目录

# 一、项目亮点

该项目在项目管理、自动化部署、微服务架构、需求分析、系统测试和文档管理等方面展现了全面性和专业性。通过 Gitee 企业版进行高效的项目管理，确保了团队成员间的良好协作；详尽的项目文档提供了透明的进展追踪，包括规划、需求、任务和缺陷管理。项目实现了 Gitee-Go 自动化部署流水线，提升了部署效率和准确性，同时采用微服务架构增强了系统的可维护性和可扩展性。详细的需求分析和系统架构设计为开发和测试提供了清晰指导，关键服务的场景流程图确保了业务流程的准确性。此外，数据库类图和全面的测试报告（包括单个服务、整合服务及压力测试）保障了软件质量与性能。接口文档详细记录了 API 调用和集成的指导，Docker 批量部署工具和自动化部署脚本简化了部署流程，提升了速度和一致性。最后，通过 Gitee 企业版和 Gitee Go+微服务自动化部署，实现了项目版本的有效控制与管理。

# 二、项目管理

采用了 Gitee 企业版进行项目管理，项目名称是智慧服务项目。[项目概览 - 智慧服务](https://e.gitee.com/easy-z/projects/704150/overview)

## 2.1 项目成员

刘博文拟定为企业拥有者，邀请各个组员加入 Gitee 企业版进行可视化的项目管理，给各个组员统筹分配任务和监控进度，以下是各个组员的联系方式。

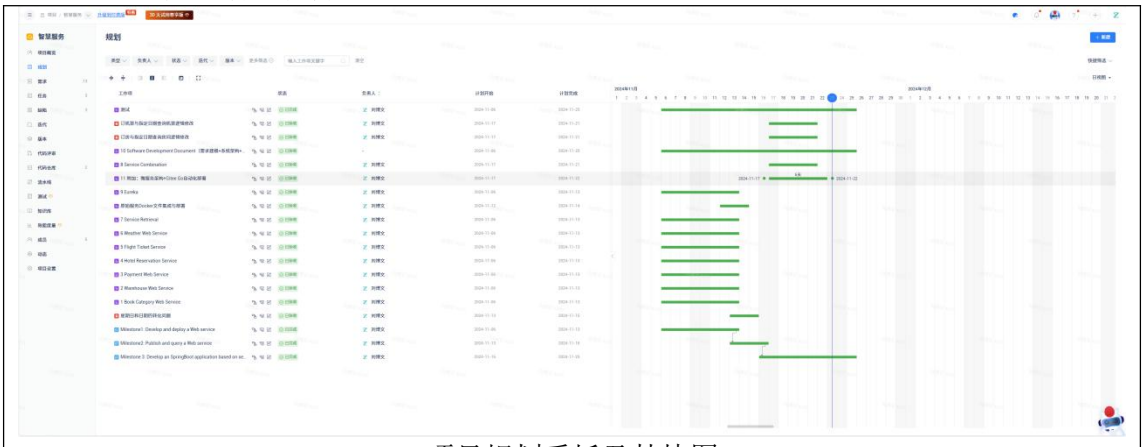
| 姓名 | 角色 | 职位 | 电话号码 | 电子邮件 |
|---|---|---|---|---|
| 刘 | 企业拥有者 | 项目经理、架构设计师、开发总包工程师 | 123456789 | 123456789@foxmail.com |
| 刘 | 普通成员 | 软件开发工程师 | 123456789 | 123456789@foxmail.com |
| 刘 | 普通成员 | 软件开发工程师 | 123456789 | 123456789@foxmail.com |
| 刘 | 普通成员 | 软件开发工程师 | 123456789 | 123456789@foxmail.com |
| 刘 | 普通成员 | 软件开发工程师 | 123456789 | 123456789@foxmail.com |
| 刘 | 普通成员 | 软件开发工程师 | 123456789 | 123456789@foxmail.com |

另外，组长刘博文要求各位组员进行员工自我评价，如下表所示：

| 姓名 | 员工自我评价 |
|---|---|
| 刘 | 进行项目规划、架构设计、需求建模、服务开发、服务整合、软件测试、员工培训、额外微服务项目开发、报告编写等任务，巩固加强了自己在项目开发、项目管理、架构设计、团队协作方面的能力。 |
| 刘 | 根据指派任务完成部分服务开发、单元测试等，熟悉了 docker 部署服务的流程，了解到敏捷开发的过程。 |
| 刘 | 完成部分 service 开发和软件测试，参与编写文档报告等，加强了团队合作能力，对微服务架构和自动化流程有了新的认识。 |
| 刘 | 根据指派的任务，完成了部分某些特定功能的设计与实现及单元测试等工作，了解了后端逻辑的实现，包括数据库操作和服务接口的定义。在项目期间，我不仅增强了个人的技术栈，还学会了使用新的工具和技术来提高工作效率。同时，我也意识到沟通对于团队协作的重要性。 |
| 刘 | 根据指派任务完成部分服务开发、单元测试等，参与了酒店预订服务的开发工作以及服务检索等工作，确保了服务的稳定性和用户预订流程的顺畅。从中熟悉了微服务架构等相关内容，巩固了项目开发的能力。 |
| 刘 | 根据指派任务完成相关服务开发、单元测试等，完成酒店预订服务的开发以及服务集成等工作，维护服务稳定性。期间熟悉了微服务架构等相关内容，深刻体会到了团队开发的重要性。 |

## 2.2 项目规划

这是项目的总体规划，项目经理刘博文根据大作业要求遵循企业项目开发的五个阶段（启动-规划-执行-监督-收尾），设计了项目的里程碑、任务需求、缺陷、迭代等过程，形成项目看板，其中的甘特图形象生动地展示了整个敏捷项目的开发周期，从这里可以清楚地看到任务流的进度。



项目规划看板及甘特图

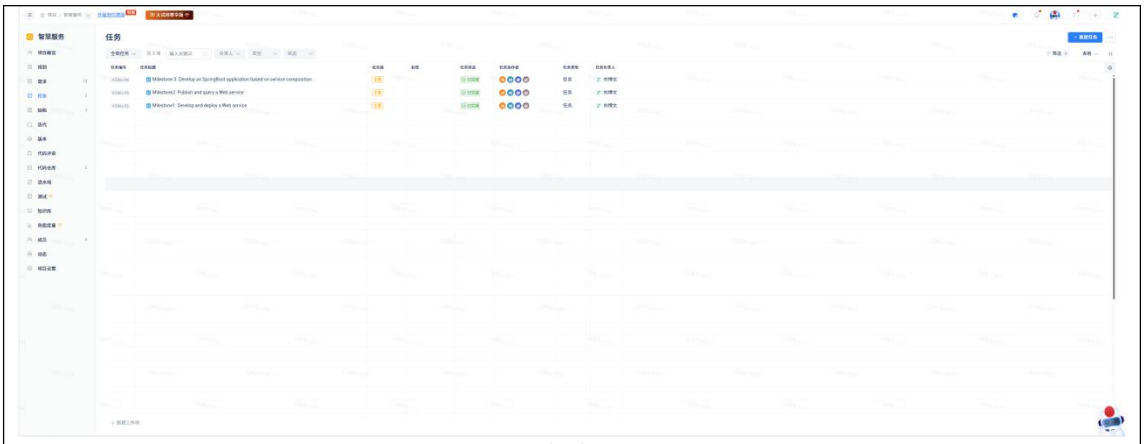
## 2.3 项目需求

通过分析大作业要求，细化了项目的需求并进行了任务指派。

项目需求看板

## 2.4 项目任务

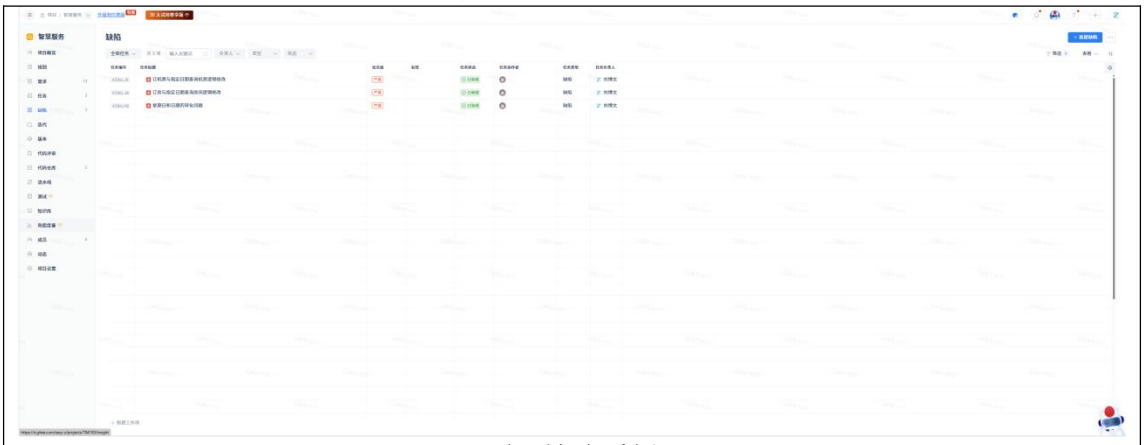
根据大作业要求，设置了三个里程碑。



项目任务看板

## 2.5 项目缺陷

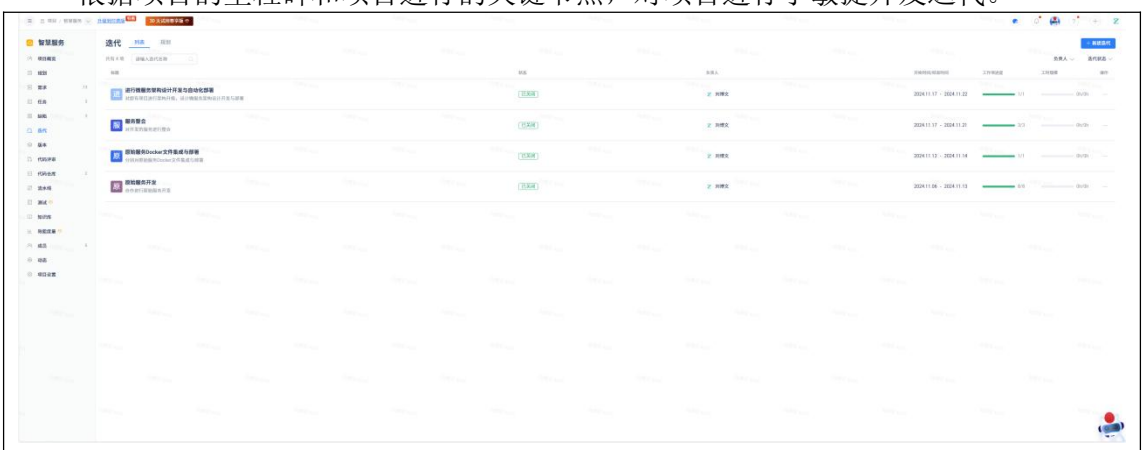
在项目开发过程中遇到了很多问题并进行了解决，这里列出一些出现的项目缺陷以及指定了对应的开发人员去进行修复和验证，



项目缺陷看板

## 2.6 项目迭代

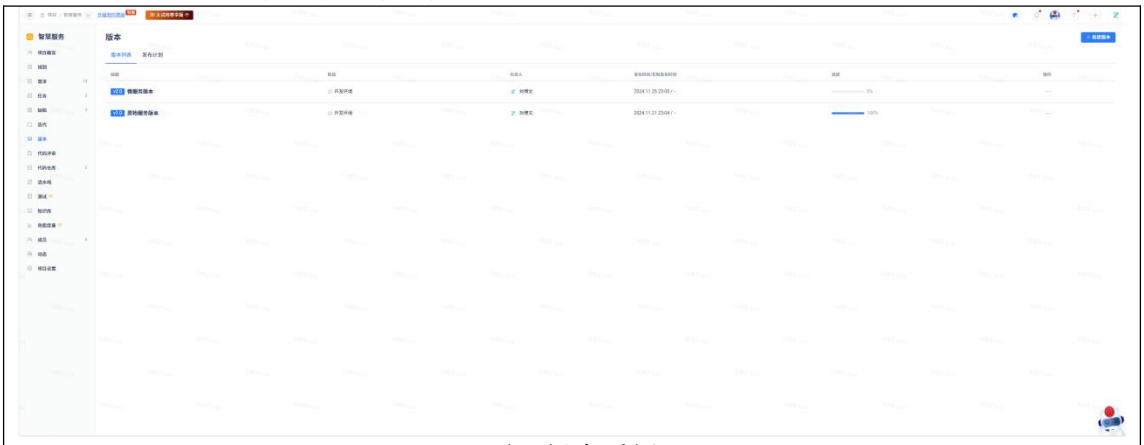
根据项目的里程碑和项目进行的关键节点，对项目进行了敏捷开发迭代。



项目迭代看板

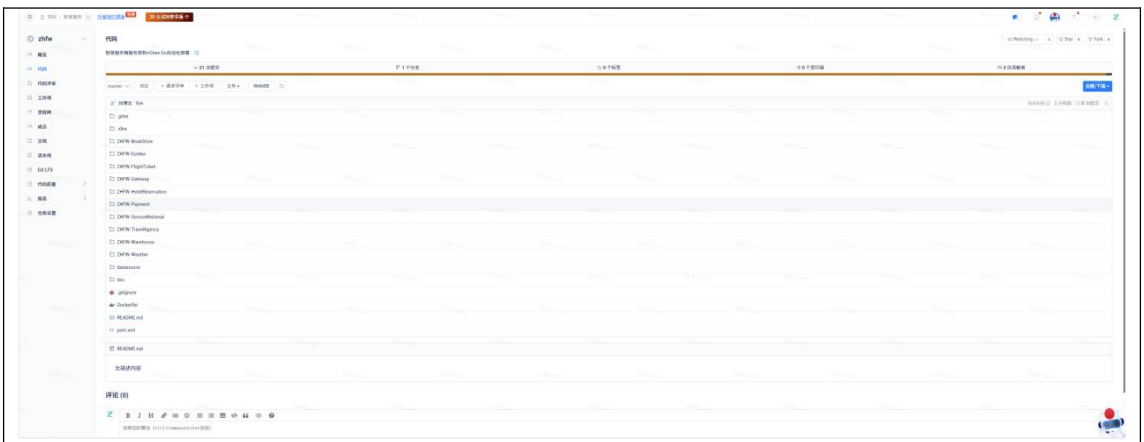
## 2.7 项目版本

我们的代码仓库有两个，大作业的需求是 Docker 项目的部署，组长刘博文在完成此工作的基础上，额外采用微服务架构对项目进行重构。



项目版本看板

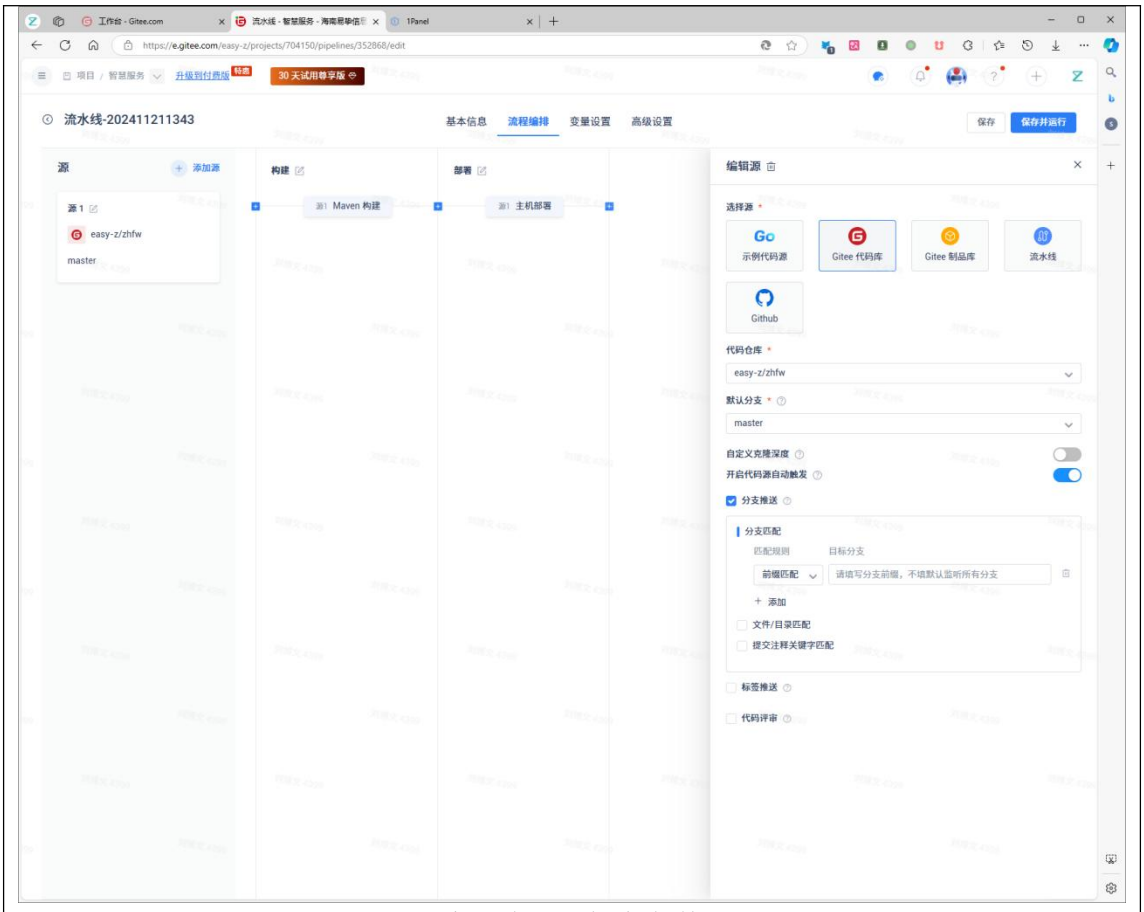
智慧服务 Docker 部署项目



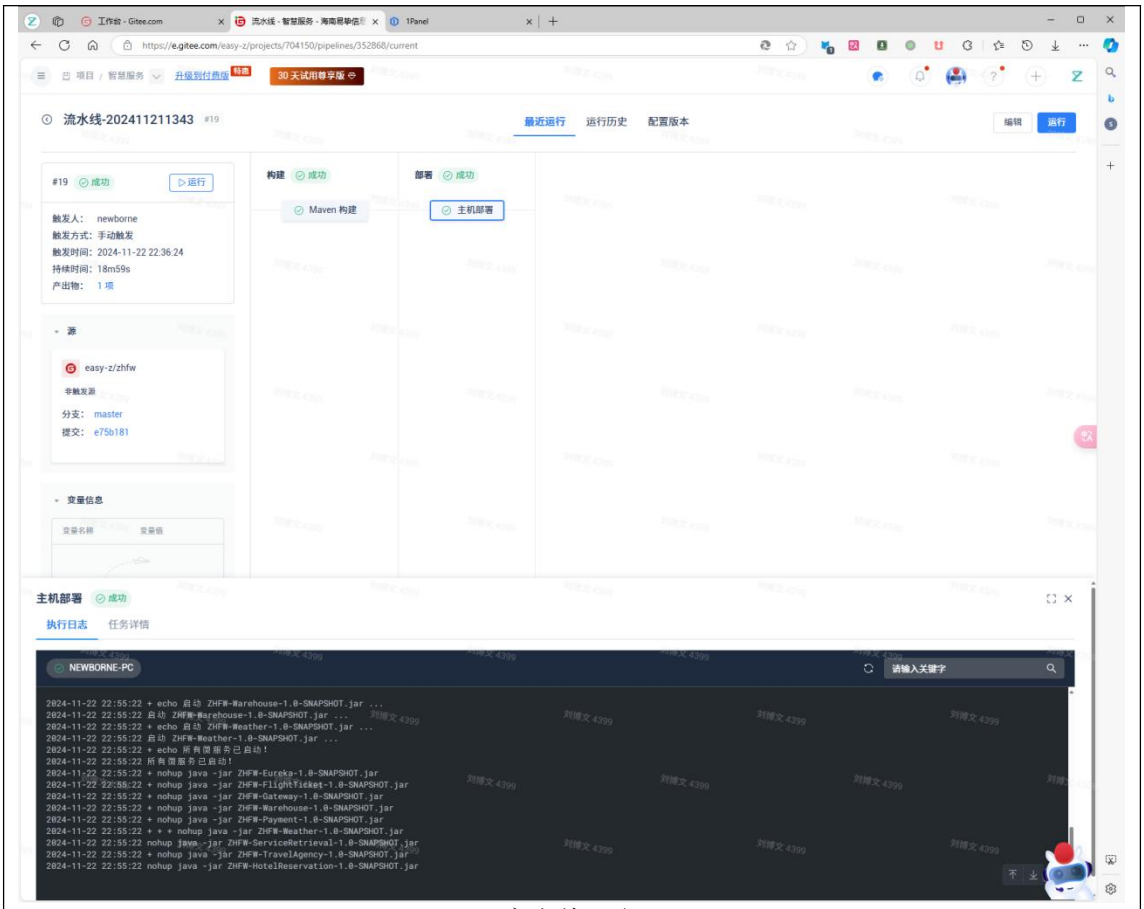
智慧服务微服务架构+Gitee Go 自动化部署项目

## 2.8 Gitee-Go 自动化部署流水线

在 Gitee-Go 自动化部署的界面可以很方便地关联企业项目的代码仓库，并设置相关触发条件以及在线编写自动化部署脚本，实现开发运维一体化，强有力地促进了敏捷开发。



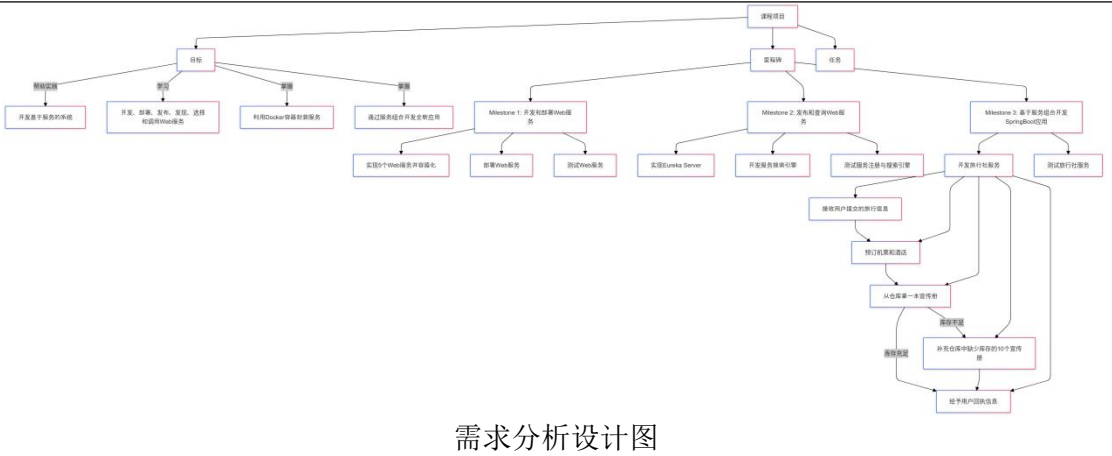

流水线自动触发条件



流水线运行

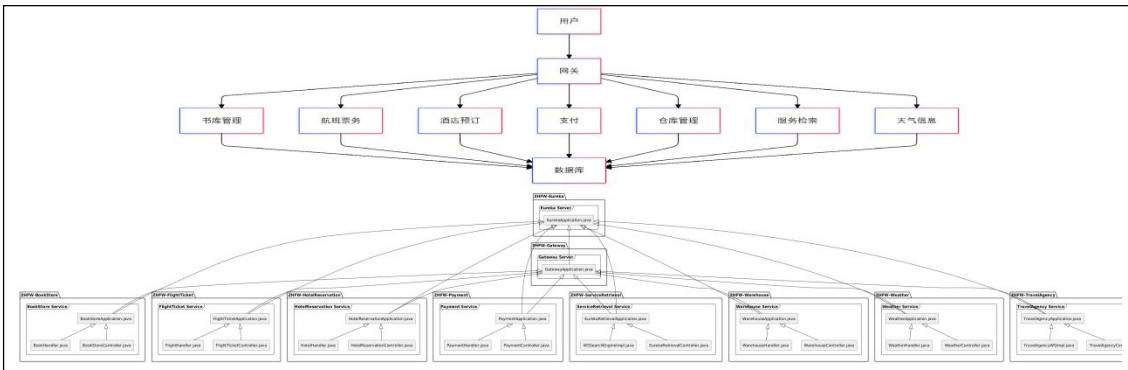
# 三、需求分析图

针对大作业要求中提到的各种需求进行精简细化，形成了需求分析设计图。


需求分析设计图

# 四、系统架构设计图

组长刘博文作为团队的架构设计师，在完成 Docker 服务架构设计的基础上，额外设计了微服务架构以实现接口请求的统一，并进行自动化部署，设计了微服务的架构。

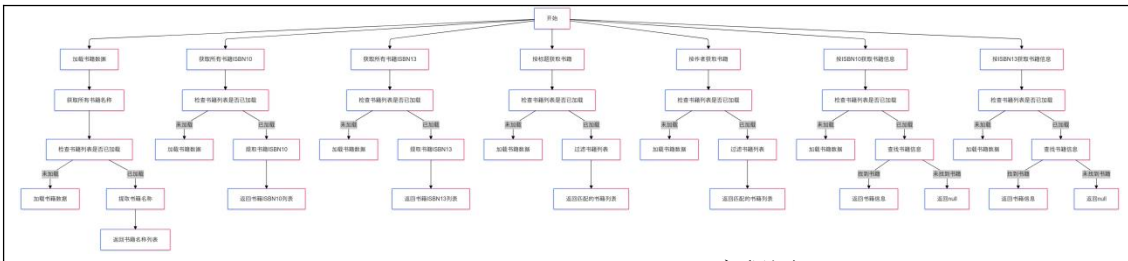


微服务架构设计图

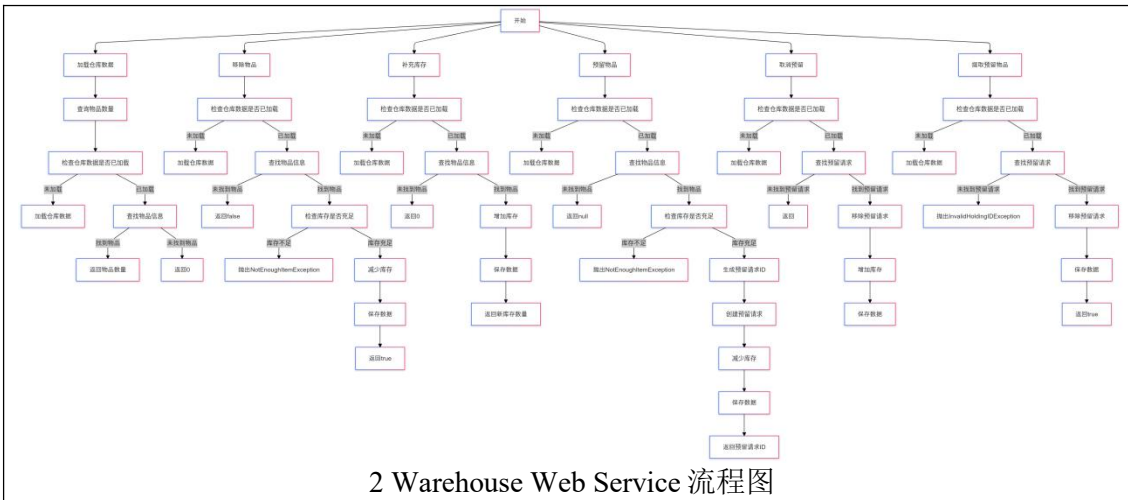
# 五、场景流程图

场景流程图是一种图形化工具，用于描述特定场景或业务流程中的步骤和决策点。它们帮助团队理解复杂流程的逻辑，识别潜在的瓶颈和改进点，并确保所有相关人员对流程有共同的理解。为了对需求进行更详尽的分析，绘制了各个场景的流程图，以辅助代码的编写。

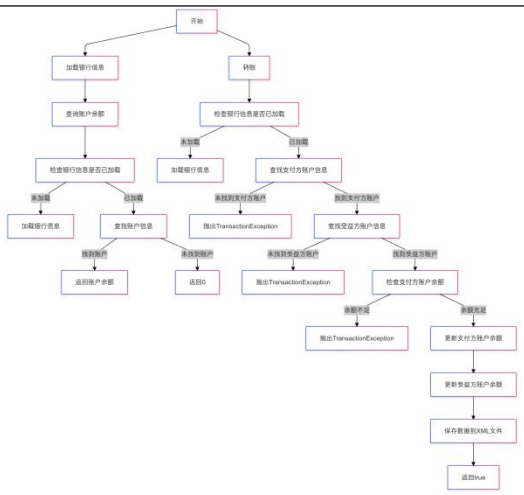
## 5.1 Book Category Web Service 流程图


1 Book Category Web Service 流程图

## 5.2 Warehouse Web Service 流程图

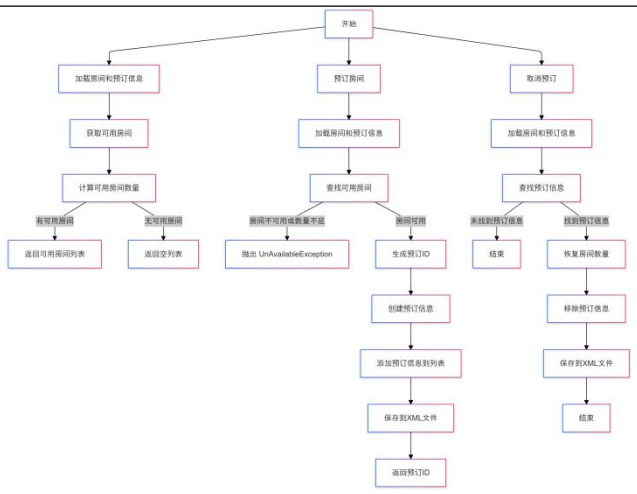

2 Warehouse Web Service 流程图

## 5.3 Payment Web Service 流程图



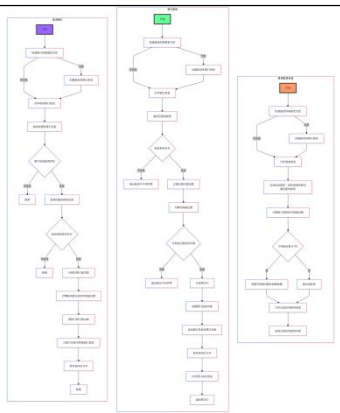

3 Payment Web Service 流程图

## 5.4 Hotel Reservation Service 流程图




4 Hotel Reservation Service 流程图

## 5.5 Flight Ticket Service 流程图

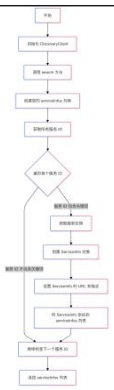


5 Flight Ticket Service 流程图

## 5.6 Weather Web Service 流程图

6 Weather Web Service 流程图

## 5.7 Service Retrieval 流程图



7 Service Retrieval 流程图

## 5.8 Service Combination 流程图

8 Service Combination 流程图

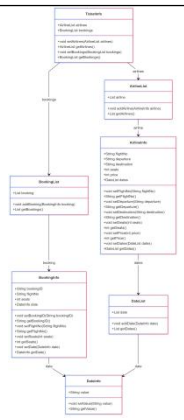
# 六、数据库类图

在数据库设计中，类图是一种常用的工具，用于展示数据类型（通常称为实体）以及这些实体之间的关系。类图可以帮助开发者和数据库管理员更好地理解数据结构和数据流。为了更加清晰的理解 xml 数据库里的数据类型和数据之间的关系，绘制了数据库类图将数据类型可视化，并定义了类中相关的方法。

## 6.1 BookStore 类图

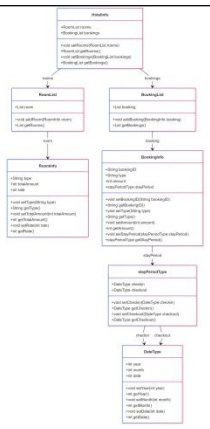
BookStore 类图

## 6.2 FlightTicket 类图



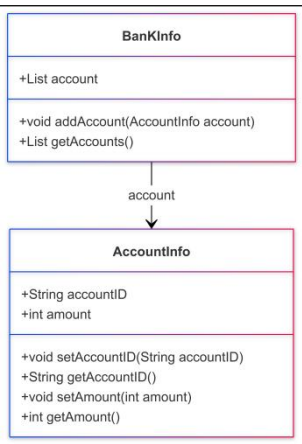
FlightTicket 类图

## 6.3 HotelReservation 类图



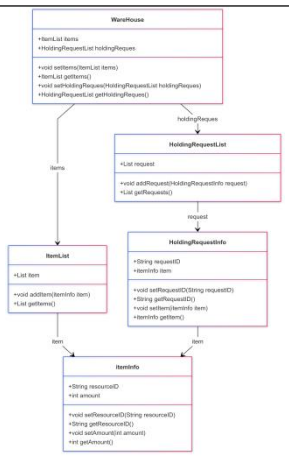
HotelReservation 类图

## 6.4 Payment 类图



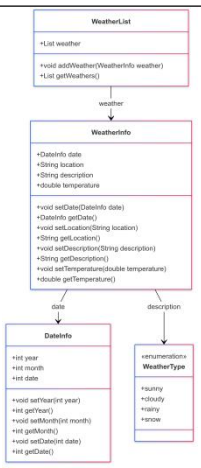

Payment 类图

## 6.5 Warehouse 类图




Warehouse 类图

## 6.6 Weather 类图




Weather 类图

# 七、Docker 部署

Docker 打包部署参考了大作业的教程中的第二种方案，通过在 maven 的 pom 依赖中引入 Docker 插件并进行相关配置，实现打包时根据编写的 Dockerfile 自动生成镜像并上传至服务器，再通过批量部署脚本进行容器的创建。

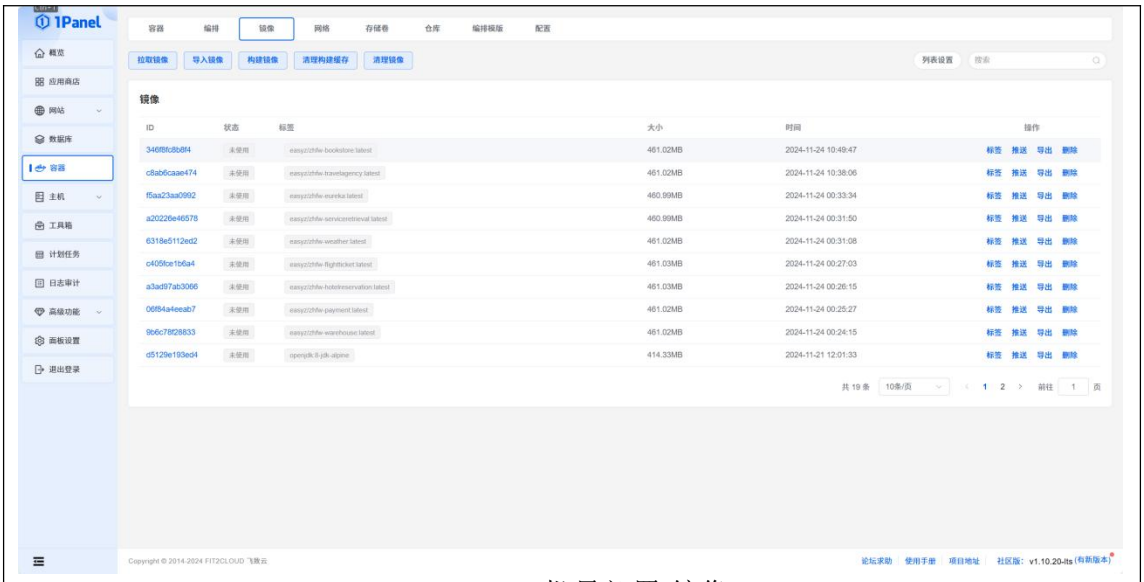
# 7.1 Docker 部署文件与步骤

容器创建遵循以下步骤：

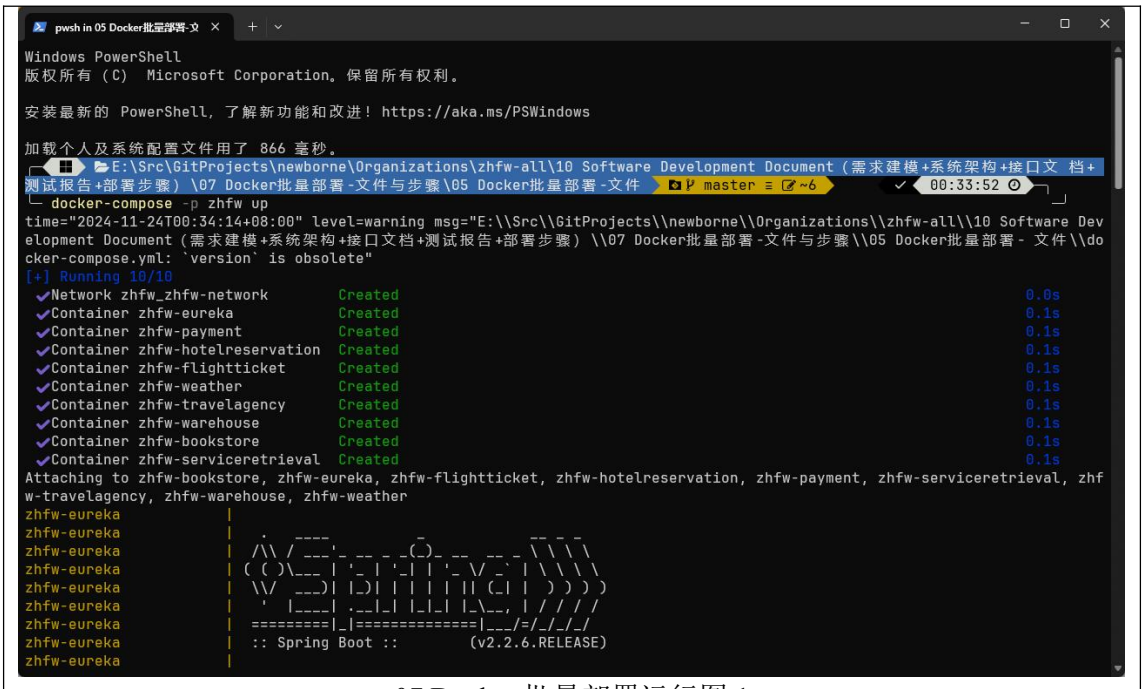
1. 对每个服务进行 maven clean install（pom 文件里已经引入 docker 打包依赖，会自动根据 Dockerfile 生成指定 Docker 镜像并上传至指定的服务器）

2. 将 docker-compose.yml 上传到服务器

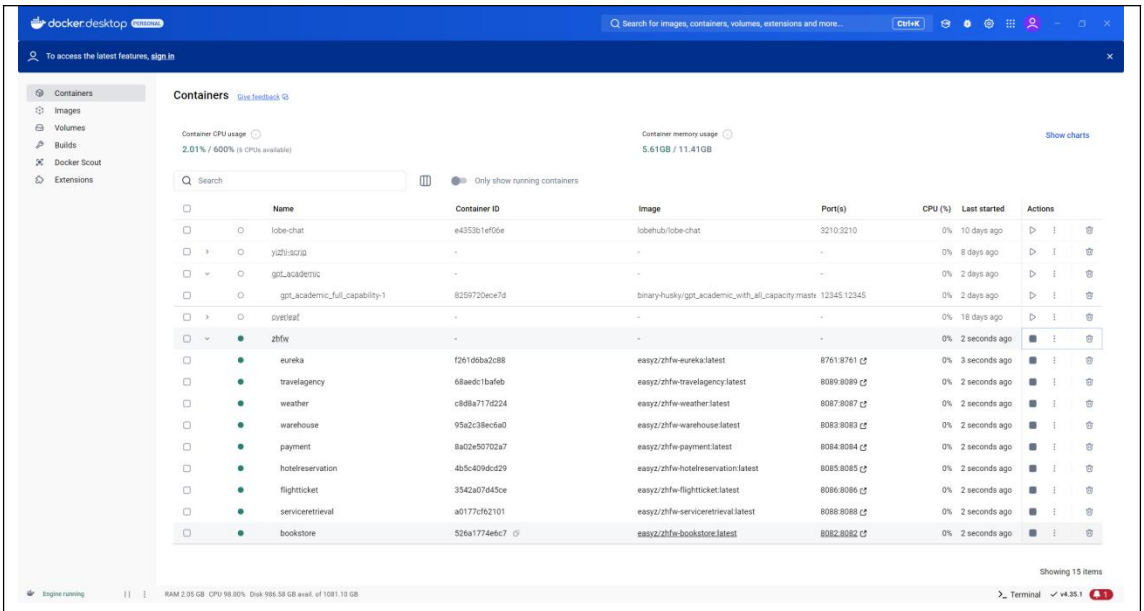
3. 在指定服务器终端内执行 docker-compose -p zhfw up




07 Docker 批量部署-镜像



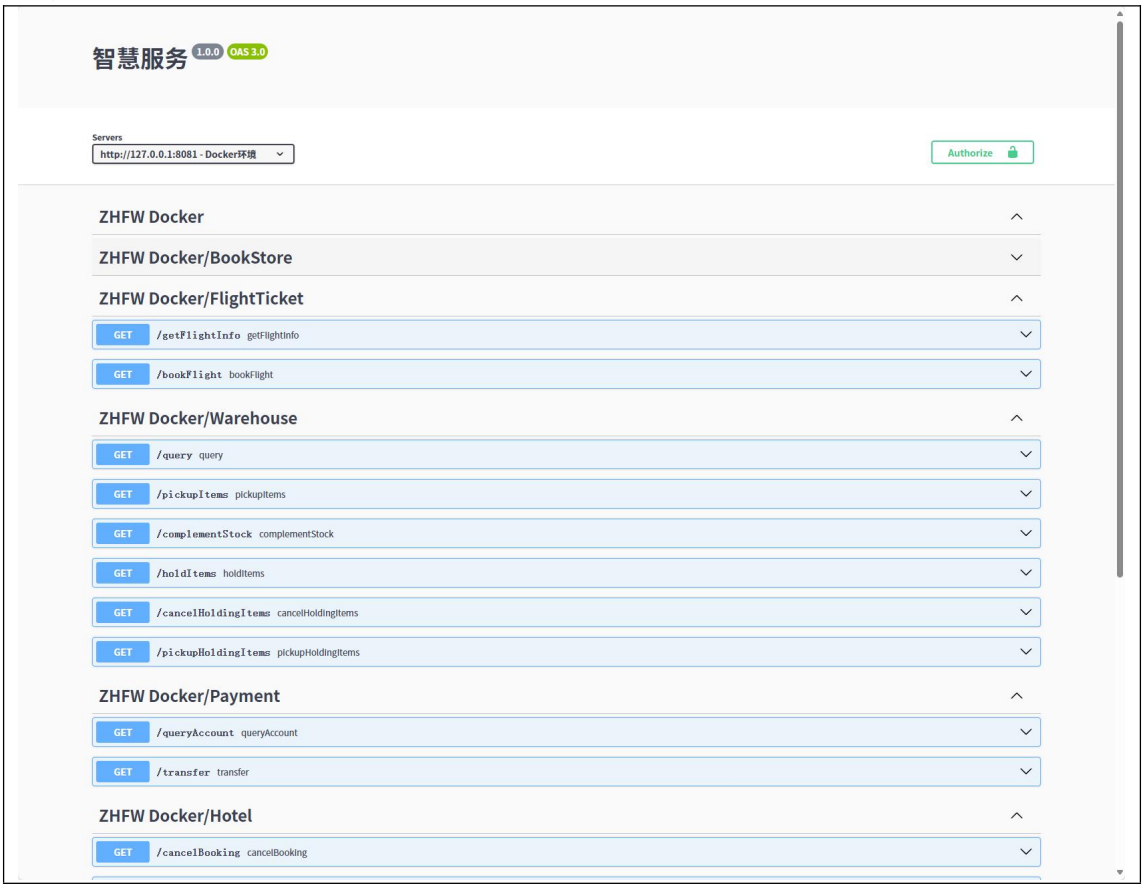

07 Docker 批量部署运行图 1




07 Docker 批量部署运行图 2

## 7.2 Docker 部署接口文档

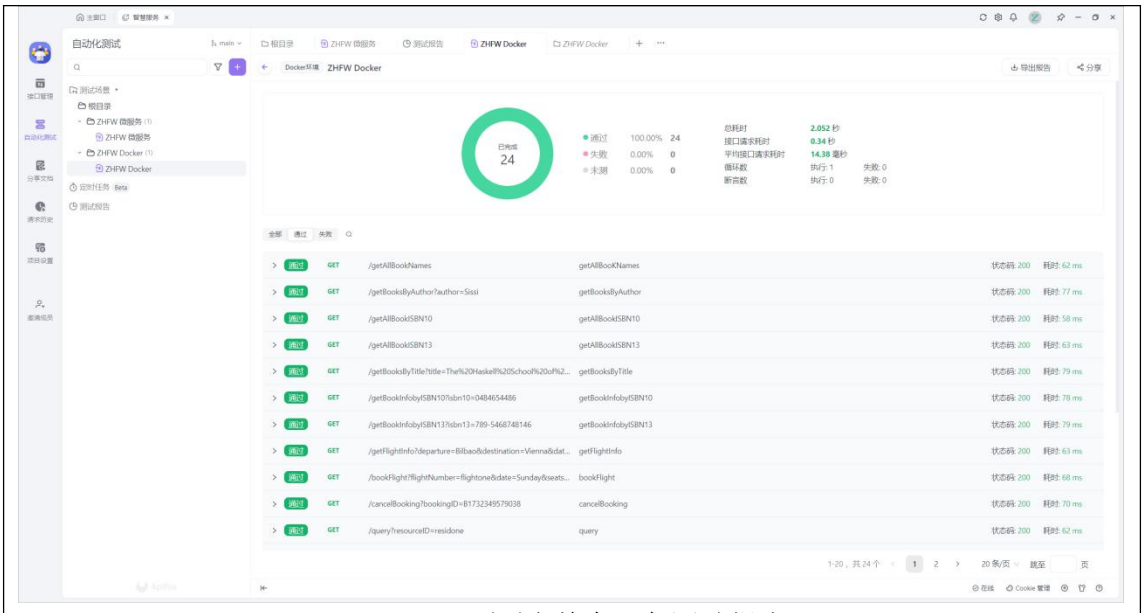
编写了 Docker 项目的接口文档，定义了各个接口的请求方式和具体要求，以及请求样例。



## 7.3 Docker 部署测试

在这一部分，针对单个服务和整合服务分别进行了测试，对需求进行验证。

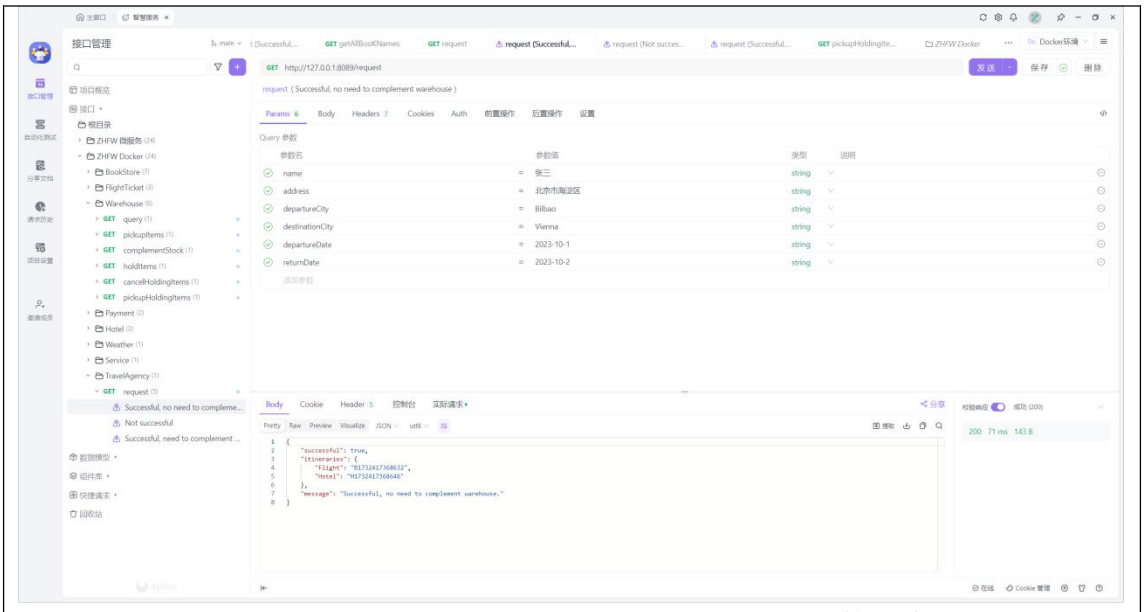
### 7.3.1 单个服务测试



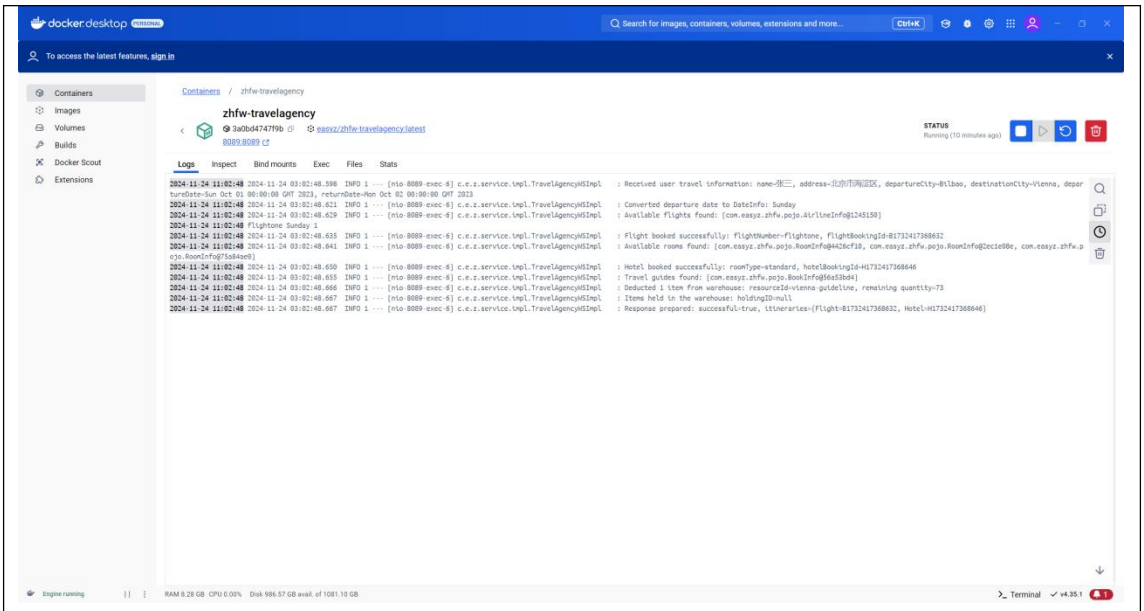
05 Docker 部署-单个服务测试报告

### 7.3.2 整合服务测试

#### ①场景一-Successful, no need to complement warehouse

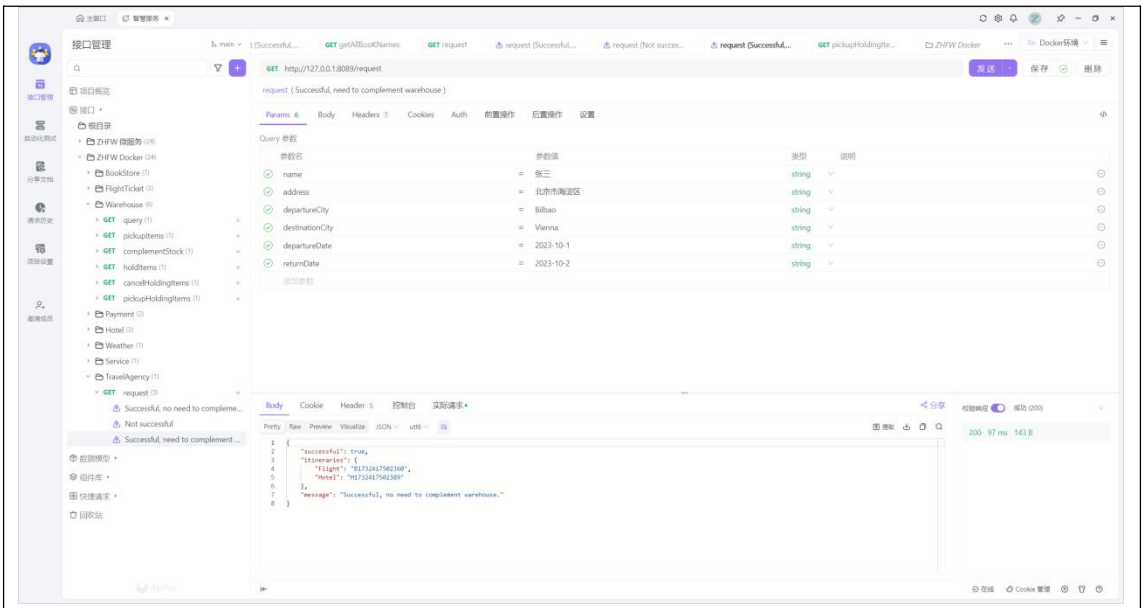


场景一-Successful, no need to complement warehouse-接口响应

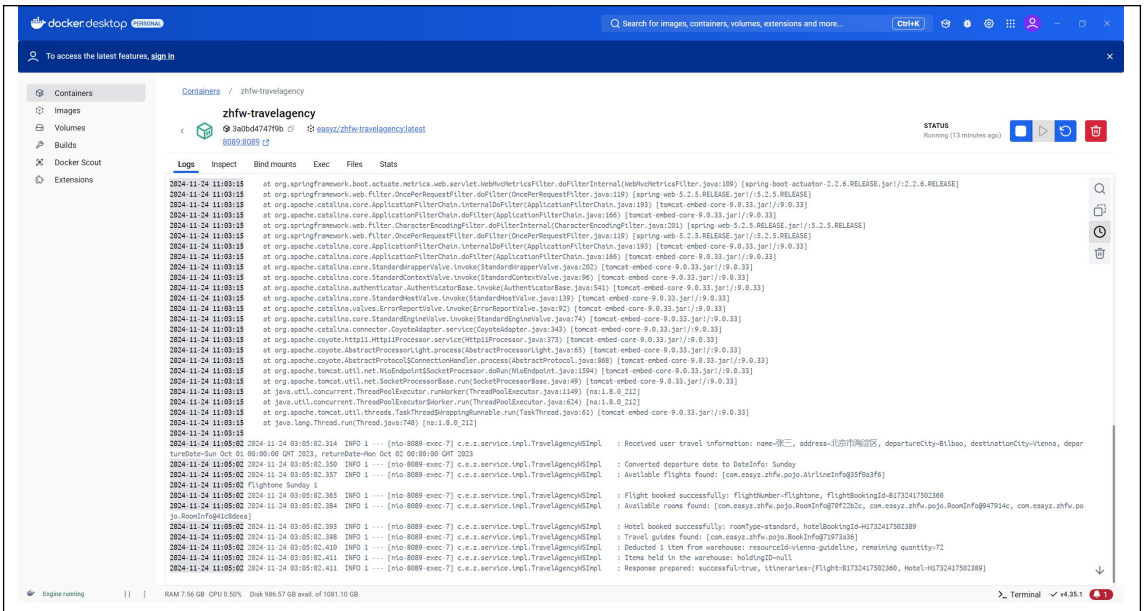


场景一-Successful, no need to complement warehouse-日志

## ②场景二-Successful, need to complement warehouse

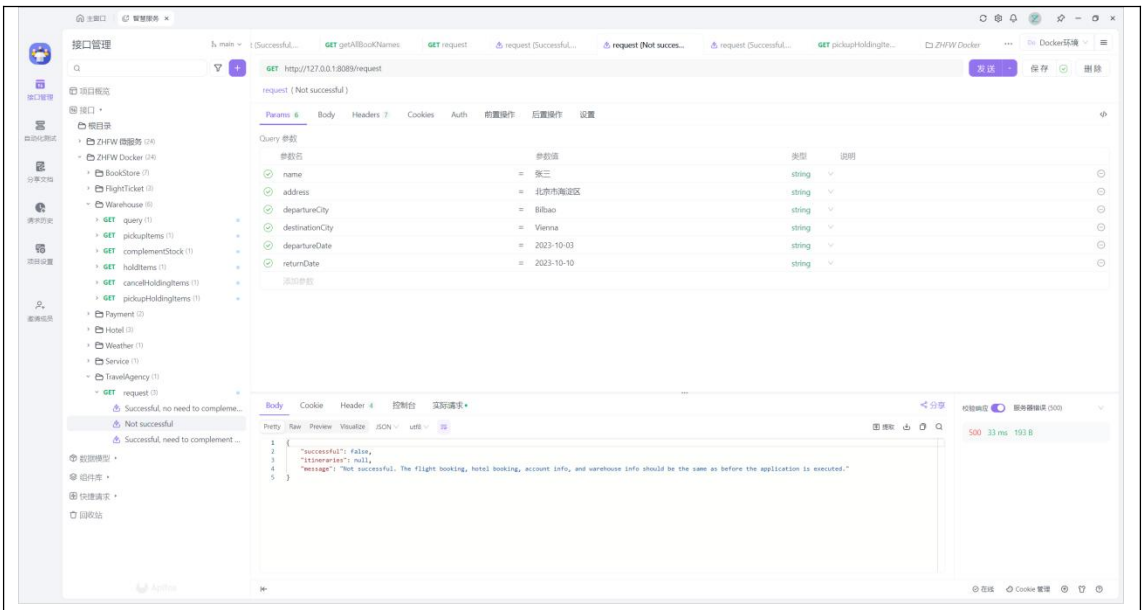


场景二-Successful, need to complement warehouse-接口响应

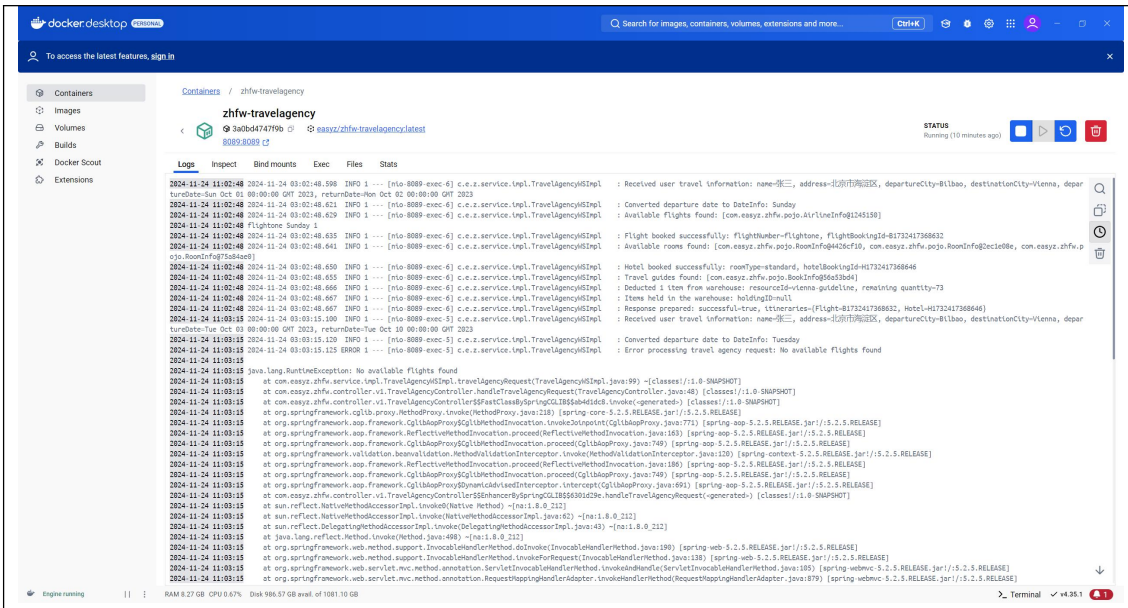


场景二-Successful, need to complement warehouse-日志

## ③场景三-Not successful



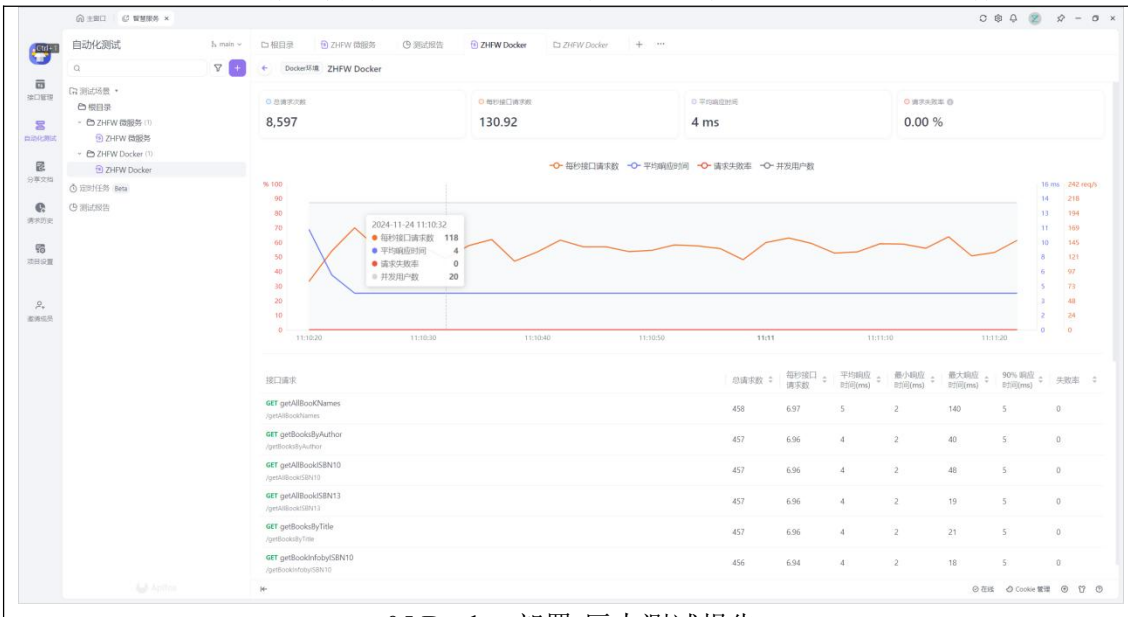
场景三-Not successful-接口响应



场景三-Not successful-日志

### 7.3.3 压力测试

针对 Gitee Go+微服务部署项目进行了压力测试，检测了接口请求的鲁棒性。



05 Docker 部署-压力测试报告

## 八、Gitee Go+微服务自动化部署

Gitee Go 是 Gitee 推出的一款企业级 CI/CD 流水线工具，它提供持续集成和持续部署的能力，帮助企业提升应用交付的质量和效率，于是借助 Gitee Go 工具实现了微服务项目的自动化部署。

# 8.1 Gitee Go+微服务自动化部署文件与步骤

Gitee Go 自动化部署遵循以下步骤：

1. Gitee Go——新建主机组——设置 linux 服务器

2. 新建流水线——选择代码仓库——设置触发条件

3. 编写编译脚本

```bash
# 功能：打包
# 参数说明：
#   -Dmaven.test.skip=true：跳过单元测试
#   -U：每次构建检查依赖更新，可避免缓存中快照版本依赖不更新问题，但会牺牲部分性能
#   -e -X ：打印调试信息，定位疑难构建问题时建议使用此参数构建
#   -B：以 batch 模式运行，可避免日志打印时出现 ArrayIndexOutOfBoundsException 异常
# 使用场景：打包项目且不需要执行单元测试时使用
mvn clean package -Dmaven.test.skip=true -U -e -X -B

# 功能：自定义 settings 配置
# 使用场景：如需手工指定 settings.xml，可使用如下方式
# 注意事项：如无需自定义 settings 配置且需要私有依赖仓库，可在该任务配置《私有仓库》处添加私有依赖
# mvn -U clean package -s ./settings.xml

```

4. 编写部署脚本

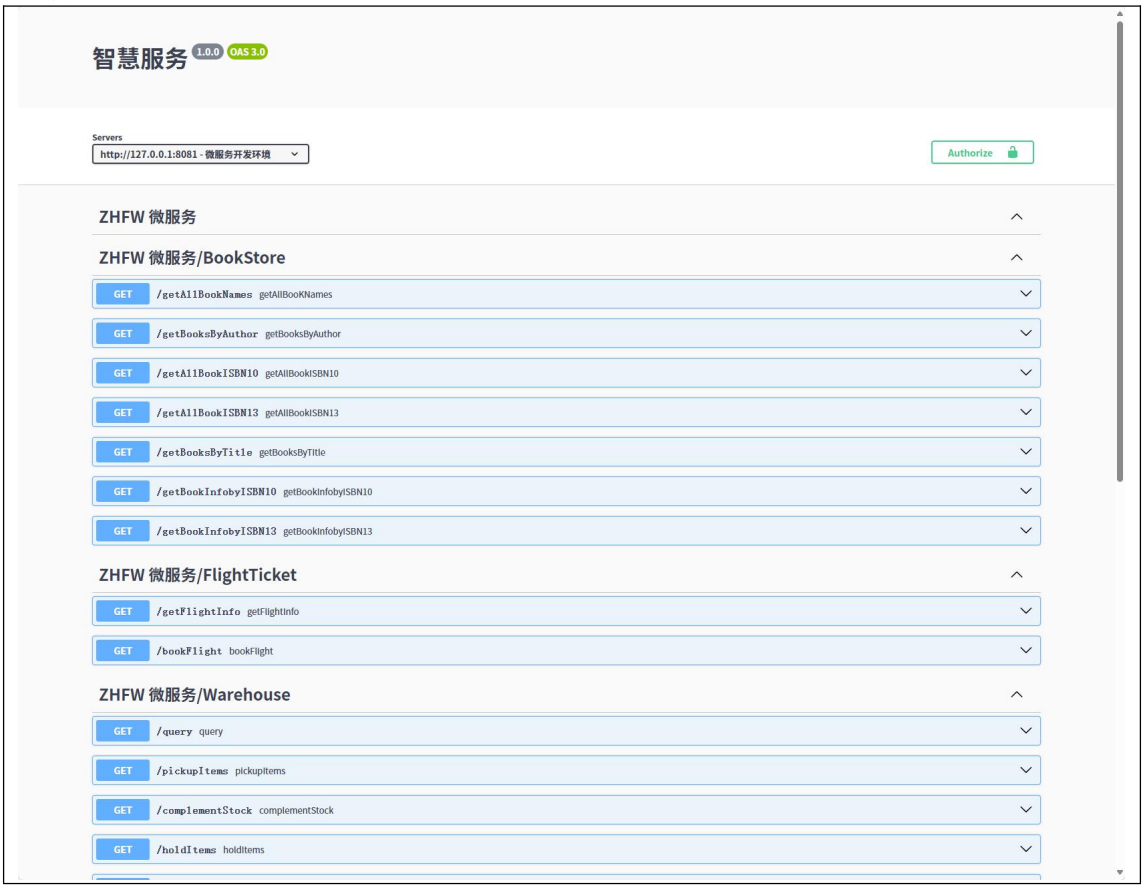
```bash
# 请在此输入部署脚本，如启动 Java 应用如下
# nohup java -jar test.jar &gt; nohup.out &amp;
#  echo 'Hello Gitee!'


cd /home/newborne/gitee_go/deploy

# 打印当前目录
echo "当前目录是: $(pwd)"
# 解压缩文件
echo "正在解压缩 $TAR_FILE ..."
tar -zxvf zhfw.tar.gz


# 启动所有微服务
echo "正在启动微服务 ..."

# 启动每个 JAR 包
for jar in ZHFW-*.jar; do
```

```
    echo "启动 $jar ..."
    nohup java -jar "$jar" > "${jar%.jar}.out" 2>&1 &
done

echo "所有微服务已启动！"

```

5. 自动化运行

## 8.2 Gitee Go+微服务自动化部署接口文档

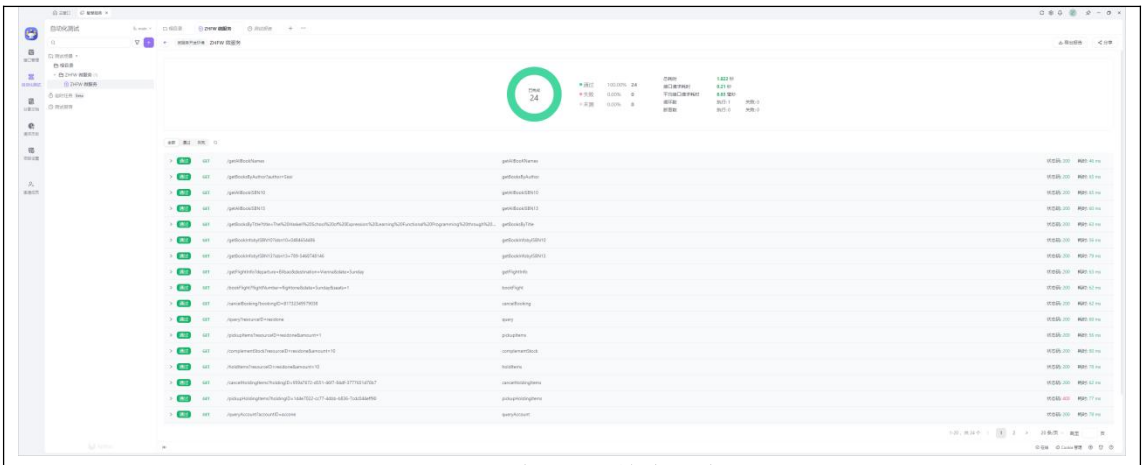
编写了微服务项目的接口文档，定义了各个接口的请求方式和具体要求，以及请求样例。




## 8.3 Gitee Go+微服务自动化部署测试

在这一部分，针对单个服务和整合服务分别进行了测试，对需求进行验证。

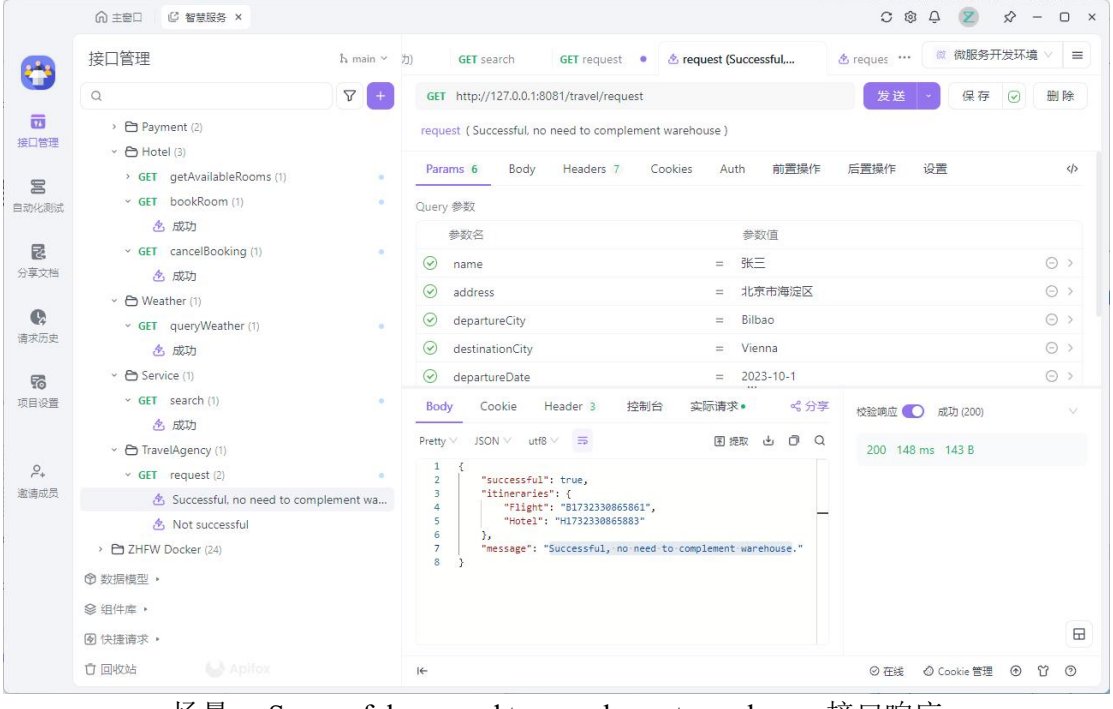
### 8.3.1 单个服务测试



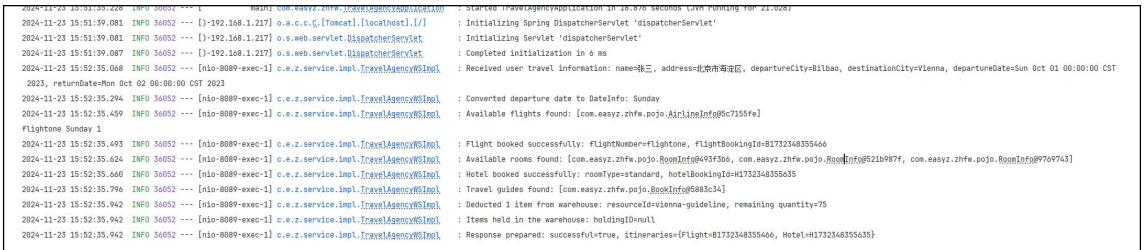
06 Gitee Go+微服务部署-单个服务测试报告

### 8.3.2 整合服务测试

#### ①场景一-Successful, no need to complement warehouse

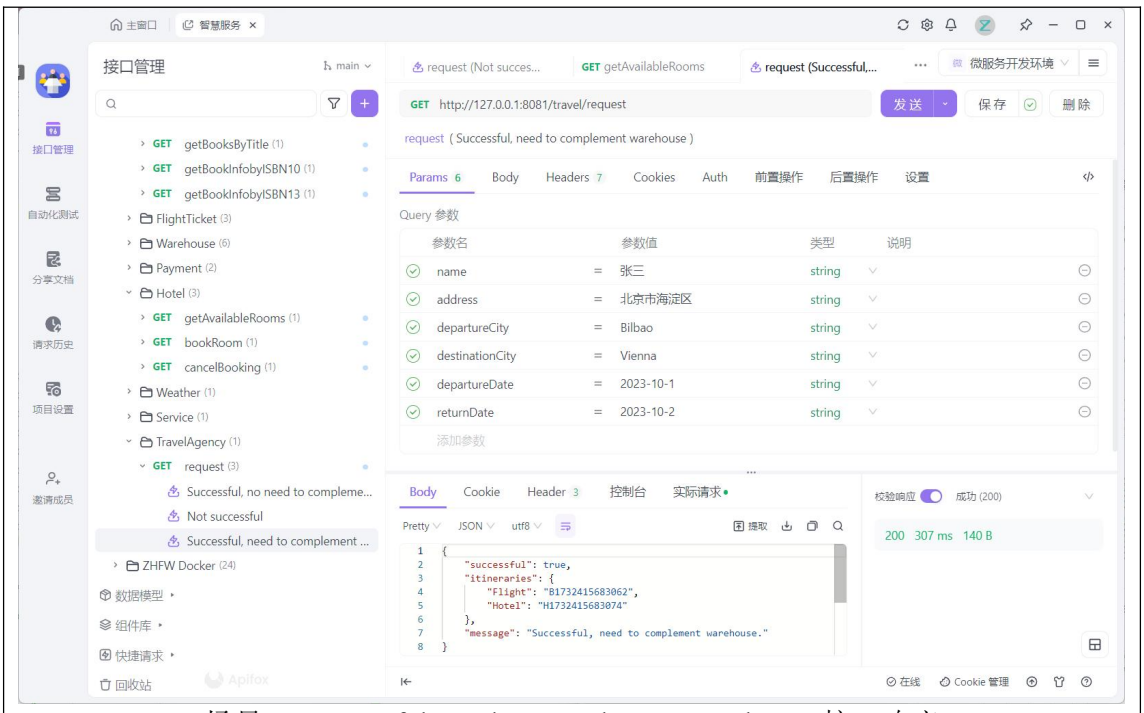



场景一-Successful, no need to complement warehouse-接口响应




场景一-Successful, no need to complement warehouse-日志

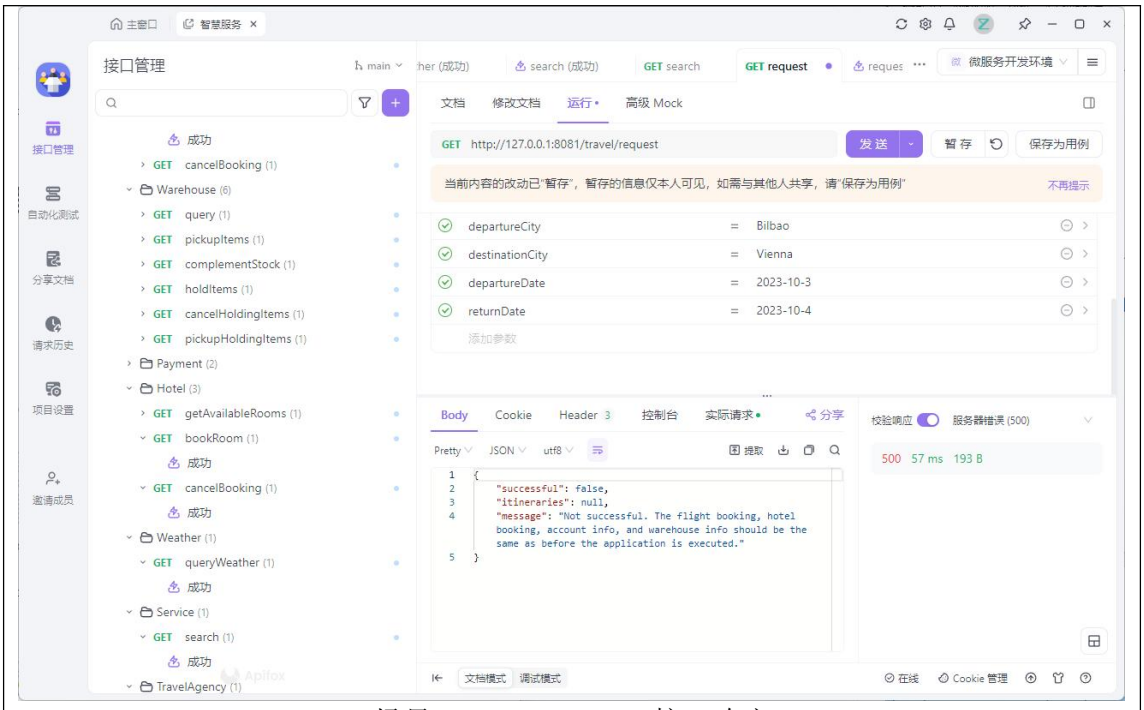
## ②场景二-Successful, need to complement warehouse



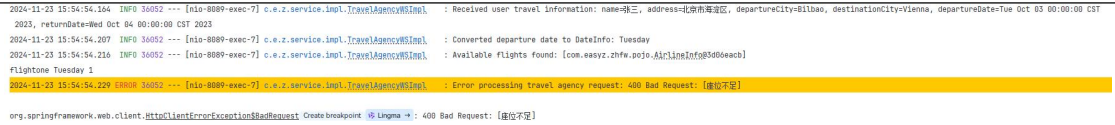

场景二-Successful, need to complement warehouse-接口响应


场景二-Successful, need to complement warehouse-日志

## ③场景三-Not successful




场景三-Not successful-接口响应




场景三-Not successful-日志

## 8.3.3 压力测试

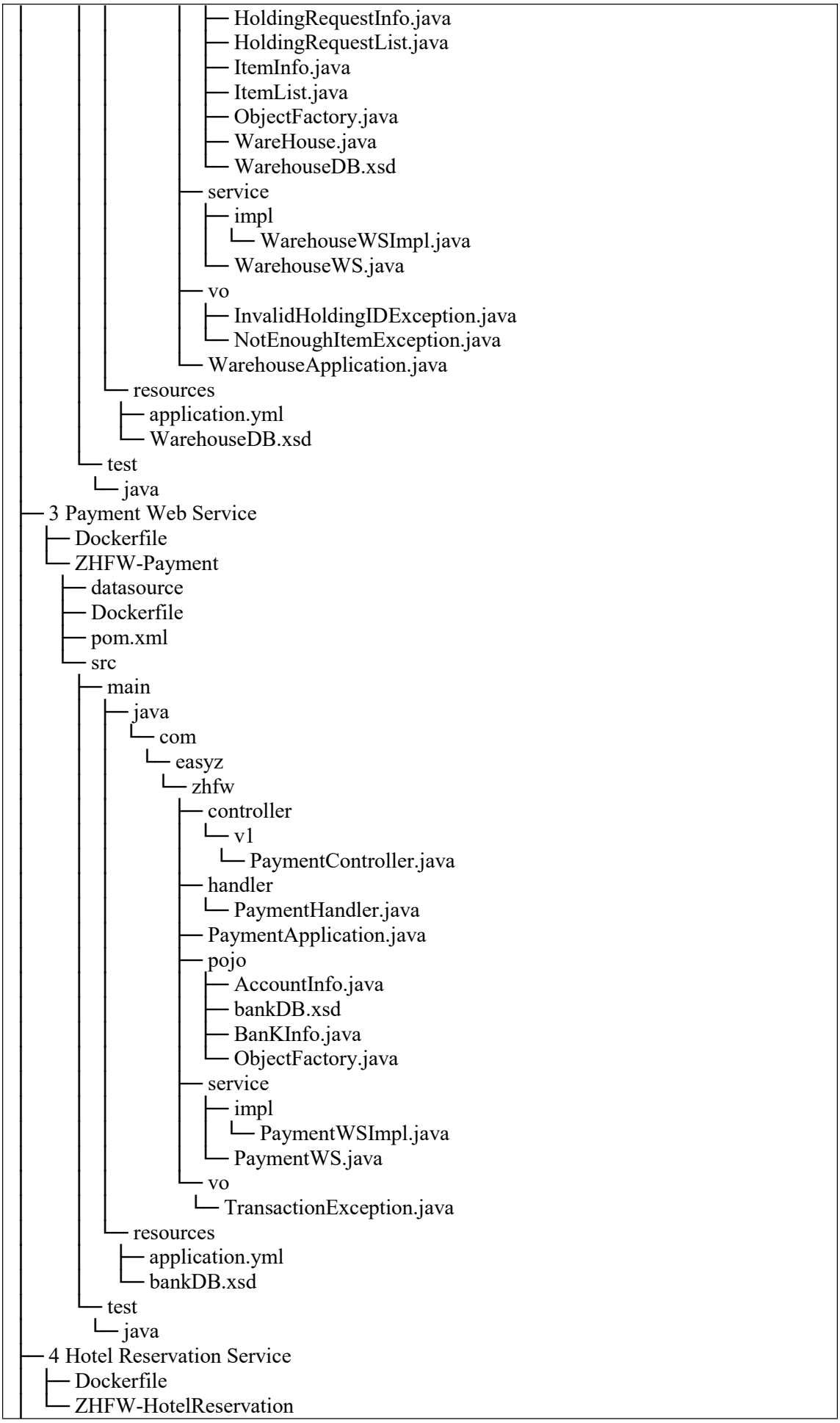
同样，也针对 Gitee Go+微服务部署项目进行了压力测试，检测了接口请求的鲁棒性。

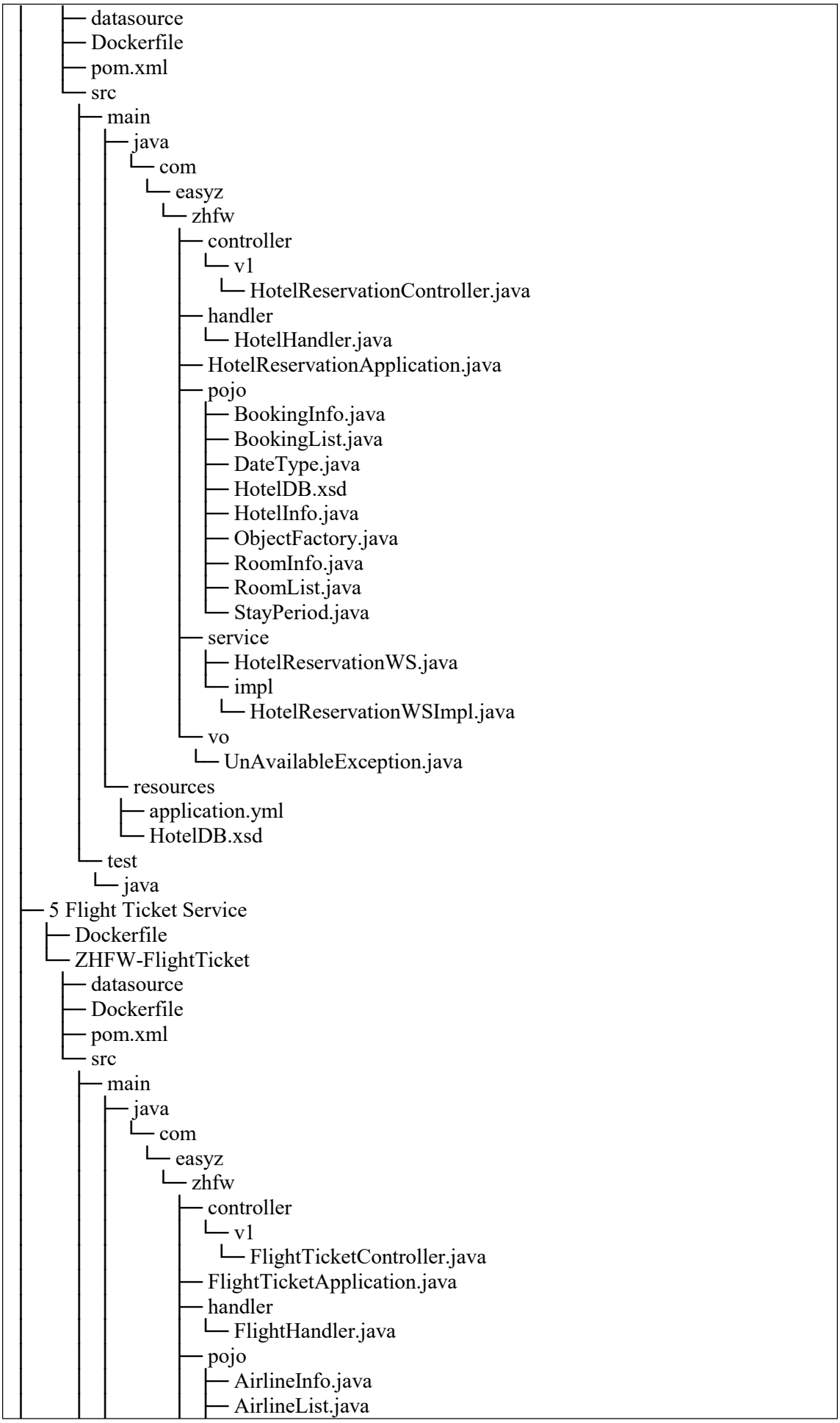

06 Gitee Go+微服务部署-压力测试报告

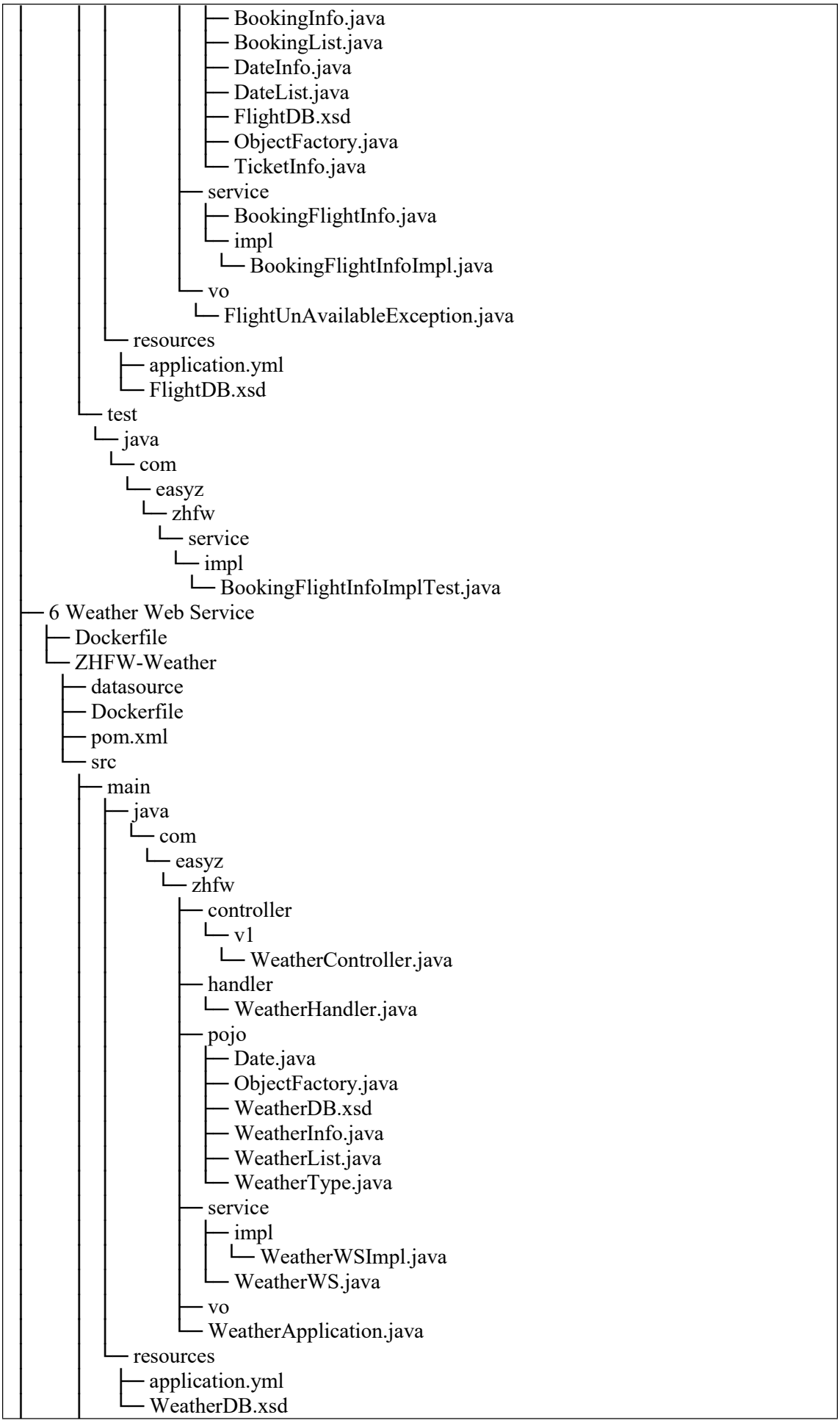
# 九、附录：项目目录结构

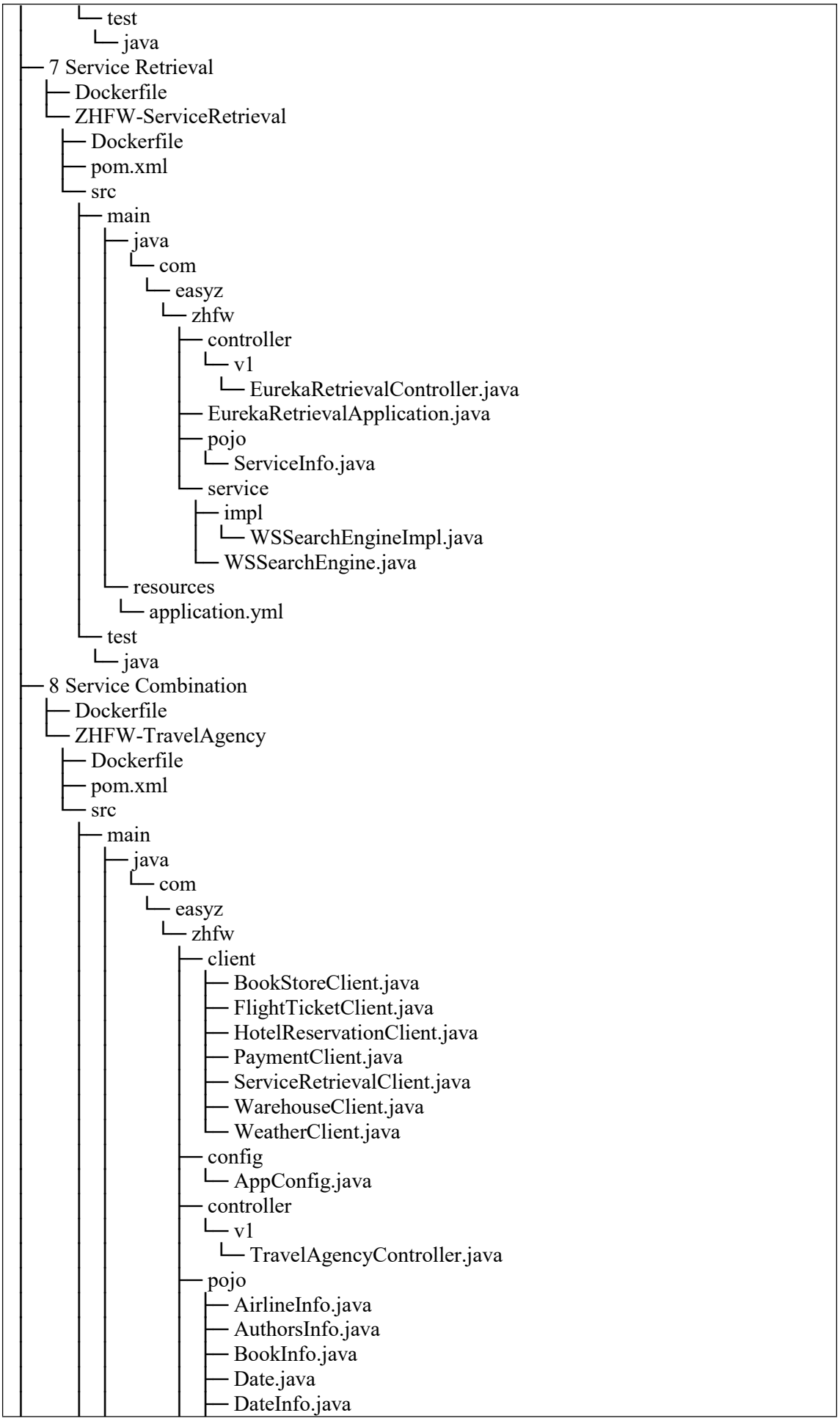
以下是项目的目录结构：

```
zhfw-all
├── 1 Book Category Web Service
│   ├── Dockerfile
│   └── ZHFW-BookStore
│       ├── datasource
│       ├── Dockerfile
│       ├── pom.xml
│       └── src
│           ├── main
│           │   ├── java
│           │   │   └── com
│           │   │       └── easyz
│           │   │           └── zhfw
│           │   │               ├── BookStoreApplication.java
│           │   │               ├── controller
│           │   │               │   └── v1
│           │   │               │       └── BookStoreController.java
│           │   │               ├── handler
│           │   │               │   └── BookHandler.java
│           │   │               ├── pojo
│           │   │               │   ├── AuthorsInfo.java
│           │   │               │   ├── BookInfo.java
│           │   │               │   └── BookInfo_Schema.json
│           │   │               ├── service
│           │   │               │   ├── BookCategoryWS.java
│           │   │               │   └── impl
│           │   │               │       └── BookCategoryWSImpl.java
│           │   │               └── vo
│           │   │                   └── UnAvailableException.java
│           │   └── resources
│           │       ├── application.yml
│           │       └── BookCategoryDB.xsd
│           └── test
│               └── java
├── 2 Warehouse Web Service
│   ├── Dockerfile
│   └── ZHFW-Warehouse
│       ├── datasource
│       ├── Dockerfile
│       ├── pom.xml
│       └── src
│           ├── main
│           │   ├── java
│           │   │   └── com
│           │   │       └── easyz
│           │   │           └── zhfw
│           │   │               ├── controller
│           │   │               │   └── v1
│           │   │               │       └── WarehouseController.java
│           │   │               ├── handler
│           │   │               │   └── WarehouseHandler.java
│           │   │               ├── pojo
```


HoldingRequestInfo.java
HoldingRequestList.java
ItemInfo.java
ItemList.java
ObjectFactory.java
WareHouse.java
WarehouseDB.xsd
service
impl
WarehouseWSImpl.java
WarehouseWS.java
vo
InvalidHoldingIDException.java
NotEnoughItemException.java
WarehouseApplication.java
resources
application.yml
WarehouseDB.xsd
test
java
3 Payment Web Service
Dockerfile
ZHFW-Payment
datasource
Dockerfile
pom.xml
src
main
java
com
easyz
zhfw
controller
v1
PaymentController.java
handler
PaymentHandler.java
PaymentApplication.java
pojo
AccountInfo.java
bankDB.xsd
BanKInfo.java
ObjectFactory.java
service
impl
PaymentWSImpl.java
PaymentWS.java
vo
TransactionException.java
resources
application.yml
bankDB.xsd
test
java
4 Hotel Reservation Service
Dockerfile
ZHFW-HotelReservation


datasource
Dockerfile
pom.xml
src
main
java
com
easyz
zhfw
controller
v1
HotelReservationController.java
handler
HotelHandler.java
HotelReservationApplication.java
pojo
BookingInfo.java
BookingList.java
DateType.java
HotelDB.xsd
HotelInfo.java
ObjectFactory.java
RoomInfo.java
RoomList.java
StayPeriod.java
service
HotelReservationWS.java
impl
HotelReservationWSImpl.java
vo
UnAvailableException.java
resources
application.yml
HotelDB.xsd
test
java
5 Flight Ticket Service
Dockerfile
ZHFW-FlightTicket
datasource
Dockerfile
pom.xml
src
main
java
com
easyz
zhfw
controller
v1
FlightTicketController.java
FlightTicketApplication.java
handler
FlightHandler.java
pojo
AirlineInfo.java
AirlineList.java

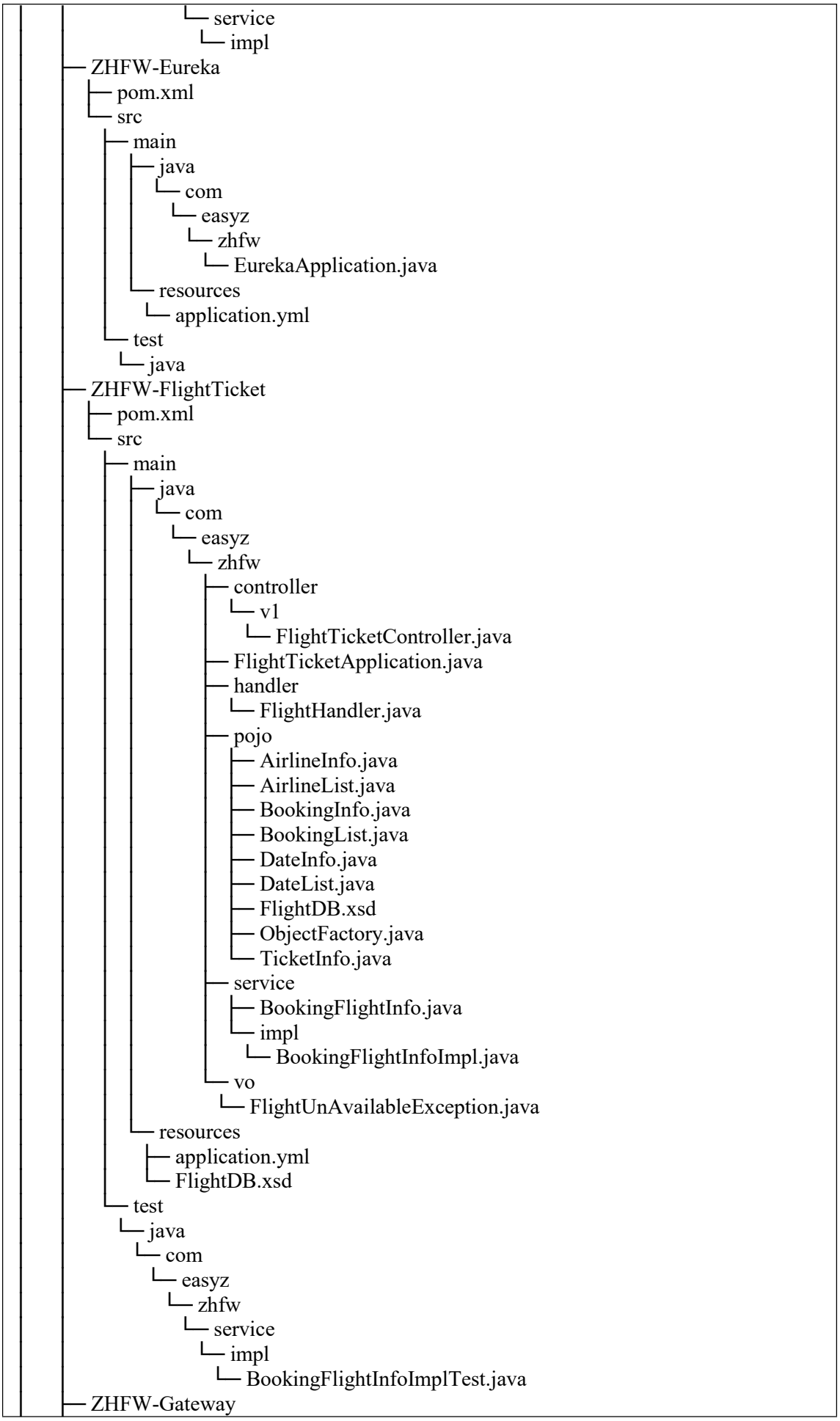

BookingInfo.java
BookingList.java
DateInfo.java
DateList.java
FlightDB.xsd
ObjectFactory.java
TicketInfo.java
service
BookingFlightInfo.java
impl
BookingFlightInfoImpl.java
vo
FlightUnAvailableException.java
resources
application.yml
FlightDB.xsd
test
java
com
easyz
zhfw
service
impl
BookingFlightInfoImplTest.java
6 Weather Web Service
Dockerfile
ZHFW-Weather
datasource
Dockerfile
pom.xml
src
main
java
com
easyz
zhfw
controller
v1
WeatherController.java
handler
WeatherHandler.java
pojo
Date.java
ObjectFactory.java
WeatherDB.xsd
WeatherInfo.java
WeatherList.java
WeatherType.java
service
impl
WeatherWSImpl.java
WeatherWS.java
vo
WeatherApplication.java
resources
application.yml
WeatherDB.xsd

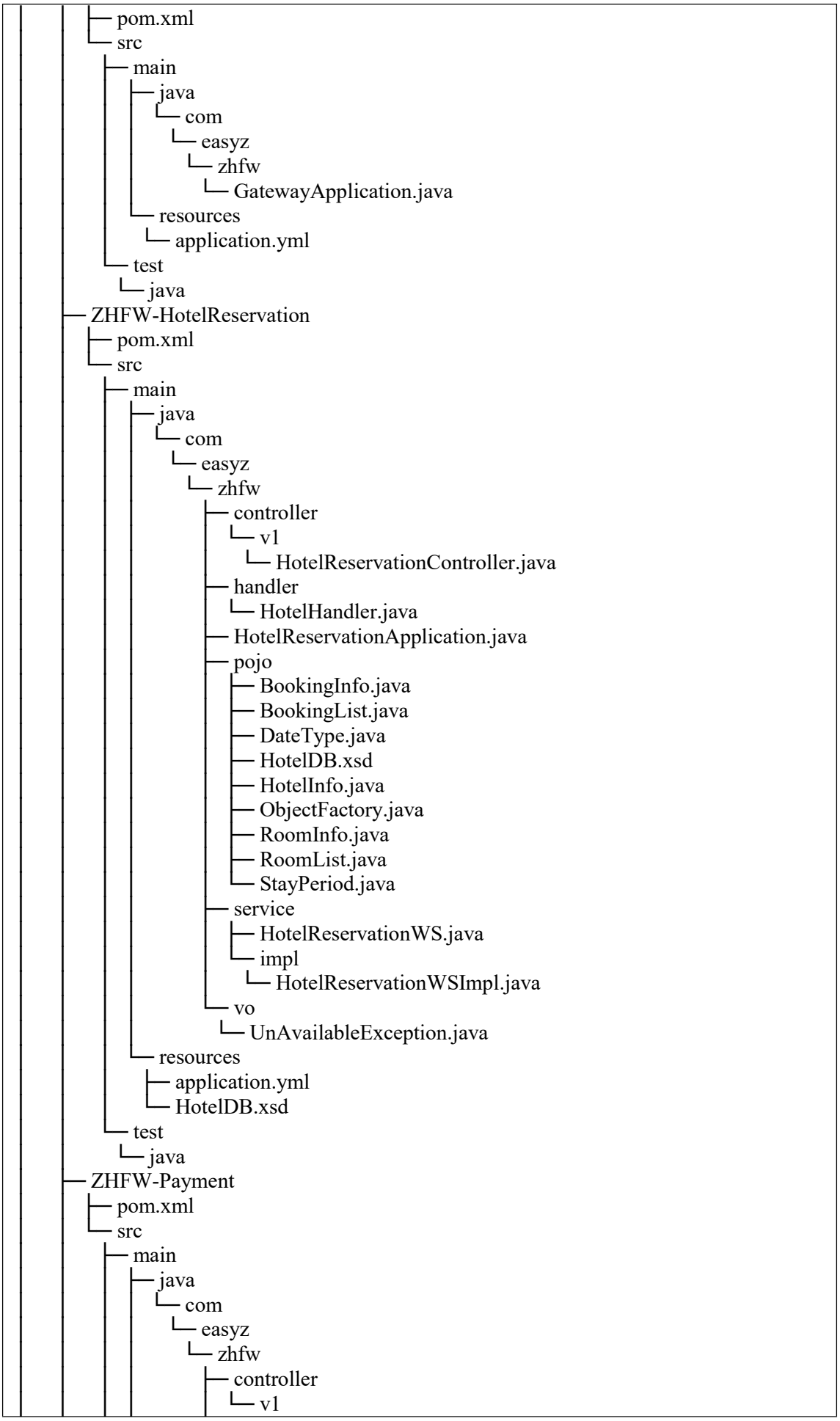

test
java
7 Service Retrieval
Dockerfile
ZHFW-ServiceRetrieval
Dockerfile
pom.xml
src
main
java
com
easyz
zhfw
controller
v1
EurekaRetrievalController.java
EurekaRetrievalApplication.java
pojo
ServiceInfo.java
service
impl
WSSearchEngineImpl.java
WSSearchEngine.java
resources
application.yml
test
java
8 Service Combination
Dockerfile
ZHFW-TravelAgency
Dockerfile
pom.xml
src
main
java
com
easyz
zhfw
client
BookStoreClient.java
FlightTicketClient.java
HotelReservationClient.java
PaymentClient.java
ServiceRetrievalClient.java
WarehouseClient.java
WeatherClient.java
config
AppConfig.java
controller
v1
TravelAgencyController.java
pojo
AirlineInfo.java
AuthorsInfo.java
BookInfo.java
Date.java
DateInfo.java

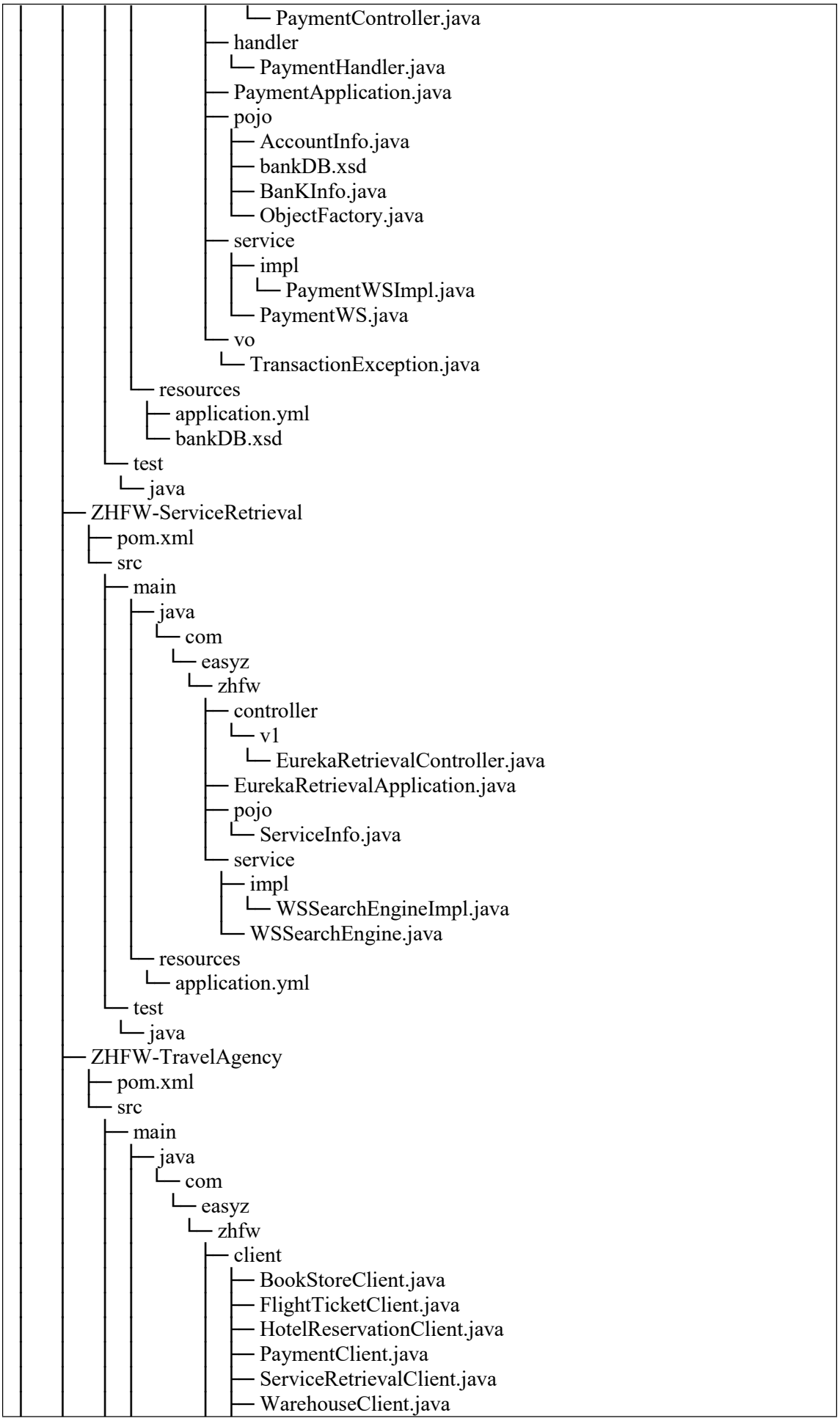
```
│   │   │   │   │   ├── DateList.java
│   │   │   │   │   ├── DateType.java
│   │   │   │   │   ├── RoomInfo.java
│   │   │   │   │   ├── ServiceInfo.java
│   │   │   │   │   ├── StayPeriod.java
│   │   │   │   │   ├── WeatherInfo.java
│   │   │   │   │   └── WeatherType.java
│   │   │   │   ├── service
│   │   │   │   │   ├── impl
│   │   │   │   │   │   └── TravelAgencyWSImpl.java
│   │   │   │   │   └── TravelAgencyWS.java
│   │   │   │   ├── TravelAgencyApplication.java
│   │   │   │   └── vo
│   │   │   │       ├── TravelAgencyRequestType.java
│   │   │   │       └── TravelAgencyResponseType.java
│   │   │   └── resources
│   │   │       └── application.yml
│   │   └── test
│   │       └── java
├── 9 Eureka
│   ├── Dockerfile
│   └── ZHFW-Eureka
│       ├── Dockerfile
│       ├── pom.xml
│       └── src
│           ├── main
│           │   ├── java
│           │   │   └── com
│           │   │       └── easyz
│           │   │           └── zhfw
│           │   │               └── EurekaApplication.java
│           │   └── resources
│           │       └── application.yml
│           └── test
│               └── java
├── 10 Software Development Document（需求建模+系统架构+接口文档+测试报告+部署步骤）
│   ├── 00 项目管理——Gitee 企业版
│   │   ├── Gitee Go 自动化部署流水线
│   │   │   ├── 流水线自动触发条件.png
│   │   │   └── 流水线运行.png
│   │   ├── 代码仓库
│   │   │   ├── 智慧服务 Docker 部署.png
│   │   │   └── 智慧服务微服务架构+Gitee Go 自动化部署.png
│   │   ├── 项目任务
│   │   │   └── 项目任务.png
│   │   ├── 项目成员
│   │   │   └── 项目成员.png
│   │   ├── 项目版本
│   │   │   └── 项目版本.png
│   │   ├── 项目缺陷
│   │   │   └── 项目缺陷.png
│   │   ├── 项目规划
│   │   │   └── 项目规划.png
```

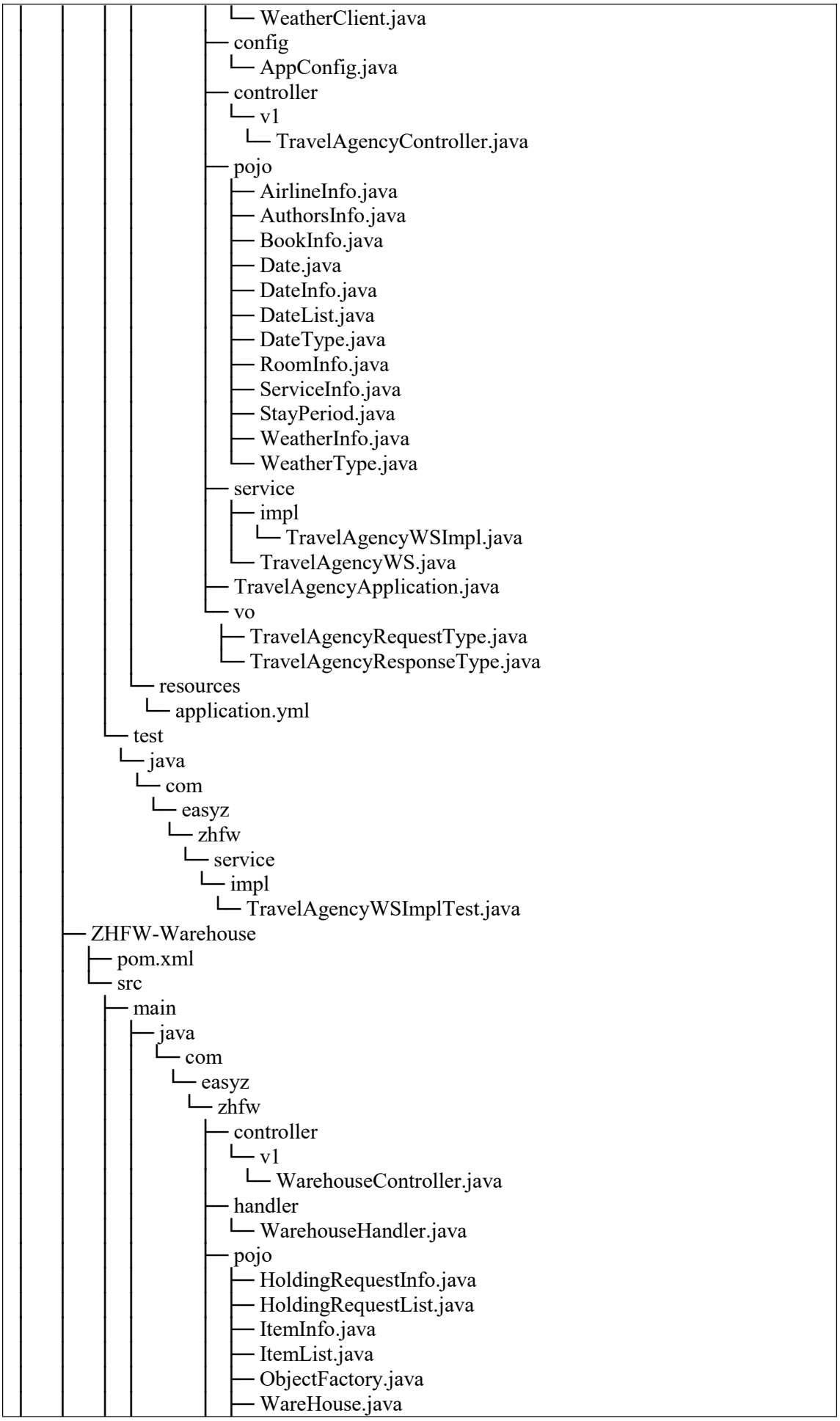
```
│   │   ├── 项目迭代
│   │   │   └── 项目迭代.png
│   │   └── 项目需求
│   │       └── 项目需求.png
│   ├── 01 ZHFW 需求分析图
│   │   └── 需求分析设计图.png
│   ├── 02 ZHFW 系统架构设计图
│   │   └── 微服务架构设计图.png
│   ├── 03 ZHFW 场景流程图
│   │   ├── 1 Book Category Web Service 流程图.png
│   │   ├── 2 Warehouse Web Service 流程图.png
│   │   ├── 3 Payment Web Service 流程图.png
│   │   ├── 4 Hotel Reservation Service 流程图.png
│   │   ├── 5 Flight Ticket Service 流程图.png
│   │   ├── 6 Weather Web Service 流程图.png
│   │   ├── 7 Service Retrieval 流程图.png
│   │   └── 8 Service Combination 流程图.png
│   ├── 04 ZHFW 数据库类图
│   │   ├── BookStore 类图.png
│   │   ├── FlightTicket 类图.png
│   │   ├── HotelReservation 类图.png
│   │   ├── Payment 类图.png
│   │   ├── Warehouse 类图.png
│   │   └── Weather 类图.png
│   ├── 05 Docker 部署-接口文档+测试报告
│   │   ├── 05 Docker 部署-单个服务测试报告
│   │   │   ├── 05 Docker 部署-单个服务测试报告.html
│   │   │   └── 05 Docker 部署-单个服务测试报告.png
│   │   ├── 05 Docker 部署-压力测试报告
│   │   │   └── 05 Docker 部署-压力测试报告.png
│   │   ├── 05 Docker 部署-接口文档.html
│   │   └── 05 Docker 部署-整合服务测试报告
│   │       ├── 场景一-Successful, no need to complement warehouse-接口响应.png
│   │       ├── 场景一-Successful, no need to complement warehouse-日志.png
│   │       ├── 场景三-Not successful-接口响应.png
│   │       ├── 场景三-Not successful-日志.png
│   │       ├── 场景二-Successful, need to complement warehouse-接口响应.png
│   │       └── 场景二-Successful, need to complement warehouse-日志.png
│   ├── 06 Gitee Go+微服务部署-接口文档+测试报告
│   │   ├── 06 Gitee Go+微服务部署-单个服务测试报告
│   │   │   ├── 06 Gitee Go+微服务部署-单个服务测试报告.html
│   │   │   └── 06 Gitee Go+微服务部署-单个服务测试报告.png
│   │   ├── 06 Gitee Go+微服务部署-压力测试报告
│   │   │   └── 06 Gitee Go+微服务部署-压力测试报告.png
│   │   ├── 06 Gitee Go+微服务部署-接口文档.html
│   │   └── 06 Gitee Go+微服务部署-整合服务测试报告
│   │       ├── 场景一-Successful, no need to complement warehouse-接口响应.png
│   │       ├── 场景一-Successful, no need to complement warehouse-日志.png
│   │       ├── 场景三-Not successful-接口响应.png
│   │       ├── 场景三-Not successful-日志.png
```

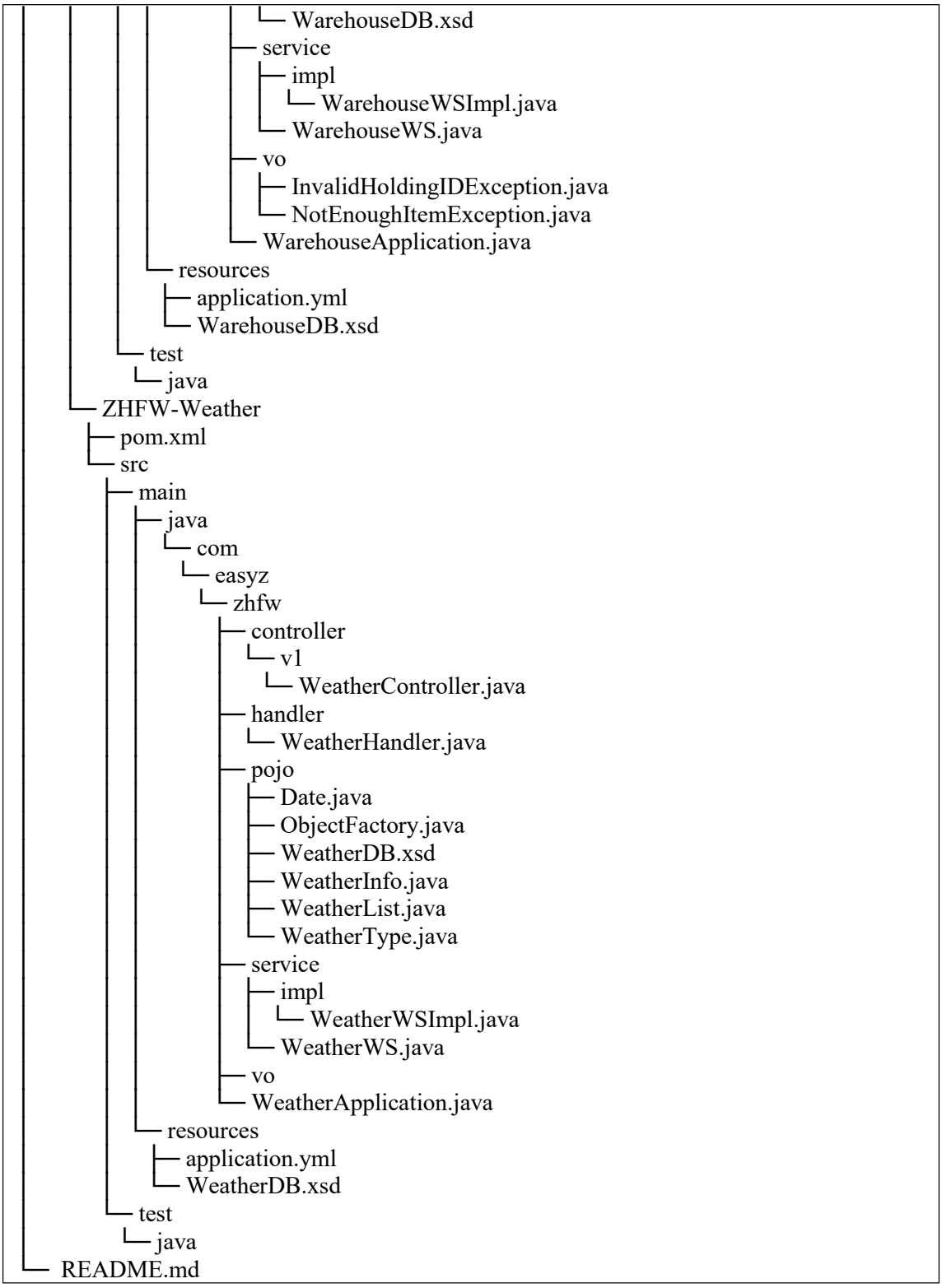
```
│   │   ├── 场景二-Successful, need to complement warehouse-接口响应.png
│   │   └── 场景二-Successful, need to complement warehouse-日志.png
│   ├── 07 Docker 批量部署-文件与步骤
│   │   ├── 07 Docker 批量部署-文件
│   │   │   └── docker-compose.yml
│   │   ├── 07 Docker 批量部署-步骤.md
│   │   ├── 07 Docker 批量部署-镜像.png
│   │   ├── 07 Docker 批量部署运行图 1.png
│   │   └── 07 Docker 批量部署运行图 2.png
│   └── 08 Gitee Go+微服务自动化部署-文件与步骤
│       ├── 07 Gitee Go+微服务自动化部署-步骤.md
│       └── 07 Gitee Go+微服务自动化部署-自动化工具产生的部署文件
│           ├── datasource
│           ├── zhfw-stop.sh
│           └── zhfw.sh
├── 11 附加：微服务架构+Gitee Go 自动化部署
│   └── ZHFW
│       ├── datasource
│       ├── datasource_b
│       ├── doc
│       ├── Dockerfile
│       ├── pom.xml
│       ├── README.md
│       ├── ZHFW-BookStore
│       │   ├── pom.xml
│       │   └── src
│       │       ├── main
│       │       │   ├── java
│       │       │   │   └── com
│       │       │   │       └── easyz
│       │       │   │           └── zhfw
│       │       │   │               ├── BookStoreApplication.java
│       │       │   │               ├── controller
│       │       │   │               │   └── v1
│       │       │   │               │       └── BookStoreController.java
│       │       │   │               ├── handler
│       │       │   │               │   └── BookHandler.java
│       │       │   │               ├── pojo
│       │       │   │               │   ├── AuthorsInfo.java
│       │       │   │               │   ├── BookInfo.java
│       │       │   │               │   └── BookInfo_Schema.json
│       │       │   │               ├── service
│       │       │   │               │   ├── BookCategoryWS.java
│       │       │   │               │   └── impl
│       │       │   │               │       └── BookCategoryWSImpl.java
│       │       │   │               └── vo
│       │       │   │                   └── UnAvailableException.java
│       │       │   └── resources
│       │       │       ├── application.yml
│       │       │       └── BookCategoryDB.xsd
│       │       └── test
│       │           └── java
│       │               └── com
│       │                   └── easyz
│       │                       └── zhfw
```


service
impl
ZHFW-Eureka
pom.xml
src
main
java
com
easyz
zhfw
EurekaApplication.java
resources
application.yml
test
java
ZHFW-FlightTicket
pom.xml
src
main
java
com
easyz
zhfw
controller
v1
FlightTicketController.java
FlightTicketApplication.java
handler
FlightHandler.java
pojo
AirlineInfo.java
AirlineList.java
BookingInfo.java
BookingList.java
DateInfo.java
DateList.java
FlightDB.xsd
ObjectFactory.java
TicketInfo.java
service
BookingFlightInfo.java
impl
BookingFlightInfoImpl.java
vo
FlightUnAvailableException.java
resources
application.yml
FlightDB.xsd
test
java
com
easyz
zhfw
service
impl
BookingFlightInfoImplTest.java
ZHFW-Gateway


pom.xml
src
main
java
com
easyz
zhfw
GatewayApplication.java
resources
application.yml
test
java
ZHFW-HotelReservation
pom.xml
src
main
java
com
easyz
zhfw
controller
v1
HotelReservationController.java
handler
HotelHandler.java
HotelReservationApplication.java
pojo
BookingInfo.java
BookingList.java
DateType.java
HotelDB.xsd
HotelInfo.java
ObjectFactory.java
RoomInfo.java
RoomList.java
StayPeriod.java
service
HotelReservationWS.java
impl
HotelReservationWSImpl.java
vo
UnAvailableException.java
resources
application.yml
HotelDB.xsd
test
java
ZHFW-Payment
pom.xml
src
main
java
com
easyz
zhfw
controller
v1


PaymentController.java
handler
PaymentHandler.java
PaymentApplication.java
pojo
AccountInfo.java
bankDB.xsd
BanKInfo.java
ObjectFactory.java
service
impl
PaymentWSImpl.java
PaymentWS.java
vo
TransactionException.java
resources
application.yml
bankDB.xsd
test
java
ZHFW-ServiceRetrieval
pom.xml
src
main
java
com
easyz
zhfw
controller
v1
EurekaRetrievalController.java
EurekaRetrievalApplication.java
pojo
ServiceInfo.java
service
impl
WSSearchEngineImpl.java
WSSearchEngine.java
resources
application.yml
test
java
ZHFW-TravelAgency
pom.xml
src
main
java
com
easyz
zhfw
client
BookStoreClient.java
FlightTicketClient.java
HotelReservationClient.java
PaymentClient.java
ServiceRetrievalClient.java
WarehouseClient.java


WeatherClient.java
config
AppConfig.java
controller
v1
TravelAgencyController.java
pojo
AirlineInfo.java
AuthorsInfo.java
BookInfo.java
Date.java
DateInfo.java
DateList.java
DateType.java
RoomInfo.java
ServiceInfo.java
StayPeriod.java
WeatherInfo.java
WeatherType.java
service
impl
TravelAgencyWSImpl.java
TravelAgencyWS.java
TravelAgencyApplication.java
vo
TravelAgencyRequestType.java
TravelAgencyResponseType.java
resources
application.yml
test
java
com
easyz
zhfw
service
impl
TravelAgencyWSImplTest.java
ZHFW-Warehouse
pom.xml
src
main
java
com
easyz
zhfw
controller
v1
WarehouseController.java
handler
WarehouseHandler.java
pojo
HoldingRequestInfo.java
HoldingRequestList.java
ItemInfo.java
ItemList.java
ObjectFactory.java
WareHouse.java


WarehouseDB.xsd
service
impl
WarehouseWSImpl.java
WarehouseWS.java
vo
InvalidHoldingIDException.java
NotEnoughItemException.java
WarehouseApplication.java
resources
application.yml
WarehouseDB.xsd
test
java
ZHFW-Weather
pom.xml
src
main
java
com
easyz
zhfw
controller
v1
WeatherController.java
handler
WeatherHandler.java
pojo
Date.java
ObjectFactory.java
WeatherDB.xsd
WeatherInfo.java
WeatherList.java
WeatherType.java
service
impl
WeatherWSImpl.java
WeatherWS.java
vo
WeatherApplication.java
resources
application.yml
WeatherDB.xsd
test
java
README.md