**系统设计说明书**

目录

[1 引言 2](#_Toc193205784)

[1.1 目的 2](#_Toc193205785)

[1.2 范围 2](#_Toc193205786)

[2 系统设计 2](#_Toc193205787)

[2.1 概念设计 2](#_Toc193205789)

[2.2 系统架构 3](#_Toc193205790)

[2.3 分层设计说明 5](#_Toc193205791)

[3 功能模块设计 5](#_Toc193205792)

[3.1 功能总览 5](#_Toc193205794)

[3.2 功能模块图 6](#_Toc193205795)

[3.3 登录与认证模块 6](#_Toc193205796)

[3.4 任务管理模块 8](#_Toc193205797)

[3.5 系统流程图 27](#_Toc193205798)

[4 数据库设计 28](#_Toc193205799)

[4.1 逻辑设计 28](#_Toc193205801)

[4.2 物理设计 29](#_Toc193205802)

[4.3 数据库设计原则 30](#_Toc193205803)

[4.4 数据库设计 30](#_Toc193205804)

[5 接口设计 33](#_Toc193205805)

[5.1 用户接口设计 33](#_Toc193205807)

[5.2 系统接口设计 34](#_Toc193205808)

[5.3 外部接口 34](#_Toc193205809)

[6 安全性设计 34](#_Toc193205810)

[6.1 用户信息安全 34](#_Toc193205812)

[6.2 数据安全传输 34](#_Toc193205813)

[6.3 系统安全 34](#_Toc193205814)

[7 易用性与维护性 35](#_Toc193205815)

[7.1 易用性设计 35](#_Toc193205817)

[7.2 系统维护 35](#_Toc193205818)

## 引言

### 目的

本文档旨在对医院运送管理系统进行概要设计说明，明确系统的整体架构、主要功能模块、数据流、接口设计、安全性与性能要求等。文档面向开发、测试、维护和项目管理人员，确保项目各阶段在同一设计指导下顺利实施，同时便于后续系统扩展与维护。

### 范围

本系统主要用于大中型医院内部的物料运输管理，涵盖以下主要业务流程：

* 任务发起：医护人员通过护士端发起运输任务，并制定交接节点；
* 任务智能分派：系统根据任务优先级及运送员状态自动分派任务；
* 任务接单、启动与交接：运送员接单后，通过扫码、拍照完成任务启动、交接及完成操作；
* 任务执行跟踪：系统实时采集定位数据，更新运输路径及任务状态，反馈异常；
* 任务记录查询：管理员通过管理端检索、查询任务执行详情；
* 系统管理：管理员对用户账号、任务数据、系统配置等进行管理。

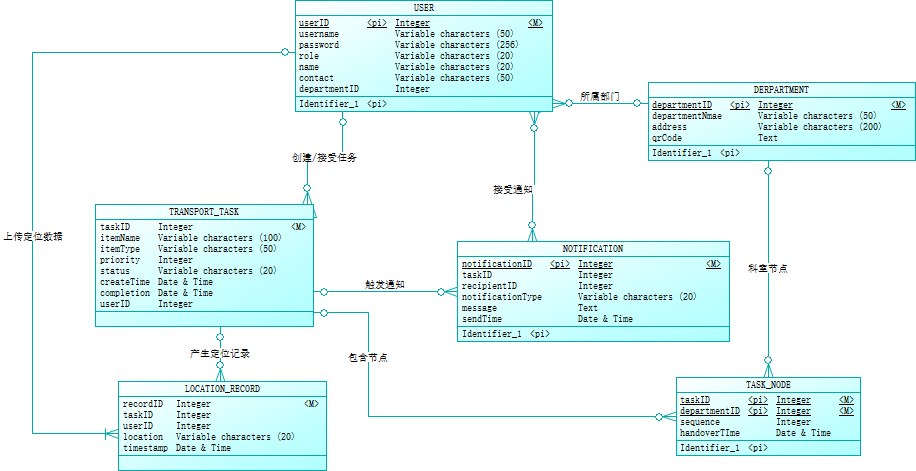
系统采用B/S架构，支持移动端（微信小程序/HTML5移动浏览器）和PC端访问。

## 系统设计



### 概念设计

图1描述了本系统中的主要数据实体及其相互关系，共包含6个主要实体：用户、运输任务、任务节点、定位记录、通知和地点。该ER模型也涉及到了实体间的关系，主要有一对一、一对多和多对多关系。



**用户与运输任务（一对多）：**一个用户（如医护人员）可以创建或分派多个运输任务，即“一对多”关系；运送员接受一个或多个运输任务，与其形成“一对多”关系。

**用户与定位记录（一对多）**：用户（主要是运送员）上传定位数据时，其记录与该用户关联。

**用户与通知（多对多）**：所有用户均可能接收到系统推送的通知消息，一个用户可能受到多条通知，一个通知可能推送给多个用户。

**运输任务与任务节点（一对多）：**每个运输任务可能包含多个交接节点。

**运输任务与定位记录（一对多）：**每个任务在执行过程中会产生多条定位记录，用于跟踪运输路径和任务状态。

**运输任务与通知（一对多）：**任务状态的变更会触发多个通知，系统将这些通知推送给相关用户。

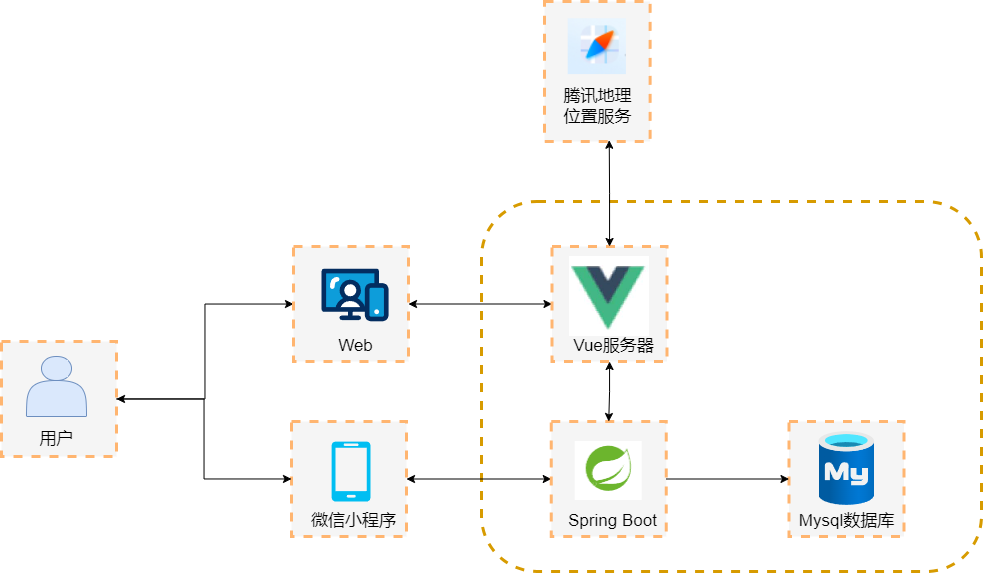
**任务节点与科室（多对一）：**每个交接节点都对应一个具体的地点信息，通过二维码扫描确认该节点位置，一个地点可以出现在多个任务节点中。



### 系统架构

本医院运送管理系统采用前后端分离的架构，前端分为微信小程序端和Web端，医护人员和运送员主要通过微信小程序访问系统，管理员可通过Web网页端访问系统，Web端采用Vue框架进行搭建，微信小程序采用uni-app框架进行搭建。

后端采用Spring Boot构建业务逻辑，通过Tomcat服务器部署，利用RESTful API和WebSocket实现前后端实时数据交互；



系统使用MySQL作为主要数据库存储信息，并结合缓存技术满足高并发需求，同时集成第三方定位服务和地图API，实现运送员实时定位和路径可视化展示。具体说明如下：

* **客户端层**：
  + 移动端：包括护士端和运送员端，通过微信小程序和HTML5移动网页实现任务操作、扫码拍照、定位上传等功能；
  + PC端：主要面向管理员，提供系统管理、任务监控、统计分析等功能。
* **应用服务器层**：
  + 采用Spring Boot架构，提供认证、任务管理、任务分派、任务跟踪、通知推送、系统管理等核心服务。
  + 通过RESTful API实现前后端数据交互，确保实时性和安全性。
* **数据存储层**：
  + 使用关系型数据库（MySQL）存储用户信息、任务数据及日志；
  + 采用Redis等缓存机制加速热点数据访问；
* **外部接口层**：
  + 集成第三方定位服务（如腾讯地图API）用于定位与地图展示；

### 分层设计说明

 表现层（客户端）：负责与用户交互，提供友好的操作界面与实时反馈。

 业务逻辑层（应用服务器）：各服务模块间分工明确，通过松耦合设计实现任务流程管理、调度及状态更新。

 数据访问层（数据库）：确保数据持久化存储，支持高并发数据访问与查询。

 外部接口层：用于对接第三方服务，提供定位、地图展示等功能。

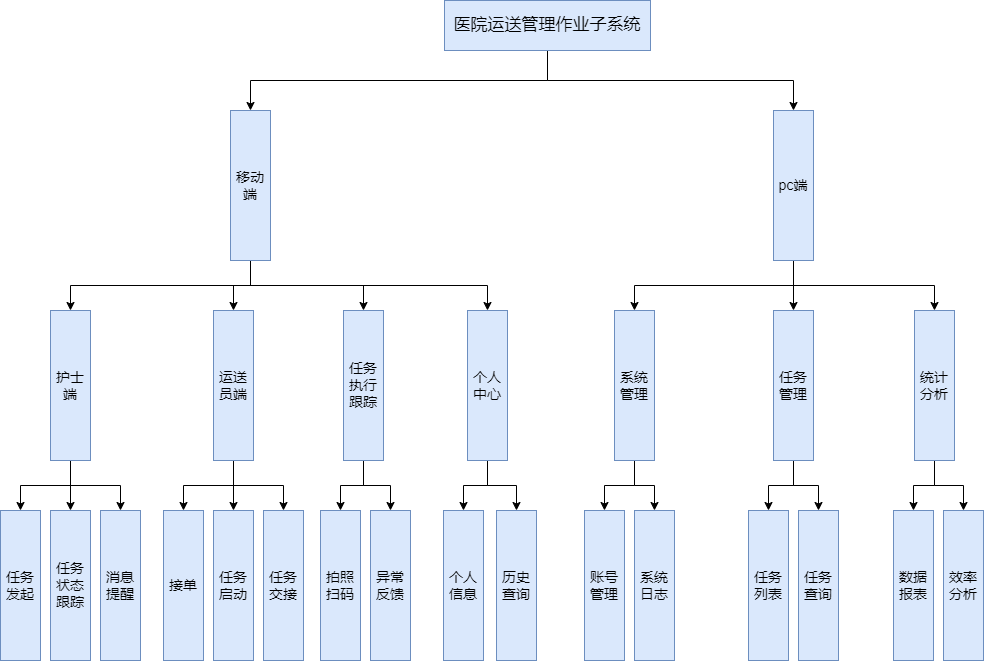
## 功能模块设计



### 功能总览

|  |  |  |
| --- | --- | --- |
| 功能模块 | 子功能 | 功能描述 |
| 登录与认证模块 | 登录系统 | 用户（医护人员、运送员、管理员）通过单一登录界面输入账号密码，系统自动验证并根据角色跳转到相应界面。 |
| 任务管理模块 | 任务发起 | 医护人员在护士端输入运输任务信息（物品名称、类型、交接节点、紧急程度等），提交任务后系统保存至Task Repository，并进入调度队列。 |
|  | 任务接单 | 运送端小程序查看已发布的运送任务并接单 |
|  | 任务智能分派 | 系统定时或事件触发，根据任务优先级与运送员状态（空闲、位置、负载）进行智能分派，调用Task Dispatch Service及运送员数据接口。 |
|  | 任务操作 | 运送员在接单后，通过扫码拍照启动任务；在各交接节点完成交接确认；完成最终交接后任务状态更新为“已完成”。 |
|  | 任务执行跟踪 | 系统实时采集运送员定位数据，通过Task Tracking Service更新任务状态与运输路径，并通过WebSocket推送至客户端。 |
|  | 任务记录查询 | 管理员在管理端可根据时间、科室、任务类型等条件查询历史任务记录，查看详细数据报表与统计分析。 |
| 系统管理模块 | 系统管理 | 提供用户账号管理、角色权限分配、任务数据维护、系统参数配置、日志监控与报表生成等功能。 |

### 功能模块图

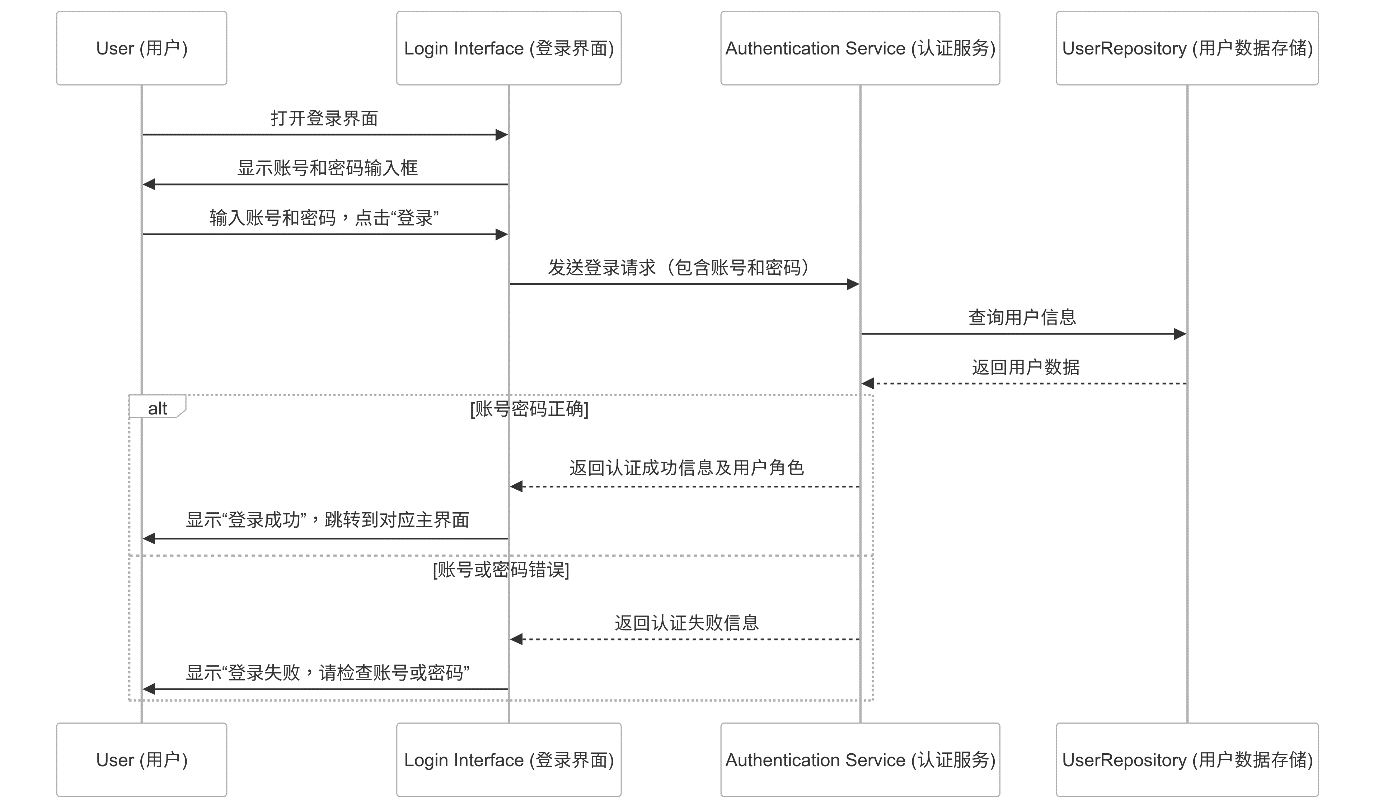


### 登录与认证模块

**功能描述**：

用户（医护人员、运送员、管理员）通过单一登录界面输入账号密码，系统自动验证并根据角色跳转到相应界面。

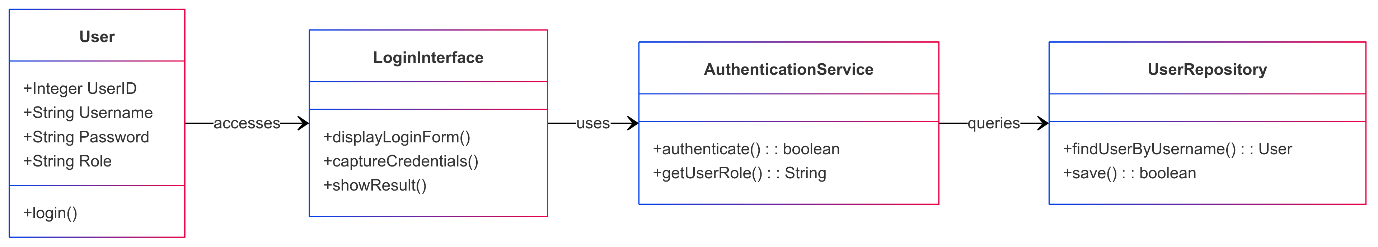
**时序图：**



**分析类：**

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 属性 |
| User | 表示所有系统用户，包括医护人员、运送员及管理员 | - UserID: integer - Username: string - Password: string - Role: string |
| LoginInterface | 提供用户登录界面，采集账号密码信息 | （主要通过方法实现，无特定属性） |
| |  | | --- | | AuthenticationService |  |  | | --- | |  | | 处理用户认证逻辑，验证账号密码 | （主要通过方法实现，无特定属性） |
| UserRepository | 存储并管理用户信息 | - UserID: integer - Username: string - Password: string - Role: string |

**类图：**



### 任务管理模块

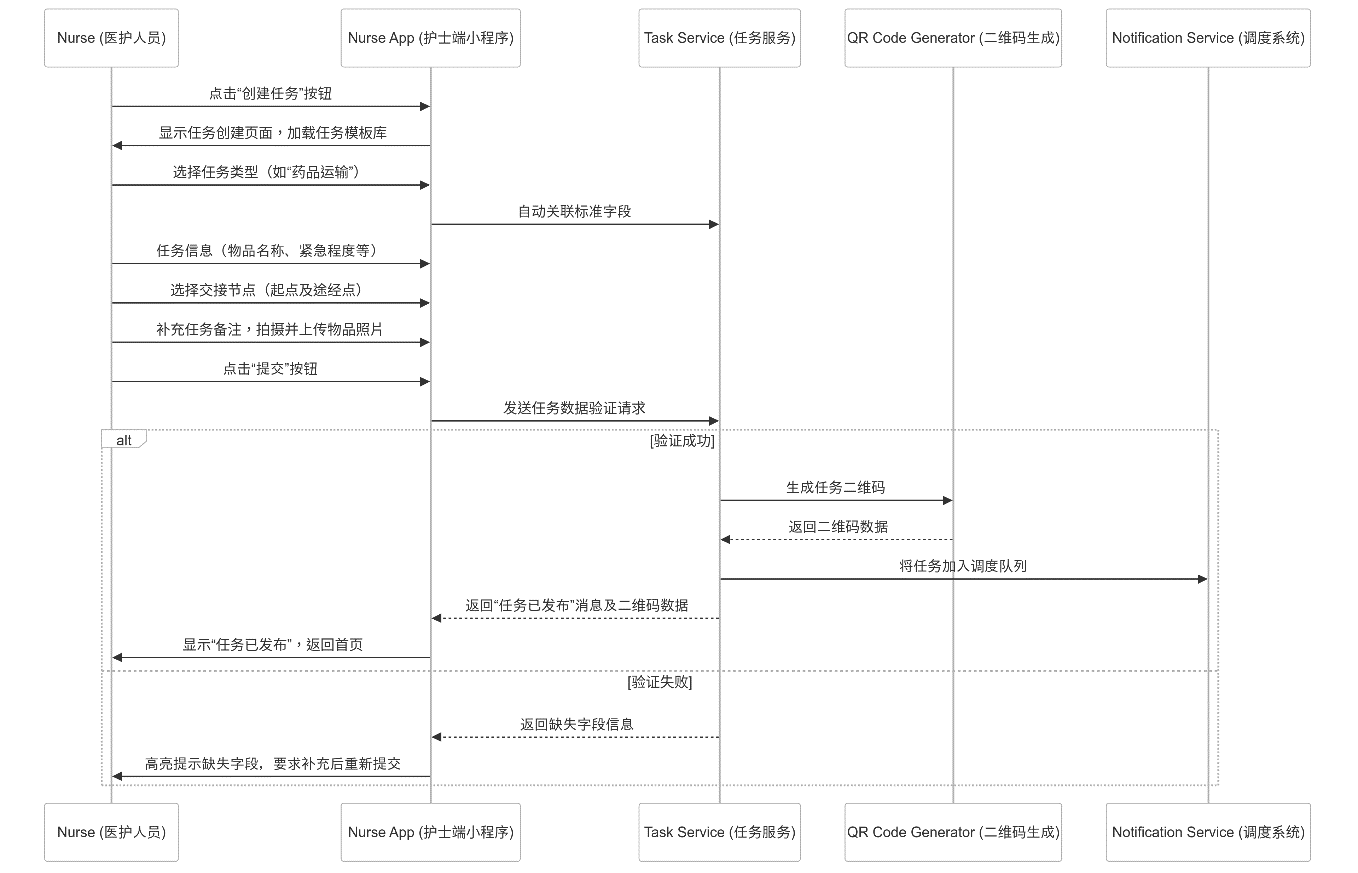


#### 任务发起

**功能描述：**

护士端小程序创建运输任务，设置运送物品名称、类型等相关信息，并制定交接节点序列（如“药房→检验科→病房”），最后发布运输任务，提交至调度系统。

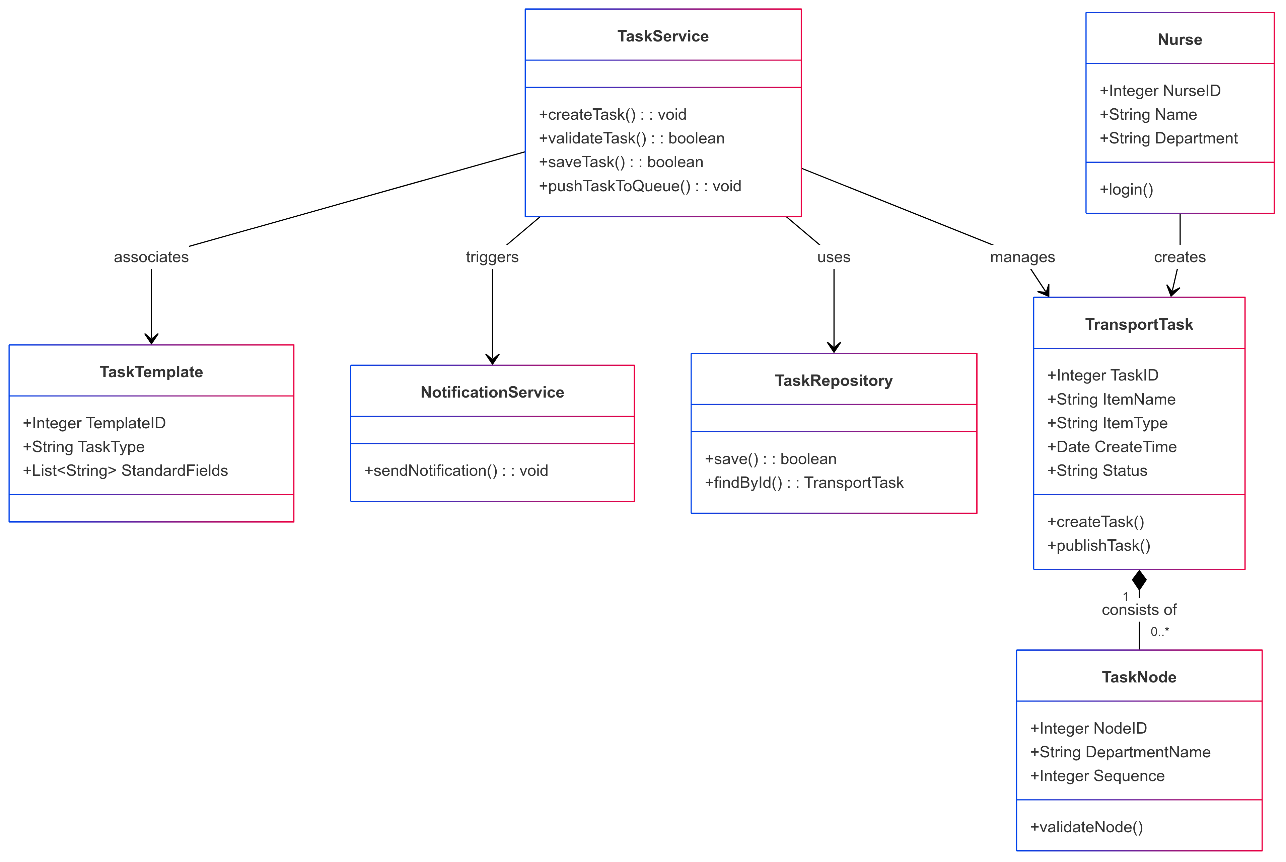
**时序图：**



**分析类：**

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 属性 |
| Nurse | 发起任务的医护人员 | - NurseID: integer - Name: string - Department: string |
| TransportTask | 运输任务基本信息及状态管理 | - TaskID: integer - ItemName: string - ItemType: string - CreateTime: Date - Status: string |
| TaskNode | 任务中各交接节点 | - NodeID: integer - DepartmentName: string - Sequence: integer |
| TaskRepository | 运输任务数据的存储 | - TaskID: integer - NurseID: integer - TaskData: string |
| NurseRepository | 医护人员信息存储 | - NurseID: integer - Department: string |

**类图****：**

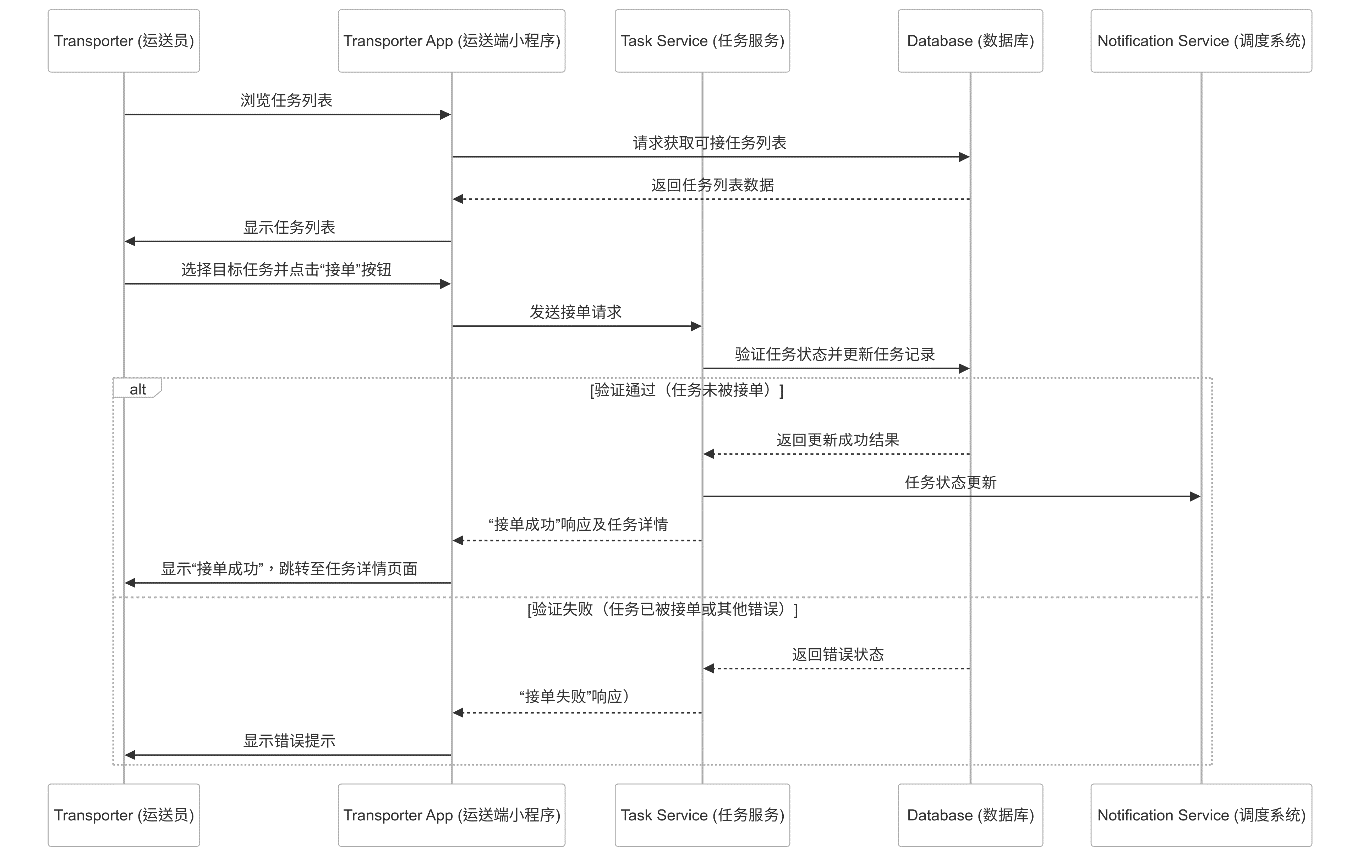


#### 运送员接单

**功能描述：**

运送端小程序展示已发布的运输任务，运送员浏览任务列表并选择接单，系统验证任务状态后更新任务信息，将任务分配给该运送员并反馈接单结果。

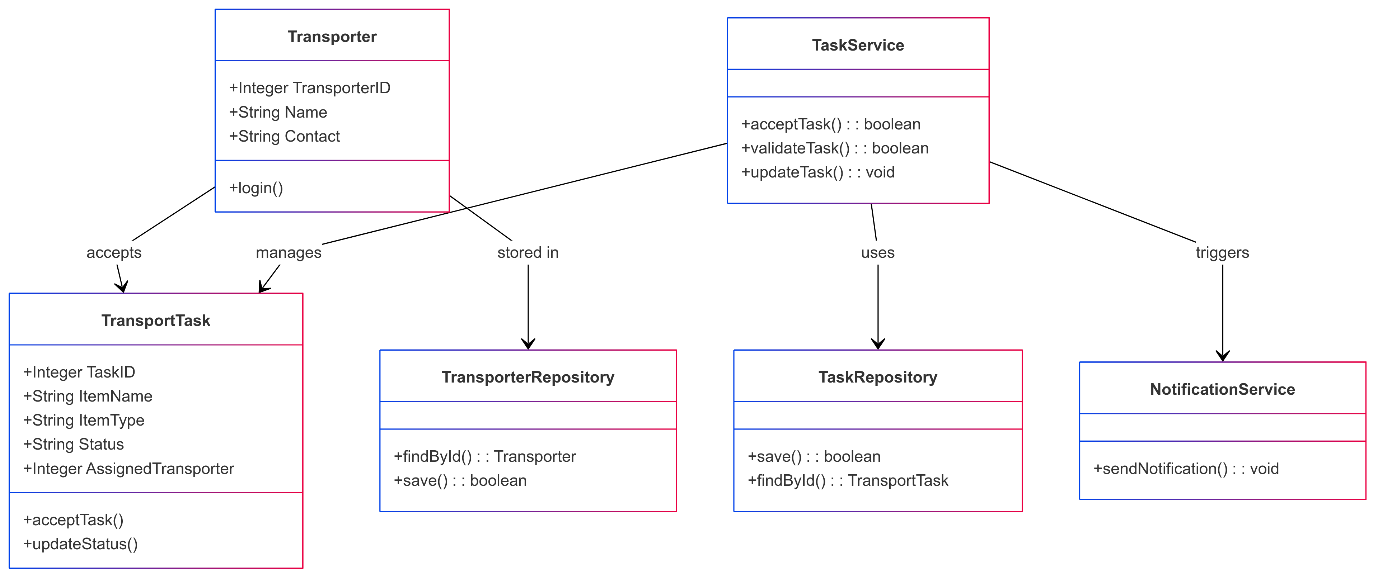
**时序图：**

****

**分析类：**

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 属性 |
| Transporter | 运送任务的执行者 | - TransporterID: integer - Name: string - Contact: string |
| TransportTask | 表示运输任务及其状态 | - TaskID: integer - ItemName: string - ItemType: string - Status: string - AssignedTransporter: integer |
| TaskRepository | 运输任务数据持久化存储 | - TaskID: integer - TaskData: string |
| TransporterRepository | 运送员信息存储 | |  | | --- | |  |  |  | | --- | | - TransporterID: integer - Name: string - Contact: string | |

**类图：**

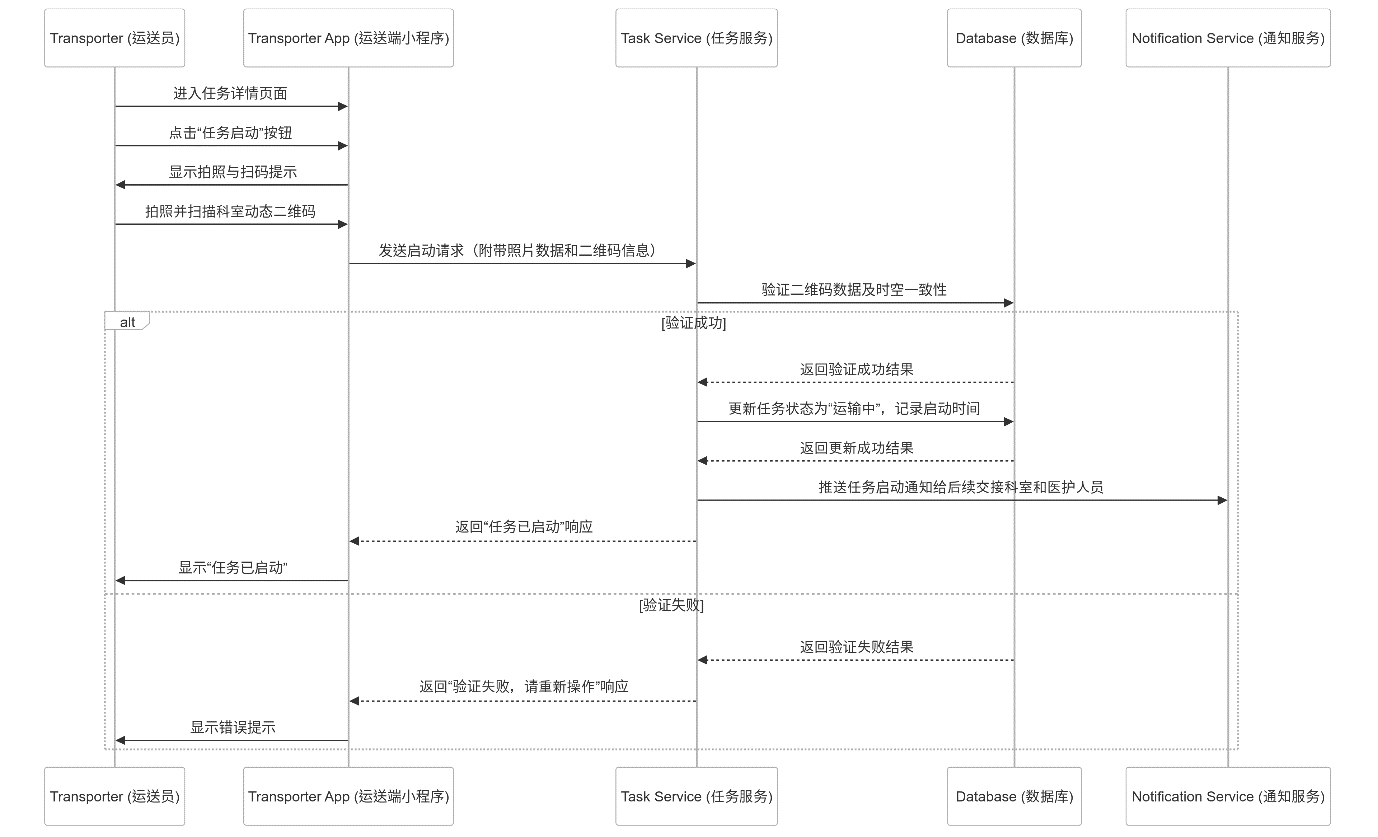
****

#### 任务启动

**功能描述：**

运送员到达任务发起科室后，接收待运物品，并使用运送端小程序进行任务启动操作。操作过程中，运送员需拍摄物品照片，并扫描部门动态二维码以确认交接。系统接收到拍照和扫码数据后，对数据进行验证，确认无误后将任务状态更新为“运输中”，并通知后续交接科室及相关医护人员。

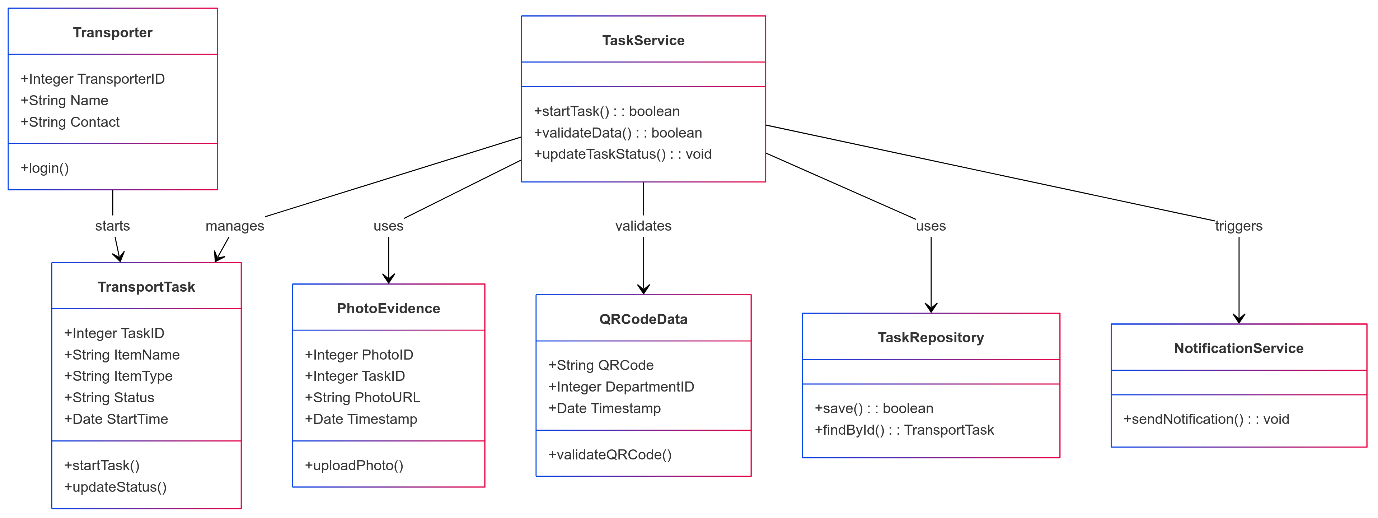
**时序图：**

****

**分析类：**

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 属性 |
| Transporter | 发起任务启动操作的运送员 | - TransporterID: integer - Name: string - Contact: string |
| TransportTask | 表示运输任务基本信息及状态 | - TaskID: integer - ItemName: string - ItemType: string - Status: string - StartTime: Date |
| TaskNode | 任务中的交接节点（不在本用例重点描述，可略） | - NodeID: integer - DepartmentName: string - Sequence: integer |
| PhotoEvidence | 拍摄的物品照片证据，用于验证任务启动 | - PhotoID: integer - TaskID: integer - PhotoURL: string - Timestamp: Date |
| QRCodeData | 二维码数据，包含科室信息及时间戳 | - QRCode: string - DepartmentID: integer - Timestamp: Date |
| TaskRepository | 运输任务数据持久化存储 | - TaskID: integer - TaskData: string |

**类图：**

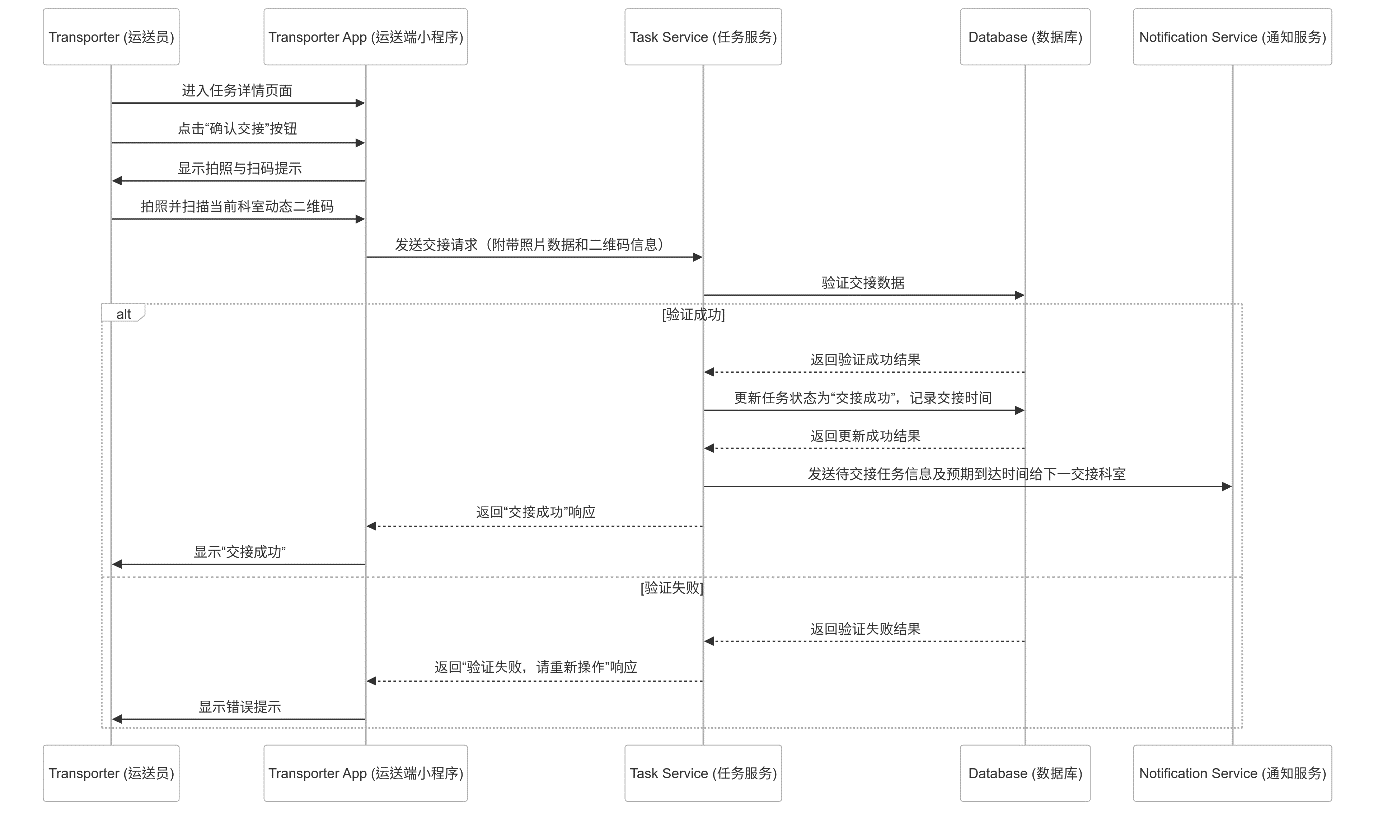
****

#### 任务交接

**用例描述：**

运送员到达指定科室后，拍照并扫描当前科室动态二维码以确认任务交接。系统验证交接信息无误后，更新任务状态，记录交接时间，并自动向下一交接科室发送待交接任务信息及预期到达时间，通知相关医护人员进行后续准备。

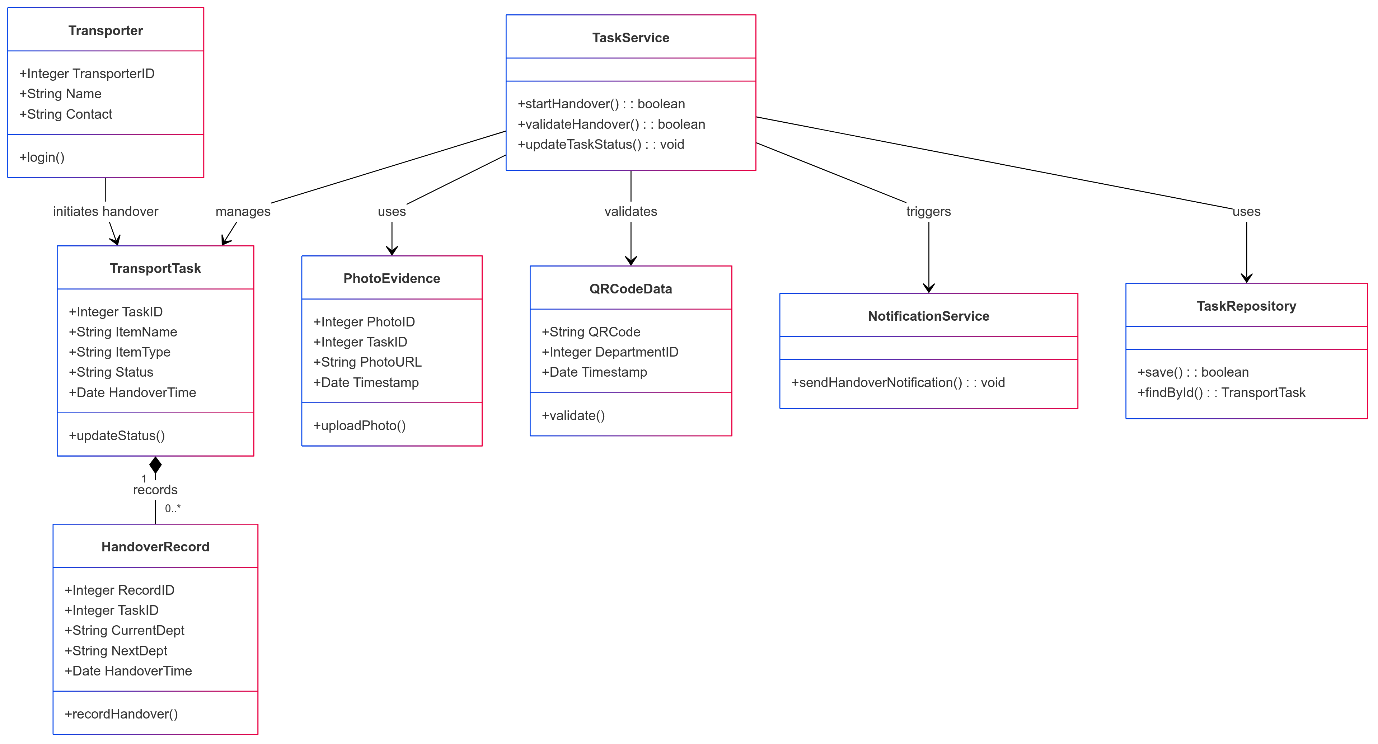
**时序图：**

****

**分析类**

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 属性 |
| Transporter | 发起任务交接操作的运送员 | - TransporterID: integer - Name: string - Contact: string |
| TransportTask | 表示运输任务的基本信息及状态管理 | - TaskID: integer - ItemName: string - ItemType: string - Status: string - HandoverTime: Date |
| PhotoEvidence | 拍摄的交接照片证据，用于记录任务交接情况 | - PhotoID: integer - TaskID: integer - PhotoURL: string - Timestamp: Date |
| QRCodeData | 二维码数据，包含当前科室信息及时间戳 | - QRCode: string - DepartmentID: integer - Timestamp: Date |
| HandoverRecord | 记录任务交接的详细过程 | - RecordID: integer - TaskID: integer - CurrentDept: string - NextDept: string - HandoverTime: Date |
| TaskRepository | 运输任务数据的持久化存储 | - TaskID: integer - TaskData: string |

**类图：**

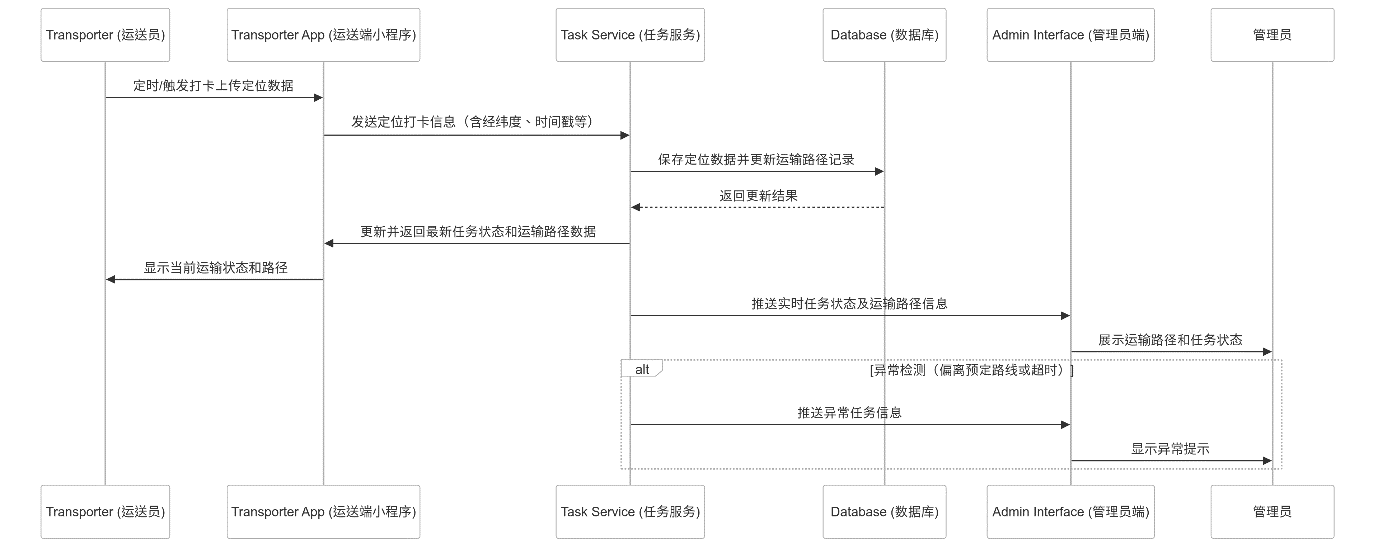
****

#### 任务执行跟踪

**用例描述：**

系统根据运送员定时上传的定位打卡信息，实时更新并可视化展示运输路径，跟踪运输任务的执行状态；同时，若发现异常（例如运送员偏离预定路线或运输时间超时），系统自动反馈异常信息给管理员，以便及时干预和处理。

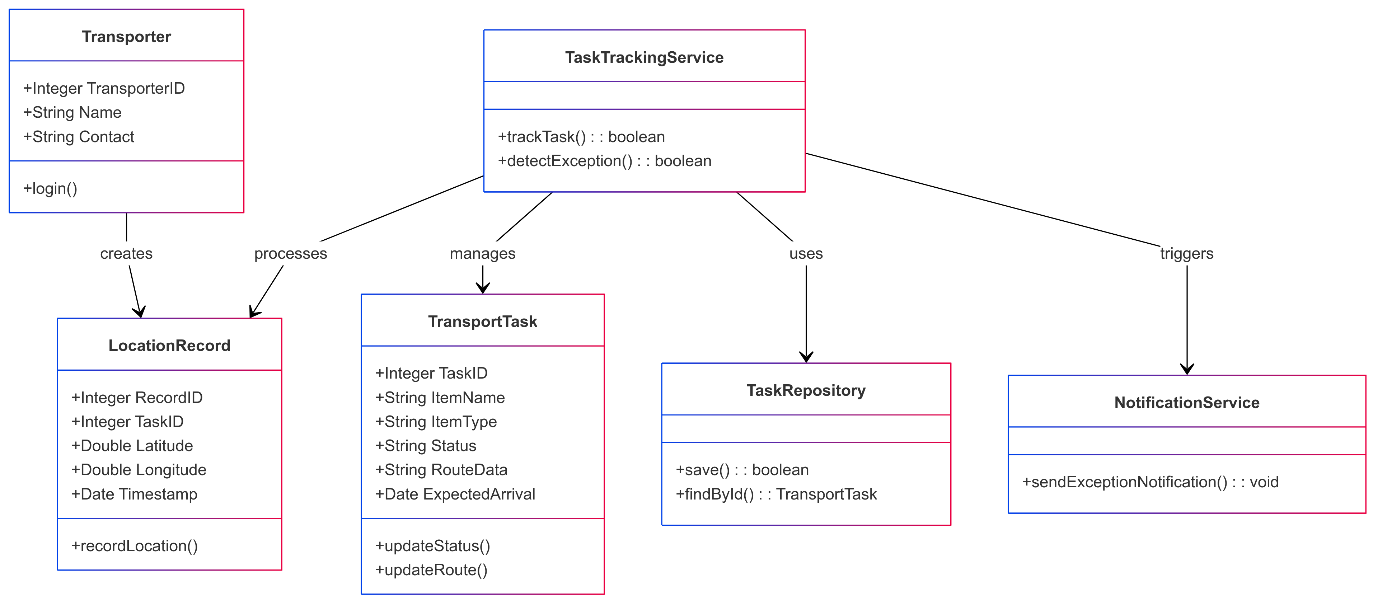
**时序图：**

****

**分析类：**

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 属性 |
| Transporter | 执行任务的运送员 | - TransporterID: integer - Name: string - Contact: string |
| LocationRecord | 定位打卡记录，记录运送员的位置信息 | - RecordID: integer - TaskID: integer - Latitude: double - Longitude: double - Timestamp: Date |
| TransportTask | 运输任务基本信息及状态管理 | - TaskID: integer - ItemName: string - ItemType: string - Status: string - RouteData: string - ExpectedArrival: Date |
| TaskRepository | 运输任务数据持久化存储 | - TaskID: integer - TaskData: string |
| TaskTrackingService | 处理定位数据，更新任务状态及运输路径 | （主要通过方法实现，无特定属性） |

**类图：**

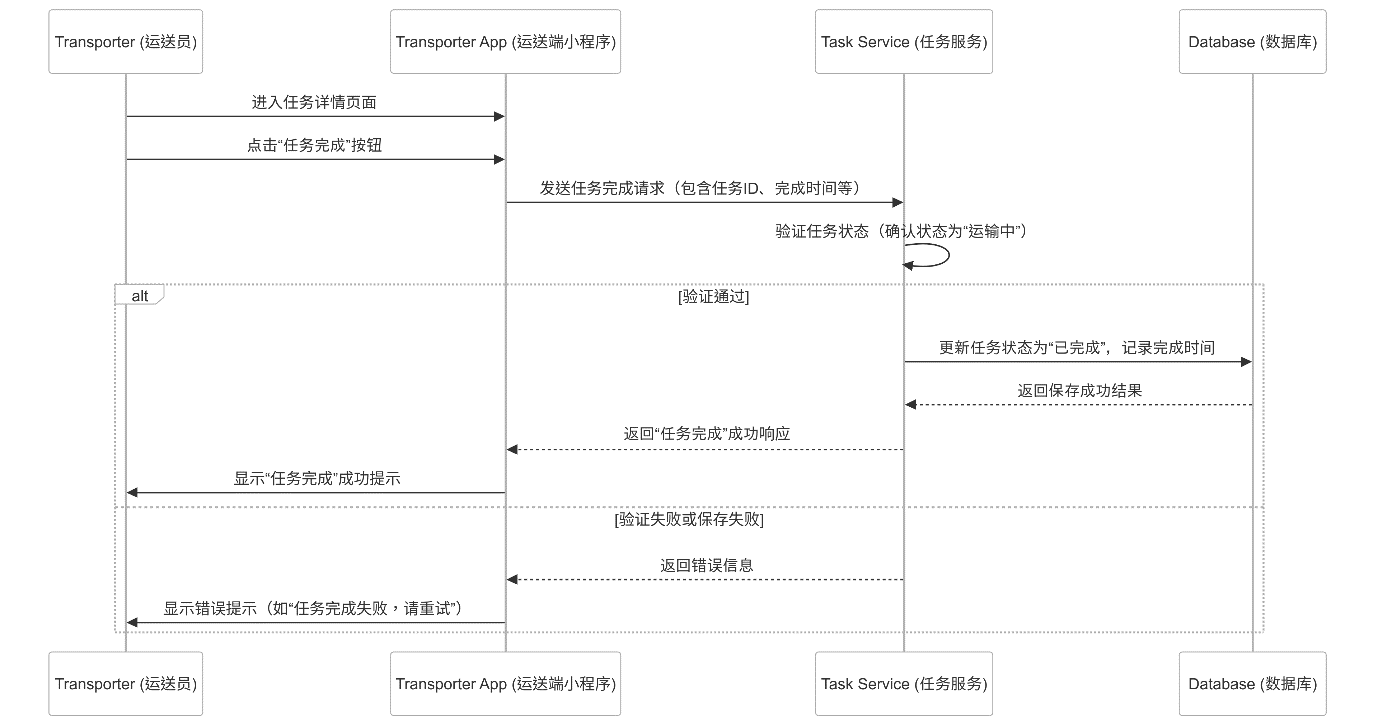


#### 任务完成

**用例描述：**

当运送员将物料送达最终科室后，运送员在运送端小程序中执行任务完成操作。系统接收到任务完成请求后，更新运输任务状态为“已完成”，记录任务完成时间和相关数据，并将完成状态反馈给运送员。

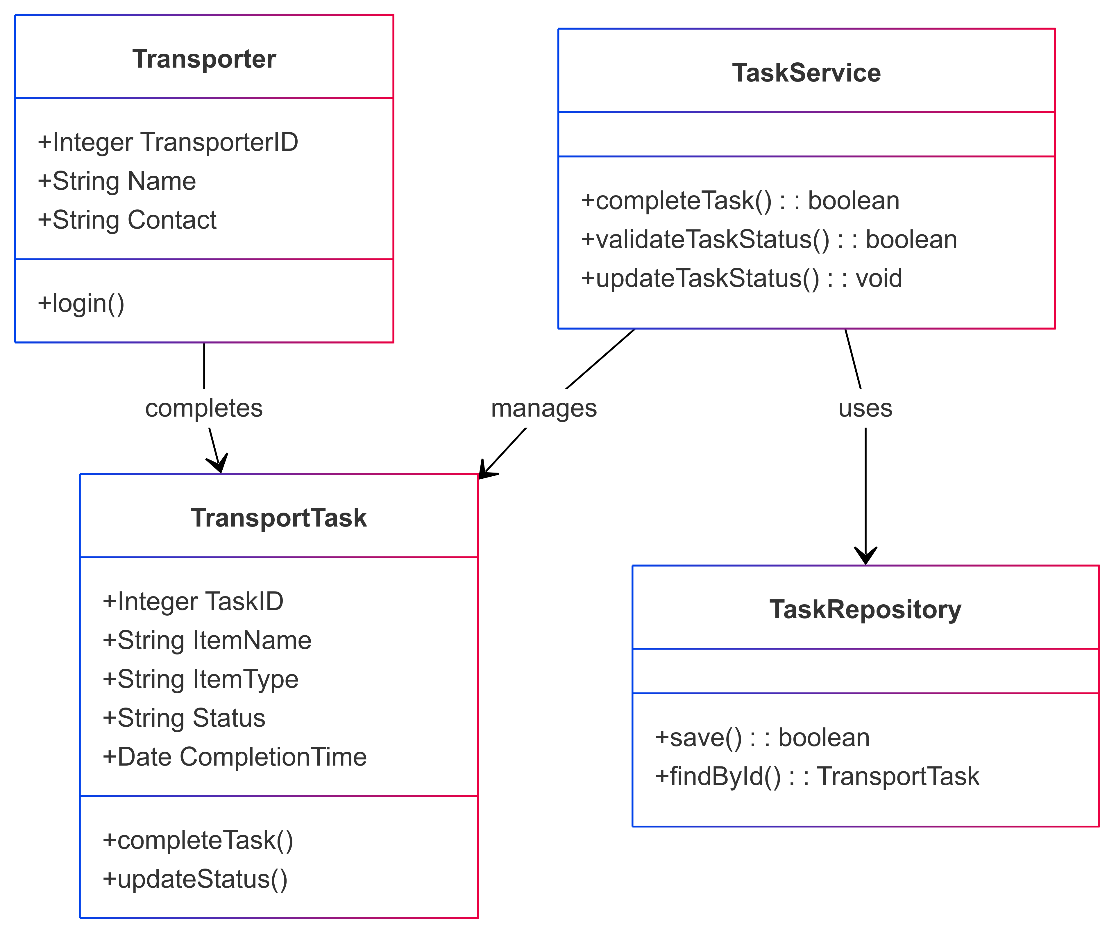
**时序图：**

****

**分析类：**

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 属性 |
| Transporter | 执行任务完成操作的运送员 | - TransporterID: integer - Name: string - Contact: string |
| TransportTask | 表示运输任务的基本信息及状态管理 | |  | | --- | |  |  |  | | --- | | - TaskID: integer - ItemName: string - ItemType: string - Status: string - CompletionTime: Date | |
| TaskService | 处理任务完成请求，验证任务状态并更新任务数据 | （主要通过方法实现，无特定属性） |
| |  | | --- | | **TaskRepository** |  |  | | --- | |  | | |  | | --- | |  |  |  | | --- | | 运输任务数据持久化存储 | | - TaskID: integer - TaskData: string |

**类图：**

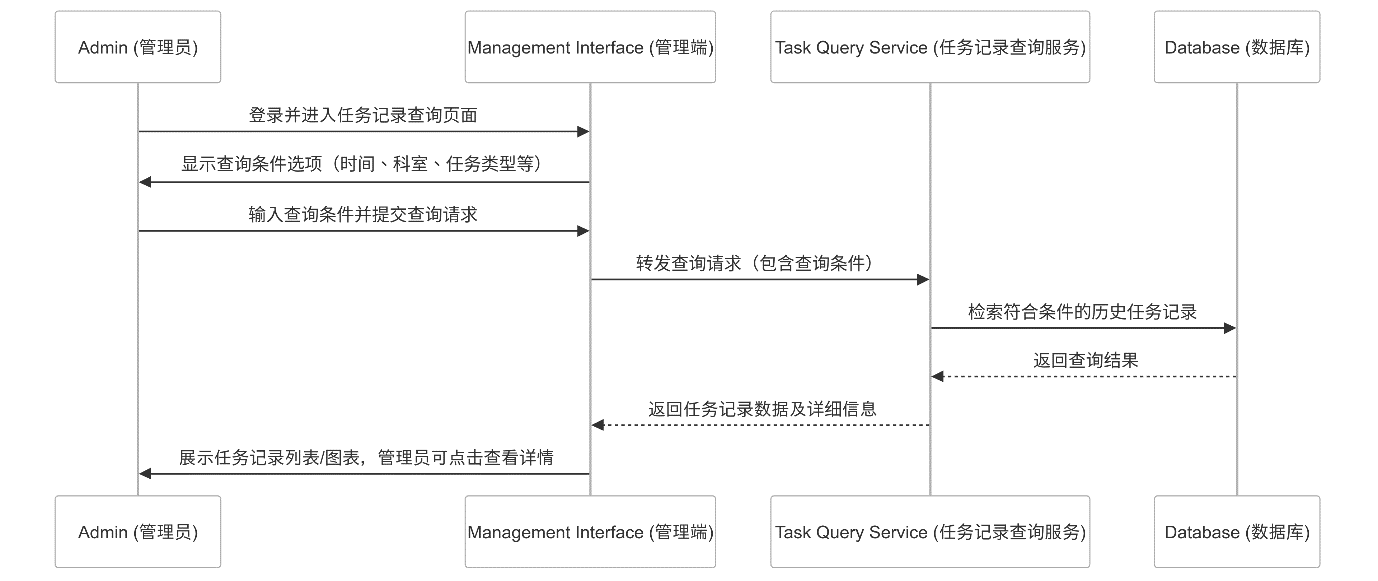
****

#### 任务记录查询

**用例描述：**

管理员通过管理端系统，根据时间、科室、任务类型等条件检索历史运输任务记录，并查看各任务的执行详情（包括任务状态、运输路径、交接情况、任务完成时间等），以便统计分析和业务评估。

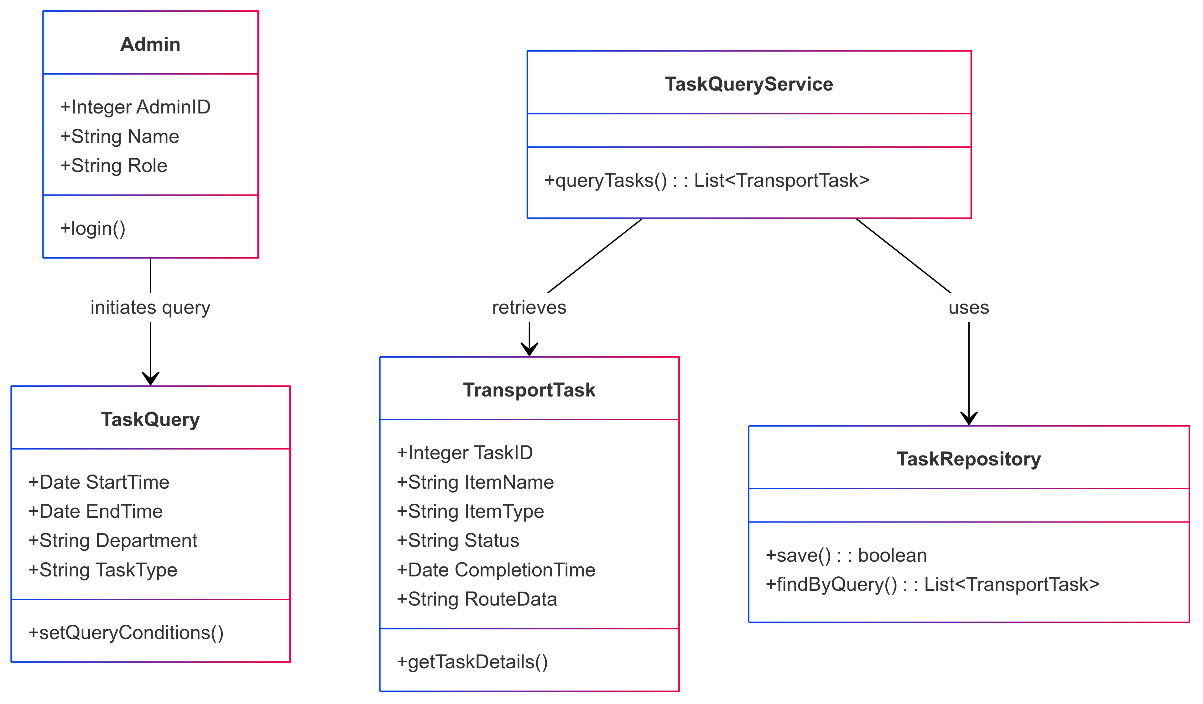
**时序图：**

****

**分析类：**

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 属性 |
| Admin | 管理员，负责查询和管理历史任务记录 | - AdminID: integer - Name: string - Role: string |
| TransportTask | 表示运输任务的基本信息及执行详情 | - TaskID: integer - ItemName: string - ItemType: string - Status: string - CompletionTime: Date - RouteData: string |
| TaskQuery | 封装任务记录查询条件 | - StartTime: Date - EndTime: Date - Department: string - TaskType: string |
| TaskQueryService | 处理任务记录查询请求，检索历史任务记录 | （主要通过方法实现，无特定属性） |
| TaskRepository | 运输任务数据的持久化存储 | - TaskID: integer - TaskData: string |

**分析类：**

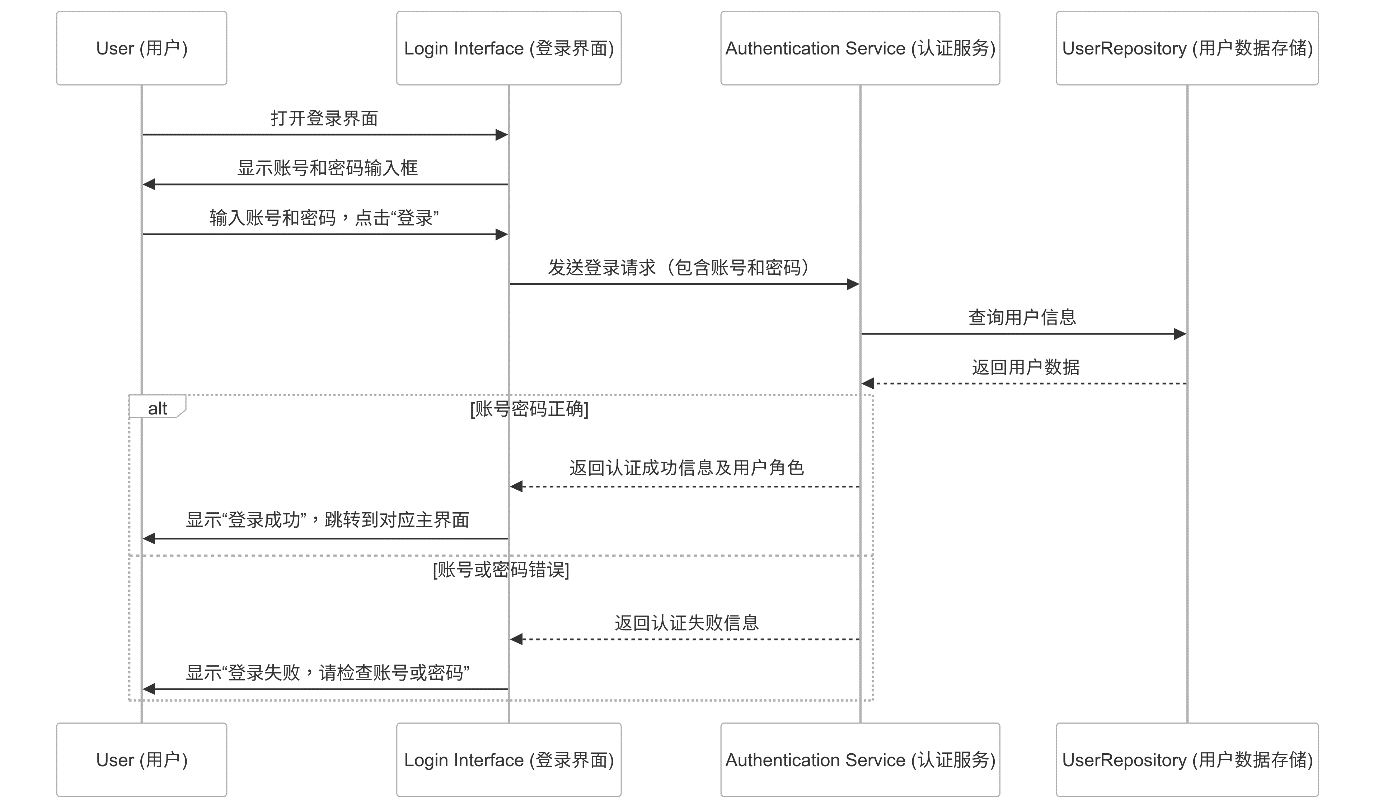
****

#### 登录

**功能描述：**

医护人员、运送员及系统管理员在登录界面输入用户名和密码，系统通过认证后允许用户进入各自的工作系统。认证过程包括验证用户输入的账号、密码，并根据用户角色跳转到相应的主界面。

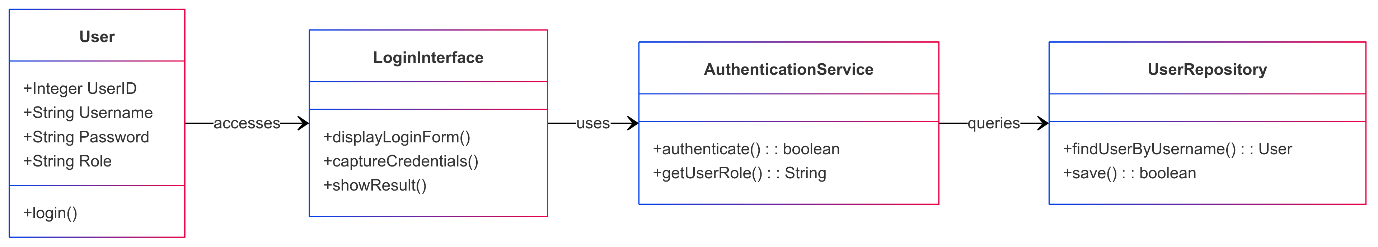
**时序图：**



**分析类：**

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 属性 |
| User | 表示所有系统用户，包括医护人员、运送员及管理员 | - UserID: integer - Username: string - Password: string - Role: string |
| LoginInterface | 提供用户登录界面，采集账号密码信息 | （主要通过方法实现，无特定属性） |
| |  | | --- | | AuthenticationService |  |  | | --- | |  | | 处理用户认证逻辑，验证账号密码 | （主要通过方法实现，无特定属性） |
| UserRepository | 存储并管理用户信息 | - UserID: integer - Username: string - Password: string - Role: string |

**类图：**

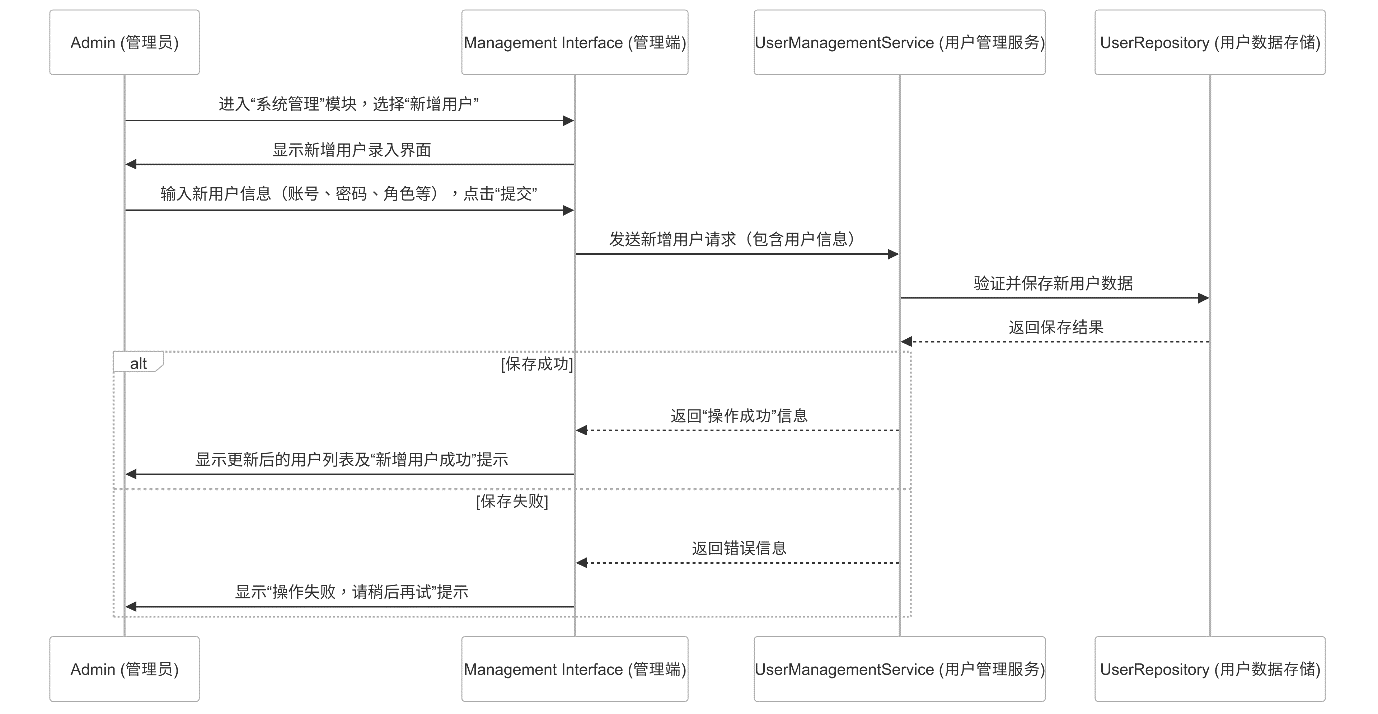


#### 系统管理

**用例描述：**

管理员通过管理端对系统中的用户账号进行管理，包括创建新用户、修改用户信息、删除用户账号和查询用户列表。该功能确保系统用户信息的准确性和安全性，并支持用户权限的分配和更新。

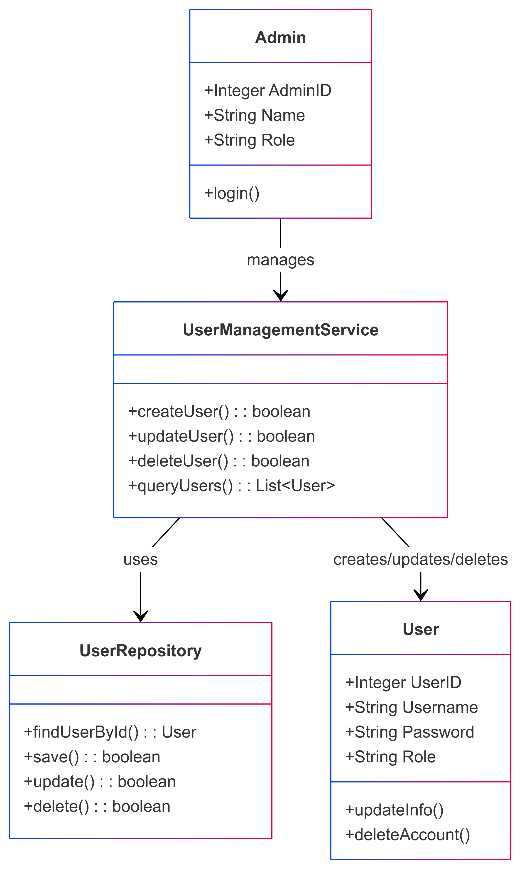
**时序图：**



**分析类：**

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 属性 |
| Admin | 系统管理员，负责对用户账号进行管理 | |  | | --- | |  |  |  | | --- | | - AdminID: integer - Name: string - Role: string | |
| User | 表示系统中的用户账号，包括医护人员、运送员等 | - UserID: integer - Username: string - Password: string - Role: string |
| |  | | --- | | **UserManagementService** |  |  | | --- | |  | | 处理用户账号的创建、修改、删除、查询等业务逻辑 | （主要通过方法实现，无特定属性） |
| UserRepository | 存储并管理用户账号数据 | - UserID: integer - Username: string - Password: string - Role: string |

**类图：**

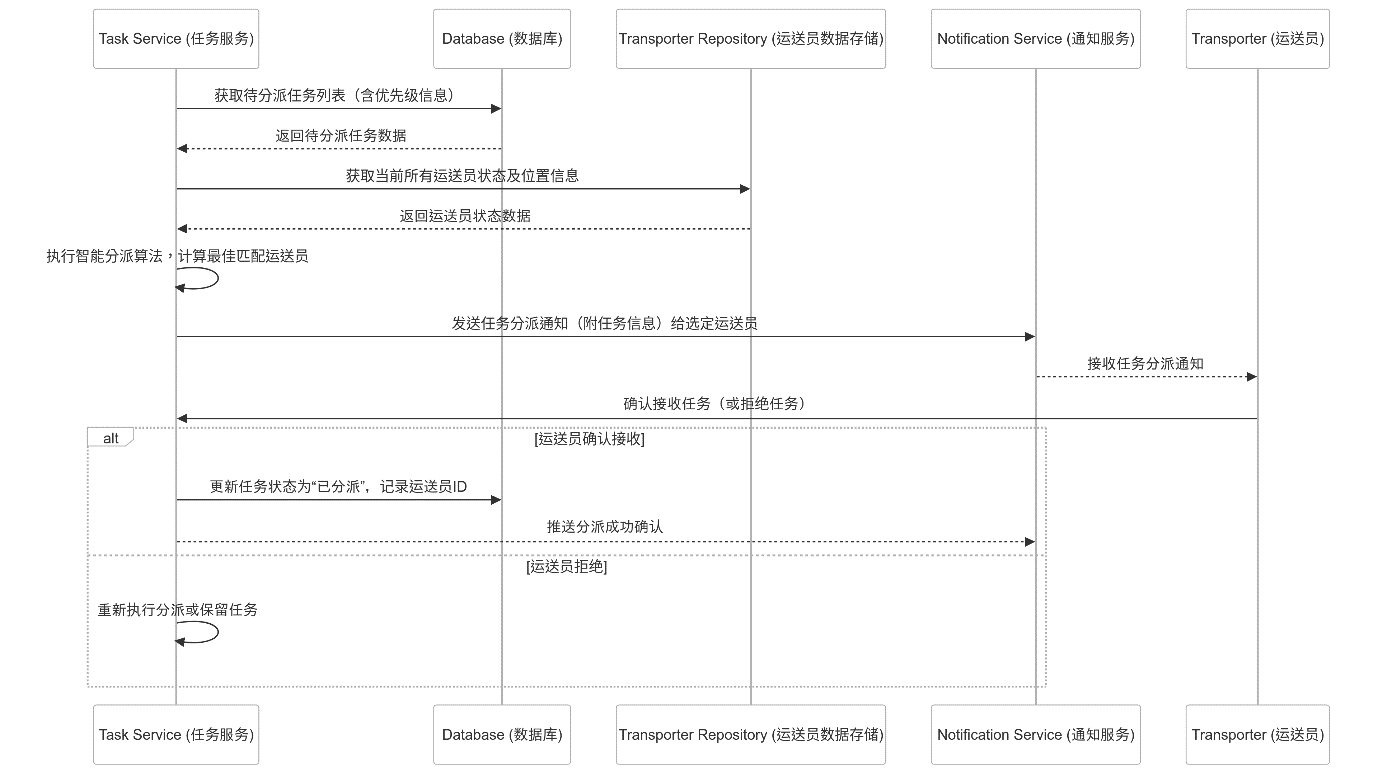
****

#### 任务分派

**用例描述：**

系统在检测到有待分派的运输任务后，根据任务的优先级和运送员的当前状态（例如空闲状态、当前位置、负载情况等）通过智能分派算法自动为任务匹配最合适的运送员。分派完成后，系统将任务通知推送给选定的运送员，并等待其确认接收。

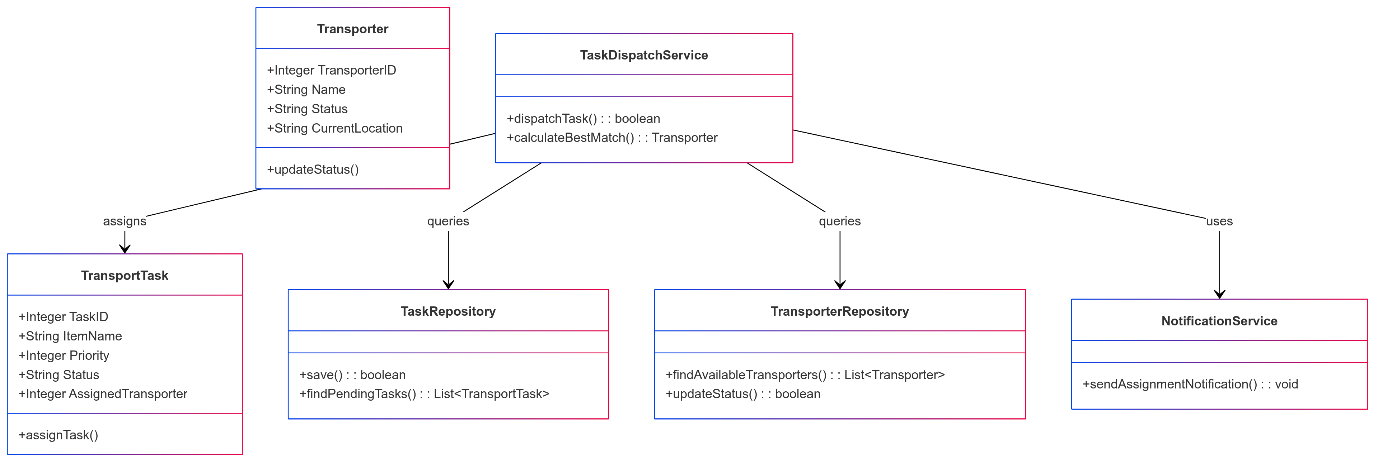
**时序图：**

****

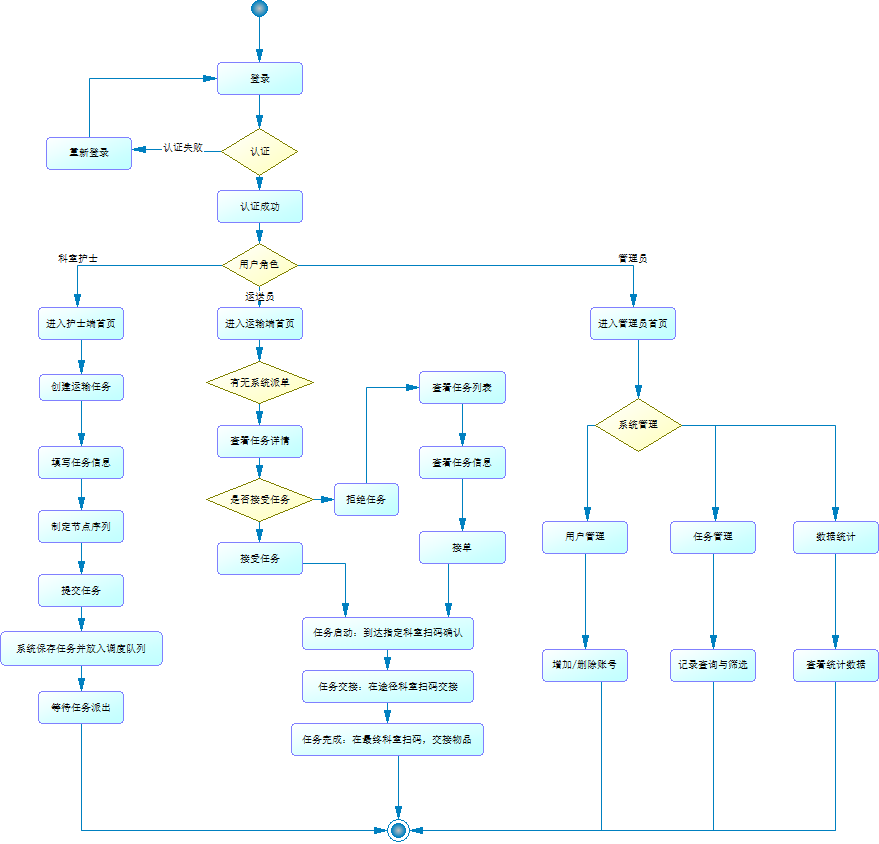
**分析类：**

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 属性 |
| TransportTask | 表示运输任务，包含任务详情、优先级及状态等 | - TaskID: integer - ItemName: string - Priority: integer - Status: string - AssignedTransporter: integer |
| Transporter | |  | | --- | |  |  |  | | --- | | 表示运送员，包含运送员基本信息及当前状态 | | |  | | --- | |  |  |  | | --- | | - TransporterID: integer - Name: string - Status: string  - CurrentLocation: string | |
| TaskDispatchService | 任务分派服务，根据任务优先级和运送员状态进行智能匹配和分派 | （主要通过方法实现，无特定属性） |
| TaskRepository | 存储和查询运输任务数据 | - TaskID: integer - TaskData: string |
| |  | | --- | | **TransporterRepository** |  |  | | --- | |  | | 存储和查询运送员状态及相关信息 | - TransporterID: integer - Status: string - Location: string |
| **NotificationService** | 负责向运送员推送任务分派通知 | （主要通过方法实现，无特定属性） |

**类图：**

****

### 系统流程图



系统流程从用户进入登录界面开始，用户输入账号与密码后，系统对其进行身份验证。若验证失败，系统会显示错误提示并允许用户重新尝试登录或退出；若验证成功，则根据用户角色跳转至对应的功能界面。对于护士角色，系统在完成登录后会进入护士端主界面。护士可通过“创建任务”功能填写运输物品名称、类型、紧急程度、交接科室等信息，并拍照或上传二维码数据以辅助后续的扫码验证。系统在确认信息完整性后，将任务数据保存到数据库并将新任务加入调度队列等待分派。护士也可以随时在护士端查看自己创建或关联的任务状态，了解任务是否已被运送员接单、正在运输还是已经完成。

当用户角色为运送员时，系统会在登录后进入运送员端主界面。运送员可以浏览并选择待接单的任务，通过点击“接单”将任务状态更新并记录任务与运送员的关联关系。到达发起科室后，运送员可使用扫码或拍照完成交接验证，系统会将信息记录到任务节点数据中，并更新任务状态为“运输中”。在随后的运输过程中，每到达一个科室，运送员都可以进行扫码或拍照确认交接，同时在系统中填写或上传定位记录，用于实时跟踪运输路径；若任务有中途停顿或异常，运送员可通过异常反馈流程向护士或管理员报告。物料最终送达指定的科室后，运送员点击“任务完成”，系统便将该任务状态更新为“已完成”，并记录完成时间与交接信息。

如果用户角色为管理员，则登录成功后进入管理员端主界面。管理员可以在系统管理功能中对用户信息、科室信息及其他系统参数进行配置，包括新建、修改或删除用户，并为其分配角色和科室归属。管理员也可以在任务管理界面中查看所有运输任务的列表和详细信息，支持手动分派任务、调整或终止异常任务等操作。通过统计分析模块，管理员可基于系统存储的历史任务数据、定位记录及通知信息生成报表，评估物料运输效率和科室工作量，并对异常情况进行追溯。

在整个流程中，无论是护士发起任务、运送员接单运输，还是管理员进行管理和统计，系统都会根据任务状态变化或异常情况触发通知，并将通知推送给相应的用户。数据库的核心表（USER、TRANSPORT\_TASK、TASK\_NODE、LOCATION\_RECORD、DEPARTMENT、NOTIFICATION）相互关联，确保任务在每个节点的状态与数据信息都能被完整记录和实时更新。通过这样的流程设计，医院运送管理系统实现了从任务创建、分派、运输、完成到异常处理和统计分析的全流程管理，为医院内部的物流运输提供了高效、准确和可追溯的技术支持。

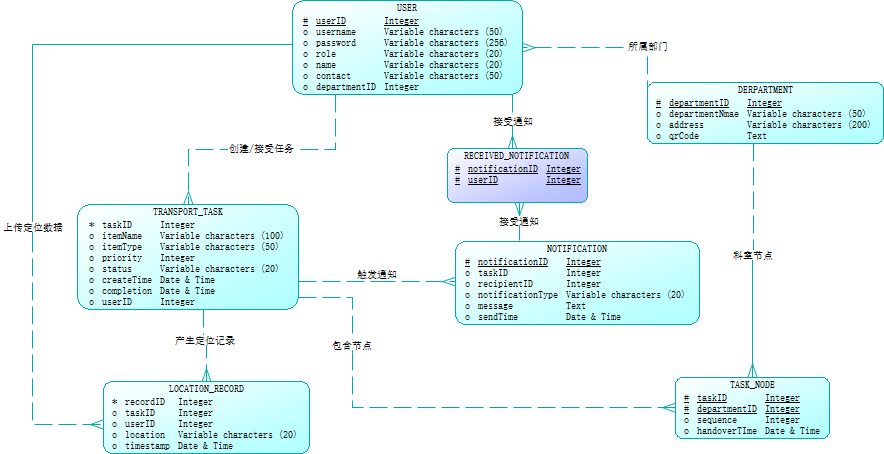
## 数据库设计

本系统数据库采用MySQL进行存储设计，通过逻辑设计和物理设计图对系统核心数据实体及其关系进行全面展示。逻辑设计包括六个核心实体：用户（USER）、运输任务（TRANSPORT\_TASK）、任务节点（TASK\_NODE）、定位记录（LOCATION\_RECORD）、部门（DEPARTMENT）以及通知（NOTIFICATION）。



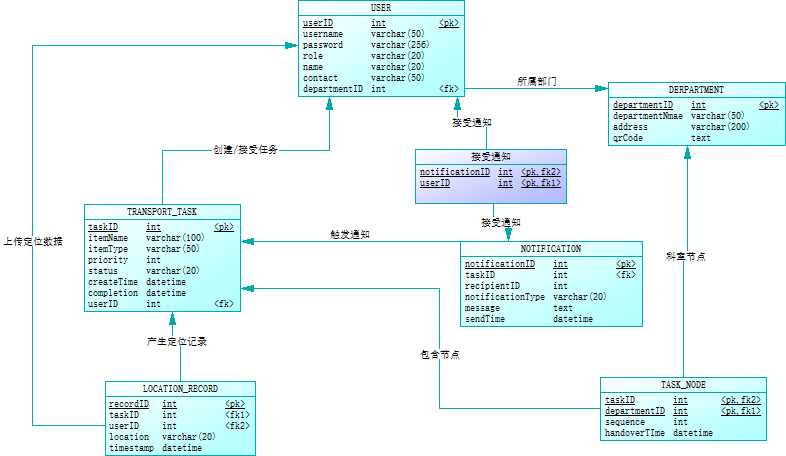
### 逻辑设计

在章节2.1概念设计中，使用PowerDesigner软件设计生成了概念模型ER图，此章节我们由ER模型生成了逻辑模型，逻辑模型中的每个实体包含了较为完整的外键约束。



### 物理设计

由关系模型生成了物理模型，包含本系统数据库中的所有表，并最终的物理模型如图所示。



### 数据库设计原则

* 保证数据一致性与完整性，采用事务机制；
* 对敏感数据采用加密存储；
* 建立适当索引，提高查询效率；
* 考虑分布式部署和读写分离策略以应对高并发。

### 数据库设计

系统数据库采用MySQL进行存储设计，主要涉及六个核心表：用户、运输任务、任务节点、定位记录、通知和部门。各表之间通过外键建立关联，确保数据一致性和完整性。

|  |  |
| --- | --- |
| 表 | 描述 |
| 用户表（USER） | 存储用户ID、账号、密码（加密）、姓名、角色、联系方式等。 |
| 任务表（TRANSPORT\_TASK） | 记录运输任务的基本信息和状态，包括任务物品、优先级、状态、创建时间、完成时间及分派给的运送员 |
| 任务节点表（TASK\_NODE） | 记录运输任务中的各个交接节点信息，采用复合主键 (taskID, departmentID) 以确保每个任务与科室的唯一关联，同时记录交接顺序和实际交接时间。 |
| 定位记录表（LOCATION\_RECORD） | 记录运送员在任务执行过程中上传的定位数据，包括任务关联、运送员、定位位置和上传时间。 |
| 部门表（DEPARTMENT） | 存放医院各科室的详细信息，包括科室名称、地址和用于扫码定位的二维码数据。 |
| 通知表（NOTIFICATION） | 记录系统向用户推送的通知信息，关联任务和接收通知的用户。 |
| 文件元数据表（FILE） | 存储文件的元数据（如文件名、存储路径、与任务的关联等） |

#### User表

存储所有系统用户信息，包括医护人员、运送员和管理员

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 描述 |
| userID | INT | AUTO\_INCREMENT PRIMARY KEY | 用户唯一标识 |
| username | VARCHAR(50) | NOT NULL UNIQUE | 登录账号 |
| password | VARCHAR(256) | NOT NULL | 密码 |
| role | ENUM('Nurse', 'Transporter', 'Admin') | NOT NULL | 用户角色，区分医护人员、运送员和管理员。 |
| name | VARCHAR(100) | NOT NULL | 用户姓名 |
| contact | VARCHAR(50) |  | 联系方式 |
| departmentID | INT | NOT NULL | 所属部门 |

#### TRANSPORT\_TASK表

记录医院运输任务的基本信息和状态。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 描述 |
| taskID | INT | AUTO\_INCREMENT PRIMARY KEY | 任务唯一标识 |
| itemName | VARCHAR(100) | NOT NULL | 运输物品名称 |
| itemType | VARCHAR(50) | NOT NULL | 物品类型（如标本、药品、文件等） |
| priority | INT | NOT NULL DEFAULT 0 | 任务优先级，数值越大优先级越高 |
| status | ENUM('待分派', '运输中', '已完成') | NOT NULL DEFAULT '待分派' | 当前任务状态 |
| createTime | DATETIME | NOT NULL DEFAULT CURRENT\_TIMESTAMP | 任务创建时间 |
| completionTime | DATETIME | NULL | 任务完成时间（完成时记录） |
| assignedTransporter | INT | NULL | 外键，关联USER表的userID，表示分派给的运送员 |

约束：assignedTransporter为外键，引用USER(userID)，在删除或更新用户时需要设定适当的联级策略

#### TASK\_NODE表

记录运输任务中的各个交接节点信息，用于跟踪任务在各科室的交接情况。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 描述 |
| taskID | INT | NOT NULL | 复合主键，外键，关联TRANSPORT\_TASK(taskID) |
| departmentID | INT | NOT NULL | 复合主键，外键，关联DEPARTMENT(departmentID) |
| sequence | INT | NOT NULL | 交接顺序，确保任务交接按预定顺序执行 |
| handoverTime | DATETIME | NULL | 实际交接时间 |

约束：taskID和departmentID为外键，分别引用TRANSPORT\_TASK和DEPARTMENT表

#### LOCATION\_RECORD表

记录运送员在任务执行过程中上传的定位数据，用于实时跟踪任务运输路径。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 描述 |
| recordID | INT | AUTO\_INCREMENT PRIMARY KEY | 定位记录唯一标识 |
| taskID | INT | NOT NULL | 外键，关联TRANSPORT\_TASK(taskID) |
| userID | INT | NOT NULL | 外键，关联USER(userID)，记录上传定位数据的运送员 |
| location | VARCHAR(20) | NOT NULL | 定位位置 |
| timestamp | DATETIME | NOT NULL DEFAULT CURRENT\_TIMESTAMP | 定位数据上传时间 |

**索引：**对taskID和userID建立索引，便于查询特定任务或运送员的定位数据

#### NOTIFICATION表

存储系统向用户推送的通知信息，用于任务状态更新、异常提醒等。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 描述 |
| notificationID | INT | AUTO\_INCREMENT PRIMARY KEY | 通知记录唯一标识 |
| taskID | INT | NOT NULL | 外键，关联TRANSPORT\_TASK(taskID) |
| recipientID | INT | NOT NULL | 外键，关联USER(userID)，表示接收通知的用户 |
| notificationType | VARCHAR(20) | NOT NULL | 通知类型 |
| message | TEXT | NOT NULL | 通知内容 |
| sendTime | DATETIME | NOT NULL DEFAULT CURRENT\_TIMESTAMP | 通知发送时间 |

**约束：**确保taskID和recipientID正确关联相应的任务和用户。

#### DEPARTMENT表

存放医院各个科室的详细信息，包括科室名称、位置以及二维码数据，用于任务交接时通过扫描二维码进行验证。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 描述 |
| departmentID | INT | AUTO\_INCREMENT PRIMARY KEY | 科室唯一标识 |
| departmentName | VARCHAR(50) | NOT NULL UNIQUE | 科室名称 |
| address | VARCHAR(200) | NOT NULL | 详细地址 |
| qrCode | TEXT | NOT NULL | 二维码数据或二维码图片链接，用于扫描定位 |

#### FILE表

存放文件的元数据（如文件名、存储路径、与任务的关联等）。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 描述 |
| fileID | INT | AUTO\_INCREMENT PRIMARY KEY | 文件唯一标识 |
| taskID | INT | NOT NULL | 所属任务id |
| fileName | VARCHAR(255) | NOT NULL | 文件名称 |
| filePath | VARCHAR(255 | NOT NULL | 文件存储路径 |
| fielType | VARCHAR(50) | NOT NULL | 文件类型 |
| Stage | VARCHAR(20) | NOT NULL | 所属任务阶段 |
| uploadTime | DATETIME | NOT NULL | 上传时间 |

## 接口设计



### 用户接口设计

**表现层接口**：

* 支持微信小程序及主流HTML5浏览器（Chrome、Firefox、Safari、Edge）。
* 提供响应式界面，支持任务创建、查询、分派、操作确认等。

**API接口**：

* 前后端通信采用RESTful API和WebSocket，数据格式为JSON；
* 接口文档详细说明各API参数、返回值及错误码，确保开发一致性

### 系统接口设计

1. 前端需支持微信小程序平台及HTML5标准的浏览器，具备通过浏览器调用系统API、使用设备硬件（如摄像头、定位模块）等功能。
2. 系统后端通过RESTful API与前端进行数据交互，同时利用WebSocket实现实时数据推送。
3. 系统应提供标准的RESTful API接口，与微信开放平台、地理位置服务平台（如腾讯、百度地图API）及第三方认证系统进行数据交互。

### 外部接口

* **定位接口**：接入腾讯或百度地图API，实现实时定位及路径规划。
* **加密接口**：使用第三方加密模块，保障用户信息和任务数据传输安全。

## 安全性设计



### 用户信息安全

采用数据库作为用户账号、任务信息、定位数据、交接记录等数据的存储后端，实现严格的访问控制与权限分配，确保不同角色只能访问对应数据。

### 数据安全传输

所有数据传输均应通过加密算法（如SSL/TLS）加密，确保浏览器与服务器之间建立安全连接，确保用户登录、任务数据及定位信息的安全传输，防止信息在传输过程中被窃取或篡改。

### 系统安全

应建立完善的日志记录和监控机制，及时发现并处理潜在的安全威胁和系统异常。

## 易用性与维护性



### 易用性设计

**界面设计**：

* 简洁直观，导航层级不超过2级；
* 异步请求时提供加载动画和明确的错误提示；
* 提供上下文帮助及操作说明。

**用户帮助**：

* 内置帮助文档和FAQ；
* 在线客服或反馈通道，及时响应用户问题。

### 系统维护

系统架构应具备高模块化、低耦合性、可重用性和可维护性，便于后续功能扩展、升级和故障排查。同时要有完善的日志记录与监控平台，便于日常维护和故障排查。