

穿搭推荐分享小程序 OOTD 交付文档

杜婉晴 2021011824 陈植 2021011798 李昊轩 2021012900 荀芳菲 2021011837

2024 年 1 月 4 日

目录

1 交付产品	2
2 产品目标	2
2.1 功能目标	2
2.1.1 个人空间	3
2.1.2 虚拟衣柜	3
2.1.3 穿搭推荐评分	3
2.1.4 穿搭分享社区	3
2.2 性能目标	3
3 开发组织管理	4
3.1 过程管理	4
3.2 人员分工	4
3.3 开发环境	4
3.4 配置管理	4
3.4.1 git 提交规范	4
3.4.2 分支管理规范	4
4 系统设计	4
4.1 前端交互	4
4.1.1 界面设计原则	4
4.1.2 用户交互流程	5
4.2 后端模块	8
4.3 接口规范	8
5 重难点问题及其解决方法	10
5.1 微信登录	10
5.2 搭配评分	10
5.2.1 搭配评分模型	10
5.2.2 利用搭配评分模型对低分搭配进行替换	10

5.2.3 随机生成推荐穿搭	10
5.2.4 根据温度筛选当季衣服	10
5.2.5 用户穿搭信息统计	10
6 测试总结	11
6.1 功能测试	11
6.1.1 单元测试	11
6.1.2 端到端测试	11
6.2 性能测试	11
6.2.1 虚拟衣柜接口测试	12
6.2.2 穿搭推荐接口测试	12
6.2.3 分享社区相关接口测试	13
7 系统部署	13

1 交付产品

本产品是一款提供穿搭推荐评分和社区分享的微信小程序，小程序体验版二维码如图 1 所示，可通过微信扫码体验。扫码后点击申请体验，申请通过后点击“前往体验版”后，点击右上角的三个点打开选项，选择右下角的开发调试，即可正常使用小程序。

代码仓库 <https://github.com/dwq531/OOTD>

开发过程文档包括：项目例会.pdf、api 文档.pdf、OOTD 需求规格说明书.pdf，均在 doc 文件夹下。

后端数据库管理员界面<http://43.138.127.14:8000/admin/>，用户名 ootd，密码 ootd。



图 1：小程序体验版二维码

2 产品目标

2.1 功能目标

集成虚拟衣柜、穿搭推荐评分和穿搭分享社区三大功能板块。

2.1.1 个人空间

- 微信授权获取用户微信名和微信头像
- 完善、修改个人信息
- 显示衣服数量、发帖数量、收藏数量和获赞数量
- 显示可选时间范围内的穿搭评分折线图
- 显示可选时间范围内的单品使用频率饼状图

2.1.2 虚拟衣柜

- 添加、删除、修改衣服
- 显示衣服列表
- 拖拽衣服至“今日穿搭”栏
- 删 除“今日穿搭”栏的衣服
- 替换“今日穿搭”栏的衣服
- 一键替换穿搭

2.1.3 穿搭推荐评分

- 对当前“今日穿搭”栏选择的衣服进行评分
- 进行局部替换生成基于用户选择的推荐穿搭
- 根据用户填写的城市自动获取天气，生成基于天气的推荐穿搭
- 随机生成推荐穿搭

2.1.4 穿搭分享社区

- 发帖
- 一键分享穿搭评分和天气
- 点赞、评论、收藏
- 浏览社区主页、自己发的帖子和收藏夹

2.2 性能目标

社区功能：支持短时间内每秒 40 个用户发帖和浏览，长时间内每秒 20 个用户浏览贴文
推荐模型：保证在低流量下正确生成推荐穿搭

3 开发组织管理

3.1 过程管理

集中开发安排：每周五 14:00-17:00 集中开发，汇报当前进度，讨论解决本周遇到的问题，确定下周任务及分工

迭代时间线如图 2 所示：

3.2 人员分工

杜婉晴：登录后端、评分算法、虚拟衣柜前端、发帖前端、社区前端、部署

陈植：个人中心前端、资料编辑前端、天气获取后端、个人中心后端、测试

荀芳菲：登录前端、虚拟衣柜后端、编辑衣服后端、汇报

李昊轩：登录后端、发帖后端、社区后端、测试

3.3 开发环境

前端：由微信提供的小程序部署，使用微信开发者工具进行开发和测试

后端：利用 docker 部署在助教提供的服务器上，数据库采用 MySQL，后端框架使用 Django 进行开发

前后端通信：基于微信小程序提供的访问接口，以及自己写的 api

3.4 配置管理

3.4.1 git 提交规范

< 提交类型 >: < 提交内容 >。提交类型包括：init（初始化）、feat（新功能）、fix（修复 bug）、docs（文档）、refactor（重构）、test（测试）等。

3.4.2 分支管理规范

main 分支：发布分支，合并需要进行 pull request

dev 分支：开发分支，写的功能需要在 dev 上新建自己的分支，写完进行 pull request 合并在 dev 分支上，dev 分支功能达到能发布的程度后合并到 main 分支上

4 系统设计

4.1 前端交互

4.1.1 界面设计原则

配色方案：本软件采用蓝粉白色为主色调，符合本软件主要用户群体的审美，带给用户清新时尚的美感。

	10.27	11.3	11.9	11.17	11.24	12.1	12.8	12.15	1.2
登录	前端、后端			前后端连接					
推荐模型			推荐模型测试			推荐模型接入			
虚拟衣柜		前端			后端		前端、后端连接		
个人主页			前端、后端				前端、后端连接		
社区					前端、后端		前端、后端连接		
测试部署									前端、后端连接

图 2: 迭代时间线

一致性：本软件的界面设计风格统一，在按钮配色、字体风格颜色、图标形状风格等方面保持较高的一致性。

简洁性：本软件的界面设计风格简洁，不会出现过多的文字和图标，保证用户在使用时不会出现视觉疲劳。

导航清晰：本软件的图标含义明确，功能模块具有文字说明提示，用户可以根据自己的需求快速找到所需功能。

4.1.2 用户交互流程

登录

进入小程序页面后，点击“抽全登录按钮”进行微信登录。如果是新用户，则会跳转到授权页面，点击头像弹出授权框，点击“允许”即可授权获取微信头像和昵称。如果是老用户，则会直接跳转到衣柜页面。



Outfit Of ToDay

申请获取以下权限
获得你的公开信息(昵称, 头像等)

授权登录

图 3: 登录

虚拟衣柜

在衣柜页面，用户可以点击“添加衣服”按钮，跳转到添加衣服页面，点击相机按钮上传一张衣服图片，填写衣服名称并选择主种类和细分种类，点击保存按钮即可保存到衣柜中。

用户可以点击衣柜中的衣服，跳转到衣服详情页面，点击“删除”按钮弹出确认框，点击“确定”即可删除衣服；编辑衣服信息的流程和添加衣服相似。

点击左侧导航栏，可以切换显示的衣服主类别。长按衣服并拖拽至上方“今日穿搭”板块，即可把衣服加入穿搭，拖拽过程会显示衣服图片。添加穿搭时，相同类型衣服会进行替换。点击“今日穿搭”内衣服右下角的减号按钮，可删除衣服。

点击“评分”按钮，对“今日穿搭”内的衣服进行搭配评分，过程中会显示“评分中”加载框，评分完成后会在右上角显示分数。如果有分数更高的替换搭配，会弹窗提示用户是否替换，点击“替换”按钮即可替换，点击“取消”按钮则不替换。

点击“推荐穿搭”按钮，以用户拥有的衣服为基准生成一套分数较高的随机穿搭，显示在“今日穿搭”内，并显示分数。如果当季衣服过少，会弹出提示框，点击“确定”按钮即可关闭提示框。

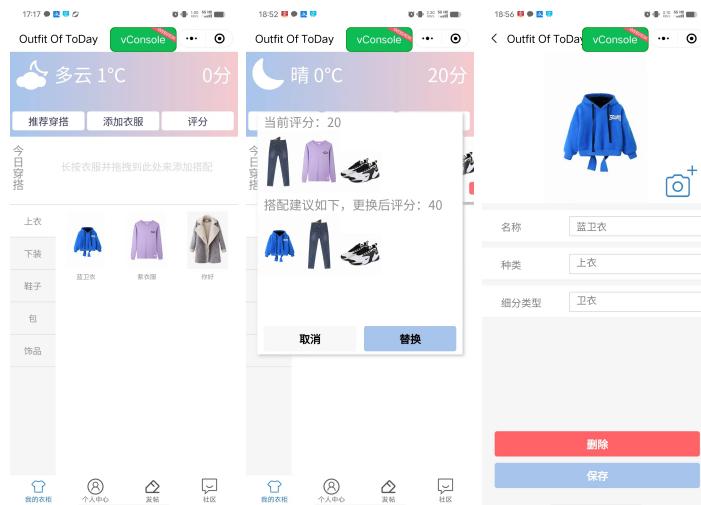


图 4: 衣柜

个人中心

个人中心页面显示用户头像、昵称、性别、年龄、城市等个人信息，并统计用户衣服数量、收藏数量、帖子数量、获赞数量等信息，展示搭配评分折线图和常穿衣服饼状图。

点击“编辑资料”按钮，跳转到修改个人信息页面。点击头像旁边相机按钮，上传新头像；通过输入框，可以修改昵称、手机号、地点、年龄，性别可通过选择框修改。点击“保存”按钮，若格式正确则返回个人主页；若格式错误则弹窗显示错误信息。

点击统计图左上角的选择框，可切换统计的时间限制，如最近一周、最近两周和最近一月。

发帖

点击加号，可上传最多 9 张图片，点击图片右上角的减号可删除图片。



图 5: 个人主页

在“添加标题”输入框中输入标题，在“添加内容”输入框中输入帖子内容，可点击“展示评分”和“分享天气”多选框来选择是否展示评分和分享天气。

点击“发表”按钮，若标题和图片不为空，则发布成功，跳转到社区页面；若标题或内容为空，则弹窗提示用户输入标题和上传图片。



图 6: 发帖

社区

点击页面上方导航栏可切换显示的帖子类型，包括“全部”、“我的”和“收藏”。

上下滑动可浏览帖子封面，包括封面图、标题、发帖者头像和昵称、点赞数、评论数、收藏数等信息。

点击帖子封面，可跳转到帖子详情页面，展示帖子内容、发帖时间、天气和 AI 评分、评论等。左右

滑动帖子详情页面上方的图片展示框，可浏览所有图片。点击点赞和收藏按钮可切换点赞和收藏状态。在评论框内输入评论，点击“发送”按钮可发表评论。

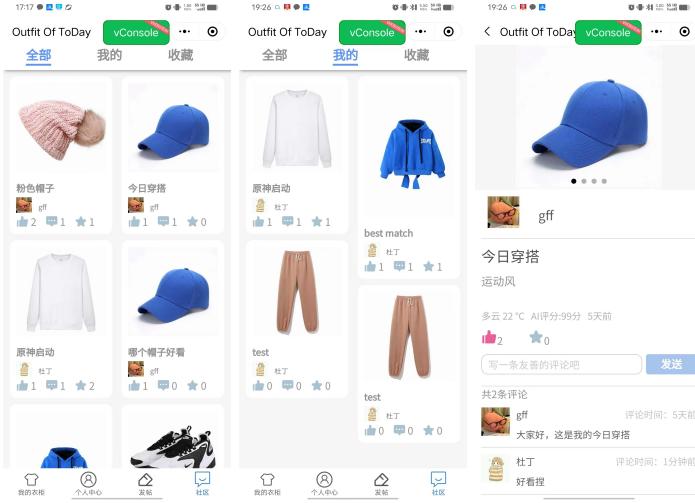


图 7: 社区

4.2 后端模块

后端采用 Django 框架，MySQL 数据库，定义好模型后通过 migrate 命令自动创建数据库表。后端由三个模块组成，分别是用户模块 (login)、衣柜模块 (OOTD_closet) 和帖子模块 (posting)。用户模块负责用户的登录、注册、修改个人信息等功能；衣服模块负责衣服的增删改查、穿搭推荐等功能；帖子模块负责帖子的发布、浏览、点赞、评论等功能。主要模型类关系如图 8 所示。

用户模块包含两个模型类，分别是用户类 (User) 和天气类 (Weather)。User 类记录用户的个人信息，包括地址、年龄、头像、性别、昵称、手机号等。Weather 类记录用户所在地区的天气信息，包括温度、天气等。

衣柜模块包括三个模型类，分别是衣服类 (Clothes)、每日穿搭类 (DailyOutfit) 和替换穿搭类 (ReplaceOutfit)。Clothes 类记录衣服的信息，包括图片、名称、主类别、细分类别等，并关联到拥有它的用户。DailyOutfit 类记录每日穿搭的信息，包括搭配衣服、穿搭时间、穿搭评分等，每天记录最后一套。ReplaceOutfit 类记录替换搭配的信息，包括穿搭、评分等，暂时存储评分算法给出的替换搭配。

社区模块包括三个模型类，分别是帖子类 (Post)、评论类 (Comment) 和图片类 (Image)。Post 类记录帖子的信息，包括标题、内容、发帖时间、点赞、收藏、是否展示天气和评分等，并关联到发帖者、评论和图片。Comment 类记录评论的信息，包括评论内容、评论时间、评论用户等，并关联到评论者和帖子。Image 类记录帖子图片的信息，并关联到帖子。

4.3 接口规范

我们的小程序前端和后端之间通过 HTTP 协议进行通信，前端通过发送 HTTP 请求，后端通过 HTTP 响应进行回复。我们的后端接口路径按照用户、衣柜和社区分为“api/user/”、“api/closet/”和“api/posting/”三个部分，每个部分下面又有不同的接口，详情见 api 文档。

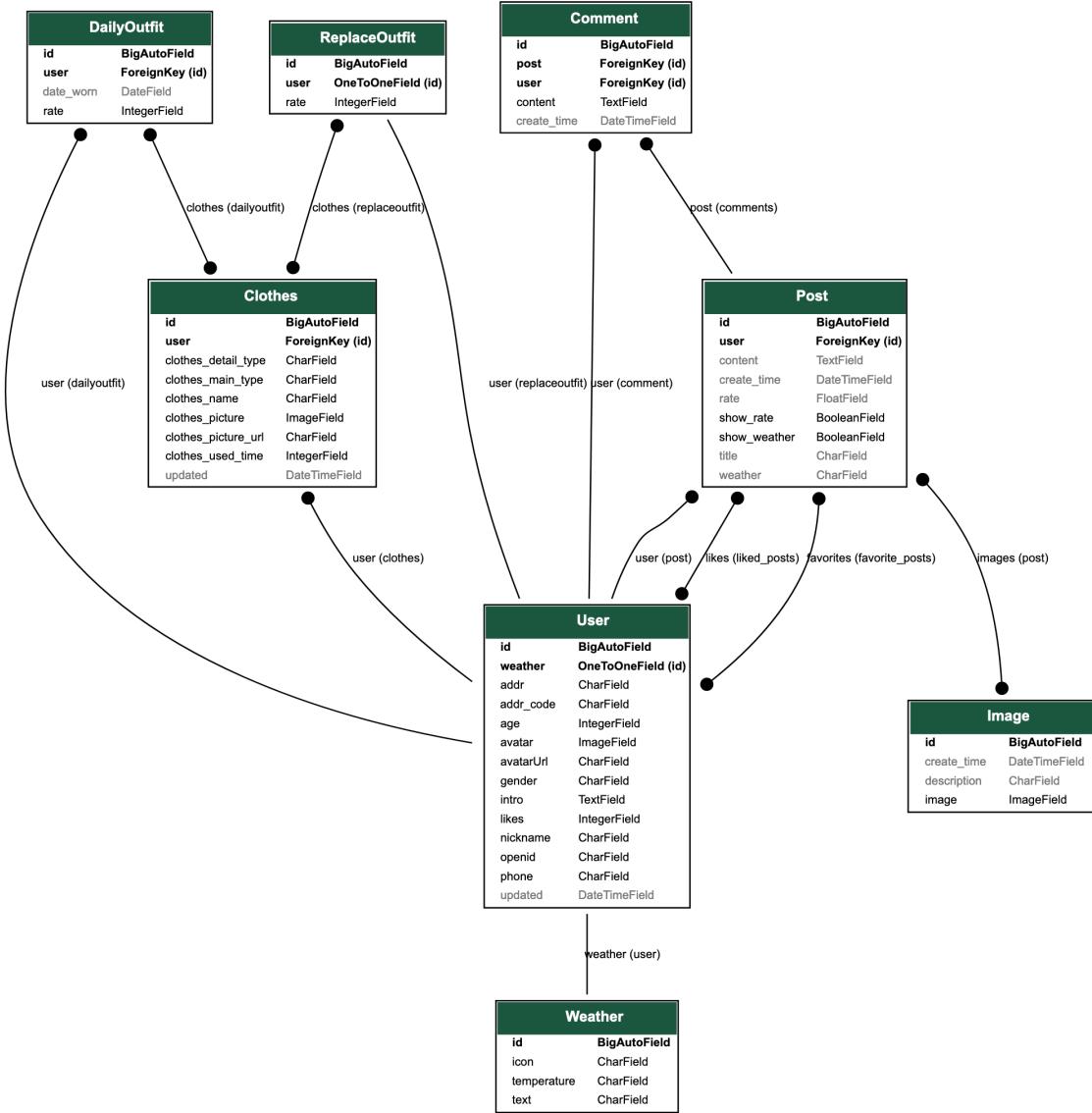


图 8: 模型类关系图

5 重难点问题及其解决方法

5.1 微信登录

我们使用微信提供的 api 实现了微信登录功能。用户点击登录按钮后，小程序会调用 `wx.login()` 方法获取临时登录凭证 `code`，然后调用 `wx.request()` 方法向后端发送 `code`，后端通过 `code` 调用微信提供的 api 获取用户的 `openid` 和 `session_key`，用 `openid` 生成 jwt 返回给前端，前端将 jwt 存储在本地，后续的请求都会带上 jwt，后端通过 jwt 解析出 `openid`，从而识别用户。

这种方法既利用了微信登录的便利性，避免了记住和输入用户名密码的烦恼，又能保证用户信息的安全性。

5.2 搭配评分

5.2.1 搭配评分模型

我们引用了论文 [1] 所实现的服装搭配评分模型，该模型基于深度学习，通过对服装图片进行特征提取，然后通过多层比较网络对服装进行搭配评分。该模型的优点是（1）通过学习所有类型指定的单品之间的成对相似性来学习整体兼容性，使用反向传播梯度来诊断不兼容的因素。（2）利用 CNN 的层次结构，比较服装从底层特征（如颜色、纹理）到高层（如风格）不同方面的兼容性。

该模型的输入是五张不同类型衣服的图片，分别为上衣、下装、鞋、包、饰品，如果某种类型有空缺则用平均值代替。输出为 0~1 之间的搭配评分和每种类型衣服的搭配度，搭配度越高表示该衣服越适合当前搭配。

5.2.2 利用搭配评分模型对低分搭配进行替换

如果评分小于 90 分，评分算法会尝试替换个别服装来提升评分。按照每件衣服在当前搭配中的搭配度进行排名，从最不搭配的衣服开始依次尝试进行替换，直到找到最高分或分数超过 90 为止。

5.2.3 随机生成推荐穿搭

随机选择一件上衣并以它为基准，遍历其他类型的衣服并选出分数最高的一套搭配。

5.2.4 根据温度筛选当季衣服

在进行以上两种算法前，会根据衣服对衣柜中的所有衣服进行预筛选，选出当季衣服进行搭配和替换。为每种衣服设定适宜穿的温度范围（为了便于搭配范围可能会稍大一些），如 t 恤在 20 度~40 度，羽绒服在 -30 度~10 度等。一些季节属性较弱的衣服种类则不设置温度范围以便生成搭配，如正装、休闲裤、饰品等。

5.3 用户穿搭信息统计

总体思路上，使用 ECharts 进行折线图和饼状图的绘制，并将其渲染到指定的 `canvas` 上。具体实现上，用 `initChart` 函数初始化图表，创建 `Chart` 实例，并通过调用 `setOption` 方法，根据传入数据设置图表的配置项，并在小程序页面中使用 `<ec-canvas>` 组件的标签引入图表。

6 测试总结

6.1 功能测试

6.1.1 单元测试

采用 Python 的标准模块 unittest，总覆盖率约 68%，测试用例见图 9 图 10，详细覆盖率报告见测试报告。

待测 API	API 功能	请求类型	测试用例
edit_info	修改用户信息	PATCH	数据正确 数据错误：性别填写格式有误 数据错误：年龄为负数 数据错误：电话号码长度有误 数据错误：地点不存在
user	获取用户信息	GET	请求无需发送数据
get_weather	获取天气	GET	请求无需发送数据

图 9: OOTD_backend/login/tests.py

待测 API	API 功能	请求类型	测试用例
get_clothes	获取衣服列表信息	GET	请求无需发送数据
add_outfit	将衣柜内的衣服添加至搭配栏	POST	数据正确：添加上衣 1 数据正确：添加裤子 数据正确：添加鞋子 数据正确：添加帽子 数据错误：发送的 clothes id 为负数
remove_outfit	移除搭配栏中的指定衣服	POST	数据正确：移除帽子 数据错误：发送的 clothes id 为负数
get_outfit	获取搭配栏内的衣服信息	GET	请求无需发送数据
score	给穿搭评分	POST	请求无需发送数据
generate	生成推荐穿搭	POST	请求无需发送数据

图 10: OOTD_backend/OOTD_closet/tests.py

6.1.2 端到端测试

端到端测试采用人工测试的方法，测试用例见图 11。

6.2 性能测试

本项目性能测试通过划分子系统分为三个部分，测试服务接口在高并发访问下是否还能正常工作、响应速度如何。以下所有测试中，均没有出现错误，即服务接口总是被正确调用的。

值得一提的是，在分享社区接口测试中，我们根据现实情况模拟了两种高并发场景。一是大量用户在短时间内创建贴文并浏览，二是大量用户在长时间内不断浏览社区和贴文。在两种测试中，系统的表现都令人满意。

页面	功能		测试结果	
登录界面	登录获取用户信息		通过	
我的衣柜	衣柜	添加衣服	通过	
		编辑衣服	通过	
		删除衣服	通过	
	搭配栏	拖拽添加衣服	通过	
		删除衣服	通过	
		同类衣服自动替换	通过	
	穿搭评分		通过	
	生成推荐穿搭		通过	
	一键替换穿搭		通过	
个人中心	展示天气		通过	
	个人信息显示		通过	
	搭配评分折线图显示		通过	
资料编辑	最常穿衣服饼状图显示		通过	
	上传头像		通过	
发帖	修改昵称、手机号、地点、年龄、性别		通过	
	添加图片		通过	
	添加标题、内容描述		通过	
	展示评分		通过	
	展示天气		通过	
社区	全部	分享天气		通过
		浏览帖子		通过
		点赞		通过
		收藏		通过
	评论			通过
	收藏栏	已收藏帖子展示正常		通过
	我的	显示当前用户的帖子		通过

图 11: 端到端测试

6.2.1 虚拟衣柜接口测试

模拟用户的行为如下：1. 编辑衣物信息（停顿 1-2 秒）2. 上传衣物图片和信息 3. 重复前两步，直到已经上传 8 件衣物 4. 浏览虚拟衣柜当在 5 秒内产生 100 个线程来对服务器进行访问时，得到的结果如图 12 所示，各项请求的响应时长的最大值都在 0.5 秒以内，响应十分迅速和稳定，满足要求。

Label	Samples	Average	Median	90% Line	95% Line	Min	Max
add clothes	800	115	107	164	184	62	443
get clothes	100	85	78	103	124	71	151
TOTAL	800	115	107	164	184	62	443

图 12: 虚拟衣柜性能测试结果

6.2.2 穿搭推荐接口测试

经过大量测试，部署在服务器上的推荐算法的响应时长在 0.5 秒左右，基本满足要求。但由于推荐算法是基于一个开源深度学习模型实现的，在服务器资源有限的情况下并不能很好的满足并发请求要求。当 5 个线程同时请求时，平均响应时长会来到 2 秒左右

6.2.3 分享社区相关接口测试

模拟用户的行为如下：1. 创建一个帖子，帖子包含三张图片 2. 浏览贴文列表 3. 翻阅（随机停顿 1-4 秒）4. 打开一个帖子浏览详情当在 5 秒内产生 200 个线程来对服务器进行访问时，得到的结果如图 13 所示，各项请求的平均响应时长在 1 秒所有，最大响应时长在 2 秒左右，满足要求。

Label	Samples	Average	Median	90% Line	95% Line	Min	Max
create post	200	927	897	1746	1863	94	1997
upload image	600	1272	1393	1969	1994	96	2060
user posts	200	1014	813	1949	1989	56	2052
post detail	200	883	510	1918	1963	51	2025
TOTAL	1200	1107	1124	1951	1988	51	2060

图 13: 创建帖子性能测试结果

接下来打开 100 个线程，每个线程重复浏览贴文的操作，持续 2 分钟，结果如图 14 所示，此时服务的平均响应时长在 2 秒左右，95% 的响应时长在 2.5 秒左右，符合要求。

Label	Samples	Average	Median	90% Line	95% Line	Min	Max
user posts	1611	2240	2288	2574	2675	144	3184
post detail	1584	2094	2125	2392	2482	52	2661
TOTAL	3195	2168	2202	2499	2587	52	3184

图 14: 浏览帖子性能测试结果

7 系统部署

后端采用 docker 部署，分别配置 app (OOTD 后端)、db (MySQL 数据库)、nginx 三个镜像。在 networks 部分，定义 web_network 和 db_network 两个网络，分别为 nginx 与 app、app 与数据库间的网络。配置文件为 docker-compose.yml。访问地址为 43.138.127.14:8000。

app 镜像在 Dockerfile 中配置，将项目文件夹中的所有文件复制到镜像中，按照 requirement.txt 安装依赖。启动时运行 Django 的数据库迁移命令，并启动 Gunicorn。

db 镜像采用官方的 MySQL8.1 镜像，配置数据库的用户名为 root，数据库名称和密码均为 ootd，并将数据库文件挂载到本地。

nginx 镜像采用官方的 nginx 镜像，配置文件为 nginx/app.conf，监听端口 8000，设置最大连接数和上传文件最大大小。

参考文献

- [1] Bo Wu Xin Wang and Yueqi Zhong. Outfit compatibility prediction and diagnosis with multi-layered comparison network. In *ACM International Conference on Multimedia*, 2019.