APLICACIÓN DE GESTION DE PEDIDOS PARA “BOGOTHRASH CAFE” IMPLEMENTANDO MICROSERVICIOS.

Order management application for “BogoThrash Café” implementing microservices.

Aplicación de gestión de pedidos para “Bogothrash café” implementando microservicios.

**Nombre de autores**

Samuel Armando Arce

[sarce@poligran.edu.co](mailto:sarce@poligran.edu.co)

Salome Dorado Suaza

[sdorado@poligran.edu.co](mailto:sdorado@poligran.edu.co)

Carlos Alberto Velandia

[cavelandia@poligran.edu.co](mailto:cavelandia@poligran.edu.co)

Politécnico grancolombiano

Colombia

# Resumen

Este apartado debe estar diseñado para invitar al lector a continuar leyendo el documento de tal manera que se debe indicar: la motivación y propósito del trabajo, el método para llevar a cabo lo que va del documento y una idea de los resultados logrados. La extensión máxima es de 300 palabras, y puede dividirse de acuerdo con los siguientes apartados: Objetivo, Métodos y materiales, Resultados y Conclusiones (generalidades)

# Palabras clave

En este apartado se deben enunciar de **3 a 6 palabras o descriptores** asociados al contenido del manuscrito. Idealmente en este apartado se deben incluir términos universalmente categorizados y organizados en el área de conocimiento en el cual se publica; estos términos o tesauros -como son conocidos- facilitan la indización del artículo en bases de datos y buscadores especializados por ello se recomienda que no sea una transcripción separada por comas del título del documento. Todas **deben ser escritas en minúscula y separadas por coma** (,).

# INTRODUCCIÓN

**¿Qué son?**

La arquitectura de microservicios ha emergido como una de las principales tendencias en el desarrollo de software moderno, especialmente en sistemas que requieren escalabilidad, flexibilidad y mantenimiento eficiente. Los microservicios se definen como un enfoque arquitectónico en el que una aplicación se construye como un conjunto de servicios pequeños, autónomos y especializados, que se comunican entre sí mediante APIs bien definidas y protocolos ligeros. [1] gracias a esta estructura modular este paradigma ha transformado la forma en que se diseñan y despliegan aplicaciones, permitiendo que cada componente evolucione de manera independiente, lo que facilita la integración continua, el despliegue automatizado y la resiliencia del sistema. [2] En contraste con las arquitecturas monolíticas tradicionales, los microservicios ofrecen ventajas como la reutilización de componentes y la posibilidad de asignar equipos de desarrollo a servicios específicos. [3]

Los microservicios funcionan como componentes distribuidos que se comunican entre sí mediante protocolos ligeros como HTTP y formatos como JSON o XML. Cada servicio expone una API que permite a otros servicios o clientes interactuar con él. Esta comunicación suele seguir los principios REST, lo que facilita la interoperabilidad y la estandarización. [4]

**Usos actuales**

En su mayoría el paradigma de desarrollo en base a microservicios ve su utilidad en el desarrollo web, con frameworks como “SpringBoot” con sus respectivas librerías que brindan útiles herramientas como métricas del microservicio, análisis de tráfico y monitoreo del estado de conexión a la base de datos [5]. En general, gracias a sus ventajas, incluyendo mayor escalabilidad, rendimiento, flexibilidad y eficiencia [6]; los microservicios están rápidamente remplazando la antigua arquitectura monolítica de sistemas legado a través del mundo.

En casos más específicos los microservicios también ven uso en sistemas que consten de varias partes que requieran comunicación entre sí. Como en la Automatización industrial, la cual implica la utilización de sistemas de control que comprenden máquinas, actuadores, sensores, procesadores y redes para realizar tareas con el objetivo de automatizar la producción [7]. Los microservicios se vuelven una parte fundamental en la Automatización industrial gracias a su capacidad de independización, permitiendo que puedan ser desplegados y evaluados de forma independiente. Entre las ventajas de los microservicios se incluyen la posibilidad de desarrollarse en distintos lenguajes, la capacidad de escalarse de forma independiente y la flexibilidad para desplegarse en el hardware más adecuado [7].

**Relevancia**

Desarrollar un sistema para la administración de una cafetería se vuelve más alcanzable con el uso de microservicios. Si se utiliza microservicios para la administración de pedidos, productos y se guardan en una base de datos como las de MySQL; registrar usuarios y gestionar su autenticación con una API de Google; utilizar la API de JavaMail o Twilio para mantener al tanto a los usuarios por medio de notificaciones de sms o correo [8] convierte la tarea de crear y gestionar distintas interfaces en una simple tarea de mantenimiento.[9]

El uso de microservicios no solo ahorra tiempo al equipo de desarrollo a la hora de conectar con distintas aplicaciones e interfaces, sino que también garantiza la seguridad y satisfacción de los usuarios a la hora de interactuar con la cafetería. [10]

**Contexto para la aplicación**

Bogothrash Café es una cafetería temática ubicada en Bogotá, reconocida por su ambiente alternativo y su enfoque en la cultura musical underground. Con el crecimiento de su clientela y la diversificación de sus productos, sus administradores han identificado la necesidad de modernizar sus operaciones para mejorar la experiencia del cliente, optimizar la gestión interna y facilitar la escalabilidad del negocio.

Actualmente, los pedidos se gestionan de forma manual, lo que genera demoras en la atención, errores en la toma de órdenes y dificultades para llevar un control eficiente del inventario y los pagos. Por ello, se ha decidido implementar un sistema digital basado en arquitectura de microservicios que permita automatizar y centralizar los procesos clave del negocio.

El sistema estará compuesto por cinco microservicios principales, cada uno encargado de una funcionalidad específica:

**Gestión de Productos**

Bogothrash Café ofrece una variedad de bebidas artesanales, snacks veganos y productos temáticos. Este microservicio permitirá administrar el catálogo de productos, incluyendo su nombre, descripción, precio, categoría y disponibilidad en inventario. Los administradores podrán actualizar el stock en tiempo real, y los clientes podrán visualizar los productos disponibles desde la aplicación.

**Gestión de Pedidos**

Los clientes podrán realizar pedidos directamente desde el catálogo, seleccionando productos y cantidades. El sistema permitirá visualizar el estado del pedido (pendiente, en preparación, listo, entregado), y los empleados podrán gestionar la cola de pedidos desde una interfaz interna. También se incluirá un historial de pedidos por usuario y la posibilidad de cancelar o modificar pedidos antes de ser preparados.

**Pagos**

Para facilitar la transacción, el sistema integrará pasarelas de pago que permitirán realizar pagos seguros mediante tarjeta de crédito, débito o billeteras digitales. Se generarán comprobantes de pago y se contemplará la posibilidad de reembolsos en caso de cancelaciones. El pago será validado antes de que el pedido sea confirmado y enviado a preparación.

**Notificaciones**

Con el fin de mantener informados a los clientes, se implementará un sistema de notificaciones que enviará alertas por correo electrónico o SMS. Estas notificaciones incluirán confirmaciones de pedido, actualizaciones de estado y promociones especiales. Los usuarios podrán configurar sus preferencias de notificación desde su perfil.

Este sistema busca no solo agilizar la operación interna de Bogothrash Café, sino también ofrecer una experiencia moderna, personalizada y eficiente para sus clientes. El modularidad de los microservicios permitirá escalar el sistema fácilmente en el futuro, integrando nuevas funcionalidades como programas de fidelización, recomendaciones personalizadas o integración con redes sociales.

**Definición de ambiente de desarrollo**

**IDE**

* Eclipse (Java)
  + <https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2023-06/R/eclipse-inst-jre-win64.exe>
* VScode, (Node.JS)
  + <https://code.visualstudio.com/download>

**Módulo de gestión de productos**

* **Frameworks- herramientas**
  + **S**pringboot
    - <https://start.spring.io/>
  + PostgreSQL
    - <https://www.postgresql.org/download/>
  + Supabase DB
    - <https://supabase.com/>

**Módulo de gestión de pedidos**

* **Frameworks- herramientas**
  + SpringBoot REST
  + PostgresSQL
  + SupaBase Functions
    - <https://supabase.com/docs/guides/functions>

**Módulo de pagos**

* **Frameworks- herramientas**
  + Mercado Pago API
    - <https://www.mercadopago.com.co/developers/es>
  + SpringBoot REST
  + Paypal SDK
    - <https://developer.paypal.com/api/rest/>

**Módulo de notificaciones**

* **Frameworks- herramientas**
  + Twilio (sms)
    - <https://www.twilio.com/en-us/messaging/apis/conversations-api>
  + JavaMail
    - <https://javaee.github.io/javamail/#Download_JavaMail_Release>
  + Spring Events

# MÉTODO

El presente proyecto se enfoca en el desarrollo de un sistema para la gestión integral de una cafetería, implementado mediante una arquitectura basada en microservicios. Su objetivo principal es ofrecer una solución escalable, modular y de fácil mantenimiento, en la que cada componente se encargue de una funcionalidad específica del negocio.

Como primer paso, se elaboraron los diagramas de clases, organizados en distintos paquetes modelo, uno por cada microservicio. A continuación, se procedió a la creación de un proyecto base en Spring Boot, utilizando el asistente Spring Initializr. Este proyecto sirvió como punto de partida para el desarrollo de los demás servicios. Para ello, se incorporaron diversas dependencias, entre ellas: Spring Data JPA, Spring Starter JDBC, Spring Starter Web, Spring DevTools, PostgreSQL y Spring Starter Mail. Además, se integraron herramientas como Lombok, para simplificar el código mediante anotaciones, y Swagger, para la generación automática de documentación.

Una vez establecido el entorno base, se inició la construcción del primer microservicio, correspondiente al registro de pedidos. Para almacenar las entidades del modelo, se configuró una base de datos en Supabase. Posteriormente, se creó un paquete modelo dentro del proyecto de Spring, en el cual se incluyeron las clases previamente definidas en los diagramas. Luego, se aplicaron las anotaciones propias de JPA, tales como @Table, @Column, @OneToOne y @OneToMany. Con estas configuraciones completadas, se estableció la conexión con la base de datos mediante la anotación @Primary DataSource.

Con la conexión operativa, se definió la estructura del microservicio, desarrollando su correspondiente controlador para exponer las funcionalidades a través de solicitudes HTTP, así como su clase service encargada de procesar las operaciones. Tras finalizar la lógica de negocio, se realizaron pruebas unitarias que permitieron validar el correcto funcionamiento del servicio.

El siguiente paso fue la implementación del segundo microservicio, orientado al envío de notificaciones. Para ello, se creó una cuenta de Google destinada a la empresa, con el propósito de utilizar el servidor SMTP de Gmail. En el archivo de propiedades del proyecto se configuraron los atributos necesarios para la dependencia Spring Starter Mail. Luego, se definieron los paquetes correspondientes: uno para el modelo y otro para el servicio. En el paquete modelo se desarrolló la clase Notificación, que almacena información como el correo destinatario y el contenido del mensaje. Por su parte, en la clase service se implementó el método encargado de enviar las notificaciones.

Durante las pruebas del servicio de pedidos, se detectó un error que provocaba la duplicación de productos en la base de datos cada vez que se incluía en un pedido un producto ya existente. Para resolver esta situación, se diseñó un nuevo microservicio de registro de inventario. Este incluyó la creación de una segunda base de datos en Supabase, destinada a almacenar los productos junto con sus respectivos identificadores. En consecuencia, se modificó la API de pedidos para que, en lugar de guardar los productos directamente, almacenara únicamente sus identificadores.

Posteriormente, se estableció la conexión entre ambas bases de datos, de modo que al registrar un pedido, los productos se consultaran en el inventario utilizando una lógica de tipo FindOrCreate. Para gestionar esta integración, se desarrolló una clase denominada DataSourceConfig, la cual contiene la configuración de conexión para ambas bases de datos.

Finalmente, se integraron los servicios desarrollados con el módulo de notificaciones, permitiendo así el envío automático de correos electrónicos tanto al registrar un nuevo pedido como cuando un producto está próximo a agotarse.

Ejemplo de tabla

|  |  |  |
| --- | --- | --- |
| Columna 1 | Columna 2 | Columna 3 |
|  |  |  |
|  |  |  |
|  |  |  |

Tabla 1

Título de la tabla.

Fuente: Referencia a la fuente

Imagen que contiene portátil, persona, interior, ordenador

Descripción generada automáticamente

Imagen 1

Ejemplo de fotografía de referencia.

imagen: Freepik.com

# RESULTADOS

Expone los hallazgos y datos resultantes del análisis de datos presentados anteriormente en los métodos, los cuales deben tener coherencia y ética respecto al tratamiento de la pregunta de la investigación planteada.

En este espacio, se realiza la constatación de las hipótesis y resultados de los objetivos de investigación planteados en el estudio argumentando sus diferentes perspectivas/hallazgos.

Se recomienda:

* Evidenciar situaciones o hallazgos con argumentos en hechos facticos.
* No hacer de este espacio o sección una lista de situaciones
* No usar o insertar notas al pie, ni notas al final
* Usar subtítulos que aporten mejoras en la legibilidad del documento

# DISCUSIÓN Y CONCLUSIÓN

Interpreta los resultados de la investigación demostrando la relevancia de la investigación e innovación en el área de conocimiento.

Necesariamente, establece recomendaciones a nuevos estudios para su abordaje o continuidad de conformidad a los resultados encontrados.

# REFERENCIAS BIBLIOGRÁFICAS

|  |  |
| --- | --- |
| [1] | F. H. Vera-Rivera, H. Astudillo, and C. Gaona, “Desarrollo de aplicaciones basadas en microservicios: tendencias y desafíos de investigación,” *RISTI - Rev. Ibér. Sist. Tecnol. Inf.*, no. E23(2019), pp. 107–120, 2019. |

|  |  |
| --- | --- |
| [2] | C. A. Candela Uribe, “Modelos transaccionales avanzados como alternativa para la implementación de transacciones de larga duración en microservicios,” in *Encuentro Internacional de Educación en Ingeniería*, 2022. |

|  |  |
| --- | --- |
| [3] | “Vista de Arquitectura de Microservicios,” *Edu.co*. [Online]. Available: https://revistas.udistrital.edu.co/index.php/tia/article/view/9687/pdf. [Accessed: 21-Aug-2025]. |

|  |  |
| --- | --- |
| [4] | D. López and E. Maya, “Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web,” *59.13.30*. [Online]. Available: http://138.59.13.30/bitstream/10786/1277/1/93%20Arquitectura%20de%20Software%20basada%20en%20Microservicios%20para%20Desarrollo%20de%20Aplicaciones%20Web.pdf. [Accessed: 21-Aug-2025]. |

|  |  |
| --- | --- |
| [5] | *Edu.co*. [Online]. Available: https://bibliotecadigital.udea.edu.co/entities/publication/c8d9121d-ce47-4cdf-8c40-668b00018f8f. [Accessed: 21-Aug-2025]. |

|  |  |
| --- | --- |
| [6] | J. D. Pilamunga Valla and L. F. Angamarca Murquincho, “Microservicios para el módulo de proyectos del sistema de gestión de investigación de la Unach,” *Riobamba, Universidad Nacional de Chimborazo*, 27-May-2024. [Online]. Available: http://dspace.unach.edu.ec/handle/51000/13043. [Accessed: 21-Aug-2025]. |

|  |  |
| --- | --- |
| [7] | A. Mejía Ortíz and A. J. Santos Revelo, “Arquitectura computacional soportada en microservicios para aplicaciones de automatización industrial flexibles y escalables,” *Red UAO Home*, 06-Mar-2024. [Online]. Available: https://red.uao.edu.co/entities/publication/12785216-234a-441a-87b8-1eb52a752e54. [Accessed: 21-Aug-2025]. |

|  |  |
| --- | --- |
| [8] | L. F. Santacruz-Erraez and D. X. Poma-Japón, “Prototipo de Arquitectura de Microservicios para Sistemas Transaccionales Financieros con Keycloak,” *MQRInvestigar*, vol. 9, no. 1, p. e189, 2025. |

|  |  |
| --- | --- |
| [9] | A. R. C. Domínguez., “Sistema de gestión de productos y servicios para proveedores y consumidores empresariales utilizando la arquitectura de microservicios,” *Uci.cu*. [Online]. Available: https://repositorio.uci.cu/bitstream/123456789/10254/1/TD\_09492\_19.pdf. [Accessed: 21-Aug-2025]. |

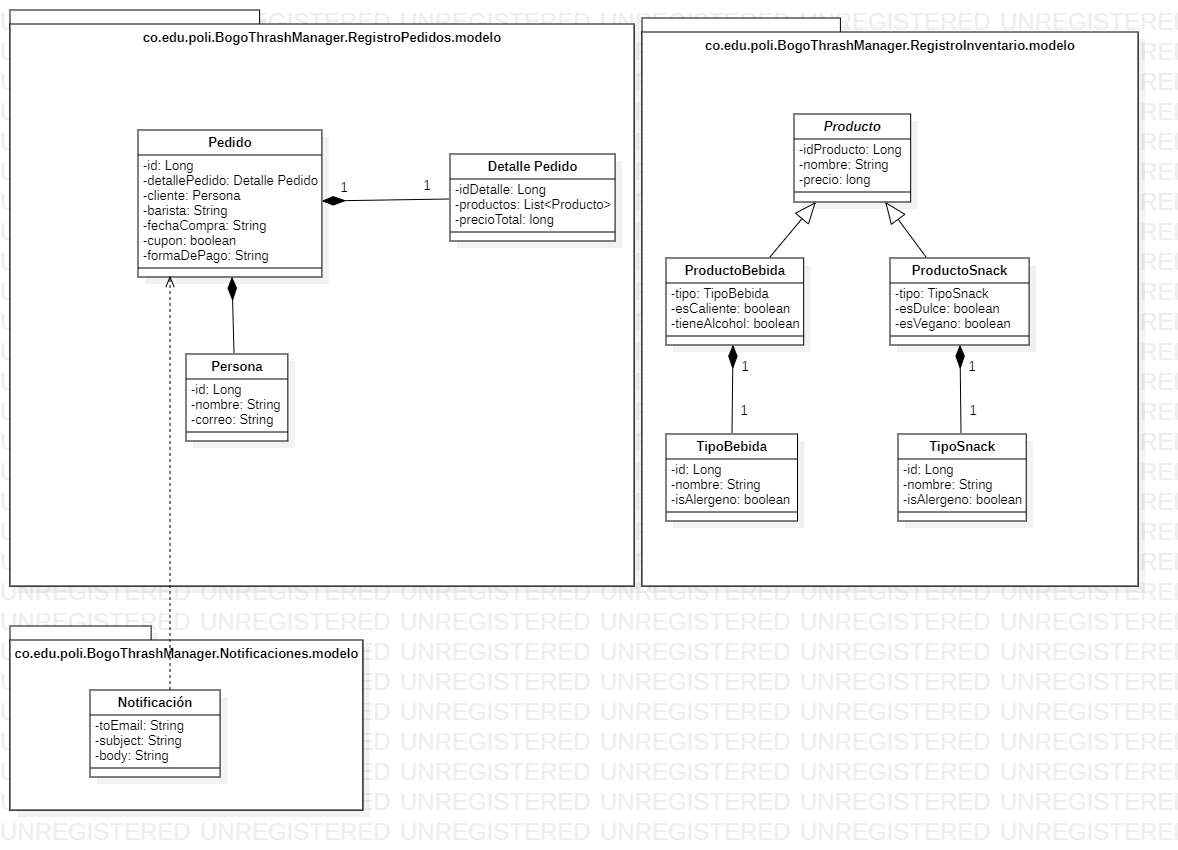
|  |  |
| --- | --- |
| [10] | *Ull.es*. [Online]. Available: https://riull.ull.es/xmlui/bitstream/handle/915/15475/Analisis%20de%20aplicaciones%20de%20arquitecturas%20monoliticas%20para%20su%20migracion%20a%20arquitecturas%20basadas%20en%20microservicios.pdf?sequence=1. [Accessed: 21-Aug-2025]. |

**LINK DE JIRA:**

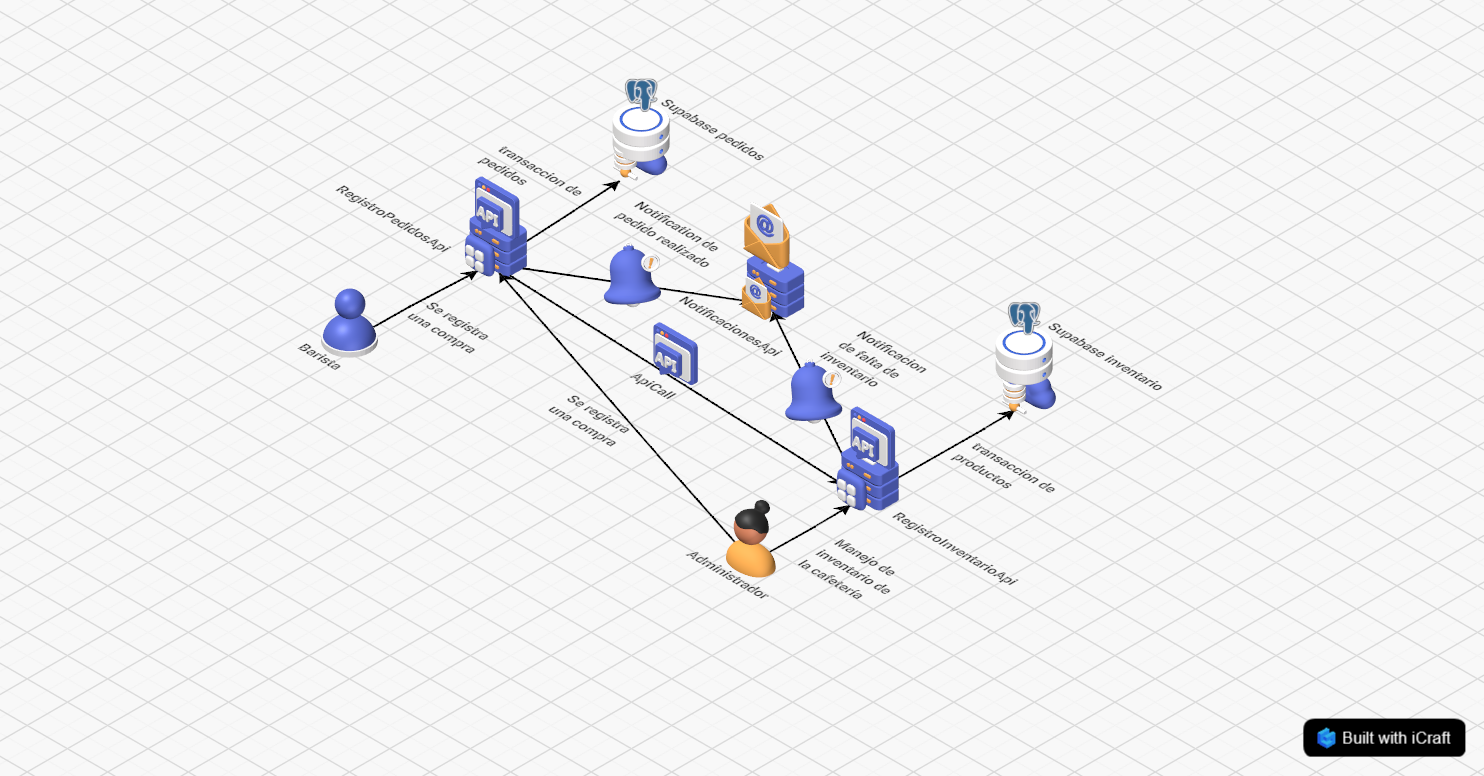
<https://proyectoingenieriasoftware2.atlassian.net/jira/software/projects/SCRUM/boards/1?atlOrigin=eyJpIjoiYjYwZGFlYzA2ZTRiNGY2YTg0YTUyNTc0YzY5ZjY1MDQiLCJwIjoiaiJ9>

**DIAGRAMAS:**

**Diagrama de clases**

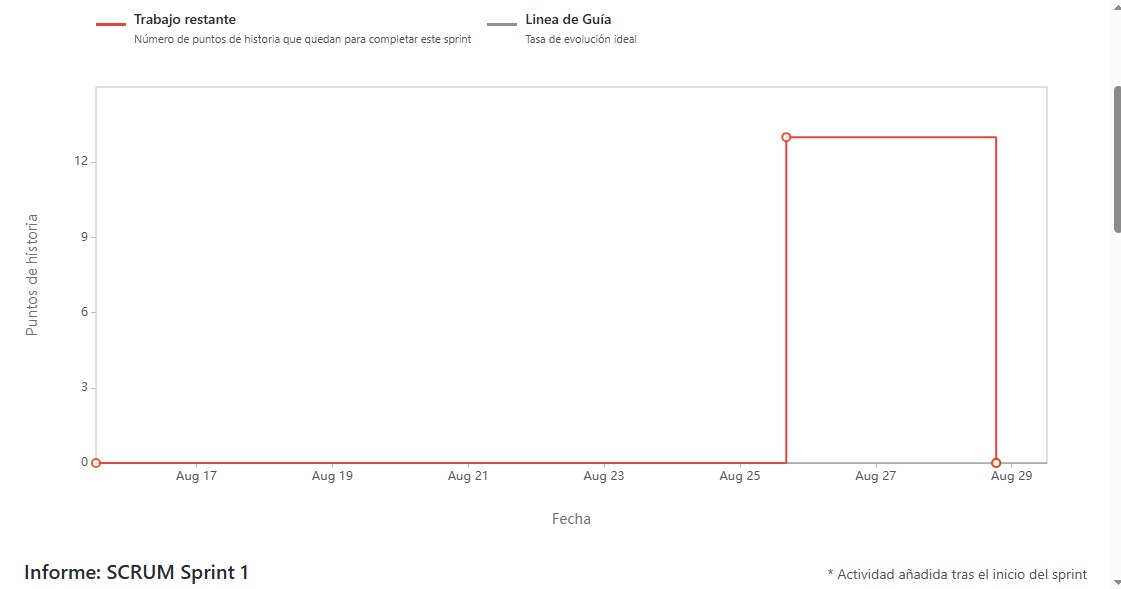


**Diagrama de arquitectura**



**IMAGENES JIRA:**

**Sprint (lo visto en este documento es lo realizado durante el primer sprint)**



**Backlog después del primer sprint:**

A screenshot of a computer

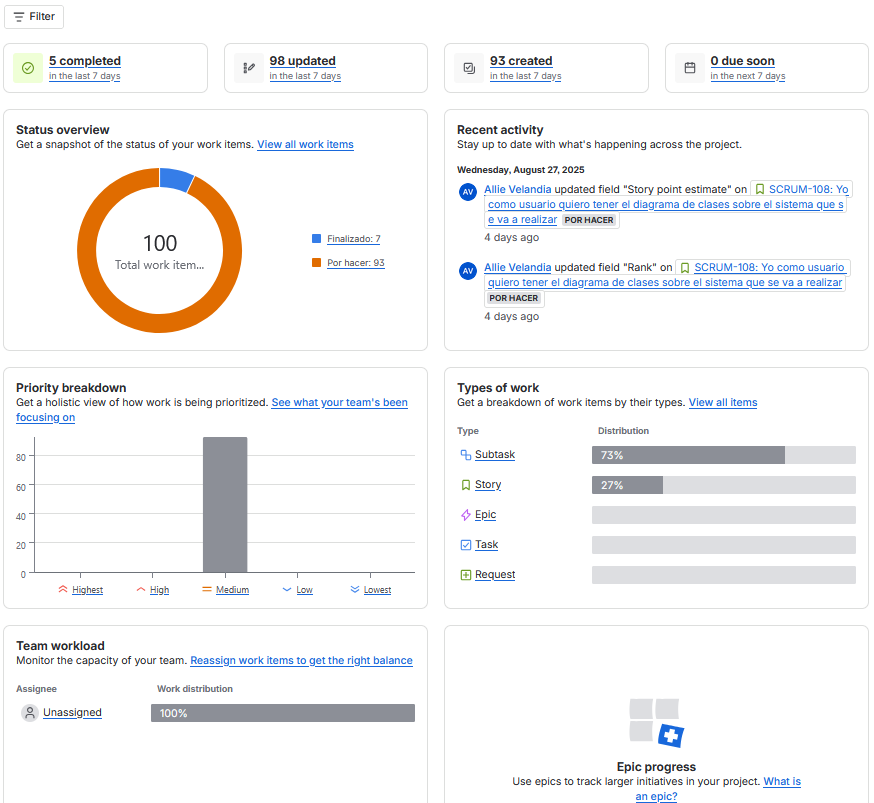
AI-generated content may be incorrect.

**Integrantes del equipo**

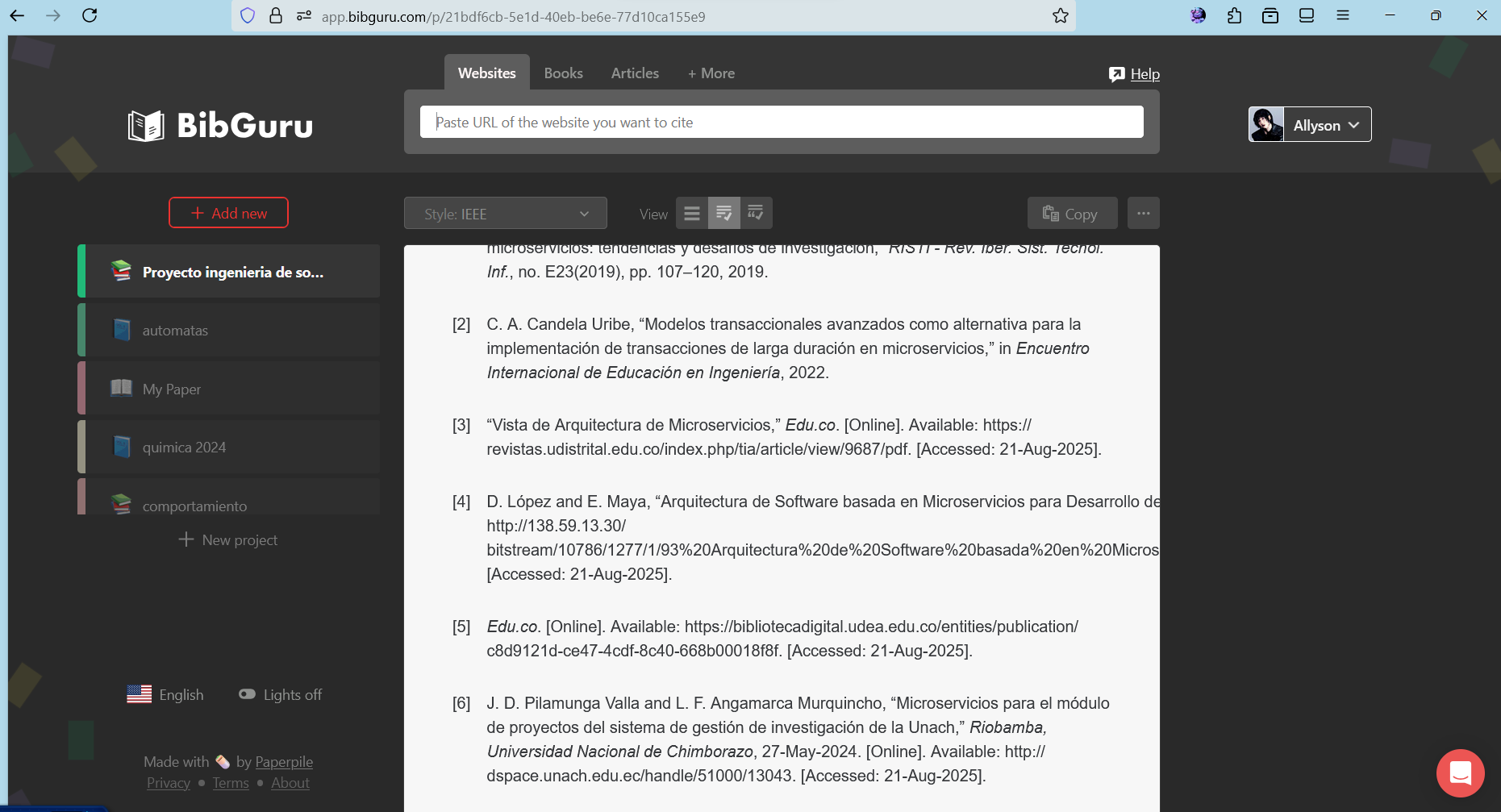
A screenshot of a computer

AI-generated content may be incorrect.

**Summary**



**EVIDENCIA DE USO DE GESTOR DE REFERENCIAS (BibGuru es el gestor de referencias utilizado)**

****

**Sprint 2  
A graph on a white background

AI-generated content may be incorrect.  
Sprint 3  
A graph on a white background

AI-generated content may be incorrect.**

**Sprint 4**

**A graph with a red line

AI-generated content may be incorrect.**

**Documentacion swagger  
A screenshot of a computer

AI-generated content may be incorrect.**