**Controllers\Usuario.controller.js**

```
// Importamos los módulos necesarios
const db = require('../Models'); // Importamos los modelos de la base de datos
const bcrypt = require('bcrypt'); // Importamos bcrypt para cifrar las contraseñas
const jwt = require('jsonwebtoken'); // Importamos jsonwebtoken para generar tokens de autenticación
const llave = require('dotenv').config().parsed.SECRET_KEY; // Importamos la llave secreta para firmar los tokens
const logger = require('../Config/logger'); // Importamos el logger

// Función para obtener la lista de todos los usuarios
exports.lista = (req, res, next) =>{
    db.Usuario.findAll() // Buscamos todos los usuarios en la base de datos
    .then(usuarios => {
        res.json(usuarios); // Enviamos los usuarios como respuesta
    }).catch(err => {
        logger.error('Error al obtener la lista de usuarios: ', err); // Imprimimos el error en el log
        next(err); // Pasamos el error al middleware de manejo de errores
    });
}

// Función para filtrar usuarios por un campo y valor específicos
exports.filtrar = (req, res, next) =>{
    const campo = req.params.campo; // Obtenemos el campo por el que queremos filtrar de los parámetros de la solicitud
    const valor = req.params.valor; // Obtenemos el valor por el que queremos filtrar de los parámetros de la solicitud
    db.Usuario.findAll({
        where: {
            [campo]: valor // Filtramos los usuarios por el campo y valor especificados
        }
    })
    .then(usuarios => {
        res.json(usuarios); // Enviamos los usuarios filtrados como respuesta
    }).catch(err => {
        logger.error('Error al obtener la lista de usuarios: ', err);
        next(err); // Pasamos el error al middleware de manejo de errores
    });
}

// Función para crear un nuevo usuario
exports.nuevo = (req, res, next) => {
    if(!req.body.nombre || !req.body.email || !req.body.password || !req.body.dni || !req.body.id_rol){
        res.status(400).send({
            message: "Faltan datos" // Enviamos un mensaje de error si faltan datos
        });
        return;
    }

    // Verificamos si el dni ya está en uso
    db.Usuario.findOne({
        where: {
            dni: req.body.dni // Buscamos un usuario con el dni proporcionado en el cuerpo de la solicitud
        }
    })
    .then(usuarioExistente => {
        if(usuarioExistente){
```

```
            res.status(400).send({
                message: "El usuario ya existe en la Base de Datos" // Enviamos un mensaje de error si el dni ya está en uso
            });
            return;
        }

        // Si el dni no está en uso, creamos el nuevo usuario
        const usuario = {
            nombre: req.body.nombre, // Obtenemos el nombre del cuerpo de la solicitud
            email: req.body.email, // Obtenemos el email del cuerpo de la solicitud
            dni: req.body.dni, // Obtenemos el DNI del cuerpo de la solicitud
            password: bcrypt.hashSync(req.body.password, 8), // Ciframos la contraseña obtenida del cuerpo de la solicitud
            id_rol: req.body.id_rol // Obtenemos el ID del rol del cuerpo de la solicitud
        }
        db.Usuario.create(usuario)
        .then(data => {
            let dataToSend = {...data.dataValues}; // Hacemos una copia del objeto de datos
            delete dataToSend.password; // Eliminamos la propiedad de la contraseña
            res.json(dataToSend); // Enviamos los datos del usuario como respuesta
        }).catch(err => {
            next(err); // Pasamos el error al middleware de manejo de errores
        });
    });
}

// Función para actualizar un usuario existente
exports.actualizar = (req, res, next) => {
    const id = req.params.id; // Obtenemos el ID del usuario de los parámetros de la solicitud
    // Verificar si se está actualizando la contraseña
    if (req.body.password) {
        req.body.password = bcrypt.hashSync(req.body.password, 8); // Ciframos la nueva contraseña si se está actualizando
    }
    db.Usuario.update(req.body, {
            where: {
                id: id // Actualizamos el usuario con el ID especificado
            }
        })
        .then(num => {
            if (num == 1) {
                res.send({
                    message: "Usuario actualizado" // Enviamos un mensaje de éxito si la actualización fue exitosa
                });
            } else {
                res.send({
                    message: "No se pudo actualizar el usuario" // Enviamos un mensaje de error si no se pudo actualizar el usuario
                });
            }
        }).catch(err => {
            next(err); // Pasamos el error al middleware de manejo de errores
        });
}

// Función para eliminar un usuario
exports.eliminar = (req,res, next) =>{
```

```
    const id = req.params.id; // Obtenemos el ID del usuario de los parámetros de la solicitud
    db.Usuario.destroy({
        where: {id: id} // Eliminamos el usuario con el ID especificado
    })
    .then(num => {
        if(num == 1){
            res.send({
                message: "Usuario eliminado" // Enviamos un mensaje de éxito si la eliminación fue exitosa
            });
        }else{
            res.send({
                message: "No se pudo eliminar el usuario" // Enviamos un mensaje de error si no se pudo eliminar el usuario
            });
        }
    }).catch(err => {
        next(err); // Pasamos el error al middleware de manejo de errores
    });
}

// Función para manejar el inicio de sesión de los usuarios
exports.login = (req,res, next) => {
    db.Usuario.findOne({
        where: {
            dni: req.body.dni // Buscamos un usuario con el DNI proporcionado en el cuerpo de la solicitud
        }
    })
    .then(usuario => {
        if(!usuario){
            res.status(404).send({
                message: "Usuario no encontrado" // Enviamos un mensaje de error si no se encuentra el usuario
            });
            return;
        }
        const passwordValido = bcrypt.compareSync(req.body.password, usuario.password); // Verificamos la contraseña proporcionada
        if(!passwordValido){
            res.status(401).send({
                message: "Contraseña incorrecta" // Enviamos un mensaje de error si la contraseña es incorrecta
            });
            return;
        }
        const token = jwt.sign({id: usuario.id}, llave, {
            expiresIn: '1h' // Generamos un token de autenticación que expira en 24 horas
        });

        let usuarioToSend = {...usuario.dataValues}; // Hacemos una copia del objeto de datos del usuario
        delete usuarioToSend.password; // Eliminamos la propiedad de la contraseña

        res.json({
            usuario: usuarioToSend, // Enviamos los datos del usuario (sin la contraseña) como respuesta
            token: token // Enviamos el token como respuesta
        });
    }).catch(err => {
        logger.error('Error al iniciar sesión: ', err); // Imprimimos el error en el log
```

```
160         next(err); // Pasamos el error al middleware de manejo de errores
161     });
162 }
```