支付中心到了后面，是我们的重头戏，业务迭代，支付中心我们会迭代的非常复杂和完善，复杂100倍，1000倍的支付架构出来。媲美任何一个企业级，商用级的支付架构。

# 1、数据架构

## 1.1 数据库ER模型

## 1.2 数据库逻辑模型

## 1.3 数据库物理模型

# 2、接口

到了实际开发的时候，给大家讲讲怎么对接第三方支付之后，我们再定接口

# 3、开发架构

## 3.1 实现类图

在后面我们实际做到跟第三方支付对接的时候，我们需要实现代理模式，这是代理模式一个经典的使用场景，就是我们在本地可以实现一套支付接口service，然后实际的service实现类，就是每个支付方式一个实际的service实现类

然后每个service实现类，实际上就是一个代理类，代理了对远程的第三方支付接口的访问

## 3.2 包设计

# 4、运行流程

## 4.1 第三方支付对接

1、获取第三方支付的二维码，需要明确指定二维码的有效时间是30分钟

2、用户会去扫码进行支付，但是也有可能用户会放弃支付

3、支付中心每次获取二维码之后，就需要记录一条支付流水，刚开始的状态是待付款

4、针对支付宝扫码支付的情况，支付中心需要有一个后台的线程，每隔10秒钟，就查询所有的待付款状态的支付流水，去第三方支付平台去轮询查询用户是否完成了支付

5、针对微信扫码支付的情况，支付中心需要提供一个回调接口，一旦用户扫码完成了微信支付之后，微信会来回调我们的接口，告诉我们微信支付的结果

6、无论是支付宝还是微信支付，只要获取到支付成功的状态之后，就会更新本地的支付交易流水的状态是支付成功；如果支付交易的状态是失败，也要更新交易状态是失败

7、如果后台线程发现支付交易流水已经创建了超过30分钟，而且用户还没有完成支付，那么此时就会将支付交易流水的状态设置为交易关闭

### 4.1.1 支付宝对接

### 4.1.2 微信支付对接

## 4.2 第三方支付退款

### 4.2.1 支付宝退款

### 4.2.2 微信支付退款

## 4.3 支付交易流水管理

### 4.3.1 保存支付交易流水

每次完成一次支付之后，就保存一条交易流水

### 4.3.2 查询支付交易流水

支持一些筛选条件，分页查询支付交易流水

# 5、测试用例设计

# 6、日志设计