Objetivo

Construir una plataforma de **votaciones electrónicas** y **control de asistencia** para un contexto universitario, con inicio de sesión sin contraseña, gestión de perfiles, panel de administración, creación y seguimiento de votaciones y reuniones, exportación de datos y personalización visual.

> **Zona horaria por defecto**: Europe/Madrid (Barcelona). Todas las horas se almacenan en UTC y se muestran en Europe/Madrid.
>
> **Exportaciones**: Todos los archivos generados por el sistema (CSV/XLSX/JSON/zip) se guardan bajo la carpeta raíz `` siguiendo la estructura definida en este documento.

Roles y permisos

- **Usuario**: Puede iniciar sesión con email universitario permitido, completar/editar su información personal (según reglas), unirse a reuniones, marcar asistencia con código, votar y consultar resultados (agregados). Ver histórico propio.
- **Administrador**: Todo lo del usuario + crear/gestionar reuniones de asistencia, crear/gestionar votaciones, ver resultados detallados (por persona), gestionar usuarios, solicitudes, ajustes del sistema, exportaciones globales.

Roadmap por fases (alto nivel)

1. **Fase 0 – Fundaciones**: setup del repo, CI/CD, entorno, i18n, diseño de datos, seguridad básica.
2. **Fase 1 – Login sin contraseña (email OTP)**: lista permitida, verificación de dominio, flujo de solicitud de alta si no está en lista.
3. **Fase 2 – Panel principal e Información Personal**: perfiles, validaciones, flujos de aprobación.
4. **Fase 3 – Portal de Asistencia**: creación, unión por código, gestión, cierre, históricos y exportación.
5. **Fase 4 – Portal de Votaciones**: creación, participación, contadores, resultados, cierre.
6. **Fase 5 – Gestor de Usuarios**: altas/bajas, dar/quitar admin, importación por plantilla Excel y gestión de solicitudes.
7. **Fase 6 – Ajustes y Personalización**: dominios, SMTP, reglas de edición de perfil, logo y colores, notificaciones.
8. **Fase 7 – Exportaciones y carpeta **********``**: estructura, formatos, backups.
9. **Fase 8 – Seguridad, auditoría y cumplimiento**: rate limits, logs, RGPD, copias de seguridad.
10. **Fase 9 – Beta, pruebas y mejoras**: QA, accesibilidad, rendimiento.

Cada fase incluye definición funcional, datos, UI, validaciones, casos borde y criterios de aceptación.

Fase 0 – Fundaciones

Tecnología sugerida (puede ajustarse)

- **Backend (Python)**: **FastAPI** (ASGI) + **Uvicorn**. Validación con **Pydantic**. Persistencia **sin BD**, basada en **archivos JSON/JSONL** en `Datos/` (capa repositorio).
- **Frontend**: React + TypeScript (Vite), UI: Tailwind + componentes. WebSockets/EventSource opcional para contadores en vivo.
- **Email**: SMTP con **aiosmtplib** o proveedor (SendGrid, etc.). Plantillas Jinja2.
- **Infra**: Docker Compose (app + opcional worker). Variables en `.env`.
- **Tareas programadas**: **APScheduler** (cron/interval) para cierres automáticos, recordatorios y backups.
- **Serialización**: `orjson` para rendimiento.
- **IDs**: `uuid4` o **ULID** (ordenable por tiempo).
- **Locks de archivo**: `filelock` (o `fcntl` en Linux) para concurrencia segura.
- **Timezones**: `zoneinfo` (stdlib) y utilidades para Europe/Madrid.

Estructura de proyecto (resumen) (resumen)

- `backend/` (API, workers)
- `frontend/` (SPA)
- `Datos/` (creada por la app si no existe)
- `usuarios/`
- `asistencia/`
- `votaciones/`
- `solicitudes/`
- `ajustes/`
- `logs/`
- `backups/`

Reglas de hora/fecha

- Guardar **UTC** en DB. Mostrar en **Europe/Madrid**. Manejar DST.
- Validación de “hora personalizada de cierre” < “ahora”.

\$1

Persistencia en archivos (sin BD)

- **Fuente de verdad**: carpeta `Datos/`.
- **Formatos**:
- **JSONL** (append-only) para eventos y colecciones grandes: `usuarios.jsonl`, `solicitudes.jsonl`, `audit_*.jsonl`, `votos.jsonl`, `asistentes.jsonl`.
- **JSON** para metadatos y agregados: `definicion_votacion.json`, `reunion.json`, `resultados_agregados.json`, `index.json`.
- **Atomicidad**: escribir en `*.tmp` y **rename** atómico → evita corrupciones.
- **Bloqueos**: `filelock` por fichero (p. ej. `Datos/.locks/usuarios.lock`) para escrituras concurrentes.

- **Índices locales** (cacheables): archivos `index.json` por módulo, p. ej. `Datos/usuarios/index.json` (por email y NIU), `Datos/votaciones/{id}/index_votos.json` (por user_id/pregunta_id). Se regeneran bajo demanda si faltan.
- **IDs y rutas**: IDs **UUID/ULID**; rutas deterministas `Datos/{modulo}/{YYYY}/{MM}/...` o por **ID** (carpetas por entidad).
- **Compacción**: tareas de **compactación** periódica (APScheduler) que leen JSONL y escriben un JSONL nuevo sin borrados/duplicados.
- **Backups**: zip incremental/daily de `Datos/` + verificación de integridad (hashes) y almacenamiento de checksums.
- **Integridad**: validación con **Pydantic** al leer/escribir; si un registro es inválido, se mueve a `Datos/_quarantine/` con motivo.

Fase 1 – Login sin contraseña (email universitario)

Requisitos

- **Sin contraseña. Sin registro manual.**
- Campo de email: validar formato y **dominio permitido** (definido en Ajustes), p.ej.: `@dominiouni.com`, `@dominiouni.es`, `@dominiouni.cat`.
- Comparar email contra **lista de permitidos** (puede venir de importación o alta manual).

Flujo

1. Usuario introduce email.
2. **Si dominio no permitido** → error: “Dominio no permitido. Contacte administración.”
3. **Si email NO está en la lista** → mostrar mensaje: “Tu email no está en la lista. Revisa si está bien escrito. ¿Quieres notificar a administradores para que te den de alta?”
4. Si **Sí**: abrir formulario con **Nombre**, **NIU**, **Email universitario**, **Mensaje (opcional)**.
5. Al enviar: crear **Solicitud de alta** (estado: pendiente) + enviar email a administradores + registrar en `Datos/solicitudes/solicitudes_alta.jsonl` y tabla `solicitudes`.
6. Mostrar confirmación al usuario (no crea cuenta hasta ser aprobada).
7. **Si email está en la lista** → enviar **código por email** (OTP). Guardar hash del código, contador de intentos, expiración.
8. Usuario introduce código.
9. **Si correcto** → iniciar sesión. **Si incorrecto** → restar intento. Si expira/bloquea → informar y permitir reenviar (con rate limit).

Mensajería/Emails

- Plantillas HTML para: código OTP, notificación de solicitud de alta.
- Ajustes definen destinatarios admin para notificaciones.

Criterios de aceptación

- No se permite acceso con emails fuera de lista.
- Flujo de solicitud de alta funcional con notificación y registro.

- OTP con expiración, un solo uso, rate limit y bloqueo por intentos.

Fase 2 – Panel principal e Información Personal

Panel principal (post-login)

- Tarjeta **Información personal** (estado: completa/incompleta). Si faltan **Nombre** o **NIU** → forzar navegación a Información Personal antes de cualquier otra acción.
- Botones: **Votaciones activas**, **Portal de asistencia**.
- Si el usuario es **Administrador**: botón **Ajustes** y **Gestor de usuarios**.

Información Personal

- Campos obligatorios globales: **NIU**, **Nombre**, **Email universitario**, **Grupo académico**, **Curso**, **Es admin** (solo visible, no editable por usuario).
- Desde **Ajustes** se define por campo si:
- es **obligatorio**,
- el usuario puede **editar libremente** o **requiere aprobación**.
- Si requiere aprobación: un cambio propuesto crea **Solicitud de modificación** con diff, visible en **Gestor de usuarios → Solicitudes** para aprobar/denegar.
- Historial de cambios por usuario con auditoría.

Validaciones y UX

- NIU único. Email único.
- Auto-rellenar Email desde login (bloqueado).
- Indicador de **completitud** del perfil.

Criterios de aceptación

- Forzado a completar perfil si Name/NIU faltan.
- Cambios sujetos a reglas de aprobación según Ajustes.

Fase 3 – Portal de Asistencia

Vistas de Administrador

1. **Registros pasados**
2. Lista con: título, fecha/hora inicio y cierre, nº asistentes, duración.
3. Detalle: asistentes (presentes/ausentes), hora de unión por usuario.
4. Acciones: **Exportar** (XLSX y CSV) con cabeceras y zona horaria indicada. Exporta a `Datos/asistencia/{YYYY}/{MM}/asistencia_{id}_{slug}.{csv|xlsx}`.
5. **Registros activos**
6. **Crear registro**: campos →
    - Título/nombre de la reunión (obligatorio)
    - Hora de inicio (por defecto **ahora** en Europe/Madrid; editable)

    - Código personalizado (opcional) o **generado aleatorio** (único, longitud configurable, p.ej., 6-8 caracteres alfanum.)
7. Listado de registros activos con **contador** de tiempo transcurrido.
8. Acciones por registro:
    - **Unirse a la reunión** (como admin también marca asistencia)
    - **Gestionar**: editar título, añadir/eliminar usuarios manualmente, ver lista parcial en tiempo real.
    - **Cerrar**: elegir **ahora** o **hora personalizada** (debe ser anterior a ahora). Confirmación antes de cerrar.

Vistas de Usuario (no admin)

- **Reuniones pasadas**: lista completa (sin entrar a detalle). **Verde** si asistió, **rojo** si no asistió.
- **Reuniones activas**: puede **unirse con código**; ve contador de cada una.

Reglas funcionales

- "Unirse a la reunión" solicita **código**; si correcto, registra asistencia del usuario (único por reunión). Reintentos con rate limit.
- Un usuario no puede marcar asistencia dos veces. Si intenta de nuevo, mostrar "Ya constas como asistente".
- Admin puede añadir/eliminar asistentes manualmente desde Gestionar (se registran en auditoría).

Exportes y datos

- XLSX/CSV con:
- ID reunión, Título, Inicio (UTC y Europe/Madrid), Cierre (UTC y Europe/Madrid), Duración.
- Por usuario: NIU, Nombre, Email, Asistencia (Sí/No), Hora de unión.

Criterios de aceptación

- Creación, unión por código, gestión y cierre operativos.
- Listas pasadas/activas correctas según rol.
- Exportes en `Datos/` con estructura.

Fase 4 – Portal de Votaciones

Listas para todos los usuarios

1. **Votaciones activas no votadas**: acceso a votar. **Contador** de tiempo restante.
2. Si faltan $\leq$ **1h**: **countdown en vivo** (segundos).
3. Si faltan **> 1h**: texto relativo ("faltan 2 días").
4. **Votaciones activas ya votadas**: muestran tiempo restante (sin acceso a volver a votar salvo que la votación permita múltiples votos por persona).
5. **Votaciones cerradas**: muestran todas. Si el usuario participó → **"Participado"** en verde. Si no → **"No has participado"** en rojo.
6. Los **usuarios** pueden consultar **resultados agregados**.

7. Los **administradores** ven además **detalle nominal** (quién votó qué y quién no votó).

Creación/gestión de votaciones (Admin)

- **Crear votación**:

- **Inicio**: fecha/hora (vacío = ahora).

- **Cierre**: fecha/hora (**obligatorio**; debe ser > inicio).
- **Título**, **Descripción**.
- **Preguntas**:
  - **Respuesta abierta** (texto).
  - **Selección múltiple**:
  - Opciones definidas por admin.
  - **Límite por persona** (por defecto **1**; configurable a N). Si N > 1, interfaz de checkboxes con contador.
- *(Opcional en Ajustes)*: permitir **edición de voto** hasta el cierre; por defecto **no**.

- *(Opcional)*: **Anonimato**. Por defecto **no anónimo** (admin puede ver quién votó qué). Si se marca **anónimo**, solo se registran agregados y no se expone el vínculo usuario→respuesta (ver implicaciones RGPD y auditoría; requiere diseño adicional si se usa).

- **Gestión**:

- Editar título/descripcion (no fechas si ya inició; si aún no inició, se puede editar inicio/cierre respetando validaciones).

- Cerrar manualmente (confirmación).
- Ver resultados en tiempo real (admin: agregados + nominal si no anónimo).

Reglas de participación

- El usuario solo puede **emitir votos** según **límite** por pregunta.
- **Prevención de votos duplicados** por pregunta según límite.
- Respuestas abiertas: límite de caracteres, anti-spam (rate limit y moderación básica: listas de palabras, longitud mínima).

Resultados y exportes

- Usuarios: gráficos/porcentajes agregados.
- Admin: tabla nominal (si no anónimo): usuario → opciones elegidas / texto.
- Exportes:
- `Datos/votaciones/{YYYY}/{MM}/{votacion_id}_{slug}/`
  - `resultados_agregados.json`
  - `resultados_agregados.csv`
  - `resultados_nominales.csv` (si procede)
  - `definicion_votacion.json` (metadatos, preguntas, opciones, reglas)

Criterios de aceptación

- Creación con validaciones, contadores, participación según reglas.
- Tres listas visibles correctas y actualizadas.
- Exportes generados en `Datos/`.

Fase 5 – Gestor de Usuarios (solo Admin)

Funcionalidad

- **Listado** con filtros (nombre, NIU, email, grupo, curso, rol).
- **Acciones**: añadir usuario, eliminar usuario, dar/quitar admin.
- **Solicitudes**:
- **Altas** (desde login) y **modificaciones de perfil**.
- Ver detalle completo y **Aceptar/Denegar** una a una. Se registra decisión, fecha, admin responsable y comentario opcional.

Importación por plantilla Excel

- Descarga de **plantilla oficial** desde el sistema: `Datos/usuarios/plantillas/plantilla_usuarios.xlsx` con columnas mínimas: **Nombre**, **NIU**, **Email universitario**. (Opcional: Grupo, Curso).
- Proceso de subida:
- Validar estructura y encabezados.
- Validar unicidad NIU/email y dominio permitido.
- Reporte de errores por fila.
- Insertar o actualizar registros según configuración (p.ej., “modo insertar únicamente” o “upsert”).
- Generar informe en `Datos/usuarios/importes/reporte_import_{timestamp}.xlsx`.

Eliminación de usuarios

- **Borrado lógico** (soft-delete) para preservar históricos de asistencia y votaciones.

Criterios de aceptación

- Alta/baja y rol admin funcionales.
- Flujo de solicitudes completo.
- Importación con reporte y validaciones.

Fase 6 – Ajustes

Secciones

1. **Login**:
2. Dominios permitidos (lista).

3. Credenciales SMTP / clave de aplicación para envío de emails.
4. Longitud y expiración del OTP, límites de reintento, ventana de reenvío.
5. **Información personal**:
6. Lista de campos activados.
7. Por cada campo: obligatorio / editable libre / requiere aprobación.
8. **Personalización**:
9. Logo del consejo/órgano (archivo almacenado en `Datos/ajustes/logo.*`).
10. Colores principales (primario/segundario/acento). Aplicados globalmente.
11. **Notificaciones**:
12. Emails de administradores para avisos.
13. Recordatorios opcionales (cierre de votaciones, resumen diario de actividad).
14. **General**:
15. Zona horaria (por defecto Europe/Madrid).
16. Políticas de retención y exportación automática.

Criterios de aceptación

- Cambios persistentes y aplicados en tiempo real o tras recarga controlada.

Fase 7 – Almacenamiento y exportaciones en `Datos/`

`Datos/` como almacenamiento primario

El sistema **no usa base de datos**; todo vive en `Datos/`. Los "exports" son simplemente vistas/derivados legibles por terceros.

Estructura propuesta

```
Datos/
  .locks/                    # ficheros de bloqueo
  ajustes/
    theming.json
    smtp.json
    dominios.json
    logo.png
  usuarios/
    index.json               # índices por email y NIU
    usuarios.jsonl           # alta/modificación/borrado lógico (append-
only)
    export_usuarios_{YYYYMMDD}.csv
    plantillas/plantilla_usuarios.xlsx
    importes/reporte_import_{timestamp}.xlsx
  solicitudes/
    solicitudes.jsonl        # {tipo: alta|modificacion, estado, payload}
  asistencia/
    {reunion_id}/
```

```
      reunion.json            # metadatos (titulo, inicio_utc,
cierre_utc,...)
      asistentes.jsonl        # entradas {user_id, hora_union_utc,
añadido_por}
      agregados.json          # cachés (n asistentes, duración)
      export_{fecha}.{csv|xlsx}
  votaciones/
    {votacion_id}/
      definicion_votacion.json
      preguntas.json          # array de preguntas/opciones
      votos.jsonl             # entradas {user_id, pregunta_id, opcion_id|
texto, emitido_en_utc}
      index_votos.json        # mapa user_id→[pregunta_id,...] para validar
límites
      resultados_agregados.json
      resultados_agregados.csv
      resultados_nominales.csv  # si no anónimo
  logs/
    audit_{YYYYMMDD}.jsonl
    email_{YYYYMMDD}.jsonl
  backups/
    backup_{YYYYMMDD}.zip
  _quarantine/
    ...
```

Reglas de escritura/lectura

- **Lectura**: vía capa repositorio que materializa vistas desde JSON/JSONL + índices locales.
- **Escritura**: append-only en JSONL con **lock**; actualizaciones se registran como eventos (`op: upsert|delete`). Vistas se recalculan al cargar o en tareas de compactación.
- **Exportes**: botones de "Exportar" generan CSV/XLSX desde las vistas y se guardan en la misma carpeta de la entidad.

Criterios de aceptación

- El sistema opera íntegramente con `Datos/` como almacenamiento.
- No hay corrupción en caídas (rename atómico + locks).
- Compacción y regeneración de índices funcionales.

Fase 8 – Seguridad, auditoría y cumplimiento

Auditoría

- Registrar eventos: login, OTP enviado/validado, altas/denegaciones, cambios de perfil, creación/edición/cierre de reuniones y votaciones, votos emitidos, importaciones, cambios de ajustes.
- Formato JSONL en `Datos/logs/audit_*.jsonl` + tabla `audit_log`.

Protección

- Rate-limit por IP/email en OTP, unión por código y emisión de votos.
- Hash de OTP (no guardar en claro). Hash de códigos de reunión si se requiere.
- Política CORS, CSP, cabeceras seguras.

RGPD

- Descarga de **datos personales** por usuario (perfil, asistencias, participaciones).

Backups

- Programar backup diario de DB + `Datos/ajustes/` y `Datos/logs/` → zip en `Datos/backups/`.

Fase 9 – Beta, QA, Accesibilidad y Rendimiento

Pruebas

- Unitarias (backend/frontend), integración (API), E2E (Cypress/Playwright).
- Casos críticos:
- OTP expirado/incorrecto/reintentos.
- Email fuera de dominio / no listado → solicitud de alta.
- Perfil incompleto → bloqueo navegación.
- Asistencia: unión con código inválido, doble unión, cierre con hora personalizada, exportes.
- Votaciones: límites por persona, edición deshabilitada, countdown < 1h, cierre automático, resultados y exportes.
- Importación Excel: encabezados erróneos, duplicados, dominio inválido.

Accesibilidad (WCAG 2.1 AA)

- Contraste de colores conforme.
- Navegación por teclado, labels y ARIA.
- Feedback claro de errores/estados.

Rendimiento

- Paginación y filtros en listas.
- Carga diferida de resultados pesados.
- Índices **locales** en disco (por email, NIU, estados) y caches en memoria del proceso.

Esquemas y almacenamiento (JSON sobre disco)

Validación con **Pydantic**. Todos los registros incluyen `id`, `created_at`, `updated_at` (ISO-8601 UTC) y `deleted_at` (nullable para borrado lógico).

Usuarios

- `usuarios.jsonl` eventos `{op, data}` donde `data` sigue:
- `id`, `email` (único), `nombre`, `niu` (único), `grupo_academico`, `curso`, `es_admin` (bool), `perfil_completo` (bool)
- `index.json`: `{ by_email: {email→user_id}, by_niu: {niu→user_id} }`

Allowed emails / Lista permitida

- En `ajustes/dominios.json` (dominios) y `usuarios/index.json` (emails permitidos subidos vía Gestor).

OTP Login

- `auth_otps.jsonl` con eventos: `{email, otp_hash, expira_en, intentos_restantes, enviado_a, ip}` (sin guardar OTP en claro). Purga por expiración.

Solicitudes

- `solicitudes.jsonl`: `{id, tipo: alta|modificacion, estado: pendiente|aceptada|denegada, payload, solicitante_email, user_id?, resuelto_por_admin_id?, comentario_admin}`

Asistencia

- Por reunión `{reunion_id}/reunion.json` y `asistentes.jsonl` con `{user_id, hora_union_utc, origen: usuario|admin}`.

Votaciones

- `{votacion_id}/definicion_votacion.json` (título, descripción, inicio/cierre, anonimato, edicion_permitida).
- `preguntas.json` con:
- `preguntas[]`: `{id, tipo: abierta|multiple, limite_opciones_por_persona, opciones[]?: [{id, texto, orden}]}`
- `votos.jsonl`:
- Para **múltiple**: entradas `{user_id, pregunta_id, opcion_id, emitido_en_utc}` (una por opción marcada).
- Para **abierta**: `{user_id, pregunta_id, texto_abierto, emitido_en_utc}`.
- **Validación** en la capa repositorio aplica límites por persona.

Auditoría

- `logs/audit_*.jsonl`: `{evento, actor_user_id?, detalles, creado_en}`

Endpoints / Casos de uso (resumen)

- **Auth**: `POST /auth/otp/request`, `POST /auth/otp/verify`
- **Perfil**: `GET/PUT /me`, `POST /me/cambios (solicitud)`
- **Asistencia**: `POST /reuniones`, `GET /reuniones?estado=…`, `POST /reuniones/{id}/unirse`, `PUT /reuniones/{id}`, `POST /reuniones/{id}/cerrar`, `GET /reuniones/{id}/export`
- **Votaciones**: `POST /votaciones`, `GET /votaciones?estado=…`, `GET /votaciones/{id}`, `POST /votaciones/{id}/votar`, `POST /votaciones/{id}/cerrar`, `GET /votaciones/{id}/resultados`, `GET /votaciones/{id}/export`
- **Usuarios**: `GET/POST /admin/usuarios`, `DELETE /admin/usuarios/{id}`, `POST /admin/usuarios/{id}/rol`
- **Solicitudes**: `GET /admin/solicitudes`, `POST /admin/solicitudes/{id}/resolver`
- **Ajustes**: `GET/PUT /admin/ajustes`

Capa de persistencia (detalle)

- Todos los endpoints llaman a **repositorios** que encapsulan E/S:
- `read_many()` → fusiona JSONL + índices.
- `append_event()` → abre lock, escribe evento, fsync, rename.
- `rebuild_index()` → tarea periódica o bajo demanda.
- **Transacciones lógicas**: para operaciones compuestas (p.ej., crear votación + preguntas) se escribe un **journal** temporal y se confirma al final (rename). Si falla, se revierte borrando el tmp.

UI / Wireframes (resumen de pantallas)

- **Login**: email + botón “Enviar código”. Si no listado → CTA “Notificar a administradores”. Form de solicitud.
- **Verificación OTP**: input de 6 dígitos con reenvío tras X segundos.
- **Panel principal**: tarjetas (Perfil, Votaciones, Asistencia, Ajustes/Usuarios si admin). Banner “Completa tu perfil”.
- **Información Personal**: formulario con estado de aprobación por campo. Historial de cambios.
- **Asistencia (Admin)**: tabs “Activos” y “Pasados”. Crear registro. Tabla con contadores y acciones.
- **Asistencia (Usuario)**: “Activos” (unirse por código) y “Pasados” (ver verde/rojo).
- **Votaciones**: tres acordeones/listas (no votadas, ya votadas, cerradas) con contadores y accesos.
- **Crear Votación**: fechas, título, descripción, constructor de preguntas (abierta/múltiple), opciones, límites.
- **Resultados**: vista agregada (gráficos) y nominal (tabla) según rol/anonimato.
- **Gestor de Usuarios**: tabla, filtros, acciones, pestaña “Solicitudes” con detalle y aprobar/denegar.
- **Ajustes**: secciones y validadores (test de email SMTP).

Casos borde y validaciones adicionales

- **OTP**: reenvío solo cada X s; invalidar códigos previos al generar uno nuevo.
- **Código de reunión**: único; evitar colisiones; opcionalmente hash en DB.
- **Doble asistencia**: impedir múltiples marcas del mismo usuario.

- **Cierre con hora personalizada**: debe ser > inicio y < ahora (cuando se usa en cierre).
- **Edición de voto**: si deshabilitada, cualquier POST subsecuente devuelve error claro.
- **Empates/empates múltiples**: mostrar en resultados sin romper layout.
- **Eliminación de usuario**: mantener su participación histórica (borrado lógico). Si votación anónima, no hay vínculo nominativo.
- **Importación Excel**: manejo de codificación, trimming de espacios, normalización de emails a minúsculas.
- **Dominios múltiples**: permitir lista; validación en servidor.
- **DST**: conversión correcta de horarios en UI (biblioteca de tz).
- **Bounced emails**: registrar en `emails_enviados` con estado y avisar a admin si supera umbral.

Notificaciones (opcionales, configurables)

- **A administradores**: nuevas solicitudes, importes con errores, fallos de envío de email, votación por cerrar (<24h).
- **A usuarios**: confirmación de solicitud enviada; recordatorio de votación (si activado).

Métricas y logs

- Métricas básicas: número de sesiones, OTP fallidos/exitosos, % perfiles completos, asistencia por reunión, participación por votación.
- Logs técnicos en `Datos/logs/` rotados por día.

Criterios de done (globales)

- Cumplidos todos los requisitos funcionales descritos.
- Seguridad básica aplicada y auditada.
- Exportes correctos en `Datos/` con estructura formal.
- Documentación de despliegue y `.env.example`.

Plan de despliegue

1. Entorno de **staging** con emails en modo sandbox.
2. Validación con un grupo piloto (admins + 5–10 usuarios).
3. Ajustes finos de dominios, plantilla Excel y roles.
4. Paso a **producción** y creación de primer backup completo en `Datos/backups/`.

Siguientes pasos inmediatos

1. Confirmar **stack** definitivo y proveedor de email.
2. Cerrar **lista de dominios** y formato exacto de NIU (longitud/regex).
3. Acordar **política de anonimato** para votaciones.
4. Crear **plantilla Excel** oficial.
5. Configurar **colores y logo** iniciales.

Si falta algún detalle que hayas mencionado en otra parte, aquí está integrado; si detectas nuevas necesidades, lo añadimos en este mismo documento y reordenamos el roadmap.