**PRUEBA DE CAJA BLANCA**

Jurado Junior, Lituma Jhonatan, Román Ivette

Departamento de Ciencias de la Computación, Universidad de las Fuerzas Armadas ESPE

8311: Análisis y Diseño de Software

Ing. Jenny Ruiz

21 de enero de 2023

**PRUEBAS CAJA BLANCA LOGIN**

 handleSubmit = (event) => {

    event.preventDefault();

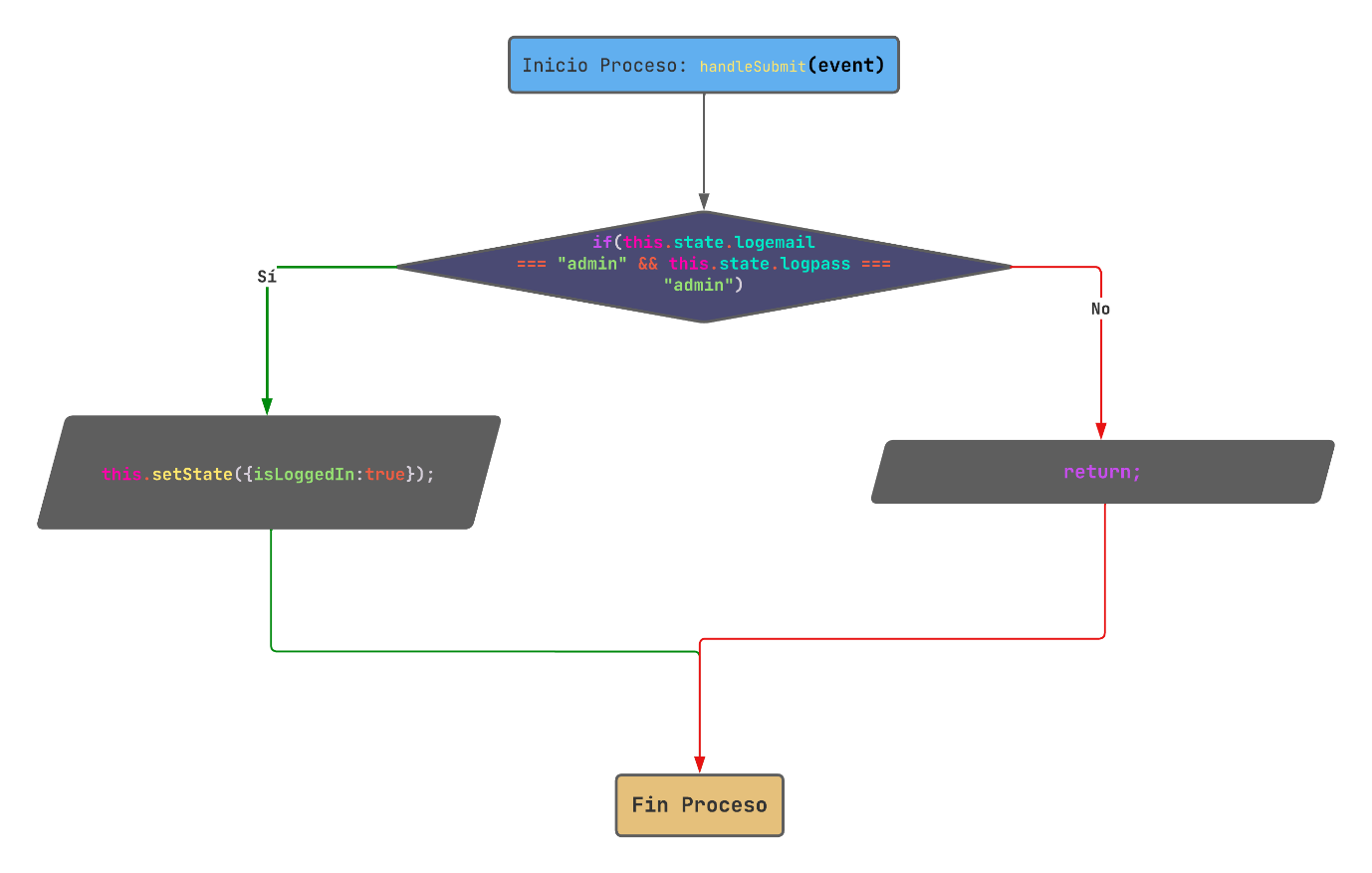
    if (this.state.logemail === "admin" && this.state.logpass === "admin") {

      this.setState({ isLoggedIn: true });

    }

  };

**DIAGRAMA DE FLUJO**

****

**DIAGRAMA DE GRAFOS**

**Imagen en blanco y negro

Descripción generada automáticamente con confianza baja**

**RUTAS:**

**R1: 1, 2, 4**

**R2: 1, 3, 4**

**Complejidad Ciclomática**

**E: Número de aristas**

**N: Número de nodos**

**P: Número de nodos predicado**

**V(G) = E – N + 2**

**V(G) = 4 - 4 + 2**

**V(G) = 2**

**V(G) = P + 1**

**V(G) = 1 nodo predicado +1 = 2**

**PRUEBAS CAJA BLANCA GESTIONAR PRODUCTOS**

**PRUEBA CAJA BLANCA OBTENER LISTA DE ELEMENTOS DE LA DB**

**/\*\***

**\* Obtener la lista de la base de datos**

**\* @param {\*} req**

**\* @param {\*} res**

**\*/**

**const getItems = async (req, res) => {**

**try {**

**const data = await productsModel.find({});**

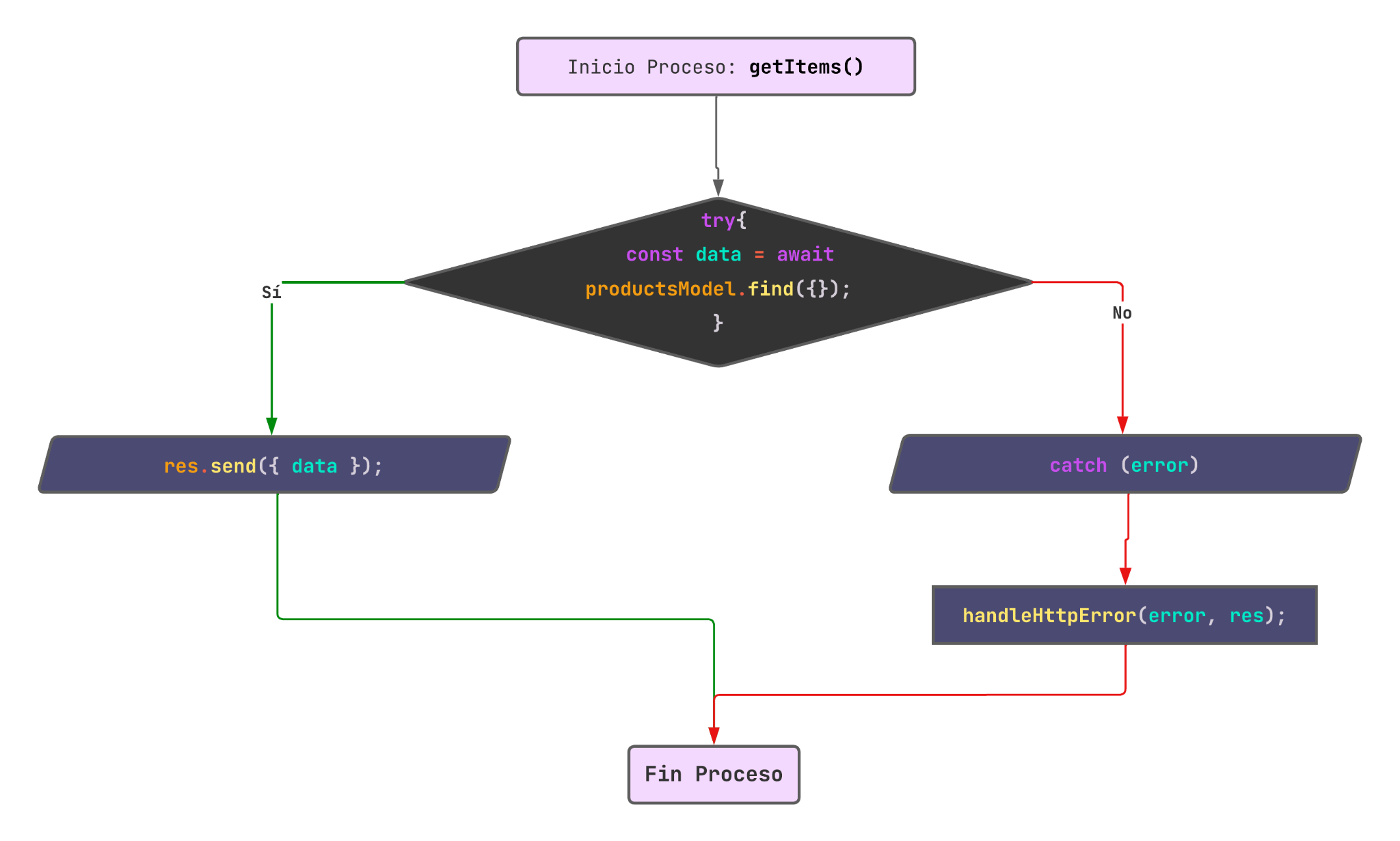
**res.send({ data });**

**} catch (error) {**

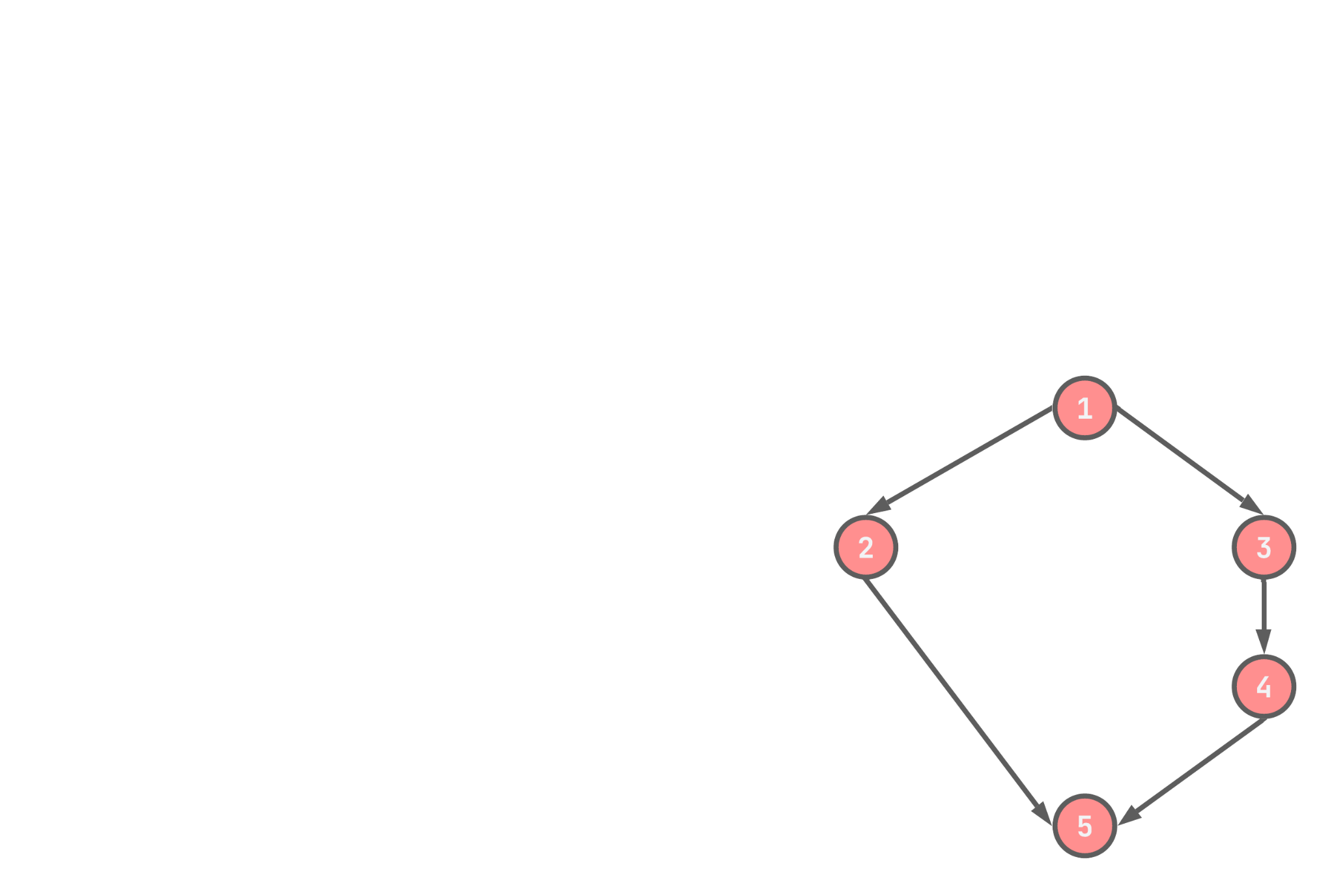
**handleHttpError(error, res);**

**}**

**};**

**Diagrama de flujo:**

**Diagrama de grafos:**

****

**RUTAS:**

**R1:** 1, 3, 4, 5

**R2:** 1, 2, 5

**Complejidad Ciclomática:**

**E:** 5 (try, await, send, catch, handleHttpError)

**N:** 5 (try, await, send, catch)

**P:** 1 (try)

**V(G)** = E - N + 2   
**V(G)** = 5 - 5 + 2   
**V(G)** = 2

**V(G)** = P + 1

**V(G)** = 1 + 1 = 2

La complejidad ciclomática del código es 2.

En conclusión, el código tiene una complejidad ciclomática de 2, lo que significa que hay 2 caminos diferentes a través del código. Esto indica que el código es bastante simple y fácil de entender y probar.

**/\*\***

**\* Crear Registro**

**\* @param {\*} req**

**\* @param {\*} res**

**\*/**

**const CreateItem = async (req, res) => {**

**try {**

**const body = req.body;**

**console.log(body);**

**const data = await productsModel.create(body);**

**res.send({ data });**

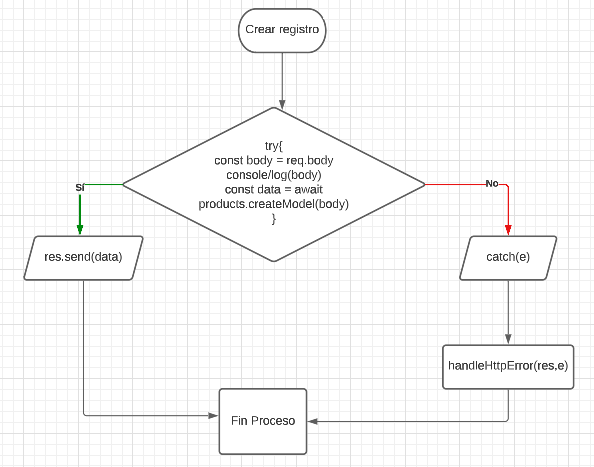
**} catch (error) {**

**handleHttpError(error, res);**

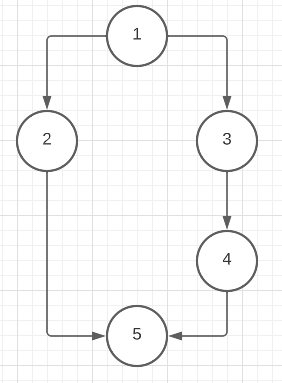
**}**

**}**

**Diagrama de flujo:**

****

**Diagrama de grafos:**

****

**RUTAS:**

**R1: 1, 2, 5**

**R2: 1, 3, 4, 5**

**Complejidad Ciclomática**

**E: Número de aristas**

**N: Número de nodos**

**P: Número de nodos predicado**

**V(G) = E – N + 2**

**V(G) = 5 - 5 + 2**

**V(G) = 2**

**V(G) = P + 1**

**V(G) = 1 nodo predicado +1 = 2**

**PRUEBA DE CAJA BLANCA DE ELIMINAR REGISTRO**

**/\*\***

**\* Eliminar registro**

**\* @param { \* } req**

**\* @param {\*} res**

**\*/**

**const deleteItem = async (req, res) => {**

**try {**

**const data = await productsModel.de1eteOne({**

**idProducto: req.params.idProducto,**

**});**

**res.send({ data });**

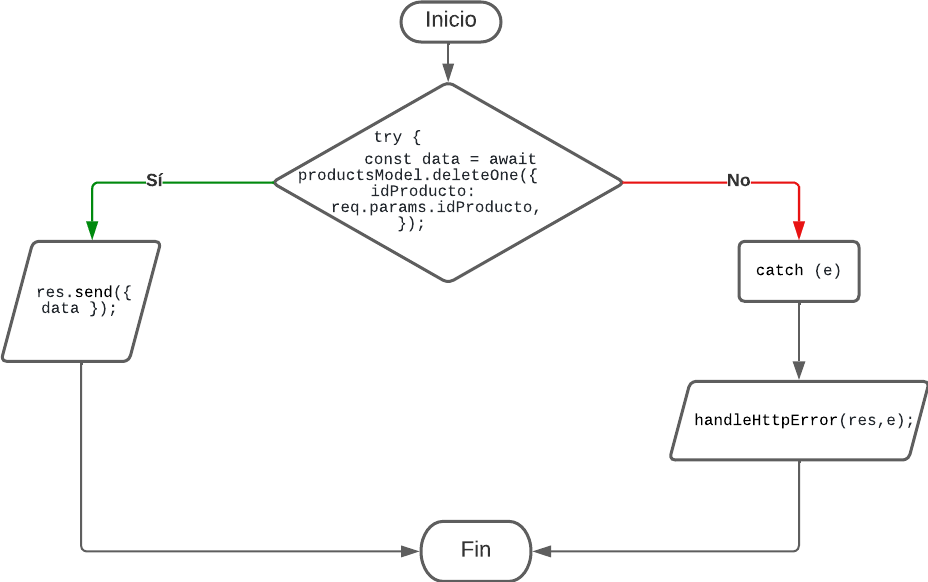
**} catch (e) {**

**handleHttpError(res, e);**

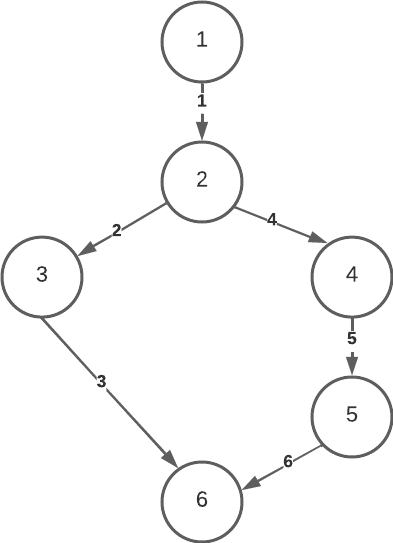
**}**

**};**

**DIAGRAMA DE FLUJO**

****

**DIAGRAMA DE GRAFOS**

****

**RUTAS:**

**R1:** 1, 2, 3, 6

**R2:** 1, 2, 4, 5, 6

**Complejidad Ciclomática**

**E: Número de aristas** 6

**N: Número de nodos** 6

**P: Número de nodos predicado** 1

**V(G) = E – N + 2A**

V(G) = 6 - 6 + 2 = 2

**V(G) = P + 1**

V(G) = 1 + 1= 2

**PRUEBA DE CAJA BLANCA DE ACTUALIZAR REGISTRO**

**/\*\***

**Actualizar registro**

**@param { \* } req**

**@param {\*} res**

**\*/**

**const updateItem = async (req, res) => {**

**try {**

**let updatedProduct = {**

**idProducto: req.body.idProduct,**

**nombre: req.body.nombre,**

**marca: req.body.marca,**

**modelo: req.body.modelo,**

**precio: req.body.precio,**

**caracteristicas: req.body.caracteristicas,**

**imagen: req.body.image,**

**cantidad: req.body.cantidad,**

**categoria: req.body.categoria,**

**};**

**const data = await productsModel.findOneAndUpdate(**

**{**

**idProducto: req.params.idProducto,**

**},**

**updatedProduct**

**);**

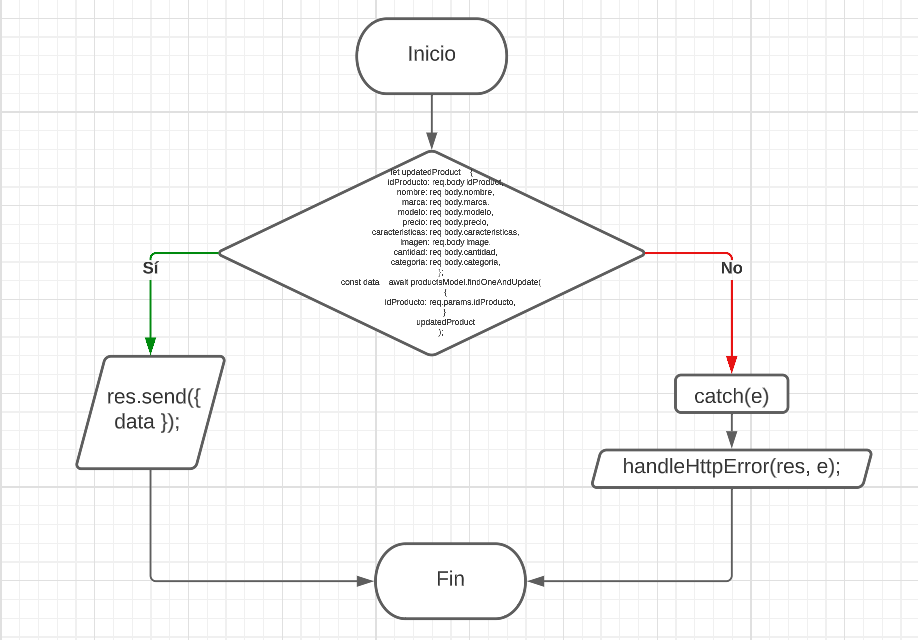
**res.send({ data });**

**} catch (e) {**

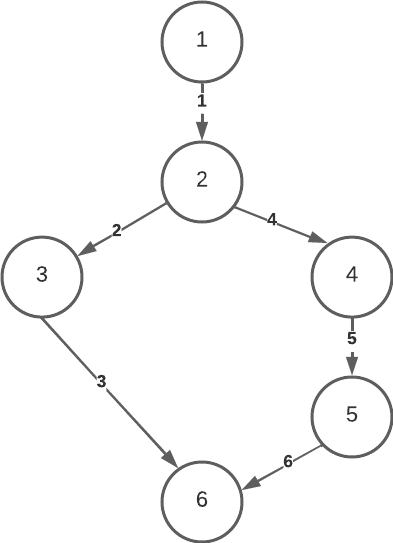
**handleHttpError(res, e);**

**}**

**};**

**DIAGRAMA DE FLUJO**

**DIAGRAMA DE GRAFOS**

****

**RUTAS:**

**R1:** 1, 2, 3, 6

**R2:** 1, 2, 4, 5, 6

**Complejidad Ciclomática**

**E: Número de aristas**

**N: Número de nodos**

**P: Número de nodos predicado**

**V(G) = E – N + 2**

V(G) = 6 - 6 + 2 = 2

**V(G) = P + 1**

V(G) = 1 + 1 = 2

**Conclusión:**

Las pruebas de caja blanca proporcionan una gran cantidad de información sobre el comportamiento interno de un sistema o aplicación. Al analizar el código ejecutado durante las pruebas, los desarrolladores pueden identificar problemas de rendimiento, bugs y otros problemas que podrían afectar la estabilidad y el rendimiento del sistema.

**PRUEBAS CAJA BLANCA GESTIONAR CATEGORÍAS**

**PRUEBA CAJA BLANCA CREAR CATEGORÍA**

**validate = () => {**

**const { form } = this.state;**

**const errors = {};**

***//Nombre***

**if (!this.validateLetters(form.nombre)) {**

**errors.nombre = 'No se permiten números.';**

**}**

**if (!form.nombre) {**

**errors.nombre = 'Campo requerido.';**

**}**

**this.setState({ errors });**

**return Object.keys(errors).length === 0;**

**};**

**postQuery = async () => {**

**if (this.validate()) {**

**this.axiosService.post('/', this.state.form).then((*response*) => {**

**this.modalInsert();**

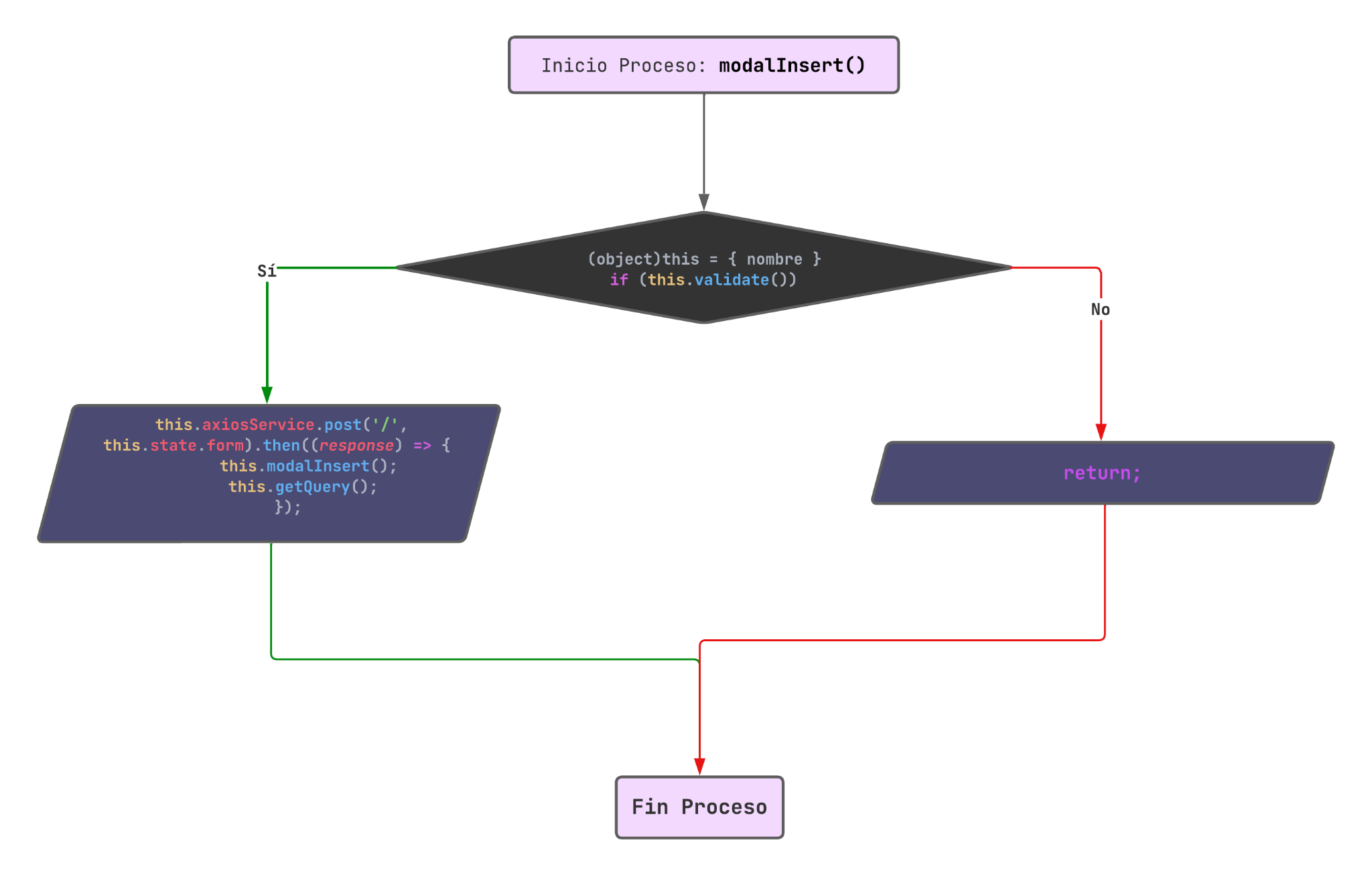
**this.getQuery();**

**});**

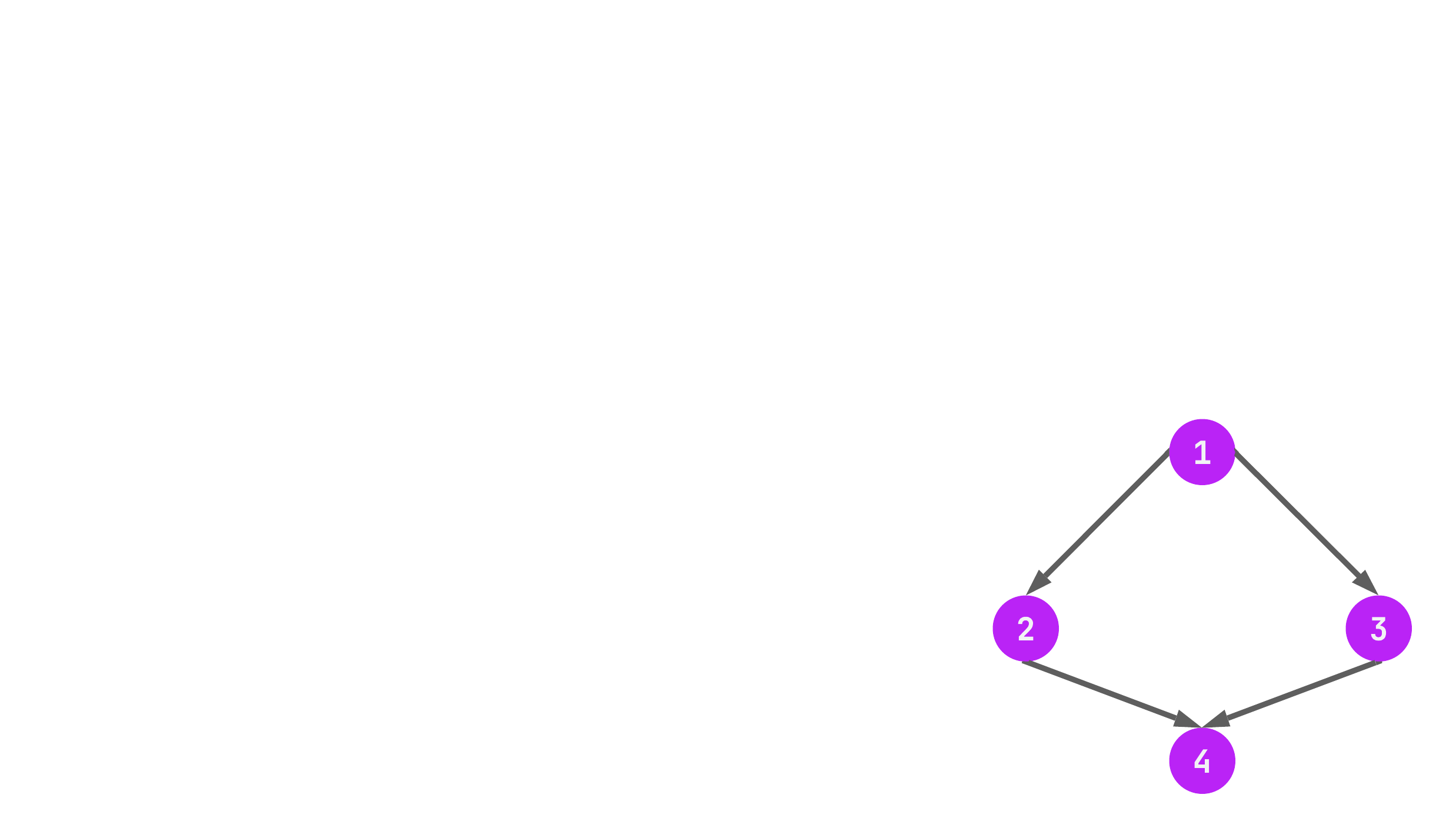
**}**

**};**

**Diagrama de flujo:**

****

**Diagrama de grafos:**

****

**RUTAS:**

**R1: 1, 2, 4**

**R2: 1, 3, 4**

**Complejidad Ciclomática**

**E: Número de aristas**

**N: Número de nodos**

**P: Número de nodos predicado**

**V(G) = E – N + 2**

**V(G) = 4 - 4 + 2**

**V(G) = 2**

**V(G) = P + 1**

**V(G) = 1 nodo predicado +1 = 2**

La complejidad ciclomática del código es 2.

En conclusión, el código tiene una complejidad ciclomática de 2, lo que significa que hay 2 caminos diferentes a través del código. Esto indica que el código es bastante simple y fácil de entender y probar.

**PRUEBA CAJA BLANCA EDITAR CATEGORÍA**

**putQuery = () => {**

**if (this.validate()) {**

**this.axiosService**

**.put('/' + this.state.form.idCategoria, this.state.form)**

**.then((response) => {**

**this.modalEdit();**

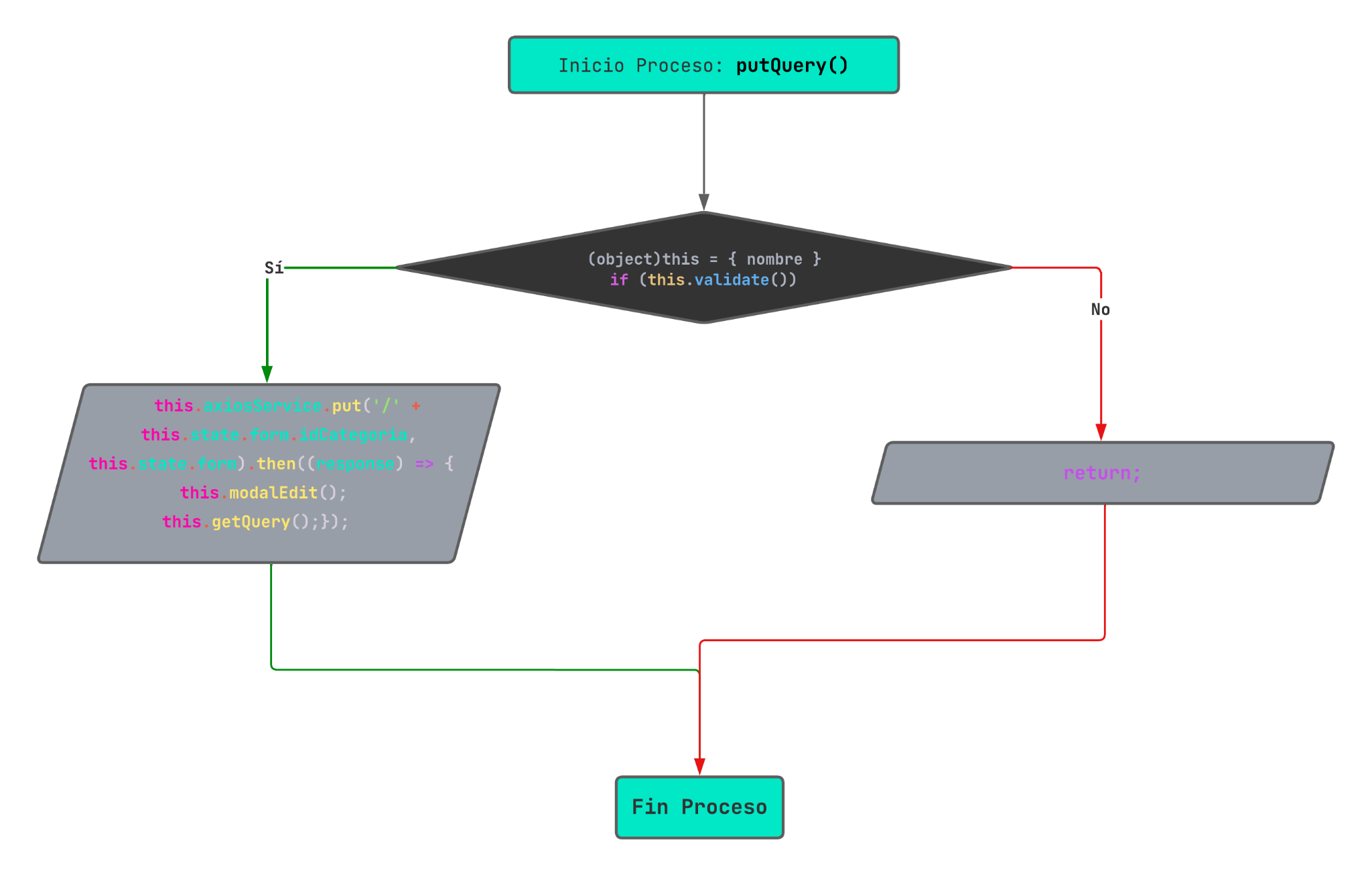
**this.getQuery();**

**});**

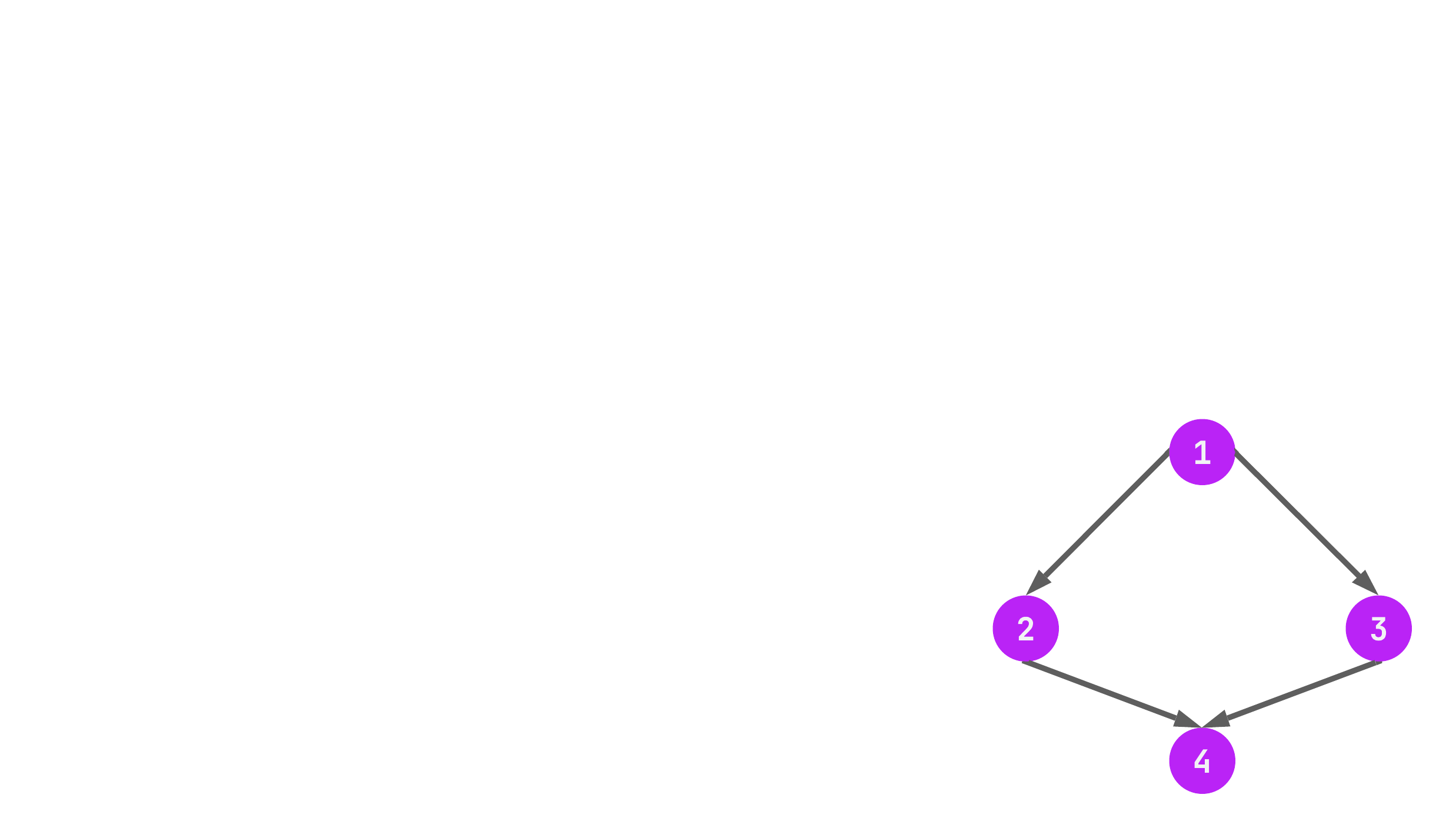
**}**

**};**

**Diagrama de flujo:**

****

**Diagrama de grafos:**

****

**RUTAS:**

**R1: 1, 2, 4**

**R2: 1, 3, 4**

**Complejidad Ciclomática**

**E: Número de aristas**

**N: Número de nodos**

**P: Número de nodos predicado**

**V(G) = E – N + 2**

**V(G) = 4 - 4 + 2**

**V(G) = 2**

**V(G) = P + 1**

**V(G) = 1 nodo predicado +1 = 2**

**PRUEBA DE CAJA BLANCA DE ELIMINAR CATEGORÍA**

**deleteQuery = () => {**

**this.axiosService**

**.delete('/' + this.state.form.idCategoria)**

**.then((response) => {**

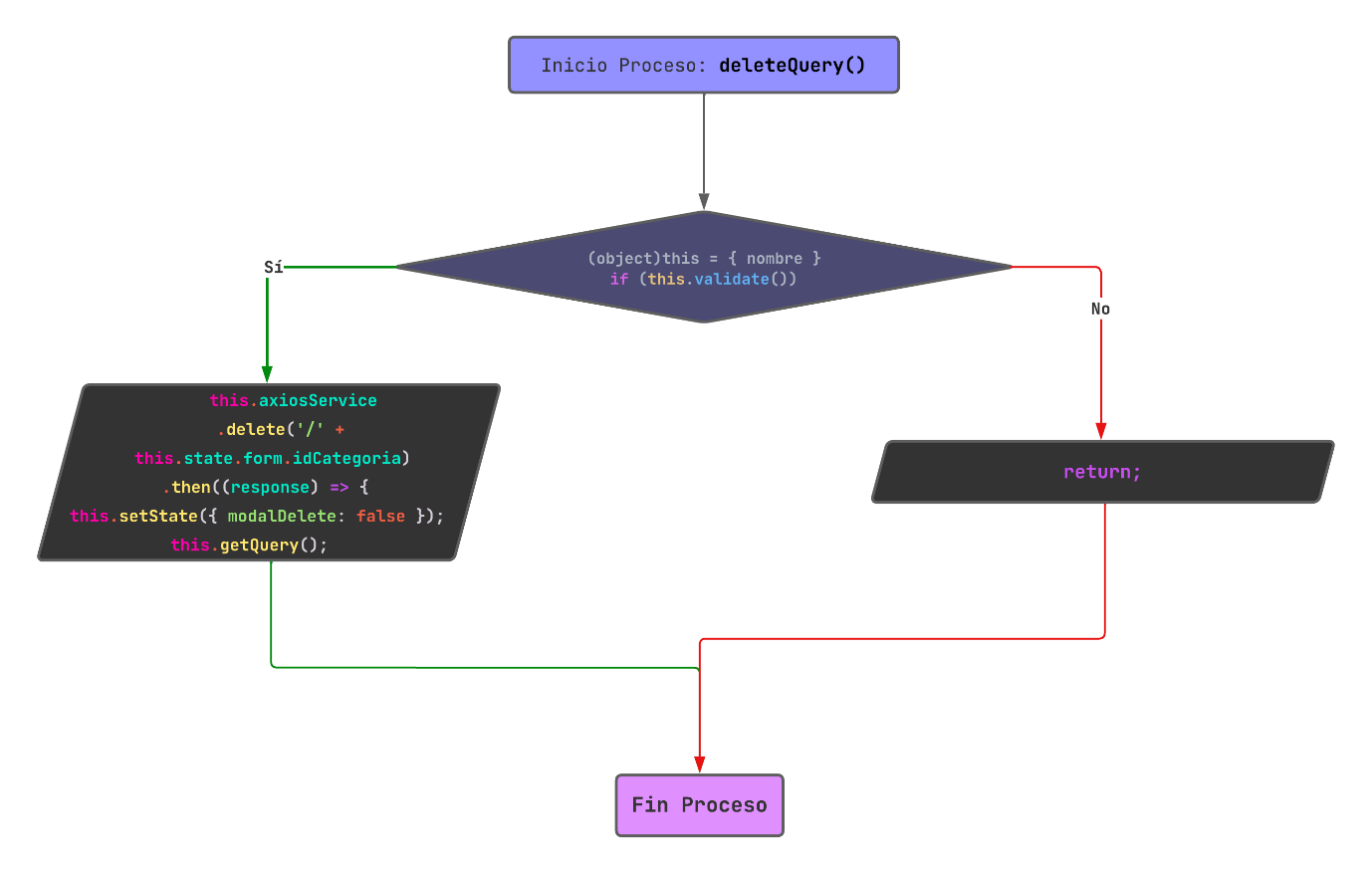
**this.setState({ modalDelete: false });**

**this.getQuery();**

**});**

**};**

**DIAGRAMA DE FLUJO**

****

**DIAGRAMA DE GRAFOS**

**Imagen en blanco y negro

Descripción generada automáticamente con confianza baja**

**RUTAS:**

**R1: 1, 2, 4**

**R2: 1, 3, 4**

**Complejidad Ciclomática**

**E: Número de aristas**

**N: Número de nodos**

**P: Número de nodos predicado**

**V(G) = E – N + 2**

**V(G) = 4 - 4 + 2**

**V(G) = 2**

**V(G) = P + 1**

**V(G) = 1 nodo predicado +1 = 2**