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.

Cite as: 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

Submitted: 16-08-2023 **Revised:** 04-01-2024 **Accepted:** 24-05-2024 **Published:** 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.

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.

INTRODUCTION

The development of modern applications poses increasingly complex challenges, especially when distributed architectures such as microservices are adopted.[1,2,3,4] This approach, although highly beneficial in terms of scalability, modularity, and maintenance, introduces significant difficulties during the design, implementation, and integration of the various components.[5,6,7,8] Faced with this reality, this project proposed as a solution the design and implementation of a low-code platform aimed at facilitating the creation of microservice architectures in a visual, intuitive, and customizable way, targeting software developers in particular.[9,10,11]

With the aim of achieving a functional and flexible product, the agile **Scrum** methodology was adopted, which allowed for the incremental development of the system through iterations known as Sprints. This strategy facilitated the incorporation of continuous improvements and rapid adaptation to technical obstacles, technology changes, or design reorientations, without compromising the overall progress of the project.[12,13,14]

During its implementation, the system integrated a variety of technologies organized into different levels. For the **front-end**, HTML5, CSS, and JavaScript were used, complemented by Bootstrap and Thymeleaf, allowing for a smooth and adaptable user experience. On the **back end**, Java with Spring Framework was used to build a robust and secure foundation, with authentication via OAuth2 provided by Okta. Tools such as PostgreSQL for data management, Apache Kafka for asynchronous communication, and MinIO for storing generated code were also incorporated.[15,16,17]

One of the most innovative features of the system was the automatic generation of code based on user interaction with the canvas. This functionality was made possible by the use of semantic technologies such as RDF and SPARQL, managed by Apache Jena and powered by Apache Velocity for the creation of customizable files.[18,19,20]

Data collection for the design and validation of the project combined specialized academic sources and technical discussion forums on social networks such as Reddit, Twitter, and Medium. This fusion of theory and practice allowed for a better understanding of the problem and informed the technological decisions made.

In summary, the platform developed seeks to lower the technical barrier in the design of distributed systems, promoting the use of reusable and configurable components, all within a low-code visual environment capable of generating architectures ready for deployment with minimal manual intervention.

How can we facilitate the design, configuration, and integration of microservice architectures using a low-code platform that reduces technical complexity and development time without compromising the quality of the software generated?

Objective

To develop a low-code platform that allows developers to design, configure, and integrate microservice architectures in a visual and intuitive way, using reusable components, automating code generation, and optimizing development times with validations that ensure system quality.

METHOD

Methodological Design

Methodological Tools

During the development of this system, the guidelines established by the agile Scrum methodology were followed, a framework that facilitates collaboration between teams to deliver products in an iterative and incremental manner, allowing for rapid adaptation to changes and encouraging continuous improvement.

In this way, at the end of each Sprint, functional portions of code were obtained, even though the complete system was not developed, learning from each iteration and applying the knowledge in the next one. Thus, the product benefited from the aforementioned agile characteristics, even allowing for a change in technologies based on the knowledge obtained during the development process without any repercussions.

Development Tools

Multiple technologies were used in the development of the project, which will be explained and justified below, organized into four groups according to their specific activity: front-end tools, those intended for the back-end for basic logical authentication and exchange with the front-end embedded in the gateway, back-end technologies for automatic code generation, and finally, technologies used to facilitate system deployment and scalability.

The tools used in the development of the front-end include the now standard set of JavaScript, HTML5, and CSS, allowing for the creation of a complete visual appearance with high compatibility, as well as correct and adaptable communication logic with the back-end. In turn, these tools were enhanced with Bootstrap, which, with the help of its pre-designed components, alleviated the workload related to aesthetic design, allowing multiple functionalities to be managed within the same page without neglecting this aspect. In this way, and with the help of Thymeleaf, the front-end was coupled to the API Gateway to avoid unnecessary complexities.

The back-end was developed mostly in Java 17, using the Spring framework and its well-known libraries for correct, clean, and efficient communication between services, mostly using the REST architecture style. Security was managed and implemented using OAuth2 based on the service offered by Okta, a standard that allows websites or applications to access resources hosted in other applications on behalf of and with the prior permission of the user. Aspects requiring relational data persistence were provided by a PostgreSQL database instance, chosen for its open source environment, as well as its remarkable flexibility, data integrity, and scalability. In addition, asynchronous communications were handled for sending notifications using Apache Kafka.

The logic related to dynamic code generation was managed semantically using RDF, which allowed the combinations and decisions made by the user on the front-end canvas to be specified correctly and in a structured manner, so that they could be sent to the back-end, queried using the SPARQL query language, and managed by Apache Jena in the Java services. To complete the process and generate the code specified and requested by the user, we decided to use Apache Velocity, due to its high capacity for the task required, and MinIO for storing and downloading the resulting files, thus avoiding dependence on specific cloud storage solutions.

Finally, the deployment and scalability of the system was managed using Docker, powered by Kubernetes. In this way, as with other decisions outlined above, dependence on cloud solutions is reduced, giving the system greater versatility and independent deployment capabilities.

Data Collection

To properly understand the problem to be addressed, we began by analyzing academic bibliographic sources. This approach allowed us to identify the main challenges in the development of distributed architectures, as well as the associated costs.

Another data collection method used was social media analysis. Due to the public nature of the project, this methodology proved highly enriching thanks to platforms such as Twitter, Reddit, and Medium, where software developers share and discuss ideas and experiences on topics and situations in the field of computing.

This resulted in two clearly contrasting approaches, thus justifying the choice of techniques presented. These approaches are divided into strictly theoretical ones, based on academic documents, and, at the other extreme, those based on social networks, which draw on the daily experiences of developers, who, at the end of the day, are the ones affected by the problem.

Project Planning

For ease of understanding, the planning will first be presented separately, followed by the Gantt chart used to organize the objectives of this Final Graduation Project.

Next, the tasks specified are presented together with the corresponding dates, followed by the aforementioned Gantt chart.

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

Figure 1. Development Plan

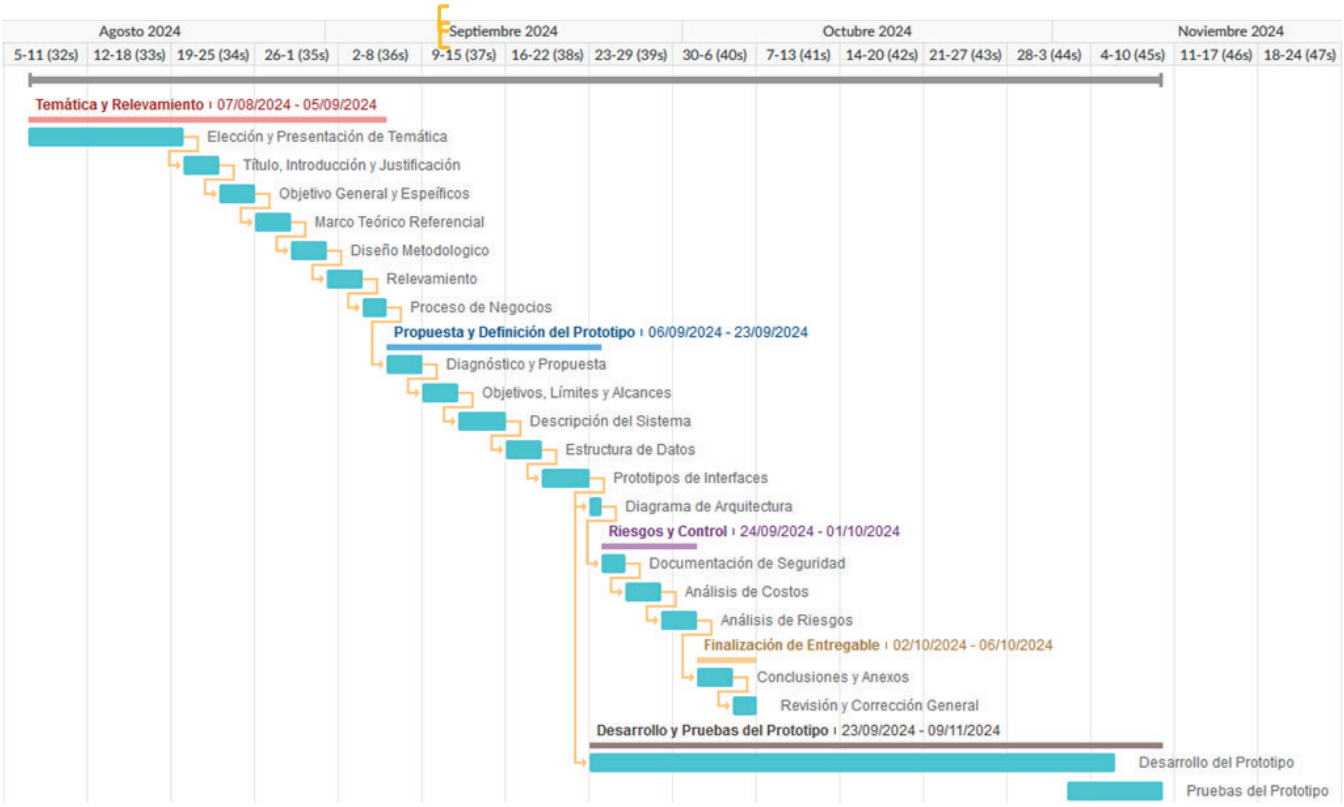

Figure 2. Gantt chart

RESULTS

System Description

Product Backlog

The Product Backlog for the prototype to be developed is presented below. It should be noted that the user stories shown correspond only to the basic functionalities necessary to demonstrate the purpose of the system. For this reason, only customizable e-commerce components are presented, as this allows the functionalities related to the combination and customization of compatible components within the application canvas to be demonstrated.

Table 1. Product Backlog

ID	User Story	Priority	Story Points	Dependencies
HU-001	Registration on the platform via Google	Registration	5	-
HU-002	Log in via Google	Sign up	2	HU-001
HU-003	Log out	Med	2	HU-002
HU-004	Drag-and-drop visual design	Media	5	-
HU-005	User service template	High	5	-
HU-006	User service	Registration	2	HU-005
HU-007	Product catalog service template	High	5	-
HU-008	Product catalog service	Registration	3	HU-007
HU-009	Shopping cart service template	Sign up	5	-
HU-010	Shopping cart service	Sign up	3	HU-009
HU-011	Order service template	High	5	-
HU-012	Order service	Registration	3	HU-011
HU-013	Shipping service template	Registration	5	-
HU-014	Shipping service	Medium	3	HU-013
HU-015	Notification service template	High	8	-
HU-016	Notification service	Medium	3	HU-015
HU-017	Interact with available components	Medium	5	HU-006
HU-018	Component compatibility	Medium	5	HU-017
HU-019	Remove components from the design	Low	3	HU-018
HU-020	Select persistence in components	Low	3	HU-016
HU-021	Configure endpoints in components	Media	3	HU-016
HU-022	Select communication methodology	Media	8	HU-016
HU-023	Specify JWT security	High	3	HU-016
HU-024	Specify global configuration	Medium	5	HU-016
HU-025	Specify API Gateway	Medium	8	HU-016
HU-026	JSON to RDF translation	High	5	HU-025
HU-027	ZIP of the resulting architecture	Low	8	HU-026
HU-028	Dynamic generation process status	Low	5	HU-027
HU-029	Access ZIP files	Download	3	HU-027
HU-030	Component Dockerfiles	High	3	HU-026
HU-031	Pending tasks guide	Low	5	HU-025
HU-032	Component documentation	Media	5	HU-025
HU-033	Platform user guide	Media	5	HU-032

User stories

Table 2. HU-001 Registration on the platform via Google

ID	HU-001	Name	Registration on the platform via Google
Description		As a user, I want to register on the system using my Google account so that I can create my profile.	
Criteria for Accept		Given an existing Google user, when they decide to to register, the system must redirect them to a tab to provide the necessary permissions, allowing registration and informing the user once the action is complete. Given a Google user previously registered through their account on the platform, when they are prompted to register, the system must inform them that an account already exists with that user, inviting them to log in through a hyperlink to the corresponding section.	
Priority		High	Estimated history points 5

Table 3. HU002 Log in via Google

ID	HU-002	Name	Log in via Google
Description		As a user, I want to log in to the system using my Google account to access the platform and my previous architectures.	
Acceptance Criteria		Given a user who is already registered with their Google account, when they log in using that account, the system should grant access and redirect them to the platform canvas. Given a user who is not registered with their Google account, when they attempt to log in using the unregistered account, the system must inform them that they are not registered, inviting them to do so via a hyperlink to the corresponding section.	
Priority		High	Estimated story points 2

Table 4. HU-003 Close user session

ID	HU-003	Name	Log out user
Description		As a user, I want to log out of the system using my Google account to prevent unwanted access to my profile.	
Acceptance Criteria		Given a user logged into the system using valid credentials, when they decide to log out, the system must revoke the temporary credentials that allow access to the current session.	
Priority		Medium	Estimated history points 2

Table 5. HU-004 Drag-and-drop visual design

ID	HU-004	Name	Drag-and-drop visual design
Description		As a user, I want to drag components onto the canvas so that I can design an architecture visually.	
Acceptance Criteria		Given a logged-in user, when the user presses and drags the visual representation of a component onto the canvas, the system must move it to the area where the user drops the element.	
Priority		Medium	Estimated story points 5

Table 6. HU-005 User service template

ID	HU-005	Name	User Service Template
Description		As a user, I want the system to use a predefined template for the automatic creation of user services, so that all the components necessary to manage users and permissions without manual intervention.	
Acceptance Criteria		Given a user who has logged into the platform, when they generate an architecture that contains "User Service" within the design canvas, the system must automatically associate the service with a predefined template for user management. Given a user who has logged into the platform, when they combine a certain service compatible with "User Service," the system must adapt the code generation accordingly for proper interaction.	
Priority		High	Estimated story points 5

Table 7. HU-006 User services

ID	HU-006	Name	User service
Description		As a user, I want to have a user management service to incorporate into my architecture, so that I can manage their information and permissions.	
Acceptance Criteria		Given that a user has logged into the platform, when dragging the "Users" visual component to the design canvas, the system must automatically associate the order service to the predefined code template.	
Priority		High	Estimated story points 3

Table 8. HU-007 Product catalog service template

ID	HU-007	Name	Product catalog service template
Description		As a user, I want the system to use a predefined template for the automatic creation of a product catalog service so that I can display my current inventory.	
Acceptance Criteria		Given a user who has logged into the platform, when they generate an architecture that contains "Product Catalog Service" within the design canvas, the system must automatically associate the service with a predefined template for user management. Given a user who has logged into the platform, when they combine a certain service compatible with "Product Catalog Service," the system must adapt the code generation accordingly for proper interaction.	
Priority		High	Estimated story points 5

Table 9. HU-008 Product catalog service

ID	HU-008	Name	Product catalog service
Description		As a user, I want to have a product catalog service to incorporate into my architecture so that I can display my current stock to the public.	
Acceptance Criteria		Given that a user has logged into the platform, when they drag the "Product Catalog" visual component onto the design canvas, the system must automatically associate the order service with the code.	
Priority		High	Estimated story points 3

Table 10. HU-009 Shopping cart service template

ID	HU-009	Name	Shopping cart service template
Description		As a user, I want the system to use a predefined template for the automatic creation of a shopping cart service, so that users can make multiple product purchases.	
Acceptance Criteria		Given a user who has logged into the platform, when they generate an architecture that contains "Shopping Cart Service" within the design canvas, the system must automatically associate the service with a predefined template for user management. Given a user who has logged into the platform, when they combine a certain service compatible with "Shopping Cart Service," the system must adapt the code generation code generation accordingly for proper interaction.	
Priority		High	Estimated story points 5

Table 11. HU-010 Shopping cart service

ID	HU-010	Name	Shopping cart service
Description		As a user, I want to have a shopping cart service to incorporate into my architecture so that users of my system can purchase multiple items simultaneously.	
Acceptance Criteria		Given that a user has logged into the platform, when they drag the "Shopping Cart" visual component onto the design canvas, the system must automatically associate the order service with the code.	
Priority		High	Estimated story points 3

Table 12. HU-011 Order service template

ID	HU-011	Name	Order Service Template
Description		As a user, I want the system to use a predefined template for the automatic creation of an order service to manage the order issuance and administration process.	
Acceptance Criteria		Given a user who has logged into the platform, when they generate an architecture that contains "Order Service" within the design canvas, the system must automatically associate the service with a predefined template for user management. Given a user who has logged into the platform, when they combine a certain service compatible with "Order Service," the system must adapt the code generation accordingly for proper interaction.	
Priority		High	Estimated story points 5

Table 13. HU-012 Order service

ID	HU-012	Name	Order service
Description		As a user, I want to have a service capable of managing orders to incorporate into my architecture, to handle orders issued within the system and act accordingly.	
Acceptance Criteria		Since a user is logged into the platform, when they drag the "Orders" visual component to the design canvas, the system must automatically associate the order service to the predefined code template.	
Priority		High	Estimated story points 3

Table 14. HU-013 Shipping service template

ID	HU-013	Name	Shipping Service Template
Description		As a user, I want the system to use a predefined template for the automatic creation of a shipping service so that I can manage the status of shipments.	
Acceptance Criteria		Given a user who is logged into the platform, when they generate an architecture that contains "Shipping Service" within the design canvas, the system must automatically associate the service with a predefined template for user management. Given a user who has logged into the platform, when they combine a certain service compatible with "Shipping Service," the system must adapt the code generation form accordingly for proper interaction.	
Priority		High	Estimated story points 5

Table 15. HU-014 Shipping service

ID	HU-014	Name	Shipping service
Description		As a user, I want to have a service capable of managing the status of my system's shipments to incorporate into my architecture, so that I can track and update the status of shipments.	
Acceptance Criteria		Since a user is logged into the platform, when they drag the "Shipments" visual component to the design canvas, the system must automatically associate the order service to the predefined code template.	
Priority		Medium	Estimated story points 3

Table 16. HU-015 Notification service template

ID	HU-015	Name	Notification Service Template
Description		As a user, I want the system to use a predefined template for the automatic creation of a notification service, so that it can send alerts and messages to users.	
Acceptance Criteria		Given a user who is logged into the platform, when they generate an architecture that contains "Notification Service" within the design canvas, the system must automatically associate the service with a predefined template for user management. Given a user who has logged into the platform, when they combine a certain service compatible with "Notification Service," the system must adapt the code generation code accordingly for proper interaction.	
Priority		High	Estimated story points 8

Table 17. HU-016 Notification service

ID	HU-016	Name	Notification service
Description		As a user, I want to have a notification service to incorporate into my architecture, to send messages and alerts to system users about important events.	
Acceptance Criteria		Given that a user has logged into the platform, when they drag the "Notifications" visual component onto the design canvas, the system must automatically associate the order service to the predefined code template.	
Priority		Medium	Estimated story points 3

Table 18. HU-017 View available components			
ID	**HU-017**	**Name**	**Interact with available components**
Description		As a user, I want to view the available components so that I can select and combine them on the canvas according to my needs.	
Acceptance Criteria		Given a user who has logged into the platform, when they place a component on the canvas and want to join it with another compatible component, the system must create a graphical link using a line to express the successful relationship. Given a user who has logged into the platform, when they place a component on the canvas and want to join it with another incompatible component, the system must display a pop-up indicating that the combination is not legal, inviting the user to review the platform documentation.	
Priority		Medium	Estimated story points 5

Table 19. HU-018 Component compatibility			
ID	**HU-018**	**Name**	**Component compatibility**
Description		As a user, I want to see which components are compatible with the selected component is compatible with, in order to combine them correctly on the canvas.	
Acceptance Criteria		Given a user who has logged into the platform, when they interact with the "+" button attached to the component on the canvas, the system should highlight the "+" buttons of the other compatible components on the canvas. Given a user who has logged into the platform, when they highlight the compatibilities of a component and tap the canvas again, the system must uncheck the previously highlighted connections.	
Priority		Medium	Estimated story points 5

Table 20. HU-019 Remove design components			
ID	**HU-019**	**Name**	**Remove design components**
Description		As a user, I want to delete a component from the canvas to remove it from the designed architecture.	
Acceptance Criteria		Given a user who is logged into the platform, when they place the mouse over a component displayed on the canvas, the system must display an "X" icon referring to the option to delete the component. Given a user who has logged into the platform, when they interact with the "X" button of a component on the canvas, the system must ask if they want to delete the component. If so, it executes the action, and if not, it closes the pop-up window no, it removes the query pop-up.	
Priority		Low	Estimated story points 3

Table 21. HU-020 Select persistence in components			
ID	**HU-020**	**Name**	**Select persistence in components**
Description		As a user, I want to configure the database of a component so that I can choose the one that best suits the requirements of my architecture.	
Acceptance Criteria		Given a user who has logged into the platform, when interacting with the default database attached to the component within the canvas, the system must display the alternative options available for the service, allowing the default to be modified.	
Priority		Low	Estimated story points 3

Table 22. HU-021 Configure endpoints in components			
ID	**HU-021**	**Name**	**Configure endpoints in components**
Description		As a user, I want to configure the URL of the endpoints of a component to suit my infrastructure requirements.	
Acceptance Criteria		Given a user who is logged into the platform, when they hover the mouse over a component on the canvas, the system must display a button representing a "nut" to access a pop-up with the service settings. Given a user who has logged into the platform, when they interact with the configuration button of a microservice, the system must present the option to modify the prefix of the microservice endpoints, applying the changes by clicking "accept."	

	Given a user who is logged into the platform, when they enter an invalid string as the endpoint prefix and press "accept," the system should notify them that no changes were made due to invalid characters, keeping the user in the pop-up displaying the string prior to the illegal entry.		
Priority	Medium	Estimated story points	3

Table 23. HU-022 Select communication methodology

ID	**HU-022**	**Name**	**Select communication methodology**
Description		As a user, I want to select the communication methodology between two components so that they adapt to the needs of my system.	
Acceptance Criteria		Given a user who has logged into the platform, when creating a connection between two compatible components, the system must display a symbol in the middle of the connection that, when interacted with, will display a list of the connection types compatible between both microservices , allowing the desired option to be selected.	
Priority	Medium	Estimated story points	8

Table 24. HU-023 Specify JWT security

ID	**HU-023**	**Name**	**Specify JWT security**
Description		As a user, I want to specify the use of JWT as a security measure so that the security of the architecture meets my requirements.	
Acceptance Criteria		Given a user who has logged into the platform, when accessing the canvas, the system must display a global option in the upper right corner to secure the architecture using JWT, allowing it to be disabled or enable it.	
High	High	Estimated story points	3

Table 25. HU-024 Specify global configuration

ID	**HU-024**	**Name**	**Specify global configuration**
Description		As a user, I want to specify the use of a global configuration server so that the architecture fits my requirements.	
Acceptance Criteria		Given a user who has logged into the platform, when accessing the canvas, the system must display an option in the upper right corner, below "JWT," to manage all service configurations through a configuration server, allowing disable or enable it.	
Priority	Medium	Estimated story points	5

Table 26. HU-025 Specify API Gateway

ID	**HU-025**	**Name**	**Specify API Gateway**
Description		As a user, I want to specify the generation of an API gateway so that components can interact directly	
Acceptance Criteria		Given a user who has logged into the platform, when accessing the canvas, the system must display an option in the upper right corner, under "Config Server," to manage all architecture interactions through an API gateway, allowing it to be disabled or enabled.	
Priority	Medium	Estimated story points	8

Table 27. HU-026 Translation from JSON to RDF

ID	**HU-026**	**Name**	**Translation from JSON to RDF**
Description		As a user, I want the system to automatically convert data that specifies the generated architecture from JSON format to RDF to ensure a structured specification.	
Acceptance Criteria		Given a user who has logged into the platform, when requesting the processing of the resulting architecture, the system must represent the information using JSON, subsequently translating it into semantic RDF content to provide structure and enable queries.	
Priority	High	Estimated story points	5

Table 28. HU-027 ZIP of the resulting architecture				
ID	**HU-027**	**Name**	**ZIP of the resulting architecture**	
Description		As a user, I want to request a ZIP file from the canvas with the designed architecture to obtain the files that comprise it and thus configure it in my local environment.		
Acceptance Criteria		Given a user who has logged into the platform, when configuring an architecture and requesting its generation, the system must convert the generated code into a ZIP file, informing the user when the process is complete and inviting them to download it from their user profile user profile.		
Low		Low	Estimated story points	8

Table 29. HU-028 Status of dynamic generation process				
ID	**HU-028**	**Name**	**Dynamic generation process status**	
Description		As a user, I want to know the status of the dynamic generation of my architecture in order to estimate when it will be completed.		
Acceptance Criteria		Given a user who has logged into the platform, when configuring an architecture and requesting its generation, the system must display a pop-up indicating the status of the process, showing the following statuses: Creating the pom.xml Generating the code Compressing the architecture Process completed		
Priority		Low	Estimated story points	5

Table 30. HU-029 Access previous ZIP files				
ID	**HU-029**	**Name**	**Access ZIP files**	
Description		As a user, I want to access my ZIP files generated so that I can download them again		
Acceptance Criteria		Given a user who has logged into the platform, when selecting the profile symbol in the upper right corner, the system should redirect the user to a screen displaying their last 5 generated architectures, allowing them to be downloaded with a simple click on the button attached to the right of each one.		
Priority		Low	Estimated history points	3

Table 31. HU-030 Component Dockerfiles				
ID	**HU-030**	**Name**	**Component Dockerfiles**	
Description		As a user, I want to have Dockerfiles associated with the components of the architecture to facilitate their deployment.		
Acceptance Criteria		Given a user who has logged into the platform, when obtaining the zip file resulting from an architecture, the system must include within that file an individual Dockerfile for each component of the architecture, to allow for its deployment and that of its persistence component.		
Priority		High	Estimated story points	3

Table 32. HU-031 Guide to pending tasks				
ID	**HU-031**	**Name**	**Guide to pending tasks**	
Description		As a user, I want to obtain a to-do guide (TO-TO's) in the generated code to adjust and customize the downloaded architecture.		
Acceptance Criteria		Given a user who has logged into the platform, when obtaining the zip file resulting from an architecture, the system must include within that file a .txt file explaining the TO-DO's present in the code and their reason for existence.		
Priority		Low	Estimated story points	5

Table 33. HU-032 Component documentation

ID	HU-032	Name	Component documentation
Description		As a user, I want to access the component documentation to understand its structure and detailed operation.	
Acceptance Criteria		Given a user who has logged into the platform, when they click on the documentation section at the top of the screen, the system should redirect them to a screen with a side menu where the available components are listed. Given a user who has logged into the platform, when they click on a specific component in the documentation, the system must display it, specifying its functionality, compatibilities, and endpoints.	
Priority		Medium	Estimated story points 5

Table 34. HU-033 Platform user guide

ID	HU-033	Name	Platform user guide
Description		As a user, I would like to access a user guide for the platform to understand how it works and get the most out of it.	
Acceptance Criteria		When a user who is logged into the platform clicks on the section referring to the platform user guide at the top of the screen, the system should direct them to the corresponding screen, where a brief user guide for the platform is presented.	
Priority		Medium	Estimated history points 5

Sprint Backlog

The following table shows the Backlog for the first Sprint of the prototype to be developed, with a set duration of 14 days.

Table 35. Sprint Backlog

Sprint	User Story	Tasks	Priority	Estimated (days)	Status
1	Registration on the platform via Google (HU-001)	Define and implement data persistence methodology and structure	High	1	Done
1		Code and integrate registration and connection module with Google OAUTH2	Registration	2	Done
1		Design graphical interface	Medium	1	Done
1		Perform unit testing on the developed module	High	1	Done
1	Log in via Google (HU-002)	Encode and integrate login module	Registration	1	Done
1		Design graphical interface	Medium	½	Done
1		Perform unit testing on the developed module	High	½	Done
1	Close user session (HU-003)	Encode and integrate logout module	Medium	1	Done
1		Design graphical interface	Medium	1	Done
1		Perform unit testing on the developed module	Medium	1	Done
1	User service template (HU- 005)	Design layout and compatibility for the user service template	High	1	Completed
1		Encode user service template	Register	2	Done
1		Perform unit testing on the components generated as a result of using the developed template	High	1	Done

Data Structure

The project has four data structures that support its processes and interactions, providing the system with everything from correct information persistence to a notable improvement in overall system efficiency and the ability to send asynchronous notifications to users within the platform.

Below is the DER diagram of the PostgreSQL relational database, which handles user administration and management and their corresponding identifiers, including those provided by the OAUTH2 identification server.

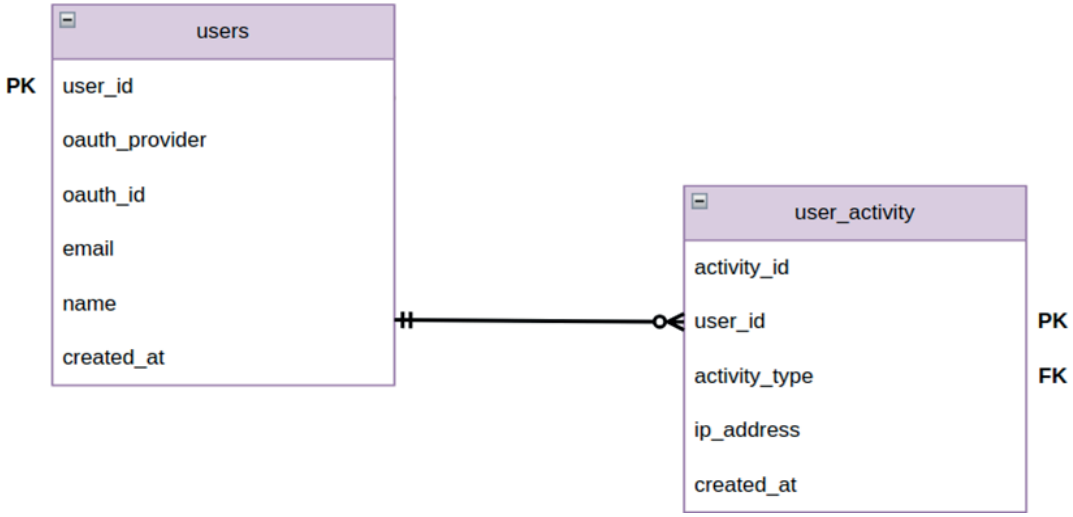

Figure 3. DER relational database for user management

The data dictionaries for the relational database just presented are shown below:

Table 36. Users table dictionary

Table: users			
Field	**Length**	**Data type**	**Description**
user_id	19	Numeric	User identifier
oauth_provider	16	Alphabetic	OAUTH provider
oauth_id	19	Numeric	Identification number provided by OAUTH
email	254	Alphabetic	Email provided by OAUTH provider
name	254	Alphabetical	Name provided by OAUTH provider
created_at	19	Timestamp	Time of account creation

Table 37. user_activity table dictionary

Table: user_activity			
Field	**Length**	**Data type**	**Description**
activity_id	19	Numeric	Session identifier
user_id	19	Numeric	User identifier
activity_type	254	Alphabetic	Type of activity performed
ip_address	15	Numeric	IPv4 address
created_at	19	Timestamp	Time the activity was performed

During the design of the system, it was found that including temporary cache storage could provide a temporary reduction in the system's response to the user, taking advantage of previously performed processing. For this reason, temporary storage was implemented with REDIS, which seeks to reduce calls to the Spring Initializr API by analyzing previous calls in a short period of time in search of one that is similar to the one to be made.

The following diagram illustrates the process mentioned above:

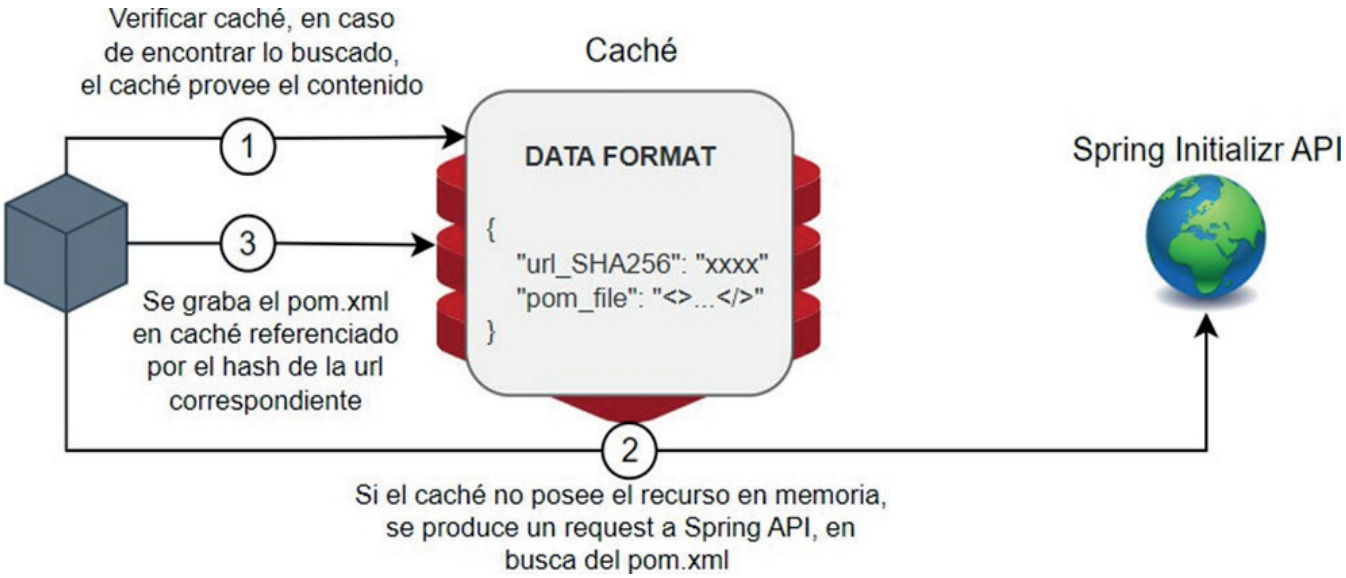

Figure 4. Explanation of cache system functionality

In this way, in many cases, valuable time consumed by a call to an external API is saved. The simple data structure of the cache mentioned above is shown below:

init_request

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

Figure 5. Cache data structure for poms generation

The dictionary for the data structure presented is shown below

Table 38. Dictionary of data present in the cache

Field	Length	Data type	Description
url_SHA256	64	Alphabetic	SHA-256 hash of the URL used for the request to the Spring Initializr API
pom_file	65535(max)	Alphabetical	Contents of the pom.xml file resulting from the request

A data structure was also defined to store the architectures designed by users on the platform and their corresponding automatically generated code. In this way, using MinIO, the system has a platform where it can work dynamically with the code, allowing users to download a .zip file with the designed architecture. The aforementioned data structure complies with the format of figure 6:

In addition, for proper functioning in asynchronous processes, the system required the inclusion of a messaging system. In this case, Apache Kafka was chosen. This tool performs two crucial tasks for the system:

- Inform of the presence of a new architecture to be generated and provide its specification.
- Keeping the user updated on the generation process of the requested code, informing them of the status of the final product.
- The topics used for the aforementioned functions are service-generation-request and service-generation-status, respectively, which interact with the system services (figure 7).

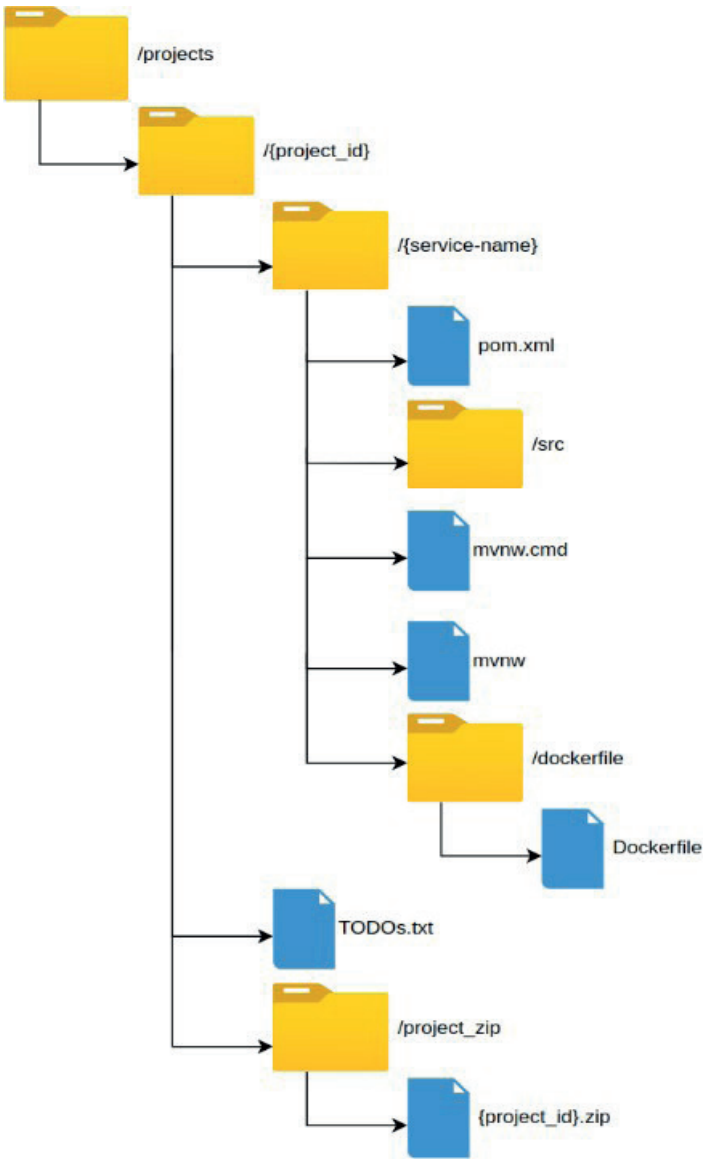

Figure 6. File structure for storing and generating code

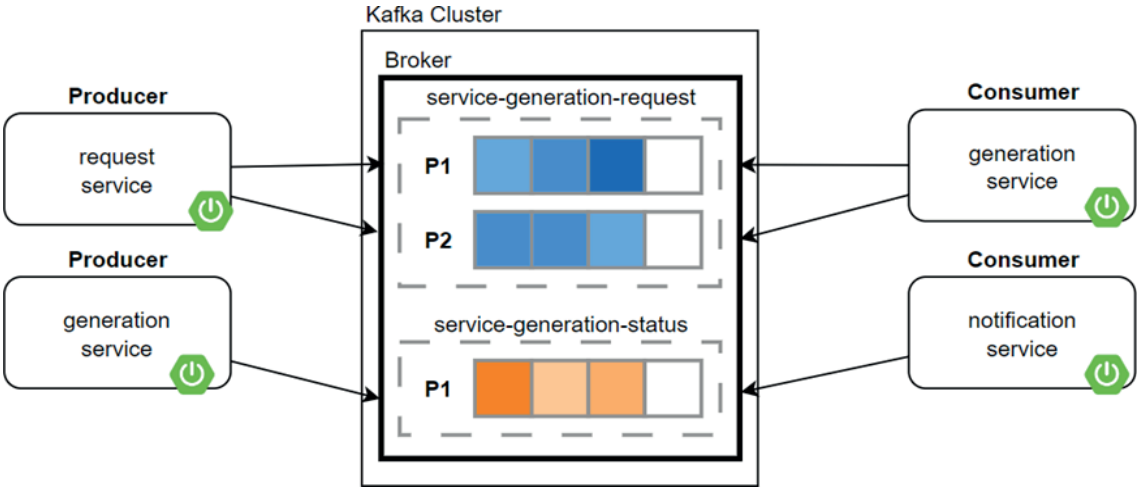

Figure 7. Diagram explaining the asynchronous message system

As can be seen in the diagram, the service consists of a single broker, and service-generation-request has two partitions, compared to service-generation-status, which only has one. This decision is based on the possibility of parallel processing of user-generated requests provided by the presence of more than one partition, which is not necessary in the case of code generation progress messages.

The data model for each topic mentioned is presented below.

Service-generation-status:

Using the information contained in the topic described above, the system is able to keep the user informed of the progress made in generating the code for the system they designed on the canvas.

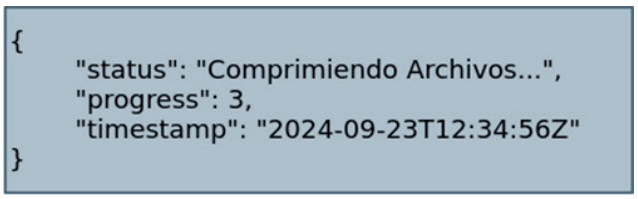

Figure 8. Data structure of the topic 'service-generation-status'

The dictionary for the data structure presented is shown below:

Table 39. Data dictionary for the 'service-generation-status' topic

Field	Length	Data type	Description
status	32	Alphabetic	Informative message about the dynamic code generation process
progress	1	Numeric	Number range from 1 to 3, indicating the step in the process where dynamic code generation is currently taking place.
timestamp	1	Alphabetic	Time of activity status update

Service-generation-request: the information carried by this defined topic is critical to the functioning of the system, since based on the JSON example below, the system asynchronously receives new tasks to be processed and builds the necessary code requested by the user. It is important to note that JSON has many attributes that may seem unnecessary for the prototype, as they do not allow customization, but this ensures that the system is ready for further customization in the event of full development.

The dictionary for the data structure presented is shown below:

Table 40. Dictionary of the 'service-generation-request' topic

Field	Data type	Description
projectName	Alphabetic	Name of the general project
version	Numeric	Project version, 1,0 by default
services	Alphabetical	Contains and defines the components of the architecture
name	Alphabetical	Component name
type	Alphabetical	Component type, i.e., what the service is about
version	Numeric	Version of the template used
description	Alphabetical	Component description
endpoints	Alphabetical	Section referring to the component's endpoints
path	Alphabetical	Base URL of the component
dependencies	Alphabetical	Component dependencies
connections	Alphabetical	Connections between components specified on the canvas
source	Alphabetical	Point A of the connection
target	Alphabetical	Connection point B
protocol	Alphabetical	Communication protocol
type	Alphabet	Methodology used for communication
databases	Alphabetical	Contains the persistence elements of the services
name	Alphabetical	Name of the persistence element
owner	Alphabetical	Service that interacts with the persistence element
type	Alphabet	Tool used
version	Numeric	Version of the tool used
infrastructure	Alphabetical	Contains configuration and infrastructure elements
name	Alphabetical	Name of the infrastructure component

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

type	Alphabetic	Infrastructure component type
port	Numeric	Infrastructure component port
security	Alphabetic	Contains the global security elements of the system
auth	Alphabetical	Specifies how authentication is performed in the system
type	Alphabetic	Authentication methodology
configurations	Alphabetic	Contains information for security customization
secretKey	Alphabetical	Secret key used to secure the system

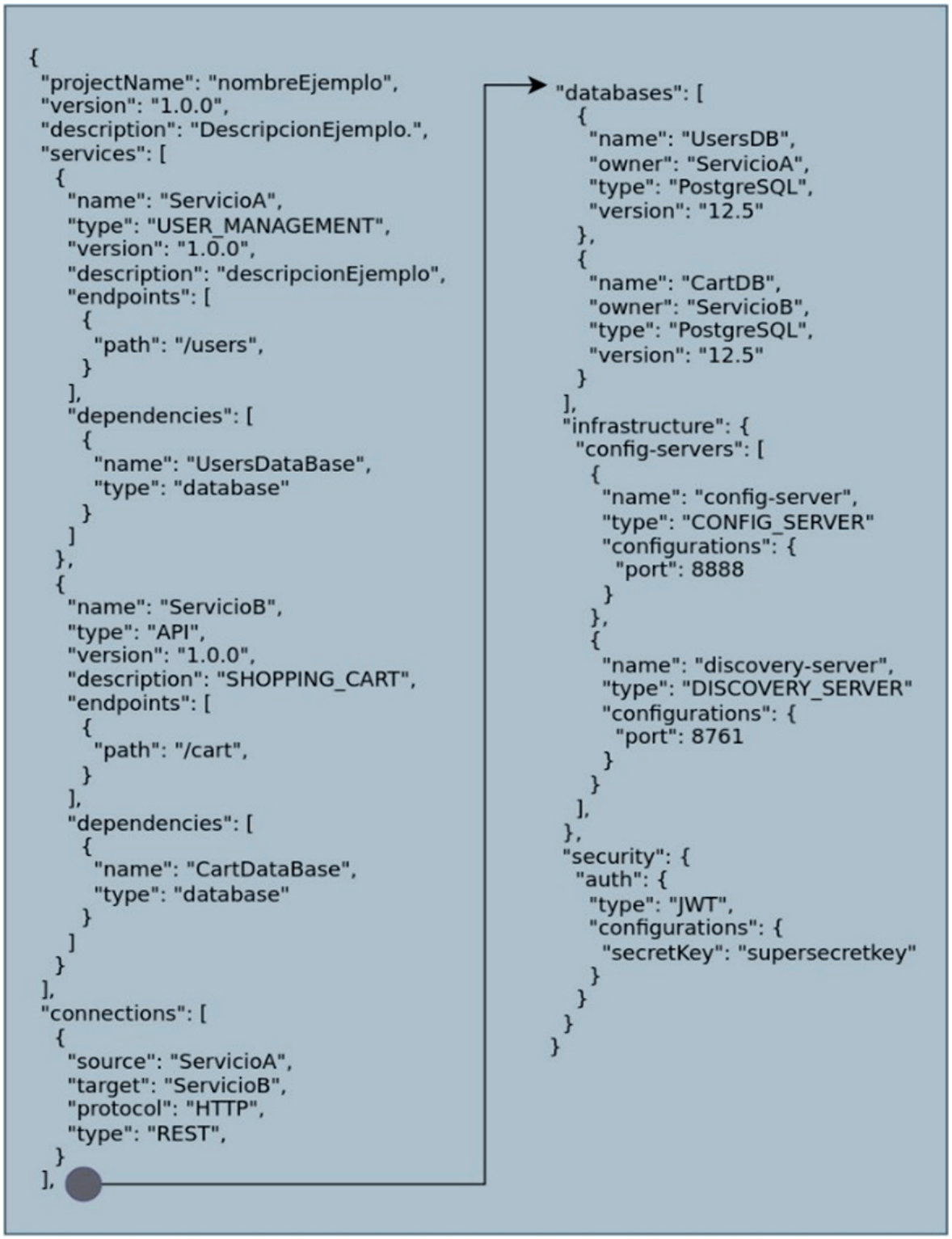

Figure 9. Data structure of the 'service-generation-request' topic

Screen Interface Prototypes

When accessing the web platform, users are required to log in using their Google account.

Figure 10. Login screen

Once the user has been validated and/or registered in the system, they are sent to the platform's home page, where its core functionality, related to distributed systems design, is located.

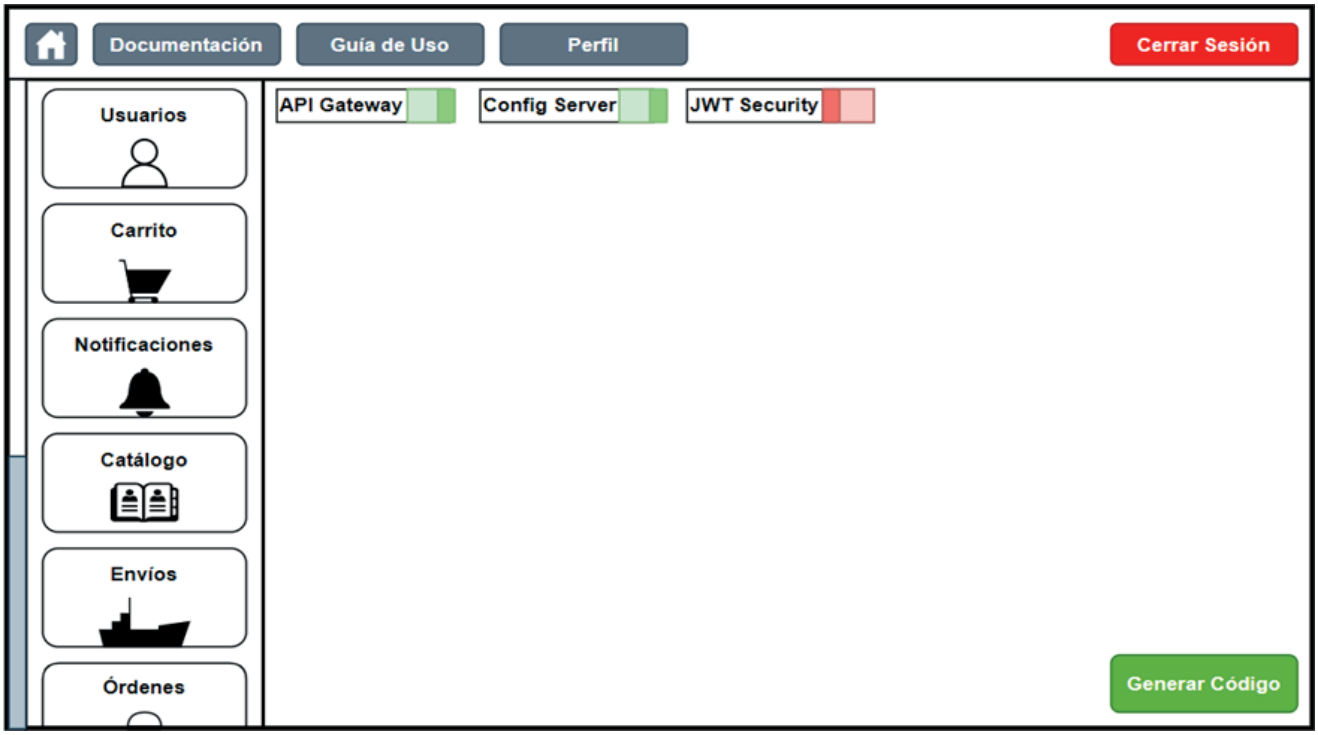

Figure 11. Home screen

On the main screen, the user can drag the components of the architecture onto the canvas to begin designing the system.

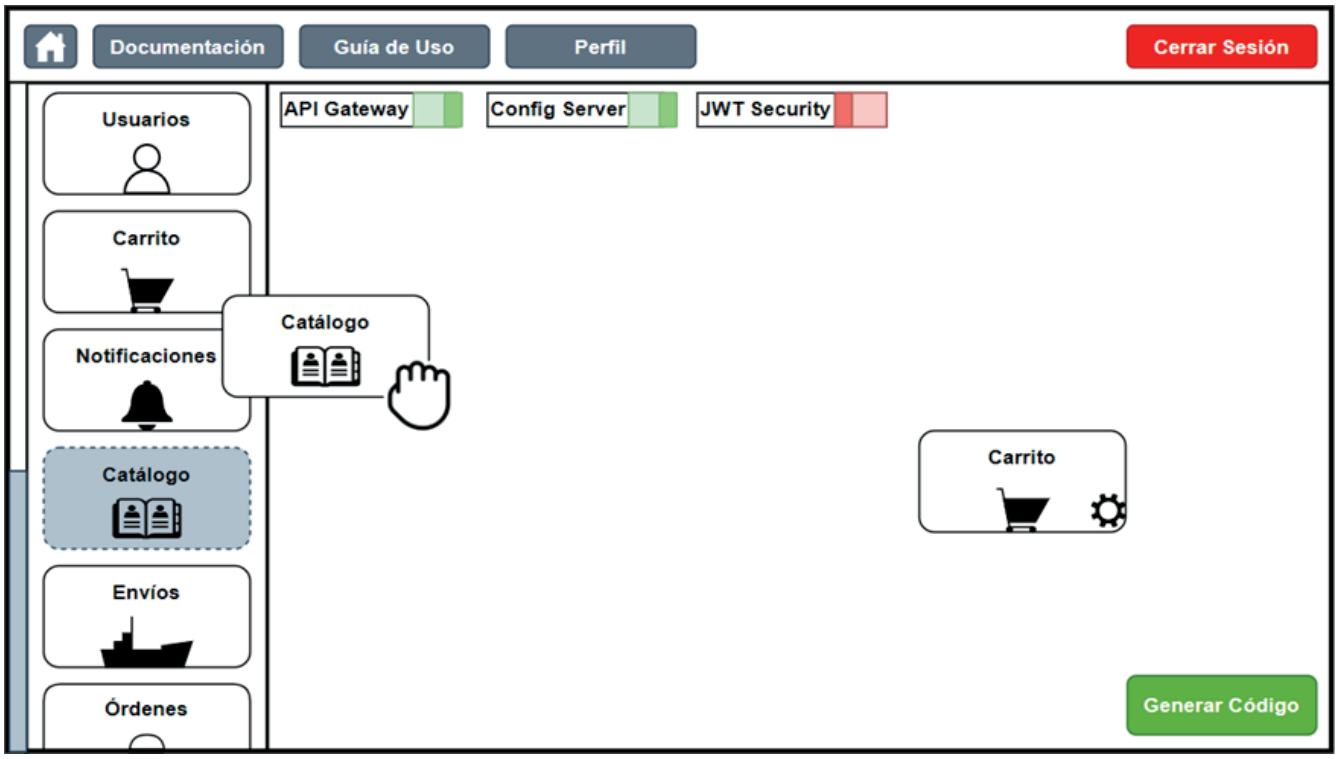

Figure 12. Drag-and-drop functionality

The components placed on the canvas can, if compatible, be joined together to generate more complex functionalities through their interaction.

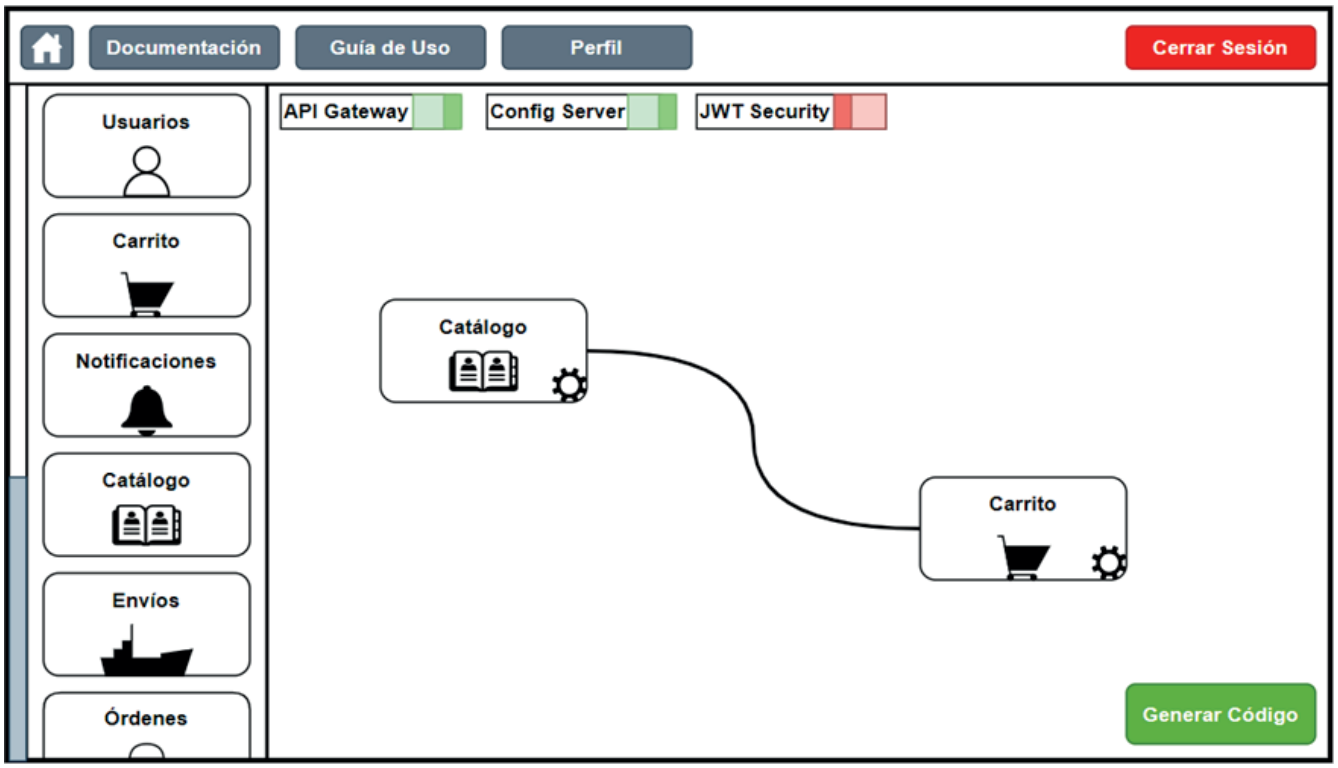

Figure 13. Component joining functionality

To customize each component placed on the canvas, the user must click on the corresponding gear icon to access the configuration menu.

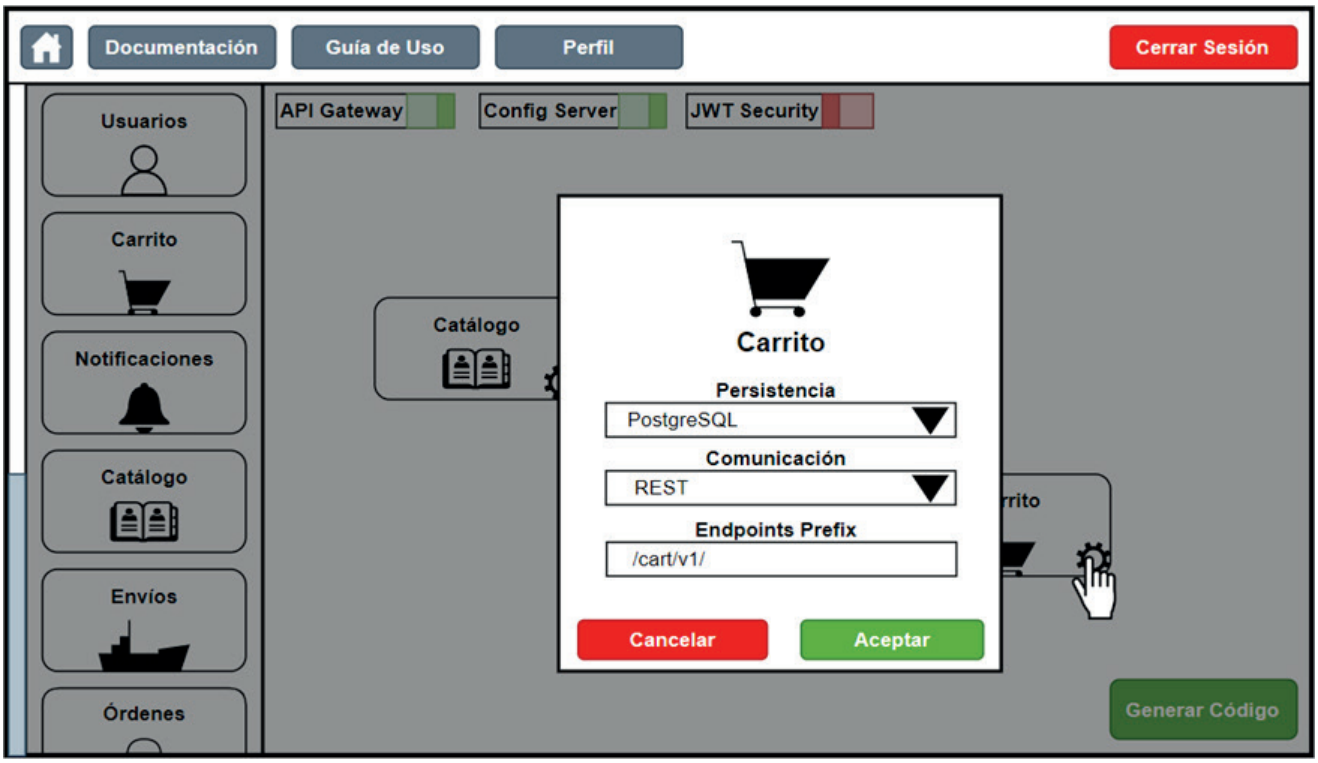

Figure 14. Component configuration screen

Once the user has designed and configured a specific architecture to their liking, they can generate the corresponding code by pressing the green button located in the lower right corner of the canvas, where they will receive real-time progress on the dynamic code generation process.

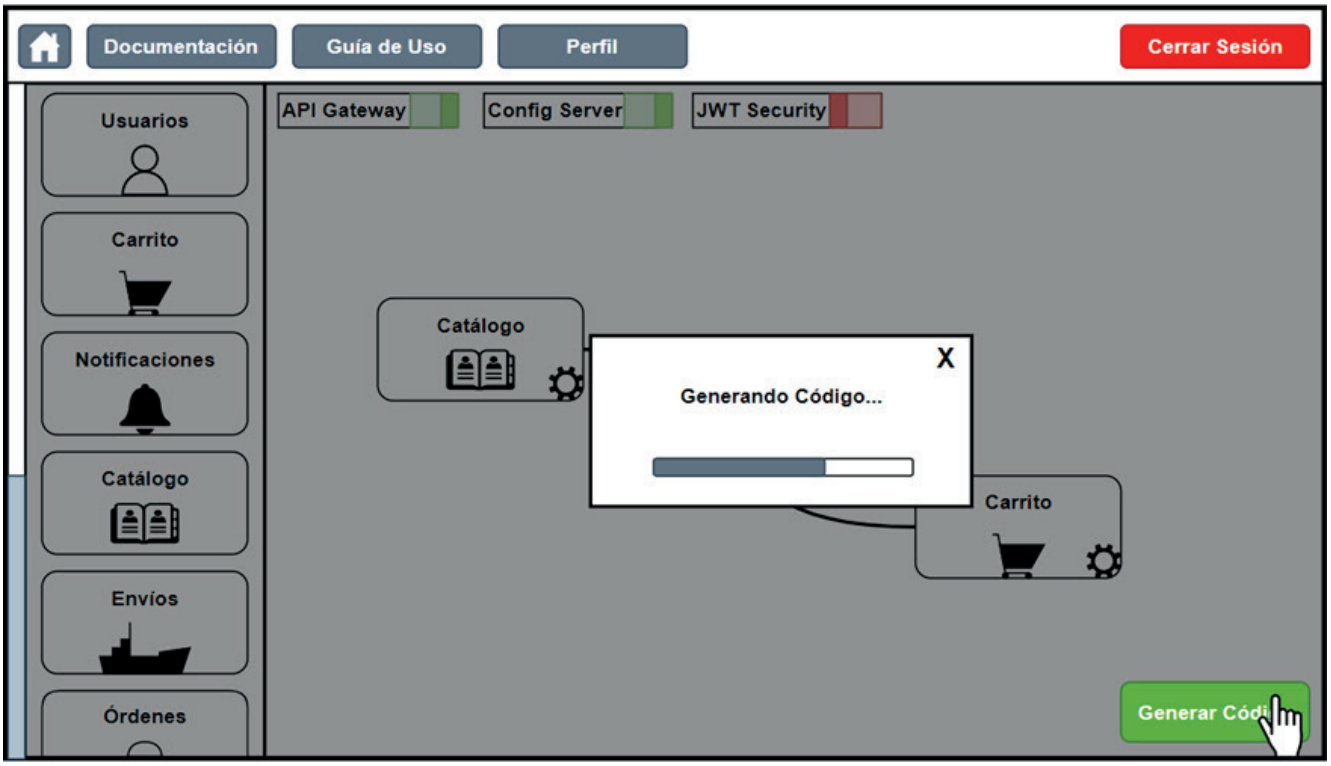

Figure 15. Code generation process screen

To learn about the different components that can be used on the canvas, their composition, operation, persistence methodology, and compatibilities, users can access the documentation section, where they will find the information necessary to understand each available piece.

Figure 16. Documentation screen

If users wish to understand how the platform works, there is a section called User Guide, where they can access the aforementioned information.

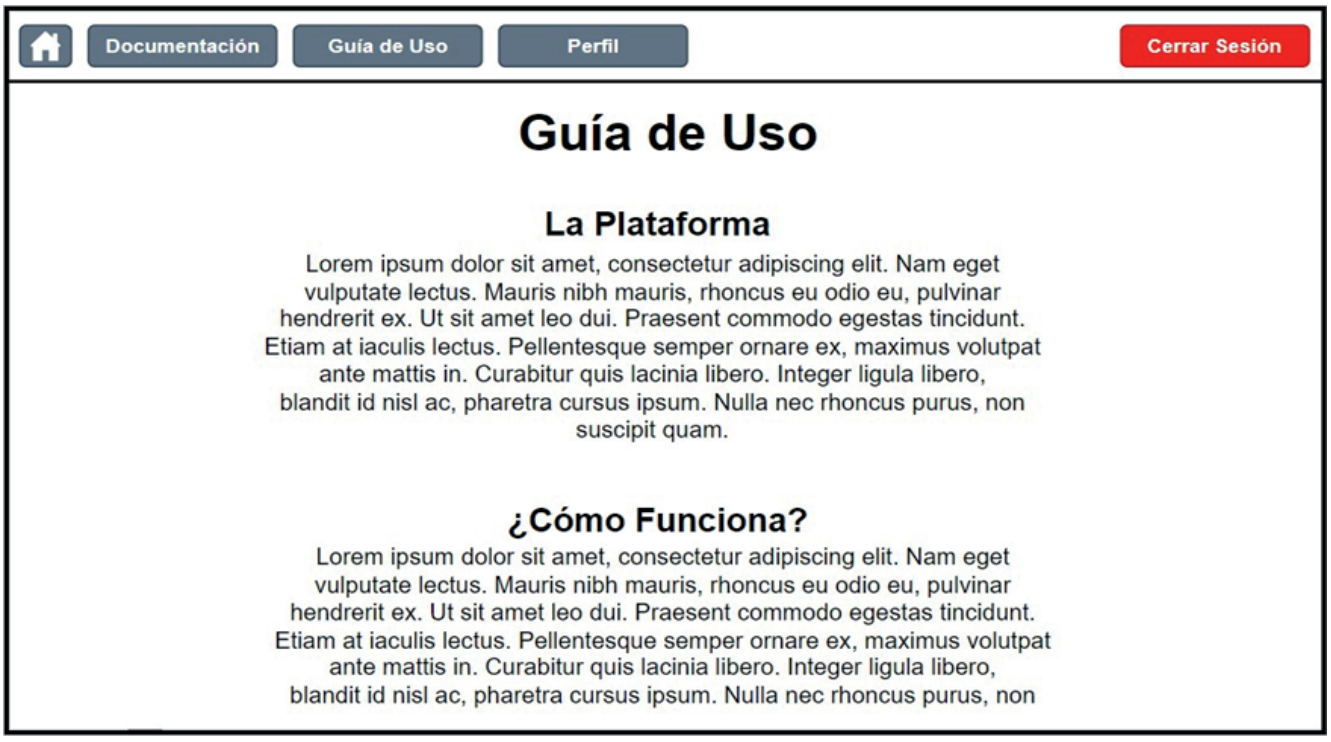

Figure 17. User guide screen

When the system user generates architectures on the canvas, the last five are saved and can be downloaded from the Profile section, which provides basic useful information about the architectures and a link to download them.

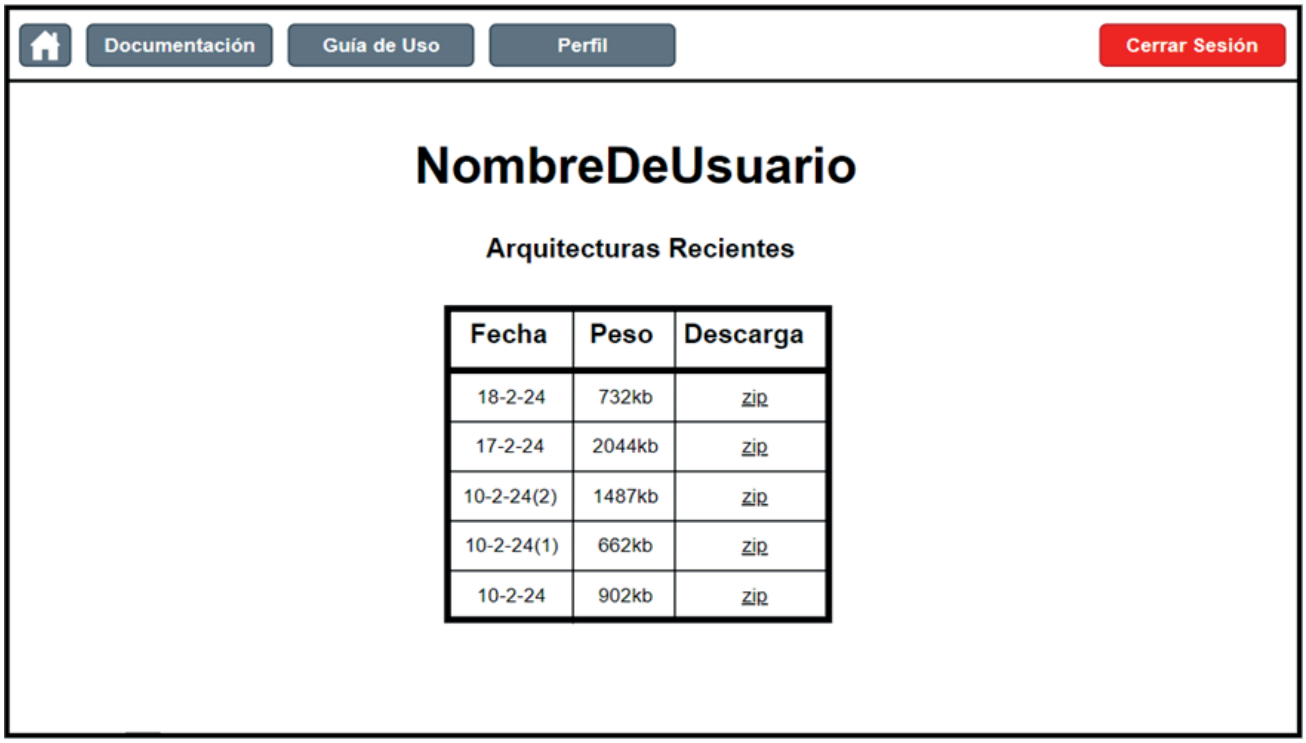

Figure 18. User profile screen

Finally, to conclude their activity on the platform, users can log out by clicking the button in the upper right corner after confirming that they wish to do so.

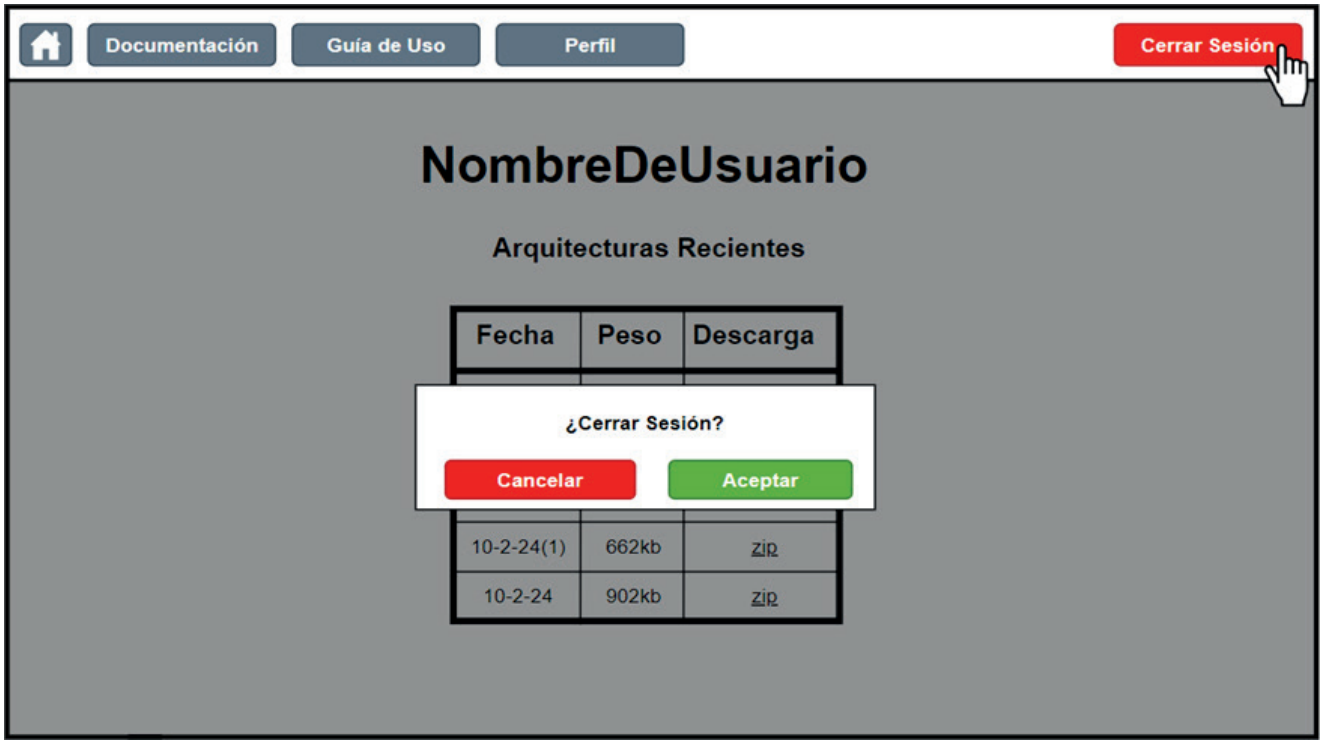

Figure 19. Logout screen

Architecture Diagram

The architecture diagram for the project discussed in this document is shown below. Due to the decisions made, as can be seen in the diagram, a highly scalable and efficient system was achieved for the intended purpose.

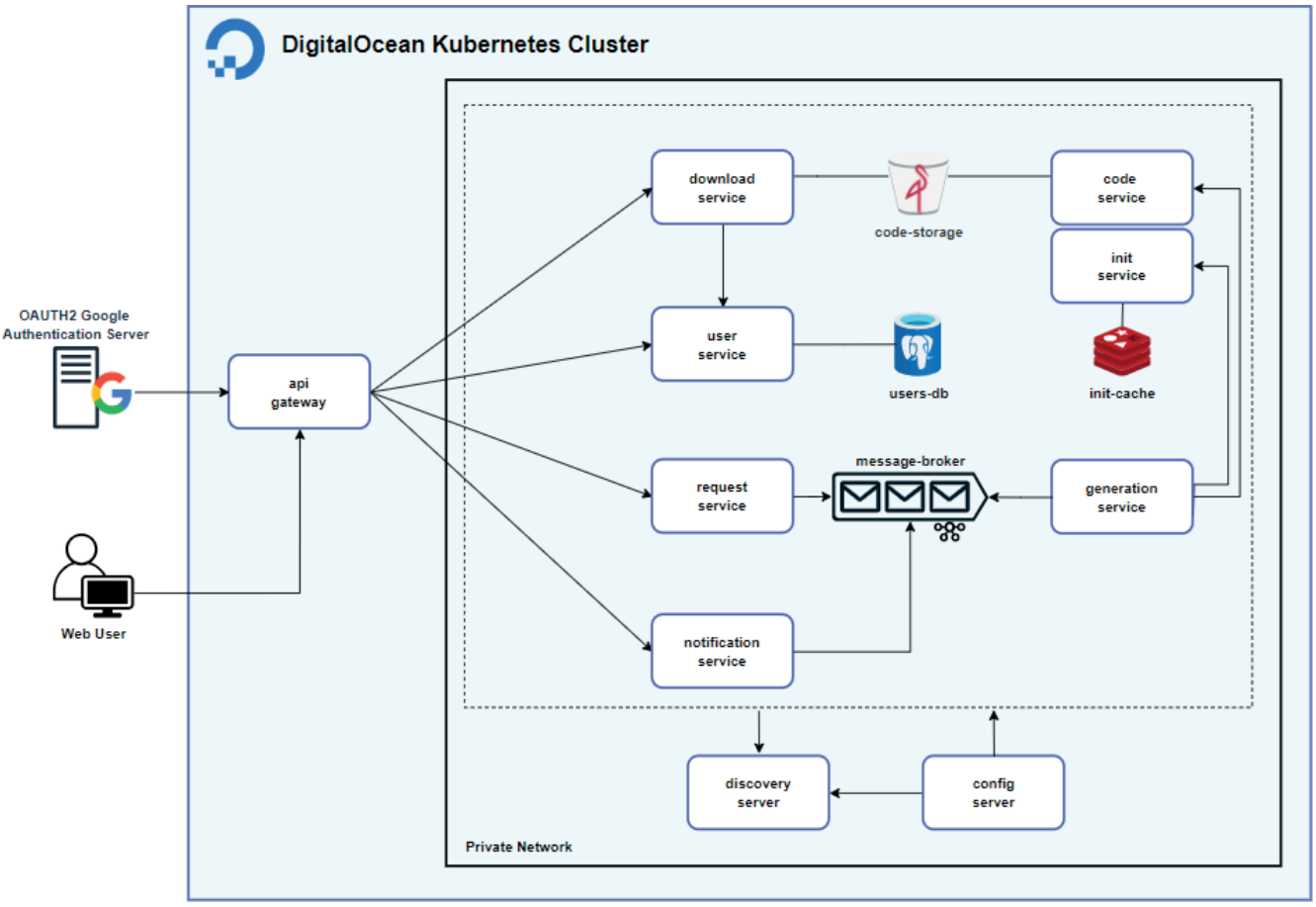

Figure 20. Architecture diagram

Security

Access to the Application

As stated repeatedly throughout this document, the project manages users through OAuth2, specifically through Google login, which is a very convenient alternative for an application such as the one developed, due to its permission for access by the general public.

Although for backup, auditing, and resource management purposes, the project itself stores the information of users registered on the platform, passwords are managed by Google, since it is through a user of the aforementioned platform that the system grants access to the project. Therefore, users must comply with Google's requirements for creating their credentials. Below is a list of Google's current requirements for creating an email address under its domain and the corresponding password.

Requirements for creating an email address:

- It must follow the standard format for an email address, i.e., nombre@dominio.com. In this case, the domain is @gmail.com.
- The username, i.e., the text before the '@', must contain between 6 and 30 characters.
- Only characters from a-z and A-Z are allowed, as well as numbers 0-9 and special characters such as the period (.), the underscore (_), and the hyphen (-).
- The email address must not be in use in Gmail. This means that each email address must be unique.

Requirements for specifying a password:

- The password must contain more than 8 characters.
- Although there is no related limitation, the use of lowercase letters, uppercase letters, numbers, and special characters is recommended.
- It must not be included in Google's list of common passwords.

In turn, Google stores user passwords using hash functions that convert the password into an irreversible text string, as well as adding a random value called a 'salt' to make brute force attacks more difficult. It is also worth noting that Google users can enable two-factor authentication, known as 2FA, which adds an extra layer of security by requiring a second element to access the user account.

With regard to the user roles outlined below, due to the nature of the project and the development of only core functionalities for the prototype, the administrator user has no advantage or differential capacity. Even so, it was added to the system in a "logical" manner, with a view to future updates that may provide features requiring extra control within the application.

Profiles present in the application:

- Regular user: has access to all the features present in the technological prototype, i.e., the specification, configuration, and download of services referenced to their user profile and the ability to consult both the documentation present on the platform and the user guide.
- Administrator user: as explained above, the administrator user does not currently have any different permissions from the common user, but is still present in the system in case of possible updates.

Information Backup Policy

In order to back up the information related to both the application code and the information produced as a result of its use and execution, two copies of the application source code and three copies of the user data are stored.

User data is initially stored on the hosting service where the database engine container volume is stored. In this case, the cloud service provider is DigitalOcean. As a second backup, the system runs a process every day at 00:00 (GMT-3) to keep a copy of the database content on the local server at the development offices. Finally, in order to maximize user data integrity, a copy of the user data is stored weekly on an external hard drive, which is stored in a confidential location in a building other than the local server and is known only to the company's management.

The application source code is handled and stored on GitHub, where developers work on it. In turn, as mentioned above, functional versions are backed up and stored in two instances upon completion. First, the code is stored manually on the company's local server and then stored on an external hard drive kept in a confidential location in a building other than the company's offices and known only to the company's management.

The local server mentioned for both source code and user data backup is a NAS located in the development offices, configured with RAID 10 to obtain a high level of redundancy and remarkable performance, ensuring outstanding information availability even in the event of incidents and providing fast recovery of persistent content.

DigitalOcean also takes availability into account and, in fact, this is one of the factors that influenced the decision to run the system on its cloud services, since the platform boasts 99,99 % uptime for the products used to deploy and run the developed project.[1]

Cost Analysis

Below is a breakdown of the estimated costs of the project in terms of the human resources required for the development of the computer system. These were obtained from the website of the Professional Council of Computer Sciences of the province of Córdoba on October 21, 2024.

Table 41. HR cost analysis

Role	Fees	Months	Subtotal (AR$)
Senior Programmer Analyst	$1 697 430,72	3	$5 092 292,16
Backend Developer	$1 985 445,37	3	$5 956 335,93
Frontend Developer	$1 883 828,08	2	$3 767 656,16
Application Testing Analyst	$1 646 481,38	2	$3 292 962,76
Total Development			$18 109 247

Having presented the figures relating to labor, we now present the operating costs considered necessary for the proper deployment and operation of the project.

Table 42. Operating cost analysis

Resource	Amount	Source	Subtotal AR$	Monthly AR$
* Kubernetes Basic Node (DigitalOcean) 8 GB RAM 4 vCPU 160GB storage	2	https://www.digitalocean.com/pricing	-	$94 464

.com domain name	1	https://www.hostinger.com.ar/domains	-	$2144
NAS Drive Linux OS RAID 10 compatible 4 HDD capacity	1	https://www.compel.com.ar/storage/storage/nas-drive-au-4b-25-35a335696.html	$816,85	-
3TB HDD NAS	4	https://www.compel.com.ar/storage/hdd-internal/hdd-3t-sea-35-nas-ironwol-328858.html	$636	-
Total Initial Cost			$1 453 511	
Total Fixed Costs				$96 608

Note: * Original value in USD, converted to AR$ considering 1 USD equivalent to 948 AR$ based on the exchange rate provided by the Central Bank of the Argentine Republic on October 21, 2024.[2]

Regarding the costs related to the software used for the project's development, the decision was made to use open source platforms, accessing free plans to save on licensing costs. Even so, these tools are presented for informational purposes.

Table 43. Analysis of development tool costs

Tool	Subtotal (AR$)
PostgreSQL	$
Apache Kafka	$
MinIO	$0
Docker	$0
Kubernetes	$0
Spring Framework	$0
Total Software Licenses	$

To conclude the cost analysis, a summary of the costs is provided, excluding the salary values detailed above.

Table 44. Summary of costs excluding HR

	Human Capital	Software and Licenses	Infrastructure and Hardware	Total
Initial Cost (AR$)*	$18 109 247,01	$	$1 453 511	19 562 758,01
Monthly Fixed Cost (AR$)**	$	$	$96 608	$96 608

Note: * Includes all costs related to the first three months of activity, i.e., until the development of the system is complete, excluding its deployment.
** Includes the costs of maintaining the system once it has been developed and deployed in the cloud.

Risk Analysis

The risks that may arise during the course of the project are described and detailed below, divided into different tables according to their cause. These tables show both the probability and impact of each risk, values that will be used to determine their significance in the matrix presented below.

Technical Risks

Table 45. Technical Risks

ID	Type	Risk	Probability	Impact	Cause
1	Technical	The platform's response time does not meet expected response standards	0,40	4	Inefficiencies in architecture or lack of optimization in communications
2	Technical	Security vulnerability in services that interact with databases.	0,7	2	Lack of proper sanitization and validation of user input

Table 46. Project risks

ID	Type	Risk	Probability	Impact	Cause
3	Project	Dependence on third-party APIs for integration or creation of services that are not available	0,80	3	Lack of control over the quality and availability of services used in the system
4	Project	Difficulty in obtaining the technical personnel necessary to carry out the development of the system	0,3	3	Very specific subject matter due to the ultimate goal of the system, which focuses on distributed systems
5	Project	Insufficient resources for project development	0	4	Lack of investment in human and/or technological resources technology
6	Project	The field of research does not provide useful information on which to base the project.	0,60	2	The field of microservices research may be incomplete due to its recent inclusion.

Once the identified project risks have been exposed, we proceed with the aforementioned risk matrix in order to weigh the probabilities of occurrence and their related impacts.

			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

Figure 21. Risk Matrix

Based on the matrix presented, both the tables shown above and the one defined below, referring to the quantitative analysis of risks, were developed.

Table 47. Quantitative risk analysis

Risk	Probability of Occurrence	Impact	Degree of Exposure	Percentage	Cumulative Percentage
Dependence on third-party APIs for integration or creation of services that are not available	0,8	3	2	25,40	25,40
Insufficient resources for project development	0,45	4	1,8	19,06	44,46
The platform's response time does not meet the expected response standards	0,40	4	1,60	16,93	61,39
Security vulnerability in services that interact with databases.	0,70	2	1,40	14,81	76,2
The field of research does not provide useful information on which to base a comparison.	0,60	2	1,2	12,70	88,90
Difficulty in obtaining the technical personnel necessary to carry out the development of the system	0,35	3	1	11,10	100

Once the degree of risk exposure for each risk detected has been presented, it is time to use the Pareto Principle to focus attention on the important and critical aspects, ignoring the more trivial ones. The corresponding graph is shown below.

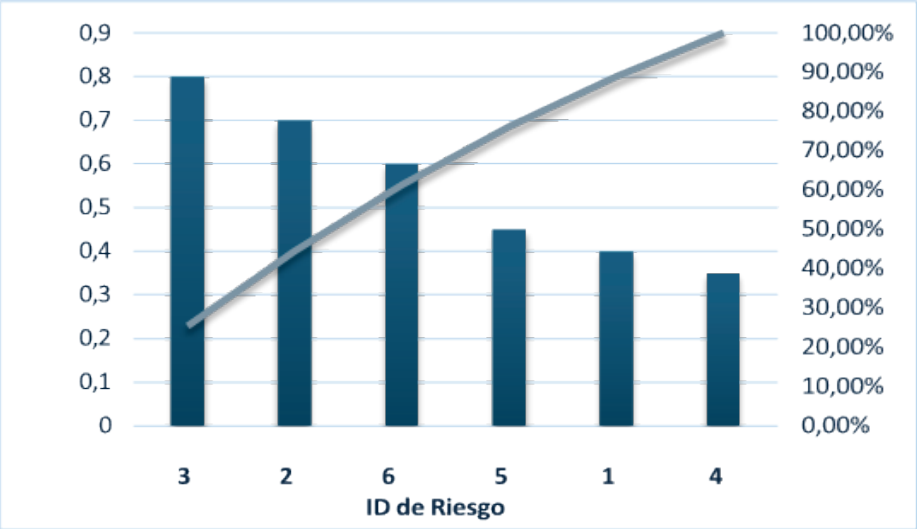

Figure 22. Pareto Principle of Risk Exposure

Once the most threatening risks have been identified using the Pareto Principle, specific contingency plans have been developed to mitigate these threats. The details of these plans are presented below.

Table 48. Contingency Plans

Risk	Contingency Plan
Dependence on third-party APIs for integration or creation of services that are unavailable	Develop minimum viable internal solutions to reduce critical dependence on external APIs. Establish contracts with key suppliers to ensure response times response times and availability, and define monitoring mechanisms to detect failures and act quickly.
Insufficient resources for project development	Establish a flexible resource allocation plan from the outset, allowing efforts to be redistributed according to project priorities. Propose additional phases or adjustments to the project scope to adapt to the available budget.
The platform's response time does not meet expected response standards	Conduct a thorough review of the architecture and apply specific optimizations in critical areas. Integrate real-time monitoring solutions to detect performance issues and adjust system capacity.
Security vulnerability in services that interact with databases.	Implement additional layers of security, such as application-level firewalls and encryption of sensitive data. Conduct regular security-focused code reviews and apply immediate patches for any vulnerabilities discovered.

CONCLUSIONS

This project demonstrated that combining microservice architectures with low-code approaches is not only possible but also highly beneficial for reducing complexity in the design and development of distributed systems. Through the implementation of a visual and intuitive platform, we were able to offer a solution capable of significantly shortening development times, facilitating component integration, and reducing common errors during manual coding.

One of the main achievements was the construction of a functional environment that allows developers to drag, configure, and relate microservices visually, and then automatically generate the corresponding code. This functionality, supported by technologies such as RDF, SPARQL, and Apache Velocity, is a significant advance in software development automation, ensuring structural consistency without sacrificing flexibility.

The choice of modern, open source tools such as Java with Spring Framework, PostgreSQL, Apache Kafka, MinIO, Docker, and Kubernetes was key to ensuring the scalability, portability, and robustness of the system. In addition, secure authentication practices were implemented with OAuth2 (via Google), along with backup and distribution mechanisms that ensure data integrity and availability.

On the methodological side, the use of Scrum as an agile framework allowed for iterative product evolution, fostering continuous improvement and rapid response to technical obstacles or changes in requirements. This dynamic was essential for adjusting details in real time, improving the functional design of the canvas, and

adapting the code generation logic according to the results obtained in each sprint.

The collection of information through the analysis of scientific literature, complemented by observations on social networks used by developers, provided a comprehensive view of the problem to be addressed. This integration of theory and practice facilitated the validation of the real needs of the target user and guided the design of key platform features.

In summary, the developed system fulfills the objective of facilitating the creation of microservice architectures, providing a powerful, accessible, and adaptable tool. Although it is a functional prototype, its structure and design anticipate future evolution with greater possibilities for customization, template expansion, and compatibility with enterprise production environments. The path towards the democratization of distributed development through low-code platforms is thus open and enhanced with this technological proposal.

BIBLIOGRAPHIC REFERENCES

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/

FINANCING

None

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

AUTHORSHIP CONTRIBUTION

Conceptualization: Tomás Darquier, Pablo Alejandro Virgolini.
Data curation: Tomás Darquier, Pablo Alejandro Virgolini.
Formal analysis: Tomás Darquier, Pablo Alejandro Virgolini.
Research: Tomás Darquier, Pablo Alejandro Virgolini.
Methodology: Tomás Darquier, Pablo Alejandro Virgolini.
Project management: Tomás Darquier, Pablo Alejandro Virgolini.
Resources: Tomás Darquier, Pablo Alejandro Virgolini.
Software: Tomás Darquier, Pablo Alejandro Virgolini.
Supervision: Tomás Darquier, Pablo Alejandro Virgolini.
Validation: Tomás Darquier, Pablo Alejandro Virgolini.
Visualization: Tomás Darquier, Pablo Alejandro Virgolini.
Writing – original draft: Tomás Darquier, Pablo Alejandro Virgolini.
Writing – review and editing: Tomás Darquier, Pablo Alejandro Virgolini.