

Diário de Bordo - Sessão de Teste #01

Smart Wallet - NotusLabs API

Data: 30/09/2025

Sessão de Teste

1. Qual é o objetivo desta sessão?

Validar os endpoints de **Smart Wallet** da API NotusLabs, focando em:

- Criação de Smart Wallets através do endpoint `/wallets/register`
- Consulta de informações de wallets específicas via `/wallets/address`
- Listagem de todas as wallets do projeto via `/wallets`
- Testar validações e tratamento de erros em diversos cenários
- Avaliar a qualidade da Developer Experience (DX) e documentação

2. Qual abordagem você vai usar?

Ferramenta: Postman para testes de API REST

Metodologia:

1. **Fase 1 (✓) - Happy Path:** testar cenários de sucesso para validar funcionamento básico
2. **Fase 2 (✗) - Edge Cases:** testar cenários de erro para validar robustez das validações

Cenários de teste:

- ✓ Criar Smart Wallet com sucesso
- ✓ Consultar Smart Wallet criada
- ✓ Listar todas as wallets do projeto
- ✗ Tentar criar wallet duplicada (mesmo salt)
- ✗ Tentar criar wallet com factory inválido
- ✗ Tentar criar wallet com EOA malformado
- ✗ Tentar criar wallet sem campo obrigatório
- ✗ Tentar acessar API sem autenticação

3. Há algo que precisa ser configurado antes de começar?

Configurações realizadas:

- Conta NotusLabs criada
- API Key obtida no portal NotusLabs
- Postman instalado e configurado

- Variáveis de ambiente configuradas:
 - `base_url` : `https://api.notus.team/api/v1`
 - `api_key` : [obtida no portal]
 - `eo_address` : `0x96832a27567538a804dfb8493513d7199ac63944`
- Private Key da carteira EOA exportada do MetaMask
- Factory Address identificado: `0x00000000000400CdFef5E2714E63d8040b700BC24`
- Documentação consultada em [www !\[\]\(694fcb4611893e9db5249daba48abfc1_img.jpg\) https://docs.notus.team/docs/guides](https://docs.notus.team/docs/guides)
- Discord da NotusLabs acessado para suporte

Ambiente:

- Rede blockchain: Sepolia (Ethereum Testnet)
- Wallet provider: MetaMask
- Tipo de conta: Externally Owned Account (EOA)






4. Você conseguiu atingir o objetivo da sessão?

- **Sim**

Detalhamento:

Conseguir testar com sucesso os principais endpoints de Smart Wallet:

-  **POST /wallets/register** - 6 cenários testados (1 sucesso + 5 casos de erro)
-  **GET /wallets/address** - Funciona, mas com workaround necessário
-  **GET /wallets** - Funciona perfeitamente

Estatísticas:

- Total de testes: 8
- Testes de sucesso: 3
- Testes de erro: 5
- Problemas de DX identificados: 4



5. Problemas encontrados

Problema #1: **GET /wallets/address exige parâmetros desnecessários** **CRÍTICO**

Descrição: o endpoint `GET /wallets/address` não funciona apenas com o parâmetro `address` . A API exige também `externallyOwnedAccount` e `factory` , tornando a consulta complexa e contraintuitiva.

Como reproduzir:

```
GET /wallets/address?address=0x34378c28a87ac84266211aa9b1c77caca241a659
```

Erro retornado:

```
{
  "message": "Bad Request",
  "id": "BAD_REQUEST",
  "errors": [
    {
      "path": ["externallyOwnedAccount"],
      "message": "Required"
    },
    {
      "path": ["factory"],
      "message": "Required"
    }
  ]
}
```

Impacto:

- Desenvolvedores precisam armazenar 3 informações (address + EOA + factory) ao invés de apenas o address
- Contraria padrões de APIs Web3 (normalmente apenas address é suficiente)
- Aumenta complexidade de integração
- Não está claro na documentação que esses parâmetros são obrigatórios

Workaround: adicionar todos os parâmetros na query:

```
GET /wallets/address?address=...&externallyOwnedAccount=...&factory=...
```

Severidade: CRÍTICA - Impacta diretamente a usabilidade

Problema #2: Mensagem de erro genérica para EOA inválido ● MÉDIO

Descrição: ao enviar um endereço EOA malformato, a API retorna apenas "Invalid" sem explicar o formato esperado.

Como reproduzir:

```
POST /wallets/register
Body: {
  "externallyOwnedAccount": "0xinvalido",
  "factory": "0x000000000000400CdFef5E2714E63d8040b700BC24",
  "salt": "100"
}
```

Erro retornado:

```
{
  "message": "Bad Request",
  "errors": [{
    "validation": "regex",
    "code": "invalid_string",
    "message": "Invalid",
    "path": ["externallyOwnedAccount"]
  }]
}
```

Impacto:

- Mensagem muito genérica - não explica que deve ser um endereço Ethereum válido
- Desenvolvedor precisa adivinhar o formato correto

- Inconsistente com outras mensagens de erro (ex: factory não suportado tem mensagem clara)

Sugestão de melhoria: mensagem mais específica como: "Invalid Ethereum address format. Expected: 0x followed by 40 hexadecimal characters"

Severidade: MÉDIA - Funciona, mas dificulta troubleshooting

Problema #3: Campo "salt" tem valor padrão não documentado ● MÉDIO

Descrição: ao omitir o campo `salt`, a API não retorna erro de validação. Em vez disso, assume valor padrão "0" e tenta criar a wallet.

Como reproduzir:

```
POST /wallets/register
Body: {
  "externallyOwnedAccount": "0x96832a27567538a804dfb8493513d7199ac63944",
  "factory": "0x00000000000400CdFef5E2714E63d8040b700BC24"
  // sem campo "salt"
}
```

Comportamento observado:

- Não retorna erro de campo obrigatório
- Assume salt="0" automaticamente
- Se salt="0" já existe, retorna erro de wallet duplicada

Impacto:

- Não fica claro que salt é opcional
- Desenvolvedor pode omitir salt acidentalmente
- Pode receber erro de duplicação sem entender o motivo
- Documentação não menciona que salt tem valor padrão

Sugestão de melhoria:

- Documentar claramente: "salt é opcional, valor padrão: '0'"
- Ou tornar salt obrigatório para forçar intenção explícita
- Melhorar mensagem de erro quando salt omitido causa duplicação

Severidade: MÉDIA - Pode causar confusão

Problema #4: Erro de autenticação impreciso ● MÉDIO

Descrição: ao fazer requisição sem o header `x-api-key`, a API retorna status 403 com mensagem "Invalid Api key", quando deveria indicar que a chave está ausente.

Como reproduzir:

```
POST /wallets/register
(sem header x-api-key)
```

Erro retornado:

```
Status: 403 Forbidden
{
  "message": "Invalid Api key.",
}
```

```
"id": "HTTP_EXCEPTION"
```

```
}
```

Problemas identificados:

1. **Status code incorreto:** deveria ser 401 Unauthorized (não autenticado) ao invés de 403 Forbidden (sem permissão)
2. **Mensagem imprecisa:** diz "Invalid" mas a chave está ausente, não inválida
3. **Não diferencia:** entre chave ausente vs chave inválida
4. **Falta orientação:** não menciona qual header é esperado (x-api-key)
5. **Inconsistência:** outros erros incluem traceId, este não

Impacto:

- Desenvolvedor pode procurar erro na chave quando na verdade esqueceu de fornecê-la
- Não diferencia entre "esqueci de adicionar" vs "chave errada"

Sugestão de melhoria:

Status: 401 Unauthorized

```
{
```

```
  "message": "API key is missing. Please provide your API key in the 'x-api-key' header.",  
  "id": "API_KEY_MISSING",  
  "traceId": "..."
```

```
}
```

Severidade: MÉDIA - Funciona, mas mensagem confusa

Observações sobre validações que funcionaram bem: ✓

Duplicate Salt:

- Status 400 apropriado
- Mensagem clara: "This wallet is already registered"
- ID específico: "WALLET_ALREADY_REGISTERED"
- Inclui traceId
- ✓ Excelente DX

Invalid Factory:

- Status 400 apropriado
- Mensagem específica com o factory inválido
- Sugere solução: "contact support to request its inclusion"
- ID específico: "FACTORY_NOT_SUPPORTED"
- ✓ Excelente DX

6. Observações adicionais

Pontos Positivos da API ✓

1. Funcionalidade Core Sólida:

- Endpoint de registro funciona perfeitamente
- Criação de wallets é rápida (~500ms)
- Multi-chain support bem implementado (9 blockchains)

2. Respostas Completas:

- Informações detalhadas sobre cada wallet
- Status de deployment por chain
- Timestamps para auditoria
- Factory e implementation addresses incluídos

3. Validações Robustas:

- Impede duplicação de wallets corretamente
- Valida factory addresses
- Valida formato de EOA (regex)

4. Algumas Mensagens de Erro Excelentes:

- Factory não suportado: mensagem clara + sugestão de ação
- Wallet duplicada: mensagem específica + endereço
- Pattern estruturado (message, id, traceId)



Sugestões de Melhoria 💡

1. Endpoint GET /wallets/address:

- **Crítico:** permitir consulta apenas por address
- Criar endpoint alternativo: GET /wallets/{address}
- Ou tornar EOA e factory opcionais
- Documentar claramente por que múltiplos parâmetros são necessários

2. Mensagens de Erro:

- Padronizar qualidade das mensagens (algumas são excelentes, outras genéricas)
- Sempre incluir traceId para debug
- Usar IDs específicos ao invés de genéricos ("INVALID_EOA_FORMAT" vs "BAD_REQUEST")
- Especificar formato esperado em validações

3. Documentação:

- Adicionar seção "Supported Networks" listando blockchains
- Explicar campos técnicos (eip7702, implementation, etc.)
- Documentar claramente parâmetros obrigatórios vs opcionais
- Explicar quando/se wallet precisa ser deployada on-chain
- Adicionar exemplos de uso do campo metadata
- Criar tabela de códigos de erro e suas soluções

4. Paginação:

- Adicionar paginação ao endpoint GET /wallets para projetos com muitas wallets
- Incluir total count de wallets
- Permitir filtros (por EOA, factory, salt, status de deployment)

5. Novos Endpoints Úteis:

- GET /factories - Listar factories suportados
- GET /wallets/{address} - Consulta simplificada por address
- GET /networks - Listar blockchains suportadas



Dúvidas Técnicas ?

1. Campo eip7702:

- O que este campo significa?
- Quando ele é true?
- Como afeta o funcionamento da wallet?

2. Deployment Status:

- Todas as wallets retornam `deployed: false`
- A wallet precisa ser deployada manualmente?
- Ou ela deploya automaticamente ao ser usada?
- Como fazer o deployment se necessário?

3. Metadata:

- Como usar o campo metadata?
- Qual formato aceita? (JSON? String?)
- Há limite de tamanho?
- Como atualizar via PATCH?

4. Multi-Chain:

- Como especificar em qual chain operar?
- A wallet funciona em todas as chains simultaneamente?
- Como fazer deployment seletivo por chain?



Insights Sobre DX 💬

O que funcionou muito bem:

- Simplicidade do endpoint de registro
- Responses ricas em informações
- Algumas mensagens de erro são exemplares
- Estrutura da API é RESTful e intuitiva

Principais fricções:

- GET /wallets/address não intuitivo (problema mais crítico)
- Inconsistência na qualidade das mensagens de erro
- Documentação incompleta sobre conceitos técnicos
- Falta de clareza sobre parâmetros opcionais

Comparação com outras APIs Web3:

- NotusLabs tem responses mais completas que a maioria
- Mas endpoint de consulta é menos intuitivo que Etherscan, Alchemy, Infura
- Mensagens de erro melhores que muitas APIs Web3
- Multi-chain support é um diferencial positivo

Impacto para desenvolvedores Web2:

- API é acessível para devs sem experiência Web3
- Mas alguns conceitos blockchain precisam de mais explicação
- Documentação poderia ter mais "learning path"
- Exemplos práticos ajudariam muito