北京大学食堂选菜系统
设计文档

——面向对象技术引论课程设计

00848024 张一沙

00848074 任玉鑫

00848234 陆　敏

大纲

PART 1 问题描述

PART2 OOA 设计

step1 建立需求模型【用况图】

step2 建立基本模型【类图】

step3 建立辅助模型【顺序图、活动图】

PART3 OOD 设计

step1 问题域部分的设计【类图】

step2 人机交互部分的设计【状态机图】

step3 数据库部分的设计【**RDBMS** 表】

step4 控制流部分的设计【顺序图】

PART4 总结

PART1 问题描述

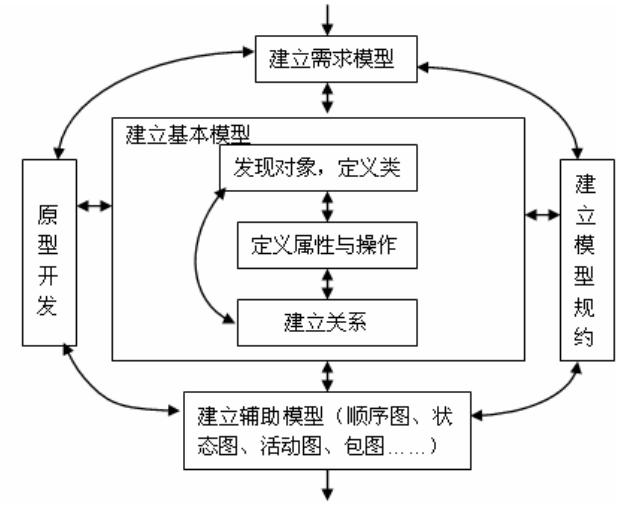
校内的食堂大约有十几家，每家食堂每天都提供品种丰富的菜样，作为食客的同学们每天都面临这一个“挑花眼”的问题，到底哪家食堂今天才有符合口味的菜品呢？而且，看上去美味的菜品究竟吃起来如何呢？有没有别人的意见可以参考呢？

于是我们决定设计一个简单的系统，解决食堂选择问题。

系统主要功能是用来根据食客的需求在当日的菜品数据库中寻找合适用户需求的菜品，并且食客可以给菜品打分，写评语。这样，每个就餐者都可以根据菜品的属性和口碑选择合适的食堂就餐了！

另外，我们希望今后在条件允许的情况下，在各个食堂门口安装来往人数检测的设施，并在菜品属性中加入食堂当前人数的属性，帮助同学们更好的避开食堂的用餐高峰，合理分配到各个食堂。

PART2 面向对象的设计（OOA 部分）



step1 建立需求模型

关键词：【用况图与用况描述】

我们的系统比较简单，并没有划分子系统的必要性。另外，参与者的发现也很简单，通过对需求的简单思考不难发现，我们系统中主要有两类参与者：食堂管理员和食客。

食堂管理员：每个食堂配有若干位，主要功能是每日将自己所在食堂的当日菜品的信息更新入系统，具体来说就是对菜品信息的添加、修改和删除，另外，为保证系统安全性，管理员行使职责的时候，需要进行身份验证。考虑到实际系统的可用性，我们还添加了密码的修改功能。如用况图所示，参与者食堂管理员与“身份验证”、“添加菜肴”、“修改菜肴”、“删除菜肴”、“修改密码”五种用况相对应。

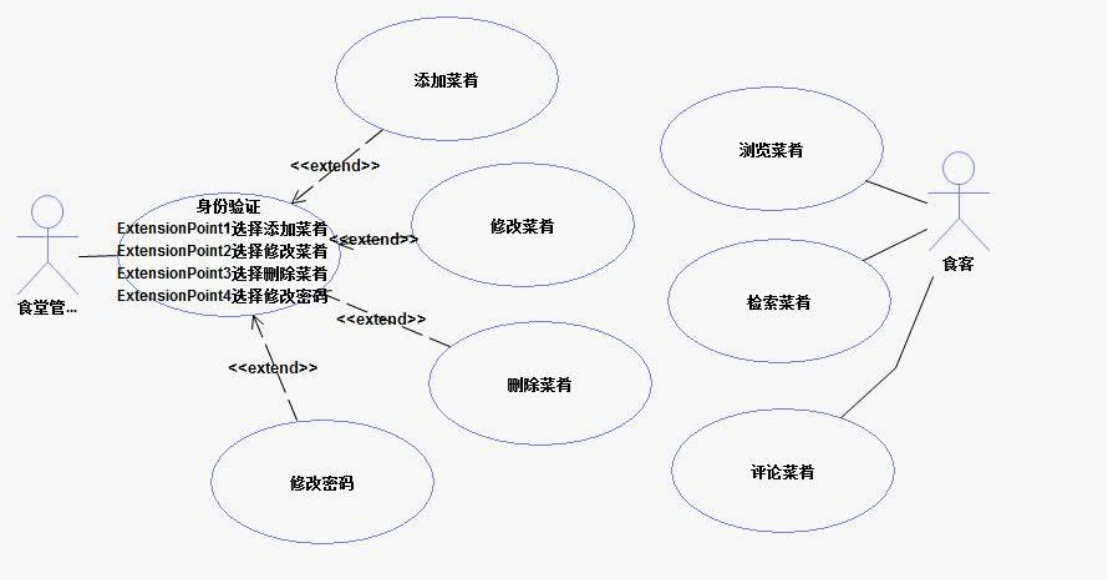


图 1 系统用况图

再考虑另一个参与者，食客，食客主要可以使用的功能是浏览菜肴，根据条件检索菜肴和评论菜肴，本系统中默认食客是访客身份，不用进行身份验证。如图所示，参与者食客与“浏览菜肴”、“检索菜肴”、“评论菜肴”三种用况相对应。

由于系统比较简单，参与者发现后，用况也很快被发现了。一些重要的用况描述如下：

1- 身份认证用况

主要完成管理员用户登录时的身份验证，登录成功后由管理员用户决定下一步需求。

用况：身份验证

用户选择管理员身份登陆
　　系统呈现身份验证界面
用户输入用户名和密码
　　if 输入信息有误
　　　提示重新输入，回到用况最初
　　else
　　{　进入系统
　　　提示选择服务类型
　用户选择服务类型
　　添加菜肴（扩展点）{如果客户选择了添加菜肴}extend 添加菜肴
　　修改菜肴（扩展点）{如果客户选择了修改菜肴}extend 修改菜肴
　　删除菜肴（扩展点）{如果客户选择了删除菜肴}extend 删除菜肴
　　修改密码（扩展点）{如果客户选择了修改密码}extend 修改密码
　　}

2- 添加菜肴

管理员通过该用况向系统内添加菜肴。

用况：添加菜肴

【前置条件：管理员已登录成功，且选择添加菜肴】

```
        系统显示菜肴登记表
管理员填写菜肴登记表并提交
        if 信息未填写完全
            提示信息不全，返回上一步的登记表
        else
        {
            提示菜肴添加成功并显示新添加的菜肴信息
            询问管理员下一步是继续添加还是返回
    管理员输入请求
            if 选择继续添加
                则回到本用况起始处
            else
                退出用况，返回主页面
```

删除菜肴和修改菜肴的用况描述和添加菜肴十分类似，在此不加赘述。下面来看密码修改用况。

3- 密码修改

该用况主要提供管理员修改个人信息的功能。今后如有需要可以扩展到除了密码之外的其他信息的修改上。

用况：修改密码

【前置条件：管理员已登录成功，且选择修改密码】

```
        系统提示输入新老密码
管理员输入新老密码
        if 输入旧密码有误
            返回用况最初
        else
        {   提示再次输入新密码
    管理员再次输入新密码
            if 两个新密码不相等
                则回到本用况起始处
            else
                提示密码修改成功
        }
```

用户浏览菜肴用况太过简单，此处不展开。

4- 检索菜肴

该用况为用户提供检索信息的输入接口，用户可以填部分信息，数据库据此进行查找。

```
用况：检索菜肴
用户选择检索菜肴选项
        系统显示菜肴检索信息表
用户填写菜肴检索信息表并提交
        系统根据输入信息在数据库内查找
        返回查找结果
```

5- 评价菜肴

该用况允许用户不记名的输入对任意菜品的评价信息。文字评价和分数评价均有。其中文字评价直接显示在菜肴下方而分数

```
用况：评价菜肴
【前置条件：用户点开了一个菜的详细信息】
        系统显示菜肴评价表
用户填写菜肴评价表并提交
        if 信息未填写完全
            提示信息不全，返回原评价表
        else
            显示新提交的评语，并重新计算菜肴平均分
```

评价加权后与之前的评分一起得出新的评分。

step2 建立基本模型

关键词：【类图】

1- 发现对象、定义类

2- 定义属性和操作

3- 建立关系

将现实世界与计算机逻辑相对接是十分困难的，画好了需求模型后，我们就要开始为建立完整的类图做准备了。通过对需求模型的研究，我们发现了以下类，并且定义了其上的操作，和类之间的关系。

1) 菜肴

菜品是我们分类检索、输入更新的元对象。其中设立菜名、价格、风味、原料等几个属性作为搜索时的索引项；得分和评论两个属性会显示在菜肴介绍中，但不能作为搜索条件。参与评论人数是为了加权计算平均分时为函数调用的。

管理员属性用来保证只有该菜肴属于的管理员才能对其修改。也即，学一食堂的管理员是不能修改农院食堂的菜肴信息的。这一点是由菜肴中的管理员属性实现的。

菜品的主要操作体现在：

a. 显示菜品信息，当菜单中的缩略图被点击时，调用该操作，完成菜肴全部信息的展示。
b. 评论菜肴，将评论的文字登记到数据库中，并更新显示在菜肴信息中。
c. 打分菜肴，根据每次打分，加权的计算出菜品当前得分，并且更新在数据库中。

2) 食客

系统的一个参与者，没有属性，但是是主动对象，会触发系统的各种行为，因此也要被单独列出来作为一个类。主要会体现在触发食客对应的三种用况上。

3) 管理员

系统的另一个参与者，有属性记录他的 ID 号，密码和所属食堂，这三个属性都是为了保障对数据的修改是安全的。

管理员属性提供的重要操作是对自身的密码进行修改。

4) 菜肴管理器

这是实现菜肴的统一管理以及系统数据与个人用户之间交流的类，对象保存着全部菜肴的信息，基本功能是提供菜肴列表的显示。

5) 菜肴前台管理器

与一般用户交互的管理器，继承了菜肴管理器的菜品信息属性，同时提供浏览和检索的功能。

6) 菜肴后台管理器

与管理员交互时的管理器，同样继承了菜肴管理器中的全部菜肴信息列表，更重要的是同时也保存着管理员列表，即其身份信息，提供身份验证函数，以及管理员可以实现的对菜肴的三种操作——添加菜肴、修改菜肴、删除菜肴。

4- 类图

管理员和食客都通过菜肴管理器来对菜肴实施操作，并不直接访问菜肴，这样可以保证菜肴信息的修改是一次只有一个对象在进行的，防止数据的混乱。以下是根据上述分析绘制的 OOA 阶段类图。

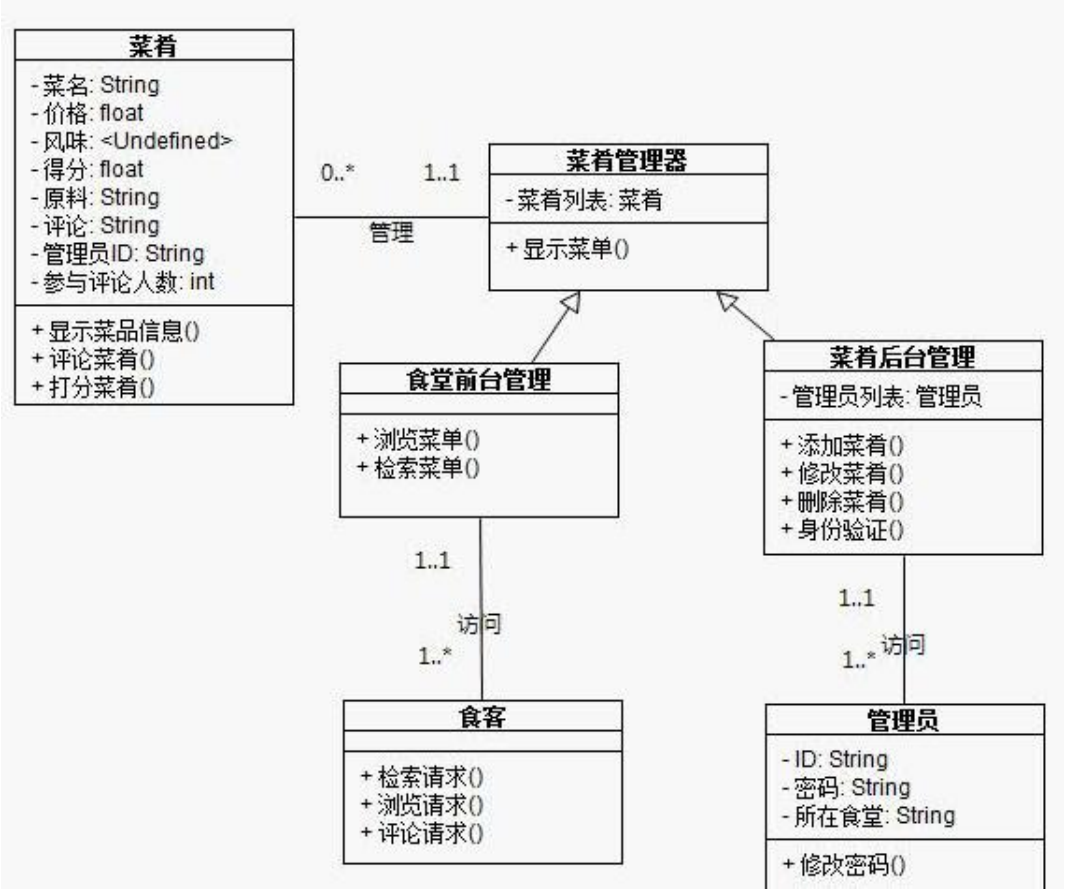


图 2 类图

图中的菜肴类为每个菜创建一个对象，并且可以提供打分和评论的接口。根据当前打分重新计算得分。

菜肴管理器将全部菜肴组织起来供管理员和食客浏览访问和修改。菜肴后台管理面向管理员，提供身份认证机制，表中有管理员列表，存储了所有管理员的信息。

step4 建立辅助模型

关键词：【顺序图、活动图】

1 顺序图

体现了多个对象间的交互过程，有助于后续的程序设计。以检索菜肴为例。

1 食客向前台管理系统发送检索请求

2 前台管理系统提供可供填写检索信息的表格

3 用户填好后提交给系统

4 系统根据菜单列表进行检索，返回符合要求的菜单列表

5 用户点击感兴趣的菜肴

6 系统调用菜肴的显示函数

7 菜肴显示自身给用户

如有需要也可以给其他的操作或者用况建立顺序图，帮助分析对象间交互情况，由于本系统比较简单，并不用借助顺序图也可以想清楚交互方式，在此并不太有用，就不再多画了。

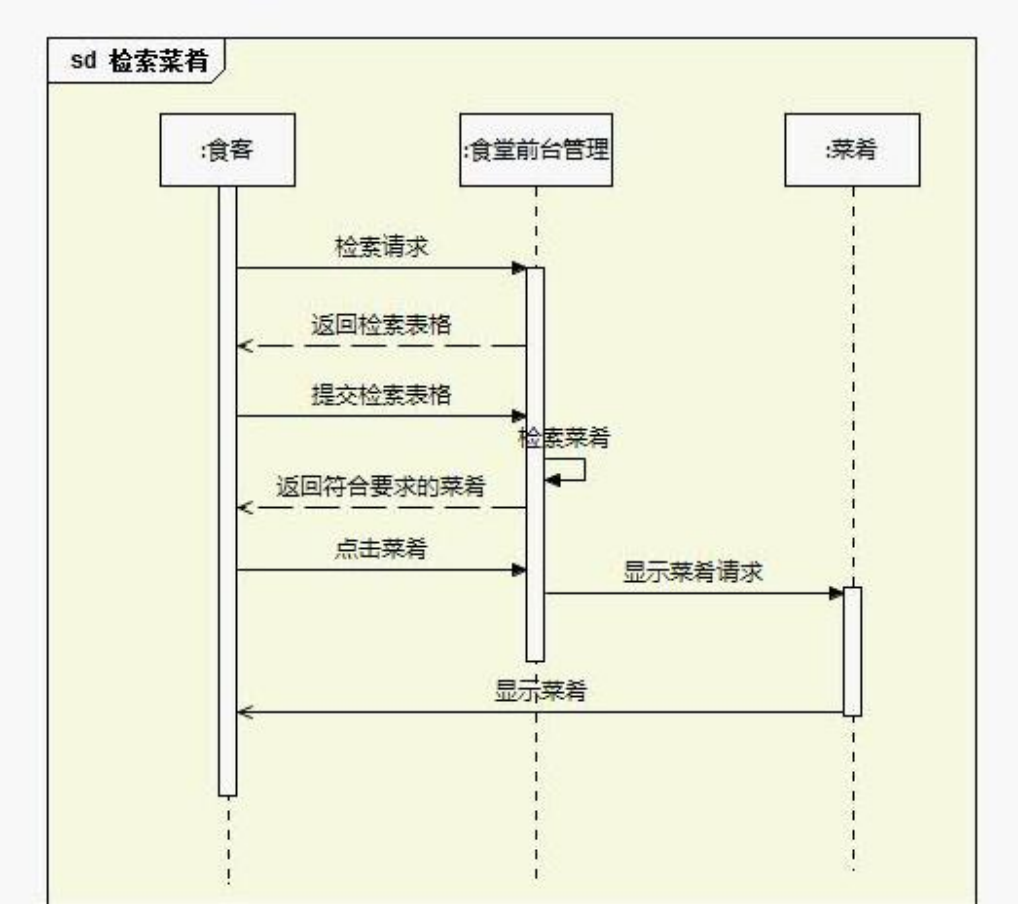


图 3 顺序图-1

2 活动图

活动图非常适合用来描述一个操作的执行过程，比如管理员类中的修改密码操作，就可以用下图表示。

首先需要输入原密码，正确后要求输入两次新密码，然后修改成功。这个过程中若出现异常，则回退到上一步。具体步骤如下图所示。

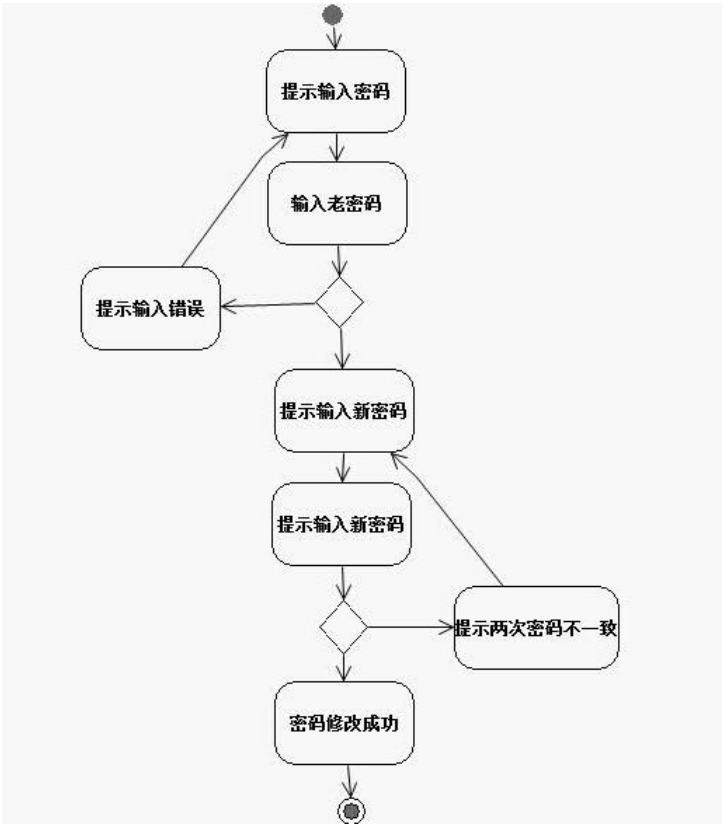
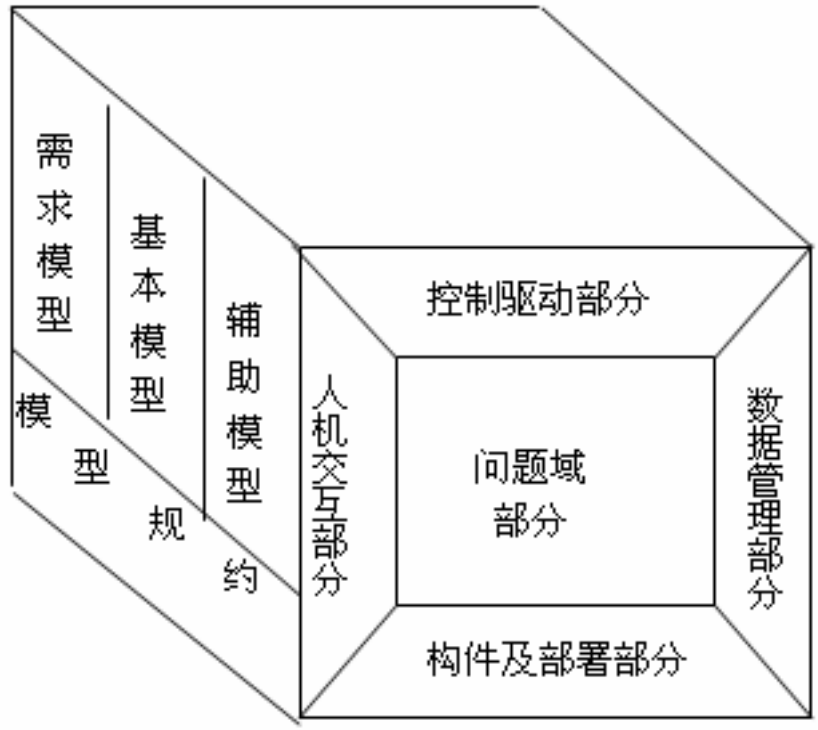


图 4 活动图

PART4 面向对象的设计（OOD 部分）



本食堂餐饮系统在实现方面，使用 Windows 操作系统，用 Html，PHP，Javascript 进行网络编程，用 MySQL 数据库存储数据。

整个系统，数据库和服务器运行在一台主机器上，客户可以通过自己的机器用 IP 访问的方式参与系统。另外，客户访问服务器引起的并发操作，由数据库本身的并发机制进行了合理的控制，因此不需要对本系统的控制流部分再进行建模。

step1 问题域部分设计

从复用类，提高性能，增加一般类以建立共同协议等方面进行考虑，由于之前没有过类似系统，不存在复用类问题。同时 OOA 阶段的分析我们也适当地考虑到了后面的实现，所以这些方面改动不大。倒是 PHP 语言没有继承机制，所以我们把类的继承关系调整成了聚合关系。(所给 PKUmodeler 软件没有聚合关系的显示，因此我们将类图调整为如下）

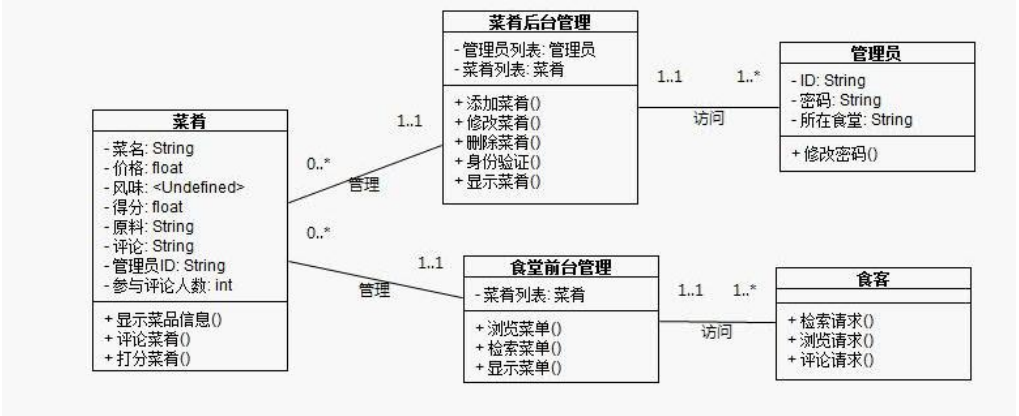


图 5 调整后的类图

根据 OOD 阶段的结合实现界面的特点，我们需要增加与数据库相关的借口。PHP 语言为我们提供了良好的数据库 API。我们需要做的是在为系统创建几个合理的 API 函数，用于对数据库数据的操作。具体来说，有如下函数涉及到对数据的访问和修改：

类	操作
食客	浏览菜肴
	检索菜肴
管理员	添加菜肴
	修改菜肴
	删除菜肴

step2 人机交互部分设计

使用餐饮系统的人群主要集中在学生以及教职工及教职工家属群体，界面设计应尽量做到简单大方明了。

主页界面展示如下：

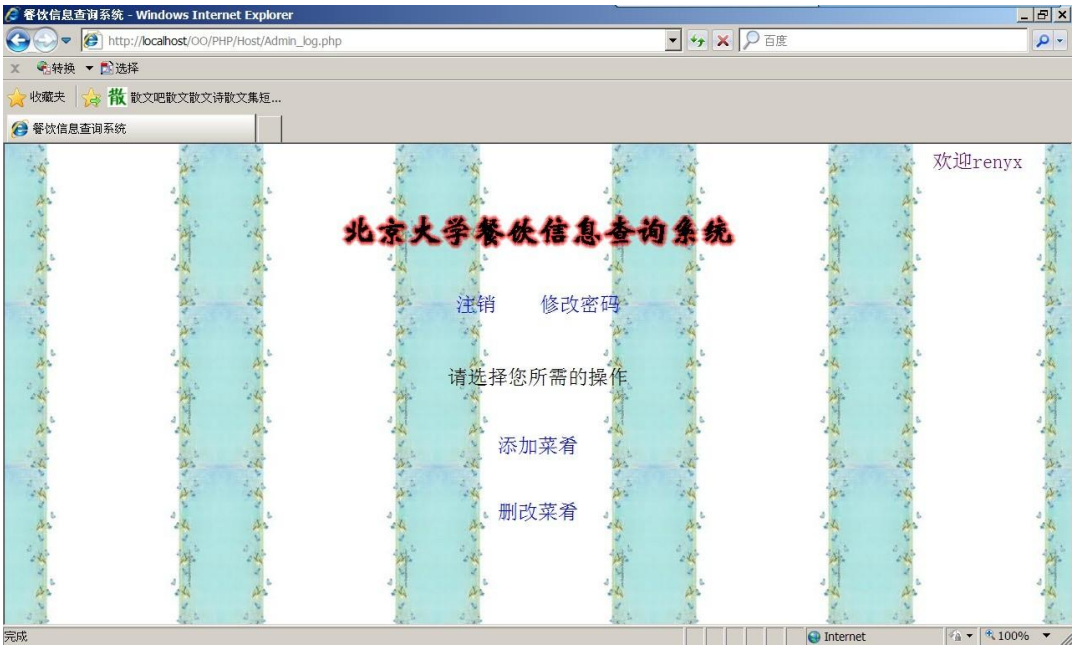
针对不同的用户（食客和管理员），界面显示方式和界面的内容会有差异，具体来说：

管理员：

1- 登陆页面

当输入信息错误时，弹出对话框：

2- 登录成功后的功能选择页面



3- 删改菜肴页面

4- 修改密码页面

食客：

1- 浏览菜肴页面

2- 查询菜肴页面

食客可填写下列表单，进行菜品选择，表单的填写可以是不完整的。

3- 菜肴信息&评论菜肴页面

当用户点击某个菜肴时，系统显示该菜肴的全部基本信息以及评论，并且在此页面下有对该菜肴添加评论信息的接口。

我们把每个页面看成系统当前的一个状态，通过状态机图表示状态的迁移。状态机图如下：

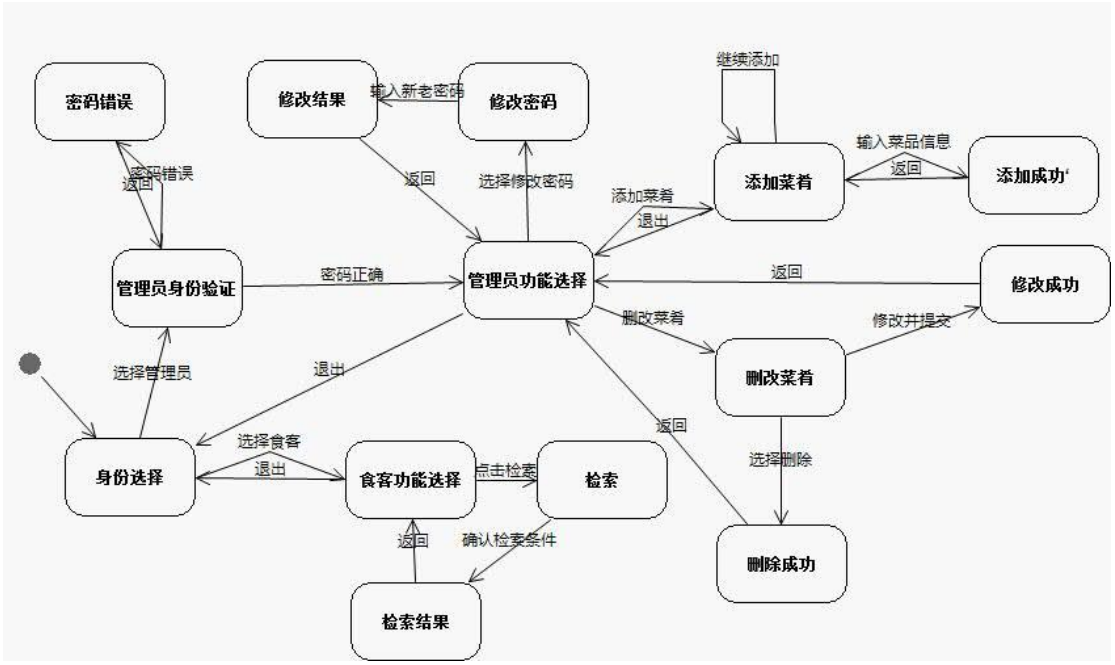


图 6 状态机图

而且通过状态机图状态的跳转，我们也可以很简单清楚的理解页面之间的跳转关系。

step3 数据管理部分设计

这里，整个系统存储的主要是菜肴的项。因此，整个数据存储结构也比较简单。下面是针对永久数据存储的数据表结构。

具体的菜肴表结构如下：

字段	类型	解释
菜名	char[]	
原料	char[]	原料用空格隔开
风味	int	不同值代表不同味道
菜系	int	不同值代表不同菜系
价格	double	
得分	double	
评论	char[]	一个指针，指向评论表
编号	Int	默认递增

主关键字：编号

菜评表的结构如下：

字段	类型	解释
评论	Text	对菜肴的评价

step4 控制流部分的设计

控制驱动部分，识别出主动对象，即用户类创建的对象和管理员对象，下面给出检索部分的顺序图。其余部分的设计与之类似。

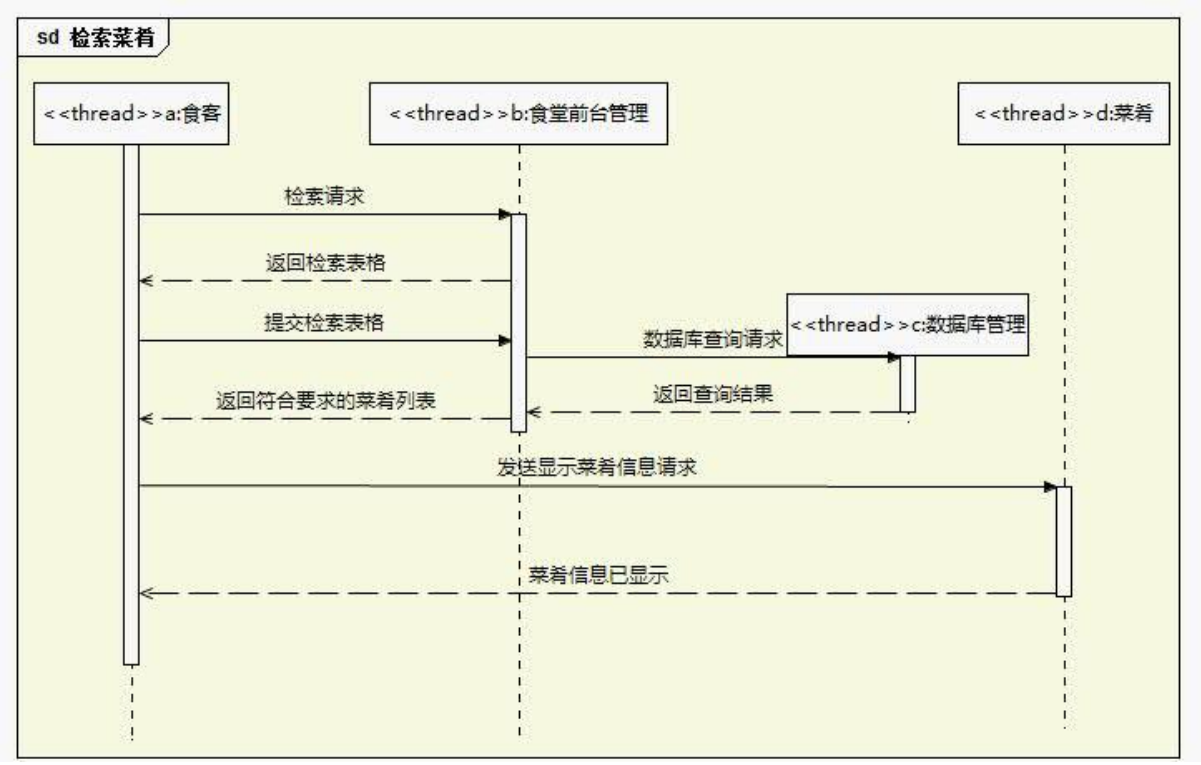


图 7 顺序图-2

说明：

1 食客向前台管理系统发送检索请求

2 活跃的前台管理系统进入执行规约状态，提供可供填写检索信息的表格

3 用户填好后提交给系统

4 系统创建一个数据库查询线程

5 数据库返回查询结果

6 前台系统根据结果制成可选菜列表，反馈给用户

7 用户浏览并点击感兴趣的菜肴

7 菜肴调用显示函数将自身信息反馈给用户

PART4 总结

本次实习中，我们小组完整的实现了网页状态下对数据库的访问，整个系统可以正常运行。在上课的时候给全班同学和老师进行了演示，并受到了老师的高度评价。

面向对象的方法在本次实习中得到了较好的运用，我们完整的经历了 OOA 和 OOD 两个阶段，实践了书上课上提到的大部分内容，对面向对象的设计方法有了更深的认识和理解。

但由于系统比较简单，我们认为面向对象设计方法的优势和作用还没有完全展现出来，在实践中，很多时候我们还是直接去想整个过程的。非常类似于早期的功能分解法，这可能是因为我们的系统比较简单，与使用功能分解法时期的程序规模相近，所以使用那种方法比较自然而然。

当然我们还是尽量强调要使用面向对象的设计思想，但是感觉有点儿高射炮打蚊子杀鸡用牛刀的感觉，我们认为，面向对象的设计方法更适合开发大规模的系统，这时的收益大于成本，而当设计小的应用时，完整的应用面向对象设计方法，可能不太有必要，而且成本较高。

另外，由于大量时间用在了具体代码的实现上，我们的文档可能稍显简略，还请老师和助教学长学姐多多理解。