天猫双十一.
实时数据大屏.

设计文档

第三组

目录

一、文档概述

1.1 文档目的

本文档为天猫双十一实时数据可视化大屏项目的技术实现指南，明确系统架构、数据模型、API 设计、UI / 动效实现等核心技术方案，用于指导开发、测试、部署全流程，确保最终产品满足商业展示、技术演示、教学实践等多场景需求。

1.2 文档范围

覆盖系统从前端展示到后端数据模拟的全技术栈设计，包括架构分层、模块实现、数据流转、动效规范、API 接口、部署测试等内容，同时关联产品需求中的视觉风格、功能指标与用户体验目标。

1.3 核心目标

1. 实现多维度数据实时可视化，支持销售、仓储、物流等模块联动
2. 打造具有未来感的深色科技风界面，满足商业展示的视觉冲击需求
3. 保障数据更新流畅无卡顿，动效统一且富有节奏感
4. 支持低门槛部署与本地运行，适配展示、学习、分析等多用户场景

二、系统架构设计

2.1 整体架构分层（逻辑架构）

```
┌-------------------------┐
| 前端展示层                | React 18 + Apache ECharts 5 + GSAP 3 + CSS3
| （组件化开发+响应式布局）|
├-------------------------┤
| 通信层                   | WebSocket (socket.io-client)
| （实时数据推送+自动重连）|
├-------------------------┤
| 服务层                   | Node.js 20 LTS
| （REST API + 数据转发 + 静态资源服务）|
├-------------------------┤
| 模拟数据层                | Faker.js + 定时任务
| （可控随机数据+峰值注入+节奏模拟）|
└-------------------------┘
```

2.2 模块职责划分

层级	核心职责	关联产品需求要点
前端展示层	图表渲染（折线图 / 热力图 / 饼图等）、动效控制、响应式布局、模块联动逻辑	科技感视觉风格、数据动态更新、多维度指标整合、沉浸式体验
通信层	实时数据推送、断连自动重连、二进制数据支持	数据实时性、展示流畅度
服务层	提供静态资源（地图 / 品牌数据）、API 接口服务、模拟数据转发	模块化部署、低门槛使用
模拟数据层	生成符合节奏规范的模拟数据（销售额增长、物流流转等）、控制数据更新频率	动态数据展示、避免静态乏味、匹配活动实时性与规模感

2.3 技术选型说明

技术类别	选型工具 / 框架	选型依据与实现目标
前端框架	React 18（函数组件 + Hooks）	组件化开发提升复用性，Hooks 简化状态管理，适配多模块联动需求
可视化图表	Apache ECharts 5	支持地图、折线图、柱状图等多图表类型，满足销售热力图、物流路径图等可视化需求
动效实现	GSAP 3 + CSS3	实现数字滚动、流光动画、粒子效果，保障动效流畅度，满足节奏感与沉浸感目标
实时通信	socket.io-client	支持跨浏览器兼容，自动重连机制，保障每秒数据推送的稳定性
后端服务	Node.js 20 LTS	非阻塞 I/O 特性适配 WebSocket 实时通信，轻量化部署满足低门槛使用需求
模拟数据	Faker.js + 定时任务	生成可控的随机数据与峰值数据，模拟真实双十一数据增长趋势

地图数据	GeoJSON	提供高精度地理空间数据，支撑地区销售热力图、物流路线分布的可视化

2.4 数据流转流程

1. 模拟数据层通过 Faker.js 生成销售、仓储、物流等维度的模拟数据，按配置频率（1-5 秒）触发数据更新
2. 服务层接收模拟数据，通过 WebSocket 实时推送到前端
3. 前端展示层接收数据后，通过 React 状态管理同步至各功能组件
4. 组件调用 ECharts 渲染图表，通过 GSAP 控制动效执行（数字跳动、图表过渡等）
5. 各模块按统一节奏更新，实现全局联动，同步触发背景粒子、流光等氛围特效

三、数据模型设计

3.1 数据库表结构总览

表名	核心用途	关联产品模块	更新频率
sales	存储销售额、销售趋势、品类排行、地区分布数据	销售数据模块	1 秒 / 次
storage	存储仓库容量、出入库比例、区域仓储占比数据	仓储数据模块	3 秒 / 次
logistics	存储发货 / 签收量、配送时效、物流路线数据	物流数据模块	5 秒 / 次
users	存储活跃用户数、用户画像、地区分布数据	用户画像与活跃模块	2 秒 / 次
system	存储系统时间、全局指标、刷新配置数据	综合展示与主控模块	1 秒 / 次
search	存储搜索热词及搜索次数数据	辅助展示模块	10 秒 / 次
file_assets	存储地图、背景图、粒子贴图等静态资源元信息	全局视觉模块	部署时初始化

3.2 核心表字段设计

3.2.1 sales 表（销售数据核心表）

字段名	数据类型	说明	示例值
id	INT	主键自增 ID	1001
timestamp	DATETIME	数据生成时间（精确到秒）	2025-11-11 14:30:25
total_sales	BIGINT	当前累计销售额（单位：元）	181266281130
sales_trend	JSON	近 60 分钟销售额数组（每分钟 1 个数据点，用于折线图）	[1250000, 1320000, ..., 1580000]
category_ranking	JSON	品类 TOP10 销量排行（含品类名称、销量、占比）	{"手机": {"sales": 25600, "rate": 18%}, ...}
region_heatmap	JSON	各省销售额数据（用于地图热力图渲染）	{"浙江": 15600000, "广东": 18900000, ...}

3.2.2 storage 表（仓储数据核心表）

字段名	数据类型	说明	示例值
id	INT	主键自增 ID	2001
timestamp	DATETIME	数据生成时间	2025-11-11 14:30:27
warehouse_status	JSON	各仓库状态（容量占比、出入库量）	{"华东 A 仓": {"capacity": 78%, "in": 5600, "out": 4800}, ...}
region_storage_ratio	JSON	各大区仓储占比（用于饼图展示）	{"华东": 45%, "华南": 35%, "华北": 15%, "其他": 5%}

3.2.3 logistics 表（物流数据核心表）

字段名	数据类型	说明	示例值

id	INT	主键自增 ID	3001
timestamp	DATETIME	数据生成时间	2025-11-11 14:30:29
shipped	INT	当日累计发货件数	123233445
delivered	INT	当日累计签收件数	89654321
avg_delivery_time	FLOAT	全国平均配送时长（单位：小时）	18.5
routes	JSON	发货路线数据（含出发地、目的地、件数）	{"杭州→上海": 12500, "广州→深圳": 18600, ...}

3.2.4 其他核心表关键字段摘要

- users 表：active_users（当前活跃用户数）、demographics（性别 / 年龄 / 设备占比 JSON）、geo_distribution（地区活跃用户 JSON）
- system 表：current_time（系统当前时间）、global_metrics（全局指标汇总 JSON）、refresh_config（各模块刷新频率配置）
- search 表：keyword（搜索关键词）、count（累计搜索次数）、last_search_time（最近搜索时间）
- file_assets 表：file_id（文件唯一标识）、file_name（文件名）、file_type（资源类型：map/bg/particle/font）

3.3 数据关联关系

- 时间关联：所有业务表通过 timestamp 字段与 system 表的 current_time 同步，确保数据更新节奏一致
- 模块联动：sales 表的 total_sales 与 users 表的 active_users 联动，同步触发前端对应模块动效
- 资源关联：file_assets 表存储的地图 JSON 文件与 sales、logistics 表的地区数据关联，支撑地理可视化渲染

四、API 接口设计

4.1 接口总体规范

- 协议：HTTP/HTTPS（REST API）、WebSocket（实时推送）
- 数据格式：请求与响应均为 JSON 格式（文件上传除外）
- 字符编码：UTF-8

- 响应状态码：200（成功）、400（参数错误）、404（接口不存在）、500（服务器错误）
- 实时推送频率：WebSocket 接口每秒推送 1 次最新数据汇总

4.2 REST API 接口（静态数据 / 配置管理）

4.2.1 销售数据接口

接口路径	请求方法	功能描述	请求参数	响应示例
/api/sales/list	GET	列出全部销售记录	page=1&size=20	{"code":200,"data":{"list":[{"id":1001,"timestamp":"2025-11-11 14:30:25","total_sales":181266281130,...}],"total":1000}}
/api/sales/latest	GET	获取最新 1 条销售记录	无	{"code":200,"data":{"id":1001,"timestamp":"2025-11-11 14:30:25","total_sales":181266281130,"sales_trend":[...]}}
/api/sales/add	POST	新增销售记录（测试用）	Body:{timestamp, total_sales, ...}	{"code":200,"msg":"新增成功","data":{"id":1002}}

4.2.2 核心业务模块接口摘要

模块	核心接口	方法	功能描述
仓储模块	/api/storage/latest	GET	获取最新仓储数据
物流模块	/api/logistics/latest	GET	获取最新物流数据
用户模块	/api/users/latest	GET	获取最新用户活跃与画像数据
系统配置	/api/system/status	GET	获取系统时间与全局指标
系统配置	/api/system/update_config	PUT	修改模块刷新频率配置
搜索热词	/api/search/hot	GET	获取 TOP N 搜索热词
资源管理	/api/assets/list_by_type	GET	按类型获取静态资源(地图 / 背景等)
资源管理	/api/assets/upload	POST	上传静态资源

			（MultipartFile）

4.3 WebSocket 实时推送接口

接口路径	通信方向	功能描述	推送数据格式示例
/ws/realtime	服务端→客户端	每秒推送销售、仓储、物流、用户、系统的最新数据汇总，支撑前端实时渲染	{"sales": {...}, "storage": {...}, "logistics": {...}, "users": {...}, "system": {...}}

五、前端实现设计

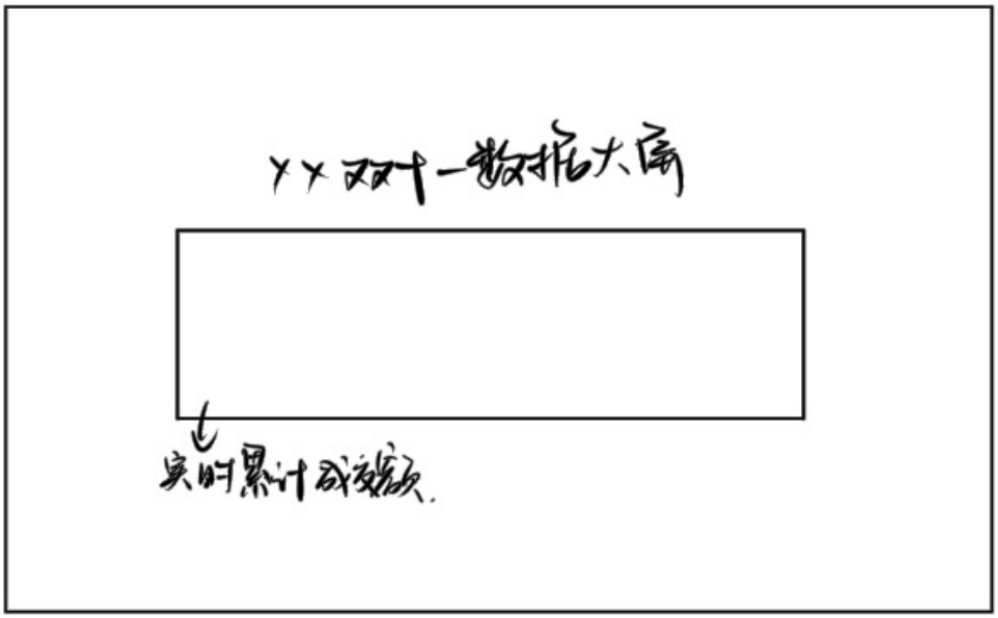
5.1 页面布局设计

5.1.1 整体布局结构

采用 “主屏+副屏” 的模块化布局，适配全屏展示场景：
主屏：滚动展示实时销售数据（累计销售额）
副屏：

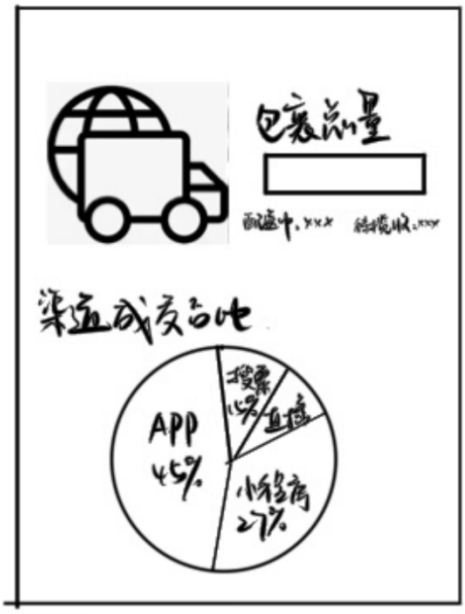
- 左上：销售概览
- 左下：用户活跃区
- 右上：仓储
- 右下：渠道成交

一、主屏

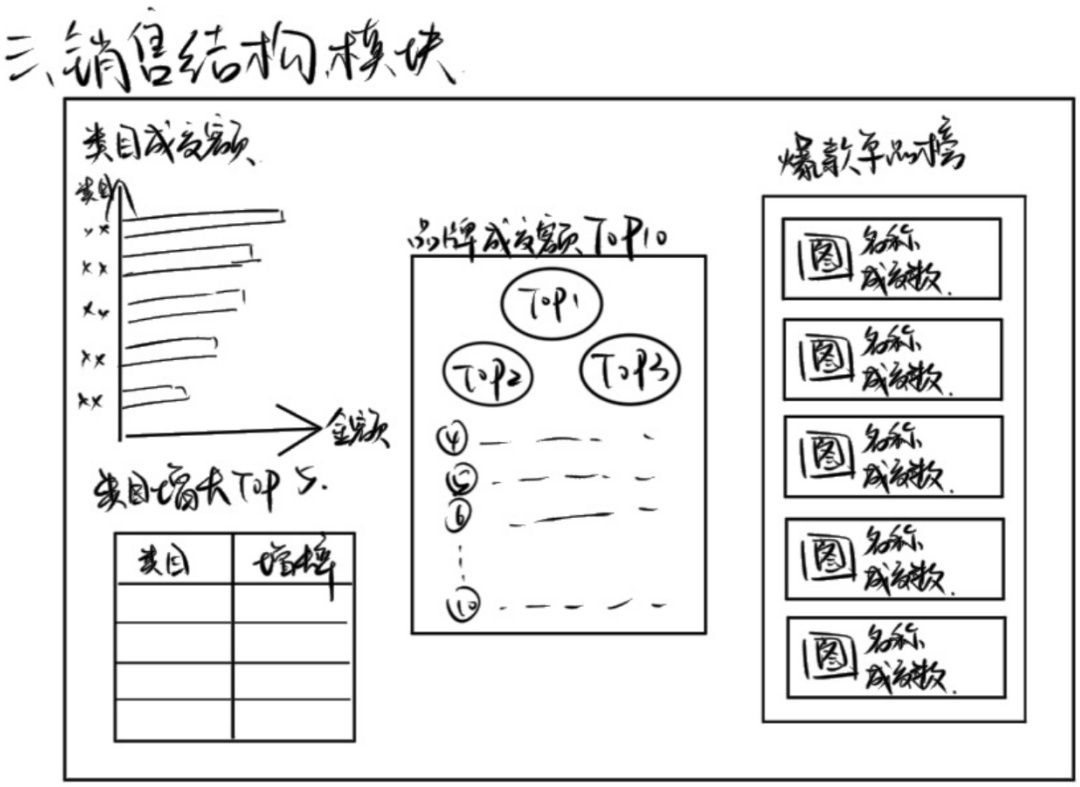

××双十一数据大屏
实时累计成交额

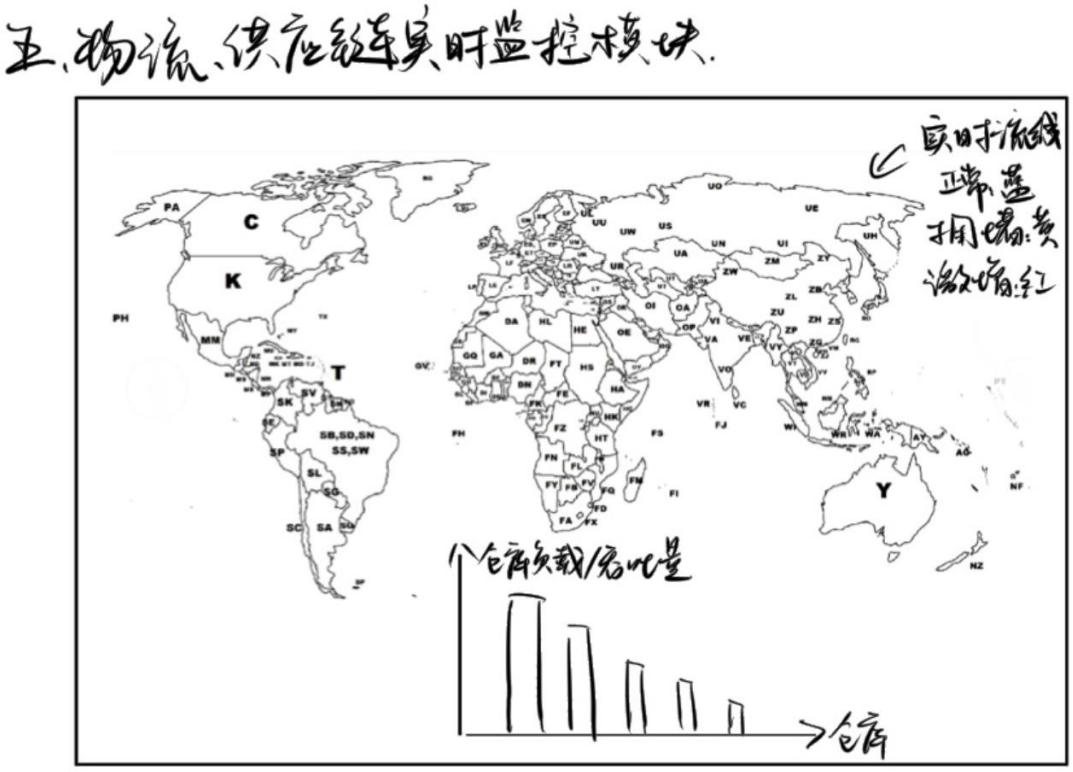
二、副屏


编号信息
美国
美国
美国
美国
美国
美国
美国
国内（热力示意）

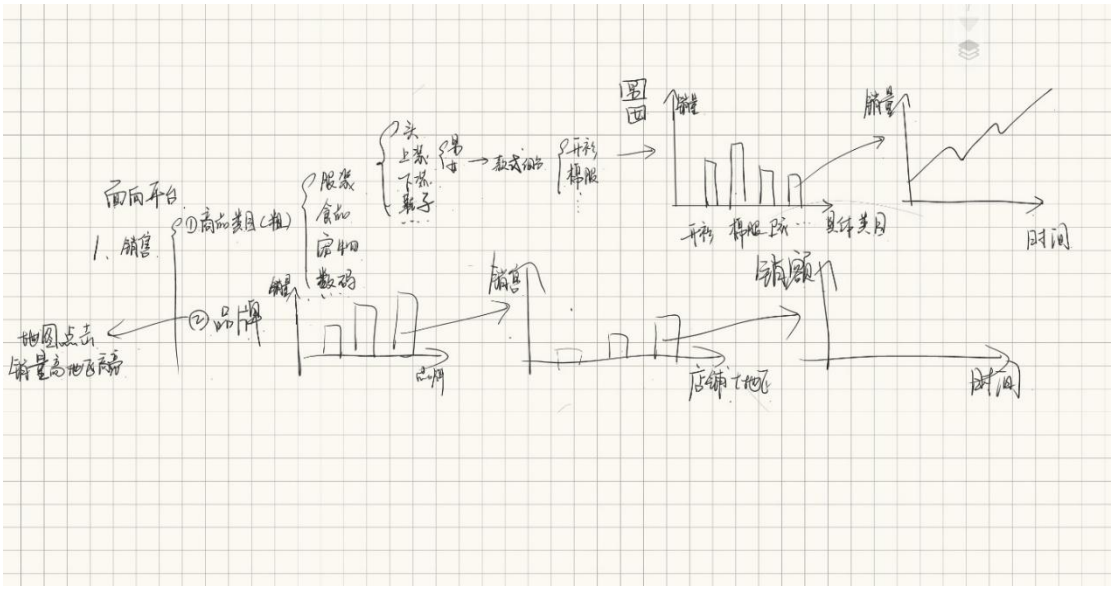

包裹总量
渠道成交占比
APP
45%
小程序
27%

5.1.2 具体核心模块设计





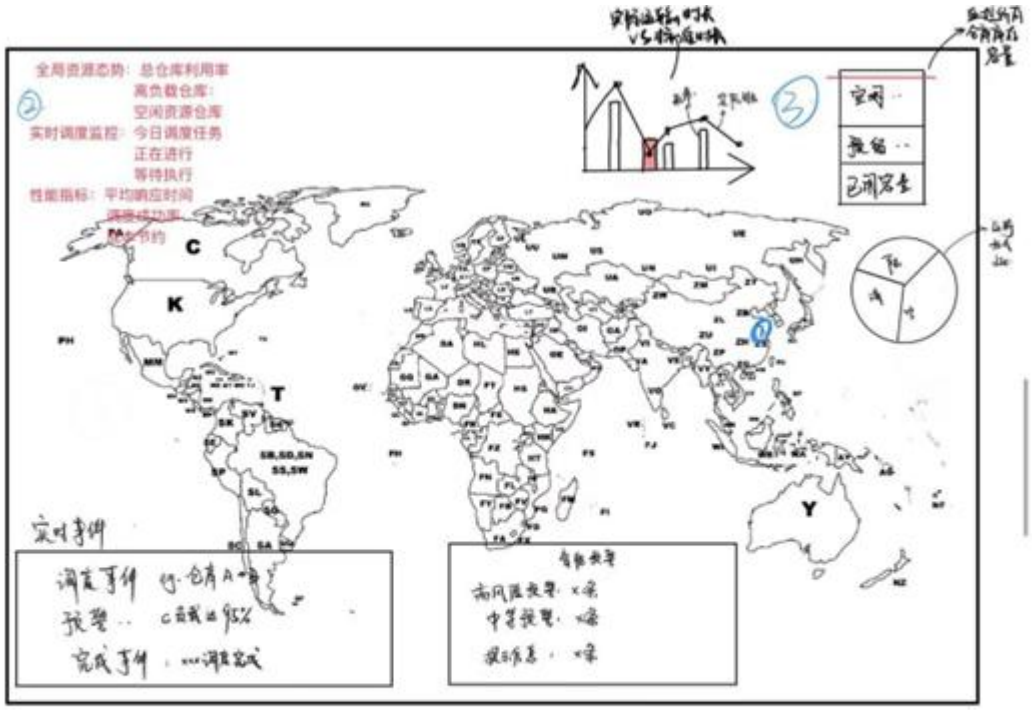
然后再在核心模块的基础下进行更加详细的展示；



销售模块：

在原有的基础上细化，可以点击查看某一类目更详细的品类销售情况。如：可以查看服装-上衣 的不同品类的销量，针对某一品如卫衣可以点击查看时间-销量图。

除了根据品类来细化，还可以看某一品牌的全国各地门店的销售情况，点击之后可以在地图上通过颜色的深浅来反馈，还可以查看具体店铺的销售额。



地图功能：仓库状态：大小表示容量规模，颜色表示负载状态（绿<70%/黄70-90%/红>90%），脉冲动画表示异常告警

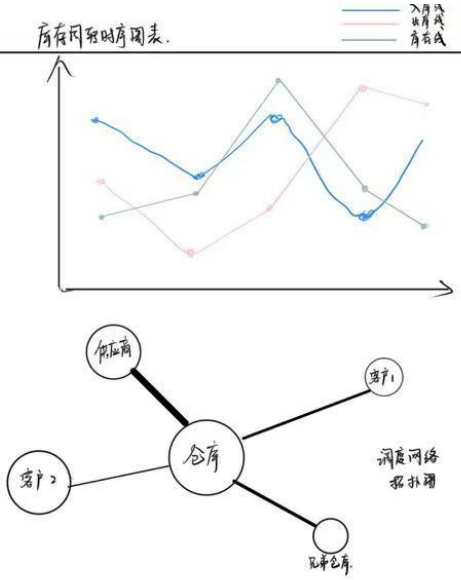
实时运输网络：动态流向线条展示货物流向，线条粗细表示运输量级，颜色区分运输状态（正常/延迟/紧急）

智能调度预测：自动高亮推荐调度路径，风险扩散可视化

仓储模块：
上级页面点击后进入新增的仓库详情、全局调度、深度分析界面。
仓库详情界面对某一仓储建立仓储档案，进行实时运营的反馈，当前仓库的订单处理情况，效率指标，实时监控是否爆单，并且展示可支援的仓库信息。
全局管理界面用需求热力图层上的颜色深浅来反馈不同区域的订单密度，客户需求强度，通过运输路线实时监控全局货物流向，并对执行进度进行追踪。
深度分析用图表形式分析该仓库的订单物流和结构信息。
仓储模块：
上级页面点击后进入新增的仓库详情、全局调度、深度分析界面。
仓库详情界面对某一仓储建立仓储档案，进行实时运营的反馈，当前仓库的订单处理情况，效率指标，实时监控是否爆单，并且展示可支援的仓库信息。
全局管理界面用需求热力图层上的颜色深浅来反馈不同区域的订单密度，客户需求强度，通过运输路线实时监控全局货物流向，并对执行进度进行追踪。
深度分析用图表形式分析该仓库的订单物流和结构信息。

① 仓库详情界面（点击地图进入）

基础信息：
仓库档案：
容量数据：总面积/可用容量/当前库存
设备状态：装卸设备/仓储系统/安防
人员配置：在岗人数/技能分布
实时运营：
进出库监控：今日入库/出库/在库
订单处理：待处理/进行中/已完成
效率指标：处理速度/准确率
调度关联：
可支援仓库：3个 [点击查看详情]
被支援记录：本月15次



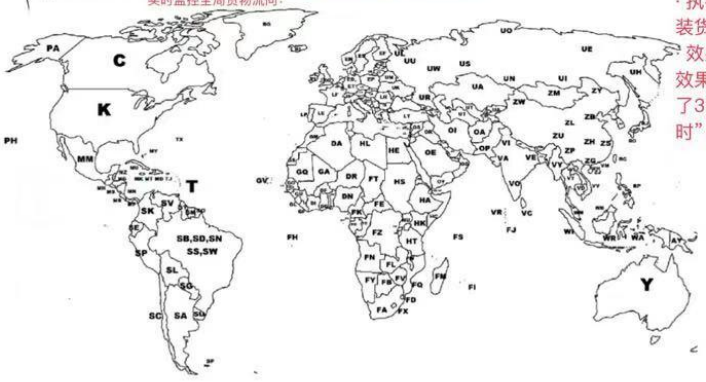
② 全局调度管理界面（点击运营看板进入）

按钮

资源态势图层：仓库、车辆、人员位置和状态

需求热力图层：用颜色深浅来标识不同区域的订单密度、客户需求强度。

调度路径图层：在地图上画出货物从A仓库到B仓库的实际或计划运输路线。
实时监控全局货物流向：



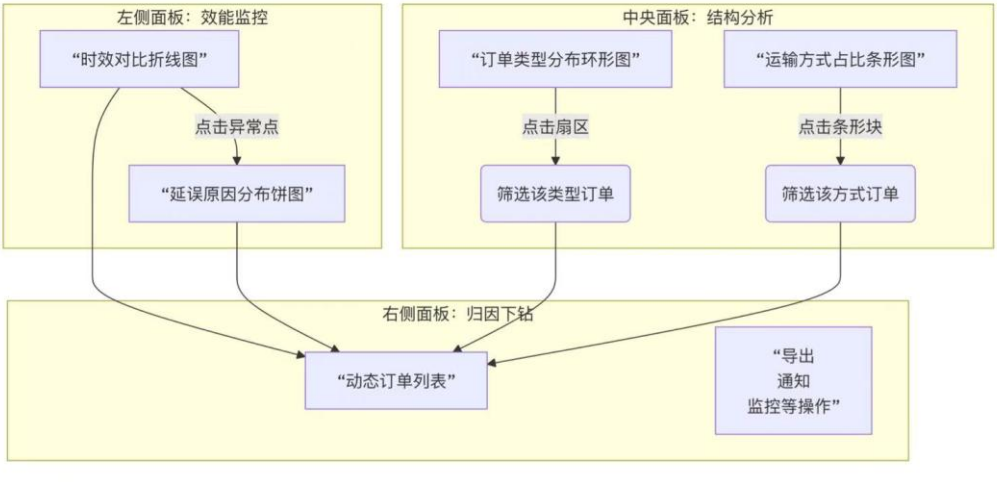
调度执行监控（跟踪任务）

一旦调度指令发出，就在这里监控全过程。

· 任务队列：所有已创建但未执行的调度任务列表。
· 执行进度：每个任务的状态（待处理、装货中、运输中、已送达）
· 效果追踪：任务完成后，自动计算实际效果，如“本次调度实际成本比预期节省了3%”或“因交通拥堵，时效延误了1小时”。

任务	备注	状态

③ 深度分析界面（点击图表进入）



仓储模块：
上级页面点击后进入新增的仓库详情、全局调度、深度分析界面。
仓库详情界面对某一仓储建立仓储档案，进行实时运营的反馈，当前仓库的订单处理情况，效率指标，实时监控是否爆单，并且展示可支援的仓库信息。
全局管理界面用需求热力图层上的颜色深浅来反馈不同区域的订单密度，客户需求强度，通过运输路线实时监控全局货物流向，并对执行进度进行追踪。
深度分析用图表形式分析该仓库的订单物流和结构信息。

5.2 视觉风格实现

5.2.1 基础风格规范

- 配色方案：深色背景（#0a0e17）+ 霓虹强调色（金色 #ffd700、青蓝色 #00f0ff、紫色 #9d00ff）
- 界面元素：半透明玻璃态卡片（opacity: 0.85）、发光描边（text-shadow: 0 0 8px 强调色）
- 背景效果：粒子流动态背景（GSAP 控制粒子运动轨迹）、流光渐变层

5.2.2 模块视觉差异化

模块	主色调	视觉元素	动效亮点
销售模块	金色 + 红橙渐变	大尺寸跳动数字、热力图	数字脉冲放大、折线流光描边
仓储模块	冷蓝 + 亮青色渐变	环形能量条、饼图	容量饱和时闪烁警示、能量条呼吸动效
物流模块	蓝紫色 + 白色光迹	地图路径线、双柱对比图	粒子流路径动画、指针平滑旋转
用户模块	亮青 + 中性灰	半透明数据卡片、环形图	数字滚动、图表平滑切换
综合模块	多色霓虹渐变	顶部时间条、全局指标卡	背景粒子雨、徽标闪烁

5.3 动效实现设计

5.3.1 核心动效规范

- 帧率要求：≥60FPS，确保视觉流畅无卡顿
- 过渡时长：动画过渡≤300ms，避免延迟感
- 节奏统一：各模块刷新频率与全局同步（销售 1 秒 / 次、仓储 3 秒 / 次等）
- 性能控制：CPU 占用≤40%，避免资源浪费

5.3.2 关键动效实现方案

动效类型	实现技术	应用场景	效果描述
数字跳动	GSAP TweenMax	销售额、活跃用户数	数值从当前值平滑过渡到目标值，伴随轻微放大→还原的脉冲效果
折线图渐变	ECharts 动画配置 + GSAP	销售趋势展示	折线随数据更新平滑上升，线条带有流光描边效果
粒子流动画	GSAP + Canvas	背景、物流路线	粒子按预设路径运动，模拟数据流动与网络覆盖
图表切换	CSS3 transition + ECharts 重绘	品类排行、用户画像	图表数据更新时无闪烁，平滑过渡到新状态
模块联动高亮	CSS3 :active 伪类 + GSAP	跨模块数据关联	点击某模块时，关联模块同步高亮，强化整体联动感

5.4 组件设计

采用 React 函数组件 + Hooks 实现组件化开发，核心组件划分如下：

- 基础组件：数字滚动组件、图表渲染组件、粒子背景组件、地图组件
- 业务组件：销售总览组件、销售趋势组件、仓储状态组件、物流路径组件、用户画像组件
- 全局组件：顶部综合控制组件、模块联动控制组件、动效管理组件

六、数据模拟设计

6.1 模拟数据生成规则

- 销售额：基础增长曲线 + 随机峰值注入，模拟双十一销售爆发特征，确保累计值持续上升
- 物流数据：发货量与销售额正相关，签收量滞后发货量 1-2 小时，配送时效随单量动态调整
- 仓储数据：出入库量与销售趋势同步，仓库容量控制在 60%-95% 区间，避免溢出或空置
- 用户数据：活跃用户数随销售额增长而上升，用户画像比例保持相对稳定（如无线成交占比 86%）

- 搜索热词：随机生成与电商相关的关键词，部分关键词按销售排行加权，模拟热门商品搜索行为

6.2 数据节奏控制

通过定时任务精准控制各模块数据更新频率，匹配产品需求的节奏感：

- 销售模块：1 秒 / 次（核心指标，强化实时增长感知）
- 用户模块：2 秒 / 次（辅助指标，保持活跃度感知）
- 仓储模块：3 秒 / 次（运营指标，平衡节奏与性能）
- 物流模块：5 秒 / 次（路径动画较复杂，控制刷新频率）
- 全局联动：所有模块数据更新时，触发顶部综合区的氛围动效同步

七、测试与性能优化设计

7.1 测试要点

- 功能测试：验证各模块数据展示准确性、动效完整性、模块联动有效性
- 性能测试：监控帧率（≥60FPS）、CPU 占用（≤40%）、内存使用，确保长时间运行无泄漏
- 兼容性测试：适配 Chrome、Firefox 等主流浏览器，支持 1920×1080 及以上分辨率
- 稳定性测试：连续运行 8 小时，验证数据推送无中断、动效无卡顿、界面无崩溃

7.2 性能优化方案

- 前端优化：图表数据分片加载、动效节流控制、非可视区域组件懒加载
- 通信优化：WebSocket 二进制传输、数据压缩、无效数据过滤
- 渲染优化：减少 DOM 操作、使用 CSS3 硬件加速、ECharts 图表缓存配置

八、部署与运维设计

8.1 部署环境要求

- 前端部署：Nginx 服务器，支持静态资源访问与 WebSocket 代理
- 后端部署：Node.js 20 LTS 环境，支持 PM2 进程守护
- 运行环境：客户端浏览器支持 ES6+、WebSocket 协议，网络带宽≥1Mbps

8.2 部署流程

1. 前端构建：执行 `npm run build` 生成静态资源包，部署至 Nginx 根目录
2. 后端部署：上传 Node.js 服务代码，通过 PM2 启动服务（`pm2 start app.js`）
3. 配置 Nginx：代理 WebSocket 请求，确保实时数据推送正常
4. 初始化：导入地图 GeoJSON、背景图等静态资源，启动数据模拟服务

8.3 运维监控

- 服务监控：通过 PM2 监控 Node.js 进程状态，异常自动重启
- 日志管理：记录接口访问日志、数据推送日志，便于问题排查
- 配置管理：支持通过`/api/system/update_config` 接口远程调整刷新频率等参数

九、需求映射与风险控制

9.1 产品需求 - 技术实现映射表

产品需求点	技术实现方案
多维数据实时展示	WebSocket 每秒推送 + ECharts 实时渲染
深色科技风视觉体验	深色背景 + 霓虹配色 + 粒子动效 + 发光描边
模块联动与统一节奏	全局动效管理组件 + 统一刷新频率配置
低门槛部署与本地运行	前后端分离架构 + 本地 Node.js 环境 + 静态资源打包
数据动态跳动与平滑过渡	GSAP 动画库 + ECharts 平滑过渡配置

9.2 风险与应对方案

潜在风险	应对方案
大量数据推送导致前端卡顿	数据分片处理 + 动效节流 + 性能监控优化
浏览器兼容性问题	前端兼容性适配 + 明确浏览器版本要求
WebSocket 断连导致数据中断	自动重连机制 + 断连提示 + 数据补发逻辑
本地部署配置复杂	提供部署文档 + 一键启动脚本