

Sistema de Votación – ingeniería Web3

Documentación 2025

Integrantes:

- Quinteros Rios Mateo
- Henry Ricardo Landivar Osuna

Ingeniero Docente:

- Ing José Miguel Alvarez Camacho

El sistema de votación son 4 sistemas interconectados entre sí para su respectivo funcionamiento, usando Django para 3 de ellos y DotNet para uno y usando React para el Fronted.

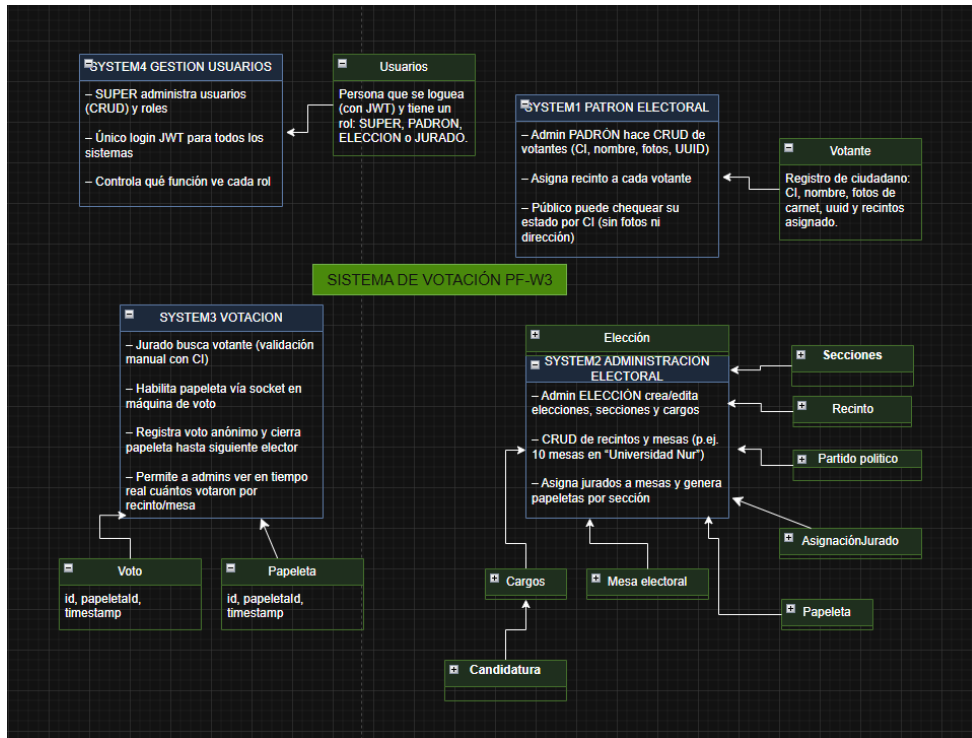
Un panorama unificado de los cuatro microservicios, sus responsabilidades, flujos principales y contratos de API. El proyecto está terminado y listo para producción.

Tabla de Contenidos

1. Arquitectura General
2. Sistema 1: Padrón Electoral
3. Sistema 2: Administración Electoral
4. Sistema 4: Gestión de Usuarios y Autenticación
5. Sistema 3: Votación en Tiempo Real
6. Flujos de Proceso
7. Resumen de Endpoints Clave
8. Modelos de Datos Principales
9. Lógica de Asignación y Generación
10. Integraciones Entre Sistemas

Arquitectura General

- Cuatro microservicios independientes, desplegados en puertos distintos.
- Comunicación HTTP/REST para consultas y WebSockets (Django Channels + Redis) para notificaciones.
- Autenticación centralizada con JWT en Sistema 4.
- Cada servicio se enfoca en un dominio concreto y expone APIs bien definidas.



Sistema 1: Padrón Electoral

Responsable de almacenar, autenticar y exponer datos de votantes.

Features

- CRUD de votantes
- Campos: id (UUID), ci, nombreCompleto, dirección, lat/Ing, fotoCI, fotoRostro
- Asignación automática de mesa por apellido paterno
- Integración a Sistema 2 al crear votante
- Endpoint público “lookup por CI”:
GET /api/Votantes/publico/{ci}

Respuesta:

```
{ "id": "...-uuid", "ci": "1234567", "nombreCompleto": "Ana Gómez", "recintoId": 9, "nombreRecinto": "Centro Comunitario", "mesaNumero": 1 }
```

Sistema 2: Administración Electoral

Gestiona elecciones, cargos, candidaturas, recintos, mesas y jurados.

Elecciones & Secciones

- CRUD de elecciones con tipo, fecha
- Secciones geográficas (polígonos) y herencia lógica (contenedora/hija)

Cargos, Candidaturas & Partido Político

- CRUD de cargos, candidaturas y partido político
- Asociación: Partido político ↔ Candidatura ↔ Cargo ↔ Sección ↔ Elección
- Generación automática de papeletas por sección

Recintos & Mesas

- CRUD de recintos (lat/lng, sección)
- CRUD de mesas dentro de recintos
- Round-robin de votantes en mesas por apellido paterno

Jurados

- Bulk-create en Sistema 4
- Modelo JuradoMesa(participante_id, mesa_id, user4_id, username)
- Orquestador:
POST /system2/api/admin/orquestar-jurados/?eleccion={id}&por_mesa={n}

Sistema 3: Votación en Tiempo Real

Flujos de interacción jurado ↔ votante ↔ resultados.

1. Validación de Identidad

GET /papeleta/buscar/?ci={ci}

- Proxy a Sistema 1
- Responde:

```
{ "id": "...-uuid", "ci": "1234567", "nombreCompleto": "Ana Gómez", "recintoId": 9, "nombreRecinto": "...", "mesaNumero": 1 }
```

2. Consulta de Estado del Jurado

GET /papeleta/jurado-estado/

- Autenticado como JWT-Jurado
- Llama a SIS2 /admin/jurados/?user4_id={request.user.id}
- Filtra votantes de su mesa, excluye al jurado
- Devuelve conteo y lista de votantes disponibles, listo para trabajar

3. Habilitación de Papeleta

POST /papeleta/habilitar/

- Body: { "votante_id": "...-uuid" }
- Registra habilitada=True, habilitada_por=user4_id, habilitada_en
- Emite evento WebSocket /ws/papeleta/

4. Registro del Voto

POST /voto/votar/

- Body: { "papeleta": "...-uuid", "candidatura_id": n }
- Marca votada_en, cierra papeleta
- Emite evento WebSocket /ws/conteo/

5. Resultados en Vivo

- GET /voto/resultados/?eleccion={id}&seccion={id} → Da los todos los resultados de la elección.
- Socket /ws/conteo/ → Notifica nuevos totales
- Front-end React muestra gráfico de torta y valores en tiempo real

Flujos de Proceso

1. **Registro de votantes** (SIS1 → SIS2)
2. **Orquestación de jurados** (SIS2 → SIS4 → SIS2)
3. **Login de jurado** (SIS4)
4. **Validación manual**
 - Jurado ingresa CI → /buscar → compara con documento físico
5. **Habilitación** → /habilitar → socket a cabina
6. **Votación** → /votar → socket a panel público
7. **Resultados** → /resultados + socket a gráfico

Sistema 4: Gestión de Usuarios y Autenticación

Maneja roles y tokens JWT para todos los servicios.

- Modelo CustomUser: id, username, password, first_name, last_name, email, role
- POST /login/ → Token JWT con payload {user_id, role}
- POST /refresh/ → Token JWT refresh para actualizar access
- GET /users/me/ → Datos del usuario autenticado
- Bulk create de jurados:
POST /users/bulk_create/ recibe array de {participant_id, base_username, ...}, devuelve mapeos {participant_id, user_id, username}
- Protección de rutas por rol

Resumen de Endpoints Clave

(no abarca todos los endpoints, para ello usar el archivo adjunto con POSTMAN para cargar todo el resto)

Servicio	Método	Ruta	Descripción
SIS1	GET	/api/Votantes/publico/{ci}	Lookup de votante por CI
SIS2	POST	/admin/orquestar-jurados/?eleccion=&por_mesa=	Bulk-create y mapeo de jurados
SIS4	POST	/login/	Autenticación y JWT
SIS4	POST	/users/bulk_create/	Creación masiva de usuarios-jurados
SIS3	GET	/papeleta/buscar/?ci=	Datos de votante y estado de papeleta
SIS3	GET	/papeleta/jurado-estado/	Lista de votantes asignados al jurado
SIS3	POST	/papeleta/habilitar/	Habilita papeleta y emite socket
SIS3	POST	/voto/votar/	Registra voto y emite socket de conteo
SIS3	GET	/voto/resultados/?eleccion=	Totales y porcentajes por candidatura
All	WS	/ws/papeleta/, /ws/conteo/	Canales Redis para notificaciones en tiempo real

Modelos de Datos Principales

Modelo	Campos Clave
Votante	id(UUID), ci, nombreCompleto, recintoId, mesaNumero, lat, lng, FotoCIAnverso, FotoCIRverso, FotoVotante
Elección	id, tipo, fecha, secciones
Sección	id, polígono GeoJSON
Cargo	id, nombre, elección, seccion
Candidatura	id, cargo_id, ci, nombre, foto_url, color
Recinto	id, lat, lng, sección
MesaElectoral	id, recinto_id, número, elección
JuradoMesa	id, participante_id(UUID), mesa_id, user4_id, username
Papeleta	id(UUID), votante_id, mesa_id, habilitada, habilitada_por, habilitada_en, votada_en
Voto	id, papeleta_id(UUID), candidatura_id, votado_en

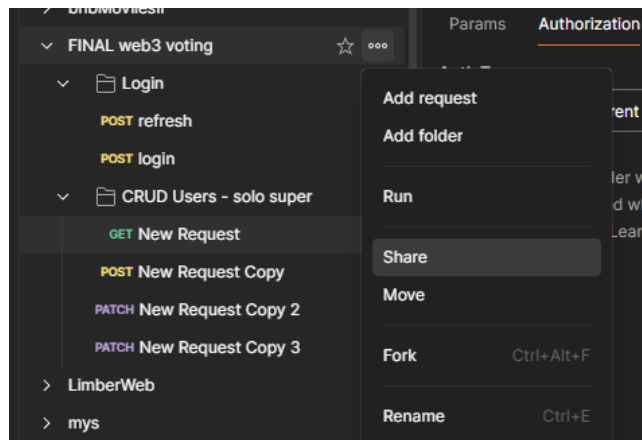
Lógica de Asignación y Generación

- **Mesas:** round-robin por apellido paterno
- **Papeletas:** bulk-create al consultar/crear con lista de candidaturas
- **Jurados:** asignados equitativamente, mapeados con System 4

Integraciones Entre Sistemas

1. **SIS1** → **SIS2**: al crear votante, se notifica su id, ci, recinto y mesa.
2. **SIS2** ↔ **SIS4**: bulk-create de jurados y mapeo de IDs.
3. **SIS3** ↔ **SIS1/SIS2**: validaciones de identidad y asignaciones en tiempo real.
4. **SIS3 WS**: Redis Channels para actualizar cabina y panel público sin recargas.

ANEXOS



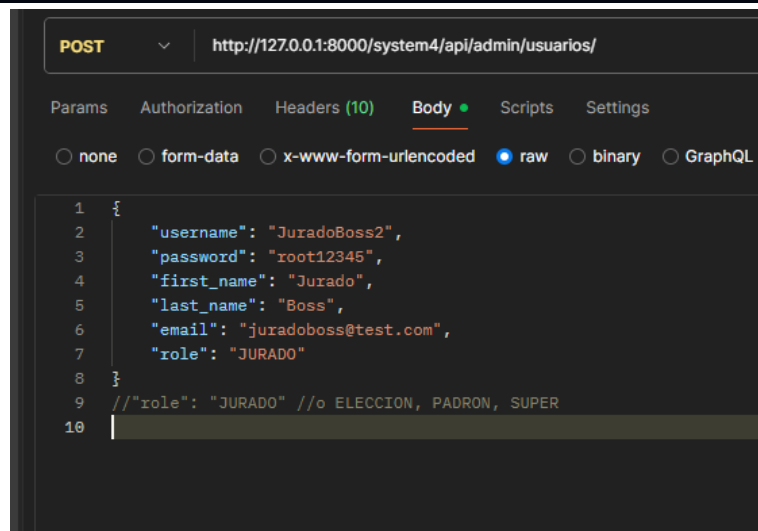
Consulta de Padrón Electoral

Ingresa tu CI

5874345

Buscar

Henry Landivar
CI: 5874345
Recinto Centro Comunitario
Mesa: 2



Sistema de Votación web3 - por Quinteros R. Mateo & Landivar Henry R.

1. Flujo básico que debe implementar en el **FRONTEND**

ENDPOINTS de Backend

POST /system4/api/users/login/ → login con JWT (devuelve access y refresh)

POST /system4/api/users/refresh/ → renovar token

ENDPOINTS de Backend ADMIN:

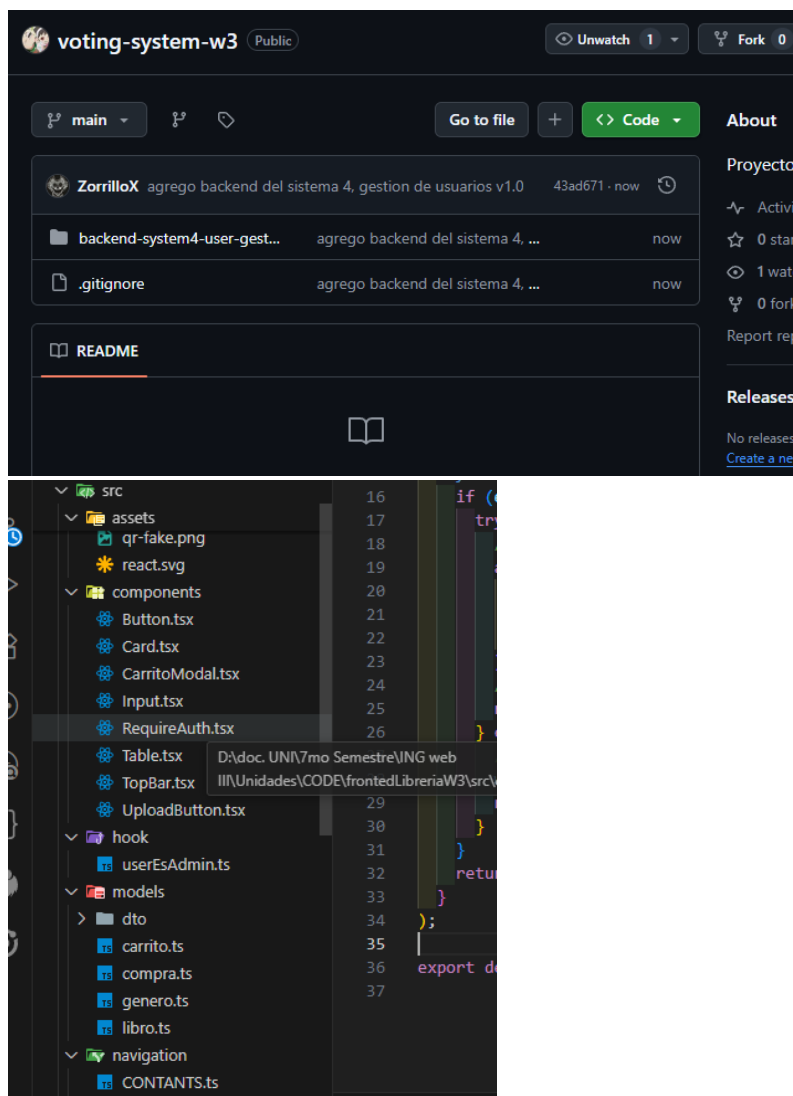
GET /system4/api/users/admin/usuarios/ → trae la lista de usuarios

POST /system4/api/users/admin/usuarios/ → crear un usuario nuevo

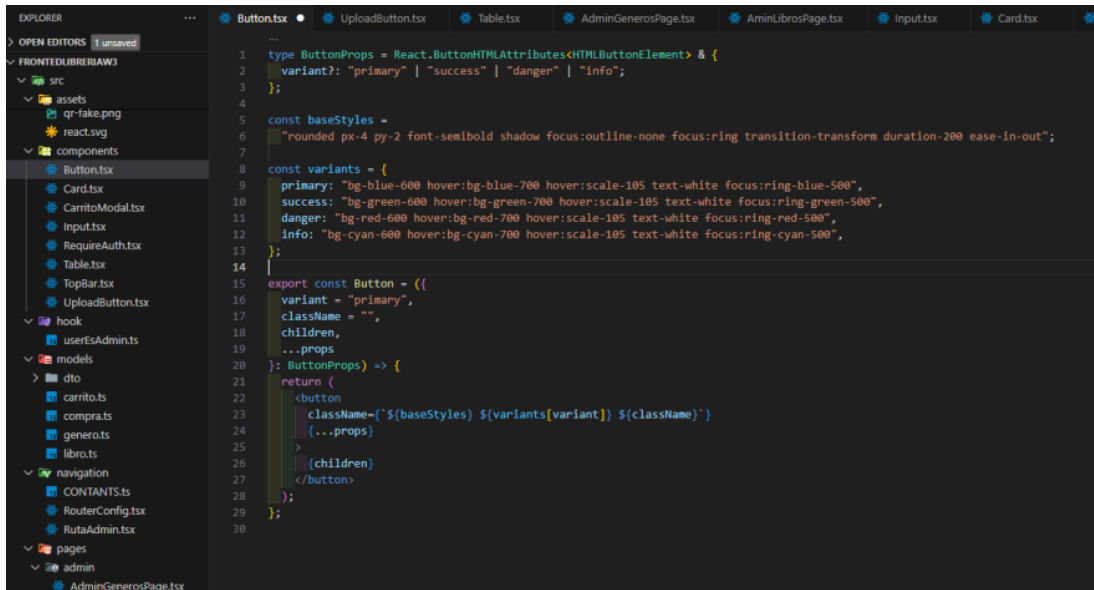
contenido del body:

```
{
  "password": "Joradobase",
  "password": "joradobase",
  "first_name": "Jorado",
  "last_name": "Jorad",
  "email": "joradobase@ceat.com",
  "role": "JURADO"
}
```

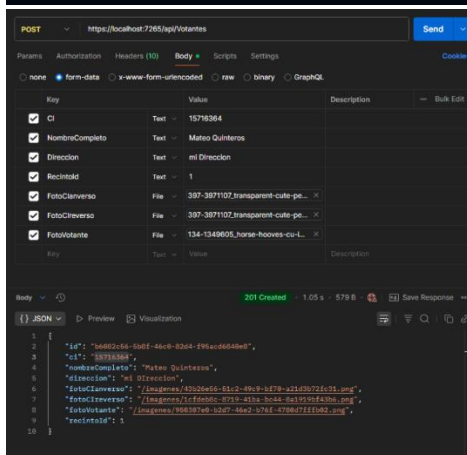
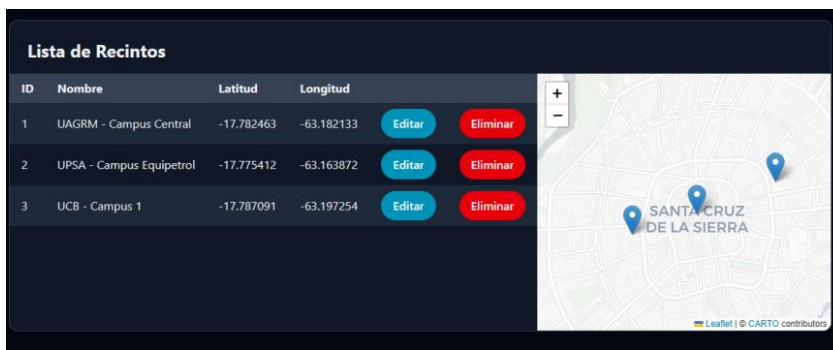
DELETE, PUT & PATCH /system4/api/users/admin/usuarios/{id}/ → Eliminar y Editar, si editas con put los campos de password y role son obligatorios



Sistema de Votación web3 - por Quinteros R. Mateo & Landivar Henry R.



```
1 type ButtonProps = React.ButtonHTMLAttributes<HTMLButtonElement> & {
2   variant?: "primary" | "success" | "danger" | "info";
3 };
4
5 const baseStyles =
6   "rounded px-4 py-2 font-semibold shadow focus:outline-none focus:ring transition-transform duration-200 ease-in-out";
7
8 const variants = {
9   primary: "bg-blue-600 hover:bg-blue-700 hover:scale-105 text-white focus:ring-blue-500",
10  success: "bg-green-600 hover:bg-green-700 hover:scale-105 text-white focus:ring-green-500",
11  danger: "bg-red-600 hover:bg-red-700 hover:scale-105 text-white focus:ring-red-500",
12  info: "bg-cyan-600 hover:bg-cyan-700 hover:scale-105 text-white focus:ring-cyan-500",
13 };
14
15 export const Button = ({
16   variant = "primary",
17   className = "",
18   children,
19   ...props
20 }: ButtonProps) => {
21   return (
22     <button
23       className={` ${baseStyles} ${variants[variant]} ${className}`}
24       {...props}
25     >
26       {children}
27     </button>
28   );
29 };
30
```





```
>>> Mesas disponibles en registro: [1, 2, 3]
>>> Count mesas: 3
Mesa 1: ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
Mesa 2: ['J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R']
Mesa 3: ['S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
[09/Jul/2025 14:40:35] "POST /system2/api/admin/registrar_vota
```