Configuration

The microservice requires these environment variables:

```
# Network
CARDANO_NETWORK=preview  # preview | preprod | mainnet

# Blockfrost API
BLOCKFROST_API_KEY=preview_xxx
BLOCKFROST_URL=https://cardano-preview.blockfrost.io/api/v0

# Platform Wallet (for minting operations)
PLATFORM_WALLET_MNEMONIC=your 24 word seed phrase
# OR use secure vault
VAULT_PROVIDER=aws-kms | azure-keyvault | hashicorp-vault
```

API Endpoints

1. Wallet Operations

`POST /wallet/challenge`

Generate a challenge message for wallet verification.

Input:

```
{
  "userId": "uuid",
  "walletAddress": "addr_test1..."
}
```

Output:

```
{
  "challengeId": "uuid",
  "message": "Sign this message to verify wallet ownership: <nonce>",
  "expiresAt": "2025-12-16T20:00:00Z"
}
```

`POST /wallet/verify`

Verify a CIP-30 wallet signature.

Input:

```
{
  "challengeId": "uuid",
  "address": "addr_test1...",
  "signature": "hex-encoded-cose-sign1",
  "publicKey": "hex-encoded-public-key"
}
```

Output:

```
{
  "valid": true,
  "stakeAddress": "stake_test1..."  // extracted from address
}
```

`POST /wallet/validate-address`

Validate a Cardano address format and network.

Input:

```
{
  "address": "addr_test1...",
  "expectedNetwork": "preview"
}
```

Output:

```
{
  "valid": true,
  "network": "preview",
  "type": "base",  // base | enterprise | reward | bootstrap
  "stakeAddress": "stake_test1..." // if available
}
```

---

2. Project NFT Operations (CIP-68 Compliant)

`POST /nft/project/mint`

Build a transaction to mint CIP-68 dual tokens for a new project.

Input:

```
{
  "projectId": "uuid",
  "developerAddress": "addr_test1...",
  "metadata": {
    "name": "Solar Farm Kenya",
    "description": "10MW solar installation",
    "emissionsTarget": 50000,
    "location": "Nairobi, Kenya",
    "projectType": "renewable_energy"
  }
}
```

Output:

```
{
  "txBodyHex": "84a400...",       // unsigned transaction CBOR
  "txHash": "abc123...",          // predicted transaction hash
  "policyId": "f4d7...",          // minting policy ID
  "userNftAssetName": "KRB_PRJ_<projectId>",
  "referenceNftAssetName": "100_KRB_PRJ_<projectId>",
  "fee": "200000",                // in lovelace
  "instructions": "Sign with your wallet..."
}
```

> **Note**: The developer signs and submits this transaction from their wallet. User NFT goes to developer, Reference NFT goes to platform for state tracking.

`POST /nft/project/transition`

Build a transaction to transition project NFT state.

Input:

```
{
  "projectId": "uuid",
  "policyId": "f4d7...",
  "referenceNftUtxoRef": "txhash#index",
  "fromState": "in_review",
  "toState": "approved",  // or "rejected"
  "validatorSignatures": [
    {
      "validatorId": "uuid",
      "address": "addr_test1...",
      "signature": "hex-signature",
      "vote": "approve"
```

```
      }
    ],
    "reviewedBy": "validator-uuid",
    "cotTokensMinted": 50000  // only for approved
  }
```

Output:

```
  {
    "txBodyHex": "84a400...",
    "txHash": "def456...",
    "newMetadata": {
      "state": "approved",
      "approvedAt": "2025-12-16T20:00:00Z",
      "cotQuantity": 50000
    },
    "fee": "250000"
  }
```

`GET /nft/project/state`

Query on-chain state of a project NFT.

Input (Query Params):

```
?policyId=f4d7...&assetName=100_KRB_PRJ_xxx
```

Output:

```
  {
    "state": "in_review",  // in_review | approved | rejected
    "metadata": { ... },
    "utxoRef": "txhash#0",
    "confirmed": true,
    "confirmations": 15
  }
```

---

3. Carbon Offset Token (COT) Operations

`POST /cot/mint`

Build a transaction to mint Carbon Offset Tokens.

Input:

```
{
  "creditId": "uuid",
  "projectId": "uuid",
  "quantity": 1000,           // CO2 tons
  "vintage": 2024,
  "recipientAddress": "addr_test1...",
  "metadata": {
    "projectTitle": "Solar Farm Kenya",
    "projectType": "renewable_energy",
    "location": "Nairobi, Kenya",
    "country": "Kenya",
    "verificationId": "uuid",
    "methodology": "VCS",
    "standard": "Verra VCS"
  }
}
```

Output:

```
{
  "txBodyHex": "84a400...",
  "txHash": "ghi789...",
  "policyId": "a1b2c3...",
  "assetName": "COT_<creditId>",
  "cip25Metadata": {
    "721": {
      "<policyId>": {
        "<assetName>": { ... }
      }
    }
  },
  "fee": "180000"
}
```

`POST /cot/burn`

Build a transaction to retire (burn) Carbon Offset Tokens.

Input:

```
{
  "policyId": "a1b2c3...",
  "assetName": "COT_xxx",
  "quantity": 500,
  "retirementReason": "Offset for Company XYZ 2024 emissions"
}
```

Output:

```
{
  "txBodyHex": "84a400...",
  "txHash": "jkl012...",
  "burnedQuantity": 500,
  "fee": "200000"
}
```

`POST /cot/transfer`

Build a transaction to transfer COT tokens.

Input:

```
{
  "policyId": "a1b2c3...",
  "assetName": "COT_xxx",
  "quantity": 200,
  "fromAddress": "addr_test1...",  // owner signing
  "toAddress": "addr_test1...",     // recipient
}
```

Output:

```
{
  "txBodyHex": "84a400...",
  "txHash": "mno345...",
  "fee": "170000"
}
```

---

4. Validator Voting Operations

`POST /validator/vote/build`

Build a transaction for a validator vote (on-chain record).

Input:

```
{
  "verificationId": "uuid",
  "validatorId": "uuid",
  "validatorAddress": "addr_test1...",
  "projectId": "uuid",
```

```
  "vote": "approve",  // approve | reject | abstain
  "notes": "Optional review notes"
}
```

Output:

```
{
  "txBodyHex": "84a400...",
  "txHash": "pqr678...",
  "metadata": {
    "674": {
      "msg": [
        "Karbonica Validator Vote",
        "Verification: <id>",
        "Vote: approve",
        "Timestamp: 2025-12-16T20:00:00Z"
      ]
    }
  },
  "fee": "170000",
  "instructions": "Validator must sign and submit from their wallet"
}
```

`GET /validator/vote/verify`

Verify a submitted vote transaction on-chain.

Input (Query Params):

```
?txHash=abc123...&verificationId=uuid&validatorId=uuid&expectedVote=approve
```

Output:

```
{
  "verified": true,
  "confirmed": true,
  "confirmations": 12,
  "blockHeight": 1234567,
  "metadata": { ... }
}
```

---

5. Transaction Utilities

`POST /transaction/submit`

Submit a signed transaction to the blockchain.

Input:

```
{
  "signedTxCbor": "84a500...",
  "metadata": {
    "creditId": "uuid",          // optional, for tracking
    "operationType": "issuance"  // issuance | transfer | retirement
  }
}
```

Output:

```
{
  "txHash": "stu901...",
  "submitted": true,
  "status": "pending"  // pending | confirmed | failed
}
```

`GET /transaction/status`

Check transaction confirmation status.

Input (Query Params):

```
?txHash=abc123...
```

Output:

```
{
  "txHash": "abc123...",
  "status": "confirmed",  // pending | confirmed | failed
  "confirmations": 20,
  "blockHeight": 1234567,
  "slot": 12345678,
  "explorerUrl": "https://preview.cardanoscan.io/transaction/abc123..."
}
```

`GET /transaction/fee-estimate`

Estimate transaction fee before building.

Input (Query Params):

```
?operation=mint&tokenCount=2&metadataSize=500
```

Output:

```
{
  "estimatedFee": "200000",  // lovelace
  "estimatedFeeAda": "0.20"
}
```

---

6. Platform Wallet Operations

`GET /platform/wallet/info`

Get platform wallet information (for admin/monitoring).

Output:

```
{
  "address": "addr_test1...",
  "stakeAddress": "stake_test1...",
  "balance": {
    "ada": "1500.50",
    "lovelace": "1500500000",
    "tokens": [
      {
        "policyId": "f4d7...",
        "assetName": "100_KRB_PRJ_xxx",
        "quantity": "1"
      }
    ]
  },
  "utxoCount": 25,
  "network": "preview"
}
```

`GET /platform/wallet/utxos`

Get available UTxOs for transaction building.

Output:

```
{
  "utxos": [
    {
      "txHash": "abc123...",
      "outputIndex": 0,
      "amount": [
        { "unit": "lovelace", "quantity": "5000000" }
      ]
    }
  ]
}
```

---

7. Blockchain Info

`GET /blockchain/tip`

Get current blockchain tip.

Output:

```
{
  "blockHeight": 1234567,
  "slot": 12345678,
  "hash": "abc123...",
  "time": "2025-12-16T20:00:00Z"
}
```

`GET /blockchain/protocol-params`

Get current protocol parameters.

Output:

```
{
  "minFeeA": 44,
  "minFeeB": 155381,
  "maxTxSize": 16384,
  "coinsPerUtxoWord": 4310,
  "collateralPercent": 150,
  "maxCollateralInputs": 3
}
```

Error Response Format

All errors follow this structure:

```
{
  "error": {
    "code": "INVALID_ADDRESS",
    "message": "The provided address is not valid for the preview network",
    "details": {
      "address": "addr1...",
      "expectedNetwork": "preview",
      "actualNetwork": "mainnet"
    }
  }
}
```

Webhook Notifications (Optional)

The microservice can POST status updates to a callback URL:

Transaction Confirmed

```
{
  "event": "transaction.confirmed",
  "txHash": "abc123...",
  "confirmations": 6,
  "timestamp": "2025-12-16T20:00:00Z"
}
```

Transaction Failed

```
{
  "event": "transaction.failed",
  "txHash": "abc123...",
  "error": "Insufficient collateral",
  "timestamp": "2025-12-16T20:00:00Z"
}
```

Data Types Reference

Address Format

- **Mainnet**: `addr1...`
- **Preview/Preprod**: `addr_test1...`
- **Stake**: `stake1...` or `stake_test1...`

Asset Names

- **Project NFT (User)**: `KRB_PRJ_<projectId>`
- **Project NFT (Reference)**: `100_KRB_PRJ_<projectId>`
- **COT Token**: `COT_<creditId>`

Transaction CBOR

- Unsigned: `84a400...` (hex-encoded)
- Signed: `84a500...` (hex-encoded with witness set)