# 项目介绍

为餐饮企业定制的软件，包括系统管理后台和移动端应用，其中系统管理后台主要提供内部员工使用，移动端应用消费者使用。

分3期进行开发

第一期主要实现基本需求，其中移动端应用通过H5实现。用户可以通过手机浏览器访问

第二期主要针对移动端应用进行改造，使用微信小程序实现，用户使用起来更加方便

第三期主要针对系统进行优化改进，提高系统的访问性能。

# 产品原型展示

# 技术选型

用户层 H5 VUE.js ElementUI 微信小程序

网关层 Nginx

应用层 Spring Boot Spring MVC Spring Session Spring Swagger lombok

数据层 Mysql Mybatis Mybatis Plus Redis

工具 Git maven junit

# 功能架构

移动端前台 手机号登录 微信登录 地址管理 历史订单

菜品规格 购物车 下单 菜品浏览

系统管理后台 分类管理 菜品管理 套餐管理 菜品口味管理

员工登录 员工退出 员工管理 订单管理

# 角色介绍

后台系统管理员：登录后台管理系统，拥有后台系统中的所有操作权限

后台系统普通员工：登录后台管理系统，对菜品、套餐、订单等进行管理

C端应用：登录移动端应用，可以浏览菜品、添加购物车、设置地址

# 开发环境搭建

#### 数据库环境搭建

|  |  |  |
| --- | --- | --- |
| **序号** | **表名** | **说明** |
| 1 | Employee | 员工表 |
| 2 | Categoryt | 菜品和套餐分类表 |
| 3 | Dish | 菜品表 |
| 4 | Setmeal | 套餐表 |
| 5 | Setmeal\_dish | 套餐菜品关系表 |
| 6 | Dish\_flavor | 菜品口味关系表 |
| 7 | User | 用户表（C端） |
| 8 | Address\_book | 地址表 |
| 9 | Shopping\_cart | 购物车表 |
| 10 | Orders | 订单表 |
| 11 | Order\_detail | 订单明细表 |

#### Maven项目搭建

1. 创建maven项目

创建完项目后，注意检查项目的编码、maven仓库配置、jdk配置等

1. 导入pom.xml文件
2. 导入Spring Boot配置文件appliaction.yml
3. 编写启动类

# 7、开发功能

#### 7.1后台登录功能开发

##### 需求分析

1）查看登录请求信息

通过浏览器调试工具，可以发现点击登录按钮时，页面会发送请求（请求地址为）

[（http://localhost:63342/employee/login）并提交参数username和password](http://localhost:63342/employee/login并提交参数username和password)

此时报404，因为后台系统还没有响应此请求的处理器，所以需要创建相关类来处理登录请求

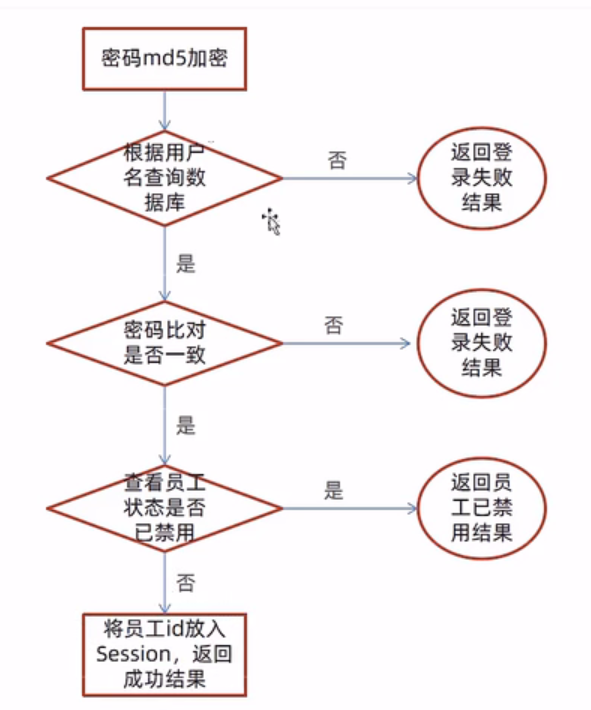
2）数据模型（employee表）

##### 代码开发

1. 创建实体类Employee，和employee表进行映射
2. 创建Controller、Service、Mapper
3. 导入返回结果类R对象
4. 在Controller中创建登录方法

处理逻辑如下

1. 将页面提交的密码password进行md5加密处理
2. 根据页面提交的用户名username查询数据库、
3. 如果没有查询到则返回登录失败结果
4. 密码比对，如果不一致则返回登录失败结果
5. 查看员工状态，如果已禁用状态，则返回员工已禁用结果
6. 登录失败，将员工id存入Session并返回登录成功结果



##### 功能测试

#### 7.2后台退出功能

##### 需求分析

1. 员工登录成功后，页面跳转到后台系统首页面（backend/index.html）,此时会显示当前登录用户的姓名，如果员工需要退出系统，直接点击退出按钮即可退出系统，退出系统后页面应跳转会登录页面，处理逻辑

##### 代码开发

用户点击退出按钮，发送请求，请求地址为/employee/logout,请求方式为POST。

我们只需要在Controller中创建对应的处理方法即可，具体的处理逻辑

1. 清理Session中的用户id
2. 返回结果

##### 功能测试

#### 7.3完善登录功能

##### 问题分析

前面我们已经完成了后台系统的员工登录功能开发，但是还存在一个问题，用户如果不登录，直接访问系统首页面，也可以正常访问

但是这种设计不合理，我们希望只有登录成功后才可以访问系统中的页面，如果没有登录则跳转到登录页面

实现方法：

使用过滤器（web组件）或者拦截器（Spring MVC提供），在过滤器或者拦截器中判断用户是否已经完成登录，如果没有则跳转到登录页面

过滤器实现步骤：

1. 创建自定义过滤器LoginCheckFilter
2. 在启动类上加入注解@ServletComponentScan
3. 完善过滤器的处理逻辑

##### 代码实现

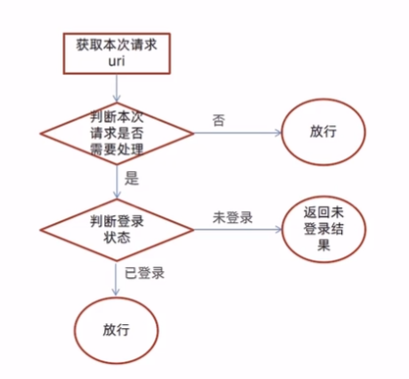
过滤器实现的具体的处理逻辑如下：  
1、获取本次请求的URL

2、判断本次请求是否需要处理

3、如果不需要处理，则直接放行

4、判断登录状态，如果已登录，则直接放行

5、如果未登录则返回未登录结果



#### 7.4新增员工

##### 需求分析

后台系统中可以管理员工信息，通过新增员工来添加后台系统用户，点击【添加员工】按钮跳转到新增页面

##### 数据模型

新增员工，将新增页面录入的员工数据插入到employee表，需要注意，employee表中对username字段加入了唯一约束，username是员工的登录账号

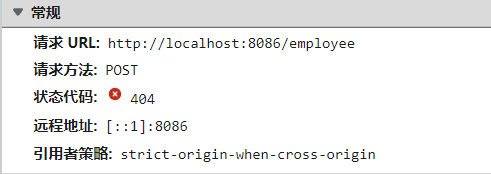
##### 代码开发

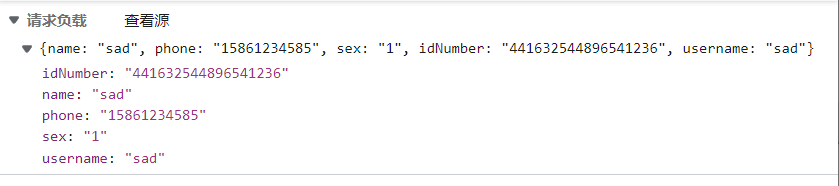
在开发代码之前，需要梳理以下整个程序的执行过程、

**Ajax：A**synchronous **J**avascript **A**nd **X**ML（异步JavaScript和[XML](https://baike.baidu.com/item/XML/86251?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/ajax/_blank)）

Ajax 不是一种新的编程语言，而是一种用于创建更好更快以及交互性更强的Web应用程序的技术。

1. 页面发送Ajax请求，将新增员工页面中输入的数据以json的形式提交到服务器
2. 服务端Controller接收页面提交的数据并调用Service将数据进行保存
3. Service调用Mapper操作数据库，保存数据



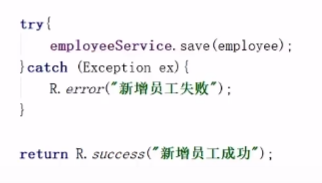


前面的程序还存在一个问题，就是当我们在新增员工时输入的账号已经存在，由于employee表中对该字段加入了唯一约束，，此时程序会抛出异常

Duplicate entry 'zhangsan' for key 'idx\_username'

此时需要我们的程序进行异常捕捉，通常有两种处理方式

1. 在Controller方法中加入try、catch进行异常捕捉
2. 使用异常处理器进行全局异常捕捉



@ControllerAdvice(annotations = {RestController.class, Controller.class})

/\*\*  
 \* 异常处理方法，Controller抛出SQLIntegrityConstraintViolationException该异常  
 \* 会被拦截到  
 \* @return  
 \*/  
@ExceptionHandler(SQLIntegrityConstraintViolationException.class)

这个类是为那些声明了（@ExceptionHandler、@InitBinder 或 @ModelAttribute注解修饰的）方法的类而提供的专业化的@Component , 以供多个 Controller类所共享。

说白了，就是aop思想的一种实现，你告诉我需要拦截规则，我帮你把他们拦下来，具体你想做更细致的拦截筛选和拦截之后的处理，你自己通过@ExceptionHandler、@InitBinder 或 @ModelAttribute这三个注解以及被其注解的方法来自定义。

总结：

1. 根据产品原型明确业务需求
2. 重点分析数据的流转过程和数据格式
3. 通过debug断点调试跟踪程序执行过程

流转数据格式为JSON格式

Request URL：<http://localhost:8080/employee>

Request Method:Post

Request Payload:

JSON格式数据

发送请求

前端页面------------------------------------》服务端（Controller）

#### 7.5员工信息分页查询

需求分析

分页展示数据，使方便查看

代码开发

程序的执行过程

1. 页面 发送AJax请求，将分页查询参数（page、pageSized、name）提交到服务端
2. 服务端Controller接收页面提交的数据并调用Service查询数据
3. Service调用Mapper操作数据库、查询分页数据
4. Controller将查询到的分页数据响应给页面
5. 页面接收到分页数据并通过ElementUI的Table组件展示到页面上

#### 7.6启用/禁用员工账号

##### 需求分析

在员工管理页面，可以对某个员工账号进行启用或者禁用操作，只有管理员才能进行操作，其他普通员工禁用按钮不显示

##### 代码开发

执行过程：

1. 页面发送Ajax请求，将参数（id，status）提交到服务器
2. 服务端Controller接收页面提交的数据并调用Service更新数据
3. Service调用Mapper操作数据库

##### 功能测试

测试过程中没有报错，但是功能并没有实现，查看数据库中的数据也没有变化

观察控制台输出的SQL：

==> Preparing: UPDATE employee SET status=?, update\_time=?, update\_user=? WHERE id=?

==> Parameters: 0(Integer), 2023-01-08T18:01:45.102(LocalDateTime), 1(Long), 1611715130160664600(Long)

<== Updates: 0

SQL执行的结果更新的数据行数为0，仔细观察前端传到服务器id的值和数据库中对应记录的id值并不相同，这是什么原因？

分页查询时服务端相应给页面的数据中id的值为19位数字，类型为long

页面中js处理long型只能精确到前16位，所以最终通过ajax请求提交给服务端的时候id变为了1611715130160664600

##### 代码修复

通过观察控制台输出的SQL发现页面传递过来的员工id的值和数据课中的id值不一致，是由于js对long型数据进行处理时丢失精度，导致提交的id和数据库id不一致

解决办法：

可以在服务端给页面相应json数据时进行处理，将long型数据统一转为String字符串，效果如下

具体实现步骤：

1. 提供对象转化器JacksonObjectMapper，基于Jackson窘相java对象到json数据的转换
2. 在WebMvcCinfig配置类中扩展Sprung mvc的消息转换器，在此消息转换器中使用提供的对象转换器进行java对象到json数据的转换



#### 7.7编辑员工信息

##### 需求分析

在员工管理页面点击编辑信息，跳转到编辑页面，在编辑页面回显员工信息并进行修改

##### 代码开发

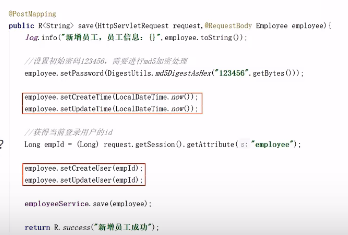
操作过程和执行流程

1. 点击编辑按钮，页面跳转到add.html,并在url中携带参数（员工id）
2. 在add.html页面获取url中的参数（员工id）
3. 发送ajax请求，请求服务端，同时提交员工id参数
4. 服务端接收请求，根据员工id查询员工信息，将员工信息以json形式相应给页面
5. 页面接收服务端相应的json数据，通过VUE的数据绑定进行员工信息回显
6. 点击保存按钮，发送ajax请求，将页面中的员工信息以json方式提交给服务端
7. 服务端接收员工信息，并进行处理，完成后给页面响应
8. 页面接收到服务端响应信息后进行响应处理

# 8分类管理业务开发

#### 8.1公共字段自动填充

###### 问题分析

前面已经完成了后台系统的员工管理功能开发，在新增员工时需要设置创建时间、创建人、修改时间、修改人等字段，在编辑员工时需要设置修改时间和修改人等字段。这些字段属于公共字段也就是很多表中都有这些字段如下



能不能对于这些公共字段在某个地方同意处理，来简化开发呢？

答案是使用Mybatis-plus提供的公共字段自动填充功能

###### 代码实现

Mybatis-Plus公共字段自动填充，也就是在插入或者更新的时候为指定字段赋予指定的值，使用它的好处就是可以同意对这些字段进行处理，避免了重复代码

实现步骤：

1. 在实体类的属性上加入@TableField注解，指定自动填充的策略



1. 按照框架要求填写元数据对象处理器，在此类中统一为公共字段赋值，此类需要实现MetaObjectHandler接口



注意：当前我们设置createUser和updateUser为固定值，后面我们需要进行改进，改为动态获得当前登录用户的id

###### 功能测试

###### 功能完善

前面我们已经完成了公共字段自动填充功能的代码开发，但是还有一个问题没有解决，就是我们在自动填充createUser和updateUser时设置的用户id是固定值，现在我们需要改造成动态获取当前登录用户的id。

有的同学可能想到，用户登录成功后我们将用户id存入了HttpSession中，现在我从HttpSession中获取不就行了？

注意，我们在MyMetaObjectHandler类中是不能获得HttpSession对象的，所以我们需要通过其他方式来获取用户id。

可以使用ThreadLocal来解决此问题，它是JDK中提供的一个类

功能完善

在学习ThreadLocal之前，我们需要先确认一个事情。就是客户端发送的每次http请求，对应的在服务端都会分配一个新的线程来处理，在处理过程中涉及到下面类中的方法都属于相同的一个线程：

1. LoginCheckFilter的doFilter方法
2. EmployeeController的update方法
3. MyMetaObjectHandler的updateFill方法

可以在上面的三个方法中分别加入下面代码（获取当前线程id）：

long id=Thread.currentThread().getId();

Log.info(“线程id:{}”,id}

什么是ThreadLocal’？

ThreadLoca并不是一个Thread，而是Thread的局部变量。当使用ThreadLocal维护变量时，ThreadLocal为每个使用该变量的线程提供独立的变量副本，所以每一个线程都可以独立地改变自己的副本，而不会影响其它线程所对应的副本。ThreadLocal为每个线程提供单独一份存储空间，具有线程隔离的效果，只有在线程内才能获取到对应的值，线程外则不能访问。

ThreadLocal常用方法：

Public void set(T value) 设置当前线程的线程局部变量的值

Public T get() 返回当前线程所对应的线程局部变量的值

我们可以在LoginCheckFilter的doFilter方法中获取当前登录用户id，并调用ThreadLocal的set方法来设置当前线程的线程局部变量的值（用户id），然后在MyMetaObjectHandler的updateFill方法中调用ThreadLocal的get方法来获取当前线程所对应的线程局部变量的值（用户id）

实现步骤

1. 编写BaseContext工具类，基于ThreadLocal封装的工具类
2. 在LoginCheckFilter的doFilter方法中调用BaseCopntext来设置当前登录用户的id
3. 在MyMetaObjectHandler的方法中调用BaseContext获取登录用户的id

#### 8.2新增分类

###### 需求分析

后台系统中可以管理分类信息，分类包括两种类型，分别是菜品分类和套餐分类，当我们在后台系统中添加菜品时需要选择一个菜品分类，当我们在后台系统中添加一个套餐时需要选择一个套餐分类，在移动端也会按照菜品分类和套餐分类来展示对应的菜品和套餐

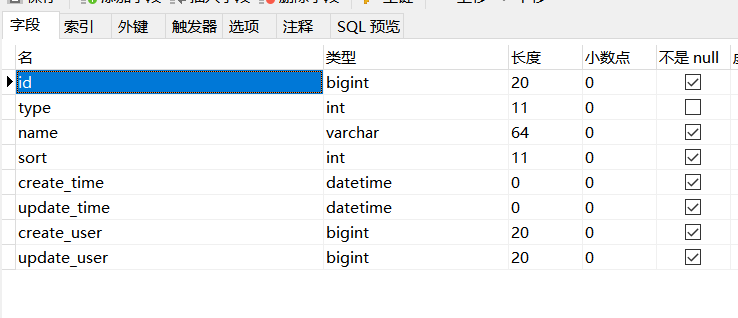
后台系统中可以管理分类信息，分类包括两种类型，分别是菜品分类和套餐分类。当我们在

后台系统中添加菜品时需要选择一个菜品分类，当我们在后台系统中添加一个套餐时需要选择一个套餐分类，在移动端也会按照菜品和套餐分类来展示对应的菜品和套餐

###### 

###### 数据模型

新增分类，其实就是将我们新增窗口录入的分类数据插入到category表，表结构如下



需要注意category表中对name字段加入了唯一约束，保证分类的名称是唯一的

###### 代码开发

在开发业务功能前，先将需要用到的类和接口基本结构创建好

实体类Category

Mapper接口CategoryMapper

业务层接口CategoryService

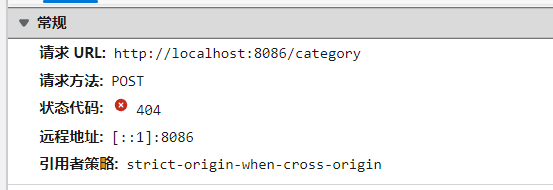
业务层实现类CategoryServiceImpl

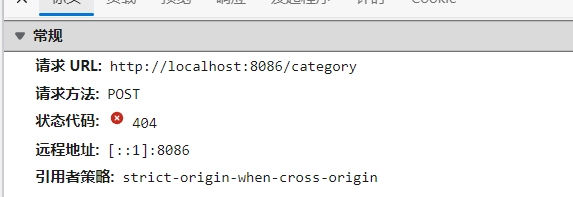
控制层CategoryController

整个程序执行过程：

1. 页面（backend/page/category/list.html)发送ajax请求，将新增分类窗口输入的数据以json形式提交到服务端
2. 服务端Controller接收页面提交的数据并调用Service将数据进行保存
3. Service调用Mapper操作数据库，保存数据

可以看到新增菜品分类和新增套餐分类请求的服务端地址和提交的json数据结构相同，所以服务端只需要提供一个方法统一处理即可





###### 功能测试

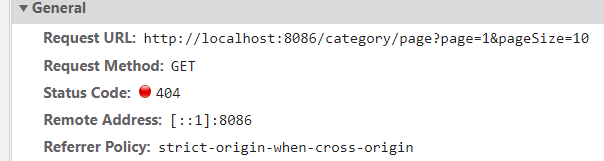
#### 8.3分类信息分页查询

###### 需求分析

系统中分类很多的时候，如果在一个页面中全部展示出来会显得比较乱，不便于查看，所以系统会以分页的方式来展示列表数据

###### 代码开发

整个程序的执行过程：

1. 页面发送ajax请求，将分页查询参数（page，pageSize）提交到服务端
2. 服务端Controller接收页面提交的数据并调用Service查询数据
3. Service调用Mapper操作数据库，查询分页数据
4. Controller将查询到的分页数据响应给页面
5. 页面接收到分页数据并通过ElementUI的Table组件展示到页面上
6. 

###### 功能测试

#### 8.4删除分类

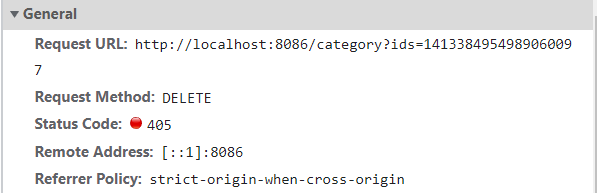
###### 需求分析

在分类管理列表页面，可以对某个分类进行删除操作。需要注意的是当分类关联了菜品或者套餐时，此分类不允许删除

###### 代码开发

整个程序的执行过程：

1. 页面发送ajax请求，将参数（id）提交到服务端
2. 服务端Controller接收页面提交的数据并调用Service删除数据
3. Service调用Mapper操作数据库



###### 功能测试

###### 功能完善

前面我们已经实现了根据id删除分类的功能，但是并没有检查删除的分类是否关联了菜品或者套装，所以我们需要进行功能完善

要完善分类删除功能，需要先准备基础的类和接口

1. 实体类Dish和Setmeal
2. Mapper接口DishMapper和SermealMapper
3. Service接口DishService和SetmealService
4. Service实现类DishServiceImpl和SetmealServiceImpl

#### 8.5修改分类

###### 需求分析

在分类管理列表页面点击修改按钮，弹出窗口，在修改窗口回显分类信息并进行修改，最后点击确认按钮完成修改操作

###### 代码开发

# Day4

#### 9.1文件上传下载

文件上传介绍

文件上传，也称upload,是指将本地图片、视频、音频等文件上传到服务器上，可以供其他用户浏览或下载的过程。文件上传在项目中应用非常广泛，我们经常发微博、发微信朋友圈都用到了文件上传功能

文件上传时，对页面的form表单有如下要求

method=”post” 采用post方式提交数据

enctype=”multipart/form-data” 采用multipart格式上传文件

type=”file” 使用input的file控件上传

举例



目前一些前端组件库也提供了相应的上传组件，但是底层原理还是基于form表单的文件上传。

例如ElementUI中提供的upload上传组件

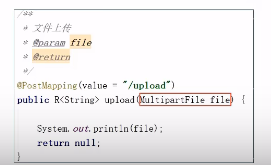


服务端要接收客户端页面上传的文件，通常都会使用Apache的两个组件：

Commons-dileupload

Commons-io

Spring框架在spring-web包中对文件上传进行了封装，大大简化了服务端代码，我们只需要在Controller的方法中声明一个MultipartFile类型的参数即可接收上传的文件、例如



文件下载介绍

文件下载，也称download，是指将文件从服务器传输到本地计算机的过程

通过浏览器进行文件下载，通常有两种表现形式

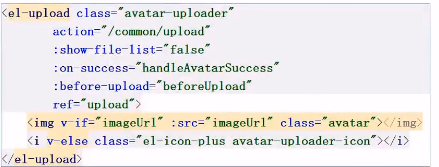
以附件形式下载，弹出保存对话框，将文件保存到指定磁盘目录

直接在浏览器中打开

通过浏览器进行文件下载，本质上就是服务端将文件以流的形式写回浏览器的过程

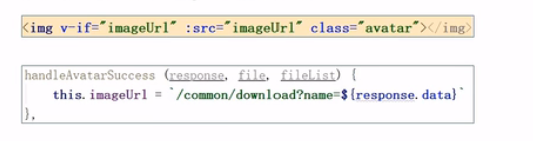
###### 文件上传代码实现

文件上传，页面端可以使用ElementUI提供的上传组件



###### 文件下载代码实现

文件下载，页面端可以实现<img>标签展示下载的图片



#### 9.2新增菜品

###### 需求分析

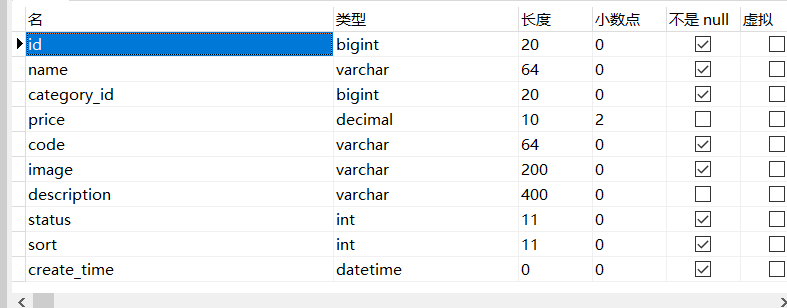
后台系统中可以管理菜品信息，通过新增功能来添加一个新的菜品，在添加菜品时需要选择当前菜品所属的菜品分类，并且需要上传菜品图片，在移动端会按照菜品分类来展示对应的菜品信息

###### 数据模型

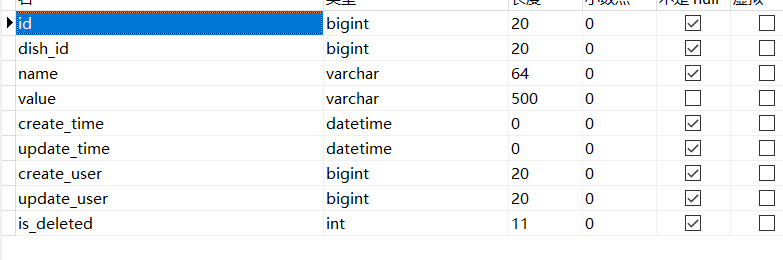
新增菜品，其实就是将新增页面录入的菜品信息插入到dish表，如果添加了口味做法，还需要向dish\_flavor表插入数据，所以在新增菜品时，涉及到两个表

dish菜品表

Dish\_flavor口味表



Dish菜品表



Dish\_flavor口味表

###### 代码开发-准备工作

在开发业务功能时，先将需要用到的类和接口基本结构创建好

实体类DishFlavor

Mapper接口DishFlavorMapper

业务层接口DishFlavorService

业务层实现类DishFlavorServiceImpl

控制层DishController

###### 代码开发-梳理交互过程

新增菜品时前端页面和服务端的交互过程

1. 页面（backend/page/food/add.html）发送ajax请求，请求服务端获取菜品分类数据并展示到下拉框中
2. 页面发送请求进行图片上传，请求服务端将图片保存到服务器
3. 页面发送请求进行图片下载，将上传的图片进行回显
4. 点击保存按钮，发送ajax请求，将菜品相关数据以json形式提交到服务端

开发新增菜品功能，其实就是在服务端编写代码去处理前端页面发送的这4次请求即可

###### 代码开发-导入DTO（DTO全程Data Transform Object，即数据传输对象，一般用于展示层与服务层之间的数据传输）

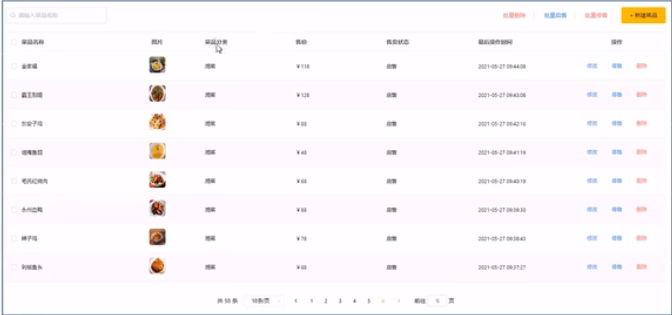
导入DishDto，用于封装页面提交的数据



#### 9.3菜品信息分页查询

###### 需求分析

系统中的菜品数据很多的时候，如果在一个页面中全部展示出来会显得比较乱，不便于查看，所以一般会以分页的方式来展示数据



###### 代码开发-梳理交互过程

进行菜品分页查询时前端页面和服务端的交互过程

1. 页面（backend/page/food/list.html）发送ajax请求，将分页查询参数(page,pageSize,name)提交到服务端，获取分页数据
2. 页面发送请求，请求服务端进行图片下载，用于页面图片展示

开发菜品信息分页查询功能，其实就是在服务端编写代码去处理前端页面发送的这2次请求即可

###### 功能测试

#### 9.4修改菜品

###### 需求分析

在菜品管理列表页面点击修改按钮，跳转到修改菜品页面，在修改页面回显菜品相关信息并进行修改，最后点击确定进行修改

###### 代码开发-梳理交互过程

修改菜品时前端页面（add.html）和服务端的交互过程

1. 页面发送ajax请求，请求服务端获取分类数据，用于菜品分类下拉框中数据展示
2. 页面发送ajax请求，请求服务端，根据id查询当前菜品信息，用于菜品信息回显
3. 页面发送请求，请求服务端进行图片下载，用于页图片回显
4. 点击保存按钮，页面发送ajax请求，将修改后的菜品相关数据以json形式提交到服务端

开发修改菜品功能，其实就是在服务端编写代码去处理前端页面发送的这4次请求即可

###### 功能测试