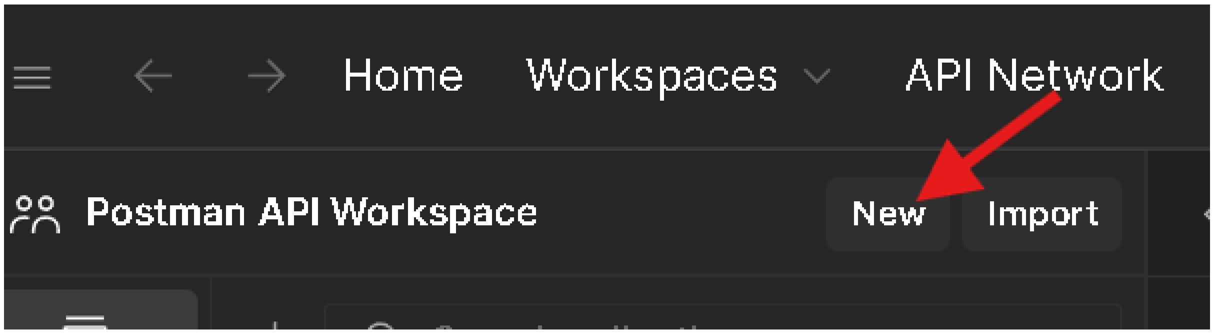
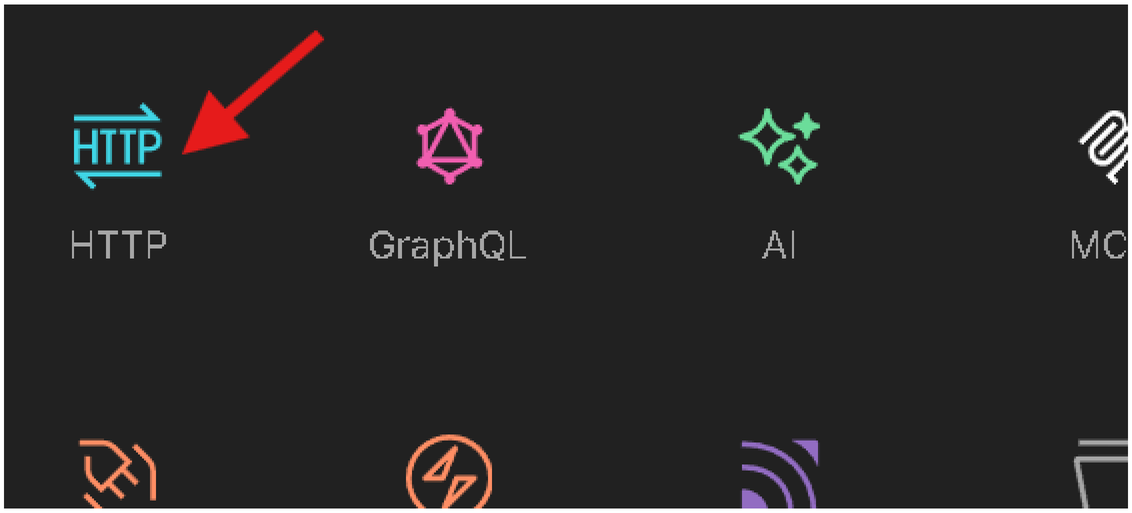
Code in notesController.js:

```
import Note from "../models/Note.js"

export async function getAllNotes(req, res) {
    try {
        // Note.find(), to get every single note
        const notes = await Note.find()
        // Send status, send notes as json
        res.status(200).json(notes)
    } catch (error) {
        // For debugging purpose
        console.error("Error in getAllNotes controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}

export function createNote(req, res) {
    // Gets data in form of json
    res.status(201).json({message: "Note created successfully!"})
}

export function updateNote(req, res) {
    res.status(200).json({message: "Note updated successfully!"})
}

export function deleteNote(req, res) {
    res.status(200).json({message: "Note deleted successfully!"})
}
```

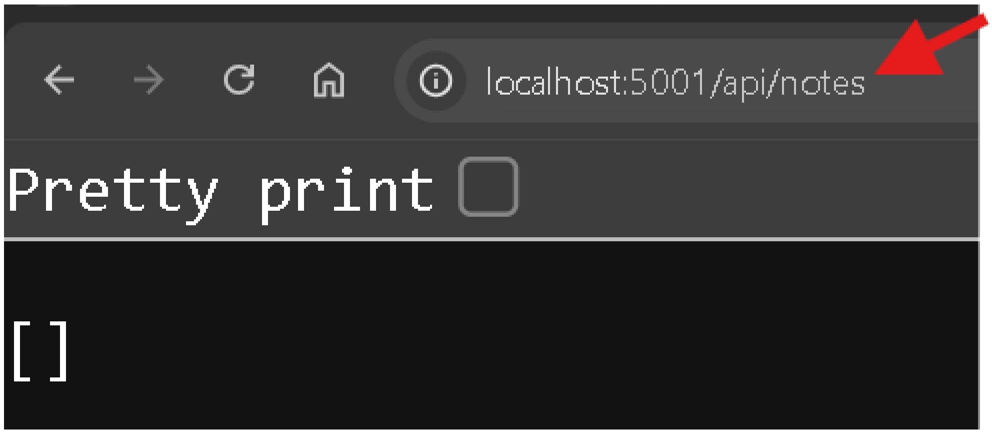
Start using Postman for HTTP Requests:

To test the http request:



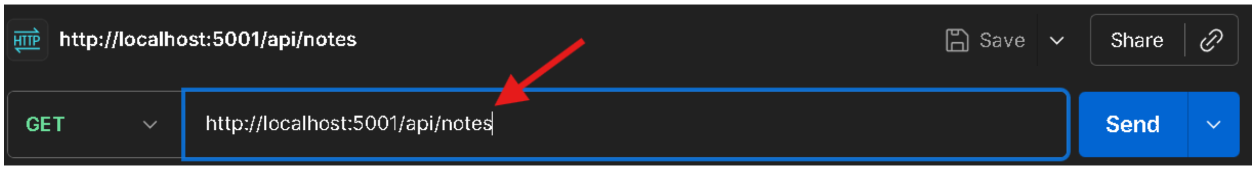


Copy the URL:

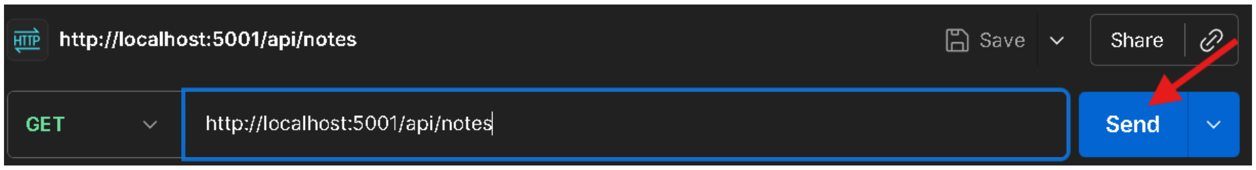


```
[]
```

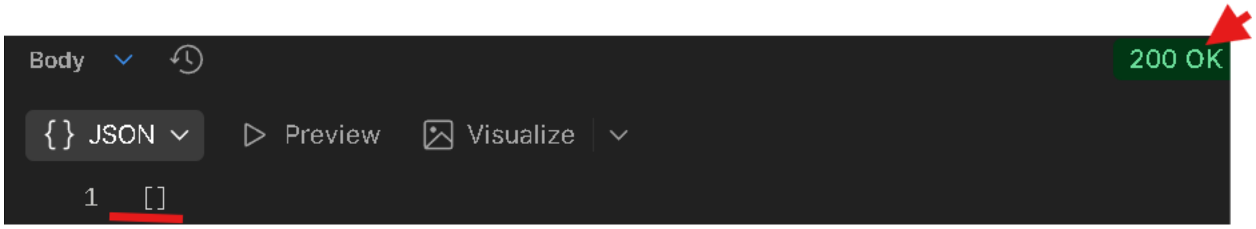
Paste it:


http://localhost:5001/api/notes
Save
Share
GET
http://localhost:5001/api/notes
Send

Send request:

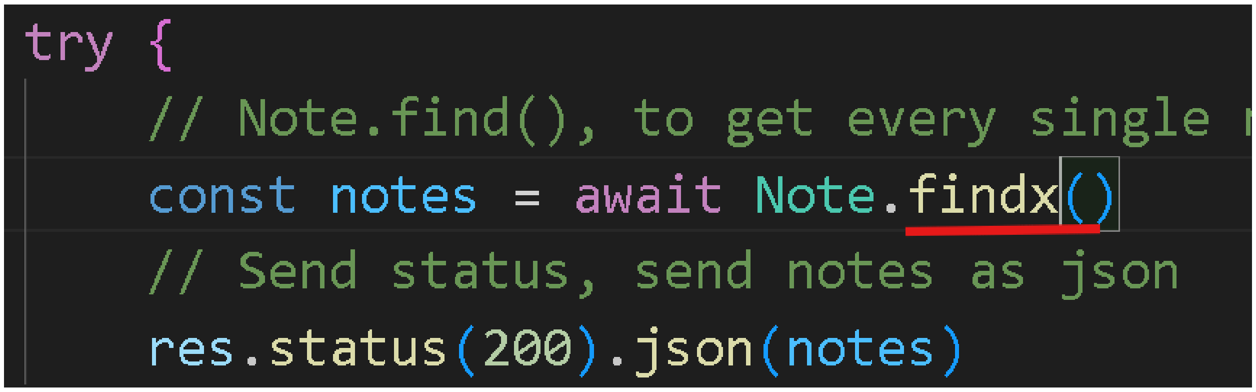

http://localhost:5001/api/notes
Save
Share
GET
http://localhost:5001/api/notes
Send

Got response:



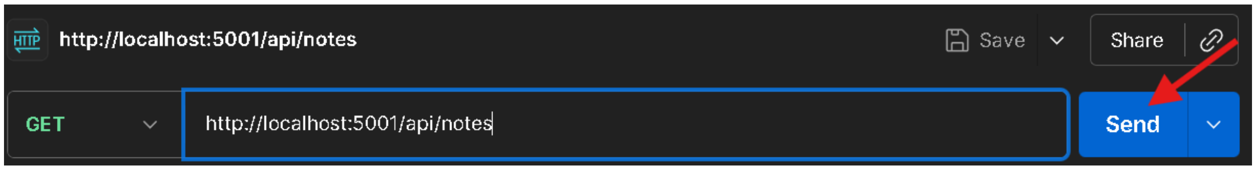
```
[]
```

If the code breaks at try block:
Changed from find to findx



```
try {
    // Note.find(), to get every single r
    const notes = await Note.findx()
    // Send status, send notes as json
    res.status(200).json(notes)
```

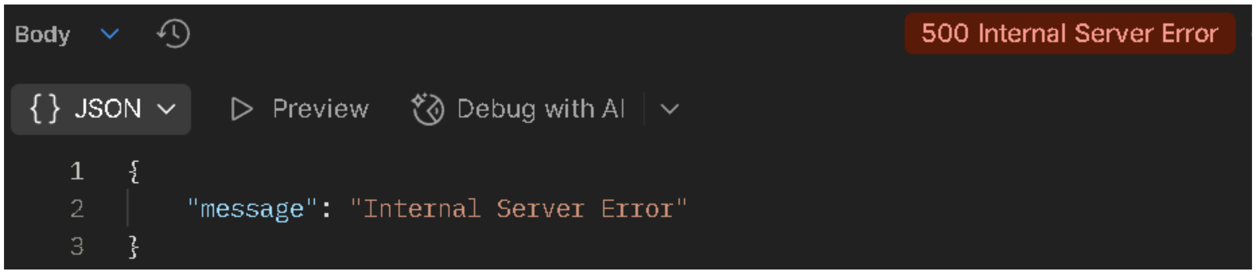
Send request:



http://localhost:5001/api/notes

GET http://localhost:5001/api/notes Send

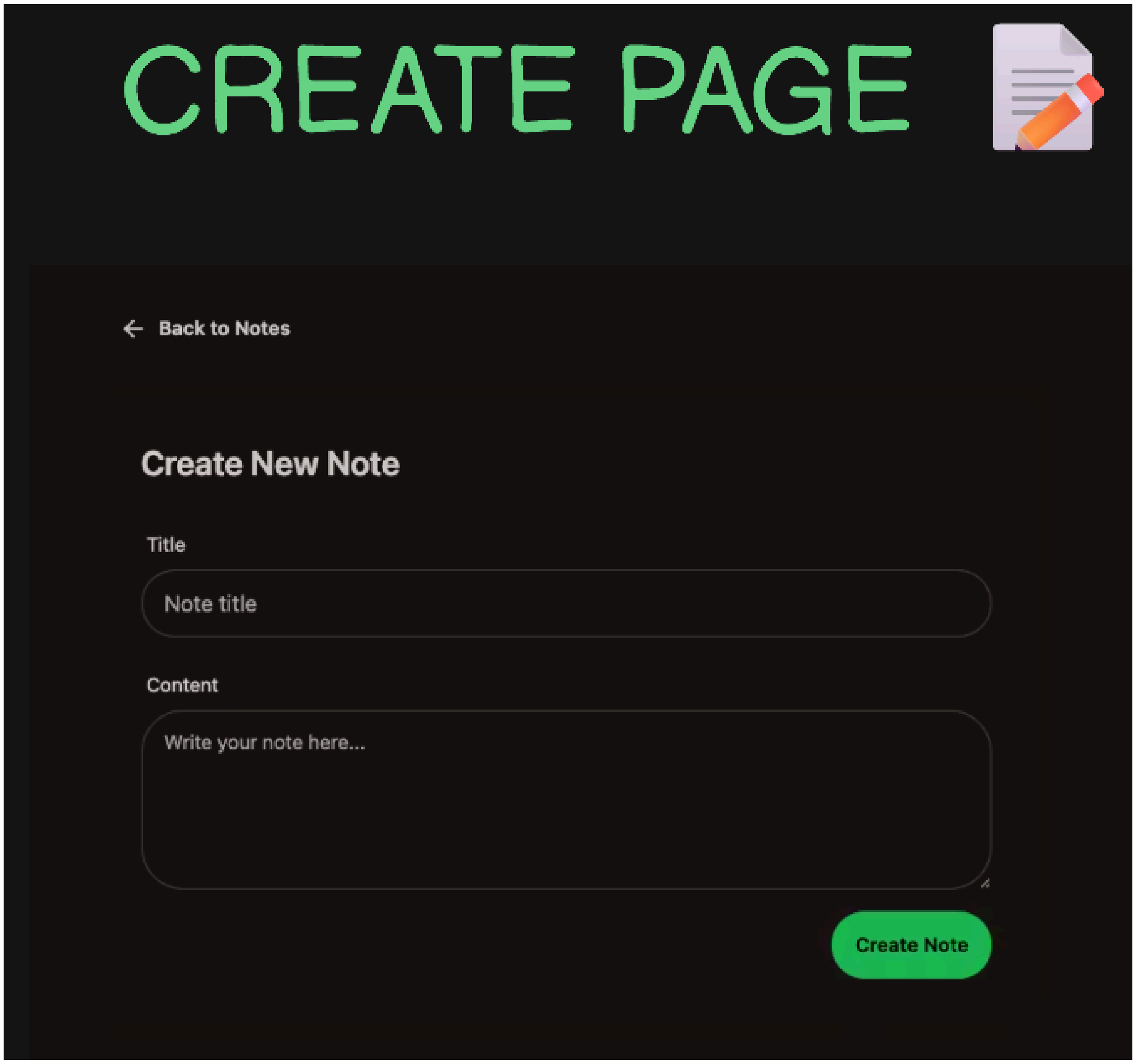
Got response:



500 Internal Server Error

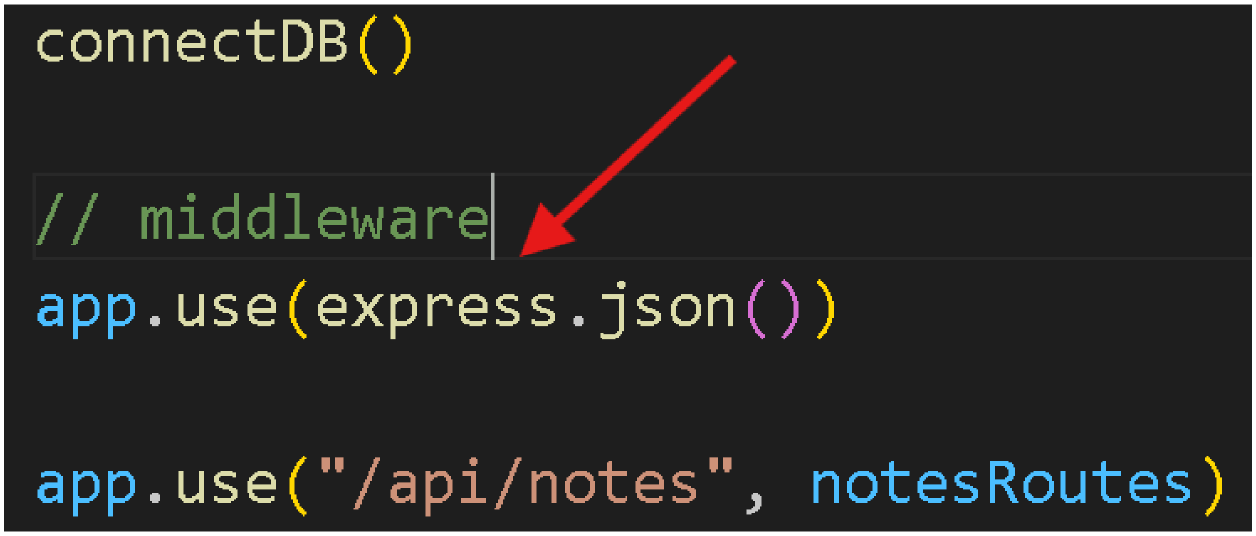
```
{
    "message": "Internal Server Error"
}
```

For creating Notes:



```
export async function createNote(req, res) {
    // If user want to create a notes:
    // includes title, content
    try {
        // title and content comes from req.body
        const {title, content} = req.body
        // by default we can't access this value
        // To access them(or console it),
        // Go to server.js, just before the routes
        // add the code -> app.use(express.json())
        // which is a middleware that we add
        console.log(title, content)
    } catch (error) {

    }
}
```

In server.js:

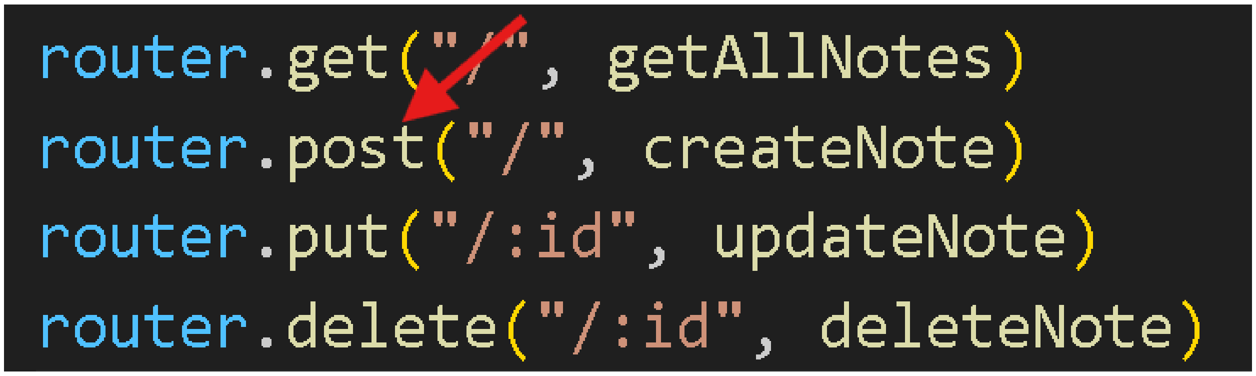


```
connectDB()

// middleware
app.use(express.json())

app.use("/api/notes", notesRoutes)
```

Create note function:

```
export async function createNote(req, res) {
    // If user want to create a notes:
    // includes title, content
    try {
        // title and content comes from req.body
        const {title, content} = req.body
        // by default we can't access this value
        // To access them(or console it),
        // Go to server.js, just before the routes
        // add the code -> app.use(express.json())
        // which is a middleware that we add
        // console.log(title, content)

        // const newNote = new Note({title:title, content: content})
        // Since key and value are the same, so the above code can
        // be replaced as
        const newNote = new Note({title, content})

        await newNote.save()
        res.status(201).json({message: "Note created Successfully"})
    } catch (error) {
        console.error("Error in createNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
```

}
Since createNotes is the post request:

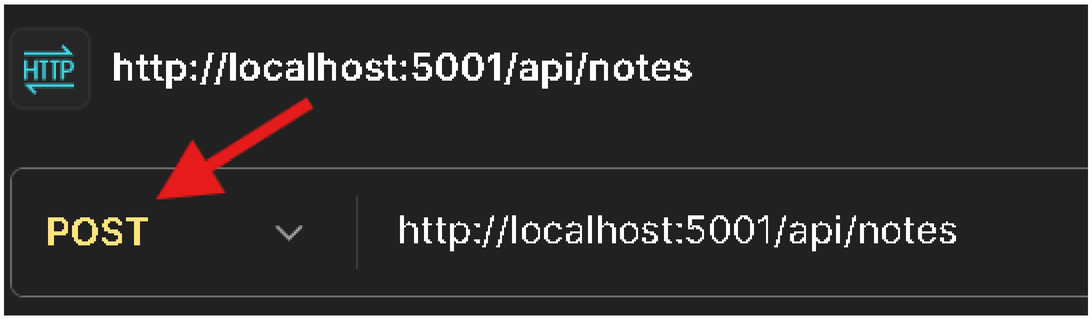


```
router.get("/", getAllNotes)
router.post("/", createNote)
router.put("/:id", updateNote)
router.delete("/:id", deleteNote)
```

Lets see it in action:

In postman:

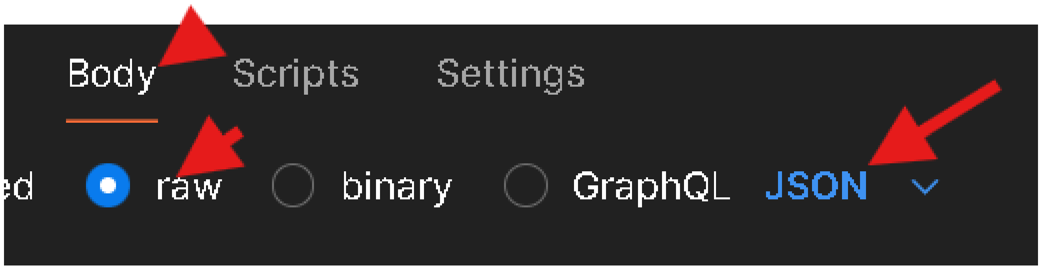
Change it to **post:**



http://localhost:5001/api/notes

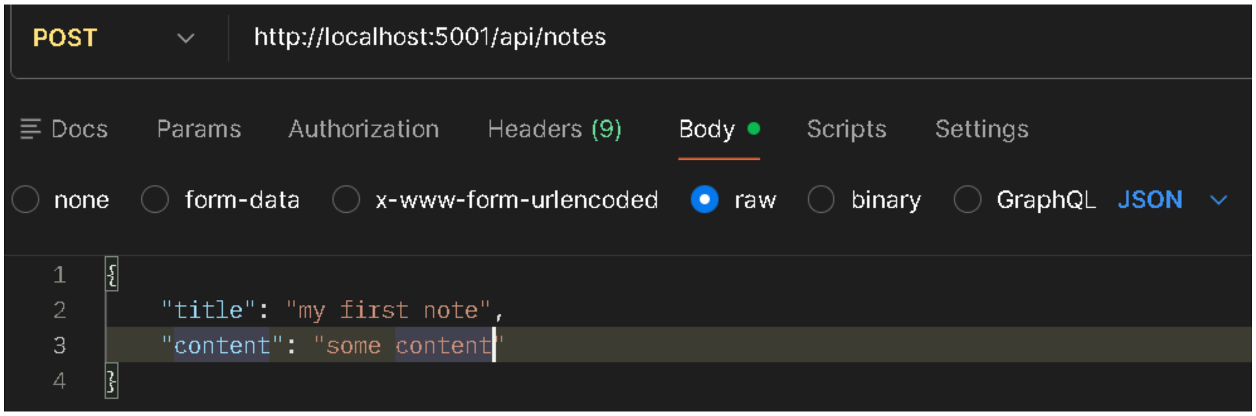
POST http://localhost:5001/api/notes

Go to body, make to raw, change to JSON:



Write title and content:

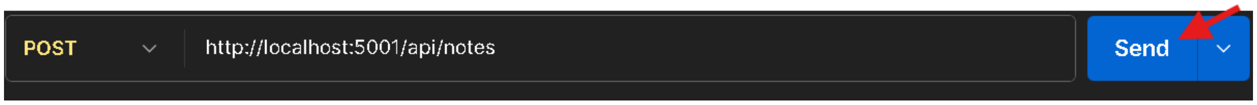
```
{
    "title": "my first note",
    "content": "some content"
}
```



```
http://localhost:5001/api/notes
```

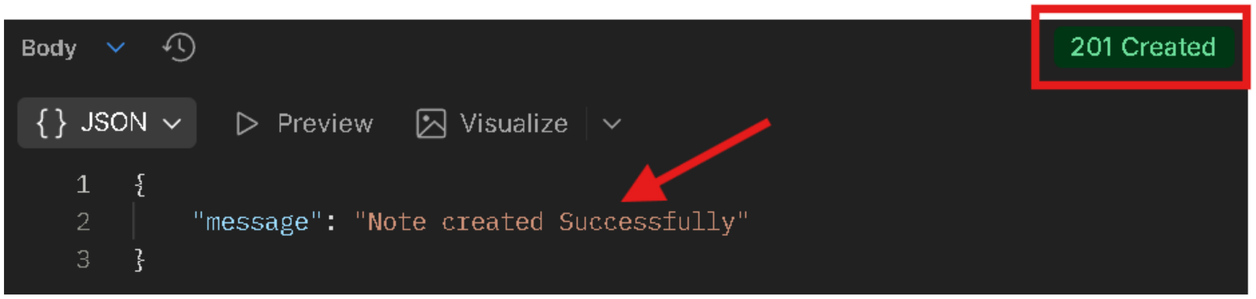
```
{
    "title": "my first note",
    "content": "some content"
}
```

Send **post request:**



```
http://localhost:5001/api/notes
```

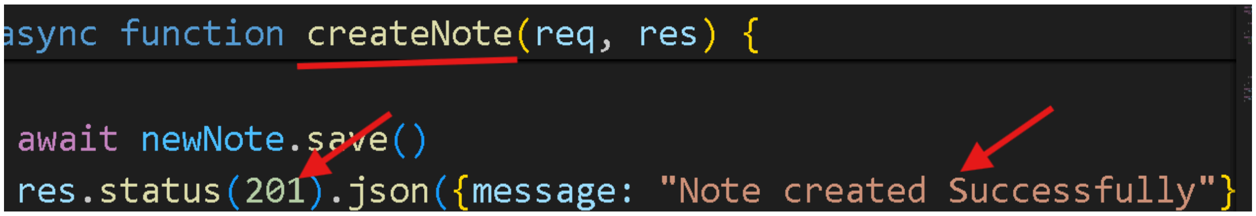
Got response:



201 Created

```
{
    "message": "Note created Successfully"
}
```

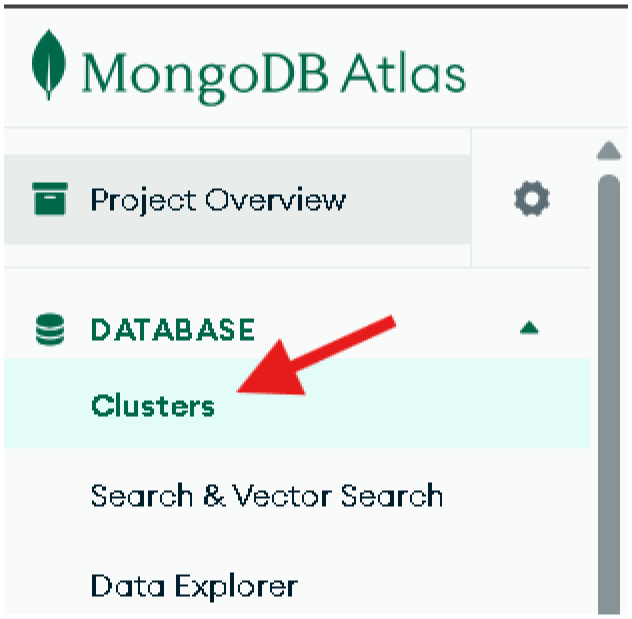
As per the code:

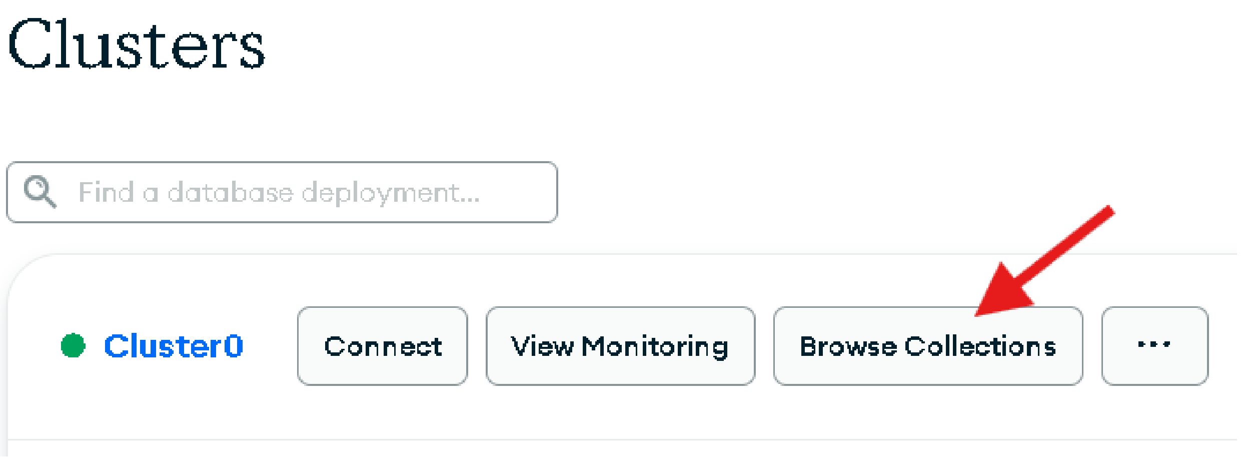


```
async function createNote(req, res) {

 await newNote.save()
 res.status(201).json({message: "Note created Successfully"}
```

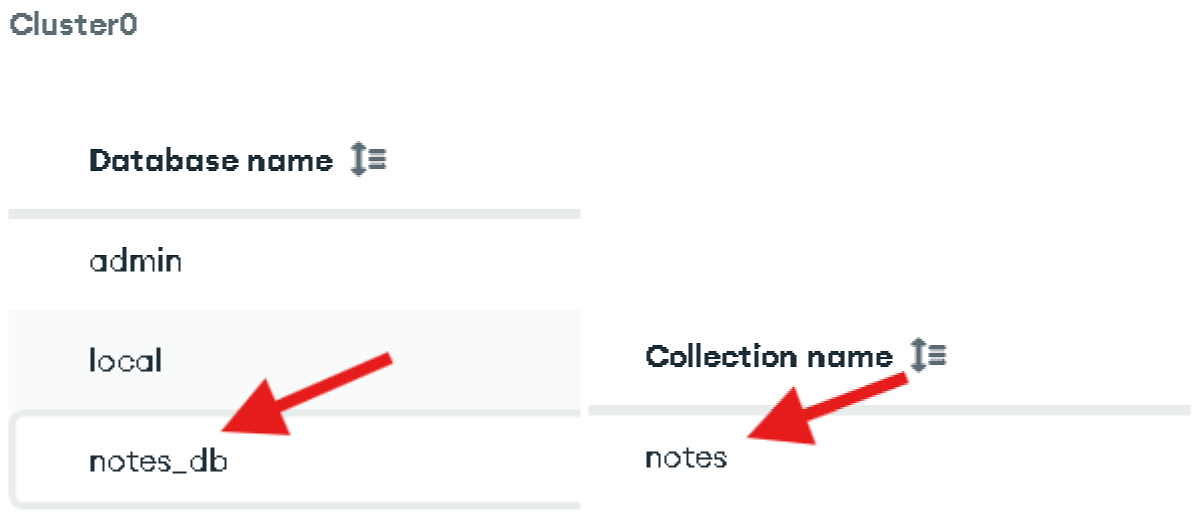
Check database:

Login using google.

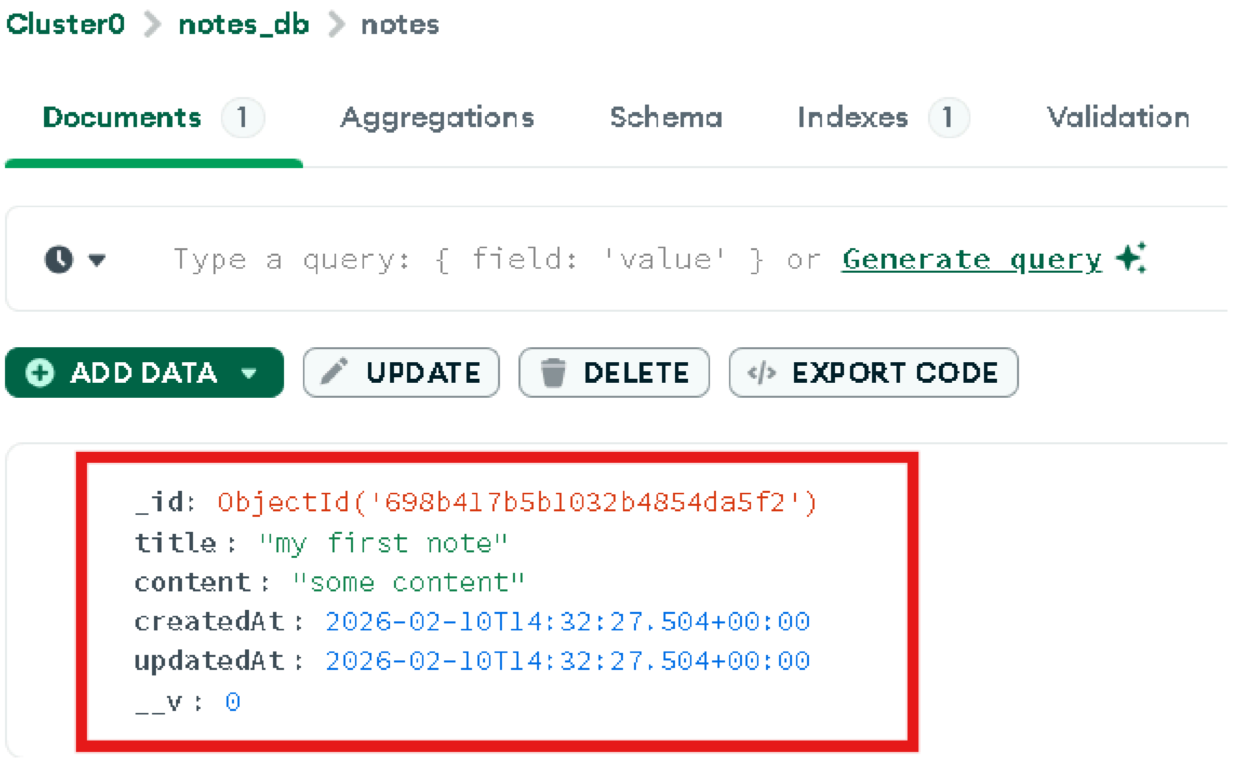




Wait to connect to Custer0

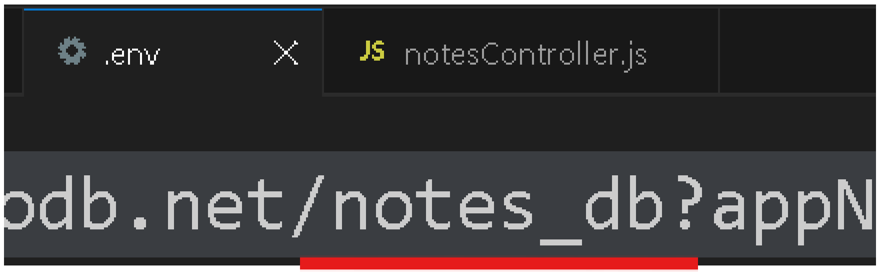


Then in the **notes collection** has:



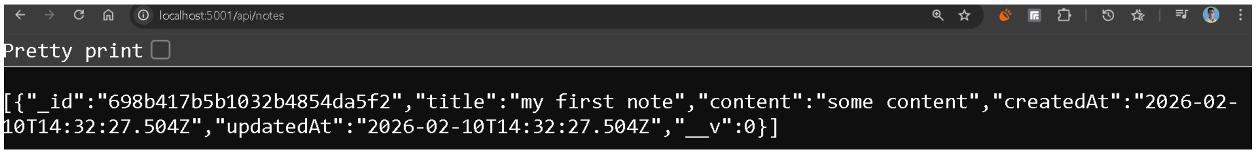
```
_id: ObjectId('698b417b5b1032b4854da5f2')
title : "my first note"
content : "some content"
createdAt : 2026-02-10T14:32:27.504+00:00
updatedAt : 2026-02-10T14:32:27.504+00:00
__v : 0
```

MongoDB has created **_id, createdAt and updatedAt.**
notes_db is actually comes from**.env** file, when setting up



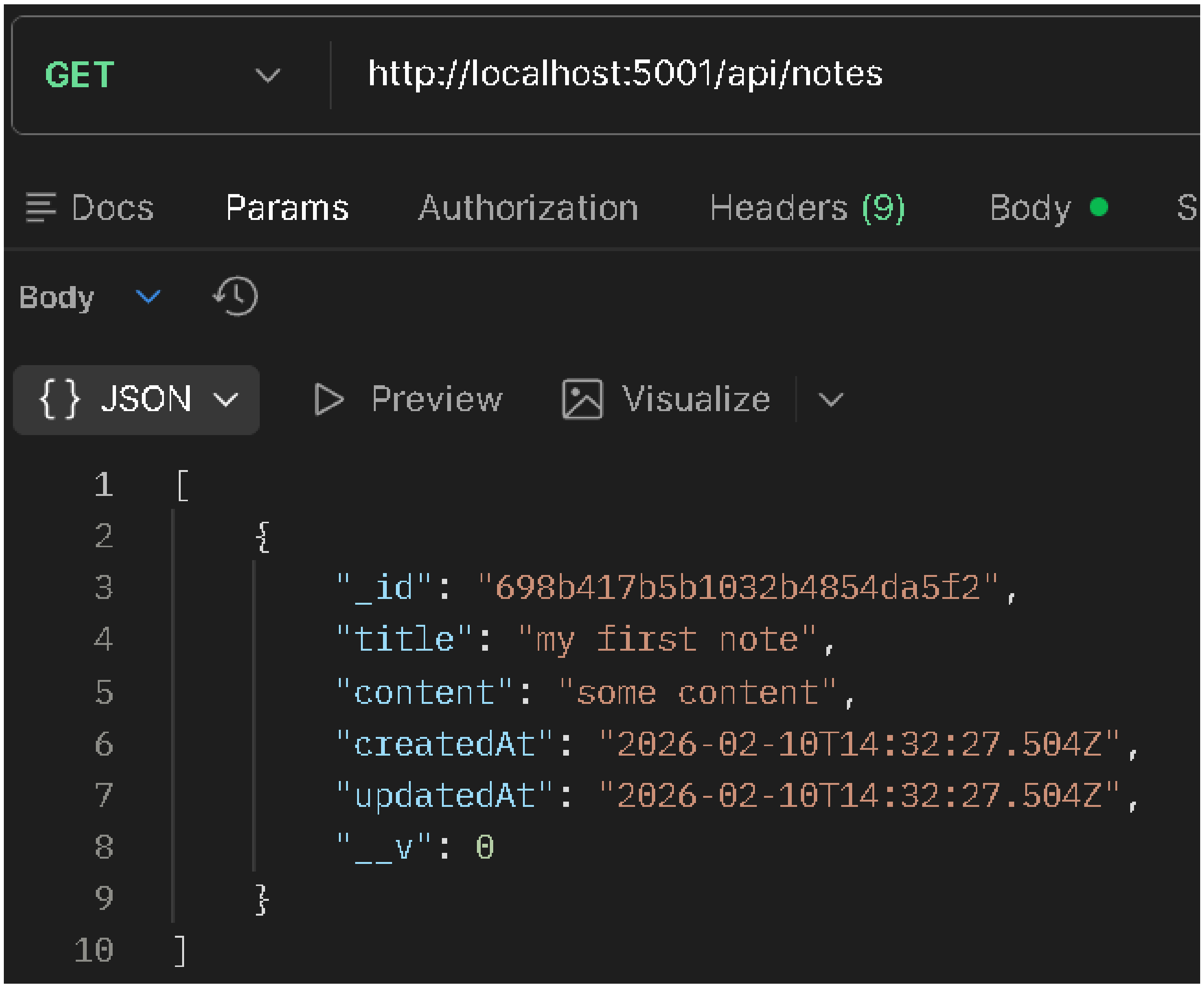
```
odb.net/notes_db?appN
```

If you have not provided, then database is named as **test.**
In the URL: http://localhost:5001/api/notes



```
[{"_id":"698b417b5b1032b4854da5f2","title":"my first note","content":"some content","createdAt":"2026-02-10T14:32:27.504Z","updatedAt":"2026-02-10T14:32:27.504Z","__v":0}]
```

Using postman, Get request:
Click send to get response



http://localhost:5001/api/notes

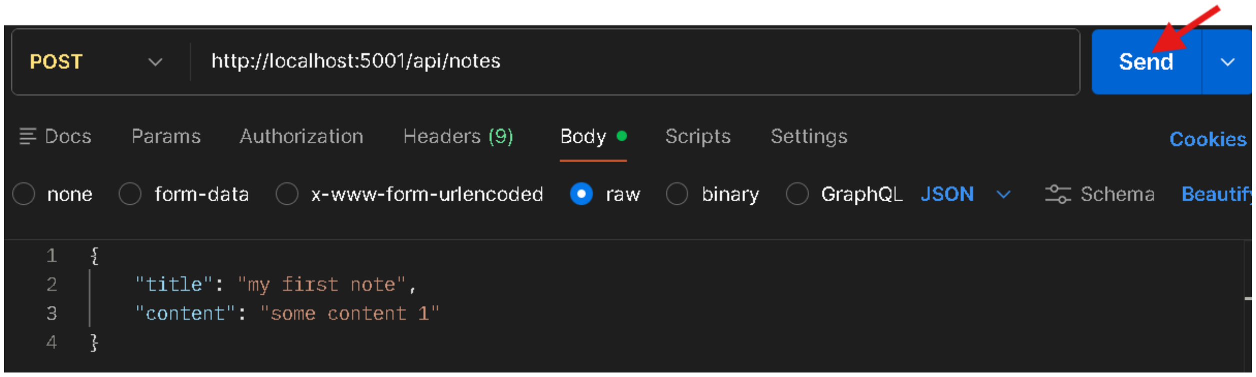
```json
[
    {
        "_id": "698b417b5b1032b4854da5f2",
        "title": "my first note",
        "content": "some content",
        "createdAt": "2026-02-10T14:32:27.504Z",
        "updatedAt": "2026-02-10T14:32:27.504Z",
        "__v": 0
    }
]
```

Code for createNote function:

```
export async function createNote(req, res) {
  // If user want to create a notes:
  // includes title, content
  try {
    // title and content comes from req.body
    const {title, content} = req.body
    // by default we can't access this value
    // To access them(or console it),
    // Go to server.js, just before the routes
    // add the code -> app.use(express.json())
    // which is a middleware that we add
    // console.log(title, content)

    // const newNote = new Note({title:title, content: content})
```

```
        // Since key and value are the same, so the above code can
        // be replaced as
        // const newNote = new Note({title, content})
        // await newNote.save()

        const newNote = new Note({title, content})
        await newNote.save()
        res.status(201).json({message: "Note Created successfully!"})

        //res.status(201).json({message: "Note created Successfully"})
    } catch (error) {
        console.error("Error in createNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}
```

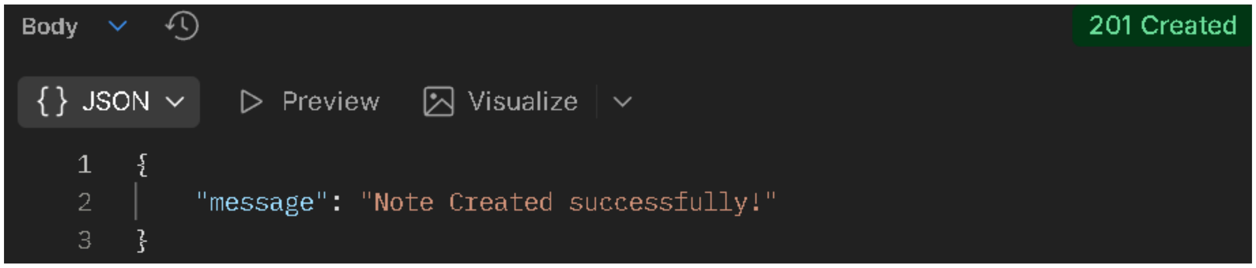
In postman:



POST http://localhost:5001/api/notes

```
{
    "title": "my first note",
    "content": "some content 1"
}
```

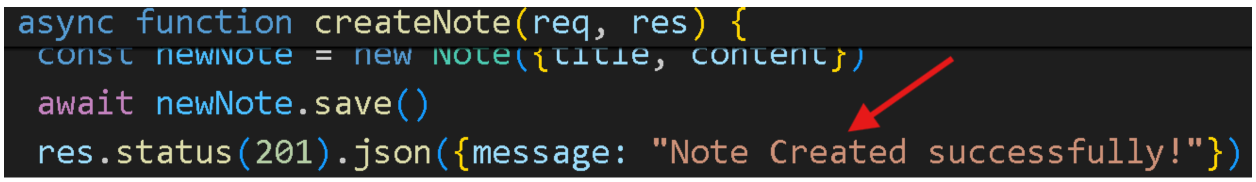
Response:



201 Created

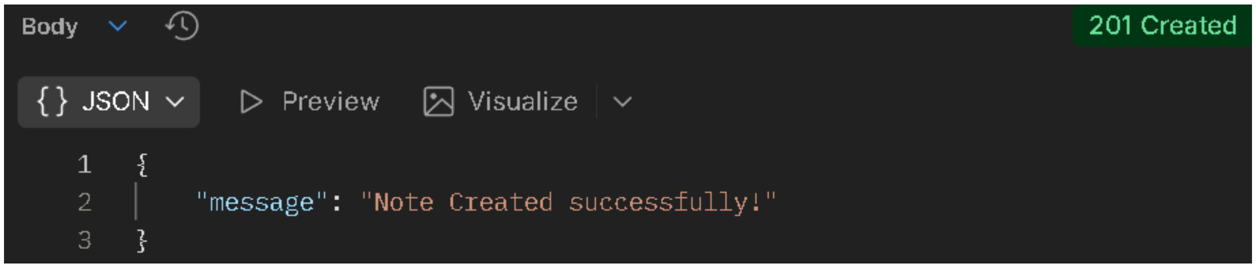
```
{
    "message": "Note Created successfully!"
}
```

As per the code:



```
async function createNote(req, res) {
  const newNote = new Note({title, content})
  await newNote.save()
  res.status(201).json({message: "Note Created successfully!"})
```

If we want informational response from postman instead of:



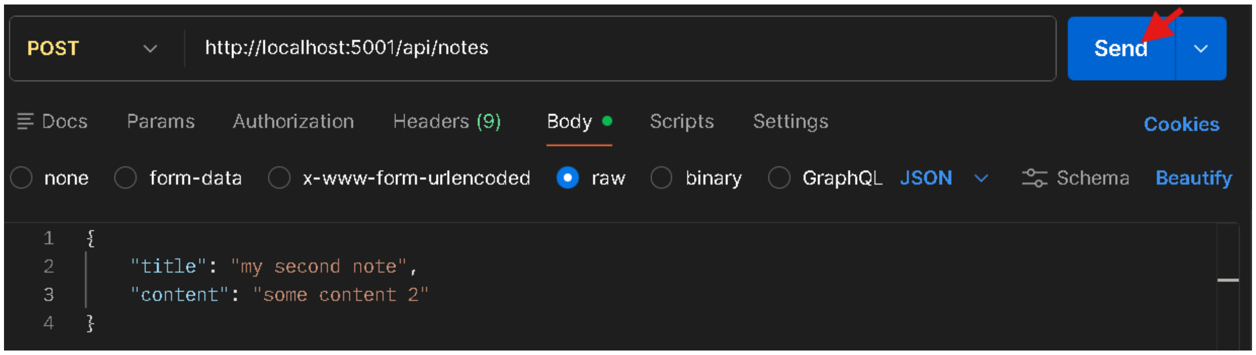
```
{
    "message": "Note Created successfully!"
}
```

Which is not providing more information.

Code for createNote function:

```
export async function createNote(req, res) {
    // If user want to create a notes:
    // includes title, content
    try {
        // title and content comes from req.body
        const {title, content} = req.body
        // by default we can't access this value
        // To access them(or console it),
        // Go to server.js, just before the routes
        // add the code -> app.use(express.json())
        // which is a middleware that we add
        // console.log(title, content)

        // const newNote = new Note({title:title, content: content})
        // Since key and value are the same, so the above code can
        // be replaced as
        // const newNote = new Note({title, content})
        // await newNote.save()

        // const newNote = new Note({title, content})
        // await newNote.save()
        // res.status(201).json({message: "Note Created successfully!"})

        const note = new Note({title, content})
        const savedNote = await note.save()
        res.status(201).json(savedNote)

    } catch (error) {
        console.error("Error in createNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}
```

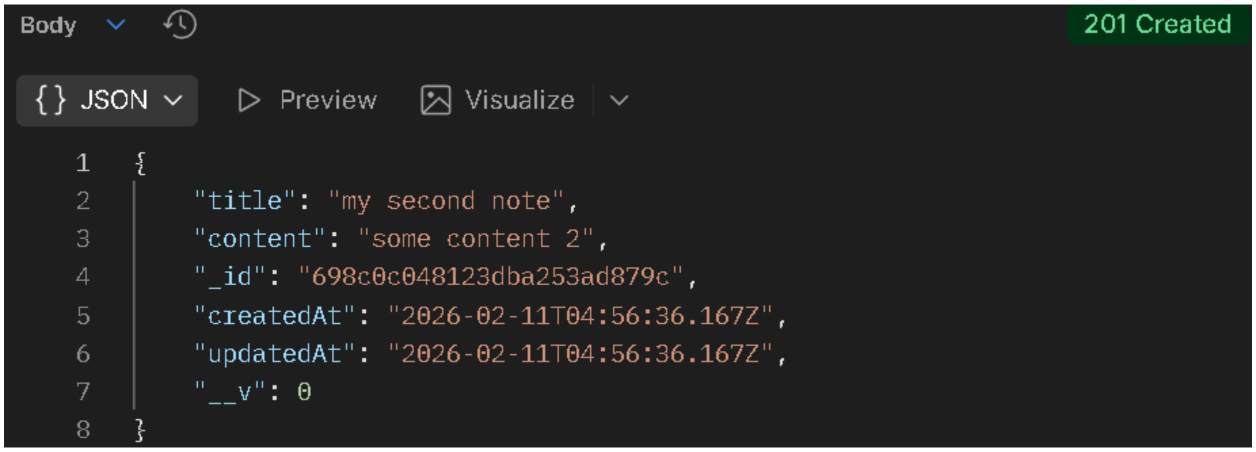
In postman:



```
http://localhost:5001/api/notes
```

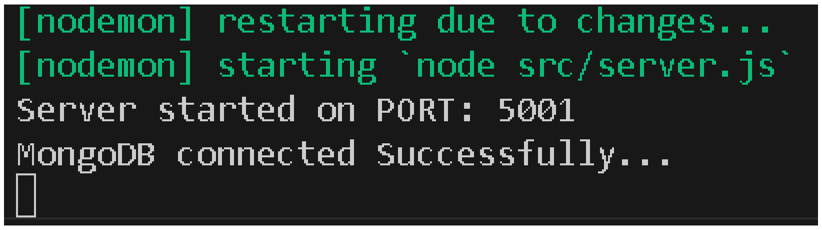
```
{
    "title": "my second note",
    "content": "some content 2"
}
```

Response:



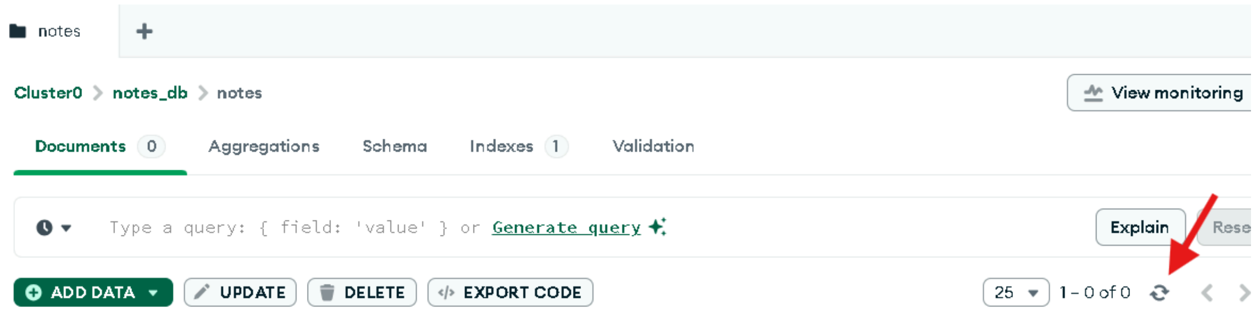
```
{
    "title": "my second note",
    "content": "some content 2",
    "_id": "698c0c048123dba253ad879c",
    "createdAt": "2026-02-11T04:56:36.167Z",
    "updatedAt": "2026-02-11T04:56:36.167Z",
    "__v": 0
}
```

In terminal:

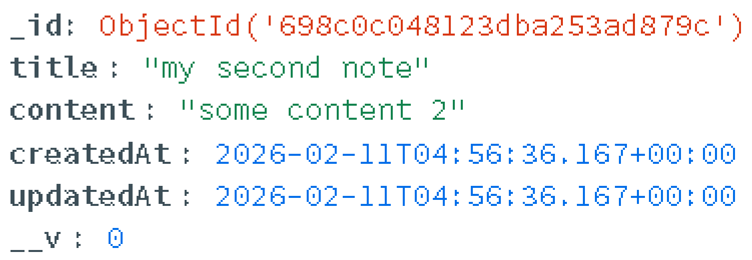


```
[nodemon] restarting due to changes...
[nodemon] starting `node src/server.js`
Server started on PORT: 5001
MongoDB connected Successfully...
```

In MongoDB(refresh it):



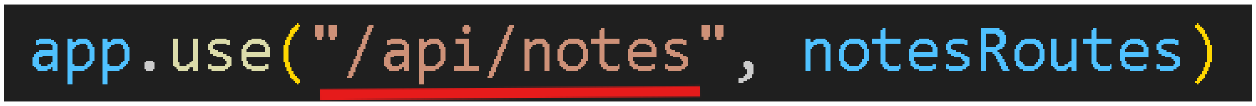
Then:



```
_id: ObjectId('698c0c048123dba253ad879c')
title : "my second note"
content : "some content 2"
createdAt : 2026-02-11T04:56:36.167+00:00
updatedAt : 2026-02-11T04:56:36.167+00:00
__v : 0
```

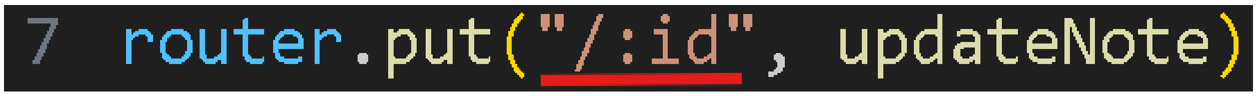
Concept of Updating:

In server.js:



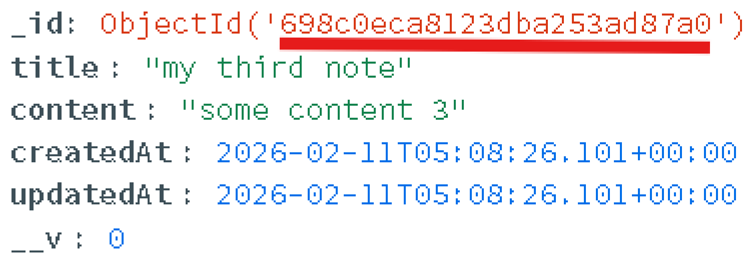
```
app.use("/api/notes", notesRoutes)
```

In notesRoutes.js:



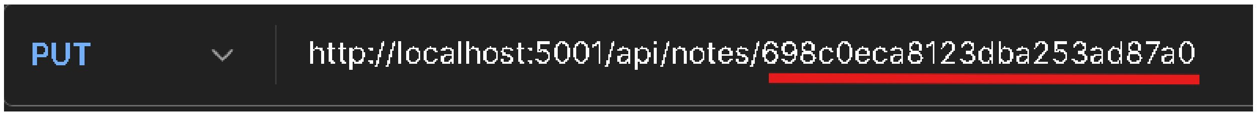
```
router.put("/:id", updateNote)
```

Example for updating, Content:



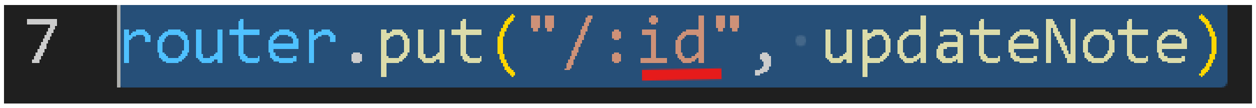
```
_id: ObjectId('698c0eca8123dba253ad87a0')
title : "my third note"
content : "some content 3"
createdAt : 2026-02-11T05:08:26.101+00:00
updatedAt : 2026-02-11T05:08:26.101+00:00
__v : 0
```

In postman, it must be like



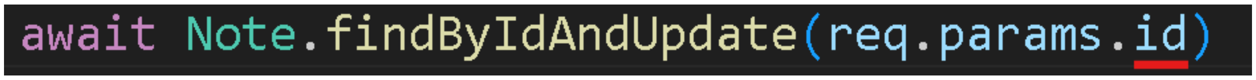
```
PUT    http://localhost:5001/api/notes/698c0eca8123dba253ad87a0
```

Code in notesRoutes.js:



```
router.put("/:id", updateNote)
```

and in notesController.js update function:



```
await Note.findByIdAndUpdate(req.params.id)
```

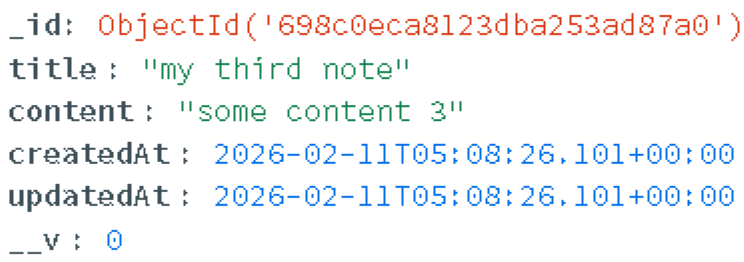
Must have same word.

Code for updateNote Function:

export async function updateNote(req, res) {

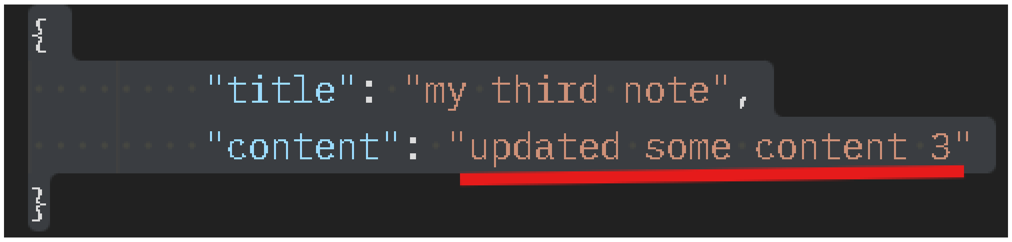
```
    try {
        const {title, content} = req.body
        // How do we know id that user sends, so that we can update
        // based on id
        // {title, content} <- things to update
        await Note.findByIdAndUpdate(req.params.id, {title, content})
        res.status(200).json({message: "Note updated successfully."})

    } catch (error) {
        console.error("Error in updateNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}
```

Initially in MongoDB:

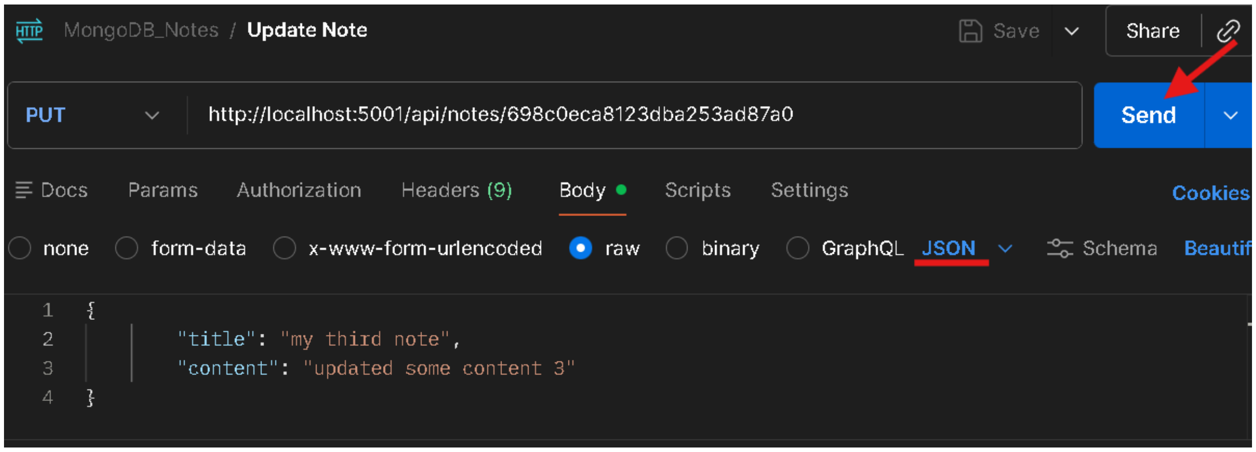


```
_id: ObjectId('698c0eca8123dba253ad87a0')
title : "my third note"
content : "some content 3"
createdAt : 2026-02-11T05:08:26.101+00:00
updatedAt : 2026-02-11T05:08:26.101+00:00
__v : 0
```

In Postman:



```
{
    "title": "my third note",
    "content": "updated some content 3"
}
```

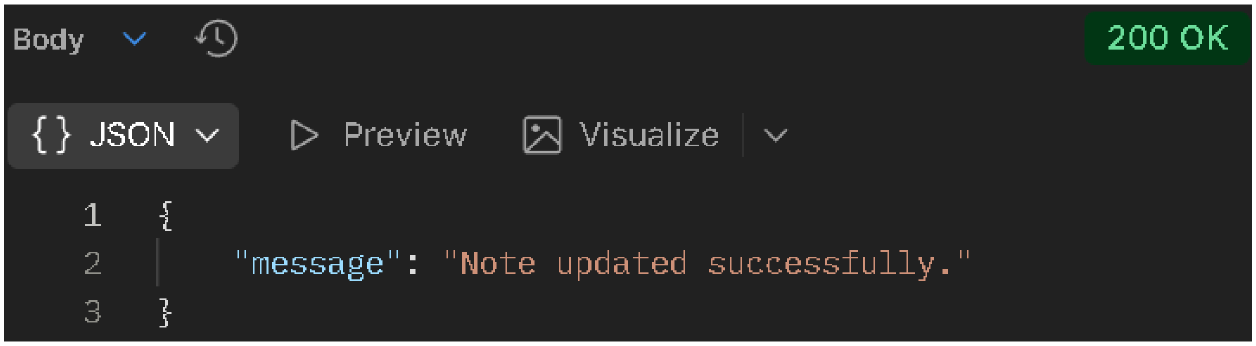


PUT http://localhost:5001/api/notes/698c0eca8123dba253ad87a0

Body: [✓ raw] JSON

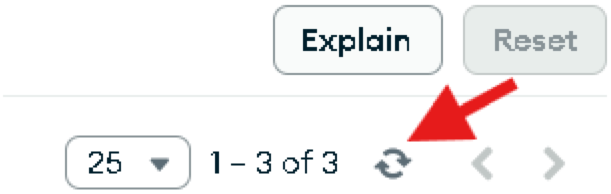
```
{
    "title": "my third note",
    "content": "updated some content 3"
}
```

Response:

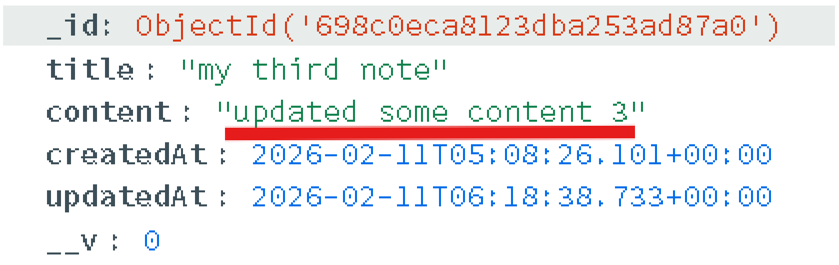


```
{
    "message": "Note updated successfully."
}
```

Refresh DB:



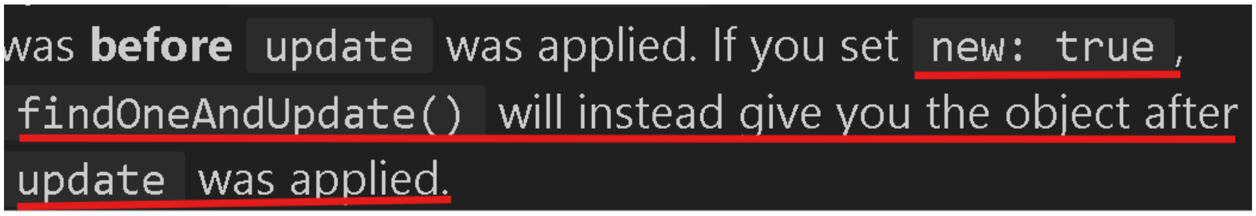
In MongoDB:



```
_id: ObjectId('698c0eca8123dba253ad87a0')
title: "my third note"
content: "updated some content 3"
createdAt: 2026-02-11T05:08:26.101+00:00
updatedAt: 2026-02-11T06:18:38.733+00:00
__v: 0
```

If user provides invalid ID, during Updating:

To get fields after update.

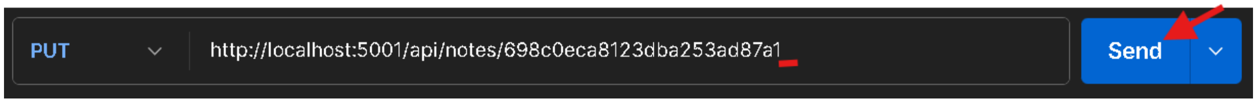


was **before** `update` was applied. If you set `new: true`, `findOneAndUpdate()` will instead give you the object after `update` was applied.

Code for updateNote Function:

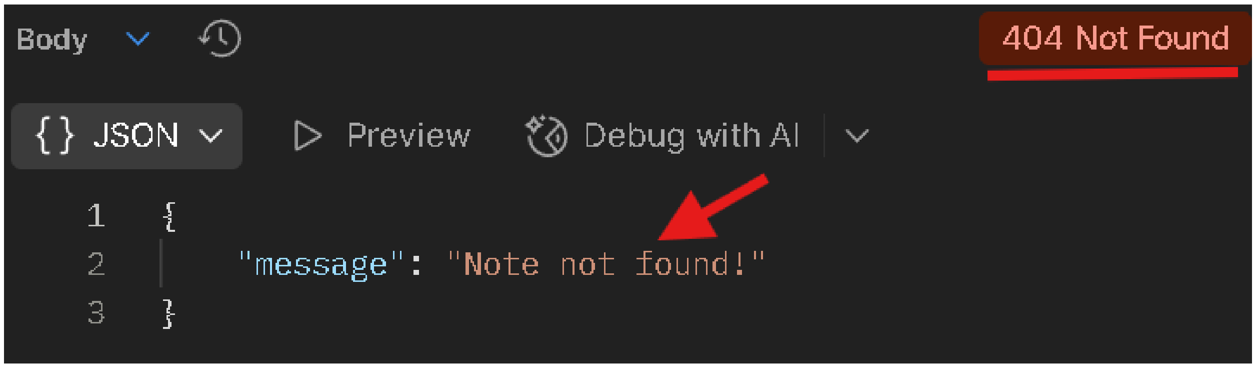
```
export async function updateNote(req, res) {
    try {
        const {title, content} = req.body
        // How do we know id that user sends, so that we can update
        // based on id
        // {title, content} <- things to update
        // {new: true} <- to get fields
        const updatedNote = await Note.findByIdAndUpdate(req.params.id, {title, content}, {new:
true})
        // For false value, 404=not found,
        if(!updatedNote) return res.status(404).json({message: "Note not found!"})
        res.status(200).json({message: "Note updated successfully."})

    } catch (error) {
        console.error("Error in updateNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}
```

Now in postman:



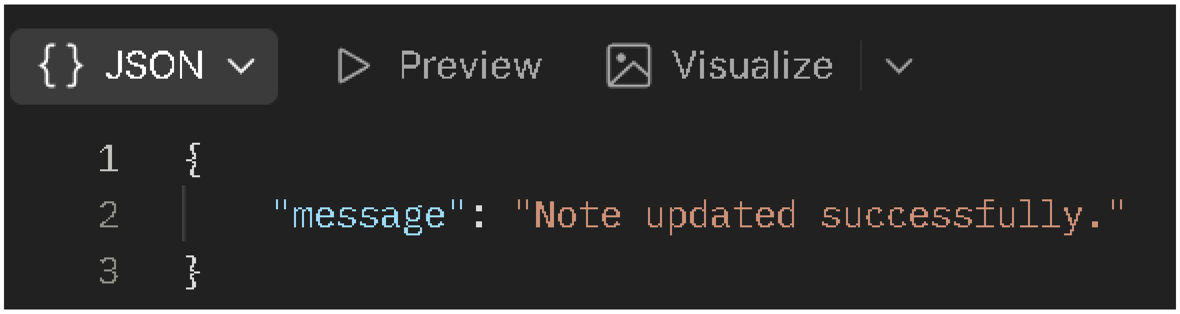
```
PUT    http://localhost:5001/api/notes/698c0eca8123dba253ad87a1
```

Response:



```
{
    "message": "Note not found!"
}
```

For successful update:



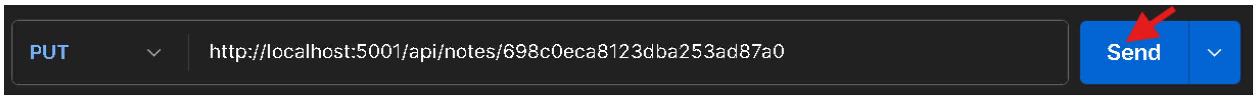
```
{
    "message": "Note updated successfully."
}
```

To change the response:

Code for updateNote Function:

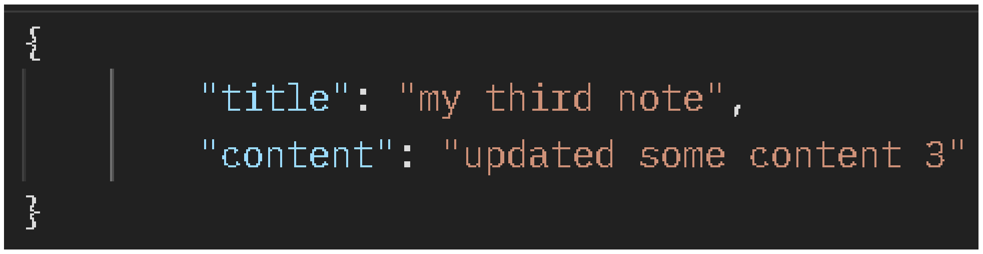
```
export async function updateNote(req, res) {
    try {
        const {title, content} = req.body
        // How do we know id that user sends, so that we can update
        // based on id
        // {title, content} <- things to update
        // {new: true} <- to get fields
        const updatedNote = await Note.findByIdAndUpdate(req.params.id, {title, content}, {new:
true})
        // For false value, 404=not found,
        if(!updatedNote) return res.status(404).json({message: "Note not found!"})
        // res.status(200).json({message: "Note updated successfully."})
        res.status(200).json(updatedNote)

    } catch (error) {
        console.error("Error in updateNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}
```

In postman:



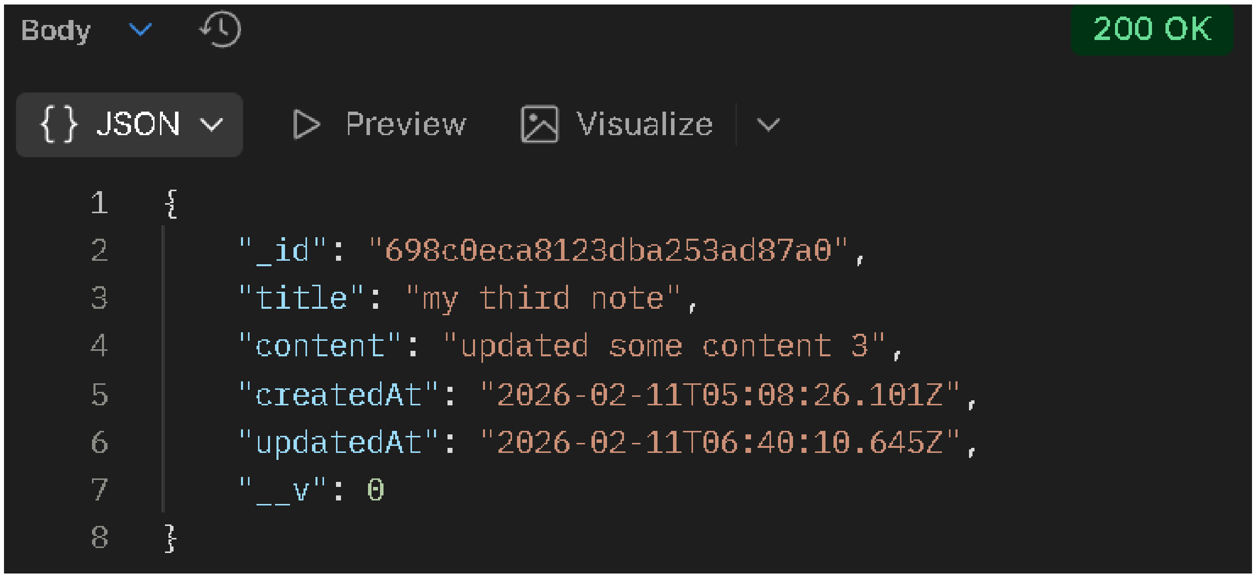
```
PUT  http://localhost:5001/api/notes/698c0eca8123dba253ad87a0
```

Body, json, raw having:



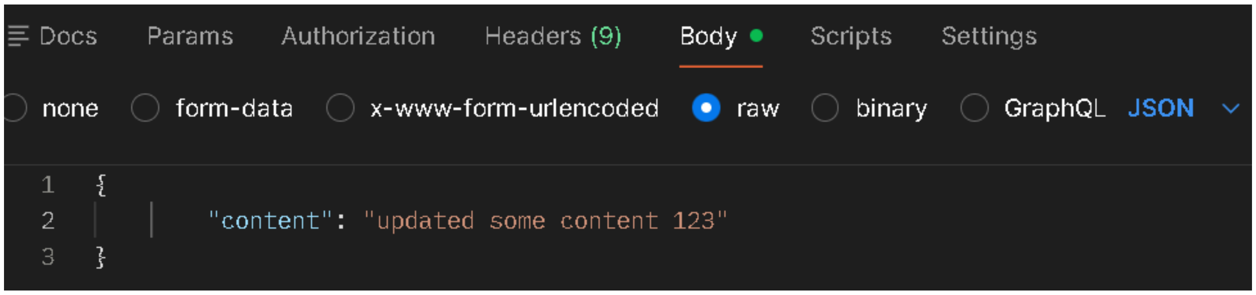
```
{
        "title": "my third note",
        "content": "updated some content 3"
}
```

Response:



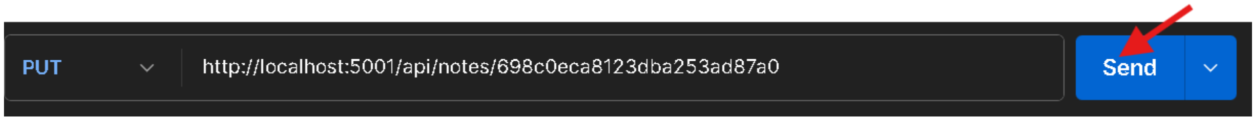
```
{
    "_id": "698c0eca8123dba253ad87a0",
    "title": "my third note",
    "content": "updated some content 3",
    "createdAt": "2026-02-11T05:08:26.101Z",
    "updatedAt": "2026-02-11T06:40:10.645Z",
    "__v": 0
}
```

What if title is not provided(JSON Type):



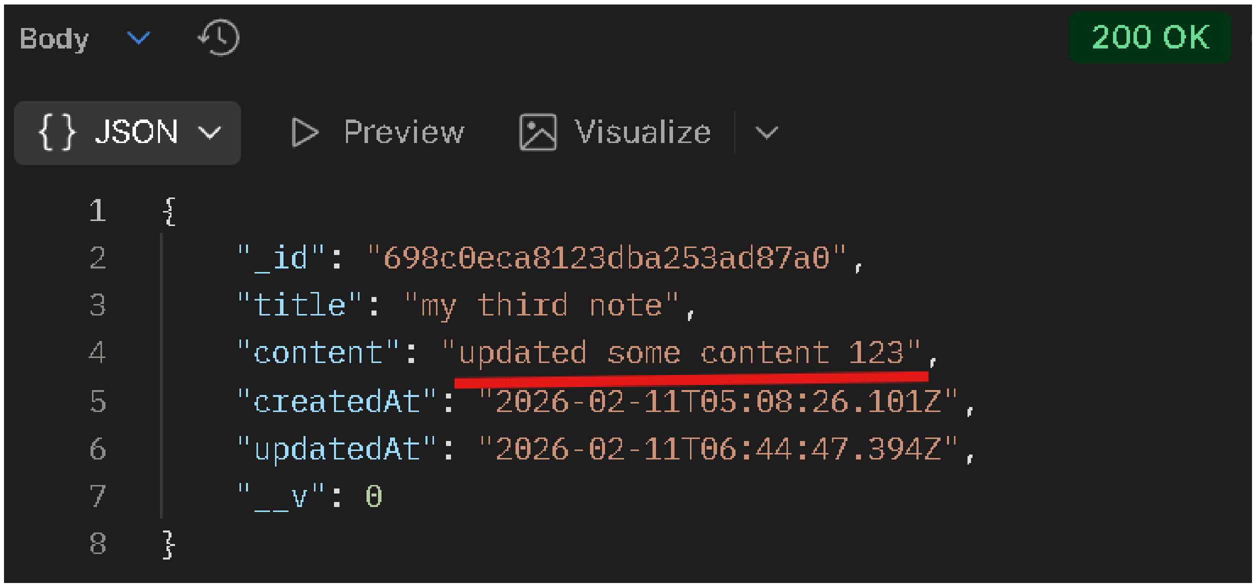
```
{
    "content": "updated some content 123"
}
```

Send it:



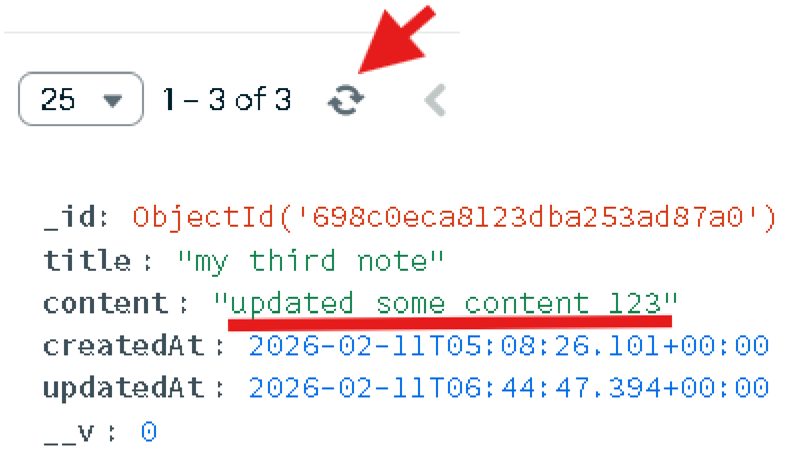
PUT http://localhost:5001/api/notes/698c0eca8123dba253ad87a0

Response:



```
{
    "_id": "698c0eca8123dba253ad87a0",
    "title": "my third note",
    "content": "updated some content 123",
    "createdAt": "2026-02-11T05:08:26.101Z",
    "updatedAt": "2026-02-11T06:44:47.394Z",
    "__v": 0
}
```

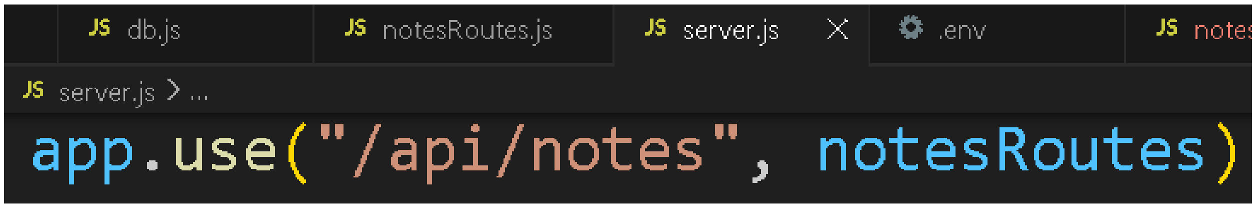
Refresh in MongoDB:



```
_id: ObjectId('698c0eca8123dba253ad87a0')
title : "my third note"
content : "updated some content 123"
createdAt : 2026-02-11T05:08:26.101+00:00
updatedAt : 2026-02-11T06:44:47.394+00:00
__v : 0
```

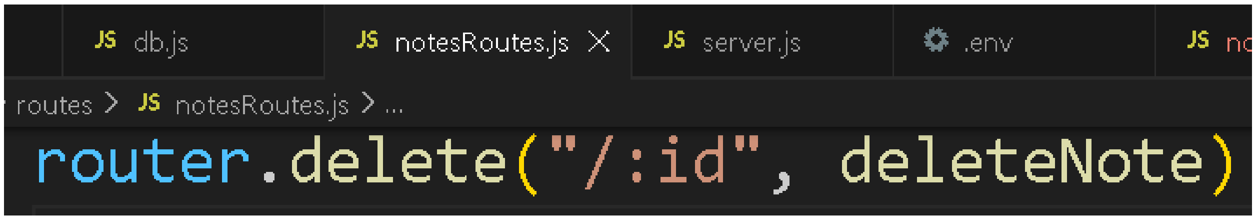
For Deleting the Notes:

In server.js:



```
app.use("/api/notes", notesRoutes)
```

In notesRoutes.js:



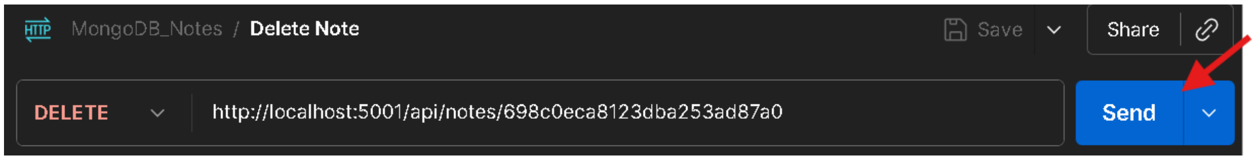
```
router.delete("/:id", deleteNote)
```

Code in deleteNote:

```
export async function deleteNote(req, res) {
    try {
        const deletedNote = await Note.findByIdAndDelete(req.params.id)
        if (!deleteNote) return res.status(404).json({message: "Note not found!"})
        // By default status=200
        res.status(200).json({message: "Note deleted successfully!"})
    } catch (error) {
        console.error("Error in deleteNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}
```

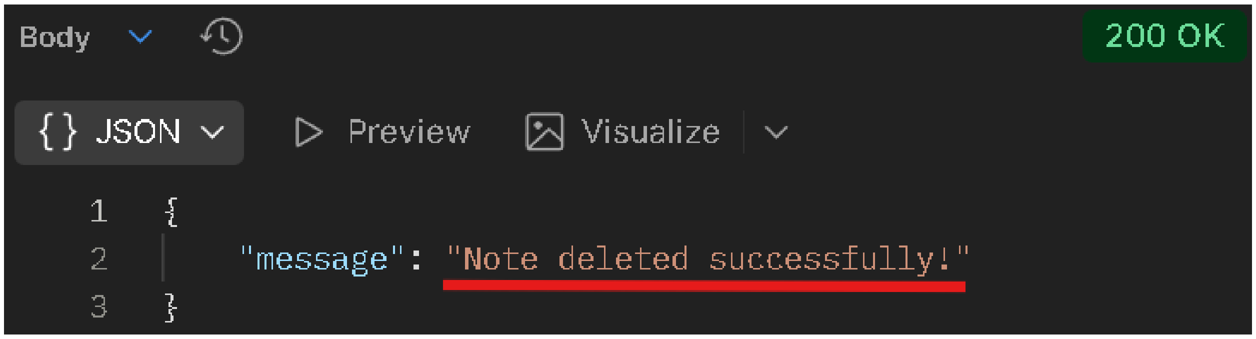
In postman:

No need to provide body



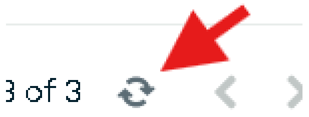
DELETE http://localhost:5001/api/notes/698c0eca8123dba253ad87a0

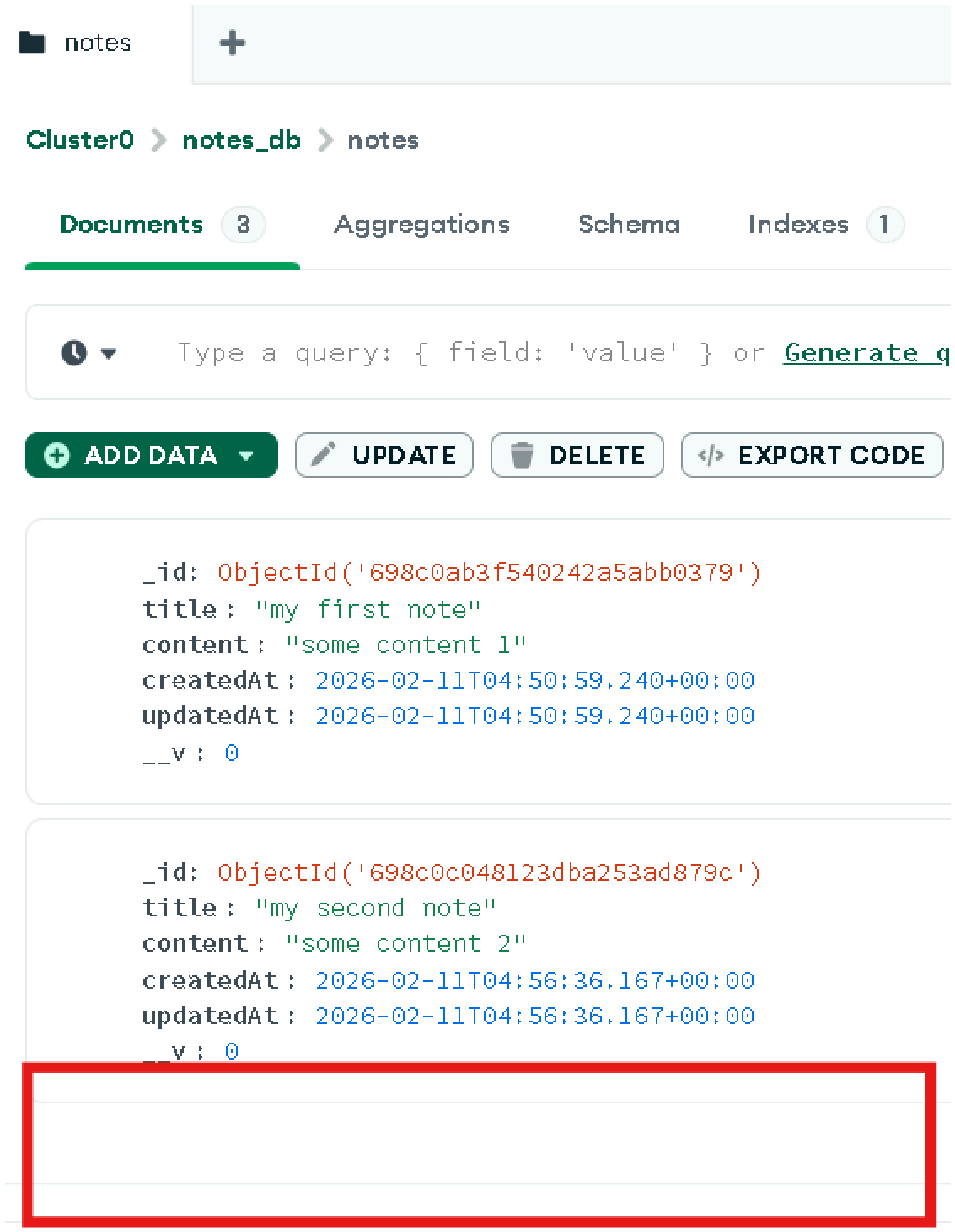
Response:



```
{
    "message": "Note deleted successfully!"
}
```

In MongoDB:
Refresh:

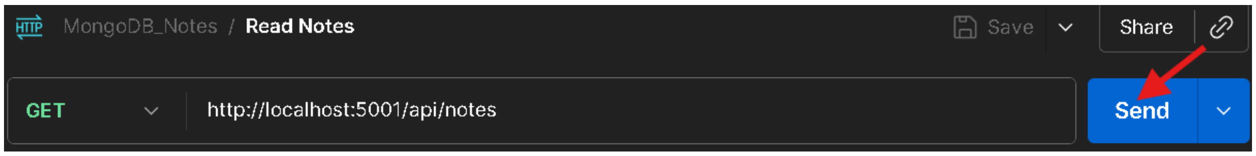




```
_id: ObjectId('698c0ab3f540242a5abb0379')
title: "my first note"
content: "some content 1"
createdAt: 2026-02-11T04:50:59.240+00:00
updatedAt: 2026-02-11T04:50:59.240+00:00
__v: 0

_id: ObjectId('698c0c048123dba253ad879c')
title: "my second note"
content: "some content 2"
createdAt: 2026-02-11T04:56:36.167+00:00
updatedAt: 2026-02-11T04:56:36.167+00:00
__v: 0
```

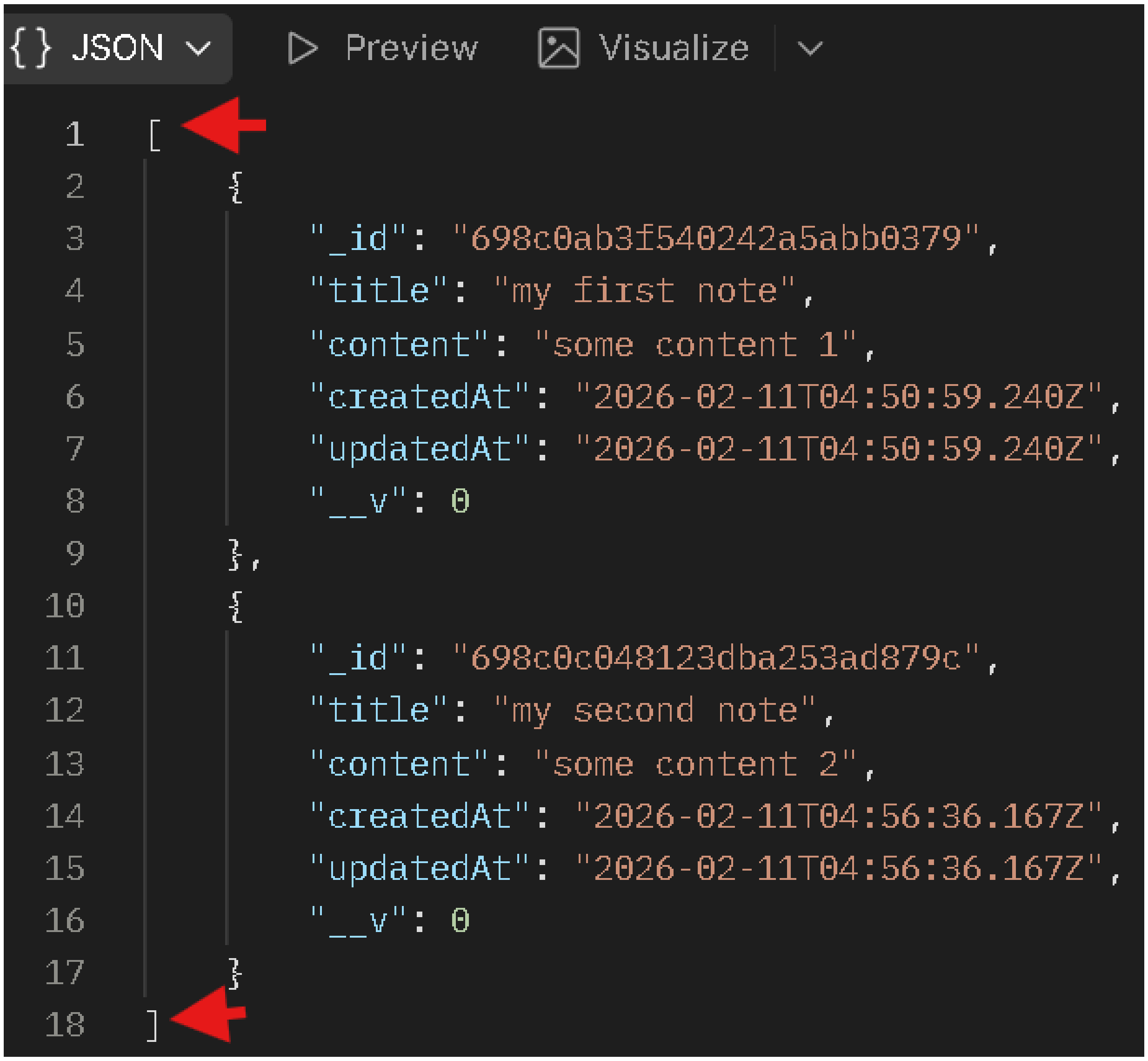
Third not visible.

In postman, get request:



GET http://localhost:5001/api/notes

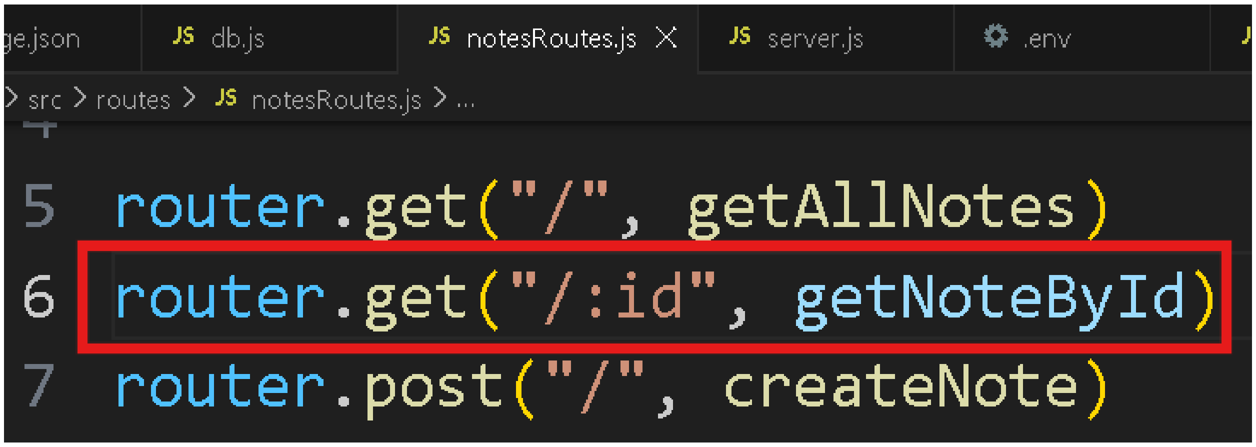
Response:



```
[
    {
        "_id": "698c0ab3f540242a5abb0379",
        "title": "my first note",
        "content": "some content 1",
        "createdAt": "2026-02-11T04:50:59.240Z",
        "updatedAt": "2026-02-11T04:50:59.240Z",
        "__v": 0
    },
    {
        "_id": "698c0c048123dba253ad879c",
        "title": "my second note",
        "content": "some content 2",
        "createdAt": "2026-02-11T04:56:36.167Z",
        "updatedAt": "2026-02-11T04:56:36.167Z",
        "__v": 0
    }
]
```

To fetch the user based on ID:

In notesRoutes.js:



```
router.get("/", getAllNotes)
router.get("/:id", getNoteById)
router.post("/", createNote)
```

Also import it:
So finally notesRouters.js has:
import express from "express"
import { getAllNotes, **getNoteById**, createNote, updateNote, deleteNote } from "../controllers/notesController.js"
const router = express.Router()

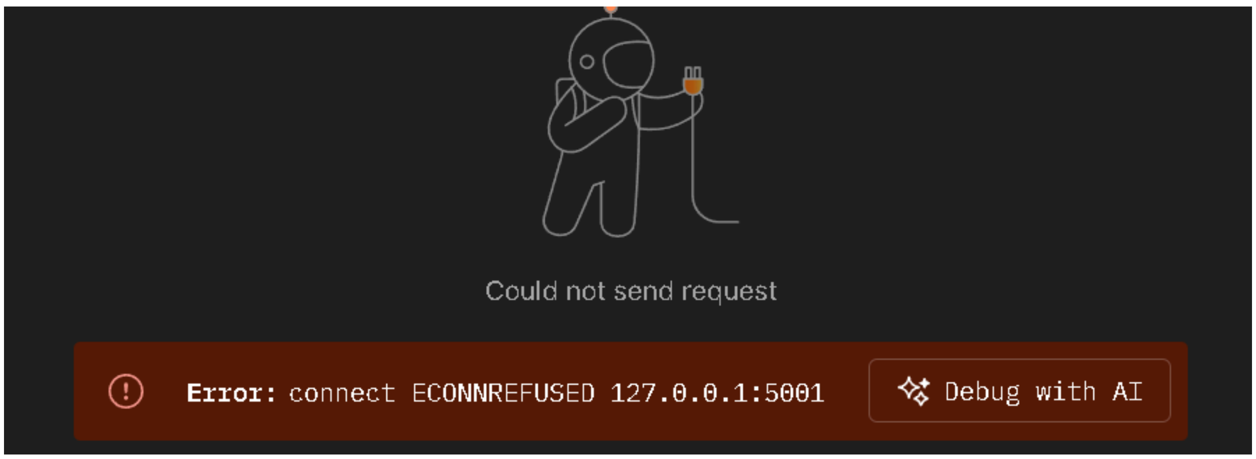
router.get("/", getAllNotes)
router.get("/:id", getNoteById)
router.post("/", createNote)
router.put("/:id", updateNote)
router.delete("/:id", deleteNote)

export default router;
Now let's create its controller.

Code for getNoteById function:

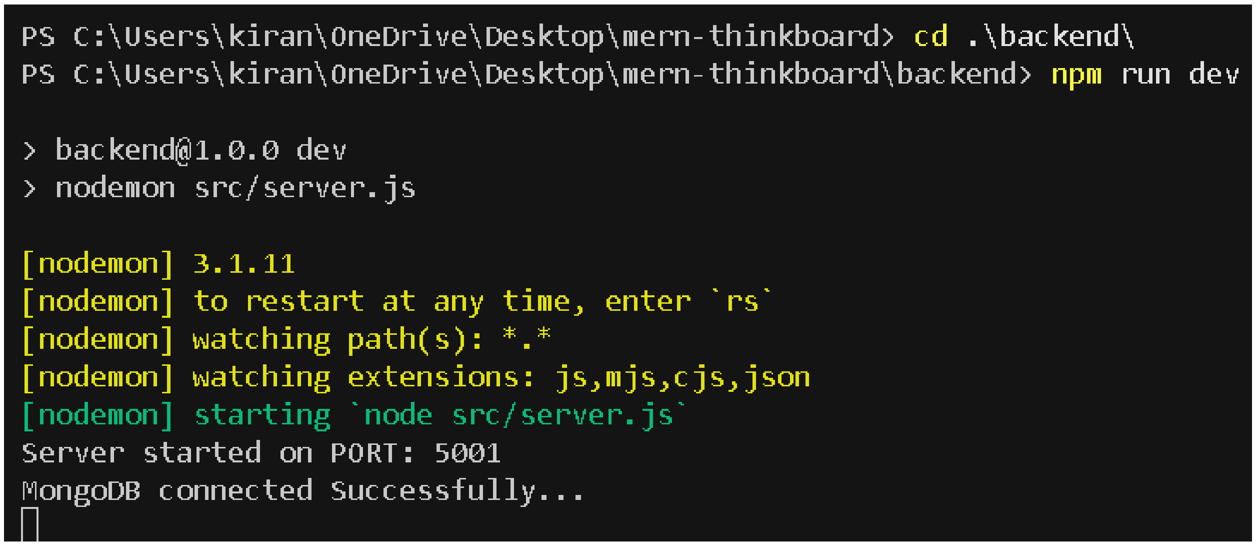
```
export async function getNoteById(req, res) {
    try {
        const note = await Note.findById(req.params.id)
        if(!note) return res.status(404).json({message: "Note not Found!"})
        res.json(note)
    } catch (error) {
        console.error("Error in getNoteById controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}
```

If in postman, if you get error:



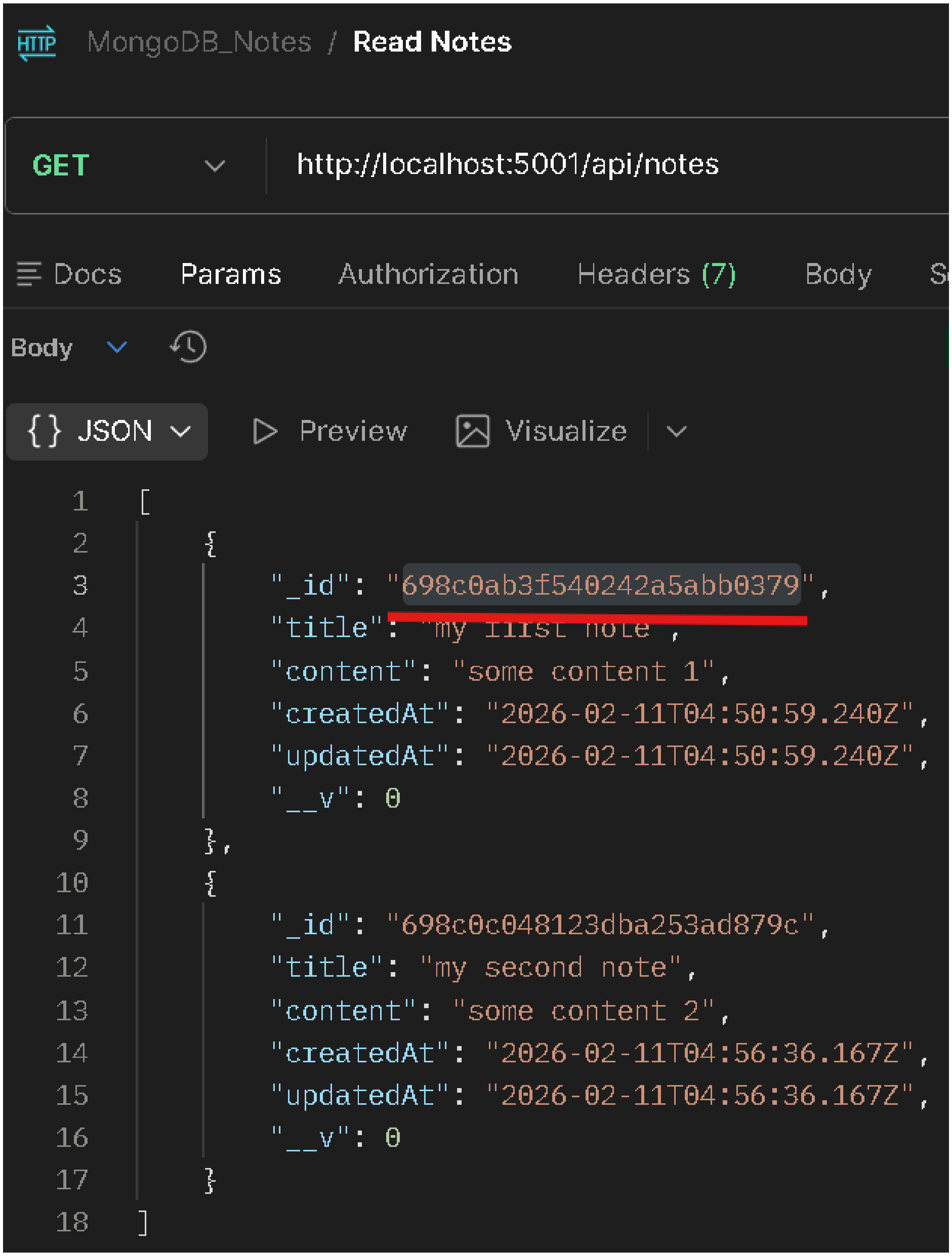
```
Error: connect ECONNREFUSED 127.0.0.1:5001
```

Solution is: run the server



```
PS C:\Users\kiran\OneDrive\Desktop\mern-thinkboard> cd .\backend\
PS C:\Users\kiran\OneDrive\Desktop\mern-thinkboard\backend> npm run dev

> backend@1.0.0 dev
> nodemon src/server.js

[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/server.js`
Server started on PORT: 5001
MongoDB connected Successfully...
```

Copy the id, from **get Request from** notes.

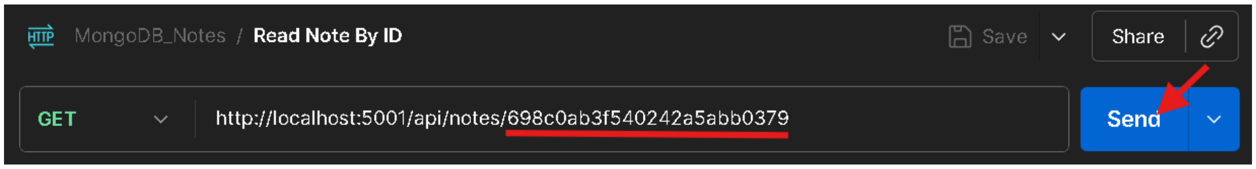


MongoDB_Notes / Read Notes

GET http://localhost:5001/api/notes

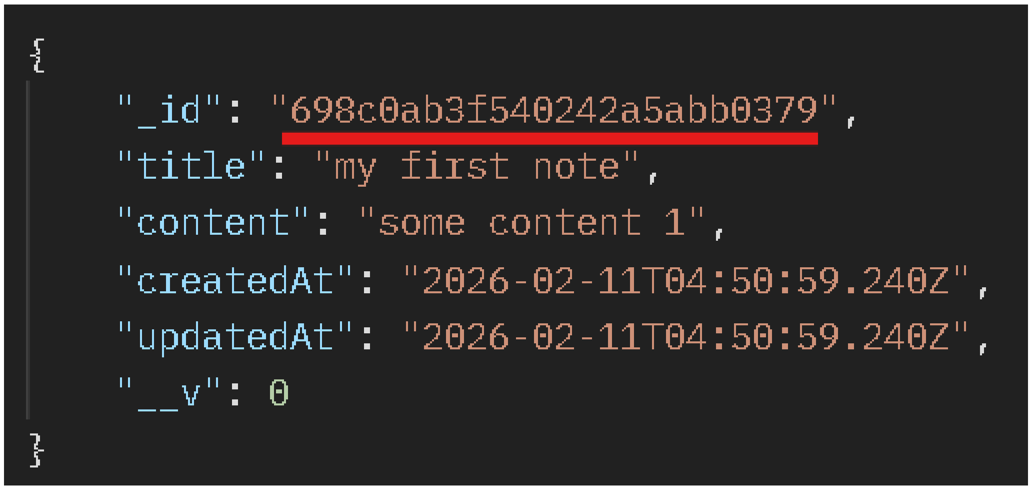
```
[
    {
        "_id": "698c0ab3f540242a5abb0379",
        "title": "my first note",
        "content": "some content 1",
        "createdAt": "2026-02-11T04:50:59.240Z",
        "updatedAt": "2026-02-11T04:50:59.240Z",
        "__v": 0
    },
    {
        "_id": "698c0c048123dba253ad879c",
        "title": "my second note",
        "content": "some content 2",
        "createdAt": "2026-02-11T04:56:36.167Z",
        "updatedAt": "2026-02-11T04:56:36.167Z",
        "__v": 0
    }
]
```

Paste to get note by ID:



```
GET http://localhost:5001/api/notes/698c0ab3f540242a5abb0379
```

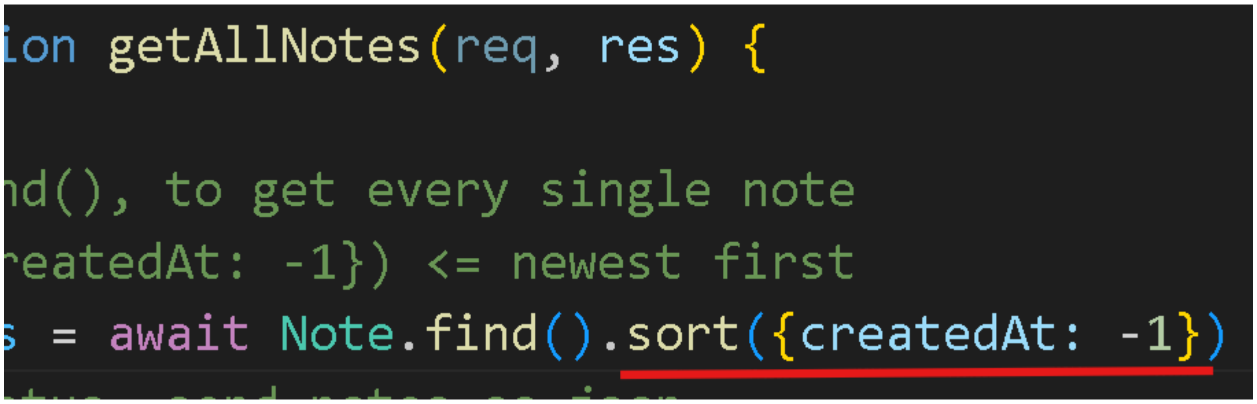
Response:



```
{
    "_id": "698c0ab3f540242a5abb0379",
    "title": "my first note",
    "content": "some content 1",
    "createdAt": "2026-02-11T04:50:59.240Z",
    "updatedAt": "2026-02-11T04:50:59.240Z",
    "__v": 0
}
```

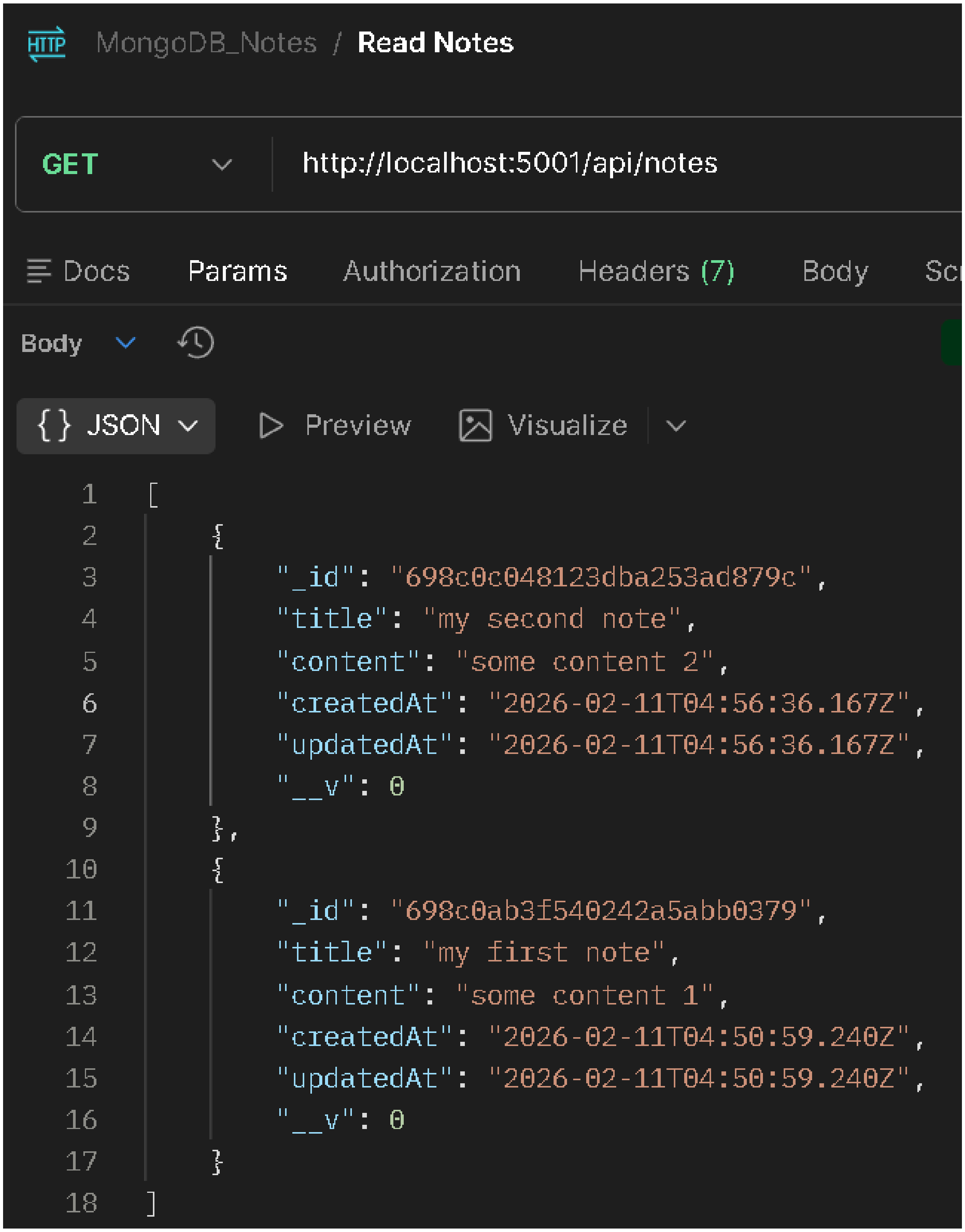
To sort all the notes:

Like newest first.



```
ion getAllNotes(req, res) {

nd(), to get every single note
reatedAt: -1}) <= newest first
s = await Note.find().sort({createdAt: -1})
```

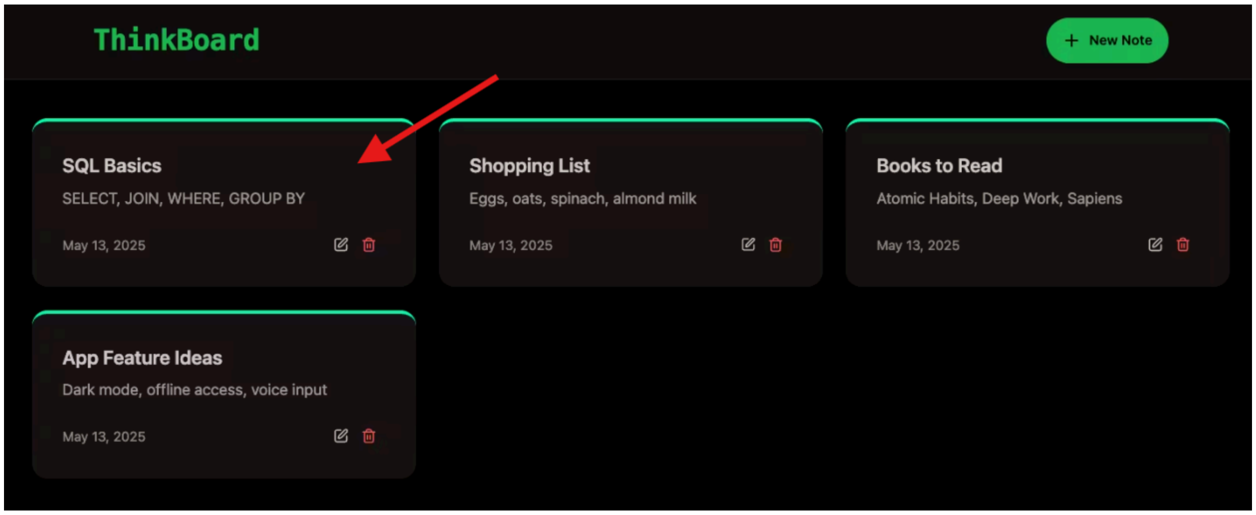
Then in postman:



```
GET http://localhost:5001/api/notes
```

```
[
    {
        "_id": "698c0c048123dba253ad879c",
        "title": "my second note",
        "content": "some content 2",
        "createdAt": "2026-02-11T04:56:36.167Z",
        "updatedAt": "2026-02-11T04:56:36.167Z",
        "__v": 0
    },
    {
        "_id": "698c0ab3f540242a5abb0379",
        "title": "my first note",
        "content": "some content 1",
        "createdAt": "2026-02-11T04:50:59.240Z",
        "updatedAt": "2026-02-11T04:50:59.240Z",
        "__v": 0
    }
]
```

So **my second note(title)** has come to the first place.
So that I can get the latest note at the beginning:


ThinkBoard
New Note
SQL Basics
SELECT, JOIN, WHERE, GROUP BY
May 13, 2025
Shopping List
Eggs, oats, spinach, almond milk
May 13, 2025
Books to Read
Atomic Habits, Deep Work, Sapiens
May 13, 2025
App Feature Ideas
Dark mode, offline access, voice input
May 13, 2025

Code in server.js:

```
import express from "express"
import notesRoutes from "./routes/notesRoutes.js"
import { connectDB } from "./config/db.js"

import dotenv from "dotenv"
dotenv.config()

//console.log(process.env.MONGO_URI)

const app = express()
// if process.env.PORT is undefined then PORT = 5001(by default value)
const PORT = process.env.PORT || 5001

connectDB()

// middleware: are used to control title and content
// instead of providing it from postman
app.use(express.json())

app.use("/api/notes", notesRoutes)


app.listen(PORT, () => {
   console.log("Server started on PORT:", PORT)
})
```

Code in notesRoutes.js:

```
import express from "express"
import { getAllNotes, getNoteById, createNote, updateNote, deleteNote } from
"../controllers/notesController.js"
const router = express.Router()

router.get("/", getAllNotes)
router.get("/:id", getNoteById)
router.post("/", createNote)
router.put("/:id", updateNote)
router.delete("/:id", deleteNote)

export default router;
```

Code in notesController.js:

```
import Note from "../models/Note.js"

export async function getAllNotes(req, res) {
  try {
    // Note.find(), to get every single note
    // sort({createdAt: -1}) <= newest first
    // by default createdAt: 1
    const notes = await Note.find().sort({createdAt: -1})
    // Send status, send notes as json
    res.status(200).json(notes)
  } catch (error) {
    // For debugging purpose
    console.error("Error in getAllNotes controller", error)
    res.status(500).json({message: "Internal Server Error"})
  }
}

export async function getNoteById(req, res) {
  try {
    const note = await Note.findById(req.params.id)
    if(!note) return res.status(404).json({message: "Note not Found!"})
    res.json(note)
  } catch (error) {
    console.error("Error in getNoteById controller", error)
    res.status(500).json({message: "Internal Server Error"})
  }
}
```

```
export async function createNote(req, res) {
    // If user want to create a notes:
    // includes title, content
    try {
        // title and content comes from req.body
        const {title, content} = req.body
        // by default we can't access this value
        // To access them(or console it),
        // Go to server.js, just before the routes
        // add the code -> app.use(express.json())
        // which is a middleware that we add
        // console.log(title, content)

        // const newNote = new Note({title:title, content: content})
        // Since key and value are the same, so the above code can
        // be replaced as
        // const newNote = new Note({title, content})
        // await newNote.save()

        // const newNote = new Note({title, content})
        // await newNote.save()
        // res.status(201).json({message: "Note Created successfully!"})

        const note = new Note({title, content})
        const savedNote = await note.save()
        res.status(201).json(savedNote)

    } catch (error) {
        console.error("Error in createNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}

export async function updateNote(req, res) {
    try {
        const {title, content} = req.body
        // How do we know id that user sends, so that we can update
        // based on id
        // {title, content} <- things to update
        // {new: true} <- to get fields
        const updatedNote = await Note.findByIdAndUpdate(req.params.id, {title, content}, {new:
true})
        // For false value, 404=not found,
        if(!updatedNote) return res.status(404).json({message: "Note not found!"})
        // res.status(200).json({message: "Note updated successfully."})
        res.status(200).json(updatedNote)

    } catch (error) {
        console.error("Error in updateNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}

export async function deleteNote(req, res) {
    try {
        const deletedNote = await Note.findByIdAndDelete(req.params.id)
        if (!deleteNote) return res.status(404).json({message: "Note not found!"})
        res.status(200).json({message: "Note deleted successfully!"})
    } catch (error) {
        console.error("Error in deleteNote controller", error)
        res.status(500).json({message: "Internal Server Error"})
    }
}
```