

Package com.mapmyjourney.backend.service

## Class UserService

java.lang.Object<sup>2</sup>  
com.mapmyjourney.backend.service.UserService

@Service  
public class UserService  
extends Object<sup>2</sup>

### Constructor Summary

#### Constructors

##### Constructor

##### Description

UserService()

### Method Summary

#### All Methods

#### Instance Methods

#### Concrete Methods

##### Modifier and Type

##### Method

##### Description

##### LoginResponseDTO

authenticate(String<sup>2</sup> email, String<sup>2</sup> password)

6.

##### void

deleteUser(Long<sup>2</sup> userId)

5.

##### UserDTO

getUserByEmail(String<sup>2</sup> email)

3.

##### UserDTO

getUserById(Long<sup>2</sup> userId)

2.

##### UserDTO

registerUser(UserCreateRequestDTO request)

1.

##### UserDTO

updateUser(Long<sup>2</sup> userId, UserCreateRequestDTO request)

4.

##### Methods inherited from class java.lang.Object<sup>2</sup>

clone<sup>2</sup>, equals<sup>2</sup>, finalize<sup>2</sup>, getClass<sup>2</sup>, hashCode<sup>2</sup>, notify<sup>2</sup>, notifyAll<sup>2</sup>, toString<sup>2</sup>, wait<sup>2</sup>, wait<sup>2</sup>, wait<sup>2</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

## deleteUser

```
@Transactional  
public void deleteUser(Long2 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(String2 email,  
                                     String2 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