

**Package** com.mapmyjourney.backend.controller

## Class ExpenseController

java.lang.Object<sup>↗</sup>  
com.mapmyjourney.backend.controller.ExpenseController

```
@RestController
@RequestMapping("/trips/{tripId}/expenses")
public class ExpenseController
extends Object↗
```

Controlador REST para gestionar gastos en viajes.

### Constructor Summary

Constructors
Constructor
Description
ExpenseController()

### Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type		
Method		
Description		
org.springframework.http.ResponseEntity<ExpenseDT0>		
createExpense(Long <sup>↗</sup> tripId, @Valid ExpenseCreateRequestDT0 request)		
1.		
org.springframework.http.ResponseEntity<Void <sup>↗</sup> >		
deleteExpense(Long <sup>↗</sup> tripId, Long <sup>↗</sup> expenseId)		
5.		
org.springframework.http.ResponseEntity<ExpenseDT0>		
getExpense(Long <sup>↗</sup> tripId, Long <sup>↗</sup> expenseId)		
3.		
org.springframework.http.ResponseEntity<List <sup>↗</sup> <ExpenseDT0>>		
getTripExpenses(Long <sup>↗</sup> tripId, int page, int size)		
2.		
org.springframework.http.ResponseEntity<ExpenseDT0>		
updateExpense(Long <sup>↗</sup> tripId, Long <sup>↗</sup> expenseId, @Valid ExpenseCreateRequestDT0 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

ExpenseController
public ExpenseController()

## Method Details

### createExpense

```
@PostMapping
@PreAuthorize("hasRole('\\USER\\')")
public org.springframework.http.ResponseEntity<ExpenseDTO> createExpense
(@PathVariable
 Long tripId,
 @Valid @RequestBody
 @Valid ExpenseCreateRequestDTO request)
```

1. Crea un nuevo gasto en el viaje. POST /api/trips/{tripId}/expenses

### getTripExpenses

```
@GetMapping
@PreAuthorize("hasRole('\\USER\\')")
public org.springframework.http.ResponseEntity<List<ExpenseDTO>> getTripExpenses
(@PathVariable
 Long tripId,
 @RequestParam(defaultValue="0")
 int page,
 @RequestParam(defaultValue="20")
 int size)
```

2. Obtiene todos los gastos del viaje (con paginación). GET /api/trips/{tripId}/expenses? page=0&size=20&sort=expenseDate,desc

### getExpense

```
@GetMapping("/{expenseId}")
@PreAuthorize("hasRole('\\USER\\')")
public org.springframework.http.ResponseEntity<ExpenseDTO> getExpense(@PathVariable
 Long tripId,
 @PathVariable
 Long expenseId)
```

3. Obtiene un gasto por ID. GET /api/trips/{tripId}/expenses/{expenseId}

### updateExpense

```
@PutMapping("/{expenseId}")
@PreAuthorize("hasRole('\\USER\\')")
public org.springframework.http.ResponseEntity<ExpenseDTO> updateExpense
(@PathVariable
 Long tripId,
 @PathVariable
 Long expenseId,
 @Valid @RequestBody
 @Valid ExpenseCreateRequestDTO request)
```

4. Actualiza un gasto. PUT /api/trips/{tripId}/expenses/{expenseId} Solo quien lo pagó puede actualizarlo.

### deleteExpense

```
@DeleteMapping("/{expenseId}")
@PreAuthorize("hasRole('\\USER\\')")
public org.springframework.http.ResponseEntity<Void> deleteExpense(@PathVariable
 Long tripId,
 @PathVariable
 Long expenseId)
```

5. Elimina un gasto. DELETE /api/trips/{tripId}/expenses/{expenseId} Solo quien lo pagó puede eliminarlo.

