Arquitectura basada en microservicios en un sistema de corrección automatizada de exámenes

Karina Ligorria, Analía Guzmán, Martín Casatti, Nicolás Horenstein, María Alejandra Paz Menvielle, Cynthia Corso

Facultad Regional Córdoba, Universidad Tecnológica Nacional, Maestro López y Cruz Roja Argentina s/n, Córdoba, Argentina.
{karinaligorria, analia.casatti, mcasatti, nicolashorenstein, pazmalejandra, corso.cynthia}@gmail.com

Resumen. El presente trabajo describe la arquitectura de un sistema de corrección automatizada de exámenes. Se toma como caso testigo los contenidos curriculares de la Cátedra Paradigmas de Programación de la Carrera Ingeniería en Sistemas de Información. Se comienza referenciando la arquitectura monolítica diseñada en 2015 cuando se comenzó este proyecto. Luego se describen los cambios realizados en ella para conformar una arquitectura basada en servicios, describiendo las distintas situaciones e inconvenientes que se presentaron y que condujeron al diseño e implementación de la nueva arquitectura.

Palabras clave: Arquitectura de software, arquitectura monolítica, arquitectura basada en microservicios, API REST

1 Contexto

Este trabajo se desarrolla en el marco del proyecto de investigación y desarrollo denominado "Análisis y detección de patrones en un grafo conceptual construido a partir de respuestas escritas en forma textual a preguntas sobre un tema específico", proyecto homologado por la Secretaría de Investigación, Desarrollo y Posgrado de la Universidad Tecnológica Nacional, reconocido con el código SIUTICO4812TC.

Este proyecto, actualmente en ejecución, continúa y amplía los trabajos realizados durante el desarrollo del Proyecto "Metodología para determinar la exactitud de una respuesta, escrita en forma textual, a un interrogatorio sobre un tema específico", (PID EIUTNCO0003592). Durante el transcurso del mismo se construyó el prototipo denominado Sibila 1.0 para realizar la corrección automatizada de los exámenes. Este prototipo accede a una base de conocimiento modelada como grafo dirigido. Inicialmente la base de conocimiento fue generada por los docentes de la cátedra Paradigmas de Programación, perteneciente al segundo año de la carrera de Ingeniería en Sistemas de Información, dictada en la Facultad Regional Córdoba, de la

Universidad Tecnológica Nacional. Cátedra que actúa como caso testigo de este proyecto y ha utilizado el prototipo para evaluar a los alumnos.

El proyecto actual tiene como objetivo analizar, detectar y evaluar patrones topológicos frecuentes en un grafo conceptual para determinar la exactitud de las respuestas, escritas en forma textual sobre un tema específico, utilizando una base de conocimiento diseñada como un grafo dirigido.

La arquitectura de Sibila 1.0 se diseñó e implementó inicialmente en forma monolítica, sin embargo con el paso del tiempo y con el objetivo del proyecto actual se tuvo la necesidad de actualizar el prototipo y por ende su arquitectura.

Así surge Sibila 2.0, que se modela y diseña con una arquitectura basada en microservicios.

El presente trabajo describe la evolución en el diseño de la arquitectura del sistema Sibila para adaptarse a los nuevos requerimientos y a los desafíos planteados por el equipo de trabajo.

2 Introducción

La Arquitectura de Software, es una disciplina de gran importancia dentro de la ingeniería de software. Se refiere a la estructura de un sistema compuesta de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos [1]. Los aportes más destacados de una arquitectura son la mejora en la compresión de los sistemas grandes y complejos, la mejora en las posibilidades de reutilización, la consideración de la evolución del sistema y al definirse principalmente a través de diagramas proporciona una forma de interpretar y documentar la estructura [2].

A continuación se describen las características principales de las arquitecturas monolítica y basada en microservicios.

2.1 Arquitectura monolítica

En una arquitectura monolítica “toda la lógica se ejecuta en un único servidor de aplicaciones” (Daya y Col., 2015). Todos los módulos corren en un único desplegable bajo la misma máquina virtual [3].

Si bien algunos enfoques plantean esta arquitectura implementando el patrón Modelo Vista Controlador (MVC), por lo general, se tiene como resultado una aplicación grande y compleja. Contar con una base grande de código [4] hace que sea un sistema difícil de mantener, para realizar algún cambio o mejora se debe determinar el impacto de dicho cambio para que no se vea afectada la ejecución del resto del sistema.

En la Figura 1 se representa una arquitectura monolítica.

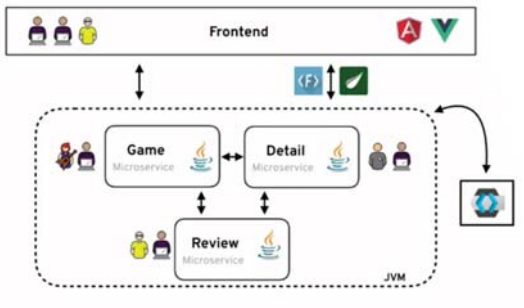

Figura 1: Arquitectura monolítica [3]

2.2 Arquitectura basada en microservicios

La arquitectura basada en microservicios es un estilo de arquitectura en donde la aplicación se encuentra constituida por un conjunto de microservicios o componentes, y se trata cada componente como si fuera una aplicación por sí misma [4, 5, 6]. La respuesta del sistema requiere la ejecución de los microservicios adecuados de acuerdo al proceso que se haya activado.

Algunas de las características de las aplicaciones basadas en microservicios que se pueden mencionar son [7, 8]:

- La funcionalidad del sistema es provista por un conjunto de servicios.
- Cada microservicio o servicio no debe cubrir una funcionalidad muy grande.
- Cada microservicio puede ser desarrollado en un lenguaje de programación diferente.
- Cada componente debe poder implementarse en forma independiente.
- La comunicación entre microservicios se realizará a través de métodos como REST (Representational State Transfer / transferencia de estado representacional) o similares [9].

En la Figura 2 se representa la arquitectura basada en microservicios:

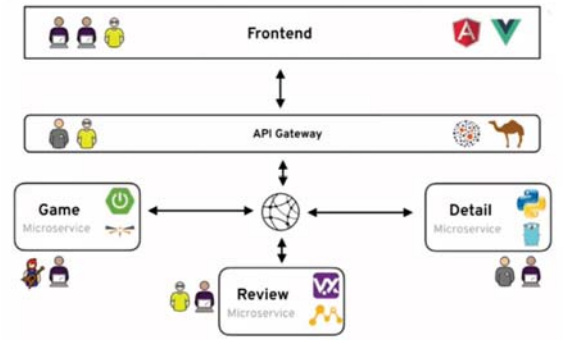

Figura 2: Arquitectura basada en microservicios [3]

A continuación se presentan algunas de las ventajas e inconvenientes que ofrecen las arquitecturas basadas en microservicios [8, 10]:

Beneficios:

- Componentes que pueden ser reutilizados en distintos escenarios.
- Mayor flexibilidad al ser más fácil reemplazar los microservicios.
- Componentes especializados en los niveles de usuario, modelo de datos y lógica de negocio.
- Mayor capacidad para adaptarse a cambios tecnológicos.
- Los equipos de desarrollo pueden trabajar en forma independiente, lo que propicia el trabajo en paralelo.
- Poca necesidad de coordinación entre los equipos, si se requiere un acuerdo de la información que espera y devuelve un servicio, y condiciones técnicas que se deben respetar.

Inconvenientes:

- Requiere de un buen diseño para poder determinar en todo momento qué componentes o microservicios participan en un proceso.
- Se puede ver afectado el rendimiento del sistema por la cantidad de intermediarios que se necesitan para ejecutar un proceso.

3 Rediseño de la arquitectura

3.1 Arquitectura Sibila 1.0

La arquitectura elegida inicialmente para Sibila 1.0, fue una arquitectura monolítica que se compone de tres capas horizontales (ver Figura 3) las cuales modelan niveles crecientes de abstracción y se utilizan interfaces de software para aislar los distintos módulos de efectos no deseados al realizar modificaciones [11].

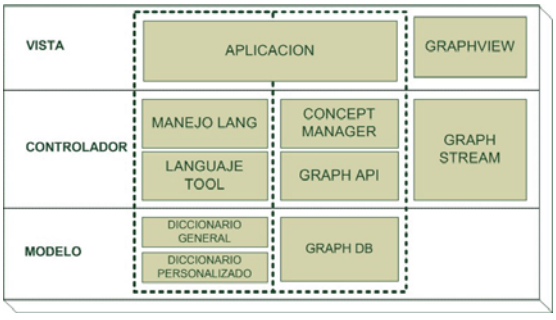

Figura 3: Arquitectura monolítica en tres capas del Sistema Sibila 1.0

En Sibila 1.0 cada módulo fue desarrollado en el mismo lenguaje de programación, utilizando una base de datos orientada a grafos para la evaluación de las respuestas obtenidas de las instancias de evaluación ejecutados por los alumnos de la cátedra de Paradigmas de Programación.

En la infraestructura planteada cada bloque 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.

La elección de una arquitectura monolítica ha posibilitado determinar fácil y rápidamente la factibilidad de representar un texto como un conjunto de conceptos y relaciones en forma de grafo, pudiendo tempranamente validar las fórmulas propuestas para determinar el grado de exactitud de una respuesta.

A continuación se realiza una breve descripción de la funcionalidad de cada módulo:

- **Módulo de corrección ortográfica**: Tiene la responsabilidad de analizar las respuestas escritas en forma de texto libre y detectar los errores ortográficos, informarlos y sugerir las correcciones necesarias.
- **Módulo de gestión de conceptos**: Tiene la responsabilidad de administrar 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 conocimiento.
- **Módulo GraphView**: Permite visualizar un conjunto de conceptos y relaciones como un grafo dirigido, información que recibe en forma de lista del Módulo de Conceptos. Utiliza una librería Open Source para realizar el gráfico propiamente dicho, implementa mecanismos propios para establecer el formato visual del gráfico.
- **Módulo aplicación**: Brinda la interfaz de iteración con el usuario, a través de Forms SWI en Java. Aplicando el patrón de diseño MVC, se plantea una capa de aplicación desacoplada e independiente, para futuros cambios, como por ejemplo, el desarrollo de vistas web.

3.2 Arquitectura Sibila 2.0

La evolución a una arquitectura basada en microservicios permite incorporar herramientas nuevas en cuanto a procesamiento de texto, utilizar 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.

Se propone para Sibila 2.0 una nueva estructura que cuenta con los siguientes componentes:

- Servidor de procesamiento de texto libre
- Servidor de gestión del conocimiento
- Servidor de patrones

- Cliente de gestión de evaluaciones
- Cliente de visualización de grafos

Cada componente se comunica con los demás a través de una API REST (Application Programming Interface / interfaz de programación de aplicaciones).

En la Figura 4 se muestran los componentes que intervienen en la nueva arquitectura propuesta.

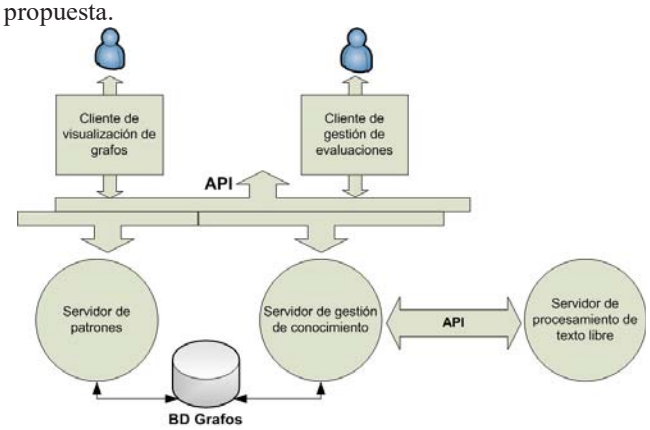

Figura 4: Arquitectura basada en microservicios de Sibila 2.0

4 Evolución de los componentes del sistema

Como parte de la evolución del sistema Sibila se planteó el reemplazo paulatino y gradual de los componentes monolíticos que componían el sistema en su versión 1.0 por un conjunto de servidores livianos de fines específicos que componen una arquitectura de microservicios.

La arquitectura basada en microservicios ofrece un conjunto de servicios de alto nivel, que se implementan a través de APIs que posibilitan el desarrollo de aplicaciones cliente que utilicen los algoritmos y datos de la base de conocimientos.

A continuación se describe la funcionalidad de cada componente:

4.1 Servidor de procesamiento de texto libre

Este servidor reemplaza al módulo de corrección ortográfica de Sibila 1.0. Se diseñó para poder realizar la validación ortográfica y el análisis del texto de tal forma que pueda enviar, al servicio que lo solicite, una cadena con el texto procesado.

Realiza dos operaciones básicas:

- Control de ortografía: analiza el texto para detectar errores de ortografía.
- Análisis de texto: identifica las entidades fundamentales para la construcción de la base de conocimiento del sistema.

4.2 Servidor de gestión de conocimiento

Este servidor tiene la misma funcionalidad que el módulo de gestión de conceptos del prototipo Sibila 1.0, su función es gestionar y consultar la base de conocimiento [11].

Algunas de las funcionalidades que implementa son la detección de tipo de término, la consulta de términos compuestos, la administración de conceptos y relaciones, la existencia de estructuras y, la generación y ejecución de consultas.

4.3 Servidor de patrones

Este servidor es el encargado de realizar la búsqueda de información en la base de datos de grafos, estás búsquedas serán realizadas a través de algoritmos que permitan detectar patrones por medios del análisis de las métricas sobre la base de conocimiento [12, 13]. Las métricas permiten analizar sobre qué componentes del grafo se realizan las mediciones, por ejemplo si son globales toman como referencia el grafo completo, con todos los nodos y arcos que lo conforman; si son locales toman como referencia un nodo o subconjunto de nodos para realizar los cálculos.

4.4 Cliente de gestión de evaluaciones

Este cliente reemplaza el módulo de aplicación de Sibila 1.0 y, a través de un sitio Web actúa de interfaz con el usuario. Realiza la gestión de alumnos, cursos, docentes, preguntas, permite la configuración y evaluación de instancias de evaluación y la visualización de notas e informes. Almacena 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.

4.5 Cliente de visualización de grafos

Este cliente reemplaza al módulo GraphView de Sibila 1.0 y permite la visualización de los datos de la base de datos que pueden ser enviados tanto desde el servidor de gestión de conocimiento como desde el servidor de patrones.

Esto implica que se podrá visualizar toda la información almacenada en la base de datos de grafos y además visualizar los resultados de las búsquedas de patrones definidos por los docentes para realizar su análisis.

5 Interfaz de comunicación entre los componentes

La comunicación entre los diferentes componentes de la arquitectura se logra a través de diferentes APIs, los cuales envían y reciben archivos JSON (Acrónimo de JavaScript Object). En este archivo la información se encuentra en un formato de texto que es completamente independiente del lenguaje de programación pero utiliza convenciones conocidos por los programadores de la familia de lenguajes C, C++, C#, Java, JavaScript, Python, y muchos otros [14].

Sibila 2.0 propone el uso de dos APIs para realizar la comunicación entre los servicios. Las APIs diseñadas son:

- API de interpretación de texto: Actúa de interfaz entre el servicio de procesamiento de texto libre y el servicio de gestión de conocimiento, la cual brinda una interfaz HTTP.
- API de procesamiento de respuestas: Actúa de interfaz entre el resto de los servicios de la arquitectura, es decir, el servidor de gestión de conocimiento, el servidor de patrones, el cliente de gestión de evaluaciones y el cliente de visualización de grafos.

6 Conclusiones y trabajos futuros

El presente trabajo muestra la nueva arquitectura propuesta para el sistema Sibila la cual consiste en una arquitectura basada en microservicios. A través de esta arquitectura se logra abstraer la complejidad técnica del sistema, logrando un sistema más flexible que permite la reutilización de los componentes como cajas negras, y es más sencillo utilizar soluciones open source disponibles en la nube.

Se propone seguir utilizando herramientas modernas de bases de datos, como lo es OrientDB, interfaz Web con el usuario e intercomunicación entre los diferentes componentes del sistema a través de API REST.

Se puede concluir que con esta nueva arquitectura basada en microservicios, el sistema ofrece 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, la base de conocimientos y la lógica de detección de patrones.

A través de esta arquitectura de software, se puede desarrollar cada microservicio con el lenguaje de programación más adecuado según la funcionalidad que debe ofrecer. Por otra parte, este enfoque facilita la convivencia de diferentes tipos de bases de datos, relacional y orientada a grafos, logrando representar o modelar los datos respetando sus características.

Como trabajo futuro se propone investigar nuevas herramientas que permitan la interpretación de texto libre escrito con mayor complejidad expresiva. Esto será posible ya que se cuenta con una arquitectura basada en microservicios para la que hay disponibles herramientas en lenguajes de programación más modernos facilitando el trabajo de base de este sistema.

También, se trabajará en la implementación de la lógica necesaria para el mantenimiento de la información para poner a prueba los algoritmos de detección y análisis de patrones sobre grafos dirigidos.

7 Referencias

1. L. Bass, P. Clements, R. Kazman. “Software Architecture in Practice, 2nd Edition”. Addison Wesley. 2003.

2. Fernández, L. F. “Arquitectura de Software”. Software Guru cono, 40-42. 2006

3. Soto, A. “Diferencia entre arquitectura monolítica y de microservicios”. https://openwebinars.net/blog/diferencia-entre-arquitectura-monolitica-y-microservicios/

4. Daya,S. y Col. “Microservices from Theory to Practice Creating Applications in IBM Bluemix Using the Microservices Approach”. 2015.

5. “DEVOPS: Arquitectura monolítica vs. microservicios”, 2016. https://www.chakray.com/es/devops-arquitectura-monolitica-vs-microservicios

6. López, D. et al. “Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web”. 2017.

7. Esaú, A. “Microservicios: Beneficios y funcionamiento”, 2016. https://openwebinars.net/blog/microservicios-beneficios-y-funcionamiento/

8. Wolff, E. “Microservices: Flexible Software Architecture”. 2017.

9. Masse, M. “REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. O'Reilly Media, Inc.". 2011.

10. Cortés, G. “IEDGE: Ventajas e inconvenientes de diferentes tipos de arquitecturas”, 2019. https://www.iedge.eu/gregorio-cortes-ventajas-e-inconvenientes-tipos-arquitecturas-de-sistemas

11. Paz Menvielle,A. y col. “Arquitectura y operatoria de un sistema de corrección de exámenes automatizado, utilizando grafos dirigidos”, en IV Congreso Nacional de Ingeniería Informática y Sistemas de Información, CONAIISI 2016, Universidad Católica de Salta, Facultad de Ingeniería, Argentina, 2016.

12. Pavlidis, Theodosios. “Structural pattern recognition”. Springer. 2013.

13. Watanabe, S. “Pattern recognition: human and mechanical”. John Wiley & Sons, Inc. 1985.

14. “Using JSON (JavaScript Object Notation) with Yahoo! Web Services”, 2019. https://web.archive.org/web/20100106010113/http://developer.yahoo.com/common/json.html

15. Sowa, J. “Conceptual graph summary, Conceptual Structures: Current Research and Practice”. Ellis Horwood, New York London Toront, pp. 3-66. 1992.