Week 8：加入用户能力

介绍基于JWT的无状态认证：

https://www.jianshu.com/p/576dbf44b2ae

那么接下来，我们就要在这个平台上添加用户登陆以及围绕用户相关的功能。

> 参考：https://segmentfault.com/a/1190000019338195
>
> https://www.cnblogs.com/pomelott/p/11026626.html
>
> 更多的参考案例请自行搜索KOA + JWT

在实现了用户信息的登陆后，我们才能够进一步构建个人库的相关功能实现。

实际上于接口而言，个人库与全平台资源的区别只在于所属用户的不同罢了。

从现在开始，所有的接口操作都需要在Header中携带jwt token，否则请求直接回报unauthorized错误（实装拦截器）。

- 在Library主页，获取个人曲库数据，需要向后端发送`<GET> /playlist?_user=me&_type=library`，后端收到payload后，使用Header中的token解析出用户uid，然后向数据库获取library.u_<uid>的数据，选出其中`type: "album"`和`type: "track"`的条目。

  单独建立用户库集合，是因为用户库是一个混合结构，其中歌曲音轨和专辑混合在一起，有一个加入时间的顺序用于管理。

  该接口默认以added_date倒序返回，当然也可以通过传入`_sort=name&_ascending=true` `_sort=artist&_ascending=false`等按其它顺序获取列表
- playlist与library中的albums和tracks是分开的，单独一行在界面呈现。请求方式与上面的类似：

  `<GET> /playlist?_user=me&_type=playlist`
- 创建播放列表的时候只需要post name和description这两个基本信息，pid由后端随机生成16位字母+数字。

  在数据库层面需要同时维护index集合和p_<pid>集合，playlist只允许存在track。

  创建的同时还需要将这个pid加入用户的library.u_<uid>中。
- `<PUT> /playlist/<pid>`仅需携带一个数据：

  ```
  { "item": "track_id" }
  ```

  后端需要根据jwt校验pid的authtor，非author不具备更改权限，<DELETE>请求同理。
- <DELETE>请求不删除playlists数据库的内容，只删除library.u_<uid>的条目，同时将public设为false，以在平台上隐藏
- 补充一点，平台Explore页面只展示public=true的playlists
- 用户是手动更改播放状态，比如手动按下播放键、手动播放音轨时，将使用node-machine-id这个依赖包获取的机器id，并post向服务器：

```
<POST>  /user/<uid>
{
    "playing": "machine_id"
}
```

- 前端在播放音乐时，每30秒向后端获取一次machine_id：

  ```
  <GET>   /user/<uid>?_item=playing
  ```

  与本机machine_id比较，如果发现被更改，则自动暂停音乐，并且弹出dialog，提示用户同一时间只能有一个设备在播放。

根据上文设计功能点和API完整交互列表，实现整个带用户能力的音乐平台。