

Package com.mapmyjourney.backend.controller

Class UserController

java.lang.Object[↗]
com.mapmyjourney.backend.controller.UserController

```
@RestController
@RequestMapping("/users")
public class UserController
extends Object↗
```

Controlador REST para gestionar usuarios.

Constructor Summary

| Constructors |
|-------------------------------|
| Constructor |
| Description |
| <code>UserController()</code> |

Method Summary

| All Methods | Instance Methods | Concrete Methods |
|---|------------------|------------------|
| Modifier and Type | | |
| Method | | |
| Description | | |
| org.springframework.http.ResponseEntity<Void [↗] > <code>deleteUser</code> (Long [↗] userId) 7. | | |
| org.springframework.http.ResponseEntity<UserDTO> <code>getUserByEmail</code> (String [↗] email) 4. | | |
| org.springframework.http.ResponseEntity<UserDTO> <code>getUserById</code> (Long [↗] userId) 3. | | |
| org.springframework.http.ResponseEntity<LoginResponseDTO> <code>login</code> (LoginRequestDTO request) 2. | | |
| org.springframework.http.ResponseEntity<LoginResponseDTO> <code>registerUser</code> (@Valid UserCreateRequestDTO request) 1. | | |
| org.springframework.http.ResponseEntity<UserDTO> <code>updateUser</code> (Long [↗] userId, @Valid UserCreateRequestDTO request) 5. | | |
| org.springframework.http.ResponseEntity<UserDTO> <code>updateUserProfile</code> (Long [↗] userId, UserUpdateRequestDTO request) 6. | | |
| Methods inherited from class java.lang.Object [↗] | | |

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Details

UserController

```
public UserController()
```

Method Details

registerUser

```
@PostMapping("/register")
public org.springframework.http.ResponseEntity<LoginResponseDTO> registerUser
(@Valid @RequestBody
 @Valid UserCreateRequestDTO request)
```

1. Registra un nuevo usuario. POST /api/users/register

login

[illegible]

2. Inicia sesión con email y contraseña. POST /api/users/login

getUserById

[illegible]

3. Obtiene un usuario por ID. GET /api/users/{userId}

getUserByEmail

[illegible]

4. Obtiene un usuario por email. GET /api/users/email/{email}

updateUser

```
@PutMapping("/{userId}")
@PreAuthorize("hasRole('USER')")
public org.springframework.http.ResponseEntity<UserDT0> updateUser(@PathVariable Long userId,
    @Valid @RequestBody
    @Valid UserCreateRequestDT0 request)
```

5. Actualiza un usuario existente. PUT /api/users/{userId}

updateUserProfile

[illegible]

6. Actualiza el perfil del usuario (versión flexible). PUT /api/users/{userId}/profile

deleteUser

```
@DeleteMapping("/{userId}")
@PreAuthorize("hasRole(\"ADMIN\")")
public org.springframework.http.ResponseEntity<Void> deleteUser(@PathVariable
                                                                Long userId)
```

7. Elimina un usuario. DELETE /api/users/{userId}