

**Package** com.mapmyjourney.backend.service

## Class UserService

java.lang.Object<sup>↗</sup>  
com.mapmyjourney.backend.service.UserService

@Service  
public class **UserService**  
extends Object<sup>↗</sup>

### Constructor Summary

Constructors
Constructor
Description
<a href="#">UserService()</a>

### Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
	<b>LoginResponseDTO</b>	
	<b>authenticate</b> (String <sup>↗</sup> email, String <sup>↗</sup> password)	
	6.	
	void	
	<b>deleteUser</b> (Long <sup>↗</sup> userId)	
	5.	
	<b>UserDTO</b>	
	<b>getUserByEmail</b> (String <sup>↗</sup> email)	
	3.	
	<b>UserDTO</b>	
	<b>getUserById</b> (Long <sup>↗</sup> userId)	
	2.	
	<b>UserDTO</b>	
	<b>registerUser</b> (UserCreateRequestDTO request)	
	1.	
	<b>UserDTO</b>	
	<b>updateUser</b> (Long <sup>↗</sup> userId, UserCreateRequestDTO request)	
	4.	

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

UserService
-------------

```
public UserService()
```

## Method Details

### registerUser

```
@Transactional
public UserDTO registerUser(UserCreateRequestDTO request)
```

1. Registra un nuevo usuario. Verifica que el email no esté ya registrado.

**Parameters:**

request - DTO con los datos del usuario

**Returns:**

DTO del usuario creado

**Throws:**

[DuplicateResourceException](#) - si el email ya existe

### getUserById

```
@Transactional(readOnly=true)
public UserDTO getUserById(Long? userId)
```

2. Obtiene un usuario por ID.

**Parameters:**

userId - ID del usuario

**Returns:**

DTO del usuario encontrado

**Throws:**

[ResourceNotFoundException](#) - si el usuario no existe

### getUserByEmail

```
@Transactional(readOnly=true)
public UserDTO getUserByEmail(String? email)
```

3. Obtiene un usuario por email.

**Parameters:**

email - Email del usuario

**Returns:**

DTO del usuario encontrado

**Throws:**

[ResourceNotFoundException](#) - si el usuario no existe

### updateUser

```
@Transactional
public UserDTO updateUser(Long? userId,
                           UserCreateRequestDTO request)
```

4. Actualiza un usuario existente. Verifica que el nuevo email no esté en uso por otro usuario.

**Parameters:**

userId - ID del usuario a actualizar

request - DTO con los nuevos datos

**Returns:**

DTO del usuario actualizado

**Throws:**

[ResourceNotFoundException](#) - si el usuario no existe

`DuplicateResourceException` - si el nuevo email ya existe

**deleteUser**

```
@Transactional
public void deleteUser(LongⒺ userId)
```

5. Elimina un usuario.

**Parameters:**

userId - ID del usuario a eliminar

**Throws:**

`ResourceNotFoundException` - si el usuario no existe

**authenticate**

```
@Transactional(readOnly=true)
public LoginResponseDTO authenticate(StringⒺ email,
                                     StringⒺ password)
```

6. Autentica un usuario con email y contraseña.

**Parameters:**

email - Email del usuario

password - Contraseña en texto plano

**Returns:**

LoginResponseDTO con token JWT

**Throws:**

`ResourceNotFoundException` - si el usuario no existe

`org.springframework.security.authentication.BadCredentialsException` - si la contraseña es incorrecta