wawiwa

MODELANDO DATOS NODE.JS

2022 © Wawiwa Tech | Confidential

wawiwa

Modelado de datos - enlace

- El desafío clave en el modelado de datos es equilibrar las necesidades de la aplicación, las características de rendimiento del motor de base de datos y los patrones de recuperación de datos. Al diseñar modelos de datos, siempre considere el uso de los datos por parte de la aplicación (es decir, solicitudes, actualizaciones y procesamiento de los datos), así como la estructura inherente de los datos en sí.

2022 © Wawiwa Tech | Confidential

Modelado de datos - enlace

En esta sección aprenderemos sobre

1. Relaciones entre datos
2. Datos de referencia vs incorporados
3. Tipos de referencias

2022 © Wawiwa Tech | Confidential

wawiwa

Datos Incorporados

- Los documentos incorporados capturan las relaciones entre datos al guardar los datos en una única estructura de documento. Los documentos de MongoDB hacen posible la incorporación de estructuras de documentos en un campo o en un arreglo en un documento. Estos datos *desnormalizados* permiten a las aplicaciones recuperar y manipular datos relacionados en una operación única de la base de datos.

```
{
    _id:"123123123",
    username:"Mikel",
    password:"123",
    contact:{
        phonenumber:"987-654-321",
        email:"abc@stamschema.com"
    },
    permission:{
        type:"admin",
        index:1
    }
}
```

Sub documento incorporado

Sub documento incorporado

Datos de referencia

Los datos de referencia guardan las relaciones entre datos incluyendo enlaces o *referencias* de uno de los documentos al otro. Las aplicaciones pueden resolver estas referencias accediendo a los datos relacionados. En general, estos son modelos de datos *normalizados*.

Documento User

```
{
    _id:"x1",
    username:"Mikel",
}
```

Acceso al documento

```
{
    _id:"id2",
    userId:"x1",
    phonenumber:"987-654-321",
    email:"abc@stamschema.com"
}
```

```
{
    _id:"id2",
    userId:"x1",
    phonenumber:"12-12-121",
    email:"zx@wawewa.com"
}
```

wawiwa

Relaciones entre datos

1:1 course -> title 1 “course” puede tener solamente un “title”

muchos:muchos course -> lesson 1 “lesson” puede tener muchos”lessons” y un “lesson” puede tener muchos “courses”

1:muchos course -> comments -1 “course” puede tener muchos “ comments”

wawiwa

1:muchos - enlace

1:muchos(**algunos**) course -> award -1 “course” puede tener algunos “awards”

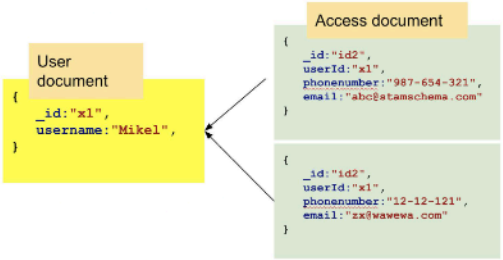
1:muchos(**tonelada**) course -> logs -1 “course” puede tener millones de “logs”

1:muchos(**muchos**) course -> comment -1 “course” puede tener muchos “comments”

desnormalizado

Con MongoDB, puedes incorporar datos relacionados en una única estructura o documento. Este esquema conocido generalmente como “desnormalizado” modela y se aprovecha de los ricos documentos de MongoDB. Considera el siguiente diagrama:

Los modelos de datos normalizados describen relaciones. utilizando referencias entre documentos.




```
{
    _id:"123123123",
    username:"Mikel",
    password:"123",
    contact:{
        phonenumber:"987-654-321",
        email:"abc@stamschema.com"
    },
    permission:{
        type:"admin",
        index:1
    }
}
```

Referenciar vs. incorporar (**de**normalización->)

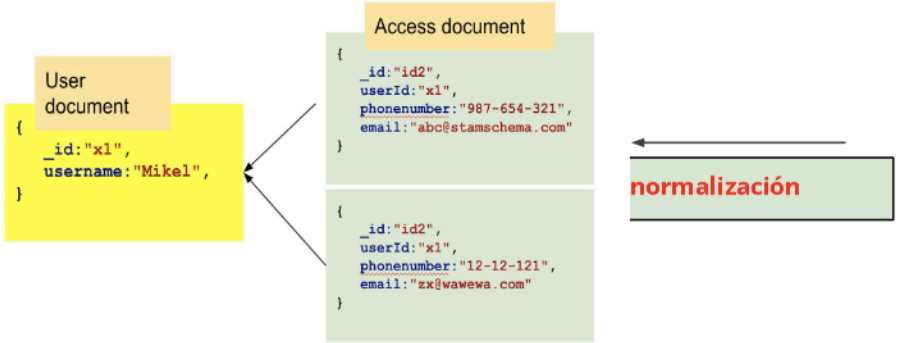
Referencia

😊 Rendimiento: es más fácil solicitar cada documento por sí mismo

😟 Necesitamos 2 solicitudes para obtener los datos

desnormalización

```
{
    _id:"123123123",
    username:"Mikel",
    password:"123",
    contact:{
        phonenumber:"987-654-321",
        email:"abc@stamschema.com"
    },
    permission:{
        type:"admin",
        index:1
    }
}
```



Incorporar

😊 Rendimiento: podemos obtener todos los datos en una solicitud

😟 Imposible consultar el documento incorporado por el mismo

wawiwa

¿Cómo decidir si incorporar o referenciar?

En general, usa modelos de datos normalizados:

- la incorporación daría como resultado la duplicación de datos pero no proporciona suficientes ventajas de rendimiento de lectura para compensar las implicaciones de la duplicación.
- Para representar más relaciones complejas muchos-a-muchos.
- Para modelar conjuntos de datos jerárquicos.

2022 © Wawiwa Tech | Confidential

wawiwa

¿Cómo decidir si incorporar o referenciar?

1. Tipos de relaciones
2. Cada cuanto se escriben y leen datos
3. Cuántos de los datos están relacionados

	Incorporar	Referencias	
1:muchos 1:pocos			1:muchos 1:toneladas muchos a muchos
los datos casi solamente se leen los datos no cambian muy rápido			los datos se actualizan mucho baja lectura/escritora de course + comments
los conjuntos de datos pertenecen juntos. Por ejemplo, user y country			frecuentemente debes consultar los dos conjuntos de datos por ellos mismos

2022 © Wawiwa Tech | Confidential

Tipos de referencias

Referenciando a los padres

Referencias de dos lados

Referenciando a los hijos

wawiwa

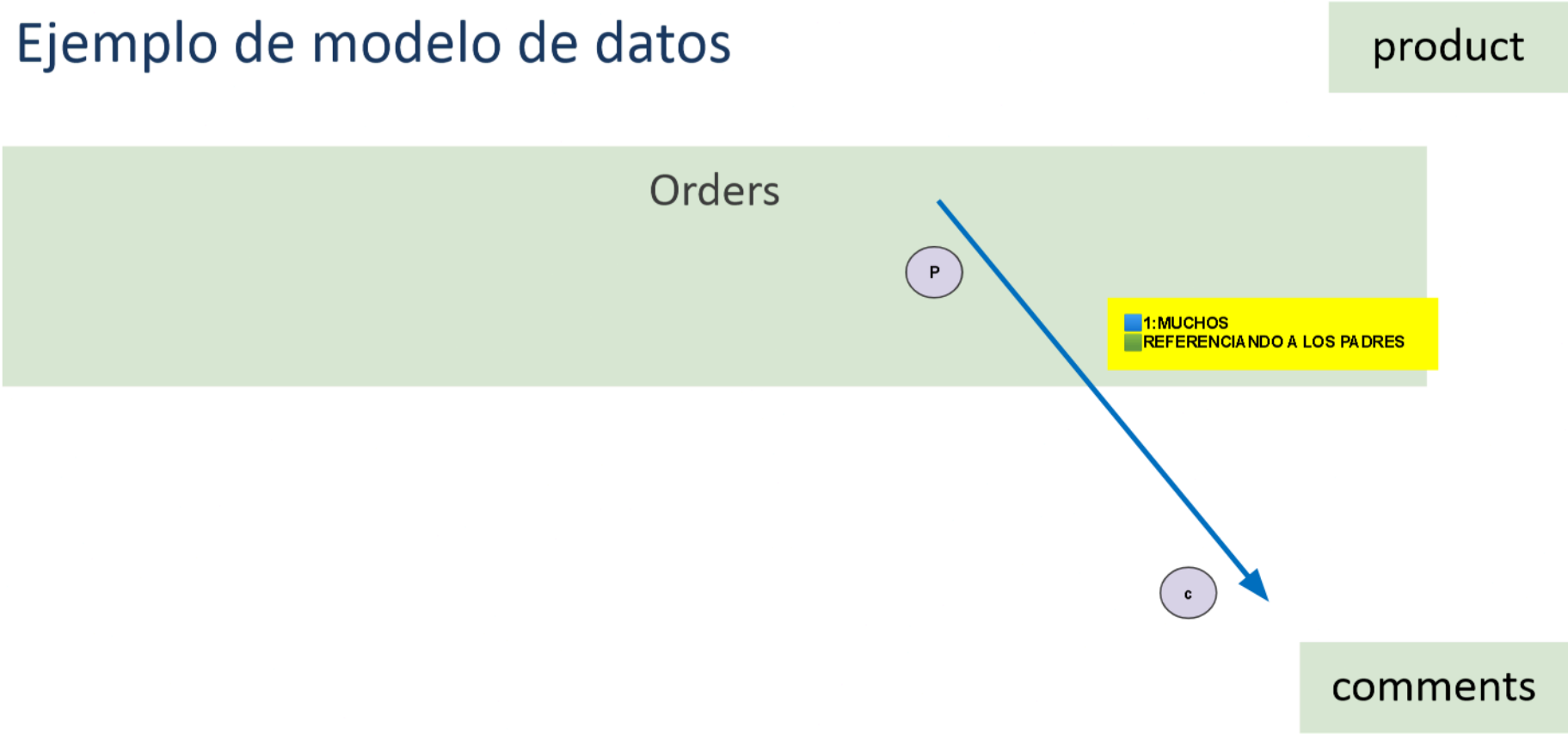
Ejemplo de modelo de datos

product

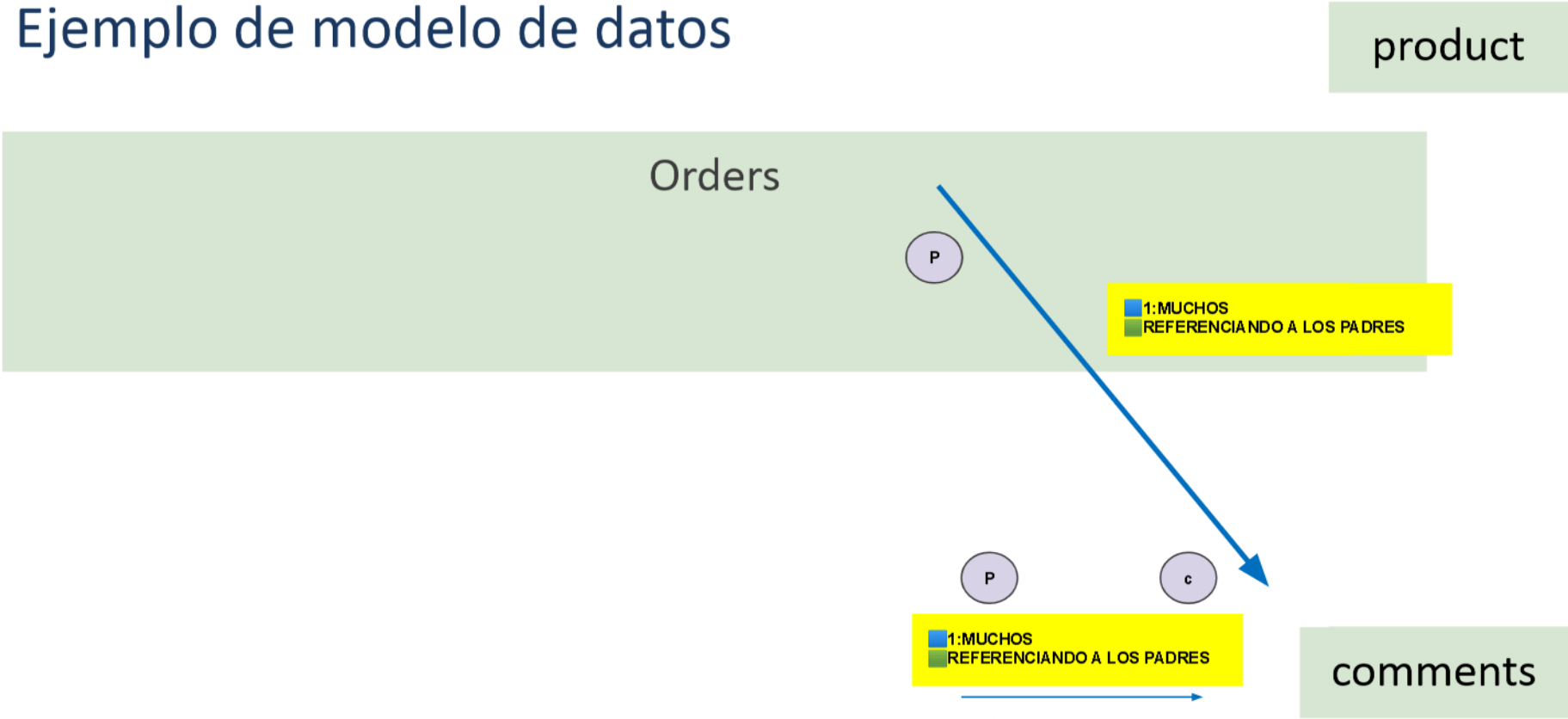
Orders

comments

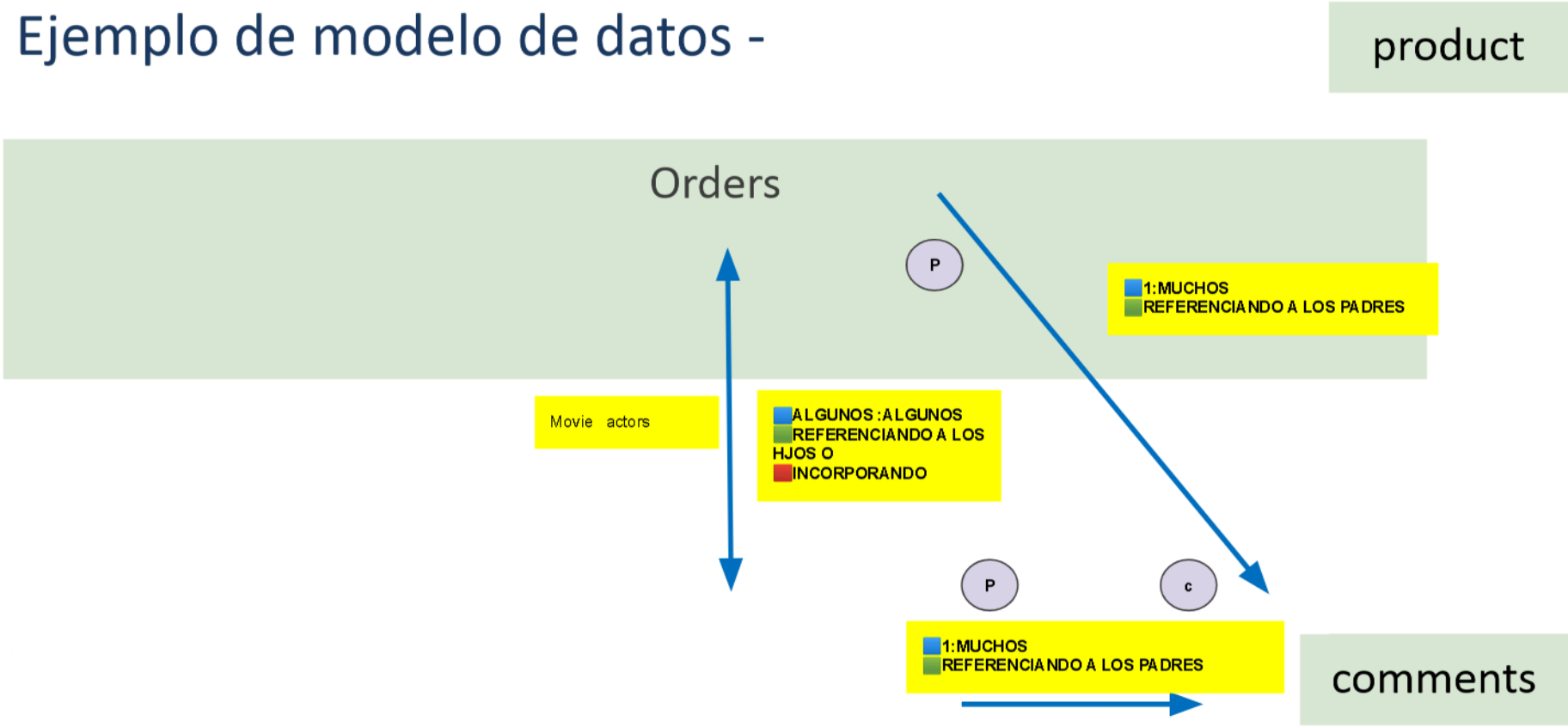
Ejemplo de modelo de datos



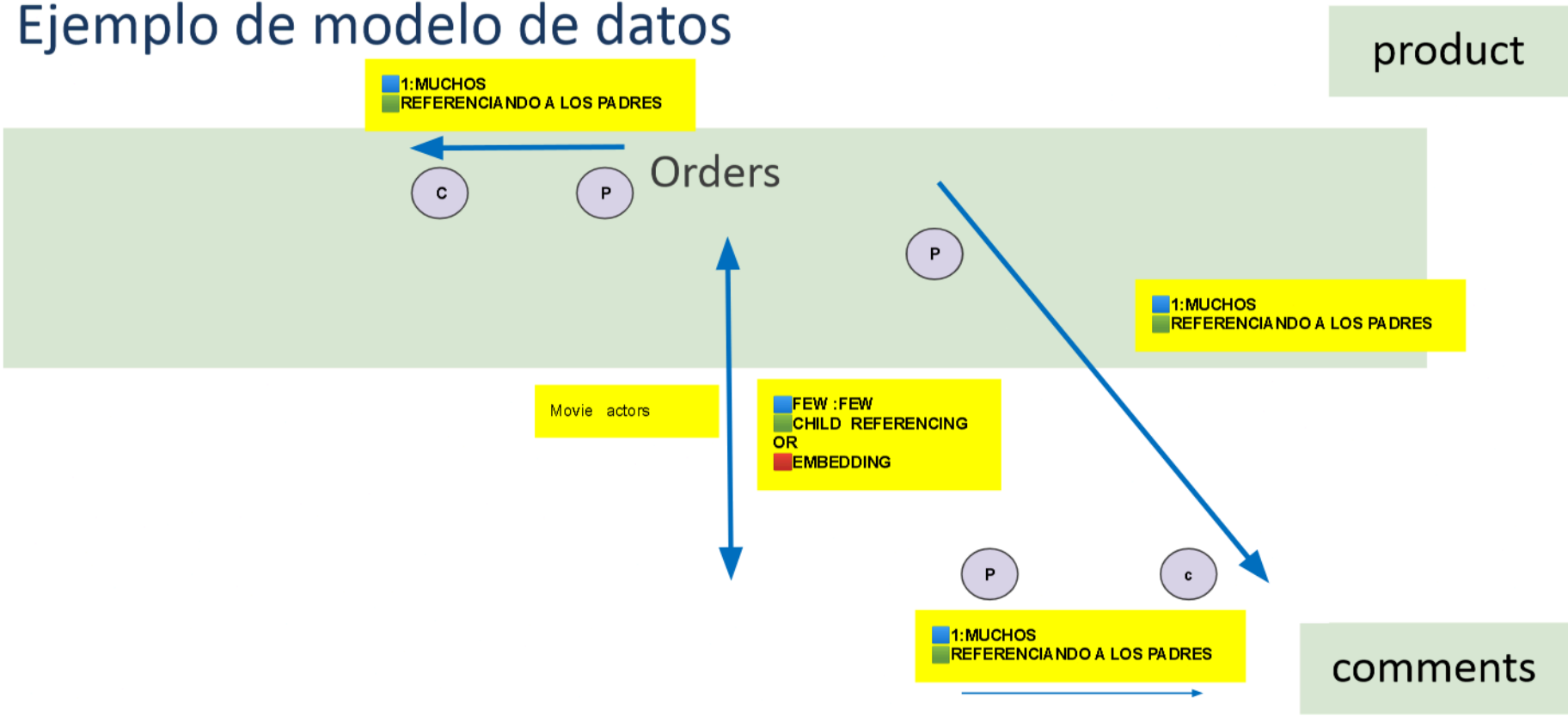
Ejemplo de modelo de datos



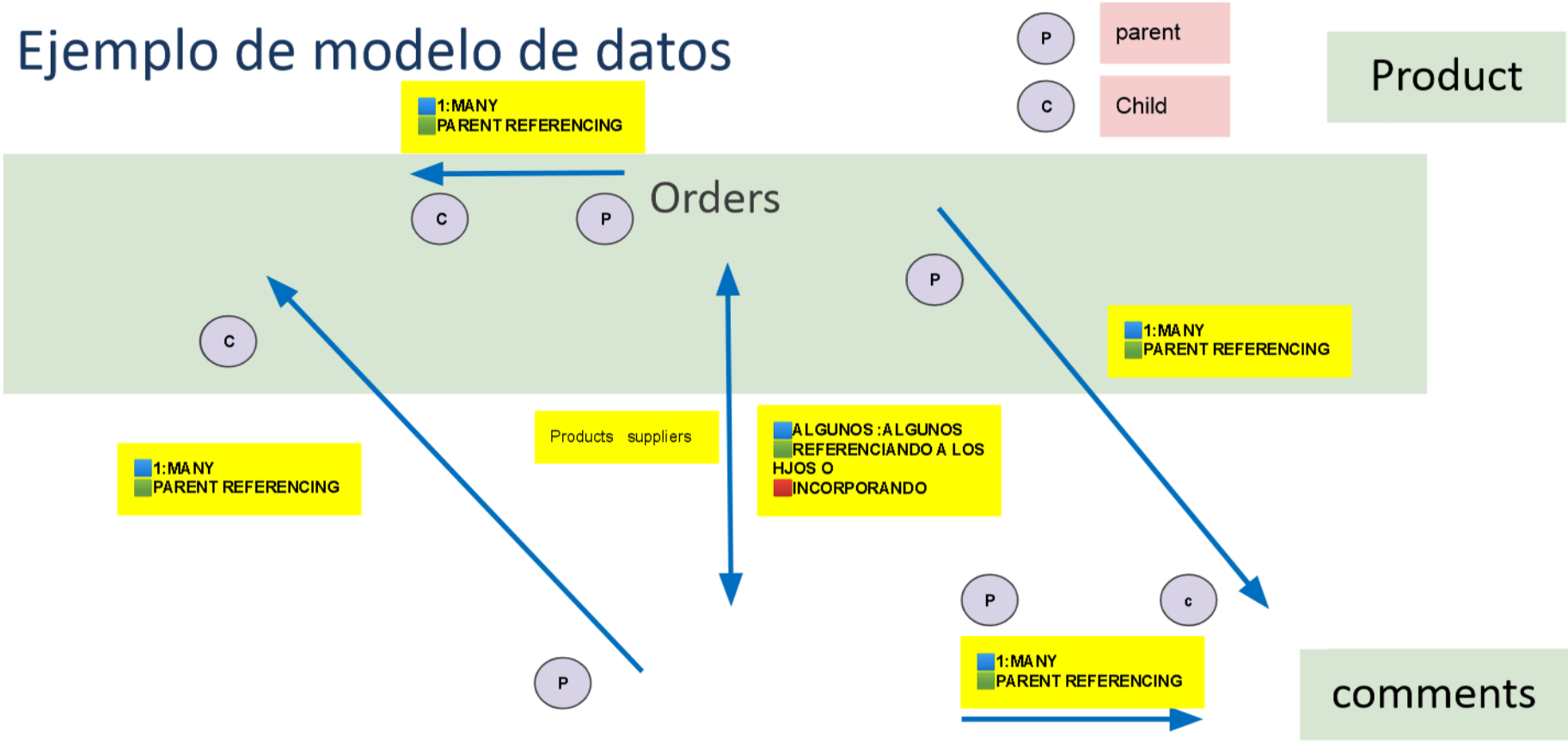
Ejemplo de modelo de datos -



Ejemplo de modelo de datos



Ejemplo de modelo de datos



wawiwa

Modelo de “product” viejo

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const ProductSchema = new Schema({
   title: {
       type:String,
       require:[true,'A product must have title' ],
       unique:true,
       trim:true
   },
   description:{
       type:String,
       minlength:[5,'Description is minimum 20 characters'],
       maxlength:[1000,'Description is maximum 1000 characters']
   },
   price:{
       type:Number,
       required:[true, 'A product must have price'],
       min:[0,'price must be above 0'],
       max:[10000,'price must be below 10000']

   },
module.exports = mongoose.model('product', ProductSchema);
```

2022 © Wawiwa Tech | Confidential

wawiwa

Modelando “product” - referenciando **hijos**

Agrega estos campos al modelo de “Product” viejo

```
imageCover:{
    type:String,
    required:[true,'A tour must have a cover image']
},
images:[String],
suppliers:[
    {
    type:mongoose.Schema.ObjectId,
    ref:'user'
    }

],//array of suppliers (supplier is _id of

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

“Image” del “product” principal

Más “images” de “product”

Arreglo de “suppliers” de “product” - referenciando al hijo

wawiwa

Ejemplo de 2 users

```
_id: ObjectId("61e989a227e4f00bb437744a")
email: "test@gmail.com"
password: "$2a$10$1zhPFGuc/7f1nZMN7020S.UgKUKEYmOCjjaeTLEn0ShZh60IngBjy"
firstname: "mikel"
lastname: "lewis"
permission: "user"
active: true
created: 2022-01-20T16:11:14.889+00:00
modified: 2022-01-20T16:11:14.889+00:00
__v: 0
```

```
_id: ObjectId("61efb0546f1fe4bb97941438")
email: "test1@gmail.com"
password: "$2a$10$LIB.mB8OtwSfA9rmenKQNeFWU.Yl37ZP7k7HIhtBQ7dZVERmuk1Ee"
firstname: "mikel"
lastname: "lewis"
permission: "admin"
active: true
created: 2022-01-25T08:09:56.480+00:00
modified: 2022-01-25T08:09:56.480+00:00
__v: 0
```

wawiwa

Agrega un nuevo “product”

```
"title": "ball-10",
"description": "This is a white ball",
"price": 1000,
"created": "2023-10-09T21:00:00.000Z",
"suppliers":["61e989a227e4f00bb437744a","61efb0546f1fe4bb97941438"],
"imageCover":"c.png"
```

Arreglo de “suppliers” en “product” - referenciando al hijo

Agrega un nuevo “product”

POST localhost:3000/api/v1/products

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "title": "ball-10",
    "description": "This is a white ball",
    "price": 1000,
    "created": "2023-10-09T21:00:00.000Z",
    "suppliers":["61e989a227e4f00bb437744a","61efb0546f1fe4bb97941438"],
    "imageCover":"c.png"

}
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "status": "success",
    "data": {
        "title": "ball-10",
        "description": "This is a white ball",
        "price": 1000,
        "imageCover": "c.png",
        "images": [],
        "suppliers": [
            "61e989a227e4f00bb437744a",
            "61efb0546f1fe4bb97941438"
        ],
        "created": "2023-10-09T21:00:00.000Z",
        "_id": "61f0e29c1697c0cc37625968",
        "__v": 0
```

Arreglo de “suppliers” en “product” - referenciando al hijo

Este “product” tiene 2 “suppliers”

Trae “product”

GET localhost:3000/api/v1/products

Params Authorization Headers (9) Body Pre-request Script Tests Se

Query Params

KEY	VALUE

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "status": "success",
    "data": [
        {
            "_id": "61f0e29c1697c0cc37625968",
            "title": "ball-10",
            "description": "This is a white ball",
            "price": 1000,
            "imageCover": "c.png",
            "images": [],
            "suppliers": [
                "61e989a227e4f00bb437744a",
                "61efb0546f1fe4bb97941438"
            ],
            "created": "2023-10-09T21:00:00.000Z",
            "__v": 0
        }
    ]
}
```


Necesitamos popular los detalles de “user”

Popular

MongoDB tiene el operador de agregación $lookup similar a una unión en versiones >= 3.2. Mongoose tiene una alternativa más poderosa llamada populate(), **que le permite hacer referencia a documentos de otras colecciones**.

Rellenar es el proceso de reemplazar automáticamente las rutas especificadas en el documento con documentos de otras colecciones. Podemos completar un solo documento, varios documentos, un objeto simple, varios objetos simples o todos los objetos devueltos por una consulta. Veamos algunos ejemplos.

Nuestro modelo de Producto tiene su campo de proveedores configurado en una serie de **ObjectIds**. La opción **ref** es lo que le dice a Mongoose qué modelo usar durante la **población**, en nuestro caso el modelo de usuario. Todos los _ids que almacenamos aquí deben ser _ids de documentos del modelo Story.

Nota: ObjectId, Number, String y Buffer son válidos para usarse como referencias. Sin embargo, debería utilizar ObjectId a menos que sea un usuario avanzado y tenga una buena razón para hacerlo.

2022 © Wawiwa Tech | Confidential

wawiwa

Agrega “populate”

```
//lee por su id
exports.getProductById =  function(req, res, next){
   let id = req.params.id

CurrentProduct.find({_id:id}).populate('user').then(function(data){
        res.status(200).json({
            status:"success",
            data:data
        })
    }).catch(err=>{
        res.status(404).json({
            status:"fail",
            message:"error:😱" + err
        })
    })
}
```

Esto no va a funcionar


```
{
  "status": "fail",
  "message": "error:😱MongooseError: Cannot populate path `user` because it is not in your schema. Set the `strictPopulate` option to false to override."
}
```

wawiwa

Agrega “populate”

```
//read by id
exports.getProductById =  function(req, res, next){
    let id = req.params.id
    CurrentProduct.find({_id:id}).populate({
        path:'suppliers',
        select:'-password -passwordChangedAt -passwordResetToken'
    }
).then(function(data){
        res.status(200).json({
            status:"success",
            data:data
        })
    }).catch(err=>{
        res.status(404).json({
            status:"fail",
            message:"error:😱" + err
        })
    })
}
```

Ruta

Puedes seleccionar campos para eliminar o agregar

Más sobre popular y ruta: enlace

wawiwa

Respuesta con campos de “user”

localhost:3000/api/v1/products/61f0e29c1697c0cc37625968

GET localhost:3000/api/v1/products/61f0e29c1697c0cc37625968

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON


```
8        "price": 1000,
9        "imageCover": "c.png",
10       "images": [],
11       "suppliers": [
12           {
13               "_id": "61e989a227e4f00bb437744a",
14               "email": "test@gmail.com",
15               "firstname": "mikel",
16               "lastname": "lewis",
17               "permission": "user",
18               "active": true,
19               "created": "2022-01-20T16:11:14.889Z",
20               "modified": "2022-01-20T16:11:14.889Z",
21               "__v": 0
22           },
23           {
24               "_id": "61efb0546f1fe4bb97941438",
25               "email": "test1@gmail.com",
26               "firstname": "mikel",
27               "lastname": "lewis",
28               "permission": "admin",
29               "active": true,
30               "created": "2022-01-25T08:09:56.480Z",
31               "modified": "2022-01-25T08:09:56.480Z",
32               "__v": 0
33           }
```

wawiwa

Agregando middleware a todas las consultas

```
ProductSchema.pre(/^find/, async function(next){
    this.populate({
        path:'suppliers',
        select:'-password -passwordChangedAt -passwordResetToken'
    })
    next();
})
```

Ahora todas las solicitudes **find** retornar los campos “suppliers”

wawiwa

COMENTARIOS DE MODELADO - REFERENCIA DE PADRES

wawiwa

Comentarios de modelado: referencias de padres

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const CommentSchema = new Schema({
   title: {
       type:String,
       require:[true,'A comment must have title' ],
   },
   description:{
       type:String,
       minlength:[5,'Description is minimum 20 characters'],
       maxlength:[1000,'Description is maximum 1000 characters']
   },
   rating:{
       type:Number,
       required:[true, 'A comment must have rating'],
       min:1,
       max:5
   },
   product:{
       type:mongoose.Schema.ObjectId,
       ref:'product',
       required:[true, 'Product must belong to a comment']
   },
   user:{
       type:mongoose.Schema.ObjectId,
       ref:'user',
       required:[true, 'Product must belong to a user']

   },
   created: Date
});
module.exports = mongoose.model('comment', CommentSchema);
```

Referenciando a los padres

Referenciando a los padres

wawiwa

Agrega CommentController.js

```
const CurrentComment = require('./CommentModel')
exports.createComment = async function(req, res, next) {
   try{
       let p1= req.body;
       const newItem = await CurrentComment.create(p1);
       res.status(201).json({
           status:"success",
           data:newItem
       })
   }
   catch(err){
       res.status(400).json({
           status:"fail",
           message:"error:😱" + err
       })
   }
};
```

wawiwa

Agrega CommentController.js

```
exports.getAllComments = async function(req, res, next){
    try{
        const comments = await CurrentComment.find()
        res.status(200).json({
            status:"success",
            data:comments
        })
    }
    catch(err){
        res.status(400).json({
            status:"fail",
            message:"error:😱" + err
        })
    }
}
```

wawiwa

Agrega CommentRouter.js

recuerda: ☝️Solamente usuarios conectados pueden agregar “comment”

```
const express = require('express')
const CommentRouter = express.Router();
const CommentController = require('./CommentConroller')
const authController = require('../users/AuthController')
CommentRouter.post('/',authController.protectSystem,CommentControll
er.createComment);
CommentRouter.get('/',CommentController.getAllComments);
module.exports = CommentRouter;
```

wawiwa

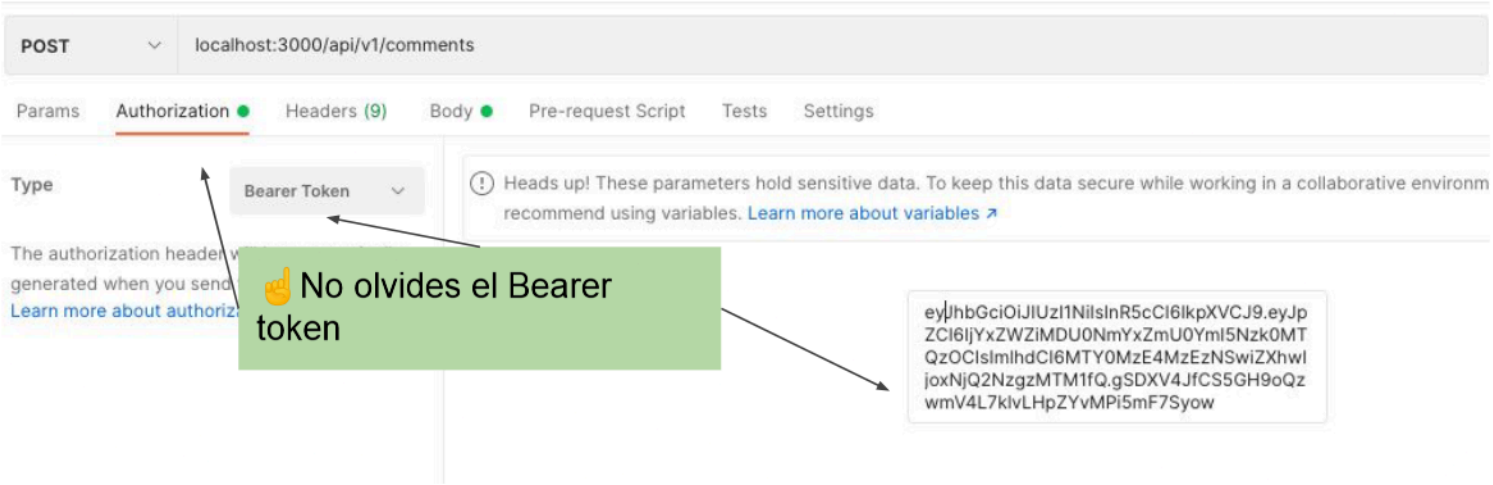
Agrega commentRouter to app.js

```
const express = require('express');
const app = express();
app.use(express.json())
const mongoose = require('mongoose');

const productRouter = require("./api/products/ProductRouter")
const userRouter = require("./api/users/UserRouter")
const commentRouter = require("./api/comments/CommentRouter")

app.use('/api/v1/products', productRouter);
app.use('/api/v1/users',userRouter);
app.use('/api/v1/comments',commentRouter);
```

2022 © Wawiwa Tech | Confidential

wawiwa

Agrega “comment” - No te olvides de estar conectado



wawiwa

Agregar “comment”

POST localhost:3000/api/v1/comments

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "title":"great product",
    "deccriptuion":"comment description",
    "rating":5,
    "product":"61f0e29c1697c0cc37625968",
    "user":"61e989a227e4f00bb437744a"

}
```

Body Cookies Headers (7) Test Results Status: 201 C

Pretty Raw Preview Visualize JSON

```
{
    "status": "success",
    "data": {
        "title": "great product",
        "rating": 5,
        "product": "61f0e29c1697c0cc37625968",
        "user": "61e989a227e4f00bb437744a",
        "_id": "61f0fc48bbb7b1c6782fe346",
        "__v": 0
    }
}
```

Parent references

wawiwa

Populando “comments” - agrega a CommentModel.js

```
CommentSchema.pre(/^find/, async function(next){
    this.populate({
        path:'product',
        select:'title -suppliers'
    }).populate({
        path:'user',
        select:'firstname lastname -_id'
    })
    next();
})
```

Referenciando al padre

Selecciona solamente los campos title, delete suppliers

Referenciando al padre

-Selecciona solamente los campos firstname y lastname -delete _id

wawiwa

Populando “comments”

GET localhost:3000/api/v1/comments

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON


```
1  {
2      "status": "success",
3      "data": [
4          {
5              "_id": "61f0fc48bbb7b1c6782fe346",
6              "title": "great product",
7              "rating": 5,
8              "product": {
9                  "_id": "61f0e29c1697c0cc37625968",
10                 "title": "ball-10"
11             },
12             "user": {
13                 "firstname": "mikel",
14                 "lastname": "lewis"
15             },
16             "__v": 0
17         }
18     ]
19 }
```


Referenciando al padre

Referenciando al padre

wawiwa

¿Preguntas?