



Sprint 3. -Configurando el backend

Consultorio Online

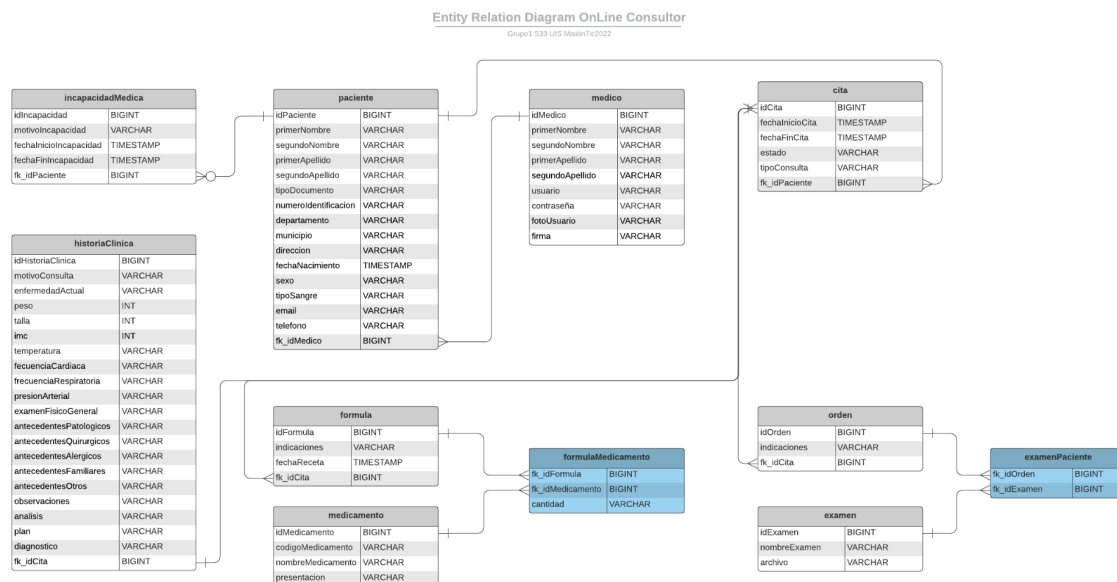
Grupo 1 B24 -16 Noviembre del 2021

Group members: Diego Lesmes, Miller Puentes, Nathalia Moreno, Natanael Barrera, Pedro Ortiz

1. Elabora con tu equipo de trabajo el diagrama relacional de la base de datos.

[Link LucidShare](#)




[Link Drive](#)








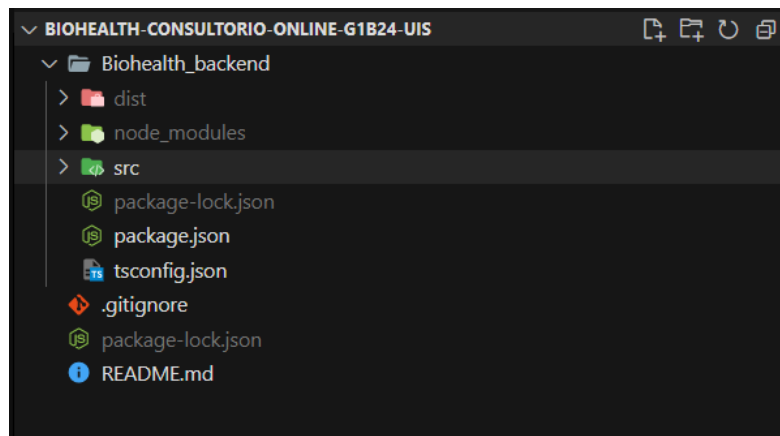
2. Configura la estructura básica del Backend del proyecto, usando los comandos adecuados para cada proceso.

Se crea la estructura básica del Backend en Gitlab

Name	Last commit	Last update
 Biohealth_backend	3rd sprint finished	6 hours ago
 .gitignore	Project started and tested on DLesmes mach...	10 hours ago
 README.md	Update README.md	2 weeks ago

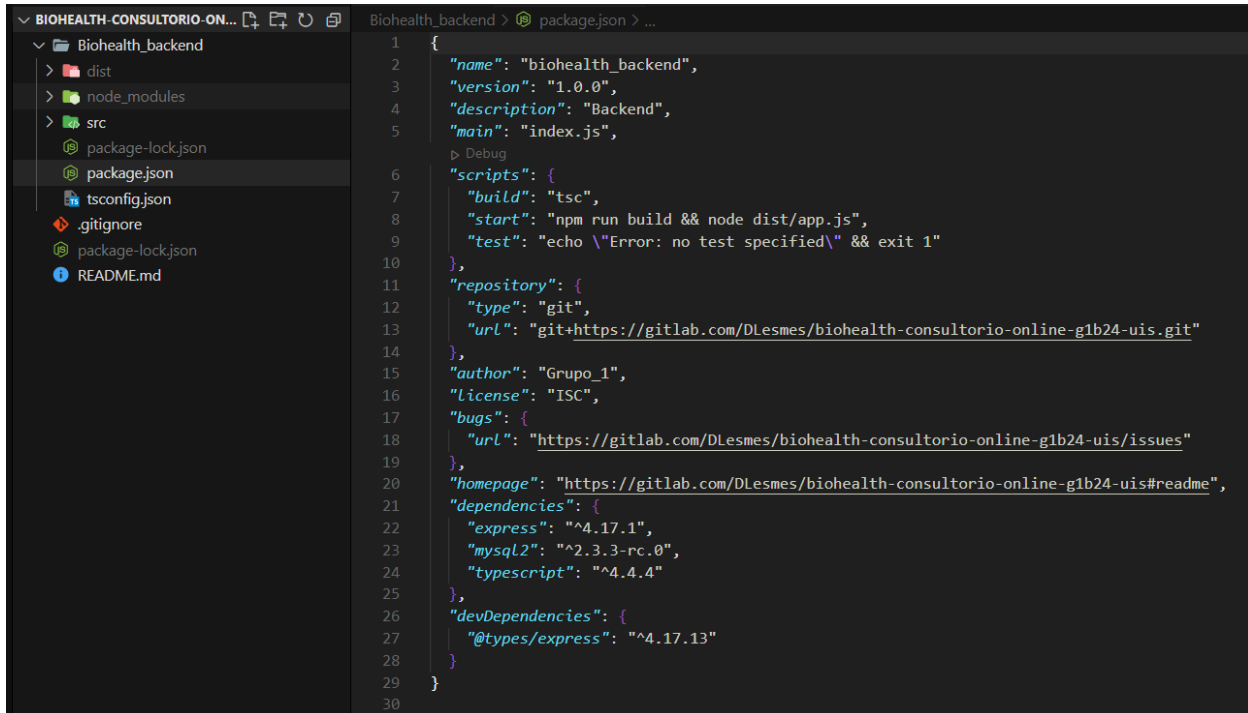
Name	Last commit	Last update
..		
 src	Se modifica la appi Delete	7 minutes ago
 package.json	3rd sprint finished	6 hours ago
 tsconfig.json	Creacion del backend e instalacion de las dependencias	1 week ago

Se clona el repositorio en con VS Code y se instalan las dependencias requeridas



3. Instala y configura las dependencias necesarias para trabajar con MongoDB, Express, JSON y las demás que sean necesarias.

Se instalan las dependencias necesarias para la ejecución del proyecto



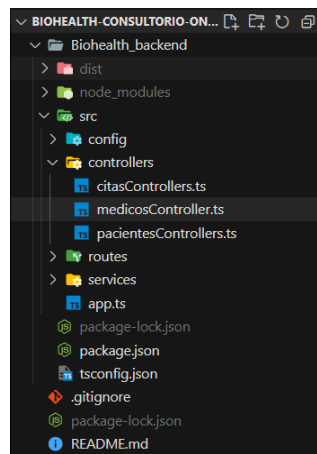
```

1 {
2   "name": "biohealth_backend",
3   "version": "1.0.0",
4   "description": "Backend",
5   "main": "index.js",
6   "scripts": {
7     "build": "tsc",
8     "start": "npm run build && node dist/app.js",
9     "test": "echo \"Error: no test specified\" && exit 1"
10  },
11  "repository": {
12    "type": "git",
13    "url": "git+https://gitlab.com/DLesmes/biohealth-consultorio-online-g1b24-uis.git"
14  },
15  "author": "Grupo_1",
16  "license": "ISC",
17  "bugs": {
18    "url": "https://gitlab.com/DLesmes/biohealth-consultorio-online-g1b24-uis/issues"
19  },
20  "homepage": "https://gitlab.com/DLesmes/biohealth-consultorio-online-g1b24-uis#readme",
21  "dependencies": {
22    "express": "^4.17.1",
23    "mysql2": "^2.3.3-rc.0",
24    "typescript": "^4.4.4"
25  },
26  "devDependencies": {
27    "@types/express": "^4.17.13"
28  }
29 }
30

```

4. Implementa los Modelos y Controladores en Backend del proyecto.

En la carpeta Controllers se crean los controladores necesarios para la ejecución de la aplicación.





Para la clase médico se tienen los siguientes controladores:

```

1  import executeQuery from "../services/mysql.service"
2
3
4  const getMedicos = async (req, res) => {
5    try{
6      const response = await executeQuery('SELECT * FROM medico');
7      const data = {
8        message: `${response.length} datos encontrados`,
9        datos: response.length > 0 ? response : null
10     }
11     res.json(data);
12   }catch(error){
13     console.log(error);
14     res.status(500).send(error);
15   }
16 }
17
18 const getMedico = (req, res) => {
19   const {id} = req.params;
20   executeQuery('SELECT * FROM medico WHERE idMedico = ${id}').then((response) => {
21     const data = {
22       message: `${response.length} datos encontrados`,
23       datos: response.length > 0 ? response[0] : null
24     }
25     res.json(data);
26   }).catch((error) => {
27     console.log(error);
28     res.status(500).send(error);
29   });
30 }
31
32 const addMedico = async(req, res) => {
33   const {primerNombre, segundoNombre, primerApellido, segundoApellido, usuario, contraseña, fotoUsuario, firma} = req.body;
34   try{
35     const response = await executeQuery('INSERT INTO medico (primerNombre, segundoNombre, primerApellido, segundoApellido, usuario, contraseña, fotoUsuario, firma) VALUES (${
36       res.status(201).json({message: 'created', id: response.insertId});
37   }catch(error){
38     console.log(error);
39     res.status(500).send(error);
40   }
41 }
42
43 const updateMedico = async(req, res) => {
44   const {primerNombre, segundoNombre, primerApellido, segundoApellido, usuario, contraseña, fotoUsuario, firma} = req.body;
45   try{
46     const response = await executeQuery('UPDATE medico SET primerNombre = ${primerNombre}', segundoNombre = ${segundoNombre}', primerApellido = ${primerApellido}', segundo
47     console.log(response);
48     if(response.affectedRows > 0){
49       res.json({message: 'updated'});
50     }else{
51       res.status(404).json({message: 'No existe registro con id: ${req.params.id}'})
52     }
53   }catch(error){
54     console.log(error);
55     res.status(500).send(error);
56   }
57 }
58
59 const deleteMedico = async(req, res) => {
60   try{
61     const response = await executeQuery('DELETE FROM medico WHERE idMedico = ${req.params.id}');
62     console.log(response);
63     if(response.affectedRows > 0){
64       res.json({message: 'deleted'});
65     }else{
66       res.status(404).json({message: 'No existe registro con id: ${req.params.id}'})
67     }
68   }catch(error){
69     console.log(error);
70     res.status(500).send(error);
71   }
72 }
73
74 export {getMedicos, getMedico, addMedico, updateMedico, deleteMedico}

```



5. Crea las rutas que permitan realizar las principales tareas con la base de datos Crear, Actualizar, Eliminar y Buscar.

En la carpeta src en el archivo “app.ts” colocamos el puerto y la conexión que tendrá con las rutas:

```

1  import express from 'express';
2  import medicosRoutes from './routes/medicos';
3
4  const app = express();
5  const port = 3000;
6
7  app.use(express.json());
8  app.use(express.urlencoded({ extended: true }));
9
10 medicosRoutes(app);
11
12
13 app.listen(port, () => {
14   return console.log(`servidor corriendo sobre el puerto ${port}`)
15 });
16

```

Luego se crea la carpeta routes y dentro de ella el archivo “medicos.ts” con las rutas que contendrá la aplicación.

```

1  import { Router } from "express";
2  import { addMedico, deleteMedico, getMedico, getMedicos, updateMedico } from "../controllers/medicosController";
3
4  const medicosRoutes = (app) =>{
5     const router = Router();
6     app.use('/', router);
7
8     router.get('/getMedicos', getMedicos);
9     router.get('/getMedico/:id', getMedico);
10    router.post('/addMedico', addMedico);
11    router.put('/updateMedico/:id', updateMedico);
12    router.delete('/deleteMedico/:id', deleteMedico);
13  }
14
15  export default medicosRoutes;

```

Luego para enlazar la base de datos descargamos la librería mysql2 y se consume en los servicios, creando una carpeta dentro de src llamada services y dentro de ella un archivo llamado “mysql.service.ts” para crear la conexión:



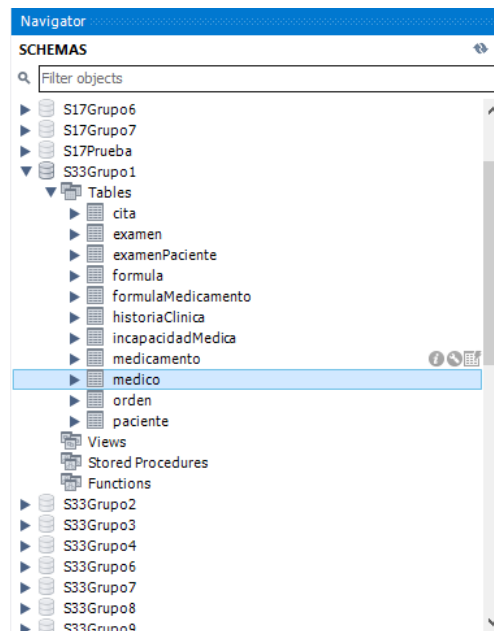
Biohealth_backend > src > services > mysql.service.ts > executeQuery > <function>

```
1  import mysql from 'mysql2';
2  import config from '../config/config';
3
4  const getConnection = () => {
5      const connection = mysql.createConnection({
6          database: config.DATABASE,
7          user: config.DB_USER,
8          password: config.DB_PASSWORD,
9          host: config.DB_HOST,
10         port: +config.DB_PORT
11     });
12     connection.connect((error) => {
13         if(error){
14             throw error;
15         }else{
16             console.log('conexión exitosa');
17         }
18     });
19     return connection;
20 }
```

Biohealth_backend > src > services > mysql.service.ts > default

```
22 const executeQuery = (query: string): Promise<any> => {
23     return new Promise((resolve, reject) => {
24         try{
25             const connection = getConnection();
26             connection.query(query, (error, result) => {
27                 if(error){
28                     reject(error);
29                 }else{
30                     resolve(result);
31                 }
32             });
33             connection.end();
34         }catch(error){
35             console.log(error);
36             reject(error);
37         }
38     })
39 }
40
41 export default executeQuery;
```

La base de datos que se consumirá en el proyecto está alojada en un servidor de la universidad (utilizado en el ciclo anterior) y tiene las siguientes tablas:



La tabla medico que es la que se consumirá en el CRUD, tiene los siguientes campos e información:

	idMedico	primerNombre	segundoNombre	primerApellido	segundoApellido	usuario	contraseña	fotoUsuario	firma
▶	34	Damar	Nicolás	Rojas	Chacón	Nico2595	nico2595	No tiene	Nicolás R.
	35	Pedro	Danjack	Ortiz	Pacheco	Jackie	jackie	No tengo foto	Jackie
	36	Jully	Nathalia	Moreno	Sánchez	NathaMu	nathamu		Natha Moreno
	37	Miller	Albeiro	Puentes	Lozano	MillerP	millerp		Miller P.
	38	Diego	Alejandro	Choco	Lesmes	DiegoChoco	diegochoco		Diego Choco
	39	Profesor		Profesor	Profesor	profesor	profesor		Profesor
	40	Profe	Tutor	Profe	Tutor	tutor	tutor	na	Profe Tutor
	42	Profe	Tutor	1	2	Profe 1	profe	NA	Profe 1
	43	jack	a	a	a	a	a	a	a
	46	Damar3	Nicolás3	Rojas3	Chacón3	Nico25953	nico25953	No tiene3	Nicolás R.2
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Para enlazar la base de datos con la appi, se crea una carpeta llamada config dentro de src y a su vez un archivo llamado “config.ts” con las siguiente estructura (está enlazado con el servidor de la universidad):

```
Biohealth_backend > src > config > ts config.ts > [e] default
1  export default {
2      PORT: process.env.PORT || 3000,
3      DB_HOST: process.env.DB_HOST || '167.99.168.84',
4      DB_USER: process.env.DB_USER || 'mision',
5      DB_PASSWORD: process.env.DB_PASSWORD || 'tics',
6      DATABASE: process.env.DATABASE || 'S33Grupo1',
7      DB_PORT: process.env.DB_PORT || 3306
8  }
```

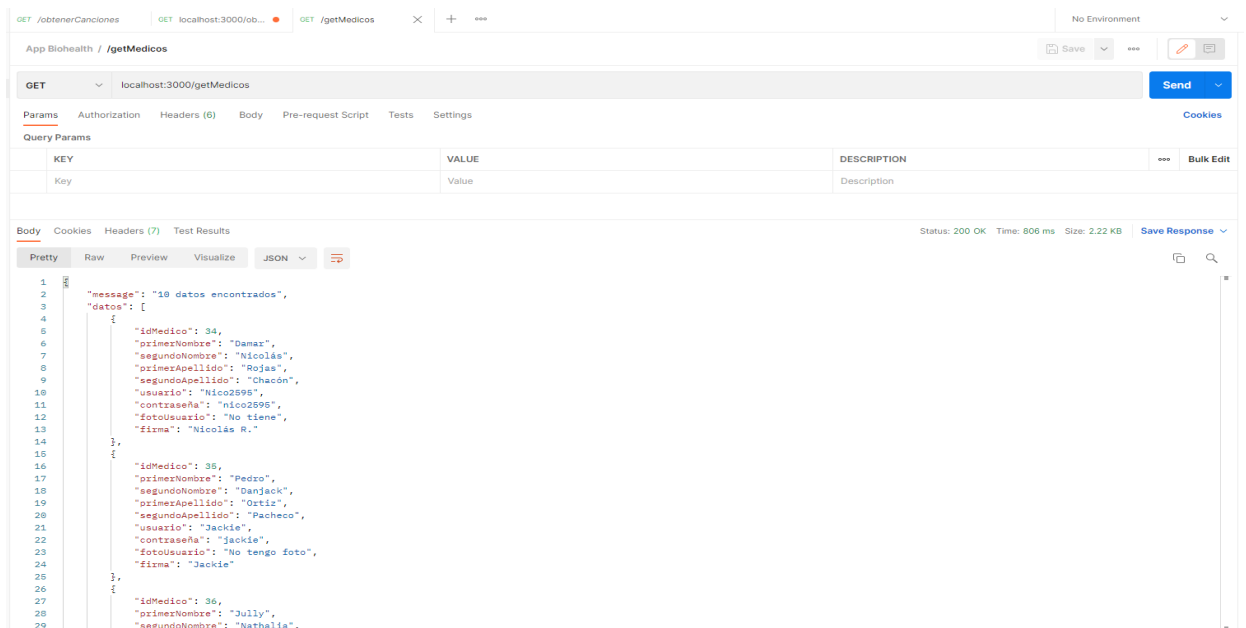


6. Realiza pruebas de las rutas usando Insomnia o Postman.

Se coloca la aplicación a correr con el comando “npm run start” y se ejecutan las peticiones en postman, en el navegador y python (se adjuntará la url del GitHub donde se alojan las peticiones http) de la siguiente manera:

- getMedicos: en esta ruta se deben listar todos los médicos que tiene la base de datos registrados:

En postman:



En el navegador:





En python:

```
response = requests.get('http://localhost:3000/getMedicos')
print(response.status_code)
print(response.text)

✓ 0.7s Python

200
{"message": "10 datos encontrados", "datos": [{"idMedico": 34, "primerNombre": "Damar", "segundoNombre": "Nicolás", "primerApellido": "Rojas", "segundoApellido": "Chacón", "usuario": "Nico2595", "contraseña": "nico2595", "fotoUsuario": "No tiene", "firma": "Nicolás R."}, {"idMedico": 35, "primerNombre": "Pedro", "segundoNombre": "Danjack", "primerApellido": "Ortiz", "segundoApellido": "Pacheco", "usuario": "Jackie", "contraseña": "jackie", "fotoUsuario": "No tengo foto", "firma": "Jackie"}, {"idMedico": 36, "primerNombre": "Jully", "segundoNombre": "Nathalia", "primerApellido": "Moreno", "segundoApellido": "Sánchez", "usuario": "NathaMu", "contraseña": "nathamu", "fotoUsuario": "", "firma": "Natha Moreno"}, {"idMedico": 37, "primerNombre": "Miller", "segundoNombre": "Albeiro", "primerApellido": "Puentes", "segundoApellido": "Lozano", "usuario": "MillerP", "contraseña": "millerp", "fotoUsuario": "", "firma": "Miller P."}, {"idMedico": 38, "primerNombre": "Diego", "segundoNombre": "Alejandro", "primerApellido": "Choco", "segundoApellido": "Lesmes", "usuario": "DiegoChoco", "contraseña": "diegochoco", "fotoUsuario": "", "firma": "Diego Choco"}, {"idMedico": 39, "primerNombre": "Profesor", "segundoNombre": "", "primerApellido": "Profesor", "segundoApellido": "Profesor", "usuario": "profesor", "contraseña": "profesor", "fotoUsuario": "", "firma": "Profesor"}, {"idMedico": 40, "primerNombre": "Profe", "segundoNombre": "Tutor", "primerApellido": "Profe", "segundoApellido": "Tutor", "usuario": "tutor", "contraseña": "tutor", "fotoUsuario": "na", "firma": "Profe Tutor"}, {"idMedico": 42, "primerNombre": "Profe", "segundoNombre": "Tutor", "primerApellido": "1", "segundoApellido": "2", "usuario": "Profe 1", "contraseña": "profe", "fotoUsuario": "NA", "firma": "Profe1"}, {"idMedico": 43, "primerNombre": "jack", "segundoNombre": "a", "primerApellido": "a", "segundoApellido": "a", "usuario": "a", "contraseña": "a", "fotoUsuario": "a", "firma": "a"}, {"idMedico": 46, "primerNombre": "Damar3", "segundoNombre": "Nicolás3", "primerApellido": "Rojas3", "segundoApellido": "Chacón3", "usuario": "Nico25953", "contraseña": "nico25953", "fotoUsuario": "No tiene3", "firma": "Nicolás R.2"}]}
```

- getMedico: con esta ruta se debe traer de la BD el médico registrado con el id que se le suministre, en este caso se requiere traer el registro con id = 36.

medico

Limit to 1000 rows

1 • SELECT * FROM S33Grupo1.medico;

	idMedico	primerNombre	segundoNombre	primerApellido	segundoApellido	usuario	contraseña	fotoUsuario	firma
▶	34	Damar	Nicolás	Rojas	Chacón	Nico2595	nico2595	No tiene	Nicolás R.
	35	Pedro	Danjack	Ortiz	Pacheco	Jackie	jackie	No tengo foto	Jackie
	36	Jully	Nathalia	Moreno	Sánchez	NathaMu	nathamu		Natha Moreno
	37	Miller	Albeiro	Puentes	Lozano	MillerP	millerp		Miller P.
	38	Diego	Alejandro	Choco	Lesmes	DiegoChoco	diegochoco		Diego Choco
	39	Profesor		Profesor	Profesor	profesor	profesor		Profesor
	40	Profe	Tutor	Profe	Tutor	tutor	tutor	na	Profe Tutor
	42	Profe	Tutor	1	2	Profe 1	profe	NA	Profe1
	43	jack	a	a	a	a	a	a	a
	46	Damar3	Nicolás3	Rojas3	Chacón3	Nico25953	nico25953	No tiene3	Nicolás R.2

En postman:

GET /obtenerCanciones GET localhost:3000/ob... GET /getMedico

App Biohealth / /getMedico

GET localhost:3000/getMedico/36

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

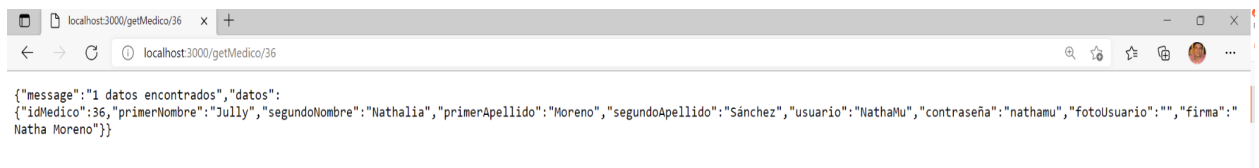
Body Cookies Headers (7) Test Results

Status: 200 OK Time: 754 ms Size: 482 B Save Response

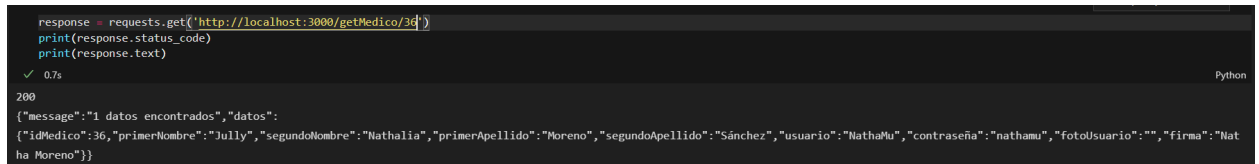
```
1 {
2   "message": "1 datos encontrados",
3   "datos": {
4     "idMedico": 36,
5     "primerNombre": "Jully",
6     "segundoNombre": "Nathalia",
7     "primerApellido": "Moreno",
8     "segundoApellido": "Sánchez",
9     "usuario": "NathaMu",
10    "contraseña": "nathamu",
11    "fotoUsuario": "",
12    "firma": "Natha Moreno"
13  }
14 }
```



En el navegador:

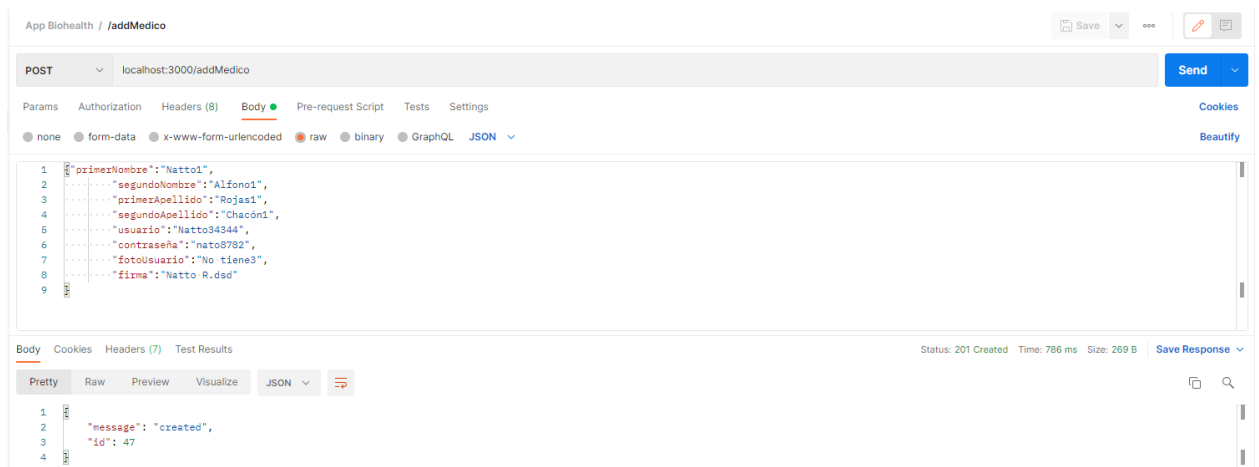


En Python:

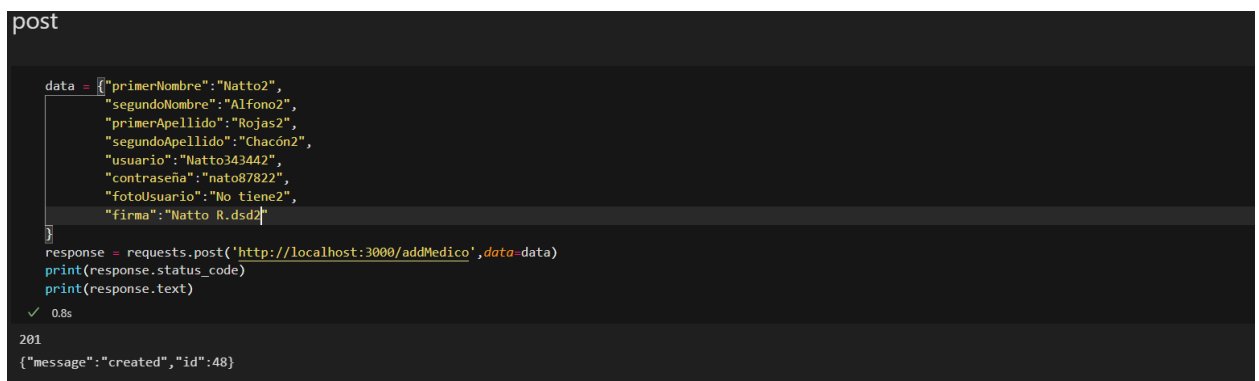


- addMedico: con esta ruta se permite agregar un nuevo usuario:

En Postman:



En Python:





La BD quedaría de la siguiente manera con los 2 registros:

medico

Limit to 1000 rows

1 • `SELECT * FROM S33Grupo1.medico;`

Result Grid

	idMedico	primerNombre	segundoNombre	primerApellido	segundoApellido	usuario	contraseña	fotoUsuario	firma
▶	34	Damar	Nicolás	Rojas	Chacón	Nico2595	nico2595	No tiene	Nicolás R.
	35	Pedro	Danjack	Ortiz	Pacheco	Jackie	jackie	No tengo foto	Jackie
	36	Jully	Nathalia	Moreno	Sánchez	NathaMu	nathamu		Natha Moreno
	37	Miller	Albeiro	Puentes	Lozano	MillerP	millerp		Miller P.
	38	Diego	Alejandro	Choco	Lesmes	DiegoChoco	diegochoco		Diego Choco
	39	Profesor		Profesor	Profesor	profesor	profesor		Profesor
	40	Profe	Tutor	Profe	Tutor	tutor	tutor	na	Profe Tutor
	42	Profe	Tutor	1	2	Profe 1	profe	NA	Profe1
	43	jack	a	a	a	a	a	a	a
	46	Damar3	Nicolás3	Rojas3	Chacón3	Nico25953	nico25953	No tiene3	Nicolás R.2
	47	Natto1	Alfono1	Rojas1	Chacón1	Natto34344	nato8782	No tiene3	Natto R.dsd
	48	Natto2	Alfono2	Rojas2	Chacón2	Natto343442	nato87822	No tiene2	Natto R.dsd2
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- updateMedico: con esta ruta se permite modificar un usuario con solo colocar su id:

En Postman: Se modifica el registro 48

PUT localhost:3000/updateMedico/48

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

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

```

1 {
2   "primerNombre": "Alfredo",
3   "segundoNombre": "Enllo",
4   "primerApellido": "Rojas",
5   "segundoApellido": "Titan",
6   "usuario": "Natto34344",
7   "contraseña": "nato8782",
8   "fotoUsuario": "No tiene3",
9   "firma": "Natto R.dsd"
10 }

```

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 860 ms Size: 256 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "message": "updated"
3 }

```



En Python: Se modifica el registro 39

```
put

data = {
    "primerNombre": "Damaris",
    "segundoNombre": "Nicolá",
    "primerApellido": "Rojas3",
    "segundoApellido": "Chacon",
    "usuario": "Nicolay",
    "contraseña": "nico5241",
    "fotoUsuario": "No tiene3",
    "firma": "Nicolás R.45"
}

response = requests.put('http://localhost:3000/updateMedico/39', data=data)
print(response.status_code)
print(response.text)

✓ 0.8s
200
{"message": "updated"}
```

La Tabla médico en la base de datos queda de la siguiente manera:

medico

Limit to 1000 rows

1 • `SELECT * FROM S33Grupo1.medico;`

Result Grid

Filter Rows:

Edits

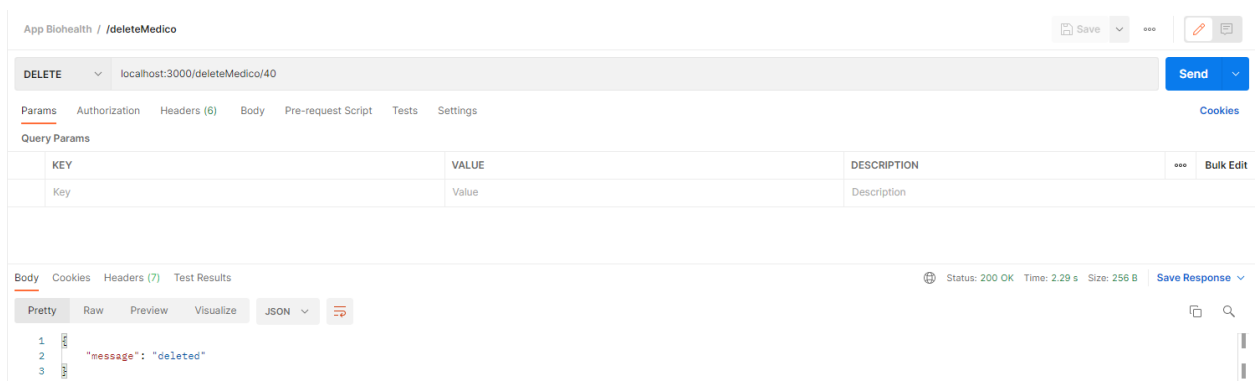
Export/Import:

Wrap Cell Content:

	idMedico	primerNombre	segundoNombre	primerApellido	segundoApellido	usuario	contraseña	fotoUsuario	firma
▶	34	Damar	Nicolás	Rojas	Chacón	Nico2595	nico2595	No tiene	Nicolás R.
	35	Pedro	Danjack	Ortiz	Pacheco	Jackie	jackie	No tengo foto	Jackie
	36	July	Nathalia	Moreno	Sánchez	NathaMu	nathamu		Natha Moreno
	37	Miller	Albeiro	Puentes	Lozano	MillerP	millerp		Miller P.
	38	Diego	Alejandro	Choco	Lesmes	DiegoChoco	diegochoco		Diego Choco
	39	Damaris	Nicola	Rojas3	Chacon	Nicolay	nico5241	No tiene3	Nicolás R. 45
	40	Profe	Tutor	Profe	Tutor	tutor	tutor	na	Profe Tutor
	42	Profe	Tutor	1	2	Profe 1	profe	NA	Profe1
	43	jack	a	a	a	a	a	a	a
	46	Damar3	Nicolás3	Rojas3	Chacón3	Nico25953	nico25953	No tiene3	Nicolás R. 2
	47	Natto1	Alfono 1	Rojas1	Chacón1	Natto34344	nato8782	No tiene3	Natto R. dsd
	48	Alfredo	Emilio	Rojas	Titan	Natto34344	nato8782	No tiene3	Natto R. dsd
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- deleteMedico: con esta ruta se permite eliminar un usuario con solo colocar su id:

En Postman: Se elimina el registro 40





En Python: Se elimina el registro 42

```
delete

response = requests.delete('http://localhost:3000/deleteMedico/42')
print(response.status_code)
print(response.text)

✓ 9.9s

200
{"message": "deleted"}
```

La Tabla médico en la base de datos queda de la siguiente manera con los registros 40 y 42 eliminados:

medico

Limit to 1000 rows

1 • SELECT * FROM S33Grupo1.medico;

	idMedico	primerNombre	segundoNombre	primerApellido	segundoApellido	usuario	contraseña	fotoUsuario	firma
▶	34	Damar	Nicolás	Rojas	Chacón	Nico2595	nico2595	No tiene	Nicolás R.
	35	Pedro	Danjack	Ortiz	Pacheco	Jackie	jackie	No tengo foto	Jackie
	36	Jully	Nathalia	Moreno	Sánchez	NathaMu	nathamu		Natha Moreno
	37	Miller	Albeiro	Puentes	Lozano	MillerP	millerp		Miller P.
	38	Diego	Alejandro	Choco	Lesmes	DiegoChoco	diegochoco		Diego Choco
	39	Damaris	Nicola	Rojas3	Chacon	Nicolay	nico5241	No tiene3	Nicolás R.45
	43	jack	a	a	a	a	a	a	a
	46	Damar3	Nicolás3	Rojas3	Chacón3	Nico25953	nico25953	No tiene3	Nicolás R.2
	47	Natto1	Alfono1	Rojas1	Chacón1	Natto34344	nato8782	No tiene3	Natto R.dsd
	48	Alfredo	Emilio	Rojas	Titan	Natto34344	nato8782	No tiene3	Natto R.dsd

7. Proyecto en Jira

[Enlace del Proyecto en Jira](#)



Proyectos / BioHealth_Grupo1_B24

Backlog

Epic ▾ Tipo ▾

Insights

▼ Sprint 3 10 nov. – 12 nov. (9 incidencias)

1 3 14 Completar sprint ...

<input type="checkbox"/> CDLAC4G1-65 [spike_4] Estudiar Angular	EN CURSO ▾	
<input type="checkbox"/> CDLAC4G1-62 [spike_1] Estudiar Mongo DB	EN CURSO ▾	
<input type="checkbox"/> CDLAC4G1-63 [spike_2] Estudiar JavaScript	EN CURSO ▾	
<input checked="" type="checkbox"/> CDLAC4G1-19 [HU_002] Cómo medico me gustaría iniciar sesión, revisar el perfil de médico, poder editarlo y visualizar un botón de eliminar cuen...	8 FINALIZADA ▾	
<input checked="" type="checkbox"/> CDLAC4G1-76 [TR_021] Realizar ajustes al diagrama de la BD relacional	3 FINALIZADA ▾	
<input checked="" type="checkbox"/> CDLAC4G1-77 [TR_022] Crear documento en drive para el Sprint 3	1 FINALIZADA ▾	
<input checked="" type="checkbox"/> CDLAC4G1-78 Hacer los cambios del diagrama a la BD del DBeaver	3 EN CURSO ▾	
<input checked="" type="checkbox"/> CDLAC4G1-79 Revisión par del documento	1 TAREAS POR HACER ▾	
<input checked="" type="checkbox"/> CDLAC4G1-80 Modificar la petición delete del API	2 FINALIZADA ▾	

Proyectos / BioHealth_Grupo1_B24

Sprint 3

Tipo ▾

POR HACER 1 INCIDENCIA

Revisión par del documento

☒ CDLAC4G1-79 1

EN CURSO 4 INCIDENCIAS

[spike_4] Estudiar Angular

☐ CDLAC4G1-65

[spike_1] Estudiar Mongo DB

☐ CDLAC4G1-62

[spike_2] Estudiar JavaScript

☐ CDLAC4G1-63

Hacer los cambios del diagrama a la BD del DBeaver

☒ CDLAC4G1-78 3

LISTO 4 INCIDENCIAS ✓

[HU_002] Cómo medico me gustaría iniciar sesión, revisar el perfil de médico, poder editarlo y visualizar un botón de eliminar cuenta.

☒ CDLAC4G1-19 ✓ 8

Modificar la petición delete del API

☒ CDLAC4G1-80 ✓ 2

[TR_021] Realizar ajustes al diagrama de la BD relacional

☒ CDLAC4G1-76 ✓ 3

[TR_022] Crear documento en drive para el Sprint 3

☒ CDLAC4G1-77 ✓ 1



8. Repositorio

Enlace a Repositorio en Gitlab

Diego Lesmes > Biohealth_ConsultorioOnline_G1B24UIS > Repository

main biohealth-consultorio-online-g1b24-uis / + History Find file Web IDE Clone

Se modifica la appi Delete
Miller Puentes authored 20 minutes ago a71c85e0

Name	Last commit	Last update
Biohealth_backend	Se modifica la appi Delete	20 minutes ago
.gitignore	Project started and tested on DLesmes mach...	10 hours ago
README.md	Update README.md	2 weeks ago

README.md

Biohealth Consultorio Online

Search GitLab

Diego Lesmes > Biohealth_ConsultorioOnline_G1B24UIS > Commits

main biohealth-consultorio-online-g1b24-uis Author Search by message

13 Nov, 2021 3 commits

Se modifica la appi Delete
Miller Puentes authored 20 minutes ago a71c85e0

3rd sprint finished
Diego Lesmes authored 6 hours ago 8033aa49

Project started and tested on DLesmes machine
Diego Lesmes authored 10 hours ago b0054634

06 Nov, 2021 2 commits

Cleannig the repo of cicle 3 app
Diego Lesmes authored 1 week ago dac52ba5

Creacion del backend e instalacion de las dependencias
Pedro Ortiz authored 1 week ago 183f4c9c

05 Nov, 2021 1 commit