Expenses API Documentation

Written by Mist

Expenses and the Expense object

This API allows you to fetch, create and delete expenses. Each expense uses the following object structure:

{
amount: number
category: string
date: string (must be convertible into a date using the "Date()" function)
description: string
__v: number
_**id**: string
}

Fetching expenses

To fetch all the expenses in the database, send a GET request to:

https://expenses-api-production-7770.up.railway.app/api/v1/expense

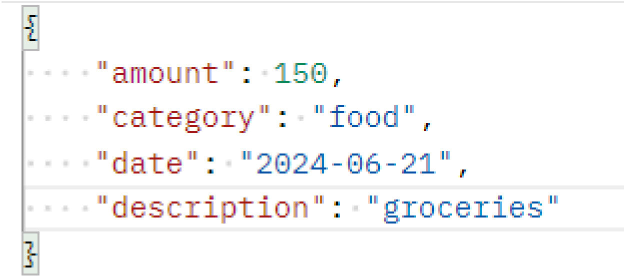
This will return either an array of expense objects (with a 200 status code), or an error object if something went wrong. See the "Error Object" section for more details on errors.

Creating expenses

To create a new expense, send a POST request to:

https://expenses-api-production-7770.up.railway.app/api/v1/expense

The body of the request must contain the amount, the category, the date, and the description, for example:



```
{
    "amount": 150,
    "category": "food",
    "date": "2024-06-21",
    "description": "groceries"
}
```

This will return either a copy of the created expense object (with a 201 status code), or an error object if something went wrong. See the "Error Object" section for more details on errors.

Deleting expenses

To delete an existing expense, send a DELETE request to:

https://expenses-api-production-7770.up.railway.app/api/v1/expense/

With the _id property of the expense you wish to delete added to the end of the url. For example if the _id property is “5541”, you would send a request to

https://expenses-api-production-7770.up.railway.app/api/v1/expense/5541

This will return either a copy of the deleted expense object (with a 200 status code), or an error object if something went wrong. See the “Error Object” section for more details on errors.

Error Object

If there is an error with any request, the API will send back an error object. The error object looks like this:

```
{
   code: number
   message: string
   params: strings array
}
```

“code” is the HTTP status code for the error (e.g. 400, 404, 500, etc.)

“message” contains details about the error

“params” is an array containing all the fields that caused the error