de distintos sectores de actividad económica en áreas de informática o de procesamiento de datos.
- Por cuenta propia o en pequeños emprendimientos asociativos de desarrollo y producción de software.
- Empresas de servicios de implantación y mantenimiento de sistemas informáticos.
- Comercialización de equipos y sistemas informáticos.
- Administración pública, en las áreas de mantenimiento y gestión de la información
- ONGs, en áreas vinculadas con el procesamiento de datos para la gestión.
- Mantenimiento de sistemas informáticos en entornos personales y de redes de área local.
- Asesoramiento técnico y venta de sistemas y aplicaciones informáticas.

2.4 Habilitaciones profesionales

Las actividades que realiza y para las cuales está capacitado el Técnico en Programación, así como el ámbito de su desempeño y el campo y condiciones de su ejercicio profesional son los descriptos en el Perfil Profesional correspondiente.

Si bien las actividades de este técnico no están orientadas a un tipo de software en particular, conviene tomar en cuenta que el software es utilizado crecientemente en sistemas que afectan a la seguridad pública. Estos sistemas, denominados *críticos para la seguridad*, son lo que, en un sentido general, involucran riesgos que conllevan la posibilidad de pérdidas inaceptables (daños para la salud o aún la vida humana, daños a la propiedad, contaminación ambiental, conflictos sociales, grandes pérdidas monetarias).

En función de estos riesgos, se establecen las siguientes habilitaciones profesionales, para el Técnico en Programación, con las limitaciones o exclusiones que se indican en cada caso. Estas habilitaciones tienen efecto para su desempeño en forma autónoma o asumiendo plenamente la responsabilidad por los resultados que obtenga su grupo de trabajo.

- Desarrollar y mantener programas de software de complejidad media, correspondientes a sistemas de información o vinculados indirectamente al hardware o a sistemas de comunicación de datos, respondiendo a especificaciones.

Queda excluido de esta habilitación el software correspondiente a sistemas críticos para la seguridad, como es el caso de los que involucren el procesamiento de información que conlleve riesgos efectivos para terceros. Particularmente, queda excluido el software destinado a:

  - control de equipos y procesos médicos, industriales o de domótica que puedan poner en riesgo inmediato o mediato la salud de personas;
  - procesamiento de información crítica para los individuos, como ser la que sirva para corroborar su identidad o características de su estado de salud, para demostrar situaciones legal, fiscal, patrimonial u otras que afecten a su patrimonio o a sus libertades;
  - procesamiento en línea de transacciones financieras importantes.

En estos casos, requerirá la supervisión de profesionales habilitados.

- Operar actividades de *testing* de software de aplicaciones
- Redactar documentación técnica.

3. En relación con la trayectoria formativa

Los planes de estudio a ser presentados para su homologación deberán evidenciar el trayecto formativo completo que conduce a la emisión del título técnico de nivel medio, independientemente de la organización institucional y curricular adoptada, de manera tal que permitan identificar los distintos tipos de contenidos a los que hace referencia.

Deberán identificarse los campos de formación general, de formación científico-tecnológica, de formación técnica específica y de prácticas profesionalizantes.

De la totalidad de la trayectoria formativa y a los fines de homologar títulos de un mismo sector profesional y sus correspondientes ofertas formativas, que operan sobre una misma dimensión de ejercicio profesional, se prestará especial atención a los campos de formación científico-tecnológica, de formación técnica específica y de prácticas profesionalizantes. Cabe destacar

que estos contenidos son necesarios e indispensables pero no suficientes para la formación integral.

3.1 Formación general

El campo de la formación general es el que se requiere para participar activa, reflexiva y críticamente en los diversos ámbitos de la vida social, política, cultural y económica y para el desarrollo de una actitud ética respecto del continuo cambio tecnológico y social. Da cuenta de las áreas disciplinares que conforman la formación común exigida a todos los estudiantes del nivel medio, de carácter propedéutica. A los fines del proceso de homologación, este campo, identificable en el plan de estudios a homologar, se considerará para la carga horaria de la formación integral del técnico.

3.2 Formación científico tecnológica

El campo de la formación científico-tecnológica identifica los conocimientos, habilidades, destrezas, valores y actitudes que otorgan particular sostén a las técnicas y métodos de trabajo, así como las tecnologías propias del campo profesional en que se desempeña este técnico. Estos saberes, que sustentan a la formación específica, profundizan o complementan a los de la formación general, por lo que conviene prever una adecuada articulación con los mismos, así como también resulta importante que el diseño curricular tome en cuenta su relación con los aspectos técnico específicos.

Las áreas disciplinares relacionadas con la formación científico tecnológica de la trayectoria formativa de este técnico son:

Provenientes del campo de la matemática y la lógica

La matemática y la lógica desarrollan capacidad de pensamiento abstracto, razonamiento por inferencias y análisis combinatorio de alternativas requerido por el pensamiento computacional. Además, la resolución de asignaciones de programación requiere permanentemente la necesidad de resolver problemas, a lo cual contribuyen las estrategias desarrolladas para la solución de problemas matemáticos. En consecuencia, se recomienda adoptar un enfoque práctico, basado en el planteo de problemas, para abordar los siguientes contenidos:

Conjuntos. Lenguaje coloquial, simbólico y gráfico; diagramas de Venn. Cardinalidad y numerabilidad. Operaciones con conjuntos: intersección, unión, diferencia, diferencia simétrica, complementación. Leyes de De Morgan. Particiones.

Sistemas de numeración. Concepto de número, formas de representación. Sistemas posicionales: binario, octal, hexadecimal. Operaciones en los distintos sistemas. Concepto de *overflow*.

Números reales: propiedades, operaciones, aproximación decimal, cálculo aproximado, redondeo y truncamiento y su influencia en los errores de cálculo, error absoluto y relativo. Codificación de información. Sistemas de representación, operaciones aritméticas en punto flotante, concepto de excepción.

Lógica Simbólica: Proposiciones. Conectivos lógicos. Operaciones lógicas. Equivalencia lógica. Clasificación de proposiciones según tabla de verdad: tautologías, contradicciones. Reducción de expresiones lógicas a su mínima expresión.

Lógica de circuitos digitales. Expresiones lógicas y funciones booleanas. Nociones de lógica difusa.

Demostración matemática: demostración directa, por contraejemplo, principio de inducción completa. Definiciones matemáticas recursivas. Buen ordenamiento.

Vectores y matrices, producto escalar y cartesiano, operaciones con matrices, transposición de matrices. Determinantes. Sistemas de ecuaciones lineales, métodos de resolución directos y aproximados, incompatibilidad e indeterminación. Inecuaciones.

Funciones, concepto y representación, tipos de funciones y relaciones. Funciones elementales más comunes, operaciones con funciones elementales, funciones polinómicas (operaciones con polinomios, raíces), valor absoluto, potencial, exponencial, logarítmica y trigonométricas. Representación de problemas por medio de funciones, métodos de solución de ecuaciones.

Probabilidades en espacios discretos: experimentos aleatorios, espacios muestrales, sucesos, probabilidad condicional e independencia. Combinatoria.. Variables aleatorias, distribuciones de probabilidad, esperanza matemática, varianza, ley de los grandes números. Nociones de estadística descriptiva, medidas de posición y dispersión, estimadores. Concepto de distribución de variable continua, distribuciones más comunes.

Provenientes del campo de la física

La física facilita una comprensión razonada de los objetos del mundo real y el desarrollo de modelos abstractos que representen su comportamiento. También resulta útil el conocimiento de los movimientos para la programación de animaciones. En tal sentido, además de lo que establezca el currículo general del secundario, se contempla profundizar los siguientes contenidos:

Cinemática, dinámica, conceptos de inercia y estabilidad. Rozamiento y choque elástico. Cargas atractivas y repulsivas.

Por otra parte, se descuenta que los aspectos de electricidad, magnetismo, ondas, óptica y movimientos de rotación recibirán la atención necesaria en el currículo como para facilitar la comprensión de los fenómenos y dispositivos propios de las tecnologías de la información y las comunicaciones.

Provenientes de la campo de la computación

Este campo figura tanto en la formación científico-tecnológica como en la técnico-específica. En la primera han sido previstos contenidos elementales y fundamentales de la disciplina de la computación que son de carácter general y deberían formar parte de la educación de cualquier ciudadano preparado para desenvolverse con soltura en la sociedad de la información. Estos contenidos pueden servir de motivación para quienes aspiren a desempeñarse en funciones técnicas del sector informático y conviene que se incorporen tempranamente a su formación. A la formación técnico-específica han sido asignados los que el Técnico en Programación va a poner en juego en forma directa al realizar sus actividades.

Introducción al mundo de la computación y al pensamiento computacional. La programación como forma de expresión.

Concepto de algoritmo, los pasos básicos en la resolución algorítmica de problemas (exploración y formulación del problema, examen de una muestra de casos particulares, estrategias de diseño, realización, prueba y verificación). El problema de la complejidad.

Estructuras fundamentales, variables, tipos, expresiones y asignaciones, entrada/salida, estructuras de control condicionales e iterativas, funciones y pasaje de parámetros, descomposición estructurada.

Concepto de lenguaje de alto nivel y la necesidad de traducción, comparación entre compiladores e intérpretes, aspectos de la traducción dependientes y no dependientes de la máquina. Programas generadores de código.

Como parte de la forma de adquirir estos aprendizajes y demostración práctica de los resultados alcanzados, los estudiantes tienen que utilizar lenguajes sencillos de programación que motiven el abordaje de problemas de interés para los alumnos y la práctica de pensamiento computacional. También se puede prever la integración de imágenes y sonido en estas realizaciones.

Provenientes de la tecnología de la información

Conceptos de tecnología de la información, evolución histórica, tecnologías que la integran, disciplinas que forman parte (ciencia de la computación, ingeniería de software, sistemas de información, ingeniería en computación) o aportan a la misma.

Evolución del computador, su organización y unidades funcionales que lo componen. Arquitectura interna de computadores, unidad central de procesamiento, instrucciones y flujo de la información Tipos y niveles de organización de la memoria interna y externa (sistemas de memoria, tecnologías y jerarquías, memoria caché, memoria virtual, dispositivos de almacenamiento secundario). Periféricos, clasificación y utilización; dispositivos de almacenamiento externo (discos duros, *CD*s, *DVD*s, tarjetas de memoria, otros). Estado actual de la tecnología. Procesadores multinúcleos, *threading*, computación paralela.

Funcionamiento de las instrucciones de un programa a nivel de una máquina simplificada (principalmente como medio de comprender características de su funcionamiento).

Provenientes de las organizaciones y los sistemas de información

Elementos de teoría general de los sistemas; objetivos, recursos, componentes, frontera, medio ambiente; niveles: suprasistema, sistemas pares, subsistemas; estructura, clasificación y características; enfoque sistémico de la organización. La información como recurso de las organizaciones y en el proceso de toma de decisiones: clases de decisión, proceso de toma de decisiones, características de las decisiones según niveles jerárquicos en la organización. Elementos de estructura y comportamiento de las organizaciones, los recursos que administran las organizaciones, organización estructurada por funciones o líneas de productos, el manejo de sedes.

Concepto de proceso. Procesos del ciclo de ventas y cobranzas; del ciclo de compras y pagos. Nociones de procesos de gestión y transformación de materiales y su organización. Comprobantes usuales, requerimientos legales y fiscales. Concepto de recurso y su gestión. Modelo hidráulico del movimiento y acumulación de bienes de cambio y dinero. El papel de los sistemas de información en la organización. Nociones de control interno. La contabilidad como sistema de información. Algunas características de organizaciones y procesos de servicios.

Los niveles de la administración: la planificación estratégica, el control de gestión, el control operativo y el tipo de sistemas de información asociados a los mismos.

Sistemas de información, métodos de procesamiento de datos; características, clasificación y función de la información. Función de un sistema de información. Sistemas de información típicos y aplicaciones usuales vinculados con la comercialización y distribución de bienes y servicios.

Provenientes del campo de la ética y del derecho

Importancia social y económica de los servicios de tecnología de la información, significado de Internet, valor de la información almacenada para las organizaciones, seguridad. Valor de la información para los individuos, normativa relativa a privacidad y “habeas data”. Bases de datos públicas y privadas. Propiedad de datos empresarios. Secretos comerciales e industriales.

Contexto normativo: responsabilidades empresarias, contratos, responsabilidades del trabajador, leyes de protección de datos personales, propiedad intelectual del software y de contenidos, conceptos jurídicos aplicables a delitos informáticos.

Privacidad de datos personales. Normas que rigen el correo electrónico. Protección legal de la propiedad intelectual (incluyendo software), derechos de reproducción y derechos sobre marcas y patentes. Licencias de fabricación, de uso, *GNU* y “creative commons”.

Provenientes del campo del idioma inglés

Inglés técnico. Lectura e interpretación de textos e información técnica en inglés. Comprensión y producción de textos de complejidad creciente en inglés para comunicarse solicitando o aportando información técnica por e-mail o en foros y listas de discusión.

3.3 Formación técnico-específica

La formación específica del Técnico en Programación es la directamente relacionada con las actividades propias de su Perfil Profesional, por ello los contenidos correspondientes a este campo están agrupados en forma tal que puedan relacionarse fácilmente con las actividades propias del técnico. Para poner en perspectiva y señalar el nivel de los contenidos, se los acompaña con ejemplos de ejercicios prácticos que contribuyan a la formación a través de desempeños que preparen al estudiante para su trabajo futuro.

Las áreas de la formación técnica específica que están relacionadas con la formación del técnico en programación son:

Aspectos formativos referidos a interpretar especificaciones en el contexto de un proyecto

Relativos a interpretar críticamente especificaciones.

El software de aplicaciones resuelve necesidades de información o automación acordadas con usuarios u otros interesados, las que son plasmadas en especificaciones de requerimientos, ya sean estas formales o informales. Programas, subsistemas y otros artefactos de software tienen que diseñarse satisfaciendo esas especificaciones y respetando buenas prácticas, así como manteniendo coherencia con la arquitectura del sistema de software en el que estarán insertos.

Estas especificaciones se refieren a las funciones que debe realizar el software, a interacciones con usuarios y otros sistemas, requisitos de calidad y comportamiento y son el punto de partida para lo que va a desarrollar. El programador de software debe ser capaz de interpretarlas, analizándolas críticamente, detectar posibles lagunas o incoherencias y validar su propia interpretación con quienes lideran el proyecto

Contenidos relacionados al análisis y especificación de software:

Requerimientos de software, el proceso, partes interesadas. Requerimientos funcionales, prioridades y criterios de realización. Requerimientos no funcionales, ejemplos y su influencia en el diseño del software. Análisis orientado a objetos y UML. Diagramas de clase. Escenarios, historias y casos de uso; diseño centrado en el usuario. Representación del comportamiento: diagramas de secuencia, máquinas de estado, diagramas de actividad. Análisis de datos: datos de referencia y de operaciones; de nivel de recursos y de volumen de actividad, diccionario de datos. Organización de datos: modelo Entidad/Relación, principales Formas Normales. Herramientas de modelización. Validación de requerimientos.

Como parte de la forma de adquirir estos aprendizajes y demostración práctica de los resultados alcanzados, en el curso de su formación los estudiantes tienen que:

Interpretar y producir diagramas de clase a partir de problemas sencillos correspondientes a diversos dominios. Analizar y discutir sus propiedades y corrección. Interpretar y especificar casos de uso básicos a partir de descripciones de situaciones realistas. Interpretar artefactos de software (clases, objetos, métodos, algoritmos, tablas).

Aspectos formativos referidos a planificar su propio trabajo en el contexto de un proyecto, identificando posibles dificultades y organizando sus actividades para encarar la solución del problema planteado

Relativos a la planificación personal dentro de un contexto de proyectos encarados por medio de procesos de ingeniería de software

El programador tiene que identificar aspectos de posible dificultad o riesgo del problema a enfrentar que requieran consulta o un cuidado mayor, evaluando a priori la magnitud del esfuerzo requerido para lograr su solución y considerar la posibilidad de subdividir la asignación en pasos o componentes menores. Esto le permite establecer informalmente un orden o secuencia de trabajo y anticipar la posibilidad de cumplir en tiempo y forma con lo requerido.

Por otra parte, tiene que desenvolverse en el marco de un equipo de trabajo organizado en función del proyecto a encarar. En consecuencia, asume responsabilidad por su asignación dentro del proyecto pero interactúa con pares y líderes del equipo para lograr un mejor proceso conjunto. Ello implica pedir y recibir ayuda para encarar su problema.

En consecuencia, además de conocer y aplicar debidamente las técnicas con que va a realizar la parte asignada, tiene que tener una comprensión del sistema y de la totalidad del proceso, tiene que comprender y cumplir estándares establecidos para el proyecto tratando de aportar lo mejor de su parte, aceptar soluciones resueltas grupalmente o por líderes o especialistas y tiene que colaborar con otros pares y juniors en la solución de los problemas.

Contenidos relacionados con el proceso de ingeniería de software

Conceptos de dinámica de grupos, grupo y equipos de trabajo, características distintivas. La tarea como eje de la convocatoria de todo grupo/equipo. Tarea explicita e implícita. Dinámica de lo grupal. La mutua representación interna, espacio y tiempo. Objetivos grupales y metas individuales. Lo individual versus lo grupal. Roles y estereotipos, rotación

de roles. La comunicación, medios, ruidos que afectan a la comunicación. Importancia de la retroalimentación. La empatía. La escucha activa. Conflictos, técnicas de resolución alternativa.

El equipo de proyectos de desarrollo de software, roles y responsabilidades de sus integrantes. Programas de trabajo y resolución conjunta de problemas. Modelos de ciclo de vida del software y de procesos de desarrollo. El problema del mantenimiento y las migraciones de plataforma.

Metodologías tradicionales y ágiles. Metodologías ágiles, concepto de *sprint*, fraccionamiento del producto en unidades realizables en un *sprint*, cola de pendientes, mejora de productos provisorios (*refactoring*), variación de los roles y la documentación en el marco de un proceso en el que se aplican metodologías ágiles.

Gestión de los cambios, conceptos de versión, *build*, producto de la asignación. Concepto de componente. Elementos de administración de la configuración y control de versiones de software. Herramientas de versionado. Otras herramientas (bibliotecas, diccionarios, repositorios) del proyecto.

Conceptos básicos de aseguramiento de la calidad y elementos de métricas. Modelos de madurez de la capacidad de desarrollo. Enfoques para la mejora del proceso. El proceso personal de software, estadísticas personales y capitalización de experiencias.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Participar de proyectos conjuntos de desarrollo de artefactos de software en los que se pongan en práctica diferentes metodologías. Poner en práctica estadísticas elementales propias del proceso personal de software. Realizar revisiones cruzadas de código proponiendo mejoras. Organizar la documentación de un proyecto. Utilizar herramientas de versionado y administración de la configuración. Reflexionar en forma conjunta sobre experiencias y conclusiones obtenidas.

Relativos a la resolución de problemas y al diseño

Los programas, subsistemas y otros artefactos de software tienen que diseñarse respetando buenas prácticas y manteniendo coherencia con la arquitectura existente o prevista del sistema de software en el que estarán insertos o tendrán que interactuar.

Lograr esto requiere no sólo conocer técnicas de diseño de software sino también comprender principios de arquitectura de sistemas de software, propiedades de calidad del software y técnicas de representación.

Contenidos relacionados al diseño de software:

Estrategias de resolución de distintos tipos de problemas. Heurísticas. Principios generales de diseño: descomposición, desacoplamiento, cohesión, reuso, reusabilidad, portabilidad, *testeabilidad*, flexibilidad, escalabilidad. Diseño estructurado. Diseño orientado a objetos. Patrones de diseño. Desarrollo de prototipos rápidos para demostración (*Rapid prototyping*).

Elementos de arquitecturas de software: concepto de vistas, arquitecturas distribuidas, *pipe-and-filter*, *model-view-controller*. Diseño orientado al reuso de componentes, incorporación de elementos disponibles al diseño. Diseño de interfases con el usuario.

Como parte de la forma de adquirir estos aprendizajes y demostración práctica de los resultados alcanzados, en el curso de su formación los estudiantes tienen que:

Resolver diversos tipos de problemas. Construir prototipos rápidos con herramientas sencillas. Diseñar soluciones a problemas dados. Diseñar tablas y bases de datos relacionales. A partir de un diseño, analizar clases de equivalencia y diseñar esquemas de prueba. Analizar críticamente la eficiencia y mantenibilidad de diseños alternativos. Relacionar situaciones dadas con patrones básicos de diseño. Analizar algunos tipos de arquitectura de sistemas de software, discutiendo sus propiedades de calidad (escalabilidad, portabilidad, seguridad, mantenibilidad). Analizar y discutir su eficiencia y escalabilidad.

Aspecto formativo referido a producir y depurar los programas, módulos, clases o subsistemas que respondan a lo requerido, aplicando patrones o reutilizando código en la medida en que resulte posible.

Esto incluye revisar código propio o ajeno para corregir defectos, optimizarlo o adaptarlo a nuevas funcionalidades requeridas. Al hacer esto se aplican buenas prácticas de programación y documentación, conforme a procedimientos de calidad establecidos. También se participa en revisiones cruzadas de artefactos de software para asegurar la calidad del producto.

Relativos a la redacción y depuración de código que responda al diseño propuesto

El programador tiene que construir programas que satisfagan efectiva y eficientemente los requisitos planteados. Estos deben enmarcarse dentro de la arquitectura prevista para el sistema, responder a buenas prácticas, siendo comprensibles y fáciles de modificar, y presentar robustez ante situaciones no previstas. Esto se logra no sólo redactando código, hay que encontrar y adaptar módulos o clases ya existentes para utilizarlas en lo que se está construyendo, verificar lo construido mediante diverso tipo de pruebas y volver a trabajar sobre lo hecho para depurar errores o malfuncionamientos encontrados, así como para optimizarlo.

Por otra parte, nuevos negocios, necesidades de usuarios o regulaciones de las autoridades plantean la necesidad de modificar aplicaciones existentes, con lo cual algún desarrollador tiene que tomar ese programa, interpretar su código para comprenderlo y ubicar dónde ese programa realiza lo que hay que cambiar. Una vez localizado el punto a modificar, tiene que plantear la forma de resolver la situación e introducir los cambios necesarios, probándolo nuevamente para verificar que haga lo esperado y que tampoco hayan cambiado funcionalidades que tenía previamente.

Esto implica programar aplicando conceptos de abstracción, descomposición, algoritmia, estructuras de datos, recursividad, herencia y polimorfismo. Por otra parte, aplicar buenas prácticas de programación y documentación, conocimientos de *testing* unitario y tener conciencia no sólo del proceso completo de desarrollo, lo que es independiente de la tecnología utilizada, sino también que los programas perduran y van a tener que ser comprendidos y mantenidos por otros.

Contenidos relacionados a algoritmos y estructuras de datos:

Elementos de programación: Estructuras de control. Algoritmos fundamentales, algoritmos numéricos simples. Variables y estructuras de datos estáticas: representación de datos numéricos, rango, precisión y errores de redondeo; arreglos; representación de datos de caracteres, listas y su procesamiento.

Estándares de nomenclatura y formato en programación, encabezado de módulos u objetos con comentarios que expliciten sus alcances y limitaciones, inserción de comentarios o advertencias en el código, documentación adicional.

Estructuras dinámicas. Manejo de memoria en tiempo de ejecución, punteros y referencias, estructuras encadenadas, pilas, colas y *hashing*. Recolección de espacios no utilizados. La elección de una estructura de datos adecuada.

Diseño orientado a objetos, encapsulamiento y ocultamiento de información, separación entre comportamiento e implementación, clases y subclases, herencia (sustitución), polimorfismo (subtipos vs. herencia), jerarquías de clases, clases colección y protocolos de iteración.

Verificación unitaria de unidades de código, concepto de cubrimiento, organización, ejecución y documentación de la prueba.

Recursión, concepto, funciones matemáticas recursivas, funciones recursivas simples, estrategia de dividir y conquistar, *backtracking* recursivo. Concepto de autómata. Elementos de complejidad de algoritmos.

Declaraciones y tipos, la concepción de tipos como conjunto de valores junto con operaciones, modelos de declaración, elementos de verificación de tipos, tipos y polimorfismo paramétrico.

Algoritmos de búsqueda sucesiva y binaria, de ordenamiento con tiempos cuadráticos (selección, inserción), con tiempos O(NlogN) (*quicksort, heapsort, mergesort*).

Estructuras dinámicas no lineales. Tablas de *hashing*, estrategias para evitar colisiones. Árboles de búsqueda binaria, operaciones básicas (búsqueda, inserción y eliminación de nodos). Representación de grafos. Algoritmos de camino mínimo.

Programación conducida por eventos, métodos para manejo de eventos, propagación de eventos, manejo de excepciones.

Integración de imágenes y sonido: estándares más comunes y sus características. Introducción a la multimedia: concepto, componentes, características, herramientas. Hipertextos, hipermedios. Animaciones: guías e interpolación de movimientos.

Máquinas virtuales, concepto, jerarquía de máquinas virtuales, lenguajes intermedios, asuntos de seguridad que surgen al ejecutar código en una máquina diferente.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Resolver ejercicios de programación, tanto con lápiz y papel como en computador, haciendo hincapié en formalizar el problema y ensayar el enfoque de su solución antes de proceder a la escritura de código, así como en verificar la corrección de la solución obtenida.

Se espera que al concluir el ciclo los estudiantes dominen al menos dos de los tres paradigmas de programación (objetos, imperativa-estructurada o funcional) y varios lenguajes (por lo menos uno correspondiente a cada paradigma, pero también otros, en particular los que tienen aplicación en páginas web). (Se entiende que el tener que adaptarse a diversos tipos de lenguajes de programación y resolver diversa clase de problemas utilizándolos ayuda al proceso de desarrollar capacidad de abstracción.)

Revisar y corregir programas dados. Resolver diversos tipos de problemas comenzando por especificar su propia comprensión de la asignación, diseñar una solución, programar o integrar partes de código utilizando el ambiente de programación indicado, documentándola de acuerdo a buenas prácticas y realizar la verificación unitaria de lo realizado. Intercambiar artefactos de software asumiendo la obligación de interpretar y criticar o mejorar lo recibido. Desarrollar proyectos grupales durante los cuales se simulen condiciones similares a las del trabajo profesional y en los que cada uno aporte componentes que deben integrarse en el producto final.

Relativos a desarrollar software que utilice bases de datos

El código de los programas se utiliza para computar datos, los que pueden ser internos del programa o, más generalmente, encontrarse o tener que ser almacenados en archivos y bases de datos. En consecuencia, el desarrollador no sólo tiene que conocer de algoritmos y lenguajes, sino también de manejo de la información.

Esto implica conocer de modelos de información que faciliten su almacenamiento y recupero, modelos de datos, indexación, lenguajes de consulta y características de los principales modelos y sistemas de bases de datos.

Actualmente, con sistemas de información distribuidos hace falta obtener o intercambiar datos con otros sistemas a través de Internet y, eventualmente, hacer uso o interactuar con herramientas externas de búsqueda.

Contenidos relacionados con bases de datos:

Modelización de datos, modelos conceptuales (*E/R*, *UML*), modelo orientado a objetos, modelo relacional, modelos semiestructurados (*XML*). Concepto y evolución de los sistemas de bases de datos, sus componentes, funciones de un sistema de base de datos.

Lenguajes de consulta (*SQL*, *QBE*), definición de datos, álgebra relacional, formulación de consultas, lenguaje de actualización, restricciones, integridad. *SQL* embebido en un lenguaje imperativo. *Scripts*. Introducción a un lenguaje de consulta de objetos. Procedimientos almacenados.

Diseño de bases de datos, dependencia funcional, formas normales, descomposición de un esquema, claves primarias y secundarias. Procesamiento de transacciones, fallas y recuperación, control de concurrencia. Bases de datos distribuidas, problemas que surgen con su explotación. Problemas de escalabilidad, eficiencia y efectividad. Privacidad, integridad, seguridad y preservación de la información.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Resolver ejercicios de álgebra relacional. Se espera que al concluir el ciclo los estudiantes resulten capaces de explotar una base de datos relacional. Revisar y corregir programas dados. Resolver diversos tipos de problemas comenzando por especificar consultas a bases de datos dadas, programar actualizaciones de datos en base a cálculos con nuevos datos, preocupándose tanto por la integridad de la información como por la eficiencia de los procesos. Diseñar tablas y bases de datos, incorporar procedimientos. Desarrollar proyectos grupales durante los cuales se simulen condiciones similares a las del trabajo profesional y cada uno aporte componentes que deben integrarse en el producto final.

Relativos a producir interfaces adecuadas para el usuario

En los sistemas de información el usuario suele proveer datos al sistema y utilizar la información que brinda el sistema para tomar decisiones de diverso tipo. En tal sentido, la calidad de las interfaces y la interacción del usuario con el sistema resultan muy importantes, ya que interfases pobremente diseñadas pueden llevar a registrar mal los datos o a dificultar el uso del sistema. En particular, cuando se producen situaciones de excepción (datos o comandos incorrectos por parte del usuario o la solicitud de algo que el sistema no puede realizar) es conveniente planificar un diálogo adecuado para resolver la situación, incluyendo ayudas para el usuario.

En consecuencia, el desarrollador, a pesar de que inscriba su componente en un diseño más general, debe conocer distintos tipos de interfases con el usuario, principios de diseño de interfases visuales, verificaciones básicas a realizar sobre los datos de entrada y manejo de ayudas y del diálogo para superar las dificultades que pueda encontrar el usuario.

En la actualidad se han difundido una serie de dispositivos (móviles, *GPS*s, tabletas de diversas características, pantallas que reaccionan al contacto, recolectores de datos) que amplían el espectro de las interfases con los usuarios, lo que genera una gama de tecnologías y modelos de interacción que un buen desarrollador de software debe conocer para su trabajo o, al menos, estar en condiciones de adaptarse rápidamente.

Contenidos relacionados con interacción ser humano-máquina:

Interacción ser humano-máquina, conceptos básicos. Distintos contextos para interfaces: visuales o de texto en aplicaciones habituales, interfaces web con dispositivos para navegación, sistemas colaborativos, juegos y otras aplicaciones multimediales, interfaces con o por medio de diversos dispositivos, lo que pueden incluir teléfonos móviles y TV digital.

Proceso de desarrollo centrado en el usuario: foco temprano en los usuarios, prueba empírica de la calidad, diseño iterativo. Medidas de evaluación: utilidad, eficiencia, facilidad de aprendizaje, satisfacción del usuario. Modelos de diseño de la interacción: atención, movimiento, cognición, percepción y reconocimiento.

Diseño para el cambio: adaptación a otras lenguas o localismos, adaptación a la diversidad de condiciones humanas. Notación para especificar interfaces. El manejo de los errores del usuario o del sistema. Técnicas y herramientas de prototipado.

Principios de interfaces gráficas, *acción-objeto vs. objeto-acción*, eventos en interfaces de usuario, estándares, errores más comunes. Interfaces para un sistema nativo, uso del *browser* para sistemas que operen en la web.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Considerar, discutir y diseñar interacciones software-usuario. Diseñar diversas pantallas que respondan a determinadas propuestas y evaluar conjuntamente lo obtenido. Diseñar interfaces para la web con ayudas para la navegación. Diseñar interfaces para alguna norma estándar (*USB*, *bluetooth*) para dispositivos.

Relativos a desarrollar software que opere en ambientes distribuidos

En la actualidad la mayor parte de los sistemas operan en forma distribuida a través de redes locales o de Internet, utilizando en muchos casos como interfase el software de navegación (*browser*) de la máquina cliente. Esto implica mantener un diálogo cliente-servidor que intercambie datos y permita acceder y actualizar bases de datos situadas a distancia.

El desarrollador tiene que conocer y poner en práctica la programación en un ambiente cliente-servidor, para lo cual tiene que comprender conceptos de arquitectura de sistemas web, aspectos de seguridad y de comportamiento.

Contenidos relacionados con computación centrada en redes:

Aplicaciones en redes. Protocolos a nivel de la capa de aplicación. Interfaces web: *browsers* y *API*s. Subprotocolos *TCP* y *UDP*. El *socket* como abstracción.

Modelo cliente servidor. Primeras acciones de ambos. Creación de *sockets* y ligado de direcciones. Par cliente/servidor *TCP*. Concepto de sesión. Par cliente/servidor *UDP*. Concepto de paquete. *Polling* con primitivas bloqueantes. *RPC*. Protocolos web. Lenguajes de programación utilizados para el desarrollo de páginas y sistemas web.

Principios de ingeniería web. Sitios web estructurados mediante bases de datos. Tecnologías de búsqueda en web. El papel del middleware, herramientas de apoyo.

Aplicaciones cooperativas. Sistemas de *workflow*. Herramientas para desarrollo en ambientes web. *Frameworks* de aplicaciones y su utilización.

Creación y administración de sitios web.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Diseñar páginas web estáticas y dinámicas. Diseñar y programar aplicaciones sencillas que interactúen en un ambiente cliente-servidor. Diseñar sitios web organizados como bases de datos para que el usuario pueda actualizarlos sin intervención de desarrolladores. Utilizar ambientes de programación para web, programar aplicaciones interactivas que actualicen bases de datos, considerar y discutir aspectos de seguridad relativos a las mismas.

Verificar los artefactos de software construidos considerando las necesidades de cobertura de la prueba.

Para ello diseña los casos considerando el entorno de pruebas y ejecuta pruebas unitarias, así como registra los datos y resultados. De ser necesario, realiza acciones correctivas sobre el código hasta satisfacerse de que cumpla con las especificaciones recibidas.

Relativos a verificar el buen funcionamiento de los programas

En el software es tan alta la distancia entre el diseño y la construcción, que resulta totalmente improbable producir inicialmente programas sin defectos. Así es que los productos tienen que ser verificados mediante pruebas que comprueben su calidad. Para ello hay que diseñar conjuntos de datos de prueba y realizar procesos en condiciones controladas que den cuenta de diversos aspectos. En primer lugar, que el programa satisfaga los requerimientos planteados. También que tenga robustez y no acepte datos incorrectos o no realice acciones imprevistas cuando un usuario se equivoca en un comando.

Esto se inscribe en el concepto de verificación unitaria, que debe realizar el desarrollador de software para satisfacerse que ha realizado lo requerido. Sin embargo, la buena práctica implica que un grupo independiente debe integrar lo realizado por cada desarrollador y someterlo a prueba conjunta, lo que puede poner de relieve fallas originadas en la interacción. La detección de fallas motiva que el desarrollador vuelva sobre el código para encontrar los defectos y los resuelva.

Para aplicar con propiedad técnicas de *testing* un desarrollador de software tiene que conocer principios generales, los diversos tipos de *testing* que se utilizan en el proceso de desarrollo de software y ser capaz de utilizar apropiadamente ambientes y herramientas específicos de *testing* unitario.

Contenidos relacionados con testing

Distinción entre validación y verificación. Enfoques estáticos y dinámicos. Fundamentos de *testing*, testeo de caja negra y de caja blanca. Pruebas funcionales: generación de casos o datos de prueba, clases de equivalencia. Pruebas estructurales: pruebas estáticas, pruebas dinámicas, cobertura de la prueba. Otro tipo de objetivos: verificación de usabilidad, confiabilidad, seguridad. Registro de fallas e informes técnicos.

Prueba unitaria, de integración, validación y prueba del sistema. Desarrollo conducido por el testeo. *Refactorización* del código. Testeo de regresión. Verificación y validación de

artefactos que no constituyen código: documentación, archivos de ayuda, material de capacitación. Inspecciones, revisiones cruzadas, auditorías.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Procesar pruebas e identificar defectos en artefactos producidos por sí mismos o por otros. Planificar y diseñar casos y conjuntos de datos para prueba de artefactos dados, respondiendo a objetivos y requisitos de cobertura. Implementar pruebas unitarias de programas y pequeños sistemas utilizando herramientas y creando los ambientes necesarios, realizar los procesos y revisar los resultados para generar informes de fallas.

Desempeño de base

Conocer y saber utilizar con propiedad y en condiciones de seguridad recursos de hardware, software y redes para emplear los ambientes que necesite para el desarrollo y la verificación del software, mantener los repositorios de información que necesite utilizar y disponer de los productos de su trabajo en condiciones de confiabilidad.

Relativos al ambiente de desarrollo

El desarrollador no sólo tiene que tener capacidades como para resolver los problemas que presenta el diseñar y programar artefactos de software que satisfagan las asignaciones planteadas en el contexto de la arquitectura propuesta. Tiene que configurar el ambiente de programación y el de *testing* que va a utilizar en su trabajo, generar o extraer datos para producir los que necesite para probar lo que realizó. Eventualmente, tratar de interpretar fallas en función de posibles problemas de compatibilidad con otro software.

Para realizar esto el desarrollador debe conocer sobre sistemas operativos y debe ser capaz de manejarse hábilmente con diversos editores, configurar aspectos de software y hardware y explotar con habilidad recursos de programación, incluyendo entre los mismos bibliotecas de objetos y programas propias, de su organización o disponibles a través de Internet, así como plantear y resolver consultas de problemas de programación a través de foros y listas públicas o privadas.

Contenidos relacionados con sistemas operativos, editores y bibliotecas de programas

Los sistemas operativos, su papel y propósito, la historia de su desarrollo, funcionalidades típicas. Mecanismos que soportan los modelos cliente-servidor y otros dispositivos. Características y objetivos de su diseño y su influencia en la seguridad, interoperabilidad, capacidad multimedial.

Aplicaciones complementarias (navegadores, defragmentadores, antivirus, traductores de medios audiovisuales).

Estructuras de sistemas operativos (monolíticos, modulares y de *micro kernel*). Abstracciones, procesos y recursos. Organización de los dispositivos, interrupciones: métodos e implementación. Concepto de estados usuario/supervisor y protección, transición al modo supervisor.

Estados y transiciones; cola de procesos, bloque de control de procesos. Despacho, *switching* de contexto, *switching* cooperativo y *preempted*. Ejecución concurrente: ventajas y desventajas. El problema de la exclusión mutua y algunas soluciones. Bloqueos: causas, condiciones, prevención. Paso de mensajes sincrónico y asincrónico. Problema de consumidor-productor y sincronización (*mutex*, semáforos). Multiprocesamiento (interrupción de ciclos, reentrada).

Políticas de despacho de procesos; programación con y sin prioridades de interrupción. Procesos y *threads*. Elementos de tiempo real y tiempos límite.

Administración de memoria. Revisión de memoria física y hardware de administración de memoria. Paginamiento y memoria virtual. *Working sets* y *trashing*. *Cacheo*.

Administración de dispositivos, características de dispositivos seriales y paralelos. Abstracción de diferencias entre dispositivos. Estrategias de *buffering*. Acceso directo a memoria. Recuperación de fallas.

Representación de caracteres, audio e imágenes. Compresión de datos, códigos para detectar o corregir errores.

Seguridad y protección. Políticas y mecanismos de separación. Métodos y dispositivos de seguridad. Protección, control de acceso y autenticación. Backups.

Sistemas de archivo (datos, metadatos, operaciones, organización, *buffering*, secuenciales y no secuenciales). Índices: contenido y estructura. Técnicas estándares de implementación. Archivos de mapeo de memoria. Sistemas de archivo para propósitos especiales. Denominación, búsqueda, acceso, backups.

Scripting. Comandos básicos del sistema, creación de *scripts*, pasaje de parámetros. Ejecución de un *script*.

Ambientes gráficos para edición, editores inteligentes. Herramientas integradas disponibles para la edición en distintos lenguajes y ambientes. Bibliotecas de clases, programas y rutinas.

Aspectos de administración de redes, uso de contraseñas y mecanismos de control de acceso, servidores de nombres de dominios y de servicios, proveedores de servicios en Internet. Aspectos de seguridad y *firewalls*. Asuntos de calidad de servicio: comportamiento, recuperación de fallos.

Como parte de la forma de adquirir estos aprendizajes y demostración práctica de los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Localizar y seleccionar artefactos de software, libre o bajo otras licencias, que respondan a ciertas características. Instalar, configurar y personalizar sistemas operativos, compiladores de lenguajes, editores y otros ambientes de programación o de prueba de programas. Crear y organizar repositorios de documentación y programas para uso personal o de proyectos.

3.4 Prácticas profesionalizantes

El mundo del trabajo, las relaciones que se generan dentro de él, sus formas de organización y funcionamiento y la interacción de las actividades productivas en contextos socio económicos locales y regionales, conjugan un conjunto de relaciones tanto socio culturales como económico productivas que sólo puede ser aprehendido a través de una participación activa de los estudiantes en distintas actividades de un proceso de producción de bienes o servicios.

La adquisición de capacidades para desempeñarse en situaciones sociolaborales concretas sólo es posible si se generan en los procesos educativos actividades formativas de acción y reflexión sobre situaciones reales de trabajo.

En este sentido, el campo de formación de la práctica profesionalizante está destinado a posibilitar la integración y contrastación de los saberes construidos en la formación de los otros campos, y garantizar la articulación teoría-práctica en los procesos formativos a través del acercamiento de los estudiantes a situaciones reales de trabajo, propiciando una aproximación progresiva al campo ocupacional hacia el cual se orienta la formación y poniendo a los estudiantes en contacto con diferentes situaciones y problemáticas que permitan tanto la identificación del objeto de la práctica profesional como la del conjunto de procesos técnicos, tecnológicos, científicos, culturales, sociales y jurídicos que se involucran en la diversidad de situaciones socioculturales y productivas que se relacionan con un posible desempeño profesional.

Un espacio de práctica profesionalizante tiene que permitir la integración de un conjunto significativo de funciones primordiales del perfil profesional en el marco de un ambiente de trabajo real o simulado. En ese sentido, las actividades formativas grupales e individuales tienen que integrar prácticas como la interpretación crítica de especificaciones de artefactos de software, el diseño de la solución, su justificación y validación; la construcción de partes no triviales, revisión, verificación unitaria y depuración, aplicando buenas prácticas de programación y documentación; así como también su integración con otros artefactos ya existentes o desarrollados por otros para conformar versiones, incluyendo la depuración de los errores encontrados. Esto requiere un conocimiento y apropiación del campo profesional y la interacción con sus distintos actores.

Esto se puede lograr en el sector productivo, realizando acuerdos en los que se planifique y verifique que el estudiante realice un conjunto de tareas del tipo de las descriptas, o en la institución educativa, creando ámbitos de desarrollo de software, típicamente denominados *software factory*, que reproduzcan las condiciones en las que desarrollan proyectos las empresas del sector, organizando equipos de desarrollo y contando con figuras docentes que asuman papeles como gerentes de desarrollo o responsables por la calidad. También resulta

importante contar con un cliente creíble que plantee demandas realistas y que se preste al juego de modificar algunos de los requerimientos durante el proceso.

Esta actividad formativa debe ser cumplida por todos los estudiantes, con supervisión docente, y la institución educativa debe garantizarla durante y a lo largo de la trayectoria formativa.

3.5. Carga horaria mínima

La carga horaria mínima total es de 6480[2] horas reloj. Al menos la tercera parte de dicha carga horaria es de práctica de distinta índole.

La distribución de carga horaria mínima total de la trayectoria por campo formativo, según lo establecido en el acápite 3.2.3 de los Lineamientos para la organización institucional y curricular de la Educación Técnica Profesional de la Educación Secundaria y Superior aprobado por Res. CFE Nro 47/08, es:

- Formación científico – tecnológica: 1700 horas reloj,
- Formación técnica específica: 2000 horas reloj,
- Prácticas profesionalizantes: equivalente al 10% del total de horas previstas para la formación técnica específica, no inferior a 200 horas reloj.

A los efectos de la homologación, la carga horaria indicada de *formación técnica específica* incluye la carga horaria de la *formación técnica* del primer ciclo. Asimismo las cargas horarias explicitadas remiten a la totalidad de contenidos de los campos formativos aunque en este marco sólo se indican los contenidos de los campos de formación científico tecnológico y técnico específico que no pueden estar ausentes de la formación de este técnico en cuestión.

[2] Esta carga horaria se desprende de considerar lo establecido en el acápite 3.2.3 de los Lineamientos para la organización institucional y curricular de la Educación Técnico Profesional de la Educación Secundaria y Superior aprobado por la Res. CFE Nro 47/08. Además, cabe señalar que, de acuerdo al primero de los párrafos, la educación secundaria obligatoria se extiende a 14 años para quienes prosigan estudios técnicos, por lo que la tecnicatura puede extenderse a 6 o 7 años, según la educación primaria implique 7 o 6 años en la jurisdicción educativa correspondiente.