



南開大學  
Nankai University

网络空间安全学院  
软件工程

# 智慧可视化系统 软件设计报告

学院：网络空间安全学院

年级：2021 级

班级：信息安全一班

学号：2111408

姓名：周钰宸

指导教师：刘健

2024 年 5 月 7 日

目录

<b>1</b>	<b>引言</b>	<b>3</b>
1.1	编写目的	3
1.2	项目背景	3
<b>2</b>	<b>用例图</b>	<b>4</b>
2.1	用例图简介	4
2.2	用例图展示	5
2.3	用例图阐述	8
<b>3</b>	<b>活动图</b>	<b>8</b>
3.1	活动图简介	8
3.2	活动图展示	10
3.3	活动图阐述	14
<b>4</b>	<b>类图</b>	<b>14</b>
4.1	类图简介	14
4.2	类图展示	16
4.3	类图阐述	16
4.3.1	类图详细解析	16
4.3.2	类之间的关系	20
<b>5</b>	<b>顺序图</b>	<b>21</b>
5.1	顺序图简介	21
5.2	顺序图展示	22
5.3	顺序图阐述	25
<b>6</b>	<b>协作图</b>	<b>26</b>
6.1	协作图简介	26
6.2	协作图展示	27
6.3	协作图阐述	29
<b>7</b>	<b>状态图</b>	<b>30</b>
7.1	状态图简介	30
7.2	状态图展示	31
7.3	状态图阐述	31
<b>8</b>	<b>构件图</b>	<b>32</b>
8.1	构件图简介	32
8.2	构件图展示	33
8.3	构件图阐述	33

<b>9 部署图</b>	<b>34</b>
9.1 部署图简介 . . . . .	34
9.2 部署图展示 . . . . .	35
9.3 部署图阐述 . . . . .	35

# 1 引言

## 软件设计文档的重要性

软件设计文档在软件开发过程中起着至关重要的作用，其主要目的是确保开发团队能够系统性地设计和实现软件项目。这份文档通常详尽地记录了系统的架构、设计原则、接口定义、数据结构、算法设计以及用户界面设计等关键设计决策。软件设计文档的重要性在于可以确保团队成员对系统设计和行为有一致理解，提高效率，便于维护并促进沟通。

### 1.1 编写目的

本文档的主要目的是为**海洋牧场监测可视化系统**的软件设计提供详细、准确且全面的指南，确保系统架构的合理性，技术实现的可行性，以及项目开发的技术细节能够满足需求分析所确定的标准。**文档将主要通过统一建模语言 UML 用例图，活动图，类图，顺序图，协作图，状态图，构件图和部署图，对整个软件的设计架构、各模块的设计原则、接口定义、数据结构、算法设计以及用户界面设计等关键设计决策进行详细阐述。**

通过本文档，开发团队将能够深入理解系统的架构设计和模块划分，确保各个部分的设计和实现与既定需求保持一致，并符合最佳实践。本文档旨在指导开发人员在编码过程中作出技术选择，并提供必要的设计依据，以便快速、有效地进行问题定位和解决。

本文档的预期读者包括系统架构师、开发人员、UI/UX 设计师、项目管理团队以及质量保证人员。通过阅读本文档，项目的各个利益相关者可以清晰地了解设计理念、系统的详细构造和操作流程。此外，本文档也将成为评估项目进度和质量的重要依据，并助力持续集成和持续部署的实施。

最终，本设计文档将确保软件开发的每个阶段都具有追溯性和透明度，同时促进团队间的沟通和协作。它也将帮助确保软件设计满足业务需求，并符合用户期望的系统性能和功能性，为实现高质量的海洋牧场监测可视化系统提供坚实的设计基础。

## UML 统一建模语言

UML (Unified Modeling Language)，是一种为面向对象系统的产品进行说明、可视化和编制文档的一种标准语言。使用面向对象设计的建模工具，但独立于任何具体程序设计语言。UML 有严格的语法和语义规范，采用一组图形符号来描述软件模型，可以直观地理解和阅读，由于具有规范性，所以能够保证模型的准确、一致。

### 1.2 项目背景

中国在全球渔业养殖领域占据重要地位，但目前面临着水产养殖系统效率低下和环境问题频发的挑战。为应对这些挑战，结合国家“十四五”期间对现代化养殖技术的重视，本项目计划开发一套**海洋牧场监测可视化系统**。该系统将采用物联网、大数据和云计算等先进技术，**实现海洋牧场环境的全面、实时、智能监控**。通过系统的四个主要模块——数据处理与分析、可视化展示、报警与通知以及用户信息管理，本系统旨在优化养殖环境，提高生产效率，减少病害发生，从而提升整体养殖业的可持续发展能力。

## 2 用例图

### 2.1 用例图简介

UML 用例图是用来描绘软件系统功能和用户交互的高层次图形。用例图通过视觉方式展示了系统的功能单元（即用例）以及这些功能单元如何对不同的用户角色（即参与者）可见。它们是用于捕获软件系统需求的重要工具，尤其是用于理解系统的业务功能及其与用户之间的交互。用例图强调了“谁”可以做“什么”，而不关注系统内部的具体操作。这种图通常在软件开发的早期阶段使用，帮助开发团队和利益相关者对系统功能达成共识。

用例图的主要组成元素包括：

- **用例：**表示系统可以执行的一个或多个动作的集合。它通常代表系统的一个业务功能。
- **参与者：**与系统交互的用户、组织或外部系统。
- **关联：**参与者与用例之间的交互，以及用例之间可能存在的依赖关系。比如继承、扩展、包含等。

关系类型	说明	表示符号
关联	参与者与用例之间的关系	—————
泛化	参与者之间或用例之间的关系	—————>
包含	用例之间的关系	- - - «includes» ->
扩展	用例之间的关系	- - - «extends» ->

图 2.1: UML 用例图使用说明

## 2.2 用例图展示

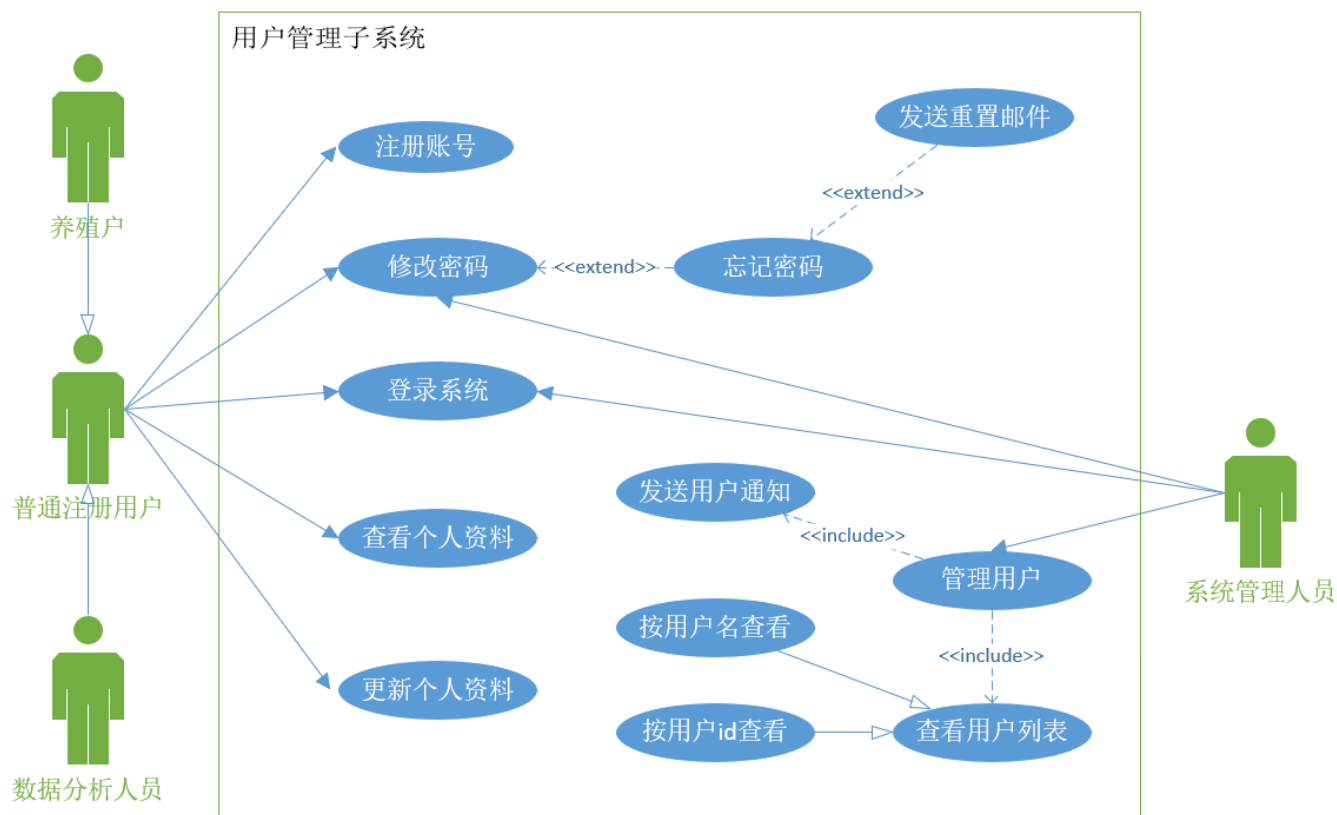


图 2.2: 用户管理子系统用例图

### 用户管理子系统阐述

关于**用户管理子系统**，如图2.2所示。首先由于对于所有正常用户（除去系统管理人员以外），包括养殖户，普通注册用户以及数据分析人员与子系统交互的行为完全一致。因此养殖户和数据分析人员都泛化继承于普通注册用户。

**普通注册用户**代表的所有正常用户的行为，包括注册账号，修改密码，登陆系统，查看个人资料，更新个人资料。其中如果用户在修改密码时候忘记了密码可以通过发送重置邮件来找回密码。

**系统管理人员**不同于普通用户，首先系统管理人员的帐号不能是随意注册得到的，而是系统在创立时候就已经分配好的。这是因为通过不允许一般用户注册成为系统管理人员来保证系统管理的安全性和可维护性。对于系统管理人员的行为，他最重要的行为就是管理用户。在管理用户时他可以查看用户列表。由于用户名可能出现较为类似的情况，并且在用户较多时候不方便管理，因此提供了两种方式，分别是按照用户名查看用户列表和按用户 id 查看用户列表，以此方便系统管理人员的操作。最后系统管理人员对用户进行任何操作后都需要向对应用户发送通知。



## 报警与通知子系统阐述

关于**报警与通知子系统**，如图2.3所示。也是整个海洋牧场可视化监测系统的关键部分之一，保证着系统的正常运行，对特殊情况进行应对。主要也分为两个用户对其进行交互，即养殖户和系统管理人员。两人都是对一些报警条件进行设置，不过一个在宏观上进行调整，一个进行微调。

对于**养殖户**，他可以查看所有报警规则，设置报警规则并接收系统的报警。其中设置报警规则时，可以进行增删改，即添加自定义报警，删除报警规则以及修改报警阈值。在接收到系统的报警通知时，养殖户便可以确认报警并执行应急措施。

对于**系统管理人员**，他不是在一些细微的规则上进行微调，而是在更大的尺度上，管理全局的报警设置。通过将系统管理人员和养殖户进行不同尺度的分工，方便两种不同的角色进行协调和合作，共同保证系统运行的稳定性

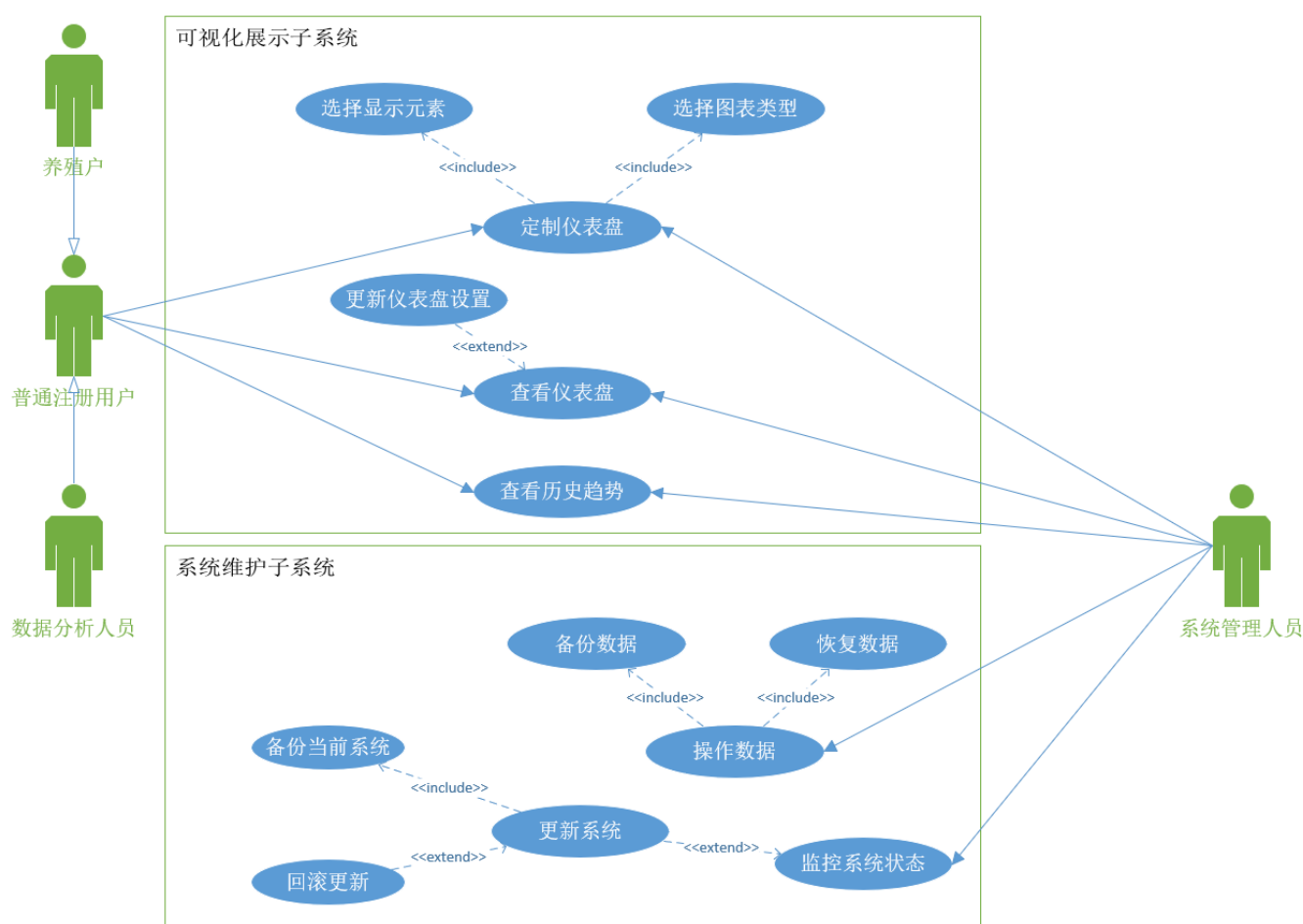


图 2.4: 可视化展示子系统和系统维护子系统

## 可视化展示子系统阐述

关于**可视化展示子系统**，如图2.4所示。主要负责对所有用户，包括普通注册用户，养殖户，数据分析人员以及系统管理人员进行可视化展示。

对于**所有用户**，不论是普通注册用户，养殖户，数据分析人员还是系统管理人员，都可以查看仪



表盘，定制仪表盘，并查看历史趋势。在查看仪表盘时候，可以根据需要更新仪表盘设置。在定制仪表盘时，可以选择显示的元素以及图表类型。

#### 系统维护子系统阐述

关于**系统维护子系统**，能与该系统进行交互的只有系统管理人员，以此来实现系统的稳定和安全。该子系统主要负责对整个海洋牧场可视化监测系统进行监控和调整。

**系统管理人员**可以监控系统状态并操作系统数据。在监控系统状态时。若发现系统老化等问题，可以更新系统。在更新系统时，若防止更新失败等意外情况，可以额外进行备份当前系统和回滚更新的操作。在操作系统数据时，也可以进行恢复数据和备份数据的操作。

### 2.3 用例图阐述

鉴于上面已经对每个子系统交互流程进行详细阐述。下面对上述用例图2.2，2.3和2.4进行整体阐述。

1. **用户管理子系统**：主要负责普通用户的注册登录，查看和修改个人信息，以及系统管理人员对用户信息进行查看和管理。参与交互的角色有普通注册用户，养殖户，数据分析人员以及系统管理人员。
2. **数据监测与分析子系统**：是整个海洋牧场可视化监测系统的核心部分。主要负责数据的查看，处理，分析以及一些监测设备的控制。参与交互的角色有养殖户和数据分析人员。
3. **报警与通知子系统**：主要负责报警规则的设置和报警的通知以及后续处理。参与交互的角色有养殖户和系统管理人员。
4. **可视化展示子系统**：主要负责通过仪表盘进行定制化的可视化展示。参与交互的角色有普通注册用户，养殖户，数据分析人员以及系统管理人员。
5. **系统维护子系统**：主要负责供系统管理人员进行系统维护，可以对系统状态进行监控，操作系统数据并更新系统等。参与交互的角色只有系统管理人员。

## 3 活动图

### 3.1 活动图简介

UML 活动图是一种用来表示工作流程或业务过程中操作的图形化表示。它主要用于描述功能的执行流程，特别是那些涉及多个步骤和可能的决策路径的过程。活动图是在软件开发的需求收集和分析阶段广泛使用的，它帮助开发者和客户共同理解系统的行为。

活动图的核心组成元素包括：

- **活动节点**：表示执行的动作，通常是系统的一个操作。
- **控制流**：箭头，显示活动之间的执行顺序。
- **初始节点**：一个充满的黑色圆圈，表示流程的起点。
- **结束节点**：一个带有充满的黑色圆圈的圆环，表示流程的终点。
- **决策节点**：通常用菱形表示，用于展示基于条件的决策路径。

- **分支和合并节点**：也使用菱形表示，用于展示流程中的并行处理。
- **同步条**：用来表示并行活动的开始或结束。

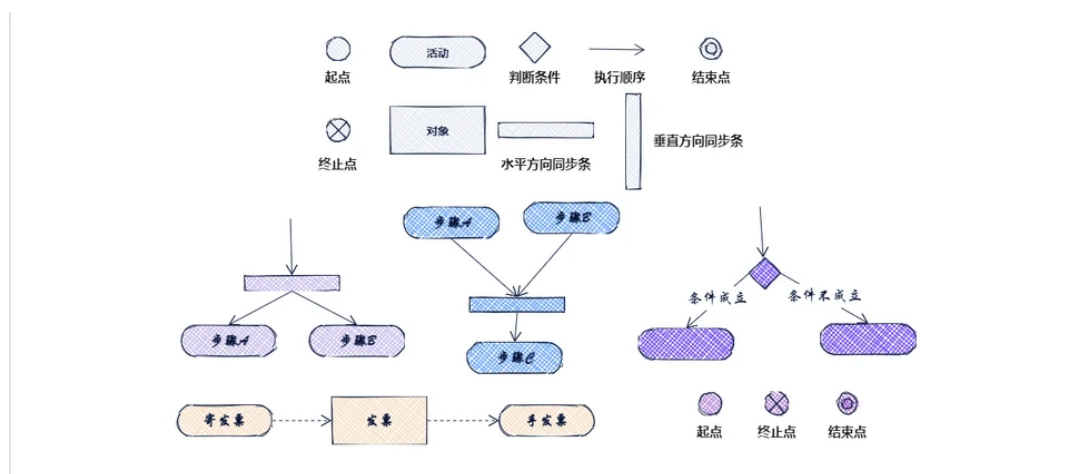


图 3.5: UML 活动图使用说明

### 3.2 活动图展示

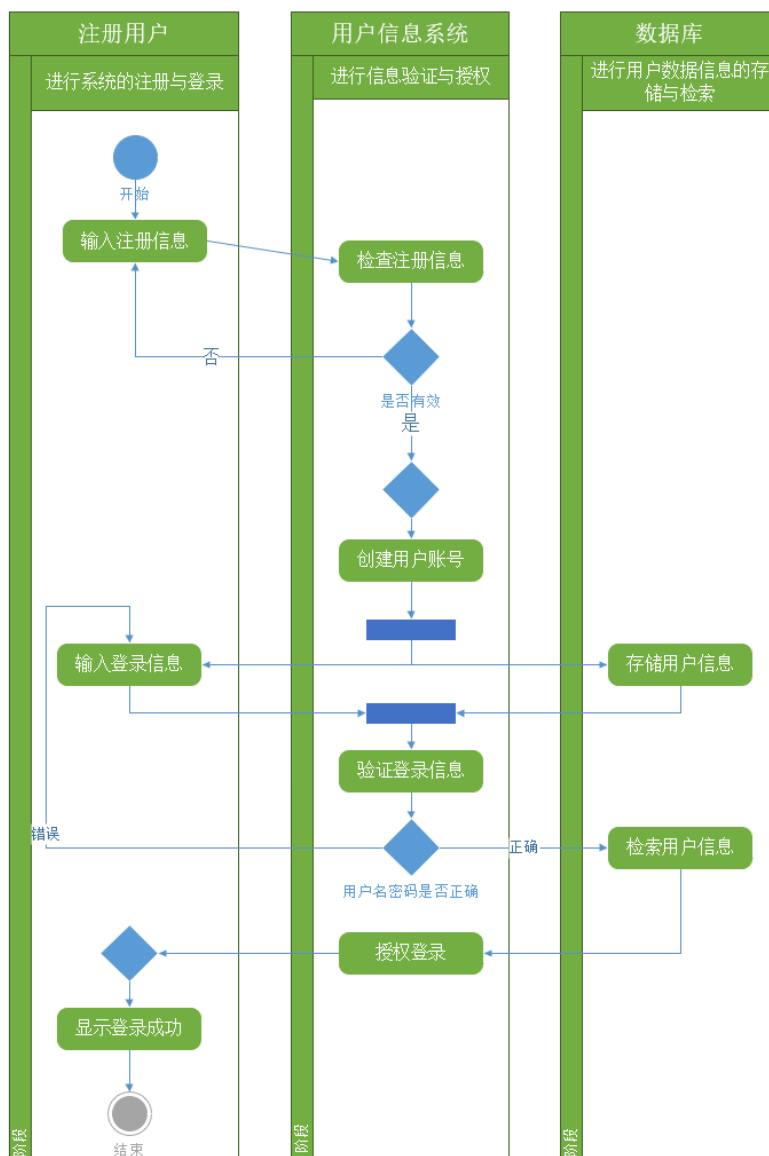


图 3.6: 用户注册和登录活动图

#### 用户注册和登录活动图阐述

关于**用户注册和登录活动图**，如图3.6所示。主要涉及到三个不同的泳道：注册用户，用户信息系统以及数据库。

活动时开始时，由外界的注册用户先发起请求，注入注册信息后，会交由用户信息系统对注册信息的有效性进行验证。若注册信息无效，则返回令其重新注册信息。有效则对控制流进行合并。因此这里使用了判断和合并的功能。

在这之后，用户信息系统创建用户账号。而后异步对控制流进行分叉，这样的目的是保证程序不会因为用户的行为而发生阻塞。一边用户信息系统传入数据库，令其存储用户信息；另一边让注册成功的用户输入登录信息。之后分支流结合。再经过一个判断和合并，验证登录信息，即用户名和密码是否匹配。若不正确，则返回重新令用户输入登录信息。若正确则对用户进行授权，分支流合

并，在用户处显示登录成功。

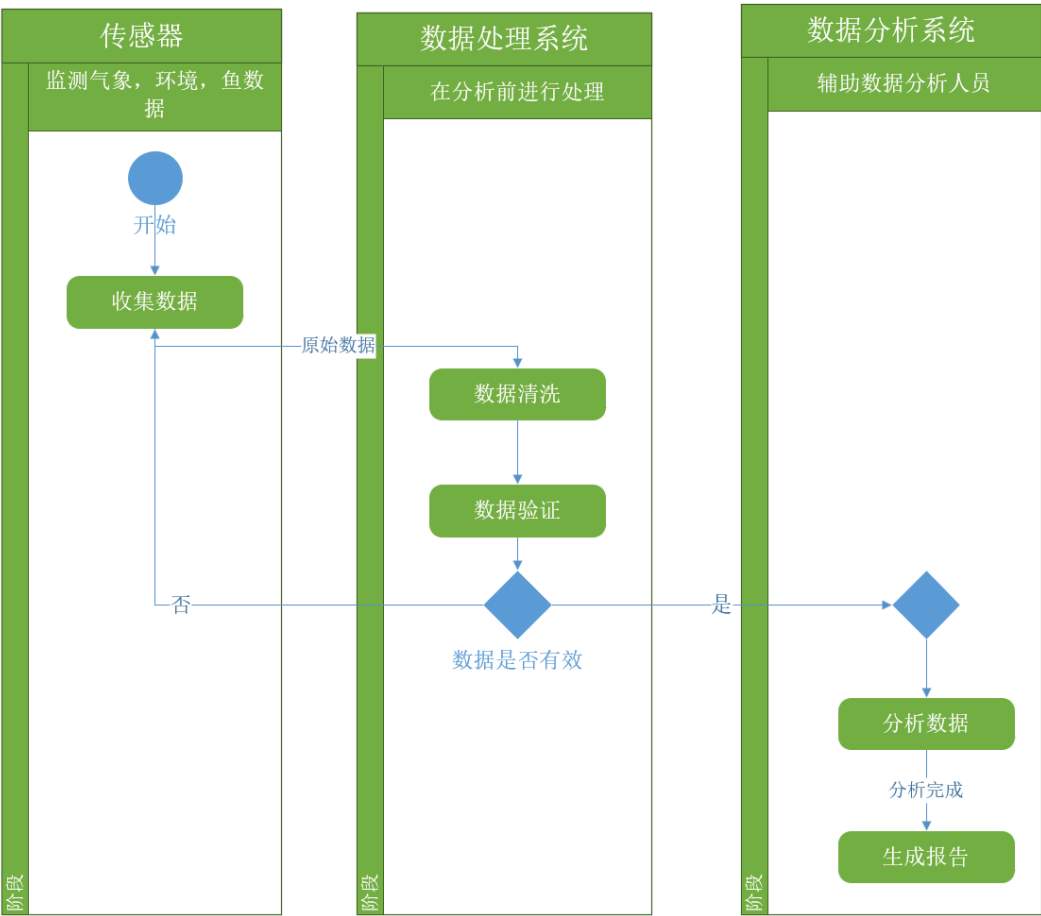


图 3.7: 数据收集和分析活动图

**用户注册和登录活动图阐述**

关于**数据收集和分析活动图**，如图3.7所示。主要涉及三个不同的泳道：传感器，数据处理系统以及数据分析系统。

首先传感器收集到对应的气象，环境和鱼群等数据后，将原始数据传入数据处理系统进行数据清洗，去除噪声和不相关信息。然后进行数据验证，以确保数据的质量和可靠性。通过一个判断分支确定数据是否有效，若数据无效，控制流继续回到收集数据的初始步骤。若数据有效，则数据流进行合并汇总，来到数据分析系统中进行数据分析识别出有意义的模式和趋势。

分析完成后，系统将生成一个详细的数据报告，供其他系统或用户进一步使用。

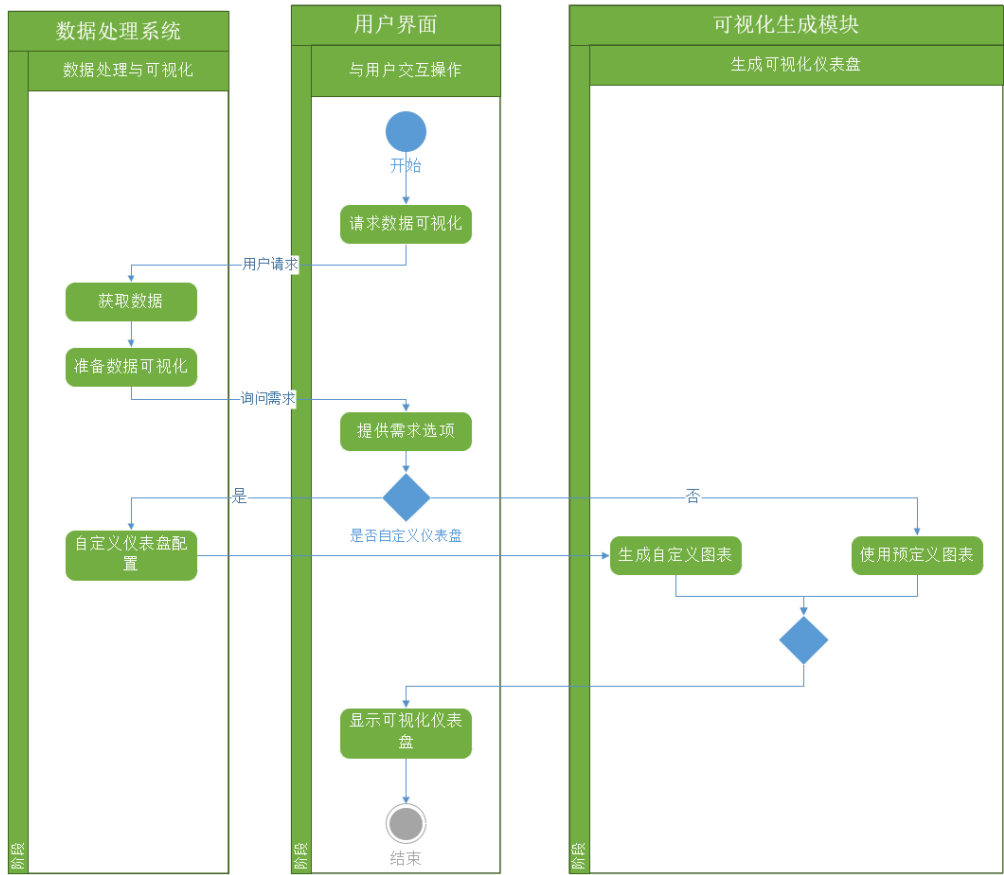


图 3.8: 可视化展示活动图

可视化展示活动图阐述

关于**可视化展示活动图**，如图3.8所示。主要涉及三个不同的泳道：用户界面，数据处理系统和可视化生成模块。

首先用户通过用户界面发起数据可视化请求。然后数据处理系统首先获取所需数据。数据准备好后，向用户发起询问需求选项，根据一个分支和合并确定根据用户选择进行标准可视化准备的预定义图表或转入自定义仪表盘配置。根据选择，流程会分叉到生成标准图表或自定义图表，最终合并回显示结果环节。

从数据获取到结果展示的整个过程，还细致地展示了用户如何根据需要定制自己的数据可视化仪表盘，从而提高了系统的交互性和用户满意度。

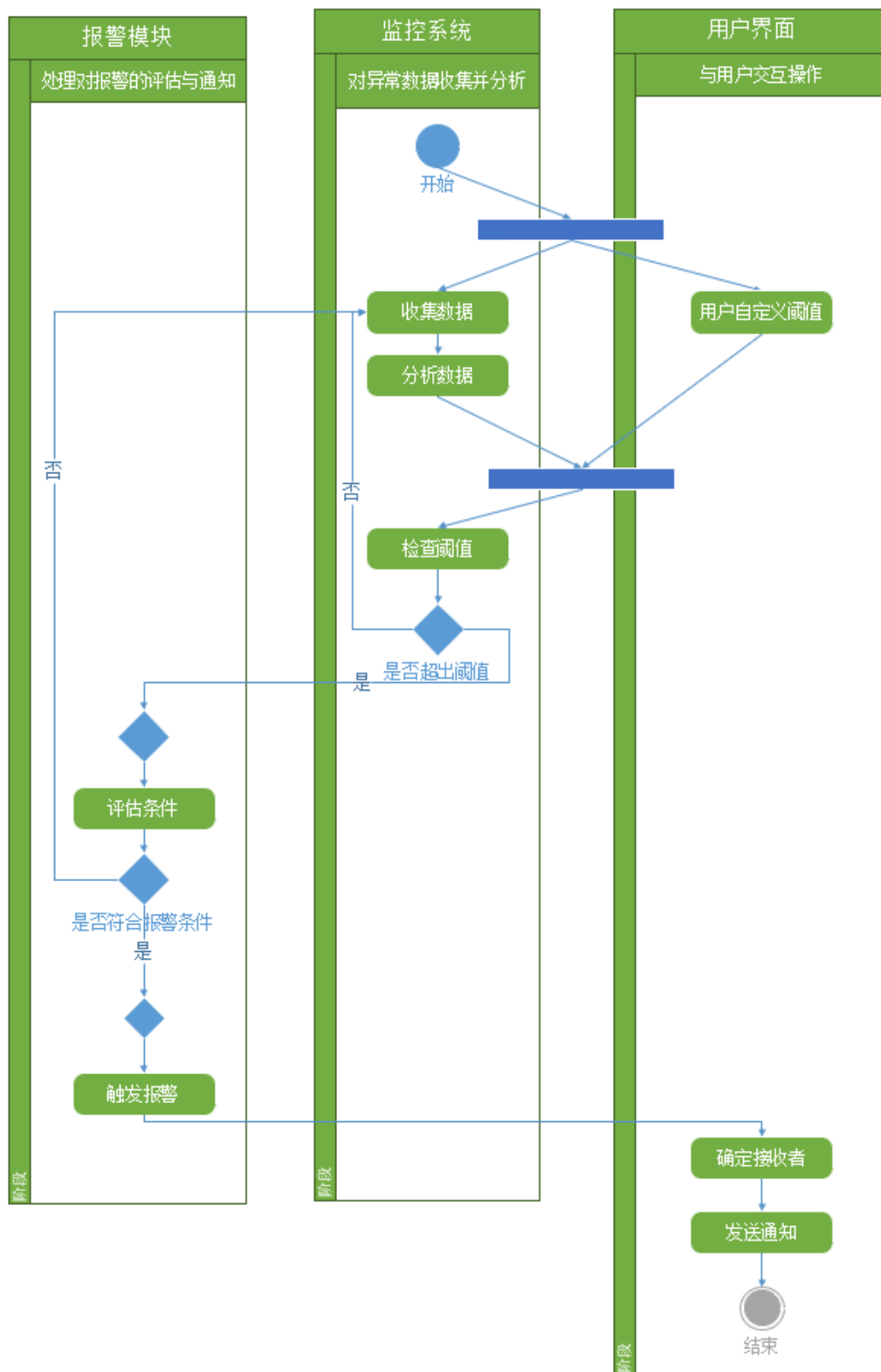


图 3.9: 报警和通知活动图

### 报警和通知活动图阐述

关于**报警和通知活动图**，如图3.9所示。主要涉及到三个不同的泳道：监控系统，报警模块以及用户界面。

首先在最开始通过异步将控制流分叉，一边进行数据的收集，一边在用户界面允许养殖户或管理员进行自定义即设置或调整报警阈值的行为。而后正常的控制流进行数据分析。数据流合并，会根据设定的阈值进行检查，以确定是否存在任何异常。

而后通过一个控制进行控制流的分支判断是否超出阈值，如果数据超出阈值，系统会进一步评估是否满足触发报警的条件。如果条件满足，将触发报警。

最后会回到用户界面，确定报警的接收者是养殖户还是系统管理员，向这些接收者发送通知，可以通过短信、电子邮件或系统内通知进行。

使用“检查阈值”和“评估条件”节点来决定是否触发报警。如果数据在正常范围内或条件不满足，流程将循环回数据收集。此活动图提供了一个清晰的视觉表示，说明了在数据超出正常运行范围时系统如何响应。这帮助确保任何潜在的问题都能迅速被识别并得到处理，从而维护海洋牧场的运行效率和安全。

## 3.3 活动图阐述

鉴于上面已经对每个子功能部分的活动图进行了详细阐述。下面对上述活动图3.6, 3.7, 3.8和3.9进行整体阐述。

1. **用户注册和登录活动图**：在这个活动图中，展示了用户如何通过不同的方式注册和登录到系统，包括与用户信息系统进行验证以及数据库对信息进行存储和检索的过程。主要涉及到三个不同的泳道：注册用户，用户信息系统以及数据库。
2. **数据收集和分析活动图**：这个活动图可以展示传感器如何收集数据，数据如何被传输至服务器，以及数据分析模块如何处理这些数据，并将结果准备供其他模块使用。主要涉及三个不同的泳道：传感器，数据处理系统以及数据分析系统。
3. **可视化展示活动图**：这张活动图展示了用户如何请求数据，系统如何从数据库中提取数据，并且如何将这些数据转换成图表或仪表盘等可视化格式呈现给用户。还包括了自定义仪表盘的地方。主要涉及三个不同的泳道：用户界面，数据处理系统和可视化生成模块。
4. **报警和通知活动图**：这张图将描述系统如何监控分析得到的数据，并在检测到异常时根据自定义的阈值或者预定义的阈值，包括进一步判断报警条件。说明了如何触发报警机制，以及如何向相关用户发送通知的过程。主要涉及到三个不同的泳道：监控系统，报警模块以及用户界面。

## 4 类图

### 4.1 类图简介

UML 类图是用于描述系统中类及其之间关系的图形表示。类图是面向对象方法中使用的最普遍的图表之一，主要用于展示系统中类的静态结构。这包括类的属性、方法以及类之间的各种关系，如关联、继承、依赖和聚合。

类图的核心组成元素包括：

- **类**：描述具有相同属性（数据元素）、操作（功能）和关系的对象的集合。

- **接口**：定义类或组件对外提供的操作，它可能由多个类实现。
- **关系**：类与类之间的连接，具体类型包括：
  - **关联**：两个类之间的结构关系，表示一类对象与另一类对象之间的连接。
  - **依赖**：一个类的改变会影响到另一个类。
  - **继承（泛化）**：表示一种分类机制，通过扩展现有类来创建新类。
  - **聚合**：表示整体与部分的关系，但整体与部分可以独立存在。
  - **组合**：一种更强的聚合关系，部分不能脱离整体存在。
- **多重性**：定义在关联关系中，一端的类可能与另一端的类关联的对象的数量。

类图不仅用于软件设计阶段详细规划系统结构，还在系统分析阶段用于确定系统需求中必要的类。通过类图，开发者可以清楚地了解系统的建构，为编码提供直接的指导。

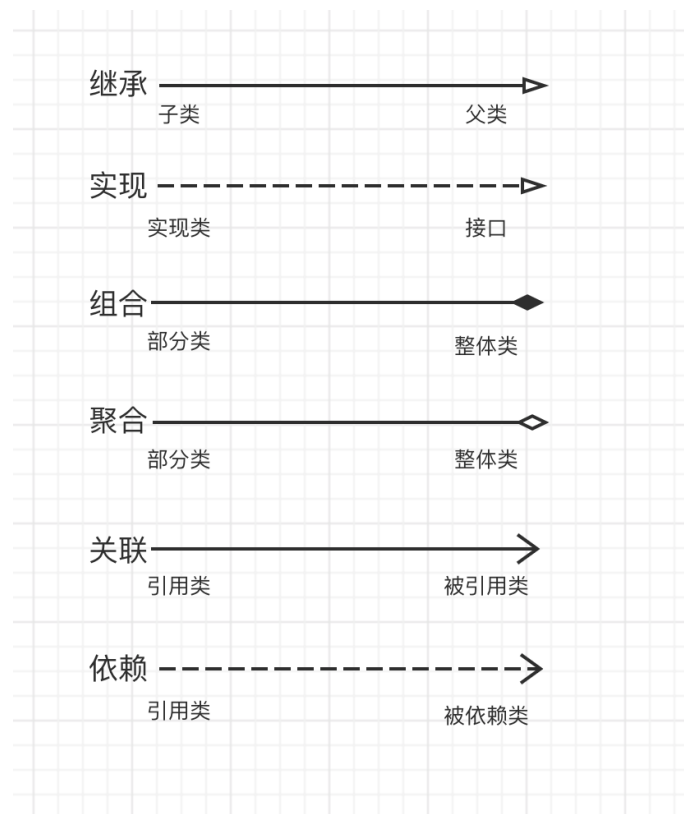


图 4.10: UML 类图使用说明



## 4.2 类图展示

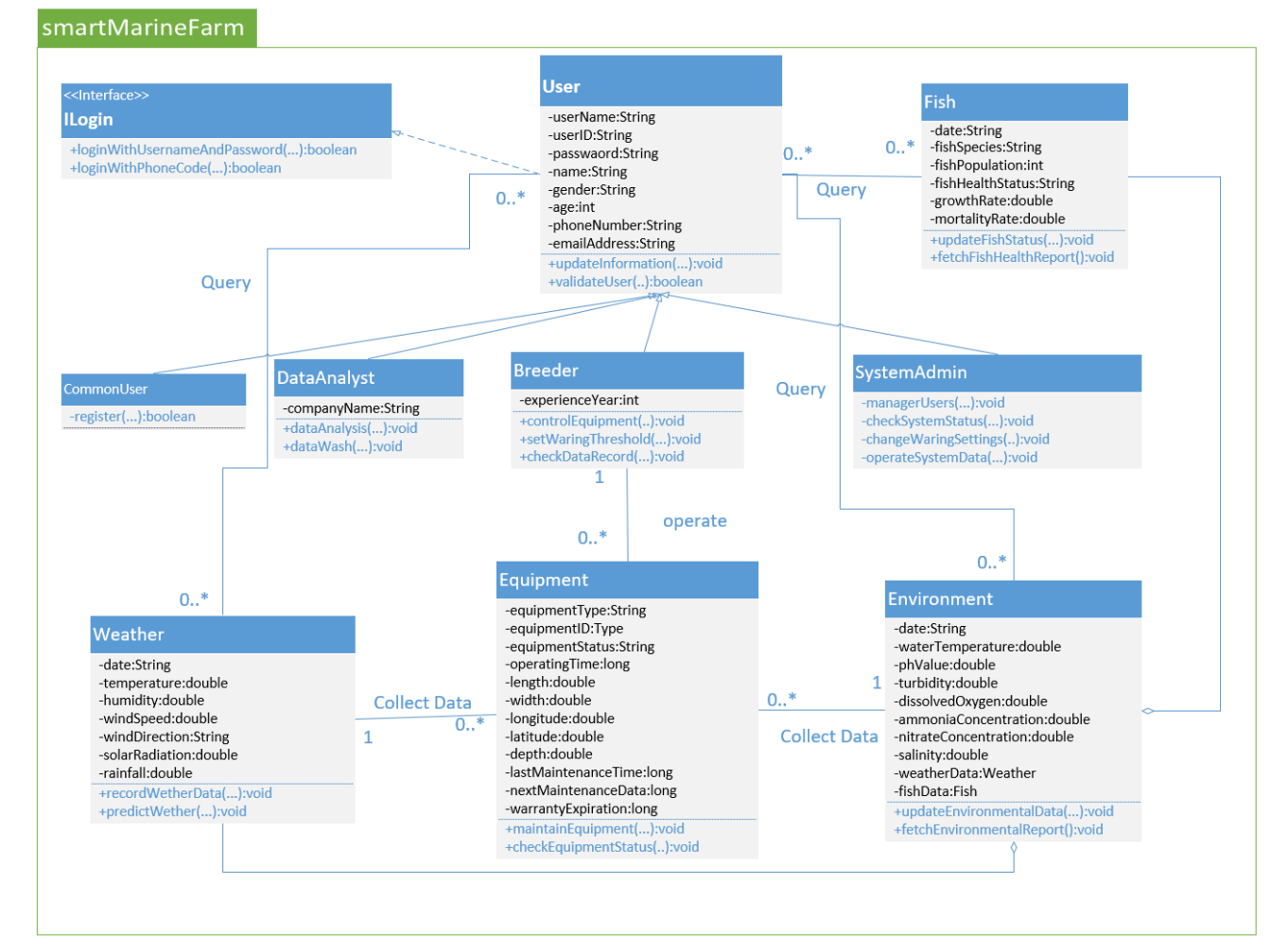


图 4.11: 海洋牧场可视化监测系统类图

## 4.3 类图阐述

如图4.11所示，展示了海洋牧场智慧可视化监测系统的类图。在本场景下，我的设计原则遵循了奥卡姆剃刀原理：如无必要，勿增实体。即尽可能用最少数量的类，简洁地完成所有海洋牧场智慧可视化监测系统稳定，安全运转所需要的功能。

### 4.3.1 类图详细解析

我是依据了该海洋牧场智慧可视化监测系统的需求文档，数据需求的数据字典部分进行的扩展。基于数据字典，我设计了如下类和方法，以实现所有系统所需的功能：（其中部分方法和详细的描述我在类图中没有写出来，在这里详细阐述重点的类以及其中的属性及方法）

1. **ILogin 类**：作为基础的登录接口。作为底层的抽象基类为上层方法提供了模板，待实现。

(1) 属性

(2) 方法：

- `boolean loginWithUsernameAndPassword(String userName, String password)`: 使用用户名与密码进行登录, 返回登录成功与否的基础方法, 并没有实现。
- `boolean loginWithPhoneCode(String phoneNumber, String code)`: 使用手机号以及验证码登录, 返回登录成功与否的基础方法, 并没有实现。

2. **User 类**: 用户为实现了 `ILogin` 相对应方法的类, 具有一些用户普遍具有的信息以及方法。作为其它用户类的父类存在。

(1) 属性

- `String userName`: 用户名
- `String userID`: 用户编号, 用户的唯一标识。保证唯一性。不同类型的用户最后的用户标识开头字母会有不同, 以此区分不用类型的子类用户。
- `String password`: 密码
- `String name`: 姓名
- `String gender`: 性别
- `int age`: 年龄
- `String phoneNumber`: 电话号码
- `String emailAddress`: 邮箱地址

(2) 方法

- `void updateInformation(int whichInfo, String info)`: 便于用户对自己的信息进行更新。`int whichInfo` 为具体对哪些用户信息进行更新。`String info` 为新增加的用户信息的描述。
- `boolean validateUser(String userId)`: 验证用户。通过唯一的用户标识 `String userId` 进行验证, 返回验证结果真或者假, 因此为 `boolean`。
- `String getUserName()` 等所有属性除了密码的公访私接口。实现公有方式访问私有变量。

3. **Environment 类**: 传感器监测到的环境数据类。可能来自不同日期。因此具体日期相当于环境类的键值, 唯一确定一个环境类。同时又是气象和鱼群的聚合类, 包含另外两个类的对象。

(1) 属性

- `String date`: 具体日期
- `double waterTemperature`: 水温
- `double pHValue`: pH 值
- `double turbidity`: 浊度
- `double dissolvedOxygen`: 溶解氧
- `double ammoniaConcentration`: 氨氮浓度
- `double nitrateConcentration`: 硝酸盐浓度
- `double salinity`: 盐度
- `Weather weatherData`: 气象数据
- `Fish fishData`: 鱼群数据

(2) 方法

- void updateEnvironmentalData(int whichInfo, String info): 更新环境数据。同样的 int whichInfo 为具体对哪项信息进行更新。String info 为新增加的信息的描述。
- void fetchEnvironmentalReport(): 获取环境报告。以输出的形式显示当前的环境报告, 包含具体日期, 水温, pH 值等等信息。

4. **Weather 类**: 传感器监测到的气象类。可能来自不同日期。因此具体日期相当于气象类的键值, 唯一确定一个气象类。

(1) 属性

- String date: 具体日期
- double temperature: 气温
- double humidity: 湿度
- double windSpeed: 风速
- String windDirection: 风向
- double solarRadiation: 太阳辐射量
- double rainfall: 降雨量

(2) 方法

- void recordWeatherData(int whichInfo, String info): 记录气象数据。同样的 int whichInfo 为具体对哪项信息进行更新。String info 为新增加的信息的描述。
- void predictWeather(int modelID, String approach, int pastDays): 预测天气的方法。int modelID 选择唯一用于预测的模型 ID, int approach 为具体预测采用的机器学习或深度学习方法, int pastDays 为具体以过去几天的天气作为参考。

5. **Fish 类**: 传感器监测到的鱼群类。可能来自不同日期。因此具体日期相当于鱼群类的键值, 唯一确定一个鱼群类。

(1) 属性

- String date: 具体日期
- String fishSpecies: 鱼种
- int fishPopulation: 鱼群数量
- String fishHealthStatus: 鱼群健康状况
- double growthRate: 生长率
- double mortalityRate: 死亡率

(2) 方法

- void updateFishStatus(int whichInfo, String info): 更新鱼群状态。参数和前面一致, 不再赘述。
- void fetchFishHealthReport(): 获取鱼群健康报告。以输出的形式显示当前的鱼群报告。同样和前面一致不再赘述。

6. **Equipment 类**: 系统的设备类, 设备编号唯一确定一个设备。包含着该设备的类型, 位置信息, 尺寸信息, 保修时间信息等。方便专业人员对设备进行监控和及时的维修, 避免意外情况发生。

(1) 属性

- String equipmentType: 设备类型
- String equipmentId: 设备编号, 唯一的标识
- String equipmentStatus: 设备状态
- long operatingTime: 运行时间
- double length: 长度
- double width: 宽度
- double longitude: 位置信息的经度
- double latitude: 位置信息的纬度
- double depth: 位置信息的深度
- long lastMaintenanceTime: 最后一次维护时间
- long nextMaintenanceDate: 下次检修时间
- long warrantyExpiration: 保修过期

(2) 方法

- void maintainEquipment(): 维护设备。
- void checkEquipmentStatus(String equipmentId): 检查设备状态。String equipmentId 为唯一的设备 ID, 输出其状态结果。

7. **CommonUser 类: 普通注册用户类, 继承自用户 User 类。**其比较特殊, 不具有任何特殊职责, 同时可以通过随意使用用户名和密码直接注册的方式形成, 其它特殊角色比如养殖户需要特殊指定, 不像是该普通注册用户一样。

(1) 属性

(2) 方法:

- boolean register(String userName, String password): 普通注册使用用户名和密码, 返回注册成功与否的结果。

8. **Breeder 类: 养殖户类, 继承自用户 User 类。**主要负责养殖场设备的操作和管理, 对报警阈值的设置以及查看一些数据记录的能力。

(1) 属性:

- String experienceYear: 表明该养殖户的养殖工作时间经历长度, 衡量其能力。

(2) 方法:

- void controlEquipment(String equipmentId): 根据设备唯一标识的设备号对设备进行操作。
- void setWarningThreshold(int newThreshold): 对报警阈值进行调整。
- void checkDataRecord(String recordId): 对某条数据记录进行查找。

9. **DataAnalyst 类: 数据分析专业人员类, 继承自用户 User 类。**主要负责对数据进行处理和分析。

(1) 属性

- String companyName: 表明公司的名字。

(2) 方法

- void dataAnalyst(): 对数据进行具体分析。

- void dataWash(): 对数据进行预处理的清洗工作, 保证后续的分析 and 处理的正确性。

10. **SystemAdmin 类: 系统管理人员类, 继承自用户 User 类。**主要负责对系统, 人员, 报警阈值等进行管理。

(1) 属性

(2) 方法

- void managerUser(String userId): 根据唯一标识用户的用户 ID 对用户进行一些操作, 包括注销, 更改信息等。
- void checkSystemStatus(): 作为系统管理人员, 对系统状态进行查看, 以进行维护和检查。
- void changeWarningSettings(): 作为高级的管理人员, 可以在更大粒度上对报警系统的一些设置进行更改。
- void operateSystemData(): 作为系统数据的管理人员, 可以对系统数据进行备份, 更新以及维护, 回溯等操作。

#### 4.3.2 类之间的关系

至于类之间的关系, 我考虑到以下几点:

- (1) **接口与实现关系:** 由于所有用户都可以通过用户名密码或者手机号登录, 因此我将这两种功能的实现抽象为一个接口 ILogin。用户类从 ILogin 类接口继承, 表明所有用户都必须实现登录功能。
- (2) **继承关系:** User 作为其它类用户的父类。最后设计四个不同的用户类 (普通注册用户、养殖户、数据分析专业人员、系统管理人员), **每个类都继承自用户类, 这表明它们共享一些基本属性和方法。并且可能有各自特定的方法和功能来满足不同的角色需求来贴合实际情况, 更好地维护系统。**也就是说 CommonUser, DataAnalyst, Breeder 和 SystemAdmin 都继承自 User 类。
- (3) **关联关系:**
  - **养殖户与设备是一对多关系:** 如前面所述, 只有养殖户有权限对设备进行控制。一个养殖户可以操作多个设备, 当然若当前没有可操作的设备, 实例数量也可以为 0。
  - **用户与环境数据、气象数据和鱼群数据通常是多对多的关系:** 由于我们的环境数据、气象数据和鱼群数据通常会在仪表盘上可视化动态显示, 因此不同用户可能会查询来自不同日期, 不同地点的数据, 因此数据可能多份。若用户查看时候数据还没有被检测出来, 也可能为 0。若此时系统没有用户, 则用户实例也可能为 0。
  - **环境数据和气象数据与设备是多对一关系:** 即来自不同日期的甚至不同地点的多组数据可以通过一个设备如传感器收集。若此时还未配备设备, 设备实例也可能为 0。
  - **环境类作为一个容器, 包含了气象和鱼群数据。**
- (4) **聚合关系:** 环境类整体大类作为一个整体包含了气象和鱼群数据, 这两种数据虽然属于环境数据, 但也可以独立存在。

## 5 顺序图

### 5.1 顺序图简介

UML 顺序图是一种用于展示对象间交互的图表，这些交互是按时间顺序组织的，用以表明对象之间如何协作完成特定的业务流程或功能。顺序图主要用于描述用例的实现细节，即展示系统的动态行为。

顺序图的关键元素包括：

- **参与者 (Actor)**：通常位于图的顶部，代表系统外部的用户或其他系统。
- **对象 (Object)**：在顺序图中表示为一个命名的矩形，是参与者交互的目标。
- **生命线 (Lifeline)**：从对象框垂直向下延伸的虚线，表示对象在时间中的存在。
- **控制焦点 (Focus of Control)**：表示对象在处理任务时的活动期，通常用窄的垂直矩形表示。
- **消息 (Message)**：在对象之间传递的信息，可以是同步消息、异步消息或返回消息，用带箭头的线表示。
- **激活 (Activation)**：表示对象执行计算的时间段，通常通过生命线上的宽条表示。
- **销毁 (Destruction)**：一个对象的生命结束，通常在生命线的底部以一个 X 标记表示。

顺序图特别适用于描述实时应用和复杂的业务流程。通过明确展示信息流和对象间的交互顺序，顺序图帮助开发者理解功能需求的实现路径和协调机制。

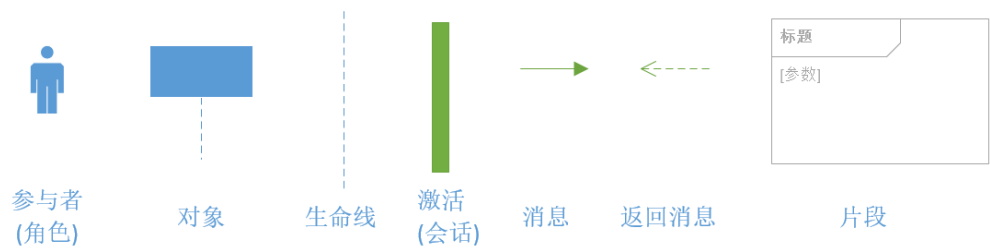


图 5.12: UML 顺序图使用说明

5.2 顺序图展示

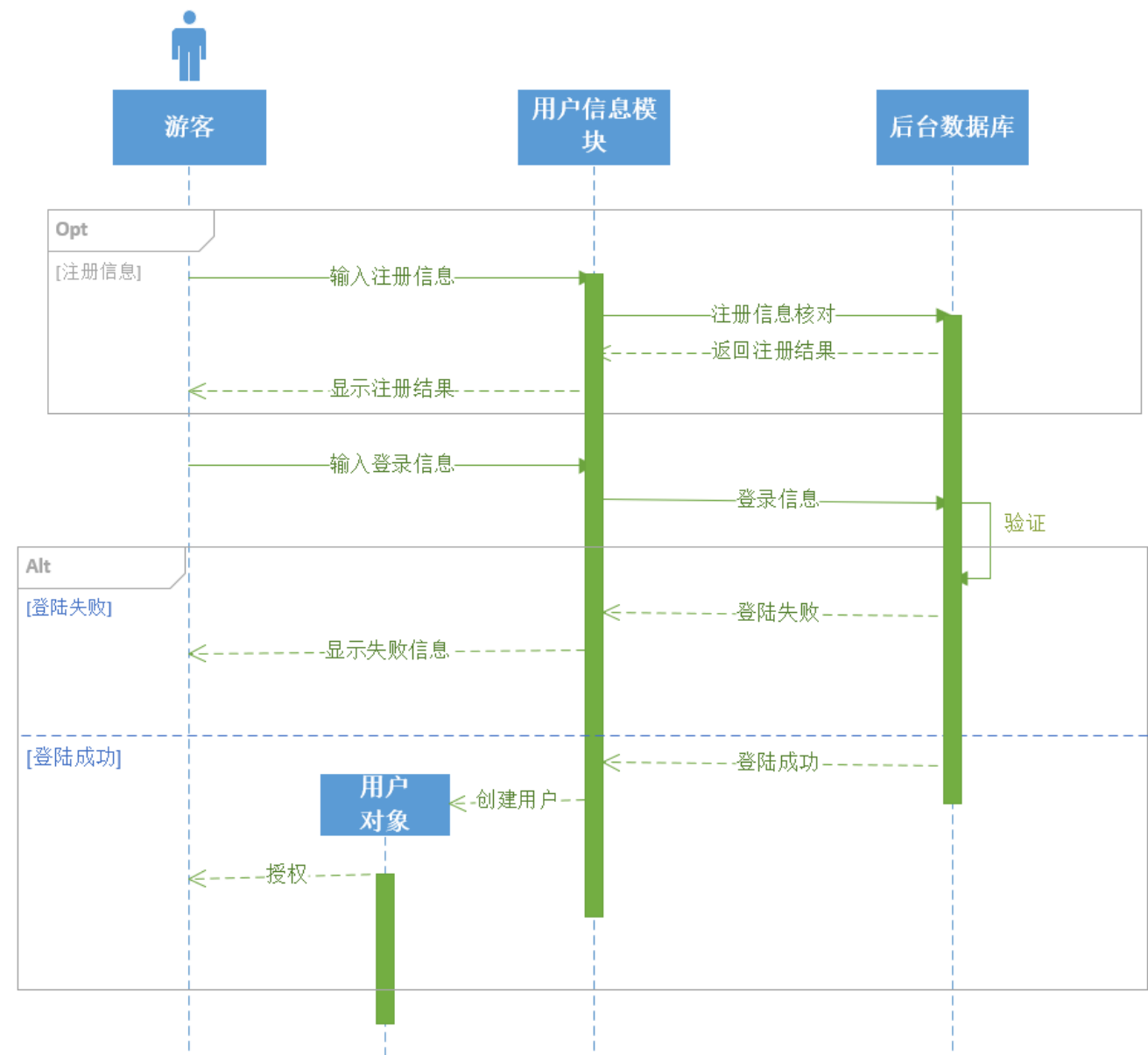


图 5.13: 用户登陆顺序图

用户登录顺序图阐述

关于用户管理顺序图，如图5.13所示，描述了新用户注册和登录的过程中与系统交互的全过程。进入系统的都是游客，与之交互的对象有用户信息模块和后台数据库。

最开始首先是可选的注册流程。如果该用户之前没有注册过，可以经过该流程。用户输入注册信息，注册信息被传入后台数据库，如果注册信息符合要求，比如用户名不会重复等要求后返回注册成功的应答，进一步返回给游客。若该过程注册失败，则返回注册失败的响应给游客。在返回了注册结果的响应给用户后，本次交互结束。

之前注册成功后再后台数据库就会保存好对应的用户信息了，包括用户名，用户 ID 和密码等。之后游客再次登陆时候，便不用经过注册，同时若之前已经注册过的用户可以直接输入登录信息。

游客先输入登录信息，用户信息模块将登录信息转发给后台数据库进行验证核对。若用户名和密码等匹配错误，验证错误，返回登陆失败的响应，最后在游客界面显示失败消息的响应。若登陆成功，创建一个新的用户对象，其生命周期继续持续下去，并为实体的真实用户的行为进行授权。

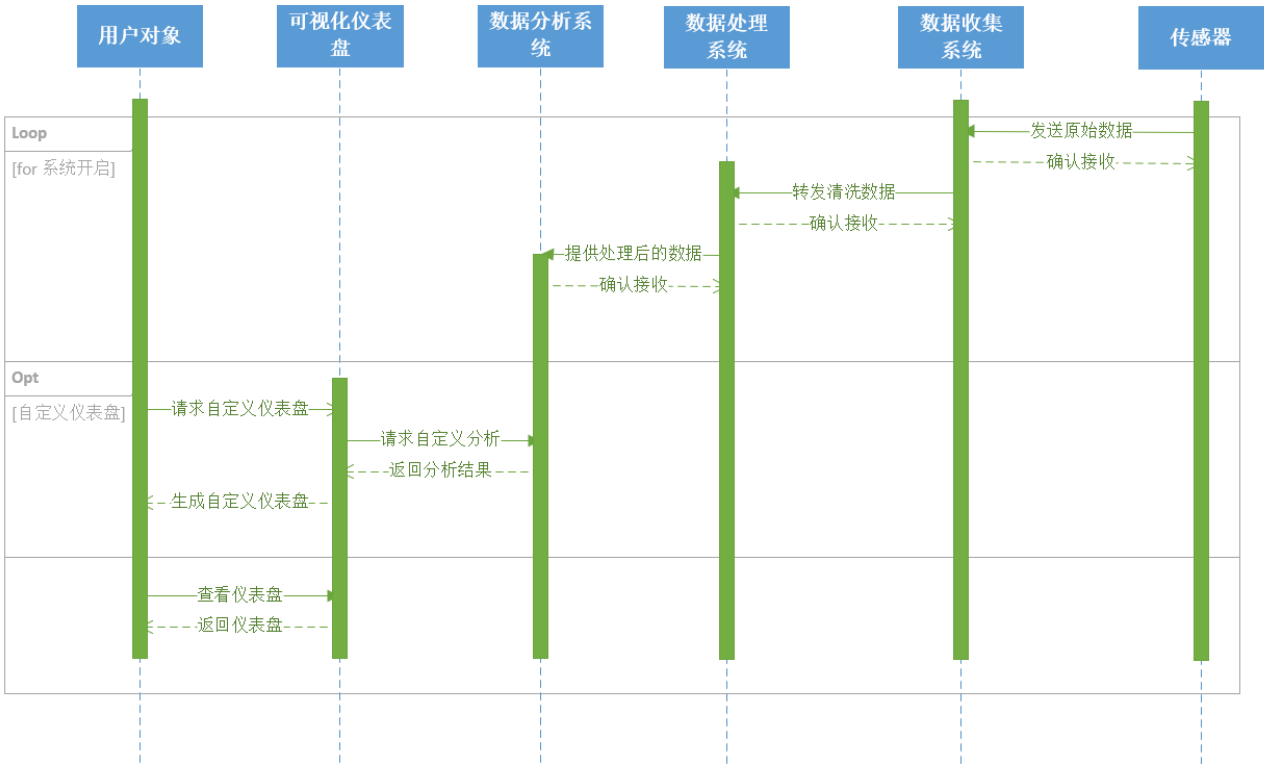


图 5.14: 数据收集与分析处理可视化顺序图

数据收集与分析处理可视化顺序图阐述

关于数据收集与分析处理可视化顺序图，如图5.14所示，描述了传感器收集数据，交给系统各个部分进行数据收集，数据清理，数据分析和处理，到最后向用户展示的过程。

参与的对象有登陆成功后创建的用户对象，传感器，数据收集，数据分析和数据可视化系统，可视化仪表盘。首先只要系统开启时，传感器就会处在一个循环中，定期监测环境中的气象，鱼群数据等，然后将原始数据交给数据收集系统，数据收集系统会汇集总结来自若干个传感器的原始数据并进行数据清洗。清洗后的数据转发给数据处理系统，数据进行特殊的处理，包括去噪声，格式化以及初步分析等。处理后的数据交给数据分析系统，数据分析系统进行深度分析以识别模式和趋势。

用户对象在此介入，若用户有自定义仪表盘的请求，则将请求发送给可视化仪表盘，仪表盘进一步向分析系统发起自定义分析请求。进行分析后将数据分析结果返回给仪表盘，仪表盘生成用户请求的仪表盘。

不过用户也可以不请求自定义仪表盘，直接请求查看仪表盘的分析结果和实时数据。返回仪表盘的可视化结果。



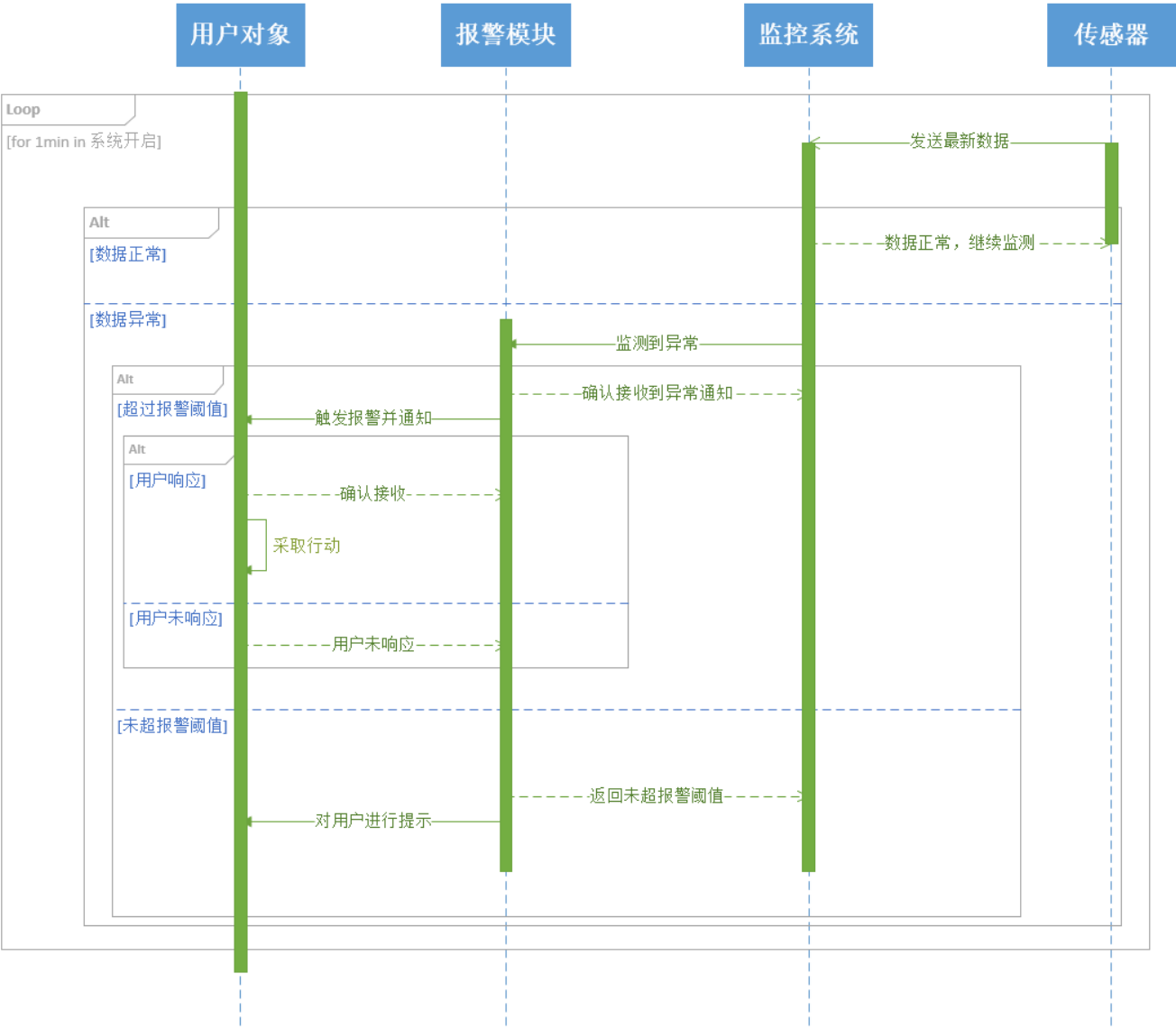


图 5.15: 报警触发和响应顺序图

报警触发和响应顺序图阐述

关于报警触发和响应顺序图，如图5.15所示，描述了传感器在系统开启后接收到最新数据后，进一步交由监控系统，报警模块，二者会进行一些条件判断后与用户进行交互。

参与的对象有登陆成功后创建的用户对象，传感器，监控系统，报警模块。首先在系统开启后，传感器会每一分钟不停地监测。监测到新的数据后传感器先将最新数据发送给监控系统。然后由监控系统进行判断，若数据正常，返回传感器响应，告知其数据正常，继续监测。

若数据异常，监控系统将监测到异常的消息发送给报警模块。然后报警模块进行判断，若异常数值超过了设定的报警阈值，则返回监控系统确认接收到异常通知，并触发报警通知用户对象。若系统接收到用户响应，则用户对象进一步采取行动。

除此之外，若异常数值还没超过了设定的报警阈值，则报警模块返回未超过监控系统的通知给监控系统，报警模块值对用户进行提示，并不触发警报。

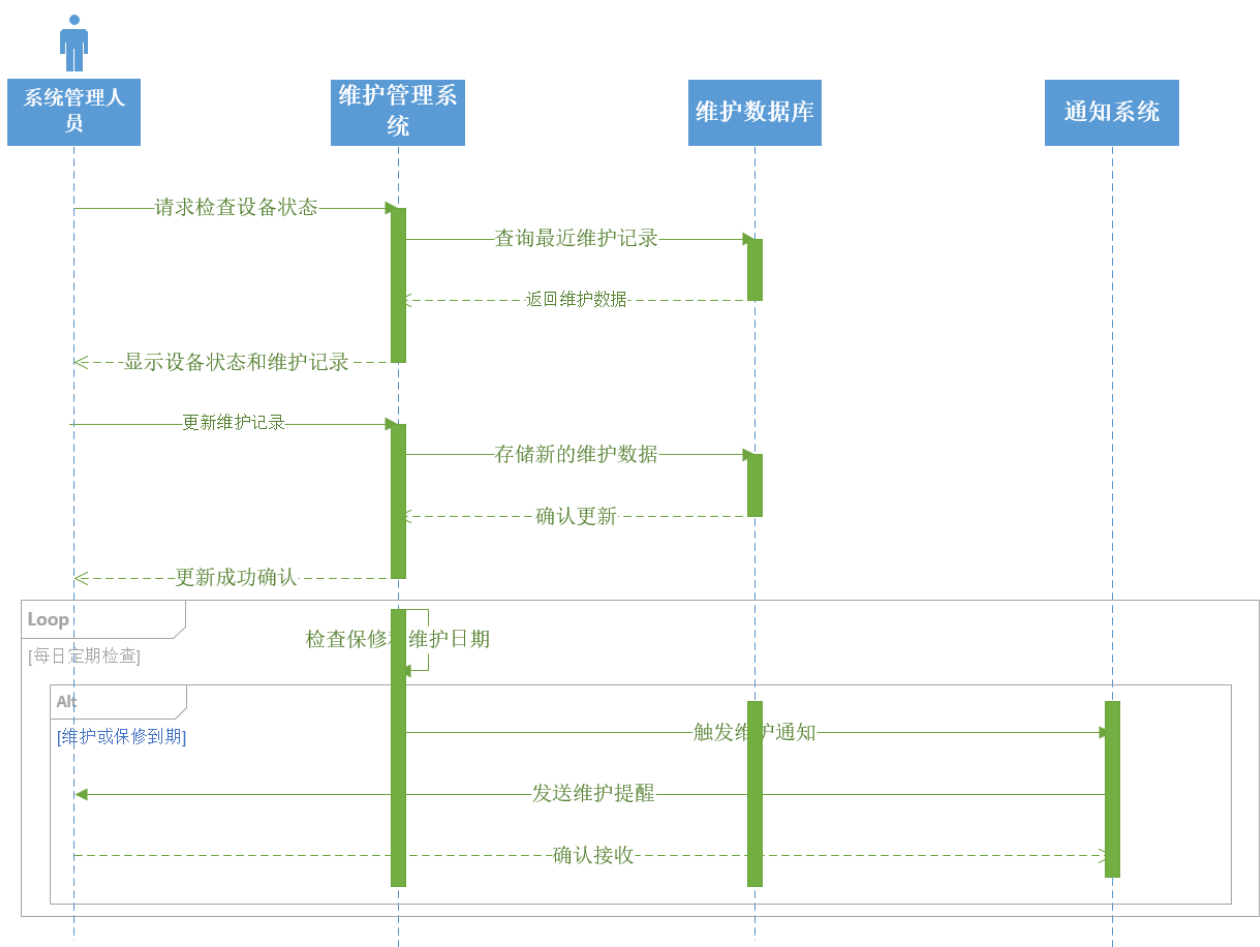


图 5.16: 系统管理设备维护和配置顺序图

系统管理设备维护和配置的顺序图阐述

关于**系统管理设备维护和配置**的顺序图，如图5.16所示。主要是**系统管理员**如何进行**设备维护**和**配置**的操作流程，详细描述了从设备检查、维护记录更新，到系统自动通知管理员关于设备保养和保修到期的整个过程。

交互开始于系统管理员向**维护管理系统**发起设备状态检查请求。系统查询**维护数据库**，返回设备的最新维护记录和状态信息给管理员。管理员根据信息更新设备维护记录，系统确认记录已更新并保存到数据库。

此外，系统中的**通知系统**每日自动检查设备的保修和维护计划。如果发现任何设备即将需要维护或保修即将到期，通知系统将自动向管理员发送提醒，确保设备得到及时的关注和处理。

管理员收到通知后，可以采取适当的维护措施，并更新系统中的设备状态，完成维护周期。这个流程不仅提高了设备的运行效率，也延长了设备的使用寿命。

5.3 顺序图阐述

鉴于上面已经对每个顺序图进行了详细阐述。下面对上述顺序图5.13，5.14，5.15和5.16进行整体阐述。

1. **用户登陆顺序图**：描述了一个实体用户注册和登录的过程中与系统交互的全过程。与之交互的对象有用户信息模块和后台数据库。

2. **数据收集与分析处理可视化顺序图**: 描述从传感器收集环境和设备数据, 如水温、pH 值、设备状态等。数据如何被系统处理和分析, 包括数据清洗和初步分析。数据分析结果如何影响决策, 如何最后通过可视化仪表盘呈现给用户。参与的对象有户对象, 传感器, 数据收集, 数据分析和数据分析系统, 可视化仪表盘。
3. **报警触发和响应顺序图**: 描述了一个特定阈值 (如溶解氧低于安全水平) 被触发的过程。系统如何处理这个信息, 包括评估情况并决定是否发送报警。报警如何被发送到相关用户 (如养殖户或系统管理员)。参与的对象有用户对象, 传感器, 报警模块和监控模块。
4. **系统管理设备维护和配置的顺序图**: 描述了系统管理员如何执行设备的常规检查和维护。如何记录和更新设备的维护日志。系统如何通知管理员即将到来的维护需求或保修到期。参与的对象有系统管理人员, 维护管理系统, 维护数据库和通知系统。

## 6 协作图

### 6.1 协作图简介

UML 协作图也称为通信图, 是用来展示对象间的交互关系及它们如何协作以实现特定行为的图表。与顺序图相比, 协作图更侧重于对象之间的关系和消息的组织, 而不是时间顺序。协作图特别适合描述系统的结构化组织, 以及消息如何在系统中传递。

协作图的主要元素包括:

- **对象 (Object)**: 表示系统中的实体, 每个对象都有一个独一无二的名称和类。
- **连接 (Link)**: 对象之间的关系, 表示两个对象可以彼此通信。
- **消息 (Message)**: 在对象之间传递的操作调用, 包括发送时间和消息顺序, 通常用带编号的箭头表示。
- **角色 (Role)**: 对象在特定交互中的功能或职责。

协作图非常有助于理解对象之间的动态协作和数据流, 使开发者能够看到系统的网络化结构及其组件如何通过消息传递来协作完成任务。

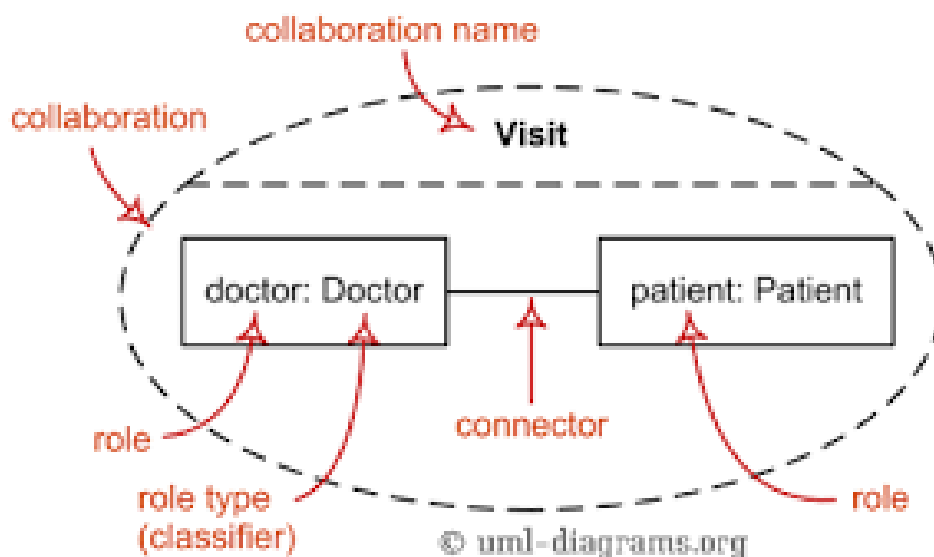


图 6.17: UML 协作图使用说明

## 6.2 协作图展示

虽然协作图可以更好的展示多个对象间的交互，但其实协作图所表达的内容与顺序图其实基本一致。因此这里对每个子系统只进行简单介绍。

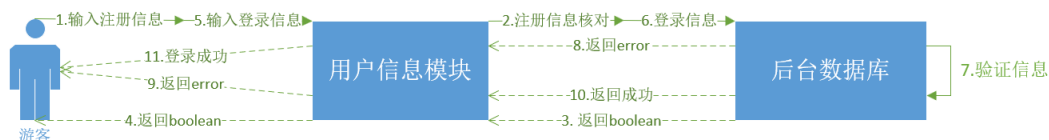


图 6.18: 用户登录协作图

### 用户登录协作图阐述

关于用户管理协作图，如图6.18所示，描述了新用户注册和登录的过程中与系统交互的全过程。进入系统的都是游客，与之交互的对象有用户信息模块和后台数据库。

本协作图绘制的基本流程与顺序图基本一致，这里不再赘述。

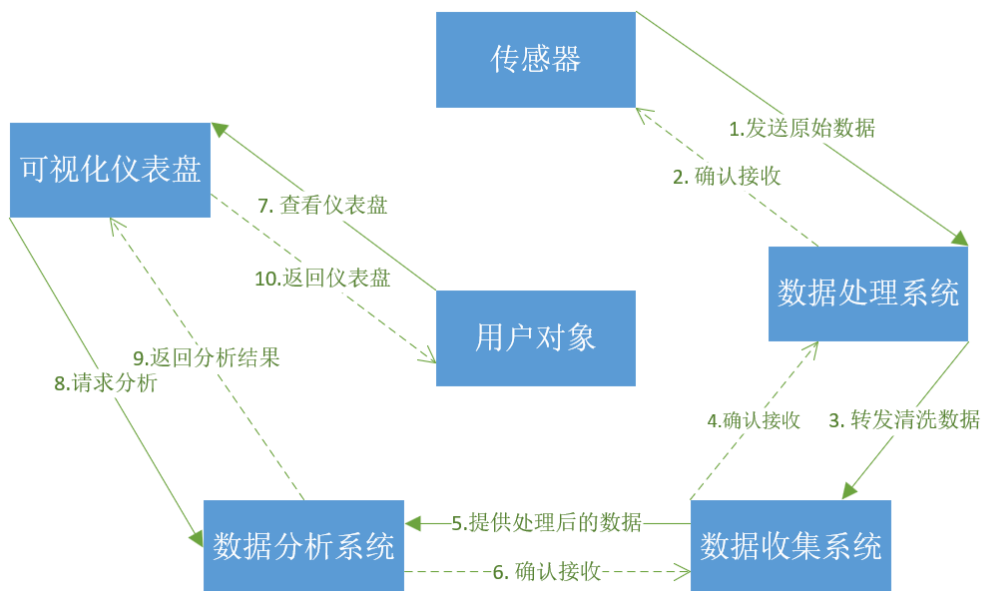


图 6.19: 数据收集与分析处理可视化协作图

### 数据收集与分析处理可视化协作图阐述

关于数据收集与分析处理可视化顺序图，如图6.19所示，描述了传感器收集数据，交给系统各个部分进行数据收集，数据清理，数据处理和分析，到最后向用户展示的过程。

参与的对象有登陆成功后创建的用户对象，传感器，数据收集，数据处理和数据分析系统，可视化仪表盘。本协作图绘制的基本流程与顺序图基本一致，这里同样不再赘述。但是由于协作图的部分局限性原因，这里没有再次呈现自定义仪表盘的功能。

其实用户对象在此介入，若用户有自定义仪表盘的请求，则将请求发送给可视化仪表盘，仪表盘进一步向分析系统发起自定义分析的请求。进行分析后将数据分析结果返回给仪表盘，仪表盘生成用户请求的仪表盘。

不过用户也可以不请求自定义仪表盘，直接请求查看仪表盘的分析结果和实时数据。返回仪表盘的可视化结果。

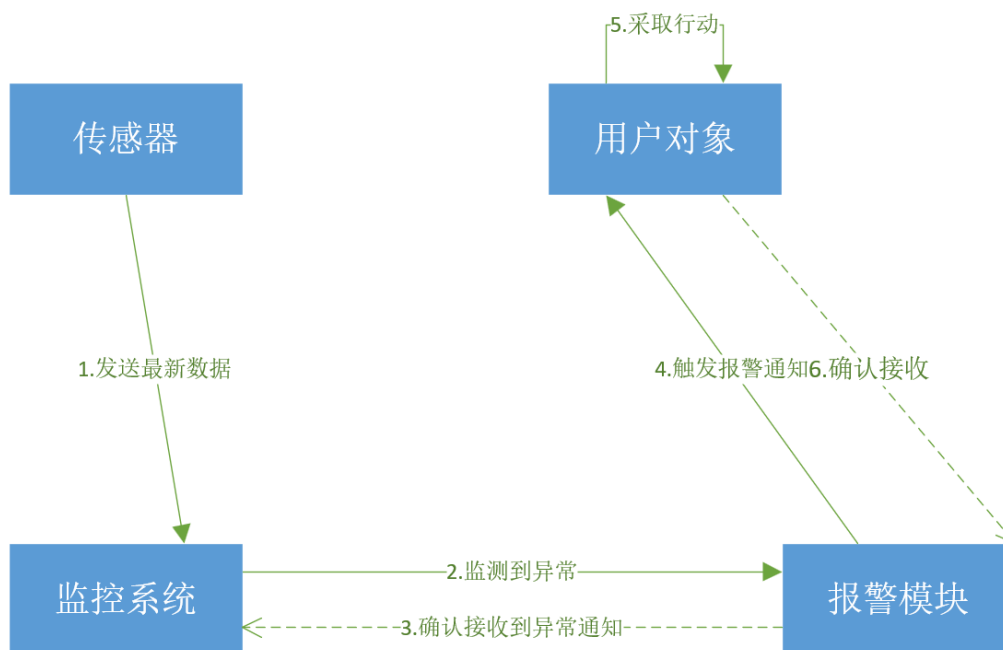


图 6.20: 报警触发和响应协作图

### 报警触发和响应协作图阐述

关于报警触发和响应顺序图，如图6.20所示，描述了传感器在系统开启后接收到最新数据后，进一步交由监控系统，报警模块，二者会进行一些条件判断后与用户进行交互。

参与的对象有登陆成功后创建的用户对象，传感器，监控系统，报警模块。本协作图绘制的基本流程与顺序图基本一致，这里同样不再赘述。

除此之外，若异常数值还没超过了设定的报警阈值，则报警模块返回未超过监控系统的通知给监控系统，报警模块值对用户进行提示，并不触发警报。

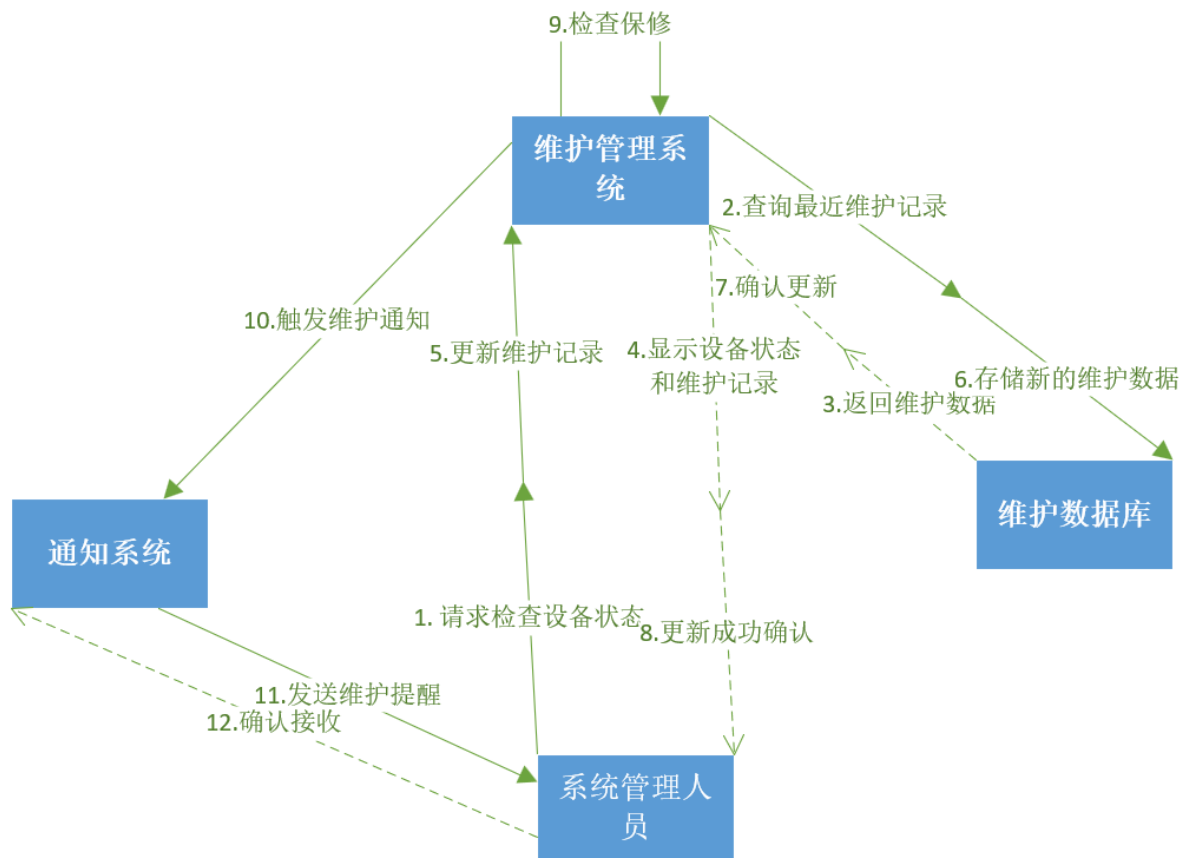


图 6.21: 系统管理设备维护和配置协作图

#### 系统管理设备维护和配置的协作图阐述

关于**系统管理设备维护和配置**的协作图，如图6.21所示。主要是**系统管理员**如何进行**设备维护**和**配置**的操作流程，详细描述了从设备检查、维护记录更新，到系统自动通知管理员关于设备保养和保修到期的整个过程。

参与的对象有系统管理人员，维护管理系统，维护数据库和通知系统。本协作图绘制的基本流程与顺序图基本一致，这里同样不再赘述。

### 6.3 协作图阐述

鉴于上面已经对每个顺序图进行了详细阐述。下面对上述顺序图6.18，6.19，6.20和6.21进行整体阐述。

1. **用户登陆协作图**：描述了一个实体用户注册和登录的过程中与系统交互的全过程。与之交互的对象有用户信息模块和后台数据库。
2. **数据收集与分析处理可视化协作图**：描述从传感器收集环境和设备数据，如水温、pH 值、设备状态等。数据如何被系统处理和分析，包括数据清洗和初步分析。数据分析结果如何影响决策，如何最后通过可视化仪表盘呈现给用户。参与的对象有户对象，传感器，数据收集，数据分析和数据分析系统，可视化仪表盘。
3. **报警触发和响应协作图**：描述了一个特定阈值（如溶解氧低于安全水平）被触发的过程。系统如何处理这个信息，包括评估情况并决定是否发送报警。报警如何被发送到相关用户（如养殖户或系统管理员）。参与的对象有用户对象，传感器，报警模块和监控模块。

4. **系统管理设备维护和配置的协作图：**描述了系统管理员如何执行设备的常规检查和维护。如何记录和更新设备的维护日志。系统如何通知管理员即将到来的维护需求或保修到期。参与的对象有系统管理人员，维护管理系统，维护数据库和通知系统。

## 7 状态图

### 7.1 状态图简介

UML 状态图也称为状态机图，是用来描述对象在其生命周期中经历的状态、状态之间的转换以及响应特定事件的行为。状态图特别适用于描述那些对象状态变化复杂的系统，如基于事件驱动的系统。它帮助开发者理解对象在特定条件下的行为变化。

状态图的主要元素包括：

- **状态 (State)：**对象在特定时间点的条件或情况。每个状态表示为一个圆角矩形。
- **初始状态 (Initial State)：**对象生命周期开始的状态，通常用一个黑色圆点表示。
- **终止状态 (Final State)：**表示对象生命周期的结束，用带有黑色填充的圆环表示。
- **转换 (Transition)：**状态之间的移动，表示为带箭头的连线。箭头指向新状态，通常包括触发事件和执行的的操作。
- **事件 (Event)：**触发状态转换的外部事件或条件。
- **动作 (Action)：**在进行状态转换时执行的活动。

通过状态图，可以清楚地看到对象如何对外部事件做出反应，并了解不同状态下可执行的操作。



图 7.22: UML 状态图使用说明







- **查看系统设备信息：**养殖和和系统管理人员可以查看设备和系统的信息。包括摄像头，传感器等设备的运行状态，维修时间等，还有系统的运行状态等。此时还可以对系统设备进行配置调整。若进行调整，则对于系统管理人员，可以对系统进行维护，对于养殖户，可以打开摄像头，控制水温等。包括对数据库同步调整完成后显示具体设备信息。若不进行更新，直接显示就设备和系统信息。
4. 退出系统：用户选择退出系统，系统会解除用户实体对象以及授权关系。用户完成需要进行的操作后，进入该状态，表示整个流程已经完成。

## 8 构件图

### 8.1 构件图简介

UML 构件图是用来描述软件系统中物理元素的静态实现视图。它主要展示了系统中软件构件的组织 and 依赖关系，以及它们如何相互连接来形成更大的系统。构件图非常适合用于模块化的系统设计，帮助开发者理解系统的软件架构和第三方组件的集成。

构件图的关键元素包括：

- **构件 (Component)：** 软件代码的物理单元，通常包含程序或库，可独立部署并且是可替换的。
- **接口 (Interface)：** 构件提供给外界的服务的说明，用圆形符号表示，并通过线连接到提供它的构件。
- **依赖关系 (Dependency)：** 表示一个构件依赖另一个构件的功能。
- **关联 (Association)：** 表示两个构件间的通信链接。
- **制品 (Artifact)：** 与构件关联的实际物理文件，如可执行文件、库文件或脚本。
- **节点 (Node)：** 表示部署构件的物理环境，如服务器或数据库。

构件图不仅帮助设计团队理解系统的构建块和它们之间的相互作用，也有助于系统管理员理解部署和维护系统所需的配置。

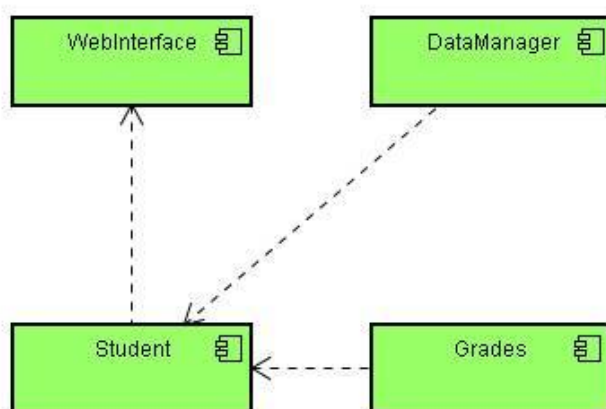


图 8.24: UML 构件图使用说明

8.2 构件图展示

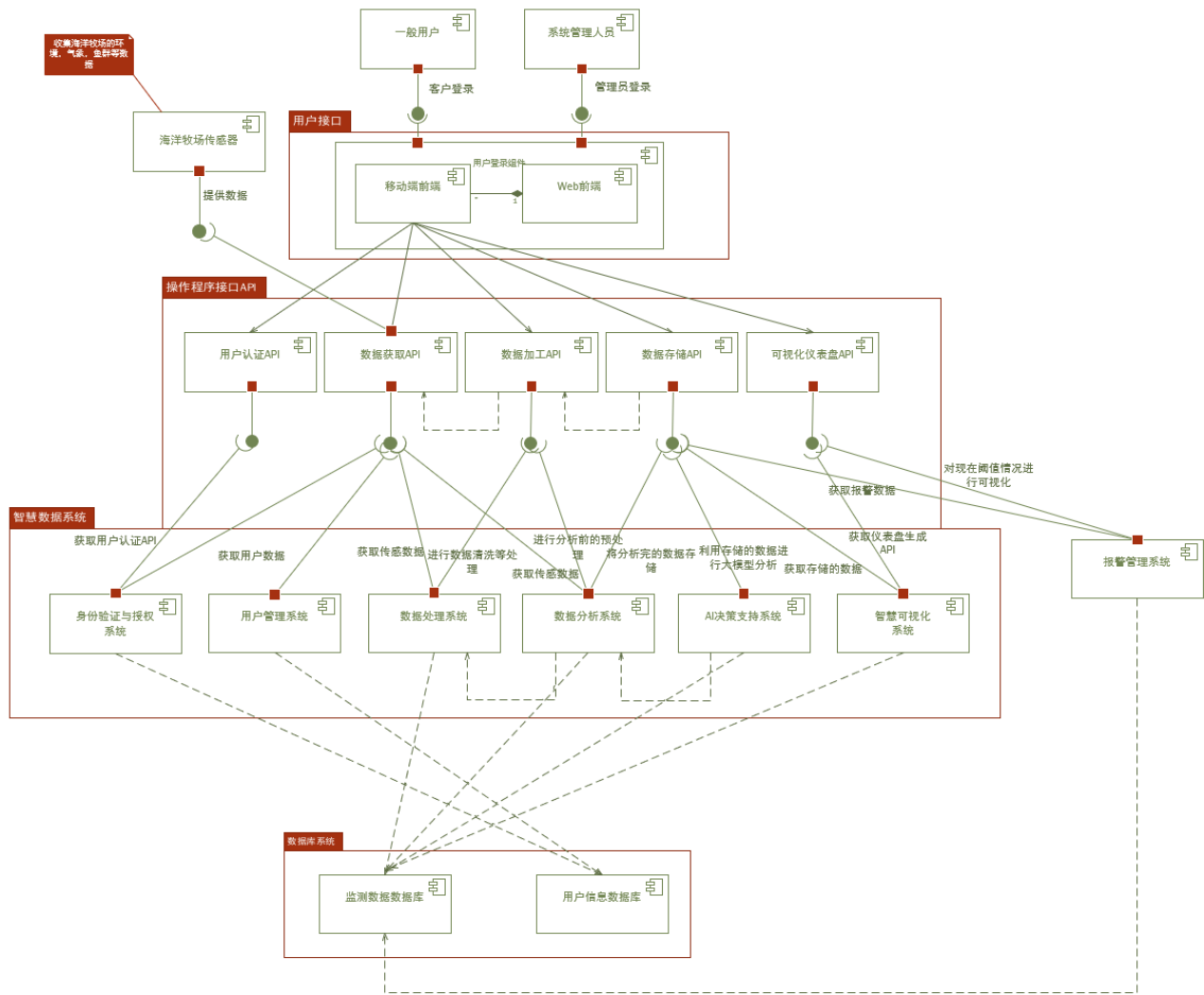


图 8.25: 海洋牧场智慧可视化监测系统构件图

8.3 构件图阐述

关于海洋牧场智慧可视化监测系统的构件图，如图8.25所示，可以放大查看细节。下面会将图构件的各个部分详细阐述：

1. 不同用户

- (1) **一般用户**：包括普通注册用户，养殖户以及数据分析专业人员。普通注册用户可查看基础的数据视图和接收通知；养殖户可以访问更详细的数据报告，进行环境控制设置；数据分析专业人员则可以进行数据挖掘，为决策提供支持。
- (2) **系统管理人员**：负责系统的日常维护，包括用户管理、系统配置、数据备份和更新等。此外，他们还负责监控系统运行状态，确保系统安全和数据完整性。

- 2. **海洋牧场传感器**：收集海洋牧场的环境、气象、鱼群等数据。这些传感器将数据实时传输到中央处理系统，为上层系统提供数据接口，以支持实时监测和环境分析。

### 3. 用户接口

- (1) **移动端前端**: 提供便捷的移动访问, 使用户能够在任何地点通过智能手机查看牧场状态, 接收警报, 和进行基本操作。
- (2) **Web 前端**: 提供更全面的视图和高级功能, 包括数据可视化、报告生成和历史数据分析等。

### 4. 操作程序接口 API

- (1) **用户认证 API**: 处理用户的身份验证和权限控制, 确保数据安全和访问控制。为上层系统提供用户认证 API 等接口。
- (2) **数据获取 API**: 允许系统获取存储在数据库或实时传输的传感器数据。为上层系统提供用户数据, 传感数据等接口。
- (3) **数据加工 API**: 用于数据清洗、格式转换和初步分析, 以供进一步处理。为上层系统提供数据清洗等处理, 以及进行分析的预处理等接口。依赖于数据获取 API 的存在。
- (4) **数据存储 API**: 管理数据的存储, 确保数据的完整性和安全。为上层系统提供对分析完数据的存储, 以及提供可视化, 报警系统, AI 大模型访问等接口。依赖于数据加工 API 的存在。
- (5) **可视化仪表盘 API**: 提供数据可视化工具, 支持动态图表和仪表板的创建, 使数据直观且易于理解。为上层系统提供阈值可视化, 智慧数据可视化等接口。

### 5. 智慧数据系统

- (1) **身份验证与授权系统**: 确保只有授权用户可以访问敏感数据和系统功能。
  - (2) **用户管理系统**: 用于管理用户账户、权限和个人信息。
  - (3) **数据处理系统**: 负责数据的接收、加工和存储。
  - (4) **数据分析系统**: 进行高级数据分析和模式识别, 支持决策制定。依赖于数据处理系统的存在。
  - (5) **AI 决策支持系统**: 使用人工智能算法提供预测分析和决策建议。依赖于数据分析系统的存在。
  - (6) **智慧可视化系统**: 根据用户需求生成定制化的可视化界面。
6. **报警管理系统**: 负责监控各种阈值和条件, 一旦检测到异常情况即时通知相关人员, 确保及时响应。依赖于监测数据数据库。

### 7. 数据库系统

- (1) **监测数据数据库**: 存储从传感器和其他数据源收集的所有监测数据。
- (2) **用户信息数据库**: 存储用户注册信息、权限设置和个人设置等。

## 9 部署图

### 9.1 部署图简介

UML 部署图是一种描述系统的物理部署的图形。它展示了软件系统的物理组件（如硬件或节点）以及这些组件上运行的软件的配置。部署图特别适用于理解系统的硬件和网络环境, 以及组件是如何在整个系统中分布的。

部署图的关键元素包括:

- **节点 (Node)**: 表示系统中的物理机器或虚拟机，如服务器、工作站或其他设备，通常用立方体符号表示。
- **构件 (Component)**: 在节点上部署的具体软件或应用程序。
- **制品 (Artifact)**: 部署在节点上的物理实体，如文件、执行程序或数据库。
- **关联 (Association)**: 节点间的通信路径，表示网络链接或数据流。
- **依赖关系 (Dependency)**: 一个节点或制品依赖另一个节点或制品的情况。

通过部署图，可以详细了解系统的硬件需求、软件部署方式及其间的通信架构。这对于系统管理员和运维团队在部署和维护阶段尤其重要。



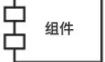
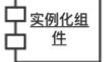
名称	解释	图例
节点&节点连线	节点(Node)是计算资源的通用名称，包括处理器和设备。在 UML 中，使用一个立方体表示一个节点。节点之间的连线表示系统之间进行交互的通信路径，在 UML 中被称为连接。	 
组件	在部署图中，组件代表可执行的物理代码模块，如一个可执行程序，逻辑上它可以与类或包对应。	 

图 9.26: UML 部署图使用说明

9.2 部署图展示

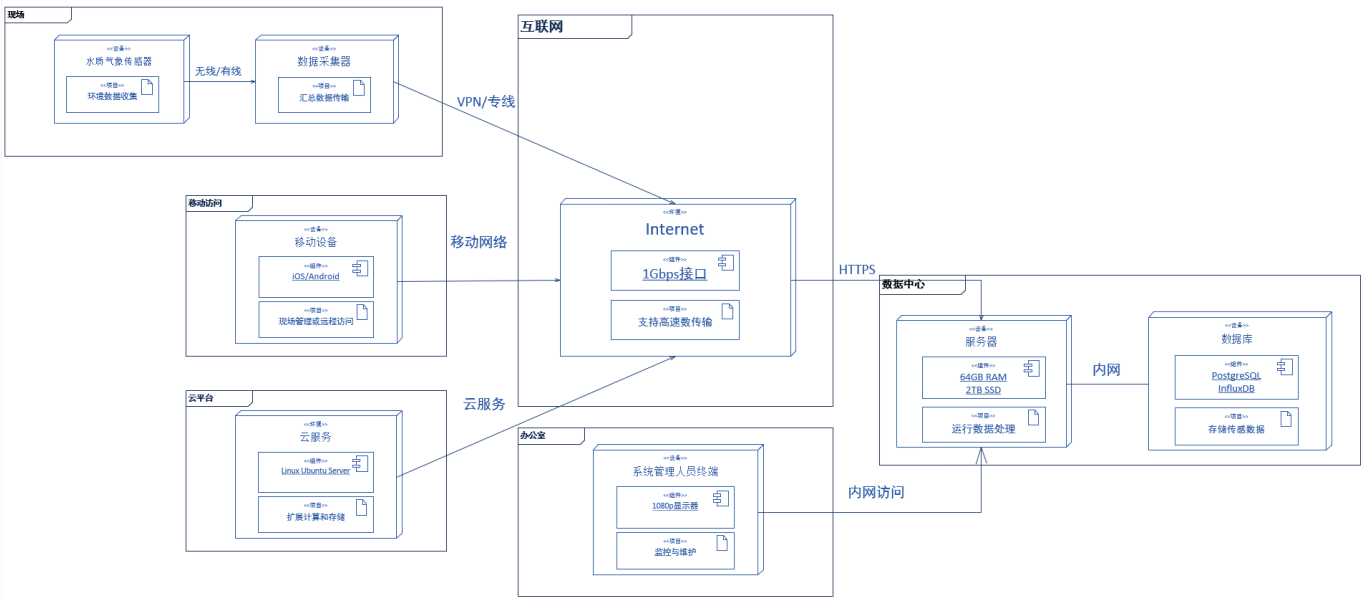


图 9.27: 海洋牧场智慧可视化监测系统部署图

9.3 部署图阐述

关于海洋牧场智慧可视化监测系统的部署图，如图9.27所示，可以放大查看细节。下面会将部署图的各个部分详细阐述：

## 1. 数据中心

### (1) 服务器

- **主要作用：**这是系统的核心，负责处理所有海洋牧场监测数据。它执行复杂的数据分析和处理任务，包括实时监控数据、预测分析以及生成报告。服务器的高性能配置允许它快速响应用户查询，并可处理来自传感器的大量数据流。
- **配置要求：**8 核心处理器，至少 64GB RAM，2TB SSD 存储。

### (2) 数据库

- **主要作用：**数据库作为系统的知识库，存储所有关键数据，包括传感器原始数据、处理后的数据以及用户信息。高性能数据库系统确保数据的快速检索和高效管理，支持高频率的数据写入操作，并保证数据的完整性和安全性。
- **推荐数据库系统：**PostgreSQL 或 InfluxDB。

## 2. 办公室

### (1) 管理员终端

- **主要作用：**作为系统管理员监控和维护系统的主要界面。终端提供了一个用户友好的界面，用于实时查看系统状态、配置系统参数、处理报警以及管理用户权限。
- **配置要求：**至少 1080p 分辨率显示器，优选大屏幕或多屏显示。

## 3. 现场

### (1) 传感器

- **主要作用：**直接部署在海洋牧场的传感器，负责收集关键的环境数据，如水温、盐度、溶解氧等。它们设计为能在恶劣条件下稳定工作，提供准确可靠的监测数据。
- **要求：**高精度，耐腐蚀性能好，适应海洋环境。

### (2) 数据采集器

- **主要作用：**位于现场的设备，作为传感器和服务器之间的桥梁，负责收集传感器数据并将其发送到中央服务器。采集器的设计要求其在低能耗下运行，同时保证数据传输的稳定和可靠。
- **要求：**高可靠性，低能耗，支持无线或有线连接。

## 4. 移动访问

### (1) 移动设备

- **主要作用：**使现场工作人员和远程用户能够通过智能手机或平板电脑访问监测系统。无论身在何处，都能实时接收数据和报警通知，以及对系统进行配置和管理。
- **配置要求：**支持 iOS 或 Android 操作系统的智能手机或平板电脑。

## 5. 互联网

### (1) 网络接口

- **主要作用：**作为数据中心和外部网络之间的连接点。它支持快速的数据传输，确保了从现场传感器到服务器，再到最终用户的数据流通畅无阻。
- **要求：**至少 1Gbps 以太网接口，推荐 10Gbps。

## 6. 云平台

### (1) 云服务

- **主要作用：**云平台提供了额外的计算资源和存储空间，以支持系统在数据量剧增时的扩展需求。
- **推荐操作系统：**Linux Ubuntu Server 最新 LTS 版本。

在网络拓扑方面，客户端与服务器通信通过 HTTPS 等加密协议实现，确保数据传输的安全性；服务器与数据库通信采用 JDBC 等高效的数据库连接协议，保障数据快速访问。

安全性方面，多因素认证，结合用户名、密码和移动设备验证实现用户身份认证；传输过程中全面采用 SSL/TLS 等加密技术，保护数据隐私；并实现访问控制。

每个组件都被精心设计，以确保整个系统的稳定、高效、可扩展和安全运行。它们共同构成了一个综合的、高度集成的监测和管理平台，能够满足海洋牧场的各项需求。

# 参考文献