EthAIca. 2024; 3:132
doi: 10.56294/ai2024132

ORIGINAL

Prototyping and Validation of a Low-Code Platform for Dynamic Code Generation in Microservices

Prototipo y Validación de una Plataforma Low-Code para Microservicios con Generación Dinámica de Código

Tomás Darquier[1], Pablo Alejandro Virgolini[1]

[1]Universidad Siglo 21, Licenciatura en Informática, S.C de Bariloche. Argentina.

Citar como: Darquier T, Virgolini PA. Prototyping and Validation of a Low-Code Platform for Dynamic Code Generation in Microservices. EthAIca. 2024; 3:132. https://doi.org/10.56294/ai2024132

Enviado: 16-08-2023 **Revisado:** 04-01-2024 **Aceptado:** 24-05-2024 **Publicado:** 25-05-2024

Editor: PhD. Rubén González Vallejo

ABSTRACT

Introduction: the project addressed the issue of developing microservice architectures, a practice increasingly adopted in the software industry due to its benefits in terms of scalability and maintenance. However, its implementation involves a high degree of technical complexity, especially in the design, integration, and deployment stages. Given this scenario, the development of a low-code web platform was proposed to facilitate the visual design of microservices, reducing development times and technical barriers.
Method: to carry out the proposal, the agile Scrum methodology was applied, allowing for iterative and incremental construction of the system. The platform was developed with technologies such as Java and Spring Framework in the backend, HTML, CSS, JavaScript, and Thymeleaf in the frontend, and PostgreSQL as the database. Apache Kafka was incorporated for asynchronous communication, MinIO for storage, and semantic technologies such as RDF and SPARQL managed by Apache Jena. Code generation was performed with Apache Velocity, based on predefined templates.
Results: the system allowed users to design microservice architectures using a visual interface (canvas), configure specific properties, and automatically generate source code. In addition, it incorporated validations that ensured the consistency of the designs and offered mechanisms for authentication and export of the generated code.
Conclusions: the platform achieved its goal of simplifying microservice development through a low-code approach. Its usefulness as a support tool for developers was validated, reducing complexity and time spent on repetitive technical tasks.

Keywords: Microservices; Low-Code; Code Generation; Distributed Architecture; Agile Development.

RESUMEN

Introducción: el proyecto abordó la problemática del desarrollo de arquitecturas de microservicios, una práctica cada vez más adoptada en la industria del software por sus beneficios en escalabilidad y mantenimiento. Sin embargo, su implementación implica una alta complejidad técnica, especialmente en las etapas de diseño, integración y despliegue. Frente a este escenario, se planteó el desarrollo de una plataforma web Low-code que facilitara el diseño visual de microservicios, reduciendo tiempos de desarrollo y barreras técnicas.
Método: para llevar a cabo la propuesta, se aplicó la metodología ágil Scrum, permitiendo una construcción iterativa e incremental del sistema. La plataforma se desarrolló con tecnologías como Java y Spring Framework en el backend, HTML, CSS, JavaScript y Thymeleaf en el frontend, y PostgreSQL como base de datos. Se incorporaron Apache Kafka para comunicación asincrónica, MinIO para almacenamiento, y tecnologías semánticas como RDF y SPARQL gestionadas por Apache Jena. La generación de código se realizó con Apache Velocity, en base a plantillas predefinidas.

© 2024; Los autores. Este es un artículo en acceso abierto, distribuido bajo los términos de una licencia Creative Commons (https://creativecommons.org/licenses/by/4.0) que permite el uso, distribución y reproducción en cualquier medio siempre que la obra original sea correctamente citada

Resultados: el sistema permitió a los usuarios diseñar arquitecturas de microservicios mediante una interfaz visual (canvas), configurar propiedades específicas y generar código fuente automáticamente. Además, incorporó validaciones que aseguraron la consistencia de los diseños y ofreció mecanismos de autenticación y exportación del código generado.
Conclusiones: la plataforma logró cumplir su objetivo de simplificar el desarrollo de microservicios a través de un enfoque Low-code. Se validó su utilidad como herramienta de apoyo para desarrolladores, reduciendo la complejidad y el tiempo invertido en tareas técnicas repetitivas.

Palabras clave: Microservicios; Low-Code; Generación de Código; Arquitectura Distribuida; Desarrollo Ágil.

INTRODUCCIÓN

El desarrollo de aplicaciones modernas plantea desafíos cada vez más complejos, especialmente cuando se adoptan arquitecturas distribuidas como los microservicios.[1,2,3,4] Este enfoque, aunque altamente beneficioso en términos de escalabilidad, modularidad y mantenimiento, introduce dificultades importantes durante el diseño, implementación e integración de los distintos componentes.[5,6,7,8] Frente a esta realidad, el presente proyecto propuso como solución el diseño e implementación de una plataforma Low-code orientada a facilitar la creación de arquitecturas de microservicios de forma visual, intuitiva y personalizable, apuntando especialmente a desarrolladores de software.[9,10,11]

Con el objetivo de lograr un producto funcional y flexible, se adoptó la metodología ágil **Scrum**, que permitió el desarrollo incremental del sistema a través de iteraciones conocidas como Sprints. Esta estrategia facilitó la incorporación de mejoras continuas y la rápida adaptación frente a obstáculos técnicos, cambios de tecnología o reorientaciones del diseño, sin comprometer el avance general del proyecto.[12,13,14]

Durante su implementación, el sistema integró una variedad de tecnologías organizadas en distintos niveles. Para el **Front-end**, se utilizó HTML5, CSS y JavaScript, complementados con Bootstrap y Thymeleaf, permitiendo una experiencia de usuario fluida y adaptable. En el **Back-end**, se empleó Java con Spring Framework para construir una base robusta y segura, con autenticación vía OAuth2 provista por Okta. También se incorporaron herramientas como PostgreSQL para la gestión de datos, Apache Kafka para la comunicación asíncrona, y MinIO para el almacenamiento de código generado.[15,16,17]

Una de las características más innovadoras del sistema fue la generación automática de código a partir de la interacción del usuario con el canvas. Esta funcionalidad fue posible gracias al uso de tecnologías semánticas como RDF y SPARQL, gestionadas mediante Apache Jena, y potenciadas por Apache Velocity para la creación de archivos personalizables.[18,19,20]

La recopilación de datos para el diseño y validación del proyecto combinó fuentes académicas especializadas y foros de discusión técnicos presentes en redes sociales como Reddit, Twitter y Medium. Esta fusión entre teoría y práctica permitió una mejor comprensión del problema y fundamentó las decisiones tecnológicas tomadas.

En resumen, la plataforma desarrollada busca disminuir la barrera técnica en el diseño de sistemas distribuidos, promoviendo el uso de componentes reutilizables y configurables, todo dentro de un entorno visual Low-code capaz de generar arquitecturas listas para desplegar con mínima intervención manual.

¿Cómo facilitar el diseño, configuración e integración de arquitecturas de microservicios mediante una plataforma Low-code, que reduzca la complejidad técnica y el tiempo de desarrollo sin comprometer la calidad del software generado?

Objetivo

Desarrollar una plataforma Low-code que permita a los desarrolladores diseñar, configurar e integrar arquitecturas de microservicios de forma visual e intuitiva, mediante el uso de componentes reutilizables, automatizando la generación de código y optimizando tiempos de desarrollo con validaciones que aseguren la calidad del sistema.

MÉTODO

Diseño Metodológico

Herramientas Metodológicas

Durante el desarrollo del presente sistema se siguieron los lineamientos establecidos por la metodología ágil Scrum, un marco de trabajo que facilita la colaboración entre equipos para entregar productos de manera iterativa e incremental, permitiendo adaptarse rápidamente a los cambios e incentivando la mejora continua.

De esta forma, al finalizar cada Sprint, se logró obtener porciones funcionales de código, aunque el sistema completo no esté desarrollado, aprendiendo en cada iteración y aplicando el conocimiento en la siguiente

a realizar. Así el producto se benefició de las características mencionadas, permitiendo incluso cambiar de tecnologías, en base al conocimiento obtenido durante el proceso de desarrollo sin represalia alguna.

Herramientas de Desarrollo

En el desarrollo del proyecto fueron utilizadas múltiples tecnologías, las cuales serán explicadas y justificadas a continuación, organizadas en 4 grupos según su actividad especifica: herramientas referidas al Front-end, aquellas destinadas al Back-end para autenticación lógica básica e intercambio con el Front-end embebido en el Gateway, por otra parte las tecnologías orientadas al Back-end para la generación automática de código y, finalmente, las tecnologías utilizadas para facilitar el despliegue y escalabilidad del sistema.

En cuanto a las herramientas empleadas en el desarrollo del Front-end se encuentra, el ya estándar conjunto de JavaScript, HTML5 y CSS, permitiendo crear un aspecto visual completo y con alta compatibilidad, además de una correcta y adaptable lógica de comunicación con el Back-end. A su vez, estas herramientas fueron potenciadas con Boostrap, que con la ayuda de sus componentes prediseñados alivianó la carga de trabajo referida al diseño estético, permitiendo gestionar múltiples funcionalidades dentro de la misma página sin descuidar este aspecto. De esta forma y con la ayuda de Thymeleaf el Front-end fue acoplado al API Gateway para evitar complejidades innecesarias.

El Back-end fue desarrollado en su gran mayoría en Java 17, haciendo uso del framework Spring y de sus conocidas librerías para una correcta, limpia y eficiente comunicación entre servicios mediante, en su mayoría, el estilo de arquitectura REST. La seguridad se administró e implementó mediante OAuth2 partiendo del servicio ofrecido por Okta, un estándar que permite a sitios web o aplicaciones acceder a recursos alojados en otras aplicaciones, en nombre y con permiso previo, del usuario. Los aspectos que requirieron persistencia de datos relacional fueron provistos de una instancia de base de datos PostgreSQL, elegida debido a su entorno open source, además de su destacable flexibilidad, integridad de datos y escalabilidad. Además, se manejaron comunicaciones asíncronas para él envió de notificaciones mediante Apache Kafka.

La lógica referida a la generación dinámica de código se administró semánticamente mediante RDF, que permitió especificar de forma correcta y estructurada las combinaciones y decisiones realizadas por el usuario en el canvas presente en el Front- end, para poder así ser enviadas al Back-end, consultadas a través del lenguaje de consulta SPARQL y administradas mediante Apache Jena en los servicios Java. Para concluir el proceso y generar el código especificado y solicitado por el usuario se decidió utilizar Apache Velocity, debido a su alta capacidad en la tarea requerida y MinIO, para el almacenamiento y descarga de los archivos resultantes, evadiendo de esta forma, dependencia a soluciones de almacenamiento en la nube específicas.

Finalmente, la administración del despliegue y escalabilidad del sistema se llevó a cabo mediante Docker, potenciado con Kubernetes. De esta forma, al igual que otras decisiones previamente expuestas, se reduce el acoplamiento a soluciones cloud brindando al sistema una mayor versatilidad y capacidad independiente de despliegue.

Recolección de Datos

Para comprender adecuadamente el problema a abordar, se inició con el análisis de fuentes bibliográficas académicas. Este enfoque permitió identificar los principales desafíos en el desarrollo de arquitecturas distribuidas, así como los costos asociados.

Otro método de recolección de datos utilizado fue el análisis de redes sociales. Esta metodología, debido al potencial público del proyecto, resultó altamente enriquecedora gracias a plataformas como Twitter, Reddit y Medium, donde desarrolladores de software comparten y discuten ideas y experiencias sobre temas y situaciones del ámbito informático.

De esta forma, se obtuvieron dos enfoques ciertamente antagónicos, justificando de esta manera la elección de las técnicas presentadas. Estos enfoques se dividen en los estrictamente teóricos, basados en documentos académicos, y por el otro extremo, mediante redes sociales, los enfoques basados en las experiencias diarias y cotidianas de los desarrolladores, quienes, al fin y al cabo, son los perjudicados por la problemática.

Planificación del Proyecto

Para facilitar la comprensión, se presentará primero la planificación de forma separada, y luego se mostrará el diagrama de Gantt utilizado para organizar los objetivos del presente Trabajo Final de Graduación.

A continuación, se presentan las tareas especificadas en conjunto con las fechas correspondientes para, posteriormente presentar el diagrama de Gantt mencionado.

Nombre de tarea	Fecha de inici	Fecha final
	07/08/2024	09/11/2024
Temática y Relevamiento	**07/08/2024**	05/09/2024
Elección y Presentación de Temática	07/08/2024	19/08/2024
Título, Introducción y Justificación	20/08/2024	22/08/2024
Objetivo General y Espeíficos	23/08/2024	25/08/2024
Marco Teórico Referencial	26/08/2024	28/08/2024
Diseño Metodologico	29/08/2024	31/08/2024
Relevamiento	01/09/2024	03/09/2024
Proceso de Negocios	04/09/2024	05/09/2024
Propuesta y Definición del Prototipo	**06/09/2024**	23/09/2024
Diagnóstico y Propuesta	06/09/2024	08/09/2024
Objetivos, Límites y Alcances	09/09/2024	11/09/2024
Descripción del Sistema	12/09/2024	15/09/2024
Estructura de Datos	16/09/2024	18/09/2024
Prototipos de Interfaces	19/09/2024	22/09/2024
Diagrama de Arquitectura	23/09/2024	23/09/2024
Riesgos y Control	**24/09/2024**	01/10/2024
Documentación de Seguridad	24/09/2024	25/09/2024
Análisis de Costos	26/09/2024	28/09/2024
Análisis de Riesgos	29/09/2024	01/10/2024
Finalización de Entregable	**02/10/2024**	06/10/2024
Conclusiones y Anexos	02/10/2024	04/10/2024
Revisión y Corrección General	05/10/2024	06/10/2024
Desarrollo y Pruebas del Prototipo	**23/09/2024**	09/11/2024
Desarrollo del Prototipo	23/09/2024	05/11/2024
Pruebas del Prototipo	02/11/2024	09/11/2024

Figura 1. Plan de Desarrollo

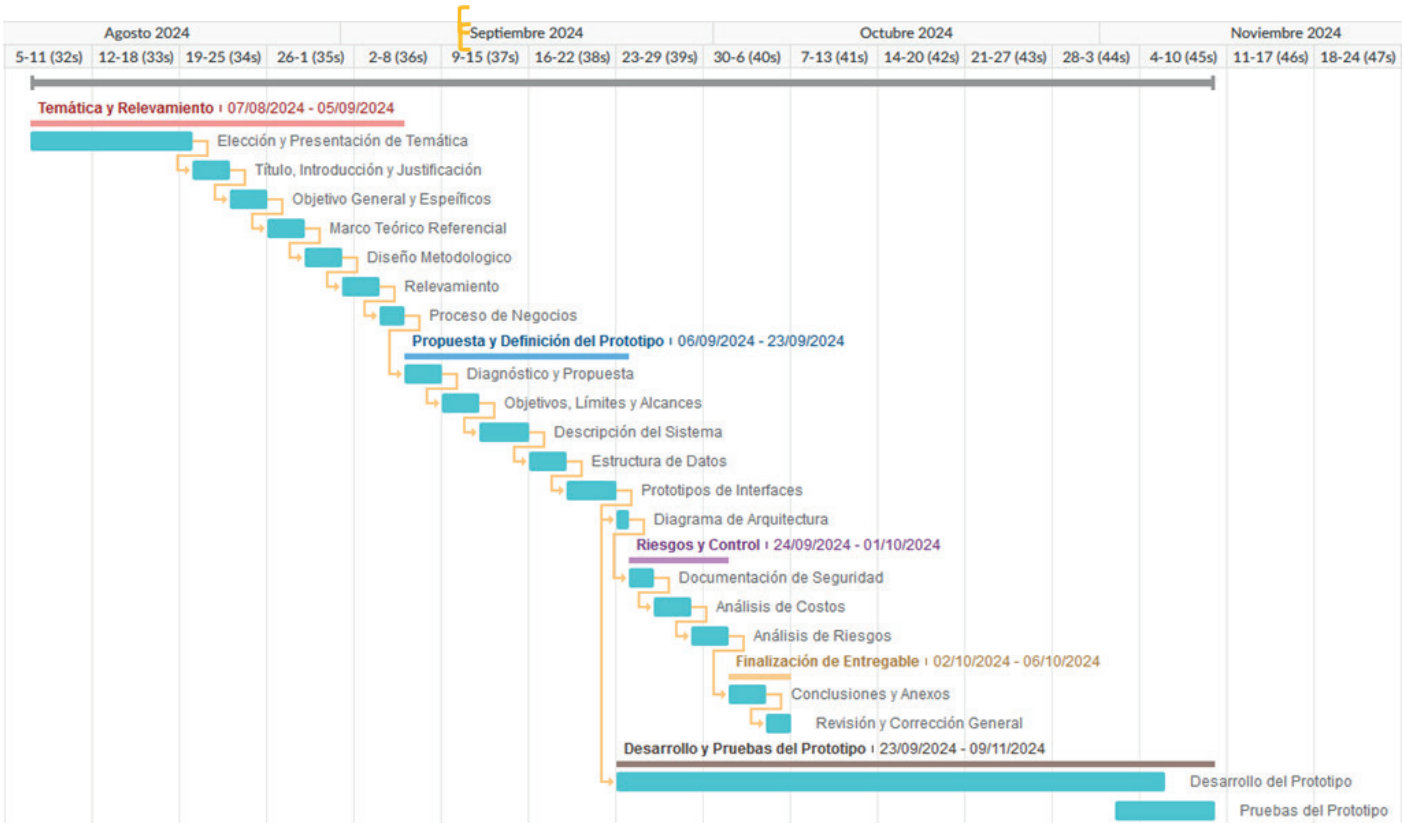

Figura 2. Diagrama de Gantt

https://doi.org/10.56294/ai2024132 ISSN: 3072-7952

RESULTADOS

Descripción del Sistema

Product Backlog

A continuación, se presenta el Product Backlog del prototipo a desarrollar. Cabe aclarar que las historias de usuario expuestas corresponden únicamente a las funcionalidades básicas necesarias para exponer el propósito del sistema. Es por este motivo que únicamente se presentan componentes personalizables del tipo ecommerce, ya que así se logran exponer las funcionalidades referidas a la combinación y personalización de componentes compatibles dentro del canvas de la aplicación.

Tabla 1. Product Backlog

ID	Historia de Usuario	Prioridad	Puntos de Historia	Dependencias
HU-001	Registro a la plataforma vía Google	Alta	5	-
HU-002	Iniciar sesión vía Google	Alta	2	HU-001
HU-003	Cerrar sesión de usuario	Media	2	HU-002
HU-004	Diseño visual drag-and-drop	Media	5	-
HU-005	Plantilla de servicio de usuarios	Alta	5	-
HU-006	Servicio de usuarios	Alta	2	HU-005
HU-007	Plantilla de servicio de catálogo de productos	Alta	5	-
HU-008	Servicio de catálogo de productos	Alta	3	HU-007
HU-009	Plantilla de servicio de carrito de compras	Alta	5	-
HU-010	Servicio de carrito de compras	Alta	3	HU-009
HU-011	Plantilla de servicio de ordenes	Alta	5	-
HU-012	Servicio de órdenes	Alta	3	HU-011
HU-013	Plantilla de servicio de envíos	Alta	5	-
HU-014	Servicio de envíos	Media	3	HU-013
HU-015	Plantilla de servicio de notificación	Alta	8	-
HU-016	Servicio de notificaciones	Media	3	HU-015
HU-017	Interactuar con componentes disponibles	Media	5	HU-006
HU-018	Compatibilidad de componentes	Media	5	HU-017
HU-019	Eliminar componentes del diseño	Baja	3	HU-018
HU-020	Seleccionar persistencia en componentes	Baja	3	HU-016
HU-021	Configurar endpoints en componentes	Media	3	HU-016
HU-022	Seleccionar metodología de comunicación	Media	8	HU-016
HU-023	Especificar seguridad JWT	Alta	3	HU-016
HU-024	Especificar configuración global	Media	5	HU-016
HU-025	Especificar API Gateway	Media	8	HU-016
HU-026	Traducción de JSON a RDF	Alta	5	HU-025
HU-027	ZIP de la arquitectura resultante	Baja	8	HU-026
HU-028	Estado de proceso de generación dinámica	Baja	5	HU-027
HU-029	Acceder a archivos ZIP	Baja	3	HU-027
HU-030	Dockerfiles de componentes	Alta	3	HU-026
HU-031	Guía de tareas pendientes	Baja	5	HU-025
HU-032	Documentación de componentes	Media	5	HU-025
HU-033	Guía de uso de la plataforma	Media	5	HU-032

Historias de Usuario

Tabla 2. HU-001 Registro a la plataforma vía Google

ID	**HU-001**	**Nombre**	**Registro a la plataforma vía Google**
Descripción		Como usuario, quiero registrarme en el sistema mediante mi cuenta de Google, para poder crear mi perfil.	
Criterios de Aceptación		Dado un usuario de Google existente, cuando este decida registrarse, el sistema debe redireccionarlo a una pestaña para brindar los permisos necesarios, permitiendo el registro e informando al usuario, una vez completada la acción. Dado un usuario de Google previamente registrado mediante su cuenta en la plataforma, cuando este sea provisto para registrarse, el sistema debe informar que ya existe una cuenta registrada con ese usuario, invitándolo a iniciar sesión mediante un hipervínculo a la sección correspondiente.	
Prioridad	Alta	Puntos de historia estimados	5

Tabla 3. HU002 Iniciar sesión vía Google

ID	**HU-002**	**Nombre**	**Iniciar sesión vía Google**
Descripción		Como usuario, quiero iniciar sesión en el sistema mediante mi cuenta de Google, para acceder a la plataforma y a mis arquitecturas previas.	
Criterios de Aceptación		Dado un usuario previamente registrado mediante su cuenta de Google, cuando ingrese mediante la cuenta mencionada, el sistema debe otorgar acceso, redirigiéndolo al canvas de la plataforma. Dado un usuario que no se encuentre registrado mediante su cuenta de Google, cuando intente ingresar mediante la cuenta no registrada, el sistema debe informar que no se encuentra registrado, invitándolo a hacerlo mediante un hipervínculo a la sección correspondiente.	
Prioridad	Alta	Puntos de historia estimados	2

Tabla 4. HU-003 Cerrar sesión de usuario

ID	**HU-003**	**Nombre**	**Cerrar sesión de usuario**
Descripción		Como usuario, quiero cerrar sesión en el sistema mediante mi cuenta de Google, para evitar accesos no deseados a mi perfil.	
Criterios de Aceptación		Dado un usuario ingresado en el sistema mediante credenciales validas, cuando decida errar su sesión, el sistema debe anular las credenciales temporales que permiten su acceso en la sesión actual.	
Prioridad	Media	Puntos de historia estimados	2

Tabla 5. HU-004 Diseño visual drag-and-drop

ID	**HU-004**	**Nombre**	**Diseño visual drag-and-drop**
Descripción		Como usuario, quiero arrastrar componentes al canvas para poder diseñar una arquitectura de forma visual.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión, cuando el usuario presione y arrastre al canvas la representación visual de un componente, el sistema debe moverlo al área donde el usuario deposite el elemento.	
Prioridad	Media	Puntos de historia estimados	5

Tabla 6. HU-005 Plantilla de servicio de usuarios

ID	**HU-005**	**Nombre**	**Plantilla de Servicio de Usuarios**
Descripción		Como usuario, quiero que el sistema utilice una plantilla predefinida para la creación automática de servicios de usuarios, de manera que se generen todos los componentes necesarios para gestionar usuarios y permisos sin intervención manual.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, cuando genere una arquitectura que contenga “Servicio de Usuarios” dentro del canvas de diseño, el sistema debe asociar automáticamente el servicio a una plantilla predefinida para la gestión de usuarios. Dado un usuario que ha iniciado sesión en la plataforma, cuando combine cierto servicio compatible a “Servicio de Usuarios”, el sistema debe adaptar la generación de código de forma acorde para una correcta interacción.	
Prioridad	Alta	Puntos de historia estimados	5

Tabla 7. HU-006 Servicio de usuarios			
ID	**HU-006**	**Nombre**	**Servicio de usuarios**
Descripción		Como usuario, quiero disponer de un servicio de manejo de usuarios para incorporar en mi arquitectura, para poder manejar su información y permisos.	
Criterios de Aceptación		Dado que un usuario ha iniciado sesión en la plataforma, cuando arrastre el componente visual "Usuarios" al canvas de diseño, el sistema debe asociar automáticamente el servicio de órdenes a la plantilla de código predefinida.	
Prioridad		Alta	Puntos de historia estimados 3

Tabla 8. HU-007 Plantilla de servicio de catálogo de productos			
ID	**HU-007**	**Nombre**	**Plantilla de Servicio de catálogo de productos**
Descripción		Como usuario, quiero que el sistema utilice una plantilla predefinida para la creación automática de un servicio de catálogo de productos, para que pueda exponer mi inventario actual.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, cuando genere una arquitectura que contenga "Servicio de Catálogo de Productos" dentro del canvas de diseño, el sistema debe asociar automáticamente el servicio a una plantilla predefinida para la gestión de usuarios. Dado un usuario que ha iniciado sesión en la plataforma, cuando combine cierto servicio compatible a "Servicio de Catálogo de Productos", el sistema debe adaptar la generación de código de forma acorde para una correcta interacción.	
Prioridad		Alta	Puntos de historia estimados 5

Tabla 9. HU-008 Servicio de catálogo de productos			
ID	**HU-008**	**Nombre**	**Servicio de catálogo de productos**
Descripción		Como usuario, quiero disponer de un servicio de catálogo de productos para incorporar en mi arquitectura, para poder exponer mi stock actual al público.	
Criterios de Aceptación		Dado que un usuario ha iniciado sesión en la plataforma, cuando arrastre el componente visual "Catálogo de Productos" al canvas de diseño, el sistema debe asociar automáticamente el servicio de órdenes a la plantilla de código predefinida.	
Prioridad		Alta	Puntos de historia estimados 3

Tabla 10. HU-009 Plantilla de servicio carrito de compras			
ID	**HU-009**	**Nombre**	**Plantilla de Servicio de carrito de compras**
Descripción		Como usuario, quiero que el sistema utilice una plantilla predefinida para la creación automática de un servicio de carrito de compras, para que los usuarios puedan realizar compras de múltiples productos.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, cuando genere una arquitectura que contenga "Servicio de Carrito de Compras" dentro del canvas de diseño, el sistema debe asociar automáticamente el servicio a una plantilla predefinida para la gestión de usuarios. Dado un usuario que ha iniciado sesión en la plataforma, cuando combine cierto servicio compatible a "Servicio de Carrito de Compras", el sistema debe adaptar la generación de código de forma acorde para una correcta interacción.	
Prioridad		Alta	Puntos de historia estimados 5

Tabla 11. HU-010 Servicio de carrito de compras			
ID	**HU-010**	**Nombre**	**Servicio de carrito de compras**
Descripción		Como usuario, quiero disponer de un servicio de carrito de compras para incorporar en mi arquitectura, para que los usuarios de mi sistema puedan realizar compras de múltiples artículos simultáneamente.	
Criterios de Aceptación		Dado que un usuario ha iniciado sesión en la plataforma, cuando arrastre el componente visual "Carrito de Compras" al canvas de diseño, el sistema debe asociar automáticamente el servicio de órdenes a la plantilla de código predefinida.	
Prioridad		Alta	Puntos de historia estimados 3

Tabla 12. HU-011 Plantilla de servicio de órdenes

ID	**HU-011**	**Nombre**	**Plantilla de Servicio de órdenes**
Descripción		Como usuario, quiero que el sistema utilice una plantilla predefinida para la creación automática de un servicio de órdenes, para gestionar el proceso de emisión y administración de órdenes.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, cuando genere una arquitectura que contenga "Servicio de Ordenes" dentro del canvas de diseño, el sistema debe asociar automáticamente el servicio a una plantilla predefinida para la gestión de usuarios. Dado un usuario que ha iniciado sesión en la plataforma, cuando combine cierto servicio compatible a "Servicio de Ordenes", el sistema debe adaptar la generación de código de forma acorde para una correcta interacción.	
Prioridad		Alta	Puntos de historia estimados 5

Tabla 13. HU-012 Servicio de órdenes

ID	**HU-012**	**Nombre**	**Servicio de órdenes**
Descripción		Como usuario, quiero disponer de un servicio capaz de administrar órdenes para incorporar a mi arquitectura, para manejar las ordenes emitidas dentro del sistema y actuar en consecuencia.	
Criterios de Aceptación		Dado que un usuario ha iniciado sesión en la plataforma, cuando arrastre el componente visual "Órdenes" al canvas de diseño, el sistema debe asociar automáticamente el servicio de órdenes a la plantilla de código predefinida.	
Prioridad		Alta	Puntos de historia estimados 3

Tabla 14. HU-013 Plantilla de servicio de envíos

ID	**HU-013**	**Nombre**	**Plantilla de Servicio de envíos**
Descripción		Como usuario, quiero que el sistema utilice una plantilla predefinida para la creación automática de un servicio de envíos, de manera que pueda gestionar el estado de los envíos.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, cuando genere una arquitectura que contenga "Servicio de Envíos" dentro del canvas de diseño, el sistema debe asociar automáticamente el servicio a una plantilla predefinida para la gestión de usuarios. Dado un usuario que ha iniciado sesión en la plataforma, cuando combine cierto servicio compatible a "Servicio de Envíos", el sistema debe adaptar la generación de código de forma acorde para una correcta interacción.	
Prioridad		Alta	Puntos de historia estimados 5

Tabla 15. HU-014 Servicio de envíos

ID	**HU-014**	**Nombre**	**Servicio envíos**
Descripción		Como usuario, quiero disponer de un servicio capaz de administrar el estado de los envíos de mi sistema para incorporar a mi arquitectura, para poder rastrear y actualizar el estado de los envíos.	
Criterios de Aceptación		Dado que un usuario ha iniciado sesión en la plataforma, cuando arrastre el componente visual "Envíos" al canvas de diseño, el sistema debe asociar automáticamente el servicio de órdenes a la plantilla de código predefinida.	
Prioridad		Media	Puntos de historia estimados 3

Tabla 16. HU-015 Plantilla de servicio de notificaciones

ID	**HU-015**	**Nombre**	**Plantilla de Servicio de notificaciones**
Descripción		Como usuario, quiero que el sistema utilice una plantilla predefinida para la creación automática de un servicio de notificaciones, de manera que pueda enviar alertas y mensajes a los usuarios.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, cuando genere una arquitectura que contenga "Servicio de Notificaciones" dentro del canvas de diseño, el sistema debe asociar automáticamente el servicio a una plantilla predefinida para la gestión de usuarios. Dado un usuario que ha iniciado sesión en la plataforma, cuando combine cierto servicio compatible a "Servicio de Notificaciones", el sistema debe adaptar la generación de código de forma acorde para una correcta interacción.	
Prioridad		Alta	Puntos de historia estimados 8

Tabla 17. HU-016 Servicio de notificaciones			
ID	**HU-016**	**Nombre**	**Servicio de notificaciones**
Descripción	Como usuario, quiero disponer de un servicio de notificaciones para incorporar en mi arquitectura, para enviar mensajes y alertas a los usuarios del sistema sobre eventos importantes.		
Criterios de Aceptación	Dado que un usuario ha iniciado sesión en la plataforma, cuando arrastre el componente visual "Notificaciones" al canvas de diseño, el sistema debe asociar automáticamente el servicio de órdenes a la plantilla de código predefinida.		
Prioridad	Media	Puntos de historia estimados	3

Tabla 18. HU-017 Visualizar componentes disponibles			
ID	**HU-017**	**Nombre**	**Interactuar con componentes disponibles**
Descripción	Como usuario, quiero visualizar los componentes disponibles para poder seleccionarlos y combinarlos en el canvas según mis necesidades.		
Criterios de Aceptación	Dado un usuario que ha iniciado sesión en la plataforma, cuanto deposite un componente en el canvas y lo desee unir con otro compatible, el sistema debe crear una unión grafica mediante una línea para expresar la relación exitosa. Dado un usuario que ha iniciado sesión en la plataforma, cuando deposite un componente en el canvas y lo desee unir con otro no compatible, el sistema debe presentar mediante "pop up" que la combinación no es legal, invitando al usuario a revisar la documentación de la plataforma.		
Prioridad	Media	Puntos de historia estimados	5

Tabla 19. HU-018 Compatibilidad de componentes			
ID	**HU-018**	**Nombre**	**Compatibilidad de componentes**
Descripción	Como usuario, quiero ver con que componentes es compatible el seleccionado, para combinarlos correctamente en el canvas.		
Criterios de Aceptación	Dado un usuario que ha iniciado sesión en la plataforma, cuando interactúe sobre el botón "+" anexado al componente presente en el canvas, el sistema debe destacar los botones "+" del resto de componentes compatibles presentes en el canvas. Dado un usuario que ha iniciado sesión en la plataforma, cuando destaque las compatibilidades de un componente y vuelva a tocar el canvas, el sistema debe desmarcar las conexiones previamente remarcadas.		
Prioridad	Media	Puntos de historia estimados	5

Tabla 20. HU-019 Eliminar componentes del diseño			
ID	**HU-019**	**Nombre**	**Eliminar componentes del diseño**
Descripción	Como usuario, quiero eliminar un componente del canvas para quitarlo de la arquitectura diseñada.		
Criterios de Aceptación	Dado un usuario que ha iniciado sesión en la plataforma, cuando ubique el ratón sobre un componente desplegado en el canvas, el sistema debe presentar un icono "X" referido a la opción de eliminar el componente. Dado un usuario que ha iniciado sesión en la plataforma, cuando interactúe con el botón "X" de un componente en el canvas, el sistema debe consultar si desea eliminar el componente, en caso positivo ejecuta la acción y en caso negativo quita el pop-up de consulta.		
Prioridad	Baja	Puntos de historia estimados	3

Tabla 21. HU-020 Seleccionar persistencia en componentes			
ID	**HU-020**	**Nombre**	**Seleccionar persistencia en componentes**
Descripción	Como usuario, quiero configurar la base de datos de un componente para poder elegir la que mejor se adapta a los requerimientos de mi arquitectura.		
Criterios de Aceptación	Dado un usuario que ha iniciado sesión en la plataforma, cuando interactúe con la base de datos predeterminada adjunta al componente dentro del canvas, el sistema debe desplegar las opciones alternativas presentes para el servicio, permitiendo modificar la predeterminada.		
Prioridad	Baja	Puntos de historia estimados	3

Tabla 22. HU-021 Configurar endpoints en componentes

ID	**HU-021**	**Nombre**	**Configurar endpoints en componentes**
Descripción		Como usuario, quiero configurar la URL de los endpoints de un componente para adecuarlo a mis requerimientos de infraestructura.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, cuando deposite el ratón sobre un componente en el canvas, el sistema debe desplegar un botón representando una "tuerca" para acceder a un pop-up con las configuraciones del servicio. Dado un usuario que ha iniciado sesión en la plataforma, cuando interactúe con el botón de configuración de un microservicio, el sistema debe presentar la opción de modificar el prefijo de los endpoints del microservicio, aplicando los cambios mediante un botón "aceptar". Dado un usuario que ha iniciado sesión en la plataforma, cuando ingrese una cadena no valida como URL como prefijo de endpoints y presione "aceptar", el sistema debe notificar que no se realizó ningún cambio debido a caracteres inválidos, manteniendo al usuario en el pop-up mostrando la cadena previa al ingreso ilegal.	
Prioridad		Media	Puntos de historia estimados 3

Tabla 23. HU-022 Seleccionar metodología de comunicación

ID	**HU-022**	**Nombre**	**Seleccionar metodología de comunicación**
Descripción		Como usuario, quiero seleccionar la metodología de comunicación entre dos componentes para que se adapten a las necesidades de mi sistema.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al crear una conexión entre 2 componentes compatibles, el sistema debe presentar en medio de la conexión un símbolo que al interactuar con el desplegara una lista con los tipos de conexión compatibles entre ambos microservicios permitiendo seleccionar la opción deseada.	
Prioridad		Media	Puntos de historia estimados 8

Tabla 24. HU-023 Especificar seguridad JWT

ID	**HU-023**	**Nombre**	**Especificar seguridad JWT**
Descripción		Como usuario, quiero especificar la utilización de JWT como medio de seguridad para que la seguridad de la arquitectura se ajuste a mis requerimientos.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al acceder al canvas, el sistema debe exponer en la esquina superior derecha una opción global para asegurar la arquitectura mediante JWT, permitiendo desactivarlo o activarlo.	
Prioridad		Alta	Puntos de historia estimados 3

Tabla 25. HU-024 Especificar configuración global

ID	**HU-024**	**Nombre**	**Especificar configuración global**
Descripción		Como usuario, quiero especificar la utilización de un servidor de configuración global para que la arquitectura se ajuste a mis requerimientos.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al acceder al canvas, el sistema debe exponer en la esquina superior derecha, debajo de "JWT" una opción para administrar la totalidad de las configuraciones de los servicios mediante un configuration server, permitiendo desactivarlo o activarlo.	
Prioridad		Media	Puntos de historia estimados 5

Tabla 26. HU-025 Especificar API Gateway

ID	**HU-025**	**Nombre**	**Especificar API Gateway**
Descripción		Como usuario, quiero especificar la generación de un API Gateway para que los componentes interactúen de forma directa	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al acceder al canvas, el sistema debe exponer en la esquina superior derecha, debajo de "Config Server" una opción para administrar la totalidad de las interacciones de la arquitectura mediante un API gateway, permitiendo desactivarlo o activarlo.	
Prioridad		Media	Puntos de historia estimados 8

Tabla 27. HU-026 Traducción de JSON a RDF			
ID	**HU-026**	**Nombre**	**Traducción de JSON a RDF**
Descripción		Como usuario, quiero que el sistema convierta automáticamente los datos que especifican la arquitectura generada de formato JSON a RDF para garantizar la una especificación estructurada.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al solicitar el procesamiento de la arquitectura resultante, el sistema debe representar la información mediante un JSON, traduciéndola posteriormente en contenido semántico RDF para brindar estructura y posibilidad de realizar consultas.	
Prioridad		Alta	Puntos de historia estimados 5

Tabla 28. HU-027 ZIP de la arquitectura resultante			
ID	**HU-027**	**Nombre**	**ZIP de la arquitectura resultante**
Descripción		Como usuario, quiero solicitar un archivo ZIP desde el canvas con la arquitectura diseñada para obtener los archivos que la componen y así configurarlo en mi entorno local.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al configurar una arquitectura y solicitar su generación, el sistema debe convertir el código generado en un archivo ZIP, informando al usuario cuando se haya completado el proceso, invitándolo a descargarlo desde su perfil de usuario.	
Prioridad		Baja	Puntos de historia estimados 8

Tabla 29. HU-028 Estado de proceso de generación dinámica			
ID	**HU-028**	**Nombre**	**Estado de proceso de generación dinámica**
Descripción		Como usuario, quiero conocer en qué estado se encuentra la generación dinámica de mi arquitectura para estimar cuando concluirá.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al configurar una arquitectura y solicitar su generación, el sistema debe presentar un pop-up indicando al usuario en qué estado se encuentra el proceso, mostrando los estados: Creando los pom.xml Generando el código Comprimiendo la arquitectura Proceso finalizado	
Prioridad		Baja	Puntos de historia estimados 5

Tabla 30. HU-029 Acceder a archivos ZIP previos			
ID	**HU-029**	**Nombre**	**Acceder a archivos ZIP**
Descripción		Como usuario, quiero acceder a mis archivos ZIP generados previamente para poder volver a descargarlos	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al seleccionar el símbolo de perfil presente en la esquina superior derecha, el sistema debe redireccionar al usuario a una pantalla donde se presentan sus últimas 5 arquitecturas generadas, permitiendo descargarlas con un simple clic en el botón adosado a la derecha de cada una.	
Prioridad		Baja	Puntos de historia estimados 3

Tabla 31. HU-030 Dockerfiles de componentes			
ID	**HU-030**	**Nombre**	**Dockerfiles de componentes**
Descripción		Como usuario, quiero disponer de Dockerfiles asociados a los componentes de la arquitectura para facilitar su despliegue.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al obtener el zip resultante de una arquitectura, el sistema debe incluir dentro de ese archivo, un Dockerfile individual por cada componente de la arquitectura, para permitir su despliegue y el de su componente de persistencia.	
Prioridad		Alta	Puntos de historia estimados 3

Tabla 32. HU-031 Guía de tareas pendientes			
ID	**HU-031**	**Nombre**	**Guía de tareas pendientes**
Descripción		Como usuario, quiero obtener una guía de tareas pendientes (TO-DO's) en el código generado para ajustar y personalizar la arquitectura descargada.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al obtener el zip resultante de una arquitectura, el sistema debe incluir dentro de ese archivo, un archivo .txt explicando los TO-DO's presentes en el código y su motivo de existencia.	
Prioridad		Baja	Puntos de historia estimados 5

Tabla 33. HU-032 Documentación de componentes			
ID	**HU-032**	**Nombre**	**Documentación de componentes**
Descripción		Como usuario, quiero acceder a la documentación de componentes para entender su estructura y funcionamiento detallado.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al presionar el apartado presente referido a la documentación en la zona superior de la pantalla, el sistema debe redireccionarlo a una pantalla con un menú lateral donde se listen los componentes disponibles. Dado un usuario que ha iniciado sesión en la plataforma, al presionar un componente especifico en la documentación, el sistema debe exponer la misma, especificando su funcionalidad, compatibilidades y endpoints.	
Prioridad		Media	Puntos de historia estimados 5

Tabla 34. HU-033 Guía de uso de la plataforma			
ID	**HU-033**	**Nombre**	**Guía de uso de la plataforma**
Descripción		Como usuario, quiero acceder a una guía de uso de la plataforma para comprender su funcionamiento general y sacarle el máximo provecho.	
Criterios de Aceptación		Dado un usuario que ha iniciado sesión en la plataforma, al presionar el apartado presente referido a la instrucción de uso de la plataforma en la zona superior de la pantalla, el sistema debe dirigirlo a la pantalla correspondiente, donde se presente una breve guía de uso de la plataforma.	
Prioridad		Media	Puntos de historia estimados 5

Sprint Backlog

En la siguiente tabla, se presenta el Backlog referido al primer Sprint del prototipo a desarrollar con una duración establecida de 14 días.

Tabla 35. Sprint Backlog

Sprint	Historia de Usuario	Tareas	Prioridad	Estimado (días)	Estado
1	Registro a la plataforma vía Google (HU-001)	Definir e implementar metodología y estructura de persistencia de datos	Alta	1	Hecho
1		Codificar e integrar módulo de registro y conexión con Google OAUTH2	Alta	2	Hecho
1		Diseñar interfaz grafica	Media	1	Hecho
1		Realizar testing unitario al módulo desarrollado	Alta	1	Hecho
1	Iniciar sesión vía Google (HU-002)	Codificar e integrar módulo de inicio de sesión	Alta	1	Hecho
1		Diseñar interfaz grafica	Media	½	Hecho
1		Realizar testing unitario al módulo desarrollado	Alta	½	Hecho
1	Cerrar sesión de usuario (HU-003)	Codificar e integrar módulo de cierre de sesión	Media	1	Hecho
1		Diseñar interfaz grafica	Media	1	Hecho
1		Realizar testing unitario al módulo desarrollado	Media	1	Hecho

1	Plantilla de servicio de usuarios (HU- 005)	Diseñar esquema y compatibilidades para la plantilla de servicio de usuarios	Alta	1	Hecho
1		Codificar plantilla de servicio de usuarios	Alta	2	Hecho
1		Realizar testing unitario a los componentes generados producto de la utilización de la plantilla desarrollada	Alta	1	Hecho

Estructura de Datos

El proyecto posee cuatro estructuras de datos en las cuales soporta sus procesos e interacciones, brindándole al sistema desde una correcta persistencia de información, hasta una notable mejoría en cuanto eficiencia general del sistema y la posibilidad de enviar notificaciones de forma asíncrona a usuarios dentro de la plataforma.

A continuación, se presenta el diagrama DER de la base de datos relacional PostgreSQL, la cual cumple con la administración y gestión de usuarios y sus correspondientes identificadores, incluyendo los provistos por el servidor de identificación OAUTH2.

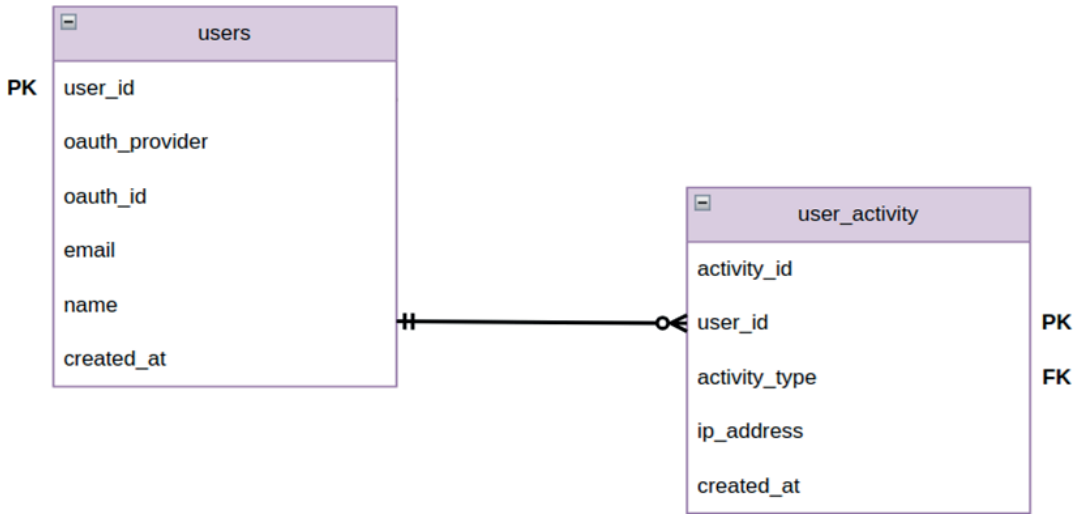

Figura 3. DER base de datos relacional para gestión de usuarios

A continuación, se presentan los diccionarios de datos de la base de datos relacional recién presentada:

Tabla 36. Diccionario tabla users

Tabla: users			
Campo	**Longitud**	**Tipo de Dato**	**Descripción**
user_id	19	Numérico	Identificador de usuario
oauth_provider	16	Alfabético	Proveedor OAUTH
oauth_id	19	Numérico	Numero identificador provisto por OAUTH
email	254	Alfabético	Email provisto por proveedor OAUTH
name	254	Alfabético	Nombre provisto por proveedor OAUTH
created_at	19	Timestamp	Momento de creación de la cuenta

Tabla 37. Diccionario tabla user_activity

Tabla: user_activity			
Campo	**Longitud**	**Tipo de Dato**	**Descripción**
activity_id	19	Numérico	Identificador de sesión
user_id	19	Numérico	Identificador de usuario
activity_type	254	Alfabético	Tipo de actividad realizada
ip_address	15	Numérico	Direccion IPv4
created_at	19	Timestamp	Momento de realización de la actividad

Durante el diseño del sistema, se detectó que incluir un almacenamiento temporal del tipo “caché” podría brindar una reducción temporal en la respuesta del sistema hacia el usuario, aprovechando los procesamientos realizados previamente. Es por esto que fue implementado un almacenamiento temporal con REDIS, el cual busca reducir las llamadas al API de Spring Initializr, analizando las llamadas previas en un tiempo cercano en busca a una igual a la que se desea realizar.

El siguiente diagrama ejemplifica el proceso mencionado:

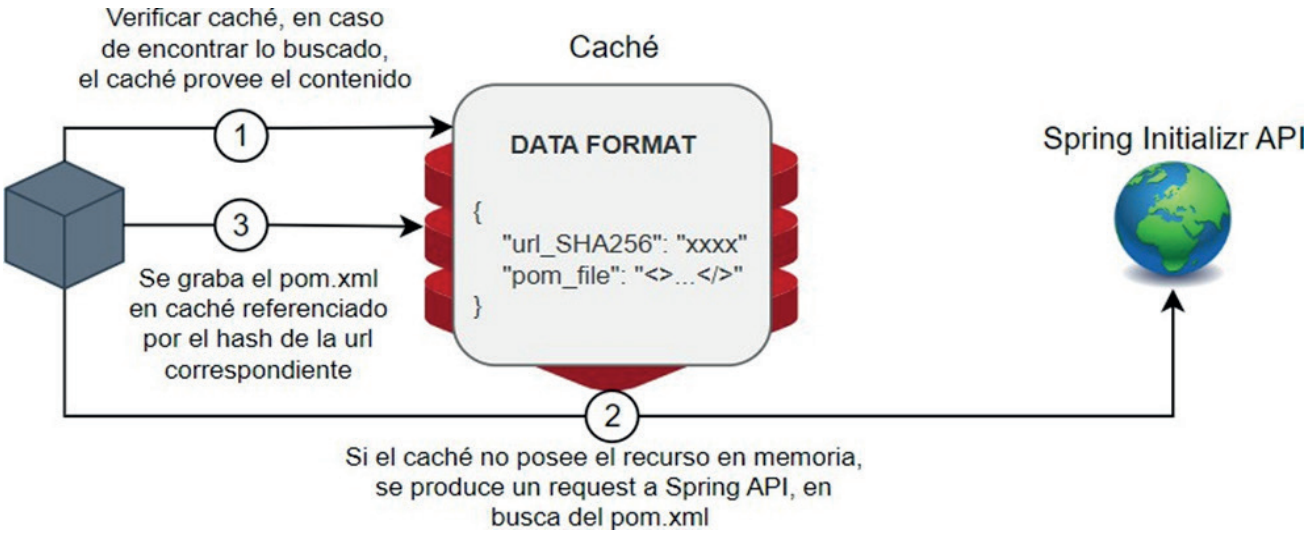

Figura 4. Explicación de funcionalidad de sistema cache

De esta forma, en muchos casos, se logra ahorrar el valioso tiempo consumido por una llamada hacia una API externa.

La simple estructura de datos del caché mencionado se expone a continuación:

init_request

```
{
    "url_SHA256": "558e7c63a0b(...)",
    "pom_file": "<>(...)</>"
}
```

Figura 5. Estructura de datos de caché para la generación de poms

A continuación, se presenta el diccionario de la estructura de datos presentada.

Tabla 38. Diccionario de datos presentes en caché

Campo	Longitud	Tipo de Dato	Descripción
url_SHA256	64	Alfabético	Hash SHA-256 de la url utilizada para la petición hacia la API de Spring Initializr
pom_file	65535(max)	Alfabético	Contenido del archivo pom.xml resultante del request

También se definió una estructura de datos encargada de almacenar las arquitecturas diseñadas por los usuarios en la plataforma y su código correspondiente generado de forma automática. De esta forma, mediante MinIO, el sistema tiene una plataforma donde trabajar dinámicamente con el código, permitiendo a los usuarios pueden descargar un archivo .zip con la arquitectura diseñada. La estructura de datos mencionada cumple con el siguiente formato:

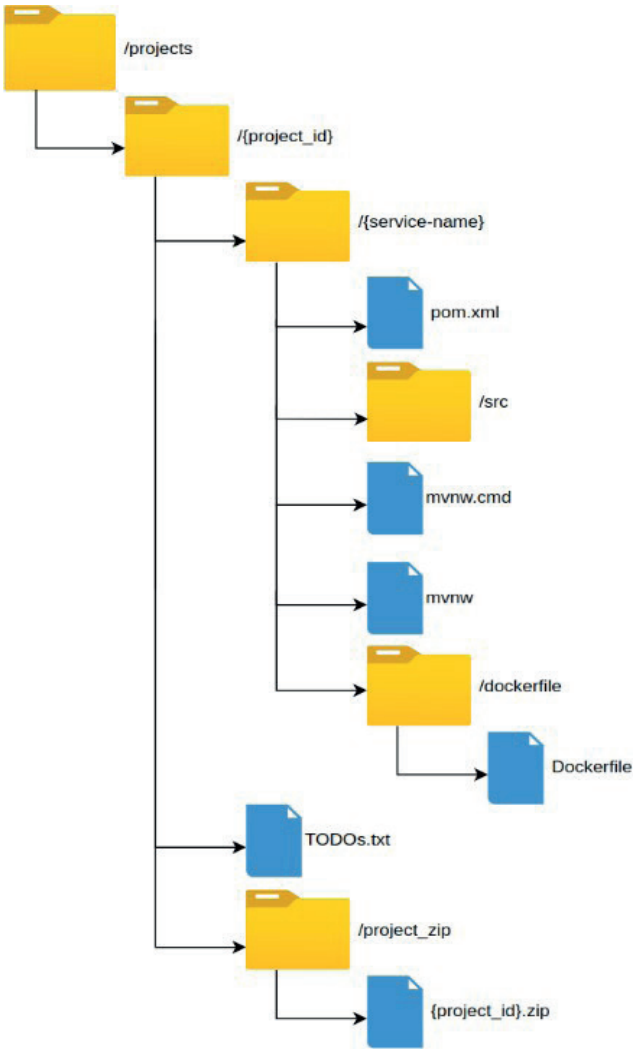

Figura 6. Estructura de archivos para almacenar y generar código

Además, el sistema requirió para un correcto funcionamiento en procesos asíncronos la inclusión de un sistema de mensajería, en este caso se optó por utilizar Apache Kafka. La herramienta mencionada cumple con dos tareas cruciales para el sistema:

- Informar de la presencia de una nueva arquitectura a generar y brindar su especificación.
- Mantener actualizado al usuario sobre el proceso de generación del código que solicitó, informándole en qué estado se encuentra la generación del producto final.
- Los tópicos utilizados para las funciones mencionadas son service-generation- request y service-generation-status respectivamente, los cuales interaccionan con los servicios del sistema (figura 7).

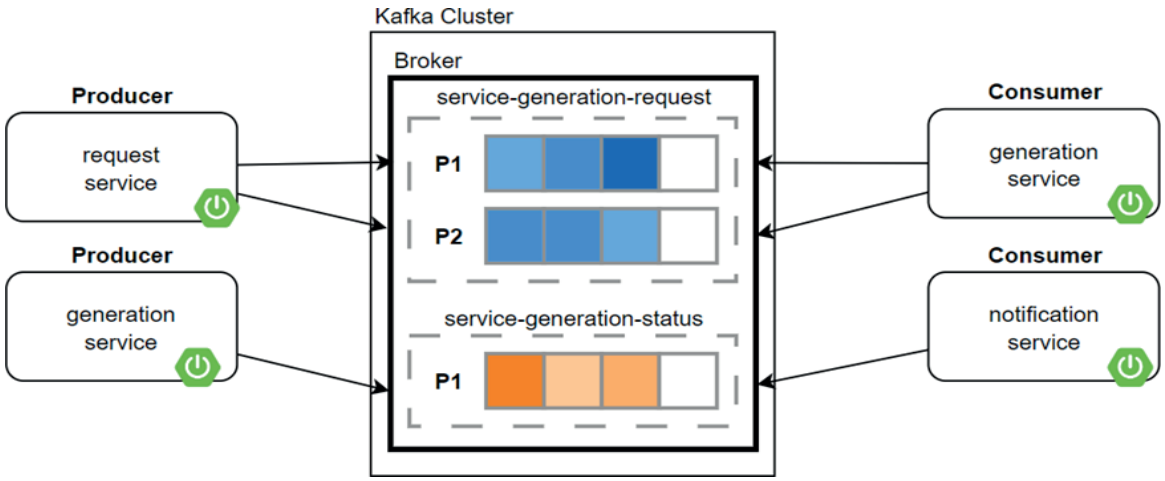

Figura 7. Diagrama de explicación del sistema asíncrono de mensajes

https://doi.org/10.56294/ai2024132 ISSN: 3072-7952

Como se puede apreciar en el diagrama presentado, el servicio está compuesto por un único bróker y service-generation-request posee dos particiones a comparación de service-generation-status, que solo posee una. Esta decisión está basada en la posibilidad de procesamiento paralelo en las solicitudes generadas por los usuarios que brinda la presencia de más de una partición, algo no tan necesario en el caso de los mensajes de progreso de la generación de código.

Se procede a exponer el modelo de datos de cada tópico mencionado.

Service-generation-status:

Mediante la información contenida por el tópico expuesto, el sistema es capaz de mantener al usuario informado en materia del progreso transcurrido referido a la generacion de codigo del sistema que diseñó en el canvas.

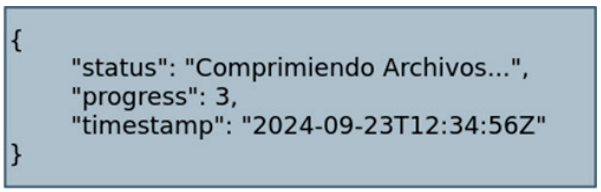

Figura 8. Estructura de datos del tópico 'service-generation-status'

A continuación, se presenta el diccionario de la estructura de datos presentada:

Tabla 39. Diccionario de datos de tópico 'service-generation-status'

Campo	Longitud	Tipo de Dato	Descripción
status	32	Alfabético	Mensaje informativo del proceso de generación dinámica de código
progress	1	Numérico	Rango de números del 1 al 3, indicando en que paso del proceso se encuentra la generación dinámica de código
timestamp	19	Alfabético	Momento de actualización de estado de la actividad

Service-generation-request: la información transportada por el presente tópico definido es crítica para el funcionamiento del sistema, ya que en base a él JSON ejemplificado a continuación, el sistema recibe de forma asíncrona las nuevas tareas a procesar y construye el código necesario solicitado por el usuario. Es importante aclarar que el JSON posee muchos atributos que para el prototipo podrían parecer innecesarios ya que no permiten personalización, pero de esta forma el sistema se encuentra listo para ampliar las personalizaciones en un hipotético caso de un desarrollo completo.

A continuación, se presenta el diccionario de la estructura de datos presentada:

Tabla 40. Diccionario de tópico 'service-generation-request'

Campo	Tipo de Dato	Descripción
projectName	Alfabético	Nombre del proyecto general
versión	Numérico	Versión del proyecto, 1,0 por defecto
services	Alfabético	Contiene y define los componentes de la arquitectura
name	Alfabético	Nombre del componente
type	Alfabético	Tipo de componente, es decir de que trata el servicio
versión	Numérico	Versión de la plantilla utilizada
description	Alfabético	Descripción del componente
endpoints	Alfabético	Sección referida a los endpoints del componente
path	Alfabético	Url base del componente
dependencies	Alfabético	Dependencias del componente
connections	Alfabético	Conexiones entre componentes especificadas en el canvas
source	Alfabético	Punto A de la conexión
target	Alfabético	Punto B de la conexión
protocol	Alfabético	Protocolo de la comunicación
type	Alfabético	Metodología utilizada para la comunicación
databases	Alfabético	Contiene los elementos de persistencia de los servicios

https://doi.org/10.56294/ai2024132 ISSN: 3072-7952

name	Alfabético	Nombre del elemento de persistencia
owner	Alfabético	Servicio que interactúa con el elemento de persistencia
type	Alfabético	Herramienta utilizada
version	Numérico	Versión de la herramienta utilizada
infrastructure	Alfabético	Contiene los elementos de configuración e infraestructura
name	Alfabético	Nombre del componente de infraestructura
type	Alfabético	Tipo de componente de infraestructura
port	Numérico	Puerto del componente de infraestructura
security	Alfabético	Contiene los elementos de seguridad global del sistema
auth	Alfabético	Especifica como se realiza la autenticación en el sistema
type	Alfabético	Metodología de autenticación
configurations	Alfabético	Contiene información para la personalización de seguridad
secretKey	Alfabético	Clave secreta utilizada para asegurar el sistema

Figura 9. Estructura de datos del tópico 'service-generation-request'

Prototipos de Interfaces de Pantallas

Al acceder a la plataforma web, se solicita de forma obligatoria ingresar por medio de la cuenta de Google perteneciente al usuario.

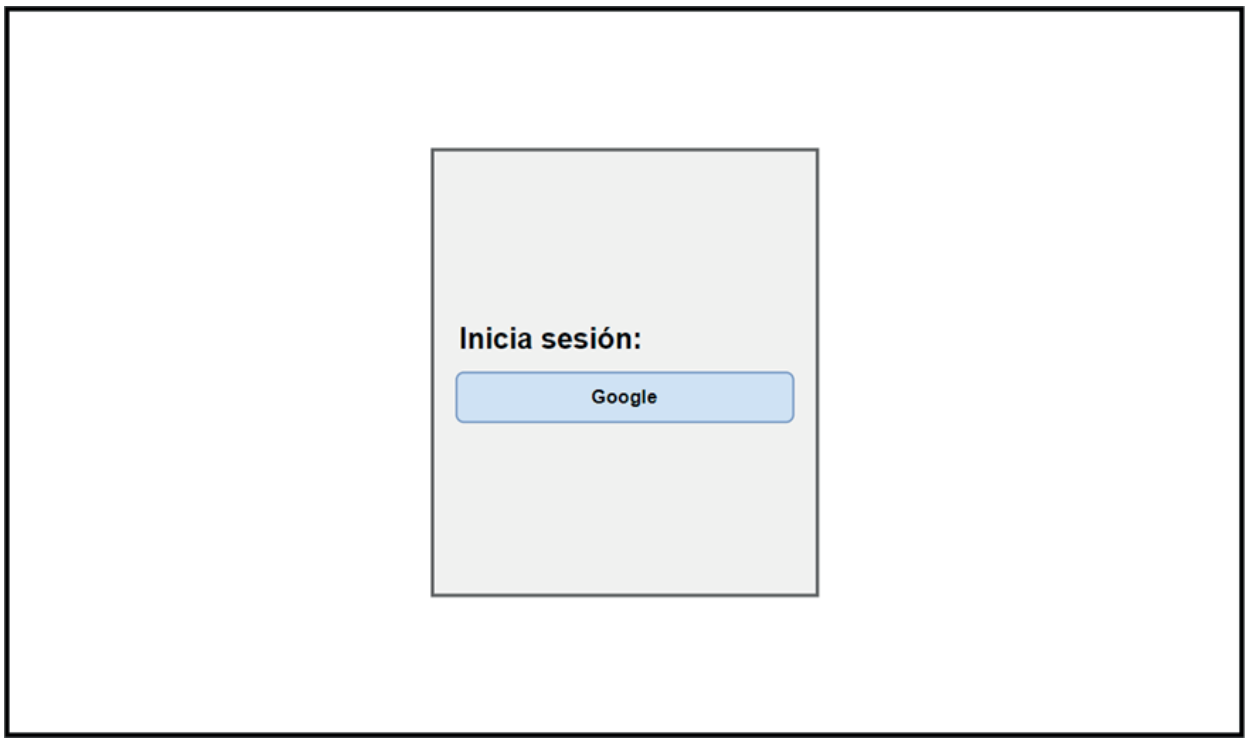

Figura 10. Pantalla de inicio de sesión

Una vez es validado y/o registrado el usuario en el sistema, el mismo es enviado a la página principal de la plataforma, donde se encuentra su funcionalidad core, referida al diseño de sistemas distribuidos.

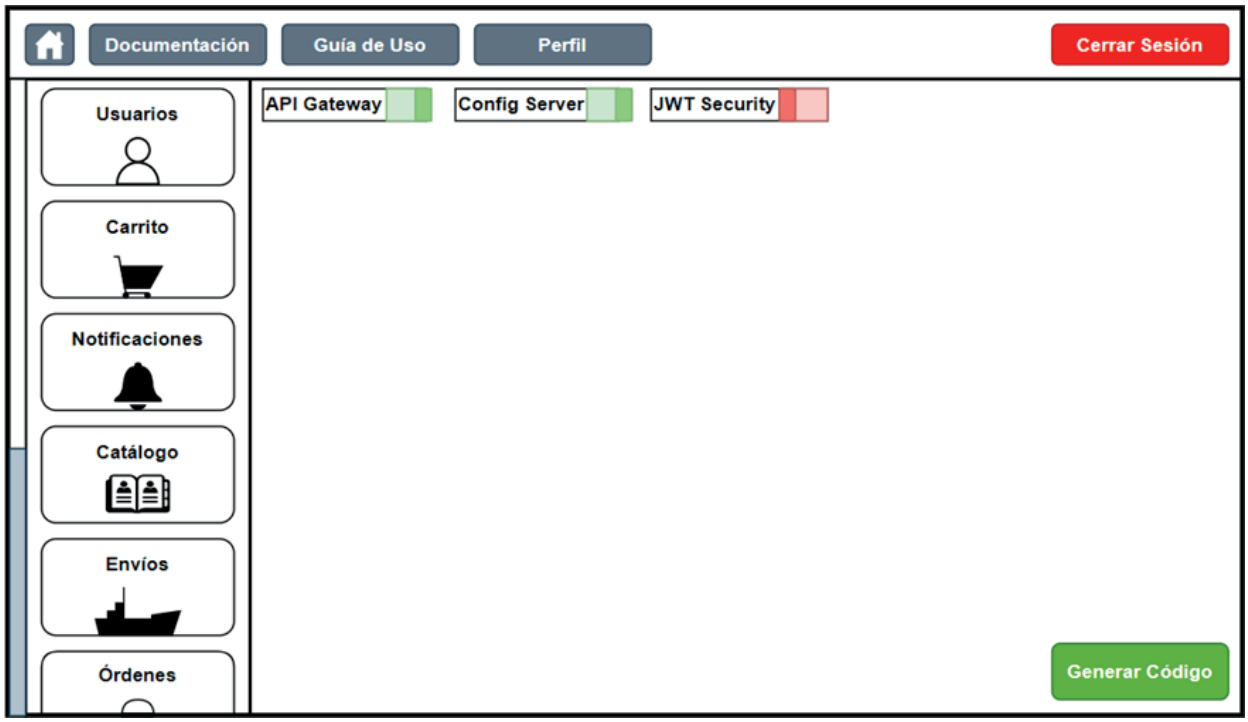

Figura 11. Pantalla de inicio

Dentro de la pantalla principal, el usuario es capaz de arrastrar los componentes participes de la arquitectura al canvas, para comenzar a diseñar el sistema.

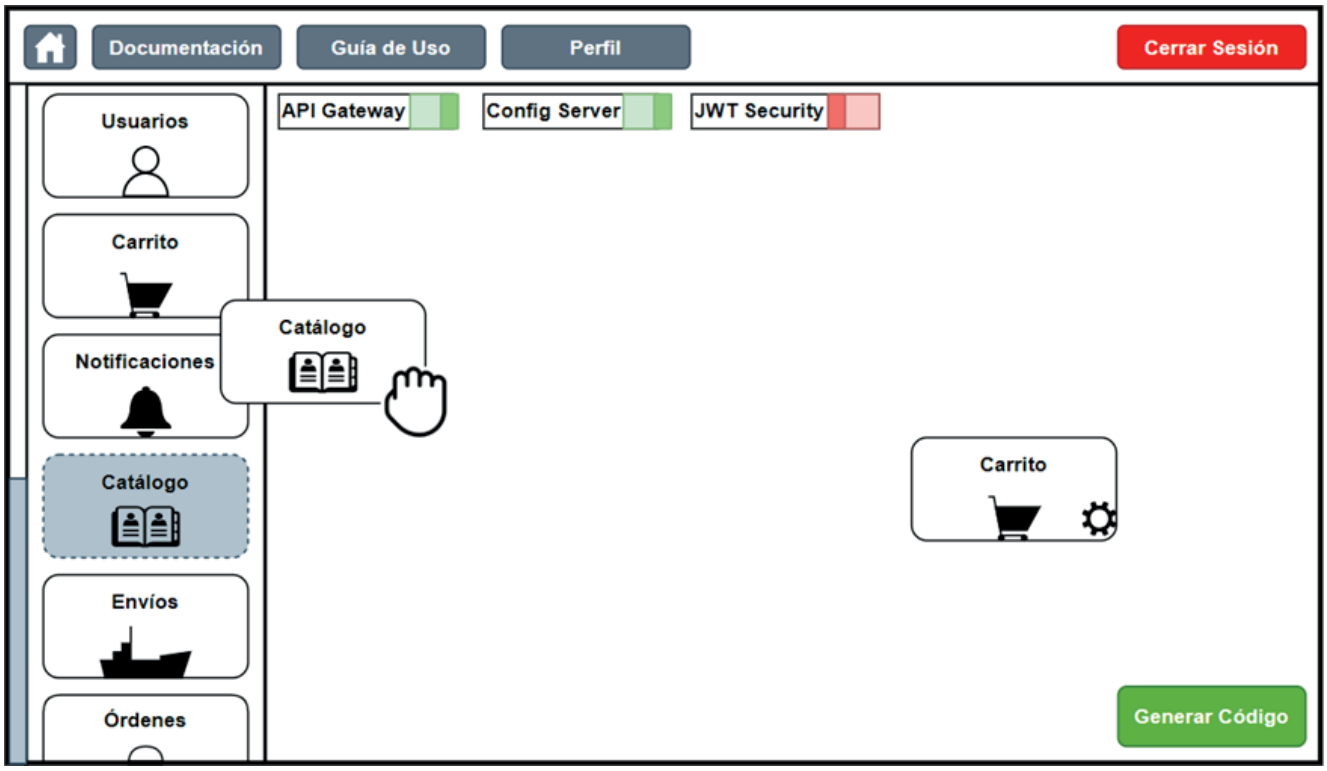

Figura 12. Funcionalidad Drag-and-Drop

Los componentes depositados sobre el canvas a su vez, si su compatibilidad lo permite, se pueden unir entre sí, para generar funcionalidades más complejas mediante la interacción de los mismos.

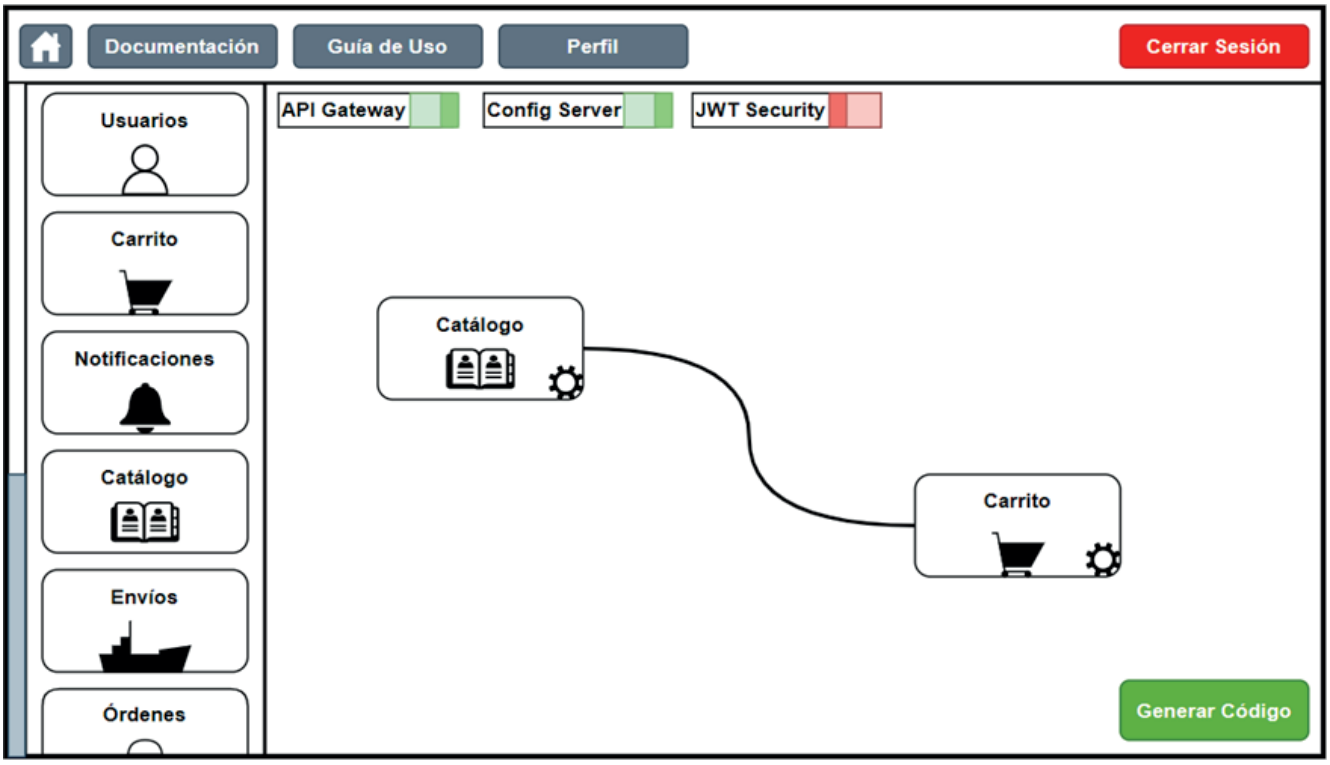

Figura 13. Funcionalidad de unión de componentes

Para permitir la personalización de cada componente depositado en el canvas, el usuario debe presionar el engranaje correspondiente para acceder al menú de configuración.

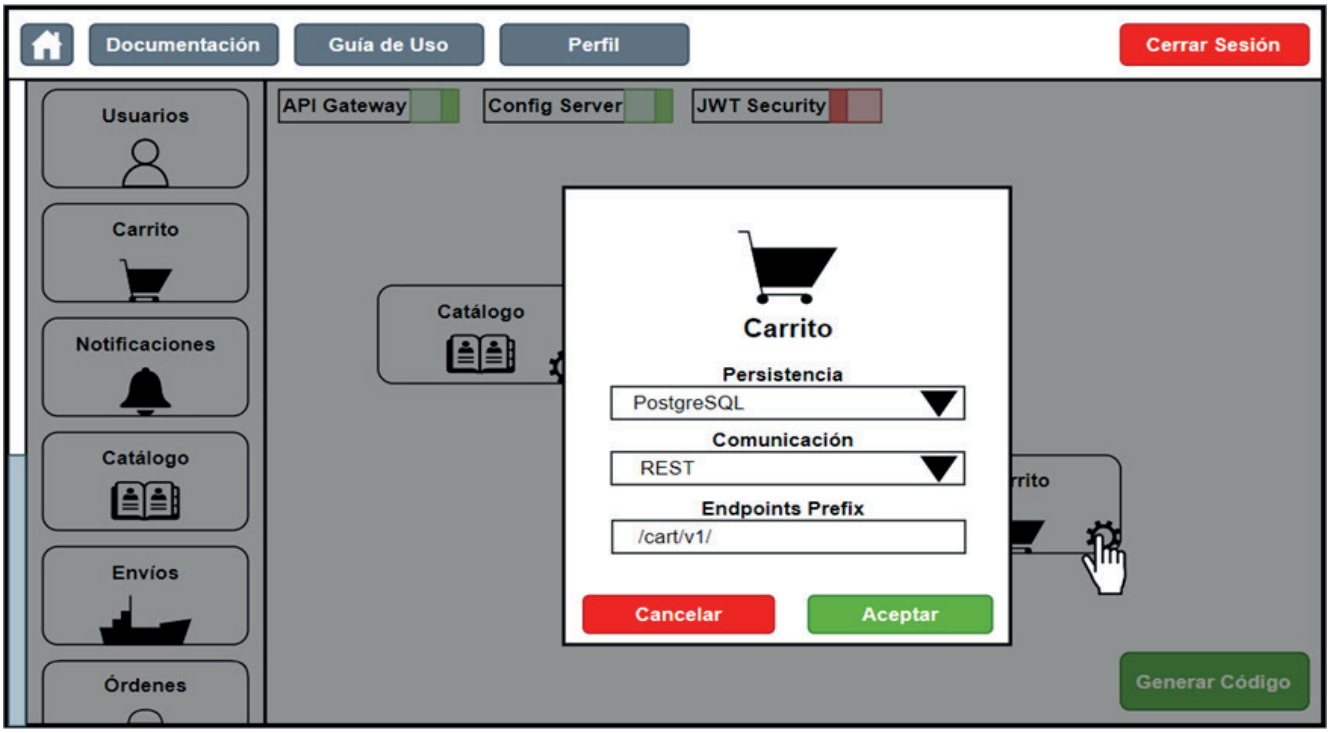

Figura 14. Pantalla de configuración de componente

Una vez que el usuario diseñó y configuró a su gusto una arquitectura específica, este puede generar el código correspondiente presionando el botón verde situado en la esquina inferior derecha del canvas, donde recibirá en tiempo real el progreso que lleva el proceso de generación dinámica de código.

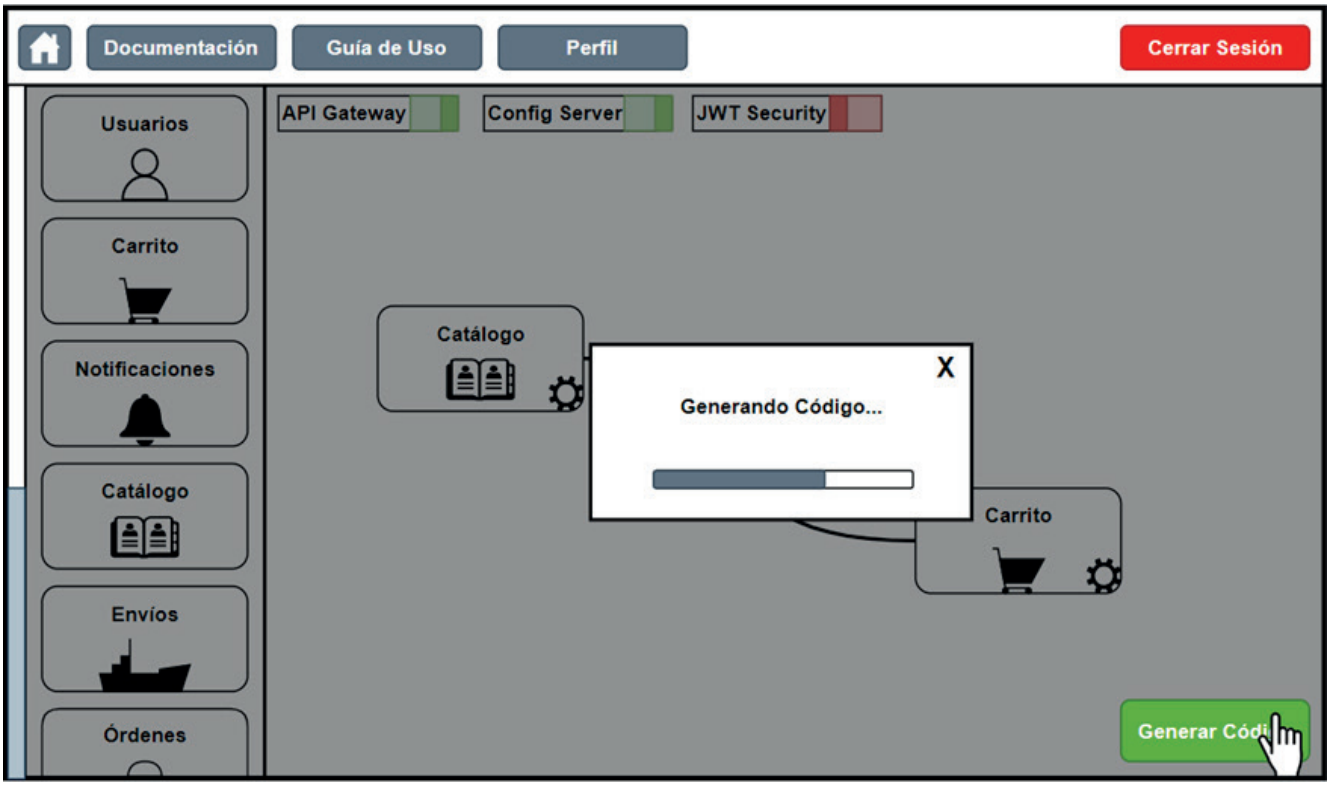

Figura 15. Pantalla de proceso de generación de código

Para que el usuario conozca los diferentes componentes utilizables en el canvas, su composición, funcionamiento, metodología de persistencia y compatibilidades, puede acceder a la sección de documentación, donde se encontrará con la información necesaria para comprender cada pieza disponible.

https://doi.org/10.56294/ai2024132 ISSN: 3072-7952

Figura 16. Pantalla de documentación

Si el usuario desea comprender el funcionamiento de la plataforma en sí, existe un apartado denominado Guía de Uso, donde puede acceder a la información mencionada

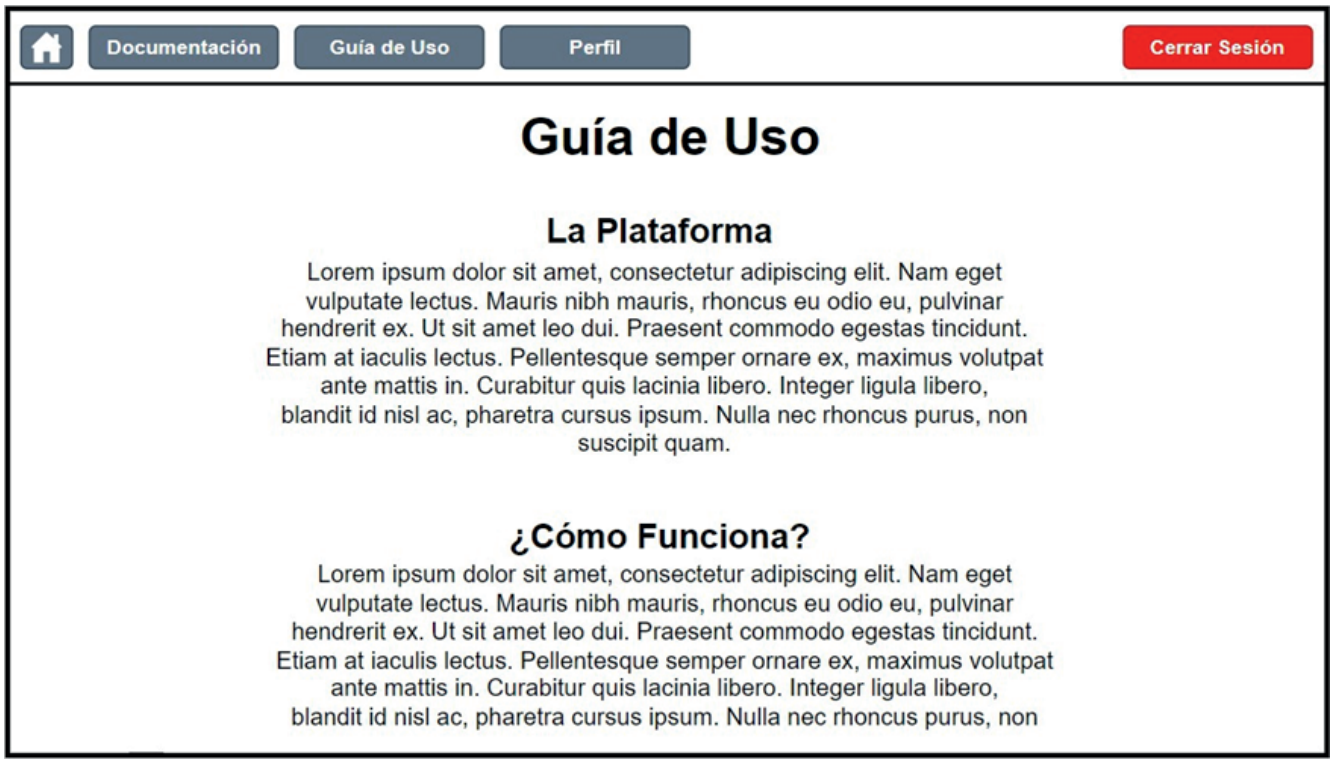

Figura 17. Pantalla de guía de uso

Cuando el usuario del sistema genera arquitecturas en el canvas, las ultimas 5 son registradas y accesibles para su descarga en el apartado Perfil, donde se expone información básica de utilidad sobre las arquitecturas mencionadas y un enlace para su correspondiente descarga.

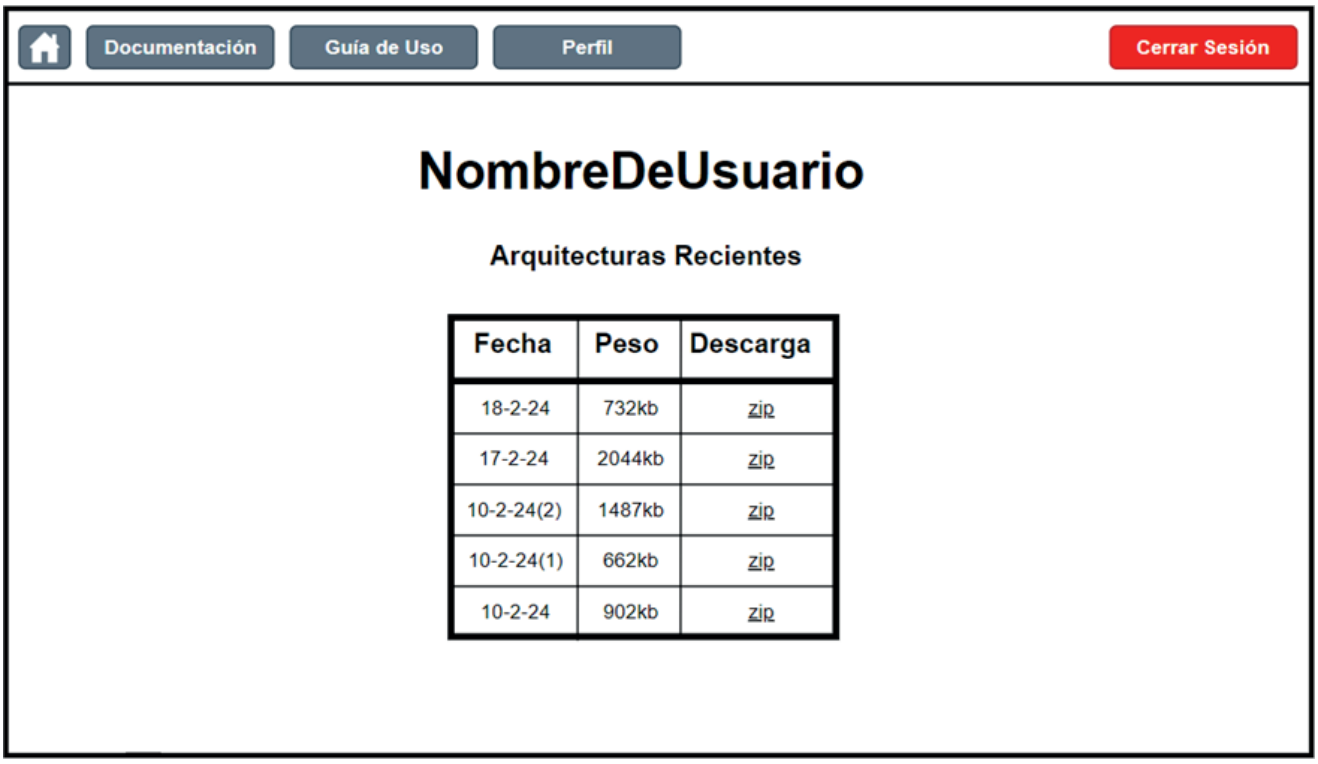

Figura 18. Pantalla de perfil de usuario

Finalmente, para concluir su actividad en la plataforma, el usuario puede cerrar su sesión mediante el botón presente en la esquina superior derecha, luego de confirmar definitivamente su deseo de realizar tal acción.

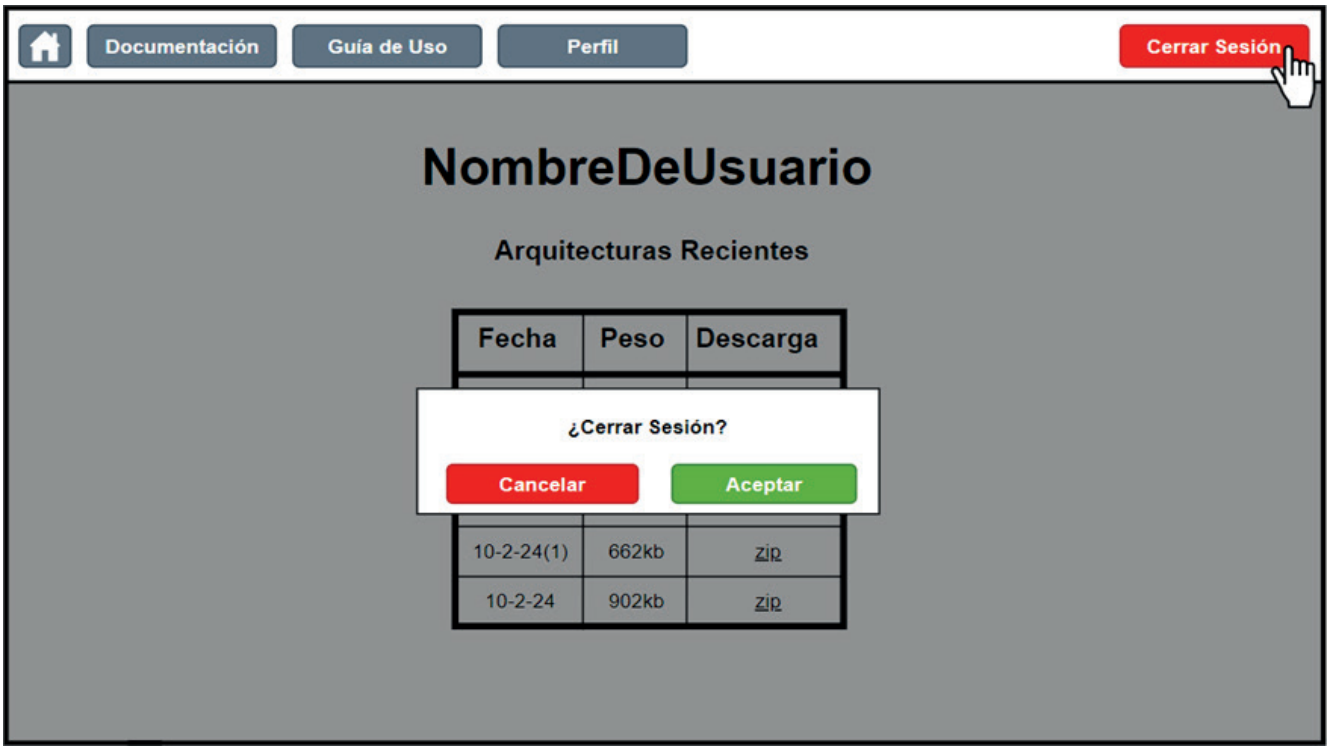

Figura 19. Pantalla de cierre de sesión

Diagrama de Arquitectura

A continuación, se presenta el diagrama de arquitectura correspondiente al proyecto tratado en el presente documento. Debido a las decisiones tomadas apreciables en el diagrama, se logró un sistema altamente escalable y eficiente para el fin que persigue.

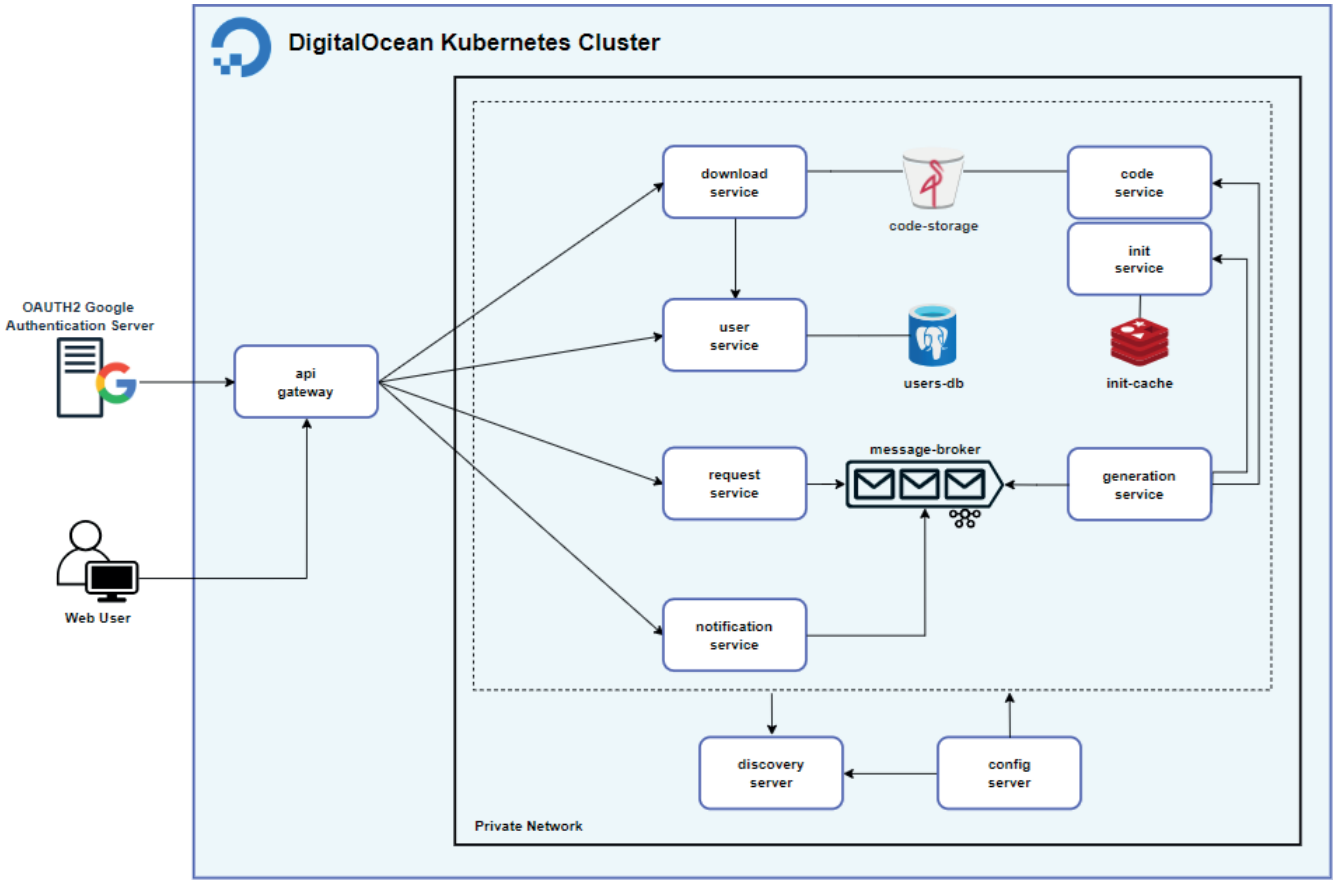

Figura 20. Diagrama de arquitectura

Seguridad

Acceso a la Aplicación

Como fue expresado en reiteradas ocasiones a lo largo del presente documento, el proyecto gestiona a los usuarios mediante OAuth2, específicamente mediante el inicio de sesión vía Google, siendo una alternativa muy conveniente para una aplicación como la desarrollada, debido a su permiso de acceso al público general.

Si bien por motivos de respaldo, auditoria y gestión de recursos el propio proyecto almacena la información de los usuarios registrados en la plataforma, las contraseñas son manejadas por Google, debido a que es mediante un usuario de la plataforma mencionada que el sistema concede el acceso al proyecto. Por lo tanto, los usuarios se deben adecuar a los requerimientos de Google para la creación de sus credenciales. A continuación, se listan los requerimientos al día de la fecha dispuestos por Google para la creación de un email bajo su dominio y su respectiva clave de acceso.

Requerimientos para la creación de un email:

- Debe seguir el formato estándar de una dirección de correo electrónico, es decir nombre@dominio.com. En este caso, el dominio es @gmail.com.
- El nombre de usuario, es decir el texto previo al '@', debe contener entre 6 y 30 caracteres.
- Únicamente se permiten caracteres del tipo a-z y A-Z, además de números 0-9 y los caracteres especiales como el punto (.), el guion bajo (_) y el medio (-).
- La dirección de correo no debe estar en uso en Gmail. Es decir que cada dirección de correo debe ser única.

Requerimientos para la especificación de una contraseña:

- La contraseña debe contener más de 8 caracteres.
- Si bien no existe una limitación relacionada, se recomienda la utilización de minúsculas, mayúsculas, números y caracteres especiales.
- No debe formar parte de la lista de contraseñas consideradas comunes por Google.

A su vez, Google almacena las contraseñas de los usuarios utilizando funciones de hash que convierten la contraseña en una cadena de texto irreversible, además de añadir un valor aleatorio denominado 'salt' para dificultar los intentos de ataque por fuerza bruta. También cabe destacar la existencia de la posibilidad para los

usuarios de Google de activar la autenticación de dos factores, conocida como 2FA, añadiendo una capa extra de seguridad al requerir un segundo elemento para acceder a la cuenta de usuario.

Respecto a los roles de usuario expuestos a continuación, debido a la naturaleza del proyecto y el desarrollo únicamente de las funcionalidades core para el prototipo, el usuario administrador no tiene ninguna ventaja o capacidad diferencial. Aun así, fue añadido de forma "lógica" en el sistema, con el fin de disponer del mismo pensando en futuras actualizaciones que provean características que requieran de un control extra dentro de la aplicación.

Perfiles presentes en la aplicación:

- Usuario común: tiene acceso a la totalidad de funcionalidades presentes en el prototipado tecnológico, es decir, la especificación, configuración y descarga de servicios referenciados a su perfil de usuario y la capacidad de consulta tanto a documentación presente en la plataforma, como a la guía de uso.
- Usuario administrador: como fue explicado de forma previa, actualmente el usuario administrador no posee permisos diferenciales respecto al usuario común, aun así, se encuentra presente en el sistema teniendo en cuenta posibles actualizaciones.

Política de Respaldo de Información

Con el fin de respaldar la información referida tanto al código de la aplicación, como a la información producida producto de su uso y ejecución, se almacenan 2 copias del código fuente de la aplicación y 3 copias de los datos de usuario.

Los datos de usuario, en primera instancia son almacenados en el propio hosting donde se almacena el volumen del contenedor referido al motor de base de datos. En este caso se trata del proveedor de servicios cloud DigitalOcean. Como segunda instancia de back-up, el sistema ejecuta un proceso cada día a las 00:00 (GMT-3), con el fin de persistir una copia del contenido de la base de datos en el servidor local de las oficinas de desarrollo. Finalmente, y con el objetivo de maximizar la integridad de los datos de usuario, se persiste de forma semanal una copia de los datos de usuario en un disco duro externo, siendo este almacenado en una ubicación confidencial, en un edificio diferente al servidor local, y de conocimiento únicamente por parte de los cargos gerenciales de la empresa.

En cuanto al código fuente de la aplicación, este es tratado y persistido en Github, donde los desarrolladores trabajan en él. A su vez, como se mencionó anteriormente, las versiones funcionales son respaldadas y persistidas al momento de su completitud en dos instancias. En primera instancia el código es almacenado de forma manual en el servidor local de la empresa, para finalmente, en segunda instancia, ser almacenado en un disco duro externo reservado en una ubicación confidencial, en un edificio diferente al que se encuentran las oficinas de la empresa, y de conocimiento únicamente por los cargos gerenciales de la misma.

El servidor local mencionado tanto en el respaldo de código fuente como de datos de usuario se trata de un NAS presente en las oficinas de desarrollo, configurado con RAID 10, para obtener un alto nivel de redundancia además de un notable rendimiento, asegurando una destacable disponibilidad de la información incluso en caso de incidentes, brindando una veloz recuperación del contenido persistido.

Por parte de DigitalOcean la disponibilidad también es tenida en cuenta y, de hecho, es uno de los factores que incidieron en que el sistema sea ejecutado sobre sus servicios en la nube, ya que, en los productos utilizados para el despliegue y ejecución del proyecto desarrollado, la plataforma mencionada presume de un tiempo de actividad del 99,99 %.[1]

Análisis de Costos

A continuación, se presentan desglosados, los costos estimados del proyecto en referencia a los recursos humanos necesarios para el desarrollo del sistema informático. Los mismos fueron obtenidos de la web del Consejo Profesional de Ciencias Informáticas de la provincia de Córdoba el día 21 de octubre del año 2024.

Tabla 41. Análisis de costos RRHH

Rol	Honorarios	Meses	Subtotal (AR$)
Analista Programador Senior	$1 697 430,72	3	$5 092 292,16
Desarrollador Backend	$1 985 445,37	3	$5 956 335,93
Desarrollador Frontend	$1 883 828,08	2	$3 767 656,16
Analista Testing de Aplicaciones	$1 646 481,38	2	$3 292 962,76
Total Desarrollo			$18 109 247,01

Ya presentados los valores referidos a la mano de obra, a continuación, se presentan los costos operativos considerados necesarios para el correcto despliegue y funcionamiento del proyecto.

Tabla 42. Análisis de costos operativos

Recurso	Cant.	Fuente	Subtotal AR$	Mensual AR$
* Kubernetes Basic Node (DigitalOcean) 8gb RAM 4 vCPU 160GB storage	2	https://www.digitalocean.com/pricing	-	$94 464
Nombre de dominio .com	1	https://www.hostinger.com.ar/dominios	-	$2144
NAS Drive S.O Linux Compatible RAID 10 Capacidad 4 HDD	1	https://www.compel.com.ar/almacenamiento/almacenamiento/nas-drive-au-4b-25- 35a335696.html	$816 851	-
HDD 3T NAS	4	https://www.compel.com.ar/almacenamiento/hdd-internos/hdd-3t-sea-35-nas- ironwol-328858.html	$636 660	-
Total Costo Inicial			$1 453 511	
Total Costos Fijos				$96 608

Nota: *Valor original en USD, convertido a AR$ considerando 1 USD equivalente a 948 AR$ en base a la cotización brindada por el Banco Central de la República Argentina el día 21 de octubre del año 2024.[2]

En cuanto a los costos referidos al software utilizado para el desarrollo del proyecto, se tomó la decisión de recurrir a plataformas de código abierto, accediendo a planes gratuitos, para ahorrar costos en cuanto a licenciamiento. Aun así, se exponen estas herramientas a modo informativo.

Tabla 43. Análisis de costos herramientas de desarrollo

Herramienta	Subtotal (AR$)
PostgreSQL	$0
Apache Kafka	$0
MinIO	$0
Docker	$0
Kubernetes	$0
Spring Framework	$0
Total Licencias de Software	$0

A modo de conclusión del análisis de costos, se expone el resumen de los mismos excluyendo los valores de los salarios previamente detallados.

Tabla 44. Resumen de costos excluyendo RHHH

	Capital Humano	Software y Licencias	Infraestructura y Hardware	Total
Costo Inicial (AR$)*	$18 109 247,01	$0	$1 453 511	19 562 758,01
Costo Fijo Mensual (AR$)**	$0	$0	$96 608	$96 608

Nota: * Comprende todo costo relacionado con los primeros tres meses de actividad, es decir, hasta concluir el desarrollo del sistema, excluyendo el despliegue de este.
** Comprende los costos de mantener el sistema, una vez desarrollado, desplegado en la nube.

Análisis de Riesgos

A continuación, se presentan descriptos y detallados los riesgos que se pueden presentar durante el transcurso del desarrollo del proyecto, dividiendo los mismos en diferentes tablas según su razón de origen. En ellas se presenta tanto la probabilidad como el impacto, valores que basaran su significado en la matriz que se expone luego.

Riesgos Técnicos

Tabla 45. Riesgos Técnicos

ID	Tipo	Riesgo	Probabilidad	Impacto	Causa
1	Técnico	El tiempo de respuesta de la plataforma no cumple con los estándares de respuesta esperados	0,40	4	Ineficiencias en la arquitectura o falta de optimización en las comunicaciones
2	Técnico	Vulnerabilidad de seguridad en servicios que interactúan con bases de datos.	0,70	2	Falta de sanitización y validación adecuada de las entradas de usuario

Tabla 46. Riesgos del Proyecto

ID	Tipo	Riesgo	Probabilidad	Impacto	Causa
3	Proyecto	Dependencia de API's de terceros para integración o creación de servicios que no estén disponibles	0,80	3	Falta de control sobre la calidad y disponibilidad de servicios utilizados en el sistema
4	Proyecto	Dificultad para conseguir el personal técnico necesario para llevar a cabo el desarrollo del sistema	0,35	3	Temática muy específica debido al objetivo final del sistema, enfocado en sistemas distribuidos
5	Proyecto	Recursos insuficientes para el desarrollo del proyecto	0,45	4	Falta de inversión en recursos humanos y/o tecnológicos
6	Proyecto	El campo de investigación no brinda información útil sobre la cual partir	0,60	2	El campo de investigación de microservicios puede ser incompleto debido a su reciente inclusión

Una vez expuestos los riesgos identificados del proyecto, se procede con la siguiente matriz de riesgo previamente mencionada, con el fin de ponderar las probabilidades de ocurrencia y sus impactos relacionados.

			IMPACTO				
			Insignificante 1	Menor 2	Significativo 3	Mayor 4	Severo 5
PROBABILIDAD	Casi Seguro	**0,9**	0,9	1,8	2,7	3,6	4,5
	Probable	**0,7**	0,7	1,4	2,1	2,8	3,5
	Moderado	**0,5**	0,5	1	1,5	2	2,5
	Poco Probable	**0,3**	0,3	0,6	0,9	1,2	1,5
	Raro	**0,1**	0,1	0,2	0,3	0,4	0,5

Figura 21. Matriz de Riesgos

En base a la matriz presentada, se desarrollaron tanto las tablas previamente expuestas como la definida a continuación, referida al análisis cuantitativo de los riesgos.

Tabla 47. Análisis cuantitativo de riesgos

Riesgo	Probabilidad de Ocurrencia	Impacto	Grado de Exposición	Porcentaje	Porcentaje Acumulado
Dependencia de API's de terceros para integración o creación de servicios que no estén disponibles	0,80	3	2,40	25,40	25,40
Recursos insuficientes para el desarrollo del proyecto	0,45	4	1,80	19,06	44,46
El tiempo de respuesta de la plataforma no cumple con los estándares de respuesta esperados	0,40	4	1,60	16,93	61,39

https://doi.org/10.56294/ai2024132 ISSN: 3072-7952

Vulnerabilidad de seguridad en servicios que interactúan con bases de datos.	0,70	2	1,40	14,81	76,20
El campo de investigación no brinda información útil sobre la cual partir	0,60	2	1,20	12,70	88,90
Dificultad para conseguir el personal técnico necesario para llevar a cabo el desarrollo del sistema	0,35	3	1,05	11,10	100

Una vez presentado el grado de exposición al riesgo correspondiente para cada riesgo detectado, es momento de utilizar el Principio de Pareto para centrar la atención en los aspectos importantes y críticos, ignorando los más triviales. A continuación, se expone el grafico correspondiente.

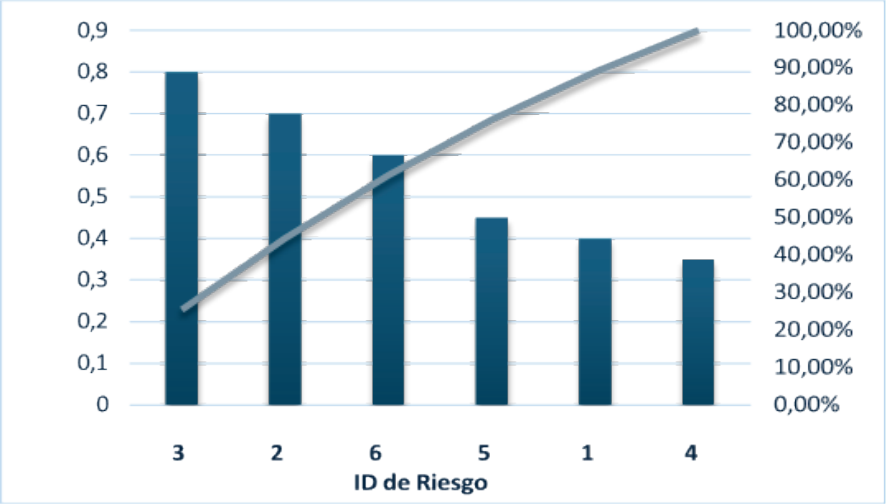

Figura 22. Principio de Pareto de la Exposición al Riesgo

Una vez identificados los riesgos más amenazantes utilizando el Principio de Pareto, se elaboraron planes de contingencia específicos para mitigar dichas amenazas. A continuación, se presentan los detalles de estos planes.

Tabla 48. Planes de contingencia

Riesgo	Plan de Contingencia
Dependencia de API's de terceros para integración o creación de servicios que no estén disponibles	Desarrollar soluciones internas mínimas viables para reducir la dependencia crítica de API's externas. Establecer contratos con proveedores clave para asegurar tiempos de respuesta y disponibilidad, y definir mecanismos de monitorización para detectar fallos y actuar rápidamente.
Recursos insuficientes para el desarrollo del proyecto	Establecer desde el inicio un plan de asignación flexible de recursos, que permita redistribuir esfuerzos en función de las prioridades del proyecto. Proponer fases adicionales o ajustes en el alcance del proyecto para adaptarse al presupuesto disponible.
El tiempo de respuesta de la plataforma no cumple con los estándares de respuesta esperados	Realizar una revisión exhaustiva de la arquitectura y aplicar optimizaciones puntuales en las áreas críticas. Integrar soluciones de monitoreo en tiempo real para detectar problemas de rendimiento y ajustar la capacidad del sistema.
Vulnerabilidad de seguridad en servicios que interactúan con bases de datos.	Implementar capas adicionales de seguridad, como firewalls a nivel de aplicación y encriptación de los datos sensibles. Realizar revisiones periódicas de código centradas en la seguridad y aplicar parches inmediatos ante cualquier vulnerabilidad descubierta.

CONCLUSIONES

El presente proyecto permitió demostrar que la combinación de arquitecturas de microservicios con enfoques Low-code no solo es posible, sino también altamente beneficiosa para reducir la complejidad en el diseño y desarrollo de sistemas distribuidos. A través de la implementación de una plataforma visual e intuitiva, se logró ofrecer una solución capaz de acortar significativamente los tiempos de desarrollo, facilitar la integración de componentes y disminuir los errores comunes durante la codificación manual.

Uno de los principales logros alcanzados fue la construcción de un entorno funcional que permite a los desarrolladores arrastrar, configurar y relacionar microservicios de forma visual, para luego generar automáticamente el código correspondiente. Esta funcionalidad, respaldada por tecnologías como RDF, SPARQL y Apache Velocity, constituye un avance relevante en la automatización del desarrollo de software, asegurando consistencia estructural sin sacrificar flexibilidad.

La elección de herramientas modernas y open source como Java con Spring Framework, PostgreSQL, Apache Kafka, MinIO, Docker y Kubernetes resultó clave para asegurar la escalabilidad, portabilidad y robustez del sistema. Además, se implementaron prácticas seguras de autenticación con OAuth2 (vía Google), junto con mecanismos de respaldo y distribución que aseguran la integridad y disponibilidad de los datos.

En el aspecto metodológico, el uso de Scrum como marco de trabajo ágil permitió una evolución iterativa del producto, fomentando la mejora continua y una rápida respuesta ante obstáculos técnicos o cambios en los requerimientos. Esta dinámica fue fundamental para ajustar detalles en tiempo real, mejorar el diseño funcional del canvas, y adaptar la lógica de generación de código según los resultados obtenidos en cada sprint.

La recopilación de información mediante el análisis de literatura científica, complementado con observaciones en redes sociales utilizadas por desarrolladores, brindó una visión completa del problema a abordar. Esta integración entre teoría y práctica facilitó la validación de las necesidades reales del usuario objetivo y guió el diseño de funcionalidades clave de la plataforma.

En síntesis, el sistema desarrollado cumple con el objetivo de facilitar la creación de arquitecturas de microservicios, proporcionando una herramienta potente, accesible y adaptable. Si bien se trata de un prototipo funcional, su estructura y diseño anticipan una futura evolución con mayores posibilidades de personalización, ampliación de plantillas, y compatibilidad con entornos de producción empresariales. El camino hacia la democratización del desarrollo distribuido, mediante plataformas Low-code, queda así abierto y potenciado con esta propuesta tecnológica.

REFERENCIAS BIBLIOGRÁFICAS

1. DigitalOcean. DigitalOcean Managed Kubernetes. 2024. https://www.digitalocean.com/products/kubernetes

2. Banco Central de la República Argentina (BCRA). Cotizaciones por fecha. 2024. http://www.bcra.gob.ar/PublicacionesEstadisticas/Cotizaciones_por_fecha_2.asp

3. Chaudhary HAA, Ahmed T. Integration of micro-services as components in modeling environments for low code development. ISP RAS. 2021;33(4):19-30. doi:10.15514/ISPRAS-2021-33(4)-2

4. Dhoke P, Lokulwar P. Evaluating the Impact of No-Code/Low-Code Backend Services on API Development and Implementation: A Case Study Approach. In: 14th International Conference on Computing Communication and Networking Technologies (ICCCNT); 2023 Jul 11-13; Chennai, India. Piscataway: IEEE; 2023. p. 1-5. doi:10.1109/ICCCNT56998.2023.10306945

5. Apache Software Foundation. Apache Kafka. 2024. https://kafka.apache.org/

6. IBM. Minio. 2021. https://www.ibm.com/docs/es/cloud-private/3.2.x?topic=private-minio

7. JetBrains. IntelliJ IDEA features. 2024. https://www.jetbrains.com/es-es/idea/features/

8. Kubernetes. ¿Qué es Kubernetes? 2022. https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/

9. Lewis J, Fowler M. Microservices: a definition of this new architectural term. 2014. https://martinfowler.com/articles/microservices.html

10. Lopez BM, Garcia JL. Impacto de arquitecturas de microservicios en el desarrollo web [Tesis de maestría]. Madrid: Universidad Politécnica de Madrid; 2019. https://oa.upm.es/55917/1/TESIS_MASTER_BRUNO_MARTIN_LOPEZ.pdf

11. Mozilla Developer Network. HTML5. 2023. https://developer.mozilla.org/es/docs/Glossary/HTML5

12. Postman. What is Postman? 2024. https://www.postman.com/product/what-is-postman/

13. Richardson C. Microservices patterns: With examples in Java. New York: Manning Publications; 2019.

14. Rock Content. What is Bootstrap? 2020. https://rockcontent.com/es/blog/bootstrap/

15. Said M, Ezzati A, Arezki S. Microservice-specific language, a step to the low-code platforms. In: Lecture Notes in Networks and Systems. 2023;637:817-28. doi:10.1007/978-3-031-26384-2_72

16. Spring. Spring Framework. 2024. https://spring.io/projects/spring-framework

17. The Thymeleaf Team. Thymeleaf. 2024. https://www.thymeleaf.org/

18. Trello. Trello Tour. 2023. https://trello.com/es/tour

19. Vincent P, Lijima K, Driver M, Wong J, Natis Y. Gartner magic quadrant for enterprise low-code application platforms. Stamford: Gartner, Inc.; 2019. https://www.gartner.com/en/documents/3956079

20. World Wide Web Consortium. SPARQL 1.1 Query Language. 2013. https://www.w3.org/TR/sparql11-query/

FINANCIACIÓN

Ninguna.

CONFLICTO DE INTERESES

Los autores declaran que no existe conflicto de intereses.

CONTRIBUCIÓN DE AUTORÍA

Conceptualización: Tomás Darquier, Pablo Alejandro Virgolini.
Curación de datos: Tomás Darquier, Pablo Alejandro Virgolini.
Análisis formal: Tomás Darquier, Pablo Alejandro Virgolini.
Investigación: Tomás Darquier, Pablo Alejandro Virgolini.
Metodología: Tomás Darquier, Pablo Alejandro Virgolini.
Administración del proyecto: Tomás Darquier, Pablo Alejandro Virgolini.
Recursos: Tomás Darquier, Pablo Alejandro Virgolini.
Software: Tomás Darquier, Pablo Alejandro Virgolini.
Supervisión: Tomás Darquier, Pablo Alejandro Virgolini.
Validación: Tomás Darquier, Pablo Alejandro Virgolini.
Visualización: Tomás Darquier, Pablo Alejandro Virgolini.
Redacción – borrador original: Tomás Darquier, Pablo Alejandro Virgolini.
Redacción – revisión y edición: Tomás Darquier, Pablo Alejandro Virgolini.