Package com.mapmyjourney.backend.service

Class ExpenseService

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

```
@Service
public class ExpenseService
extends Object
```

Constructor Summary

Constructors

Constructor Description
ExpenseService()

Method Summary

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

Modifier and Type Method Description
ExpenseDTO createExpense(Long tripId, ExpenseCreateRequestDTO request, Long creatorUserId) 1.
void deleteExpense(Long expenseId, Long userId) Elimina un gasto.
ExpenseDTO getExpenseById(Long expenseId) 3.
List<ExpenseDTO> getTripExpenses(Long tripId) 2.
void markSplitAsPaid(Long splitId) 4.
ExpenseDTO updateExpense(Long expenseId, ExpenseCreateRequestDTO request, Long userId) Actualiza un gasto existente.

Methods inherited from class java.lang.Object

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

Constructor Details

ExpenseService

```
public ExpenseService()
```

Method Details

createExpense

```
@Transactional
public ExpenseDTO createExpense(Long tripId,
                                ExpenseCreateRequestDTO request,
                                Long creatorUserId)
```

1. Crea un nuevo gasto y divide entre participantes. Valida que el viaje y los usuarios existan.

Parameters:

`tripId` - ID del viaje donde se registra el gasto

`request` - DTO con los datos del gasto

`creatorUserId` - ID del usuario que crea el gasto (quien paga)

Returns:

DTO del gasto creado

Throws:

`ResourceNotFoundException` - si el viaje o usuario no existe

`ValidationException` - si el monto es inválido o no hay participantes

getTripExpenses

```
@Transactional(readOnly=true)
public List<ExpenseDTO> getTripExpenses(Long tripId)
```

2. Obtiene todos los gastos de un viaje.

Parameters:

`tripId` - ID del viaje

Returns:

Lista de DTOs de gastos del viaje

getExpenseById

```
@Transactional(readOnly=true)
public ExpenseDTO getExpenseById(Long expenseId)
```

3. Obtiene un gasto por ID.

Parameters:

`expenseId` - ID del gasto

Returns:

DTO del gasto encontrado

Throws:

`ResourceNotFoundException` - si el gasto no existe

markSplitAsPaid

```
@Transactional
public void markSplitAsPaid(Long splitId)
```

4. Marca una división como pagada.

Parameters:

`splitId` - ID de la división a marcar como pagada

updateExpense

```
@Transactional
public ExpenseDTO updateExpense(Long expenseId,
                                ExpenseCreateRequestDTO request,
                                Long userId)
```

Actualiza un gasto existente. Solo quien pagó el gasto puede actualizarlo.

Parameters:

`expenseId` - ID del gasto a actualizar

`request` - DTO con los nuevos datos

`userId` - ID del usuario que hace la actualización

Returns:

DTO del gasto actualizado

Throws:

`ResourceNotFoundException` - si el gasto no existe

`ValidationException` - si el usuario no es quien pagó o los datos son inválidos

deleteExpense

```
@Transactional
public void deleteExpense(Long expenseId,
                          Long userId)
```

Elimina un gasto. Solo quien pagó el gasto puede eliminarlo.

Parameters:

`expenseId` - ID del gasto a eliminar

`userId` - ID del usuario que hace la eliminación

Throws:

`ResourceNotFoundException` - si el gasto no existe

`ValidationException` - si el usuario no es quien pagó

Copyright © 2026. All rights reserved.