# API Performance Dashboard Implementation

## Dashboard Structure and Rationale

This dashboard is designed to rapidly identify and diagnose API performance issues by organizing information in a problem-oriented rather than tool-oriented way.

## Key Dashboard Features

### 1. Service Health Overview

- **RED Metrics Summary**: Single-panel overview showing Request Rate, Error Rate, and Duration (p95) with color-coded health indicators
- **SLO Compliance**: Percentage of requests meeting latency and availability targets
- **Time Comparison**: Current metrics vs. same time yesterday/last week to identify anomalies

### 2. Request Performance Breakdown

- **Latency Heatmap**: Shows distribution of response times across endpoints and time
- **Endpoint Table**: Sortable table of endpoints with key metrics (volume, errors, latency)
- **Resource Correlation**: CPU/Memory/DB Connection usage overlaid with latency spikes
- **User Impact**: Affected users/tenants during performance degradation periods

### 3. Error Analysis

- **Error Timeline**: Error count by type and endpoint over time
- **Status Code Distribution**: Breakdown of HTTP status codes
- **Log Context**: Direct links to relevant error logs with context
- **Exception Patterns**: Common exception types and frequency

### 4. API Component Analysis

- **Latency Breakdown**: Visualization of where time is spent in the request lifecycle:
  - Database query time
  - External service call time
  - Application processing time
  - Network/serialization time
- **Bottleneck Identification**: Highlights the component contributing most to latency

### 5. Dependency Health

- **Database Performance**: Query execution time, connection usage, query volume
- **External Services**: Call volume, latency, and error rates for dependencies
- **Infrastructure**: Node-level metrics for relevant infrastructure components

### 6. Trace Explorer

- **Slowest Traces**: Automatically surfaces the slowest recent request traces
- **Error Traces**: Links to traces for recent error conditions
- **Trace Comparison**: Compare normal vs. slow request traces side-by-side

### 7. Debugging Accelerators

- **Common Issues Panel**: One-click filters for typical problems (DB slowness, auth issues)
- **Runbook Links**: Direct access to relevant troubleshooting guides
- **Historical Context**: Quick access to similar past incidents

## Implementation Details

### Metrics Configuration in Prometheus

- Custom API middleware to capture detailed timing at key request processing stages
- Histogram metrics for latency with appropriate bucket sizes
- Counter metrics for errors with detailed labels
- Gauge metrics for connection pools and resource utilization

### Logs Configuration in Loki

- Structured logging format with consistent fields
- Log level filters to focus on actionable information
- Log queries pre-configured for common error patterns

### Traces Configuration in Tempo

- Sampling strategy that ensures problematic requests are always traced
- Span naming conventions for consistent visualization
- Automatic instrumentation of database and HTTP clients

### Alert Rules

- Multi-level alerting based on severity
- Rate of change alerts for early detection
- Compound alerts that combine multiple indicators

### Expandability

- Template variables for environment, service, and endpoint selection
- Consistent label structure across all metrics
- Dashboard JSON can be parametrized and version-controlled
- Automated dashboard provisioning via Grafana API