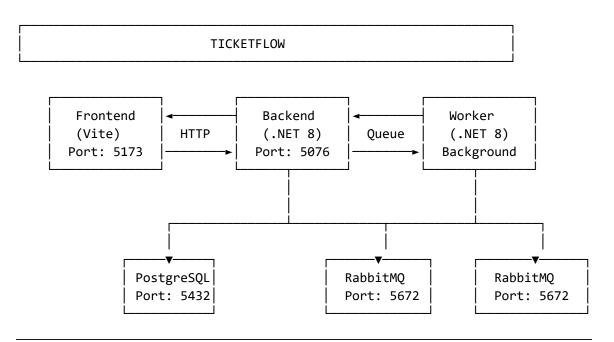
T Arquitectura y Flujo de Colas -**TicketFlow**

Diagrama simple de la arquitectura del sistema y el flujo de mensajería.

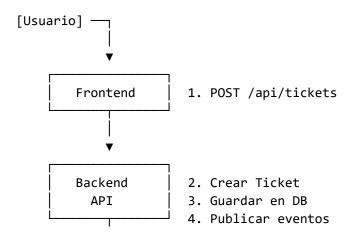


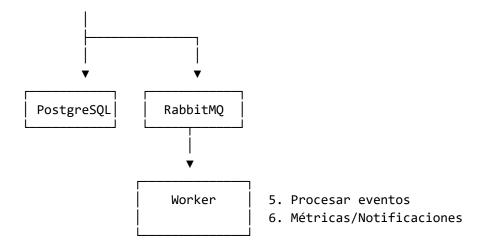
Arquitectura General



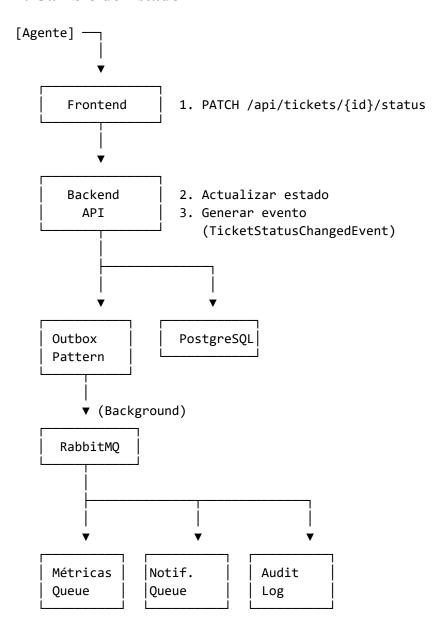
Flujo de Datos

1. Creación de Ticket





2. Cambio de Estado



© Componentes Principales

Frontend (Vite + TypeScript)

Frontend Components

- Tablero Kanban
- Gestión de Usuarios
- Detalles de Ticket
- Login/Auth

HTTP/REST

Backend API (.NET 8)

REST API Endpoints

- /api/tickets
- /api/users
- /api/auth

| ▼

Application Layer

- Commands (CQRS)
- Queries
- Handlers

| ▼

Domain Layer

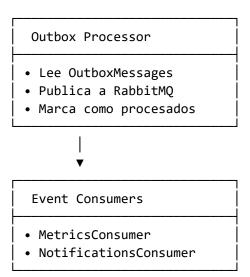
- Entities (Ticket, User)
- Value Objects
- Domain Events

| ▼

Infrastructure

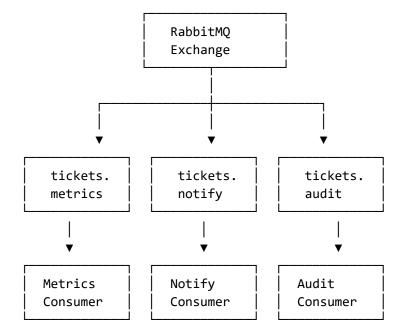
- EF Core (PostgreSQL)
- RabbitMQ Publisher
- Outbox Pattern

Worker (Background Processor)



Colas de RabbitMQ

Topología de Colas



Eventos Publicados

Evento	Cola	Propósito
TicketCreatedEvent	tickets.metrics	Contadores, estadísticas
TicketStatusChangedEvent	tickets.metrics	Métricas de cambios
TicketAssignedEvent	tickets.notifications	Notificar a agente
TicketCreatedEvent	tickets.notifications	Notificar creación

Outbox Pattern

Flujo del Outbox

Por qué Outbox?

- **✓ Garantiza entrega** No se pierden eventos
- **Transaccional** Todo o nada
- Resiliente Reintenta en caso de falla
- **V** Trazabilidad Auditoría completa

Base de Datos

Tablas Principales

Tickets
Id (PK) Title Description Status Priority AssignedTo (FK) CreatedBy (FK) CreatedAt
UpdatedAt

TicketActivities

Id (PK) TicketId (FK) UserId (FK) Action OldValue NewValue Timestamp

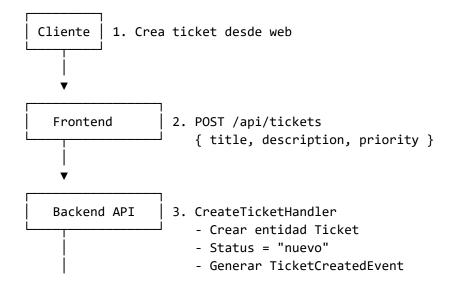
Users

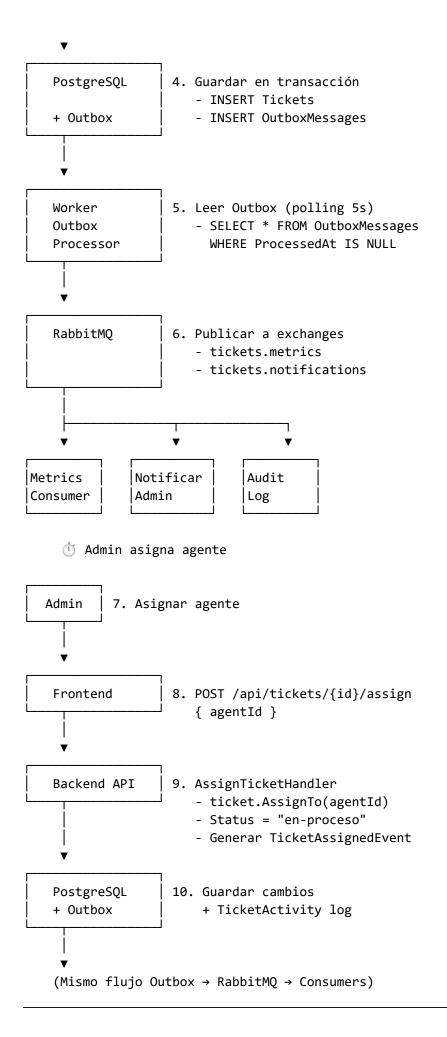
Id (PK) Name Email (Unique) Role CreatedAt

${\tt Outbox Messages}$

Id (PK) EventType Payload (JSON) CreatedAt ProcessedAt

Flujo Completo: Crear y Asignar Ticket





Monitoreo

URLs de Monitoreo

Servicio	URL	Información
RabbitMQ Management	http://localhost:15672	Colas, mensajes, consumidores
API Health	http://localhost:5076/health	Estado de la API
Swagger	http://localhost:5076/swagger	Documentación API

Notas Técnicas

• Outbox Polling: Cada 5 segundos

• Message TTL: Sin límite (mensajes persistentes) • Retry Policy: 3 reintentos con backoff exponencial • Transacciones: PostgreSQL nivel Read Committed

• Serialización: JSON (System.Text.Json)

Ver también:

- docs/rabbitmq-topology.md Detalles de RabbitMQ
- docs/OUTBOX_WORKER.md Implementación del Outbox Pattern
- EXECUTION.md Cómo ejecutar el sistema