Archivo de documentación compartido (para accesibilidad más cómoda durante las decisiones de proyecto): <https://docs.google.com/document/d/1tBdhLmyCRIP0TY55EGo1GK1JJkTR8xhFnlG4fBBtYxk/edit?tab=t.0>

**Especificación de la API – APIModelAccess**

**Descripción**

La clase APIModelAccess proporciona una interfaz de programación para gestionar la autenticación y administración de usuarios (supervisores y recepcionistas).

**Convención de nombres**

* Todos los métodos con sufijo **API** forman parte de la **API pública**: pueden ser llamados desde otros módulos del sistema.
* Los métodos **sin sufijo API** son de **uso interno** y no deben ser invocados directamente desde fuera de la clase.

Todos los métodos devuelven un objeto JSON con al menos las siguientes claves:

{

"status": true | false,

"result": string | object | null,

"type": string | null (si aplica)

}

**Métodos Públicos (API)**

**1. authenticateUserAPI(username, password)**

**Descripción**: Autentica un usuario en el sistema.  
**Parámetros**:

* username *(string)* -> nombre de usuario.
* password *(string)* -> contraseña.

**Retorno**:

* status: true si la autenticación fue exitosa, false en caso contrario.
* result:
  + "USER\_AUTHENTICATED" si la autenticación fue exitosa.
  + "USER\_NOT\_FOUND", "USER\_PASSWORD\_FAILED", "BLOCKED\_USER" en caso de error.
* type: "SUPERVISOR" o "RECEPTIONIST" si la autenticación fue exitosa.

**2. createUserAPI(currentUsername, username, password, type)**

**Descripción**: Crea un nuevo usuario.  
**Parámetros**:

* currentUsername *(string)* -> nombre de usuario que ejecuta la acción, debe ser supervisor.
* username *(string)* -> nombre del nuevo usuario.
* password *(string)* -> contraseña validada con validatePass.
* type *(string)* -> "SUPERVISOR" o "RECEPTIONIST".

**Retorno**:

* "USER\_CREATED" si se creó.
* "INVALID\_PASSWORD" si la contraseña no es válida según los requerimientos, "USER\_ALREADY\_EXISTS" si el nombre usuario ya existe en el sistema, "USER\_NOT\_AUTHORIZED" si el usuario que intenta ejecutar la acción no está autorizado.

**3. deleteUserAPI(currentUsername, username)**

**Descripción**: Elimina un usuario existente.  
**Parámetros**:

* currentUsername *(string)* -> nombre de usuario que ejecuta la acción, debe ser supervisor.
* username *(string)* -> usuario a eliminar.

**Retorno**:

* "USER\_DELETED" si éxito.
* "USER\_NOT\_FOUND" si no se encontró al usuario, "USER\_NOT\_AUTHORIZED" si el usuario que intenta ejecutar la acción no está autorizado.

**4. modifyUserAPI(currentUsername, username, newPassword)**

**Descripción**: Modifica la contraseña de un usuario existente.  
**Parámetros**:

* currentUsername *(string)* -> nombre de usuario que ejecuta la acción, debe ser supervisor.
* username *(string)* -> usuario a modificar.
* newPassword *(string)* -> nueva contraseña (debe ser válida).

**Retorno**:

* "USER\_UPDATED" si éxito.
* "INVALID\_PASSWORD" si la contraseña no es válida según los requerimientos, "USER\_NOT\_FOUND" si no se encontró al usuario, "USER\_NOT\_AUTHORIZED" si el usuario que intenta ejecutar la acción no está autorizado.

**5. searchUserAPI(currentUsername, username)**

**Descripción**: Busca un usuario en el sistema.  
**Parámetros**:

* currentUsername *(string)* -> nombre de usuario que ejecuta la acción, debe ser supervisor.
* username *(string)* -> usuario a buscar.

**Retorno**:

* result: objeto usuario si se encontró.
* "USER\_NOT\_FOUND" si no se encontró al usuario o "USER\_NOT\_AUTHORIZED" " si el usuario que intenta ejecutar la acción no está autorizado.

**6. listUserAPI(currentUsername)**

**Descripción**: Devuelve la lista de todos los supervisores y recepcionistas.  
**Parámetros**:

* currentUsername *(string)* -> nombre de usuario que ejecuta la acción, debe ser supervisor.

**Retorno**:

Si el usuario está autorizado:

"status": true,

"result":

"supervisors": [ ... ],

"receptionists": [ ... ]

En caso de que no esté autorizado:

"status": false,

"USER\_NOT\_AUTHORIZED" si el usuario que intenta ejecutar la acción no está autorizado.

**7. enableBlockedUserAPI(currentUsername, username, type)**

**Descripción**: Habilita nuevamente un usuario bloqueado (restablece su contador de intentos fallidos).  
**Parámetros**:

* currentUsername *(string)* -> nombre de usuario que ejecuta la acción, debe ser supervisor.
* username *(string)* -> usuario bloqueado.
* type *(string)* -> "SUPERVISOR" o "RECEPTIONIST".

**Retorno**:

* "USER\_ENABLED" si se desbloqueó correctamente.
* "USER\_NOT\_AUTHORIZED" si no tiene permisos.

**Métodos Internos (uso interno)**

**isPasswordCorrect(user, input)**

Verifica si la contraseña ingresada coincide con la almacenada.

**incrementFailedLogin(user)**

Incrementa el contador de intentos fallidos para el usuario.

**resetFailedloginCounter(user)**

Reinicia el contador de intentos fallidos.

**isAuthorizedUser(currentUsername)**

Verifica si el usuario actual es **Supervisor** (permiso de administrador).

**validatePass(pass)**

Valida la seguridad de la contraseña.

* Longitud: 8–16 caracteres.
* Al menos 1 mayúscula, 1 número y 2 caracteres especiales.

**Notas**

* Solo los usuarios de tipo **Supervisor** pueden realizar acciones administrativas (crear, modificar, eliminar, listar).
* Los recepcionistas solo pueden autenticarse.
* Los intentos fallidos de login bloquean al usuario después de 3 intentos.