

Package com.mapmyjourney.backend.service

Class UserService

java.lang.Object²
com.mapmyjourney.backend.service.UserService

```
@Service  
public class UserService  
extends Object2
```

Constructor Summary

Constructors

Constructor

Description

UserService()

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type

Method

Description

LoginResponseDTO

authenticate(String² email, String² password)

6.

void

deleteUser(Long² userId)

5.

UserDTO

getUserByEmail(String² email)

3.

UserDTO

getUserById(Long² userId)

2.

Long²

getUserByIdByEmail(String² email)

Obtiene el ID del usuario a partir del email.

UserDTO

registerUser(UserCreateRequestDTO request)

1.

LoginResponseDTO

registerUserAndAuthenticate(UserCreateRequestDTO request)

Registra un nuevo usuario y lo autentica automáticamente.

UserDTO

updateUser(Long² userId, UserCreateRequestDTO request)

4.

Methods inherited from class java.lang.Object

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

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

registerUserAndAuthenticate

```
@Transactional  
public LoginResponseDTO registerUserAndAuthenticate(UserCreateRequestDTO request)
```

Registra un nuevo usuario y lo autentica automáticamente. Retorna un token JWT junto con los datos del usuario.

Parameters:

request - DTO con los datos del usuario

Returns:

LoginResponseDTO con token JWT y datos del usuario

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

getUserIdByEmail

```
@Transactional(readOnly=true)
public Long get.getUserIdByEmail(String email)
```

Obtiene el ID del usuario a partir del email.

Parameters:

email - Email del usuario

Returns:

ID del usuario

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 y datos del usuario

Throws:

ResourceNotFoundException - si el usuario no existe

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