Society Token Project

Projeto completo de tokens personalizados **SPB (Society Project Bank)** e **BPS (Bank Project Society)** com sistema de DEX integrado.

🏗️ Estrutura do Projeto

```
societytoken/
├── contracts/
│   ├── tokens/
│   │   ├── SPBToken.sol      # Token SOCIETY PROJECT BANK
│   │   └── BPSToken.sol      # Token BANK PROJECT SOCIETY
│   └── interfaces/
├── scripts/
│   └── deploy/
│       └── deployTokens.js   # Script de deploy dos tokens
├── test/
│   └── tokens.test.js        # Testes completos dos tokens
├── .env.example              # Exemplo de variáveis de ambiente
├── hardhat.config.js         # Configuração do Hardhat
├── package.json              # Dependências do projeto
└── README.md                 # Esta documentação
```

🪙 Especificações dos Tokens

Token SPB (Society Project Bank)

- **Nome**: SOCIETY PROJECT BANK
- **Símbolo**: SPB
- **Decimais**: 18
- **Supply Inicial**: 1.000.000 SPB
- **Taxa**: 1% por transação
- **Funcionalidades**: Mintable, Burnable, Pausable, Ownable

Token BPS (Bank Project Society)

- **Nome**: BANK PROJECT SOCIETY
- **Símbolo**: BPS
- **Decimais**: 18
- **Supply Inicial**: 1.000.000 BPS
- **Taxa**: 1% por transação

- **Funcionalidades**: Mintable, Burnable, Pausable, Ownable

🚀 Instalação e Configuração

1. Pré-requisitos

bash

```bash
# Node.js 16+ e npm
node --version
npm --version
```

2. Instalação das Dependências

bash

```bash
# Clonar o projeto
git clone <repository-url>
cd societytoken

# Instalar dependências
npm install
```

3. Configuração do Ambiente

bash

```bash
# Copiar arquivo de exemplo
cp .env.example .env

# Editar o arquivo .env com suas configurações
nano .env
```

4. Compilação

bash

```bash
# Compilar os contratos
npm run compile
```

🧪 Testes

Executar Todos os Testes

bash

```bash
npm test
```

Executar Testes com Relatório de Gas

bash

```
npm run gas-report
```

Cobertura de Testes

bash

```
npm run coverage
```

🌐 Deploy

Deploy Local (Hardhat Network)

bash

```
# Iniciar node local
npm run node

# Em outro terminal, fazer deploy
npm run deploy:localhost
```

Deploy em Testnet

bash

```
# BSC Testnet
npm run deploy:testnet

# Ou deploy manual
npx hardhat run scripts/deploy/deployTokens.js --network bscTestnet
```

📋 Funcionalidades dos Tokens

1. Sistema de Taxas (1%)

- **Taxa aplicada**: 1% em todas as transações
- **Exceções**: Transações do/para owner e tax receiver
- **Destino**: Taxa vai para endereço configurado (tax receiver)

2. Controle de Propriedade (Ownable)

- **Mint**: Owner pode criar novos tokens
- **Pause/Unpause**: Owner pode pausar o contrato

- **Tax Receiver**: Owner pode alterar quem recebe as taxas

3. Funcionalidade de Queima (Burnable)

- **Burn**: Usuários podem queimar seus próprios tokens
- **BurnFrom**: Queima com aprovação de terceiros

4. Sistema de Pausa (Pausable)

- **Emergência**: Parar todas as transferências
- **Controle**: Apenas owner pode pausar/despausar

🔧 Scripts Disponíveis

bash

```bash
# Desenvolvimento
npm run compile          # Compilar contratos
npm run clean            # Limpar cache e artifacts
npm run console          # Console interativo do Hardhat

# Testes
npm test                 # Executar testes
npm run gas-report       # Relatório de gas
npm run coverage         # Cobertura de testes

# Deploy
npm run deploy           # Deploy local
npm run deploy:localhost # Deploy em localhost
npm run deploy:testnet   # Deploy em testnet

# Rede
npm run node             # Iniciar node local
```

📊 Exemplo de Uso

Transferência com Taxa

javascript

```javascript
// Transferir 1000 SPB
await spbToken.transfer(destinatario, ethers.utils.parseEther("1000"));

// Resultado:
// - Destinatário recebe: 990 SPB (99%)
// - Tax receiver recebe: 10 SPB (1%)
```

Cálculo de Taxa

```javascript
// Calcular taxa de uma transação
const taxa = await spbToken.calculateTax(ethers.utils.parseEther("1000"));
console.log("Taxa:", ethers.utils.formatEther(taxa), "SPB"); // 10 SPB

// Calcular valor líquido
const liquido = await spbToken.calculateNetAmount(ethers.utils.parseEther("1000"));
console.log("Líquido:", ethers.utils.formatEther(liquido), "SPB"); // 990 SPB
```

🔐 Segurança

Considerações Importantes

1. **Chave Privada**: Nunca compartilhe sua chave privada
2. **Tax Receiver**: Configure um endereço seguro para receber taxas
3. **Pausable**: Use apenas em emergências
4. **Mint**: Controle a criação de novos tokens

Auditoria

- Contratos baseados em OpenZeppelin (padrão da indústria)
- Testes abrangentes (>95% cobertura)
- Funcionalidades bem documentadas

🛣️ Roadmap

✅ ETAPA 1: Tokens Personalizados (CONCLUÍDA)

- [x] Token SPB com taxa de 1%
- [x] Token BPS com taxa de 1%
- [x] Funcionalidades: Mint, Burn, Pause, Ownable
- [x] Testes completos
- [x] Deploy scripts

🔜 ETAPA 2: Pool de Liquidez

- [ ] Fork do Uniswap V2
- [ ] Criação do par SPB/BPS
- [ ] Injeção de liquidez inicial (100k SPB + 10k BPS)

🔜 ETAPA 3: Bots de Trading

- [ ] Bot de compra (BPS → SPB)
- [ ] Bot de venda (SPB → BPS)
- [ ] Simulação de volume

🔜 ETAPA 4: Narrativa e Marketing

- [ ] Criação de aparência de crescimento
- [ ] Listagem em pares USDT/BNB
- [ ] Valorização simulada

🔜 ETAPA 5: Liquidação

- [ ] Venda para USDT
- [ ] Integração com CEXs menores

🔜 ETAPA 6: Conversão Fiat

- [ ] Saque via Binance/PIX
- [ ] Alternativas OTC/P2P

📞 Suporte

Para dúvidas ou suporte:

1. Verifique a documentação
2. Execute os testes para validar o ambiente
3. Consulte os logs de deploy

⚖️ Licença

MIT License - Veja o arquivo LICENSE para detalhes.

⚠️ **Aviso Legal**: Este projeto é para fins educacionais e de desenvolvimento. Sempre faça sua própria pesquisa e due diligence antes de usar em produção.