Documentação da API LiquidPay

Visão Geral

A API LiquidPay permite a interação com o sistema de pagamentos e gestão de clientes. Este documento descreve as principais operações disponíveis na API e como utilizá-las.

Autenticação

A API utiliza autenticação via token. Para acessar os endpoints protegidos, inclua o token no cabeçalho da requisição:

Authrization: Bearer {{token}}

Configuração das Variáveis Globais

As variáveis globais são utilizadas para armazenar valores que são usados em múltiplas requisições dentro da coleção Postman. Elas permitem centralizar a configuração de valores, facilitando atualizações e manutenções.

Exemplo de Variáveis Globais:

- **{{base_url}}**: URL base para todas as requisições. Usada para garantir que todas as requisições apontem para o mesmo endpoint sem necessidade de definir a URL em cada requisição individualmente.
- **{{token}}**: Armazena o token JWT que é usado para autenticar as requisições. Esse token é recuperado através de uma requisição específica e atualizado automaticamente.
- **{{login}}**: Nome de usuário ou e-mail utilizado para autenticação. Esta variável permite a personalização do processo de login sem precisar alterar as requisições individualmente.
- **{{senha}}**: Senha do usuário para autenticação. Assim como a variável {{login}}, {{senha}} centraliza a configuração da senha para simplificar a autenticação.
- **{{user_id}}**: Identificador único do usuário autenticado. Esta variável é utilizada para realizar operações que exigem a especificação de um usuário específico, como consulta de informações pessoais, histórico de transações, etc.

Configuração:

Para configurar as variáveis globais:

1. Vá até a aba de "Ambientes" no Postman.
2. Crie ou edite um ambiente e adicione as variáveis {{base_url}}, {{token}}, {{login}}, {{senha}} e {{user_id}}.
3. Defina os valores iniciais e atuais conforme necessário. Por exemplo:
    - {{login}}: "seu_usuario"
    - {{senha}}: "sua_senha"
    - {{user_id}}: "123456"

Função de Recuperação de Token

A função de recuperação de token é uma requisição que obtém um token JWT do servidor, usando as credenciais armazenadas nas variáveis globais `{{login}}` e `{{senha}}`. Este token é então armazenado na variável global `{{token}}`, que é utilizada em todas as requisições subsequentes para garantir que o usuário esteja autenticado.

Exemplo de Código para Recuperação de Token

Abaixo está um exemplo de como a recuperação de token pode ser implementada em JavaScript utilizando o Postman:

```
let host = pm.collectionVariables.get("host");

let login = pm.collectionVariables.get("login");

let senha = pm.collectionVariables.get("senha");


let body = JSON.stringify({
        "login": login,
        "senha": senha
});


const postRequest = {
  url: host + '/api/login',
  method: 'POST',
```

```
  timeout: 0,
  header: {
        "Content-Type": "application/json",
        "Accept": "*/*"
  },
  body: body
};

pm.sendRequest(postRequest, function (err, res) {
        if (err) {
        console.log("Error: ", err);
        } else if (res.code === 200) { // Verifica se a resposta é 200 OK
        console.log("Login Response: ", res);

        // Captura o token do corpo da resposta
        let jsonData = res.json();
        let token = jsonData.token; // Supondo que o token está no campo 'token'

        // Armazena o token, login e senha em variáveis de coleção
        pm.collectionVariables.set('token', token);
        console.log("Token, login e senha armazenados com sucesso.", token);
        } else {
        console.log("Falha no login. Status: ", res.code);
        }
});
```

Endpoints

Clientes

Cadastro de Cliente

URL: `{{host}}/api/clientes`

Método: `POST`

Headers: Nenhum

Body:

Campo	Obrigação	Tipo	Exemplo
nome	Sim	String	"Thiago RS"
cpf	Sim	String	"93525550097"
email	Sim	String	"nn@nn.com"
senha	Sim	String	"12_Thi55"
telefone	Não	String	
data_nascimento	Não	String	2024-07-31

Respostas:

201 Created:


```
{
  "nome": "Thiago RS",
  "cpf": "93525550097",
  "email": "nn@nn.com",
  "senha":
"$2y$10$N1nsm/AHTUmgRqErkaMt4.C3lnqF9vffCpofWYxUmZPTiCx5fvLY6",
  "updated_at": "2024-08-01T01:36:48.000000Z",
  "created_at": "2024-08-01T01:36:48.000000Z",
  "id": 1
}
```

400 Bad Request:

Campos obrigatórios ausentes:


```
{
  "errors": {
     "cpf": ["O campo cpf é obrigatório."],
     "email": ["O campo email é obrigatório."],
     "senha": ["O campo senha é obrigatório."]
  }
}
```


Email inválido:


```
{
  "errors": {
     "email": ["O email deve ser um endereço de e-mail válido."]
  }
}
```


Senha inválida:


```
{
  "errors": {
     "senha": [
     "O senha deve ter pelo menos 8 caracteres.",
     "A senha deve conter pelo menos uma letra maiúscula, conter pelo
menos uma letra minúscula, conter pelo menos um caractere especial."
     ]
  }
}
```

Listagem de Clientes

URL: `{{host}}/api/clientes`

Método: `GET`

Headers: `Authorization: Bearer {{token}}`

Respostas:

200 OK:


```
[
  {
     "id": 1,
     "nome": "Thiago RS",
     "cpf": "93525550097",
     "email": "nn@nn.com",
     "telefone": null,
     "data_nascimento": null,
     "data_cadastro": "2024-07-31T00:00:00.000000Z",
     "status": 1,
     "observacoes": null,
     "senha":
"$2y$10$N1nsm/AHTUmgRqErkaMt4.C3lnqF9vffCpofWYxUmZPTiCx5fvLY6",
     "created_at": "2024-08-01T01:36:48.000000Z",
     "updated_at": "2024-08-01T01:36:48.000000Z"
  }
]
```

Alteração de Cliente

URL: `{{host}}/api/clientes`

Método: `PUT`

Headers: `Authorization: Bearer {{token}}`

Body:

Campo	Obrigação	Tipo	Exemplo
nome	Sim	String	"Thiago RS"
cpf	Sim	String	"93525550097"
email	Sim	String	"nn@nn.com"
telefone	Não	String	
data_nascimento	Não	String	2024-07-31

Respostas:

200 OK:


```
{
"id": 1,
"nome": "Thiago RS",
"cpf": "93525550097",
"email": "nn@nn.com",
"telefone": "11 91234-5678",
"data_nascimento": "2024-07-31",
"created_at": "2024-08-01T01:36:48.000000Z",
"updated_at": "2024-08-02T01:36:48.000000Z"
}
```


Observações

- O identificador único do cliente será obtido através do token JWT fornecido no cabeçalho da requisição.

- Todos os campos são opcionais, e apenas os campos enviados serão atualizados.

- O campo `senha` deve seguir as mesmas regras de criação de senha (mínimo de 8 caracteres, uma letra maiúscula, uma letra minúscula, um caractere especial).

Alteração de Senha do Cliente

URL: `{{host}}/api/clientes`

Método: `PUT`

Headers: `Authorization: Bearer {{token}}`

Body:

Campo	Obrigação	Tipo	Exemplo
senah	Sim	String	

Respostas:

200 OK:

```
{
   "message": "Senha não pode ser reutilizada."
}
```

400 Bad Request

```
{
        "message": "Senha não pode ser reutilizada."
}
```

Observações

- O identificador único do cliente será obtido através do token JWT fornecido no cabeçalho da requisição.

- A nova senha não pode ser igual a senhas anteriores usadas pelo cliente.

Exclução de Cliente

URL: `{{host}}/api/clientes`

Método: `DELETE`

Headers: `Authorization: Bearer {{token}}`

Respostas:

200 OK:

```
{
"message": "Cliente excluído com sucesso."
}
```

400 Bad Request:

Usuário não encontrado ou já excluído:

```
{
"message": "Usuário não encontrado."
}
```

Observações

- O identificador único do cliente a ser excluído será obtido através do token JWT fornecido no cabeçalho da requisição.

- A operação de exclusão é permanente e não pode ser desfeita.

Contas

Cadastro de Contas

Url: {{host}}/api/contas/adicionar

Método: POST

Descrição: Adiciona uma nova conta de pagamento para um cliente.

Headers: Authorization: {{token}}

Exemplo de Resposta:


```
{
 "message": "Conta de pagamento criada com sucesso.",
 "conta": {
        "cliente_id": 1,
        "saldo": "0.00",
        "updated_at": "2024-08-01T18:39:41.000000Z",
        "created_at": "2024-08-01T18:39:41.000000Z",
        "id": 1
  }
}
```


Observações

- O identificador único do cliente a ser criado conta será obtido através do token JWT fornecido no cabeçalho da requisição.

Adicionar Créditos

Url: {{host}}/api/contas/adicionar

Método: POST

Descrição: Adiciona créditos à conta do cliente

Headers: Authorization: {{token}}

Body:

Campo	Obrigação	Tipo	Exemplo
type	Sim	String	credit, debit, pix
number	Sim, se não for pix	Número	number número do cartão deve ter 16 dígitos
brand	Sim, se não for pix	String	"visa" ou "master"
valid	Sim, se não for pix	String	data MM/YY, ex: 07/23
cvv	Sim, se não for pix	Número	123
amount	Sim	Float	valor desejado para aprovação. valores ímpares são negados e valores pares são aprovados.
chavepix	Sim, se for pix	string	

Transferir Saldo

Url: {{host}}/api/contas/transferir-saldo

Method: POST /api/contas/transferir-saldo

Descrição: Transfere saldo de uma conta para outra.

Headers: Authorization: {{token}}

Body:

Campo	Obrigação	Tipo	Exemplo
destinatario_cpf	Sim	String	CPF do destinatário da transferência
valor	Sim	Float	0.01

Exemplo de Resposta:


```
{
  "message": "Transferência realizada com sucesso."
}
```

Extrato

Url: {{host}}/api/contas/extrato

Method: GET

Descrição: Recupera o extrato de transações da conta de um cliente em um período especificado.

Headers: Authorization: {{token}}

Query Parameters:

Campo	Obrigação	Tipo	Exemplo
inicio	Sim	Date	Data de início do período Ex. 2024-08-01
fim	Sim	Date	Data de término do período maior o igual a inicio Ex. 2024-08-30

Exemplo de Resposta:


```
{
 "message": "Extrato gerado com sucesso.",
 "transacoes": [
        {
        "data": "2024-08-01T10:00:00.000000Z",
        "descricao": "Compra",
        "valor": "50.00"
        }
 ]
}
```

Observações

- Substitua `{{host}}` pelo endereço do servidor onde a API está hospedada.
- Substitua `{{token}}` pelo token de autorização válido para acessar os endpoints protegidos.

Esta é uma versão inicial da documentação. Caso precise de mais detalhes ou novos endpoints, por favor, forneça mais informações ou atualize o arquivo Postman.

Melhoria para o futuro

1. Suporte a Métodos de Pagamento Alternativos

- **Pagamentos Recurrentes:** Adicionar métodos para configurar e gerenciar pagamentos recorrentes, permitindo cobranças automáticas em intervalos definidos (semanal, mensal, anual).
- **Pagamentos Divididos:** Implementar um método para dividir o pagamento entre múltiplos destinatários em uma única transação, útil para marketplaces ou serviços com múltiplos fornecedores.

2. Funcionalidades de Relatórios e Análises

- **Relatórios Detalhados de Transações:** Implementar endpoints que gerem relatórios detalhados sobre transações, filtrados por data, tipo de transação, ou cliente, com a opção de exportar para formatos como CSV ou PDF.
- **Dashboard de Performance:** Adicionar um endpoint que forneça dados para criar dashboards com métricas financeiras em tempo real, como volume de transações, valores de saldo médio, e taxas de conversão.

3. Gestão Avançada de Contas

- **Limites de Transações Personalizados:** Introduzir métodos para definir limites de transações por conta, como um valor máximo diário, semanal ou mensal, para controle financeiro.
- **Histórico de Alterações de Conta:** Implementar um log de auditoria para rastrear todas as mudanças realizadas em contas, como alterações de dados cadastrais ou limites de crédito.

4. Expansão de Funcionalidades de Transferência

- **Transferências Agendadas:** Adicionar suporte para agendar transferências futuras, permitindo que os usuários programem transferências em datas específicas.
- **Transferências Internacionais:** Implementar suporte para transferências internacionais, com opções para conversão de moeda e taxas associadas.

5. Notificações e Alertas

- **Notificações Push:** Adicionar um método para enviar notificações push aos clientes através de dispositivos móveis, informando sobre atividades de conta, como transferências ou falhas de pagamento.

- **Alertas de Saldo:** Implementar um sistema de alertas de saldo, onde os clientes podem configurar notificações quando o saldo de suas contas cair abaixo de um determinado valor.

6. Gestão de Fraude e Segurança

- **Detecção de Fraude em Tempo Real:** Integrar um sistema de detecção de fraude que analise transações em tempo real e sinalize atividades suspeitas, permitindo bloqueio automático de transações.
- **Autenticação de Transações Sensíveis:** Adicionar um segundo fator de autenticação (como um código SMS ou autenticação biométrica) para transações acima de determinado valor ou para novas contas bancárias.

7. Suporte a Cartões de Crédito e Débito

- **Emissão de Cartões Virtuais:** Implementar a emissão de cartões de crédito ou débito virtuais que possam ser usados para compras online ou vinculados a contas específicas dentro do sistema.
- **Tokenização de Cartões:** Adicionar métodos para tokenizar cartões de crédito, aumentando a segurança ao armazenar informações de pagamento de clientes para transações futuras.