Proyecto I

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

1. **Objetivo General**
   - ➔ Desarrollar un prototipo de aplicación que utilice bases de datos SQL y NoSQL.
2. **Objetivos Específicos**
   - ➔ Implementar una base de datos en tiempo real mediante la herramienta Firebase.
   - ➔ Implementar una base de datos SQL mediante Oracle Autonomous Database.
   - ➔ Implementar una base de datos NoSQL mediante Mongo Atlas.
   - ➔ Automatizar una solución mediante el uso de Docker.
   - ➔ Implementar un manejador de archivos en Ract y Object Storage.
   - ➔ Desarrollar un API en el lenguaje de programación Python.
3. **Datos Generales**
   - ➔ El valor del proyecto: 30%
   - ➔ Nombre del proyecto: WikiSearch.
   - ➔ La tarea debe ser implementada en grupos de máximo 5 personas.
   - ➔ La **fecha de entrega** es 02/10/2023 antes de las 6:00 pm.
   - ➔ 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.

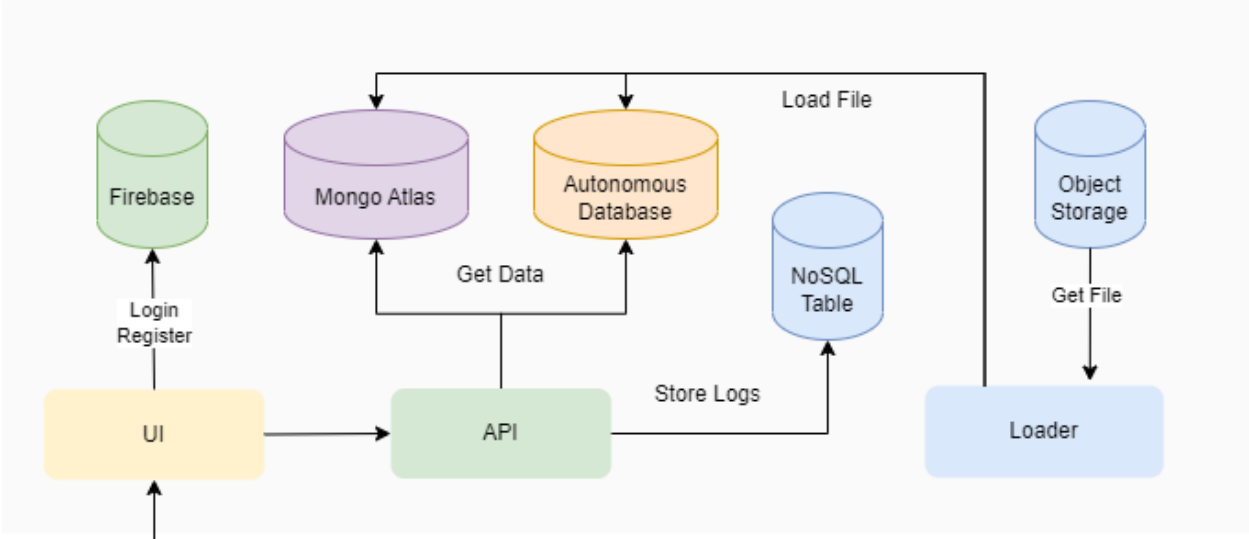
- ➔ En caso de no entregar documentación se obtendrá una nota de 0.
- ➔ El email de entrega debe contener una copia del proyecto en formato tar.gz y un enlace al repositorio donde se encuentra almacenado, debe seguir los lineamientos en el programa de curso.
- ➔ Los grupos de trabajo se deben autogestionar, 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.

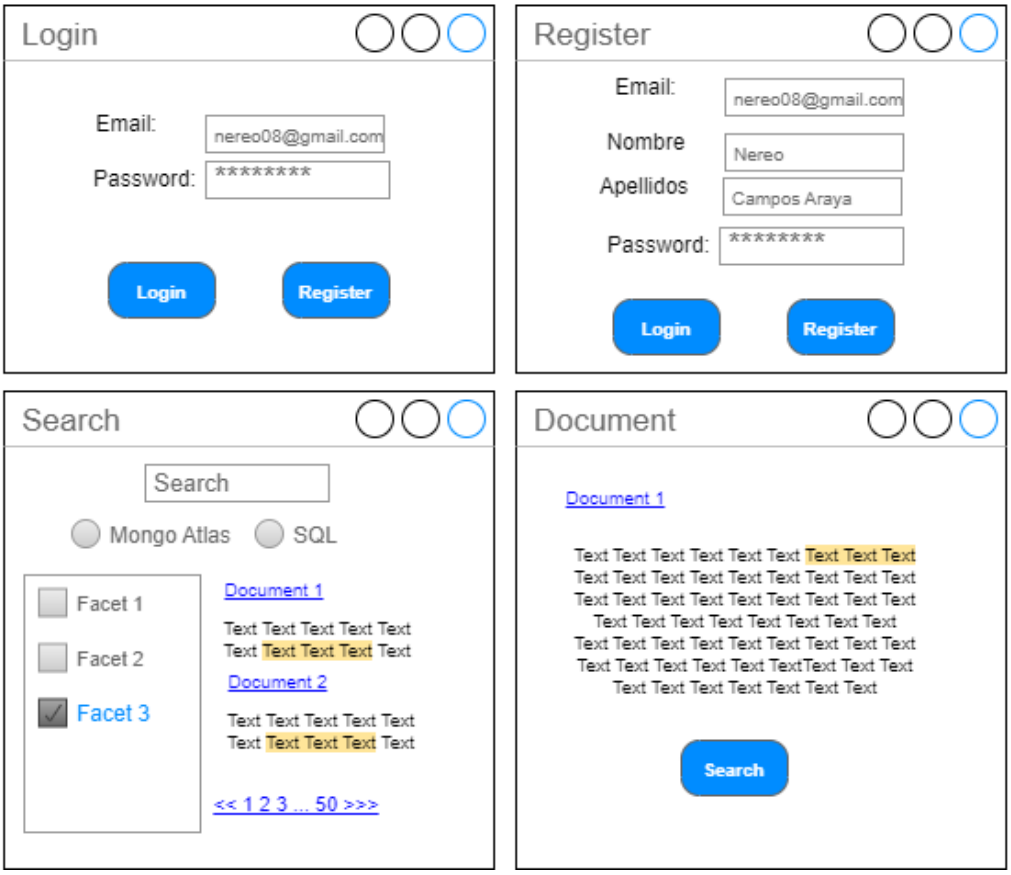
4. **Descripción**

Para el presente proyecto implementaremos un buscador para el sitio web Wikipedia, el mismo estará basado en motores de base de datos SQL y NoSQL, para obtener el contenido de este sitio utilizaremos los XML dumps, estos contienen un snapshot de todo el contenido textual de Wikipedia, el cual puede ser procesado y cargado en un motor de base de datos . Esta es una lista de los motores de base de datos y servicios que estaremos utilizando:

- Mongo Atlas y Mongo Atlas Search: Este es un servicio gratuito que permite crear una base de datos MongoDB y sobre esta ejecutar un índice invertido Lucene, muy parecido al que utiliza OpenSearch, Elasticsearch o Solr, esta es una solución nativa para implementar bases de datos documentales con un motor de búsqueda.
- Oracle Autonomous Database: Servicio de Oracle que permite crear bases de datos SQL.
- Oracle NoSQL Database Cloud: Servicio de base de datos NoSQL que permite crear bases de datos key-value de alto rendimiento.
- Firebase Real Time Database: Firebase es una plataforma para el desarrollo de aplicaciones, el mismo contiene módulos para autenticar usuarios y ofrece una base de datos en tiempo real entre muchas otras funcionalidades.
- Object Storage: Almacenamiento de objetos que permite almacenar archivos de cualquier tamaño.

A continuación, se presenta un diagrama del funcionamiento del proyecto:





- El componente **Object Storage** contendrá archivos de XML Dumps descomprimidos, estos se encuentran en el URL https://dumps.wikimedia.org/enwiki/latest/, los archivos que utilizaremos

son los que siguen el formato “enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2”, “enwiki-latest-abstract*.xml.gz” y “enwiki-latest-pages-articles-multistream-index*.txt-p*p*.bz2”, estos archivos serán manualmente descargados, descomprimidos y subidos dentro de Object Storage, no se espera que se carguen todos los archivos, solo los suficientes para demostrar el funcionamiento del proyecto.

- **Loader** es una aplicación que se debe escribir en Python y se debe ejecutar en un Docker container dentro de una máquina virtual del Cloud Provider, la misma cada cierto tiempo, debe estar escaneando el **Object Storage**, cuando detecta un nuevo archivo, deberá realizar el procesamiento de este, luego de procesarlo deberá cargar los datos en **Mongo Atlas** y **Autonomous Database**, para realizar el procesamiento de los archivos XML se recomienda el uso del modulo python-mwxml, el mismo permite obtener de forma sencilla los datos que conforman cada documento, es necesario realizar una lectura cuidadosa de esta documentación en especial de los apartados mwxml.SiteInfo, mwtypes.Page y mwtypes.Revision, es importante tener en cuenta los siguientes aspectos:
    - Es necesario crear un modelo relacional para almacenar los datos en **Autonomous Database**, el mismo se debe basar en la información que se debe recolectar para cada página, es importante la definición de índices y cualquier otro objeto de base de datos para maximizar el rendimiento.
    - Se debe crear un mapping adecuado para **Mongo Atlas**, esto es especialmente necesario para los fields que serán utilizados como facets.
    - No se cargarán todos los datos del sitio Wikipedia, solo algunos archivos y registros para validar el funcionamiento del proyecto.
    - Es importante implementar un mecanismo para no procesar mas de una vez el mismo archivo, cada grupo deberá idear la forma de que el loader cargue un archivo una sola vez.
- **API** es una aplicación que se debe escribir en Python/Flask (se puede usar algún otro modulo) y se deberá ejecutar en un Docker container dentro de una máquina virtual del Cloud Provider, esta aplicación verifica la identidad de un usuario contra Firebase (solo si es un usuario valido le permite ingresar), todo **request** será almacenado en el componente **NoSQL Table** como un log, este API deberá implementar todos los endpoints requeridos para el funcionamiento de la aplicación, queda a discreción de cada grupo la definición de estos, se espera una documentación clara de su funcionamiento. Esta aplicación deberá estar instrumentada mediante Prometheus, deberá contar con métricas para la cantidad de requests que ha recibido, tiempo promedio, tiempo máximo y tiempo mínimo de resolución por endpoint.
- **UI** se debe implementar en React (se puede usar cualquier framework u otra tecnología), se deberá ejecutar en un Docker container, esta UI presentará las siguientes funcionalidades:
    - **Login:** Mediante un email y password, la persona usuaria podrá ingresar a la aplicación, la identidad deberá ser validada contra Firebase.
    - **Register:** En caso de que una persona usuaria no cuente con una cuenta en la aplicación, esta podrá crearla contra Firebase, se recolectaran los datos Email, Nombre, Apellidos,

Teléfono y Password, una vez que la persona usuaria se ha registrado en Firebase, podrá ingresar a la aplicación.

- **Search:** Permitirá realizar una búsqueda de documentos/artículos seleccionando el motor de su preferencia, además se le permitida refinar la búsqueda mediante facets, los requerimientos de estos serán explicados en la siguiente sección del documento, cuando la búsqueda sea realizada mediante Mongo Atlas, se deberá realizar un Highlight del texto que ha sido encontrado.
- **Document:** En esta pantalla se mostrará toda la información del documento/articulo de Wikipedia, este debe ser recuperado de Mongo Atlas o de Autonomous Database, en esta sección se permitirá realizar una votación a favor o en contra por el documento (cada voto suma o resta 1), además tendremos un link directo al documento en el sitio de Wikipedia.

Documento Wikipedia

De los XML Dumps será necesario obtener los siguientes fields para el documento que deberá ser guardado en Mongo Atlas y en Autonomous Database:

- PageId: Esta información se obtiene del archivo "enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2" mediante mwtypes.Page y mwtypes.Revision.
- PageTitle: Esta información se obtiene del archivo "enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2" mediante mwtypes.Page y mwtypes.Revision.
- PageNamespaces: Esta información se obtiene del archivo "enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2" mediante mwtypes.Page y mwtypes.Revision.
  - Requiere un facet de tipo string.
- PageRedirect: Esta información se obtiene del archivo "enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2" mediante mwtypes.Page y mwtypes.Revision.
- PageHasRedirect: Si se existe información en el campo anterior, este deberá ser true de lo contrario false:
  - Requiere un facet de tipo string.
- PageRestriction: Esta información se obtiene del archivo "enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2" mediante mwtypes.Page y mwtypes.Revision.
  - Requiere un facet de tipo string.
- SiteInfoName: Esta información se obtiene del archivo "enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2" mediante mwxml.SiteInfo.
  - Requiere un facet de tipo string.
- SiteInfoDBName: Esta información se obtiene del archivo "enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2" mediante mwxml.SiteInfo.
  - Requiere un facet de tipo string
- SiteLanguage: Siempre "English"
  - Requiere un facet de tipo string.

- PageLastModified: Esta información se obtiene del archivo “enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2” mediante mwtypes.Page y mwtypes.Revision.
    - Requiere un facet de tipo date.
- PageLastModifiedUser: Esta información se obtiene del archivo “enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2” mediante mwtypes.Page y mwtypes.Revision.
    - Requiere un facet de tipo string.
- PageBytes: Esta información se obtiene del archivo “enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2” mediante mwtypes.Page y mwtypes.Revision.
    - Requiere un facet de tipo numeric.
- PageText: Esta información se obtiene del archivo “enwiki-latest-pages-articles-multistream*.xml-p*p*.bz2” mediante mwtypes.Page y mwtypes.Revision.
- PageWikipediaLink: Este field se encuentra en el archivo “enwiki-latest-abstract*.xml.gz”, para asociarlo con el documento se hace mediante PageTitle y el field <title> removiendo el texto “Wikipedia: ”, también se puede hacer mediante el archivo “enwiki-latest-pages-articles-multistream-index*.txt-p*p*.bz2” que contiene un mapeo PageTitle contra PageId.
- pageWikipediaGenerated: http://en.wikipedia.org/?curid=**{PageId}**
- PageLinks: Este field se encuentra en el archivo “enwiki-latest-abstract*.xml.gz”, para asociarlo con el documento se hace mediante PageTitle y el field <title> removiendo el texto “Wikipedia: ”, también se puede hacer mediante el archivo “enwiki-latest-pages-articles-multistream-index*.txt-p*p*.bz2” que contiene un mapeo PageTitle contra PageId.
- PageNumberLinks: Numero de links en el campo anterior.
    - Requiere un facet de tipo numeric.

En el caso de Autonomous Database, es necesario crear un modelo entidad relación y normalizarlo para crear la base de datos y almacenar la información.

Infraestructura en Oracle Cloud Infrastructure (OCI)

Es importante mencionar que los grupos no deberán crear la infraestructura en OCI de forma manual, el profesor proporcionará scripts de infraestructura como código para generar la infraestructura requerida, cada grupo deberá modificar algunos parámetros para generar su propia infraestructura y apuntar a sus imágenes de Docker, para estos efectos deberán instalar las herramientas Terraform y Oracle CLI. El profesor brindará una demostración de cómo crear la infraestructura.

Extra

Si algún grupo elimina el WikiText del campo PageText y además muestra y hace rendering correcto del WikiText en el UI recibirán un 3% extra sobre el proyecto (para lograr este comportamiento agregan otro campo PageTextRaw que contiene el texto con el WikiText).

Documentación

Se deberá entregar una documentación que al menos debe incluir:

- Instrucciones claras de como ejecutar su proyecto.
- Pruebas realizadas, con pasos para reproducirlas.
- Resultados de las pruebas unitarias.
- Documentación de los endpoints de Mongo Atlas utilizados, se debe apoyar con ejemplos de su código.
- 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 telnet/curl/postman para interactuar con los componentes y verificar que los mismos están funcionando correctamente.
   - Utilizar Oracle Cloud CLI para ver el estado de los recursos en la nube.
   - Utilizar Docker Desktop para instalar Docker en Windows.
   - 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.
   - Para la aplicación de React se recomienda usar algún tipo de JS Framework como VueJS.
   - Para la revisión, se recomienda tener clientes de administración de las bases de datos para poder realizar consultas y observar su comportamiento.
   - Algunos links de interés para este proyecto:
     - https://vuejs.org/
     - https://www.mongodb.com/docs/atlas/atlas-search/tutorial/create-index/
     - https://www.mongodb.com/docs/atlas/atlas-search/tutorials/
     - https://www.mongodb.com/docs/drivers/pymongo/
     - https://www.mongodb.com/docs/atlas/atlas-search/tutorial/run-query/#std-label-fts-tutorial-run-query
     - https://www.mongodb.com/docs/atlas/atlas-search/query-syntax/#std-label-query-

syntax-ref
    - https://www.mongodb.com/docs/upcoming/reference/operator/aggregation/facet/#mongodb-pipeline-pipe.-facet
    - https://www.mongodb.com/docs/atlas/atlas-search/highlighting/

6. **Entregables**

- Documentación.
- Documentación con las pruebas unitarias del API.
- Diagrama entidad/relación junto con su explicación.
- Archivos .sql para crear la base de datos.
- Mongo Atlas Mapping con su explicación.
- Docker files y código para generar el API, Loader y la aplicación UI.

7. **Evaluación**

Funcionalidad / Requerimiento	Porcentaje
Documentación (*)	15%
Pruebas unitarias (*)	10%
Diagrama entidad relación y archivo .SQL para crear la base de datos.	5%
Mongo Atlas Mapping con su explicación	5%
Loader	25%
API	25%
UI	15%
	100

(*) La completitud de la documentación, así como pruebas unitarias se encuentra acotado por la completitud de otros requerimientos.