Proyecto miVoto - Documentación Funcional y Técnica

Generado el 28/10/2025 18:19:28

# 1. Visión general

miVoto es una plataforma de votación digital que combina autenticación con MiArgentina, almacenamiento de resultados en Firestore y registro de votos en una blockchain EVM mediante un Soulbound Token (SBT). El sistema permite realizar el ciclo completo de una elección: login, emisión de token de elegibilidad, voto, recuento y verificación.

# 2. Arquitectura

Componentes principales:

* Frontend Vite/React (puerto 3000).
* Backend Spring Boot (puerto 8080).
* Mock MiArgentina OAuth (puerto 9999) con selección de identidades.
* Firestore (emulador o proyecto real).
* Nodo EVM (Hardhat/Anvil o testnet) con contratos MiVotoSoulboundToken y MiVotoElection.

# 3. Flujo de autenticación y elegibilidad

1. El usuario ingresa al mock MiArgentina y elige una identidad demo. La pantalla muestra quiénes ya votaron consultando /internal/vote-status.
2. El backend guarda la sesión (subject, nombre, email) y expone /auth/session para el frontend.
3. En la UI se solicita un token de elegibilidad mediante /eligibility/issue/session, enviando la dirección Ethereum que recibirá el SBT.
4. El backend valida el id\_token, genera un token firmado con expiración de 2 horas, registra la emisión en Firestore y opcionalmente invoca MiVotoElection.issueToken.

# 4. Flujo de voto y emisión del SBT

1. El usuario selecciona boleta y candidatos y envía /votes/cast con el token de elegibilidad.
2. VotingService valida que la boleta esté abierta, que el token sea válido y que el sujeto no haya votado previamente (hash del subject).
3. Calcula voteHash y receipt, guarda el voto en Firestore, marca el token como consumido y registra el evento de auditoría.
4. En modo real, VoteContractService invoca MiVotoElection.castVote, que acuña un SBT en la wallet del votante y emite el evento VoteCast. El backend devuelve receipt, txHash y sbtTokenId.
5. La verificación puede hacerse desde la UI (recuento, verificación de recibo) o vía ownerOf(tokenId) y balanceOf(wallet) sobre el contrato del SBT.

# 5. Contratos inteligentes

MiVotoSoulboundToken (SBT):

* Extiende ERC-721, pero bloquea transferencias (sólo mint/burn).
* setMinter permite delegar la emisión al contrato de votación.

MiVotoElection:

* issueToken registra hash de token y dirección del votante.
* castVote consume el token, acuña el SBT si aún no existe y marca el receipt.
* Eventos TokenIssued y VoteCast facilitan la auditoría on-chain.

# 6. Configuración clave

* WEB3\_RPC\_URL, WEB3\_CHAIN\_ID, VOTE\_CONTRACT\_ADDR, WEB3\_PRIVATE\_KEY.
* WEB3\_MOCK\_ENABLED=true para modo simulador; false para blockchain real.
* FIREBASE\_\* para conectarse a Firestore real o emulador.
* FRONT\_ORIGIN y CORS configurados en ProdCorsConfig.

# 7. Mock MiArgentina

Características principales:

* Endpoints /oauth/authorize, /oauth/token y /oauth/userinfo.
* 15 usuarios demo; la pantalla de selección consulta /internal/vote-status y marca quién votó.
* Genera id\_token con formato stub-id-token.<payload base64url> que MiArgentinaTokenVerifier decodifica.

# 8. Persistencia

* Firestore almacena instituciones, candidatos, boletas, votos, elegibilidades, resultados y auditoría.
* Los votos guardan ballotId, institutionId, candidateIds, voteHash, tokenHash, subjectHash, receipt, txHash, sbtTokenId y createdAt.
* SeedService crea datos demo y /seed/default permite resembrar rápidamente.

# 9. Scripts CLI y automatización

* Makefile: login, seed, seed-close, token, token-json, vote, tally, clean-cookies.
* Scripts en /scripts: common, login (LOGIN\_USER), issue-token (WALLET\_ADDRESS), vote, tally, seed.\*.
* Hardhat scripts: deploy.sample.js despliega contratos; check-owner.js revisa ownerOf/tokenId.

# 10. Pruebas y despliegue rápido

1. Levantar `npx hardhat node` para red local.
2. Ejecutar `npx hardhat run --network localhost contracts/deploy.sample.js` para desplegar SBT + Election.
3. Actualizar docker/env/backend.env y reiniciar `docker compose up --build`.
4. Iniciar sesión con un usuario disponible, ingresar wallet, pedir token y votar.
5. Verificar con `/votes/{receipt}/verify` y `npx hardhat run scripts/check-owner.js`.

# 11. Consideraciones de seguridad

* Un voto por subject: VotingService bloquea duplicados con subjectHash.
* Los tokens de elegibilidad expiran y se marcan como consumidos tras el voto.
* CORS habilitado únicamente para FRONT\_ORIGIN.
* En producción conviene restringir /internal/vote-status.

# 12. Próximos pasos sugeridos

* Integrar MiArgentina y Firestore reales con políticas de seguridad.
* Aplicar reglas de Firestore y control de acceso por rol.
* Automatizar pruebas end-to-end y scripts de despliegue a testnet.
* Extender auditoría y observabilidad (Actuator, métricas, logs estructurados).