前后端分工：

前端开发人员专注于 UI 交互和展示逻辑的设计，后端开发人员专注于业务逻辑和数

据存储等，前后端通过接口进行数据的交换。

前后端可以并行开发，提升开发效率：在商定好接口文档后，前后端系统可以同时进行开发和 测试。如果后端系统暂时无法提供接口的数据，可以利用模拟数据提供接口，使得前端开发的工作不会因此停滞。

前后端系统独立部署，提升性能 ：与传统的 Web 开发系统不同，前后端系统分别部署在单独的服务器上，前端系统负责渲染页面和前端的路由，将 数据加载进入到页面的模板中 ，后端系统只需要提供接口和数据，从而降低的服务器的负荷，极大的提高了系统的响应速度。

MVVM模式

随着前端技术的发展，在 Web 应用开发中，很多的业 务逻辑和数据开始转向前端处理。MVVC（Model-ViewViewModel）模式被引入前端开发领域 [5]。MVVM 模式是对 传统的 MVC（Model-View-Controller）模式的进一步的发展， 目的都是为实现模型层和视图层的分离，具备有低耦合、高 复用性、独立开发和方便测试的优点 [2]。 MVVM模式包括三个部分，模型层（Model） 存放着程序所需要的数据，视图层（View）是系统的 UI 界 面部分，视图模型层（ViewModel）是 MVVM 模式的核心 部分，实现了视图层与模型层的解耦 [6]。视图模型层将原本图层的业务逻辑进行封装，负责视图层的显示逻辑，并实现模型层与视图层的双向数据绑定，起到上启下的作用。当模型层的数据发生变化时，视图模型层能同步更新到视图层；当视图层发生改变时，视图模型层能及时做出响应，进行相应的处理并同步到模型层。

技术需求方面可从

1) 稳定性

1. 安全性 系统具备数据自动备份、手动备份和故障恢复能力，文件的访问限制、用户信息的加密存储，API 的访问控制;

3) B /S 架构 用户无需安装客户端/App，通过浏览器即可使用;

4) 全平台可用 可以在 PC、智能手机和平板电 脑上正常使用;

5) 兼容性 兼容主流浏览器( Chrome、Firefox、 Safari、Edge、360 浏览器等) ．

这些要素考虑

前后端分离开发框架

Vue：

Vue 是一个关注于视图层，用于构建用户界面的渐进式轻量级 JavaScript 框架。Vue 框架采用自底向上增量开发的设计，通过与 Vue 生态系统的支持库和第三方库的整合，可以为复杂的单页面应用SAP）提供驱动[7]。

相对于其他框架，Vue 容易使用，学习难度低，开发效率高。 Vue 虽然没有完全遵循 MVVM 模式，但是 Vue 设计收到 MVVM 模式的启发。Vue 最核心的功能包括组件化的系 统和响应式的双向数据绑定 [2]。组件是可复用的 Vue 实例， 是 Vue 最强大的功能之一，可以扩展 HTML 元素，封装可重用的代码。Vue 的组件化系统可以通过使用小型的、独立可复用的组件来构建一个大型复杂的Web 应用程序。

响应式的双向数据绑定是 Vue 的另一个最独特的功能，数据模型是普通的 JavaScript 对象，当它发生改变时，对应的视图层的数据会自动改变，反过来也是一样的，这使得状态管理变得非常简单。Vue 响应式系统原理如图 3 所示，当一个普通的 JavaScript 对象传入 Vue 实例的 data 配置项中，Vue 将遍历这个对象的所有属性，使用 Object. defineProperty() 方法将这些属性添加到 Vue 的实例对象中，并添加相应的 getter 和 setter 方法。通过 getter 和 setter 方法，Vue 实例可以追踪到属性被访问或发生改变。每一个组件都对应着一个 watcher 实例，它会把在组件渲染过程中接触到属性记录为依赖。当依赖属性的 setter 方法被调用时，会通知 watcher 实例，使得相关联的组件重新渲染。

除了 Vue 框架本身提供的核心功能外，Vue 官方还提供脚手架工具 Vue-cli、路由管理 Vue Router、状态管理库 Vuex和Vue的服务渲染等优秀的插件扩展Vue框架的功能。

在 Vue 的开发生态圈中，Element UI 是一款由饿了么前端团队推出的基于 Vue2.0 的的桌面端的 UI 组件库，是目前和VUE 结合做项目开发的主流 UI 框架。

Spring Boot：

Spring Boot 是目前在 Java 中比较热门的一个微服务框架 [8][9]。

在 2013 年，为了简化基于 Spring 框架的应用程序的开发过程，Pivotal 团队推出的基于 Spring 的开源开发框架Spring Boot[11]。Spring Boot框架具有以下的特点：

（1）配置更简单，基于“约定优于配置”，极大减少了配置的工作量，最终的目标是实现“零配置”。

（2）使编码简单。通过 Spring Boot 提供的各种注解，可以实现丰富的功能，极大减少了代码的编写量。绝对不需要生成代码，也不需要进行 XML 配置。

（3）自动装配，通过 @EnableAutoConfiguration 注解，Spring Boot 启动后会扫描类路径下的所有类，合理地推断出应用程序所需的 bean，实现 bean 类的自动装配。

（4）内嵌了各种 Servlet 容器，如 Tomcat，可以直接通过 jar 包的方式快速部署和运行，而无需使用外部的服务器应用。

（5）开箱即用的工具包，Spring Boot 提供了很多开箱即用的工具包，只需将相应的 starter 配置项加入到项目的 pom 文件中，就能很方便地使用对应的功能，如 springboot-starter。

（6）便于监控程序的状况，Spring Boot 提供了便于查看当前项目的运行情况的各种度量和性能指标的功能。

Spring Boot 通过践行“约定优于配置”的理念，使开发者不再需要像之前 Spring 那样定义大量的配置，只需要非常少的几个配置，就可以迅速方便的搭建起来一个应用或者微服务，做到开箱即用（out of box），从而使开发者从繁琐的配置工作中解脱出来，专注于业务逻辑的开发。结合 Spring Cloud 和 Docker 技术，使得在云中部署和管理微服务变得更加方便 [12]。

案例

学院固定资产系统设计与实现（采用基于 Vue+Element UI+spring boot 的前后端分离技术）

第一步、需求分析：。。。。。哪些功能，及需要注意的

第二步、系统总体设计：系统采用前后分离的架构开发，其中前端系统采用 vue+Element UI 组件实现，后端系统使用 Spring Boot 框架开发，前后端之间通过 RESTful 架构的接口进行通信。RESTful 架构是遵循统一 REST 风格接口原则的Web 服务，使用标准的 HTTP 方法如 GET，PUT、POST 和 DELETE 对 URL 资源进行增删改查的操作

后端系统开发：

系统后端采用 Spring Boot 框架开发，在 pom.xml 文件中添加 spring-boot-starter-web 的启动器配置项，使用 @ SpringBootApplication 注解来标注主程序类，系统使用多层架构：

（1）实体层（Entity 层），是业务领域的最基本的一层，存放的是实体类（entity 类），与数据库的表一致，作用是在不同层之间传递数据。

（2）数据访问层（Dao 层），系统通过 Dao 层实现与数据库的交互，系统开发使用 Spring Data JPA，这是 Spring 的一个基于 ORM 的一个 JPA（Java Persistence API）应用框架，底层基于 Hibernate 实现。通过 Spring Data JPA，可以使用很简单的代码就能实现持久层的服务，它提供了包括增删改查等在内的常用功能，并且非常容易扩展。

（3）业务逻辑层（Service 层），位于数据访问层和控制层之间，封装了主要的业务逻辑和对数据访问层的操作。

（4）控制层（Controller 层），使用 @RestController 注解，在控制层中定义了 RESTful API 的 Web 服务，负责接受前端发来的请求，调用 Service 层的服务，将 service 层返回的数据传回到前端。通过 @GetMapping、@PostMapping 等注解简化常用的 HTTP 请求的映射。

关键技术

（1）CROS 跨域访问。

当前后端系统分别部署在不同服务器上，通过 API 接口 通信时，会面临跨域的问题，浏览器会组织Ajax的访问请求，CORS（跨域资源共享）是跨域的的一种解决方案，可以通过设置 CorsFilter 过滤器解决跨域问题。

前端系统开发：

系统前端部分采用基于 vue+element UI 框架实现，首先通过 npm 安装 vue-cli 脚手架，通过脚手架可以快速生成Vue 项目基础的架构。然后在项目中引入 Vue Router 前端路由器和 element UI 组件库，Vue Router 是 Vue 的官方路由器，通过配置 Vue Router，在前端根据不同的 URL 渲染不用的

Vue 组件展示，而无需后台服务器的参与，因此让 Vue.js 构建单页应用程序（SAP）变得更加容易。

在系统中通过安装和配置 axios 库来实现 AJAX 请求，axios 是基于 promise 的 HTTP 网络请求库。因为采用前后端分离模式开发系统，用户的认证通过 web token 完成。因此在后端提供的 api 接口中，除了登录接口之外，每个接口都需要 token 的验证。当用户第一次登陆成功后，web 端会接收到服务器返回的 token，保存到浏览器的 sessionStorage 中。在 axios 上添加一个全局的请求拦截器，保证除了登陆以外的 api 请求，都会将 token 添加到 HTTP 的头部的Authorization 对象中。当 token 丢失或失效的时候，设置路由守卫，返回登陆页重新登陆。经过多个浏览器平台的测试，

系统运行正常。