**Implementación Web de un sistema de corrección automatizada de exámenes escritos en forma textual**

Nicolás Horenstein, Karina Ligorria, María Alejandra Paz Menvielle, Martín Casatti, Analía Guzmán, Federico Benito

*Departamento de Ingeniería en Sistemas de Información*

*CIDS – Centro de Investigación Desarrollo y Transferencia*

*de Sistemas de Información*

*Facultad Regional Córdoba – Universidad Tecnológica Nacional*

*Maestro Marcelo López esq. Cruz Roja Argentina – Córdoba*

*0351 – 4686385*

{nicolashorenstein, karinaligorria, pazmalejandra, mcasatti, analia.casatti, fedebutn}@gmail.com

*Categoría del congreso: Tecnologías aplicadas a la educación (TAE)*

**Resumen**

*El presente trabajo describe la arquitectura propuesta para llevar a cabo la gestión académica a través de una interfaz Web (creación, asignación de instancias de evaluación) y los servicios implementados a través de una API para realizar la corrección automática de preguntas cuyas respuestas son escritas en texto libre. Además, se describe la operatoria del sistema, y las experiencias obtenidas en ámbitos reales, explicando los fundamentos y expresando hacía donde va a evolucionar el trabajo del equipo para continuar con el proyecto.*

***Palabras clave:*** *Arquitectura, API, Modelo Vista Controlador.*

**1. Identificación**

Código del PID: PIDEIUTNCO0004812.

Tema en que se inserta: Investigación aplicada en Sistemas de Información e informática, en el campo de aplicación del desarrollo de la educación y de la promoción

general del conocimiento.

Fecha de Inicio: 01/01/2018

Fecha de Fin: 31/12/2019

**2. Introducción**

El trabajo que aquí se presenta es la continuación del PID EIUTNCO0003592 “Metodología para determinar la exactitud de una respuesta, escrita en forma textual, a un interrogatorio sobre un tema específico”. El objetivo de dicho proyecto es implementar un sistema que permita la corrección automática de exámenes constituidos por preguntas cuyas respuestas sean escritas por los alumnos utilizando texto en formato libre, comúnmente conocidas como preguntas a desarrollar. Para llevar a cabo esta experiencia, se cuenta con la colaboración de la cátedra de Paradigmas de Programación, perteneciente a la carrera Ingeniería en Sistemas de Información, dictada en la Facultad Regional Córdoba, de la Universidad Tecnológica Nacional.

Para cumplir el objetivo planteado se modeló una base de conocimiento orientada a grafos, en la cual se mantiene toda la información relativa al contenido de la materia y de las respuestas sugeridas por los docentes a las preguntas de los exámenes.

Con el objeto de mejorar los instrumentos de evaluación, el proyecto actual propone la búsqueda y análisis de patrones[6] que permitan obtener métricas referidas a las evaluación de los contenidos de la materia, aprendizaje de los alumnos, etc.

Para ello se planteó la construcción de un segundo grafo, en el cual se modelan todas las respuestas realizadas por los alumnos a las preguntas de exámenes.

Contando con ambos grafos, se propone implementar los algoritmos que se estimen convenientes para obtener las métricas planteadas.

**3. Objetivos, Avances y Resultados**

**3.1. Objetivos**

El objetivo del PID es “Analizar y detectar patrones en un grafo conceptual construido a partir de respuestas escritas en forma textual a preguntas sobre un tema específico, utilizando una base de conocimientos diseñada como un grafo dirigido”. Para ello se han identificado los siguientes objetivos particulares:

1. Explorar patrones topológicos[7] de grafos que contengan información relevante para la

identificación de estructuras dentro de la base de conocimientos de la materia Paradigmas de Programación.

2. Analizar la existencia de patrones recurrentes o subyacentes en los grafos generados a partir de las respuestas base de los docentes y los obtenidos de las respuestas dadas por los alumnos.

3. Proponer algoritmos[8] que permitan detectar patrones conocidos en la teoría de grafos como son las “comunidades”, “pares”, “rutas principales”y otros patrones comunes, por medios del análisis de las métricas sobre la base de conocimiento.

4. Identificar características como exactitud,

coherencia y consistencia, entre otras, de las

respuestas escritas en forma textual, en la base de conocimiento diseñada como un grafo dirigido.

**3.2. Avances**

Con el objeto de obtener los datos requeridos para armar la base de conocimiento necesaria para la búsqueda y análisis de patrones, se evolucionó en la capa de aplicación inicialmente desarrollada a través de formularios Swing Java integrados en una aplicación con una arquitectura monolítica, a una interfaz Web. A través de la cual se realiza la gestión de alumnos, cursos, docentes, preguntas, permite la configuración y evaluación de instancias de evaluación y la visualización de notas e informes. Almacenando dicha información en una base de datos relacional.

Este cliente se comunica con el servidor de gestión de conocimiento a través de la API de procesamiento de respuestas, para obtener y enviar información desde y hacia la base de datos de grafos. Siendo el desarrollo de esta API el punto inicial de la evolución de la arquitectura del sistema Sibila a una arquitectura basada en microservicios.

**Arquitectura del sistema Sibila**

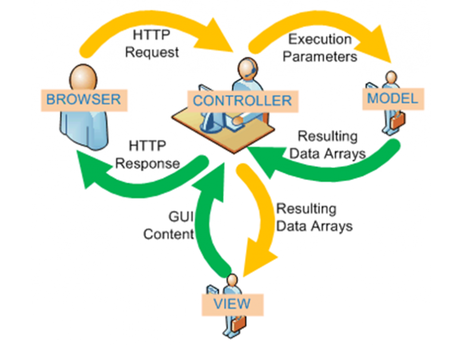
Cuando se hace referencia a la arquitectura de un sistema, es posible incurrir en varias definiciones, pero el trabajo se basa en la expuesta por Larry Bass (Bass et al. (1998)) donde expresa: “La arquitectura de software de un programa o sistema de computación es la estructura o estructuras del sistema, que comprende los componentes de software, las propiedades externamente visibles de esos componentes y las relaciones entre ellos” [2].

La arquitectura elegida para la confección de este sitio Web es conocida como MVC en donde se compone de 3 capas que se interrelacionan entre sí para dar respuesta a las solicitudes.

Estás capas son [3]:

* Modelo: contiene una representación de los datos que maneja el sistema, su lógica de negocio y sus mecanismos de persistencia.
* Vista: compone la información que se envía al cliente y los mecanismos de interacción con éste.
* Controlador: actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

En la siguiente figura se muestra el flujo de la información.

Figura 1: Flujo en la arquitectura MVC [3]

Tal como se observa en la imagen anterior podemos notar la relación entre las capas y el navegador web que hace una petición al sistema, por lo tanto, para poder terminar de entender el funcionamiento de esta arquitectura es posible detallar el flujo de acciones cuando se realiza una petición al sistema:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega.
3. El controlador accede al modelo, realizando una acción de consulta, actualización, modificación, etc. dependiendo de la acción solicitada por el usuario.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado alumnos por curso). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

El sistema web fue desarrollado exclusivamente con la tecnología de Microsoft .NET[1] utilizando el lenguaje de C#.

**Persistencia de la información**

Para la persistencia de los modelos se utilizó Entity Framework[4] el cual es un conjunto de API de acceso a datos para Microsoft .NET, que simplifica la forma de insertar, obtener, actualizar, etc. un modelo relacional.

Actualmente se está utilizando un motor de base datos relacional Microsoft SQL Server en su versión 2014.

No está de más aclarar que podría trabajarse de igual forma con versiones más antiguas o más modernas.

**Usuarios del sistema**

Todo usuario del sistema Sibila, puede llevar a cabo lo siguiente:

* Actualizar su perfil (información básica).
* Cambiar su contraseña de acceso.

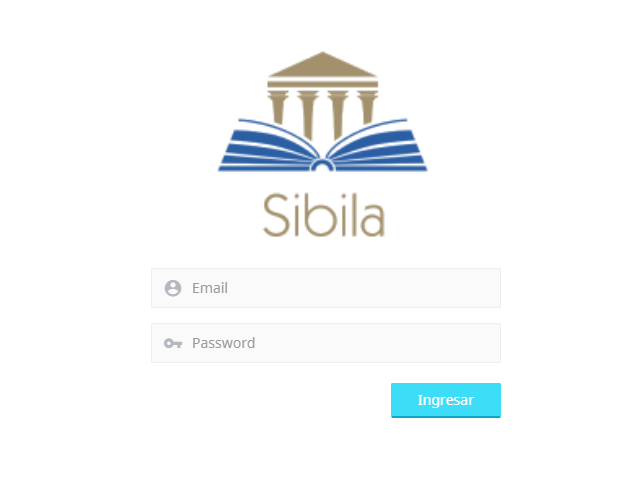


Figura 2: Página de acceso al sistema Sibila

Para la operatividad del sistema, encontramos 3 tipos de usuarios/roles:

* Alumno.
* Docente.
* Jefe de cátedra.

**Funcionalidades del sistema de acuerdo al rol asignado**

**Rol “Profesor”:** el docente podrá realizar las siguientes acciones:

* Observar a todos sus cursos asignados, tanto los que se encuentran activos como los finalizados.
* Obtener listado de alumnos por curso.
* Acceder al perfil de cada alumno.
* Importar listado de alumnos al curso.
* Crear instancias de evaluación.
* Crear preguntas con sus respectivas respuestas sugeridas.
* Activar una instancia de evaluación para ser rendido.
* Obtener listado de alumnos que se encuentran rindiendo el examen en tiempo real.
* Revisar y modificar la evaluación llevada a cabo por el sistema Sibila.
* Exportar exámenes rendidos por alumnos en formato .txt y .xslx.

**Rol “Jefe de cátedra**”: el coordinador de una cátedra podrá realizar las siguientes acciones:

* Mismas acciones que el rol “Profesor”.
* Obtener listado de cátedras en las cuales posee el rol de jefe.
* Administrar permisos por cátedras para los docentes de estas (permiso para crear examen, permiso para asignar examen).

**Rol “Alumno”:** el alumno podrá realizar las siguientes acciones dentro del sistema:

* Observar a todos sus cursos activos o finalizados.
* Consultar las instancias de evaluación rendidas.
* Acceder a su estado académico.
* Visualizar las instancias de evaluación activas para ser rendidas.
* Rendir.

**Operatoria del sistema al rendir una instancia de evaluación**

Los pasos para rendir una instancia de evaluación se pueden resumir en:

1. El docente crea un examen, con una cantidad definida de preguntas, las cuales van formando parte de una base de datos de preguntas que luego podrán ser reutilizadas en cualquier momento para otra instancia de evaluación de la cátedra.
2. Para poder asignar la evaluación el docente debe seleccionar el día y la hora de la evaluación y el o los cursos asociados a esa instancia de evaluación.
3. Los alumnos ingresan al sistema.
4. Una vez que el docente activa la instancia de evaluación los alumnos pueden empezar a rendir el examen, respondiendo a cada pregunta con texto libre.
5. Cuando el alumno lo considera, finalizar el examen.
6. Una vez que algún alumno haya finalizado el examen, éste podrá ser corregido por el sistema automáticamente.
7. El profesor puede escribir observaciones y modificar la puntuación sugerida por el sistema a las preguntas.
8. Cuando el docente le da el ok final a la calificación de una instancia de evaluación, habilita al alumno para ver su calificación y leer las observaciones específicas del profesor.

**Comunicación con API**

Una vez que el alumno rinde la instancia de evaluación determinada por el docente, el sistema lo corrige de forma automática. Para realizar dicha corrección el sitio web deberá conectarse con una API también desarrollada por este grupo de investigación.

Esta API está basada en una arquitectura de microservicios[9], la cual permite incorporar herramientas nuevas en cuanto a procesamiento de texto, trabajar con diferentes lenguajes de programación en una misma aplicación y utilizar la colaboración de otros servicios disponibles en la nube. Esta evolución asegura un sistema más flexible, sobre todo previendo su crecimiento y su aplicación en otros ámbitos.

El desarrollo de este servicio está realizado en JAVA, el cual recibe peticiones del tipo REST[10] (Representational State Transfer).

Entre los distintos servicios que ofrece es posible mencionar: corrección ortográfica (tanto de las preguntas, como las respuestas de los docentes y de los alumnos), equivalencias, corrección de respuestas, etc.

**3.3. Resultados**

A través del sitio web del sistema Sibila, se han realizado pruebas con los alumnos de la cátedra Paradigmas de Programación, en el ciclo lectivo 2018 y 2019. Desde el sitio web, las instancias de evaluación son configuradas por los docentes de la cátedra y los alumnos pueden responder las preguntas en formato de texto libre.

Esto ha permitido ejecutar y corroborar las fórmulas propuestas para la corrección de las instancias de evaluación, almacenar la información relativa a la gestión académica de alumnos y docentes en una base de datos relacional, y obtener los resultados de la evaluación a través de una base de conocimiento implementada en una base de datos orientada a grafos.

La ejecución de las pruebas a través del sitio web, dió como resultado una respuesta positiva por parte de los alumnos y los docentes, ya que, por el lado de los docentes, se pudo generar, asignar y exportar las instancias de evaluación de una forma muy intuitiva e importar al sistema los datos de los alumnos y, de parte de los alumnos, pudieron rendir sin ningún tipo de complicación.

Además, se logró obtener los datos iniciales para armar la base de conocimiento requerida para implementar los algoritmos de búsqueda y análisis de patrones.

Así también, se logró establecer una propuesta de rediseño global de la arquitectura monolítica del sistema Sibila a una arquitectura basada en servicios.

**4. Formación de Recursos Humanos**

El equipo de investigación está conformado por un director, un codirector, dos investigadores tesistas, un investigador de apoyo, un becario graduado y dos becarios alumnos

También participan del proyectos docentes que han obtenido su título de Especialista en Ingeniería en Sistemas de Información, y dos de ellos están realizando su tesis de maestría en el marco del presente proyecto.

Los becarios se incorporan con la finalidad de que inicien su formación en investigación

científica y tecnológica, quienes también colaborarán en la recolección, manipulación y desarrollo de este marco metodológico. En el marco del proyecto los estudiantes tendrán la posibilidad de hacer la Práctica Supervisada de quinto año. Los avances, propuestas y herramientas construidas, estarán disponibles para su transferencia y aplicación en el Centro de Investigaciones, Desarrollo y Transferencia de Sistemas de Información - CIDS. Del mismo modo la detección de patrones sobre el dominio de conocimiento de la materia Paradigmas de Programación continuará beneficiando a los integrantes de la cátedra y a los estudiantes.

**5. Publicaciones relacionadas con el PID**

- Patente: PID EIUTN3592. Título: “Metodología para determinar la exactitud de una respuesta escrita en forma textual sobre un tema específico, aplicando herramientas informáticas”. Género: Cientifico. Expediente Nro 5325962. Fecha: 14 Dic. 2014.

- María Alejandra Paz Menvielle y col. “Metodología para realizar evaluación de respuestas a preguntas escritas en formato de texto”. indexadas por ISI Thomson. 11ª Conferencia Ibérica de Sistemas y Tecnologías de la Información (CISTI 2016, España).

- María Alejandra Paz Menvielle y col. “Arquitectura y operatoria de un sistema de corrección de exámenes automatizado, utilizando grafos dirigidos”, en el 4to Congreso Nacional de Ingeniería Informática / Sistemas de Información (CoNaIISI 2016, Salta). ISSN 2347-0372.

- María Alejandra Paz Menvielle y col. “Detección de conceptos y relaciones para evaluación de respuestas”, en el 3er Congreso Nacional de Ingeniería Informática / Sistemas de Información (CoNaIISI 2015, Buenos Aires), ISBN: 978-987-1896-47-9. ISSN: 2346-9927 (en Línea).

- María Alejandra Paz Menvielle y col. “Metodología para determinar la exactitud de una respuesta, escrita en forma textual, a un interrogante sobre un tema específico, aplicando herramientas informáticas. Poster en WICC 2015 (Argentina). ISBN: 978-987-633-134-0.

- María Alejandra Paz Menvielle y col. “Metodología para determinar la exactitud de una respuesta, escrita en forma textual, a un interrogatorio sobre un tema específico”, en el 2do Congreso Nacional de Ingeniería Informática / Sistemas de Información (CoNaIISI 2014, San Luis). ISSN: 2346-9927.

**Referencias**

[1] Microsoft .NET. Última consulta: Julio 2019. <https://dotnet.microsoft.com/>

[2] Urciuolo Adriana y col. “Arquitectura de software para Sistemas de Información Ambiental”. Universidad Nacional de la Patagonia San Juan Bosco – Sede Ushuaia, 2018.<http://sedici.unlp.edu.ar/bitstream/handle/10915/21505/Documento_completo.pdf?sequence=1&isAllowed=y>

[3] “Modelo Vista Controlador (MVC)”. Servicio de informática ASP.Net MVC 3 Framework. Universidad de Alicante. Última consulta: Julio 2019.

<https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html>

[4] Entity Framework. Última consulta: Julio 2019.

<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview>

[5] Alonso Romero L. y Calonge Cano, T. “Redes neuronales y reconocimiento

de patrones”. (2001).

[6] María Alejandra Paz Menvielle y col. “Análisis de patrones en un grafo conceptual construido a partir de respuestas escritas en forma textual a preguntas sobre un tema específico”. 4º CADI y 10º CAEDI (2018, Córdoba).

[7] PAVLIDIS, Theodosios. Structural pattern recognition. (Springer, 2013).

[8] WATANABE, Satosi. Pattern recognition: human and mechanical. (John Wiley & Sons, Inc., 1985)

[9] LÓPEZ, Daniel, et al. Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web. 2017.

[10] Masse, Mark. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. " O'Reilly Media, Inc.", 2011.