



Desarrollo de Aplicaciones Distribuidas

2° Examen Parcial

Docente: Reyna Barreto, Benjamín David

Estudiante: Mamani Huayta Elias

Lima-2021

Base de Datos en Heroku

Salesforce Platform

DATA

Databases > postgresql-tetrahdral-83049

SERVICE heroku-postgresql PLAN hobby-dev BILLING APP dad-app-rest-ellias

Overview Durability Settings Dataclips

ADMINISTRATION

Database Credentials

Get credentials for manual connections to this database. [Cancel](#)

Please note that **these credentials are not permanent**. Heroku rotates credentials periodically and updates applications where this database is attached.

Host	ec2-52-23-45-36.compute-1.amazonaws.com
Database	ddttie2qj89527
User	cputibmwqoyjnz
Port	5432
Password	93538d7e91fe664c4aee3ad893580ec4aa5acc8b4b8fc0cabd533bf8239ddfc
URI	postgres://cputibmwqoyjnz:93538d7e91fe664c4aee3ad893580ec4aa5acc8b4b8fc0cabd533bf8239ddfc@ec2-52-23-45-36.compute-1.amazonaws.com:5432/ddttie2qj89527
Heroku CLI	heroku pg:sql postgresql-tetrahdral-83049 --app dad-app-rest-ellias

pgAdmin 4

pgAdmin File Object Tools Help

Browser

- Servers (3)
 - PostgreSQL 12
 - concurso-docencia-ordinaria
 - dad-bd-heroku-pg**
 - Databases (1)
 - ddttie2qj89527
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views

dad-bd-heroku-pg

General Connection SSL SSH Tunnel Advanced

Host name/address ec2-52-23-45-36.compute-1.amazonaws.com

Port 5432

Maintenance database ddttie2qj89527

Username cputibmwqoyjnz

Role

Service

Cancel Reset Save

Schemas (1)

- public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (7)
 - cliente
 - detalle
 - empleado
 - producto
 - rol
 - usuario
 - venta
 - Trigger Functions

Visual Studio Code

Venta.controller

The image displays two screenshots of the Visual Studio Code editor, showing the implementation of the `venta.controller.js` file. The Explorer panel on the left shows the project structure, including the `src/controllers` directory where the file is located.

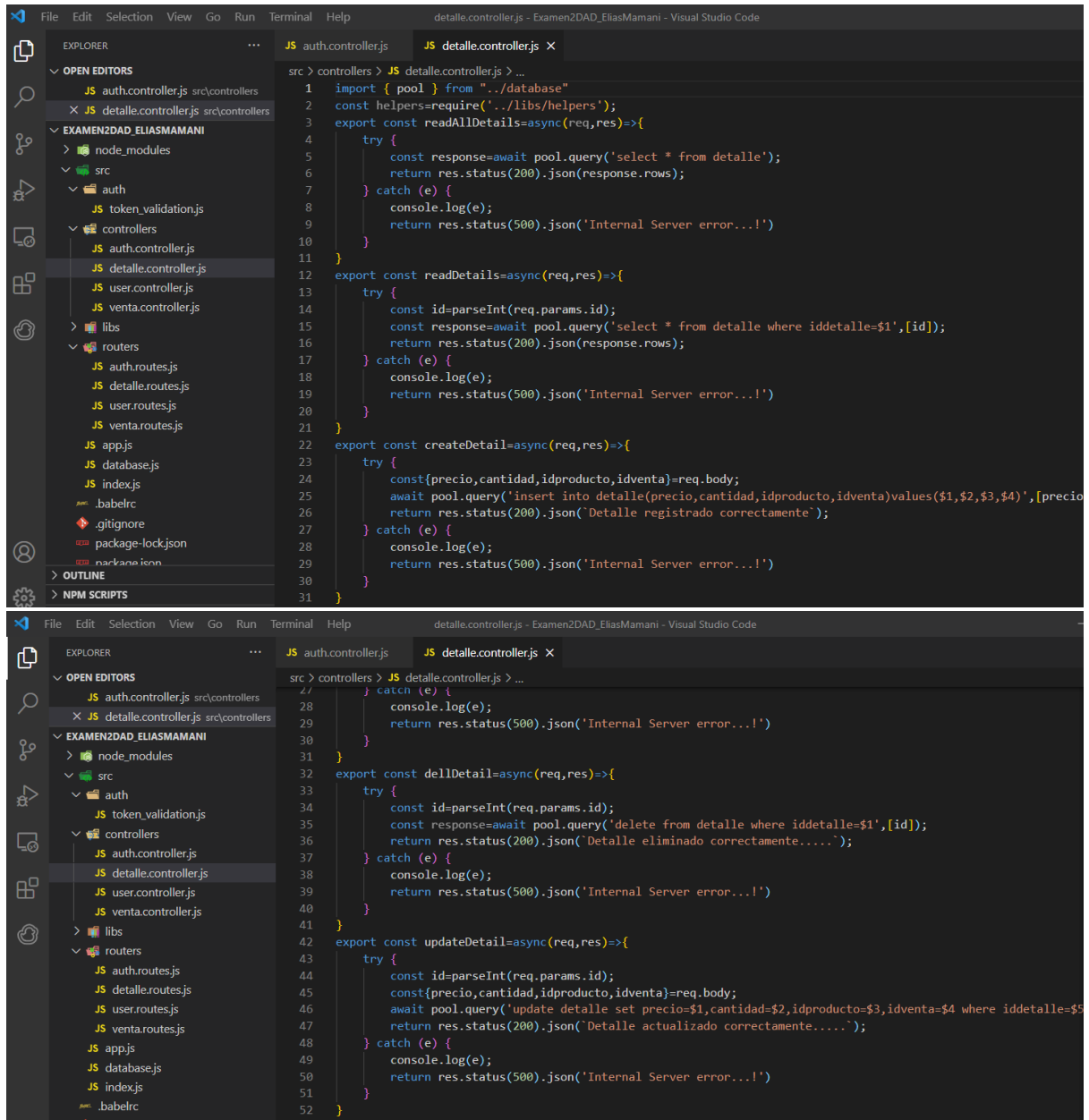
First Screenshot: The code defines three functions: `readAllVenta`, `readVenta`, and `createVenta`.

```
1 import {pool} from "../database"
2 export const readAllVenta=async(req,res)=>{
3   try {
4     const response=await pool.query('select * from venta');
5     return res.status(200).json(response.rows);
6   } catch (e) {
7     console.log(e);
8     return res.status(500).json('Internal Server error...!')
9   }
10 }
11 export const readVenta=async(req,res)=>{
12   try {
13     const id=parseInt(req.params.id);
14     const response=await pool.query('select * from venta where idventa=$1',[id]);
15     return res.status(200).json(response.rows);
16   } catch (e) {
17     console.log(e);
18     return res.status(500).json('Internal Server error...!')
19   }
20 }
21 export const createVenta=async(req,res)=>{
22   try {
23     const {fecha,tipodoc,numdoc,idcliente,idusuario,idempleado}=req.body;
24     await pool.query('insert into matricula(fecha,tipodoc,numdoc,idcliente,idusuario,idempleado)values($1,$2,$3,$4,$5,$6)');
25     return res.status(200).json('Venta realizada');
26   } catch (e) {
27     console.log(e);
28     return res.status(500).json('Internal Server error...!')
29   }
30 }
```

Second Screenshot: The code is extended with two more functions: `delVenta` and `updateVenta`.

```
30 }
31 export const delVenta=async(req,res)=>{
32   try {
33     const id=parseInt(req.params.id);
34     const response=await pool.query('delete from venta where idventa=$1',[id]);
35     return res.status(200).json('Venta eliminada correctamente.....');
36   } catch (e) {
37     console.log(e);
38     return res.status(500).json('Internal Server error...!')
39   }
40 }
41 export const updateVenta=async(req,res)=>{
42   try {
43     const id=parseInt(req.params.id);
44     const {fecha,tipodoc,numdoc,idcliente,idusuario,idempleado}=req.body;
45     await pool.query('update venta set fecha=$1,tipodoc=$2,numdoc=$3,idcliente=$4,idusuario=$5,idempleado=$6 where idventa=$7');
46     return res.status(200).json('Venta Modificada.....');
47   } catch (e) {
48     console.log(e);
49     return res.status(500).json('Internal Server error...!')
50   }
51 }
```

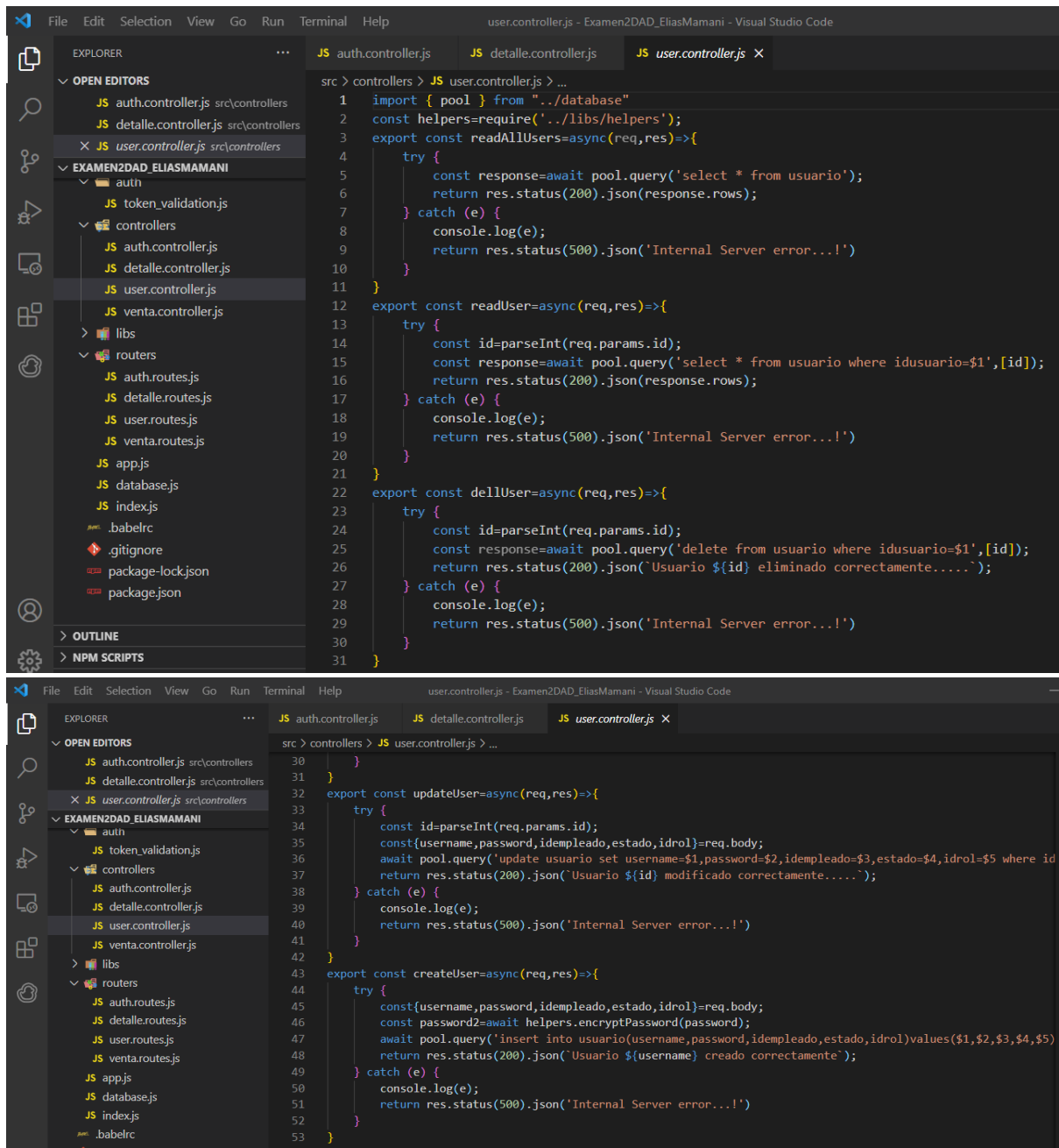
Detalle.controller



```
src > controllers > JS detalle.controller.js > ...
1 import { pool } from "../database"
2 const helpers=require('../libs/helpers');
3 export const readAllDetails=async(req,res)=>{
4   try {
5     const response=await pool.query('select * from detalle');
6     return res.status(200).json(response.rows);
7   } catch (e) {
8     console.log(e);
9     return res.status(500).json('Internal Server error...!')
10  }
11 }
12 export const readDetails=async(req,res)=>{
13   try {
14     const id=parseInt(req.params.id);
15     const response=await pool.query('select * from detalle where iddetalle=$1',[id]);
16     return res.status(200).json(response.rows);
17   } catch (e) {
18     console.log(e);
19     return res.status(500).json('Internal Server error...!')
20   }
21 }
22 export const createDetail=async(req,res)=>{
23   try {
24     const{precio,cantidad,idproducto,idventa}=req.body;
25     await pool.query('insert into detalle(precio,cantidad,idproducto,idventa)values($1,$2,$3,$4)', [precio,cantidad,idproducto,idventa]);
26     return res.status(200).json('Detalle registrado correctamente');
27   } catch (e) {
28     console.log(e);
29     return res.status(500).json('Internal Server error...!')
30   }
31 }

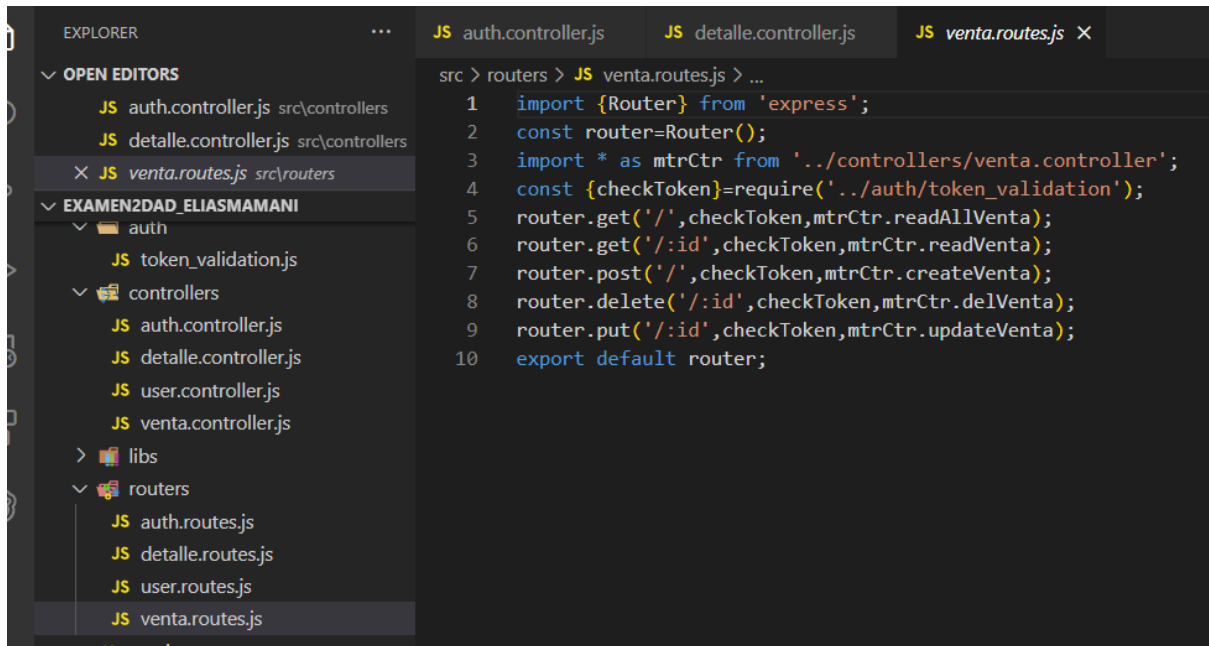
src > controllers > JS detalle.controller.js > ...
28   } catch (e) {
29     console.log(e);
30     return res.status(500).json('Internal Server error...!')
31   }
32 }
33 export const dellDetail=async(req,res)=>{
34   try {
35     const id=parseInt(req.params.id);
36     const response=await pool.query('delete from detalle where iddetalle=$1',[id]);
37     return res.status(200).json('Detalle eliminado correctamente.....');
38   } catch (e) {
39     console.log(e);
40     return res.status(500).json('Internal Server error...!')
41   }
42 }
43 export const updateDetail=async(req,res)=>{
44   try {
45     const id=parseInt(req.params.id);
46     const{precio,cantidad,idproducto,idventa}=req.body;
47     await pool.query('update detalle set precio=$1,cantidad=$2,idproducto=$3,idventa=$4 where iddetalle=$5',[precio,cantidad,idproducto,idventa,id]);
48     return res.status(200).json('Detalle actualizado correctamente.....');
49   } catch (e) {
50     console.log(e);
51     return res.status(500).json('Internal Server error...!')
52   }
53 }
```

User.controller



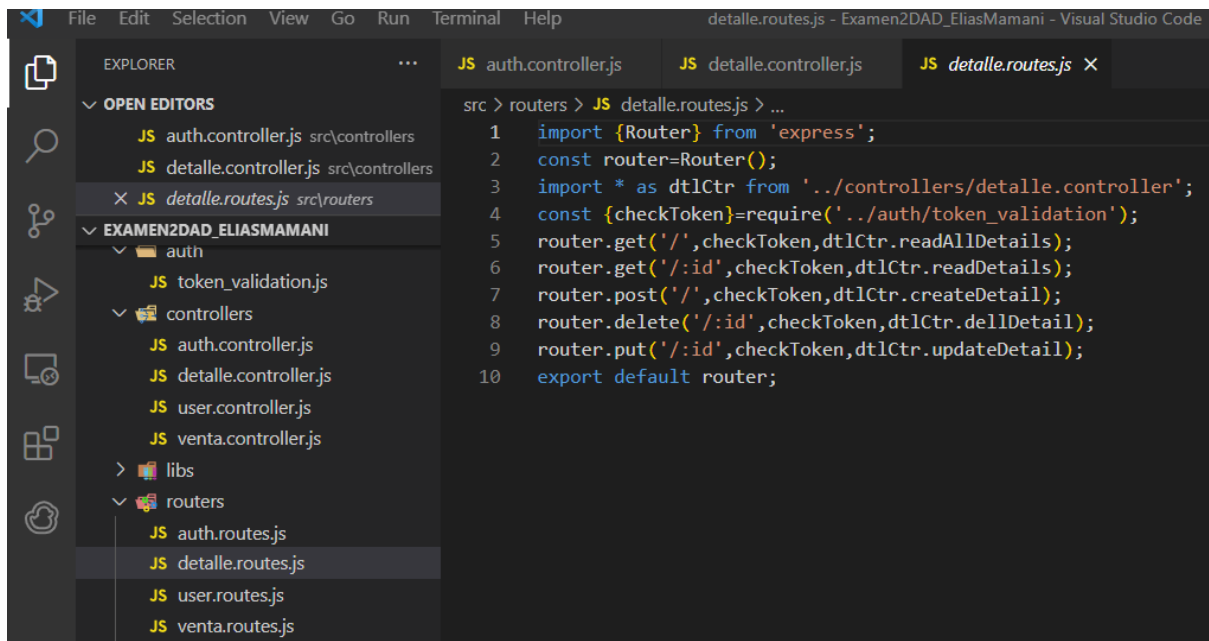
```
src > controllers > JS user.controller.js > ...
1  import { pool } from "../database"
2  const helpers=require('../libs/helpers');
3  export const readAllUsers=async(req,res)=>{
4
5      try {
6          const response=await pool.query('select * from usuario');
7          return res.status(200).json(response.rows);
8      } catch (e) {
9          console.log(e);
10         return res.status(500).json('Internal Server error...!')
11     }
12 }
13 export const readUser=async(req,res)=>{
14     try {
15         const id=parseInt(req.params.id);
16         const response=await pool.query('select * from usuario where idusuario=$1',[id]);
17         return res.status(200).json(response.rows);
18     } catch (e) {
19         console.log(e);
20         return res.status(500).json('Internal Server error...!')
21     }
22 }
23 export const dellUser=async(req,res)=>{
24     try {
25         const id=parseInt(req.params.id);
26         const response=await pool.query('delete from usuario where idusuario=$1',[id]);
27         return res.status(200).json(`Usuario ${id} eliminado correctamente.....`);
28     } catch (e) {
29         console.log(e);
30         return res.status(500).json('Internal Server error...!')
31     }
32 }
33 }
34 export const updateUser=async(req,res)=>{
35     try {
36         const id=parseInt(req.params.id);
37         const {username,password,idempleado,estado,idrol}=req.body;
38         await pool.query('update usuario set username=$1,password=$2,idempleado=$3,estado=$4,idrol=$5 where id=$6',
39             [username,password,idempleado,estado,idrol,id]);
40         return res.status(200).json(`Usuario ${id} modificado correctamente.....`);
41     } catch (e) {
42         console.log(e);
43         return res.status(500).json('Internal Server error...!')
44     }
45 }
46 export const createUser=async(req,res)=>{
47     try {
48         const {username,password,idempleado,estado,idrol}=req.body;
49         const password2=await helpers.encryptPassword(password);
50         await pool.query('insert into usuario(username,password,idempleado,estado,idrol)values($1,$2,$3,$4,$5)',
51             [username,password2,idempleado,estado,idrol]);
52         return res.status(200).json(`Usuario ${username} creado correctamente`);
53     } catch (e) {
54         console.log(e);
55         return res.status(500).json('Internal Server error...!')
56     }
57 }
```

Venta.routes



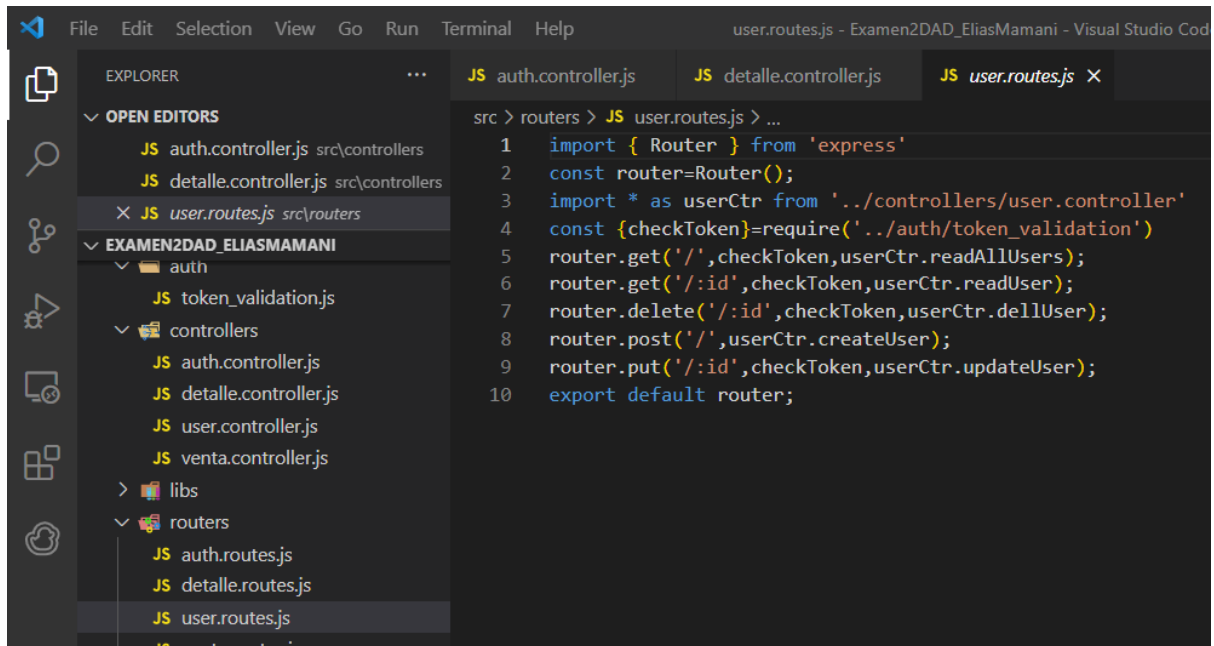
```
src > routers > JS venta.routes.js > ...
1  import {Router} from 'express';
2  const router=Router();
3  import * as mtrCtr from '../controllers/venta.controller';
4  const {checkToken}=require('../auth/token_validation');
5  router.get('/',checkToken,mtrCtr.readAllVenta);
6  router.get('/:id',checkToken,mtrCtr.readVenta);
7  router.post('/',checkToken,mtrCtr.createVenta);
8  router.delete('/:id',checkToken,mtrCtr.delVenta);
9  router.put('/:id',checkToken,mtrCtr.updateVenta);
10 export default router;
```

Detalle.routes



```
src > routers > JS detalle.routes.js > ...
1  import {Router} from 'express';
2  const router=Router();
3  import * as dtlCtr from '../controllers/detalle.controller';
4  const {checkToken}=require('../auth/token_validation');
5  router.get('/',checkToken,dtlCtr.readAllDetails);
6  router.get('/:id',checkToken,dtlCtr.readDetails);
7  router.post('/',checkToken,dtlCtr.createDetail);
8  router.delete('/:id',checkToken,dtlCtr.dellDetail);
9  router.put('/:id',checkToken,dtlCtr.updateDetail);
10 export default router;
```

User.routes



The screenshot shows the Visual Studio Code interface. The EXPLORER sidebar on the left displays the project structure for 'EXAMEN2DAD_ELIASMAMANI'. Under the 'routers' folder, 'user.routes.js' is selected. The main editor window shows the content of 'user.routes.js', which defines an Express.js router for user management.

```
src > routers > JS user.routes.js > ...
1  import { Router } from 'express'
2  const router=Router();
3  import * as userCtr from '../controllers/user.controller'
4  const {checkToken}=require('../auth/token_validation')
5  router.get('/',checkToken,userCtr.readAllUsers);
6  router.get('/:id',checkToken,userCtr.readUser);
7  router.delete('/:id',checkToken,userCtr.dellUser);
8  router.post('/',userCtr.createUser);
9  router.put('/:id',checkToken,userCtr.updateUser);
10 export default router;
```

```
> clase@1.0.0 dev D:\UPEU\Examen2DAD_EliasMamani
> nodemon src/index.js --exec babel-node
```

```
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node src/index.js`
server listen on port 3000
█
```

Postman

Sin Autenticacion

Untitled Request

GET `http://localhost:3000/api/auth/users` Send Save

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** Beautify

```
1 {
2   "password": "123456",
3   "username": "ELIASMH123",
4   "idempleado": 1,
5   "idrol": 1,
6   "estado": "1"
7 }
```

Body Cookies Headers (8) Test Results Status: 401 Unauthorized Time: 10 ms Size: 333 B Save Response

Pretty Raw Preview Visualize **JSON** ≡

```
1 {
2   "success": 0,
3   "message": "Access denied unauthorized user"
4 }
```

Creacion de un Usuario

Launchpad GET http://localhost... GET http://localhost... GET http://localhost... **POST http://localhost...** + ... No Environment 👁 🔗

Untitled Request

POST `http://localhost:3000/api/auth/users` Send Save

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** Beautify

```
1 {
2   "password": "123456",
3   "username": "ELIASMH123",
4   "idempleado": 1,
5   "idrol": 1,
6   "estado": "1"
7 }
```

Body Cookies Headers (8) Test Results Status: 200 OK Time: 1068 ms Size: 308 B Save Response

Pretty Raw Preview Visualize **JSON** ≡

```
1 "Usuario ELIASMH123 creado correctamente"
```


[illegible]

The screenshot shows the Postman application interface. At the top, a GET request is defined for the URL 'http://localhost:3000/api/auth/users'. The 'Authorization' header is set to 'Bearer Token'. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables'. The response body is displayed in JSON format, showing two user objects. The first object has 'idusuario': '6', 'password': '\$2a\$10\$.mnxkK10Fv.Y5NJtkZrFdOnPx.e136E501Eg.oY9RZdrkQskPXHBW', 'username': 'eliasmh', 'idempleado': null, 'idrol': 1, and 'estado': '1'. The second object has 'idusuario': '6', 'password': '\$2a\$10\$2xqHt/zIUxnpCS0q8q6j8eWf0sobDc/T1WxXXnN7j9Uyq6WfEB6q2', 'username': 'ELIASMH123', 'idempleado': 1, 'idrol': 1, and 'estado': '1'.

Listando Ventas

Untitled Request

BUILD

GET

http://localhost:3000/api/auth/ventas

Send

Save

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

Code

TYPE

Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c3VhcmVljp7lmlkdXN1YXJpbyI6IjYiLCJ1c...

Body

Cookies

Headers (8)

Test Results

Status: 200 OK

Time: 738 ms

Size: 414 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

{

"idventa": 4,

"fecha": "2021-02-23T05:00:00.000Z",

"tipodoc": "boleto",

"numdoc": "3654",

"idcliente": 1,

"idusuario": 4,

"idempleado": 1

}

Listando Detalles

GET

http://localhost:3000/api/auth/detalles

Send

Save

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

Cr

TYPE

Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c3VhcmVljp7lmlkdXN1YXJpbyI6IjYiLCJ1c...

Body

Cookies

Headers (8)

Test Results

Status: 200 OK

Time: 883 ms

Size: 339 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

{

"iddetalle": 1,

"precio": "110",

"cantidad": 2,

"idproducto": 3,

"idventa": 4

}

Crear venta

Untitled Request

POST http://localhost:3000/api/auth/ventas

Send Save

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

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

```
1 {
2   "fecha": "2021-02-20",
3   "tipodoc": "factura",
4   "numdoc": "635412",
5   "idcliente": 1,
6   "idusuario": 6,
7   "idempleado": 1
8 }
```

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 726 ms Size: 284 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "Venta realizada"
```

Crear detalle

Launchpad GET http://localhost:3000/api/auth/ventas GET http://localhost:3000/api/auth/ventas POST http://localhost:3000/api/auth/detalles

Untitled Request

POST http://localhost:3000/api/auth/detalles

Send Save

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

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

```
1 {
2   "precio": "130",
3   "cantidad": 2,
4   "idproducto": 2,
5   "idventa": 5
6 }
```

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 757 ms Size: 301 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "Detalle registrado correctamente"
```

Modificando venta

Untitled Request

PUT http://localhost:3000/api/auth/ventas/5

Send Save

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

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

```
1 {
2   "fecha": "2021-02-25",
3   "tipodoc": "boleta",
4   "numdoc": "635412",
5   "idcliente": 1,
6   "idusuario": 6,
7   "idempleado": 1
8 }
```

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 879 ms Size: 290 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "Venta Modificada....."
```

Modificando detalle

PUT ▼ http://localhost:3000/api/auth/detalles/2 Send ▼

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

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ▼

```
1 {
2   "precio": "110",
3   "cantidad": 3,
4   "idproducto": 3,
5   "idventa": 5
6 }
```

Body Cookies Headers (8) Test Results ⊕ Status: 200 OK Time: 962 ms Size: 307 B Save R

Pretty Raw Preview Visualize **JSON** ▼ ≡

```
1 "Detalle actualizado correctamente....."
```

Eliminando venta

DELETE ▼ http://localhost:3000/api/auth/ventas/6 Send ▼ Save

Params Authorization ● Headers (9) **Body** ● Pre-request Script Tests Settings Cookies C

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ▼ Beaut

```
1 {
2   "precio": "110",
3   "cantidad": 3,
4   "idproducto": 3,
5   "idventa": 5
6 }
```

Body Cookies Headers (8) Test Results ⊕ Status: 200 OK Time: 733 ms Size: 303 B Save Respons

Pretty Raw Preview Visualize **JSON** ▼ ≡

```
1 "Venta eliminada correctamente....."
```

Eliminando detalle

DELETE ▼ http://localhost:3000/api/auth/detalles/1 Send ▼

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

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ▼

```
1 {
2   "precio": "110",
3   "cantidad": 3,
4   "idproducto": 3,
5   "idventa": 5
6 }
```

Body Cookies Headers (8) Test Results ⊕ Status: 200 OK Time: 728 ms Size: 305 B Save

Pretty Raw Preview Visualize **JSON** ▼ ≡

```
1 "Detalle eliminado correctamente....."
```

Repositorio Git: https://github.com/EliasMH123/Examen2DAD_EliasMH.git