# 快乐点餐系统

版权：XXXXXX公有限司

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **人员** | **记录** |
| V.1.0.0 | 2019-07-24 | 张冬 | 创建 |
|  |  |  |  |

# 背景

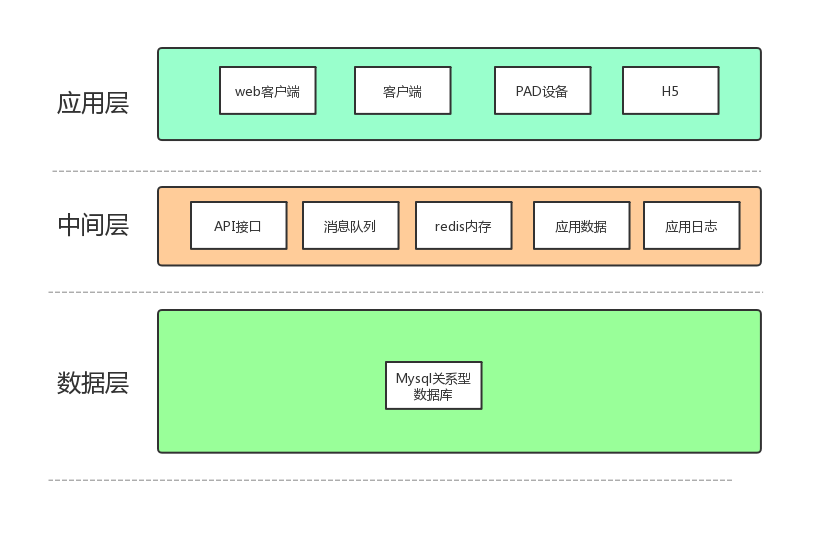
随着餐饮业的不断发展，人们对于餐饮的管理和服务的需求不断的提升，本系统只要解决餐厅的管理问题、业务流水记录和服务评价，让管理者发现问题并且改善问题。

# 名词术语

* 分布式
* 消息队列
* 微服务

# 系统设计

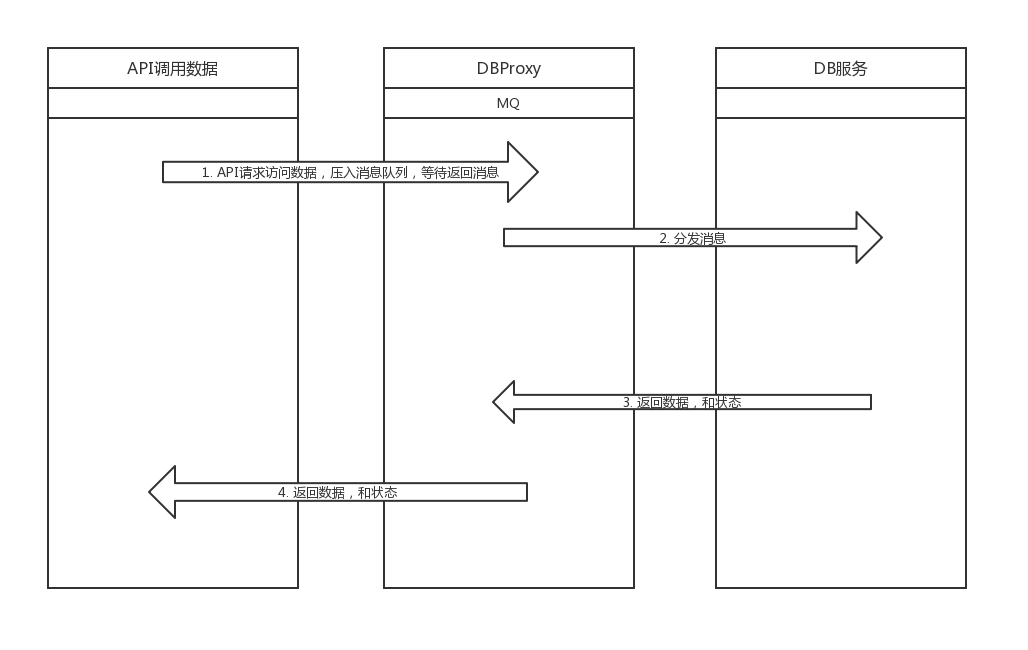
## 系统总体架构



系统主要分成应用层、中间层和数据层，其中：

* 应用主要是业务的操作：服务员PAD点餐、客户微信点餐，后厨菜单显示大屏，餐桌订单进度显示，前台操作，后台ADMIN web管理
* 中间层：API服务、消息队列服务、数据操作服务
* 数据层：Mysql数据库

## 微服务逻辑架构图



# 主要设计思路和折衷方法

1. 设备访问资源（数据库、文件等等），要先经过API服务，认证身份
2. API获取客户端发送的身份进行验证
   1. 如果验证失败，返回没有权限（401）
   2. 验证成功，进行资源的访问
3. 资源提供者（provider），得到访问的请求，发送一条消息到消息队列中，将持久化到硬盘，立刻监听返回对列中的数据
4. 资源消费者（consumer），获取一条消息，操作数据库，并且将结果返回给制定的消息队列中，并且返回ACK应答
5. 资源提供者（provider），监听到返回的数据，返回给API服务
6. API服务返回给客户端

几点考虑

1. 当rabbitmq请求超时如何处理，API服务请求数据微服务链接断开如何处理

业务量大，将rabbitmq单个服务替换成集群服务，配合Haproxy负载均衡软件，使得mq集群具有高可用的特点。

业务量小，设置rabbitmq的超时时间

1重试的次数，当达到最大的重试次数的时候，将这个失败的消息，放到数据库里面

2消息通知到界面上反应到界面上，保证在消息队列不可用的情况下下，消息不被丢失

3此时，需要一个哨兵服务，监控数据库的失败的消息，如果有，则重新放入消息队列中，并且记录失败的次数，并且反映到界面上

4失败的次数越多，失败的消息的优先级越高，就越被优先处理

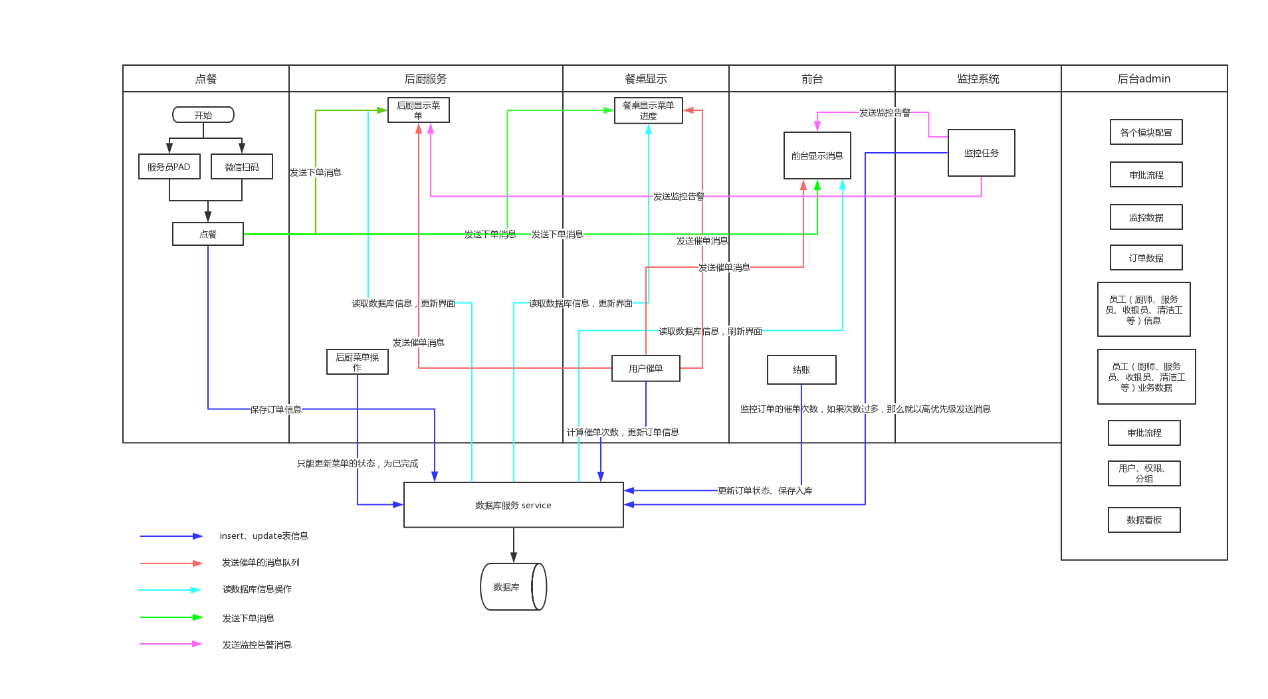
1. 数据库增量大，超过单张表的容量（查询的速度变慢）

按照月份、或者年份进行分表（根据业务量来区分）

# 逻辑流程图

## 总体流程图

总体流程图如下图所示：



系统业务层面，分为点餐客户端操作、后厨操作、餐桌订单操作、前台结账操作和监控系统，5部分组成

其中各个部分的流程图如下：

### 点餐流程

