"Arquitectura y operatoria de un sistema de corrección de exámenes automatizado, utilizando grafos dirigidos"

María Alejandra Paz Menvielle, Mario Alberto Groppo, Marcelo Martín Marciszack, Analía Guzmán, Karina Ligorria, Martín Cassatti
Departamento de Ingeniería en Sistemas de Información
Facultad Regional Córdoba – Universidad Tecnológica Nacional
Maestro Marcelo López esq. Cruz Roja Argentina – Córdoba
0351 – 4686385
pazmalejandra@gmail.com, proyale@groppo.com.ar, marciszack@gmail.com, , aguzman@sistemas.frc.utn.edu.ar, karinaligorria@hotmail.com, mcasatti@gmail.com

Resumen

El presente documento describe la arquitectura diseñada y la operatoria implementada para analizar las respuestas escritas por alumnos en forma de texto redactado en lenguaje natural, a preguntas de un examen.
Se muestran las técnicas que permitirán asignar valores a los conceptos y relaciones a fin de poder ponderar la respuesta suministrada por el alumno y compararla con la ponderación de la respuesta base elaborada por un docente.
Estas técnicas consideran todos los casos, incluyendo los distintos grados de acierto que pueda tener la respuesta del alumno, exponiendo los mecanismos con los que se deben analizar los conceptos y las relaciones para obtener la ponderación de la respuesta provista.

Palabras clave: *análisis de textos; grafos; detección de patrones; detección de rutas, arquitectura.*

1. Contexto

El presente trabajo forma parte del proyecto de investigación y desarrollo homologado por la Secretaría de Investigación, Desarrollo y Posgrado de la Universidad Tecnológica Nacional, en el ámbito de la Universidad Tecnológica Nacional.

Los contenidos que conforman el dominio de aplicación corresponden a los contenidos mínimos fijados para la asignatura Paradigmas de Programación, los cuales pertenecen al bloque de tecnologías básicas dentro del área programación, que están principalmente referidos a los paradigmas lógicos, funcional y de orientación a objetos.

2. Introducción

La implementación de un sistema de corrección automatizada de exámenes tiene, no solo una complejidad teórica importante, sino que también su implementación práctica es compleja, ya que los elementos constitutivos de dicho sistema cuentan con un conjunto de características particulares que no pueden ni deben dejarse de lado.

Una arquitectura correctamente desarrollada es el elemento fundamental mediante el cual es posible gestionar la complejidad del sistema mencionado y a la vez dejar previstos los fundamentos para su evolución futura.

La arquitectura y operatoria presentadas en este trabajo han sido diseñadas y están en proceso de implementación, con el objetivo de lograr la aplicación de las técnicas de grafos, la utilización de bases de datos y herramientas de software actualizadas, que facilitan y hacen posible conocer si las respuesta a las preguntas de examen son o no correctas a la vez que permite mantener actualizado el dominio de conocimiento de la materia, previendo además el crecimiento del sistema y su aplicación a otros ámbitos.

3. Arquitectura

Se ha optado por una arquitectura en tres capas horizontales (ver Figura 1) que modelan niveles crecientes de abstracción y que utilizan interfaces de software definidas para aislar los distintos módulos de efectos no deseados al realizar modificaciones.

El patrón de diseño utilizado se denomina MVC (Modelo/Vista/Controlador) y es adecuado para realizar la separación de responsabilidades y lograr una mayor flexibilidad a la hora de realizar modificaciones o mantenimiento del sistema.

El patrón MVC establece tres componentes fundamentales:

El Modelo: Es el encargado de gestionar el almacenamiento de la información y su accesibilidad para los componentes de capas superiores. En este caso está representado por el componente GraphDB y los diccionarios general y personalizado [4].

La Vista: Es el encargado de la presentación de la información a los usuarios y de la recepción de los comandos que se utilizan para interactuar con el sistema. Está representado por el componente de Aplicación y GraphView.

El Controlador: Es el contenedor de toda la lógica de negocios o también llamada lógica del dominio. Es el responsable de coordinar la interacción entre los demás módulos, recibir comandos de la Vista, gestionar los datos del Modelo y presentar los resultados, nuevamente en la Vista. Incluye además la lógica de validaciones y la funcionalidad específica del sistema. Está formado por los componentes ManejoLang, LangTool, ConceptManager, GraphAPI y GraphStream.

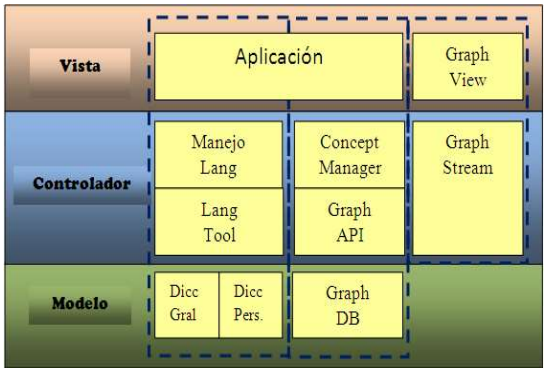

Figura 1: Arquitectura en tres capas.

En el nivel de Controlador se cuenta con dos niveles de abstracción diferentes según se trate de entidades de bajo nivel, como por ejemplo palabras en un diccionario o nodos y arcos o si se está trabajando con entidades de nivel superior, como términos del dominio, conceptos, relaciones, preguntas o respuestas.

Si bien la separación de responsabilidades se resuelve utilizando el patrón MVC existe una relación entre los componentes del sistema que permite agruparlos lógicamente a través de su funcionalidad, se puede apreciar en forma de grupos verticales en la vista de arquitectura (Figura 1).

De esta manera es posible relacionar la arquitectura con el diseño y definir el Módulo Ortografía (Language Tools y Manejo Lang), el Módulo Gestión de Conocimiento (GraphAPI y ConceptManager) y el Módulo Visualización (GraphStream y GraphView). Esta distribución brinda una gran flexibilidad en la implementación de las mejores soluciones en cada uno de los módulos componentes, manteniendo un bajo nivel de acoplamiento y minimizando los efectos laterales no deseados.

4. Entidades importantes en la arquitectura

El sistema de corrección automatizada define algunas entidades que son de uso general en todos los módulos involucrados y que sirven, además, para modelar toda la operatoria del mismo. Estas entidades son:

Término: Un término es el equivalente a una palabra en el texto de la respuesta, pero además contiene información de gestión asociada al modelo de dominio, por ejemplo si tiene equivalencias o qué tipo de entidad es en la base de conocimientos, ya sea Concepto o Relación, tanto simples como compuestas, las sugerencias de corrección en caso de que tenga errores ortográficos y la ubicación en que se encuentra en la oración, entre otros.

Un Término contiene, además, una representación de su información interna en formato XML para su intercambio con aplicaciones de terceros, en caso de que las mismas estén implementadas en tecnologías diferentes a la del sistema de corrección automatizado.

Todos los Términos están compuestos por una y sólo una palabra.

Concepto: Un concepto es una palabra o conjunto de palabras almacenado en la base de conocimiento y que tiene una equivalencia directa con uno o más conceptos de la materia Paradigmas de Programación.

Concepto Compuesto: Un concepto compuesto está formado por más de una palabra, o término. A los fines de la evaluación el Concepto Compuesto se trata como una unidad indivisible.

Relación: Es un arco, etiquetado, que une dos conceptos. El conjunto Concepto-Relación-Concepto representa la mínima cantidad de información textual que se debe analizar. Una relación tiene sentido únicamente entre dos Conceptos, aún cuando un Concepto tiene sentido por sí mismo, aunque no participe de ninguna relación [6].

Relación Compuesta: Una relación compuesta está formada por más de una palabra. Aún así se procesa como una unidad indivisible. La relación compuesta se utiliza de forma equivalente a la relación simple.

Estructura: Una estructura (ver Figura 2) es la mínima unidad que se debe procesar para una evaluación. Está compuesta de dos Conceptos unidos por medio de una Relación. Hay que mencionar que una Relación, Simple o Compuesta puede unir dos

Conceptos, Simples o Compuestos, en cualquier orden.

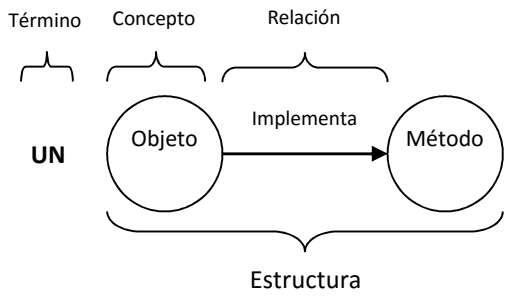

Figura 2: Estructura

5. Flujos de trabajo

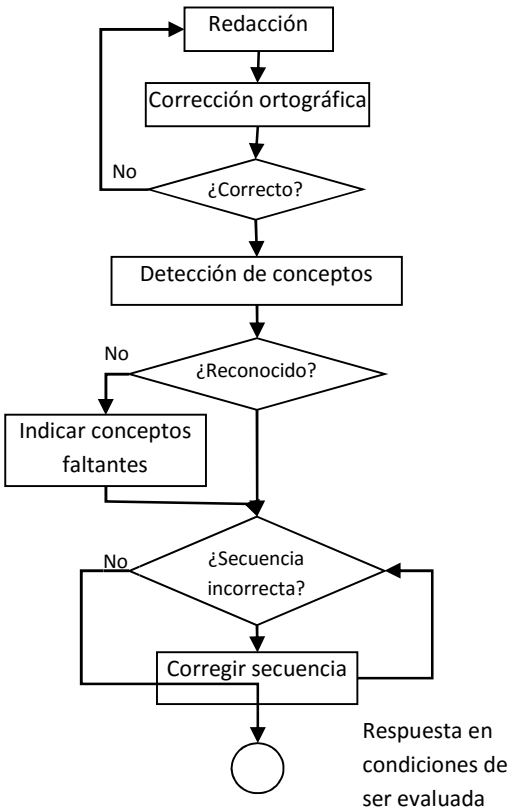

Figura 3: Flujos de trabajo.

Existen dos flujos de trabajo principales en el sistema de corrección automatizada que plantea el presente trabajo (ver Figura 3).

El primer flujo es el encargado de procesar una respuesta para asegurarse de que tiene la estructura correcta para ser procesada por el mecanismo de corrección.

Este flujo comienza con la redacción de la respuesta en formato libre, en idioma castellano, dando lugar a un texto que luego se envía al módulo de corrección ortográfica. En caso de que la corrección ortográfica indique que el texto contiene errores, los mismos se reportan y son corregidos por el usuario. El proceso se repite hasta completar la redacción de la respuesta. Una vez validada la ortografía se interactúa con el Módulo ConceptManager para la detección de los conceptos existentes en el texto. En este punto se pueden detectar qué conceptos de los vertidos en el texto de la respuesta se encuentran en la base de datos de conocimientos y cuáles no.

Si se trata de una respuesta ingresada por un docente, para los conceptos que el sistema detectó como faltantes, el docente tiene la posibilidad de realizar consultas sobre los conceptos existentes y sus equivalentes y/o darlos de alta. Si se trata de la respuesta de examen escrita por un estudiante, los conceptos faltantes serán puntuados con valor 0 (cero).

Para poder construir una ruta que permita evaluar la respuesta, los conceptos deben estar correctamente encadenados, siguiendo la secuencia Concepto>Relación>Concepto. Esto se analiza en el siguiente paso. En caso de detectar un mal encadenamiento se detiene el procesamiento hasta tanto todas las secuencias involucradas sean válidas.

Una vez finalizada esta etapa la respuesta está en condiciones de ser enviada al siguiente flujo: evaluación y calificación.

El segundo flujo de trabajo es el encargado de realizar la evaluación y la calificación de la respuesta del alumno (ver Figura 4), y se compone de los siguientes pasos:

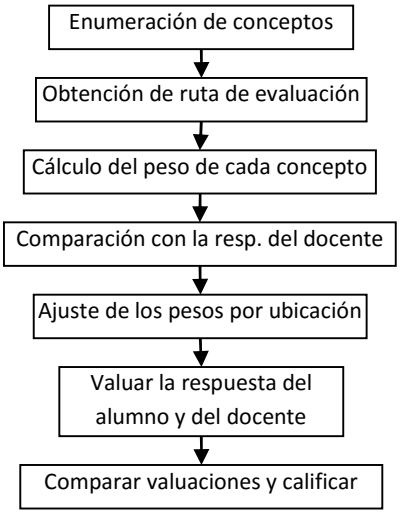

Figura 4: flujo de la respuesta del alumno

En el primer paso se realiza una enumeración de los conceptos contenidos en la respuesta del alumno. La lista de conceptos se envía en el paso siguiente, al módulo ConceptManager que es el encargado de determinar si existe alguna ruta de Conceptos y Relaciones que incluya los conceptos vertidos por el alumno en su respuesta.

Una vez obtenida la ruta, cada uno de los conceptos y relaciones se evalúa para obtener el peso de cada uno de ellos, teniendo en cuenta si es un Concepto o Relación exacta o si es una equivalencia, las cuales poseen menor peso [5].

A continuación, la respuesta del alumno se compara con la respuesta ofrecida por el docente, la que también se encuentra en la base de conocimientos, y se procede a realizar un análisis de la estructura de la respuesta. Esto sirve para ponderar los conceptos teniendo en cuenta el grado de semejanza entre la respuesta candidata (escrita por el alumno) y la respuesta base (dada por el docente), utilizando para ello la ubicación dentro del texto de los Conceptos y Relaciones. Aquellos Conceptos y/o Relaciones cuyas ubicaciones en la respuesta del alumno no coincidieran, ven sus pesos calculados de acuerdo a la distancia que hay entre su ubicación en la respuesta candidata con respecto a la respuesta base elaborada por el docente.

Una vez ajustados los pesos, se procede a calcular el valor de cada una de las respuestas (candidata y base). Estos valores se comparan entre sí y el resultado nos indica el grado de acercamiento de la respuesta candidata a la respuesta base, siendo este cociente una forma de calificación.

$$C_r = \frac{V_c}{V_b}$$

Siendo C_r la calificación relativa, V_b el valor de la respuesta base (del docente) y V_c el valor de la respuesta candidata (del alumno).

6. Componentes del sistema

El sistema se compone de cuatro bloques fundamentales que interactúan para obtener el resultado esperado, es decir, la evaluación de un examen redactado en texto libre.

Cada uno de estos bloques tiene responsabilidades bien definidas que posibilitan un alto grado de especialización y brindan la capacidad, existente en todo sistema de información que está modularizado, de reemplazar un módulo por otro implementado de manera diferente pero que cumpla con los mismos requisitos de funcionamiento.

Esto último es de fundamental importancia ya que el estado actual del proyecto, como investigación en curso, no está exento de cambios que requieran modificar la implementación o la tecnología subyacente, y frente a este posible escenario es necesario aislar las demás partes del sistema para protegerlas del impacto de estas posibles modificaciones.

Los módulos que componen el sistema son:

Módulo Gestión Ortográfica: Responsable de analizar la respuesta escrita e informar los errores ortográficos y sugerir las correcciones necesarias. Se utiliza una librería Open Source como base pero se implementan mecanismos que permiten que la misma reconozca terminología técnica específica de la materia Paradigmas de Programación y tenga en cuenta dicha terminología durante el análisis ortográfico.

Módulo Gestión de Conceptos: Administra la base de conocimientos y sirve de capa de abstracción a las librerías de gestión de la base de datos de grafos en la que se implementa la base de conocimientos. Utiliza como almacenamiento una base de datos de grafos sobre la cual se implementa la persistencia y los mecanismos de consulta necesarios para la evaluación.

Módulo GraphView: Es el módulo que permite visualizar un conjunto de conceptos y relaciones como un grafo dirigido. Utiliza una librería Open Source para realizar el gráfico propiamente dicho, pero implementa mecanismos propios para establecer el formato visual del gráfico y la forma de reflejar en el mismo las particularidades de la base de conocimiento de la materia y el proceso de corrección.

Módulo Aplicación: Este módulo ofrece un conjunto de servicios de alto nivel que actualmente se utilizan como una aplicación en sí misma, pero en futuros desarrollos va a incluir una API (Application Programming Interface, Interfaz de Programación de Aplicaciones) para el desarrollo de aplicaciones externas. Este esquema dual de trabajo busca proveer a los docentes con una herramienta de corrección intuitiva y de fácil uso pero además propone un esquema de SaaS (Software as a Service-Software como Servicio) en el cual la infraestructura se hace disponible a través de un conjunto de API's de alto nivel que posibilitan el desarrollo de aplicaciones cliente que utilicen los algoritmos y datos de la base de conocimientos. A continuación se describen las entidades que participan en la Arquitectura del sistema, luego se desarrollan con más detalle las funciones propias de cada módulo así como las interacciones con los demás componentes del sistema.

7. Módulo: Corrección ortográfica

Un pre-requisito importante para la evaluación automatizada de texto libre es que el mismo esté escrito correctamente según las reglas ortográficas y sintácticas del idioma, para evitar que los mecanismos automáticos malgasten tiempo y ciclos de cómputo intentando analizar términos que no tienen sentido según la lengua castellana.

El módulo ortográfico es el primero que interviene, precisamente para hacer esta evaluación.

El módulo utiliza una librería de análisis ortográfico implementada en Java, de código abierto, llamada LanguageTool, para hacer una revisión general del texto de la respuesta a corregir. Esta librería está extensivamente probada y avalada por la comunidad ya que se utiliza, entre otros proyectos de envergadura, como corrector ortográfico y gramatical asociado a los productos Open Source Mozilla Firefox y Libre Office.

Sobre esta librería se realizan, además, algunas adecuaciones que permiten utilizar terminología técnica concreta, propias del dominio del problema (la materia Paradigmas de Programación), que no forman parte de la lengua castellana pero que deben ser tomadas como correctas por el corrector ortográfico.

Esa terminología específica se almacena en un diccionario personalizado que ampliará el vocabulario del idioma español pre-existente con un conjunto de términos propios de la asignatura, el cual irá creciendo dinámicamente a medida que los docentes vayan incorporando nuevos términos mediante la creación de preguntas y respuestas de examen¨ [1].

El módulo de corrección ortográfica crea, a partir de las palabras que conforman el texto ingresado, una lista de términos.

Esta lista incluye todas las palabras detectadas sobre el texto libre que constituye la respuesta en el mismo orden en el que se las encontró en el texto.

El módulo, además de analizar el texto y construir la mencionada lista, procesa las palabras en busca de errores ortográficos. En caso de encontrarlos construye una lista de palabras sugeridas como corrección y la adjunta al Término erróneo en la lista. Se puede apreciar un ejemplo en la (ver Figura 5).

Un	ovjeto	contiene	atrivutos
	Objeto		atributos
	Objetó		

Figura 5: Lista de palabras sugeridas

8. Módulo: Gestión de Conceptos

El elemento fundamental para el correcto análisis y eventual corrección de una respuesta de examen es la posibilidad de poder detectar los conceptos y relaciones que componen una unidad de información, es decir, una respuesta.

La principal responsabilidad en cuanto al mantenimiento de la base de conocimientos y las consultas necesarias para analizar una respuesta recae en el módulo Gestor de Conceptos.

El mismo sirve como capa de abstracción para las librerías que permiten el acceso a la base de datos subyacente, que canalizan las consultas a la misma, y que además proveen mecanismos de más alto nivel para realizar operaciones sobre entidades de dominio.

El módulo Gestión de Conceptos implementa las siguientes funcionalidades:

Detección de Tipo de Término: El gestor de conceptos es el encargado de recibir la lista de Términos entregada por el módulo ortográfico y detectar qué tipo de entidad es la representada por cada uno de ellos y anexa dicha información a cada término de la lista de Términos. Esto se hace consultando la base de datos de grafo e indicando para cada Término si el mismo se trata de un Concepto (simple o compuesto) de una Relación (simple o compuesta) o si el Término no es reconocido como una entidad válida del dominio.

Consulta de Complejos: Identifica y marca los términos como Compuestos. Para ello se realiza una consulta a la base de datos que obtiene todos los conceptos y relaciones que están compuestos por más de una palabra. Una vez marcado un Término como Compuesto, ya sea Relación o Concepto, el mismo se trata como una unidad, indivisible.

Administración de Conceptos y Relaciones: El módulo de gestión de conceptos también es el responsable de permitir las operaciones básicas de administración de la base de conocimientos, tales como agregar, modificar y eliminar conceptos, agregar y modificar relaciones, y establecer y remover relaciones entre dos conceptos. El sistema implementa mecanismos de seguridad, por ejemplo en la eliminación de conceptos, que impiden eliminar un concepto que está siendo utilizado en una relación ya establecida.

Existencia de estructuras: Considerando que la estructura Concepto-Relación-Concepto es central para el análisis de las respuestas, el Gestor de Conceptos incorpora mecanismos que verifican la existencia o no de una estructura determinada. De esta manera, se le suministra al módulo un par de conceptos unidos por una relación y se consulta la base de conocimientos completa, informando al usuario si dicha estructura ya se encuentra. Hay que mencionar que el cambio en el orden de los conceptos modifica la estructura en sí, por lo que la consulta de C_1-R-C_2

puede, potencialmente, devolver valores distintos a la consulta de C_2-R-C_1.

Construcción y ejecución de consultas: El Gestor de Conceptos incorpora una funcionalidad para generar un "query" (consulta) que obtenga los resultados definidos por un conjunto de criterios, si la misma se ejecuta sobre la base de dato de grafos subyacente. Hasta el momento hay dos generadores implementados que se están utilizando en la aplicación. Consulta por conjunto de conceptos y consulta por concepto y profundidad.

La consulta por conjunto de conceptos (ver Figura 6) toma un concepto inicial C_i y un conjunto de conceptos relacionados C_{r1}, C_{r2},…, C_{rn} y obtiene una consulta que al ser ejecutada devuelve todas las rutas que, iniciando en el concepto C_i contiene alguno de los conceptos C_{r1}, C_{r2},…, C_{rn}

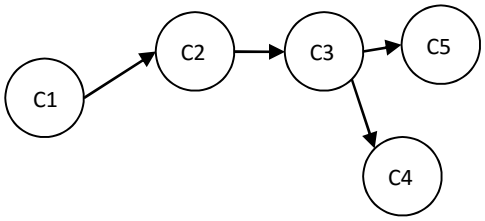

Figura 6: Consulta por concepto.

La consulta por concepto y profundidad (ver Figura 7) parte de un concepto C_i inicial y recorre todas las rutas que salen desde el mismo y las recorre hasta el nivel de profundidad *d* indicado [7].

En este segundo caso no es importante que las rutas sean aún más profundas que lo especificado, el proceso se detiene al alcanzar la profundidad solicitada.

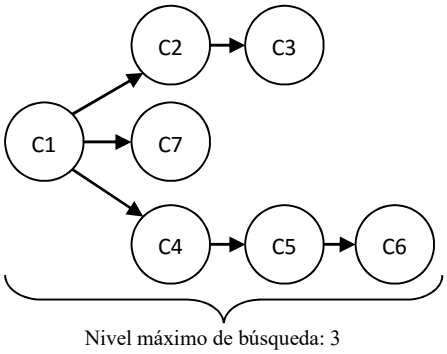

Figura 7: Consulta por concepto y profundidad.

Una vez generada la sintaxis de las consultas las mismas pueden ser ejecutadas sobre la base de datos de grafos.

Ejecución de consultas: Las consultas mencionadas anteriormente pueden ser ejecutadas sobre la base de datos de grafos, al igual que otras consultas más simples, utilizadas generalmente para operaciones de administración. Estas consultas una vez que se ejecutan devuelven una lista de conceptos y relaciones, denominados Vertex, los cuales conforman una ruta. Si uno recorre la lista de Vertex se obtiene la ruta que modela la consulta en la base de conocimientos. Hay que mencionar que el módulo encargado de la visualización de los grafos trabaja con listas de Vertex.

9. Módulo: GraphView

Éste módulo es el responsable de tomar una lista de nodos y arcos y visualizarla, incluyendo la información asociada a cada uno de los elementos tal como se obtiene de la base de conocimientos por medio del Gestor de Conceptos.

El módulo de visualización recibe de los demás módulos que lo utilizan, la información como una lista de elementos que contiene tanto Conceptos (C_1 a C_n) como Relaciones (R_1 a R_n) en el siguiente formato mostrado en la Figura 8.

C_1	$R_{1,2}$	C_2	$R_{2,3}$	…	C_{n-1}	$R_{n-1,n}$	C_n

Figura 8: Lista de conceptos y relaciones.

Y lo grafica como se puede apreciar en la Figura 9.

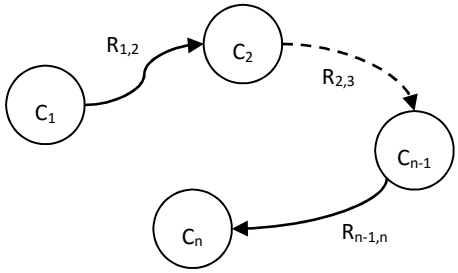

Figura 9: Gráfico de conceptos y relaciones.

El módulo utiliza una librería Open Source, implementada en Java, llamada GraphStream, que provee toda la funcionalidad de visualización y

trazado de rutas con una gran flexibilidad y facilidad de uso.

El módulo hace un uso intensivo de las propiedades de ruteo disponibles en la librería GraphView de forma tal que la visualización sea clara y con la menor cantidad de cruces de líneas entre los nodos.

Un detalle interesante es que el algoritmo de distribución (layout) de la librería GraphStream es un módulo reemplazable (plugin) por lo que se puede definir un algoritmo distinto, de acuerdo a las necesidades del proyecto, y reemplazarlo por el layout por defecto.

Se ha ampliado, en el presente proyecto, la funcionalidad de GraphStream de forma tal que las distintas consultas tengan estilos visuales claramente definidos y que sean adecuados a la visualización que en cada momento se requiera.

Actualmente se cuenta con tres visualizaciones principales:

Visualización de ruta: Se establecen los criterios necesarios, tal como se mencionó en el módulo Gestor de Conceptos y se ejecuta la consulta. El resultado se presenta como una ruta, cuyo origen se encuentra en el nodo C_i, el cual se resalta visualmente y se grafican las relaciones y conceptos relacionados, diferenciándolos del concepto inicial [3].

Visualización de concepto: Se toma un concepto inicial y una profundidad determinada y se ejecuta la consulta por concepto y profundidad [8]. Una vez obtenido el resultado, el nodo inicial se grafica y se resalta, utilizando propiedades de texto y color. El resto de los nodos, hasta el nivel de profundidad deseado d, se grafican utilizando un layout de tipo "estrella" (ver Figura 10) que facilita la visualización de los conceptos relacionados.

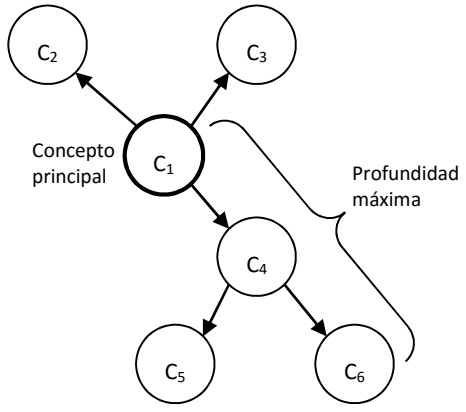

Figura 10: Grafico de nodos nivel de profundidad "d".

Visualización de relación: En este caso se ejecuta una consulta que obtiene todas las estructuras C_1-R-C_2, existentes en la base de conocimientos, que contengan la relación R especificada. En este caso no se obtiene un único grafo conexo sino un conjunto de grafos independientes, cada uno conteniendo R. El módulo de visualización grafica esta situación como varios grafos, de pequeño tamaño, y los distribuye de manera equidistante en el espacio de visualización.

10. Aplicación

La capa de aplicación hace uso de las funciones publicadas por las capas de niveles inferiores para unificar todo el proceso en una aplicación que interactúe con el usuario de manera integrada y uniforme.

La capa de aplicación es la responsable de presentar los datos al usuario de manera adecuada. En esta primera iteración la visualización se lleva a cabo a través de Forms SWI en Java. Aplicando el patrón de diseño MVC hemos logrado desacoplar la capa de aplicación para futuros cambios, como por ejemplo, el desarrollo de vistas web.

La capa de aplicación tiene las siguientes como responsabilidades centrales:

Administrar la base de conocimientos, permitiendo a los docentes incorporar nuevas preguntas que posteriormente pueden ser utilizadas para confeccionar un examen.

Adjuntar, a las preguntas mencionadas anteriormente, una o más respuestas, denominadas "respuestas ideales" o "respuestas base" que se tomarán como las respuestas que se asumen correctas al momento de la corrección de un examen.

Es de destacar que cada pregunta puede tener una o más respuestas correctas, redactadas de diferente manera y que involucren diferentes conceptos, a fin de ampliar la capacidad expresiva con que cuenta el alumno a la hora de responder.

El docente no debe registrar todas las respuestas posibles que pudiera dar un alumno. Cada respuesta ideal puede navegarse por diferentes caminos y los Términos que la conforman tener diferentes Términos equivalentes pudiendo de esta manera contemplar las diferentes respuestas del alumno. Caso contrario estaríamos frente a un sistema de "selección múltiple" que no es el objetivo del presente trabajo.

El módulo también tiene como responsabilidad analizar la base de conocimientos y contrastarla contra las respuestas base para determinar si todos los conceptos que intervienen en las mismas ya existen para, en caso contrario, agregar los nuevos conceptos y relaciones. Dicha información se la provee al módulo de gestión de conceptos.

De esta forma el proceso de redacción de respuestas permite no sólo validar la consistencia de la base de conocimientos sino que a la vez permite ir completándola, sin que sea necesario un mantenimiento formal de la misma.

10.1. Tecnología utilizada

La tecnología descripta en el presente trabajo tuvo un papel central en la elección de la arquitectura ya que presenta desafíos interesantes que involucran temas tan diversos como el almacenamiento de una base de conocimientos, la capacidad de consultar de forma rápida el contenido de la misma, la flexibilidad expresiva que esa consulta debe tener, la disponibilidad de herramientas de desarrollo, así como los conocimientos y experiencias de los miembros del equipo de investigación para proponer soluciones factibles y de calidad.

Además, se priorizó el uso de una tecnología que pudiera evolucionar en el tiempo y no quedar obsoleta rápidamente, con el riesgo de perder el conocimiento ya sistematizado almacenado en ella.

Se realizó la implementación en lenguaje Java ya que permite implementar la arquitectura multicapa propuesta y por la fácil intercambiabilidad de componentes que ofrece. A esto debe agregarse la perfecta integración que tiene con la base de datos de grafos, la librería de corrección ortográfica y la de visualización, todas desarrolladas en este mismo lenguaje.

La característica multi plataforma de Java no es secundaria en un proyecto de investigación relacionado con la educación universitaria, en donde es muy factible que distintas unidades académicas cuenten con distintas infraestructuras de hardware y software para implementar una solución de este tipo.

11. Conclusiones y trabajos futuros

El presente trabajo muestra la arquitectura de un sistema de corrección automática de examen, las preguntas son del tipo a desarrollar por el estudiante y escritas en lenguaje natural, de modo que es necesario analizar la ortografía y la redacción de la respuesta.

La arquitectura presentada implementa la utilización de modernas herramientas de bases de datos, como lo es OrientDB, y expone el análisis realizado y las investigaciones que se llevaron a cabo para su adaptación e implementación.

La arquitectura en tres capas permite trabajar con las distintas partes del sistema logrando la transferencia de información de manera ordenada y modularizada con vistas a futuras implementaciones y al crecimiento del sistema, ya que con otro tipo de arquitectura sería muy complejo implementar.

Como trabajos futuros y sobre el núcleo de la aplicación se está desarrollando una API REST que permitirá interactuar con otras aplicaciones, desarrolladas de manera externa.

Se ha elegido REST porque ha pasado a ser un estándar de facto en la comunicación entre aplicaciones en entornos diversos, tiene la ventaja de ser de fácil comprensión, de estar disponible en prácticamente todos los lenguajes de programación y de ser una tecnología probada y eficiente sobre todo en entornos de alta latencia como los existentes en aplicaciones Web, precisamente para los que fue diseñado REST.

Podemos concluir que con esta división en tres capas permitirá ofrecer un conjunto de “servicios” ligeros tanto para uso de la propia cátedra como de terceros, a la vez que se mantiene compacto y eficiente el núcleo que contiene la lógica de corrección y la base de conocimientos.

12. Referencias

[1] J Glenn Brookshear, Teoría de la Computación, Addison Wesley, 1989.

[2] Social networks sites adoption at firm level: a literatura review. J. Martins, R. Gonçalves, J. Pereira, T. Oliveira, M. Pérez Cota. Information Systems and technologies (CISTI), 2014 8th. Iberican conference.

[3] Sowa, John F, “Semantic networks" en Encyclopedia of Cognitive Science, 2006.

[4] Sowa, John F. "Conceptual graph summary", en Conceptual Structures: Current Research and Practice. Ellis Horwood, New York London Toront, 1992, pp. 3-66.

[5] Hopcroft, Motwani, Ullman. Teoría de autómatas, lenguajes y computación. Pearson, Addison-Wesley, 2008.

[6] Pavlidis, Theodosios, Structural pattern recognition. Vol. 2. New York: Springer-verlag, 1977.

[7] Olmos, Ivan, Jesus A. Gonzalez, and Mauricio Osorio. "Inexact Graph Matching: A Case of Study", en FLAIRS Conference, 2006.

[8] Buckley, Fred, and Frank Harary. Distance in graphs. Addison-Wesley Longman, 1990.