Sistema de experimentación MiMoQ para evaluar Atributos de Calidad en Microservicios

MicroQ

ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

Mayo, 20 de 2024

Versión 2.0

Mauren Natalia Rivera
Angie Katherine Castro
Kevin Leonardo Lopez
Anderson Jair Alvarado

HISTORIAL DE CAMBIOS

Versión	Fecha	Sección del documento modificada	Descripción de cambios (corta)	Responsable (S)
1.0.0	15/11/2023	Versión inicial del documento	Descripción inicial de todos los elementos del documento.	Equipo MicroQ.
1.0.1	27/03/2024	Refinamiento general del documento.	Refinamiento de especicación de requerimientos.	Equipo MicroQ.
2.0.0	20/05/2024	Versión final del documento.	Refinamiento final de los elementos.	Equipo MicroQ.

Tabla 1: Historial de cambios

Contenido

Lista de Tablas

1. Introducción

1.1 Propósito

El presente documento tiene como objetivo principal proporcionar una guía clara y detallada para el desarrollo exitoso de un sistema de software. En esta sección, se expone la intención con la que se realiza este documento, destacando las razones fundamentales que hacen que sea crucial. Además, se identifica el producto de software al cual se le van a especificar los requerimientos, se define la audiencia interesada en el contenido y se delimita el alcance del documento, especificando el alcance del producto que será descrito.

Este documento abarca la especificación de requerimientos para todo el Sistema de Experimentación MiMoQ. para evaluar Atributos de Calidad en Microservicios. Esto significa que se describirán los requerimientos para todos los aspectos del sistema, incluyendo sus funciones, características, módulos, interfaces, desempeño y cualquier otro aspecto relevante. El alcance de este documento no se limita a un subsistema o componente específico, sino que se extiende a la totalidad del producto de software.

La audiencia interesada en este documento es diversa y abarca varios grupos clave:

- **Equipo de Desarrollo de Software:** Este equipo incluye a los ingenieros, diseñadores y programadores encargados de la creación y el desarrollo del Sistema de Experimentación. Deben comprender los requerimientos detallados aquí para implementar con éxito el sistema.
- **Stakeholders del Proyecto:** Los inversores, patrocinadores, usuarios finales y cualquier entidad interesada en el éxito del proyecto son parte de la audiencia. Este documento proporciona una visión clara de lo que se espera del Sistema de Experimentación y ayuda a alinear las expectativas de todas las partes involucradas.
- **Usuarios Finales del Sistema:** Los usuarios finales que interactuarán con el Sistema de Experimentación necesitan comprender cómo funcionará el sistema y qué características se implementarán.

El presente documento de requerimientos del Sistema de Experimentación es esencial por varias razones:

- **Comunicación Efectiva:** Facilita una comunicación clara y efectiva entre todas las partes interesadas, lo que ayuda a evitar malentendidos y asegura que todos compartan una comprensión común del sistema.
- **Calidad y Consistencia:** Proporciona una base sólida para garantizar la calidad y la consistencia en el desarrollo del Sistema de Experimentación. Los requerimientos precisos son la piedra angular de un producto exitoso.
- **Gestión del Alcance:** Delimita el alcance del proyecto, lo que es fundamental para evitar desviaciones, mantenerse dentro de los límites de tiempo y presupuesto, y garantizar la satisfacción del cliente.
- **Éxito del Proyecto:** El cumplimiento de los requerimientos especificados en este documento es esencial para el éxito del proyecto de experimentación. El Sistema

de Experimentación debe cumplir con las expectativas de los usuarios y lograr sus objetivos.

1.2 Alcance

El sistema de experimentación, denominado "Sistema de Experimentación MiMoQ de Atributos de Calidad para Aplicaciones basadas en Microservicios", está diseñado para ser utilizado por analistas de rendimiento y expertos en calidad de software. Su propósito es automatizar la realización de experimentos para medir los atributos de calidad de aplicaciones que utilizan la arquitectura de microservicios desplegadas en contenedores orquestados por Kubernetes.

El sistema de experimentación permitirá a los usuarios realizar las siguientes funciones:

1. **Selección de Microservicios:** Los usuarios pueden cargar o subir los microservicios específicos que desean evaluar. Esto permite centrarse en áreas particulares de una aplicación basada en microservicios y medir su calidad de manera aislada.
2. **Selección de Atributos de Calidad y Métricas:** Los usuarios pueden seleccionar los atributos de calidad que desean evaluar, como la disponibilidad, el desempeño y la elasticidad. Además, pueden especificar las métricas exactas que desean medir para cada atributo. Esto proporciona flexibilidad y personalización en la evaluación de la calidad.
3. **Configuración del Período de Medición:** Los usuarios pueden configurar la duración de los experimentos, lo que les permite definir cuánto tiempo se realizará la medición de los atributos de calidad. Esto es útil para realizar evaluaciones a corto o largo plazo según las necesidades del usuario.
4. **Establecimiento de Parámetros de Carga:** El sistema permite a los usuarios definir la intensidad de la carga de trabajo de cada microservicio, como el número de usuarios generando peticiones, y la estrategia de despliegue. Esto les permite simular diferentes escenarios de uso y evaluar el comportamiento de los microservicios en condiciones variadas.
5. **Almacenamiento y Recuperación de Datos:** El sistema organiza y almacena los datos de los experimentos de manera eficiente, lo que facilita la recuperación de resultados y la comparación de mediciones a lo largo del tiempo. Esto es fundamental para un análisis en profundidad y la identificación de tendencias.
6. **Creación de Usuarios:** El sistema permite la creación de usuarios con roles diferenciados: "administrador" y "experimentador". Los usuarios experimentadores pueden crear experimentos seleccionando las métricas a evaluar, la intensidad de carga de cada microservicio. A diferencia de los usuarios experimentadores, los usuarios administradores pueden gestionar los usuarios registrados en el sistema.
7. **Autenticación de Usuarios:** Para garantizar la seguridad y la trazabilidad, el sistema incluye un sistema de autenticación para los usuarios. Esto garantiza que solo personas autorizadas puedan acceder y utilizar el sistema.

El sistema tiene como objetivo simplificar y agilizar el proceso de medición de atributos de calidad en aplicaciones basadas en microservicios. Al automatizar tareas como la generación de carga, el monitoreo, la recopilación de métricas y la configuración de experimentos, se busca ahorrar tiempo y recursos. Esto permite a los usuarios identificar problemas de rendimiento, cuellos de botella y deficiencias en la calidad de las aplicaciones de manera más eficiente. La flexibilidad del sistema les permite personalizar sus experimentos según sus necesidades específicas. Además, contribuye al desarrollo de soluciones más sólidas y optimizadas en el contexto de microservicios. En última instancia, el sistema de experimentación es una herramienta valiosa para mejorar la calidad y el rendimiento de las aplicaciones en un entorno empresarial altamente competitivo.

1.3 Definiciones, Acrónimos, y Abreviaciones

Acrónimos:

API	Aplication Programming Interface
CRUD	Create, Retrieve, Update, Delete
DBMS	DataBase Management System
NVM	Node Version Manager
NPM	Node Package Manager
K8S	Kubernetes
REST	Representational State Transfer
S.O.	Sistema Operativo
JS	JavaScript
SDD	Software Design Description
SQL	Structured Query Language
SRS	Software Requirement Specification

Tabla 2: Acrónimos

1.4 Apreciación Global

Esta sección proporciona una visión general del contenido y la organización del presente documento de SRS para el Sistema de Experimentación MiMoQ. Su propósito es guiar al lector a través de la estructura y el alcance de la información que encontrará en las secciones subsiguientes.

Estructura del Documento

El SRS se ha dividido en secciones claramente definidas, cada una de las cuales aborda aspectos específicos relacionados con el Sistema de Experimentación MiMoQ. A continuación, se presenta un resumen de la estructura del documento:

1. **Propósito (Sección 1.1):** Esta sección introduce el propósito del documento, identifica el producto de software en cuestión, la audiencia interesada y delimita el alcance de este.
2. **Alcance (Sección 1.2):** Aquí se describe el alcance del producto de software, incluyendo su nombre, funcionalidades, utilidad y su relación con las metas corporativas o la estrategia de negocio.
3. **Definiciones, Acrónimos y Abreviaciones (Sección 1.3):** Esta sección proporciona definiciones de términos clave y acrónimos utilizados en el documento para garantizar una comprensión óptima.
4. **Apreciación Global (Sección 1.4):** La sección actual en la que se encuentra proporciona una visión general de la estructura y el contenido del SRS.

Organización del Contenido

El contenido del SRS se presenta de manera lógica y secuencial para facilitar la comprensión y navegación del lector. Cada sección posterior abordará los detalles específicos relacionados con el Sistema de Experimentación MiMoQ. Estas secciones incluyen:

- **Requerimientos Funcionales (Sección 2):** Describe las funciones y operaciones del sistema.
- **Requerimientos No Funcionales (Sección 3):** Especifica los requisitos de rendimiento, seguridad, usabilidad y otros aspectos no funcionales del sistema.
- **Requerimientos de Interfaces (Sección 4):** Detalla las interfaces con otros sistemas o componentes.
- **Requerimientos de Datos (Sección 5):** Define la estructura y manipulación de datos.
- **Otros Requerimientos (Sección 6):** Incluye requisitos adicionales, como regulaciones legales, consideraciones de privacidad, y otros aspectos relevantes.
- **Anexos (Sección 7):** Contiene información adicional, como diagramas, tablas, o detalles técnicos.

Importancia del Documento

El SRS es esencial para el éxito del proyecto de desarrollo del Sistema de Experimentación MiMoQ. Facilita la comunicación efectiva entre todas las partes interesadas, asegura la calidad y consistencia en el desarrollo, y ayuda en la gestión del alcance del proyecto. Cumplir con los requerimientos especificados en este documento es fundamental para lograr los objetivos del proyecto.

Esta sección de "Apreciación Global" proporciona una visión general para que el lector pueda navegar por el documento de manera eficiente y comprender su estructura y relevancia en el contexto del proyecto.

2. Descripción Global

En esta sección se describe de manera general el Sistema de experimentación MiMoQ, abarcando los aspectos generales y sus requisitos. Es fundamental destacar que esta sección no detalla formalmente los requisitos, en su lugar, proporciona información contextual que ofrece a los lectores una descripción completa del sistema. El lenguaje utilizado aquí es orientado al usuario, lo que facilita la comprensión para diversos interesados en el proyecto.

2.1 Perspectiva del Producto

El sistema de experimentación, MiMoQ surge de la necesidad de evaluar atributos de calidad en aplicaciones con arquitectura de microservicios. Actualmente se están llevando a cabo otros proyectos que se enfocan en el estudio de métricas asociadas a Microservicios, entrenamiento de modelos de IA con datos de estas métricas y exploración de diferentes maneras de despliegue de los microservicios. Por lo anterior, uno de los requerimientos que debe satisfacer MiMoQ es la mantenibilidad para poder integrar nuevas métricas a evaluar en el sistema y agregar nuevas opciones de despliegue. Sin embargo, MiMoQ no se puede describir como un producto que pertenece a una familia ya que es un sistema independiente que se puede integrar con otros productos enfocados en el estudio de diferentes aspectos de los microservicios.

2.1.1 Interfaces con el usuario

Teclado: Por esta interfaz el usuario ingresará información como credenciales de acceso, datos para registrarse y parámetros de los experimentos.

Mouse: Con esta interfaz el usuario podrá navegar fácilmente al seleccionar algunos parámetros con valores posibles preestablecidos, seleccionar las diferentes opciones del sistema y ejecutar acciones sobre los experimentos (envío de parámetros y cancelación de experimentos).

Pantalla: Por esta interfaz el usuario podrá observar el sistema. Se espera crear una aplicación responsive por lo que aún no se ha establecido una resolución máxima.

Interfaz GUI: La interfaz de usuario será desarrollada en JavaScript usando el framework NextJS. Se busca que la interfaz de usuario sea fácil de comprender y su diseño sea el adecuado para presentar las diferentes características del sistema.

Tarjeta de red: Para poder instalar y ejecutar el sistema es necesario que se cuenta con una tarjeta de red ya sea inalámbrica o Ethernet.

2.1.2 Interfaces con el Hardware

Es importante señalar que Kubernetes es compatible con entornos heterogéneos y puede administrar clústeres que incluyen nodos con diferentes sistemas operativos, como Windows y Linux. La configuración específica de los protocolos y puertos dependerá de la arquitectura y la configuración exacta del entorno Kubernetes.

- **Protocolo de comunicación TCP/IP:** Kubernetes utiliza el protocolo de red TCP/IP para la comunicación entre sus componentes distribuidos. Este protocolo es elegido debido a su confiabilidad y capacidad de proporcionar una conexión orientada a la conexión. En entornos mixtos de Windows y Linux, el uso de TCP/IP es crucial ya que es un estándar ampliamente aceptado y compatible con ambos sistemas operativos.[1]

3. **Puertos de red TCP:** Kubernetes utiliza varios puertos de red para la comunicación entre sus componentes. Por ejemplo, el puerto 6443 se utiliza comúnmente para la comunicación segura con el servidor de control de Kubernetes (API Server). Además, otros puertos, como 2379 y 2380, se utilizan para la comunicación entre nodos del clúster (etcd). Al especificar puertos de red TCP, es posible configurar firewalls en entornos de ejecución mixtos para permitir el tráfico necesario. Por ejemplo, se puede asignar el puerto TCP 6443 para la comunicación segura en un entorno con Windows y Linux.[1]

2.1.3 Interfaces con el Software

A continuación, se presentan los productos de software que se integran en nuestro sistema para facilitar la comunicación entre componentes y garantizar la compatibilidad en diferentes entornos de ejecución.

Producto de Software	Docker	PostgreSQL	NodeJS
Descripción	Docker es un software open source que se usa para desplegar aplicaciones en contenedores, siendo un contenedor un paquete de software que contiene todas las dependencias y librerías necesarias para ejecutar una aplicación, lo que, a diferencia de una máquina virtual, lo hace un componente liviano.[2]	Este sistema de gestión de bases de datos soporta la realización de transacciones en simultánea, multiusuario y multihilo; además, cuenta con la GNU GPL. [3]	Node.js es un entorno en tiempo de ejecución multiplataforma para la capa del servidor basado en JavaScript. Node.js utiliza el motor de JavaScript V8 de Google Chrome para compilar el código JavaScript en código de máquina. [5]
Propósito de Uso	Asegurar que la mayoría de los usuarios finales puedan utilizar el producto sin inconvenientes, pues usando Docker solo se requiere ejecutar un par de comandos para desplegar la aplicación. Docker es compatible con Windows y Linux	Utilizar una base de datos segura que soporte múltiples consultas, proporcione una capacidad de respuesta y almacenamiento eficiente y, además, que asegure la integridad de la información.	La interfaz de usuario se desarrollará con el framework Next.JS, este framework usa el lenguaje JavaScript. JavaScript no es un lenguaje de bajo nivel o de máquina, es por esto que se necesita un intérprete para ejecutar las instrucciones correctamente. El intérprete

Tabla 3:

	siendo estos sistemas operativos los más usados actualmente.[6][7]		es un programa que lee el código fuente escrito en JavaScript y lo convierte en un formato que la computadora puede entender y ejecutar, es por esto que NodeJS se hace necesario para ejecutar el frontend de nuestra aplicación.
Versión	Docker Engine 24.0	Versión 4.1.18	Versión LTS: 20.9.0
Fuente	*Docker Docs* Documentación: https://docs.docker.com/get-docker/ *Install Docker Desktop on Linux | Docker Docs*	*PostgreSQL.* Documentación: https://www.postgresql.org/	*Introducción a Node.js* Documentación: *https://nodejs.org/en/download*
Comentarios Adicionales	La aplicación podría funcionar en otros sistemas operativos que soporten el JRE versión 1.6.[2]	La base de datos se desplegó en render para un trabajo más eficiente entre frotend y backend. Esta configuración también se puede hacer localmente.	Una gran ventaja de NodeJS es la gran portabilidad que tiene ya que se puede ejecutar en Sistemas operativos UNIX, Mac OS y Windows, sin embargo, es necesario tener instalado Python 3.

Interfaces de software

Producto de Software	Kubernetes	Prometheus	Grafana	Windows	Linux
Descripción	Kubernetes, también conocido como K8s, es una herramienta de orquestación de contenedores. Agrupa los contenedores que componen una aplicación en unidades lógicas para facilitar la administración y la detección. Además, el despliegue y escalado es automático.[8]	Prometheus es una herramienta de monitoreo, es open source. Recopila y almacena las métricas obtenidas como datos temporales, es decir, se almacena con la marca de tiempo en la que se hizo su registro. Usa un lenguaje de consulta llamado PromQL.[9]	Grafana es una herramienta que permite visualizar datos de series temporales, lo que la hace fácilmente integrable con Prometheus. Es open source bajo la licencia Apache 2.0.[10][11]	La aplicación será desarrollada para que se pueda ejecutar en la plataforma Windows que es una familia de sistemas operativos basados en una interfaz gráfica de usuario, además, es el sistema operativo más usado actualmente.[12]	La aplicación será desarrollada en el sistema operativo Linux que es una serie de sistemas operativos de tipo UNIX, la mayoría son gratuitos y se distribuyen bajo la licencia GNU GPL.[13]
Propósito de Uso	Asegurar un despliegue sencillo de la aplicación en los sistemas operativos Windows y Linux, además, garantiza escalado	Con esta herramienta se busca hacer la medición de métricas de los microservicios que se van a evaluar. Al ser open source se garantiza la libertad	Grafana va a permitir observar los resultados obtenidos con Prometheus de manera grafica lo que le brinda al usuario facilidad de comparación entre	Asegurar que la mayoría de los usuarios finales puedan utilizar el producto sin inconvenientes, pues este sistema operativo es el más difundido y usado	Linux ofrece facilidad para desarrollar aplicaciones al poder manejarse a través de su línea de comandos y contar con una gran comunidad. Además, por las herramientas

	automática y tolerancia a fallos.	de uso sin generar costos, además, cuenta con mucha documentación lo que hace que sea fácil de usar y se cuenta con gran soporte por parte de la comunidad.	los diferentes experimentos.	en la actualidad. Además, la herramienta usada para desplegar la aplicación es compatible con Windows.	que vamos a usar, se hace más fácil la construcción del sistema usando Linux al tener mayor soporte y documentación.
Versión	Kubernetes v1.24	Versión 2.48.0-rc.2 / 2023-11-02	grafana-oss:10.0.0	Versiones basadas en la tecnología NT	Ubuntu 22.04.3 LTS
Fuente	*Kubernetes Documentation : Documentación de Kubernetes | Kubernetes*	*Prometheus Docs : Overview | Prometheus Downloads: Download | Prometheus*	*Grafana Docs : Run Grafana Docker image | Grafana documentation Downloads: Download Grafana | Grafana Labs*	*Microsoft Corporation* Ayuda : http://windowshelp.microsoft.com/Windows/es-ES/default.mspx	*Linux.org* Documentación: https://www.linux.org/pages/download/
Comentarios Adicionales	El único requisito es que se tenga instalado Docker para poder realizar el despliegue correctamente usando Kubernetes.	Se usará una imagen de Docker para desplegar esta herramienta en Kubernetes.[14]		La aplicación podría funcionar en otros sistemas operativos que soporten el JRE versión 1.6.	Se utilizará leguaje SQL para realizar el CRUD de las tablas en la base de datos.

Tabla 4: Interfaces de software

2.1.5 Operaciones

Se presenta la especificación de los modos de operación del sistema MiMoQ:

2.1.5.1 Modos de operación de usuarios

El sistema de experimentación MiMoQ manejará dos modos de operación:

- **Modo administrador:** Este usuario será el que administrará todos los usuarios del sistema y modificará las funciones disponibles en la aplicación, es decir, se encargará del mantenimiento de la aplicación.
- **Modo usuario final:** El usuario final tendrá dos roles que dependerán del nivel de experiencia que tenga.

2.1.5.2. Modos de actividad e inactividad

El sistema dejará de estar en operación cuando se estén agregando o modificando las herramientas usadas para la medición de atributos de calidad y los generadores de carga, también se suspenderá su funcionamiento cuando se estén agregando nuevas métricas que podrán ser evaluadas. Se estima que el tiempo de inactividad será de máximo una semana, esto por el tiempo que se requiere para verificar la correcta integración de nuevas funcionalidades o herramientas.

2.2 Funciones del Producto

A continuación, se presentan las funciones que debe realizar el sistema de experimentación, MiMoQ, según el tipo de usuario para satisfacer los requerimientos del cliente:

- Cargar los microservicios a evaluar.
- Seleccionar los atributos de calidad y sus respectivas métricas.
- Definir el periodo de medición.
- Seleccionar herramientas o método de inyección de fallas.
- Definir ciertos factores a variar en la experimentación tales como la intensidad de la carga de trabajo y la estrategia de despliegue.
- Almacenar y recuperar los datos de forma organizada.
- Crear usuarios. Esta plataforma contará con dos roles de usuario: principiante y experto. Al rol de principiante se podrán acceder las métricas a evaluar y el sistema elegirá las herramientas pertinentes para medirlas; mientras que, en el rol de experto, el usuario podrá ingresar las métricas a evaluar, elegir la o las herramientas necesarias para esta medición según las disponibles en la plataforma.
- Autenticar usuarios.

Las funcionalidades listadas anteriormente exponen en términos generales las funcionalidades que debe brindar el sistema MiMoQ.

2.3 Características del Usuario

Características del Usuario	Descripción
Nivel de Seguridad o de Privilegios	**Usuario experimentador:** El usuario podrá ingresar al sistema con la opción de parametrizar las métricas que serán evaluadas en el experimento, podrá seleccionar las herramientas de medición proporcionadas que desea usar. **Usuario administrador:** Este usuario podrá acceder a las herramientas usadas para el monitoreo y generación de carga y modificarlas, además, podrá agregar y eliminar las métricas asociadas a los atributos de calidad existentes, también tendrá acceso al listado de usuarios del sistema y podrá gestionarlos.
Roles	**Usuario expeimentador:** es un usuario con conocimiento sobre atributos de calidad y herramientas para medirlos, se le permitirá la selección de las métricas a evaluar y el sistema elegirá las herramientas pertinentes para la medición de estas métricas. **Usuario administrador:** Este usuario se encargará de actualizar las herramientas disponibles para el monitoreo y generación de carga, además, podrá agregar nuevas métricas a evaluar asociadas a un atributo de calidad existente o también crear uno nuevo y gestionará los usuarios del sistema.
Frecuencia de Uso	Los usuarios expertos y principiantes usarán la aplicación cuando necesiten evaluar atributos de calidad de microservicios, por lo que no se tiene una frecuencia de uso aproximada del sistema de experimentación. Los usuarios administradores ingresaran al sistema para hacerle mantenimiento, también podrán usarlo para obtener métricas de otros sistemas que estén desarrollando.

Tabla 5: Descripción de las Características del Usuario

2.4 Restricciones

2.4.5 Restricciones de hardware y software

- Para poder desplegar correctamente el sistema de experimentación MiMoQ el cliente contar con las siguientes especificaciones a nivel de hardware:
- En sistemas operativos Windows para poder ejecutar Docker, Minikube y k6 se debe contar como mínimo con un equipo que tenga:
    1. Una versión de 64 bits de Windows 10 Pro, Enterprise o Education. Si se usa una versión anterior de Windows como 7, 8 u 8.1, se tendrá que actualizar a Windows 10 primero para poder instalar Docker.
    2. Al menos 2 GB de memoria RAM
    3. 4 GB de espacio libre en el disco duro para la instalación. Sin embargo, se recomienda tener 24GB libres ya que luego de su instalación se van a ejecutar contenedores que pueden requerir gran cantidad de espacio.}
    4. Además, el sistema debe tener Hyper-V habilitado, configurar un conmutador virtual y habilitar la función de extensiones de virtualización.
- Para poder desplegar el sistema en equipos con sistemas operativos Linux para poder ejecutar Docker, Minikube y k6 se requiere:
    1. Núcleo de 64 bits y soporte de CPU para la virtualización
    2. Compatible con la virtualización KVM
    3. QEMU en versión 5.2 como mínimo
    4. Entorno de escritorio de Gnome, KDE o MATE
    5. Al menos 4 GB de memoria RAM (especialmente si se instala Docker en una máquina virtual con Linux).
    6. Se recomienda tener 20GB de almacenamiento para garantizar una correcta instalación y ejecución de Docker y Kubernetes.
- Para poder desplegar correctamente el Sistema de Experimentación MiMoQ el cliente contar con las siguientes especificaciones a nivel de software:
- Docker, una herramienta de contenerización de aplicación. Se recomienda la versión 21.5.0.
- Kubernetes, una herramienta de orquestación de contenedores. Se recomienda la versión de cliente 1.28.2.
- NodeJS con versión mayor a 18.4 para tener compatibilidad con librerías y dependencias tanto del componente frontend como el componente backend.
- Cliente de NestJS para la ejecución de comandos de NestJS, framework con el que se construyó el backend del sistema.
- Cliente de Angular 17 para la ejecuci´on de comandos de Angular, framework que se usó para construir el frontend del sistema.
- k6, generador de carga usado para parametrizar cada experimento. Se recomienda usar la versión 0.50.0 de k6.
- Helm, un gestor de paquetes de Kubernetes utilizado para automatizar el despliegue de aplicaciones. A la fecha, se recomienda la ´última versión, esta es la 3.14.2.
- PostgreSQL, un sistema de administración de base de datos relacionales de código abierto usado para almacenar la información de los usuarios, proyectos, despliegues y métricas del sistema.

- PgAdmin4, es una herramienta de gestión de código abierto para Postgres que provee una interfaz gráfica que permite la creación, el mantenimiento y uso de objetos de bases de datos PostgreSQL.
- El sistema debe desarrollarse usando el lenguaje de programación TypeScript, esto ya que es un lenguaje compatible con los frameworks usados para el backend y frontend del sistema.
- HTML y CSS.

2.4.6 Restricciones de internacionalización

El Sistema MiMoQ está disponible en idioma español latinoamericano, actualmente no se ha contemplado extenderlo a más idiomas, por esto, los usuarios deben contar con buen nivel técnico de este idioma.

2.4.7 Restricciones de tecnología

- Por solicitud del cliente, la aplicación se despliega en Kubernetes, lo que sustenta las restricciones de hardware del numeral 1 de esta sección. Cabe resaltar que Kubernetes es una herramienta altamente documentada lo que garantiza buen soporte en su uso.
- Las herramientas usadas para construir el sistema de experimentaci´on son open source garantizando un uso libre y sin costos.

2.4.8 Restricciones de usuario

- El usuario debe proporcionar el o los repositorios que contienen los microservicios de la aplicación. Se presentan las siguientes opciones:
    - **Repositorio por cada microservicio:** En la raíz del repositorio debe estar el Dockerfile que permita construir la aplicación.
    - **Repositorio que contiene todos los microservicios:** En la raíz del repositorio debe estar el Docker Compose que permite construir o descargar rápidamente las imágenes del proyecto. Se presentan dos alternativas:
        - El Docker Compose contiene las imágenes y éstas deben construirse. Según el nombre del servicio, el sistema accede a la carpeta correspondiente desde la raíz del repositorio y construye la imagen. Cada carpeta debe contener su propio Dockerfile.
        - El Docker Compose contiene las imágenes en un registro público, por lo que es suficiente descargarlas para construir y desplegar la aplicación.

2.5 Modelo del Dominio

A continuación, se presenta la documentación de los elementos en el diagrama del modelo del dominio. Sin embargo, se aclara que este modelo puede cambiar a medida que se desarrolla el sistema ya que se pueden identificar nuevos elementos o se modificarán los actuales.

ID	1	***Elemento del Dominio***	Usuario
Descripción	Representa a los usuarios del sistema, con esto se va a garantizar el control de acceso a las diferentes funcionalidades del sistema**.**		
Atributos			
Nombre	***Descripción***		***Tipo de Dato***
id_usuario	Identificador único del usuario		Integer
nombre	Nombre del usuario		Varchar (100)
correo	Correo del usuario		Varchar (80)
documento	Documento de identificación del usuario		Integer
contraseña	Clave de acceso del usuario		Varchar (50)
fk_id_rol	Llave foránea del rol		Integer

Tabla 6: Modelo del Dominio - Usuario

También se va a manejar una entidad Experimento que va a permitir almacenar la información de los experimentos ejecutados para recuperarlos posteriormente.

ID	2	***Elemento del Dominio***	Rol Usuario
Descripción	Representa los roles disponibles en el sistema, en este caso, administrador y experimentador		
Atributos			
Nombre	***Descripción***		***Tipo de Dato***
id_rol	Identificador único del rol del usuario		Integer
nombre	Nombre del rol del usuario		String

Tabla 7: Modelo del Dominio – Rol Usuario

ID	3	***Elemento del Dominio***	Proyecto
Descripción	Representa a los usuarios del sistema, con esto se va a garantizar el control de acceso a las diferentes funcionalidades del sistema**.**		
Atributos			
Nombre	***Descripción***		***Tipo de Dato***
Id_proyecto	Identificador único del proyecto		Integer
nombre	Nombre del proyecto		Varchar (40)
descripcion	Descripción referente al proyecto		Varchar (250)
url_repositorio	Correo del usuario		Varchar (150)

urls_repositorios	Clave de acceso del usuario	Varchar (50)
nombres_microservicios	Nombres de cada uno de los microservicios que hacen parte de la aplicación	Varchar (50)
docker_compose	Determina si el repositorio en el que está la aplicación tiene o no Docker Compose	Boolean
dockerfile	Determina si el repositorio en el que está la aplicación tiene Dockerfile o no	Boolean
imagenes_deploy	Son las imágenes que se usan para construir los contenedores y desplegarlos en Kubernetes	Varchar (50)
puertos_imagenes	Son los puertos de cada una de las imágenes	Varchar (50)
puertos_deploy	Son los puertos a través de los cuales se exponen las aplicaciones en Kubernetes	Varchar (50)
fk_id_usuario	Llave foránea del usuario	Integer

Tabla 8. Modelo del Dominio – Proyecto

ID	4	***Elemento del Dominio***	Despliegue
Descripción	Representa los despliegues que se han hecho en el sistema.		
Atributos			
Nombre	***Descripción***		***Tipo de Dato***
id_despliegue	Identificador único del despliegue		Integer
nombre	Nombre del despliegue		Varchar (50)
cant_replicas	Cantidad de réplicas que tendrá el microservicio. Esto se ve representado en Kubernetes como la cantidad de pods por microservicio.		Integer
cant_pods	Cantidad de pods que tendrá el despliegue.		Integer
namespace	Nombre del espacio lógico en el que se creará el despliegue de la aplicación.		Varchar (50)
puerto	Puerto en el que cual se expondrá el microservicio en los servicios de Kubernetes.		Integer
nombre_helm	Nombre que utilizará Helm para identificar los recursos de Kubernetes que se crean con relación a dicho nombre.		Varchar (50)

fk_id_proyecto	Llave foránea del proyecto.	Integer

Tabla 9. Modelo del Dominio – Despliegue

ID	5	***Elemento del Dominio***	Experimento
Descripción	Representa a los experimentos del sistema que se realizan sobre aplicaciones que se encuentran desplegadas sobre Kubernetes.		
Atributos			
Nombre	***Descripción***		***Tipo de Dato***
id_experimento	Identificador único del experimento		Integer
duracion	Duración del experimento.		Varchar (5)
cant_replicas	Cantidad de réplicas del experimento. Hace referencia a la cantidad de veces que se repetirá el experimento en su totalidad. En otros documentos se le llama “repeticiones”		Integer
endpoints	Son los endpoints de cada microservicio a los cuales se les generará carga.		Varchar (200)
fk_id_carga	Llave foránea de la carga.		Integer

Tabla 10. Modelo del Dominio – Experimento

ID	6	***Elemento del Dominio***	Carga
Descripción	Representa la carga que se generará en los microservicios.		
Atributos			
Nombre	***Descripción***		***Tipo de Dato***
id_carga	Identificador único de la carga.		Integer
cant_usuarios	Cantidad de usuarios que simulan interacción en el endpoint del microservicio especificado.		Varchar (50)
duracion	Es el tiempo que se estará generando carga en el endpoint del microservicio especificado.		Varchar (50)

Tabla 11. Modelo del Dominio – Carga

ID	7	***Elemento del Dominio***	Atributo
Descripción	Representa a los atributos que se tendrán en cuenta en el sistema. Estos se definieron a partir de la ISO 25010 del 2011		
Atributos			
Nombre	***Descripción***		***Tipo de Dato***
id_atributo	Identificador único del atributo		Integer

nombre	Nombre del atributo	Varchar (50)
descripcion	Descripción de lo que significa el atributo	Varchar (200)

Tabla 12. Modelo del Dominio – Atributo

ID	8	***Elemento del Dominio***	Subatributo
Descripción	Representa a los subatributos que se tendrán en cuenta en el sistema. Estos se definieron a partir de la ISO 25010 del 2011		
Atributos			
Nombre	***Descripción***		***Tipo de Dato***
id_subatributo	Identificador único del subatributo		Integer
nombre	Nombre del subatributo		Varchar (50)
descripcion	Descripción de lo que significa el subatributo		Varchar (200)
fk_id_atributo	Llave foránea del atributo		Integer

Tabla 13. Modelo del Dominio – Sub atributo

ID	9	***Elemento del Dominio***	Métrica
Descripción	Representa a las métricas que se tendrán en cuenta en el sistema. Estos se definieron a partir de la ISO 25010 del 2011		
Atributos			
Nombre	***Descripción***		***Tipo de Dato***
id_metrica	Identificador único de la métrica		Integer
nombre	Nombre de la métrica		Varchar (50)
formula	Formula que con la que se cálcula la métrica		Varchar (200)
descripcion	Descripción de lo que significa la métrica		Varchar (500)
fk_id_subatributo	Llave foránea del subatributo		Integer

Tabla 14. Modelo del Dominio – Métrica

2.6 Suposiciones y Dependencias

Para el uso del sistema MiMoQ se asume que el usuario cumple con los siguientes requerimientos:

- Para tener un buen funcionamiento del sistema se asume que el computador donde el cliente va a ejecutar el sistema cumple con las restricciones de hardware y software especificadas en la sección 2.4 Restricciones dependiendo del sistema operativo donde se quiera ejecutar la aplicación.
- Se espera que el cliente tenga instalado Docker y Kubernetes y que se guie por el paso a paso que se entregará para instalar la aplicación correctamente.

- En la instalación y ejecución de la aplicación se requiere conexión a internet para garantizar la correcta descargado de los paquetes de software necesarios.
- En esta sección se describen con más detalle los requerimientos funcionales y no funcionales con lo que debe cumplir el sistema de Experimentación MiMoQ para garantizar la satisfacción de todas las necesidades del cliente.

3. Requerimientos específicos

Especificación de requerimientos funcionales

- Crear usuarios. Esta plataforma contará con dos roles de usuario: experimentador y administrador.
- Actualizar usuarios. El usuario administrador podrá actualizar los datos de los usuarios.
- Consultar usuarios. El usuario administrador podrá ver y consultar los datos de los usuarios registrados en la plataforma.
- Eliminar usuarios. El usuario administrador podrá eliminar los usuarios registrados en la plataforma.
- Autenticar usuarios.
- Crear proyectos en los cuáles se realiza la carga de la o las URLs de los repositorios donde se encuentra el código fuente de los microservicios a evaluar.
- Crear despliegues asociados a un proyecto seleccionando los microservicios y la cantidad de réplicas para cada uno. Definir experimentos, lo cual requiere de:
    1. SUT (sistema bajo estudio): seleccionar los microservicios que serán evaluados (no necesariamente serán todos los microservicios que conforman la aplicación).
    2. Duración del experimento.
    3. Número de repeticiones del experimento
    4. Entradas del experimento: Definir la carga de trabajo
    5. Salidas del experimento: a) Definir los atributos de calidad y las métricas deseadas. b) Método de salida: ¿por la BD y por los dashboards? (algo de esto leí en la descripción de Grafana)
    6. Almacenar y recuperar los datos de forma organizada.
    7. Visualizar los resultados experimentales en paneles de control.
        - Factores a variar durante el experimento: en esta primera versión de MiMoQ se permitirá variar: 1. La intensidad de carga de trabajo inyectada en el sistema de la siguiente manera: número de usuarios y por cada microservicio se podrá definir un patrón de carga de la siguiente forma:
            - Carga constante: Mantener una carga constante durante todo el experimento para simular un tráfico uniforme.
            - Carga con picos: Introducir picos de carga en momentos específicos para simular momentos de alta demanda o eventos especiales.
        - La cantidad de réplicas de cada microservicio.

Especificación de requerimientos no funcionales y atributos de calidad

ID	DESCRIPCIÓN	ATRIBUTO
1	El sistema debe implementar mecanismos de validación y confirmación para prevenir errores críticos por parte del usuario, proporcionando advertencias claras a los usuarios.	Usabilidad
2	El sistema debe tener una interfaz de usuario sencilla de navegar, permitiendo a los usuarios operar y controlar el sistema de manera eficiente sin la necesidad de formación extensa	Usabilidad
3	El sistema debe estar construido de manera que todas sus partes puedan ser sometidas a pruebas detalladas.	Mantenimiento
4	El sistema debe estar diseñado de manera modular, permitiendo agregar nuevas funcionalidades o realizar modificaciones en las funciones existentes sin grandes afectaciones al funcionamiento general del sistema.	Mantenimiento
5	El sistema debe implementar un sistema de autenticación que verifique de manera inequívoca la identidad de cada usuario antes de permitir el acceso.	Seguridad
6	El sistema MiMoQ debe permitir una instalación sencilla en los sistemas operativos Windows y Linux, esto si se cumple con las restricciones del sistema.	Compatibilidad

4. Referencias

[1] IBM Corporation, “Preparación del clúster de Kubernetes (K8s)”, https://www.ibm.com/docs/es/tncm-p/1.4.2?topic=software-preparing-kubernetes-k8s-cluster.

[2] Docker Inc., “Get Docker”, https://docs.docker.com/get-docker/.

[3] Oracle, “MySQL Documentation”, http://dev.mysql.com/doc/.

[4] Inc. MongoDB, “MongoDB BI Connector ODBC Driver”, https://www.mongodb.com/docs/bi-connector/current/reference/odbc-driver/

[5] OpenJS Foundation, “Downloads | NodeJS”, https://nodejs.org/en/download.

[6] Docker Inc., “Install Docker Desktop on Linux”, https://docs.docker.com/desktop/install/linux-install/.

[7] Jesús, “¿Cómo instalar Docker en Windows 10?”, https://www.dongee.com/tutoriales/como-instalar-docker-en-windows-10/.

[8] The Linux Foundation, “Kubernetes Documentation”, https://kubernetes.io/docs/home/.

[9] Prometheus Authors, “Overview | Prometheus”, https://prometheus.io/docs/introduction/overview/.

[10] Grafana Labs, “Run Grafana Docker image”, https://grafana.com/docs/grafana/latest/setup-grafana/installation/docker/.

[11] Grafana Labs, “Download Grafana”, https://grafana.com/grafana/download/10.0.0?pg=oss-graf&platform=docker&plcmt=hero-btn-1&edition=oss.

[12] Microsoft, “Disfruta del poder del SO, computadoras y apps Windows 11”, https://www.microsoft.com/es-co/windows/.

[13] XenForo Ltd, “Download Linux | Linux.org”, https://www.linux.org/pages/download/.

[14] Prometheus Authors, “Download | Prometheus”, https://prometheus.io/download/.

[15] Digital Guide IONOS, “Cómo instalar Docker en Ubuntu 22.04: guía paso por paso”, https://www.ionos.es/digitalguide/servidores/configuracion/docker-ubuntu-2204/.