

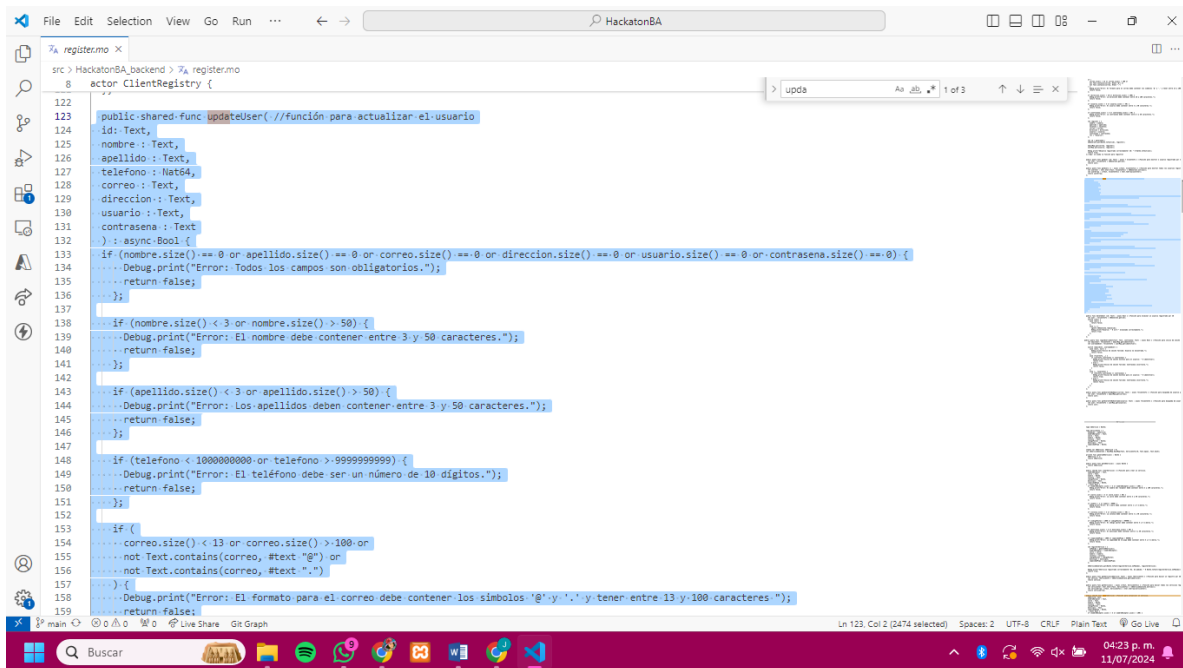
DOCUMENTACION DE PRUEBAS BACKEND

CRUD

- **UPDATE**

Esta función se utiliza para actualizar la información de un usuario en el sistema. Primero, verifica que todos los campos requeridos no estén vacíos y que cumplan con ciertos criterios de formato y longitud. Por ejemplo, se asegura de que el nombre y el apellido tengan un número adecuado de caracteres, que el correo electrónico tenga el formato correcto, y que el número de teléfono sea válido.

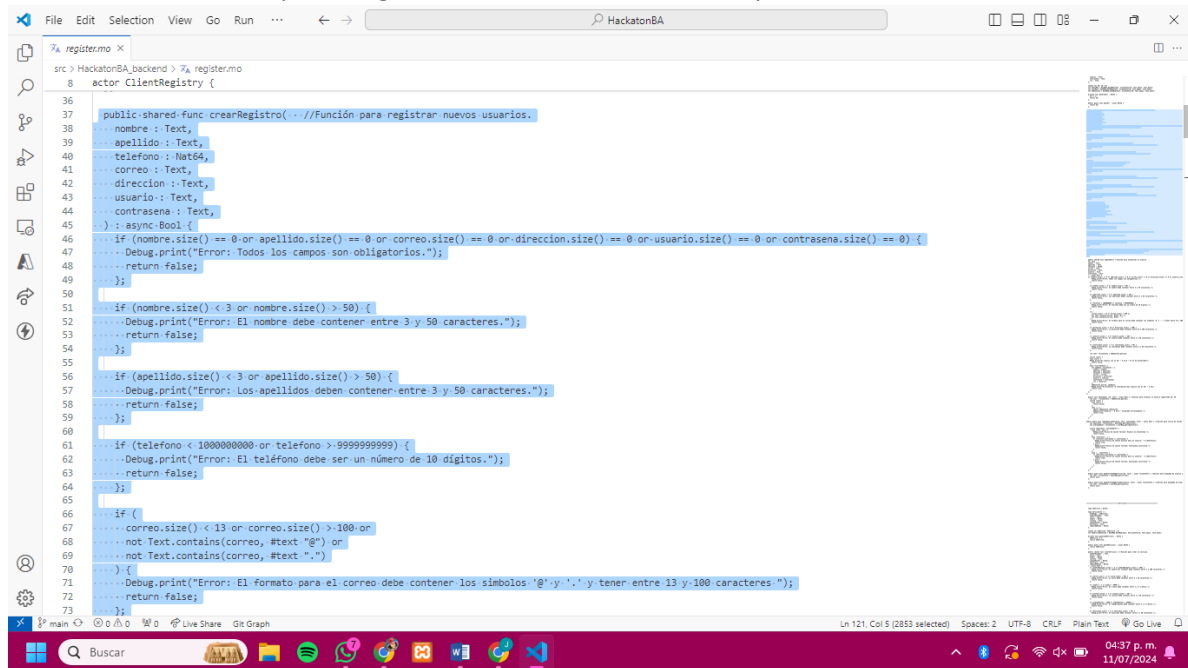
Una vez que se realizan todas estas verificaciones, la función busca al usuario en la base



```
File Edit Selection View Go Run ... HacktonBA
registro.mong
src > HacktonBA_backend > registro.mong
8 actor ClientRegistry {
122
123   public shared func updateUser( //función para actualizar el usuario
124     id: Text,
125     nombre: Text,
126     apellido: Text,
127     telefono: Nat64,
128     correo: Text,
129     direccion: Text,
130     usuario: Text,
131     contrasena: Text
132   ) : async Bool {
133     if (nombre.size() == 0 or apellido.size() == 0 or correo.size() == 0 or direccion.size() == 0 or usuario.size() == 0 or contrasena.size() == 0) {
134       Debug.print("Error: Todos los campos son obligatorios.");
135       return false;
136     };
137
138     if (nombre.size() < 3 or nombre.size() > 50) {
139       Debug.print("Error: El nombre debe contener entre 3 y 50 caracteres.");
140       return false;
141     };
142
143     if (apellido.size() < 3 or apellido.size() > 50) {
144       Debug.print("Error: Los apellidos deben contener entre 3 y 50 caracteres.");
145       return false;
146     };
147
148     if (telefono < 1000000000 or telefono > 9999999999) {
149       Debug.print("Error: El teléfono debe ser un número de 10 dígitos.");
150       return false;
151     };
152
153     if (
154       correo.size() < 13 or correo.size() > 100 or
155       not Text.contains(correo, #text "@" or
156       not Text.contains(correo, #text "."))
157     ) {
158       Debug.print("Error: El formato para el correo debe contener los símbolos '@' y '.' y tener entre 13 y 100 caracteres.");
159       return false;
160     };
161   }
162 }
```

- **CREATE**

Esta función registra nuevos usuarios en el sistema. Primero, verifica que toda la información proporcionada cumpla con ciertos criterios (por ejemplo, que los campos no estén vacíos y que el formato del correo electrónico sea correcto). Si todas las verificaciones son exitosas, se crea un nuevo registro de usuario con un ID único y se guarda en la base de datos. También se añaden entradas en mapas de correo electrónico y nombre de usuario para facilitar futuras búsquedas. Finalmente, confirma que el registro se realizó correctamente y devuelve true.

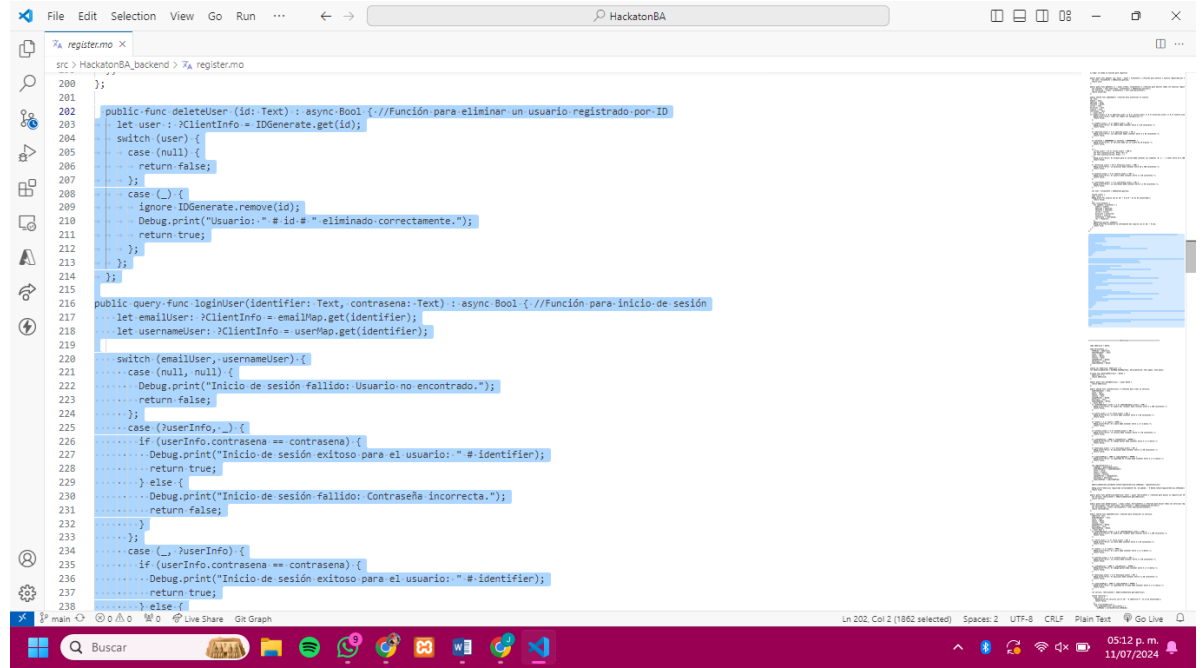


```
src > HackatonBA_backend > %A registro.mo
8  actor ClientRegistry {

36
37     public shared func crearRegistro( //Función para registrar nuevos usuarios.
38         nombre : Text,
39         apellido : Text,
40         telefono : Nat64,
41         correo : Text,
42         direccion : Text,
43         usuario : Text,
44         contrasena : Text,
45     ) : async Bool {
46         if (nombre.size() == 0 or apellido.size() == 0 or correo.size() == 0 or direccion.size() == 0 or usuario.size() == 0 or contrasena.size() == 0) {
47             Debug.print("Error: Todos los campos son obligatorios.");
48             return false;
49         };
50
51         if (nombre.size() < 3 or nombre.size() > 50) {
52             Debug.print("Error: El nombre debe contener entre 3 y 50 caracteres.");
53             return false;
54         };
55
56         if (apellido.size() < 3 or apellido.size() > 50) {
57             Debug.print("Error: Los apellidos deben contener entre 3 y 50 caracteres.");
58             return false;
59         };
60
61         if (telefono < 1000000000 or telefono > 9999999999) {
62             Debug.print("Error: El teléfono debe ser un número de 10 dígitos.");
63             return false;
64         };
65
66         if (
67             correo.size() < 13 or correo.size() > 100 or
68             not Text.contains(correo, #text "@" ) or
69             not Text.contains(correo, #text "." )
70         ) {
71             Debug.print("Error: El formato para el correo debe contener los símbolos '@' y '.' y tener entre 13 y 100 caracteres.");
72             return false;
73         };
74     }
75 }
```

- **DELETE**

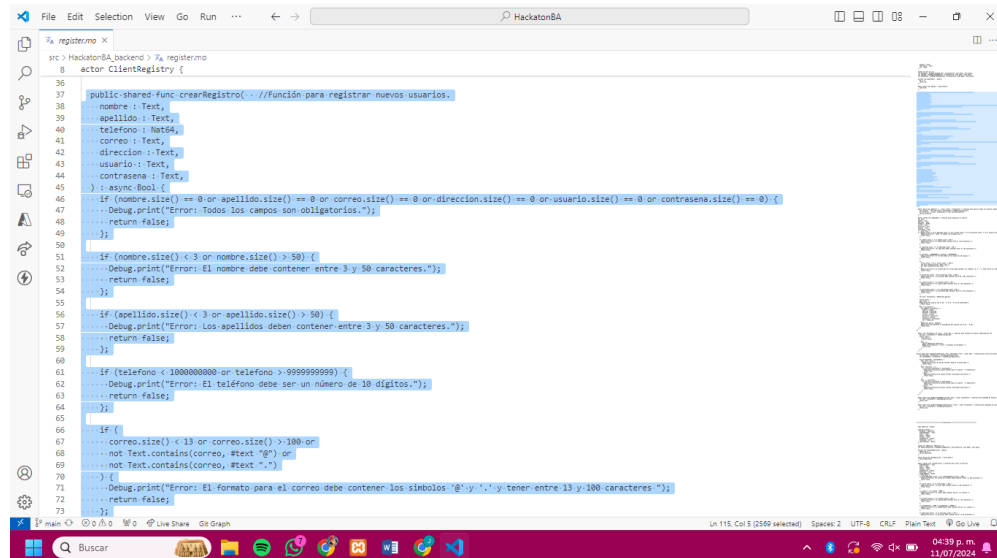
Esta función elimina a un usuario registrado del sistema utilizando su ID. Primero, verifica si el usuario existe en la base de datos. Si se encuentra, se elimina y se confirma la operación. Si no se encuentra, la función devuelve `false`, indicando que el usuario no pudo ser eliminado.



```
src > HackatonBA_backend > %a register.mo
200  });
201
202  public func deleteUser (id: Text) : async Bool { //Función para eliminar un usuario registrado por ID
203      let user = >ClientInfo = IDGenerate.getId(id);
204      switch (user) {
205          case (null) {
206              return false;
207          };
208          case (.) {
209              ignore IDGenerate.remove(id);
210              Debug.print("Usuario: " # id # " eliminado correctamente.");
211              return true;
212          };
213      };
214  };
215
216  public query func loginUser (identifier: Text, contrasena: Text) : async Bool { //Función para inicio de sesión
217      let emailUser = >ClientInfo = emailMap.get(identifier);
218      let usernameUser = >ClientInfo = userMap.get(identifier);
219
220      switch (emailUser, usernameUser) {
221          case (null, null) {
222              Debug.print("Inicio de sesión fallido: Usuario no encontrado.");
223              return false;
224          };
225          case (userInfo, _) {
226              if (userInfo.contrasena == contrasena) {
227                  Debug.print("Inicio de sesión exitoso para el usuario: " # identifier);
228                  return true;
229              } else {
230                  Debug.print("Inicio de sesión fallido: Contraseña incorrecta.");
231                  return false;
232              };
233          };
234          case (_, userInfo) {
235              if (userInfo.contrasena == contrasena) {
236                  Debug.print("Inicio de sesión exitoso para el usuario: " # identifier);
237                  return true;
238              } else {
239                  Debug.print("Inicio de sesión fallido: Contraseña incorrecta.");
240                  return false;
241              };
242          };
243      };
244  };
245  }
```

- **REGISTER**

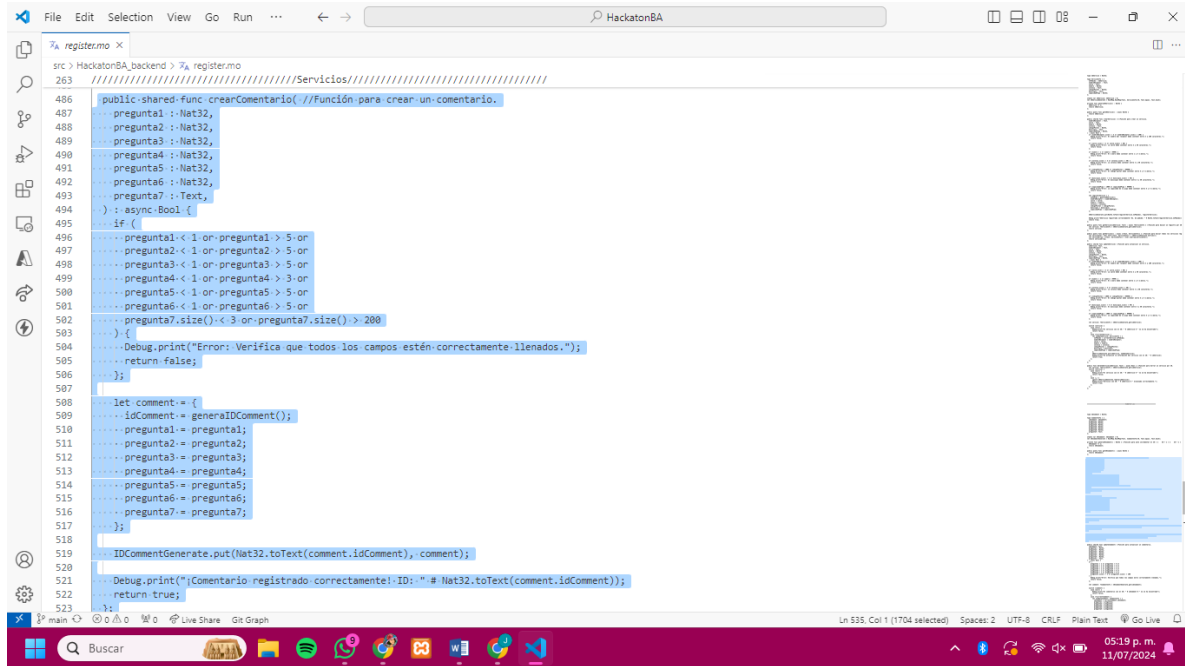
Esta función registra nuevos usuarios en el sistema y valida que toda la información proporcionada sea correcta. Si todo está en orden, crea un nuevo usuario y lo guarda en la base de datos, asignándole un ID único. También actualiza los registros de correo electrónico y nombre de usuario para facilitar futuras búsquedas. Además, se incluyen funciones para recuperar la información de un usuario específico o listar todos los usuarios registrados.



```
src > HackatonBA_backend > %a register.mo
8  actor ClientRegistry {
36
37  public shared func createRegistro (... //Función para registrar nuevos usuarios.
38      nombre : Text,
39      apellido : Text,
40      telefono : Nat64,
41      correo : Text,
42      direccion : Text,
43      usuario : Text,
44      contrasena : Text,
45  ) : async Bool {
46      if (nombre.size() == 0 or apellido.size() == 0 or correo.size() == 0 or direccion.size() == 0 or usuario.size() == 0 or contrasena.size() == 0) {
47          Debug.print("Error: Todos los campos son obligatorios.");
48          return false;
49      };
50
51      if (nombre.size() < 3 or nombre.size() > 50) {
52          Debug.print("Error: El nombre debe contener entre 3 y 50 caracteres.");
53          return false;
54      };
55
56      if (apellido.size() < 3 or apellido.size() > 50) {
57          Debug.print("Error: Los apellidos deben contener entre 3 y 50 caracteres.");
58          return false;
59      };
60
61      if (telefono < 1000000000 or telefono > 9999999999) {
62          Debug.print("Error: El teléfono debe ser un número de 10 dígitos.");
63          return false;
64      };
65
66      if (
67          correo.size() < 13 or correo.size() > 100 or
68          not Text.contains(correo, #text "g") or
69          not Text.contains(correo, #text ".")
70      ) {
71          Debug.print("Error: El formato para el correo debe contener los símbolos '@' y '.' y tener entre 13 y 100 caracteres.");
72          return false;
73      };
74  };
75  }
```

COMENTARIOS

Estas funciones permiten registrar, acceder y gestionar comentarios dentro del sistema de manera eficiente, facilitando tanto la entrada como la consulta de información detallada sobre los comentarios registrados.

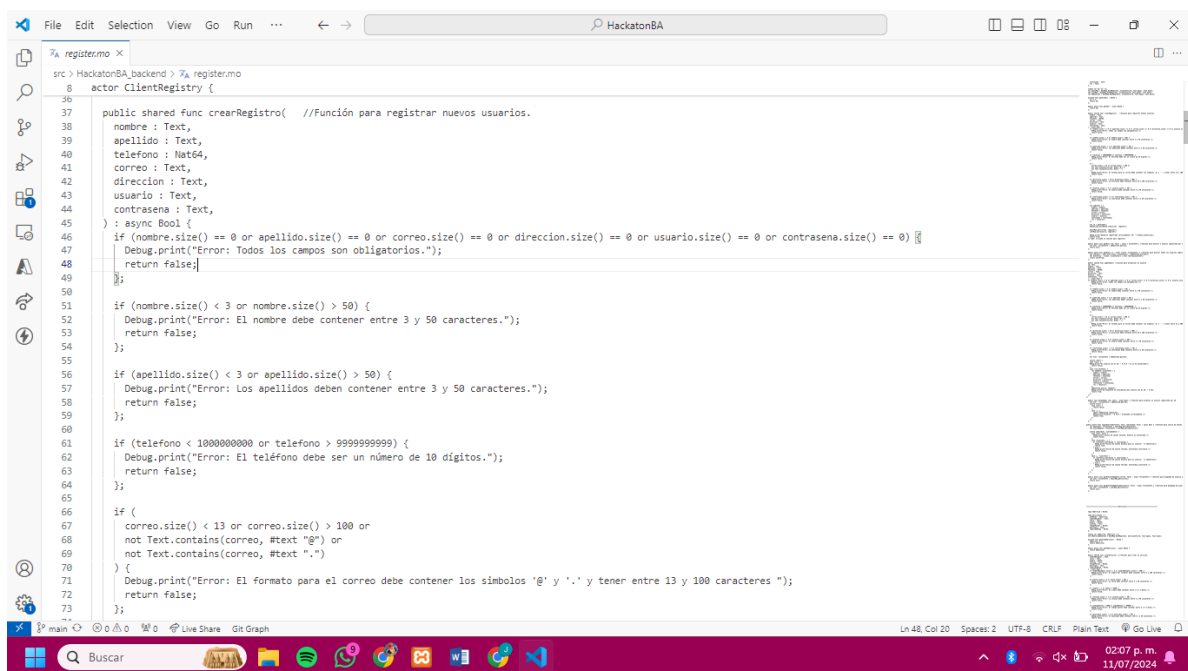


```
src > HackatonBA_backend > %A register.mo
263 ///////////////////////////////////////////////////////////////////Servicios/////////////////////////////////////////////////////////////////
486 public shared func crearComentario( //Función para crear un comentario.
487   pregunta1 : Nat32,
488   pregunta2 : Nat32,
489   pregunta3 : Nat32,
490   pregunta4 : Nat32,
491   pregunta5 : Nat32,
492   pregunta6 : Nat32,
493   pregunta7 : Text,
494   ) : async Bool {
495   if (
496     pregunta1 < 1 or pregunta1 > 5 or
497     pregunta2 < 1 or pregunta2 > 5 or
498     pregunta3 < 1 or pregunta3 > 5 or
499     pregunta4 < 1 or pregunta4 > 5 or
500     pregunta5 < 1 or pregunta5 > 5 or
501     pregunta6 < 1 or pregunta6 > 5 or
502     pregunta7.size() < 3 or pregunta7.size() > 200
503   ) {
504     Debug.print("Error: Verifica que todos los campos estén correctamente llenados.");
505     return false;
506   }
507   let comment = {
508     idComment := generaIDComment();
509     pregunta1 := pregunta1;
510     pregunta2 := pregunta2;
511     pregunta3 := pregunta3;
512     pregunta4 := pregunta4;
513     pregunta5 := pregunta5;
514     pregunta6 := pregunta6;
515     pregunta7 := pregunta7;
516   };
517   IDCommentGenerate.put(Nat32.toText(comment.idComment), comment);
518   Debug.print("Comentario registrado correctamente! ID: " # Nat32.toText(comment.idComment));
519   return true;
520 }
```

LOGIN

- **Registro**

Esta función se encarga de registrar un nuevo usuario verificando que todos los datos proporcionados sean válidos y cumplan con ciertos criterios. Si todo está en orden, se guarda el registro y se genera un ID único para el usuario.

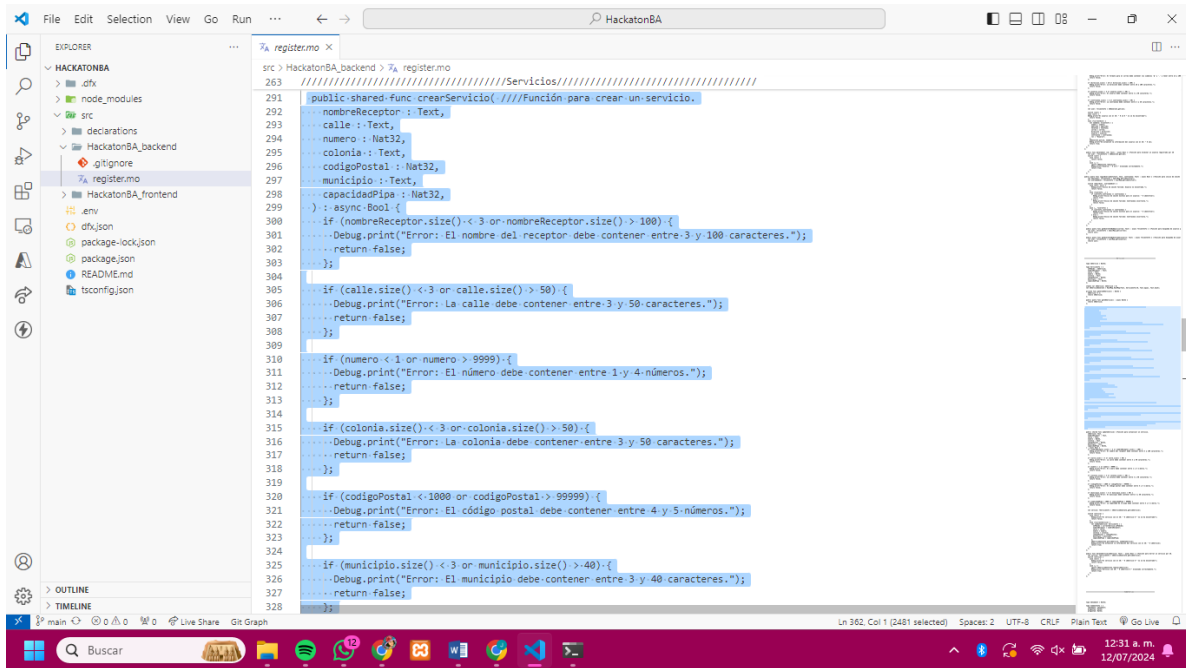


```
src > HackatonBA_backend > %A register.mo
8 actor ClientRegistry {
36
37 public shared func crearRegistro( //Función para registrar nuevos usuarios.
38   nombre : Text,
39   apellido : Text,
40   telefono : Nat64,
41   correo : Text,
42   direccion : Text,
43   usuario : Text,
44   contrasena : Text,
45   ) : async Bool {
46   if (nombre.size() == 0 or apellido.size() == 0 or correo.size() == 0 or direccion.size() == 0 or usuario.size() == 0 or contrasena.size() == 0) {
47     Debug.print("Error: Todos los campos son obligatorios.");
48     return false;
49   }
50
51   if (nombre.size() < 3 or nombre.size() > 50) {
52     Debug.print("Error: El nombre debe contener entre 3 y 50 caracteres.");
53     return false;
54   }
55
56   if (apellido.size() < 3 or apellido.size() > 50) {
57     Debug.print("Error: Los apellidos deben contener entre 3 y 50 caracteres.");
58     return false;
59   }
60
61   if (telefono < 1000000000 or telefono > 9999999999) {
62     Debug.print("Error: El teléfono debe ser un número de 10 dígitos.");
63     return false;
64   }
65
66   if (
67     correo.size() < 13 or correo.size() > 100 or
68     not Text.contains(correo, #text "@") or
69     not Text.contains(correo, #text ".")
70   ) {
71     Debug.print("Error: El formato para el correo debe contener los símbolos '@' y '.' y tener entre 13 y 100 caracteres ");
72     return false;
73   }
74 }
```

SERVICIOS

Este código se encarga de gestionar el registro y consulta de servicios de pipas de agua, verifica que los datos proporcionados sean correctos.

Si los datos son válidos, registra un nuevo servicio y le asigna un ID único.



```
src > HackatonBA_backend > %A register.mo
263 ///////////////////////////////////////////////////Servicios////////////////////////////////////
291 public shared func crearServicio( //Función para crear un servicio.
292     nombreReceptor : Text,
293     calle : Text,
294     numero : Nat32,
295     colonia : Text,
296     codigoPostal : Nat32,
297     municipio : Text,
298     capacidadPipa : Nat32,
299     ) : async Bool {
300     if (nombreReceptor.size() < 3 or nombreReceptor.size() > 100) {
301         Debug.print("Error: El nombre del receptor debe contener entre 3 y 100 caracteres.");
302         return false;
303     };
304     if (calle.size() < 3 or calle.size() > 50) {
305         Debug.print("Error: La calle debe contener entre 3 y 50 caracteres.");
306         return false;
307     };
308     if (numero < 1 or numero > 9999) {
309         Debug.print("Error: El número debe contener entre 1 y 4 números.");
310         return false;
311     };
312     if (colonia.size() < 3 or colonia.size() > 50) {
313         Debug.print("Error: La colonia debe contener entre 3 y 50 caracteres.");
314         return false;
315     };
316     if (codigoPostal < 1000 or codigoPostal > 9999) {
317         Debug.print("Error: El código postal debe contener entre 4 y 5 números.");
318         return false;
319     };
320     if (municipio.size() < 3 or municipio.size() > 40) {
321         Debug.print("Error: El municipio debe contener entre 3 y 40 caracteres.");
322         return false;
323     };
324     //
325     //
326     //
327     //
328     //
```