# 项目环境配置

## **后端数据库**

### 后端数据库

如果在教室上课（学生宿舍无法使用），可以使用学校部署的mysql服务器。

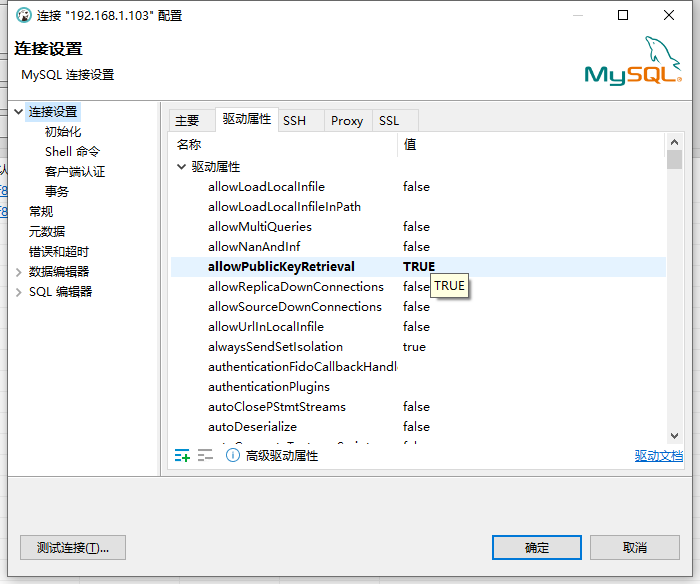
ip: 10.60.1.38:3306

### 数据库客户端dbeaver

可以使用自己喜欢的工具或者Dbeave来做客户端访问数据库。

在使用Dbeaver时，有可能会报错：Public Key Retrieval is not allowed

报错的纠正办法是设置一下连接参数：



## **后端开发工具：Idea community**

### 使用idea intelij community

访问jetbrains网站下载页面，滚动页面，社区版在页面的下半部分。

地址：<https://www.jetbrains.com/idea/download/?section=windows>

### mvn镜像源

maven是一个java项目依赖包仓库的本地管理管，它的安装不分windows版本或者linux版本。只需要下载解压包解压。

1. maven不须安装的解压包下载地址：<https://maven.apache.org/download.cgi>，或者使用下发包直接解压。
2. 让ide指向使用它的路径。
3. 让环境变量指向它的bin路径。
4. 配置setting.xml，让它在中国大陆区域使用阿里云仓库，具体内容如下。
   1. 创建D:/mavenRepositery.
   2. 打开c:/用户/.m2/setting.xml 替换成下面的内容

设置配置文件

<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.2.0"

          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.2.0 https://maven.apache.org/xsd/settings-1.2.0.xsd">

  <mirrors>

    <mirror>

      <id>aliyun-repo</id>

      <mirrorOf>central</mirrorOf>

      <url>https://maven.aliyun.com/repository/public</url>

      <releases>

        <enabled>true</enabled>

      </releases>

      <snapshots>

        <enabled>true</enabled>

      </snapshots>

    </mirror>

  </mirrors>

  <localRepository>D:/mavenRepositery</localRepository>

  <profiles>

    <profile>

      <id>default</id>

      <activation>

        <activeByDefault>true</activeByDefault>

      </activation>

      <repositories>

        <repository>

          <id>central</id>

          <url>https://repo.maven.apache.org/maven2</url>

          <releases>

            <enabled>true</enabled>

          </releases>

          <snapshots>

            <enabled>true</enabled>

          </snapshots>

        </repository>

      </repositories>

    </profile>

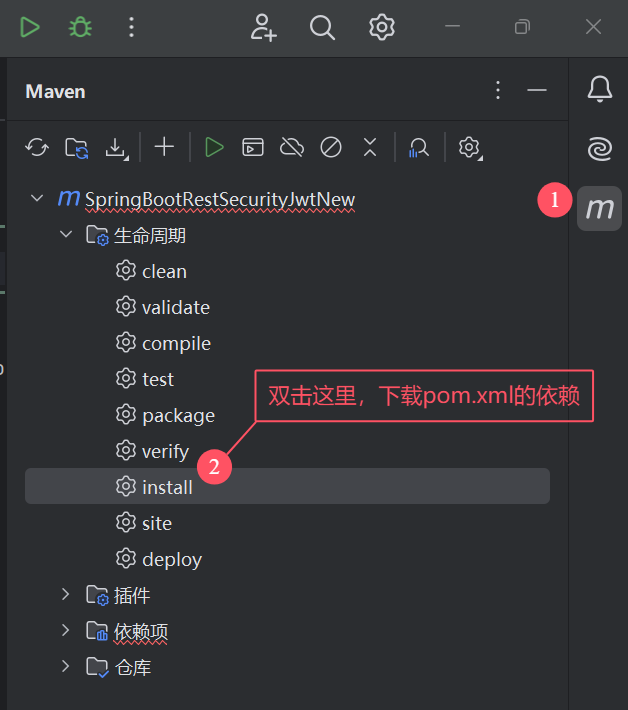
  </profiles>

</settings>

在ideaintellij设置中让maven指向它的bin. 并且配置文件指向上面这个setting.xml

### **下载依赖**

项目是依赖，pom.xml类似package.json。 类似npm install . 后台项目使用mvn install来安装。



## 前端开发工具vscode

### **安装插件**

安装gitlen

### **下载**

使用最新版

### **解决终端乱码**

终端如果报乱码执行： chcp 65001

## **github命令**

### **报证书过期错误**

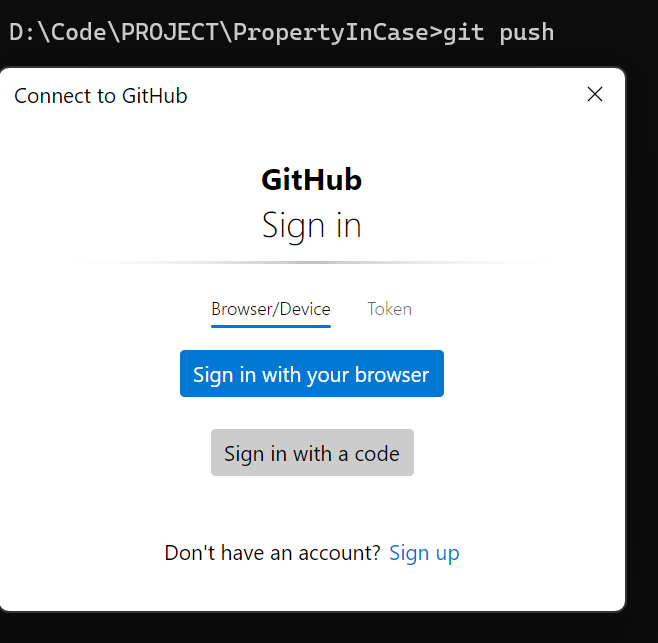
\WeGame>githttps://github.com/www617/PropertyInCase.gitcloneloning intoPropertyInCasfatal: unablehttps://github. com/www617/PropertyIncase. git/’: SsL certificate problem: unable to get local issuer certificate

解决办法：

git config --global http.sslVerify false

### **windows推送到github**

1. 已经使用token来设置windows, 不再使用用户名和密码。
2. 在终端中运行 git push 时，会出现弹出框， 使用sign in with your browser. 在浏览器上认证后就可以使用。



### **切换多个github账户**

此功能一般人忽略。

在windows上，如何切换自己的多个git账户。

控制面板====> 用户账户====>管理windows凭据====>git:https://github.com ： 删除。

删除后，在弹出上面的对话框时再重新登录。

### 常用命令

在项目根目录下执行

git add .

git commit -a -m 提交原因 --no-verify

git push

## **后端框架**

### 通过postman初始化数据

使用接口

http://localhost:9090/user/save

带上json数据

{

    "name":"root",

    "username":"root",

    "password":"root"

}

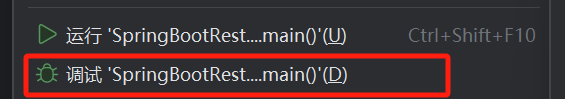
这一步，通常不用执行，由开发权限的团队执行。

### **运行springboot**

找到

src/main/java/com/app/raghu/SpringBootRestSecurityJwtNewApplication.java

点击右键：



## 前端框架ant-design-pro

ant.design 官网文档

<https://pro.ant.design/zh-CN/docs/overview>

### **本地源安装依赖包**

镜像（不需要自己安装）

npm install -g verdaccio

verdaccio

我在学校服务器上已经安装verdaccio服务，可以替代npm原来的阿里云镜像源

指向本地的verdaccio中转镜像池，第二个命令纯粹是查看，没有也可以

npm set registry <https://registry.npmmirror.com/> 或者

npm set registry <http://10.60.1.38:4873>

npm config list

npm install 或者

npm install --ignore-scripts

### 运行

先安装（忽略后期处理功能，我们用不到）

pnpm install --ignore-scripts --force

再运行

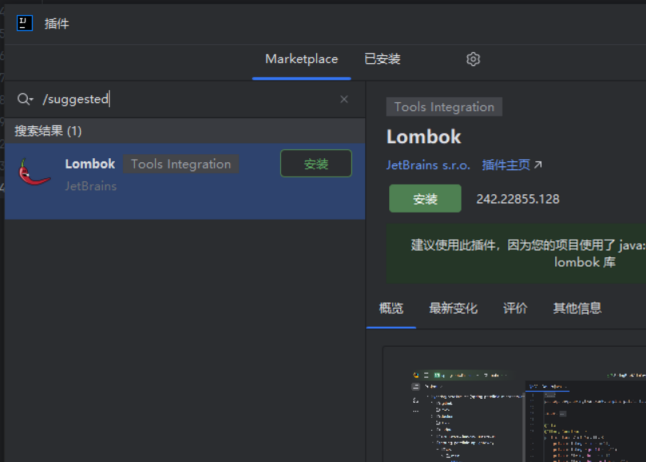
pnpm run dev

# **本项目环境清单**

后端开发搭配清单

最必须必要的是ideal community 2024.3,其它的可以略有差异，下面的组合是我测试的组合，但不是唯一正确的组合。不使用这个组合也可以，但需要自己测试。

1. java jdk 17
2. ideal community 2024.3 (2020的ideal 不支持java 17)
3. 数据库 mariadb
4. mavan3.9.9 或者直接配置 maven的 settings.xml, (settings.xml的内容见前面）
5. 插件：在第一次编译时，会出现提示，让安排lombok的插件，务必安装。



* **前端**

nodejs 20版本

vscode最新版本

# 需求分析

## 用户与权限

### **总体设计与需求说明**

用户与角色：一个用户可以拥有多个角色，一个角色也可以分配给多个用户。通过 user\_roles 表来实现多对多关系。

角色与权限：一个角色可以拥有多个权限，一个权限可以被多个角色拥有。通过 role\_permissions 表来实现多对多关系。

### 用户表

数据库设计： 用户表（users）

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 用户ID，主键，自增 |
| username | VARCHAR(50) | 用户名，唯一 |
| password | VARCHAR(255) | 密码 |
| email | VARCHAR(100) | 邮箱，唯一 |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |
| status | TINYINT | 用户状态（0: 禁用, 1: 启用） |

### **角色表**

角色表设计（roles）

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 角色ID，主键，自增 |
| name | VARCHAR(50) | 角色名称，唯一 |
| description | TEXT | 角色描述 |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |

### **权限表**

权限表设计（permissions）

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 权限ID，主键，自增 |
| name | VARCHAR(50) | 权限名称，唯一 |
| description | TEXT | 权限描述 |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |

### **用户角色关联表**

用户角色关联设计（user\_roles）

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| user\_id | INT | 用户ID，外键，指向users |
| role\_id | INT | 角色ID，外键，指向roles |
| PRIMARY KEY | (user\_id, role\_id) | 复合主键 |

### **角色权限关联表**

角色权限关联设计

角色权限关联表 (role\_permissions)

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| role\_id | INT | 角色ID，外键，指向roles |
| permission\_id | INT | 权限ID，外键，指向permissions |
| PRIMARY KEY | (role\_id, permission\_id) | 复合主键 |

## 入库与流转

### **总体设计**

数据表关系说明

物料表（items）：存储系统管理的所有物料信息，每个物料有唯一的 SKU 编号。

仓库表（warehouses）：存储所有仓库的信息，每个仓库可能有自己的管理者、地址等信息。

库存表（inventory）：记录每个物料在每个仓库中的库存数量，通过 warehouse\_id 和 item\_id 进行唯一组合约束。

调拨单表（transfer\_orders）：记录不同仓库之间的调拨操作，保存调拨来源仓库、目标仓库、调拨状态等信息。

调拨详情表（transfer\_order\_items）：存储具体调拨单中的物料明细，包含物料及调拨的数量。

调拨流程示例

创建调拨单：

从源仓库创建一个调拨单，将物料调拨至目标仓库。

调拨单的状态可能从 “已创建” 到 “已发货” 再到 “已完成”。

更新库存：

调拨完成后，源仓库的库存数量减少，目标仓库的库存数量增加。

调拨查询：

通过 transfer\_orders 和 transfer\_order\_items 查询特定调拨单的详细信息，追踪每次调拨的物料和数量。

### **可扩展功能**

1. **批次管理**：可以增加批次号和生产日期等字段，用于追踪同一物料的不同批次。
2. **库存预警**：在库存表中添加最低库存量，当库存低于该值时触发预警提醒。
3. **调拨审核流程**：可以引入调拨单的审核和审批流程，增加额外的状态，如“待审核”、“审核通过”等。

这个设计可以帮助有效管理物料的库存和调拨过程，支持不同仓库之间的物资调动和库存量的实时更新。

### 物料表

物料表设计：ruku\_items

用于存储所有管理的物料或商品信息。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 物料ID，主键，自增 |
| name | VARCHAR(100) | 物料名称 |
| sku\_code | VARCHAR(50) | 物料编号或SKU代码，唯一 |
| category | VARCHAR(50) | 物料类别（如：电子产品、服装等） |
| description | TEXT | 物料描述 |
| unit | VARCHAR(10) | 计量单位（如：件、千克等） |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |

### 仓库表

仓库表设计 (warehouses)

用于存储所有仓库的基本信息。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 仓库ID，主键，自增 |
| name | VARCHAR(100) | 仓库名称 |
| location | VARCHAR(255) | 仓库地址 |
| capacity | INT | 仓库容量（可选） |
| manager\_id | INT | 仓库管理员ID（关联用户表） |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |

### 库存表

库存表设计 (inventory)

用于存储某个仓库中某个物料的当前库存量。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 库存ID，主键，自增 |
| warehouse\_id | INT | 仓库ID，外键，指向 warehouses |
| item\_id | INT | 物料ID，外键，指向 items |
| quantity | DECIMAL(10, 2) | 库存数量 |
| min\_quantity | DECIMAL(10, 2) | 最低库存预警值（可选） |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |
| UNIQUE (warehouse\_id, item\_id) | 组合唯一键 |  |

### **调拨单**

调拨单表设计 (transfer\_orders)

用于记录库存调拨的订单信息，包含调拨的日期、状态等信息。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 调拨单ID，主键，自增 |
| transfer\_code | VARCHAR(50) | 调拨单编号，唯一 |
| source\_warehouse\_id | INT | 调拨源仓库ID，外键，指向 warehouses |
| target\_warehouse\_id | INT | 调拨目的仓库ID，外键，指向 warehouses |
| status | VARCHAR(20) | 调拨状态（如：已创建、已发货、已完成） |
| created\_at | DATETIME | 调拨单创建时间 |
| updated\_at | DATETIME | 调拨单更新时间 |

### **调拨详情表**

调拨详情表设计 (transfer\_order\_items)

用于存储调拨单中的具体物料及其数量。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 调拨详情ID，主键，自增 |
| transfer\_order\_id | INT | 调拨单ID，外键，指向 transfer\_orders |
| item\_id | INT | 物料ID，外键，指向 items |
| quantity | DECIMAL(10, 2) | 调拨数量 |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |

### **仓库出入库**

表财物出入库表（inventory\_movements）

记录财物的出入库操作。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 主键，自增 |
| item\_id | INT | 财物ID，外键，指向items表 |
| case\_id | INT | 案件ID，外键，指向case\_details表 |
| movement\_type | VARCHAR(20) | 操作类型（入库、出库、调拨） |
| quantity | INT | 数量 |
| from\_location | VARCHAR(100) | 源位置 |
| to\_location | VARCHAR(100) | 目标位置 |
| handled\_by | INT | 经办人ID，外键，指向users表 |
| approved\_by | INT | 批准人ID，外键，指向users表 |
| created\_at | DATETIME | 操作时间 |

## 档案管理

### **组成员分工表**

### 总体设计与需求分析

针对与案件相关的档案管理系统，数据库设计需要能够管理与案件相关的所有文档、图片等文件，并与案件及其相关实体（如客户、律师等）进行关联。这个设计不仅要存储文档和图片的元数据，还要管理其与案件的关联。

核心需求

案件管理：每个档案与特定案件关联，一个案件可能有多个相关文档和图片。

文档与图片的管理：支持上传和存储案件相关的文件和图片。

分类与标签：支持对文件进行分类和标记，便于检索和管理。

权限控制：确保只有特定用户（如负责该案件的律师或相关人员）能够访问特定档案。

文件历史：记录文件的历史版本，以便追溯文件的修改和变更。

### 案件表

案件表 (cases)

用于存储案件的基本信息，每个案件都有一个唯一的编号。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 案件ID，主键，自增 |
| case\_number | VARCHAR(50) | 案件编号，唯一 |
| title | VARCHAR(255) | 案件标题 |
| description | TEXT | 案件描述 |
| status | VARCHAR(50) | 案件状态（如：进行中、已结案） |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |

### 文件表

文件表 设计(file\_case\_files)

用于存储与案件相关的文件，包括文档、图片等的元数据。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 文件ID，主键，自增 |
| case\_id | INT | 案件ID，外键，指向cases表 |
| name | VARCHAR(255) | 文件名称 |
| path | VARCHAR(500) | 文件存储路径或URL |
| size | BIGINT | 文件大小（字节数） |
| file\_type | VARCHAR(50) | 文件类型（如image/jpeg、application/pdf） |
| uploaded\_by | INT | 上传者ID，外键，指向用户表 |
| created\_at | DATETIME | 上传时间 |
| updated\_at | DATETIME | 更新时间 |
| description | TEXT | 文件描述（可选） |
| version | INT | 文件版本号 |
| is\_current | BOOLEAN | 是否为当前版本（用于版本控制） |

### **文件分类表**

文件分类表 (file\_categories)

用于对文件进行分类管理，比如法律文档、证据图片、判决书等。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 分类ID，主键，自增 |
| name | VARCHAR(100) | 分类名称 |
| description | TEXT | 分类描述 |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |

### **文件-分类关联**

文件-分类关联表 (file\_category\_relations)

用于关联文件与分类，实现文件的分类管理。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 关联ID，主键，自增 |
| file\_id | INT | 文件ID，外键，指向case\_files表 |
| category\_id | INT | 分类ID，外键，指向file\_categories表 |
| created\_at | DATETIME | 创建时间 |

### **文件标签**

文件标签表 (file\_tags)

用于存储文件的标签信息，标签是一种灵活的方式来对文件进行额外的标记，例如“机密”、“优先处理”等。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 标签ID，主键，自增 |
| name | VARCHAR(50) | 标签名称 |
| created\_at | DATETIME | 创建时间 |

### **文件标签关联表**

文件-标签关联表 (file\_tag\_relations)

用于关联文件与标签，实现灵活的文件标记和管理。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 关联ID，主键，自增 |
| file\_id | INT | 文件ID，外键，指向case\_files表 |
| tag\_id | INT | 标签ID，外键，指向file\_tags表 |
| created\_at | DATETIME | 创建时间 |

### **用户权限表**

户权限表 (case\_file\_permissions)

用于控制用户对特定文件的访问权限，确保只有相关用户可以查看或编辑文件。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 关联ID，主键，自增 |
| file\_id | INT | 文件ID，外键，指向case\_files表 |
| user\_id | INT | 用户ID，外键，指向用户表 |
| permission\_type | VARCHAR(50) | 权限类型（如：查看、编辑、下载） |
| created\_at | DATETIME | 创建时间 |

## 案情与流转

### **总体设计与需求**

1. 案情表（case\_details）：管理案件的基本信息，包括案件编号、案件标题、描述和当前所在法院（current\_court\_id）。该表关联案件的流转历史、档案和涉案财物。
2. 法院表（courts）：存储法院的信息，每个案件关联到一个法院。法院信息用于案件的流转操作。
3. 案件流转表（case\_transfers）：记录案件从一个法院到另一个法院的转移情况，包含转移的时间、转移原因、转移的原法院和目标法院。
4. 案件涉案财物表（case\_items）：关联案件和物品，用于跟踪每个案件所涉及的财物，并记录财物的描述和数量。
5. 案件档案关联表（case\_files）：存储案件和档案的关联信息，关联到文件表，方便追踪案件所涉及的所有文件和档案。

### 案情表

案情表 (case\_details)

用于存储案件的基本信息。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 案情ID，主键，自增 |
| case\_number | VARCHAR(50) | 案件编号，唯一 |
| title | VARCHAR(255) | 案件标题 |
| description | TEXT | 案件描述 |
| status | VARCHAR(50) | 案件状态（如：在审、结案等） |
| current\_court\_id | INT | 当前法院ID，外键，指向courts表 |
| created\_at | DATETIME | 案件创建时间 |
| updated\_at | DATETIME | 案件更新时间 |
| created\_by | INT | 案件创建人ID，外键，指向users表 |
| updated\_by | INT | 案件更新人ID，外键，指向users表 |

### **法院表**

法院表 (courts)

用于存储法院信息，支持案件流转。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 法院ID，主键，自增 |
| name | VARCHAR(100) | 法院名称 |
| address | VARCHAR(255) | 法院地址 |
| contact\_number | VARCHAR(20) | 联系电话 |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |

### **案件流转表**

案件流转表 (case\_transfers)

记录案件的流转历史，便于追踪案件从一个法院到另一个法院的转移情况。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 流转ID，主键，自增 |
| case\_id | INT | 案件ID，外键，指向case\_details表 |
| from\_court\_id | INT | 原法院ID，外键，指向courts表 |
| to\_court\_id | INT | 目标法院ID，外键，指向courts表 |
| transferred\_by | INT | 操作人ID，外键，指向users表 |
| transfer\_date | DATETIME | 流转时间 |
| reason | TEXT | 流转原因 |
| created\_at | DATETIME | 创建时间 |

### **涉案财物表**

案件涉案财物表 (case\_items)

用于管理案件所涉及的仓库物品，方便在案件管理中查看涉案财物。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 涉案物品ID，主键，自增 |
| case\_id | INT | 案件ID，外键，指向case\_details表 |
| item\_id | INT | 物品ID，外键，指向items表 |
| quantity | INT | 涉案物品数量 |
| description | TEXT | 涉案物品描述 |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |

### **案件档案表**

案件档案关联表 (case\_files)

将案件与档案管理中的文件关联，便于查看每个案件的相关档案。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 关联ID，主键，自增 |
| case\_id | INT | 案件ID，外键，指向case\_details表 |
| file\_id | INT | 文件ID，外键，指向files表 |
| created\_at | DATETIME | 创建时间 |

## 现场管理

### 现场管理总体分析与需求

在涉案财物的现场管理中，确保财物的安全、清点、存储环境和出入库记录尤为重要。以下是一些常见的现场管理方面的建议和思路：

1. 出入库管理

入库管理：涉案财物在移交到仓库时，需要进行入库登记，包括财物的类型、数量、来源、案件编号等详细信息。

出库管理：当财物需要出库时（例如用于法庭审理），需要记录出库时间、用途、经办人及批准人信息，并在财物返还时进行入库登记。

流转管理：如财物在不同仓库或存储地点之间移动，也需详细记录流转信息，包括原仓库、目标仓库、运输人员及车辆等信息。

2. 存储位置管理

存储区域划分：根据财物的特性（例如易腐、易燃物品），在仓库中划分不同的存储区域，确保适当的存储条件。

位置编码：给仓库中的每个存储位置编码，确保可以快速定位财物。

货架/位置标记：在实际存储过程中，将每件财物分配到具体的货架或位置，并记录这些位置，便于盘点和取用。

3. 安全管理

安保措施：确保仓库有足够的安保设备和系统，如监控摄像头、安防系统、门禁卡等。

权限控制：只有经过授权的人员可以进入仓库，且不同人员的操作权限可能不同，例如只有特定权限的人员可批准出库或移交。

夜间巡逻：安排值班人员或安保巡逻，确保财物在非工作时间也处于安全状态。

4. 环境控制

温湿度监控：对于易损或对环境要求较高的涉案物品（如文件、电子设备等），需要监控仓库的温湿度，确保符合存储要求。

防火防潮措施：在仓库中配置灭火器、吸湿剂、防潮布等，并定期检查设备状态。

5. 财物盘点

定期盘点：安排固定时间（如每月或每季度）对仓库内的财物进行盘点，核对实物和系统记录的一致性，发现差异及时查明原因。

突击抽查：随机选择某些涉案财物进行抽查，以确保财物真实存在且状态正常。

6. 库存状态管理

财物状态：记录每件涉案财物的状态，例如“正常”、“损坏”、“待维修”等。

定期检查：特别是对于可能变质的物品，定期检查其状态并更新系统中的状态记录。

7. 涉案财物的报废与处置

报废管理：对于已审结且无保留价值的财物，经过审批后可进行报废处理。需要记录报废原因、日期及审批人信息。

处置流程：对于无法返还或销毁的涉案物品，需有规范的处置流程，包括物品的销毁或移交第三方机构的记录。

8. 记录和审核

操作日志：记录每一项操作，如入库、出库、盘点等，确保出现问题时可以追溯。

定期审核：由独立的内部审计人员对涉案财物管理流程和仓库状况进行审核，确保没有漏洞。

9. 移动端管理支持

现场扫码入库：在入库时，可使用移动端设备扫描财物标签条码，以便快速录入、确认或查找财物。

实时查询和更新：通过移动设备随时查询财物状态和位置，更新盘点数据，减少人工录入的误差。

10. 应急预案

应急处理：制定紧急情况下（如火灾、盗窃）的应急处理流程，确保财物尽可能安全。

数据备份和恢复：定期对涉案财物的数据进行备份，确保在出现系统故障时可以快速恢复数据。

### 环境状态表

库存环境记录表（environment\_conditions）

用于记录仓库的温湿度等环境条件。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 主键，自增 |
| location\_id | INT | 位置ID，外键，指向仓库表 |
| temperature | DECIMAL(5,2) | 温度 |
| humidity | DECIMAL(5,2) | 湿度 |
| recorded\_at | DATETIME | 记录时间 |
|  |  |  |

### **财物状态表**

财物状态表（item\_statuses）

记录涉案财物的状态及检查记录。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 主键，自增 |
| item\_id | INT | 财物ID，外键，指向items表 |
| status | VARCHAR(50) | 状态（如：正常、损坏等） |
| checked\_at | DATETIME | 检查时间 |
| notes | TEXT | 备注或描述 |
| checked\_by | INT | 检查人ID，外键，指向users表 |

### **访问权限表**

访问权限表（access\_permissions）

记录人员的仓库访问权限，控制权限等级。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 主键，自增 |
| user\_id | INT | 用户ID，外键，指向users表 |
| location\_id | INT | 位置ID，外键，指向storage\_locations表 |
| access\_level | VARCHAR(20) | 访问等级（如：管理员、普通人员） |
| valid\_from | DATETIME | 权限有效期开始 |
| valid\_until | DATETIME | 权限有效期结束 |
| created\_at | DATETIME | 创建时间 |

### **访问日志表**

访问日志表（access\_logs）

记录进入和离开仓库的日志，确保访问信息的可追溯性。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 主键，自增 |
| user\_id | INT | 用户ID，外键，指向users表 |
| location\_id | INT | 仓库位置ID，外键，指向storage\_locations表 |
| access\_type | VARCHAR(20) | 访问类型（进入、离开） |
| timestamp | DATETIME | 访问时间 |
| access\_granted | BOOLEAN | 是否允许进入（记录授权情况） |
| reason | TEXT | 进入原因或备注 |

### **监控设备表**

监控设备表（monitoring\_devices）

记录仓库内的监控设备信息，包括设备类型、位置及状态。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 主键，自增 |
| device\_type | VARCHAR(50) | 设备类型（摄像头、传感器等） |
| location\_id | INT | 设备所在位置ID，外键，指向storage\_locations表 |
| status | VARCHAR(20) | 设备状态（正常、故障等） |
| installation\_date | DATETIME | 安装日期 |
| last\_maintenance | DATETIME | 上次维护时间 |
| notes | TEXT | 备注 |

### **安全告警表**

安全告警表（security\_alerts）

记录发生的安全告警事件，如未授权访问、温湿度异常等。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 主键，自增 |
| alert\_type | VARCHAR(50) | 告警类型（如：未授权访问、温度异常） |
| location\_id | INT | 告警发生位置ID，外键，指向storage\_locations表 |
| timestamp | DATETIME | 告警时间 |
| severity | VARCHAR(20) | 告警严重等级（低、中、高） |
| description | TEXT | 告警描述 |
| resolved | BOOLEAN | 是否已解决 |
| resolved\_by | INT | 处理人ID，外键，指向users表 |
| resolution\_notes | TEXT | 处理说明 |

### **巡查记录表**

巡查记录表（security\_patrols）

记录安保人员的巡查信息，确保日常巡查和安全监控。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 主键，自增 |
| patrol\_user\_id | INT | 巡查人员ID，外键，指向users表 |
| location\_id | INT | 巡查位置ID，外键，指向storage\_locations表 |
| patrol\_start | DATETIME | 巡查开始时间 |
| patrol\_end | DATETIME | 巡查结束时间 |
| notes | TEXT | 巡查备注 |
| issues\_found | BOOLEAN | 是否发现问题 |
| issues\_description | TEXT | 问题描述 |

## 大数据信息

### 总体设计

关于涉案财物管理系统的大数据页面，可以聚焦在数据的分析、统计和可视化，以帮助用户更全面地了解系统运行情况和涉案财物的安全管理。大数据页面的内容一般可以包含以下几个主要模块：

1. 涉案财物总览

财物数量统计：显示涉案财物的总数量、不同状态（在库、出库、流转中、待处置）的财物数量。

财物类型分布：根据财物类型（如文件、电子设备、现金、贵重物品等）展示不同类别的数量和比例，帮助管理者了解主要物品类型。

案件分布：按案件类型或案件所属部门（如不同法院、公安分局等）展示涉案财物的数量分布。

财物存储区域分布：展示各仓库或存储区域的财物存放情况及其占用率，便于优化仓库管理。

2. 出入库和流转分析

出入库频率分析：展示某段时间内财物出入库的频率趋势，可用柱状图或折线图表示。

流转路径可视化：在地图上展示涉案财物的流转路径，跟踪财物从一个位置到另一个位置的流转情况，确保财物的安全性。

关键物品出入库记录：显示高价值物品或敏感物品的出入库记录，支持按日期筛选或分类别筛选，便于查看特定财物的活动情况。

3. 安全事件与告警统计

告警事件数量及趋势：展示各类告警事件的数量变化趋势（如温湿度异常、未授权访问等），可以帮助管理者关注安全问题的频发时间段或高频区域。

告警类型分布：按告警类型（未授权访问、设备故障、温湿度异常等）展示不同事件的数量和比例，便于管理者识别主要的安全风险来源。

告警响应时间：展示不同告警事件的平均响应时间，分析告警处理效率。

4. 巡查和监控记录分析

巡查频率与覆盖率：展示仓库各区域的巡查频率、巡查时间和覆盖率，确保巡查是否按要求执行。

安保人员活动统计：按安保人员统计巡查活动频率、区域覆盖范围及发现问题的数量，帮助评估安保工作有效性。

监控录像记录分析：展示各监控设备的状态（正常、故障等）及录制情况，确保监控系统的可靠性。

5. 存储环境监控

温湿度监控图表：实时展示仓库温湿度变化趋势，或按仓库/区域分布展示温湿度当前状态。

异常环境记录：显示温湿度超标或其他环境异常的记录，包括发生时间、仓库位置、处理情况等。

实时环境数据展示：实时显示仓库各区域的温湿度数据，可用于敏感物品的存储环境监控。

6. 财物盘点和库存状态

库存状态统计：展示当前库存的完整性，包括已盘点和未盘点的财物数量，以及盘点异常的物品数量。

盘点完成率与问题率：展示最近盘点的完成率和盘点过程中发现的问题率，帮助管理者了解仓库管理的规范性。

财物异常状态统计：如有破损、丢失、待维修等财物状态，可以按状态分类展示异常财物数量。

7. 报废与处置分析

财物报废数量统计：展示已报废和待报废的财物数量、报废原因、报废时间，帮助管理者掌握无用财物的处理情况。

财物处置记录：展示已处置财物的处置方式（如销毁、捐赠等）及处置时间，便于追溯财物的去向。

处置效率分析：展示不同类型财物的平均处置时间，分析处置流程是否高效。

8. 数据图表和可视化建议

柱状图/折线图：用于展示时间趋势，例如出入库频率、告警数量变化等。

饼图/环形图：用于显示比例数据，例如财物类型分布、告警类型分布等。

热力图：可在仓库平面图上以热力图展示不同区域的财物存放情况和告警频次。

仪表盘（Dashboard）：将关键指标集中展示，例如财物总数、当前告警数量、环境异常次数等，方便管理者快速了解系统总体情况。

## 业务入口

### 系统字典设计

**一个系统字典功能，用于保存一些全局设置或参数**

1. 系统的配置参数（如公司名称、联系信息等）
2. 一些常用的下拉菜单选项（如性别、部门、职位等）
3. 其他需要动态配置的参数

**字典分类表 (dictionary\_categories)**

用于保存不同类别的字典，比如系统设置类、业务参数类等。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 分类ID，主键，自增 |
| name | VARCHAR(100) | 分类名称，唯一 |
| description | TEXT | 分类描述 |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |

**字典项表 (dictionary\_items)**

用于存储具体的字典项，比如各类配置项的键值对。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 描述 |
| id | INT | 字典项ID，主键，自增 |
| category\_id | INT | 分类ID，外键，指向 dictionary\_categories |
| key | VARCHAR(100) | 字典项的键 (key)，唯一 |
| value | VARCHAR(255) | 字典项的值 (value) |
| description | TEXT | 字典项描述 |
| created\_at | DATETIME | 创建时间 |
| updated\_at | DATETIME | 更新时间 |

### 其它

1. 用户管理与权限控制

细粒度权限管理：支持按角色、用户或部门的权限分配，确保每个用户只能访问和操作其权限范围内的数据和功能。

登录认证与安全审计：包括多重身份认证（MFA）、登录日志、操作日志等，增强系统安全性并满足合规要求。

权限动态调整：可基于案件进展或人员角色变化自动更新权限，减少人工维护成本。

2. 案件与业务流程管理

案件流转管理：设计案件的流转流程图，从立案、流转、处理到结案，支持多个节点间的自动化流转管理。

业务流程管理（BPM）：通过灵活的流程管理引擎，定义和管理自定义工作流，比如财物审核流程、案件审批流程等，以适应不同需求。

自动化通知：案件关键环节支持系统通知或短信通知，提醒用户跟进案件进展。

3. 审批和任务管理

审批流程配置：支持自定义的审批流程配置，用于不同类型的业务（如物品报废、流转审批等）。

待办事项与提醒：显示用户需要处理的任务、到期的待办事项，支持邮件、短信或系统内通知提醒。

任务分配与跟踪：能够分配具体任务（如物品盘点、仓库巡查等）给用户，并实时跟踪完成情况。

4. 统计分析与报表中心

自定义报表：支持用户自定义查询条件，生成各类数据报表（如案件数量、财物状态、告警事件等），满足不同角色的需求。

导出与分享：支持将报表导出为PDF、Excel等格式，便于分享或归档。

高级数据分析：基于历史数据进行预测分析，如高峰期告警事件预测、仓库空间需求预测等，为业务决策提供支持。

5. 系统设置与配置中心

系统参数配置：允许管理员配置系统的关键参数，如告警阈值、温湿度标准、流程审批配置等，确保系统灵活性。

仓库和区域配置：支持动态添加、修改仓库和区域，满足扩展性需求。

业务字典：保存常用的业务数据字典，如案件类型、物品分类、处置方式等，保证数据一致性。

6. 审计与日志管理

操作日志：记录用户操作的详细信息，便于追溯和问题排查。

系统日志与告警日志：保存系统运行日志和告警日志，方便系统运维人员监控系统健康状态。

日志归档与导出：支持日志自动归档和导出，便于长期保存和合规要求。

7. 通知与消息中心

消息通知：支持在系统内消息、短信、邮件等方式进行通知。

告警提醒：当出现仓库温湿度异常、门禁异常等情况时，实时通知相关人员及时响应。

站内消息管理：记录所有系统通知和告警信息，便于用户回顾和查看历史通知。

8. 帮助与用户支持

帮助文档与操作指南：包含系统功能说明、操作步骤，帮助新用户快速上手。

问题反馈与技术支持：提供反馈渠道（如反馈表单、客服联系方式等），便于用户反馈问题。

版本更新日志：展示系统的更新和维护情况，让用户了解最新的功能和修复内容。

9. 接口与数据集成

API接口：提供标准API接口，支持与其他业务系统（如司法系统、ERP、OA系统等）进行数据集成。

第三方数据接入：接入外部数据源，如与公安、法院等部门系统进行数据同步或共享。

数据交换标准：设计数据交换格式和标准，确保不同系统间数据交换的兼容性。

10. 系统性能监控与维护

性能监控：实时监控系统性能，如页面加载速度、响应时间等，确保系统稳定性。

数据库备份与恢复：提供数据库自动备份和恢复机制，确保数据安全。

定期维护计划：记录系统的定期维护信息，确保数据完整性和系统性能。

11. 智能化与辅助决策支持

数据智能分析：利用机器学习算法，分析历史数据并生成预测报告，如财物滞留趋势、案件流转效率等。

推荐与警示系统：基于历史告警和环境数据，提供智能警示或仓库优化建议。

智能搜索与过滤：利用自然语言处理技术提供智能化搜索功能，支持用户快速查找案件、物品等信息。

# 技术要领

## 前后端对接

### 注意事项

不要直接使用Axios来接

### 前端service.ts纠正

通常我们请chatgpt写的文档，有一个service.ts文件。 它里面应该使用下面的组件：

import { request } from '@umijs/max';

并使用类似下面的方式来访问后端， 下面两个代码恰好阐述了get和post两种方式。

/\*\* 获取当前的用户 GET /api/user/getUser \*/

export async function currentUser(options?: { [key: string]: any }) {

const token = localStorage.getItem('token') || '';

return request<{

data: API.CurrentUser;

}>('/api/user/getUser', {

method: 'GET',

headers: {'Authorization': `${token}`, },

...(options || {}),

});

}

/\*\* 登录接口 POST /api/user/login \*/

export async function login(body: API.LoginParams, options?: { [key: string]: any }) {

return request<API.LoginResult>('/api/user/login', {

method: 'POST',

headers: {'Content-Type': 'application/json',},

data: body,

...(options || {}),

}).then((response) => {

if (response?.token) {

localStorage.setItem('token', response.token);

}

return response;

});

}

### 后端

后端依赖controller .

### 如何问chatgpt

比如你有axios的代码如下  
import axios from 'axios';

const API\_URL = '/api/dictionary\_categories';

export const DictionaryCategoriesService = {

  fetchList: async () => {

    const response = await axios.get(API\_URL);

    return response.data;

  },

  deleteById: async (id: number) => {

    await axios.delete(`${API\_URL}/${id}`);

  },

};

那么你需要象这样要求chat帮你改写：

下面是我原来的代码，它需要重新使用request来编写：  
import axios from 'axios';

const API\_URL = '/api/dictionary\_categories';

export const DictionaryCategoriesService = {

  fetchList: async () => {

    const response = await axios.get(API\_URL);

    return response.data;

  },

  deleteById: async (id: number) => {

    await axios.delete(`${API\_URL}/${id}`);

  },

};

我系统框架中使用

import { request } from '@umijs/max';

来和后端对接，它的get和post的用法如下：  
import { request } from '@umijs/max';

export async function currentUser(options?: { [key: string]: any }) {

  const token = localStorage.getItem('token') || '';

  return request<{

    data: API.CurrentUser;

  }>('/api/user/getUser', {

    method: 'GET',

    headers: {

      'Authorization': `${token}`,

    },

    ...(options || {}),

  });

}

export async function login(body: API.LoginParams, options?: { [key: string]: any }) {

  return request<API.LoginResult>('/api/user/login', {

    method: 'POST',

    headers: {

      'Content-Type': 'application/json',

    },

    data: body,

    ...(options || {}),

  }).then((response) => {

    if (response?.token) {

      localStorage.setItem('token', response.token);

    }

    return response;

  });

}

请你根据我的模板，帮我重新写上面axios的代码，并注意带上token. 谢谢