Documentação do projeto

Money Manager API

VERSAO 1.0

Outubro 02, 2023

Equipe:

Davi Siqueira de Carvalho Torres - 01633448

João Antônio de Lima Carrazzoni- 01523892

Luann Henrique de Sousa Lucas - 01356035

Roberto Henrique Cavalcanti Freitas - 01536220

Descrição:

A Money Manager API é uma aplicação de gerenciamento financeiro que permite aos usuários registrar e controlar suas atividades financeiras, como despesas e ganhos.

1. Objetivo e escopo do projeto:

O objetivo principal deste projeto é desenvolver uma API financeira que permita aos usuários controlar suas finanças pessoais de maneira eficaz e conveniente. A API visa oferecer funcionalidades essenciais para o gerenciamento de despesas e receitas, proporcionando aos usuários uma ferramenta para o controle de suas atividades financeiras.

O escopo inicial inclui as seguintes funcionalidades-chave:

1. **Inserção de Atividades Financeiras:** Os usuários podem inserir informações sobre suas atividades financeiras, incluindo a data, descrição, tipo (receita ou despesa) e valor.

2. **Listagem de Atividades Financeiras:** A API permite que os usuários visualizem uma lista de todas as atividades financeiras registradas.

3. **Cálculo de Saldo:** A API é capaz de calcular o saldo atual com base nas atividades financeiras registradas, proporcionando aos usuários uma visão clara de sua situação financeira.

4. **Exclusão de Atividades:** Os usuários têm a capacidade de excluir atividades financeiras específicas que não são mais relevantes.

5. **Autenticação e Autorização:** A API implementa autenticação e autorização de usuários para proteger as informações financeiras dos usuários e garantir que apenas usuários autorizados possam acessar e modificar os dados.

6. **Documentação Adequada:** O projeto inclui documentação detalhada da API, incluindo exemplos de uso, para facilitar a integração e o desenvolvimento de aplicativos que utilizam a API.

2. Público-Alvo:

API Money Manager destina-se a indivíduos que desejam controlar suas finanças pessoais e a desenvolvedores que buscam integrar funcionalidades de gerenciamento financeiro em seus aplicativos ou sistemas.

3. Especificações:

3.1. Tecnologias Utilizadas:

Neste projeto, foram adotadas várias tecnologias essenciais para a criação e implantação da API financeira, cada uma desempenhando um papel fundamental na construção e operação da aplicação:

- **Java Persistence API (JPA):** O projeto faz uso da Java Persistence API (JPA) para mapear objetos Java para entidades de banco de dados e simplificar a interação com o banco de dados MySQL. Isso permite a persistência e recuperação eficiente dos dados financeiros dos usuários.
- **JSON Web Tokens (JWT):** A autenticação e autorização dos usuários na API são gerenciadas por JSON Web Tokens (JWT). Esse mecanismo seguro permite que os usuários acessem suas contas e protege as informações financeiras sensíveis.
- **Spring Boot:** O Spring Boot é o framework principal utilizado no desenvolvimento da API. Ele fornece um ambiente simplificado e eficaz para a criação de aplicativos web. Com o Spring Boot, é possível configurar rapidamente recursos como segurança, endpoints RESTful e muito mais.
- **Gradle:** O Gradle foi escolhido como a ferramenta de gerenciamento de pacotes da aplicação. Ele facilita a gestão de dependências e a construção do projeto, garantindo a organização e atualização eficiente das bibliotecas utilizadas.
- **Docker com MySQL:** O Docker Compose foi adotado para simplificar a implantação da API e o gerenciamento de suas dependências. Ele cria e configura containers Docker para a aplicação e o banco de dados MySQL, garantindo ambientes consistentes de desenvolvimento e produção.

Essas tecnologias trabalham em conjunto para fornecer uma base sólida e eficaz para a API, garantindo sua confiabilidade, segurança e desempenho.

3.2. Arquitetura:

- **Arquitetura:** A API Money Manager segue uma arquitetura RESTful para expor endpoints que podem ser acessados por clientes para realizar operações financeiras. Isso permite uma comunicação simples e eficaz com os clientes da API.

3.3. Rotas e Endpoints:

Obs: (JWT em desenvolvimento)

A API inclui os seguintes endpoints:

Protegido:

- `/activities` (GET): Listar todas as atividades financeiras.
- `/activities` (POST): Inserir uma nova atividade financeira (Protegido)
- `/activities/{id}` (DELETE): Excluir uma atividade financeira por ID ()
- `/activities/balance` (GET): Calcular o saldo disponível.

Validação e criação:

- `/auth/login` (POST): Autenticar-se e obter um token JWT para acesso seguro.

3.2. Autenticação e Autorização:

- A API irá utilizar autenticação via tokens JWT para garantir o acesso seguro aos endpoints. As políticas de autorização controlam quais operações cada usuário pode realizar.

3.3. Banco de dados:

- O banco de dados MySQL é usado para armazenar informações sobre as atividades financeiras dos usuários. A modelagem irá incluir tabelas para atividades, usuários e autenticação.

A estrutura do banco de dados associada ao aplicativo de gerenciamento financeiro **atualmente** se resume à tabela activities.

Tabela activities:

Esta tabela armazena informações sobre as atividades financeiras, como receitas e despesas.

Coluna	Tipo	Descrição
id	VARCHAR(36)	Chave primária, identificador único da atividade
description	VARCHAR(255)	Descrição da atividade
date	TIMESTAMP	Data e hora da atividade
value	FLOAT	Valor da atividade
type	INTEGER	Tipo da atividade (0 para despesa, 1 para receita)
created_at	TIMESTAMP	Data e hora de criação da atividade
updated_at	TIMESTAMP	Data e hora da última atualização da atividade

A tabela armazena informações sobre cada atividade financeira. Ela inclui campos como **id, description, date, value, type, created_at e updated_at.**

O campo type é representado como um valor inteiro, onde 0 indica uma despesa e 1 indica uma receita. Isso corresponde ao uso da enumeração **ActivityType** em Java.

3.4. Formato de Dados:

- A API aceita e retorna dados no formato **JSON**, tornando-a facilmente integrável com diferentes tipos de aplicativos e sistemas.

3.5. Campos e Validações:

A API Money Manager já inclui algumas validações nos campos de entrada. Abaixo estão as validações que já estão implementadas:

ID da Atividade:

- O campo id é validado para garantir que não esteja em branco e que tenha um comprimento de 36 caracteres, correspondendo a um UUID válido.

Descrição da Atividade:

- O campo description é validado para garantir que não esteja em branco e tenha pelo menos 3 caracteres.

Tipo de Atividade:

- O campo type é validado para garantir que seja uma das opções válidas: "revenue" ou "expense". Qualquer outro valor é considerado inválido.

Valor da Atividade:

- O campo value é validado para garantir que seja maior que 0.1.

Estas validações são implementadas no método **validate** da classe **Activity**, que é chamado no momento da criação de uma atividade. Elas garantem que os dados inseridos sejam coerentes e atendam aos requisitos esperados, evitando atividades inválidas no sistema.

Vale mencionar que as validações adicionais podem e irão ser implementadas conforme necessário para garantir a integridade dos dados da API.

3.5. Tratamento de Erros:

A API Money Manager implementa tratamento de erros em conformidade com as melhores práticas. Abaixo estão alguns exemplos de como o tratamento de erros é realizado:

- Classe **ExceptionResponseBody**:

  A classe **ExceptionResponseBody** é usada para representar respostas de erro padronizadas. Ela inclui os campos timestamp, status, error e path.

- **ControllerAdvice** para Tratamento de Exceções:

  O controller **ActivityControllerExceptionHandler** é anotado com @ControllerAdvice e lida com exceções específicas lançadas pelos controladores da API.

- Tratamento de Exceções Específicas:

  A classe **ActivityControllerExceptionHandler** inclui métodos anotados com @ExceptionHandler para lidar com exceções específicas, como **DomainException**, **PersistenceException**, **ServiceException**, e **Exception** (para tratamento genérico).

- Exemplo de Tratamento de Exceção:

  Quando uma exceção **DomainException** é lançada, o método **handleDomainExceptions** cria um objeto **ExceptionResponseBody** com

  informações sobre a exceção, incluindo a data e hora em que ocorreu, o status HTTP correspondente, a mensagem de erro e o caminho da solicitação.

- Resposta de Erro Padronizada:

  - A resposta de erro padronizada é retornada ao cliente com um status HTTP apropriado, fornecendo informações detalhadas sobre o erro.

Essa abordagem garante que os erros sejam tratados de forma adequada e consistente em toda a API, facilitando a compreensão e o tratamento de problemas por parte dos desenvolvedores e usuários.

3.6. Desempenho e Escalabilidade:

- **Banco de Dados Otimizado:** O uso do MySQL como banco de dados é uma escolha adequada para aplicações financeiras. No entanto, à medida que a quantidade de dados cresce, medidas adicionais, como índices apropriados e particionamento de tabelas, podem ser implementadas para otimizar ainda mais o desempenho das consultas.
- **Padrões de Consulta Eficientes:** O código da API emprega o Spring Data JPA, o que simplifica a criação de consultas. Para melhorar o desempenho, consultas mais complexas podem ser otimizadas e índices adicionais podem ser introduzidos, conforme necessário.

No geral, a API Money Manager está bem preparada para lidar com cargas de trabalho substanciais e pode ser dimensionada para atender às demandas crescentes. No entanto, um monitoramento cuidadoso e uma abordagem proativa à otimização de desempenho são essenciais para garantir uma experiência de usuário confiável à medida que a base de usuários e o volume de dados aumentam.

3.7. Documentação da API:

- Swagger (Em Desenvolvimento):
    - A API Money Manager planeja incorporar o Swagger para a documentação interativa. O Swagger permitirá que os desenvolvedores visualizem e interajam com os endpoints da API diretamente por meio de uma interface amigável, facilitando o entendimento e o teste dos recursos oferecidos.
- Javadoc no Projeto:
    - O código-fonte do projeto inclui comentários Javadoc para a documentação interna das classes, métodos e variáveis. Isso auxilia os desenvolvedores que

trabalham no projeto a entender a estrutura do código, os propósitos das classes e os detalhes das funções.

A combinação do Swagger para documentação interativa e os comentários Javadoc no código torna o projeto acessível e bem documentado, simplificando o desenvolvimento, o teste e a manutenção da API. Isso proporcionará aos desenvolvedores uma experiência mais eficiente ao trabalhar com a API Money Manager.

4. Implementações até o Momento

Classe de Modelagem:

- Foi implementada a classe **Activity** que representa uma atividade financeira com atributos como **id**, **date**, **description**, **value**, **type**, **createdAt**, e **updatedAt**. Esta classe serve como a representação dos dados das atividades financeiras.

Classe de Repositórios:

- A classe **ActivityJpaRepository** foi implementada, que estende **JpaRepository** e é responsável por fornecer métodos para interagir com a entidade **Activity** no banco de dados. Isso inclui operações como salvar, excluir e consultar atividades no banco de dados.

Classe de Controles (Controller):

- A classe **ActivityController** foi implementada, fornecendo endpoints para várias operações relacionadas às atividades financeiras. Alguns dos endpoints incluem listar atividades, inserir atividades, excluir atividades e calcular o saldo.

DTOs (Objetos de Transferência de Dados):

- Foram definidos diversos DTOs para transferência de dados entre a API e o cliente. Isso inclui DTOs para solicitação e resposta, como **InsertActivityRequestDto**, **InsertActivityResponseDto**, **ListActivitiesResponseDto**, etc.

Mapeamento de DTOs:

- Foram implementadas classes de mapeamento (DTO mappers) que são responsáveis por mapear objetos DTO para objetos de domínio e vice-versa, garantindo uma separação adequada entre as camadas de controle e serviço.

Serviços:

- A classe **ActivityServiceImplementation** foi implementada para fornecer a lógica de negócios relacionada às atividades financeiras. Isso inclui operações como **inserção**, **exclusão**, **listagem de atividades** e **cálculo de saldo**..

Conclusões da primeira versão:

No geral, o projeto está em andamento e já possui as bases necessárias para gerenciar atividades financeiras, incluindo persistência em banco de dados, endpoints de API e tratamento de dados com DTOs. A documentação do projeto e outros detalhes específicos, como autenticação e segurança, serão ser adicionados nas próximas etapas de desenvolvimento.