# **北京文化旅游信息系统技术文档**

## **一、项目概述**

1.1项目背景

1.2项目目标

1.3应用场景

1.4系统功能概述

## **二、技术栈**

2.1后端技术栈

* Flask Web框架
* SQLAlchemy ORM
* SQLite数据库
* RESTful API设计

2.2前端技术栈

* Vue.js 3前端框架
* Vite构建工具
* Vue Router路由管理
* Element Plus UI组件库
* 高德地图API (@amap/amap-jsapi-loader)
* Three.js 3D可视化库
* Axios网络请求
* GSAP动画库

2.3开发环境与工具

* Node.js环境
* Python环境
* 版本控制工具
* IDE及开发工具

## **三、需求分析**

3.1用户需求分析

* 主要用户群体
* 用户行为分析
* 用户痛点分析

3.2功能需求分析

* 景点信息展示需求
* 历史地图展示需求
* 导航规划需求
* 旅游推荐需求
* 全景展示需求

3.3非功能性需求

* 性能需求
* 安全需求
* 可靠性需求
* 可用性需求

## **四、系统设计**

4.1架构设计

* 前后端分离架构
* 模块化设计
* 数据流设计

4.2数据库设计

* 景点信息表(attraction)
* 历史地图表(historical\_map)
* 地图兴趣点表(map\_point)
* 推荐表(recommendation)
* ER图
* 索引设计

4.3 API接口设计

* RESTful API规范
* 接口认证机制
* 错误处理机制
* 主要接口说明
* 景点信息接口
* 历史地图接口
* 导航规划接口
* 推荐生成接口

4.4前端设计

* 组件设计
* 路由设计
* 状态管理
* UI/UX设计原则
* 地图交互设计

## **五、核心功能实现**

5.1景点信息展示

* 数据加载流程
* 列表与详情切换
* 交互设计实现

5.2历史地图功能

* 历史地图加载与展示
* 历史地点标注实现
* 时空旅行功能实现

5.3导航规划功能

* 基于高德地图API的导航实现
* 路线规划算法
* 用户位置获取

5.4旅游推荐系统

* 推荐生成流程
* 推荐结果展示
* 推荐分享功能

5.5全景展示功能

* 全景接入方式
* 全景控制交互

## **六、开发过程**

6.1开发方法论

* 敏捷开发方法
* 迭代开发流程

6.2开发计划

* 阶段划分
* 里程碑设定
* 任务分配

6.3测试策略

* 单元测试
* 集成测试
* 用户测试

6.4部署策略

* 开发环境部署
* 测试环境部署
* 生产环境部署

## **七、系统性能优化**

7.1前端性能优化

* 资源加载优化
* 代码分割
* 缓存策略

7.2后端性能优化

* 数据库优化
* API响应优化
* 服务器配置优化

## **八、项目总结**

8.1系统特色与亮点

8.2技术难点与解决方案

8.3未来功能拓展

8.4经验教训总结

## **九、附录**

9.1开发环境配置指南

9.2 API文档

9.3数据字典

9.4 用户手册