苍穹外卖项目

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

管理端 - 外卖商家使用

用户端 - 点餐用户使用


学习
收获
1
增长开发经验
2
提高业务分析能力
3
提高接口设计能力
4
提高编码能力
5
提高文档阅读能力
6
提高代码调试能力

项目概述、环境搭建

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

目录

Contents

- 软件开发整体介绍
- 苍穹外卖项目介绍
- 开发环境搭建
- 导入接口文档
- Swagger

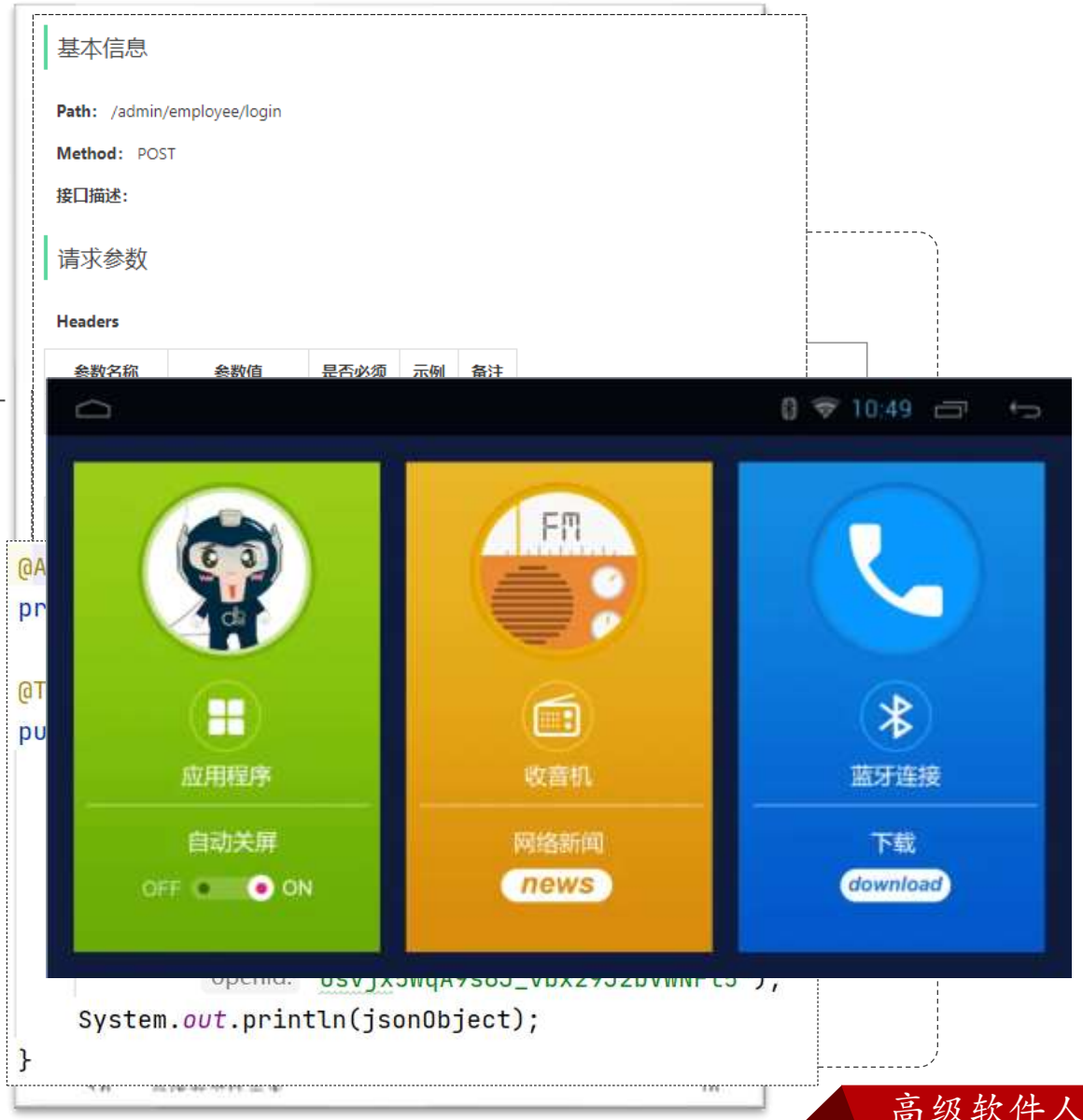
软件开发整体介绍

- 软件开发流程
- 角色分工
- 软件环境

软件开发流程

需求分析 → 设计 → 编码 → 测试 → 上线运维

- 需求分析：需求规格说明书、产品原型
- 设计：UI设计、数据库设计、接口设计
- 编码：项目代码、单元测试
- 测试：测试用例、测试报告
- 上线运维：软件环境安装、配置

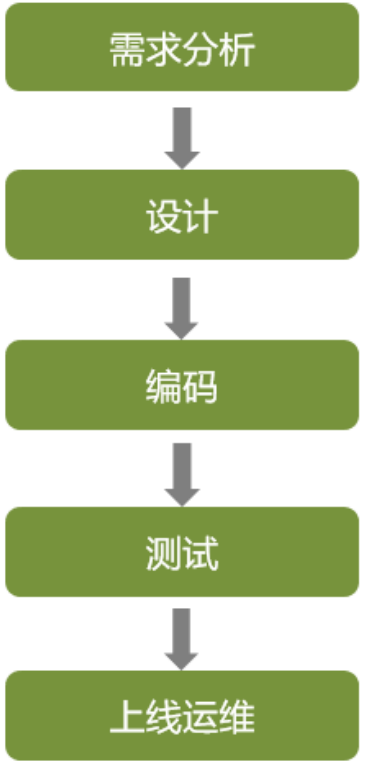


软件开发整体介绍

- 软件开发流程
- 角色分工
- 软件环境

角色分工

- 项目经理：对整个项目负责，任务分配、把控进度
- 产品经理：进行需求调研，输出需求调研文档、产品原型等
- UI设计师：根据产品原型输出界面效果图
- 架构师：项目整体架构设计、技术选型等
- 开发工程师：代码实现
- 测试工程师：编写测试用例，输出测试报告
- 运维工程师：软件环境搭建、项目上线



软件开发整体介绍

- 软件开发流程
- 角色分工
- 软件环境

软件环境

- 开发环境(development)：开发人员在开发阶段使用的环境，一般外部用户无法访问
- 测试环境(testing)：专门给测试人员使用的环境，用于测试项目，一般外部用户无法访问
- 生产环境(production)：即线上环境，正式提供对外服务的环境



目录

Contents

- 软件开发整体介绍
- 苍穹外卖项目介绍
- 开发环境搭建
- 导入接口文档
- Swagger

苍穹外卖项目介绍

- 项目介绍
- 产品原型
- 技术选型

项目介绍

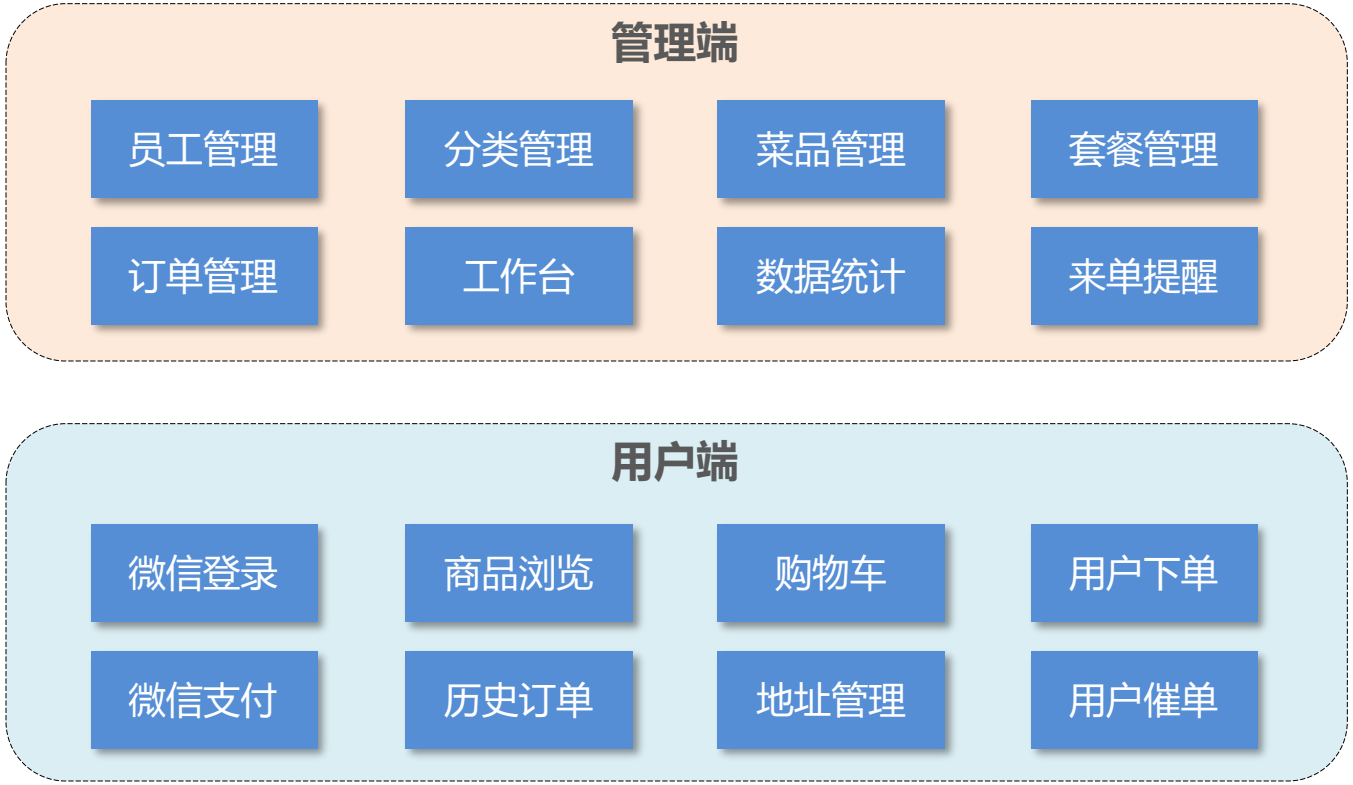
定位：专门为餐饮企业（餐厅、饭店）定制的一款软件产品

管理端 - 外卖商家使用

用户端 - 点餐用户使用

项目介绍

功能架构：体现项目中的业务功能模块



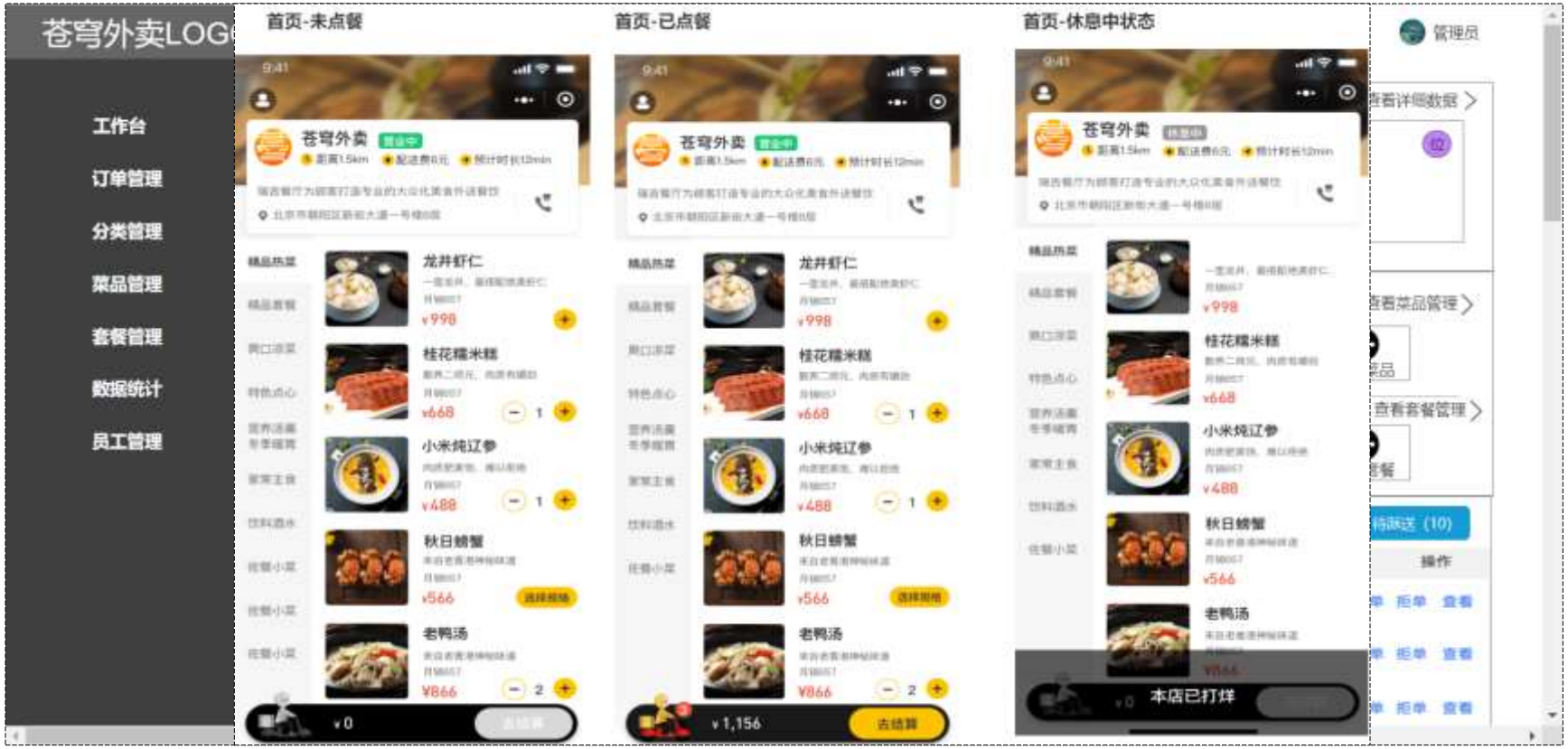
苍穹外卖项目介绍

- 项目介绍
- 产品原型
- 技术选型

产品原型

产品原型：用于展示项目的业务功能，一般由产品经理进行设计

- 苍穹外卖_管理端
- 苍穹外卖_用户端

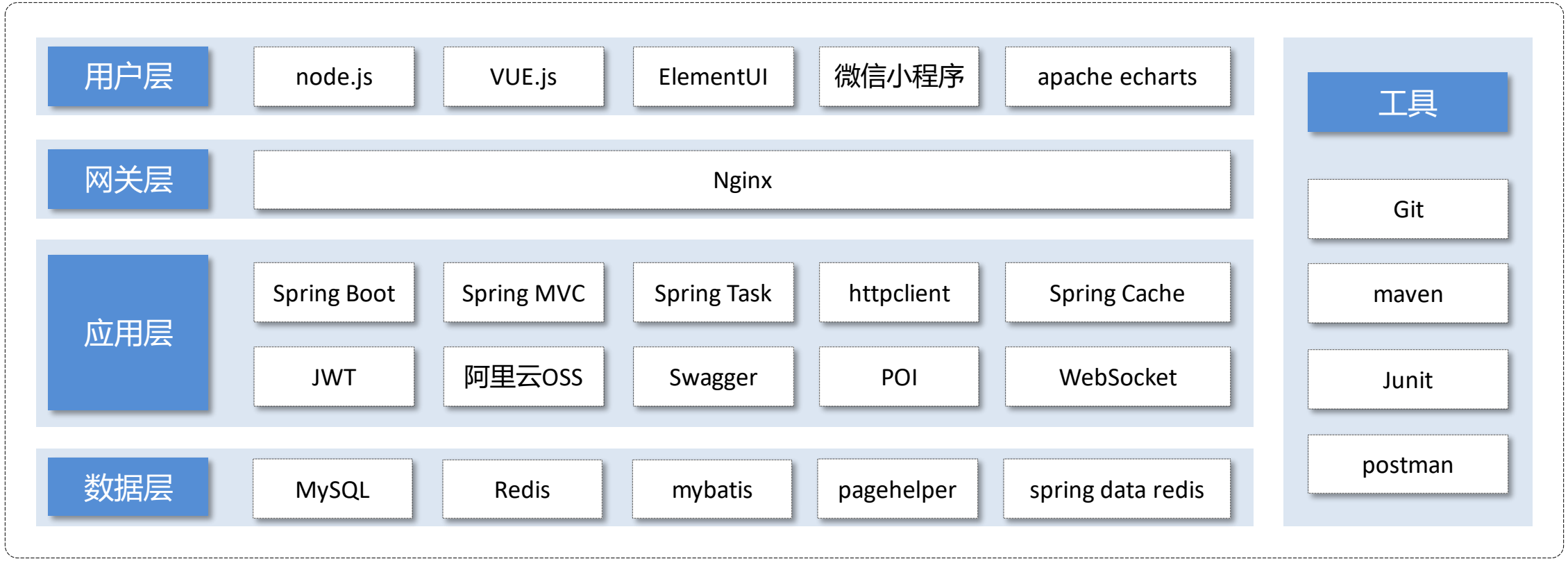


苍穹外卖项目介绍

- 项目介绍
- 产品原型
- 技术选型

技术选型

技术选型：展示项目中使用到的技术框架和中间件等



目录

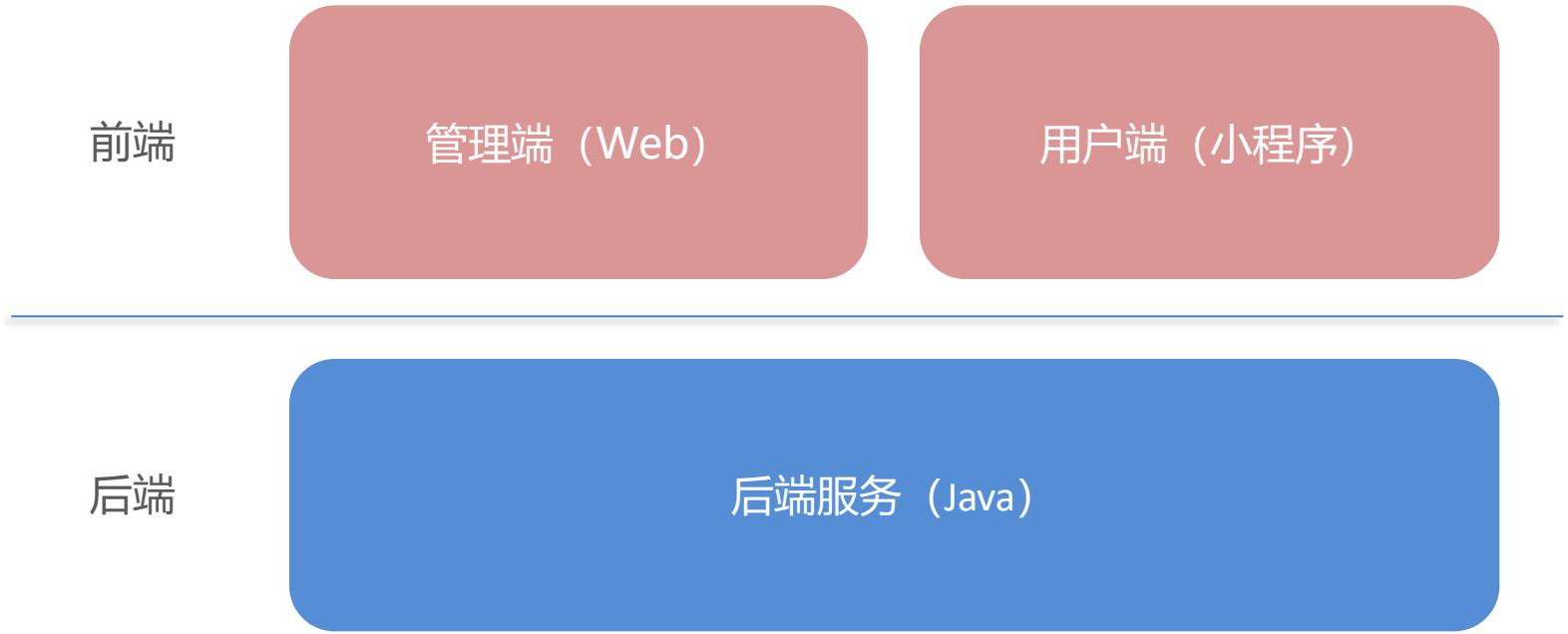
Contents

- 软件开发整体介绍
- 苍穹外卖项目介绍
- 开发环境搭建
- 导入接口文档
- Swagger

开发环境搭建

- 前端环境搭建
- 后端环境搭建
- 完善登录功能

整体结构

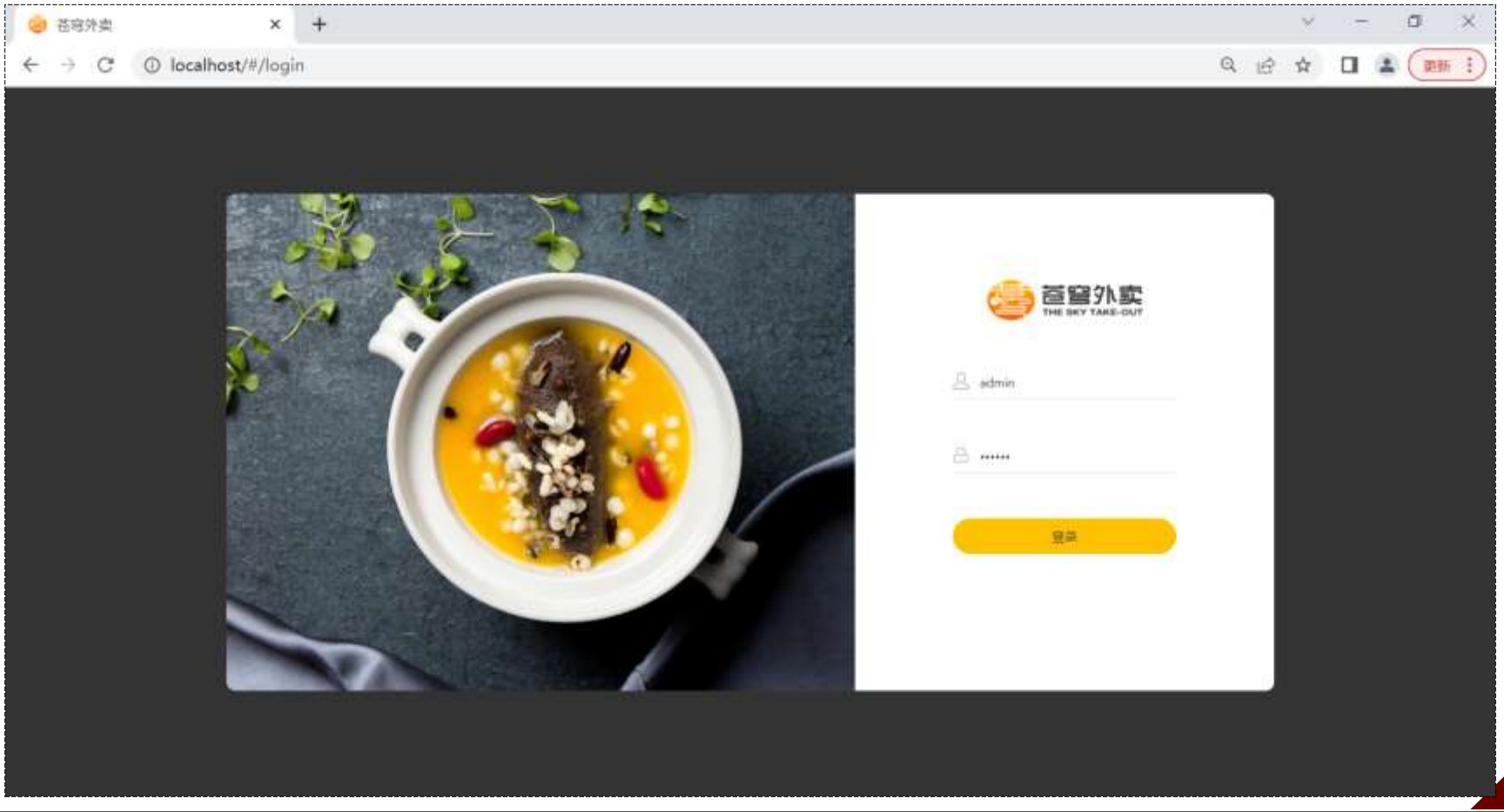


前端环境搭建

前端工程基于 nginx 运行

前端环境搭建

启动nginx：双击 nginx.exe 即可启动 nginx 服务，访问端口号为 80



开发环境搭建

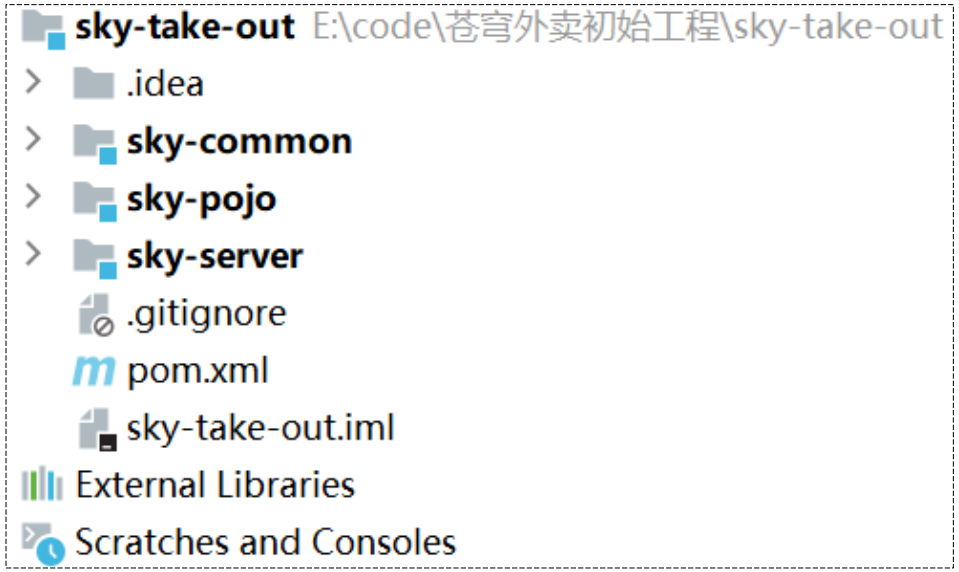
- 前端环境搭建
- 后端环境搭建
- 完善登录功能

后端环境搭建

后端工程基于 maven 进行项目构建，并且进行分模块开发

后端环境搭建 – 熟悉项目结构

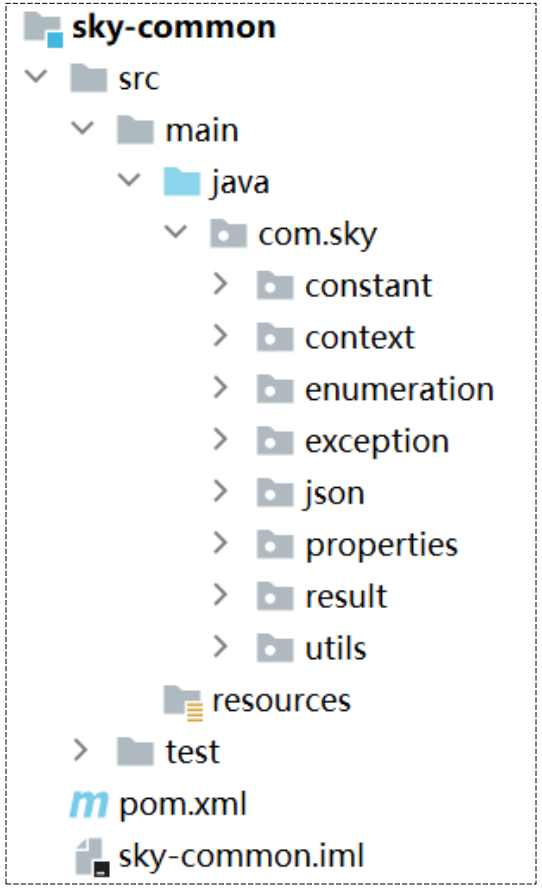
用 IDEA 打开初始工程，了解项目的整体结构



序号	名称	说明
1	sky-take-out	maven父工程，统一管理依赖版本，聚合其他子模块
2	sky-common	子模块，存放公共类，例如：工具类、常量类、异常类等
3	sky-pojo	子模块，存放实体类、VO、DTO等
4	sky-server	子模块，后端服务，存放配置文件、Controller、Service、Mapper等

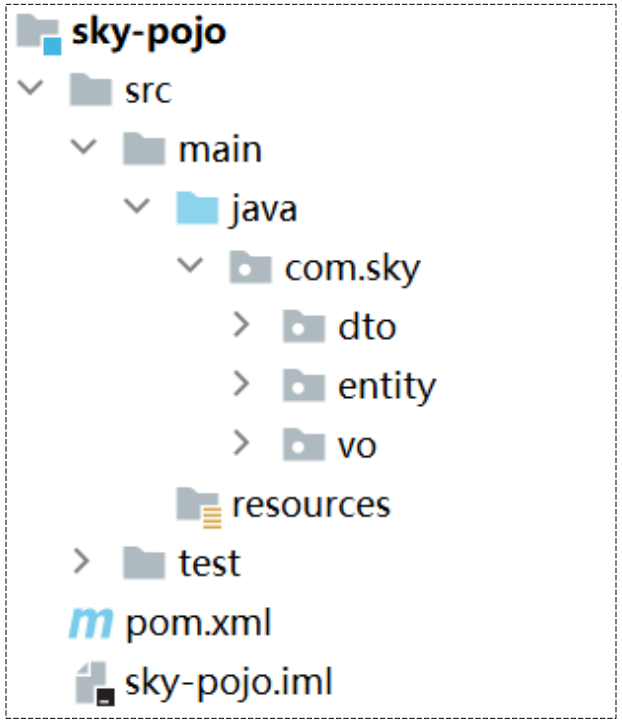
后端环境搭建 – 熟悉项目结构

sky-common 子模块中存放的是一些公共类，可以供其他模块使用



后端环境搭建 – 熟悉项目结构

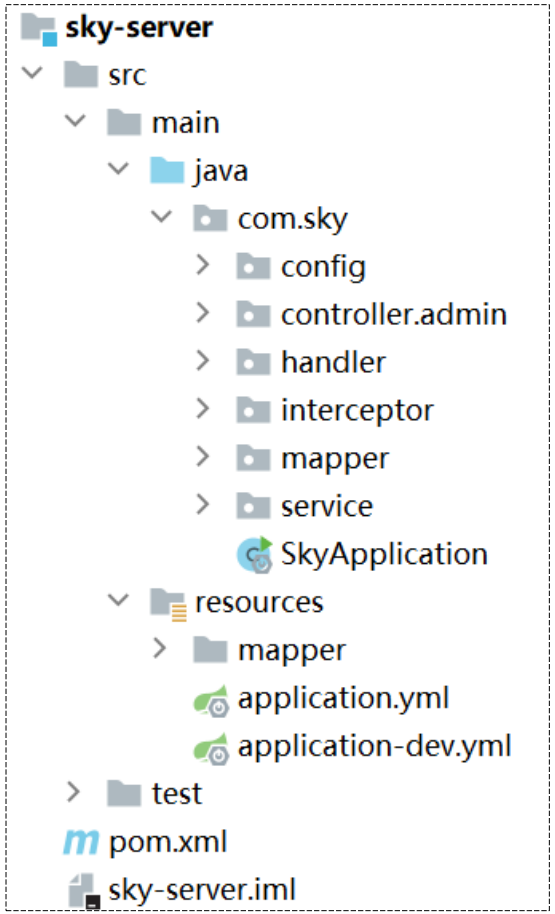
sky-pojo 子模块中存放的是一些 entity、DTO、VO



名称	说明
Entity	实体，通常和数据库中的表对应
DTO	数据传输对象，通常用于程序中各层之间传递数据
VO	视图对象，为前端展示数据提供的对象
POJO	普通Java对象，只有属性和对应的getter和setter

后端环境搭建 – 熟悉项目结构

sky-server 子模块中存放的是 配置文件、配置类、拦截器、controller、service、mapper、启动类等



后端环境搭建 – 使用Git进行版本控制

使用Git进行项目代码的版本控制，具体操作：

- 创建Git本地仓库
- 创建Git远程仓库
- 将本地文件推送到Git远程仓库

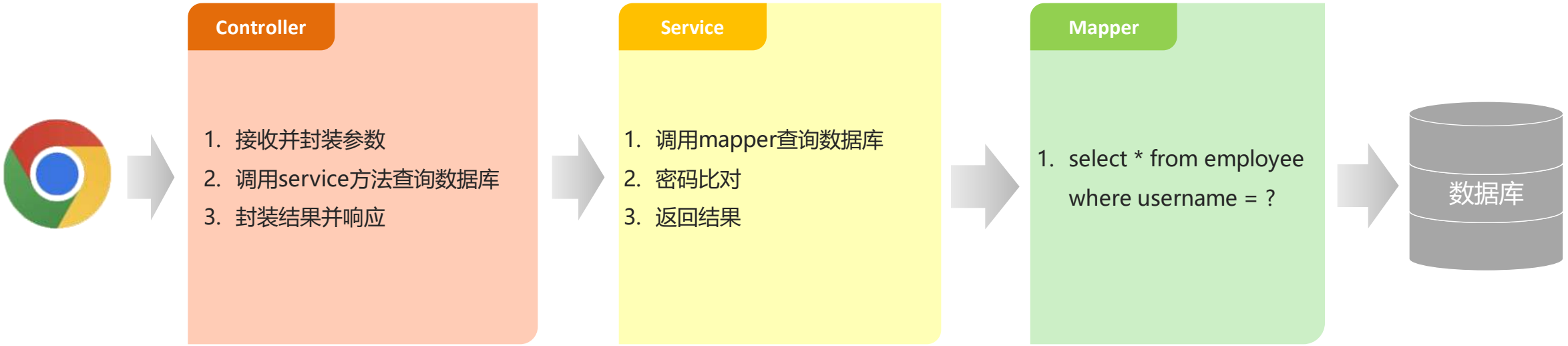
后端环境搭建 – 数据库环境搭建

通过数据库建表语句创建数据库表结构：

序号	表名	中文名
1	employee	员工表
2	category	分类表
3	dish	菜品表
4	dish_flavor	菜品口味表
5	setmeal	套餐表
6	setmeal_dish	套餐菜品关系表
7	user	用户表
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

后端环境搭建 – 前后端联调

后端的初始工程中已经实现了登录功能，直接进行前后端联调测试即可



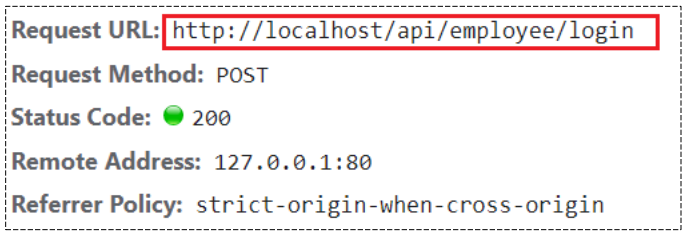
注：可以通过断点调试跟踪后端程序的执行过程

思考

前端发送的请求，是如何请求到后端服务的？

前端请求地址：http://localhost/api/employee/login

后端接口地址：http://localhost:8080/admin/employee/login




```java
@RestController
@RequestMapping("/admin/employee")
@Slf4j
public class EmployeeController {

    @Autowired
    private EmployeeService employeeService;
    @Autowired
    private JwtProperties jwtProperties;

    /** 登录 ...*/
    @PostMapping("/login")
    public Result<EmployeeLoginVO> login(@RequestBody EmployeeLoginDTO employeeLoginDTO) {...}
```

后端环境搭建 – 前后端联调

nginx 反向代理，就是将前端发送的动态请求由 nginx 转发到后端服务器

nginx 播报 编辑 讨论 上传视频

HTTP和反向代理web服务器

Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。Nginx是由伊戈尔·赛索耶夫为俄罗斯访问量第二的Rambler.ru站点（俄文：Рамблер）开发的，公开版本1.19.6发布于2020年12月15日。 [12]

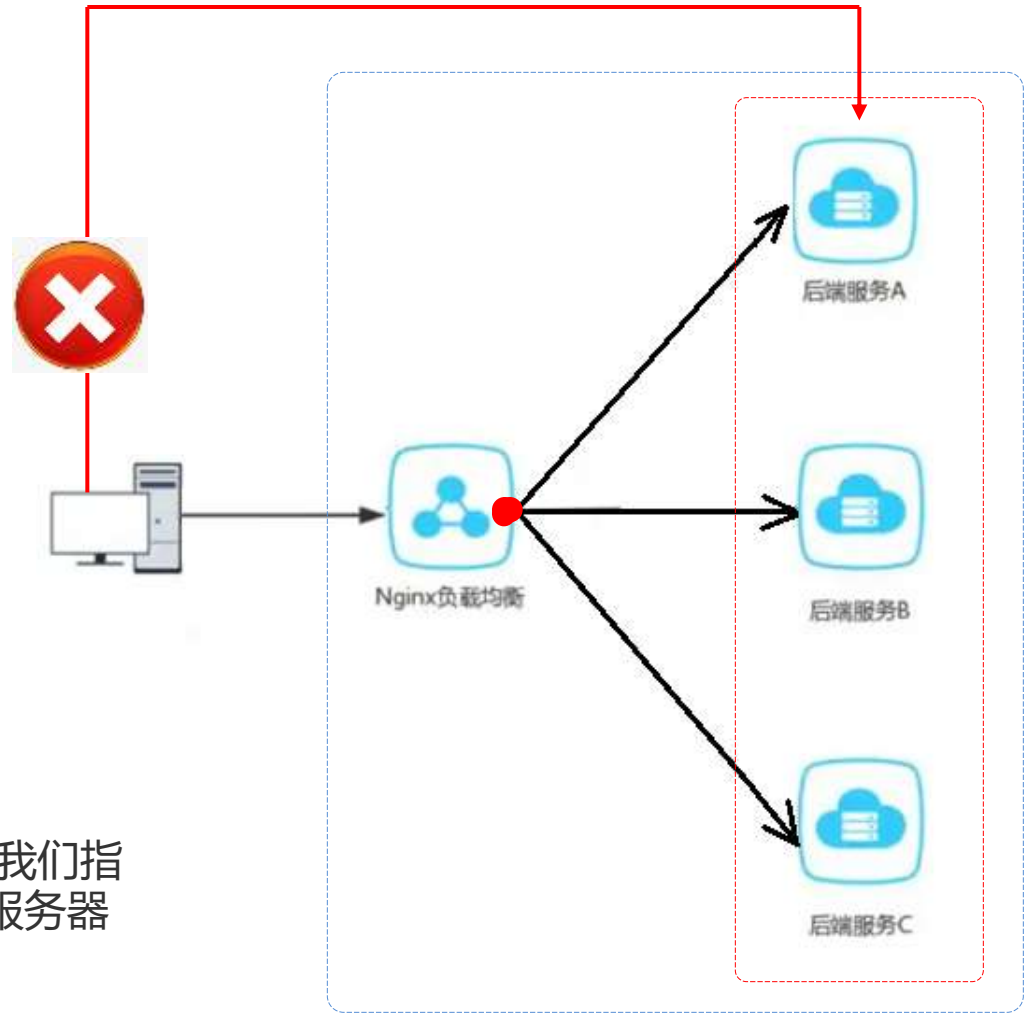
其将源代码以类BSD许可证的形式发布，因它的稳定性、丰富的功能集、简单的配置文件和低系统资源的消耗而闻名。2022年01月25日，nginx 1.21.6发布。 [13]

Nginx是一款轻量级的Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器，在BSD-like 协议下发行。其特点是占有内存少，并发能力强，事实上nginx的并发能力在同类型的网页服务器中表现较好。

后端环境搭建 – 前后端联调

nginx 反向代理的好处：

- 提高访问速度
- 进行负载均衡
- 保证后端服务安全



所谓负载均衡,就是把大量的请求按照我们指定的方式均衡的分配给集群中的每台服务器

后端环境搭建 – 前后端联调

nginx 反向代理的配置方式：

```
server{
        listen 80;
        server_name localhost;

        location /api/ {
                proxy_pass   http://localhost:8080/admin/;  #反向代理
        }

}
```

nginx.conf

http://localhost/api/employee/login

NGINX

http://localhost:8080/admin/employee/login

后端环境搭建 – 前后端联调

nginx 负载均衡的配置方式：

```
upstream webservers{
        server 192.168.100.128:8080;
        server 192.168.100.129:8080;
}

server{
        listen 80;
        server_name localhost;

        location /api/ {
                proxy_pass   http://webservers/admin/;  #负载均衡
        }

}
```

nginx.conf

后端环境搭建 – 前后端联调

nginx 负载均衡策略：

名称	说明
轮询	默认方式
weight	权重方式，默认为1，权重越高，被分配的客户端请求就越多
ip_hash	依据ip分配方式，这样每个访客可以固定访问一个后端服务
least_conn	依据最少连接方式，把请求优先分配给连接数少的后端服务
url_hash	依据url分配方式，这样相同的url会被分配到同一个后端服务
fair	依据响应时间方式，响应时间短的服务将会被优先分配

开发环境搭建

- 前端环境搭建
- 后端环境搭建
- 完善登录功能

完善登录功能

问题：员工表中的密码是明文存储，安全性太低。

	id	name	username	password	phone	sex	id_number	status	create_time	update_time	create_user	update_user
▶	1	管理员	admin	123456	13812312312	1	110101199001010047	1	2022-02-15 15:51:20	2022-02-17 09:16:20	10	1

思路

1. 将密码加密后存储，提高安全性
2. 使用MD5加密方式对明文密码加密

MD5信息摘要算法（英语：MD5 Message-Digest Algorithm），一种被广泛使用的密码散列函数，可以产生出一个128位（16字节）的散列值（hash value），用于确保信息传输完整一致。MD5由美国密码学家罗纳德·李维斯特（Ronald Linn Rivest）设计，于1992年公开，用以取代MD4算法。这套算法的程序在 RFC 1321 标准中被加以规范。1996年后该算法被证实存在弱点，可以被加以破解，对于需要高度安全性的数据，专家一般建议改用其他算法，如SHA-2。2004年，证实MD5算法无法防止碰撞（collision），因此不适用于安全性认证，如SSL公开密钥认证或是数字签名等用途。

123456 → MD5加密 → e10adc3949ba59abbe56e057f20f883e

步骤

- 完善登录功能

1. 修改数据库中明文密码，改为MD5加密后的密文
2. 修改Java代码，前端提交的密码进行MD5加密后再跟数据库中密码比对

EmployeeServiceImpl

```
//进行md5加密，然后再进行比对
password = DigestUtils.md5DigestAsHex(password.getBytes());
if (!password.equals(employee.getPassword())) {
    //密码错误
    throw new PasswordErrorException(MessageConstant.PASSWORD_ERROR);
}
```

目录

Contents

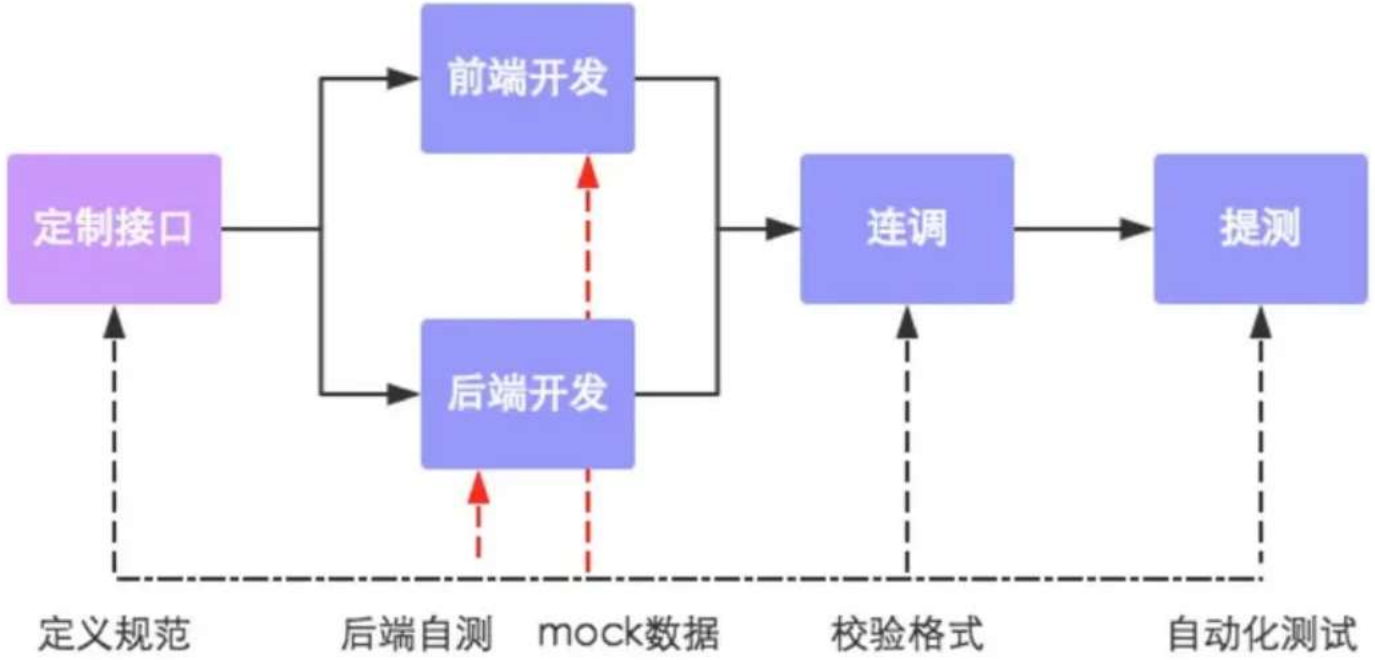
- 软件开发整体介绍
- 苍穹外卖项目介绍
- 开发环境搭建
- 导入接口文档
- Swagger

导入接口文档

- 前后端分离开发流程
- 操作步骤

说明

前后端分离开发流程：



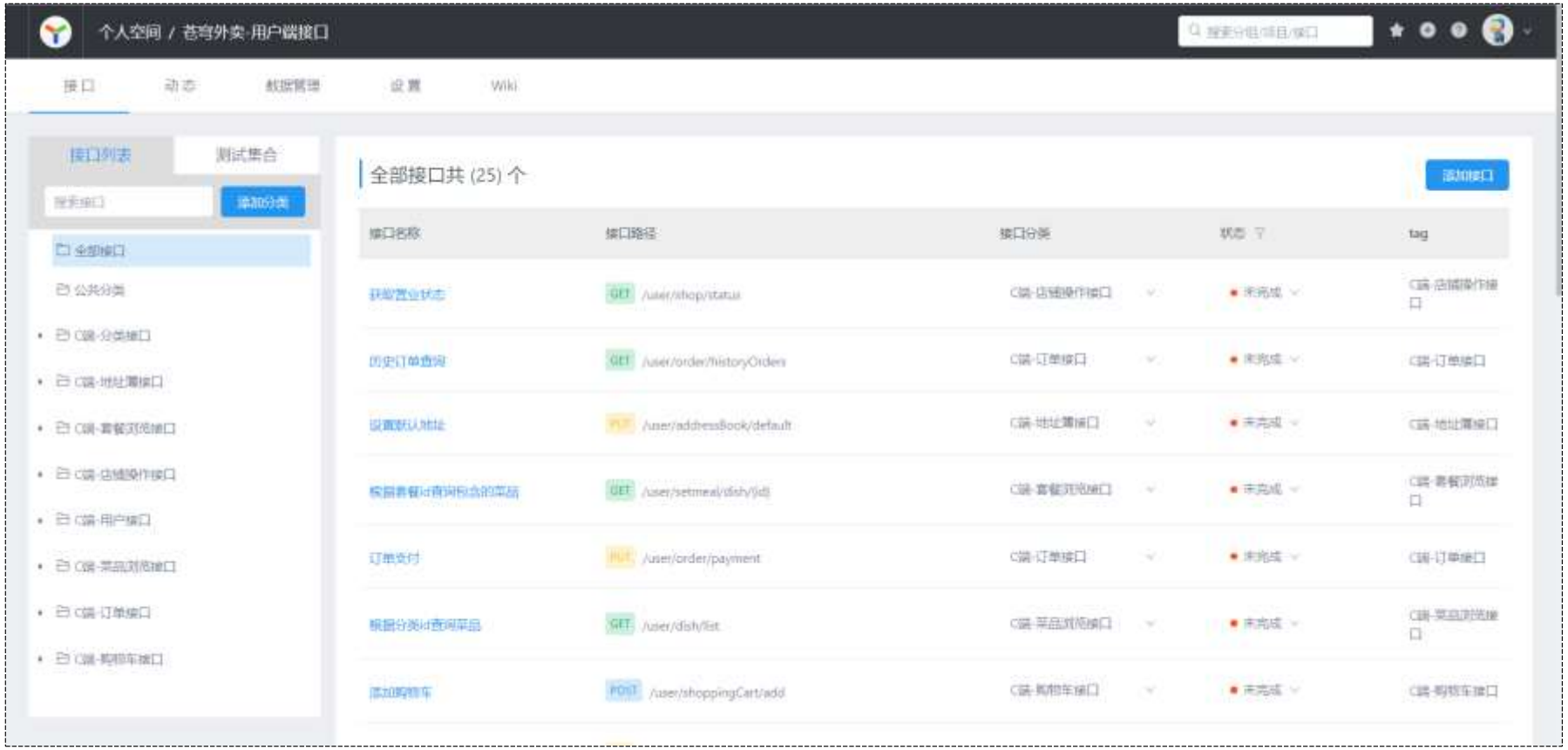
导入接口文档

- 前后端分离开发流程
- 操作步骤

操作步骤

将课程资料中提供的项目接口导入YApi。

- 苍穹外卖-管理端接口.json
- 苍穹外卖-用户端接口.json



目录

Contents

- 软件开发整体介绍
- 苍穹外卖项目介绍
- 开发环境搭建
- 导入接口文档
- Swagger

Swagger

- 介绍
- 使用方式
- 常用注解

介绍

使用Swagger你只需要按照它的规范去定义接口及接口相关的信息，就可以做到生成接口文档，以及在线接口调试页面。

官网：https://swagger.io/

Knife4j 是为Java MVC框架集成Swagger生成Api文档的增强解决方案。

```
<dependency>
        <groupId>com.github.xiaoymin</groupId>
        <artifactId>knife4j-spring-boot-starter</artifactId>
        <version>3.0.2</version>
</dependency>
```

pom.xml

Swagger

- 介绍
- 使用方式
- 常用注解

步骤

- 使用方式

1. 导入 knife4j 的maven坐标
2. 在配置类中加入 knife4j 相关配置
3. 设置静态资源映射，否则接口文档页面无法访问

WebMvcConfiguration

```
/**
 * 设置静态资源映射
 * @param registry
 */
protected void addResourceHandlers(ResourceHandlerRegistry registry) {
    log.info("开始设置静态资源映射...");
    registry.addResourceHandler("/doc.html").addResourceLocations("classpath:/META-INF/resources/");
    registry.addResourceHandler("/webjars/**").addResourceLocations("classpath:/META-INF/resources/webjars/");
}
```

WebMvcConfiguration

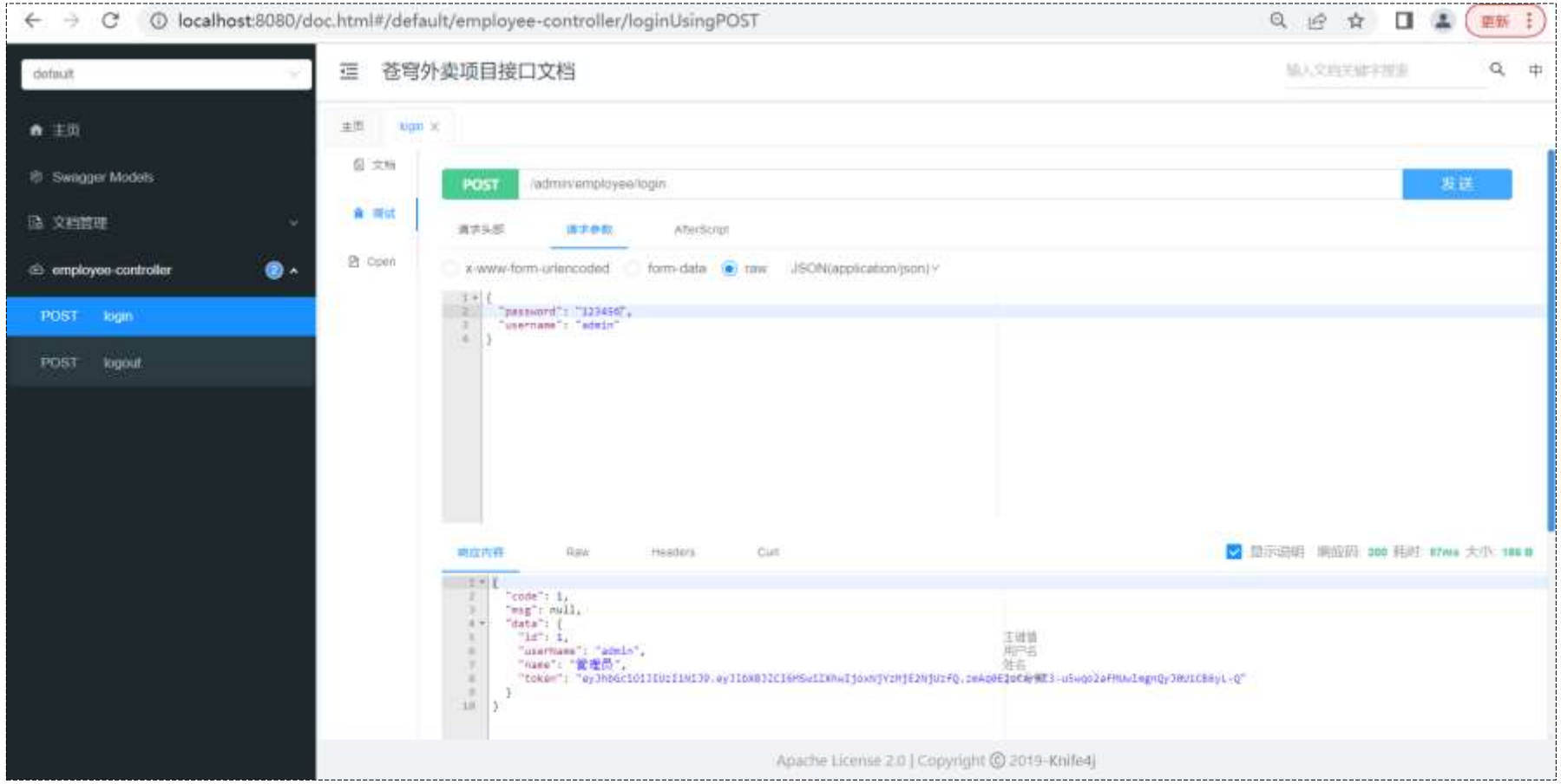
```
            .select()
            //指定生成接口需要扫描的包
            .apis(RequestHandlerSelectors.basePackage("com.sky.controller"))
            .paths(PathSelectors.any())
            .build();

    return docket;
}
```

使用方式

接口文档访问路径为 http://ip:port/doc.html

使用方式

接口测试


localhost:8080/doc.html#/default/employee-controller/loginUsingPOST
苍穹外卖项目接口文档
default
主页
Swagger Models
文档管理
employee-controller
POST login
POST logout
POST
/admin/employee/login
发送
x-www-form-urlencoded
form-data
raw
JSON(application/json)
{
"password": "123456",
"username": "admin"
}
Raw
Headers
Curl
{
"code": 1,
"msg": null,
"data": {
"id": 1,
"username": "admin",
"name": "管理员",
"token": "eyJhbGciOiJIUzI1NiJ9.eyJlbXBJZCI6MSwiZXhwIjoxNjYzMjE2NjUzfQ.zmAq0EIpCAqWEI-uSwqo2aFMAvImgnQyJWUiCBkyL-Q"
}
}
Apache License 2.0 | Copyright © 2019-Knife4j

通过 Swagger 就可以生成接口文档，那么我们就不需要 Yapi 了？

1、Yapi 是设计阶段使用的工具，管理和维护接口

2、Swagger 在开发阶段使用的框架，帮助后端开发人员做后端的接口测试

Swagger

- 介绍
- 使用方式
- 常用注解

常用注解

通过注解可以控制生成的接口文档，使接口文档拥有更好的可读性，常用注解如下：

注解	说明
@Api	用在类上，例如Controller，表示对类的说明
@ApiModel	用在类上，例如entity、DTO、VO
@ApiModelProperty	用在属性上，描述属性信息
@ApiOperation	用在方法上，例如Controller的方法，说明方法的用途、作用