|  |
| --- |
| Prueba de Caja Blanca |
|  |
|  |
|  |
| **Integrantes:**  David Muñoz  Alan Bermúdez  Nayeli Arellano  **Fecha 2022-01-26** |
|  |

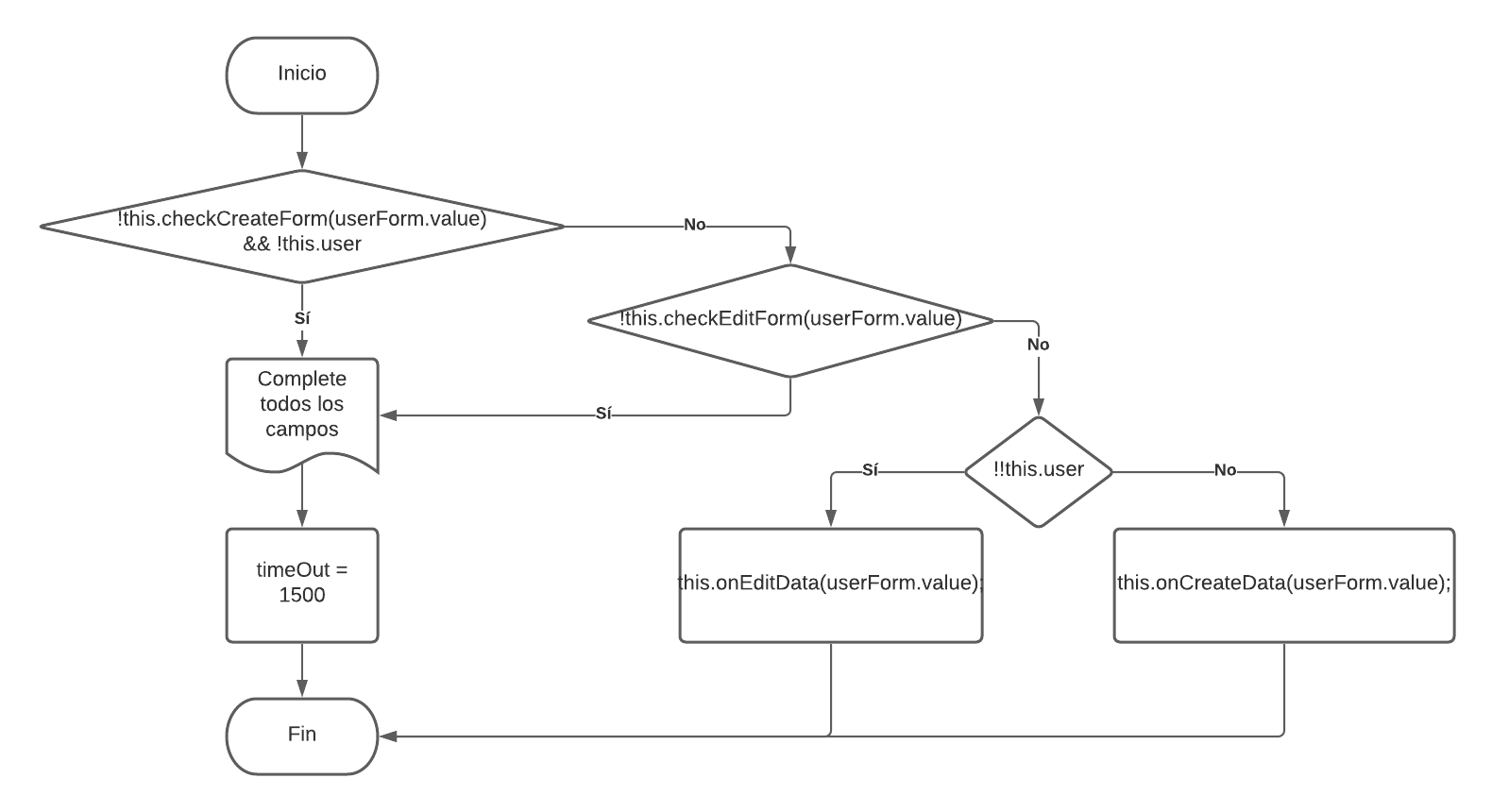
**MÓDULO: Creación de Usuario**

**FUNCION:**

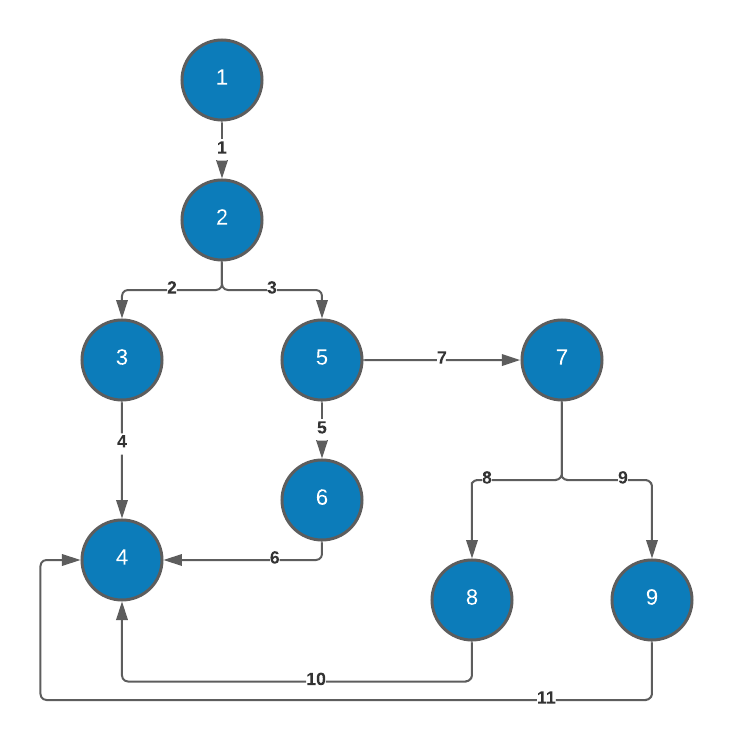
**Código fuente:**

****

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,2,4 | Verifica si los campos y el usuario no estan vacios | Probar |
| **B:** 1,3,5,6 | Verifica si los campos no estan vacios | Probar |
| **C:** 1,3,7,8,10 | Si el usuario existe edita los datos | Probar |
| **D:** 1,3,7,9,11 | Si el usuario no existe lo crea | Probar |

**Complejidad Ciclomática**:

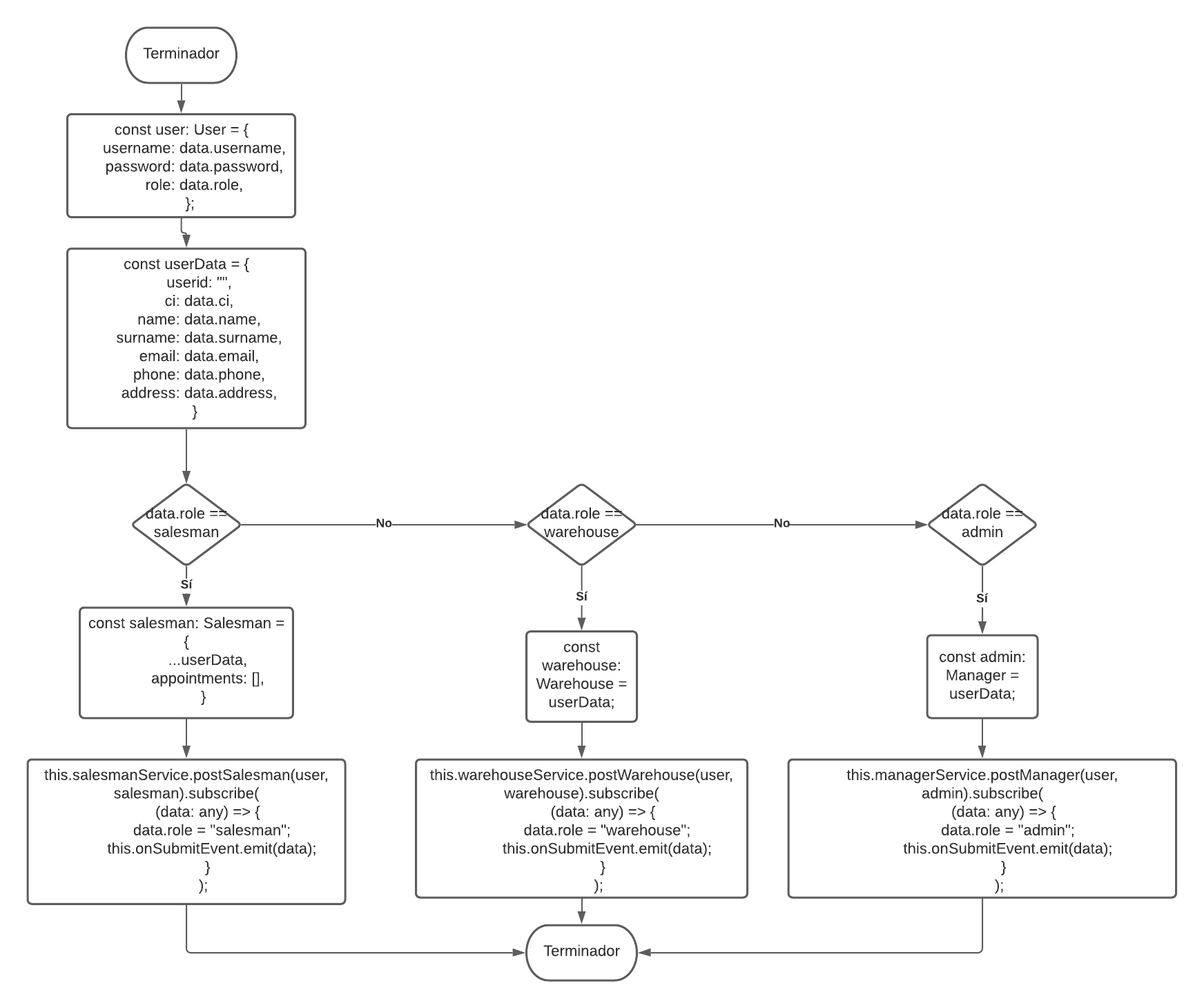
**MODULO: Asignación de Perfiles**

**FUNCION:**

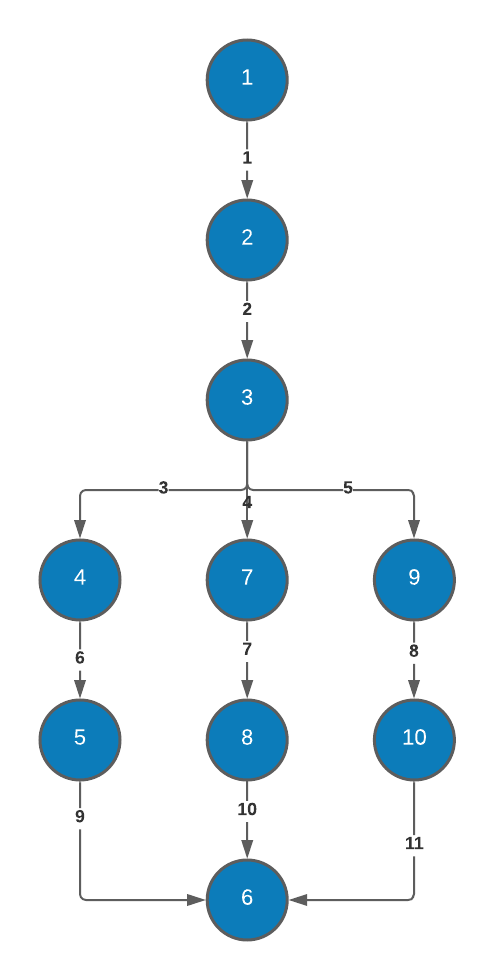
**Código fuente:**

****

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,2,3,6,9 | Asigna el rol vendedor | Probar |
| **B:** 1,2,4,7,10 | Asigna el rol bodeguero | Probar |
| **C:** 1,2,5,8,11 | Asigna el rol gerente | Probar |

**Complejidad Ciclomática**:

**MODULO: Ingreso al sistema**

**FUNCION:**

**Código fuente:**

  onChange(*value*: any) {

    if (*value*.type === 'checkbox') {

      this.input = { ... this.input, [*value*.name]: *value*.checked };

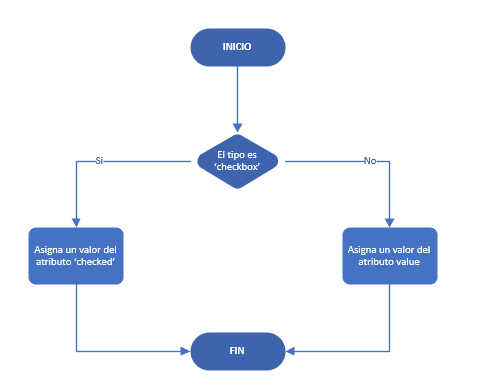
    } else {

      this.input = { ... this.input, [*value*.name]: *value*.value };

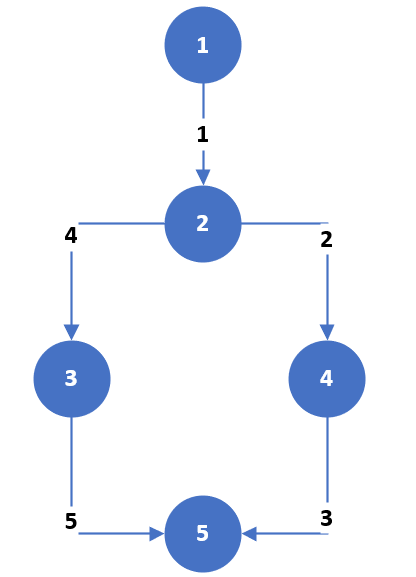
    }

  }

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,2,4,5 | Asigna valores a variables no booleanas | Probar |
| **B:** 1,2,3,5 | Asigna valores a variables booleanas | Probar |

**Complejidad Ciclomática**:

**MODULO: Modificar Contraseña**

**FUNCION: Reducir un paso**

**Código fuente:**

  decreaseStep() {

    this.stepCounter--;

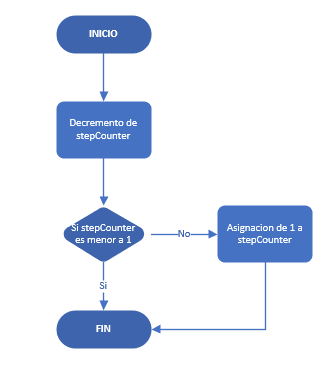
    if (this.stepCounter < 1) {

      this.stepCounter = 1;

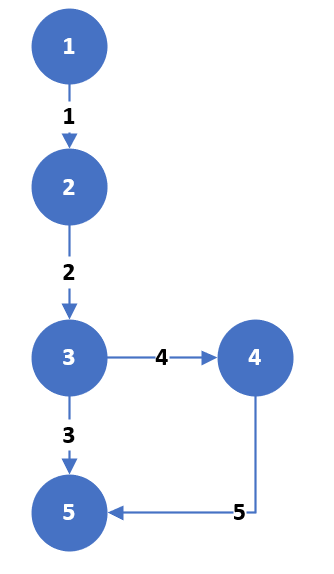
    }

  }

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,2,5 | Decremento del stepCounter | Probar |
| **B:** 1,2,3,4,5 | Impide que el stepCounter se reduzca de 1 | Probar |

**Complejidad Ciclomática**:

**FUNCIÓN: Actualizar Contraseña**

**Código fuente:**

  onSubmitForgetPassword(*forgetPasswordForm*: NgForm) {

    if (this.stepCounter === 1) {

      this.forgetPasswordInput.username = *forgetPasswordForm*.value.username;

      this.checkUserExist(this.forgetPasswordInput.username);

    } else if (this.stepCounter === 2) {

      this.forgetPasswordInput.newPassword = *forgetPasswordForm*.value.password;

      this.forgetPasswordInput.confirmPassword = *forgetPasswordForm*.value.confirm;

      if (!this.checkPassword(this.forgetPasswordInput.newPassword, this.forgetPasswordInput.confirmPassword)) {

        return;

      }

      this.userService.updatePassword(this.forgetPasswordInput.username, this.forgetPasswordInput.newPassword)

        .subscribe((*res*: any) => {

          this.toast.success("Password updated", "Success",

            {

              timeOut: 3000,

            });

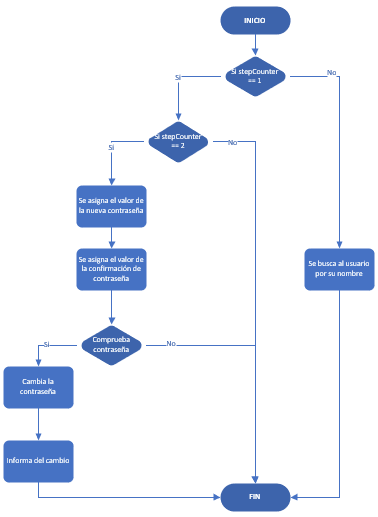
          this.modalClose();

        });

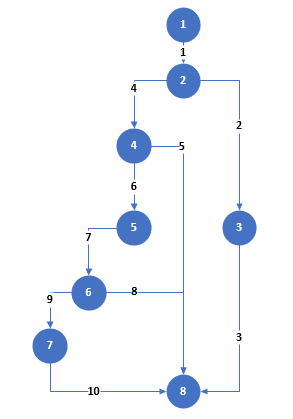
    }

  }

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,2,3,8 | Comprueba la existencia del usuario | Probar |
| **B:** 1,2,4,8 | Verifica si se encuentra en el segundo paso | Probar |
| **B:** 1,2,4,5,6,8 | Las contraseñas no son correctas o no coinciden | Probar |
| **B:** 1,2,4,5,6,7,8 | Cambia la contraseña | Probar |

**Complejidad Ciclomática**:

**FUNCION: Comprobar la existencia del Usuario**

**Código fuente:**

  checkUserExist(*username*: string) {

    this.userService.getUserByUsername(*username*)

      .subscribe((*res*: any) => {

        if (!*res*.\_id) {

          this.toast.error("User not found", "Error", {

            timeOut: 3000,

          });

          this.stepCounter = 1;

        } else {

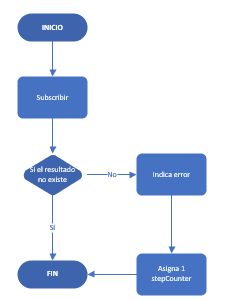
          this.increaseStep();

        }

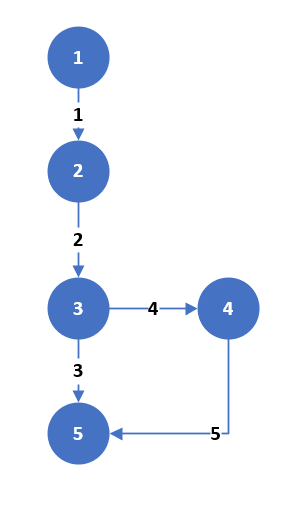
      });

  }

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

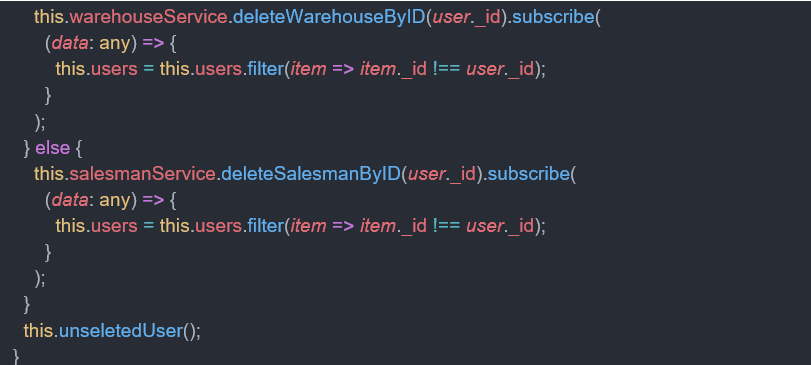
| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,2,3,5 | El usuario existe | Probar |
| **B:** 1,2,3,4,5 | El usuario no existe | Probar |

**Complejidad Ciclomática**:

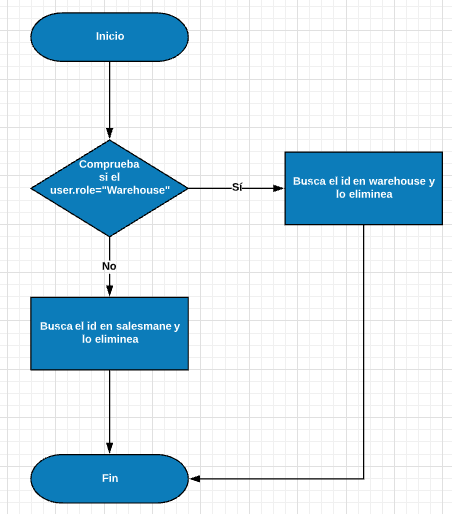
**MODULO: Eliminar Usuario**

**FUNCION:** Eliminar usuario de la base de datos

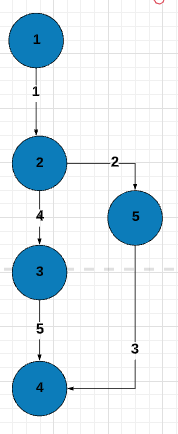
**Código fuente:**

****

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

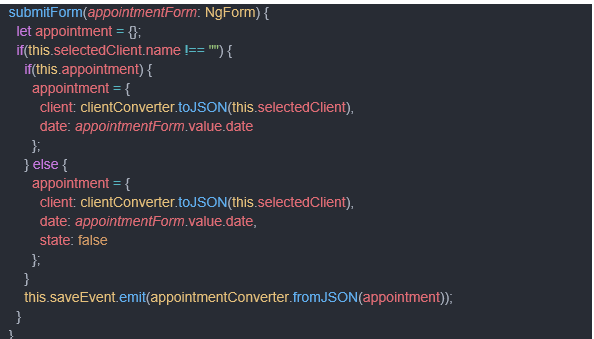
| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,2,3,4 | El usuario salesman se elimina | Probar |
| **B:** 1,2,3 | El usuario warehouse se elimina | Probar |

**Complejidad Ciclomática**:

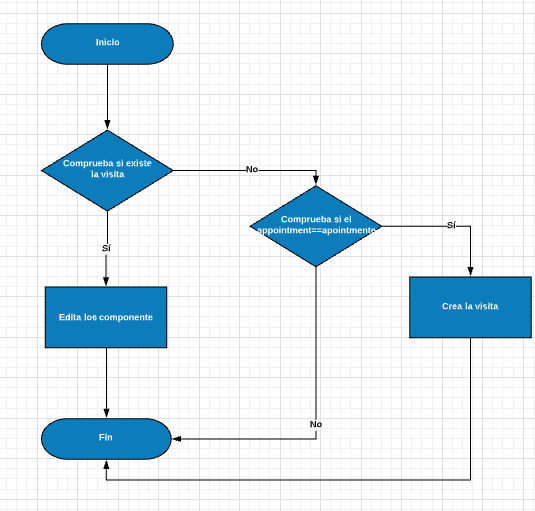
**MODULO: Registrar Visita / Modificar Visita**

**FUNCION:**

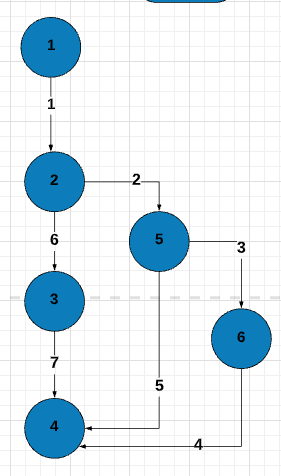
**Código fuente:**

****

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

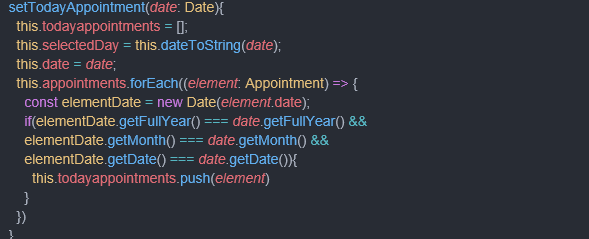
| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,6,7 | La visita no existe se registra una nueva | Probar |
| **B:** 1,2,5 | la visita si existe se modifica | Probar |
| **C:** 1,2,3,4 | La visita no se selecciona | Probar |

**Complejidad Ciclomática**:

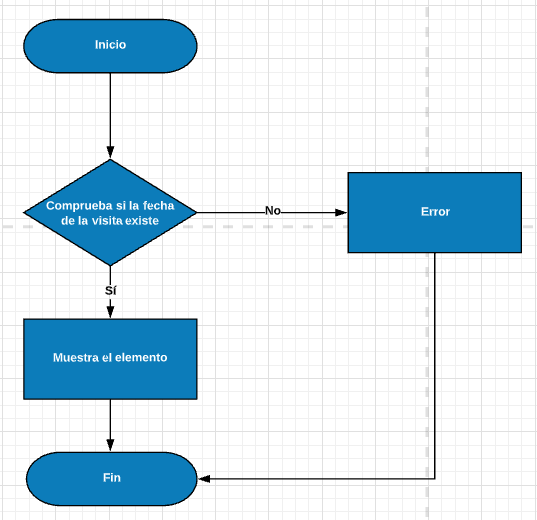
**MODULO: Mostrar Visita**

**FUNCION:** Indica si la visita existe

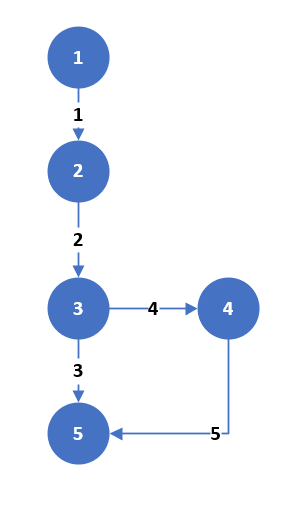
**Código fuente:**

****

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,2,3,5 | La visita existe | Probar |
| **B:** 1,2,3,4,5 | La visita no existe | Probar |

**Complejidad Ciclomática**:

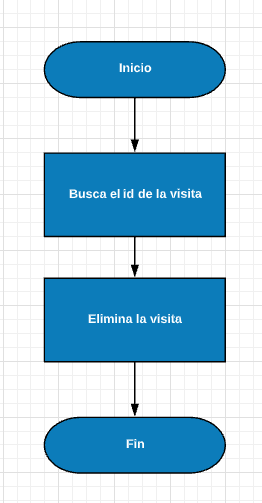
**MODULO: Eliminar Visita**

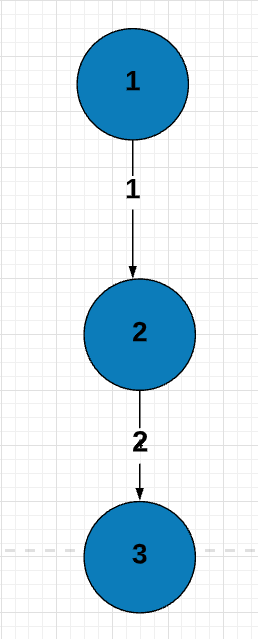
**FUNCION:** Elimina una visita

**Código fuente:**

****

**Diagrama de Flujo:**



**Grafo:**

**Rutas:**

| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,2,3 | Indica que la visita se elimino | Probar |

**Complejidad Ciclomática**:

**MODULO: Registrar Cliente y Modificar Cliente**

**FUNCION: onSubmit**

**Código Fuente:**

onSubmit(*clientForm*: NgForm) {

if (!this.checkCreateForm(*clientForm*.value)) {

this.toast.error("Complete todos los campos", 'Error', {

timeOut: 1500,

});

return;

}

if (!!this.client) {

this.onEditData(*clientForm*.value);

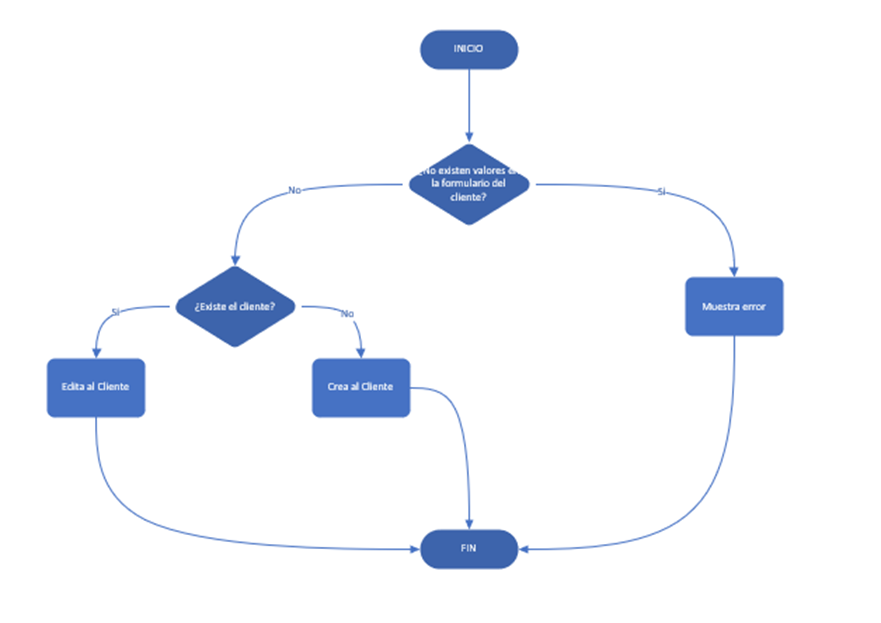
} else {

this.onCreateData(*clientForm*.value);

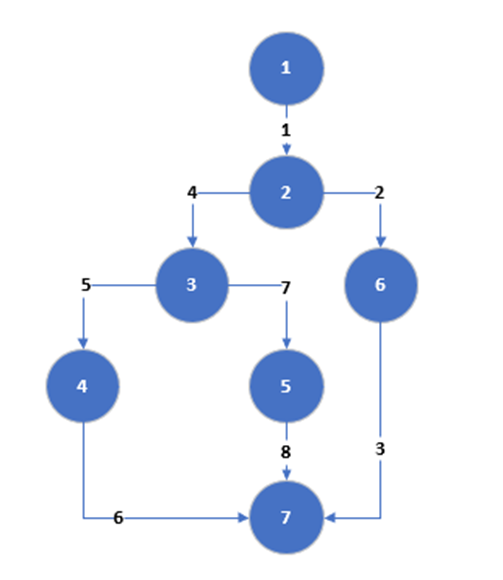
}

}

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,2,6,7 | Indica que el formulario está incompleto | Probar |
| **B:** 1,2,3,5,7 | Registra un nuevo cliente | Probar |
| **C: 1,2,3,4,7** | Edita un cliente existente | Probar |

**Complejidad Ciclomática**:

A*ristas=8*

*Nodos=7*

*VG=Aristas-Nodos+2*

*VG=8-7+2*

*VG=3*

**MODULO: Eliminar Cliente**

**Funcion: onDelete**

**Codigo Fuente:**

**onDelete(*client*: any) {**

**this.clientService.deleteClientByID(*client*.\_id).subscribe(**

**(*data*: any) => {**

**this.clients = this.clients.filter(*item* => *item*.\_id !== *client*.\_id);**

**}**

**);**

**this.unselectedClient();**

**}**

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,2 | Indica que el usuario se eliminó | Probar |

**Complejidad Ciclomática:**

A*ristas=2*

*Nodos=3*

*VG=Aristas-Nodos+2*

*VG=2-3+2*

*VG=1*

**MODULO: Ingresar Producto y Modificar Producto**

**Funcion: onSubmit**

**Codigo Fuente:**

**onSubmit(*clientForm*: NgForm) {**

**if (!this.checkCreateForm(*clientForm*.value) && !this.client) {**

**this.toast.error("Complete todos los campos", 'Error', {**

**timeOut: 1500,**

**});**

**return;**

**} else if (!this.checkEditForm(*clientForm*.value)) {**

**this.toast.error("Complete todos los campos", 'Error', {**

**timeOut: 1500,**

**});**

**return;**

**}**

**if (!!this.client) {**

**this.onEditData(*clientForm*.value);**

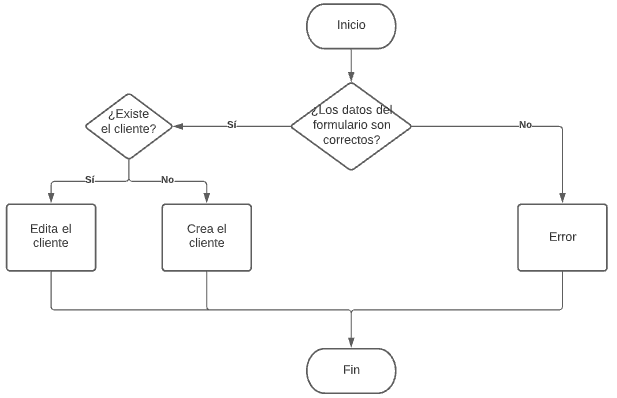
**} else {**

**this.onCreateData(*clientForm*.value);**

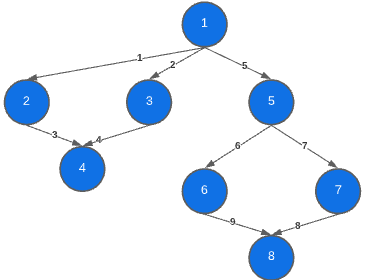
**}**

**}**

**Diagrama de Flujo:**

****

**Grafo:**

****

**Rutas:**

| **Camino** | **Representa** | **Decisión** |
| --- | --- | --- |
| **A:** 1,3 | Indica que los campos llenados no son correctos y el cliente no existe | Probar |
| **B:** 2,4 | Indica que los campos llenados no son correctos | Probar |
| **C:** 5,6,9 | Indica que el cliente existe y por lo tanto va a ser editado | Probar |
| **D:** 5,7,8 | Indica que el cliente no existe y que se creara el mismo | Probar |

**Complejidad Ciclomática:**

A*ristas=9*

*Nodos=8*

*VG=Aristas-Nodos+2*

*VG=9-8+2*

*VG=3*