

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

## Class TripMemberService

java.lang.Object  
com.mapmyjourney.backend.service.TripMemberService

@Service  
public class TripMemberService  
extends Object

Servicio para gestionar la membresía de usuarios en viajes. Maneja la adición, eliminación y cambio de roles de miembros.

### Constructor Summary

#### Constructors

##### Constructor

##### Description

TripMemberService()

### Method Summary

#### All Methods

#### Instance Methods

#### Concrete Methods

##### Modifier and Type

##### Method

##### Description

##### TripMemberDTO

**addMemberToTrip(Long tripId, Long userId, TripMemberRole role)**

Agrega un nuevo miembro a un viaje.

##### TripMemberDTO

**changeMemberRole(Long tripId, Long userId, TripMemberRole newRole)**

Cambia el rol de un miembro en un viaje.

##### TripMemberDTO

**getMember(Long tripId, Long userId)**

Obtiene un miembro específico de un viaje.

##### List<TripMemberDTO>

**getTripMembers(Long tripId)**

Obtiene todos los miembros de un viaje.

##### boolean

**hasRole(Long tripId, Long userId, TripMemberRole role)**

Verifica si un usuario tiene un rol específico en un viaje.

##### boolean

**isMemberOfTrip(Long tripId, Long userId)**

Verifica si un usuario es miembro de un viaje.

##### void

**removeMemberFromTrip(Long tripId, Long userId)**

Elimina un miembro de un viaje.

### Methods inherited from class java.lang.Object

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

## Constructor Details

### TripMemberService

```
public TripMemberService()
```

## Method Details

### addMemberToTrip

```
@Transactional  
public TripMemberDTO addMemberToTrip(Long tripId,  
                                     Long userId,  
                                     TripMemberRole role)
```

Agrega un nuevo miembro a un viaje.

**Parameters:**

tripId - ID del viaje

userId - ID del usuario a agregar

role - Rol inicial del usuario (por defecto VIEWER)

**Returns:**

DTO del miembro creado

### getMember

```
@Transactional(readOnly=true)  
public TripMemberDTO getMember(Long tripId,  
                               Long userId)
```

Obtiene un miembro específico de un viaje.

**Parameters:**

tripId - ID del viaje

userId - ID del usuario

**Returns:**

DTO del miembro

### getTripMembers

```
@Transactional(readOnly=true)  
public List<TripMemberDTO> getTripMembers(Long tripId)
```

Obtiene todos los miembros de un viaje.

**Parameters:**

tripId - ID del viaje

**Returns:**

Lista de DTOs de miembros

### changeMemberRole

```
@Transactional  
public TripMemberDTO changeMemberRole(Long tripId,  
                                       Long userId,  
                                       TripMemberRole newRole)
```

Cambia el rol de un miembro en un viaje.

**Parameters:**

`tripId` - ID del viaje

`userId` - ID del usuario

`newRole` - Nuevo rol

**Returns:**

DTO del miembro actualizado

## removeMemberFromTrip

`@Transactional`

```
public void removeMemberFromTrip(Long2 tripId,  
                                 Long2 userId)
```

Elimina un miembro de un viaje.

**Parameters:**

`tripId` - ID del viaje

`userId` - ID del usuario a eliminar

## isMemberOfTrip

`@Transactional(readOnly=true)`

```
public boolean isMemberOfTrip(Long2 tripId,  
                           Long2 userId)
```

Verifica si un usuario es miembro de un viaje.

**Parameters:**

`tripId` - ID del viaje

`userId` - ID del usuario

**Returns:**

true si es miembro, false en caso contrario

## hasRole

`@Transactional(readOnly=true)`

```
public boolean hasRole(Long2 tripId,  
                      Long2 userId,  
                      TripMemberRole role)
```

Verifica si un usuario tiene un rol específico en un viaje.

**Parameters:**

`tripId` - ID del viaje

`userId` - ID del usuario

`role` - Rol a verificar

**Returns:**

true si el usuario tiene el rol, false en caso contrario