Enjin Governance

MANIFEST · NOVEMBER 2026

Von der Chain lesen. Jeden Text belegen. Ehrlich anzeigen.



Enjin Governance ist eine Weboberfläche für die On-Chain-Governance von Enjin: Referenden lesen, abstimmen, Anträge einreichen, der Fellowship folgen. Dieses Manifest sagt, woran das Projekt glaubt, wie es gebaut ist und was es heute kann. Wo etwas fehlt, steht es dabei.

INHALT

01	Was wir glauben	2	08	Wie es aufgebaut ist	14
02	Das Problem	5	09	Was es heute kann	15
03	Die Idee	6	10	Im Einsatz	16
04	Die Chain lesen	7	11	Moderation	18
05	Der EGOV1-Eintrag	9	12	Sicherheit und Datenschutz	19
06	Abstimmen und einreichen	11	13	Wie es gebaut wird	20
07	Die Fellowship	13	14	Grenzen heute	22

Chris Zemmel (Zyric)

01

Was wir glauben

Über öffentliches Geld und über Änderungen an einer Blockchain entscheiden Menschen, die einen Text lesen und dann signieren. Eine Oberfläche dafür ist nur so gut wie die Regeln, nach denen sie gebaut ist. Enjin Governance folgt acht Grundsätzen. Jeder ist im Code umgesetzt, und hinter jedem steht ein echter Fall.

19. Juni 2026 erster Commit im Repository, Version 1.0 am 28. Juni	**180+** Commits auf der Hauptlinie	**15** zusammengeführte Pull Requests
1.200+ automatische Tests, dazu Browsertests	**seit 2026** öffentlich unter gov.enjin.cloud, EGOV1-Einträge dort ab August 2026	**10+** Referenden auf der Enjin Relaychain mit EGOV1-Eintrag dieser App

1 Die Chain ist die Quelle der Wahrheit

Was auf der Chain steht, liest der Browser direkt von ihr. Datenbank, Dateispeicher und Block-Explorer sind Zwischenspeicher oder Ergänzung, nie die Quelle.

WARUM

Ein Indexer kann hinterherhinken, ausfallen oder sich irren. Wer direkt vom Knoten liest, zeigt, was gilt, und braucht keinen Dienst, der zwischen Wählern und Chain steht. Dafür ist kein Indexer nötig.

DER FALL

Ein entschiedenes Referendum trägt auf der Chain seinen Track nicht mehr. Die Treasury-Seite zählte deshalb 0 angenommene und 0 abgelehnte Anträge. Heute liest der Server die Tracks entschiedener Referenden aus dem Archivknoten und hält sie vor. Scheitert auch der Browser daran, steht „-“ da, keine Null.

2 Die Regeln kommen aus der Runtime

Tracks, Kautionen, Ausgabegrenzen, Sperrfristen und die Messung der Unterstützung werden von der Chain gelesen und an die Version der Runtime gebunden.

WARUM

Die Regeln ändern sich mit jedem Runtime-Upgrade. Fest eingetragene Werte stimmen am Tag danach nicht mehr, und niemand merkt es. Deshalb übernimmt sogar ein Tab, der über ein Upgrade hinweg offen bleibt, die neuen Werte von selbst.

DER FALL

Die kleinste Treasury-Stufe wurde bis 250 ENJ angeboten, das Mainnet erlaubte damals 100 ENJ. Ein solcher Antrag konnte erst bei der Ausführung scheitern, nach der ganzen Abstimmung. Heute stehen die Grenzen je Runtime-Version in einer Tabelle, und eine unbekannte Version bekommt eine Warnung.

3 Jeder Text lässt sich nachprüfen

Jeder hier eingereichte Antrag nagelt seinen Text mit einem Hash auf der Chain fest. Jeder kann prüfen, ob der Text seitdem getauscht wurde, auch ohne diese App.

WARUM

Wähler entscheiden über einen Aufruf, aber sie lesen einen Text. Liegt der Text nur auf einem Server, kann er sich nachträglich ändern, und niemand merkt es. Eine Änderung soll erlaubt sein, aber sichtbar.

DER FALL

Referendum #19 auf der Enjin Relaychain wurde am 4. November vier Minuten nach dem Einreichen vom Antragsteller bearbeitet. Der Hash auf der Chain passt seitdem nicht mehr zum Text, und genau das zeigt die Seite: „EGOV1 · Edited“.

4 Ehrlicher Status

Angezeigt wird, was die Chain belegt. Was geschätzt ist, trägt „≈“. Was nicht gelesen werden konnte, sagt das, statt eine Null oder eine leere Liste zu zeigen.

WARUM

Angenommen heißt nicht ausgeführt, und knapp heißt nicht verloren. Wer abstimmt oder Geld erwartet, muss den echten Stand sehen. Eine falsche Warnung ist so schädlich wie eine fehlende.

DER FALL

Mainnet-Referendum #12 wurde einen Tag nach der Annahme ausgeführt. Heute bleibt der Schritt „Ausführen“ offen, bis der Scheduler den Aufruf wirklich ausgeführt hat, mit Block und Ergebnis. Und die Warnung „zu wenig Unterstützung“ erscheint nur noch, wenn die Stimmen von heute beide Kurven nicht mehr rechtzeitig erreichen.

Belegt

Von der Chain gelesen, mit Block. Was angezeigt wird, ist der Stand.

Geschätzt

Ein künftiger Zeitpunkt, aus den Regeln der Runtime berechnet, markiert mit „≈“.

Nicht lesbar

Die Quelle antwortete nicht. Angezeigt wird „-“ oder der Grund, nie „0“.

5 Keine Signatur ohne Ansage

Vor jeder Signatur steht, was sie reserviert und kostet. Nichts wird doppelt eingereicht, und ein Knopf erscheint nur, wenn die Runtime die Aktion auch annimmt.

WARUM

Eine Signatur bindet echtes Geld: Einreichungskaution, Preimage-Kautionen, Gebühren. Wer das erst nachher erfährt, verliert Vertrauen und manchmal Geld. Fehlermeldungen der Runtime werden deshalb in lesbare Sätze übersetzt.

DER FALL

Landete ein Antrag, aber seine Statusmeldungen gingen verloren, findet „Erneut versuchen“ heute das schon eingereichte Referendum, statt ein zweites mit zweiter Kaution einzureichen. Und „Erstatten“ für eine Einreichungskaution erscheint nur nach Annahme oder Abbruch, weil die Runtime es sonst mit `BadStatus` ablehnt.

6 Moderation hält den Raum sauber, nie die Chain

Moderatoren entscheiden, was die App zeigt. Daten auf der Chain und das festgenagelte JSON rühren sie nie an, und jede Entscheidung steht mit Grund im öffentlichen Protokoll.

WARUM

Eine Oberfläche, die still Inhalte entfernt, wird selbst zum Machtfaktor in der Governance. Moderation muss deshalb begründet, öffentlich und auf die eigene Anzeige begrenzt sein.

DER FALL

Ein verborgener Antrag war über zwei API-Routen noch mit Titel und Text abrufbar. Heute lassen beide ihn weg und sagen, warum. Das JSON mit dem Hash auf der Chain bleibt dagegen bewusst öffentlich: Sonst ließe sich der Text nicht mehr prüfen.

7 Der Beweis gegen die Wirklichkeit

Browsertests laufen gegen den Produktions-Build mit echten Daten der Canary-Chain. Die Regeln der Runtime wurden laut Änderungsprotokoll auf Kopien der echten Chains erprobt.

WARUM

Nachbildungen kennen nur die Fehler, die man ihnen beigebracht hat. Die Fehler, auf die es ankommt, entstehen an der Grenze zur echten Chain: Rundung, Versionen, verschwundene Daten.

DER FALL

Auf dem Mainnet zeigte die App rund 28 Prozent zu wenig Unterstützung. Sie teilte durch die gesamte Ausgabe von ENJ, die damalige Runtime durch die aktive. Heute wählt die App den Nenner nach der Version der Runtime (Kapitel 4).

8 Offen und übergebbar

Der Code steht unter der AGPL-3.0. Jeder kann eine eigene Instanz betreiben, mit eigenen Konten und ohne Zugangsdaten des Betreibers.

WARUM

Eine Oberfläche für öffentliche Entscheidungen darf nicht an einer Person hängen. Der verbindliche Datensatz, also Referenden, Stimmen, Sperren und jeder EGOV1-Umschlag, liegt ohnehin auf der Chain.

DER FALL

Das Übergabedokument im Repository beschreibt den Fall, dass der Betreuer aufhört: eine neue Instanz aus dem Quellcode, eine leere Datenbank genügt. Mit einer Kopie des Dateispeichers lösen alte EGOV1-Links weiter auf, ohne sie bleibt der Hash auf der Chain als Beleg.

02

Das Problem

Die Enjin Relaychain entscheidet über ihre Treasury und über Änderungen an sich selbst mit OpenGov: Referenden auf Tracks, Stimmen mit Conviction, Kautionen und Preimages. Das ist mächtig und schwer zu lesen.

Ein Referendum ist auf der Chain ein Eintrag mit einem Hash und einem Aufruf in kodierten Bytes. Wer wissen will, worüber abgestimmt wird, braucht den Aufruf entschlüsselt, den Text dazu und die Regeln des Tracks. Nach der Entscheidung verschwinden Teile davon wieder aus dem aktuellen Zustand: die Auszählung, der Aufruf, der Block der Einreichung. Was bleibt, findet nur, wer einen Archivknoten befragt.

Der Text eines Antrags liegt dagegen irgendwo außerhalb der Chain. Ob er heute noch derselbe ist wie am Tag der Abstimmung, lässt sich meist nicht prüfen. Und die Regeln selbst wandern: Ausgabegrenzen, Kautionen und die Art, wie Unterstützung gemessen wird, ändern sich mit Runtime-Upgrades.

Bytes statt Sätze

Ein Aufruf in kodierter Form sagt Wählern nichts. Wer ihn nicht entschlüsselt, stimmt über ein Etikett ab.

Text ohne Beleg

Liegt der Text nur auf einem Server, kann er sich ändern. Niemand sieht, ob er noch der ist, über den abgestimmt wurde.

Regeln, die wandern

Grenzen, Kautionen und Messungen hängen an der Runtime. Eine Oberfläche mit festen Werten zeigt nach dem Upgrade Falsches.

Signieren unterwegs

Viele Wähler signieren am Telefon. Eine Anfrage, die nie in der Wallet ankommt, sieht aus wie eine Wallet, die nicht reagiert.

Geld mit Fristen

Kautionen, Sperren und Auszahlungen folgen eigenen Fristen. Wer sie nicht kennt, verliert Geld oder wartet vergeblich.

Ein offener Raum

Wo jeder schreiben darf, landen auch Betrugsversuche. Wer moderiert, darf dabei nicht still in die Governance eingreifen.

03

Die Idee

Enjin Governance beruht auf zwei Gedanken. Erstens: Was auf der Chain steht, liest der Browser direkt von der Chain. Zweitens: Was nicht auf die Chain passt, der Text eines Antrags, wird mit einem Hash auf der Chain festgenagelt.

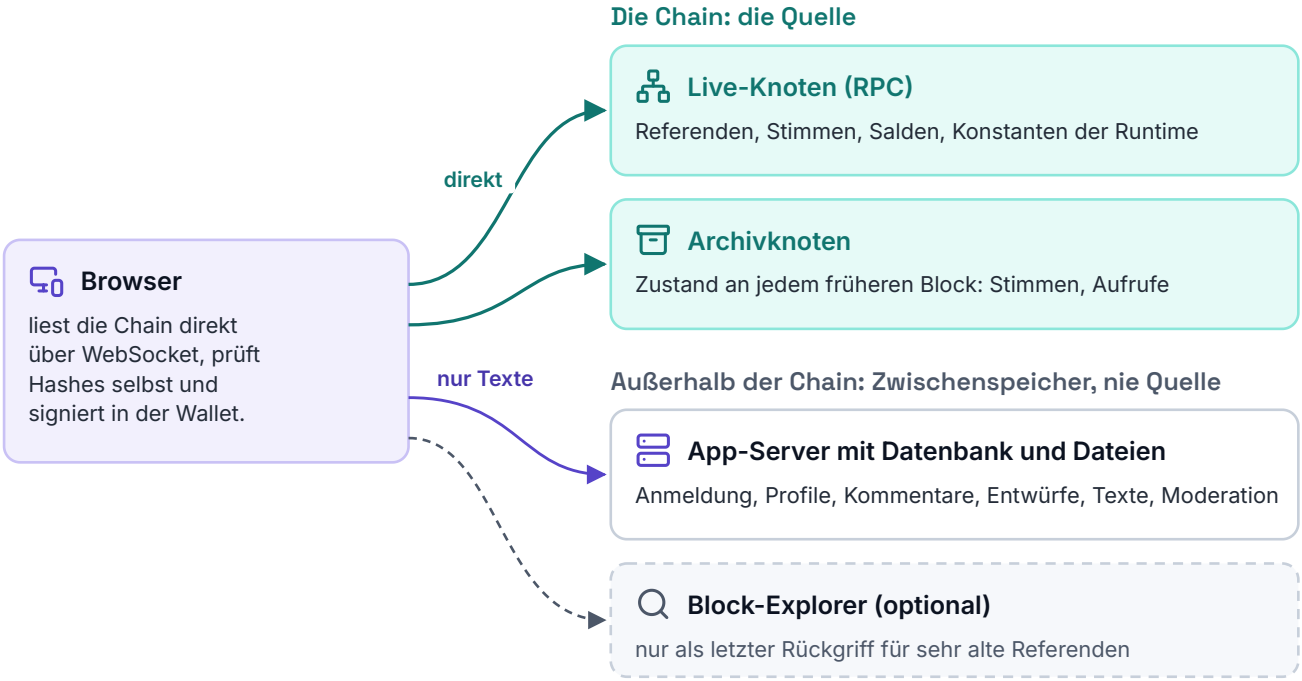


Abb. 1 Woher die Daten kommen. Der Browser liest Referenden, Stimmen und Regeln direkt vom Live-Knoten und vom Archivknoten. Der App-Server hält nur, was nicht auf die Chain gehört, und ist nie die Quelle für den Zustand der Chain. Ein Block-Explorer ist ein optionaler letzter Rückgriff.

Der Browser verbindet sich über WebSocket mit einem Knoten der Enjin Relaychain und, für vergangene Zustände, mit einem Archivknoten. Daraus entstehen Liste, Detailseite, Abstimmung und Treasury. Der App-Server liest die Chain nur, bevor er selbst handelt: etwa wenn er prüft, ob ein Entwurf wirklich zu einem Referendum gehört.

Daneben gibt es, was die Chain nicht kennt: die Anmeldung per Signatur, Profile, Kommentare, Entwürfe, hochgeladene Bilder und PDFs, die Moderation. Dafür hat die App eine Postgres-Datenbank und einen S3-kompatiblen Dateispeicher. Fallen beide aus, bleiben Lesen und Abstimmen möglich. Ohne jede Einstellung startet die App mit öffentlichen Knoten und Wallets aus Browser-Erweiterungen.

Das Netz lässt sich in der App umschalten, ohne neues Deployment: Enjin Relaychain oder Canary Relaychain, das Testnetz. Ein Link mit `?network=` öffnet auf dem Netz, das er nennt. Eine Wallet-Sitzung gilt immer nur für ein Netz.

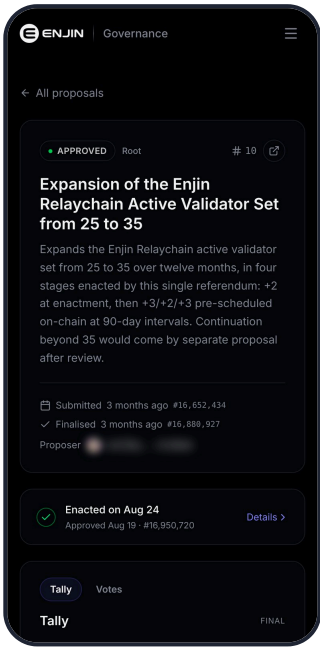
04

Die Chain lesen

Ein laufendes Referendum ist leicht zu lesen. Schwer wird es danach: Die Runtime löscht Auszählung, Aufruf und Einreichungsblock aus dem aktuellen Zustand. Enjin Governance holt sie aus dem Archiv zurück.

Für ein entschiedenes Referendum liest die App den Zustand an dem Block, bevor es entschieden wurde. So bleiben auch Wähler sichtbar, die ihre Stimme später entfernt haben. Den Aufruf entschlüsselt sie aus dem Preimage-Speicher, live oder, wenn die Runtime das Preimage nach der Ausführung gelöscht hat, aus dem Archiv. Kleine Aufrufe stehen direkt im Referendum und werden mit der Runtime des Blocks entschlüsselt, an dem sie gelesen wurden, denn ein späteres Upgrade kann Aufrufe neu nummerieren.

Jede Stimme kommt aus dem Speicher der Abstimmung auf der Chain, mit ihrer Währung: ENJ oder sENJ aus einem Staking-Pool. Ein Diagramm zeigt jede Stimme als Abschnitt im Verhältnis zu ihrem Gewicht, dazu die Verteilung über die Conviction-Stufen: ein großer Halter oder viele kleine, auf einen Blick.



Ein Referendum auf dem Mainnet

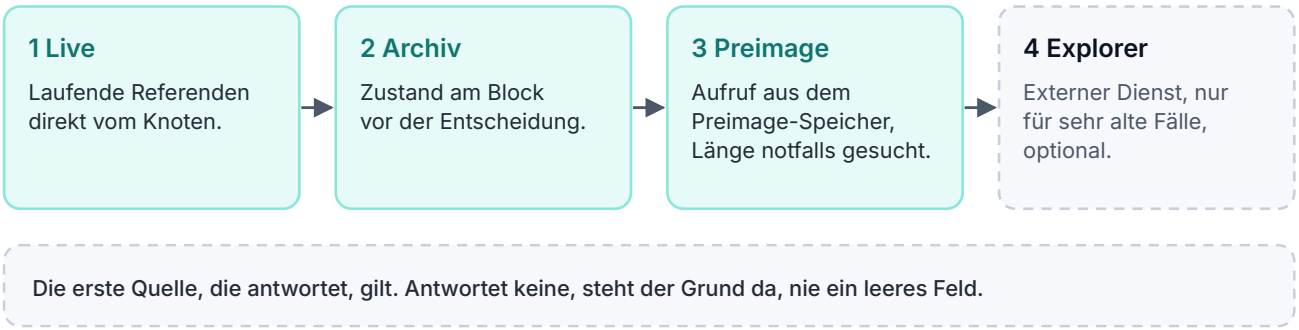


Abb. 2 Die Reihenfolge der Quellen für den Aufruf eines Referendums. Ein Explorer kommt nur zum Zug, wenn Chain und Archiv nichts liefern. Auf Canary arbeitet die App ganz ohne ihn.

Der Lebenslauf

Jedes Referendum durchläuft Vorbereitung, Entscheidung, Bestätigung und Ausführung, Treasury-Ausgaben zusätzlich die Auszahlung. Die Karte auf der Detailseite nennt für jeden Schritt ein Datum, solange etwas aussteht zuerst die verbleibende Zeit. Ist alles geschehen, faltet sie sich zu einer Zeile wie „Ausgezahlt“ oder „Ausgeführt am“, mit jedem Ereignis und Block dahinter.

1	**Vorbereiten**	Fehlt die Entscheidungskaution, sagt die Karte, wie viel nötig ist. `referendumInfoFor`
2	**Entscheiden**	Ist der Track voll, steht da „In der Warteschlange“, kein Countdown. `track.maxDeciding`
3	**Bestätigen**	Beginnt dort, wo der Alarm der Runtime es sagt, nicht geschätzt. `alarm`
4	**Ausführen**	Bleibt offen, bis der Scheduler den Aufruf ausgeführt hat. `scheduler.lookup`
5	**Auszahlen**	Nur bei Treasury-Ausgaben: nächste Periode, dann bezahlt, mit Block. `treasury.proposals`

Geschätzte Daten tragen „≈“. Was geschehen ist, kommt mit Block und Ergebnis aus dem Archivknoten.

Abb. 3 Fünf Schritte, jeder aus einer eigenen Quelle auf der Chain. Was noch aussteht, ist als Schätzung markiert. Was geschehen ist, steht mit Block und Ergebnis da, ein gescheiterter Aufruf als gescheitert.

Wie viel Unterstützung reicht

Ein Referendum braucht Zustimmung und Unterstützung, beide über Kurven, die mit der Zeit fallen. Die Unterstützung misst die Runtime gegen die Ausgabe von ENJ, und welche Ausgabe gilt, hat sich zwischen zwei Versionen geändert: bis Version 1070 die aktive Ausgabe (gesamt minus inaktiv), ab 1080 die gesamte. Die App wählt den Nenner nach der Version, mit der sie verbunden ist, und die Beschriftung des Diagramms sagt, welcher gilt.

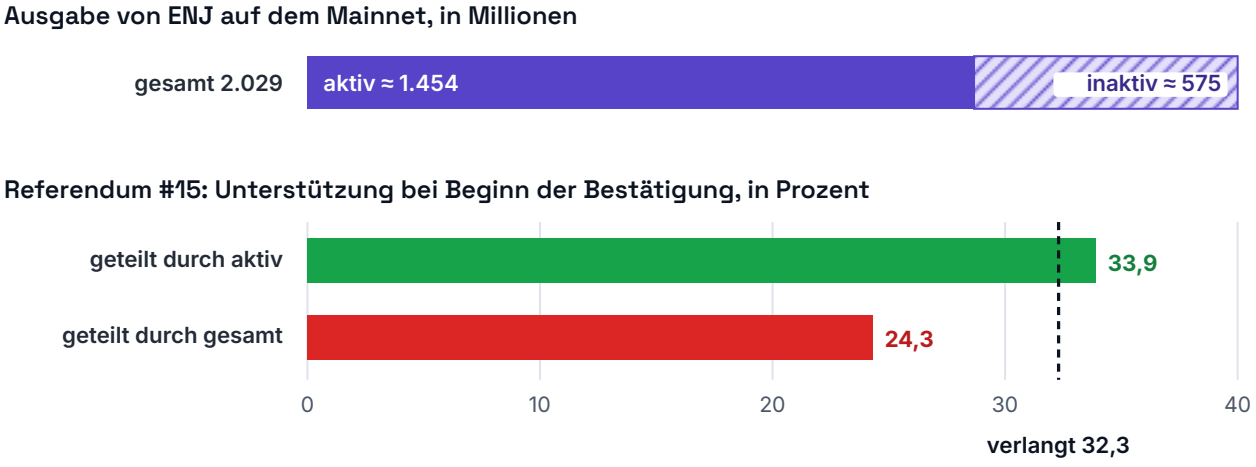


Abb. 4 Warum der Nenner zählt. Auf dem Mainnet war rund ein Viertel der Ausgabe inaktiv. Referendum #15 begann die Bestätigung mit 33,9 Prozent der aktiven Ausgabe gegen verlangte 32,3 Prozent. Gegen die gesamte Ausgabe gerechnet, zeigte die App 24,3 Prozent, ein Referendum auf Kurs als zu schwach. Werte aus der Projektdokumentation.

Auf dieser Rechnung baut die Einschätzung auf der Detailseite auf. Sie rechnet mit der Rundung der Runtime aus, ob und wann die Stimmen von heute beide Kurven vor dem Ende der Entscheidungszeit erreichen. „Auf Kurs“ sagt, wann das Referendum bei gleichbleibenden Stimmen durchkäme. Die gelbe Warnung bleibt Referenden, die es nicht schaffen können, mit dem, was fehlt, und der Frist.

05

Der EGOV1-Eintrag

EGOV1 ist ein kleines, offenes Format: Es bindet den Text eines Antrags an sein Referendum, so dass jeder ihn ohne diese App finden und prüfen kann.

Der Text eines Antrags liegt als JSON-Datei im Dateispeicher der App, ausgeliefert über ihre eigene Adresse. Auf die Chain kommt nur ein kurzer Umschlag: `EGOV1:{"u":"…","h":"…"}`, mit der Adresse `u` des JSON und seinem SHA-256 `h`. Der Hash wird über eine feste Form gebildet, Schlüssel auf jeder Ebene sortiert, damit jedes Programm denselben Wert errechnet.

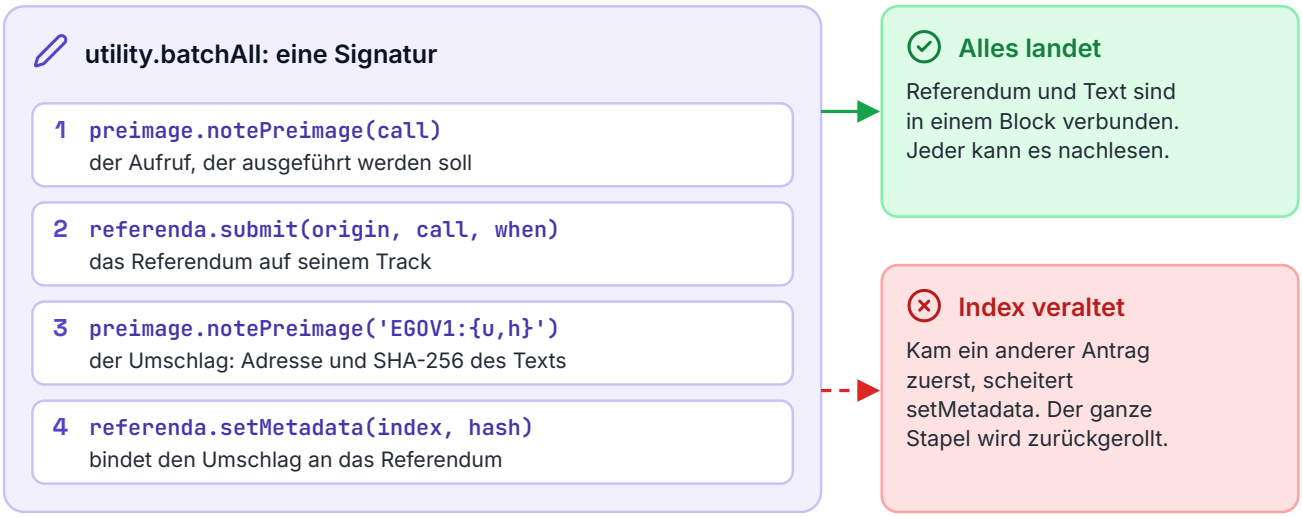


Abb. 5 Ein Antrag, eine Signatur. Der Aufruf, das Referendum, der Umschlag und der Verweis darauf gehen in einem Stapel auf die Chain. Ist der vorausberechnete Index veraltet, lehnt die Runtime den Verweis ab, und der ganze Stapel wird zurückgerollt. So kann ein veralteter Index nie das Referendum eines anderen beschriften.

Die Runtime erlaubt den Verweis über `setMetadata` nur dem, der das Referendum eingereicht hat, und nur solange es läuft. Kommt ein anderer Antrag zuvor, scheitert der Stapel als Ganzes. Die App baut ihn dann neu und versucht es noch einmal. Ältere Anträge aus der Zeit vor diesem Verweis tragen denselben Umschlag in einem `system.remark` innerhalb ihres Stapels.

Wie jeder prüfen kann

Wer einen Text prüfen will, liest den Metadaten-Hash des Referendums, löst den Hash über den Preimage-Speicher zum Umschlag auf, lädt das JSON von `u` und vergleicht seinen SHA-256 mit `h`. Das geht mit jedem Knoten, in jedem Browser, aus jedem Skript. Ein Indexer, ein Explorer oder ein anderer Client kann so den ganzen Bestand an Antragstexten aufbauen.

Die App zeigt den Stand als Abzeichen am Text. Ändert ein Antragsteller den Text eines laufenden Referendums, wird das JSON an derselben Adresse neu geschrieben. Sein Hash passt dann nicht mehr zum Umschlag auf der Chain. Genau diese Abweichung ist das öffentliche Zeichen einer Bearbeitung. Aufruf, Betrag und Empfänger lassen sich nicht ändern.

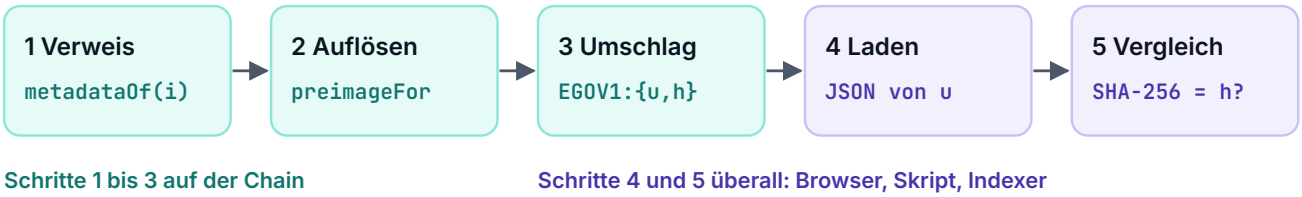


Abb. 6 Fünf Schritte vom Referendum zum geprüften Text. Die ersten drei liest man von der Chain, die letzten beiden kann jeder selbst ausführen.

Verified
Der geladene Text hat den Hash, den die App beim Verknüpfen gegen die Chain geprüft hat.

Edited
Der Antragsteller hat nach dem Einreichen geändert. Die Abweichung zur Chain ist gewollt sichtbar.

Unverified
Der Text passt nicht und keine Bearbeitung ist verzeichnet. Er wird gezeigt, aber als nicht belegt.

```
{
  "schema": "enjin-governance-proposal", "version": "1.2.0",
  "network": "enjin-relay", "proposer": "en…", "track": "Root",
  "title": "…", "summary": "…", "body_markdown": "…",
  "attachments": [ { "name", "size", "content_type", "sha256" } ],
  "call": { "section": "system", "method": "authorizeUpgrade", "code_hash": "0x…" },
  "enactment": { … wann der Aufruf läuft … }, "preimage_hash": "0x…"
}
```

Abb. 7 Ein EGOV1-JSON, gekürzt. Version 1.1.0 beschreibt Treasury-Anträge. 1.2.0 fügt die Abschnitte `call` und `enactment` hinzu, damit auch Upgrades, Abbrüche und andere Aufrufe neben ihrem Text dokumentiert sind. Leser von 1.1.0 funktionieren weiter. Die Größe, der Typ und der Hash jedes Anhangs werden beim Ablegen mit der gespeicherten Datei verglichen.

06

Abstimmen und einreichen

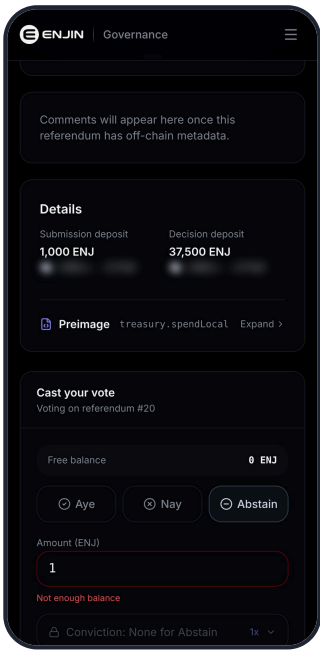
Lesen geht ohne Wallet. Wer abstimmen oder einreichen will, verbindet eine Wallet und signiert. Jede Signatur läuft über denselben Dialog, und vor jeder steht, was sie kostet und bindet.

Abstimmen

Gestimmt wird mit Aye, Nay oder Abstain. Conviction multipliziert das Gewicht einer Stimme gegen eine Sperre: ohne Sperre zählt sie ein Zehntel, mit wachsender Sperre einfach bis sechsfach. Die Dauer der Sperre liest die App aus der Runtime. Abstain zählt zur Unterstützung, nicht zur Zustimmung, und sperrt nichts über das Referendum hinaus.

Gestimmt wird mit ENJ oder mit sENJ aus einem Staking-Pool. Die Chain zählt sENJ eins zu eins wie ENJ, und das Abstimmungsfeld sagt das, ebenso, dass gesperrtes sENJ weiter Belohnungen sammelt, aber bis zum Ende der Sperre nicht abgezogen werden kann. Das Stimmgewicht steht in ENJ da, mit der Rechnung darunter, damit eine sechsfache Stimme nicht wie das Sechsfache des Guthabens aussieht.

Stimmen lassen sich je Track oder für alle Tracks auf einmal delegieren, mit einer Signatur. Auf der Kontoseite gibt die App abgelaufene Sperren frei und erstattet Kautionen, aber nur die, die die Runtime auch erstattet.



Abstain, Test-Wallet ohne Guthaben

Sechs Wallets, eine Signatur-Oberfläche

Unterstützt werden die Enjin Wallet über WalletConnect, WalletConnect allgemein und vier Browser-Erweiterungen für Polkadot-Wallets. Hat eine Wallet mehrere Konten, fragt die App, welches gelten soll, statt still das erste zu nehmen.

Am Telefon war der Weg in die Wallet die schwierigste Stelle. iOS hält eine Seite im Hintergrund an, auch mitten im Senden. Der Knopf „In Enjin Wallet öffnen“ öffnet die Wallet deshalb erst, wenn das Relais die Anfrage bestätigt hat, und ein zweiter Tipp sendet dieselbe Anfrage erneut, nie eine zweite. Kehrt man zur Seite zurück, ersetzt sie die Verbindung zum Relais, die iOS im Hintergrund sterben lässt.

Einreichen

Treasury-Anträge entstehen in einem Assistenten in drei Schritten. Er wählt die kleinste Stufe, deren Grenze den Betrag deckt, warnt, wenn die Treasury weniger hält als verlangt, und prüft, ob eine Empfängeradresse aus dem richtigen Netz stammt. Eine Adresse aus einem anderen Netz wird erkannt und erst nach Rückfrage umgewandelt. Ein zweites Formular reicht jeden anderen Aufruf ein: ein Referendum abbrechen, einen Aufruf auf die Whitelist setzen, ein Runtime-Upgrade per Code-Hash freigeben, einen beliebigen kodierten Aufruf. Der Code-Hash eines Upgrades wird im Browser aus der Datei errechnet.

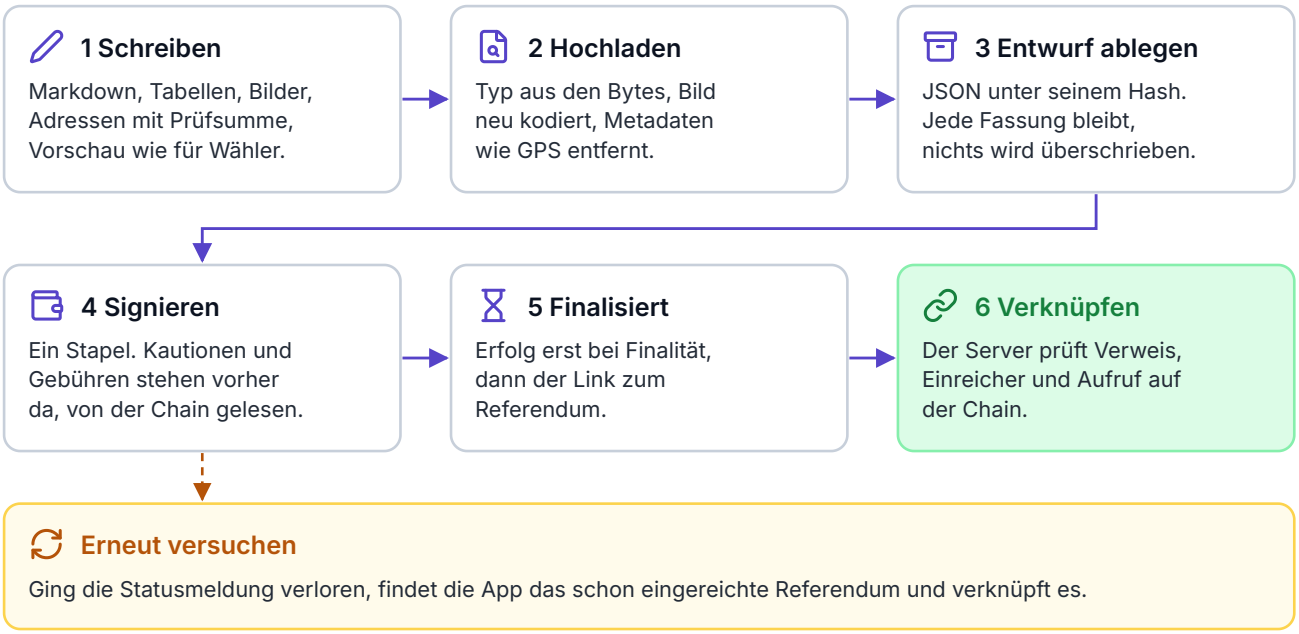
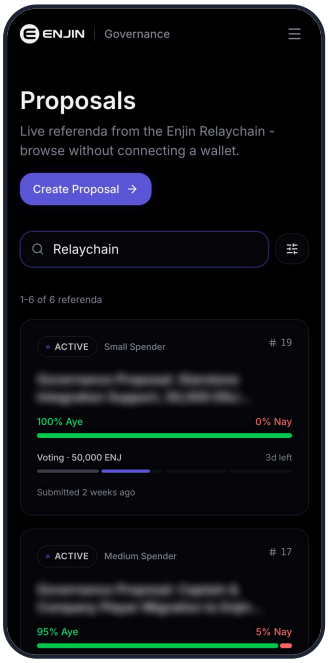
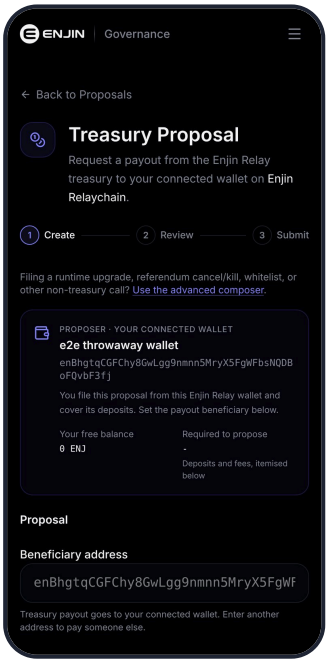


Abb. 8 Der Weg eines Antrags. Entwürfe liegen im Speicher, bevor jemand signiert, und lassen sich fortsetzen. Jede abgelegte Fassung bleibt unter ihrem eigenen Hash, damit ein Stapel, der aus einer älteren Fassung signiert wurde, genau auf die Bytes zeigt, die er festnagelt. Erst die Prüfung auf der Chain verknüpft Entwurf und Referendum.

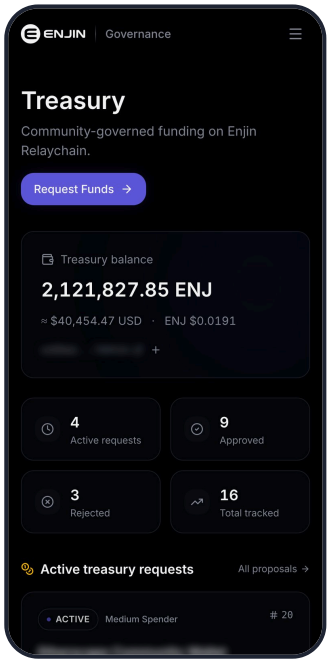
Vor dem Signieren listet die App, was der Antrag reserviert: die Einreichungskaution, die Kautionen für die Preimages von Aufruf und EGOV1-Eintrag, die Gebühren und das Mindestguthaben, alles von der Chain gelesen. Weiter geht es nur, wenn das freie Guthaben reicht. Die Entscheidungskaution des Tracks steht dabei, wird aber nicht verlangt: Sie kann später jeder hinterlegen.



Referenden, ohne Wallet lesbar



Treasury-Antrag, Schritt 1



Treasury

Am Telefon: Referenden ohne Wallet lesen und suchen, einen Treasury-Antrag beginnen, die Treasury mit ihren laufenden Anfragen. Bildschirmfotos der echten App auf der Enjin Relaychain. Links und rechts die öffentliche Instanz gov.enjin.cloud, die Mitte ein lokaler Produktions-Build mit einer Test-Wallet ohne Guthaben, die nie signiert. Adressen und Titel fremder Anträge unkenntlich gemacht.

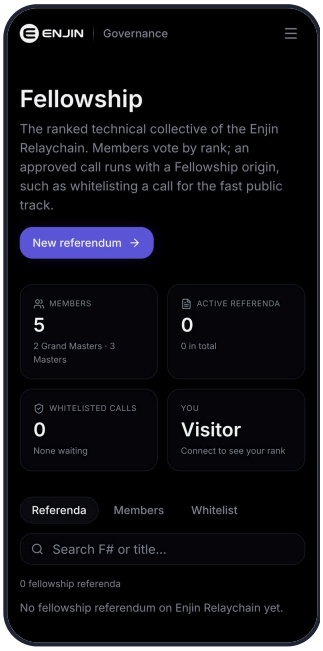
07

Die Fellowship

Die Fellowship ist das gerankte Gremium der Relaychain. Ihre Mitglieder stimmen nach Rang gewichtet ab und können Aufrufe auf eine Whitelist setzen, die dann über einen schnelleren öffentlichen Track laufen.

Die Seite `/fellowship` zeigt Mitglieder und Ränge, jedes Fellowship-Referendum und die Aufrufe auf der Whitelist. Sie erscheint nur, wo die Runtime die nötigen Pallets hat. Fellowship-Referenden werden getrennt gezählt und heißen F#N, damit #7 und F#7 nie verwechselt werden. Auch in der Datenbank ist ein Index nur je Netz und Pallet eindeutig.

Ein Mitglied vom Rang r zählt auf einem Track mit Mindestrang m mit dem Gewicht (e+1)(e+2)/2, wobei e = r - m. Zustimmung wird nach Gewicht gemessen, Unterstützung nach Köpfen, die mit Aye stimmen. Wer unter dem Mindestrang liegt, erfährt, welcher Rang nötig wäre.



Die Fellowship

Rang über dem Minimum (e)	Gewicht
0	1
1	3
2	6
3	10

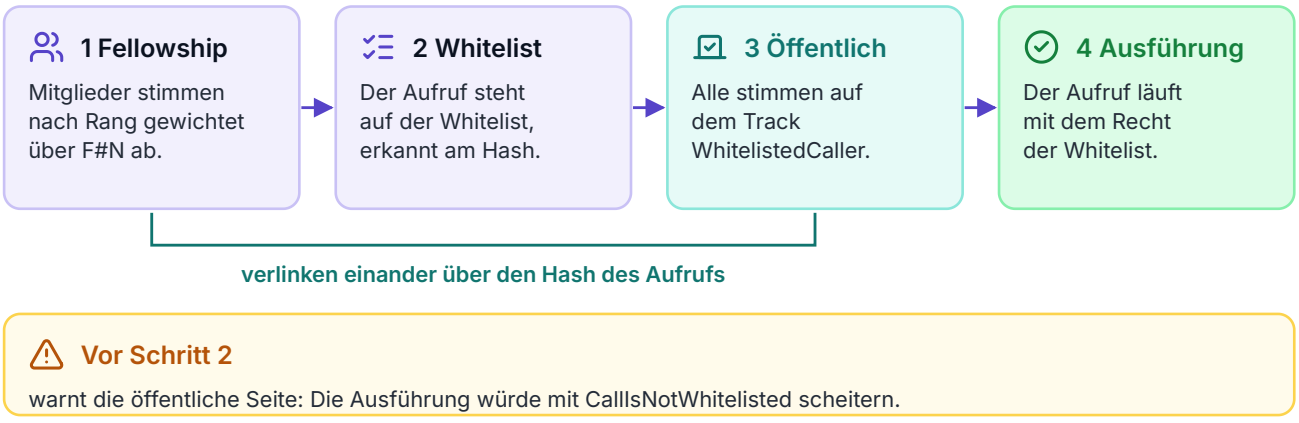


Abb. 9 Der schnelle Weg. Ein Fellowship-Referendum setzt einen Aufruf auf die Whitelist, ein öffentliches Referendum auf dem Track WhitelistedCaller führt ihn aus. Beide Seiten zeigen dieselben vier Schritte, verweisen über den Hash des Aufrufs aufeinander und vergleichen die Bedingungen beider Tracks mit denen von Root.

Mitglieder ab Rang I reichen Fellowship-Referenden im selben Formular ein wie alle anderen Anträge, mit Titel, Zusammenfassung, Text und EGOV1-Eintrag: einen Aufruf auf die Whitelist setzen oder von ihr nehmen, ein Mitglied aufnehmen, befördern, zurückstufen oder entfernen. Die Herkunft, die eine Änderung braucht, steht unter der Auswahl und wird gelb, wenn eine andere gewählt ist.

08

Wie es aufgebaut ist

Eine Next.js-Anwendung in TypeScript, mit wenigen flachen Schichten und klaren Zuständigkeiten. Neue Schichten entstehen nur, wenn es nicht anders geht.

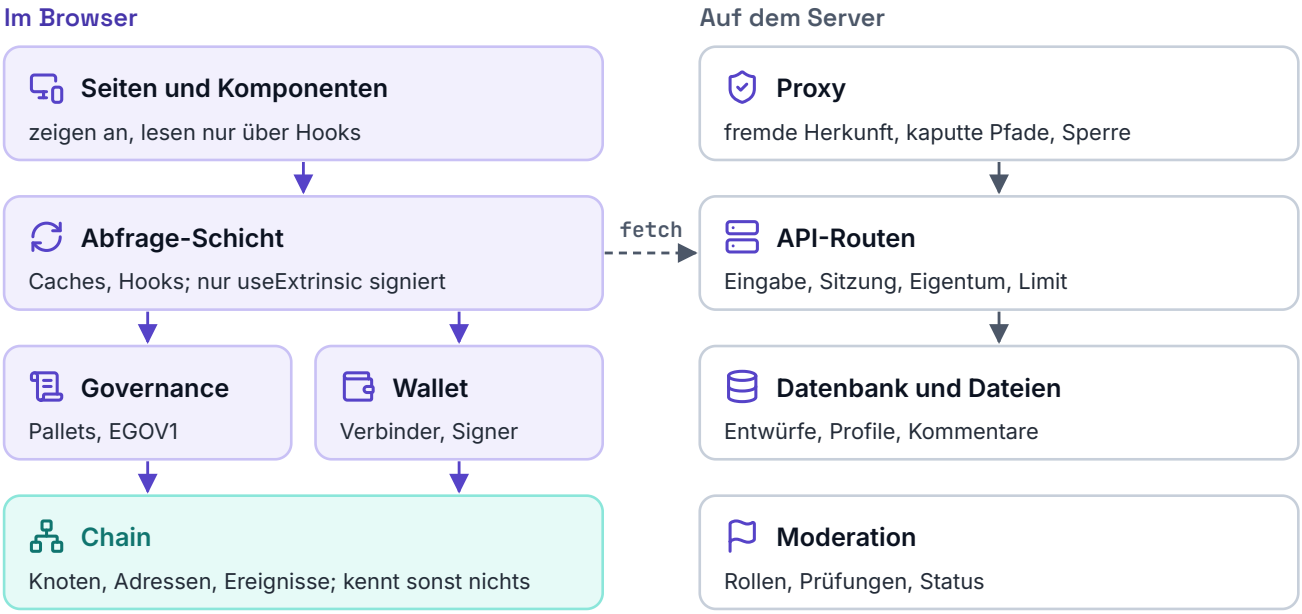


Abb. 10 Links der Browser, rechts der Server. Die Chain-Schicht ganz unten weiß nichts von Governance oder Speicher. Seiten lesen nur über Hooks. Signiert wird an genau einer Stelle. Der Server spricht mit der Chain nur, um vor eigenen Handlungen zu prüfen.

Regeln im Aufbau

- **Nur eine Stelle signiert.** Komponenten übergeben einen fertig gebauten Aufruf. Signieren, Senden und Verfolgen bis zur Finalität leben in einem einzigen Hook.
- **Server-Code bleibt auf dem Server.** Module mit Zugriff auf Datenbank, Speicher oder ein Secret sind markiert. Sie in den Browser zu importieren, bricht den Build.
- **Eine Quelle je Wahrheit.** Die Netze stehen in einer Datei, die Umgebungsvariablen in einem typisierten Schema, das Upload-Limit an einer Stelle für Browser und Server.

Bestandteile

- Next.js 16 mit App Router, React 19, TypeScript im strikten Modus
- polkadot.js für die Chain, hinter einer eigenen Schicht
- WalletConnect und die Schnittstelle der Browser-Erweiterungen
- Postgres und ein S3-kompatibler Dateispeicher außerhalb der Chain
- Vitest, PGlite und Playwright für Tests auf drei Ebenen
- Rund 340 eigene Quelldateien, sieben Dokumente zu Architektur, Abläufen und Betrieb

Der Server prüft jede schreibende Anfrage in fester Folge: gleiche Herkunft, Form der Eingabe, angemeldeter Nutzer, Eigentum über den öffentlichen Schlüssel, eine Schreibsperre der Moderation und ein Ratenlimit. Fehlt eine Einstellung, antworten die betroffenen Routen mit einem Fehler, statt halb zu arbeiten. Eine Statusseite für Admins zeigt nach jedem Deployment, was eingerichtet ist und was fehlt.

09

Was es heute kann

Jede Fähigkeit steht hier mit ihrem Stand. in Betrieb heißt: auf der Hauptlinie ausgeliefert und in der öffentlichen Instanz unter gov.enjin.cloud nutzbar.

Bereich	Was es tut	Stand
Referenden lesen	Liste, Suche und Filter, Detailseite mit Aufruf, Auszählung, Wählern, Lebenslauf und Einschätzung, alles von der Chain, auch für entschiedene Referenden.	in Betrieb
Abstimmen	Aye, Nay und Abstain mit Conviction, mit ENJ oder sENJ, Stimmgewicht mit Rechnung.	in Betrieb
Delegieren und freigeben	Delegation je Track oder für alle, Sperren freigeben, Kautionen erstatten, soweit die Runtime es erlaubt.	in Betrieb
Treasury-Anträge	Assistent in drei Schritten, Stufe nach Betrag, Kosten vor der Signatur, fortsetzbare Entwürfe.	in Betrieb
Jeder andere Aufruf	Abbrechen, Whitelist, Upgrade per Code-Hash, Bemerkung oder kodierter Aufruf, unter jeder Admin-Herkunft.	in Betrieb
EGOV1	Text, Anhänge und Aufruf mit Hash auf der Chain festgenagelt, Prüfung im Browser, Bearbeitungen sichtbar.	in Betrieb
Antragstexte	Markdown mit Tabellen, Bildern und PDFs, Adressen mit Prüfsumme, Vorschau wie für Wähler.	in Betrieb
Fellowship	Mitglieder, Ränge, F#-Referenden, gewichtete Abstimmung, Einreichen, der schnelle Weg.	in Betrieb
Wallets	Sechs Wallets, eine Oberfläche für jede Signatur, Enjin Wallet mit einem Tipp.	in Betrieb
Gemeinschaft	Anmeldung per Signatur ohne Gebühr, Profile, Kommentare je Antrag.	in Betrieb
Moderation	Meldungen, eine Warteschlange, Rollen per Wallet, öffentliches Protokoll, optionale automatische Prüfung.	in Betrieb
Betrieb	Statusseite für Admins, Sicherung als ein Archiv mit Wiederherstellungsskript, Meldeformular für Sicherheitslücken.	in Betrieb
Auffindbar	Sitemap mit jedem Referendum, Vorschaubild je Referendum, Metadaten je Seite, private Bereiche nicht indexiert.	in Betrieb
Rechtliches	Impressum, Datenschutz und Nutzungsbedingungen, Link zum Quellcode im Fuß jeder Seite.	in Betrieb

Die beiden Matrixchains sind in der Netzliste eingetragen, aber nicht freigeschaltet. Sie stehen deshalb nicht in dieser Tabelle (Kapitel 14).

10

Im Einsatz

Enjin Governance ist kein Prototyp. Die öffentliche Instanz läuft unter gov.enjin.cloud, und ihre EGOV1-Einträge stehen auf der Enjin Relaychain, wo jeder sie nachprüfen kann.

Die öffentliche Instanz läuft unter gov.enjin.cloud, und ihre Sitemap führt jedes Referendum des Standardnetzes. Die erste öffentliche Version erschien am 28. Juni 2026, Version 2.0 am 11. November. Die App folgt den Regeln der jeweils aktiven Runtime-Version: Sie liest sie von der Chain und übernimmt ein Upgrade, ohne neu ausgeliefert zu werden.

Enjin Relaychain, die ersten 21 Referenden



Abb. 11 Die ersten 21 Referenden der Enjin Relaychain, direkt von der Chain gelesen. Ab #10 trägt jedes einen Metadaten-Hash, also einen Verweis auf seinen Text, und alle elf Texte liegen in der App. Bei sieben passt der SHA-256 des Texts zum Hash auf der Chain. Vier wurden nach dem Einreichen bearbeitet.

Eine Bearbeitung schreibt das JSON an derselben Adresse neu, der Hash auf der Chain zeigt weiter auf die erste Fassung (Kapitel 5). Bei drei der vier bearbeiteten Referenden (#11, #14, #15) wurde zudem der ursprüngliche Umschlag von der Chain entfernt, so dass dort nur noch der Hash der ersten Fassung steht.

1 Der erste Eintrag

17. August 2026

Am 17. August ersetzte der Verweis über `setMetadata` die ältere Bemerkung im Stapel. Am selben Tag nagelte Referendum #10 als erster Antrag auf dem Mainnet seinen Text so fest. Der Text liegt heute noch unter derselben Adresse, und sein Hash stimmt.

2 Die sichtbare Bearbeitung

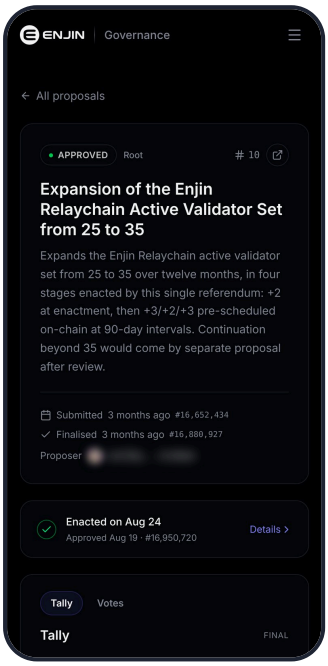
4. November 2026

Referendum #19 wurde vier Minuten nach dem Einreichen von seinem Antragsteller bearbeitet. Die App zeigt „Edited“, der Hash auf der Chain zeigt weiter auf die erste Fassung. Wer beide vergleicht, sieht die Änderung, ohne der App vertrauen zu müssen.

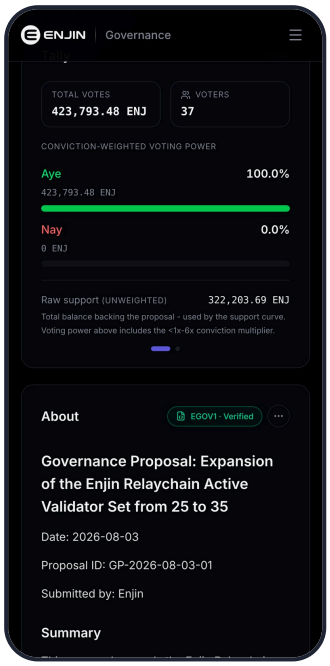
3 Das Upgrade auf 1080

Version 2.0, 11. November 2026

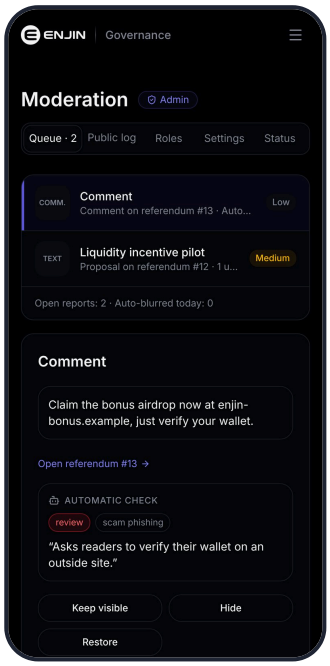
Mit Version 1080 änderten sich Ausgabegrenzen, Kautionen und die Messung der Unterstützung. Laut Änderungsprotokoll wurden Treasury-Stufen, Unterstützung, Erstattungen, Abstain und die Fellowship auf einer Kopie des Mainnets mit 1080 erneut durchgespielt.



Ein Referendum auf dem Mainnet



Antragstext, EGOV1 geprüft



Moderation: eine Warteschlange

Referendum #10 auf gov.enjin.cloud, sein geprüfter Text und die Warteschlange der Moderation. Die Warteschlange sehen nur Moderatoren. Sie stammt deshalb aus einem lokalen Produktions-Build auf der Enjin Relaychain und zeigt die Testdaten des Repositorys. Adressen fremder Konten unkenntlich gemacht.

11

Moderation

Ein offener Raum für Anträge und Kommentare zieht auch Betrug an, etwa Bilder mit einer lesbaren Wiederherstellungsphrase oder Links zu falschen Wallets. Moderation hält ihn sauber, ohne in die Governance einzugreifen.

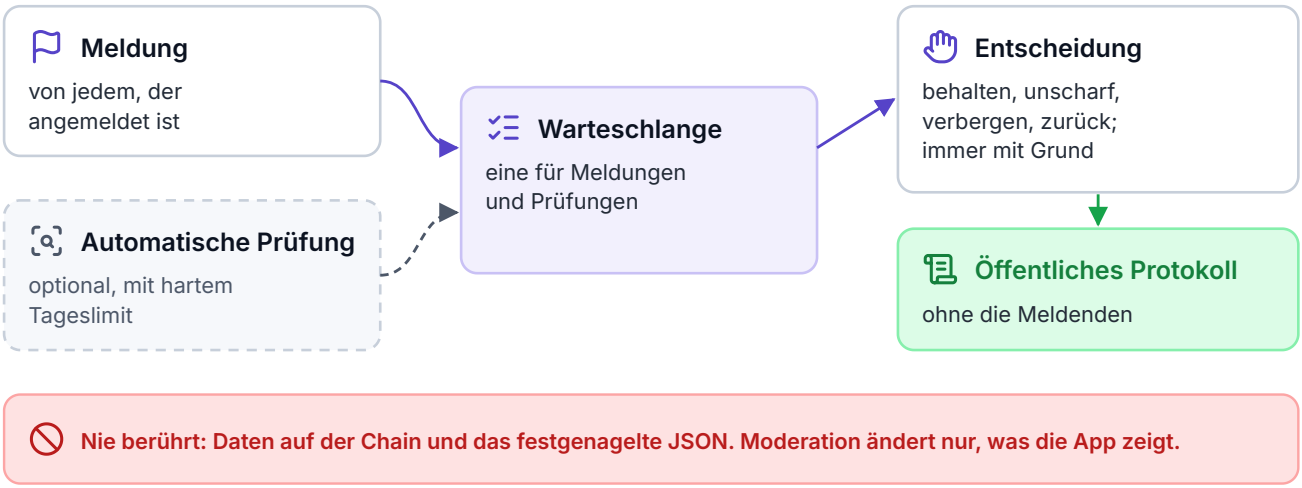


Abb. 12 Meldungen und automatische Prüfungen landen in einer Warteschlange. Moderatoren entscheiden mit Grund, die Entscheidung erscheint öffentlich. Die Chain und das festgenagelte JSON bleiben unberührt.

	Alle	Moderator	Admin
Inhalte melden, öffentliches Protokoll lesen	ja	ja	ja
Warteschlange, behalten, unscharf, verbergen, zurück		ja	ja
Datei löschen, Schreiben einer Wallet pausieren			ja
Rollen, Prüfeinstellungen, Statusseite, Sicherungen			ja

Rollen hängen am öffentlichen Schlüssel einer Wallet und gelten damit in jedem Adressformat. Die ersten Admins stehen in der Konfiguration und lassen sich in der App nicht entfernen. Jede Rolle wird auf dem Server geprüft, nie nur in der Oberfläche.

Automatische Prüfung, mit Grenzen

Admins können eine Prüfung durch ein Sprachmodell einschalten. Bilder und PDFs werden vor dem Speichern geprüft: Ein klarer Verstoß wird abgelehnt oder, wenn die Admins es so wählen, zurückgehalten, ein Grenzfall unscharf gespeichert und erst nach einer Entscheidung ausgeliefert. Antragstexte und Kommentare werden nach dem Veröffentlichen geprüft. Ein Text kann dabei nur in die Warteschlange kommen, nie automatisch verborgen werden. Jede Prüfung wird gezählt, bevor sie gesendet wird, so dass ein Tageslimit die Kosten hart begrenzt. Fällt der Prüfdienst aus, wird nichts blockiert. Ein Einrichtungsfehler wie ein abgelehnter Schlüssel erscheint auf der Statusseite und wird einmal am Tag an die Moderatoren gemeldet.

12

Sicherheit und Datenschutz

Die App hält kein Geld und keine Schlüssel. Signiert wird in der Wallet. Was sie schützen muss, sind Sitzungen, Entwürfe, hochgeladene Dateien und die Entscheidungen der Moderation.

Bereich	Umsetzung heute
Anmeldung	Per Signatur, ohne Passwort, ohne Transaktion. Der Server gibt eine Einmal-Nonce aus und speichert die genaue Nachricht, die signiert werden soll. Die Prüfung verbraucht die Nonce im selben Schritt.
Sitzungen	Ein httpOnly-Cookie. Die Datenbank kennt nur den SHA-256 des Tokens, nie das Token. Abmelden löscht die Sitzung.
Eigentum	Routen vergleichen öffentliche Schlüssel, damit dieselbe Wallet in jedem Adressformat dieselbe bleibt und keine andere passt.
Fremde Seiten	Jede schreibende Anfrage an die API, deren Herkunft eine andere Seite nennt, wird abgelehnt. Schreibende und anonyme Routen haben Ratenlimits.
Uploads	Der Typ wird aus den Bytes gelesen, nicht aus der Angabe des Browsers. Bilder werden neu kodiert, Metadaten wie GPS entfernt. SVG und HTML werden nie angenommen. Dateien gehen nur über eine Route der App hinaus, die die Moderation beachtet.
Darstellung	Markdown wird zu React-Elementen, nie zu eingefügtem HTML. Links müssen http(s) oder Pfade der eigenen Seite sein. Eine Content-Security-Policy und Schutz vor Einbettung kommen dazu.
Geheimnisse	Server- und Browser-Variablen sind im Schema getrennt, kein Secret erreicht den Browser. Die Statusseite meldet nur, ob etwas eingerichtet ist, nie Werte oder Fehlertexte.
Kautionen	Der Aufruf eines laufenden Referendums wird nie zur Erstattung angeboten. Scheitern die Prüfungen dahinter, lädt die Liste gar nicht, statt Ungeprüftes anzubieten.
Sicherungen	Nur Admins erstellen und laden sie, über einen kurzlebigen signierten Link. Sitzungen und Nonces sind nie enthalten.
Meldestelle	Ein öffentliches Formular für Sicherheitslücken, ohne Konto. Die IP-Adresse wird nur als Hash gespeichert.

Das Projekt hat laut Änderungsprotokoll mehrere Sicherheitsprüfungen durchlaufen, darunter eine vor Version 2.0. Diese schloss unter anderem einen präparierten Link, der einem Antragsteller unter harmlosem Titel einen beliebigen Aufruf hätte unterschieben können: Links zum Formular wählen heute nur noch Netz und Kammer. 18 Testdateien führen die echten Routen gegen Angriffe aus und ersetzen dabei nur Datenbank, Speicher und externe Dienste. Eine externe Prüfung gibt es nicht (Kapitel 14).

13

Wie es gebaut wird

Die Geschichte im Repository beginnt am 19. Juni 2026 mit Version 1.0 als erstem Commit. In den ersten fünf Monaten kamen über 180 Commits und 15 zusammengeführte Pull Requests auf die Hauptlinie.

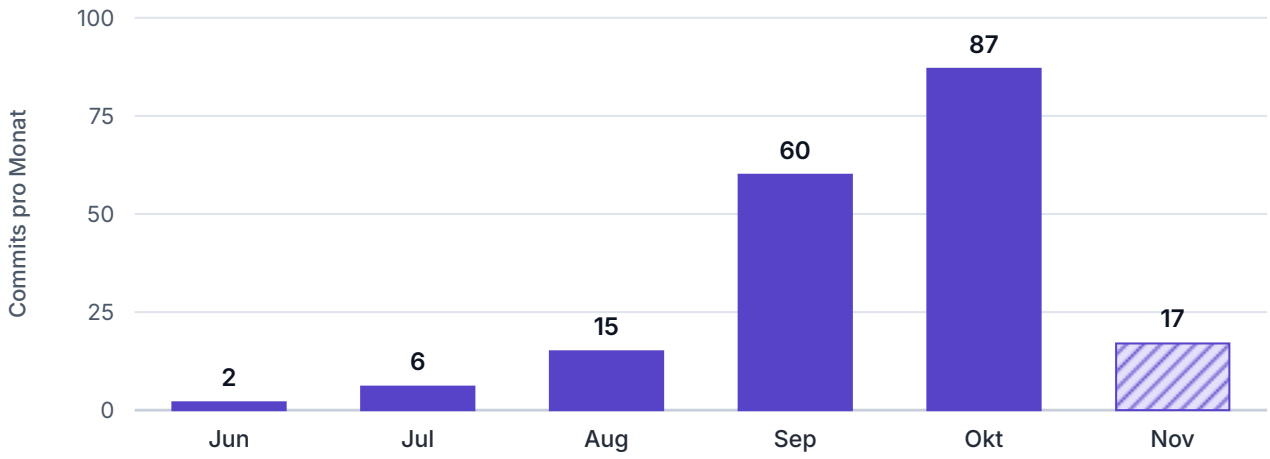


Abb. 13 Commits auf der Hauptlinie pro Monat, maßstabsgetreu. Der letzte Monat ist schraffiert, weil er beim Schreiben noch lief. Was vor dem ersten Commit lag, ist im Repository nicht enthalten. Die Zahl misst Schritte, nicht Wert.

28. Juni 2026
Version 1.0: lesen, abstimmen, delegieren, Treasury-Anträge mit EGOV1

17. August 2026
EGOV1 über `setMetadata`, erster Eintrag auf dem Mainnet

8. Oktober 2026
Upgrades per Code-Hash, Treasury-Stufen nach der Runtime

11. November 2026
Version 2.0: Fellowship, Moderation, Regeln von 1080

13. November 2026
Version 2.0.3

Gebaut und betreut wird das Projekt von Chris Zemmel. Gelegentlich tragen weitere Entwickler bei: etwa jeder zwölfte Commit stammt von einer weiteren Person.

Die Prüfstrecke

Jeder Pull Request und jeder Push auf die Hauptlinie durchläuft dieselben Schritte: Typprüfung, Lint, Suche nach ungenutztem Code, eine Prüfung der Abhängigkeiten auf bekannte Lücken, die Tests und den Build, danach die Browsertests gegen den Produktions-Build. Die Prüfung der Abhängigkeiten läuft zusätzlich jeden Montag, weil Lücken bekannt werden, ohne dass jemand etwas ändert.

- **Unit- und Routentests.** Über 1.200 Tests in über 100 Dateien.
- **Datenbanktests.** Das SQL läuft gegen ein echtes Postgres im Prozess, mit allen Migrationen, die auch die Produktion bekommt.
- **Browsertests.** 29 Tests mit Chromium gegen den Produktions-Build. Die Daten der Chain sind echt, sie kommen vom Canary-Netz. Die Test-Wallet signiert nie.

Fehler, die zu Regeln wurden

Fall	Was geschah	Regel seitdem
Die doppelte Sperre	Auf den meisten Tracks zeigte die App die doppelte Sperrfrist einer Stimme.	Sperrfristen kommen aus der Runtime, nie aus einer Annahme.
Die zu große Stufe	Die kleinste Treasury-Stufe wurde bis 250 ENJ angeboten, erlaubt waren 100.	Grenzen je Runtime-Version, Warnung bei unbekannter Version.
Der falsche Nenner	Die Unterstützung erschien auf dem Mainnet rund 28 Prozent zu niedrig.	Jeder Wert aus Konstanten der Runtime hängt an ihrer Version.
Die leere Wallet	Auf iOS öffnete sich die Wallet, bevor die Anfrage beim Relais war, und fand nichts.	Erst die Bestätigung des Relais, dann die Wallet. Ein zweiter Tipp sendet dieselbe Anfrage.
Der zweite Antrag	Nach verlorenen Statusmeldungen hätte „Erneut versuchen“ ein zweites Mal eingereicht.	Vor jedem neuen Versuch wird nach dem schon eingereichten Referendum gesucht.
Der verborgene Text	Ein verborgener Antrag war über die API noch lesbar.	Verborgen heißt überall verborgen, außer im JSON, dessen Hash auf der Chain steht.

14

Grenzen heute

Ein Manifest, das seine Grenzen verschweigt, verdient kein Vertrauen.

Ein-Personen-Projekt

Eine Person baut und betreut die App, mit gelegentlichen Beiträgen anderer. Fällt sie aus, steht die Weiterentwicklung. Der Code bleibt offen, und jeder kann eine eigene Instanz betreiben.

Gehostete Dienste

Die öffentliche Instanz läuft bei Anbietern für Hosting, Datenbank und Dateispeicher. Fallen sie aus, fehlen Texte, Profile und Kommentare. Lesen und Abstimmen hängen nur an der Chain.

Das Abzeichen prüft gegen die App

„Verified“ vergleicht mit dem Hash, den der Server beim Verknüpfen gegen die Chain geprüft und gespeichert hat. Den Umschlag auf der Chain liest der Browser dafür nicht selbst neu. Die unabhängige Prüfung aus Kapitel 5 bleibt jedem offen.

Nur die Relaychains

Die beiden Matrixchains sind eingetragen, aber nicht freigeschaltet. Die Enjin Matrixchain nutzt noch ein älteres Governance-Modul, das die App nicht abbildet.

Öffentliche Knoten

Ohne eigenen Knoten liest die App über öffentliche Endpunkte. Sind sie langsam oder fallen sie aus, zeigt die App den Fehler und versucht es erneut.

Kein externes Audit

Die Sicherheitsprüfungen sind im Änderungsprotokoll beschrieben und durch Tests belegt. Eine Prüfung durch Dritte steht aus. Die automatische Prüfung von Inhalten ist eine Hilfe, keine Garantie.

WAS NICHT GEPRÜFT IST

„Nicht geprüft“ ist ein gültiger Stand. Diese Punkte stehen im Repository, wurden für dieses Manifest aber nicht gegen die Wirklichkeit nachgeprüft:

- Die Durchläufe auf Kopien des Mainnets und des Canary-Netzes sind im Änderungsprotokoll beschrieben. Ein Skript, das sie wiederholt, liegt nicht im Repository.
- Die Zahlen zum Fall des Nenners (inaktive Ausgabe, Referendum #15) stammen aus der Projektdokumentation, nicht aus einer eigenen Messung.
- Die Browsertests liefen für dieses Manifest nicht vollständig. Die Bildschirmfotos stammen von gov.enjin.cloud, die Ansichten mit Wallet und Moderation aus einem lokalen Produktions-Build auf der Enjin Relaychain.
- Wie die öffentliche Instanz im Betrieb verfügbar war, ist nicht gemessen.
- Für #11, #14 und #15 ist belegt, dass der ursprüngliche Umschlag nicht mehr auf der Chain liegt. Wie er entfernt wurde, ist nicht einzeln nachgeprüft.

KONTAKT

Chris Zemmel Zyric · gov.enjin.cloud · github.com/chriszemmel/enjin-governance

Enjin Governance ist freie Software unter der AGPL-3.0.