Documento Técnico: Sistema de Gestión de Tickets con Django

Versión 1.0

**índice**

[1. Introducción 2](#_Toc188639716)

[2. Requisitos Funcionales 2](#_Toc188639717)

[2.2 Gestión de Tickets 2](#_Toc188639718)

[2.3 Notificaciones 2](#_Toc188639719)

[2.4 Historial de Cambios 3](#_Toc188639720)

[2.5 Comentarios y Colaboración 3](#_Toc188639721)

[2.6 Traslado y Reasignación de Tickets 3](#_Toc188639722)

[3. Requisitos No Funcionales 4](#_Toc188639723)

[3.1 Rendimiento 4](#_Toc188639724)

[3.2 Seguridad 4](#_Toc188639725)

[3.3 Usabilidad 4](#_Toc188639726)

[4. Consideraciones Técnicas Adicionales 5](#_Toc188639727)

[4.1 Integración con Django Admin 5](#_Toc188639728)

[4.2 Logs y Auditoría 5](#_Toc188639729)

[5. Diagrama de Flujo de Tickets 5](#_Toc188639730)

[6. Pruebas y Validación 5](#_Toc188639731)

[7. Entrega y Despliegue 5](#_Toc188639732)

[8. Esquema de Base de Datos 7](#_Toc188639733)

[8.1 Tabla: auth\_user (Modelo de Django estándar) 7](#_Toc188639734)

[8.2 Tabla: Usuario (Extensión personalizada) 7](#_Toc188639735)

[8.3 Tabla: Área 8](#_Toc188639736)

[8.4 Tabla: CategoríaUsuario 8](#_Toc188639737)

[8.5 Tabla: Ticket 8](#_Toc188639738)

[8.6 Tabla: HistorialTicket 9](#_Toc188639739)

[8.7 Tabla: Comentario 10](#_Toc188639740)

[8.8 Tabla: Notificación 10](#_Toc188639741)

[9. Flujo de Registro en Historial 10](#_Toc188639742)

[10. Consideraciones Clave 11](#_Toc188639743)

# 1. Introducción

Este documento describe los requisitos técnicos y características de un sistema de gestión de tickets desarrollado en Django. El sistema está diseñado para empresas que requieren gestionar solicitudes internas, asignar tareas y mantener un historial de cambios, priorizando la simplicidad, eficiencia y escalabilidad.

# 2. Requisitos Funcionales

**2.1 Gestión de Usuarios y Permisos**

* **Modelo de Usuario Extendido**:
  + Extender el modelo auth\_user de Django para incluir:
    - Relación con **Área** (clave foránea).
    - Relación con **Categoría de Usuario** (clave foránea).
    - Campo **supervisor** (autorrelación para jerarquías).
  + **Categorías de Usuario**:
    - Roles predefinidos: Empleado, Jefe de Área, Supervisor General.
    - Permisos basados en roles usando Django Groups y Permissions.

## 2.2 Gestión de Tickets

* **Estados de un Ticket**:
  + Abierto → En Progreso → Cerrado (con opción a reabrir).
  + Estados adicionales: Pendiente de Aprobación, Rechazado.
* **Prioridades**:
  + Baja, Media, Alta (configurable desde el admin de Django).
* **Campos Obligatorios**:
  + Título, Descripción (con soporte para texto enriquecido), Prioridad, Creado por.
* **Campos Opcionales**:
  + Fechas de asignación, inicio y finalización.

## 2.3 Notificaciones

* **Tipos de Notificaciones**:
  1. **Automáticas**:
     + **Asignación de Ticket**: Al asignar un ticket a un usuario.
     + **Traslado de Área**: Al mover un ticket a otra área.
  2. **Manuales**:
  3. **Avisos Generales**: Mensajes creados por jefes/supervisores para áreas o usuarios específicos.
* **Implementación**:
  1. Modelo Notificación con campos:
     + mensaje (texto), tipo (elección fija), fecha\_creación.
     + Relaciones opcionales: usuario\_destino (FK a User), área\_destino (FK a Área).
  2. **Sin seguimiento de "leído"**: Las notificaciones son mensajes informativos, no requieren confirmación.
  3. **Entrega**:
     + Mostrar en un panel de notificaciones en el dashboard del usuario.
     + Opcional: Integración con correo electrónico para notificaciones críticas (ej. prioridad alta)

## 2.4 Historial de Cambios

* **Registro Automático**:
  + Todas las acciones relevantes (creación, reasignación, cambio de estado, edición de campos clave).
  + Modelo HistorialTicket con:
    - ticket (FK), usuario (FK), acción (texto), detalles (para indicar cambios específicos en ticket), fecha.

## 2.5 Comentarios y Colaboración

* **Hilo de Comentarios**:
  + Cada comentario incluye: usuario, fecha, contenido (texto).
  + Soporte para adjuntar archivos (opcional, configuración por settings).

## 2.6 Traslado y Reasignación de Tickets

* **Flujo de Traslado**:
  1. Un empleado solicita traslado mediante un ticket (tipo "Solicitud de Traslado").
  2. El jefe de área aprueba/rechaza la solicitud.
  3. Si se aprueba, el ticket original se mueve a la nueva área y se notifica al solicitante.
* **Reasignación Directa**:
  1. Solo usuarios con permisos (jefes/supervisores) pueden reasignar tickets.

# 3. Requisitos No Funcionales

## 3.1 Rendimiento

* **Base de Datos**:
  + Índices en campos frecuentemente consultados (ej. estado, prioridad, fecha\_creación).
* **Optimización de Consultas**:
  + Usar select\_related y prefetch\_related en Django para reducir consultas en vistas de tickets.
* **Notificaciones Ligeras**:
  + No usar tareas en segundo plano (ej. Celery) para notificaciones. En su lugar, generar mensajes al guardar modelos.

## 3.2 Seguridad

* **Permisos**:
  + Usar el sistema de permisos de Django para restringir acciones (ej. solo jefes pueden reasignar tickets).
* **Validación de Datos**:
  + Asegurar que los usuarios solo puedan editar tickets de su área (salvo supervisores).
* **Protección CSRF y XSS**:
  + Habilitar middleware de seguridad de Django y sanitizar entradas de texto enriquecido.

## 3.3 Usabilidad

* **Dashboard Integrado**:
  + Vista resumen con:
    - Tickets asignados al usuario.
    - Notificaciones recientes.
    - Filtros por estado/prioridad.
* **Búsqueda y Filtros**:
  + Campos de búsqueda por título, descripción y usuario.
  + Filtros combinables (ej. prioridad=Alta + estado=Abierto).

# 4. Consideraciones Técnicas Adicionales

## 4.1 Integración con Django Admin

* Personalizar el panel de administración para:
  + Filtrar tickets por área, estado y prioridad.
  + Ver historial de cambios directamente desde la vista de un ticket.

## 4.2 Logs y Auditoría

* Registrar acciones críticas (ej. eliminación de tickets) en archivos de log.
* Usar el módulo logging de Django para seguimiento de errores.

# 5. Diagrama de Flujo de Tickets

1. Creación → Asignación → Trabajo → Cierre

2. Solicitud de Traslado → Aprobación → Reasignación de Área

# 6. Pruebas y Validación

* **Casos de Prueba Clave**:
  + Creación de tickets con diferentes prioridades.
  + Reasignación entre áreas con notificaciones automáticas.
  + Validación de permisos (ej. empleado no puede cerrar tickets de otra área).

# 7. Entrega y Despliegue

* **Entornos**:
  + Desarrollo (local), Staging (pruebas), Producción (servidor).
* **Requisitos Mínimos**:
  + Servidor con Python 3.8+, PostgreSQL/MySQL, y acceso a SMTP para notificaciones por correo (opcional).

# 8. Esquema de Base de Datos

## 8.1 Tabla: auth\_user (Modelo de Django estándar)

Extendido para integrar lógica empresarial.  
**Campos incorporados por defecto en Django:**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| username | Char(150) | Nombre de usuario único. |
| first\_name | Char(30) | Nombre del usuario. |
| last\_name | Char(150) | Apellido del usuario. |
| email | Char(254) | Correo electrónico. |
| is\_staff | Boolean | Indica si es parte del equipo. |
| is\_active | Boolean | Indica si la cuenta está activa. |

## 8.2 Tabla: Usuario (Extensión personalizada)

Relacionada con auth\_user mediante OneToOneField.

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Null/Blank | Descripción |
| user | OneToOne (User) | No | Relación al usuario de Django. |
| area | FK (Área) | Sí | Área asignada al usuario. |
| categoria | FK (CategoríaUsuario) | Sí | Rol del usuario. |
| supervisor | FK (Usuario) | Sí | Supervisor directo (autorrelación). |

**Lógica:**

* Hereda first\_name, last\_name, y email del modelo auth\_user.
* Los jefes de área deben tener is\_staff = True para acceder al admin de Django.

## 8.3 Tabla: Área

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Null/Blank | Descripción |
| id | AutoField | No | Clave primaria. |
| nombre | Char(100) | No | Nombre del área (único). |
| descripción | Text | Sí | Detalles adicionales. |

## 8.4 Tabla: CategoríaUsuario

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Null/Blank | Descripción |
| id | AutoField | No | Clave primaria. |
| nombre | Char(100) | No | Rol (ej. "Jefe de Área"). |
| descripción | Text | Sí | Funciones del rol. |

## 8.5 Tabla: Ticket

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Null/Blank | Descripción |
| id | AutoField | No | Clave primaria. |
| título | Char(200) | No | Título breve del ticket. |
| descripción | Text | No | Detalles del problema/solicitud. |
| estado | Char(50) | No | Valores: Abierto, En Progreso, Cerrado, Pendiente. |
| prioridad | Char(50) | No | Valores: Baja, Media, Alta. |
| fecha\_creación | DateTime | No | Auto-generado al crear. |
| fecha\_asignación | DateTime | Sí | Fecha de asignación a un usuario. |
| fecha\_inicio | DateTime | Sí | Fecha de inicio del trabajo. |
| fecha\_finalización | DateTime | Sí | Fecha de cierre. |
| creado\_por | FK (Usuario) | No | Usuario creador. |
| asignado\_a | FK (Usuario) | Sí | Usuario asignado. |
| área | FK (Área) | No | Área asociada al ticket. |

**Lógica:**

* Al crear un ticket, fecha\_creación se asigna automáticamente.
* Si asignado\_a se modifica, se actualiza fecha\_asignación.

## 8.6 Tabla: HistorialTicket

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Null/Blank | Descripción |
| id | AutoField | No | Clave primaria. |
| ticket | FK (Ticket) | No | Ticket asociado. |
| usuario | FK (Usuario) | Sí | Usuario que realizó la acción. |
| acción | Char(200) | No | Descripción de la acción (ver tabla de nomenclatura). |
| detalles | JSON | Sí | Cambios específicos (ej. {"prioridad": "Alta"}). |
| fecha | DateTime | No | Fecha de registro. |

**Nomenclatura de Acciones (Ejemplos):**

|  |  |
| --- | --- |
| Acción | Texto de acción |
| Creación de ticket | "Ticket creado por [usuario]" |
| Cambio de estado | "Estado actualizado a '[nuevo\_estado]'" |
| Reasignación | "Ticket reasignado a [usuario]" |
| Traslado de área | "Área cambiada a [nombre\_área]" |
| Edición de prioridad | "Prioridad actualizada a '[nueva\_prioridad]'" |
| Comentario añadido | "Comentario agregado por [usuario]" |

De ser necesario a la nomenclatura se le puedo añadir el estado actual y el nuevo, por ejemplo "De [nombre\_área] cambiada a [nombre\_área\_nueva]"

## 8.7 Tabla: Comentario

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Null/Blank | Descripción |
| id | AutoField | No | Clave primaria. |
| ticket | FK (Ticket) | No | Ticket asociado. |
| usuario | FK (Usuario) | No | Autor del comentario. |
| contenido | Text | No | Texto del comentario. |
| fecha\_creación | DateTime | No | Fecha de creación automática. |

## 8.8 Tabla: Notificación

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo | Null/Blank | Descripción |
| id | AutoField | No | Clave primaria. |
| mensaje | Char(500) | No | Contenido de la notificación. |
| tipo | Char(50) | No | Valores: Asignación, Traslado, General. |
| usuario\_destino | FK (Usuario) | Sí | Destinatario individual. |
| área\_destino | FK (Área) | Sí | Destinatarios de un área completa. |
| fecha\_creación | DateTime | No | Fecha de creación automática. |

**Lógica:**

* Si usuario\_destino y área\_destino son nulos, la notificación es global.
* Ejemplo de mensaje automático:
  + Tipo **Asignación**: "Se te ha asignado el ticket #ID: [título]".
  + Tipo **Traslado**: "El ticket #ID ha sido movido al área [nombre\_área]".

# 9. Flujo de Registro en Historial

**Creación de Ticket**:

HistorialTicket.objects.create(

ticket=ticket,

usuario=creador,

acción=f"Ticket creado por {creador.user.get\_full\_name()}",

)

**Cambio de Estado**:

HistorialTicket.objects.create(

ticket=ticket,

usuario=usuario,

acción=f"Estado actualizado a '{nuevo\_estado}'",

detalles={"estado\_anterior": estado\_anterior, "estado\_nuevo": nuevo\_estado}

)

**Reasignación**:

HistorialTicket.objects.create(

ticket=ticket,

usuario=usuario,

acción=f"Reasignado a {nuevo\_asignado.user.get\_full\_name()}",

detalles={"asignado\_anterior": anterior\_asignado.id, "asignado\_nuevo": nuevo\_asignado.id}

)

# 10. Consideraciones Clave

1. **Herencia de Usuario**:
   * Los nombres (first\_name, last\_name) se gestionan desde el modelo estándar de Django.
   * Para obtener el nombre completo: usuario.user.get\_full\_name().
2. **Rendimiento**:
   * Usar indexes en campos como estado, prioridad en la tabla Ticket.
3. **Seguridad**:
   * Restringir ediciones de área y asignado\_a solo a usuarios con permisos de jefatura.
4. **Notificaciones Livianas**:
   * No almacenar historial de notificaciones antiguas: Implementar limpieza periódica (ej. eliminar notificaciones de más de 30 días).