

UNIVERSIDAD DE NARIÑO  
FACULTAD DE INGENIERÍA  
PROGRAMA DE INGENIERÍA DE SISTEMAS  
DIPLOMADO DE ACTUALIZACIÓN EN NUEVAS TECNOLOGÍAS PARA DESARROLLO DE SOFTWARE

TALLER UNIDAD 2 – BACKEND

PRESENTADO POR:  
JOSÉ ALEJANDRO CASTRILLÓN ORTEGA

DOCENTE  
VICENTE AUX REVELO

SAN JUAN DE PASTO  
17 DE DICIEMBRE DE 2023

## SOLUCION TALLER

Creación de base de datos en MySQL para llevar registro de mascotas (perros y gatos), y el proceso de adopción de estas.

- En mysql fue creada la base de datos **petsdb** y para acceder a esta el usuario **petsadmin**.
- Inicialmente fue creada la tabla **pet\_types** para almacenar los tipos de mascota disponibles.

```
petsadmin [ petsdb ]
# describe pet_types
```

Field	Type	Null	Key	Default	Extra
id	char(1)	NO	PRI	<null>	
description	varchar(20)	NO		<null>	

```
petsadmin [ petsdb ]
# select * from pet_types;
```

id	description
C	Cat
D	Dog

- Y la tabla **adoption\_statuses** para los tipos de estado de las mascotas y las solicitudes de adopción.

```
petsadmin [ petsdb ]
# describe adoption_statuses
```

Field	Type	Null	Key	Default	Extra
id	char(1)	NO	PRI	<null>	
description	varchar(20)	NO		<null>	

```
petsadmin [ petsdb ]
# select * from adoption_statuses;
```

id	description
1	Available
2	In process
3	Adopted
4	Approved
5	Rejected

- Luego fue creada la tabla pets que almacenará registros de mascotas.

```
petsadmin [ petsdb ]
# describe pets
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	<null>	auto_increment
type	char(1)	NO	MUL	<null>	
name	varchar(50)	NO		<null>	
age	int(11)	YES		<null>	
adoption_status	char(1)	NO	MUL	1	
createdAt	timestamp	NO		current_timestamp()	
updatedAt	timestamp	NO		current_timestamp()	on update current_timestamp()

- También la tabla adopters para aquellas personas que hacen la solicitud de adopción.

```
petsadmin [ petsdb ]
# describe adopters
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	<null>	auto_increment
name	varchar(50)	NO		<null>	
address	varchar(100)	YES		<null>	
contact	varchar(20)	YES		<null>	
createdAt	timestamp	NO		current_timestamp()	
updatedAt	timestamp	NO		current_timestamp()	on update current_timestamp()

- Por último la tabla adoption\_requests que contendrá información de las solicitudes de adopción.

```
petsadmin [ petsdb ]
# describe adoption_requests
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	<null>	auto_increment
pet_id	int(11)	NO	MUL	<null>	
adopter_id	int(11)	NO	MUL	<null>	
status	char(1)	NO	MUL	2	
createdAt	timestamp	NO		current_timestamp()	
updatedAt	timestamp	NO		current_timestamp()	on update current_timestamp()

Para el desarrollo de la aplicación Backend se hace uso de la base de datos y permite realizar el registro y administración de mascotas dadas en adopción por la empresa **MeowMates Adoption Center**.

- Se creó un archivo de configuración de base de datos para inicializar un objeto de Sequelize el cual permite crear modelos para conectar sus atributos con el de las tablas de la base de datos. Importa las variables de entorno definidas con anterioridad.

```

1 DB_HOST=localhost
2 DB_USER=petsadmin
3 DB_PASSWD=petspasswd
4 DB_NAME=petsdb
```

```

const db = new Sequelize({
  host: process.env.DB_HOST,
  username: process.env.DB_USER,
  password: process.env.DB_PASSWD,
  database: process.env.DB_NAME,
  dialect: 'mysql',
  dialectModule: mysql2,
})
```

## Modelos

- Los modelos de objetos fueron creados de tal forma que sus propiedades se asemejen a las de las tablas de la base de datos.

```
export const Pet = db.define('pet', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    allowNull: false,
    autoIncrement: true
  },
  type: {
    type: DataTypes.CHAR,
    allowNull: false
  },
  name: {
    type: DataTypes.STRING,
    allowNull: false
  },
  age: {
    type: DataTypes.INTEGER
  },
  adoption_status: {
    type: DataTypes.CHAR,
    allowNull: false,
    defaultValue: '1'
  }
})

export const petTypes = { 'C': 'Cat', 'D': 'Dog' }

export const getPet = id => Pet.findByPk(id)

export const formatPet = pet => ({
  ...pet.dataValues,
  type: petTypes[pet.type],
  adoption_status: adoptionStatuses[pet.adoption_status]
})

export const AdoptionRequest = db.define('adoption_request', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    allowNull: false,
    autoIncrement: true
  },
  pet_id: {
    type: DataTypes.INTEGER,
    allowNull: false
  },
  adopter_id: {
    type: DataTypes.INTEGER,
    allowNull: false
  },
  status: {
    type: DataTypes.CHAR,
    defaultValue: '2',
    allowNull: false
  }
})

export const Adopter = db.define('adopter', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    allowNull: false,
    autoIncrement: true
  },
  name: {
    type: DataTypes.TEXT,
    allowNull: false
  },
  address: {
    type: DataTypes.TEXT,
  },
  contact: {
    type: DataTypes.TEXT,
  }
})
```

- Dentro de los archivos de cada modelo se definieron diccionarios y funciones que permiten un mejor manejo y visualización de los datos de los registros.

```
export const getAdopter = id => Adopter.findByPk(id)
```

```

export const adoptionStatuses = {
  '1': 'Available',
  '2': 'In process',
  '3': 'Adopted',
  '4': 'Approved',
  '5': 'Rejected'
}

export const getAdoptionRequest = id => AdoptionRequest.findByPk(id)

export const formatRequest = request => ({
  ...request.dataValues,
  status: adoptionStatuses[request.status]
})

```

## Controladores

Para manejar las peticiones de la api, se definió el tratamiento de estas mediante controladores, las funciones siempre devuelven objetos JSON que muestran el resultado de la petición o un mensaje de error.

## Mascotas

- La siguiente función permite mostrar todas las mascotas registradas.

```

export const readPets = async (req, res) => {
  let pets = await Pet.findAll()
  pets = pets.map(pet => formatPet(pet))

  const length = pets.length
  if (length == 0)
    return res.status(500).jsonPretty({ type: 'error', message: 'There is no pets', length })
  res.status(200).jsonPretty({ pets, length })
}

```

- Mostrar una sola mascota en específico mediante su id.

```

export const readPet = async (req, res) => {
  const id = req.params.id
  const pet = await getPet(id)

  if (pet == null)
    return res.status(404).jsonPretty({ type: 'error', message: `Pet [${id}] not found` })
  res.status(200).jsonPretty({ pet: formatPet(pet) })
}

```

- Registrar una nueva mascota, con un tipo, un nombre y una edad, esta última no es obligatoria y el estado de adopción al momento de ser registrada es por defecto: 'Disponible'.

```
export const createPet = async (req, res) => {
  if (req.body.type == undefined)
    return res.status(400).jsonPretty({ type: 'error', message: 'The pet type is required' })
  if (req.body.name == undefined)
    return res.status(400).jsonPretty({ type: 'error', message: 'The pet name is required' })
  if (petTypes[req.body.type] == undefined)
    return res.status(400).jsonPretty({ type: 'error', message: 'The pet type is not valid' })

  const pet = formatPet(await Pet.create(req.body))
  res.status(201).jsonPretty({
    type: 'success',
    message: `${pet.type} [${pet.id}] created successfully`,
    pet
  })
}
```

- Actualizar la información de una mascota identificada con un id, aquí no es posible modificar el estado de adopción.

```
export const updatePet = async (req, res) => {
  const id = req.params.id
  const { adoption_status, ...body } = req.body

  if (!body.type && !body.name && !body.age)
    return res.status(400).jsonPretty({ type: 'error', message: 'There is not valid data for update' })
  if (body.type && petTypes[body.type] == undefined)
    return res.status(400).jsonPretty({ type: 'error', message: 'The pet type is not valid' })

  let pet = await getPet(id)
  if (pet == null)
    return res.status(404).jsonPretty({ type: 'error', message: `Pet [${id}] does not exists` })

  pet.update(body, { where: { id } })
  res.status(200).jsonPretty({
    type: 'success',
    message: `${petTypes[pet.type]} [${id}] updated successfully`,
    pet: formatPet(pet)
  })
}
```

- Eliminar el registro de una mascota identificada por un id de la base de datos.

```
export const removePet = async (req, res) => {
  const id = req.params.id
  let pet = await getPet(id)

  if (pet == null)
    return res.status(404).jsonPretty({ type: 'error', message: `Pet [${id}] does not exists` })

  pet.destroy()
  res.status(200).jsonPretty({
    type: 'success',
    message: `${petTypes[pet.type]} [${id}] deleted successfully`,
    pet: formatPet(pet)
  })
}
```

## Adoptadores

- Función que permite registrar una nueva persona que realiza la solicitud de adopción ingresando un nombre, una dirección y el teléfono de contacto, solo el nombre es obligatorio.

```
export const createAdopter = async (req, res) => {
  if (req.body.name == undefined)
    return res.status(400).jsonPretty({ type: 'error', message: 'The adopter name is required' })

  const adopter = await Adopter.create(req.body)
  res.status(201).jsonPretty({
    type: 'success',
    message: `Adopter [${adopter.id}] created successfully`,
    adopter
  })
}
```

- Actualizar el registro de un adoptador.

```
export const updateAdopter = async (req, res) => {
  const id = req.params.id

  if (!req.body.name && !req.body.address && !req.body.contact)
    return res.status(400).jsonPretty({ error: 'There is not valid data for update' })

  const adopter = await getAdopter(id)
  if (adopter == null)
    return res.status(404).jsonPretty({ type: 'error', message: `Adopter [${id}] does not exists` })

  adopter.update(req.body)
  res.status(200).jsonPretty({
    type: 'success',
    message: `Adopter [${id}] updated successfully`,
    adopter
  })
}
```

- Mostrar todos los adoptadores registrados.

```
export const readAdopters = async (req, res) => {
  const adopters = await Adopter.findAll()
  const length = adopters.length

  if (length == 0)
    return res.status(500).jsonPretty({ type: 'error', message: 'There is no adopters', length })
  res.status(200).jsonPretty({ adopters, length })
}
```

- Mostrar un adoptador seleccionado identificado por un id.

```
export const readAdopter = async (req, res) => {
  const id = req.params.id
  const adopter = await getAdopter(id)

  if (adopter == null)
    return res.status(404).jsonPretty({ type: 'error', message: `Adopter [${id}] not found` })
  res.status(200).jsonPretty({ adopter })
}
```

- Eliminar el registro de un adoptador identificado por un id.

```
export const deleteAdopter = async (req, res) => {
  const id = req.params.id
  const adopter = await getAdopter(id)

  if (adopter == null)
    return res.status(404).jsonPretty({ type: 'error', message: `Adopter [${id}] does not exists` })

  adopter.destroy()
  res.status(200).jsonPretty({
    type: 'success',
    message: `Adopter [${id}] deleted successfully`,
    adopter
  })
}
```

## Solicitudes de Adopción

- Registrar una nueva solicitud de adopción, la mascota a adoptar y el adoptador deben existir, El estado de adopción de la solicitud es por defecto 'En proceso', y el estado de la mascota a adoptar también.

```
export const createRequest = async (req, res) => {
  const petId = req.body.pet_id
  const adopterId = req.body.adopter_id
  const pet = await getPet(petId)
  const adopter = await getAdopter(adopterId)

  if (petId == undefined)
    return res.status(400).jsonPretty({ type: 'error', message: 'The pet id is required' })
  if (adopterId == undefined)
    return res.status(400).jsonPretty({ type: 'error', message: 'The adopter id is required' })
  if (pet == null)
    return res.status(404).jsonPretty({ type: 'error', message: `Pet [${id}] not found` })
  if (adopter == null)
    return res.status(404).jsonPretty({ type: 'error', message: `Adopter [${id}] not found` })

  const request = await AdoptionRequest.create(req.body)
  pet.update({ adoption_status: '2' })

  res.status(201).jsonPretty({
    type: 'success',
    message: `Adoption request [${request.id}] created successfully`,
    adoptionRequest: formatRequest(request)
  })
}
```



- Mostrar todas las solicitudes de adopción.

```
export const readRequests = async (req, res) => {
  let requests = await AdoptionRequest.findAll()
  const length = requests.length
  requests = requests.map(request => formatRequest(request))

  if (length == 0)
    return res.status(500).jsonPretty({
      type: 'error', message: 'There is no adoption requests', length
    })
  res.status(200).jsonPretty({ adoptionRequests: requests, length })
}
```

- Mostrar solo una de las solicitudes de adopción identificada por id.

```
export const readRequest = async (req, res) => {
  const id = req.params.id
  const request = await getRequest(id)

  if (request == null)
    return res.status(404).jsonPretty({
      type: 'error', message: `Adoption request [${id}] not found`
    })
  res.status(200).jsonPretty({ adoptionRequest: formatRequest(request) })
}
```

- Aprobar una solicitud de adopción identificada con id, al ser aprobada el estado de la solicitud cambia a: 'Aprobada' y el estado de la mascota cambia a: 'Adoptada'.

```
export const approveRequest = async (req, res) => {
  const id = req.params.id

  const request = await getRequest(id)
  if (request == null)
    return res.status(404).jsonPretty({
      type: 'error', message: `Adoption request [${id}] does not exists`
    })

  const pet = await getPet(request.pet_id)
  pet.update({ adoption_status: '3' })
  request.update({ status: '4' })

  res.status(200).jsonPretty({
    type: 'success',
    message: `Adoption request [${id}] approved successfully`,
    adoptionRequest: formatRequest(request)
  })
}
```

- Rechazar solicitud de adopción identificada por un id, al rechazar la solicitud su estado cambia a: 'Rechazada' y el estado de la mascota cambia a: 'Disponible'.

```
export const rejectRequest = async (req, res) => {
  const id = req.params.id

  const request = await getRequest(id)
  if (request == null)
    return res.status(404).jsonPretty({
      type: 'error', message: `Adoption request [${id}] does not exists`
    })

  const pet = await getPet(request.pet_id)
  pet.update({ adoption_status: '1' })
  request.update({ status: '5' })

  res.status(200).jsonPretty({
    type: 'success',
    message: `Adoption request [${id}] rejected successfully`,
    adoptionRequest: formatRequest(request)
  })
}
```

- Actualizar una solicitud de adopción identificada por un id, aquí no es posible actualizar el estado de la solicitud.

```
export const updateRequest = async (req, res) => {
  const id = req.params.id
  const { status, ...body } = req.body

  if (!body.pet_id && !body.adopter_id)
    return res.status(400).jsonPretty({
      type: 'error', message: 'There is not valid data for update'
    })

  const request = await getRequest(id)
  if (request == null)
    return res.status(404).jsonPretty({
      type: 'error', message: `Adoption request [${id}] does not exists`
    })

  request.update(body)
  res.status(200).jsonPretty({
    type: 'success',
    message: `Adoption request [${id}] updated successfully`,
    adoptionRequest: formatRequest(request)
  })
}
```

- Eliminar solicitud de adopción identificada por un id.

```
export const deleteRequest = async (req, res) => {
  const id = req.params.id
  const request = await getRequest(id)

  if (request == null)
    return res.status(404).jsonPretty({
      type: 'error', message: `Adoption request [${id}] does not exists`
    })

  request.destroy()
  res.status(200).jsonPretty({
    type: 'success',
    message: `Adoption request [${id}] deleted successfully`,
    adoptionRequest: formatRequest(request)
  })
}
```

## Rutas

Se definen archivos de router para los 3 tipos de datos de la base de datos.

- Rutas de las mascotas.

```
import * as controller from '../controllers/pets-controller.js'

const router = express.Router()

router.get('/', controller.readPets)
router.get('/:id', controller.readPet)
router.post('/create', controller.createPet)
router.put('/update/:id', controller.updatePet)
router.delete('/delete/:id', controller.removePet)

export { router as petsRouter }
```

- Rutas de los adoptadores.

```
import * as controller from "../controllers/adopters-controller.js"

const router = express.Router()

router.get('/', controller.readAdopters)
router.get('/:id', controller.readAdopter)
router.post('/create', controller.createAdopter)
router.put('/update/:id', controller.updateAdopter)
router.delete('/delete/:id', controller.deleteAdopter)

export { router as adopterRouter }
```

- Rutas de las solicitudes de adopción.

```
import * as controller from "../controllers/adoption-requests-controller.js"

const router = express.Router()

router.get('/', controller.readRequests)
router.get('/:id', controller.readRequest)
router.post('/create', controller.createRequest)
router.put('/update/:id', controller.updateRequest)
router.put('/approve/:id', controller.approveRequest)
router.put('/reject/:id', controller.rejectRequest)
router.delete('/delete/:id', controller.deleteRequest)

export { router as requestRouter }
```

## App

Al final se usa un archivo app.js con el fin de definir el servidor de expressjs y asignarle las rutas definidas anteriormente con el fin de ser ejecutada en el puerto 8000.

- Se usa express.json() para que el servidor acepte solicitudes de tipo POST y PUT con un objeto JSON.
- Se usan las funciones authenticate() y sync() para verificar la conexión con la base de datos.
- También un cors() en caso de que la API sea usada correctamente por una aplicación frontend.
- Por último se agrega una redirección a la documentación de la API en el repositorio de GitHub.

```
const app = express()
const PORT = 8000

db.authenticate().then(() => {
  console.log('Database connected successfully')
}).catch(err => {
  console.error(`Error at database connection: ${err}`)
})

app.use(express.json())
app.use(cors())
app.use('/pets', petsRouter)
app.use('/adopters', adopterRouter)
app.use('/adoptionrequests', requestRouter)

app.get('/', (req, res) => res.status(301).redirect(
  'https://github.com/alejo-c/pets-api?tab=readme-ov-file#api-usage'
))

db.sync().then(() => {
  app.listen(PORT, () => console.log(`Server started at http://localhost:\${PORT}`)
}).catch(err => {
  console.error(`Error at database sync: ${err}`);
})
```

## Uso mediante cliente gráfico (Thunder Client)

### Mascotas

- Mostrar todas las mascotas.



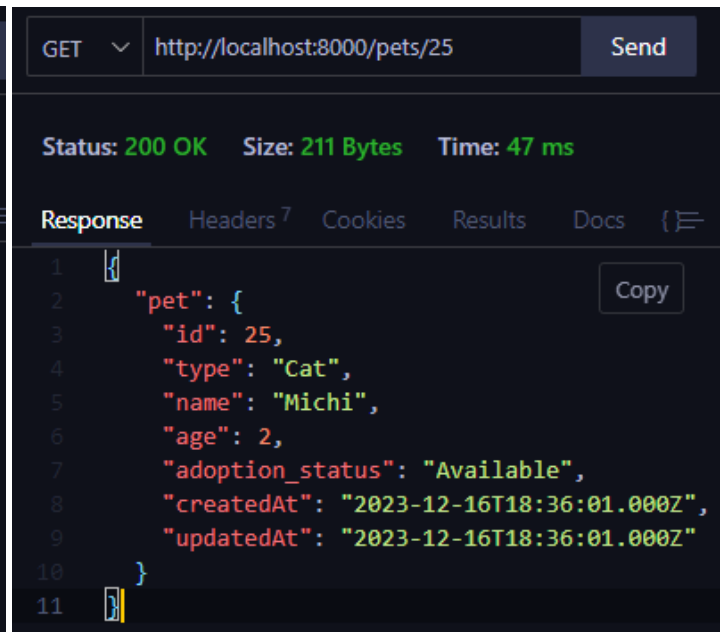
GET  Send

Status: 200 OK Size: 4.1 KB Time: 21 ms

Response Headers 7 Cookies Results Docs {}

```
144  "createdAt": "2023-12-11T17:12:56.000Z",
145  "updatedAt": "2023-12-11T17:12:56.000Z"
146  },
147  {
148    "id": 17,
149    "type": "Cat",
150    "name": "Tucker",
151    "age": 1,
152    "adoption_status": "Available",
153    "createdAt": "2023-12-11T17:12:49.000Z",
154    "updatedAt": "2023-12-11T17:12:49.000Z"
155  },
156  {
157    "id": 24,
158    "type": "Dog",
159    "name": "Luna",
160    "age": 1,
161    "adoption_status": "Available",
162    "createdAt": "2023-12-16T18:26:03.000Z",
163    "updatedAt": "2023-12-16T18:26:03.000Z"
164  },
165  {
166    "id": 25,
167    "type": "Cat",
168    "name": "Michi",
169    "age": 2,
170    "adoption_status": "Available",
171    "createdAt": "2023-12-16T18:36:01.000Z",
172    "updatedAt": "2023-12-16T18:36:01.000Z"
173  }
174  ],
175  "length": 19
176  }
```

- Mostrar una sola mascota.



GET  Send

Status: 200 OK Size: 211 Bytes Time: 47 ms

Response Headers 7 Cookies Results Docs {}

```
1  {
2    "pet": {
3      "id": 25,
4      "type": "Cat",
5      "name": "Michi",
6      "age": 2,
7      "adoption_status": "Available",
8      "createdAt": "2023-12-16T18:36:01.000Z",
9      "updatedAt": "2023-12-16T18:36:01.000Z"
10   }
11 }
```

- Registrar una nueva mascota .

POST  http://localhost:8000/pets/create

Query Headers<sup>3</sup> Auth **Body<sup>1</sup>** Tests Pre Run

JSON XML Text Form Form-encode GraphQL

JSON Content Format

```
1 {
2   "type": "D",
3   "name": "Buddy",
4   "age": 2
5 }
```

Status: 201 Created Size: 278 Bytes Time: 174 ms

Response Headers<sup>7</sup> Cookies Results Docs {} ≡

```
1 {
2   "type": "success",
3   "message": "Dog [26] created successfully",
4   "pet": {
5     "adoption_status": "Available",
6     "id": 26,
7     "type": "Dog",
8     "name": "Buddy",
9     "age": 2,
10    "updatedAt": "2023-12-18T01:45:20.161Z",
11    "createdAt": "2023-12-18T01:45:20.161Z"
12  }
13 }
```

- Editar el registro de una mascota.

PUT  http://localhost:8000/pets/update/26

Query Headers<sup>2</sup> Auth **Body<sup>1</sup>** Tests Pre Run

JSON XML Text Form Form-encode GraphQL

JSON Content Format

```
1 {
2   "name": "Bobo",
3   "type": "C"
4 }
```

Status: 200 OK Size: 277 Bytes Time: 26 ms

Response Headers<sup>7</sup> Cookies Results Docs {} ≡

```
1 {
2   "type": "success",
3   "message": "Cat [26] updated successfully",
4   "pet": {
5     "id": 26,
6     "type": "Cat",
7     "name": "Bobo",
8     "age": 2,
9     "adoption_status": "Available",
10    "createdAt": "2023-12-18T01:45:20.000Z",
11    "updatedAt": "2023-12-18T01:46:35.609Z"
12  }
13 }
```

- Eliminar el registro de una mascota.

DELETE  http://localhost:8000/pets/delete/26

Status: 200 OK Size: 277 Bytes Time: 19 ms

Response Headers<sup>7</sup> Cookies Results Docs {} ≡

```
1 {
2   "type": "success",
3   "message": "Cat [26] deleted successfully",
4   "pet": {
5     "id": 26,
6     "type": "Cat",
7     "name": "Bobo",
8     "age": 2,
9     "adoption_status": "Available",
10    "createdAt": "2023-12-18T01:45:20.000Z",
11    "updatedAt": "2023-12-18T01:46:35.000Z"
12  }
13 }
```

## Adoptadores

- Mostrar todos los adoptadores.

```
GET http://localhost:8000/adopters Send

Status: 200 OK Size: 2.37 KB Time: 28 ms

Response Headers 7 Cookies Results Docs {}

56 "createdAt": "2023-12-15T21:27:28.000Z",
57 "updatedAt": "2023-12-15T21:27:28.000Z"
58 },
59 {
60 "id": 8,
61 "name": "Camila Perez",
62 "address": "505 Birch Street, Town, Country",
63 "contact": "+987 654 3210",
64 "createdAt": "2023-12-15T21:27:28.000Z",
65 "updatedAt": "2023-12-15T21:27:28.000Z"
66 },
67 {
68 "id": 9,
69 "name": "Nicolas Castro",
70 "address": "606 Willow Lane, Village, Country",
71 "contact": "+123 456 7890",
72 "createdAt": "2023-12-15T21:27:28.000Z",
73 "updatedAt": "2023-12-15T21:27:28.000Z"
74 },
75 {
76 "id": 10,
77 "name": "Mariana Flores",
78 "address": "707 Redwood Road, City, Country",
79 "contact": "+210 678 5432",
80 "createdAt": "2023-12-15T21:27:28.000Z",
81 "updatedAt": "2023-12-15T21:27:28.000Z"
82 }
83 ],
84 "length": 10
85 }
```

- Mostrar un solo adoptador.

```
GET http://localhost:8000/adopters/1 Send

Status: 200 OK Size: 240 Bytes Time: 5 ms

Response Headers 7 Cookies Results Docs {}

1 {
2   "adopter": {
3     "id": 1,
4     "name": "Alejandro Rodriguez",
5     "address": "123 Main Street, City, Country",
6     "contact": "+123 456 7890",
7     "createdAt": "2023-12-15T21:27:28.000Z",
8     "updatedAt": "2023-12-15T21:27:28.000Z"
9   }
10 }
```

- Registrar un nuevo adoptador.

```
POST http://localhost:8000/adopters/create Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

1 {
2   "name": "Alejandro Castrillón",
3   "address": "789 Pine Street, City, Country'",
4   "contact": "+345 678 9012"
5 }

Status: 201 Created Size: 315 Bytes Time: 92 ms

Response Headers 7 Cookies Results Docs {}

1 {
2   "type": "success",
3   "message": "Adopter [11] created successfully",
4   "adopter": {
5     "id": 11,
6     "name": "Alejandro Castrillón",
7     "address": "789 Pine Street, City, Country'",
8     "contact": "+345 678 9012",
9     "updatedAt": "2023-12-18T01:52:59.119Z",
10    "createdAt": "2023-12-18T01:52:59.119Z"
11  }
12 }
```

- Editar el registro de un adoptador.

PUT ▼ http://localhost:8000/adopters/update/11 Send

Query Headers <sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "name": "Jose Ortega"
3 }
```

Status: 200 OK Size: 305 Bytes Time: 20 ms

Response Headers <sup>7</sup> Cookies Results Docs {} ≡

```
1 {
2   "type": "success",
3   "message": "Adopter [11] updated successfully",
4   "adopter": {
5     "id": 11,
6     "name": "Jose Ortega",
7     "address": "789 Pine Street, City, Country'",
8     "contact": "+345 678 9012",
9     "createdAt": "2023-12-18T01:52:59.000Z",
10    "updatedAt": "2023-12-18T01:54:22.678Z"
11  }
12 }
```

- Eliminar el registro de un adoptador.

DELETE ▼ http://localhost:8000/adopters/delete/11 Send

Status: 200 OK Size: 305 Bytes Time: 31 ms

Response Headers <sup>7</sup> Cookies Results Docs {} ≡

```
1 {
2   "type": "success",
3   "message": "Adopter [11] deleted successfully",
4   "adopter": {
5     "id": 11,
6     "name": "Jose Ortega",
7     "address": "789 Pine Street, City, Country'",
8     "contact": "+345 678 9012",
9     "createdAt": "2023-12-18T01:52:59.000Z",
10    "updatedAt": "2023-12-18T01:54:22.000Z"
11  }
12 }
```

## Solicitudes de Adopción

- Registrar una nueva mascota.

POST ▼ http://localhost:8000/adoptionrequests/create Send

Query Headers <sup>3</sup> Auth Body<sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "pet_id": "10",
3   "adopter_id": "9"
4 }
```

Status: 201 Created Size: 284 Bytes Time: 47 ms

Response Headers <sup>7</sup> Cookies Results Docs

```
1 {
2   "type": "success",
3   "message": "Adoption request [11] created successfully",
4   "adoptionRequest": {
5     "status": "In process",
6     "id": 11,
7     "pet_id": "10",
8     "adopter_id": "9",
9     "updatedAt": "2023-12-18T01:56:12.488Z",
10    "createdAt": "2023-12-18T01:56:12.488Z"
11  }
12 }
```



- Mostrar todas las solicitudes de adopción.

```
GET http://localhost:8000/adoptionrequests/ Send

Status: 200 OK Size: 2.11 KB Time: 16 ms

Response Headers 7 Cookies Results Docs {} ≡

64 "createdAt": "2023-12-15T21:35:54.000Z",
65 "updatedAt": "2023-12-15T22:57:16.000Z"
66 },
67 {
68   "id": 9,
69   "pet_id": 9,
70   "adopter_id": 9,
71   "status": "Rejected",
72   "createdAt": "2023-12-15T21:35:54.000Z",
73   "updatedAt": "2023-12-15T22:56:56.000Z"
74 },
75 {
76   "id": 10,
77   "pet_id": 10,
78   "adopter_id": 10,
79   "status": "In process",
80   "createdAt": "2023-12-15T21:35:54.000Z",
81   "updatedAt": "2023-12-15T21:35:54.000Z"
82 },
83 {
84   "id": 11,
85   "pet_id": 10,
86   "adopter_id": 9,
87   "status": "In process",
88   "createdAt": "2023-12-18T01:56:12.000Z",
89   "updatedAt": "2023-12-18T01:56:12.000Z"
90 }
91 ],
92 "length": 11
93 }
```

- Mostrar una sola solicitud de adopción.

```
GET http://localhost:8000/adoptionrequests/11 Send

Status: 200 OK Size: 200 Bytes Time: 13 ms

Response Headers 7 Cookies Results Docs {} ≡

1 {
2   "adoptionRequest": {
3     "id": 11,
4     "pet_id": 10,
5     "adopter_id": 9,
6     "status": "In process",
7     "createdAt": "2023-12-18T01:56:12.000Z",
8     "updatedAt": "2023-12-18T01:56:12.000Z"
9   }
10 }
```

- Editar una solicitud de adopción.

PUT http://localhost:8000/adoptionrequests/update/11 Send

Query Headers<sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "pet_id": "9",
3   "adopter_id": "10"
4 }
```

Status: 200 OK Size: 284 Bytes Time: 27 ms

Response Headers<sup>7</sup> Cookies Results Docs {} ≡

```
1 {
2   "type": "success",
3   "message": "Adoption request [11] updated successfully",
4   "adoptionRequest": {
5     "id": 11,
6     "pet_id": "9",
7     "adopter_id": "10",
8     "status": "In process",
9     "createdAt": "2023-12-18T01:56:12.000Z",
10    "updatedAt": "2023-12-18T03:00:09.549Z"
11  }
12 }
```

- Eliminar una solicitud de adopción.

DELETE http://localhost:8000/adoptionrequests/delete/11 Send

Status: 200 OK Size: 278 Bytes Time: 17 ms

Response Headers<sup>7</sup> Cookies Results Docs {} ≡

```
1 {
2   "type": "success",
3   "message": "Adoption request [11] deleted successfully",
4   "adoptionRequest": {
5     "id": 11,
6     "pet_id": 9,
7     "adopter_id": 10,
8     "status": "Approved",
9     "createdAt": "2023-12-18T01:56:12.000Z",
10    "updatedAt": "2023-12-18T03:03:41.000Z"
11  }
12 }
```

- Aprobar una solicitud de adopción.

PUT http://localhost:8000/adoptionrequests/approve/11 Send

Status: 200 OK Size: 279 Bytes Time: 29 ms

Response Headers<sup>7</sup> Cookies Results Docs {} ≡

```
1 {
2   "type": "success",
3   "message": "Adoption request [11] approved successfully",
4   "adoptionRequest": {
5     "id": 11,
6     "pet_id": 9,
7     "adopter_id": 10,
8     "status": "Approved",
9     "createdAt": "2023-12-18T01:56:12.000Z",
10    "updatedAt": "2023-12-18T03:03:41.830Z"
11  }
12 }
```

- Rechazar una solicitud de adopción.

PUT http://localhost:8000/adoptionrequests/reject/10 Send

Status: 200 OK Size: 280 Bytes Time: 31 ms

Response Headers<sup>7</sup> Cookies Results Docs {} ≡

```
1 {
2   "type": "success",
3   "message": "Adoption request [10] rejected successfully",
4   "adoptionRequest": {
5     "id": 10,
6     "pet_id": 10,
7     "adopter_id": 10,
8     "status": "Rejected",
9     "createdAt": "2023-12-15T21:35:54.000Z",
10    "updatedAt": "2023-12-18T03:05:06.080Z"
11  }
12 }
```