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
UserController()	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
org.springframework.http.ResponseEntity<Void>	deleteUser(Long userId)	7.
org.springframework.http.ResponseEntity<UserDTO>	getUserByEmail(String email)	4.
org.springframework.http.ResponseEntity<UserDTO>	getUserById(Long userId)	3.
org.springframework.http.ResponseEntity<LoginResponseDTO>	login(LoginRequestDTO request)	2.
org.springframework.http.ResponseEntity<LoginResponseDTO>	registerUser(@Valid UserCreateRequestDTO request)	1.
org.springframework.http.ResponseEntity<UserDTO>	updateUser(Long userId, @Valid UserCreateRequestDTO request)	5.
org.springframework.http.ResponseEntity<UserDTO>	updateUserProfile(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")
@CrossOrigin(origins={"https://mapmyjourney-
4w93.onrender.com","http://localhost:4200","http://localhost:3000"})
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}

updateUserProfile

```
@PutMapping("/{userId}/profile")
@PreAuthorize("hasRole(\'USER\')")
```

```
public org.springframework.http.ResponseEntity<UserDTO> updateUserProfile(@PathVariable
                                                                           Long userId,
                                                                           @RequestBody
                                                                           UserUpdateRequestDTO request)
```

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}