技术栈选择（前端）

* 前端开发工具：Visual Code
* 原型设计工具：墨刀（APP设计、浏览器页面设计）
* 前端框架：react + react-router + react-redux + superagent
* 移动端框架： react-native + react-native-router-flux + react-redux + superagent
* 移动端：跨平台ios、android
* 前端组件库+CSS框架：Ant Design
* 移动端组件库： Ant Mobile
* 前端CSS预处理器：Less
* 前端工程打包工具：WebPack
* 前端模块化：ES6
* 前端基础技术：Html5\CSS3\ES6
* 前端开发理念：响应式布局、移动优先

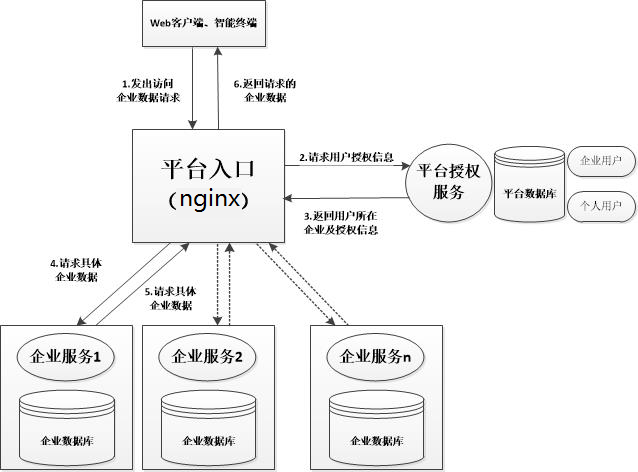
技术栈选择（后端）

* 服务器端开发工具：Eclipse
* 服务器端打包组件管理工具：Maven
* 服务器端认证：Web Token
* 服务器端微服务框架： docker + spring boot + spring cloud + Eureka
* 服务器端持久化框架： Spring Data JPA
* 服务器端单元测试：Junit
* 容器管理：docker registry + docker-compose + rancher
* 服务器端服务注册：Eureka
* 服务器端消息队列：RabbitMQ
* 服务器端定时任务：Quatz
* 服务器端缓存：Reddis
* 服务器端日志服务： log4j
* 数据库：MySQL+MongoDB
* 负载均衡：Rancher

软件研发过程

* 代码本地管理：GitLab（前端）+ svn (后端）
* 代码审查：GitLab Merge Request
* 后端三方库管理： Nexus（Maven私服）
* 持续集成：Jenkins + svn + gitlab
* Docker镜像仓库：docker registry
* 开发环境：运行构建的docker镜像，并打tag推送至测试环境docker镜像仓库
* 测试环境：运行测试环境的最新版本docker镜像，发版本时将docker镜像推送至生产环境docker镜像仓库
* 生产环境：从生产环境docker镜像仓库拉取特定版本docker镜像并运行
* 版本发布管理工具1：管理docker容器和栈， 将开发环境镜像推送到测试环境，将测试环境镜像推送到生产环境。
* 版本发布管理工具2：管理生产环境服务器信息，将docker镜像部署到具体企业的服务器
* 接口文档化：Swagger
* 协同文档：MarkupDown + gitlab
* 容器技术：Docker、Rancher
* 通信协议：Restful（JSON、命名规范）

Nomes系统总体架构



Nomes企业服务架构

