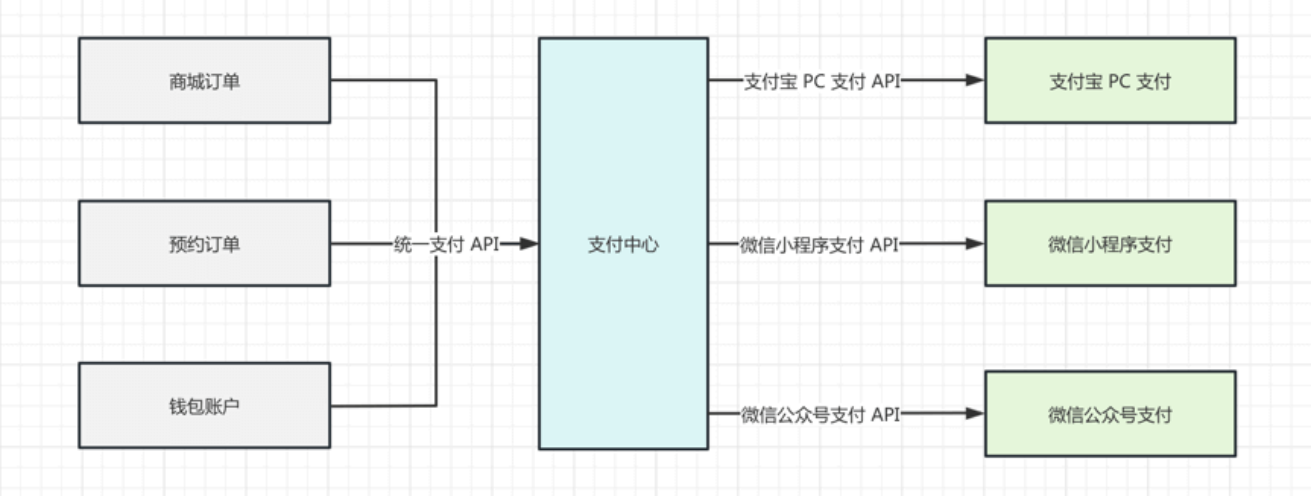
芋道源码 2023-07-07

功能开启

> **支付中心最新代码**
>
> - 后端：已经在 master 分支
> - 前端：暂时只能使用 Vue2 版本，也是在 该分支，预计 8 月份会同步到 Vue3 版本

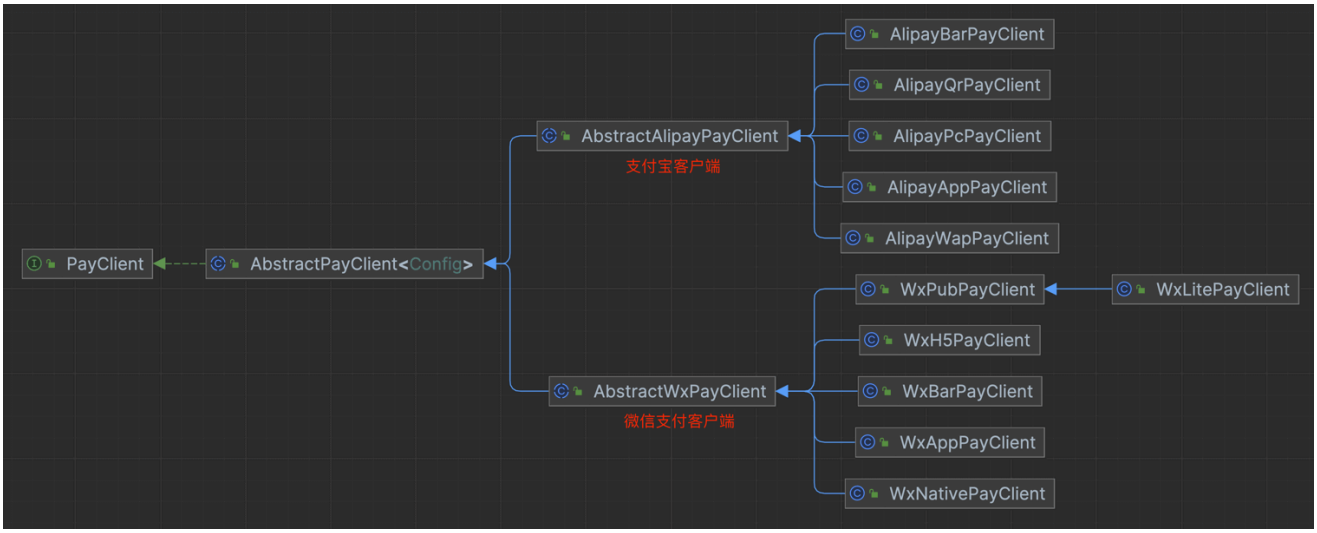
项目提供统一的支付中心，提供微信、支付宝等支付渠道的支付、退款等能力，方便业务模块进行快速接入，无需关注各种繁琐的支付 API。



1. 概述

它由如下 3 部分组成：

① `yudao-spring-boot-starter-biz-pay` 组件：对接微信、支付宝等支付，提供统一的 PayClient 支付客户端。



② `yudao-module-pay` 后端模块：实现支付中心的后端功能，包括支付、退款等能力。

- 基于 PayClient 支付客户端，对接微信、支付宝等支付渠道。
- 对内提供 PayOrderApi 统一支付 API 能力、PayRefundApi 统一退款 API 能力。

③ 支付中心的前端，提供支付中心的管理后台，可进行支付渠道的配置、支付订单、退款单的查看等操作。

- Vue2 版本：`@/views/pay` 目录
- Vue3 版本：`@/views/pay` 目录

2. 功能开启

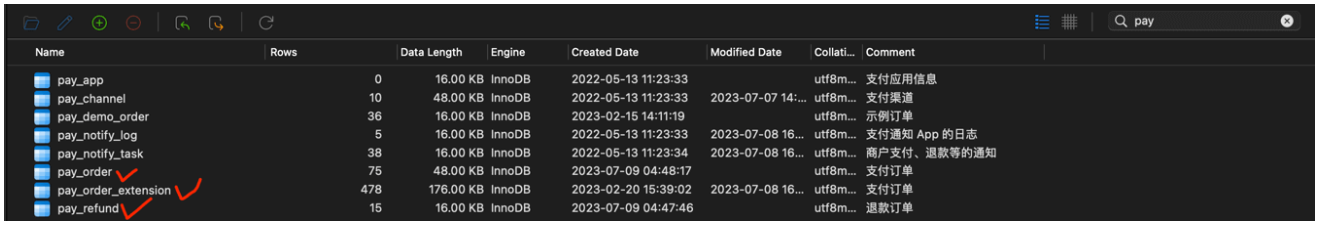
考虑到编译速度，默认 `yudao-module-pay` 模块是关闭的，需要手动开启。步骤如下：

- 第一步，导入支付的 SQL 数据库脚本
- 第二步，启动 `yudao-module-pay` 服务
- 第三步，开启支付相关的 Job 任务

2.1 第一步，导入 SQL

点击 `pay.sql` 下载，然后导入到数据库中。

> **友情提示：↑↑↑ pay.sql 是可以点击下载的！ ↑↑↑**

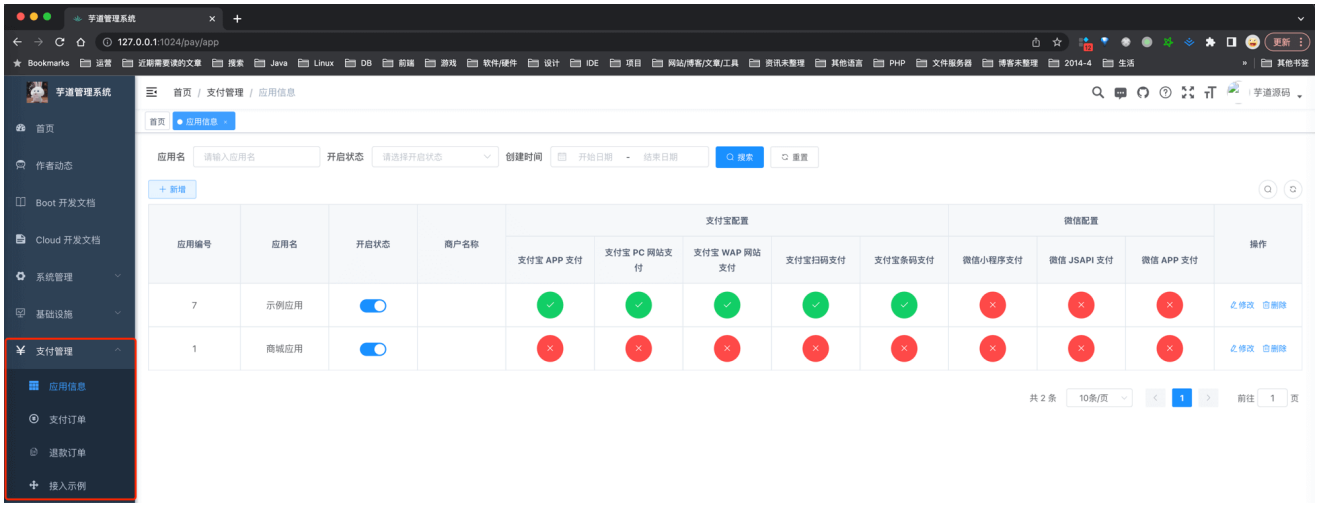


Name	Rows	Data Length	Engine	Created Date	Modified Date	Collati...	Comment
pay_app	0	16.00 KB	InnoDB	2022-05-13 11:23:33		utf8m...	支付应用信息
pay_channel	10	48.00 KB	InnoDB	2022-05-13 11:23:33	2023-07-07 14:...	utf8m...	支付渠道
pay_demo_order	36	16.00 KB	InnoDB	2023-02-15 14:11:19		utf8m...	示例订单
pay_notify_log	5	16.00 KB	InnoDB	2022-05-13 11:23:33	2023-07-08 16...	utf8m...	支付通知 App 的日志
pay_notify_task	38	16.00 KB	InnoDB	2022-05-13 11:23:34	2023-07-08 16...	utf8m...	商户支付、退款等的通知
pay_order	75	48.00 KB	InnoDB	2023-07-09 04:48:17		utf8m...	支付订单
pay_order_extension	478	176.00 KB	InnoDB	2023-02-20 15:39:02	2023-07-08 16...	utf8m...	支付订单
pay_refund	15	16.00 KB	InnoDB	2023-07-09 04:47:46		utf8m...	退款订单

2.2 第二步，启动 pay 服务

① 运行该服务的 PayServerApplication 启动类，看到 `"Started PayServerApplication in 18.105 seconds"` 说明开启成功。

② 然后，访问前端的支付菜单，确认功能是否生效。如下图所示：

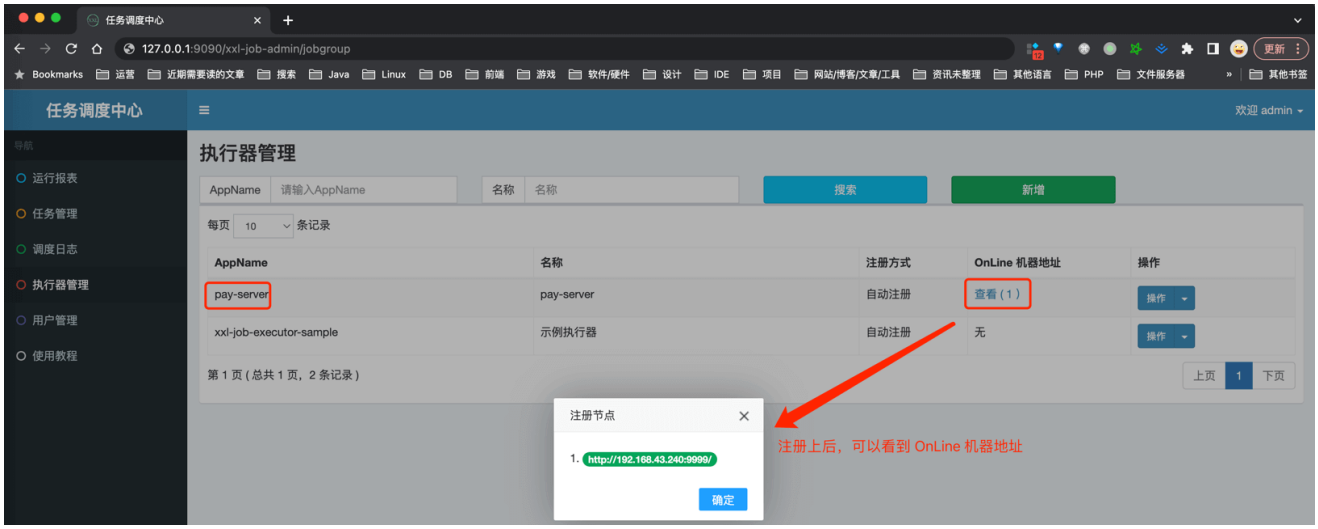


至此，我们就成功开启了支付的功能 🙂

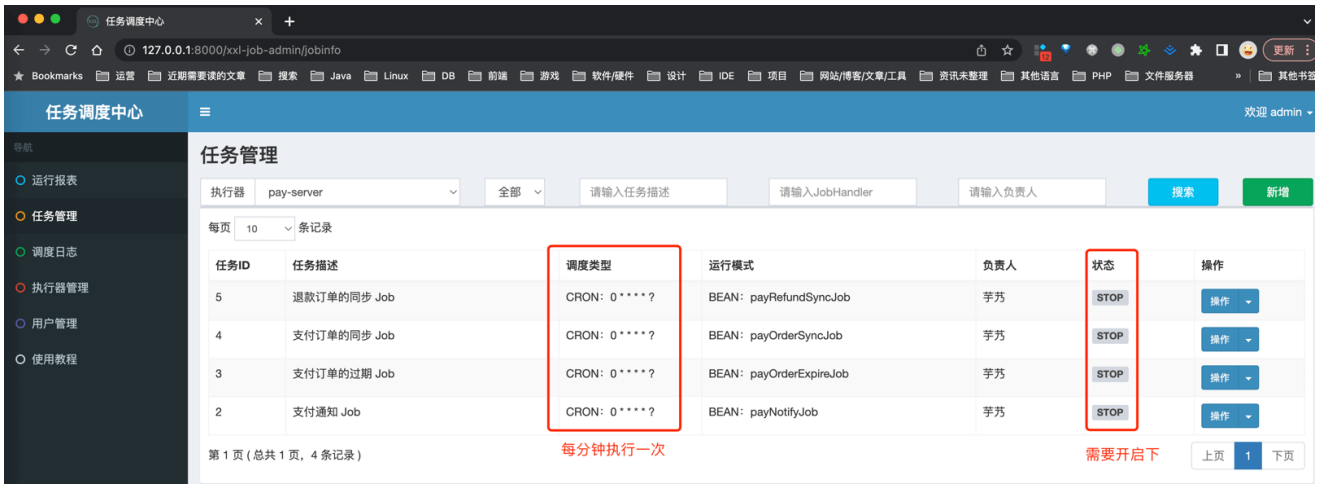
2.3 第三步，开启支付 Job

① 参考 《定时任务》 文档，将 Job 定时任务开启。

② 在 XXL-Job 的 [执行器管理] 菜单，添加 `pay-server` 执行器。然后，需要重启 PayServerApplication 服务，成功注册到 XXL-Job 上。如下图所示：



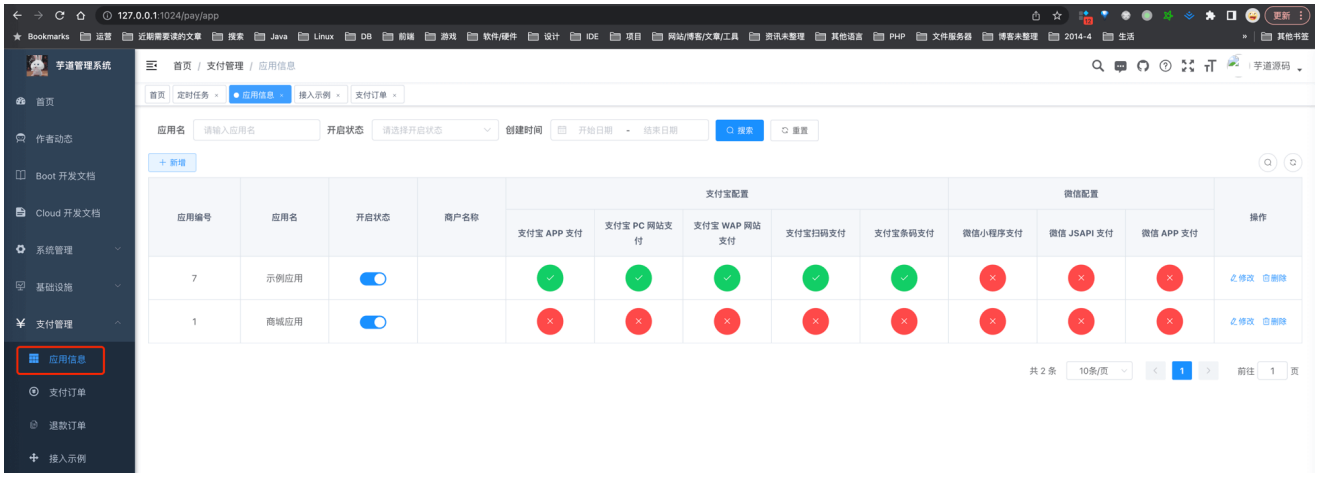
③ 在 XXL-Job 的 [任务管理] 菜单，添加 `payNotifyJob` 、 `payOrderSyncJob` 、 `payOrderExpireJob` 、 `payRefundSyncJob` 任务，并进行开启。如下图所示：



3. 功能介绍

3.1 应用信息

对应 [支付管理 -> 应用信息] 菜单，进行支付渠道、支付应用的管理。如下图所示：



3.1.1 支付应用

每个要接入支付中心的业务，对应一个支付应用。例如说：商城订单算一个应用，预约订单算一个应用。

点击【新增】按钮，可以进行支付应用的配置，保存在 `pay_app` 表。如下图所示：

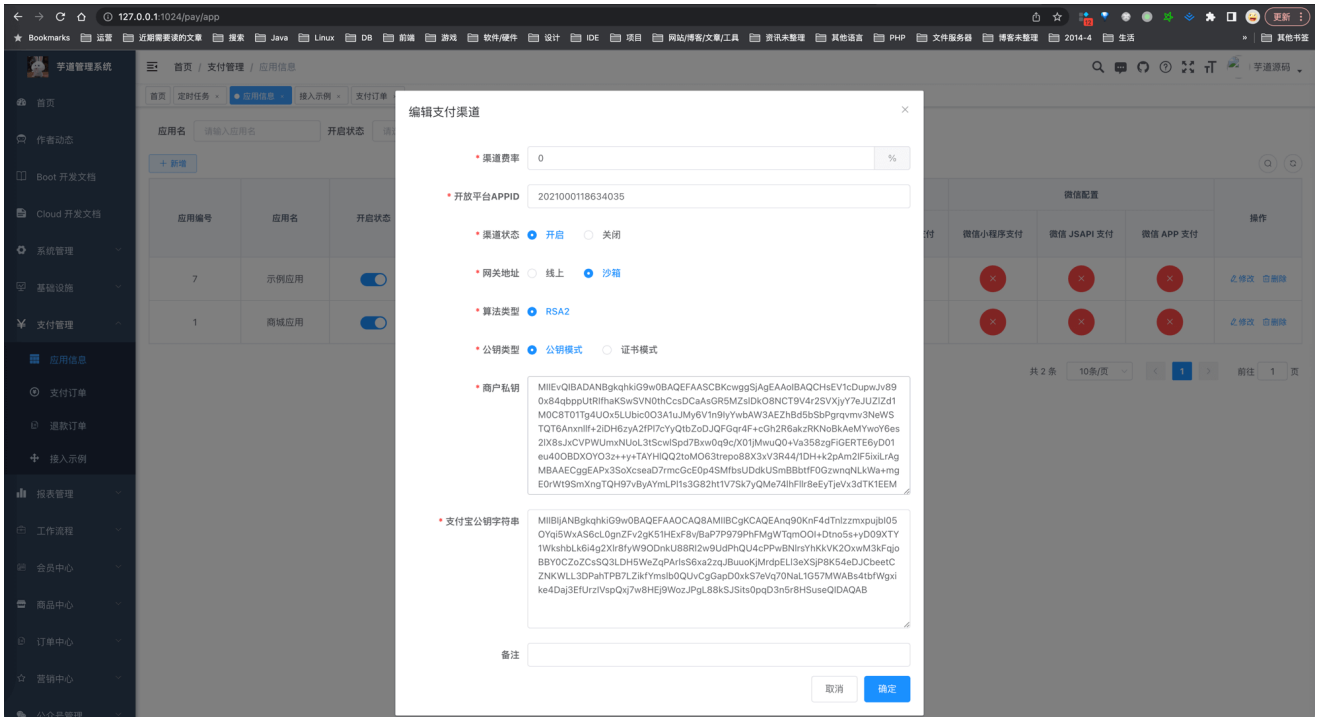
- 支付结果的回调地址：每个业务需要实现一个支付回调接口，在用户支付成功时，支付中心会进行回调。示例 1、示例 2
- 退款结果的回调地址：每个业务需要实现一个退款回调接口，在用户退款成功时，支付中心会进行回调。示例 1

> **为什么要有支付应用？直接配置支付渠道不行吗？**
>
> 1. 一个系统中，可能有多个业务需要，每个业务的支付、退款回调地址不同。
> 2. 同时，每个业务的订单编号可能重复，需要使用支付应用进行隔离，只要求在每个支付应用下保持唯一即可。
> 3. 另外，每个业务可能想要配置不同的支付渠道。

3.1.2 支付渠道

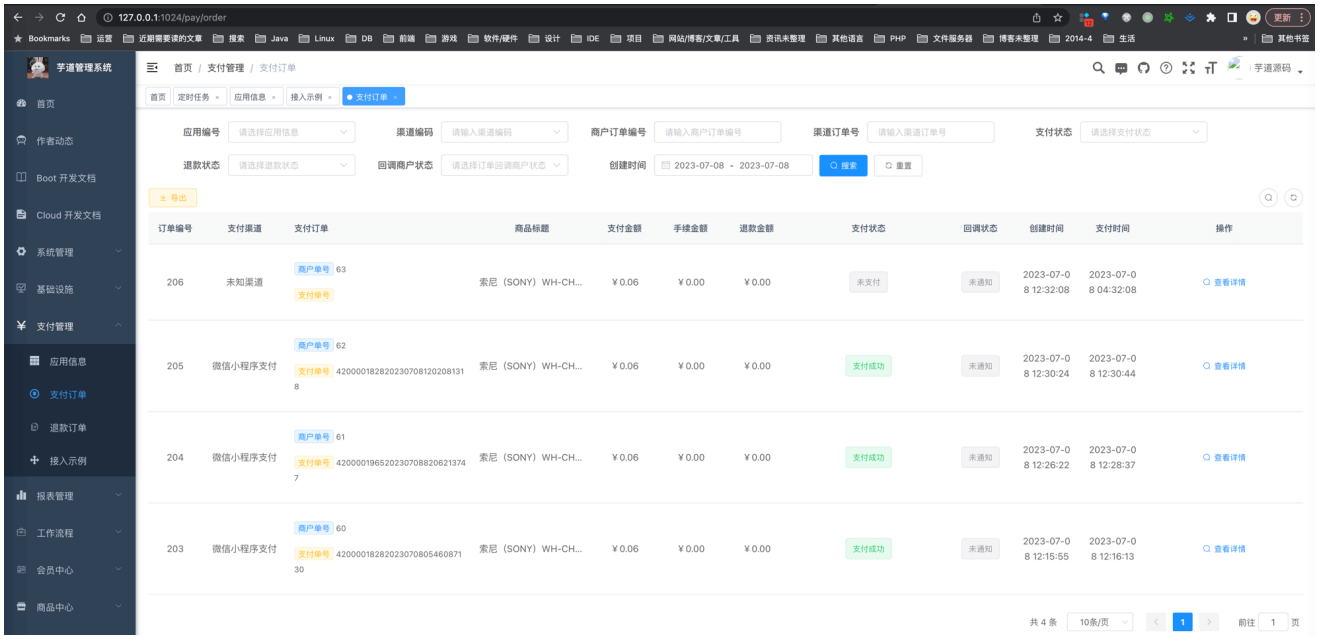
每个支付应用下，可以配置多个支付渠道。例如说：这里“示例应用”就配置了支付宝 PC 网站支付、支付宝 Wap 网站支付等等。

点击【√】或者【×】图标，可以进行支付应用的配置，保存在 `pay_channel` 表。如下图所示：



3.2 支付订单

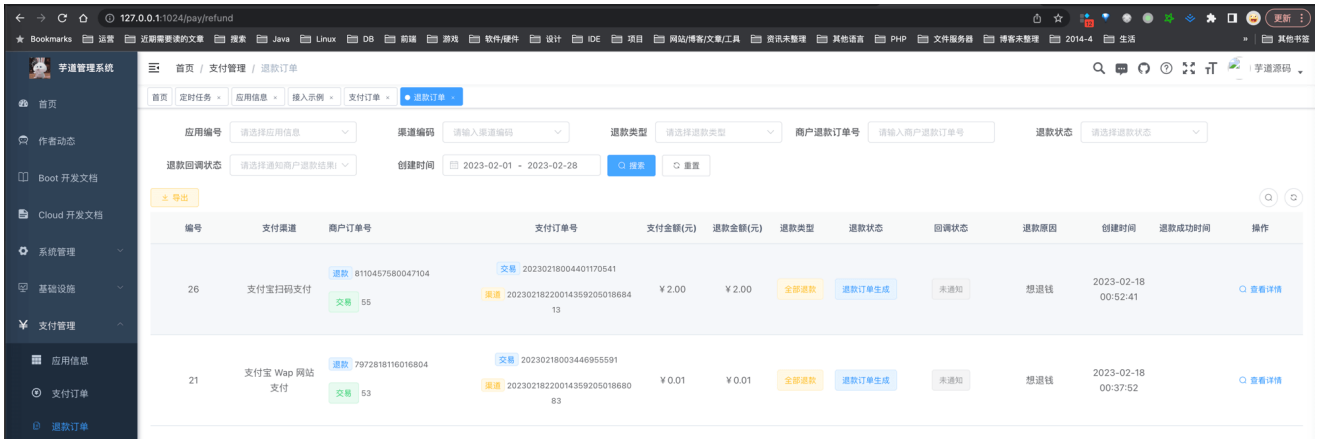
对应 [支付管理 -> 支付订单] 菜单，进行支付订单的查看。如下图所示：



一般情况下，每个业务订单对应一条支付订单，保存在 `pay_order` 表，通过 `merchant_order_id` 字段关联。

3.3 退款订单

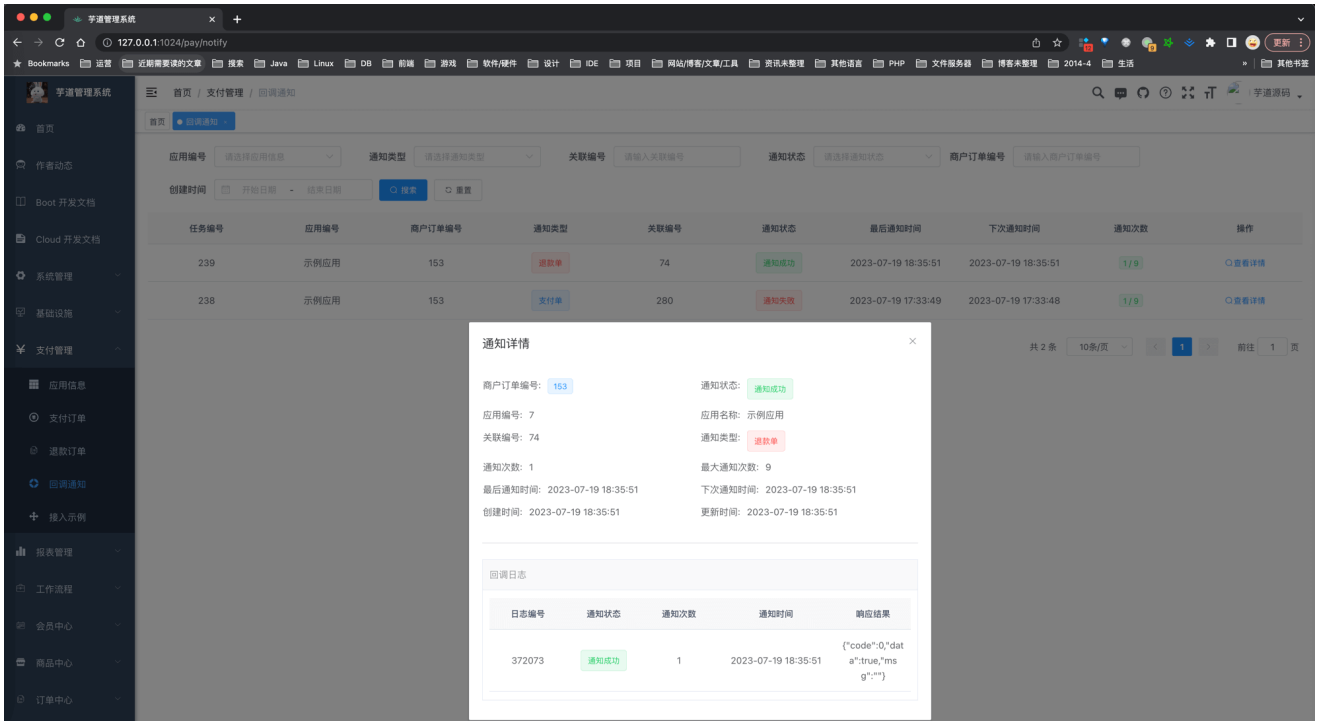
对应 [支付管理 -> 退款订单] 菜单，进行退款订单的查看。如下图所示：



一般情况下，每个业务退款对应一条退款订单，保存在 `pay_refund` 表，通过 `merchant_refund_no` 字段关联。

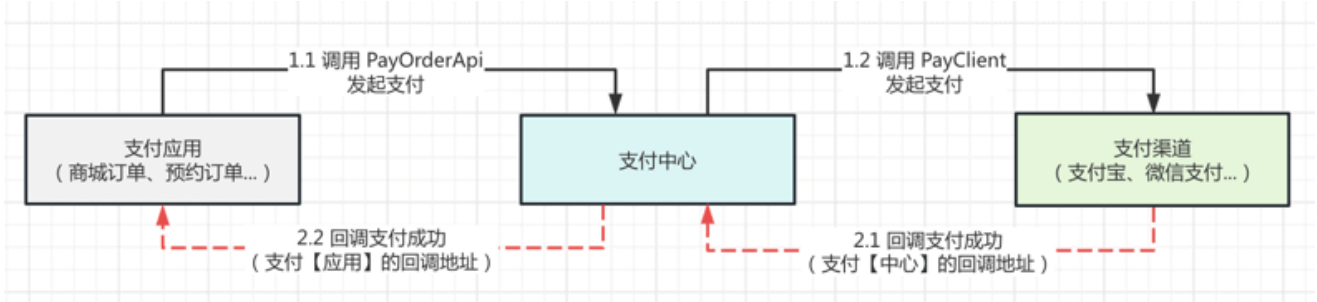
3.4 回调通知

对应 [支付管理 -> 回调通知] 菜单，查看支付、退款的回调业务的结果。如下图所示：



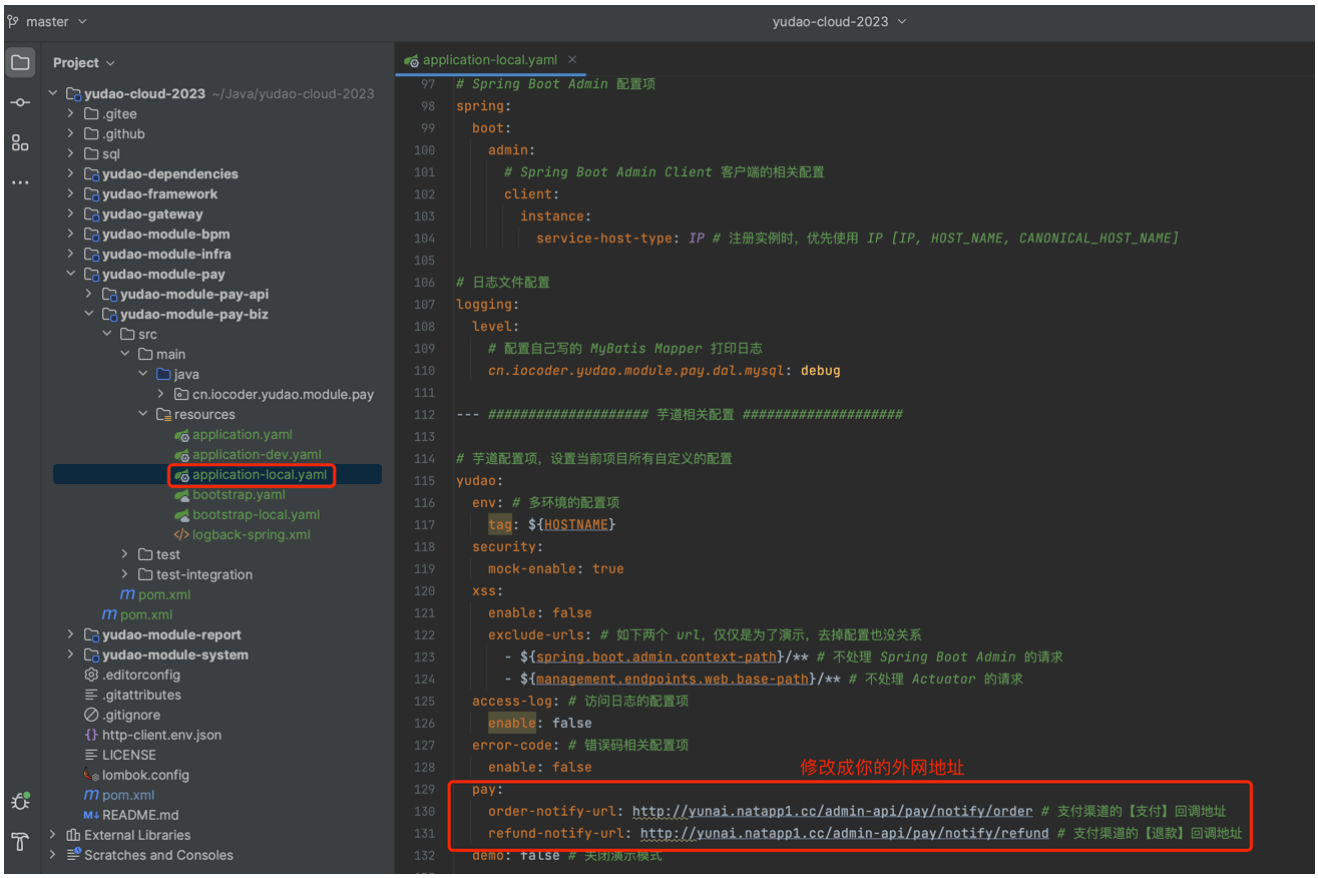
3.5 支付回调【重要】

这里，我们要配置支付【**中心**】提供给支付【**渠道**】的回调地址，不同于上面支付【**应用**】的回调地址。整体的回调关系如下图所示：



① 由于支付回调需要外网，可参考 《内网穿透》 文档，将本地的 48080 端口，转发到外网中。这里，我的域名是 `http://yunai.natapp1.cc` 。

② 在 `application-local.yaml` 配置文件中，修改 `yudao.pay` 配置项，设置为支付【中心】的回调地址。如下图所示：

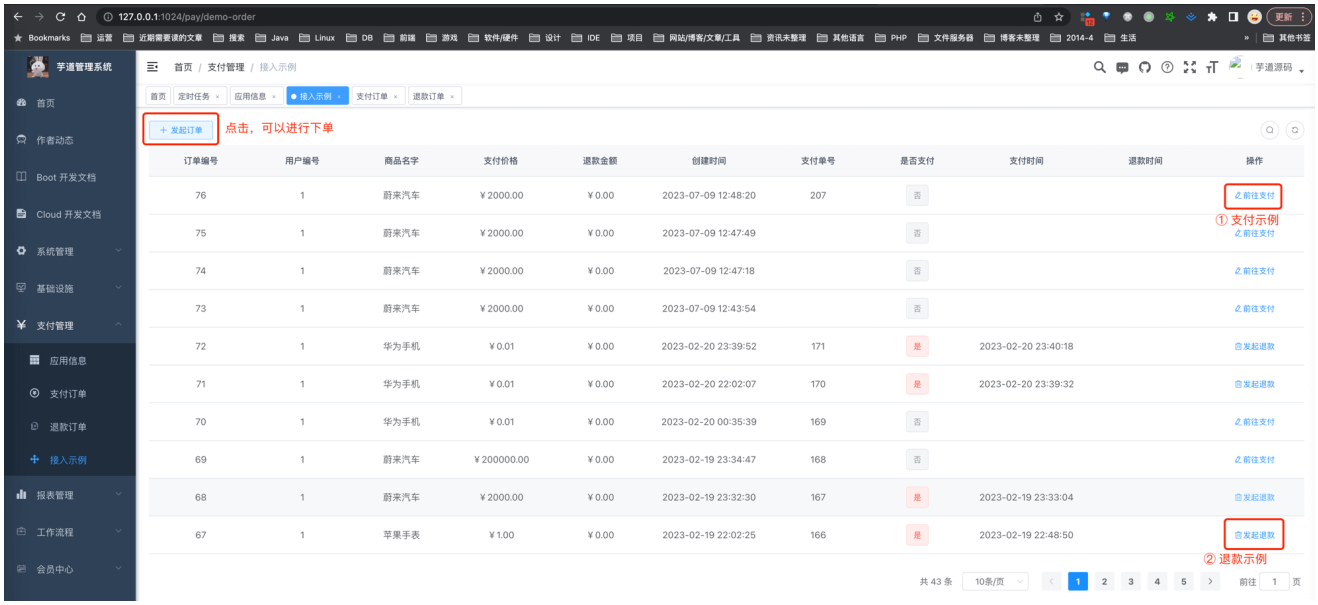


- `yudao.pay.order-notify-url` 配置项：对应 PayNotifyController 的 `#notifyOrder(...)` 方法
- `yudao.pay.refund-notify-url` 配置项：对应 PayNotifyController 的 `#notifyRefund(...)` 方法

如果你想理解的更深入，可以后续 debug 断条调试。

3.6 接入示例

对应 [支付管理 -> 接入示例] 菜单，提供一个支付、退款的接入示例。如下图所示：



详细说明，可见如下文档：

- 《支付宝支付接入》
- 《支付宝、微信退款接入》

← 大屏设计器 支付宝支付接入→