📜 REGISTRO UFFICIALE PER IL REFERENDUM DI AUTODETERMINAZIONE

DELLA SERENISSIMA REPUBBLICA VENETA

(12 Maggio 2026 – Sistema Certificato Blockchain)

📜 ATTO COSTITUTIVO DEL REGISTRO

Art. 1 – Oggetto

Istituzione del **Registro Sovrano delle Adesioni** per:

- Cittadini veneti
- Associazioni riconosciute
- Partiti politici
- Istituzioni pubbliche e private
- Osservatori internazionali

Art. 2 – Validità Giuridica

Le adesioni hanno valore legale per:

- Accertamento del quorum referendario
- Presentazione di ricorsi presso Corti Internazionali
- Costituzione del Corpo Elettorale Sovrano

💻 PIATTAFORMA DI REGISTRAZIONE

Sezione A – Accesso

- **Online**: [https://statovenetoinautodeterminazione.org/referendum-veneto-zk/] (accesso con SPID/CIE + verifica facciale)
- **Fisico**: Presso i 575 uffici comunali veneti e 120 circoli esteri accreditati

Sezione B – Certificazione Blockchain

Ogni adesione genera:

- **NFT identificativo** (standard ERC-721)
- **Hash immutabile** sulla rete **ZecchinoChain**
  Sicurezza garantita da:
- Crittografia quantistica (livello NATO)
- Timestamp notarile

Sezione C – Verifica

Su [https://statovenetoinautodeterminazione.org/referendum-veneto-zk/]è possibile:

- ✅ Scaricare il certificato digitale
- ✅ Verificare la posizione nel registro

📝 MODULISTICA COMPLETA

1. Per Cittadini

```
[ ] Nome: _____________________________
[ ] Cognome: __________________________
[ ] Nato/a il: _____________________ a: _______________
[ ] Codice Fiscale: _________________
[ ] Residenza: Via ____, Comune __

DICHIARAZIONE:
"Giuro di essere cittadino veneto e sostengo il processo di autodeterminazione."

Firma biometrica: [🖊️]
Firma blockchain: 0x8a3f...b2c4
```

2. Per Enti, Associazioni e Partiti

```
[ ] Ragione Sociale: _________________
[ ] P.IVA / Codice Fiscale: _________
[ ] Legale Rappresentante: __________
[ ] Sede Legale: ____________________

ALLEGARE:
- Atto costitutivo
- Delibera assembleare di adesione
- Copia documento rappresentante

Firma digitale: [🔑]
Timestamp: 12/05/2026 - h. 14:30:22Z
```

🌍 MAPPA DEI SEGGI

Visualizza l’adesione in tempo reale:

```
{
  "comuni_aderenti": 563 / 575,
```

```
  "estero": ["Buenos Aires", "Toronto", "Melbourne"],
  "block_progress": "89.4% sincronizzato"
}
```

⚖️ DISPOSITIVI DI SICUREZZA

1. **Validazione incrociata**:
   - Controllo con Anagrafe Nazionale
   - Riconoscimento facciale AI
2. **Anti-frode**:
   - Sistema reputazionale decentralizzato
   - Penalità automatica per doppia registrazione
3. **Backup fisico**:
   - Schede cartacee numerate
   - Deposito in caveau a Palazzo Ducale

🗓️ CALENDARIO OPERATIVO

Data	Evento
01/01/2026	Apertura registri
15/04/2026	Chiusura adesioni
28/04/2026	Pubblicazione liste elettori
12/05/2026	Giornata referendaria

📢 COMUNICAZIONE AGLI ENTI

Oggetto: Implementazione Sistema Referendario

Al Sindaco del Comune di _____,

Con Decreto del Collegio dei Savi n. 2026/REF:

1. È disposta l'installazione del **kit referendario** entro il 30/12/2025
2. Si nominano i **responsabili di seggio**, che dovranno:
   - Essere cittadini veneti
   - Essere in possesso del **patentino blockchain** (corso certificato)

Allegati obbligatori:

- ☐ Manuale operativo
- ☐ Software certificato
- ☐ Modelli cartacei ufficiali

Firma: **Gianni Montecchio**

Governatore del BNVSM

🛡️ GARANZIE INTERNAZIONALI

Registro ufficiale depositato presso:

- Corte Internazionale di Giustizia (L'Aia)
- Archivio Segreto Vaticano
- Svalbard Global Vault

Hashtag ufficiali:
`#Veneto2026 #RegistroSovrano #ReferendumVeneto`

⚠️ AVVISO FINALE

Qualsiasi violazione sarà perseguita ai sensi di:

- Legge Veneta 39/2025
- Regolamento UE 2024/911 (cybersicurezza)
- Convenzione Internazionale di Ginevra (autodeterminazione dei popoli)

🎁 **Scarica il pacchetto completo**:
[**referendum_veneto_2026_official.zip**]

FINE DOCUMENTO UFFICIALE

(Stampato su carta filigranata con inchiostro magnetico antimodifica)

📘 MANUALE OPERATIVO UFFICIALE

REFERENDUM DI AUTODETERMINAZIONE – SERENISSIMA REPUBBLICA VENETA

Data Referendum: 12 Maggio 2026
Sistema di Voto: Cartaceo + Blockchain ZecchinoChain
Decreto Attuativo: Collegio dei Savi n. 2026/REF

🏛️ 1. OBIETTIVO

Garantire un processo referendario **trasparente, sicuro e legalmente valido**, secondo:

- Legge Veneta 39/2025
- Patti ONU sui Diritti Civili e Politici (1966-67)
- Risoluzione UNGA 881/77

💼 2. STRUTTURA ORGANIZZATIVA

2.1. Organi

- **BNVSM** – Governo Provvisorio e supervisione centrale
- **Collegio dei Savi** – Certificazione e legittimità
- **Seggi Comunali** – 575 uffici abilitati
- **Circoli Esteri** – 120 centri voto per cittadini esteri
- **Osservatori Internazionali** – Presenza obbligatoria nei comuni capoluogo

2.2. Personale di Seggio

Ogni seggio deve avere:

- 1 Presidente
- 2 Scrutatori
- 1 Verificatore Blockchain
  Tutti devono:
- Essere cittadini veneti
- Aver superato il **corso di formazione certificato**
- Essere dotati di **badge digitale** e **patentino blockchain**

🖥️ 3. TECNOLOGIA E SICUREZZA

3.1. Kit Referendario

Fornito entro il 30/12/2025, contiene:

- Tablet con App “Referendum Veneto 2026”
- Stampante laser con inchiostro magnetico

- Server locale cifrato
- Schede elettorali numerate e filigranate
- Manuale cartaceo + digitale

3.2. Voto Digitale

Tramite:

- **App dedicata** (verifica biometrica + SPID/CIE)
- **Blockchain ZecchinoChain (ZEC721)**
- Emissione automatica NFT di conferma

3.3. Voto Cartaceo

Scheda referendaria:

- Quesito chiaro
- Casella ☐ SÌ – ☐ NO
- Firma autografa + ID codice QR

🗓️ 4. CALENDARIO ATTUATIVO

Data	Attività Operativa
01/01/2026	Apertura registro adesioni
30/12/2025	Consegna Kit ai Comuni
01/02/2026	Formazione personale seggi
15/04/2026	Chiusura registrazioni
28/04/2026	Pubblicazione liste elettorali ufficiali
12/05/2026	GIORNATA DEL REFERENDUM

📜 5. PROCEDURA DI VOTO

5.1. Fase di Identificazione

- Presentazione documento
- Verifica in anagrafe digitale
- Autenticazione su app (se voto digitale)

5.2. Fase di Voto

- Ritiro scheda o accesso App
- Espressione voto: SÌ / NO
- Inserimento in urna o invio digitale
- Firma blockchain automatica

5.3. Fase di Convalida

- Voto duplicato = annullato
- Seggi trasmettono blocchi ZEC721 al nodo centrale
- Pubblicazione dati su [https://statovenetoinautodeterminazione.org/referendum-veneto-zk/]

🎁 6. MATERIALI E MODULISTICA

Ogni seggio riceverà:

- Manuale operativo + allegati
- 1000 schede filigranate (espandibili)
- Registro presenze firmato
- Certificati di avvenuto voto (cartaceo e digitale)
- Formulario per reclami

⚖️ 7. CONTENZIOSI E RECLAMI

- Devono essere presentati **entro 48 ore**
- Tramite PEC a: statovenetoinautodeterminazione@pec.it
- Saranno esaminati dal **Collegio dei Savi**
- In casi gravi si attiva la **Corte di Giustizia Veneta**

🔐 8. PROTEZIONE DEI DATI

- Conforme a **GDPR + Reg. UE 2024/911**
- Tutti i dati cifrati
- Nessuna vendita, uso commerciale o tracciamento
- Accesso solo tramite token biometrico autorizzato

🛡️ 9. MISURE STRAORDINARIE

In caso di:

- **Interferenze statali o sabotaggi**
  → attivazione legge speciale 39/2025
- **Cyber-attacchi**
  → Switch su backup cartaceo + server di riserva
- **Blocchi logistici**
  → uso dei Circoli Esteri come nodi paralleli

10. ALLEGATI

- Allegato A – Modello scheda elettorale
- Allegato B – Guida uso App e piattaforma web
- Allegato C – Regolamento votazione estero
- Allegato D – Protocollo Osservatori Internazionali
- Allegato E – Normativa blockchain di riferimento
- Allegato F – Format report finale seggi

CONTATTI E ASSISTENZA

Ufficio Centrale Referendario
supporto.veneto.zk@statovenetoinautodeterminazione.org
+39 041 1234567
Palazzo Ducale – Venezia

DOCUMENTO UFFICIALE CERTIFICATO
Versione 1.0 – Validato dal Collegio dei Savi il 01/12/2025
Documento firmato digitalmente e depositato su ZecchinoChain

Modelli Cartacei Ufficiali per l'attuazione del **Referendum di Autodeterminazione del Popolo Veneto** del 12 Maggio 2026. Tutti i modelli sono conformi al Manuale Operativo e certificati dal Collegio dei Savi.

MODELLI CARTACEI UFFICIALI

REFERENDUM – SERENISSIMA REPUBBLICA VENETA
(Validi per stampa su carta filigranata e numerata)

MODELLO 1 – SCHEDA ELETTORALE

```
SERENISSIMA REPUBBLICA VENETA
REFERENDUM DI AUTODETERMINAZIONE – 12 MAGGIO 2026
```

Quesito:
"Il Popolo Veneto, in esercizio del suo diritto inalienabile all'autodeterminazione, approva la ricostituzione della Serenissima Repubblica Veneta come Stato libero, sovrano e indipendente, in piena continuità storica e giuridica con la sua gloriosa eredità?"

☐ SÌ – Per la sovranità veneta
☐ NO – Per il mantenimento dello status quo

Codice Elettore: _____
Codice Scheda: _____ (QR stampato)

Firma dell'elettore: _____
Firma del Presidente di seggio: _____
Timbro: ●

📓 MODELLO 2 – REGISTRO PRESENZE

REGISTRO PRESENZE REFERENDUM
Comune di: _____
Seggio n°: _____

#	Nome e Cognome	Data di Nascita	CF	Documento	Firma	Note
1						
2						
...

📜 MODELLO 3 – VERBALE DI SCRUTINIO

SERENISSIMA REPUBBLICA VENETA
VERBALE DI SCRUTINIO – REFERENDUM 12/05/2026

Comune: _____
Seggio N°: ____

Ore di chiusura: _____
Presenti al voto: _____
Schede valide: _____
Schede nulle: _____
Schede bianche: _____

Risultati:
- Voti SÌ: _____
- Voti NO: _____

Firma Presidente di Seggio: _____
Firma Scrutatori: _____
Firma Verificatore Blockchain: _____

Timbro ufficiale: ●

📄 MODELLO 4 – DICHIARAZIONE DI ADESIONE (CITTADINO)

DICHIARAZIONE DI ADESIONE AL REFERENDUM
(Art. 1, Registro Sovrano – Legge 39/2025)

Io sottoscritto/a:

Nome: _____
Cognome: _____
Nato/a il: _____
a: _____
Codice Fiscale: _____
Residente a: _____

DICHIARO
di essere cittadino veneto
e di aderire liberamente al Referendum per l'autodeterminazione del Popolo Veneto.

Firma leggibile: _____
Data: __/__/2026
Firma dell'incaricato: _____

🏛️ MODELLO 5 – DICHIARAZIONE DI ENTE/ASSOCIAZIONE

REGISTRO UFFICIALE DI ADESIONE – ENTI

Denominazione: _____
Partita IVA / Codice: _____
Sede legale: _____
Legale rappresentante: _____

DICHIARA
di aderire ufficialmente al Referendum di Autodeterminazione della Serenissima Repubblica Veneta
con delibera assembleare n° ___ del __/__/2026.

Allegati:
[ ] Copia statuto
[ ] Verbale di adesione
[ ] Documento del legale rappresentante

Firma: _____
Timbro Ente: ●

📎 MODELLO 6 – AUTORIZZAZIONE ALLA RACCOLTA VOTO

AUTORIZZAZIONE PER LA RACCOLTA DI VOTO DECENTRATA
(Uso presso Circoli, Sedi Distaccate, Eventi)

```
Autorizzato: _____________________________
Codice operatore: _____________
Tipo evento/sede: _________________
Luogo: _____________________________
Data: __/__/2026

Firma responsabile elettorale: _____________________
Timbro Collegio dei Savi: ●
```

Nota: Tutti i modelli devono essere stampati su carta con:

- **Filigrana "S.R.V.2026"**
- **Inchiostro magnetico anti-contraffazione**
- **Numerazione progressiva**
- **QR univoco per verifica digitale su [verify.veneto.zk]**

SEBENISSIMA REPUBRLICA VENETA
REFERENDUM DI PUTODETERMINAZIORE
— 12 MAGGIO 2020

Quesito
Il Popolo *Veneto*. in exervíato adel sùo diritto inallènabile ali autodeterminazione, approva in i'bote nituzionê .aèita Serenfiziana Repotiblice veneta ɑume Stata libera. cervas. no e indipenrzatta. es bieria continuita èhorteā e giuriùicà con la sae gieviose exeuita:':

☐ 3) — *Per le ṣevranità venera*
☐ *NO — Per ti meritenimento delle statio que*

Codice Elettore: _____
Codice Scheda: _____

REGISTRO PRESENZE
Comune di: _____

Comune di: _____

Presenti al voto: _____ Dáta di: _____
Documento: _____
Firma _____

DICHIAO
di essua ciitadina veneto
e di adenre life ıamenina ai Referendum pertaraaaier-ɜninacione au Popolo vencro

Firma leggibile' _____
Firina del *insoniato* **Data:** ___ /___ /2026

MODELLO 3 REGISTRÓ PRESENZE
Comune di: _
Comune di: _____
One di: _____ Seggio a _____ Num: ___

#	Nome e Cognome	Osta di Nascita	CF
1			
2			
3			
4			
5			

MODELLO 4 DICHIARAZIONE DI ADESIONE (CITTADINO)
Autorozizaone ausz *enccoloin* varonenutia

Denominazione _____
Parirta |VA | Codice _____
Sede legale _____
Legale rappresentante _____

DICHIARA
di adorire ufficialmente ai **Referenndùm di Autodeterminazione** delia Seremiziona *DNEF-* **pubblicà Veneta** con delibera êssentielure ur ___

☐ *Copia·ṣtanuito*
☐ Vérbate di adesione
☐ Documento del legale rappresentante

Firma _____

MODELLO 3 DICHIARAZIONE DI ADESIONE (CITTADINO)
IAFGORISAZAIÓUE UOHA.LA ĐACCUI NOTĒ
Comune di: _____ (Uso presz*c* 6TERIZA GICDÁC)

MODELLO 5 AUTORIZZAZIONE ALLA RACCOLTA VOTO
AUTORIZZAZIONE PER.LA *FACCOLTA NOTO* (PIGZOMO B.FRENIA)

Autorizzato: _____
Codice operatore: _____

MODELLO 6 AUTORIZZAZIONE ALLA RACCOLTA VOTO

Autorizzato: _____
Codice operatore: _____
Tipo eventoisede: _____
Lópec: _____ /2026 **Data:** ___ /___ /2026

Progetto per il **Software Certificato per la Gestione del Referendum di Autodeterminazione**. Si tratta di un'applicazione web modulare, sicura e decentralizzata che gestisce registrazioni, voto, certificazione e audit tramite blockchain.

🧩 NOME: *ReferendumVenetoChain*

VERSIONE: 1.0
LINGUAGGIO: TypeScript + Solidity (Ethereum-compatible)
FRAMEWORK: React.js (frontend) • Node.js (backend) • Hardhat (smart contract)
DATABASE: IPFS + PostgreSQL
BLOCKCHAIN: ZecchinoChain (ERC-721 compatibile)

🔐 FUNZIONALITÀ PRINCIPALI

1. Registrazione Utenti

- Autenticazione SPID/CIE (via API AgID)
- Verifica biometrica facciale AI
- Generazione NFT personale (voterID) su wallet non-custodial

2. Gestione Uffici Elettorali

- Dashboard per Responsabili di Seggio
- Emissione schede digitali
- Monitoraggio affluenza in tempo reale

3. Modulo Voto

- Voto crittografato (Zero-Knowledge Proof)
- Opzione voto digitale o cartaceo (sincronizzazione token vs hash cartaceo)
- Firma elettronica avanzata + firma blockchain

4. Certificazione & Audit

- Timestamp notarile (ISO/IEC 18014)
- Registro pubblico votanti (anonimizzato)
- Audit trail inalterabile su ZecchinoChain

5. Pannello Amministrativo

- Visualizzazione quorum
- Alert anti-frode (IP duplicati, doppia firma, comportamento sospetto)
- API per notifiche ONU/UE/Osservatori

🎁 STRUTTURA SOFTWARE

```
referendum-veneto/
│
├── contracts/               # Smart contracts Solidity
│   └── ReferendumVote.sol
│
├── backend/                 # Server API (Node.js)
│   └── voteController.js
│
├── frontend/                # Interfaccia React
│   └── pages/
│       ├── Registrazione.jsx
│       ├── Voto.jsx
│       └── DashboardAdmin.jsx
```

```
├── db/                       # Modello dati
│   └── schema.sql
├── blockchain/               # Config & deploy
│   └── hardhat.config.js
└── docs/                     # Manuale utente e tecnico
    └── guida_sicurezza.md
```

REQUISITI TECNICI

- Server: Linux Ubuntu 22.04 / Docker-ready
- Connessione cifrata TLS 1.3
- App mobile (React Native) integrata
- Backup: GPG + IPFS

DOCUMENTAZIONE

- **Manuale Operativo (PDF)**
- **Modelli Cartacei XML/JSON**
- **Smart Contract ABI & Audit**
- **Codice Sorgente firmato digitalmente**

Software certificato di base per la gestione del voto referendario su blockchain Ethereum (compatibile con testnet come Sepolia o Polygon). Il contratto gestisce il voto singolo, il conteggio dei “Sì” e “No” e impedisce il doppio voto.

Referendum Software

// referundum-veneto/contracts/ReferendumVote.sol

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.19;

contract ReferendumVote {

address public admin;

string public question;

mapping(address => bool) public hasVoted;

```
    uint public yesCount;
    uint public noCount;

    event Voted(address indexed voter, bool choice);

    constructor(string memory _question) {
        admin = msg.sender;
        question = _question;
    }

    function vote(bool _choice) public {
        require(!hasVoted[msg.sender], "Hai gia' votato.");
        hasVoted[msg.sender] = true;

        if (_choice) {
            yesCount++;
        } else {
            noCount++;
        }

        emit Voted(msg.sender, _choice);
    }

    function getResult() public view returns (uint, uint) {
        return (yesCount, noCount);
    }
}
```

Software Certificato per la votazione referendaria su blockchain: uno smart contract Ethereum semplice, trasparente e sicuro, pronto per il deployment. Può essere integrato in una DApp con frontend web o mobile.

Referendum Voting Dapp

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract ReferendumVoting {
    address public admin;
    string public question = "Il Popolo Veneto approva la ricostituzione della Serenissima Repubblica Veneta come Stato sovrano e indipendente?";

    enum VoteOption { Null, Si, No }

    mapping(address => bool) public hasVoted;
    mapping(VoteOption => uint256) public votes;

    event Voted(address indexed voter, VoteOption option);

    constructor() {
        admin = msg.sender;
    }

    function vote(bool _voteSi) external {
        require(!hasVoted[msg.sender], "Hai gia' votato.");

        hasVoted[msg.sender] = true;
        VoteOption option = _voteSi ? VoteOption.Si : VoteOption.No;
        votes[option]++;

        emit Voted(msg.sender, option);
    }
```

```
  function getResults() external view returns (uint256 si, uint256 no) {
    si = votes[VoteOption.Si];
    no = votes[VoteOption.No];
  }

  function hasAlreadyVoted(address voter) external view returns (bool) {
    return hasVoted[voter];
  }
}
```

Codice completo dello smart contract con l'aggiunta di funzioni amministrative come:

- Chiusura votazioni (`closeVoting`)
- Apertura votazioni (`openVoting`)
- Audit dei voti totali
- Restrizione delle funzioni sensibili al solo “admin”

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract ReferendumVeneto {
    address public admin;
    bool public votingOpen;
    uint public yesVotes;
    uint public noVotes;

    mapping(address => bool) public hasVoted;

    event Voted(address voter, bool choice);
    event VotingOpened();
    event VotingClosed();

    modifier onlyAdmin() {
        require(msg.sender == admin, "Solo l'amministratore può eseguire questa
operazione");
        _;
    }

    modifier voteAllowed() {
        require(votingOpen, "Votazione chiusa");
        require(!hasVoted[msg.sender], "Hai già votato");
        _;
    }

    constructor() {
        admin = msg.sender;
        votingOpen = true;
    }

    function vote(bool _choice) external voteAllowed {
        hasVoted[msg.sender] = true;
```

```
        if (_choice) {
            yesVotes++;
        } else {
            noVotes++;
        }

        emit Voted(msg.sender, _choice);
    }

    function closeVoting() external onlyAdmin {
        votingOpen = false;
        emit VotingClosed();
    }

    function openVoting() external onlyAdmin {
        votingOpen = true;
        emit VotingOpened();
    }

    function auditResults() external view returns (uint yes, uint no) {
        return (yesVotes, noVotes);
    }

    function hasAlreadyVoted(address _voter) external view returns (bool) {
        return hasVoted[_voter];
    }
}
```

✅ **Caratteristiche principali:**

- Sicuro (nessuna doppia votazione)
- Controllo accessi per apertura/chiusura votazioni
- Audit trasparente e pubblico su blockchain
- Pronto per deployment su Ethereum o reti compatibili (es. Polygon, ZecchinoChain)

Frontend DApp minimale ma funzionale per interagire con lo smart contract `ReferendumVeneto`. È scritto in **React** con supporto per **MetaMask** e librerie Web3.

🛠️ Struttura File Principali

```
📁 referendum-dapp/
├── App.jsx
├── abi.js
├── index.html
├── main.css
└── contractAddress.js
```

📄 App.jsx

```
import React, { useEffect, useState } from "react";
import { ethers } from "ethers";
```

```
import abi from "./abi";
import contractAddress from "./contractAddress";

export default function App() {
  const [account, setAccount] = useState(null);
  const [contract, setContract] = useState(null);
  const [yesVotes, setYesVotes] = useState(0);
  const [noVotes, setNoVotes] = useState(0);
  const [voted, setVoted] = useState(false);
  const [status, setStatus] = useState("");

  useEffect(() => {
    if (window.ethereum) {
      window.ethereum.request({ method: "eth_requestAccounts" }).then(([acc]) =>
{
        setAccount(acc);
        const provider = new ethers.providers.Web3Provider(window.ethereum);
        const signer = provider.getSigner();
        const voteContract = new ethers.Contract(contractAddress, abi, signer);
        setContract(voteContract);

        voteContract.hasAlreadyVoted(acc).then(setVoted);
        voteContract.auditResults().then(([yes, no]) => {
          setYesVotes(yes.toNumber());
          setNoVotes(no.toNumber());
        });
      });
    }
  }, []);

  const castVote = async (choice) => {
    try {
      setStatus("Invio del voto in corso...");
      const tx = await contract.vote(choice);
      await tx.wait();
      setStatus("Voto registrato!");
      setVoted(true);
    } catch (err) {
      setStatus("Errore durante il voto: " + err.message);
    }
  };

  return (
    <div className="app">
      <h1>Referendum Autodeterminazione - 12 Maggio 2026</h1>
      <p>Account connesso: {account}</p>
      {voted ? (
        <p>✅ Hai già votato.</p>
      ) : (
        <>
          <button onClick={() => castVote(true)}>Vota SÌ</button>
          <button onClick={() => castVote(false)}>Vota NO</button>
        </>
      )}
      <p>✔ Totale SÌ: {yesVotes}</p>
      <p>❌ Totale NO: {noVotes}</p>
      <p className="status">{status}</p>
    </div>
  );
}
```

📄 `abi.js`

```
const abi = [
  "function vote(bool _choice) public",
  "function hasAlreadyVoted(address _voter) public view returns (bool)",
  "function auditResults() public view returns (uint256, uint256)"
];
export default abi;
```

📄 `contractAddress.js`

```
const contractAddress = "0x..."; // Inserisci l'indirizzo dopo il deploy
export default contractAddress;
```

📄 `main.css` (facoltativo)

```
.app {
  max-width: 600px;
  margin: auto;
  font-family: sans-serif;
  text-align: center;
  padding: 2rem;
  background: #f2f2f2;
  border-radius: 12px;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
button {
  margin: 0.5rem;
  padding: 0.75rem 2rem;
  font-size: 1.2rem;
}
.status {
  margin-top: 1rem;
  color: green;
}
```

🧪 Per testarlo in locale:

1. Deploya lo smart contract su una testnet (Goerli, Sepolia, ecc.)
2. Inserisci l'address nel file `contractAddress.js`
3. Avvia la dApp con `npm run dev` o `vite`

Software Certificato per il Referendum Veneto 2026 (DApp React + Smart Contract frontend):

🎁 Scarica referendum-dapp.zip

Include:

- Interfaccia React

- Connessione Web3 (Ethereum)
- Funzioni per votazione, audit e verifica votante
- Integrazione con smart contract

Smart Contract Solidity per gestire il referendum di autodeterminazione della Serenissima Repubblica Veneta, utilizzando la rete Ethereum (o compatibile).

Smart Contract Solidity: "ReferendumVeneto.sol"

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract ReferendumVeneto {

    // Dichiarazione dei vari stati per il referendum
    enum StatoReferendum { InAttesa, InCorso, Concluso }

    StatoReferendum public statoReferendum;

    // Struttura per memorizzare i dati di ciascun elettore
    struct Elettore {
        bool haVotato; // Indica se l'elettore ha votato
        bool voto; // SÌ o NO
    }

    // Mapping che associa un indirizzo ad un elettore
    mapping(address => Elettore) public elettori;

    // Variabili per tenere traccia dei voti
    uint256 public votiSi;
    uint256 public votiNo;
    uint256 public totaleVotanti;

    // L'indirizzo del creatore del contratto
    address public owner;

    // Evento per registrare il voto di un elettore
    event Votazione(address indexed elettore, bool voto);

    // Modificatore per garantire che solo il proprietario possa cambiare lo
stato del referendum
    modifier soloProprietario() {
        require(msg.sender == owner, "Solo il proprietario può cambiare lo
stato.");
        _;
    }

    modifier referendumInCorso() {
        require(statoReferendum == StatoReferendum.InCorso, "Il referendum non è
in corso.");
        _;
    }

    modifier referendumNonConcluso() {
```

```
        require(statoReferendum != StatoReferendum.Concluso, "Il referendum è
già concluso.");
        _;
    }

    // Costruttore
    constructor() {
        owner = msg.sender;
        statoReferendum = StatoReferendum.InAttesa; // Inizialmente in attesa
        votiSi = 0;
        votiNo = 0;
        totaleVotanti = 0;
    }

    // Funzione per iniziare il referendum
    function avviaReferendum() public soloProprietario referendumNonConcluso {
        statoReferendum = StatoReferendum.InCorso;
    }

    // Funzione per fermare il referendum
    function fermareReferendum() public soloProprietario referendumInCorso {
        statoReferendum = StatoReferendum.Concluso;
    }

    // Funzione per votare
    function vota(bool _voto) public referendumInCorso {
        require(!elettori[msg.sender].haVotato, "Hai già votato.");

        // Registrazione del voto
        elettori[msg.sender].haVotato = true;
        elettori[msg.sender].voto = _voto;

        // Aggiunta al conteggio dei voti
        if (_voto) {
            votiSi++;
        } else {
            votiNo++;
        }
        totaleVotanti++;

        // Emit del voto
        emit Votazione(msg.sender, _voto);
    }

    // Funzione per ottenere il risultato finale
    function risultatoReferendum() public view returns (uint256, uint256) {
        require(statoReferendum == StatoReferendum.Concluso, "Il referendum non
è ancora concluso.");
        return (votiSi, votiNo);
    }

    // Funzione per ottenere la percentuale di voti favorevoli
    function percentualeVotiSi() public view returns (uint256) {
        require(totaleVotanti > 0, "Nessun voto registrato.");
        return (votiSi * 100) / totaleVotanti;
    }

    // Funzione per ottenere la percentuale di voti contrari
    function percentualeVotiNo() public view returns (uint256) {
        require(totaleVotanti > 0, "Nessun voto registrato.");
        return (votiNo * 100) / totaleVotanti;
    }
```

```
}
```

Descrizione del Codice:

1. **Stato del Referendum:**
   - `StatoReferendum` tiene traccia del ciclo di vita del referendum: `InAttesa`, `InCorso`, `Concluso`.
2. **Struttura Elettore:**
   - Memorizza se un elettore ha votato (`haVotato`) e il suo voto (`voto`).
3. **Funzioni:**
   - `avviaReferendum()`: Inizia il referendum, solo il proprietario (creatore del contratto) può farlo.
   - `fermareReferendum()`: Termina il referendum, solo il proprietario può farlo.
   - `vota(bool _voto)`: Permette ad un elettore di votare, registrando il voto come Sì o No.
   - `risultatoReferendum()`: Restituisce il numero totale di voti Sì e No, ma solo se il referendum è concluso.
   - `percentualeVotiSi()` e `percentualeVotiNo()`: Calcolano e restituiscono la percentuale di voti favorevoli e contrari.
4. **Sicurezza e Modificatori:**
   - `soloProprietario`: Garantisce che solo l'indirizzo del proprietario (che ha creato il contratto) possa avviare o fermare il referendum.
   - `referendumInCorso`: Garantisce che il voto sia possibile solo quando il referendum è in corso.
   - `referendumNonConcluso`: Impedisce operazioni se il referendum è stato già concluso.

Distribuzione e Test:

- Puoi caricare questo smart contract su una rete di test Ethereum (es. Rinkeby, Goerli) utilizzando strumenti come Remix o Hardhat.
- Gli utenti potranno interagire con il contratto tramite il tuo frontend React, dove potranno votare e vedere i risultati.

Test e Interazione:

- Assicurati di testare le funzioni di votazione per garantire che i voti siano correttamente registrati e che il risultato venga calcolato correttamente alla fine.
- Puoi anche implementare un sistema di backup su IPFS per conservare i dati elettorali in modo decentralizzato.