Proyecto II

Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Bases de Datos (IC 4302)
Segundo Semestre 2023

1. **Objetivo General**
   - ➔ Desarrollar un prototipo de plataforma para realizar búsquedas y recomendaciones de películas.
2. **Objetivos Específicos**
   - ➔ Implementar una aplicación que interactúe con bases de datos documentales.
   - ➔ Implementar una aplicación que interactúe con bases de datos orientadas a grafos.
   - ➔ Implementar un API en Python.
   - ➔ Desarrollar habilidades en lenguaje de programación Python.
   - ➔ Instrumentar aplicaciones Python para ser monitoreadas con Prometheus.
   - ➔ Diseñar e implementar Dashboards para Grafana.
   - ➔ Automatizar una solución mediante el uso de Docker, Kubernetes y Helm Charts.
   - ➔ Desarrollar arquitecturas de microservicios en Kubernetes.
3. **Datos Generales**
   - ➔ El valor del proyecto: 20%
   - ➔ La tarea debe ser implementada en grupos de máximo 5 personas.
   - ➔ La **fecha de entrega**:
     - ◆ Documentación: 16 de noviembre del 2023 antes de las 08:00 am
     - ◆ Proyecto: 17 de noviembre del 2023, la hora de entrega es contra cita de revisión.
   - ➔ Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo con el reglamento. La copia incluye código/configuraciones que se puede encontrar en Internet y que sea utilizado parcial o totalmente sin el debido reconocimiento al autor.
   - ➔ La revisión es realizada de forma virtual mediante citas en las cuáles todos los y las integrantes del grupo deben estar presentes, el proyecto debe estar completamente automatizado, el profesor decide en que computadora probar.
   - ➔ Se debe incluir documentación con instrucciones claras para ejecutar el proyecto.
   - ➔ Se deben incluir pruebas unitarias para verificar los componentes individuales de su proyecto, si estas no están presentes se obtendrá una nota de 0.
   - ➔ Se espera que todos y todas las integrantes del grupo entiendan la implementación suministrada.
   - ➔ Se deben seguir buenas prácticas de programación en el caso de que apliquen, entre ellas:
     - ◆ Documentación interna y externa.
     - ◆ Estándares de código.
     - ◆ Diagramas de arquitectura.
     - ◆ Diagramas de flujo

- Pruebas unitarias.

➔ Toda documentación debe ser implementada en Markdown.

➔ Si el proyecto no se encuentra automatizado obtendrá una nota de 0. Si el proyecto no se entrega completo, la porción que sea entregada debe estar completamente automatizada, de lo contrario se obtendrá una nota de 0.

➔ En caso de no entregar documentación se obtendrá una nota de 0.

➔ El email de entrega debe contener una copia del proyecto en formato tar.gz y un enlace al repositorio donde se encuentra almacenado, debe seguir los lineamientos en el programa de curso.

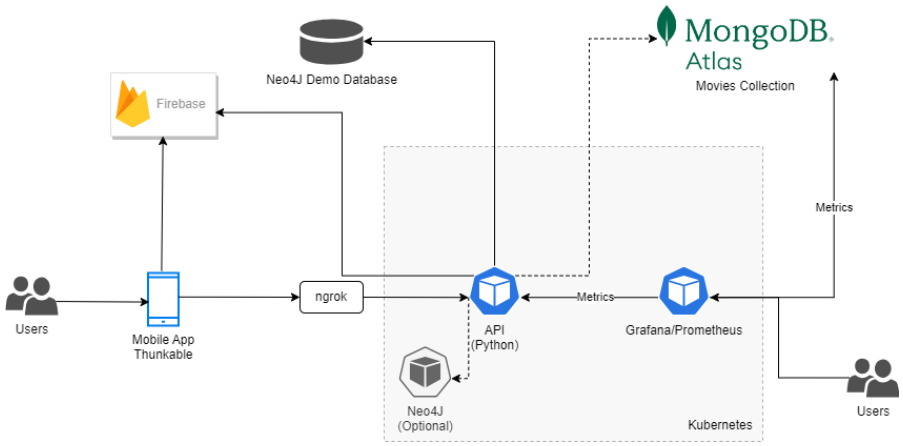
➔ Los grupos de trabajo se deben autogestionar, la persona facilitadora del curso no es responsable si un miembro del equipo no realiza su trabajo.

➔ En consulta o en mensajes de correo electrónico o Telegram, como mínimo se debe haber leído del tema y haber tratado de entender sobre el mismo, las consultas deben ser concretas y se debe mostrar que se intentó resolver el problema en cuestión.

➔ Como parte de la evaluación, se revisarán los aportes de código en el repositorio compartido con el profesor, esto con el fin de determinar que el trabajo en equipo este siendo realizado equitativamente por parte de los integrantes del grupo, no basta con cantidad de commits por persona, se evaluara calidad de los aportes.

4. **Descripción**

En el presente proyecto se harán uso de bases de datos NoSQL de tipo documentales (Mongo Atlas Search) y orientadas a objectos (Neo4J). El siguiente diagrama expone la solución a desarrollar:



- **Neo4J Demo Database**: Esta base de datos es ofrecida de forma online por Neo4J, la forma de conectarse a esta puede ser encontrada en https://neo4j.com/developer/example-project/ y la base de datos se accede en https://demo.neo4jlabs.com:7473/. En caso de que la base de datos no se encuentre disponible en línea por cuestiones de mantenimiento, esta puede ser ejecutada

en Kubernetes y accedida localmente, documentación en cómo usarla puede ser encontrada en https://neo4j.com/developer/example-data/.

- **Mongo Atlas Movies Collection**: Esta base de datos forma parte de los ejemplos ofrecidos por Mongo Atlas, es necesario crear un índice y un mapping, se debe documentar debidamente.
- **API**: Este consiste en una aplicación escrita en Python, se debe ejecutar con un Deployment que usa un servicio para exponer un REST API, este debe autenticar contra Firebase, implementa todos los métodos/endpoints requeridos para para realizar las búsquedas necesarias por parte de la aplicación, toda la interacción (requests) debe ser almacenada en una colección de Mongo Atlas, para su análisis, debe incluir el cuerpo del request en JSON, timestamp y el usuario que realizó la consulta. Los métodos/endpoints deben ser definidos y documentados por cada grupo, se espera un buen manejo de errores y la definición de unit tests.
- **Grafana/Prometheus**: Se deberá instalar en Kubernetes un sistema Grafana y Prometheus que deberá monitorear el API y Mongo Atlas, cada método/endpoint debe tener al menos una métrica count (número de peticiones) y request time (duración), también debe contar con gráficos para cada una.
- Ngrok: Esta es una herramienta que permite realizar un port-forward de la computadora y publicarlo en Internet, el mismo funcionara como un intermediario entre Thunkable y API que se ejecuta localmente. Si algún grupo tiene la posibilidad de ejecutar el API en un Cloud Provider, podrían correr el container y publicar el puerto (como en el proyecto 1).
- Mobile App: Se debe implementar una aplicación en Thunkable X que permita:
  - Autenticar y autorizar usuarios (Firebase).
  - La aplicación deberá permitir realizar búsquedas de películas tanto en MongoDB Atlas como en Neo4J mediante título, cast, plot y directors (en cada base de datos, se deberá usar sus campos equivalentes, en caso de no existir simplemente no se hace mediante ese campo), cuando se ingrese al documento encontrado se deberá mostrar toda la información disponible en la base de datos.
  - Una vez que se ha encontrado una película, se debe dar una opción en la cual, se puedan visualizar todas las películas en las cuales:
    - Un actor del cast, es actor.
    - Un actor del cast es director.
    - Un director es director.
    - Un director es actor.
  - Toda comunicación con el API se hace mediante Ngrok.

Documentación

Se deberá entregar una documentación que al menos debe incluir:

- Instrucciones claras de como ejecutar su proyecto.
- Pruebas realizadas, con pasos para reproducirlas.
- Pruebas unitarias y resultados de estas.

- Recomendaciones y conclusiones (al menos 10 de cada una).
- La documentación debe cubrir todos los componentes implementados o instalados/configurados, en caso de que algún componente no se encuentre implementado, no se podrá documentar y tendrá un impacto en la completitud de la documentación.
- Referencias bibliográficas donde aplique.

Imaginen que la documentación está siendo creada para una persona con conocimientos básicos en el área de computación y ustedes esperan que esta persona pueda ejecutar su proyecto y probarlo sin ningún problema, el uso de imágenes, diagramas, code snippets, videos, etc. son recursos que pueden ser útiles.
La documentación debe estar implementada en Markdown y compilada en un PDF.

5. **Recomendaciones**
   - Utilizar Docker Desktop para instalar Kubernetes en Windows.
   - Utilizar Minikube para ejecutar Kubernetes en Linux
   - Realicen peer-review/checkpoints semanales y no esperen hasta el último momento para integrar su aplicación.
   - Crear un repositorio de GitHub e invitar al profesor.
   - Realizar commits y push regulares.

6. **Entregables**

   - Documentación.
   - Docker files, Docker Compose files y Helm Charts junto con todos los archivos/scripts requeridos para ejecutar su proyecto.

7. **Evaluación**

Funcionalidad / Requerimiento	Porcentaje
Documentación (**)	20%
Implementación (*): ● API (40%) ● Mongo Atlas (5%) ● Grafana/Prometheus (15%) ● Thunkable App (20%)	80%
	100%

(*) Todo tiene que estar debidamente automatizado utilizando Helm Charts, de lo contrario se calificará con una nota de 0.

(**) Incluye pruebas unitarias.