

Controllers\Plan.controller.js

```
1 // Importamos el modelo de la base de datos
2 const db = require('../Models');
3
4 // Función para obtener la lista de todos los planes
5 exports.lista = (req, res, next) =>{
6   // Usamos el método findAll de Sequelize para obtener todos los planes
7   db.Plan.findAll()
8   .then(Planes => {
9     // Si la operación es exitosa, devolvemos los planes como respuesta JSON
10     res.json(Planes);
11   }).catch(err => {
12     next(err); // Pasamos el error al middleware de manejo de errores
13   });
14 }
15
16 // Función para filtrar planes por un campo específico
17 exports.filtrar = (req, res, next) =>{
18   // Obtenemos el campo y el valor de los parámetros de la solicitud
19   const campo = req.params.campo;
20   const valor = req.params.valor;
21   // Buscamos todos los planes que coinciden con el valor en el campo especificado
22   db.Plan.findAll({
23     where: {
24       [campo]: valor
25     }
26   })
27   .then(Planes => {
28     // Si la operación es exitosa, devolvemos los planes como respuesta JSON
29     res.json(Planes);
30   }).catch(err => {
31     next(err); // Pasamos el error al middleware de manejo de errores
32   });
33 }
34
35 // Función para crear un nuevo plan
36 exports.nuevo = (req, res, next) =>{
37   // Verificamos que los datos necesarios estén presentes en el cuerpo de la solicitud
38   if(!req.body.codigo || !req.body.descripcion){
39     // Si faltan datos, enviamos un mensaje de error con un código de estado 400
40     res.status(400).send({
41       message: "Faltan datos"
42     });
43     return;
44   }
45   // Creamos un objeto con los datos del nuevo plan
46   const Plan = {
47     codigo: req.body.codigo,
48     descripcion: req.body.descripcion,
49     id_ciclo: req.body.id_ciclo
50   }
51   // Usamos el método create de Sequelize para crear el nuevo plan
52   db.Plan.create(Plan)
53   .then(data => {
54     // Si la operación es exitosa, devolvemos los datos del nuevo plan como respuesta
55     res.json(data);
56   }).catch(err => {
```

```

57     next(err); // Pasamos el error al middleware de manejo de errores
58   });
59 }
60
61 // Función para actualizar un plan existente
62 exports.actualizar = (req, res, next) =>{
63   // Obtenemos el ID del plan de los parámetros de la solicitud
64   const id = req.params.id;
65   // Usamos el método update de Sequelize para actualizar el plan
66   db.Plan.update(req.body,{
67     where: {id_plan: id}
68   })
69   .then(num => {
70     // Si la operación es exitosa, enviamos un mensaje de éxito
71     if(num == 1){
72       res.send({
73         message: "Plan actualizado"
74       });
75     }else{
76       // Si no se pudo actualizar el plan, enviamos un mensaje de error
77       res.send({
78         message: "No se pudo actualizar el Plan"
79       });
80     }
81   }).catch(err => {
82     next(err); // Pasamos el error al middleware de manejo de errores
83   });
84 }
85
86 // Función para eliminar un plan existente
87 exports.eliminar = (req, res, next) =>{
88   // Obtenemos el ID del plan de los parámetros de la solicitud
89   const id = req.params.id;
90   // Imprimimos un mensaje en la consola indicando que vamos a eliminar el plan
91   console.log(`Eliminar Plan con id: ${id}`);
92   // Usamos el método destroy de Sequelize para eliminar el plan
93   db.Plan.destroy({
94     where: {id_plan: id}
95   })
96   .then(num => {
97     // Si la operación es exitosa, enviamos un mensaje de éxito
98     if(num == 1){
99       res.send({
100         message: "Plan eliminado"
101       });
102     }else{
103       // Si no se pudo eliminar el plan, enviamos un mensaje de error
104       res.send({
105         message: "No se pudo eliminar el Plan"
106       });
107     }
108   }).catch(err => {
109     next(err); // Pasamos el error al middleware de manejo de errores
110   });
111 }

```