第三章：技术实现  
项目采用react前端框架搭建项目结构，使用ant-design UI组件库，使用axios进行网络请求，引入echarts框架进行数据的展示，最后运用服务端框架Nginx进行前端项目的部署。

开发流程：项目采用react框架，**React**是一个用于构建用户界面的 JavaScript 库.

他拥有以下特点1.声明式设计 −React采用声明范式，可以轻松描述应用（自动dom操作）。2.高效 −React通过对DOM的模拟(虚拟dom)，最大限度地减少与DOM的交互。

3.灵活 −React可以与已知的库或框架很好地配合。

4.JSX − JSX 是 JavaScript 语法的扩展。

5.组件 − 通过 React 构建组件，使得代码更加容易得到复用，能够很好的应用在大项目的开发中。

6.单向响应的数据流 − React 实现了单向响应的数据流，从而减少了重复代码，这也是它为什么比传统数据绑定更简单。

为了提高用户体验及开发效率综合考虑决定用react作为项目的前端结构搭建。

Ant Design是阿里蚂蚁金服团队基于[React](https://so.csdn.net/so/search?q=React&spm=1001.2101.3001.7020)开发的UI组件库，提炼自企业级中后台产品的交互语言和视觉风格，为用户带来极佳的使用体验。

采用Axios进行网络请求，Axios 是一个基于 promise的异步 ajax 请求库.

1.可以转换请求数据和响应数据，并对响应回来的内容自动转换成 JSON类型的数据；

2.拦截请求和响应（就是前端发送请求前，可以设置自动拦截请求，相当于给请求加条件）；

3.Axios 是一个基于 promise 的异步 ajax 请求库，支持promise所有的API ；

4.axios.all(promises): 批量发送多个（异步）请求；

5.axios在览器端/node 端都可以使用，在node环境使用http对象发送ajax请求；在浏览器端使用XMLHttpRequests对象发送ajax请求（XMLHttpRequest-来实现发送和接收HTTP请求与响应信息）。即创建 http 请求，向后端发送请求并接收 返回的数据；

6.安全性更高，客户端支持防御 XSRF（跨站请求伪造），就是让你的每个请求都带一个从cookie中拿到的key, 根据浏览器同源策略，假冒的网站是拿不到你cookie中得key的，这样，后台就可以轻松辨别出这个请求是否是用户在假冒网站上的误导输入，从而采取正确的策略；

Aixos可在请求数据时提供良好的请求环境，同时也保证了数据的安全性，降低了数据泄露的风险，异步请求也进一步提升了用户体验。

ECharts，缩写来自 Enterprise Charts，商业级数据图表，是百度的一个开源的数据可视化工具，一个纯 [Javascript](https://www.w3cschool.cn/javascript/" \t "_blank) 的图表库，能够在 PC 端和移动设备上流畅运行，兼容当前绝大部分浏览器（IE6/7/8/9/10/11，chrome，firefox，Safari等），底层依赖轻量级的 Canvas 库 ZRender，ECharts 提供直观，生动，可交互，可高度个性化定制的数据可视化图表。创新的拖拽重计算、数据视图、值域漫游等特性大大增强了用户体验，赋予了用户对数据进行挖掘、整合的能力。

Echarts良好的数据展示能力，能够清晰的将数据通过各种方式展示，如项目中用到的柱状图，折线图，词云及水滴等各种图标类型，提升了与用户的交互性，且兼容pc与移动端及各种浏览器，同时也保证了项目在各个环境下正常运行与用户的使用门槛。

前端项目部署在阿里云服务器，采用Nginx服务端框架.

Nginx 是高性能的 HTTP 和反向代理的**服务器**，处理[高并发](https://so.csdn.net/so/search?q=%E9%AB%98%E5%B9%B6%E5%8F%91&spm=1001.2101.3001.7020)能力是十分强大的，能经受高负载的考验,有报告表明能支持高达 50,000 个并发连接数。

Nginx的良好性能提升了项目运行的可靠性，优秀的高并发能力也是作为部署前端项目的不二之选。

第四章：

项目难点：

[React](https://so.csdn.net/so/search?q=React&spm=1001.2101.3001.7020)**的组件化**

组件化是React的核心思想，前面我们封装的App本身就是一个组件：

组件化提供了一种抽象，让我们可以开发出一个个独立可复用的小组件来构造我们的应用。

任何的应用都会被抽象成一颗组件树。

在项目进行组件化拆分时决定以每种类型的图标作为组件进行拆分，但组件间通信的问题又显现了出来，采用react中的props属性进行父子组件中的通信成为了最终选择，

完成请求后的数据处理：

数据处理的方式决定着数据展示的难易，所以请求完成后的数据处理问题成为了项目的一大重点及难点。在请求不同图标的数据时，需按照不同图标需要的数据类型及格式进行处理后存储，最后通过组件化中的数据传递思想完成图标数据的处理、

Echarts图表渲染图表渲染大数据量导致卡顿加载时间慢等问题的解决方案

在进行echarts图表渲染时，大量的数据需要进行处理，会导致以下几个问题

1.首次加载时间过慢，友好性和体验性极差；

2.筛选条件，如默认1个小时，筛选24小时以上的数据时，加载数据过慢；

3.折线图拖动时，dataZoom组件使用时的卡顿；

4.浏览器大小变化时，windows监听myChart.resize()导致的重新加载，卡顿；

解决思路：

1.流加载/分页懒加载方式以及post/get等网络请求方式；

2.对Echarts自身加载大量属性的使用和掌握。

3.API接口的读取速度：mySql读取/Redis缓存读取；

4.网络请求获取数据和处理数据的方法；

通过以上几点的思路，逐一实现，大幅度提高了图标渲染的速度，提升了用户体验,同时降低了项目对性能的消耗.