

DIPLOMADO DE ACTUALIZACION EN NUEVAS TECNOLOGIAS PARA  
DESARROLLO DE SOFTWARE

PROYECTO FINAL

PREPARADO POR:  
MILER ANDRES ESPAÑA  
COD: 2130341131

PREPARADO PARA:  
VICENTE AUX

UNIVERSIDAD DE NARIÑO  
INGENIERIA DE SISTEMAS  
ENERO 2024

## 1. Modificamos base de datos

- Creamos nuevas tablas de los adoptantes y el estado de las mascotas, si están libres o solicitadas

```
MySQL [mascotas]> show tables;
+-----+
| Tables_in_mascotas |
+-----+
| adoptantes          |
| estado              |
| mascotas            |
+-----+
```

- Dentro de la tabla adoptantes y estado creamos los siguientes datos para guardar los datos del adoptador y el estado en que se encuentran las mascotas.

```
MySQL [mascotas]> describe adoptantes;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int           | NO   | PRI | NULL    | auto_increment |
| nombre     | varchar(100)  | NO   |     | NULL    |                |
| telefono   | varchar(100)  | NO   |     | NULL    |                |
| direccion  | varchar(100)  | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.017 sec)
```

```
MySQL [mascotas]> describe estado;
+-----+-----+-----+-----+-----+-----+
| Field        | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id           | int           | NO   | PRI | NULL    | auto_increment |
| descripcion  | varchar(100)  | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.007 sec)
```

## 2. Creamos las rutas necesarias para el funcionamiento de la aplicación.

- Creamos las rutas de usuario, este puede ser administrador o usuario.

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

const router = express.Router()

router.get('/', controller.readUsuario)
router.get('/:id', controller.readUsuario)
router.post('/crear', controller.crearUsuario)
router.post('/login', controller.loginUsuario)
router.post('/salir', controller.salirUsuario)
router.put('/actualizar/:id', controller.actualizarusuario)
router.delete('/eliminar/:id', controller.eliminarUsuario)

export { router as usersRouter }
```

b. Creamos las rutas de las solicitudes de adopción

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

const router = express.Router()

router.get('/', controller.readRequests)
router.get('/:id', controller.readRequest)
router.post('/crear', controller.crearRequest)
router.put('/actualizar/:id', controller.actualizarRequest)
router.put('/aprobar/:id', controller.aprobarRequest)
router.delete('/eliminar/:id', controller.eliminarRequest)

export { router as requestsRouter }
```

3. Ahora pasamos a crear el controlador de los usuarios, aquí encontraremos:

- Crear usuario
- Iniciar sesión
- Cerrar sesión
- Eliminar usuario
- Actualizar usuario

```
export const crearUsuario = async (req, res) => {
  if (req.body.name === undefined) return res.status(400).jsonPretty({
    type: 'error',
    message: 'campo vacio'
  })
  if (req.body.email === undefined) return res.status(400).jsonPretty({
    type: 'error',
    message: 'campo vacio'
  })
  if (req.body.password === undefined) return res.status(400).jsonPretty({
    type: 'error',
    message: 'campo vacio'
  })

  req.body.password = await hashPassword(req.body.password)
  const user = await User.create(req.body)

  res.status(201).jsonPretty({
    type: 'success',
    message: `User ${user.nombre} (id: ${user.id}) usuario creado correctamente`,
    created: user
  })
}
```

```
export const loginUsuario = async (req, res) => {
  if (req.body.email === undefined) return res.status(400).jsonPretty({
    type: 'error',
    message: 'campo vacio'
  })
  if (req.body.password === undefined) return res.status(400).jsonPretty({
    type: 'error',
    message: 'campo vacio'
  })

  const user = await User.findOne({ where: { email: req.body.email } })
  if (user === null) return res.status(404).jsonPretty({
    type: 'error',
    message: 'usuario incorrecto'
  })

  const isPasswordValid = await bcrypt.compare(req.body.password, user.password)
  if (!isPasswordValid)
    return res.status(401).jsonPretty({ type: 'error', message: 'contraseña incorrecta' })

  req.session.user = user
  res.status(200).jsonPretty({
    type: 'success',
    message: 'login correctamente',
    user
  })
}
```

```
export const salirUsuario = (req, res) => {
  if (req.session.user === undefined) return res.status(401).jsonPretty({
    type: 'error',
    message: 'usuario no registrado'
  })

  req.session.destroy()
  res.status(200).jsonPretty({ type: 'success' })
}
```

```
export const actualizarUsuario = async (req, res) => {
  const id = req.params.id
  const { nombre, role, email, password, direccion } = req.body

  const user = await getUserById(id)
  if (user === null) return res.status(404).jsonPretty({
    type: 'error',
    message: `usuario no existe`
  })

  const isValidData = !nombre && !role && !email && !password && !direccion
  if (isValidData) return res.status(400).jsonPretty({
    type: 'error',
    message: `datos no validos`
  })

  if (password !== undefined)
    req.body.password = await hashPassword(password)
  user.update(req.body)

  res.status(200).jsonPretty({
    type: 'success',
    message: `actualizado correctamente`,
    updated: user
  })
}
```

```
export const eliminarUsuario = async (req, res) => {
  const id = req.params.id
  const user = await getUserById(id)

  if (user === null) return res.status(404).jsonPretty({
    type: 'error',
    message: `usuario no existe`
  })

  user.destroy()
  res.status(200).jsonPretty({
    type: 'success',
    message: `eliminado correctamente`,
    deleted: user
  })
}
```

4. Modificamos el FrontEnd de nuestra aplicación, agregando las nuevas vistas que tendrá el usuario que no sea administrador.
  - a. Agregamos un buscador, el cual buscara las mascotas por nombre y raza

```
<header>
  <div className="buscar">
    <input className="search-bar" type="text" placeholder="Buscar Mascota"></input>
    <button className="btn btn-success">
      <i className="fa-solid fa-circle-search"></i>Buscar
    </button>
  </div>
  <div className="container1 cont-boton">
    <button className="este"><span class="fa-circle-plus lead mr-8">
      account_circle</span></button>
  </div>
</header>
```

Buscar

□ account\_circle

- b. Agregamos un botón “Adoptar” y “donar”; adoptar se encargará de enviar la solicitud a la base de datos y cambiar de estado la mascota, el botón de donar permitirá aceptar donaciones de alimento o dinero para la fundación.

```
<div className="row mt-2">
  <div className="col-md-11 mb-11">
    <button
      onClick={()=>adoptarMascota(mascota.id,mascota.nombre)}
      className="btn btn-success col-md-5 mb-5">
      <i className="fa-solid"></i>Adoptar
    </button>
    <button
      className="btn btn-warning col-md-5 mb-5">
      <i className="fa-solid"></i>Donar
    </button>
  </div>
</div>
```

Nombre: Laika Raza: Golden Perro Edad: 4 <button>Detalles</button>	Nombre: Pancho Raza: Angora Gato Edad: 2 <button>Detalles</button>
<button>Adoptar</button> <button>Donar</button>	<button>Adoptar</button> <button>Donar</button>

- c. Agregamos la funcionalidad de adoptar mascota mediante SweetAlert.

```
const adoptarMascota=(id,nombre)=>{
  const MySwal = withReactContent(Swal);
  MySwal.fire({
    title: `Estas seguro de ADOPTAR la mascota ${nombre} ?`,
    icon: 'question',
    text: 'Se ADOPTARÁ Definitivamente',
    showCancelButton: true,
    confirmButtonText: 'Si, ADOPTAR',
    cancelButtonText: 'Cancelar'
  }).then((result)=>{
    if(result.isConfirmed){
      mostrarAlerta("Mascota feliz","info");
    }
    else{
      mostrarAlerta("No se elimino la mascota","info");
    }
  })
}
```

# Estas seguro de ADOPTAR la mascota Pancho ?

Se ADOPTARÁ Definitivamente

Si, ADOPTAR

Cancelar



## Mascota feliz

OK

- d. Importamos el archivo estilos.css para darle formato y estilo al FrontEnd de la página.


```
import className from "../estilos.css";
```

```
header{
  display: flex;
  height: 60px;
  background-color: #252525;
  align-items: center;
  padding: 10px;
}

*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: Arial,Roboto,"-apple-system",Helvetica,sans-serif;
}

.buscar{
  display: flex;
  align-items: center;
  position: absolute;
  left: 400px;
  height: 40px;
  width: 490px;
  background: rgb(252, 252, 252);
  padding: 10px;
  border-radius: 10px;
```


5. Finalmente tenemos las vistas de la siguiente manera




# Corazón Gatuno

FUNDACIÓN DE PROTECCIÓN ANIMAL


☐ account\_circle




Nombre: Laika  
Raza: Golden  
Perro  
Edad: 4




Nombre: Pancho  
Raza: Angora  
Gato  
Edad: 2



Nombre: Pepe  
Raza: Comun  
Gato  
Edad: 5



Nombre: Kaiser  
Raza: Bernes  
Perro  
Edad: 2



**Estas seguro de ADOPTAR la mascota Pepe ?**

Se ADOPTARÁ Definitivamente

