Week 7：接口逻辑编写

关于rest：https://www.runoob.com/w3cnote/restful-architecture.html

实际上，restful是一个接口风格规范，就比如说我们在之前编写的串流接口：

```
<GET>       /stream/<track_id>
```

我们最终的音乐播放，需要借助这个接口完成，即vuetify-audio-player需要播放的音频url来自于这个接口。

那么这个前端页面需要实现与后端哪些接口的对接？

```
<GET>       /album                  →获取数据库中的所有album列表
<GET>       /album/<pid>            →获取单个album的信息
<PUT>       /album/<pid>                    →将专辑添加到用户收藏中
<DELETE>    /album                  →删除用户收藏的专辑

<GET>       /playlist               →获取数据库的所有playlist列表
<GET>       /playlist/<pid>         →获取单个playlist的信息
<POST>      /playlist               →创建播放列表
<PUT>       /playlist/<pid>         →将播放列表添加到用户收藏中
<DELETE>    /playlist/<pid>         →删除用户收藏的播放列表

<GET>       /track                  →获取数据库的所有歌曲列表
<GET>       /track/<track_id>       →获取单条歌曲信息
<PUT>       /track/<track_id>       →将歌曲添加到用户收藏中
<DELETE>    /track/<track_id>       →删除用户收藏的歌曲

// User用户信息
<GET>       /user/login             →登陆，获取用户的jwt token（这一部分先跳过）
<POST>      /user/login             →注册
<DELETE>    /user/login             →登出
<GET>       /user/<uid>             →获取用户统计信息

// image图片信息
<GET>       /image/<album_id>       →获取专辑封面图片，返回文件
```

接下来参考https://github.com/chenshenhai/koa2-note/tree/master/demo设计文件框架，逐步实现上述接口。

需要强调一点，在这个项目中，playlist和album采用的是同样的数据库结构，但对用户以不同的形式呈现，所以在接口和数据库关系之间的映射稍微比较复杂。

任务1

我们暂时跳过login页面的校验流程，即登录页不输入任何信息按登陆都能够跳到主页。

然后尝试调通Explore页的下述两个接口，在Explore能够看到推荐Album和推荐Playlists。

`<GET> /album`

`<GET> /playlist`

`<GET> /track`

后端参考的返回格式

成功

获取列表

```
{
    "data": [],
    "total": 0 // 条目数量
}
```

获取单个条目

```
{
    "data":{

    }
}
```

失败

```
{
    "err": 201,
    "msg": "Resource doesn't exist"
}
```

在获取列表时，data中包含的数据视显示内容需求而定。最基本的，需要有id、封面图、标题和description。

id是进一步索引请求信息的必须项，剩下三个是在首页显示所需的必要元素。

任务2

REST API的一种请求payload形式如图所示：

JSON Server

This Data Provider fits REST APIs powered by JSON Server , such as JSONPlaceholder .

Method	API calls
`getList`	GET `/posts?_sort=title&_order=ASC&_start=0&_end=24&title=bar&_embed=comments&_expand=user`
`getOne`	GET `/posts/123`
`getMany`	GET `/posts?id=123&id=456&id=789`
`create`	POST `/posts/123`
`update`	PUT `/posts/123`
`updateMany`	Multiple calls to PUT `/posts/{id}`
`delete`	DELETE `/posts/123`
`deleteMany`	Multiple calls to DELETE `/posts/{id}`

在后续的API开发中，必然会遇到筛选和排序问题，我们将会按照类似JSONPlaceholder这种数据请求格式去完成前后端query的资源对接。

现在，你需要实现下面几个接口：

`<GET> /album/<pid>`

`<GET> /playlist/<pid>`

`<GET> /track/<track_id>`

后端参考的返回格式

成功

```
{
    "data":{
        // 包含数据库条目中所有信息
    }
}
```

失败

```
{
    "err": 201,
    "msg": "Resource doesn't exist"
}
```

第一版上线

（尽量在3天之内完成，根据行为顺序理出功能模块的KPI，逐个实现）

在你完成上面的6条接口功能后，实际上一个免登录的开放音乐平台已经实现了。

尝试将前后端调通，接下来将给到一个粗略的行为顺序：

1. 启动数据库，启动后端，后端进行曲库内容的初始化和刷新
2. 启动前端项目，跳转到/login，在登陆页面不输入任何内容直接进入/explore
3. 在/explore，前端首先会请求如下内容：

   `<GET> /album`

   `<GET> /playlist`

   并且将获取到的数据绑定到视图中。其中，图片通过请求`<GET> /image/<album_id>`获得。后端的处理逻辑是通过album_id去向数据库查询playlist.index中的条目，将其中image中的path读取，并以文件返回给前端。如果没有对应的图片文件，则返回null，前端使用预置的占位图片替代。
4. 用户通过点击playlist或者album使得url跳转到/album/<pid>或者/playlist/<pid>页面，前端通过请求同名接口获取信息，绑定并渲染到界面上。同时，对列表中的每一首track，调用`<GET> /track/<track_id>`获取详细信息用于展示或存储到后台
5. 播放音乐时，将`<GET> /stream/<track_id>`送入vuetify-audio-player播放串流