Desarrollo de aplicación basada en microservicios para la generación de consultas SQL con un enfoque no-code

Development of an application based on microservices for the generation of SQL queries with a no-code approach

Mauricio Rojas-Contreras, PhD[1], Steban José Castro-Pérez, Ing[2]
[1]Universidad de Pamplona, Pamplona, Norte de Santander, Colombia mrojas@unipamplona.edu.co
[2]Universidad de Pamplona, Pamplona, Norte de Santander, Colombia steban.castro@unipamplona.edu.co

***Resumen*– El presente trabajo tiene como objetivo el diseño y desarrollo de una aplicación web basada en microservicios implementados en diferentes lenguajes de programación, con el fin de crear una herramienta no-code para generar consultas SQL sin la necesidad de usar Native Query. El resultado central del trabajo es la implementación de una aplicación web basada en microservicios con una interfaz gráfica que le facilite a un usuario, independientemente de su nivel de conocimiento en programación, realizar reportes basados en sus necesidades. Para lograr dicha meta se realizó un análisis a la experiencia de usuario con interfaces gráficas para lograr la accesibilidad que se requiere, junto a un análisis de patrones de arquitectura y el planteamiento de un stack tecnológico**

***Palabras claves*—Microservicio, No-code, SQL, Patron de arquitectura, experiencia de usuario.**

***Abstract*- The objective of this work is the design and development of a web application based on microservices, implemented in different programming languages, in order to create a no-code tool to generate SQL queries without the need to use Native Query. The central result of the work is the implementation of a web application based on microservices with a graphical interface that makes it easier for a user, regardless of their level of programming knowledge, to make reports based on their needs. To achieve this goal, an analysis of the user experience with graphic interfaces was carried out to achieve the accessibility that is required, together with an analysis of architecture patterns and the approach of a technological stack.**

***Key words*- microservice, No-code, SQL, architecture pattern, user experience**

I. Introducción

En el modelo empresarial se le asigna a un departamento el manejo de datos relevantes con el cual se prevé obtener métricas de diferentes parámetros que generan información valiosa para la toma de decisiones y conocer el estado actual de la empresa, esta situación obliga a una ardua capacitación en cuanto conocimiento sobre manejo de bases de datos, de lo cual se desglosan problemas como, el tiempo de aprendizaje, la necesidad de conocimientos previos sobre informática, la sintaxis SQL y la lógica que conlleva[1][2].

Además, existen conceptos como el pool de conexiones, el cual gestiona las sesiones para el acceso de la base de datos, estas son limitadas y un mal manejo de las mismas termina en un pool sin la capacidad de atender solicitudes nuevas, convirtiéndose en un bloqueo de acceso hasta no terminar o cerrar sesiones abiertas[3].

Por otro lado, una vez se obtienen datos tratados, estos aún no representan por sí mismos información valiosa, dado que necesitan un manejo sobre un contexto para que tengan sentido, tal como puede ser crear gráficas, las cuales muestran lo que normalmente los datos en bruto no reflejan, lo que conlleva a una exportación de datos para luego ser tratados, siendo en algunos casos imposible hacer un tratado de los mismos en SQL[4].

De acuerdo a lo anterior, se genera la necesidad de una solución que permita disminuir la complejidad de escribir consultas SQL de manera significativa y teniendo en cuenta la flexibilidad de generar las mismas ya que está orientado a personas sin conocimiento en este campo, además, junto con una solución centralizada se debe tener en cuenta el uso adecuado de las conexiones a las bases de datos con el fin de tener disponibilidad y evitar saturar el pool de conexiones, la solución debe ser sencilla de usar, siendo de vital importancia dar una experiencia de usuario donde sin importar el nivel de conocimientos se pueda entender, como en el caso particular de Excel, el cual es un entorno que permite su uso independientemente del nivel de documentación del sujeto[5].

Por lo tanto, el alcance central de este trabajo es generar una herramienta no-code de consultas a bases de datos postgres, orientado principalmente en la interfaz gráfica con la funcionalidad principal de crear reportes, teniendo en cuenta que el público objetivo de las herramientas no-code es principalmente usuarios con pocos conocimientos de programación, por ello el énfasis sobre la experiencia de usuario con la interfaz gráfica. Adicionalmente, la utilización

de una arquitectura basada en microservicios es fundamentales para mantener un rendimiento óptimo, considerando que tienen la particularidad de escalar de manera individual y siendo cada uno de ellos responsables de un dominio de la herramienta, para crear una buena experiencia de usuario.

II. MATERIALES Y MÉTODOS

En esta se sección se especifican los materiales y métodos utilizados en el desarrollo de la aplicación para la generación de consultas SQL con un enfoque No-code. En forma particular, las etapas desarrolladas en el proceso de investigación se muestran en la figura 1, las cuales se adaptan del modelo de ciclo de vida en cascada de la ingeniería del software[6], [7][8].

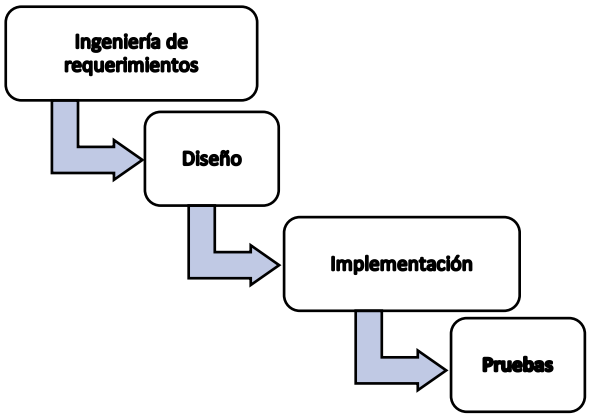

Fig. 1. Proceso de Desarrollo de software aplicación de generación SQL

La etapa de ingeniería de requerimientos tiene por alcance la identificación y análisis de los requerimientos funcionales y no funcionales de la aplicación para la construcción de consultas SQL bajo un enfoque No-code[9]. En forma específica, los requerimientos funcionales hacen referencia a los microservicios que implementan las funcionalidades de la aplicación y los requerimientos no funcionales hacen referencia a las características de la aplicación que son perceptibles al usuario pero que no tienen asociada una funcionalidad específica en la aplicación[10].

La etapa de diseño tiene por alcance generar la arquitectura de software del producto, la cual especifica los módulos o subsistemas que van a integrar el producto final y en este caso están asociados a los microservicios implementados para la aplicación[11].

La etapa de implementación tiene por alcance el Desarrollo de cada uno de los microservicios identificados en la etapa de diseño y los cuales pueden ser implementados en diferentes lenguajes de programación. En forma complementaria, en esta etapa se lleva a cabo el proceso de integración de los microservicios con el fin de responder a los requerimientos del product software.

En la etapa de pruebas se ejecutaron las pruebas unitarias para cada uno de los microservicios implementados. En forma complementaria, se aplicaron las pruebas de integración para la ejecución de los microservicios en forma conjunta[12][13].

III. RESULTADOS

La aplicación para la generación de consultas SQL bajo un enfoque de Desarrollo no-code, tuvo como resultado de la identificación de requerimientos no funcionales la utilización de una arquitectura de microservicios y la experiencia de usuario de la aplicación debe ser muy similar a la utilizadas en las herramientas de Desarrollo no-code. En forma complementaria, se identificaron como requerimientos no funcionales la interfaz gráfica del Query Builder debe tener altos niveles de usabilidad y el soporte de bases datos de la aplicación está restringido solo a bases de datos PostgressSQL.

En cuanto a los requerimientos funcionales, se identificaron las funcionalidades para la aplicación de la siguiente manera:

Registrarse e iniciar sesión en la herramienta.

Crear una integración de su base de datos con la plataforma.

Consultar las integraciones existentes con sus bases de datos.

Generar consultas de información mediante las integraciones con la base de datos.

Consultar una lista de joins predeterminados basados en el esquema de la base de datos.

Generar reportes con gráficas a partir de las consultas.

En la figura 2, se visualiza el diagrama de casos de uso con los requerimientos funcionales de la aplicación.

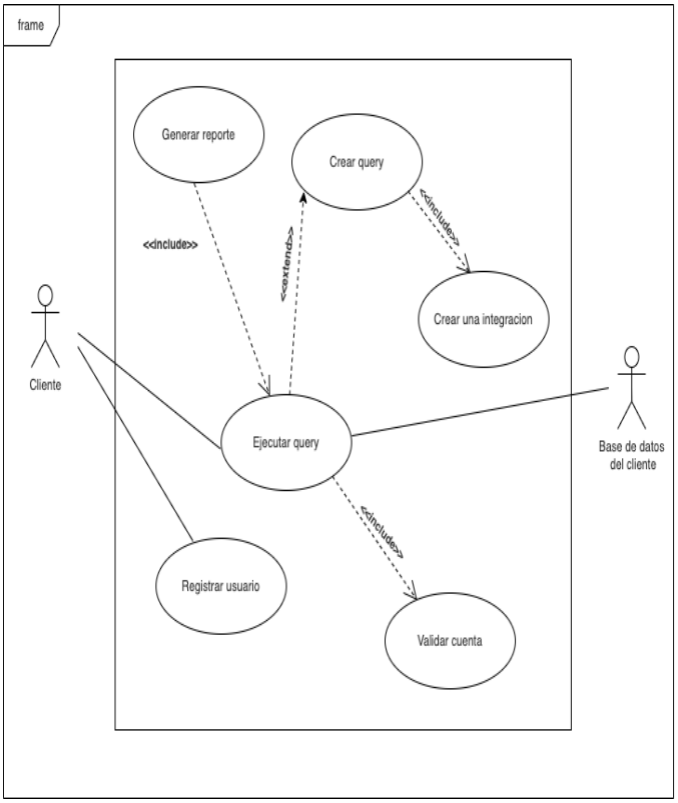

Fig. 2. Diagrama de casos de uso aplicación para la generación de consultas SQL.

La etapa de diseño generó como artefacto de software principal la arquitectura de software del producto, la cual estuvo basada en una arquitectura de microservicios. Este tipo de arquitectura, permite bajar la complejidad de implementación del producto, aumentar la probabilidad de escalabilidad de la aplicación y adicionalmente mejora las condiciones para la implementación de nuevas funcionalidades requeridas por una organización dedicada al desarrolño de software. En forma complementaria, permite mejorar los indicadores de estimación de tiempos para el Desarrollo de software y permite la utilización de diferentes lenguajes de programación en la implementación de cada uno de los microservicios.

La figura 3 visualiza la arquitectura de microservicios de la aplicación y en forma específica muestra los microservicios identificados para la aplicación.

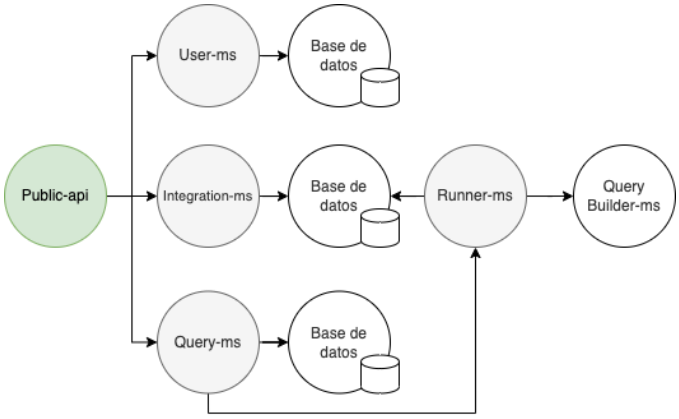

Fig. 3. Arquitectura de microservicios para la aplicación de generación de consultas SQL

En forma específica, se presenta la especificación de cada uno de los microservicios identificados en la arquitectura:

El microservicio Public-API-ms, es un microservicio escrito en go usando el framework de echo, encargado de exponer los endpoints de los microservicios y la autenticación con la aplicación mediante JWT (Json Web Token), el único endpoint que no requiere la autenticación es crear usuario.

El microservicio User.ms se desarrolló en go usando el Framework Echo, tiene por alcance administrar el dominio de los usuarios en donde se almacenan todos los datos básicos.

El microservicio Integration-ms se escribió kotlin usando el framework spring, encargado de administrar el dominio de las integraciones en donde se almacena la información de la integración como son los datos de conexión a las bases de datos de los clientes y los esquemas de las mismas, también encargado de los servicios CRUD de las integraciones.

El microservicio Query-ms, es un microservicio escrito en kotlin usando el framework spring, encargado de administrar el dominio de las queries en donde se almacena la información relacionada a las queries que se generan en SQL y JSON generadas por el frontend y además tiene por alcance la generación de las gráficas.

El microservicio Runner-ms, es un microservicio escrito en kotlin usando el framework spring, tiene el dominio de conexión con las bases de datos de los clientes siendo un administrador de pool de conexiones y encargado de ejecutar consultas SQL, hace uso de la información del microservicio Integration-Ms con el rol de lectura.

El microservicio Query-Builder-ms, es un microservicio escrito en javascript con node.js usando el framework express.js, se encarga de traducir la request que se genera en el frontend en JSON a SQL con ayuda de la librería knex.js.

Para diseñar la arquitectura de microservicios de la aplicación se usó el Diseño dirigido por dominio (DDD), [14] el cual consiste en proporcionar un framework que tiene como finalidad construir un conjunto de microservicios bien diseñados[15][16].

Los pasos para definer microservicios usando DDD se pueden visualizer en la figura 4.

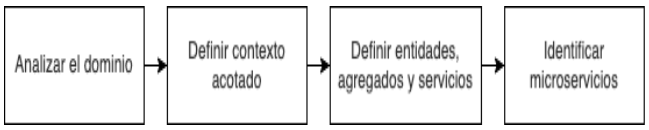

Fig. 4. Pasos para identificar microservicios usando DDD

En el análisis del dominio, se mapean todas las funciones del negocio y la manera cómo se conectan. En la figura 5, se visualiza el diagrama de análisis inicial del dominio.

Fig. 5. Diagrama de análisis inicial del dominio

En la definición del contexto acotado, aún no se ha definido nada sobre la implementación o las tecnologías que se usarán, pero sí un contexto inicial de la aplicación. Para generar dichos contextos acotados se agruparán las funcionalidades del diagrama anterior que tienen dependencias directas, para generar modelos de dominio menos acoplados. En la figura 6 se visualiza el diagrama de análisis de dominio con contexto acotado.

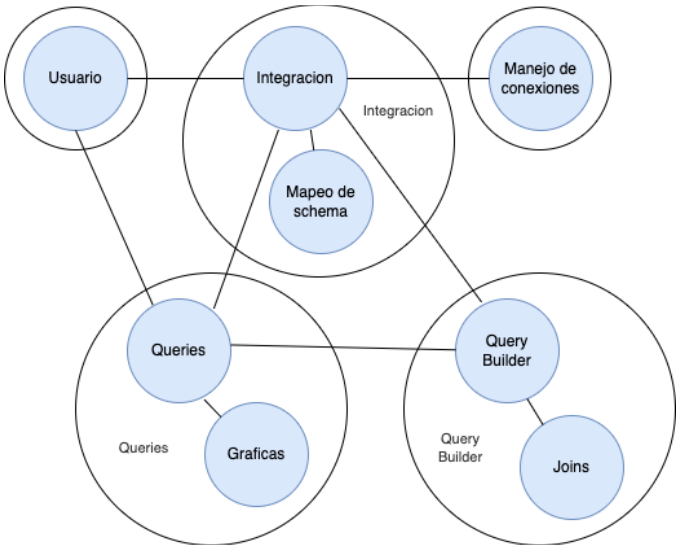

Fig. 6. Diagrama de análisis de dominio con contexto acotado

Una vez construido el diagrama de análisis de dominio con contexto acotado, se concluye la fase estratégica y entra en ejecución la fase táctica del DDD en donde se definen los modelos de dominio con mayor precisión mediante patrones.

En la definición de entidades, agregados y servicios, dentro de los patrones se encuentran las entidades, objetos de valor, agregados, servicios de dominio y aplicación, y eventos de dominio que se definen de la siguiente manera:

Entidades: Una entidad es un objeto único. En el Proyecto las entidades identificadas corresponden a Integración, Usuario, Conexión, Query.

Objetos de valor: Son objetos los cuales no tienen una identidad única, definen los valores de un atributo.

Agregados: Se define como un límite de consistencia alrededor de una o más entidades.

Servicios de dominio y aplicación: un servicio es un objeto que implementa lógica sin necesidad de mantener algún estado, la distinción entre servicios de dominio y de aplicación recae principalmente en el tipo de lógica que encapsula.

Eventos de dominio: estos eventos se usan para notificar a otras partes del sistema que se ejecute alguna acción o evento del dominio, por ende, no aplica acciones como “insertar un registro en una tabla”, hace referencia a una acción de dominio como puede ser “la orden fue enviada”. Los eventos de dominio que se definieron son:

-Mientras se pueda establecer conexión con una integración su estado será true y en caso contrario false.

-Cuando se crea una integración y se establezca la conexión se mapeara un esquema.

-Una query puede ejecutarse siempre y cuando la conexión sea exitosa.

En la figura 7 se visualizan los eventos de dominio, servicios de dominio y aplicación del Sistema.

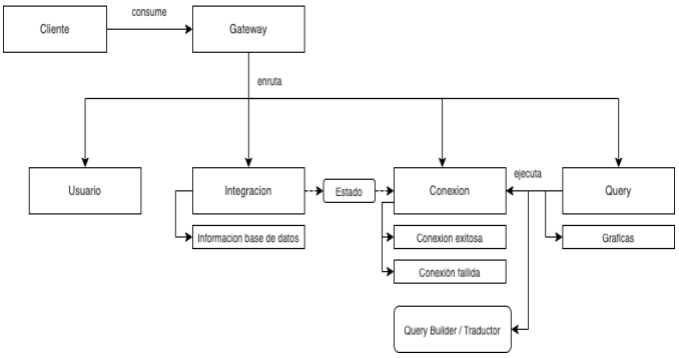

Fig. 7. Diagrama de eventos de dominio y servicios de dominio y aplicación.

En la identificación de microservicios, se especificaron los siguientes microservicios:

-Public-api: Es un servicio de dominio llamado "Gateway" definido por el patrón enrutador por puerta de enlace.
User-ms: Es el agregado llamado "Usuario" tiene el dominio de la informaciónón de los usuarios.
-Integration-ms: Es el agregado llamado "Integración" tiene el dominio de la informaciónón de las bases de datos de los usuarios.
-Query-ms: Es el agregado llamado "Query" tiene el dominio de la informaciónón de las consultas que genera el usuario basado en una integración.
-Runner-ms: Es el agregado llamado "Conexión" tiene la responsabilidad de manejar el pool de conexiones hacia las bases de datos de las integraciones, además de ejecutar las consultas SQL.
-Query-builder-ms: Es un servicio de dominio llamado "Query builder" encargado de la construcción de las consultas SQL a partir de la solicitud que genera el frontend.

En la etapa de pruebas, se comparó el comportamiento esperado con el comportamiento observado del Sistema, obteniendo los siguientes resultados.

En la figura 8, se visualiza la interfaz principal del Proyecto, la cual integra los microservicios y da acceso a las funcionalidades del Sistema.

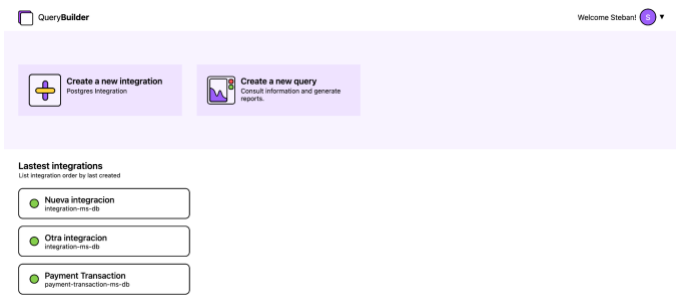

Fig. 8 Interfaz principal del sistema

En la figura 9, se visualiza la interfaz para la creación de integraciones.

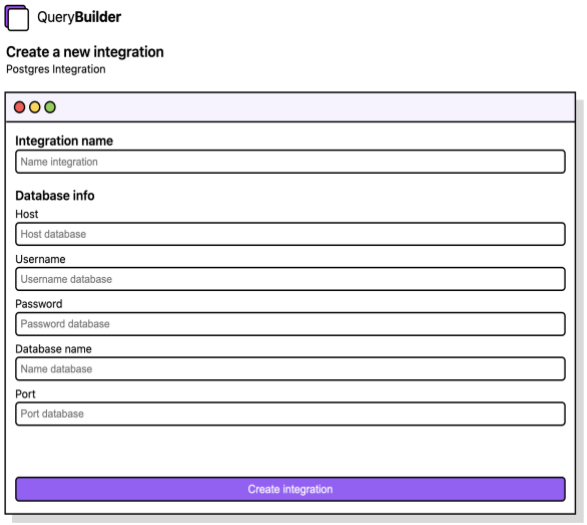

Fig. 9. Interfaz para crear integraciones

En la figura 10, se muestra el resultado de la prueba de listado de queries de una integración.

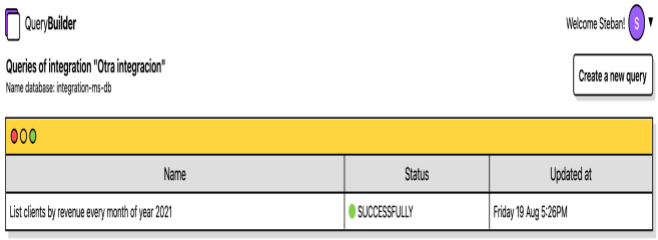

Fig. 10. Lista de queries de una integración

En la figura 11, se muestra la prueba de la interfaz del query-builder.

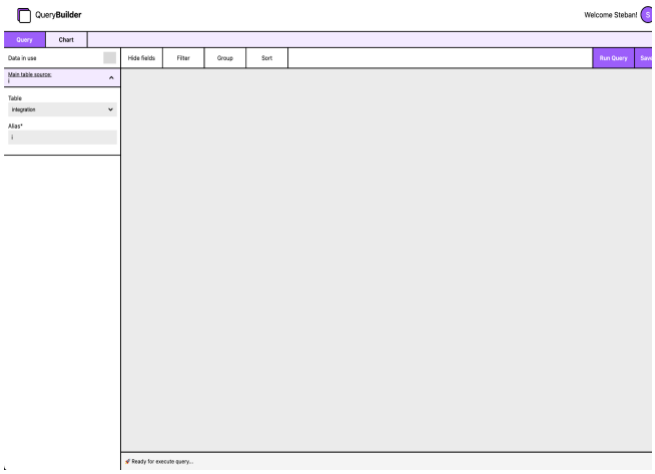

Fig. 11. Prueba interfaz del Query-builder

IV CONCLUSIONES

El uso de la arquitectura de microservicios potencializa características como la comunicación autónoma entre microservicios, la flexibilidad en los lenguajes de desarrollo utilizados para la implementación de cada microservicio y la escalabilidad en las aplicaciones al poder integrar microservicios para responder a nuevos requerimientos funcionales y no funcionales.

La arquitectura de microservicios permite el control descentralizado de datos, con un enfoque arquitectónico y organizacional distribuido, lo cual potencializa niveles de acoplamiento bajo en la comunicación entre microservicios, lo cual minimiza el impacto de errores en un microservicio, permitiendo que la aplicación no se interrumpa en la ejecución en forma integral.

El diseño dirigido por dominio se constituye en un marco metodológico para la identificación de microservicios a partir de los requerimientos del sistema en el proceso de desarrollo de software, adicionalmente permite identificar de manera sencilla e intuitiva la especificación de los microservicios.

Los procesos de desarrollo de software soportados en tecnologías no code permiten aumentar la probabilidad de que personas no técnicas con alto conocimiento de los procesos de negocio desarrollen sus propias aplicaciones de acuerdo a sus necesidades y a las reglas del negocio.

El desarrollo de arquitecturas basadas en microservicios permite una gran versatilidad a la hora de desarrollar y escalar de manera automática una infraestructura.

La arquitectura de microservicios permite aumentar la probabilidad de escalabilidad en sistemas orientados a microservicios, soportados en los tiempos de respuesta en la implementación de nuevas funcionalidades a través de la integración de microservicios desarrollados previamente en la organización.

REFERENCES

[1] J. Quishpe-Armas and Camacho-Leon. Sergio, "Análisis del desarrollo de software en no desarrolladores, mediante el uso de mendix, como herramienta para el aprendizaje y creación de apps," in *9 Congreso Internacional de Innovación educativa*, 9 Congreso Inte, 2023, pp. 100–105.

[2] N. Ramirez-Mosquera and C. Barbosa-Gironza, "APLICACIÓN WEB PARA LA GESTIÓN DE INFORMES Y REPORTES ASOCIADOS A LA SEGURIDAD Y SALUD EN EL TRABAJO," Universidad Autonoma de Occidente, Cali, 2022.

[3] B. Nevárez-Chávez, "Análisis de Tecnologías de bases de datos en el WWW," Tésis, Chihuahua, 2020.

[4] C. Jaime, M. Moctezuma, M. Gastón, C. Campos, M. Barrón, and B. Bernardo Hernández Sánchez, "Arquitectura de microservicios para optimizar el acceso a datos del SiT Log Lab," *Comunicaciones*, 2023.

[5] M. Jaramillo, A. Hernán, and F. G. Arias, "Desarrollo de la interfaz de usuario para la aplicación web 'Mapa de Crecimiento,'" Medellín, 2023. [Online]. Available: www.udea.edu.co

[6] S. Sánchez, M. Ä. Sicilia, and D. Rodríguez, *Ingeniería del software. Un enfoque desde la guía SWEBOK*. 2012.

[7] M. Rojas-Contreras, C. Peña-Cortés, and N. Gamba-Peñaloza, "Desarrollo de aplicación móvil para la gestión de neuroseñales registradas a través de una interfaz cerebro computador," in *20 LACCEI Multiconference for engineering, education and Technology*, Boca Raton, 2022.

[8] C. Máñez-carvajal, J. Francisco, U. Católica, D. V. San, V. Mártir, and S. Corazón, "Desarrollo de aplicación móvil para niños con dificultades de aprendizaje de la lectura y escritura Development of a mobile application for children with reading and writing learning difficulties," vol. 33, no. 1, pp. 271–278, 2022.

[9] B. Bruegge and A. Dutoit, *Ingeniería de software orientado a objetos*. 2002.

[10] C. Merino-Ibañez, "Metodología en la determinación de la granularidad de un Microservicio," *Centro nacional de Investigación y desarrollo tecnológico*, 2023.

[11] W. J. Trebejo Loayza and F. Sobero Rodríguez, "Herramienta para el modelado y generación de código de Arquitecturas de Software basadas en Microservicios y Diseño guiado por el dominio (DDD)," *Revista peruana de computación y sistemas*,

vol. 4, no. 2, pp. 3–14, Dec. 2022, doi: 10.15381/rpcs.v4i2.24855.
[12] C. Adolfo and L. Mamani, “Pruebas de Software para Microservicios Software Testing for Microservices,” *Revista Innovación y Software*, vol. 4, no. 1, 2023.
[13] M. I. Lund, S. B. Chavez, D. Checcarelli, V. Alferillo, A. Martin, and E. G. Ormeño, “TESTING GUIADO POR EL DISEÑO. DISEÑO DIRIGIDO POR CASOS DE USO,” 2017.
[14] A. Echeverría *et al.*, “"DDD (diseño dirigido por el dominio) y aplicaciones Enterprise: ¿fidelidad al modelo o a las herramientas?”
[15] M. G. Cambarieri, F. Difabio, and N. García Martínez, “Implementación de una Arquitectura de Software guiada por el Dominio,” *Simposio Argentino de Ingeniería del Software*, pp. 192–209, 2020.
[16] T. I. Rivas, J. Manuel, and A. Jiménez, “Diseño Dirigido por Modelos de Aplicaciones para Dispositivos Móviles,” Universidad de Almeria, 2016.