智慧农业云平台系统软件

体系结构设计文档

〈1.0〉

2023.6.19

李影（2024523008）

2022211319 班

软件体系结构

2025 春

修改记录

日期	描述	修改人姓名	内容
2025.5.22	版本 1.0	李影	完成目录和提纲整理

文档审批

该软件需求说明书被如下人员支持批准：

签名	打印版签名	名称	日期

目录

1. 介绍

1.1. 编写目的

本文档描述智慧农业云平台的整体架构设计，旨在为系统开发团队提供清晰的架构指导，确保系统能够满足农业生产管理的实际需求，并具备良好的可扩展性、可维护性和可靠性。

1.2. 范围

1.2.1. 本文档的读者

项目经理：了解系统整体架构和开发计划

开发人员：理解系统组件和实现方法

测试人员：了解系统功能和结构，制定测试策略

运维人员：了解系统部署和维护要求

农业领域专家：评估系统是否满足农业生产需求

1.2.2. 文档范围

本体系结构文档将从项目背景，项目开发的目的以及项目风险出发，说明系统的整体架构、设计思想与目标。文档还从逻辑分解视图、分层视图、进程视图与部署视图四个方面，结合非功能性需求对项目结构进行详细描述，并从数据流风格、抽象数据类型、分层风格的设计模式进行分析，提出了一个体系结构解决方案，以满足智慧农业云平台的需求。最后，文档还将从体系结构的质量分析与评价标准入手，对整个系统的设计与场景进行总体评估，从而为需求方与项目开发团队带来项目体系结构的总体架构情况与设计指导，为后续详细设计、代码实现和集成、运行和维护提供依据。

1.2.3. 系统适用范围

大中型农场和种植基地：实现生产全过程管理

农业合作社：集中管理多个小农户的生产活动

农业管理部门：监督和指导农业生产

农产品加工企业：追溯原材料生产情况

农业科研机构：收集和分析农业生产数据

1.3. 术语定义

云平台：基于互联网提供计算和存储服务的平台

传感器：采集农业环境数据（温度、湿度、光照等）的设备

控制设备：执行灌溉、施肥等操作的自动化设备

数据库：存储和管理数据的软件系统

接口：系统与外部设备或系统通信的连接点

网关：连接传感网络和互联网的设备

API：应用程序编程接口，提供软件间的通信方法

GIS：地理信息系统，用于处理地理空间数据

1.4. 参考资料

《农业信息化建设指南》（2020版）

《软件系统架构设计基础》第3版

《农业物联网应用规范》GB/T 35768-2017

《信息系统设计与实现规范》GB/T 14394-2008

《数据中心设计规范》GB 50174-2017

2. 体系结构需求

2.1. 产品总体描述

智慧农业云平台是一套面向现代农业生产的综合信息管理系统，集成了环境监测、智能灌溉、生产管理、病虫害防治和产品溯源等功能。平台通过连接各类农业传感器和控制设备，实时采集农田环境数据，自动控制农业生产设备，并提供生产计划管理、农事记录、专家知识库等功能，帮助农业生产者科学决策，提高产量和质量，降低成本和风险。

系统主要包括以下功能板块：

环境监测与预警

智能灌溉与施肥

农事活动管理

病虫害监测与防治

农产品质量追溯

农业知识库

生产分析与决策支持

2.2. 体系结构用例

2.2.1. 系统管理员

管理用户账号和权限

配置系统参数和设备信息

监控系统运行状态

管理数据备份和恢复

处理系统异常和故障

2.2.2. 农场管理者

查看农场整体运营数据和统计分析

制定生产计划和任务安排

分配农事活动任务

监控生产进度和成本

查看预警信息并做出决策

生成各类生产报表

2.2.3. 农业技术人员

监控农田环境数据（温度、湿度、光照等）

操作和调整农业设备（灌溉系统、施肥系统等）

记录农事活动（播种、施肥、用药等）

监测作物生长状况

识别和处理病虫害问题

查询专业知识库

2.3. 各相关方对体系结构的要求

2.3.1. 用户层面

易用性：系统操作简单直观，界面友好，适合不同文化程度的用户

响应性：系统响应速度快，实时显示关键数据

可靠性：系统稳定运行，数据准确可靠

移动支持：支持手机、平板等移动设备访问，方便田间作业时使用

离线功能：在网络不稳定时仍能使用基本功能，待网络恢复后自动同步

2.3.2. 其他涉众层面

设备厂商：需要简单的设备接入规范和接口

第三方应用开发者：需要标准的API接口获取数据

农业专家：需要便捷的知识库管理和远程诊断工具

政府监管部门：需要农产品质量追溯和生产统计数据

农产品加工企业：需要原材料生产全过程记录

2.4. 约束条件

2.4.1. 基本约束

网络环境：系统需适应农村地区网络条件差、不稳定的情况

设备多样性：支持各类传感器和控制设备的接入

用户水平：考虑用户计算机技术水平有限，提供简单操作方式

成本控制：系统总体拥有成本需控制在农业生产者可接受范围内

安全可靠：保证农业生产数据安全和系统可靠运行

2.4.2. 设计约束

技术选择：使用成熟稳定的技术框架，避免使用过于前沿的技术

兼容性：支持主流浏览器和移动设备

扩展性：支持新设备、新功能的便捷添加

部署灵活：支持云端部署和本地部署两种方式

标准遵循：遵循农业信息化相关标准规范

2.5. 非功能需求

2.5.1. 性能

页面加载时间不超过3秒

数据更新频率可配置，最快支持5秒一次

支持100个并发用户访问

系统处理1000个传感器数据无明显延迟

历史数据查询响应时间不超过5秒

2.5.2. 可用性

系统年可用率不低于99%

计划内维护时间每月不超过4小时

系统故障恢复时间不超过2小时

关键数据自动备份，确保数据不丢失

提供操作引导和帮助文档

2.5.3. 密安性

用户访问需要身份认证

敏感数据传输加密

不同角色用户权限严格区分

重要操作需要审核和记录

定期安全审计和漏洞扫描

2.5.4. 可维护性

模块化设计，便于维护和升级

提供完整的系统日志

支持远程诊断和问题定位

配置参数可在线调整

提供系统监控和性能分析工具

2.5.5. 可移植性

支持不同操作系统环境

数据可导入导出为标准格式

支持与其他农业系统对接

适应不同规模的农业生产主体

2.6. 风险

网络风险：农村地区网络不稳定可能导致数据采集中断

设备风险：农业物联网设备在恶劣环境中可能故障率高

数据风险：大量传感器数据的存储和处理压力

安全风险：系统可能面临网络攻击和数据泄露

使用风险：用户接受度和操作熟练度不足

3. 体系结构分析

3.1. 体系结构参考模型

3.1.1. 模型适应性分析

智慧农业云平台采用"三层架构+服务组件"的模型，这种模型简单清晰，适合农业信息化系统的特点，能够满足数据采集、处理和应用的需求，同时具有良好的扩展性。

3.1.2. 具体介绍

系统分为以下几个层次：

数据采集层：连接各类农业传感器和控制设备，采集环境数据和设备状态，执行控制命令

数据处理层：存储和处理采集的数据，包括数据清洗、分析和挖掘

应用服务层：提供各类业务功能和用户界面，实现农业生产管理

此外，系统还包含若干公共服务组件，如认证授权、消息通知、文件存储等，为各层次提供支持。

3.2. 体系结构基本框架示意图

3.3. 体系结构关键技术

3.3.1. 物联网技术

无线传感器网络技术用于农田环境数据采集

低功耗广域网技术(LoRa)解决农村通信距离问题

标准协议(MQTT)实现设备统一接入管理

3.3.2. 云计算技术

云服务器提供计算和存储资源

分布式数据库存储大量农业生产数据

弹性计算适应季节性业务波动

3.3.3. 数据处理技术

实时数据处理技术处理传感器数据流

数据挖掘技术分析作物生长规律

地理信息系统(GIS)进行空间数据分析

4. 视图

4.1. 模块分解视图

4.1.1. 一级系统分解图

智慧农业云平台一级系统分为四大模块：

设备管理模块：负责设备接入、数据采集和设备控制

数据管理模块：负责数据存储、处理和分析

业务功能模块：实现各类农业生产管理功能

系统管理模块：负责用户、权限和系统配置管理

4.1.2. 二级系统分解图

业务功能模块进一步分解为：

环境监测模块：实时监测农田环境参数
智能控制模块：自动化控制灌溉、施肥等设备
生产管理模块：管理生产计划、任务和记录
病虫害防治模块：监测、预警和处理病虫害
产品追溯模块：记录农产品生产全过程
决策支持模块：提供数据分析和决策建议

4.1.3. 三级系统分解图

环境监测模块进一步分解为：

数据采集单元：采集环境数据
数据显示单元：可视化展示数据
阈值设置单元：设置监测参数阈值
预警处理单元：处理超阈值预警
历史数据单元：查询历史监测数据
报表生成单元：生成环境监测报表

4.2. 分层视图

4.2.1. 表示层

Web门户：提供PC端浏览器访问界面
移动应用：提供手机和平板访问界面
大屏展示：提供数据可视化大屏
设备控制面板：提供设备操作界面
表示层关注用户体验和数据可视化，采用响应式设计适应不同终端设备。

4.2.2. 应用层

环境监测模块：温度、湿度、光照等环境参数监测

智能控制模块：灌溉、施肥、通风等设备控制

生产管理模块：计划制定、任务分配、进度跟踪

病虫害防治模块：识别、预警、防治方案

产品追溯模块：种植记录、投入品使用记录

决策支持模块：数据分析、生产建议

应用层实现各类业务功能，封装业务规则和流程。

4.2.3. 数据层

关系型数据库：存储结构化业务数据

时序数据库：存储传感器时间序列数据

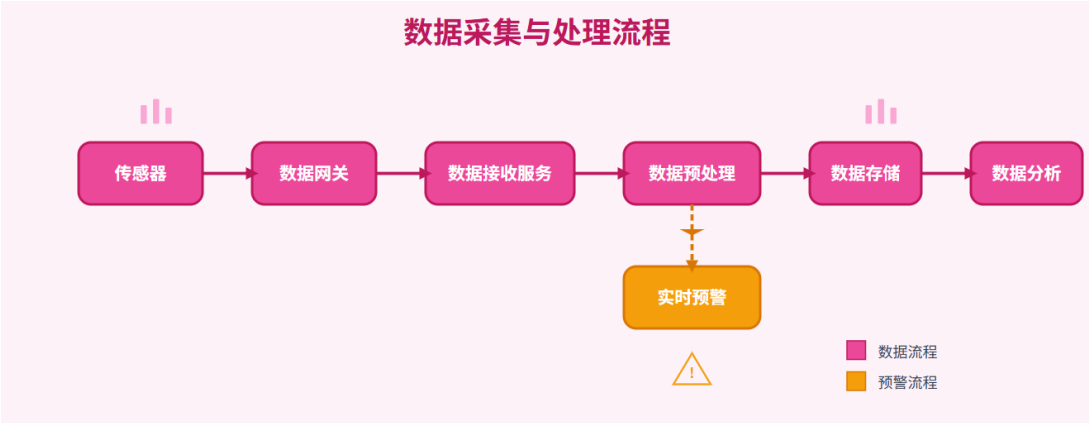
文件存储：存储图片、文档等非结构化数据

缓存系统：提高频繁访问数据的响应速度

数据层负责数据的存储、访问和管理，确保数据安全和一致性。

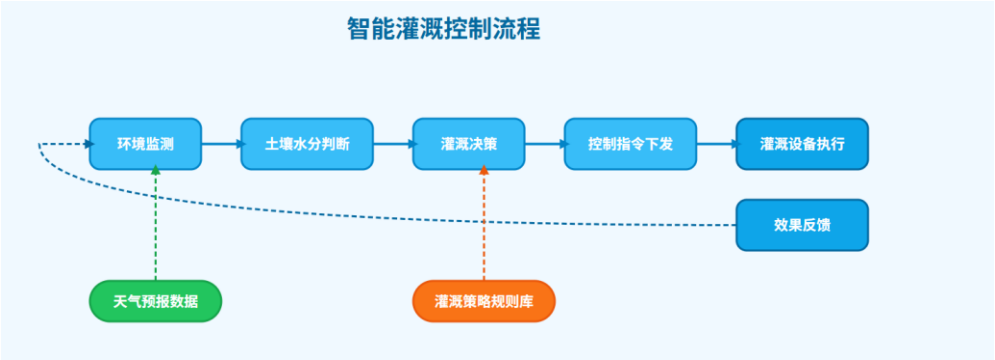
4.3. 进程视图

4.3.1. 数据采集与处理流程



这一流程展示了从传感器数据采集到存储分析的完整过程，包括数据接收、预处理、存储和分析等步骤。

4.3.2. 智能灌溉控制流程



这一流程展示了智能灌溉的决策和执行过程，结合环境监测数据和灌溉规则，实现精准灌溉。

4.4. 部署视图

4.4.1. 云端部署架构

系统主要组件部署在云服务器上，包括：

应用服务器：运行业务功能模块

数据库服务器：存储系统数据

文件服务器：存储文档和图片

缓存服务器：提供数据缓存

网关服务器：连接物联网设备

4.4.2. 现场设备部署

在农田现场部署的设备包括：

各类传感器：温度、湿度、光照等

控制设备：灌溉阀门、水泵等

数据网关：连接传感器和互联网

边缘计算设备：处理部分本地数据

5. 体系结构解决方案

5.1. 数据流风格

5.1.1. 应用原理

系统采用数据流风格，围绕数据的采集、传输、处理和利用设计系统架构。数据从传感器采集后，经过网关传输到云平台，然后经过存储、处理和分析，最终提供给用户使用或驱动设备控制。

5.1.2. 质量属性分析

优势：适合处理大量实时数据流，结构清晰，扩展性好

劣势：在网络不稳定时可能影响数据完整性，需要考虑数据缓存和恢复机制

适用性：非常适合农业物联网场景下的数据处理需求

5.2. 抽象数据类型

5.2.1. 应用原理

系统将农田、作物、设备等实体抽象为标准数据模型，定义它们的属性和操作方法。例如，"农田"模型包含位置、面积、土壤类型等属性，以及耕种、灌溉等操作。

5.2.2. 质量属性分析

优势：提高系统的模块化程度，便于功能扩展和代码重用

劣势：初始设计难度较大，需要深入理解农业领域知识

适用性：适合需要处理复杂农业对象关系的系统

5.3. 分层风格

5.3.1. 应用原理

系统采用经典的分层架构，将表示、应用逻辑、数据处理分离，各层之间通过定义好的接口通信。上层只能调用下层服务，不能跨层调用。

5.3.2. 质量属性分析

优势：责任划分清晰，降低系统复杂度，有利于团队协作和系统维护

劣势：层次过多可能影响性能，需要适当权衡

适用性：适合功能复杂、团队规模较大的项目

5.4. 服务架构方案

5.4.1. 服务划分

系统按照业务功能划分为多个相对独立的服务模块：

设备管理服务

数据存储服务

环境监测服务

智能控制服务

生产管理服务

病虫害防治服务

用户认证服务

5.4.2. 服务通信

服务之间通过标准API接口通信，采用HTTP/JSON格式，便于集成和扩展。关键服务可以使用消息队列实现异步通信，提高系统响应能力。

5.5. 数据处理方案

5.5.1. 实时数据处理

对传感器实时数据进行处理，包括：

数据清洗：过滤异常值

数据转换：单位转换和格式标准化

实时计算：计算衍生指标

预警判断：与阈值比较并触发预警

5.5.2. 历史数据分析

对历史积累的农业生产数据进行分析：

趋势分析：环境参数变化趋势

相关性分析：环境与产量关系

模式识别：发现生产规律

预测模型：产量和质量预测

6. 系统安全架构

6.1. 安全策略

6.1.1. 设备安全

设备注册认证：新设备接入需要认证

设备通信加密：数据传输过程加密

设备固件更新：支持远程更新设备固件

异常监测：监测设备异常行为

6.2. 认证与授权

6.2.1. 用户认证

账号密码认证：基本认证方式

短信验证：敏感操作二次认证

登录限制：异常登录检测和限制

密码策略：强制密码复杂度和定期更换

6.2.2. 访问控制

基于角色的访问控制：不同角色有不同权限

数据访问控制：用户只能访问授权的数据

操作审计：记录重要操作日志

敏感数据保护：加密存储敏感信息

7. 软件生产线方案

7.1. 组织结构

7.1.1. 团队分工

前端开发团队：负责Web和移动应用界面开发

后端开发团队：负责业务逻辑和数据处理开发

物联网团队：负责设备接入和通信

测试团队：负责功能和性能测试

运维团队：负责系统部署和维护

7.1.2. 协作模式

采用敏捷开发方法，进行迭代开发：

两周一次迭代计划会

每日站会交流进度

迭代结束进行演示和回顾

持续集成确保代码质量

7.2. 系统框架

7.2.1. 技术选择

前端技术：HTML5, CSS3, JavaScript, Vue.js

后端技术：Java, Spring Boot

数据库：MySQL(业务数据), InfluxDB(时序数据)

Web服务器：Nginx

缓存：Redis

消息队列：RabbitMQ

7.2.2. 开发工具

代码管理：Git

项目管理：Jira

文档管理：Confluence

持续集成：Jenkins

测试工具：JUnit, Selenium

8. 系统演进策略

8.1. 版本规划

8.1.1. 迭代计划

系统开发分为以下几个阶段：

第一阶段：基础平台搭建和核心功能实现

第二阶段：完善业务功能和优化用户体验

第三阶段：增加高级分析功能和决策支持

第四阶段：完善生态体系和第三方集成

8.1.2. 功能优先级

功能开发优先级排序：

环境监测和数据采集

设备控制和农事记录

生产计划和任务管理

病虫害监测和预警

决策分析和专家系统

8.2. 扩展性设计

8.2.1. 功能扩展

模块化设计，便于添加新功能

插件机制支持第三方功能扩展

标准API接口支持外部系统集成

配置化设计减少代码修改

8.2.2. 容量扩展

数据库分表分库应对数据增长

服务器集群支持用户规模扩大

缓存机制提高系统并发处理能力

分布式存储支持海量数据存储