

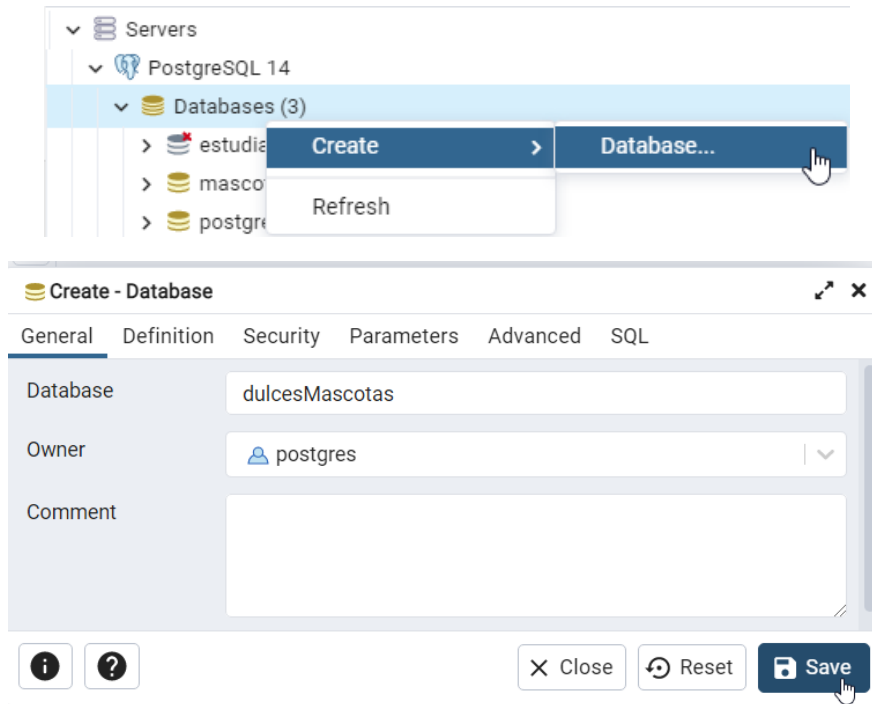
Taller Unidad 2 Backend

Leidy Carolina Botina Botina
Fredery Duvan Buesaquillo Guacales
Daniel Guancha
Cristhian Alejandro Escobar Diaz

Universidad de Nariño
Ingeniería de sistemas
Diplomado de actualización en nuevas tecnologías para el
desarrollo de Software.
San Juan de Pasto
2022

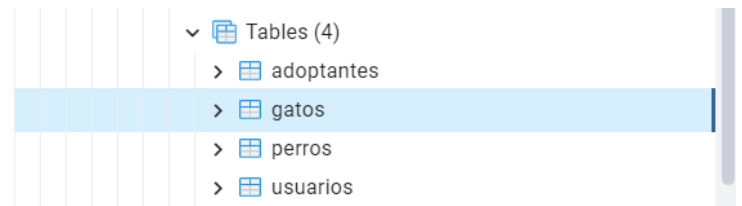
1. Creación de bases de datos postgres.

1.1. Nos dirigimos a PgAdmin y creamos ahí nuestra base de datos con nombre: dulcesMascotas.



1.2. Creación de tablas mediante código en el apartado index.js sincronizando con gestor de base de datos.

```
indexjs > ...
1 import express from 'express';
2 const app = express();
3 import router from './src/Routes/routes.js';
4 import { sequelize } from './src/Database/Database.js';
5 import cors from 'cors';
6 import './src/Models/Adoptante.js';
7 import './src/Models/Gatos.js';
8 import './src/Models/Perros.js';
9
10 const testDb = async () => {
11   try {
12     await sequelize.sync();
13     console.log('Connection has been established successfully.');
```



2. Desarrollo de la aplicación Backend.

2.1. Instalación de bcrypt para cifrar contraseñas.

```
PS D:\Diplomado\Modulo Uno\dulcesMacotasBk> npm i bcrypt
```

2.2. Creación del package.json mediante el comando **npm init -y**.

2

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2> cd .\macotasBk\  
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm init -y
```

2.3. Instalación de express y postgres mediante el comando **npm install express pg**.

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm install express pg
```

2.4. Instalación de nodemon mediante comando **npm install nodemon -D**.

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm install nodemon -D
```

2.5. Instalación sequelize mediante comando **npm install --save sequelize**.

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm install --save sequelize
```

2.6. Instalación pg-hstore mediante comando **npm install --save pg-hstore**.

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm install --save pg-hstore
```

2.7. Instalación cors mediante comando **npm install cors**.

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm install cors
```

2.8. Configuración completa del archivo **package.json**.

```
package.json X  
package.json > ...  
1 {  
2   "name": "macotasbk",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "type": "module",  
7   "scripts": {  
8     "test": "nodemon index.js"  
9   },  
10  "keywords": [],  
11  "author": "",  
12  "license": "ISC",  
13  "dependencies": {  
14    "bcrypt": "^5.1.0",  
15    "bcryptjs": "^2.4.3",  
16    "cors": "^2.8.5",  
17    "express": "^4.18.2",  
18    "pg": "^8.8.0",  
19    "pg-hstore": "^2.3.4",  
20    "sequelize": "^6.24.0"  
21  },  
22  "devDependencies": {  
23    "nodemon": "^2.0.20"  
24  }  
25 }
```

2.9. Creación de la estructura del proyecto con sus respectivos controladores, módulos y rutas.



2.10. Implementación archivo **ControllerUsuario.js** con su respectivo get, post, login, put, y delete Usuario.

```
ControllerUsuario.js M X
src > Controllers > ControllerUsuario.js > postusuario
1 import { Usuario } from '../Models/Usuarios.js';
2 import bcrypt from 'bcryptjs';
3 //const { bcrypt } = bcrypt;
4
5 const getUsuario = async (req, res) => {
6   try {
7     const usuarios = await Usuario.findAll();
8     res.status(200).json(usuarios);
9   } catch (error) {
10    res.status(400).json({ mensaje: `${error}` });
11  }
12 }
13
14
15 const postusuario = async (req, res) => {
16   try {
17     const { id } = req.body;
18     const { nombre } = req.body;
19     const { correo } = req.body;
20     const { contraseña } = req.body;
21     const newUsuario = await Usuario.create({
22       id,
23       nombre,
24       correo,
25       contraseña: await bcrypt.hash(contraseña, 10)
26     });
27     res.status(200).json(newUsuario);
28   } catch (error) {
29     res.status(400).json({ mensaje: `${error}` });
30   }
31 }
32
33
34 const login = async (req, res) => {
35   try {
36     const { contraseña } = req.body;
37     const { correo } = req.body;
38
39     const user = await Usuario.findOne({ where: { correo: correo } });
40     const isSame = await bcrypt.compare(contraseña, user.contraseña);
41     if (isSame) {
42       res.status(200).json(user);
43     }
44   } catch (error) {
45     res.status(400).json({ mensaje: `${error}` });
46   }
47 }
48 }
```

```
50 const putUsuario = async (req, res) => {
51   try {
52     const { id } = req.params;
53     const { nombre } = req.body;
54     const { correo } = req.body;
55     const { contraseña } = req.body;
56
57     const oldUsuario = await Usuario.findById(id);
58
59     oldUsuario.nombre=nombre;
60     oldUsuario.correo=correo;
61     oldUsuario.contraseña=contraseña;
62     const modUsuario = await oldUsuario.save();
63     res.status(200).json(modUsuario);
64   } catch (error) {
65     res.status(400).json({ mensaje: `${error}` });
66   }
67 }
68
69
70 const deleteUsuario = async (req, res) => {
71   try {
72     const { id } = req.params;
73     const respuesta = await Usuario.destroy({
74       where: {
75         id
76       }
77     });
78     res.status(200).json({
79       body: {
80         id
81       }
82     });
83   } catch (error) {
84     res.status(400).json({ mensaje: `${error}` });
85   }
86 }
87
88
89 export {
90   getUsuario,
91   postusuario,
92   putUsuario,
93   deleteUsuario,
94   login
95 }
```

2.11. Implementación archivo **ControllerGatos.js** con su respectivo get, post, login, put, y delete Gatos.

```
ControllerGatos.js X
src > Controllers > ControllerGatos.js > postGatos
1 import { Gatos } from '../Models/Gatos.js';
2
3 const getGatos = async (req, res) => {
4   try {
5     const gatos = await Gatos.findAll();
6     res.status(200).json(gatos);
7   } catch (error) {
8     res.status(400).json({ mensaje: `${error}` });
9   }
10 }
11
12 const postGatos = async (req, res) => {
13   try {
14     const { id, foto, nombre, edad, talla, descripcion } = req.body;
15     const newGato = await Gatos.create({
16       id,
17       foto,
18       nombre,
19       edad,
20       talla,
21       descripcion
22     });
23     res.status(200).json(newGato);
24   } catch (error) {
25     res.status(400).json({ mensaje: `${error}` });
26   }
27 }
28
29 const putGatos = async (req, res) => {
30   try {
31     const { id } = req.params;
32     const { foto } = req.body;
33     const { nombre } = req.body;
34     const { edad } = req.body;
35     const { talla } = req.body;
36     const { descripcion } = req.body;
37     const oldGatos = await Gatos.findByPk(id);
38     oldGatos.foto = foto;
39     oldGatos.nombre = nombre;
40     oldGatos.edad = edad;
41     oldGatos.talla = talla;
42     oldGatos.descripcion = descripcion;
43     const modGato = await oldGatos.save();
44     res.status(200).json(modGato);
45   } catch (error) {
46     res.status(400).json({ mensaje: `${error}` });
47   }
48 }
```

```
50 const deleteGatos = async (req, res) => {
51   try {
52     const { id } = req.params;
53
54     const respuesta = await Gatos.destroy({
55       where: {
56         id
57       }
58     });
59
60     res.status(200).json({
61       body: {
62         id
63       }
64     });
65   } catch (error) {
66     res.status(400).json({ mensaje: `${error}` });
67   }
68 }
69
70
71
72
73
74 export {
75   getGatos,
76   postGatos,
77   putGatos,
78   deleteGatos
79 }
```

2.12. Implementación archivo **ControllerPerros.js** con su respectivo get, post, login, put, y delete Perros.

```
ControllerPerros.js M X
src > Controllers > ControllerPerros.js > postPerros
1 import { Perros } from '../Models/Perros.js';
2
3 const getPerros = async (req, res) => {
4   try {
5     const perros = await Perros.findAll();
6     res.status(200).json(perros);
7   } catch (error) {
8     res.status(400).json({ mensaje: `${error}` });
9   }
10 }
11
12 const postPerros = async (req, res) => {
13   try {
14     const { id, foto, nombre, edad, talla, descripcion } = req.body;
15     const newPerro = await Perros.create({
16       id,
17       foto,
18       nombre,
19       edad,
20       talla,
21       descripcion
22     });
23     res.status(200).json(newPerro);
24   } catch (error) {
25     res.status(400).json({ mensaje: `${error}` });
26   }
27 }
28
29 const putPerros = async (req, res) => {
30   try {
31     const { id } = req.params;
32     const { foto } = req.body;
33     const { nombre } = req.body;
34     const { edad } = req.body;
35     const { talla } = req.body;
36     const { descripcion } = req.body;
37     const oldPerro = await Perros.findByPk(id);
38     oldPerro.foto = foto;
39     oldPerro.nombre = nombre;
40     oldPerro.edad = edad;
41     oldPerro.talla = talla;
42     oldPerro.descripcion = descripcion;
43     const modPerro = await oldPerro.save();
44     res.status(200).json(modPerro);
45   } catch (error) {
46     res.status(400).json({ mensaje: `${error}` });
47   }
48 }
```

```
50 const deletePerros = async (req, res) => {
51   try {
52     const { id } = req.params;
53
54     const respuesta = await Perros.destroy({
55       where: {
56         id
57       }
58     });
59
60     res.status(200).json({
61       body: {
62         id
63       }
64     });
65   } catch (error) {
66     res.status(400).json({ mensaje: `${error}` });
67   }
68 }
69
70
71
72
73
74 export {
75   getPerros,
76   postPerros,
77   putPerros,
78   deletePerros
79 }
```

2.13. Implementación archivo **Database.js**.

```
Database.js X
src > Database > Database.js > ...
1  import Sequelize from "sequelize";
2
3  const sequelize = new Sequelize('dulcesMascotas', 'postgres', '12345678', {
4    host: 'localhost',
5    dialect: 'postgres'
6  });
7
8  export {
9    sequelize
10 }
```

2.14. Implementación archivo **Usuarios.js**.

```
Usuarios.js X
src > Models > Usuarios.js > ...
1  import { DataTypes } from 'sequelize';
2  import { sequelize } from '../Database/Database.js';
3
4  const Usuario = sequelize.define('usuario', {
5    id: {
6      type: DataTypes.INTEGER,
7      primaryKey: true
8    },
9    nombre: {
10     type: DataTypes.STRING
11   },
12   correo: {
13     type: DataTypes.STRING
14   },
15   contraseña: {
16     type: DataTypes.STRING,
17     unique: true
18   }
19 });
20
21 export {
22   Usuario
23 }
24 |
```

2.15. Implementación archivo **Adoptante.js**.

```
Adoptante.js X
src > Models > Adoptante.js > ...
1  import { DataTypes } from 'sequelize';
2  import { sequelize } from '../Database/Database.js';
3
4  const Adoptante = sequelize.define('adoptante', {
5    id: {
6      type: DataTypes.INTEGER,
7      primaryKey: true
8    },
9    nombre: {
10     type: DataTypes.STRING
11   },
12   correo: {
13     type: DataTypes.STRING
14   },
15   telefono: {
16     type: DataTypes.STRING
17   }
18 });
19
20 export {
21   Adoptante
22 }
```

2.16. Implementación archivo **Perros.js**.

```
Perros.js X
src > Models > Perros.js > ...
1 import { DataTypes } from 'sequelize';
2 import { sequelize } from '../Database/Database.js';
3
4 const Perros = sequelize.define('perros', {
5   id: {
6     type: DataTypes.INTEGER,
7     primaryKey: true,
8     autoIncrement: true
9   },
10  foto: {
11    type: DataTypes.STRING
12  },
13  nombre: {
14    type: DataTypes.STRING
15  },
16  edad: {
17    type: DataTypes.STRING,
18  },
19  talla: {
20    type: DataTypes.STRING
21  },
22  descripcion: {
23    type: DataTypes.STRING
24  }
25 });
26
27 export {
28   Perros
29 }
```

2.17. Implementación archivo **Gatos.js**.

```
Gatos.js X
src > Models > Gatos.js > ...
1 import { DataTypes } from 'sequelize';
2 import { sequelize } from '../Database/Database.js';
3
4 const Gatos = sequelize.define('gatos', {
5   id: {
6     type: DataTypes.INTEGER,
7     primaryKey: true,
8     autoIncrement: true
9   },
10  foto: {
11    type: DataTypes.STRING
12  },
13  nombre: {
14    type: DataTypes.STRING
15  },
16  edad: {
17    type: DataTypes.STRING,
18  },
19  talla: {
20    type: DataTypes.STRING
21  },
22  descripcion: {
23    type: DataTypes.STRING
24  }
25 });
26
27 export {
28   Gatos
29 }
```

2.18. Implementación archivo **routes.js**.

```
routes.js  X
src > Routes > routes.js > [0] default
1  import { Router } from 'express';
2  import { deleteAdoptante, getAdoptante, postAdoptante, putAdoptante } from '../Controllers/ControllerAdoptante.js';
3  import { getGatos, postGatos, putGatos, deleteGatos } from '../Controllers/ControllerGatos.js';
4  import { getPerros, postPerros, putPerros, deletePerros } from '../Controllers/ControllerPerros.js';
5  import { postusuario, login } from '../Controllers/ControllerUsuario.js';
6
7  const router=Router();
8
9  router.get('/',(req,res)->{
10 |   res.send("Hola estamos en Index");
11 | });
12
13 router.post('/usuario',postusuario);
14 router.post('/login',login);
15
16 router.get('/gatos',getGatos);
17
18 router.post('/gatos',postGatos);
19
20 router.put('/gatos/:id',putGatos);
21
22 router.delete('/gatos/:id',deleteGatos);
23
24
25 router.get('/perros',getPerros);
26
27 router.post('/perros',postPerros);
28
29 router.put('/perros/:id',putPerros);
30
31 router.delete('/perros/:id',deletePerros);
32
33
34 router.get('/adoptante',getAdoptante);
35
36 router.post('/adoptante',postAdoptante);
37
38 router.put('/adoptante/:id',putAdoptante);
39
40 router.delete('/adoptante/:id',deleteAdoptante);
41
42 export default router;
```

3. VERIFICACION DE OPERACIONES A TRAVEZ DE INSOMNIA.

3.1. Post adoptante.

POST http://127.0.0.1:5000/adoptante Send 200 OK 147 ms 158 B

JSON Auth Query Headers 1 Preview Headers 8 Cookies Time

```
1 {
2   "nombre": "Fredery",
3   "correo": "holamundo@gmail.com",
4   "telefono": "12345678"
5 }
```

```
1 {
2   "id": 1,
3   "nombre": "Fredery",
4   "correo": "holamundo@gmail.com",
5   "telefono": "12345678",
6   "updatedAt": "2022-10-13T23:54:32.486Z",
7   "createdAt": "2022-10-13T23:54:32.486Z"
8 }
```

3.2. Get adoptante.

GET http://127.0.0.1:5000/adoptante Send 200 OK 87.3 ms 160 B

Body Auth Query Headers Docs Preview Headers 8 Cookies Time

```
1 [
2   {
3     "id": 1,
4     "nombre": "Fredery",
5     "correo": "holamundo@gmail.com",
6     "telefono": "12345678",
7     "createdAt": "2022-10-10T22:24:46.096Z",
8     "updatedAt": "2022-10-10T22:24:46.096Z"
9   }
10 ]
```


3.3. Put adoptante.

PUT http://127.0.0.1:5000/adoptante/1 Send 200 OK 78.4 ms 156 B

JSON Auth Query Headers 1 Docs Preview Headers 8 Cookies Time

```
1 {
2   "nombre": "Duvan"
3 }
```

```
1 {
2   "id": 1,
3   "nombre": "Duvan",
4   "correo": "holamundo@gmail.com",
5   "telefono": "12345678",
6   "createdAt": "2022-10-10T22:24:46.096Z",
7   "updatedAt": "2022-10-10T22:27:44.483Z"
8 }
```

3.4. Delete adoptante.

DELETE http://127.0.0.1:5000/adoptante/1 Send 200 OK 176 ms

Body Auth Query Headers Docs Preview Headers

```
1 {
2   "body": {
3     "id": "1"
4   }
5 }
```

3.5. Post perros.

POST http://127.0.0.1:5000/perros Send 200 OK 188 ms 188 B

JSON Auth Query Headers 1 Docs Preview Headers 8 Cookies Time

```
1 {
2   "id": 1,
3   "nombre": "Toby",
4   "raza": "Pastor Aleman",
5   "descripcion": "Nomble",
6   "detalles": "alcanzan como mazimo 65 cm"
7 }
```

```
1 {
2   "id": 1,
3   "nombre": "Toby",
4   "raza": "Pastor Aleman",
5   "descripcion": "Nomble",
6   "detalles": "alcanzan como mazimo 65 cm",
7   "updatedAt": "2022-10-10T22:30:51.864Z",
8   "createdAt": "2022-10-10T22:30:51.864Z"
9 }
```

3.6. Get perros.

GET http://127.0.0.1:5000/perros Send 200 OK 71.2 ms 190 B

Body Auth Query Headers Docs Preview Headers 8 Cookies Timeli

```
1 [
2   {
3     "id": 1,
4     "nombre": "Toby",
5     "raza": "Pastor Aleman",
6     "descripcion": "Nomble",
7     "detalles": "alcanzan como mazimo 65 cm",
8     "updatedAt": "2022-10-10T22:30:51.864Z",
9     "createdAt": "2022-10-10T22:30:51.864Z"
10  }
11 ]
```

3.7. Put perros.

PUT http://127.0.0.1:5000/perros/1 Send 200 OK 191 ms 112 B

JSON Auth Query Headers 1 Docs Preview Headers 8 Cookies Timeline

```
1 {
2   "descripcion": "juguetón"
3 }
```

```
1 {
2   "id": 1,
3   "descripcion": "juguetón",
4   "createdAt": "2022-10-10T22:30:51.864Z",
5   "updatedAt": "2022-10-10T22:33:55.954Z"
6 }
```

3.8. Delete perros.

DELETE http://127.0.0.1:5000/perros/1 Send 200 OK 74.2 ms 19 B

Body Auth Query Headers Docs Preview Headers 8 Cookies Timeline

```
1 {
2   "body": {
3     "id": "1"
4   }
5 }
```

3.9. Post gatos.

POST http://127.0.0.1:5000/gatos Send 200 OK 101 ms 194 B

JSON Auth Query Headers 1 Docs Preview Headers 8 Cookies Timeline

```
1 {
2   "id": 1,
3   "nombre": "Blue",
4   "raza": "Común",
5   "descripcion": "Traviieso",
6   "detalles": "juegan todo el día, les gusta el sol"
7 }
```

```
1 {
2   "id": 1,
3   "nombre": "Blue",
4   "raza": "Común",
5   "descripcion": "Traviieso",
6   "detalles": "juegan todo el día, les gusta el sol",
7   "updatedAt": "2022-10-10T22:41:04.947Z",
8   "createdAt": "2022-10-10T22:41:04.947Z"
9 }
```

3.10. Get gatos.

GET http://127.0.0.1:5000/gatos Send 200 OK 74.3 ms 196 B

Body Auth Query Headers Docs Preview Headers 8 Cookies Timeline

```
1 [
2   {
3     "id": 1,
4     "nombre": "Blue",
5     "raza": "Común",
6     "descripcion": "Traviieso",
7     "detalles": "juegan todo el día, les gusta el sol",
8     "createdAt": "2022-10-10T22:41:04.947Z",
9     "updatedAt": "2022-10-10T22:41:04.947Z"
10  }
11 ]
```

3.11. Put gatos.

PUT http://127.0.0.1:5000/gatos/1 | Send | 200 OK | 52.2 ms | 101 B

JSON | Auth | Query | Headers 1 | Docs | Preview | Headers 8 | Cookies | Timeline

```
1 {
2   "raza": "Persa"
3 }
```

```
1 {
2   "id": 1,
3   "raza": "Persa",
4   "createdAt": "2022-10-10T22:41:04.947Z",
5   "updatedAt": "2022-10-10T22:43:28.326Z"
6 }
```

3.12. Delete gatos.

DELETE http://127.0.0.1:5000/gatos/1 | Send | 200 OK | 104 ms

Body | Auth | Query | Headers | Docs | Preview | Headers

```
1 {
2   "body": {
3     "id": "1"
4   }
5 }
```

3.13. Post usuario.

POST http://127.0.0.1:5000/usuario | Send | 200 OK | 301 ms | 215 B

JSON | Auth | Query | Headers 1 | Docs | Preview | Headers 8 | Cookies | Timeline

```
1 {
2   "nombre": "Fredery",
3   "correo": "holamundote@gmail.com",
4   "contraseña": "123456789"
5 }
```

```
1 {
2   "id": 1,
3   "nombre": "Fredery",
4   "correo": "holamundote@gmail.com",
5   "contraseña": "$2a$10$s4kFoh56LLUvCGMHZetB0hNJVDmsvwsS/I6FskmyeV3qU1o0bp4W",
6   "updatedAt": "2022-10-14T00:09:27.726Z",
7   "createdAt": "2022-10-14T00:09:27.726Z"
8 }
```

3.14. Login usuario.

POST http://127.0.0.1:5000/login | Send | 200 OK | 222 ms | 215 B

JSON | Auth | Query | Headers 1 | Docs | Preview | Headers 8 | Cookies | Timeline

```
1 {
2   "correo": "holamundote@gmail.com",
3   "contraseña": "123456789"
4 }
```

```
1 {
2   "id": 1,
3   "nombre": "Fredery",
4   "correo": "holamundote@gmail.com",
5   "contraseña": "$2a$10$s4kFoh56LLUvCGMHZetB0hNJVDmsvwsS/I6FskmyeV3qU1o0bp4W",
6   "createdAt": "2022-10-14T00:09:27.726Z",
7   "updatedAt": "2022-10-14T00:09:27.726Z"
8 }
```