EasyOrder 软件架构文档

一、功能性需求及非功能性需求：

功能性需求	非功能性需求		
	平台兼容性	开发阶段质量	运行阶段质量
顾客需求： 1.扫码 2.点餐 3.支付 4.评价 商家需求： 1.显示当前队列 2.显示当前成交额	客户端支持在不同手机浏览器中打开点餐网页，服务端支持 Windows 和 Linux 操作系统。	代码模块耦合度低，代码可拓展性、可重用性高、测试方便。	程序运行效率高，安全性良好。

前端

1. XSS 注入攻击
2. CSRF 攻击 etc... 参考： https://blog.csdn.net/fengyinchao/article/details/52303118

后端

1. 数据库保护(防注入)
2. 认证机制， 参考：https://www.jianshu.com/p/52768a0e7cc7

可靠性

1. 较多用户同时访问时， 服务器能否正常工作（负载抗压）
2. 在发生突发状况时（如环境系统未知错误崩溃，宕机，etc），需要有**日志信息**帮助应用恢复至崩溃前。

健壮性

测试应包含异常输入测试，检查程序是否进行过异常处理。

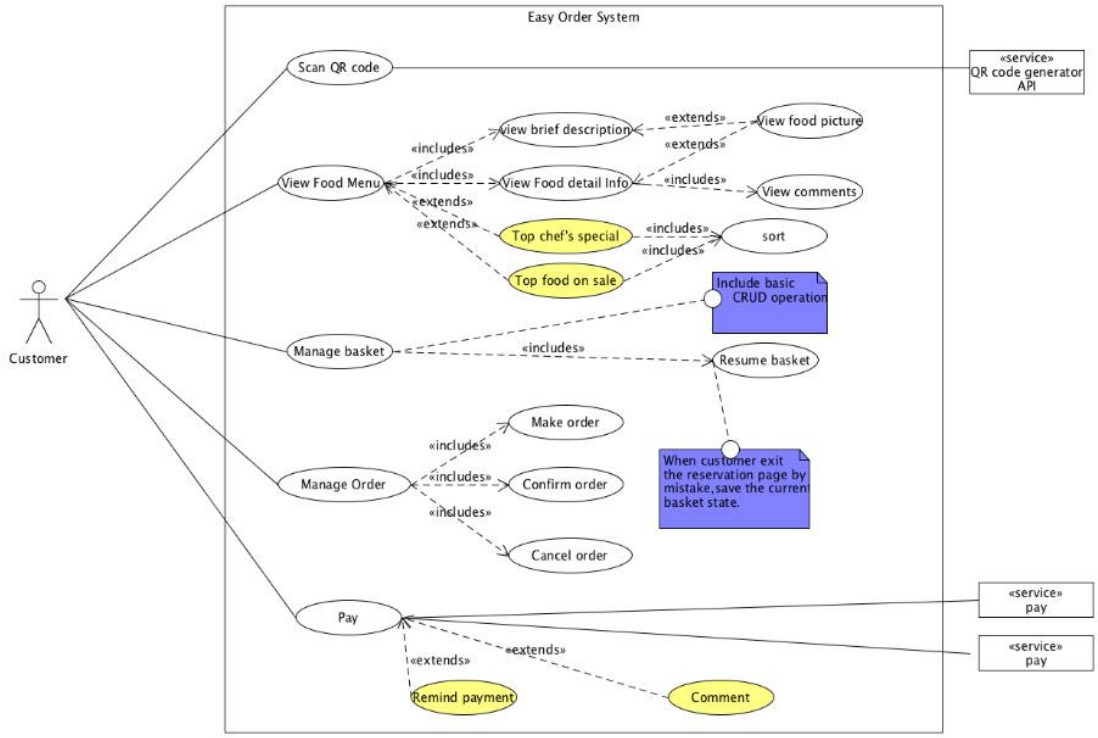
可移植性

1. 要求可以适应不同大小屏幕
2. 可以在不同系统环境运行（由于这个项目是一个 Web APP， 不存在这个问题）

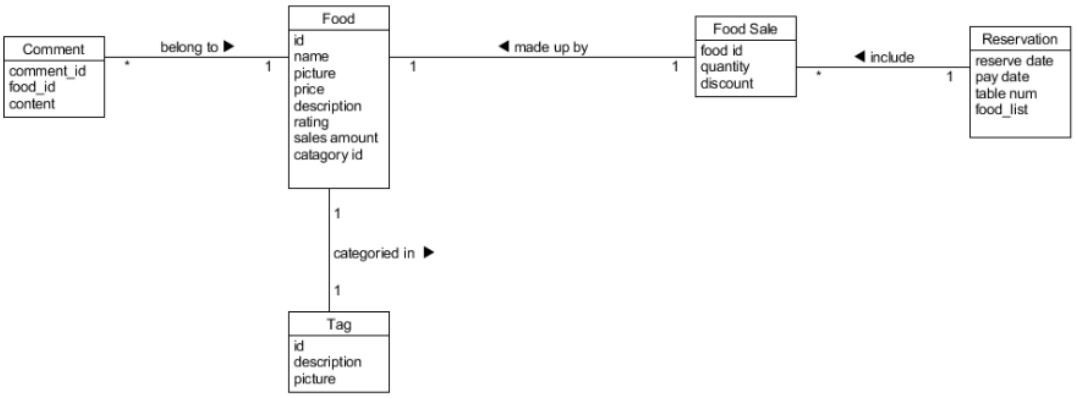
可扩充性

1. 要求系统严格遵循开发框架， 确保去掉不必要的耦合性
2. 接口设计应清晰合理， 方面拓展模块调用。

二、系统用例：



三、数据库设计



数据库初始化使用 ORM sqlalchemy，一共 4 个表

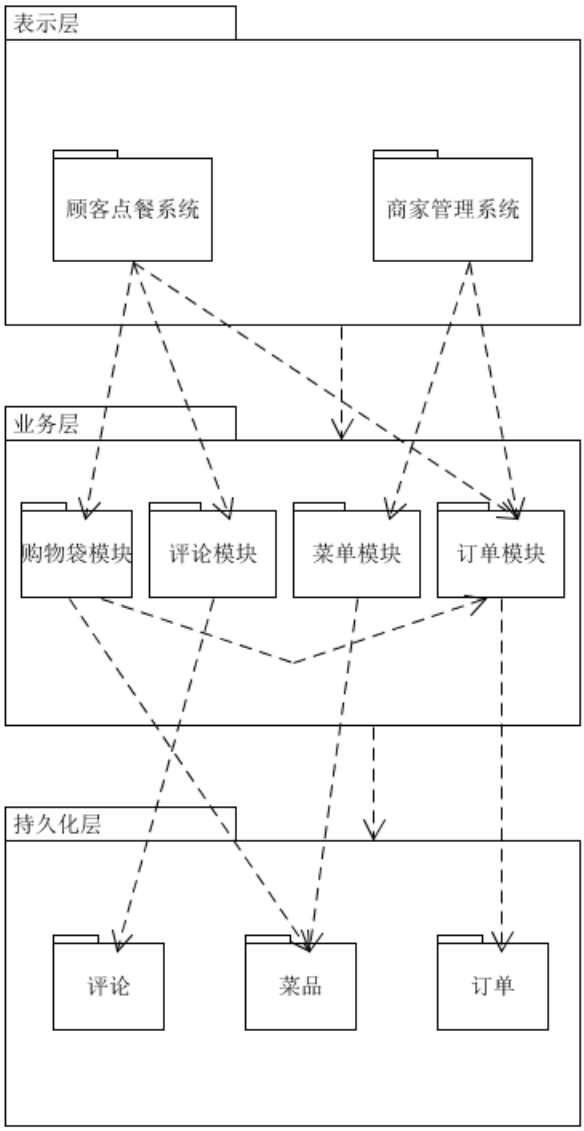
- tag 标签
- food 菜
- comment 评论
- reservation 订单

创建一个名为 model 的包来创建数据结构和初始化数据库，包结构为

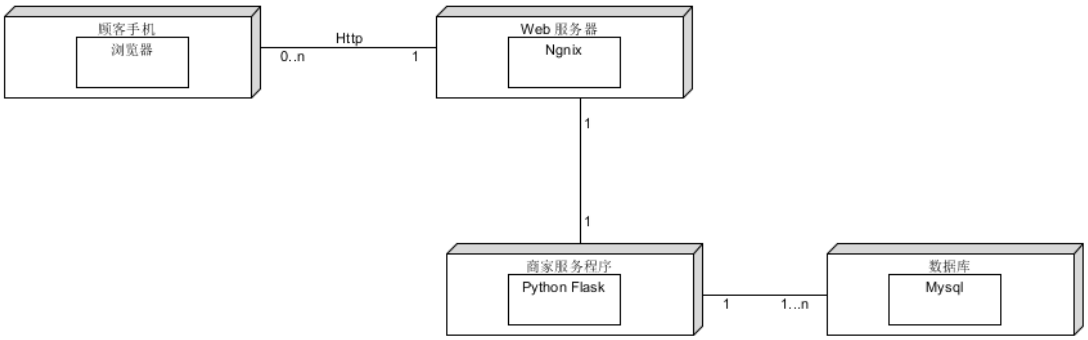
- model
- *_init_*.py
- base.py
- tag.py
- food.py
- comment.py

- reservation.py
- db_setting.py
- init_db.py

四、三层架构逻辑视图：



五、系统部署：



客户端：顾客使用的手机的浏览器即可，通过 Http/Https 协议与商家后台交互。
商家 Web 服务器：使用 Nginx 反向代理。
商家服务程序：在 Windows 或 Linux 系统上运行的 Python Flask 框架编写并执行的程序，用于接收处理顾客订单并与数据库交互。
数据库：使用 Mysql，持久性储存顾客订单、评价等信息，并可被服务程序读取。

六、软件质量
功能性
1.日志和错误处理：在持久层记录所有错误信息。
2.安全性：任何使用都需要经过用户认证。

可用性
1.反馈途径：顾客可以反馈用户体验。
2.界面简洁易懂无二义性。。

可靠性
1.可恢复性：如遇到系统因未知原因崩溃，能将系统回退到安全状态。
2.用户信息被加密保存，不会外泄。

可支持性
1.支持在各种操作系统平台下使用。
2.支持用户手动配置。

七、软件技术
Web 前端

HTML + CSS + Javascript(Vue.js)

Web 后端

Python(Flask 框架)

数据持久层

Mysql

八、代码规范：

代码规范

Web 前端

- 语言：HTML、CSS、Javascript
- 代码风格规范：Google HTML/CSS 代码风格指南、Google JavaScript Style Guide

Web 后端

- 语言：python
- 代码风格规范：谷歌 python 代码风格指南