1. **Definición de Equipo:**

Juan Felipe Rojas
correo: jrojasm@eafit.edu.co
Sergio Andres Rojas Muñoz
correo: smunozr2@eafit.edu.co
Santiago Baenas Rivera
correo: sbaenar1@eafit.edu.co

2. **Asignación de roles y responsabilidades de cada integrante del equipo en el desarrollo del proyecto 2:**

Juan Felipe Rojas - Disponibilidad
Sergio Andrés Muñoz - Rendimiento
Santiago Baenas Rivera - Seguridad

3. **Dir GitHub del proyecto 2:**

https://github.com/oigreslol/proyectotelematica

4. **Especificación de requisitos no funcionales:**

Disponibilidad

Alta disponibilidad (High availability) es un protocolo de diseño del sistema y su implementación asociada que asegura un cierto grado absoluto de continuidad operacional durante un período de medición dado. Disponibilidad se refiere a la habilidad de la comunidad de usuarios para acceder al sistema, someter nuevos trabajos, actualizar o alterar trabajos existentes o recoger los resultados de trabajos previos. Si un usuario no puede acceder al sistema se dice que está no disponible. El término tiempo de inactividad (downtime) es usado para definir cuándo el sistema no está disponible.

Rendimiento

El rendimiento es la capacidad del sistema para poder atender una cantidad de peticiones u operaciones a un tiempo determinado, no se trata de la mejora del código si no de la capacidad de soportar un estrés y no disminuir la calidad y el tiempo en el que se hace una tarea para un usuario.

Seguridad
Capacidad que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático. De todas formas, no existe ninguna técnica que permita asegurar la inviolabilidad de un sistema.

5. **Rediseño de la aplicación del Proyecto 1 (aspectos que mejoraron del proyecto original).**

    Disponibilidad

    En el apartado de la disponibilidad se creó una instancia en EC2 de AWS para crear un balanceador de cargas, el cual nos permitió dar más abasto frente al proyecto original, haciendo que se puedan atender más solicitudes entrantes a la página.

    Rendimiento

    Se usó una instancia específica de css, se pusieron imágenes en caché, las consultas a los temas y a los comentarios que el usuario ya había visto anteriormente se guardan en caché, se creó una imagen redis en el docker compose como servicio para usar como almacenamiento en caché o diccionario, también se implementó un crud con consultas teniendo en cuenta el caché de la máquina y se implementó cookies parse para los req.

    Seguridad:

    Se agregó autentificación de cabecera para las consultas a base de datos y los api, se agrego seguridad a la contraseña al registrarse, se usó un captcha para poder ingresar.

Diseño para la escalabilidad (disponibilidad, rendimiento y seguridad)

Qué patrones de arquitectura específicos (patrones de arquitectura y patrones de escalabilidad) y mejores prácticas se utilizarán en la APLICACIÓN para apoyar esta escalabilidad

- *Modelo-vista-controlador*

    Este patrón, también conocido como patrón MVC, divide una aplicación interactiva en 3 partes, como:

    Modelo: contiene la funcionalidad y los datos básicos
    Vista: muestra la información al usuario (se puede definir más de una vista)
    Controlador: maneja la entrada del usuario

    Esto se hace para separar las representaciones internas de información de las formas en que se presenta y acepta la información del usuario. Desacopla los componentes y permite la reutilización eficiente del código.

- *Escalabilidad horizontal*

Podemos ver que nuestra aplicación se define como una escalabilidad horizontal ya que no se hizo ningún cambio de hardware importante para aumentar la disponibilidad y rendimiento, si no por el contrario, se usó software y otras herramientas para ayudar a suplir esta necesidad de escalar el proyecto.

Qué patrones de arquitectura específicos (patrones de escalabilidad y buenas prácticas) se utilizarán en el SISTEMA para apoyar esta escalabilidad:

Mejores prácticas

No hacer una consulta por cada refresh a la página
Tener una fábrica productora para los comentarios
Singleton para garantizar que solo exista una instancia del id del comentario/tema

Definición de Herramientas a utilizar:

- AWS EC2 para la creación del balanceador de carga, en una instancia de linux AMI.
- Docker-Compose
- Imagen contenedor Redis en docker-compose.
- Cookie-parser librería.