Universidad de las Ciencias Informáticas
Facultad 3

Desarrollo del sistema de planificación de tareas de superación Xendero

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor:

Álvaro Quiles López

Tutor:

Henry Raúl González Brito

Ciudad de la Habana, Julio del 2012.

Declaración de autoría

Declaramos ser autores de la presente tesis y autorizamos a la facultad 3 de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los __ días del mes de _____ del año _____.

Álvaro Quiles López	**Henry Raúl González Brito**
_____	_____
Firma del autor	Firma del Tutor

Pensamiento

“Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar.”

Kent Beck.

Agradecimientos

A mi madre por ser mi mejor profesora, mi mejor amiga, mi apoyo más necesitado y la responsable de los logros más importantes en mi vida.

A mis hermanos y mi padre por estar ahí cuando los necesité.

A mis amigos que desde primer año supieron darme su amistad y hacer sentir la escuela como mi segunda casa, Driggs, la pelusita, Leydi, tito, luiso, Sandy, Sisley, Ariadna, el vice, el mero, grabiel, liban, la vieja, mi pulguita dana, anki, y los que no menciono están igual en mi corazón.

A todos los profes que aportaron su granito de arena para hacer de mí el profesional que soy hoy en día.

Dedicatoria

Dedico esta tesis a mi madre, mi padre, mis hermanos y a los que vieron en mí más de lo que yo mismo fui capaz de ver.

Resumen

Lograr que el proceso de superación postgraduada cuente con el máximo de calidad en el centro de informatización para la gestión de entidades (CEIGE) es tarea de marcado interés. Por este motivo se hace necesario la planificación personalizada de las tareas de superación, asignando labores a los profesionales en función de su categoría docente o los programas de formación postgraduada a los que esté asociado el mismo.

El presente trabajo de diploma, se centra en brindar un sistema que facilite el proceso de planificación de tareas de superación de los profesionales, por lo que se procedió al diseño del software utilizando patrones, contribuyendo a una mejor implementación, haciendo uso de diferentes herramientas, lenguaje y tecnologías.

Todo este proceso, concluyó en el cumplimiento exitoso de las metas trazadas, pudiendo verificar mediante pruebas funcionales la calidad del sistema resultante y contando por ende con la satisfacción del cliente.

Palabras claves: Superación postgraduada, patrones de diseño, motor de reglas.

Índice

Introducción

Incrementar la proyección social de los egresados de las universidades cubanas a través de la actividad de postgrado, constituye una de las principales premisas de la educación superior como parte de sus acciones estratégicas camino a la formación de profesionales de excelencia, que respondan a las demandas actuales de desarrollo social.

La enseñanza superior cubana, desarrolla más de 500 programas de maestría para distintas ramas de la ciencia, entre las que destacan las ciencias agropecuarias, biomédicas, pedagógicas y técnicas. Más de medio millón de profesionales participan en actividades de postgrado anualmente en Cuba, donde además se gradúan alrededor de 600 doctores en igual periodo. (2)

Resulta significativo subrayar la participación internacional de numerosos aspirantes a maestrías y doctorados en las diferentes áreas del postgrado cubano. Donde han sido formados hasta el año 2011 más de 1 500 másteres extranjeros de 71 países y alrededor de 840 doctores de 62 naciones (fundamentalmente de México, Colombia, Brasil y Venezuela). (2)

Para los diferentes programas de postgrado en Cuba, existen varias modalidades o formas organizativas como cursos, entrenamientos, diplomados y programas académicos como maestrías y especialidades. Otras formas de superación son la auto-preparación, la conferencia especializada, el seminario, el taller, el debate científico y otras que complementan y posibilitan el estudio y la divulgación de los avances del conocimiento, la ciencia, la tecnología y el arte. Los programas correspondientes a la superación profesional son proyectados y ejecutados por centros de educación superior. (1)

Uno de estos centros de altos estudios que posee un programa amplio de postgrado es la Universidad de las Ciencias Informáticas (UCI), la cual a pesar de ser la más joven del país ha trazado diferentes líneas de trabajo con respecto a la priorización de la superación de sus profesionales. Es por ello que la actividad de postgrado se ha ido descentralizando y para cada una de las áreas relacionadas con el entorno académico-profesional de la universidad, resulta imprescindible tener el control de las tareas orientadas a la gestión de la categoría docente de los profesores y a los diversos programas de superación. Entre dichas áreas se encuentra la Dirección de Postgrado del Centro de Informatización para la Gestión de Entidades (CEIGE), en la cual a pesar de darle gran importancia a las actividades de superación, existen dificultades e ineficiencias en el manejo, seguimiento y control de la información y las tareas asociadas a

los profesionales ya que todo el proceso se realiza empleando documentos MS Excel. Esta condición de trabajo provoca que las planificaciones sean engorrosas y que la proyección del plan de tareas de los profesionales se vea entorpecida por la gestión manual que requiere la consulta de varias páginas tabuladas para generar dicha información en un momento determinado.

Derivado de la situación problémica expuesta previamente, se plantea que el **problema a resolver** radica en las deficiencias del manejo de la información para la toma de decisiones referente a la superación de los profesionales, provocando retrasos, pérdida de información e inconsistencia en el proceso de planificación de las actividades de superación de los profesionales del CEIGE.

En función de resolver el problema anterior se definen como **objeto de estudio** los procesos de planificación de tareas para la formación postgraduada, centrándose en el **campo de acción** de los procesos de planificación de tareas para la formación postgraduada en el CEIGE. Además se propone como **objetivo general** desarrollar un sistema de planificación de tareas de superación basado en reglas, que posibilite informatizar el proceso de planificación de las tareas de superación de los profesionales del CEIGE y para su desglose se tienen los siguientes **objetivos específicos**:

1. Realizar un estudio del estado del arte sobre los sistemas de planificación de tareas de la formación postgraduada.
2. Diseñar el sistema de planificación de tareas de superación.
3. Implementar el sistema de planificación de tareas de superación.
4. Validar el sistema de planificación de tareas de superación.

Para desarrollar los objetivos específicos se planifican las siguientes **tareas de la investigación**:

1. Caracterización de determinados sistemas, tanto nacionales como internacionales, para la conformación del estado del arte sobre los sistemas de planificación de tareas de la formación postgraduada.
2. Selección del motor de reglas a utilizar para el manejo de las reglas de negocio del sistema.
3. Definición del tipo de archivo y estructura para importar los datos al sistema.
4. Descripción de la metodología a definir para el sistema.
5. Identificación de los requisitos funcionales y no funcionales del proceso de planificación de tareas de superación.
6. Realización del diseño de la solución.

7. Implementación del sistema de planificación de tareas de superación.
8. Validación de la solución planteada.

Para una mejor representación y distribución de la información inherente a esta investigación, se propone la siguiente estructura del documento:

Capítulo 1: Fundamentación teórica

Se reflejará en este capítulo el estado del arte referente a la investigación. De igual manera se exponen las tecnologías que se utilizarán y su justificación de uso, así como la metodología de desarrollo a emplear.

Capítulo 2: Modelación del sistema

En este capítulo se traza como objetivo principal el proceso de construcción de la solución, en aras de obtener como resultado los artefactos que propone la metodología de desarrollo que se identifique y seleccione para esta investigación.

Capítulo 3: Implementación y Validación de la solución propuesta

Se abarcará en este capítulo todo lo concerniente a la implementación del sistema. Se realizan además pruebas a la aplicación, en pos de validar y garantizar su calidad.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se exponen los principales conceptos afines al problema en cuestión que aborda el presente trabajo de diploma. Se brinda información sobre los procesos de categorización docente y los cursos de la formación postgraduada, detallando cada uno de los mismos. Por otra parte se ofrecen detalles sobre la metodología y las tecnologías que darán soporte a la investigación.

1.2 Marco conceptual

Cuando una nueva universidad es creada, debe pensarse en la conformación del claustro de profesores y de los profesionales en general que laborarán en la misma. Tal es el caso de la Universidad de las Ciencias Informáticas (UCI), que en su génesis tuvo que asistirse de profesores provenientes de otras instituciones y universidades para poder comenzar la actividad docente y de producción de software. Todos esos profesionales contaban con categorías docentes principales como (Profesor Titular, Profesor Auxiliar, Asistente e Instructor).

Con el paso del tiempo y el incremento del estudiantado a razón de 2000 estudiantes por año aproximadamente, se hizo necesario aumentar el universo profesoral con profesionales recién graduados y cuyos perfiles estuviesen relacionados con la carrera que se imparte en la Universidad. Por ello, su categorización docente y superación profesional se convertirían en los procesos de mayor prioridad en la universidad para garantizar la calidad de los resultados en los entornos académico y productivo. A continuación se detallan los conceptos más importantes de ambos procesos, utilizando como base el Reglamento de Postgrado del Ministerio de Educación Superior y la Resolución 128 del 2006 para la Categorización Docente.

1.2.1 Categorización docente

El proceso de categorización docente, da comienzo a la conformación y actualización del expediente del docente aspirante, labor en la que interviene el departamento de cuadros de la dirección de desarrollo del capital humano de la universidad. El aspirante, es el responsable de que en su expediente esté registrada y documentada la trayectoria de su desempeño laboral, académico y científico.

El proceso de categorización se divide en dos grupos principales: las categorías docentes principales y las categorías docentes especiales.

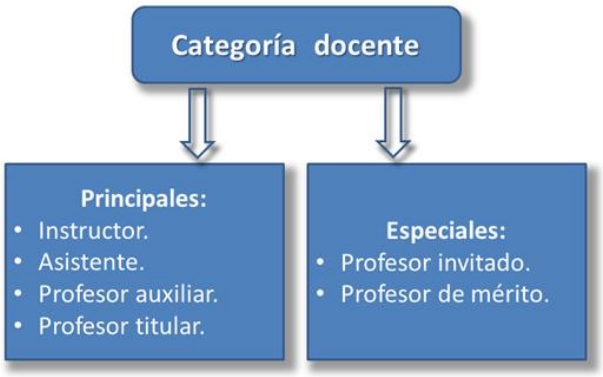


Figura 1: Categorías docentes.

1.2.2 Educación de postgrado

La educación de postgrado es una de las líneas principales en las que se centra el trabajo de la educación superior en Cuba, representando el nivel más alto del sistema de educación superior, enmarcado en la formación permanente de los graduados universitarios, concurriendo no solo los procesos de enseñanza y aprendizaje, sino también otros como investigación, innovación, creación artística, etc. La importancia de la educación de postgrado se evidencia en la necesidad de la superación a lo largo de la vida profesional, también en la gestión del aprendizaje y su socialización en la construcción del conocimiento, es la fuerza social transformadora que promueve el desarrollo sostenible de la sociedad.

Las formas organizativas principales de la superación profesional en la educación de postgrado son el curso, el entrenamiento y el diplomado, de estos 3, solo el diplomado en conjunto con los programas académicos son nombrados programas de postgrado.

Los programas de postgrado son sustentados por un sistema acumulativo de créditos académicos que facilita la flexibilidad organizativa de los planes de estudio, la transferencia de docentes, y la homologación de estudios realizados entre diversas instituciones.

El crédito académico es una unidad de expresión cuantitativa y cualitativa que valora los resultados alcanzados teniendo en cuenta la profundidad, el volumen y la intensidad del trabajo que realiza el estudiante para lograr las metas trazadas en los programas. Los mismos se otorgan al considerar

cumplidos los objetivos de las actividades planificadas. Los créditos pueden ser obligatorios, opcionales y libres, de acuerdo con los objetivos y la estrategia de formación. Tanto los créditos opcionales, como los libres, se obtendrán bajo la orientación del comité académico.

- Obligatorios: son aquellos que se exigen a todos los estudiantes, sin distinción.
- Opcionales: son seleccionados por los alumnos dentro del propio programa matriculado, a partir de un conjunto de ofertas.
- Libres: son los que se obtienen en cursos y entrenamientos de posgrado fuera del programa en el que está matriculado el profesional.

Los programas de postgrado, por otra parte, tienen asociados un grupo de cursos, que no son más que la organización de un conjunto de contenidos que abordan resultados de investigación relevantes o asuntos trascendentes con el propósito de complementar o actualizar los conocimientos de los profesionales que los reciben.

Otra forma organizativa de superación es el entrenamiento, el cual posibilita la formación básica y especializada de los graduados universitarios, particularmente en la adquisición de habilidades y destrezas en la asimilación e introducción de nuevos procedimientos y tecnologías.

Por último se encuentra el diplomado como otra de las formas organizativas; el mismo tiene como objetivo la especialización en un área particular del desempeño y propicia la adquisición de conocimientos y habilidades académicas, científicas y/o profesionales en cualquier etapa del desarrollo de un graduado universitario, de acuerdo con las necesidades de su formación profesional o cultural.

Apoyándose en el conocimiento de los principales conceptos anteriormente abarcados, pertenecientes a los procesos de categorización docente y educación de postgrado, se realiza la planificación de las tareas de superación de los profesionales.

1.2.3 Planificación

La planificación, de manera general, es el proceso de toma de decisiones para lograr un objetivo deseado, teniendo en cuenta la situación actual y factores del medio ambiente que pueden influir en el logro del resultado esperado. Una de las variantes de la planificación, es la planificación estratégica, la misma consiste en un patrón de decisiones coherente, unificador e integrado, que determina y revela el propósito

de un centro u organización en términos de objetivos a largo plazo, programas de acción y prioridades para la asignación de recursos. Para llevar a cabo la planificación estratégica se tienen en cuenta las siguientes fases. (10)

- Visión.
- Análisis del entorno y escenarios futuros.
- Misión.
- Análisis interno de la organización.
- Definición de objetivos y metas.
- Identificación e implementación de factores estratégicos.

Aplicando estas fases a los procesos de superación postgraduada, la visión sería la apreciación idealizada de lo que se desea lograr, en otras palabras la obtención de una planificación exitosa, que permita un mejor control de las tareas de superación postgraduada.

La segunda fase, es el análisis del entorno y escenarios futuros. Se debe tener en cuenta identificar y prever los cambios que se produzcan en la realidad actual o el comportamiento futuro, esos cambios pueden verse reflejados en la variación de la cantidad de años para el cambio de categoría docente, variación que influiría en la futura planificación de las tareas.

Otra fase importante, es la de definición de objetivos y metas, los objetivos son los resultados a largo plazo y las metas los resultados a corto plazo. Los objetivos se establecen en términos generales y amplios, por ejemplo, un objetivo sería lograr la planificación de las tareas pertenecientes a los programas docentes de postgrado y una meta sería el desglose de los cursos aún no concluidos por los profesionales que cursan los programas de superación. Por último, la fase de implementación de factores estratégicos se centra en construir planes y asignar recursos. O sea, hacer operativos los objetivos y metas obtenidos del análisis de la situación. Todos estos conceptos referentes a la categorización docente, la educación de postgrado y la planificación, de una manera u otra, ayudan a una mejor comprensión de los procesos del negocio y contribuyen a formar bases para sustentar la investigación de los sistemas que puedan brindar una posible solución al problema que se plantea en el presente trabajo de diploma.

1.3 Tendencias y paradigmas actuales

La categorización docente de profesionales y la educación postgraduada son procesos que complementan la gestión universitaria, representando una pequeña parte de los procesos que conforman el medio ambiente en las universidades. Los sistemas de gestión universitarios abarcan los más disimiles procesos y están representados desde servicios universitarios como la gestión de matrículas, cursos, consultas de plan de estudio y cronograma de evaluaciones hasta los servicios de índole financiero; así como desde el análisis de la situación corporativa de la universidad o la gestión de los recursos humanos y económicos.

1.3.1 Sistemas a nivel internacional

Desde el ámbito internacional, los sistemas de gestión universitaria son ampliamente empleados por diferentes instituciones de gran prestigio, un ejemplo de ello es el **UNIVERSITAS XXI**. Esta solución integral de gestión fue creada por el proyecto español de las universidades públicas de Alcalá, Carlos III de Madrid, Castilla-La Mancha, Rey Juan Carlos, Salamanca, Valladolid y del Grupo Santander, el cual es conocido como la Oficina de Cooperación Universitaria (OCU). Dicha solución es un Planificación de Recursos Empresariales (ERP) para las universidades, que consta de módulos que cubren algunos procesos de la gestión universitaria tales como la gestión académica, los recursos humanos, el ámbito económico, las investigaciones, la inteligencia institucional y la comunicación entre los usuarios del sistema mediante SMS, E-Mail y Twitter. De estos procesos que informatiza de alguna manera, solo las investigaciones y la gestión académica presentan algún tipo de relación con la superación postgraduada. El módulo de investigación aborda el tratamiento de la información relativa a la actualidad investigadora y la producción científico-técnica del personal investigador, además automatiza el control de proyectos, tramitación de patentes, gestión de contratos y gestión de grupos de investigación. Por otra parte el módulo académico cubre la gestión de becas nacionales e internacionales, organización de prácticas en empresas, definición de planes de estudio y planificación de la docencia. La caracterización de estos módulos, demuestra de manera concreta la ausencia de alguna forma de planificación de tareas de superación para la formación postgraduada.

Otro de los sistemas de gestión universitaria investigado fue **CLASS**. El cual es un software creado por la empresa INNOVASOFT.SA que lo comercializa. Esta empresa tiene su sede en la casa matriz de San José, Costa Rica y se especializa en la gestión académica y contable. Por su parte CLASS define como principal propósito: optimizar los diferentes procesos entre el área de servicios universitarios y el área financiera. Con una integración entre sus modulaciones, pretende una autoadministración con poca

intervención de usuarios, basada en herramientas de plataforma de auto-gestión para estudiantes, profesores y catedráticos por medio de Internet.
Entre los beneficios más importantes que presenta este sistema se encuentran los siguientes:

1. Aplicaciones universitarias-financieras integradas bajo un solo proveedor de software.
2. Centralización de información de multicompañías y sedes en un único servidor de datos para análisis y toma de decisiones.
3. Analizador de la situación corporativa de la universidad. Plataforma de pagos (conexión en línea con la red local bancaria y pago con tarjeta).
4. Descongestionamiento de las funciones del departamento de registro y matrícula, fomentando el autoservicio de los usuarios (estudiantes y profesores) por medio de Internet.

El descongestionamiento de las funciones del departamento de registro y matrícula, contribuye a la informatización del proceso de matriculación en cursos, permitiendo controlar los cursos a los que están asociados tanto un estudiante en su formación como un profesional en su superación de postgrado. Sin embargo, el sistema como tal no cuenta con un módulo centrado en la planificación de tareas de superación postgraduada para profesionales, lo que dificulta su adaptabilidad a las especificidades de esta investigación.

Además de los sistemas descritos inicialmente se tiene el **Sistema de autogestión de alumnos SIU-Guaraní,** que es un software argentino que tiene como finalidad, brindar a los alumnos de la Universidad Nacional del Noreste (UNNE) y sus Unidades Académicas una herramienta que les permita acceder desde cualquier lugar a través de Internet y realizar acciones como: inscripción a exámenes y cursos, reinscripción a carrera, consulta de créditos, consulta de inscripciones, consulta de plan de estudios e historia académica, consulta de cronograma de evaluaciones parciales, notas de evaluaciones parciales, materias regulares, agenda de clases, solicitud de certificados, actualización de datos censales y recepción de mensajes. Esta posibilidad de acceder a la información y realizar algunas gestiones administrativas evitando traslados innecesarios de la persona, contribuye a agilizar los procesos administrativos en general, en un ambiente más seguro y mejorando la calidad de la información obtenida. De las funcionalidades de este sistema, solamente consulta de créditos y la historia académica, presentan elementos que tributan a la formación de postgrado, pero no son suficientemente abarcadores para cumplir con el objetivo de planificar tareas para la superación postgraduada de los profesionales.

1.3.2 Sistemas a nivel nacional

Luego de explorar el escenario internacional, resulta necesario centrarse en el ámbito nacional, específicamente en un sistema de gestión universitaria que se encuentra en desarrollo en la UCI. Dicha aplicación está siendo creada en el Centro de Informatización Universitaria (CENIA) y no es más que un ERP Universitario denominado **Gestión Universitaria UCI**. Este incluye varios subsistemas, agrupados en las siguientes áreas de proceso: Pregrado, Postgrado, Producción, Investigación, Ingreso, Ubicación Laboral, Laboratorio, Residencia, Extensión Universitaria, Cooperación Internacional, Biblioteca y Teleformación.

Específicamente el módulo de postgrado, está pensado para gestionar los procesos de formación postgraduada con el objetivo de garantizar dichas actividades para el claustro de profesores del centro y contribuir a la superación de los profesionales. Aún con la futura liberación de este módulo, el sistema de Gestión Universitaria UCI, no contaría con la opción de planificar tareas a los profesionales, en función de las metas a cumplir para terminar los programas de superación postgraduada o en función de los objetivos a superar según la categoría docente avalada.

1.4 Valoración de los sistemas estudiados

El estudio de los sistemas anteriores, contribuyó con el establecimiento de las bases del conocimiento para la realización y complementación del proceso de desarrollo de software en respuesta al problema planteado. Para ello se pretende utilizar la experiencia adquirida en función de una mejor comunicación con el cliente, permitiendo sugerir elementos que se convirtieron en nuevas funcionalidades atractivas para el mismo y que serán integradas a la solución. De igual manera, se pudo verificar que ninguno de los sistemas gestiona las funcionalidades referidas en la investigación para la planificación de tareas, en dependencia de la categoría docente de un profesional o en función de las metas a lograr para vencer los programas académicos de superación postgraduada.

1.5 Metodología y tecnologías

Terminado el análisis de los sistemas de gestión universitaria, el siguiente paso es la selección de la metodología y las tecnologías que brindarán soporte a la solución, por lo que en los segmentos desarrollados a continuación se describen las principales características inherentes al tema de este epígrafe.

1.5.1 Metodología SXP

Debido a la necesidad de un rápido desarrollo de la solución y la facilidad de contar con amplia comunicación y retroalimentación con el cliente, se decidió optar por la metodología ágil SXP. La misma procede de la investigación realizada por la Ing. Malay Rodríguez Villar en el año 2007, teniendo como resultado la propuesta nombrada MA-MRV-R1. En el 2008, la entonces estudiante Gladys Marsi Peñalver Romero, en su trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas, presenta una propuesta más refinada nombrada en este caso MA-GMPR-UR2, adjuntando en su entrega la documentación necesaria para aplicar la solución en proyectos productivos. Posteriormente en marzo del año 2009 se procede a renombrarla como SXP, nombre con el cual se conoce hoy en día. Esta metodología es el resultado general de la unión de las mejores prácticas de la metodología de desarrollo XP[1] y la metodología de gestión del trabajo SCRUM[2].

En la **Figura 2**, se puede apreciar el esquema de SXP, donde se representa gráficamente lo que propone para la gestión del proyecto. La misma emplea SCRUM para la planificación, supervisión y control no solo del proceso de software, sino también del personal y los eventos que ocurren mientras evoluciona el software, desde la fase preliminar a la implementación operacional. SCRUM no es aplicada en su totalidad, sino que son tomadas algunas características de interés. Para la ingeniería de software, SXP basa sus prácticas en la metodología ágil XP asimilando características esenciales para el diseño e implementación de software bajo el uso de esta metodología.

---

[1] La Programación Extrema (XP) es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado.

[2] SCRUM es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos. (6)

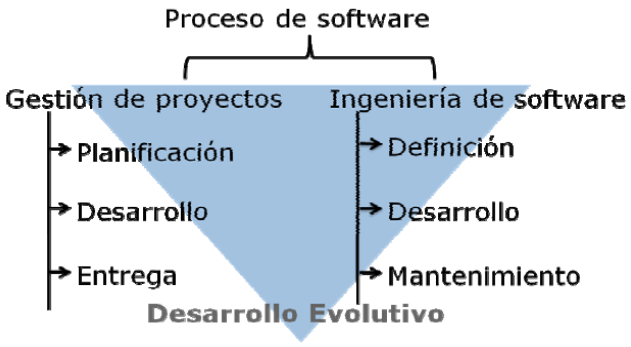


Figura 2: Esquema de la metodología SXP.

En SXP, el trabajo de la ingeniería de software se divide en tres fases genéricas, con independencia del área de aplicación, tamaño o complejidad del proyecto, las mismas son:

Fase de Definición

"Se centra en el qué". En esta fase se intenta definir qué información ha de ser procesada, qué función y rendimiento se desea, qué comportamiento del sistema, qué interfaces van a ser establecidas, qué duración se necesita para definir un sistema correcto, en fin, han de identificarse los requisitos claves del sistema y del software. Cuenta con tres tareas principales: ingeniería de sistemas o de información, planificación del proyecto y análisis de los requisitos. (5)

Fase de Desarrollo

"Se centra en el cómo". En esta fase se intenta definir cómo han de diseñarse las estructuras de datos, cómo ha de implementarse la función dentro de una arquitectura de software, cómo han de implementarse los detalles de los procedimientos, cómo han de caracterizarse las interfaces, cómo ha de traducirse el diseño en un lenguaje de programación y cómo ha de realizarse la prueba. Cuenta con tres tareas: diseño del software, generación de código y prueba del software. (5)

Fase de Mantenimiento

Se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que va evolucionando el entorno del software y a cambios debido a las mejoras producidas por los requisitos cambiantes del cliente. Tipos de cambios: Corrección, Adaptación, Mejora, Prevención. (5)

1.5.2 Tecnologías

Junto a la metodología de desarrollo, las tecnologías que se emplean para la solución del problema, son parte importante para la obtención del resultado esperado. Dichas tecnologías se describen a continuación.

Lenguaje de modelado

Para el modelado de la solución se utilizará el Lenguaje de Modelado Unificado (UML por sus siglas en inglés). El mismo está compuesto por diversos elementos gráficos que se combinan para conformar diagramas, para visualizar, especificar, construir y documentar los artefactos de un sistema. UML propone un lenguaje de modelado común para todos los desarrollos, por lo que se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo. (11)

Herramienta CASE

Como herramienta CASE[3] (Computer Aided Software Engineering o Ingeniería de Software Asistida por Computadora) se tomará al Visual Paradigm en su versión 8.0. Este soporta el ciclo de vida completo del desarrollo de software. Presenta además, soporte para diagramas UML tales como diagramas de colaboración, diagramas de paquetes, diagramas de casos de uso, diagramas de clases, entre otros. Permite generar códigos desde los diagramas creados y documentación. Brinda tutoriales de UML, así como demostraciones interactivas de UML. Soporta la generación de objetos java desde la base de datos. Transforma diagramas de entidad-relación en tablas de base de datos. Ingeniería inversa desde base de datos en Sistemas Gestores de Base de Datos (SGBD) existentes a diagramas de entidad-relación. (12)

[3] **CASE**: es un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un sistema informático. (3)

Fue escogida esta herramienta por brindar la facilidad de abarcar todas las necesidades de diseño planteadas en la metodología seleccionada.

Ambiente de desarrollo

Por otra parte, para el desarrollo de la solución se cuenta con el ambiente de desarrollo compuesto por la plataforma Java Standard Edition, el lenguaje de programación Java y el IDE (Entorno Integrado de Desarrollo) NetBeans. Este ambiente fue escogido para la presente investigación en busca de lograr que la solución resultante estuviese disponible tanto en el sistema operativo Linux como en Windows, en aras de apoyar la estrategia de migración a software libre planteada por el país, por otra parte, el equipo de desarrollo se encuentra familiarizado con el ambiente de desarrollo, permitiendo acortar la curva de aprendizaje de las tecnologías a emplear en el desarrollo del sistema. A continuación se exponen algunas de las principales características del ambiente de desarrollo.

Plataforma Java Standard Edition (Java SE)

Java SE es una plataforma abierta y estándar que proporciona técnicas específicas que describen el lenguaje de programación Java, pero además, provee las herramientas para implementar productos de software de alto rendimiento, escalabilidad y seguridad. Cuenta con implementaciones de APIs[4] para soporte de seguridad, acceso a base de datos, rutinas de dibujado 2D para visualizar los componentes de interfaz de usuario en lugar de utilizar el soporte de la Interfaz Gráfica de Usuario (GUI por sus siglas en inglés) nativa del sistema operativo, suministra el soporte para comunicación remota entre aplicaciones, etc. (13)

Lenguaje Java

De acuerdo a la plataforma que se empleará, se deduce la selección del lenguaje Java para la implementación de la solución. El mismo es un lenguaje de programación simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico. Es un lenguaje multiplataforma, que proporciona un conjunto de clases para su uso en aplicaciones de diferente índole. También gestiona el ciclo de vida de los objetos, liberando el uso en

[4] Las API no es más que una serie de servicios o funciones que se le ofrece al programador, como por ejemplo, imprimir un caracter en pantalla, escribir en un fichero, etc. (4)

memoria de los mismos, permitiéndole al programador concentrarse únicamente en la creación de los objetos. (13)

Entono Integrado de Desarrollo (IDE)

En aras de desarrollar la solución haciendo uso del lenguaje seleccionado, se tuvieron en cuenta los principales IDE de programación en Java, entre los que se encuentran el Netbeans y el Eclipse como los más utilizados a nivel internacional. Independientemente de que ambos están muy parejos en cuanto a las ventajas que brindan para el desarrollo de aplicaciones, se selecciona el NetBeans, debido a que se adapta más a las necesidades de la presente investigación, fundamentado por las características siguientes: es un IDE de código abierto, con todas las facilidades del programa disponibles de forma gratuita; para ampliar sus funcionalidades hace uso de plugins[5], aportándole gran facilidad de utilización. Permite además, integrarse con DROOLS[6] de manera sencilla para el manejo de las reglas del negocio, la librería XSTREAM[7] para el trabajo con XML y la librería POI para la gestión de los ficheros del paquete de Microsoft Office. Importante también resaltar la familiarización del equipo de desarrollo con este IDE de programación.

Gestor de reglas de negocio (BRMS por sus siglas en inglés)

Después de escogido el ambiente de desarrollo, otro paso importante es la selección del motor de reglas a utilizar para el manejo de las reglas del negocio. A continuación se muestra una tabla comparativa, resultado de la investigación realizada en cada uno de los sitios oficiales de los motores de reglas JRule Engine, Sweet Rules, OpenRules, Hammurapi Rules, Jess Rules y Drools.

Motores de reglas	Propiet.	JSR-94	Proyecto activo	Document. existente	Acept. Internac.
Drools		x	x	Buena	Excelente
JessRules	x	x		Mala	Buena
Sweet Rules			x	Regular	Buena

[5] Un **plugin** es una aplicación informática que añade funcionalidades específicas a un programa principal. (8)

[6] Drools es un motor de reglas de negocio para Java de la compañía Red Hat.

[7] XStream es una librería para el manejo de ficheros de extensión XML.

OpenRules		x	x	Regular	Buena
Hammurapi Rules		x	x	Regular	Buena

***Tabla 1**: Comparación de los motores de reglas estudiados.*

Como se puede observar, de los motores evaluados, Drools presenta mayor aceptación en la comunidad internacional con mejor documentación existente. Es un proyecto de software libre bajo la licencia Apache, cumple con el estándar JSR-94[8] para motores de inferencia. Consta de un lenguaje propio y acoplamiento con los lenguajes de programación Java y Python. Presenta una excelente integración con el IDE de programación a utilizar en el desarrollo del software. Al igual tiene una clara separación entre los datos y la lógica del dominio. Para su funcionamiento usa la implementación del algoritmo RETE, diseñado por el Dr. Charles L. Forgy[9], siendo uno de los algoritmos más utilizados entre los gestores de reglas de negocio.

Librería POI

Otro detalle de gran peso para el desarrollo de la solución, es la generación del resultado de la planificación de tareas de superación en documentos Microsoft (MS) Excel. Para ello, se utilizará la librería POI en su versión 3.0.2, creada por la organización no lucrativa Apache Software Foundation. Esta librería cuenta con APIs para el manejo de ficheros del paquete de MS Office como lo son las presentaciones de MS Power Point, documentos de MS Word, libros de cálculos de MS Excel, incluye también documentos de MS Visio y MS Publisher. (14)

Librería XSTREAM

Para la gestión de los datos se definió la librería XStream. La cual se encarga de serializar los objetos, permitiendo guardarlos en formato XML. Al generar los ficheros XML, las etiquetas toman el nombre de los atributos de las clases. También provee un conjunto de conversores para llevar tipos de objetos particulares a XML o viceversa. Algunos de los tipos de objetos con conversores son: String, File, Collections, Arrays, Date, entre otros. (15) Esta librería fue escogida para la gestión de la información de

[8] JSR-94 (Java Specification Request 94) define una API de Java de tiempo de ejecución para motores de inferencia, permitiendo llamar a un motor de inferencia desde programas Java.

[9] Profesor de la universidad de Carnegie Mellon en Pensilvania Estados Unidos, creador del algoritmo RETE, uno de los más utilizados a nivel mundial en los motores de reglas inferenciales.

la base datos debido a la necesidad de cumplir con el requisito no funcional planteado por el cliente, donde expresa que la solución no puede depender de conexiones a sistemas gestores de base de datos para el almacenamiento de la información.

1.6 Conclusiones

A manera de conclusión, en este capítulo se dio cumplimiento al primero de los objetivos específicos realizando un estudio que permitió la identificación objetiva de la situación actual en torno a la existencia de sistemas de gestión universitaria tanto nacionales como internacionales, donde se evidenció la falta de correspondencia entre esas soluciones y las especificidades del negocio inherente a esta investigación. Por lo que justifica la intención del presente trabajo de llevar a cabo el desarrollo de una solución para el problema planteado. De igual manera se investigó sobre los principales conceptos que se manejan, logrando un mejor entendimiento del negocio y la comunicación con el cliente. Importante también resaltar la selección de las tecnologías y la metodología de desarrollo a emplear, facilitando sentar las bases para el futuro desarrollo del software.

Capítulo 2: Arquitectura y diseño del Sistema de Planificación de Tareas de Superación Xendero.

2.1 Introducción

En este capítulo se expondrán los detalles concernientes a los requisitos captados, seguido de la arquitectura por la que se regirá el desarrollo del sistema y la modelación del diseño con su respectiva validación. De igual manera se detallarán los patrones de diseño empleados y las particularidades del modelo de datos a emplear para la gestión de los mismos.

2.2 Requisitos del sistema

Según Ian Sommerville[10], "Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema." Los requisitos representan las necesidades del cliente a ser satisfechas por el software resultante del proceso de desarrollo. De ahí la importancia de ser bien captadas dichas necesidades, pues independientemente de un diseño e implementación excelente, si no se centra en el cumplimiento de los requisitos planteados, entonces no tiene sentido alguno.

Para extraer los requisitos del sistema se utilizaron las siguientes técnicas:

Entrevistas: Se utiliza en los encuentros con el cliente. Se realizaron preguntas con el objetivo de obtener toda la información posible sobre la visión que el entrevistado tiene de los requisitos y comprender los propósitos de la solución buscada.

Tormenta de ideas: Se realizan reuniones, donde cada uno expresa sus ideas. Su objetivo fundamental es dar una visión general de las necesidades del sistema.

Al aplicar las técnicas descritas con anterioridad, se obtuvieron los siguientes requisitos funcionales:

- Importar datos para el sistema desde fichero de extensión csv.
- Gestionar reglas de cambio de categoría docente.

[10] Profesor de la Facultad de Ciencias de la Computación en la Universidad de St. Andrews, donde imparte cursos en técnicas avanzadas de ingeniería de software y la ingeniería de sistemas críticos.

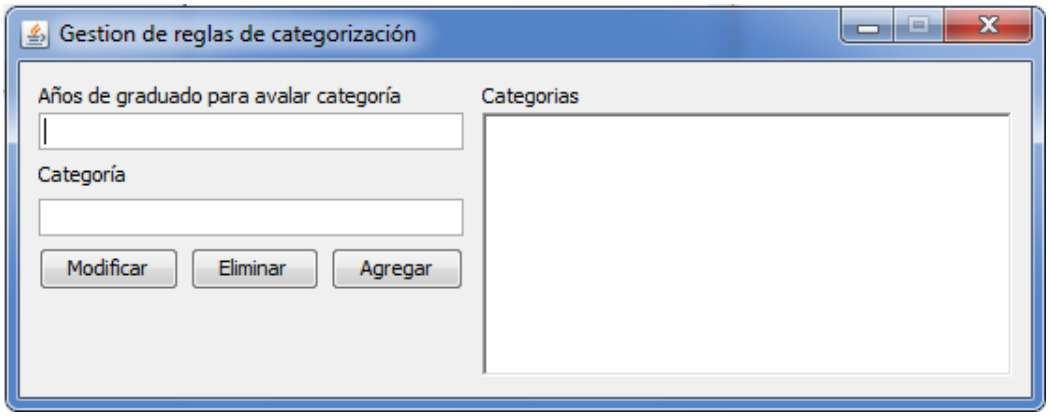
- Gestionar tareas por categoría docente.
- Generar tareas de superación para profesionales.
- Exportar proyecto.
- Importar proyecto.
- Gestionar profesionales.
- Gestionar programas.
- Gestionar cursos pasados.

Luego del proceso de captación de requisitos, el próximo paso es la especificación de los requisitos funcionales, los cuales se convertirán en las principales funcionalidades del software a desarrollar, la metodología adoptada se apoya en el artefacto (Historia de usuario) para la especificación de los mismos.

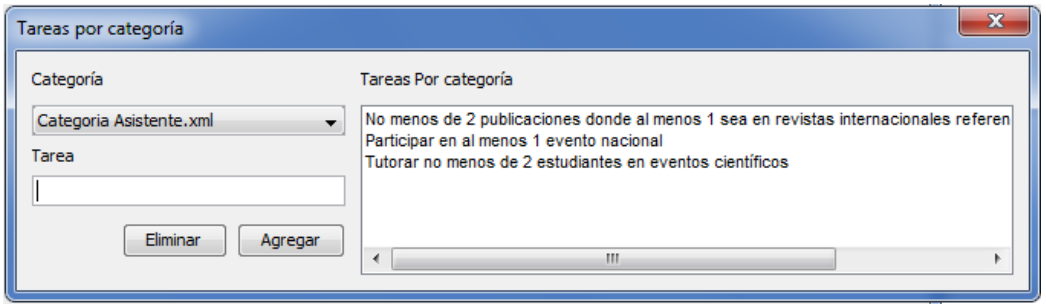
2.2.1 Requisitos funcionales

Las historias de usuarios (HU) son una técnica utilizada en la metodología ágil XP para especificar los requisitos de software, práctica incorporada en SXP. Las HU guían la construcción de las pruebas para verificar que la historia ha sido correctamente implementada. Representan las tareas o funcionalidades que el sistema debe hacer, son escritas por los clientes sin un formato específico y en lenguaje natural, no excediendo las pocas líneas de su formato. (16)

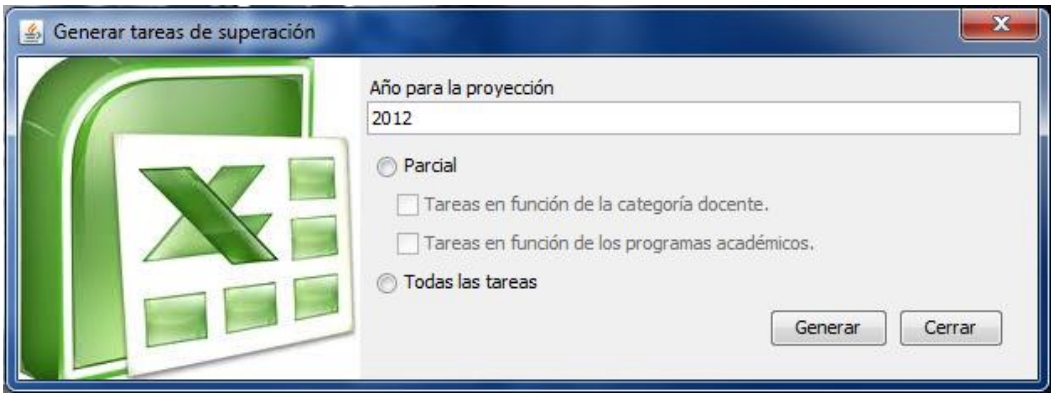
En las mismas se define el programador que realizará la implementación del requisito que representa la historia de usuario, tiempo y esfuerzo que se invertirá, acompañado de un prototipo de interfaz de usuario. Elementos con los que se cuenta para el posterior diseño del software. De igual manera son utilizadas en la estimación del tiempo de desarrollo. A continuación se exponen las historias de usuario más críticas, representando las funcionalidades en las cuales el cliente muestra mayor interés, dejando el resto de las especificaciones en el Anexo 3.

Historia de Usuario	
Código: HU_2	Nombre Historia de Usuario**: Gestionar reglas de cambio de categoría docente.**
Modificación de Historia de Usuario Número: Ninguna.	
Referencia: Se tiene como objetivo que el usuario gestione las reglas para asignar la tarea que define el cambio de categoría de los profesionales.	
Programador: Álvaro Quiles López.	**Iteración Asignada:** 1
Prioridad: Alta.	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto.	**Puntos Reales:** 1
Descripción: Esta opción debe permitir la modificación de las reglas de negocio que definen la asignación de la tarea para indicar el cambio de categoría a un profesional.	
Observaciones:	
Prototipo de interfaz: 	

Historia de Usuario	
Código: HU_3	Nombre Historia de Usuario**: Gestionar tareas por categoría docente.**
Modificación de Historia de Usuario Número: Ninguna.	
Referencia: Se tiene como objetivo que el usuario gestione las tareas que se asignarán a los profesionales según la categoría docente.	
Programador: Álvaro Quiles López.	**Iteración Asignada:** 1

Prioridad: Alta.	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto.	**Puntos Reales:** 1
Descripción: El usuario procede a generar las tareas a asignar según la categoría, las mismas serán por las que se regirá el motor de reglas para asignar las tareas a los profesionales en función de su categoría docente.	
Observaciones:	
Prototipo de interfaz: 	

	Historia de Usuario
Código: HU_4	Nombre Historia de Usuario**: Generar tareas de superación para profesionales.**
Modificación de Historia de Usuario Número: Ninguna.	
Referencia: Se tiene como objetivo que el usuario de inicio a la generación de tareas de superación postgraduada para cada uno de los profesionales.	
Programador: Álvaro Quiles López.	**Iteración Asignada:** 2
Prioridad: Alta.	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto.	**Puntos Reales:** 1
Descripción: El usuario procede a dar inicio a la generación de las tareas que el motor de reglas asignará a los profesionales en dependencia de los programas a los que este asociado y los cursos concluidos. También dependiendo de la categoría docente avalada. La salida consistirá en un fichero de MS Excel que contendrá las tareas que se generaran a partir de la selección de los cursos que no ha cursado aún el profesional, cursos pertenecientes a los programas a los cuales esté asociado el profesional. Otras tareas serían las que se asignen por la categoría docente que tenga avalada el	

profesional.
Observaciones:
Prototipo de interfaz: 

2.2.2 Validación de los requisitos funcionales

Un detalle importante luego de captados y especificados los requisitos funcionales del software a implementar, es la validación de los mismos. Para ello se cuentan con varias técnicas de las cuales se utilizaron las siguientes:

Prototipo de interfaz de usuario

Se realizaron prototipos de interfaz que simulasen las funcionalidades que debía cumplir el sistema, para representar la estructura de la aplicación al cliente en busca de aceptación.

Revisión técnica informal

Esta otra técnica, consiste en revisar de manera informal las especificaciones de requisitos (Historias de usuario) en busca de ambigüedades, errores de conceptos y verificando que cumpliesen con las necesidades planteadas por el cliente. Se logró mediante reuniones y revisiones en conjunto con el cliente.

2.2.3 Requisitos no funcionales

Si bien los requisitos funcionales son importantes por su impacto en el funcionamiento del software que se desea desarrollar, los requisitos no funcionales se centran en detallar las restricciones a las que se deben adaptar los sistemas informáticos, ya sean restricciones de tiempo, proceso, desarrollo o estándares.

A continuación se especifican los requisitos no funcionales asociados al sistema informático Xendero:

Portabilidad

- El sistema no debe ser dependiente de sistemas gestores de base de datos.
- El sistema debe permitir ejecutarse tanto en el sistema operativo Linux como en el sistema operativo Windows.

Interfaz

- El sistema debe presentar una interfaz sencilla, que brinde fácil interacción al usuario con la aplicación.

2.3 Arquitectura

Un paso importante en todo proceso de software, es la concepción y puesta en práctica de la arquitectura a emplear. Para el desarrollo del sistema informático Xendero, se define la arquitectura de cuatro capas: capa de presentación, capa de negocio, capa de acceso a datos y por último la capa de dominio.

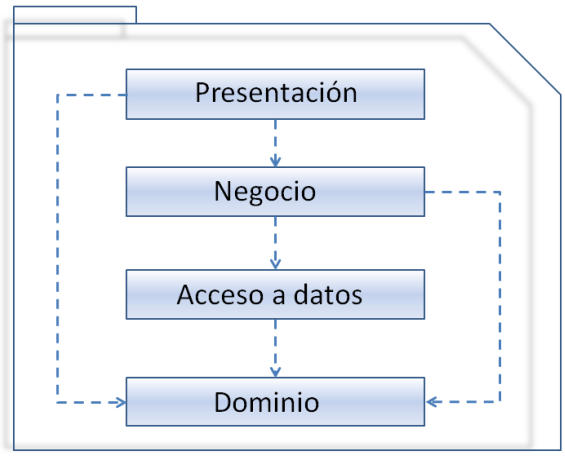


Figura 3: Arquitectura del sistema Xendero.

2.3.1 Capa de presentación

Esta capa contendrá todas las clases pertenecientes a las interfaces visuales de la aplicación (JFrame, JDialog), que se encargarán de brindarle al usuario un medioambiente de intercambio con el sistema, permitiendo hacer uso de todas sus funcionalidades. Controla el flujo de información hacia y desde la capa de negocio. Tiene acceso también a la capa de dominio.

2.3.2 Capa de negocio

La capa de negocio contiene todos los métodos que influyen en la lógica de negocio del sistema, apoyándose en el motor de reglas de negocio Drools para gestionar parte de la lógica sujeta a cambios. En su funcionamiento se sirve de la capa de acceso a datos, para el trabajo con la información que da soporte a los procesos del sistema. También cuenta con implementaciones de funcionalidades que facilitan el intercambio con la capa de presentación. De igual manera tendrá acceso a la capa de dominio.

2.3.3 Capa de acceso a datos

En esta capa se implementarán los métodos encargados de interactuar con la Base de Datos. Esta capa tendrá solamente dependencia con la Capa de Dominio y la interacción con la Capa de Negocio se realizará mediantes interfaces. Para el desarrollo de esta capa se utilizará el patrón Data Access Object (DAO por sus siglas en inglés).

2.3.4 Capa de dominio

En esta capa se declararán todas las clases que representan entidades del negocio y estarán accesibles desde todas las capas anteriormente descritas.

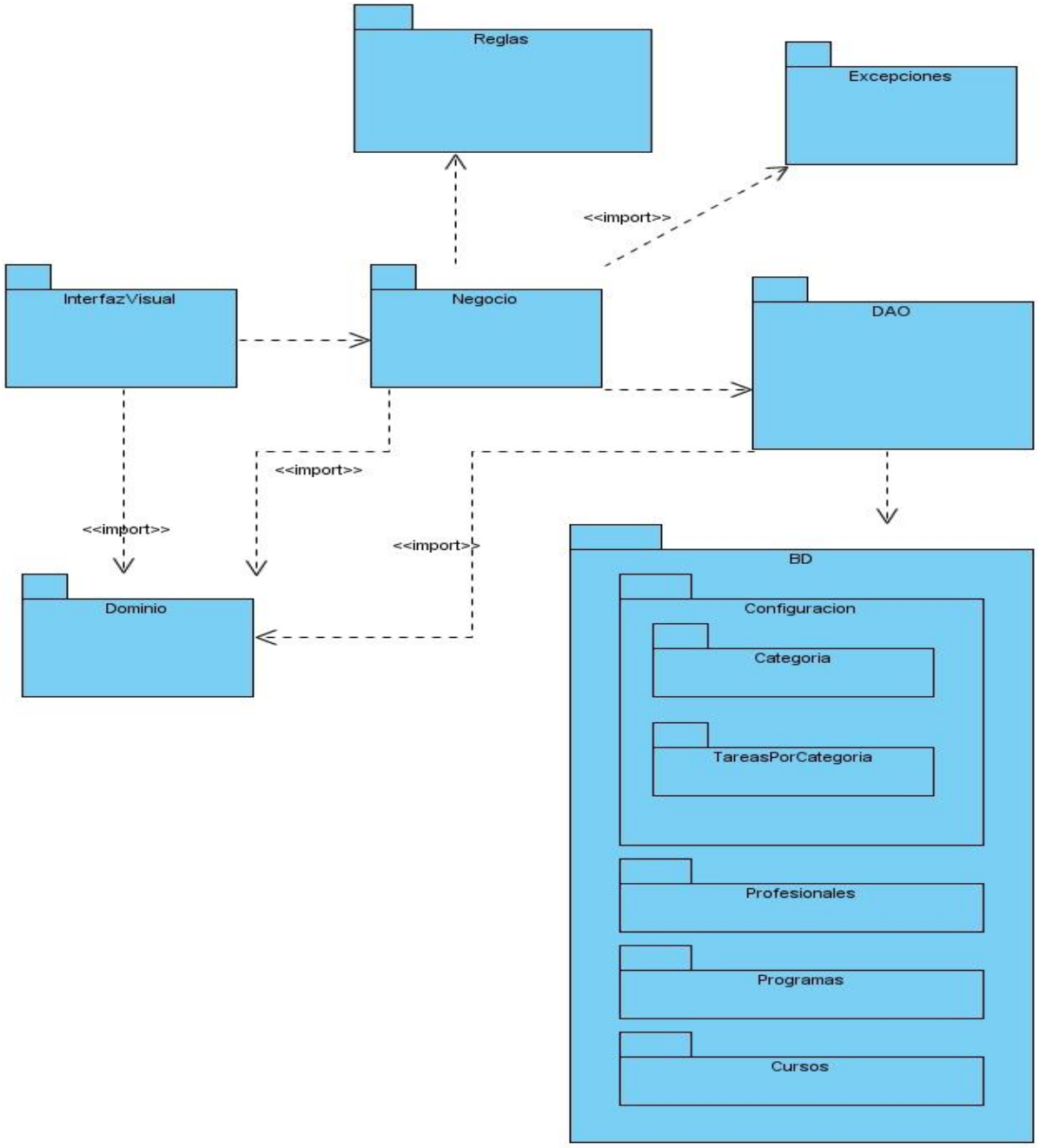
2.4 Diseño del sistema

En todo proceso de software, el diseño es parte esencial del desarrollo de sistemas informáticos, permitiendo modelar la solución que luego será objeto de implementación.

2.4.1 Plantilla modelo de diseño

SXP propone el artefacto (Plantilla modelo de diseño), el mismo consta de dos secciones principales. En la primera sección debe aparecer el diagrama de paquetes y en la segunda sección los diagramas de clases. Este artefacto describe los paquetes y sus relaciones al igual que las clases que darán soporte a la implementación del sistema.

Diagrama de paquetes

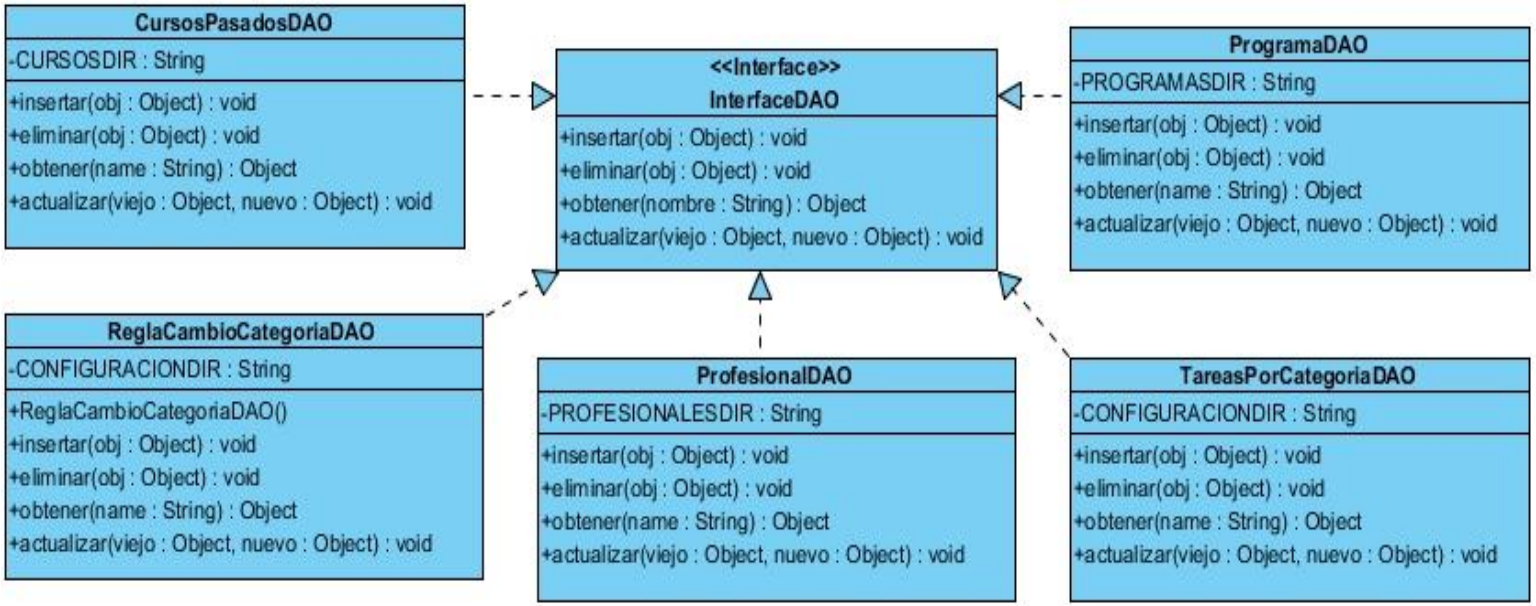


Observaciones

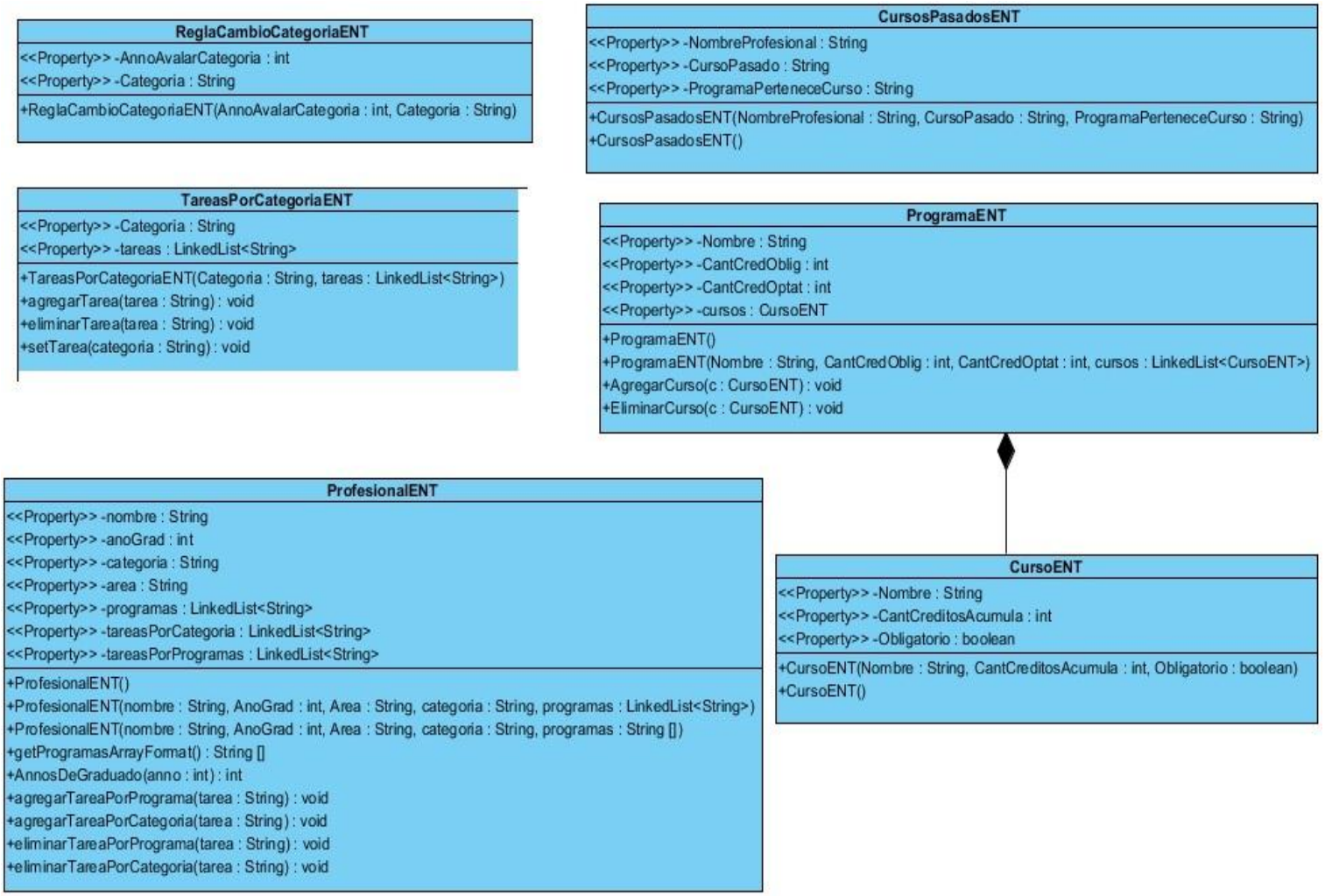
1) En el paquete Reglas se encuentran los ficheros de reglas de negocio manejadas con el motor de reglas Drools.

2) El paquete BD cuenta con una estructura interna de paquetes que conforman la base de datos compuesta también por ficheros xml.

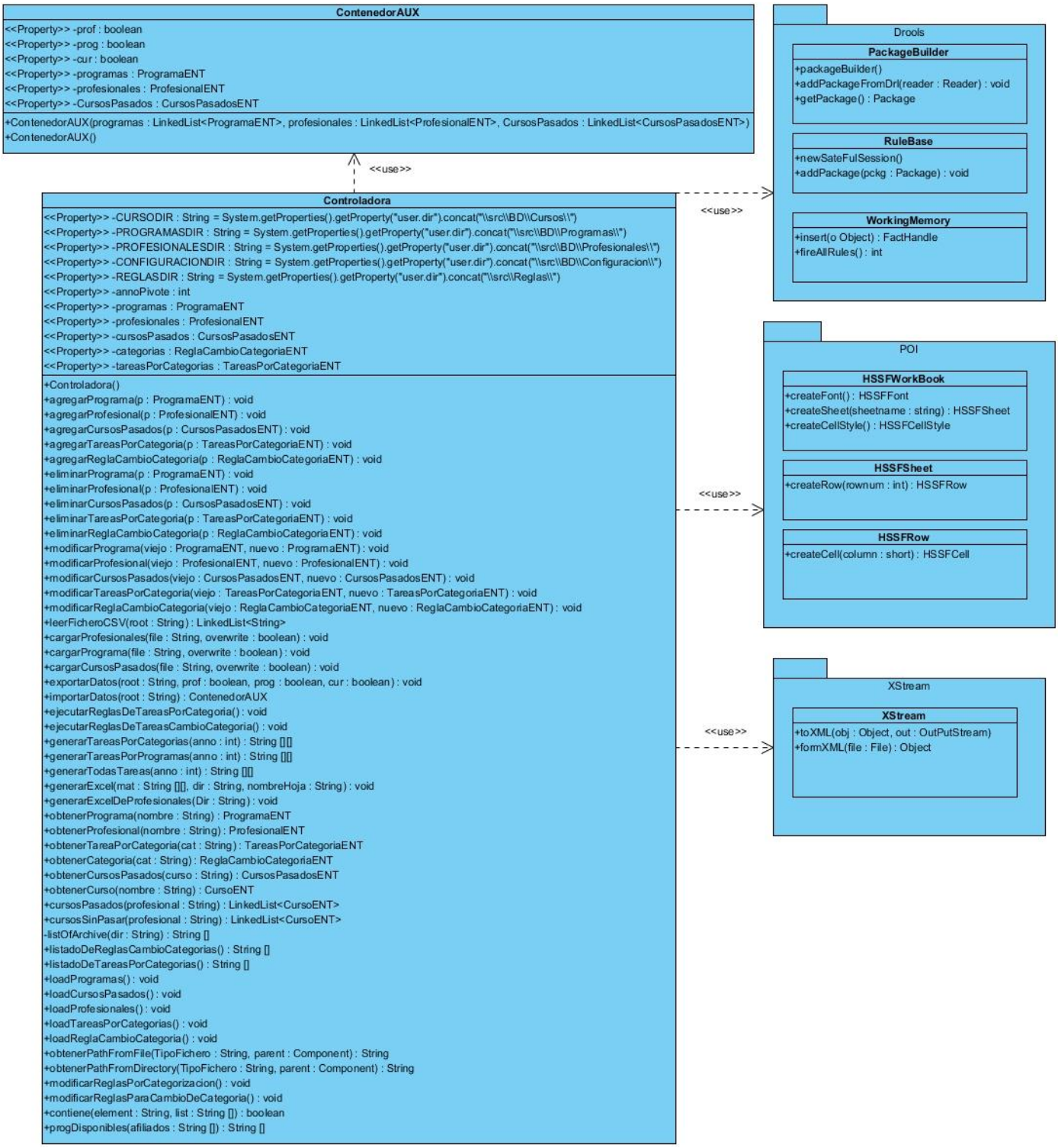
Capa de acceso a datos:



Capa de dominio:



Capa de negocio



2.5 Validación del diseño

Si bien es importante la realización del diseño, también es de gran peso la validación del mismo, dicha validación se lleva a cabo mediante métricas que permiten evaluar la calidad con que se ha realizado. Para la validación se utilizaron las métricas planteadas por Lorenz y Kidd Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC), estas métricas incluyen medidas de los siguientes atributos de calidad:

Responsabilidad: Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.

Complejidad del mantenimiento: Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la planificación del proyecto.

Complejidad de implementación: Grado de dificultad que tiene implementar un diseño de clases determinado.

Reutilización: Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.

Acoplamiento: Dependencia o interconexión de una clase o estructura de clase respecto a otras.

Cantidad de pruebas: Grado de esfuerzo necesario para realizar las pruebas de calidad al producto diseñado.

2.5.1 Métrica TOC: Tamaño operacional de clase

Se refiere al número de procedimientos existentes en una clase. Determina una relación directa entre los atributos Responsabilidad y Complejidad de implementación, sin embargo, establece una relación inversa entre estos últimos y el atributo Reutilización.

Los umbrales aplicados para la evaluación de la métrica Tamaño Operacional de Clase (TOC) se muestran a continuación:

Atributo de calidad	Categoría	Criterio	
Responsabilidad	Baja	<= promedio	<= 13
	Media	> promedio y <= 2*promedio	> 13 y <= 26
	Alta	> 2*promedio	> 26
Complejidad de implementación	Baja	<= promedio	<= 13
	Media	> promedio y <= 2*promedio	> 13 y <= 26
	Alta	> 2*promedio	> 26
Reutilización	Baja	> 2*promedio	> 26
	Media	> promedio y <= 2*promedio	> 13 y <= 26
	Alta	<= promedio	<= 13

Tabla 2: *Umbrales de la métrica Tamaño Operacional de Clases.*

Al evaluar esta métrica se obtuvieron los siguientes resultados:

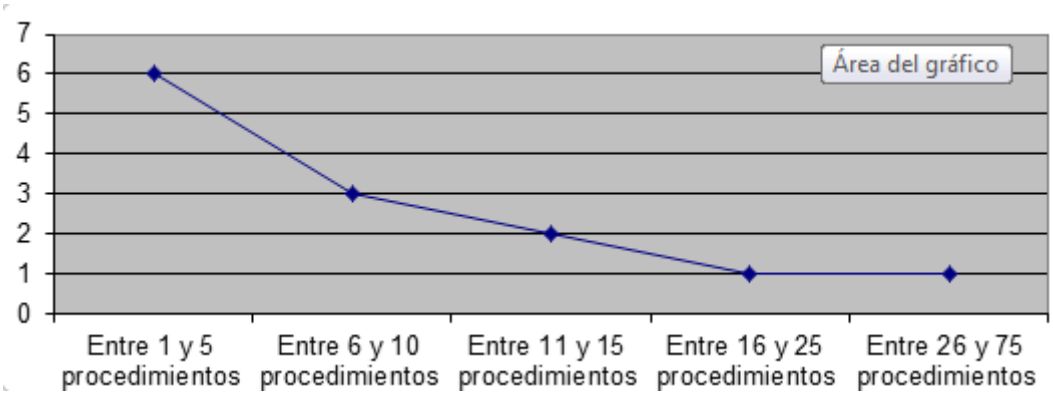


Figura 4: Representación de la cantidad de clases y el número procedimientos que contienen.

Representación en % de los resultados obtenidos, agrupados en los intervalos definidos:

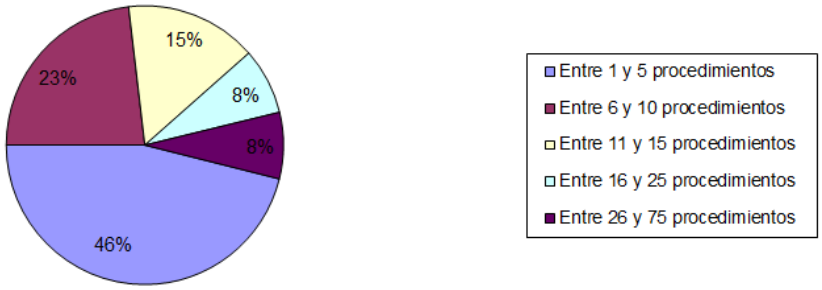


Figura 5: Representación en % de la cantidad de clase y el número procedimientos que contienen.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Responsabilidad**:

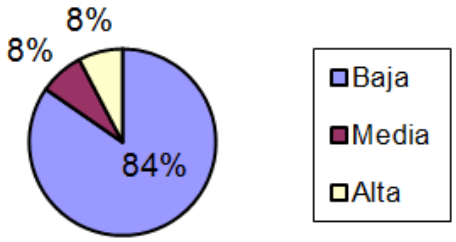


Figura 6: Representación del valor en % del atributo Responsabilidad.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Complejidad de Implementación.**

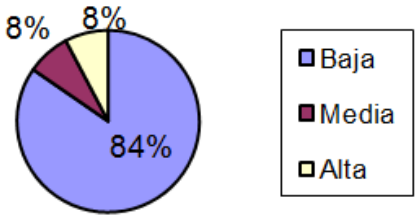


Figura 7: Representación del valor en % del atributo Complejidad de Implementación.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Reutilización.**

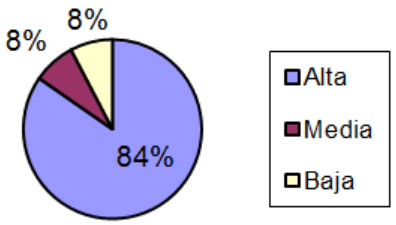


Figura 8: Representación del valor en % del atributo Reutilización.

Análisis de los resultados obtenidos en la evaluación de la métrica TOC

Luego de aplicadas las métricas se puede observar que un 84% de las clases presenta valores bajos de responsabilidad, o sea, valores inferiores al promedio de procedimientos por clases, por lo tanto se tornan más reutilizables y con una baja complejidad de implementación.

2.5.2 Métrica RC: Relaciones entre clases

Se refiere al número de relaciones de uso de una clase. Determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

Los umbrales aplicados para la evaluación de la métrica Relaciones entre Clases (RC) se muestran a continuación:

Atributo de calidad	Categoría	Criterio	
Acoplamiento	Ninguno	0	
	Bajo	1	
	Medio	2	
	Alto	> 2	
Complejidad de mantenimiento	Baja	<= promedio	<= 2
	Media	> promedio y <= 2*promedio	> 2 y <= 4
	Alta	> 2*promedio	> 4
Reutilización	Baja	> 2*promedio	> 4
	Media	> promedio y <= 2*promedio	> 2 y <= 4

	Alta	<= promedio	<= 13
Cantidad de pruebas	Baja	<= promedio	<= 2
	Media	> promedio y <= 2*promedio	> 2 y <= 4
	Alta	> 2*promedio	> 4

***Tabla 3**: Umbrales para la métrica Relaciones entre Clases.*

La evaluación de esta métrica arrojó los siguientes resultados:

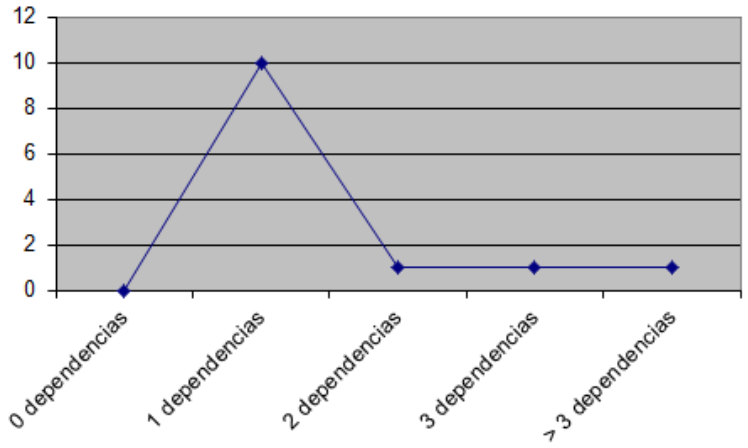


Figura 9: Representación de las asociaciones de uso por cantidad a de clases.

Representación en % de los resultados obtenidos, agrupados en los intervalos definidos:

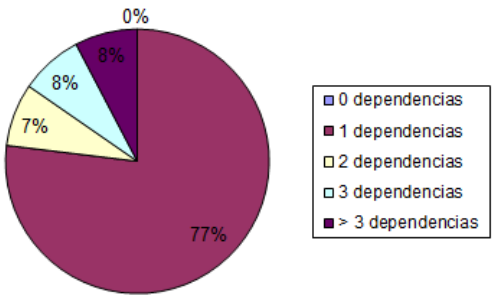


Figura 10: Representación en % de las asociaciones de uso por cantidad a de clases.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Acoplamiento**:

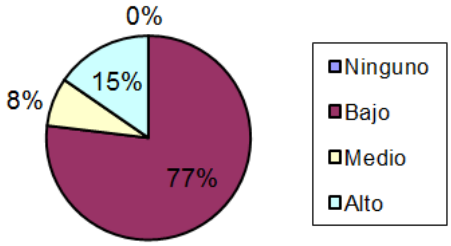


Figura 11: Representación del valor en % del atributo Acoplamiento.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Complejidad de Mantenimiento**.

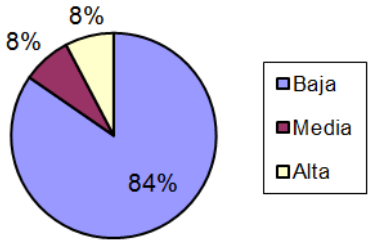


Figura 12: Representación del valor en % del atributo Complejidad de Mantenimiento.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Cantidad de Pruebas**.

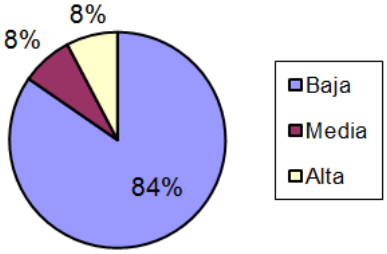


Figura 13: Representación del valor en % del atributo Cantidad de Pruebas.

Representación en % de la incidencia de los resultados obtenidos en el atributo **Reutilización**.

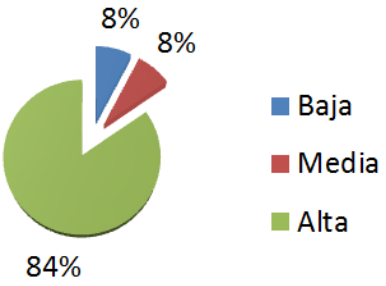


Figura 14: Representación del valor en % del atributo Reutilización.

Análisis de los resultados obtenidos en la evaluación de la métrica RC

Teniendo en cuenta los atributos de calidad aplicados, se puede notar que las clases diseñadas cuentan con un bajo acoplamiento, contribuyendo a una complejidad de mantenimiento baja. De igual manera la cantidad de comprobaciones que se produzcan cuando se efectúen modificaciones será baja. En consecuencia, el grado de reusabilidad será alto.

2.6 Patrones de diseño empleados

En aras de lograr los objetivos esperados, es muy frecuente el uso de patrones en la elaboración del diseño de los sistemas informáticos. Son consideradas soluciones experimentadas y simples a los problemas comunes y específicos del diseño. En el diseño de la solución propuesta, se usaron varios patrones, destacándose los GRASP (en inglés General Responsability Assigment Software Patterns), los mismos describen principios fundamentales de asignación de responsabilidad a objetos. Algunos de los utilizados se detallan a continuación. (17)

2.6.1 Experto

Este patrón indica que la responsabilidad de la creación de un objeto o la implementación de métodos, debe caer sobre la clase que conoce toda la información necesaria para hacerlo. Permitiendo el fácil entendimiento y mantenimiento de las clases. La representación de este patrón se puede ver por ejemplo en las clases de dominio, las mismas son expertas en tener los atributos que se le definen, otro ejemplo se tiene en la clase de acceso a datos ProgramaDAO, la cual es responsable de permitir la gestión de la información referente a los programas de superación en la base de datos. (17)

2.6.2 Alta cohesión

La estructura del sistema fue diseñada de manera tal que las clases tuviesen bien enmarcadas sus responsabilidades, organizadas por capas, logrando que fuese menos susceptible a los cambios, garantizando una alta cohesión y bajo acoplamiento. (17)

2.6.3 Bajo acoplamiento

Este patrón se centra en lograr que el nivel de dependencia entre las clases sea lo más bajo posible, de manera que al producirse un cambio se tenga mínima repercusión en el resto de las clases. Se encuentra representado en las clases de acceso a datos del software, cada una se centra en gestionar el acceso a la base de datos, encapsulando la información para las clases clientes, logrando que se relacionen exclusivamente con ellas para el intercambio con la base de datos, minimizando de esta forma el impacto de cambios posteriores. (17)

2.6.4 Controlador

La representación de este patrón se tiene en la clase Controladora del sistema, la cual se responsabiliza por recibir peticiones de las clases visuales enmarcadas en la capa de presentación, y de igual manera sostener la comunicación con la capa de negocio. (17)

Otros patrones utilizados son los patrones GOF, de los cuales se utilizaron los siguientes:

2.6.5 Estrategia (en inglés Strategy)

Este patrón, se aplica cuando muchas clases que están relacionadas solo se diferencian en su comportamiento o cuando se necesita diferentes variantes de un algoritmo. Se puede ver reflejado en la implementación de la InterfazDAO por las clases ProgramaDAO, ProfesionalDAO, TareasPorCategoriaDAO, CursosPasadosDAO, ReglaCambioCategoriaDAO, las cuales cuentan con implementación diferente de los métodos insertar, eliminar, actualizar, obtener, siendo comportamientos diferentes para un mismo método.

2.6.6 Fachada (en inglés Facade)

Este patrón se centra en proveer una interfaz intermediaria entre un cliente y un grupo de interfaces más complejas. Utilizado para reducir la dependencia entre clases, ya que ofrece un punto de acceso al resto

de las clases. Si estas cambian o se sustituyen por otras solo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones clientes. Este patrón se evidencia en la clase (InterfazDAO), la cual se encarga de servir de interfaz para las clases de acceso a datos.

2.7 Base de datos

Todo sistema informático que necesite información persistente, define una forma de almacenamiento para los datos que maneja, siendo muy común el empleo de bases de datos. Una base de datos, es un conjunto de información perteneciente a un mismo contexto y almacenados sistemáticamente para su posterior uso. A continuación se muestra la estructura de paquetes ubicados en el directorio raíz de la solución que conforman la base de datos del sistema. (18)

Figura 6: Estructura de la base de datos.

Uno de los requisitos no funcionales planteados para la conformación de la solución, es la no dependencia de un SGBD para el manejo de la información, permitiendo portabilidad y minimizando dependencias. Dando cobertura a este requisito no funcional, se creó una estructura de carpetas similar a la Figura 6, donde se guardan los datos, siendo gestionados por el sistema mediante la librería XStream, permitiendo el almacenado de la información en ficheros XML.

La carpeta (Configuración) cuenta con 2 subcarpetas, (Categoría y Tareas por categoría). La primera se encarga de almacenar la configuración de las reglas de cambio de categoría, existiendo un fichero por

cada configuración de regla de cambio. La segunda contiene varios XML, donde cada uno representa una configuración de las reglas de asignación de tareas por categoría docente.

La carpeta (Profesionales) contiene los datos personales de los profesionales, esta información es extraída de los objetos instanciados de la clase Profesional. Por otra parte la carpeta (Programas) almacena los diferentes programas posibles a cursar por los profesionales. Y por último la carpeta (Cursos) guarda los datos concernientes a los cursos concluidos por los profesionales.

2.8 Conclusiones

El presente capítulo, fue crucial en la definición y descripción de la arquitectura de la solución y la base de datos que se encargará de la persistencia de la información. También se realizó la captura y especificación de requisitos del sistema, permitiendo identificar las necesidades del cliente, las cuales se convertirán en funcionalidades del software resultante. Teniendo como origen los requisitos y las tecnologías seleccionadas en el capítulo anterior para la modelación, se realizó el diseño utilizando patrones, obteniéndose como resultado el cumplimiento del segundo de los objetivos específicos. Por consiguiente, una vez terminado el diseño, se procedió a la validación utilizando métricas que mostraron indicadores de calidad aceptables para el diseño realizado, ayudando de esta manera a obtener una mayor claridad para la posterior implementación del sistema.

Capítulo 3: Implementación y prueba.

3.1 Introducción

En este capítulo, se presenta el estándar de codificación a utilizar en la implementación del sistema. También, se presentan aspectos esenciales de la implementación, referentes a las principales funcionalidades y se visualizan y explican fragmentos de código claves en el funcionamiento del software. Por otra parte se cuenta con los casos de prueba diseñados para comprobar el buen funcionamiento del sistema y el cumplimiento de las funcionalidades de las HU. De igual manera se muestran los resultados de las pruebas realizadas al sistema.

3.2 Implementación

La implementación del software comienza transformando el resultado del modelo de diseño en el código final, utilizando estándares de codificación.

3.2.1 Estándar de codificación

El estándar de codificación se enfoca en la estructura y apariencia física del código fuente del software que se desarrolla, para su mejor comprensión y mantenimiento. Define nomenclatura de objetos, métodos, funciones, variables, asegurando legibilidad del código entre diferentes programadores, facilitando el mantenimiento o actualización del software. Buscando lograr uniformidad en el desarrollo del sistema se definió una convención de código para cada capa mediante el artefacto Plantilla de estándar de codificación. A continuación se muestran los elementos comunes en el estándar de codificación para todas las capas:

1) Las clases deben estar ubicadas a razón de una por cada fichero java, estructuradas de la siguiente manera:

- Las primeras líneas serán las correspondientes al nombre del paquete al que pertenece dicha clase.
- Seguidamente se ubicaran las sentencias import.
- La siguiente sentencia será class o interface en dependencia de lo que se desee definir.
- Seguido de las variables estáticas.
- Después las variables de instancia quedando ordenadas por la visibilidad de la siguiente manera: publicas, protegidas, nivel de paquetes, privadas.
- Luego constructor(es).
- A continuación los métodos agrupados por funcionalidad.

2) Las líneas de código no deberán ser muy largas, teniendo como límite entre 70 u 80 caracteres y una sola sentencia.

3) La declaración de variables deberá hacerse teniendo en cuenta el uso de una por línea de código, con letra inicial minúscula y al principio de los bloques. Se evitará declaraciones de igual nombre que declaraciones de niveles superiores.

4) Los paquetes tendrán nombres con letra inicial minúscula.

5) Las clases serán nombradas mediante el estilo de escritura UpperCamelCase, el cual consiste en escribir con letra inicial mayúscula y en caso de ser compuesto el nombre, la letra inicial de cada palabra que lo componga será mayúscula.

6) El nombre de las interfaces sigue el mismo patrón que el de las clases.

7) Los métodos serán nombrados con el estilo de escritura lowerCamelCase, el mismo consiste en escribir con letra inicial minúscula y en caso de ser compuestos comienzan con letra inicial minúscula y la letra inicial del resto de las palabras que componen el nombre serán con mayúscula.

8) Los nombres de las variables declaradas como constantes serán escritas por completo en mayúscula.

El resto de los elementos del estándar de codificación se encuentran detallados por capas en la plantilla de estándar de codificación (Anexo: 1) en la cual pueden ser consultados.

3.2.2 Aspectos principales de la implementación.

Con basamento en el estándar de codificación descrito, se realizó la implementación del sistema utilizando las tecnologías para el manejo de las reglas de negocio, la base de datos y la generación de documentos MS Excel que contiene la planificación de las tareas de superación.

Utilización del motor de reglas Drools.

Para el manejo de algunas de las reglas de negocio del sistema; se hizo necesario que fuesen manipuladas desde la aplicación, permitiendo cambiarlas en tiempo de ejecución y que quedasen activas para futuras ejecuciones del software. El motor de reglas de negocio Drools, en su versión 5.3.0, brinda esta facilidad, permitiendo definir las reglas a utilizar en el negocio en ficheros *drl, los cuales serán cargados por el sistema e interpretados, dando la oportunidad de interactuar con funcionalidades implementadas en dependencia del negocio.

La estructura básica de un fichero de reglas del motor inferencial Drools, está constituida en sus primeras líneas por las importaciones de las clases que representan los diferentes tipos de objetos a manejar en las reglas a ser implementadas, seguido de la sentencia “rule” y el nombre de la regla. La próxima sentencia se encabeza con la palabra “when” y el compendio de condiciones a cumplirse para ejecutar la regla. Luego la palabra “then” da paso a las acciones a seguir en caso de cumplirse las condiciones descritas y por último finalizando con la sentencia “end”.

En caso de necesitar definir más reglas se comenzaría nuevamente con la sentencia “rule” y la especificación de la próxima regla hasta terminar las necesarias.

A continuación se muestra un pequeño fragmento que servirá para ejemplificar las reglas implementadas en el desarrollo del sistema:

1. import Dominio.ProfesionalENT

2. rule "Asistente"
3. when
4. pro: ProfesionalENT (pro.getCategoria() == "Asistente")
5. then
6. pro.agregarTareaPorCategoria("No menos de 2 publicaciones donde al menos 1 sea en revistas internacionales referenciadas como autor principal");
7. pro.agregarTareaPorCategoria("Participar en al menos 1 evento nacional");
8. pro.agregarTareaPorCategoria("Tutorar no menos de 2 estudiantes en eventos científicos");
9. end

En la primera línea se importa la clase ProfesionalENT, debido a que se trabaja con instancias de esa clase. En la segunda línea se muestra la definición del nombre de la nueva regla de negocio. La línea número 4, define la condición a cumplirse para ejecutar la regla, en este caso, es verificar la categoría docente que presenta el profesional que es evaluado, en esta ocasión es “Asistente”. En las líneas 6, 7 y 8 se le asigna al profesional que se está evaluando tareas de superación, utilizando el método agregarTareaPorCategoria().

De esta forma se puede describir las reglas de negocio en los ficheros afines a Drools y al modificarlos en tiempo de ejecución se lograría un funcionamiento diferente del sistema, sin necesidad de una nueva implementación de la aplicación, evitando una nueva compilación o mantenimiento de código.

Solo queda mostrar la integración de los ficheros desde el código Java para culminar el desarrollo de este tema particular. El siguiente método, es ejemplo de la llamada de los ficheros de reglas de Drools y su puesta en ejecución:

Primero se ubica el fichero de reglas a leer mediante la clase Reader.

```
public void ejecutarReglasDeTareasPorCategoria()
        throws DroolsParserException, IOException {

    Reader reader = new InputStreamReader(
            Controladora.class.getResourceAsStream(
            "/Reglas/Tareas por categoria.drl"));
```

Luego, utilizando el objeto de la clase Reader, se carga y compila el paquete de reglas.

```
    PackageBuilder pbuilder = new PackageBuilder();
    pbuilder.addPackageFromDrl(reader);
```

Después se verifica que ha culminado con éxito la compilación del paquete de reglas cargadas.

```
  if (pbuilder.hasErrors()) {
      throw new RuntimeException(
              pbuilder.getErrors().toString()
              + "\nNo se pudo compilar el archivo de reglas.");
  }
```

Luego se agrega a la base de reglas el paquete compilado.

```
Package pkg = pbuilder.getPackage();
RuleBase ruleBase = RuleBaseFactory.newRuleBase();
ruleBase.addPackage(pkg);
WorkingMemory workingMemory = ruleBase.newStatefulSession();
```

Por último se agregan los objetos a la memoria de trabajo y se ejecutan las reglas para dicha memoria de trabajo.

```
for (int i = 0; i < profesionales.size(); i++) {
    workingMemory.insert(profesionales.get(i));
}

workingMemory.fireAllRules();
```

De esta manera, quedan descritos los pasos realizados para ejecutar las reglas de negocio implementadas para la asignación de tareas en función de la categoría docente de los profesionales, una de las funcionalidades claves del sistema.

Librería para la creación de ficheros MS Excel (POI).

Otra de las principales funcionalidades del software, es la generación del fichero MS Excel que contiene las tareas de superación para los profesionales. Cuya funcionalidad se implementó utilizando la librería POI en su versión 3.0.2.

El método que se describe a continuación, recibe por parámetro una matriz de cadenas de texto, siendo la información a imprimir en el fichero MS Excel de salida, también una cadena de texto con la dirección donde se generará el mismo y por último otra cadena con el nombre de la hoja de cálculo del libro MS Excel que se creará.

Acto seguido, se muestran las líneas de código concernientes a la creación del libro de MS Excel y la asignación de la hoja de cálculo que contendrá la información obtenida por parámetro, seguido de la especificación del formato que se le establecerá a las celdas.

```
public void generarExcel(String[][] mat,
        String dir,
        String nombreHoja) throws FileNotFoundException, IOException {

    HSSFWorkbook wb = new HSSFWorkbook();
    HSSFSheet sheet = wb.createSheet(nombreHoja);
    HSSFFont FontOfCabecera = wb.createFont();
    FontOfCabecera.setBoldweight(Font.BOLDWEIGHT_BOLD);
    HSSFCellStyle cabecera = (HSSFCellStyle) wb.createCellStyle();
    cabecera.setFillForegroundColor(IndexedColors.LIGHT_BLUE.getIndex());
    cabecera.setFont(FontOfCabecera);
    HSSFCellStyle datos = (HSSFCellStyle) wb.createCellStyle();
```

Luego, utilizando la matriz recibida por parámetro, se va generando las filas y celdas que conformarán la estructura de la hoja de cálculo, incluido el formato anteriormente definido para las celdas.

```
for (int i = 0; i < mat.length; i++) {
    HSSFRow row = sheet.createRow(i);
    for (int j = 0; j < mat[i].length; j++) {
        HSSFCell cell = row.createCell(((short) (j)));
        if (i == 0) {
            cell.setCellStyle(cabecera);
        } else {
            cell.setCellStyle(datos);
        }
        cell.setCellValue(mat[i][j]);
    }
}
```

Por último, se utiliza la clase FileOutputStream para guardar la estructura definida como un libro de MS Excel en la dirección pasada por parámetro.

```
FileOutputStream out = new FileOutputStream(dir);
wb.write(out);
out.close();
```

Librería XStream.

En la implementación, fue de vital importancia el uso de la librería XStream para la persistencia de los datos, logrando contenerlos en una estructura de carpetas, pudiendo obtenerlos en otras futuras ejecuciones de la aplicación. A continuación se describen dos métodos que visualizan la gestión de los ficheros XML mediante esta librería:

El método insertar de la clase ProfesionalDAO comienza validando si el objeto pasado por parámetro es instancia de ProfesionalENT.

```
public void insertar(Object obj)
        throws ErrorAlGuardar,
        InstanciaDeObjetoNoValido {
    if (obj instanceof ProfesionalENT) {
        ProfesionalENT c = (ProfesionalENT) obj;
```

Luego mediante la llamada al método toXML(), al cual se le pasa por parámetro el objeto a ser almacenado en formato XML, en la dirección de memoria hacia la que apunte el objeto de la clase FileOutputStream, pasado también por parámetro a este método.

```
try {
    XStream xstream = new XStream();
    FileOutputStream fo = new FileOutputStream(
            PROFESIONALESDIR.concat(
            c.getNombre().concat(".xml")));
    xstream.toXML(c, fo);
    fo.close();
} catch (Exception ex) {
    throw new ErrorAlGuardar("Error al guardar el"
            + " profesional en la base de datos.");
}
```

Por otra parte, el método obtener() de la misma clase, describe el proceso inverso, permitiendo obtener una instancia de la clase ProfesionalENT desde un fichero XML. Esta vez utilizando el método fromXML().

```
public Object obtener(String name)
        throws InformacionNoEncontrada {
    XStream xstream = new XStream();
    try {
        FileInputStream fi = new FileInputStream(
                PROFESIONALESDIR.concat(name).concat(".xml"));
        ProfesionalENT c = (ProfesionalENT) xstream.fromXML(fi);
        fi.close();
        return c;
    } catch (Exception ex) {
    }
    throw new InformacionNoEncontrada("No se encontró el archivo.");
}
```

3.3 Prueba realizadas

Las pruebas constituyen una etapa imprescindible durante el proceso de desarrollo del software. Su objetivo principal es asegurar que el software cumpla con las especificaciones requeridas y eliminar los posibles defectos que este pudiera tener. (19) De forma más específica los propósitos de las pruebas son:

- Realizar la validación del software desarrollado, entendiendo como validación el proceso que determina si el software satisface los requisitos.
- Realizar la verificación del software desarrollado, entendiendo como verificación el proceso que determina si los productos de una fase satisfacen las condiciones de la misma.

3.3.1 Métodos de pruebas

Los métodos de pruebas, contribuyen a encontrar un conjunto de casos de prueba que ayuden a detectar diferentes tipos de errores. El método de caja blanca se centra en las funciones internas, usando la estructura de control del diseño procedimental para derivar los casos de prueba. Por otra parte, el método de caja negra, se enfoca sobre la interfaz del sistema que se desea validar, proporcionando unas entradas y estudiando las salidas, evaluando si concuerdan con los resultados esperados. (19)

3.3.2 Método de prueba utilizado

Para verificar el correcto funcionamiento del software, se realizaron pruebas funcionales aplicando el método de caja negra y específicamente la técnica de partición de equivalencia. En el Anexo (2), se describe uno de los casos de prueba creados.

3.3.3 Resultados de las pruebas realizadas

Luego de realizadas las pruebas al software por el departamento de calidad del CEIGE, se obtuvieron los siguientes resultados.

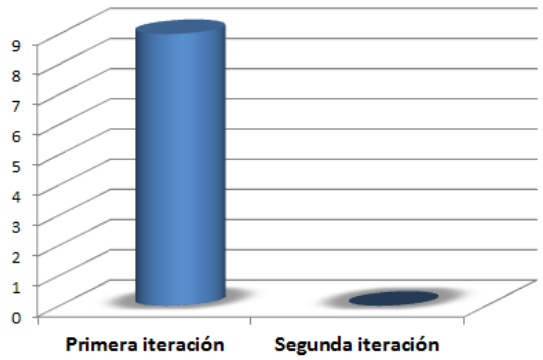


Figura 15: Resultados de las pruebas realizadas.

En la primera iteración se encontraron un total de nueve no conformidades, atendidas debidamente, logrando solucionarlas para la segunda iteración en la cual no se encontraron nuevas no conformidades.

Permitiendo de esta manera la liberación del software, avalando el buen funcionamiento del sistema desarrollado.

3.4 Conclusiones

En el presente capítulo culminado, haciendo uso del diseño realizado, se procedió a la implementación del sistema, siguiendo el estándar de codificación concebido, en busca de un mejor entendimiento entre los integrantes del equipo de desarrollo y permitiendo generar código con mayores facilidades de mantenimiento. Luego de concluida la implementación se crearon diseños de casos de prueba de caja negra, utilizados en las pruebas realizadas al sistema por el departamento de calidad del CEIGE, pudiendo constatar el buen desempeño de las funcionalidades implementadas. Todo contribuyó al cumplimiento de los objetivos específicos: Implementar el sistema de planificación de tareas de superación y Validar el sistema de planificación de tareas de superación.

Conclusiones generales

En el desarrollo de la presente investigación se obtuvo como resultado el cumplimiento de los objetivos trazados, permitiendo arribar a las conclusiones siguientes:

- El análisis del ambiente nacional e internacional, permitió reconocer la no existencia de una solución informática que diese soporte a las necesidades planteadas por el cliente.
- La modelación y construcción de la solución propuesta viabiliza el proceso de planificación de tareas de superación de los profesionales del CEIGE.
- La solución es totalmente confiable en los resultados que brinda y es portable, no dependiente de gestores de base de datos y disponible en todos los sistemas operativos en los que brinda soporte la máquina virtual de Java.
- Las pruebas realizadas a la solución, validan la estabilidad y confiabilidad del sistema implementado.
- Por tanto el sistema desarrollado Xendero, representa una notable mejora en el proceso de planificación de tareas de superación.

Recomendaciones

1. Para incrementar los beneficios de un sistema como el propuesto en esta investigación, se recomienda agregar una funcionalidad que permita enviar las planificaciones de tareas de superación por correo a los profesionales implicados u otros destinatarios de interés.
2. Desde el punto de vista estructural del fichero MS Excel resultante de la planificación de tareas de superación, permitir separar la información por pestañas desde la interfaz visual para una mejor organización.

Bibliografía

1. **Sitio dirección de postgrado UH**. [En línea] http://www.vriep.uh.cu/index.php?option=com_content&view=section&id=5&Itemid=53.

2. **Sitio dirección de postgrado UH**. [En línea] http://www.vriep.uh.cu/index.php?option=com_content&view=category&layout=blog&id=34&Itemid=58.

3. **Larman, Craig.** *UML y Patrones.* s.l.: Prentice Hall, 1999. 970-17-0261-1.

4. **Sitio Web Periódico Granma**. [En línea] 03 de 2012. http://www.granma.cubaweb.cu/2012/02/23/nacional/artic07.html.

5. **Ruiz, Olga Díaz.** Retos y perspectivas del posgrado en Cuba. *Granma.* Única, 2012, 52.

6. **Sitio de la Real Academia Española**. [En línea] 04 de 2012. http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=lenguaje%20de%20programacion.

7. **Centro de estudios e investigaciones técnicas España** [En línea] http://www.ceit.es/Asignaturas/Informat2/Clases/Clases9899/Clase01/JavaEntorno/tsld003.htm.

8. **Chi, MC Genaro Alberto Gómez.** [En línea] http://es.scribd.com/doc/19162245/Unidad-5-Modelo-Desarrollo-Software.

9. **Romero, Gladys Marsi Peñalver.** *MA-GMPR-UR2 Metodología ágil para proyectos de software libre.* La Habana : s.n., 2008.

10. **Sitio oficial Universidad de Flores Argentina**. [En línea] http://institucional.uflo.edu.ar/2011/files/investigacion/4-sistema-de-categorizaci-n.pdf.

11. *Resolución para la categorización docente.* **MES, Ministerio de Educación Superior.** 128, La Habana : s.n., 2006.

12. **Sitio oficial JBoss**. [En línea] http://www.jboss.org/drools.

13. **Universidad de Chile.** Universidad de Chile. [En línea] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

14. **Sitio oficial Java**. [En línea] http://www.java.com.

15. **definicion.ogg**. [En línea] http://www.definicion.org/lenguaje-de-programacion.

16. **utopicainformatica.com**. [En línea] http://www.utopicainformatica.com/2010/12/lenguajes-de-modelado.html.

17. mastermagazine.info. [En línea] http://www.mastermagazine.info/termino/7006.php.

18. **scribd.com**. [En línea] http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

19. **proyectosagiles.org**. [En línea] http://www.proyectosagiles.org/que-es-scrum.

20. **Sitio oficial XStream**. [En línea] 03 de 2012. http://xstream.codehaus.org/.

21. **Sitio oficial POI**. [En línea] 03 de 2012. http://poi.apache.org.

22. **Sitio oficial Visual Paradigm**. [En línea] 04 de 2012. http://www.visual-paradigm.com/.

23. Introducción al concepto de planificación estratégica. Villalaz, **Ing. Luis Pimentel**, Santa Clara: Biblioteca de la red de la ciencia en Villa Clara, 1999.

24. **Wiederhold, Gio**. Desing Database. Nueva York: s.n., 1977.

25. Curso de doctorado sobre procesos de software y gestión del conocimiento. **Usaola, Dr. Macario Polo**. 2006.

Referencias bibliográficas

1. **Sitio dirección de postgrado UH**. [En línea] http://www.vriep.uh.cu/index.php?option=com_content&view=section&id=5&Itemid=53.

2. **Ruiz, Olga Díaz.** Retos y perspectivas del posgrado en Cuba. *Granma.* Única, 2012, 52.

3. **Scribd** [En línea] http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

4. [En línea] http://www.rastersoft.com/OS2/CURSO/APIEXPL.HTM.

5. **Romero, Gladys Marsi Peñalver.** *MA-GMPR-UR2 Metodología ágil para proyectos de software libre.* La Habana : s.n., 2008.

6. **proyectoagiles.org** [En línea] http://www.proyectosagiles.org/que-es-scrum.

7. [En línea] http://www.ecured.cu/index.php/Diagrama_Entidad_Relaci%C3%B3n

8. **saberia.com** [En línea] http://www.saberia.com/2010/01/que-es-un-plugin/

9. **Sommerville, Ian. 2005.** *Ingeniería de Software.* 2005.

10. Introducción al concepto de planificación estratégica. Villalaz, **Ing. Luis Pimentel**, Santa Clara: Biblioteca de la red de la ciencia en Villa Clara, 1999.

11. **utopicainformatica.com**. [En línea] http://www.utopicainformatica.com/2010/12/lenguajes-de-modelado.html.

12. **Sitio oficial Visual Paradigm**. [En línea] 04 de 2012. http://www.visual-paradigm.com/.

13. **Sitio oficial Java**. [En línea] http://www.java.com.

14. **Sitio oficial POI**. [En línea] 03 de 2012. http://poi.apache.org.

15. **Sitio oficial XStream**. [En línea] 03 de 2012. http://xstream.codehaus.org/.

16. **Romero, Gladys Marsi Peñalver.** *MA-GMPR-UR2 Metodología ágil para proyectos de software libre.* La Habana : s.n., 2008.

17. **2010.** El mundo informático. [En línea] [Citado el: 03 de 2012.] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.

18. **Wiederhold, Gio**. Desing Database. Nueva York: s.n., 1977.

19. Curso de doctorado sobre procesos de software y gestión del conocimiento. **Usaola, Dr. Macario Polo**. 2006.

Anexos

Anexo 1: Plantilla de estándar de codificación

Estándar de codificación sistema Xendero
Elementos comunes Para el estándar básico inicial común en todas las capas se utilizaron las convenciones de código para el lenguaje de programación Java. 9) Las clases deben estar ubicadas a razón de una por cada fichero java, estructuradas de la siguiente manera: • Las primeras líneas serán las correspondientes al nombre del paquete al que pertenece dicha clase. • Seguidamente se ubicaran las sentencias import. • La siguiente sentencia será class o interface. • Seguido de las variables estáticas. • Después las variables de instancia quedando ordenadas por la visibilidad de la siguiente manera: publicas, protegidas, nivel de paquetes, privadas. • Luego constructor(es). • A continuación los métodos agrupados por funcionalidad. 10) Las líneas de código se evitará sean muy largas, teniendo como límite entre 70 u 80 caracteres y una sola sentencia. 11) La declaración de variables se harán una por línea de código, con letra inicial minúscula y al principio de los bloques. Se evitará declaraciones de

igual nombre que declaraciones de niveles superiores.

12) Los paquetes tendrán nombres con letra inicial minúscula.

13) Las clases serán nombradas mediante el estilo de escritura UpperCamelCase, el cual consiste en escribir con letra inicial mayúscula y en caso de ser compuesto el nombre, la letra inicial de cada palabra que lo componga será mayúscula.

14) El nombre de las interfaces sigue el mismo patrón que el de las clases.

15) Los métodos tendrá serán nombrados con el estilo de escritura lowerCamelCase, el mismo consiste en escribir con letra inicial minúscula y en caso de ser compuestos comienzan con letra inicial minúscula y la letra inicial del resto de las palabras que componen el nombre serán con mayúscula.

16) Los nombres de las variables declaradas como constantes serán escritas por completo en mayúscula.

Capa de presentación

La capa de presentación presenta particularidades en el estándar de codificación, las mismas son descritas a continuación:

1) Las clases visuales cumplirán los mismos requisitos descritos en los elementos comunes para la formulación del nombre y se le añadirá al final la letra V, para identificarlas.

2) En caso de existir clases auxiliares, usadas por las clases encargadas de las interfaces visuales, llevaran al final la terminación AUX.

Capa de negocio

El estándar definido para las implementaciones referentes a la capa de negocio

se describe a continuación:

1) La clase controladora principal se nombrará (Controladora), las clases controladoras específicas de un subproceso del negocio empezaran con el nombre del subproceso que controla seguido de la palabra (Controladora).

2) En caso de la existencia de alguna clase auxiliar, la misma tendrá como terminación AUX.

Capa de acceso a datos

Para la capa de acceso a datos se define a continuación el estándar de codificación:

1) Las clases encargadas de gestionar la conexión de la base de datos con las clases entidades del dominio empezaran con el nombre de la entidad en específico y terminarán con el sufijo DAO.

Capa de dominio

Esta capa contendrá solo las clases entidades del negocio y su estándar es el siguiente:

1) Comienza con el nombre específico seguido del sufijo ENT.

Anexo 2: Diseño de Caso de prueba de caja negra para HU_2 Gestionar reglas de cambio de categoría docente.

1. **Descripción General.**

   ✓ Se tiene como objetivo que el usuario gestione las reglas para asignar la tarea que define el cambio de categoría de los profesionales.

2. **Condiciones de Ejecución.**

   ✓ El usuario debe encontrarse en la ventana gestión de reglas para cambio de categoría.

3. **Secciones a probar en la historia de usuario.**

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC Agregar regla para cambio de categoría docente.	EC Agregar regla para cambio de categoría con éxito.	Se agrega la nueva regla para el cambio de categoría introduciendo los datos en los campos de texto mostrados en la interfaz visual, acto seguido se presiona el botón agregar. El sistema a continuación valida los datos y agrega la nueva regla.
	EC Agregar regla para cambio de categoría con datos incorrectos.	En el proceso de agregar una nueva regla para el cambio de categoría pueden no haberse introducido los datos pertinentes en alguno de los campos o valores incorrectos. El sistema en ese caso muestra un mensaje de error certificando que la entrada no es válida.
SC Eliminar regla para cambio de categoría docente.	EC Eliminar regla para cambio de categoría con éxito.	Se elimina la regla para el cambio de categoría seleccionando en el listado la regla a eliminar, acto seguido se presiona el botón eliminar. El sistema a continuación procede a eliminar la regla.
	EC Eliminar regla para cambio de categoría con pasos incorrectos.	En el proceso de eliminar una regla para el cambio de categoría pueden no seleccionar del listado la regla a eliminar o intentar presionar el botón eliminar sin que existan reglas en el listado. El sistema en ese caso muestra un mensaje de error certificando el mal

		proceder.
SC Modificar regla para cambio de categoría docente.	EC Modificar regla para cambio de categoría éxito.	Se modifica una regla seleccionándola en el listado y luego introduciendo los datos en los campos de texto mostrados en la interfaz visual, acto seguido se presiona el botón modificar. El sistema a continuación valida los datos y modifica la regla.
	EC Modificar regla para cambio de categoría con datos incorrectos o mal proceder.	En el proceso de modificar una regla para el cambio de categoría, pueden no haberse introducido los datos pertinentes en alguno de los campos o valores incorrectos, de igual manera pudiese no haberse seleccionado la regla a ser modificada. El sistema en ese caso muestra un mensaje de error certificando que la entrada no es válida o que no se ha seleccionado la regla objeto de modificación.

1. **Descripción de variable.**

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Años de graduado para avalar categoría	Campo de texto	No	Campo numérico.
2	Categoría	Campo de texto	No	

2. Matriz de Datos.

Escenario	Años de graduado para avalar categoría	Categoría	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC Agregar regla para cambio de categoría con éxito.	2	Instructor	El sistema agrega la nueva regla para el cambio de categoría a la base de datos y actualiza el listado de reglas.	Satisfactoria	1. El usuario selecciona la siguiente ruta en el menú principal del sistema: (Regla -> Cambio de categoría). 2. Luego escribe los datos en los campos y presiona el botón (Agregar). 3. El sistema valida los datos entrados. 4. El sistema agrega la nueva regla y la muestra en el listado.
EC Agregar regla para cambio de categoría con datos incorrectos.	dos	Instructor	El sistema alerta la existencia de datos incorrectos.	Satisfactoria	1. El usuario selecciona la siguiente ruta en el menú principal del sistema: (Regla -> Cambio de categoría). 2. Luego escribe los datos en los campos y presiona el botón (Agregar). 3. El sistema valida los datos

					entrados y muestra el mensaje: "Error al escribir el año de graduado.". 4. El usuario corrige los datos de entrada y presiona (Agregar). 5. El sistema agrega la nueva regla y la muestra en el listado.
	2	Null	El sistema alerta la existencia de datos incorrectos.	Satisfactoria	1. El usuario selecciona la siguiente ruta en el menú principal del sistema: (Regla -> Cambio de categoría). 2. Luego escribe los datos en los campos y presiona el botón (Agregar). 3. El sistema valida los datos entrados y muestra el mensaje: "Dato(s) de entrada no valido(s).". 4. El usuario corrige los datos de entrada y presiona (Agregar). 5. El sistema agrega la nueva regla y la muestra en el listado.
EC Eliminar regla para cambio de categoría con	---	---	El sistema elimina la regla seleccionada y actualiza el listado de	Satisfactoria	1. El usuario selecciona la siguiente ruta en el menú principal del sistema: (Regla ->

éxito.			**reglas.**		**Cambio de categoría).** **2. Luego selecciona la regla que desea eliminar del listado de reglas existentes.** **3. A continuación presiona el botón (Eliminar).** **4. Se elimina la regla seleccionada y se actualiza el listado de reglas.**
EC Eliminar regla para cambio de categoría con pasos incorrectos.	**---**	**---**	**El sistema alerta un mal proceder.**	**Satisfactoria**	**1. El usuario selecciona la siguiente ruta en el menú principal del sistema: (Regla -> Cambio de categoría).** **2. A continuación presiona el botón (Eliminar) sin antes haber seleccionado en el listado la regla a eliminar.** **3. El sistema muestra un mensaje notificando el mal proceder: "No hay categoría seleccionada para eliminar".** **4. Luego el usuario selecciona la regla que desea eliminar del listado de reglas existentes.** **5. Acto seguido presiona el botón**

					(Eliminar). 6.Se elimina la regla seleccionada y se actualiza el listado de reglas.
EC Modificar regla para cambio de categoría éxito.	5	Asistente	El sistema actualiza con los datos introducidos la regla seleccionada en el listado.	Satisfactoria	1.El usuario selecciona la siguiente ruta en el menú principal del sistema: (Regla -> Cambio de categoría). 2.Selecciona en la lista la regla a ser modificada. 3.A continuación introduce los datos en los campos y presiona el botón (Modificar). 4.El sistema valida los datos y modifica la regla seleccionada. 5.El sistema actualiza la lista de reglas.
EC Modificar regla para cambio de categoría con datos incorrectos.	Cinco	Asistente	El sistema alerta la entrada de datos incorrectos.	Satisfactoria	1. El usuario selecciona la siguiente ruta en el menú principal del sistema: (Regla -> Cambio de categoría). 2.Selecciona en la lista la regla a ser modificada. 3.A continuación introduce los

					datos en los campos y presiona el botón (Modificar). 4. El sistema alerta de errores en la entrada de datos mediante el mensaje siguiente: "Error al escribir el año de graduado.". 5. El usuario selecciona nuevamente la regla a modificar en el listado. 6. Luego rectifica los datos de entrada y presiona nuevamente el botón (Modificar). 7. El sistema valida los datos y modifica la regla seleccionada. 8. El sistema actualiza la lista de reglas.
	5	Null	El sistema alerta la entrada de datos incorrectos	Satisfactoria	1. El usuario selecciona la siguiente ruta en el menú principal del sistema: (Regla -> Cambio de categoría). 2. Selecciona en la lista la regla a ser modificada. 3. A continuación introduce los datos en los campos y presiona el botón (Modificar).

				4. El sistema alerta de errores en la entrada de datos mediante el mensaje siguiente: “Dato(s) de entrada no valido(s).”. 5. El usuario selecciona nuevamente la regla a modificar en el listado. 6. Luego rectifica los datos de entrada y presiona nuevamente el botón (Modificar). 7. El sistema valida los datos y modifica la regla seleccionada. 8. El sistema actualiza la lista de reglas.
	5	Asistente	El sistema alerta de un mal proceder para modificar una regla.	1. El usuario selecciona la siguiente ruta en el menú principal del sistema: (Regla -> Cambio de categoría). 2. A continuación introduce los datos en los campos y presiona el botón (Modificar). 3. El sistema alerta de error en la selección de la categoría a

		ser modificada mediante el mensaje siguiente: “No hay categoría seleccionada para modificar.”. **4. El usuario selecciona la regla a modificar en el listado.** **5. Luego y presiona el botón (Modificar).** **6. El sistema valida los datos y modifica la regla seleccionada.** **7. El sistema actualiza la lista de reglas.**

Anexo 3: Plantilla de Historia de usuario.

<table>
<tr><td colspan="3">Historia de Usuario</td></tr>
<tr><td>Código: HU_1</td><td colspan="2">Nombre Historia de Usuario: Importar datos para el sistema desde fichero de extensión csv.</td></tr>
<tr><td colspan="3">Modificación de Historia de Usuario Número: Ninguna.</td></tr>
<tr><td colspan="3">Referencia: Se tiene como objetivo que el usuario importe la información con que trabajará la aplicación.</td></tr>
<tr><td colspan="2">Programador: Álvaro Quiles López.</td><td>Iteración Asignada: 1</td></tr>
<tr><td colspan="2">Prioridad: Alta.</td><td>Puntos Estimados: 1</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: Medio.</td><td>Puntos Reales: 1</td></tr>
<tr><td colspan="3">Descripción: El usuario mediante un cuadro de diálogo busca la ubicación del fichero de extensión *.csv en el cual está la información a ser cargada por la aplicación. Existirán 3 estructuras diferentes en los ficheros en dependencia del tipo de datos a recibir:
1) El fichero referente a los datos personales de los profesionales, tendrá tantas filas como profesionales a ser cargados y la siguiente estructura por la que se regirá.

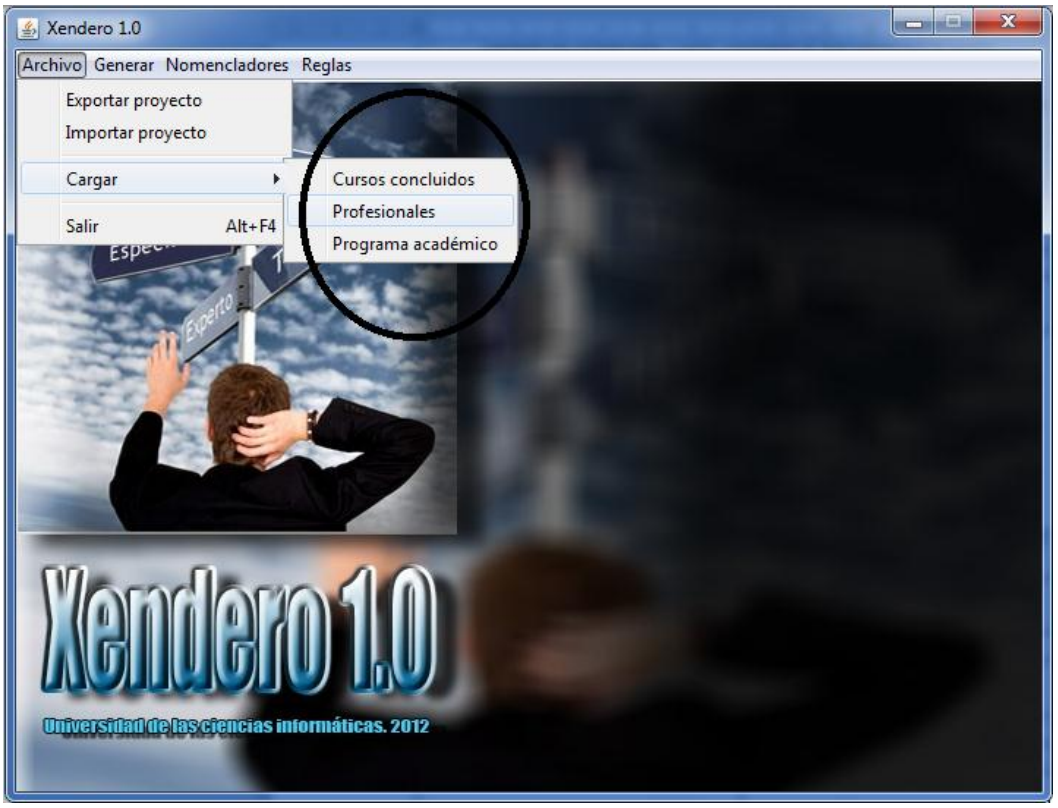
Nombre; AñoGraduado; Area; Categoria; Programa1; Programa2; ProgramaX.

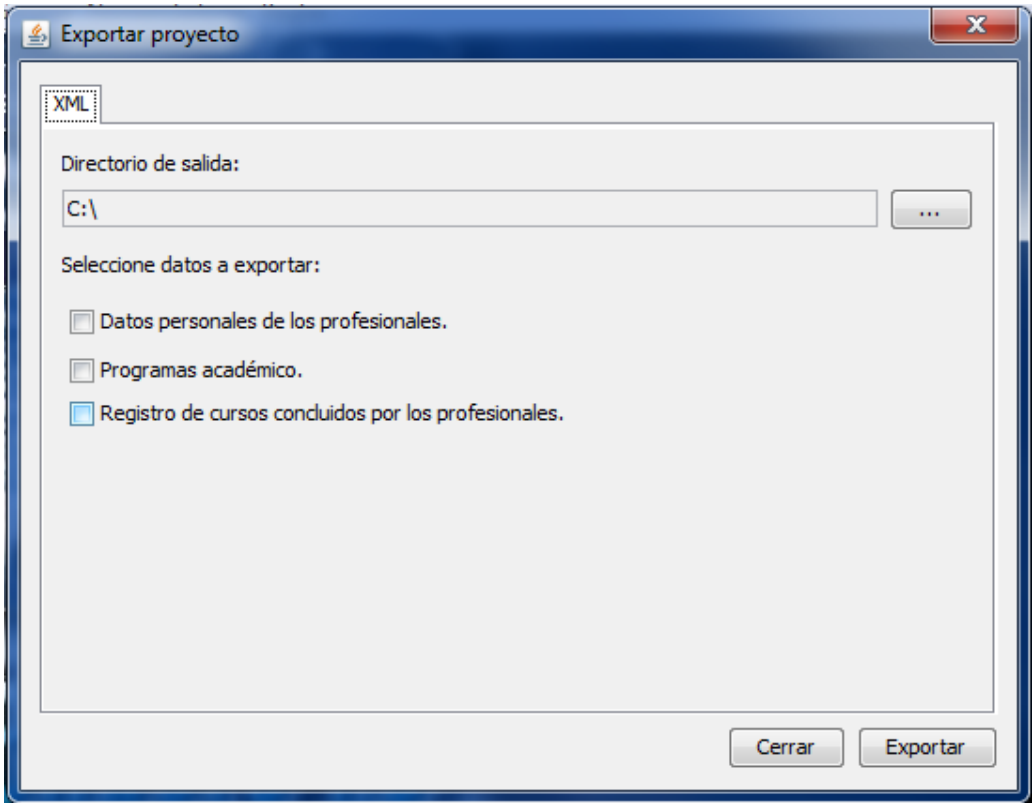
2) El fichero referente a los cursos concluidos por los profesionales, tendrá la siguiente estructura por la que se regirá. Donde la cantidad total de filas será equivalente a la cantidad de cursos concluidos por cada uno de los profesionales.

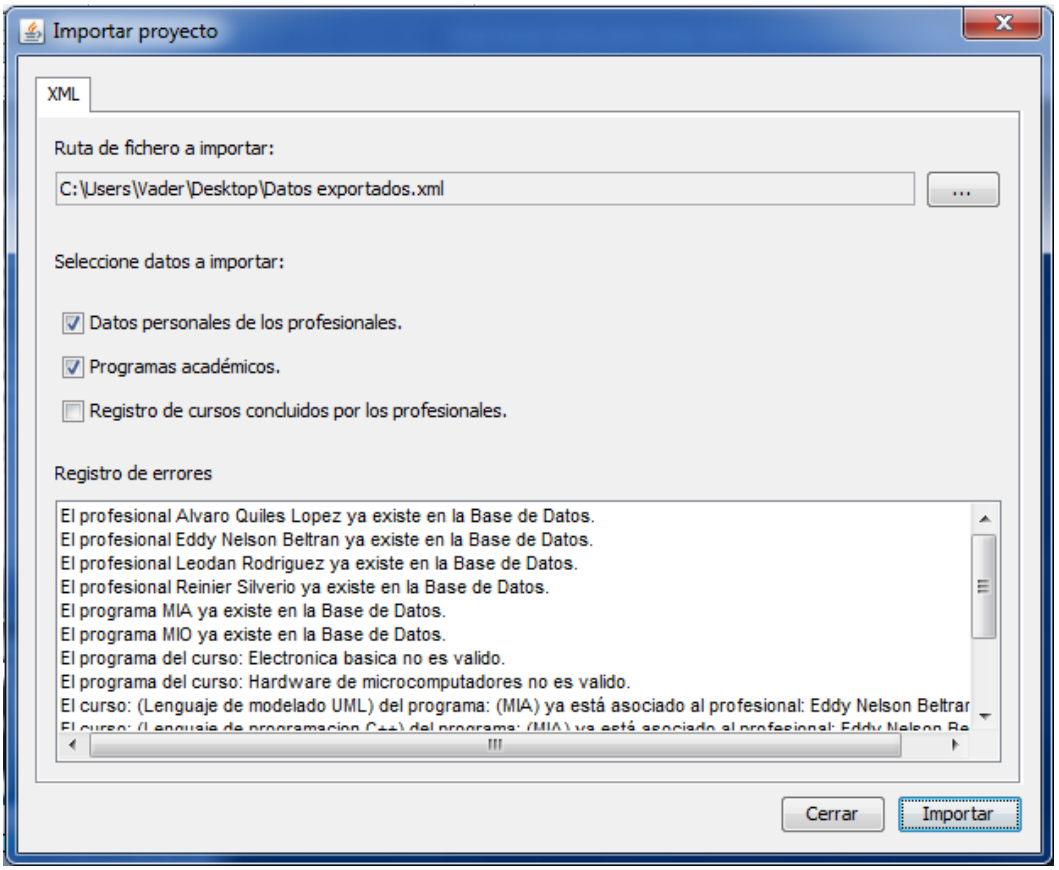
Nombre del profesional; Curso; Programa al que pertenece el curso;

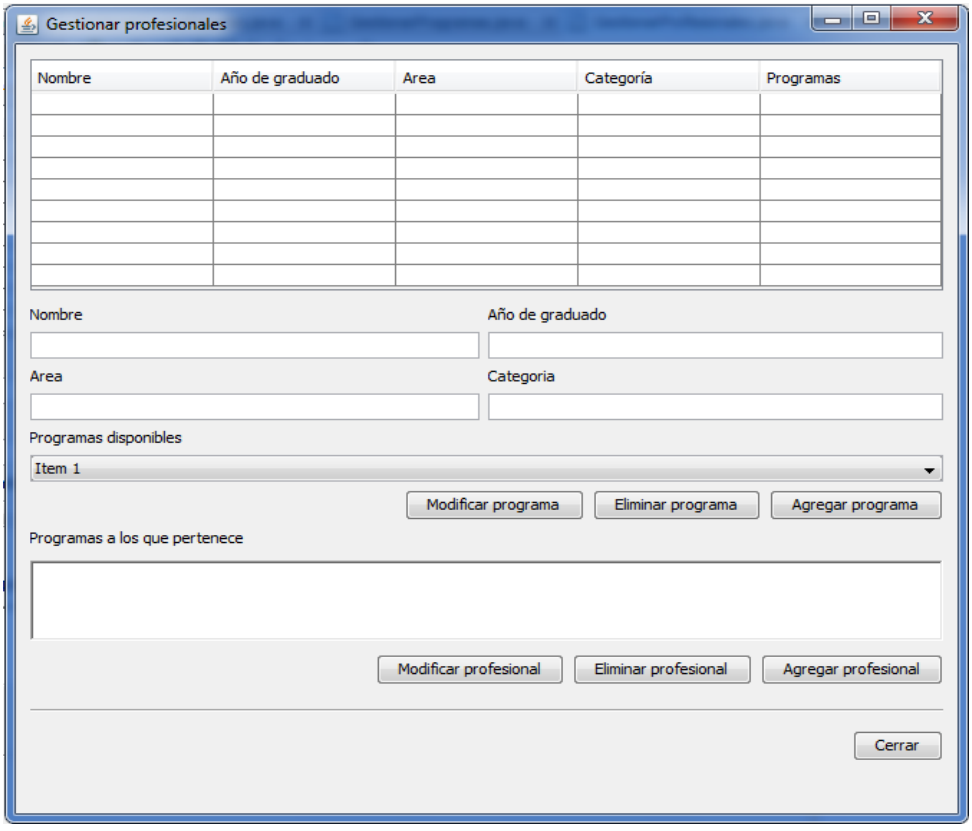
3) El fichero referente a los programas de superación existentes tendrá la siguiente estructura por la que se regirá.

1ra Fila: NombrePrograma; CantCreditosObligatorios; CantCreditosOptativos;</td></tr>
</table>

2da Fila en adelante: Nombre Curso; CantCreditosAcumula; Optativo o no(1/0)
Observaciones:
Prototipo de interfaz: 

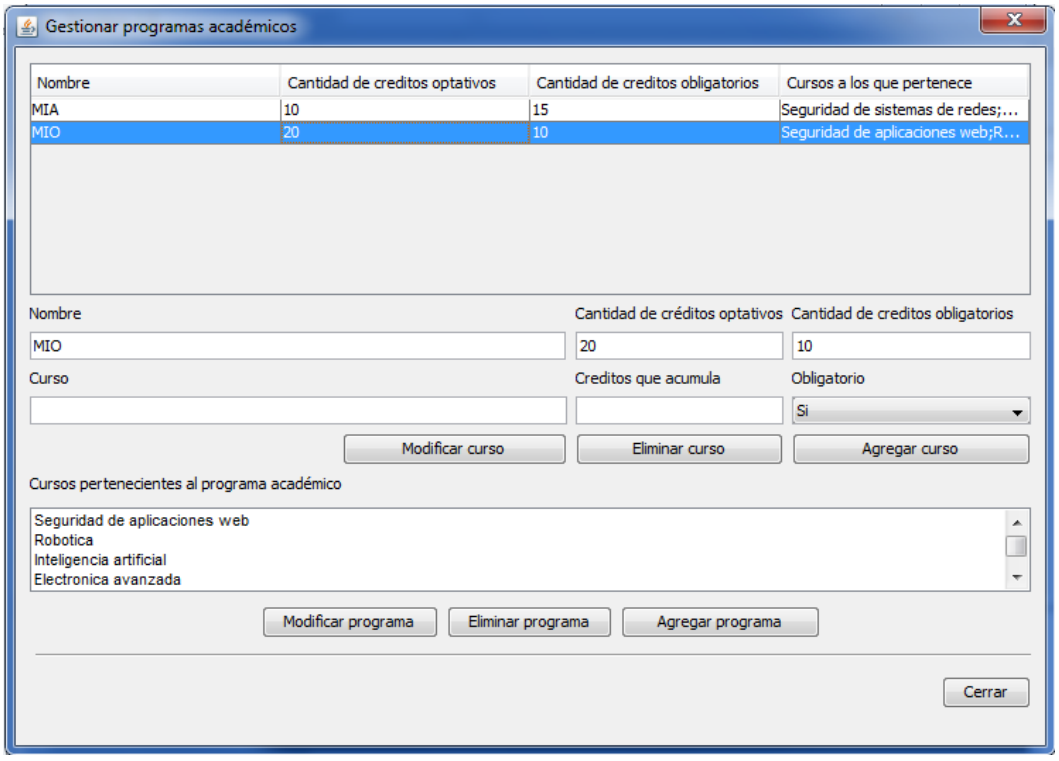
Historia de Usuario	
Código: HU_5	Nombre Historia de Usuario**: Exportar proyecto.**
Modificación de Historia de Usuario Número: Ninguna.	
Referencia: Se tiene como objetivo que el usuario exporte los datos que maneja la aplicación.	
Programador: Álvaro Quiles López.	**Iteración Asignada:** 2
Prioridad: Media.	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio.	**Puntos Reales:** 1
Descripción: El usuario procede a iniciar la generación de un empaquetamiento de los datos seleccionados para ser exportados en un fichero (*.xml). Tendrá las opciones de exportar los programas de superación, los datos personales de los profesionales y los cursos concluidos por los mismos. Teniendo la oportunidad de exportar cada una de estas opciones de manera individual o todos al unísono.	
Observaciones:	
Prototipo de interfaz: 	

Historia de Usuario	
Código: HU_6	Nombre Historia de Usuario**: Importar proyecto.**
Modificación de Historia de Usuario Número: Ninguna.	
Referencia: Se tiene como objetivo que el usuario importe los datos que maneja la aplicación desde un empaquetamiento en un fichero de extensión *.xml.	
Programador: Álvaro Quiles López.	**Iteración Asignada:** 2
Prioridad: Media.	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio.	**Puntos Reales:** 1
Descripción: El usuario procede a iniciar la carga de los datos seleccionados en un fichero (*.xml). Tendrá las opciones de importar parte o toda la información.	
Observaciones:	
Prototipo de interfaz: 	

Historia de Usuario	
Código: HU_7	Nombre Historia de Usuario**: Gestionar profesionales.**
Modificación de Historia de Usuario Número: Ninguna.	
Referencia: Se tiene como objetivo que el usuario pueda gestionar los datos personales de los profesionales que tiene almacenado el sistema.	
Programador: Álvaro Quiles López.	**Iteración Asignada:** 3
Prioridad: Media.	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio.	**Puntos Reales:** 1
Descripción: El usuario, tendrá la posibilidad de mostrar en pantalla un listado con la información de los profesionales que tiene almacenado el sistema, al mismo tiempo, tendrá la posibilidad de gestionarlos.	
Observaciones:	
Prototipo de interfaz: 	

Historia de Usuario

Código: HU_8	Nombre Historia de Usuario**: Gestionar programas.**
Modificación de Historia de Usuario Número: Ninguna.	
Referencia: Se tiene como objetivo que el usuario pueda gestionar los programas de la formación postgraduada que tiene almacenado el sistema.	
Programador: Álvaro Quiles López.	**Iteración Asignada:** 3
Prioridad: Alta.	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio.	**Puntos Reales:** 1
Descripción: El usuario tendrá en pantalla una tabla con la información de todos los programas de superación, al mismo tiempo, contará con opciones mediante las cuales agregará, eliminará o modificará información de los mismos.	
Observaciones:	
Prototipo de interfaz:	



Historia de Usuario	
Código: HU_9	Nombre Historia de Usuario: **Gestionar cursos pasados.**
Modificación de Historia de Usuario Número: Ninguna.	
Referencia: Se tiene como objetivo, que el usuario pueda gestionar los cursos concluidos por los profesionales.	
Programador: Álvaro Quiles López.	**Iteración Asignada:** 3
Prioridad: Media.	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio.	**Puntos Reales:** 1
Descripción: El usuario tendrá en pantalla un listado con la información de todos los cursos concluidos por los profesionales, de igual manera, contará con opciones mediante las cuales agregará, eliminará o modificará información de los mismos.	
Observaciones:	
Prototipo de interfaz: