**Breez vs LNBits Comparison**

Based on your codebase, you're already using LNBits. Here's a comparison to help you decide:

**LNBits (Current Implementation)**

* **Custodial solution**: Your code uses a centralized wallet with admin/invoice keys
* **Simple API**: Your implementation shows straightforward invoice creation and status checking
* **Self-hosted option**: Can be deployed on your own server
* **Free and open-source**

**Breez SDK**

* **Non-custodial**: Users control their own funds (better security)
* **Mobile-first**: Designed for mobile apps with native SDKs
* **Built-in swap services**: Fiat on/off ramps
* **Simplified UX**: Abstracts Lightning complexity
* **Production-ready**: Used by many commercial apps

**Recommendation**

For your Sats Jar Junior project:

1. **If targeting parents with limited technical knowledge**: Breez SDK provides better UX and non-custodial security
2. **Implementation approach**:
   * Replace your LNBits service with Breez SDK
   * Each parent/child could have their own wallet instead of using a centralized one
3. **Migration considerations**:
   * You'd need to modify wallet creation, invoice generation, and payment verification
   * Add Breez SDK to your dependencies

**Why Breez SDK Over LNBits for Sats Jar Junior**

**Key Advantages for Your Specific Use Case**

1. **Non-custodial architecture**
   * Your current LNBits implementation uses a centralized wallet with shared keys
   * Breez gives each user their own wallet, reducing your liability and security concerns
2. **Better for educational goals**
   * Sats Jar Junior aims to teach children about digital finance
   * Non-custodial solutions better demonstrate true ownership concepts
3. **Mobile-optimized**
   * Your notes mention targeting Kenyan families with basic smartphones
   * Breez SDK is specifically designed for mobile experiences
4. **Simplified parent experience**
   * Parents with limited technical knowledge will find Breez's UX more approachable
   * Reduces support burden for your team
5. **Built-in fiat on/off ramps**
   * Your app needs KES to BTC conversion
   * Breez has integrated swap services that could simplify this process

**Cost Comparison: Breez SDK vs. LNBits for Production App**

**Breez SDK Costs**

* **License Fee**: Free and open-source
* **Transaction Fees**:
  + 0.4% fee on incoming payments (can be passed to users)
  + No fee for outgoing payments
* **Infrastructure**:
  + No need to run your own Lightning node
  + Reduced server costs as wallet logic runs on user devices

**LNBits Costs (Current Implementation)**

* **License Fee**: Free and open-source
* **Infrastructure Costs**:
  + Must run and maintain your own Lightning node
  + Higher server costs for centralized wallet management
  + Backup and security costs for custodial wallet
* **Liability Insurance**: Potentially needed since you hold user funds

**For Sats Jar Junior in Kenya**

* **Breez Advantage**: Lower operational costs, especially important for a service targeting children in Kenya
* **Scaling Costs**: Breez scales better as user costs are distributed across users' devices
* **Risk Management**: Reduced financial and legal liability with non-custodial approach

The 0.4% Breez fee on incoming payments is likely offset by your reduced infrastructure and liability costs, making it more economical for your target market and use case.

**Understanding the Lightning Payment Process - Step by Step**

Let me explain the entire process in simple terms:

**1. How Lightning Payments Work**

Think of LNBits like a digital cash register:

1. **Creating an Invoice** = Making a price tag
   * User wants to deposit 100 sats
   * You add 10 sats fee
   * Total: 110 sats
2. **Payment** = Customer pays the price tag
   * Someone scans QR code
   * They pay 110 sats
   * LNBits receives the money
3. **Confirmation** = Cash register dings
   * LNBits tells your app "payment received!"
   * Your app updates user's balance (+100 sats)
   * Your app records your fee (+10 sats)

**2. Where Does the Money Go?**

The money goes to your LNBits wallet. Think of it like:

* You have one big company bank account (LNBits wallet)
* You keep records of how much belongs to each user
* Users don't have separate physical wallets - just balances in your database

**3. Collecting Transaction Fees**

When someone pays 110 sats:

* 100 sats get added to user's balance in your database
* 10 sats are your fee (stays in your LNBits wallet but not added to user balance)

**4. Verifying Real Money**

To make sure you actually received money:

* LNBits sends a webhook notification when payment arrives
* You can also check your LNBits wallet balance directly
* Regular reconciliation ensures your records match reality

**5. Complete Flow with Code Explanation**

1. **User requests deposit**:
   * App shows QR code with 110 sats invoice (100 + 10 fee)
   * Your database records: "expecting 110 sats payment"
2. **User pays invoice**:
   * Money arrives in your LNBits wallet
   * LNBits sends webhook to your server
3. **Your server processes payment**:
   * Updates invoice status to "paid"
   * Adds 100 sats to user's balance
   * Records 10 sats as your fee
   * Creates transaction records
4. **Verification**:
   * Periodically check LNBits wallet balance
   * Compare with sum of all user balances + collected fees
   * They should match (or be very close)