Secuencia de trabajo de editar prestdor:

 **El usuario abre el modal**

* Está viendo la lista de proveedores (ProviderList) y hace clic en *Editar* en la fila de un proveedor.
* El componente padre (ProviderList) llama a openModal pasándole el proveedor seleccionado.
* El ProviderModal se abre y, en su useEffect, carga los datos actuales del proveedor en el estado local del formulario.

 **El usuario elige qué hacer con la imagen**

* El modal muestra tres opciones: *Mantener*, *Reemplazar*, *Quitar*.
* Si elige **Mantener**, no se hace nada con la imagen.
* Si elige **Reemplazar**, aparece el componente ImageUpload y el usuario selecciona un archivo (File).
* Si elige **Quitar**, se marca la opción y no se enviará archivo.

 **El usuario hace clic en “Guardar”**

* El handleSave del modal se ejecuta.
* Se construye un objeto providerData con los campos del formulario (name, lastName, address, phone, description, isActive, categoryId).
* Se crea un FormData y se agrega el provider como Blob JSON (FormData.append("provider", Blob)).
* Según la opción elegida en imageAction:
  + **replace**: se añade también el archivo (FormData.append("image", File)) y imageAction="replace".
  + **remove**: se añade solo imageAction="remove".
  + **keep**: se añade solo imageAction="keep".

 **El modal llama al hook useUpdateProvider**

* Este hook llama al servicio editProviderService pasando el FormData y la bandera isMultipart=true.
* El servicio hace un axios.put al endpoint /api/providers/update/{id} con Content-Type: multipart/form-data.

 **El backend recibe la petición**

* El ProvidersController tiene un @PutMapping que recibe:
  + @RequestPart("provider") → DTO con los datos del proveedor.
  + @RequestPart(value="image", required=false) → archivo de imagen (si llega).
  + @RequestPart("imageAction") → texto "keep", "replace" o "remove".
* El controlador llama al UpdateProviderService.

 **El servicio de actualización procesa la lógica**

* Busca el proveedor en la base de datos (findById).
* Si cambió categoryId, busca la nueva categoría y la asigna.
* Actualiza los campos básicos (name, lastName, address, etc.).
* Según el valor de imageAction:
  + **replace** y archivo presente: guarda la imagen en disco (con storeImage), obtiene su URL y actualiza photoUrl.
  + **remove**: asigna photoUrl="".
  + **keep**: no modifica la imagen.
* Guarda el proveedor actualizado (save).
* Convierte la entidad en DTO (providerMapper.toDTO) y lo retorna.

 **Respuesta al frontend**

* El backend responde 200 OK con el ProviderResponseDTO.
* El editProviderService devuelve estos datos al hook, y el hook los devuelve al modal.

 **Actualización de la UI**

* El modal muestra un mensaje de éxito con Swal.fire.
* Llama a la función onSuccess que le pasó el padre (ProviderList).
* El ProviderList ejecuta table.refresh(), que vuelve a pedir la lista de proveedores al backend y actualiza la tabla.
* Finalmente, el modal se cierra.