Sistema de Caché de Consultas a Base de Datos para la Plataforma “Comunidad Académica” (COMA)

Anderson Yeseth Acuña Vargas

Trabajo de Grado para Optar al Título de Ingeniero de Sistemas

Director

Luis Ignacio González Ramírez

Magíster en Informática

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería de Sistemas e Informática

Ingeniería de Sistemas

Bucaramanga

2026

Dedicatoria

Al temple de mis padres, Blanca Vargas y Rodrigo Acuña, quiénes me han ayudado de todas las formas que han podido para completar mi formación académica. Ellos me han guiado por el camino irreprochable del civismo y la forma disciplinada de pensar por una sociedad mejor.

A la inspiración genuina del verso que esa persona transgredió en mí. Coherencia y ánimo detrás de cada idea que proferí en este trabajo son producto de esa inspiración. Inherentemente cada palabra ha tenido su autoría, aunque no haya esgrimido su nombre, sin ella este trabajo no hubiera sido posible.

Agradecimientos

Sinceramente agradecido con toda la comunidad universitaria y con la UIS misma, quiénes hicieron que el paso del tiempo y el fortalecimiento académico en mi disciplina fuera lo más ameno posible.

Agradezco a todos los profesores que conocí durante el avance de mis estudios, pero también a aquellos a los cuáles no conocí directamente en un curso pero sí intercedieron conmigo en una charla o consejo.

Agradezco al grupo CALUMET por haberme enseñado cosas más allá de las que había aprendido en el programa y por haber sido mi lugar de encuentro con compañeros con tanto potencial y calidad. También a mi director, líder del grupo, por haberme permitido desarrollar esta idea como proyecto.

Tabla de Contenidos

Lista de Tablas

Lista de Figuras

Lista de Apéndices

Resúmen

Título: Sistema de caché de consultas a base de datos para la plataforma COMA[1].

Autor: Anderson Yeseth Acuña Vargas[2].

Palabras Clave: Caché, Memoria Principal, Sistema Caché, Latencia, Base de Datos.

Descripción:

La plataforma COMA es un proyecto universitario enfocado en brindar servicios académicos e institucionales. Es un proyecto de gran trayectoria con un mantenimiento activo desarrollado por el grupo CALUMET. Actualmente una de las problemáticas que tiene es que algunos módulos tienen rendimiento no deseable debido a la naturaleza de las consultas a base de datos que realizan.

En este trabajo se exploran y analizan las estrategias para almacenamiento en caché. Se define una propuesta acorde a las limitaciones técnicas y humanas. Se implementa la propuesta usando un conjunto de tecnologías también analizado y se valida la efectividad de la propuesta a través de pruebas de rendimiento. Usando los datos se realizan un conjunto de gráficos estadísticos comunes, cómo gráfico de dispersión, de caja, de densidad de probabilidad, entre otros.

Se analizan los resultados estadísticos para cuantificar la diferencia entre el sistema normal y el sistema propuesto. Se describen los hallazgos producto del análisis y se explica cómo el grupo CALUMET puede aprovechar los resultados de este trabajo. También se hacen observaciones sobre otros hallazgos que pueden ser utilizados en trabajos futuros y cómo puede mejorarse el sistema propuesto con miras de agregar más funcionalidades, proveer más estabilidad, facilitar la integración en nuevos módulos y mejorar aún más el rendimiento.

1Trabajo de grado.
2Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Ingeniería de Sistemas. Director Luis Ignacio González Ramírez.

Abstract

Title: Cache system of queries to database for the COMA platform[3].

Author: Anderson Yeseth Acuña Vargas[4].

Key Words: Cache, Main Memory, Cache System, Latency, Database.

Description:

The COMA platform is a university project focused on providing academic and institutional services. It is a long-standing project with active maintenance developed by the CALUMET group. Currently, one of its problems is that some modules have undesirable performance due to the nature of the database queries they perform.

This paper explores and analyzes caching strategies. A proposal is defined that takes into account technical and human limitations. The proposal is implemented using a set of technologies that were also analyzed, and its effectiveness is validated through performance tests. Using the data, a set of common statistical graphs are generated, such as scatter plots, box plots, and probability density plots, among others.

The statistical results are analyzed to quantify the difference between the standard system and the proposed system. The findings from the analysis are described, and it is explained how the CALUMET group can leverage the results of this work. Observations are also made on other findings that can be used in future work and how the proposed system can be improved with a view to adding more functionalities, providing more stability, facilitating integration into new modules and further improving performance.

3Degree work.
4Faculty of Physical-Mechanical Engineering. School of Systems Engineering and Informatics. Systems Engineering. Advisor Luis Ignacio González Ramírez.

Introducción

En el campo de la programación web uno de los factores más importantes es la optimización de los procesos que realizan los servidores para que la respuesta del sistema sea lo más rápida y eficiente posible. En el contexto de la Universidad Industrial de Santander (UIS) existe una plataforma web llamada "Comunidad Académica" (COMA). Esta plataforma brinda varios servicios que permiten agilizar diferentes procesos académicos y administrativos. Uno de los principales problemas que ha tenido la plataforma es el deterioro de rendimiento en algunas de las funcionalidades debido a limitaciones en el hardware o en la arquitectura del software en sí mismo. El grupo de innovación y desarrollo CALUMET, encargado del software de la plataforma, ha avanzando en esfuerzos para modernizar, actualizar, corregir deficiencias y obsolescencias en la arquitectura del software; sin embargo, un desafío transversal en todos los módulos es el rendimiento de algunas consultas de bases de datos que hacen que funcionalidades del sistema tengan bajo rendimiento.

El grupo CALUMET ha planteado varias formas para abordar este problema. Unas formas más conservadoras, como por ejemplo: optimizar las consultas para que se limiten sólo a las columnas y registros que se necesitan, reformular consultas que unían varias tablas al mismo tiempo, entre otras; y otras más reformadoras, como por ejemplo: cambiar relaciones de tablas, hacer tablas intermedias, crear índices, cambiar la forma como se hacen las consultas, entre otras. Cada una de las formas tiene sus beneficios y limitaciones para tenerlas como soluciones generales, por eso es un trabajo laborioso determinar qué solución es la más adecuada para el caso puntual. Recientemente, se planteó la posibilidad de que se usara un sistema que permitiera administrar y recuperar los datos de esas consultas en un motor de base de datos externo,

enfocado en la velocidad de respuesta. En este punto es donde se concreta la idea de crear un sistema que proporcione funciones de caché para la plataforma COMA.

Se usó una combinación de tecnologías de fácil adopción para el grupo, para preservar la mantenibilidad en el futuro. Acompañado el progreso con una evaluación de indicadores estadísticos de las métricas que se iban tomando para cuantificar el efecto de la implementación del sistema.

La finalidad de este trabajo consiste en crear un sistema que permita administrar la caché, para: mantener la información relevante para las funcionalidades más lentas y usadas, y crear una solución más flexible para aplicar en diferentes módulos del sistema cuando se requiera una mejora de rendimiento.

1. Planteamiento del Problema

La plataforma “Comunidad Académica” (COMA), del grupo CALUMET, ofrece un espacio de encuentro e interacción para apoyar la misión de los diferentes actores relacionados con un programa de formación académica de una universidad. Ellos son: Docentes, directivos, administrativos, empleadores, estudiantes y egresados. Actualmente, ofrece sus servicios en todas las escuelas de la Universidad Industrial de Santander (UIS) y, de ahí la necesidad de prestar un servicio eficiente.

En el funcionamiento habitual de la plataforma, los diferentes actores realizan operaciones rutinarias que demandan trabajo de recolección y procesamiento de datos como, por ejemplo, cuando se realiza el inicio de sesión. Estos datos, que recurrentemente se recuperan de la base de datos, suponen siempre un tiempo de espera para los usuarios que, sobre todo en franjas de alto tráfico, congestionan el servidor de la plataforma, degradando la experiencia de usuario.

Debido a la cantidad de usuarios y a la presencia de la plataforma COMA en todas las escuelas cualquier mejora en el rendimiento conlleva a la prestación de un mejor servicio disminuyendo las latencias y evitando retrasos incómodos al usuario.

2. Justificación del Proyecto

La implementación de un sistema que permita almacenar los datos más solicitados en memoria principal para las funcionalidades objetivo se plantea como un medio para mejorar los tiempos de respuesta a los usuarios. Siendo las funcionalidades objetivo aquellas donde se

encuentren las consultas a bases de datos más repetitivas y lentas.

Al final de la integración del sistema propuesto con la plataforma, se busca mejorar los tiempos de respuesta para esas funcionalidades determinadas en el análisis de COMA.

El sistema se acoplará con la arquitectura actual actuando como un servicio. Los diferentes módulos de COMA (que sean elegidos) podrán consultar los datos necesarios para sus operaciones, con una latencia menor que si tuvieran que ser leídos del disco.

En el futuro de la plataforma, el desarrollo de este sistema posibilitará la introducción de nuevos módulos con tiempos de respuesta optimizados.

3. Objetivos

3.1. Objetivo General

Reducir los tiempos de respuesta a los usuarios de la plataforma COMA mediante la creación de un sistema de caché de consultas a base de datos que disminuya la latencia en el acceso a la información.

3.2. Objetivos Específicos

- Realizar un análisis estadístico sobre los tiempos de respuesta de las funcionalidades que son abordadas .
- Determinar la información relevante para mantener en caché, así como: su estructura, permisos de lectura, tiempos de vida y respuesta a situaciones excepcionales.
- Crear sistema que use tecnologías modernas adecuadas que permita el uso eficiente de los datos que van a estar en caché.
- Evaluar el impacto del sistema de caché de consultas a base de datos mediante pruebas de rendimiento comparativas.

4. Marco Teórico

4.1. Almacenamiento en Caché

El almacenamiento en caché es una técnica de optimización de rendimiento que almacena datos frecuentemente accedidos en una capa de almacenamiento de alta velocidad para reducir la latencia (Tessier, 2025). No es una técnica específica de ciertos campos tecnológicos, puede existir en muchos niveles de uno o varios sistemas.

Según Tessier (2025) algunos beneficios del almacenamiento en caché son:

- **Rendimiento mejorado y latencia reducida:** Si los datos están en caché pueden ser entregados de memoria principal, lo que en general es más rápido que si tuvieran que ser leídos desde consultas a base de datos en disco.
- **Cargas y costes de bases de datos reducidos:** Los datos que están en caché no necesitan ser consultados a través de la base de datos, lo que reduce la carga y los costos que los proveedores de servicio asocian a ella.
- **Experiencia de usuarios mejorada:** Reduce los tiempos de respuesta que el usuario percibe, entregando una experiencia más fluida.
- **Mayor escalabilidad:** Permite procesar un volumen mayor de peticiones debido a la mejora en la latencia, ya que los datos está en caché.
- **Mejor tolerancia a fallos:** Permite a los sistemas seguir sirviendo datos - aunque puedan estar desactualizados - en momentos de alta concurrencia o mientras procesan eventos con gran latencia.

4.1.1. Estrategias de Caché

Son enfoques que definen qué sucede cuando hay un fallo de caché, lo que pasa con las

escrituras en base de datos y sobre la expiración de los datos (Tessier, 2025, sec. Caching strategies: An overview). Las principales estrategias son:

- **Write-through (Escritura directa):** Se escribe tanto a la base de datos como a la caché. La idea es mantener los datos lo más actualizados posibles. El problema de esta estrategia reside en que en alguno de almacenamientos la escritura falle, entonces entraría la inconsistencia y deben haber mecanismos para prevenir que suceda.
- **Write-behind (Escritura posterior):** Se escriben los cambios en la caché primero y después de un tiempo de retraso estos son escritos de forma asíncrona en la base de datos. Esto mejora la latencia de operaciones de escritura. El problema de esta estrategia es que si la caché falla al escribir en base de datos y no tiene los mecanismos de recuperación, habrá pérdida de datos.
- **Write-around (Escritura alternada):** Se escriben primero los cambios a base de datos y cuando se haga una petición de lectura se toman los datos hacia la caché.
- **Read-through (Lectura continua):** La caché se pone entre la base de datos y la aplicación. Cuando se solicita información se hace a través de la caché y si hay un fallo de caché entonces los datos son cargados de base de datos.
- **Cache-aside (Caché aparte):** La aplicación administra en qué situaciones se escribe hacia la caché. Los datos son cargados bajo demanda y se especifica cómo se va a comportar. El problema de este enfoque es que la primera petición que ocasione el fallo de caché va a tener más latencia de lo normal.
- **Pre-fetch (Precarga):** Los datos son pre-cargados a la caché y se manejan los eventos de actualización y modificación para que cada vez que sucedan se vean reflejados en la

caché. Lo difícil de este enfoque son los mecanismos de sincronización y de escucha de eventos que se tienen que tener para mantener la consistencia.

4.2. Latencia

La latencia, desde un punto de vista general, es un retraso de tiempo entre la causa y el efecto de un cambio físico del sistema que está siendo observado (Wikipedia, 2024).

4.3. Redis

Redis es una base de datos NoSQL, pero también es mucho más. Redis es una base de datos multimodelo que permite búsqueda, mensajería, transmisión, graficación, y otras capacidades más allá de las de un simple almacén de datos” (Suehring, 2021, 7).

4.4. Sistema de Gestión de Base de Datos

Un sistema de gestión de bases de datos (DBMS) es una tecnología de software que gestiona la información de su base de datos. Una base de datos es una recopilación de datos almacenados electrónicamente que le permite leer, escribir, eliminar y actualizar datos. Un DBMS facilita el almacenamiento de datos al tiempo que aumenta la disponibilidad, la fiabilidad y el rendimiento. También proporciona herramientas para identificar correlaciones de datos y realizar análisis dentro del sistema según sea necesario (Amazon Web Services, s. f.).

4.5. MySQL

El software MySQL ofrece un servidor de base de datos muy rápido, multihilado, multiusuario, y robusto SQL (Structured Query Languaje) (MySQL, 2024).

4.6. Java

Java es un lenguaje de programación y una plataforma informática que fue lanzada por primera vez por Sun Microsystems en 1995. Ha evolucionado desde sus humildes inicios para

empoderar una gran parte del mundo digital de hoy, proporcionando la plataforma confiable sobre la cual muchas aplicaciones y servicios son construidos (Oracle, 2022).

4.7. Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos" (Wikipedia, 2024).

4.8. Mantenibilidad en el Software

La mantenibilidad en el software según SEBoK (2025) :

> The probability that a given maintenance action for an item under given usage conditions can be performed within a stated time interval when the maintenance is performed under stated conditions using stated procedures and resources [La probabilidad que dada una acción de mantenimiento para un elemento bajo condiciones de uso dadas puede ser realizado dentro de un intervalo de tiempo establecido cuando el mantenimiento es realizado bajo condiciones establecidas usando procedimientos y recursos establecidos].

5. Metodología

El proceso general que cumple la metodología consiste en: Adaptación, Identificación, Propuesta, Realización, Validación. En las fases que están relacionadas con la Validación se utilizan métodos cuantitativos para discernir la diferencia.

5.1. Análisis del Contexto Actual

En esta fase se realizan los procesos de adaptación e identificación. Se conoce sobre cuál

contexto está enmarcada y qué problemas aborda la aplicación. Se revisan los aspectos más técnicos, como: cómo está construida, tecnologías que se usan, prácticas que tiene el equipo de desarrollo, estándares, entre otros.

Las partes que contempla son:

- Adaptación a la base del código.
- Definición de los módulos que van a ser objeto de optimización.

5.2. Propuesta de Solución

En esta fase se realizan los procesos de propuesta. Se revisa cuáles son las posibles soluciones que pueden existir o que abordan problemas similares. Se crea una solución que cubra de forma integral las necesidades del sistema así como las limitaciones humanas y técnicas que existen y se reconocieron en la fase de Análisis del Contexto Actual.

Las partes que contempla son:

- Análisis de las posibles soluciones.
- Definición de requerimientos funcionales.
- Diseño de la arquitectura final.

5.3. Implementación de Solución

En esta fase se ejecuta el proceso de realización. Se utilizan las herramientas tecnológicas, la arquitectura final y los conocimientos sociales adquiridos del grupo CALUMET. El resultado de esta fase es iterado en la fase de validación hasta que se logre cumplir con los requerimientos.

Las partes que contempla son:

- Selección del conjunto de tecnologías.

- Codificación.
- Documentación técnica.

5.4. Pruebas de Rendimiento

En esta fase se realiza el proceso de validación. Se plantea un esquema de pruebas de rendimiento que ayudan a establecer el punto de control (sistema actual) y en qué medida el sistema implementado provee mejora en la problemática. Las métricas estadísticas se usan como un medio cuantitativo y una variable central para análisis es la latencia.

Las partes que contempla son:

- Plan de pruebas.
- Condiciones técnicas para el plan de pruebas.
- Comparación de muestras.

Se desglosan los resultados de las pruebas de rendimiento y se hacen observaciones sobre los mismos.

5.5. Despliegue

Esta fase tiene la finalidad de definir los procesos, particularidades y limitaciones del sistema creado para el entorno de producción que posee actualmente el grupo CALUMET para la plataforma. Es una fase de recomendaciones y formas que pueden serle de utilidad al grupo para obtener comportamiento similar al que se dio en la fase de Pruebas de Rendimiento.

Las partes que contempla son:

- Configuración del entorno de producción.
- Monitorización.

6. Desarrollo de Proyecto

6.1. Análisis del Contexto Actual

La plataforma COMA es un proyecto de código cerrado del Grupo de Innovación y Desarrollo CALUMET. Este proyecto ha sido desarrollado a lo largo de los años por estudiantes que han sido parte del grupo. Algunos ex-miembros también han contribuido en mejoras. El proyecto tiene alrededor de 20 años en desarrollo continuo y ha sufrido cambios en la arquitectura de software debido principalmente a los problemas y obsolescencias que se han venido generando.

6.1.1. Adaptación a la Base del Código

El proyecto COMA es de tipo monolítico, con un módulo para cada funcionalidad que provee. Hay alto acoplamiento entre los módulos internos porque requieren de lógica que está definida en otros módulos que no tienen relación contextual, un efecto del código legado.

El proyecto está construido sobre Java, específicamente Java Enterprise Edition (JavaEE). Se compila, empaqueta y se despliega en un servidor de Apache Tomcat. Para la creación de los componentes visuales se utiliza la tecnología JavaServer Pages (JSP) junto con una librería interna llamada Elise.

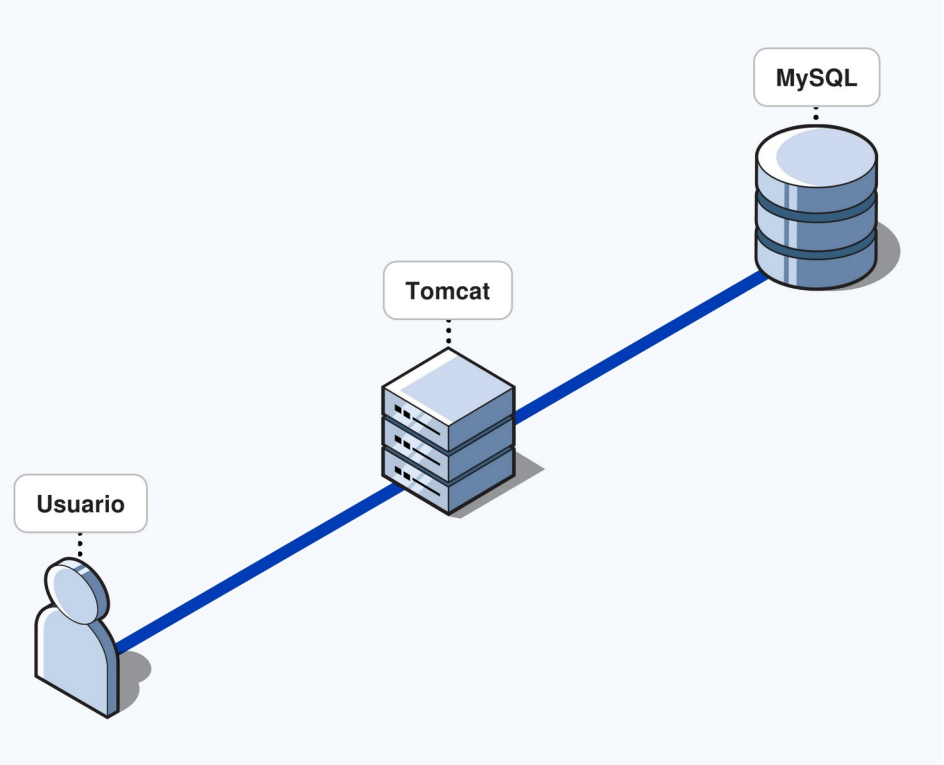
El sistema de gestión de base de datos relacional (RDBMS, por sus siglas en inglés) es MySQL y se usa para almacenar todos los datos que producen los módulos. La estructura se divide principalmente en dos bases de datos: diamante, para datos relacionados con las Escuelas; poseidon, para los datos que se comparten o se quieren compartir a nivel de toda la universidad. Hay una tercera base de datos llamada "division" pero sólo tiene usos en procesos de sincronización externos y no tiene influencia directa en las operaciones que realizan los usuarios

de COMA, por esto no se incluye en el análisis.

La herramienta Apache Ant es usada para compilar, empaquetar, agregar dependencias al proyecto, entre otras cosas. La configuración no se realiza directamente sino que se usa el Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) llamado Apache Netbeans.

Figura 1

Representación para la arquitectura de COMA



Se empezaron a realizar acciones de mejora y mantenimiento al código del proyecto como parte de reconocer los patrones, herramientas, estándares y estilo de desarrollo de software que tiene el grupo CALUMET para el proyecto COMA. Este reconocimiento tuvo el objetivo de entender como adecuar el sistema de caché al contexto real que tiene el equipo que mantiene el proyecto, para proveer una experiencia mejorada de desarrollo y una integración sin muchas

dificultades.

La primer funcionalidad realizada estuvo relacionada con el módulo de Aula Virtual. Dentro del aula virtual se necesitaba que existiera un botón para poder eliminar las notas de trabajos por subgrupos que ya se habían asignado individualmente a los estudiantes. Para este primer desarrollo se recibe guía general de cómo se relacionan los conceptos con el código y las tablas de la base de datos.

Se implementó el botón “Eliminar notas” para cada tipo de actividad. Se puede ver presente dentro de la sección “Calificar Actividades” del menú lateral izquierdo de un aula, específicamente en el cuadro etiquetado con “Calificados”.

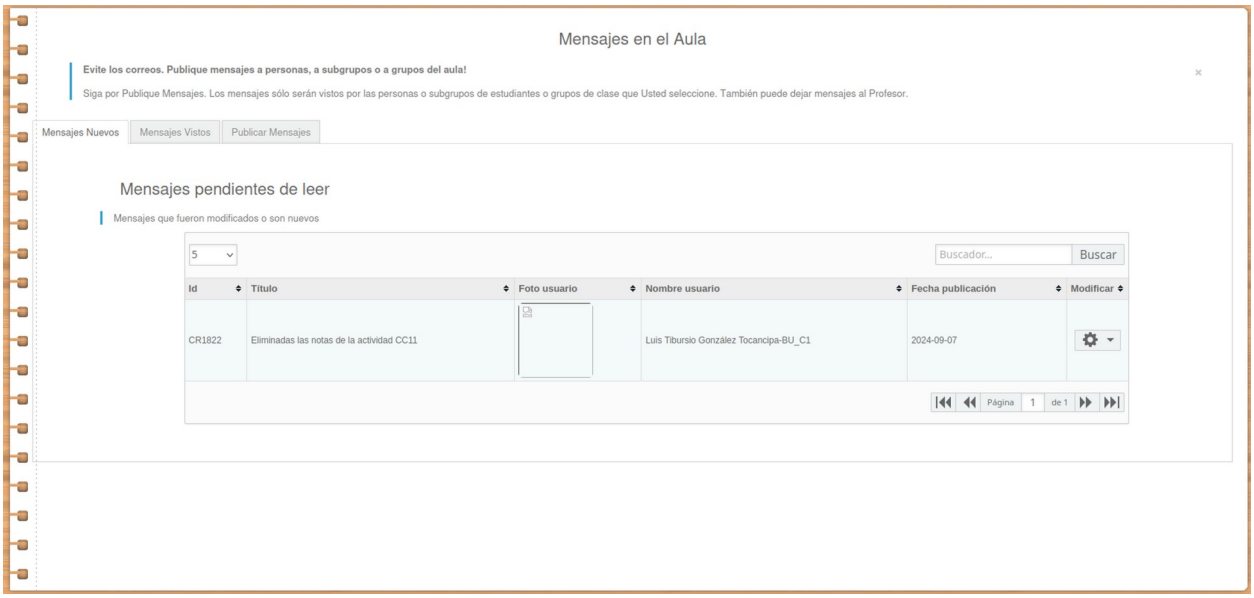
Figura 2

Visualización del botón de eliminar notas

Para cualquier actividad en el lugar indicado, se pueden eliminar sus notas. Las actividades sobre las que se realiza esto vuelven a la sección que tiene por nombre “Pendientes”. Si el profesor realizó una modificación manual en la nota de un alumno para esa actividad, entonces se notifica de estas diferencias. Las diferencias son reportadas como mensajes en el aula, escritos por el profesor (en realidad es el sistema).

Figura 3

Mensaje generado por la eliminación de notas de una actividad (laboratorio)



Para el cuerpo del mensaje se insertó HTML como mensaje, ya que, no se cuenta con una API para controlar la personalización de estos mensajes. Las diferencias son calculadas del lado del servidor. En este punto se interiorizaron los flujos de trabajo, el estilo del código y las funciones más comunes a lo largo de la plataforma.

Figura 4

Contenido del mensaje generado cuando se encuentran diferencias

El segundo desarrollo estuvo relacionado con un error en la generación de nombres para guardar archivos. Esto ocasionaba que los archivos que se entregaban en actividades se sobrescribieran. El último archivo que se entregaba se le asignaba un nombre con un formato que no podía distinguir entre entregas de diferentes actividades, por lo que, sobrescribía siempre el archivo que ya existía de anteriores entregas. Se profundizó acerca de cómo se guardan los archivos y cuáles son las prácticas para mantener esas referencias en base de datos.

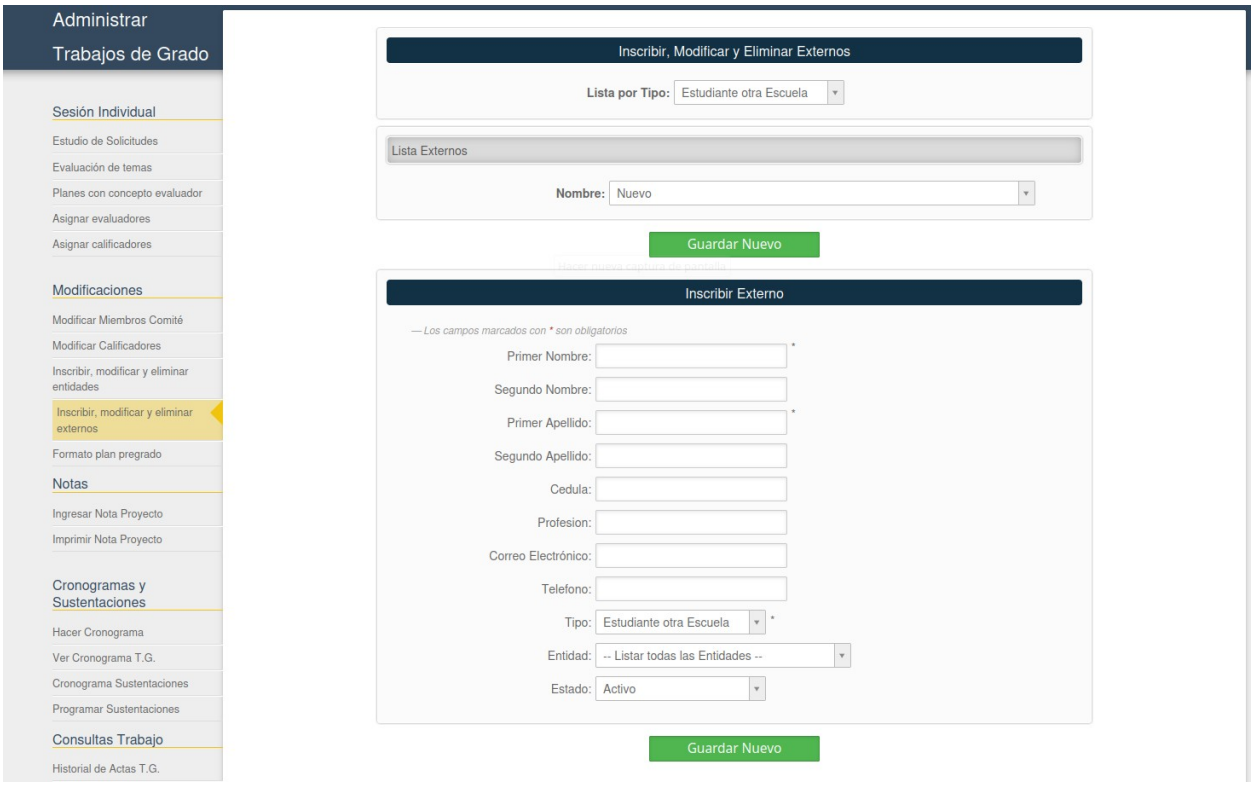
El tercer desarrollo es un arreglo de error que se reportaba cuando se querían crear subgrupos de estudiantes que estaban en un aula virtual. Este error se ocasionaba porque algunos de los estudiantes que el sistema agregaba como estudiantes del curso no estaban registrados en la plataforma. Esto se debe a que en el proceso de sincronización externo se traen los datos de los cursos y sus estudiantes, sin validar si estos estudiantes ya están registrados. En el momento de crear el subgrupo de estudiantes, hay un flujo que enviaba correos a los estudiantes avisando que ya los habían agregado a un subgrupo. Lo estudiantes no registrados no tenían ese dato en la

plataforma, por lo que se reportaba el error. Al arreglar esto se conoció más acerca de las particularidades del proceso de sincronización con la base de datos externa.

El cuarto desarrollo estuvo relacionado con la gestión de actores para el módulo de trabajos de grado. Algunos de esos actores se denominan "Externos", ya que no tienen una relación directa con la Escuela específica para la que está desplegada la plataforma. Los actores externos pueden ser estudiantes de otra escuela, profesores de otra escuela, administrativos o ajenos a la UIS. La persona que tiene rol de administrador de trabajo de grado puede crear, modificar y eliminar los externos que tiene la Escuela. Con el tiempo la cantidad de personas que eran de tipo externas se volvió muy extensa y realizar acciones directamente sobre éstos se volvió más complejo. Por esta razón se creó una interfaz independiente de la actual para gestionar acciones más específicas para los externos, de forma individual o masiva. En esta interfaz las acciones disponibles son la activación/desactivación masiva o individual, y el cambio de entidad individual. Se implementaron estas acciones porque son las que son de uso más recurrente por el administrador de trabajos de grado, que tiene que ejecutarlas de manera individual por la anterior interfaz.

Figura 5

Visualización de la anterior interfaz para la gestión de externos



Finalmente, durante gran parte del tiempo de permanencia en el grupo CALUMET también se contribuyó en labores administrativas y de dirección sobre los proyectos en curso.

6.1.2. Módulos para Optimización

En guía con el director del grupo CALUMET, se definieron los siguientes módulos candidatos para optimización:

- Página de inicio.
- Autenticación, para el inicio de sesión.
- Trabajos de grado, para la información de autores y directores.

Después de haber revisado el tamaño, el acoplamiento y el alcance dentro del proyecto se determinó que sólo se optimizaría el módulo para la Página de Inicio. Esto incluye todos los métodos externos que se invocan implícitamente para obtener la información.

La Página de Inicio requiere información de la base de datos diamante y de poseidon. Principalmente, la información está relacionada con los profesores de la escuela, los grupos de investigación y de otros tipos, las noticias propias y de otras escuelas. Otro aspecto importante sobre éste módulo es que es el primer paso para entrar a realizar acciones en la plataforma, desde ahí se redirigen a otros módulos; por lo tanto, el efecto tiene una gran cantidad de usuarios que podrán percibirlo.

6.2. Propuesta de Solución

La exploración de las soluciones estuvo determinada por las estrategias de caché del lado del servidor que existen, junto con la necesidad de que sea lo más independiente y mantenible posible. La independencia y mantenibilidad son aspectos importantes para CALUMET ya que no tienen una grupo de desarrollo de un tamaño definido y responsables con larga trayectoria. Esto hace que la independencia sea deseable, para que al seguir introduciendo mejoras los posibles errores no tengan un impacto catastrófico; la mantenibilidad también se vuelve deseable, porque se necesita que el sistema resultante pueda tener mejoras y revisiones continuamente en el futuro sin sacrificar la estabilidad.

6.2.1. Análisis de las Posibles Soluciones

Teniendo en cuenta la arquitectura monolítica de COMA, se escogieron dos estrategias de caché que serían posibles de implementar: cache-aside y pre-fetching.

En el caso de cache-aside, se propuso como un candidato porque permite que los datos que vaya necesitando la plataforma se vayan cargando bajo demanda, reduciendo el consumo de recursos sólo a lo necesario o demandado. El punto de conflicto fue determinar el proceso que indicaba cuáles datos estaban desactualizados en caché para que se sincronizaran con la base de

datos en MySQL, es decir, para mantener la consistencia.

En el caso de pre-fetching, se propuso para candidato cuándo se pensó en módulos como trabajos de grado y el de autenticación. La ventaja de tener cargados anticipadamente los datos para estos módulos ayudaría a mejorar los tiempos de respuesta para los servicios que presta; ya que, las consultas suelen implicar gran cantidad de registros o de operaciones de relación entre tablas. Pero la creación de los mecanismos de sincronización para la consistencia fuerte no son viables para la arquitectura actual en el corto plazo. Para implementar esta estrategia de forma general se tendría que hacer cambios profundos a la capa de acceso a la base de datos y a la estructura de la misma, esto último aún más complejo debido a que gran parte de las tablas tienen falta de integridad referencial.

La estrategia que se usó como punto de partida fue cache-aside, pero con modificaciones. La aplicación en sí misma no incorporó el control de la caché, sino que lo hace un sistema externo. En la aplicación sólo permanece lo mínimo para la comunicación entre ella y el sistema externo. El mecanismo para invalidar la caché se definió como expiración basada en tiempo y esta característica reside únicamente en el sistema externo.

La incorporación de un sistema externo permite que el desarrollo y todo el contexto sobre cómo administrar el almacenamiento en caché permanezca independiente de la aplicación. Este sistema puede seguir manteniéndose de forma paralela y la intervención en la aplicación sólo es necesaria en el momento en el que se hagan cambios en la forma de comunicación. Los detalles concernientes a la comunicación con la base de datos subyacente y el DBMS de caché se abstraen de la aplicación y se pueden gestionar de manera independiente. Esto también hace que la aplicación sea agnóstica con respecto a qué tecnología se utilizar como DBMS de caché o la

que se usa en sí misma para implementar la lógica de administración.

6.2.2. Definición de Requerimientos Funcionales

Los requerimientos funcionales comprenden capacidades de almacenamiento en caché para el sistema externo. Algunos son necesarios y otros son deseables. En resumen, los requerimientos diseñados son:

- Sincronizador de la información en caché con la residente en el servidor MySQL (Ver Apéndice A).
- Registro de nuevas consultas para almacenar en la caché (Ver Apéndice B).
- Permitir consultas con o sin filtros (Ver Apéndice C).
- Permitir consultas combinadas (Ver Apéndice D).
- Limpieza de datos que ya no son tan usados (Ver Apéndice E).
- Monitor de estado del sistema (Ver Apéndice F).

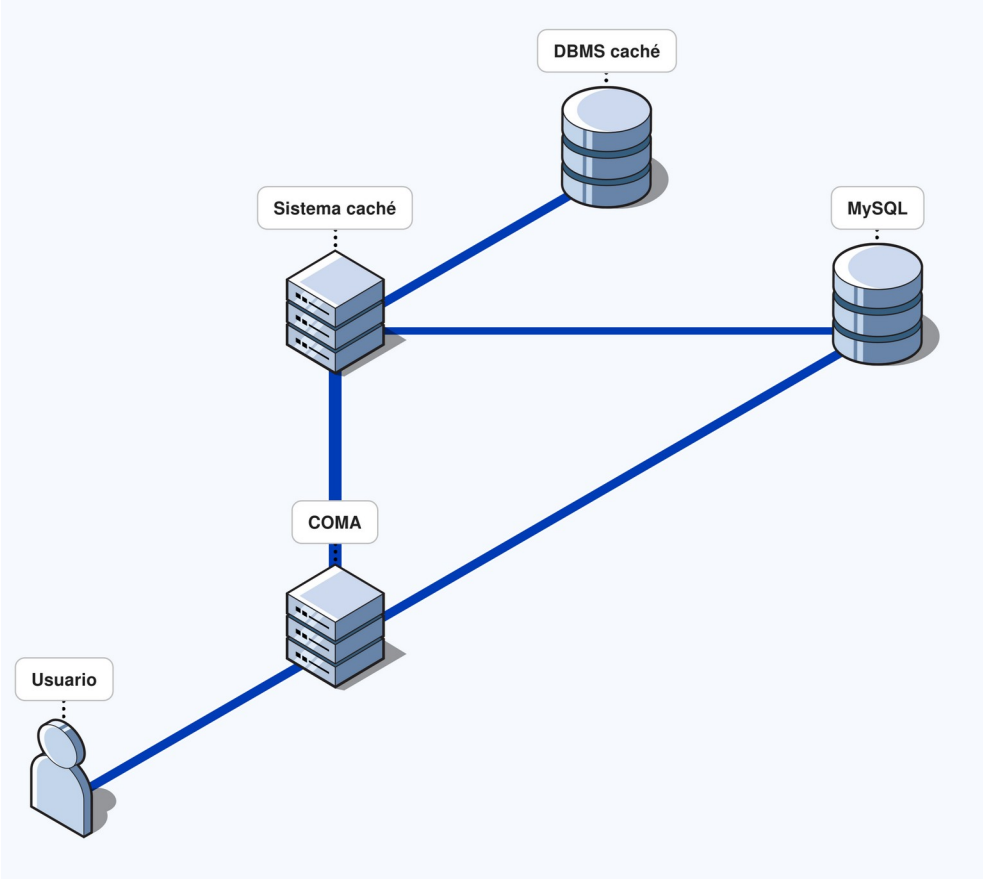
6.2.3. Diseño de la Arquitectura Final

Para la arquitectura final se dividió en dos niveles: Infraestructura del sistema de caché y Arquitectura del sistema de caché.

6.2.3.1. Infraestructura del sistema de caché. Se detalla cómo los elementos del sistema de caché se integran con los elementos de la plataforma COMA. La infraestructura contempla el sistema externo, que administra todo lo relacionado con el almacenamiento en el DBMS de caché; también el DBMS para hacer de caché, que es el que permite el almacenamiento en memoria principal.

Figura 6

Diagrama de la infraestructura del sistema final

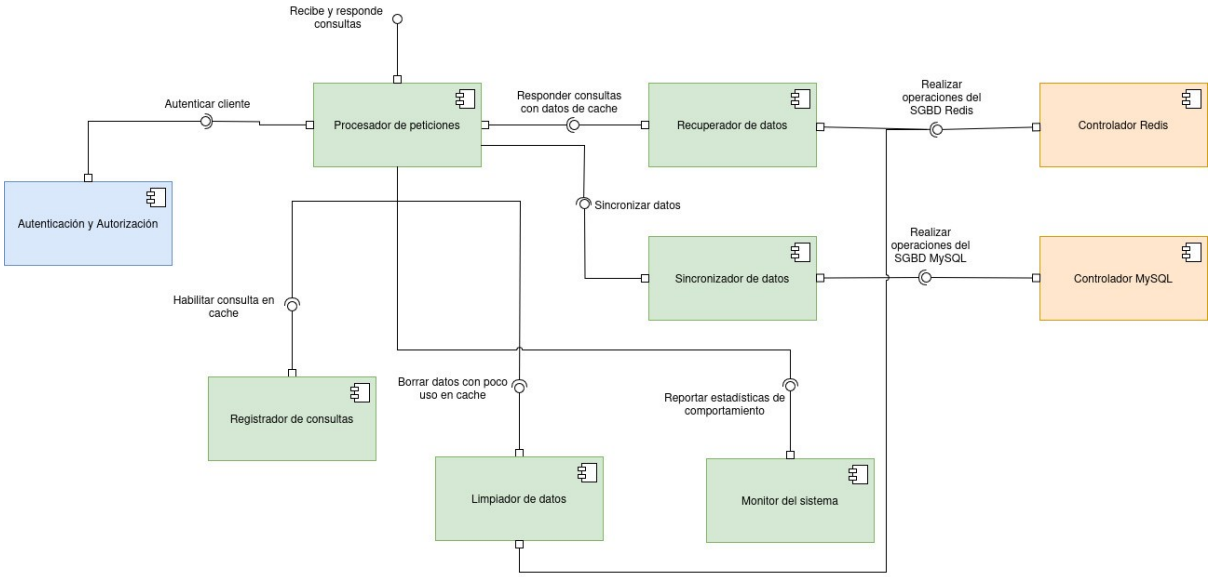


El “sistema cache” va a actuar como un servidor externo el cual recibirá y procesará peticiones a través de su API. Va a ser accesible sólo para el servidor que aloja la lógica de la aplicación.

6.2.3.2. Arquitectura del sistema de caché. El diseño de la arquitectura se basó en la creación de componentes que pudieran cumplir los requerimientos funcionales. Se definió la mínima cantidad de componentes necesarios tomando en cuenta que algunos componentes fueron dispuestos por terceros.

Figura 7

Diagrama de componentes del sistema externo



Nota. Los componentes en verde corresponden a los nuevos, los azules y los naranja los que proveen terceros.

Cada componente se especializa en un aspecto de la operación del sistema, que permitió la realización o apoyo de uno o más requerimientos funcionales. El componente Procesador de Peticiones es una parte clave, porque decide qué funciones ejecuta de otros componentes y la información que les suministra. Los componentes controladores ya vienen desarrollados por las organizaciones que mantienen el desarrollo del DBMS de caché y el RDBMS, entonces sólo se integró el código de configuración. El componente de Autenticación y Autorización también fue implementado a través de librerías preexistentes.

Tabla 1

Descripción general del propósito de cada componente

Nombre	Requerimiento funcional que apoya o realiza	Descripción

Autenticación y Autorización	Todos	Se encarga de permitir el ingreso de peticiones de clientes autorizados.
Procesador de peticiones	Apoya: RF2, RF3, RF4.	Procesa todas las peticiones entrantes, valida su consistencia y las redirige hacia el componente especializado en responderlas.
Recuperador de datos	Realiza: RF3, RF4.	Obtiene los datos necesarios de caché para dar respuesta a consultas.
Sincronizador de datos	Realiza: RF1.	Mantiene los datos actualizados del servidor Redis con los de MySQL.
Controlador Redis	Apoya: RF3, RF4.	Realiza operaciones específicas de DBMS Redis.
Controlador MySQL	Apoya: RF1.	Realiza operaciones específicas de RDBMS MySQL.
Registrador de consultas	Realiza: RF2.	Amplía la cantidad de consultas registradas como disponibles en el sistema.
Limpiador de datos	Realiza: RF5.	Selecciona y elimina el conjunto de datos que tengan muy poco uso en el DBMS Redis. Realiza ajustes necesarios para mantener la consistencia.
Monitor del sistema	Realiza: RF6.	Obtiene parámetros de los distintos componentes del sistema, los condensa en indicadores estadísticos y los reporta.

6.3. Implementación de la Solución

6.3.1. Selección del Conjunto de Tecnologías

Las tecnologías candidatas para implementar el sistema tuvieron en cuenta restricciones específicas del equipo de trabajo en el grupo CALUMET, así como consideraciones técnicas y de viabilidad con el tiempo propuesto. Se agruparon las propuestas acorde al área específica que podría ayudar a implementar. Las tecnologías para el servidor deben ser para el lenguaje de programación Java como restricción específica.

Tabla 2

Tecnologías candidatas para implementar la infraestructura

Elemento	Tecnología
Servidor	Marco de trabajo Spring Boot
Servidor	Apache Tomcat
Servidor	Eclipse Glassfish
Servidor	WildFly
Servidor	Undertow
DBMS de caché	Redis
DBMS de caché	Memcached
DBMS de caché	Volt DB

La elección en tecnologías para el servidor estuvo determinada por el factor de curva de aprendizaje.

- El marco de trabajo Spring Boot es el de más baja curva de aprendizaje debido a que está enfocado en tener mínima configuración acompañado de gran cantidad de abstracciones, pero esto mismo hace que hacer modificaciones puntuales por fuera de los establecido

sean más complejas.

- Apache Tomcat es una implementación de un conjunto de especificaciones de Jakarta EE Platform (Apache Software Foundation [ASF], s. f., sec. Home). Permite manejar Servlets y JSP, también el despliegue de aplicaciones es muy directo en archivos Web Application Archive (WAR).
- Eclipse GlassFish es un servidor de aplicaciones de código abierto ligero pero potente que implementa completamente la plataforma Jakarta EE. Diseñado para ofrecer flexibilidad, escalabilidad y fiabilidad, proporciona un entorno listo para producción que cumple estrictamente con estándares abiertos sin dependencias propietarias (Eclipse Foundation, s. f.).
- WildFly es un entorno de ejecución de aplicaciones flexible, ligero y administrado (Commonhaus Foundation [CF], s. f.). También cumple por completo con la plataforma Jakarta EE
- Undertow es un servidor web flexible y de alto rendimiento escrito en Java, que proporciona API bloqueantes y no bloqueantes basadas en NIO (*Undertow Home*, s. f.). Se puede usar entre la especificación Servlet o solamente usar manejadores no bloqueantes de bajo nivel.

Entre las opciones para servidor la que más se hizo fácil de adoptar fue Apache Tomcat. La primera razón es que la plataforma en sí misma ya lo utiliza, entonces el equipo de desarrollo está más acostumbrado al flujo de trabajo. La segunda razón es que aunque no provee de todas las especificaciones de Jakarta EE, las que posee ya lo hacen adecuado para el uso que se le quería dar.

La elección para la tecnología DBMS de caché se centró en Redis y Memcached. VoltDB como candidato fue descartado debido a que carece de funcionalidades importantes como: eliminación automática y expiración basada en tiempo; además, no es puramente un almacenamiento clave-valor, en cambio, usa SQL que supone para las consultas un tiempo de ejecución más elevado en comparación con los otros candidatos. Para el caso de Memcached es una buena opción si sólo queremos tener almacenamiento caché sin preocupaciones por la persistencia y la variedad en estructuras de datos; sin embargo, para lograr la mayor flexibilidad en cómo se guarda la información y las políticas de eliminación de datos se eligió Redis.

6.3.2. Codificación

La codificación de la solución final tuvo dos bases de código en las que se tuvo que dividir. Una base de código relacionada con el cliente que permite comunicar COMA con el sistema de caché, la otra base de código fue el sistema de caché en sí.

Para el desarrollo se tuvieron que tener en cuenta versiones específicas de Java y de MySQL. La versión de Java JDK fue la 21 y para MySQL la versión 5.7.44. La restricción de Java 21 se aplicó para el código del cliente, ya que, esta versión es la que usa COMA. La restricción de MySQL se debe principalmente a la librería que se usa para conectarse a él desde Java y a problemas de las bases de datos en sí mismas, ya que, en versiones más recientes de la librería no se permite operar los resultados si se cierra la conexión y por el lado de la base de datos la integridad referencial es más estricta en versiones recientes (como la versión 8 y 9).

Para el sistema de caché se utilizaron las siguientes versiones:

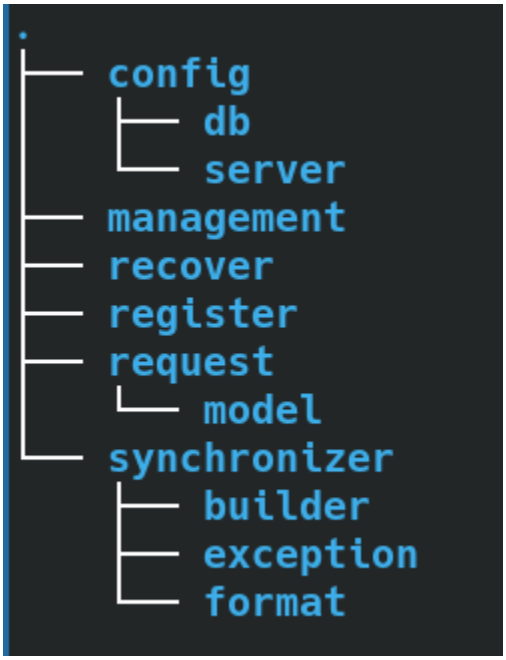
- Apache Tomcat 11.0.9
- JDK 24

- SQL Statement Parser 1.0[5]
- Redis 8.0.2

Para cada componente se creó un paquete con el fin de mantener lo más modular posible el sistema.

Figura 8

Árbol de paquetes



Cada paquete siguió una convención de estructura similar. Se hizo una clase en la que se pusieron métodos de apoyo o que realizan acciones particulares sobre Redis o MySQL. Esta primera clase se encarga de hacer operaciones que por sí solas no cumplen ningún requerimiento. En la segunda clase, se le ponía el sufijo "Service" después del nombre que lleva la primera clase (clase utilitaria o de operaciones) y en ella se declararon los métodos que utilizando la primera clase cumplen alguno de los requerimientos.

5 Esta es una librería propia que se usó para analizar las consultas SQL.

Figura 9

Ejemplo de la convención de clases en el paquete "Register"

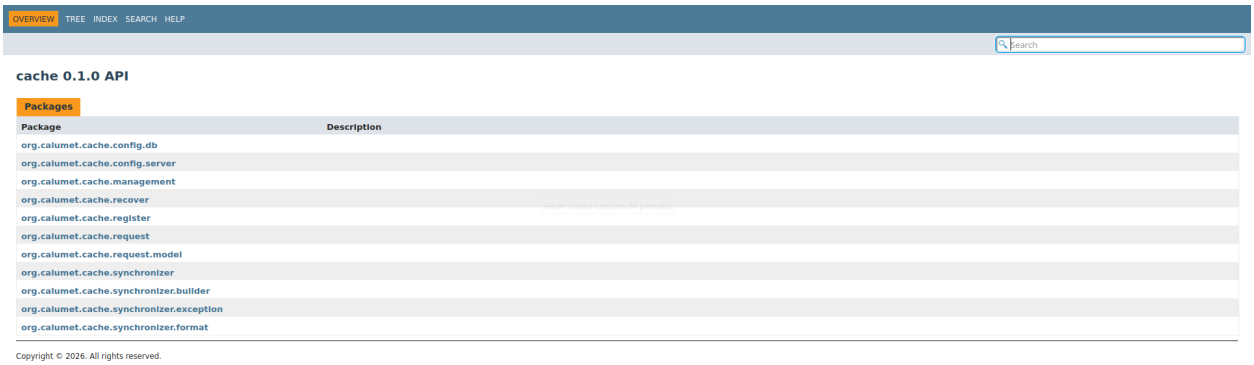
En todas las clases del primer tipo se creó una forma de reportar los errores y de mostrar otros mensajes con objetivos de depuración del programa. Estos registros se lograron a través de la librería SLF4J y de la librería nativa para hacer registros (más conocidos como *logs*) de servidor.

6.3.3. Documentación Técnica

Para la documentación se tienen dos niveles: documentación técnica a nivel de código o a nivel de sistema. La documentación técnica a nivel de código es la que concierne a los métodos, atributos y otras declaraciones; en la cuál se explica la funcionalidad o los detalles de implementación. Esta documentación es generada con la herramienta Javadoc, la cual genera un sitio web con lo que se haya puesto en comentarios de documentación para cada una de las declaraciones.

Figura 10

Javadoc del sistema de caché



La documentación técnica a nivel de sistema se detalla en cada una de las bases de código, específicamente en el archivo llamado "README.md". En este se describe la forma en cómo instalar, configurar, modificar (dado caso) y ejecutar cada uno; es decir, uno para el cliente y otro para el sistema de caché. Se hizo de esta manera porque lo que se describe no es extenso y en su totalidad está compuesto por texto.

6.4. Pruebas de Rendimiento

Estas pruebas se enfocaron en pruebas de carga, que tienen el fin de garantizar que el sistema procese la carga que se requiere (Sommerville & Velázquez, 2011, p. 227). La verificación con estas pruebas fue determinante para cuantificar en qué medida la plataforma COMA se veía afectada.

6.4.1. Plan de Pruebas

Se crearon 5 niveles de carga para ejecutar en el escenario sin y con caché. El plan está enfocado enteramente en la funcionalidad de la Página de Inicio. Cada nivel tiene una carga que

se mantiene estable durante una ventana de tiempo de 2 minutos.

Tabla 3

Niveles del plan de pruebas

Nivel	Carga propuesta [rps]
1	1
2	3
3	5
4	7
5	10

Nota. La sigla rps viene de "requests per second" que significa solicitudes por segundo.

6.4.2. Condiciones Técnicas para el Plan de Pruebas

Las pruebas se ejecutaron en un equipo de escritorio. Tanto la base de datos MySQL como Redis fueron instalados de manera local y los servidores Tomcat también estuvieron en ejecución en puertos diferentes.

Tabla 4

Especificaciones de técnicas del equipo de pruebas

Especificación	Nombre
Sistema operativo	Debian GNU/Linux 12 (bookworm)
CPU	Intel i7-7700K (8) @ 4500GHz
RAM	15944MiB
Red	Killer E2500 Gigabit Ethernet Controller

La ejecución de las pruebas no fue inmediata, se tomaba un tiempo entre prueba y prueba

para que los objetos y otras referencias que se generaron en memoria durante la prueba sean limpiados del sistema.

6.4.3. Comparación de Muestras

Para cada uno de los niveles se graficó: frecuencia relativa de los códigos de respuesta (códigos HTTP) para verificar si existían fallos, gráficos de caja para observar la distribución de la latencia, gráficos de puntos para analizar la dispersión o tendencia, gráficos de densidad de probabilidad para analizar el comportamiento y series de tiempo para observar el comportamiento de la latencia en el tiempo.

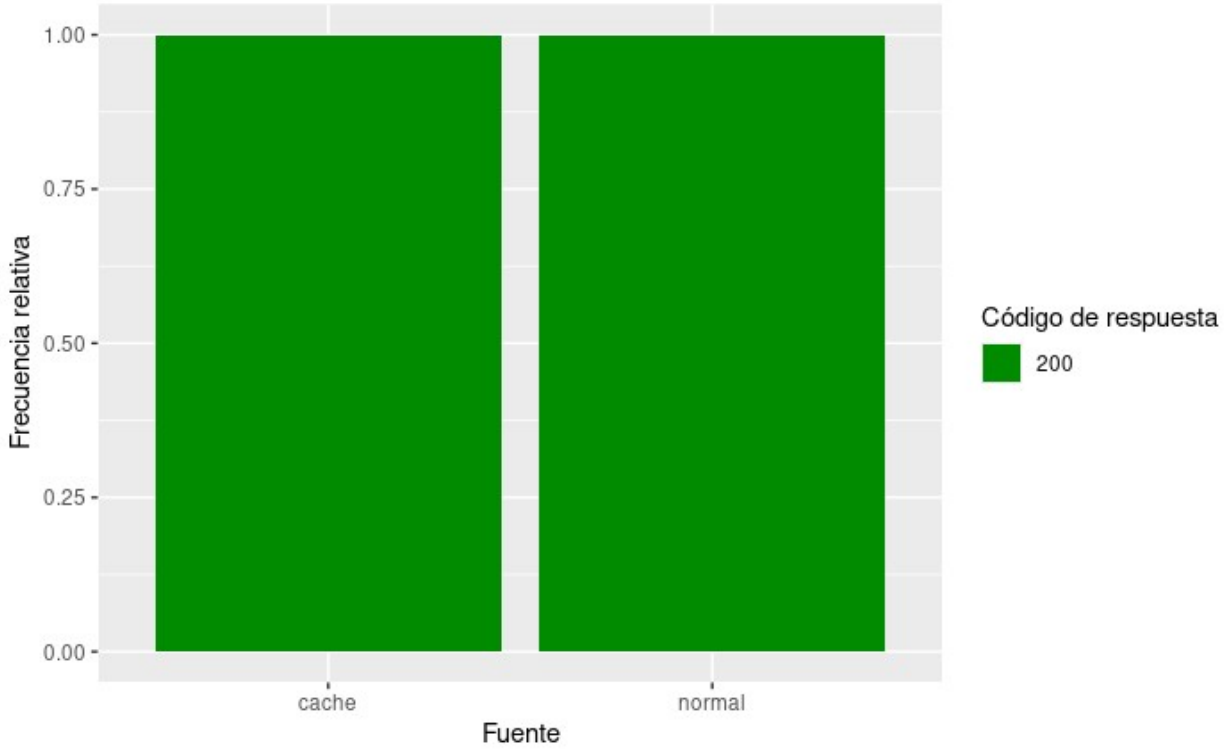
En las pruebas se definió una variable llamada "Fuente" en la cual el sistema sin cache se refiere como "normal", mientras que el sistema con caché se refiere como "cache". Para las figuras de códigos de respuesta, el significado de cada código corresponde a:

- 200: Indica que la petición terminó exitosamente.
- 500: Ocurrió un error no especificado del lado del servidor.
- ConnectionClosedException: Esto no es propiamente un código HTTP, es un error reportado de la herramienta JMeter. Lo que indica es que se estaba recibiendo la respuesta pero el servidor cerró la conexión de forma abrupta, sin terminar la respuesta.

6.4.3.1. Nivel 1. Todas las respuestas fueron exitosas.

Figura 11

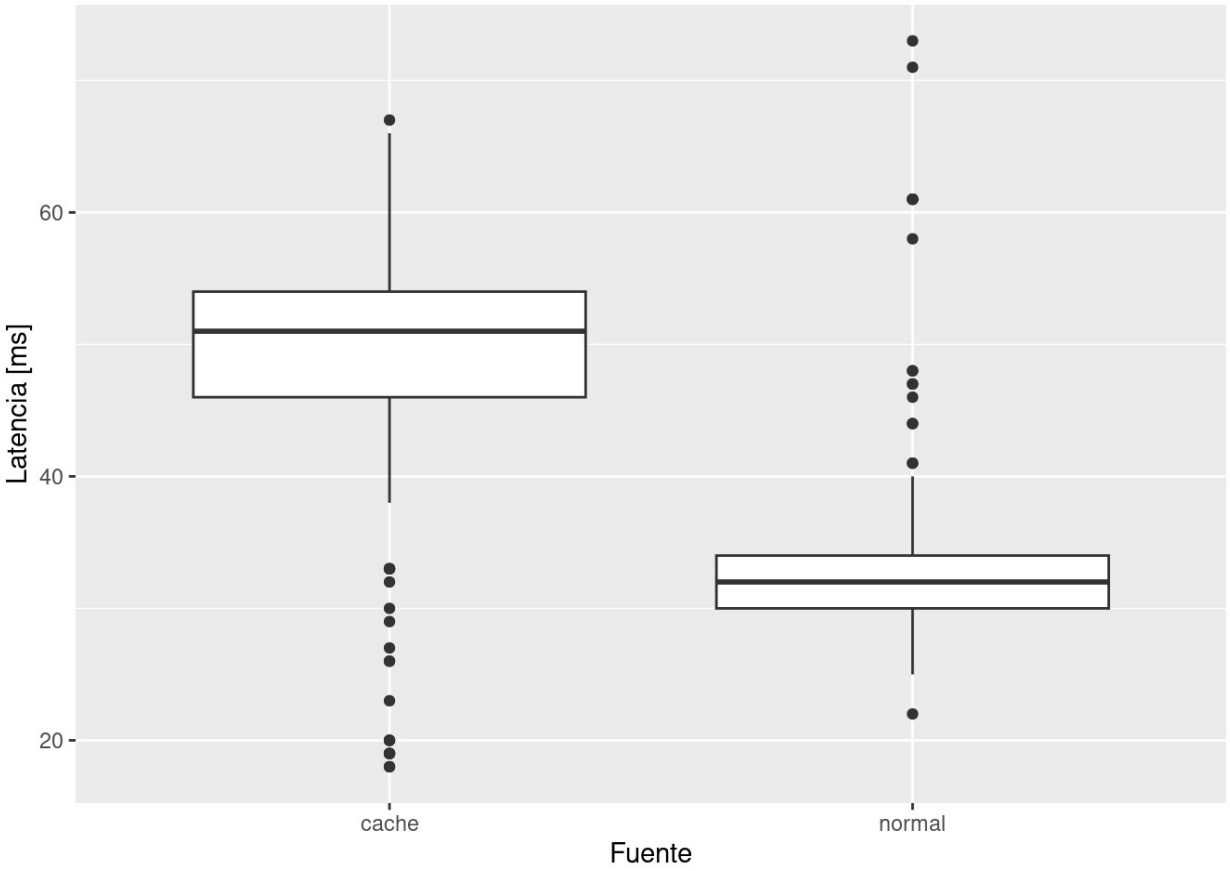
Códigos de respuesta para 1rps



La latencia para el fuente normal tuvo un promedio de 34.62903[ms] con una desviación de 8.514424[ms]; mientras que, para fuente cache tuvo un promedio de 47.47581[ms] con una desviación de 11.228205[ms]. La Figura 12 muestra que la mayor parte de latencia en la fuente cache fue superior y más dispersa que la fuente normal, se presentan puntos atípicos en ambos casos. Para los puntos atípicos de la fuente cache, tienden a estar en latencias más pequeñas, mientras que para la fuente normal tienden a estar en latencias más grandes.

Figura 12

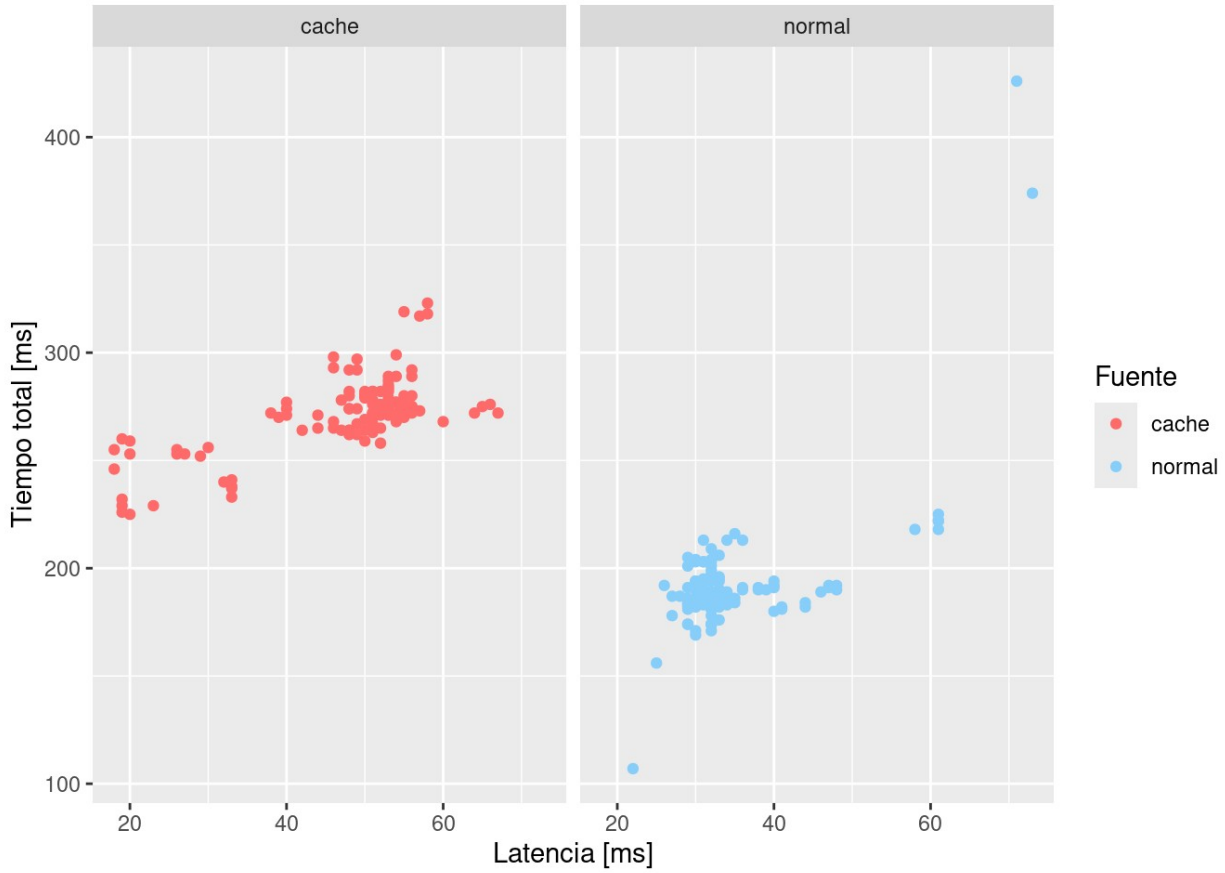
Gráfico de caja para 1rps



En la Figura 13 no se puede determinar tendencia o relación aparente entre las variables.

Figura 13

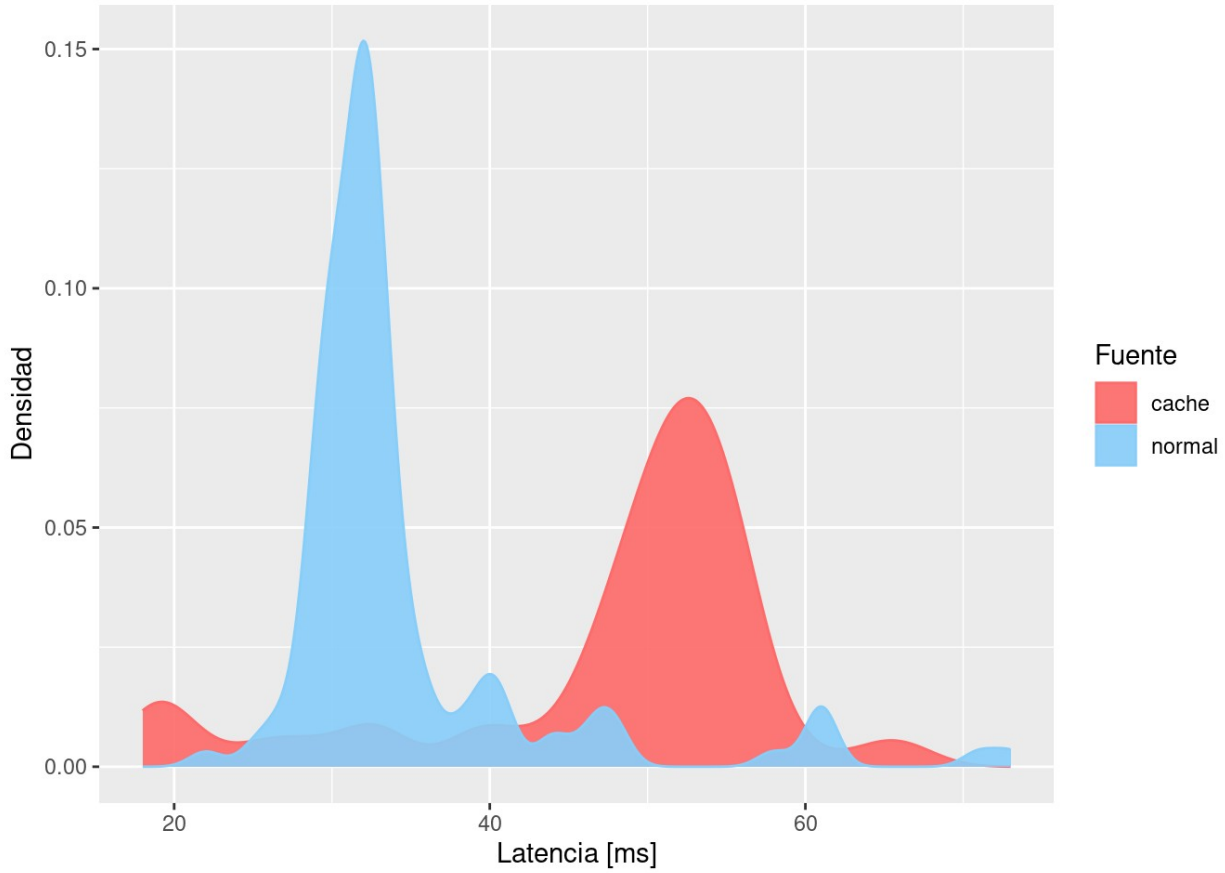
Dispersión de tiempo total contra latencia para 1rps



En la Figura 14, la fuente normal tiene alta densidad entre los 25[ms] y 40[ms]; mientras que, la fuente cache tiene su mayor densidad entre los 40[ms] y 60[ms] pero no tan fuerte como la fuente normal.

Figura 14

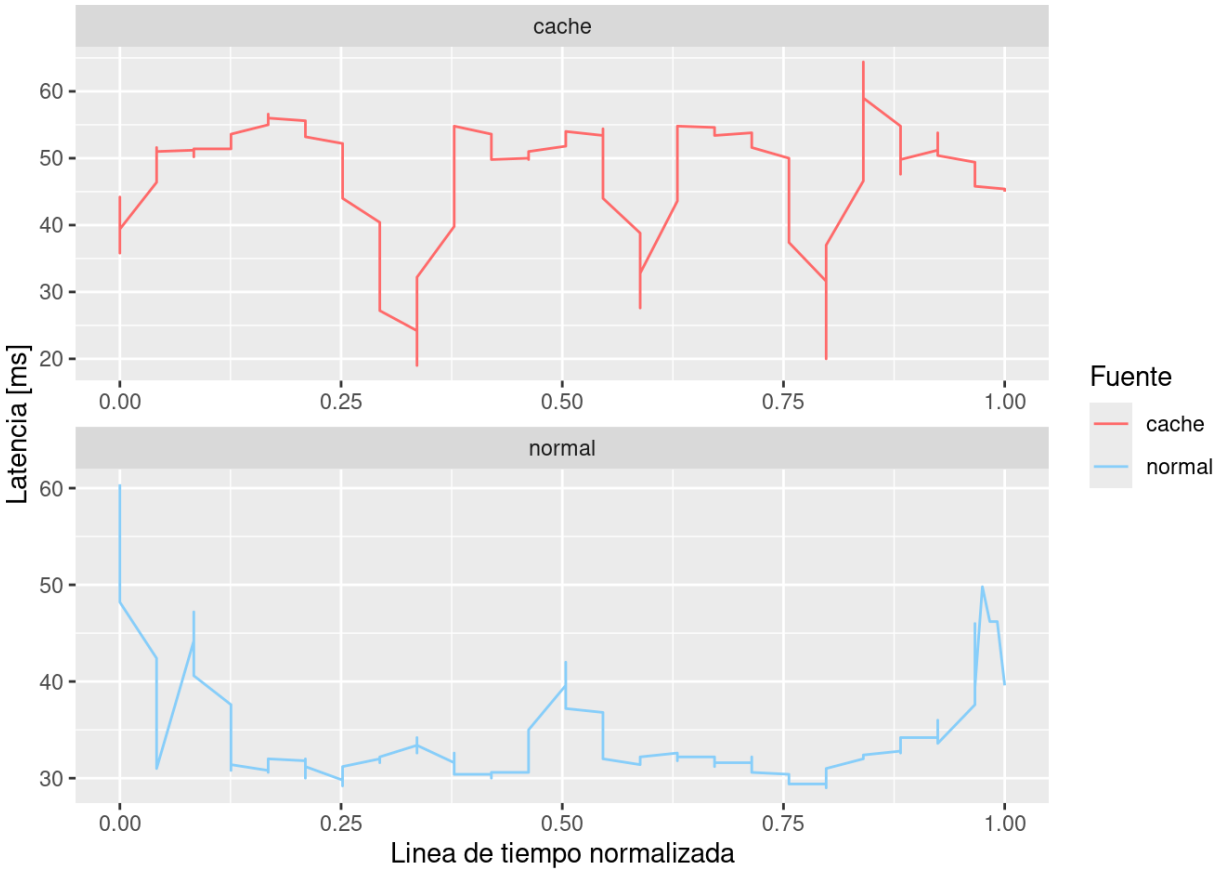
Curvas de densidad para 1rps



Finalmente, la Figura 15 muestra que la fuente cache tuvo un comportamiento más errático que la fuente normal.

Figura 15

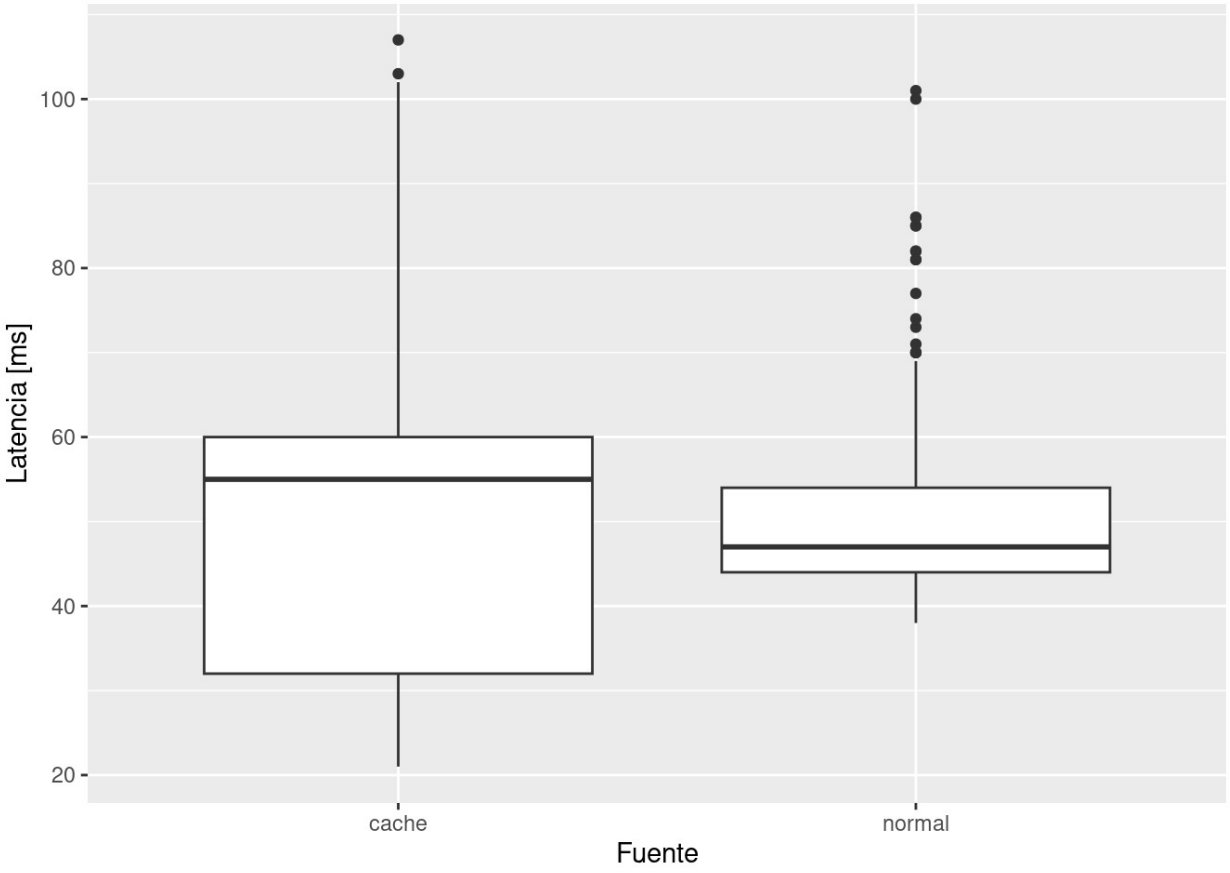
Serie de tiempo para 1rps



6.4.3.2. Nivel 2. Todas las respuestas fueron exitosas. La latencia para la fuente normal tuvo un promedio de 50.14402[ms] con una desviación de 9.651328[ms]; mientras que, para fuente cache tuvo un promedio de 50.44293[ms] con una desviación de 18.693192[ms]. La Figura 16 muestra que para la fuente cache, el 50% de la latencia se localiza en valores inferiores a 60[ms] pero superiores a 30[ms], pero esta vez la dispersión se extiende hasta los 100[ms]; mientras que, la fuente normal se mantiene el 50% de la latencia en valores inferiores a 55[ms] y superiores a 40[ms] aproximadamente, con una dispersión más pequeña que la fuente cache y sólo se extiende hasta los 70[ms]. Para la fuente normal se presenta una cantidad superior de datos atípicos que para la fuente cache.

Figura 16

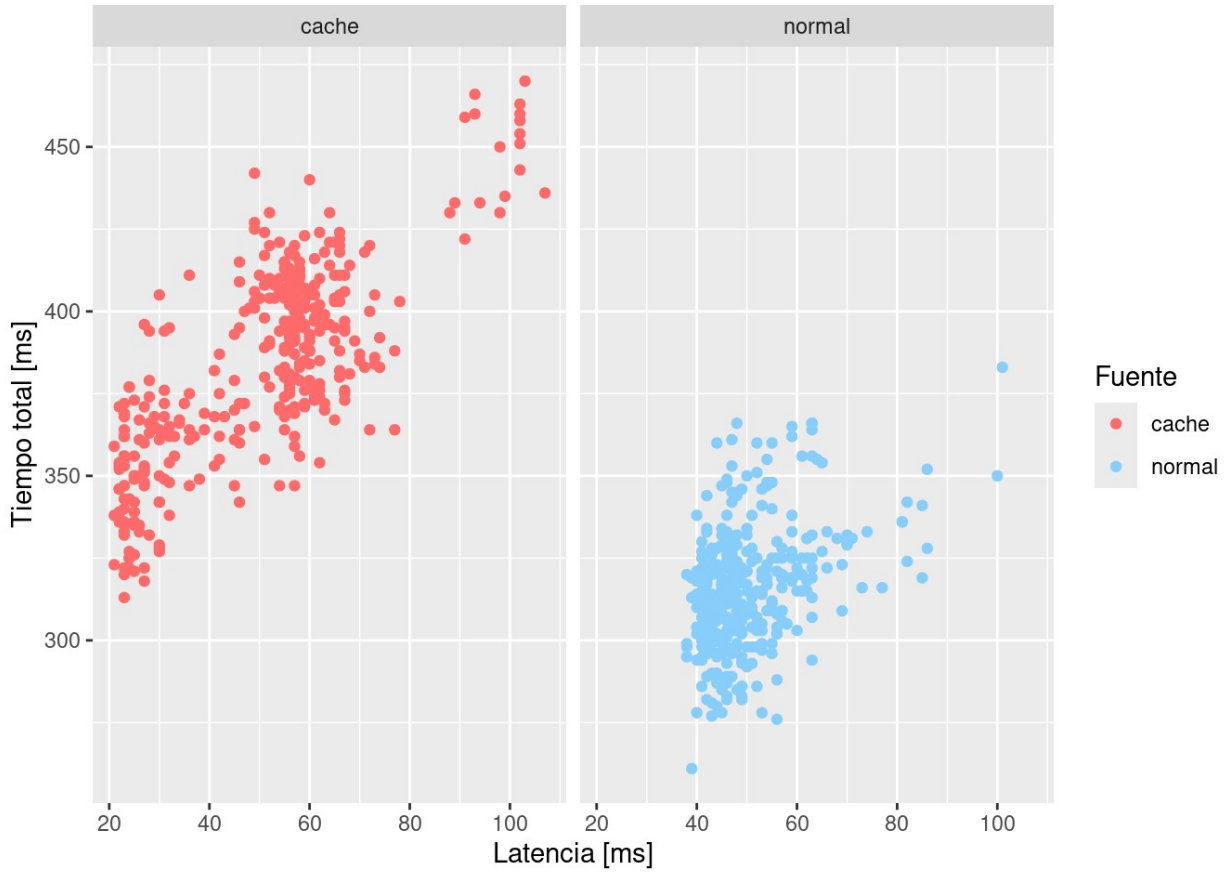
Gráfico de caja para 3rps



En la Figura 17 se muestra que para la fuente cache existen dos nodos en los cuales se concentran los datos; mientras que, para la fuente normal se tiene un solo nodo en el que los puntos se agrupan muy cercanamente. La tendencia del tiempo total para la fuente cache es de mayor latencia con respecto a los valores de la fuente normal.

Figura 17

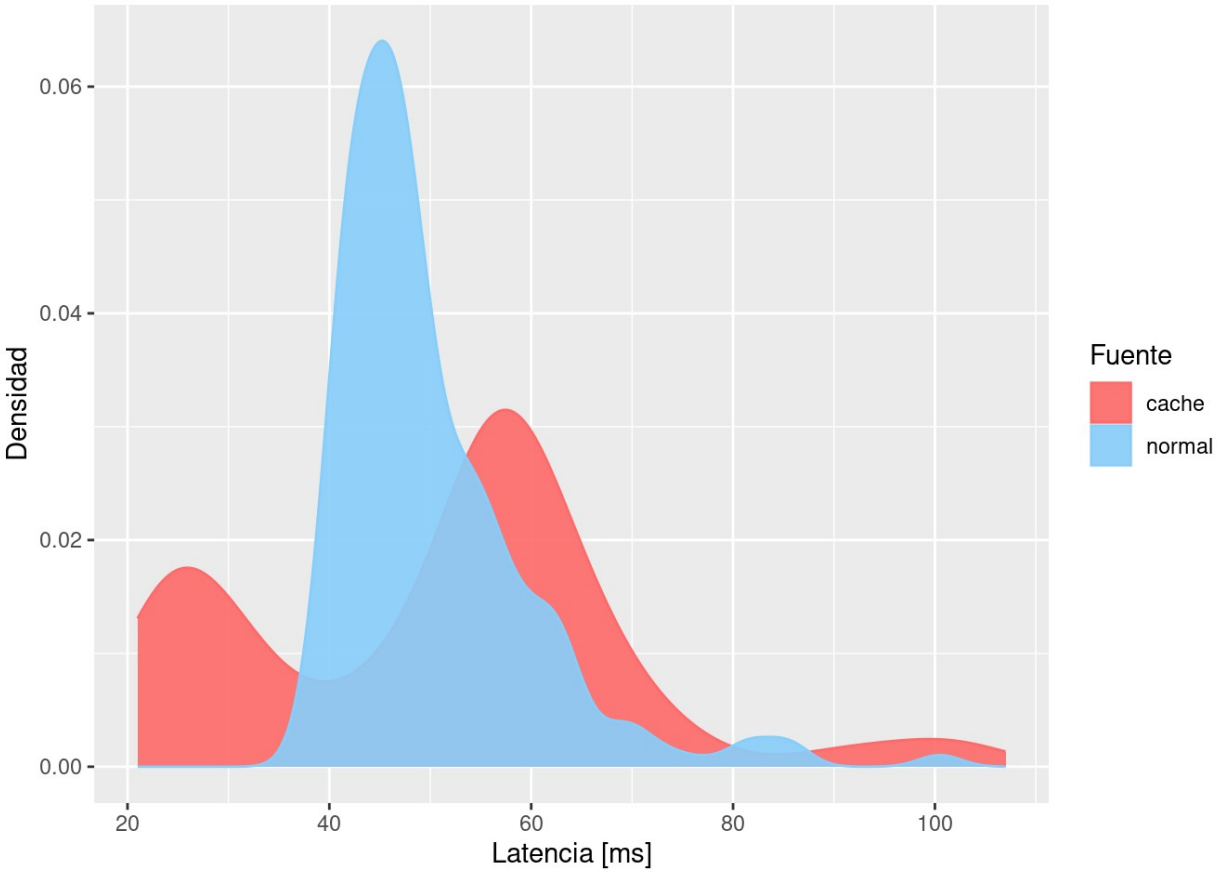
Dispersión de tiempo total contra latencia para 3rps



Para la gráfica de densidad de la Figura 18, la fuente normal muestra densidad concentrada entre los 30[ms] y los 80[ms] aproximadamente; mientras que, la fuente cache muestra una distribución bimodal en la que el pico de mayor densidad se encuentra entre los 40[ms] y los 80[ms] aproximadamente.

Figura 18

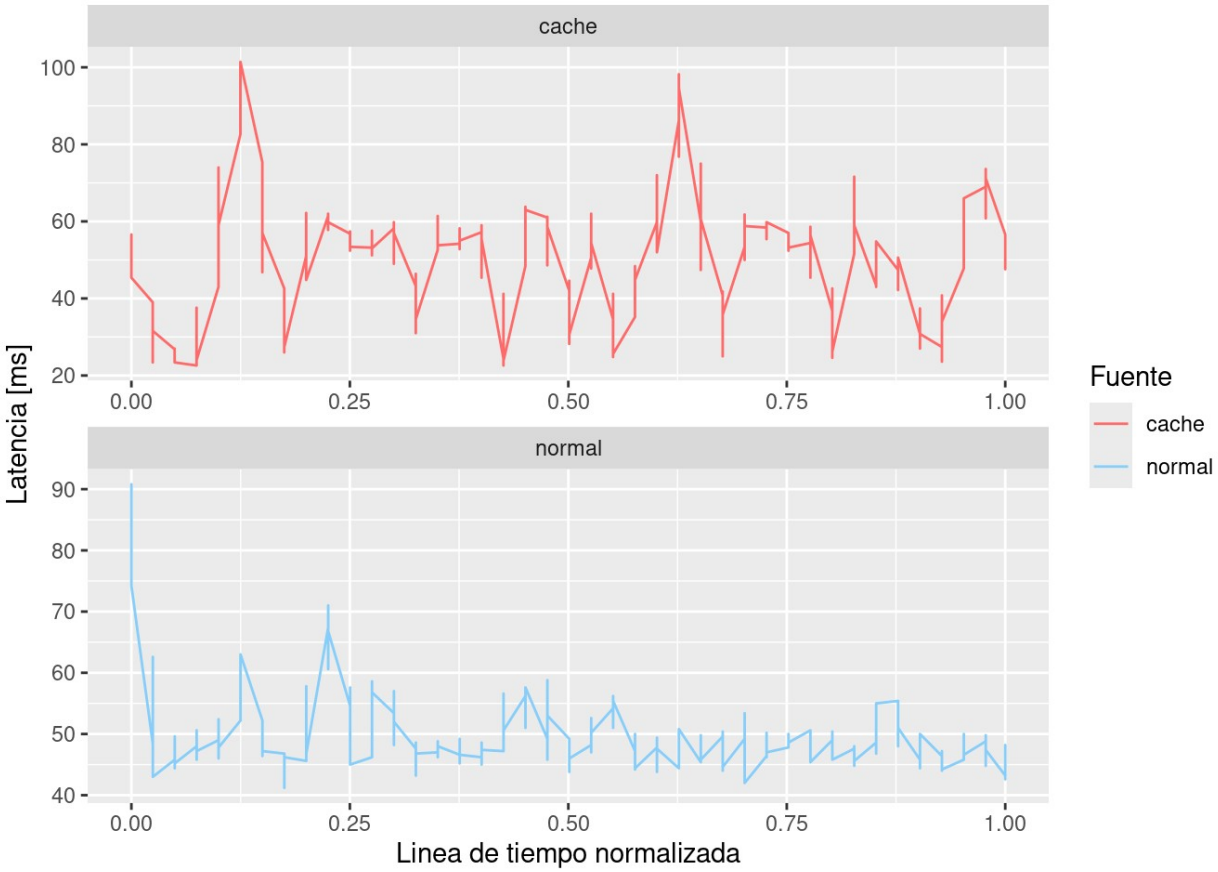
Curvas de densidad para 3rps



En la Figura 19, la serie de tiempo para la fuente de cache muestra un comportamiento oscilante, con algunos picos por encima de los demás. Para la fuente normal se muestra un comportamiento más estable alrededor de los 50[ms].

Figura 19

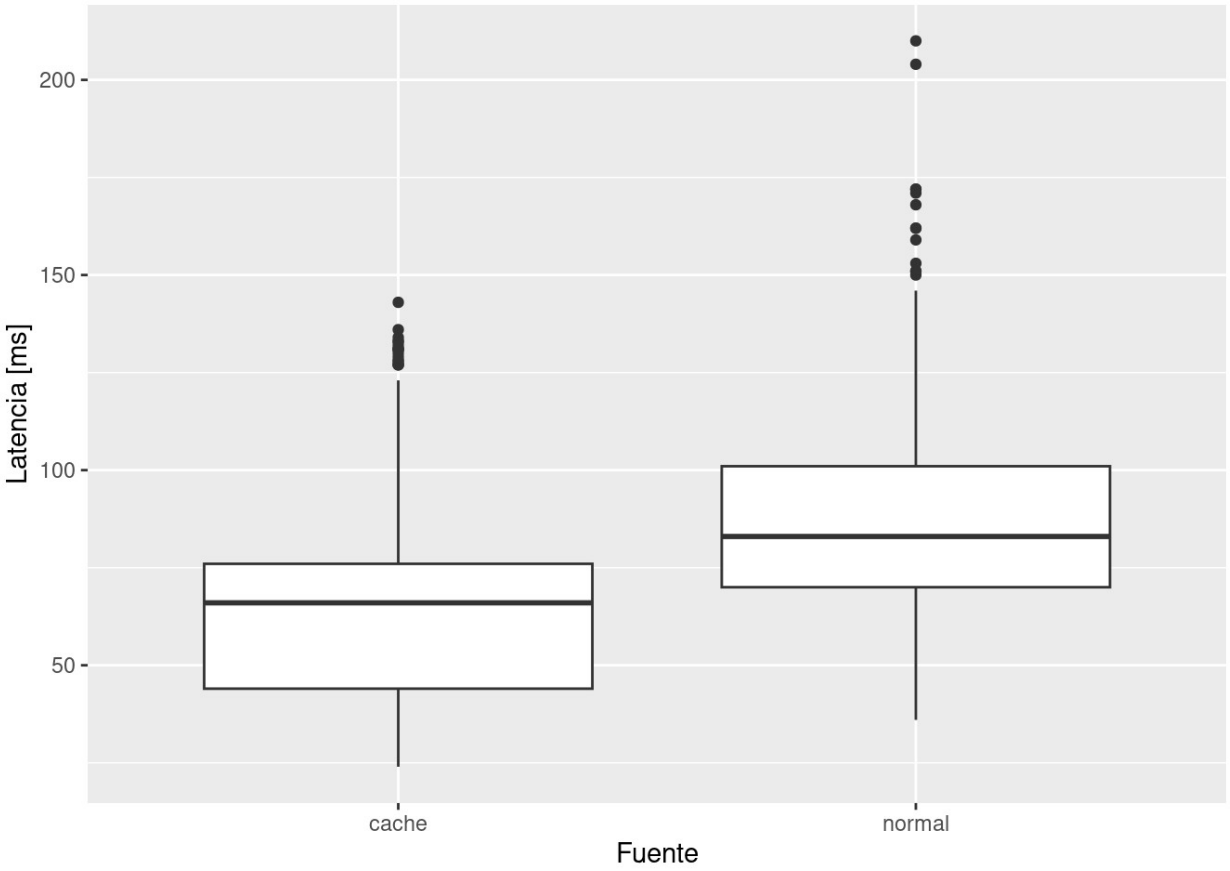
Serie de tiempo para 3rps



6.4.3.3. Nivel 3. Todas las respuestas fueron exitosas. La latencia para la fuente normal tuvo un promedio de 86.73127[ms] con una desviación de 24.01049[ms]; mientras que, para fuente cache tuvo un promedio de 64.28664[ms] con una desviación de 23.46938[ms]. La Figura 20 muestra que para la fuente cache, el 50% de la latencia se localiza en valores inferiores a 80[ms] pero superiores a 40[ms], pero esta vez la dispersión se extiende hasta los 125[ms]; mientras que, la fuente normal se mantiene el 50% de la latencia en valores inferiores a 100[ms] y superiores a 60[ms] aproximadamente, con una dispersión similar a la fuente cache que se extiende hasta los 150[ms].

Figura 20

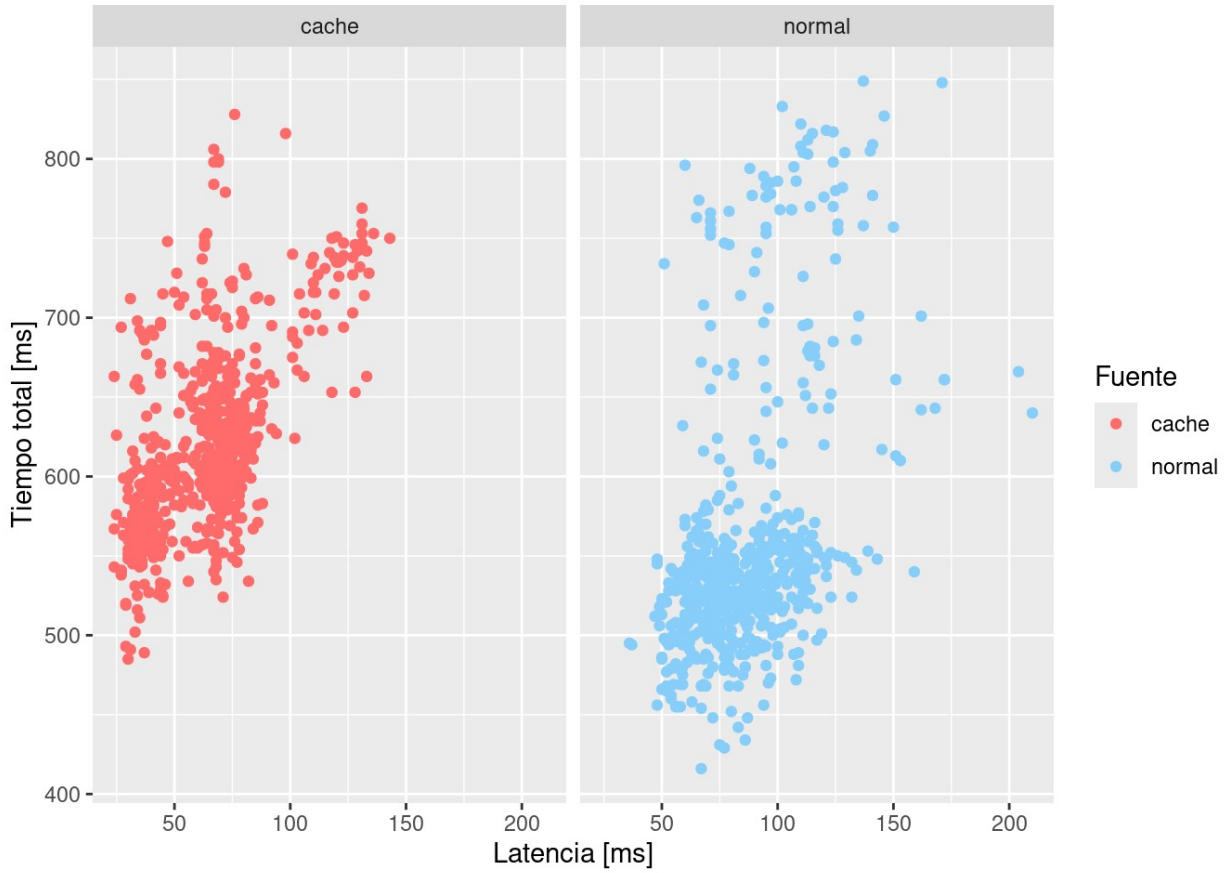
Gráfico de caja para 5rps



En la Figura 21, la fuente cache muestra dos nodos cercanos con alta concentración, también existen puntos dispersos sobre los 700[ms] para el tiempo total. Para la fuente normal se observó un nodo con alta concentración situado alrededor de los 510[ms] para el tiempo total y varios puntos dispersos sobre los 600[ms] para tiempo total. Los puntos que no se acercan a los nodos, para el caso de la fuente normal, tienden a estar más dispersos y más distantes con respecto al mismo grupo en la fuente cache.

Figura 21

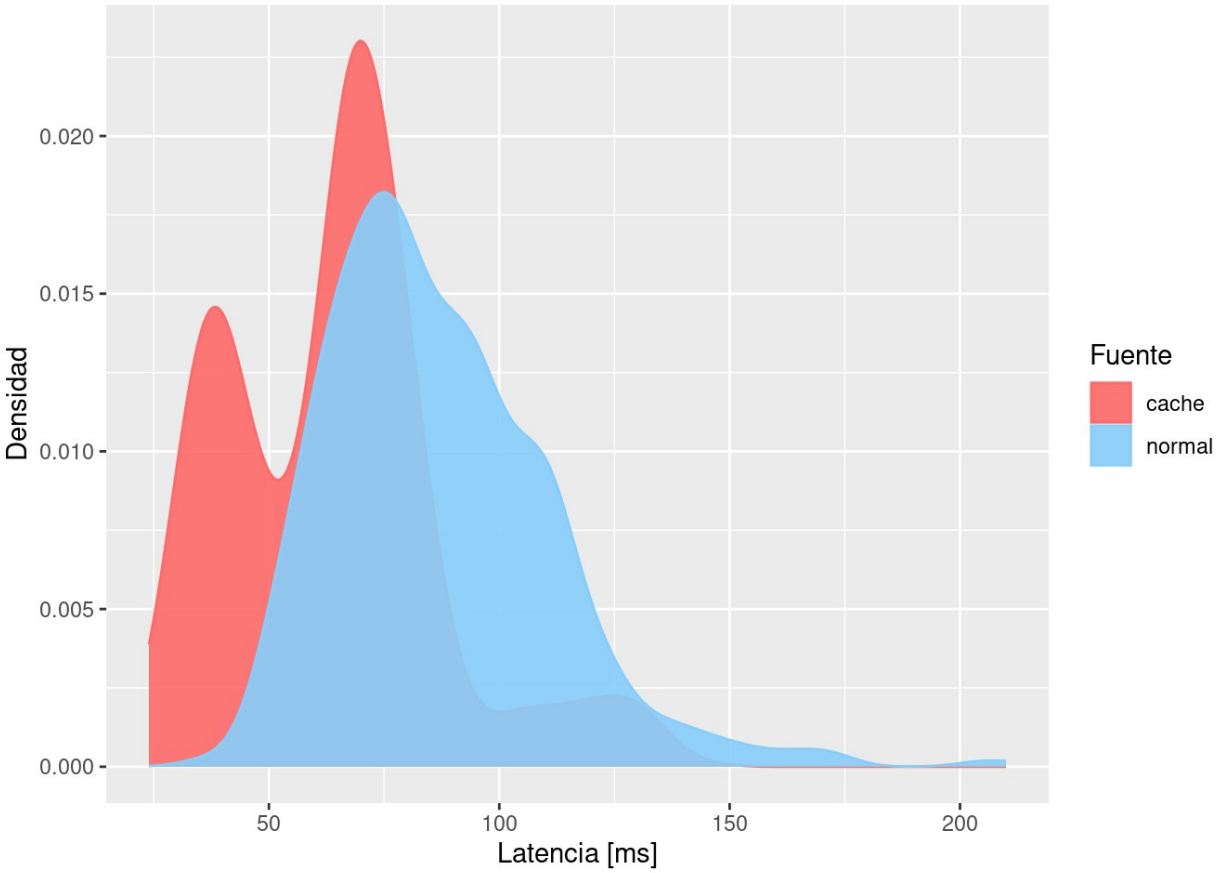
Dispersión de tiempo total contra latencia para 5rps



En la Figura 22, la fuente normal muestra un rango más amplio para el pico, comprendido entre 0[ms] y 150[ms]; mientras que, la fuente cache mantuvo una forma bimodal comprendida por el rango 0[ms] y 100[ms] (ambos picos). La fuente normal se muestra más dispersa en latencia pero se mantiene con un solo pico; por otro lado, la fuente cache se muestra más concentrada pero se divide en dos modos.

Figura 22

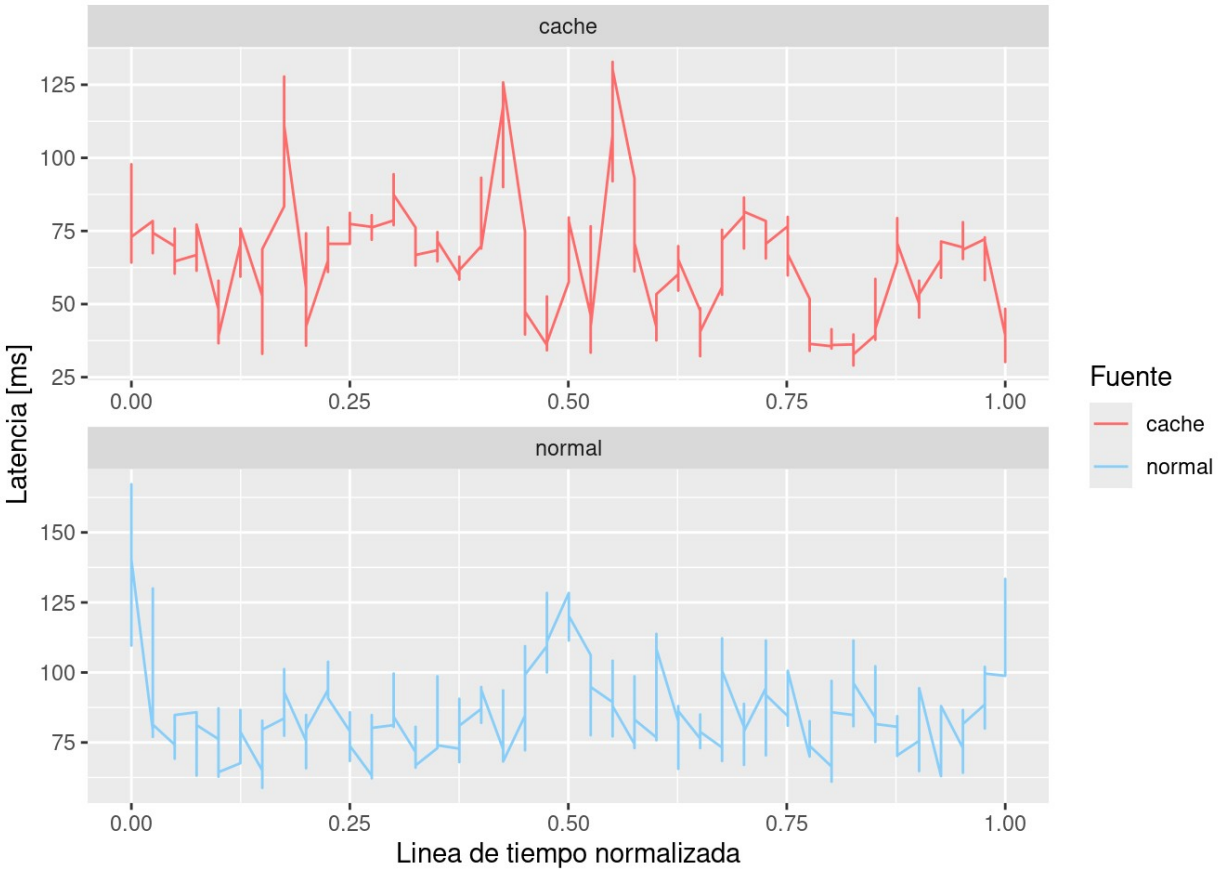
Curvas de densidad para 5rps



La serie de tiempo de la Figura 23, muestra para la fuente normal oscilaciones alrededor de los 75[ms] aproximadamente, con un pico sobrepasando los 100[ms]. En el caso de la fuente cache no hay un eje de oscilación claro y posee 3 picos sobrepasando los 100[ms].

Figura 23

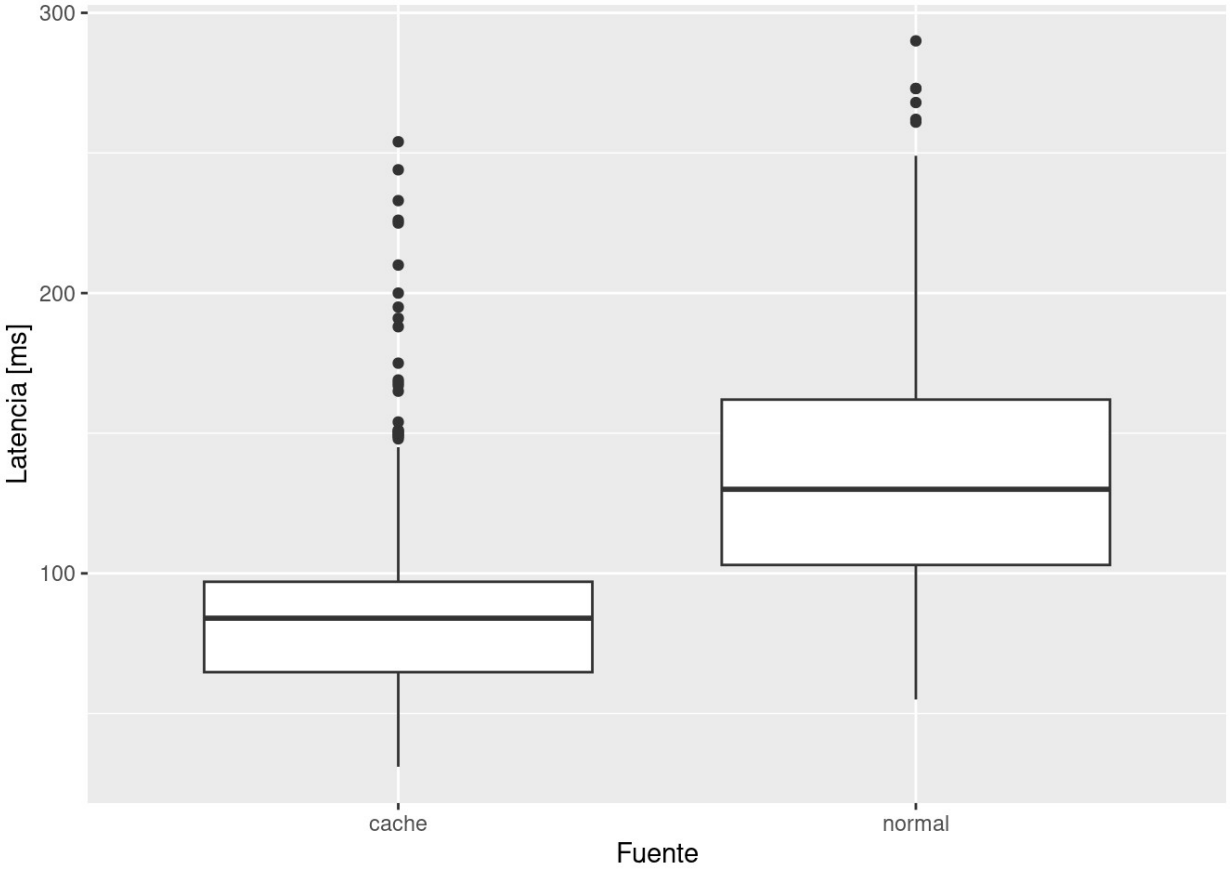
Serie de tiempo para 5rps



6.4.3.4. Nivel 4. Todas las respuestas fueron exitosas. La latencia para la fuente normal tuvo un promedio de 135.37674[ms] con una desviación de 42.20356[ms]; mientras que, para fuente cache tuvo un promedio de 83.75581[ms] con una desviación de 28.31390[ms]. La diferencia es más grande tanto en promedio como en desviación estándar, en la Figura 24 se evidencia la diferencia de concentración. El gráfico de caja muestra que la fuente normal está por encima de los 100[ms] mientras que fuente cache está por debajo de ese umbral.

Figura 24

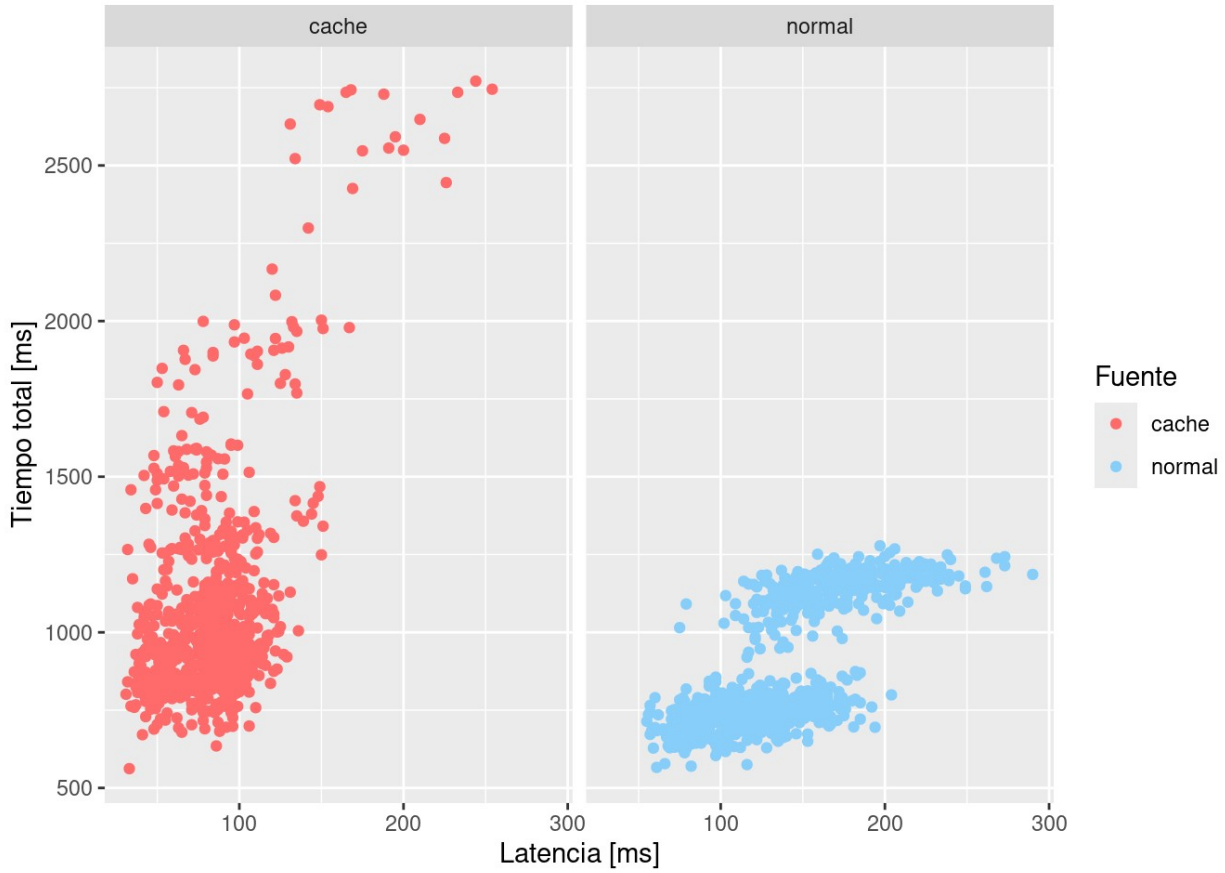
Gráfico de caja para 7rps



En cuanto a la dispersión, mostrada en la Figura 25, hay cambios significativos. La fuente cache mantiene la tendencia a concentrarse en un nodo, pero aún ahora presenta puntos mucho más distantes. La fuente normal exhibe una tendencia muy fuerte hacia dos nodos, no hay puntos distantes.

Figura 25

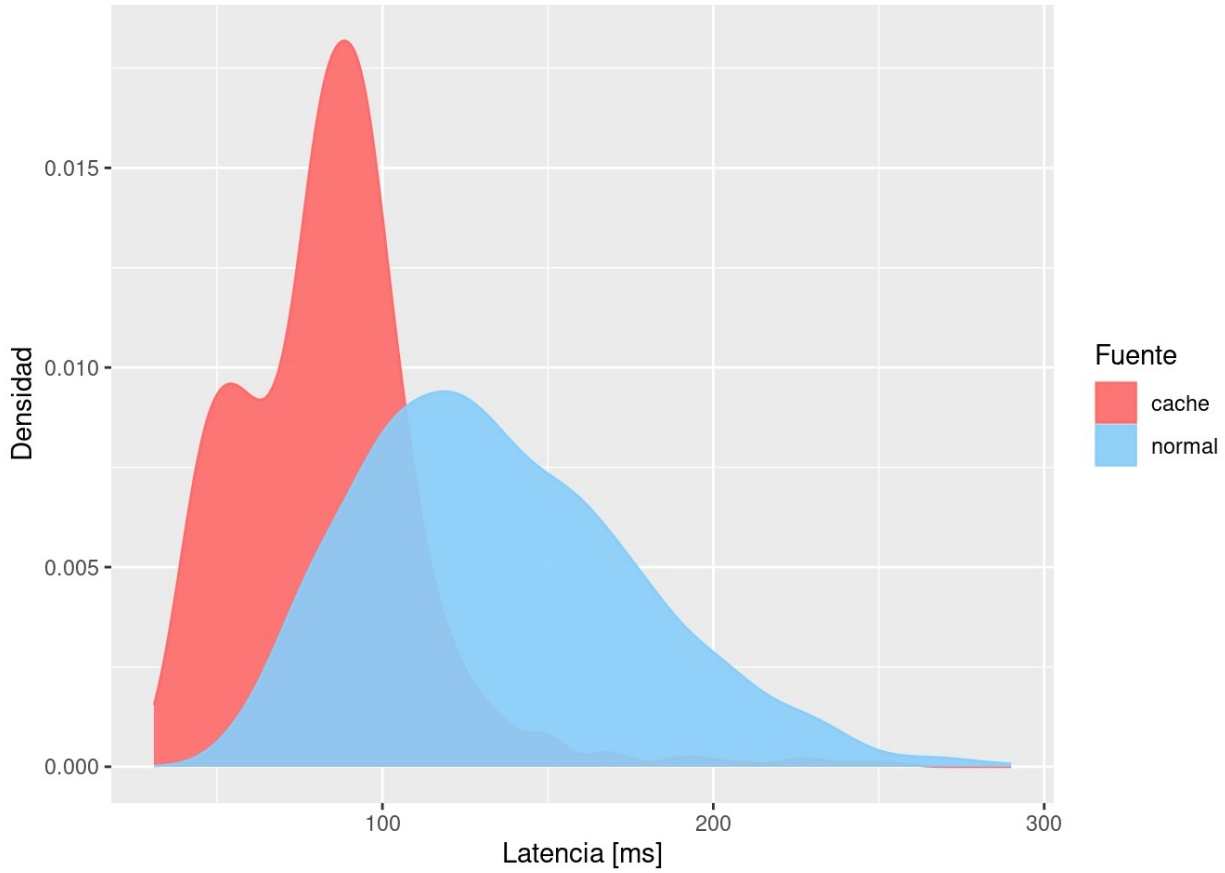
Dispersión de tiempo total contra latencia para 7rps



En la Figura 26, existe un cambio importante en las formas de la densidad con respecto al anterior nivel. La fuente cache muestra una tendencia a concentrarse para formar un solo pico; mientras que, la fuente normal sigue aumentando su desviación estándar pero mantiene su forma.

Figura 26

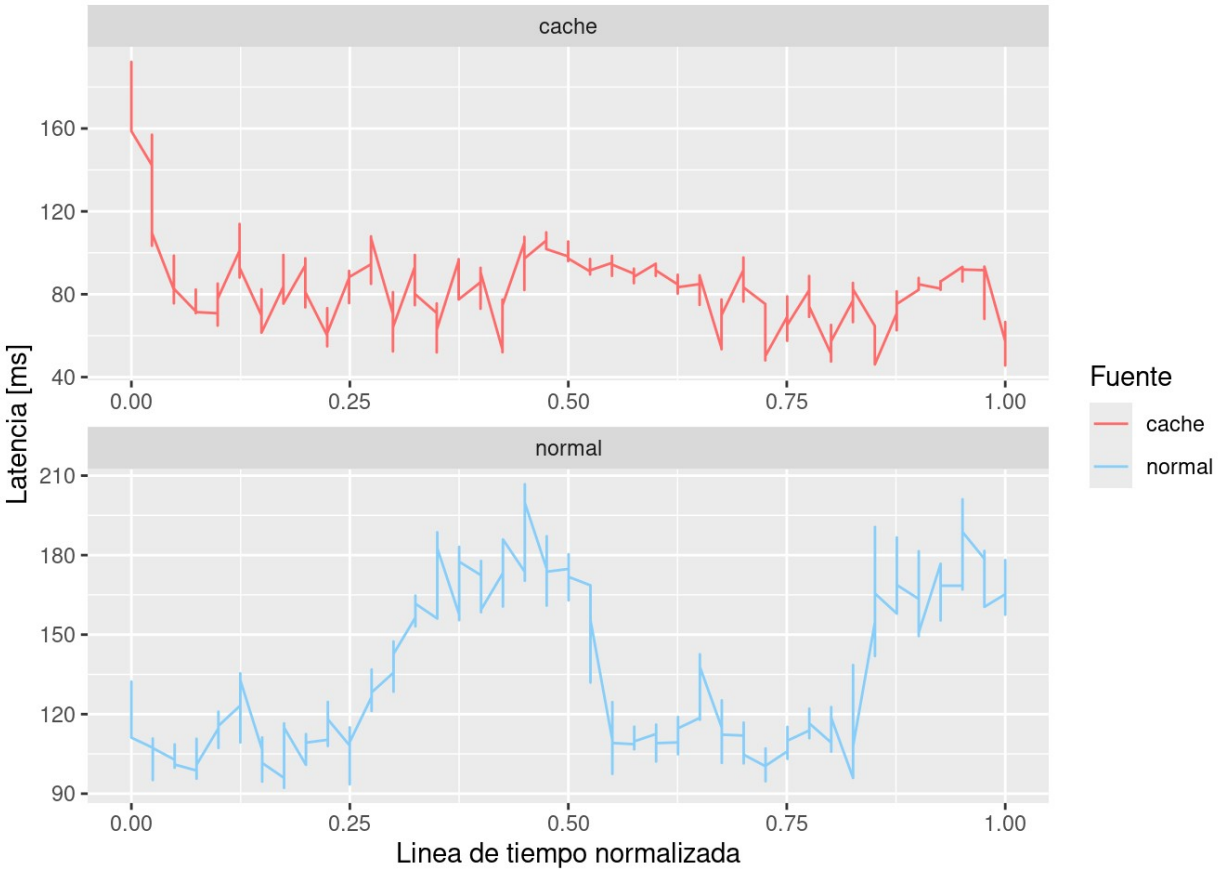
Curvas de densidad para 7rps



En la Figura 27, la fuente cache aún muestra oscilaciones pero ya se comporta más estable alrededor de 80[ms]; mientras que, la fuente normal muestra oscilaciones más agresivas y no hay estabilidad alrededor de algún valor. Los picos de la fuente normal se sitúan alrededor de los 180[ms] y los de la fuente cache alrededor de los 100[ms].

Figura 27

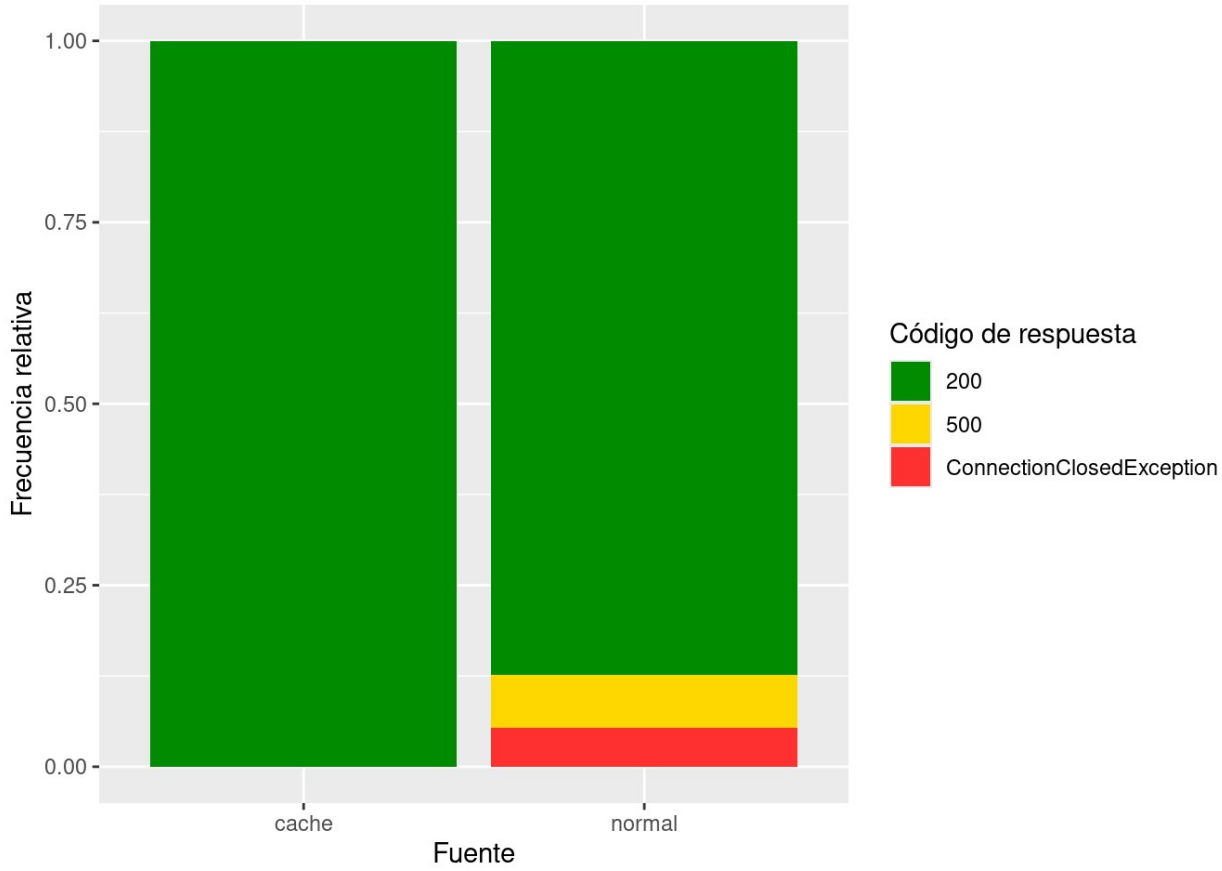
Serie de tiempo para 7rps



6.4.3.5. Nivel 5. La Figura 28 mostró diferentes respuestas en menor medida, unas fueron peticiones que sí llegaron al servidor pero este falló internamente, otros errores en la asignación de direcciones IP. Las peticiones con respuesta "ConnectionClosedException" fueron causadas por un agotamiento del servidor, se cerraba la conexión sin terminar de responder la petición y esto se mostraba como errores en el cliente de JMeter.

Figura 28

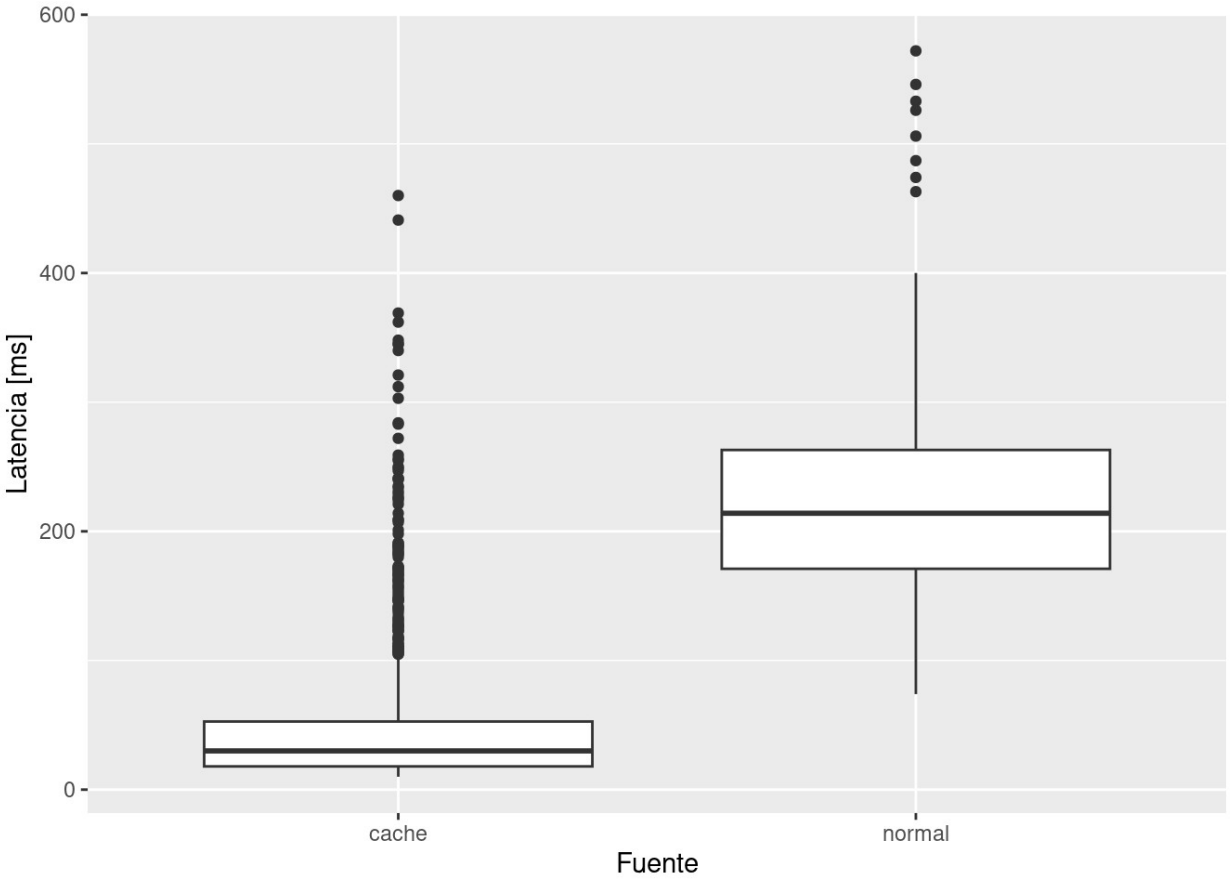
Códigos de respuesta para 10rps



Para los siguientes gráficos sólo se utilizaron las respuestas con código 200. La fuente normal tuvo un promedio de 218.64212[ms] con una desviación de 69.38765[ms], y la fuente cache tuvo un promedio de 47.91379[ms] con una desviación de 53.24756[ms]. La Figura 29 evidenció una gran diferencia entre las fuentes.

Figura 29

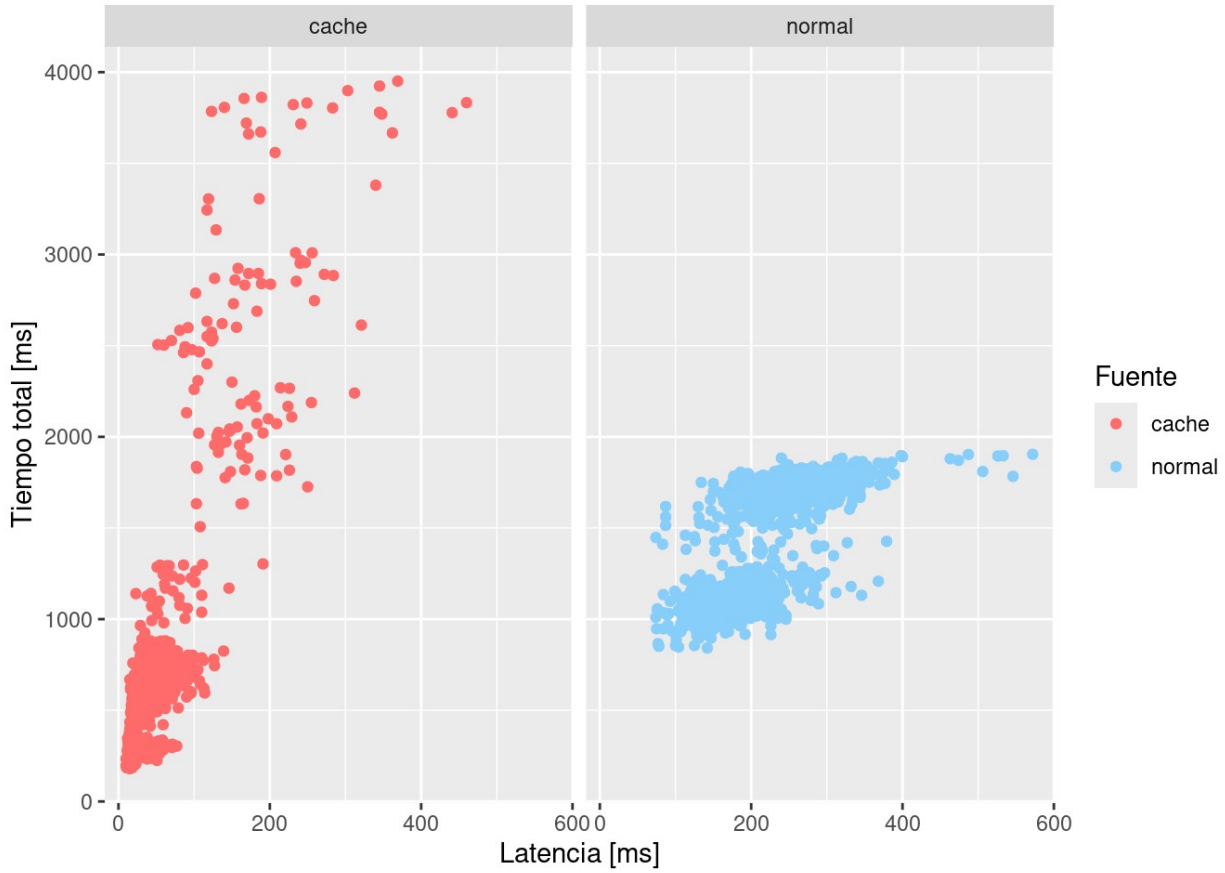
Gráfico de caja para 10rps



La dispersión, mostrada en la Figura 30, siguió la tendencia del anterior nivel, hubo mayor conformación de los nodos para cada fuente. La fuente cache siguió mostrando puntos cada vez más distantes; por otro lado, la fuente normal no tiene puntos distantes.

Figura 30

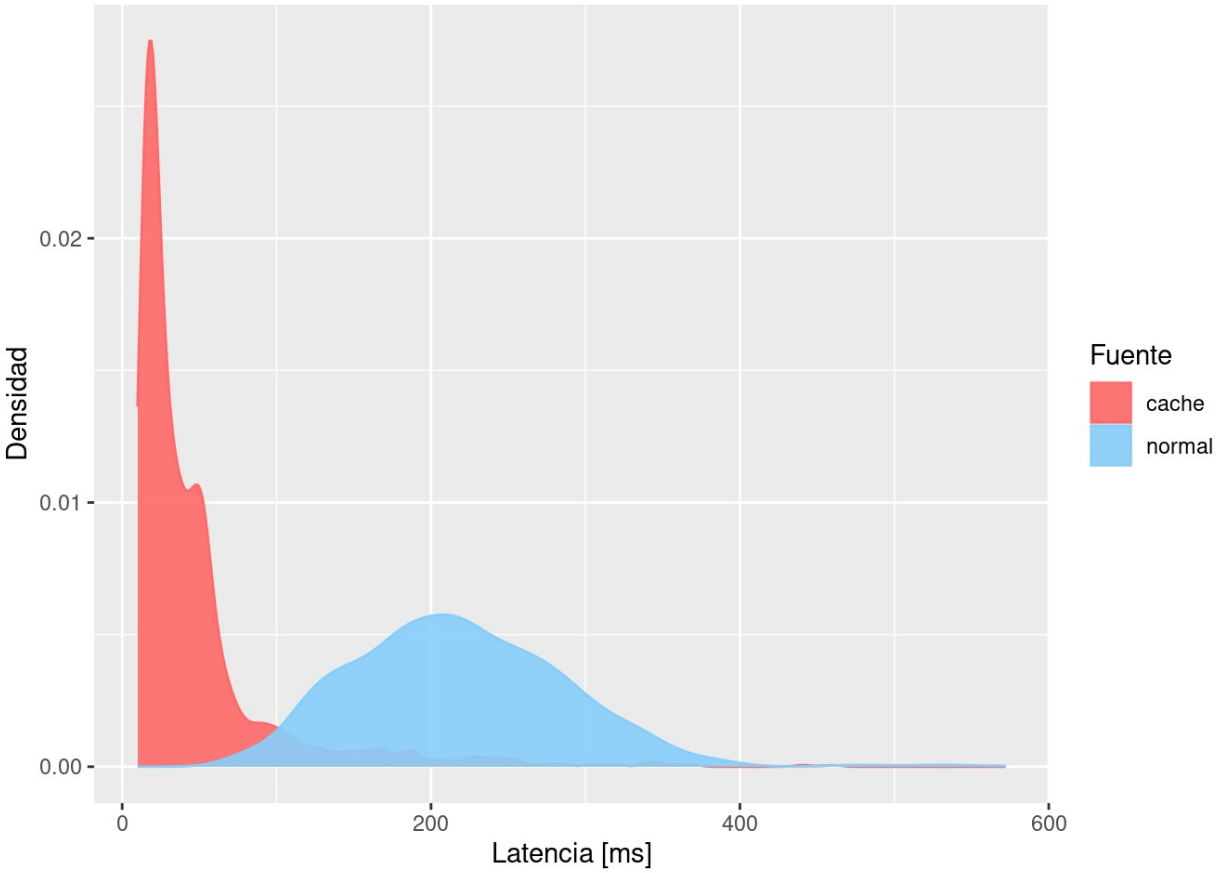
Dispersión de tiempo total contra latencia para 10rps



Las curvas de la Figura 31 mantuvieron la tendencia del anterior nivel, la fuente cache ahora sólo posee un pico prácticamente y la fuente normal siguió aumentando su desviación estándar.

Figura 31

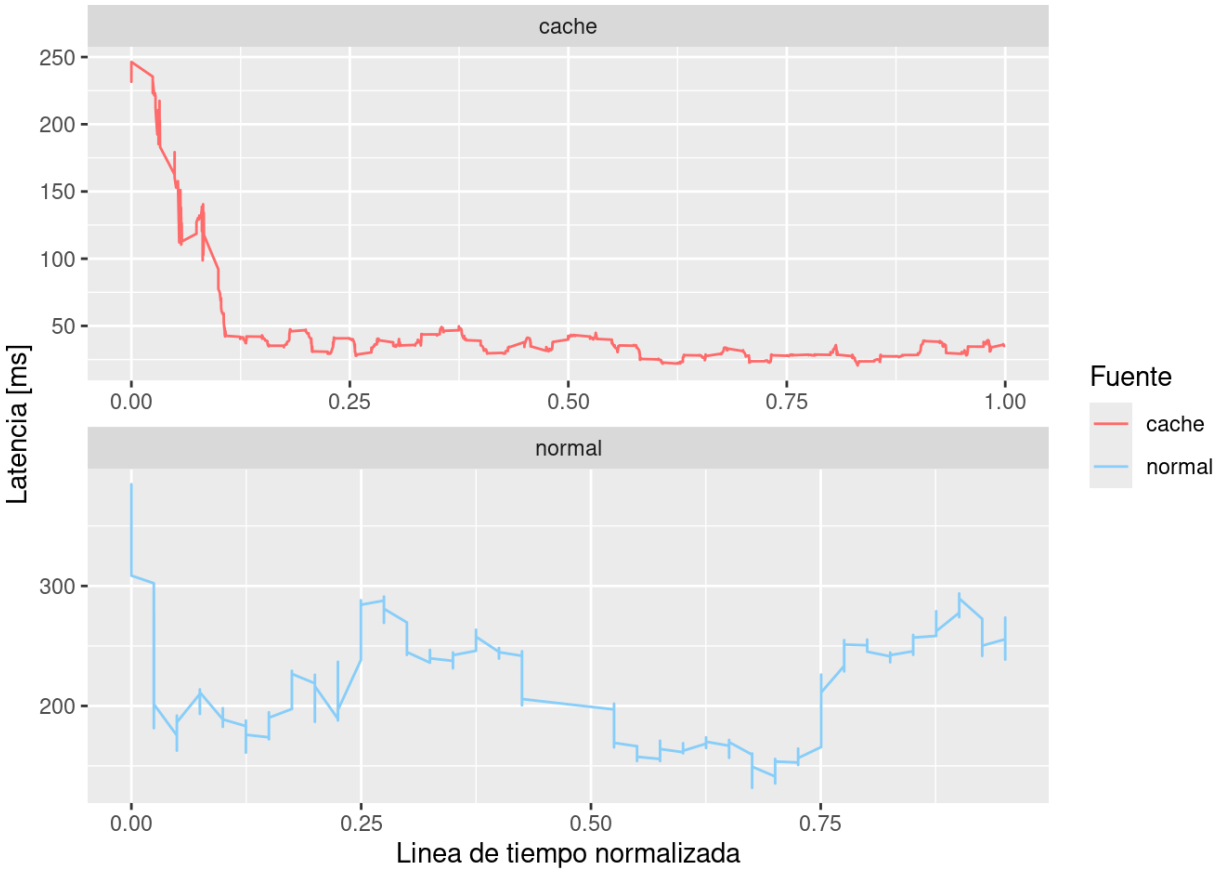
Curvas de densidad para 10rps



En la Figura 32, el comportamiento en el tiempo de la fuente cache se estabilizó a valores inferiores a los 50[ms]; mientras que, la fuente normal se mostró con oscilaciones menos agresivas pero a unos valores de latencia superiores, por ejemplo, los picos ahora están por encima de los 250[ms].

Figura 32

Serie de tiempo para 10rps



6.5. Despliegue

6.5.1. Configuración del Entorno de Producción

Se creó un repositorio en Github para alojar un ejemplo de cómo debería ser el archivo para usar en el despliegue de los contenedores.

Figura 33

Ejemplo para despliegue en contenedores

```yaml
name: coma-cache
services:
  tomcat-11:
    container_name: coma-cache-server
    build:
      dockerfile: tomcat11.Dockerfile
    ports:
      - "9999:8080"
    environment:
      MYSQL_HOST: coma-dev-mysql-5
      MYSQL_PORT: 3306
      MYSQL_USER: root
      MYSQL_PASSWORD: piolin
      REDIS_HOST: coma-cache-redis
      REDIS_PORT: 6379
    volumes:
      - type: volume
        source: tomcat-home
        target: /usr/local/tomcat
    networks:
      - net
  redis-8:
    container_name: coma-cache-redis
    image: "redis:8.0.2"
    ports:
      - "6379:6379"
    networks:
      - net

volumes:
  tomcat-home:

networks:
  net:
    name: coma-dev_net
    external: true
```

Todos los valores mostrados pueden ser cambiado para ajustarlos a la arquitectura que se está ejecutando en el momento, en específico para MySQL o para los puertos del servidor de

caché o de Redis. Si es la primera vez que se ejecuta se creará la imagen personalizada de Tomcat.

En el mismo repositorio también existen archivos de *Bash* que permiten iniciar, detener y desplegar el WAR en el servidor. Se pueden extender para agregar más cosas para el inicio o al momento que se detiene.

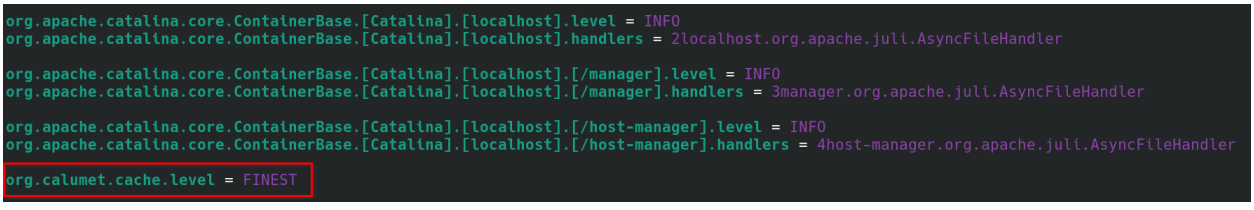
6.5.2. Monitorización

Se recomienda que para monitorizar errores o obtener mensajes más detallados de lo que está sucediendo en el servidor, se activen los registros para la aplicación al nivel más explícito "FINEST". Esto permite que en los registros de Tomcat queden los mensajes que son generados por el sistema de caché.

Una forma de activar los registros para la aplicación es agregar lo que se muestra resaltado en rojo en la Figura 34, en los archivos de configuración. Lo primero que se hace es entrar a la carpeta "conf" que tiene el servidor Tomcat, dentro de esa carpeta se debe buscar el archivo "logging.properties". Si el archivo no existe se debe crear. Dentro de ese archivo se debe agregar el manejador para el registro en el contexto de la aplicación, como se muestra en la siguiente figura:

Figura 34

Configuración para habilitar el mensajes de la aplicación



7. Conclusiones

La exploración técnica de la plataforma COMA y sus desafíos actuales permitieron identificar necesidades y nuevos puntos de mejora en cuanto al rendimiento de la plataforma. Este proyecto también marca un precedente en cuánto a la estrategias y el análisis que se puede hacer para integrar nuevas tecnologías a la plataforma sin comprometer la base de código existente.

La evaluación de carga progresiva se convirtió en un elemento valioso en dos formas: En la primera forma, permitió evaluar la adaptación de la infraestructura y las características de su respuesta; en la segunda forma, para determinar el punto en el que el sistema normal empezaba a tener fallos y anticipar las causas para que no se presenten cuando se requiera tener más escalabilidad.

El sistema de caché implementado logró evidenciar que para cargas pequeñas existe una degradación en cuanto a la estabilidad y el rendimiento base. Esto debido a la sobrecarga generada por realizar peticiones a un servicio externo; además, el sistema de caché también llega a ejercer presión sobre MySQL cuando hay que traerse datos por primera vez o en una sincronización. Para cargas más grandes, como se pudo ver desde los 5rps, los tiempos de respuesta siempre presentan mejores resultados con respecto al sistema normal.

La plataforma COMA junto con el sistema de caché logró mejorar los tiempos de respuesta sin agregar complejidad de administración del almacenamiento en caché. Esto permite al grupo CALUMET aplicar bajo demanda el uso de caché para ciertos módulos, sin comprometerlos funcionalmente y sin estar atados a tecnologías específicas que podrían afectar la estabilidad de toda la plataforma. La independencia y modularidad hacen posible llevar el

desarrollo paralelo y coordinar esfuerzos para agregar nuevas funcionalidades.

Referencias bibliográficas

Amazon Web Services. (s. f.). *What is a Database Management System (DBMS)?* Amazon Web Services, Inc. https://aws.amazon.com/what-is/dbms/

Apache Software Foundation [ASF]. (s. f.). *Apache Tomcat*®. https://tomcat.apache.org/

Commonhaus Foundation [CF]. (s. f.). *About WildFly*. WildFly. https://www.wildfly.org/about/

Deitel, P., & Deitel, H. (2016). *Java: cómo programar* (A. V. Romero Elizondo, Trans.; 10th ed.). Pearson Educación.

Eclipse Foundation. (s. f.). *About*. Eclipse Glassfish. Recuperado 14 de enero de 2026, de https://glassfish.org/

Kofler, M. (2005). *The Definitive Guide to MySQL 5* (D. Kramer, Trans.; 3rd ed.). Apress.

MySQL. (2024, Abril 30). *Chapter 1 General Information*. MySQL 8.4 Reference Manual. Retrieved Diciembre 4, 2024, from https://dev.mysql.com/doc/refman/8.4/en/introduction.html

Oracle. (2022, Junio 6). *What is Java and why do I need it?* Java. Retrieved Diciembre 4, 2024, from https://www.java.com/en/download/help/whatis_java.html

SEBoK. (2025, 20 octubre). *System Reliability, Availability, and Maintainability*. SEBoK. https://sebokwiki.org/wiki/System_Reliability,_Availability,_and_Maintainability

Sommerville, I., & Velázquez, S. F. (2011). *Ingeniería de software*.

Suehring, S. (2021). *Redis For Dummies* (2nd ed.). John Wiley & Sons, Inc.

Tessier, J. (2025, 13 junio). *Why your caching strategies might be holding you back (and what to*

consider next). Redis. https://redis.io/blog/why-your-caching-strategies-might-be-holding-you-back-and-what-to-consider-next/

Undertow home (De Red Hat). (s. f.). Undertow. Recuperado 14 de enero de 2026, de https://undertow.io/

Apéndices

Apéndice A. Sincronizador de la información en caché con la residente en el servidor MySQL.

Identificador	RF1	**Nombre**	Sincronizador de la información en caché con la residente en el servidor MySQL.
Tipo	Necesario	**Requerimiento que lo utiliza o especializa**	Utilizado por: RF2
Entrada	Nombre de las tablas. Filtros o limitaciones sobre los datos que se requieren.	**Salida**	Datos sincronizados en estructuras del servidor de caché.
Descripción			
Precondiciones	El sistema dispara periódica y automáticamente la acción según configuración. El administrador requiere la sincronización manual.		
Poscondiciones	El sistema reporta las estructuras sincronizadas y los cambios.		
Manejo de situaciones anormales			
Si la sincronización falla en alguna de las tablas se reportará en los archivos de registros y se intentará con el resto.			
Criterios de aceptación			
El sincronizador actualiza la información y es capaz de generar el reporte con los detalles de la sincronización.			

Apéndice B. Registro de nuevas consultas para almacenar en la caché

Identificador	RF2	**Nombre**	Registro de nuevas consultas para almacenar en la caché.
Tipo	Deseable	**Requerimiento que lo utiliza o especializa**	
Entrada	HASH de la consulta	**Salida**	Registro de los resultados de la consulta.
Descripción			
Precondiciones	El hash suministrado debe ser de la consulta parametrizada.		
Poscondiciones	Los datos necesarios, así como las tablas, empezarán a ser sincronizados.		
Manejo de situaciones anormales			
Si los datos para dar respuesta a la consulta no pueden ser traducidos en estructuras del servidor de caché, se reportará la falla y no se pondrá a disponibilidad la consulta.			
Criterios de aceptación			
El registro permite que nuevas consultas que no se han cargado puedan ser realizadas desde la caché. El servidor creará las estructuras necesarias para mantener la información consistente y disponible.			

Apéndice C. Permitir consultas con o sin filtros

Identificador	RF3	**Nombre**	Permitir consultas con o sin filtros.
Tipo	Necesario	**Requerimiento que lo utiliza o especializa**	Especializado por: RF4
Entrada	El hash de la consulta parametrizada. Argumentos para la consulta.	**Salida**	Resultado con los datos.
Descripción			
Precondiciones	El hash de la consulta debe estar registrado. La cantidad de argumentos debe ser la mínima a la esperada por la consulta.		
Poscondiciones	Se actualizará la fecha de última vez de uso del hash de la consulta.		
Manejo de situaciones anormales			
Si la cantidad de argumentos es insuficiente, se reportará la inconsistencia; mientras que, si la cantidad es mayor se reportará una advertencia y se descartarán los excedentes.			
Criterios de aceptación			
La consultas retornan los mismos resultados tanto en el servidor de caché como en MySQL, siempre y cuando la ventana de actualización esté vigente.			

Apéndice D. Permitir consultas combinadas

Identificador	RF4	**Nombre**	Permitir consultas combinadas.
Tipo	Deseable	**Requerimiento que lo utiliza o especializa**	
Entrada	Nombre de las tablas. Filtros sobre cada una.	**Salida**	Resultado con los datos.
Descripción			
Precondiciones	Las tablas deben estar registradas. Los filtros deben ser aplicables a las tablas individuales.		
Poscondiciones	Se actualizará la fecha de última vez de uso del hash de la consulta.		
Manejo de situaciones anormales			
Si la cantidad de argumentos es insuficiente, se reportará la inconsistencia; mientras que, si la cantidad es mayor se reportará una advertencia y se descartarán los excedentes.			
Criterios de aceptación			
La consultas retornan los mismos resultados tanto en el servidor de caché como en MySQL, siempre y cuando la ventana de actualización esté vigente.			

Apéndice E. Limpieza de datos que ya no son tan usados

Identificador	RF5	**Nombre**	Limpieza de datos que ya no son tan usados.
Tipo	Deseable	**Requerimiento que lo utiliza o especializa**	Utilizado por: RF1
Entrada	Fecha de último uso de los hash de las consultas que están asociados a los datos.	**Salida**	Eliminación de los datos asociados que dan respuestas a los hash menos usados.
Descripción			
Precondiciones	Los hash no deben estar asociados a datos de nivel crítico, es decir, datos clasificados manualmente como de alta importancia para el funcionamiento de la plataforma.		
Poscondiciones	Se retiran los hash de la lista de disponibles.		
Manejo de situaciones anormales			
Las fechas de último uso de los hash son muy cercanas entre sí, en ese caso, no se eliminará ninguno.			
Criterios de aceptación			
Se eliminan los hashes de las consultas junto con los datos asociados, sí y sólo sí, las fechas de último uso son significativamente más antigüas que las demás.			

Apéndice F. Monitor de estado del sistema

Identificador	RF6	**Nombre**	Monitor de estado del sistema.
Tipo	Deseable	**Requerimiento que lo utiliza o especializa**	
Entrada		**Salida**	Estadísticas de uso del servidor de caché y del hardware.
Descripción			
Precondiciones	El sistema debe estar operando.		
Poscondiciones	El sistema sigue su operación sin alteración.		
Manejo de situaciones anormales			
El sistema reporta estadísticas incompletas o erradas, en ese caso, el sistema se considera en estado de falla crítica.			
Criterios de aceptación			
El reporte generado contiene estadísticas completas del estado del sistema, sin retraso significativo.			