

# 快递柜综合应用系统需求规格说明书

项目名称：快递柜综合应用系统

编制团队：抗渗开剥啦！

文档日期：2026-01-25

## 1. 引言

### 1.1 编写目的

本文档旨在明确“快递柜综合应用系统”的功能需求、非功能需求及接口规范。重点界定软件端（微信小程序、Web 管理后台、云服务端）的业务逻辑，以及硬件部分（基于 SysML 图纸定义的模块）与软件逻辑的映射关系，为后续的开发、集成与测试提供统一标准。

### 1.2 项目背景

本项目为软硬件结合的课程设计项目，旨在通过物联网技术解决物流配送“最后一百米”的难题。

特别说明：本项目采用“硬件逻辑全仿真，软件功能全实现”的模式。硬件部分主要体现在 MDesigner 工业设计与 SysML 系统设计中；软件端需通过数据库状态字段精准映射硬件传感器（如重量传感器、电磁锁）的反馈逻辑。

### 1.3 适用范围

- 服务端：核心业务逻辑处理、数据存储与硬件状态中转中心。
- Web 管理端：管理员对柜体状态（温湿度、占用情况）的监控与异常干预。
- 客户端（快递员/用户）：业务操作入口，触发物理逻辑变更。
- 硬件逻辑层：基于数据库状态模拟的物理反馈系统（替代真实硬件驱动）。

## 2. 总体描述

### 2.1 系统架构与建模

本系统采用分层架构设计，严格遵循 SysML 建模规范。



云端服务	系统后台	校验身份、下发开锁指令、推送短信/微信通知。
------	------	------------------------

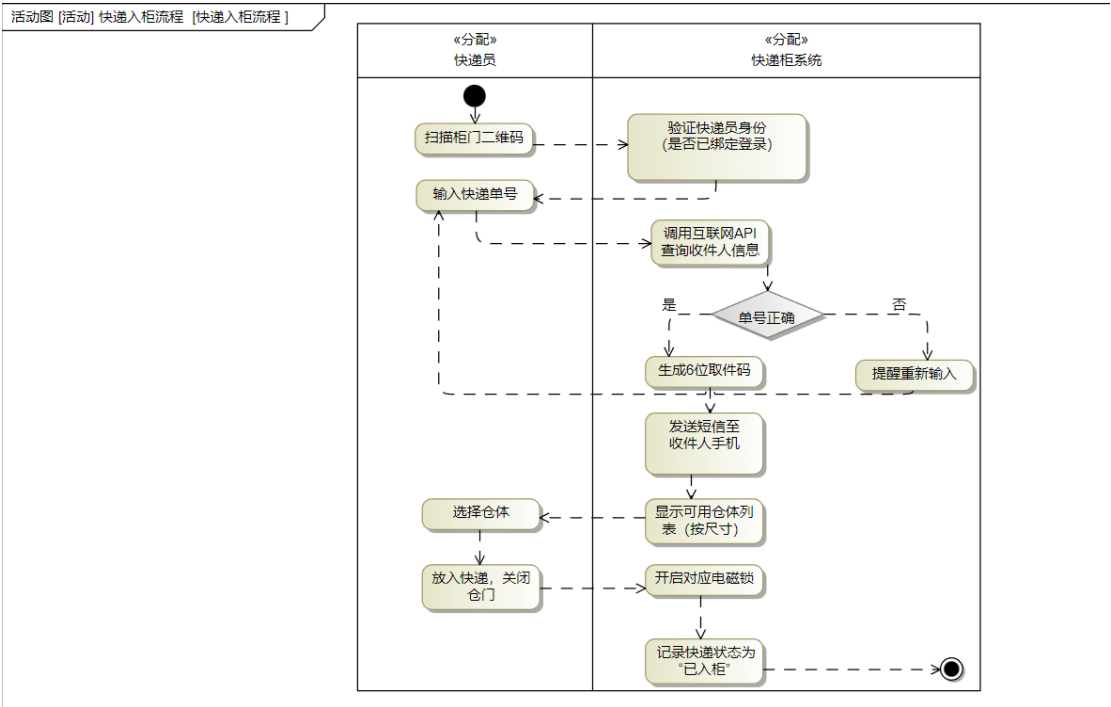
### 3. 功能需求详细说明

#### 3.1 硬件逻辑集成与传感器模拟

系统必须实现“前端操作”与“后台状态”的实时逻辑映射，以软件逻辑模拟《内部模块图》中的硬件行为。

- 柜门控制逻辑（电磁锁模拟）
  - 操作：App/人脸识别触发开门指令。
  - 反馈：App/屏幕弹出“X 号仓门已弹开”提示。
  - 状态：数据库 compartments 表 lock\_status 字段由 1 (Locked) 变为 0 (Open)。
- 物品检测逻辑（重量传感器模拟）
  - 场景：快递员投递或用户寄件。
  - 逻辑：用户在前端点击“确认放入” -> 系统模拟重量传感器信号置位 -> 后端更新 has\_item 为 1 -> 只有当检测到重量增加后，系统才允许“关门并结束订单”。
- 环境监控逻辑（温湿度传感器模拟）
  - 场景：根据《内部模块图》，部分储物格（如 1 号、2 号）配备温湿度传感器，用于药品或生鲜存储。
  - 逻辑：Web 后台需模拟周期性上传温湿度数据，并在监控大屏显示。

#### 3.2 快递员端功能



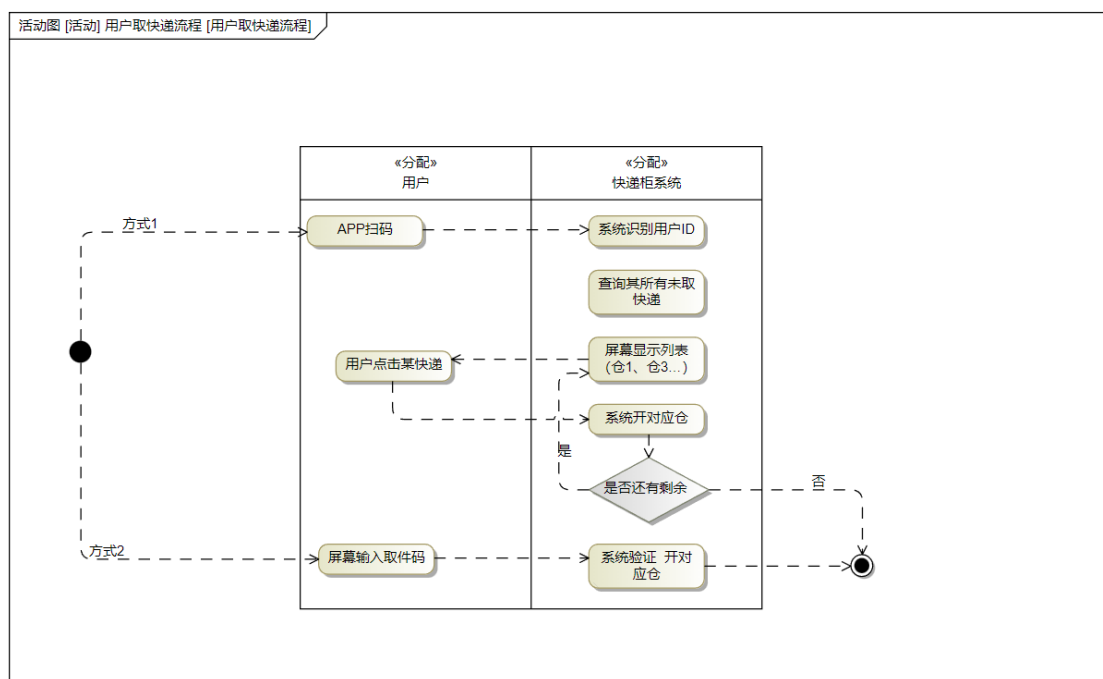
- 身份认证与登录
  - 快递员扫描柜体二维码，系统校验其是否为注册快递员。

## 2. 投递入柜流程

- 单号录入：支持扫码或手动输入快递单号。
- 联网校验：系统调用云端 API 查询收件人信息（校验单号有效性）。
- 规格选择：根据包裹大小选择大/中/小格口。
- 开门投递：系统自动弹开匹配的空闲格口。
- 关门确认：模拟“放入物品并关门”动作，系统检测到“关门信号+重量信号”后，标记包裹状态为“已入柜”。
- 通知推送：系统自动生成取件码并发送短信/微信通知收件人。

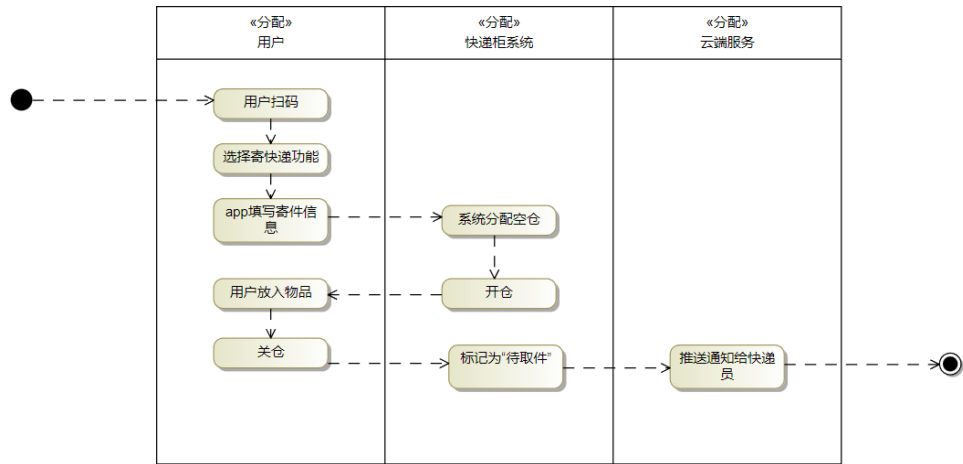
## 3.3 普通用户端功能

### 3.3.1 取快递（双模式）



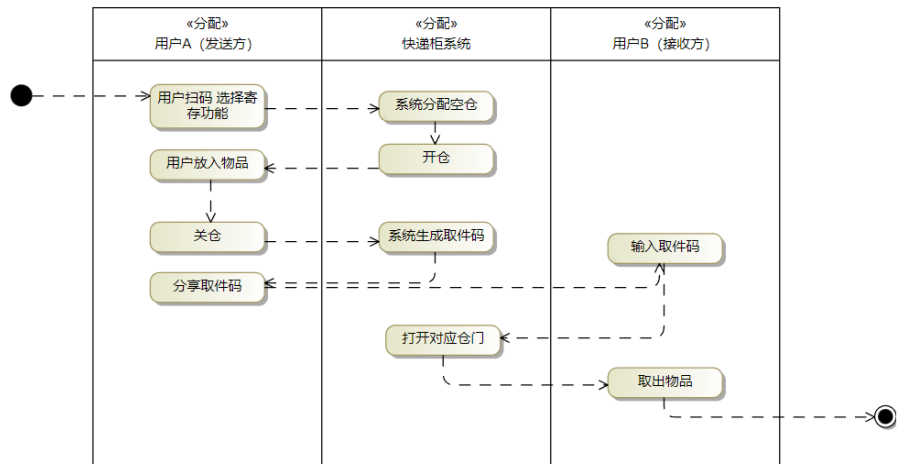
- 方式一：扫码/输码取件
  - 用户通过 App 扫描柜体码或在屏幕输入 6 位取件码。
  - 系统校验通过后，弹开对应格口。
- 方式二：人脸识别取件
  - 【插入图片：人脸识别系统活动图.png】
  - 用户选择“人脸识别开柜” -> 摄像头采集人脸 -> 云端匹配注册面部数据。
  - 匹配成功：系统查询该用户所有未取包裹 -> 列表展示 -> 用户选择特定包裹或“一键全取” -> 对应仓门打开。
  - 匹配失败：提示转为输入取件码模式。

### 3.3.2 寄快递（物流业务）



- 用户扫码 -> 填写寄件信息（地址/物品） -> 支付运费。
- 系统分配空闲格口 -> 弹开柜门 -> 用户放入 -> 关门。
- 系统通知快递员揽件（状态标记为“待揽收”）。

### 3.3.3 物品寄存（个人 P2P 业务）



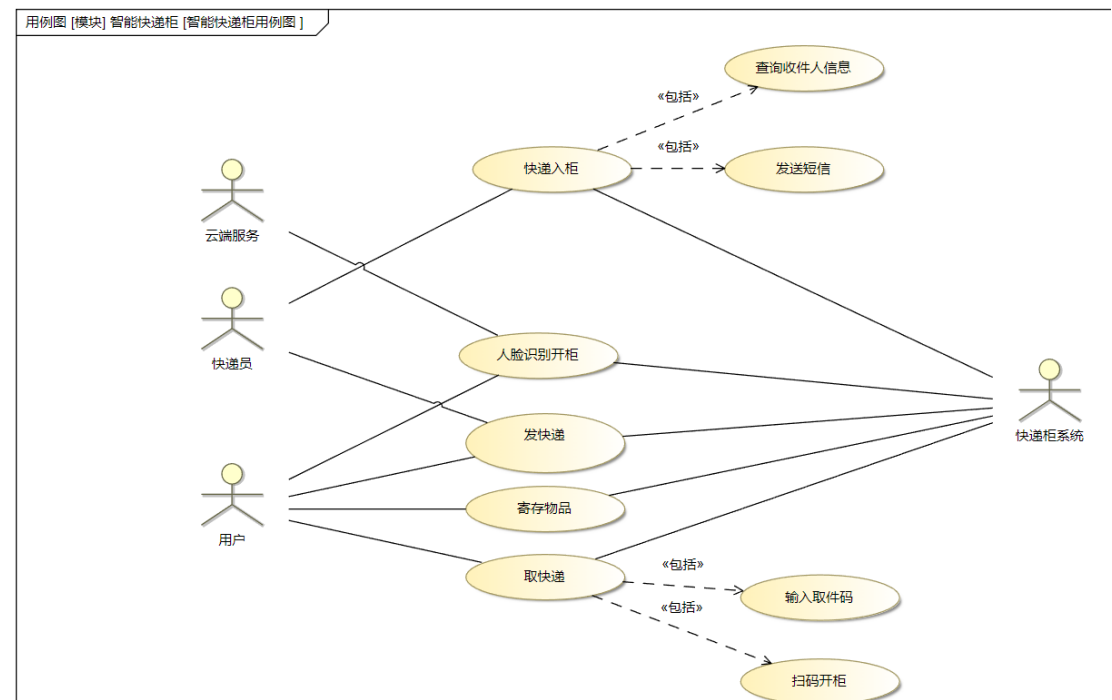
- 场景描述：用户 A 存入物品，指定用户 B（或仅生成通用码）来取，非物流流转。
- 流程：
  1. 存：用户 A 扫码选择“寄存” -> 系统分配空柜 -> A 放入物品 -> 关门。
  2. 码：系统生成唯一的“临时取件码”展示给用户 A。
  3. 享：用户 A 通过微信/口头将取件码分享给用户 B。
  4. 取：用户 B 输入该码 -> 开门取物 -> 流程结束。

## 3.4 Web 管理平台

- 全景监控：以可视化网格展示所有柜门状态（空闲/占用/故障）。
- 传感器数据：显示特殊格口的模拟温湿度数值。
- 异常处理：支持“远程一键开柜”和“状态强制复位”（应对传感器模拟误差导致的逻辑卡死）。

## 4. 数据与用例需求

### 4.1 核心用例



- 系统需完整覆盖用例图中定义的：快递入柜、发送短信、人脸识别开柜、发快递、寄存物品、取快递（含输码/扫码）等核心场景。

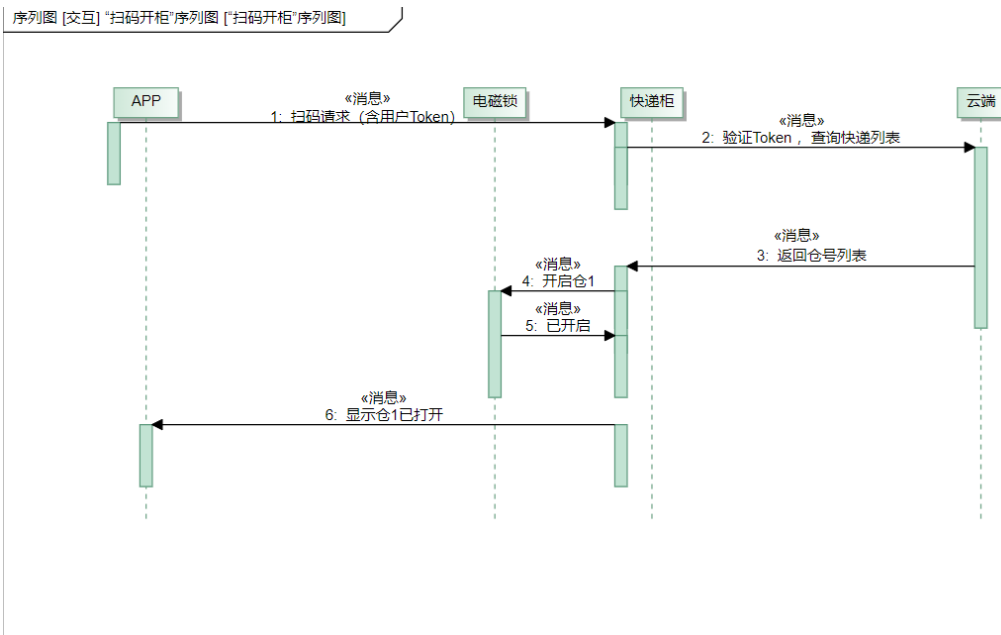
### 4.2 数据实体逻辑

基于 MySQL 8.0 设计：

1. Cabinets (柜体): 编号、位置、MAC 地址。
2. Compartments (格口):
  - type: 规格（大/中/小）。
  - has\_sensor: 是否含温湿度传感器（Boolean）。
  - status: 硬件锁状态（0=开, 1=关）。
  - storage\_state: 业务占用状态（0=空闲, 1=快递占用, 2=寄存占用, 3=寄件待揽）。
3. Transactions (流水): 记录每一次开门的操作人、时间、类型（存/取）、关联订单号。
4. FaceData (人脸库): 存储已注册用户的面部特征值（模拟哈希值或图片 URL）。

## 5. 接口与交互流程

### 5.1 扫码开柜交互时序



本系统核心交互遵循以下标准协议（RESTful API + JWT）：

1. App 端：发起 POST /api/unlock，携带 {user\_token, qr\_code\_id}。
2. 云端：
  - 验证 Token 有效性。
  - 查询该用户关联的待取订单。
  - 向柜体发送指令（模拟 MQTT/TCP 消息）。
3. 柜体端：
  - 接收指令 -> 驱动电磁锁（更新数据库状态）。
  - 返回消息：{status: "opened", cabinet\_id: 1}。
4. App 端：接收回调，界面显示“1 号仓已打开”。

## 6. 开发计划与交付物

### 6.1 开发环境

- 服务端：Java Spring Boot
- 客户端：微信小程序 (Uni-app / Native)
- 数据库：MySQL
- 设计工具：MDesigner (硬件结构)

### 6.2 交付清单

1. 系统原型：包含快递员端、用户端、Web 后台的可运行代码。
2. 图纸集：
  - MDesigner 组件构成图与系统集成图。
  - UML/SysML 完整建模图纸（即本文档引用的所有图片）。
3. 测试报告：重点测试“人脸识别”模拟逻辑的连通性，以及“寄存功能”中取件码流转的闭环测试。