



Propuesta de mejoras a la calidad en base al estándar ISO 25010:2011 – Caso: MineGuard

Hugo Sepúlveda Ordoñez

hugo.sepulvedao@usm.cl

Dr. Marcello Visconti
Profesor Guía

Resumen: MineGuard es una aplicación de gestión de camiones mineros desarrollada para y presentada en la XXXII Feria de Software de la Universidad Técnica Federico Santa María, sin embargo, su calidad como producto no ha sido evaluada. Este trabajo tiene como objetivo evaluar MineGuard utilizando el modelo de calidad de producto del estándar ISO 25010:2011, identificar brechas y proponer mejoras a fin de mejorar la calidad y robustez de MineGuard. Se realiza un análisis cualitativo según la relación de MineGuard con la industria minera para identificar la relevancia de cada criterio principal del modelo de calidad de producto y se evalúa MineGuard en base a aspectos técnicos y no técnicos para identificar brechas según sus fortalezas y debilidades. Se identifican las características de Fiabilidad, Seguridad y Mantenibilidad como las más críticas según su combinación de importancia y nivel de cumplimiento. Finalmente, se proponen mejoras específicas enfocadas en estas características, basadas en el manejo de registros, implementación de un sistema CI/CD, y reforzamiento de los mecanismos de autenticación y autorización; además de mejoras generales para características con brechas menores, con el propósito de que estas puedan servir como una base para mejorar la calidad general de la aplicación.

Palabras Clave: ISO 25010, Calidad, Software, Minería, Feria de Software

1 Introducción

La minería es una de las actividades económicas con mayor importancia y trayectoria en Chile, con impacto en múltiples factores económicos, así como factores sociales, incluyendo salud, vivienda y educación, a lo largo de todo el país y en especial en regiones mineras [1]. La industria minera fue responsable, en promedio, de 10,3% del PIB en el periodo que comprende desde 1996 hasta 2021, con impacto indirecto y no menos importante en otras áreas de la economía como comercio, construcción, transporte y comunicaciones, entre otras; más recientemente, la industria minera es responsable de 11,9% del PIB en 2023 de forma directa [2, 3].

El mantenimiento de los camiones utilizados en minería es fundamental para garantizar la seguridad y la eficiencia en las operaciones de estos. Sin embargo, la detección temprana de fallas o problemas en estos vehículos puede resultar desafiante debido a la complejidad de monitorear y recopilar una gran cantidad de variables en tiempo real para dar un análisis representativo del estado de la máquina. Las fallas operacionales, como consecuencia, pueden ocasionar accidentes graves, retrasos en la producción y costos adicionales.

En el marco de la XXXII Feria de Software, celebrada el 15 de Noviembre de 2024, organizada por la Universidad Técnica Federico Santa María (UTFSM), y en relación a las asignaturas de Gestión de Proyectos Informáticos y Taller de Desarrollo de Proyectos de Informática, cuya conclusión es la presentación de un producto finalizado en la Feria de Software, uno de los productos desarrollados fue MineGuard, una solución relacionada a la gestión y mantención de camiones mineros en Chile, en forma de una aplicación web que pueda servir como herramienta a utilizarse en contextos de empresas mineras, teniendo como algunas de sus funcionalidades:

- Analizar predictivamente datos proporcionados por los camiones, en tiempo real, como presión en las suspensiones, carga que llevan los camiones, temperatura de los frenos, velocidad a la que se



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Propuesta de mejoras a la calidad en base al estándar ISO 25010:2011 – Caso: MineGuard

Nombre del candidato(a): Hugo Sepúlveda Ordoñez

Carrera / Grado: Ingeniería Civil Informática

Campus: Santiago San Joaquín ; **Departamento:** Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Marcello Visconti Zamora, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 02/08/2025

; Firma:

Estudiante o Candidato(a):

Fecha: 01/08/2025

; Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.



- conduce, inclinación de la tolva, entre otros, utilizando técnicas de inteligencia artificial para identificar patrones de comportamiento anormal que indiquen una posible falla inminente.
- Generar alertas automáticas y recomendaciones de mantenimiento preventivo junto con alertas de malas prácticas, como conducir el camión a una velocidad mayor de la debida o llevar más carga de lo que está permitido, con el fin de evitar tiempos de inactividad no planificados, maximizar la vida útil de los equipos, prevenir accidentes, entre otros.
 - Proporcionar una interfaz personalizable que permita a los usuarios visualizar los datos de manera clara y acceder a informes detallados sobre el estado de la flota de camiones.
 - Generar reportes configurables y personalizados que entreguen información relacionada a las distintas variables relacionadas a los componentes de los camiones, los conductores, y las alertas, según parámetros seleccionados.
 - Mostrar un mapa a través del cual poder visualizar la localización de los camiones en tiempo real y hacer un seguimiento de camiones específicos, incluyendo información sobre los camiones, como una imagen de este, su conductor asociado, entre otros.

MineGuard permite a las empresas mineras optimizar la gestión de sus camiones al ofrecer una solución integral para la supervisión y mantenimiento predictivo de estos vehículos, alineada con las exigencias del sector en Chile. Gracias a la aplicación, es posible reducir los riesgos asociados a fallas mecánicas y operacionales, prevenir accidentes y mejorar la seguridad de los trabajadores en el entorno minero. Al anticipar necesidades de mantenimiento y generar alertas en tiempo real, MineGuard no solo minimiza los costos operacionales, sino que también contribuye a una mayor eficiencia productiva, al evitar paradas imprevistas y extender la vida útil de la flota.

Estándares como el ISO 25010 o el ISO 9126 han sido utilizados como herramientas para el análisis de calidad y/o desarrollo de aplicaciones en trabajos de tesis de la UTFSM [4, 5]. En esta memoria se analizará la calidad de uno de los productos desarrollados en Taller de Desarrollo de Proyectos de Informática, MineGuard, a través de un contraste con el estándar definido en la norma internacional ISO 25010:2011, la cual se basa en una serie de características y subcaracterísticas para definir un modelo de calidad para un producto de software. A través de esta evaluación, se podrá determinar en qué medida MineGuard cumple con los requisitos de calidad necesarios para una aplicación confiable y eficiente, y cómo puede optimizarse para responder de manera efectiva a las demandas de las empresas mineras chilenas.

En la sección 2 se define un marco teórico con información relevante para contextualizar la memoria, en la sección 3 se identifican los criterios relevantes del modelo de calidad de producto del ISO 25010 a través de un análisis de la relación entre la industria minera y MineGuard, en la sección 4 se identifican las fortalezas y debilidades de MineGuard a través de un análisis de la aplicación para entender el nivel de cumplimiento de MineGuard según los criterios del modelo de calidad de producto, en la sección 5 se proponen mejoras según las brechas identificadas en la sección 4, con mayor enfoque en las brechas más críticas; se finaliza con las conclusiones en la sección 6.

1.1 Objetivo General

Evaluar la aplicación MineGuard en relación con el modelo de calidad de producto del ISO 25010:2011, con el fin de identificar brechas y proponer mejoras para elevar su calidad.

1.2 Objetivos Específicos

- Analizar las fortalezas y debilidades de MineGuard en términos de las características principales del modelo para identificar brechas existentes.
- Identificar los criterios de calidad relevantes del modelo aplicables a MineGuard.

- Proponer mejoras para optimizar la calidad de MineGuard en base a los resultados del contraste con el modelo de calidad.

2 Marco Teórico

2.1 Calidad de Software

Según el IEEE/ISO/IEC 24765:2017 [6], que provee un vocabulario aplicable para trabajos de ingeniería de software, recolectando y estandarizando terminología, se entiende *Calidad* como:

1. Grado en que el sistema satisface las necesidades declaradas e implícitas de sus diversas partes interesadas y, por lo tanto, aporta valor.
2. Capacidad de un producto, servicio, sistema, componente o proceso para satisfacer las necesidades, expectativas o requisitos del cliente o usuario.
3. Grado en que un conjunto de características inherentes cumple los requisitos

La calidad de software ha estado vigente desde hace más de 50 años con el desarrollo de las primeras aplicaciones de gran escala, siendo hoy en día un pilar fundamental de la ingeniería de software. Se entiende *Aseguramiento de Calidad* como los procesos, y su definición, que ayudan a garantizar que los productos y servicios cumplan con los estándares de calidad y las necesidades del cliente, pudiendo cambiar esta definición ligeramente según la organización.

La calidad en la industria del software tiene diferencias importantes a la de la manufactura, de donde se inspira, como puede ser la ambigüedad en la especificación de requerimientos en las fases iniciales del desarrollo y el entendimiento de estos durante fases más tardías, pudiendo distintos desarrolladores o participantes del proyecto interpretar los requerimientos de diferentes maneras. Por otro lado, es posible que los requerimientos no tomen en cuenta a todos los *stakeholders*, pudiendo generar problemas con los stakeholders excluidos a pesar de haber desarrollado el software según lo pactado. Además, si bien la aplicación puede tener sus funcionalidades correctamente implementadas, esta también depende de características no funcionales, como la rapidez de la aplicación, que de no ser tomadas en cuenta, pueden llevar a que los usuarios se nieguen a utilizarla. [7]

De acuerdo con el reporte realizado en 2020 por el *Consortium for Information & Software Quality* (CISQ) [8], el costo total de la mala calidad de software en los Estados Unidos es de aproximadamente 2 billones de dólares, siendo el mayor contribuidor las fallas operacionales de software. En este contexto, los modelos de calidad de software se presentan como herramientas valiosas para mitigar estas pérdidas económicas, ya que proporcionan marcos de trabajo y directrices para asegurar que las aplicaciones se desarrollen y mantengan con los estándares necesarios para su correcto funcionamiento.

2.2 Modelos de Calidad

A lo largo de los años, diversos modelos y normas han sido desarrollados para estandarizar la evaluación de la calidad del software, estableciendo características y subcaracterísticas que permiten medirla de forma sistemática, siendo el primero el presentado por McCall et al. (1977) [9], basado en tres aspectos del producto: revisión, transición y operación; además, consideraba relaciones entre características y métricas.

En su revisión de modelos de calidad de software para la evaluación de productos de software, Miguel et al. (2014) [10] describen 28 terminologías, que corresponden a características y subcaracterísticas, dependiendo del modelo, encontrando además que el ISO 25010 es el modelo más completo entre los analizados, cubriendo 26 de las 28 características.

2.2.1 ISO 9126

El estándar ISO/IEC 9126 [11], publicado en 1991 y actualizado en 2001, fue uno de los primeros esfuerzos internacionales para estructurar la evaluación de la calidad del software y se basa en el modelo de McCall, entre otros. Este modelo propuso una estructura jerárquica de características de calidad, organizadas en seis características principales: Funcionalidad, Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad. Cada característica se dividía en subcaracterísticas más específicas, las cuales podían evaluarse mediante atributos o métricas específicas, definidas en función del tipo de software. La Figura 1 muestra un esquema del modelo de calidad presentado en el ISO 9126.

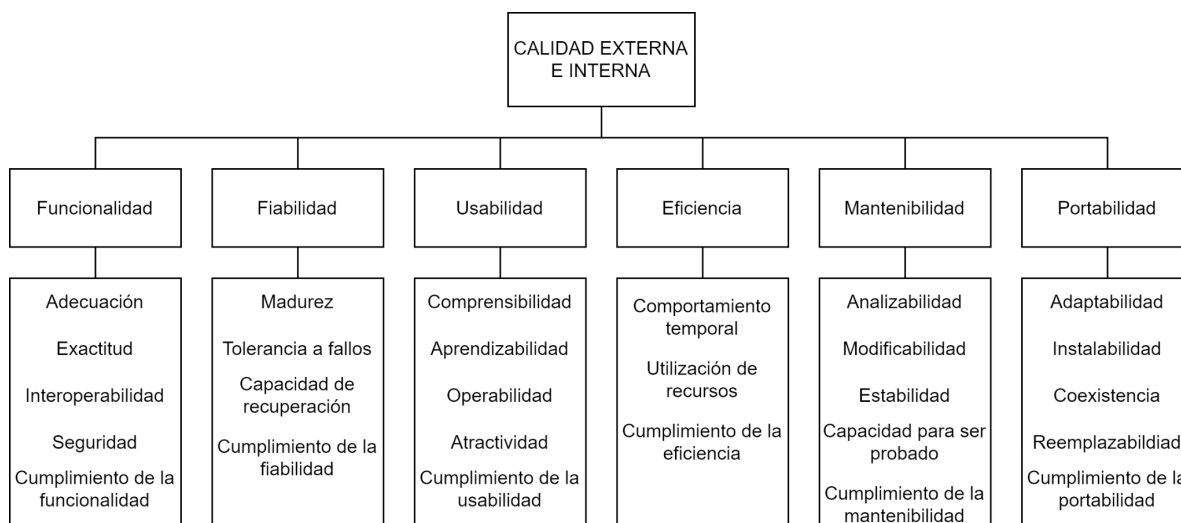


Figura 1. Modelo de calidad para la calidad externa e interna del ISO 9126

Elaboración propia

2.2.2 ISO/IEC 25010:2011

Es un estándar internacional que define un modelo de calidad para productos y sistemas de software, el cual reemplaza al ISO 9126, reorganizando las características y subcaracterísticas de calidad de software, e incluyendo aspectos relevantes para las tendencias actuales, como Seguridad y Compatibilidad. Este estándar pertenece a la familia ISO 25000 (también conocida como SQuaRE, *Systems and Software Quality Requirements and Evaluation*) y se utiliza para evaluar y garantizar la calidad del software en diferentes dimensiones [12]. El modelo de calidad de ISO 25010:2011 consta de dos categorías principales:

Modelo de calidad del producto, que evalúa la calidad interna y externa del software mediante 8 características principales:

- **Adecuación funcional:** El software cumple con las funciones que necesita el usuario. Sus subcaracterísticas son: completitud funcional, corrección funcional, y pertinencia funcional.
- **Eficiencia de desempeño:** El sistema responde rápidamente y usa adecuadamente los recursos. Sus subcaracterísticas son: comportamiento temporal, utilización de recursos, y capacidad.
- **Compatibilidad:** El software es capaz de operar con otros sistemas o productos. Sus subcaracterísticas son: coexistencia, e interoperabilidad.

- **Usabilidad:** Es fácil de entender, aprender y usar por los usuarios. Sus subcaracterísticas son: reconocibilidad de la adecuación, aprendizabilidad, operabilidad, protección contra errores de usuario, estética, y accesibilidad.
- **Fiabilidad:** El software funciona de manera estable y consistente bajo condiciones específicas. Sus subcaracterísticas son: madurez, disponibilidad, tolerancia a fallos, y capacidad de recuperación.
- **Seguridad:** Protege la información y los datos de accesos no autorizados. Sus subcaracterísticas son: confidencialidad, integridad, no repudio, responsabilidad, y autenticidad.
- **Mantenibilidad:** El software es fácil de modificar, mejorar y corregir. Sus subcaracterísticas son: modularidad, reusabilidad, analizabilidad, capacidad para ser modificado, y capacidad para ser probado (testabilidad).
- **Portabilidad:** El software puede ser transferido de un entorno a otro sin problemas. Sus subcaracterísticas son: adaptabilidad, instalabilidad, y reemplazabilidad.

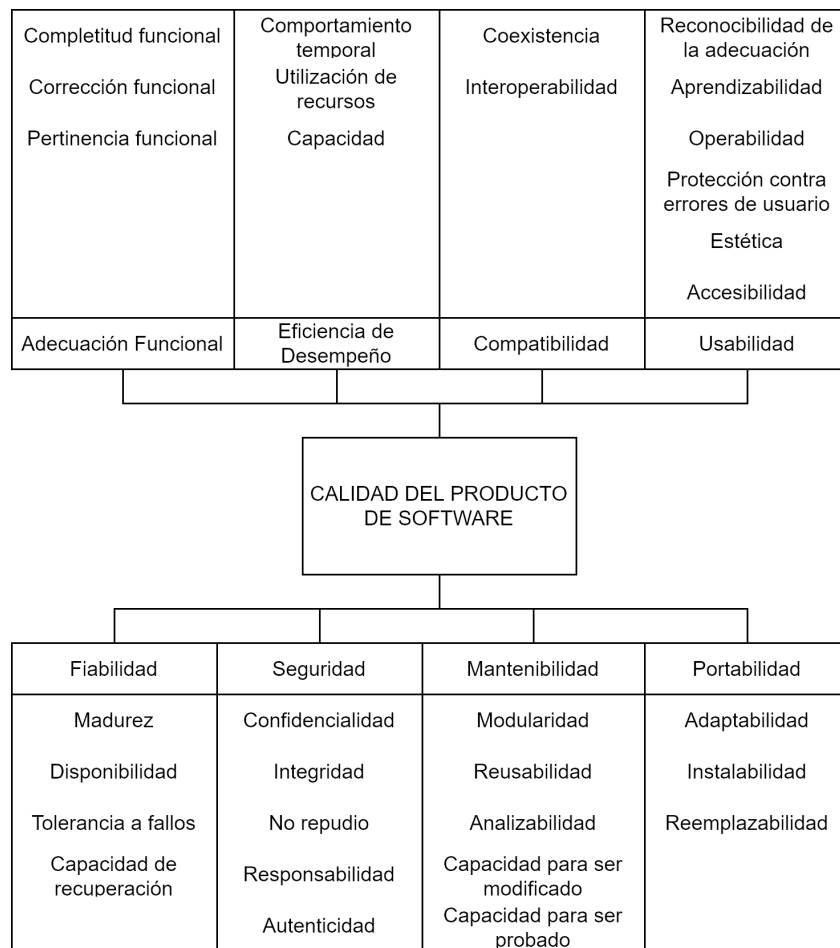


Figura 2. Modelo de calidad de producto del ISO 25010:2011

Elaboración propia

Modelo de calidad en uso, que evalúa la experiencia del usuario y el valor que el software ofrece mediante 5 características:

- **Eficiencia en el uso:** El usuario puede alcanzar sus objetivos con el mínimo esfuerzo.



- **Satisfacción:** El software brinda satisfacción al usuario al cumplir sus expectativas.
- **Libertad de riesgo:** El uso del software no causa daños o riesgos a personas, empresas o el medio ambiente.
- **Cobertura de contexto:** El software puede usarse eficazmente en diferentes contextos.
- **Efectividad:** El usuario logra sus objetivos utilizando el software correctamente.

ISO 25010 es utilizado tanto en fases de desarrollo, para garantizar que se cumplan requisitos de calidad, como en la evaluación posterior de productos ya desarrollados. En este caso se utilizará como herramienta de evaluación del producto ya desarrollado, con enfoque en el modelo de calidad del producto. La Figura 2 muestra un esquema del modelo de calidad de producto, incluyendo características principales y subcaracterísticas. La elección de este estándar para analizar la calidad del software desarrollado en este proyecto se debe a su reconocimiento internacional como un marco de referencia integral para evaluar y garantizar la calidad en diferentes etapas del ciclo de vida del software.

2.3 Software y Transformación Digital en la Industria Minera

La industria minera ha experimentado en las últimas décadas una transformación profunda impulsada por la Industria 4.0 (también conocida como Cuarta Revolución Industrial), caracterizada por la integración de tecnologías digitales en los procesos productivos, combinando internet, sensores inteligentes y sistemas de TI. Entre estas tecnologías se destacan la automatización, el internet de las cosas, la inteligencia artificial y los sistemas ciberfísicos, todos soportados por soluciones de software avanzadas.

Sánchez & Hartlieb (2020) [13] describen algunas de las principales tecnologías relacionadas a la transformación digital en la industria minera:

- **Automatización, Robótica y Operación remota:** Mejoran la seguridad y reducen costos en minería al disminuir la necesidad de operadores en terreno. El uso de equipos autónomos crece rápidamente, aumentando la productividad y eficiencia.
- **Internet de las Cosas y Sensores Inteligentes:** Conectan equipos y sensores para capturar datos en tiempo real, mejorando la planificación y la toma de decisiones de manera integrada.
- **Inteligencia Artificial y Machine Learning:** Transforman grandes volúmenes de datos en mejores planes operativos y decisiones rápidas. Además, optimizan el mantenimiento con el uso de modelos predictivos y mejoran actividades de prospección y exploración.
- **Gemelo Digital:** Se crean réplicas virtuales de las operaciones para simular escenarios, prever fallos y optimizar la producción y el mantenimiento.

La creciente dependencia de software especializado en la industria minera hace que la evaluación rigurosa de su calidad sea crítica para garantizar operaciones seguras y eficientes. En este contexto, surge la necesidad de soluciones como MineGuard, diseñada para fortalecer la gestión de mantenimiento de camiones mineros aprovechando las nuevas tecnologías de la última revolución digital. Dado que cualquier falla en las funcionalidades críticas de este tipo de sistemas podría derivar en riesgos operativos, pérdidas económicas considerables o incluso accidentes laborales, es fundamental contrastar MineGuard con un marco de referencia sólido como el modelo de calidad de producto del ISO 25010.

3 Identificación de Criterios Relevantes en Relación a la Industria Minera

La industria minera, siendo la industria más importante de Chile, tiene altos estándares en lo que refiere al funcionamiento de la industria, incluyendo procesos, equipamiento, maquinaria, trabajadores, entre otros, y es esperable que esto aplique para las aplicaciones que se utilizan en la misma, por lo que es importante que

todos los criterios de evaluación del ISO 25010 tengan un buen nivel de cumplimiento, sin embargo, se identificarán los criterios más importantes para entender cómo MineGuard cumple con estos y dónde son más necesarias y críticas las posibles mejoras a la aplicación.

3.1 Adecuación Funcional

El software debe cumplir con precisión las tareas específicas para las que fue diseñado, especialmente en un entorno crítico y de alto riesgo como la industria minera. En este contexto, la adecuada respuesta del sistema frente a situaciones reales de operación representa una necesidad esencial para mantener la seguridad, la eficiencia y la continuidad operativa. La industria minera chilena, caracterizada por su escala masiva, condiciones extremas y alta inversión en maquinaria, requiere soluciones digitales que respondan con exactitud a necesidades operativas concretas y que se alineen con los objetivos estratégicos del negocio.

MineGuard debe ser capaz de garantizar que todas sus funcionalidades principales, como el análisis de datos, la generación de alertas y de reportes, y la visualización geográfica de la flota de camiones, funcionen de manera precisa, confiable y consistente en todo momento. Cada una de estas funcionalidades responde a una necesidad real de la operación minera moderna, donde la anticipación de fallos y la optimización de mantenimientos son clave para reducir tiempos de inactividad no planificados y maximizar la vida útil de activos de alto valor.

3.2 Eficiencia de Desempeño

La eficiencia de desempeño adquiere especial relevancia para MineGuard por trabajar con análisis de datos en tiempo real de múltiples camiones simultáneamente. La utilización de recursos computacionales debe ser eficiente, especialmente considerando que el sistema puede operar en infraestructuras remotas con limitaciones de conectividad o capacidad de procesamiento, situación común en operaciones mineras chilenas ubicadas en zonas geográficamente desafiantes.

Adicionalmente, MineGuard debe garantizar tiempos de respuesta mínimos y procesamiento veloz de la información para cumplir con su propósito de prevención. Un retardo en la generación de alertas o en la actualización de datos podría anular la efectividad del mantenimiento predictivo, permitiendo que una falla evolucione sin ser detectada a tiempo; así, la eficiencia de desempeño no solo impacta en la experiencia de usuario o en la percepción de calidad del sistema, sino que se convierte en un aspecto de alta importancia para asegurar la seguridad operativa y la disponibilidad.

3.3 Compatibilidad

La aplicación podría necesitar integrarse con sistemas de planificación de recursos empresariales (ERPs, por sus siglas en inglés) corporativos y otras herramientas de gestión operacional, garantizando la interoperabilidad y el intercambio fluido de información; además, la incorporación efectiva al flujo de trabajo existente puede ser crucial para su adopción. Dependiendo de la infraestructura tecnológica específica de cada empresa minera, la necesidad de integración podría volverse más o menos importante, sin embargo, la integración con otros sistemas, si bien valiosa, no es un requisito indispensable para que MineGuard cumpla con sus funciones principales, por lo tanto, aunque la compatibilidad influye positivamente en la flexibilidad y escalabilidad del producto, en términos de criticidad para la operación base del software, otras características como la fiabilidad, la eficiencia de desempeño y la seguridad tienen un impacto más directo y urgente en el cumplimiento de su propósito en el entorno minero.

3.4 Usabilidad

Para que MineGuard sea utilizado eficazmente por el personal minero en sus operaciones diarias, debe tener una interfaz intuitiva y fácil de usar. Los operadores y el personal de mantenimiento deben poder comprender fácilmente los datos visualizados, interpretar las alertas y recomendaciones, generar informes y utilizar las



funciones de seguimiento de ubicación sin necesidad de una capacitación exhaustiva o con dificultad, más aún considerando que los trabajadores podrían no tener amplios conocimientos en tecnologías de la información.

Visto desde otro ángulo, un sistema que no cumple adecuadamente con el criterio de usabilidad puede derivar en errores de interpretación de datos o en acciones incorrectas por parte del usuario, lo que no solo podría generar costos económicos, sino también poner en riesgo la seguridad del personal y la continuidad de las operaciones. Por estas razones, la usabilidad se posiciona como un elemento de alta importancia que MineGuard debe priorizar para asegurar su adopción y contribución al mejoramiento de los procesos operativos mineros.

3.5 **Fiabilidad**

La fiabilidad emerge como un criterio particularmente relevante para MineGuard, considerando que estará operando en un contexto industrial donde los fallos pueden tener consecuencias catastróficas; un software de monitoreo para camiones mineros debe mantener su nivel de rendimiento en condiciones normales de uso sin interrupciones. La tolerancia a fallos y la capacidad de recuperación son subcaracterísticas particularmente importantes, ya que cualquier interrupción en el monitoreo podría significar la pérdida de alertas críticas sobre fallos inminentes en los camiones.

En la industria minera, es prioritario minimizar los tiempos de inactividad y mantener las operaciones andando con la menor cantidad de incidentes posibles, pues parar las operaciones debido a fallas de equipamiento o maquinaria puede resultar en grandes pérdidas monetarias y riesgos a los trabajadores, por lo que es necesario asegurar el estado, condición y efectividad de las herramientas a utilizar, lo que aplica para MineGuard. La industria minera busca hacer frente a los problemas de fiabilidad a través de soluciones emergentes basadas en inteligencia artificial e internet de las cosas, entre otras. [14]

3.6 **Seguridad**

La aplicación debe garantizar la confidencialidad de información operativa estratégica de las empresas mineras, protegiendo datos como ubicación y estado de los camiones, y parámetros operacionales que podrían ser valiosos para competidores. Por otro lado, la autenticidad y el control de acceso son fundamentales para garantizar que solo personal autorizado pueda modificar parámetros críticos del sistema o acceder a información sensible, especialmente considerando el alto valor de los activos monitoreados en la industria minera.

Barkat Ullah et al. (2025) [15] describen cómo la ciberseguridad se ha vuelto un elemento cada vez más importante en el contexto de la industria minera, donde la adopción de tecnologías modernas, como el internet de las cosas, abren mayores posibilidades a ataques cibernéticos, encontrando además que las empresas mineras no parecen darle el foco suficiente a la seguridad a pesar de amenazas crecientes de ciberataques.

3.7 **Mantenibilidad**

La capacidad para implementar mejoras y correcciones de manera eficiente resulta especialmente valiosa en un software como MineGuard, que se encarga de monitorear equipos de alto valor cuya tecnología y requerimientos evolucionan constantemente en el contexto minero. La naturaleza dinámica de las operaciones mineras, con cambios en los modelos de camiones, actualizaciones de sensores o introducción de nuevas variables de monitoreo, exigen que el software pueda adaptarse rápidamente sin necesidad de grandes esfuerzos de reestructuración.

El código debe ser modular y fácilmente modificable para incorporar estos nuevos modelos de camiones o variables de monitoreo, además de estar bien documentado para que futuros desarrolladores puedan localizar,

entender y modificar partes específicas del sistema. Por otro lado, la capacidad para ser probado del sistema es relevante para verificar que las modificaciones no introduzcan vulnerabilidades o fallos que pudieran comprometer las funciones que cumple el software.

3.8 Portabilidad

La portabilidad presenta relevancia limitada para MineGuard al ser una aplicación web que ya cuenta con cierta independencia de plataforma inherente. Aunque debe considerarse la adaptabilidad a diferentes navegadores y dispositivos, permitiendo operación tanto desde centros de control como en campo, esta característica está parcialmente cubierta por el diseño web responsivo estándar en el desarrollo contemporáneo.

3.9 Importancia de cada Criterio

Tomando en cuenta el análisis realizado de cada criterio en base al contexto de la industria minera y su relación con MineGuard y sus funcionalidades, se define la importancia de cada criterio en la Tabla 1. En base a esto, se priorizará la propuesta de mejoras en los criterios donde haya mayor importancia.

Tabla 1. Importancia de criterios para MineGuard
Elaboración propia

Característica	Nivel de Importancia	Justificación
Adecuación Funcional	Alta	MineGuard debe cumplir con precisión funciones críticas como análisis predictivo, alertas y visualización geográfica; cualquier error puede llevar a decisiones equivocadas de mantenimiento o riesgos operativos.
Eficiencia de Desempeño	Alta	El software debe procesar datos en tiempo real de múltiples camiones, ser eficiente en infraestructura remota y ofrecer tiempos de respuesta mínimos para prevenir fallos.
Compatibilidad	Media	Aunque la integración con ERPs y sistemas existentes puede ser importante, no es indispensable para que MineGuard cumpla sus funciones principales.
Usabilidad	Alta	La interfaz debe ser intuitiva para que operadores y personal de mantenimiento utilicen el sistema eficazmente sin necesidad de capacitación extensa, reduciendo errores y riesgos.
Fiabilidad	Máxima	La operación continua y la alta disponibilidad son esenciales para evitar fallas, accidentes y pérdidas económicas graves en el entorno minero.
Seguridad	Máxima	MineGuard debe proteger información estratégica y controlar el acceso para prevenir modificaciones no autorizadas, en un contexto de crecientes amenazas cibernéticas en minería.
Mantenibilidad	Alta	Es crucial actualizar el software fácilmente para nuevos equipos y variables, garantizando modularidad y testabilidad para mantener la robustez del sistema.
Portabilidad	Baja	Aunque se requiere compatibilidad con diferentes navegadores y



		dispositivos, el diseño web responsivo cubre en gran medida las necesidades de portabilidad de MineGuard.
--	--	---

4 Identificación de Fortalezas y Debilidades

En esta sección se analizará el estado actual de MineGuard, mismo estado en el que se encontró durante la Feria de Software, de tal forma que se pueda ahondar en distintas partes de su desarrollo, i.e., arquitectura, base de datos, librerías, frameworks, entre otros, además de características no técnicas como la percepción de la aplicación y la retroalimentación recibida, y explicar cómo es que estos aspectos afectan el cumplimiento de las distintas características y subcaracterísticas definidas en el modelo de calidad de producto del ISO 25010, para finalmente localizar qué áreas necesitan mejoras.

4.1 Aspectos Generales de MineGuard

La aplicación fue desarrollada de forma intermitente a lo largo de 4 meses, utilizando la metodología Scrum con 3 sprints, incluyendo tareas de definición de requerimientos, revisión de sprint, retrospectiva de sprint y *daily meetings*. La aplicación fue desarrollada en una plantilla de Horizon UI, utilizando Next.js junto con TypeScript para el desarrollo de frontend, incluyendo componentes entregados por la plantilla, más componentes desarrollados con Chakra UI. Para el desarrollo del backend se utilizó Node.js con GraphQL y Apollo Server, y paralelamente, Django con Django REST Framework, dividiendo la API y las funcionalidades entre ambos. Por otro lado, se utilizó la base de datos NoSQL MongoDB. Adicionalmente, se utilizó Docker como herramienta para la virtualización completa de la aplicación. La Figura 3 ilustra la arquitectura general de MineGuard.

4.2 Adecuación Funcional

La adecuación funcional de MineGuard fue sistemáticamente evaluada durante todo su proceso de desarrollo, con énfasis en verificar que cada funcionalidad cumpliera con los objetivos definidos y aportará valor real en el contexto previsto. Al término de cada uno de los tres sprints del proyecto, tanto profesores como ayudantes realizaron revisiones formales del sistema, validando que las funcionalidades implementadas respondieran de manera completa, correcta y pertinente a las historias de usuario previamente definidas. Estas historias fueron elaboradas considerando necesidades específicas del sector minero, e incluyeron criterios de aceptación concretos para asegurar que los requisitos funcionales estuvieran bien alineados con los problemas a resolver.

Adicionalmente, la aplicación fue sometida a validación externa durante la Feria de Software, instancia en la que múltiples usuarios, incluyendo jueces con experiencia tanto en la industria minera como en el desarrollo de software, exploraron y probaron las distintas funcionalidades del sistema. Esto permitió confirmar la completitud funcional de MineGuard desde una perspectiva cercana a la de usuario final. La interacción con expertos del dominio contribuyó así a reforzar la validación de las funcionalidades implementadas, aumentando el nivel de confianza en que el sistema cumple de manera efectiva con los objetivos para los cuales fue diseñado.

4.3 Eficiencia de Desempeño

Con respecto a las alertas automáticas, estas fueron implementadas utilizando la aplicación Twilio, que proporciona una API web para distintas funcionalidades de mensajería y comunicación; en este caso, los mensajes son enviados a través de Whatsapp, de esta forma, el tiempo para el envío y recepción del mensaje depende principalmente de la conectividad; por otro lado, Twilio puede alcanzar hasta 400 mensajes por

segundo, por lo que, teniendo en cuenta que se envía solo una alerta genérica a un conductor comunicando los errores y que fallas en los camiones no tienen ese nivel de frecuencia, se cumple con las necesidades de capacidad. [16]

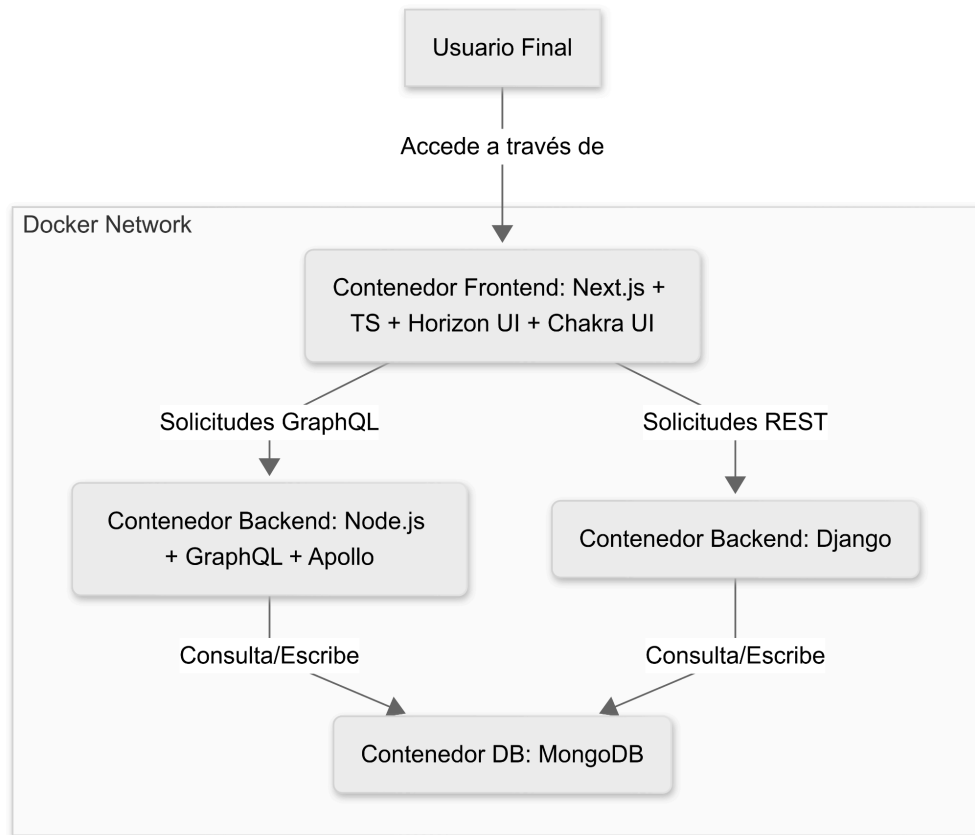


Figura 3. Arquitectura general de MineGuard

Elaboración propia

Los reportes son generados utilizando la herramienta xhtml2pdf en Django, que convierte código HTML en PDF, está escrita completamente en Python y no tiene dependencias externas de motor gráfico, dependiendo la velocidad de generación del reporte principalmente de los estilos CSS utilizados; MineGuard utiliza pocos estilos CSS, por lo que para casos típicos en los que se genera un reporte de pocos camiones o conductores, la generación puede tomar segundos, en los casos más extremos puede tomar minutos, lo que sigue siendo un tiempo aceptable, considerando la urgencia y frecuencia de estos.

El apartado más exigente de MineGuard corresponde al monitoreo de camiones, donde se utiliza un WebSocket implementado con la herramienta Channels de Django, esta implementación fue realizada de la forma más directa posible, sin herramientas o configuraciones adicionales, por lo que toda la lógica corre en un único proceso, limitando el *throughput* a un solo núcleo de la CPU. Teniendo en cuenta que cada camión tiene más de 100 datos y habiendo inicialmente 20 camiones, se estarían enviando 2000 datos en un objeto JSON a través de un único mensaje por segundo que pesa alrededor de 20 KB, lo que es manejable para el WebSocket; sin embargo, debido a la implementación, cada cliente tiene su propio bucle de envío independiente, de esta forma, para 10 clientes mirando el mismo camión, se generan 10 mensajes JSON idénticos por separado, lo cual es redundante y puede saturar la CPU dependiendo de la escala.



En la implementación de GraphQL existen queries mal diseñadas, que violan las convenciones de GraphQL al no ser declarativas: los clientes deberían elegir qué campos quieren, no el servidor; al haber queries especializadas para cada consulta al mismo objeto, por ejemplo, una query para el nombre y apellido de un usuario, y otra aparte para el email, en lugar de una sola query de usuario con todos los datos, se genera una redundancia de llamadas al backend, lo que aumenta el número de llamadas a la base de datos; además, este diseño puede romper el mecanismo de *caching* de Apollo, lo que afecta al rendimiento general del sistema.

4.4 Compatibilidad

En cuanto a la coexistencia, la aplicación fue contenedorizada utilizando Docker, lo que permite que cada uno de sus componentes, frontend, APIs, WebSocket y base de datos, operen de manera aislada, compartiendo recursos de hardware sin afectar el rendimiento de otros sistemas que puedan ejecutarse en el mismo entorno. Este enfoque reduce conflictos de dependencias y permite gestionar el consumo de recursos de manera eficiente.

Respecto a la interoperabilidad, MineGuard fue diseñado utilizando tecnologías ampliamente adoptadas que favorecen el intercambio de información con otros sistemas, GraphQL y REST, como mecanismos de comunicación, los cuales facilitan tanto consultas flexibles como integraciones sencillas con herramientas externas o sistemas legados. Además, el uso de Next.js con TypeScript en el frontend asegura contratos de datos más robustos, mientras que el diseño basado en componentes estándar garantiza que la interfaz pueda integrarse sin mayores problemas en otros entornos de visualización. Además, MongoDB como base de datos permite una fácil exposición de datos mediante formatos estándar como JSON gracias a su compatibilidad nativa con este.

4.5 Usabilidad

MineGuard logra comunicar de forma efectiva su propósito al usuario desde los primeros contactos. Según la retroalimentación obtenida en la feria y en una prueba de usuario, los usuarios reconocieron con facilidad para qué sirve el sistema. Aunque la interfaz es intuitiva, la presencia de una *landing page* informativa representa el único recurso de apoyo inicial. La ausencia de tutoriales, documentación más detallada o ayudas visuales limita el soporte para usuarios completamente nuevos o externos al ámbito minero.

El sistema fue percibido como fácil de aprender a usar, según la retroalimentación obtenida tanto en la feria como en pruebas informales. Los usuarios logran operar funciones clave tras pocos minutos, por lo que es esperable que adquirir experticia con la aplicación tome desde minutos a algunas horas. Esto se debe principalmente a un diseño bastante intuitivo en la mayoría de las secciones, aún así, algunas secciones, como generación de reportes, presentan complejidades que pueden ralentizar la comprensión inicial.

En cuanto a operabilidad, MineGuard es mayormente intuitivo y permite operar funciones esenciales mediante pocos pasos, sin embargo, ciertas decisiones de diseño afectan la experiencia del usuario: botones ocultos, flujos poco claros (como el acceso a generación de alertas manuales) y la falta de visibilidad de elementos interactivos (como los íconos de camiones clickeables). Además, hay tareas que pueden volverse tediosas, como seleccionar múltiples camiones y datos en la generación de reportes, debido a la ausencia de herramientas de filtrado o selección múltiple eficiente.

El sistema implementa validaciones en algunos formularios (inicio de sesión, alertas manuales, etc.), pero otras secciones críticas carecen de protección básica contra errores (por ejemplo, configuración de alertas automáticas o configuración de umbrales de alertas). En muchos casos, los errores del sistema no se comunican en absoluto al usuario, y las recomendaciones de acción son mínimas o inexistentes. Tampoco se impide el ingreso de datos incorrectos en todos los formularios.

La interfaz mantiene un diseño visual coherente en gran parte de la aplicación, con una paleta de colores definida y elementos visuales agradables. Se han recibido comentarios positivos respecto a la estética, especialmente en la feria y revisiones de sprint; no obstante, existen pequeñas inconsistencias entre secciones, i.e., colores, botones, espaciado, lo que puede afectar la percepción de profesionalismo o unidad visual.

El sistema no considera aspectos de accesibilidad. No se han seguido lineamientos o directrices de ningún tipo ni se han implementado mecanismos de inclusión para usuarios con discapacidades. Tampoco se diseñó pensando en navegación por teclado, uso de lectores de pantalla o contrastes visuales mejorados. Esta ausencia de accesibilidad limita significativamente la usabilidad del sistema para una parte importante de la población y representa una oportunidad crítica de mejora.

4.6 Fiabilidad

MineGuard ha sido probado principalmente durante su presentación en la Feria de Software, donde estuvo operativo durante aproximadamente 10 horas sin fallos críticos. Durante el uso normal, se ha observado una baja frecuencia de errores, lo cual sugiere un nivel razonable de madurez para un sistema en desarrollo. Sin embargo, algunas funcionalidades presentan fallas con mayor frecuencia, particularmente la monitorización (pérdida de conexión con el WebSocket) y la generación de reportes (problemas en la selección de datos y copia de parámetros de reporte), lo que indica que aún existen áreas del sistema susceptibles a mejoras.

En cuanto a disponibilidad, el sistema ha demostrado estar accesible y operativo mientras se encuentra en ejecución, sin caídas registradas durante su uso prolongado en la feria. No obstante, actualmente no se emplean mecanismos formales de monitoreo de disponibilidad, como *health checks* o herramientas de *uptime*, lo que limita la capacidad para detectar y responder proactivamente a interrupciones. Además, el entorno de despliegue no está estandarizado, lo cual introduce incertidumbre sobre la confiabilidad operativa en diferentes escenarios.

MineGuard presenta cierto grado de tolerancia a fallos gracias al desacoplamiento entre funcionalidades, lo que permite que errores en una parte del sistema no afecten a otras. Sin embargo, el manejo de errores no está implementado de manera consistente a lo largo de la aplicación, y no existen mecanismos de recuperación automática como *retries* o *fallbacks*. Esto implica que, aunque el sistema no se detiene completamente ante fallos, no cuenta con defensas robustas para lidiar con errores inesperados.

Actualmente, MineGuard no incorpora mecanismos específicos de recuperación ante fallos, como respaldo de datos, persistencia de sesiones o restauración del estado del sistema, tampoco existen estrategias para garantizar la integridad de los datos en caso de desconexiones o caídas. En caso de reinicio, el sistema puede volver a un estado funcional en cuestión de segundos o minutos, pero no hay soporte automatizado para recuperar el estado anterior al fallo, lo que limita su capacidad de recuperación.

4.7 Seguridad

La aplicación implementa JSON Web Token (JWT) para manejo de sesiones y solo utilizando un *access token*, o sea, sin utilizar un *refresh token*, guardando este JWT en una cookie simple. Esta implementación acarrea problemas como [17, 18]:

- **Cookies sin flags *HttpOnly* ni *Secure*:** guardar el access token en una cookie sin las flags *HttpOnly* y *Secure* deja el token expuesto a ataques XSS, pues un script malicioso puede leerlo y enviarlo a un atacante. Además, al no usar la opción *SameSite*, la cookie podría enviarse en solicitudes *cross-site*, ampliando el riesgo de ataques CSRF.
- **Ausencia de *Refresh Token*:** solo usar un token de acceso de corta duración sin un mecanismo de refresh token impide la rotación segura de credenciales y fuerza a otorgar JWT de larga vida o a recargar manualmente la sesión, lo que aumenta la ventana de exposición ante un token robado.



Por otro lado, la aplicación presenta problemas en torno a la autenticación e integridad, pues el JWT es utilizado de forma inconsistente a lo largo de esta: en Node.js con GraphQL solo se autentica al usuario a través del JWT en algunas partes de la API; por otro lado, en Django, este no es utilizado en ningún momento, es decir, no se verifica al usuario en ninguna parte de la API implementada en Django, lo que incluye el WebSocket. Esto implica que todos los datos de los camiones están completamente expuestos a cualquier persona que haga una solicitud de estos, además de acceso libre a los procesos relacionados a la generación de reportes, lo que representa una vulnerabilidad crítica.

En cuanto a No Repudio y Responsabilidad, MineGuard no presenta *logs* ni historial de ningún tipo, lo que tiene implicancias severas: no registrar eventos críticos (*login*, *refresh*, fallos de verificación, cambios de privilegios, etc.) impide detectar intrusiones o anomalías en tiempo real, dejando la aplicación ciega ante ataques o fugas de credenciales, además de dejar sin posibilidad de identificar y amonestar a un usuario por uso indebido de la aplicación. Adicionalmente, esto implica que no se controla la creación, uso o destrucción de JWT, ni se puede invalidar tokens revocados o expuestos, lo que dificulta aplicar bloqueos de sesión ante un incidente. Combinando esta vulnerabilidad con la carencia de autenticación en la aplicación, abre la posibilidad de que, por ejemplo, un usuario no registrado realice cambios en los umbrales de fallo en los camiones, lo que podría hacer que se envíen alertas falsas o, peor aún, que no se envíen alertas, pudiendo generar accidentes.

4.8 Mantenibilidad

MineGuard logra una cierta modularidad en su arquitectura al dividir sus servicios en distintos contenedores para Node.js, Django y MongoDB. Dentro del frontend, existe una organización de componentes en carpetas, pero no todas las páginas utilizan componentes pequeños y reutilizables. Además, el uso de "macrocomponentes" con mucha lógica interna acoplada limita severamente la capacidad de modificar una parte del sistema sin afectar otras. En el backend, aunque las APIs están separadas por funcionalidad, la falta de un patrón formal, como Modelo-Vista-Controlador (MVC) o similares, genera riesgos de acoplamiento entre distintas funciones.

El nivel de reusabilidad en MineGuard es moderado. En el frontend, algunos componentes basados en Horizon UI son reutilizables gracias a su diseño configurable, pero los componentes propios tienden a ser poco configurables y están moderadamente acoplados a casos específicos. En el backend, pocas funciones son realmente reutilizables; la mayoría de las llamadas a APIs están diseñadas para cumplir funcionalidades específicas, limitando su uso en otros contextos o nuevos módulos; asimismo, el diseño en Node.js, que rompe con las convenciones de GraphQL al haber queries para obtener información específica de camiones o conductores, afecta negativamente a la reusabilidad.

La capacidad para analizar el sistema es limitada. Si bien se tienen archivos README que ayudan a montar la aplicación en lo que respecta a instalación y funcionamiento básico, estos no están completos; además, no existe documentación técnica detallada de las APIs o de los componentes propios del frontend (no así para los componentes de la plantilla), lo que dificulta comprender rápidamente el sistema. El uso de comentarios en el código es inconsistente, variando mucho entre distintas partes, aunque puede ser de gran ayuda, esta inconsistencia implica que no se puede depender de estos. Por otro lado, si bien se utiliza ESLint en el frontend para mantener cierto orden, no hay herramientas de análisis de código para el backend, ni tampoco prácticas formales de seguimiento de errores (como logs estructurados o uso de herramientas tipo Sentry).

En cuanto a modificabilidad, el sistema presenta dificultades importantes para la modificación. Realizar cambios, tanto en el backend como en el frontend, incluso menores, puede ser riesgoso, especialmente debido a la falta de documentación y la complejidad acumulada en algunos componentes y funcionalidades. Aunque ciertas separaciones lógicas en el backend reducen el riesgo de romper múltiples áreas a la vez, la ausencia de pruebas automatizadas agrava el riesgo de introducir errores. La curva de aprendizaje es alta para nuevos

desarrolladores y refactorizaciones importantes serían complejas incluso para personas familiarizadas con el sistema actual.

Finalmente, MineGuard presenta una testabilidad baja. Actualmente solo existen pruebas manuales para algunas llamadas de API en Node.js, mientras que Django y el frontend carecen totalmente de pruebas automatizadas. No se utilizan frameworks de testing ni existen entornos de pruebas formalizados, lo que complica la verificación de la funcionalidad antes de pasar cambios a producción, pues esta verificación debe hacerse de forma manual, lo que implica un considerable gasto de tiempo, además de añadir riesgo humano. A raíz de esto, no solo se dificulta encontrar y corregir errores, sino también se limita la posibilidad de mejorar o escalar el sistema de manera segura.

4.9 Portabilidad

La adaptabilidad de MineGuard se ve beneficiada por su diseño web responsivo, el uso de tecnologías modernas como Django, Next.js y Node.js, y comunicación basada en REST y GraphQL, que permiten que la aplicación funcione en múltiples navegadores y dispositivos, ya sea estaciones de monitoreo en centros de control o dispositivos en campo. Además, su arquitectura contenedorizada permite trasladar el sistema a distintos entornos operacionales, como: servidores locales, servicios en la nube, o distintos sistemas operativos *host*, con relativa facilidad. Sin embargo, esta adaptabilidad se ve parcialmente afectada por la falta de pruebas sistemáticas en diferentes configuraciones de hardware y software, así como por una dependencia marcada en ciertas herramientas y configuraciones de entorno no completamente documentadas; además, el diseño web responsivo está mal implementado en algunas secciones, por lo que no se cumple en su totalidad.

La instalabilidad del sistema se facilita mediante el uso de Docker, que permite encapsular dependencias y servicios en contenedores separados (Node.js, Django y MongoDB). Esta estrategia permite una instalación controlada y repetible, aunque actualmente la falta de scripts automatizados o documentación detallada de instalación, i.e., configuración de puertos, variables de entorno, volúmenes de datos, entre otros, puede dificultar el proceso para nuevos usuarios o desarrolladores no familiarizados con el sistema. Si bien la instalación es posible y funcional, aún puede ser mejorada.

MineGuard no ha sido diseñado específicamente con reemplazabilidad como objetivo, y no existen evidencias de que pueda sustituir o ser sustituido fácilmente por otro sistema equivalente sin modificaciones significativas. Asimismo, no se han adoptado estándares de interoperabilidad que favorezcan la portabilidad de datos o la compatibilidad cruzada con sistemas existentes en la industria minera, por lo tanto, el nivel de reemplazabilidad actual puede considerarse bajo.

4.10 Cumplimiento de cada Criterio

En base al análisis realizado de MineGuard según sus distintos aspectos, se define el cumplimiento de cada criterio del modelo de calidad de producto del ISO 25010 en MineGuard en la Tabla 2. Se priorizará la propuesta de mejoras en áreas donde el cumplimiento sea menor.

Tabla 2. Cumplimiento de criterios en MineGuard
Elaboración propia

Característica	Nivel de Cumplimiento	Justificación	Brechas Identificadas.
Adecuación Funcional	Alto	Cumple de forma completa, correcta y pertinente con los requisitos funcionales definidos; fue validado por usuarios y	No se identifican brechas significativas.



		expertos en contexto real.	
Eficiencia de Desempeño	Medio	Las funcionalidades clave funcionan con tiempos aceptables, pero existen redundancias en WebSocket y mal diseño de queries GraphQL que afectan el rendimiento.	Optimización de consultas y mejora en el manejo de WebSockets para evitar consumo innecesario de recursos.
Compatibilidad	Alto	El uso de Docker, APIs REST/GraphQL, y formatos estándar como JSON garantizan buena interoperabilidad y coexistencia.	No se identifican brechas significativas.
Usabilidad	Medio	La interfaz es intuitiva y agradable, pero hay problemas en accesibilidad, validaciones, y falta de ayuda visual o documentación.	Falta de validaciones adecuadas, accesibilidad limitada y ausencia de asistencia contextual.
Fiabilidad	Bajo	No hay mecanismos de recuperación, manejo de errores inconsistente y falta de monitoreo; algunas fallas frecuentes afectan funciones críticas.	Ausencia de pruebas automatizadas, registro de errores y monitoreo de fallos; falta de tolerancia a fallos.
Seguridad	Bajo	Vulnerabilidades graves en autenticación, exposición de datos, falta de logs y manejo inseguro de tokens comprometen la seguridad del sistema.	Manejo inseguro de sesiones, carencia de logs seguros, control deficiente de permisos y sesiones activas.
Mantenibilidad	Bajo	Código acoplado, documentación deficiente y ausencia de pruebas automatizadas dificultan la modificación, análisis y testeo.	Falta de modularización, escasa documentación técnica, y ausencia de herramientas de testing y CI/CD.
Portabilidad	Medio	Buena contenedorización y uso de tecnologías web modernas, pero hay fallas en diseño responsivo y falta de pruebas sistemáticas.	Diseño responsivo incompleto y ausencia de pruebas en múltiples entornos operativos o configuraciones.

5 Propuesta de Mejoras

Teniendo en cuenta el análisis realizado en relación a la importancia y cumplimiento de los distintos criterios del modelo de calidad de producto del ISO 25010 para MineGuard, se identificó que los criterios de Fiabilidad, Seguridad y Mantenibilidad presentan la combinación más crítica: alta o máxima importancia para el sistema, junto a un nivel de cumplimiento bajo, por lo que las mejoras a proponer en esta sección se enfocarán en estas características.

5.1 Implantar un Sistema de Logging y Monitoreo

Herramientas como Winston, en el lado de Node.js, o structlog, en el lado de Django, permiten gestionar el registro de eventos y mensajes de manera flexible para una aplicación, incluyendo funcionalidades de personalización de logs como marcas de tiempo, identificadores, nombres de archivos, entre otros; sistemas de niveles de seguridad como *error*, *info*, *debug*, etc.; adición de metadatos contextuales como la dirección IP de una petición; configuración de destino de los logs, entre otras funcionalidades. El formato estructurado recomendable sería JSON, teniendo en cuenta el amplio uso actual de este en MineGuard.

El formateo de logs a través de estas herramientas permite y facilita la interacción con sistemas de gestión de logs como el basado en Grafana y Loki, donde Loki actúa como un sistema de agregación y almacenamiento de logs, indexando los logs a través de etiquetas asociadas a estos, logrando mayor eficiencia y mejores costos de almacenamiento, y Grafana como una plataforma de visualización y análisis de datos, donde los logs pueden ser consultados y visualizados de forma eficiente a través de su propio lenguaje de consulta.

De esta forma se obtiene un flujo de información donde los eventos son registrados y adaptados a un formato específico para las necesidades de MineGuard, posteriormente enviados a Loki, que actuará como la plataforma centralizada de almacenamiento de logs, y finalmente serán visualizados a través de la interfaz provista por Grafana, culminando así en la capacidad de poder almacenar los registros estructurados y etiquetados, y de poder monitorear de manera efectiva el sistema. En la Figura 4 se muestra la interfaz de Grafana con un ejemplo de demostración de Loki, obtenido del entorno de pruebas online Grafana Play.

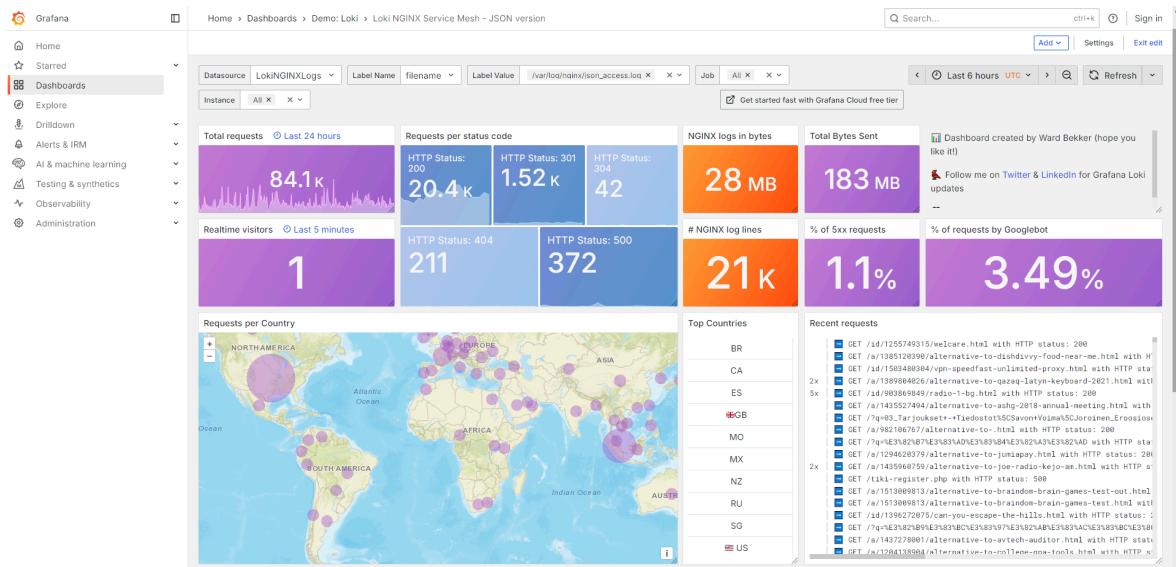


Figura 4. Interfaz de Grafana con un ejemplo de demostración de Loki

Elaborado por Ward Bekker, obtenido desde Grafana Play

5.2 Estrategia de Pruebas Automatizadas y CI/CD

La Integración Continua (CI) es una práctica fundamental en el desarrollo de software donde los desarrolladores fusionan regularmente sus cambios de código en un repositorio centralizado. Este proceso se acompaña de la ejecución automática de construcciones y pruebas, lo que permite una detección temprana de problemas de integración. La Entrega Continua (CD), por su parte, extiende la CI al automatizar la preparación y el despliegue de los cambios de código a entornos de producción o de prueba.



Una estrategia integral de pruebas automatizadas junto con un sistema de Integración y Entrega Continua (CI/CD) constituye un pilar fundamental para asegurar la calidad y estabilidad de sistemas de software modernos; en el caso de MineGuard, esta estrategia resulta especialmente relevante dado su necesidad de monitoreo activo y seguridad informática, donde cualquier error en producción podría comprometer seriamente la funcionalidad o la integridad del sistema. La automatización de pruebas permite detectar errores de forma temprana, evitar regresiones funcionales y reducir el error humano junto con las validaciones manuales, mientras que el CI/CD facilita un flujo constante, seguro y controlado de cambios hacia los entornos de producción.

La implementación propuesta se estructura en torno a tres niveles de prueba: pruebas unitarias, pruebas de integración y pruebas *end-to-end* (de extremo a extremo). Las pruebas unitarias validan el comportamiento de componentes individuales, como funciones o clases, mientras que las pruebas de integración comprueban el correcto funcionamiento de módulos que interactúan, por ejemplo, entre el backend (Django) y la base de datos. Finalmente, las pruebas *end-to-end* verifican flujos completos desde la perspectiva del usuario final en la interfaz web. Para la ejecución de estas pruebas se pueden emplear herramientas como Pytest, para el backend en Python; Jest, para el backend en Javascript; y Cypress o Playwright, para la automatización de pruebas de interfaz gráfica.

En paralelo, se propone implementar un *pipeline* de CI/CD mediante plataformas como GitHub Actions, o Jenkins, el cual se active automáticamente con cada confirmación de cambios (*commit*) en el repositorio. Este pipeline debe incluir etapas de instalación de dependencias, ejecución de pruebas, análisis estático del código (usando herramientas como SonarQube o ESLint, el cual ya se utiliza en el frontend), y despliegue automatizado en entornos de preproducción o producción. De esta manera, cada nuevo cambio es validado de forma sistemática antes de su incorporación definitiva al sistema, lo que aumenta significativamente la fiabilidad de MineGuard y mejora su mantenibilidad, al contar con una serie de pruebas que actúan como documentación viva del comportamiento esperado. La Figura 5 muestra un ejemplo en Jenkins de cómo se vería el estado del pipeline habiendo ocurrido un error en las pruebas de integración; cada nodo representa una etapa del pipeline, las cuales contienen una serie de tareas a ejecutarse de forma automática.

5.3 Fortalecimiento de los Mecanismos de Autenticación y Autorización

Una de las áreas críticas para la seguridad de cualquier sistema de software es la gestión de los mecanismos de autenticación y autorización, es decir, verificación de identidad y control de acceso a recursos. Como ya fue comentado en la sección 3, en el caso de MineGuard, sistema que puede ser utilizado en entornos sensibles como la operación minera en tiempo real, este aspecto cobra particular relevancia, y, según el análisis realizado en la sección 4, la Seguridad es un apartado que se encuentra en un estado altamente carente en la aplicación; el fortalecimiento de estos mecanismos es una mejora que apunta directamente a esta característica.

Una mejora prioritaria es la incorporación de autenticación multifactor (MFA), especialmente para usuarios con roles administrativos o acceso a funciones críticas. Este mecanismo añade una capa adicional al requerir un segundo factor de verificación, como puede ser un código temporal (TOTP) desde una aplicación móvil como Google Authenticator, además de la contraseña. La implementación de MFA representa un refuerzo significativo en la seguridad del sistema al prevenir accesos no autorizados, incluso en casos de ataques por robo de credenciales, además de ser un recurso útil en contra de técnicas de ingeniería social como el Phishing. La Figura 6 muestra un bosquejo del inicio de sesión con autenticación de dos factores en MineGuard.

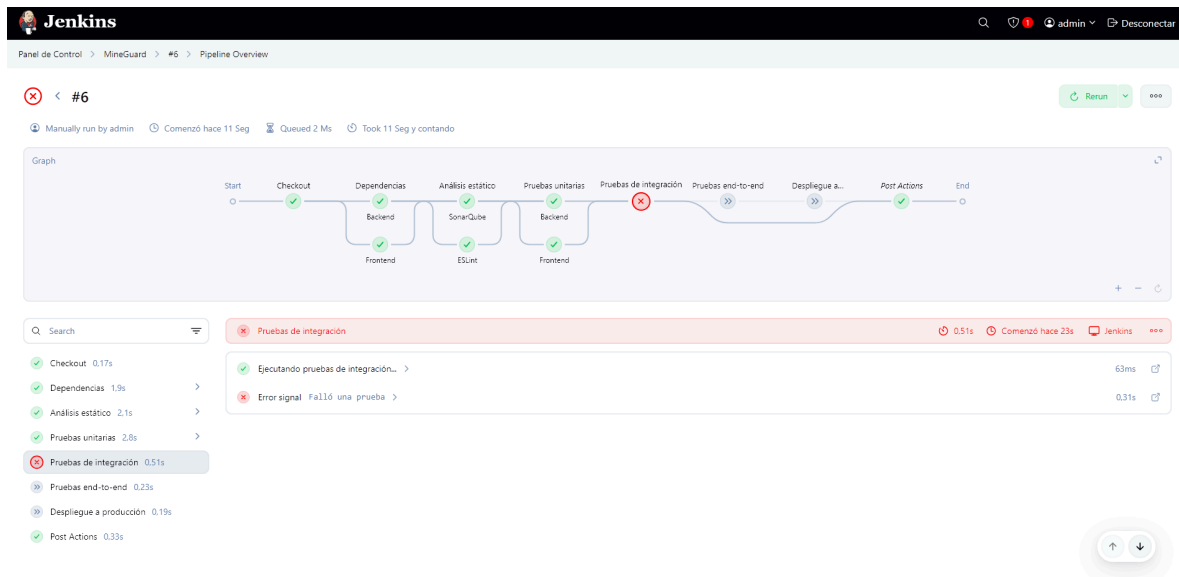


Figura 5. Vista de Ejecución de un Pipeline de Jenkins con Error en las Pruebas de Integración

Elaboración propia

Otra medida clave es la implementación de un modelo de control de acceso basado en roles (RBAC). Esta estrategia permite definir roles preestablecidos, por ejemplo, administrador, operador, auditor, entre otros, con permisos diferenciados, de modo que los usuarios sólo accedan a las funcionalidades que les corresponden según su perfil. Esta técnica simplifica la gestión de permisos, mejora la mantenibilidad del sistema y reduce los errores de configuración que podrían exponer funciones no autorizadas.

Idealmente, el backend debe validar todos los permisos de forma estricta, evitando confiar únicamente en restricciones del lado del cliente; considerando el estado actual de MineGuard, es vital modificar el sistema de tal forma que no exista ninguna parte de la API sin verificación de autenticación y autorización. Se deben proteger todos los puntos de entrada al sistema (*endpoints*) mediante middleware o filtros de autorización en el backend, garantizando una validación robusta incluso en presencia de manipulaciones del cliente, junto a medidas adicionales como la protección de cookies, con atributos como Secure, HttpOnly y SameSite.

Finalmente, se recomienda complementar estas medidas con un sistema de trazabilidad que registre los eventos de acceso, cambios críticos en la configuración y acciones relevantes dentro del sistema, como pueden ser los cambios de umbrales en las alertas. Esta estrategia de registro de eventos puede complementarse de forma sinérgica con la infraestructura de logging y monitoreo descrita anteriormente, donde herramientas como Loki y Grafana permiten no solo visualizar eventos técnicos del sistema, sino también integrar registros relacionados con la seguridad. Mediante el uso de formatos estructurados (como JSON) y etiquetas adecuadas, es posible complementar el seguimiento operativo con la trazabilidad de acciones relevantes, maximizando así la utilidad de la plataforma de monitoreo.

5.4 Otras Propuestas de Mejora

Si bien los aspectos de Fiabilidad, Seguridad y Mantenibilidad presentan la combinación más crítica en cuanto a importancia y cumplimiento, se identificaron oportunidades de mejora relevantes en otras características del modelo de calidad de producto del ISO 25010. A continuación, se proponen acciones concretas para elevar el nivel de calidad general de MineGuard.

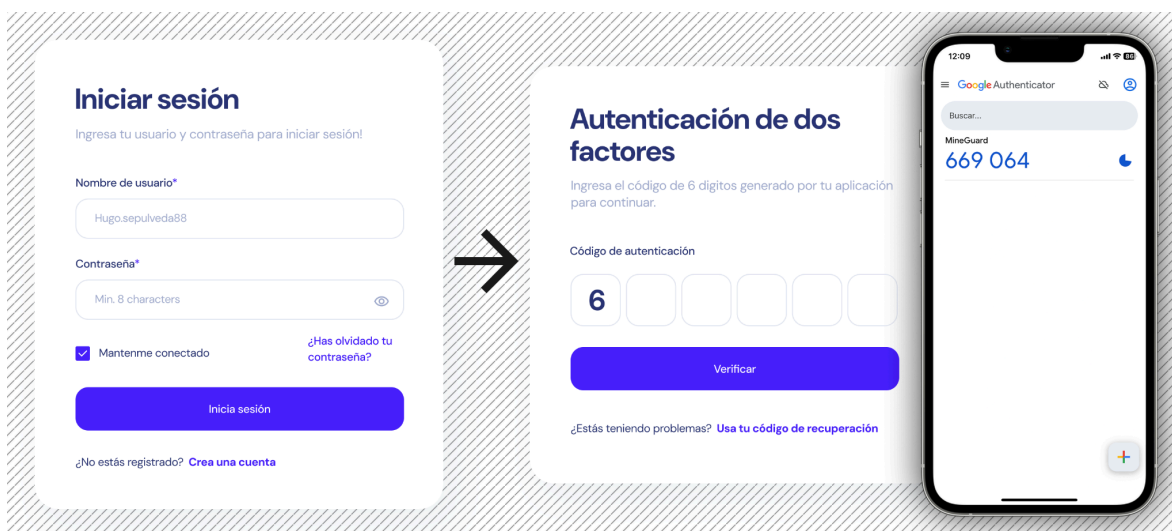


Figura 6. Bosquejo de Inicio de Sesión con Autenticación de dos Factores

Elaboración propia

Eficiencia de Desempeño

Para mejorar la eficiencia, especialmente en el módulo de monitoreo, se sugiere rediseñar el WebSocket para reutilizar mensajes idénticos entre múltiples clientes en lugar de generar flujos independientes. Asimismo, revisar y refactorizar las queries de GraphQL bajo las buenas prácticas declarativas permitiría reducir llamadas redundantes y mejorar el caching del cliente. También se recomienda realizar pruebas de carga controladas para validar el comportamiento bajo diferentes niveles de uso concurrente.

Usabilidad

Para abordar las brechas detectadas, se sugiere implementar mecanismos básicos de accesibilidad, como el uso de etiquetas ARIA, soporte para navegación por teclado, contrastes adecuados, y agregar ayudas visuales como tooltips, tutoriales interactivos o una guía de inicio rápido. También se recomienda simplificar flujos complejos como la generación de reportes mediante filtros avanzados y selección múltiple, y mejorar la validación y retroalimentación de formularios para una experiencia más robusta y confiable.

Portabilidad

A fin de fortalecer la adaptabilidad del sistema, se propone realizar pruebas formales en distintas combinaciones de navegadores, resoluciones de pantalla y entornos operativos. Asimismo, se recomienda documentar completamente el entorno de ejecución, incluyendo variables de entorno, puertos y dependencias, y generar scripts de instalación automatizados con distintas configuraciones, i.e., de desarrollo, producción, entre otras, para facilitar el despliegue en nuevos entornos.

6 Conclusiones

A través del análisis cualitativo de la relación entre MineGuard y la industria minera, y del análisis de los distintos aspectos de MineGuard, tanto técnicos como no técnicos, se logró identificar las características de

calidad más relevantes para MineGuard en base al modelo de calidad de producto del ISO 25010, además de las brechas existentes en la aplicación, sentando las bases para proponer mejoras específicas que optimicen su robustez y desempeño en el entorno minero. Se propusieron mejoras concretas para elevar la calidad de MineGuard, siendo las principales: la incorporación de un sistema de logging y monitoreo basado en herramientas como Loki y Grafana, que permita una observación más eficaz del comportamiento de la aplicación; la implementación de una estrategia de pruebas automatizadas junto con un pipeline de integración y entrega continua (CI/CD) para aumentar la fiabilidad del sistema; y el fortalecimiento de los mecanismos de autenticación y autorización, con medidas como la autenticación multifactor (MFA), el control de acceso basado en roles (RBAC) y la trazabilidad de acciones sensibles. Las mejoras propuestas tienen un enfoque principal en los criterios con las brechas identificadas como más críticas, por lo que se espera que la introducción de estas, junto a las mejoras generales propuestas para otros criterios menos críticos, puedan ayudar a que la aplicación eleve su nivel de calidad para alcanzar un nivel acorde con las exigencias de la industria minera.

El trabajo realizado tiene como principal alcance el análisis desde una perspectiva de calidad de producto, sin abordar en profundidad aspectos de calidad en uso o experiencia del usuario en terreno. Como línea futura, sería valioso complementar esta evaluación con estudios centrados en la experiencia de los usuarios finales como operadores, supervisores y mantenedores, para integrar la dimensión de calidad en uso. El alcance del trabajo se centró en un análisis de alto nivel, sin ejecutar pruebas de rendimiento cuantitativas ni validar las propuestas en un entorno productivo, i.e., en terreno; si bien esto limita la generalización de algunos hallazgos, los resultados obtenidos son valiosos como diagnóstico inicial y como base para futuras iteraciones de desarrollo o evaluación. Asimismo, sería útil profundizar en el análisis cuantitativo del desempeño del sistema en condiciones reales, mediante métricas de fiabilidad, latencia, carga y escalabilidad.

Por otro lado, la retroalimentación recibida durante la Feria de Software fue altamente positiva por parte del público asistente, mostrando alto interés en aplicaciones de este tipo y en su importancia con relación al contexto de la industria minera. De la mano de esto, más allá del aporte directo a MineGuard, el trabajo realizado en esta memoria puede ser de gran utilidad para futuras aplicaciones parecidas o proyectos similares.

Referencias

- [1] Cardemil Winkler, M. (2023). Impactos socioeconómicos de la minería en Chile. Biblioteca del Congreso Nacional de Chile.
- [2] Comisión Chilena del Cobre; Monsalve, R., Donoso, F. (2022). Medición de Encadenamientos productivos de la industria minera en Chile.
- [3] Banco Central. (2024). Boletín Estadístico 7 de septiembre 2024. www.bcentral.cl
- [4] Velasco Balazs, R. A. (2017) Análisis de la calidad de una aplicación para el desarrollo de un plan de mejora de acuerdo a las normas iso [Tesis de pregrado]. Universidad Técnica Federico Santa María.
- [5] Jara Almendra, C. A. (2024) Transformación Digital en el manejo de peticiones dentro del Dpto. de Informática en Direcciones Comunes de Salud [Tesis de pregrado]. Universidad Técnica Federico Santa María.
- [6] ISO, IEC, & IEEE. (2017). *Systems and software engineering — Vocabulary* (ISO/IEC/IEEE 24765:2017). <https://www.iso.org/standard/71952.html>
- [7] Sommerville, I. (2011). *Software engineering* (9th ed.). Addison-Wesley.
- [8] Krasner, H. (2021). The cost of poor software quality in the US: A 2020 report. *Proc. Consortium Inf. Softw. QualityTM (CISQTM)*, 2, 3.
- [9] McCall, J. A., Richards, P. K., & Walters, G. F. (1977). *Factors in software quality* (Vols. I–III). General Electric.
- [10] Miguel, J. P., Mauricio, D., & Rodríguez, G. (2014). A review of software quality models for the evaluation of software products. arXiv preprint arXiv:1412.2977.
- [11] ISO & IEC. (2001). *Software engineering — Product quality — Part 1: Quality model* (ISO/IEC 9126-1:2001).
- [12] ISO & IEC. (2011). *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models* (ISO/IEC 25010:2011). <https://www.iso.org/standard/35733.html>
- [13] Sánchez, F., & Hartlieb, P. (2020). Innovation in the mining industry: Technological trends and a case study of the challenges of disruptive innovation. *Mining, Metallurgy & Exploration*, 37(5), 1385-1399.
- [14] Rojas, L., Peña, Á., & García, J. (2025). AI-Driven Predictive Maintenance in Mining: A systematic literature review on fault detection, digital twins, and intelligent asset management. *Applied Sciences*, 15(6), 3337.



- [15] Barkat Ullah, A., Ma, W., Ahmed, M., Rashid, B., Saeed, M. A., Arshad, O., & Raghav, U. (2025). A comprehensive review of cyber security and current practices in global mining critical infrastructure. *Journal of Cyber Security Technology*, 1-27.
- [16] Twilio. (s.f.). *The WhatsApp Business Platform with Twilio: Best Practices and FAQs*. <https://www.twilio.com/docs/whatsapp/best-practices-and-faqs>
- [17] OWASP. (s.f.). *JSON Web Token (JWT) for Java Cheat Sheet*. OWASP Cheat Sheet Series. https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_for_Java_Cheat_Sheet.html
- [18] OWASP. (s.f.). *HttpOnly*. OWASP Foundation. <https://owasp.org/www-community/HttpOnly>