可观测性监控平台 - 项目需求文档

Project Overview

为 SRE 工程师和测试工程师开发一个现代化的综合监控平台前端演示系统。该平台整合指标监控、链路追踪、日志分析等多维度可观测性数据,提供直观的可视化界面和高效的故障排查工作流。本项目重点打造精美的前端界面和流畅的交互体验,使用 Mock 数据模拟真实监控场景,无需实现真实后端服务。

核心价值: 快速故障定位 → 详细信息查看 → 根因分析的完整闭环

Functional Requirements

1. 核心功能模块

1.1 综合仪表盘 (Overview Dashboard)

- **服务健康状态看板**: 展示所有监控服务的整体健康度(健康/警告/故障)
- **关键指标概览**: 错误率、平均响应时间、QPS、资源使用率等核心指标的实时展示
- **告警面板**: 当前活跃告警列表,按严重程度分级(Critical/Warning/Info)
- **趋势图表**: 关键指标的时序趋势图(最近1h/6h/24h/7d)

1.2 指标监控 (Metrics Monitoring)

- **服务列表**: 展示所有被监控的服务/应用
- **多维度指标**:
  - 业务指标: 错误率、成功率、请求量、响应时间分位数(P50/P90/P99)
  - 系统指标: CPU使用率、内存使用率、磁盘I/O、网络带宽
- **多图表类型**: 折线图、柱状图、饼图、热力图
- **对比视图**: 支持多个服务或时间段的指标对比

1.3 链路追踪 (Distributed Tracing)

- **Trace列表**: 展示请求链路列表,包含TraceID、服务名、耗时、状态
- **链路拓扑图**: 可视化展示服务调用关系和依赖
- **Span详情**:
  - 火焰图/甘特图展示请求各阶段耗时
  - 每个Span的详细信息(服务名、操作、时长、标签、日志)
- **慢查询分析**: 识别并高亮显示性能瓶颈节点

1.4 日志分析 (Log Analytics)

- **日志流**: 实时日志流展示(支持虚拟滚动)
- **高级搜索**:
  - 关键词搜索(支持正则表达式)
  - 字段筛选(服务名、日志级别、TraceID等)
  - 日志级别过滤(DEBUG/INFO/WARN/ERROR/FATAL)

- **日志详情**: 展开查看完整日志内容、上下文日志
- **日志统计**: 日志数量趋势、级别分布饼图

1.5 自定义Dashboard

- **拖拽式布局**: 支持图表的拖拽、调整大小
- **图表配置**:
  - 选择数据源和指标
  - 自定义图表类型
  - 设置阈值告警线
- **Dashboard模板**: 预设模板(应用监控、基础设施监控、业务监控等)
- **保存与分享**: 保存个人Dashboard配置(使用LocalStorage)

2. 交互功能

2.1 时间控制

- **快捷时间选择**: 最近5分钟、15分钟、1小时、6小时、24小时、7天
- **自定义时间范围**: 日期时间选择器
- **实时模式**: 自动刷新数据(可设置刷新间隔: 5s/10s/30s/1min)
- **时间对比**: 与上一时段对比功能

2.2 数据筛选

- **多维度筛选器**:
  - 服务/应用名称
  - 环境(生产/预发布/测试)
  - 区域/可用区
  - 实例ID
  - 标签/标记
- **筛选条件保存**: 保存常用筛选组合

2.3 告警管理

- **告警规则展示**: 显示已配置的告警规则(Mock数据)
- **告警历史**: 历史告警记录,支持按时间、严重程度筛选
- **告警详情**: 触发条件、影响范围、恢复时间

2.4 联动与钻取

- **跨模块联动**:
  - 从指标异常跳转到对应的Trace
  - 从Trace跳转到相关日志(通过TraceID)
- **下钻分析**: 从概览→服务详情→实例详情的逐层深入

Technical Architecture

3.1 技术栈

前端框架

- **核心框架**: Vue 3 (Composition API)
- **构建工具**: Vite
- **TypeScript**: 全面使用TS提升代码质量
- **状态管理**: Pinia

UI与可视化

- **UI组件库**: Element Plus / Ant Design Vue (推荐Element Plus,暗色主题支持更好)
- **图表库**:
  - **主力**: Apache ECharts 5.x (支持丰富的主题定制)
  - **备选**: @antv/g2 或 Chart.js
- **拓扑图**: AntV G6 或 Cytoscape.js (用于链路拓扑)
- **图标**: @iconify/vue (包含丰富的监控类图标)

样式与动画

- **CSS预处理器**: SCSS
- **动画库**: GSAP (用于高级动画效果)
- **布局方案**: CSS Grid + Flexbox
- **响应式**: 媒体查询,支持1920px及以上屏幕优先

Mock数据

- **Mock方案**: Mock.js / Faker.js
- **数据生成策略**:
  - 时序数据生成器(模拟真实波动)
  - 随机Trace生成器(模拟服务调用链)
  - 日志生成器(模拟不同级别日志)

3.2 项目结构

```
src/
├── views/              # 页面组件
|   ├── Dashboard/      # 综合仪表盘
|   ├── Metrics/        # 指标监控
|   ├── Tracing/        # 链路追踪
|   ├── Logs/           # 日志分析
|   └── Custom/         # 自定义Dashboard
├── components/         # 通用组件
|   ├── Charts/         # 图表组件封装
|   ├── Filters/        # 筛选器组件
|   ├── TimePicker/     # 时间选择器
|   └── Layout/         # 布局组件
```

```
├── composables/          # 组合式函数
│   ├── useTimeRange.ts # 时间范围管理
│   ├── useFilters.ts    # 筛选逻辑
│   └── useChartTheme.ts# 图表主题
├── mock/                 # Mock数据
│   ├── metrics.ts
│   ├── traces.ts
│   ├── logs.ts
│   └── generators/      # 数据生成工具
├── stores/               # Pinia状态管理
├── router/               # 路由配置
├── styles/               # 全局样式
│   ├── variables.scss   # 暗色主题变量
│   └── themes/          # 主题配置
└── utils/                # 工具函数
```

3.3 设计规范

暗色主题配色 (参考Grafana)

- **主背景**: #0b0c0e ~ #181b1f
- **卡片背景**: #1f1f24 ~ #23252b
- **主色调**: #3274d9 (蓝色,用于高亮)
- **成功/健康**: #73bf69 (绿色)
- **警告**: #ff9830 (橙色)
- **错误/危险**: #f2495c (红色)
- **文字**: #d8d9da (主文字), #9fa7b3 (次要文字)

图表风格

- **配色方案**: 使用专业的监控配色(蓝、绿、黄、红渐变)
- **网格线**: 低饱和度,避免视觉干扰
- **动画**: 流畅的过渡效果(300-500ms)
- **响应式**: 图表根据容器大小自适应

交互规范

- **加载状态**: 骨架屏 + Loading动画
- **空状态**: 友好的空数据提示
- **错误状态**: 清晰的错误信息展示
- **响应速度**: 操作响应时间<100ms

3.4 核心技术实现要点

Mock数据生成策略

```
// 时序数据生成:基于正弦波+随机噪声模拟真实波动
// Trace生成:随机生成3-10层调用链,模拟微服务场景
// 日志生成:根据时间分布,模拟真实日志密度
```

性能优化

- **虚拟滚动**: 日志列表使用虚拟滚动(vue-virtual-scroller)
- **图表懒加载**: 使用Intersection Observer延迟加载图表
- **时间序列优化**: 大数据量时自动聚合采样
- **LocalStorage缓存**: 缓存用户配置和筛选条件

路由设计

```
/                       # 重定向到 /dashboard
/dashboard              # 综合仪表盘
/metrics                # 指标监控
/metrics/:service       # 服务详情
/tracing                # 链路追踪
/tracing/:traceId       # 链路详情
/logs                   # 日志分析
/custom                 # 自定义Dashboard
```

Performance & Scalability

4.1 性能指标

- **首屏加载**: < 2s (1920x1080屏幕)
- **页面切换**: < 300ms
- **图表渲染**: < 500ms (单图表)
- **数据刷新**: < 200ms (实时模式)

4.2 数据规模考虑

- **日志展示**: 支持渲染10,000+条目(虚拟滚动)
- **时序数据点**: 单图表支持1000+数据点
- **Trace展示**: 支持100层Span的链路
- **Dashboard**: 支持同屏15-20个图表

4.3 兼容性

- **浏览器**: Chrome 90+, Firefox 88+, Edge 90+
- **屏幕分辨率**: 主要适配1920x1080, 2560x1440
- **不支持移动端**(SRE工作场景以桌面为主)

4.4 扩展性考虑

- **组件复用**: 图表组件高度可配置,易于扩展新指标
- **Mock数据**: 数据生成器独立,方便替换为真实API
- **主题切换**: 预留浅色主题接口(可选实现)
- **国际化**: 预留i18n接口(可选)

交付标准

1. **代码质量**: TypeScript严格模式,ESLint + Prettier规范
2. **文档**: README包含项目说明、运行指南、Mock数据说明
3. **演示数据**: 至少3个服务的完整Mock数据(指标、Trace、日志)
4. **视觉还原度**: 达到Grafana 80%以上的视觉专业度
5. **交互流畅度**: 无明显卡顿,动画流畅自然