

Universidad de Nariño.

Ingeniería de Sistemas.

Diplomado de actualización en nuevas tecnologías para el desarrollo de Software.

Fredilton Daniel Getial Torres

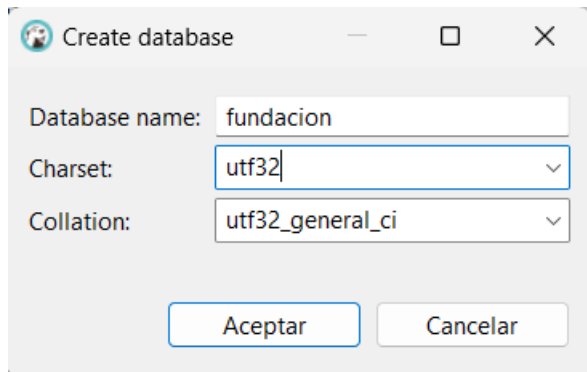
Código: 219036091

Teléfono: 3187863224

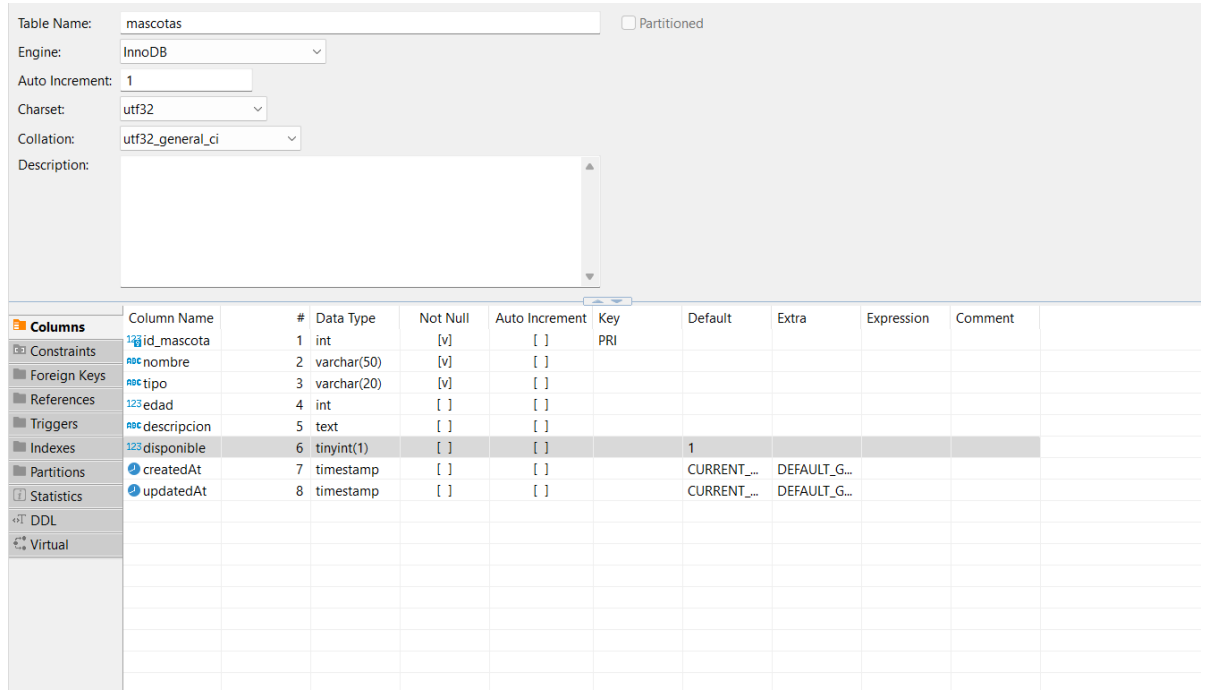
Taller Unidad 2 Backend

1. Crear una base de datos MYSQL que permita llevar el registro de mascotas (perros y gatos), así como también el proceso de solicitud de adopción de estas.

Creación de base de datos



Creación de tabla mascotas



	Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
	id_mascota	1	int	[v]	[]	PRI				
	nombre	2	varchar(50)	[v]	[]					
	tipo	3	varchar(20)	[v]	[]					
	edad	4	int	[]	[]					
	descripcion	5	text	[]	[]					
	disponible	6	tinyint(1)	[]	[]		1			
	createdAt	7	timestamp	[]	[]		CURRENT_...	DEFAULT_G...		
	updatedAt	8	timestamp	[]	[]		CURRENT_...	DEFAULT_G...		

Creación de tablas solicitudes

Propiedades Datos Diagrama ER localhost Databases fundacion

Table Name: solicitudes ☐ Partitioned

Engine: InnoDB

Auto Increment: 1

Charset: utf32

Collation: utf32_general_ci

Description:

	Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
Columns	id_Solicitud	1	int	[v]	[]	PRI				
Constraints	id_mascota	2	int	[]	[]	MUL				
Foreign Keys	nombreSol...	3	varchar(50)	[v]	[]					
References	correoSol...	4	varchar(50)	[v]	[]					
Triggers	estadoSol...	5	varchar(20)	[]	[]		'Pendiente'			
Indexes										
Partitions										
Statistics										
DDL										
Virtual										

Creación de usuario

Login

User Name: fundacion

Host: localhost

Password:

Confirm:

Limits

Max Queries: 0

Max Updates: 0

Max Connections: 0

Max User Connections: 0

DBA Privileges

Privilege	Enabled	Description
Create role	[]	To create new roles
Create user	[]	To create new users
Drop role	[]	To drop roles
Event	[]	To create, alter, drop and execute events
File	[]	To read and write files on the server
Process	[]	To view the plain text of currently executing queries
Reload	[]	To reload or refresh tables, logs and privileges
Replication client	[]	To ask where the slave or master servers are
Replication slave	[]	To read binary log events from the master
Show databases	[]	To see all databases with SHOW DATABASES
Shutdown	[]	To shut down the server
Super	[]	To use KILL thread, SET GLOBAL, CHANGE MASTER, ...
Create tablespace	[]	To create/alter/drop tablespaces
Usage	[v]	No privileges - allow connect only

2. Desarrollar una aplicación Backend implementada en NodeJS y ExpressJS que haga uso de la base de datos del primer punto y que permita el desarrollo de todas las tareas asociadas al registro y administración de las mascotas dadas en adopción por la empresa (La empresa debe contar con un nombre).

Se debe hacer uso correcto de los verbos HTTP dependiendo de la tarea a realizar.

conexión con la base de datos

```
import Sequelize from "sequelize";

const db = new Sequelize("fundacion","fundacion","fundacion123",{
  dialect: "mysql",
  host: "localhost"
});

export {db}
```

Configuración archivo app

```
import express from "express";
import { routerMascotas } from "../rutas/mascotasRouter.js";
import { routerSolicitudes } from "../rutas/solicitudesRouter.js";
import {db} from "../database/conexion.js";

//Crear Instancia de Express
const app = express();

//Middleware
app.use(express.json());

//Verificar Conexion a Base de Datos
db.authenticate().then(()=>{
  console.log(`Base de Datos conectada de manera exitosa`);
}).catch(err=>{
  console.log(`Error al conectarse a la Base de Datos ::: ${err}`);
})

//Definir Rutas
app.get("/",(req,res)=>{
  res.send("Hola Backend Mysql");
});

//Rutas
app.use("/mascotas",routerMascotas);
app.use("/solicitudes",routerSolicitudes);

//Puerto de Servidor
const PORT=8000;

//Verificar que pueda sincronizar con la base de datos
db.sync().then(()=>{
  app.listen(PORT,()=>{
    console.log(`Servidor Inicializado en puerto ${PORT}`);
  });
}).catch(err=>{
  console.log(`Error al sincronizar Base de Datos ${err}`);
});
```

Para tabla mascotas

Crear modelo Mascotas

```
import Sequelize from "sequelize";
import {db} from "../database/conexion.js";

const mascotas = db.define("mascotas",{
  id_mascota: {
    type: Sequelize.INTEGER,
    allowNull: false,
    primaryKey: true
  },
  nombre: {
    type: Sequelize.STRING(50),
    allowNull: false
  },
  tipo: {
    type: Sequelize.STRING(20),
    allowNull: false
  },
  edad: {
    type: Sequelize.INTEGER,
    allowNull: true
  },
  descripcion: {
    type: Sequelize.TEXT,
    allowNull: true
  },
  disponible: {
    type: Sequelize.BOOLEAN,
    defaultValue: true
  }
});

export {mascotas}
```

Crear controlador Mascotas

```
controladores > JS mascotasController.js > [0] buscarPerros
1 import {mascotas} from "../modelos/mascotasModelo.js";
2 import {solicitudes} from "../modelos/solicitudesModelo.js";
3
```

Función registrar mascotas

```
// Crear un recurso
const crearMascota = (req, res) => {
  // Verificar si el nombre, tipo, edad y id_mascota están presentes y no son nulos
  if (!req.body.id_mascota || !req.body.nombre || !req.body.tipo ) {
    return res.status(400).json({
      mensaje: "id_mascota, nombre, tipo y edad son campos requeridos y no pueden ser nulos."
    });
  }

  // Validar que el tipo sea 'gato' o 'perro'
  const tiposValidos = ['gato', 'perro'];
  if (!tiposValidos.includes(req.body.tipo)) {
    return res.status(400).json({
      mensaje: "El campo 'tipo' debe ser 'gato' o 'perro'."
    });
  }

  // Crear un objeto dataset con los campos relevantes
  const dataset = {
    id_mascota: req.body.id_mascota,
    nombre: req.body.nombre,
    tipo: req.body.tipo,
    edad: req.body.edad,
    descripcion: req.body.descripcion,
    disponible: req.body.disponible,
  };

  // Utilizar Sequelize para crear el recurso
  mascotas.create(dataset)
    .then((resultado) => {
      res.status(200).json({
        mensaje: "Registro creado correctamente",
        resultado: resultado
      });
    })
    .catch((err) => {
      res.status(500).json({
        mensaje: `Error al crear el registro: ${err.message.errors}`
      });
    });
};
```

Función buscar por id Mascotas

```
//Buscar recurso por ID
const buscarIdMascotas = (req,res)=>{
  const id = req.params.id;
  if(id == null){
    res.status(203).json({
      mensaje: `El id no puede estar vacio`
    });
    return;
  }

  mascotas.findByIdPk(id).then((resultado)=>{
    res.status(200).json(resultado);
  }).catch((err)=>{
    res.status(500).json({
      mensaje: `Registro no encontrado ::: ${err}`
    });
  });
}
```

Función buscar todas las mascotas

```
const buscarMascotas = (_req, res)=>{

  mascotas.findAll().then((resultado)=>{
    res.status(200).json(resultado);
  }).catch((err)=>{
    res.status(500).json({
      mensaje: `No se encontraron Registros ::: ${err}`
    });
  });
};
```

Función buscar mascotas disponibles

```
const buscarMascotasDiponibles = (_req,res)=>{

  mascotas.findAll({ where: { disponible: true }}).then((resultado)=>{
    res.status(200).json(resultado);
  }).catch((err)=>{
    res.status(500).json({
      mensaje: `No se encontraron Registros ::: ${err}`
    });
  });
}
```

Función buscar gatos

```
const buscarGatos = (req,res)=>{  
  mascotas.findAll({ where: { tipo: "gato" } }).then((resultado)=>{  
    res.status(200).json(resultado);  
  }).catch((err)=>{  
    res.status(500).json({  
      mensaje: `No se encontraron Registros ::: ${err}`  
    });  
  });  
}
```

Función buscar perros

```
const buscarPerros = (req,res)=>{  
  mascotas.findAll({ where: { tipo: "perro" } }).then((resultado)=>{  
    res.status(200).json(resultado);  
  }).catch((err)=>{  
    res.status(500).json({  
      mensaje: `No se encontraron Registros ::: ${err}`  
    });  
  });  
}
```

Función actualizar mascotas

```
const actualizarMascota = (req, res) => {
  const id_mascota = req.params.id;
  // Verificar si el registro existe antes de intentar actualizar
  mascotas.findByIdPk(id_mascota)
    .then((registroExistente) => {
      if (!registroExistente) {
        return res.status(404).json({
          mensaje: "Registro no encontrado"
        });
      }

      // El registro existe, ahora procedemos con la actualización
      if (!req.body.nombre && !req.body.edad && !req.body.tipo && !req.body.descripcion && !req.body.disponible) {
        return res.status(400).json({
          mensaje: "No se encontraron datos para actualizar"
        });
      }

      const nombre = req.body.nombre || registroExistente.nombre;
      const tipo = req.body.tipo || registroExistente.tipo;
      const edad = req.body.edad || registroExistente.edad;
      const descripcion = req.body.descripcion || registroExistente.descripcion;
      const disponible = req.body.disponible || registroExistente.disponible;

      mascotas.update({ nombre, tipo, edad, descripcion, disponible }, { where: { id_mascota } })
        .then(() => {
          res.status(200).json({
            mensaje: "Registro actualizado correctamente"
          });
        })
        .catch((err) => {
          res.status(500).json({
            mensaje: `Error al actualizar registro ::: ${err}`
          });
        });
    })
    .catch((err) => {
      res.status(500).json({
        mensaje: `Error al verificar la existencia del registro ::: ${err}`
      });
    });
};
```


Función eliminar mascotas

```
controladores > JS mascotasController.js > buscarMascotas
166 const eliminarMascota = (req, res) => {
167   const id_mascota = req.params.id;
168
169   if (id_mascota == null) {
170     res.status(203).json({
171       mensaje: "El id no puede estar vacío"
172     });
173     return;
174   }
175
176   // Verificar si la mascota con el id_mascota existe antes de intentar eliminar
177   mascotas.findById(id_mascota)
178     .then(mascotaExistente => {
179       if (!mascotaExistente) {
180         return res.status(404).json({
181           mensaje: "Mascota no encontrada"
182         });
183       }
184
185       // Verificar si la mascota se utiliza como clave foranea en otraTabla
186       solicitudes.findOne({ where: { idMascota : id_mascota } })
187         .then(otraTablaExistente => {
188           if (otraTablaExistente) {
189             return res.status(400).json({
190               mensaje: "La mascota está siendo utilizada como clave foranea en solicitudes. No se puede eliminar."
191             });
192           }
193
194           // La mascota no se utiliza como clave foranea, procedemos con la eliminación
195           mascotas.destroy({ where: { id_mascota : id_mascota } })
196             .then((resultado) => {
197               res.status(200).json({
198                 mensaje: "Registro Eliminado"
199               });
200             })
201             .catch((err) => {
202               res.status(500).json({
203                 mensaje: "Error al eliminar Registro ::: ${err}"
204               });
205             });
206           })
207           .catch((err) => {
208             res.status(500).json({
209               mensaje: "Error al verificar la existencia en otraTabla ::: ${err}"
210             });
211           });
212         })
213         .catch((err) => {
214           res.status(500).json({
215             mensaje: "Error al verificar la existencia de la mascota ::: ${err}"
216           });
217         });
218     });
219 }
```

Creación de archivo rutas mascotas

```

rutas > JS mascotasRouter.js > ...
1  import express from "express";
2  import {crearMascota,buscarIdMascotas,buscarMascotas,actualizarMascota, eliminarMascota,
3  |   | buscarMascotasDiponibles, buscarGatos, buscarPerros} from "../controladores/mascotasController.js";
4  const routerMascotas = express.Router();
5
6  routerMascotas.get("/",(req,res)=>{
7  |   res.send("Bienvenido a Mascotas");
8  | });
9
10 routerMascotas.post("/crearMascota",(req,res)=>{
11 |   crearMascota(req,res);
12 | });
13
14 routerMascotas.get("/buscarMascota/:id",(req,res)=>{
15 |   buscarIdMascotas(req,res);
16 | });
17
18 routerMascotas.get("/buscarMascota",(req,res)=>{
19 |   buscarMascotas(req,res);
20 | });
21
22 routerMascotas.get("/buscarPerros",(req,res)=>{
23 |   buscarPerros(req,res);
24 | });
25
26 routerMascotas.get("/buscarGatos",(req,res)=>{
27 |   buscarGatos(req,res);
28 | });
29
30 routerMascotas.get("/buscarMascotasDiponibles",(req,res)=>{
31 |   buscarMascotasDiponibles(req,res);
32 | });
33
34
35 routerMascotas.put("/actualizarMascota/:id",(req,res)=>{
36 |   actualizarMascota(req,res);
37 | });
38
39 routerMascotas.delete("/eliminarMascota/:id",(req,res)=>{
40 |   eliminarMascota(req,res);
41 | });
42
43 export {routerMascotas}
44
```

Tabla solicitudes

crear modelos solicitudes

```
modelos > JS solicitudesModelo.js > solicitudes > estadoSolicitud
1  import Sequelize from "sequelize";
2  import {db} from "../database/conexion.js";
3
4  const solicitudes = db.define("solicitudes", {
5    id_Solicitud: {
6      type: Sequelize.INTEGER,
7      allowNull: false,
8      primaryKey: true,
9    },
10   idMascota: {
11     type: Sequelize.INTEGER,
12     allowNull: false,
13     references: {
14       model: "mascotas", // Referencia a la tabla mascotas
15       key: "id_mascota",
16     },
17   },
18   nombreSolicitante: {
19     type: Sequelize.STRING(50),
20     allowNull: false,
21   },
22   correoSolicitante: {
23     type: Sequelize.STRING(50),
24     allowNull: false,
25   },
26
27   estadoSolicitud: {
28     type: Sequelize.STRING(20),
29     defaultValue: "Pendiente",
30   },
31 });
32
33 export {solicitudes}
```

Creación de archivos controlador

```
import {mascotas} from "../modelos/mascotasModelo.js";
import {solicitudes} from "../modelos/solicitudesModelo.js";
```

Función registrar solicitudes

```
const crearSolicitud = (req, res) => {
  // Verificar si el idMascota está presente y no es nulo
  if (!req.body.idMascota || !req.body.nombreSolicitante || !req.body.correoSolicitante) {
    return res.status(400).json({
      mensaje: "idMascota, nombreSolicitante y correoSolicitante son campos requeridos y no pueden ser nulos."
    });
  }

  // Verificar si la mascota con el idMascota existe y está disponible
  mascotas.findByPk(req.body.idMascota)
    .then((mascota) => {
      if (!mascota) {
        return res.status(404).json({
          mensaje: `La mascota con ID = ${req.body.idMascota} proporcionado no existe.`
        });
      }

      if (!mascota.disponible) {
        return res.status(400).json({
          mensaje: "La mascota no está disponible para adopción."
        });
      }

      // Crear un objeto dataset con los campos relevantes
      const dataset = {
        id_Solicitud: req.body.id_Solicitud,
        idMascota: req.body.idMascota,
        nombreSolicitante: req.body.nombreSolicitante,
        correoSolicitante: req.body.correoSolicitante,
      };

      // Utilizar Sequelize para crear el recurso
      solicitudes.create(dataset)
        .then((resultado) => {
          res.status(200).json({
            mensaje: "Registro creado correctamente",
            resultado: resultado
          });
        })
        .catch((err) => {
          res.status(500).json({
            mensaje: `Error al crear el registro: ${err.message}`
          });
        });
    })
    .catch((err) => {
      res.status(500).json({
        mensaje: `Error al buscar la mascota: ${err.message}`
      });
    });
};
```

Función buscar solicitudes por id

```
const buscarIdSolicitud = (req, res) => {
  const id = req.params.id;

  // Validar si el id es nulo o indefinido
  if (id == null) {
    res.status(203).json({
      mensaje: `El id no puede estar vacío`
    });
    return;
  }

  // Validar si el id es un número válido
  if (isNaN(id)) {
    res.status(203).json({
      mensaje: `El id debe ser un número válido`
    });
    return;
  }

  solicitudes.findByPk(id)
    .then((resultado) => {
      if (resultado) {
        // Si el resultado existe, devolverlo
        res.status(200).json(resultado);
      } else {
        // Si el resultado no existe, devolver un mensaje de error
        res.status(404).json({
          mensaje: `Registro no encontrado`
        });
      }
    })
    .catch((err) => {
      // Si ocurre un error durante la búsqueda, devolver un mensaje de error
      res.status(500).json({
        mensaje: `Error al buscar el registro ::: ${err}`
      });
    });
}
```

Buscar todas las solicitudes

```
const buscarSolicitudes = (_req, res) => {
  solicitudes.findAll().then((resultado) => {
    res.status(200).json(resultado);
  }).catch((err) => {
    res.status(500).json({
      mensaje: `No se encontraron Registros ::: ${err}`
    });
  });
};
```

Buscar solicitudes pendientes

```
const buscarSolicitudesPendientes = (req, res) => {  
  solicitudes.findAll({ where: { estadoSolicitud: "pendiente" }}).then((resultado) => {  
    res.status(200).json(resultado);  
  }).catch((err) => {  
    res.status(500).json({  
      mensaje: `No se encontraron Registros ::: ${err}`  
    });  
  });  
};
```

Función actualizar solicitud

```
const actualizarSolicitud = (req, res) => {  
  const id_Solicitud = req.params.id;  
  
  // Verificar si el registro de solicitud existe antes de intentar actualizar  
  solicitudes.findPk(id_Solicitud)  
    .then((registroExistente) => {  
      if (!registroExistente) {  
        return res.status(404).json({  
          mensaje: "Registro no encontrado"  
        });  
      }  
  
      // El registro existe, ahora procedemos con la actualización  
      if (!req.body.idMascota && !req.body.nombreSolicitante && !req.body.correoSolicitante && !req.body.estadoSolicitud) {  
        return res.status(400).json({  
          mensaje: "No se encontraron datos para actualizar"  
        });  
      }  
  
      const idMascota = req.body.idMascota || registroExistente.idMascota;  
  
      // Verificar si la mascota con el idMascota existe  
      mascotas.findPk(idMascota)  
        .then((mascotaExistente) => {  
          if (!mascotaExistente) {  
            return res.status(400).json({  
              mensaje: `La mascota con el ID = ${req.body.idMascota} no existe`  
            });  
          }  
  
          const nombreSolicitante = req.body.nombreSolicitante || registroExistente.nombreSolicitante;  
          const correoSolicitante = req.body.correoSolicitante || registroExistente.correoSolicitante;  
          const estadoSolicitud = req.body.estadoSolicitud || registroExistente.estadoSolicitud;  
  
          solicitudes.update({ idMascota, nombreSolicitante, correoSolicitante, estadoSolicitud }, { where: { id_Solicitud } })  
            .then(() => {  
              res.status(200).json({  
                mensaje: "Registro actualizado correctamente"  
              });  
            })  
            .catch((err) => {  
              res.status(500).json({  
                mensaje: `Error al actualizar registro ::: ${err}`  
              });  
            });  
        })  
        .catch((err) => {  
          res.status(500).json({  
            mensaje: `Error al verificar la existencia de la mascota ::: ${err}`  
          });  
        });  
    })  
    .catch((err) => {  
      res.status(500).json({  
        mensaje: `Error al verificar la existencia del registro de solicitud ::: ${err}`  
      });  
    });  
};
```

Función eliminar solicitud

```
const eliminarSolicitud = (req, res) => {
  const id_Solicitud = req.params.id;

  if (id_Solicitud == null) {
    res.status(203).json({
      mensaje: 'El id no puede estar vacío'
    });
    return;
  }

  // Verificar si el registro de solicitud existe antes de intentar eliminar
  solicitudes.findByPk(id_Solicitud)
    .then((registroExistente) => {
      if (!registroExistente) {
        return res.status(404).json({
          mensaje: "Registro no encontrado"
        });
      }

      // El registro existe, ahora procedemos con la eliminación
      solicitudes.destroy({ where: { id_Solicitud } })
        .then((resultado) => {
          res.status(200).json({
            mensaje: 'Registro Eliminado'
          });
        })
        .catch((err) => {
          res.status(500).json({
            mensaje: 'Error al eliminar Registro ::: ${err}'
          });
        });
    })
    .catch((err) => {
      res.status(500).json({
        mensaje: 'Error al verificar la existencia del registro de solicitud ::: ${err}'
      });
    });
};
```

Función aceptar solicitud

```
const aceptarSolicitud = (req, res) => {
  const id_Solicitud = req.params.id;

  if (id_Solicitud == null) {
    return res.status(203).json({
      mensaje: "El id no puede estar vacío"
    });
  }

  // Verificar si la solicitud con el id_Solicitud existe antes de intentar aceptar
  solicitudes.findByPk(id_Solicitud)
    .then((solicitudExistente) => {
      if (!solicitudExistente) {
        return res.status(404).json({
          mensaje: "Solicitud no encontrada"
        });
      }

      // Verificar si la solicitud ya ha sido aceptada
      if (solicitudExistente.estadoSolicitud === 'Aceptada') {
        return res.status(400).json({
          mensaje: "La solicitud ya ha sido aceptada previamente"
        });
      }

      // Actualizar el estado de la solicitud a "Aceptada"
      solicitudes.update({ estadoSolicitud: 'Aceptada' }, { where: { id_Solicitud } })
        .then(() => {
          res.status(200).json({
            mensaje: "Solicitud aceptada correctamente"
          });
        })
        .catch((err) => {
          res.status(500).json({
            mensaje: 'Error al aceptar la solicitud ::: ${err}'
          });
        });
    })
    .catch((err) => {
      res.status(500).json({
        mensaje: 'Error al verificar la existencia de la solicitud ::: ${err}'
      });
    });
};
```

Función rechazar solicitud

```
const rechazarSolicitud = (req, res) => {
  const id_Solicitud = req.params.id;

  if (id_Solicitud == null) {
    return res.status(203).json({
      mensaje: "El id no puede estar vacío"
    });
  }

  // Verificar si la solicitud con el id_Solicitud existe antes de intentar aceptar
  solicitudes.findByIdPk(id_Solicitud)
    .then((solicitudExistente) => {
      if (!solicitudExistente) {
        return res.status(404).json({
          mensaje: "Solicitud no encontrada"
        });
      }

      // Verificar si la solicitud ya ha sido aceptada
      if (solicitudExistente.estadoSolicitud === 'Aceptada') {
        return res.status(400).json({
          mensaje: "La solicitud ya ha sido aceptada previamente"
        });
      }

      // Actualizar el estado de la solicitud a "Aceptada"
      solicitudes.update({ estadoSolicitud: 'Rechazada' }, { where: { id_Solicitud } })
        .then(() => {
          res.status(200).json({
            mensaje: "Solicitud aceptada correctamente"
          });
        })
        .catch((err) => {
          res.status(500).json({
            mensaje: `Error al aceptar la solicitud ::: ${err}`
          });
        });
    })
    .catch((err) => {
      res.status(500).json({
        mensaje: `Error al verificar la existencia de la solicitud ::: ${err}`
      });
    });
};
```


Crear archivo rutas solicitud

```
import express from "express";
import { buscarIdSolicitud, crearSolicitud, buscarSolicitudes,
  actualizarSolicitud, eliminarSolicitud, buscarSolicitudesPendientes,
  aceptarSolicitud, rechazarSolicitud, buscarSolicitudesAceptada,
  buscarSolicitudesRechazada } from "../controladores/solicitudesController.js";
const routerSolicitudes = express.Router();

routerSolicitudes.post("/crearSolicitud", (req, res) => {
  crearSolicitud(req, res);
});

routerSolicitudes.get("/buscarIdSolicitud/:id", (req, res) => {
  buscarIdSolicitud(req, res);
});

routerSolicitudes.get("/buscarSolicitudes", (req, res) => {
  buscarSolicitudes(req, res);
});

routerSolicitudes.get("/buscarSolicitudesPendientes", (req, res) => {
  buscarSolicitudesPendientes(req, res);
});

routerSolicitudes.get("/buscarSolicitudesAceptada", (req, res) => {
  buscarSolicitudesAceptada(req, res);
});

routerSolicitudes.get("/buscarSolicitudesRechazada", (req, res) => {
  buscarSolicitudesRechazada(req, res);
});

routerSolicitudes.put("/actualizarSolicitud/:id", (req, res) => {
  actualizarSolicitud(req, res);
});
routerSolicitudes.put("/aceptarSolicitud/:id", (req, res) => {
  aceptarSolicitud(req, res);
});

routerSolicitudes.put("/RechazarSolicitud/:id", (req, res) => {
  rechazarSolicitud(req, res);
});

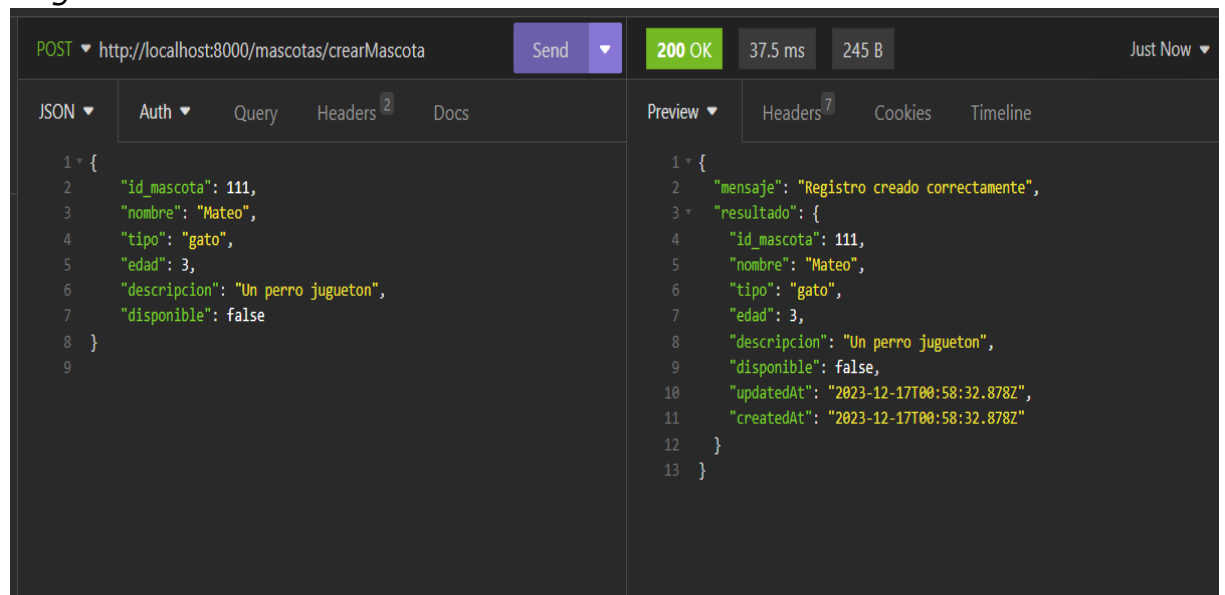
routerSolicitudes.delete("/eliminarSolicitud/:id", (req, res) => {
  eliminarSolicitud(req, res);
});

export { routerSolicitudes }
```

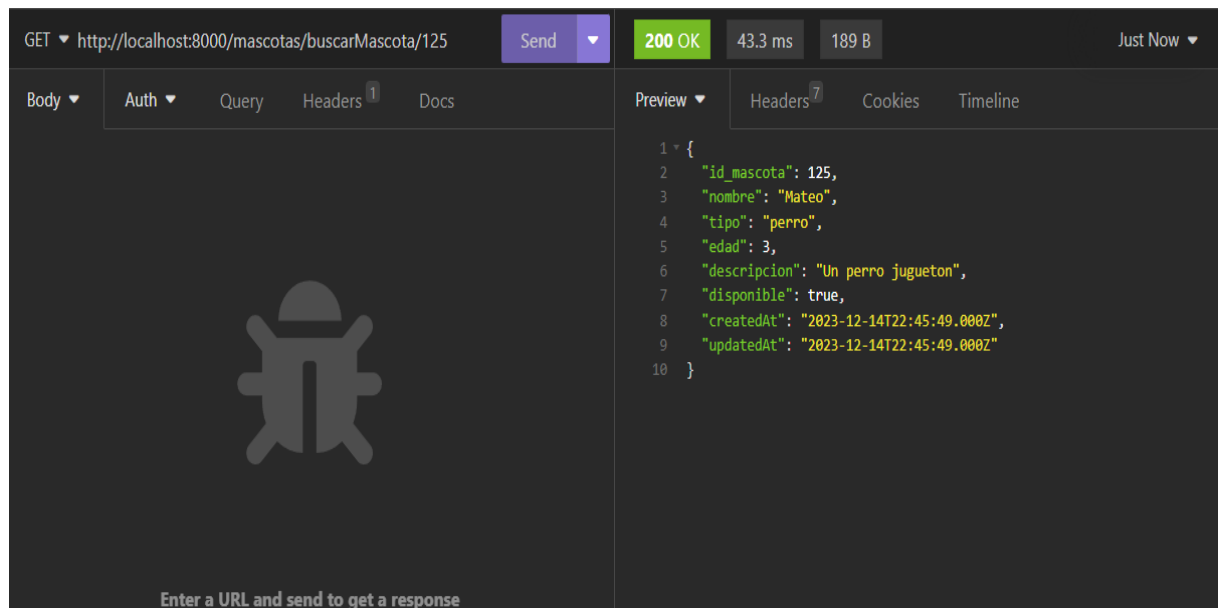
3. Realizar verificación de las diferentes operaciones a través de un cliente grafico (Postman, Imnsomia, etc.), tomar capturas de pantalla que evidencien el resultado de las solicitudes realizadas.

Tabla mascotas

Registro de mascotas




Buscar por id mascotas



Buscar mascotas disponibles


GET ▼ http://localhost:8000/mascotas/buscarMascotasDiponibl Send ▼ 200 OK 22.2 ms 949 B Just Now ▼

Body ▼ Auth ▼ Query Headers 1 Docs



Enter a URL and send to get a response

Select a body type from above to send data in the body of a request

[Introduction to Insomnia](#) 


Preview ▼ Headers 7 Cookies Timeline

```
1 [
2 {
3   "id_mascota": 1,
4   "nombre": "Mateo",
5   "tipo": "perro",
6   "edad": 3,
7   "descripcion": "Un perro jugueton",
8   "disponible": true,
9   "createdAt": "2023-12-14T22:52:05.000Z",
10  "updatedAt": "2023-12-14T22:52:05.000Z"
11 },
12 {
13   "id_mascota": 124,
14   "nombre": "Mateo",
15   "tipo": "perro",
16   "edad": 3,
17   "descripcion": "Un perro jugueton",
18   "disponible": true,
19   "createdAt": "2023-12-14T22:03:44.000Z",
20   "updatedAt": "2023-12-14T22:03:44.000Z"
21 },
22 {
23   "id_mascota": 125,
24   "nombre": "Mateo",
25   "tipo": "perro",
26   "edad": 3,
27   "descripcion": "Un perro jugueton",
28   "disponible": true,
29   "createdAt": "2023-12-14T22:45:49.000Z",
30   "updatedAt": "2023-12-14T22:45:49.000Z"
31 },
32 ]
```

Buscar gatos


GET ▼ http://localhost:8000/mascotas/buscarGatos Send ▼ 200 OK 20.9 ms 571 B Just Now ▼

Body ▼ Auth ▼ Query Headers 1 Docs



Enter a URL and send to get a response

Select a body type from above to send data in the body of a request

[Introduction to Insomnia](#) 

Preview ▼ Headers 7 Cookies Timeline

```
1 [
2 {
3   "id_mascota": 111,
4   "nombre": "Mateo",
5   "tipo": "gato",
6   "edad": 3,
7   "descripcion": "Un perro jugueton",
8   "disponible": false,
9   "createdAt": "2023-12-17T00:58:32.000Z",
10  "updatedAt": "2023-12-17T00:58:32.000Z"
11 },
12 {
13   "id_mascota": 121,
14   "nombre": "Mateo",
15   "tipo": "gato",
16   "edad": 3,
17   "descripcion": "Un perro jugueton",
18   "disponible": false,
19   "createdAt": "2023-12-15T23:21:10.000Z",
20   "updatedAt": "2023-12-15T23:21:10.000Z"
21 },
22 {
23   "id_mascota": 122,
24   "nombre": "Mateo",
25   "tipo": "gato",
26   "edad": 3,
27   "descripcion": "Un perro jugueton",
28   "disponible": false,
29   "createdAt": "2023-12-15T23:14:21.000Z",
30   "updatedAt": "2023-12-15T23:14:21.000Z"
31 },
32 ]
```

Buscar perros

GET ▼ http://localhost:8000/mascotas/buscarperros Send ▼ 200 OK 12.1 ms 1902 B Just Now ▼

Body ▼ Auth ▼ Query Headers 1 Docs

Preview ▼ Headers 7 Cookies Timeline

49 "createdAt": "2023-12-14T22:45:49.000Z",
50 "updatedAt": "2023-12-14T22:45:49.000Z"
51 },
52 {
53 "id_mascota": 126,
54 "nombre": "Mateo",
55 "tipo": "perro",
56 "edad": 3,
57 "descripcion": "Un perro jugueton",
58 "disponible": true,
59 "createdAt": "2023-12-14T22:46:28.000Z",
60 "updatedAt": "2023-12-14T22:46:28.000Z"
61 },
62 {
63 "id_mascota": 127,
64 "nombre": "Mateo",
65 "tipo": "perro",
66 "edad": 3,
67 "descripcion": "Un perro jugueton",
68 "disponible": true,
69 "createdAt": "2023-12-14T22:47:35.000Z",
70 "updatedAt": "2023-12-14T22:47:35.000Z"
71 },
72 {
73 "id_mascota": 128,
74 "nombre": "Mateo",
75 "tipo": "perro",
76 "edad": 3,
77 "descripcion": "Un perro jugueton",
78 "disponible": false,
79 "createdAt": "2023-12-14T22:47:59.000Z",
80 "updatedAt": "2023-12-14T22:47:59.000Z"

Enter a URL and send to get a response

Select a body type from above to send data in the body of a request

Introduction to Insomnia [↗](#)

Buscar mascotas

GET ▼ http://localhost:8000/mascotas/buscarMascota Send ▼ 200 OK 19 ms 2.4 KB Just Now ▼

Body ▼ Auth ▼ Query Headers 1 Docs

Preview ▼ Headers 7 Cookies Timeline

1 [
2 {
3 "id_mascota": 1,
4 "nombre": "Mateo",
5 "tipo": "perro",
6 "edad": 3,
7 "descripcion": "Un perro jugueton",
8 "disponible": true,
9 "createdAt": "2023-12-14T22:52:05.000Z",
10 "updatedAt": "2023-12-14T22:52:05.000Z"
11 },
12 {
13 "id_mascota": 16,
14 "nombre": "Mateo",
15 "tipo": "perro",
16 "edad": 3,
17 "descripcion": "Un perro jugueton",
18 "disponible": false,
19 "createdAt": "2023-12-15T00:28:33.000Z",
20 "updatedAt": "2023-12-15T00:28:33.000Z"
21 },
22 {
23 "id_mascota": 111,
24 "nombre": "Mateo",
25 "tipo": "gato",
26 "edad": 3,
27 "descripcion": "Un perro jugueton",
28 "disponible": false,
29 "createdAt": "2023-12-17T00:58:32.000Z",
30 "updatedAt": "2023-12-17T00:58:32.000Z"
31 },
32]

Enter a URL and send to get a response

Select a body type from above to send data in the body of a request

Introduction to Insomnia [↗](#)

Actualizar mascota

PUT http://localhost:8000/mascotas/actualizarMascota/166 Send 200 OK 34 ms 48 B Just Now

JSON Auth Query Headers 2 Docs

```
1 {  
2   "nombre": "Max",  
3   "edad": 10  
4 }
```

Preview Headers 7 Cookies Timeline

```
1 {  
2   "mensaje": "Registro actualizado correctamente"  
3 }
```

Eliminar registro

DELETE http://localhost:8000/mascotas/eliminarMascota/111 Send 200 OK 55.9 ms 32 B Just Now

Body Auth Query Headers 1 Docs

```
1 {  
2   "mensaje": "Registro Eliminado"  
3 }
```

Tabla solicitudes

Registrar solicitud

POST http://localhost:8000/solicitudes/crearSolicitud Send 200 OK 20.2 ms 264 B Just Now

JSON Auth Query Headers ² Docs

```
1 {
2   "id_Solicitud": 111,
3   "idMascota": 1,
4   "nombreSolicitante": "Fredy",
5   "correoSolicitante": "FREDY@gmail.com"
6 }
```


Preview Headers ⁷ Cookies Timeline

```
1 {
2   "mensaje": "Registro creado correctamente",
3   "resultado": {
4     "estadoSolicitud": "Pendiente",
5     "id_Solicitud": 111,
6     "idMascota": 1,
7     "nombreSolicitante": "Fredy",
8     "correoSolicitante": "FREDY@gmail.com",
9     "updatedAt": "2023-12-17T01:14:19.011Z",
10    "createdAt": "2023-12-17T01:14:19.011Z"
11  }
12 }
```

Buscar solicitudes

GET http://localhost:8000/solicitudes/buscarSolicitudes Send 200 OK 12.5 ms 627 B Just Now

Body Auth Query Headers ¹ Docs



Enter a URL and send to get a response

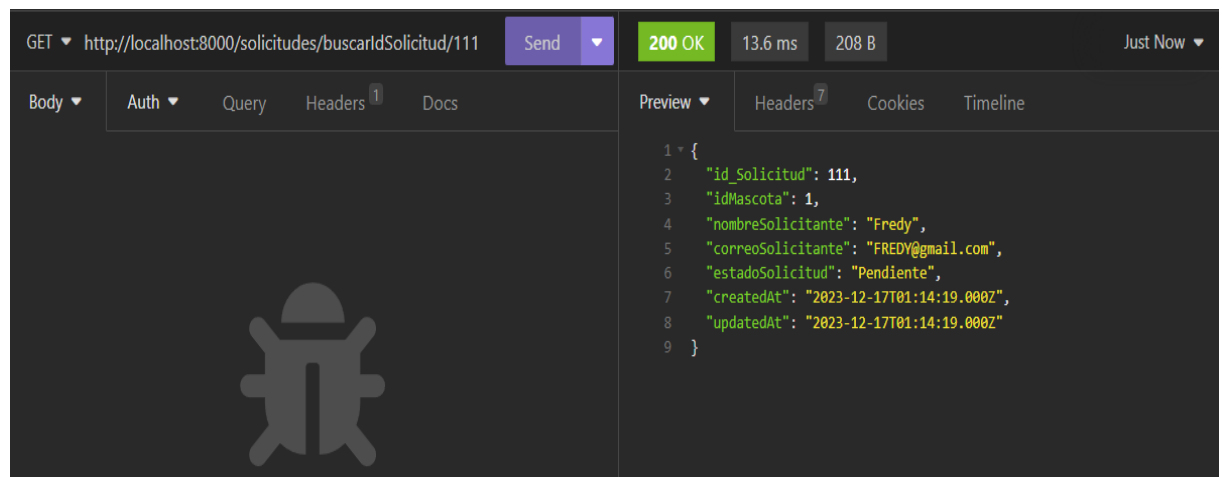
Select a body type from above to send data in the body of a request

[Introduction to Insomnia](#)

Preview Headers ⁷ Cookies Timeline

```
1 [
2   {
3     "id_Solicitud": 111,
4     "idMascota": 1,
5     "nombreSolicitante": "Fredy",
6     "correoSolicitante": "FREDY@gmail.com",
7     "estadoSolicitud": "Pendiente",
8     "createdAt": "2023-12-17T01:14:19.000Z",
9     "updatedAt": "2023-12-17T01:14:19.000Z"
10  },
11   {
12     "id_Solicitud": 124,
13     "idMascota": 1,
14     "nombreSolicitante": "Fredy",
15     "correoSolicitante": "FREDY@gmail.com",
16     "estadoSolicitud": "Aceptada",
17     "createdAt": "2023-12-15T23:35:46.000Z",
18     "updatedAt": "2023-12-17T00:42:01.000Z"
19  },
20   {
21     "id_Solicitud": 154,
22     "idMascota": 1,
23     "nombreSolicitante": "Fredy",
24     "correoSolicitante": "FREDY@gmail.com",
25     "estadoSolicitud": "Rechazada",
26     "createdAt": "2023-12-16T12:37:52.000Z",
27     "updatedAt": "2023-12-17T00:51:10.000Z"
28   }
29 ]
```

Buscar solicitudes por id



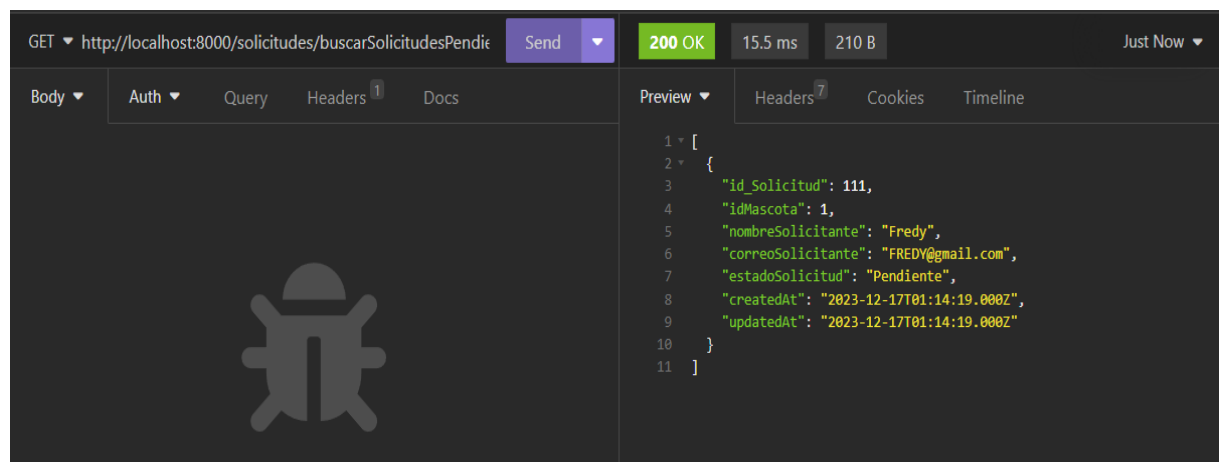
GET ▼ http://localhost:8000/solicitudes/buscarIdSolicitud/111 Send ▼ **200 OK** 13.6 ms 208 B Just Now ▼

Body ▼ Auth ▼ Query Headers 1 Docs

Preview ▼ Headers 7 Cookies Timeline

```
1 {
2   "id_Solicitud": 111,
3   "idMascota": 1,
4   "nombreSolicitante": "Fredy",
5   "correoSolicitante": "FREDY@gmail.com",
6   "estadoSolicitud": "Pendiente",
7   "createdAt": "2023-12-17T01:14:19.000Z",
8   "updatedAt": "2023-12-17T01:14:19.000Z"
9 }
```

Buscar solicitudes pendientes



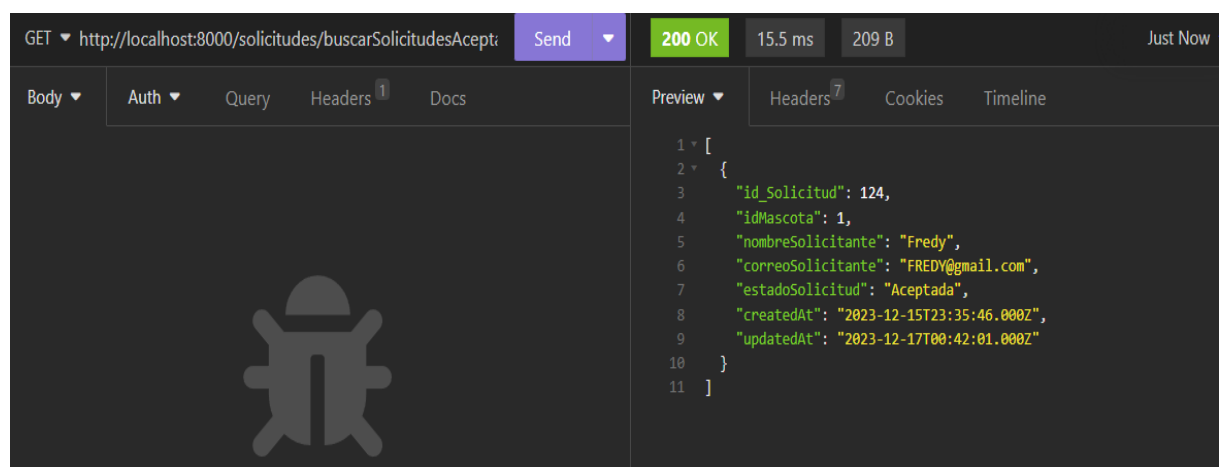
GET ▼ http://localhost:8000/solicitudes/buscarSolicitudesPendie Send ▼ **200 OK** 15.5 ms 210 B Just Now ▼

Body ▼ Auth ▼ Query Headers 1 Docs

Preview ▼ Headers 7 Cookies Timeline

```
1 [
2   {
3     "id_Solicitud": 111,
4     "idMascota": 1,
5     "nombreSolicitante": "Fredy",
6     "correoSolicitante": "FREDY@gmail.com",
7     "estadoSolicitud": "Pendiente",
8     "createdAt": "2023-12-17T01:14:19.000Z",
9     "updatedAt": "2023-12-17T01:14:19.000Z"
10  }
11 ]
```

Buscar solicitudes aceptadas



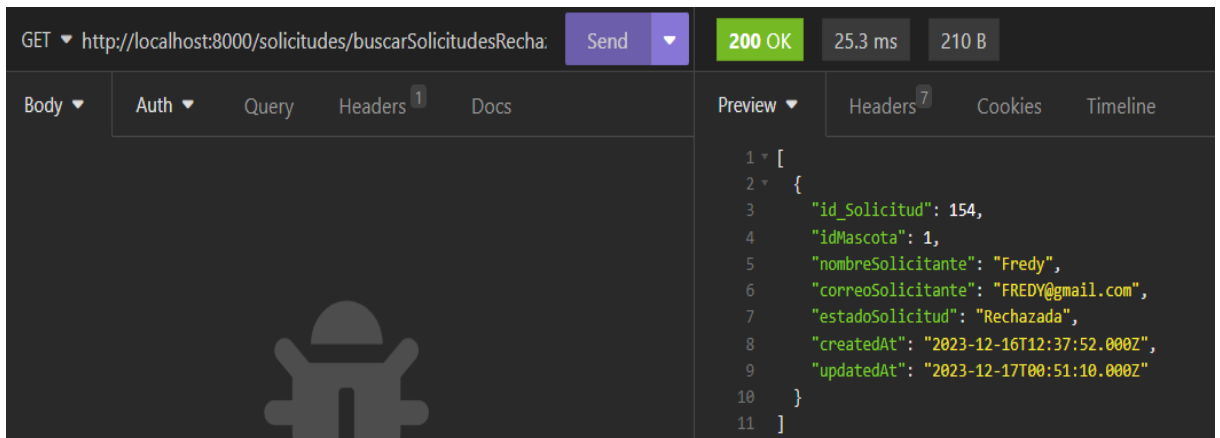
GET ▼ http://localhost:8000/solicitudes/buscarSolicitudesAcepta Send ▼ **200 OK** 15.5 ms 209 B Just Now ▼

Body ▼ Auth ▼ Query Headers 1 Docs

Preview ▼ Headers 7 Cookies Timeline

```
1 [
2   {
3     "id_Solicitud": 124,
4     "idMascota": 1,
5     "nombreSolicitante": "Fredy",
6     "correoSolicitante": "FREDY@gmail.com",
7     "estadoSolicitud": "Aceptada",
8     "createdAt": "2023-12-15T23:35:46.000Z",
9     "updatedAt": "2023-12-17T00:42:01.000Z"
10  }
11 ]
```

Buscar solicitudes rechazadas



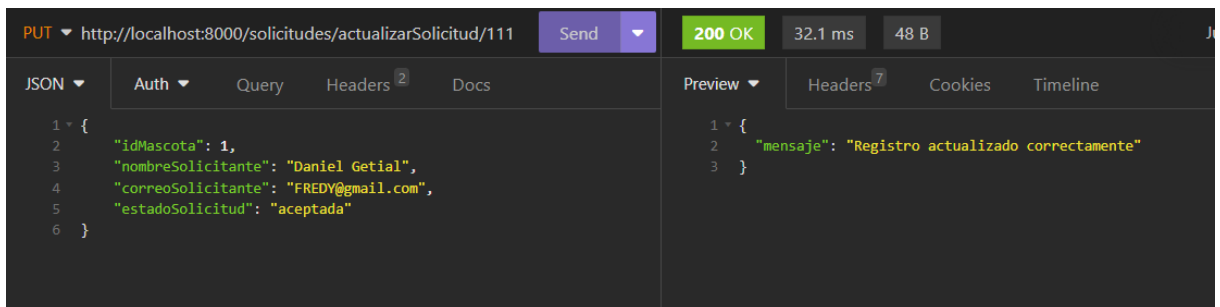
GET ▼ http://localhost:8000/solicitudes/buscarSolicitudesRecha: Send ▼ **200 OK** 25.3 ms 210 B

Body ▼ Auth ▼ Query Headers ¹ Docs

Preview ▼ Headers ⁷ Cookies Timeline

```
1 [
2   {
3     "id_Solicitud": 154,
4     "idMascota": 1,
5     "nombreSolicitante": "Fredy",
6     "correoSolicitante": "FREDY@gmail.com",
7     "estadoSolicitud": "Rechazada",
8     "createdAt": "2023-12-16T12:37:52.000Z",
9     "updatedAt": "2023-12-17T00:51:10.000Z"
10  }
11 ]
```

Actualizar solicitud



PUT ▼ http://localhost:8000/solicitudes/actualizarSolicitud/111 Send ▼ **200 OK** 32.1 ms 48 B

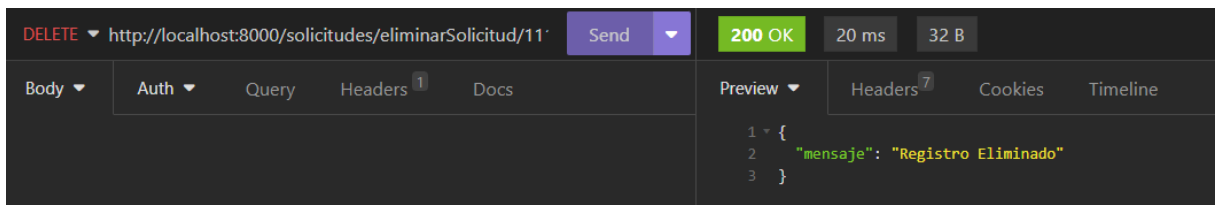
JSON ▼ Auth ▼ Query Headers ² Docs

Preview ▼ Headers ⁷ Cookies Timeline

```
1 {
2   "idMascota": 1,
3   "nombreSolicitante": "Daniel Getial",
4   "correoSolicitante": "FREDY@gmail.com",
5   "estadoSolicitud": "aceptada"
6 }
```

```
1 {
2   "mensaje": "Registro actualizado correctamente"
3 }
```

Eliminar solicitud



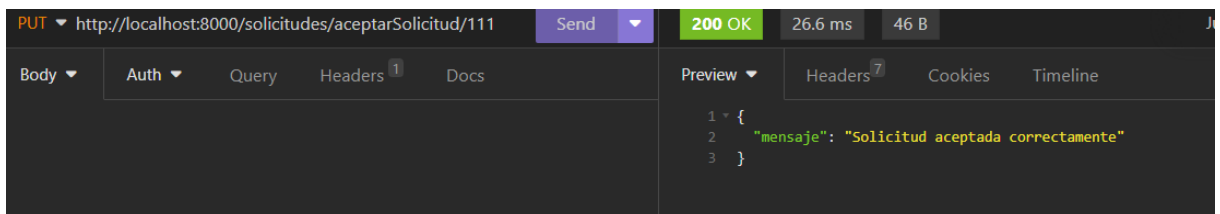
DELETE ▼ http://localhost:8000/solicitudes/eliminarSolicitud/111 Send ▼ **200 OK** 20 ms 32 B

Body ▼ Auth ▼ Query Headers ¹ Docs

Preview ▼ Headers ⁷ Cookies Timeline

```
1 {
2   "mensaje": "Registro Eliminado"
3 }
```

Aceptar solicitud



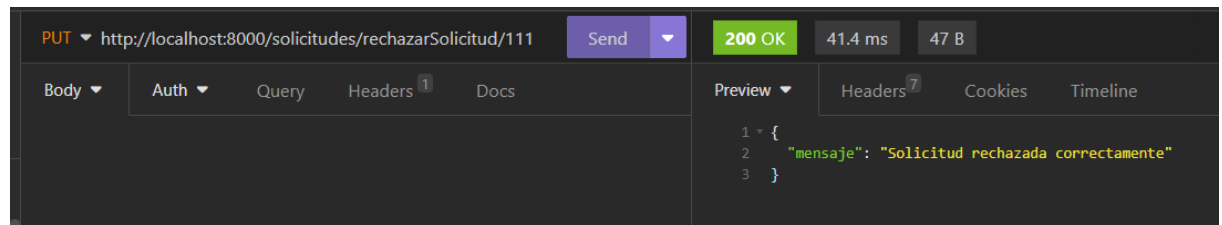
PUT ▼ http://localhost:8000/solicitudes/aceptarSolicitud/111 Send ▼ **200 OK** 26.6 ms 46 B

Body ▼ Auth ▼ Query Headers ¹ Docs

Preview ▼ Headers ⁷ Cookies Timeline

```
1 {
2   "mensaje": "Solicitud aceptada correctamente"
3 }
```


Rechazar solicitud



Conclusion

En este taller, he creado una base de datos MySQL para gestionar el registro de mascotas y el proceso de adopción, estableciendo una estructura que abarca perros y gatos. Además, he desarrollado una aplicación backend utilizando NodeJS y ExpressJS, la cual interactúa con la base de datos para llevar a cabo tareas relacionadas con el registro y la administración de mascotas para adopción. Se ha implementado el uso adecuado de verbos HTTP según la operación requerida. Para validar la funcionalidad, he verificado las operaciones a través de un cliente gráfico, generando capturas de pantalla que documentan los resultados de las solicitudes realizadas con herramientas como Insomnia. Finalmente, he preparado un informe detallado que describe el proceso de construcción e implementación del aplicativo. El código y el informe están disponibles en un repositorio remoto en GitHub, cuyo enlace ha sido enviado al correo especificado antes del plazo establecido. Este proyecto demuestra mi habilidad para diseñar y desarrollar soluciones completas, integrando tecnologías como bases de datos y servidores web para abordar de manera efectiva los requisitos propuestos.

