PROYECTO 2 - TÓPICOS ESPECIALES EN TELEMÁTICA

Grupo 23

- Isabela Muriel Roldan imurielr@eafit.edu.co
- Rafael Villegas Michel rvillegasm@eafit.edu.co
- Felipe Cortes Jaramillo fcortesj@eafit.edu.co
- Luisa Maria Vasquez Gomez lmvasquezg@eafit.edu.co

Roles Asignados

- **Disponibilidad:** Felipe Cortes Jaramillo
- **Rendimiento:** Rafael Villegas Michel
- **Seguridad:** Luisa Maria Vasquez Gomez
- **Integracion:** Isabela Muriel Roldan

Repositorio del proyecto: https://github.com/lmvasquezg/Proyecto2-Moodle

Especificación de requisitos no funcionales:

- **Disponibilidad:**
  - La arquitectura debe admitir *failover* y *fallback* sin pérdida de datos.
  - La arquitectura debe incluir una estrategia de monitoreo proactivo para permitir la autodetección y la autocorrección.
  - La arquitectura debe consistir en una infraestructura robusta con suficiente redundancia para satisfacer el tiempo disponible requerido.
  - El sistema debe estar disponible más del 98% del tiempo.

- **Rendimiento:**
  - El tiempo promedio de respuesta por parte de la aplicación debe ser menor a tres segundos.
  - El tiempo de carga de una página percibido por el usuario debe ser menor a cuatro segundos.
  - El sistema debe soportar 200 transacciones simultáneas.
  - El consumo promedio de recursos debe ser cercano a los siguientes valores:
    - Uso de CPU : 70%
    - Uso de Memoria: 70%
    - Uso de disco: 80%

- **Seguridad:**
  - Los datos no deben divulgarse a usuarios no autorizados o involuntariamente.
  - Los datos no deben ser modificados por usuarios no autorizados.
  - La integridad de los datos debe mantenerse cuando los datos se transmiten y cuando se guardan en la base de datos.

- **Integración:**
    - El sistema debe ofrecer al usuario múltiples alternativas de autenticación.
    - Las redes sociales integradas debe brindar los datos del usuario necesarios para su registro.
    - Las redes sociales integradas deben brindar funcionalidades nuevas al usuario además de las ya existentes en la plataforma a desplegar.

Diseño para la escalabilidad:

Disponibilidad:

I. Mejores prácticas:

- **Failover:** Durante los escenarios de carga máxima, la capacidad del sistema para funcionar en caso de falla de un nodo o componente al cambiar de forma transparente a otro componente de respaldo a menudo se denomina "conmutación por error".

- **Failback:** Generalmente ocurre después del *failover*. El nodo primario o el sitio primario se recuperarán de la falla y estarán completamente operativos.

- **Replicación:** Esto implica copiar los datos del nodo primario a todos sus nodos de respaldo para que sea fácil cambiar en caso de conmutación por error

    - Replicación activa: en esta configuración, todos los nodos procesan la solicitud del cliente.
    - Replicación pasiva: en esta configuración, el nodo primario procesa la solicitud y luego se copiará a los nodos secundarios.

- **Redundancia:** Habrá múltiples componentes redundantes en el sistema para facilitar el *failover*.

- **Mantenimiento continuo:** las actividades de mantenimiento regulares para los componentes de hardware son clave para mantener la infraestructura en buen estado. El mantenimiento regular aumenta la confiabilidad del hardware, asegura operaciones confiables y extiende la vida útil general del hardware.

- **Modelo 5R** (Reliability, Replicability, Recoverability, Reporting and Monitoring, and Redundancy):

    - Fiabilidad: La confiabilidad general del sistema es la probabilidad de que un sistema no contenga fallas. Se puede seguir el enfoque del proceso de mitigación de fallas para disminuir la probabilidad de estas en una aplicación de software: Predicción, prevención, detección y tolerancia a fallos.

- Replicabilidad: La estrategia de replicación de sitio involucra esencialmente todos los procesos y pasos para replicar la funcionalidad, los datos y los servicios en varios sitios “espejo”.

- Recuperabilidad: Indica qué tan bien el sistema proporciona una “*failover* transparente” a los nodos en espera cuando los nodos primarios fallan. La recuperación de datos y software automatizado consiste principalmente en un módulo de recuperación, que realiza continuamente dos tareas para sistemas bajo monitoreo : monitoreo y respaldo.

- Reporte y monitoreo: Un elemento crucial del modelo 5R es el componente de informe y monitoreo, que realiza el monitoreo continuo de los sistemas y servicios internos y externos y proporciona notificaciones instantáneas a los administradores del sistema.

- Redundancia: La redundancia se puede lograr mediante la planificación y dimensionamiento adecuados de infraestructura / capacidad y acomodando módulos y sistemas en espera para almacenar los datos de respaldo en caso de falla.

II. Selección de tácticas:

- Disponibilidad de clústeres en espera para hacerse cargo del clúster primario en caso de fallas inesperadas.

- Replicación y duplicación de disco para evitar fallas del mismo**.**

- Replicar los objetos de la base de datos, las actualizaciones del esquema y los datos en varios sitios espejo.

- El clúster de servidores web y el clúster de base de datos serán una implementación simple de un diseño N + 1.

III. Decisiones de diseño

- Implementar un grupo de máquinas virtuales autoescalable que se mantenga en constante ejecución con mínimo dos instancias, de manera que si una instancia falla o tiene una carga demasiado alta, se crea otra que permita regular el tráfico sin bajar el rendimiento de la aplicación en general. De esta manera se implementan varios factores del modelo 5R.

- Implementar un balanceador de carga que distribuya de manera equitativa las peticiones entre las instancias corriendo en el grupo autoescalable, evitando la sobrecarga de una única máquina virtual.

- Usar un servicio de base de datos que cree un clúster donde los datos estén guardados en múltiples instancias de la base de datos y posea un balanceador de carga propio para

estas. De esta manera, la información estará guardada redundantemente y se tendrá una mayor confiabilidad en la información. El cluster también asegurará la salud de las instancias que este contenga, para asegurar una persistencia en la disponibilidad y acceso de los datos en la aplicación.

IV. Definición de herramientas a utilizar

- **Elastic Load Balancer:** Para la distribución equitativa de tráfico entre las instancias existentes.
- **Auto-scaling groups:** Para tener varias instancias de EC2 corriendo simultáneamente y permitir la creación de una nueva en caso de falla.
- **Amazon RDS:** Para base de datos distribuida y cifrada en cada uno de los cluster.

Rendimiento:

I. Mejores prácticas

- Para mejorar el rendimiento de la aplicación las mejores prácticas están relacionadas con el desarrollo de la aplicación. Sin embargo, a nivel de infraestructura se pueden implementar las siguientes optimizaciones.

- Una red **CDN** es un aspecto que debe considerarse si el sitio web tiene muchos archivos multimedia. Las redes CDN aceleran la entrega de contenido utilizando su red de servidores distribuida globalmente para representar los activos globales estáticos y el contenido de transmisión para mejorar los tiempos de carga.

- El **almacenamiento en caché** elimina el tiempo que se consume al hacer costosas llamadas de recursos, como llamadas a bases de datos, invocaciones de servicios web, llamadas remotas a API, lecturas de contenido de CMS, etc., almacenando los "datos de interés" localmente en la aplicación. Esto aceleraría las páginas web y las transacciones que dependen de estas llamadas de recursos.

- **Monitoreo continuo de infraestructura:**

    - Monitoreo y pruebas de CPU y memoria durante las pruebas de carga y prueba de esfuerzo para descubrir cualquier problema de utilización de la CPU y volcado de memoria.
    - Validación de sesión del usuario y la replicación de objetos de caché realizando cambios en un nodo y verificando otros nodos del clúster.

- **Arquitectura informática distribuible:** al diseñar un componente de hardware o software, asegurarse de que pueda ser replicable y distribuible.

- **Componentes de software livianos:** los módulos de software deben probarse exhaustivamente para garantizar que el consumo de recursos, como la CPU y la memoria, se mantenga al mínimo.

- **Distribución inteligente de la carga:** los balanceadores de carga y los administradores de clúster deben asignar la carga por igual entre todos los nodos de computación para garantizar que cada nodo maneje un número óptimo de solicitudes y que ningún nodo esté sobrecargado en comparación con otro.

- La infraestructura de red debe optimizarse para admitir el ancho de banda suficiente generado por la carga máxima. Durante las pruebas de carga y resistencia, la CPU, la memoria y la red de todos los sistemas constituyentes deben ser monitoreados para asegurar que soporten adecuadamente la carga y respondan dentro de tiempos de respuesta aceptables. La planificación de la infraestructura también debe ser compatible con un entorno de recuperación ante desastres, (DR) para manejar catástrofes naturales inesperados.

II. Selección de tácticas

- Implementar componentes y metodologías de infraestructura de monitoreo basados en el desempeño interno y externo.

- Usar múltiples instancias de nodos de servidor o componentes de software de manera que la arquitectura sea distribuida.

- Determinar un número apropiado de nodos de servidor en el clúster, núcleos de CPU, capacidad del disco duro y tamaño de RAM para soportar la carga máxima especificada.

- Mejorar el tiempo de representación de activos y el tiempo de carga de la página mediante el uso de un sistema de almacenamiento en caché CDN.

III. Decisiones de diseño

- Diseñar una arquitectura distribuida donde hayan varias instancias de servidores y de bases datos con balanceadores de carga que impidan la saturación de un solo nodo.

- Uso de un sistema de almacenamiento en caché CDN que permite mejorar los tiempos de carga de las páginas de la aplicación que contengan contenido audiovisual.

- Realizar un monitoreo constante de la infraestructura para determinar el tiempo de respuesta en pruebas de estrés aplicadas, de manera que sea fácil identificar fallas en el diseño que hagan que la aplicación se alente. Las variables a ser continuamente monitoreadas serían las siguientes:

Componente	Parámetro a ser monitoreado
Servidor	● Rendimiento (bytes/s) ● Uso de la CPU ● Uso de memoria ● Uso de disco

	● Parámetros de memoria (crecimiento del tamaño de almacenamiento dinámico, utilización de memoria) ● Tiempo de espera de subproceso
Base de datos	● Rendimiento de la consulta (p. Ej., Exploración de tabla completa realizada por consulta, uniones por consulta) ● Uso de índices. ● Número de sesiones abiertas.

IV. Definición de herramientas a utilizar

- **Amazon CloudWatch:** Para el monitoreo continuo de todos los recursos de la infraestructura.
- **Amazon Auto-scaling group:** Para tener varias instancias de EC2 corriendo simultáneamente y permitir la creación de una nueva en caso de falla.
- **Amazon Load Balancer:** Para la distribución equitativa de tráfico entre las instancias existentes.
- **CloudFlare:** Para implementar sistema CDN y mejorar el tiempo de carga de la aplicación.

Seguridad:

I. Mejores prácticas

- **Principios de seguridad:** Los principios de seguridad actúan como pautas de diseño y arquitectura durante la implementación de seguridad ya que están diseñados en base a las mejores prácticas de esta categoría, tendencias de la industria, estándares empresariales y cualquier ley y normativa aplicable.

  - **Seguridad de defensa en profundidad:** este principio aplica políticas de seguridad apropiadas en todas las capas, componentes, sistemas y servicios utilizando técnicas, políticas y operaciones de seguridad apropiadas. La política y los controles de seguridad en cada capa deben ser diferentes de una a otra, lo que dificulta que un hacker rompa el sistema.

  - **Proteger el eslabón más débil:** Identificar y reparar el punto más vulnerable en la cadena de extremo a extremo de la empresa.

  - **Principio de privilegio mínimo:** de manera predeterminada, un usuario o rol de seguridad debe tener el privilegio más bajo para un recurso o una función. Los privilegios no se elevarán automáticamente por medios directos o indirectos y, por lo tanto, se debe mantener una política de "denegación predeterminada".

- **Segmentación:** todos los recursos y funciones de software y hardware deben clasificarse en varias categorías de seguridad, y el acceso debe restringirse a los usuarios con los roles y privilegios adecuados.

- **Punto de entrada único**: la aplicación debe permitir a los usuarios autenticarse a través de un único punto. Se deben evitar las funciones de entrada de puerta trasera y las URL de acceso directo.

- **Administración del manejo de la seguridad**: El sistema debe proporcionar una visión holística de la funcionalidad administrativa para manejar las funcionalidades importantes de seguridad.

- **Validación de datos de usuario:** los datos ingresados por el usuario debe validarse y limpiarse a fondo en varias capas. Los datos también deben codificarse correctamente cuando se almacenan y transfieren a varias capas.

- Los estándares y herramientas del Proyecto de seguridad de aplicaciones web abiertas **(OWASP)** se pueden aprovechar para las pruebas de seguridad. Este marco de trabajo ofrece una lista de riesgos más críticos de aplicaciones web y cómo prevenirlos, algunos de estos son:

    - **Pérdida de autenticación y gestión de sesiones:** Las funciones de la aplicación relacionadas con autenticación y gestión de sesiones son frecuentemente implementadas incorrectamente, permitiendo a los atacantes comprometer contraseñas, claves, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios. La recomendación principal para una organización es facilitar a los desarrolladores un conjunto único de controles de autenticación y gestión de sesiones fuertes.

    - **Cross-site scripting (XSS):** Las fallas XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación. La opción preferida para prevenirlas es codificar los datos no confiables basados en el contexto HTML (cuerpo, atributo, JavaScript, CSS, o URL) donde serán ubicados. También se recomienda la validación de entradas positivas o de"lista blanca".

    - **Configuración de Seguridad Incorrecta:** Una buena seguridad requiere tener definida e implementada una configuración segura para todos los componentes de la aplicación. Todas estas políticas de seguridad deben ser definidas, implementadas y mantenidas, ya que por lo general no son seguras por defecto. Las recomendaciones primarias son el establecimiento de:
        - Un proceso rápido, fácil y repetible de mejora para obtener un entorno adecuadamente asegurado.
        - Un proceso para mantener y desplegar las nuevas actualizaciones y parches de software de una manera oportuna para cada entorno
        - Una fuerte arquitectura de aplicación que proporcione una separación segura y efectiva entre los componentes.

        - Ejecutar escaneos y realizar auditorías identificadas para ayudar a detectar fallos de configuración o parches omitidos.

    - **Ausencia de control de acceso a funciones:** la mayoría de las aplicaciones web verifican los derechos de acceso a nivel de función antes de hacer visible algo en la misma interfaz de usuario. A pesar de esto, las aplicaciones necesitan verificar el control de acceso en el servidor cuando se accede a cada función. La aplicación debería tener un módulo de autorización consistente y fácil de analizar, invocado desde todas las funciones de negocio. Frecuentemente, esa protección es provista por uno o más componentes externos al código de la aplicación.

    - **Falsificación de Peticiones en Sitios Cruzados (CSRF):** Un ataque CSRF obliga al navegador de una víctima autenticada a enviar una petición HTTP falsificado, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Esto permite al atacante forzar el navegador de la víctima para generar pedidos que la aplicación piensa son peticiones legítimas provenientes de la víctima. La opción preferida para prevenir CSRF es incluir el token único en un campo oculto. Esto hace que el valor de dicho campo se incluya en el cuerpo de la solicitud HTTP, evitando su inclusión en la URL, sujeta a mayor exposición.

    - **Redirecciones y reenvíos no validados:** Las aplicaciones web redirige y reenvía a los usuarios hacia otras páginas o sitios web, y usan datos no confiables para determinar la página de destino. Sin una validación adecuada, los atacantes pueden redirigir a las víctimas hacia sitios de phishing o malware, o utilizar reenvíos para acceder a páginas no autorizadas. Una buena práctica es que el valor de cualquier parámetro de destino al que se va a redireccionar, sea un valor de mapeo, en lugar de la dirección URL real o una porción de esta y en el código del servidor traducir dicho valor a la dirección URL de destino.

II. Selección de tácticas

- Usar una infraestructura de monitoreo de redes para comprender los cambios en los patrones de tráfico.

- Las páginas de administración de las cuentas de usuario solo serán accesibles para los usuarios que tienen un rol de administrador.

- La aplicación no permitirá las credenciales de usuario como parámetros de URL y no admitirá ningún punto de entrada de puerta trasera.

- Una aplicación web valida los datos de entrada del usuario a nivel de cliente y a nivel de servidor utilizando técnicas de validación de listas negras y listas blancas.

III. Decisiones de diseño

A pesar de que la mayoría de buenas prácticas y patrones para construir una aplicación web segura son basados en la implementación a nivel de código, a nivel de arquitectura y despliegue se pueden aplicar las siguientes medidas:

- Implementar sistemas de autenticación que verifiquen la identidad del usuario enviando confirmación por email e implementen técnicas de validación de listas negras y listas blancas para sitios de redirección. Asimismo, estos sistemas no deben usar parámetros en URL para la autenticación de usuarios.

- Soportar el sistema con una infraestructura robusta y constantemente monitoreada que identifique rápidamente algún ataque o vulnerabilidad en ambiente de producción.

- Hacer que todo el tráfico entrante y saliente de la aplicación sea seguro (HTTPS).

- Aplicar algoritmos de cifrado a todos los datos de usuarios tanto en tráfico como almacenados en en base de datos.

IV. Definición de herramientas a utilizar

- **CloudFlare** Para expedición de certificado SSL y sistema CDN.
- **Amazon RDS** Para base de datos distribuida y cifrada en cada uno de los cluster.
- **Plugins Moodle (OAuth, OpenID Connect)** Para autenticación de usuarios.
- **Amazon EC2 Security Groups** como firewall virtual que controla el tráfico de una o varias instancias.

Integración:

I. Mejores prácticas

- El 73% de los usuarios prefieren iniciar sesión en un sitio con inicio de sesión social, en lugar de proporcionar una dirección de correo electrónico y crear una nueva cuenta. Al proporcionar a los visitantes la oportunidad de registrarse o iniciar sesión a través de sus perfiles de redes sociales, está acortando el proceso de registro, lo que tiende a aumentar la tasa de conversión para los registros de usuarios y agregarle usabilidad a la aplicación.

- Siempre que se agregue una nueva API de redes sociales a una aplicación, asegurarse de que funcione correctamente antes de su próxima versión. Las pruebas de software son la mejor manera de garantizar la calidad de los productos digitales y detectar errores mientras la aplicación aún está en desarrollo.

- Usar funcionalidades que aumentan la seguridad en la aplicación como la incorporación de la autenticación de dos factores, el aprovechamiento de los controles de seguridad específicos de la plataforma y el cifrado de datos e información confidenciales.

- Leer detalladamente la documentación de cada una de las API a usar de manera que sea posible integrar a la aplicación la mayor cantidad de servicios posibles para aumentar la experiencia de usuario en la aplicación.

II. Selección de tácticas

- Implementar métodos de autenticación con redes sociales para facilitar la creación y autenticación de usuarios.

- Usar API de redes sociales que vayan de acuerdo al público objetivo que usará la aplicación

- Utilizar servicios de las API adicionales a la autenticación de usuarios para ofrecer una mejor experiencia a los usuarios.

- Verificar la seguridad de paso de mensajes entre el sistema y el servicio de autenticación.

III. Decisiones de diseño

- Autenticar usuarios usando los proveedores de correo electrónico más usados por estudiantes, maestros y personas relacionadas con el mundo educativo: Google y Microsoft.

- Permitir gentes externos acceder a la plataforma por medio de redes sociales como Facebook.

- Usar Gmail para envío de correo electrónico de la aplicación para confirmación de correo al momento del registro.

- Integrar el sistema con diferentes servicios de Office 365 desde los cuales podrá acceder desde el sistema.

- Implementar sincronización de actividades programadas en el sistema con calendario personal del usuario.

IV. Definición de herramientas a utilizar

- **OAuth2** Para autenticación de Facebook y Gmail.
- **SMTP (Gmail)** Para envío de correos de confirmación desde la aplicación.
- **OpenID Connect** Para autenticacion de Microsoft.
- **Office 365** Para integración de calendario y acceso a otros servicios desde la aplicación.

Proceso de instalación y DevOps:

1. Documentación del DNS

   Para la implementación del DNS en el proyecto se consiguió un dominio en freenom.com. El dominio registrado fue *proyecto2320201.tk*. Después se prosiguió a CloudFlare en la cual se creó

una cuenta, luego se agregó un nuevo sitio que se va a configurar utilizando esta plataforma. Para ello se ingresa el dominio ya existente, luego en la pestaña del DNS se agregan 2 registros: uno tipo A, el cual dirige a la dirección IP del recurso en AWS y uno tipo CNAME, que dirige al DNS del recurso especificado. Se vuelve a ingresar a freenom para agregar las configuraciones para redirigir al Cloudflare.

2. Documentación de certificados de seguridad

   Para el certificados de seguridad son entregados por la plataforma CloudFlare. Para activarlos se ingresa a la pestaña SSL/TLS y se selecciona el modo de encriptación como full. Esto encripta de inicio a fin utilizando un certificado de seguridad propio en el servidor. Finalmente, en la parte donde se configurar los certificados, se configura para que todos los request HTTP sean redirigidos a requests HTTPS.

3. Documentación de automatización DevOps

   Para automatizar el proceso de despliegue y generación de la arquitectura necesaria para soportar la aplicación se usó el servicio CloudFormation de AWS, el cual permite administrar la infraestructura como código. Con esto se creó un archivo de configuración tipo JSON, en el cual se especificó la creación y configuración de todos los recursos, como el grupo autoescalable de instancias EC2, el cluster de MariaDB en RDS y el balanceador de carga ELB. Allí mismo se especifican los comandos y servicios que cada una de las instancias tiene que correr al crearse, como por ejemplo instalar dependencias y correr los contenedores docker de moodle.

   Para generar dicha infraestructura, solamente se necesita montar el archivo JSON a la plataforma de CloudFormation como un Template, configurar ciertas variables y darle aceptar, y en cuestión de minutos se tendrá la aplicación corriendo sobre dicha infraestructura sin problema alguno.

4. Documentación de los servicios utilizados en nube

   Para los servicios en la nube se usó el proveedor AWS, con los siguientes elementos:

   - EC2 para manejar las instancias de máquinas virtuales.
   - ELB para crear y administrar el balanceador de carga entre las instancias.
   - RDS para crear el cluster de bases de datos relacionales.
   - CloudWatch para monitorear el estado del sistema en general.
   - CloudFormation para automatizar todo el despliegue de la arquitectura.

5. Análisis de Costo de la solución. Inversión inicial y mensual.

   El costo mensual de la aplicación es de alrededor de 101,86 USD por mes, tal como se puede evidenciar en la siguiente tabla:

SERVICIO	CANTIDAD	PRECIO
EC2	3	16,07
ELB	1	16,43
RDS	1	66,36
CloudWatch	1	3
Total		101,86

Referencias

Williams, J. and Winchers, D., 2013. *OWASP Top 10 De Riesgos De Seguridad En Aplicaciones*. [online] Owasp.org. Available at: <https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Espa%C3%B1ol.pdf> [Accessed 18 March 2020].