# 实验报告

## 一、 小组成员

**徐敏**

工作：

1. 数据库的设计与ER图构建
2. 基本功能中用户接口功能的实现
3. 额外功能中收货与发货接口的实现和搜索索引接口的设计与实现
4. 负责协作工作

**段连杰**

工作:

1. 数据库的设计与ER图构建
2. 基本功能中商户接口功能的实现与部分买家接口的实现
3. 额外功能中收货与发货接口的实现和搜索索引接口的设计与实现

**莫根杰**

工作：

1. 基本功能中部分买家接口的实现
2. 额外功能中查询历史订单和、取消订单和逾期订单的接口功能实现
3. 添加了一些毫无用处的功能接口
4. 书写实验报告

## 二、 实现的功能接口

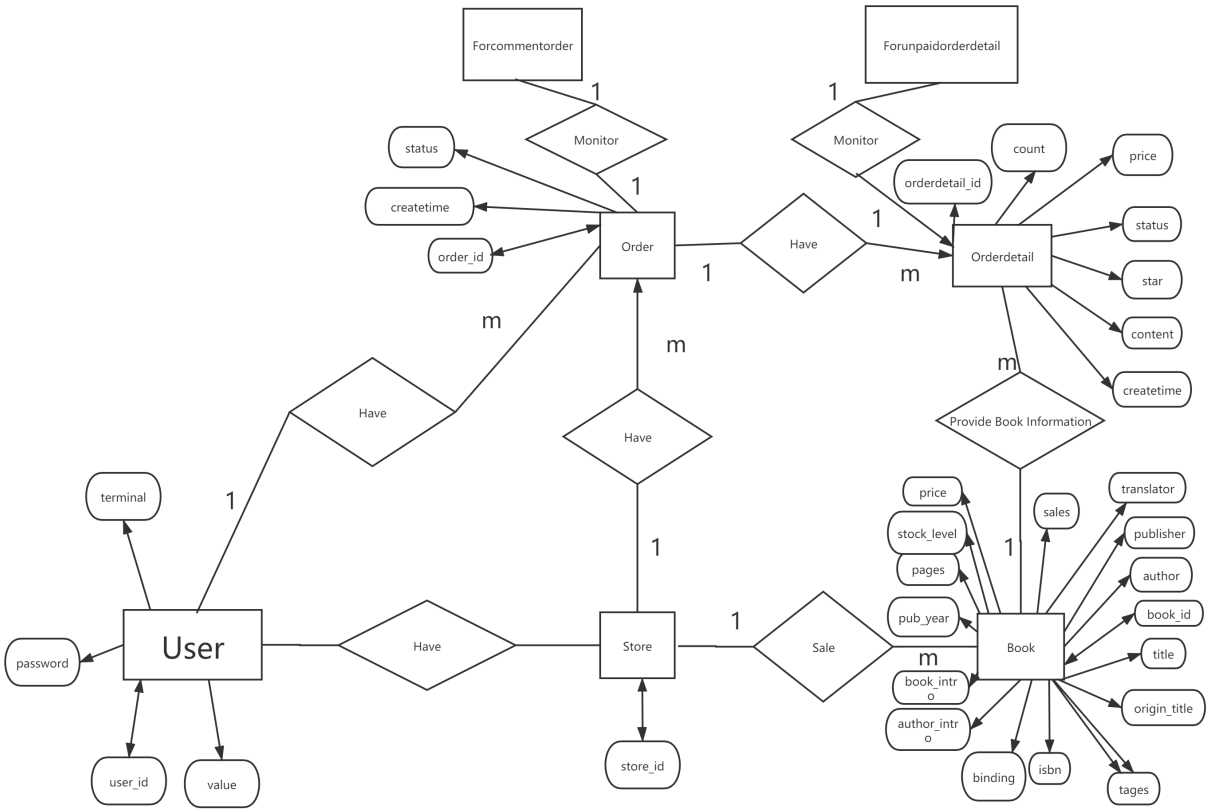
1. 基本的功能
2. 买家收货
3. 卖家发货
4. 买家取消订单
5. 买家根据id查询订单
6. 买家分别按进行中和已结束的分类，按时间顺序查询所有订单
7. 搜索图书功能
8. 逾期自动取消订单

9. 买家评论功能
10. 到时没有评价，自动默认好评功能
11. 获取商品所有评论

# 三、 项目目录结构

```
├─.idea
|
├─doc # 文档
├─fe
|   ├─access # 实现访问接口
|   ├─bench # 负载测试
|   ├─data # book数据
|   └─test # 测试文件
├─gtmd # 后端实现
└─venv # 虚拟环境
```

# 四、 数据库模式设计




# 1、表的作用

| 表名 | 作用 |
| --- | --- |

| | |
|---|---|
| User | 储存用户信息 |
| Store | 储存商店信息 |
| Order | 储存订单信息 |
| Book | 储存书店书籍信息 |
| Orderdetail | 储存订单内订单商品信息 |
| Forunpaidorder | 用于方便处理未付订单 |
| Foruncommentorderdetail | 用于方便处理未评论订单商品 |

## 2、表User

| 属性 | 类型 | 描述 | 设置 |
|---|---|---|---|
| user_id | varchar | 用户名 | 设置主键、设置索引类型UNIQUE、不为空 |
| password | varchar | 密码 | 不为空 |
| terminal | varchar | 用户登录设备 | 不为空 |
| value | int | 用户余额 | 默认为0、非负、不为空 |

## 3、表Store

| 字段 | 类型 | 描述 | 设置 |
|---|---|---|---|
| store_id | varchar | 商店名称 | 设置主键、设置索引类型UNIQUE、不为空 |
| seller_id | varchar | 商人名称 | 设置外键（参考表User中字段user_id)、设置索引类型NORMAL、不为空 |

## 4、表Order

| 字段 | 类型 | 描述 | 设置 |
|---|---|---|---|
| order_id | varchar | 订单ID | 设置主键、设置索引类型UNIQUE、不为空 |
| buyer_id | varchar | 下单用户ID | 设置外键（参考表User中字段user_id）、设置索引类型NORMAL、不为空 |

| 字段 | 类型 | 描述 | 设置 |
| --- | --- | --- | --- |
| store_id | varchar | 下单店铺ID | 设置外键（参考表Store中字段store_id）、设置索引类型NORMAL、不为空 |
| createtime | datetime | 创建时间 | 默认当前时间、不为空 |
| receivedtime | datetime | 订单接收时间 | |
| status | varchar | 订单状态 | 订单当前状态 |

## 5、表Book

| 字段 | 类型 | 描述 | 设置 |
| --- | --- | --- | --- |
| book_id | varchar | 书籍ID | 设置主键、设置索引类型UNIQUE、不为空 |
| store_id | varchar | 售卖店铺ID | 设置外键（参考表Store中字段Store_id）、设置索引类型NORMAL、不为空 |
| title | text | 书籍名称 | |
| author | text | 书籍作者 | |
| publisher | text | 书籍出版商 | |
| original_title | text | 书籍原始标题 | |
| translator | text | 书籍译者 | |
| pub_year | text | 书籍出版年份 | |
| pages | int | 书籍页数 | |
| price | int | 书籍价格 | |
| binding | text | 书籍装订 | |

| 字段 | 类型 | 描述 | 设置 |
| --- | --- | --- | --- |
| isbn | text | 书籍ISBN号 | |
| author_intro | text | 书籍作者介绍 | |
| book_intro | text | 书籍介绍 | |
| content | text | 书籍内容摘录 | |
| tags | text | 书籍标签 | |
| pictures | mediumblob | 书籍图片 | |
| sales | int | 书籍销量 | 默认为0、非负、不为空 |
| stock_level | int | 书籍库存 | 默认为0、非负、不为空 |

## 6、表Orderdetail

| 字段 | 类型 | 描述 | 设置 |
| --- | --- | --- | --- |
| orderdetail_id | varchar | 订单内物品ID | 设置主键、设置索引类型UNIQUE、不为空 |
| order_id | varchar | 物品所属的订单id | 设置外键（参考表Order字段order_id）、设置索引类型NORMAL、不为空 |
| book_id | varchar | 物品对应的书籍商品ID | 设置外键（参考表Book字段book_id）、设置索引类型NORMAL、不为空 |
| count | int | 下单时订单物品数量 | 不为空 |
| prince | int | 下单时物品当前的售价 | 不为空 |
| status | varchar | 订单对应物品的评价状态 | 不为空 |

| 字段 | 类型 | 描述 | 设置 |
| --- | --- | --- | --- |
| createtime | datetime | 评论创建时间 | |
| star | int | 评分 | |
| comment | text | 评论内容 | |

## 7、表Forunpaidorder

| 字段 | 类型 | 描述 | 设置 |
| --- | --- | --- | --- |
| id | varchar | 自动增长的id，用于记录监视未完成订单的先后顺序 | 设置为主键、不为空 |
| order_id | varchar | 监视的订单ID | 不为空 |

## 8、表Foruncommentorderdetail

| 字段 | 类型 | 描述 | 设置 |
| --- | --- | --- | --- |
| id | varchar | 自动增长的id，用于记录监视订单的先后顺序 | 设置为主键、不为空 |
| orderdetail_id | varchar | 监视的订单物品ID | 不为空 |

# 五、 接口设计

## 1、 用户接口

**实现文件:**

gtmd/blueprints/auth.py

### (1) 用户登录

**a. 实现函数**

register()

**b. URL**

POST http://$address$/auth/register

**c. Request**

Body

```
{
    "user_id": "user_id",
    "password": "password",
}
```

变量名：

| 变量名 | 类型 | 描述 | 是否可以为空 |
| --- | --- | --- | --- |
| user_id | string | 用户名 | N |
| password | string | 登录密码 | N |

**d. Response**

Status Code:

| 码 | 描述 | 产生错误原因 |
| --- | --- | --- |
| 200 | 注册成功 | |
| 501 | 注册失败，用户名重复 | 表User中user_id为主键，不能重复，插入表中已存在的user_id，产生错误 |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |

## (2) 注销用户

**a. 实现函数:**

unregister()

**b. URL**

POST http://$address$/auth/unregister

**c. Request**

Body:

```
{
    "user_id": "user_id",
    "password": "password"
}
```

**d. Response**

Status Code:

| 码 | 描述 | 产生错误原因 |
| --- | --- | --- |
| 200 | 注销成功 | |
| 401 | 注销失败，用户名不存在或密码不正确 | 在表User中根据字段user_id字段为user_id 查找到的结果为空，或查询结果的字段password中的值与值password不等 |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |

## (3) 用户更改密码

**a. 实现函数**

password()

**b. URL**

POST http://$address$/auth/password

**c. Request**

Body:

```
{
    "user_id": "user_id",
    "oldPassword": "oldPassword",
    "newPassword": "newPassword"
}
```

属性说明:

| 变量名 | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| user_id | string | 用户名 | N |
| password | string | 登陆密码 | N |
| terminal | string | 终端代码 | N |

**d. Response**

Status Code:

| 码 | 描述 | 产生错误原因 |
| --- | --- | --- |
| 200 | 更改密码成功 | |
| 401 | 更改密码失败 | 根据字段user_id为值user_id和字段password为值oldPassword在表User中进行查询的返回的结果为空 |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |

### (4) 用户登出

#### a. 实现函数

logout()

#### b. URL

POST http://$address$/auth/logout

#### c. Request

Headers:

| key | 类型 | 描述 |
| --- | --- | --- |
| token | string | 访问token |

Body:

```
{
    "user_id": "user_id"
}
```

属性说明:

| 变量名 | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| user_id | string | 用户名 | N |

#### d. Response

Status Code:

| 码 | 描述 | 产生错误原因 |
| --- | --- | --- |
| 200 | 登出成功 | |
| 401 | 登出失败，用户名或token错误 | token解析错误或token解析出来的的json中的user_id的值与Request中json的user_id的值不一致 |

Body:

```
{
    "message": error message
```

```
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |

## 2. 商家接口

**实现文件:**

gtmd/blueprints/seller.py

### (1) 创建商铺

**a. 实现函数:**

create_store()

**b. URL**

POST http://$address$/seller/create_store

**c. Request**

Headers:

| key | 类型 | 描述 | 是否为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

Body:

```
{
    "user_id": "$seller_id$",
    "store_id": "$store_id$"
}
```

属性说明:

| key | 类型 | 描述 | 是否为空 |
| --- | --- | --- | --- |
| user_id | string | 卖家用户ID | N |
| store_id | string | 商铺ID | N |

**d. Response**

Status Code:

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 200 | 创建商铺成功 | |
| 401 | 创建失败，用户名或token错误 | token解析错误或token解析出来的的json中的user_id的值与Request中json的user_id的值不一致 |
| 501 | 商铺ID已存在 | 字段store_id为表Store中的主键不能重复 |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |

### (2) 商家添加书籍信息

**a. 实现函数:**

add_book()

**b. URL:**

POST http://$address$/seller/add_book

**c. Request**

Headers:

| key | 类型 | 描述 | 是否为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

Body:

```
{
  "user_id": "seller user id$",
```

```
  "store_id": "$store id$",
  "book_info": {
    "tags": [
      "tags1",
      "tags2",
      "tags3",
      "..."
    ],
    "pictures": [
      "$Base 64 encoded bytes array1$",
      "$Base 64 encoded bytes array2$",
      "$Base 64 encoded bytes array3$",
      "..."
    ],
    "id": "$book id$",
    "title": "$book title$",
    "author": "$book author$",
    "publisher": "$book publisher$",
    "original_title": "$original title$",
    "translator": "translater",
    "pub_year": "$pub year$",
    "pages": 10,
    "price": 10,
    "binding": "平装",
    "isbn": "$isbn$",
    "author_intro": "$author introduction$",
    "book_intro": "$book introduction$",
    "content": "$chapter1 ...$"
  },
  "stock_level": 0
}
```

属性说明：

| 变量名 | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| user_id | string | 卖家用户ID | N |
| store_id | string | 商铺ID | N |
| book_info | class | 书籍信息 | N |
| stock_level | int | 初始库存，大于等于0 | N |

book_info类：

| 变量名 | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | string | 书籍ID | N |
| title | string | 书籍题目 | N |
| author | string | 作者 | Y |

| 变量名 | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| publisher | string | 出版社 | Y |
| original_title | string | 原书题目 | Y |
| translator | string | 译者 | Y |
| pub_year | string | 出版年月 | Y |
| pages | int | 页数 | Y |
| price | int | 价格(以分为单位) | N |
| binding | string | 装帧，精状/平装 | Y |
| isbn | string | ISBN号 | Y |
| author_intro | string | 作者简介 | Y |
| book_intro | string | 书籍简介 | Y |
| content | string | 样章试读 | Y |
| tags | array | 标签 | Y |
| pictures | array | 照片 | Y |

**d. Response**

Status Code:

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 200 | 创建商铺成功 | |
| 501 | 卖家用户ID不存在 | 查询User表中字段user_id为seller_id的元组，返回值为空 |
| 502 | 添加失败，用户名或token错误 token解析返回值为空 | token解析错误或token解析出来的的json中的user_id的值与Request中json的user_id的值不一致 |
| 503 | 商铺ID不存在 | 查询Store表中字段store_id为store_id的元组，返回值为空 |

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 5<br>0<br>4 | 图书ID已存在 | book_id由店铺和卖家添加的id组成，字段book_id是表Book的主键，经由组合产生的book_id不能重复，即同一个店铺中不能存在两个相同的book_id |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |

### (3) 商家添加书籍库存

#### a. 实现函数

update_stock_level()

#### b. URL

POST http://$address$/seller/add_stock_level

#### c. Request:

Headers:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

Body:

```
{
  "user_id": "$seller id$",
  "store_id": "$store id$",
  "book_id": "$book id$",
  "add_stock_level": 10
}
```

属性说明:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |

| | | | |
|---|---|---|---|
| user_id | string | 卖家用户ID | N |
| store_id | string | 商铺ID | N |
| book_id | string | 书籍ID | N |
| add_stock_level | int | 增加的库存量 | N |

**d. Response**

Status Code:

| 码 | 描述 | 产生错误的原因 |
|---|---|---|
| 200 | 创建商铺成功 | |
| 501 | 卖家用户ID不存在 | 查询User表中字段user_id为seller_id的元组，返回的结果为空 |
| 502 | 添加失败，用户名或token错误 | token解析错误或token解析出来的的json中的user_id的值与Request中json的user_id的值不一致 |
| 503 | 商铺ID不存在 | 查询Store表中字段store_id为store_id的元组，返回的结果为空 |
| 504 | 图书ID不存在 | 查询Book表中字段book_id为book_id的元组，返回的结果为空 |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
|---|---|---|---|
| message | string | 返回错误消息，成功时为ok | N |

## (4) 商家发货

**a. 实现函数**

change_unreceived()

**b. URL**

POST http://$address$/seller/change_unreceived

**c. Request**

Headers:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

Body:

```
{
  "user_id": "$seller id$",
  "store_id": "$store id$",
  "book_id": "$book id$",
  "order_id": "$order id"
}
```

属性说明:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| user_id | string | 卖家用户ID | N |
| store_id | string | 商铺ID | N |
| book_id | string | 书籍ID | N |
| order_id | string | 订单ID | N |

**d. Response**

Status Code:

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 200 | 创建商铺成功 | |
| 501 | 卖家用户ID不存在 | 查询User表中字段user_id为seller_id的元组，返回的结果为空 |

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 502 | 修改状态失败，用户名或token错误 | token解析错误或token解析出来的的json中的user_id的值与Request中json的user_id的值不一致 |
| 503 | 商铺ID不存在 | 查询Store表中字段store_id为store_id的元组，返回的结果为空 |
| 504 | 订单查询失败 | 查询Order表中字段order_id为order_id的元组，返回的结果为空 |
| 505 | 该订单物品状态不是待发货状态，无法切换为发货状态 | Order表中字段order_id为order_id的元组中字段status的值不为paid状态，商家无权对其进行发货操作 |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |

# 3. 买家接口

**实现文件:**

gtmd/blueprints/buyer.py

## (1) 买家下单

### a. 实现函数

new_order()

### b. URL

POST http://$address$/buyer/new_order

### c. Request

Header:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

Body:

```
{
  "user_id": "buyer_id",
  "store_id": "store_id",
  "books": [
    {
      "id": "1000067",
      "count": 1
    },
    {
      "id": "1000134",
      "count": 4
    }
  ]
}
```

属性说明：

| 变量名 | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| user_id | string | 买家用户ID | N |
| store_id | string | 商铺ID | N |
| books | class | 书籍购买列表 | N |

books数组：

| 变量名 | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | string | 书籍的ID | N |
| count | string | 购买数量 | N |

**d. Response**

Status Code:

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 200 | 下单成功 | |

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 501 | 买家用户ID不存在 | 查询User表中字段user_id为user_id的元组，返回的查询结果为空 |
| 502 | 用户名或token错误 | token解析返回值为或token解析成json格式后user_id对应的值与buyer_id不等 |
| 503 | 商户ID不存在 | 查询Store表中字段store_id为store_id的元祖，返回的查询结果为空 |
| 504 | 购买的图书不存在 | 对订单中的书籍进行查询，查询Book表中字段Book_id为book_id的元祖，返回的查询结果为空 |
| 505 | 商品库存不足 | 查询Book表中字段book_id为book_id，若返回值不为空但其字段stock_level小于买家期望购买的值 |

Body:

```
{
  "order_id": "uuid"
}
```

属性说明：

| 变量名 | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| order_id | string | 订单号，只有返回200时才有效 | N |

## (2) 买家付款

**a. 实现函数**

payment()

**b. URL**

POST http://$address$/buyer/payment

**c. Request**

```
Body:
{
```

```
  "user_id": "buyer_id",
  "order_id": "order_id",
  "password": "password"
}
```

属性说明：

| 变量名 | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| user_id | string | 买家用户ID | N |
| order_id | string | 订单ID | N |
| password | string | 买家用户密码 | N |

**d. Response**

Status Code:

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 200 | 付款操作 | |
| 401 | 授权失败 | 查询User表中字段user_id为buyer_id和字段password为password的元组，返回的查询结果为空 |
| 501 | 无效参数 | 查询Order表中字段order_id为order_id的元祖，若返回值为空，则该order_id无效 |
| 502 | 当前订单状态无法支付 | 该订单状态不是unpaid，用户无权对该订单进行支付操作 |
| 503 | 账户余额不足 | 账户的余额小于账单需支付的总金额 |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |

| | | | |
|---|---|---|---|
| message | string | 返回错误消息，成功时为ok | N |

## (3) 买家充值

#### a. 实现函数

add_funds()

#### b. URL

POST http://$address$/buyer/add_funds

#### c. Request

```
Body:
{
  "user_id": "user_id",
  "password": "password",
  "add_value": 10
}
```

属性说明:

| key | 类型 | 描述 | 是否可为空 |
|---|---|---|---|
| user_id | string | 买家用户ID | N |
| password | string | 用户密码 | N |
| add_value | int | 充值金额，以分为单位 | N |

#### d. Response

Status Code:

| 码 | 描述 | 产生错误的原因 |
|---|---|---|
| 200 | 充值成功 | |
| 401 | 授权失败 | 查询User表中字段user_id为buyer_id和字段password为password的元组，返回的结果为控制 |
| 501 | 无效参数 | add_value（以分为单位）不为整数值 |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |

## (4) 买家收货

### a. 实现函数

change_received()

### b. URL

POST http://$address$/buyer/change_received

### c. Request

Headers:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

Body:

```
{
  "user_id": "$buyer id$",
  "store_id": "$store id$",
  "book_id": "$book id$",
  "order_id": "$order id$"
}
```

属性说明：

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| user_id | string | 买家用户ID | N |
| store_id | string | 商铺ID | N |
| book_id | string | 书籍ID | N |
| order_id | string | 订单ID | N |

**d. Response**

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 200 | 收货成功 | |
| 501 | 买家用户ID不存在 | 查询表User中字段user_id为值user_id的元组，返回的查询结果为空 |
| 502 | 用户名或token错误 | token解析返回值为或token解析成json格式后user_id对应的值与buyer_id不等 |
| 503 | 当前用户没有该订单或订单不存在 | 查询表Order中字段order_id为值order_id的元组且字段user_id为值buy_id的元组，返回的查询结果为空 |
| 504 | 当前订单物品状态不为待发货，无法切换为收货状态 | 查询表Order中字段order_id为值order_id的元组且字段user_id为值buy_id的元组，返回的结果中其字段status状态不为unreceived，用户无权将其切换为收货状态 |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |

## (5) 买家取消订单

**a. 实现函数**

cancel_order()

**b. URL**

POST http://$address$/buyer/cancel_order

**c. Request**

Headers:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

**Body**

```
{
  "user_id": "$buyer id$",
  "store_id": "$store id$",
  "book_id": "$book id$",
  "order_id": "$order id$"
}
```

属性说明：

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| user_id | string | 买家用户ID | N |
| store_id | string | 商铺ID | N |
| book_id | string | 书籍ID | N |
| order_id | string | 订单ID | N |

**d. Response**

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 200 | 取消订单成功 | |
| 501 | 买家用户ID不存在 | 查询表User中字段user_id为值user_id的元组，返回的查询结果为空 |
| 502 | 用户名或token错误 | token解析返回值为或token解析成json格式后user_id对应的值与buyer_id不等 |
| 503 | 当前用户没有该订单或订单不存在 | 查询表Order中字段order_id为值order_id的元组且字段user_id为值buy_id的元组，返回的查询结果为空 |
| 504 | 当前订单无法取消 | 查询表Order中字段order_id为值order_id的元组且字段user_id为值buy_id的元组，返回的结果中其字段status状态不为unpaid或paid |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |

## (6) 买家根据订单id查询订单

### a. 实现函数

track_order_by_order_id()

### b. URL

POST http://$address$/buyer/track_order_by_order_id

### c. Requset

Headers:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

### Body

```
{
  "user_id": "$buyer id$",
  "order_id": "$order id$"
}
```

属性说明：

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| user_id | string | 买家用户ID | N |
| order_id | string | 订单ID | N |

### d. Response

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |

| | | |
|---|---|---|
| 200 | 查询订单成功 | |
| 501 | 买家用户ID不存在 | 查询表User中字段user_id为值user_id的元组，返回的查询结果为空 |
| 502 | 用户名或token错误 | token解析返回值为或token解析成json格式后user_id对应的值与buyer_id不等 |
| 503 | 当前用户没有该订单或订单不存在 | 查询表Order中字段order_id为值order_id的元组且字段user_id为值buy_id的元组，返回的查询结果为空 |

Body:

```
{
    "message": error message,
    "orderdetail": [
        {"orderdetail_id": "$orderdetail_id$",
        "book_id": "$book_id$",
        "book_title": "$book_title$",
        "book_author": "$book_author$",
        "count": "$count$",
        "unit_price": "$price$"}
        ....
    ]
    "total_price": "$total_price$"
}
```

| 变量名 | 类型 | 描述 | 非空 |
|---|---|---|---|
| message | string | 返回错误消息，成功时为ok | N |
| orderdetail | array | 含有订单物品信息的信息 | |
| orderdetail_id | string | 订单物品id | N |
| book_id | string | 订单物品对应书籍的id | N |
| book_title | string | 订单物品对应书籍的标题 | N |
| book_author | string | 订单物品对应书籍的作者 | |
| count | int | 订单对应对书籍数量 | N |
| unit_price | int | 订单物品对应的书籍单价 | N |
| total_price | int | 订单对应的书籍总价 | N |

## (7) 买家根据订单id查询订单

### a. 实现函数

track_order()

### b. URL

POST http://$address$/buyer/track_order

### c. Requset

Headers:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

### Body

```
{
  "user_id": "$buyer id$",
}
```

属性说明:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| user_id | string | 买家用户ID | N |

### d. Response

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 200 | 查询订单成功 | |
| 501 | 买家用户ID不存在 | 查询表User中字段user_id为值user_id的元组，返回的查询结果为空 |
| 502 | 用户名或token错误 | token解析返回值为或token解析成json格式后user_id对应的值与buyer_id不等 |

Body:

```
{
    "message": error message,
    "order": [
        {"order_id": "order_id",
        "createtime": "createtime",
        "store_id": "store_id",
        "status": "status",
        "orderdetail": [ {
            "orderdetail_id": "$orderdetail_id$",
            "book_id": "$book_id$",
            "book_title": "$book_title$",
            "book_author": "$book_author$",
            "count": "$count$",
            "unit_price": "$price$",
            }
        ....
        ]
        "total_price": "$total_price$"
        }
        ....
    ]
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |
| order | array | 含有订单信息的数组 | |
| order_id | string | 订单id | N |
| createtime | string | 订单创建日期 | N |
| store_id | string | 下单店铺id | N |
| status | string | 订单状态 | N |
| orderdetail | array | 含有订单物品信息的数组 | |
| orderdetail_id | string | 订单物品id | N |
| book_id | string | 订单物品对应书籍的id | N |
| book_title | string | 订单物品对应书籍的标题 | N |
| book_author | string | 订单物品对应书籍的作者 | |
| count | int | 订单对应对书籍数量 | N |
| unit_price | int | 订单物品对应的书籍单价 | N |
| total_price | int | 订单对应的书籍总价 | N |

## (7) 买家根据订单id查询订单

**a. 实现函数**

add_comment()

**b. URL**

POST http://$address$/buyer/add_comment

**c. Requset**

Headers:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

**Body**

```
{
  "user_id": "$buyer_id$",
  "orderdetail_id"; "$orderdetail_id$",
  "star": "$star$",
  "content": "$content$"
}
```

属性说明：

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| user_id | string | 买家用户ID | N |
| orderdetail_id | string | 买家评论购买书籍的id | N |
| star | int | 买家给书籍的评分 | N |
| content | string | 买家评论内容 | N |

**d. Response**

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 200 | 添加评论成功 | |
| 501 | 买家用户ID不存在 | 查询表User中子段user_id为值user_id的元组，返回的查询结果为空 |

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 502 | 用户名或token错误 | token解析返回值为或token解析成json格式后user_id对应的值与buyer_id不等 |
| 503 | 订单物品出错 | 在Orderdetail表中搜索字段orderdetail_id为值orderdetail的结果为空 |
| 504 | 该物品对应订单还未收货，无法评价 | 在Orderdetail表中搜索字段orderdetail_id为值orderdetail的返回的结果中外键字段order_id参考的表中对应的元祖中的字段status的不为"received" |
| 505 | 该物品已被评论，评论失败 | 在Orderdetail表中搜索字段orderdetail_id为值orderdetail的返回的结果中字段status中不为uncommented |
| 506 | 评分参数错误 | json传入参数为star的值不在1-5范围内 |
| 507 | 评论不能为空 | json传入参数为content的值为空 |

Body:

```
{
    "message": error message
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |

## (8) 买家查看书籍评论

**a. 实现函数**

get_comment_by_book_id()

**b. URL**

POST http://$address$/buyer/get_comment_by_book_id

**c. Requset**

**Body**

```
{
    "store_id": "$store_id$",
    "book_id": "$book_id$"
}
```

属性说明:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| store_id | string | 书籍对应店铺id | N |
| book_id | string | 书籍的id | N |

### d. Response

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 200 | 查询书籍评论成功 | |
| 501 | 商店不存在 | 查询表Store中字段store_id为值store_id的元组，返回的查询结果为空 |
| 502 | 该图书不存在 | 查询表Book中字段book_id为值book_id的元组，返回的查询结果为空 |

Body:

```
{
    "message": error message,
    "comment":[{
        "createtime": "createtime",
        "buyer_id": "buyer_id",
        "star":  "star",
        "content": "content"
        }
        ...
    ]
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |
| comment | array | 商品对应的评论 | N |
| createtime | string | 评论时间 | N |

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| buyer_id | string | 买家id | N |
| star | int | 买家评分 | N |
| content | string | 买家评论 | N |

## (9) 买家在书店搜索

#### a. 实现函数

search_book_store()

#### b. URL

POST http://$address$/buyer/search_book_store

#### c. Requset

Headers:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

#### Body

```
{
    "buyer_id": "$buyer_id$",
    "store_id": "$store_id$",
    "key_word": "$keyword"
}
```

属性说明:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| buyer_id | string | 买家的id | N |
| store_id | string | 搜索的店铺id | N |
| key_word | string | 搜索关键字 | N |

#### d. Response

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |

| | | |
|---|---|---|
| 200 | 查询店铺成功 | |
| 501 | 买家用户ID不存在 | 查询表User中字段user_id为值user_id的元组，返回的查询结果为空 |
| 502 | 用户名或token错误 | token解析返回值为或token解析成json格式后user_id对应的值与buyer_id不等 |

Body:

```
{
    "message" : error message,
    "search_book_result":[{
        "book_id": "book_id",
        "title": "title",
        "author": "author",
        "publisher": "publisher",
        "original_title": "original_title",
        "translator": "translator",
        "pub_year": "pub_year",
        "pages": "pages",
        "price": "price",
        "binding": "binding",
        "isbn": "isbn",
        "author_intro": "author_intro",
        "book_intro": "book_intro",
        "content: "content",
        "tags": "tags",
        }
        ...
    ]
}
```

| 变量名 | 类型 | 描述 | 非空 |
|---|---|---|---|
| message | string | 返回错误消息，成功时为ok | N |
| search_book_result | string | 存储搜索到的图书信息 | N |
| book_id | string | 书籍的id | N |
| title | string | 书籍标题 | N |
| author | string | 书籍作者 | |
| original_title | string | 原始标题 | |
| translator | string | 书籍译者 | |

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| pub_year | string | 书籍出版年份 | |
| pages | int | 书籍页数 | |
| price | int | 书籍价格 | N |
| binding | string | 书籍订装 | |
| isbn | string | 书籍isbn | |
| author | string | 作者简介 | |
| book_intro | string | 书籍简介 | |
| content | string | 书籍摘录 | |
| tags | string | 书籍标签 | |
| picture | string | 书籍的base64图片编码 | |

## (9) 买家全站搜索

**a. 实现函数**

search_book_site()

**b. URL**

POST http://$address$/buyer/search_book_site

**c. Requset**

Headers:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| token | string | 登录产生的会话标识 | N |

**Body**

```
{
    "buyer_id": "$buyer_id$",
    "key_word": "$keyword"
}
```

属性说明:

| key | 类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |

| buyer_id | string | 买家id | N |
| --- | --- | --- | --- |
| key_word | string | 搜索关键字 | N |

**d. Response**

| 码 | 描述 | 产生错误的原因 |
| --- | --- | --- |
| 200 | 查询店铺成功 | |
| 501 | 买家用户ID不存在 | 查询表User中字段user_id为值user_id的元组，返回的查询结果为空 |
| 502 | 用户名或token错误 | token解析返回值为或token解析成json格式后user_id对应的值与buyer_id不等 |

Body:

```
{
    "message" : error message,
    "search_book_result":[{
        "book_id": "book_id",
        "store_id": "store_id",
        "title": "title",
        "author": "author",
        "publisher": "publisher",
        "original_title": "original_title",
        "translator": "translator",
        "pub_year": "pub_year",
        "pages": "pages",
        "price": "price",
        "binding": "binding",
        "isbn": "isbn",
        "author_intro": "author_intro",
        "book_intro": "book_intro",
        "content: "content",
        "tags": "tags",
        }
        ...
    ]
}
```

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| message | string | 返回错误消息，成功时为ok | N |
| search_book_result | string | 存储搜索到的图书信息 | N |

| 变量名 | 类型 | 描述 | 非空 |
| --- | --- | --- | --- |
| book_id | string | 书籍的id | N |
| store_id | string | 店铺id | N |
| title | string | 书籍标题 | N |
| author | string | 书籍作者 | |
| original_title | string | 原始标题 | |
| translator | string | 书籍译者 | |
| pub_year | string | 书籍出版年份 | |
| pages | int | 书籍页数 | |
| price | int | 书籍价格 | N |
| binding | string | 书籍订装 | |
| isbn | string | 书籍isbn | |
| author | string | 作者简介 | |
| book_intro | string | 书籍简介 | |
| content | string | 书籍摘录 | |
| tags | string | 书籍标签 | |
| picture | string | 书籍的base64图片编码 | |

# 七、部分功能事务设计

## 1、用户注册

**(1) user_id与表User中字段user_id已存在的值不重复吗，向User表中插入一条新的数据， 否则返回错误码和信息。**

**(2) 提交事务，注册成功。**

## 2、用户登录

**(1) 搜索表User中字段user_id值为user_id，结果不为空，否则返回错误码和信息。**

**(2) 搜索结果中字段password与password相同，否则返回错误值和信息。**

**(3) 登录成功，返回登录成功信息和token。**

## 3、用户注销

**(1) 搜索表User中字段user_id为值user_id和字段password为值password的数据，结果不为空，否则返回错误码和信息。**

**(2) 删除对应的数据元组，提交事务，注销成功。**

## 4、用户修改密码

**(1) 搜索表User中字段user_id为值user_id和字段password为值xoldPassword，数据，结果不为空，否则返回错误码和信息。**

**(2) 将字段password的值修改为newPassword，提交事务 ，修改成功。**

## 5、用户登出

**(1) 搜索表User中字段user_id为值user_id和字段password为值password的数据，结果不为空，且验证token成功，否则返回错误码和信息。**

**(2) 用户登出成功**

## 6、卖家创建店铺

**(1) 搜索表User中字段user_id为值seller_id，结果不为空，且验证token成功，否则返回错误码和信息。**

**(2) 向Store表中插入字段store_id为值store_id，字段user_id为seller_id的数据。**

(3) 提交事务，创建成功，报错则回滚并返回错误码和信息。

## 7. 卖家添加书籍

**(1) 验证token正确，否则返回错误码和信息**

**(2) 搜索表User中字段user_id为值seller_id，结果不为空，否则返回错误码和信息。**

**(3) 搜索表Store中字段seller_id为值seller_id、字段user_id为值seller_id，结果不为空，否则返回错误码和信息**

**(4) 向表Book中插入书籍数据，提交事务，插入成功，报错则回滚并返回错误码和信息**

## 8、卖家增加库存

**(1) 验证token正确，否则返回错误码和信息。**

**(2) 搜索表User中字段user_id为值seller_id，结果不为空，否则返回错误码和信息。**

(3) 搜索表Store中字段store_id为值store_id、字段user_id为值seller_id，结果不为空，否则返回错误码和信息

(4) 搜索表Book中字段book_id为值book_id的数据，结果不为空，否则返回错误码和信息。

(5) 增添对应的库存。

(6) 提交事务，卖家增加库存成功。

## 8、卖家更改订单状态为未收货

(1) 验证token正确，否则返回错误码和信息。

(2) 搜索表User中字段user_id为值seller_id，结果不为空，否则返回错误码和信息。

(3) 搜索表Store中字段store_id为值store_id、字段user_id为值seller_id，结果不为空，否则返回错误码和信息

(4) 搜索表Order中字段order_id为值order_id的数据，结果不为空，否则返回错误码和信息。

(5) 判断订单状态为paid，否则返回错误码和信息。

(6) 更改订单状态为发货，提交事务，订单状态更改成功。

## 9、 买家创建新订单

(1) 验证token正确，否则返回错误码和信息

(2) 搜索表User中字段user_id为值buyer_id，结果不为空，否则返回错误码和信息。

(3) 搜索表Store中字段store_id为值store_id，结果不为空，否则返回错误码和信息

(4) 在表Order中创建order_id为order_id的订单数据，提交事务，然后依次搜索判断想购买的图书是否在表Book中和库存是否足够，否则，回滚事务，并删除之前提交事务创建的订单。

(5) 生成订单信息，和更新库存信息，并在表Forunpaidorder生成一个待支付订单数据，提交事务，订单创建成功。

## 10、买家充值金额

(1) 充值金额正确，否则返回错误码和信息

(2) 验证token正确，否则返回错误码和信息

(3) 搜索表User中字段user_id为值buyer_id和字段password为值password，结果不为空，否则返回错误码和信息。

(4) 更新上一条搜索结果字段value对应的值，提交事务，则买家充值成功。

## 11、买家支付订单

**(1) 验证token正确，否则返回错误码和信息。**

**(2) 搜索表User中字段user_id为值seller_id，结果不为空，否则返回错误码和信息。**

**(3) 搜索表Store中字段store_id为值store_id、字段user_id为值seller_id，结果不为空，否则返回错误码和信息**

**(4) 搜索表Order中字段order_id为值order_id的数据，结果不为空，否则返回错误码和信息。**

**(5) 根据创建的外键遍历订单物品信息，计算要购买书籍的总价低于当前账户余额，否则回滚事务并返回错误码和信息。**

**(6) 更新用户的余额和订单的状态，提交事务，订单支付成功。**

## 12、买家收货

**(1) 验证token正确，否则返回错误码和信息。**

**(2) 搜索表User中字段user_id为值seller_id，结果不为空，否则返回错误码和信息。**

**(3) 搜索表Order中字段order_id为值order_id的数据，结果不为空，否则返回错误码和信息。**

**(4) 判断订单数据的字段status为"unreceived"， 否则返回错误码和信息。**

**(5) 将订单状态进行更新，并更改对应商品的销量和在表Foruncommentorderdetail创建对应所有的含orderdetail_id的数据，提交事务，买家收货成功。**

## 13、买家取消订单

**(1) 验证token正确，否则返回错误码和信息。**

**(2) 搜索表User中字段user_id为值seller_id，结果不为空，否则返回错误码和信息。**

**(3) 搜索表Order中字段order_id为值order_id的数据，结果不为空，否则返回错误码和信息。**

**(4) 对订单数据的状态字段"status"进行验证，不为"unpaid"或"paid"，否则返回错误码和信息**

**(5) 更改订单数据的状态，若订单状态为"paid"，则同时查询表Order1detail交易的商品数据，更新买家的字段Value，提交事务，取消订单成功。**

## 14、买家在店铺搜索数据

**(1) 验证token正确，否则返回错误码和信息。**

**(2) 搜索表User中字段user_id为值seller_id，结果不为空，否则返回错误码和信息。**

(3) 根据关键词和store_id在表Book中查询对应的数据，返回成功的编码和查询得到的数据。

## 15、买家在店铺搜索数据

(1) 验证token正确，否则返回错误码和信息。

(2) 搜索表User中字段user_id为值seller_id，结果不为空，否则返回错误码和信息。

(3) 根据关键词在表Book中查询对应的数据，返回成功的编码和查询得到的数据。

## 16、买家查根据订单Id查询订单

(1) 验证token正确，否则返回错误码和信息。

(2) 搜索表User中字段user_id为值seller_id，结果不为空，否则返回错误码和信息。

(3) 搜索表Order中字段order_id为值order_id的数据，结果不为空，否则返回错误码和信息。

(4) 根据参考字段order_id的外键获取对应的订单物品的信息，同时再根据表Orderdeatail的外键字段book_id获取表中对应Book的信息，返回成功的编码和对应的json数据。

## 16、买家查询历史订单

(1) 验证token正确，否则返回错误码和信息。

(2) 搜索表User中字段user_id为值seller_id，结果不为空，否则返回错误码和信息。

(3) 搜索表Order中字段user_id为值buyer_id的数据，根据字段status分为已完成和未完成的订单两批次进行查询，根据参考字段order_id的外键获取对应的订单物品的信息，同时再根据表Orderdeatail的外键字段book_id获取表中对应Book的信息，优先构建未完成订单的数据，返回成功的编码和对应的json数据。

## 16、买家查询历史订单

(1) 验证token正确，否则返回错误码和信息。

(2) 搜索表User中字段user_id为值seller_id，结果不为空，否则返回错误码和信息。

## 17、系统自动取消逾期订单

(1) 在另起的一线程中，查询表Forunpaidorder中自增长字段id最小的数据，根据参考的外键查看其对应在Order的数据字段"status"不为"unpaid"，删除表中该数据，提交事务，重复执行该流程。

(2) 若该数据的对应在Order的数据"createtime"离当前时间不足10s，则关闭事务，重复该流程。

(3) 将该数据对应在Order中的数据字段status的值改为"canceled",并删除该数据，提交事务。

# 八、开发流程

1. 先构建好测试文件，再编写对应的接口功能。若发现其中一项有错，则修改其中一项，直至所有测试准确无误。
2. 利用Github进行协作开发和版本控制，也能有效解决代码冲突问题

# 九、测试结果

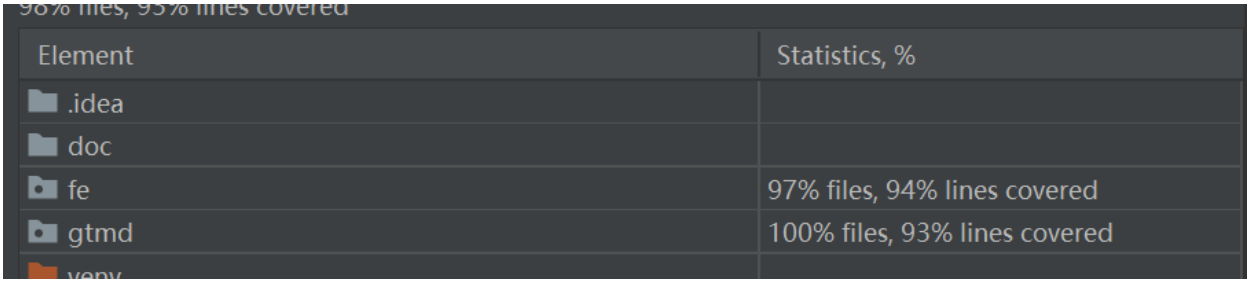
负载吞吐和延迟测试：

```
    # session=1
    WARNING:root:TPS_C=10010, NO=OK:496000 Thread_num:1000 TOTAL:496000
LATENCY:0.08174518053185555 , P=OK:460878 Thread_num:999 TOTAL:495000
LATENCY:0.018163551394144693

    # session=2
    WARNING:root:TPS_C=9004, NO=OK:602880 Thread_num:1000 TOTAL:602880
LATENCY:0.09041327285499266 , P=OK:429383 Thread_num:999 TOTAL:601416
LATENCY:0.020668427774622855

    # session=4
    WARNING:root:TPS_C=6595, NO=OK:1489378 Thread_num:1000 TOTAL:1489378
LATENCY:0.1241836854581099 , P=OK:686536 Thread_num:999 TOTAL:1486326
LATENCY:0.027457024604242715
```

对应的代码覆盖率：




对应test cases: