**Proposta de Projeto**

|  |
| --- |
| **Uma API para controle de transações bancárias (back-end)** |
| **Proposta**  Você deverá montar uma simples API onde um usuário poderá acessá-la para enviar, editar, remover, e obter informações sobre um conjunto de dados sobre transações bancárias. Sua API deverá seguir o padrão RESTful até o nível 2 do modelo de maturidade de Richardson.  Você poderá montar a sua API utilizando-se de uma das seguintes linguagens de programação: Java, C#, PHP, Python, Ruby ou JavaScript. Você poderá utilizar um framework, contando que os requisitos listados na próxima seção sejam todos atendidos sem comprometimento. |
| **Requisitos**  A sua API deverá seguir os seguinte requisitos listados abaixo:   1. Sua API deverá ter um endpoint onde o usuário consiga fazer a operação de adição de novos registros.    1. Um registro é uma transação monetária, e deverá ter como campos no mínimo: um identificador, a data e hora da transação, o modo como foi feita, uma categoria, uma nota de observação, um valor, e o tipo da transação.       1. O identificador deverá ser gerado quando a API aceitar a transação como válida e armazenar a mesma.       2. O campo “data e hora de transação” deverá aceitar um valor no formato ISO 8601       3. O campo “modo de transação” poderá ser somente os valores “Dinheiro”, “Cartão de Débito”, “Cartão de Crédito”, “Transferência Bancária”.       4. Os campos “categoria” e “nota de observação” são campos de textos livres.       5. O tipo da transação poderá ser somente os valores “Receita” ou “Despesa”. 2. Sua API deverá ter um endpoint onde o usuário poderá fornecer uma lista de identificadores que identificam um conjunto de transações a serem removidas:    1. A remoção de uma transação acontece através da criação de registros idênticos, com a diferença de que o tipo da transação deverá ser oposta (correção através de estorno); 3. Sua API deverá ter um endpoint onde o usuário poderá fornecer um conjunto de registros contendo seu identificador junto com valores a serem editados:    1. A edição de uma transação acontece primeiro fazendo a remoção do mesmo (pela operação de estorno) e então a criação do novo registro (lançamento retificado);    2. Deve ser possível identificar quando um registro foi editado, e o registro mais novo deverá armazenar os identificadores dos registros mais antigos; 4. Sua API deverá ter endpoints onde o usuário consiga realizar as seguintes operações de busca, listagem e filtragem, exibindo todos os campos da transação que atender a respectiva operação:    1. Obter todos os registros de um ano;    2. Obter todos os registros de um mês e ano;    3. Obter todos os registros de um dia, mês e ano.    4. Obter todos os registros com uma categoria de transação;    5. Obter todos os registros pelo modo da transação;    6. Filtrar apenas registros de entrada, gasto ou ambos;    7. Filtrar apenas registros maiores que ou menores que um certo valor;    8. Saber se o registro foi removido e apresentar o registro complementar (a transação que faz o estorno do primeiro valor).    9. Saber se o registro foi editado, e apresentar os valores dos registros antigos. 5. Sua API deverá retornar uma mensagem com o apropriado “status code” quando receber uma requisição:    1. Uma mensagem e status code de erro quando ela não conseguir atender a requisição solicitada (exemplos como, dados enviados no formato errado, acesso a endpoints inexistente ou sem permissão, etc);    2. Uma mensagem e status code de sucesso quando ela conseguir atender a solicitação. Caso for uma operação de criação, ela deverá também retornar o valor de identificador dos registros que foram criados. 6. Para usar os endpoints da sua API o usuário deverá estar autenticado através do uso de uma chave de API que deverá ser passada como valor para um header de nome “X-API-Key” da requisição; 7. Sua API deverá aceitar receber valor de corpo das requisições POST somente no formato JSON; 8. Todas as respostas da sua API deverão retornar os dados no formato JSON com o header “Content-Type” com o valor  “application/json”. |
| **Entregas**   1. Documento de especificação da sua API:    1. Lista dos endpoints da sua API, contendo o nome do endpoint, o método HTTP, descrição do formato de requisição que aceita e das respostas possíveis que o mesmo pode gerar; 2. Código fonte da aplicação da API em uma das linguagens de programação definidas:    1. Instrução de como configurar o ambiente de produção da sua aplicação (variáveis de ambiente, banco de dados, diretórios e arquivos, etc);    2. Instruções de como compilar/executar o código-fonte da aplicação;    3. Pelo menos 20 registros já armazenados no sistema, disponíveis para consulta através de endpoints apropriados. |
| **Material**  [O que é API? REST e RESTful? | Mayk Brito](https://www.youtube.com/watch?v=ghTrp1x_1As)  [O QUE É UMA API RESTful NA PRÁTICA? MATURIDADE DE RICHARDSON](https://www.youtube.com/watch?v=P92SBaN42mQ)  [RESTful APIs in 100 Seconds // Build an API from Scratch with Node.js Express](https://www.youtube.com/watch?v=-MTSQjw5DrM)  [What is REST - REST API Tutorial](https://restfulapi.net/)  [Richardson Maturity Model](https://martinfowler.com/articles/richardsonMaturityModel.html) |