



# 互联网与移动 GIS 开发课程实习报告

## 武汉周边生活服务设施 POI-Map

小组成员：

李安南（111211） 肖江伟（111211）

黄诚 （111211） 但佳豪（111212）

任课教师： 郭明强

中国地质大学（武汉）计算机学院软件工程系

2024 年 4 月

# 目录

一、 系统背景.....	1
二、 需求分析.....	2
三、 开发环境介绍及配置.....	6
3.1 后端开发环境配置.....	7
3.2 前端开发环境配置.....	7
四、 系统设计.....	8
4.1 功能模块划分.....	8
4.2 功能流程设计.....	9
五、 数据库设计.....	11
六、 后端设计与实现.....	14
6.1 分层架构.....	14
6.2 接口列表.....	26
七、 前端设计与实现.....	27
7.1 界面设计.....	27
7.2 前后端交互.....	32
八、 部署到云服务器.....	34
九、 功能展示.....	42

# 一、系统背景

近年来，中国经济持续增长，城市居民的生活水平和消费能力显著提升。在这股浪潮下，人们对生活服务的需求日益丰富和多样化。无论是寻找美食、物流服务还是购物商场，消费者对于高品质、多样性和便利性的要求不断提高。创新业态的涌现、线上外卖的普及以及数字化转型的潮流，共同塑造了现代生活服务业的格局。为满足现代消费者的多样化需求，行业需要更先进的技术手段来提升服务和用户体验。

与此同时，WebGIS（Web 地理信息系统）技术也在蓬勃发展，它将 Web 技术和地理信息系统相结合，为各种场景提供了地理信息数据的可视化展示和深入分析。通过 WebGIS，用户可以利用普通浏览器轻松访问各种地理信息数据，方便快捷地进行空间数据的展示、分析和共享。WebGIS 在生活服务、物流和购物商场等领域中发挥着关键作用，为从业者和消费者提供了有力的支持。

在此背景下，我们开发了一款产品，旨在利用 MySQL 数据库和阿里云服务进行云端数据存储，使用 Spring Boot 框架构建后端服务，并以 Vue.js 作为前端框架，结合 OpenLayers 地图技术实现地图展示和交互功能。该产品的核心功能涵盖各类生活服务、物流和购物商场的信息查询、添加、编辑和删除，以及地图展示、定位和数据分析。通过这个架构，我们提供了高效的数据信息存储、稳定的后端服务和友好的前端体验。

## 应用场景：

我们的产品适用于多种场景，覆盖了生活服务的方方面面，为用户提供全面的地理信息服务：

### 1. 旅游规划

我们的产品为游客提供周边生活服务设施、购物商场和美食的信息。无论是寻找餐馆、咖啡店，还是探索购物商场和特色市场，系统都能满足游客的需求，帮助他们制定更丰富的旅行计划。用户可以在系统中找到中餐、西餐、日料、冷饮、咖啡店等各种餐饮信息，并了解花卉市场、建材市场、手机销售店等购物商场的信息。

### 2. 生活服务探索

用户可以方便地查找周边的生活服务设施，包括餐饮、超市、药店和娱乐设施，快速获取生活所需的信息。通过系统，用户可以找到各类便利商店、科教文化设施、农副产品市场和五金市场等信息。在需要帮助时，用户还能查找律师事务所和其他服务机构的具体位置。

### 3. 物流和购物行业分析

企业可以利用产品进行市场分析和用户行为研究，了解行业发展趋势和消费者偏好，以优化运营策略和制定更有效的营销方案。系统提供物流速递和配送服务等信息，帮助物流企业优化路线和服务。企业也可以通过分析数据，了解不同类型店铺和服务的分布情况，从而制定更精准的营销策略。

### 4. 商业服务

系统提供酒店、酒楼、烟酒专卖店和手机营业厅等商业信息，方便用户快速找到所需服务。对于计划举办活动或会议的用户，可以通过系统了解当地酒店和酒楼的具体信息，为活动策划提供数据支持。

### 5. 娱乐休闲

系统还涵盖了娱乐和休闲方面的信息，如电影、KTV、健身中心等设施，用户可以根据自己的需求查找附近的休闲娱乐场所。通过系统，用户能够轻松获取这些设施的详细信息，包括位置、服务类型和营业时间。

## 系统功能及意义：

随着城市化进程的加速，居民和游客对能够快速找到周边生活设施的需求日益迫切。科技的发展推动了信息获取方式的转型，人们希望通过智能设备即时获取相关信息。因此，开

发一个能够实时更新和管理这些信息的地理信息系统（GIS）至关重要。它不仅满足个人用户的需求，还为城市规划者和商业运营者提供实时、准确的数据支持。城市规划者可以利用系统数据进行更合理的资源配置和城市规划，而商业运营者可以根据用户偏好和活动热点调整营销策略和服务布局。

#### 技术架构及数据流：

系统基于先进的 Web 技术和地理信息系统（GIS），通过用户友好的交互界面，让用户能够以直观的方式获取信息。系统支持多种数据源，包括高德地图等商业 API 和开放数据集，以确保信息的全面性和多样性。通过高德 API 抓取最新数据，并使用数据清洗与验证确保数据的实时性和准确性。同时，系统还提供数据增删改查的管理功能，允许用户手动维护和修正数据，以应对市场和城市环境的动态变化。

#### 技术选项：

前端技术：采用 Vue.js 框架构建前端，利用其单页面应用（SPA）特性，提供动态的用户交互体验。通过 Element UI 组件库，快速构建美观且功能丰富的界面。

后端技术：后端选用 Spring Boot 框架，以其快速开发和部署能力，并提供强大的 REST API 支持和数据库交互能力。它的高性能与灵活性为系统运行提供保障。

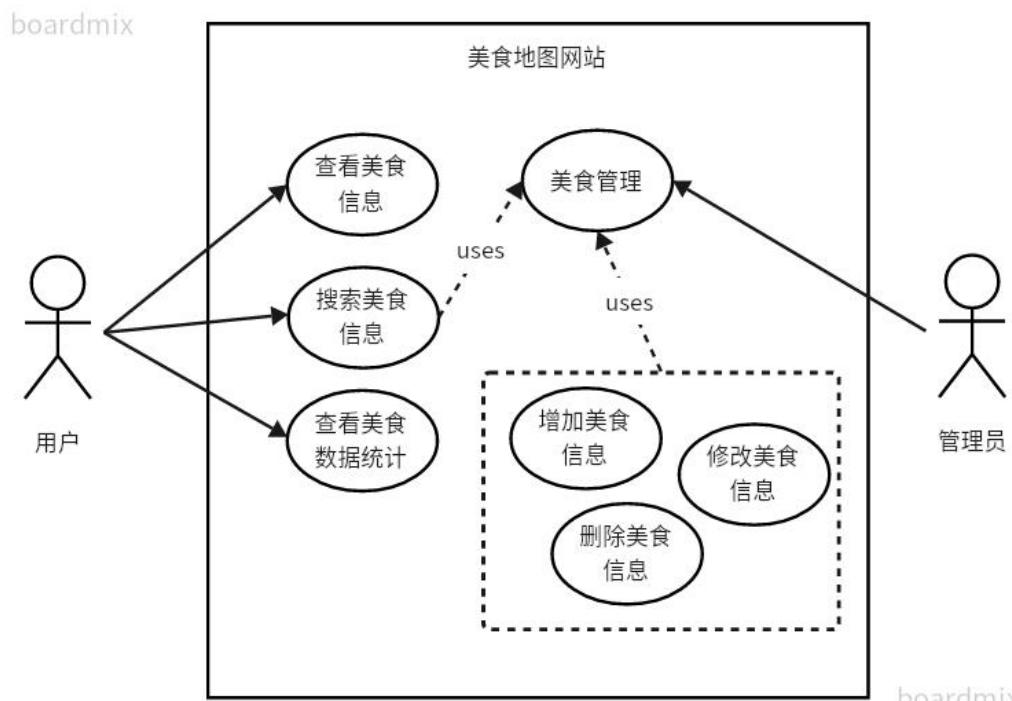
数据库选择：使用 MySQL 数据库存储用户信息、兴趣点数据等。MySQL 的稳定性、可扩展性和社区支持使其成为理想选择。

#### 系统架构和数据流：

系统采用前后端分离的架构，前端通过 RESTful API 与后端交互，后端处理请求并与 MySQL 数据库交互，然后将响应返回给前端。这种架构不仅提高了开发效率，还增强了系统的可维护性和扩展性。

## 二、需求分析

网站的使用者主要分为普通用户和网站管理员两类。二者在使用网站的过程中存在显著的需求差异：用户主要使用网站来查询和获取信息，而管理员的主要任务是管理和更新数据。以下是两类参与者的详细需求分析：



参与者	参与者说明
用户	网站的主要服务对象，能够查看美食地点的基本信息，并且能够根据需要通过地区与分类查询，还能查看不同地区的数据分布。
网站管理人员	网站的管理人员，能够对美食地图上的数据进行增删改查，能够及时对网站上的数据进行更新。

## 2.1 用户需求

系统的主要用户是美食和生活服务的爱好者，他们的需求主要包括查询和发现美食信息、获取服务和设施信息、直观的地图展示和定位、友好的界面和交互体验以及数据分析。

### 1. 查询和发现信息

用户场景：用户希望找到各种美食和生活服务信息，包括特色菜品、餐厅、酒店、购物中心、物流设施等。

用户行为：用户会利用系统内置的搜索功能，或浏览不同类别的美食和生活服务信息。

用户体验目标：提供快速准确的搜索结果，展示丰富的信息以满足用户的不同需求。

初步解决方案：设计一个易于使用的搜索界面，确保搜索结果准确且丰富，同时提供过滤和排序选项，帮助用户快速找到所需信息。

### 2. 地图展示和定位

用户场景：用户希望能够在地图上查看店铺或服务设施的位置，并导航到目的地。

用户行为：用户会在地图界面上搜索或浏览附近的服务设施，选择目标并导航。

用户体验目标：提供直观、易用的地图界面，让用户快速找到店铺或设施的位置，并提供方便的导航功能。

初步解决方案：整合地图服务，提供交互式地图界面，标注店铺位置，并提供导航按钮，方便用户轻松找到目的地。

### 3. 友好的界面和交互体验

用户场景：用户希望产品提供简洁、操作便捷的界面和交互体验。

用户行为：用户会浏览信息、使用搜索和过滤功能、进行导航等操作。

用户体验目标：提供直观、流畅的界面和交互体验，让用户能够轻松完成各种操作。

初步解决方案：设计简洁清晰的界面，采用直观的布局和导航方式，减少操作步骤，提高用户满意度和使用效率。

### 4. 信息统计和分析

用户场景：餐厅经理、物流经理和美食博主希望分析趋势和消费偏好。

用户行为：输入查询条件，浏览统计数据，进行对比并导出报告。

用户体验目标：界面友好，数据直观，提供定制化分析，确保数据准确及时。

初步解决方案：设计易用的界面，提供多种数据可视化选项和强大的分析工具，优化数据处理流程。

## 2.2 管理员需求

系统管理员负责维护系统的正常运行，确保数据的准确性、安全性。管理员的需求包括增加、删除和修改数据。

### 1. 增加信息

应用场景：管理员需要向系统中添加新数据时。

应用行为：管理员登录后台管理系统，进入管理界面，点击“新增”按钮，填写信息并保存。

初步解决方案：设计直观的表单界面，包含所有必填和可选字段，并提供明确的保存按钮，方便管理员轻松添加新信息。

## 2. 删除信息

应用场景：管理员需要从系统中移除不需要的数据时。

应用行为：管理员登录后台管理系统，进入管理界面，找到需要删除的数据并点击“删除”按钮，确认操作。

初步解决方案：在数据列表中添加删除按钮，并在执行操作前要求确认，以避免误操作导致数据丢失。

## 3. 修改信息

应用场景：数据需要更新或修正时，管理员可以对其进行修改。

应用行为：管理员登录后台系统，找到需要修改的信息，点击相应项进入编辑界面，修改后保存。

初步解决方案：提供清晰的编辑界面，显示当前信息的所有字段，并允许管理员进行修改，保存修改后更新数据库中的信息。

## 2.3 功能需求

### 1. 地图展示功能

查看信息：用户可以在地图上查看各个地点的信息，包括经纬度、名称、地址和类型。

图层切换：地图需要支持多种图层显示，用户可以在不同图层间切换，以查看不同类型的地图数据，如卫星地图、道路地图。

### 2. POI 点管理功能

添加：用户可以在地图上添加新的兴趣点，并输入完整的信息，地图上会显示相应标记。

编辑：用户可以编辑已有兴趣点的信息，以及时更新数据。

删除：用户可以删除不需要的兴趣点，清理地图上的标记，使数据更加清晰。

### 3. 数据过滤和搜索功能

搜索：用户可以根据省、市、区、名称和类型等条件对地点数据进行搜索，从而快速找到感兴趣的地点信息。

过滤：过滤和搜索功能帮助用户在大量数据中快速定位到需要的信息，提高用户体验。

### 4. 数据可视化和统计功能

用户可以查看地点数据的统计信息，比如按省、市、区和类型统计数量，了解数据分布情况。

数据统计功能提供图表展示，用户可以通过图表了解数据的分布和各类型的占比。

## 2.4 性能需求

### 1. 加载速度

地图数据加载：地图数据应在用户进入页面后迅速加载，以提供即时的地图浏览体验。

系统界面加载：系统界面应快速加载，避免长时间等待或白屏。

### 2. 响应时间

用户操作：用户操作（如添加、编辑、删除兴趣点）应获得快速响应，避免明显的延迟。

数据过滤和搜索：数据过滤和搜索应在提交请求后快速完成，迅速呈现结果。

### 3. 资源利用率

服务器资源：系统应合理利用服务器资源，提高系统的吞吐量和处理能力。

**数据库优化：**优化数据库查询和索引设计，以降低数据库负载和响应时间。

#### 4. 缓存策略

**缓存技术：**使用适当的缓存技术（如 Redis、Memcached）缓存频繁访问的数据，减轻数据库负担，提高响应速度。

#### 5. 安全性和数据隐私

**数据安全：**保证用户数据的安全和隐私，防止未授权访问和数据泄露。

**加密与权限控制：**实现数据传输加密，采取身份验证和权限控制措施，确保数据安全。

#### 6. 系统稳定性

**容错能力：**系统应具备容错能力，能处理异常情况并保持正常运行。

**性能测试：**定期进行系统性能和负载测试，发现并解决潜在问题，确保系统稳定性。

#### 7. 扩展性和灵活性

**系统架构：**系统架构应具备扩展性，可根据业务需求灵活扩展和调整。

## 2.5 用户画像

属性	描述
姓名	李某
年龄	45岁
职业	餐厅经理
生活地点	武汉
兴趣爱好	阅读、美食
行为习惯	负责管理餐厅的日常运营，优化服务流程。 借助数据统计功能分析餐厅业务，制定营销策略。 关注消费者反馈，调整菜单和服务。 依赖地图展示功能来了解竞争对手的分布情况。

属性	描述
姓名	林某
年龄	28岁
职业	营销经理
生活地点	武汉
兴趣爱好	喜欢寻找特色美食、与朋友聚餐、旅游和记录生活
行为习惯	经常使用美食类应用寻找新的餐厅。 喜欢通过地图功能查看附近的美食地点。 注重界面体验，喜欢简单、易用的交互界面。 关注餐厅评价和推荐，依赖评分和评论进行选择。

属性	描述
姓名	王某
年龄	32岁
职业	物流公司经理
生活地点	武汉
兴趣爱好	喜欢运动、旅行、户外活动

行为习惯	使用物流系统管理公司业务，分析客户配送需求。 通过数据统计和分析了解物流网络的分布和客户偏好。 需要准确且实时的地图信息来规划最佳配送路线。 喜欢高效且直观的数据展示界面。
------	---

属性	描述
姓名	张某
年龄	40 岁
职业	网站运营管理员
生活地点	武汉
兴趣爱好	喜欢健身、阅读技术类书籍
行为习惯	负责网站的数据管理，维护和更新网站内容。 对后台管理系统的可操作性和安全性要求高。 需要定期导出数据报告，了解网站流量和数据变化。 希望系统稳定，提供清晰的反馈机制。

属性	描述
姓名	赵某
年龄	20 岁
职业	美食博主
生活地点	武汉
兴趣爱好	摄影、旅行、写作
行为习惯	定期访问餐厅，拍摄美食照片并写评测。 使用地图应用寻找特色餐厅和热门美食地点。 借助数据分析功能了解美食消费趋势。 注重平台数据的准确性，以提高内容的可信度。

### 三、开发环境介绍与配置

开发环境信息	详情
开发的硬件环境	前端: CPU: i5-11320H; RAM: 16G 后端: CPU: i7-11390H; RAM: 16G
运行的硬件环境	前端: CPU: i5-11320H; RAM: 16G 后端: CPU: i7-11390H; RAM: 16G
测试的硬件环境	CPU: i5-10300H; RAM: 16G
云服务器环境	Alibaba Cloud Linux 3.2104 LTS 64 位
开发的操作系统	OS: Windows 11
软件开发环境/工具	IntelliJ IDEA 2022.3.3、Edge、ApiPost7

运行平台/系统	Windows 10、Windows 11
软件运行支撑环境/软件	MySQL8.0、阿里云服务器
编程语言	Java、JavaScript、Vue2
编程环境	node 16.19.0、npm 7.24.2、vue 2.6.14、eslint 7.32.0、openlayers 9.1.0、Java JDK 18

### 3.1 后端开发环境配置

**开发工具:** 使用 IntelliJ IDEA 进行后端开发。IntelliJ IDEA 是一款专业的 Java IDE，具备卓越的代码分析和重构能力。其智能的代码补全功能通过内置的语义分析，实现语法和逻辑错误检测。它还拥有强大的调试工具、版本控制集成、Maven 和 Gradle 项目管理等特性，可全面满足企业级应用开发需求。

**框架:** 采用 Spring Boot 构建后端应用。作为 Spring 框架的扩展，Spring Boot 通过自动配置和嵌入式服务器支持实现了零配置启动，简化了基于 Spring 应用的开发流程。Spring Boot 的强大生态系统涵盖了从 Web 应用、微服务架构到企业级集成的各个方面，提供了安全性、数据库交互、REST API 等诸多组件。它遵循“约定优于配置”的理念，将繁琐的配置抽象化，开发者可以直接使用预设的功能模块，高效完成业务开发。

**数据库:** 选用 MySQL 作为后端数据库。MySQL 以其快速查询能力和高可用性成为互联网应用的首选数据库之一。它支持多线程处理和高级索引优化，能够在高并发场景中保持出色的性能。借助其丰富的功能特性，如分区、复制、分布式架构等，MySQL 能够满足大型企业级应用对数据存储的需求。此外，它还提供灵活的权限管理和事务支持，为开发人员提供可靠的数据保护和一致性保障。

**数据库管理工具:** 借助 Navicat 进行 MySQL 数据库的可视化管理。Navicat 为开发人员提供直观的界面，通过图形化操作和可视化工具集，大幅降低数据库管理的复杂度。它支持从简单的表设计到复杂的查询优化，并提供数据迁移、备份恢复等功能。Navicat 的 SQL 代码自动补全和模板功能有助于开发人员快速编写复杂查询语句，提高开发效率。

**后端测试工具:** 使用 ApiPost 作为测试工具。ApiPost 提供了直观的界面和强大的测试功能，允许开发人员轻松编写、执行和管理 API 测试。它可以模拟 HTTP 请求，并为接口开发提供实时反馈，有助于快速发现和解决问题。

### 3.2 前端开发环境配置

**开发工具:** 前端开发同样使用 IntelliJ IDEA。作为全功能的集成开发环境（IDE），IntelliJ IDEA 提供了强大的代码编辑和调试能力。它支持 HTML、CSS、JavaScript 以及 Vue.js 等前端技术的代码补全和语法高亮。通过对 Vue.js 等前端框架的深度集成，IDEA 提供了组件自动补全、模板语法高亮、Lint 检查和格式化工具，显著提升前端开发效率。同时，它还具备强大的插件生态系统，支持代码版本控制、调试和部署等多项功能，满足现代前端开发的全方位需求。

**框架:** 采用 Vue.js 框架进行前端开发，具体使用 Vue2 版本。Vue2 具备灵活的组件化设计和轻量级的框架特性，支持单页面应用（SPA）的开发模式。通过其 Virtual DOM 和双向数据绑定机制，可以实现高性能的动态数据渲染。Vue2 的组件复用能力允许开发者以模块化方式构建复杂的用户界面。此外，Vue Router 和 Vuex 等插件进一步增强了 Vue.js 的应用能力，实现路由管理和全局状态管理，适用于构建大型交互式 Web 应用。

**UI 组件库：**使用 Element UI 作为 UI 组件库。Element UI 是基于 Vue.js 的一款现代化组件库，提供了丰富的 UI 组件，如按钮、表格、表单、对话框等。它的设计风格简洁、时尚且易用，使得开发者能够快速构建美观且功能丰富的用户界面。其组件具备高度的可定制性，能够满足各类应用场景的需求。此外，Element UI 还拥有详细的文档和活跃的社区支持，方便开发者快速上手和解决问题。

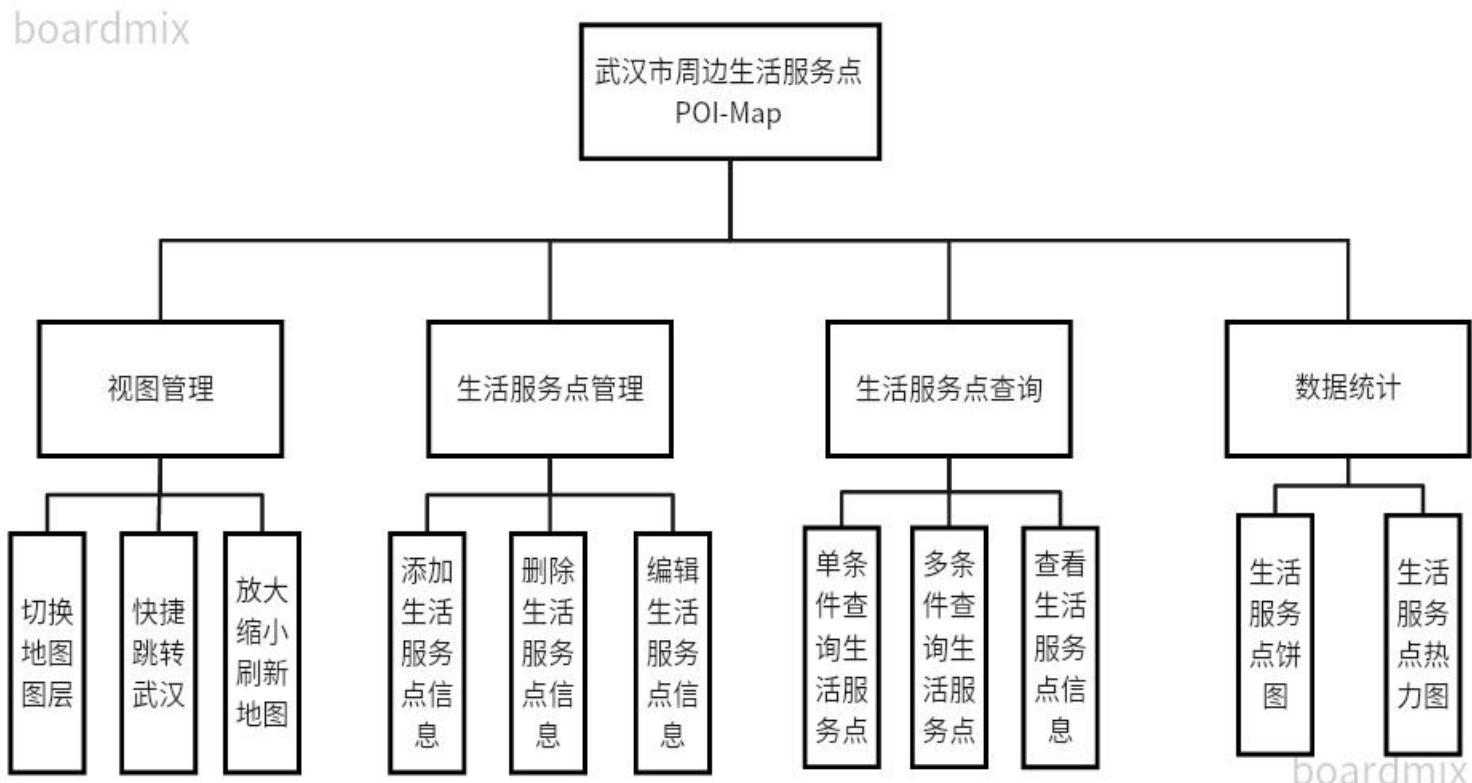
**地图组件：**使用 OpenLayers 作为地图组件，为应用提供交互式地图服务。OpenLayers 是一款开源的 JavaScript 地图库，支持多种地图服务和矢量数据格式。它具有高度的灵活性和扩展性，能够通过插件化的方式加载各类地图数据，并提供丰富的地图交互功能。通过 OpenLayers，用户可以在地图上实现兴趣点的标注、路径规划、热力图等功能，以满足复杂的地理信息展示和交互需求。

**前端测试工具：**使用 Edge 浏览器作为前端测试工具。Edge 浏览器的开发者工具为前端开发者提供了完整的调试环境，支持实时编辑 HTML、CSS 和 JavaScript，同时还具备性能分析、网络请求监控等功能，有助于优化前端应用性能。

## 四、系统设计

### 4.1 功能模块划分

根据需求分析，系统划分为以下四个主要模块：视图管理、生活购物点管理、生活购物点查询和数据统计。



#### 1. 视图管理

视图管理模块主要负责控制地图界面，为用户提供友好的地图交互功能，并优化用户体验。其关键功能包括：

**切换地图图层:** 用户可以在不同地图图层间进行切换,如高德地图和高德卫星图。通过在不同的图层之间切换,用户可以查看不同类型的地理信息,例如基础地图、卫星地图等。此功能确保用户可以根据自己的需求选择合适的地图视图,满足多样化的查看需求。

**快捷跳转武汉:** 提供一键快捷定位的功能,将地图中心直接定位在武汉区域,并自动调整缩放级别。该功能让用户无需手动操作缩放和移动地图,就可以快速查看武汉区域的地图信息,提升地图操作的便利性。

**放大缩小刷新地图:** 允许用户通过拖拽、缩放等操作来控制地图的显示区域,还可以通过刷新功能将地图恢复至初始位置和大小。这个功能使用户能够灵活地查看不同区域的地图详情,在探索城市区域时拥有更好的导航体验。

## 2. 生活服务点管理

生活服务地点管理模块主要用于对生活服务地点数据进行管理,确保前端数据和后端数据库保持一致性和完整性。主要功能包括:

**添加生活购物点信息:** 通过提供易用的表单或界面,方便管理员添加新的生活服务地点信息。管理员可以输入生活服务地点的名称、地址、类型、经纬度等详细信息,从而确保数据的完整性和准确性。

**删除生活购物点信息:** 允许管理员删除指定的生活服务地点信息,以保持数据库数据的准确性和清洁性。通过界面或表单,管理员可以精确定位到需要删除的生活服务地点,并通过确认操作安全地删除数据。

**编辑生活购物点信息:** 提供界面和工具,让管理员可以更新购物点信息。该功能允许管理员根据实际情况更新购物点的详细信息,以确保数据的时效性和准确性。

## 3. 生活服务点查询

生活服务地点查询模块通过整合地图服务和数据库数据,为用户提供多样化的查询功能。主要功能包括:

**单条查询生活服务地点:** 根据特定条件,查询并展示单个购物点的信息。用户可以通过输入特定的查询条件,精准定位到某个购物点,以获取其详细信息。

**多条查询生活服务地点:** 根据用户自定义的查询条件,精确查询并展示符合条件的购物点列表。此功能通过让用户灵活设置查询条件,可以检索到符合要求的多个购物点信息,满足用户的多样化需求。

**查看生活服务地点信息:** 通过与地图相结合的方式,展示生活服务地点的详细信息,包括名称、地址、类型、经纬度等。用户可以在地图上查看生活服务地点的位置和详细数据,以便更加直观地获得生活服务地点信息。

## 4. 数据统计

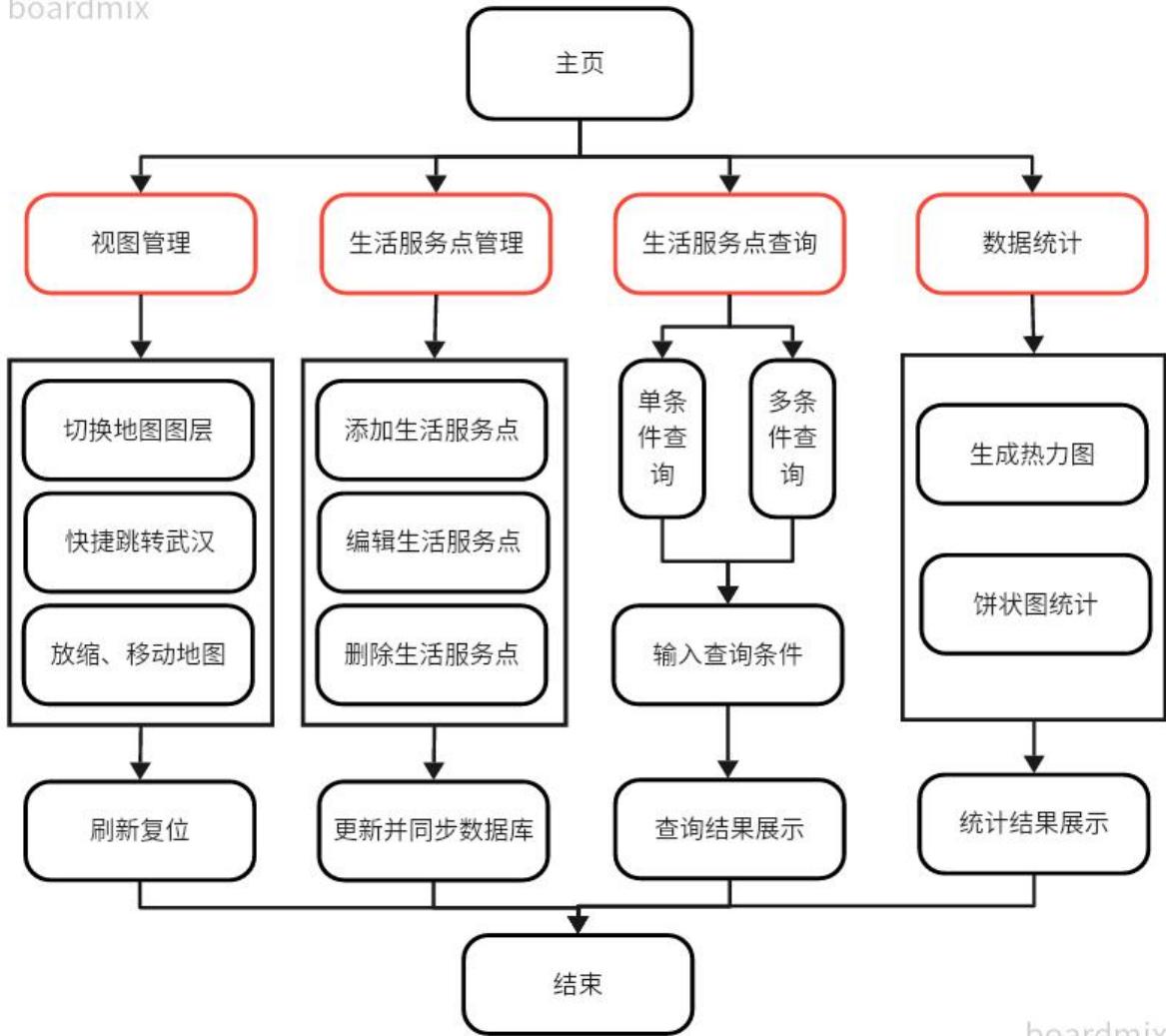
数据统计模块通过对地图上的购物点数据进行分析,生成可视化的统计图表和热力图,提供更加直观的显示。主要功能包括:

**生活购物点饼图:** 按不同分类统计当前地图上购物点的数量和占比。饼图提供了一个直观的方式来查看不同类型的购物点在数量上的占比,有助于用户对数据进行分类分析。

**生活购物点热力图:** 通过展示购物点的密度分布,并允许用户在热力图和普通地图之间切换,以满足不同的数据展示需求。通过热力图,用户可以快速识别购物点的高密度区域,辅助城市规划者和商业运营者进行布局规划。

## 4.2 功能流程设计

系统的功能流程设计如下:



## 1. 地图主页

进入系统后，系统首先加载主页，用户可以在主页中进行地图操作。主页功能包含：

**地图视图管理：**通过缩放、拖拽、刷新操作来控制地图视图。用户可以通过点击刷新按钮将地图重置至初始位置和大小。当组件挂载时，系统会创建一个 OpenLayers 地图实例，初始视图中心设置为北京的坐标。地图附带图层、比例尺等控件，用户可以缩放、拖拽和查看地图。

**图层切换：**用户可以在不同地图图层间切换，如高德地图和高德卫星图。通过切换按钮，前端发送请求到后端，以获取不同的地图图层数据并在视图中切换显示。

**快捷跳转武汉：**通过点击快捷跳转按钮，前端会发送请求到后端获取武汉的地理坐标信息，并使用动画效果将地图中心切换到武汉区域。

## 2. 生活服务点管理

生活服务地点管理模块提供对生活服务地点数据的增删改功能。用户可以在此模块中：

**新增生活服务地点：**用户左键选中新增按钮，页面进入新增模式。用户点击地图任意点，系统会获取该点的经纬度，并将其填入新生活服务地点表单。用户在添加页面中填写生活服务地点的详细信息，包括名称、类型、地址、经纬度等，并点击确认按钮将数据通过 POST 请求发送到后端，后端将数据保存至数据库并返回操作结果。

**编辑生活服务地点:** 用户左键选中编辑按钮，页面进入编辑模式。用户点击地图上的生活服务地点标记后，系统会自动填充生活服务地点的现有信息到编辑表单。编辑信息后，用户通过点击确认按钮将更新后的信息通过 API 请求发送到后端，后端根据生活服务地点的唯一标识更新数据库中的数据。

**删除生活服务地点:** 用户左键选中删除按钮，页面进入删除模式。用户点击地图上的购物点标记，将通过 DELETE 请求通知后端删除指定生活服务地点。后端从数据库中删除生活服务地点，并返回操作结果。系统会自动同步数据库，确保前端与后端数据的一致性。

### 3. 生活服务点查询

生活服务设施查询模块通过整合地图服务和数据库数据，为用户提供多样化的查询功能：

**单条件查询:** 用户在页面下方搜索框输入精确的查询条件后点击确认按钮，前端将查询请求发送到后端，后端根据条件查询数据库，将符合条件的生活服务地点信息返回给前端并在地图上高亮显示。

**多条件查询:** 用户在页面下方搜索框设置多个条件进行组合查询后点击确认按钮，前端将查询条件通过 POST 请求发送给后端，后端返回符合条件的生活服务地点列表，前端会将结果展示在地图上。用户可以通过点击查询结果查看生活服务地点的详细信息，地图也会跳转并高亮显示该点。

### 4. 数据统计

数据统计模块通过对购物点数据进行可视化分析，主要提供：

**热力图:** 根据地图上当前已有的购物点数据密度生成热力图。前端将地图上当前的可见区域发送到后端，后端根据该区域的购物点数据生成热力图，并将结果返回给前端进行展示。

**饼状图:** 根据购物点的类型和数量生成饼状图。前端发送请求到后端以获取当前区域的购物点分类信息，后端将统计好的数据返回前端，前端利用该数据生成饼状图展示给用户。

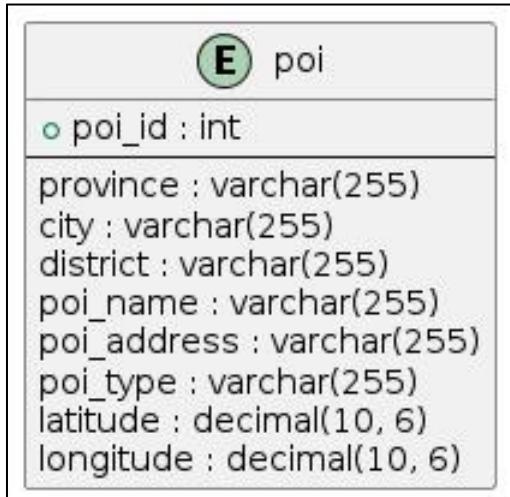
## 五、数据库设计

数据库采用的是 MySQL 关系型数据库，选择 MySQL 的优势包括：

1. 性能优越：MySQL 支持大量并发连接和高效查询，能够处理大规模数据和高并发请求，满足互联网应用的高性能需求。
2. 易于使用：MySQL 支持标准的 SQL 语法，开发人员可以轻松掌握并使用。
3. 可扩展性：通过主从复制、分区和集群等技术，MySQL 可以扩展数据库的容量和吞吐量，支持海量数据和高并发应用。
3. 强大的功能集：MySQL 提供事务支持、触发器、存储过程、索引和优化器等功能，确保数据操作的完整性、稳定性和高效性。

### 1. 表设计：

poi 表用于存储每个兴趣点的详细信息，其结构如下：



字段名	数据类型	描述
poi_id	int	主键，自动递增的整数
province	varchar(255)	兴趣点所在的省份名称，可选字段
city	varchar(255)	兴趣点所在的城市名称，可选字段
district	varchar(255)	兴趣点所在的区域名称，必填字段
poi_name	varchar(255)	兴趣点的名称，必填字段
poi_address	varchar(255)	兴趣点的具体地址，必填字段
poi_type	varchar(255)	兴趣点的类型，必填字段
latitude	decimal(10, 6)	兴趣点的纬度，必填字段

#### 详细介绍：

**poi\_id:** 表的主键，类型为自动递增的整数，每个兴趣点都有一个唯一的 poi\_id。

**province:** 用于存储兴趣点所在的省份名称信息，可选字段，允许为 NULL 值。数据类型为 varchar(255)，支持最长 255 个字符。

**city:** 用于存储兴趣点所在的城市名称信息，可选字段，允许为 NULL 值。数据类型为 varchar(255)，支持最长 255 个字符。

**district:** 存储兴趣点所在的区域名称信息，必填字段，允许为 NULL 值。数据类型为 varchar(255)，支持最长 255 个字符。

**poi\_name:** 存储兴趣点名称，必填字段。数据类型为 varchar(255)，支持最长 255 个字符。

**poi\_address:** 存储兴趣点具体的地址信息，必填字段。数据类型为 varchar(255)，支持最长 255 个字符。

`poi_type`: 存储兴趣点类型，必填字段。数据类型为 `varchar(255)`，支持最长 255 个字符。  
`latitude`: 存储兴趣点的纬度，必填字段。数据类型为 `decimal(10, 6)`，能精确表示纬度信息。  
`longitude`: 存储兴趣点的经度，必填字段。数据类型为 `decimal(10, 6)`，能精确表示纬度信息。

MySQL 语句：

```
CREATE TABLE `poi` (
  `poi_id` int NOT NULL AUTO_INCREMENT,
  `province` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci
NULL DEFAULT NULL,
  `city` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NULL
DEFAULT NULL,
  `district` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci
NOT NULL,
  `poi_name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci
NOT NULL,
  `poi_address` varchar(255) CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci NOT NULL,
  `poi_type` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci
NOT NULL,
  `latitude` decimal(10, 6) NOT NULL,
  `longitude` decimal(10, 6) NOT NULL,
  PRIMARY KEY (`poi_id`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_unicode_ci
ROW_FORMAT = Dynamic;
```

## 2. 表分析：

### 表的目的和用途：

`poi` 表是数据库结构的重要组成部分，专门用于存储兴趣点（Point of Interest）的详细信息，包括名称、地址、类型及其地理位置（经纬度）等。在应用程序中，这个表的作用至关重要，因为它支持以下功能：

1. 地图显示：系统能够利用表中存储的经纬度信息，将兴趣点在地图上精确标记，提供给用户直观的地理位置展示。
2. 数据查询：通过名称、地址、类型和地理位置等信息的检索，用户可以在数据库中快速查找到特定的兴趣点，满足各种不同场景的查询需求。
3. 信息浏览：利用 `poi` 表的详细信息，用户能够浏览兴趣点的相关信息，比如名称、位置、类型等，为用户提供丰富的参考信息。

### 字段选择：

1. `poi_id`: 这个字段作为主键，唯一标识每个兴趣点，是数据库中检索和操作数据的重要依据。它是自动递增的整数类型，确保每个兴趣点都有唯一标识符。
2. `province`、`city`、`district`: 这些字段用于存储兴趣点的地理位置信息，涵盖省、市、区等各级行政区域信息。通过这些字段，用户可以根据区域条件来筛选兴趣点，进行区域性的分析和展示。

3. **poi\_name**、**poi\_address**、**poi\_type**: 这些字段存储兴趣点的基本信息，包括名称、地址和类型。名称提供了兴趣点的识别信息；地址用于定位兴趣点的详细位置；类型描述了兴趣点的属性，方便用户分类查询。

4. **latitude**、**longitude**: 这两个字段分别存储兴趣点的纬度和经度信息，提供兴趣点的精确地理位置。通过这些字段，系统能够在地图上精确显示兴趣点的位置，支持地图展示、导航等功能。

#### 数据类型选择:

1. 字符串字段: **province**、**city**、**district**、**poi\_name**、**poi\_address**、**poi\_type** 使用 **varchar(255)** 类型，可以灵活存储长度不一的字符串信息，确保数据存储的灵活性和完整性。
2. 数字字段: **poi\_id** 使用自动递增的 **int** 类型，保证每个兴趣点都有唯一的标识符。**latitude** 和 **longitude** 使用 **decimal(10, 6)** 类型，确保经纬度信息的精确度的同时满足地理位置的精确展示需求。

#### 索引使用:

在 **poi\_id** 字段上使用 **B-tree** 索引，作为主键索引能够提高数据库的查询效率。**B-tree** 索引的结构适用于各种数据范围查询，可以快速检索大量数据集中的特定兴趣点信息。

#### 数据库设计的优势:

1. 数据完整性: **district**、**poi\_name**、**poi\_address**、**poi\_type**、**latitude** 和 **longitude** 等字段被设计为必填字段，确保每个兴趣点的基本信息完整，为系统提供稳定的数据支撑。
2. 数据一致性: **poi\_id** 作为自动递增的主键，确保每个兴趣点都有唯一标识符，避免数据重复，确保数据的准确性和一致性。
3. 数据精确性: **latitude** 和 **longitude** 使用 **decimal(10, 6)** 类型，确保地理位置的经纬度信息精确，满足地理展示、导航等应用的需求。
4. 数据可读性: 所有字段名设计得直观清晰，涵盖了兴趣点的各种关键属性，使数据易于理解和使用，方便开发人员和用户进行操作和管理。

## 六、后端设计与实现

### 6.1 分层架构

本项目选择 **Spring Boot** 框架，作为后端项目的基础架构。

#### 选择原因:

快速构建项目: **Spring Boot** 能够快速创建基于 **Spring** 的应用程序，大大提高开发效率。

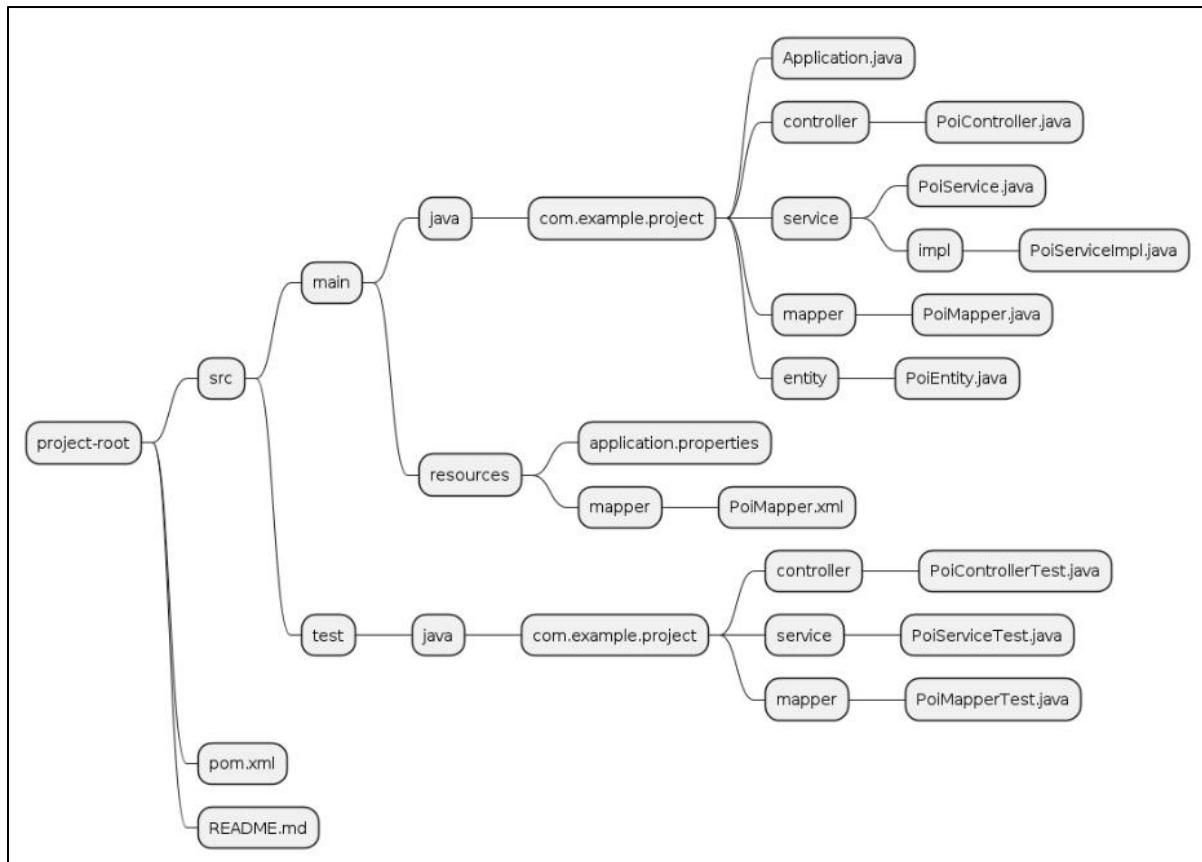
自动配置: 提供自动配置和约定优于配置的方式，使开发人员可以专注于业务逻辑，而不必过多关注底层技术细节。

独立运行: **Spring Boot** 应用程序可以独立运行，无需外部依赖 **Servlet** 容器。

运行时监控: 提供运行时的应用监控，帮助开发者更好地理解和优化他们的应用程序。

#### pom.xml 配置

项目使用了以下关键依赖:



### 1. spring-boot-starter-data-jpa

**用途：**集成 Spring Data JPA，简化与数据库的交互。

**详细描述：**Spring Data JPA 是一个用于持久化数据的框架，通过简化数据访问代码，实现对 Java 持久层 API (JPA) 的支持。提供了开箱即用的 CRUD 功能、与 Hibernate 的集成、自动生成 SQL 查询等功能。

### 2. spring-boot-starter-thymeleaf

**用途：**集成 Thymeleaf 模板引擎。

**详细描述：**Thymeleaf 是一个用于生成动态网页的服务器端 Java 模板引擎。通过与 Spring MVC 集成，可生成 HTML、XML、JavaScript 等不同格式的模板，提供动态数据渲染的功能，适用于单页应用或传统 Web 应用。

### 3. spring-boot-starter-web

**用途：**快速构建 Web 应用，支持 RESTful 服务。

**详细描述：**该依赖包含 Spring MVC 框架所需的所有组件，用于构建 HTTP 服务和 REST API。支持 Spring Boot 内置的 Tomcat 服务器，实现开发和部署的快速启动。

### 4. mysql-connector-j

**用途：**提供 MySQL 与 Java 应用程序之间的连接。

**详细描述：**这是 MySQL 官方提供的 JDBC 驱动程序，可用于与 MySQL 数据库交互。通过标准的 JDBC 接口，支持使用 Java 代码直接执行 SQL 查询和事务处理。

### 5. lombok

**用途：**简化 Java 类的代码编写。

**详细描述：**Lombok 是一个 Java 库，提供一组注解（如 @Getter、@Setter、@ToString）来简化代码。它通过注解生成常用方法如 getter、setter、构造函数、equals、hashCode 等，从而减少模板代码，增强代码可读性。

## 6. spring-boot-starter-test

用途：支持单元测试和集成测试。

详细描述：该依赖包含 JUnit、Mockito 和 Spring Test 等测试框架的工具和类库。它为应用程序提供全方位的测试支持，如模拟 Bean 环境、HTTP 请求和数据库操作。

## 7. xmlunit-core

用途：用于验证和比较 XML 数据。

详细描述：提供一套用于 XML 验证和比较的工具。支持比较 XML 文档的相似性、执行 XPath 查询和 XML 验证等操作，适用于需要处理复杂 XML 数据的应用程序。

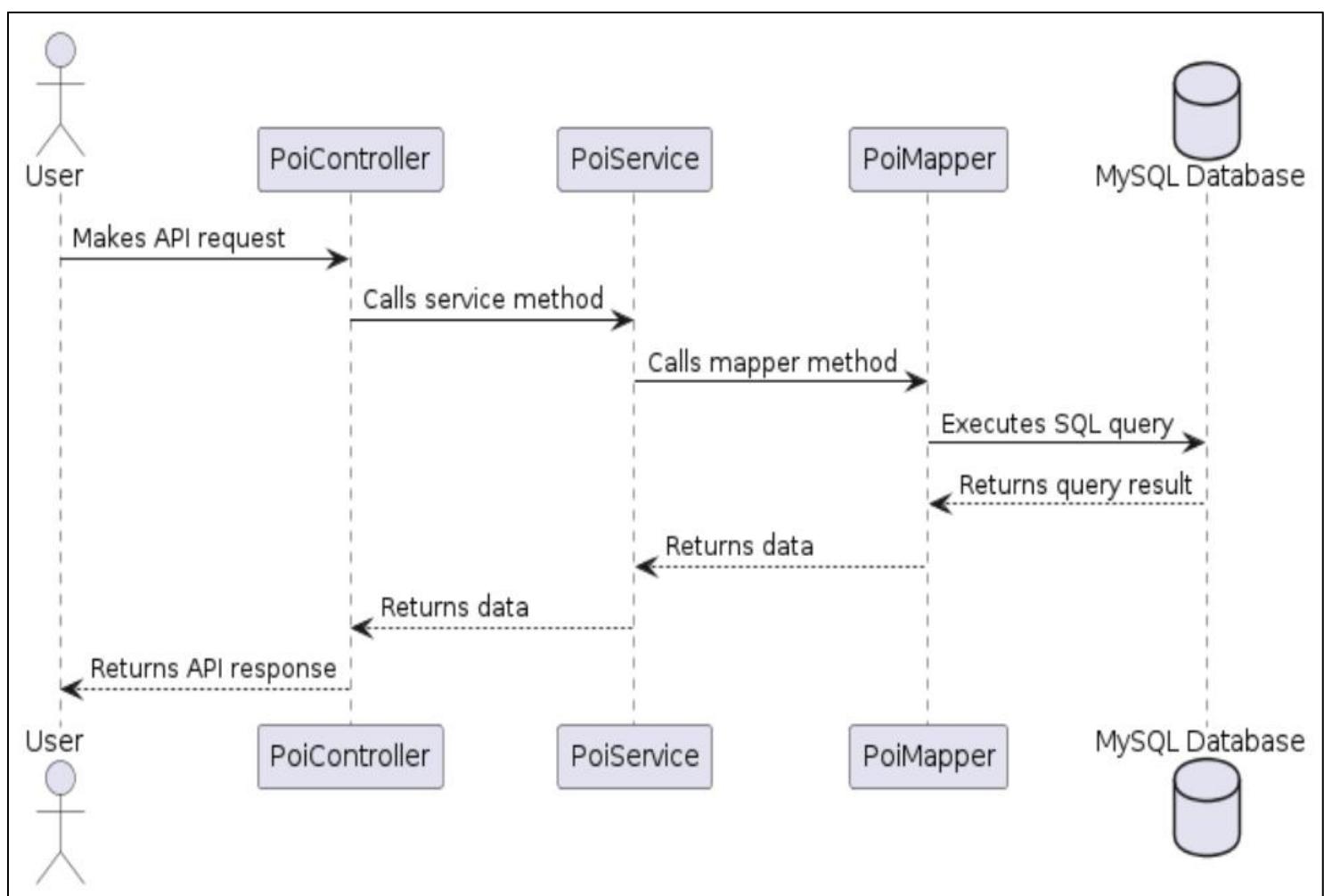
## 8. mybatis-spring-boot-starter

用途：集成 MyBatis 与 Spring Boot。

详细描述：该依赖简化了 MyBatis 与 Spring Boot 应用的集成。MyBatis 是一种持久层框架，支持 SQL 查询的直接执行和对象关系映射。通过与 Spring Boot 集成，提供自动配置、Mapper 接口、数据源配置等功能。

## 项目分层架构

项目分为 Entity、Mapper、Service、Controller 四层：



‘数据流图’

## 1. Entity 层:

定义与数据库表对应的实体类，与数据库进行直接关联和数据交互。

下面是 Entity 层的 poi 类中各个属性的定义，和数据库各字段一一对应：

```
1. @JsonProperty(value = "poi_id")
2.     Integer poi_id;
3.     @JsonProperty(value = "province")
4.     String province;
5.     @JsonProperty(value = "city")
6.     String city;
7.     @JsonProperty(value = "district")
8.     String district;
9.     @JsonProperty(value = "poi_name")
10.    String poi_name;
11.    @JsonProperty(value = "poi_address")
12.    String poi_address;
13.    @JsonProperty(value = "poi_type")
14.    String poi_type;
15.    @JsonProperty(value = "latitude")
16.    double latitude;
17.    @JsonProperty(value = "longitude")
18.    double longitude;
```

该 Java 类代表了数据库中的 poi 表。Entity 类的每个属性都对应了 poi 表中的一个字段，并使用了@JsonProperty 注解来指定 JSON 属性的名称。以下是对每个属性的详细介绍：

**poi\_id:** 这是一个 Integer 类型的属性，对应了 poi 表中的 poi\_id 字段。这个字段是表的主键，每个兴趣点都有一个唯一的 poi\_id。

**province:** 这是一个 String 类型的属性，对应了 poi 表中的 province 字段。这个字段用于存储兴趣点所在的省份名称。

**city:** 这是一个 String 类型的属性，对应了 poi 表中的 city 字段。这个字段用于存储兴趣点所在的城市名称。

**district:** 这是一个 String 类型的属性，对应了 poi 表中的 district 字段。这个字段用于存储兴趣点所在的区域名称。

**poi\_name:** 这是一个 String 类型的属性，对应了 poi 表中的 poi\_name 字段。这个字段用于存储兴趣点的名称。

**poi\_address:** 这是一个 String 类型的属性，对应了 poi 表中的 poi\_address 字段。这个字段用于存储兴趣点的具体地址。

**poi\_type:** 这是一个 String 类型的属性，对应了 poi 表中的 poi\_type 字段。这个字段用于存储兴趣点的类型。

**latitude:** 这是一个 double 类型的属性，对应了 poi 表中的 latitude 字段。这个字段用于存储兴趣点的纬度。

**longitude:** 这是一个 double 类型的属性，对应了 poi 表中的 longitude 字段。这个字段用于存储兴趣点的经度。

该层的实体类通过，公有的 get 和 set 方法将这各个属性读取和封装出来供上一层使用：

```

1.     public Integer getPoi_id() {
2.         return poi_id;
3.     }
4.     public void setPoi_id(Integer poi_id) {
5.         this.poi_id = poi_id;
6.     }

```

## 2. Mapper 层:

使用 MyBatis 直接操作数据库，并提供对数据库操作的方法。

Mapper 层在 MyBatis 框架中起着非常重要的作用，它使用了 MyBatis 的@Select、@Insert 等注解来使用指定 SQL 查询等语句来对数据库进行操作，主要负责与数据库进行交互。

Mapper 层中的每个方法都对应了一个特定的 SQL 语句，这些 SQL 语句用于查询、插入、更新或删除数据库中的数据。

在本项目中，Mapper 层的方法都被定义在一个接口中，这个接口被称为 Mapper 接口。每个 Mapper 接口的方法都使用了 MyBatis 的注解(如@Select、@Insert、@Update 和@Delete)来指定对应的 SQL 语句。

Mapper 层的主要有以下优点：

**模块化：**通过将数据库操作封装在 Mapper 层，可以使得代码更加模块化。这样，你可以在 Service 层中调用 Mapper 层的方法，而不需要直接编写 SQL 语句。

**易于维护：**如果需要修改数据库操作的逻辑，只需要修改 Mapper 层的代码。这样，可以更加轻松地维护和更新代码。

**灵活性：**你可以在 Mapper 接口中定义任何你需要的方法，每个方法都可以对应一个特定的 SQL 语句。这样，你可以根据你的实际需求来创建和使用 Mapper 接口。

**防止 SQL 注入：**通过使用#{ }来绑定参数，MyBatis 会自动处理参数的转义和格式化，从而防止 SQL 注入攻击。

### 单条件查询操作：

```

1.     //查询所有POI
2.     @Select("SELECT * FROM poi")
3.     List<PoiEntity> showAllPoi();
4.     //根据poi_id查询
5.     @Select("SELECT * FROM poi WHERE poi_id = #{poi_id}")
6.     List<PoiEntity> showPoiByPoi_Id(@Param("poi_id") Integer poi_
id);
7.     //根据距离查询
8.     @Select("SELECT * FROM poi WHERE (6371 * acos(cos(radians(#{l
atitude})) * cos(radians(latitude)) * cos(radians(longitude) - rad
ians(#{longitude})) + sin(radians(#{latitude})) * sin(radians(lati
tude)))) < #{radius}")
9.     List<PoiEntity> findNearbyPoi(@Param("latitude") double latit
ude, @Param("longitude") double longitude, @Param("radius") double
radius);
10.    @Select("SELECT DISTINCT poi_type FROM poi")
11.    List<String> getDistinctPoiTypes();

```

**showAllPoi():** 这个方法用于查询 poi 表中的所有记录。它使用了 `SELECT * FROM poi` 这个 SQL 语句来获取所有的兴趣点信息。

**showPoiByPoi\_Id(Integer poi\_id):** 这个方法用于根据 poi\_id 查询特定的兴趣点信息。它使用了 `SELECT * FROM poi WHERE poi_id = #{poi_id}` 这个 SQL 语句来获取指定 poi\_id 的兴趣点信息。

**findNearbyPoi(double latitude, double longitude, double radius):** 这个方法用于查询在指定半径内的兴趣点。它使用了一个复杂的 SQL 语句，这个语句使用了地理位置的计算公式（Haversine 公式）来计算两点之间的距离，然后筛选出距离小于指定半径的兴趣点。

**getDistinctPoiTypes():** 这个方法用于查询所有不重复的兴趣点类型。它使用了 `SELECT DISTINCT poi_type FROM poi` 这个 SQL 语句来获取所有不重复的 poi\_type。

### 组合查询操作：

```
1.  @Select("<script>"  
2.          + "SELECT * FROM poi"  
3.          + "<where>"  
4.          + "<if test='province != null'>AND province = #{provi  
nce}</if>"  
5.          + "<if test='city != null'>AND city = #{city}</if>"  
6.          + "<if test='district != null'>AND district = #{distri  
ct}</if>"  
7.          + "<if test='poi_name != null'>AND poi_name = #{poi_n  
ame}</if>"  
8.          + "<if test='poi_address != null'>AND poi_address = #  
{poi_address}</if>"  
9.          + "<if test='poi_type != null'>AND poi_type = #{poi_t  
ype}</if>"  
10.         + "</where>"  
11.         + "</script>")  
12.     List<PoiEntity> showPoiByUserChoice(@Param("province") String  
        province,  
13.                                         @Param("city") String  
        city,  
14.                                         @Param("district") St  
        ring district,  
15.                                         @Param("poi_name") St  
        ring poi_name,  
16.                                         @Param("poi_address")  
        String poi_address,  
17.                                         @Param("poi_type") St  
        ring poi_type);
```

这是一个动态 SQL 查询语句，它使用了 MyBatis 的 `<script>`、`<where>` 和 `<if>` 标签来构建条件查询。这个查询语句的目的是根据用户的选择来查询兴趣点信息。

`<script>` 标签：这个标签用于包含整个 SQL 语句。

**<where>**标签：这个标签用于包含查询条件。它的好处是如果其中的**<if>**标签都没有满足条件（即没有生成任何查询条件），那么它会自动忽略掉 WHERE 关键字，避免了生成无效的 SQL 语句。

**<if>**标签：这个标签用于根据特定的条件来生成查询条件。例如，`<if test='province != null'>AND province = #{province}</if>`这行代码的意思是，如果 province 参数不为 null，那么就生成 AND province = #{province}这个查询条件。

多条件查询语句主要有以下优点：

**灵活性：**通过动态生成查询条件，可以根据用户的实际需求来查询数据，而不是预先定义好所有可能的查询语句。

**可读性：**使用**<script>**、**<where>**和**<if>**等标签，使得 SQL 语句的结构更加清晰，更容易理解，更加直观。

**易于维护：**如果需要修改查询条件，只需要修改对应的**<if>**标签即可，无需修改整个 SQL 语句，大大削减了维护和修改的工程量。

**防止 SQL 注入：**通过使用#{ }来绑定参数，MyBatis 会自动处理参数的转义和格式化，从而防止 SQL 注入攻击。

### 插入操作：

```
1.      @Insert("INSERT INTO poi(province, city, district, poi_name,
    poi_address, poi_type, latitude, longitude) " +
2.                  "VALUES(#{province}, #{city}, #{district}, #{poi_name},
    }, #{poi_address}, #{poi_type}, #{latitude}, #{longitude})")
3.      @Options(useGeneratedKeys = true, keyProperty = "poi_id")
4.      int addNewPoiInMap(PoiEntity poi);
```

该方法用于向 poi 表中插入新的兴趣点信息。

**addNewPoiInMap(PoiEntity poi):** 这个方法用于向 poi 表中插入新的兴趣点信息。它使用了@Insert 注解来指定插入数据的 SQL 语句。在 SQL 语句中，#{ }用于绑定参数，参数的值来自于传入的 PoiEntity 对象。@Options 注解用于指定插入数据后的操作，useGeneratedKeys = true 表示要返回自动生成的主键，keyProperty = "poi\_id"表示将自动生成的主键赋值给 poi\_id 属性。同样地，通过使用 MyBatis 的注解来指定 SQL 语句使得 SQL 语句的编写和维护变得更加简单和直观。同时，也使用#{ }来绑定参数，MyBatis 会自动处理参数的转义和格式化，从而防止 SQL 注入攻击。

### 删除操作：

```
1.      // 根据指定 id 删除数据，返回删除个数，因为是根据唯一值来搜索删除的，
    所以最多也只删除一个，所一返回值要么是 1，要么是 0，可以表示是否删除成功
2.      @Delete("DELETE FROM poi WHERE poi_id = #{poi_id}")
3.      int deletePoiByPoi_Id(@Param("poi_id") Integer poi_id);
```

该方法用于向 poi 表中删除已有的兴趣点信息。

**deletePoiByPoi\_Id(Integer poi\_id):** 这个方法用于删除 poi 表中的兴趣点信息。它使用了@Delete 注解来指定删除数据的 SQL 语句。在 SQL 语句中，#{poi\_id}用于绑定参数，参数的值来自于方法的参数。这个方法会返回一个整数，表示删除的行数。因为 poi\_id 是唯一的，所以最多只会删除一行数据，所以返回值要么是 1（表示删除成功），要么是 0（表示删除失败）。同样地，该方法通过使用 MyBatis 的注解来指定 SQL 语句，使 SQL 语句的编写和维护变得更加简单；同时使用#{ }来绑定参数，而 MyBatis 会自动处理参数的转义和格式化，从

而防止 SQL 注入攻击。

### 编辑操作:

```
1.      @Update("<script>"  
2.                  + "UPDATE poi"  
3.                  + "<set>"  
4.                  + "<if test='province != null'>province = <choose><when  
en test='province==\"NULL_VALUE\"'>null</when><otherwise>#{provinc  
e}</otherwise></choose>, </if>"  
5.                  + "<if test='city != null'>city = <choose><when test=  
'city==\"NULL_VALUE\"'>null</when><otherwise>#{city}</otherwise></  
choose>, </if>"  
6.                  + "<if test='district != null'>district = #{district}  
, </if>"  
7.                  + "<if test='poi_name != null'>poi_name = #{poi_name}  
, </if>"  
8.                  + "<if test='poi_address != null'>poi_address = #{poi  
_address}, </if>"  
9.                  + "<if test='poi_type != null'>poi_type = #{poi_type}  
, </if>"  
10.                 + "<if test='latitude != null'>latitude = #{latitude}  
, </if>"  
11.                 + "<if test='longitude != null'>longitude = #{longitu  
de}, </if>"  
12.                 + "</set>"  
13.                 + "WHERE poi_id = #{poi_id}"  
14.                 + "</script>")  
15.     int updatePoiByPoi_Id(@Param("poi_id") Integer poi_id,  
16.                             @Param("province") String province,  
17.                             @Param("city") String city,  
18.                             @Param("district") String district,  
19.                             @Param("poi_name") String poi_name,  
20.                             @Param("poi_address") String poi_addres  
s,  
21.                             @Param("poi_type") String poi_type,  
22.                             @Param("latitude") Double latitude,  
23.                             @Param("longitude") Double longitude);
```

该方法用于更新 `poi` 表中指定兴趣点信息。方法使用了 MyBatis 的 `@Update` 注解来指定更新数据的 SQL 语句。这个 SQL 语句是动态生成的，它会根据传入的参数来决定需要更新哪些字段。

`<script>` 标签：这个标签用于包含整个 SQL 语句。

`<set>` 标签：这个标签用于包含更新的字段和值。

**<if>标签：**这个标签用于根据特定的条件来生成更新的字段和值。

(例如：`<if test='province != null'>province = <choose><when test='province==\"NULL_VALUE\">null</when><otherwise>#{province}</otherwise></choose>`,  
`</if>`的意思是，如果 `province` 参数不为 `null`，那么就生成 `province = #{province}` 这个更新条件。  
如果 `province` 参数的值为 "`NULL_VALUE`"，那么就将 `province` 字段的值更新为 `null`。)

**<choose>、<when>和<otherwise>标签：**这些标签用于处理特殊的情况，例如直辖市的 `province` 和 `city` 字段可能为空。如果传入的参数值为 "`NULL_VALUE`"，那么就将对应的字段值更新为 `null`。

以上方法的优点主要有：

**灵活性：**通过动态生成更新语句，可以根据实际的需求来更新数据，而不是预先定义好所有可能的更新语句。

**可读性：**使用 `<script>`、`<set>`、`<if>` 等标签，使得 SQL 语句的结构更加清晰，更容易理解。

**易于维护：**如果需要修改更新条件，只需要修改对应的 `<if>` 标签即可，无需修改整个 SQL 语句，减少了工作量。

**防止 SQL 注入：**通过使用 `#{}</code>` 来绑定参数，MyBatis 会自动处理参数的转义和格式化，从而防止 SQL 注入攻击。

### 3. Service 层：

Service 层，将其应用的 Mapper 那一层的各种方法进行进一步封装，并添加一些相关操作来规范一些参数的输入和返回的结果。通过添加一些错误处理的操作相关操作，使得程序依赖封装性更好，安全性能优良。

Service 层通常用于处理业务逻辑，而 Mapper 层则负责与数据库进行交互。通过在 Service 层中调用 Mapper 层的方法，可以将业务逻辑和数据库操作分离，使得代码更加模块化和易于维护。

在这个方法中，我们传入了一系列参数，包括 `poi_id`、`province`、`city`、`district`、`poi_name`、`poi_address`、`poi_type`、`latitude` 和 `longitude`。这些参数分别对应了 `poi` 表中的各个字段。通过调用 `poiMapper.updatePoiByPoi_Id` 方法，可以将这些参数传入，执行更新操作。

该方法返回值一个布尔值，表示更新操作是否成功。若 `poiMapper.updatePoiByPoi_Id` 方法的返回值大于 0，那么这个方法就返回 `true`，表示更新成功；否则，返回 `false`，表示更新失败。

该方法的主要优点包括：

1. 模块化设计：通过将业务逻辑与数据库操作分离到不同的层次，代码更加模块化。Mapper 层专注于与数据库的直接交互，而 Service 层负责处理业务逻辑，使代码更清晰、更容易理解。

2. 简化上层逻辑：Service 层封装了复杂的业务逻辑，上层（如 Controller 层）调用时无需关心底层的实现细节。这样，Controller 层可以专注于处理请求并将结果返回给客户端。

3. 灵活性和复用性：Service 层方法可以通过组合多个 Mapper 层的方法来处理更复杂的业务逻辑。在不同的 Service 层方法中也可以重复使用相同的 Mapper 方法，提高代码的复用性和灵活性。

4. 统一的错误处理：Service 层可以统一处理 Mapper 层可能出现的各种错误，并返回明确的错误信息。通过规范化错误处理，提高了代码的安全性和稳定性。

5. 业务逻辑的集中化：业务逻辑集中在 Service 层，使得代码更易于维护和测试。只需修改 Service 层即可调整业务逻辑，而不需要在各个 Controller 或 Mapper 层修改代码。

6. 防止 SQL 注入：通过使用参数化查询方式，Service 层调用的 Mapper 方法可以有

效防止 SQL 注入风险，确保应用程序的安全性。

### 单条件查询：

```
1.     // 查询所有 POI
2.     public List<PoiEntity> showAllPoi(){
3.         return poiMapper.showAllPoi();
4.     }
5.     public List<PoiEntity> showPoiByPoi_Id(Integer poi_id){
6.         return poiMapper.showPoiByPoi_Id(poi_id);
7.     }
8.     public List<PoiEntity> findNearbyPoi(double latitude, double longitude, double radius){
9.         return poiMapper.findNearbyPoi(latitude, longitude, radius);
10.    }
11.    public List<String> getDistinctPoiTypes(){
12.        return poiMapper.getDistinctPoiTypes();
13.    }
```

**showAllPoi()**: 这个方法用于查询 poi 表中的所有记录。它调用了 poiMapper.showAllPoi() 方法来执行这个操作。这个方法的返回值是一个 PoiEntity 对象的列表，每个 PoiEntity 对象代表了一个兴趣点。

**showPoiByPoi\_Id(Integer poi\_id)**: 这个方法用于根据 poi\_id 查询特定的兴趣点信息。它调用了 poiMapper.showPoiByPoi\_Id(poi\_id) 方法来执行这个操作。这个方法的返回值也是一个 PoiEntity 对象的列表。

**findNearbyPoi(double latitude, double longitude, double radius)**: 这个方法用于查询在指定半径内的兴趣点。它调用了 poiMapper.findNearbyPoi(latitude, longitude, radius) 方法来执行这个操作。这个方法的返回值是一个 PoiEntity 对象的列表，每个 PoiEntity 对象代表了一个在指定半径内的兴趣点。

**getDistinctPoiTypes()**: 这个方法用于查询所有不重复的兴趣点类型。它调用了 poiMapper.getDistinctPoiTypes() 方法来执行这个操作。这个方法的返回值是一个字符串的列表，每个字符串代表了一个不重复的兴趣点类型。

以上方法都提供了一种简单的方式来访问数据库中的数据，使得上层的代码（如 Controller 层或 UI 层）可以更加简洁和易于理解。同时，这些方法也隐藏了底层的数据库操作细节，使得代码更加模块化和易于维护。

### 组合查询：

```
1.     // 组合查询，有限制条件索引的就在指定参数位置传参，没有限制条件索引的就在指定参数位置传NULL，全NULL 其实就是 showAllPoi()，最后返回所有搜到的PoiEntity
2.     public List<PoiEntity> showPoiByUserChoice(String province,
3.                                                 String city,
4.                                                 String district,
5.                                                 String poi_name,
6.                                                 String poi_address,
```

```
7.                                         String poi_type){  
8.             return poiMapper.showPoiByUserChoice(province, city,distr  
ict,poi_name,poi_address,poi_type);  
9.         }  
10.    }
```

该方法封装了 Mapper 层的 showPoiByUserChoice 方法，用于执行组合查询，根据用户的选择来查询兴趣点信息。

在这个方法中，传入了一系列参数，包括 province、city、district、poi\_name、poi\_address 和 poi\_type。这些参数对应了 poi 表中的各个字段。然后调用 poiMapper.showPoiByUserChoice 方法，将这些参数传入，执行组合查询。

这个方法的最大优势是灵活性，它允许用户根据自己的需求来选择查询条件。例如，如果用户只想查询某个省份的兴趣点，那么他们可以只传入 province 参数，而其他参数都传入 NULL。如果用户想查询所有的兴趣点，那么系统就可以将所有的参数都传入 NULL，相当于调用了 showAllPoi()方法。

#### 添加操作：

```
1.     // 添加相关数据，最后返回是否添加成功  
2.     public PoiEntity addNewPoiInMap(PoiEntity new_poiEntity){  
3.         //PoiEntity poi = new PoiEntity(province,city,district,po  
i_name,poi_address,poi_type,latitude,longitude);  
4.         if(poiMapper.addNewPoiInMap(new_poiEntity)==0){  
5.             return null;  
6.         }  
7.         return new_poiEntity;  
8.     }
```

以上是 Service 层中的增添数据方法，它封装了 Mapper 层的 addNewPoiInMap 方法。

在这个方法中，传入了一个 PoiEntity 对象，这个对象包含了要添加的兴趣点的所有信息。然后调用 poiMapper.addNewPoiInMap(new\_poiEntity)方法，将这个 PoiEntity 对象传入，执行了一个插入操作。

这个方法的返回值是一个 PoiEntity 对象。如果插入操作成功，那么这个方法就返回传入的 PoiEntity 对象；否则，返回 null，表示插入失败。

该方法提供了一种简单的方式来添加新的兴趣点信息，使得上层的代码（如 Controller 层或 UI 层）可以更加简洁和易于理解。同时，这种方式也隐藏了底层的数据库操作细节，使得代码更加模块化和易于维护。

#### 删除操作：

```
1.     // 根据指定 id 删除数据，最后返回是否删除成功  
2.     public boolean deletePoiByPoi_Id(Integer poi_id){  
3.         return (poiMapper.deletePoiByPoi_Id(poi_id))>0;  
4.     }
```

以上是 Service 层中的删除数据方法，它封装了 Mapper 层的 deletePoiByPoi\_Id 方法。

在这个方法中，传入了一个 poi\_id 参数，该参数对应了要删除的兴趣点的 poi\_id。然后调用 poiMapper.deletePoiByPoi\_Id(poi\_id)方法，将这个 poi\_id 参数传入，执行删除操作。

该方法的返回值是一个布尔值。若 `poiMapper.deletePoiByPoi_Id(poi_id)` 方法的返回值大于 0，那么这个方法就返回 `true`，表示删除成功；否则，返回 `false`，表示删除失败。

该方法提供了一种简单的方式来删除兴趣点信息，使得上层的代码（如 Controller 层或 UI 层）可以更加简洁和易于理解。同时也隐藏了底层的数据库操作细节，使得代码更加模块化和易于维护。

### 选择更新操作：

```
1.      // 更改指定的poi_id 的指定数据，注意对于直辖市province 和city 可能
        空为，所以对这两个要加以限制，最后返回是否更新成功
2.      public boolean updatePoiByPoi_Id(Integer poi_id,
3.                                         String province,
4.                                         String city,
5.                                         String district,
6.                                         String poi_name,
7.                                         String poi_address,
8.                                         String poi_type,
9.                                         Double latitude,
10.                                        Double longitude){}
11.     return (poiMapper.updatePoiByPoi_Id(poi_id,province,city,
12.                                         district,poi_name,poi_address,poi_type,latitude,longitude))>0;
13. }
```

该方法封装了 Mapper 层的 `updatePoiByPoi_Id` 方法。这个方法用于更新 `poi` 表中指定兴趣点的信息。

在这个方法中，传入了一系列参数，包括 `poi_id`、`province`、`city`、`district`、`poi_name`、`poi_address`、`poi_type`、`latitude` 和 `longitude`。方法调用 `poiMapper.updatePoiByPoi_Id`，将这些参数传入，执行了一个更新操作。

方法的返回值是一个布尔值，表示更新操作是否成功。如果 `poiMapper.updatePoiByPoi_Id` 方法的返回值大于 0，那么这个方法就返回 `true`，表示更新成功；否则，返回 `false`，表示更新失败。

### 4. Controller 层：

通过 API 提供前端与后端的交互接口。将其应用的 Service 的各种方法进一步封装起来，并给每一个封装起来的方法提供一个 API 调用接口，供前端连接调用使用。

Controller 层在 Spring MVC 框架中起着非常重要的作用，它主要负责处理用户的请求。Controller 层中的每个方法都对应了一个特定的请求路径（URL），这些方法用于处理对应的 HTTP 请求。

在本项目中，每个 Controller 类的方法都使用了 Spring MVC 的注解（如 `@RequestMapping`、`@GetMapping`、`@PostMapping` 等）来指定对应的请求路径。

本项目 Controller 层的主要优点有：

1. 模块化：将请求处理逻辑封装在 Controller 层使代码更具模块化。Controller 层只关注处理 HTTP 请求和响应，将复杂的业务逻辑转交给 Service 层处理。这种职责分离简化了代码的组织结构。

2. 易于维护：将请求处理逻辑集中在 Controller 层，任何与请求处理相关的更改都可以在 Controller 层直接实现，不会影响其他层的代码。这使得代码维护和更新变得更加容易。

3. 灵活性：可以在 Controller 类中定义多个方法，每个方法可以对应一个特定的请求路径或 HTTP 方法。这样开发者可以根据应用需求来灵活创建和管理 Controller 类中的方法，适应不同的请求处理需求。

4. 请求处理桥梁：Controller 层是应用程序和用户之间的桥梁。它将用户的 HTTP 请求映射到应用程序的业务逻辑中，并生成适当的响应返回给用户。Controller 层确保应用程序可以准确接收和处理来自客户端的请求。

5. 丰富的注解支持：Spring Boot 提供了大量用于 Controller 层的注解，开发者可以使用这些注解轻松处理请求参数、路径变量、请求体等，提高开发效率。

6. 数据转换：在 Controller 层，可以灵活地使用 Spring 的数据绑定功能，将 HTTP 请求数据转换为 Java 对象，也可以将 Java 对象转换为 JSON、XML 等响应格式，从而方便地处理不同类型的需求。

## 6.2 接口列表

方法	路径	参数	描述
GET	/api/poi/all	无	获取所有 POI
GET	/api/poi/{id}	id（路径参数）	根据 ID 获取 POI
GET	/api/poi/nearby	latitude（查询参数），longitude（查询参数），radius（查询参数）	查询附近的 POI
GET	/api/poi/search	province、city、district、poi_name、poi_address、poi_type (查询参数均可选)	根据用户选择查询 POI
POST	/api/poi/add	poiEntity（请求体）	在地图中添加新的 POI
DELETE	/api/poi/delete/{id}	id（路径参数）	根据 ID 删除 POI
PUT	/api/poi/update/{id}	id（路径参数），poiEntity（请求体）	根据 ID 更新 POI

## 七、前端设计与实现

### 7.1 界面设计

本项目的前端主要使用 `Vue.js` 和 `Element UI` 框架来实现。

前端交互逻辑主要通过设定不同模式开关实现，在一般状态下，用户通过主页提供的交互工具调取、查询数据，并根据用户锁定的数据范围生成统计图、热力图等信息图表以供用户参考和观察。而当用户点击增、删、改操作对应按钮，模式切换发生，点击不同按钮进入对应模式，以此和一般状态做出区分来控制当前页面中不同的鼠标活动响应函数。

界面设计包括以下组件：

#### 1. 搜索和过滤栏

功能：

用户可以通过输入框按城市、区县、名称和类型对 POI 数据进行过滤和搜索。



对应函数逻辑：

##### 1. filterPOIData:

从绑定的数据模型中获取城市、区县、名称和类型的搜索条件。

构建带有搜索参数的 `GET` 请求。

将请求发送到后端接口以获取过滤后的 POI 数据。

根据响应数据更新地图上展示的 POI 图层。

##### 2. clearSearch:

清空搜索栏中各个输入框的内容。

将绑定的数据模型中的搜索条件也清空。

#### 2. 地图展示

功能：

使用 `OpenLayers` 库实现地图绘制和显示。

组件伪代码：

```
<地图 id="map" 类="map-container">
```

对应函数逻辑：

##### 1. initMap:

创建一个新的 `OpenLayers` 地图实例，设置地图目标为指定的 DOM 元素。

配置地图的初始视图，包括中心位置和缩放级别。

注册地图点击事件，以便在添加、编辑和删除模式下相应操作。

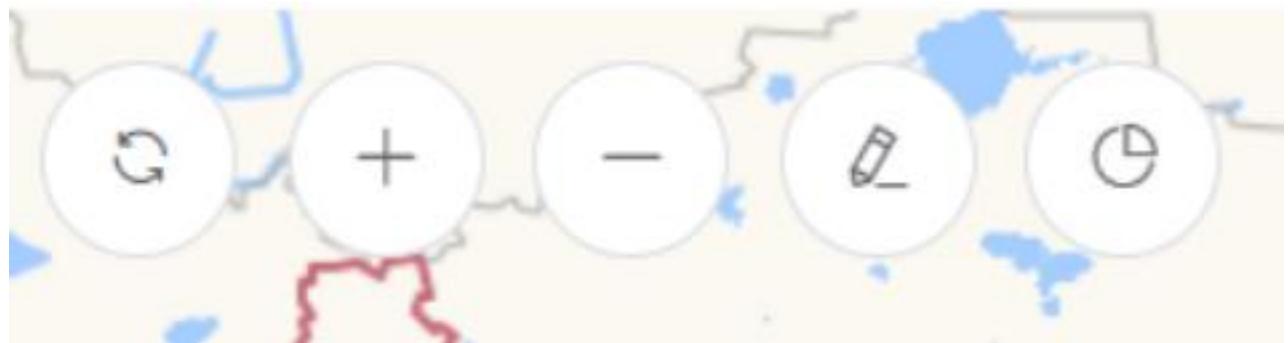
注册地图右键点击事件，以便退出特定的操作模式。

#### 3. 地图控制按钮

功能：

包含刷新、添加、删除、编辑、统计按钮。用以实现对地图上 POI 数据的增删改查操作和统

计功能。



对应函数逻辑：

**1. refreshMap:**

清空当前的 POI 数据和地图上的 POI 图层。

重新定位地图视图到初始中心位置，并设置默认缩放级别。

**2. addData:**

激活添加模式，使地图点击事件用于获取新增 POI 的经纬度。

将新的 POI 数据传递给相应的添加表单。

**3. removeData:**

切换删除模式的状态，使地图点击事件用于选择删除的 POI。

**4. editData:**

激活编辑模式，使地图点击事件用于选择待编辑的 POI。

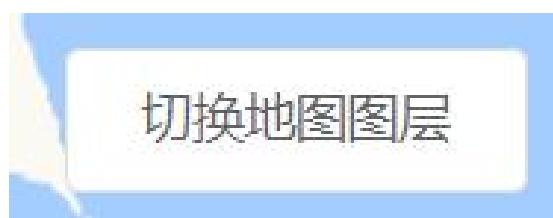
**5. handleClick:**

打开数据统计弹窗，将数据统计图展示在弹窗中。

#### **4. 图层切换按钮**

功能：

用于切换地图图层，以查看不同的地图数据。



对应函数逻辑：

**switchLayer:**

切换当前的地图图层索引到下一个可用图层。

清除当前地图上的所有图层。

使用新的图层配置刷新地图视图。

#### **5. 快捷跳转按钮**

功能：

通过点击按钮快速跳转到特定的城市，如武汉。



对应函数逻辑:

jumpToWUHAN:

设置地图视图的目标中心为武汉市的经纬度。

以平滑的动画方式将地图视图移动到目标位置。

## 6. 添加地点数据表单

功能:

通过弹窗表单，让用户填写新地点信息并添加到地图中。

添加地点数据 ×

省份

城市

\* 区县

\* 地点名称

\* 地点地址

\* 地点类型

纬度

经度

清空 取消 确认添加

对应函数逻辑：

1. confirmAddData:

验证新 POI 数据的完整性，确保所有必要字段均有值。

将新 POI 数据通过 POST 请求发送到后端接口。

将后端返回的 POI 数据添加到当前 POI 数据列表中，并刷新地图视图。

关闭添加表单弹窗并退出添加模式。

2. validateNewPOI:

检查传入的 POI 数据对象的所有必要字段，确保它们均有值。

3. cancelAddData:

清空新 POI 数据表单。

关闭添加表单弹窗并退出添加模式。

## 7. 编辑地点数据表单

功能：

通过弹窗表单，让用户编辑现有地点信息。

编辑地点数据 ×

省份

城市

\* 区县

\* 地点名称

\* 地点地址

\* 地点类型

取消 确认修改

对应函数逻辑：

1. `confirmEditData`:

验证编辑的 POI 数据的完整性。

通过 PUT 请求将修改后的 POI 数据发送到后端接口。

更新本地 POI 数据列表，并刷新地图视图。

关闭编辑表单弹窗并退出编辑模式。

2. `validateEditPOI`:

检查传入的 POI 数据对象的所有必要字段，确保它们均有值。

3. `updatePOIData`:

向后端发送 PUT 请求更新 POI 数据。

更新 POI 数据列表中的相应项，并刷新地图上的 POI 图层。

4. `cancelEditData`:

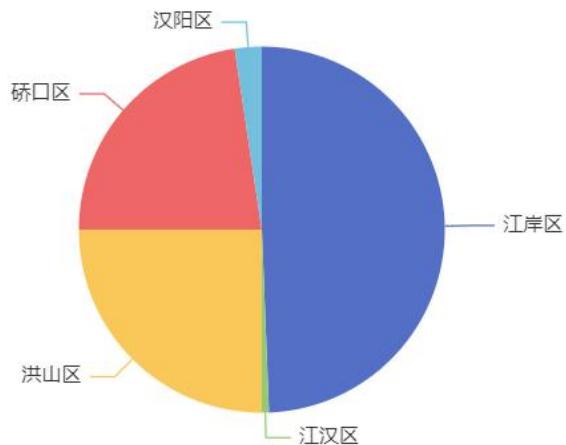
关闭编辑表单弹窗并退出编辑模式。

## 8. 统计图弹窗

地点数据统计

×

地点数据统计



[按省份](#) [按城市](#) [按区县](#) [按类型](#)

对应函数逻辑：

`calculateStatistics`: 遍历 POI 数据，根据省份、城市、区县和类型等维度统计 POI 数量，将结果转换为 ECharts 需要的格式。

`switchChart`: 根据用户选择的维度更新统计图表，调用 `renderChart` 渲染新的数据。

`renderChart`: 初始化 ECharts 图表，配置各项图表参数，包括标题、提示框和数据系列，将传入的数据设置为图表的系列并更新图表。

`showStatisticsPopup`: 设置弹窗可见并在下一次 Vue 更新周期中调用 `calculateStatistics`

计算统计数据并渲染图表。

`closeStatisticsPopup`: 关闭统计弹窗。

对话框标题为地点数据统计，并在其中嵌入了一个 `ECharts` 图表容器。

图表默认显示 `POI` (兴趣点) 按类型 (`poi_type`) 的分类统计结果。

图表切换:

对话框底部提供了四个按钮，分别对应按省份、城市、区县和类型的分类统计。

用户点击任意一个按钮，都会触发 `switchChart` 方法，该方法会调用 `renderChart` 方法来更新图表的数据和展示。

数据计算:

在对话框显示之前（通过 `showStatisticsPopup` 方法），会调用 `calculateStatistics` 方法来计算 `POI` 数据在不同分类下的统计结果。

`calculateStatistics` 方法遍历 `POI` 数据，统计每个 `POI` 在不同分类下的数量，并将结果转换为 `ECharts` 所需的格式。

图表渲染:

`renderChart` 方法负责使用 `ECharts` 渲染图表。它首先初始化一个 `ECharts` 实例，然后配置图表的各项选项（如标题、提示框、系列等）。

根据用户选择的分类（或默认分类），`renderChart` 方法将对应的统计数据设置到图表的系列中，并通过 `setOption` 方法更新图表。

对话框控制:

对话框的显示和隐藏通过 `Vue` 的数据绑定和事件监听实现。

`showStatisticsPopup` 方法设置 `showStatisticsDialog` 为 `true` 以显示对话框，并在下一个 `Vue` 更新周期（通过 `$nextTick`）中计算和渲染图表。

`closeStatisticsPopup` 方法设置 `showStatisticsDialog` 为 `false` 以隐藏对话框。

图表类型:

当前实现的图表类型为饼图 (`type: 'pie'`)，用于展示不同分类下的数量占比。

饼图的配置项包括标题、提示框、半径、数据项等，同时设置了数据项在选中状态下的样式（如阴影模糊度、阴影颜色等）。

通过上述功能，用户可以方便地查看 `POI` 数据在不同分类下的统计结果，并通过简单的按钮点击操作来切换分类，实现快速的数据分析和可视化展示。

## 7.2 前后端交互

前端与后端交互主要通过 `Axios` 进行异步请求，使用 `RESTful API` 与后端进行交互。以下是主要的交互场景和相应的代码实现，通过以下代码和后端的交互，整个应用得以实现数据的增删改查，并保持数据的同步更新与界面呈现的连贯性：

### 1. 获取所有 `POI`:

`getAllPoi()`

说明：该函数通过发送 `GET` 请求到后端的 `/api/poi/all` 接口，获取所有 `POI` 数据，并将其存储在组件的 `poiData` 变量中。之后会调用 `updatePOILayer` 函数更新地图视图。

处理:

错误处理：在 `.catch` 块中使用用户友好的错误消息提示。

数据刷新：调用 `updatePOILayer` 函数，在地图上刷新 `POI` 图层数据。

## 2. 搜索 POI:

`filterPOIData()`

说明：通过读取用户输入的搜索条件，将其转换为请求参数，并使用 `axios.get` 方法发送带有这些参数的 GET 请求至 `/api/poi/search`。请求成功后，更新 `poiData` 并刷新地图视图。

处理：

空搜索提示：在没有输入搜索条件的情况下提示用户至少输入一个搜索条件。

加载指示：在请求发送和处理期间显示加载指示器，以改善用户体验。

数据刷新：在地图上刷新 POI 图层数据。

## 3. 添加 POI:

`confirmAddData()`

说明：使用 `axios.post` 方法，将新 POI 数据发送到 `/api/poi/add` 接口。若成功，将返回的数据添加至 `poiData` 列表中，并刷新地图视图。

处理：

表单验证：在添加数据之前调用 `validateNewPOI` 方法，确保新 POI 数据的完整性。

数据刷新：在地图上刷新 POI 图层数据。

## 4. 删除 POI:

`deletePOI(poId)`

说明：使用 `axios.delete` 方法，将请求发送到 `/api/poi/delete/{poId}` 接口。请求成功后，更新 `poiData` 列表并刷新地图视图。

处理：

删除确认：在删除操作前，弹出对话框要求用户确认删除。

数据刷新：在地图上刷新 POI 图层数据。

## 5. 编辑 POI:

`updatePOIData(poi)`

说明：使用 `axios.put` 方法，将编辑后的 POI 数据发送到 `/api/poi/update/{poi_id}` 接口。请求成功后，更新 `poiData` 列表，并刷新地图视图。

处理：

表单验证：在提交数据前，调用 `validateEditPOI` 方法确保 POI 数据的完整性。

数据刷新：在地图上刷新 POI 图层数据。

以上设计的优势：

模块化设计：将获取、添加、搜索、删除和编辑 POI 的功能独立成单独的函数，实现职责明确且模块化的代码设计，有助于提高代码的可维护性。

数据同步：通过与后端 RESTful API 的交互，确保前端的 POI 数据与后端数据库保持一致，提供实时、准确的数据呈现。

用户反馈：在每个操作过程中都提供了用户友好的错误提示和状态提示信息，有助于改善用户体验，让用户清楚操作状态。

一致性：每个操作之后都会调用 `updatePOILayer` 刷新地图上的 POI 数据，确保地图视图与最新的数据保持一致。

表单验证：在添加或编辑 POI 数据前进行验证，确保发送到后端的数据是完整且正确的，防止数据出错。

## 八、部署到云服务器

### 1. 云服务器的申请

本项目使用云服务器 ECS，这是最简单易用的，也是阿里云主流的一种服务器：

The screenshot shows the Alibaba Cloud product catalog under the '计算' (Compute) category. On the left, there's a sidebar for '类目筛选' (Category Filter) and '产品类别' (Product Category), listing categories like '计算' (25), '容器' (7), '存储' (16), etc. The main area displays several server types:

- 云服务器 ECS 免费试用**: Described as a safe, elastic, and伸缩的云计算服务.
- GPU 云服务器**: Provides GPU computing power for deep learning, scientific computing, and video processing.
- 弹性裸金属服务器**: Offers elastic and isolated high-performance computing services.
- 弹性容器实例 ECI**: No need to manage底层 ECS servers; just provide a镜像 to run containers.
- 专有宿主机**: Tailored solutions for enterprise users, featuring physical resource独享 and flexible deployment.
- 云虚拟主机**: Enhanced performance for LNMP websites with improved stability and response speed.
- 计算巢服务 免费试用**
- 轻量应用服务器**
- FPGA 云服务器**

本项目选择的服务器配置为 2 核 4G 的版本：

云服务器配置这里，地域位置选择靠近本地的，操作系统选择 linux（更加方便体量小），其他项使用默认配置。

The screenshot shows the configuration interface for a '2核4G' (2 vCPU, 4 GB) ECS instance. The fields are as follows:

- 地域**: Set to '华东 1 (杭州)'.
- 可用区**: Set to '随机分配'.
- 实例规格**: Set to '2CPU 4GiB' ( ecs.s6-c1m2.large ).
- 操作系统**: Set to 'Alibaba Cloud Linux / Alibaba Cloud Linux 3.2104 LTS 64位'.
- 系统盘**: Set to '高效云盘' with a size of '100 GiB'.
- 网络类型**: Set to '专有网络'.
- 专有网络**: Set to '默认专有网络'.
- 虚拟交换机**: Set to '默认交换机'.
- 带宽**: Set to '按固定带宽' with a value of '3 Mbps'.

## 2.云服务器的配置

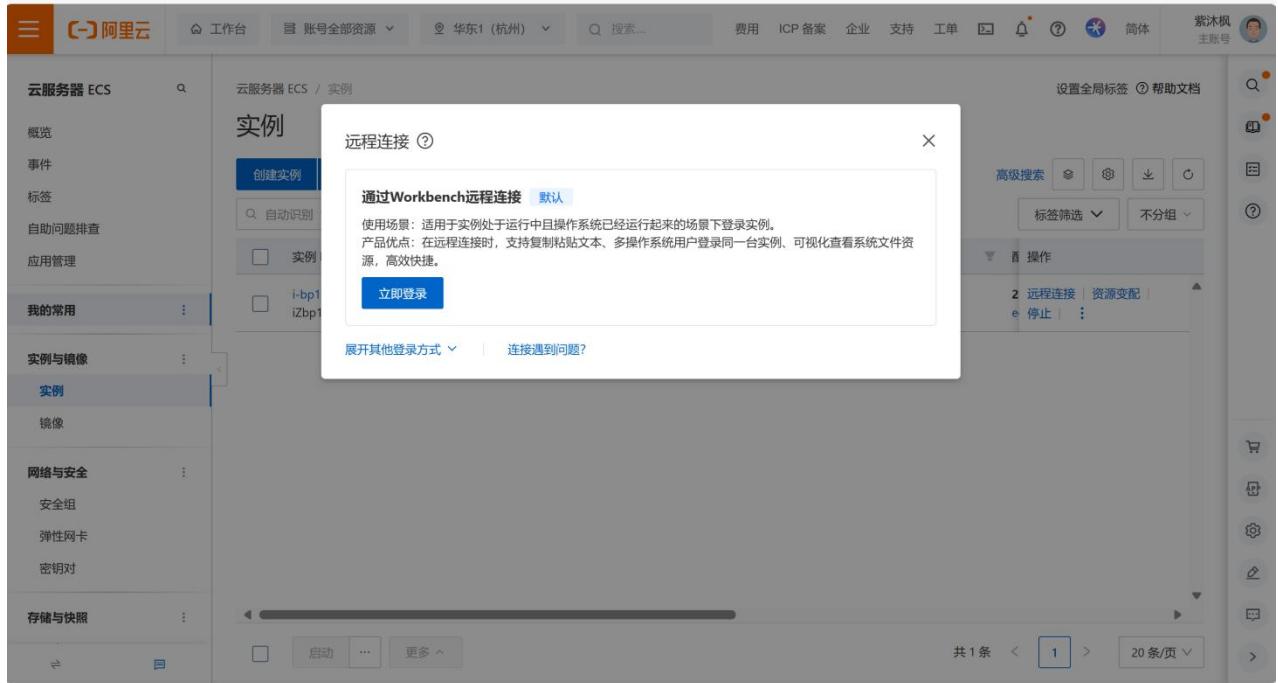
进入服务器的概览界面，点击页面左侧-实例与镜像-实例，可以跳转到实例界面，对我们的服务器进行管理。

The screenshot shows the Alibaba Cloud ECS console's overview page. On the left sidebar, under '我的常用' (My常用), the '实例与镜像' (Instances and Images) section is selected. It displays a summary of resources: 1 Cloud Server (运行中 - Running), 0即将过期 (Due to expire), 0已过期 (Expired), 0近期创建 (Recently created), and 0快照 (Snapshots). Below this, a specific instance 'i-bp10nfmt3mmfc9bv71a' is listed with details: Name: iZbp10nfmt3mmfc9bv71aZ, Region: 华东1 (杭州), Creation Time: 2024年4月15日 21:24:00, Public IP: 120.55.191.136. To the right, there are tabs for 'CPU使用率' (CPU Usage Rate), '内存使用率' (Memory Usage Rate), and '云盘使用率' (Cloud Disk Usage Rate). A '使用快照备份' (Use Snapshot Backup) button is also present. The main content area features a '我的教程' (My Tutorials) section with links to '一键部署幻兽帕鲁游戏', '快速搭建网站', and '部署开发环境'.

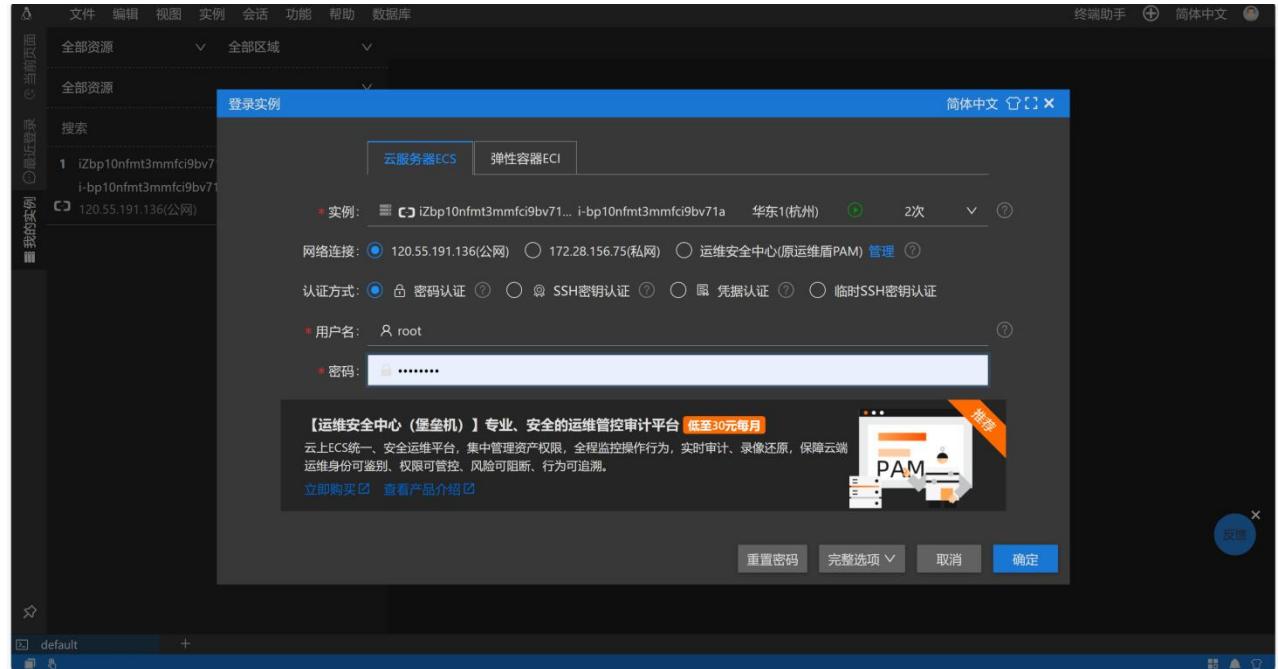
在实例界面，可以看到服务器实例，选择启动。

The screenshot shows the Alibaba Cloud ECS console's instance management interface. The left sidebar has '实例与镜像' (Instances and Images) selected, with '实例' (Instances) highlighted. In the main area, a table lists one instance: 'i-bp10nfmt3mmfc9bv71a' (Status: 已停止 (已停止), 普通停机模式). To the right of the table are buttons for '资源变配' (Resource Scaling), '启动' (Start), and '取消释放' (Release). At the bottom of the table, there are buttons for '启动' (Start), '更多' (More), and pagination controls (共1条, 1, 20条/页).

启动成功后，点击远程连接，登录服务器，进入服务器命令行界面。



第一次配置服务器时，需要重置密码，设定好密码再进入。



在配置服务器时，需要安装和配置 MySQL

The screenshot shows a terminal window titled '2. Terminal' with the following content:

```
Welcome to Alibaba Cloud Elastic Compute Service !  
Updates Information Summary: available  
 31 Security notice(s)  
    3 Important Security notice(s)  
    22 Moderate Security notice(s)  
    6 Low Security notice(s)  
Run "dnf upgrade-minimal --security" to apply all updates.More details please refer to:  
https://help.aliyun.com/document\_detail/416274.html  
Last login: Tue Apr 30 16:03:14 2024 from 47.96.60.211  
[root@iZbp10nfmt3mmfc9bv71aZ ~]# mysql -hlocalhost -uroot -p  
Enter password:  
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)  
[root@iZbp10nfmt3mmfc9bv71aZ ~]# mysql -hlocalhost -uroot -p  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 3  
Server version: 5.7.44-log Source distribution  
  
Copyright (c) 2000, 2023, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
MySQL [(none)]>
```

At the bottom of the terminal window, it says '命令终端 已连接 华东1(杭州) i-bp10nfmt3mmfc9bv71a 120.55.191.136.22 o5j6ws0nd8 14 pt 29,17 29 Rows 156 Cols 语言: en\_US.UTF-8'.

数据库云端部署成功，接下来使用数据库可视化管理工具 Navicat 远程管理服务器，实现项目与服务器的连接。

### 3. Navicat 远程连接服务器

首先，修改阿里云安全组，按以下步骤：

1.进入阿里云控制台

2.点击实例->实例详情->安全组->管理规则

The screenshot shows the Alibaba Cloud ECS instance management interface. On the left sidebar, under '实例与镜像' > '实例' tab, there is a table listing an instance:

实例 ID / 名称	状态	标签	操作系统	监控	可用区	操作
i-bp10nfmt3mmfc9bv71a iZbp10nfmt3mmfc9bv71aZ	运行中				华东1 (杭州) I	<a href="#">2 远程连接</a> <a href="#">资源变配</a> <a href="#">停止</a>   <a href="#">更多</a>

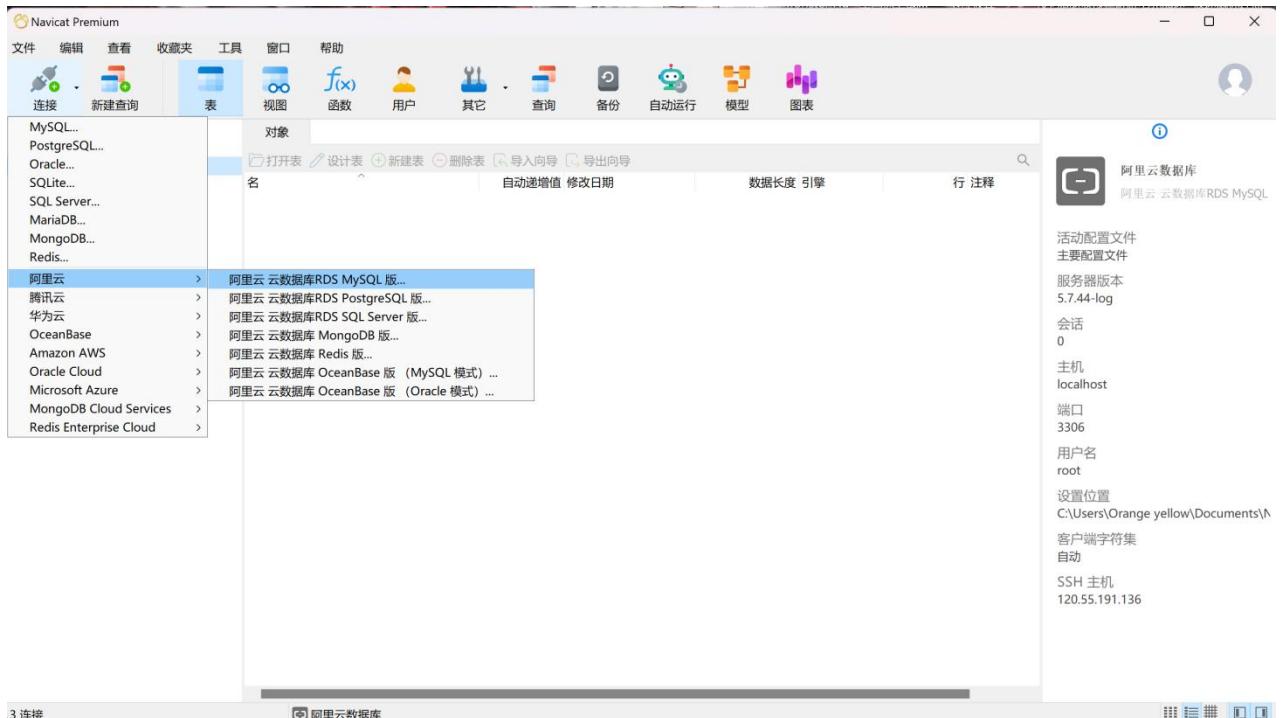
On the right side, there is a '设置全局标签' (Set Global Tag) button and a '帮助文档' (Help Document) link.

将安全组与实例绑定并点击管理规则。

进入页面后，在入方向这里选择快速添加，将 MySQL 的 3306 端口添加到入方向。

进入控制台重启服务器即可 (`systemctl restart mysql` #重启 mysql 服务器命令)

打开 Navicat，点击连接-阿里云-MySQL：



端口需要选择 MySQL 的 3306 端口，用户名与密码即为登录 MySQL 的用户名与密码。

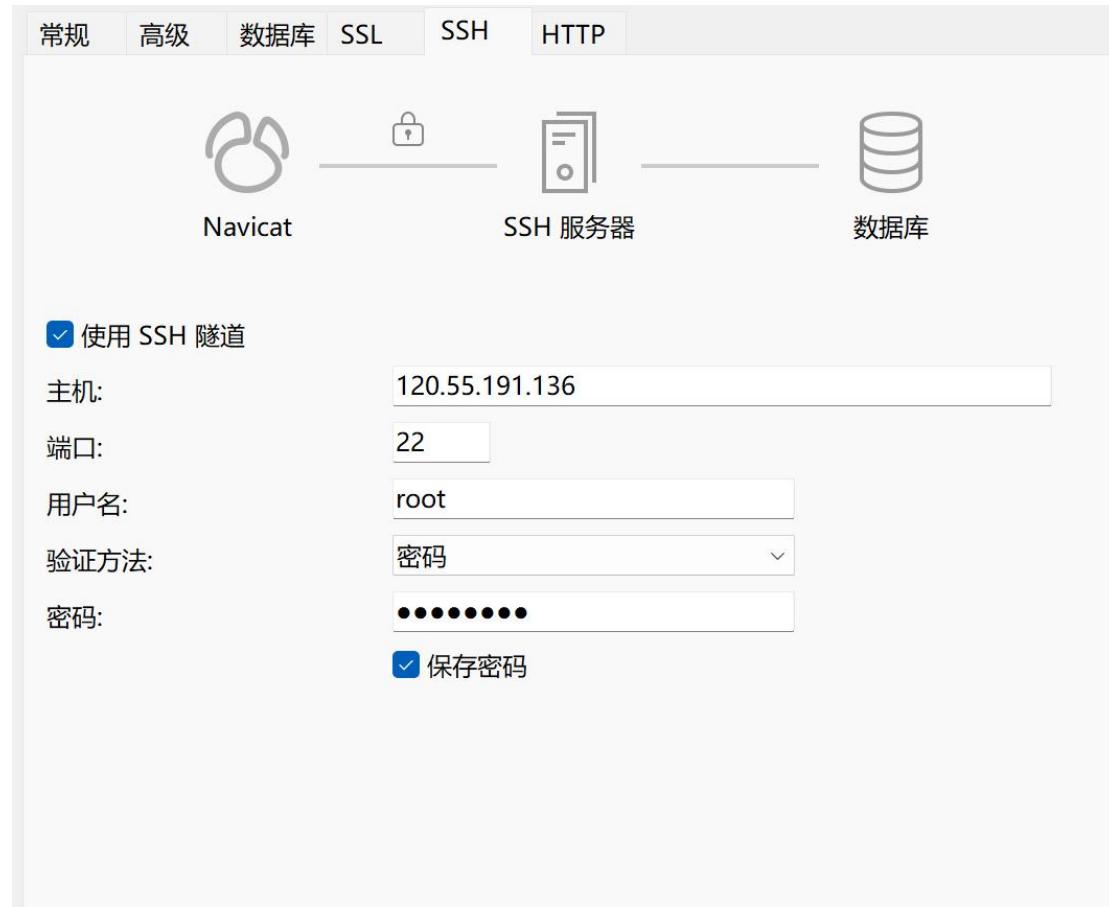
This is a detailed view of the MySQL connection configuration in Navicat. The tabs at the top are 常规 (General), 高级 (Advanced), 数据库 (Database), SSL, SSH, and HTTP. The '常规' tab is active.

Below the tabs are three connection methods represented by icons: Navicat (a circular icon with a stylized 'N'), SSH 服务器 (an icon of a server with a lock), and 数据库 (an icon of two stacked cylinders).

The configuration fields are as follows:

- 连接名: 阿里云数据库 (Connection Name)
- 主机: localhost (Host)
- 端口: 3306 (Port)
- 用户名: root (Username)
- 密码: (Password) - A masked input field showing six dots.
- 保存密码 (Save Password) - A checked checkbox.

由于使用的是 SSH 连接，因此这里只需要编辑 SSH，主机这一栏输入服务器的 IP 地址，端口选择 22（默认的服务器端口），用户名与密码与上一步同。



之后即可连接成功，下一步需要创建一个与本地数据库名称相同的数据库，打开，用运行 SQL 文件即可将我们的数据传到服务器。

The screenshot shows the Navicat Premium interface. The left sidebar displays a tree view of databases and tables, with '运行 SQL 文件...' (Run SQL File...) selected. The central area shows a table of data with columns: 名 (Name), 自动递增值 (Auto Increment Value), 修改日期 (Modification Date), 数据长度 (Data Length), 引擎 (Engine), 行 (Rows), and 注释 (Comments). On the right side, there is a detailed view of the 'mysql' database, showing objects like '阿里云数据库' (Aliyun Database), '字符集' (Character Set), and '排序规则' (Collation).

下图也可以看到能够管理我们的 poi 表数据:

The screenshot shows the Navicat Premium interface with the 'poi @webgis (阿里云数据库)' database selected. In the left sidebar, under the '表' (Tables) section, the 'poi' table is listed. The main pane displays the data from the 'poi' table, which contains 24 rows of information about various points of interest in Wuhan, China. The columns include: poi\_id, province, city, district, poi\_name, poi\_address, poi\_type, and latitude. A detailed view of one row is shown on the right, with the primary key 'poi\_id' highlighted.

poi_id	province	city	district	poi_name	poi_address	poi_type	latitude
1	湖北省	武汉市	江岸区	都椰都椰(洞庭街店)	洞庭街135号	冷饮店	30.5913
2	湖北省	武汉市	江岸区	再就业小吃店	一元街办事处二曜路8号	餐饮相关	30.5946
3	湖北省	武汉市	江岸区	没有名字的甜点店(洞庭街)	一元街洞庭街135号	购物相关场所	30.5912
4	湖北省	武汉市	江岸区	江城明珠豪生酒店(行政酒)	沿江大道182号	餐饮相关	30.5910
5	湖北省	武汉市	江岸区	周黑鸭(武汉江岸区一元路)	一元路7号锋尚时代大厦1层	餐饮相关	30.5927
6	湖北省	武汉市	江岸区	老汉口民生美食广场(烽尚)	一元街办事处一元路7号锋尚	中餐厅	30.5929
7	湖北省	武汉市	江岸区	江城明珠豪生大酒店(天堂鸭)	沿江大道182号江城明珠豪	餐饮相关	30.5909
8	湖北省	武汉市	江岸区	一元路怪味烧饼(锋尚时代)	一元路7号	餐饮相关	30.5927
9	湖北省	武汉市	江岸区	泰国金兰花旋转餐厅	沿江大道182号江城明珠豪	泰国/越南菜品餐厅	30.5907
10	湖北省	武汉市	江岸区	宜红云端茶馆(江城明珠店)	一元街街道江城明珠豪生大	茶艺馆	30.5907
11	湖北省	武汉市	EAT ME	洞庭街133号(明珠豪生酒店)	中餐厅	30.5910	
12	湖北省	武汉市	江岸区	武昌董油淹梅店(一元路总)	一元路7号锋尚时代大厦F1	中餐厅	30.5930
13	湖北省	武汉市	江岸区	老五热干面(沿江大道店)	沿江大道196号	中餐厅	30.5947
14	湖北省	武汉市	江岸区	烤烤屋海鲜碳烤酒吧	蔡锷路首善里11号楼	餐饮相关	30.5913
15	湖北省	武汉市	江岸区	两江游船餐厅(粤汉码头店)	沿江大道与蔡锷路交叉口旁	餐饮相关	30.5902
16	湖北省	武汉市	江岸区	岁时记楚宴	沿江大道198号	湖北菜(鄂菜)	30.5949
17	湖北省	武汉市	SOLEY-楼里	蔡锷路2路(与洞庭街十字路口)	西餐厅(综合风味)	30.5907	
18	湖北省	武汉市	江岸区	故襄牛肉面	蔡锷路2号	中餐厅	30.5908
19	湖北省	武汉市	江岸区	烧饼	蔡锷路与洞庭街交叉口西北	餐饮相关	30.5908
20	湖北省	武汉市	江岸区	川香园(江岸区店)	蔡锷路10号	四川菜(川菜)	30.5909
21	湖北省	武汉市	江岸区	烧guo鸟酒馆店	蔡锷路6号	日本料理	30.5908
22	湖北省	武汉市	江岸区	劉師父	蔡锷路14号	综合酒楼	30.5910
23	湖北省	武汉市	江岸区	9601精酿啤酒	蔡锷路16号	中餐厅	30.5911
24	湖北省	武汉市	江岸区	vert森蕉咖啡	蔡锷路一号	咖啡厅	30.5904

#### 4.项目连接到服务器

打开 IDEA, 编辑我们的 application.properties 文件, 将本地端口替换为服务器端口, 再确认我们的用户名与密码无误即可:

```
server.port=8082

spring.datasource.url=jdbc:mysql://120.55.191.136/webgis?serverTimezone=GMT%2B8&characterEncoding=utf-8&
spring.datasource.username=root
spring.datasource.password=888888
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# MyBatis ??
mybatis.mapper-locations=classpath:mapper/*.xml
mybatis.type-aliases-package=com.example.poi_map.Entity
```

运行项目, 可以看到我们可以成功查询存储在云端的数据:



## 九、功能展示

### 9.1 主页面辅助功能

主页概览:



从左到右依次为：地图缩放按钮、刷新按钮、新增地点按钮、删除地点按钮、编辑地点按钮、统计图表。



比例尺信息:



我们可以通过两种方式来对地图进行的放大与缩小操作(通过鼠标滚轮转动或者点击系统界面的放大缩小按钮)

放大:



缩小：



放大缩小均是以当前视图中心点为中心，调整地图的缩放级别，如图所示，地图右下角有当前地图比例尺，我们可以发现随着地图的放大与缩小操作，地图比例尺也跟着不停的发生变化。



**切换地图图层按钮：**

武汉卫星图：



全球卫星图：



在系统中设置了两个不同的地图图层，一个是高德地图，一个是高德卫星图，其设置如下：

```
mapLayers: [
  {
    id: 1, name: 'Gaode Map New', source: new XYZ({
      attributions: 'Map data © <a href="https://amap.com">Gaode Map</a>',
      url:
        'https://wprd0{1-4}.is.autonavi.com/appmaptile?lang=zh_cn&size=1&style=7&x={x}&y={y}&z={z}',
      wrapX: false
    })
  }
]
```

```

        }
    },
    {
        id: 2, name: 'Gaode Satellite', source: new XYZ({
            attributions: 'Satellite imagery © <a href="https://amap.com">Gaode Map - Satellite</a>',
            url: 'https://webst02.is.autonavi.com/appmaptile?style=6&x={x}&y={y}&z={z}'
        })
    }
],

```

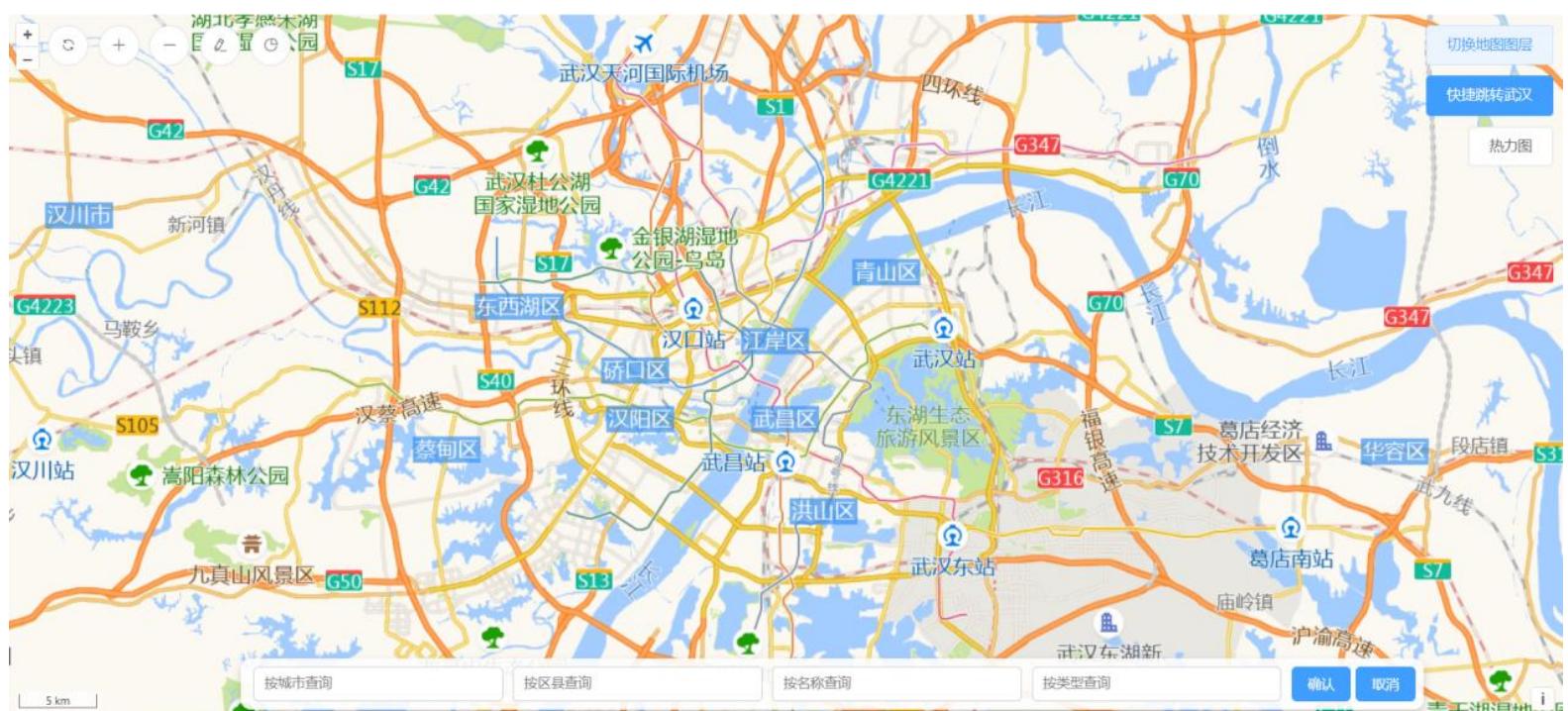
### 快速跳转到武汉按钮：

通过设置武汉的坐标，调整地图缩放级别，同时预设了缓动效果，设置 2 秒的动画时间，使得地图跳转更加平滑、自然。

```

this.map.getView().animate({
    center: universityCoordinates,
    zoom: 11,
    duration: 2000,
    easing: easeOut
});

```



### 地图刷新按钮：

点击刷新图标按钮，地图将回归初始进入系统时的默认位置与默认缩放级别。

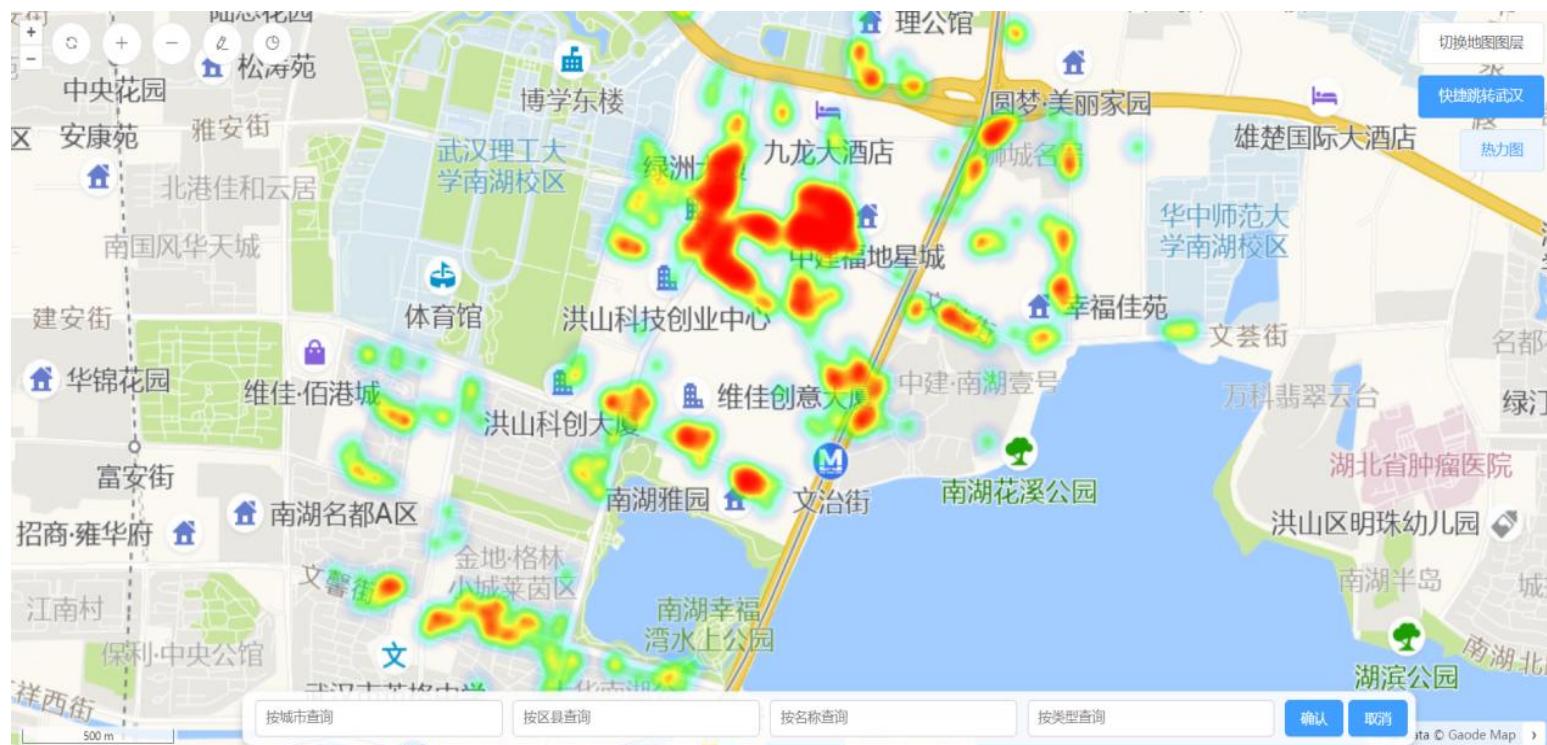


### 热力图按钮：

点击热力图按钮，系统将统计当前界面上已有的查询结果数据。统计完成后，地图将隐藏地点图标，显示热力图，再次点击则显示图标则可以取消热力图。



以上地图转化为热力图效果如下：



悬浮信息框：

展示鼠标悬停对应位置地点的相关数据信息，包括唯一标识编号、所属上级行政区划、POI类型、具体地址和经纬度信息；



统计图按钮：

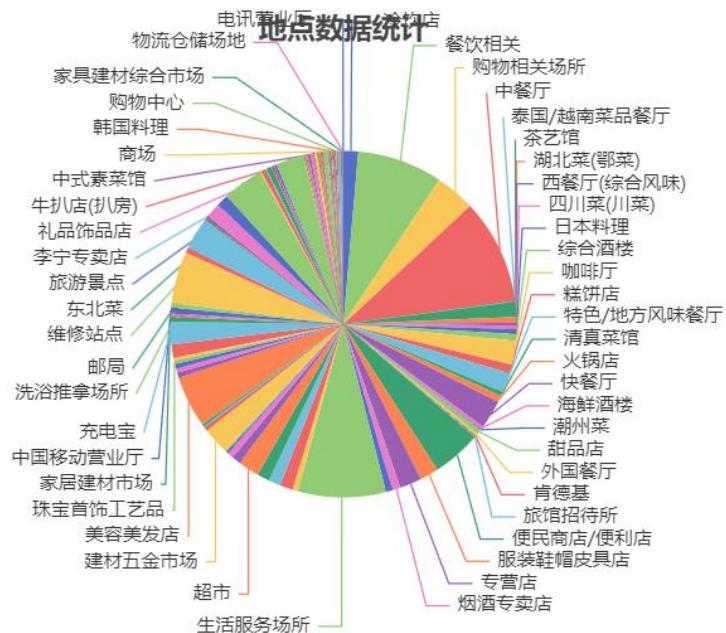
根据地图上的已有数据统计各个字段的数据信息：



点击统计图图标对应按钮，弹出扇形统计图弹窗，展示页默认为按类型统计。（统计图的数据统计效果是实时的，只统计当前查询搜索框选定的数据地点数据）

## 地点数据统计

×



按省份 按城市 按区县 按类型

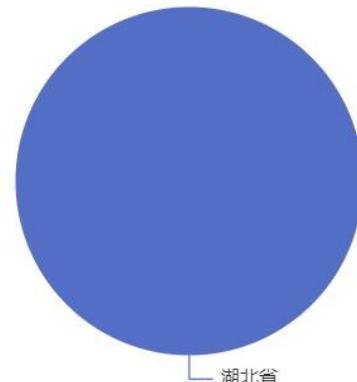
在图表弹窗下方可以选择其他字段统计信息：

按省份：

## 地点数据统计

×

## 地点数据统计



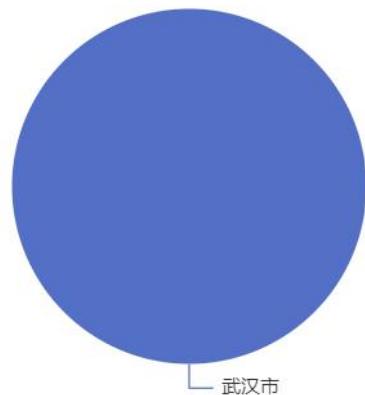
按省份 按城市 按区县 按类型

按城市：

地点数据统计

×

### 地点数据统计



[按省份](#) [按城市](#) [按区县](#) [按类型](#)

按区县：

地点数据统计

×

### 地点数据统计

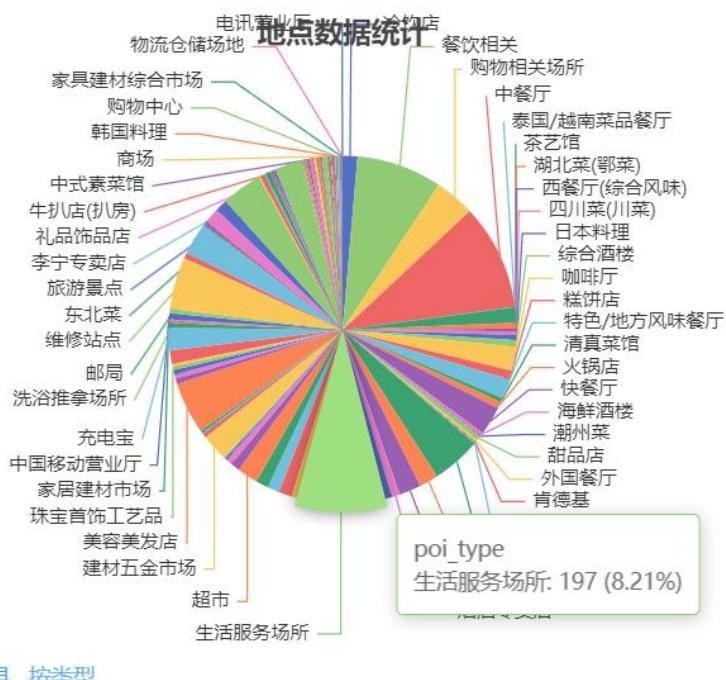


[按省份](#) [按城市](#) [按区县](#) [按类型](#)

鼠标悬停时，统计图可以展示悬停位置对应的具体数据信息：

### 地点数据统计

×



### 地点数据统计

×

### 地点数据统计

按省份 按城市 按区县 按类型

## 地点数据统计

×

### 地点数据统计

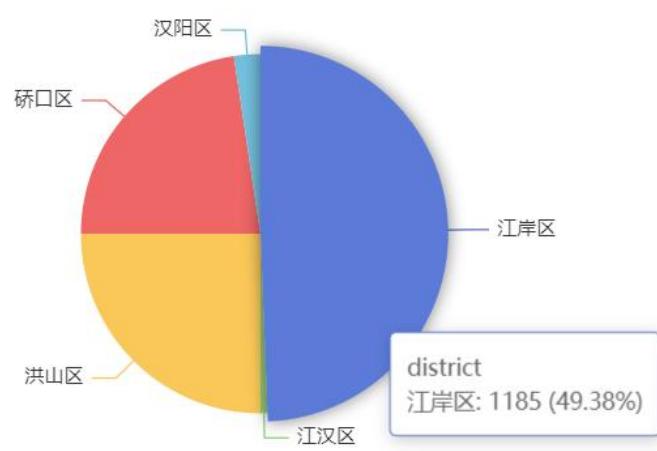


[按省份](#) [按城市](#) [按区县](#) [按类型](#)

## 地点数据统计

×

### 地点数据统计



[按省份](#) [按城市](#) [按区县](#) [按类型](#)

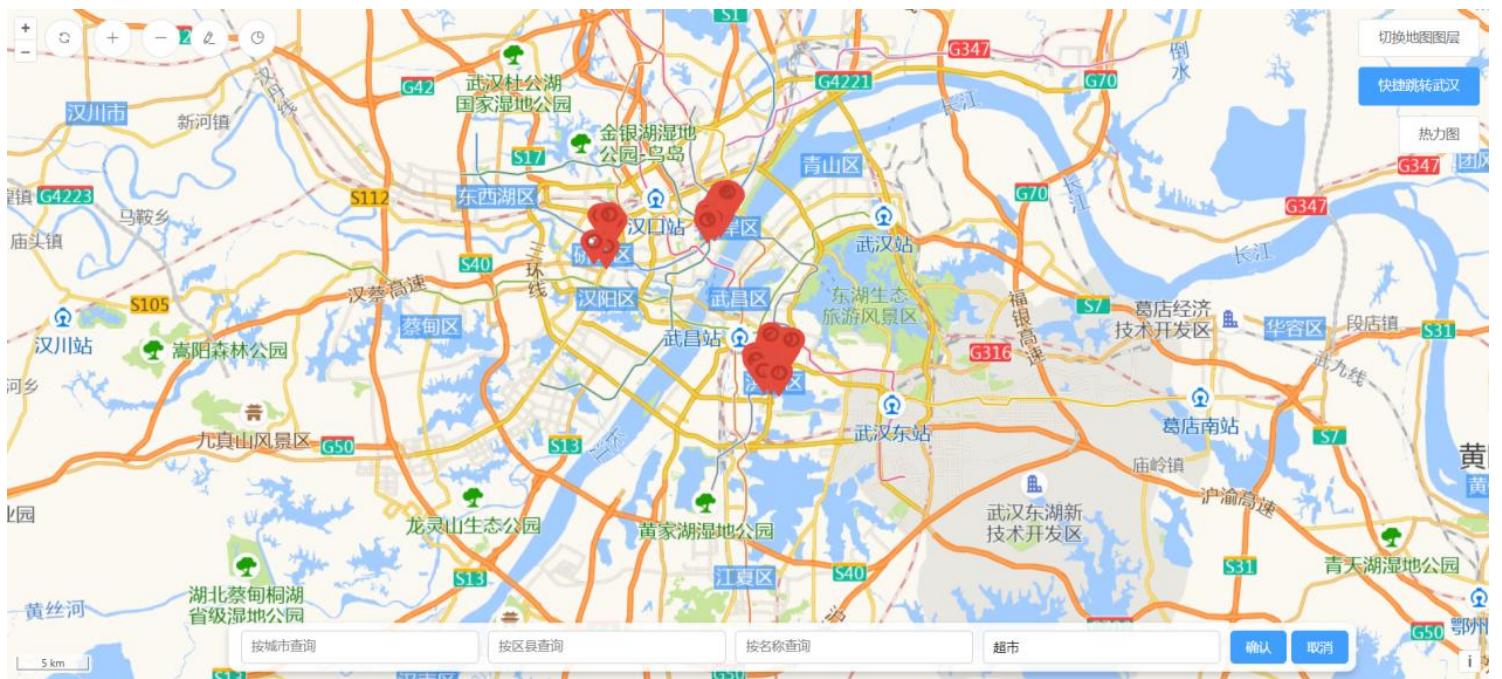
## 9.2 生活服务设施信息查询

### 查询地点搜索框：

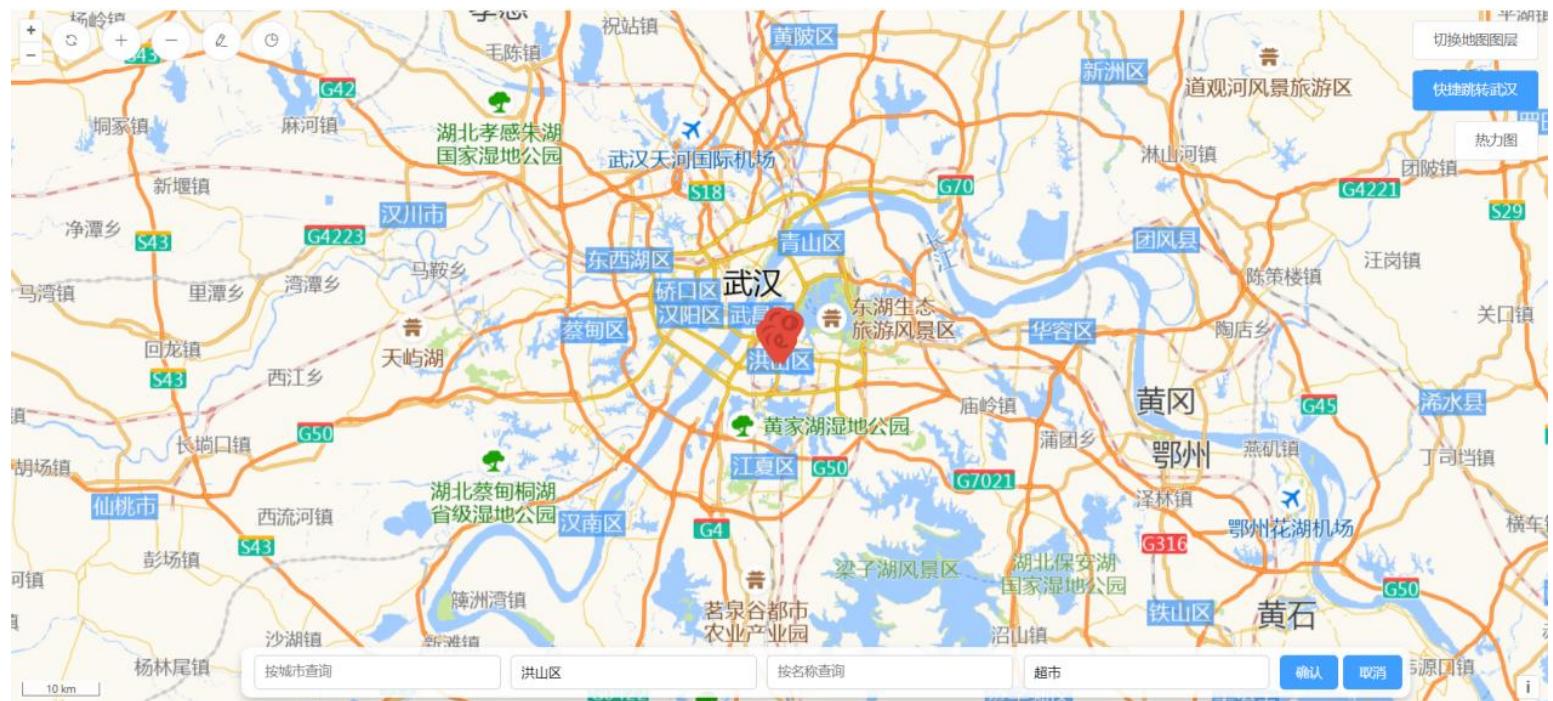
从左到右传递的查询条件分别为城市、区/县、地点名称、POI类型，和数据库字段相对应。  
点击取消按钮可以清空当前搜索框中填入的所有信息。



单条件查询（超市）：



多条件查询（洪山区、超市）：



### 9.3 生活服务设施信息编辑

#### 编辑地点按钮：

点击编辑按钮，左键点击地图地点图标，将进入编辑地点界面；若需要取消编辑操作，可以在点击编辑按钮后右键地图或者在左键点击地点图标进入编辑地点界面后点击取消。进入编辑地点数据后我们需要编辑地点的相关信息：省份、城市、区县、地点名称、地点地址、地点类型。

## 编辑地点数据

×

省份 湖北省

城市 武汉市

\* 区县 洪山区

\* 地点名称 test

\* 地点地址 tset

\* 地点类型 test

取消

确认修改

其中区县、地点名称、地点地址、地点类型为必填项，若为空，系统则会弹出警告信息。

## 编辑地点数据

请输入完整的地点信息

X

省份

湖北省

城市

武汉市

\* 区县

洪山区

\* 地点名称

test

\* 地点地址

\* 地点类型

test

取消

确认修改

在修改完整地点信息后点击确认修改，后端确认后，系统弹窗提示修改成功：



对比编辑前与编辑后的信息框信息，可以发现对地点数据的编辑成功了：

编辑前：



编辑后：



经过安全验证后，数据库中数据同步更新：

2397	湖北省	武汉市	汉阳区	武汉市汉阳区旭升电脑郭茨口二桥西村159号	维修站点	30.569465	114.2088
2398	湖北省	武汉市	硚口区	秀域智能健康(古四店)	古田四路和解放大道交美容美发店	30.589953	114.2118
2399	湖北省	武汉市	硚口区	兴盛图文广告(保利香槟国际)	保利香槟国际金座一层摄影冲印	30.590225	114.2225
2400	湖北省	武汉市	硚口区	痘医生(古田四路店)	古田四路和解放大道交生活服务场所	30.589907	114.2117
2410	湖北省	武汉市	洪山区	test	test111111	test	30.580497

## 9.4 新增生活服务设施

### 添加地点：

点击添加地点按钮，左键点击地图任意位置，将获取该点经纬度，并进入添加地点界面。若需要取消添加操作，我们可以在点击添加按钮后右键地图取消当前操作或者在左键点击地图进入添加弹窗后点击取消来取消当前操作。

进入添加地点数据后需要用户填写添加地点的相关信息：省份、城市、区县、地点名称、地点地址、地点类型，点击清空按钮则会清除所有信息，经纬度信息无法修改。

#### 添加地点数据

×

省份

城市

\* 区县

\* 地点名称

\* 地点地址

\* 地点类型

纬度

30.583182296165873

经度

114.3518231682884

其中区县、地点名称、地点地址、地点类型为必填项，若为空，系统则会弹出警告信息。

城市  \*

\* 区县

\* 地点名称

\* 地点地址

\* 地点类型

纬度

经度

清空 取消 确认添加

在输入完整地点信息后点击确认，在地图和数据库中添加地点。操作成功后，系统会弹窗添加成功同时在地图上更新相应地点图标。



经过安全验证后，数据库中数据同步更新：

ID	省份	城市	区县	设施名称	经度	纬度
2398	湖北省	武汉市	硚口区	秀域智能健康(古四店) 古田四路和解放大道交 美容美发店	30.589953	114.2118
2399	湖北省	武汉市	硚口区	兴盛图文广告(保利香槟国际金座一层摄影冲印)	30.590225	114.2225
2400	湖北省	武汉市	硚口区	痘医生(古田四路店) 古田四路和解放大道交 生活服务场所	30.589907	114.2117
2410	湖北省	武汉市	洪山区	test	tset	test

## 9.5 删 除生活服务设施

### 删除地点：

点击删除按钮，左键点击地图地点图标，将直接删除地点并弹窗删除成功；若需要取消删除操作，可以在点击删除按钮后右键地图来取消当前操作。



验证数据变化，数据库对应数据实时更新：

2396	湖北省	武汉市	硚口区	中艺养生会所(保利香槟古田五路18号保利香槟美容美发店)	30.590345	114.2221
2397	湖北省	武汉市	汉阳区	武汉市汉阳区旭升电脑郭茨口二桥西村159号 维修站点	30.569465	114.2088
2398	湖北省	武汉市	硚口区	秀域智能健康(古四店) 古田四路和解放大道交 美容美发店	30.589953	114.2118
2399	湖北省	武汉市	硚口区	兴盛图文广告(保利香槟国际金座一层摄影冲印)	30.590225	114.2225
2400	湖北省	武汉市	硚口区	痘医生(古田四路店) 古田四路和解放大道交 生活服务场所	30.589907	114.2117