**Especificación del diseño de la base de datos**

**Versión 3.0**

**Autores:**

Roberto Palacios Guevara

Valentín Martínez Motta

**Proyecto:**

Knowl App

**Fecha:**

10 de agosto de 2021

**1. Especificación del diseño de la base de datos**

**1.1. Objetivo del documento:**

Este documento es actualizable de acuerdo a los cambios técnicos de la base de datos que vayan apareciendo.

El objetivo del documento es especificar la base de datos y las tecnologías relacionadas a ellas que se empleará para el sistema Knowl.

**1.2. Tecnología a emplear:**

Con respecto a la base de datos que se utilizara en el proyecto Knowl utilizaremos el gestor de base de datos MongoDB, en este caso gracias MongoDB trabajaremos con colecciones y la agrupamos en users y technology. Dentro de una colección guardamos distintos datos y todo ello se guardará dentro de un objeto JSON (documentos); en conclusión, son una colección de objetos BSON.

Si tuviéramos que resumir a una la principal característica a destacar de MongoDB, sin duda esta sería la velocidad, que alcanza un balance perfecto entre rendimiento y funcionalidad gracias a su sistema de consulta de contenidos. Pero sus características principales no se limitan solo a esto, MongoDB cuenta, además, con otras que lo posicionan como el preferido de muchos desarrolladores.

Entre las características principales de MongoDB tenemos las siguientes:

* **Indexación**. El concepto de índices en MongoDB es similar al empleado en bases de datos relacionales, con la diferencia de que cualquier campo documentado puede ser indexado y añadir múltiples índices secundarios.
* **Replicación**. Del mismo modo, la replicación es un proceso básico en la gestión de bases de datos. MongoDB soporta el tipo de replicación primario-secundario. De este modo, mientras podemos realizar consultas con el primario, el secundario actúa como réplica de datos en solo lectura a modo copia de seguridad con la particularidad de que los nodos secundarios tienen la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.
* **Balanceo de carga**. Resulta muy interesante cómo MongoDB puede escalar la carga de trabajo. MongoDB tiene la capacidad de ejecutarse de manera simultánea en múltiples servidores, ofreciendo un balanceo de carga o servicio de replicación de datos, de modo que podemos mantener el sistema funcionando en caso de un fallo del hardware.
* **Almacenamiento de archivos**. Aprovechando la capacidad de MongoDB para el balanceo de carga y la replicación de datos, Mongo puede ser utilizado también como un sistema de archivos. Esta funcionalidad, llamada GridFS e incluida en la distribución oficial, permite manipular archivos y contenido.
* **Ejecución de JavaScript del lado del servidor**. MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas. Esto representa una de las diferencias más importantes con respecto a las bases de datos relacionales. Y resulta que no es necesario seguir un esquema. Los documentos de una misma colección - concepto similar a una tabla de una base de datos relacional -, pueden tener esquemas diferentes.

Respecto a las ventajas, tenemos las siguientes:

* Validación de documentos.
* Motores de almacenamiento integrado.
* Menor tiempo de recuperación ante fallos.

Respecto a las desventajas, tenemos las siguientes:

* No es una solución adecuada para aplicaciones con transacciones complejas.
* No tiene un reemplazo para las soluciones de herencia.
* Aún es una tecnología joven.

**1.3. Especificación de la base de datos:**

Las colecciones y los datos a almacenar en la base de datos son las siguientes:

Dentro de la colección de **Users** al momento de crearnos una cuenta y se guardarán dentro de la base de datos, tenemos los siguientes campos:

* id (este es un identificador único) el tipo de dato es ObjectId.
* firstName (nombre del usuario) el tipo de dato será String.
* lastName (apellidos del usuario) el tipo de dato será String.
* email (correo electrónico del usuario) el tipo de dato será String.
* Password (contraseña del correo del usuario) el tipo de dato será String.

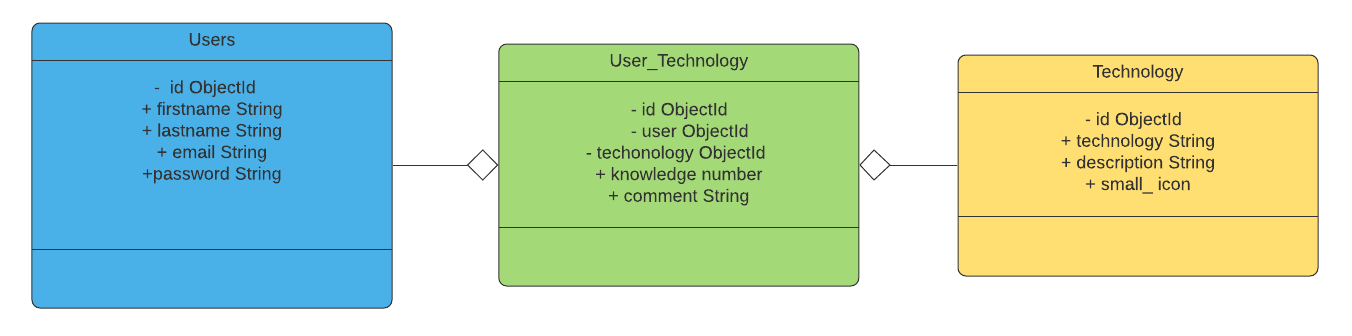
Dentro de la colección de **User\_Technology** abarca las tecnologías que un usuario domina y se mostrarán en su perfil, así como también habrá secciones donde podrá colocar una descripción y la puntuación en la misma, tenemos a los siguientes campos:

* id (este es un identificador único) el tipo de dato es ObjectId.
* user (referencia al usuario) el tipo de dato es ObjectId.
* technology (referencia a la tecnología) el tipo de dato es ObjectId.
* knowledge (conocimiento sobre esta tecnología 1-5) el tipo de dato es Number.
* comment (comentario que tiene sobre esta tecnología) el tipo de dato es String.

Dentro de la colección de **Technology** estarán las tecnologías que se toman en consideración a lo largo de toda la plataforma, se van actualizando conforme salgan nuevas tecnologías, tenemos a los siguientes campos:

* id (este es un identificador único) el tipo de dato es ObjectId.
* technology (nombre de la tecnología) el tipo de dato será String.
* description (una breve descripción acerca del dominio de tal tecnología) el tipo de dato será String.
* small\_icon (se podrá insertar una imagen referencial de la tecnología).

Presentamos el siguiente diagrama referencial, para ilustrar las relaciones que existirán entre las diversas colecciones a emplear en la base de datos:

****

**1.4. Implementación de la base de datos:**

Como se ha reportado ya en documentación anterior y siguiendo lo especificado en el presente documento, la base de datos ha sido implementada siguiendo las pautas y la estructura indicada. La URI también fue generada por el mismo servicio y definida como variable de entorno en el back-end.

El módulo de Mongoose de NestJS es el que emplea la URI para conectarse a la base de datos. Toda transacción debe realizarse únicamente a través del módulo de Mongoose, el cual será inyectado directamente en los módulos que necesiten de acceso a la base de datos.

Se prevé que para el final del segundo sprint se puedan realizar las operaciones relevantes con las tres colecciones definidas (como insertar, actualizar, leer y borrar) para darle la funcionalidad al sistema.

**1.5. Esquemas y modelos:**

Se definieron dos esquemas y dos modelos para el acceso a los datos. Tenemos userSchema, technologySchema, User y Technology.

Los modelos representan a los documentos (datos) devueltos por la base de datos, y contienen los atributos de cada entidad:

* User contiene los atributos: email, password, firstName, lastName, birthday, verified, profileImage, occupation, country y city; siendo todas de tipo string menos verified que es boolean.
* Technology contiene los atributos: technology, description e image; siendo todas de tipo string.

Los esquemas definen las reglas que deben seguirse para el almacenamiento de los datos. Tanto userSchema y technologySchema definen que todos los atributos son requeridos, definen los tipos de datos que deberían recibirse para almacenar datos, y algunos valores por defecto.