

¿Cómo es la organización de un equipo de desarrollo, mantenimiento y operaciones del proyecto votación?

*Utilizamos una metodología híbrida inspirada en entornos Ágiles y DevOps. Priorizamos la flexibilidad sin sprints rígidos ni reuniones diarias, apoyándonos en la infraestructura como código y comunicación asíncrona.*

*Utilizamos esta metodología debido a que nos permite ser muy flexibles y más rápidos en el desarrollo.*

¿Quiénes son los miembros del equipo y cuáles son sus roles?

- Jon Arriazu
- Diogo Da Cunha
- Cristian Meneses

*No hay roles definidos de forma “cerrada” si no que son todos perfiles multidisciplinares y se sigue un sistema de reparto de trabajo equilibrado.*

*Esto lo hacemos así para tener una distribución del trabajo equitativa y seguir una jerarquía horizontal en la que todos tenemos las mismas responsabilidades y la misma carga de trabajo.*

¿Cuáles son las herramientas de gestión del proyecto?

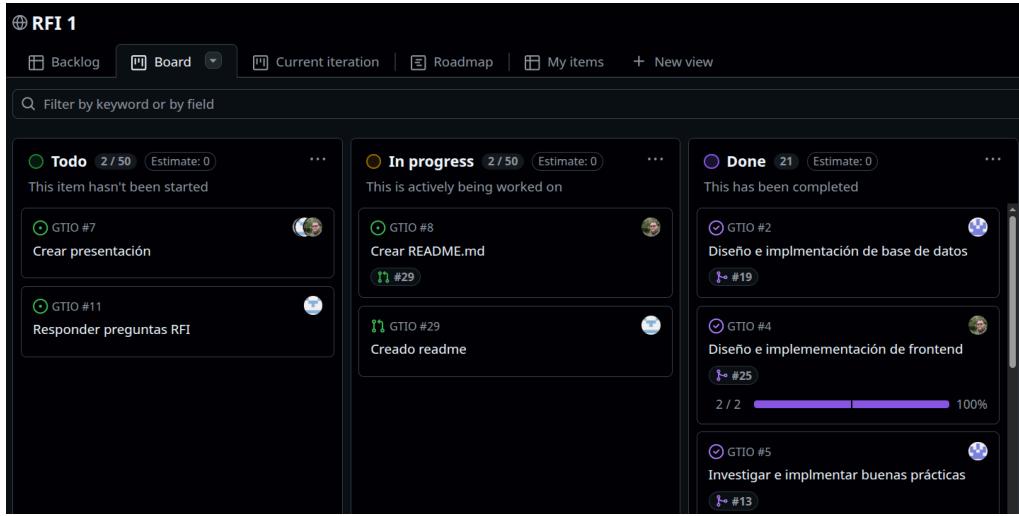
**GitHub Projects** es nuestro sistema de gestión principal, en el que llevamos un control de la planificación y evolución del proyecto.

Hemos elegido esta herramienta por su facilidad de uso y su integración con GitHub (nuestra herramienta de gestión de configuración).

Evidencias GitHub:

The screenshot shows a GitHub Projects board titled "RFI 1". The board has a "Backlog" filter selected. It displays two sections: "Todo" and "Done".

Section	Item	Assignee	Status	Linked pull requests
Todo	Crear presentación #7	DevcrewCM, Diogo-AA	Todo	
	Crear README.md #8	DevcrewCM	Todo	
	Responder preguntas RFI #11	Diogo-AA	Todo	
+ Add item				
Done	Diseño e implementación de base de datos #2	JonArriazu	Done	#19
	Diseño e implementación de frontend #4	DevcrewCM	Done	#25
	Investigar e implementar buenas prácticas #5	JonArriazu	Done	#13
	Implementar docker compose #6	Diogo-AA	Done	#28
	Crear ADR organización y metodología #9	DevcrewCM	Done	#18
	Crear ADR herramientas de gestión #10	JonArriazu	Done	#12



De forma auxiliar utilizamos **Discord** (Para reuniones y registro de notas de dichas reuniones) y **WhatsApp** (para comunicación directa y puntual).

Hemos elegido estas herramientas adicionales debido a su facilidad y comodidad de uso.

Evidencias Discord:

# notas-reunión

GET /usuarios/{id} (obtener info usuario)  
GET /usuarios/{id}/votos (obtener votos usuario)  
POST /votos (crear voto)  
GET /votacion (estado votación: candidatos, votos y gala)

Frontend  
App Web  
Totalmente libre, el que le toque que elija

Docker  
Dockerfiles (cada uno dockeriza lo suyo) + Docker Compose

Buenas prácticas  
Investigar e implementar buenas prácticas de desarrollo

Documentación  
Crear readme.md con explicación e instrucciones de levantar el sistema

23 de febrero de 2026

Diego ayer a las 22:36  
Notas reunión 23/02/2026

Cosas por hacer:

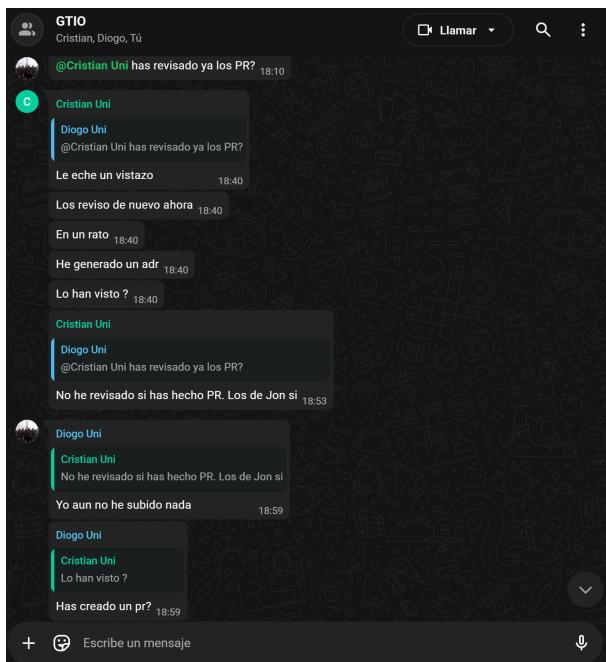
- Aceptar ADR organización y metodología (hecho ya); Cristian
- Terminar/arreglar frontend: Cristian + Diogo
- Docker Compose: Diogo
- ADR Backend: Diogo
- ADR Frontend (justificar decisión temporal); Jon

Entre los tres:

- Readme.md (empieza Jon)
- Responder preguntas RFI (empieza Jon)
- Presentación (empieza Jon)

(edited)

## Evidencias WhatsApp:



¿Cuáles son las herramientas de gestión de la configuración del proyecto?

Nuestra herramienta de gestión de la configuración y control de versiones es Github. En esta, tenemos centralizado el código fuente en un único repositorio público (Monorepo).

A screenshot of the GTIO GitHub repository page. The repository is public and has 11 branches and 0 tags. The main branch is 'main'. The repository was last updated 30 minutes ago by 'Diogo-AA' with 41 commits. The commit history includes: 'feat: creado proyecto base API' (last week), 'Merge pull request #27 from Diogo-AA/docs/issue-26-dis...' (47 minutes ago), 'build: unificación de docker compose de bd y api' (2 days ago), 'Frontend\_mejorado' (yesterday), 'build: creado docker compose unificado' (34 minutes ago), 'feat: creado proyecto base API' (last week), 'feat: creado proyecto base API' (last week), 'Initial commit' (3 weeks ago), 'Initial commit' (3 weeks ago), and 'build: creado docker compose unificado' (34 minutes ago). The repository has 0 stars, 0 forks, and 0 watching. It also lists 3 contributors: Diogo-AA, JonArriazu, and DevcrewCM. A language bar at the bottom shows the distribution of code: C# (47.7%), JavaScript (25.4%), CSS (14.9%), HTML (11.2%), and Dockerfile (0.8%).

*Utilizamos esta herramienta porque centraliza código y tareas y da trazabilidad completa (issues–commits–branches–PRs), es gratis y reduce herramientas externas facilitando la colaboración.*

¿Cómo se plantea la arquitectura del programa? Se requiere un acercamiento inicial a la arquitectura del sistema.

*La arquitectura del proyecto se ha estructurado en tres capas:*

- *Frontend* estático usando html, css y js.
- *API REST* con ASP.NET Core (.NET 10) y C#.
- *Base de Datos MySQL 8.4.*

*Hemos elegido esta arquitectura debido a su facilidad de uso y experiencia previa por parte de los integrantes del equipo.*

¿Cuáles son los servicios/microservicios de los que consiste el sistema de votación?

*Tenemos tres servicios desplegados en contenedores de docker: Frontend, Backend y Base de Datos.*

*Esta separación de responsabilidades permite que cada capa evolucione y escale de manera independiente.*

¿Cómo es la metodología y buenas prácticas utilizadas en el desarrollo de un proyecto votación?

*Metodología basada en GitHub: todo el trabajo se organiza en issues, rama por issue, commits claros, PR obligatorio a main, con mínimos de calidad (tests/linter) y código limpio y coherente (nomenclatura según lenguaje, comentarios útiles, modularidad). La IA se usa solo como apoyo (doc/borradores/consultas) pero siempre con revisión y responsabilidad humana.*

¿Cómo se despliega el sistema en un entorno de test local? Se requiere una pequeña guía de explicación de como levantar el programa.

*Para desplegar el proyecto seguir lo siguientes pasos:*

- *Situarse en el directorio raíz del proyecto (donde se encuentra el archivo docker-compose.yaml).*
- *Levantar toda la infraestructura ejecutando el siguiente comando:*
  - *docker compose up --build -d*
- *Acceder a la web a través de la ruta <http://localhost:5500>*

¿Cómo se prueba la funcionalidad del sistema en un entorno de test local?

*Para probar el frontend se debe acceder a la siguiente ruta: <http://localhost:5500>.*

*Para probar el backend de forma individual se puede enviar peticiones usando herramientas como cURL o Postman a la ruta <http://localhost:8080>. Para consultar la documentación de la API acceder a la ruta <http://localhost:8080/swagger>.*

¿Cuáles son los Dockerfiles utilizados para cada microservicio?

*Tenemos dos Dockerfiles:*

- *Frontend: usa una imagen de nginx y despliega el frontend.*
- *Backend: usa una imagen del sdk de .NET 10 para la compilación y otra imagen del runtime de .NET 10 para la ejecución.*

¿Cuál es la configuración del Docker Compose para levantar el sistema?

*El frontend se expone en el puerto 5500 y se añade una dependencia al contenedor backend.*

*El backend se expone en el puerto 8080, se definen algunas variables de entorno y se añade una dependencia al contenedor de base de datos.*

*La base de datos se expone en el puerto 3306, se asigna un .env con las credenciales de MySQL y se asigna un volumen de datos para persistir la información entre despliegues.*

¿Por qué es escalable y elástica la solución?

*La solución es escalable y elástica gracias a la implementación de una arquitectura basada en contenedores y elección de tecnologías altamente optimizadas para el rendimiento:*

- *Arquitectura de contenedores*
- *Arranque rápido para elasticidad inmediata*
- *Backend preparado para alta concurrencia*
- *Frontend estático usando html, css y js.*
- *Base de datos eficiente en concurrencia*