## Requerimientos

Usuario Normal.

●        Puede Registrarse.

●        Iniciar y cerrar sesión.

●        Editar información de usuario, asi como foto de perfil.

●        Puede ver perfiles de otros usuarios, asi como iniciar una amistad.

●        Eliminar amistades.

●        Buscar lugares mediante un filtro de tipo de lugar.

●        Ver información relacionada con el lugar.

●        Realizar una invitación a lugares con amigos.

●        La invitación puede ser aceptada o rechazada.

●        Solicitar ser un propietario.

●        Reportar un lugar.

Propietario

●        Registrar y solicitar su aprobación lugar.

●        Editar información del lugar.

●        Subir fotos y/o menú.

●        Eliminar lugar.

●        Responder reporte.

Administrador

●        Aceptar y rechazar lugar.

●        Aceptar y rechazar solicitud de propietario.

●        Revisar reportes.

●        Ver reporte.

●        Confirmar reporte.

●        Eliminar reporte.

●        CRUD categoría.

## Apps

|  |  |
| --- | --- |
| Usuario | Lugar |
| UsuariotipoUsuariolistaAmigosInvitacionNotificacionStatusUsuario\_notificacionUsuario\_invitacionUsuario\_lugar | LugarCategoríaReporteReporte\_lugar |

## Funciones

### Funciones de usuarios

* **Registro:**
  + *Front-end:* El usuario ingresará sus datos en 6 inputs (nombre de usuario, contraseña, confirmación de contraseña, nombre, apellidos y correo). Antes de realizar el POST, JS verificará que los campos: contraseña y confirmación de contraseña sean iguales, además de que se hayan llenado todos los demás campos,  si esto no ocurre se mandará error.
  + *Back-end:* Se verificará que no exista un usuario con el nombre de usuario y/o contraseña ingresados. Se utilizará la función placeholderProfilePicture para asignar una foto de perfil. Se creará el usuario con los datos ingresados y tipo: usuario, status: ninguno, descripción: null y urlFotoPerfil como el resultado de la función palceholderPP.
* **Login:**
  + *Front-end:* El usuario ingresará su usuario y contraseña. JS verificará que los campos no estén vacios.
  + *Back-end:* Se utilizará el método authenticate y login para iniciar sesión. También se crearán variables de sesión con el username, el id del usuario en la dB y la url de su foto de perfil
* **Cerrar sesión:**
  + *Front-end:* El usuario presionará un botón para cerrar sesión
  + *Back-end:* Se utilizará el método logout y se redirigirá a la página de login
* **Editar información de usuario:**
  + *Front-end:* Al usuario se le presentarán contenedores con su información (correo, nombre y apellidos), este podrá presionar un botón para poder cambiar su información. Al cambiarla, este presionará un botón para confirmar el cambio y después otro botón para guardar todos los cambios que el usuario haya hecho. JS creará un diccionario con la información que el usuario haya cambiado, la llave del diccionario será el dato que se haya cambiado (p. ej. nuevoNombre: “panchito”,)
  + *Back-end:* Al recibir el diccionario, el servidor recorrerá todas las llaves y utilizando un switch realizará las actualizaciones necesarias en la base de datos.
* **Editar contraseña de usuario:** 
  + *Front-end:* Elusuario presionará un botón para cambiar su contraseña, causando que aparezca un modal donde se ingresará su nueva contraseña y una confirmación de la nueva contraseña. JS verificará que la nueva contraseña y la confirmación sean las mismas.
  + *Back-end:* Se utilizaran los métodos set\_password y update\_session\_auth\_hash para actualizar la contraseña.
* **Editar imagen de perfil:**
  + *Front-end:* El usuario presionará un botón para cambiar su imagen, esto abrirá un cuadro de diálogo para seleccionar su imagen. Al cambiarla, este presionará un guardar todos los cambios que el usuario haya hecho. JS creará un diccionario con la información que el usuario haya cambiado, la llave del diccionario será el dato que se haya cambiado (p. ej. nuevoNombre: “panchito”,)
  + *Back-end:* Al recibir el diccionario, el servidor recorrerá todas las llaves y utilizando un switch invocará a la función handle\_uploaded\_file
* **Iniciar amistad:**
  + *Front-end:* El usuario presionará un botón para iniciar una amistad. JS mandará el ID del usuario al que se le mandó la solicitud
  + *Back-end:* Se crearán dos elementos en la tabla listaAmigos en el primero el usuario\_id será el usuario y en el otro usuario\_id será la persona a la que se le envió la solicitud; en la primera entrada, status será amistadSolicitada, en la segunda será amistadPendiente
* **Aceptar amistad:**
  + *Front-end:* El usuario presionará un botón para aceptar la amistad. JS mandará el ID del usuario que aceptó la solicitud y el que la mandó.
  + *Back-end:* Se buscará en la base de datos las entradas que se generaron al iniciar la amistad y se cambiará su estado a amistadActiva
* **Eliminar amistad:**
  + *Front-end:* El usuario presionará un botón para eliminar la amistad. JS mandará el ID del usuario y el ID del amigo
  + *Back-end:* Se buscará en la base de datos las entradas que se generaron al iniciar la amistad y se cambiará su estado a amistadEliminada