## 1. 项目简介

### 1.1 项目基本信息

项目的项目名为**软件管理平台**。（下文简称项目）

项目的基本类型是一个**针对工业软件的在线授权平台**，其旨在工业软件的**发布、更新、下载、升级(手动升级和自动升级)**  
**和授权。用户可以在此平台进行工业软件的搜索，购买，下载，升级**以及自身的**硬件指纹管理**，也可以作为开发者自行上传软件，管理软件版本，为用户提供了更多选择。管理员可以进行用户上传的软件的审核，以及对软件库的各个软件的增删改查处理。

项目还在软件授权方面进行了设计， 让所有经由此平台管理的软件都被保护， 尽可能避免了盗版软件以及核心代码库盗用的问题。

项目实现了在线和离线两种情况下的授权认证。

### 1.2 项目来源

项目出自广东工业大学QG工作室2024年暑期训练营的中期考核，为前后端分离开发的综合Web项目。

### 1.3 项目基本技术栈以及使用框架

**后台**

|  |  |
| --- | --- |
| 技术栈 | 版本 |
| Springboot | 2.7.6 |
| jdk | 1.8 |
| mybatis | 2.3.0 |
| mysql | 本地版本 |
| redis | windows下的 3.2.0 |
| fastjson2 | 2.0.9.graal |

**前端**

* **React** (^18.3.1): 用于构建用户界面和组件。
* **React Router DOM** (^6.25.1): 实现客户端路由和页面导航。
* **样式和组件库**
  + **Ant Design (antd)** (^5.19.3): 提供了一整套企业级 UI 组件。
  + **Styled Components** (^6.1.12): 允许使用 CSS-in-JS 进行组件样式管理。
* **数据处理与请求**
  + **Axios** (^1.7.2): 用于处理 HTTP 请求。
* **构建工具**
  + **CRACO** (^7.1.0): 配置工具，允许对 Create React App 进行定制化配置。
  + **CRACO Less** (^3.0.1): 支持 Less 文件的解析和编译。
* **测试**
  + **Jest** 和 **React Testing Library**: 用于组件测试和断言。
* **开发与构建工具**
  + **Babel Plugin Import** (^1.13.8): 按需加载 Ant Design 组件以减少打包体积。
  + **CSS Loader** (^7.1.2) 和 **Style Loader** (^4.0.0): 用于处理 CSS 文件的加载和注入。
* **代理**
  + 配置代理服务器：http://47.113.224.195:31108。

该技术栈支持高效的开发流程、丰富的 UI 组件、可靠的数据处理和可视化功能，以及灵活的构建和测试工具。

### 1.4 项目意义

此项目是广东工业大学QG工作室2024级前端组与后台组实习生的第一个大型合作项目。此项目极大程度地检验了这两周的知识学习后我们实习生的学习成果，同时为我们打开了前后端分离开发Web项目的大门。项目要求使用现阶段企业常用技术栈进行项目开发，使用这些新兴技术与生态丰富的技术也为我们将来就业和升学提供了宝贵的经验。

## 2. 设计思路

在设计项目时，我们使用**页面驱动设计**的方式推进整个项目的接口以及数据库的设计。

### 2.1 页面概述

#### 2.1.1 登录注册界面

登录注册界面，由于手机发送验证码模块需要一些资质验证，实现起来有些困难，故采用了邮箱发送验证码方案。

此处可确定用户表user。

#### 2.1.2 主页

登陆后，主页将展示用户的头像、用户名以及一些软件简略信息。软件简略信息存储在软件表software中。

#### 2.1.3 软件详情页

点击软件，可进入软件详情页进行查看，其中有软件的简略描述、对应普通版和高级版（如果有）的详细描述，以及包括历史版本的下载地址。为了保存软件下载地址以及针对特定版本的特定版本号的详细描述，可确定软件-版本-下载表software-version-download。

而为了防止用户没有购买软件就进行下载，建立用户-软件-权限表user\_software\_auth记录用户对当前软件的下载是否合法。

用户若可以合法下载软件，后台将会记录用户的下载版本，以便在后面进行实时的版本更新提醒。此处对应表user\_software\_download。

#### 2.1.4 软件购买页

进入此页面，将会展示用户未购买过的软件，用户可以进行软件的选择与购买，支持批量购买。

#### 2.1.5 软件购买确认订单页面

用户在软件购买页购买一批软件后，进入确认订单页面。当模拟的支付进行完毕后，将会生成两条信息。

* 订单信息 对应表order\_tb
* 用户授权信息 对应表user\_software\_license

#### 2.1.6 用户上传软件以及硬件指纹信息

用户可以作为开发者上传软件，上传的软件信息将暂存在software\_info\_temp表中。

用户也可以上传、管理自己的硬件指纹信息，上传的指纹数据也需要一张表存储，即user\_hardware表

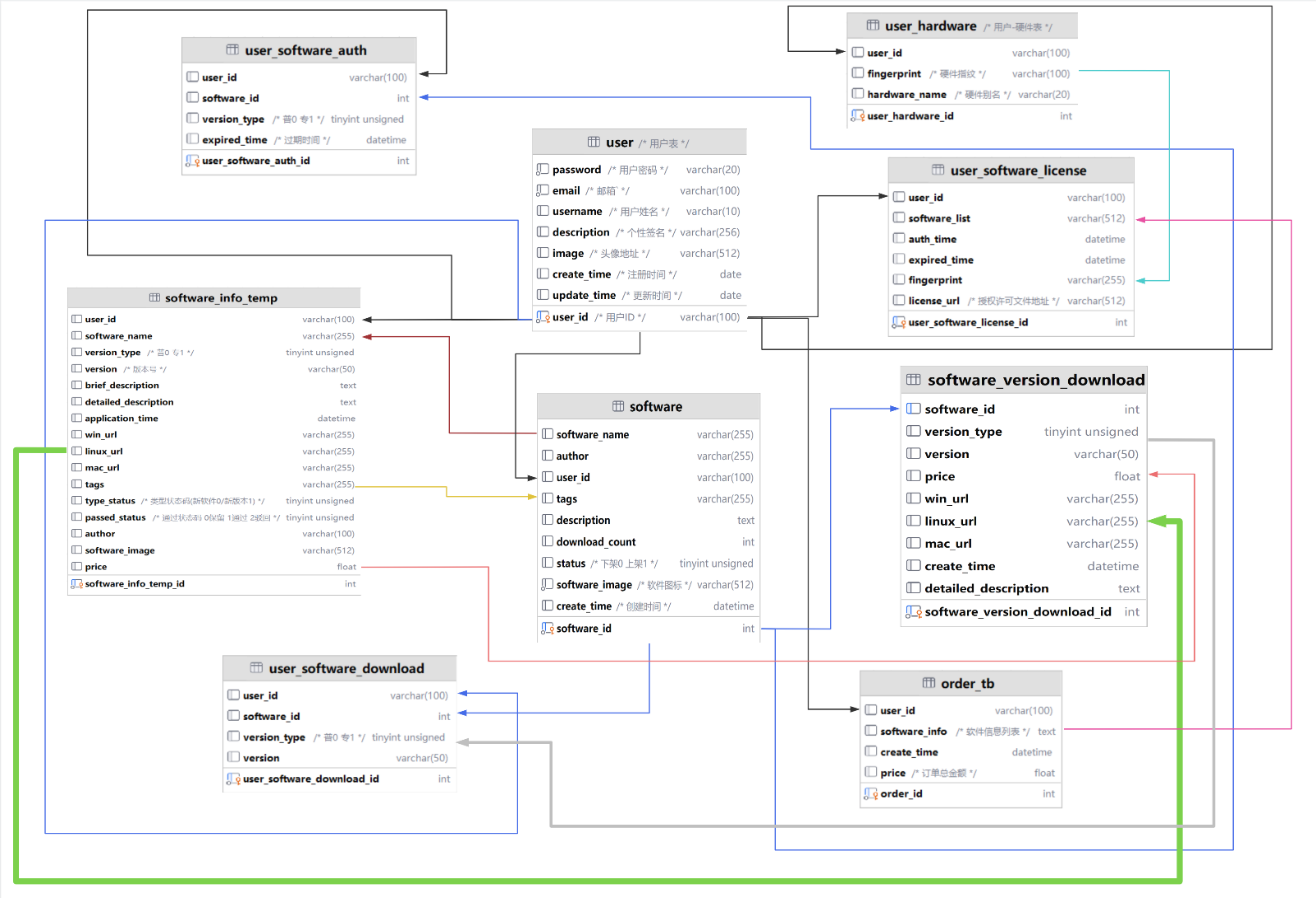
#### 2.1.7 用户个人主页

在个人主页，用户可以查看自己上传过的软件，购买过的软件，以及修改自己的个人信息。

#### 2.1.8 管理员审核软件信息

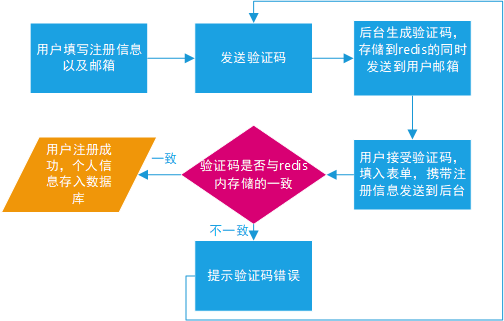
管理员可以查看用户申请上架的软件，服务器提供下载地址和详细软件信息。管理员有对软件信息的所有操作权，包括软件信息以及上下架状态。

#### 2.1.9 表结构

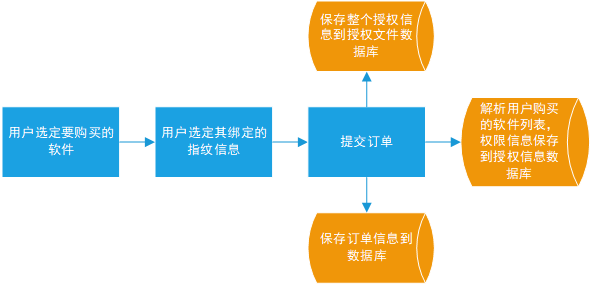


### 2.2 业务逻辑流程图

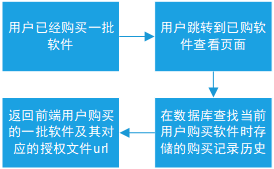
#### 2.2.1 用户登录注册



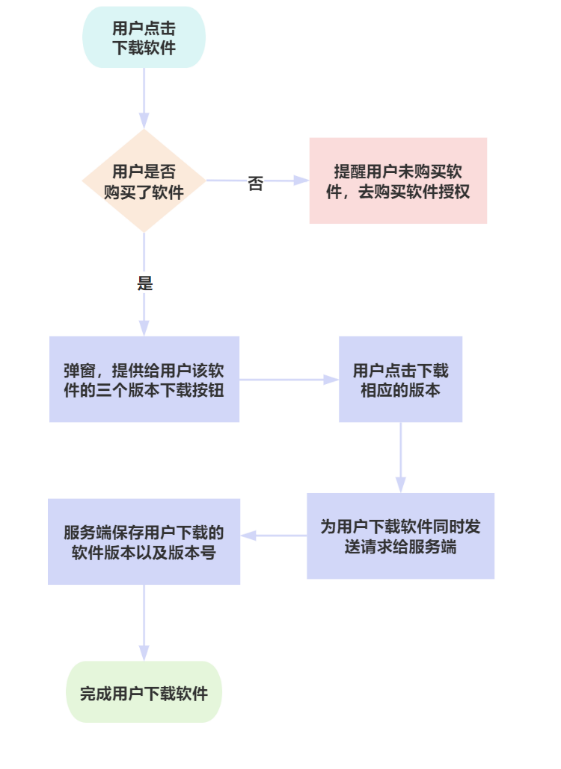
#### 2.2.2 用户购买软件



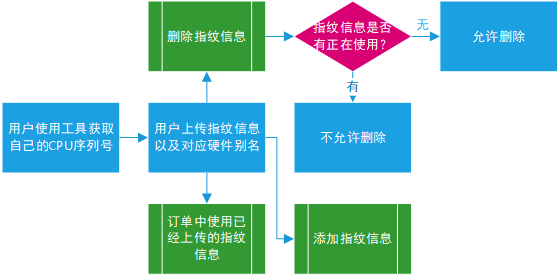
#### 2.2.3 用户获取授权文件



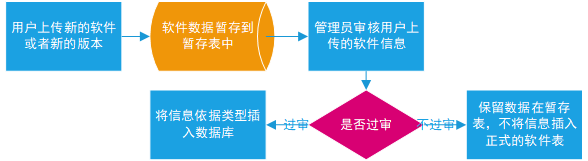
#### 2.2.4 用户下载软件



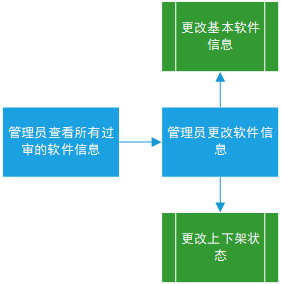
#### 2.2.5 用户管理指纹信息



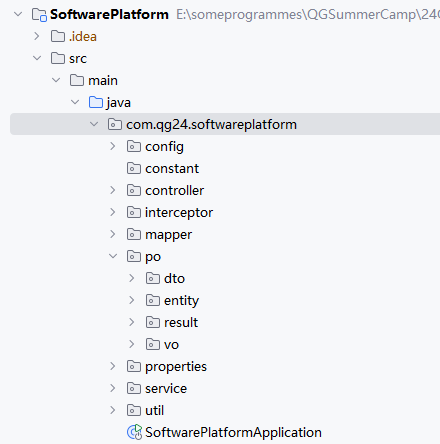
#### 2.2.6 用户上传软件，管理员审核软件



#### 2.2.7 管理员更改软件信息



### 2.3 项目分包



## 3. 详细设计

此模块，将根据接口的分类进行项目各个接口的详细设计说明。

### 3.1 AccountController /account

这个控制器用来负责账号注册登录以及一些个人信息获取。

#### 3.1.1 /sendVerificationCode 发送验证码

前端提供一个邮箱地址。服务器拿到地址后，先生成六位随机的验证码，再调用Spring提供的mail服务发送验证码到给定的邮箱地址。同时，服务器将拼接一个与此邮箱唯一关联的键值对，值即为生成的六位随机验证码，存入redis中。

#### 3.1.2 /register 确认注册

前端携带用户邮箱、验证码、密码信息发送到服务器。

服务器接受信息，首先判断此邮箱是否已被被注册。通过邮箱查询数据库，没有数据返回的话则说明是新用户注册。

通过新用户认证后，拿到前端传来的邮箱验证码，与redis中存储的验证码进行比对，如果通过验证则注册成功，向用户表插入数据。

#### 3.1.3 /login 登陆

获取前端用户登陆的账密，其中也携带角色状态码。根据状态码判断识别管理员登录还是用户登录。

如果是管理员登录，则会将账号作为用户ID，与密码配套查询数据库，查找得到用户则返回管理员ID以及token

如果是用户登录，则会将账号作为用户邮箱，与密码配套查询数据库，查找得到用户则返回用户ID以及token

如果登录失败，则返回状态码为0的结果，提示登录失败。

#### 3.1.4 /homePageUserInfo 主页用户信息回显

前端发送用户ID到服务器请求用户的昵称、头像以及个人描述

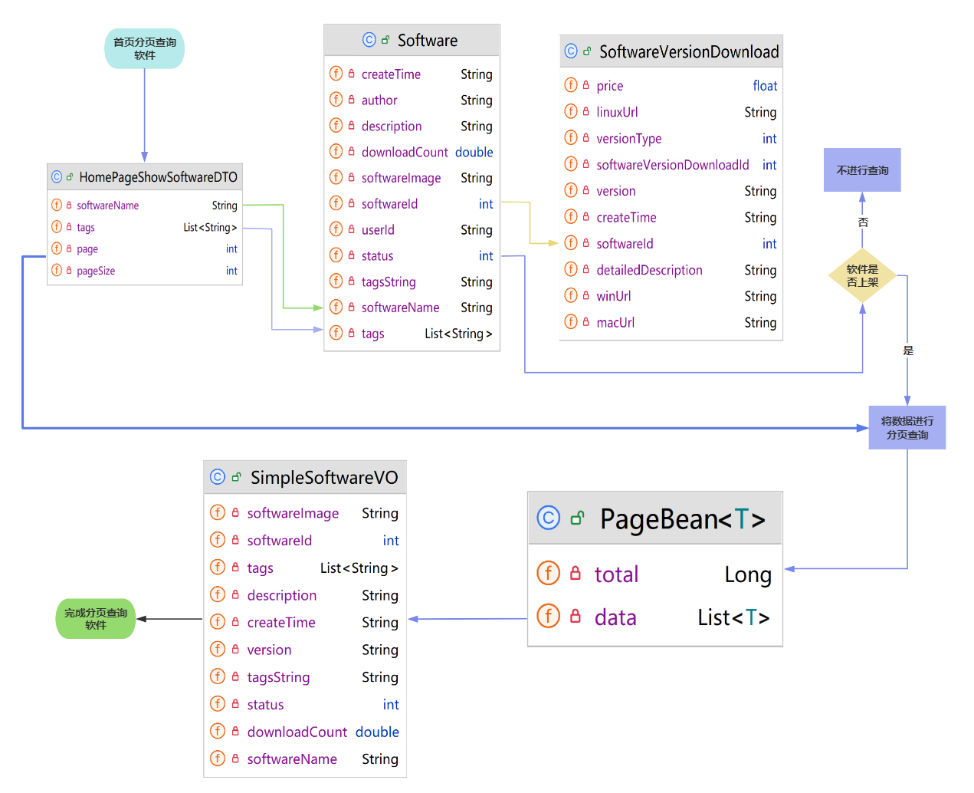
#### 3.1.5 /updatePassword 修改密码

前端要先通过/sendVerificationCode 发送验证码 接口放行用户，此接口才能被调用。用户输入自己的邮箱和新密码，后台接受之后进行密码的修改。

### 3.2 SoftwareController /software

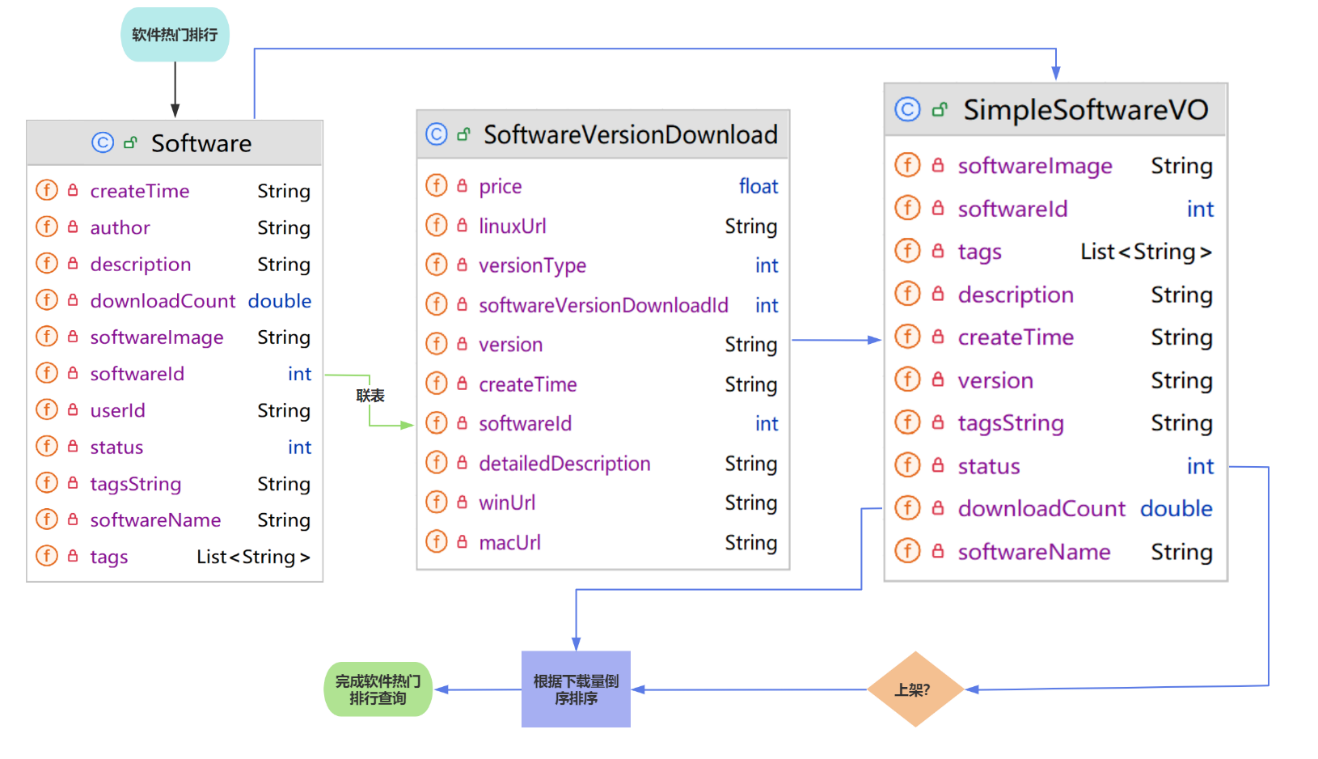
#### 3.2.1 /homePageShowSoftware 首页分页条件查询软件信息

首页支持分页、名字模糊查询以及标签模糊查询。



#### 3.2.2 /softwareRanking 软件热门排行

通过下载量对查询的软件进行排序。

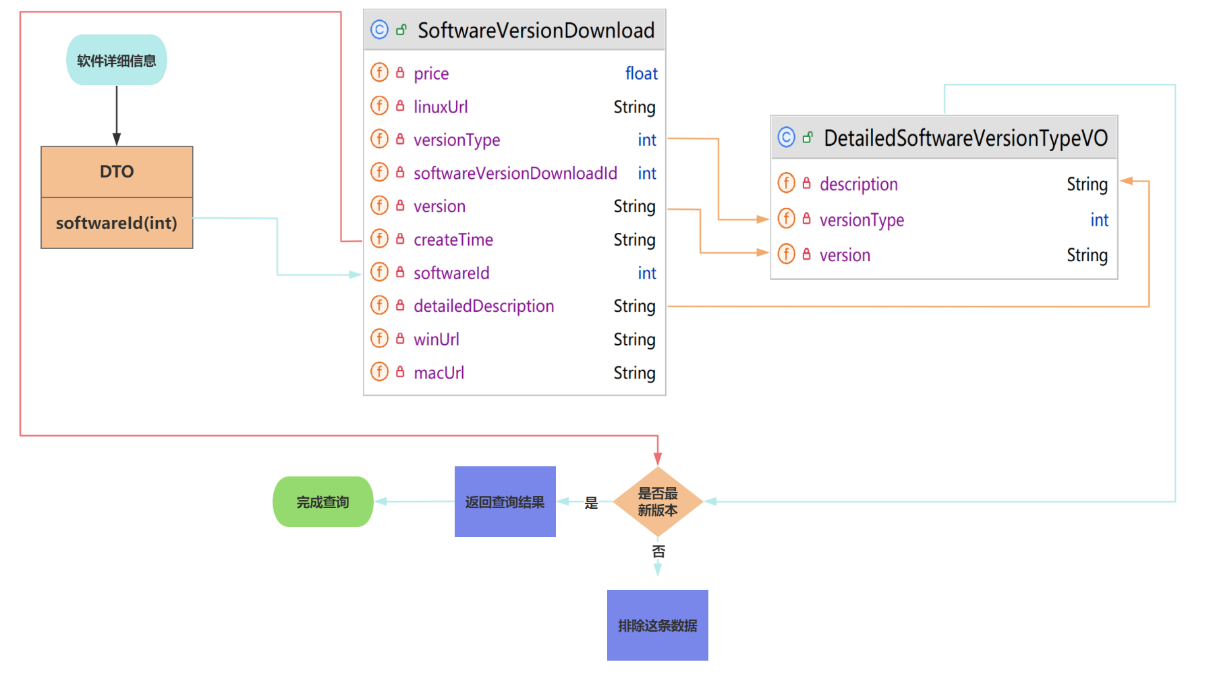


#### 3.2.3 /basicSoftwareInfo 基本软件信息

前端发送软件Id到服务器，服务器直接查询software表，获取基本软件信息返回给前端。

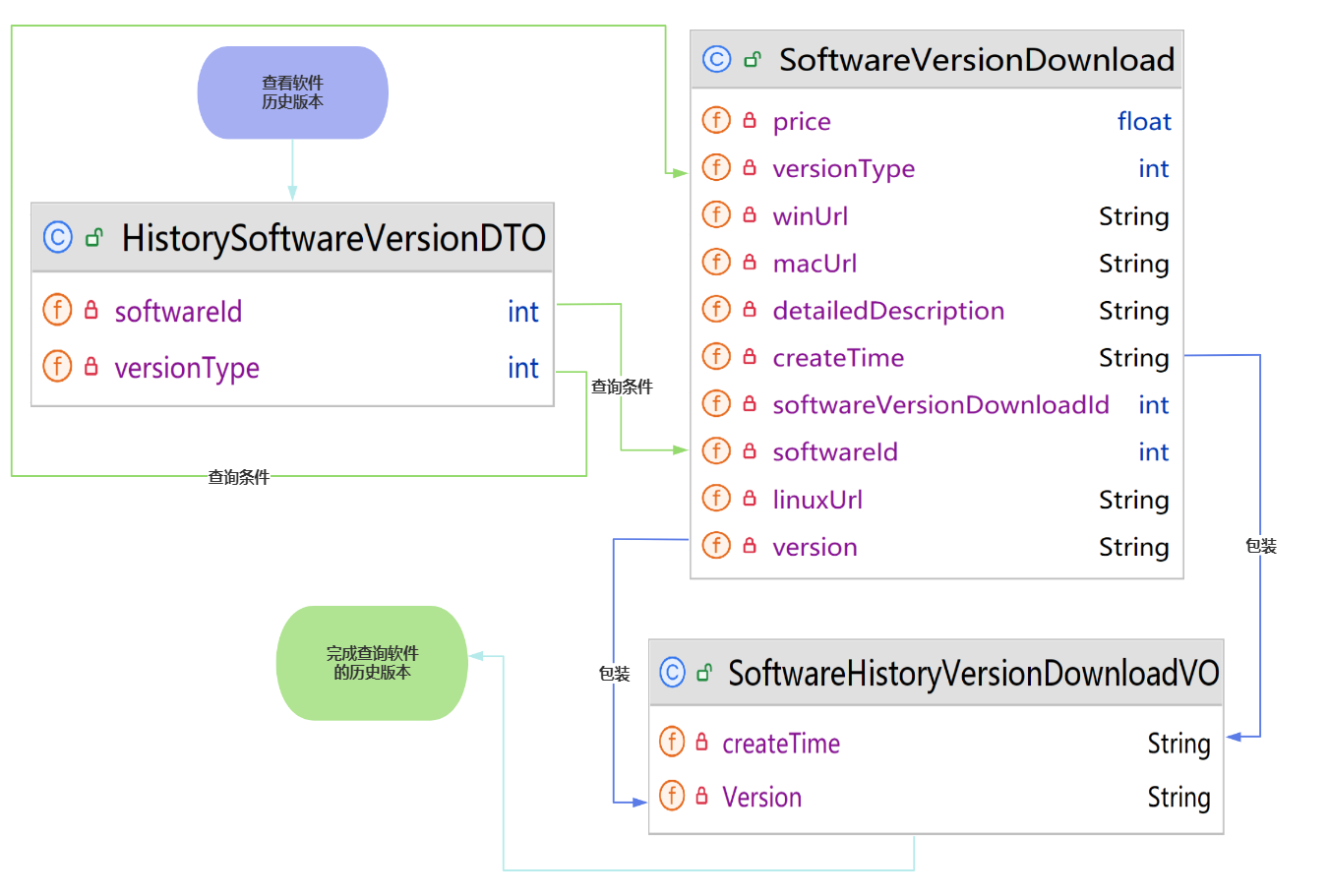
#### 3.2.4 /detailedSiftwareInfo 详细软件版本信息

同样发送软件ID到服务器，但是这次将会连表查询，对software 表和software\_version\_download表进行联查，同时将会依据时间判断最新版本，在软件详情页优先展示最新的版本。



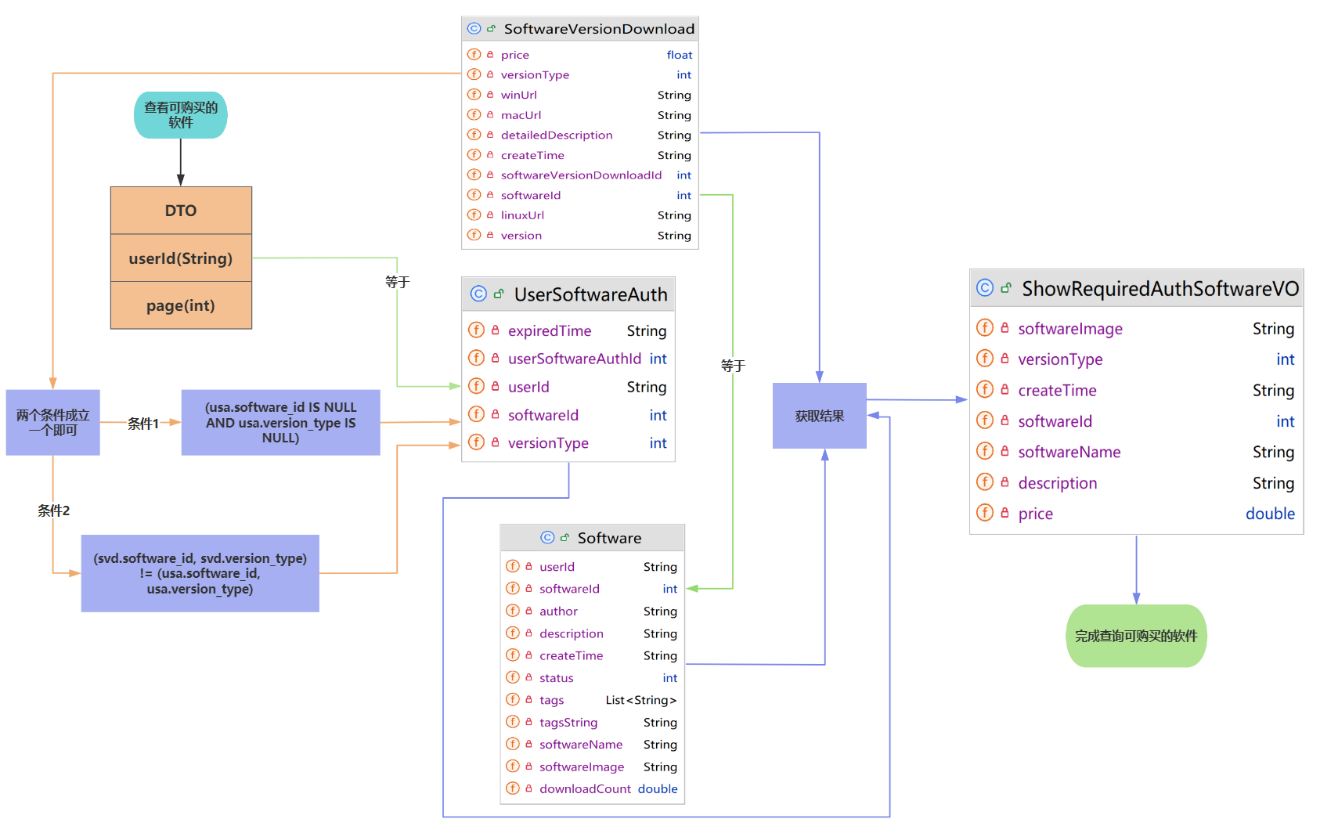
#### 3.2.5 /historySoftwareVersion 历史版本查看

在展示详细软件版本信息后，也支持用户查看历史版本。



#### 3.2.6 /showRequiredAuthSoftware 查看可购买的软件

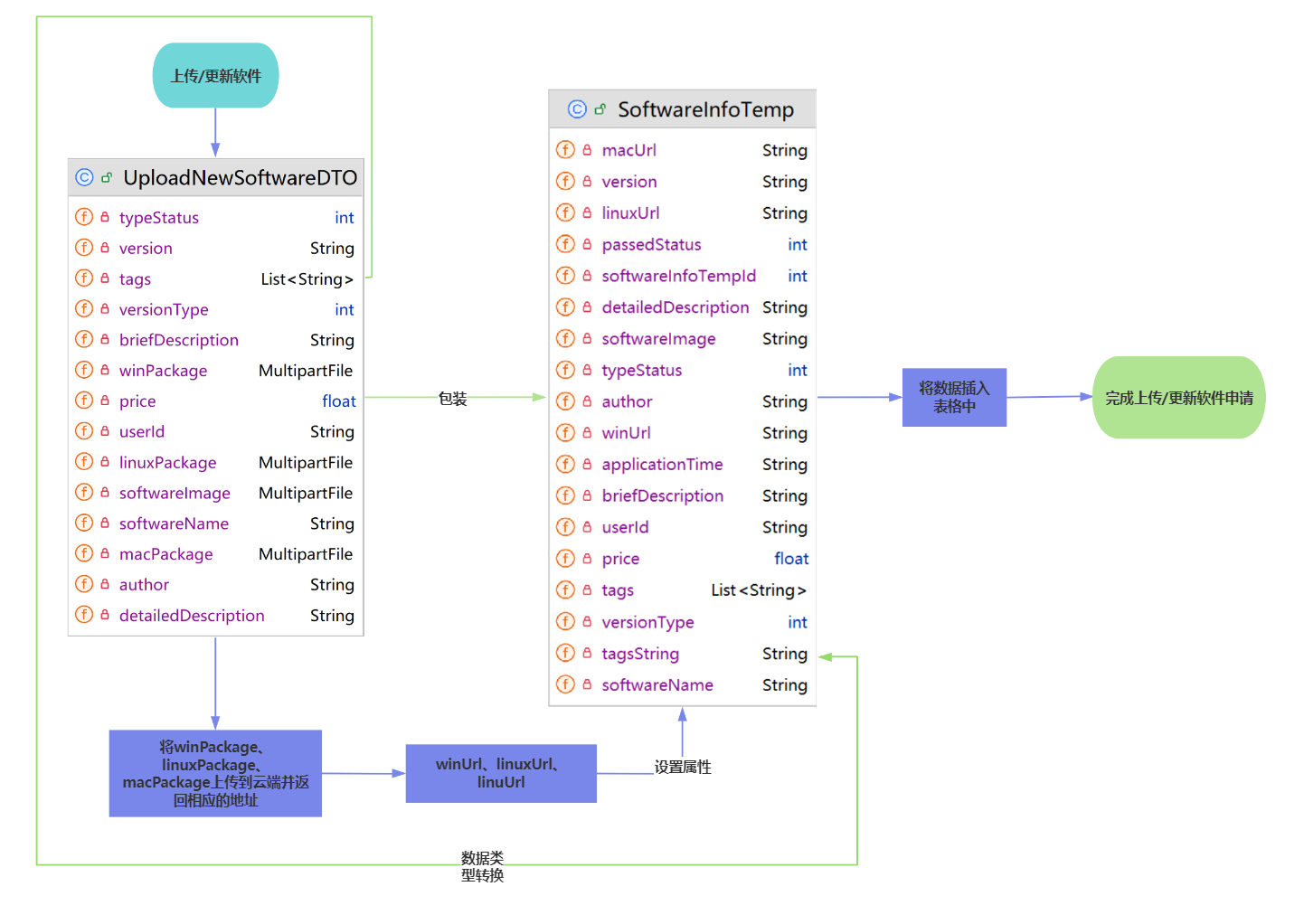
在购买页面，回显用户可以购买的软件，包括从未购买过的软件或者已购买过但是授权已经过期的软件。



#### 3.2.7 /upload 上传/更新软件

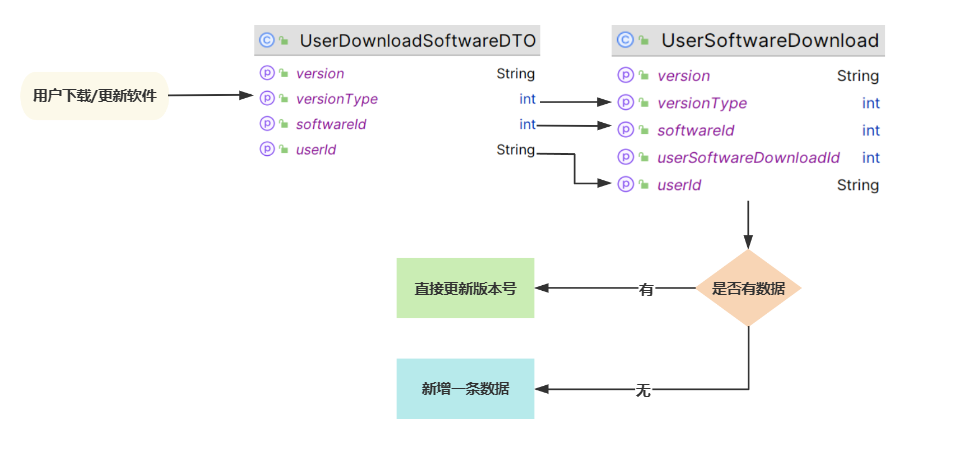
此接口接受所有上传新软件或者更新迭代版本或者添加专业版软件的请求。

用户将会选择上传新的软件或者新的版本，附带软件安装包以及详细信息。后台接受到这些附带文件的数据，首先会将这些文件传输到云端并获得下载url，然后将信息插入暂存表soft\_info\_temp中，等待管理员审核。

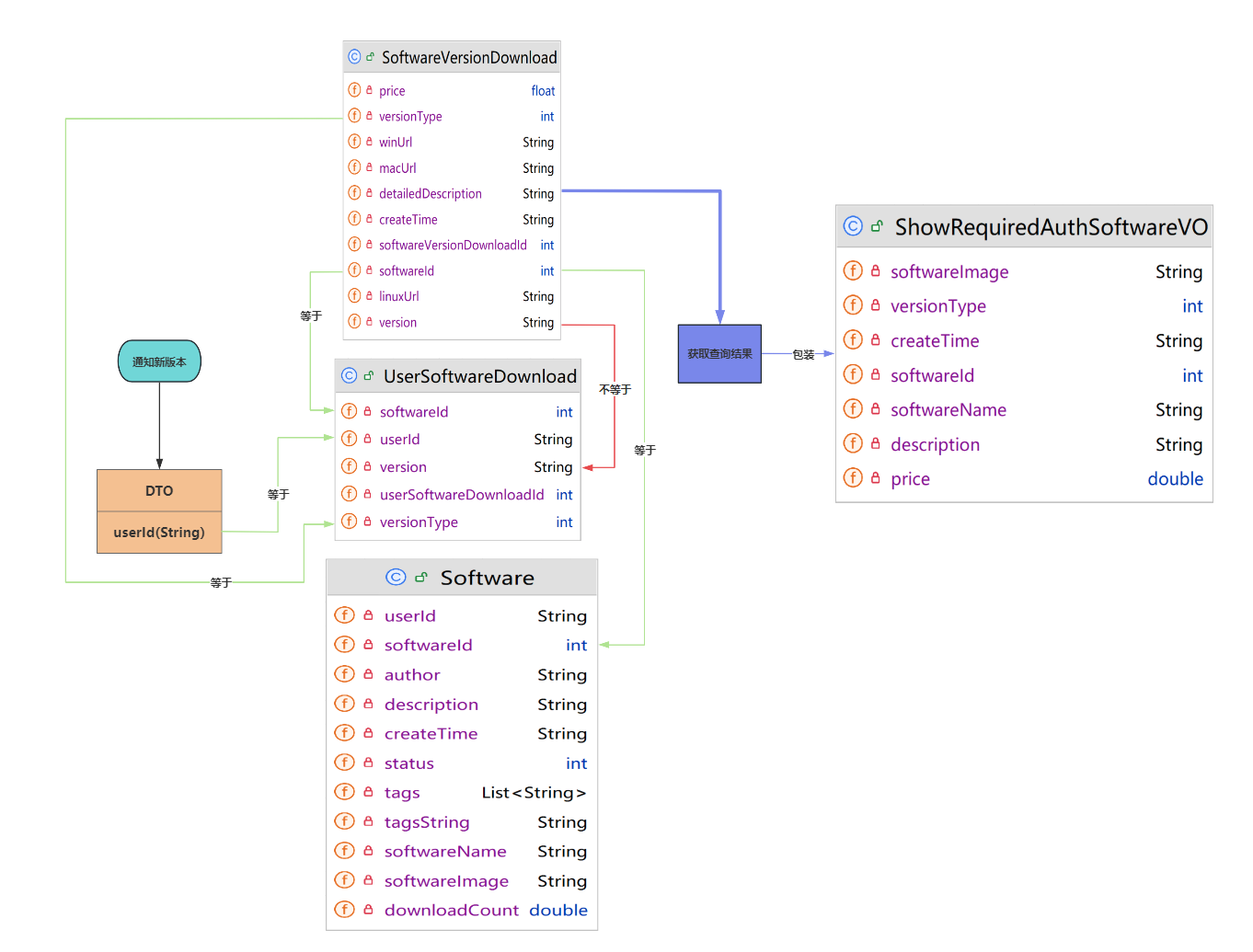


#### 3.2.8 /downloadSoftware 保存用户下载历史

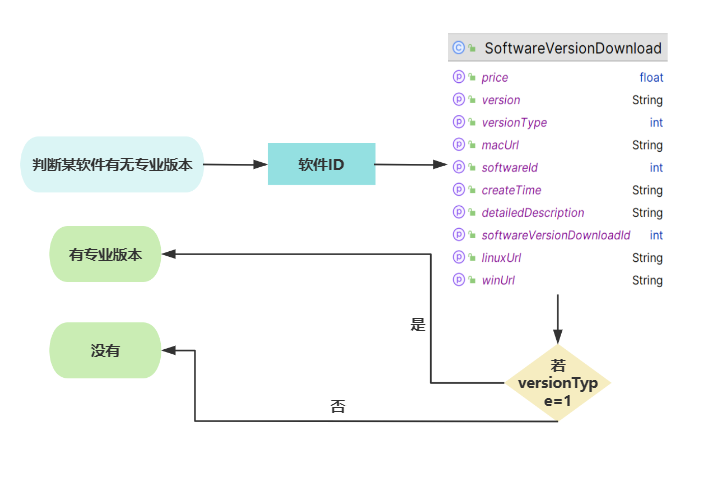
用户下载一个特定软件某个版本的某个版本号之后， 将会走这个接口记录用户下载的版本，用于后期的实时提醒用户更新版本。



#### 3.2.9 /checkLatestSoftware 查看用户是否有可更新的软件



#### 3.2.10 /judgeWhetherHavaPro 判断某个软件是否有专业版本

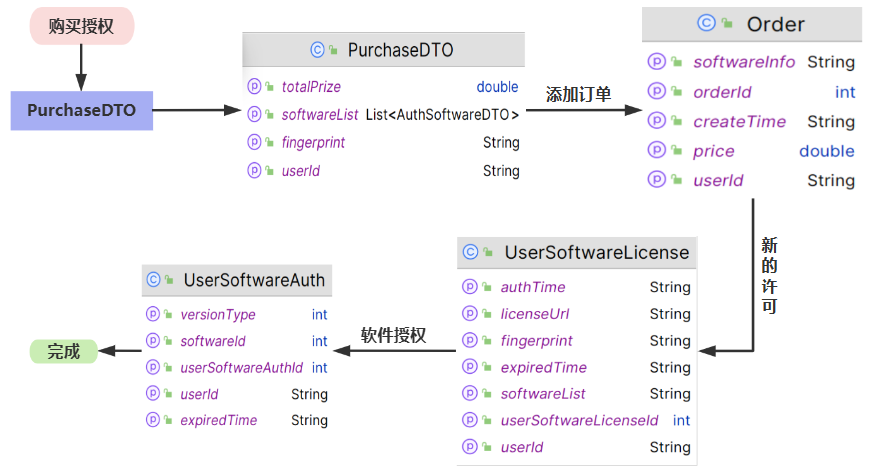
用户点击上传新版本时，服务器将会判断当前软件是否已有专业版本。

### 3.3 AuthorizationController /auth

#### 3.3.1 /purchaseAuth 用户购买一批软件授权许可

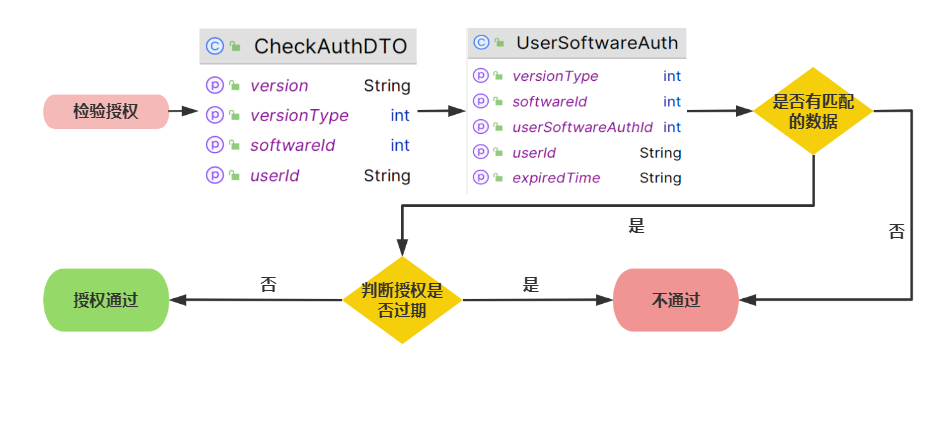
用户购买一批软件的授权之后，所有信息随着订单一起发送到后台。订单数据需要以下多步操作

* 将订单金额与购买的软件列表信息存入数据库，作为一条订单数据
* 解析附带的指纹信息以及购买的软件列表，加密后通过IO流写入授权文件，上传后获取授权文件在云端的url，将url附带着授权软件列表存入授权文件数据库
* 通过软件列表获取每一个软件的信息，附带用户id信息以及过期时间写入用户软件授权记录表



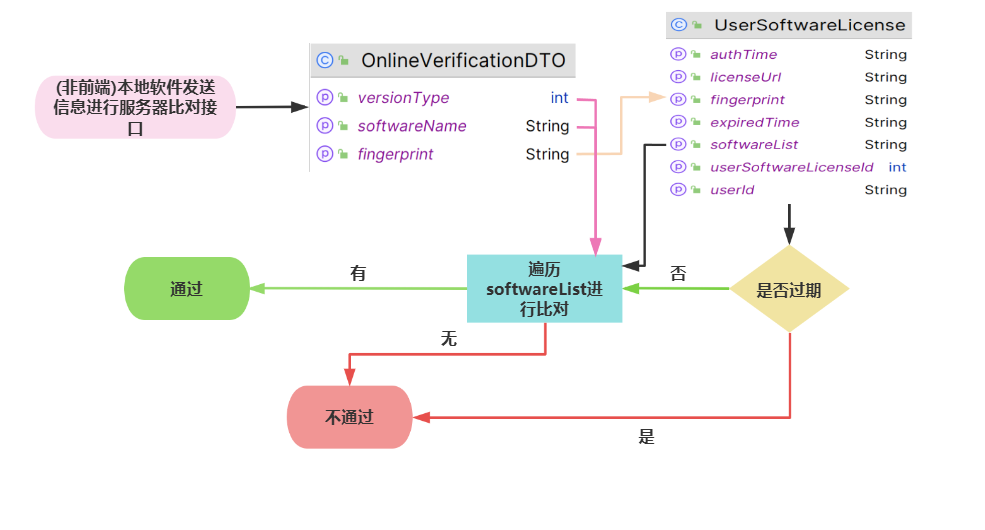
#### 3.3.2 /checkAuth 检验授权

用户点击某个软件的某个版本的某个版本号的下载按钮后，首先会携带用户ID，软件ID以及版本发送到服务器进行验证，查看用户是否购买过软件，如果查询到有数据则说明用户购买了软件，可以返回下载地址；反之则会返回空数据，前端进行提示。



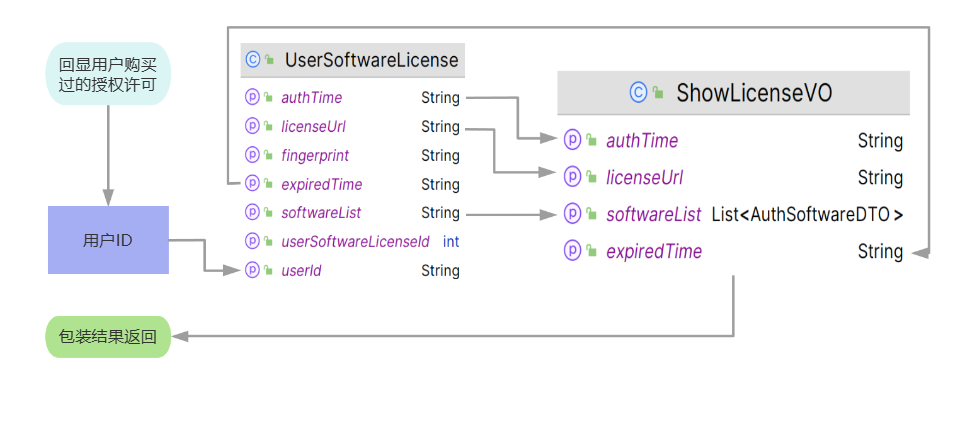
#### 3.3.3 /onlineVertification （非软件）本地软件发送信息到服务器进行比对

软件将会发送自己的软件信息以及指纹信息到服务器，服务器进行信息的检验和比对，返回结果给软件，软件通过响应结果判断用户当前设备是否为验证设备。



#### 3.3.4 /getLicense 获取授权历史记录和授权文件下载地址

前端发送用户ID到服务器进行查询，服务器查询数据库后返回用户曾经购买过的软件以及对应授权文件。



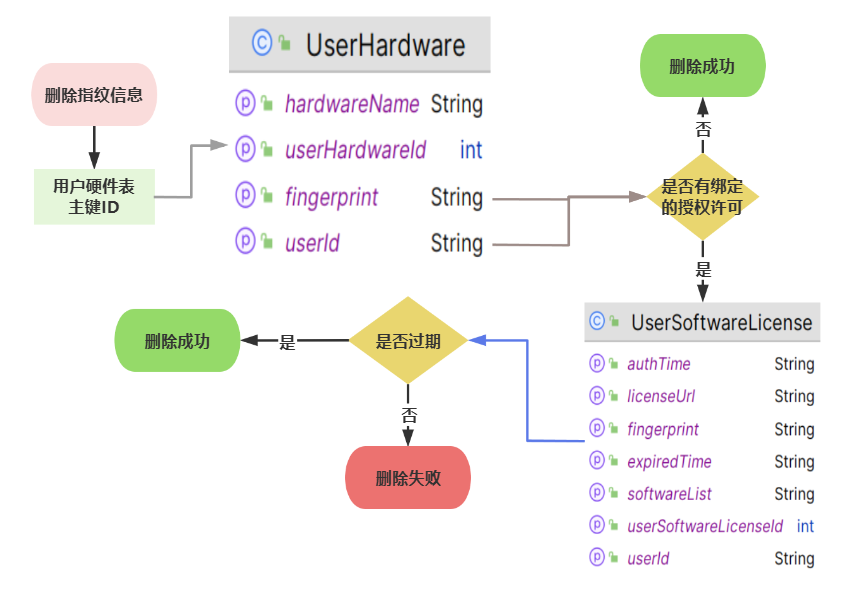
### 3.4 HardwareController/hardware

#### 3.4.1 insertFingerprint 用户上传硬件指纹信息

给用户提供我们准备好的查询本机硬件的程序，用户读取到之后可以上传他的指纹信息。

#### 3.4.2 /deleteFingerprint 用户删除硬件指纹信息

不允许用户随意删除指纹。当指纹有正在生效的许可记录与其绑定时将不允许删除；当指纹信息没有任何关联许可数据或者关联的许可数据全部过期了，才允许删除数据。



#### 3.4.3 /getFingerprint 查询用户指纹信息

提供用户Id，直接返回用户曾经上传过的硬件指纹信息。

### 3.5 UserController /user

#### 3.5.1 /homePageShowSoftware 个人页面展示个人开发的软件

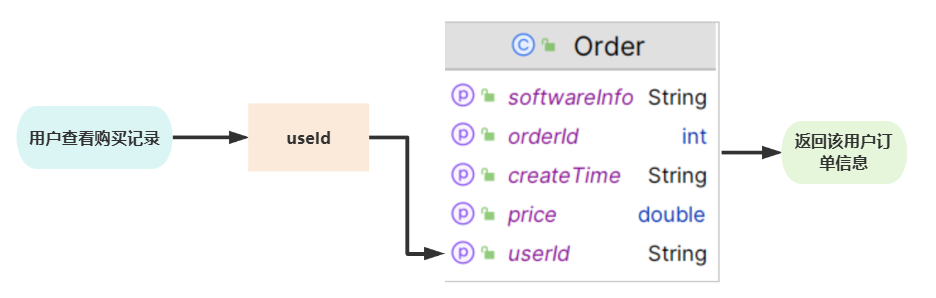
依据软件记录的上传用户ID返回给用户其自己上传的软件

#### 3.5.2 /applicationRecord 个人页面展示申请记录

依据软件记录的上传用户ID返回给用户其自己上传软件的申请记录

#### 3.5.3 /PagedQueryAvailableSoftware 个人页面查询可购买过（授权过但是可能过期了）的软件(通过过期时间判断是否过期)

依据用户Id进行连表查询，返回用户未购买或者购买过但是授权已经过期的软件\



#### 3.5.4 /updateUserInfo 用户修改头像，描述，昵称

用户修改个人数据，传输到后台进行数据库的更新。

#### 3.5.5 /showUserOrder 用户查看购买记录

查询购买过的软件，解析存储在user\_software\_auth的数据，返回前端对应数据。

### 3.6 AdminController /admin

#### 3.6.1 /updateSoftwareBasicInfo

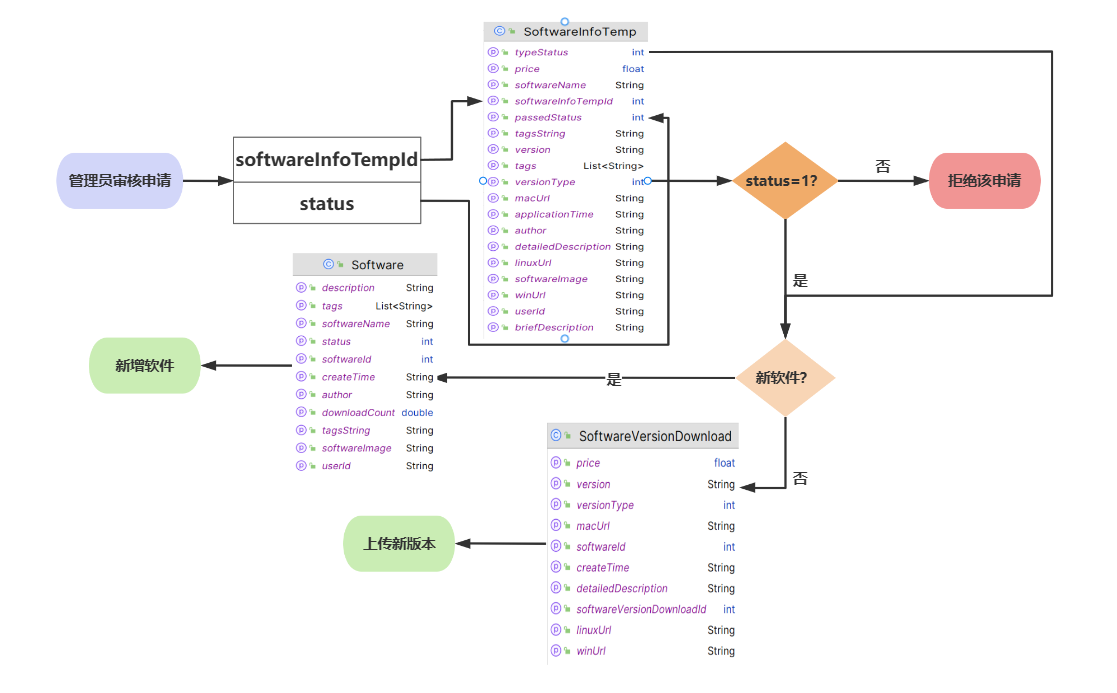
管理员根据软件ID修改软件基本信息

#### 3.6.2 /showVerificationHistory

管理员查看用户上传文件的所有申请，包括软件下载地址

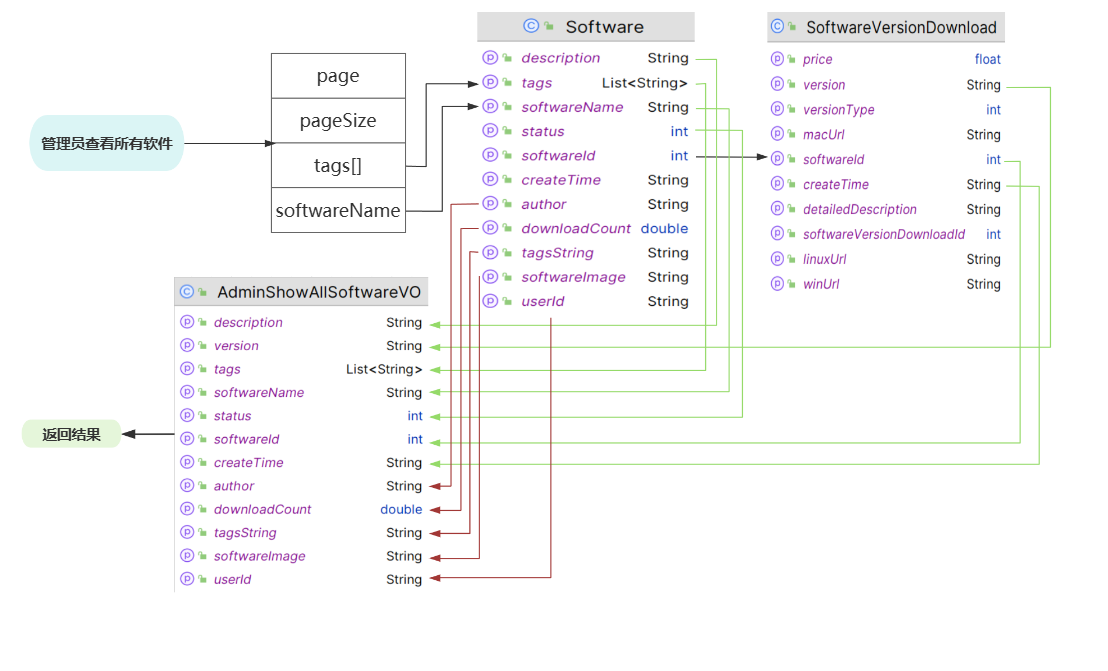
#### 3.6.3 /verifyApplication

管理员审核软件内容之后，可以选择是否过审。未过审的软件则会留存在暂存表，而过审的软件则会正式地插入软件表和下载地址表。



#### 3.6.4 /homePageShowAllSoftware

管理员可以根据标签和软件名模糊查询所有上架和下架的产品。



## 4. 项目亮点

### 前端

#### 样式管理

* **CSS Modules**：使用了 CSS Modules (styles 对象) 来管理样式，以确保样式只作用于当前组件，避免全局样式冲突。
* **动态样式**：在 `` 组件中使用内联样式来调整 Card 组件的外观。结合 CSS Modules，样式管理更加灵活和模块化。

#### 代码可读性和维护性

* **清晰的函数和变量命名**：代码中的函数和变量名明确描述了其功能，例如 handleFingerprintSelection 和 desireBuy。
* **组织良好的代码结构**：组件、函数和样式被合理组织在不同的文件中，使得代码易于理解和维护。

#### 查看软件详情页面

##### 软件版本管理功能：

历史版本记录：能够查看和下载软件的历史版本，满足用户回溯需求。  
版本更新提示：提供最新版本的更新信息，让用户及时了解新特性和改进。  
多版本选择下载：用户可以选择不同操作系统（Windows, Mac, Linux）下载相应版本的软件。

##### 用户友好交互设计：

详细的版本描述：每个版本都有详细的描述，用户可以了解每次更新的具体内容和改进。  
简洁的UI设计：整体布局简洁清晰，重要信息一目了然，提升用户体验。  
便捷的下载按钮：每个版本旁边都有下载按钮，用户可以方便快捷地下载需要的版本。

##### 高级版和基础版区分：

高级版功能展示：有高级版和基础版的区分，用户可以了解不同版本的特性和功能。  
购买引导：提供购买高级版的按钮，用户可以直接点击进行购买，增加了用户转化率。

##### 完整的用户体验流程：

高效的数据处理和展示：  
异步数据获取：使用异步函数获取历史版本数据，提高页面加载速度。  
状态管理：使用 React 的 useState 和 useEffect 进行状态管理和数据更新，保证页面的实时性和一致性。

#### 购买与订阅页面

**动态数据加载与分页**

* 通过 Pagination 组件实现了分页功能，优化了大数据集的展示，提升了用户体验。onChange 方法动态加载不同页的数据，保证页面性能。

**集成购物车功能**

* Subscription 组件整合了购物车功能，包括添加软件到购物车、查看购物车内容、计算总价等操作。通过 Drawer 组件显示购物车内容，实现了用户友好的交互。通过在底部区域提供购物车总价和操作按钮，用户可以方便地查看购物车内容并进行支付。handleBuyClick 方法导航到支付页面，增强了购物体验。MyShopping 组件展示了购物车的详细信息，并提供删除购物车项的功能。通过与 setShoppingOrder 的交互，实现了购物车项的动态更新。组件内部逻辑确保了用户只能添加尚未添加的软件到购物车中。在 IndividualSoftware 组件中，用户可以查看软件的详细信息，并且支持将软件添加到购物车。组件内的 handleClick 方法实现了购物车的更新逻辑，同时提供了软件详细信息的导航功能。

#### 总体配置上

**现代化用户界面**

* 使用了现代化的 CSS-in-JS 解决方案，如 Styled Components 和自定义内联样式，确保了界面的灵活性和一致性，同时提升了代码的可维护性。

**响应式布局**

* 使用 Ant Design 的 Row 和 Col 组件实现了响应式布局，确保界面在不同设备上的良好展示。结合自定义样式和 Ant Design 组件，实现了美观且一致的用户界面。

**错误处理与数据展示**

* 对数据请求进行了错误处理，确保了即使在请求失败的情况下，用户界面也能提供友好的反馈（例如 "没有数据可显示"）。提升了应用的稳定性和用户体验。

#### 实现懒加载

只有当用户滚动到视口内的内容时，相关数据或组件才会被加载。这种方式避免了在初始页面加载时加载所有内容，从而加快了页面的响应速度，减少了首屏渲染时间。

#### Context API状态管理

* **集中状态管理**  
  通过使用 Context API，实现了集中式的状态管理。AdminContext 提供了全局的管理员状态和更新函数，允许应用中的任何组件访问和更新管理员状态，而无需通过 props 进行层层传递。

### 后台

#### 模块化项目架构

* 分包规范，项目结构清晰，有利于项目开发效率和团队协作的流畅性

#### 高效数据传输

* 数据传输格式化，避免重复利用实体类，浪费资源或获取数据不完整，，不仅节省了资源，还确保了数据的完整性和一致性

#### 灵活的数据库设计

* 建表合理，耦合度低，为业务多元化、全方位拓展提供了坚实的基础

#### 跨平台兼容性

* 我们的软件下载地址支持多系统版本：win、mac、Linux，确保了广泛的用户覆盖

#### 强化数据安全

* 对授权文件中的信息进行AES+RSA加密解密，确保数据的安全性，为数据安全提供了双重保障

#### 便捷的离线授权

* 离线授权的时候，可以软件上进行信息比对并成功授权用户使用，实现快速而又安全授权，提升了用户体验

#### 安全验证机制

* 登录注册用了邮箱验证码登录

#### 严格的在线授权流程

* 在线授权时先检测硬件指纹信息，要进行账密登录，确保用户的使用权限

#### 智能更新服务

* 自动检测用户当下下载的软件是否为最新版本，帮助用户进行自动更新功能

#### 个性化硬件指纹服务

* 我们可以给用户提供自动获取其设备的硬件指纹信息的功能，帮助用户记录自己的硬件指纹信息，方便其购买授权

#### 授权文件绑定

* 用户购买的授权文件可以绑定要用户购买的软件 确保其权限完整有保障

#### 多样化软件版本

* 我们提供细分软件的不同类型版本适合不同专业人群、不同年龄段人群的需求

#### 用户自开发软件

* 用户可以上传属于自己开发的软件，并且让管理端进行审核上架

#### 严格的内容管理

* 管理端可以区别软件的上架下架，确保有些含有不良影响的软件不被购买

## 5. 项目难点

**后台**

### 1. 软件升级

* **识别已安装的软件版本**：需要开发系统能够识别并记录主机上已经安装的软件版本，并与服务器上的最新版本进行对比。
  + 通过判断用户下载的软件数据，与软件的最新版本进行比较，若版本不同则告知用户需要更新
* **自动升级和手动升级**：实现自动和手动升级功能，并确保升级过程的稳定性和可靠性。
  + 用户可以设置自动升级和手动升级。自动升级即提醒用户后自动为用户升级软件；手动升级提醒用户去有新的版本，用户自己点击升级按钮进行软件的升级处理

### 2. 软件授权

* **硬件指纹的获取和管理**：需要开发自动获取硬件指纹的功能，并管理多个设备的硬件指纹信息。\
  + 为用户提供自动获取硬件指纹的脚本，让用户自行输入，用户也可以输入其他设备的硬件指纹，并为其取别名，方便用户进行辨别，选择
* **授权许可的管理**：需要详细的授权许可信息管理，包括授权数量、有效时间、应用范围等。
  + 用户自行选择想要授权的软件同时提供硬件指纹信息，然后前端包装成数组的形式，发送给后台，后台将这些数据封装到数据库中，并为用户对这些软件进行授权
* **离线授权和在线授权**：分别实现离线和在线授权功能，并确保离线授权文件的安全和防篡改。
  + 通过软件来自动验证授权许可，软件判断用户是否在线，若在线，则进行在线验证，软件会将软件中的软件信息以及硬件指纹信息发送给服务器进行验证，验证通过则可直接使用软件。若判断用户不在线，则软件会读取软件目录下的包含加密过的软件信息和硬件信息的license许可文件，并进行解密以及验证操作，验证通过则可使用软件。

### 3. 安全性要求

* **授权文件加密**：离线授权文件需要通过严密的加密方式来保证安全，防止被恶意篡改。
  + 使用RSA加密方法对授权文件进行加密，加密后的数据再保存到离线授权文件当中

### 4. 产品模块

* **产品信息展示和下载**：设计合理的产品信息展示页面，并提供多版本、多类型的下载链接。
  + 对于多张表格进行相互联系查询，实现前端需要的数据包装，并保证数据传输的稳定性和可靠性
* **搜索和筛选功能**：开发高效的产品搜索和筛选功能。
  + 实现分页查询，以及标签的精准查询，和软件名的模糊查询，增加用户的体验感。

### 5. 管理端功能

* **软件信息上传和更新**：实现用户能够上传和管理软件信息，包括发布新版本。
  + 用户可以上传软件，对软件各项信息进行上传包括软件的三种打包方式的文件，三种打包方式是指的是，Windows系统的压缩方式和Linux系统的压缩方式还有Mac系统的压缩方式；同时用户上传了新软件后，可以对于已经上传的软件进行更新以及上传新版本，上传新版本需要提供新的金额。
* **软件管理**： 实现管理员能够对用户上传的软件信息进行审核，并修改现有软件的信息
  + 软件平台管理员可以对于用户上传的软件申请信息（包括，上传新软件、更新软件、上传软件新版本），管理员可以下载用户上传的软件，并对软件进行测试，通过后方可许可申请，否则拒绝申请；用户可以联系管理员，提供需要更改的软件信息，经过管理员的审核提供信息是否符合规范，再酌情更改。
* **前端**

1.组件的使用，在项目进行的过程中，由于缺乏对antd等网站提供的组件的充分认知，导致项目初期被样式修改等等难题严重拖慢了项目推进速率。后来了解过后才学会如何正确利用好组件。

2.较多个页面的搭建，此外还由于一开始没有做好规范的布局，导致后续需要花费不少时间去重新修改。

3.页面组件的精细化算个难题，我们对此经验还是尚有缺乏，页面拆分得不够精细，需要后续培养精细拆分的能力以便于后期维护。

4.与后台、设计的交接也是需要自习思考的难题，如果没有充分交流，很容易导致事倍功半。

5.在多个组件之间数据管理也较为复杂，想要花时间学习使用redux等技术去研究攻克。

6.错误处理：防御性编程（后台传输不符合预期结果的数据时的处理防止报错）

## 6. 存在问题

**前端**

1. 组件运用还是不够熟练，出现运用组件后无法解决css样式修改及覆盖问题。
2. 响应式出现问题（已解决）。
3. react代码模块应该再分细一点，提高开发效率，方便代码维护。
4. CSS类名污染（已解决）。
5. react各个钩子函数运用还需加强。
6. 目前代码规范性还是有所欠缺，需要适当提高这方面的意识。
7. 代码的书写还是要多加注释，缺乏注释会极大程度减低代码的易懂程度，不利于日后对项目代买的维护和复盘。无独有偶，毫无注释的代码会拖累小组之间的团队合作，不仅别人看不懂你的代码，甚至可能自己都看不懂自己的代码。

##### 合作问题

1. 项目合法开发的重中之重不仅仅只限于开发，还要求着重于合作。除了小组内成员的有序分工外，还要加强与设计、后台成员的沟通交流。通过与设计成员的沟通，可以加快页面的构筑搭建，此外还能保证页面符合先前的设想。通过与后台成员的交流，则能保证后台能充分准确地提供前端所需求的数据。团队的协调合作，可谓是任重而道远（目前仍需努力）。 以下则是目前合作所欠缺的：
2. 与后台和设计之间沟通出现问题，出现已经写好页面后，却发现后台没有对应的接口或者后台的接口不能与设 计给的页面匹配，导致有时候页面写好后要重写。
3. 应该让设计先给出页面的布局大纲再进行详细设计，我们前端与后台应该先通过设计的页面进行讨论，确定没 问题后再进行下一步。

4.应该与后台设计多开会讨论项目相关问题

5.在设计没给页面之前，前端可以先把一些需要用的函数写好，不能只是一味的等待设计出稿

## 7. 心得体会

**后台**

第一次参与合作开发项目，没有什么经验，大家一起在这个过程中不停地一步一步摸索，虽然可能我们并不是那么顺利在这个过程中，但是我肯定相信，经过这十分有影响的一次考核，每一个人都受益匪浅，不仅仅是知识上的提升，也是平时交流，个人的团队协作能力、沟通能力，思维能力，等一些都得到了实质性的提升。

在这次考核中，我深刻体会到了团队协作的力量。在这个过程中，我学会了如何与不同组的同学沟通和协调，不论是自己组内的小伙伴，或者是其他组的小伙怕，理解了每个人的角色和责任对于项目成功的重要性。而且我也深刻意识到，有效的沟通不仅能减少误解，还能激发团队的创造力和效率。记得我们开正式写项目业务前，我们组的成员都是一起开会交流，沟通好之间的每个业务的逻辑，每个分工的协作，大家尽可能同步进行，从而确保项目的正常发展。

此外，我也认识到了自己在项目的协作开发和自己的技术实现方面的不足，这激励我在未来的学习中更加努力地学习和提升。不管是自己学习也好，有问题时请教他人也罢，这次经历都会促使我更好地成长。通过这次经历，我更加坚信，只有不断学习和适应，才能不能提升自己，成长自己。

总之，这次合作开发项目不仅让我获得了宝贵的实践经验，也让我对团队合作有了更深的理解和认识。我相信，我们每一个人在这次项目中都获得了不可多得，无价的财富，我们以后也会继续学习，提升。

**前端**

这次项目经历，使得我感触颇深，第一次从前后端都需要自己一个人写，到前后端分开写，且业务也不需要自己一个人完成，是小组一起共同完成。在项目开发效率上，后台接口文档一旦定下，业务可以很快的实现，完成项目业务逻辑的速度远比一个人要快得多，比起自己一个人写前端加上业务逻辑相比，的确快了不少；但是同时也有很多问题，在于前端与后台交互之间有很大的冲突，在对接口的过程中，常常出现不同的想法，同时后台定制文档后，会有很多考虑不完全的地方，需要前后端人员配合设计组共同努力，方可减少在页面设计，接口文档的定夺方面的冲突。

这次项目完成后，我感受到了合作的重要性，一个人单干对于团队协作来说非常致命。这也是我第一次切实的体会到，感悟到团队协作对于完成一件事有多么重要。虽然是第一次做项目，肯定会有意见的各不相同，前后端的争执，前端与设计的争执，这都需要各自磨合，去锻炼团队协作能力。我们要在一次次的项目合作当中，不断进步，不断成长，变成一个具备团队协作精神，具有团队协作能力的合格的QG工作室成员。