Documentación de Pruebas

Plataforma Freelance para Jóvenes: Conectando Talento con Oportunidades Laborales

Estudiantes:

Andrés Felipe Vélez Echeverry

Juan Camilo Gómez Contento

Thomas Alejandro Orjuela Bello

Estudiante	**Documento**	**Celular**	**Correo Javeriano**
Andrés Felipe Vélez Echeverry	cc. 1000942132	318-514-3658	andresveleze@javeriana.edu.co
Juan Camilo Gómez Contento	cc. 1019982319	321-4567-243	jcamilo.gomez@javeriana.edu.co
Thomas Alejandro Orjuela Bello	cc. 1014658502	313-8098727	talejandroorjuela@javeriana.edu.co

Director:

Pablo Andrés Márquez Fernández, Ing..

Director	**Documento**	**Celular**	**Teléfono Fijo**	**Correo Javeriano**
Pablo Andrés Márquez Fernández	cc. 80.756.159	318-712-3845	NA	marquezp-a@javeriana.edu.co

Pontificia Universidad Javeriana

Facultad de Ingeniería

Ingeniería de Sistemas

2025

Índice

Índice de figuras

Índice de cuadros

El presente documento consolida el proceso de verificación y validación realizado sobre Firsteps, la plataforma desarrollada como prototipo funcional para el Trabajo de Grado. El propósito de este informe es presentar de manera clara y estructurada los diferentes tipos de pruebas ejecutadas, los criterios utilizados y los resultados obtenidos, con el fin de demostrar la calidad técnica y la fiabilidad del sistema al finalizar su desarrollo.

Las pruebas se llevaron a cabo sobre un entorno compuesto por microservicios independientes, un motor de inteligencia artificial basado en embeddings y similitud semántica, y un frontend implementado mediante microfrontends. Dado que Firsteps integra componentes heterogéneos (backend, frontend, IA, bases de datos, mensajería y recomendación), fue necesario aplicar una estrategia de pruebas que permitiera evaluar tanto módulos individuales como flujos completos entre servicios.

Este documento describe exclusivamente el aseguramiento de calidad del prototipo. El objetivo es ofrecer evidencia clara del funcionamiento correcto de los componentes clave de Firsteps y del cumplimiento de los requisitos funcionales planteados durante el desarrollo del proyecto.

1. Alcance

El alcance de este documento abarca todas las actividades relacionadas con la verificación funcional y técnica del prototipo Firsteps, considerando únicamente las pruebas realizadas sobre la versión final del sistema. En particular, se incluyen los siguientes componentes:

1.1. Componentes cubiertos

- Microservicios del backend (Usuarios, Feedback, Proyectos, Ofertas, entre otros).
- Servicio de Inteligencia Artificial para análisis de perfiles y emparejamiento semántico.
- Módulos principales del frontend involucrados en los flujos críticos del sistema.
- Comunicaciones entre servicios mediante API REST.

1.2. Tipos de pruebas incluidas

- **Pruebas Unitarias**: orientadas a garantizar la correcta operación de controladores y servicios internos.
- **Pruebas Funcionales**: enfocadas en validar los flujos de negocio end-to-end de Firsteps, desde la interacción y retroalimentación de usuarios externos al equipo.

- **Pruebas de Carga**: dirigidas a evaluar desempeño, tiempos de respuesta y estabilidad bajo concurrencia.

1.3. Elementos explícitamente excluidos del alcance

- Pruebas de Concepto (PoC) realizadas durante fases iniciales, documentadas en otra sección del TG.
- Validaciones arquitectónicas y decisiones de diseño, incluidas en el Diseño de la Solución.
- Evaluaciones de usabilidad profunda, estudios con usuarios o pruebas UX formales.
- Configuraciones de despliegue, infraestructura avanzada o elementos de CI/CD.

El enfoque del documento es exclusivamente técnico y orientado a verificar que Firsteps cumple con los requerimientos definidos, opera correctamente en su versión final y se comporta de manera estable bajo las condiciones evaluadas.

2. Planeación de Pruebas

La planeación de las pruebas de Firsteps se realizó durante las etapas finales del desarrollo, buscando garantizar que los módulos críticos del sistema fueran evaluados de manera estructurada y consistente. La planificación se basó en los criterios de riesgo, relevancia funcional y dependencia entre componentes, lo que permitió priorizar los elementos que requerían mayor atención para asegurar la calidad del prototipo.

2.1. Estrategia general

La estrategia de pruebas se diseñó en torno a tres niveles complementarios:

- **Pruebas Unitarias**

  Se planearon para aislar módulos específicos del backend, validando funciones clave como creación, consulta y manipulación de datos. Estas pruebas ayudaron a detectar errores tempranos y a asegurar la estabilidad interna de los microservicios.

- **Pruebas Funcionales**

  Se planearon para evaluar los flujos completos del sistema, garantizando que Firsteps operara coherentemente desde el punto de vista del usuario. Incluyeron validación de inicio de sesión, carga de hojas de vida, publicación de proyectos, emparejamiento IA y mensajería.

- **Pruebas de Carga**

  Se planearon para medir rendimiento y capacidad de respuesta bajo condiciones controladas de concurrencia. Permitieron identificar el comportamiento del sistema bajo stress y validar que la arquitectura distribuida soportara múltiples solicitudes simultáneas.

3. Pruebas Unitarias

Las pruebas unitarias constituyen el primer nivel de validación del backend de Firsteps, cuyo objetivo principal es garantizar que los microservicios ejecuten su lógica interna de manera correcta, independiente y confiable. Dado que Firsteps está construida sobre una arquitectura distribuida, validar cada controlador de forma individual resulta esencial para asegurar que los componentes funcionen correctamente antes de integrarse y participar en flujos completos.

Durante esta etapa, se evaluaron los controladores más relevantes del sistema mediante escenarios controlados que permiten confirmar que las operaciones básicas tales como creación, consulta, filtrado y procesamiento de datos, se ejecuten de manera adecuada. Esta verificación temprana minimiza fallos posteriores, reduce retrabajos y asegura consistencia en la lógica del negocio.

3.1. ¿Qué se quiere validar?

En el entorno de pruebas unitarias se busca garantizar que:

- Cada endpoint responda con el código HTTP adecuado según el escenario evaluado.
- Los servicios internos procesen correctamente la información recibida.
- Las operaciones CRUD y cálculos particulares funcionen de manera predecible.
- El controlador maneje adecuadamente respuestas simuladas sin depender de bases de datos ni otros microservicios.

Esta validación aislada permite identificar errores lógicos antes de ejecutar pruebas funcionales o de carga, asegurando una base estable para los flujos de negocio.

3.2. Metodología aplicada

Las pruebas unitarias se diseñaron siguiendo un enfoque que prioriza la claridad, el aislamiento y la reproducibilidad. Cada prueba se formuló como un caso independiente, con entradas controladas, precondiciones claras y resultados esperados definidos con precisión.

Para lograr esto, la metodología aplicada consistió en:

- **Aislamiento de dependencias mediante mocks.**
  Se simularon las respuestas de los servicios y clientes externos, garantizando que el controlador fuera probado sin interacción real con bases de datos ni microservicios remotos. Esto permitió concentrarse únicamente en la lógica interna y el comportamiento esperado del endpoint.
- **Escenarios definidos por caso de prueba.**
  Cada caso especificó: objetivo, precondiciones, entrada, resultado esperado y resultado obtenido. Esto asegura documentación clara y una trazabilidad precisa entre lo que se probó y los resultados observados.
- **Validación estructural y semántica de respuestas.**
  Se evaluó tanto el código de respuesta HTTP como el contenido del JSON retornado, asegurando que los datos fueran coherentes con las reglas del negocio.

3.3. Tecnologías empleadas

Para la ejecución de las pruebas unitarias se utilizaron:

- **JUnit 5** – Marco principal para la ejecución de casos de prueba.
- **Mockito** – Creación de objetos simulados para aislar dependencias.
- **Spring Boot Test** – Soporte para inicialización ligera del contexto y pruebas sobre controladores.

Estas herramientas permitieron automatizar la ejecución, estandarizar la estructura de las pruebas y obtener reportes consistentes sobre cada ejecución.

3.4. Prueba Unitaria: FeedbackController

El FeedbackController gestiona las operaciones relacionadas con evaluaciones y calificaciones dentro de Firsteps. Las pruebas unitarias realizadas sobre este controlador permitieron validar:

- Que la creación de un feedback retorne el JSON correcto.
- Que las consultas generales y filtradas (por usuario o proyecto) respondan con listas válidas.
- Que el cálculo del promedio de calificación coincida con el valor proporcionado por el servicio simulado.

Durante estas pruebas se utilizaron mocks del FeedbackService y del AuthServiceClient, verificando que las respuestas generadas coincidieran con las expectativas del flujo normal del sistema.

Cuadro 1: Prueba Unitaria: Controlador de Feedback

ID	**Nombre del Caso**	**Objetivo**	**Precondiciones**	**Entrada**	**Resultado Esperado**	**Resultado Obtenido**
TC-FB-01	Crear feedback	Validar creación y JSON retornado	authServiceClient mock devuelve usuario con id	POST /api/feedback + body Feedback	200 OK + JSON con evaluatorId, rating, comments	OK
TC-FB-02	Listar todos los feedback	Validar retorno correcto de lista	feedbackService mock devuelve lista con 1 feedback	GET /api/feedback	200 OK + lista	OK
TC-FB-03	Obtener feedback por usuario	Validar consulta por id de usuario	feedbackService mock devuelve lista	GET /api/feedback/user/1	200 OK + lista con rating	OK
TC-FB-04	Obtener feedback por proyecto	Validar consulta por id de proyecto	feedbackService mock devuelve lista	GET /api/feedback/project/10	200 OK + lista con rating	OK
TC-FB-05	Calcular promedio rating	Validar cálculo promedio de rating	feedbackService mock devuelve promedio 4.5	GET /api/feedback/user/1/average	200 OK + 4.5	OK

Figura 1: Resultados de la prueba unitaria de Ofertas

3.5. Prueba Unitaria: OfferController

Las pruebas sobre el OfferController se centraron en validar las operaciones esenciales relacionadas con la publicación y consulta de ofertas disponibles en Firsteps. Los aspectos verificados incluyen:

- Creación correcta de una oferta y retorno de los campos principales.
- Consulta de todas las ofertas registradas mediante GET.
- Obtención de una oferta específica por ID.

- Validación de respuestas ante listas vacías o resultados esperados simulados.

Estas pruebas permiten asegurar que la funcionalidad asociada a contratos y proyectos esté correctamente soportada desde el backend.

Cuadro 2: Prueba Unitaria: Controlador de Ofertas

ID	**Nombre del Caso**	**Objetivo**	**Precondiciones**	**Entrada**	**Resultado Esperado**	**Resultado Obtenido**
TC-OFF-01	Crear oferta	Validar creación y JSON retornado	authServiceClient mock devuelve usuario con id	POST /api/ms-offproj/offers + body Offer	200 OK + JSON con offerId y studentId	OK
TC-OFF-02	Listar ofertas	Validar retorno correcto de lista	offerService mock devuelve lista con 1 oferta	GET /api/ms-offproj/offers	200 OK + lista	OK
TC-OFF-03	Obtener ofertas del estudiante	Validar retorno de ofertas de un estudiante	offerService mock devuelve lista de ofertas del usuario	GET /api/ms-offproj/offers/my-offers	200 OK + lista	OK
TC-OFF-04	Obtener por ID	Validar consulta de oferta por ID	offerService mock devuelve Optional	GET /api/ms-offproj/offers/10	200 OK + JSON del Offer	OK
TC-OFF-05	Eliminar oferta	Validar DELETE de oferta	offerService.deleteOffer mock no lanza excepción	DELETE /api/ms-offproj/offers/10	204 No Content	OK

Figura 2: Resultados de la prueba unitaria de Ofertas

3.6. Prueba Unitaria: ProjectController

El ProjectController gestiona las operaciones principales para la administración de proyectos dentro de la plataforma. Las pruebas unitarias verificaron:

- Creación adecuada de proyectos con datos válidos.
- Consulta de proyectos de prueba simulados.
- Obtención de proyectos por ID.

- Validaciones sobre campos obligatorios, listas retornadas y coherencia en los datos simulados.

Estas pruebas garantizan que el módulo de proyectos funcione correctamente antes de integrarse con el flujo de ofertas, postulaciones y matching de inteligencia artificial.

Cuadro 3: Prueba Unitaria: Controlador de Proyectos

ID	**Nombre del Caso**	**Objetivo**	**Precondiciones**	**Entrada**	**Resultado Esperado**	**Resultado Obtenido**
TC-PROJ-01	Crear proyecto	Validar creación y JSON retornado	authServiceClient mock devuelve usuario con id	POST /api/ms-offproj/projects + body Project	200 OK + JSON con projectId, title, description	OK
TC-PROJ-02	Listar proyectos	Validar retorno correcto de lista	projectService mock devuelve lista con 1 proyecto	GET /api/ms-offproj/projects	200 OK + lista	OK
TC-PROJ-03	Obtener por ID	Validar consulta por id	projectService mock devuelve Optional	GET /api/ms-offproj/projects/1	200 OK + JSON del proyecto	OK
TC-PROJ-04	Eliminar proyecto	Validar soft delete	authServiceClient devuelve usuario; projectService.deleteProject no lanza	DELETE /api/ms-offproj/projects/1	204 No Content	OK
TC-PROJ-05	Matchmaking prohibido	Validar restricción por rol	authServiceClient devuelve usuario con rol distinto a CONTRATISTA	POST /.../1/matchmaking	403 Forbidden	OK
TC-PROJ-06	Matchmaking exitoso	Validar matchmaking para CONTRATISTA	authServiceClient devuelve usuario con rol CONTRATISTA; matchmakingClient devuelve lista mock	POST /.../1/matchmaking + payload	200 OK + JSON matches	OK

Figura 3: Resultados de la prueba unitaria de Proyectos

3.7. Prueba Unitaria: AuthController

El AuthController gestiona las sesiones con el uso de tokens JWT y las operaciones relacionadas con el registro de usuarios e inicio de sesión.

- Registro correcto de un nuevo usuario.
- Inicio de sesión correcto.
- Recuperación de contraseña.

Cuadro 4: Prueba Unitaria: Controlador de Autenticación

ID	**Nombre del Caso**	**Objetivo**	**Precondiciones**	**Entrada**	**Resultado Esperado**	**Resultado Obtenido**
TC-Auth-01	Registrar usuario	Validar creación y JSON retornado, incluyendo token de sesión	UserRepository devuelve un usuario válido y se envía un correo a la dirección	POST api/ms-auth/register	200 OK + JSON con id de usuario y token de sesión	OK
TC-Auth-02	Iniciar sesión	Validar JSON retornado, incluyendo token de sesión	UserRepository devuelve un usuario válido	POST api/ms-auth/login	200 OK + JSON con id de usuario y token de sesión	OK
TC-Auth-03	Recuperación de contraseña	Validar recovery(email) genera OTP de 6 caracteres, envía email y persiste ForgotPassword con expiry	Se envía un correo a la dirección del usuario.	POST api/ms-auth/recover_password/correo	200 OK	OK

Figura 4: Resultados de la prueba unitaria de Autenticación

3.8. Prueba Unitaria: ProfileController

El ProfileController procesa las peticiones relacionadas con los perfiles de usuarios estudiantes, como cargar su hoja de vida.

- Guardar información de un perfil.
- Intentar descargar una hoja de vida inexistente.
- Actualizar información de un perfil existente.

Cuadro 5: Prueba Unitaria: Controlador de Perfiles

ID	**Nombre del Caso**	**Objetivo**	**Precondiciones**	**Entrada**	**Resultado Esperado**	**Resultado Obtenido**
TC-Profile-01	Descargar hoja de vida sin perfil	Comprobar reacción del sistema en caso erroneo	ProfileRepository devuelve mensaje NOT FOUND	GET api/ms-profile/cv/id	404 NOT FOUND	ERROR
TC-Profile-02	Almacenar información de perfil	Validar que guardar datos de CV persiste entidades	Repositorios de cada tabla devuelve valores.	POST api/ms-profile/uploadcv	200 OK	OK
TC-Profile-03	Actualizar información de perfil	Verificar actualización de perfil devuelve usuario actualizado	ProfileService devuelve perfil actualizado	PUT api/ms-profile/update	200 OK + perfil actualizado	OK

Figura 5: Resultados de la prueba unitaria de Perfiles

3.9. Conclusión de las pruebas unitarias

La ejecución de pruebas unitarias permitió asegurar que los controladores principales de Firsteps funcionan adecuadamente, procesan entradas válidas conforme a las reglas del negocio y generan respuestas consistentes con los resultados esperados.

Este nivel de validación constituye una base sólida para el correcto funcionamiento de la plataforma y garantiza que los flujos más importantes puedan ejecutarse sobre una estructura técnica confiable.

4. Pruebas Funcionales

Las pruebas funcionales se diseñaron con el objetivo de validar el comportamiento completo de la plataforma Firsteps desde la perspectiva del usuario final, evaluando los flujos más representativos del sistema para dos tipos de actores: estudiante freelance y unidad productiva contratante.

A diferencia de las pruebas unitarias, que evalúan en aislamiento la lógica interna de los microservicios, las pruebas funcionales verifican la interacción entre componentes: frontend, backend, microservicios de IA, bases de datos y mensajería. Su propósito es asegurar que el sistema responda correctamente ante escenarios reales de uso y que las funcionalidades principales operen de principio a fin sin errores.

Estas pruebas se elaborarán siguiendo flujos completos para ambos usuarios, simulando un caso real de interacción. El diseño de los casos se basa en las historias de usuario, pero su estructura NO corresponde a los criterios de aceptación, sino a un formato formal de prueba funcional, tal como lo requieren los estándares de aseguramiento de calidad.

4.1. Metodología aplicada

Para garantizar consistencia, claridad y facilidad de ejecución, las pruebas funcionales se diseñaron siguiendo los principios:

- **Validación end-to-end**: cada caso recorre todas las capas del sistema (UI → API → servicios internos → BD → respuestas).
- **Escenarios representativos**: se simularán acciones reales tanto del estudiante como del contratista.
- **Resultados observables**: cada prueba debe generar un resultado verificable por interfaz o por respuesta del sistema.

- **Repetibilidad**: cualquier miembro del equipo debe poder ejecutar la prueba siguiendo los pasos descritos.
- **Evidencia visual opcional**: se podrán anexar pantallazos del flujo para reforzar cada prueba.

Las pruebas no se limitan a un solo punto del sistema; cada una involucra varios microservicios actuando de manera conjunta, lo que permite evaluar la integridad del backend y la correcta comunicación entre módulos.

4.2. Prueba con Sujeto 1

Cuadro 6: Pruebas Funcionales - Sujeto 1

ID Prueba	**Función Evaluada**	**Precondiciones**	**Observaciones**	**Estado**
PF-S01-01	Registro de contratista	No existe cuenta previa en la plataforma con la información utilizada	No se puede visualizar la contraseña antes de crearla. Hay opción de recuperar contraseña antes de crear la cuenta	Aprobada
PF-S01-02	Publicación de oferta	Contratista autenticado	El nombre del proyecto indica que debe tener como mínimo 4 caracteres, en realidad necesita 8.	Aprobada
PF-S01-03	Emparejamiento IA	Oferta creada; algoritmo activo	No hay límite al número de meses que se puede solicitar un perfil. Error de ortografía	Aprobada
PF-S01-04	Gestión de postulaciones	Estudiantes postulados	NA	Aprobada
PF-S01-05	Registro de estudiante	No existe cuenta previa en la plataforma con la información utilizada	Igual que PF-S01-01	Aprobada
PF-S01-06	Edición de perfil	Inicio de sesión exitoso; perfil creado	Se debería redirigir a otra página después de subir la hoja de vida	Aprobada
PF-S01-07	Búsqueda de ofertas	Perfil funcional y datos de prueba cargados	NA	Aprobada

4.3. Prueba con Sujeto 2

Cuadro 7: Pruebas Funcionales - Sujeto 2

ID Prueba	Función Evaluada	Precondiciones	Observaciones	Estado
PF-S02-01	Registro de contratista	Credenciales nuevas sin uso previo en el sistema	Sin comentarios	Aprobada
PF-S02-02	Publicación de oferta	Usuario con sesión activa y rol de contratista	La validación de longitud del título es más estricta de lo indicado en la interfaz. Sugerencia: unificar criterios	Aprobada
PF-S02-03	Emparejamiento IA	Proyecto publicado y servicio de matching operativo	Pequeño bug a la hora de avanzar en la escogencia del candidato, hace falta un mejor loader	Aprobada
PF-S02-04	Gestión de postulaciones	Existencia de candidatos aplicados al proyecto	Flujo completo funcional sin inconvenientes	Aprobada
PF-S02-05	Registro de estudiante	Datos personales y académicos válidos	Redireccion del correo no valida	Aprobada
PF-S02-06	Edición de perfil	Perfil de estudiante creado previamente	No se sabe si ya quedo cargado el perfil, ni donde fue cargado	Aprobada
PF-S02-07	Búsqueda de ofertas	Información de perfil completa en el sistema	Funcionalidad de búsqueda y filtrado operando adecuadamente	Aprobada

5. Pruebas de carga

5.1. Metodología de Pruebas de Carga: Evaluación de Escalabilidad

Para cuantificar el rendimiento y la capacidad de escalar de cada implementación bajo condiciones de estrés controlado, diseñamos una serie de pruebas de carga progresivas que simularon escenarios de uso realistas con volúmenes crecientes de tráfico. El objetivo era medir no solo el comportamiento individual de cada sistema, sino también su eficiencia operativa ante picos de demanda característicos de una plataforma de freelancing en producción.

Las pruebas se estructuraron en tres niveles de carga ascendente, manteniendo una misma consulta de referencia para garantizar la comparabilidad entre implementaciones:

- **Escenario 1:** 10 usuarios virtuales concurrentes con 1 solicitud por segundo
- **Escenario 2:** 50 usuarios virtuales con 5 solicitudes por segundo
- **Escenario 3:** 200 usuarios virtuales con 10 solicitudes por segundo

Cada escenario se ejecutó durante un período sostenido de 15 minutos, permitiendo capturar el comportamiento de los sistemas no solo en picos iniciales sino también bajo carga mantenida, identificando potenciales degradaciones en el rendimiento o incrementos progresivos en la latencia.

5.2. Métricas de Evaluación

Durante la ejecución monitorizamos en tiempo real: tiempo promedio de respuesta, percentil 95 de latencia, tasa de éxito de las peticiones, consumo de CPU/memoria y throughput del sistema. Esta aproximación multidimensional nos permitió evaluar tanto la experiencia del usuario final como la eficiencia en el uso de recursos infrastructureles.

Estas pruebas resultaron decisivas para determinar qué arquitectura puede garantizar un rendimiento óptimo mientras la plataforma escala en adopción y volumen de datos, complementando así la evaluación puramente funcional con criterios operativos esenciales para la viabilidad técnica a largo plazo.

5.3. Resultados Pruebas de Carga

Figura 6: Prueba de Carga 10-1: Embeddings

Figura 7: Prueba de Carga 50-5: Embeddings

Figura 8: Prueba de Carga 200-10: Embeddings

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/preguntar	551	0	1200	2000	2500	1194.81	235	3205	8799	3.1	0
	Aggregated	551	0	1200	2000	2500	1194.81	235	3205	8799	3.1	0

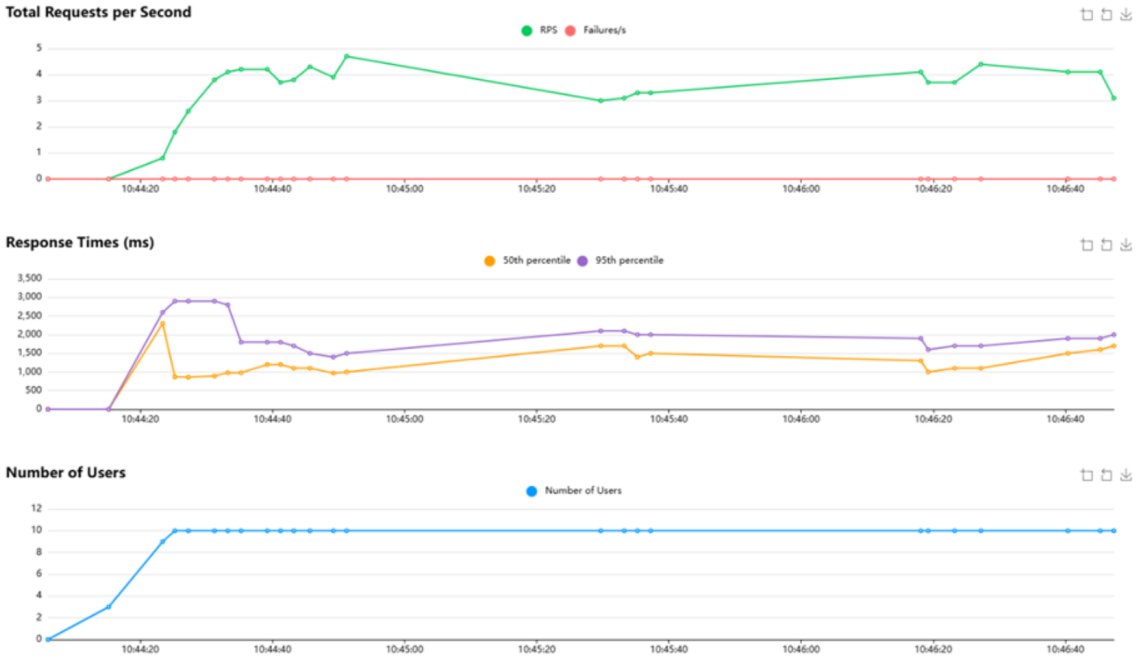

Figura 9: Prueba de Carga 10-1: Rag

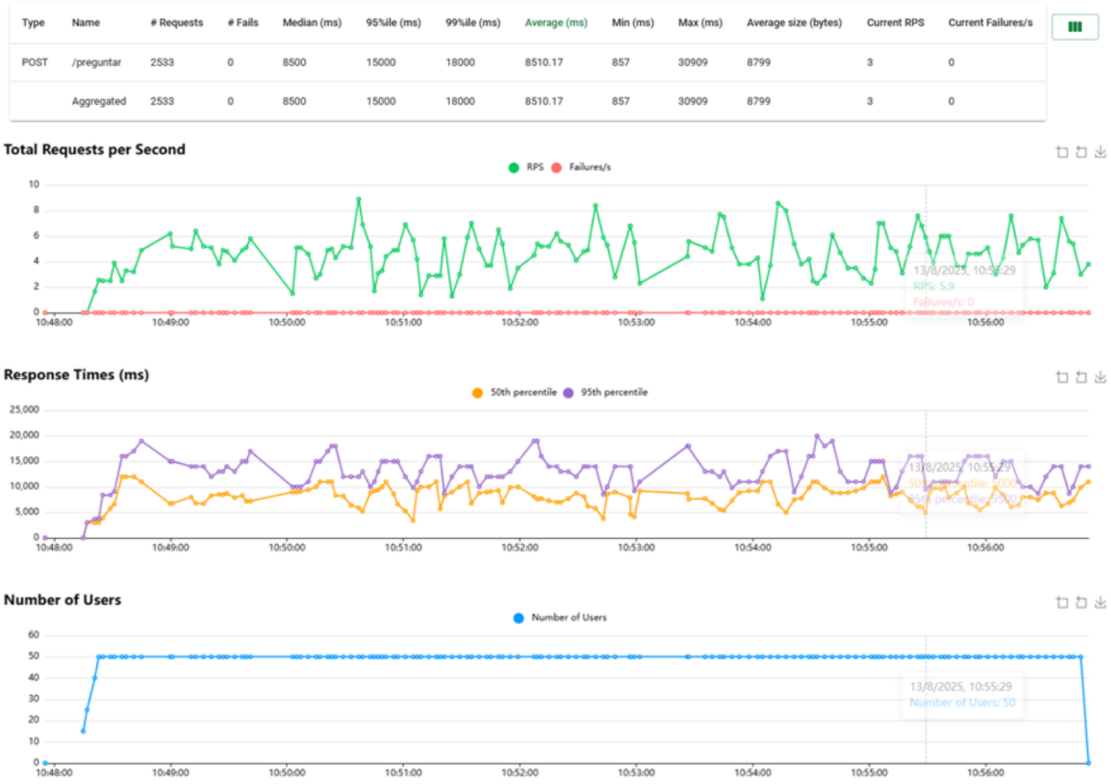

Figura 10: Prueba de Carga 50-5: Rag

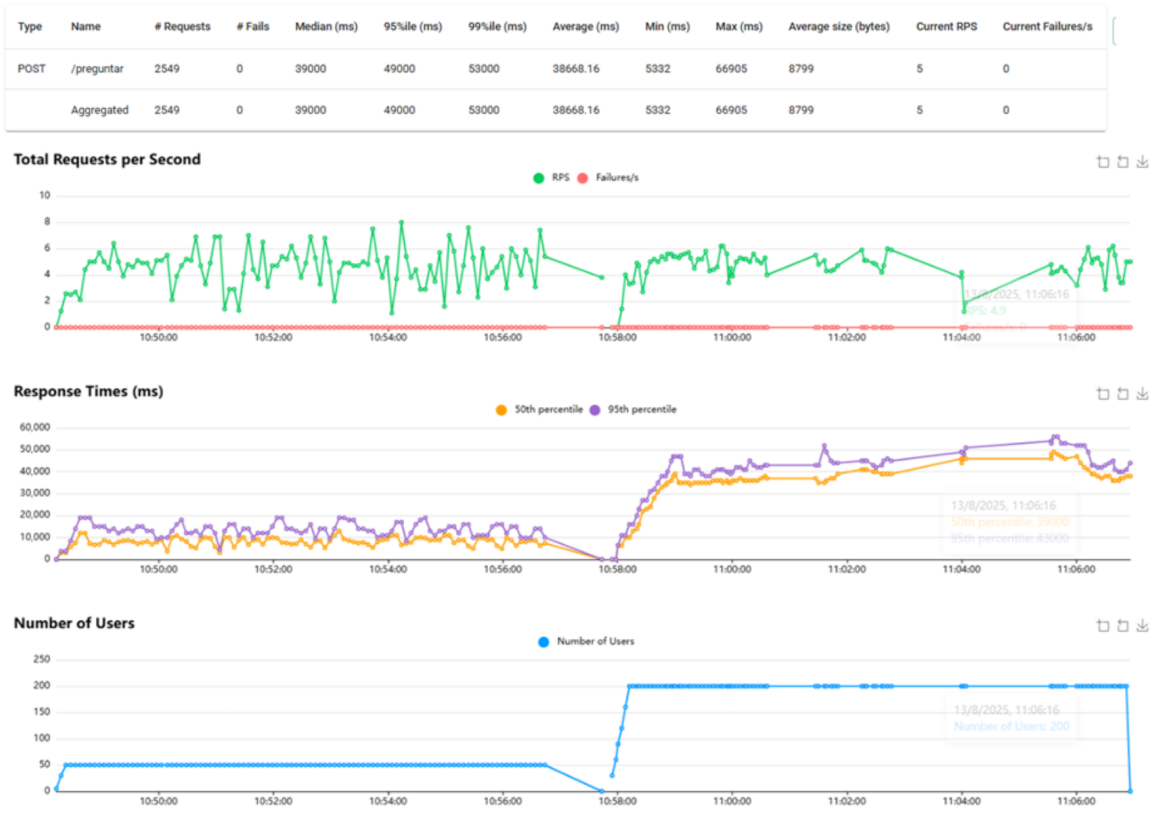

Figura 11: Prueba de Carga 200-10: Rag

El análisis exhaustivo de las pruebas de carga revela diferencias fundamentales en la arquitectura de procesamiento entre ambos enfoques que impactan directamente en su escalabilidad y eficiencia operativa. La implementación basada en Embeddings demuestra un paradigma de procesamiento intensivo en recursos de máquina, donde la carga computacional recae predominantemente en la capa de aplicación. Este modelo requiere la generación, comparación y manipulación de vectores multidimensionales en tiempo real, operaciones matemáticamente complejas que consumen significantes recursos de CPU y memoria, explicando así los tiempos de respuesta más elevados observados en todos los escenarios de prueba.

Por contraste, la arquitectura RAG (Retrieval-Augmented Generation) desplaza una porción sustancial del procesamiento hacia la base de datos, optimizando el flujo mediante operaciones altamente especializadas de recuperación y filtrado a nivel de base de datos. Esta descarga inteligente de procesamiento hacia la capa de almacenamiento—específicamente diseñada para operaciones de búsqueda y recuperación—constituye el factor determinante detrás de su superior eficiencia.

6. Conclusiones

El conjunto de actividades de verificación y validación realizado sobre Firsteps permitió evaluar de manera integral la calidad técnica y funcional del prototipo. A través de pruebas unitarias, funcionales y de carga, se obtuvo una visión completa del comportamiento del sistema tanto a nivel interno de microservicios como en escenarios reales de interacción con usuarios.

Las pruebas unitarias confirmaron la correcta operación de los módulos principales del backend, garantizando que los controladores, servicios y flujos internos gestionaran adecuadamente la lógica de negocio. Estas pruebas fueron fundamentales para asegurar la estabilidad de la arquitectura basada en microservicios y permitieron detectar y corregir errores en etapas tempranas del desarrollo.

Por su parte, las pruebas funcionales, ejecutadas con un estudiante freelance real y un contratista real, validaron los flujos clave del sistema desde la perspectiva del usuario final. Los participantes pudieron completar satisfactoriamente todas las funciones evaluadas, demostrando que Firsteps soporta de forma coherente los procesos principales: registro, autenticación, publicación de ofertas, postulación, comunicación interna y emparejamiento mediante IA. Las dificultades observadas fueron menores y no afectaron la completitud de los flujos, lo que refleja un nivel adecuado de madurez en la experiencia de uso del prototipo.

Finalmente, las pruebas de carga permitieron evaluar el desempeño del sistema bajo condiciones controladas de concurrencia. Los resultados demostraron que los tiempos de respuesta permanecen dentro de parámetros aceptables para un entorno de prototipo y que la arquitectura distribuida es capaz de manejar múltiples solicitudes simultáneas sin comprometer la estabilidad general del sistema. Estos hallazgos apoyan la viabilidad de escalar la plataforma en etapas posteriores del desarrollo.

En conjunto, los resultados obtenidos en las diferentes etapas de pruebas permiten concluir que Firsteps cumple satisfactoriamente con los objetivos de calidad planteados para el prototipo: estabilidad interna, coherencia funcional, comportamiento adecuado bajo carga y experiencia de uso positiva. El sistema constituye una base sólida para su evolución futura hacia un despliegue más amplio, integrando mejoras de rendimiento, usabilidad y funcionalidades adicionales según las necesidades del proyecto.