Arquitectura de microservicios para extracción de características en sistemas de recuperación de imágenes basada en contenido

Microservices architecture for feature extraction in content-based image retrieval systems

DOI: https://doi.org/10.17981/ingecuc.16.2.2020.15

Artículo de Investigación Científica. Fecha de Recepción: 28/07/2020 Fecha de Aceptación: 17/09/2020

Andrés Felipe Ruiz Velasco
Universidad del Cauca. Popayán (Colombia)
andres_ruiz@unicauca.edu.co

Sandra Milena Roa Martínez
Universidad del Cauca. Popayán (Colombia)
smroa@unicauca.edu.co

Para citar este artículo:
A. Ruiz Velasco & S. Roa Martínez, "Arquitectura de microservicios para extracción de características en sistemas de recuperación de imágenes basada en contenido", INGE CUC, vol. 16, no. 2, pp. 202–213, 2020. DOI: http://doi.org/10.17981/ingecuc.16.2.2020.15

Resumen

Introducción— Los sistemas de recuperación de imágenes basada en contenido permiten a los usuarios, por medio de una imagen de referencia, recuperar aquellas similares a su consulta. En la concepción de dichos sistemas para la Web, deben ser considerados aspectos relacionados al alto volumen de imágenes digitales existentes, que generan problemas durante su procesamiento en tiempo real, específicamente en la extracción de sus características visuales, objeto de esta investigación.

Objetivos— Contribuir en la mitigación de los problemas de escalabilidad, elasticidad, disponibilidad y confiabilidad presentada por el módulo de extracción de sus características visuales de un sistema de recuperación de imágenes basada en contenido.

Metodología— Se realizó la definición, diseño e implementación de una propuesta de arquitectura basada en microservicios y posteriormente la ejecución de pruebas mediante experimentos basados en simulación para la evaluación de dicha propuesta, presentando el respectivo análisis y discusión de los resultados entregados por el tablero de indicadores de la consola de Google Cloud.

Resultados— Una arquitectura basada en microservicios donde cada algoritmo/técnica de extracción de características de una imagen digital fue implementada como un microservicio bajo la infraestructura de Google Cloud.

Conclusiones— Esta propuesta arquitectural soportada en microservicios favorece su escalabilidad automática durante la extracción de características de grandes volúmenes de imágenes y puede ser usada en el diseño y construcción de otros módulos de un sistema de recuperación de imágenes basada en contenido.

Palabras clave— Arquitectura CBIR; microservicios; extracción de características; google cloud; recuperación de imágenes

Abstract

Introduction— Content-based image retrieval systems allow users, using a reference image, to retrieve those similar to their query. In the conception of such systems for the Web, aspects related to the high volume of existing digital images must be considered, which generate problems during their processing in real time, specifically in the extraction of their visual features, the object of this investigation.

Objectives— Contribute to the mitigation of scalability, elasticity, availability and reliability problems presented by the module for extracting its visual characteristics from a content-based image retrieval system.

Methodology— The definition, design and implementation of a proposal for architecture based on microservices was carried out, followed by the execution of tests using simulation-based experiments for the evaluation of said proposal, presenting the respective analysis and discussion of the results provided by the indicator panel of the Google Cloud console.

Results— A microservices-based architecture where each algorithm / technique for extracting features from a digital image was implemented as a microservice under the Google Cloud infrastructure.

Conclusions— This architectural proposal supported by microservices favors its automatic scalability during the extraction of features from large volumes of images and can be used in the design and construction of other modules of a content-based image retrieval system.

Keywords— CBIR architecture; microservices; feature extraction; google cloud; image retrieval

I. Introducción

Los recursos audiovisuales se han incrementado considerablemente en los últimos años, en parte gracias a la facilidad con la que se genera y consume información en Internet, este crecimiento requiere cada vez más de soluciones computacionales que se adapten a la gran cantidad de información que se transmite a diario por la red. Por ello, estas soluciones deben procesar gran cantidad de información minimizando problemas de: 1) escalabilidad, la cual le permite al sistema incrementar su capacidad de procesamiento usando recursos adicionales [1], 2) elasticidad (capacidad del sistema para adicionar o remover recursos de acuerdo a la demanda de trabajo que requiera [1]), 3) disponibilidad, la cual se asocia al porcentaje de tiempo en el cual un sistema es accesible [2]) y 4) confiabilidad (capacidad del sistema de ejecutar sus funciones bajo ciertas condiciones en un período de tiempo [3]), sin desconocer la mejora en los tiempos de ejecución.

Una de las operaciones más comunes en procesamiento de contenido audiovisual, específicamente con imágenes, es la consulta a partir de un ejemplo (imagen de referencia), en la cual mediante un buscador se ingresa una imagen o vídeo y el sistema recupera los contenidos más similares con respecto a la imagen de entrada de referencia. Para esto, puede ser usado un sistema de recuperación de imágenes basado en contenido (*Content-Based Image Retrieval,* CBIR), que está compuesto por un módulo de extracción de características, objeto de estudio en este trabajo, el cual recibe imágenes y mediante algoritmos de procesamiento de imágenes extrae un conjunto de características/patrones previamente definidas, que son estructuradas en un vector de características y posteriormente se almacenan en una base de datos asociando cada uno de estos vectores a una imagen, en este módulo también se extraen las características de una determinada imagen que reciba el sistema CBIR durante una consulta y se genera igualmente un vector de características de la misma. El criterio que indica si una imagen es similar o no a otra está dado por la distancia entre sus arreglos de características, conocida como métrica de similaridad, la cual en el caso de arreglos idénticos permite afirmar que corresponden a la misma imagen, y su distancia será cero, asimismo, entre más cercano sea este valor a uno, indicará que la imagen es menos similar a la que se desea recuperar [4].

En cuanto a un sistema CBIR web, este puede recibir muchas imágenes de entrada de manera simultánea de diferentes fuentes, por ejemplo puede recibir imágenes obtenidas en tiempo real mediante un dispositivo de captura [5] o mediante imágenes cargadas por usuarios manualmente en una interfaz gráfica [6], las cuales podrían ser cargadas por lotes, dado esto, en el módulo de extracción de características se deben ejecutar varios algoritmos para extraer patrones sobre cada una de estas imágenes, por ello los algoritmos de extracción dependen de los recursos computacionales del equipo para ejecutarse correctamente, así como del tamaño o complejidad de la imagen a procesar. Si la ejecución de la extracción de características es “lenta” o consume gran parte de los recursos del sistema, al procesar muchas imágenes o peticiones en simultánea se pueden generar bloqueos, caídas o pérdida de información en el sistema.

Por lo tanto, la extracción de características es una de las etapas que requiere mayor capacidad de procesamiento, asociado a la complejidad y costo computacional de los diversos algoritmos que pueden ser utilizados para este proceso [7]-[8], de tal modo que si este módulo debe obtener un alto número de características a extraer, el proceso consumirá un mayor tiempo dependiendo de la complejidad de cada uno de estos algoritmos, y en caso de no encontrarse optimizados para ejecutarse de forma paralela, entonces todos podrán procesarse sobre un mismo ambiente de cómputo, lo que conllevará a problemas de rendimiento al tener varios de estos procesos en espera de ejecución.

Sumada a la extracción, el procesamiento de una gran cantidad de imágenes se ve afectada si se usa una base de datos relacional para el almacenamiento de los vectores de características, pues el número de transacciones en el sistema al generarse una alta cantidad de información también puede generar bloqueos sobre la base de datos si no se administra la concurrencia correctamente [9]. Adicionalmente, si la aplicación usa un disco duro para almacenar las imágenes, la velocidad de escritura y lectura sobre el disco se ve afectada ante una alta cantidad de peticiones sobre el servidor realizando operaciones sobre el mismo, a esto se suma que es necesaria una gestión de infraestructura para el mantenimiento de dicho almacenamiento (discos duros).

A partir de las consideraciones descritas, se tornan múltiples desafíos relacionados con el diseño de una arquitectura de un CBIR en un entorno Web que sea capaz de soportar un alto flujo de información. Para ello, en este trabajo se presenta una arquitectura de microservicios que pretende mitigar problemas de rendimiento que puede presentar un módulo de extracción de características frente a la recepción simultanea de un alto volumen de información, esta arquitectura contempla un enfoque no convencional para este módulo de procesamiento y extracción de características en paralelo, dado que usa los algoritmos de extracción de características como microservicios y se beneficia de la infraestructura de servicios de *Google Cloud*, contribuyendo así en la creación de sistemas CBIR en los que se puedan definir módulos independientes que usen la misma arquitectura basada en microservicios. Vale destacar que, la propuesta arquitectural de este módulo, se integra a una arquitectura general de tipo monolito usada generalmente en este tipo de sistemas CBIR.

A continuación, serán presentados elementos conceptuales y técnicos que fueron usados para la construcción de la propuesta, seguidamente es descrita en la sección tres, la arquitectura propuesta como principal resultado con consideraciones de definición, diseño e implementación, y posteriormente la evaluación y discusión de los resultados y pruebas realizadas, finalizando con las conclusiones y referencias generadas en la ejecución de este trabajo.

II. Marco teórico

A. *Extracción de características con arquitectura de monolito*

El módulo de extracción de características en un sistema CBIR consiste en recibir una imagen de entrada, almacenar esa imagen, calcular sus características mediante algoritmos de procesamiento de imágenes, obtener un vector de características, almacenar ese vector y asociarlo a la imagen original que se ingresó al sistema.

Un monolito puede ser definido como una arquitectura de aplicación que se desarrolla sobre una misma base de código [10] y que realiza cada parte del proceso en un mismo programa y se soporta en su mayoría sobre la misma tecnología.

Una arquitectura monolítica para un sistema CBIR no está orientada exclusivamente a una aplicación de escritorio, por ejemplo, en [11] se presenta una arquitectura basada en inyección de dependencias para un CBIR que se ejecuta sobre una sola aplicación web [12]. Entre las desventajas de este tipo de arquitectura se encuentra que, si uno de los módulos del CBIR requiere procesar una gran cantidad de imágenes al tiempo, es probable que ese módulo presente problemas de rendimiento debido a que depende de la plataforma sobre la cual el módulo se esté ejecutando, si esta plataforma no puede escalar a medida que la demanda de operaciones crece, entre más componentes tenga el sistema, más recursos necesita para funcionar correctamente. Además, se debe garantizar que las operaciones de escritura y lectura de datos concurrentes que se ejecuten sobre la base de datos, se realicen de manera correcta, y para garantizar estos aspectos sobre un proceso que depende de un solo servidor, se tienen que considerar los altos costos operativos y financieros.

B. *Arquitectura de Servicios*

Se encuentra en la literatura, que el módulo de extracción de características usando tecnologías Web, realiza la separación de los elementos que componen el módulo en servicios más pequeños que se comunican mediante colas de mensajes o llamados directos.

Vale destacar que, algunas propuestas de CBIR orientadas a arquitectura de servicios presentadas usan servicios en la nube de Azure (Servicio de computación en la nube de Microsoft) para conexiones con nubes privadas [13]. Adicionalmente, otras presentan una arquitectura de este tipo para aplicaciones móviles usando servicios distribuidos [14].

Una vez analizadas, se encuentra que, en este tipo de arquitecturas, cuando hay múltiples peticiones actuando simultáneamente en el sistema, escalar y ejecutar un servicio en paralelo, por ejemplo, el de extracción de características, es posible porque cada servicio puede escalar automáticamente sin afectar a los demás servicios. Sin embargo, si el flujo de peticiones sobre el sistema es muy alto o crece repentinamente, o si las características que se van a extraer son numerosas, este escalamiento puede no ser suficiente al requerirse muchos recursos para

extraer todas estas características, lo que puede ocasionar que, el servicio consuma y bloquee múltiples recursos mientras se ejecutan varios algoritmos para una sola imagen sobre el mismo bloque de código, adicionalmente si estos algoritmos consumen mucho tiempo en finalizar, entonces el escalamiento debe ser muy alto para que el sistema funcione correctamente, todo esto generará mayores costos de infraestructura.

C. *Arquitectura de Microservicios y Google Cloud*

Es un esquema arquitectónico para desarrollar una aplicación web donde cada funcionalidad de la aplicación se diseña como pequeños servicios [15], a nivel de código, de componentes o funcionalidad, conocidos como microservicios. Estos se ejecutan individualmente y de manera independiente del resto de microservicios, de tal modo que su despliegue y mantenimiento no debe afectar a los demás. Esta independencia permite que los diferentes microservicios en una aplicación puedan ser implementados en diferentes tecnologías y que aun así puedan mantener comunicación entre sí mediante eventos o peticiones. También al ser pequeños requieren baja capacidad de cómputo para ejecutarse, y si la necesitan no será tan alta como si se ejecutaran todos los servicios en un esquema monolítico [16].

Por otra parte, las funciones en la nube - *Google Cloud Functions* son un entorno de ejecución de aplicaciones sin servidores y se utilizan principalmente para crear los microservicios que se conectan a otros servicios en la nube. Entre las principales ventajas de las funciones en la nube se encuentra que: 1) Se escalan automáticamente en función del número de peticiones que debe atender; 2) tienen una gran tolerancia a fallos; 3) no requieren mantenimiento de infraestructura; 4) su costo depende de su uso y; 5) se pueden implementar en diferentes lenguajes de programación [17]. Las funciones en la nube se ejecutan mediante eventos específicos, los cuales son recibidos por la función que ejecuta su función específica o hace un llamado a otros servicios y devuelve una respuesta. En la Fig. 1 se presenta el flujo de un evento de llamado a una función. Entre los eventos más usados para invocar funciones en la nube se encuentran: Llamado HTTP directo, la carga a un almacenamiento de archivos y la carga a un servicio de colas de mensajes. El uso de cada uno de estos llamados depende de la funcionalidad a implementar. Los llamados directos generalmente se usan para operaciones de respuesta inmediata. Las cargas de archivos para operaciones que funcionan en segundo plano, y la carga a colas de mensajes cuando las operaciones no se pueden hacer asíncronas entre cada componente del sistema.

Fig. 1. Flujo de eventos de llamado a una función en la nube.
Fuente Google Cloud [18].

En cuanto al almacenamiento en la Nube - *Google Cloud Storage* es un servicio de almacenamiento de archivos escalable y duradero, que reduce la dependencia y riesgos adyacentes al uso de discos duros y servidores para almacenamiento de la información, este servicio funciona mediante contenedores que almacenan la información [19] y este servicio puede llamar un evento al ejecutarse una carga de tal modo que dispare una función que realice un procesamiento sobre la información que se acaba de subir al contenedor.

Adicionalmente, se encuentra el servicio de colas de mensajes - *Google Pub/Sub*, en el cual se pueden publicar mensajes desde otro servicio para que sean consumidos a su vez por otros servicios [20]. Este consumo se realiza mediante una suscripción que se dispara cuando un mensaje entra a la cola de mensajes. Todos los servicios suscritos a la cola reciben el mensaje y lo consumen para utilizarlo. Por lo tanto, es un servicio adecuado para realizar la comunicación entre microservicios [20].

Como base de datos, Google Cloud cuenta con *NoSQL - Google DataStore*, que contiene documentos con escalabilidad automática sin el uso de servidores físicos [21], lo cual se torna ideal para sistemas que no requieran procesamiento de transacciones en línea, para sistemas con datos no estructurados y su alta disponibilidad de operaciones de escritura y lectura permite que se puedan modificar datos en tiempo real sin afectar el rendimiento o resultado de dichas operaciones.

III. Propuesta de arquitectura basada en microservicios para extracción de características en un sistema CBIR

A. *Definición y diseño de la arquitectura propuesta de extracción de características*

Este trabajo plantea usar la arquitectura de servicios propuesta por diferentes autores y extenderla para implementar cada componente del módulo de extracción como un microservicio, donde a diferencia de dichas propuestas, cada algoritmo de extracción de características actuará como un microservicio y no como parte del mismo bloque de código para la extracción de característica de cada imagen en el módulo de extracción de características. De modo que, los algoritmos más costosos computacionalmente no afecten la ejecución de los demás, y se puedan escalar automáticamente a medida que requieran más recurso computacional.

Como puede observarse en la Fig. 2, donde se presenta el esquema de la arquitectura propuesta, cada elemento del sistema se despliega como un microservicio y la comunicación entre ellos se basa principalmente en eventos, con esto se logra comunicación asíncrona principalmente, de forma tal que se previenen bloqueos en el sistema por espera de respuesta.

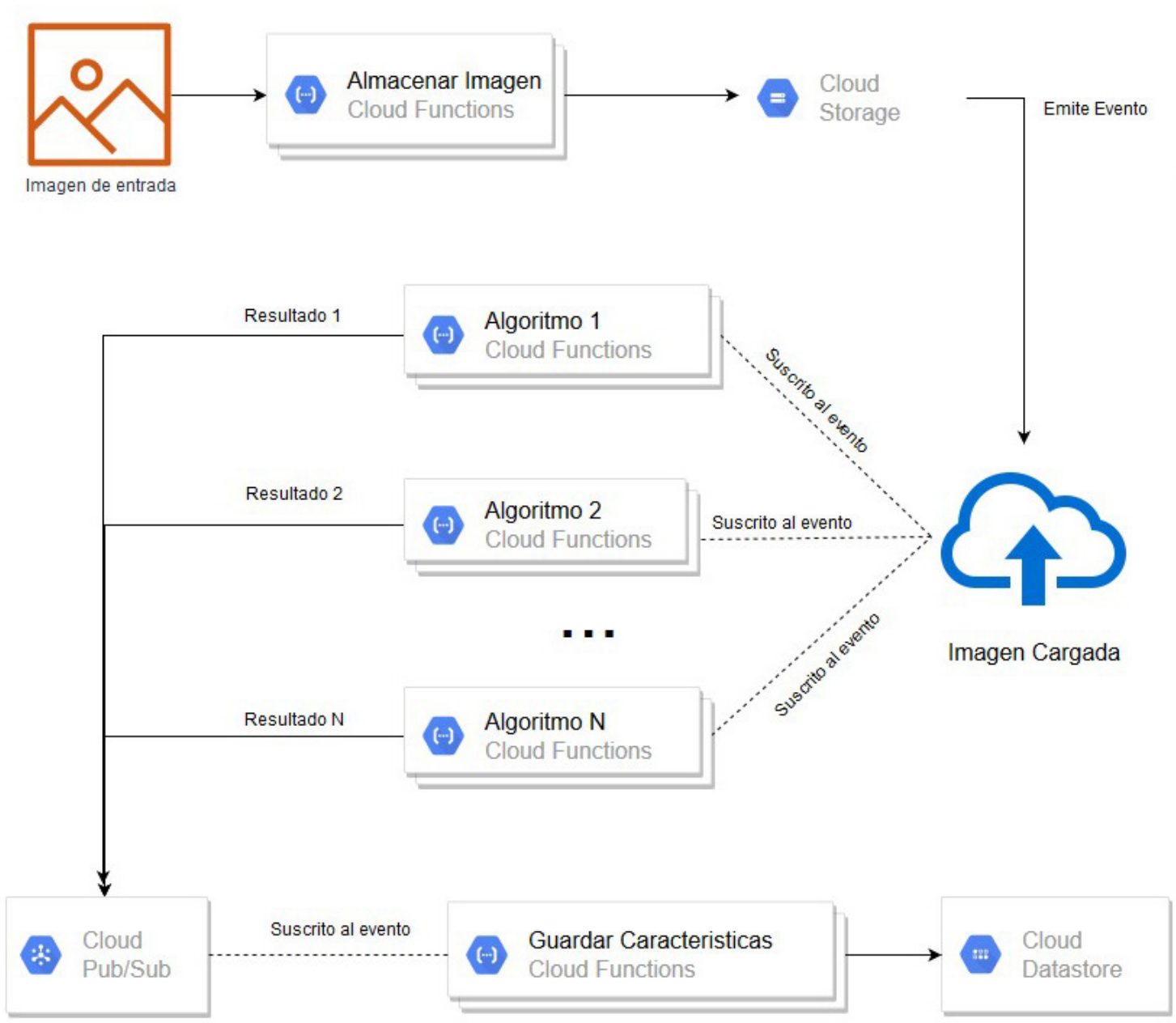

Fig. 2. Arquitectura de microservicios propuesta para el módulo de extracción de características.
Fuente Autor(es)

En la misma Fig. 2, el sistema tiene como entrada una imagen, la cual mediante un microservicio se almacena en el *Cloud Storage* usando el API de *Google Cloud*. Cuando la imagen se termina de cargar, el *Cloud Storage* emite un evento informando que se acaba de subir una imagen nueva, se debe resaltar que este proceso se realiza en la arquitectura, siendo independiente de la fuente de las imágenes de entrada, es decir, que el sistema puede estar recibiendo simultáneamente información desde un formulario en un sitio web, desde una aplicación de escritorio, dispositivo móvil o desde una cámara o dispositivo de captura en tiempo real. De este modo, para la arquitectura será indiferente si se sube una (1) o, por ejemplo, mil (1000) imágenes o más al tiempo (en lotes).

Cada algoritmo de extracción de características se implementa como un microservicio independiente y cada uno de estos configura una suscripción a los eventos del *Cloud Storage*. La suscripción significa que cuando se emita un evento del *Cloud Storage* todos los microservicios suscritos a este evento, ejecutan el algoritmo para la imagen y guardan el resultado de cada algoritmo en forma de arreglo en una cola de mensajes asociando un identificador único a cada imagen, esto lo hacen de forma paralela. El servicio de mensajes emite un evento, al cual va a estar suscrito otro microservicio para almacenar los resultados de cada microservicio de extracción y se almacena para cada imagen todos los resultados de todas las ejecuciones independientes en el *Cloud Datastore*.

La operación completa de extracción de características de una imagen sobre el módulo basado en la arquitectura propuesta puede resumirse en los siguientes pasos:

1) Cargar imagen al módulo.
2) El microservicio de almacenamiento recibe la petición y carga la imagen al *Cloud Storage*. Al terminar la carga de la imagen al *Cloud Storage*, se emite un evento informando que hay una imagen nueva.
3) Los microservicios con los algoritmos para extraer características que se encuentren suscritos a los eventos que emite el *Cloud Storage*, reciben el evento mencionado anteriormente y procesan la información de la imagen, ejecutando el algoritmo que tiene implementado.
4) Se registra el resultado del algoritmo en una cola de mensajes, con un identificador de la imagen procesada, el cual va a ser igual para todos los microservicios que procesaron la imagen.
5) La cola de mensajes registra un evento a otro microservicio para que se realice el guardado de los resultados en una base de datos no relacional.

En la arquitectura propuesta se realiza una separación de cada algoritmo y etapa del proceso en microservicios separados, los cuales van a escalar automáticamente si hay un alto flujo de información. Los servicios más simples no van a crecer tanto como los algoritmos de extracción, y cada uno de estos se va liberando más rápido del procesamiento al ejecutarse en paralelo, por lo tanto, se utilizan menos recursos computacionales en mantener una operación correcta y rápida del sistema que al tener todos los algoritmos ejecutándose de forma secuencial. Si bien la ejecución de los microservicios de extracción se hace de manera asíncrona el resultado se puede integrar a un sistema CBIR que esté construido en una arquitectura monolítica, por ejemplo.

El esquema de suscripción a los eventos de cada parte del sistema también permite que se pueda ejecutar una cantidad ilimitada de algoritmos de extracción para cada imagen debido a que se pueden implementar servicios o algoritmos nuevos para aplicar a cada imagen sin necesidad de modificar los que ya existen en el sistema, por la independencia entre los servicios solo requiere implementar un servicio que se suscriba al *Cloud Storage* y que procese la imagen y retorne un resultado, para lo cual se define una interfaz que todos los microservicios deben seguir, si se desea desplegar sobre el módulo un algoritmo de extracción de características nuevo, solo se deberá crear el recurso nuevo y se puede desplegar de inmediato sin afectar componentes existentes a nivel de infraestructura o de lógica del programa.

Adicionalmente, el uso de almacenamiento en la nube reduce los costos al evitar la administración de discos físicos y optimiza la carga y descarga de los datos, reduciendo riesgos de pérdida de información. Los eventos de este servicio permiten realizar programación asíncrona de los microservicios, lo cual evita bloqueos sobre los recursos mientras esperan respuestas para continuar su procesamiento.

El almacenamiento de los resultados de los algoritmos, se escala en cuanto se aumenta la cantidad de peticiones. Cada imagen que entra al sistema se representa como una entidad con una identificación única, donde sus atributos son cada uno de los resultados de los algoritmos ejecutados a la imagen. La base de datos no relacional al soportar un esquema de guardado simultáneo no presenta problemas de bloqueos por concurrencia de operaciones de lectura y escritura.

El diseño presentado delega la responsabilidad de definir un diseño de un módulo de recuperación a partir de los algoritmos que se desean usar para realizar la búsqueda, por ejemplo, definiendo cuáles atributos de cada algoritmo podrán ser utilizados y las funciones de distancia que serán definidas para la comparación y determinación de similaridad de una imagen con respecto a otra.

B. *Implementación de la arquitectura propuesta*

La arquitectura propuesta fue implementada sobre *Google Cloud* y para cada uno de sus microservicios se habilitó una aplicación por consola que realiza la carga de imágenes en bloque. Fueron utilizados como algoritmos de extracción de características, los basados en matrices de coocurrencia, específicamente los patrones de contraste, disimilaridad, segundo momento angular (ASM), energía y homogeneidad, los cuales se implementaron como microservicios y su ejecución se realizó en paralelo.

Para la implementación de la extracción de características como fue mencionado, se usaron Características de la Matriz de Coocurrencia (GLCM), la cual se define como una forma estadística que permite analizar texturas, a partir de la coocurrencia de la escala de grises y relación espacial de los pixeles [22]. A partir de la matriz de coocurrencia se pueden extraer diversas características o patrones de una imagen, entre ellas las utilizadas en este trabajo, que son presentadas en la Fig. 3 con una breve definición y el modo en que son calculadas. Cada característica presentada en la Fig. 3 se implementó como un microservicio independiente sobre la arquitectura y cada una se ejecutó sobre una función en la nube diferente. Adicionalmente, cada algoritmo de extracción de características se desarrolló en Python. Vale destacar que, puede ser utilizado cualquier otro conjunto de algoritmos de extracción, pues la selección de estos en particular para evaluar la propuesta fue dada por el uso en otros proyectos anteriores, es decir, pudieron haber sido utilizados otras técnicas de textura, de forma o de color, etc.

Característica	Descripción	Forma de calcularla
Contraste	Indica las variaciones a nivel de grises en la matriz de coocurrencia (1).	$\sum_{ij=0}^{N-1} P_{ij}(i-j)^2$ (1) Contraste
Disimilaridad	Mide la variación de distribución de niveles de gris en la imagen (2).	$\sum_{ij=0}^{N-1} P_{ij}\lvert i-j\rvert$ (2) Disimilaridad
Homogeneidad	Mide la cercanía de la distribución de los elementos de la matriz de coocurrencia a su diagonal principal (3).	$\sum_{ij=0}^{N-1} \frac{P_{ij}}{1+(i-j)^2}$ (3) Homogeneidad
ASM (Angular Second Moment) y energía	Mide la uniformidad de la imagen (4).	$\sum_{ij=0}^{N-1} P_{ij}^2$ a) ASM. $\sum_{ij=0}^{N-1} \sqrt{ASM}$ b)Energía (4)

Fig. 3. Características basadas en la matriz de coocurrencia
Fuente: Autores.

IV. Evaluación y Discusión

En esta sección, se evalúa la implementación de la arquitectura propuesta en cuanto a su respuesta en términos de escalabilidad, elasticidad, disponibilidad y confiabilidad. Para ello, previamente fueron indicados los detalles de los algoritmos de extracción de características a utilizar, luego realizada la ejecución de pruebas mediante experimentos basados en simulación sobre la arquitectura y posteriormente se analizaron los resultados del tablero de indicadores entregados por la consola de *Google Cloud*.

Para la prueba de carga del sistema se usaron imágenes que hacen parte del dataset de imágenes Barcelona (Disponible en http://www.cs.unc.edu/~jtighe/Papers/ECCV10/), en la Fig. 4 se muestran algunos ejemplos de las imágenes de este conjunto de datos.

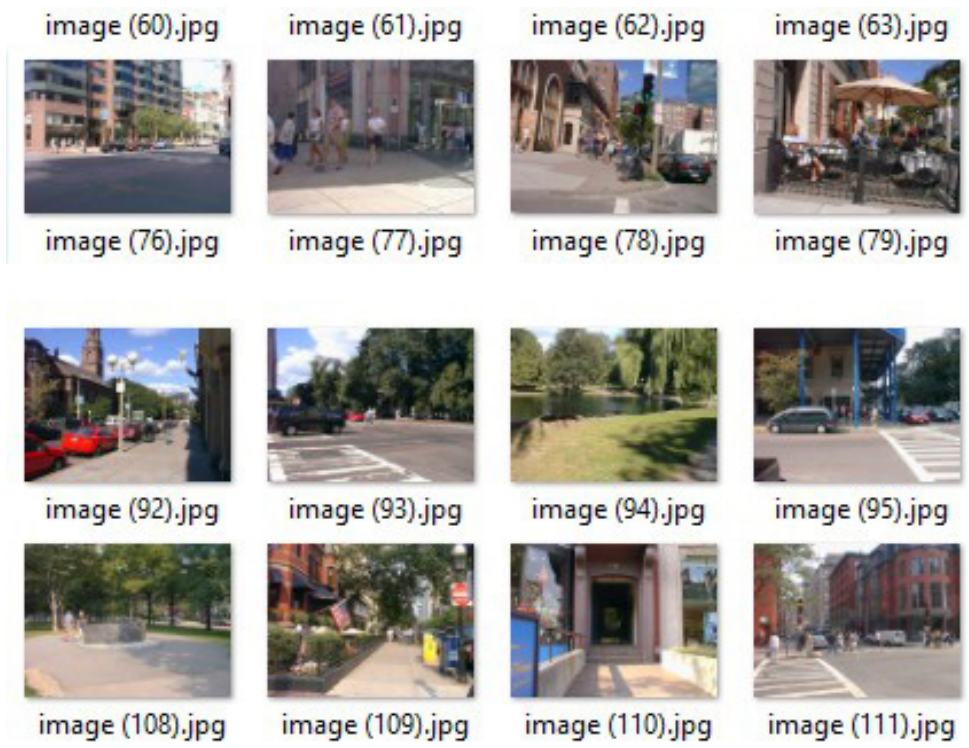

Fig. 4. Ejemplo de imágenes disponibles en el dataset Barcelona.
Fuente: Autores.

Las imágenes utilizadas se unieron en una misma carpeta, fueron renombradas y se realizó una carga inicial de 2500 imágenes simultáneamente mediante carga en bloque al *Cloud Storage*. Después de un tiempo (t = 20 minutos) se activó el mismo proceso en paralelo con mil (1000) imágenes adicionales, este proceso se repitió durante un intervalo de dos (2) horas, con la intención de simular un aumento repentino de imágenes al módulo y así evaluar la escalabilidad automática y otros elementos previamente indicados de la arquitectura propuesta.

A partir del proceso indicado de carga de las imágenes, inicialmente se encontró que, cuando se realiza la carga en paralelo de más imágenes al módulo el sistema, este escaló automáticamente y estuvo disponible a medida que aumentaba el número de imágenes a procesar.

En la Tabla 1 se presentan los microservicios de extracción de características basados en la matriz de coocurrencia (ASM, Contrast, Dissmilarity, Energy, Homogeneity) y el microservicio que recibe las imágenes que se van a subir al módulo (ProcessImage), la columna de la derecha muestra la información del número de instancias máximas que cada microservicio debió ejecutar en determinados momentos durante el período de ejecución de la prueba.

Tabla 1. Máximo número de instancias ejecutadas por microservicio.

Microservicio	Máximo número de instancias
ASM	10.24
Contrast	8.83
Dissimilarity	4.67
Energy	27.67
Homogeneity	8.44
ProcessImage	7,97

Fuente: Autores.

De la información obtenida se puede observar que cada microservicio requirió más de una instancia de ejecución para procesar las peticiones que entraban al módulo y que, por lo tanto, la arquitectura estuvo en capacidad de escalar automáticamente, dado que una sola instancia equivale a una unidad de procesamiento del servicio.

Posteriormente, en la Fig. 5, se presenta el número de instancias activas por minuto de cada microservicio del módulo, en la cual, a medida que transcurría el tiempo de la prueba, se observa que el número de instancias por minuto aumentó en el tiempo. Esto se debe a que el número de imágenes fue aumentando gradualmente durante el tiempo de la prueba. Por ejemplo, en la misma Fig. 5, el servicio ASM (Color naranja) incrementó el número de instancias máximas a medida que avanzó el tiempo de la prueba.

Se puede observar adicionalmente, que el número de instancias por minuto se redujo al final de la prueba debido a que no se cargaron más imágenes en paralelo. En la Fig. 5 también se muestra que el número de instancias presentó variaciones durante el tiempo, confirmando aspectos de escalado automático y disponibilidad, y que el microservicio *ProcessImage* (Color ámbar) que carga la imagen al módulo necesitó más instancias que el resto de microservicios durante la carga inicial de imágenes, pero a medida que el tiempo de prueba avanzó, el número de instancias por minuto fue menor, lo que indica que un microservicio puede escalar automáticamente más que otros, dependiendo de la cantidad de imágenes que procese el sistema en determinado instante de tiempo. También se observa la elasticidad presentada por el sistema, ya que, durante los períodos evaluados, este se adaptó a la carga de trabajo dada en cada momento. Lo anterior, permite confirmar la motivación de esta propuesta que, a diferencia de las propuestas mencionadas inicialmente, la extracción de todas las características se ejecuta en el mismo servicio o programa.

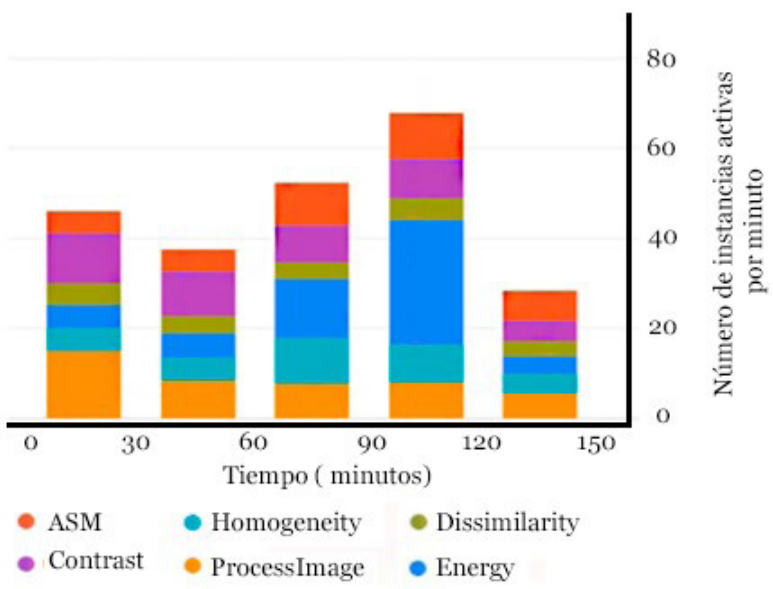

Fig. 5. Instancias activas durante el tiempo de ejecución por cada microservicio.
Fuente: Autores.

Adicionalmente, se puede verificar a partir de los resultados presentados en la Fig. 6, que la arquitectura logró escalar automáticamente como se evidencia en el microservicio para extracción de la característica energía (Energy), que fue el microservicio que necesitó más instancias durante la prueba, mientras que la disimilaridad fue la característica que menos instancias necesitó para procesar las imágenes del módulo, lo cual se atribuye a que se realizan cálculos adicionales, como una operación de raíz doble sobre el valor de ASM según la definición de la Ecuación 4b. Por ello, se puede afirmar que, no todos los microservicios necesitan escalar de igual forma, lo cual pudo ser observado durante las pruebas realizadas.

Por otra parte, se encontró que todas las peticiones fueron procesadas correctamente por lo que la arquitectura presentó confiabilidad durante la ejecución, y también se observa que el número de peticiones al *Cloud storage* fue aumentando gradualmente, como se muestra en la , evidenciando escalabilidad y elasticidad del servicio, sin que se presentaran bloqueos en la aplicación. Esto, debido a que dichos llamados se realizaron usando una cola de mensajes, en donde los picos altos indican que se incluyeron más mensajes, lo cual se representa en el ingreso de más imágenes al módulo durante la prueba. Las peticiones se realizaron

en paralelo sin afectar la ejecución de otros microservicios y mientras se aumentaban las peticiones se escaló el servicio logrando cumplir con todas las peticiones. No se detectaron errores en la consola de *Google Cloud* sobre alguna petición, esto permite observar que la arquitectura no presentó inconvenientes de confiablidad o disponibilidad durante la ejecución de la prueba.

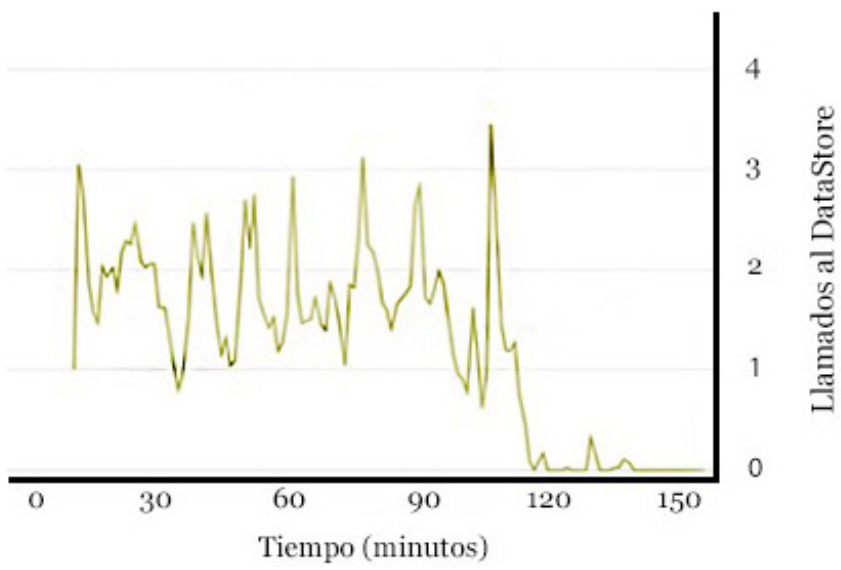

Fig. 6. Número de peticiones al DataStore.
Fuente: Autores.

Siendo los algoritmos de extracción de características implementados de forma individual como microservicios, es indiferente la clase o tipo de imágenes que se usen en un sistema CBIR, lo cual permite que se puedan implementar todos los algoritmos de extracción de características que se deseen, considerando que incluso se puedan realizar pruebas para determinar aquellas características más adecuadas a ser usadas en un sistema CBIR para un determinado ámbito o grupo de imágenes de un dominio específico.

Dado que las características a usar en la arquitectura propuesta se pueden desplegar en microservicios, para incluir nuevos algoritmos al sistema, no será necesario que se realice un despliegue completo del módulo. Un microservicio nuevo con un algoritmo de extracción de características determinado solo debe implementar la suscripción al evento del *Cloud Storage*, y podrá recibir las imágenes que se suben al módulo y extraer su información. Como ventaja adicional, este microservicio nuevo puede implementarse sobre cualquier lenguaje de programación soportado en la nube de *Google Cloud*.

V. Conclusiones

Se presentó una propuesta novedosa de una arquitectura basada en microservicios, que soporta el módulo de extracción de características en un sistema de recuperación de imágenes basada en contenido, en la cual cada algoritmo o técnica de extracción fue asociada a un microservicio, con independencia del tipo de imágenes de entrada a este módulo. Igualmente, el proceso de carga y guardado de los resultados fueron implementados como microservicios. A partir de esto, se evidencia la escalabilidad automática, elasticidad, confiabilidad y disponibilidad como ventajas del uso de este tipo de arquitectura con respecto a una arquitectura monolítica para el módulo de extracción de características específicamente.

La arquitectura propuesta hace uso de tecnologías en la nube tales como: funciones, almacenamiento de archivos en la nube, colas de mensajes y base de datos NoSQL, implementada sobre *Google Cloud*, y está basada en componentes independientes comunicados a través de eventos de tal modo que un componente nuevo puede ser implementado en el sistema sin tener que desplegar o realizar ajustes sobre los componentes existentes.

En términos de confiabilidad, las pruebas de carga de imágenes realizadas en paralelo, a las cuales se extrajeron sus características, permitieron evidenciar que la arquitectura escala y es elástica a medida que el número de peticiones aumenta y además que, el módulo termina la ejecución sin presentar inconveniente frente a la extracción de características del total de las imágenes cargadas.

El módulo propuesto hace uso de un esquema asíncrono para el proceso de extracción de características de las imágenes, destacándose que, el resultado entregado por este módulo de extracción podrá ser integrado a un determinado sistema CBIR con independencia del tipo de arquitectura general que soporte el sistema, el cual puede ser de tipo monolito, por ejemplo.

La separación de componentes en la arquitectura propuesta torna este trabajo como base de futuros estudios enfocados en el diseño de otros módulos de sistemas CBIR basados en microservicios, por ejemplo, el diseño de un módulo de selección de características que permita la parametrización o definición de diferentes configuraciones de un conjunto de características iniciales y determine el conjunto final a ser utilizadas en la recuperación de una imagen según el dominio o ámbitos a los cuales pertenezcan las imágenes.

Agradecimientos

Los autores agradecen al Grupo de Investigación en Inteligencia Computacional (GICO) adscrito al Departamento de Sistemas de la Universidad del Cauca por el apoyo en el desarrollo de este trabajo.

Referencias

[1] M. Becker, S. Lehrig & S. Becker, "Systematically deriving quality metrics for cloud computing systems," presented at *6th ACM/SPEC Int Conf Perform Eng,* ICPE 2015, TX, USA, pp. 169–174, 31 Jan-4 Feb. 2015. https://doi.org/10.1145/2668930.2688043

[2] M. Nabi, M. Toeroe & F. Khendek, "Availability in the cloud: State of the art," *J Netw Comput Appl,* vol. 60, pp. 54–67, Jan. 2016. https://doi.org/10.1016/j.jnca.2015.11.014

[3] X. Wang & J. Grabowski, "A Reliability Assessment Framework for Cloud Applications," presented at *Cloud Computing 2015*, IARIA, Nnc., Fr., 2015. Available: http://www.thinkmind.org/index.php?view=article&articleid=cloud_computing_2015_6_10_20143

[4] A. Latif, A. Rasheed, U. Sajid, J. Ahmed, N. Ali, N. I. Ratyal, B. Zafar, S. H. Dar, M. Sajid & T. Khalil, "Content-based image retrieval and feature extraction: A comprehensive review," *Math Probl Eng,* vol. 4, pp. 121, 2019. https://doi.org/10.1155/2019/9658350

[5] L. Kaliciak, H. Myrhaug & A. Goker, "Content-Based Image Retrieval in Augmented Reality," presented at *8th International Symposium on Ambient Intelligence*, ISAmI 2017, PO, PT, pp. 95–103, 21-23 Jun. 017. https://doi.org/10.1007/978-3-319-61118-1

[6] M. Meena, V. A. Bharadi & K. Vartak, "Hybrid Wavelet Based CBIR System Using Software as a Service (SaaS) Model on Public Cloud," *Procedia Comput Sci*, vol. 79, pp. 278–286, 2016. https://doi.org/10.1016/j.procs.2016.03.036

[7] M. B. Suresh & B. M. Naik, "A novel scheme for extracting shape and texture features using CBIR approach," *Int Conf Energy Commun Data Anal Soft Comput ICECDS 2017*, pp. 3399–3404, Jun. 21, 2018. https://doi.org/10.1109/ICECDS.2017.8390091

[8] J. Pradhan, S. Kumar, A. K. Pal & H. Banka, "A hierarchical CBIR framework using adaptive tetrolet transform and novel histograms from color and shape features," *Digit Signal Process A Rev J*, vol. 82, pp. 258–281, Nov. 2018. https://doi.org/10.1016/j.dsp.2018.07.016

[9] A. B. Raut, "NOSQL Database and Its Comparison with RDBMS," *Int J Comput Intell Res*, vol. 13, núm. 7, pp. 1645–1651, 2017.

[10] M. Villamizar, O. Garcés, H. Castro, M. Verano, L. Salamanca & S. Gil, "Evaluating the Monolithic and the Microservice Architecture Pattern to Deploy Web Applications in the Cloud," presented at *10th Comput Colomb Conf*, 10CCC, Bog., Co., 21-25 Sep. 2015. https://doi.org/10.1109/ColumbianCC.2015.7333476

[11] R. Grycuk, P. Najgebauer, R. Nowicki & R. Scherer, "Multilayer Architecture for Content-based Image Retrieval Systems," presented at *IEEE 12th Conf Serv Comput Appl*, SOCA, Khh., Tw., 18-21 Nov. 2019. https://doi.org/10.1109/SOCA.2019.00025

[12] S. Easwaramoorthy, U. Moorthy, C. A. Kumar, S. B. Bhushan & V. Sadagopan, "Content Based Image Retrieval with Enhanced Privacy in Cloud Using Apache Spark," *Commun Comput Inf Sci*, vol. 804, pp. 114–128, Feb. 2018. https://doi.org/10.1007/978-981-10-8603-8_10

[13] M. Meena, A. R. Singh & V. A. Bharadi, "Architecture for Software as a Service (SaaS) Model of CBIR on Hybrid Cloud of Microsoft Azure," *Procedia Comput Sci*, vol. 79, pp. 569–578, 2016. https://doi.org/10.1016/j.procs.2016.03.072

[14] A. Rahman, E. Winarko, y M. E. Wibowo, "Mobile content based image retrieval architectures," presented at *Int Conf Electr Eng Comput Sci Informatics,* IEEE, Yo. Id., pp. 208–211, 19-21 Sep. 2017. https://doi.org/10.1109/EECSI.2017.8239111

[15] A. Balalaie, A. Heydarnoori & P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," *IEEE Softw*, vol. 33, núm. 3, pp. 42–52, Mar. 2016. https://doi.org/10.1109/MS.2016.64

[16] T. Cerny, M. J. Donahoo & J. Pechanec, "Disambiguation and comparison of SOA, microservices and self-contained systems,"presented at *Res Adapt Converg Syst*, RACS '17, Krk., Pol., pp. 228–235, Sep. 2017. https://doi.org/10.1145/3129676.3129682

[17] Google LLC, "Cloud Functions," *Cloud.google*, [online], 2017. Available: https://cloud.google.com/functions/
[18] Google LLC, "Cloud Functions," *Cloud.google*, [online], 2020. Available: https://cloud.google.com/functions/
[19] Google LLC, "Cloud Storage," *Cloud.google*, [online], 2019. Available: https://cloud.google.com/storage/?hl=es
[20] Google LLC, "Pub/Sub," *Cloud.google*, [online], 2019. Available: https://cloud.google.com/storage/?hl=es
[21] Google LLC, "Cloud Datastore," *Cloud.google*, [online], 2019. Available: https://cloud.google.com/datastore/
[22] S. Mishra, B. Majhi, P. K. Sa & L. Sharma, "Gray level co-occurrence matrix and random forest based acute lymphoblastic leukemia detection," *Biomed Signal Process Control*, vol. 33, pp. 272–280, Mar. 2017. https://doi.org/10.1016/j.bspc.2016.11.021

Sandra Milena Roa Martínez es Profesora titular del Departamento de Sistemas y miembro del Grupo de Investigación en Inteligencia Computacional (GICO) de la Facultad de Ingeniería Electrónica y de Telecomunicaciones de la Universidad del Cauca (Colombia). Doctora en Ciencia de la Información y miembro del Grupo de Investigación - Nuevas Tecnologías en Información (GPNTI) de la Universidad Estadual Paulista (UNESP, Brasil) en la línea de investigación: Información y Tecnología. Magíster en Ingeniería con énfasis en Electrónica de la Universidad del Valle (Colombia) y Especialista en Redes de Comunicación de la Universidad del Valle. Ingeniera de Sistemas en la Universidad Industrial de Santander (Colombia). Actuando principalmente en las áreas de investigación: Findability, recuperación y representación de imágenes digitales, arquitectura de la información, contenidos digitales, tecnologías inmersivas y publicación de datos de investigación. https://orcid.org/0000-0002-2271-6101

Andrés Felipe Ruiz Velasco es Ingeniero de sistemas de la Universidad del Cauca (Colombia). Actualmente es estudiante de la Maestría en computación en la Universidad del Cauca. Líder técnico en una empresa de Software en la nube. Sus áreas de investigación son tecnologías inmersivas, arquitecturas de software, computación en la nube y procesamiento de imágenes. https://orcid.org/0000-0002-7741-707X