Documentación de Controladores - Proyecto Huevícola

# 1. FirebaseUserController

Este controlador maneja las operaciones relacionadas con los usuarios autenticados por Firebase en el sistema. Se accede a través de la ruta base `api/firebase-users`.

## 1.1 Métodos

### @GetMapping - listarUsuarios()

Este método responde a peticiones GET a la ruta `api/firebase-users`. Retorna una lista de todos los usuarios registrados en Firebase. Utiliza el servicio `FirebaseUserServices` para obtener los datos y los envuelve en un `ResponseEntity` con código HTTP 200.

### @DeleteMapping("/{uid}") - eliminarUsuario(String uid)

Este método responde a peticiones DELETE en la ruta `api/firebase-users/{uid}`. Recibe el UID del usuario a eliminar como parámetro de la URL. Llama al método `delete(uid)` del servicio `FirebaseUserServices`. Si el usuario es eliminado exitosamente, retorna HTTP 204 No Content. Si el usuario no existe, retorna HTTP 404 Not Found.

# 2. UserController

Este controlador maneja operaciones relacionadas con la autenticación de usuarios mediante el sistema propio o con Google OAuth2. Su ruta base es `api/users`.

## 2.1 Métodos

### @PostMapping("/login") - loginUser(User user)

Este método maneja peticiones POST en la ruta `api/users/login`. Recibe un objeto `User` con los datos de inicio de sesión en el cuerpo de la petición. Llama al servicio `UserServices` para validar las credenciales y retorna un objeto `ResponseMessage` con el resultado del proceso de login.

### @GetMapping("/google") - redirectToGoogle()

Este método responde a peticiones GET en la ruta `api/users/google`. Redirige al usuario al endpoint de autenticación OAuth2 de Google. Utiliza `HttpServletResponse` para enviar la redirección al cliente hacia `/oauth2/authorization/google`.

# 3. Servicios Utilizados

## FirebaseUserServices

Contiene la lógica para listar y eliminar usuarios autenticados vía Firebase.

## UserServices

Contiene la lógica para autenticar usuarios con las credenciales proporcionadas en el login.

# 4. Observaciones Generales

- Ambos controladores siguen el patrón REST usando anotaciones de Spring Boot.  
- El uso de `ResponseEntity` permite retornar respuestas HTTP personalizadas.  
- La integración con Google mediante redirección facilita la autenticación externa sin almacenar credenciales directamente.