

**Package** com.mapmyjourney.backend.controller

## Class UserController

java.lang.Object<sup>↗</sup>  
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 <sup>↗</sup> >		
<code>deleteUser(Long<sup>↗</sup> userId)</code>		
6.		
org.springframework.http.ResponseEntity<UserDTO>		
<code>getUserByEmail(String<sup>↗</sup> email)</code>		
4.		
org.springframework.http.ResponseEntity<UserDTO>		
<code>getUserById(Long<sup>↗</sup> userId)</code>		
3.		
org.springframework.http.ResponseEntity<LoginResponseDTO>		
<code>login(LoginRequestDTO request)</code>		
2.		
org.springframework.http.ResponseEntity<LoginResponseDTO>		
<code>registerUser(@Valid UserCreateRequestDTO request)</code>		
1.		
org.springframework.http.ResponseEntity<UserDTO>		
<code>updateUser(Long<sup>↗</sup> userId, @Valid UserCreateRequestDTO request)</code>		
5.		
Methods inherited from class java.lang.Object <sup>↗</sup>		
clone <sup>↗</sup> , equals <sup>↗</sup> , finalize <sup>↗</sup> , getClass <sup>↗</sup> , hashCode <sup>↗</sup> , notify <sup>↗</sup> , notifyAll <sup>↗</sup> , toString <sup>↗</sup> , wait <sup>↗</sup> , wait <sup>↗</sup> , wait <sup>↗</sup>		

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

```
@PostMapping("/login")
public org.springframework.http.ResponseEntity<LoginResponseDTO> login(@RequestBody
                                                                    LoginRequestDTO request)

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

getUserById

```
@GetMapping("/{userId}")
@PreAuthorize("hasRole('\\USER\\')")
public org.springframework.http.ResponseEntity<UserDTO> getUserById(@PathVariable
                                                                    Long? userId)

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

getUserByEmail

```
@GetMapping("/email/{email}")
@PreAuthorize("hasRole('\\USER\\')")
public org.springframework.http.ResponseEntity<UserDTO> getUserByEmail(@PathVariable
                                                                    String? email)

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

updateUser

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

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

deleteUser

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

6. Elimina un usuario. DELETE /api/users/{userId}
```

