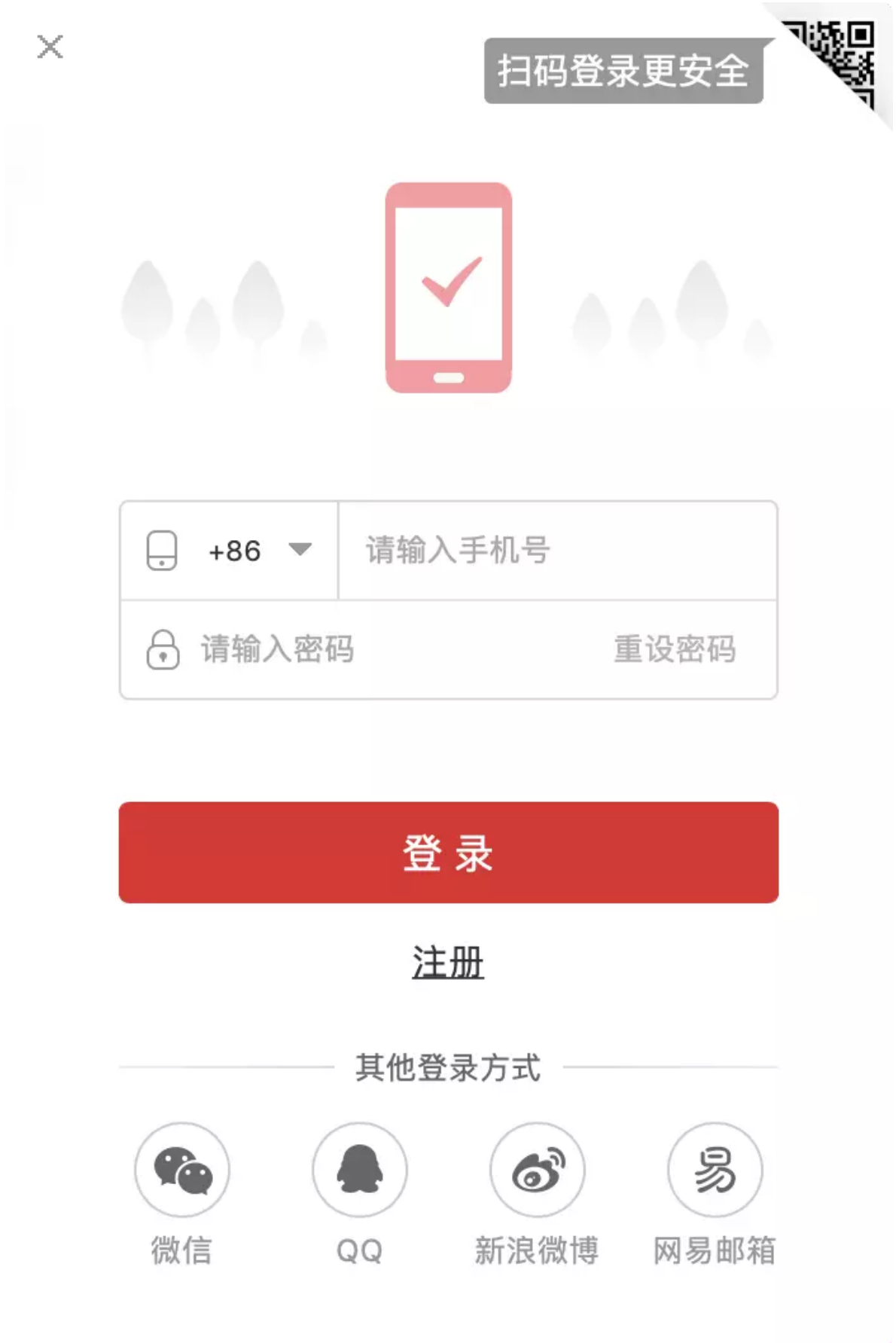
多账户的统一登录

名称解释

这里的多账户区别于系统级别的，我们讲的多账户系统是指，在我们互联网应用当中，我们的应用会使用多个第三方账号进行登录，必须现在常用的APP（网易云音乐）登录方式包含：网易、微信、QQ

内容

通过这一篇文章， 可以学到：多用户下面的技术方案细节，以及相应的表设计，流程设计。 不可以：与其他文章一样，我这里不会有具体代码实现细节，方案做的对，代码咋写都不会太烂。 网易登录.png



架构演进

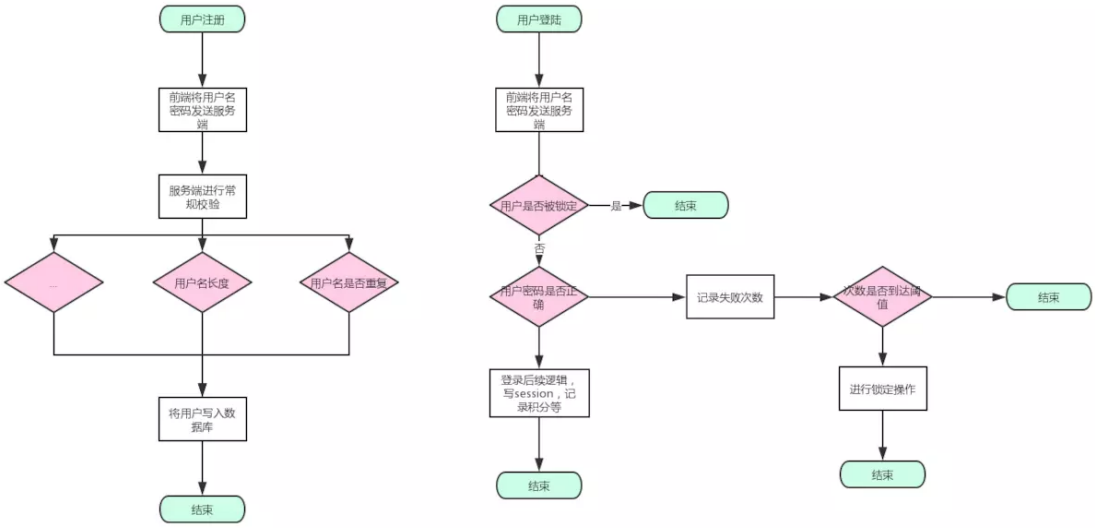
创业初期

归结为创业初期是因为这个时候用户量比较少，甚至还没有接入上面所说的其他第三方的账户系统，只是自建的体系就可以满足，自建体系的话，目前常用的有

用户名密码注册登陆

这种方式在很多初期网站建设会使用，先注册，再进行登录，在老一点的cms中都能找到这个影子。

流程图：

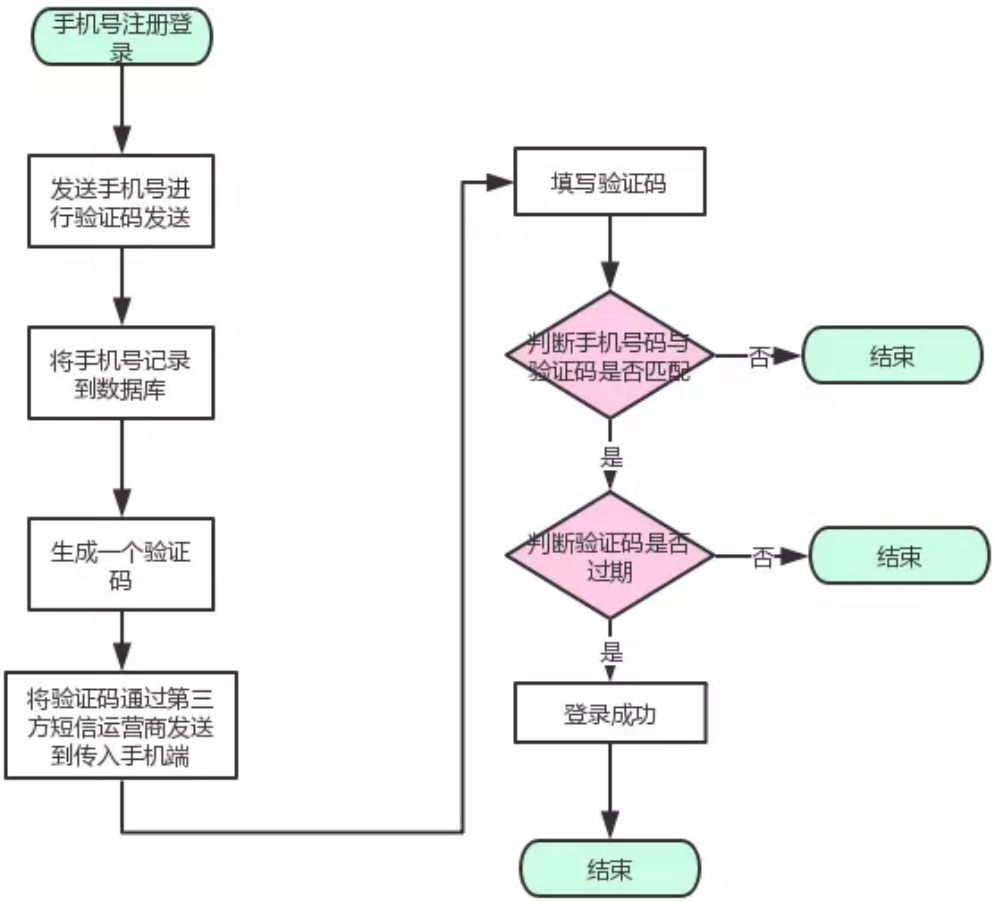


流程说明：

> 1. 前端将用户名、密码发送到服务器，服务器进行常规的判断，判断用户名、密码长度是否满足，用户名是否重复等条件，条件不通过直接返回对应错误码给到前端，这里密码字段，为了防止传输过程中被截胡，建议加密再上传，我们的传输密码默认都是会进行一个md5加密，然后记录到数据库再进行一层加密，就算是脱库也没事，密码不要明文存储。
> 2. 校验通过后，就将用户名密码写入数据库，并进行后面积分发放等操作，这里不展开。
> 3. 现在进行登录，前端将用户名，密码发送给到服务端，服务端首先会校验登录次数是否超过设置的阈值，如果超过只能继续等待被关小黑屋。
> 4. 如果未超过继续登录逻辑，判断用户名、密码是否正确，不正确密码则进行阈值的判断，如果超过则关小黑屋，记住小黑屋必须设置过期时间，要不然就会永久关上了，这个可以用redis的过期来做。
> 5. 登录成功后进行后续的一切后置逻辑，比如加积分。。。等操作。

手机号注册登陆

流程图：



流程说明：

> 1. 首先输入手机号，然后发送到服务端，服务端将手机号记录在我们数据库中，然后生成随机验证码，并将手机号和验证码绑定到一个redis里面，然后记录过期时间，这个过期时间一般是10分钟左右，这就是我们一般手机验证码的有效期。
> 2. 手机接收到手机短信后，那么就在界面填写验证码发送服务端，服务端收到验证码后就会在redis里面查询到这个手机号对应的验证码，失败就返回错误码。
> 3. 成功后就进行登录操作。

这里看起来没有明确的注册登录操作，其实在发送手机号码就可以认为是一个常规的注册，然后后面的验证码输入就是一个登陆操作，

问： 那我要密码咋办?

答： 在后续产品里面增加一个手机号码密码补录的功能即可，这也是现在很常规的手法，但是现在移动互联网大爆炸时代，密码已经显得不是那么重要了，反正我从来记不住密码，如果手机号码能操作的app，绝对不用密码来操作。

数据库设计

表结构

自增id	用户名	密码	手机号	错误次数
1	user1	7fef6171469e80d32c0559f88b377245	13456789012	0

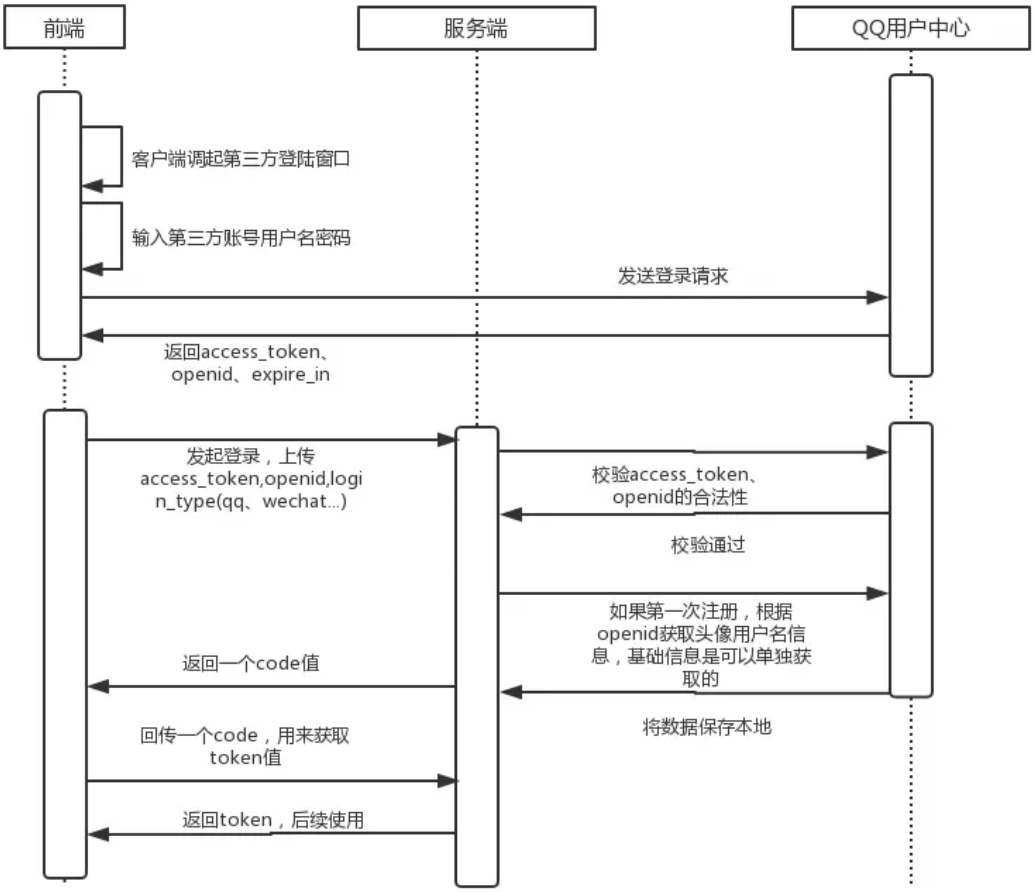
自增id	用户名	密码	手机号	错误次数
2	user2	7fef6171469e80d32c0559f88b377245	123456789013	0

说明

> 1. 这里只是单纯说明需要用到的数据，没有扩展具体场景,这个表结构能够满足上面两个方案的设计。

引入第三方账户方案

这里是以QQ-SDK的登录逻辑， 我们先来一波时序图



说明:

> 1. 客户端自己调起登录的界面，进行输入用户名、密码，这里的是第三方的用户名，密码，登录成功后，会返回access_token openid expire_in,这过程会使用到oauth2.0，不过在sdk里面进行内置回调获取了，后面我们会说明我们自身实现的oauth2.0
> 2. 客户端拿到access_token、openid、login_type（qq、wechat...）请求应用服务器，应用服务器拿到这些数据后就会根据对应的login_type去对应的用户中心进行access_token和openid进行校验。校验不通过则返回对应错误码
> 3. 校验通过后就会判断本地是否有这个login_type和openid是否存在，不存在则进行获取远程的用户名、头像等基础信息来作为本地基础数据，并且返回code值
> 4. 如果已经存在，那就是进行登录操作，返回code值。

> 5. 客户端拿到code值后进行token值的换取，这个完全遵照oauth2.0的协议来走的，后续每次请求必须带上token，token值在服务端的时间比较久，因为我们想要做的是那种永不下线的操作，所以每次请求我们都将token过期时间进行累加。

数据库设计

表结构

对于评论处 @讲不出再见1486617502000 的建议，我这里做一下数据库的整理 用户基础表（users）

字段	备注
user_id	用户id
token	用户登陆的token
expire_in	token过期时间
try_times	登录失败次数

用户验证关联表（user_auth_rel）

字段	备注
id	自增id
user_id	用户id
auth_id	验证表id
auth_type	验证类型(local、third)

本地用户表（user_local_auth）

字段	备注
auth_id	认证id，自增id
user_name	用户唯一标识
password	用户密码
mobile	用户手机

第三方用户表（user_third_auth）

字段	备注
auth_id	用户id
openid	第三方用户唯一标识
login_type	第三方平台标识(qq、wechat...)
access_token	第三方获取的access_token,校验使用

说明

> 1. users表只是单纯针对我们业务侧的登录，主要是做自身业务的oauth2.0业务，
> 2. user_local_auth是做自己用户名、密码登录，手机号码登录信息记录，
> 3. user_third_auth是我们第三方用户体系的数据记录，
> 4. user_auth_rel是用来关联我们users表与user_local_auth、user_third_auth。
> 5. 整个设计理念就是将自建用户与第三方在存储上区分，这在架构演进上也是合乎情理的，开始用户体系大多自建，而后才是对外接入。

总结

1. 总的来讲，第三方用户的接入技术上来讲是比较简单的，这里设计多一个user_thirds是可以支持足够多的第三方接入，当然一般我们也就两三个登录就好，太多登录方不仅自身维护成本，界面摆盘也不好看不是。
2. 希望大家能够通过以上学习，能够对于我们多账户登录有一个比较好的认知，这里设计方案不包含分表分库、没有服务化，就是简单直接的设计，当然用户量和需要的不一样，在这个基础上还要加很多东西，谢谢大家阅读！！！