



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

TALLER DE PROYECTO 2 INGENIERÍA EN COMPUTACIÓN

G3 – SmartPoll: Sistema de votación segura y transparente con blockchain

Informe de Avance 2

Grupo de trabajo:

- Blasco, Gonzalo Gabino - Legajo N° 03282/6
- Cabral, Ramiro Nicolás - Legajo N° 03226/6
- Polanis, Iván Valentín - Legajo N° 03266/5

Docente responsable: Gastón Maron



FACULTAD DE INFORMÁTICA

Índice General

1. Descripción General del Proyecto.....	5
1.a. Propuesta original.....	5
1.b. Propuestas o modificaciones a partir de evaluaciones de la cátedra.....	5
1.c. Decisiones propias o complementarias.....	6
2. Materiales y Presupuesto.....	7
Tabla 1: Materiales utilizados en SmartPoll.....	7
3. Esquema Gráfico del Proyecto.....	8
Figura 1: Flujo de funcionamiento de Smartpoll.....	8
4. Grado de Avance, Problemas y Soluciones.....	11
Figura 2: Calendario de avance del proyecto.....	12
4.1. Desarrollo Backend.....	13
Tecnologías utilizadas.....	13
Manejo de datos del usuario.....	13
Errores y soluciones.....	13
Comunicación en tiempo real.....	14
4.2. Desarrollo frontend y webserver.....	15
Frontend.....	15
Tecnologías utilizadas.....	15
4.3. Webserver.....	16
Tecnologías utilizadas.....	16
Comunicación con la interfaz de votación.....	16
Desarrollo de Arquitectura Cloud y Despliegue.....	17
4.4. Arquitectura Cloud.....	17
Figura 3: Arquitectura en la nube.....	18
4.5. CI/CD – Integración y Despliegue Continuo.....	18
4.6. Terraform – Infraestructura como Código.....	19
4.7. Desarrollo del sistema de lectura de QR-Pase.....	19
4.8. Desarrollo de blockchain permisionada.....	20
Arquitectura e implementación.....	20
Configuración y despliegue.....	21
Figura 4: Diagrama de componentes de la red.....	22
Pruebas y resultados obtenidos.....	22



FACULTAD DE INFORMÁTICA

Justificación de decisiones técnicas.....	23
4.9. Interfaz de Auditoría.....	24
4.10. Integración del lector de QR con la interfaz de votación.....	24
5. Documentación Relacionada.....	26
5.1 Hardware construido.....	26
Figura 5: Materiales utilizados para la lectura del código QR.....	26
5.2 Interfaz de Escaneo de QR.....	27
Figura 6: Interfaz Web del inicio de sesión de SmartPoll.....	27
Figura 7: Interfaz Web QR-Pase.....	28
Figura 8: Interfaz Web QR-Pase escaneado con éxito.....	29
5.3 Interfaz de votación.....	30
Figura 9: Interfaz de votación de SmartPoll en estado de espera.....	30
Figura 10: Interfaz de votación de SmartPoll con los candidatos.....	31
5.4 Interfaz de auditoría.....	31
Figura 11: Interfaz de auditoría de SmartPoll.....	32
5.5 Bibliografía.....	32



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

Índice de Figuras

Figura 1: Flujo de funcionamiento de SmartPoll.....	8
Figura 2: Calendario de avance del proyecto.....	12
Figura 3: Arquitectura en la nube.....	18
Figura 4: Diagrama de componentes de la red.....	22
Figura 5: Materiales utilizados para la lectura del código QR.....	26
Figura 6: Interfaz Web del inicio de sesión de SmartPoll.....	27
Figura 7: Interfaz Web QR-Pase.....	28
Figura 8: Interfaz Web QR-Pase escaneado con éxito.....	29
Figura 9: Interfaz de votación de SmartPoll en estado de espera.....	30
Figura 10: Interfaz de votación de SmartPoll con los candidatos.....	31
Figura 11: Interfaz de auditoría de SmartPoll.....	32

Índice de Tablas

Tabla 1: Materiales utilizados en SmartPoll.....	7
---	---



FACULTAD DE INFORMÁTICA

1. Descripción General del Proyecto

El proyecto **SmartPoll** propone el desarrollo de un sistema de votación electrónica basado en blockchain, con el objetivo de superar las limitaciones de los métodos tradicionales y de otras tecnologías previas de voto electrónico. El enfoque principal es garantizar transparencia, verificabilidad y confianza en el proceso electoral, minimizando riesgos de fraude y preservando la privacidad del votante.

La propuesta se centra en implementar un flujo de votación controlado mediante pases de acceso digitales. Cada votante recibe un código QR único, aleatorio y firmado, que valida su derecho a participar. Una vez autenticado, el sistema emite un token anónimo de voto, independiente del QR inicial, que se consume al emitir el sufragio, evitando el doble voto y asegurando la separación entre identidad e intención de voto. El registro final se realiza en una blockchain permissionada, garantizando inmutabilidad, auditoría abierta y conteo verificable.

1.a. Propuesta original

El diseño inicial incluyó:

- Generación de códigos QR únicos asociados al DNI para autenticar votantes.
- Validación de dichos QR mediante un sistema de ingreso seguro.
- Emisión de un token de voto anónimo tras la validación.
- Registro de cada sufragio como transacción en una blockchain permissionada.
- Conteo automático, público y verificable de los resultados.
- Separación estricta entre la identidad del votante y el contenido del voto.

Se definieron como **objetivos primarios** el desarrollo de la interfaz web, el mecanismo de validación, la interfaz de sufragio y la transparencia del proceso; y como **objetivos secundarios** la posibilidad de soportar múltiples votaciones y el padrón electoral.

1.b. Propuestas o modificaciones a partir de evaluaciones de la cátedra

Hasta el momento, la **propuesta original se mantiene intacta**, sin cambios solicitados por la cátedra en esta primera etapa.



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

1.c. Decisiones propias o complementarias

Durante el desarrollo se modificó el flujo original con el fin de reforzar la seguridad y simplificar la experiencia del votante:

- Se resolvió que el votante solo pueda presentarse con un dispositivo móvil con conexión a internet. Esta decisión elimina el uso de QR impresos, reduciendo riesgos de pérdida, duplicación o falsificación, y asegura que toda la comunicación del pase se realice en forma digital.
- Se decidió que el Token Anónimo de Voto (TAV) no sea entregado al votante, sino generado directamente en la estación de votación. De esta manera, se evita que el votante transporte el TAV y se garantiza que solo se utilice dentro del entorno controlado de la mesa. En este nuevo esquema, es la estación de votación la que habilita al dispositivo de la mesa de ingreso para que se pueda escanear un QR. Una vez escaneado y validado el pase, el dispositivo de ingreso notifica a la estación de votación que se autorice una nueva votación y, en ese momento, se genera el TAV en la propia estación.
- Se decidió realizar el despliegue completo de la plataforma en la nube, utilizando Amazon Web Services (AWS) como entorno de producción. Esta decisión tuvo como finalidad asegurar la disponibilidad, estabilidad y coherencia entre los distintos módulos del sistema, permitiendo que tanto el backend como el frontend funcionen de manera integrada bajo condiciones reales de operación.



FACULTAD DE INFORMÁTICA

2. Materiales y Presupuesto

En la **Tabla 1**, se presentan las listas de materiales que serán utilizados para llevar a cabo el proyecto, en ella se detallan: la cantidad, su ubicación y costo unitario.

Componente	Cantidad	Ubicación / Uso	Costo unitario (USD)
Raspberry Pi 3 Model B	2	Mesa de ingreso y cuarto oscuro	\$72.00
Cámara USB Logitech C170	1	Mesa de ingreso (lectura de QR-Pase)	\$36.00
Pantalla táctil para Raspberry Pi	1	Cuarto oscuro	\$42.00

Tabla 1: Materiales utilizados en SmartPoll.

Todos los materiales que se observan en la **Tabla 1** ya se encuentran disponibles para ser utilizados en el proyecto. Si bien para el proyecto se requiere de dos Raspberry Pi, solo se dispone de una puesto que la interfaz de votación se realizará con una notebook. No se requerirá hardware adicional con fondos de la cátedra.

Además fue necesario contratar servicios en la nube para alojar distintas aplicaciones desarrolladas. La mayoría de servicios utilizados están cubiertos por el *Free-Tier* de Amazon Web Services (AWS), por lo que no representan un costo adicional. Solo poseemos un costo bajo demanda de los siguientes recursos:

- **IPv4 pública:** \$0.005 por hora.
- **Instancia t4g.micro de EC2:** \$0.0084 por hora.

FACULTAD DE INFORMÁTICA

3. Esquema Gráfico del Proyecto

En la **Figura 1** se representa gráficamente el flujo de SmartPoll.

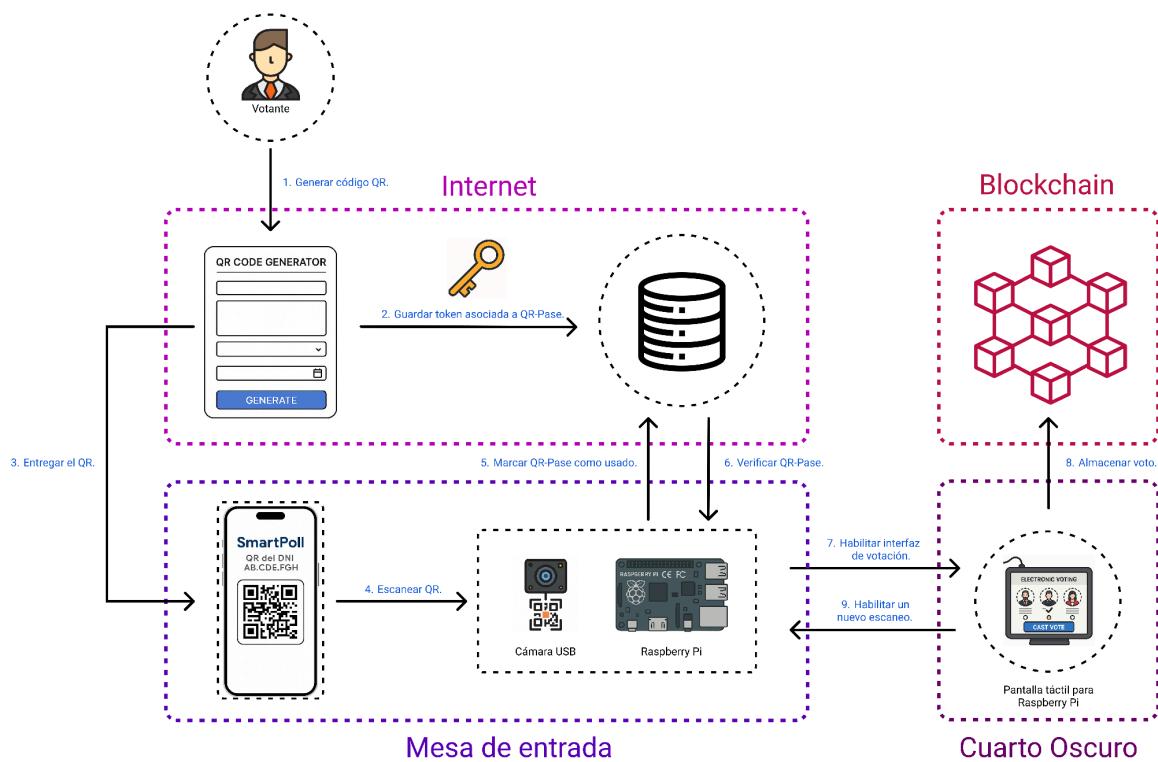


Figura 1: Flujo de funcionamiento de SmartPoll.

A continuación, se detalla cada una de las etapas del flujo representado en la **Figura 1**, describiendo las acciones que realiza el votante y el hardware involucrado en cada paso.

Antes de iniciar el proceso, el votante debe registrarse en la plataforma web, ingresando correo electrónico, DNI y contraseña.

1. Generación del QR-Pase (Plataforma Web + Servidor en la nube):

Una vez validado el registro, el sistema genera un token aleatorio y lo asocia a un QR-Pase único y firmado, vinculado al DNI del votante. El QR se muestra en la plataforma para que el votante pueda presentarlo en la mesa de ingreso.



FACULTAD DE INFORMÁTICA

2. **Almacenamiento del token asociado al QR-Pase (Base de datos en la nube):**
El token generado se guarda en la base de datos en la nube, quedando listo para ser validado en el momento del sufragio.
3. **Presentación del QR-Pase (Dispositivo del votante):**
El votante presenta el QR-Pase en su dispositivo en la mesa de entrada para su verificación.
4. **Escaneo del QR-Pase (Raspberry Pi con cámara USB):**
Una Raspberry Pi equipada con cámara USB escanea el QR-Pase presentado por el votante.
5. **Marcado del QR-Pase como utilizado (Raspberry Pi + Servidor en la nube):**
La Raspberry Pi valida contra la base de datos si el QR-Pase es válido y no ha sido consumido.
 - Si es válido, el sistema lo marca como utilizado.
 - Si no es válido o ya fue consumido, no se permite el acceso al cuarto oscuro.
6. **Respuesta del servidor (Servidor + SSE activo):**
A través de la conexión SSE, el servidor notifica al cliente que el QR-Pase ya fue consumido. El dispositivo del votante muestra que está habilitado para ingresar al cuarto oscuro y la conexión se cierra.
7. **Habilitar Interfaz de votación: (Raspberry Pi en cuarto oscuro):**
Cuando la Raspberry de la mesa de entrada notifica el ingreso de un votante, la Raspberry del cuarto oscuro genera y firma el Token Anónimo de Votación (TAV), habilitando la interfaz de votación. El votante emite su voto, que se envía junto con el TAV firmado al servidor para su validación y registro.
8. **Almacenamiento del TAV en la blockchain permisionada (Servidor + Red blockchain):**
El TAV se registra en la blockchain permisionada, asegurando inmutabilidad, trazabilidad y conteo verificable.
9. **Habilitar un nuevo escaneo (Raspberry Pi en cuarto oscuro):**
Una vez finalizada una votación, la Raspberry Pi del cuarto oscuro se comunica con la de la mesa de entrada para permitir un nuevo escaneo de QR.



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

El esquema de conexiones físicas del sistema SmartPoll se caracteriza por su sencillez y modularidad, dado que solo intervienen dos unidades de Raspberry Pi 3 Model B, cada una con un propósito bien definido:

- Raspberry Pi de Mesa de Ingreso: conectada a una cámara USB Logitech C170, encargada de la lectura del QR-Pase presentado por el votante.
- Raspberry Pi del Cuarto Oscuro: conectada a una pantalla táctil, utilizada para la interfaz de votación y emisión del sufragio.

Ambas placas se alimentan mediante un adaptador de 5V y 3A, y se comunican con el servidor alojado en la nube mediante conexión Ethernet o Wi-Fi, dependiendo de la disponibilidad de red.



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

4. Grado de Avance, Problemas y Soluciones

De acuerdo con lo planificado en el Plan de Proyecto, para la fecha de entrega del Informe de Avance de Octubre se esperaba haber concluido y documentado en la bitácora los siguientes componentes:

- **Interfaz Web:** desarrollo de la plataforma base de interacción.
- **Interfaz de Registro y Escaneo de QR:** implementación de la funcionalidad destinada a validar el acceso mediante la lectura de un código QR.
- **Interfaz de Votación:** diseño y puesta en marcha de la interfaz de usuario destinada a la emisión del voto.

Al momento de la entrega del presente informe, se verificó que las tareas previstas se cumplieron en los plazos acordados, quedando cada una de ellas registradas en la bitácora y debidamente documentadas. En este sentido, el avance planificado para Octubre se corresponde con lo efectivamente alcanzado.

En cuanto al hardware, se realizaron pruebas sobre el **módulo de lectura de QR** y la **comunicación entre el dispositivo móvil de un votante con el backend, y del backend con la Raspberry Pi**.

Se verificaron los siguientes consumos en los dispositivos efectivamente probados:

- **Raspberry Pi**
 - Tensión: 5 V
 - Corriente: 3 A
 - Conector de entrada: Micro-USB
- **Cámara USB (mesa de ingreso)**
 - Tensión: 5 V
 - Corriente: 250–300 mA
 - Alimentación: USB-A

En las próximas etapas se continuará con la integración de los módulos restantes y la validación de la plataforma en condiciones de uso más cercanas a las previstas en el



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

escenario real. Estas actividades han sido organizadas de forma cronológica en el calendario presentado a continuación.

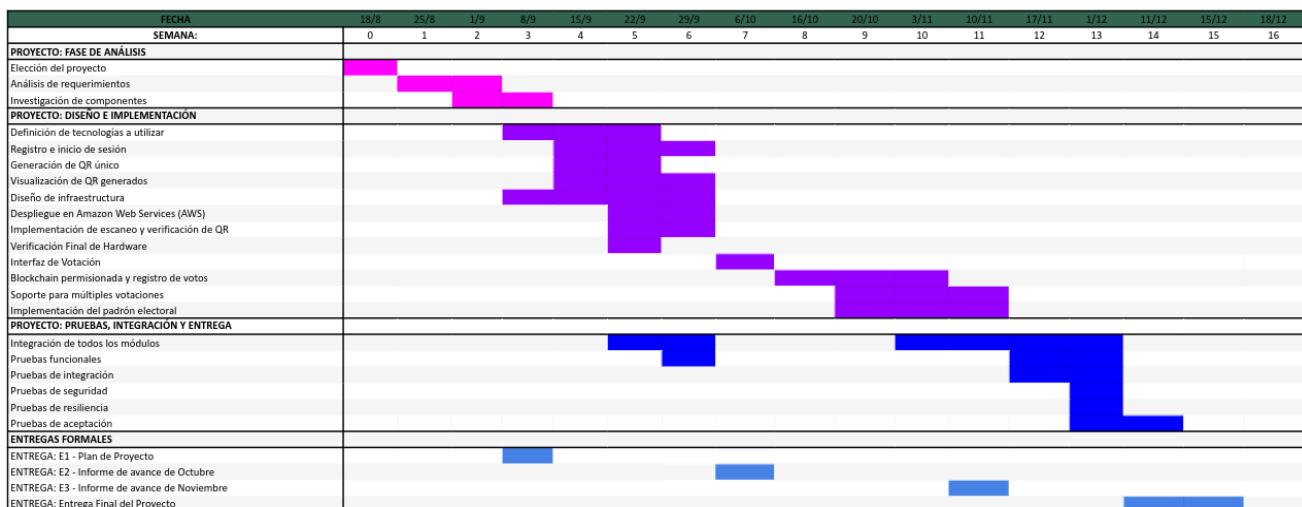


Figura 2: Calendario de avance del proyecto.

En la **Figura 2** se muestra el diagrama de Gantt semanal, donde se señalan tanto las actividades ya completadas como las que restan por realizar. Para una mejor visualización, el diagrama completo puede consultarse en formato ampliado [aquí](#).



FACULTAD DE INFORMÁTICA

4.1. Desarrollo Backend

Tecnologías utilizadas

Para el desarrollo del sistema se emplearon diversas tecnologías que permitieron garantizar la eficiencia, escalabilidad y mantenibilidad del proyecto.

- **Lenguaje y framework:** Se utilizó **Java 21.0.8** junto con **Spring Boot**, dado que Java 21 es la versión LTS más reciente, lo que asegura soporte extendido y actualizaciones de seguridad. Spring Boot, por su parte, simplifica la creación de aplicaciones robustas, permitiendo una estructura modular facilitando la gestión de dependencias y configuraciones. Además, el responsable del desarrollo del backend ya contaba con experiencia en este entorno, lo que aceleró el proceso de implementación y redujo la curva de aprendizaje.
- **Entorno de desarrollo (IDE):** El backend fue desarrollado en **IntelliJ IDEA Ultimate**, un entorno que ofrece integración nativa con proyectos Spring Boot, soporte para testing, manejo de control de versiones y herramientas avanzadas para la gestión de dependencias.
- **Dependencias adicionales:** Para mejorar la calidad del código y reducir tareas repetitivas, se incorporaron las siguientes librerías:
 - **Lombok:** utilizada para eliminar la necesidad de escribir manualmente getters, setters, constructores y otros métodos comunes, lo que mejora la legibilidad del código.
 - **MapStruct:** implementada para el mapeo automático entre entidades y Data Transfer Objects (DTOs), evitando la creación manual de clases de conversión y reduciendo posibles errores humanos.

Manejo de datos del usuario

Para la creación de cuentas, se solicitaron los datos mínimos necesarios: DNI, correo electrónico y contraseña. El DNI se representó mediante el tipo Long, en lugar de un String, con el fin de facilitar la validación numérica y evitar inconsistencias por el formato (con o sin puntos). Esta decisión garantiza una representación uniforme en toda la aplicación y simplifica las validaciones de integridad de datos.

Errores y soluciones

- Necesidad de reiniciar la aplicación tras cada modificación del código en el backend.

La causa es la configuración por defecto de Spring Boot e IntelliJ IDEA sin soporte para recarga automática.



FACULTAD DE INFORMÁTICA

Cómo solución se incorporó la dependencia Spring Boot DevTools en el archivo pom.xml, habilitando el Hot Reload. Luego, se ajustaron las opciones del IDE para permitir la compilación automática incluso con la aplicación en ejecución. Esto mejoró la experiencia de desarrollo y redujo significativamente los tiempos de prueba y ajuste.

- **Dificultad para probar la aplicación en distintos entornos de ejecución (desarrollo, pruebas locales y producción).**

Como solución se definieron perfiles de configuración en Spring Boot:

- **dev**: entorno de desarrollo habitual.
- **local**: pensado para pruebas locales independientes del resto de los servicios.
- **prod**: configurado para el despliegue en la nube (AWS).

De esta forma, cada integrante puede ejecutar el backend con el perfil deseado simplemente configurando la variable de entorno **SPRING_PROFILES_ACTIVE**.

Comunicación en tiempo real

Se desarrollaron las interfaces de usuario para validación de identidad y presentación del QR y la interfaz de votación en el cuarto oscuro.

Para la interacción en tiempo real entre el cliente y el servidor se analizaron principalmente dos alternativas de comunicación:

- **SSE**: permite que el servidor envíe datos de manera continua al cliente a través de una conexión HTTP abierta. Es ideal para aplicaciones en las que la comunicación es principalmente unidireccional, ya que el cliente solo escucha los eventos emitidos por el servidor. Además, es más simple de implementar y mantener.
- **Websockets**: establecen un canal de comunicación **bidireccional** entre cliente y servidor, permitiendo el envío de mensajes en ambos sentidos en tiempo real. Si bien ofrecen mayor flexibilidad, implican una configuración más compleja y un consumo de recursos ligeramente superior.

En base al análisis de las características de ambas comunicaciones se decidió utilizar SSE porque la comunicación requerida en este caso es unidireccional: el servidor es quien envía siempre la información al cliente, transmitiendo eventos como el *payload* para generar el QR o la notificación de que el código fue consumido.



FACULTAD DE INFORMÁTICA

4.2. Desarrollo frontend y webserver

Frontend

Se decidió desarrollar dos aplicaciones web separadas: una para la interfaz de QR y otra para la interfaz de votación. Para mantener coherencia y facilitar el desarrollo, se optó por utilizar el mismo stack tecnológico en ambas, unificando herramientas, librerías y flujos de trabajo.

Tecnologías utilizadas

Para el desarrollo del sistema se emplearon diversas tecnologías que permitieron garantizar eficiencia, escalabilidad y mantenibilidad.

- **Stack tecnológico:** Se utilizó **Vite**, **React**, **TypeScript** y **SWC** en ambas aplicaciones, con Node.js como entorno de ejecución y Docker para asegurar consistencia entre los entornos de desarrollo y producción. Esta elección permitió unificar el stack, facilitar la integración de componentes y simplificar la gestión de dependencias y scripts de desarrollo.
- **Gestión de dependencias y compilación:** Se utilizó **pnpm** como gestor de paquetes, por su eficiencia en espacio y velocidad.
- **UI y maquetado:** Para agilizar el desarrollo de la interfaz y garantizar consistencia visual, se emplearon librerías como **TailwindCSS**, **Radix-UI** y **ShadCn**, permitiendo construir componentes reutilizables y estilizados de manera coherente en ambas aplicaciones.
- **Consistencia del entorno:** Se definió una versión específica de Node.js (22.19 LTS) y se replicó en Docker, asegurando que los entornos de desarrollo y ejecución fueran idénticos y reproducibles.
- **Entorno de desarrollo:** Se utilizó Visual Studio Code como editor principal, aprovechando su ecosistema de extensiones, integración con sistemas de control de versiones y herramientas de depuración.



FACULTAD DE INFORMÁTICA

4.3. Webserver

Se decidió implementar un webserver dedicado para gestionar la comunicación entre la Raspberry de entrada, la Raspberry del cuarto oscuro y la interfaz de votación. Este servidor se encarga de generar y gestionar los **Tokens Anónimos de Votación (TAV)**, notificar al frontend cuando un votante está autorizado y recibir los votos emitidos para su validación temporal antes de subirlos a la blockchain.

Tecnologías utilizadas

Para el desarrollo del webserver se emplearon diversas tecnologías que permitieron garantizar eficiencia, escalabilidad y mantenibilidad.

- **Stack tecnológico:** Se utilizó **Node.js** y **Express** para levantar el webserver, proporcionando un entorno ligero, flexible y ampliamente conocido. Esta elección permitió definir rutas HTTP para la autorización de votantes y la recepción de votos, así como integrar fácilmente un servidor SSE (Server-Sent Events) para notificaciones en tiempo real hacia la interfaz de votación.
- **Gestión de tokens y persistencia temporal:** Para almacenar los TAVs de manera segura y temporal se utilizó **Redis**, configurado con **TTL** para que los tokens caduquen automáticamente si no son utilizados, evitando que un mismo token pueda ser reutilizado.
- **Consistencia del entorno:** Se definió la misma versión de Node.js (22.19 LTS) que en el frontend y se replicó en Docker, asegurando entornos idénticos y reproducibles, facilitando la escalabilidad y el despliegue.
- **Entorno de desarrollo:** Se utilizó Visual Studio Code como editor principal, aprovechando su ecosistema de extensiones y herramientas de depuración para Node.js.

Comunicación con la interfaz de votación

De manera similar a lo evaluado para la comunicación con la interfaz del QR, se determinó que el webserver debía conectarse con la interfaz de votación mediante SSE para enviar actualizaciones en tiempo real y garantizar un flujo eficiente de información. Por otro lado, la emisión de los votos se realiza mediante peticiones HTTP, lo que simplifica la integración y permite mantener un control claro y seguro sobre la validez de cada voto.



FACULTAD DE INFORMÁTICA

Desarrollo de Arquitectura Cloud y Despliegue

Se avanzó en la implementación de la infraestructura completa del sistema de votación, abarcando tanto el despliegue cloud como la automatización del ciclo de integración y entrega y la definición de infraestructura como código. El proyecto alcanzó un estado funcional estable y escalable, con despliegue automatizado en la nube y conexión entre todos los componentes del sistema.

4.4. Arquitectura Cloud

Se diseñó e implementó una arquitectura two-tier sobre AWS, con una subred pública destinada al balanceador de carga y una subred privada que aloja las instancias de aplicación y la base de datos. La arquitectura se replicó en dos *Availability Zones* para garantizar alta disponibilidad y tolerancia a fallos.

La infraestructura se compone de los siguientes servicios principales:

- **ECR:** Registro privado de imágenes Docker integrado con ECS y GitHub Actions.
- **ECS (con EC2):** Orquestación de contenedores con control total del entorno y bajo costo.
- **S3:** Almacenamiento de sitio web estático y archivos públicos.
- **CloudFront:** CDN global para distribución rápida y segura del contenido.
- **ALB (Application Load Balancer):** Balanceo de carga del backend y soporte HTTPS.
- **RDS (PostgreSQL):** Base de datos relacional gestionada en subnets privadas.
- **ACM:** Certificados SSL/TLS gratuitos para el dominio personalizado.
- **IAM:** Gestión de permisos y credenciales seguras para despliegue y CI/CD.

En la **Figura 3** se detalla la integración y conexión de estos servicios en la arquitectura desarrollada en AWS.

FACULTAD DE INFORMÁTICA

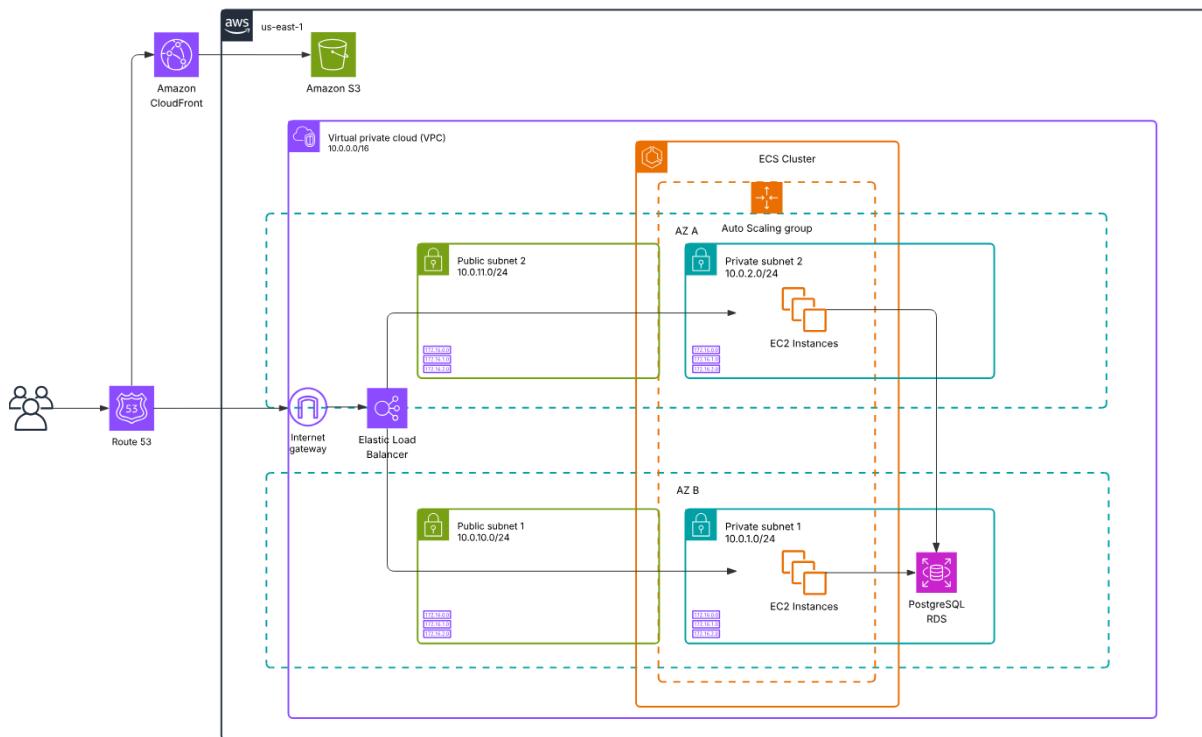


Figura 3: Arquitectura en la nube.

En la **Figura 3** se muestra la arquitectura desarrollada en AWS, compuesta por una parte de acceso público que recibe las requests de los clientes, y una parte privada en la que se alojan las instancias de EC2 y la base de datos PostgreSQL. También, se incluye el Bucket de S3 y la distribución de CloudFront para servir el contenido con mínima latencia a los usuarios. La misma puede encontrarse en mejor calidad [aquí](#).

Además, se utilizó un dominio personalizado adquirido mediante el GitHub Student Pack en get.tech. La validación del dominio se realizó con registros CNAME, y las conexiones HTTP se redirigen automáticamente a HTTPS en ambos extremos (CloudFront y ALB).

4.5. CI/CD – Integración y Despliegue Continuo



FACULTAD DE INFORMÁTICA

Para automatizar el proceso de despliegue, se configuraron pipelines mediante GitHub Actions, tanto para el frontend como para el backend.

Para el **Frontend**, se realiza el build del proyecto utilizando el comando `npm run build`, luego se sincronizan los archivos automáticamente con el bucket de S3 preconfigurado. Por último, se invalida la caché de CloudFront para actualizar el contenido al instante.

Para el **Backend**, se buldeea una nueva imagen de Docker y se la almacena en el registro privado de ECR, con un tag correspondiente al hash del commit. Luego, se redefine la tarea de ECS para usar la nueva imagen.

Se aplicó el principio de Least Privilege, utilizando un usuario de IAM dedicado, con permisos estrictamente necesarios para las acciones de despliegue. Las credenciales fueron almacenadas como secrets en GitHub, manteniendo la seguridad del flujo CI/CD.

Este sistema elimina por completo las intervenciones manuales y permite una entrega continua en cada commit a la branch Master.

4.6. Terraform – Infraestructura como Código

La totalidad del entorno cloud se gestionó con Terraform, utilizando un diseño modular para mantener la infraestructura ordenada y fácilmente escalable.

Se implementaron módulos independientes para cada componente: ECR, ECS, S3, ELB, VPC, y más adelante se añadieron RDS, CloudFront y Route 53.

Se añadió un archivo `.gitignore` para evitar el versionado de los state files de Terraform, los cuales contienen información sensible.

El flujo de trabajo de Terraform permite planificar y aplicar cambios con los comandos `terraform plan` y `terraform apply`, otorgando despliegues predecibles y reproducibles.

4.7. Desarrollo del sistema de lectura de QR-Pase

Como complemento del sistema de votación, se desarrolló un módulo físico para la validación de códigos QR.

Se configuró una Raspberry Pi conectada por Ethernet, accedida mediante SSH, y equipada con una cámara USB. Se instalaron las librerías necesarias para el procesamiento de imágenes y la comunicación con la API.



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

El script desarrollado en Python ejecuta un bucle continuo que detecta códigos QR mediante la cámara. Cuando se detecta un código válido, se realiza una solicitud POST al endpoint de la API (<https://api.smartpoll.tech/api/qr/consume/>) con el contenido del código. Este proceso permite verificar la validez del QR en tiempo real, funcionando como una terminal de registro en la mesa de entrada de la votación.

El sistema fue probado con el frontend y backend locales, validando su correcto funcionamiento y su integración con la API.

4.8. Desarrollo de blockchain permisionada

Durante el mes de noviembre se desarrolló la integración entre el servidor web y una red blockchain permisionada implementada con Hyperledger Fabric, con el objetivo de garantizar la trazabilidad, transparencia y auditoría distribuida del proceso de votación en el sistema SmartPoll.

Arquitectura e implementación

La infraestructura está compuesta por dos organizaciones, cada una con un nodo (*peer*) activo dentro de la red. Una de las organizaciones actúa como entidad administradora, responsable de la emisión y conteo de votos, mientras que la segunda representa una entidad auditora, con permisos únicamente de lectura y verificación del *ledger*.

Tanto la red de Hyperledger Fabric como el servidor web se despliegan en contenedores Docker independientes, los cuales se comunican a través de una red interna del mismo entorno. Esta configuración reproduce el esquema que se utilizaría en un entorno distribuido, permitiendo que cada organización mantenga su infraestructura de manera aislada y segura.

El servidor web, desarrollado sobre Express.js, expone endpoints HTTP que permiten:

- Emitir un voto: genera una transacción sobre el contrato inteligente (*chaincode*).
- Consultar resultados o auditorías: accede a la información registrada en el *ledger*.

Para esto se implementó un módulo cliente denominado *VotingClient*, encargado de gestionar la conexión al *gateway* de Fabric. Este módulo maneja la inicialización de los certificados de identidad de la organización, el establecimiento de la conexión gRPC con el *peer* correspondiente y la invocación de las transacciones del contrato inteligente.



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

Por su parte, el contrato inteligente, denominado *VoteContract*, fue implementado en TypeScript y define las principales operaciones del sistema de votación:

- Creación de votos y registro de transacciones.
- Lectura del *ledger* para auditorías.
- Verificación de existencia de votos.
- Conteo total de los votos emitidos.

El contrato incluye un control de permisos que restringe las operaciones de escritura y conteo exclusivamente a la organización administradora, mientras que el resto de las organizaciones pueden mantener nodos participantes con capacidad de lectura. Este enfoque garantiza la integridad del proceso electoral, evitando manipulaciones o registros indebidos en el ledger.

Configuración y despliegue

La red Fabric se ejecuta en el mismo entorno local de desarrollo, utilizando Docker Compose para la orquestación de contenedores. Esto incluye la creación de los nodos *peer*, los *orderers*, las bases de datos de respaldo (*CouchDB*), y los certificados de identidad generados mediante Fabric CA.

Cada organización dispone de sus propios certificados y *Membership Service Provider (MSP)*, lo que permite definir políticas de acceso y autenticación independientes. Los canales de comunicación se definen de forma aislada por elección, lo cual evita la contaminación de datos entre procesos electorales y facilita las auditorías posteriores.

La **Figura 4** representa la estructura de la red blockchain utilizada por SmartPoll, implementada sobre Hyperledger Fabric.

FACULTAD DE INFORMÁTICA

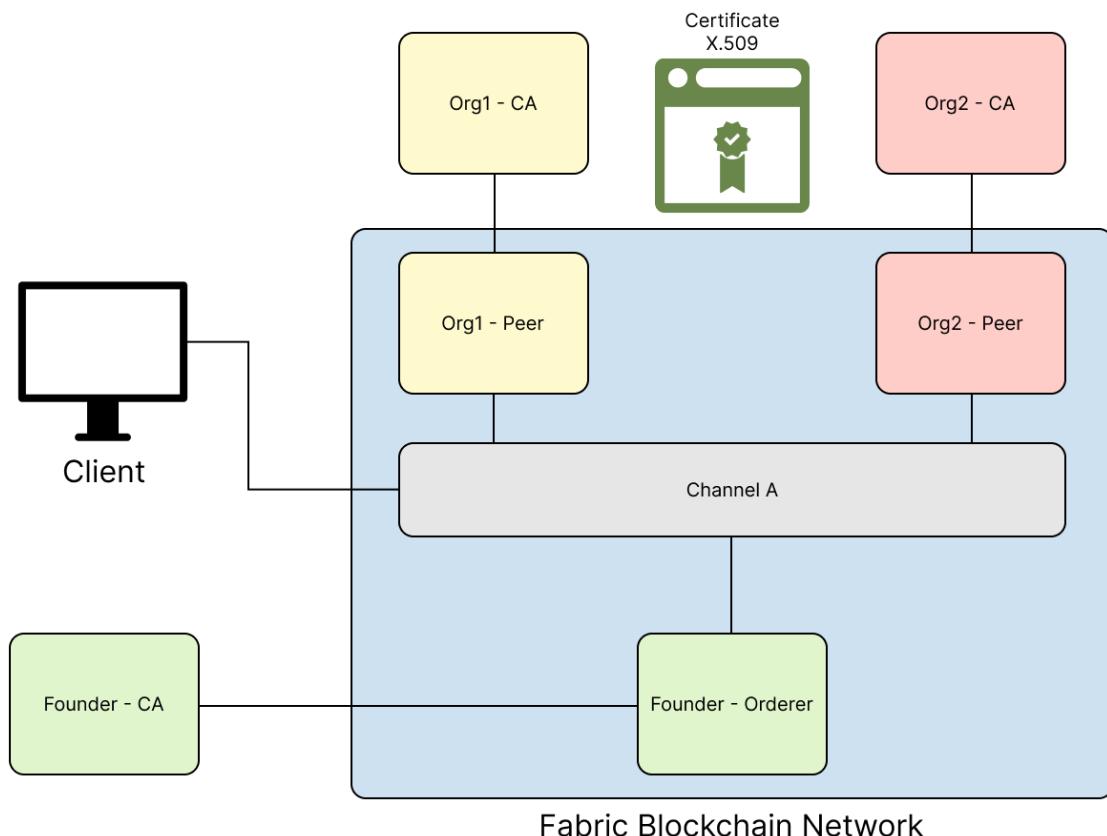


Figura 4: Diagrama de componentes de la red.

La red está compuesta por tres organizaciones: Founder, Org1 y Org2, cada una con su Autoridad Certificadora (CA) y su respectivo Peer, encargados de validar y mantener el estado del libro distribuido.

El Founder además ejecuta el nodo Orderer, responsable de ordenar las transacciones y garantizar la coherencia del canal de comunicación Channel A, donde se registran los votos.

El cliente, que representa la aplicación SmartPoll, interactúa con la red a través de certificados X.509, asegurando autenticidad y trazabilidad en cada operación.

Pruebas y resultados obtenidos



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

Se realizaron pruebas funcionales del flujo completo de votación, verificando el correcto funcionamiento de la interacción entre los distintos componentes del sistema:

1. **Generación y escaneo del código QR** por parte de la Raspberry Pi asociada al proceso de votación.
2. **Validación del Token de Autorización de Voto (TAV)** contra el backend de usuarios.
3. **Emisión del voto** a través del endpoint del servidor Express, que a su vez ejecuta la transacción correspondiente sobre la red Fabric.
4. **Registro y persistencia del voto** en el *ledger* permissionado, verificable por cualquier nodo participante.

Los ensayos confirmaron la correcta trazabilidad de cada voto, la inmutabilidad de los registros y la sincronización consistente entre los peers. Además, se comprobó que el sistema mantiene una latencia estable en la emisión y lectura de transacciones, garantizando una respuesta fluida dentro del entorno de prueba local.

Justificación de decisiones técnicas

- **Separación de contenedores:** permite mantener independencia entre el servidor de aplicación y la infraestructura blockchain, reflejando un diseño modular que facilita el mantenimiento y escalabilidad.
- **Control de acceso en el contrato:** se implementó un sistema de permisos que centraliza la emisión y conteo de votos en la organización administradora, preservando la integridad de los resultados.
- **Uso de canales independientes:** cada elección se ejecuta dentro de su propio canal, limitando el acceso y asegurando la privacidad de los datos asociados a cada proceso electoral.
- **Comunicación local mediante certificados:** aunque el entorno es completamente local, la autenticación mediante certificados y gRPC replica fielmente la lógica de conexión de un entorno distribuido.



FACULTAD DE INFORMÁTICA

4.9. Interfaz de Auditoría

Para el desarrollo de la Interfaz de Auditoría se decidió mantener la coherencia tecnológica con las aplicaciones previas, utilizando el mismo stack y herramientas. Esto permitió asegurar compatibilidad, facilitar la integración y mantener un flujo de trabajo unificado entre las distintas interfaces del sistema.

De este modo, la Interfaz de Auditoría fue desarrollada también con Vite, React y TypeScript, empleando las mismas librerías de UI y gestión de estilos (TailwindCSS, Radix-UI y ShadCn), así como la versión definida de Node.js replicada en Docker. Esta homogeneidad tecnológica facilita el mantenimiento, la escalabilidad y la consistencia visual de toda la plataforma.

4.10. Integración del lector de QR con la interfaz de votación

En esta etapa se incorporó la comunicación bidireccional entre el lector de códigos QR (Raspberry Pi) y la interfaz de votación presente en la red local, permitiendo una coordinación directa entre ambos sistemas.

Anteriormente, el lector se limitaba a detectar un código QR y enviar la información a la API remota de SmartPoll. Ahora, con esta modificación el proceso de interacción se encuentra completamente sincronizado con la terminal de votación.

El flujo operativo es el siguiente:

1. El sistema inicia en estado bloqueado, sin escanear QR hasta recibir una señal de inicio desde la interfaz de votación (**POST /start**).
2. Cuando se habilita el escaneo y se detecta un QR válido:
 - Se realiza el consumo remoto del código en la API de SmartPoll.
 - Se notifica a la interfaz de votación local, iniciando el proceso de votación.
 - El lector se bloquea automáticamente, evitando nuevos escaneos mientras el votante interactúa con la terminal.
3. Al finalizar la votación, la interfaz envía un **POST /resume** al lector, lo que reactiva el escaneo y prepara el sistema para el siguiente usuario.



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

De esta manera, el flujo de control se vuelve completamente bidireccional, garantizando que solo un votante pueda interactuar con el sistema a la vez y eliminando cualquier riesgo de lectura simultánea de códigos.

FACULTAD DE INFORMÁTICA

5. Documentación Relacionada

5.1 Hardware construido

Al ser este proyecto mayoritariamente de Software, de lo físico solo se puede mostrar la Raspberry Pi 3 con cámara USB:



Figura 5: Materiales utilizados para la lectura del código QR.

La **Figura 5** muestra cómo se conecta la misma. Este hardware se encuentra en la mesa de entrada y se destaca la comunicación mediante **conexión USB** entre la **Raspberry Pi** y la **cámara**, utilizada para la lectura del código QR.



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

5.2 Interfaz de Escaneo de QR

La interfaz web es presentada en el dispositivo móvil del votante. En la **Figura 6** se pueden observar la interfaz de inicio de sesión:

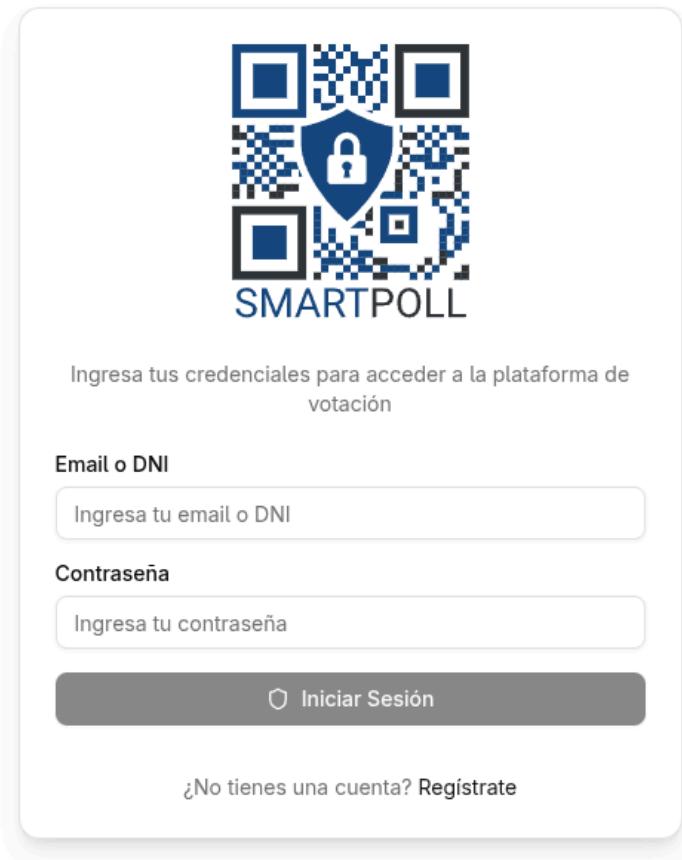


Figura 6: Interfaz Web del inicio de sesión de SmartPoll.

Donde se puede destacar que el usuario puede hacerlo a partir de su Email o DNI, ambos únicos en el sistema, y su contraseña de más de 4 caracteres.



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

Una vez que el usuario inicia la sesión, se muestra en pantalla el QR-Pase, como se puede ver en la **Figura 7**.



Figura 7: Interfaz Web QR-Pase.

Cuando el código es presentado frente a la cámara de la Raspberry Pi, el sistema lo valida y, una vez verificado, se muestra la siguiente interfaz **Figura 8**.



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

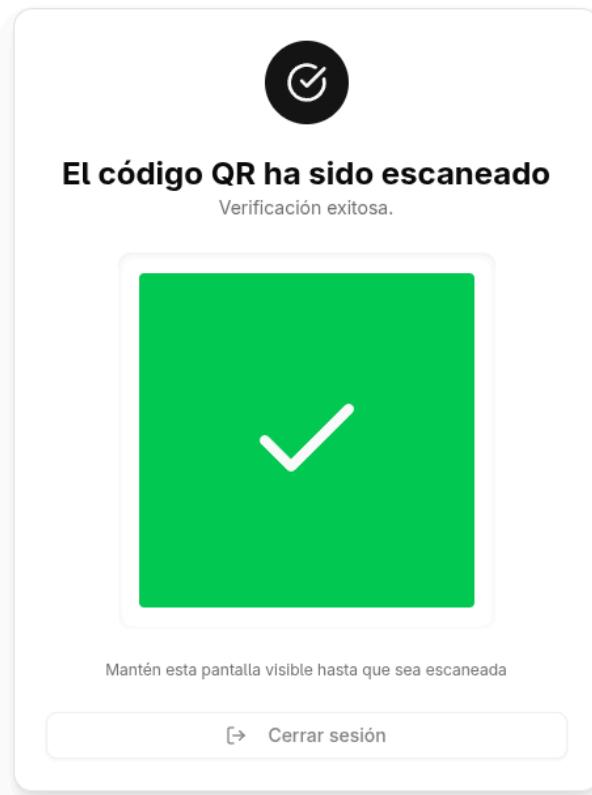


Figura 8: Interfaz Web QR-Pase escaneado con éxito.

En esta se observa que se le indica al usuario que ya puede proceder a emitir su voto.

El funcionamiento completo de SmartPoll en la mesa de entrada se encuentra en el siguiente link: <https://youtu.be/DNYpIzXZ5zw>



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

5.3 Interfaz de votación

En la pantalla táctil de la Raspberry Pi del cuarto oscuro, se presenta la interfaz de votación la cual muestra una pantalla de espera, que indica que todavía no hay un usuario que haya escaneado el QR-Pase para votar, como se muestra en la **Figura 9**.



Figura 9: Interfaz de votación de SmartPoll en estado de espera.

Una vez que el usuario escanea con éxito el QR-Pase, la Raspberry Pi de la mesa de entrada se comunica con la de votación, pasando a la pantalla donde el usuario podrá seleccionar al candidato y votar como se presenta en la **Figura 10**.



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

Smartpoll
Elecciones 2025

Emite tu voto

Seleccione un candidato a continuación para enviar su voto

Martina López
Frente Renovador

Julián Fernández
Unión Federal

Camila Pereyra
Movimiento Popular

Tomás Álvarez
Partido Progreso

Figura 10: Interfaz de votación de SmartPoll con los candidatos.

5.4 Interfaz de auditoría

En la **Figura 11** se puede observar la interfaz de auditoría, la cual permite observar la cantidad de votos por candidato:



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

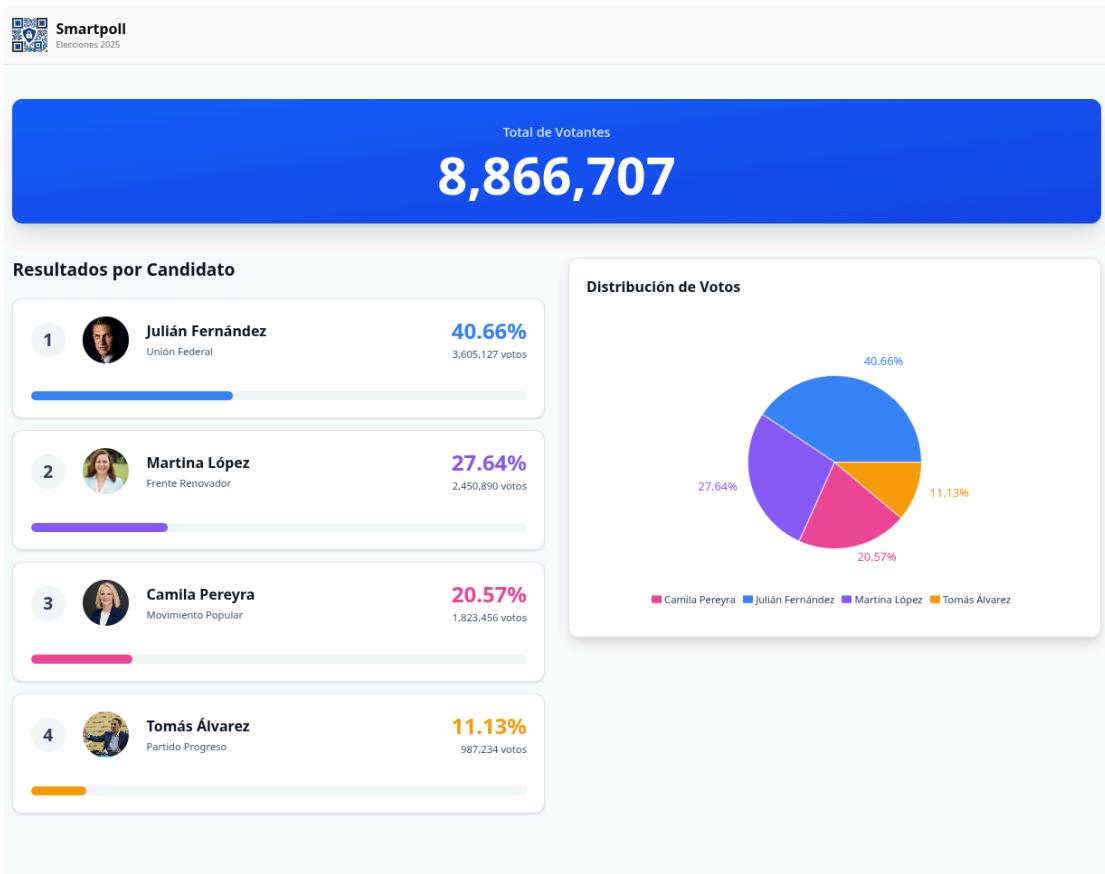


Figura 11: Interfaz de auditoría de SmartPoll.

La interfaz permite visualizar los porcentajes obtenidos por cada candidato. Esta interfaz estará disponible una vez finalizada la votación.

5.5 Bibliografía

Algunos links a recursos utilizados para llevar a cabo el proyecto:

- [Hyperledger Fabric Docs](#)
- [Mapstruct Docs](#)
- [Terraform Docs](#)
- [AWS Docs](#)
- [OpenCV Docs](#)
- [Vite](#)
- [React](#)
- [Shadcn Docs](#)
- [Spring Boot](#)



UNIVERSIDAD
NACIONAL
DE LA PLATA

FACULTAD DE INFORMÁTICA

- [Server-Sent Events](#)
- [Postman Docs](#)
- [Raspberry Pi 3 Model B Datasheet](#)
- [Webcam C170 Guia](#)
- [Raspberry Pi Touch Display 2 Datasheet](#)

Link al código fuente: <https://github.com/tpII/2025-G3-SmartPoll>

Link al video de presentación: <https://youtu.be/ENbJnc9eZks>

Link a la bitácora: <https://github.com/tpII/2025-G3-SmartPoll/wiki/Bit%C3%A1cora>