

Pues estoy haciendo el botón de borrar cuenta. He copiado el botón de editar y he hecho modificaciones me faltan algunos detalles que podría buscar de otras pestañas, pero prefiero dejartelo a ti que lo tendrás más rápido.

```

<div
class="modal-overlay" *ngIf="isVisible" (click)="onClose()"> <div class="modal-
content" (click)="$event.stopPropagation()"> <div class="modal-header">
<h2>Editar Perfil</h2> <button class="close-btn"
(click)="onClose()">&times;</button> </div> <div class="modal-body"> <form
(ngSubmit)="onSubmit()"> <div class="form-group mb-3"> <label class="form-
label">Nuevo nombre</label> <input type="email" class="form-control"
[(ngModel)]="email" name="email" placeholder="Dejar vacío para no cambiar"
> <div comprobar que no es email + mensaje personalizado </div> <div
class="form-group mb-3"> <label class="form-label">Contraseña</label> <input
type="password" class="form-control" [(ngModel)]="Password"
name="Password" placeholder="Dejar vacío para no cambiar" > <div
class="modal-footer"> <button type="button" class="btn btn-outline-secondary"
(click)="onClose()">Cancelar</button> <button type="submit" class="btn btn-
primary" [disabled]= > <span *ngIf="!isSubmitting">Borrar Cuenta</span>
<span *ngIf="isSubmitting"> <span class="spinner-border spinner-border-sm"
role="status" aria-hidden="true"></span> Guardando... </span> </button>
</div> </form> </div> </div> </div>
import { Component, Inject } from
'@angular/core'; import { MAT_DIALOG_DATA, MatDialogRef } from
'@angular/material/dialog'; import { ApiService } from
'../shared/services/api.service'; @Component({ selector: 'app-edit-profile-
modal', standalone: false, templateUrl: './del-profile-modal.component.html',
styleUrls: ['./del-profile-modal.component.scss'] }) export class
DelProfileModalComponent { email: string = ''; Password: string = '';
isSubmitting = false; isVisible = true; passwordTouched = false; constructor(
private apiService: ApiService, private dialogRef:
MatDialogRef<DelProfileModalComponent>, @Inject(MAT_DIALOG_DATA)
public data: { currentName: string } ) {} onSubmit(): void { this.isSubmitting =
true; const deleteData: any = {}; //aquí debería ir lógica de comparación de email y
bcrypt con petición al método //get por ID o EMAIL de la persona que quiere
borrar. Se puede obtener de ambas formas //Recomiendo obtener datos del token
JWT y comprar tanto email como password bcrypt. if (this.name && this.name
!== this.data.currentName) { updateData.name = this.name; } if
(this.newPassword) { updateData.password = this.newPassword; } if
(Object.keys(updateData).length === 0) { this.dialogRef.close(); return; }
this.apiService.delete('users/me', deleteData).subscribe({ next: () => {
this.dialogRef.close({ success: true, name: deleteData.name ||
this.data.currentName }); }, error: (err) => { this.isSubmitting = false;
console.error('Error updating profile:', err); alert('Error updating profile: ' +
(err.error?.message || 'Please try again')); } }); } onClose(): void {
this.dialogRef.close(); } }

```

Implementación del Modal para Borrar Cuenta

Aquí tienes una implementación completa del modal para borrar cuenta, con validaciones y seguridad:

Componente TypeScript (del-profile-modal.component.ts)

typescript

```
import { Component, Inject } from '@angular/core';
import { MAT_DIALOG_DATA, MatDialogRef } from
'@angular/material/dialog';
import { ApiService } from '../.../shared/services/api.service';
import { AuthService } from '../.../shared/services/auth.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-del-profile-modal',
  templateUrl: './del-profile-modal.component.html',
  styleUrls: ['./del-profile-modal.component.scss']
})
export class DelProfileModalComponent {
  email: string = '';
  password: string = '';
  isSubmitting = false;
  isVisible = true;
  errorMessage: string | null = null;

  constructor(
    private apiService: ApiService,
    private authService: AuthService,
    private router: Router,
    private dialogRef: MatDialogRef<DelProfileModalComponent>,
    @Inject(MAT_DIALOG_DATA) public data: { currentEmail: string }
  ) {}

  onSubmit(): void {
    if (!this.email || !this.password) {
      this.errorMessage = 'Por favor ingresa tu email y contraseña';
      return;
    }

    if (this.email !== this.data.currentEmail) {
      this.errorMessage = 'El email no coincide con tu cuenta';
      return;
    }

    this.isSubmitting = true;
    this.errorMessage = null;

    // Primero verificamos las credenciales
    this.authService.login(this.email, this.password).subscribe({
      next: () => {
        // Si las credenciales son correctas, procedemos a borrar
        this.apiService.delete('users/me').subscribe({
          next: () => {
            this.authService.logout();
            this.dialogRef.close({ success: true });
            this.router.navigate(['/login']);
          },
          error: (err) => {
```

```

        this.handleError(err);
    }
    });
},
error: (err) => {
    this.handleError(err);
}
});
}

private handleError(err: any): void {
    this.isSubmitting = false;
    console.error('Error:', err);
    this.errorMessage = err.error?.message || 'Credenciales
incorrectas o error al borrar la cuenta';
}

onClose(): void {
    this.dialogRef.close();
}
}

```

Plantilla HTML (del-profile-modal.component.html)

html

```

<div class="modal-overlay" *ngIf="isVisible" (click)="onClose()">
    <div class="modal-content" (click)="$event.stopPropagation()">
        <div class="modal-header bg-danger text-white">
            <h2>Borrar Cuenta Permanentemente</h2>
            <button class="close-btn text-white"
(click)="onClose()">&times;</button>
        </div>

        <div class="modal-body">
            <div class="alert alert-danger mb-4">
                <strong>¡Advertencia!</strong> Esta acción es irreversible.
                Todos tus datos serán eliminados permanentemente.
            </div>

            <form (ngSubmit)="onSubmit()">
                <div class="form-group mb-3">
                    <label class="form-label">Ingresa tu email para
confirmar</label>
                    <input
                        type="email"
                        class="form-control"
                        [(ngModel)]="email"
                        name="email"
                        placeholder="tu@email.com"
                        required
                    >
                    <div class="text-danger mt-1"
*ngIf="errorMessage?.includes('email')">
                        {{errorMessage}}
                    </div>
                </div>

                <div class="form-group mb-4">
                    <label class="form-label">Ingresa tu contraseña</label>
                    <input

```

```

        type="password"
        class="form-control"
        [(ngModel)]="password"
        name="password"
        placeholder="Tu contraseña actual"
        required
    >
    <div class="text-danger mt-1"
*ngIf="errorMessage?.includes('contraseña') ||
errorMessage?.includes('Credenciales')">
        {{errorMessage}}
    </div>
</div>

<div class="modal-footer">
    <button type="button" class="btn btn-outline-secondary"
(click)="onClose()" [disabled]="isSubmitting">
        Cancelar
    </button>
    <button
        type="submit"
        class="btn btn-danger"
        [disabled]="isSubmitting || !email || !password"
    >
        <span *ngIf="!isSubmitting">Confirmar Borrado</span>
        <span *ngIf="isSubmitting">
            <span class="spinner-border spinner-border-sm"
role="status" aria-hidden="true"></span>
            Procesando...
        </span>
    </button>
</div>
</form>
</div>
</div>
</div>

```

Cómo usarlo desde el componente de perfil

En tu `profile.component.ts`:

typescript

```

openDeleteModal(): void {
    const dialogRef = this.dialog.open(DelProfileModalComponent, {
        width: '500px',
        data: { currentEmail: this.user.email }
    });

    dialogRef.afterClosed().subscribe(result => {
        if (result?.success) {
            // Opcional: Mostrar mensaje de confirmación
            this.authService.logout();
            this.router.navigate(['/login']);
        }
    });
}

```

Y en tu `profile.component.html` añade el botón:

html

```
<button class="btn btn-outline-danger px-4"
(click)="openDeleteModal()">
  <i class="bi bi-trash-fill me-2"></i>
  Borrar Cuenta
</button>
```

Características implementadas:

1. **Doble confirmación** mediante email y contraseña
2. **Validación en tiempo real** de los campos
3. **Protección contra borrado accidental**
4. **Feedback visual** durante la operación
5. **Manejo de errores** con mensajes claros
6. **Redirección automática** tras borrado exitoso
7. **Estilos consistentes** con el resto de la aplicación

El componente verifica las credenciales antes de proceder al borrado, lo que añade una capa extra de seguridad.

```
import { Component, Inject } from '@angular/core'; import {
MAT_DIALOG_DATA, MatDialogRef } from '@angular/material/dialog'; import
{ ApiService } from '../shared/services/api.service'; import { Router } from
'@angular/router'; import { AuthService } from
'../shared/services/auth.service'; @Component({ selector: 'app-edit-profile-
modal', standalone: false, templateUrl: './del-profile-modal.component.html',
styleUrls: ['./del-profile-modal.component.scss'] }) export class
DelProfileModalComponent { email: string = ''; password: string = ''; isSubmitting
= false; isVisible = true; errorMessage: string | null = null; constructor( private
apiService: ApiService, private authService: AuthService, private dialogRef:
MatDialogRef<DelProfileModalComponent>, private router: Router,
@Inject(MAT_DIALOG_DATA) public data: { currentEmail: string } ) {}
onSubmit(): void { if (!this.email || !this.password) { this.errorMessage = 'Por favor
ingresa tu email y contraseña'; return; } if (this.email !== this.data.currentEmail) {
this.errorMessage = 'El email no coincide con tu cuenta'; return; }
this.isSubmitting = true; this.errorMessage = null; this.isSubmitting = true; const
deleteData: any = {}; deleteData.email = this.email; deleteData.password =
this.password; this.authService.login(deleteData).subscribe({ next: () => { // Si las
credenciales son correctas, procedemos a borrar
this.apiService.delete('users/me').subscribe({ next: () => { this.authService.logout();
this.dialogRef.close({ success: true }); this.router.navigate(['/login']); }, error: (err)
=> { this.handleError(err); } }); }, error: (err) => { this.handleError(err); } }); }
onClose(): void { this.dialogRef.close(); } } Handle error no existe, como me
recomiendas arreglarlo?
```

Implementación de `handleError` en DelProfileModalComponent

Para solucionar el error de que `handleError` no existe, aquí está la implementación completa con el método de manejo de errores:

typescript

```
import { Component, Inject } from '@angular/core';
import { MAT_DIALOG_DATA, MatDialogRef } from
'@angular/material/dialog';
import { ApiService } from '../.../shared/services/api.service';
import { Router } from '@angular/router';
import { AuthService } from '../.../shared/services/auth.service';

@Component({
  selector: 'app-del-profile-modal',
  templateUrl: './del-profile-modal.component.html',
  styleUrls: ['./del-profile-modal.component.scss']
})
export class DelProfileModalComponent {
  email: string = '';
  password: string = '';
  isSubmitting = false;
  isVisible = true;
  errorMessage: string | null = null;

  constructor(
    private apiService: ApiService,
    private authService: AuthService,
    private dialogRef: MatDialogRef<DelProfileModalComponent>,
    private router: Router,
    @Inject(MAT_DIALOG_DATA) public data: { currentEmail: string }
  ) {}

  onSubmit(): void {
    if (!this.email || !this.password) {
      this.errorMessage = 'Por favor ingresa tu email y contraseña';
      return;
    }

    if (this.email !== this.data.currentEmail) {
      this.errorMessage = 'El email no coincide con tu cuenta';
      return;
    }

    this.isSubmitting = true;
    this.errorMessage = null;

    const credentials = {
      email: this.email,
      password: this.password
    };

    this.authService.login(credentials.email,
credentials.password).subscribe({
      next: () => {
        this.apiService.delete('users/me').subscribe({
          next: () => {
            this.authService.logout();
            this.dialogRef.close({ success: true });
            this.router.navigate(['/login']);
          },
          error: (err) => {
```

```

        this.handleError(err, 'Error al eliminar la cuenta');
    }
    });
},
error: (err) => {
    this.handleError(err, 'Credenciales incorrectas');
}
});
}

private handleError(error: any, defaultMessage: string): void {
    this.isSubmitting = false;
    console.error('Error:', error);

    // Manejo específico de errores HTTP
    if (error.status === 0) {
        this.errorMessage = 'Error de conexión. Verifica tu conexión a internet';
    } else if (error.status === 401) {
        this.errorMessage = 'Credenciales inválidas. Verifica tu email y contraseña';
    } else if (error.error?.message) {
        this.errorMessage = error.error.message;
    } else {
        this.errorMessage = defaultMessage;
    }
}

onClose(): void {
    this.dialogRef.close();
}
}

```

Mejoras implementadas:

1. **Método `handleError` completo:**
 - Maneja diferentes tipos de errores (conexión, autenticación, etc.)
 - Proporciona mensajes específicos según el tipo de error
 - Registra el error en la consola para depuración
2. **Optimización del código:**
 - Eliminé la variable `deleteData` redundante y usé `credentials` directamente
 - Mejoré el flujo de las suscripciones anidadas
 - Agregué un parámetro `defaultMessage` para personalizar los mensajes
3. **Manejo de errores específicos:**
 - Error 401 para credenciales inválidas
 - Error 0 para problemas de conexión
 - Mensajes personalizados del servidor cuando están disponibles
4. **Seguridad:**
 - Siempre limpia el estado de `isSubmitting` cuando ocurre un error
 - Proporciona feedback claro al usuario sobre lo que salió mal

Uso recomendado en la plantilla:

Asegúrate de mostrar el `errorMessage` en tu HTML:

```
html
<div class="alert alert-danger" *ngIf="errorMessage">
  {{ errorMessage }}
</div>
```

Esta implementación proporciona un manejo robusto de errores mientras mantiene una buena experiencia de usuario.

la redirección y el borrado de local STORAGE donde debería ir? en el componente del modal o en el componente de la página?

Ubicación de la Redirección y Borrado de LocalStorage

La mejor práctica es **dividir estas responsabilidades** entre ambos componentes:

1. En el Modal (DelProfileModalComponent) - Lógica de Confirmación:

```
typescript
// del-profile-modal.component.ts
onSubmit(): void {
  // ... validaciones previas

  this.authService.login(credentials.email,
    credentials.password).subscribe({
    next: () => {
      this.apiService.delete('users/me').subscribe({
        next: () => {
          // Solo cerramos el modal con estado de éxito
          this.dialogRef.close({ success: true });
        },
        error: (err) => this.handleError(err)
      });
    },
    error: (err) => this.handleError(err)
  });
}
```

2. En el Componente de Página (ProfileComponent) - Lógica de Post-Borrado:

```
typescript
// profile.component.ts
onDeleteModal(): void {
  const dialogRef = this.dialog.open(DelProfileModalComponent, {
    data: { currentEmail: this.user.email }
  });

  dialogRef.afterClosed().subscribe(result => {
    if (result?.success) {
```



```

// 1. Cerrar sesión (limpia localStorage/token)
this.authService.logout();

// 2. Redirección
this.router.navigate(['/login']);

// 3. Feedback opcional (podría ser un snackbar/toast)
this.snackBar.open('Cuenta eliminada permanentemente', 'Cerrar',
{
    duration: 3000
});
}
});
}

```

¿Por qué esta estructura?

- Separación de responsabilidades:**
 - El modal solo se encarga de **validar y confirmar** el borrado
 - El componente padre maneja las **consecuencias** del borrado confirmado
- Ventajas:**
 - Más fácil de mantener (cambios en redirección/flujo no afectan al modal)
 - El modal puede reutilizarse en otros contextos
 - La lógica de post-borrado (como mostrar notificaciones) queda centralizada
- Seguridad:**
 - El logout (que limpia el localStorage) se ejecuta en el contexto del componente que tiene acceso al AuthService
 - La redirección se maneja desde el componente que conoce la estructura de rutas

Implementación recomendada del AuthService.logout():

```

typescript
// auth.service.ts
logout(): void {
    // 1. Limpiar localStorage
    localStorage.removeItem('auth_token');
    localStorage.removeItem('user_data');

    // 2. Limpiar cualquier otro estado de la app
    this.currentUserSubject.next(null);

    // 3. Opcional: Notificar a otros servicios
    this.eventsService.emit('logout');
}

```

Esta separación hace tu código más modular, testeable y mantenible. El modal solo confirma la acción peligrosa, mientras que el componente padre ejecuta las consecuencias de esa confirmación.

Ya tengo un método `logout` en mi `auth`, que hace ambas cosas, donde lo incluyo?

```
openDeleteModal(): void { const dialogRef =
this.dialog.open(DelProfileModalComponent, { width: '500px', data: {
currentName: this.user.email } }); dialogRef.afterClosed().subscribe(result => { if
(result?.success) { this.user.name = result.name; // Actualiza el nombre si se editó }
}); } this.authService.login(deleteData).subscribe({ next: () => { // Si las
credenciales son correctas, procedemos a borrar
this.apiService.delete('users/me').subscribe({ next: () => { this.authService.logout();
this.dialogRef.close({ success: true }); }, error: (err) => { this.handleError(err,
'Error al eliminar la cuenta'); } }); }, error: (err) => { this.handleError(err,
'Credenciales incorrectas'); } });
```

Implementación Óptima con Logout Existente

Dado que ya tienes un método `logout()` completo en tu `AuthService`, aquí está la estructura recomendada:

En el Modal (`DelProfileModalComponent`) - Solo confirmación de borrado:

```
typescript
onSubmit(): void {
  // ... validaciones previas

  this.authService.login(this.email, this.password).subscribe({
    next: () => {
      this.apiService.delete('users/me').subscribe({
        next: () => {
          // Solo cerramos el modal con estado de éxito
          this.dialogRef.close({
            success: true,
            shouldLogout: true // Bandera adicional
          });
        },
        error: (err) => this.handleError(err, 'Error al eliminar la
cuenta')
      });
    },
    error: (err) => this.handleError(err, 'Credenciales incorrectas')
  });
}
```

En el Componente Padre (`ProfileComponent`) - Manejo completo:

```
typescript
openDeleteModal(): void {
  const dialogRef = this.dialog.open(DelProfileModalComponent, {
    width: '500px',
```

```

    data: { currentEmail: this.user.email } // Cambiado a currentEmail
    para consistencia
  });

  dialogRef.afterClosed().subscribe(result => {
    if (result?.success) {
      // Ejecuta el logout que ya tienes implementado
      this.authService.logout();

      // Redirección (mejor aquí que en el modal)
      this.router.navigate(['/login']);

      // Opcional: Mensaje de confirmación
      this.snackBar.open('Cuenta eliminada exitosamente', 'Cerrar', {
        duration: 3000,
        panelClass: ['success-snackbar']
      });
    }
  });
}

```

Beneficios de esta estructura:

1. **El modal sigue siendo reutilizable** para otros flujos de confirmación
2. **Centralizas la lógica de post-borrado** en un solo lugar
3. **Mantienes limpio tu AuthService** sin mezclar responsabilidades
4. **Más fácil de testear** cada componente por separado

Si prefieres mantener el logout en el modal:

En caso que necesites que el logout sí ocurra en el modal (por ejemplo, si hay múltiples lugares que podrían llamarlo), puedes hacerlo así:

typescript

```

// En DelProfileModalComponent
next: () => {
  this.apiService.delete('users/me').subscribe({
    next: () => {
      this.authService.logout(); // Ejecutas el logout existente
      this.dialogRef.close({
        success: true,
        alreadyLoggedOut: true // Informa al componente padre
      });
      this.router.navigate(['/login']); // Redirección aquí
    }
  });
}

```

typescript

```

// En ProfileComponent
dialogRef.afterClosed().subscribe(result => {
  if (result?.success && !result.alreadyLoggedOut) {
    this.authService.logout();
    this.router.navigate(['/login']);
  }
});

```

Recomendación final:

La primera opción (logout en el componente padre) es más limpia porque:

- El modal no necesita conocer sobre rutas
- Todo el flujo post-borrado está en un solo lugar
- El modal solo se preocupa por confirmar la acción peligrosa

```
X [ERROR] NG8002: Can't bind to 'ngModel' since it isn't a known property of
'input'. [plugin angular-compiler] src/app/components/ui/del-profile-modal/del-
profile-modal.component.html:34:12: 34 | [(ngModel)]="password" |
~~~~~ Error occurs in the template of component
DelProfileModalComponent. src/app/components/ui/del-profile-modal/del-profile-
modal.component.ts:10:15: 10 | templateUrl: './del-profile-
modal.component.html', | ~~~~~
<div
class="modal-overlay" *ngIf="isVisible" (click)="onClose()"> <div class="modal-
content" (click)="$event.stopPropagation()"> <div class="modal-header bg-
danger text-white"> <h2>Borrar Cuenta Permanentemente</h2> <button
class="close-btn text-white" (click)="onClose()">&times;</button> </div> <div
class="modal-body"> <div class="alert alert-danger mb-4">
<strong>¡Advertencia!</strong> Esta acción es irreversible. Todos tus datos serán
eliminados permanentemente. </div> <form (ngSubmit)="onSubmit()"> <div
class="form-group mb-3"> <label class="form-label">Ingresa tu email para
confirmar</label> <input type="email" class="form-control"
[(ngModel)]="email" name="email" placeholder="tu@email.com" required >
<div class="text-danger mt-1" *ngIf="errorMessage?.includes('email')">
{{errorMessage}} </div> </div> <div class="form-group mb-4"> <label
class="form-label">Ingresa tu contraseña</label> <input type="password"
class="form-control" [(ngModel)]="password" name="password"
placeholder="Tu contraseña actual" required > <div class="text-danger mt-1"
*ngIf="errorMessage?.includes('contraseña') ||
errorMessage?.includes('Credenciales')"> {{errorMessage}} </div> </div> <div
class="modal-footer"> <button type="button" class="btn btn-outline-secondary"
(click)="onClose()" [disabled]="isSubmitting"> Cancelar </button> <button
type="submit" class="btn btn-danger" [disabled]="isSubmitting || !email ||
!password"> <span *ngIf="!isSubmitting">Confirmar Borrado</span> <span
*ngIf="isSubmitting"> <span class="spinner-border spinner-border-sm"
role="status" aria-hidden="true"></span> Procesando... </span> </button>
</div> </form> </div> </div> </div>
```

Solución para el Error NG8002 con ngModel

El error ocurre porque necesitas importar el módulo `FormsModule` en el módulo que declara tu componente `DelProfileModalComponent`. Aquí te muestro cómo solucionarlo:

1. Verifica las importaciones en tu módulo

Asegúrate de que el módulo que contiene tu componente tenga:

```
typescript
import { FormsModule } from '@angular/forms';

@NgModule({
  imports: [
    FormsModule, // <-- Esta importación es crucial
    // ... otros módulos
  ],
  // ...
})
export class TuModulo { }
```

2. Soluciones específicas según tu estructura:

Opción A: Si usas un módulo dedicado para el modal:

En del-profile-modal.module.ts:

```
typescript
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms'; // <-- Importa
FormsModule
import { DelProfileModalComponent } from '../del-profile-
modal.component';

@NgModule({
  declarations: [DelProfileModalComponent],
  imports: [
    CommonModule,
    FormsModule // <-- Añade esta línea
  ]
})
export class DelProfileModalModule { }
```

Opción B: Si el componente está declarado en AppModule:

En app.module.ts:

```
typescript
import { FormsModule } from '@angular/forms';

@NgModule({
  imports: [
    // ... otros imports
    FormsModule // <-- Asegúrate de que esté importado
  ],
  // ...
})
export class AppModule { }
```

3. Solución alternativa sin ngModel:

Si prefieres no usar `ngModel`, puedes modificar el template así:

html

```
<input
  type="email"
  class="form-control"
  [value]="email"
  (input)="email = $event.target.value"
  name="email"
  placeholder="tu@email.com"
  required
>

<input
  type="password"
  class="form-control"
  [value]="password"
  (input)="password = $event.target.value"
  name="password"
  placeholder="Tu contraseña actual"
  required
>
```

4. Verificación final:

1. Asegúrate de que:
 - `FormsModule` esté importado en el módulo correcto
 - No tengas errores de importación circular
 - El componente no esté declarado en más de un módulo
2. Si usas Angular 17+ con standalone components, la solución sería:

typescript

```
@Component({
  standalone: true,
  imports: [FormsModule], // <-- Importa FormsModule aquí
  // ...
})
export class DelProfileModalComponent { }
```

¿Por qué ocurre este error?

Angular necesita que `FormsModule` esté importado para poder usar:

- `[(ngModel)]` (two-way data binding)
- Directivas de formularios template-driven
- Validadores básicos

La solución más limpia es importar `FormsModule` en el módulo que declara tu componente, lo que te permitirá seguir usando `ngModel` como lo tienes actualmente en tu template.

```

<div class="modal-overlay" *ngIf="isVisible" (click)="onClose()"> <div
class="modal-content" (click)="$event.stopPropagation()"> <div class="modal-
header bg-danger text-white"> <h2>Borrar Cuenta Permanentemente</h2>
<button class="close-btn text-white" (click)="onClose()">&times;</button>
</div> <div class="modal-body"> <div class="alert alert-danger mb-4">
<strong>¡Advertencia!</strong> Esta acción es irreversible. Todos tus datos serán
eliminados permanentemente. </div> <form (ngSubmit)="onSubmit()"> <div
class="form-group mb-3"> <label class="form-label">Ingresa tu email para
confirmar</label> <input type="email" class="form-control"
[(ngModel)]="email" name="email" placeholder="tu@email.com" required >
<div class="text-danger mt-1" *ngIf="errorMessage?.includes('email')">
{{errorMessage}} </div> </div> <div class="form-group mb-4"> <label
class="form-label">Ingresa tu contraseña</label> <input type="password"
class="form-control" [(ngModel)]="password" name="password"
placeholder="Tu contraseña actual" required > <div class="text-danger mt-1"
*ngIf="errorMessage?.includes('contraseña') ||
errorMessage?.includes('Credenciales')"> {{errorMessage}} </div> </div> <div
class="modal-footer"> <button type="button" class="btn btn-outline-secondary"
(click)="onClose()" [disabled]="isSubmitting"> Cancelar </button> <button
type="submit" class="btn btn-danger" [disabled]="isSubmitting || !email ||
!password" > <span *ngIf="!isSubmitting">Confirmar Borrado</span> <span
*ngIf="isSubmitting"> <span class="spinner-border spinner-border-sm"
role="status" aria-hidden="true"></span> Procesando... </span> </button>
</div> </form> </div> </div> </div> import { Component, Inject } from
'@angular/core'; import { MAT_DIALOG_DATA, MatDialogRef } from
'@angular/material/dialog'; import { ApiService } from
'../shared/services/api.service'; import { Router } from '@angular/router';
import { AuthService } from '../shared/services/auth.service'; @Component({
selector: 'app-edit-profile-modal', standalone: false, templateUrl: './del-profile-
modal.component.html', styleUrls: ['./del-profile-modal.component.scss'] }) export
class DelProfileModalComponent { email: string = ''; password: string = '';
isSubmitting = false; isVisible = true; errorMessage: string | null = null;
constructor( private apiService: ApiService, private authService: AuthService,
private dialogRef: MatDialogRef<DelProfileModalComponent>, private router:
Router, @Inject(MAT_DIALOG_DATA) public data: { currentEmail: string } ) {
onSubmit(): void { if (!this.email || !this.password) { this.errorMessage = 'Por favor
ingresa tu email y contraseña'; return; } if (this.email !== this.data.currentEmail) {
this.errorMessage = 'El email no coincide con tu cuenta'; return; }
this.isSubmitting = true; this.errorMessage = null; this.isSubmitting = true; const
deleteData: any = {}; deleteData.email = this.email; deleteData.password =
this.password; this.authService.login(deleteData).subscribe({ next: () => { // Si las
credenciales son correctas, procedemos a borrar
this.apiService.delete('users/me').subscribe({ next: () => {
//this.authService.logout(); //mejor en el componente padre //TODO Confirmación
de borrado this.dialogRef.close({ success: true }); }, error: (err) => {
this.handleError(err, 'Error al eliminar la cuenta'); } }); }, error: (err) => {
this.handleError(err, 'Credenciales incorrectas'); } }); } private
handleError(error: any, defaultMessage: string): void { this.isSubmitting = false;
console.error('Error:', error); // Manejo específico de errores HTTP if
(error.status === 0) { this.errorMessage = 'Error de conexión. Verifica tu conexión

```

```
a internet'; } else if (error.status === 401) { this.errorMessage = 'Credenciales
inválidas. Verifica tu email y contraseña'; } else if (error.error?.message) {
this.errorMessage = error.error.message; } else { this.errorMessage =
defaultMessage; } } onClose(): void { this.dialogRef.close(); } } import { NgModule
} from '@angular/core'; import { CommonModule } from '@angular/common';
import { DelProfileModalComponent } from './del-profile-modal.component';
import { FormsModule } from '@angular/forms'; import { MatDialogModule }
from '@angular/material/dialog'; @NgModule({ declarations:
[DelProfileModalComponent], imports: [ CommonModule, FormsModule,
MatDialogModule ] }) export class DelProfileModalModule { } Hazme la
confirmación de borrado y dime qué falla para que me de ese error
```

Solución Completa para el Error y Confirmación de Borrado

Problema del Error NG8002

El error persiste porque aunque has creado DelProfileModalModule con FormsModule, necesitas asegurarte de que:

1. El módulo esté correctamente importado en el módulo que usa el componente
2. El componente esté correctamente declarado

Solución Definitiva

1. Modifica del-profile-modal.module.ts:

```
typescript
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { MatDialogModule } from '@angular/material/dialog';
import { DelProfileModalComponent } from './del-profile-
modal.component';

@NgModule({
  declarations: [DelProfileModalComponent],
  imports: [
    CommonModule,
    FormsModule, // Para ngModel
    MatDialogModule // Para el diálogo
  ],
  exports: [DelProfileModalComponent] // Añade esta línea
})
export class DelProfileModalModule { }
```

2. Importa el módulo donde usas el componente (normalmente app.module.ts):

```
typescript
```



```
import { DelProfileModalModule } from '../components/ui/del-profile-modal/del-profile-modal.module';

@NgModule({
  imports: [
    // ... otros imports
    DelProfileModalModule
  ],
  // ...
})
export class AppModule { }
```

Implementación de Confirmación de Borrado

En del-profile-modal.component.ts:

typescript

```
onSubmit(): void {
  if (!this.email || !this.password) {
    this.errorMessage = 'Por favor ingresa tu email y contraseña';
    return;
  }

  if (this.email !== this.data.currentEmail) {
    this.errorMessage = 'El email no coincide con tu cuenta';
    return;
  }

  // Confirmación adicional
  const confirmDelete = confirm('¿Estás absolutamente seguro de que quieres eliminar tu cuenta? Esta acción no se puede deshacer.');
```

```
  if (!confirmDelete) {
    return;
  }

  this.isSubmitting = true;
  this.errorMessage = null;

  this.authService.login(this.email, this.password).subscribe({
    next: () => {
      this.apiService.delete('users/me').subscribe({
        next: () => {
          this.dialogRef.close({
            success: true,
            message: 'Cuenta eliminada permanentemente'
          });
        },
        error: (err) => this.handleError(err, 'Error al eliminar la cuenta')
      });
    },
    error: (err) => this.handleError(err, 'Credenciales incorrectas')
  });
}
```

En el componente padre (donde abres el modal):

typescript

```

openDeleteModal(): void {
  const dialogRef = this.dialog.open(DelProfileModalComponent, {
    width: '500px',
    data: { currentEmail: this.user.email }
  });

  dialogRef.afterClosed().subscribe(result => {
    if (result?.success) {
      // 1. Cerrar sesión
      this.authService.logout();

      // 2. Redirección
      this.router.navigate(['/login']);

      // 3. Mostrar confirmación
      this.snackBar.open(result.message || 'Cuenta eliminada
exitosamente', 'Cerrar', {
        duration: 5000,
        panelClass: ['success-snackbar']
      });
    }
  });
}

```

¿Por qué seguías viendo el error?

1. **Posible causa:** El módulo `DelProfileModalModule` no estaba importado en el módulo principal
2. **Otra causa:** El componente podría estar declarado en más de un módulo
3. **Solución alternativa:** Si usas Angular 14+ con standalone components:

typescript

```

@Component({
  standalone: true,
  imports: [FormsModule, MatDialogModule], // <-- Importa aquí
  templateUrl: './del-profile-modal.component.html',
  styleUrls: ['./del-profile-modal.component.scss']
})
export class DelProfileModalComponent {
  // ... resto del código
}

```

Verificación Final

1. Asegúrate de que:
 - o Solo hay una declaración del componente
 - o `FormsModule` está importado en el módulo correcto
 - o No hay errores de importación circular
2. Prueba estos comandos si persisten problemas:

bash

```

npm install @angular/forms --save
ng serve

```

snackBar me sale en rojo en el componente padre `import { Component } from '@angular/core'; import { AuthService } from '../shared/services/auth.service';`

```

import { jwtDecode } from 'jwt-decode'; import { ApiService } from
'../../shared/services/api.service'; import { MatDialog } from
'@angular/material/dialog'; import { EdProfileModalComponent } from
'../../components/ui/ed-profile-modal/ed-profile-modal.component'; import {
DelProfileModalComponent } from ' ../../components/ui/del-profile-modal/del-
profile-modal.component'; @Component({ selector: 'app-profile', standalone:
false, templateUrl: './profile.component.html', styleUrls:
['./profile.component.scss'] }) export class ProfileComponent { user = { id: 3, name:
'Alex García', email: 'alex@ejemplo.com', stats: { tasksCompleted: 0,
tasksPending: 0, categories: 0 }, }; isLoading = true; errorMessage: string | null =
null; constructor( private authService: AuthService, private apiService:
ApiService, private dialog: MatDialog ) {} ngOnInit(): void { this.loadUserData();
//this.loadStatistics(); } loadUserData(): void { this.isLoading = true;
this.errorMessage = null; // Suponiendo que tienes un método para obtener los
datos del usuario logueado const cut = this.authService.getDecodedToken(); //if
(cut) { // this.user.id = cut.sub; // this.user.name = cut.name; // this.user.email =
cut.email; // } // <- Actualmente desde el token podría sacarse todo
//this.apiService.get(`users/id/${this.user.id}`) // <- Este endpoint sirve pero mejor
usar el seguro this.apiService.get(`auth/getProfile`).subscribe({ // <- Este endpoint
actualmente saca la info del //propio JWT, lo que no es lo suyo para obtener más
datos. Se puede modificar de cara a futuro next: (userDetails) => { this.user = {
...this.user, id: userDetails.sub, name: userDetails.name, email: userDetails.email, }
this.isLoading = false; }, error: (err) => { console.error('Error loading profile:',
err); this.errorMessage = 'Error al cargar el perfil'; this.isLoading = false; } }) }
openEditModal(): void { const dialogRef =
this.dialog.open(EdProfileModalComponent, { width: '500px', data: {
currentName: this.user.name } }); dialogRef.afterClosed().subscribe(result => { if
(result?.success) { this.user.name = result.name; // Actualiza el nombre si se editó }
}); } openDeleteModal(): void { const dialogRef =
this.dialog.open(DelProfileModalComponent, { width: '500px', data: {
currentEmail: this.user.email } }); dialogRef.afterClosed().subscribe(result => { if
(result?.success) { this.authService.logout(); this.snackBar.open(result.message ||
'Cuenta eliminada exitosamente', 'Cerrar', { duration: 5000, panelClass: ['success-
snackbar'] }); } }); } logout(): void { this.authService.logout(); } }

```

Solución para el Error de snackBar

El error ocurre porque no has importado e inyectado el servicio MatSnackBar en tu ProfileComponent. Aquí está la solución completa:

1. Importa MatSnackBar en tu componente

typescript

```

import { Component } from '@angular/core';
import { AuthService } from ' ../../shared/services/auth.service';
import { ApiService } from ' ../../shared/services/api.service';
import { MatDialog } from '@angular/material/dialog';
import { MatSnackBar } from '@angular/material/snack-bar'; // <--
Añade esta importación

```

```
import { EdProfileModalComponent } from '../../components/ui/ed-profile-modal/ed-profile-modal.component';
import { DelProfileModalComponent } from '../../components/ui/del-profile-modal/del-profile-modal.component';
```

2. Inyecta el servicio en el constructor

```
typescript
constructor(
  private authService: AuthService,
  private apiService: ApiService,
  private dialog: MatDialog,
  private snackBar: MatSnackBar // <-- Añade esta inyección
) {}
```

3. Asegúrate de tener importado MatSnackBarModule

En el módulo donde está declarado tu ProfileComponent (normalmente app.module.ts o un módulo de características):

```
typescript
import { MatSnackBarModule } from '@angular/material/snack-bar';

@NgModule({
  imports: [
    // ... otros imports
    MatSnackBarModule // <-- Añade esta línea
  ],
  // ...
})
export class AppModule { }
```

4. Implementación completa corregida

```
typescript
openDeleteModal(): void {
  const dialogRef = this.dialog.open(DelProfileModalComponent, {
    width: '500px',
    data: { currentEmail: this.user.email }
  });

  dialogRef.afterClosed().subscribe(result => {
    if (result?.success) {
      this.authService.logout();
      this.snackBar.open(
        result.message || 'Cuenta eliminada exitosamente',
        'Cerrar',
        {
          duration: 5000,
          panelClass: ['success-snackbar']
        }
      );
    }
  });
}
```

5. Si usas un módulo específico para el perfil

Si `ProfileComponent` está en su propio módulo, asegúrate de importar `MatSnackBarModule` allí:

typescript

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { MatSnackBarModule } from '@angular/material/snack-bar';
import { ProfileComponent } from '../profile.component';

@NgModule({
  declarations: [ProfileComponent],
  imports: [
    CommonModule,
    MatSnackBarModule // <-- Añade esta línea
  ]
})
export class ProfileModule { }
```

Verificación adicional

1. Asegúrate de tener instalado `@angular/material`:

bash

- `npm install @angular/material @angular/cdk`
- Verifica que tengas un tema de Angular Material en tu `styles.css`:

css

2. `@import "@angular/material/prebuilt-themes/indigo-pink.css";`
3. Si usas Angular 17+ con standalone components, la solución sería:

typescript

```
@Component({
  standalone: true,
  imports: [MatSnackBarModule], // <-- Importa aquí
  // ...
})
export class ProfileComponent { }
```

Con estos cambios, el `snackBar` debería funcionar correctamente mostrando el mensaje de confirmación cuando se elimina la cuenta.

tnego 2 componentes casi iguales y en el edición no me tira el error mientras que en el de borrado sí: [ERROR] NG8002: Can't bind to 'ngModel' since it isn't a known property of 'input'. [plugin angular-compiler] src/app/components/ui/del-profile-modal/del-profile-modal.component.html:34:12: 34 | [(ngModel)]="password" | ~~~~~ Error occurs in the template of component DelProfileModalComponent. src/app/components/ui/del-profile-modal/del-profile-modal.component.ts:10:15: 10 | templateUrl: './del-

```

profile-modal.component.html', | ~~~~~
import { NgModule } from '@angular/core'; import { CommonModule } from
'@angular/common'; import { DelProfileModalComponent } from './del-profile-
modal.component'; import { FormsModule } from '@angular/forms'; import {
MatDialogModule } from '@angular/material/dialog'; @NgModule({ declarations:
[DelProfileModalComponent], imports: [ CommonModule, FormsModule,
MatDialogModule ] }) export class DelProfileModalModule { } import {
Component, Inject } from '@angular/core'; import { MAT_DIALOG_DATA,
MatDialogRef } from '@angular/material/dialog'; import { ApiService } from
'../../shared/services/api.service'; import { Router } from '@angular/router';
import { AuthService } from '../../shared/services/auth.service'; @Component({
selector: 'app-edit-profile-modal', standalone: false, templateUrl: './del-profile-
modal.component.html', styleUrls: ['./del-profile-modal.component.scss'] }) export
class DelProfileModalComponent { email: string = ''; password: string = '';
isSubmitting = false; isVisible = true; errorMessage: string | null = null;
constructor( private apiService: ApiService, private authService: AuthService,
private dialogRef: MatDialogRef<DelProfileModalComponent>, private router:
Router, @Inject(MAT_DIALOG_DATA) public data: { currentEmail: string } ) { }
onSubmit(): void { if (!this.email || !this.password) { this.errorMessage = 'Por favor
ingresa tu email y contraseña'; return; } if (this.email !== this.data.currentEmail) {
this.errorMessage = 'El email no coincide con tu cuenta'; return; } const
confirmDelete = confirm('¿Estás absolutamente seguro de que quieres eliminar tu
cuenta? Esta acción no se puede deshacer. '); if (!confirmDelete) { return; }
this.isSubmitting = true; this.errorMessage = null; this.isSubmitting = true; const
deleteData: any = {}; deleteData.email = this.email; deleteData.password =
this.password; this.authService.login(deleteData).subscribe({ next: () => { // Si las
credenciales son correctas, procedemos a borrar
this.apiService.delete('users/me').subscribe({ next: () => { this.dialogRef.close({
success: true, message: 'Cuenta eliminada permanentemente' }); }, error: (err) =>
{ this.handleError(err, 'Error al eliminar la cuenta'); } }); }, error: (err) => {
this.handleError(err, 'Credenciales incorrectas'); } }); } private
handleError(error: any, defaultMessage: string): void { this.isSubmitting = false;
console.error('Error:', error); // Manejo específico de errores HTTP if
(error.status === 0) { this.errorMessage = 'Error de conexión. Verifica tu conexión
a internet'; } else if (error.status === 401) { this.errorMessage = 'Credenciales
inválidas. Verifica tu email y contraseña'; } else if (error.error?.message) {
this.errorMessage = error.error.message; } else { this.errorMessage =
defaultMessage; } } onClose(): void { this.dialogRef.close(); } } <div class="modal-
overlay" *ngIf="isVisible" (click)="onClose()"> <div class="modal-content"
(click)="$event.stopPropagation()"> <div class="modal-header bg-danger text-
white"> <h2>Borrar Cuenta Permanentemente</h2> <button class="close-btn
text-white" (click)="onClose()">&times;</button> </div> <div class="modal-
body"> <div class="alert alert-danger mb-4"> <strong>¡Advertencia!</strong>
Esta acción es irreversible. Todos tus datos serán eliminados permanentemente.
</div> <form (ngSubmit)="onSubmit()"> <div class="form-group mb-3"> <label
class="form-label">Ingresa tu email para confirmar</label> <input type="email"
class="form-control" [(ngModel)]="email" name="email"
placeholder="tu@email.com" required > <div class="text-danger mt-1"
*ngIf="errorMessage?.includes('email')"> {{errorMessage}} </div> </div> <div
class="form-group mb-4"> <label class="form-label">Ingresa tu

```

```

contraseña</label> <input type="password" class="form-control"
[(ngModel)]="password" name="password" placeholder="Tu contraseña actual"
required > <div class="text-danger mt-1"
*ngIf="errorMessage?.includes('contraseña') ||
errorMessage?.includes('Credenciales')"> {{errorMessage}} </div> </div> <div
class="modal-footer"> <button type="button" class="btn btn-outline-secondary"
(click)="onClose()" [disabled]="isSubmitting"> Cancelar </button> <button
type="submit" class="btn btn-danger" [disabled]="isSubmitting || !email ||
!password" > <span *ngIf="!isSubmitting">Confirmar Borrado</span> <span
*ngIf="isSubmitting"> <span class="spinner-border spinner-border-sm"
role="status" aria-hidden="true"></span> Procesando... </span> </button>
</div> </form> </div> </div> </div> import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common'; import {
EdProfileModalComponent } from './ed-profile-modal.component'; import {
FormsModule } from '@angular/forms'; import { MatDialogModule } from
'@angular/material/dialog'; @NgModule({ declarations:
[EdProfileModalComponent], imports: [ CommonModule, FormsModule,
MatDialogModule ] }) export class EdProfileModalModule { } import {
Component, Inject } from '@angular/core'; import { MAT_DIALOG_DATA,
MatDialogRef } from '@angular/material/dialog'; import { ApiService } from
'../shared/services/api.service'; @Component({ selector: 'app-edit-profile-
modal', standalone: false, templateUrl: './ed-profile-modal.component.html',
styleUrls: ['./ed-profile-modal.component.scss'] }) export class
EdProfileModalComponent { name: string = ''; newPassword: string = '';
isSubmitting = false; isVisible = true; passwordTouched = false; // Variables para
los requisitos de contraseña reqLength = false; reqLowerCase = false;
reqUpperCase = false; reqNumber = false; reqSpecialChar = false; constructor(
private apiService: ApiService, private dialogRef:
MatDialogRef<EdProfileModalComponent>, @Inject(MAT_DIALOG_DATA)
public data: { currentName: string } ) { this.name = data.currentName; }
checkPasswordRequirements(): void { const password = this.newPassword;
this.reqLength = password?.length >= 8; this.reqLowerCase = /[a-
z]/.test(password); this.reqUpperCase = /[A-Z]/.test(password); this.reqNumber =
/\\d/.test(password); this.reqSpecialChar = /[\\W_]/.test(password);
this.passwordTouched = !!password && password.length > 0; } get
passwordValid(): boolean { if (!this.newPassword) return true; return
this.reqLength && this.reqLowerCase && this.reqUpperCase &&
this.reqNumber && this.reqSpecialChar; } get formValid(): boolean { return
(this.name.length >= 3 || this.newPassword.length >= 8) && (this.newPassword ?
this.passwordValid : true); } onSubmit(): void { if (!this.formValid) return;
this.isSubmitting = true; const updateData: any = {}; if (this.name && this.name
!== this.data.currentName) { updateData.name = this.name; } if
(this.newPassword) { updateData.password = this.newPassword; } if
(Object.keys(updateData).length === 0) { this.dialogRef.close(); return; }
this.apiService.patch('users/me', updateData).subscribe({ next: () => {
this.dialogRef.close({ success: true, name: updateData.name ||
this.data.currentName }); }, error: (err) => { this.isSubmitting = false;
console.error('Error updating profile:', err); alert('Error updating profile: ' +
(err.error?.message || 'Please try again')); } }); } onClose(): void {
this.dialogRef.close(); } } <div class="modal-overlay" *ngIf="isVisible"

```

```

(click)="onClose()"> <div class="modal-content"
(click)="$event.stopPropagation()"> <div class="modal-header"> <h2>Editar
Perfil</h2> <button class="close-btn" (click)="onClose()">&times;</button>
</div> <div class="modal-body"> <form (ngSubmit)="onSubmit()"> <div
class="form-group mb-3"> <label class="form-label">Nuevo nombre</label>
<input type="text" class="form-control" [(ngModel)]="name" name="name"
placeholder="Dejar vacío para no cambiar" > <div class="text-danger mt-1"
*ngIf="name.length > 0 && name.length < 3"> Mínimo 3 caracteres </div> </div>
<div class="form-group mb-3"> <label class="form-label">Nueva
contraseña</label> <input type="password" class="form-control"
[(ngModel)]="newPassword" name="newPassword" placeholder="Dejar vacío
para no cambiar" (input)="checkPasswordRequirements()" > <div class="mt-2"
*ngIf="passwordTouched"> <div class="d-flex align-items-center mb-1"> <span
class="me-2" [class.text-success]="reqLength" [class.text-danger]="!reqLength">
<i class="fas" [class.fa-check-circle]="reqLength" [class.fa-times-
circle]="!reqLength"></i> </span> <span [class.text-success]="reqLength"
[class.text-muted]="!reqLength"> Al menos 8 caracteres </span> </div> <div
class="d-flex align-items-center mb-1"> <span class="me-2" [class.text-
success]="reqLowerCase" [class.text-danger]="!reqLowerCase"> <i class="fas"
[class.fa-check-circle]="reqLowerCase" [class.fa-times-
circle]="!reqLowerCase"></i> </span> <span [class.text-
success]="reqLowerCase" [class.text-muted]="!reqLowerCase"> Al menos 1 letra
minúscula </span> </div> <div class="d-flex align-items-center mb-1"> <span
class="me-2" [class.text-success]="reqUpperCase" [class.text-
danger]="!reqUpperCase"> <i class="fas" [class.fa-check-
circle]="reqUpperCase" [class.fa-times-circle]="!reqUpperCase"></i> </span>
<span [class.text-success]="reqUpperCase" [class.text-muted]="!reqUpperCase">
Al menos 1 letra mayúscula </span> </div> <div class="d-flex align-items-center
mb-1"> <span class="me-2" [class.text-success]="reqNumber" [class.text-
danger]="!reqNumber"> <i class="fas" [class.fa-check-circle]="reqNumber"
[class.fa-times-circle]="!reqNumber"></i> </span> <span [class.text-
success]="reqNumber" [class.text-muted]="!reqNumber"> Al menos 1 número
</span> </div> <div class="d-flex align-items-center"> <span class="me-2"
[class.text-success]="reqSpecialChar" [class.text-danger]="!reqSpecialChar"> <i
class="fas" [class.fa-check-circle]="reqSpecialChar" [class.fa-times-
circle]="!reqSpecialChar"></i> </span> <span [class.text-
success]="reqSpecialChar" [class.text-muted]="!reqSpecialChar"> Al menos 1
carácter especial ($!%*?&) </span> </div> </div> <div class="text-danger
mb-3" *ngIf="(name.length < 3 || !newPassword) && (name.length > 0 ||
newPassword.length > 0)"> Debes completar al menos un campo válido </div>
<div class="modal-footer"> <button type="button" class="btn btn-outline-
secondary" (click)="onClose()">Cancelar</button> <button type="submit"
class="btn btn-primary" [disabled]="!formValid || isSubmitting" > <span
*ngIf="!isSubmitting">Guardar</span> <span *ngIf="isSubmitting"> <span
class="spinner-border spinner-border-sm" role="status" aria-
hidden="true"></span> Guardando... </span> </button> </div> </form> </div>
</div> </div>

```


Solución al Problema del Error NG8002 en el Componente de Borrado

El error ocurre porque aunque ambos componentes son similares, hay diferencias clave en su configuración que afectan el binding de `ngModel`. Aquí está el análisis y solución:

Causa del Problema

1. **Inconsistencia en los módulos:** Aunque ambos tienen su propio módulo (`EdProfileModalModule` y `DelProfileModalModule`), el error sugiere que el módulo del componente de borrado no está siendo correctamente importado donde se usa.
2. **Diferencia en los selectores:** Noté que el componente de borrado usa el selector `app-edit-profile-modal` (incorrecto) en lugar de `app-del-profile-modal`.

Solución Completa

1. Corrige el selector en `del-profile-modal.component.ts`:

```
typescript
@Component({
  selector: 'app-del-profile-modal', // <-- Cambia esto
  standalone: false,
  templateUrl: './del-profile-modal.component.html',
  styleUrls: ['./del-profile-modal.component.scss']
})
```

2. Asegúrate que el módulo del componente de borrado esté importado:

En el módulo donde usas el diálogo (normalmente `app.module.ts`):

```
typescript
import { DelProfileModalModule } from '../components/ui/del-profile-modal/del-profile-modal.module';

@NgModule({
  imports: [
    // ... otros imports
    DelProfileModalModule
  ],
  // ...
})
export class AppModule { }
```

3. Verifica la estructura completa del módulo (`del-profile-modal.module.ts`):

```
typescript
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
```

```
import { FormsModule } from '@angular/forms';
import { MatDialogModule } from '@angular/material/dialog';
import { DelProfileModalComponent } from './del-profile-
modal.component';

@NgModule({
  declarations: [DelProfileModalComponent],
  imports: [
    CommonModule,
    FormsModule, // Necesario para ngModel
    MatDialogModule
  ],
  exports: [DelProfileModalComponent] // Asegúrate de exportarlo
})
export class DelProfileModalModule { }
```

4. Alternativa si prefieres standalone components (Angular 14+):

```
typescript
// En del-profile-modal.component.ts
@Component({
  standalone: true, // <-- Cambia a standalone
  imports: [FormsModule, MatDialogModule], // <-- Importa los módulos
necesarios
  selector: 'app-del-profile-modal',
  templateUrl: './del-profile-modal.component.html',
  styleUrls: ['./del-profile-modal.component.scss']
})
```

¿Por qué funciona el de edición y no el de borrado?

1. **Posible causa:** El módulo de edición está siendo importado correctamente en otro lugar
2. **Otra posibilidad:** El componente de edición podría estar declarado en el módulo principal

Verificación Final

1. Ejecuta este comando para asegurarte que todo está instalado:

```
bash
npm install @angular/forms @angular/material @angular/cdk --save
```

2. Verifica que no tengas errores de importación circular
3. Revisa que no haya declaraciones duplicadas de los componentes

Solución Temporal (si persiste el error)

Puedes reemplazar temporalmente [(ngModel)] por binding manual:

```
html
<input
  type="password"
```

```

class="form-control"
[value]="password"
(input)="password = $event.target.value"
name="password"
placeholder="Tu contraseña actual"
required
>

```

Pero la solución recomendada es corregir la configuración del módulo como se indicó anteriormente, ya que `ngModel` es la forma estándar de Angular para two-way data binding en formularios template-driven.

```

X [ERROR] NG8002: Can't bind to 'ngModel' since it isn't a known property of
'input'. [plugin angular-compiler] src/app/components/ui/del-profile-modal/del-
profile-modal.component.html:34:12: 34 | [(ngModel)]="password" |
~~~~~ Error occurs in the template of component
DelProfileModalComponent. src/app/components/ui/del-profile-modal/del-profile-
modal.component.ts:10:15: 10 | templateUrl: './del-profile-
modal.component.html', | ~~~~~ <div
class="modal-overlay" *ngIf="isVisible" (click)="onClose()"> <div class="modal-
content" (click)="$event.stopPropagation()"> <div class="modal-header bg-
danger text-white"> <h2>Borrar Cuenta Permanentemente</h2> <button
class="close-btn text-white" (click)="onClose()">&times;</button> </div> <div
class="modal-body"> <div class="alert alert-danger mb-4">
<strong>¡Advertencia!</strong> Esta acción es irreversible. Todos tus datos serán
eliminados permanentemente. </div> <form (ngSubmit)="onSubmit()"> <div
class="form-group mb-3"> <label class="form-label">Ingresa tu email para
confirmar</label> <input type="email" class="form-control"
[(ngModel)]="email" name="email" placeholder="tu@email.com" required >
<div class="text-danger mt-1" *ngIf="errorMessage?.includes('email')">
{{errorMessage}} </div> </div> <div class="form-group mb-4"> <label
class="form-label">Ingresa tu contraseña</label> <input type="password"
class="form-control" [(ngModel)]="password" name="password"
placeholder="Tu contraseña actual" required > <div class="text-danger mt-1"
*ngIf="errorMessage?.includes('contraseña') ||
errorMessage?.includes('Credenciales')"> {{errorMessage}} </div> </div> <div
class="modal-footer"> <button type="button" class="btn btn-outline-secondary"
(click)="onClose()" [disabled]="isSubmitting"> Cancelar </button> <button
type="submit" class="btn btn-danger" [disabled]="isSubmitting || !email ||
!password"> <span *ngIf="!isSubmitting">Confirmar Borrado</span> <span
*ngIf="isSubmitting"> <span class="spinner-border spinner-border-sm"
role="status" aria-hidden="true"></span> Procesando... </span> </button>
</div> </form> </div> </div> </div> import { Component, Inject } from
'@angular/core'; import { MAT_DIALOG_DATA, MatDialogRef } from
'@angular/material/dialog'; import { ApiService } from
'../shared/services/api.service'; import { Router } from '@angular/router';
import { AuthService } from '../shared/services/auth.service'; @Component({
selector: 'app-del-profile-modal', standalone: false, templateUrl: './del-profile-
modal.component.html', styleUrls: ['./del-profile-modal.component.scss'] }) export
class DelProfileModalComponent { email: string = ''; password: string = '';
isSubmitting = false; isVisible = true; errorMessage: string | null = null;

```

```

constructor( private apiService: ApiService, private authService: AuthService,
private dialogRef: MatDialogRef<DelProfileModalComponent>, private router:
Router, @Inject(MAT_DIALOG_DATA) public data: { currentEmail: string } ) {}
onSubmit(): void { if (!this.email || !this.password) { this.errorMessage = 'Por favor
ingresa tu email y contraseña'; return; } if (this.email !== this.data.currentEmail) {
this.errorMessage = 'El email no coincide con tu cuenta'; return; } const
confirmDelete = confirm('¿Estás absolutamente seguro de que quieres eliminar tu
cuenta? Esta acción no se puede deshacer. '); if (!confirmDelete) { return; }
this.isSubmitting = true; this.errorMessage = null; this.isSubmitting = true; const
deleteData: any = {}; deleteData.email = this.email; deleteData.password =
this.password; this.authService.login(deleteData).subscribe({ next: () => { // Si las
credenciales son correctas, procedemos a borrar
this.apiService.delete('users/me').subscribe({ next: () => { this.dialogRef.close({
success: true, message: 'Cuenta eliminada permanentemente' }); }, error: (err) =>
{ this.handleError(err, 'Error al eliminar la cuenta'); } }); }, error: (err) => {
this.handleError(err, 'Credenciales incorrectas'); } }); } private
handleError(error: any, defaultMessage: string): void { this.isSubmitting = false;
console.error('Error:', error); // Manejo específico de errores HTTP if
(error.status === 0) { this.errorMessage = 'Error de conexión. Verifica tu conexión
a internet'; } else if (error.status === 401) { this.errorMessage = 'Credenciales
inválidas. Verifica tu email y contraseña'; } else if (error.error?.message) {
this.errorMessage = error.error.message; } else { this.errorMessage =
defaultMessage; } } onClose(): void { this.dialogRef.close(); } } import { NgModule
} from '@angular/core'; import { CommonModule } from '@angular/common';
import { DelProfileModalComponent } from './del-profile-modal.component';
import { FormsModule } from '@angular/forms'; import { MatDialogModule }
from '@angular/material/dialog'; @NgModule({ declarations:
[DelProfileModalComponent], imports: [ CommonModule, FormsModule,
MatDialogModule ] }) export class DelProfileModalModule { } import { NgModule
} from '@angular/core'; import { BrowserModule } from '@angular/platform-
browser'; import { AppRoutingModuleModule } from './app-routing.module'; import {
AppComponent } from './app.component'; import { LayoutModule } from
'./components/layout/layout/layout.module'; import { PagesModule } from
'./pages/pages.module'; import { provideHttpClient, withInterceptorsFromDi }
from '@angular/common/http'; import { HTTP_INTERCEPTORS } from
'@angular/common/http'; import { JwtInterceptor } from
'./shared/interceptors/jwt.interceptor'; @NgModule({ declarations: [
AppComponent ], imports: [ BrowserModule, AppRoutingModuleModule,
LayoutModule, PagesModule, ], providers:
[provideHttpClient(withInterceptorsFromDi() ), { provide:
HTTP_INTERCEPTORS, useClass: JwtInterceptor, multi: true }], bootstrap:
[AppComponent] }) export class AppModule { } //No está importado ni el de
borrado ni el de edición y el de edición no tiene problemas...

```

Solución Definitiva para el Error NG8002

El problema persiste porque aunque has creado `DelProfileModalModule` con `FormsModule`, no está siendo importado en ningún módulo que forme parte de tu aplicación. Veamos cómo solucionarlo:

1. Importa el módulo en tu estructura de módulos

Dado que no estás importando ni `DelProfileModalModule` ni `EdProfileModalModule` en `AppModule`, pero el de edición funciona, probablemente esté declarado en otro módulo. Para el componente de borrado:

Opción A: Importar en `PagesModule` o `LayoutModule`

```
typescript
// En pages.module.ts o layout.module.ts
import { DelProfileModalModule } from '../..components/ui/del-profile-modal/del-profile-modal.module';

@NgModule({
  imports: [
    // ... otros imports
    DelProfileModalModule
  ],
  // ...
})
export class PagesModule { } // o LayoutModule
```

Opción B: Declarar directamente en `PagesModule`

```
typescript
// En pages.module.ts
import { FormsModule } from '@angular/forms';
import { DelProfileModalComponent } from '../..components/ui/del-profile-modal/del-profile-modal.component';

@NgModule({
  declarations: [
    // ... otros componentes
    DelProfileModalComponent
  ],
  imports: [
    FormsModule,
    // ... otros imports
  ],
  // ...
})
export class PagesModule { }
```

2. Verifica cómo está funcionando el de edición

Revisa dónde está declarado `EdProfileModalComponent`. Probablemente está en `PagesModule` o `LayoutModule`, por eso funciona sin necesidad de importar su módulo.

3. Solución recomendada

La solución más limpia sería:

1. **Elimina DelProfileModalModule** (ya que no lo estás usando)
2. **Declara el componente directamente** en PagesModule:

typescript

```
// En pages.module.ts
import { DelProfileModalComponent } from '../..components/ui/del-profile-modal/del-profile-modal.component';

@NgModule({
  declarations: [
    // ... otros componentes
    DelProfileModalComponent
  ],
  imports: [
    FormsModule, // Asegúrate de que FormsModule esté importado aquí
    // ... otros imports
  ],
  // ...
})
export class PagesModule { }
```

4. Alternativa si prefieres mantener módulos separados

Si quieres mantener la estructura modular:

typescript

```
// En pages.module.ts
import { DelProfileModalModule } from '../..components/ui/del-profile-modal/del-profile-modal.module';

@NgModule({
  imports: [
    // ... otros imports
    DelProfileModalModule
  ],
  // ...
})
export class PagesModule { }
```

¿Por qué el de edición funciona sin esto?

Probablemente porque:

- EdProfileModalComponent está declarado directamente en PagesModule o LayoutModule
- O está siendo importado mediante otro módulo que ya incluye FormsModule

Verificación final

1. Ejecuta:

bash

ng serve

2. Si persiste el error, verifica:
 - Que no hay declaraciones duplicadas del componente
 - Que `FormsModule` está importado en el módulo que declara el componente
 - Que no hay errores de importación circular

Con estos cambios, el error debería resolverse y el componente de borrado funcionará correctamente con `ngModel`.

Tenías razón, había que importarle en el componente `profile`, en el padre, que era donde lo tenía importado, gracias.

¡Perfecto! ☐ Me alegra que hayas resuelto el problema.

Resumen de la Solución:

1. **El error ocurría** porque aunque el módulo `DelProfileModalModule` tenía `FormsModule`, no estaba siendo importado en el módulo donde realmente se usaba el componente.
2. **La solución fue** importar `DelProfileModalModule` en el módulo del componente padre (`ProfileModule`), que es donde se utilizaba el diálogo de borrado.

Lección clave para el futuro:

- ☐ Cuando uses componentes modales o diálogos:
- Asegúrate que su módulo esté importado en **el módulo del componente que lo llama** (no solo en el `AppModule`)
 - Verifica siempre la **jerarquía de imports** en Angular

Por qué funcionaba el de edición:

Probablemente porque `EdProfileModalModule` ya estaba importado en algún módulo padre compartido, mientras que el de borrado era nuevo y faltaba esa importación.

¡Excelente trabajo de debugging! ☐☐ Si surgen más problemas con Angular, ya sabes que los módulos y sus imports son siempre el primer lugar donde mirar.

Oye, el token no me lo está metiendo en el cuerpo, por qué era eso?

```
import { NgModule } from '@angular/core'; import { CommonModule } from '@angular/common'; import { DelProfileModalComponent } from './del-profile-modal.component'; import { FormsModule } from '@angular/forms'; import { MatDialogModule } from '@angular/material/dialog'; @NgModule({ declarations:
```

```

[DelProfileModalComponent], imports: [ CommonModule, FormsModule,
MatDialogModule ] }) export class DelProfileModalModule { } <div class="modal-
overlay" *ngIf="isVisible" (click)="onClose()"> <div class="modal-content"
(click)="$event.stopPropagation()"> <div class="modal-header bg-danger text-
white"> <h2>Borrar Cuenta Permanentemente</h2> <button class="close-btn
text-white" (click)="onClose()">&times;</button> </div> <div class="modal-
body"> <div class="alert alert-danger mb-4"> <strong>¡Advertencia!</strong>
Esta acción es irreversible. Todos tus datos serán eliminados permanentemente.
</div> <form (ngSubmit)="onSubmit()"> <div class="form-group mb-3"> <label
class="form-label">Ingresa tu email para confirmar</label> <input type="email"
class="form-control" [(ngModel)]="email" name="email"
placeholder="tu@email.com" required > <div class="text-danger mt-1"
*ngIf="errorMessage?.includes('email')"> {{errorMessage}} </div> </div> <div
class="form-group mb-4"> <label class="form-label">Ingresa tu
contraseña</label> <input type="password" class="form-control"
[(ngModel)]="password" name="password" placeholder="Tu contraseña actual"
required > <div class="text-danger mt-1"
*ngIf="errorMessage?.includes('contraseña') ||
errorMessage?.includes('Credenciales')"> {{errorMessage}} </div> </div> <div
class="modal-footer"> <button type="button" class="btn btn-outline-secondary"
(click)="onClose()" [disabled]="isSubmitting"> Cancelar </button> <button
type="submit" class="btn btn-danger" [disabled]="isSubmitting || !email ||
!password"> <span *ngIf="!isSubmitting">Confirmar Borrado</span> <span
*ngIf="isSubmitting"> <span class="spinner-border spinner-border-sm"
role="status" aria-hidden="true"></span> Procesando... </span> </button>
</div> </form> </div> </div> </div> import { Component, Inject } from
'@angular/core'; import { MAT_DIALOG_DATA, MatDialogRef } from
'@angular/material/dialog'; import { ApiService } from
'../shared/services/api.service'; import { Router } from '@angular/router';
import { AuthService } from '../shared/services/auth.service'; @Component({
selector: 'app-del-profile-modal', standalone: false, templateUrl: './del-profile-
modal.component.html', styleUrls: ['./del-profile-modal.component.scss'] }) export
class DelProfileModalComponent { email: string = ''; password: string = '';
isSubmitting = false; isVisible = true; errorMessage: string | null = null;
constructor( private apiService: ApiService, private authService: AuthService,
private dialogRef: MatDialogRef<DelProfileModalComponent>, private router:
Router, @Inject(MAT_DIALOG_DATA) public data: { currentEmail: string } ) {
onSubmit(): void { if (!this.email || !this.password) { this.errorMessage = 'Por favor
ingresa tu email y contraseña'; return; } if (this.email !== this.data.currentEmail) {
this.errorMessage = 'El email no coincide con tu cuenta'; return; } const
confirmDelete = confirm('¿Estás absolutamente seguro de que quieres eliminar tu
cuenta? Esta acción no se puede deshacer. '); if (!confirmDelete) { return; }
this.isSubmitting = true; this.errorMessage = null; this.isSubmitting = true; const
deleteData: any = {}; deleteData.email = this.email; deleteData.password =
this.password; this.authService.login(deleteData).subscribe({ next: () => { // Si las
credenciales son correctas, procedemos a borrar
this.apiService.deleteUser('users/me', deleteData).subscribe({ next: () => {
this.dialogRef.close({ success: true, message: 'Cuenta eliminada permanentemente'
}); }, error: (err) => { this.handleError(err, 'Error al eliminar la cuenta'); } }); },
error: (err) => { this.handleError(err, 'Credenciales incorrectas'); } }); } private

```



```
handleError(error: any, defaultMessage: string): void { this.isSubmitting = false;
console.error('Error:', error); // Manejo específico de errores HTTP if
(error.status === 0) { this.errorMessage = 'Error de conexión. Verifica tu conexión
a internet'; } else if (error.status === 401) { this.errorMessage = 'Credenciales
inválidas. Verifica tu email y contraseña'; } else if (error.error?.message) {
this.errorMessage = error.error.message; } else { this.errorMessage =
defaultMessage; } } onClose(): void { this.dialogRef.close(); } }
```

Problema con el Token en las Peticiones

El token JWT no se está enviando en las peticiones porque hay varios problemas en tu implementación:

1. Problema Principal: Falta el Interceptor JWT

En tu AppModule tienes configurado el JwtInterceptor, pero parece que no está funcionando correctamente. Revisa tu interceptor:

```
typescript
// jwt.interceptor.ts
import { Injectable } from '@angular/core';
import {
  HttpRequest,
  HttpHandler,
  HttpEvent,
  HttpInterceptor
} from '@angular/common/http';
import { Observable } from 'rxjs';
import { AuthService } from '../services/auth.service';

@Injectable()
export class JwtInterceptor implements HttpInterceptor {
  constructor(private authService: AuthService) {}

  intercept(request: HttpRequest<unknown>, next: HttpHandler):
  Observable<HttpEvent<unknown>> {
    const token = this.authService.getToken();

    if (token) {
      request = request.clone({
        headers: {
          Authorization: `Bearer ${token}`
        }
      });
    }

    return next.handle(request);
  }
}
```

2. Problema en el Servicio API

Veo que estás usando `deleteUser()` en lugar del método `delete()` estándar. Asegúrate que tu `ApiService` tenga:

typescript

```
// api.service.ts
delete(endpoint: string, data?: any): Observable<any> {
  return this.http.delete(`${this.apiUrl}/${endpoint}`, {
    body: data // Para enviar datos en una petición DELETE
  });
}
```

3. Solución Recomendada

Actualiza tu método `onSubmit()` en el componente:

typescript

```
onSubmit(): void {
  if (!this.email || !this.password) {
    this.errorMessage = 'Por favor ingresa tu email y contraseña';
    return;
  }

  if (this.email !== this.data.currentEmail) {
    this.errorMessage = 'El email no coincide con tu cuenta';
    return;
  }

  const confirmDelete = confirm('¿Estás absolutamente seguro de que quieres eliminar tu cuenta? Esta acción no se puede deshacer.');
```

if (!confirmDelete) return;

```
  this.isSubmitting = true;
  this.errorMessage = null;

  // Primero verifica credenciales
  this.authService.login(this.email, this.password).subscribe({
    next: (loginResponse) => {
      // Luego elimina la cuenta (el interceptor añadirá el token)
      this.apiService.delete('users/me').subscribe({
        next: () => {
          this.dialogRef.close({
            success: true,
            message: 'Cuenta eliminada permanentemente'
          });
        },
        error: (err) => this.handleError(err, 'Error al eliminar la cuenta')
      });
    },
    error: (err) => this.handleError(err, 'Credenciales incorrectas')
  });
}
```

4. Verifica tu AuthService

Asegúrate que tu `AuthService` tenga:

```
typescript
getToken(): string | null {
  return localStorage.getItem('auth_token'); // O donde guardes el token
}

login(email: string, password: string): Observable<any> {
  return this.apiService.post('auth/login', { email, password });
}
```

¿Por qué no se enviaba el token?

1. El interceptor no estaba procesando correctamente las peticiones
2. Usabas un método personalizado (deleteUser) en lugar del estándar
3. La estructura de la petición DELETE no era la adecuada para enviar datos

Con estos cambios, el token JWT debería incluirse automáticamente en todas las peticiones HTTP gracias al interceptor.

No, a ver el error es porque me falta algo por hacer para que se inyecten los headers. // services/api.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http'; import { Observable } from 'rxjs'; @Injectable({ providedIn: 'root' }) export class ApiService { private apiUrl = 'http://localhost:3000'; constructor(private http: HttpClient) { } // Métodos genéricos para diferentes tipos de solicitudes
get(endpoint: string): Observable<any> { return this.http.get(`${this.apiUrl}/${endpoint}`); }
post(endpoint: string, data: any): Observable<any> { return this.http.post(`${this.apiUrl}/${endpoint}`, data); }
put(endpoint: string, data: any): Observable<any> { return this.http.put(`${this.apiUrl}/${endpoint}`, data); }
patch(endpoint: string, data: any): Observable<any> { return this.http.patch(`${this.apiUrl}/${endpoint}`, data); }
deleteUser(endpoint: string, data: any): Observable<any> { return this.http.delete(`${this.apiUrl}/${endpoint}`, data); } } // services/auth.service.ts
import { Injectable } from '@angular/core';
import { Router } from '@angular/router'; import { ApiService } from './api.service'; import { BehaviorSubject, tap } from 'rxjs'; import { jwtDecode } from 'jwt-decode'; @Injectable({ providedIn: 'root' }) export class AuthService { private readonly TOKEN_KEY = 'auth_token'; private authStatus = new BehaviorSubject<boolean>(false); public isLoggedIn$ = this.authStatus.asObservable();
constructor( private apiService: ApiService, private router: Router, ) {this.authStatus.next(!this.getToken()); // Escucha cambios en otras pestañas
window.addEventListener('storage', this.handleStorageEvent.bind(this)); }
ngOnDestroy(): void { // Limpia el listener para evitar memory leaks
window.removeEventListener('storage', this.handleStorageEvent.bind(this)); }
private handleStorageEvent(event: StorageEvent): void { // Solo reacciona a cambios en nuestro token if (event.key === this.TOKEN_KEY) { const currentPath = this.router.url; const isAuthenticated = !!localStorage.getItem(this.TOKEN_KEY); // Actualiza el estado
this.authStatus.next(isAuthenticated); // Redirige si se perdió la autenticación y está en zona privada
if (!isAuthenticated && currentPath.startsWith('/private')) { this.router.navigate(['/login']); } } }
private isTokenExpired(token: string): boolean { try { const payload =
```

```

JSON.parse(atob(token.split('.')[1])); if (!payload.iat) return true; const
tokenLifetime = 3600; const expirationTime = payload.iat + tokenLifetime; //
Compara con el tiempo actual (conversión de ms a segundos) return
expirationTime < Math.floor(Date.now() / 1000); } catch { return true; } }
login(credentials: {email: string, password: string}) { return
this.apiService.post('auth/login', credentials).pipe( tap(response => { if
(!response.access_token) throw new Error('Invalid response');
this.saveToken(response.access_token); this.router.navigate(['/tutorial']); }) ); }
register(userData: any) { return this.apiService.post('auth/register',
userData).pipe( tap({ next: (response) => { }, error: (err) => { throw err; } }) ); }
saveToken(token: string): void { if (!token || this.isTokenExpired(token)) {
this.logout(); return; } localStorage.setItem(this.TOKEN_KEY, token);
this.authStatus.next(true); } getToken(): string | null { const token =
localStorage.getItem(this.TOKEN_KEY); if (!token || this.isTokenExpired(token))
{ this.logout(); return null; } return token } getDecodedToken(): any { const token
= this.getToken() if (token == null) { return null } return jwtDecode(token); }
isLoggedIn(): boolean { return this.authStatus.value; } logout(): void {
localStorage.removeItem(this.TOKEN_KEY); this.authStatus.next(false);
this.router.navigate(['/login']); } } import { Component } from '@angular/core';
import { AuthService } from '../shared/services/auth.service'; import { jwtDecode
} from 'jwt-decode'; import { ApiService } from '../shared/services/api.service';
import { MatDialog } from '@angular/material/dialog'; import {
EdProfileModalComponent } from '../components/ui/ed-profile-modal/ed-profile-
modal.component'; import { DelProfileModalComponent } from
'../components/ui/del-profile-modal/del-profile-modal.component'; import {
MatSnackBar } from '@angular/material/snack-bar'; @Component({ selector:
'app-profile', standalone: false, templateUrl: './profile.component.html', styleUrls:
['./profile.component.scss'] }) export class ProfileComponent { user = { id: 3, name:
'Alex García', email: 'alex@ejemplo.com', stats: { tasksCompleted: 0,
tasksPending: 0, categories: 0 }, }; isLoading = true; errorMessage: string | null =
null; constructor( private authService: AuthService, private apiService:
ApiService, private dialog: MatDialog, private snackBar: MatSnackBar ) {
ngOnInit(): void { this.loadUserData(); //this.loadStatistics(); } loadUserData():
void { this.isLoading = true; this.errorMessage = null; // Suponiendo que tienes un
método para obtener los datos del usuario logueado const cut =
this.authService.getDecodedToken(); //if (cut) { // this.user.id = cut.sub; //
this.user.name = cut.name; // this.user.email = cut.email; // } // <- Actualmente
desde el token podría sacarse todo //this.apiService.get(`users/id/${this.user.id}`) //
<- Este endpoint sirve pero mejor usar el seguro
this.apiService.get(`auth/getProfile`).subscribe({ // <- Este endpoint actualmente
saca la info del //propio JWT, lo que no es lo suyo para obtener más datos. Se
puede modificar de cara a futuro next: (userDetails) => { this.user = { ...this.user,
id: userDetails.sub, name: userDetails.name, email: userDetails.email, }
this.isLoading = false; }, error: (err) => { console.error('Error loading profile:',
err); this.errorMessage = 'Error al cargar el perfil'; this.isLoading = false; } }) }
openEditModal(): void { const dialogRef =
this.dialog.open(EdProfileModalComponent, { width: '500px', data: {
currentName: this.user.name } }); dialogRef.afterClosed().subscribe(result => { if
(result?.success) { this.user.name = result.name; // Actualiza el nombre si se editó }
}); } openDeleteModal(): void { const dialogRef =

```

```

this.dialog.open(DelProfileModalComponent, { width: '500px', data: {
currentEmail: this.user.email } }); dialogRef.afterClosed().subscribe(result => { if
(result?.success) { this.authService.logout(); this.snackBar.open(result.message ||
'Cuenta eliminada exitosamente', 'Cerrar', { duration: 5000, panelClass: ['success-
snackbar'] }); } }); } logout(): void { this.authService.logout(); } } import {
NgModule } from '@angular/core'; import { CommonModule } from
'@angular/common'; import { ProfileComponent } from './profile.component';
import { EdProfileModalModule } from '../components/ui/ed-profile-modal/ed-
profile-modal.module'; import { FormsModule } from '@angular/forms'; import {
DelProfileModalModule } from '../components/ui/del-profile-modal/del-profile-
modal.module'; @NgModule({ declarations: [ProfileComponent], imports: [
CommonModule, EdProfileModalModule, DelProfileModalModule, FormsModule
], exports: [ProfileComponent] }) export class ProfileModule { } import {
Component, Inject } from '@angular/core'; import { MAT_DIALOG_DATA,
MatDialogRef } from '@angular/material/dialog'; import { ApiService } from
'../shared/services/api.service'; import { Router } from '@angular/router';
import { AuthService } from '../shared/services/auth.service'; @Component({
selector: 'app-del-profile-modal', standalone: false, templateUrl: './del-profile-
modal.component.html', styleUrls: ['./del-profile-modal.component.scss'] }) export
class DelProfileModalComponent { email: string = ''; password: string = '';
isSubmitting = false; isVisible = true; errorMessage: string | null = null;
constructor( private apiService: ApiService, private authService: AuthService,
private dialogRef: MatDialogRef<DelProfileModalComponent>, private router:
Router, @Inject(MAT_DIALOG_DATA) public data: { currentEmail: string } ) {
onSubmit(): void { if (!this.email || !this.password) { this.errorMessage = 'Por favor
ingresa tu email y contraseña'; return; } if (this.email !== this.data.currentEmail) {
this.errorMessage = 'El email no coincide con tu cuenta'; return; } const
confirmDelete = confirm('¿Estás absolutamente seguro de que quieres eliminar tu
cuenta? Esta acción no se puede deshacer. '); if (!confirmDelete) { return; }
this.isSubmitting = true; this.errorMessage = null; this.isSubmitting = true; const
deleteData: any = {}; deleteData.email = this.email; deleteData.password =
this.password; this.authService.login(deleteData).subscribe({ next: () => { // Si las
credenciales son correctas, procedemos a borrar
this.apiService.deleteUser('users/me', deleteData).subscribe({ next: () => {
this.dialogRef.close({ success: true, message: 'Cuenta eliminada permanentemente'
}); }, error: (err) => { this.handleError(err, 'Error al eliminar la cuenta'); } }); },
error: (err) => { this.handleError(err, 'Credenciales incorrectas'); } }); } private
handleError(error: any, defaultMessage: string): void { this.isSubmitting = false;
console.error('Error:', error); // Manejo específico de errores HTTP if
(error.status === 0) { this.errorMessage = 'Error de conexión. Verifica tu conexión
a internet'; } else if (error.status === 401) { this.errorMessage = 'Credenciales
inválidas. Verifica tu email y contraseña'; } else if (error.error?.message) {
this.errorMessage = error.error.message; } else { this.errorMessage =
defaultMessage; } } onClose(): void { this.dialogRef.close(); } } import { NgModule
} from '@angular/core'; import { CommonModule } from '@angular/common';
import { DelProfileModalComponent } from './del-profile-modal.component';
import { FormsModule } from '@angular/forms'; import { MatDialogModule }
from '@angular/material/dialog'; @NgModule({ declarations:
[DelProfileModalComponent], imports: [ CommonModule, FormsModule,
MatDialogModule ] }) export class DelProfileModalModule { } import {

```

```

Component, Inject } from '@angular/core'; import { MAT_DIALOG_DATA,
MatDialogRef } from '@angular/material/dialog'; import { ApiService } from
'../shared/services/api.service'; @Component({ selector: 'app-edit-profile-
modal', standalone: false, templateUrl: './ed-profile-modal.component.html',
styleUrls: ['./ed-profile-modal.component.scss'] }) export class
EdProfileModalComponent { name: string = ''; newPassword: string = '';
isSubmitting = false; isVisible = true; passwordTouched = false; // Variables para
los requisitos de contraseña reqLength = false; reqLowerCase = false;
reqUpperCase = false; reqNumber = false; reqSpecialChar = false; constructor(
private apiService: ApiService, private dialogRef:
MatDialogRef<EdProfileModalComponent>, @Inject(MAT_DIALOG_DATA)
public data: { currentName: string } ) { this.name = data.currentName; }
checkPasswordRequirements(): void { const password = this.newPassword;
this.reqLength = password?.length >= 8; this.reqLowerCase = /[a-
z]/.test(password); this.reqUpperCase = /[A-Z]/.test(password); this.reqNumber =
/d/.test(password); this.reqSpecialChar = /[\\W_]/.test(password);
this.passwordTouched = !!password && password.length > 0; } get
passwordValid(): boolean { if (!this.newPassword) return true; return
this.reqLength && this.reqLowerCase && this.reqUpperCase &&
this.reqNumber && this.reqSpecialChar; } get formValid(): boolean { return
(this.name.length >= 3 || this.newPassword.length >= 8) && (this.newPassword ?
this.passwordValid : true); } onSubmit(): void { if (!this.formValid) return;
this.isSubmitting = true; const updateData: any = {}; if (this.name && this.name
!== this.data.currentName) { updateData.name = this.name; } if
(this.newPassword) { updateData.password = this.newPassword; } if
(Object.keys(updateData).length === 0) { this.dialogRef.close(); return; }
this.apiService.patch('users/me', updateData).subscribe({ next: () => {
this.dialogRef.close({ success: true, name: updateData.name ||
this.data.currentName }); }, error: (err) => { this.isSubmitting = false;
console.error('Error updating profile:', err); alert('Error updating profile: ' +
(err.error?.message || 'Please try again')); } }); } onClose(): void {
this.dialogRef.close(); } } import { NgModule } from '@angular/core'; import {
CommonModule } from '@angular/common'; import {
EdProfileModalComponent } from './ed-profile-modal.component'; import {
FormsModule } from '@angular/forms'; import { MatDialogModule } from
'@angular/material/dialog'; @NgModule({ declarations:
[EdProfileModalComponent], imports: [ CommonModule, FormsModule,
MatDialogModule ] }) export class EdProfileModalModule { } import { NgModule
} from '@angular/core'; import { BrowserModule } from '@angular/platform-
browser'; import { AppRoutingModuleModule } from './app-routing.module'; import {
AppComponent } from './app.component'; import { LayoutModule } from
'./components/layout/layout/layout.module'; import { PagesModule } from
'./pages/pages.module'; import { provideHttpClient, withInterceptorsFromDi }
from '@angular/common/http'; import { HTTP_INTERCEPTORS } from
'@angular/common/http'; import { JwtInterceptor } from
'../shared/interceptors/jwt.interceptor'; @NgModule({ declarations: [
AppComponent ], imports: [ BrowserModule, AppRoutingModuleModule,
LayoutModule, PagesModule, ], providers:
[provideHttpClient(withInterceptorsFromDi() ), { provide:

```

HTTP_INTERCEPTORS, useClass: JwtInterceptor, multi: true }], bootstrap: [AppComponent] }) export class AppModule { } En el de edición si me funciona.

Solución para el Problema del Token en las Peticiones DELETE

El problema principal es que tu interceptor JWT no está siendo aplicado correctamente a las peticiones DELETE. Veamos cómo solucionarlo:

1. Problema en el ApiService

El método `deleteUser` en tu `ApiService` está mal implementado. Para enviar datos en una petición DELETE con Angular, debes usar la opción `body`:

```
typescript
// api.service.ts
delete(endpoint: string, data?: any): Observable<any> {
  return this.http.delete(`${this.apiUrl}/${endpoint}`, {
    body: data // Esta es la forma correcta de enviar datos en DELETE
  });
}
```

2. Actualiza el Componente de Borrado

Cambia la llamada en `DelProfileModalComponent`:

```
typescript
// del-profile-modal.component.ts
this.apiService.delete('users/me', deleteData).subscribe({
  next: () => {
    this.dialogRef.close({
      success: true,
      message: 'Cuenta eliminada permanentemente'
    });
  },
  error: (err) => {
    this.handleError(err, 'Error al eliminar la cuenta');
  }
});
```

3. Verifica tu JwtInterceptor

Asegúrate que tu interceptor esté correctamente configurado:

```
typescript
// jwt.interceptor.ts
import { Injectable } from '@angular/core';
import {
  HttpRequest,
  HttpHandler,
  HttpEvent,

```

```

    HttpInterceptor
  } from '@angular/common/http';
import { Observable } from 'rxjs';
import { AuthService } from '../services/auth.service';

@Injectable()
export class JwtInterceptor implements HttpInterceptor {
  constructor(private authService: AuthService) {}

  intercept(request: HttpRequest<unknown>, next: HttpHandler):
  Observable<HttpEvent<unknown>> {
    // Clone la request para añadir headers
    const token = this.authService.getToken();

    if (token) {
      request = request.clone({
        headers: {
          Authorization: `Bearer ${token}`
        }
      });
    }

    return next.handle(request);
  }
}

```

4. Configuración en AppModule

Tu configuración en AppModule parece correcta:

```

typescript
// app.module.ts
providers: [
  provideHttpClient(withInterceptorsFromDi()),
  { provide: HTTP_INTERCEPTORS, useClass: JwtInterceptor, multi: true
}
]

```

¿Por qué funciona en edición y no en borrado?

1. **El método PATCH** está correctamente implementado en tu ApiService
2. **El método DELETE** tenía una implementación incorrecta que evitaba que el interceptor funcionara
3. **La estructura de la petición DELETE** es diferente a otras peticiones HTTP

Solución Completa

1. Elimina el método `deleteUser` y reemplázalo por `delete` en tu ApiService
2. Actualiza todas las llamadas DELETE para usar el nuevo método
3. Verifica que el token se esté guardando correctamente después del login

Verificación Final

Para confirmar que el token se está enviando:

1. Abre las Herramientas de Desarrollo del navegador (F12)
2. Ve a la pestaña Network (Red)
3. Filtra por peticiones "users/me"
4. Verifica que el header `Authorization` esté presente en ambas peticiones (PATCH y DELETE)

Con estos cambios, el token JWT debería incluirse correctamente en todas tus peticiones, incluyendo las DELETE.

```
// services/api.service.ts import { Injectable } from '@angular/core'; import {
HttpClient } from '@angular/common/http'; import { Observable } from 'rxjs';
@Injectable({ providedIn: 'root' }) export class ApiService { private apiUrl =
'http://localhost:3000'; constructor(private http: HttpClient) { } // Métodos
genéricos para diferentes tipos de solicitudes get(endpoint: string):
Observable<any> { return this.http.get(`${this.apiUrl}/${endpoint}`); }
post(endpoint: string, data: any): Observable<any> { return
this.http.post(`${this.apiUrl}/${endpoint}`, data); } put(endpoint: string, data:
any): Observable<any> { return this.http.put(`${this.apiUrl}/${endpoint}`, data); }
patch(endpoint: string, data: any): Observable<any> { return
this.http.patch(`${this.apiUrl}/${endpoint}`, data); } deleteUser(endpoint: string,
data: any): Observable<any> { return this.http.delete(`${this.apiUrl}/${endpoint}`,
{ body: data // Esta es la forma correcta de enviar datos en DELETE }); } } Y por
qué en el delete se hace diferente;?;?;?
```

Diferencia en el manejo de DELETE vs otros métodos HTTP

La diferencia en cómo se manejan los datos en las peticiones DELETE tiene que ver con las especificaciones HTTP y cómo Angular implementa estos métodos:

Razón técnica

1. **Especificación HTTP:** Según el estándar HTTP, los métodos GET y DELETE no deberían llevar un cuerpo (body) en la petición, mientras que POST, PUT y PATCH sí lo incluyen.
2. **Implementación de Angular:**
 - Para **POST/PUT/PATCH**, los datos van directamente como segundo parámetro:

typescript

- `post(endpoint: string, data: any)`

- Para **DELETE**, Angular requiere que los datos se envíen como parte de las opciones de configuración:

typescript

2.
 - o `delete(endpoint: string, options: { body?: any })`

Por qué esta diferencia

1. **Compatibilidad con estándares:** Algunos servidores no aceptan cuerpos en peticiones DELETE
2. **Flexibilidad:** La opción `body` permite enviar datos cuando el servidor lo soporte
3. **Consistencia con fetch API:** Este es el mismo enfoque que usa la API nativa `fetch()`

Solución recomendada

Para mantener consistencia en tu `ApiService`, puedes hacer lo siguiente:

```
typescript
// Versión mejorada del ApiService
@Injectable({
  providedIn: 'root'
})
export class ApiService {
  private apiUrl = 'http://localhost:3000';

  constructor(private http: HttpClient) { }

  // Método DELETE mejorado
  delete(endpoint: string, data?: any): Observable<any> {
    const options = data ? { body: data } : {};
    return this.http.delete(`${this.apiUrl}/${endpoint}`, options);
  }

  // Mantén los otros métodos igual
  get(endpoint: string): Observable<any> {
    return this.http.get(`${this.apiUrl}/${endpoint}`);
  }

  post(endpoint: string, data: any): Observable<any> {
    return this.http.post(`${this.apiUrl}/${endpoint}`, data);
  }

  // ... otros métodos
}
```

Cómo usar el método DELETE

```
typescript
// Sin datos
this.apiService.delete('users/123').subscribe(...);

// Con datos
this.apiService.delete('users/me', { reason: 'voluntary'
}).subscribe(...);
```

Buenas prácticas

1. Usa el nombre `delete` en lugar de `deleteUser` para mantener consistencia
2. Documenta el método para que otros desarrolladores sepan cómo usarlo:

typescript

```
• /**
 * Send DELETE request
 * @param endpoint API endpoint
 * @param data Optional data to send in request body
 */
delete(endpoint: string, data?: any): Observable<any>
```

- Considera usar interfaces para los datos:

typescript

```
3. delete(endpoint: string, data?: DeleteRequestData):
    Observable<any>
```

Esta implementación mantendrá la coherencia con los estándares HTTP mientras proporciona flexibilidad para aquellos casos donde necesites enviar datos en peticiones DELETE.

es que posiblemente necesite otro método delete para cuando no haya cuerpo de la petición. De todas formas, por qué se establecieron esos estándares si para eliminar una cuenta lo más normal es tener que introducir datos¿?

Explicación sobre los estándares HTTP y DELETE con cuerpo

¿Por qué DELETE es diferente?

Los estándares HTTP (RFC 7231) establecen que:

1. **GET y DELETE** no deberían llevar cuerpo por diseño original
2. **POST/PUT/PATCH** están diseñados para enviar datos

Sin embargo, en la práctica moderna:

Razones históricas del estándar:

1. **Simplicidad inicial:** GET/DELETE eran para operaciones simples
2. **Caché:** Los cuerpos en DELETE complican el caching
3. **Seguridad:** Algunos proxies/firewalls bloquean DELETE con cuerpo

Problema actual: Necesidad de enviar datos en DELETE

Para operaciones como borrar una cuenta (que necesita verificación), hay 3 enfoques comunes:

1. Usar DELETE con cuerpo (no estándar pero ampliamente soportado)

```
typescript
// ApiService
deleteWithBody(endpoint: string, data: any): Observable<any> {
  return this.http.delete(`${this.apiUrl}/${endpoint}`, { body: data });
}
```

2. Usar parámetros en la URL (estándar)

```
typescript
deleteAccount(password: string): Observable<any> {
  return
    this.http.delete(`${this.apiUrl}/users/me?password=${encodeURIComponent(password)}`);
}
```

3. Usar POST para operaciones complejas (recomendado REST)

```
typescript
postAccountDeletion(data: any): Observable<any> {
  return this.http.post(`${this.apiUrl}/users/me/deletion`, data);
}
```

Solución recomendada para tu caso

Implementa ambos métodos en tu ApiService:

```
typescript
// services/api.service.ts
@Injectable({ providedIn: 'root' })
export class ApiService {
  // ... otros métodos

  // DELETE estándar (sin cuerpo)
  delete(endpoint: string): Observable<any> {
    return this.http.delete(`${this.apiUrl}/${endpoint}`);
  }

  // DELETE con cuerpo (para casos especiales)
  deleteWithBody(endpoint: string, data: any): Observable<any> {
    return this.http.delete(`${this.apiUrl}/${endpoint}`, { body: data });
  }
}
```

¿Por qué funciona el de edición y no el de borrado?

1. **PATCH/PUT** están diseñados para enviar datos (cuerpo incluido)
2. **DELETE** tradicional no incluye cuerpo, por eso tu interceptor podría fallar

Conclusión

Para tu caso de borrado de cuenta, recomiendo:

1. Usa **deleteWithBody** para operaciones que necesiten verificación
2. Mantén **delete simple** para borrados sin datos
3. Considera usar **POST** para operaciones complejas de borrado (más RESTful)

Ejemplo de uso:

typescript

```
// Para borrar con verificación
this.apiService.deleteWithBody('users/me', { password: '123'
}).subscribe(...);

// Para borrado simple (sin verificación)
this.apiService.delete('posts/123').subscribe(...);
```

Esta aproximación te da flexibilidad manteniendo la claridad del código.

Una cosilla!, cuando hago la confirmación de borrado me redirige dos veces, una a una página y luego al login. Me gustaría que no se redirigiera a esa otra página, solo al login. Me ayudas a identificarlo?

```
import { Component, Inject } from '@angular/core';
import { MAT_DIALOG_DATA, MatDialogRef } from '@angular/material/dialog';
import { ApiService } from '../shared/services/api.service';
import { Router } from '@angular/router';
import { AuthService } from '../shared/services/auth.service';
@Component({
  selector: 'app-del-profile-modal',
  standalone: false,
  templateUrl: './del-profile-modal.component.html',
  styleUrls: ['./del-profile-modal.component.scss']
})
export class DelProfileModalComponent {
  email: string = '';
  password: string = '';
  isSubmitting = false;
  isVisible = true;
  errorMessage: string | null = null;

  constructor(
    private apiService: ApiService,
    private authService: AuthService,
    private dialogRef: MatDialogRef<DelProfileModalComponent>,
    private router: Router,
    @Inject(MAT_DIALOG_DATA) public data: { currentEmail: string }
  ) {}

  onSubmit(): void {
    if (!this.email || !this.password) {
      this.errorMessage = 'Por favor ingresa tu email y contraseña';
      return;
    }
    if (this.email !== this.data.currentEmail) {
      this.errorMessage = 'El email no coincide con tu cuenta';
      return;
    }
    const confirmDelete = confirm('¿Estás absolutamente seguro de que quieres eliminar tu cuenta? Esta acción no se puede deshacer. ');
    if (!confirmDelete) {
      return;
    }
    this.isSubmitting = true;
    this.errorMessage = null;
    this.isSubmitting = true;
    const deleteData: any = {};
    deleteData.email = this.email;
    deleteData.password = this.password;
    this.authService.login(deleteData).subscribe({
      next: () => {
        // Si las credenciales son correctas, procedemos a borrar
        this.apiService.deleteWithBody('users/me', deleteData).subscribe({
          next: () => {
            this.dialogRef.close({ success: true, message: 'Cuenta eliminada permanentemente' });
          },
          error: (err) => {
            this.handleError(err, 'Error al eliminar la cuenta');
          }
        });
      },
      error: (err) => {
        this.handleError(err, 'Credenciales incorrectas');
      }
    });
    private handleError(error: any, defaultMessage: string): void {
      this.isSubmitting = false;
      console.error('Error:', error);
      // Manejo específico de errores HTTP
      if (error.status === 0) {
        this.errorMessage = 'Error de conexión. Verifica tu conexión a internet';
      } else if (error.status === 401) {
        this.errorMessage = 'Credenciales';
      }
    }
  }
}
```

```

inválidas. Verifica tu email y contraseña'; } else if (error.error?.message) {
this.errorMessage = error.error.message; } else { this.errorMessage =
defaultMessage; } } onClose(): void { this.dialogRef.close(); } } <div class="modal-
overlay" *ngIf="isVisible" (click)="onClose()"> <div class="modal-content"
(click)="$event.stopPropagation()"> <div class="modal-header bg-danger text-
white"> <h2>Borrar Cuenta Permanentemente</h2> <button class="close-btn
text-white" (click)="onClose()">&times;</button> </div> <div class="modal-
body"> <div class="alert alert-danger mb-4"> <strong>¡Advertencia!</strong>
Esta acción es irreversible. Todos tus datos serán eliminados permanentemente.
</div> <form (ngSubmit)="onSubmit()"> <div class="form-group mb-3"> <label
class="form-label">Ingresa tu email para confirmar</label> <input type="email"
class="form-control" [(ngModel)]="email" name="email"
placeholder="tu@email.com" required > <div class="text-danger mt-1"
*ngIf="errorMessage?.includes('email')"> {{errorMessage}} </div> </div> <div
class="form-group mb-4"> <label class="form-label">Ingresa tu
contraseña</label> <input type="password" class="form-control"
[(ngModel)]="password" name="password" placeholder="Tu contraseña actual"
required > <div class="text-danger mt-1"
*ngIf="errorMessage?.includes('contraseña') ||
errorMessage?.includes('Credenciales')"> {{errorMessage}} </div> </div> <div
class="modal-footer"> <button type="button" class="btn btn-outline-secondary"
(click)="onClose()" [disabled]="isSubmitting"> Cancelar </button> <button
type="submit" class="btn btn-danger" [disabled]="isSubmitting || !email ||
!password"> <span *ngIf="!isSubmitting">Confirmar Borrado</span> <span
*ngIf="isSubmitting"> <span class="spinner-border spinner-border-sm"
role="status" aria-hidden="true"></span> Procesando... </span> </button>
</div> </form> </div> </div> </div>

```

Solución para la Redirección Doble

El problema de la doble redirección ocurre porque:

1. **El componente modal** (`DelProfileModalComponent`) está cerrando el diálogo y emitiendo un resultado
2. **El componente padre** (`ProfileComponent`) está manejando ese resultado y haciendo `logout()`
3. **El servicio `AuthService`** también tiene lógica de redirección en su método `logout()`

Pasos para solucionarlo:

1. Modifica el `DelProfileModalComponent`:

Elimina la inyección del `Router` ya que no debería manejar redirecciones (eso le corresponde al componente padre):

```

typescript
// Elimina esta línea del constructor
private router: Router,

```

2. Actualiza el AuthService:

Modifica el método `logout()` para que no haga redirección automática:

```
typescript
// auth.service.ts
logout(): void {
  localStorage.removeItem(this.TOKEN_KEY);
  this.authService.next(false);
  // Elimina esta línea:
  // this.router.navigate(['/login']);
}
```

3. Actualiza el ProfileComponent:

Aquí es donde debe manejarse la redirección:

```
typescript
// profile.component.ts
openDeleteModal(): void {
  const dialogRef = this.dialog.open(DelProfileModalComponent, {
    width: '500px',
    data: { currentEmail: this.user.email }
  });

  dialogRef.afterClosed().subscribe(result => {
    if (result?.success) {
      this.authService.logout();
      // Solo una redirección aquí
      this.router.navigate(['/login']);
      this.snackBar.open(result.message || 'Cuenta eliminada exitosamente', 'Cerrar', {
        duration: 5000,
        panelClass: ['success-snackbar']
      });
    }
  });
}
```

¿Por qué ocurría la doble redirección?

1. **Primera redirección:** Ocurría en el `AuthService.logout()`
2. **Segunda redirección:** Ocurría en el `ProfileComponent` después del `logout`

Beneficios de esta solución:

1. **Single Responsibility Principle:** Cada componente tiene una sola responsabilidad
 - El modal solo confirma y ejecuta el borrado
 - El servicio solo maneja la autenticación
 - El componente padre maneja el flujo de la UI
2. **Mayor control:** La navegación se maneja en un solo lugar (el componente padre)
3. **Más fácil de mantener:** Si necesitas cambiar el flujo, solo lo haces en un lugar

Verificación adicional:

Si aún ves comportamientos extraños, revisa:

1. **Event listeners:** En tu `AuthService` tienes un listener para cambios en el storage que podría estar disparando redirecciones
2. **Guards de ruta:** Verifica si tienes algún guard que redirija al perder autenticación
3. **Interceptores:** Revisa si algún interceptor HTTP está manejando errores 401/403 con redirecciones

Vale, el problema no es una doble redirección a la misma página. El problema que tengo es que cuando termino el borrado de cuenta me lleva a `/tutorial` y luego rápidamente a `/login`. ¿Por qué?

Solución para la Redirección a `/tutorial` antes del `/login`

El problema ocurre porque hay una redirección configurada en dos lugares diferentes:

Causa del Problema

1. En el `AuthService.login()` tienes una redirección a `/tutorial` cuando el login es exitoso:

typescript

```
login(credentials: {email: string, password: string}) {  
  return this.apiService.post('auth/login', credentials).pipe(  
    tap(response => {  
      if (!response.access_token) throw new Error('Invalid response');  
      this.saveToken(response.access_token);  
      this.router.navigate(['/tutorial']); // <-- ESTA ES LA CAUSA  
    })  
  );  
}
```

2. Luego en el `ProfileComponent` haces otra redirección a `/login` después del logout.

Solución Recomendada

1. Modifica el `AuthService.login()`

Elimina la redirección automática del método login:

typescript

```
// auth.service.ts  
login(credentials: {email: string, password: string}) {
```



```

return this.apiService.post('auth/login', credentials).pipe(
  tap(response => {
    if (!response.access_token) throw new Error('Invalid response');
    this.saveToken(response.access_token);
    // Elimina esta línea:
    // this.router.navigate(['/tutorial']);
  })
);
}

```

2. Maneja la redirección donde corresponda

Ahora puedes manejar la redirección directamente donde necesites:

```

typescript
// En tu componente donde haces login normalmente
this.authService.login(credentials).subscribe({
  next: () => {
    this.router.navigate(['/tutorial']); // Solo aquí si es necesario
  }
});

```

3. Para el flujo de borrado de cuenta

En tu `DelProfileModalComponent`, el flujo quedaría así:

```

typescript
this.authService.login(deleteData).subscribe({
  next: () => {
    this.apiService.deleteWithBody('users/me', deleteData).subscribe({
      next: () => {
        this.dialogRef.close({
          success: true,
          message: 'Cuenta eliminada permanentemente'
        });
        // No redirijas aquí, déjalo para el componente padre
      }
    });
  }
});

```

¿Por qué ocurría este comportamiento?

1. Primero hacías login (que redirigía a `/tutorial`)
2. Luego borrabas la cuenta y hacías logout (que redirigía a `/login`)
3. El flujo completo era: login → `/tutorial` → logout → `/login`

Beneficios de la solución:

1. **Control preciso:** Decides exactamente cuándo redirigir
2. **Flujo más limpio:** Cada operación tiene un propósito único
3. **Más mantenible:** Fácil de modificar el comportamiento de redirección

Verificación final:

Después de estos cambios:

1. Al borrar cuenta irá directamente a /login
2. Al hacer login normal podrás decidir a dónde redirigir
3. No habrá redirecciones intermedias no deseadas

`login(credentials: {email: string, password: string, redireccion = false}) { return this.authService.post('auth/login', credentials).pipe(tap(response => { if (!response.access_token) throw new Error('Invalid response'); this.saveToken(response.access_token); this.router.navigate(['/tutorial']); })); }` y algo como propiedad opcional? para no tener que quitar las redirecciones desde mi `auth.service`?

Solución con Redirección Opcional en el Login

Puedes implementar una solución elegante añadiendo un parámetro opcional para controlar la redirección. Aquí te muestro cómo:

1. Modifica el método `login` en `AuthService`

```
typescript
login(credentials: {email: string, password: string}, redirect:
boolean = true): Observable<any> {
  return this.authService.post('auth/login', credentials).pipe(
    tap(response => {
      if (!response.access_token) throw new Error('Invalid response');
      this.saveToken(response.access_token);

      // Redirige solo si redirect es true
      if (redirect) {
        this.router.navigate(['/tutorial']);
      }
    })
  );
}
```

2. Uso en el componente de borrado de cuenta

```
typescript
// En DelProfileModalComponent
this.authService.login(deleteData, false).subscribe({ // <-- false
para evitar redirección
next: () => {
  this.authService.deleteWithBody('users/me', deleteData).subscribe({
    next: () => {
      this.dialogRef.close({
        success: true,
        message: 'Cuenta eliminada permanentemente'
      });
    }
  });
}
```

```

        });
    }
    });
}
});

```

3. Uso normal (con redirección)

```

typescript
// En otros componentes donde quieres redirección normal
this.authService.login(credentials).subscribe(); // Redirige por defecto
// o explícitamente:
this.authService.login(credentials, true).subscribe();

```

Ventajas de esta solución:

1. **Mantienes el comportamiento existente** en la mayoría de los casos
2. **Control granular** donde necesites evitar la redirección
3. **Código más limpio** que usar un método separado
4. **Documentación implícita** del parámetro hace claro el propósito

Alternativa con objeto de opciones (más escalable)

Si necesitas más flexibilidad:

```

typescript
interface LoginOptions {
  redirect?: boolean;
  redirectUrl?: string;
}

login(credentials: {email: string, password: string}, options:
LoginOptions = { redirect: true, redirectUrl: '/tutorial' }):
Observable<any> {
  return this.apiService.post('auth/login', credentials).pipe(
    tap(response => {
      if (!response.access_token) throw new Error('Invalid response');
      this.saveToken(response.access_token);

      if (options.redirect) {
        this.router.navigate([options.redirectUrl]);
      }
    })
  );
}

// Uso:
this.authService.login(credentials, { redirect: false }).subscribe();
this.authService.login(credentials, { redirectUrl: '/dashboard'
}).subscribe();

```

Esta solución te da el máximo control mientras mantienes limpio tu servicio de autenticación.

A type literal property cannot have an initializer El problema es este mensaje de error.
¿No se puede?

Solución para el Error "A type literal property cannot have an initializer"

El error ocurre porque estás intentando asignar un valor por defecto directamente en la definición de tipo. En TypeScript, no puedes poner valores iniciales en las definiciones de tipo (type literals), pero hay varias formas de solucionarlo:

Forma Correcta de Hacerlo

Opción 1: Usa una interfaz con parámetros opcionales

typescript

```
interface LoginOptions {
  redirect?: boolean;
  redirectUrl?: string;
}

login(credentials: {email: string; password: string}, options:
LoginOptions = { redirect: true, redirectUrl: '/tutorial' }) {
  // Implementación
}
```

Opción 2: Define el objeto por defecto por separado

typescript

```
const DEFAULT_LOGIN_OPTIONS = {
  redirect: true,
  redirectUrl: '/tutorial'
};

login(credentials: {email: string; password: string}, options:
Partial<LoginOptions> = DEFAULT_LOGIN_OPTIONS) {
  // Implementación
}
```

Opción 3: Usa parámetros separados (la solución más simple para tu caso)

typescript

```
login(
  credentials: {email: string; password: string},
  redirect: boolean = true,
  redirectUrl: string = '/tutorial'
) {
  if (redirect) {
    this.router.navigate([redirectUrl]);
  }
  // Resto de la implementación
}
```

¿Por qué no funciona tu versión original?

TypeScript no permite esto:

```
typescript
// Esto genera el error "A type literal property cannot have an
initializer"
login(credentials: {email: string, password: string, redireccion =
false})) {
  // ...
}
```

Porque `{email: string, password: string, redireccion = false}` es una `type` literal (definición de tipo en línea) y no puede contener inicializadores.

Solución Recomendada para tu Caso

La forma más limpia sería:

```
typescript
login(
  credentials: {email: string; password: string},
  options: { redirect?: boolean } = { redirect: true }
) {
  return this.apiService.post('auth/login', credentials).pipe(
    tap(response => {
      if (!response.access_token) throw new Error('Invalid response');
      this.saveToken(response.access_token);

      if (options.redirect) {
        this.router.navigate(['/tutorial']);
      }
    })
  );
}

// Uso:
this.authService.login(credentials); // Redirige por defecto
this.authService.login(credentials, { redirect: false }); // Sin
redirección
```

Esta solución:

1. Es type-safe (TypeScript la entenderá perfectamente)
2. Mantiene limpio el código
3. Permite extensibilidad para añadir más opciones en el futuro
4. Sigue el patrón común de opciones como objeto en JavaScript/TypeScript