UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingeniería

Desarrollo de una Aplicación Web Progresiva para la gestión y monitoreo de servicios, dirigida a ingenieros de fiabilidad del sitio

Juan Francisco Cisneros Guzmán

Ingeniería en Ciencias de la Computación

Trabajo de Titulación presentado como requisito para la obtención del título de Ingeniero en Ciencias de la Computación

Quito, 13 de diciembre del 2024

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingeniería

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Desarrollo de una Aplicación Web Progresiva para la gestión y monitoreo de servicios, dirigida a ingenieros de fiabilidad del sitio

Juan Francisco Cisneros

Nombre del profesor, Título académico **Alejandro Proaño, PhD**

Quito, 13 de diciembre del 2024

© DERECHOS DE AUTOR

Nombres y apellidos: Juan Francisco Cisneros Guzmán

Código: 00323665

Cédula de identidad: 1725793804

Lugar y fecha: Quito, 13 de diciembre de 2024

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

El presente trabajo aborda el desarrollo de un sistema de gestión integral que permite a los ingenieros de confiabilidad de sitios (SRE por sus iniciales en inglés) detectar fallos y posibles ataques en tiempo real, mediante la asistencia de inteligencia artificial (IA). La motivación principal radica en la necesidad de una herramienta robusta que garantice la continuidad de los servicios. Se busca minimizar los tiempos de inactividad y maximizar la disponibilidad del software. Se emplearon técnicas de aprendizaje automático y una arquitectura de microservicios para mejorar la precisión y eficiencia en la detección de incidentes. Se ha desarrollado una aplicación web progresiva (PWA por sus iniciales en inglés) y una interfaz de programación de aplicaciones (API por sus siglas en inglés) para brindar accesibilidad y escalabilidad. Las conclusiones señalan que el avance en esta área podría transformar significativamente la gestión de los SRE, desarrolladores y otros ingenieros involucrados en el mantenimiento, brindando mayor estabilidad a largo plazo de los sistemas.

Palabras clave: ingeniería en confiabilidad del sitio, inteligencia artificial, detección de fallos, aprendizaje automático, microservicios, monitoreo en tiempo real, gestión de incidentes, aplicación web progresiva, API.

ABSTRACT

This work addresses the development of an integrated management system that enables Site Reliability Engineers (SRE) to detect failures and potential attacks in real-time with the assistance of Artificial Intelligence (AI). The main motivation lies in the need for a robust tool to ensure service continuity. The goal is to minimize downtime and maximize software availability. Machine learning techniques and a microservices architecture were employed to enhance the accuracy and efficiency of incident detection. A Progressive Web App (PWA) and an Application Programming Interface(API) were developed to provide accessibility and scalability. The conclusions indicate that advancements in this area could significantly transform the management of SREs, developers, and stakeholders, offering greater long-term stability for systems.

Key words: site reliability engineering, artificial intelligence, failure detection, machine learning, microservices, real-time monitoring, incident management, progressive web application, API.

TABLA DE CONTENIDOS

INTRODUCCIÓN **10**
Reflexión Teórica 11
Proyectos relacionados 13
Grafana. 13
PagerDuty. 14
Jira. 15
Justificación del proyecto 15

IMPLEMENTACIÓN **16**
Arquitectura de la Aplicación 18
Aplicación Web Progresiva - PWA 21
Modelo de inteligencia artificial y notificaciones push 26
Interfaz de programación de aplicaciones (API) 26
Autenticación y seguridad 28

VERSIONAMIENTO **28**

PRIVACIDAD Y PROTECCIÓN DE DATOS **29**

CONCLUSIONES **30**
Antes y Después de la aplicación 30
Ventajas de usar la aplicación 31
Ventajas de usar el API 31
Aporte hacia la comunidad SRE 31

TRABAJO FUTURO **32**
Nuevos módulos 32
Autenticación Institucional 32
Mejoras en la seguridad y privacidad de datos 33

REFERENCIAS **34**

ANEXOS **36**
ANEXO 1: MANUAL DE USUARIO DE LA PWA 36
ANEXO 2: MANUAL DE USUARIO DE LA API 36
ANEXO 3: REPOSITORIO GITHUB DE LA PWA 36
ANEXO 4: REPOSITORIO GITHUB DE LA API 36
ANEXO 6: DESPLIEGUE A PRODUCCIÓN DE LA API 36
ANEXO 7: TECNOLOGÍAS UTILIZADAS 36
ANEXO 8: POLÍTICAS DE PRIVACIDAD DE TERCEROS 37

ANEXO 9: GITHUB PROJECT, SCRUM AGILE 38

TABLA DE FIGURAS

FIGURA 1: ARQUITECTURA GENERAL DEL PROYECTO 18

FIGURA 2: ARQUITECTURA FUNCIONAL DE LA BASE DE DATOS 20

FIGURA 3: ESQUEMA DE LA BASE DE DATOS 20

FIGURA 5: ARQUITECTURA GENERAL DE LA REST API 27

FIGURA 6: ARQUITECTURA API REST EXPRESS IMPLEMENTADA 27

FIGURA 4: AUTENTICACIÓN CON FIREBASE AUTHENTICATION 28

INTRODUCCIÓN

"Las pruebas de aseguramiento de calidad han evolucionado considerablemente desde sus inicios en la década de 1930 con el ciclo PDSA de Walter Shewhart. Inicialmente arraigadas en la manufactura, gigantes tecnológicos como IBM ayudaron a adaptar estos principios a proyectos tecnológicos" [1]

El aseguramiento de calidad (QA) tradicional se centra en la identificación y prevención de errores antes de que los sistemas o productos lleguen al entorno de producción. Este enfoque permite a los equipos detectar defectos en las etapas tempranas del desarrollo, minimizando costos y riesgos. Sin embargo, a medida que los sistemas tecnológicos se han vuelto más complejos, interconectados y dependientes de múltiples componentes, este enfoque muestra limitaciones. Las pruebas previas al despliegue no siempre pueden prever todas las posibles fallas que puedan surgir en entornos reales, donde factores como la escalabilidad, la interacción entre servicios y la carga inesperada pueden generar problemas imprevistos.

En este contexto, nació la ingeniería de fiabilidad del Sitio (SRE), la cual representa una nueva forma de abordar la confiabilidad y la escalabilidad en sistemas complejos. Mientras que el QA tradicional se enfocaba en prevenir errores antes del despliegue, los SRE llevan esta mentalidad un paso más allá, extendiendo las prácticas de QA hacia la operación en producción.

En la actualidad, los Ingenieros de Fiabilidad del Sitio (SRE) desempeñan un papel crucial en garantizar el correcto funcionamiento, estabilidad y disponibilidad de los sistemas tecnológicos. Estos profesionales se enfrentan a desafíos complejos que incluyen la detección y resolución de errores, la implementación de mejoras en la infraestructura y el mantenimiento de estándares de rendimiento. Sin embargo, las herramientas que utilizan a menudo no están diseñadas específicamente para sus necesidades, lo que dificulta su labor diaria.

DevProbe es el nombre del aplicativo web, una aplicación que permite el monitoreo y detección de errores en conjunto con la intervención de inteligencia artificial. Se proporciona así una herramienta completa para la gestión de los SRE.

Este proyecto se desarrolló bajo la metodología ágil Scrum, utilizando GitHub Projects para la gestión de tareas, el proyecto se encuentra en el anexo 9. Entre los principales requerimientos funcionales, se incluyen la creación y despliegue de la aplicación con Ionic Framework (como una Aplicación Web Progresiva) y Express JS (para el API) en Firebase Hosting y Render.com, respectivamente, se ha realizado además una integración con GitHub Actions para despliegues automáticos.

Se implementaron diversas pantallas y funcionalidades, incluyendo autenticación con Firebase para el registro/inicio de sesión de usuarios, gestión de equipos, creación de productos, pruebas de latencia y traceroute con la API de RIPE Atlas, y conexión con Gemini Vertex como modelo de inteligencia artificial. Además, se desarrollaron herramientas para monitorear métricas del hardware corriendo en el sistema como: CPU y memoria. Creación de pruebas unitarias, de integración y de sistema, pruebas de carga usando la librería Artillery, un manejador de incidentes y notificaciones push para alertar a los usuarios de cambios en la aplicación, todos los módulos se detallan más adelante.

En este proyecto se detallará y justificará la creación de esta aplicación, además de la creación de un API, de esta manera estos ingenieros podrán enviar datos directamente al aplicativo en caso de que lo requieran.

Reflexión Teórica

La ingeniería en fiabilidad del sitio (Site Reliability Engineering, SRE), creada en 2003 por Ben Treynor, surgió como una práctica innovadora en Google para abordar los desafíos del mantenimiento y la mejora continua de sistemas tecnológicos a gran escala. Treynor definió a un SRE como “un programador que pasa el 50% de su tiempo dedicado a resolver problemas

de operaciones y el otro 50% a mejorar la fiabilidad del sistema" [1]. Esta definición subraya la intersección entre el desarrollo de software y las operaciones, marcando un cambio respecto al modelo tradicional en el que ambos roles solían estar separados.

Por un lado, los ingenieros de testing, se enfocan en garantizar la calidad del software antes de su despliegue en producción. Su principal tarea es identificar y reportar errores en la etapa de desarrollo, utilizando herramientas y técnicas como pruebas funcionales, de integración y de sistema. Aunque desempeñan un papel crucial, su alcance termina una vez que el producto es entregado al entorno de producción.

Por otro lado, los Ingenieros de Fiabilidad del Sitio son especialistas que tienen como objetivo principal asegurar que los sistemas y servicios críticos se mantengan confiables, disponibles y escalables a medida que crecen. Como explica Google, la empresa que acuñó este rol," SRE es lo que sucede cuando se encarga a un ingeniero de software lo que tradicionalmente sería un trabajo de un ingeniero de operaciones" [3].Los SREs aplican principios de ingeniería de software para manejar tareas de operación, como la automatización de despliegues, la monitorización proactiva y la respuesta a incidentes, con el fin de minimizar la intervención humana y maximizar la eficiencia del sistema.

Según Niall Murphy, autor de Site Reliability Engineering, "un SRE debe buscar constantemente formas de automatizar tareas repetitivas y manuales" [4] para evitar errores humanos y garantizar la consistencia en los procesos. Esto implica implementar herramientas de monitoreo avanzadas y desarrollar soluciones personalizadas que permitan detectar y resolver problemas de manera proactiva antes de que afecten al usuario final.

La metodología SRE también enfatiza la automatización como medio para reducir el trabajo operativo manual y mejorar la consistencia. Los SREs buscan identificar tareas

repetitivas y propensas a errores, automatizándolas para liberar tiempo y enfocarse en la resolución de problemas complejos y en mejoras continuas.

Los SRE adoptan una visión pragmática de la fiabilidad, no buscan eliminar todos los errores, sino minimizar su impacto en el usuario. Esto se alinea con la "tolerancia a fallos", una premisa que establece límites en la cantidad de errores permitidos y que da margen a la innovación al no exigir perfección absoluta. Como sostiene Google en su guía oficial sobre SRE, "aceptar fallos como parte de la operación permite que los sistemas evolucionen y se adapten" [6]. Con estos pilares de medición de fiabilidad, automatización y tolerancia a fallos, el SRE se convierte en un modelo que equilibra la estabilidad y el cambio constante, habilitando a las organizaciones a mantener la competitividad sin comprometer la experiencia del usuario.

Proyectos relacionados

En esta sección se ha decidido realizar una investigación sobre posibles aplicativos similares a la aplicación DevProbe. Estas son las herramientas utilizadas actualmente por los ingenieros de fiabilidad del sitio.

Grafana.

Grafana es un software de tipo open source para el monitoreo y análisis de datos en tiempo real. Es utilizado por grandes empresas como fuente central de análisis de servidores y servicios debido a su versatilidad y amplia aceptación de fuentes de datos.

DevProbe, por otro lado, está diseñado específicamente para las necesidades de los SRE. Además de monitorear, DevProbe incluye en todos sus módulos análisis basados en inteligencia artificial, herramientas para realizar pruebas de fiabilidad y funcionalidades que permiten no solo detectar errores, sino también optimizar y resolver problemas en un flujo integral.

PagerDuty.

PegerDuty es un software de paga, utilizado como manejador de incidentes, los SRE utilizan este software principalmente cuando se detecta cualquier tipo de incidente en producción, de esta manera se puede monitorear el estado del problema, asignar miembros para la resolución de este y mantener la disponibilidad que ofrecen a sus clientes. Entre las características más usadas se encuentran las notificaciones en tiempo real a miembros del equipo y la integración de un chat por incidente, esta funcionalidad PagerDuty la llama coordinación en tiempo real (on-call), permitiendo contactar con un ingeniero automáticamente por varios medios.

PagerDuty se centra exclusivamente en la gestión de incidentes y la comunicación durante los mismos. DevProbe por otro lado amplía este alcance al incluir capacidades adicionales como la generación de análisis post-mortem con inteligencia artificial.

Un análisis post-mortem es un proceso de evaluación detallada que se realiza después de resolver un incidente en sistemas, aplicaciones o servicios en producción. Su objetivo principal es identificar las causas raíz del problema, documentar las acciones realizadas durante la gestión del incidente, y extraer aprendizajes clave para prevenir que se repita. Este análisis suele incluir una cronología del evento, un resumen de los impactos, las medidas correctivas implementadas, y recomendaciones para fortalecer la infraestructura o los procesos.

DevProbe entonces permite automatizar el proceso de documentación, detectar patrones repetitivos de fallos y generar sugerencias proactivas para mitigar riesgos futuros. Además, DevProbe incluye notificaciones push personalizadas, mejorando la experiencia de los equipos SRE al proporcionar información en tiempo real de manera más accesible.

Jira.

Jira es de los softwares más completos, es un software basado en la metodología Agile. La ventaja que ofrece Jira es que permite realizar manejo de proyectos, realizar testing y monitorear incidentes en el ámbito de bugs, tareas e historias de usuario. De esta forma los SRE se encargan de poder realizar una entrega de software de calidad.

Jira tiene un enfoque que está más orientado hacia la planificación, testing y seguimiento de historias de usuario, bugs y tareas. DevProbe, en cambio, está especializado en las operaciones críticas de los SRE, permitiendo asignar y realizar pruebas de software, pero basar esas pruebas en un producto gracias a su módulo "Model The Product", mismo que se detalla en próximas secciones.

Justificación del proyecto

DevProbe es un aplicativo que se distingue por ofrecer una solución integral diseñada específicamente para las necesidades de los Ingenieros de Fiabilidad del Sitio (SRE). Su diseño incorpora módulos avanzados que abordan las cuatro principales funciones de un SRE: monitorear, automatizar, mejorar y resolver problemas.

DevProbe incluye un módulo de monitoreo en tiempo real, capaz de registrar métricas clave como la latencia, el rendimiento y el estado de los servicios. Estas métricas se visualizan en tableros de control intuitivos que permiten a los SRE identificar anomalías de manera inmediata. Además, su sistema de alertas, basado en inteligencia artificial, prioriza los incidentes críticos, ayudando a los equipos a concentrarse en resolver problemas que realmente afectan la disponibilidad y la experiencia del usuario.

DevProbe integra herramientas para la automatización de tareas repetitivas, como ayuda de escalabilidad de servicios en función de la carga, envió y análisis de resultados y la ejecución de pruebas de software. Esto reduce la carga operativa de los SRE y les permite enfocarse en actividades más estratégicas.

Uno de los puntos clave de DevProbe es su capacidad para medir continuamente la latencia y enrutamiento de la red. Esto permite a los SRE ejecutar pruebas A/B para evaluar mejoras en los servicios, comparando métricas antes y después de implementar cambios. Un SRE puede modificar un servicio de backend y, mediante DevProbe, medir si existe una mejora en comparación con la versión anterior, optimizando así la experiencia del usuario.

DevProbe facilita la resolución de problemas al implementar análisis post-mortem inteligentes, basados en la metodología SRE de Google. Este módulo recopila automáticamente datos relacionados con un incidente, los organiza en un informe detallado y utiliza algoritmos de IA para identificar patrones recurrentes. Esto no solo acelera la identificación de la causa raíz, sino que también sugiere medidas preventivas para evitar incidentes similares en el futuro.

En la actualidad, no existe una única aplicación que reúna todas estas capacidades en un solo lugar. Por ello, DevProbe surge como una solución innovadora que simplifica el trabajo de los Ingenieros de Fiabilidad del Sitio, ofreciéndoles una herramienta que cubre todas sus necesidades operativas y estratégicas.

IMPLEMENTACIÓN

Se ha implementado un aplicativo web progresivo que permita a los ingenieros de fiabilidad del sitio monitorear sus servicios de manera eficiente. El aplicativo incluye herramientas y funcionalidades esenciales para las actividades diarias de los SRE, además de la capacidad para compartir y colaborar en equipo en tiempo real.

Se usa un aplicativo web progresivo debido a su capacidad para ofrecer una experiencia de usuario rápida y responsiva en cualquier dispositivo o entorno. Los PWA combinan las ventajas de las aplicaciones web y móviles, permitiendo a los ingenieros de fiabilidad del sitio acceder a la plataforma sin necesidad de instalaciones complejas, e incluso cuando están desconectados, gracias a su funcionalidad de almacenamiento en caché.

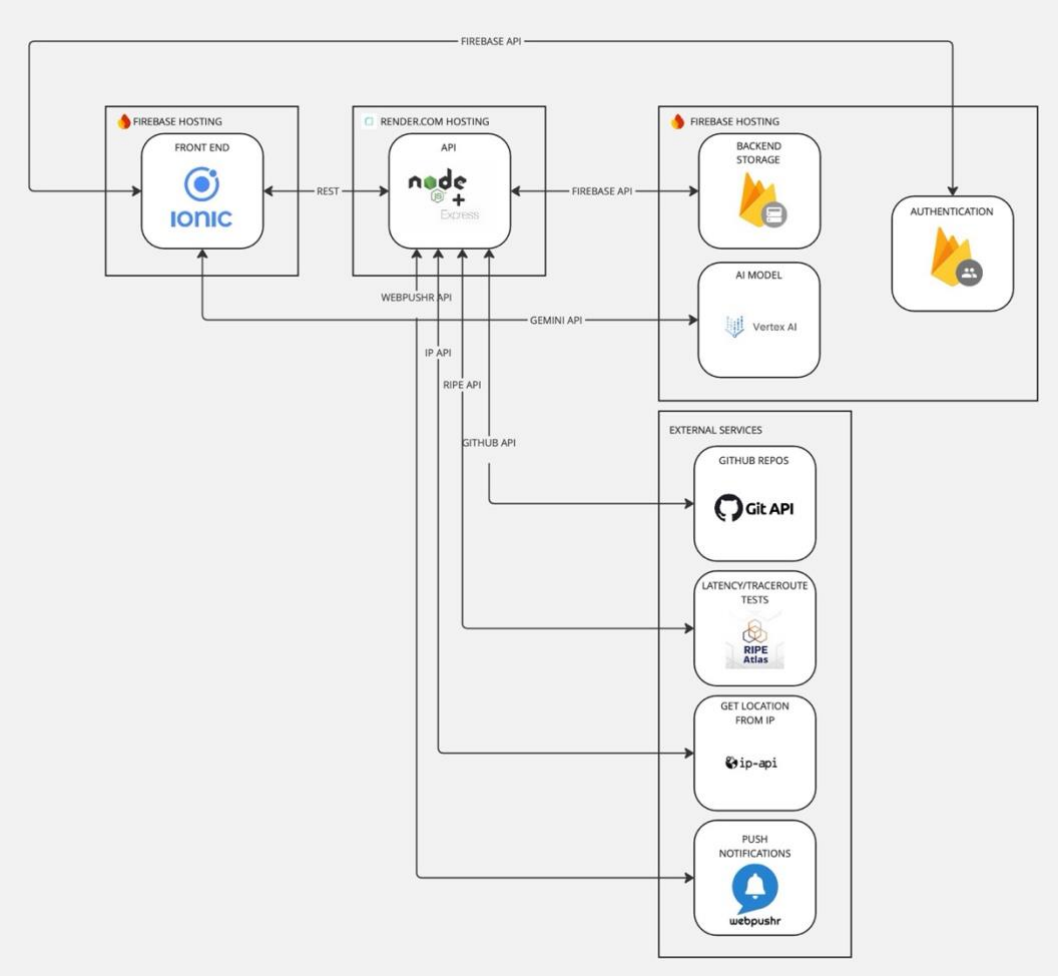
El diseño es 100% responsivo, optimizado para su uso en una amplia variedad de tamaños de pantalla, desde dispositivos móviles hasta pantallas de escritorio, garantizando así una experiencia de usuario fluida y adaptable. Además, se ha implementado un servicio de notificaciones push, que permite alertar a los ingenieros de fiabilidad del sitio sobre cambios que ocurren en ese momento.

Se ha tomado en cuenta en todo momento la escalabilidad, la disponibilidad, rendimiento y mantenimiento de la aplicación, así como del API. En términos de escalabilidad, la aplicación es capaz de manejar un gran número de servicios monitoreados y usuarios concurrentes sin afectar el rendimiento. Todo esto manteniendo la disponibilidad de la aplicación, la cual es alta para no afectar al día a día del SRE y permitir al ingeniero trabajar las 24 horas al día los 7 días de la semana. De igual forma, se ha tomado en cuenta el rendimiento de la aplicación, desplegar un aplicativo web implica que los navegadores aún no pueden entregar la máxima capacidad del sistema en donde se ejecutan.

Finalmente, este aplicativo funciona independientemente del producto que se requiere monitorear, si el producto o sistema principal falla, esta aplicación debe recolectar la información necesaria para que el ingeniero de fiabilidad del sitio pueda resolver la incidencia.

Arquitectura de la Aplicación

FIGURA 1: Arquitectura General del Proyecto



Se ha decidido implementar una arquitectura basada en el modelo de cliente servidor, esta arquitectura moderna se encuentra en su totalidad implementada en la nube. A continuación, se detallan los componentes claves:

La arquitectura de la aplicación presentada combina diferentes tecnologías para implementar una solución modular y escalable. En el frontend, se utiliza Ionic, una plataforma hospedada en Firebase Hosting que interactúa con una API REST para enviar y recibir datos. El frontend se comunica con el backend, desarrollado en Node.js con Express, el cual está

alojado en Render.com. Esta separación entre la interfaz y la lógica de negocio permite una gestión más clara y una mayor flexibilidad al implementar nuevas funcionalidades.

El backend está conectado con diversos servicios y APIs, tanto internas como externas. Por un lado, interactúa con Firebase para acceder al almacenamiento y a los servicios de autenticación. Se emplea Vertex AI para el modelo de inteligencia artificial, que está integrado para realizar análisis avanzados, tales como la detección de anomalías o predicciones sobre los datos. Por otro lado, se consumen servicios externos como GitHub API para gestionar repositorios, RIPE Atlas para pruebas de latencia y trazado de rutas, IP-API para obtener la ubicación geográfica de direcciones IP y Webpushr para manejar notificaciones push, integrando funcionalidades avanzadas de comunicación y monitoreo. Todos estos servicios se detallan más adelante a profundidad.

Firebase Hosting no solo aloja el frontend, sino que también se utiliza para proporcionar autenticación de usuarios. Esta arquitectura aprovecha la escalabilidad de Firebase para almacenar datos y manejar la autenticación, mientras que Render.com se usa para el despliegue del backend. La integración de servicios externos y APIs especializadas asegura que la aplicación pueda ofrecer capacidades avanzadas de monitoreo, notificaciones y análisis, todo gestionado desde un backend centralizado que actúa como intermediario entre el cliente y los servicios.

Para la base de datos se decidió utilizar el servicio de Firebase Firestore, esta es una base de datos tipo documental o también conocida como una base de datos NoSQL. La ventaja principal de utilizar este tipo de bases de datos es la velocidad de acceso a los datos. Además de la habilidad de guardar y obtener objetos tipo JSON nativamente.

Se ha requerido el uso de este tipo de base de datos principalmente por el volumen alto de mediciones que se van a almacenar a corto y largo plazo. Estas mediciones esperadas por

empresa podrían ser realizadas en intervalos de minutos o segundos si el equipo SRE así lo requiere. Es por esto por lo que el acceso a los datos debe ser inmediato, y esta base NoSQL nos permite justamente que los usuarios de la aplicación obtengan y visualicen sus mediciones al instante.

FIGURA 2: Arquitectura funcional de la base de datos

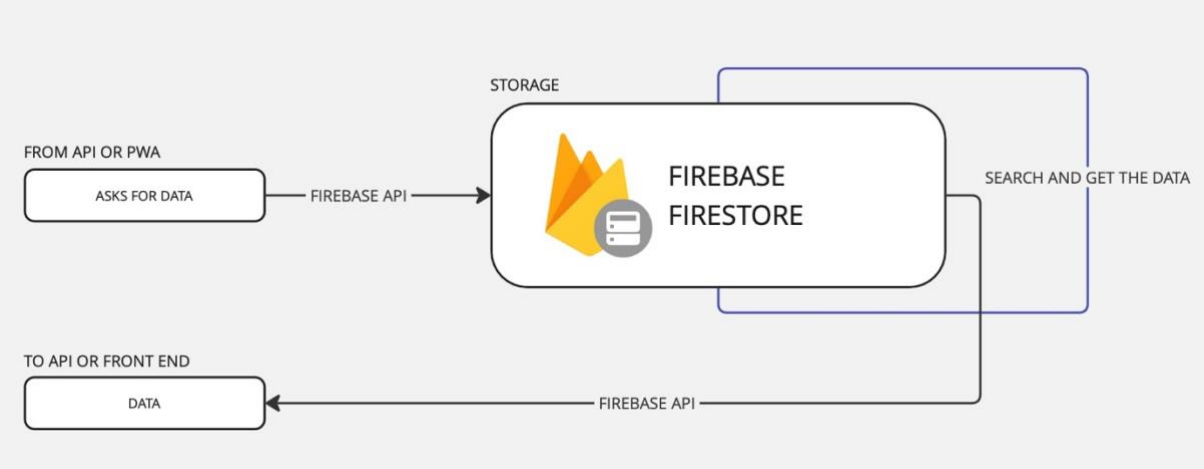
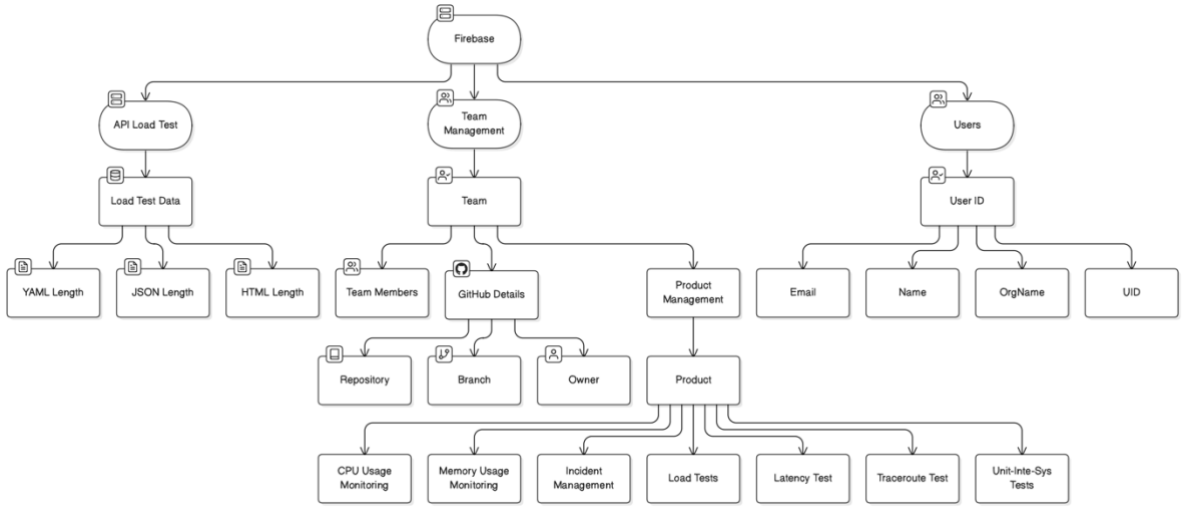


FIGURA 3: Esquema de la base de datos



El esquema de base de datos definido incluye las colecciones API, USERS y TEAMS. La colección API documenta los estados principales para el funcionamiento de la API Express. Mientras que la colección USERS, permite que al registrar un usuario con Firebase Authentication, se cree automáticamente un documento que vincula al usuario a un equipo y

copia sus datos (correo, nombre, uid) en Firebase Storage, sin almacenar contraseñas. Finalmente, la colección TEAMS guarda todas las mediciones y datos por equipo, esta colección contiene varias sub-colecciones que permiten la organización de estas mediciones.

El beneficio principal de esta arquitectura es la comunicación cliente-servidor donde el Frontend (Ionic) realiza peticiones a la API REST Express o API´s terceras, las mismas que responden con los datos almacenados y ya previamente procesados que vienen de Firebase Firestore.

Aplicación Web Progresiva - PWA

En esta sección se describen los módulos principales de la Aplicación Web Progresiva. En esta primera versión, la aplicación incluye 9 módulos esenciales que cubren las funcionalidades principales, además de un módulo adicional para mostrar analíticas y otro dedicado a la configuración de la aplicación, se detallan los módulos principales y su funcionamiento.

El módulo **“Model The Product”** permite a los ingenieros modelar un producto en la PWA, lo cual es esencial para entender y facilitar los objetivos de los usuarios finales. Basándose en la metodología de ingeniería de fiabilidad del sitio, el modelo de producto incluye “user objectives” (objetivos de usuario), “product steps” (pasos necesarios), “product services” (servicios requeridos), y “product SLOs” (Service Level Objectives). A través de la ruta /new-product, se puede crear un producto, y el módulo proporciona rutas adicionales (/model-product y /view-product) para gestionar y visualizar los productos. Este proceso conecta con Firestore para realizar operaciones CRUD, siendo la base para el monitoreo y mantenimiento del producto en la aplicación, el aplicativo requiere de al menos un producto para su funcionamiento.

La PWA permite la colaboración entre ingenieros de fiabilidad del sitio a través de una pantalla de gestión de equipos o módulo “**My Team**”, donde los miembros pueden agregar o

revocar acceso a otros. Todos los miembros de un equipo pueden realizar, visualizar, crear y ejecutar acciones en el aplicativo. En esta primera versión, todos los miembros tienen los mismos permisos, sin roles administrativos ni restricciones por módulos. Futuras versiones introducirán controles de permisos y roles para una gestión más granular, además de auditoría de acciones. La ruta /myteam gestiona esta pantalla, utilizando el servicio TeamsService que permite actualizar y eliminar miembros. Este servicio depende de AuthService y Firebase Authentication para validar la identidad de los usuarios mediante autenticación por correo electrónico.

Para monitorear la velocidad de sus productos, los ingenieros de fiabilidad realizan pruebas de latencia y traceroute, estas pruebas se realizan en los módulos "**Latency Test**" o "**Traceroute Test**" respectivamente. Los módulos simulan el uso de un servicio desde ubicaciones externas para detectar problemas de conectividad y enrutamiento en conjunto con el análisis automatizado de IA, se usa el modelo Gemini 1.5 flash para señalar inconsistencias de enrutamiento y permitir consultas sobre el análisis. "RIPE Atlas es una plataforma de medición activa, es decir, se basa en sensores de medida desplegados sobre Internet." [6] Estos sensores permiten realizar mediciones de latencias y traceroutes desde un probe o anchor (dispositivos proporcionados por RIPE NCC) localizado en cualquier parte del mundo hacia un servidor, una página web o a un servicio que cuente con una conexión a internet.

RIPE es un servicio abierto al público y es posible no incurrir en gastos en caso de que el usuario colabore en la red de mediciones. En este caso la aplicación si incurre en gastos adicionales por medición de latencia o traceroute realizada, pero para el desarrollo de esta se han utilizado los créditos gratuitos proporcionados a cuentas nuevas.

Con RIPE Atlas, se miden latencias y rutas de red desde diversas ubicaciones, ayudando a mantener los SLOs. Las pantallas permiten configurar las pruebas, incluyendo el URL, país de

origen y una descripción. Los resultados se visualizan en tablas, mapas, y gráficos que muestran promedios diarios por país. IP-API, es una API que nos permite obtener la localización de una dirección IP. Esta API funciona en conjunto con la API de RIPE Atlas permitiendo obtener no solo el país o región de donde se realiza la medición, pero a su vez la ciudad desde donde se realizan las pruebas de latencia y de tipo traceroutes. IP-API permite obtener los datos sin costos adicionales, se comienza a incurrir en gastos cuando se excede el volumen de peticiones de su capa gratuita, misma que durante el desarrollo de la PWA no se excedieron.

Para que los ingenieros de fiabilidad (SRE) puedan monitorear el estado del hardware y los servicios en ejecución de forma detallada y eficiente, se han desarrollado módulos especializados dentro de la PWA. Los módulos "**CPU Usage**" y "**Memory Usage**" permiten a los usuarios visualizar y comparar métricas actuales y pasadas en tiempo real, facilitando el análisis y la resolución de problemas relacionados con el desempeño del sistema.

La visualización de estas métricas se realiza mediante Flame Graphs, una herramienta avanzada que representa el uso de recursos de manera jerárquica. En el contexto de estos gráficos, cada bloque en el diseño invertido tipo "icicle" corresponde a una función que está utilizando CPU o memoria en el sistema. Los niveles más bajos reflejan las llamadas más recientes, mientras que los superiores representan funciones padres que generaron esas llamadas. Esto proporciona una visión clara y detallada del stack de ejecución a nivel de kernel, mostrando todas las funciones involucradas en el consumo de recursos. El uso de Flame Graphs es crucial no solo para monitorear el estado actual del sistema, sino también para identificar patrones ineficientes en el código o rastrear problemas complejos como fugas de memoria o cuellos de botella en el uso de la CPU. Por ejemplo, un ingeniero puede analizar qué funciones están consumiendo una cantidad desproporcionada de CPU o memoria y tomar decisiones informadas para optimizar el desempeño del software. Además, estos gráficos permiten detectar problemas que, de otro modo, podrían pasar desapercibidos en sistemas grandes o

altamente dinámicos. Así mismo, la inteligencia artificial complementa esta funcionalidad analizando automáticamente las métricas recolectadas. Este análisis proporciona insights útiles, como identificar tendencias anómalas en el uso de recursos o alertar sobre posibles problemas de rendimiento antes de que afecten a los usuarios finales.

Por otro lado, el módulo "**Load Testing"** está diseñado para que los ingenieros de fiabilidad del sitio (SREs) puedan realizar pruebas de carga en sistemas, identificando sus límites en condiciones de alta demanda. Se utiliza la librería Artillery para generar usuarios virtuales que simulan tráfico, poniendo el sistema a prueba hasta que falla o supera los desafíos planteados. La API procesa los resultados, los transforma en métricas clave, como estados HTTP (200, 300, 400 y 500) con sus tiempos de respuesta, y los almacena en Firestore. Estos datos se visualizan gráficamente en la PWA, facilitando su análisis. Además, el módulo integra inteligencia artificial para interpretar los resultados y notifica al usuario mediante una notificación push la finalización de la prueba, esto debido a que estas pruebas suelen tomar un tiempo considerable. Tal como lo menciona el manual de fiabilidad del sitio de Google, los ingenieros deben entender los límites de sus sistemas para poder actuar rápidamente ante situaciones de estrés o fallos inesperados. Este módulo no solo permite identificar los puntos críticos de rendimiento, sino que también ofrece una base para optimizar recursos y mejorar la resiliencia de los sistemas.

En este caso se ha decidido agregar un módulo extra, los ingenieros en fiabilidad del sitio requieren prevenir los incidentes y tal como se menciona en el libro Site Reliability Engineering de Google "Si no lo has probado, asume que está roto", Google hace énfasis en su libro a la realización de pruebas básicas como las unitarias, de integración y de sistema.

El módulo de "**Software Testing**" permite a los ingenieros de fiabilidad del sitio realizar pruebas unitarias, de integración y de sistema, asegurando actualizaciones y despliegues

seguros. Basado en el manual SRE de Google, este módulo facilita la reducción de incertidumbre en sistemas mediante pruebas exhaustivas. Las pruebas unitarias evalúan funciones y clases individuales; las de integración prueban componentes ensamblados; y las de sistema validan la funcionalidad completa en un entorno de producción simulado.

GitAPI es el API oficial de GitHub, permite al aplicativo obtener datos sobre los repositorios de código que usa una organización, además de archivos específicos de esos repositorios. Esta API es en su mayoría de uso gratuito y se comienzan a realizar cobros cuando existe un volumen alto de peticiones a la misma. Se ha decido implementar esta conexión debido a que la misma en conjunto con Gemini nos permite realizar pruebas unitarias del código y de integración de manera autónoma, el ingeniero SRE es capaz de elegir el archivo específico del cuál se requiere realizar una prueba y el modelo de IA escribe el código del test. Las pruebas se pueden asignar a testers con notificaciones push, y el API permite actualizar los resultados, habilitando la automatización de pruebas.

Finalmente, el módulo "**Incident Management"** ayuda a gestionar y resolver incidentes, enfocándose en mitigar el impacto y restaurar el servicio rápidamente. Inspirado en la interfaz de Jira y la metodología de Google, se permite asignar miembros del equipo a roles clave como Incident Commander, Operations Leader y Communication Leader, quienes coordinan los esfuerzos para minimizar el impacto del incidente. Cada incidente debe pasar por los estados abierto, cerrado y postmortem además se asigna niveles de urgencia, y categorías de impacto por incidente. Finalmente, la cultura postmortem de Google permite analizar lo ocurrido, respondiendo a las preguntas: What went wrong? (¿Qué salió mal?), What can be learned? (¿Qué se puede aprender?) y How to prevent similar incidents in the future? (¿Cómo prevenir incidentes similares en el futuro?). Además, el modelo de IA, Gemini, asiste al SRE en esta etapa de análisis post-incidente, entregando

Modelo de inteligencia artificial y notificaciones push

A lo largo del documento se ha mencionado cómo la inteligencia artificial analiza y colabora con los ingenieros de fiabilidad del sitio además de cómo en todo momento los ingenieros reciben actualizaciones en tiempo real, gracias a las notificaciones push.

El modelo de IA electo es Gemini 1.5 Flash, la familia de modelos de Gemini es considerada multimodal, se refiere a que podemos no solamente enviar texto plano, pero a su vez imágenes, videos y hasta PDFs. [7] Para este proyecto se ha decidido implementar la familia Gemini 1.5 Flash, modelo que permite tiempos de respuesta menores.

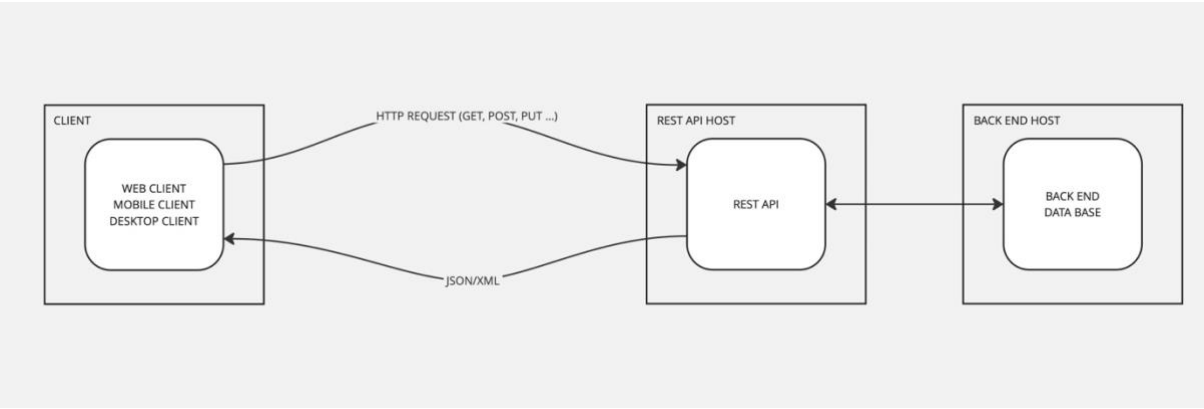
El modelo Flash cuenta con soporte para 1 millón de tokens lo que le permite entender una cantidad de datos superior a otros modelos. Este modelo tal como Google lo menciona "Está específicamente diseñado alto volumen de datos y aplicativos rentables" [7]. En este caso debido a que se enviaran los datos sin modificaciones para el análisis al modelo, es necesario que el mismo pueda analizar correctamente grandes cantidades de información y al mismo tiempo no incurrir en gastos innecesarios.

Finalmente, Webpushr es un servicio de tipo REST, el mismo actúa como un servidor Web Push. Webpushr mediante verbos HTTP nos permite que la API REST Express envié notificaciones a usuarios suscritos, de esta manera nos evitamos el desarrollo del servidor Web Push. Webpushr permite que los usuarios se suscriban o de suscriban correctamente mediante llaves VAPID a sus servidores Push.

Interfaz de programación de aplicaciones (API)

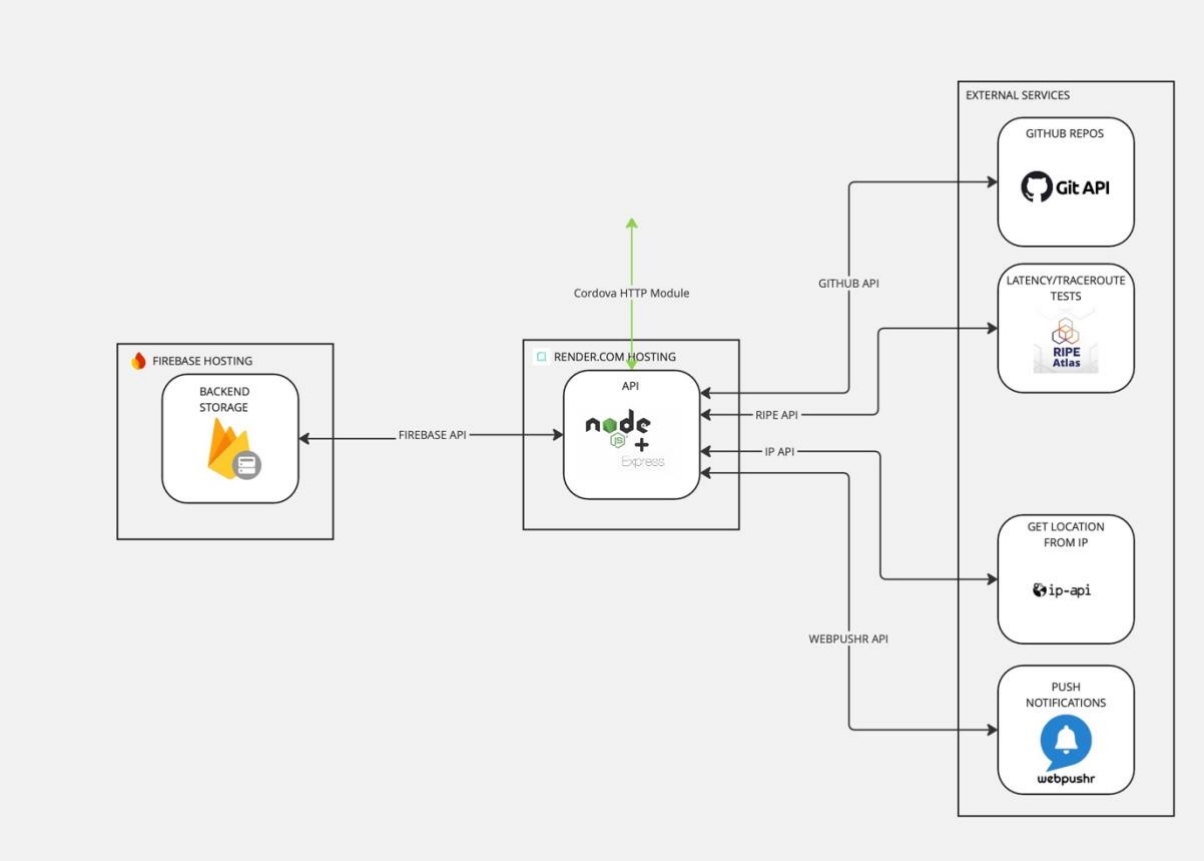
Para este proyecto se ha decido utilizar la arquitectura API REST, REST proviene de transferencia de estado representacional, la misma es la más popular, y la que la mayoría de las webs utilizan para intercambiar datos. "El cliente envía las solicitudes al servidor. El servidor utiliza esta entrada del cliente para iniciar funciones internas y devuelve los datos de salida al cliente." [9]

FIGURA 5: Arquitectura general de la REST API



Debido a que las PWA se basan en la web, se ha decidido implementar una REST API con Express, la misma que comunica al cliente con los servicios de Firebase, pero a su vez la REST API en este proyecto es el enlace principal desde y hacia servicios externos como la API de GitHub, la API de RIPE Atlas, IP-API y hasta el proveedor gratuito para notificaciones tipo push, Webpushr, servicios ya mencionados anteriormente.

FIGURA 6: Arquitectura API REST Express implementada

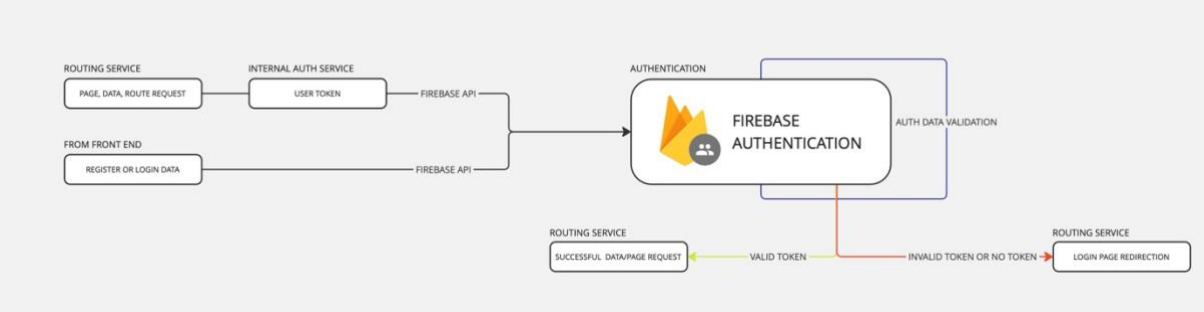


Autenticación y seguridad

Este proyecto maneja y almacena datos empresariales, nombres, correos y contraseñas personales que no pueden ser accedidas por malos actores. Es por esto por lo que se ha optado por implementar un servicio de autenticación de parte del Frontend y Backend/API, además de que este servicio protege y restringe la modificación de la base de datos Firestore.

El servicio de autenticación a utilizar es el de Firebase Authentication, el mismo permite a nuevos y antiguos usuarios realizar la tarea de registro y autenticación contra el PWA y el API REST Express.

FIGURA 4: Autenticación con Firebase Authentication



VERSIONAMIENTO

El versionamiento de un aplicativo es un proceso clave que permite organizar y gestionar las actualizaciones y cambios de un proyecto de software. En este caso se ha decidido implementar versionamiento tanto para el API como para la PWA, utilizando repositorios de GitHub, mismos que se encuentran adjuntos en los anexos de este documento.

Usando el esquema de "**Versión: 0.0.0"**, se facilita el seguimiento de modificaciones y el despliegue de nuevas funcionalidades o correcciones. En este caso se entregará la Versión 1.11.240 de la PWA y la Versión 1.10.30 de la API Express.

La primera cifra representa la versión principal o "major". Generalmente, este número solo se incrementa cuando hay un cambio significativo o una evolución importante en el aplicativo, en este caso es el primer despliegue a producción de la aplicación y de API.

La segunda cifra indica la versión secundaria o “minor”. Este número se incrementa cada vez que se implementa una nueva funcionalidad relevante, pero que no altera drásticamente la estructura del aplicativo. En este caso para la PWA, 11 debido a que existen 11 módulos en la aplicación y 10 para la API debido a que son el número de rutas implementadas.

La última cifra corresponde al número de “build” o de “commit”. Este número se incrementa con cada cambio o corrección que se hace en la rama principal del proyecto. En el caso de la PWA 240 commits en la rama main y 30 commits a la rama main del API.

Este sistema de versionamiento fue y es crucial para mantener la estabilidad del aplicativo y garantizar que todos los cambios se gestionen de forma controlada. Así mismo, se han implementado políticas de control en GitHub para que solo se puedan realizar cambios a la rama principal si previamente se ha realizado un pull request.

PRIVACIDAD Y PROTECCIÓN DE DATOS

El manejo de los datos de este aplicativo protege la intimidad y los datos personales de los individuos y empresas. El aplicativo no permite el acceso a los datos, mediciones, nombres, emails ni contraseñas a terceros. Es por esto por lo que junto con Firebase Authentication salvaguardamos y protegemos los datos de todos los usuarios de la PWA y API.

Al trabajar con implementaciones de terceros, el aplicativo no recibe ni envía datos sensibles a estas empresas. Los datos que se intercambian con terceros están limitados estrictamente a información técnica necesaria para realizar pruebas de rendimiento o monitoreo, sin involucrar nombres, emails, contraseñas o información de identificación personal. Esto garantiza que los datos personales de los usuarios y empresas estén protegidos, enviando la información mínima y necesaria permitiendo al aplicativo cumplir su función de

análisis sin comprometer la privacidad. Todas las comunicaciones se realizan a través de canales seguros y cifrados.

Por otro lado, el modelo de inteligencia artificial realiza los análisis sin necesidad de almacenar información privada o sensible, todas las políticas de uso de datos del modelo se pueden encontrar en el anexo 8 al igual que las políticas de uso de datos de todas las integraciones terceras.

Adicional, todos los resultados de los análisis y mediciones se almacenan en Firebase bajo altos estándares de seguridad proporcionados por Google LLC, el acceso a la base de datos está estrictamente controlado por los administradores del sistema.

Finalmente, la implementación de estas prácticas de privacidad permite que el aplicativo ofrezca funcionalidades avanzadas sin comprometer la integridad de los datos personales ni la confianza de los usuarios, quienes pueden sentirse seguros de que su información permanece privada y protegida en todo momento.

CONCLUSIONES

Antes y Después de la aplicación

Antes de implementar la aplicación, los equipos SRE dependían de herramientas de monitoreo y análisis fragmentadas, lo que dificultaba la identificación y resolución de fallos de manera oportuna. El seguimiento de métricas y la detección de anomalías eran procesos manuales y reactivos, lo que aumentaba el riesgo de tiempos de inactividad y afectaba la experiencia del usuario final. Con la introducción de esta PWA, la gestión de confiabilidad se vuelve más centralizada, automatizada y proactiva. La aplicación permite al equipo de SRE detectar, diagnosticar y mitigar problemas de manera certera y eficiente, lo que reduce tiempos de inactividad y mejora la eficiencia operativa.

Ventajas de usar la aplicación

La aplicación facilita la centralización de datos, al recopilar métricas y organizarlas en un solo lugar, permitiendo al equipo de SRE tomar decisiones informadas de manera más eficiente y efectiva.

Otra ventaja significativa es la detección de anomalías manualmente o atuomatizadas con IA. La aplicación puede identificar patrones inusuales y notificarñas al equipo de SRE, permitiendo una respuesta temprana y una rápida.

Por último, la optimización del rendimiento es otra ventaja fundamental. La aplicación ofrece sugerencias específicas para mejorar el uso de recursos, lo cual no solo optimiza el rendimiento de las aplicaciones supervisadas, sino que también ayuda a reducir costos operativos al hacer un uso más eficiente de los recursos disponibles.

Ventajas de usar el API

El uso del API ofrece un acceso eficiente a los datos, permitiendo que otras aplicaciones o servicios obtengan la información relevante de manera directa. Esto facilita la integración con terceros agilizando los procesos de obtención de datos.

La escalabilidad y flexibilidad, permite expandir fácilmente las capacidades del aplicativo y conexiones con terceros. Esto asegura que la infraestructura de monitoreo puede adaptarse a las necesidades cambiantes de las organizaciones en futuras versiones.

Finalmente, el API posibilita la automatización y la integración con aplicativos internos de las organizaciones. Gracias a esta funcionalidad, los equipos SRE pueden enviar y recibir datos directamente, sin el uso de la PWA.

Aporte hacia la comunidad SRE

Esta aplicación representa un avance significativo para la comunidad de ingenieros en fiabilidad del sitio, se ha proporcionado un marco robusto para la gestión de confiabilidad en aplicaciones críticas. El aplicativo ayuda en la detección temprana de problemas con respuesta

automatizadas con IA y la centralización de datos. La aplicación no solo mejora la eficiencia operativa, sino que también impulsa una cultura de confiabilidad y proactividad. Esta herramienta contribuye a estandarizar buenas prácticas dentro de la comunidad, beneficiando tanto a ingenieros como a empresas a establecer un nuevo estándar en la gestión de confiabilidad y resiliencia en el entorno digital.

TRABAJO FUTURO

Nuevos módulos

Debido a que el aplicativo se maneja por módulos, como actualizaciones futuras se podrían implementar nuevas funcionalidades. Una de estas funcionalidades sería el módulo Load Balancing se plantea implementar una funcionalidad que permita a los SRE distribuir eficientemente la carga de trabajo entre los recursos disponibles en el sistema.

La visión modular que implementa la aplicación asegura que, a medida que evolucionen las necesidades de los SRE, el aplicativo pueda adaptarse, facilitando la gestión de infraestructuras complejas y distribuidas, anticipándose a futuros desafíos tecnológicos.

Autenticación Institucional

En el módulo actual de autenticación, DevProbe utiliza Firebase Auth con autenticación mediante correo electrónico y contraseña. Sin embargo, se ha considerado mejorar la flexibilidad del sistema implementando con opciones de autenticación que se alineen con entornos institucionales. Una futura funcionalidad sería permitir a los usuarios registrarse e iniciar sesión mediante sus cuentas institucionales.

Con esta mejora, se busca no solo ofrecer una experiencia de inicio de sesión más fluida y conveniente para usuarios que pertenecen a instituciones, sino también fortalecer los protocolos de seguridad y simplificar la gestión de accesos, especialmente en entornos donde la verificación de identidad es crítica.

Mejoras en la seguridad y privacidad de datos

Para fortalecer la confianza de los usuarios y asegurar la integridad de los datos en DevProbe, se prevé implementar mejoras en seguridad y privacidad. Una de las principales propuestas es la adopción de cifrado de extremo a extremo en la transmisión de datos sensibles, asegurando que la información esté protegida tanto en tránsito como en reposo.

Otro aspecto clave de esta mejora es el desarrollo de controles de acceso más específicos y personalizados, que permitan gestionar los permisos de usuarios de forma granular, asegurando que solo las personas autorizadas puedan acceder a datos críticos.

REFERENCIAS

[1]	T. Aamer, «QA Testing History: Its Start and the World's First Tester,» 22 11 2023. [En línea]. Available: https://www.linkedin.com/pulse/qa-testing-history-its-start-worlds-first-tester-taimur-aamer-bxdaf.
[2]	IBM, «Three Differences Between DevOps and SRE,» 23 07 2021. [En línea]. Available: https://www.ibm.com/think/topics/devops-vs-sre. [Último acceso: 2024].
[3]	A. Perry y M. Luebbe, «Testing for Reliability,» Google SRE, 2023. [En línea]. Available: https://sre.google/sre-book/testing-reliability/. [Último acceso: 04 11 2024].
[4]	N. R. Murphy, L. Fong-Jones, B. Beyer, T. Underwood, L. Nolan y D. Rensin, «How SRE Relates to DevOps,» 2018. [En línea]. Available: https://sre.google/workbook/how-sre-relates/. [Último acceso: 12 11 2024].
[5]	B. Treynor, «Introduction,» 2017. [En línea]. Available: https://sre.google/sre-book/part-I-introduction/. [Último acceso: 13 11 2024].
[6]	J. Mace, J. Oertel, S. Thorne, A. Chakrabarti, J. Ma y J. Yang, «Incident Response,» Google SRE, 2018. [En línea]. Available: https://sre.google/workbook/incident-response/. [Último acceso: 06 11 2024].
[7]	Firebase, «Learn about the Gemini models,» 2024. [En línea]. Available: https://firebase.google.com/docs/vertex-ai/gemini-models?hl=en&authuser=0. [Último acceso: 17 10 2024].
[8]	Lacnic, «RIPE Atlas en Latinoamérica y Caribe,» 2020. [En línea]. Available: https://www.lacnic.net/1000/1/lacnic/ripe-atlas-en-latinoamerica-y-caribe#:~:text=RIPE%20Atlas%20es%20una%20plataforma,permiten%20la%20generación%20de%20mediciones.. [Último acceso: 20 10 2024].
[9]	Amazon Web Services, «¿Qué es una interfaz de programación de aplicaciones (API)?,» 2023. [En línea]. Available: https://aws.amazon.com/es/what-is/api/. [Último acceso: 18 10 2024].
[10]	Microsoft, «Get started with Progressive Web Apps,» 27 03 2023. [En línea]. Available: https://learn.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/how-to/. [Último acceso: 16 10 2024].
[11]	IONIC, «Hi, we’re Ionic.,» 2024. [En línea]. Available: https://ionic.io/about. [Último acceso: 16 10 2024].
[12]	Google Cloud, «Innovate faster with enterprise-ready AI, enhanced by Gemini models,» 2024. [En línea]. Available: https://cloud.google.com/vertex-ai?utm_source=google&utm_medium=cpc&utm_campaign=latam-EC-all-es-dr-SKWS-all-all-trial-b-dr-1707800-LUAC0020716&utm_content=text-ad-none-any-DEV_c-CRE_706203535866-ADGP_Hybrid+%7C+SKWS+-+BRO+%7C+Txt_AI+and+ML_Vertex+AI-KWID. [Último acceso: 17 10 2024].
[13]	Learn Microsoft, «Creación de una aplicación web del marco Express,» 2024. [En línea]. Available: https://learn.microsoft.com/es-es/training/modules/build-web-api-nodejs-express/2-create-app. [Último acceso: 18 10 2024].

[14]	P. Magaz, «Montando un servidor de notificaciones Web Push,» 5 4 2019. [En línea]. Available: https://pablomagaz.com/blog/montando-servidor-notificaciones-webpush/#:~:text=Configuración,utilizando%20la%20librería%20web%2Dpush.&text=Estas%20claves%20serán%20necesarias%20para,a%20comenzar%20con%20el%20cliente.. [Último acceso: 20 10 2024].
[15]	C. Crous, R. Parker y H. Victoria, «Product-Focused Reliability for SRE,» Google Site Reliability Engineering , 2023. [En línea]. Available: https://sre.google/resources/practices-and-processes/product-focused-reliability-for-sre/. [Último acceso: 22 10 2024].
[16]	G. Brendan, «Flame Graphs,» 31 10 2020. [En línea]. Available: https://www.brendangregg.com/flamegraphs.html. [Último acceso: 23 10 2024].

ANEXOS

ANEXO 1: MANUAL DE USUARIO DE LA PWA

https://github.com/juanfranciscocis/DevProbe_Tesis/blob/main/README.md

ANEXO 2: MANUAL DE USUARIO DE LA API

https://documenter.getpostman.com/view/28389822/2sAY55bJit

ANEXO 3: REPOSITORIO GITHUB DE LA PWA

https://github.com/juanfranciscocis/DevProbe_Tesis

ANEXO 4: REPOSITORIO GITHUB DE LA API

https://github.com/juanfranciscocis/DevProbeApi

ANEXO 5: DESPLIEGUE A PRODUCCIÓN DE LA PWA

https://devprobe-89481.web.app

ANEXO 6: DESPLIEGUE A PRODUCCIÓN DE LA API

https://devprobeapi.onrender.com

ANEXO 7: TECNOLOGÍAS UTILIZADAS

Con base en la arquitectura utilizada, se realiza una descripción de las tecnologías utilizadas y enlaces a su documentación oficial:

1. **Ionic (Frontend)**:

Framework para construir aplicaciones multiplataforma (web, iOS, Android) usando tecnologías web como HTML, CSS y JavaScript.

Referencia principal: https://ionicframework.com

2. **Node.js + Express (Backend API)**:

Plataforma y framework para construir aplicaciones y APIs del lado del servidor.

Referencia Principal: https://expressjs.com

3. **Firebase Hosting**:

Servicio de Google para hospedar sitios web estáticos y dinámicos con SSL incorporado y alta velocidad.

Referencia Principal: https://firebase.google.com/docs/hosting

4. **Backend Storage (Firestore)**:

Base de datos NoSQL de Firebase utilizada para almacenar datos estructurados de forma flexible y escalable.

Referencia Principal: https://firebase.google.com/docs/firestore

5. **Firebase Authentication**:

Servicio de autenticación que permite gestionar usuarios con email y contraseña, entre otros métodos.

Referencia Principal: https://firebase.google.com/docs/auth

6. **Vertex AI**:

Plataforma de Google Cloud para integrar y entrenar modelos de inteligencia artificial, utilizada aquí para análisis y predicciones avanzadas.

Referencia Principal: https://firebase.google.com/docs/vertex-ai

7. **APIs externas**:

GitHub API: Permite interactuar con repositorios de GitHub, como acceder a información de código.

Referencia Principal: https://docs.github.com/en/rest?apiVersion=2022-11-28

RIPE Atlas API: Herramienta para pruebas de latencia y traceroute para medir el rendimiento de redes.

Referencia Principal: https://atlas.ripe.net

IP-API: Proporciona datos de ubicación basados en direcciones IP.

Referencia Principal: https://ip-api.com

WebPushr API: Servicio para enviar notificaciones push a los usuarios.

Referencia Principal: https://www.webpushr.com

ANEXO 8: POLÍTICAS DE PRIVACIDAD DE TERCEROS

Firebase y Gemini AI: https://firebase.google.com/support/privacy

GitHub: https://docs.github.com/en/site-policy/privacy-policies/github-general-privacy-statement

IP-API: https://ip-api.com/docs/legal

Webpushr: https://www.Webpushr.com/privacy-policy

Ripe Atlas: https://atlas.ripe.net/docs/faq/security-and-privacy.html

ANEXO 9: GITHUB PROJECT, SCRUM AGILE

https://github.com/users/juanfranciscocis/projects/8