06

Compondo funções

Transcrição

Vamos alterar o módulo `app/nota/service.js` e realizar a composição das funções:

```
// app/app.js

import { handleStatus } from '../utils/promise-helpers.js';
import { partialize } from '../utils/operators.js';

const API = `http://localhost:3000/notas`;

const getItemsFromNotas = notas => notas.$flatMap(nota => nota.itens);
const filterItemsByCode = (code, items) => items.filter(item => item.codigo === code);
const sumItemsValue = items => items.reduce((total, item) => total + item.valor, 0);

export const notasService = {

    listAll() {
        return fetch(API)
        .then(handleStatus)
        .catch(err => {
            console.log(err);
            return Promise.reject('Não foi possível obter as notas fiscais');
        });
    },

    sumItems(code) {

        // utilizando partialize
        const filterItems = partialize(filterItemsByCode, code);

        // realizando a composição
        return this.listAll().then(notas =>
            sumItemsValue(
                filterItems(
                    getItemsFromNotas(notas)
                )
            )
        );
    }
};
```

Melhoramos nosso código separando a responsabilidade da função `sumItem` em mais de uma função, mas realizar manualmente a composição deixa um tanto a desejar. Será que podemos criar uma função utilitária no estilo de `partialize`, mas que nos ajude na composição de funções? Sim, podemos e é exatamente isso que veremos na próxima seção.