第5章 产品信息管理模块

本章学习目标

- 了解产品信息管理库表设计。
- 了解产品信息管理模块的功能。
- 掌握产品信息管理服务端的功能实现。
- 掌握 Android App 移动端页面的开发。

产品信息管理模块是整个系统的核心，后续章节中大大小小的功能基本都是围绕产品实现的，产品信息管理模块需要有严格的权限控制系统，不允许登录系统的人员随意改动产品信息，或者添加、删除产品信息。为了使销售人员在查看产品信息时能够更直观地区分不同型号的产品，此系统实现了图片上传的功能，每一件产品都有一张或一组图片与之关联对应。

5.1 产品信息库表设计

5.1.1 设计产品表结构

产品信息表主要用来存储销售人员销售的产品详情，每个产品都有自己的编号、产品名称、销售价格等关键信息。除此之外，每个产品都会匹配一张图片，展示该产品的外观。因此，在数据库中需要新建两张表：一张用来存储产品的详细信息；另一张用来存放产品的照片信息。

根据业务需求，产品信息表中的字段及其含义如下：duiying(对应基准产品)、name(产品名称)、model(产品型号)、specification(产品规格)、category(产品类别)、number(产品编号)、status(产品状态)、origin(产品产地)、brand(产品品牌)、costPrice(成本价格)、unit(产品单位)、weight(产品重量)、retailPrice(零售价格)、managerPrice(经理优惠价)、directorPrice(总监优惠价)、generalManagerPrice(总经理优惠价)、sendProduct(是否需要发货)、maxInventory(库存上限)、productDescription(产品说明)、productNote(产品备注)。产品图片表中的字段信息主要有 pictureName(图片名称)、PictureUrl(图片地址)、committime(上传时间)和 productId(产品信息表 id)。

5.1.2 创建数据表

1. 创建产品信息表

```
create table db_product_information(
id int not null identity(1,1) primary key,
```

```
duiying nvarchar(50),
     -- 对应基准产品
name nvarchar(50),
     -- 产品名称
model varchar(50),
    -- 产品型号
specification  nvarchar(50),
    -- 产品规格
category  nvarchar(50),
     -- 产品类别
number nvarchar(50),
     -- 产品编号
status nvarchar(20),
    -- 产品状态
origin nvarchar(50),
     -- 产品产地
brand nvarchar(20),
    -- 产品品牌
costPrice decimal(12,2),
    -- 成本价格
unit nvarchar(10),
     -- 产品单位
weight nvarchar(20),
     -- 产品重量
retailPrice decimal(12,2),
     -- 零售价格
managerPrice decimal(12,2),
     -- 经理优惠价
directorPrice decimal(12,2),
     -- 总监优惠价
generalManagerPrice decimal(12,2),
     -- 总经理优惠价
sendProduct nvarchar(10),
     -- 是否需要发货
maxInventory nvarchar(10),
     -- 库存上限
productDescription  nvarchar(200),
     -- 产品说明
productNote nvarchar(200)
     -- 产品备注
)
```

2. 创建产品图片表

```
create table db_productInformation_picture(
id int not null identity(1,1)
 primary key,
pictureName varchar(200),
    -- 图片名称
PictureUrl varchar(200),
```

```
    -- 图片地址
committime datetime2,
    -- 上传时间
productId int
    -- 产品信息表 id
)
```

5.2 产品信息服务端接口

5.2.1 产品信息增、删、改、查接口

修改 SqlserverGenerator 类中指定生成的 Bean 的数据库表名，运行 main()方法，自动生成相关代码文件。

1. 编辑 mapper.xml 文件

在通过 MyBatis-Plus 插件自动生成的 DbProductInformationmapper 文件中编辑新增产品信息，删除、查询并修改产品信息的 SQL 语句，代码如下所示。

```
<!-- 新增产品信息 -->
<insert id="addProductInformation"
parameterType="com.hongming.demo.entity.DbProductInformation">
    <selectKey resultType="java.lang.Integer" keyProperty="id"
order="AFTER">
        SELECT @@IDENTITY
    </selectKey>
    <![CDATA[
    insert into db_product_information(
        duiying,name,model,specification,category,number,status,
        origin,brand,costPrice,unit,weight,retailPrice,
managerPrice,directorPrice,generalManagerPrice,sendProduct,
maxInventory,productDescription,productNote)
values(          #{duiying},#{name},#{model},#{specification},#{category},
#{number},#{status},
#{origin},#{brand},#{costPrice},#{unit},#{weight},#{retailPrice},

#{managerPrice},#{directorPrice},#{generalManagerPrice},#{sendProduct},#{
maxInventory},#{productDescription},          #{productNote})            ]]>
  </insert>
<!-- 删除产品信息 -->
<delete id="deleteProductInformation" parameterType="Integer">
    DELETE FROM db_product_information WHERE id = #{id}
</delete>
<!-- 修改产品信息 -->
<update id="updateProductInformation">
    update db_product_information  set
duiying=#{duiying},name=#{name},model=#{model},specification=#{specificat
ion},category=#{category},number=#{number},status=#{status},
```

```
origin = #{origin},brand = #{brand},costPrice = #{costPrice,jdbcType = DECIMAL},u
nit = #{unit},weight = #{weight},retailPrice = #{retailPrice,jdbcType = DECIMAL},
managerPrice = #{managerPrice,jdbcType = DECIMAL},directorPrice = #{directorPri
ce,jdbcType = DECIMAL},generalManagerPrice = #{generalManagerPrice,jdbcType = D
ECIMAL},sendProduct = #{sendProduct},maxInventory = #{maxInventory},productDe
scription = #{productDescription},productNote = #{productNote}  where
id = #{id}</update>
<!-- 分页查询产品信息 -->
<select id = "queryProductInformation" resultMap = "BaseResultMap"
parameterType = "com.hongming.demo.entity.DbProductInformation">
    select top 20 id,
duiying,name,model,specification,category,number,status,
origin,brand,costPrice,unit,weight,retailPrice,
managerPrice,directorPrice,generalManagerPrice,sendProduct,
maxInventory,productDescription,productNote    from (select ROW_NUMBER()
OVER(order by id DESC) AS rownumber, * FROM db_product_information) AS T
where T.rownumber BETWEEN (#{page} - 1) * 20 + 1 and #{page} * 20 + 1    order by
id DESC</select>
<!-- 根据产品名称关键字模糊查询所有产品信息 -->
<select id = "queryAllProductinformationByName" resultMap = "BaseResultMap"
parameterType = "com.hongming.demo.entity.DbProductInformation">
SELECT top 8 id,duiying,name,model,specification,category,number,status,
origin,brand,costPrice,unit,weight,retailPrice,
managerPrice,directorPrice,generalManagerPrice,sendProduct,
maxInventory,productDescription,productNote    FROM
db_product_information    WHERE name like CONCAT('%',#{name},'%')
</select>
<!-- 根据产品 id 查询产品信息 -->
<select id = "queryPriceById" resultMap = "BaseResultMap"
parameterType = "com.hongming.demo.entity.DbProductInformation">
   SELECT id,duiying,name,model,specification,category,number,status,
        origin,brand,costPrice,unit,weight,retailPrice,
managerPrice,directorPrice,generalManagerPrice,sendProduct,
maxInventory,productDescription,productNote
FROM db_product_information    WHERE id = #{id}
</select>
```

在以上代码中，新增一条数据时，会返回一个 int 类型的 id 值，该值作为关联图片表的重要数据，每当增加产品信息需要附带一张产品的照片时，可以将该 id 作为一个字段，存储在图片数据表中，这样以后想要查询该产品图片时，只需通过这个产品的 id 查询即可。在 DbProductinformationPicturemapper.xml 文件中添加增加图片、删除图片、查询图片的 SQL 语句，代码如下所示。

```
<!-- 根据产品信息表的 id 增加图片 -->
<insert id = "addProductionPicture">
    insert into
db_productInformation_picture(pictureName,PictureUrl,committime,productId
)      values(#{pictureName},#{PictureUrl},#{committime},#{productId})
```

```
</insert>
<!-- 根据产品信息表的 id 删除图片 -->
<delete id="deleteProductionPicture" parameterType="Integer">
    DELETE FROM db_productInformation_picture WHERE id = #{id}
</delete>
<!-- 根据产品信息表的 id 查询图片 -->
<select id="queryProductionPicture" resultMap="BaseResultMap"
parameterType="com.hongming.demo.entity.DbProductinformationPicture">
    select id,pictureName,PictureUrl,convert(nvarchar(100),committime,20)
AS committime,productId    FROM db_productInformation_picture    WHERE
productId = #{productId}    order by id DESC
</select>
```

2. 编辑 Mapper 接口

在通过 MyBatis-Plus 插件自动生成的 DbProductInformationMapper 接口中新增产品信息、删除产品信息、修改产品信息、分页查询所有产品信息、根据关键字查询产品信息和根据产品 id 查询产品信息的接口的 SQL 语句，代码如下所示。

```
@Mapper
public interface DbProductInformationMapper extends
BaseMapper<DbProductInformation> {
    //新增产品信息
    int addProductInformation(DbProductInformation dbProductInformation);
    //删除产品信息
    int deleteProductInformation(@Param("id") Integer id);
    //修改产品信息
    int updateProductInformation(DbProductInformation
dbProductInformation);
    //分页查询所有产品信息
    List<DbProductInformation> queryProductInformation(@Param("page") int
page);
    //根据关键字查询产品信息
    List<DbProductInformation>
queryAllProductinformationByName(@Param("name") String name);
    //根据产品 id 查询产品信息
    DbProductInformation queryPriceById(@Param("id") Integer id);
}
```

在 DbProductinformationPictureMapper 接口中添加新增产品图片、删除产品图片和查询产品图片的接口的 SQL 语句，代码如下所示。

```
@Mapper
public interface DbProductinformationPictureMapper extends
BaseMapper<DbProductinformationPicture> {
    //根据产品信息表的 id 新增产品图片
    int addProductionPicture(
            @Param("pictureName") String pictureName,
```

```
            @Param("PictureUrl") String PictureUrl,
            @Param("committime") String committime,
            @Param("productId") Integer xiaoshoujihuiId);
    //根据 id 删除产品图片
    int deleteProductionPicture(@Param("id") Integer id);
    //根据产品信息表的 id,查询所有该产品附带的所有图片
    List < DbProductinformationPicture >
queryProductionPicture(@Param("productId") Integer productId);
    //根据图片 id 查询图片的存放路径
    DbProductinformationPicture queryPictureName(@Param("id") Integer
id);
}
```

3. 编辑 Service 接口

在通过 MyBatis-Plus 插件自动生成的 DbProductInformationService 接口中新增产品信息、删除产品信息、修改产品信息、分页查询所有产品信息、根据关键字查询产品信息和根据产品 id 查询产品信息的接口的 SQL 语句,代码如下所示。

```
public interface DbProductInformationService extends IService < DbProductInformation > {
    //新增产品信息
    int addProductInformation(DbProductInformation dbProductInformation);
    //删除产品信息
    int deleteProductInformation(@Param("id") Integer id);
    //修改产品信息
    int updateProductInformation(DbProductInformation dbProductInformation);
    //分页查询所有产品信息
    List < DbProductInformation > queryProductInformation(@Param("page") int page);
    //根据关键字查询产品信息
    List < DbProductInformation > queryAllProductinformationByName(@Param("name") String
name);
    //根据产品 id 查询产品信息
    DbProductInformation queryPriceById(@Param("id") Integer id);
}
```

在 DbProductinformationPictureService 接口中添加新增产品图片、删除产品图片和查询产品图片的接口的 SQL 语句,代码如下所示。

```
public interface DbProductinformationPictureService extends IService
< DbProductinformationPicture > {
    //根据产品信息表的 id 新增产品图片
    int addProductionPicture(
            @Param("pictureName") String pictureName,
            @Param("PictureUrl") String PictureUrl,
            @Param("committime") String committime,
            @Param("productId") Integer xiaoshoujihuiId);
    //根据下属图片的 id 删除产品图片
    int deleteProductionPicture(@Param("id") Integer id);
    //根据产品信息表的 id 查询所有下属图片
```

```
    List<DbProductinformationPicture> queryProductionPicture(@Param("productId")
 Integer productId);
    //根据图片 id 查询图片的存放路径
    DbProductinformationPicture queryPictureName(@Param("id") Integer id);
}
```

4. 编辑 ServiceImpl 实现类

在通过 MyBatis-Plus 插件自动生成的 DbProductInformationServiceImpl 实现类中新增产品信息、删除产品信息、修改产品信息、分页查询所有产品信息、根据关键字查询产品信息和根据产品 id 查询产品信息的接口的 SQL 语句，代码如下所示。

```
@Service
public class DbProductInformationServiceImpl extends
ServiceImpl<DbProductInformationMapper, DbProductInformation> implements
DbProductInformationService {
    @Override
    public int addProductInformation(DbProductInformation
    dbProductInformation) {
        return baseMapper.addProductInformation(dbProductInformation);
    }
    @Override
    public int deleteProductInformation(Integer id) {
        return baseMapper.deleteProductInformation(id);
    }
    @Override
    public int updateProductInformation(DbProductInformation
    dbProductInformation) {
        return baseMapper.updateProductInformation(dbProductInformation);
    }
    @Override
    public List<DbProductInformation> queryProductInformation(int page) {
        return baseMapper.queryProductInformation(page);
    }
    @Override
    public List<DbProductInformation>
    queryAllProductinformationByName(String name) {
        return baseMapper.queryAllProductinformationByName(name);
    }
    @Override
    public DbProductInformation queryPriceById(Integer id) {
        return baseMapper.queryPriceById(id);
    }
}
```

在 DbProductinformationPictureServiceImpl 实现类中添加新增产品图片、删除产品图片和查看产品图片的接口的 SQL 语句，代码如下所示。

```
@Service
public class DbProductinformationPictureServiceImpl extends
```

```
ServiceImpl < DbProductinformationPictureMapper,
DbProductinformationPicture > implements
DbProductinformationPictureService {
    @Override
    public int addProductionPicture(String pictureName, String
    PictureUrl, String committime, Integer xiaoshoujihuiId) {
    return
    baseMapper.addProductionPicture(pictureName,PictureUrl,committime,
    xiaoshoujihuiId);
}
    @Override
    public int deleteProductionPicture(Integer id) {
        return baseMapper.deleteProductionPicture(id);
    }
    @Override
    public List < DbProductinformationPicture >
    queryProductionPicture(Integer productId) {
        return baseMapper.queryProductionPicture(productId);
    }
    @Override
    public DbProductinformationPicture queryPictureName(Integer id) {
        return baseMapper.queryPictureName(id);
    }
}
```

编辑 DbProductInformationController，实现产品的添加、产品信息的修改、产品的删除和产品信息的查看等功能，代码如下所示。

```
@Controller
@RequestMapping("ProductInformation")
public class DbProductInformationController {
    @Autowired
    private DbProductInformationService dbProductInformationService;
    @Autowired
    private DbProductinformationPictureService dbProductinformationPictureService;
    @Autowired
    private DbProductinformationPicture dbProductinformationPicture;
    @Autowired
    private DbProductInformation dbProductInformation;
    public static int ADDCUSTOM = 0;                                    //添加权限
    public static int DELETECUSTOM = 0;                                 //删除权限
    public static int UPDATECUSTOM = 0;                                 //修改权限
    //新增产品信息,有照片
    @PostMapping("addProductInformation")
    @ResponseBody
    @Authorized
    public Response addProductInformation(DbProductInformation
    dbProductInformation, MultipartFile[ ] files, HttpServletRequest request)
    {
```

```
        Response<DbProductInformation> response = new Response<>();
        String token = request.getHeader("token");
        List<RoleAndPower> list = (List<RoleAndPower>) RedisOps.getObject(token);
        System.out.println(RedisOps.getObject(token));
        for (RoleAndPower n: list) {
            if (n.getRole().equals("管理员")) {
                ADDCUSTOM++;
                System.out.println(ADDCUSTOM);
                break;
            }
        }
        if (ADDCUSTOM != 0) {
            ADDCUSTOM = 0;
            int i = dbProductInformationService.
addProductInformation(dbProductInformation);
            int b = 0;
            int productId = dbProductInformation.getId();                    //产品信息表 id
            String LUJIN = "D: \\images\\productInformation\\";
            String URL = "productInformation";
            if (files != null && files.length >= 1) {
                for (MultipartFile file: files) {
                    String NAME = UUID.randomUUID().toString() +
                                  file.getOriginalFilename();
                    String filePath = LUJIN + NAME;
                    String url = "http://192.127.180.88: 8888/web/images/" +
                    URL + "/" + NAME;
                    System.out.println("上传的图片名称: " + NAME);
                    System.out.println("上传的图片路径: " + url);
                    Date date = new Date();
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
                    HH: mm: ss");
                    String committime = sdf.format(date);
                    try {
                        file.transferTo(new File(filePath));
                        b = dbProductinformationPictureService.
addProductionPicture(NAME, url, committime, productId);
                    } catch (IOException e) {
                        e.printStackTrace();
                        response.setCode(-1);
                        response.setMessage("照片提交失败,请重新提交!");
                    }
                }
            }
            if (i > 0) {
                response.setCode(200);
                response.setMessage("产品信息添加成功!");
            } else {
```

```
                response.setCode(-1);
                response.setMessage("产品信息添加失败,请重新提交!");
            }
        } else {
            response.setCode(-1);
            response.setMessage("您没有添加权限,请联系管理员!");
        }
        return response;
    }
    //根据 id 删除产品信息
    @PostMapping("deleteProductInformation")
    @ResponseBody
    @Authorized
    public Response deleteProductInformation(Integer id, HttpServletRequest request) {
        Response<Integer> response = new Response<>();
        String token = request.getHeader("token");
        List<RoleAndPower> list = (List<RoleAndPower>) RedisOps.getObject(token);
        System.out.println(RedisOps.getObject(token));
        for (RoleAndPower n: list) {
            if (n.getRole().equals("管理员")) {
                DELETECUSTOM++;
                System.out.println(DELETECUSTOM);
                break;
            }
        }
        if (DELETECUSTOM != 0) {
            DELETECUSTOM = 0;
            int i = dbProductInformationService.deleteProductInformation(id);
            if (i > 0) {
                response.setCode(200);
                response.setMessage("该条记录删除成功!");
            } else {
                response.setCode(-1);
                response.setMessage("该条记录删除失败!");
            }
        } else {
            response.setCode(-1);
            response.setMessage("您没有删除权限,请联系管理员!");
        }
        return response;
    }
    //根据 id 修改该条产品信息
    @PostMapping("updateProductInformation")
    @ResponseBody
    @Authorized
    public Response updateProductInformation(DbProductInformation
    dbProductInformation, HttpServletRequest request) {
```

```
        Response<Integer> response = new Response<>();
        String token = request.getHeader("token");
        List<RoleAndPower> list = (List<RoleAndPower>) RedisOps.getObject(token);
        System.out.println(RedisOps.getObject(token));
        for (RoleAndPower n: list) {
            if (n.getRole().equals("管理员")) {
                UPDATECUSTOM++;
                System.out.println(UPDATECUSTOM);
                break;
            }
        }
        if (UPDATECUSTOM != 0) {
            UPDATECUSTOM = 0;
            int i = dbProductInformationService.
updateProductInformation(dbProductInformation);
            System.out.println(i);
            if (i > 0) {
                response.setCode(200);
                response.setMessage("产品信息修改成功!");
            } else {
                response.setCode(-1);
                response.setMessage("产品信息修改失败!");
            }
        } else {
            response.setCode(-1);
            response.setMessage("您没有修改权限,请联系管理员!");
        }
        return response;
    }
    //查询所有产品信息
    @PostMapping("queryProductInformation")
    @ResponseBody
    public Response queryProductInformation(int page) {
        Response<List<DbProductInformation>> response = new Response<>();
        List<DbProductInformation> list =
        dbProductInformationService.queryProductInformation(page);
        for (DbProductInformation l: list) {
            if (dbProductinformationPictureService.
            queryProductionPicture(l.getId()) != null &&
            dbProductinformationPictureService.queryProductionPicture(l.getId()).
            size() >= 1) {
                String url = dbProductinformationPictureService.
                queryProductionPicture(l.getId()).get(0).getPictureUrl();
                l.setUrl(url);
            }
        }
        if (list.size() == 0) {
```

```
            response.setCode(-1);
            response.setMessage("没有相关产品");
            return response;
        } else {
            response.setCode(200);
            response.setResult(list);
            return response;
        }
    }
    //根据产品信息表的 id 查询下属图片
    @PostMapping("queryProductionPicture")
    @ResponseBody
    public Response queryProductionPicture(Integer productId) {
        Response<List<DbProductinformationPicture>> response = new Response<>();
        List<DbProductinformationPicture> list =
        dbProductinformationPictureService.queryProductionPicture(productId);
        if (list.size() == 0) {
            response.setCode(-1);
            response.setMessage("未查询到下属附件!");
            return response;
        } else {
            response.setCode(200);
            response.setResult(list);
            return response;
        }
    }
    //根据 id 删除产品信息的下属图片
    @PostMapping("deleteProductionPicture")
    @ResponseBody
    @Authorized
    public Response deleteProductionPicture(Integer id, HttpServletRequest request) {
        Response<Integer> response = new Response<>();
        String token = request.getHeader("token");
        List<RoleAndPower> list = (List<RoleAndPower>) RedisOps.getObject(token);
        System.out.println(RedisOps.getObject(token));
        for (RoleAndPower n: list) {
            if (n.getRole().equals("管理员")) {
                DELETECUSTOM++;
                System.out.println(DELETECUSTOM);
                break;
            }
        }
        if (DELETECUSTOM != 0) {
            DELETECUSTOM = 0;
            dbProductinformationPicture =
            dbProductinformationPictureService.queryPictureName(id);
            String name = dbProductinformationPicture.getPictureName();
```

```
            String url = "D:\\images\\productInformation\\" + name;
            int i = dbProductinformationPictureService.deleteProductionPicture(id);
            File file = new File(url);
            if (file.exists()) {
                file.delete();
            }
            if (i > 0) {
                response.setCode(200);
                response.setMessage("删除成功!");
            } else {
                response.setCode(-1);
                response.setMessage("删除失败!");
            }
        } else {
            response.setCode(-1);
            response.setMessage("您没有删除权限,请联系管理员!");
        }
        return response;
    }
    //查询所有产品信息
    @PostMapping("queryAllProductinformationByName")
    @ResponseBody
    public Response queryAllProductinformationByName(String name) {
        Response<List<DbProductInformation>> response = new Response<>();
        List<DbProductInformation> list =
        dbProductInformationService.queryAllProductinformationByName(name);
        if (list.size() == 0) {
            response.setCode(-1);
            response.setMessage("没有相关产品");
            return response;
        } else {
            response.setCode(200);
            response.setResult(list);
            return response;
        }
    }
    //根据产品 id 查询产品信息  dbProductInformationService
    @PostMapping("queryPriceById")
    @ResponseBody
    public Response queryPriceById(Integer id) {
        Response<DbProductInformation> response = new Response<>();
        dbProductInformation = dbProductInformationService.queryPriceById(id);
        if (dbProductInformation != null) {
            response.setCode(200);
            response.setResult(dbProductInformation);
            return response;
```

```
            } else {
                response.setCode(-1);
                response.setMessage("该产品信息不存在!");
                return response;
            }
    }
    }
```

以上代码中定义了三个静态常量：ADDCUSTOM、DELETECUSTOM 和 UPDATECUSTOM，这三个常量的初始值都是 0，当调用添加、删除、修改产品的接口时，首先通过 HttpServletRequset 获取请求头中携带的 token，并对 token 值做解析，即从 Redis 中获取该 token 键对应的值，从而获取到当前用户的角色。如果该角色拥有新增权限，那么程序在执行新增接口的代码时，会将常量 ADDCUSTOM 值加 1，然后在 insert 操作时会通过 ADDCUSTOM 的值来判断是否可以执行，如果 ADDCUSTOM 为 0 则代表当前用户没有添加产品信息的权限，如果值为 1 则可以进行产品信息的添加。同理，产品信息的修改和产品的删除都是通过此方式来判断当前操作人是否具有操作权限。

此外，需要注意的是，产品表和产品图片表之间是通过产品的 id 关联起来的，图片表中的 productId 就是对应产品在产品表中的 id。

5.2.2 产品信息增、删、改、查接口测试

1. 新增产品接口测试

新增产品信息需要拥有新增的权限，该系统只允许管理员新增产品，因此在测试新增产品接口时，需要登录系统，得到返回的 token 值，然后携带 token 值测试新增接口。当调用新增接口时，系统先通过 token 值判断用户是否为管理员，如果不是，则不能新增。在地址栏中输入“192.168.1.127：8888/web/ProductInformation/addProductInformation? duiying＝HM-580A 自动调模成型机/台 &name＝移盒机械手 &model＝HM-LS2019-11-24-01&specification＝HM-LS2019-11-24-01&category＝盖盒机系列 &number＝CP20200310001&status＝正常 &origin＝中国 &brand＝鸿铭 &costPrice＝12000&unit＝套 &weight＝500kg&retailPrice＝200000&managerPrice＝190000&directorPrice＝180000&generalManagerPrice＝150000&sendProduct＝需要 &maxInventory＝10000 套 &productDescription＝&productNote”，选择 Body→form-data 命令，在 KEY 中输入图片传递的形参，然后在 VALUE 中选择要上传的图片信息，按 Enter 键，如果出现如图 5.1 所示场景则代表产品新增成功。

2. 查看产品信息接口测试

打开 postman，在地址栏中输入“192.168.1.127：8888/web/ProductInformation/queryProductInformation? page＝1”，此时，查看到第一页的产品详情，如图 5.2 所示。

3. 修改产品信息接口测试

修改产品信息需要拥有修改权限，在输入调用接口时需要先登录，通过登录者的 token 值查看角色信息，如果该用户为管理员，即可修改。在地址栏中输入“192.168.1.127：8888/web/ProductInformation/updateProductInformation? id＝40&duiying＝千锋教育

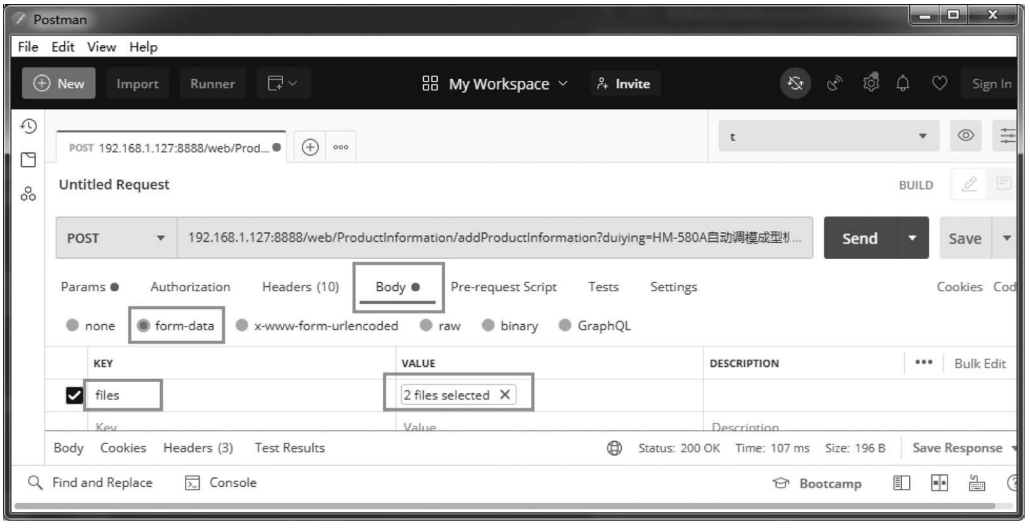

图 5.1　新增产品接口测试

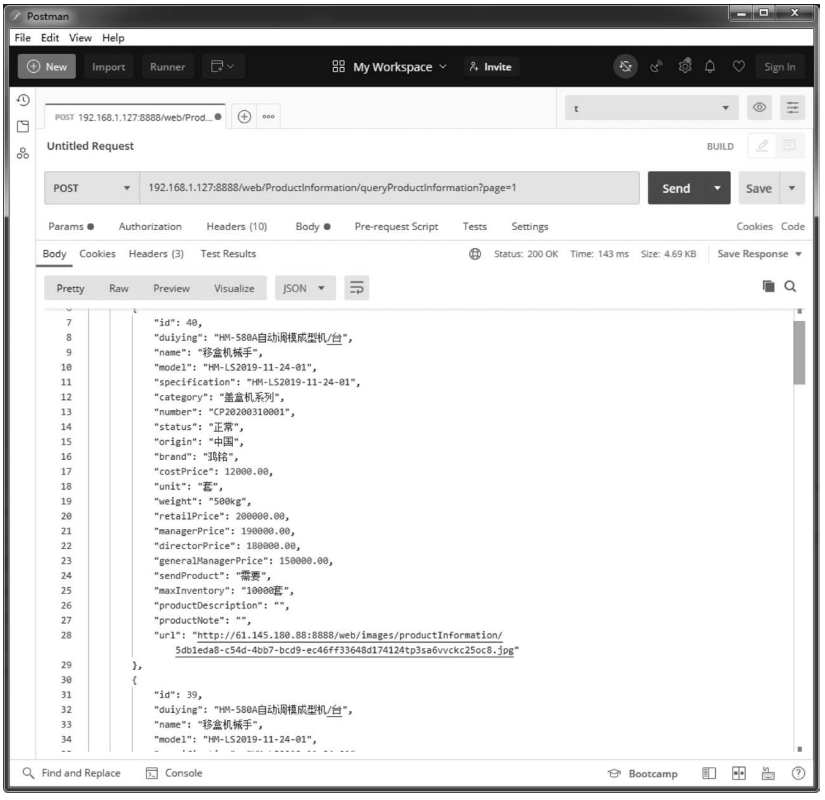

图 5.2　查看产品信息

&name=好程序员 &model=Java&specification=JDK1.8&category=大数据 &number=CP20200310001&status=正常 &origin=中国 &brand=千锋 &costPrice=12000&unit=套 &weight=500kg&retailPrice=200000&managerPrice=190000&directorPrice=180000&generalManagerPrice=150000&sendProduct=需要 &maxInventory=

&productDescription=&productNote”，如图 5.3 所示。

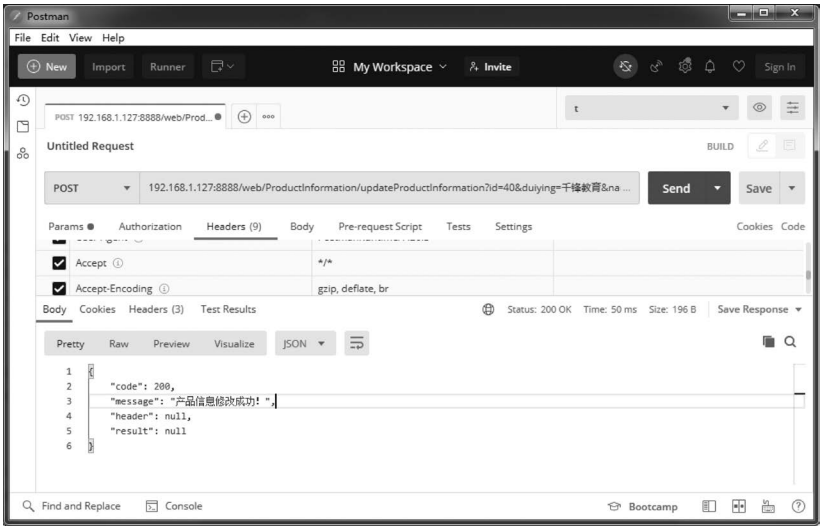

图 5.3　修改产品信息

4. 根据产品信息 id 查询下属图片接口测试

在地址栏中输入“192.168.1.1278888/web/ProductInformation/queryProductionPicture?productId=40”调用接口，查看产品 id 为 40 的图片信息，如图 5.4 所示。

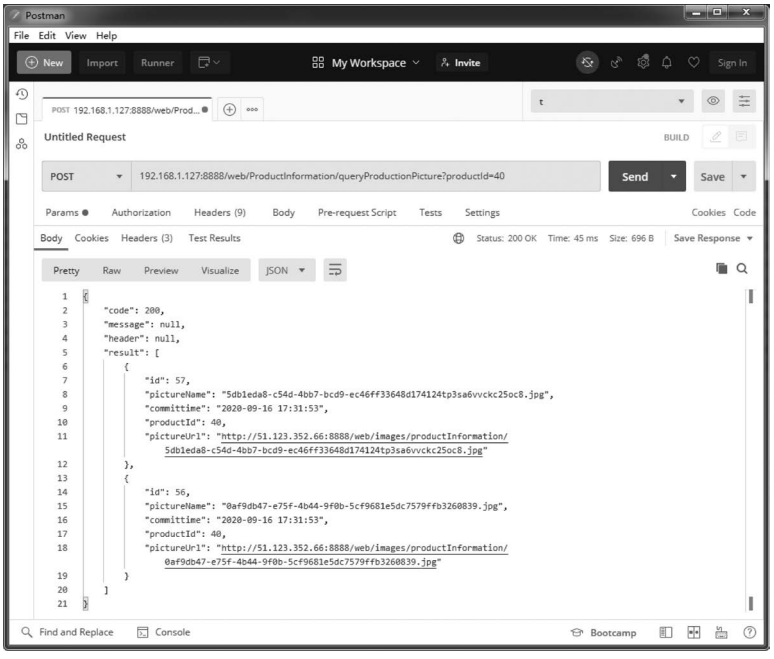

图 5.4　根据产品 id 查看产品图片信息

5. 根据产品 id 查看产品信息接口测试

在地址栏中输入“192.168.1.127：8888/web/ProductInformation/queryPriceById?

id=40”调用接口，查看产品 id 为 40 的产品信息，如图 5.5 所示。

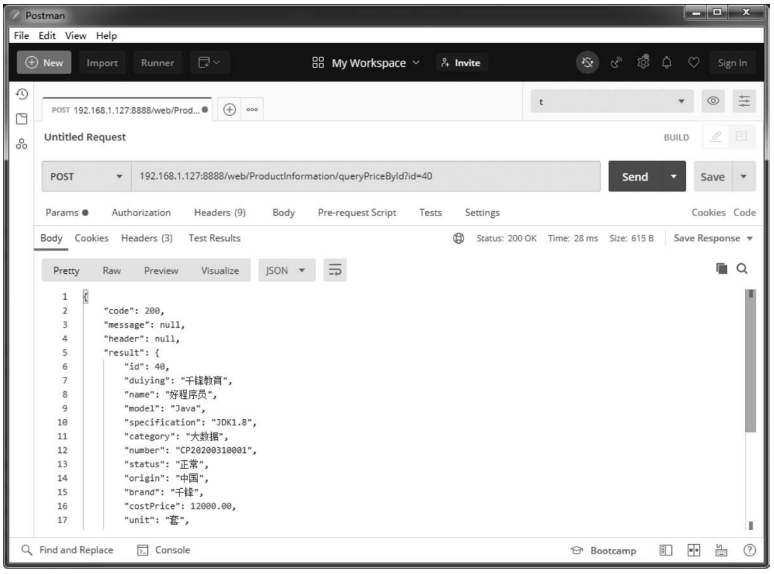

图 5.5　根据产品 id 查看产品信息

6. 根据图片 id 删除图片接口测试

删除产品图片的操作只能由管理员执行，因此在调用该接口时，同样需要先登录获取 token 值，通过 token 值获取角色信息，如果是管理员则可以进行删除操作。在地址栏中输入“192.168.1.127：8888/web/ProductInformation/deleteProductionPicture? id=57”，如图 5.6 所示。

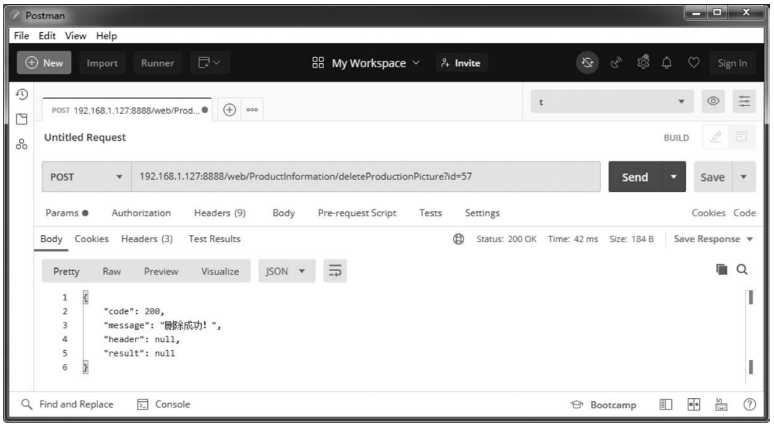

图 5.6　根据图片 id 删除图片

7. 删除产品信息接口测试

删除产品的操作与删除图片类似，在地址栏中输入“192.168.1.127：8888/web/ProductInformation/deleteProductInformation? id=36”，如图 5.7 所示。

8. 根据产品名字关键字模糊查询接口测试

给定关键字“好程序员”，调用模糊查询接口，在地址栏中输入“192.168.1.127：8888/

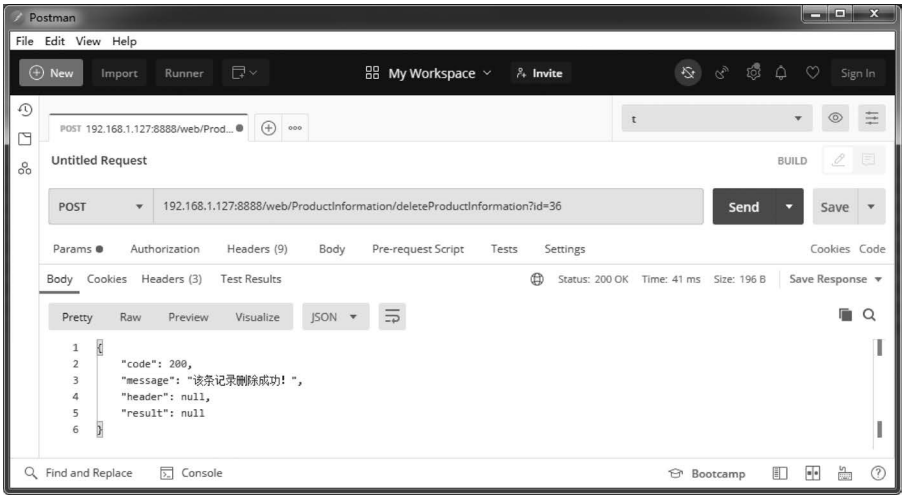

图 5.7　根据产品 id 删除产品

web/ProductInformation/queryAllProductinformationByName? name＝好程序员”，查询结果如图 5.8 所示。

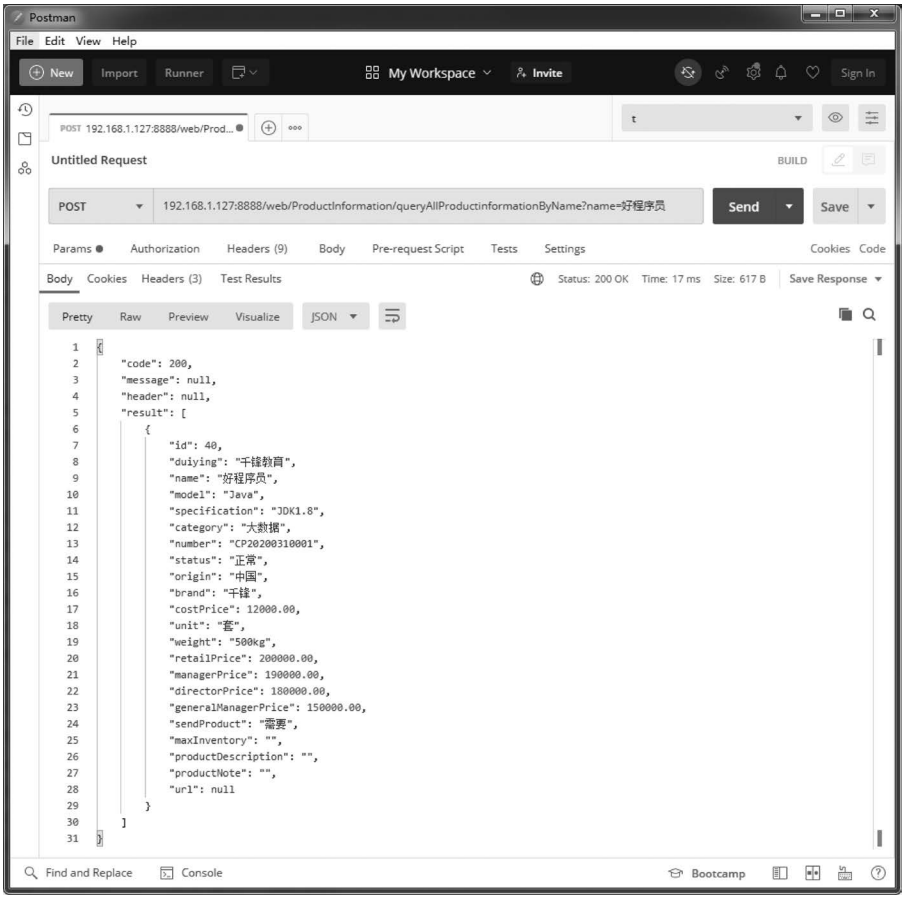

图 5.8　根据产品名字关键字模糊查询产品信息

5.3 实现产品信息管理功能

5.3.1 TakePictureManager 相机与相册上传图片类

为了解决 Android 6.0 权限授权与适配 Android 7.0 文件访问权限的问题，在此引入 TakePictureManager 类，该类只需调用相关方法即可实现相机与相册上传图片的功能。

1. 集成方法

(1)在 res 目录下创建 file_provider_paths.xml 文件。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < paths >
        < external - path path = "" name = "myFile"/>
    </paths >
</resources >
```

(2) 在 AndroidManifest.xml 下的 Application 节点下加入如下代码：

```
< provider
    android: name = "android.support.v4.content.FileProvider"
    android: authorities = "com.qianfeng.mis.fileprovider"
    android: exported = "false"
    android: grantUriPermissions = "true">
    < meta - data
        android: name = "android.support.FILE_PROVIDER_PATHS"
        android: resource = "@xml/file_provider_paths" />
</provider >
```

(3) 将 TakePictureManager 文件复制到项目，并在清单文件中添加相关权限，代码参考 AndroidManifest.xml。

2.使用方法

```
TakePictureManager takePictureManager takePictureManager = new
TakePictureManager(this);
                //开启裁剪，比例为 1：3，宽、高分别为 350、350  (默认不裁剪)
                takePictureManager.setTailor(1, 3, 350, 350);
                //拍照方式
                takePictureManager.startTakeWayByCarema();
                //监听回调
                takePictureManager. setTakePictureCallBackListener ( new TakePictureManager.
takePictureCallBackListener() {
         //成功拿到图片，isTailor 为是否裁剪，outFile 为拿到的文件，filePath 为拿到的 Url
        @Override
        public void successful(boolean isTailor, File outFile, Uri filePath) {
                    }
        //失败回调
```

```
        @Override
        public void failed(int errorCode, List<String> deniedPermissions) {}});
    //把本地的 onActivityResult()方法回调绑定到对象
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        takePictureManager.attachToActivityForResult(requestCode, resultCode, data);
    }
    //权限回调绑定到对象
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
    @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        takePictureManager.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
```

3. 使用的详细步骤

(1) 新建一个 TakePictureManager 对象,构造方法只需传入当前实例 this 即可。

(2) 重写 onActivityResult()方法,调用对象的 attachToActivityForResult()方法,即可实现把拍照或相册回调发数据绑定到对象方法。

(3) 重写 onRequestPermissionsResult()方法,调用对象的 onRequestPermissionsResult()方法,即可实现把权限回调绑定到对象方法。

(4) 只需调用对象方法,即可轻松调用相机或相册。具体的方法参数与描述如表 5.1 所示,具体接口参数与描述如表 5.2 所示。

表 5.1 TakePictureManager()方法参数与描述

方 法 名	参 数	描 述
setTailor(int aspectX, int aspectY, int outputX, int outputY)	要裁剪的宽比例、要裁剪的高比例、要裁剪图片的宽、要裁剪图片的高	一旦调用,表示要裁剪,默认不裁剪
startTakeWayByAlbum()	无参数	调用相机
startTakeWayByCarema()	无参数	调用相机
setTakePictureCallBackListener (takePictureCallBackListener listener)	takePictureCallBackListener 回调接口	调用相机或相册后的回调

表 5.2 TakePictureManager 接口参数与描述

接 口	方 法	描 述
takePictureCallBackListener	successful (boolean isTailor, File outFile, Uri filePath)	成功回调。isTailor:是否已裁剪;outFile:输出的照片文件;filePath:输出的照片 Uri
	failed (int errorCode, List deniedPermissions)	失败回调。errorCode:0 表示相片已移除或不存在;1 表示权限被拒绝。deniedPermissions 当权限被拒绝时,会通过列表传回

5.3.2 产品信息列表展示

1. 编写产品信息列表 View

编写代码,实现效果如图 1.38 所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout xmlns: android = "http://schemas.android.com/apk/res/android"
    android: layout_width = "match_parent"
    android: layout_height = "match_parent"
    android: orientation = "vertical">
    < include
        android: id = "@ + id/layout_title"
        layout = "@layout/title" /
    < com.lcodecore.tkrefreshlayout.TwinklingRefreshLayout
        android: id = "@ + id/refreshLayout"
        android: layout_width = "match_parent"
        android: layout_height = "match_parent"
        android: layout_below = "@ + id/layout_title"
        android: background = " # fff">
        < android.support.v7.widget.RecyclerView
            android: id = "@ + id/rv_list"
            android: layout_width = "match_parent"
            android: layout_height = "match_parent"
            android: background = "@color/colorAccent"
            android: overScrollMode = "never" />
    </com.lcodecore.tkrefreshlayout.TwinklingRefreshLayout >
    < ImageView
        android: id = "@ + id/iv_add"
        android: layout_width = "@dimen/px100"
        android: layout_height = "@dimen/px100"
        android: layout_alignParentRight = "true"
        android: layout_alignParentBottom = "true"
        android: layout_marginTop = "@dimen/px30"
        android: layout_marginRight = "@dimen/px30"
        android: layout_marginBottom = "@dimen/px150"
        android: src = "@mipmap/icon_add" />
</RelativeLayout >
```

产品信息列表的布局较为简单,需要注意的是 TwinklingRefreshLayout 控件、RecyclerView 控件与 ImageView 控件的使用。

2. 编写产品信息列表接口

```
@POST("ProductInformation/queryProductInformation")
Observable < Response < List < DbProductInformation >>> queryProductInformation ( @ Query
("page") int page);
```

在该接口,产品信息列表只需传递 page 分页参数即可请求后端的列表数据。

3. 编写产品信息列表 Controller

在 com. qianfeng. mis. ui. sale. productinfo. activity 包下新建 ProductInformationListActivity 类，该类用于产品信息的展示。通过请求产品信息列表接口，获取产品信息数据进行处理与展示，具体代码如下所示。

```
public class ProductInformationListActivity extends BaseActivity {
    @BindView(R.id.rv_list)
    RecyclerView mRvList;
    @BindView(R.id.refreshLayout)
    TwinklingRefreshLayout mRefreshLayout;
    @BindView(R.id.iv_add)
    ImageView iv_add;
    private ProductInformationAdapter adapter;
    private int page = 1;
    private List<DbProductInformation> listData = new ArrayList<>()
    @Override
    protected int setLayoutResId() {
        return R.layout.activity_product_information_list;
   }
    @Override
    public void initView() {
        setTitle("产品信息");
        setTitleLeftImg(R.mipmap.back_white);
        setTitleRightImg(R.mipmap.icon_search);
        LinearLayoutManager manager = new LinearLayoutManager
        (ProductInformationListActivity.this);
        mRvList.setLayoutManager(manager);
        adapter = new ProductInformationAdapter(listData);
        mRvList.setAdapter(adapter);
        //查看产品信息
        adapter.setOnItemClickListener(new BaseQuickAdapter.OnItemClickListener() {
            @Override
            public void onItemClick(BaseQuickAdapter adapter, View view, int position) {
                Intent intent = new Intent(ProductInformationListActivity.this,
                ProductInformationDesActivity.class);
                intent.putExtra("data", (Serializable) listData.get(position));
                startActivity(intent);
            }
        });
        //下拉刷新，上滑加载更多操作监听
        mRefreshLayout.setOnRefreshListener(new RefreshListenerAdapter() {
            @Override
            public void onRefresh(final TwinklingRefreshLayout refreshLayout) {
                page = 1;
                queryProductInformation();
            }
            @Override
            public void onLoadMore(final TwinklingRefreshLayout refreshLayout) {
                page++;
```

```
                queryProductInformation();
            }
        });
        //初始化产品信息列表
        queryProductInformation()
    }
    //获取产品信息列表数据
    private void queryProductInformation() {
        RestClient.getInstance()
                .getStatisticsService()
                .queryProductInformation(page)
                .subscribeOn(Schedulers.io())
                .compose(bindToLifecycle())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(response -> {
                    if (page == 1) {
                        mRefreshLayout.finishRefreshing();
                    } else {
                        mRefreshLayout.finishLoadmore();
                    }
                    if (response.getCode() == 200) {
                        if (page == 1) {
                            listData.clear();
                            if (response.getResult().size()<= 0) {
                                ToastUitl.showShort("无产品信息");
                            }
                        }
                        if (response.getResult().size() < 20) {
                            mRefreshLayout.setEnableLoadmore(false);
                        }
                        listData.addAll(response.getResult());
                        adapter.setNewData(listData);
                    } else {
                        ToastUitl.showShort(response.getMessage());
                    }
                }, throwable -> {
                    if (page == 1) {
                        mRefreshLayout.finishRefreshing();
                    } else {
                        mRefreshLayout.finishLoadmore();
                    }
                });
    }
    @OnClick({R.id.iv_add})
    public void onViewClicked(View view) {
        switch (view.getId()) {
            //新增产品信息
            case R.id.iv_add:
                Intent intent = new Intent(ProductInformationListActivity.this,
                AddProductInformationActivity.class);
```

```
                startActivity(intent);
                break;
            }
        @Override
        protected void onRightImageViewClick(View view) {
            //查询产品信息
            startActivity(new Intent(this, SearchProductInformationActivity.class));
        }
        @Override
        protected void onResume() {
            super.onResume();
            queryProductInformation();
        }
    }
```

本类作为列表展示类，逻辑相对简单，首先是初始化相关的控件，声明ProductInformationAdapter与初始化分页参数page。

在initView()方法中，将adapter设置给RecyclerView。对下拉刷新与上拉加载设置监听，当下拉刷新时，设置page为1，当上拉加载时，让page自增，然后再调用queryProductInformation()方法做网络请求。接下来通过queryProductInformation()方法初始化产品信息列表。

在queryProductInformation()方法中，逻辑较为简单，主要是处理服务端返回的数据。在该类onViewClicked(View view)方法跳转到“新增产品信息”页面，onRightImageViewClick(View view)方法跳转到“查询产品信息”页面。

由于用户可以在此页面跳转到“新增产品信息”页面，单击查看产品信息后可以删除产品信息，所以在onResume()方法中重新进行网络请求，确保数据是最新的。

5.3.3 新增、修改产品信息

1. 编写新增产品信息View

编写代码，实现效果如图1.39所示，由于“新增产品信息”页面与“修改产品信息”页面相似，在此不再赘述。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout xmlns: android = "http://schemas.android.com/apk/res/android"
    android: layout_width = "match_parent"
    android: layout_height = "match_parent"
    android: orientation = "vertical">
    < include
        android: id = "@ + id/ll_title"
        layout = "@layout/title" />
    < android.support.v4.widget.NestedScrollView
        android: layout_width = "match_parent"
        android: layout_height = "match_parent"
        android: layout_below = "@ + id/ll_title"
        android: background = " # fff"
        android: orientation = "vertical"
```

```
            android: overScrollMode = "never"
            android: padding = "@dimen/px10">
            < LinearLayout
                android: layout_width = "match_parent"
                android: layout_height = "match_parent"
                android: orientation = "vertical">
                < LinearLayout style = "@style/ll_edit_min">
                    < TextView
                        android: id = "@ + id/tv_duiying"
                        style = "@style/text_edit_min"
                        android: text = "对应基准产品: "/>
                    < TextView
                        android: id = "@ + id/et_duiying"
                        style = "@style/edittext_333_28"
                        android: layout_weight = "1"
                        android: hint = ""
                        android: minHeight = "@dimen/px80" />
                    < ImageView
                        android: id = "@ + id/et_1_sousuo"
                        android: layout_width = "@dimen/px80"
                        android: layout_height = "@dimen/px80"
                        android: ellipsize = "end"
                        android: singleLine = "true"
                        android: src = "@drawable/sousuo" />
                </LinearLayout >
    …
        <! -- 由于本书篇幅有限,省略相似布局代码 -- >
                < LinearLayout
                    android: id = "@ + id/ll_fujian"
                    style = "@style/ll_edit_min">
                    < TextView
                        android: id = "@ + id/tv_xiashufujian"
                        style = "@style/text_edit_min"
                        android: text = "下属附件: " />
                    < ImageView
                        android: id = "@ + id/et_photos"
                        android: layout_width = "@dimen/px60"
                        android: layout_height = "@dimen/px60"
                        android: src = "@drawable/photo" />
                    < TextView
                        android: id = "@ + id/et_wenjian"
                        style = "@style/edittext_333_28"
                        android: layout_weight = "1"
                        android: text = "照片/文件"
                        android: visibility = "gone" />
                </LinearLayout >
                < android. support. v7. widget. RecyclerView
                    android: id = "@ + id/rv_list_img"
                    android: layout_width = "match_parent"
                    android: layout_height = "wrap_content"/>
```

```
            <RelativeLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">
                <TextView
                    android:id="@+id/tv_quxiao"
                    style="@style/textview_fff_30"
                    android:layout_width="@dimen/px250"
                    android:layout_height="@dimen/px69"
                    android:layout_marginLeft="@dimen/px100"
                    android:layout_marginTop="@dimen/px40"
                    android:layout_marginBottom="@dimen/px40"
                    android:background="#9292b1"
                    android:gravity="center"
                    android:text="取消" />
                <TextView
                    android:id="@+id/tv_sure"
                    style="@style/textview_fff_30"
                    android:layout_width="@dimen/px250"
                    android:layout_height="@dimen/px69"
                    android:layout_marginLeft="@dimen/px400"
                    android:layout_marginTop="@dimen/px40"
                    android:layout_marginBottom="@dimen/px40"
                    android:background="#4ab5af"
                    android:gravity="center"
                    android:text="提交" />
            </RelativeLayout>
        </LinearLayout>
    </android.support.v4.widget.NestedScrollView>
</RelativeLayout>
```

在该布局中，由于与 LinearLayout 布局属性类似，可以使用 style 属性将通用属性抽取到 styles.xml 文件中，然后使用<LinearLayout style="@style/ll_edit_min">完成相似的布局。

2. 编写新增与修改产品信息接口

```
//新增产品信息(含有图片附件)
@Multipart
@POST("ProductInformation/addProductInformation")
Observable<Response<String>> addProductInformation(
@Query("duiying") String duiying,
                @Query("name") String name,
                @Query("model") String model,
                @Query("specification") String specification,
                @Query("category") String category,
                @Query("number") String number,
                @Query("status") String status,
                @Query("origin") String origin,
                @Query("brand") String brand,
```

```
                        @Query("costPrice") BigDecimal costPrice,
                        @Query("unit") String unit,
                        @Query("weight") String weight,
                        @Query("retailPrice") BigDecimal retailPrice,
                        @Query("managerPrice") BigDecimal managerPrice,
                        @Query("directorPrice") BigDecimal directorPrice,
                        @Query("generalManagerPrice") BigDecimal generalManagerPrice,
                        @Query("sendProduct") String sendProduct,
                        @Query("maxInventory") String maxInventory,
                        @Query("productDescription") String productDescription,
                        @Query("productNote") String productNote,

                        @Part List<MultipartBody.Part> imagePicFiles);
    //新增产品信息(不含图片附件)
    @POST("ProductInformation/addProductInformation")
    Observable<Response<String>> addProductInformation(
    @Query("duiying") String duiying,@Query("name") String name,
                        @Query("model") String model,

      @Query("specification") String specification,
      @Query("category") String category,
      @Query("number") String number,
      @Query("status") String status,
      @Query("origin") String origin,
      @Query("brand") String brand,
      @Query("costPrice") BigDecimal costPrice,
      @Query("unit") String unit,
      @Query("weight") String weight,
      @Query("retailPrice") BigDecimal retailPrice,
      @Query("managerPrice") BigDecimal managerPrice,
      @Query("directorPrice") BigDecimal directorPrice,

  @Query(
"generalManagerPrice")
BigDecimal generalManagerPrice,
@Query("sendProduct") String sendProduct,
@Query("maxInventory") String maxInventory,
@Query("productDescription") String productDescription,
@Query("productNote") String productNote);
```

在新增产品信息中主要有两个接口,不同之处在于第一个接口的最后一个参数可以上传图片,而第二个接口仅仅可以上传表单字段。

```
//修改产品信息
@POST("ProductInformation/updateProductInformation")
Observable<Response<String>>
 updataProductInformation(@Query("id") String id,
 @Query("duiying") String duiying ......);
```

在修改产品信息中只有一个接口，该接口与新增产品信息(不含图片附件类似)一样，仅仅是更新表单字段。

3. 编写新增产品信息 Controller

在 com. qianfeng. mis. ui. sale. productinfo. activity 包下新建 AddProductInformationActivity 类，该类用于新增产品信息。完整代码请扫描右侧二维码下载。


代码
5.3.3-3

这个类里的代码虽然非常多，但是逻辑却不复杂。由于初始化控件代码是使用 Butter Knife 自动生成的，在此进行省略，然后初始化产品相关属性的列表与存储图片附件的列表。在 initView()方法中初始化 Select_Image_Adapter，将其设置为 RecyclerView 的适配器，并给 RecyclerView 设置删除的单击事件。在 onViewClicked(View view)方法中主要有单选框、上传图片与提交表单，在单选框 onRoadPicker(List < String > list, int type)中传入选择的数据以及对应的类型即可完成单选框的填写功能，上传图片功能通过图片上传方式选择弹框(即对话框)，用户可以选择相机与相册的方式上传图片，通过调用 TakePictureManager 中的 startTakeWayByCarema()或 startTakeWayByAlbum()方法实现图片的选取功能，特别注意需要重写 onActivityResult()方法与 onRequestPermissionsResult()方法。最后当用户单击“提交”按钮时会调用 addProductInfomation()方法进行数据的提交。

新增产品信息与修改产品信息的逻辑类似，主要区别是在 EditProductInformationActivity 中，首先通过 getIntent(). getSerializableExtra("data")获取到用户单击的 item 数据，并对其初始化，在 onViewClicked(View view)方法中对单选框进行事件监听与提交取消的监听，当用户修改完成，单击“提交”按钮时调用 updataProductInformation()方法即可完成产品信息的修改。源代码参考 com. qianfeng. mis. ui. sale. productinfo. activity 包下的 EditProductInformationActivity 类。

5.3.4 查看产品信息

1. 编写查看产品信息 View

编写代码，实现效果如图 1.40 所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns: android = "http://schemas.android.com/apk/res/android"
    android: layout_width = "match_parent"
    android: layout_height = "match_parent"
    android: clipChildren = "false"
    android: orientation = "vertical">
    < include layout = "@layout/title" />
    < android. support. v4. widget. NestedScrollView
        android: layout_width = "match_parent"
        android: layout_height = "0dp"
        android: layout_weight = "1"
        android: layout_below = "@ + id/ll_title"
        android: background = " # fff"
        android: orientation = "vertical"
        android: overScrollMode = "never"
        android: padding = "@dimen/px10">
        < LinearLayout
            android: layout_width = "match_parent"
```

```
            android: layout_height = "match_parent"
            android: orientation = "vertical">
            < LinearLayout style = "@style/ll_edit_min">
                < TextView
                    android: id = "@ + id/tv_duiying"
                    style = "@style/text_edit_min"
                    android: text = "对应基准产品： "
                    />
                < TextView
                    android: id = "@ + id/et_duiying"
                    style = "@style/edittext_333_28"
                    android: layout_weight = "1"
                    android: hint = ""
                    android: minHeight = "@dimen/px80" />
                < ImageView
                    android: id = "@ + id/et_1_sousuo"
                    android: layout_width = "@dimen/px80"
                    android: layout_height = "@dimen/px80"
                    android: ellipsize = "end"
                    android: singleLine = "true"
                    android: visibility = "gone"
                    android: src = "@drawable/sousuo" />
            </LinearLayout >
            ...
    </android.support.v4.widget.NestedScrollView >
    < View
        android: layout_width = "match_parent"
        android: layout_height = "0.3dp"
        android: background = "#33666666" />
    < RadioGroup
        android: id = "@ + id/radioGroup"
        android: layout_width = "match_parent"
        android: layout_height = "56dp"
        android: layout_gravity = "bottom|center"
        android: background = "#eee"
        android: clipChildren = "false"
        android: gravity = "center"
        android: orientation = "horizontal">
        < RadioButton
            android: id = "@ + id/rb_one"
            android: layout_width = "0dp"
            android: layout_height = "match_parent"
            android: layout_weight = "1"
            android: background = "@null"
            android: button = "@null"
            android: drawablePadding = "6dp"
            android: gravity = "center"
            android: padding = "5dp"
            android: text = "操作"
            android: textColor = "@color/navigator_color" />
```

```
        <LinearLayout
            android:gravity="center_horizontal"
            android:orientation="vertical"
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="90dp">
            <ImageView
                android:id="@+id/rbAdd"
                android:layout_width="55dp"
                android:layout_height="55dp"
                android:src="@mipmap/comui_tab_post" />
        </LinearLayout>
        <RadioButton
            android:id="@+id/rb_two"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="@null"
            android:button="@null"
            android:drawablePadding="6dp"
            android:gravity="center"
            android:padding="5dp"
            android:text="附件"
            android:textColor="@color/navigator_color" />
    </RadioGroup>
</LinearLayout>
```

在这个布局文件中，使用 TextView 显示数据，在底部的导航栏中采用 RadioButton 作为相关操作与附件的按钮。

2. 编写查看产品信息接口

```
    //根据产品 id 删除产品信息
    @POST("ProductInformation/deleteProductInformation")
    Observable<Response<String>> deleteProductInformation(@Query("id") Integer id);
    //根据产品 id 查询下属图片
    @POST("ProductInformation/queryProductionPicture")
    Observable<Response<List<DbProductinformationPicture>>>
queryProductionPicture(@Query("productId") int productId);
    //根据图片 id 删除图片
    @POST("ProductInformation/deleteProductionPicture")
    Observable<Response<String>> deleteProductionPicture(@Query("id") int id);
```

代码 5.3.4-3

在以上代码中，主要有三个接口，分别是根据产品 id 删除产品信息、根据产品 id 查询下属的图片附件以及根据图片 id 删除图片。

3. 编写查看产品信息 Controller

在 com.qianfeng.mis.ui.sale.productinfo.activity 包下新建 ProductInformationDesActivity，该 Activity 用于查看产品信息。具体代码请扫描右侧二维码下载。

在 ProductInformationDesActivity 中，通过 getIntent(). getSerializableExtra("data") 获得产品信息并初始化，在 initListener()方法中，对底部导航栏进行初始化与监听操作。showPopCaoZuo()与 showPopImg()方法分别是弹出操作与查看附件图片弹窗。在操作中可以进行编辑与删除，编辑则跳转到 EditProductInformationActivity。调用 deleteProductInformation()方法即可删除该产品信息。在 showPopImg()方法中，通过产品 id 获取相应的下属附件图片，通过 RecyclerView 进行展示，当用户单击“删除”按钮删除图片时，则会调用 deleteProductionPicture(int id, int position)方法删除指定的图片。

5.3.5 查询产品信息

1. 编写查询产品信息 View

编写代码，实现效果如图 1.43 所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns: android = http://schemas.android.com/apk/res/android
    android: layout_width = "match_parent"
    android: layout_height = "match_parent"
    android: background = "＃ffffff"
    android: orientation = "vertical">
    < include layout = "@layout/title" />
    < LinearLayout
        android: id = "@ + id/ll_custom"
        android: layout_width = "match_parent"
        android: layout_height = "wrap_content"
        android: layout_margin = "@dimen/px40"
        android: background = "@drawable/stoke_e1e2e4_bg"
        android: orientation = "horizontal"
        android: gravity = "center_vertical"
        android: paddingRight = "@dimen/px30">
        < ImageView
            android: layout_width = "@dimen/px30"
            android: layout_height = "@dimen/px30"
            android: layout_marginLeft = "@dimen/px30"
            android: src = "@drawable/sousuo" />
        < EditText
            android: id = "@ + id/et_shuru"
            style = "@style/edittext_333_28"
            android: hint = "请输入"
            android: textColorHint = "＃222"
            android: background = "@null"
            android: padding = "@dimen/px20"/>
    </LinearLayout >
    < View
      style = "@style/view_line_xi"/>
< android. support. v7. widget. RecyclerView
    android: id = "@ + id/rv_list"
    android: layout_width = "match_parent"
    android: layout_height = "match_parent"/>
</LinearLayout >
```

在该布局中，主要是通过 EditText 获取用户输入的产品名称，查询相应产品显示到 RecyclerView 中。

2. 编写查询产品信息接口

```
@POST("ProductInformation/queryAllProductinformationByName")
    Observable < Response < List < DbProductInformation >>> queryAllProductinformationByName
(@Query("name") String name);
```

通过用户输入的产品名称查询产品信息接口，即可返回对应的产品信息数据。

3. 编写查询产品信息 Controller

在 com. qianfeng. mis. ui. sale. productinfo. activity 包下新建 SearchProductInformationActivity，该 Activity 用于查询产品信息。具体代码如下所示。

```
public class SearchProductInformationActivity extends BaseActivity {
    @BindView(R. id. et_shuru)
    EditText etShuru;
    @BindView(R. id. rv_list)
    RecyclerView rvList;
    private ProductInformationAdapter adapter;
    List < DbProductInformation > listData = new ArrayList < >();
    @Override
    protected int setLayoutResId() {
        return R. layout. activity_search_custom;
    }
    @Override
    public void initView() {
        setTitle("查询产品信息");
        setTitleLeftImg(R. mipmap. back_white);
        LinearLayoutManager manager = new LinearLayoutManager(this);
        rvList. setLayoutManager(manager);
        adapter = new ProductInformationAdapter(listData);
        rvList. setAdapter(adapter);
        //网络请求
        queryAllProductinformationByName();
        adapter. setOnItemClickListener(new BaseQuickAdapter. OnItemClickListener() {
            @Override
            public void onItemClick(BaseQuickAdapter adapter, View view, int position) {
                Intent intent = new Intent(SearchProductInformationActivity. this,
                ProductInformationDesActivity. class);
                intent. putExtra("data", (Serializable) listData. get(position));
                startActivity(intent);
            }
        });
        etShuru. addTextChangedListener(new TextWatcher() {
            public void onTextChanged(CharSequence s, int start, int before, int count) {
            }
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {
```

```
            }
            public void afterTextChanged(Editable s) {
                queryAllProductinformationByName();
            }
        });
    }
    //获取列表数据
    private void queryAllProductinformationByName() {
        RestClient.getInstance()
                .getStatisticsService()
                .queryAllProductinformationByName(etShuru.getText().toString().trim())
                .subscribeOn(Schedulers.io())
                .compose(bindToLifecycle())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(response -> {
                    if (response.getCode() == 200) {
                        if (response.getResult() != null && response.getResult().size() > 0) {
                            listData.clear();
                            listData.addAll(response.getResult());
                            adapter.replaceData(listData);
                        } else {
                            ToastUitl.showShort("无相关产品");
                        }
                    }else {
                        ToastUitl.showShort(response.getMessage());
                    }
                }, throwable -> {
                });
    }
    @Override
    protected void onResume() {
        super.onResume();
        queryAllProductinformationByName();
    }
}
```

在 SearchProductInformationActivity 中，调用 addTextChangedListener() 方法，为 etShuru 添加监听器，在用户输入关键字后会调用 afterTextChanged(Editable s)，进而调用 queryAllProductinformationByName() 进行网络请求操作，最后将数据显示到 RecyclerView 中。