开发手册

美宜佳MPOS系统开放平台

v0.0.13

美宜佳便利店有限公司
东莞彩星信息科技有限公司

美宜佳 MPOS 系统开放平台开发手册

0.0.1（非正式预览版）

廖宁

中国 - 东莞

1 引言

1.1 背景及定义

现 MPOS 系统作为一个独立系统的堆砌，存在一些诸如在扩展性、可靠性和维护成本上的一些问题。为此公司适时决策在开发工具不变的情况下对 MPOS 系统进行一次重构，故推出美宜佳 MPOS 系统开放平台（以下简称“平台”）作为项目的架构（不是框架）。

本手册作为平台的组成部分。对平台的结构和开发流程作了说明，提出了基本的规范和约束条件。

1.2 平台概述

平台是一个基于在提供基础的运行生态(主要包含日志、接口寄存、参数、主题和一种抽象图形接口等等)后，通过加载基本运行库和各业务模块的形式实现系统的运行。各业务接口间可以实现快速、模块化的开发，达到解耦、可插拔、可伸缩的目的。也为将来第三方的接入，以及顺应移动端发展的趋势做好准备。

做到信息系统设计的系统性、灵活性、可靠性、经济性和安全性要求。

有效的拆分应用，实现敏捷开发和部署

1.3 平台思路

现单体架构随着系统规模的扩大，它暴露出来的问题也越来越多，主要有以下几点：

1. 复杂性逐渐变高
2. 技术债务逐渐上升
3. 部署速度逐渐变慢
4. 阻碍技术创新
5. 无法按需伸缩。

可扩展性

可控性

系统的各种功能是由许许多多的不同对象协作完成的。在这种情况下，各个对象内部是如何实现自己的对系统设计来讲就不那么重要了；而各个对象之间的协作关系则成为系统设计的关键。小到不同类之间的通信，大到各模块之间的交互，在系统设计之初都是要着重考虑的，这也是系统设计的主要工作内容。

面向接口编程我想就是指按照这种思想来编程吧！实际上，在日常工作中，你已经按照接口编程了，只不过如果你没有这方面的意识，那么你只是在被动的实现这一思想；表现在频繁的抱怨别人改的代码影响了你（接口没有设计到），表现在某个模块的改动引起其他模块的大规模调整(模块接口没有很 好的设计）等等。

为什么要这么做

面向接口编程就是先把客户的业务逻辑线提取出来，作为接口，业务具体实现通过该接口的实现类来完成。当需求变化时，只需编写该业务逻辑的新的实现类，通过更改配置文件中该接口的实现类就可以完成需求，不需要改写现有代码，减少对系统的影响。

降低程序的耦合性。其能够最大限度的解耦，所谓解耦既是解耦合的意思，它和耦合相对。耦合就是联系 ，耦合越强，联系越紧密。在程序中紧密的联系并不是一件好的事情，因为两种事物之间联系越紧密，你更换 其中之一的难度就越大，扩展功能和 debug 的难度也就越大。
2 易于程序的扩展；
3 有利于程序的维护；

接口编程在设计模式中的体现：
开闭原则其遵循的思想是：对扩展开放，对修改关闭。其恰恰就是遵循的是使用接口来实现。在使用面向接口的编程过程中，将具体逻辑与实现分开，减少了各个类之间的相互依赖，当各个类变化时，不需要对已经编写的系统进行改动，添加新的实现类就可以了，不在担心新改动的类对系统的其他模块造成影响。

1.程序结构清晰，使用方便
2.接口经过合理设计后，有利于程序设计的规范化，并可以并行开发，提高工作效率
3.实现了程序的可插拔性，对于不同的需求切换不同的实现，降低了耦合度，随着系统复杂性的提高这个优势会越来越明显
4.允许多重实现，弥补了继承的缺陷

为什么是接口而不是类

实际上接口和抽象类的选择不是随心所欲的。要理解接口和抽象类的选择原则，有两个概念很重要：对象的行为和对象的实现。如果一个实体可以有多种实现方式，则在设计实体行为的描述方式时，应当达到这样一个目标：在使用实体的时候，无需详细了解实体行为的实现方式。也就是说，要把对象的行为和对象的实现分 离开来。既然 Java 的接口和抽象类都可以定义不提供具体实现的方法，在分离对象的行为和对象的实现时，到底应该使用接口还是使用抽象类呢？

通过抽象类建立行为模型，通过接口建立行为模型

有人问 Jams Gosling（Java 之父）：“如果你重新构造 Java，你想改变什么？”。“我想抛弃 classes”他回答。在笑声平息后，它解释说，真正的问题不是由于 class 本身，而是实现继承(extends 关系)。接口继承（implements 关系）是更好的。你应该尽可能的避免实现继承。

现行条件不允许

编者的话
业务声明用接口，业务内部实现重用部分继承

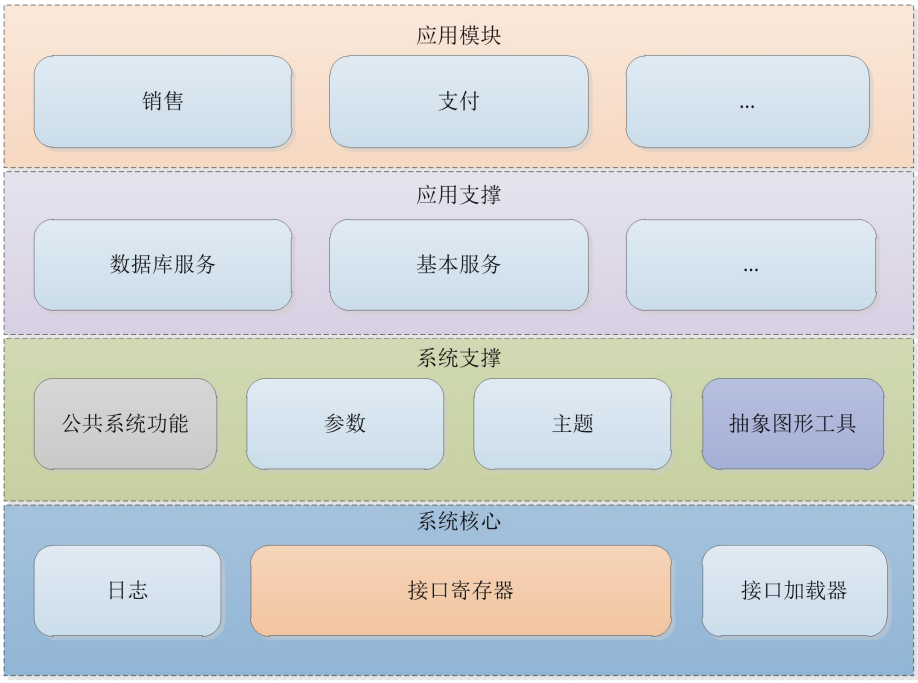
开发工具的局限性

平台不是一个框架，而是一次架构。作为一种抽象的解决方案

这不是一个框架，在类 Delphi 的体系下其实就只有一个框架，那就是 VCL。

2 体系结构

2.1 平台层次



2.1.1 系统核心层

提供接口寄存器、接口加载器和日志驱动。

2.1.2 系统支撑层

提供主题、参数、抽象图形接口，提供公共系统功能。

2.1.3 应用支撑层

应用支撑层是我们从事平台开发的基础，很多核心应用模块也是通过这一层来实现其核心功能的，该层可以简化一些组件的重用，开发人员可以直接使用其提供的组件来进行快速的应用程序开发，也可以通过拓展而实现个性化。

当然不是所有的应用模块都会依赖于应用支撑层的功能。TODO

a）数据库服务

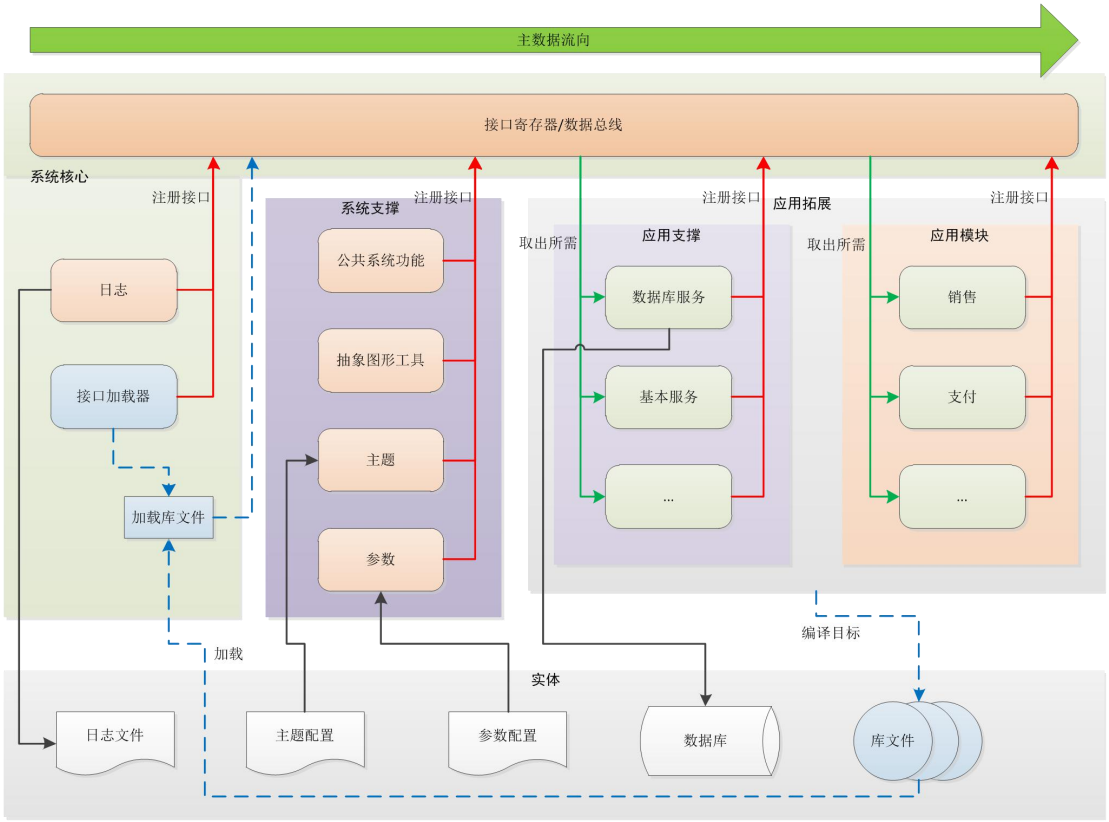
对应用模块提供基础数据库访问功能

b）基本服务

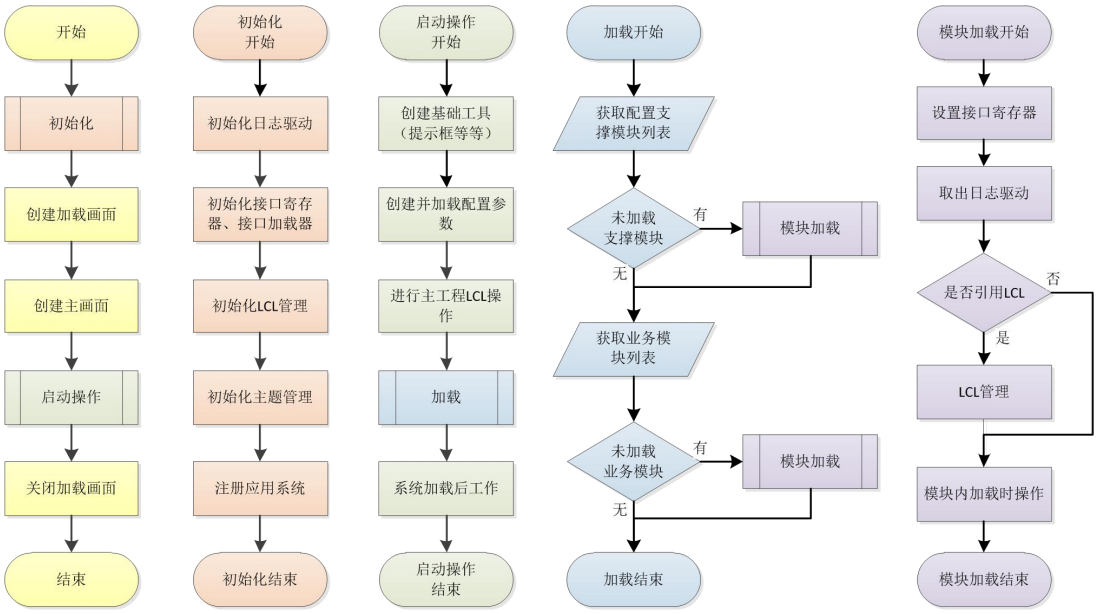
2.1.4 应用模块层

作为一个开放式平台可以自由的加入各应用模块，诸如销售、支付等应用模块。这些应用模块都是用指定的准入规范编写的，并且这些应用模块都是可以被开发人员开发的其他应用模块所替换，能够做到灵活和个性化。

2.3 工作形式



2.4 启动加载流程

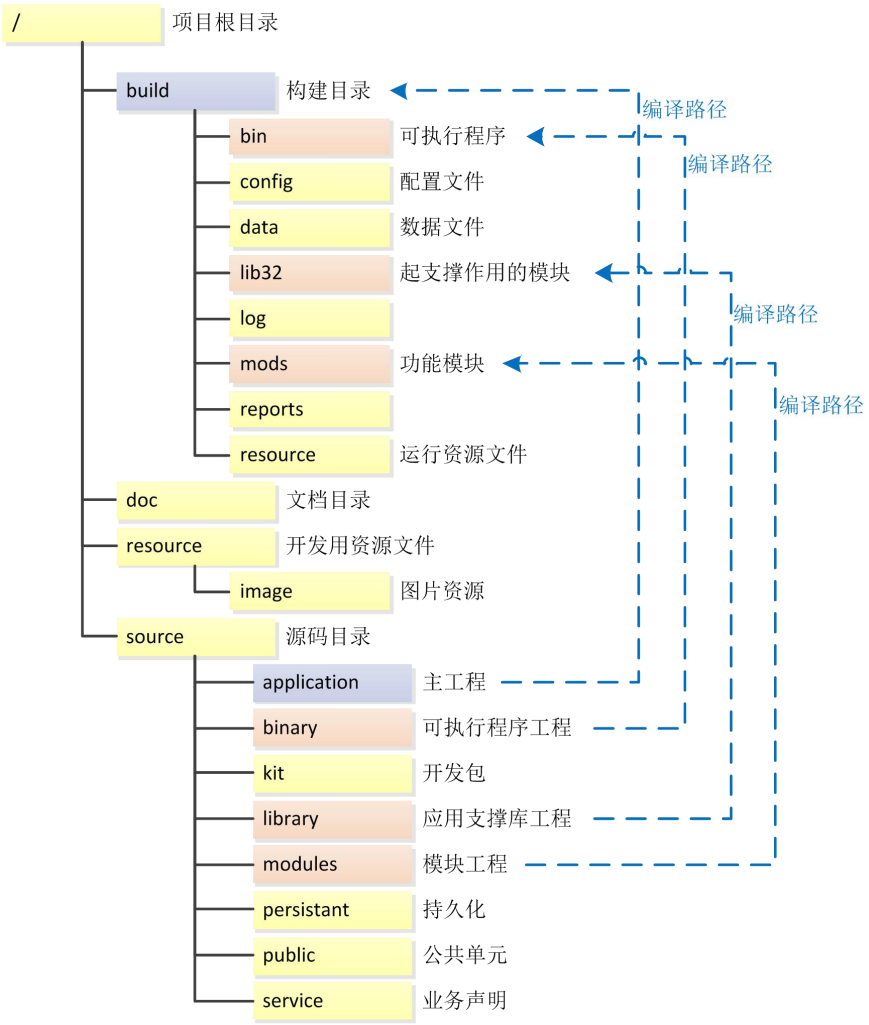


3 平台说明

目录规范

源码目录

```
// 向支付中心发起支付的请求
}
IPayRequest = interface(ICMBase)
['{4B93095D-F6DC-401B-8FC7-AF194E3D6D82}']
function GetOrderUUID: string;
function GetPayAmount: Currency;
function GetPayParameter: ICMConstantParameterDataList;
end;
```



目录安排

代码路径
主项目 application
bin
  可直接运行的文件和基本支撑库
lib
  程序运行时的依赖文件

在 lib 这个目录下面,一定有很很多以.jar 为后缀的文件(尤其是 dt.jar 和 tools.jar),这是压缩文件,你可以用 winRAR 解压查看的.SUN 公司发布的一些系统类就在这里,是 JAVA 程序运行所依赖的.例如:在 JAVA 程序在进行输入和输出的时候要用到很输入输出类,如 StreamInput,StreamOutput,你直接在程序的开头写上 import java.io.*,编译器就到会 lib 目录下找相关的系统类.

bin 的文件夹，里面提供了一些工具，一些命令，供开发或者运行 java 程序时调用
是 java 编译时需要调用的程序（如 java，javac 等）所在的地方。

开发文档应与项目目录一一对应。

基础生态环境

一个模块中可以包含多个业务

主要模块
  SYSTEM
  Parameter
  theme
  servlet 后续

接口的接入

基础生态
  启动流程

日志与异常

日志
　日志驱动
错误与异常
　不同包的异常是独立的，所以错误无法抛出包外
　所有异常都应在包内处理
异常可以分为系统异常（如网络突然断开）和业务异常（如用户的输入值超出最大范围），业务异常必须被转化为业务执行的结果。
　1．DataAccess 层不得向上层隐藏任何异常（该层抛出的异常几乎都是系统异常）。

　　2．要明确区分业务执行的结果和系统异常。比如验证用户的合法性，如果对应的用户 ID 不存在，不应该抛出异常，而是返回（或通过 out 参数）一个表示验证 结果的枚举值，这属于业务执行的结果。但是，如果在从数据库中提取用户信息时，数据库连接突然断开，则应该抛出系统异常。

　　3．在有些情况下，BL 层应根据业务的需要捕获某些系统异常，并将其转化为业务执行的结果。比如，某个业务要求试探指定的数据库是否可连接，这时 BL 就需要将数据库连接失败的系统异常转换为业务执行的结果。

　　4．UI 层(包括 Service 层)除了从调用 BL 层的 API 获取的返回值来查看业务的执行结果外，还需要截获所有的系统异常，并将其解释为友好的错误信息呈现给用户。

持久化

开发协作方式
开发规范
　　六大原则
　　mods 应统一命名前缀
UI 层和 BL 层禁止出现任何 SQL 语句。
不允许显示创建计时器
对外公布的方法（过程、函数）应在方法体开始和结尾输出调试日志。

数据库

单元测试

调试

基础说明
InterfaceRes

附
一个 hello world 业务的开发过程
主要模块
数据服务

基本服务

销售

支付

时间线

监控

配置

4 开发流程与开发规范

2.1 开发环境
"Pascal Visual Programming Studio"CodeTyphon
基于 6.5

单一职责原则

意思是每个微服务只需要实现自己的业务逻辑就可以了，比如订单管理模块，它只需要处理订单的业务逻辑就可以了，其它的不必考虑。

服务自治原则

意思是每个微服务从开发、测试、运维等都是独立的，包括存储的数据库也都是独立的，自己就有一套完整的流程，我们完全可以把它当成一个项目来对待。不必依赖于其它模块。

轻量级通信原则

首先是通信的语言非常的轻量，第二，该通信方式需要是跨语言、跨平台的，之所以要跨平台、跨语言就是为了让每个微服务都有足够的独立性，可以不受技术的钳制。

接口明确原则

由于微服务之间可能存在着调用关系，为了尽量避免以后由于某个微服务的接口变化而导致其它微服务都做调整，在设计之初就要考虑到所有情况，让接口尽量做的更通用，更灵活，从而尽量避免其它模块也做调整。

开发指引

线下当面付产品支持条码支付、扫码支付、声波支付。

条码支付（接入指引）
条码支付是支付宝给到线下传统行业的一种收款方式。商家使用扫码枪等条码识别设备扫描用户支付宝钱包上的条码/二维码，完成收款。用户仅需出示付款码，所有收款操作由商家端完成。

业务流程：

接入步骤

附
图形界面方案

一、与技术现状说

图形界面并不是 linux 的一部分，linux 只是一个基于命令行的操作系统。在 Linux 下常用的图形界面库有 GTK 和 QT。
目前我们使用的 LCL 在 Linux 下是采用 GTK 的，而 Windows 下并没有采用 GTK 而是直接用的 Windows 的 API。

主程序创建一个窗体，在库中增加需要的组件

直接在库中创建窗体

作为编译型语言
在编译时将窗体文件解析成相应信息放入类型信息中，
动态创建的控件类，只能是已注册的类型。而在不同库中的类型是无法获知的，尽管它们的内存结构可能一致。

也就是第一个 LCL 都是单独管理，无法实现模态窗口

当然你可以有你自己更好的解决方案，但要符合一原则：“不要影响别人，也不要人来管理你”

常见问答

参考文献

[1] 泊川.面向接口编程优缺点[EB/OL].https://blog.csdn.net/wantken/article/details/31763669，2018-10-27.
[1] 有梦就能实现.软件各种架构图收集[EB/OL].https://www.cnblogs.com/firstdream/p/7145481.html，2018-10-27.
[2] 刘腾红.信息系统分析与设计[M].北京:清华大学出版社，2010.9.

法律声明

本手册是依据《阿里巴巴 Java 开发手册》进行适应性改编的，阿里巴巴集团仅授权供大家交流、学习及研究使用，禁止用于商业用途。所以使用本手册亦当遵守上述规定，违者责任自负。