HACK ACADEMY

Coding Bootcamp

Sprint 3

Temario

Temario

- Borrar documentos.
- Editar documentos.
- Update Operators.
- Relaciones en MongoDB.
    - Anidación de documentos
    - Referencias a documentos.

Borrar documentos en Mongoose

Borrar documentos en Mongoose (1/4)

Borrar un (1) documento, dada su instancia:

```
unArticle.remove()
  .then(articleBorrado => /* ... */);
```

👉 Ver documentación.

Borrar documentos en Mongoose (2/4)

Borrar documentos que cumplan cierto criterio, a partir de un modelo:

```
Article.remove({ author: "Bill Gates" })
  .then(articlesBorrados => /* ... */);
```

👉 Ver documentación.

Los criterios de búsqueda son los mismos que cuando se utiliza `find`.

Borrar documentos en Mongoose (3/4)

Borrar un (1) documento que cumpla cierto criterio, dado su modelo:

```
Article.findOneAndRemove({ author: "Bill Gates" })
  .then(articleBorrado => /* ... */);
```

👉 Ver documentación.

Los criterios de búsqueda son los mismos que cuando se utiliza `find`.

Borrar documentos en Mongoose (4/4)

Borrar un (1) documento que tenga el ID correspondiente, dado su modelo:

```
Article.findByIdAndRemove('5b4d2d0957616e188ef34eb0')
  .then(articleBorrado => /* ... */);
```

👉 Ver documentación.

Editar documentos en Mongoose

Editar documentos en Mongoose (1/4)

Editar un (1) documento, dada su instancia:

```
unArticle.update({ published: true })
  .then(articleEditado => /* ... */);
```

👉 Ver documentación.

Editar documentos en Mongoose (2/4)

Editar documentos que cumplan cierto criterio, dado su modelo:

```
Article.updateMany({ author: "Bill Gates" }, { published: false })
  .then(articlesEditados => /* ... */);
```

👉 Ver documentación.

En este caso se editan todos los artículos del autor "Bill Gates" (se despublican).

Editar documentos en Mongoose (3/4)

Editar un (1) documento que cumpla cierto criterio, dado su modelo:

```
Article
  .findOneAndUpdate({ author: "Bill Gates" }, { published: false })
  .then(articleEditado => /* ... */);
```

👉 Ver documentación.

👉 Notar que podrían existir varios artículos del autor “Bill Gates”. En este caso, se actualiza sólo uno.

Editar documentos en Mongoose (4/4)

Editar un (1) documento que tenga el ID correspondiente, dado su modelo:

```
Article
  .findByIdAndUpdate("5b4d2d0957616e188ef34eb0", { published: true })
  .then(articleEditado => /* ... */);
```

👉 Ver documentación.

Update Operators

Update Operators (1/2)

Problema: ¿Qué pasa si necesitamos hacer un cambio relativo al valor actual almacenado en la base de datos?

Ej: necesitamos incrementar el campo `likes` de un artículo de un blog.

Con el teórico visto hasta ahora, primero sería necesario buscar el artículo en la BD para obtener su valor actual de `likes`. Y luego habría que actualizarlo con el nuevo valor. Son dos interacciones con la BD para una simple modificación. 👎

Afortunadamente, esto se puede hacer de una forma más eficiente.

Update Operators (2/2)

Los Update Operators son modificadores para usar en operaciones de actualización.

Todos estos operadores llevan el prefijo: `$`.

Ejemplo, el operador de incremento `$inc`:

```
Article
  .findByIdAndUpdate("5b4d2d0957616e188ef34eb0", { $inc: { likes: 1 } })
  .then(articleEditado => {/* */});
```

Por otros operadores, consultar los docs.

Relaciones en MongoDB

Relaciones en MongoDB

Hay dos grandes maneras de establecer relaciones en MongoDB:

1. Colocar (anidar) un documento dentro de otro (*embed*).
   Esto es muy similar colocar un objeto JSON dentro de otro.
   Por ejemplo, un artículo de un blog (`article`) podría tener un atributo `comments` conteniendo un array de comentarios. A su vez, cada uno de estos comentarios puede tener varios atributos.
2. Colocar referencias a documentos de otras colecciones.
   Esto es similar a lo que se realizar el modelo relacional de BD.
   Por ejemplo, un artículo de un blog (`article`) podría estar vinculado con autor que está en otra colección.

Relaciones en MongoDB

Anidación de documentos

Anidación de documentos (1/4)

La forma más sencilla de relacionar documentos es anidándolos. Ejemplo:

```
const articleSchema = new Schema({
  title: String,
  author: String,
  content: String,
  comments: [{ body: String, date: Date }],
});
```

En Mongoose, esto se llama *"subdocuments"*. Ver documentación.

Incluso se podría haber creado un esquema para los comentarios llamado `commentSchema` y luego usarlo de esta manera:

```
  comments: [commentSchema],
```

Anidación de documentos (2/4)

Pero hay que tener cuidado con modelar datos de forma muy anidada.

Hay una premisa que proviene del Python Zen (una serie de “mandamientos” para el programador) que dice:

“Flat is better than nested”

Esta premisa no es sólo válida para programar sino también para modelar datos.

Anidación de documentos (3/4)

Buenas prácticas:

- Sólo anidar información que sea muy intrínseca al modelo.
- En general, no anidar más de un nivel.
- Usar anidación para relaciones “uno-a-pocos”, no para “uno-a-muchos”.
  Ej: una persona puede tener algunas pocas direcciones.
- No anidar documentos que pueden tener “vida propia”, y que probablemente se los quiera acceder de forma independiente.

Por una buena lectura al respecto, ver este link.

Anidación de documentos (4/4)

Para agilizar la manipulación de documentos anidados, darle una vichada a los siguientes operadores:

- `$push`: https://docs.mongodb.com/manual/reference/operator/update/push.
- `$pull`: https://docs.mongodb.com/manual/reference/operator/update/pull.
- `$pop`: https://docs.mongodb.com/manual/reference/operator/update/pop.
- `$elemMatch`: https://docs.mongodb.com/manual/reference/operator/query/elemMatch.
- `$`: https://docs.mongodb.com/manual/reference/operator/projection/positional.
- `$set`: https://docs.mongodb.com/manual/reference/operator/update/set.

Relaciones en MongoDB

Referencias a documentos

Referencias a documentos (1/3)

En lugar de anidar documentos, es posible realizar referencias a documentos de otras colecciones. Eso es útil para relaciones “uno-a-muchos”.

Ejemplo en Mongoose:

```
const articleSchema = new Schema({
  title: String,
  author: {
    type: Schema.Types.ObjectId,
    ref: "Author",
  },
  content: String,
});
```

En Mongoose, esto se llama “populate”. Ver documentación.

Referencias a documentos (2/3)

Para asignarle un autor a un artículo, podemos hacer lo siguiente:

```
const author = new Author({
  name: "Hack Academy",
  email: "hola@ha.edu.uy",
});
author.save();

const article = new Article({
  title: "Historia de la academia",
  author: author, // También se podría haber puesto `author._id`.
});
article.save();
```

Referencias a documentos (3/3)

Ahora, si queremos poder acceder a los datos de un autor a partir de una artículo, debemos indicarle a Mongoose que “populate” la información:

```
Article.findOne()
  .populate("author")
  .then((article) => console.log(article.author));
```

Este código accede al último artículo de la colección y muestra los datos de su autor.