AnGoTest

Plataforma inteligente para crear, gestionar y realizar tests online.

Documentación técnica del proyecto
Versión 1.0 — Agosto de 2026

Autor: Jaime Terciado Lirola

“ La evaluación no debería ser un fin, sino un medio para aprender mejor."

Esta documentación recoge el análisis, diseño, implementación y funcionalidades de **AnGoTest**, una aplicación web que combina inteligencia artificial, corrección automática y análisis de rendimiento para transformar la forma en que educadores, estudiantes y profesionales evalúan y mejoran sus conocimientos.

Ver AnGoTest en acción
https://youtu.be/JOpi1MysM9g

Tabla de Contenidos

1. Definición del problema

Contexto

En el ámbito educativo, formativo y de selección de personal, la evaluación del conocimiento es una herramienta fundamental. Profesores, formadores, responsables de RR.HH. y estudiantes autodidactas necesitan medir el nivel de comprensión, identificar áreas de mejora y certificar el progreso. Sin embargo, los métodos tradicionales de evaluación presentan limitaciones significativas que obstaculizan la eficiencia, la personalización y la motivación de los participantes.

Carencias detectadas

1. **Procesos de corrección lentos y manuales**: La corrección de tests consume un tiempo valioso que podría emplearse en la enseñanza o en el análisis de resultados. Esto retrasa la retroalimentación y la identificación temprana de dificultades.
2. **Falta de inmediatez en los resultados**: Los alumnos y candidatos no reciben feedback al instante, lo que reduce la efectividad del aprendizaje y la capacidad de corregir errores sobre la marcha. La demora en conocer los aciertos y fallos desaprovecha el momento óptimo para el aprendizaje.
3. **Dificultad para crear tests personalizados**: Crear preguntas relevantes, ajustadas al nivel y al tema específico, es una tarea que requiere mucho tiempo y experiencia. Los educadores y formadores a menudo carecen de herramientas ágiles para generar pruebas adaptadas a sus necesidades concretas.
4. **Baja motivación y compromiso**: Los estudiantes y opositores suelen perder interés cuando no disponen de mecanismos que les permitan comparar su rendimiento con el de otros, ni de analíticas que visualicen su progreso. La ausencia de rankings y de seguimiento detallado reduce el estímulo para superarse.
5. **Falta de escalabilidad en procesos de selección**: En entornos empresariales, evaluar a múltiples candidatos con tests estandarizados y obtener resultados comparables de forma ágil es un desafío. La gestión manual de estos procesos es ineficiente y propensa a errores.

Impacto

Estas carencias generan consecuencias negativas:

- Pérdida de tiempo para educadores y formadores, que dedican horas a corregir en lugar de a enseñar.

- Retroalimentación ineficaz, que limita el aprendizaje significativo y la mejora continua.
- Desmotivación de los estudiantes, al no percibir avances claros ni sentirse parte de una comunidad competitiva.
- Desaprovechamiento de la tecnología, ya que no se utilizan herramientas de IA para agilizar la creación y corrección de pruebas.
- Ineficiencia en procesos de selección, con largos ciclos de evaluación y dificultad para comparar resultados objetivamente.

Solución propuesta: AnGoTest

AnGoTest nace para dar respuesta a estas necesidades mediante una plataforma integral de evaluación y aprendizaje. La aplicación permite:

- Crear tests de forma rápida y personalizada, tanto manualmente como mediante inteligencia artificial, con un sistema guiado para usuarios y con opción de instrucciones libres para administradores. Esto reduce drásticamente el tiempo de preparación de pruebas.
- Obtener resultados inmediatos, con corrección automática y visualización detallada de aciertos y errores, facilitando el aprendizaje a partir de los fallos y permitiendo a los educadores centrarse en la enseñanza.
- Ofrecer estadísticas y rankings que comparan el rendimiento con otros usuarios, fomentando la motivación y el espíritu de superación. Los alumnos y opositores pueden seguir su progreso y establecer metas.
- Proporcionar un panel de administración completo para gestionar usuarios, tests, resultados, cuotas e invitaciones, lo que permite a las empresas y centros educativos controlar todo el proceso de evaluación.
- Permitir la invitación a usuarios registrados o invitados, facilitando la evaluación externa sin necesidad de crear cuentas previas, ideal para procesos de selección.

Con AnGoTest, educadores, estudiantes y responsables de RR.HH. disponen de una herramienta ágil, inteligente y motivadora que transforma la evaluación en una experiencia eficiente, personalizada y enriquecedora. La plataforma incluye además guías de usuario y administrador integradas en la interfaz, que explican paso a paso todas las funcionalidades y reducen la curva de aprendizaje.

2. Reflexión: aportación y eficiencia

AnGoTest transforma la evaluación y el aprendizaje al automatizar tareas clave, simplificar la gestión y proporcionar información valiosa en tiempo real. A continuación, se detallan las principales mejoras y beneficios.

Automatizaciones y Simplificaciones Clave

1. **Creación de tests**
   - **Generación con IA**: Tanto usuarios como administradores pueden crear tests completos en segundos, eligiendo la temática, nivel, número de preguntas e idioma. Los administradores además pueden usar instrucciones libres para tests totalmente personalizados.
   - **Creación manual estructurada**: Formularios intuitivos con validación en tiempo real, listas de temas jerárquicos y gestión ágil de preguntas y respuestas.
   - **Importación JSON**: Permite crear tests a partir de estructuras predefinidas, ideal para migración o generación mediante chatbots de IA externos.
2. **Gestión de tests y contenidos**
   - **Filtros y ordenación avanzados**: En todas las listas (tests disponibles, en progreso, completados, administración), se pueden buscar y ordenar por múltiples criterios, reduciendo el tiempo de localización.
   - **Edición masiva**: Desde el panel de administración, se pueden eliminar tests, resultados o usuarios de forma masiva.
   - **Control de versiones**: Edición de tests existentes con actualización automática de preguntas y respuestas, además de la opción de guardar como copia para preservar el original y no afectar a los resultados.
3. **Corrección y feedback inmediato**
   - **Corrección automática**: Al finalizar un test, el sistema calcula puntuación, respuestas correctas e incorrectas y tiempo invertido. Además muestra la fecha de finalización y el número de intentos.
   - **Revisión de errores**: Modal con desglose de fallos, permitiendo al usuario aprender de sus errores.

4. **Seguimiento y análisis**
    - **Dashboard de usuario**: Estadísticas de rendimiento, progreso, mejoras entre intentos y rankings comparativos.
    - **Dashboard de administración**: KPIs globales, tests más/menos completados, usuarios más/menos activos, tasas de acierto, tiempos medios, etc.
    - **Exportación de datos**: Resultados exportables a CSV para análisis externos.
5. **Gestión de usuarios y accesos**
    - Registro e inicio de sesión seguro con JWT.
    - Roles diferenciados: Admin, user, guest con permisos específicos.
    - Invitaciones: Generación de enlaces únicos de un solo uso para usuarios registrados o invitados, con mensaje personalizado y expiración en 7 días.
6. **Control de cuotas y configuraciones**
    - Cuotas mensuales para generación de tests con IA, gestionables por administradores.
    - Configuraciones del sistema (SystemConfig) ajustables desde el panel de admin para personalizar comportamiento de la app (ej: días de expiración de un test).

Beneficios por Tipo de Usuario

- **Educadores y formadores**: Ahorro de tiempo en creación y corrección; personalización rápida; analítica educativa detallada; flexibilidad para usar IA o creación manual.
- **Estudiantes y autodidactas**: Feedback inmediato que acelera el aprendizaje; motivación aumentada por rankings y comparativas; seguimiento de progreso con métricas claras; repetición de tests efectiva para consolidar conocimientos.
- **Empresas y RR.HH.**: Procesos de selección ágiles y escalables; evaluación objetiva; invitación a candidatos externos sin registro previo; exportación de resultados.
- **Administradores**: Visión global del sistema; gestión centralizada de todos los recursos; mantenimiento ágil mediante SystemConfig; control de uso de IA a través de cuotas.

Resumen de Impacto

Área	Sin AnGoTest	Con AnGoTest
Creación de tests	Horas de trabajo manual	Minutos con IA o formularios avanzados
Corrección	Corrección manual y lenta	Automática e inmediata
Feedback	Días o semanas	Instantáneo
Seguimiento	Difícil y fragmentado	Centralizado con dashboards y rankings
Gestión de usuarios	Procesos manuales	Roles, invitaciones y control centralizado
Escalabilidad	Limitada por recursos	Alta, gracias a la automatización y la IA

AnGoTest no solo automatiza tareas tediosas, sino que proporciona una plataforma integral que mejora la eficiencia, la personalización y la motivación en cualquier entorno de evaluación y aprendizaje.

3. Listado de tecnologías utilizadas

3.1. Backend

Tecnología	Versión / Componente	Justificación
Django	5.0.3	Framework web robusto y seguro que proporciona ORM, sistema de autenticación, migraciones y administración integrada. Su estructura MVT facilita la organización del código en aplicaciones reutilizables.
Django REST Framework	3.14.0	Extensión para construir APIs RESTful de forma rápida, con serializadores, vistas basadas en clases, autenticación y permisos.
Simple JWT	djangorestframework-simplejwt 5.5.1	Implementación de tokens JWT para autenticación

		stateless, utilizada junto con cookies HttpOnly para mayor seguridad.
PostgreSQL	(driver psycopg2-binary)	Base de datos relacional potente y fiable, con soporte nativo para JSON, índices avanzados y alta concurrencia. Elegida por su madurez y rendimiento.
CORS Headers	django-cors-headers 4.3.1	Gestión de políticas CORS para permitir comunicaciones seguras entre el frontend (Angular) y el backend, con credenciales habilitadas.
Django Filter	django-filter 24.2	Proporciona filtros dinámicos y reutilizables para las vistas de listado, mejorando la experiencia de usuario en la administración.
DRF Spectacular	drf-spectacular 0.30.0	Generación de documentación OpenAPI (Swagger UI) a partir de los serializadores y vistas. Para ilustrar el uso, se han desarrollado serializers específicos y decoradores @extend_schema en la app accounts, que sirven como ejemplo de buenas prácticas para el resto de la aplicación.
Python dotenv	python-dotenv 1.0.1	Gestión de variables de entorno para separar configuración sensible (secretos, credenciales) del código fuente.
Redis / LocMemCache	Configuración de caché	Caché para almacenar jerarquías de temas y respuestas correctas, reduciendo el tiempo de respuesta en peticiones frecuentes. En desarrollo se

		usa LocMemCache; en producción se prevé Redis.
Groq API	SDK groq 0.4.2	Integración con el servicio de inteligencia artificial para la generación de tests. Se eligió por poseer una velocidad de inferencia altísima y un nivel gratuito muy generoso para modelos abiertos populares como Llama y Mistral .
Gunicorn	(servidor WSGI)	Servidor HTTP para entornos de producción, con soporte para múltiples workers y tiempo de espera configurable.

Librerías adicionales destacadas:

- PyJWT, django-stubs, typing_extensions para tipado y seguridad.
- django.contrib.* para administración, sesiones y mensajes.

Elección de APIView frente a ViewSet

En AnGoTest se ha optado por utilizar APIView y vistas genéricas (ListAPIView, RetrieveAPIView, etc.) en lugar de ViewSet. Esta decisión se fundamenta en:

- **Mayor control y flexibilidad**: La lógica de negocio incluye operaciones complejas (invitaciones, generación de tests con IA, gestión de cuotas, expiración automática) que no encajan en el patrón CRUD estándar de un ViewSet. APIView permite implementar cada endpoint con total libertad.
- **Claridad y mantenimiento**: Cada vista tiene una única responsabilidad y está anotada con decoradores @extend_schema para documentación OpenAPI, facilitando la lectura y el debugging.
- **Adaptabilidad**: El proyecto evoluciona de forma incremental; mantener vistas separadas reduce el acoplamiento y permite añadir o modificar funcionalidades sin afectar a otras partes.

Uso de plantillas para documentación estática

Aunque la interfaz principal de usuario está construida con Angular como SPA, las páginas de documentación (guías de usuario y administrador) se sirven directamente desde el backend utilizando el sistema de plantillas de Django. Estas guías están escritas en HTML con Tailwind CSS y se renderizan en el servidor, lo que permite actualizar la documentación de forma ágil, mantenerla sincronizada con el código y hacerla accesible desde rutas específicas (/docs/ y /docs/admin/) sin necesidad de recompilar el frontend. Esta decisión simplifica el mantenimiento de contenidos estáticos y evita que la documentación dependa del ciclo de build de Angular.

3.2. Frontend

Tecnología	Versión	Justificación
Angular	20.3.0	Framework completo para construir aplicaciones SPA con arquitectura basada en componentes, inyección de dependencias y enrutamiento. La versión 20 incluye mejoras en Signals, rendimiento y compatibilidad con SSR.
Angular CLI	21.0.2	Herramienta de línea de comandos para generar componentes, servicios y gestionar el build del proyecto.
TypeScript	5.9.2	Tipado estático que reduce errores y facilita el mantenimiento, especialmente en proyectos con muchos modelos y servicios.
RxJS	7.8.0	Programación reactiva para manejar peticiones HTTP, eventos y estados asíncronos de forma declarativa.
Angular Router	(incluido)	Gestión de rutas con guards (authGuard, adminGuard, roleGuard) para control de acceso según roles y autenticación.

Angular HTTP Interceptor	auth.interceptor.ts	Interceptor que añade credenciales (withCredentials) a todas las peticiones y maneja errores 401/403 redirigiendo a login o forbidden.
Servicios personalizados	AuthService, TopicsService, DashboardService, TestsManagementService, etc.	Centralización de la lógica de negocio y comunicación con la API, con uso de HttpClient y Signals para actualización reactiva de la UI.
Componentes reutilizables	ModalComponent, NavbarComponent, IdWithIconButtonComponent, ThemeToggleComponent	Promueven la consistencia visual y reducen la duplicación de código.
Tailwind CSS	4.1.17	Framework CSS utility-first que permite un diseño rápido y responsivo sin salir del HTML, con soporte para modo oscuro. Configurado mediante PostCSS.
PostCSS	8.5.6	Procesador CSS que permite utilizar Tailwind y otras características modernas.
Font Awesome	@fortawesome/angular-fontawesome 4.0.0, free-solid-svg-icons 7.1.0	Biblioteca de iconos vectoriales que mejora la interfaz de usuario con iconos escalables y personalizables.
Build y optimización	@angular/build 20.3.2	Sistema de construcción optimizado para producción, con minificación, tree-shaking y generación de bundles.

Vistas destacadas que obtienen datos reales del backend:

- **AdminDashboardComponent**: muestra KPIs, listas de tests más completados, usuarios activos, etc., consumiendo los endpoints /api/admin/dashboard/, /api/admin/dashboard/activity-summary/, etc.
- **AdminTestListComponent**: lista paginada y filtrada de tests con datos de TestListSerializer desde /api/test/admin/list/.

- **GenerateTestComponent**: genera tests mediante IA usando el endpoint /api/ai-requests/generate-ai-test/.
- **ResultsListComponent**, **UsersListComponent**, **QuotaManagementComponent**: obtienen datos de administración desde endpoints específicos.

3.3. DevOps y Despliegue

Tecnología / Herramienta	Componente	Justificación
Git	Control de versiones	Seguimiento de cambios, colaboración y ramificación de características.
GitHub	Repositorio remoto	Alojamiento del código, gestión de issues y (opcionalmente) integración continua.
Docker	Contenedores (backend, frontend, postgres)	Aislamiento y reproducibilidad del entorno de ejecución. Se definen tres servicios: PostgreSQL, backend (Python + Gunicorn) y frontend (Nginx sirviendo el build de Angular).
Docker Compose	Orquestación	Define y ejecuta los contenedores con redes y volúmenes compartidos. El archivo docker-compose.yml incluye: - **postgres**: imagen oficial de PostgreSQL 15 Alpine, con volumen persistente y healthcheck. - **backend**: construido con Dockerfile propio, inicia Gunicorn con 4 workers y ejecuta migraciones y recolección de estáticos. - **frontend**: construido con Dockerfile multietapa (build con Node.js y servido con Nginx), expone el puerto 80 internamente. - **Redes**: utiliza una red

		externa web_network para comunicación entre servicios. - **Volúmenes**: postgres_data, static_volume, media_volume para persistencia de datos.
Nginx	Servidor frontend	Sirve los archivos estáticos del build de Angular con configuración para SPA (fallback a index.html). También se utiliza como proxy inverso en producción (no mostrado en el compose).
Gunicorn	Servidor backend	Entorno WSGI de producción, configurado con 4 workers y timeout de 120 segundos.
Cron	Tarea programada	Ejecución diaria del comando expire_in_progress_results para marcar automáticamente como expirados los tests en progreso que superen el tiempo configurado. El cron se configura en el contenedor backend mediante el archivo docker/backend/cronjob.
Variables de entorno	Archivo .env	Configuración de secretos, credenciales de base de datos, API keys (Groq, JWT) y parámetros de entorno (DEBUG, ALLOWED_HOSTS, etc.).
VPS / Nube	DigitalOcean (u otro)	El proyecto está diseñado para ser desplegado en un VPS con Docker. La configuración de CORS y cookies seguras (secure) se adapta a producción. El archivo docker-compose.yml incluye un volumen para montar certificados SSL de Let's Encrypt (/etc/letsencrypt).

Justificación adicional:

- **Docker** simplifica el despliegue en cualquier entorno, garantiza la compatibilidad de versiones y facilita la escalabilidad horizontal.
- La combinación de **Gunicorn** y **Nginx** proporciona un rendimiento sólido y manejo de conexiones concurrentes.
- La tarea **cron** automatiza la limpieza de resultados expirados, liberando al administrador de esta tarea manual.
- La **red externa** permite que el frontend y backend se comuniquen internamente sin exponer puertos innecesarios, y el volumen de Let's Encrypt facilita la configuración de HTTPS.

3.4. Resumen de dependencias clave

Categoría	Herramientas
Backend	Django 5.0.3, DRF, JWT, PostgreSQL, Groq, Redis (opcional) + Django Templates para páginas de documentación
Frontend	Angular 20.3.0, TypeScript, RxJS, Tailwind CSS
DevOps	Docker, Docker Compose, Gunicorn, Nginx, Git, GitHub
Seguridad	JWT + cookies HttpOnly, CORS, hashing de contraseñas, variables de entorno, HTTPS con Let's Encrypt
Documentación	DRF Spectacular (OpenAPI)
Testing	(No se incluye en los extractos, pero se aplicará como mejora en el futuro)

Esta selección tecnológica asegura un desarrollo ágil, un mantenimiento sostenible y un despliegue fiable, cubriendo los requisitos funcionales y no funcionales del proyecto.

4. Definición de tipos de usuarios

La aplicación gestiona tres roles principales, definidos en el modelo User mediante el campo role con las siguientes opciones:

Rol	Identificador	Descripción	Acciones permitidas
Usuario	user	Usuario con cuenta creada y perfil verificado.	- Acceder a su dashboard personal con estadísticas y rankings. - Ver, iniciar y completar tests disponibles. - Gestionar su perfil (actualizar datos, email, contraseña). - Generar tests mediante IA (sujeto a cuota mensual). - Ver historial de tests en progreso y completados. - Invitar a otros usuarios a tests (creación de invitaciones). - Visualizar ranking global.
Invitado	guest	Usuario creado automáticamente al aceptar una invitación sin autenticación previa.	- Realizar tests a los que ha sido invitado. - Ver tests en progreso y completados. - Completar su perfil para convertirse en usuario registrado (actualizar datos, email, contraseña). - **Restricciones**: no puede generar tests con IA, no accede a rankings ni dashboard completo, no puede crear invitaciones.
Administrador	admin	Usuario con privilegios de gestión del sistema.	- Todas las acciones de usuario registrado. Panel de administración completo: - Gestión de usuarios (listar, ver perfil, eliminar). - Gestión de tests (crear, editar, eliminar, duplicar, importar JSON). - Gestión de resultados (listar,

			filtrar, eliminar, exportar CSV). - Gestión de cuotas de IA (crear, editar, eliminar, estadísticas). - Gestión de configuraciones del sistema (variables clave- valor). - Gestión de invitaciones (listar, filtrar, eliminar). - Dashboard global con métricas de usuarios, tests, actividad, rendimiento. - Exportar datos (CSV).
Eliminado	deleted	Usuario que ha solicitado la desactivación de su cuenta.	- No puede iniciar sesión. - Sus datos personales son anonimizados y sus tests y resultados transferidos a un usuario contenedor. - No aparece en listados activos.

Nota: Los roles se gestionan mediante el sistema de permisos de Django y el guard roleGuard en el frontend, que redirige a /forbidden si el usuario no tiene el rol requerido.

5. Casos de uso

Se describen casos de uso representativos para cada tipo de usuario, indicando flujo, resultado esperado y posibles errores.

5.1. Visitante (no autenticado)

Caso Uso 1: Registro de nuevo usuario

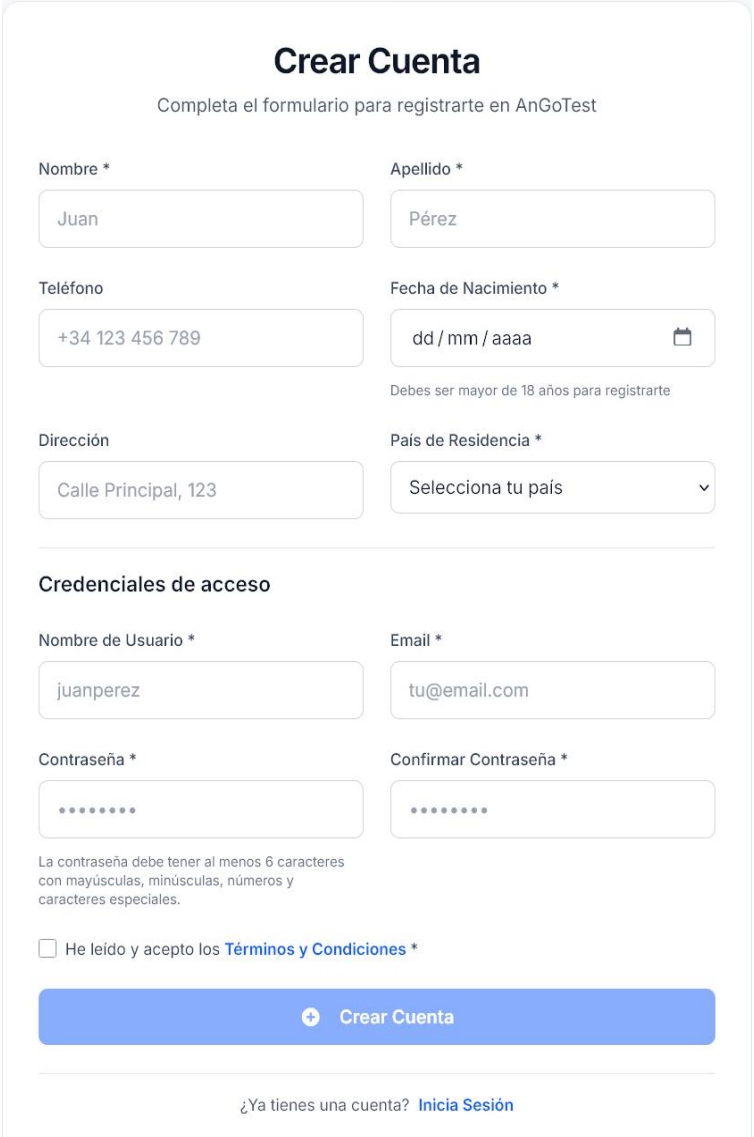
Flujo:

1. Accede a /register.
2. Completa el formulario (username, email, contraseña, país, fecha de nacimiento, etc.).
3. Envía la petición al endpoint POST /api/auth/register.
4. El backend valida que email y username no estén en uso y crea el usuario con rol user.

Resultado: Usuario registrado; se muestra mensaje de éxito y se redirige a login.

Posibles errores:

1. Email o username ya registrados (código 400).
2. Contraseña demasiado corta (validación en frontend y en serializer).
3. Fecha de nacimiento inválida.



Formulario de registro para usuarios

Caso Uso 2: Recuperación de contraseña

Flujo:

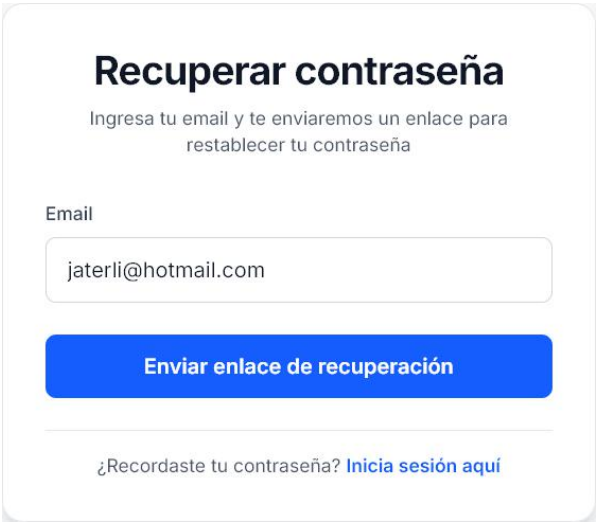
1. Solicita recuperación en /forgot-password con su email.
2. El backend genera un token, lo almacena en PasswordResetToken y envía un correo con el enlace de restablecimiento.

3. El usuario hace clic en el enlace y accede a la página de restablecimiento.
4. Introduce nueva contraseña y la confirma.
5. El endpoint POST /api/auth/reset-password-with-token valida el token y actualiza la contraseña.

Resultado: Contraseña actualizada; el token se marca como usado.

Posibles errores:

1. Token inválido o expirado (404/400).
2. Contraseñas no coinciden (400).
3. Email no registrado (el sistema responde con mensaje genérico por seguridad).



Formulario para la recuperación de la contraseña

Formulario para la restablecer de la contraseña

5.2. Usuario registrado

Caso Uso 3: Realizar un test

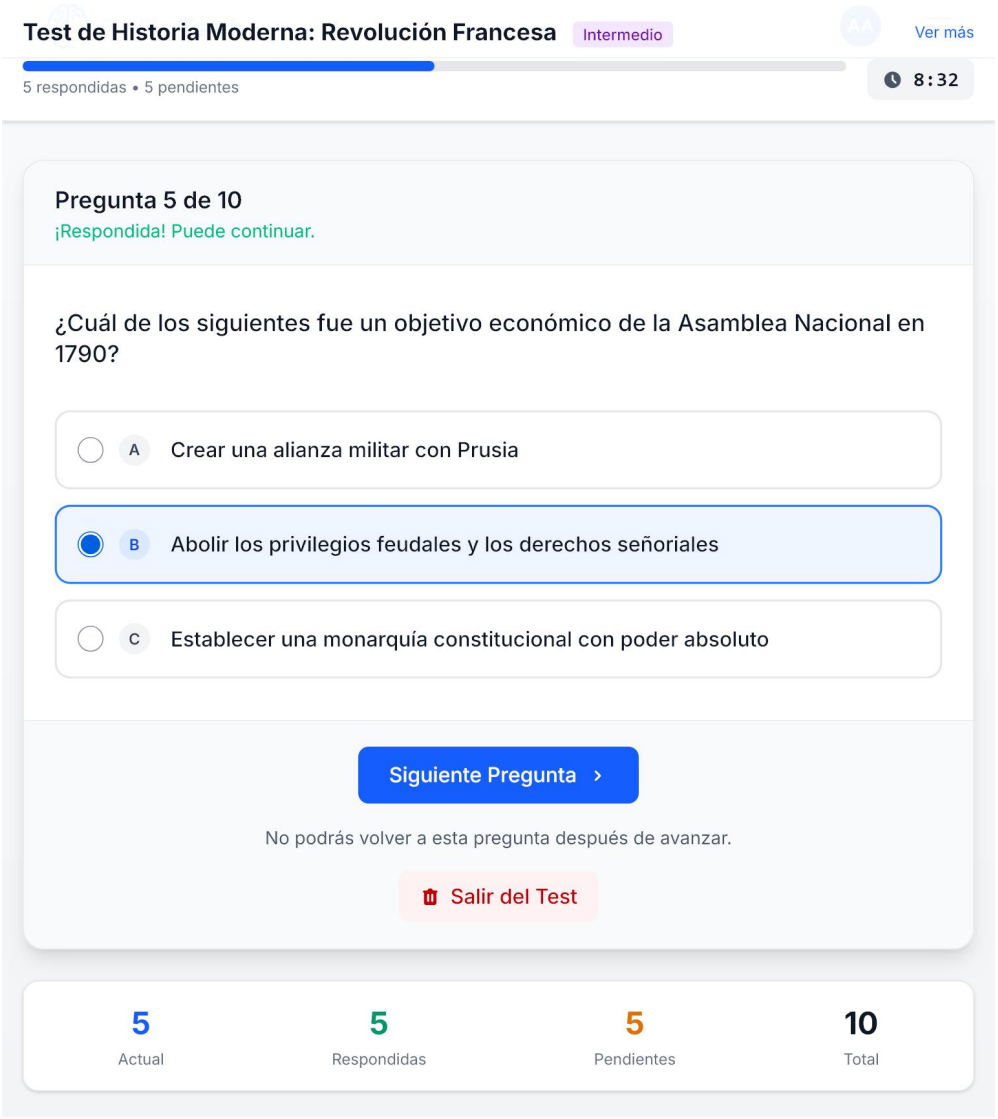
Flujo:

1. Desde la lista de tests disponibles (/tests/not-started) selecciona un test.
2. Se carga la primera pregunta (petición GET /api/test/{test_id}/next-question/).
3. Responde y guarda progreso mediante POST /api/test/{test_id}/save/ (estado in_progress).
4. Al finalizar todas las preguntas, envía el resultado con estado completed.
5. El backend calcula respuestas correctas/incorrectas y guarda el resultado.

Resultado: El test queda completado; se muestra puntuación y se guarda en el historial.

Posibles errores:

1. Test inactivo o no encontrado (404).
2. Intentar guardar respuestas sin autenticación (401).
3. Tiempo de sesión expirado (401).



Formulario de realización de test

Caso Uso 4: Generar un test con IA (modo guiado)

Flujo:

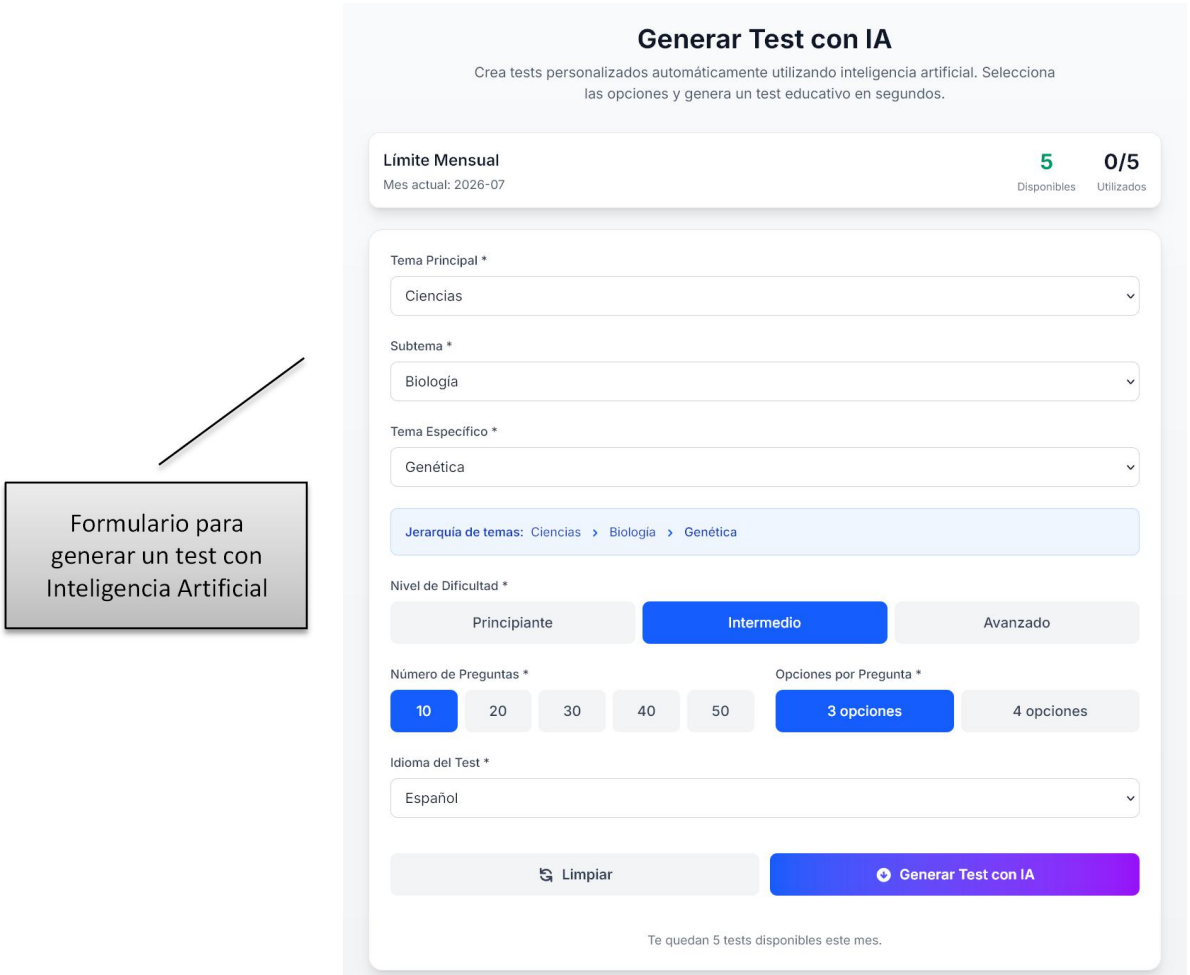
1. Accede a /generate-test (componente GenerateTestComponent).
2. Selecciona tema principal, subtema y tema específico (o modo libre de prompt si es admin).
3. Configura nivel, número de preguntas y respuestas.

4. Envía solicitud a POST /api/ai-requests/generate-ai-test/.
5. El backend verifica cuota disponible (UserQuota), consume una solicitud, llama a Groq API, parsea la respuesta y crea el test.

Resultado: Test creado y asociado al usuario; se redirige al test para realizarlo.

Posibles errores:

1. Cuota mensual agotada (403).
2. Error en la API de IA (500).
3. Validación de entrada fallida (400).



Formulario para generar un test con Inteligencia Artificial

5.3. Usuario invitado

Caso Uso 5: Aceptar invitación y completar test

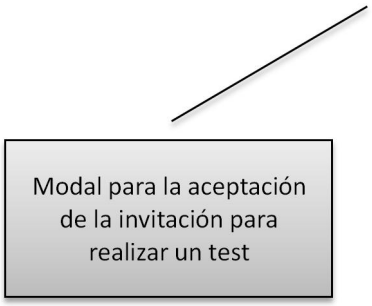
Flujo:

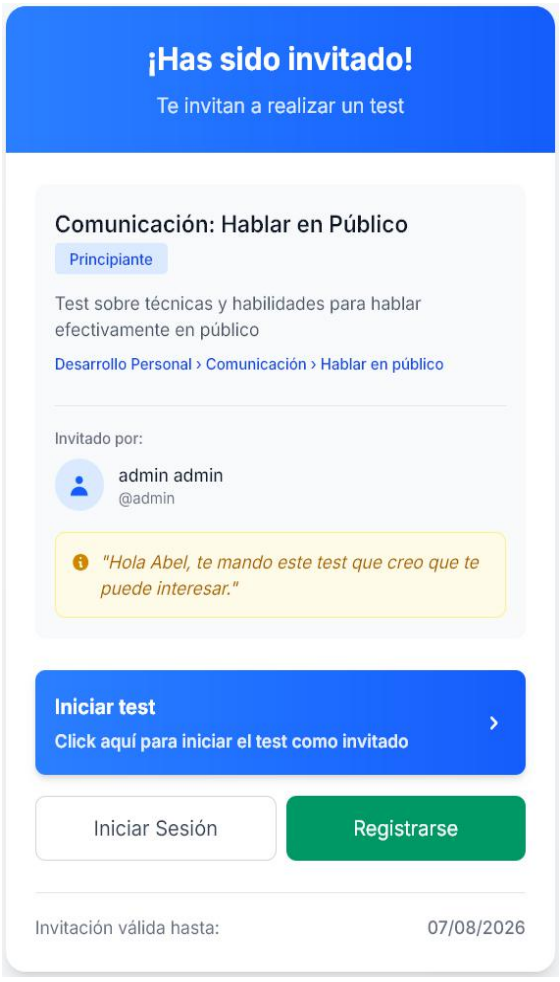
1. Recibe un enlace con token de invitación (ej. https://.../invitation/accept?token=...).
2. Accede al endpoint GET /api/invitations/check-invitation/ que valida el token.
3. Si el usuario no está autenticado, se le ofrece la opción de continuar como invitado (creación automática de un usuario con rol guest).
4. Se le asigna el test invitado y se redirige a la interfaz de realización del test.
5. Realiza el test; el progreso se guarda con su ID de usuario guest.

Resultado: Test completado, resultado asociado al guest.

Posibles errores:

1. Token expirado o ya usado (404/400).
2. Test eliminado o inactivo.
3. El usuario autenticado con rol user puede reclamar la invitación transfiriendo los resultados del guest a su cuenta (flujo adicional).





5.4. Administrador

Caso Uso 6: Gestión de usuarios – eliminar usuario

Flujo:

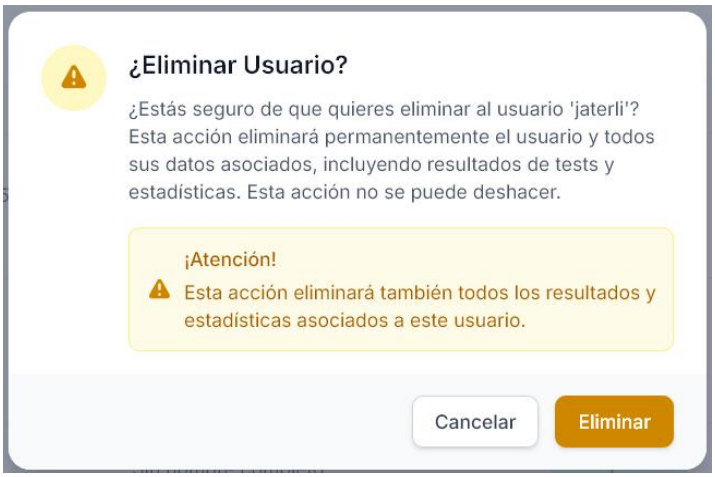
1. En el panel de administración (/admin/users), filtra y selecciona un usuario.
2. Pulsa “Eliminar”.
3. El frontend muestra un modal de confirmación.
4. Envía petición DELETE /api/user/admin/delete/{user_id}.
5. El backend valida que no sea el único admin ni el usuario contenedor, eliminará permanentemente el usuario y todos sus datos asociados, incluyendo resultados de tests y estadísticas.

Resultado: Usuario eliminado junto con sus datos personales y resultados.

Posibles errores:

6. Intentar eliminar al único administrador (400).
7. Usuario no encontrado (404).
8. Falta de permisos (403).

Modal para confirmar la eliminación de un usuario y sus datos asociados



Caso Uso 7: Exportar resultados a CSV

Flujo:

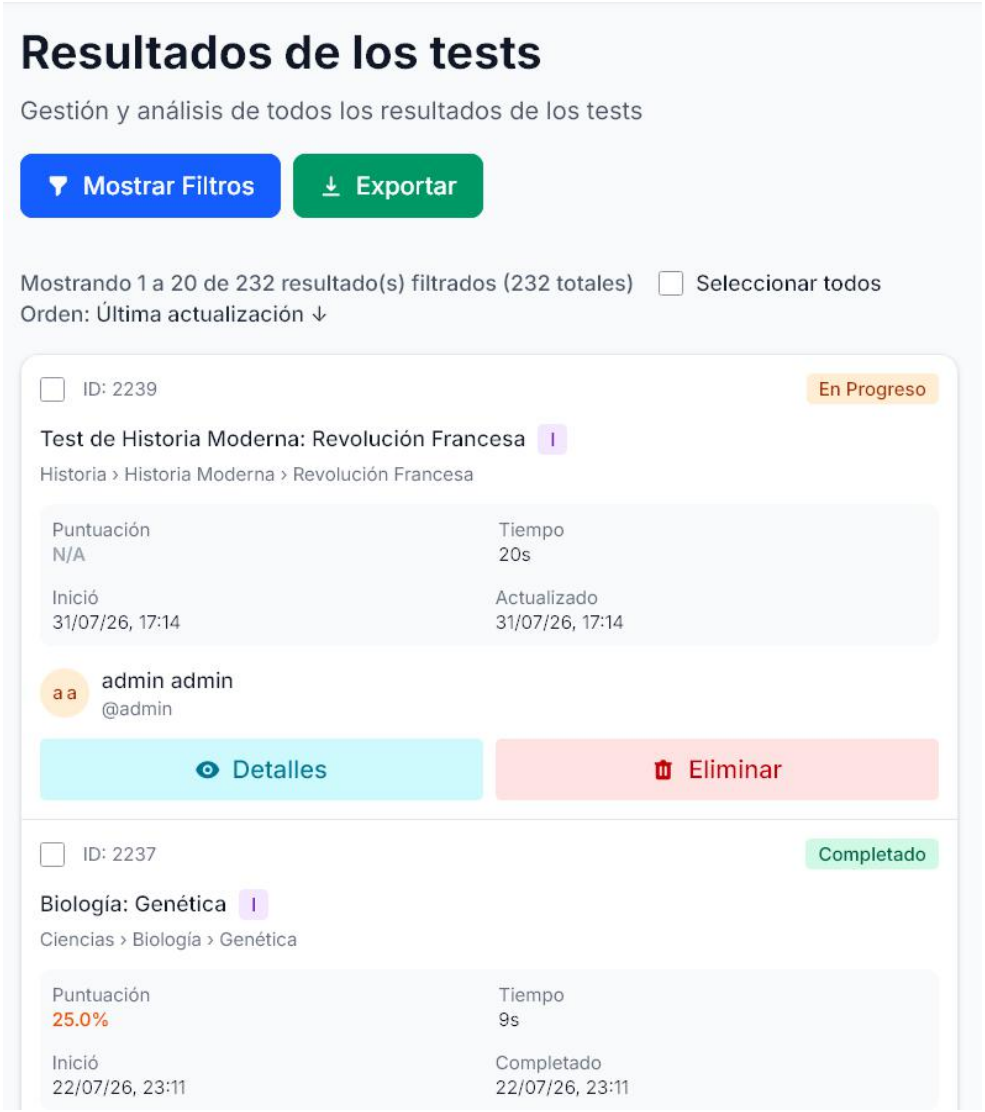
1. Desde la vista de resultados (/admin/results) aplica filtros (fechas, estado, tema, etc.).
2. Pulsa “Exportar”.

3. El frontend solicita GET /api/results/export-csv/ con los filtros.
4. El backend genera un archivo CSV con los datos de los resultados filtrados y lo envía como descarga.

Resultado: Archivo CSV descargado con las columnas relevantes (usuario, test, puntuación, tiempo, etc.).

Posibles errores:

1. Sin resultados para exportar (se podría devolver CSV vacío o error 404).
2. Error en la generación del archivo (500).



Vista parcial de resultados de los test ajustada a pantallas pequeñas. Desde esta vista podemos exportar los resultados filtrados a CSV haciendo clic en el botón Exportar

6. Seguridad y protección de datos

La aplicación implementa varias capas de seguridad, detalladas a continuación.

6.1. Autenticación y gestión de sesiones

JWT con cookies HttpOnly:

- El token de acceso se almacena en una cookie auth_token con la bandera HttpOnly (no accesible desde JavaScript) y Secure en producción.
- El interceptor HTTP del frontend envía las cookies con withCredentials: true.
- El backend utiliza JWTCookieAuthentication para validar el token en cada petición.

Renovación de sesión: El token tiene una duración de 24 horas (configurable en SIMPLE_JWT). No se implementa refresh token automático, pero se puede extender mediante nueva autenticación.

6.2. Validación y sanitización de datos

Backend:

- Los serializadores de Django REST Framework validan tipos, longitudes, unicidad y reglas de negocio (ej. num_questions entre 5 y 50).
- Se utiliza is_valid() y excepciones ValidationError para evitar datos maliciosos.
- Las consultas a la base de datos se realizan con el ORM, que escapa automáticamente las entradas (protección contra SQL Injection).

Frontend:

- Validación reactiva con Validators (required, minLength, etc.) antes de enviar peticiones.
- Se desinfectan las entradas mostradas en la UI (ej. mediante slice y truncate para evitar inyección de HTML en modales).
- **Ejemplo de escape en Angular**: Al usar {{ variable }} en las plantillas, Angular escapa automáticamente cualquier contenido HTML, convirtiendo caracteres como <, >, & en sus entidades HTML. Esto previene la ejecución de scripts maliciosos (XSS) incluso si un usuario introduce código en un campo de texto.

6.3. Protección JWT con cookies HttpOnly y XSS

JWT (JSON Web Tokens) con cookies HttpOnly:

- Para las APIs REST se usa JWTCookieAuthentication (clase personalizada en apps.accounts.authentication).
- La cookie auth_token se establece con:
  - httponly=True → no accesible desde JavaScript.
  - secure=True en producción.
  - samesite='Strict' o 'Lax' según entorno.
- Este enfoque mitiga los ataques CSRF porque:
  - El navegador no envía la cookie en solicitudes cross- site si SameSite es Strict o Lax.
  - El token JWT no se envía automáticamente en peticiones de terceros (no hay sesión basada en cookies de sesión tradicional).

Permisos:

- Por defecto, todas las vistas requieren IsAuthenticated (salvo excepciones públicas).

XSS:

- Los datos dinámicos mostrados en las plantillas de Angular están automáticamente escapados por el motor de templates (ejemplo anterior).
- En el backend, las respuestas JSON se serializan de forma segura.
- No se utiliza safe innecesariamente ni se inyecta HTML desde el servidor sin sanitizar.

6.4. Hashing de contraseñas

- Django utiliza PBKDF2 con sal por defecto (configurable en PASSWORD_HASHERS).
- Las contraseñas se almacenan mediante make_password() y se verifican con check_password().
- Se exige una longitud mínima de 6 caracteres en el registro y cambios de contraseña.

6.5. Gestión de secretos y variables de entorno

- Todos los secretos (SECRET_KEY, JWT_SECRET, credenciales de base de datos, API keys de Groq, credenciales de email) se cargan desde variables de entorno mediante python-dotenv.
- No se incluyen valores sensibles en el repositorio.
- En entornos de producción, se recomienda utilizar sistemas de gestión de secretos (Vault, Kubernetes secrets) o variables de entorno del sistema para mayor seguridad.

6.6. HTTPS

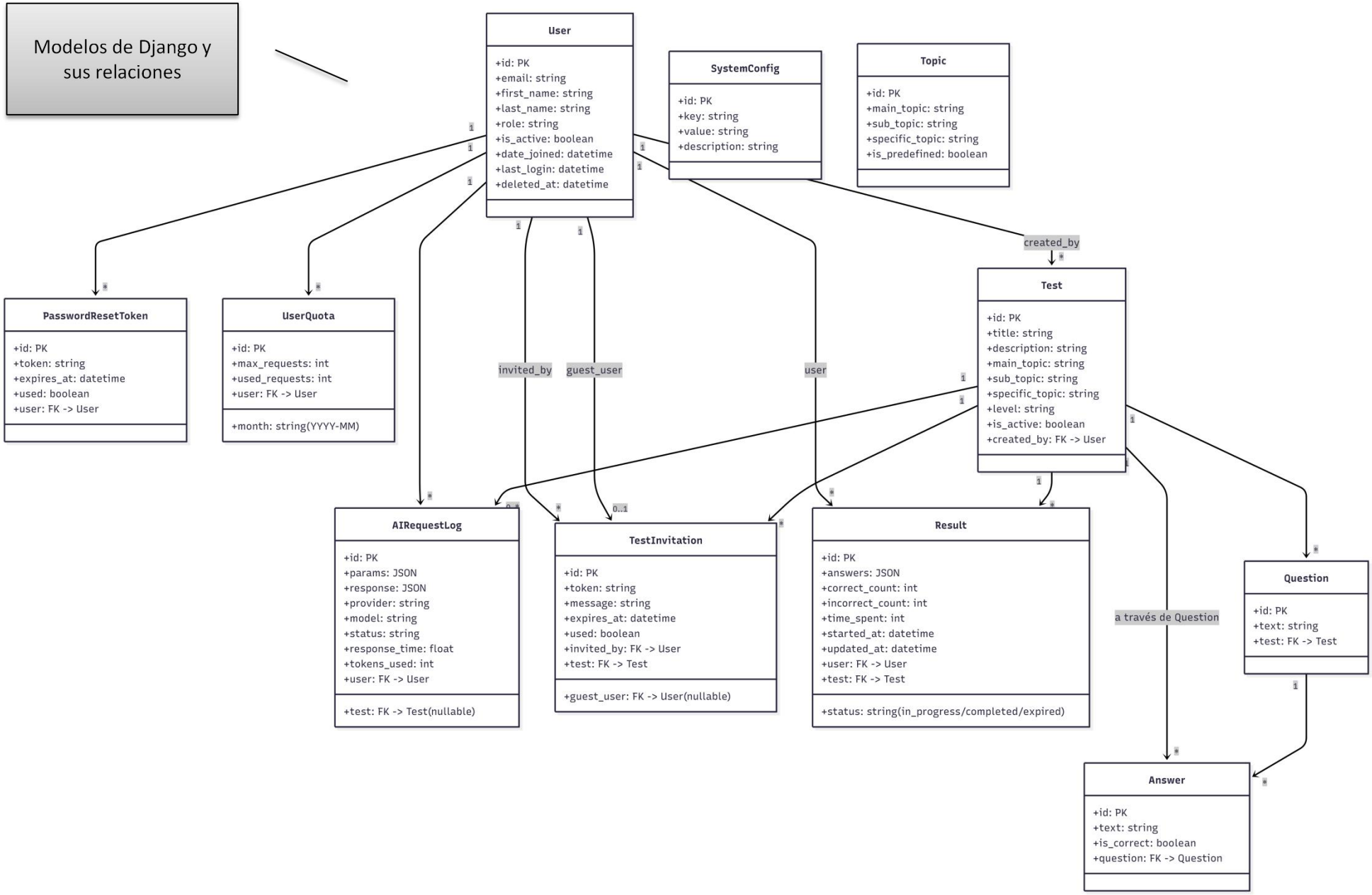
- En entornos de producción, la cookie auth_token se marca con Secure (solo se envía sobre HTTPS).
- La configuración de CORS permite solo orígenes seguros (https://).
- En el despliegue actual en DigitalOcean, se ha configurado un certificado SSL/TLS mediante Let's Encrypt, montando los certificados en el contenedor frontend (volumen /etc/letsencrypt), lo que garantiza que todas las comunicaciones estén cifradas.

6.7. Copias de seguridad

- Se ha implementado un sistema de backups nativo de PostgreSQL utilizando pg_dump programado con cron.
- La tarea se ejecuta diariamente (por ejemplo, a las 3:00 AM) y los archivos de backup se almacenan en un directorio persistente (volumen). Este cron se ha integrado en el mismo archivo docker/backend/cronjob que ya se usa para la expiración de resultados.

6.8. Otras medidas

- **CORS**: Solo se permiten orígenes definidos en CORS_ALLOWED_ORIGINS.
- **Registro de actividad: Las solicitudes de IA se registran en AIRequestLog para auditoría.**
- **Control de acceso**: Los endpoints administrativos están protegidos con IsAuthenticated y el permiso personalizado IsAdminUser.
- **Desactivación de cuentas**: Los datos se anonimizan y se transfieren a un usuario contenedor, en lugar de borrarse físicamente, permitiendo trazabilidad.


Modelos de Django y sus relaciones
User
+id: PK
+email: string
+first_name: string
+last_name: string
+role: string
+is_active: boolean
+date_joined: datetime
+last_login: datetime
+deleted_at: datetime
SystemConfig
+id: PK
+key: string
+value: string
+description: string
Topic
+id: PK
+main_topic: string
+sub_topic: string
+specific_topic: string
+is_predefined: boolean
PasswordResetToken
+id: PK
+token: string
+expires_at: datetime
+used: boolean
+user: FK -> User
UserQuota
+id: PK
+max_requests: int
+used_requests: int
+user: FK -> User
+month: string(YYYY-MM)
created_by
Test
+id: PK
+title: string
+description: string
+main_topic: string
+sub_topic: string
+specific_topic: string
+level: string
+is_active: boolean
+created_by: FK -> User
invited_by
guest_user
user
AIRequestLog
+id: PK
+params: JSON
+response: JSON
+provider: string
+model: string
+status: string
+response_time: float
+tokens_used: int
+user: FK -> User
+test: FK -> Test(nullable)
TestInvitation
+id: PK
+token: string
+message: string
+expires_at: datetime
+used: boolean
+invited_by: FK -> User
+test: FK -> Test
+guest_user: FK -> User(nullable)
Result
+id: PK
+answers: JSON
+correct_count: int
+incorrect_count: int
+time_spent: int
+started_at: datetime
+updated_at: datetime
+user: FK -> User
+test: FK -> Test
+status: string(in_progress/completed/expired)
a través de Question
Question
+id: PK
+text: string
+test: FK -> Test
Answer
+id: PK
+text: string
+is_correct: boolean
+question: FK -> Question

¿Qué muestra este diagrama?

1. Cada **recuadro** representa un modelo de Django con sus **campos más importantes** (clave primaria, atributos y claves foráneas).
2. **Las flechas** indican las relaciones `ForeignKey` con sus cardinalidades:
   `1` → `*` (uno a muchos)
   `1` → `0..1` (opcional, uno a cero o uno)
   `0..*` (cero a muchos, para el caso de `AIRequestLog` con `Test` nullable).
3. **Modelos sin relaciones** (`Topic` y `SystemConfig`) aparecen como entidades aisladas, tal como están en tu código.

Notas importantes

4. **Topic** no tiene FK con `Test` porque en tu diseño los temas se guardan como campos de texto en `Test`, no como claves foráneas. Por eso no hay línea de relación.
5. **TestInvitation** tiene **dos relaciones** con `User`: una para el invitador (`invited_by`) y otra para el invitado (`guest_user`, que puede ser nulo). El diagrama las muestra por separado.
6. La relación entre `Test` y `Answer` es indirecta a través de `Question`; la he representado con una flecha punteada para evitar duplicar, pero la cardinalidad real es `Test 1 -- * Answer` a través de `Question`.

Enlaces y contacto

Esta documentación forma parte del proyecto **AnGoTest**, una plataforma de evaluación y aprendizaje diseñada para facilitar la creación de tests, la corrección automática y el seguimiento del progreso.

Sitio web oficial: angotest.com

Repositorio: github.com/Jaterli/angotest

Blog del creador: jaterli.com — donde comparto mi portfolio y reflexiones sobre tecnología, educación y desarrollo de software.

Si tienes preguntas, sugerencias o deseas colaborar en el proyecto, no dudes en contactar a través del formulario de contacto en el sitio web o en el blog. También puedes abrir un issue en el repositorio público (si está disponible) para reportar errores o proponer mejoras.