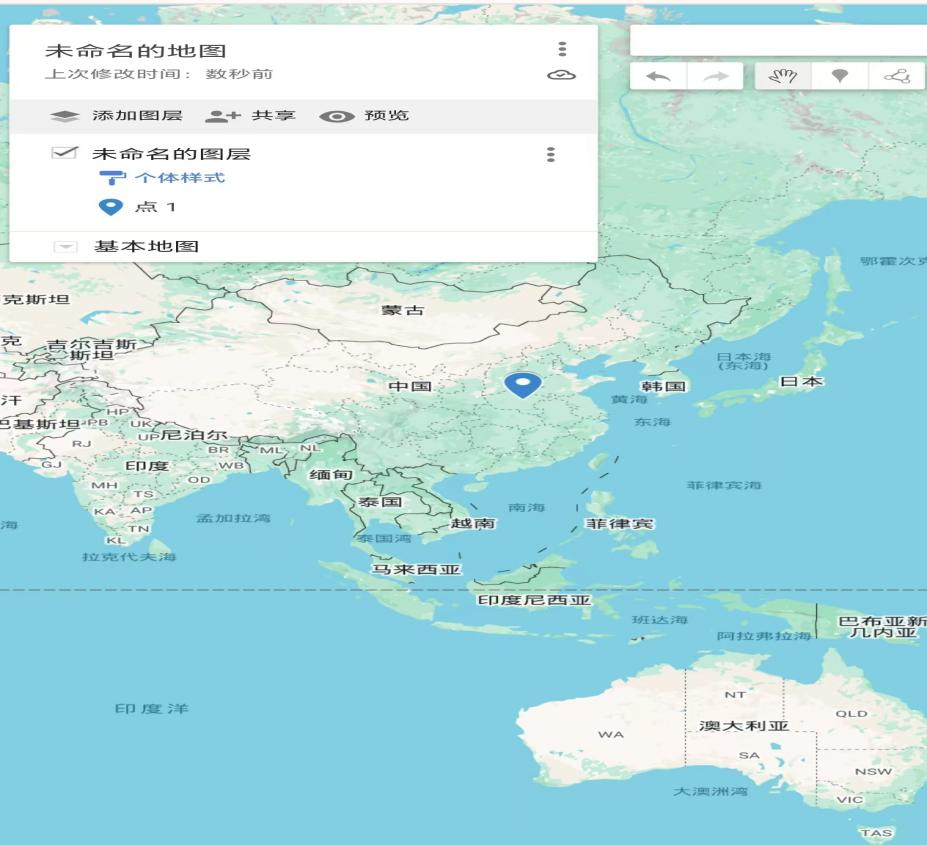
**4D gis标记系统**

1. **市场调研**
2. Google my maps

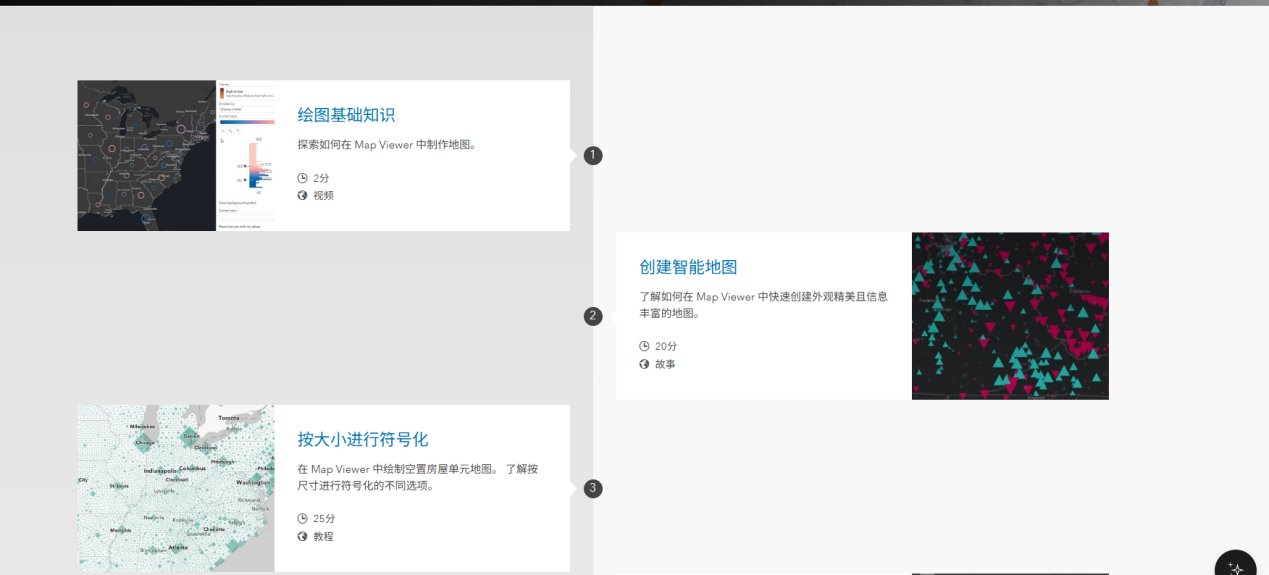
优势：强大的地图数据，免费使用

劣势：功能相对简单，没有时间维度和高度维度，自定义功能有限

****

1. ArcGIS Online

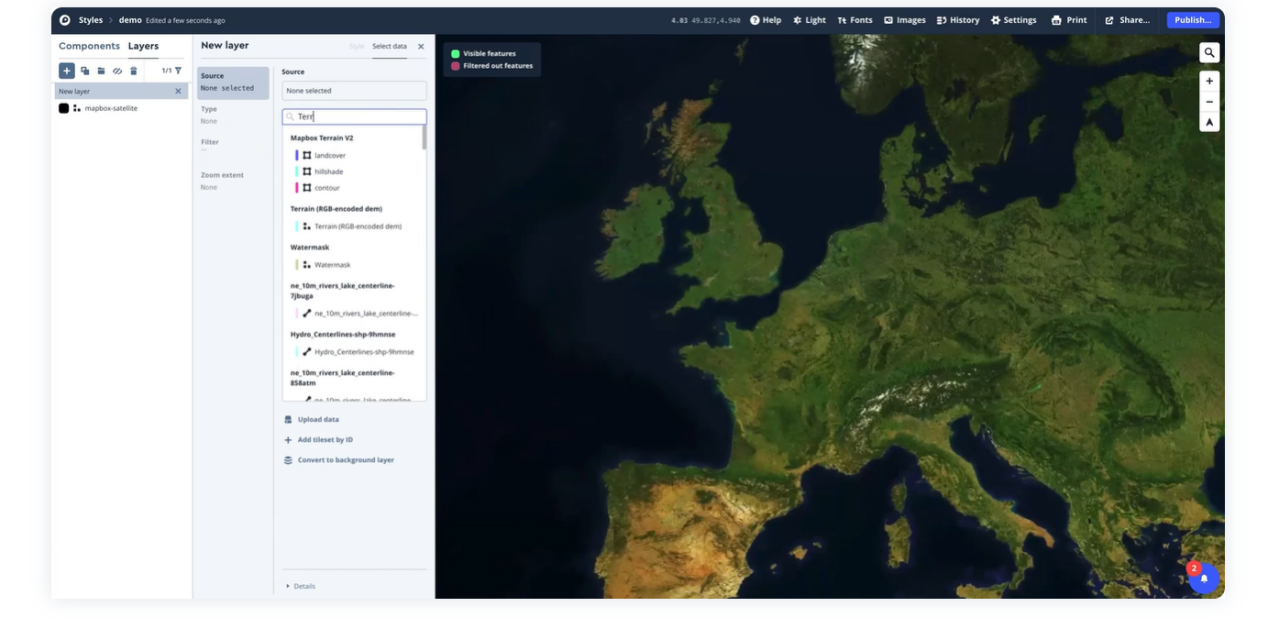
优势：专业的GIS功能，强大的分析工具，丰富的数据源  
 劣势：价格昂贵，对个人用户不友好，丰富的数据源



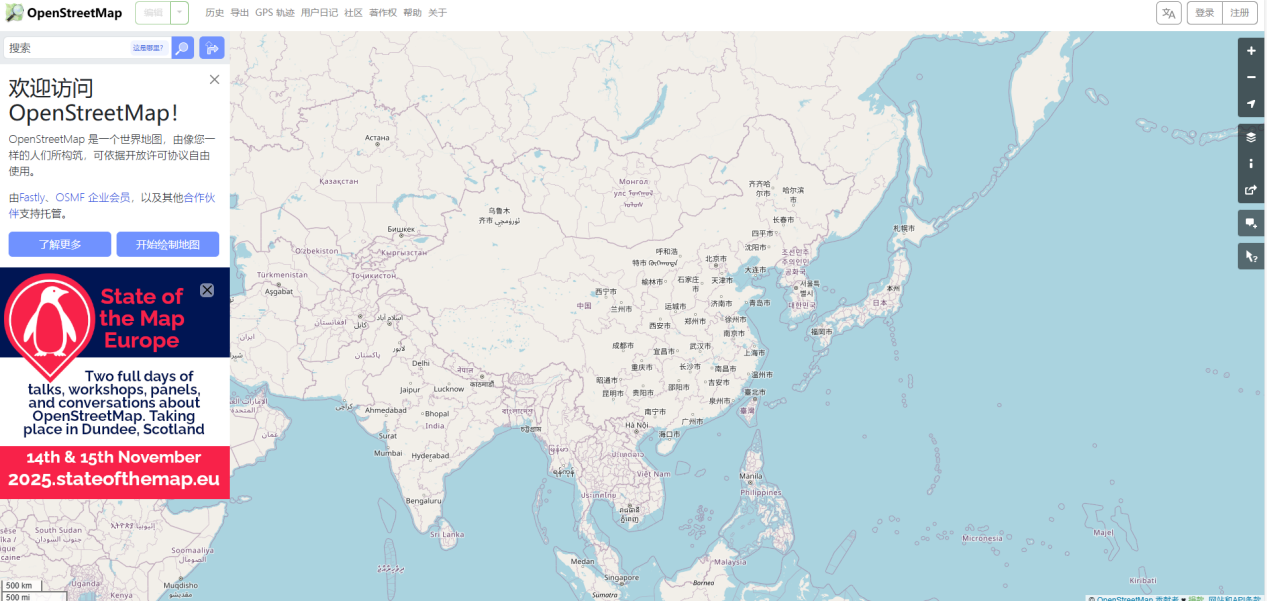
1. Mapbox Studio

优势：强大的3D功能，自定义地图样式，开发者友好

劣势：标记管理功能弱，价格较高



1. Open Street Map  
    优势：开源免费，强大可视化，支持标记

劣势：标记内容不可编辑，需要技术背景  


5.高德地图（AR打卡）  
 优势：使用方便，打卡标记功能完善

劣势：无高度空间维度，需要上传照片  


**二.4D GIS 标记系统需求规格说明书**1.引言  
1.1 项目背景  
 本项目旨在开发一款低成本、易操作的 4D GIS 标记系统，实现 “空间定位 + 属性标注 + 时间追踪 + 标记分享” 的一体化功能。  
1.2 文档目的  
 本文档明确系统的功能需求、非功能需求、数据规范及验收标准，作为开发、测试与验收的依据。

1. 项目需求  
    2.1 核心功能需求  
    2.1.1多维度地图显示  
    需求描述：支持2D平面地图显示，//支持3D立体地图显示，支持地图缩放，拖拽，旋转操作，地图样式的切换  
    2.1.2 4D标记点管理  
    需求描述：支持点击地图添加标记点，支持标记点四维数据：经度，纬度，高度，时间，支持标记点分类和颜色自定义，支持标记点编辑内容和删除操作，支持修改标记点查看范围  
    2.1.3 标记类型自定义与扩展  
    需求描述：用户可根据需求自定义新的标记类型，系统应提供灵活的扩展接口和工具，方便添加新的标记样式和属性。  
    2.1.4 多维度过滤系统  
    需求描述：时间维度过滤（按照时间范围筛选标记点），高度维度过滤（按照海拔高度筛选标记点），多维度组合过滤（时间+高度联合过滤），实时过滤统计显示  
    2.1.5 跨平台客户端支持  
    需求描述：Web端（现代浏览器支持），Android端，Ios端，//PC端  
      
    2.2扩展功能需求  
    2.2.1 数据管理功能  
    需求描述：本地数据存储和备份，数据导入导出功能，多数据源整合//服务器  
    2.2.2 协作功能  
    需求描述：多用户//协作编辑，数据共享和权限管理  
    2.2.3 离线标记功能  
    需求描述：允许用户在离线状态下进行标记操作，标记数据在本地存储，待网络恢复后再进行同步到服务器  
    2.2.4 在线集中上传  
    需求描述：支持将本地标记数据在线集中上传到服务器，实现数据的共享和统一管理  
      
    2.3 性能需求  
    2.3.1 响应时间  
    需求描述：系统对用户的操作响应时间小于2秒，如创建标记，查询等操作  
    2.3.2 数据处理能力  
    需求描述：能够高效的处理大规模的标记数据，保证系统正常运行和响应速度

2.4 界面需求  
 需求描述：系统界面应简洁、直观、易用，采用合理的布局和色彩搭配，各功能模块的操作流程应符合用户习惯，提供清晰的操作提示和帮助文档。

2.5 其他需求  
 2.5.1 安全性需求

需求描述：系统应具备用户认证、授权功能，确保只有授权用户才能访问和操作标记数据，采用加密技术保护数据传输和存储的安全。  
 2.5.2 可维护性需求  
 需求描述：系统应具备完善的日志记录功能，便于系统维护和故障排查，代码应具有良好的可读性和可维护性。  
   
  
  
  
  
  
**三.4D GIS 标记系统技术框架文档**1.技术框架分层  
**1.1 表现层**  
 负责与用户交互，展示地图以及标记信息，接受用户操作指令  
 ·2D地图展示：预计采用Open Street Map开源库，加载Open Street Map数据，实现地图的基本操作，如缩放、平移等。通过 CSS 和 HTML 进行样式定制，提升用户视觉体验。  
 //·3D地图展示：预计采用 Mapbox GL JS 库调用 Mapbox Studio 接口，实现 3D 地图的渲染和交互，如 3D 视角切换、模型展示等。

·用户交互界面：采用 Vue.js 或 React.js 等前端框架进行开发，设计直观简洁的操作界面，包括标记创建、编辑、查询等功能按钮，以及时间轴控制组件。  
**1.2业务逻辑层**

处理系统的核心业务逻辑，包括标记的管理、数据查询等。

·标记管理模块：负责标记的创建、编辑、删除操作，根据标记类型（2D、3D、4D）进行相应处理。例如，对于 4D 标记，记录其在不同时间点的空间位置和属性信息。

·数据查询模块：接收用户的查询请求，根据关键词、空间范围、时间区间等条件，从数据存储层获取相关标记数据，并进行整理和返回。

**1.3数据访问层**

负责与数据存储层进行交互，实现数据的读取、写入和更新。

·数据读取：从数据库或文件系统中读取地图数据（OpenStreetMap 数据、Mapbox 相关数据）和标记数据，根据业务逻辑层的需求进行格式转换和预处理。

·数据写入：将用户创建或编辑的标记数据写入数据库或文件系统，确保数据的完整性和一致性。

**1.4数据存储层**

用于存储地图数据和标记数据。

·地图数据存储：OpenStreetMap 数据以合适的格式（如 PostGIS 数据库结合 OSM 导入工具）存储，方便快速查询和加载。Mapbox 相关数据根据 Mapbox 的规范进行存储和管理。

·标记数据存储：采用关系型数据库（如 MySQL、PostgreSQL）或 NoSQL 数据库（如 MongoDB），存储标记的空间位置、属性信息以及时间维度信息。

2.关键技术组件

**2.1 OpenStreetMap 数据处理**

·数据获取：通过官方提供的下载工具或 API 获取 OpenStreetMap 数据，包括地图矢量数据、POI 信息等。

·数据转换与加载：使用 GDAL 等工具将获取的数据转换为适合系统使用的格式，并加载到数据库或内存中。

**//2.2 Mapbox Studio 接口应用**

·地图创建与配置：在 Mapbox Studio 平台上创建 3D 地图，设置地图样式、地形、建筑物等要素。

·接口调用：在前端通过 Mapbox GL JS 库调用 Mapbox Studio 接口，实现 3D 地图的实时渲染和交互功能。

**2.3时空数据管理**

·时间维度处理：在数据库设计中，为标记数据添加时间字段，记录标记的创建时间、有效时间范围等信息。在业务逻辑层，根据时间维度进行数据筛选和展示。

·空间数据索引：对于空间位置信息，采用空间索引技术（如 R - Tree），提高空间查询的效率。

**2.4多终端适配**

·响应式设计：在表现层采用响应式设计原则，根据不同终端（Web、Android、iOS、PC client）的屏幕尺寸和分辨率，自动调整界面布局和元素大小。

·跨平台框架：对于移动终端（Android、iOS），可以使用 React Native 或 Flutter 等跨平台框架进行开发，减少代码重复，提高开发效率。  
  
3.系统流程

**3.1系统初始化流程**

·启动系统，前端框架初始化，加载基础样式和脚本。

·从数据存储层读取 OpenStreetMap 地图数据，通过 OpenLayers 或 Leaflet 库进行 2D 地图渲染。

·调用 Mapbox GL JS 库，从 Mapbox Studio 获取 3D 地图配置信息，进行 3D 地图初始化渲染。

·从数据库中读取标记数据，根据时间轴的初始状态，展示相应时间点的标记信息。

**3.2标记创建流程**

·用户在界面上选择标记类型（2D、3D、4D）。

·对于 2D 标记，用户在 2D 地图上指定位置，输入标记属性信息，业务逻辑层将标记数据发送到数据访问层，数据访问层将数据写入数据库。

·对于 3D 标记，用户在 3D 地图上选择位置，设置 3D 相关参数（如高度、朝向等），输入属性信息，同样由业务逻辑层和数据访问层完成数据的存储。

·对于 4D 标记，用户除了设置空间位置和属性外，还需指定时间区间或时间点，系统记录每个时间点的状态，存储到数据库中。

**3.3数据查询流程**

用户在界面上输入查询条件，如关键词、空间范围、时间区间等。

业务逻辑层将查询请求发送到数据访问层，数据访问层根据查询条件从数据库中检索标记数据。

数据访问层将查询结果返回给业务逻辑层，业务逻辑层对数据进行整理和过滤，然后发送给表现层进行展示。  
  
4.技术难点

**4.1数据融合问题**

·OpenStreetMap 数据和 //Mapbox 数据格式不同，且标记数据的时空信息也需要与地图数据融合。

**4.2 3D 地图性能优化**

在加载大量 3D 模型和标记时，可能会出现性能瓶颈。

**4.3 时空数据查询效率**

随着标记数据量的增加，时空数据的查询效率可能会降低。

**四.4D GIS 标记系统开发计划**

|  |  |  |
| --- | --- | --- |
| 阶段 | 功能 | 交付物 |
| 需求分析与设计 | 完成需求分析 完成系统设计 | 需求规格说明书 系统设计文档 |
| 核心功能开发 | 基础地图显示 标记点管理  过滤功能  用户界面完善 | 可运行的基础版本  可使用的标记系统 多维度过滤 |
| 高级功能开发 | 地图集成与完善 数据功能 性能优化 | 3D可视功能  本地存储功能  响应式优化 |
| 测试与优化 | 功能测试  文档完善 | 测试报告  完整项目文档 |
| 项目收尾 | 演示准备  项目总结 | 项目演示文档  项目总结报告 |

第一阶段：需求分析与设计  
1.需求分析  
 任务清单：功能需求说明  
 交付物：功能需求规格说明书

2.系统设计  
 任务清单：系统设计  
 交付物：系统设计文档与开发计划  
  
第二阶段：核心功能开发  
1.基础框架搭建  
 任务清单：项目初始化与配置，React + TypeScript环境搭建，基础组件结构搭建，状态管理架构实现

2.地图基础功能

任务清单：地图继承，基础地图显示，缩放和拖拽功能，坐标系处理  
3.标记点管理系统  
 任务清单：标记点数据结构设计，点击地图添加标记点，标记点编辑与删除功能  
4.过滤系统  
 任务清单：时间维度，高度维度过滤算法，多维度组合过滤  
5.用户界面开发  
 任务清单：界面设计，标记点弹窗设计  
  
第三阶段：高级功能开发  
1. //3D地图集成  
 任务清单：Mapbox JS集成，3D建筑物，地形渲染，2D/3D切换功能  
2.数据持久化

任务清单：本地存储集成，数据导入导出功能，离线数据存储  
3.性能优化  
 任务清单：地图渲染性能优化，大量标记点渲染优化

第四阶段：测试与优化  
  
第五阶段：项目收尾