概要设计规约文档

植保精灵(PGUARD)

宋宇然 陈晓坤 许经宝

**修订历史记录：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编写日期 | SEPG | 版本 | 说明 | 作者 |
| 2024/12/ |  | v1.0 |  | 许经宝、  陈晓坤、  宋宇然 |
|  |  |  |  |  |

1. **引言**

**1.1概要设计依据**

1. 杜庆峰老师上课所讲的概要设计基本原则；

2. 需求分析规约文档——植保精灵(PGUARD)

**1.2参考资料**

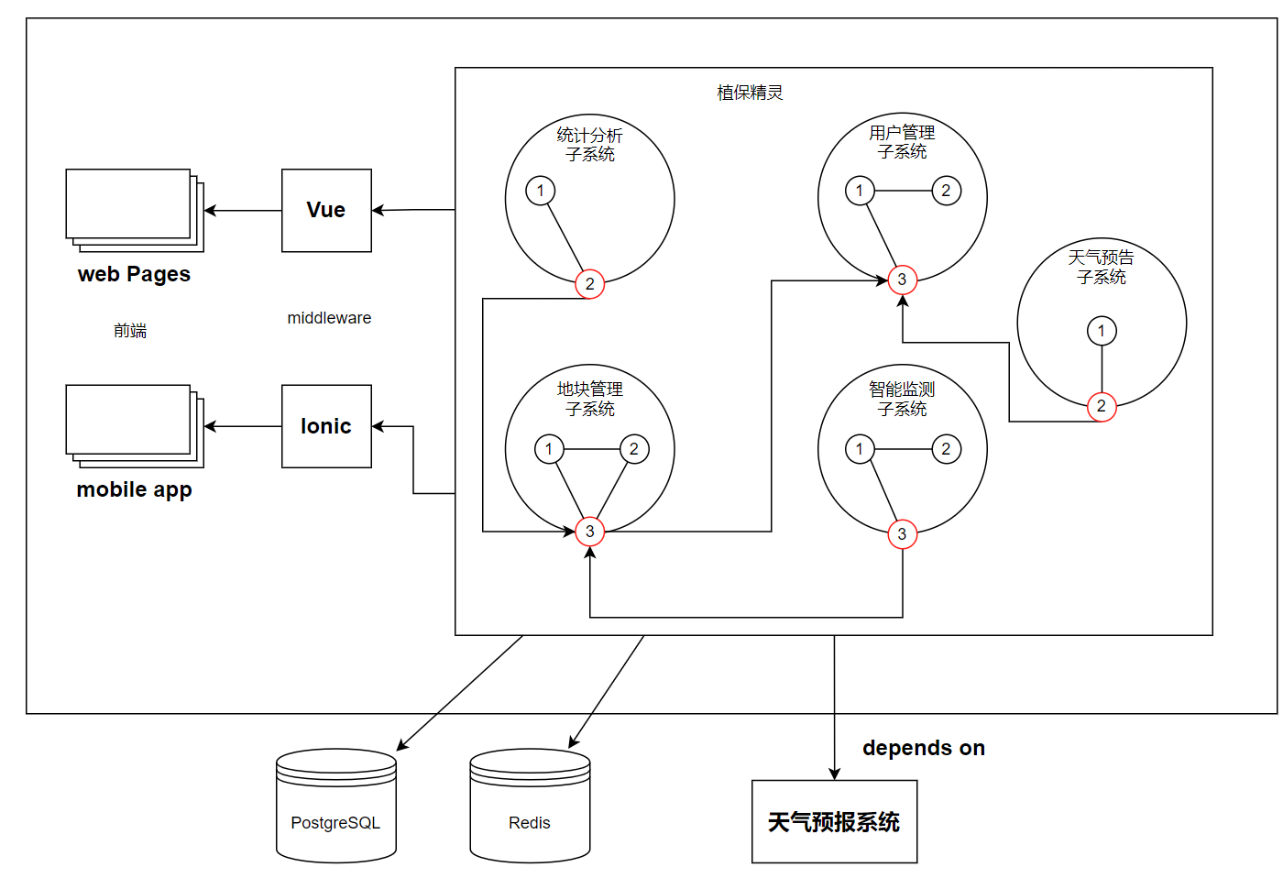
1. 《软件工程: 体系结构、设计、编码》
2. 《Design Patterns: Elements of Reusable Object-Oriented Software》
3. 需求规约文档——植保精灵(PGUARD)
4. 需求分析规约文档——植保精灵(PGUARD)

**1.3假定和约束**

本项目开发主要受软件工程课程的约束，因此本项目的假定和约束如下所示：

1. 项目开发期限为2个月，时间为2024年11月上旬~2025年1月上旬
2. 项目开发无经费，设备条件为3台个人电脑以及华为云服务器
3. 项目在开发前已对同领域软件耘眼的工作流程进行了充分的研究
4. 在开发过程中，每周周一、周四线下开组会，汇报工作进度，同时通过git和Github进行版本控制
5. **概要设计**

**2.1系统体系架构设计**



用户管理子系统：

1：User类；2：Package类；3：UserController

地块管理子系统：

1：Plot类；2：Plant类；3：PlotController

天气预告子系统：

1：City类；2：WeatherController

智能检测子系统：

1：Disease类；2：DetectController

统计分析子系统：

1：Log类；2：DetectPicture类；3：LogController

**2.2软件体系结构设计**

系统按照功能划分为以下五个子系统，以下是子系统的功能说明、主要类和调用关系：

2.2.1用户管理子系统

功能描述：

提供管理用户信息功能，包括用户注册、用户登入等功能，同时还有套餐充值服务功能

核心类及调用关系

User：用户类，存储用户信息

Package：套餐包类，存储了用户充值套餐的相关信息

UserController：控制器类，接收用户请求，调用User服务

调用关系图



2.2.2地块管理子系统

功能描述：

管理地块信息，包括地块的创建、更新和种植作物

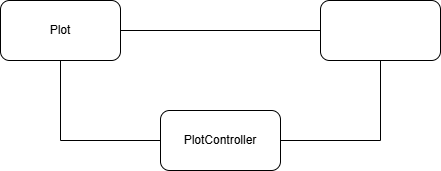
核心类及调用关系

Plot：地块类，存储地块的地理信息和基本属性

Plant：作物类，存储作物的相关数据，包括作物名、作物特点

PlotController：控制类，处理地块管理的业务逻辑

调用关系图



2.2.3天气预告子系统

功能描述：

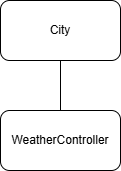
基于地理位置获取对应的天气数据

核心类及调用关系

City：城市类，存储了城市名和该城市在外部天气系统中的编码号

WeatherController：控制器类，调用外部天气系统API并处理结果

调用关系图



2.2.4智能检测子系统

功能描述：

上传植物叶片图片，进行病虫害检测，并生成防治建议

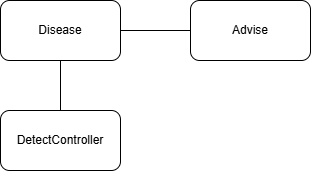
核心类及调用关系

Disease：病虫害类，定义病虫害种类

Advise：建议类，提供对病虫害的防治信息

DetectController：控制器类，调用智能病虫害检测逻辑

调用关系图



2.2.5统计分析子系统

功能描述：

记录用户病虫害检测日志，存储检测的时间和检测结果信息

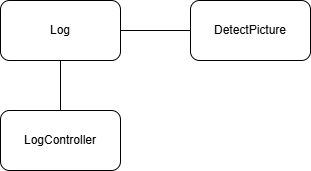
核心类及调用关系

Log：日志类，存储检测日志

DetectPicture：检测图片类，存储检测前截取的图片

LogController：控制器类，处理日志查询和分析功能

调用关系图



2.2.6类调用关系综合图

交互关系

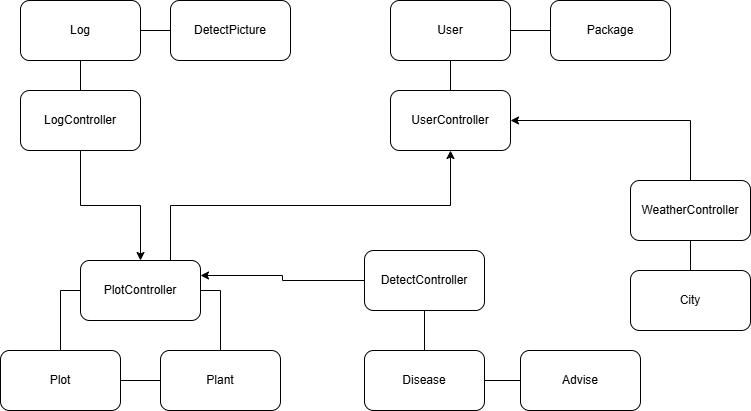
1.用户管理子系统

用户管理子系统和地块管理子系统以及天气预告子系统直接交互，用户可以通过这直接进行管理自己的地块信息，包括新增修改和删除地块等操作。同时可以直接查看该地区的天气情况。

2.地块管理子系统

地块管理子系统与智能检测子系统和总结分析子系统直接交互，用户只能通过地块子系统与智能检测和统计分析进行交互，只有在进入地块后才能使用智能检测和获取统计分析的功能。

交互图



**2.3接口设计**

**2.3.1智能检测接口**

描述：

该接口根据输入的 model\_type 和 image\_path，调用对应的检测脚本（如 Grape\_defect.py 或 Potato\_defect.py），运行目标检测任务，并解析检测结果。输入参数：

model\_type：智能检测模型类型

image\_path：要检测图片的位置

返回值：

智能检测后的数据，包含病虫害结果和识别概率

异常：

ValueError：

当 model\_type 不被支持时，抛出异常提示。

2.3.2用户识别接口

描述：该接口是一个异步函数，用于通过 OAuth2 令牌 (token) 验证用户身份，检查用户是否存在并返回用户对象。  
它支持依赖注入机制，供其他模块调用时动态注入当前用户信息。

输入参数：

token：通过 OAuth2 认证方案提供的用户访问令牌。使用 Depends 注入 oauth2\_scheme 获取令牌。

返回值：

返回经过认证的 User 对象，代表当前登录用户。

异常：

状态码 401 UNAUTHORIZED:

 如果令牌无法通过验证（如格式错误、黑名单存在、用户 ID 缺失）。

 如果无法通过用户 ID 查询到对应的用户信息。

2.3.3创建日志接口

描述：该接口通过接收地块 ID、病害名称、建议内容及图片 URL，生成检测日志并存储到数据库中。

**输入参数**:

plotId (str): 地块的唯一标识符，用于关联日志到具体地块。

diseaseName (str): 检测到的病害名称，例如 "霜霉病"。

advice (str): 针对病害的建议内容。

imageURL (str): 相关病害图片的 URL。

**返回值**:

str: 操作结果信息：

成功返回 "创建日志成功"。

失败返回 500 错误，包含具体错误原因。

**异常**:

**HTTPException**:

状态码：500。

当日志创建失败时，返回错误详情。

2.3.4根据Id获取地块接口

描述：通过接收地块 ID (plotId)，从数据库中检索地块记录，同时预加载其关联的用户信息 (userId) 和植物信息 (plantId)。

输入参数：plotId (str): 地块的唯一标识符，用于定位数据库中的特定地块。

返回值：Plot: 返回包含 Plot 对象的完整信息，包括预加载的 userId 和 plantId。

**异常**:

**HTTPException**:

状态码：404。

当地块 ID 不存在或查询失败时，抛出异常并返回错误详情。

2.3.5根据用户获取所有地块接口

描述：通过依赖项获取当前登录用户，并根据用户 ID (userId) 查询该用户的所有地块信息，同时预加载关联的植物信息 (plantId)。

输入参数：user (User): 通过依赖项 get\_current\_user 获取当前登录的用户对象。

返回值：List[Plot]: 返回包含用户地块信息的列表，每个地块对象都包含预加载的植物信息。

异常：**HTTPException**:

状态码：404。

当查询不到地块记录或发生错误时，抛出异常并返回错误详情。

2.3.6获取日志接口

描述：通过地块 ID (plotId) 获取与地块关联的所有日志信息，按时间正序排列后返回。

输入参数：plotId (str): 地块的唯一标识符，用于查询日志记录。

返回值：List[LogDetail]: 包含日志详细信息的列表。

异常：如果地块不存在或查询失败，将抛出异常。

2.3.7分析日志接口

描述：该异步接口用于分析一组 plot\_details，计算地块数量、植物占用的地块数量、每月病害发生的统计、植物相关病害统计、每种病害的发生次数等。最终，接口将返回最常发生的病害类型的预测结果。

输入参数：

**plot\_details (List[PlotDetails])**: 一个包含多个 PlotDetails 对象的列表。每个 PlotDetails 对象包含植物信息、地块ID和日志记录等。

返回值：该接口返回一个字典，包含以下信息：

**plot\_count**: 不同地块的数量（通过去重 plotId 实现）。

**plant\_plot\_count**: 每种植物在不同地块上占用的数量，按植物名称进行分组。

**monthly\_disease\_count**: 每月病害发生的次数，长度为12，表示每个月的病害数量。

**plant\_disease\_count**: 每种植物相关的病害发生次数。

**disease\_count**: 每种病害发生的次数。

**prediction**: 最常发生的病害的预测结果，通过 get\_prediction\_by\_name 函数获取。

异常：在解析日志的时间戳时，若发生错误（例如格式不符），会捕获异常并打印错误信息，然后继续处理下一个日志。

2.3.8检测结果查找接口

描述：该接口通过病害名称 diseaseName 查询对应的病害预测信息。如果找到了相关病害记录，返回其预测结果。如果查询失败或发生异常，则抛出异常。

输入参数：**diseaseName (str)**: 病害的名称，用于查询数据库中对应病害的预测信息。

返回值：**prediction (str)**: 返回与病害名称对应的预测结果。如果找到病害记录，将返回其相关的预测信息。

异常：如果查询过程中发生错误或无法找到对应的病害记录，则抛出 HTTPException，状态码为 404，并返回异常信息。

**2.4界面设计**

**2.5外部接口设计**

**2.6数据库设计**

**数据库结构名称**

1. User (用户)
2. Package (套餐)
3. Plant (植物)
4. Plot (地块)
5. Log (日志)
6. Disease (疾病)
7. City (城市)

**标识符及数据项详情**

**User**

* userId: UUID, 主键, 默认值为uuid.uuid4()
* userName: 字符串, 最大长度40, 唯一
* password: 字符串, 最大长度100
* location: 字符串, 最大长度40
* sumCount: 小整数

**Package**

* packageId: UUID, 主键, 默认值为uuid.uuid4()
* packageName: 字符串, 最大长度40
* price: 浮点数
* sumNum: 小整数

**Plant**

* plantId: UUID, 主键, 默认值为uuid.uuid4()
* plantName: 字符串, 最大长度40, 唯一
* plantFeature: 文本字段
* plantIconURL: 字符串, 最大长度100

**Plot**

* plotId: UUID, 主键, 默认值为uuid.uuid4()
* plotName: 字符串, 最大长度40
* userId: 外键, 关联到User表的userId
* plantId: 外键, 关联到Plant表的plantId

**Log**

* logId: UUID, 主键, 默认值为uuid.uuid4()
* plotId: 外键, 关联到Plot表的plotId, 级联删除
* timeStamp: 日期时间, 自动添加当前时间
* diseaseName: 字符串, 最大长度40
* content: 文本字段
* imagesURL: 字符串, 最大长度200

**Disease**

* diseaseId: UUID, 主键, 默认值为uuid.uuid4()
* plantId: 外键, 关联到Plant表的plantId, 级联删除
* diseaseName: 字符串, 最大长度40
* advice: 文本字段
* prediction: 文本字段, 可为空

**City**

* cityCode: 字符串, 主键, 最大长度10
* cityName: 字符串, 最大长度40, 唯一

**层次或表格相互关系**

* User与Plot之间是一对多的关系，一个用户可以拥有多个地块。
* Plant与Plot之间是一对多的关系，一种植物可以出现在多个地块中。
* Plant与Disease之间是一对多的关系，一种植物可能会有多种疾病。
* Plot与Log之间是一对多的关系，一个地块可以有多个日志记录。
* City没有直接与其他表建立关系，是独立的。

**2.7系统出错处理设计**

**2.7.1 出错信息**

|  |  |  |
| --- | --- | --- |
| **可能的错误** | **出错信息** | **处理方法** |
| **图像或视频上传格式不正确或损坏** | **图像格式错误或损坏，请上传有效的图像文件。** | **抛出异常，图像格式不正确或图像损坏，要求上传有效图像** |
| **数据库连接超时。** | **无法连接到数据库，系统正在维护中，请稍后再试。** | **抛出异常，无法连接到数据库，可能处于维护中。** |
| **图像质量不佳，无法进行准确的病虫害检测** | **图像质量过低，无法进行有效分析，请上传更清晰的图像** | **抛出异常，图像质量过低，请上传清晰的图像** |
| **存储空间不足，导致无法存储日志或模型数据** | **存储空间不足，无法保存日志或结果，请联系管理员** | **抛出异常，存储空间不足，无法保存日志或结果** |
| **网络请求超时或错误响应** | **外部服务响应错误，请检查网络或稍后再试** | **抛出异常，服务相应错误，请检查网络设置** |

**2.7.2补救措施**

**2.7.2.1输入数据错误**

**在用户提交数据时，进行输入验证（例如，检查图像文件是否符合要求）。**

**如果输入无效，提供明确的错误提示信息并要求用户重新输入或修复数据。**

**2.7.2.2数据库连接错误**

**重新尝试连接数据库，设置合理的重试次数和重试间隔。**

**提供备份数据库或离线操作功能，确保在数据库不可用时系统仍能提供部分服务。**

**2.7.2.3模型推理错误**

**在上传图像时，增加图像质量提示，并尽量提供支持的图像格式。**

**提供图像预处理功能（例如降噪、增强对比度），提高模型识别准确性。**

**2.7.2.4API调用错误**

**对外部API调用增加重试机制，避免由于短暂的网络问题导致系统无法提供服务。**

**如果API调用连续失败，可以向用户提供替代方案（例如，显示历史数据或进行本地推理）。**

**2.7.2.5系统资源不足错误**

**采用负载均衡和资源调度策略，确保系统能够平稳处理高并发请求。**

**定期清理不必要的临时数据和过期日志文件，确保系统存储空间充足。**

**2.7.3系统维护设计**

**2.7.3.1输入数据错误**

**数据验证：在前端进行基本的输入验证，确保数据符合预期格式。后端进行进一步验证，确保输入的地块ID、病虫害名称和图像数据的合法性。**

**日志记录：记录每次无效输入的具体信息，包括时间、用户ID、输入数据、错误类型等，便于后续分析和改进。**

**2.7.3.2数据库连接错误**

**数据库监控：定期检查数据库连接状态，确保数据库可用。如果数据库出现异常，自动切换到备用数据库或通知运维人员。**

**日志记录：记录数据库连接失败的日志，包含错误时间、原因、重试次数等信息，便于后续分析和优化。**

**2.7.3.3模型推理错误**

**图像处理优化：定期优化图像处理算法，确保能够处理不同类型的图像数据。利用深度学习模型进行自动图像修复或补充。**

**模型监控与更新：监控模型的推理效果，当发现推理失败率过高时，启动模型重训练或微调，并确保使用最新版本的病虫害检测模型。**

**故障分析与报告：记录每次图像处理失败的日志，包括图像ID、错误类型、失败原因等，便于后续调查和修复。**

**2.7.3.4API调用错误**

**API监控：监控外部API的可用性，记录每次调用的响应时间、状态码等信息，确保外部服务可用。**

**异常通知：当API调用失败时，及时通知管理员或相关人员，并根据预设的规则进行自动处理或修复。**

**2.7.3.5系统资源不足错误**

**资源监控：使用资源监控工具（如 Prometheus、Grafana）实时监控系统的CPU、内存、磁盘空间使用情况，及时发现并处理资源瓶颈。**

**定期清理：定期清理过期的日志文件和临时数据，防止磁盘空间占满。使用数据归档或压缩存储较旧的历史数据。**