HathorChat: Tokenized Communities in Telegram

Frictionless tipping, rewards & gated access—Web3 made as easy as chat.

At HathorChat, we turn any Telegram group into a living, breathing token economy—no crypto expertise required. In a single Mini App, users get an auto-provisioned wallet, can mint or tip in native HTR or custom tokens, and earn on-chain NFT badges for participation. Communities unlock paid content or private channels simply by holding or earning tokens, while admins tap powerful engagement tools—leaderboards, rewards, access control—all running on Hathor's ultra-low-fee nano contracts. The result? Deeper engagement, new monetization streams, and a seamless Web3 experience inside the world's favorite chat platform.

Key Value Propositions

- **Frictionless Tipping**
  Send HTR or community tokens in-chat with `/tip` or one-click buttons—no wallet installs, no gas headaches.

- **Tokenized Communities**
  Create branded tokens on-demand, reward top contributors, and gate premium content or private groups—all without writing a line of smart contract code.

- **No-Code Web3**
  Auto-provisioned wallets, off-chain logic, and a familiar React-based Mini App UI mean admins and users interact entirely through Telegram—zero blockchain jargon.

Problem & Market Opportunity

---

1. Pain Points

- **High Onboarding Friction**
  Most Telegram communities interested in crypto still force users to juggle external wallets, seed-phrase backups, and confusing gas-fee mechanics. For non-tech users, this creates a steep learning curve that drives away 60–70% of would-be participants before they ever tip or transact.

- **Fragmented Web3 Tooling**
  Today's ecosystem of Telegram bots is piecemeal: one bot for tipping, another for NFT drops, a third for token gating—and none of them share wallet state or UI. This fragmentation means admins must cobble together multiple services, and users must learn several disparate interfaces.

- **Cost & Complexity of Smart Contracts**
  Deploying custom tokens or NFTs on platforms like Ethereum or Solana requires writing, auditing, and paying for contracts—often hundreds to thousands of dollars in gas alone. Small communities lack the resources and expertise to absorb these costs, so they never get off the ground.

- **Limited Engagement Incentives**
  Without built-in mechanisms to reward or recognize contributions, communities depend on manual shout-outs, spreadsheets, or off-chain leaderboards that quickly become stale. This absence of meaningful, on-chain rewards stifles organic growth and reduces daily active participation.

---

2. Market Sizing

- **Telegram's Massive Reach**
  With over **700 million monthly active users**, Telegram is one of the world's largest messaging platforms. Roughly **20%** of these users participate in topic-focused groups—everything from hobbyist communities to professional networks—making Telegram a prime vector for embedded Web3 experiences.

- **Rapid Web3 Adoption**
  Global crypto ownership has surpassed **400 million users**, and DeFi/NFT engagement continues to grow by **15–20%** year-over-year. Yet, the vast majority of these users are power-users who endure clunky UX; a much larger potential audience remains locked out by complexity.

- **Bot & Mini App Economy**
  Telegram bots already power billions of messages daily; the introduction of Mini Apps unlocks even richer interactive experiences. Early adopters report **2–3×** higher engagement when features are native to the chat interface rather than off-platform.

---

3. Why Now?

- **Telegram Mini Apps Are Here**
  Telegram's Mini App SDK provides a secure, in-chat webview environment with HTTPS hosting and native auth—perfect for embedding Web3 logic without redirecting users.

- **Demand for No-Code Web3**
  Communities are clamoring for plug-and-play tokenization: a way to reward, gate, and gamify without hiring Solidity devs or paying exorbitant gas fees.

- **Hathor's Nano‑Contract Advantage**
  By leveraging Hathor's lightweight, UTXO‑based token framework and off‑chain nano contracts, we can deliver on‑chain transfers, token minting, and NFT issuance at a fraction of the cost and complexity of traditional smart‑contract platforms.

Solution Overview & Core Features

What It Is

HathorChat is a **native Telegram Mini App + Bot** stack built on the Hathor Network. Users interact entirely within Telegram's secure Mini App webview and chatbot interface—no external wallets, no browser redirects, no Solidity to learn. Behind the scenes, our Node.js/TypeScript backend calls Hathor's Wallet Library and nano‑contracts to mint, transfer, and manage tokens and NFTs on‑chain, while all business logic (leaderboards, gating rules, badge thresholds) runs off‑chain for speed and cost efficiency.

Why It's Unique

- **Nano‑Contract Simplicity**
  We leverage Hathor's UTXO‑based, off‑chain nano‑contract framework. All token logic (mint rules, badge triggers, gating thresholds) lives in lightweight server‑side modules—no complex on‑chain contracts to audit or pay for.

- **Near‑Zero Fees**
  Hathor's fee model charges fractions of a cent per transaction. Tipping, minting, and NFT issuance stay blockchain‑secure yet economically invisible to end users.

- **No‑Code Web3**
  Admins and community members never see private keys or seed phrases. Wallets are auto‑provisioned, tokens are created via simple forms, and all flows are driven by Telegram UI—zero blockchain jargon.

Core Feature Set

Feature	Description	Example / Flow

1. Auto-Provisioned Wallet *[Wallet Icon]*	The moment a user opens HathorChat, we generate & encrypt an HTR wallet tied to their Telegram ID. No manual setup—just instant on-chain capability.	**Flow:** 1. User clicks “Open Wallet” in Mini App 2. Backend calls `/api/createWallet` → generates keypair 3. Encrypted key & public address stored in DB 4. Mini App displays address & QR code
2. One-Click Token Minting *[Token Icon]*	Admins or users create branded tokens via a simple form: name, symbol, max supply, optional image. Backend calls `createToken()` and mints tokens in one transaction. Supports both **admin-controlled** and **user-driven** models.	**Flow:** 1. Admin opens “Create Token” form 2. Inputs: “CoffeeCoin”, `$BREW`, 10,000 supply 3. Backend uses Hathor API → mints token 4. Tokens appear in admin’s wallet; bot confirms “Your $BREW token is live!”
3. In-Chat Tipping *[Tip Icon]*	Send HTR or any custom token with `/tip @username 5` or via a “Send” button. Instant feedback in chat, fully on-chain P2P transfer.	**Flow:** 1. Alice types `/tip @Bob 5 $VIBE` 2. Bot fetches Alice’s encrypted key, Bob’s address 3. Backend signs & broadcasts transaction

		4. Chat: “✅ Alice tipped Bob 5 $VIBE. 🎉”
4. NFT Badge Rewards *[Badge Icon]*	Automatically mint claimable NFTs when users hit milestones (e.g., first tip, 100 messages, top contributor). Badges are on-chain collectibles that live in users’ wallets.	**Flow:** 1. Nano-contract logic detects “first tip” event 2. Backend calls `mintNFT()` with badge metadata 3. Bot posts badge image & link: “Congrats! Your Helper Badge is yours.”
5. View Balance & History *[History Icon]*	Real-time balance and transaction history panels inside the Mini App. Queries on-chain data via Hathor’s API for transparency and auditability.	**Flow:** 1. User taps “My Wallet” → “Balance” 2. Mini App calls `/api/getBalance` 3. Shows current HTR & token balances + recent tx list
. Token-Gated Access *[Lock Icon]*	Gate private groups, polls, or downloadable content based on token holdings. Simply set a threshold, and users above it see “Join VIP” or get content links.	**Flow:** 1. User clicks “Join VIP Lounge” 2. Backend checks `getBalance(user) ≥ 100 $ALPHA` 3. If yes → bot sends group invite; if no → “Earn/purchase more tokens.”

User-Created Tokens in Admin Mode

Example Flow: From Zero to Web3 Magic

1. **Onboard in 10 s**
   - User opens group, taps “Open Mini App.”
   - HathorChat auto-creates a wallet and displays a welcome NFT badge “Founding Member.”
2. **Create & Distribute Tokens**
   - Admin mints “CoffeeCoin” ($BREW) via form (30 s).
   - `$BREW` shows up in everyone’s wallet.
3. **Engage & Reward**
   - Members tip each other for helpful posts (`/tip @user 5 $BREW`).
   - First tip triggers “Helper Badge” NFT.
4. **Monetize & Gate**
   - Admin sets “Hold ≥50 $BREW to join VIP.”
   - Bot auto-invites token-holders to a premium chat.

🔧 Typical Flow (Admin-Approved Token Creation)

1. **User Opens “Create My Token”**
   - Fills out a form:
     - Name: “ArtieCoin”
     - Symbol: `$ARTY`
     - Max Supply: `10,000`
     - Description + logo
2. **Backend Receives Request**

    - You (or your backend) **review it manually** OR auto-approve if within quota.
3. **Mint Happens via Admin Wallet**
    - Backend uses admin's wallet (with HTR balance) to mint via Hathor's API.
4. **Token Is Transferred to User**
    - Once minted, the full supply is sent to the requesting user's wallet.
5. **Bot Confirms in Chat**

> "✅ Your token **$ARTY** is live with 10,000 supply!"

⚖️ Why This Is Still "Admin-Controlled"

- **You control minting rules** — like max supply limits, naming conventions, daily quotas.
- **You manage the minting cost (HTR)** — either fund it yourself or require users to tip in advance.
- **You prevent spam** — no one's minting garbage tokens without your filters.

Custom Token Creation & Customization

Beyond tipping in HTR, **HathorChat** lets you create fully **branded, custom tokens** tailored to your community's identity and purpose—no smart contracts needed. Here's how it works and what you can customize:

What You Can Customize

1. **Token Name & Symbol**
    - Pick any memorable name (e.g. "CoffeeCoin", "MemePoints")
    - Choose a shorthand symbol (e.g. **$BREW**, **$LOL**)

2. **Total Supply & Decimals**
    - Fixed cap or mint-on-demand up to your chosen max
    - Decide precision (e.g. allow fractions like 0.1 TOK)
3. **Burnability**
    - Make tokens **burnable** so users can redeem them for rewards (e.g., burn 50 $ARTY for an eBook)
4. **Icon & Metadata**
    - Upload a 128×128 PNG logo
    - Add a description, website URL, or social links
5. **Minting Model**
    - **Admin-Controlled**: You (the platform owner) approve and fund every mint
    - **User-Driven**: Any user can mint within daily quotas—perfect for creator "token packs"
    - **Hybrid**: Free mints up to a community quota, then admin approval for excess

---

Example Flow: Create Your Community Token

1. **Open "Create Token" Form**
    - In the Mini App, the admin (or any authorized user) taps **"New Token"**.
2. **Fill in Details**
    - **Name**: "MemePoints"
    - **Symbol**: `$LOL`
    - **Max Supply**: `100,000`
    - **Decimals**: `0` (whole units only)
    - **Burnable**: ☑️ (to let users redeem for swag)

   - **Upload Icon**: A funny meme face PNG

3. **Submit & Mint**
   - Backend calls Hathor's `createToken()` API; one on-chain transaction mints the entire supply into the creator's wallet.

4. **Instant Confirmation**
   - Bot posts:

     "✅ Your token **$LOL** is live! Total supply: 100 000. Share `/tip @user 10 $LOL` to start rewarding memes."

Using Your Custom Token

- **Tipping**:
  `/tip @friend 5 $LOL` sends 5 MemePoints instantly.
- **Burn & Redeem**:
  Users burn `50 $LOL` to claim a limited-edition NFT or download a premium meme pack.
- **Gating**:
  Hold `≥100 $LOL` to join the "Top Meme Squad" private channel.

Advanced Features & Community Modules

1. NFT Badge Gallery / Profile Integration

- Let users view their earned NFT badges inside a "My Profile" tab
- Encourages collection, achievement, and bragging rights

2. Group Leaderboards

- Weekly or all-time rankings for most active members, most tipped, most helpful
- Drives competition and engagement

3. Admin Dashboard

- Track community token flow, tips given/received, badge redemptions
- Set thresholds for rewards or token-gated access
- Manage user requests (e.g., token minting)

4. Group Analytics Panel

- Visual graphs for token activity, top contributors, tipping velocity, etc.
- Useful for community managers and DAO organizers

5. Mini Games (Gamification Layer)

- Lightweight in-chat games (quizzes, contests, raffles) that reward custom tokens or NFTs
- Adds another layer of engagement and entertainment

. Token-Gated Content Downloads

- Give access to premium files (eBooks, PDFs, videos, tools) when users hold X amount of a token
- Can work alongside Telegram’s new “paid content” feature

7. Marketplace / P2P Exchange

- Allow users to trade custom tokens within the group (future module)
- Could include basic order-matching, or a “Tipbank” that facilitates swaps

8. Burnable Tokens (Reward Conversion)

- Let users burn tokens in exchange for real or digital rewards (e.g., digital art, whitelist spots, discount codes)

9. Referral or Invite System

- Users get rewarded in tokens or NFTs when they refer new members who join and create a wallet

10. Soulbound (Non-Transferable) NFT Badges

- Represent participation, rank, or achievements that can't be transferred
- Could be used for voting weight, OG status, event attendance

Detailed Process Flows

.1 Token Minting (User-Driven)

1. **User Action**: Opens "Create Token" form in Mini App
2. **Inputs**: Token name, symbol, max supply, optional image
3. **Backend**:
   - Validates inputs
   - Calls `createToken()` via Hathor Wallet Library
   - Pays creation fee (user-subsidized or platform-funded)
4. **Result**: New token appears in user's wallet; bot confirms

.2 Sending / Tipping

1. **User Action**: `/tip @Bob 5 $VIBE` or click "Send" in UI
2. **Backend**:
   - Builds transaction with Hathor API (inputs: sender priv key, recipient address, token ID, amount)
   - Broadcasts to network

3. **Confirmation**: Bot replies “You tipped Bob 5 $VIBE. 🎉”

.3 NFT Badge Issuance

1. **Trigger**: Nano contract logic detects milestone (e.g., 10 tips given)
2. **Backend**:
    - Constructs mint NFT transaction
    - Broadcasts to network
3. **Notification**: Bot sends badge image & claim message in chat

.4 Access Control

1. **User Action**: Clicks “Join VIP Chat”
2. **Backend**: Queries user’s token balance
3. **Decision**:
    - If balance ≥ threshold → invite to group or reveal link
    - Else → prompt to earn/purchase tokens

Technical Feasibility

Frontend: React + Telegram Mini App SDK

Backend: Node.js/TypeScript + Hathor Wallet Library

Data layer: PostgreSQL (user profiles, encrypted keys, token metadata)

Blockchain:

- On-chain: token mint/transfer/NFT issuance
- Off-chain: nano-contract logic (leaderboards, thresholds)

Security & Key Management: Encryption, KMS, HTTPS + Telegram auth

🧩Telegram Mini App + Bot Architecture in Detail

- **Frontend**:
  Built using **React + Telegram Mini App SDK**, the Mini App runs inside Telegram's native WebView. This means users don't leave the chat, and everything feels instant and familiar.

- **Bot Integration**:
  A connected Telegram Bot handles message-based commands like `/tip`, `/balance`, or `/create_token`, ensuring full conversational control alongside the Mini App UI.

Backend Logic

- **Node.js + TypeScript Backend**
  The backend handles:
  - Wallet creation (using the official Hathor Wallet Library)
  - Token minting and transfer logic
  - NFT badge minting and metadata management
  - Business logic (e.g., tip history, milestone triggers, gating rules)
- **Nano-Contracts (Off-Chain Rules Engine)**
  All rules—like "first tip → badge," "hold 100 $VIBE → unlock chat"—are enforced off-chain using nano-contract logic written in Node. This allows full control with zero smart contract complexity.

Hathor Integration

- **Hathor Wallet Library (TypeScript)**
  Used server-side to generate user wallets, sign transactions, and interact with the network.

- **HTR Testnet**
  We develop on Hathor's testnet for cost-free testing, then deploy to mainnet when ready.

- **Public API or Self-Hosted Node**
  Transactions are broadcast via either a hosted Hathor node or official API endpoints.

Security & User Wallets

- **Auto Wallet Provisioning**
  When a user opens the app, the backend generates a wallet tied to their Telegram ID. Private keys are encrypted and stored securely.

- **Encryption & Key Storage**
  We use strong server-side encryption (or optionally cloud KMS) to protect user keys. The wallet is **non-custodial in behavior**, even if server-generated.

- **Telegram Native Auth**
  Telegram passes a signed auth payload when the Mini App loads, ensuring secure session handling without passwords.

Data Layer

- **PostgreSQL / Supabase** for:
  - Telegram user ↔ wallet address mapping
  - Token metadata (name, supply, symbol, burnable)
  - Tip history, badge milestones, referral tracking
- **Optional IPFS/Pinata**
  For hosting NFT badge metadata (images, descriptions, links)

👥 Use Cases & User Journeys

HathorChat transforms regular Telegram users into active participants in token-based economies. Here's how it works under the hood — in clear, real-life examples.

🪙 1. Tipping as Social Currency

Use Case: A user tips someone for being helpful in a Telegram group.

Frontend (Mini App or Bot):

- User types `/tip @Alice 5 $VIBE` or uses the “Send Tip” button.
- Frontend collects recipient username, amount, and token type.

Backend:

- Resolves both users' wallet addresses from Telegram IDs via database.
- Decrypts sender's private key securely.
- Constructs and signs transaction using Hathor Wallet Library.

Blockchain:

- Transaction is broadcast to Hathor network via public API or self-hosted node.
- Network confirms token transfer from sender to recipient wallet.

Database:

- Logs the tip (sender, recipient, token, amount, timestamp) for analytics and user history.
- Triggers milestone check: “Was this the first tip?”

🎖️ 2. NFT Badge Reward (Milestone)

Use Case: A user earns a badge after sending their first tip.

Frontend:

- User receives an in-chat bot message:

  “🎉 You earned the *First Tip Sent* NFT badge!”

Backend:

- Nano-contract logic detects milestone triggered from previous tip.
- Calls Hathor's NFT creation API using admin wallet (or badge contract wallet).
- Generates badge metadata (image, title, rarity) and signs mint transaction.

Blockchain:

- NFT is minted and sent directly to the user's wallet address.

Database:

- Logs badge: {user_id, badge_id, timestamp, token_id}
- Updates "My Badges" gallery in the Mini App

💼 3. Creating a Custom Token

> **Use Case:** A user wants to create a token called "CoffeeCoin" ($BREW) with a 10,000 supply.

Frontend:

- User opens "Create Token" form in the Mini App.
- Fills in: name, symbol, supply, and optional logo.

Backend:

- Validates input (no offensive names, supply limits, etc.).
- Uses admin wallet to call Hathor's `createToken()` API.
- Mints full supply and transfers to the user's wallet.

Blockchain:

- Token is minted as a fungible token on the Hathor network.
- Appears in the user's wallet with its own token ID.

Database:

- Stores token metadata: name, symbol, supply, creator_id, logo URL
- Logs mint event under the user's history
- Adds token to available tip/send options in Mini App

🔐 4. Token-Gated Access

Use Case: A user tries to join a VIP Telegram group that requires holding 100 $ALPHA tokens.

Frontend:

- User taps "Join VIP Group" in the Mini App.

Backend:

- Calls `getBalance(user_id)` and checks if balance ≥ required threshold (100 $ALPHA).
- If yes: proceeds to invite flow. If not: shows "Need 100 $ALPHA to join."

Blockchain:

- Token balance is queried from the Hathor network (via explorer or node API).

Database:

- Stores group gating rules: token ID, min balance, access level.
- Logs access attempt for analytics.

Output:

- If approved, bot sends invite link or auto-adds the user to the VIP group.

📊 5. Viewing Wallet & Transaction History

Use Case: A user wants to check how much HTR and custom tokens they hold.

Frontend:

- User taps “My Wallet” → “View Balance & History”

Backend:

- Calls `getWalletBalance(user_id)` and `getTransactionHistory(user_id)`
- Fetches and formats on-chain data

Blockchain:

- Queries real-time balances and transaction history using Hathor’s explorer API.

Database:

- Caches basic wallet info for quick display.
- Merges history with local logs (tips, mints, badges) for richer view.

Output:

- User sees full balance in HTR and all tokens
- Transaction history includes tips, rewards, and mints with timestamps

---

📈 6. Admin Dashboard Usage

> **Use Case:** Admin wants to view top users, tipping activity, and control token minting.

Frontend:

- Admin opens Dashboard Panel inside Mini App
- Views graphs, lists of top users, group-wide token flow

Backend:

- Queries PostgreSQL for:

    - Tip history logs
    - Badge milestones
    - Custom token stats
- Offers token approval UI (for user-submitted token requests)

Database:

- Aggregated analytics, group-specific config, admin settings

---

🎮 7. Gamification (Future Layer)

Use Case: Run a weekly challenge—“Top tipper wins a badge.”

Frontend:

- Weekly leaderboard shown in Mini App
- “Challenge of the Week” card with prize preview

Backend:

- Nano-contract logic counts user tips over 7-day window
- Winner automatically mints “Top Tipper” NFT on Sunday night

Blockchain:

- NFT badge minted on Hathor and sent to winning wallet

Database:

- Logs all weekly winners and badges
- Updates leaderboard cache

🧠 Summary Table

Action	Frontend	Backend	Blockchain	Database
Tip a user	Form or command	Build & sign tx	Transfer HTR/token	Log tx, trigger badge
Earn badge	Auto-prompted UI	Nano-contract check → mint NFT	Mint NFT	Badge metadata stored
Create token	Token form	Validate → call `createToken()`	Mint custom token	Store token info
Gate access	Join button	Check balance threshold	Query wallet	Access rules logged
View wallet	Tap UI button	Call getBalance	Fetch balances	Merge on-chain + local
Admin tools	Dash UI	Analytics queries	N/A	Group logs, settings