本项目是一个前端可视化系统，主要用于展示热浪事件及相关数据，同时集成了缓存管理机制、动态地图渲染和数据可视化。项目中使用了 Vue.js、Element UI、Leaflet、IndexedDB 等技术，对数据进行预处理和可视化呈现。

1. 项目所需安装配置介绍

项目基于 Node.js 开发，建议安装 Node.js 版本 14.x 或更高版本。

（1）**依赖包安装**

使用 npm 或 yarn 管理依赖。在项目根目录下运行以下命令安装依赖：

npm install或yarn install

（2）**ESLint 配置**

项目中集成了 ESLint 用于代码风格检查。请确保编辑器中安装 ESLint 插件以便实时提示错误。

2. 项目运行环境

**操作系统**：Windows 11（兼容 Linux/MacOS）

**运行时环境**：Node.js (14.x 或更高版本)

**前端框架**：Vue.js（Vue 3）

**其他库：**

- Leaflet 用于地图和热力图渲染

- Element UI 用于 UI 组件（如按钮、切换、日期选择器等）

- IndexedDB 用于本地数据缓存

3. 项目运行方式

在项目根目录下，运行以下命令启动开发服务器：

npm run serve

默认情况下，开发服务器会运行在 `http://localhost:8080`。运行之后，可在浏览器中访问该地址进行项目预览。

4. 项目目录结构描述

frontend

├── public/

│ ├── data/

│ │ ├── eventually\_heatwave.csv //

│ │ └── final\_heatwaves.geojson // 热浪事件数据文件（GeoJSON 格式）

│ └── index.html // HTML 模板

├── src/

│ ├── assets/ // 图片、样式等静态资源

│ ├── components/ // Vue 组件

│ │ └── CacheMonitor.vue // 缓存监控组件，用于显示缓存命中率和状态  
│ ├── modules/

│ │ └── history.js // 管理和持久化储存操作历史记录

│ ├── layout/

│ │ └── MainLayout.vue //布局的组件,包裹了 Task3 和 operation\_record

│ ├── lib/ // 工具库

│ │ └── cacheManager.js // 缓存管理器（基于 IndexedDB）及缓存策略

│ ├── views/ // 页面视图

│ │ ├── Task1

│ │ │ └── Task1.vue // 任务1视图（时空分析）

│ │ ├── Task2

│ │ │ └── Task2.vue // 任务2视图（热浪强度可视化与缓存应用）

│ │ └── Task3

│ │ ├── opration\_record.vue //管理用户操作记录的视图

│ │ └── Task3.vue // 任务3视图（关联分析）

│ ├── App.vue // 根组件

│ ├── main.js // 应用入口，初始化 Vue、路由、状态管理等

│ └── router.js // 路由配置

├── package.json // 项目依赖及脚本配置

├── jsconfig.json //用于配置 JavaScript 项目的模块解析、路径别名等

└── vue.config.js //用于Vue CLI项目的配置

5. 项目关键模块介绍

（1）Task1.vue

Task1.vue 主要负责基于地图当前视图，通过计算网格区块来确定需要加载的数据区域，然后从本地 GeoJSON 数据中过滤并加载适合该区块的热浪事件多边形，同时统一设置要素样式和交互事件，确保地图上静态展示的热浪事件与视图范围同步更新。设置了动态播放热浪事件多边形变化过程的功能。

（2）Task2.vue

实现热浪事件可视化、热力图渲染以及数据过滤功能。利用 Leaflet 渲染地图，支持切换显示多边形和热力图；并通过 GeoJSONFixer 对原始数据进行预处理。

（3）Task3.vue

主要实现热浪事件数据的综合分析看板，利用 ECharts 渲染多种图表（如持续时间分布、异常趋势、涡旋覆盖比例、盐度异常与热浪强度关联及时间线分析），通过加载并解析 CSV 数据，为用户提供多角度互动式的数据可视化界面，从而帮助发现热浪事件背后的时空模式和特征。

（4）operation\_record.vue

组件用于展示用户在热浪事件可视化过程中产生的操作记录，通过从 Vuex 历史记录模块中读取数据，并以卡片形式显示每条操作的类型、时间和详情，当没有操作记录时则提示“暂无操作记录”。

（5）cacheManager.js

该模块基于 IndexedDB 实现了数据缓存机制，支持数据的存取和自动清理。项目中采用了基于时间和 LRU (最近最少使用) 策略的缓存清理算法，提升数据读取效率并降低服务器请求。

6. 项目 AI 赋能点介绍

（1）项目利用AI 帮助对输入数据进行预处理和修正

通过 `GeoJSONFixer` 对原始数据中常见的格式问题（如单引号、None、缺失分隔符等）进行预处理，确保数据符合 GeoJSON 标准，便于后续可视化展示。利用AI帮助实现对事件数据进行智能筛选，比如根据事件持续时间、累计异常值和最大异常值自动过滤低质量数据，为后续的地图展示提供更精准的数据输入。

（2）时空动画生成与渲染优化

基于AI生成的时空序列处理代码，实现热浪事件动态演变过程的可视化。

（3）IndexedDB缓存管理​

利用AI编写代码来实现IndexedDB缓存最近访问的数据块，采用了混合策略（时间过期+LRU）来管理缓存数据。

7. 参考资料与借鉴资源

（1）Vue.js 官方文档：https://vuejs.org/guide/introduction.html

（2）Leaflet 官方手册: https://leafletjs.com/reference.html

（3）Flask 官方教程：https://flask.palletsprojects.com/en/3.0.x/tutorial

（4）Element Plus 组件库. UI 控件使用示例 [EB/OL]. https://element-plus.org/en-US/

8. 项目运行过程中的问题与解决方案

问题1：地图图层与热力图显示问题，Leaflet 图层加载失败或热力图数据不更新

解决方案：可能与网络请求、图例更新、Canvas 补丁等有关。建议检查网络状态、确保 Leaflet 及相关插件正确加载。

问题2： IndexedDB 初始化或数据缓存问题，部分浏览器在隐私模式下可能无法正常使用 IndexedDB

解决方案：确保在兼容的浏览器中运行项目，并在加载数据时增加缓存有效性检查和错误处理。