
System Resource
Monitoring Dashbood

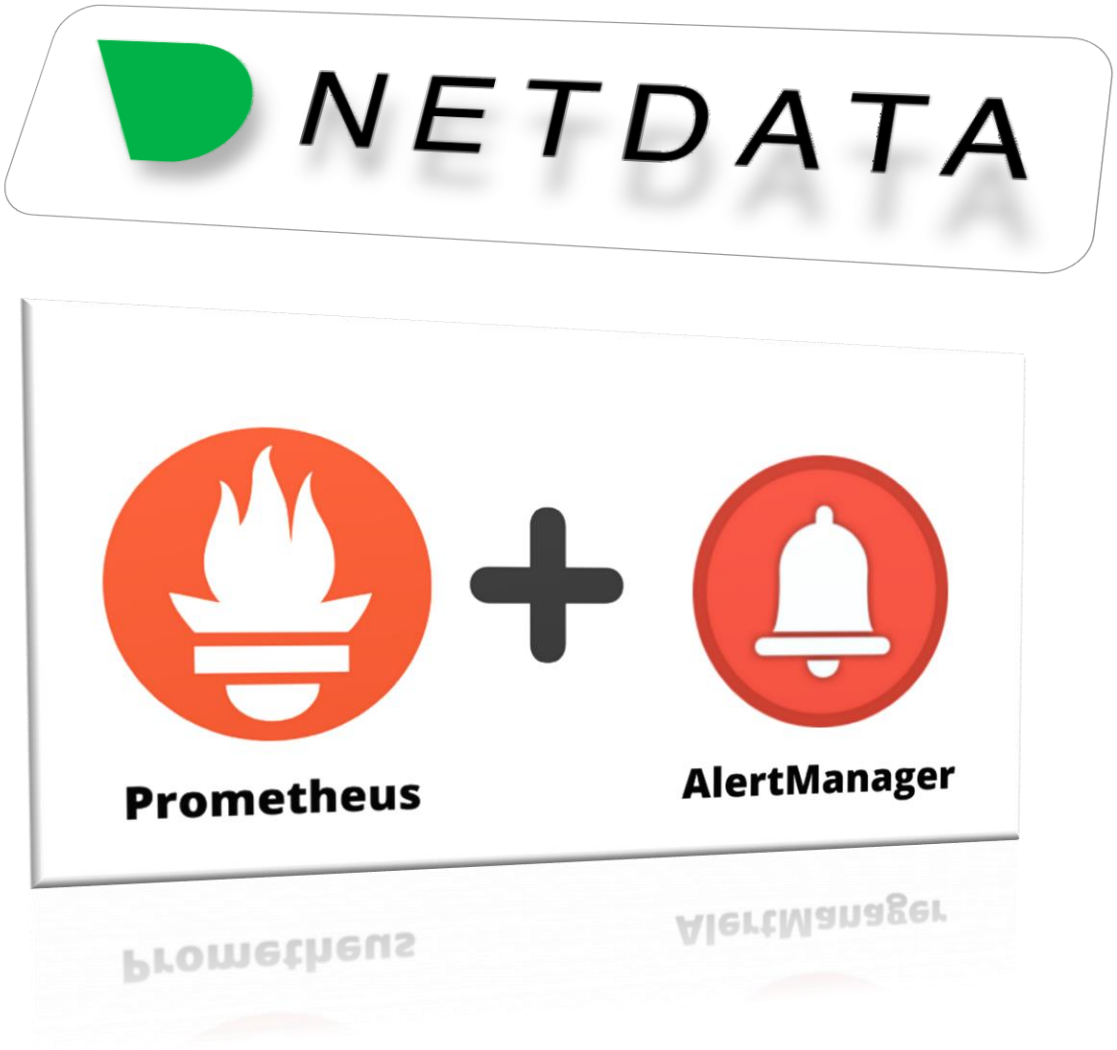
PROYECTO INTEGRADO

Dashboard de Monitoreo de Recursos del Sistema

Autor: Ismael Mariscal Santos

Tutor: Castillo García, Juan Pedro

I.E.S. Francisco Romero Vargas (Jerez de la Frontera)



Administración de Sistemas Informáticos en Red

Curso: 2023/2024

Índice

..... 1

1. Introducción 5

1.1 Introducción 5

1.2 Finalidad 5

2. Objetivos 5

2.1 Medios necesarios 5

2.2 Hardware: 5

2.3 Software: 5

3. Planificación 6

3.1 Estimación del tiempo necesario para cada tarea: 6

3.2 Diagrama de Gantt 6

3.3 Tabla 7

3.4 Realización 8

Paso 1: Planificar el proyecto y buscar información sobre las tecnologías a utilizar 8

Paso 2: Estructura HTML 8

Paso 3: Estilo CSS 8

Paso 4: Gráficos con Chart.js 8

Paso 5: Integración con API de Netdata 8

Paso 6: Actualización de gráficos 8

Paso 7: Sistema de alertas 8

Paso 8: Cambio de tema 9

Paso 9: Pruebas y ajustes finales 9

Paso 10: Documentación del proyecto 9

4. Realización del Proyecto 9

4.1 Trabajos realizados 9

Fase 1: Planificación del Proyecto 9

Fase 2: Diseño de la Estructura HTML 9

..... 10

Fase 3: Creación y Estilo del CSS 10

Fase 4: Implementación de Gráficos con Chart.js 10

Fase 5: Integración de la API de Netdata para Obtener Datos 10
Fase 6: Implementación de la Lógica de Actualización de Gráficos en JavaScript 10
Fase 7: Creación del Sistema de Alertas 11
Fase 8: Pruebas y Ajustes Finales 11
Fase 9: Documentación del Proyecto 11
5. Problemas encontrados 12
6. Modificaciones sobre el proyecto planteado inicialmente 12
7. Posibles mejoras al proyecto 15
8. Bibliografía 15
9. Fuentes del Proyecto 16
10. Anexos 17
10.1 Capturas de pantalla: 17
10.2 Repositorio en GitHub 17
10.3 Documentación del código 17

1. Introducción

1.1 Introducción

El presente proyecto tiene como objetivo crear un dashboard para monitorear en tiempo real los recursos del sistema, tales como la CPU, la RAM y el disco duro. Utilizando tecnologías web, este sistema permitirá visualizar gráficamente el uso de estos recursos y proporcionar alertas cuando ciertos umbrales sean superados.

1.2 Finalidad

La finalidad de este proyecto es facilitar el monitoreo continuo de los recursos del sistema, permitiendo a los administradores de sistemas identificar rápidamente cualquier problema de rendimiento. Al tener una visualización en tiempo real, se pueden tomar decisiones informadas para optimizar el uso de los recursos y mantener el sistema funcionando eficientemente.

2. Objetivos

Los objetivos de este proyecto son:

- Desarrollar un dashboard interactivo que muestre el uso de la CPU, RAM y disco duro.
- Implementar gráficos de tipo "doughnut" y de líneas utilizando Chart.js para la visualización de datos.
- Integrar el dashboard con una API (Netdata) para obtener datos en tiempo real de los recursos del sistema.
- Proporcionar alertas visuales y de texto cuando los recursos superen ciertos umbrales.
- Implementar un sistema de cambio de tema (oscuro y claro) para mejorar la usabilidad del dashboard.

2.1 Medios necesarios

Para la realización de este proyecto se necesita lo siguiente:

2.2 Hardware:

- Un ordenador personal de configuración media.
- Conexión a Internet para acceder a la API de Netdata.

2.3 Software:

- Editor de código (Visual Studio Code).
- Biblioteca Chart.js para la creación de gráficos.
- GitHub para el control de versiones y subir el repositorio
- API de Netdata para la obtención de datos de los recursos del sistema.
- Lenguajes: HTML, CSS, y JavaScript.

- Prometheus, Alertmanger y sus APIs
- Software de diseño grafico como Canva, draw.io excalidraw.com y Genially

---

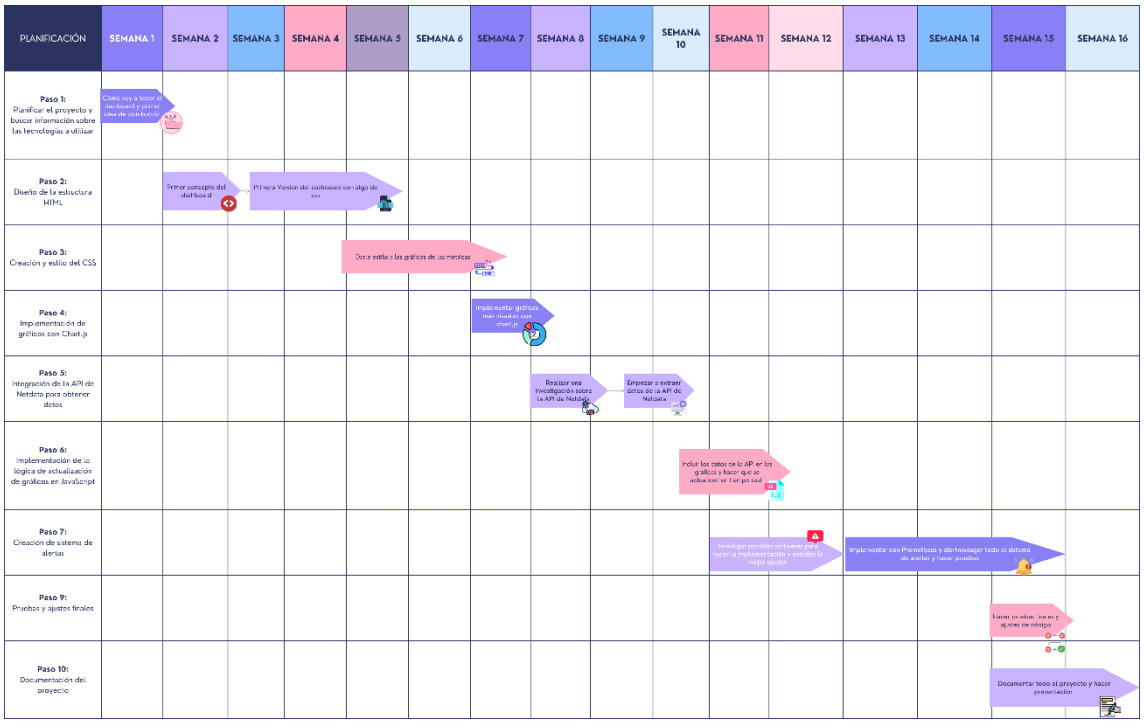
3. Planificación

3.1 Estimación del tiempo necesario para cada tarea:

- Planificación del proyecto: 20 Horas
- Diseño de la estructura HTML: 40 horas.
- Creación y estilo del CSS: 30 horas.
- Implementación de gráficos con Chart.js: 20 horas.
- Integración de la API de Netdata para obtener datos: 30 horas.
- Implementación de la lógica de actualización de gráficos en JavaScript: 10 horas.
- Creación de sistema de alertas: 50 horas.
- Implementación del cambio de tema y css responsive: 5 horas.
- Pruebas y ajustes finales: 10 horas.
- Documentación del proyecto: 20 horas.

Total estimado: 235 horas.

3.2 Diagrama de Gantt



Ubicación: Proyecto/Documents/img/Gantt.png

3.3 Tabla

PASOS	TAREA	DESCRIPCIÓN	SEMANAS	PERFIL	COSTE POR HORA	HORAS	COSTE TOTAL
1	Planificar el proyecto y buscar información sobre las tecnologías a utilizar	Identificar tecnologías y realizar planificación inicial	1	Gestor de Proyecto	€30	20	€600
2	Estructura HTML	Crear una página HTML con contenedores para los gráficos de CPU, RAM y disco	2	Desarrollador Front-end	€30	40	€1200
3	Estilo CSS	Definir estilos para el dashboard, incluyendo la disposición de gráficos y cambio de tema	3 - 4	Desarrollador Front-end	€30	30	€900
4	Gráficos con Chart.js	Implementar gráficos "doughnut" para CPU, RAM y disco, y gráficos de líneas para uso del disco	6 - 7	Desarrollador Front-end	€30	20	€600
5	Integración con API de Netdata	Desarrollar funciones en JavaScript para obtener datos en tiempo real de la API de Netdata	8 - 9 - 10	Desarrollador Back-end	€50	30	€1500
6	Actualización de gráficos	Implementar lógica para actualizar gráficos cada segundo con datos de la API	11 - 12	Desarrollador Full Stack	€50	10	500
7	Sistema de alertas	Añadir alertas visuales cuando uso de recursos supera ciertos umbrales	13 - 14	Desarrollador Full Stack	€50	50	€2500
8	Cambio de tema	Implementar función para alternar entre temas claro y oscuro	14 - 15	Desarrollador Front-end	€50	5	€250
9	Pruebas y ajustes finales	Realizar pruebas finales y ajustes de código	15 -16	QA (Quality Assurance)	€30	10	€300
10	Documentación del proyecto	Documentar todo el proyecto y preparar presentación	16	Gestor de Proyecto	€30	20	€600

Ubicación: Proyecto/Documents/img/tabla.png

3.4 Realización

Paso 1: Planificar el proyecto y buscar información sobre las tecnologías a utilizar

Identificar las tecnologías adecuadas para el desarrollo del dashboard, como frameworks, bibliotecas y herramientas de monitoreo de recursos. Realizar una planificación detallada que incluya los objetivos, el cronograma y la distribución de tareas.

Paso 2: Estructura HTML

Crear una página HTML estructurada con contenedores y elementos necesarios para los gráficos que mostrarán el uso de CPU, RAM y disco. Asegurarse de que la estructura sea clara y fácil de mantener.

Paso 3: Estilo CSS

Definir y aplicar estilos CSS al dashboard para mejorar su apariencia visual. Esto incluye la disposición y diseño de los gráficos, el uso de colores y tipografías adecuadas, y la implementación de un diseño responsivo que se adapte a diferentes dispositivos.

Paso 4: Gráficos con Chart.js

Implementar gráficos utilizando la biblioteca Chart.js. Crear gráficos tipo 'doughnut' para representar el uso de CPU, RAM y disco, y gráficos de líneas para mostrar el uso del disco a lo largo del tiempo. Configurar los gráficos para que sean interactivos y visualmente atractivos.

Paso 5: Integración con API de Netdata

Desarrollar funciones en JavaScript para obtener datos en tiempo real desde la API de Netdata. Integrar estos datos en los gráficos del dashboard, asegurando que la información se actualice continuamente y refleje el estado actual de los recursos monitoreados.

Paso 6: Actualización de gráficos

Implementar la lógica necesaria en JavaScript para actualizar los gráficos en tiempo real cada vez que se reciben nuevos datos de la API. Asegurarse de que las actualizaciones sean fluidas y no afecten el rendimiento del dashboard.

Paso 7: Sistema de alertas

Añadir un sistema de alertas visuales que se activen cuando el uso de los recursos supere ciertos umbrales definidos. Configurar alertas que sean fáciles de entender y que permitan a los usuarios tomar medidas rápidas y eficaces.

Paso 8: Cambio de tema

Implementar una función que permita a los usuarios alternar entre diferentes temas de color para el dashboard, como un tema claro y un tema oscuro, para mejorar la usabilidad en distintas condiciones de iluminación.

Paso 9: Pruebas y ajustes finales

Realizar pruebas exhaustivas del dashboard para identificar y corregir errores. Ajustar el código y la configuración según sea necesario para asegurar que el dashboard funcione correctamente y cumpla con los requisitos establecidos.

Paso 10: Documentación del proyecto

Documentar todo el proceso de desarrollo del proyecto, incluyendo la planificación, las tecnologías utilizadas, el diseño y la implementación. Preparar una presentación final que resuma los resultados y los beneficios del nuevo dashboard de monitoreo de recursos.

---

4. Realización del Proyecto

4.1 Trabajos realizados

Fase 1: Planificación del Proyecto

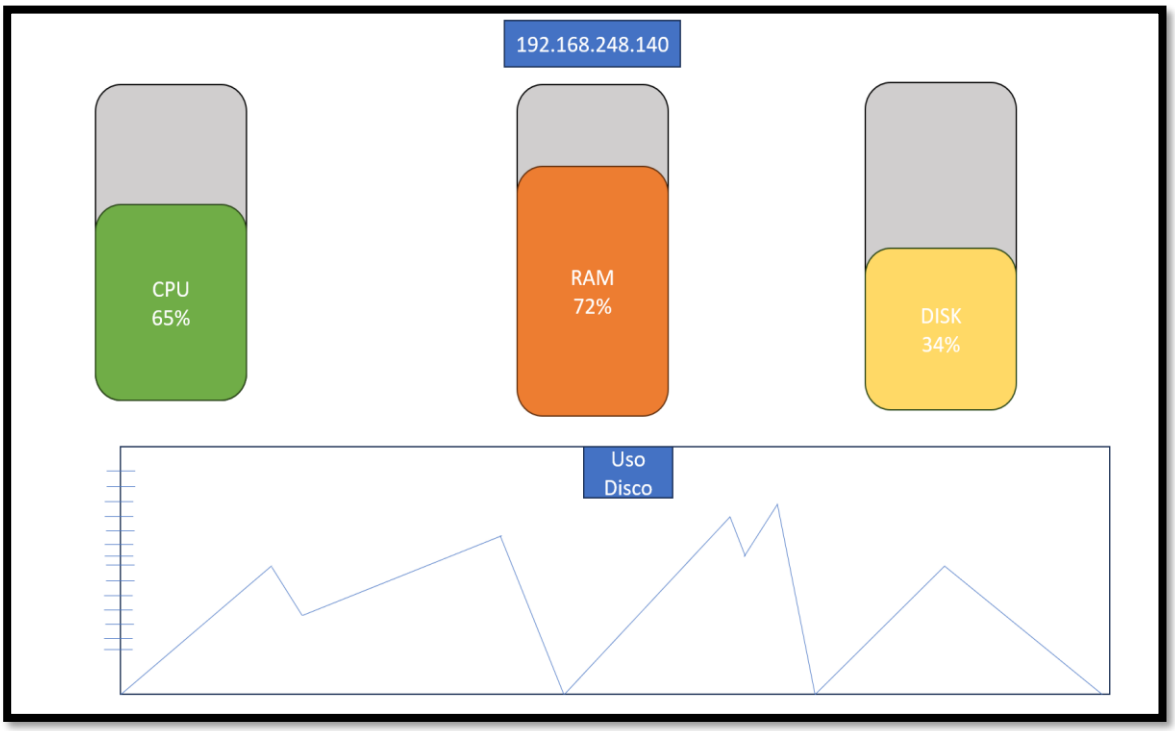
Semana 1

- **Planificar el proyecto y buscar información sobre las tecnologías a utilizar:** Esta fase inicial implica investigar y seleccionar las tecnologías más adecuadas para el proyecto. Se realiza una planificación general de cómo se desarrollará el proyecto.

Fase 2: Diseño de la Estructura HTML

Semanas 2 y 3

- **Diseño de la estructura HTML:** Durante estas semanas, se diseña la estructura básica del dashboard utilizando HTML. Se desarrolla el primer concepto del dashboard.
- Primera idea



Fase 3: Creación y Estilo del CSS

Semanas 4 a 6

- **Creación y estilo del CSS:** En esta fase, se estiliza el dashboard para que sea visualmente atractivo. Se desarrollan los estilos CSS que darán forma y diseño al dashboard. Durante la semana 5, se da estilo a las gráficas de las métricas.

Fase 4: Implementación de Gráficos con Chart.js

Semanas 7 a 8

- **Implementación de gráficos con Chart.js:** Se comienza a trabajar en la implementación de gráficos utilizando Chart.js para que las métricas sean más visuales y fáciles de entender. Esto incluye la realización de una investigación sobre cómo utilizar Chart.js de manera efectiva.

Fase 5: Integración de la API de Netdata para Obtener Datos

Semanas 9 a 12

- **Investigación y extracción de datos de la API de Netdata:** Se investiga cómo utilizar la API de Netdata y se empiezan a extraer datos. En la semana 12, se incluyen los datos de la API en los gráficos y se asegura que se actualicen en tiempo real.

Fase 6: Implementación de la Lógica de Actualización de Gráficos en JavaScript

Semanas 13 a 14

- **Implementar la lógica de actualización en JavaScript:** Se desarrolla la lógica necesaria para actualizar los gráficos en tiempo real utilizando JavaScript.

Fase 7: Creación del Sistema de Alertas

Semanas 15 a 16

- **Implementación con Prometheus y Alertmanager:** Se implementa un sistema de alertas utilizando Prometheus y Alertmanager, y se realizan pruebas para asegurar su correcto funcionamiento.

Fase 8: Pruebas y Ajustes Finales

Semana 16

- **Pruebas finales y ajustes de código:** Se realizan pruebas exhaustivas del dashboard y se hacen los ajustes necesarios para asegurar que todo funcione correctamente.
- **Resultado Final**

Fase 9: Documentación del Proyecto

Semana 16

- **Documentar el proyecto:** Se documenta todo el proceso de desarrollo del proyecto y se prepara una presentación final.

5. Problemas encontrados

- **Retrasos en la obtención de datos**:
    - Descripción: Se identificaron problemas con la latencia en la respuesta de la API de Netdata, lo que resultaba en retrasos significativos al obtener datos de monitoreo en tiempo real.
    - Solución: Se implementó un sistema de caché local para almacenar temporalmente los datos obtenidos de la API. Esto permitió reducir la latencia al acceder a métricas previamente consultadas y mejorar la velocidad de carga de los datos en la interfaz de usuario.

- **Compatibilidad de Chart.js**:
    - Descripción: Se experimentaron dificultades iniciales al configurar la biblioteca Chart.js para generar gráficos de líneas con los datos de monitoreo obtenidos de la API.
    - Solución: Se resolvieron estos problemas consultando detalladamente la documentación oficial de Chart.js y ajustando la configuración de los gráficos según las recomendaciones proporcionadas. Esto permitió mostrar de manera efectiva los datos de monitoreo en forma de gráficos interactivos en la interfaz de usuario.

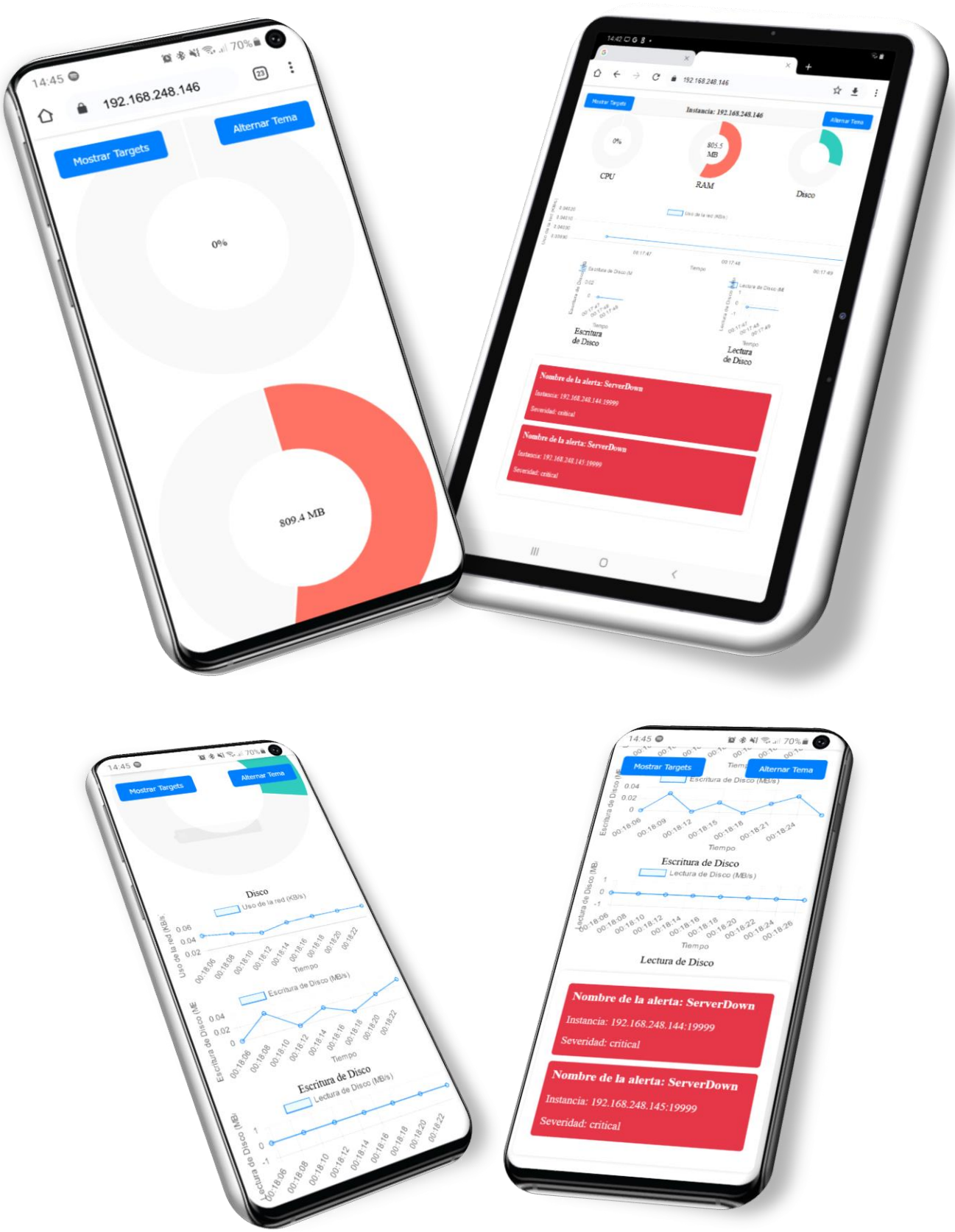
- **Sistema de alertas**:
    - Descripción: Se enfrentaron varios desafíos al configurar un sistema de alertas para detectar y notificar eventos de interés basados en las métricas de Netdata. Estos problemas incluyeron dificultades para integrar las métricas de Netdata con Prometheus, problemas de configuración entre Prometheus y Alertmanager, así como dificultades para implementar notificaciones de alerta por correo electrónico.
    - Solución: Se resolvieron estos problemas mediante un enfoque paso a paso, que implicó una revisión detallada de la configuración de cada componente del sistema de alertas. Se realizaron ajustes en la configuración de Prometheus y Alertmanager para garantizar una comunicación adecuada entre ellos, y se implementaron soluciones adicionales para habilitar las notificaciones por correo electrónico de manera efectiva.

- **Problemas para crear un histórico de alertas**:
    - Descripción: Se encontraron dificultades para desarrollar un sistema que mantuviera un registro histórico de todas las alertas generadas por el sistema de monitoreo.
    - Solución: No se pudo completar

6. Modificaciones sobre el proyecto planteado inicialmente

- Se añadió un sistema de caché para reducir la latencia en la obtención de datos de la API.

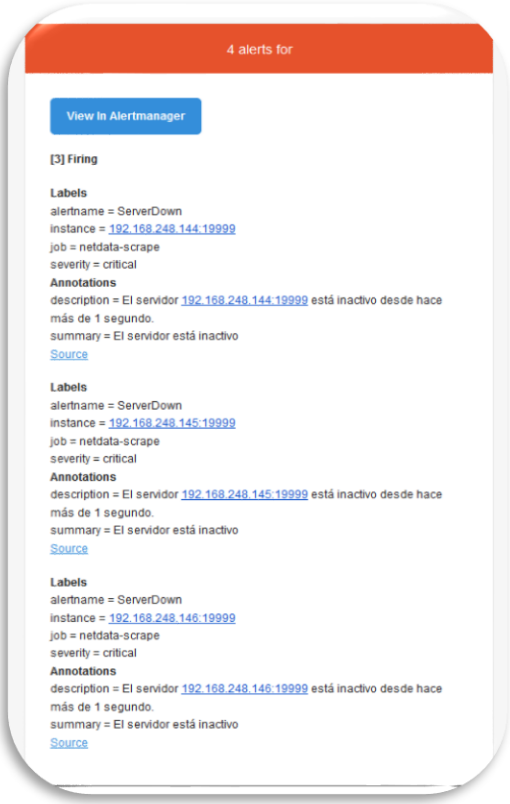
- Se optimizó el CSS para mejorar la responsividad del dashboard en dispositivos móviles y Tablets.



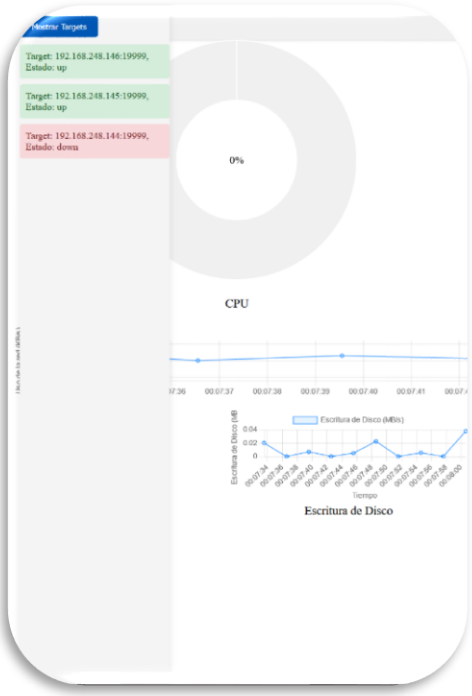
- Se añadió un sistema de alertas en el dashboard además de un sistema que manda un correo electrónico para las alertas.

  Dashboard:

Correo:



- Se añadió un sistema para alternar de máquina y ver cual está activa



- Se ha mejorado mucho el aspecto del dashboard respecto a la idea inicial

7. Posibles mejoras al proyecto

- **Ampliar el monitoreo a otros recursos:**
  - Añadir gráficos para monitorear la temperatura del sistema y el uso de la red.
- **Implementar notificaciones push:**
  - Integrar notificaciones push para alertar a los administradores sobre altos usos de recursos.
- **Mejorar la interfaz de usuario:**
  - Incluir más opciones de personalización y mejoras visuales en la interfaz del dashboard.

---

8. Bibliografía

- **Chart.js Documentation:**
  - https://www.chartjs.org/docs/latest/
- **Netdata API Documentation:**

    - https://learn.netdata.cloud/docs/agent/web/api
- **Prometheus Documentation:**
    - https://prometheus.io/docs/introduction/overview/
- **Alertmanager Documentation**
    - https://prometheus.io/docs/alerting/latest/alertmanager/

---

9. Fuentes del Proyecto

• **HTML, CSS y JavaScript:** - Los archivos index.html, style.css, y script.js están incluidos en el repositorio del proyecto. Estos archivos contienen el código fuente necesario para la estructura, diseño y funcionalidad de la interfaz de usuario del sistema de monitoreo.

• **Herramientas utilizadas:** - **Git:** Se empleó Git como sistema de control de versiones para gestionar el desarrollo del proyecto, lo que permitió realizar un seguimiento detallado de los cambios en el código y colaborar de manera efectiva entre los miembros del equipo. - **Visual Studio Code (VS Code):** Se utilizó VS Code como el entorno de desarrollo integrado (IDE) principal para escribir, editar y depurar el código HTML, CSS y JavaScript del proyecto. Su amplia gama de extensiones proporcionó funcionalidades adicionales que facilitaron el desarrollo y la productividad del equipo.

- **GitHub:** GitHub se utilizó como plataforma de alojamiento para el repositorio del proyecto, lo que permitió almacenar, compartir y colaborar en el código fuente de manera remota. Además, GitHub proporcionó herramientas para la gestión de problemas, seguimiento de errores y control de versiones a lo largo del ciclo de desarrollo del proyecto.

- **Chart.js:** Se empleó la biblioteca Chart.js para generar gráficos interactivos que visualizan los datos de monitoreo obtenidos de la API de Netdata. Chart.js ofrece una amplia gama de opciones de personalización y es compatible con varios tipos de gráficos, lo que lo convierte en una opción ideal para la representación visual de datos en el proyecto.

- **Netdata API:** Se hizo uso de la API de Netdata para acceder a datos de monitoreo en tiempo real de sistemas y aplicaciones. La API proporciona endpoints HTTP que permiten a los usuarios obtener métricas detalladas sobre el rendimiento del sistema, incluyendo el uso de CPU, memoria y red, entre otros.

- **Prometheus y Alertmanager:** Aunque se encontraron problemas en su integración, se consideraron como herramientas potenciales para la recopilación y gestión de alertas generadas por el sistema de monitoreo. Prometheus se utiliza para recopilar métricas de las aplicaciones y almacenarlas en una base de datos de series temporales, mientras que Alertmanager gestiona las notificaciones de alerta enviadas por Prometheus según las reglas definidas por el usuario.

• Módulos del ciclo en los que se basa el proyecto (ASIR):

- Administración de Sistemas Operativos: Este proyecto se basa en los conocimientos adquiridos en el módulo de Administración de Sistemas Operativos, donde se exploran temas relacionados con la gestión, configuración y optimización de sistemas operativos en entornos de servidor. Se aplican técnicas de monitoreo y diagnóstico para garantizar el rendimiento y la disponibilidad del sistema, lo que se refleja en el desarrollo de un sistema de monitoreo en tiempo real en este proyecto.

- Implantación de Aplicaciones Web: El desarrollo de la interfaz de usuario del sistema de monitoreo implica la aplicación de conceptos y técnicas aprendidas en el módulo de Implantación de Aplicaciones Web, donde se abordan temas como el diseño de interfaces de usuario utilizando HTML, CSS y JavaScript, así como la integración de bibliotecas y herramientas externas para mejorar la funcionalidad y la experiencia del usuario.

10. Anexos

10.1 Capturas de pantalla:

- Se incluyen capturas de pantalla del dashboard en funcionamiento en la carpeta screenshots.

10.2 Repositorio en GitHub

- https://github.com/ismaael4/Proyecto_ASIR

10.3 Documentación del código

- Se incluyen documentos PDF en la carpeta /PDFs donde se explica todo el código por partes de los distintos ficheros de código.
- Se incluye un documento en formato MarkDown donde se explica como realizar algunas pruebas y se puede ver de forma visual otros apartados.
- Se incluye un documento PDF con una guía de uso de prometheus y alertmanager.
- Enlaces:

  Están en /PDFs

  - HTML
  - CSS
  - JS
  - BASH
  - MARKDOWN