如何做系统设计及技术选型概谈

（个人观点）

主讲人：姜建武

2021.8.03 · 桂林

目录
CONTENTS

1 系统设计的要素
2 系统设计的流程
3 系统技术选型
4 系统设计案例

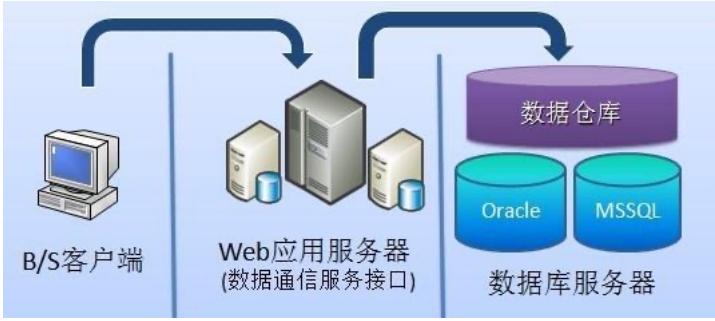
一、系统设计的要素

系统分析

- 目标、需求、任务、现状、可行性分析、技术、人力、资源、资金投入预算、技术方案等
- **逻辑模型**

系统设计

- UI设计、数据库设计、系统架构设计、模块及模块间通讯方式设计等
- **物理模型**



一、系统设计的要素

系统设计时需要搞清楚哪些问题？

- 系统的目标是什么？想要解决什么问题？现有基础如何？（确定初步需求）
- 系统面向的用户群体是那些？各用户群体对系统功能有什么要求？（确定详细需求）
- 系统分为多少模块？模块间的关联关系如何？模块间如何通信？（确定系统模块）
- 系统的应用场景是哪些？根据场景选择何种架构（B/S、C/S）？（确定系统架构）
- 选择哪几种技术？选择哪几种数据库?（确定技术选型）
- 功能模块如何设计？（确定详细功能）
- 数据库如何设计？（根据详细功能确定数据库）
- UI如何设计？（根据功能和数据库确定UI界面）
- 其它。

一、系统设计的要素

系统设计的要素

- **系统概述**
  - 发包、接包单位概况、建设背景、建设目标、主要功能
- **需求分析**
  - 现状分析、业务流程分析、用户群分析、用户需求分析、功能分析、资源分析
- **性能指标**
  - 鲁棒性、响应时间、数据量、安全级别等
- **总体设计**
  - 总体架构、模块设计、总体逻辑、总体流程、技术选型、系统部署、技术路线、通讯方式设计、系统更新方案

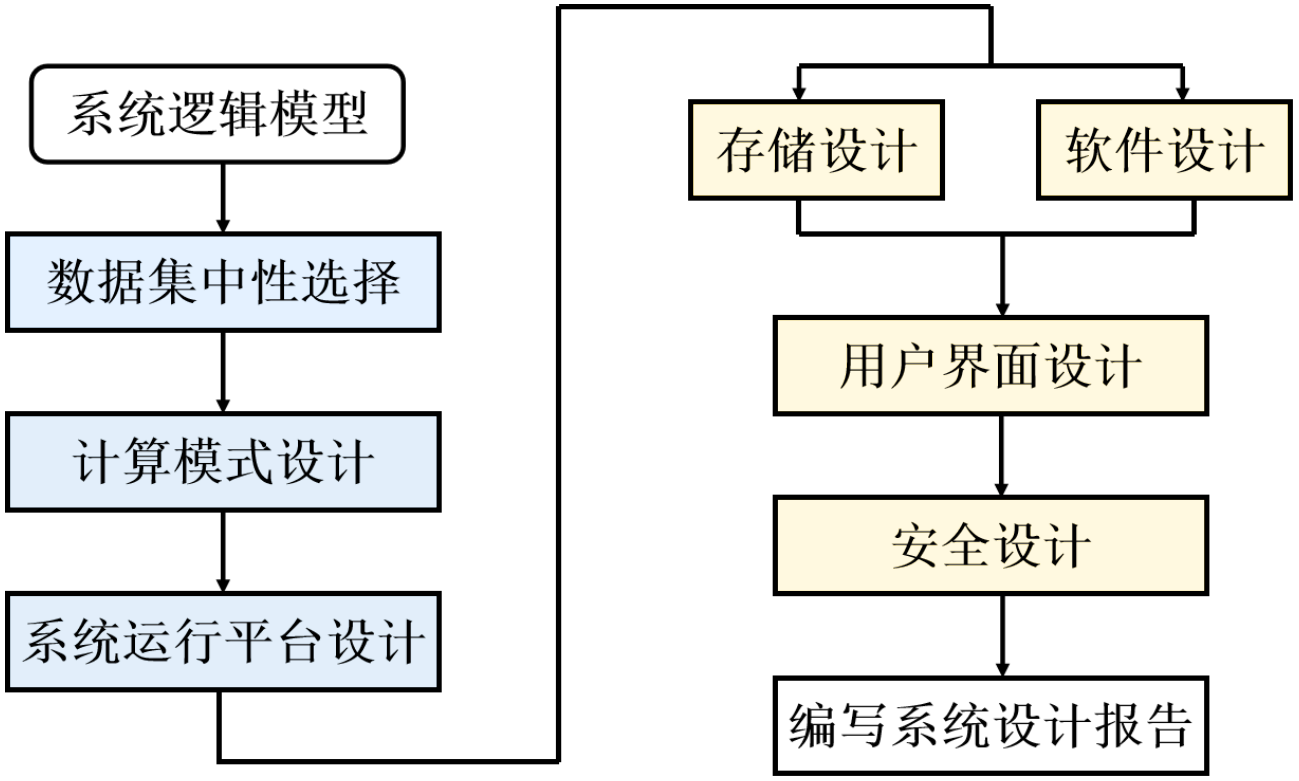
一、系统设计的要素

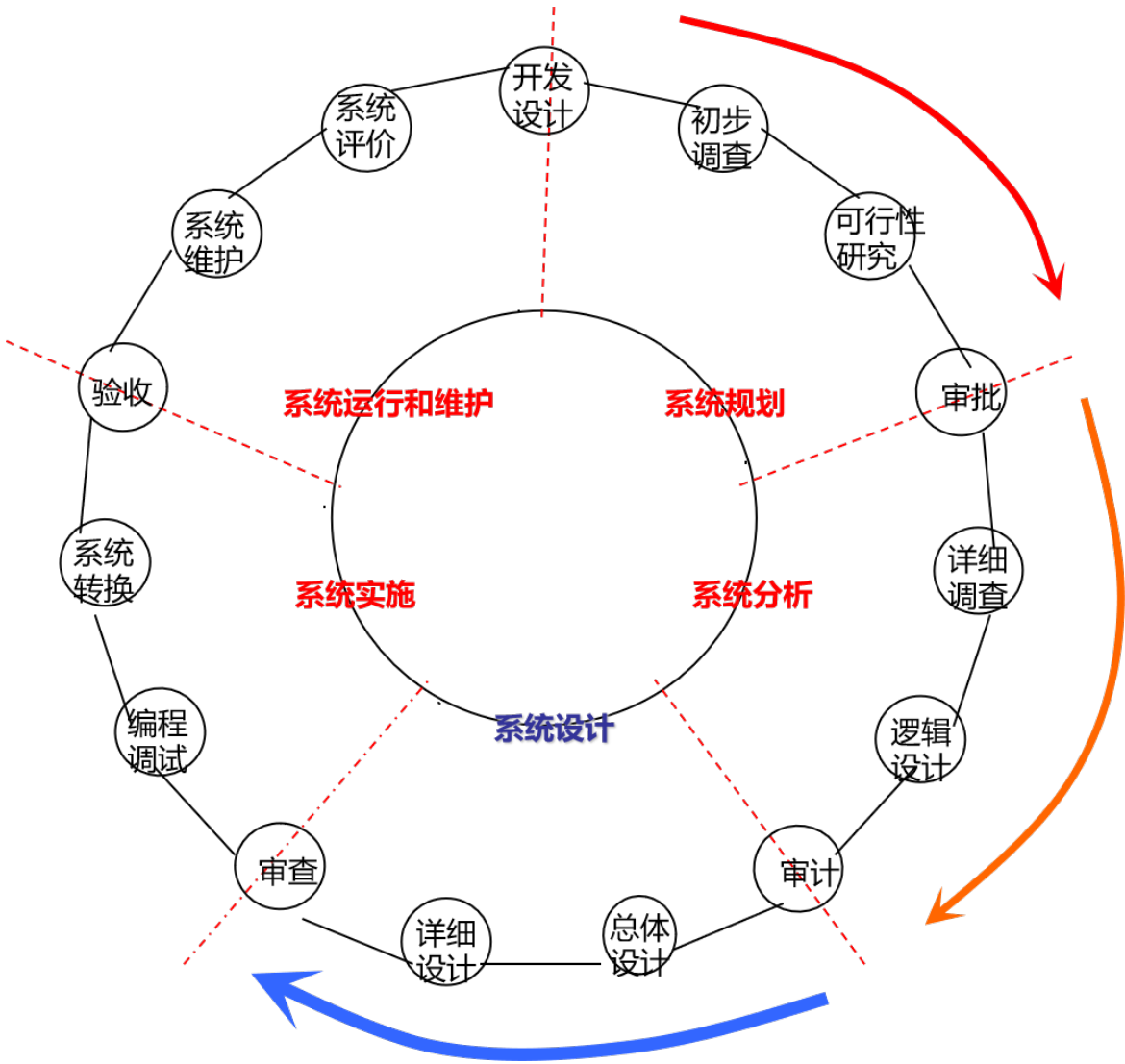
系统设计的要素

- **数据库设计**
  - 数据表详细设计、关联关系设计、视图设计、
  - 索引设计、优化设计
- **功能设计**
  - 子模块详细设计、功能详细设计
- **UI设计**
  - 配色、图标、风格、交互、展示
- **系统概算**
  - 人力概算、工期概算、资金概算

二、系统设计的流程

标准系统设计任务与流程





二、系统设计的流程

（1）系统概述

- 发包、接包单位概况、建设背景、建设目标、主要功能

核心要点：

系统名称： XXX信息管理系统

发包方：XXX单位

承包方：XXX单位（可介绍下单位情况，尤其是对完成该系统的优势之处）

建设背景：阐述清楚为何建设该系统、建设该系统能够为发包方带来哪些便利等

建设目标：阐述清楚建成后在管理、宣传、效率、工作上等方面能够实现什么提升、突破和优化

主要功能：围绕建设目标和原始需求初步拟定主要系统功能

二、系统设计的流程

（2）需求分析

- 现状分析、业务流程分析、用户群分析、用户需求分析、功能分析、资源分析

核心要点：

系统现状：弄清发包方在数据、信息化基础设施、信息化系统、员工信息化水平、业务信息化普及度、尚欠缺的条件等方面的现状

总体业务流程：该系统面向的具体业务是什么，流程是什么样的

用户群体：系统主要面向那些人使用

用户需求：不同人员使用该系统有什么样的需求

功能分析：根据不同用户需求初步分析系统功能

系统建设所需资源分析：系统建设需要哪些数据、资源、配合等

二、系统设计的流程

（3）性能指标

- 鲁棒性、响应时间、数据量、安全级别

核心要点：

鲁棒性：即系统的稳定性，需要从系统使用频率、吞吐量等各方面综合分析，鲁棒性影响技术选型和硬件配置；系统日志、系统监控等

响应时间：系统的效率，设定后需要从技术和硬件两方面进行优化，比如缓存、索引、数据虚拟化、内存常驻等技术和方案

数据量：综合考虑数据的增长量、数据的操作方式（查询为主还是写入为主）等综合确定数据库引擎

安全级别：确定数据传输加密方案、用户、数据分级权限、系统整体安全监控、内外网隔离、时间段服务等

二、系统设计的流程

（4）总体设计

- 总体架构、模块设计、模块通讯、总体逻辑、总体流程、技术选型、技术路线、更新方案、系统部署

核心要点：

总体架构：根据（2）的结论，确定B/S、C/S、APP、WebApp、H5、分布式等系统架构；确定系统层次即各层次包含的内容；确定数据处理方式（胖、瘦客户端确定）；

模块设计：根据（2）确定系统模块即模块内容；模块内高内聚，模块间松耦合；确定模块间通讯方式（WebService、OpenAPI等）；

总体逻辑：根据（2）确定各模块内部及模块间的逻辑关系

总体流程：根据（2）确定系统总体数据流图、业务流图等

技术选型：根据（4）确定最终技术架构，具体见第三节技术选型，包括前端、后端、插件等

二、系统设计的流程

（4）总体设计

- 总体架构、模块设计、模块通讯、总体逻辑、总体流程、技术选型、技术路线、更新方案、系统部署

核心要点：

技术路线：绘制整体技术路线图

更新方案：确定系统更新方式，全量更新还是增量更新、更新渠道、更新频率、回滚方案、数据备份方案等

系统部署：根据前几方面内容确定系统部署方案（内外网分离、通讯方案、分布式部署方案、上云方案、自建服务器方案等）

二、系统设计的流程

（5）功能设计

- 子模块详细设计、功能详细设计

核心要点：

子模块详细设计：确定子模块包含哪些功能、各功能间在模块内如何起作用、在模块间如何参与处理等

功能详细设计：对子模块包含的功能进行详细设计、明确需要哪些数据、数据格式如何、数据如何处理、有哪些相关数据参与计算、需要什么权限、返回什么信息

二、系统设计的流程

（6）数据库设计

- 数据表详细设计、关联关系设计、视图设计、索引设计、优化设计

核心要点：

数据表详细设计：根据（5），按照模块化和面向对象的思想设计数据库，确定数据库包含的数据表数量和类别、各数据表包含的字段、字段类型、字段大小、是否为空、是否主键、是否不可重复、是否外键等信息

关联关系设计：确定表间如何关联、通过何种字段关联

视图设计：对于涉及到多个表的普遍操作如何设计视图

索引设计：为表建立索引

优化设计：根据数据量、操作频率和类型等确定数据表切分、范式方案

二、系统设计的流程

（7）UI设计

- 配色、图标、风格、交互、展示

核心要点：

配色方案：根据发包方要求和常规做法确定配色方案，系统设计时要围绕配色方案设计不可随意更改，确定主色、辅助色、中性色等。（资源：IView配色）

图标方案：确定使用图片还是SVG文字图标。（资源：IconFont 、FontAweSome）

系统风格：扁平化、原质化等。（资源：设计风格）

交互设计：确定用户输入方式、登录方式、确认方式、提示方式等

展示设计：瀑布流、卡片式等

常用原型设计工具：Axure、Mockplus、蓝湖、墨刀、更多

二、系统设计的流程

（8）系统概算

- 人力概算、工期概算、资金概算

核心要点：

人力概算：确定投入人员数量

工期概算：确定需求调研、系统设计、系统开发、系统优化、试运行、上线、培训等时间

资金概算：根据投入的人力、时间确定资金，根据利润、成本、人员薪资、项目提成、发包方预算等确定最终资金概算

三、技术选型

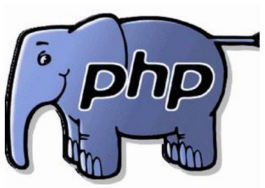
常见前端开发技术

- 基础：HTML（4、5）【元素】、CSS（2、3）【样式】、JavaScript【交互】
- 常用前端框架：BootStrap、IView、LayUI
- 常见JS库：JQuery、Vue.JS、TypeScript、React、Angular
- 常用功能：Ajax、Json、JsonP、XML

三、技术选型

常见后端开发技术

编程语言	用途	优点	缺点
PHP	网页	容易、灵活	大型网站性能较差
Java（JSP）	网页、桌面、APP、DL	安全性高、框架多	学习成本较高
C#（ASP、ASP）	网页、桌面、APP、DL	安全性高、IDE好用	桌面程序跨终端支持不好、多平台兼容性不好
Python	网页、桌面（QT）、DL	容易、第三方库、适合深度学习应用	缩进很容易造成混乱、编码容易出问题
JavaScript(Node.JS)	网页	容易、前后端一体	安全性和大型网站性能差



三、技术选型

常见网页服务器

服务器	渲染支持	功能
Apache	静态页面、PHP	Apache是世界上应用最多的WEB服务器，优势主要在于源代码开放、有一支开放的开发队伍、支持跨平台应用以及其可移植性等。Apache的模块支持非常丰富，虽在速度和性能上不及其他轻量级WEB服务器，是属于重量级产品，所消耗的内存也比其他WEB服务器要高。
Tomcat	静态页面、JSP	Tomcat是一个开放源代码、运行servlet和JSP Web应用软件的基于Java的web应用软件容器，Tomcat server是根据servlet和jsp规范执行的，因此也可以说Tomcat server实行了Apache-Jakarta规范，且比绝大多数商用应用软件服务器要好，但是，tomcat对静态文件和高并发的处理比较弱。
Microsoft IIS	静态页面、ASP、ASP.NET、PHP	IIS是一种WEB服务组件，允许在公共Intranet或Internet上发布信息的WEB服务器，其中包括WEB服务器、FTP服务器、NNTP服务器和SMTP服务器，分别用于网页浏览、文件传输、新闻服务和邮件发送等方面，它使得在网络上发布信息成了一件很容易的事情。它提供ISAPI作为扩展WEB服务器功能的编程接口，同时还提供了Internet，可以实现对数据库的查询和更新。
Nginx	静态页面、PHP	Nginx是一款高性能的HTTP和反向代理服务器，能够选择高效的epoll、kqueue、eventport最为网络I/O模型，在高连接并发的情况下，能够支持高达5万个并发连接数的响应，而内存、CPU等系统资源消耗却非常低，运行非常稳定。
Node.JS	静态页面、JavaScript	Node.js 是一个基于Chrome JavaScript 运行时建立的一个平台。 Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎，V8引擎执行Javascript的速度非常快，性能非常好。

更多：https://blog.csdn.net/l_liangkk/article/details/84064321

三、技术选型

常见关系型数据库引擎

数据库	级别	定位	应用场景	选择注意
MySQL	中小型	开源、多平台、关系型数据库，目前使用最广泛、流行度最高的的开源数据库。	默认的innodb存储引擎，支持高并发，简单的绝大部分OLTP（联机事务处理）场景；Tokudb存储引擎，使用高并发insert的场景；Inforbright存储引擎，可以进行列压缩和OLAP（联机事务分析）统计查询场景；	使用MySQL进行OLTP业务时，需要注意数据量级，如果数据量级过大，需要进行水平拆分；
SQL Server	大型	商业、Windows平台、关系型数据库	大多数OLTP场景	绝大多数工作都是通过图形界面完成；使用SQL Server满足OLTP业务，会有比较好的效果，但对于大数据量的OLAP业务，最好还是选用专门的OLAP架构，不要在同一个SQL Server实例上混用OLTP和OLAP业务；
Oracle	大型	商业、多平台、关系型数据库	绝大多数OLTP场景，部分OLAP	Oracle从架构到运维，可以说是最难的数据库，学习和使用难度较高。
PostgreSQL	中大型	开源、多平台、关系型数据库，功能最强大的开源数据库。	绝大多数OLTP场景，部分OLAP，适合目前互联网需要的一些信息，比如地理位置信息处理	Postgresql的架构、使用难度、功能性介于Oracle数据库和MySQL数据库之间，Postgresql目前还没有比较主流和好用的监控平台
Access	小型	商业，小型本地数据库	本地小数据量存储	只能本地使用、数据量有限

更多：https://blog.csdn.net/gengkui9897/article/details/89294936

三、技术选型

常见非关系型数据库引擎

数据库种类	代表
列族数据库	BigTable、HBase、Cassandra、Amazon SimpleDB、HadoopDB
键值数据库	Redis、SimpleDB、Scalaris、Memcached
文档数据库	MongoDB、CouchDB、Perservere、Terrastore、RavenDB
图数据库	Neo4J、InfoGrid、OrientDB、GraphDB
内存数据网格	Hazelcast、Oracle Coherence、Terracotta BigMemorry、GemFire、Infinispan、GridGain、GigaSpaces

更多：https://zhuanlan.zhihu.com/p/109517355

三、技术选型

常见GIS开发引擎

GIS引擎	维度	适用场景	特点
ArcGIS Engine ArcGIS Objects	二维、三维	桌面、Web、APP	开发文档完善、案例多、开发简单；收费、国外产品
MapGIS SDK	二维、三维	桌面、Web、APP	开发文档较完善、有官方案例；兼容性较差、收费、国内产品
SurperMap	二维、三维	桌面、Web、APP	开发文档较完善、有官方案例；兼容性较差、收费、国内产品
Three.JS	二维、三维	Web、APP	基于WebGL，适用于浏览器类应用开发；难度较高
Cesium	二维、三维	Web、APP	基于WebGL，适用于浏览器类应用开发；案例少

三、技术选型

常见组合方式

- LAMP：Linux+Apache+MySQL+PHP 快速网页开发，全开源
- LNMP：Linux+Nginx+MySQL+PHP 快速网页开发，全开源
- TJJO ：Tomcat + Java + JSP + Oracle 中大型网站开发，性能高、稳定
- ICAS ：IIS + C# + ASP、ASP.NET + Sql Server 中大型网站开发，性能高、稳定

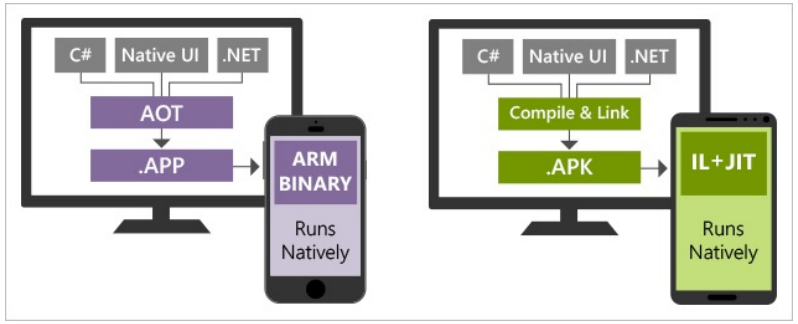
三、技术选型

APP及WebApp开发

- APP开发：Java或者C#
- WebAPP：Html+CSS+JavaScript+API

常用工具： Android Studio　Xamarin　DCLOUD 为开发者而生　uni-app 开发一次，多端覆盖

Android架构



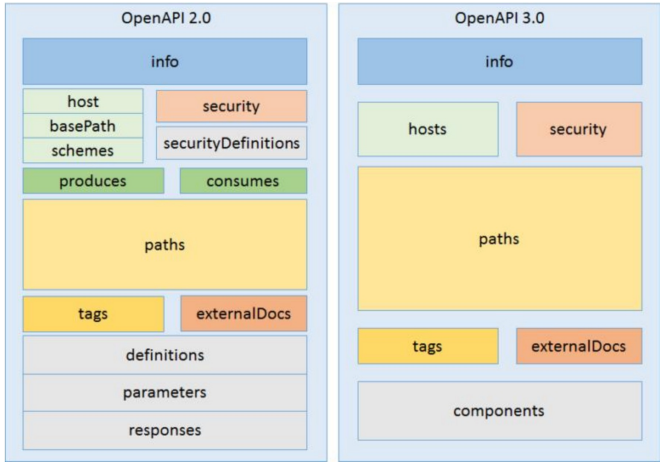
C#开发APP架构

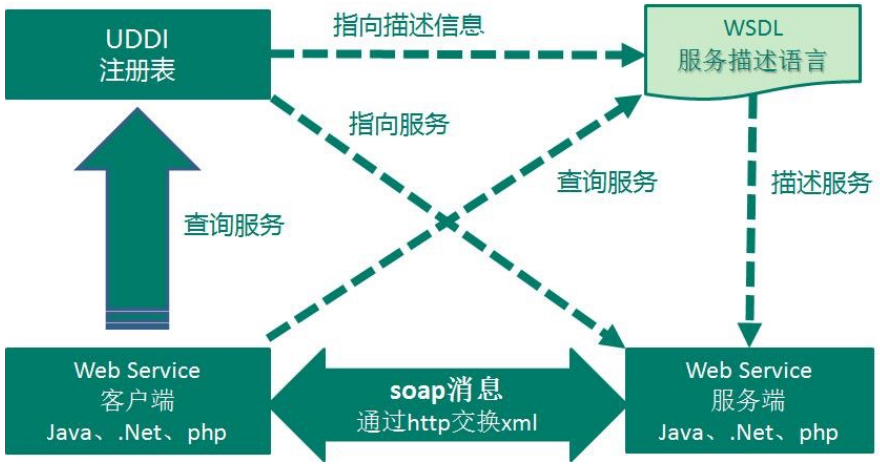
Uni-APP

三、技术选型

通讯及中间件

- API： API（Application Programming Interface，应用程序接口）是一些预先定义的接口（如函数、HTTP接口），或指软件系统不同组成部分衔接的约定。 用来提供应用程序与开发人员基于某软件或硬件得以访问的一组例程，而又无需访问源码，或理解内部工作机制的细节。
- WebService： Web Service是一个平台独立的，低耦合的，自包含的、基于可编程的web的应用程序，可使用开放的XML（标准通用标记语言下的一个子集）标准来描述、发布、发现、协调和配置这些应用程序，用于开发分布式的交互操作的应用程序。





三、技术选型

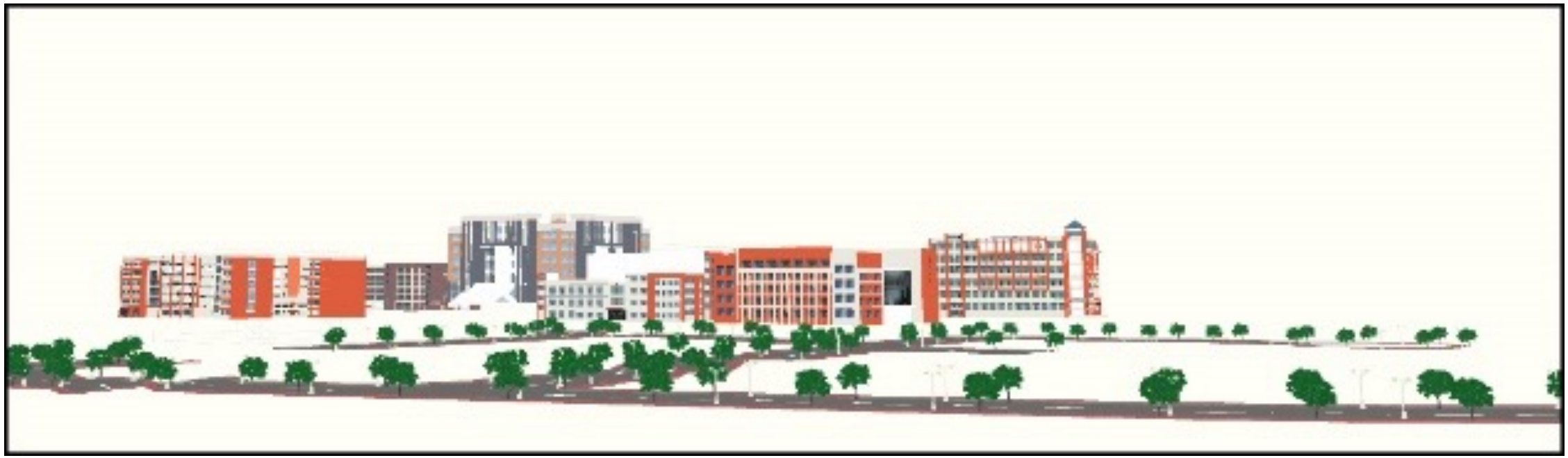
选型建议

- 应用场景：
  - 集中控制、满足多端融合：B/S
  - 更新主要发生在服务端，客户端需要较高稳定性且需要处理一定任务：C/S
  - 需要端使用、同步更新且客户端处理任务不多：WebAPP
- 小型网站：建议选择LAMP或者LNMP方案+中间件
- 中大型网站：
  - WEB+桌面+APP：建议选择ICAS方案
  - Web+APP：建议选择TJJO
- GIS系统开发
  - 无特殊要求：ArcGIS系列
  - 要求国内：MapGIS系列

四、系统设计的案例

原始需求

- 开发一个桂林理工大学三维校园信息管理系统，能够实现校园真三维模型漫游、校园设施、部门、学院、宿舍等信息查询与统计；实现教室上课情况查询统计、实现校园监控在线查看。要求系统漫游和查询统计功能能够在手机、微信和易班中使用。



谢 谢 聆 听