



CheckMed Proyecto

Fecha: 30 de Junio de 2021
Autor: Santillán Marcelo
Diseño de Bases de Datos 2020

ÍNDICE

[Descripción general del proyecto](#)

[Propuestas de la Cátedra para el TP](#)

[Mi propuesta](#)

[Material presentado](#)

[Explicación del proyecto](#)

[Criterios de aceptación del sistema a presentar](#)

[Aplicación Checkmed](#)

[Base de Datos](#)

[Datos de Prueba](#)

[ODM Mongoose](#)

[API Rest](#)

[CRUD de comentarios](#)

[Palabras finales](#)

Descripción general del proyecto

Se propone una Api Rest, que dado un flujo de recetas electrónicas, valide la medicación prescrita comparándola con la información en un vademecum.



El siguiente proyecto está inspirado en el paper [\(Snomed CDT Medicines Database of a General Hospital\)](#) en el cual se detalla la problemática en los errores de transcripción de medicamentos. Siendo este fragmento el más representativo:

When analysing the traditional components of medication use and where medication errors occur, Leape and cols. in their renowned study published found that 39% of errors occurred during the prescribing phase, 12% during transcription, 11% during dispensing, and 38% during administration. Consequently, much emphasis has been placed on the use of technology in prescribing. There are good reasons for this, including the many errors that involve misinterpreted handwriting or errors-prone abbreviations, the prescribing of inappropriate doses, and the occurrence of interactions and allergic reactions. Eliminating handwritten orders and implementing medication-checking software with decision support in prescribing systems can prevent such errors. However, the goal of every health care organisation should be to incorporate technology across the whole medication management spectrum.

El propósito del trabajo es cumplir con las pautas establecidas por la cátedra durante la cursada de 2020. Criterios:

Propuestas de la Cátedra para el TP

El trabajo práctico final de la materia puede tener varios posibles enfoques dependiendo de los conocimientos técnicos y de las inquietudes de cada grupo:

- enfocado en la aplicación de los conceptos y tecnologías vistos, con el fin de aprovechar la oportunidad para conocer nuevas formas de trabajo y eventualmente poder aplicarlas luego en el ambiente profesional. En este escenario el TP consta básicamente de un desarrollo de una pequeña aplicación que cuente con las principales capas de las arquitecturas más modernas y utilice todos los conceptos vistos:

Capas: base de datos (relacional o nosql), acceso a datos vía API REST para invocación de los servicios.

Funcionalidad mínima requerida: más allá del dominio del problema a resolver, el TP debe presentar operaciones CRUD (alta, baja, modificación y listado de información) contemplando el control de la edición concurrente por parte de dos usuarios en forma simultánea. No se requiere de un frontend.

- un enfoque comparativo con los esquemas más tradicionales: en el que se ponga de manifiesto algún criterio de comparación (performance, escalabilidad, etc). En este caso el TP se enfoca en la selección de una tecnología a utilizar, por ejemplo Mongo, y luego un reporte escrito en el que se documenta los hallazgos realizados mediante la ejecución de una batería de pruebas diseñadas para poder comparar algunos aspectos de interés. La entrega consta de documentación técnica para poder replicar localmente cada una de las pruebas llevadas a cabo, scripts y juegos de datos a utilizar, así como todos aquellos artefactos requeridos.

El documento debe contar con las conclusiones a las que llegó el grupo en base a las pruebas realizadas.

Trabajo práctico

Características

El trabajo práctico a presentar como mecanismo de evaluación es básicamente una implementación acotada en la que se ponen de manifiesto los conocimientos adquiridos en la materia. Se puede realizar en grupo de hasta 3 personas.

El grupo puede plantear un tema a resolver, y en caso de que no cuenten con uno con las características necesarias la cátedra les asignará uno.

Las tecnologías a utilizar en principio son de libre elección, siempre y cuando permitan obtener una implementación con el nivel deseado de independencia y abstracción de los detalles de la persistencia. Como recomendación se sugiere que se trate de una aplicación Web y en lo posible desarrollada sobre la plataforma Java.

La fecha límite para la presentación del TP es **30 de junio de 2021**.

Mi propuesta

Para el trabajo Práctico a presentar elegí principalmente el enfoque comparativo entre distintas técnicas aplicadas para el procesamiento de información con Node js, cree un informe con los resultados y una conclusión. Asimismo para aplicar las pruebas desarrolle una Api Rest sencilla para el acceso de datos de una base MongoDB, con operaciones Crud.

Material presentado

TP_Postgrado_DiseñoBD_1_proyecto	Descripción del trabajo práctico y de la aplicación
TP_Postgrado_DiseñoBD_2_reporte	Informe donde se comparan distintas técnicas para procesar registros de la base de datos, con una conclusión (documento principal de la entrega)
TP_Postgrado_DiseñoBD_3_instalacion	Manual de instalación de la aplicación
código	En github se encuentra el código en el link
Artefactos	Scripts para pruebas, base de datos generación de colecciones, y comandos para

Explicación del proyecto

La aplicación recibirá durante la jornada, el flujo de recetas electrónicas enviada por los profesionales a través de principalmente dos canales, un servicio que toma recetas digitales de un universo de profesionales o mediante el frontend de una aplicación, posteriormente se valida la medicación prescrita en las recetas digitales y se envía una notificación/observación de advertencia a la fuente generadora de la transacción en el caso de existir alguna anomalía. Por lo que por ejemplo si se trata de una aplicación pwa, se podrá agregar un comentario para realizar las correcciones que se consideren, asimismo durante el ciclo del ticket se pueden agregar nuevos comentarios de otras áreas/farmacias/profesionales, etc.

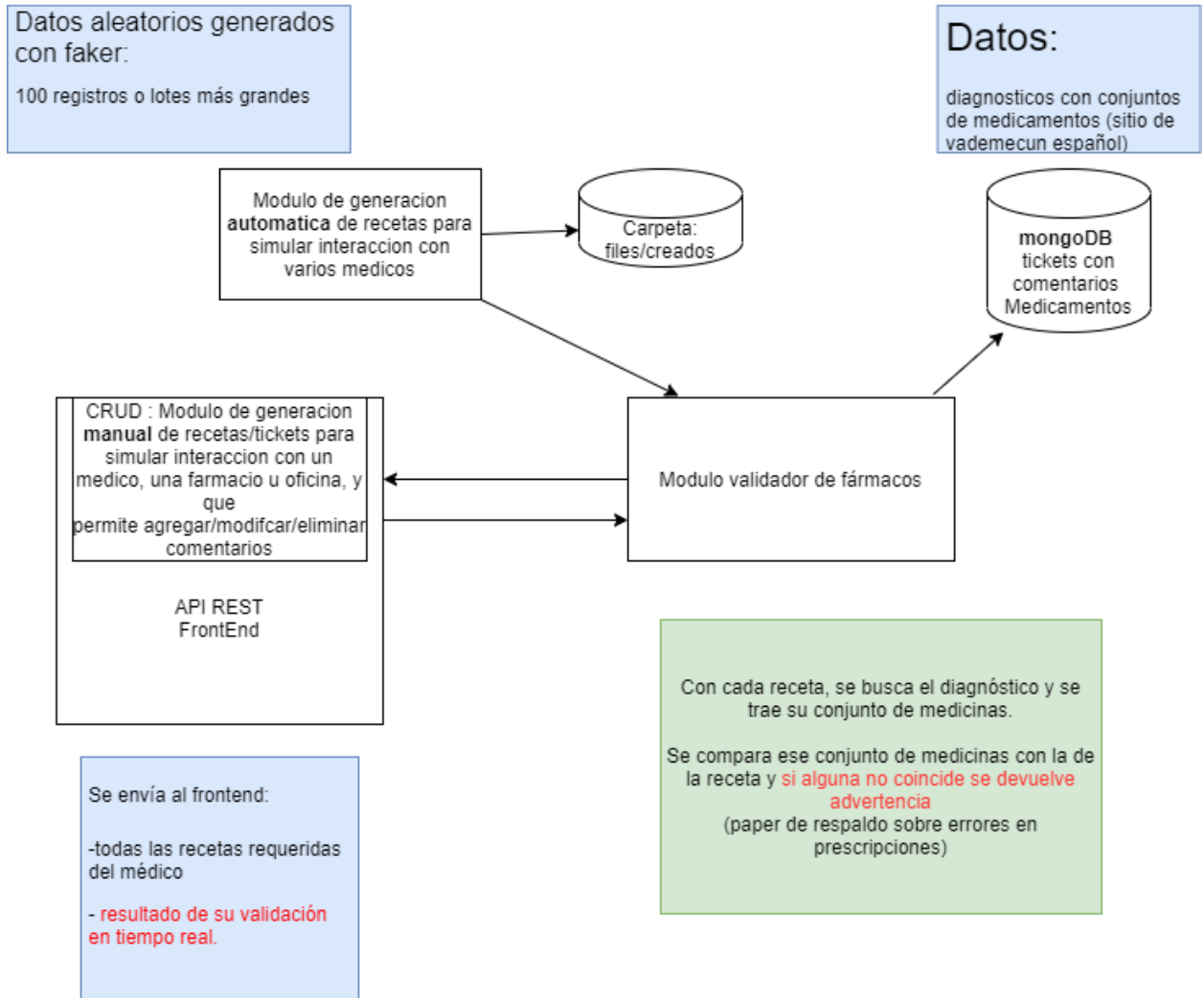
Estos comentarios pueden crearse, listar, editarse o eliminarse (crud).

Elegí como tecnologías aquellas que son [más ampliamente utilizadas actualmente de acuerdo al informe de stackoverflow \(javascript/nodejs/mongodb/api rest/npm\)](#) de las cuales me interesa profundizar su conocimiento a nivel profesional y aportar humildemente esta experiencia a la ingeniería de software por medio de este trabajo práctico.

Criterios de aceptación del sistema a presentar

- Debe estar funcionando constantemente procesando las validaciones de las recetas/tickets.
- Si los medicamentos recetados no están de acuerdo al diagnóstico y al vademecum, se debe agregar un comentario indicandolo en el ticket.
- También se pueden agregar otros comentarios durante distintas etapas en el ciclo de vida del ticket.
- los comentarios se pueden agregar, modificar, eliminar y realizar búsquedas sobre los mismos.
- los tickets no pueden ser eliminados.
- los tickets generados automáticamente tienen un respaldo físico en disco
- Tiempo de respuesta del resultado de las validaciones menor a 2 segundos desde el frontend.

<https://app.diagrams.net/#G1fBR7c1zWu7o-r7UdWU2GZjRjZ2iyyXkp>

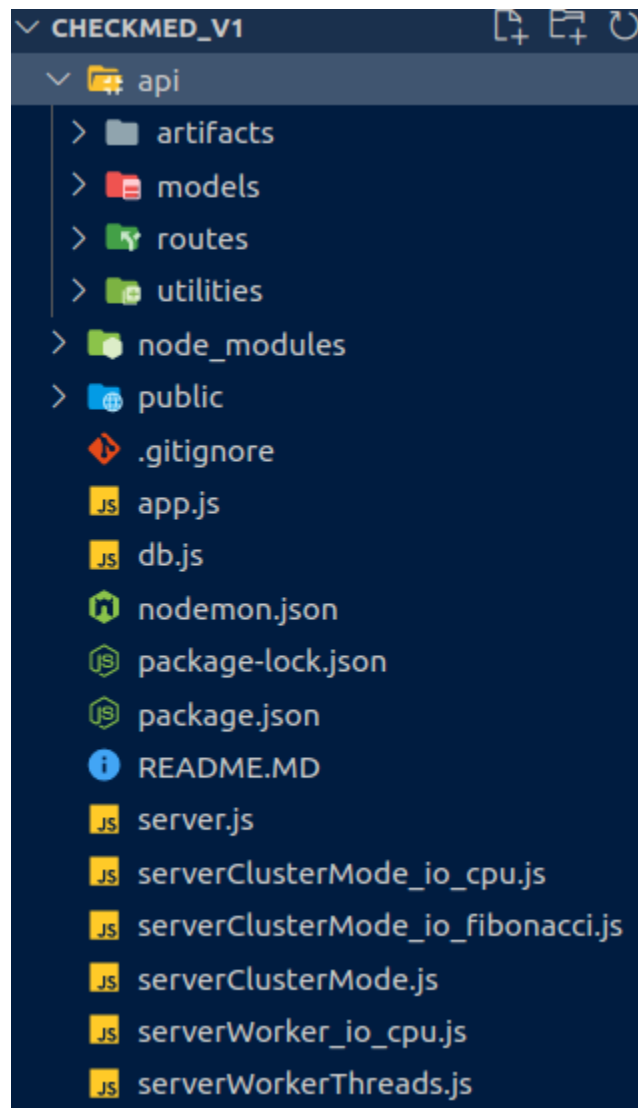


Aplicación Checkmed

Diseño

Bajo acoplamiento : dividido en capas de forma tal que al cambiar el comportamiento en una capa las demás no se vean afectadas lo cual ayuda al mantenimiento y escalabilidad. Se aplicarán la arquitectura Cliente/Servidor y la arquitectura de N capas.

Capas:



api : archivos de la capa de presentación con la cual van a interactuar los usuarios con el backend. Esta capa está relacionada con todos los recursos que salen a la web, siendo el intermediario entre el backend y el cliente.

models: aquí se encuentran las entidades del dominio. En arquitecturas basadas en el dominio esta es de las capas más importantes. Esta capa también se la conoce como business y es donde se encuentran nuestras entidades principales, como ticket, comentario y medicina.

routes: tiene los servicios que pueden comunicarse con otras apis, o con la capa de repositorio responsable del acceso a la base de datos. Esta capa llama a api cuando se hace una petición. Cuando el cliente quiere acceder al backend, primero ingresa a api, donde se aplican distintas validaciones

Utilities: Contiene la generación aleatoria de tickets y comentarios que van a ser la fuente de datos de las pruebas, junto con otras funciones que se van a utilizar por los distintos scripts durante las pruebas.

Artifacts: Contiene elementos que podemos utilizar para las pruebas:

checkmedPostman.postman_collection: los métodos de postman que va utilizar el api rest creado.

medicinas.txt : Al crear la base de datos se puede utilizar estos comandos para crear la colección de medicinas. Es la única colección que necesita ser creada manualmente ya que la de tickets y comentarios se realiza en forma automática. Se agrega esta explicación en el documento de intalación.

pruebas.txt : Comandos necesarios para ejecutar por terminal(console) los scripts de prueba.

La carpeta Node_Modules, contiene las dependencias, pero no es parte de esta división entre capas, de hecho no se incluye en git en el documento de instalación se encuentran los pasos que se necesitan seguir.

Base de Datos

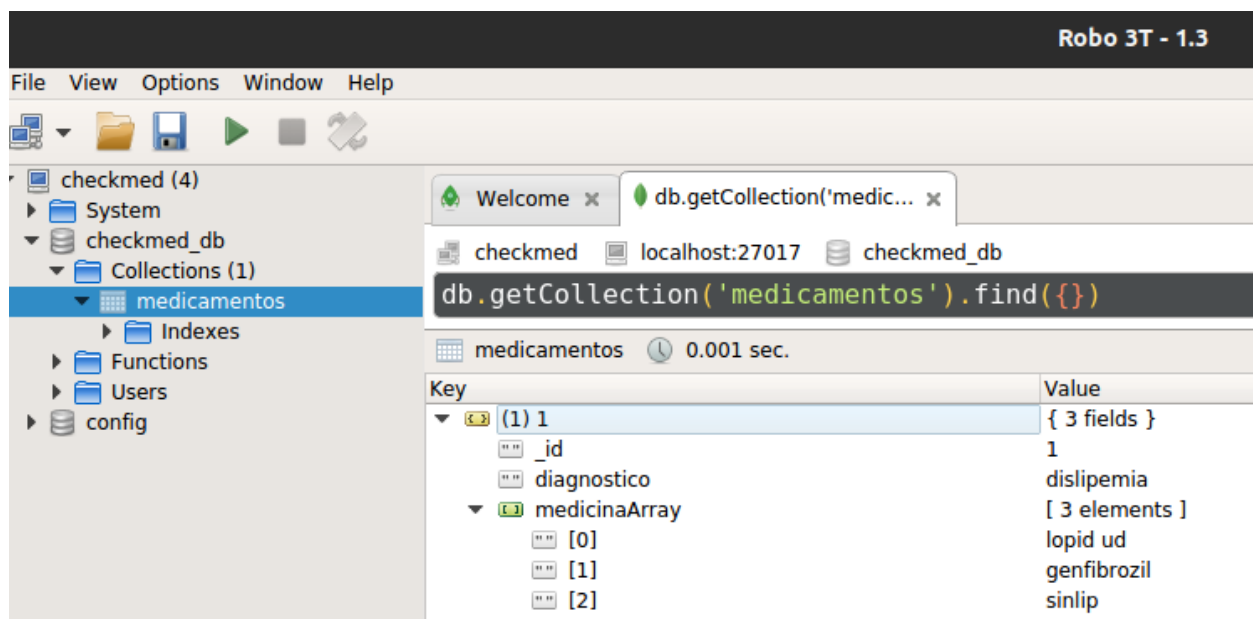
Voy a utilizar MongoDB, lo vimos en clase y me pareció interesante probar con una base de datos nosql, y la utilización de documentos, también voy a utilizar Robo3t como herramienta de gestión.

```
marcelo880@marcelo880-N56VB:~$ mongo
MongoDB shell version v4.4.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("edec928f-afda-4784-9703-fe85b241e14b") }
MongoDB server version: 4.4.5
---
```

Inserción de registro de medicinas:

<https://kb.objectrocket.com/mongo-db/how-to-add-elements-into-an-array-in-mongodb-1195>

```
use checkmed_db
switched to db checkmed_db
> db.medicamentos.insert({
... "_id": "1",
... "diagnostico": "dislipemia",
... "medicinaArray": ['lopid ud', 'genfibrozil', 'sinlip']
... });
```



Robo 3T - 1.3

File View Options Window Help

checkmed (4)

- System
- checkmed_db
 - Collections (1)
 - medicamentos
 - Indexes
 - Functions
 - Users
 - config

db.getCollection('medicamentos').find({})

medicamentos 0.001 sec.

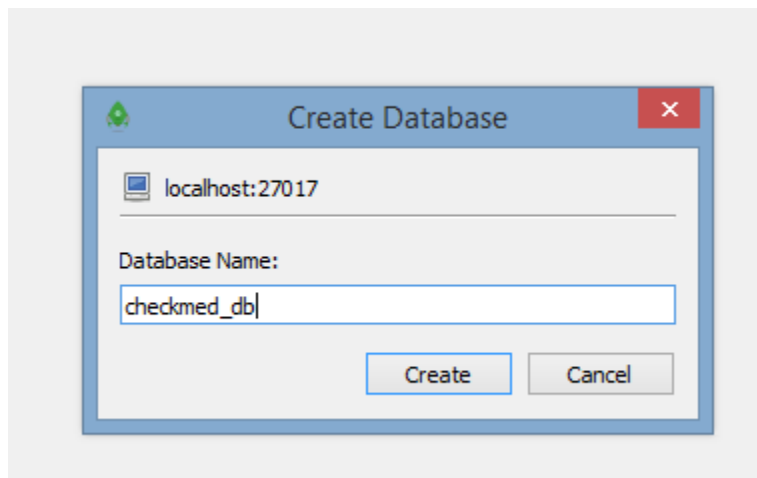
Key	Value
(1) 1	{ 3 fields }
_id	1
diagnostico	dislipemia
medicinaArray	[3 elements]
[0]	lopid ud
[1]	genfibrozil
[2]	sinlip

Búsqueda de los medicamentos para un determinado diagnóstico:

// para el diagnóstico dislipemia, se devuelve el array de medicinas, y no se muestra el id

```
db.medicamentos.find({diagnostico:"dislipemia"},{medicinaArray:1,_id:0})  
{ "medicinaArray" : [ "lopid ud", "genfibrozil", "sinlip" ] }
```

Voy a utilizar MongoDB con estos datos para su conexión:



MONGODB_URI = mongodb://localhost/checkmed_db

Colecciones de la base de datos:

- + Medicamentos
- + Tickets

---> comments

<https://stackoverflow.com/questions/17297632/automatic-persistence-of-node-js-objects-in-database>

Datos de Prueba

Para generar los datos de prueba, utilizo la librería faker js lo cual me permite generar datos aleatorios

<https://www.npmjs.com/package/faker>

Ejemplo en libfaker.js

Es posible acotar en algunos casos los valores a utilizar, por ejemplo en diagnóstico, prestación y idPrestador, para que de esta forma alternar entre valores de una manera controlada:

```
function fakeValues() {
  return {
    id: uuid.v4(),
    apyn: faker.name.lastName() + " " + faker.name.firstName(),
    nroAfiliado: faker.finance.account(),
    diagnostico: faker.random.arrayElement(["asma", "dislipemia", "herpes"]),
    idPrestador: faker.random.arrayElement(["10001", "10002", "10003", "10004"]),
    prestacion: faker.random.arrayElement(["lopid ud,genfibrozil", "sinlip", "asmabron", "asmavitan", "aciclovir", "herpial"]),
    fechaCreacion: faker.time.recent()
  }
}
```

```
module.exports.creaTickets = creaTickets;
```

Faker.js para crear api rest responses

<https://stackoverflow.com/questions/32010910/faker-js-random-number-between-2-values>


Array Element *faker.random.arrayElement(array[])* Selects a random element from an array of possible values. This function is useful to create custom lists of possibilities.

```
faker.random.arrayElement(["one","two","three","four"]); //returns "two"
var phTyp = faker.random.arrayElement(["cell","work","home"]); //returns "work"
```

La información relacionado a diagnósticos y medicamentos las tome del site del vademecun español: https://www.vademecum.es/enfermedades-d_1

Por ejemplo para dislipemia, tendríamos este listado:

Su fuente de conocimiento



Medicamentos	Monografías ATC	Clasif. ATC	Laboratorios	Noti
FLUVASTATINA RATIOPHARM Comp. de liberación prolongada 80 mg				
FLUVASTATINA SANDOZ Cáps. 20 mg				
FLUVASTATINA SANDOZ Cáps. 40 mg				
FLUVASTATINA STADA Comp. de liberación prolongada 80 mg				
FLUVASTATINA TEVA Cáps. dura 20 mg				
FLUVASTATINA TEVA Cáps. dura 40 mg				
FLUVASTATINA TEVA Comp. de liberación prolongada 80 mg				
GEMFIBROZILO ARISTO Comp. recub. 600 mg				
GEMFIBROZILO STADA Comp. recub. con película 600 mg				
GEMFIBROZILO STADA Comp. recub. con película 900 mg				
GEMFIBROZILO TARBIS Comp. recub. 900 mg				
GEMFIBROZILO TARBIS Comp. recub. con película 600 mg				
LOPID Comp. recub. con película 600 mg				
LOPID Comp. recub. con película 900 mg				
PROVISACOR Comp. recub. con película 20 mg				
PROVISACOR Comp. recub. con película 40 mg				


y para herpes:

Su fuente de conocimiento farmacológico

VADEMECUM Spain (España)


[Indíces](#) [Vademecum Box](#) [Noticias](#) [Productos](#)

[INICIO](#) / [ÍNDICE DE INDICACIONES](#) / [HERPES LABIAL](#)



Medicamentos

- ACICLOVIR ARISTO Crema 50 mg/g
- ACICLOVIR AUROVITAS Crema 50 mg/g
- ACICLOVIR COMBIX Crema 50 mg/g
- ACICLOVIR KERN PHARMA Crema 50 mg/g
- ACICLOVIR MABO Comp. 200 mg
- ACICLOVIR MABO Comp. 800 mg
- ACICLOVIR MABO Crema 50 mg/g
- ACICLOVIR MYLAN Crema 50 mg/g
- ACICLOVIR NORMON Crema 50 mg/g
- ACICLOVIR PENSA Crema 50 mg/g



Aprovechá HotSale
Farmaconline

ODM Mongoose

Mongoose es un Object Document Mapper (ODM). Esto significa que nos permite definir objetos con un esquema fuertemente tipado que se asigna a un documento MongoDB.

Mongoose proporciona una increíble cantidad de funcionalidades para crear y trabajar con esquemas.

Los esquemas definidos para este trabajo son:

Ticket

```
const mongoose = require('mongoose')

var Schema = mongoose.Schema;

var Comments = require('./comment').schema;

var BlogPosts = new Schema({
  apyn      : String,
  nroAfiliado : String,
  diagnostico : String,
  idPrestador : String,
  prestacion  : String,
  fechaCreacion : Date,
  comments   : [Comments]
}, { optimisticConcurrency: true, versionKey: 'version' });

module.exports = mongoose.model('Ticket', BlogPosts);
```

Comment

```
const mongoose = require('mongoose')

var Schema = mongoose.Schema;
var ObjectId = mongoose.ObjectId;
var Comments = new Schema({
  whoComment : String
, textBody   : String
, dateComment : Date
}, { optimisticConcurrency: true, versionKey: 'version' });
module.exports = mongoose.model('Comment', Comments);
```

Medicina

```
const mongoose = require('mongoose')

var Schema = mongoose.Schema;

var medicamentos = new Schema({
  _id: String,
  diagnostico: String,
  medicinaArray: [String]
})

module.exports = mongoose.model('medicamentos', medicamentos);
```

Para el diseño en general considere la colección de tickets con el campo comentarios formado por un array de subdocumentos comments.

Me pareció más interesante ese diseño dado que es el recomendado para este tipo de relaciones de uno a muchos, también las queries fueron sencillas, y no tuve necesidad de complicarme con joins.

Agrego algunos de los enlaces que utilice para tomar la decisión:

<https://docs.mongodb.com/manual/core/write-operations-atomicity/>

IMPORTANT:

In most cases, **multi-document transaction incurs a greater performance cost over single document writes**, and the availability of multi-document transactions should not be a replacement for effective schema design. For many scenarios, the **denormalized data model (embedded documents and arrays)** will continue to be optimal for your data and use cases. That is, for many scenarios, modeling your data appropriately will minimize the need for multi-document transactions.

For additional transactions usage considerations (such as runtime limit and oplog size limit), see also [Production Considerations](#).

<https://docs.mongodb.com/manual/core/data-model-design/#data-modeling-embedding>

Embedded data models allow applications to store related pieces of information in the same database record. As a result, applications may need to issue fewer queries and updates to complete common operations.

In general, use embedded data models when:

- you have “contains” relationships between entities. See [Model One-to-One Relationships with Embedded Documents](#).
- you have one-to-many relationships between entities. In these relationships the “many” or child documents always appear with or are viewed in the context of the “one” or parent documents. See [Model One-to-Many Relationships with Embedded Documents](#).

In general, embedding provides better performance for read operations, as well as the ability to request and retrieve related data in a single database operation. Embedded data models make it possible to update related data in a single atomic write operation.

<https://docs.mongodb.com/manual/crud/>

Podría por ejemplo tener varios comentarios en un ticket y modificar los cuatro a la vez, o eliminar los cuatro a la vez..... pero todo en una sola operación, es decir, estaría logrando persistir estos cambios en un solo paso, conteniendo toda la información relacionada en una sola operación.

<https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-many-relationships-between-documents/#data-modeling-example-one-to-many>

Porque utilizar subdocumentos embebidos: El modelo elegido cumple con estos requerimientos aconsejados

When to Use an Embedded MongoDB Document

There are a few key points to consider before using an embedded document structure in MongoDB:

1. Will the MongoDB child documents be needed every time the parent MongoDB document is required via a query?
2. The deletion of the parent MongoDB document will also delete the child MongoDB document—will this be acceptable for your data set?
3. In MongoDB, a document's size must be smaller than the maximum BSON size for documents, which is 16 megabytes. If document size will not be an issue, then an embedded document may be feasible as the MongoDB document structure.

<https://kb.objectrocket.com/mongo-db/how-to-work-with-an-embedded-document-in-a-mongodb-collection-378>

Y para poder aplicarlo seguí este artículo como guía:

<https://stackoverflow.com/questions/37537493/how-to-create-an-embedded-document-that-follows-a-model-with-mongoose>

http://seanhess.github.io/2012/02/01/mongodb_relational.html

<https://thecodebarbarian.com/a-node-js-perspective-on-mongodb-4-transactions.html>

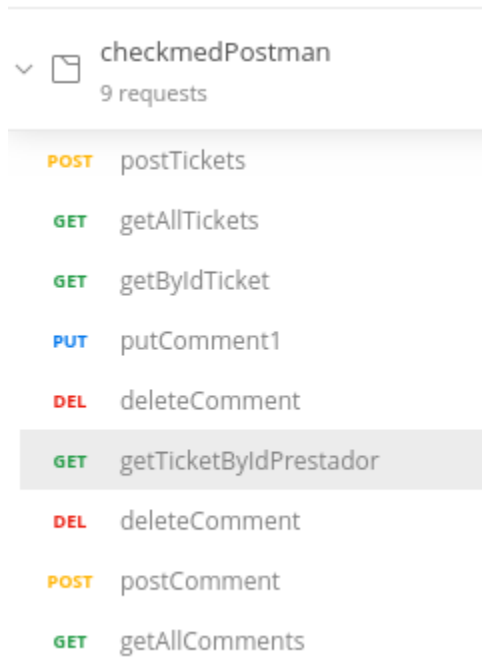
<https://developer.mongodb.com/quickstart/node-transactions/>

<https://stackoverflow.com/questions/46190153/update-object-inside-the-array-in-mongodb-using-mongoose?noredirect=1&lq=1>

API Rest

Cree un api rest sencillo el cual puede por ejemplo interactuar con Postman para simular un frontEnd.

Los métodos creados y utilizados con Postman son los siguientes:



- Creación de tickets:

Para crear tickets, se puede utilizar este método donde de acuerdo a los datos generados por la función creaTicket realiza una validación que puede crear un comentario.

```
outer.post("/", (req, res, next) => {  
  try {  
    let objFake = creaFakeTicket.creaTicket();  
    console.log("externo: " + objFake);  
  }  
})
```

```

res.status(201).json({
  message: "Se creo por POST un nuevo ticket",
  createdProduct: objFake._id
});
} catch (err) {
  console.log(err);
  res.status(500).json({
    error: err
  });
}

```

En postman

postTickets Examples 0 BUILD

POST http://localhost:3000/tickets Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

KEY	VALUE	DESCRIPTION	..
Key	Value	Description	

Body Cookies Headers (8) Test Results Status: 201 Created Time: 25 ms Size: 432 B Sa

Pretty Raw Preview Visualize JSON 🔍

```

1 {
2   "message": "Se creo por POST un nuevo ticket",
3   "createdProduct": "60cab6add1b41ee91c94e4d6"
4 }

```

En la terminal se ve el detalle

```

gen result: { _id: 60cab6add1b41ee91c94e4d6,
  apyn: 'Bashirian Jody',
  nroAfilado: '52278787',
  diagnostico: 'herpes',
  idPrestador: '10002',
  prestacion: 'lopid ud,genfibrozil',
  fechaCreacion: 2021-06-17T02:42:53.178Z,
  comments:
    [ { _id: 60cab6add1b41ee91c94e4d7,
      whoComment: 'Checkmed',
      textBody: ' Verifique la medicacion, por favor',
      dateComment: 2021-06-17T02:42:53.188Z } ],
  version: 0 }

```

Se realizó la validación y como para el diagnóstico “herpes” la medicación “lopid ud” o “genfibrozil” no son correctos, se agregó un comentario donde se advierte sobre esta situación.

En la base de datos:

Key	Value	Type
(1) ObjectId("60cab6add1b41ee91c94e4d6")	{ 9 fields }	Object
_id	ObjectId("60cab6add1b41ee91c94e4d6")	ObjectId
apyn	Bashirian Jody	String
nroAfilado	52278787	String
diagnostico	herpes	String
idPrestador	10002	String
prestacion	lopid ud,genfibrozil	String
fechaCreacion	2021-06-16 23:42:53.178-03:00	Date
comments	[1 element]	Array
[0]	{ 4 fields }	Object
_id	ObjectId("60cab6add1b41ee91c94e4d7")	ObjectId
whoComment	Checkmed	String
textBody	Verifique la medicacion, por favor	String
dateComment	2021-06-16 23:42:53.188-03:00	Date
version	0	Int32

- Búsqueda por id del prestador:

Podría darse una aplicación web o mobile donde un profesional por ejemplo, ingrese con su idPrestador para ver sus tickets.

```

router.get("/prestador/:idPrestador", (req, res, next) => {

```

```

const id = req.params.idPrestador;
Ticket.find({ idPrestador: id })
  .exec()
  .then(doc => {
    console.log("From database", doc);
    if (doc) {
      res.status(200).json(doc);
    } else {
      res
        .status(404)
        .json({ message: "No valid entry found for Prestador ID" });
    }
  })
  .catch(err => {
    console.log(err);
    res.status(500).json({ error: err });
  });
});

```

En postman:

► getTicketByIdPrestador Examples 0 ▾ BUILD

GET ▾ http://localhost:3000/tickets/prestador/10001 Send ▾

Params Authorization Headers (6) Body Pre-request Script Tests Settings

KEY	VALUE	DESCRIPTION	...
Key	Value	Description	

Body Cookies Headers (8) Test Results 🌐 Status: 200 OK Time: 691 ms Size: 629.74 KB Savi

Pretty Raw Preview Visualize JSON ▾ 🔧

```

1  [
2    {
3      "_id": "60bec872b69cb4fd9e02dfa9",
4      "apyn": "Larkin Henriette",
5      "nroAfiliado": "61943078",
6      "diagnostico": "dislipemia",
7      "idPrestador": "10001",
8      "prestacion": "asmabron",
9      "fechaCreacion": "2021-06-08T01:31:30.886Z",
10     "comments": [
11       {
12         "_id": "60bec872b69cb4fd9e02dfaa",
13         "whoComment": "Checkmed",
14         "textBody": " Verifique la medicacion, por favor",

```

En la terminal: Se pueden ver todos los json del prestador 10001

```

    version: 0 },
  { _id: 60c54de89b547aae990e4e30,
    apyn: 'Botsford Jakayla',
    nroAfiliado: '34443885',
    diagnostico: 'dislipemia',
    idPrestador: '10001',
    prestacion: 'aciclovir',
    fechaCreacion: 2021-06-13T00:14:32.168Z,
    comments: [ [Object] ],
    version: 0 },
  { _id: 60c54de89b547aae990e4e36,
    apyn: 'Muller Chandler',
    nroAfiliado: '39785611',
    diagnostico: 'dislipemia',
    idPrestador: '10001',
    prestacion: 'asmabron',
    fechaCreacion: 2021-06-13T00:14:32.181Z,
    comments: [ [Object] ],
    version: 0 },
  { _id: 60c54de89b547aae990e4e3a,
    apyn: 'Heathcote Armando',
    nroAfiliado: '51040627',
    diagnostico: 'asma',
    idPrestador: '10001',
    prestacion: 'lopid ud,genfibrozil',
    fechaCreacion: 2021-06-13T00:14:32.193Z,
    comments: [ [Object] ],
    version: 0 },
  ... 1936 more items ]
GET /tickets/prestador/10001 200 678.839 ms - 644505

```

- Búsqueda de tickets por id:

```

router.get("/:id", (req, res, next) => {
  const myId = req.params.id;

  //-----
  Ticket.findById(myId, function (err, doc) {
    if (err) {

```

```

    console.log(err);
    res
      .status(404)
      .json({ message: "No valid entry found for ID ticket" });
  }
  else {
    console.log("Result : ", doc);
    res.status(200).json(doc);
  }
});

```

en Postman:

getByldTicket Examples 0 BUILD

GET http://localhost:3000/tickets/60bec872b69cb4fd9e02dfbd Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 92 ms Size: 1.08 KB

Pretty Raw Preview Visualize JSON

```

1 {
2   "_id": "60bec872b69cb4fd9e02dfbd",
3   "apyn": "Zboncak Emma",
4   "nroAfiliado": "89931867",
5   "diagnostico": "herpes",
6   "idPrestador": "10004",
7   "prestacion": "sinlip",
8   "fechaCreacion": "2021-06-08T01:31:30.993Z",
9   "comments": [
10    {
11      "_id": "60bec872b69cb4fd9e02dfbe",
12      "whoComment": "Checkmed",
13      "textBody": " Verifique la medicacion, por favor",
14      "dateComment": "2021-06-08T01:31:30.997Z"

```

- Listar todos los tickets:

getAllTickets

```
router.get("/", (req, res, next) => {
```

```

Ticket.find()
  .exec()
  .then(docs => {
    console.log(docs);
    res.status(200).json(docs);
  })
  .catch(err => {
    console.log(err);
    res.status(500).json({
      error: err
    });
  });
});
});

```

En Postman:

getAllTickets Examples 0 BUILD

GET http://localhost:3000/tickets/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 1318 ms Size: 2.41 MB Sav

Pretty Raw Preview Visualize JSON ≡

```

37552 },
37553 {
37554   "_id": "60c63b282cd0678873eaa95e",
37555   "apyn": "Leffler Kasey",
37556   "nroAfiliado": "95883298",
37557   "diagnostico": "herpes",
37558   "idPrestador": "10001",
37559   "prestacion": "asmabron",
37560   "fechaCreacion": "2021-06-13T17:06:48.522Z",
37561   "comments": [
37562     {
37563       "_id": "60c63b282cd0678873eaa95f",
37564       "whoComment": "Checkmed",
37565       "textBody": " Verifique la medicacion, por favor",

```

CRUD de comentarios

- Creación

Como parámetro se pasa el id del ticket, y se generan comentarios al azar

```
router.post("/:id", (req, res, next) => {
  //-----
  //
  const myId = req.params.id;
  Ticket.findById(myId, function (err, Ticket) {
    if (!err) {
      Ticket.comments.push(newComment.creaComments());
      Ticket
        .save()
        .then(result => {
          console.log(result);
          res.status(201).json({
            message: "Se creo por POST un nuevo comment en el ticket " +
myId,
            createdProduct: result
          });
        })
        .catch(err => {
          console.log(err);
          res.status(500).json({
            error: err
          });
        })
    }
  });
});
```

El método para crear comentarios también utiliza faker para que sean aleatorios acotado entre ciertos valores:

```
const faker = require('faker');
const uuid = require('uuid');

function creaComments() {
  return {
    id: uuid.v4(),
    whoComment: faker.random.arrayElement(["Auditoria", "Farmacia
Caronte", "Dpto Legal", "Dr Ceres Pedro"]),
    textBody: faker.random.arrayElement(["Ticket ilegible",
"Correccion invalida", "Revise fechas", "Aplica codigo 25040"]),
    dateComment: faker.time.recent()
  }
}

module.exports.creaComments = creaComments;
```

Ejemplo con postman (se utilizará el ticket creado anteriormente):

```
localhost:27017 checkmed_db tickets
{
  "_id" : ObjectId("60cab6add1b41ee91c94e4d6"),
  "apyn" : "Bashirian Jody",
  "nroAfiliado" : "52278787",
  "diagnostico" : "herpes",
  "idPrestador" : "10002",
  "prestacion" : "lopid ud,genfibrozil",
  "fechaCreacion" : ISODate("2021-06-16T23:42:53.178-03:00"),
  "comments" : [
    {
      "_id" : ObjectId("60cab6add1b41ee91c94e4d7"),
      "whoComment" : "Checkmed",
      "textBody" : " Verifique la medicacion, por favor",
      "dateComment" : ISODate("2021-06-16T23:42:53.188-03:00")
    }
  ],
  "version" : 0
}
```

Se creó el nuevo comentario, y el Odm cambio el número de versión:

postComment Examples 0 BUILD

POST http://localhost:3000/comments/60cab6add1b41ee91c94e4d6 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 201 Created Time: 65 ms Size: 957 B Save F

Pretty Raw Preview Visualize JSON

```

12 {
13   "_id": "60cab6add1b41ee91c94e4d7",
14   "whoComment": "Checkmed",
15   "textBody": " Verifique la medicacion, por favor",
16   "dateComment": "2021-06-17T02:42:53.188Z"
17 },
18 {
19   "_id": "60cac020d1b41ee91c94e4d9",
20   "whoComment": "Dpto Legal",
21   "textBody": "Revise fechas",
22   "dateComment": "2021-06-17T03:23:12.627Z"
23 }
24 ],
25 "version": 1

```

En la base de datos:

Key	Value	Type
(1) ObjectId("60cab6add1b41ee91c94e4d6")	{ 9 fields }	Object
_id	ObjectId("60cab6add1b41ee91c94e4d6")	ObjectId
apyn	Bashirian Jody	String
nroAfilado	52278787	String
diagnostico	herpes	String
idPrestador	10002	String
prestacion	lopid ud.genfibrozil	String
fechaCreacion	2021-06-16 23:42:53.178-03:00	Date
comments	[2 elements]	Array
[0]	{ 4 fields }	Object
_id	ObjectId("60cab6add1b41ee91c94e4d7")	ObjectId
whoComment	Checkmed	String
textBody	Verifique la medicacion, por favor	String
dateComment	2021-06-16 23:42:53.188-03:00	Date
[1]	{ 4 fields }	Object
_id	ObjectId("60cac020d1b41ee91c94e4d9")	ObjectId
whoComment	Dpto Legal	String
textBody	Revise fechas	String
dateComment	2021-06-17 00:23:12.627-03:00	Date
version	1	Int32

- Modificación de comentarios

También para facilitar la tarea se generan datos al azar con faker

```
router.put("/", function (req, res, next) {
  var idParent = "60a858529bede09016f1bac1"
  var idChild = "60a858ead0be0891c9e8e7c4"
  var _whoComment = newComment.creaComments().whoComment;
  var _textBody = newComment.creaComments().textBody;

  try {
    let __ticket = updateComment(idParent, idChild, _whoComment,
    _textBody, Ticket);
    res.json({
      message: 'Success Update Comment ' + new Date(),
    })
  } catch (error) {
    res.status(500).json({ message: error.message })
  }
})
```

```

        createdProduct: Ticket
    })
} catch (error) {
    console.log(error);
}
})

```

para las modificaciones cree una función preparada con código de transacciones:

```

async function updateComment(idParent, idChild, _whoComment, _textBody,
post) {
    const session = await mongoose.startSession();

    session.startTransaction();

    //intento del 01/06 excelent!!
    await post.findById(idParent)
        .then((user) => {
            const address = user.comments.id(idChild); // returns a matching
subdocument
            console.log(address);
            address.whoComment = _whoComment;
            address.textBody = _textBody;
            address.dateComment = new Date();

            return user.save(); // saves document with subdocuments and
triggers validation
        })
        .then((user) => {
            console.log(user)
        })
        .catch(e => console.log(e));
}

```

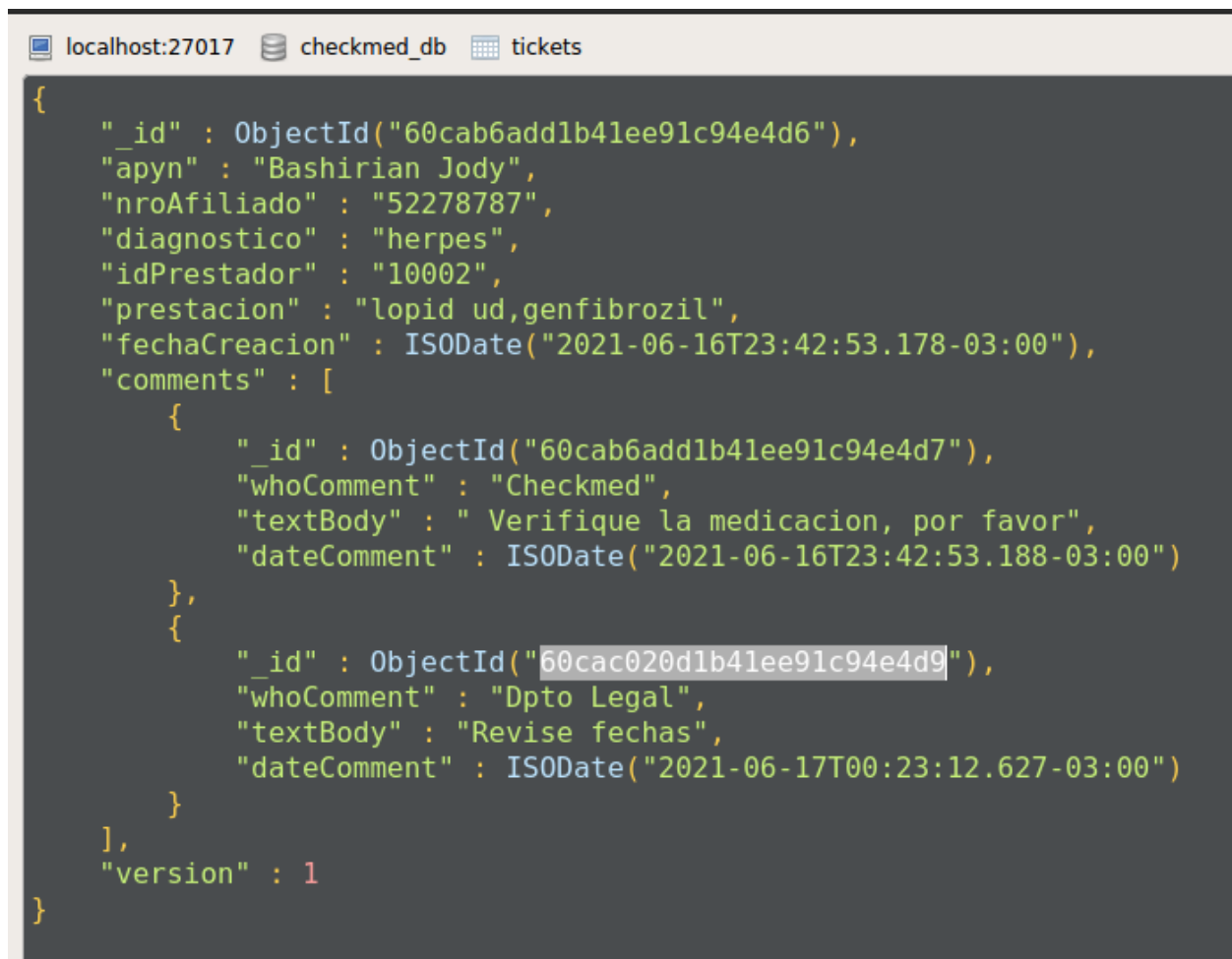
```

session.commitTransaction();
session.endSession();
return post// findOne

} // funcion update

```

Para el ejemplo visto, voy a modificar el 2do comentario:



The screenshot shows a web browser window with the address bar displaying 'localhost:27017'. The browser has two tabs: 'checkedmed_db' and 'tickets'. The main content area displays a JSON object representing a ticket and its comments. The JSON structure is as follows:

```

{
  "_id" : ObjectId("60cab6add1b41ee91c94e4d6"),
  "apyn" : "Bashirian Jody",
  "nroAfiliado" : "52278787",
  "diagnostico" : "herpes",
  "idPrestador" : "10002",
  "prestacion" : "lopid ud,genfibrozil",
  "fechaCreacion" : ISODate("2021-06-16T23:42:53.178-03:00"),
  "comments" : [
    {
      "_id" : ObjectId("60cab6add1b41ee91c94e4d7"),
      "whoComment" : "Checkmed",
      "textBody" : " Verifique la medicacion, por favor",
      "dateComment" : ISODate("2021-06-16T23:42:53.188-03:00")
    },
    {
      "_id" : ObjectId("60cac020d1b41ee91c94e4d9"),
      "whoComment" : "Dpto Legal",
      "textBody" : "Revise fechas",
      "dateComment" : ISODate("2021-06-17T00:23:12.627-03:00")
    }
  ],
  "version" : 1
}

```

En postman:

putComment1 Examples 0 BUILD

PUT http://localhost:3000/comments/ Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cook

KEY	VALUE	DESCRIPTION	...	Bi
Key	Value	Description		

Body Cookies Headers (8) Test Results Status: 200 OK Time: 64 ms Size: 433 B Save Res

Pretty Raw Preview Visualize JSON

```

1 {
2   "message": "Success Update Comment Thu Jun 17 2021 00:38:11 GMT-0300 (Argentina Standard Time)"
3 }

```

En la colección se observa un nuevo comentario distinto y se actualizó la versión:

```

PUT /comments/ 200 52.302 ms - 96
[
  {
    _id: 60cac020d1b41ee91c94e4d9,
    whoComment: 'Dpto Legal',
    textBody: 'Revise fechas',
    dateComment: 2021-06-17T03:23:12.627Z }
]
[
  {
    _id: 60cab6add1b41ee91c94e4d6,
    apyn: 'Bashirian Jody',
    nroAfiliado: '52278787',
    diagnostico: 'herpes',
    idPrestador: '10002',
    prestacion: 'lopid ud,genfibrozil',
    fechaCreacion: 2021-06-17T02:42:53.178Z,
    comments:
      [
        {
          _id: 60cab6add1b41ee91c94e4d7,
          whoComment: 'Checkmed',
          textBody: ' Verifique la medicacion, por favor',
          dateComment: 2021-06-17T02:42:53.188Z },
        {
          _id: 60cac020d1b41ee91c94e4d9,
          whoComment: 'Farmacia Caronte',
          textBody: 'Revise fechas',
          dateComment: 2021-06-17T03:38:11.377Z } ],
    version: 2 }
]

```

En la base de datos:

db.getCollection('tickets').find({})

tickets 0.002 sec. 7950

Key	Value	Type
(1) ObjectId("60cab6add1b41ee91c94e4d6")	{ 9 fields }	Object
_id	ObjectId("60cab6add1b41ee91c94e4d6")	ObjectId
apyn	Bashirian Jody	String
nroAfiliado	52278787	String
diagnostico	herpes	String
idPrestador	10002	String
prestacion	lopid ud,genfibrozil	String
fechaCreacion	2021-06-16 23:42:53.178-03:00	Date
comments	[2 elements]	Array
[0]	{ 4 fields }	Object
_id	ObjectId("60cab6add1b41ee91c94e4d7")	ObjectId
whoComment	Checkmed	String
textBody	Verifique la medicacion, por favor	String
dateComment	2021-06-16 23:42:53.188-03:00	Date
[1]	{ 4 fields }	Object
_id	ObjectId("60cac020d1b41ee91c94e4d9")	ObjectId
whoComment	Farmacia Caronte	String
textBody	Revise fechas	String
dateComment	2021-06-17 00:38:11.377-03:00	Date
version	2	Int32

- Eliminación de Comentarios

el id del ticket y el del comentario son los parámetros de entrada

```
router.delete("/:idTicket/:idComment", (req, res, next) => {
  const id0 = req.params.idTicket;
  const id1 = req.params.idComment;
  //const session2 = mongoose.startSession();
  // session2.startTransaction();
  Ticket.findById(id0)
    .then((user2) => {
      const address2 = user2.comments.id(id1); // returns a matching
subdocument
      console.log(address2);
      address2.remove();
      user2.save();
      res.status(200).json(user2);
    })
    .then((user2) => {
      console.log(user2)
    })
    .catch(e => {
```



```

console.log(e)
res.status(500).json({
  error: err
});
});

```

Voy a eliminar el 2do comentario que se modifiko anteriormente:

En postman metodo deleteCommand:

deleteComment Examples 0 BI

DELETE http://localhost:3000/comments/60cab6add1b41ee91c94e4d6/60cac020d1b41ee91c94e4d9 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 39 ms Size: 720 B

Pretty Raw Preview Visualize JSON

```

3  "apyn": "Bashirian Jody",
4  "nroAfiliado": "52278787",
5  "diagnostico": "herpes",
6  "idPrestador": "10002",
7  "prestacion": "lopid ud,genfibrozil",
8  "fechaCreacion": "2021-06-17T02:42:53.178Z",
9  "comments": [
10   {
11     "_id": "60cab6add1b41ee91c94e4d7",
12     "whoComment": "Checkmed",
13     "textBody": " Verifique la medicacion, por favor",
14     "dateComment": "2021-06-17T02:42:53.188Z"
15   }
16 ],

```

Y así quedó en la base de datos, donde se puede ver un solo comentario y que cambio la versión:

```
db.getCollection('tickets').find({})
```

tickets 0.002 sec.	
Key	Value
(1) ObjectId("60cab6add1b41ee91c94e4d6")	{ 9 fields }
_id	ObjectId("60cab6add1b41ee91c94e4d6")
apyn	Bashirian Jody
nroAfiliado	52278787
diagnostico	herpes
idPrestador	10002
prestacion	lopid ud,genfibrozil
fechaCreacion	2021-06-16 23:42:53.178-03:00
comments	[1 element]
[0]	{ 4 fields }
_id	ObjectId("60cab6add1b41ee91c94e4d7")
whoComment	Checkmed
textBody	Verifique la medicacion, por favor
dateComment	2021-06-16 23:42:53.188-03:00
version	3

Búsquedas:

Por ejemplo para buscar todos los comentarios creados por el módulo de validación (checkmed)

getAllComments Examples 0 BUILD

GET http://localhost:3000/comments/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 1156 ms Size: 1.81 MB Sa

Pretty Raw Preview Visualize JSON

```

91721
91722     "_id": "60cac39b1c8841efaf383c9a",
91723     "whoComment": "Checkmed",
91724     "textBody": " Verifique la medicacion, por favor",
91725     "dateComment": "2021-06-17T03:38:03.917Z"
91726   },
91727   ],
91728   "version": 0
91729 },
91730 {
91731   "_id": "60cac39b1c8841efaf383ca0",
91732   "apyn": "Dibbert Anna",
91733   "nroAfilado": "18734495",
91734   "diagnostico": "dislipemia",

```

Bootcamp Build Browse

En la terminal:

```

[{"_id": "60bec872b69cb4fd9e02dfad", "whoComment": "Checkmed", "textBody": " Verifique la medicacion, por favor", "dateComment": "2021-06-08T01:31:30.947Z", "version": 0}, {"_id": "60bec872b69cb4fd9e02dfaf", "apyn": "Upton Lulu", "nroAfilado": "59504832", "diagnostico": "herpes", "idPrestador": "10004", "prestacion": "asnavitan", "fechaCreacion": "2021-06-08T01:31:30.951Z", "comment": "Verifique la medicacion, por favor", "dateComment": "2021-06-08T01:31:30.955Z", "version": 0}, {"_id": "60bec872b69cb4fd9e02dfb2", "apyn": "Wunsch Elmo", "nroAfilado": "16899976", "diagnostico": "asma", "idPrestador": "10003", "prestacion": "sinlip", "fechaCreacion": "2021-06-08T01:31:30.956Z", "comment": "Verifique la medicacion, por favor", "dateComment": "2021-06-08T01:31:30.962Z", "version": 0}, {"_id": "60bec872b69cb4fd9e02dfb3", "whoComment": "Checkmed", "textBody": " Verifique la medicacion, por favor", "dateComment": "2021-06-08T01:31:30.962Z", "version": 0}, {"_id": "60bec872b69cb4fd9e02dfb5", "apyn": "Mayer Gilda", "nroAfilado": "41818996", "diagnostico": "herpes", "idPrestador": "10004", "prestacion": "sinlip", "fechaCreacion": "2021-06-08T01:31:30.963Z", "comment": "Verifique la medicacion, por favor", "dateComment": "2021-06-08T01:31:30.970Z", "version": 0}, {"_id": "60bec872b69cb4fd9e02dfb6", "whoComment": "Checkmed", "textBody": " Verifique la medicacion, por favor", "dateComment": "2021-06-08T01:31:30.970Z", "version": 0}, {"_id": "60bec872b69cb4fd9e02dfb8", "apyn": "Leffler Judah", "nroAfilado": "39099017", "diagnostico": "herpes", "idPrestador": "10002", "prestacion": "sinlip", "fechaCreacion": "2021-06-08T01:31:30.987Z", "comment": "Verifique la medicacion, por favor", "dateComment": "2021-06-08T01:31:30.990Z", "version": 0}, {"_id": "60bec872b69cb4fd9e02dfbd", "apyn": "Zboncak Emma", "nroAfilado": "89931867", "diagnostico": "herpes", "idPrestador": "10004", "prestacion": "sinlip", "fechaCreacion": "2021-06-08T01:31:30.993Z", "comment": "Verifique la medicacion, por favor", "dateComment": "2021-06-08T01:31:30.993Z", "version": 0}

```

Palabras finales

Ante todo quería expresar mi agradecimiento a la cátedra. Lamento no haber logrado Reflejar en este trabajo todo lo que nos enseñó este último 2020, actualmente no Tengo un perfil de desarrollador, pero es algo que quiero mejorar y este fue un puntapié Inicial importante.

He visto muchos temas y tome distintos cursos durante el último año antes de llegar a finalizar este trabajo y también profesor le agradezco Que me haya acotado el alcance sino hubiese seguido agregando nuevas tecnologías por demás interesantes .

Parte de la enseñanza y de la vida laboral es entregar a tiempo, y aunque sea sobre el límite pero lo estoy haciendo.

Nuevamente le agradezco y sé que de aquí en más se abren nuevas puertas de conocimiento.

