文件编号：3107-SWC2018-20180045

受控状态：■受控 □非受控

保密级别：□公司级 □部门级 ■项目级 □普通级

采纳标准：CMMI DEV V1.2



图美集

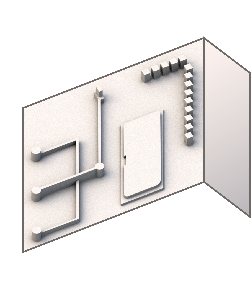
**Temage**

项目开发文档

**Version 1.2.1**

2019.5.9

**Written by 3107**



**All Rights Reserved**

目录

[1 引言 1](#_Toc8315814)

[1.1 编写目的 1](#_Toc8315815)

[1.2 项目概述 1](#_Toc8315816)

[1.3 项目背景 1](#_Toc8315817)

[1.4 术语和缩略语 2](#_Toc8315818)

[1.5 参考资料 2](#_Toc8315819)

[1.6 项目定位 3](#_Toc8315820)

[1.6.1 应用场景 3](#_Toc8315821)

[1.6.2 目标人群 3](#_Toc8315822)

[1.7 项目目标 3](#_Toc8315823)

[1.8 项目价值 4](#_Toc8315824)

[1.9 创新点 5](#_Toc8315825)

[1.9.1 功能性创新 5](#_Toc8315826)

[1.9.2 非功能性创新 5](#_Toc8315827)

[1.10 解决思路 6](#_Toc8315828)

[1.11 系统亮点 6](#_Toc8315829)

[1.11.1 功能性亮点 6](#_Toc8315830)

[1.11.2 非功能性亮点 6](#_Toc8315831)

[2 开发计划 7](#_Toc8315832)

[2.1 最终呈现形式 7](#_Toc8315833)

[2.2 主要功能描述 7](#_Toc8315834)

[2.3 运行环境 8](#_Toc8315835)

[2.4 验收标准 8](#_Toc8315836)

[2.5 关键问题 8](#_Toc8315837)

[2.6 进度安排： 9](#_Toc8315838)

[2.7 开发预算 9](#_Toc8315839)

[3 可行性分析 10](#_Toc8315840)

[3.1 市场可行性分析 10](#_Toc8315841)

[3.1.1 与现行软件对比 10](#_Toc8315842)

[3.2 技术可行性分析 10](#_Toc8315843)

[3.2.1 功能简述 10](#_Toc8315844)

[3.2.2 技术要素 10](#_Toc8315845)

[3.3 资源可行性分析 11](#_Toc8315846)

[4 需求分析 11](#_Toc8315847)

[4.1 数据描述 11](#_Toc8315848)

[4.1.1 静态数据 11](#_Toc8315849)

[4.1.2 动态数据 12](#_Toc8315850)

[4.1.3 数据词典 14](#_Toc8315851)

[4.1.4 数据采集 17](#_Toc8315852)

[4.2 功能需求 18](#_Toc8315853)

[4.2.1 功能模块结构图 18](#_Toc8315854)

[4.2.2 核心功能模块描述 18](#_Toc8315855)

[4.3 非功能性需求 19](#_Toc8315856)

[4.3.1 扩展性和可维护性 19](#_Toc8315857)

[4.3.2 易用性 19](#_Toc8315858)

[4.4 性能需求 19](#_Toc8315859)

[4.4.1 时间特性 19](#_Toc8315860)

[4.4.2 适应性 19](#_Toc8315861)

[4.5 运行需求 20](#_Toc8315862)

[4.5.1 用户界面 20](#_Toc8315863)

[4.5.2 硬件接口 22](#_Toc8315864)

[4.5.3 软件接口 22](#_Toc8315865)

[4.5.4 故障处理 22](#_Toc8315866)

[4.6 用例图 23](#_Toc8315867)

[4.7 核心模块用例规约 23](#_Toc8315868)

[4.8 其他需求 27](#_Toc8315869)

[5 概要设计 28](#_Toc8315870)

[5.1 处理流程 28](#_Toc8315871)

[5.2 总体结构和模块设计 29](#_Toc8315872)

[5.3 功能分配 30](#_Toc8315873)

[5.4 接口设计 31](#_Toc8315874)

[5.4.1 外部接口 31](#_Toc8315875)

[5.4.2 内部接口 40](#_Toc8315876)

[5.4.3 用户界面设计 43](#_Toc8315877)

[5.5 数据结构设计 43](#_Toc8315878)

[5.5.1 逻辑结构设计 43](#_Toc8315879)

[5.5.2 物理结构设计 45](#_Toc8315880)

[5.5.3 数据结构与程序的关系 48](#_Toc8315881)

[5.6 运行设计 49](#_Toc8315882)

[5.6.1 运行模块关系 49](#_Toc8315883)

[5.6.2 运行控制 49](#_Toc8315884)

[5.6.3 运行时间 49](#_Toc8315885)

[5.7 错误/异常处理设计 50](#_Toc8315886)

[5.7.1 错误/异常输出信息 50](#_Toc8315887)

[5.7.2 错误/异常处理对策 51](#_Toc8315888)

[5.8 系统配置策略 52](#_Toc8315889)

[5.9 系统部署方案 53](#_Toc8315890)

[5.10 代码规范 55](#_Toc8315891)

[5.11 其他相关技术与方案 55](#_Toc8315892)

[6 数据库设计 55](#_Toc8315893)

[7 详细设计 59](#_Toc8315894)

[7.1 图文排版功能模块 60](#_Toc8315895)

[7.1.1 功能描述 60](#_Toc8315896)

[7.1.2 性能描述 60](#_Toc8315897)

[7.1.3 输入 61](#_Toc8315898)

[7.1.4 输出 61](#_Toc8315899)

[7.1.5 算法 62](#_Toc8315900)

[7.1.6 程序逻辑 62](#_Toc8315901)

[7.1.7 接口 63](#_Toc8315902)

[7.1.8 存储分配 63](#_Toc8315903)

[7.1.9 限制条件 63](#_Toc8315904)

[7.2 用户社区模块 64](#_Toc8315905)

[7.2.1 功能描述 64](#_Toc8315906)

[7.2.2 性能描述 64](#_Toc8315907)

[7.2.3 输入 64](#_Toc8315908)

[7.2.4 输出 65](#_Toc8315909)

[7.2.5 算法 65](#_Toc8315910)

[7.2.6 程序逻辑 65](#_Toc8315911)

[7.2.7 接口 65](#_Toc8315912)

[7.2.8 存储分配 66](#_Toc8315913)

[7.2.9 限制条件 66](#_Toc8315914)

记录更改历史

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **更改原因** | **版本** | **作者** | **更改日期** | **备 注** |
| 1 | 创建 | 1.0.0 | 队员D | 2018/11/10 | 创建项目开发文档 |
| 2 | 更新 | 1.0.1 | 队员D | 2018/11/12 | 文档的第一第二部分撰写 |
| 3 | 更新 | 1.0.2 | 队员B | 2018/11/13 | 文档的第四部分撰写 |
| 4 | 更新 | 1.0.3 | 队员D | 2018/11/14 | 文档的第三部分撰写 |
| 5 | 更新 | 1.0.4 | 队员D | 2018/11/18 | 文档整体修改 |
| 6 | 更新 | 1.0.5 | 队员D | 2018/11/19 | 文档细节修改 |
| 7 | 更新 | 1.0.6 | 队员B | 2018/11/20 | 调整图片 |
| 8 | 更新 | 1.1.0 | 队员B | 2019/02/17 | 修改文档初赛部分 |
| 9 | 更新 | 1.1.1 | 队员B | 2019/02/17 | 撰写错误/异常处理部分 |
| 10 | 更新 | 1.1.2 | 队员D | 2019/02/22 | 撰写数据库设计部分 |
| 11 | 更新 | 1.1.3 | 队员B | 2019/02/26 | 撰写详细设计部分 |
| 12 | 更新 | 1.1.4 | 队员B | 2019/02/27 | 完成复赛需要提交的部分 |
| 13 | 更新 | 1.1.5 | 队员D | 2019/02/28 | 文档细节修改 |
| 14 | 更新 | 1.1.6 | 队员D | 2019/03/01 | 文档细节修改 |
| 15 | 更新 | 1.2.0 | 队员B | 2019/04/09 | 更新文档排版 |
| 16 | 更新 | 1.2.1 | 队员B | 2019/05/07 | 修改文件细节 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 引言

## 编写目的

此开发文档的编写目的是对Temage（图美集）智能图文排版web软件的各方面做详细的介绍、分析与总结。本团队从问题出发，确立项目的目标、定位和构想，细化项目的 c搭建方案。开发文档包含了详细的市场分析、功能介绍、和设计细节。通过此开发文档，可以更深入地了解项目的理念；同时本团队也希望此文档能够有助于规范项目的管理，保证项目质量，为开发、使用和维护提供有效的指导、解惑和帮助。

## 项目概述

Temage（Text + Image，意为文本与图片的完美融合，简称Temage）智能图文排版是一个运用深度学习技术实现自动图文排版的web应用。用户给定文本和图片，即可根据文本的主题，在线智能排版文本与图片，选择合适的风格，生成一篇优美悦目的文章。能够极大地节省文章从排版到发布的时间，提高用户的效率和生产力。

Temage针对当下大众资讯获取、媒体信息传播的新变化，具有很好的应用前景和广大的目标用户群体。

Temage注重用户的体验，旨在为用户提供优美友好的界面和优质流畅的服务。

## 项目背景

当今是一个社交媒体的时代。得益于技术的日新月异，信息传播从未如此迅捷。自Facebook、Twitter始，再到微信、微博、今日头条等常用的社交平台，人们每天浏览、转发、评论的文章不计其数。根据统计，早在2016年，微信公众号账号数量就已近2000万，活跃用户数量更是达到了7.62亿，如今这些数据还在高速增长。在这样的大环境下，自媒体等行业的发展也得到了极大的推动。同时政府、企业、社会团体、组织等也纷纷在各平台发声，通过文章向公众传达信息。

不管是生活中还是工作中，越来越多的人需要撰写、发布文章。而其中一个重要环节就是排版，优秀的排版往往能够吸引大众眼球，让文章锦上添花。对此本团队主要针对在微信公众号、微博等平台发布文章的用户进行分析与调研，发现存在以下几点问题：

1. 排版需要消耗用户不少的时间和精力，排版过程枯燥乏味。
2. 没有设计的基础的用户，很难做出优秀的版面设计。
3. 排版可能会与文章主题不搭配，排版效果不佳。

针对上述问题，有一些现有的软件正着手于解决这些难题。现在用户常用的排版软件，如秀米图文排版等，需要用户手动选取样式和装饰组件，手动插入图片。用户仍需要大量的操作才能得到满意的排版，文章的整体结构布局仍旧需要用户手动进行设计，并不能完全解决用户的问题，简化图文排版的工作和流程。

为解决用户排版难题，结合当下大热的深度学习技术，本团队构想出Temage，期望它能根据输入的文本和图片，自动为用户提供专业且合适的版面设计，生成优质的图文排版，为用户节省90%以上的排版时间。同时降低版面设计的门槛，使没有美工基础的用户也能发布精品文章，还能降低自媒体创业的成本，为自媒体行业加速。除此之外，Temage紧跟时代设计的风向与潮流，对同时代的版面设计进行学习，不断提升人工智能设计水准。

## 术语和缩略语

1. Temage：Temage（图美集）是一个在线自动排版的web应用，旨在为用户发布文章提供快捷优质服务。
2. Tensorflow： Tensorflow是一个采用数据流图(data flow graphs),用于数值计算的开源软件库。
3. Keras：Keras是一个高层神经网络API，Keras由纯Python编写而成并基于Tensorflow、Theano以及CNTK后端。
4. Tornado: Tornado 是一种 Web 服务器软件的开源版本。Tornado 和现在的主流 Web 服务 器框架（包括大多数 Python 的框架）有着明显的区别：它是非阻塞式服务器，而且速度相 当快。
5. Django: Django 是 Python 编程语言驱动的一个开源模型-视图-控制器（MVC）风格的 Web 应用程序框架。
6. Kubernetes: Kubernetes简称K8s，是用8代替8个字符“ubernete”而成的缩写。是一个开源的，用于管理云平台中多个主机上的容器化的应用，Kubernetes的目标是让部署容器化的应用简单并且高效（powerful）,Kubernetes提供了应用部署，规划，更新，维护的一种机制。
7. Istio: Istio提供了一个完整的解决方案，通过为整个服务网格提供行为洞察和操作控制来满足微服务应用程序的多样化需求。Istio 有助于降低这些部署的复杂性，并减轻开发团队的压力。它是一个完全开源的服务网格，可以透明地分层到现有的分布式应用程序上。它也是一个平台，包括允许它集成到任何日志记录平台、遥测或策略系统的 API。Istio 的多样化功能集使您能够成功高效地运行分布式Microservice架构，并提供保护、连接和监控Microservice的统一方法。

## 参考资料

[1] TalkingData. 2016年自媒体行业洞察报告[R]

[2] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[3] Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations[J]. arXiv preprint arXiv:1802.05365, 2018.

[4] Lai S, Xu L, Liu K, et al. Recurrent Convolutional Neural Networks for Text Classification[C]//AAAI. 2015, 333: 2267-2273.

[6] Wang L, Li Y, Huang J, et al. Learning two-branch neural networks for image-text matching tasks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019, 41(2): 394-407.

## 项目定位

### 应用场景

无论是在工作中还是在日常生活中，当用户想要缩短网络带图文章发布进程、节省时间精力成本时，当没有设计基础的用户想要做出精美排版时，当用户不知道如何贴合文章主题设计版面时，当用户被乏味的排版工作包围时，Temage都可以为用户提供专业、优质、个性化的排版服务和强大的管理服务。用户仅需要上传图片和文本，90%以上的排版工作，Temage来为用户完成。

### 目标人群

此项目适用于有文章排版需求的人，主要包括：

1. 自媒体运营者/团队，帮助他们节省时间、降低成本，提高工作效率。鼓励普通大众发声，成为为自媒体创作者。
2. 企业的宣传部门文案工作者。服务企业，打造与企业自己匹配的风格。
3. 门户网站、媒体的专业编辑。为其提供便捷的编辑服务和灵感参考。

## 项目目标

Temage致力于实现一个综合、迅捷、便利的图文自动排版系统。

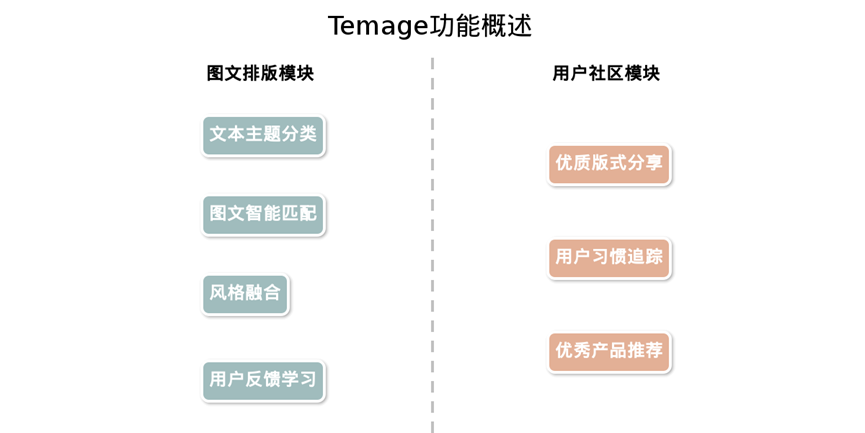


图 1.1 Temage功能概述

首先，Temage通过对文本的分析，对文本主题进行智能分类，分类将很大程度上影响到文章的排版和美化。

同时，Temage能够对从文本、图片中提取的关键信息进行分析，自动将图片嵌入至文章合适的位置，实现初步的图文排版。

最终，Temage能够输出长图、PDF等方便用户发布、分享的文件格式，加速自媒体行业。

Temage注重用户个性化体验，从用户的反馈中学习，对用户习惯进行追踪。

Temage用户可以既可以将满意的作品分享至社区，也可借鉴社区中其他优秀作品。 Temage还可以根据用户近期的收藏和作品，为其推荐社区中同类型的优秀作品。

根据本团队的调查统计，用户对Temage的要求如下图所示：

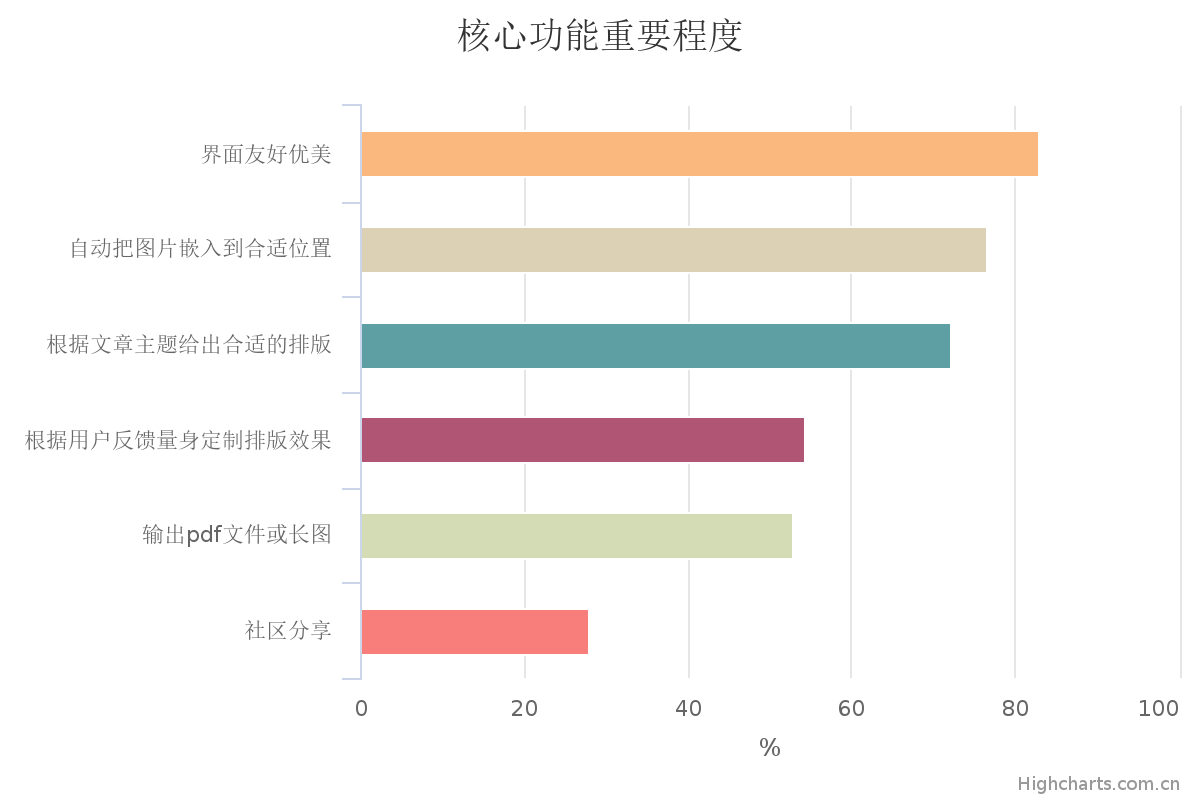


图 1.2 用户心目中的核心功能重要程度

Temage在实现全部核心功能的基础上，还需要根据上述调查结果对界面、排版功能进行重点优化。做到界面简洁易用，服务优质迅速。

## 项目价值

在当今这个信息交流迅捷普遍，社交媒体当道的大环境下，此项目具有广大的潜在用户群体。市场上也正好缺少像Temage这样功能完善、服务优质的智能图文排版软件，

根据发放的调查问卷，被调查人群中的大部分常进行文案创作，其中排版时长占创作总时长比例超过20%的超过了半数。有85%的受调查者表示很期待Temage的诞生。

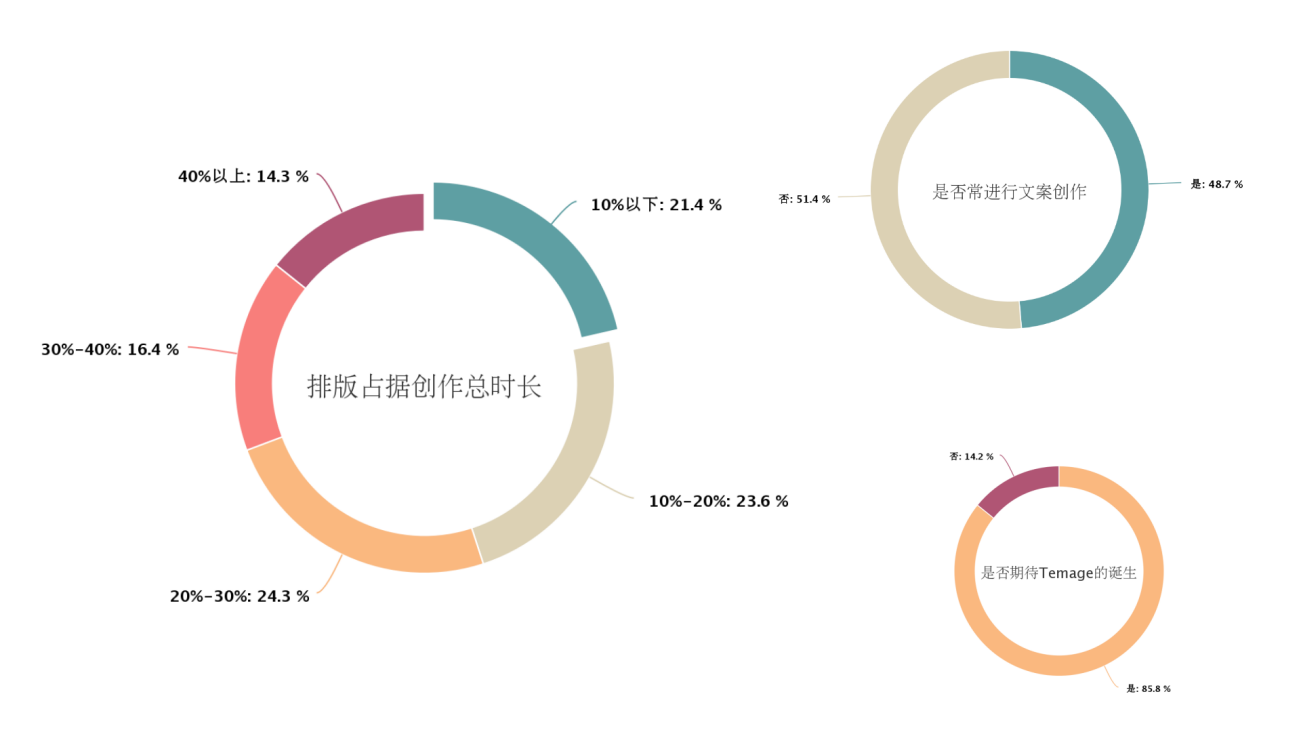


图 1.3 调查报告结果分析

## 创新点

### 功能性创新

Temage将排版与深度学习技术结合。

首先可以通过分析文章，对其主题进行分类，推荐最符合文章主题的版式以供用户选择。

其次，对输入的图片进行信息压缩、提取，与文字相匹配，找到图片在文字最合适的位置，嵌入文章中得到初排版。结合初排版和选取的版式，自动生成排版结果。

排版完成后允许用户根据自身需求对结果微调。用户对结果进行评分，反馈传入系统进行迭代，改善应用。

用户社区模块允许用户相互分享优秀的产品，帮助用户快速上手。通过分析用户行为，Temage为用户推荐社区中同类型的排版案例作为借鉴。

### 非功能性创新

分布式推断的理念是用户在寻求服务的同时也应提供一定的计算资源。用户本地即前端用Tensorflow.js调用模型进行推断，后端使用Tensorflow、pyTorch调用模型进行推断，相互协作，分摊计算压力，减少运行所需时间，保证应用的流畅性和稳定性，提高用户的使用体验。

在商业理念上，Temage保留用户对自己作品的知识产权。用户有公开或交易自己版权的权利，营造一个开放、包容、多样化的创意市场。

Temage有着完善、优质、创新的服务理念和全面的业务服务逻辑模块, 形成了完整和具有潜力的商业设想.   
 1. 图文的排版业务  
 结合深度学习以及AI+的热门技术和概念、Temage利用精准, 高效的机器学习模型处理繁琐乏味的排版任务。实现了AI+排版的颠覆性的创新使用模式  
 2. 社区化建设业务  
 Temage秉承开放, 共享, 多彩的社区精神。在社区画廊中整合所有用户创建的优质 作品/为用户提供充足的创意灵感, 提高用户粘性。塑造出Temage智能, 丰富, 优雅的创意灵感工厂品牌形象。  
 3. 作品分享传播业务

用户可以以多种格式(长图，html，pdf等)下载或发布其作品到微信公众号,个人微博等社交媒体上。发布的作品附带用户水印和Temage商业标识。完善排版产品的生产——发布流程,构建更广大的品牌宣传网络, 提高Temage在互联网生态圈中的活力。

## 解决思路

Temage将问题分解为文本分类、图文匹配和搜索推荐三个子问题。

风格推荐问题，基于用户文本输入和用户习惯，可以分别使用以TextCNN和LSTM为基础的成熟模型进行训练、推断。推荐问题Temage可通过文本分类的结果进行top-k推荐。

图文匹配的问题可分为图像识别，文本嵌入和图文匹配三个子问题。图像识别问题可以使用是使深度学习在众多机器学习算法中脱颖而出的 CNN 模型，基于CNN开发的模型种类繁多，可供本项目进行挑选和fine-tuning。文本嵌入问题可以使用RNN-LSTM对文中单词或句子进行encode，得到表示单词或句子的向量。对于图文匹配问题，Temage可以基于余弦计算等方法找到最为匹配的图片与文字，再使用基于统计的方法，对文章进行排版。

搜索推荐问题，Temage对比相似性的依据应当是文本中的关键词和推断出的风格主题。Temage使用关键词匹配来进行推荐。为提高搜索效率，Temage可以使用ElasticSearch开源搜索引擎框架对数据库建立索引。

## 系统亮点

### 功能性亮点

Temage将排版与深度学习技术结合。通过分析文章，向用户推荐最符合文章主题的版式。对输入的图片进行信息压缩、提取，与文字向量相匹配，将图片嵌入到文章中最合适的位置。再将用户选取的版式与此结果相结合，生成最终排版结果。

用户可以在排版完成后根据自己的实际需求，对排版结果进行在线的微调。

系统在运营过程中会不断地收集用户的反馈和评价，进而动态地更新模型，实现智能排版生成作品风格的时尚性和先进性，保持生成作品的高质量，为用户提供更具潮流性的智能排版服务。

用户社区功能使得用户可以将优秀的排版结果与他人分享，也为用户提供交流和反馈的平台，还可以根据用户处理的文章类型，为用户推荐社区中的优秀案例。

### 非功能性亮点

Temage服务系统分布式推断的理念是指，系统为用户提供服务所使用的计算资源不仅包括服务端的资源，同时包括客户端浏览器的计算资源，实现资源的合理分配和利用，为用户提供高效、流畅的服务。

在处理错误恢复方面，Temage使用Kubernetes框架，利用Deployment进行部署，能够在版本错误的情况下快速回滚到上一稳定版本，错误恢复时间极短。

# 开发计划

## 最终呈现形式

Temage的最终呈现形式为一个能流畅运行于浏览器的Web应用，运用深度学习技术为用户智能排版。完成图文排版和用户社区两大核心模块，能够有效地为用户提供排版服务。保证图片嵌入位置的准确性，保证文章主题分析的准确性，保证排版结果的优质和应用的鲁棒性。同时保证界面简洁、优美、友好，服务流畅、稳定、优质，具有跨平台、跨浏览器的特性。

## 主要功能描述

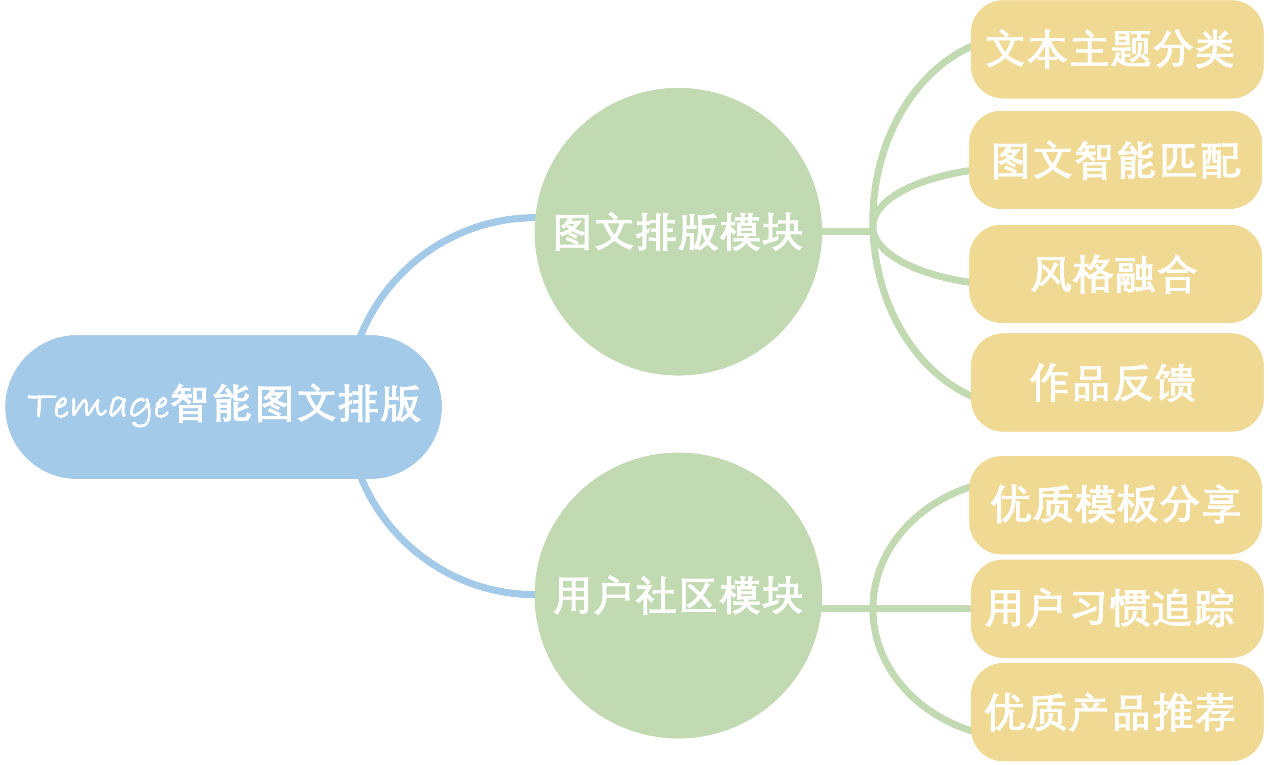


图 2.1 Temage两大功能模块

Temage功能分为两大模块——图文排版生成和用户社区模块。

图文排版模块分为文本主题分类和图文智能匹配两个子模块，文本主题分类影响作品美化和作品推荐。图文智能匹配控制文章图片的排版。

用户社区模块分为优质版式分享、用户反馈学习和优秀产品推荐三个子模块，Temage支持用户之间分享优质的作品，并能根据用户对产品的反馈迭代模型，使服务更加贴合用户，并可随时代潮流的变化而变化，Temage提供游廊，根据用户的习惯向用户提供推荐，同时，用户可以通过搜索查找到相关作品进行借鉴。

## 运行环境

主流浏览器，如Chrome、Firefox，Safari，Edge等。拥有Intel强大处理器的用户端。

## 验收标准

应用实现效果符合预期，能够得到一个大致符合用户预期的排版效果。

应用要具备上述提到的两大功能模块，具有完整性。

应用能够满足目标人群的需求，提供流畅、稳定的优质服务。

界面设计符合用户人群的审美，对用户友好。

## 关键问题

每个用户的写作领域相对固定，但各个用户之间的差距较大。如果使用一个统一的模型则会使不同题材、不同领域的文章排版趋向同质化，对排版的效果产生负面影响。如何对每个用户进行精准定位从而进行个性化排版是本项目的一个关键问题。Temage通过LSTM神经网络，基于用户的选择历史，结合本次的用户输入，推断最适合用户的排版方案。

新用户进入本平台，在无任何历史纪录的情况下，用户习惯追踪模型是无法正常工作的。通过在注册时让用户选择自己经常编辑的文章类型，对用户习惯进行初始化。

图文匹配问题是一个O(n\*m)的问题，且常数较大，需要用户等待的时间较长，Temage将尝试使用异步、并发的方法调用，以减少所需的时间。

更多细节详见技术文档。

## 捕获进度安排：

图 2.2 Temage智能图文排版项目计划甘特图

## 开发预算

表 2.1 开发预算

|  |  |  |
| --- | --- | --- |
| 项目 | 预算 | 备注 |
| 英特尔® AI DevCloud | ¥0.00 | 感谢英特尔主办方 |
| 持续集成/持续部署服务器 | ¥0.00 | 谷歌云平台提供的Gitlab共享服务器 |
| 测试服务器 | ¥3000.00 | 价格参考阿里云 |
| 前期调研费用 | ¥300.00 | 调研问卷推广费用 |
| sentry监控服务 | ¥0.00 | sentry官方免费服务 |
| 技术学习费用 | ¥300.00 |  |
| 合计 | ¥3600.00 |  |

# 可行性分析

## 市场可行性分析

### 与现行软件对比

现有的在线排版软件分两类，一类是在线的非自动排版应用，即给用户提供排版所需的功能，用户需要自己进行操作。比较典型的如用于公众号排版的“秀米”。适用于有美工基础的用户，不熟悉使用的用户需要一定的时间学习使用，具有一定的学习成本。

第二类是在线的自动排版应用，但只支持文字的在线排版，功能单一，包含修正标点、调整空行缩进、调整字号等功能，而且需要用户预先设置格式要求。

Temage则支持自动的图文排版，其优势主要有以下几点：

1. Temage致力于为用户提供专业且合适的版面设计，生成优质的图文排版，为用户节省90%以上的排版时间，降低版面设计的门槛没，减少繁琐枯燥的工作。
2. 在商业理念上，Temage保留用户对自己作品的知识产权。用户有公开或交易自己版权的权利，营造一个开放、包容、多样化的创意市场。
3. 并行推断的理念是用户在寻求服务的同时也应提供一定的计算资源。用户设备Intel处理器的强大算力，结合Tensorflow.js调用模型进行推断，后端使用Tensorflow、pyTorch调用模型进行推断，相互协作，分摊计算压力，减少运行所需时间，保证应用的流畅性和稳定性，提高用户的使用体验。
4. 系统在运营过程中会不断地收集用户的反馈和评价，进而动态地更新模型，实现智能排版生成作品风格的时尚性和先进性，保持生成作品的高质量，为用户提供更具潮流性的智能排版服务。

## 技术可行性分析

### 功能简述

Temage包含两大功能模块——图文排版模块和用户社区模块——都将用到深度学习的技术。

图文排版模块包含四个部分：文本主题分类、图文匹配、风格融合和作品反馈。

用户社区模块包含三个部分：优质版式分享、用户习惯追踪，优秀产品推荐。

### 技术要素

上述功能所需的技术主要有以下几点：

优质产品推荐，分为搜索推荐和用户习惯推荐，搜索推荐预先使用统计的方法找出文本中关键词，结合文章的分类让用户进行搜索。为提高搜索效率，Temage使用ElasticSearch开源搜索引擎框架对数据库建立索引。用户习惯推荐根据用户相关操作的时序分析，将用户习惯与文章风格进行比对，向用户进行推荐。

图片的识别，Temage借鉴VGG16的网络结构，设计基于CNN神经网络的模型。

文本的分类使用ELMo+LSTM+TextCNN模型。根据文本的分类，使用Top-k方法推荐合适的版式。

用户习惯追踪，使用基于LSTM神经网络的模型，对用户收藏或创作的产品进行时序分析。

图片文本的匹配，使用双分支模型，基于余弦计算等方法匹配图片在文字的位置，进行图片与文字的初排版。

更多细节详见技术文档。

## 资源可行性分析

人力资源包括队员四人。

模型训练数据集：THUCNews数据集（http://thuctc.thunlp.org/）。THUCNews是根据新浪新闻RSS订阅频道的历史数据筛选过滤生成，包含74万篇新闻文档，均为UTF-8纯文本格式，分布于14个分类中，用于文本分类进行风格推荐的模型训练。哔哩哔哩数据集，共约140万篇文章以及120万个图文匹配样例（剔除无效文章之后）。使用爬虫在哔哩哔哩专栏（https://www.bilibili.com/read/home）获取数据。用于用户习惯追踪模型和图文匹配模型的训练。

Temage需要一台训练专用服务器，由Intel® AI DevCloud 赞助提供。

Temage需要一个高可用集群，由校方提供。

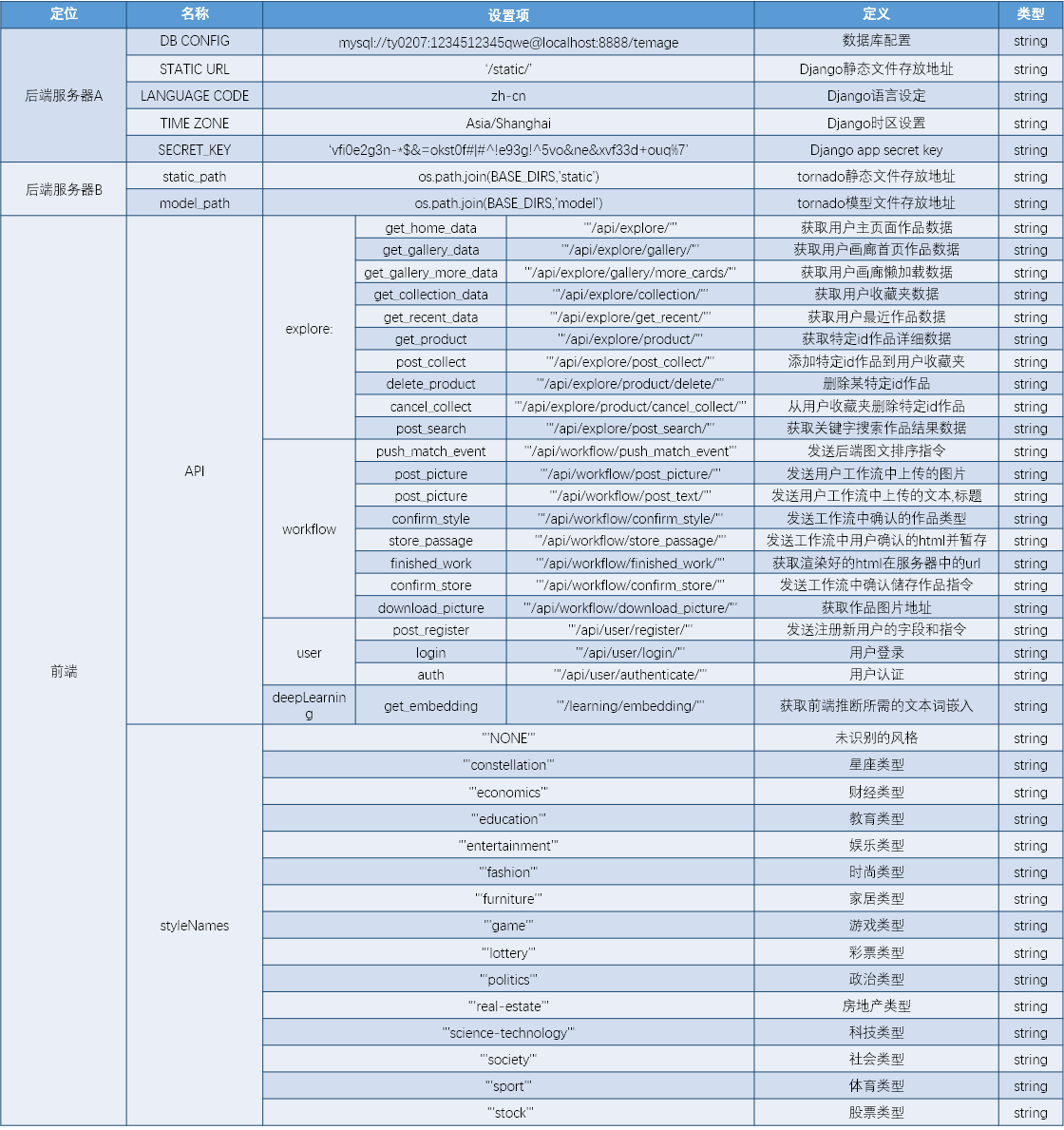
# 需求分析

## 数据描述

### 静态数据

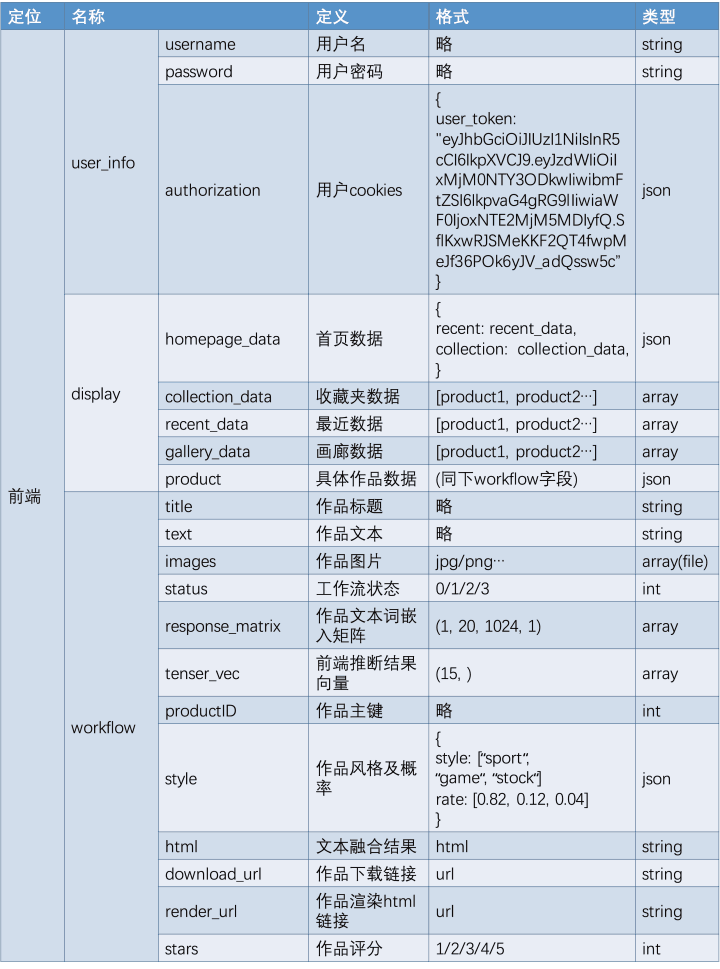
静态数据是指在运行过程中主要作为控制或参考用的数据,它们在很长的一段时间内不会变化，一般不随运行而变。

表 4.1 静态数据表



### 动态数据

表 4.2 动态数据表



### 数据词典

表 4.3 数据词典

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 描述 | 是否允许空 | 字段名称 |
| id | int(11) | user主键，用户的ID编号 | NOT NULL | id |
| username | varchar(150) | 用户名 | NOT NULL | username |
| password | varchar(128) | 密码 | NOT NULL | password |
| is\_superuser | tinyint(1) | 是否为超级用户 | NOT NULL | is\_superuser |
| is\_staff | tynyint(1) | 是否为员工 | NOT NULL | is\_staff |
| email | varchar(254) | 邮箱地址 | NOT NULL | email |
| first\_name | varchar(30) | 名 |  | first\_name |
| last\_name | varchar(250) | 姓 |  | last\_name |
| date\_joined | datetime(6) | 用户创建日期 | NOT NULL | date\_joined |
| last\_login | datetime(6) | 上次登录日期 | NOT NULL | last\_login |
| avator | varchar(100) | 用户头像的相对存储路径 | NOT NULL | avator |
|  |  |  |  |  |
| 作品数据表 | | | | 作品数据表 |
| 字段名称 | 数据类型 | 描述 | 是否允许空 | 字段名称 |
| id | int(11) | 主键，作品的id编号 | NOT NULL | id |
| title | varchar(255) | 作品标题 | NOT NULL | title |
| image\_src | varchar(100) | 作品长图的相对路径 | NOT NULL | image\_src |
| vector | longtext | 模型处理作品产生的向量 | NOT NULL | vector |
| html | longtext | 生成结果的html字符串 |  | html |
| html\_file | varchar(100) | 生成的html文件的相对路径 | NOT NULL | html\_file |
| pdf\_file | varchar(100) | 根据html生成的pdf文件的相对路径 |  | pdf\_file |
| width | int(11) | 长图设置的宽度 | NOT NULL | width |
| score | double | 用户评分 |  | score |
| is\_finished | tinyint(1) | 是否完成并发布 | NOT NULL | is\_finished |
| create\_time | datetime(6) | 创建时间 | NOT NULL | create\_time |
| update\_time | datetime(6) | 修改时间 | NOT NULL | update\_time |
| creator\_id | int(11) | 外键，创建者的id | NOT NULL | creator\_id |
| style\_id | int(11) | 外键，使用样式的id | NOT NULL | style\_id |
|  |  |  |  |  |
| 主题数据表 | | | | 主题数据表 |
| id | int(11) | 主键，主题的id编号 | NOT NULL | id |
| name | varchar(50) | 主题的名称 | NOT NULL | name |
|  |  |  |  |  |
| 风格数据表 | | | | 风格数据表 |
| id | int(11) | 主键，风格的id编号 | NOT NULL | id |
| name | varchar(50) | 样式名称 | NOT NULL | name |
| css | varchar(100) | 样式表的相对路径 | NOT NULL | css |
|  |  |  |  |  |
| 卡片数据表 | | | | 卡片数据表 |
| id | int(11) | 主键，卡片的id编号 | NOT NULL | id |
| title | varchar(50) | 卡片标题 | NOT NULL | title |
| head | varchar(255) | 卡片描述字段1 |  | head |
| prompt | longtext | 卡片描述字段2 |  | prompt |
| foot\_text | varchar(255) | 卡片描述字段3 |  | foot\_text |
| create\_time | datetime(6) | 创建时间 | NOT NULL | create\_time |
| update\_time | datetime(6) | 更新时间 | NOT NULL | update\_time |
| creator\_id | int(11) | 外键，创建者id | NOT NULL | creator\_id |
| prduct\_id | int(11) | 外键，对应作品id | NOT NULL | prduct\_id |
|  |  |  |  |  |
| 收藏夹数据表 | | | | 收藏夹数据表 |
| id | int(11) | 主键，收藏夹的id编号 | NOT NULL | id |
| name | varchar(50) | 收藏夹名称 |  | name |
| prompt | longtext | 收藏夹描述字段 | NOT NULL | prompt |
| url | varchar(255) | 收藏夹链接 | NOT NULL | url |
| create\_time | datetime(6) | 创建时间 | NOT NULL | create\_time |
| user\_id | int(11) | 外键，对应用户id | NOT NULL | user\_id |
|  |  |  |  |  |
| workflow缓存文件数据表 | | | | workflow缓存文件数据表 |
| id | int(11) | cache实例的对象 | NOT NULL | id |
| imgs\_urls | longtext | 用户提交的文章 |  | imgs\_urls |
| text | longtext | 用户提交的图片的相对路径的list |  | text |
| match\_list | longtext | 图文匹配模型返回的结果 |  | match\_list |
| user\_id | int(11) | 对应的用户的id | NOT NULL | user\_id |
| 字段名称 | 数据类型 | 描述 | 是否允许空 | 字段名称 |
| id | int(11) | user主键，用户的ID编号 | NOT NULL | id |
| username | varchar(150) | 用户名 | NOT NULL | username |
| password | varchar(128) | 密码 | NOT NULL | password |
| is\_superuser | tinyint(1) | 是否为超级用户 | NOT NULL | is\_superuser |
| is\_staff | tynyint(1) | 是否为员工 | NOT NULL | is\_staff |
| email | varchar(254) | 邮箱地址 | NOT NULL | email |
| first\_name | varchar(30) | 名 |  | first\_name |
| last\_name | varchar(250) | 姓 |  | last\_name |
| date\_joined | datetime(6) | 用户创建日期 | NOT NULL | date\_joined |
| last\_login | datetime(6) | 上次登录日期 | NOT NULL | last\_login |
| avator | varchar(100) | 用户头像的相对存储路径 | NOT NULL | avator |
|  |  |  |  |  |
| 作品数据表 | | | | 作品数据表 |
| 字段名称 | 数据类型 | 描述 | 是否允许空 | 字段名称 |
| id | int(11) | 主键，作品的id编号 | NOT NULL | id |
| title | varchar(255) | 作品标题 | NOT NULL | title |
| image\_src | varchar(100) | 作品长图的相对路径 | NOT NULL | image\_src |
| vector | longtext | 模型处理作品产生的向量 | NOT NULL | vector |
| html | longtext | 生成结果的html字符串 |  | html |
| html\_file | varchar(100) | 生成的html文件的相对路径 | NOT NULL | html\_file |
| pdf\_file | varchar(100) | 根据html生成的pdf文件的相对路径 |  | pdf\_file |
| width | int(11) | 长图设置的宽度 | NOT NULL | width |
| score | double | 用户评分 |  | score |
| is\_finished | tinyint(1) | 是否完成并发布 | NOT NULL | is\_finished |
| create\_time | datetime(6) | 创建时间 | NOT NULL | create\_time |
| update\_time | datetime(6) | 修改时间 | NOT NULL | update\_time |
| creator\_id | int(11) | 外键，创建者的id | NOT NULL | creator\_id |
| style\_id | int(11) | 外键，使用样式的id | NOT NULL | style\_id |
|  |  |  |  |  |
| 主题数据表 | | | | 主题数据表 |
| id | int(11) | 主键，主题的id编号 | NOT NULL | id |
| name | varchar(50) | 主题的名称 | NOT NULL | name |
|  |  |  |  |  |
| 风格数据表 | | | | 风格数据表 |
| id | int(11) | 主键，风格的id编号 | NOT NULL | id |
| name | varchar(50) | 样式名称 | NOT NULL | name |
| css | varchar(100) | 样式表的相对路径 | NOT NULL | css |
|  |  |  |  |  |
| 卡片数据表 | | | | 卡片数据表 |
| id | int(11) | 主键，卡片的id编号 | NOT NULL | id |
| title | varchar(50) | 卡片标题 | NOT NULL | title |
| head | varchar(255) | 卡片描述字段1 |  | head |
| prompt | longtext | 卡片描述字段2 |  | prompt |
| foot\_text | varchar(255) | 卡片描述字段3 |  | foot\_text |
| create\_time | datetime(6) | 创建时间 | NOT NULL | create\_time |
| update\_time | datetime(6) | 更新时间 | NOT NULL | update\_time |
| creator\_id | int(11) | 外键，创建者id | NOT NULL | creator\_id |
| prduct\_id | int(11) | 外键，对应作品id | NOT NULL | prduct\_id |
|  |  |  |  |  |
| 收藏夹数据表 | | | | 收藏夹数据表 |
| id | int(11) | 主键，收藏夹的id编号 | NOT NULL | id |
| name | varchar(50) | 收藏夹名称 |  | name |
| prompt | longtext | 收藏夹描述字段 | NOT NULL | prompt |
| url | varchar(255) | 收藏夹链接 | NOT NULL | url |
| create\_time | datetime(6) | 创建时间 | NOT NULL | create\_time |
| user\_id | int(11) | 外键，对应用户id | NOT NULL | user\_id |
|  |  |  |  |  |

### 数据采集

1. 哔哩哔哩数据集，共约140万篇文章以及120万个图文匹配样例（剔除无效文章之后）。凭借此Temage将得到新闻中的图片和图片上下文，上述数据可用于图文匹配和用户习惯追踪模型的训练。
2. THUCNews数据集。THUCNews（http://thuctc.thunlp.org/）是根据新浪新闻RSS订阅频道的历史数据筛选过滤生成，包含74万篇新闻文档（2.19 GB），均为UTF-8纯文本格式，分布于14个分类中，用于文本分类进行风格推荐的模型训练。
3. 用户上传数据，用户在使用本产品中将上传相应的文本和图片，在最终产出产品之后该数据可用于继续进行模型的更新。

## 功能需求

### 功能模块结构图

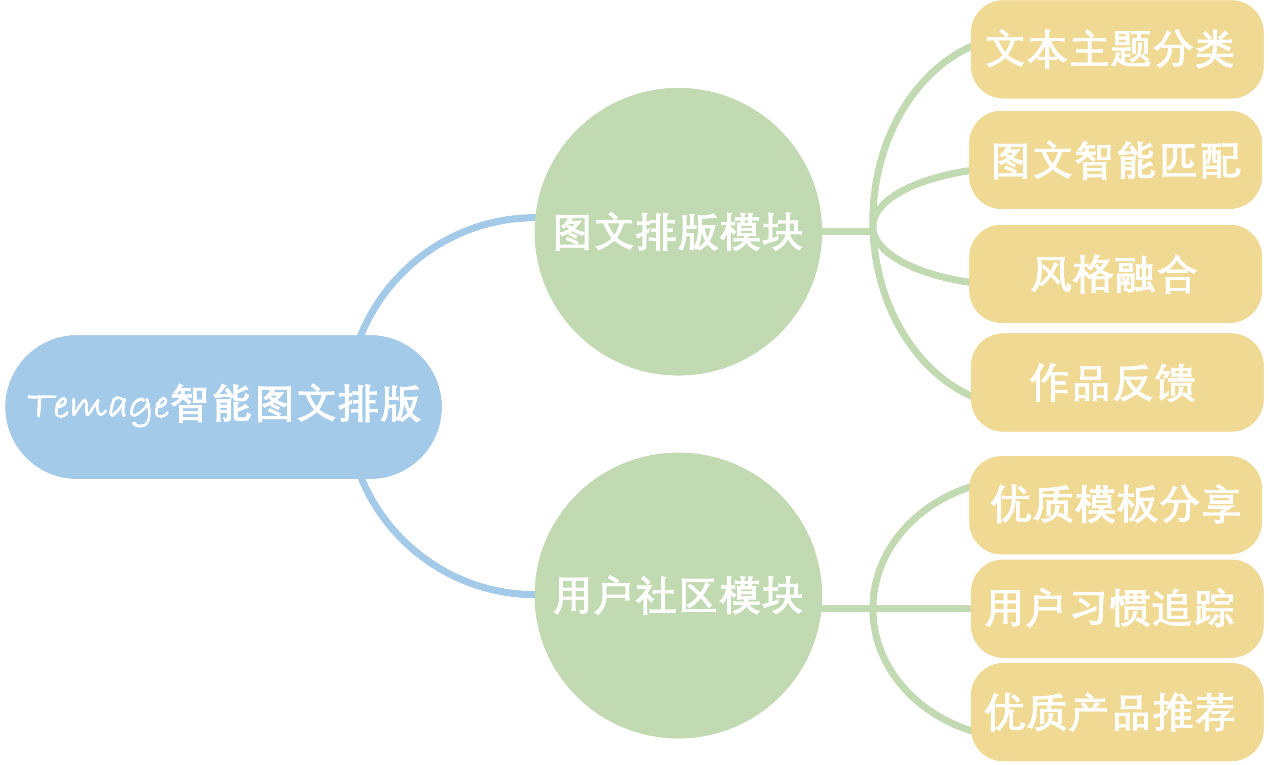


图 4.1 Temage的两大功能模块

### 核心功能模块描述

表 4.4 核心功能模块描述

|  |  |  |  |
| --- | --- | --- | --- |
| **功能模块** | **功能** | **功能描述** | **优先级** |
| **图文排版模块** | 文本主题分类 | 使用TextCNN模型对用户上传文本进行分析，对文本进行分类，为作品提供关键字，影响后期美化 | 10 |
| 图文匹配 | 使用双分支模型进行图文匹配，形成初排版 | 10 |
| 风格融合 | 将用户选择的风格融合到作品中，进行美化 | 10 |
| 作品反馈 | 记录用户最终选择的风格于用户习惯中，进一步更新用户习惯，数据可进一步更新文本分类模型 | 9 |
| **用户社区模块** | 优秀产品推荐 | 以LSTM为基础建立模型，对用户的习惯进行追踪，推断的风格选择 | 10 |
| 用户搜索推荐 | 用户使用关键词搜索提供相应作品的显示，供用户参考 | 8 |
| 优质版式分享 | 用户可以对平台中优质的作品进行二次创作 | 7 |

## 非功能性需求

### 扩展性和可维护性

1. 功能扩展一，可以与各大社交媒体平台进行对接，例如微信，微博，有利于自媒体的进一步发展。
2. 功能扩展二，进一步完善社区功能，在保障用户版权的条件下，鼓励用户之间进行交流和借鉴，提升图文编辑的体验感。
3. 可维护性：使用MVVM可维护性强。
4. 与深度学习相关的大计算量组件（ServerA）与维护社区和日常用户操作组件（ServerB）相互独立，可分别进行扩展与维护。
5. Temage能够支持多款浏览器，提高用户的使用舒适程度。

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 42+ | 73+ | 66+ | 12+ |

### 易用性

Temage始终站在用户的角度进行思考。

1. 用户在进行操作时具有详尽的操作导航。
2. 页面的text editor功能齐全，可以让用户拥有一站式编辑体验。
3. 对于用户编辑结果进行实时保存，防止用户网络不畅通或不经意间关闭页面带来的不必要损失。

## 性能需求

### 时间特性

Temage主要耗时模块为文本预处理模块和图像预处理模块，为分担服务器压力，加快运行速度，提升用户体验，Temage将部分文本预处理工作交给前端（浏览器）进行处理。Temage将文本分类TextCNN模型放在前端进行执行，并在用户进行风格选择的同时，ServerA继续进行图像预处理的相关工作。

### 适应性

1. Temage作为web应用在浏览器中运行，由于需要在前端进行深度学习的推断，用户只需使用有相应能力的浏览器（例如Chrome Safari Firefox Edge）均可以使用Temage进行工作。
2. Temage在进行维护升级后，如若对操作方式产生影响，均会在新版本中给予用户显著的提示，确保用户的流畅使用。

## 运行需求

### 用户界面

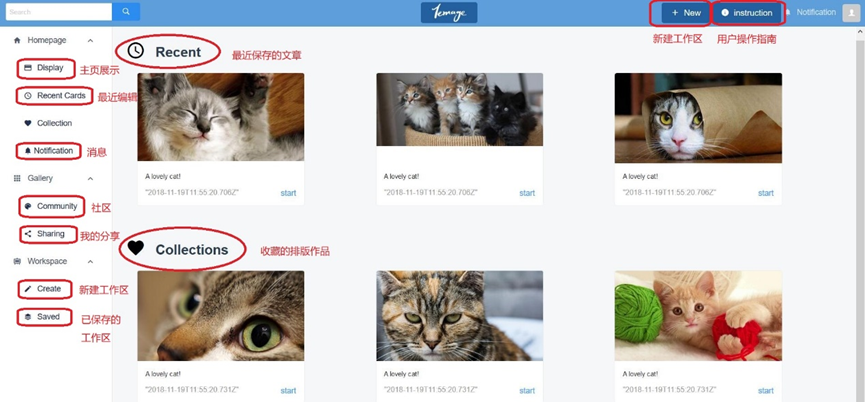


图 4.2 homepage页面



图 4.3 用户输入界面

图 4.4 推荐风格选取界面



图 4.5 用户微调界面



图 4.6 成果预览界面



图 4.7 用户评分反馈

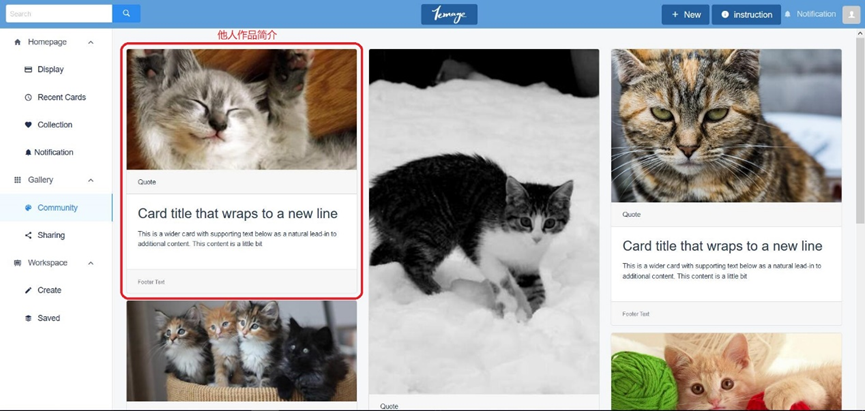


图 4.8 社区界面

### 硬件接口

无。

### 软件接口

无外部依赖软件接口，Temage相关接口见下文。

### 故障处理

Temage使用Docker + Kubernetes + Istio实现Microservice架构，能够性能降级、恢复再启动，仍将有部分功能可以继续使用，例如，深度学习的相关的serverA发生故障，不能进行深度学习的推断，serverB正常服务，用户仍旧可以登录进行游廊的游览。服务端端故障将生成详细的log日志于服务器中，并且会有友好的用户提示。

## 用例图

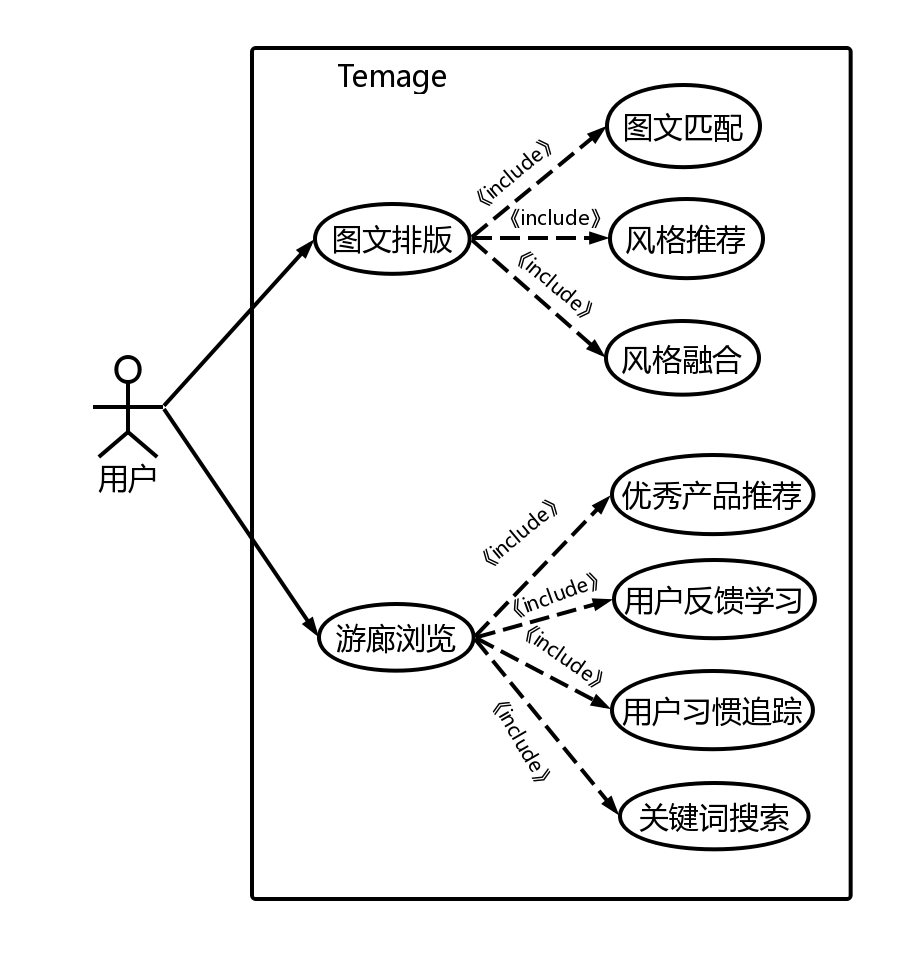


图 4.9 用例图

## 核心模块用例规约

表 4.5 风格推荐用例规约

|  |  |
| --- | --- |
| 用例名称 | 风格推荐 |
| 功能简述 | 讲用户上传的文本分为十六类 |
| 用例编号 | 001 |
| 执行者 | Front-End |
| 前置条件 | 1. 用户上传文本为中文文本  2. 用户上传文本逻辑清晰，行文流畅  3. 用户登录 |
| 后置条件 | 内存储存文本分类和风格推荐的结果 |
| 涉众利益 | 用户：希望得到合适的风格推荐  风格融合：服务器希望得到合适的分类选取合适的美化方式 |
| 基本路径 | 1. 用户上传文字  2. 服务器对文本进行句子嵌入后将结果和用户习惯发送至前端  3. 前端TextCNN模型进行推断得到推荐风格  4. 向用户推荐风格  5. 用户最终确定风格，结果返回后端 |
| 扩展路径 | 1. 文本内容为空，将提醒用户  2. 文本内容中文内容过少或不存在，提示用户 |
| 字段列表 | 文本合理，文本类别 |
| 设计规则 | 单独模块 |
| 未解决的问题 | 大量非中文文本将降低分类的准确度 |
| 备注 | 无 |

表 4.6图文匹配用例规约

|  |  |
| --- | --- |
| 用例名称 | 图文匹配 |
| 功能简述 | 用户上传图像进行向量嵌入 |
| 用例编号 | 002 |
| 执行者 | ServerB |
| 前置条件 | 1. 用户上传图片大小限制  2. 用户上传图片与文本有关联  3. 用户登录 |
| 后置条件 | 无 |
| 涉众利益 | 用户：获取优质图文匹配 |
| 基本路径 | 1. 用户上传图片和文字  2. ServerB对图片和文本进行预处理  2. ServerB使用双分支模型进行模型匹配  3. ServerA向ServerB发送结果 |
| 扩展路径 | 1. 图片内容为空，将提醒用户  2. 图片图像分辨率过大或过小都将警告用户  3. 图片文本匹配率太低，将向用户进行提示 |
| 字段列表 | 图像编号-文本编号匹配率列表 |
| 设计规则 | 单独模块 |
| 未解决的问题 | 该请求的时间复杂度为O(m\*n)，将消耗较长时间 |
| 备注 | 后期将使用多线程提升性能，降低时间复杂度 |

表 4.7 风格融合用例规约

|  |  |
| --- | --- |
| 用例名称 | 风格融合 |
| 功能简述 | 将用户初排版与用户风格融合，完成对用户素材的排版 |
| 用例编号 | 003 |
| 执行者 | ServerA |
| 前置条件 | 1. 用户已上传符合要求的图片与文字  2. 用户已选取作品风格  3. 用户登录 |
| 后置条件 | 无 |
| 涉众利益 | 用户：获取最终产品。  用户反馈学习模块：获取反馈实例，进入用户反馈阶段。 |
| 基本路径 | 1. 用户对初排版进行调整，确认初排版  2. ServerA将图文匹配结果和风格对应样式融合，发送至前端 |
| 扩展路径 | 无 |
| 字段列表 | Html文本，状态码 |
| 设计规则 | 单独模块 |
| 未解决的问题 | 无 |
| 备注 | 无 |

表 4.8用户反馈学习例规约

|  |  |
| --- | --- |
| 用例名称 | 用户反馈学习 |
| 功能简述 | 用户对产品进行反馈，将反馈结果记录进入数据库 |
| 用例编号 | 004 |
| 执行者 | ServerA |
| 前置条件 | 存在已生成的作品，且不存在反馈  用户登录 |
| 后置条件 | 无 |
| 涉众利益 | Temage：数据可用于改进现有模型。 |
| 基本路径 | 1. 用户对最终模型给出反馈  2. ServerA将用户反馈存入数据库，以待优化模型 |
| 扩展路径 | 无 |
| 字段列表 | 状态码 |
| 设计规则 | 单独模块 |
| 未解决的问题 | 无 |
| 备注 | 为减少用户的响应时间，该操作可能为异步操作 |

表 4.9 关键词搜索用例规约

|  |  |
| --- | --- |
| 用例名称 | 关键词搜索 |
| 功能简述 | 用户进行搜索，寻找相关的已有作品 |
| 用例编号 | 005 |
| 执行者 | ServerA |
| 前置条件 | 1.存在已生成的作品 |
| 后置条件 | 无 |
| 涉众利益 | 用户：获取相关作品，用于借鉴。 |
| 基本路径 | 1. 用户进行搜索  2. ServerA使用ElasticSearch框架搜索，得到相关数据返回前端 |
| 扩展路径 | 无 |
| 字段列表 | 作品数据 |
| 设计规则 | 单独模块 |
| 未解决的问题 | 无 |
| 备注 | 无 |

表 4.10 优秀产品推荐用例规约

|  |  |
| --- | --- |
| 用例名称 | 优秀产品推荐 |
| 功能简述 | 用户在游廊浏览，根据用户收藏和产品向用户进行推荐 |
| 用例编号 | 006 |
| 执行者 | ServerA |
| 前置条件 | 1.用户登录 |
| 后置条件 | 无 |
| 涉众利益 | 用户：用户可看到已存在的作品。 |
| 基本路径 | 用户浏览游廊看到作品 |
| 扩展路径 | 服务器出错或无权限观看（403） |
| 字段列表 | 作品列表，状态码 |
| 设计规则 | 单独模块 |
| 未解决的问题 | 无 |
| 备注 | 无 |

表 4.11用户习惯追踪例规约

|  |  |
| --- | --- |
| 用例名称 | 用户习惯追踪 |
| 功能简述 | 针对用户相关操作预测用户习惯，将结果记录进入数据库 |
| 用例编号 | 004 |
| 执行者 | ServerA、ServerB |
| 前置条件 | 1. 用户登录  2. 用户完成作品反馈 |
| 后置条件 | 无 |
| 涉众利益 | Temage：数据可用于改进现有模型。 |
| 基本路径 | 1. 用户对最终模型给出反馈  2. ServerA将用户反馈存入数据库，以待优化模型 |
| 扩展路径 | 无 |
| 字段列表 | 状态码 |
| 设计规则 | 单独模块 |
| 未解决的问题 | 无 |
| 备注 | 为减少用户的响应时间，该操作可能为异步操作 |

## 其他需求

1. 为保证用户的文字图片的版权，用户可以选择在生成的长图上使用本平台的水印，以保障自己的权益。

2. 为用户推荐高效的Intel处理器，助力客户端推断，保障用户使用体验。

# 概要设计

## 处理流程

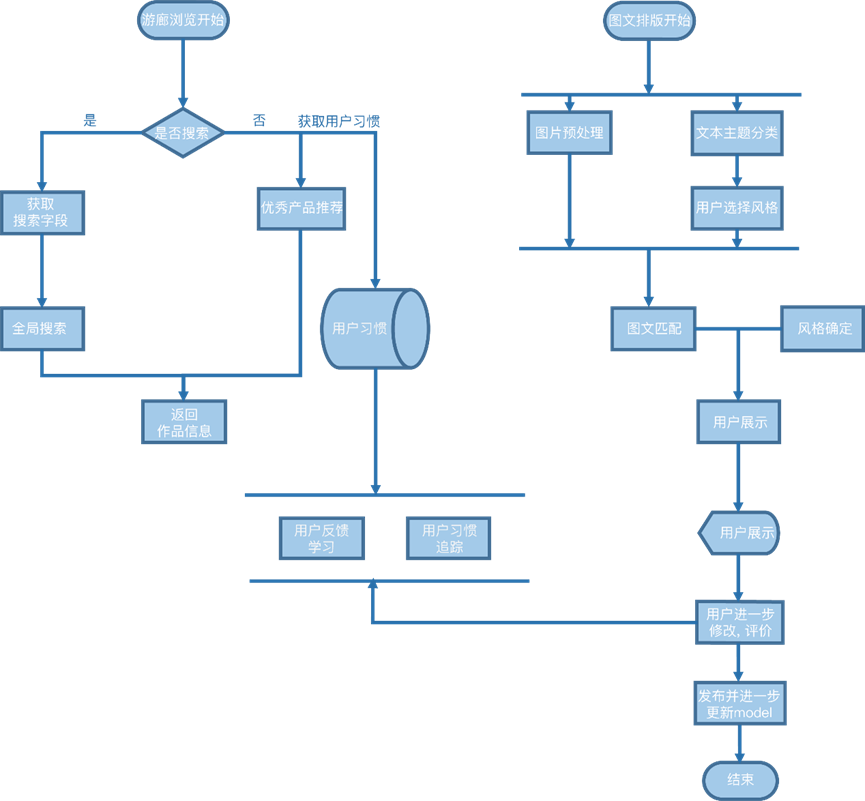


图 5.1 程序流程图

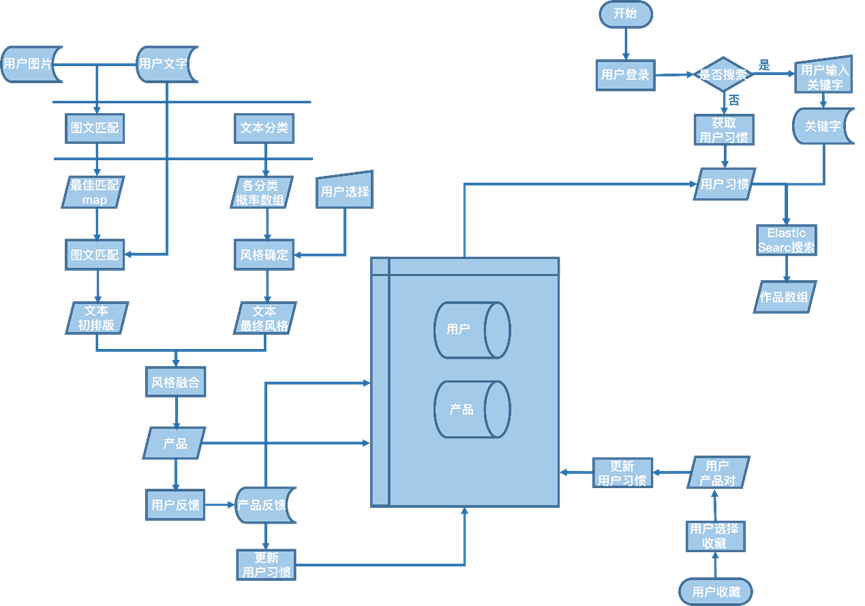


图 5.2 数据流程图

## 总体结构和模块设计

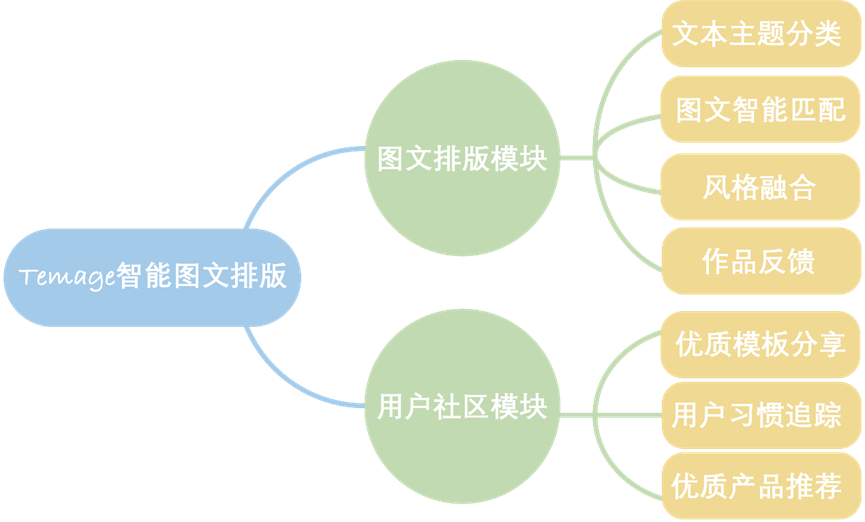


图 5.3 模块架构图

Temage功能分为两大模块——图文排版生成和用户社区模块。

图文排版模块分为文本主题分类、图文智能匹配、风格融合和作品反馈四个子模块，文本主题分类影响作品美化和作品推荐。图文智能匹配控制文章图片的排序，风格融合获取最终产品，作品反馈助力模型成长。

用户社区模块分为优质版式分享、用户习惯追踪和优秀产品推荐三个子模块，Temage支持用户之间分享优质的作品，并能根据用户对产品的反馈迭代模型，使服务更加贴合用户，并可随时代潮流的变化而变化，Temage提供游廊，根据用户的习惯向用户提供推荐，同时，用户可以通过搜索查找到相关作品进行借鉴。

## 功能分配

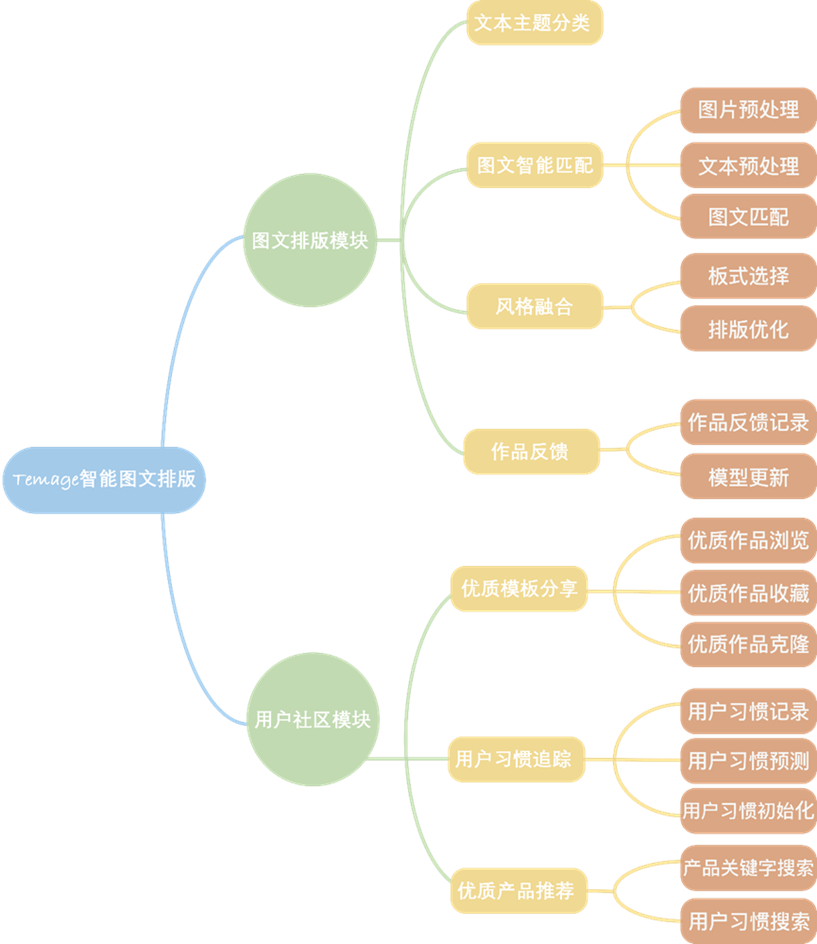


图 5.4 功能结构图

Temage分为两大功能模块——图文排版模块和用户社区模块。

图文排版模块分为文本主题分类、图文智能匹配、风格融合和作品反馈四个子模块。文本主题分类通过用户上传文本，为用户推荐文本分类；图文智能匹配将会进行文本预处理，图片预处理，之后通过双分支模型进行图文匹配；作品反馈通过记录用户最终对作品的反馈，用于各模型的更新。

用户社区模块分为优质版式分享、用户习惯追踪和优秀产品推荐三个子模块。用户浏览优质作品，可对作品进行克隆和收藏，实现优质版式分享；Temage要求用户注册是进行习惯初始化，在用户使用过程中，通过记录用户习惯，并对用户习惯进行预测，实现用户习惯追踪。Temage支持用户通过产品关键词搜索和用户习惯搜索两种方式向用户进行产品推荐。

## 接口设计

### 外部接口

前端与ServerA之间的接口

表 5.1 接口get\_homepage\_data

|  |  |
| --- | --- |
| 接口名 | get\_homepage\_data |
| 接口功能描述 | 针对用户对给出homepage的相关数据 |
| 参数描述 | {HEADERS:{ AUTHORIZATION: String:jwt\_code}} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {  'recent\_pics':[  {  'title':'Jenkins Added.',  'imgsrc':picurl1,  'prompt':'A lovely cat Lily!',  'id':0000000 // click at  }  ],  'collect\_pics':[  {  'title':'cat1\_lily',  'imgsrc':picurl2,  'prompt':'A lovely cat Lily!',  'id':0000000  }  ],  'gallery\_pics':[  {} // same above  ],  'user\_info': {  'username':,  'id': 1,  'avator':picurl2,  }  }; |

表 5.2 接口post\_register

|  |  |
| --- | --- |
| 接口名 | post\_register |
| 接口功能描述 | 注册用户 |
| 参数描述 | {  body:  {  'username':,String  'password': String,  'email': String,  'interest':Array,  'desc':  }  } |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {  'username': 'tmg',  'password': '123',  'email': '123123@qq.com',  'interest': ['Porn','Sport'], // not sure  'desc': 'love and peace'  } |

表 5.3 接口post\_login\_submit

|  |  |
| --- | --- |
| 接口名 | post\_login\_submit |
| 接口功能描述 | 用户登录返回jwt\_code |
| 参数描述 | {  body:  {  'username': 'user1',  'password': '123'  }  } |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {"token": 'anyway'} |

表 5.4 接口post\_jwt\_authenticate

|  |  |
| --- | --- |
| 接口名 | post\_jwt\_authenticate |
| 接口功能描述 | 前端使用jwt\_code换取用户信息 |
| 参数描述 | {HEARDERS:jwt\_code} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {'username':'user1',  'id': 1,  'avator':’https://www.baidu.com/s.png’  } |

表 5.5 接口get\_gallery\_data

|  |  |
| --- | --- |
| 接口名 | get\_gallery\_data |
| 接口功能描述 | 前端获取游廊信息 |
| 参数描述 | {HEARDERS: ‘AUTHORIZATION’:jwt\_code} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | [  {  'title':'SumpleCat',  'imagesrc':picurl3,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  },  {  'title':'SumpleCat',  'imagesrc':picurl2,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  },  {  'title':'SumpleCat',  'imagesrc':picurl1,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  },  {  'title':'SumpleCat',  'imagesrc':picurl1,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  },  {  'title':'SumpleCat',  'imagesrc':picurl3,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  }  ] |

表 5.6 接口get\_gallery\_more\_cards

|  |  |
| --- | --- |
| 接口名 | get\_gallery\_more\_cards |
| 接口功能描述 | 前端获取游廊信息 |
| 参数描述 | {HEARDERS: ‘AUTHORIZATION’:jwt\_code} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | [  {  'title':'SumpleCat',  'imagesrc':picurl3,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  },  {  'title':'SumpleCat',  'imagesrc':picurl2,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  },  {  'title':'SumpleCat',  'imagesrc':picurl1,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  },  {  'title':'SumpleCat',  'imagesrc':picurl1,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  },  {  'title':'SumpleCat',  'imagesrc':picurl2,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  },  {  'title':'SumpleCat',  'imagesrc':picurl3,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  },  {  'title':'SumpleCat',  'imagesrc':picurl3,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  }  ] |

表 5.7 接口post\_search

|  |  |
| --- | --- |
| 接口名 | post\_search |
| 接口功能描述 | 前端获取游廊信息 |
| 参数描述 | {body: {‘keys’: ‘Sports Tech’}} |
| 接口间的依赖关系 | 该接口依赖于 ES RestFul API /temage/product/\_search |
| 返回参数样例 | [  {  'title':'SumpleCat',  'imagesrc':picurl3,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  },  {  'title':'SumpleCat',  'imagesrc':picurl3,  'head': 'head content',  'maintext': 'What a lovely cat! Its name is Lily!',  'foottest': 'foot content',  'id':0000000,  'creator':user\_info,  }  ] |

表 5.8 接口get\_collection\_data

|  |  |
| --- | --- |
| 接口名 | get\_collection\_data |
| 接口功能描述 | 获取用户收藏夹信息 |
| 参数描述 | {HEARDERS: ‘AUTHORIZATION’:jwt\_code} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | [  {  'name':'cat1\_lily',  'imgsrc':picurl3,  'prompt':'A lovely cat Lily!',  'id':0000000,  'creator':user\_info,  'title': 'cat1\_lily'  },  {  'name':'cat1\_lily',  'imgsrc':picurl2,  'prompt':'A lovely cat Lily!',  'id':0000000,  'creator':user\_info,  'title': 'cat1\_lily'  },  {  'name':'cat1\_lily',  'imgsrc':picurl1,  'prompt':'A lovely cat Lily!',  'id':0000000,  'creator':user\_info,  'title': 'cat1\_lily'  },  {  'name':'cat1\_lily',  'imgsrc':picurl2,  'prompt':'A lovely cat Lily!',  'id':0000000,  'creator':user\_info,  'title': 'cat1\_lily'  },  ]; |

表 5.9 接口get\_recent\_data

|  |  |
| --- | --- |
| 接口名 | get\_recent\_data |
| 接口功能描述 | 获取用户近期收藏/创建的作品 |
| 参数描述 | {HEARDERS: ‘AUTHORIZATION’:jwt\_code} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | [  {  'name':'cat1\_lily',  'imgsrc':picurl3,  'prompt':'A lovely cat Lily!',  'id':0000000,  'creator':user\_info,  'title': 'cat1\_lily'  },  {  'name':'cat1\_lily',  'imgsrc':picurl2,  'prompt':'A lovely cat Lily!',  'id':0000000,  'creator':user\_info,  'title': 'cat1\_lily'  },  {  'name':'cat1\_lily',  'imgsrc':picurl1,  'prompt':'A lovely cat Lily!',  'id':0000000,  'creator':user\_info,  'title': 'cat1\_lily'  },  {  'name':'cat1\_lily',  'imgsrc':picurl2,  'prompt':'A lovely cat Lily!',  'id':0000000,  'creator':user\_info,  'title': 'cat1\_lily'  },  ]; |

表 5.10 接口get\_product

|  |  |
| --- | --- |
| 接口名 | get\_product |
| 接口功能描述 | 通过product\_id获取到作品具体信息 |
| 参数描述 | {HEARDERS: {‘AUTHORIZATION’:jwt\_code }, body:{‘productID’: 1} } |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {  id: card\_id, // this id is textID, may differ from card\_id  text: result\_html,  creator: user\_info,  title: '梅西重返赛场',  style: ['Sports', 'Celebrity'],  canBeDelete: true,  hasBeenCollected: false  } |

表 5.11 接口post\_collect

|  |  |
| --- | --- |
| 接口名 | post\_collect |
| 接口功能描述 | 收藏作品，更新用户 |
| 参数描述 | {HEARDERS: {‘AUTHORIZATION’:jwt\_code }, body:{‘productID’: 1} } |
| 接口间的依赖关系 | 依赖于ServerB predict\_history |
| 返回参数样例 | succeed |

表 5.12 接口delete\_product

|  |  |
| --- | --- |
| 接口名 | delete\_product |
| 接口功能描述 | 删除作品 |
| 参数描述 | {HEARDERS: {‘AUTHORIZATION’:jwt\_code }, body:{‘productID’: 1} } |
| 接口间的依赖关系 | ES 模块/temage/product/\_delete\_by\_query/ |
| 返回参数样例 | succeed |

表 5.13 接口cancel\_collect

|  |  |
| --- | --- |
| 接口名 | cancel\_collect |
| 接口功能描述 | 取消用户对某作品的收藏 |
| 参数描述 | {HEARDERS: {‘AUTHORIZATION’:jwt\_code }, body:{‘productID’: 1} } |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | succeed |

表 5.14 接口post\_text

|  |  |
| --- | --- |
| 接口名 | post\_text |
| 接口功能描述 | 接收用户传来的文本 |
| 参数描述 | {HEARDERS: {‘AUTHORIZATION’:jwt\_code }, body:{‘title’:’title’, ‘text’:’text’ }} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | succeed |

表 5.15 接口post\_picture

|  |  |
| --- | --- |
| 接口名 | post\_picture |
| 接口功能描述 | 接收用户传来的图片 |
| 参数描述 | {HEARDERS: {‘AUTHORIZATION’:jwt\_code }, ‘form-data’:{files:{file1,file2}}} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | succeed |

表 5.16 接口push\_match\_event

|  |  |
| --- | --- |
| 接口名 | push\_match\_event |
| 接口功能描述 | 对用户产品进行图文匹配 |
| 参数描述 | {HEARDERS: {‘AUTHORIZATION’:jwt\_code },’body’:{‘productID’:1}} |
| 接口间的依赖关系 | ServerB /image\_match |
| 返回参数样例 | succed |

表 5.17 接口post\_confirmed\_style

|  |  |
| --- | --- |
| 接口名 | post\_confirmed\_style |
| 接口功能描述 | 返回对应产品的html |
| 参数描述 | {HEARDERS: {‘AUTHORIZATION’:jwt\_code },’body’:{‘productID’:1}} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {‘html’:’<h1 class=”class-1”>Hello，World!</h1>’} |

表 5.18 接口store\_passage

|  |  |
| --- | --- |
| 接口名 | store\_passage |
| 接口功能描述 | 返回对应产品的html初稿 |
| 参数描述 | {HEARDERS: {‘AUTHORIZATION’:jwt\_code},’body’:{‘styles:[‘Sport’],‘res\_html’:’sss’, ‘title’:’title’, ‘t\_width’:200 }} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {‘html’:’<h1 class=”class-1”>Hello，World!</h1>’} |

表 5.19 接口finished\_work

|  |  |
| --- | --- |
| 接口名 | finished\_work |
| 接口功能描述 | 返回产品的html最终稿 |
| 参数描述 | {HEARDERS: {‘AUTHORIZATION’:jwt\_code},’body’:{‘styles:[‘Sport’],‘res\_html’:’sss’, ‘title’:’title’, ‘t\_width’:200 }} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {‘html’:’<h1 class=”class-1”>Hello，World!</h1>’} |

表 5.20 接口download

|  |  |
| --- | --- |
| 接口名 | download |
| 接口功能描述 | 下载相应作品的pdf |
| 参数描述 | {HEARDERS:{‘AUTHORIZATION’:jwt\_code},’body’: {‘productID’:1} } |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {file: File} |

表 5.21 接口confirm\_store

|  |  |
| --- | --- |
| 接口名 | confirm\_store |
| 接口功能描述 | 将作品进行发布 |
| 参数描述 | {HEARDERS:{‘AUTHORIZATION’:jwt\_code},’body’: {‘productID’:1} } |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {code:200} |

表 5.22 接口embedding

|  |  |
| --- | --- |
| 接口名 | embedding |
| 接口功能描述 | 返回对应产品的html |
| 参数描述 | {HEARDERS:{‘AUTHORIZATION’:jwt\_code},’body’: {‘productID’:1} } |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {Array : shape(1, 20, 1024, 1) } |

### 内部接口

表 5.23 接口image\_match

|  |  |
| --- | --- |
| 接口名 | image\_match |
| 接口功能描述 | 通过给定的文本和图片，提供文本和图片的最佳匹配 |
| 参数描述 | form-data{’ text\_array’: ["我王小明第一个不服。你是什么怪物。","你说什么呢。我是一个路人。"] } |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {Array : shape(1, 20, 1024, 1) } |

表 5.24 接口 history\_predict

|  |  |
| --- | --- |
| 接口名 | history\_predict |
| 接口功能描述 | 返回更新过的用户习惯 |
| 参数描述 | {body:Array(shape(5,15)) } |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {Array : shape(1, 20, 1024, 1) } |

表 5.25 接口/temage/product/\_delete\_by\_query/

|  |  |
| --- | --- |
| 接口名 | /temage/product/\_delete\_by\_query/ |
| 接口功能描述 | 删除文件索引 |
| 参数描述 | {body:Array(shape(5,15)) } |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {Array : shape(1, 20, 1024, 1) } |

表 5.26 接口/temage/product/ \_search

|  |  |
| --- | --- |
| 接口名 | /temage/product/\_search |
| 接口功能描述 | 搜索文件索引 |
| 参数描述 | {  "query": {  "bool": {  "should": [  {  "terms": {  "style": [  "math",  "tech"  ]  }  },  {  "match": {  "title": "keywords"  }  }  ]  }  }  } |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {  "took" : 13,  "timed\_out" : false,  "\_shards" : {  "total" : 3,  "successful" : 3,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : 2,  "max\_score" : 1.0,  "hits" : [  {  "\_index" : "temage",  "\_type" : "product",  "\_id" : "nyYTC2kBw7-bFgz3UBhh",  "\_score" : 1.0,  "\_source" : {  "title" : "test title",  "style" : [  "style\_1",  "style\_2"  ],  "ID" : 21  }  },  {  "\_index" : "temage",  "\_type" : "product",  "\_id" : "miazCGkBw7-bFgz3PBgb",  "\_score" : 0.56478655,  "\_source" : {  "title" : "Messi",  "style" : [  "Sport",  "Tech"  ],  "ID" : 1  }  }  ]  }  } |

表 5.27 接口/temage/product/

|  |  |
| --- | --- |
| 接口名 | /temage/product/ |
| 接口功能描述 | 搜索文件索引 |
| 参数描述 | {"title":’title’, "style":style\_names, "ID": 1} |
| 接口间的依赖关系 | 无 |
| 返回参数样例 | {code:200} |

### 用户界面设计

## 数据结构设计

### 逻辑结构设计

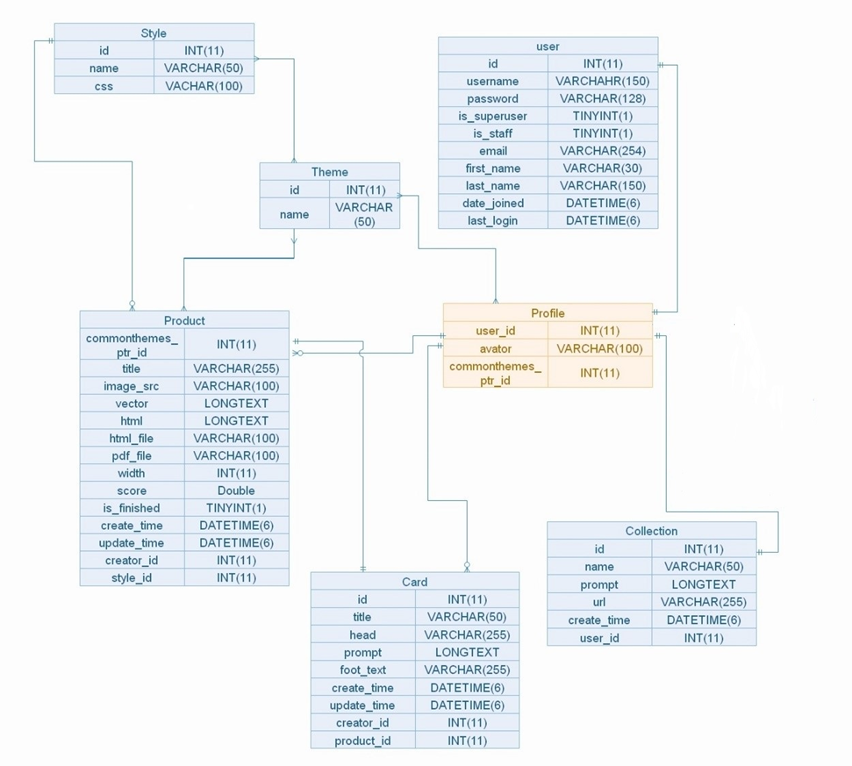


图 5.5 数据库逻辑结构图

数据库逻辑结构图如上图所示。Temage中一共有7个数据结构，由于使用了Django框架中自带的auth\_user，在此基础上设计Profile来代表一个用户，user和Profile之间是一对一的关系。

1. 排版模块

用户创建作品（Product）时，会得到作品的主题标记（Theme），标记可能有一个或以上，由模型处理文字确定或由用户根据实际情况更改。作品根据主题关联上样式（Style），Style中的CSS样式表加上用户的微调生成最后的结果，该结果被保存成html文件，用于生成长图等。最后对保存成功的Product，用户可以选择将其发布成Card，Card是对Product的封装，加入了更多描述性的内容。在从图文匹配模型得到匹配结果之前，先将用户上传的文章和图片存至Cache实例中，再根据得到的匹配结果生成html。

1. 用户习惯追踪模块

通过分析用户的操作，项目可以检测出用户感兴趣的Theme，并将他们关联，用户与Themes是多对多的关系。可以通过用户的Theme检索到拥有相同Theme的作品。

1. 社区模块

每个用户拥有一个收藏夹（Collection），用户可将其他用户的Card加入自己的收藏夹，Card和收藏夹是多对多的关系。

1. 搜索

Temage使用ElasticSearch实现搜索，ElasticSearch创建Index加速搜索。



图 5.6 ES Index概要

图xx为Index的配置，一共存在三个Field——text，style，ID。title储存文章标题，使用ik分词器进行处理；style储存作品可能风格的列表，ID存储MySQL数据库主键。

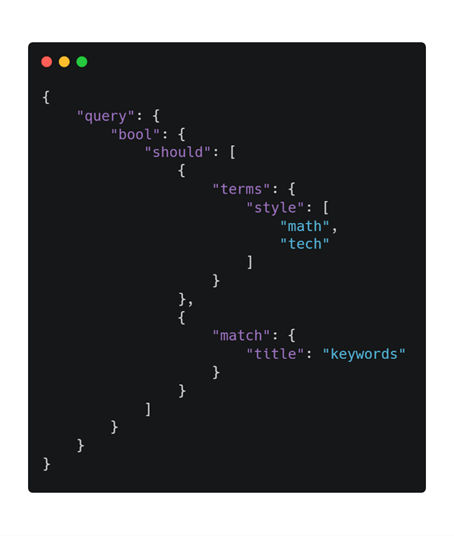


图 5.7 ElasticSearch搜索方式

Temage使用上述的搜索规则实现项目的搜索功能。

### 物理结构设计

1. 存储要求：

以上数据结构均存储在MySQL数据库中。各数据存取的数据类型和长度如图5.1所示。值得注意的是，Profile存取用户头像的avator字段、Product存取作品html文件的html\_file字段和存取长图的image\_src字段、Style存取css样式表的css字段，存取类型为字符串，是对应文件的相对路径。实际文件的存储地点如下：

表 5.28 文件类型和存储位置

|  |  |  |
| --- | --- | --- |
| 文件 | 类型 | 相对路径 |
| 用户头像图片 | .jpg | /media/img/avator |
| 作品长图 | .jpg | /media/img/pimg |
| css样式表 | .css | /media/css |
| 作品html文件 | .html | /media/html |

1. 访问方法：

该项目使用Django的ORM框架对数据库中的数据进行访问。通过存储文件相对路径的字段信息访问文件。

1. 存取单位

表 5.29 数据存取单位

|  |  |
| --- | --- |
| 数据名称 | 存取单位大小 |
| auth\_user | 32KB |
| Profile | 32KB |
| Product | 16KB |
| Card | 32KB |
| Collection | 32KB |
| Theme | 16KB |
| Style | 16KB |

1. 存取的物理关系

i. 索引

本项目使用的MySQL存储引擎是InnoDB引擎，索引采用的是B-Tree。

ii. 设备

宿主机为cluster的k03 node节点，端口为3306，在集群内可用mysql:3306访问

搜索部分的物理逻辑结构如下：

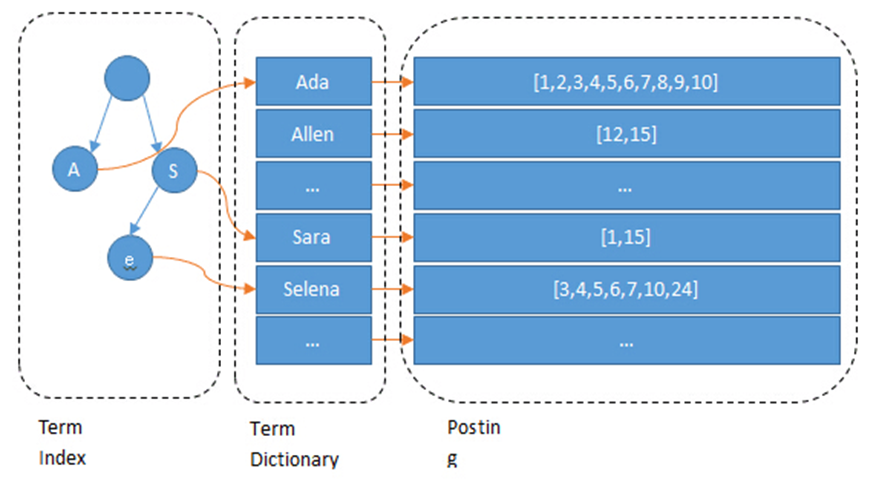


图 5.8 ES物理结构图

1. ES index

ES建立索引的基本结构如图X.X，Term Index通过FST（Finite State Transducers）的压缩技术，其大小压缩到可以缓存在内存中。从term index查到对应的term dictionary的block位置之后，再去磁盘上找term，大大减少了磁盘随机读的次数。

1. （主）分片

Temage结合ElasticSearch与Kubernetes，在ElasticSearch中设置分片实现分布式搜索，ElasticSearch中所有数据均衡的存储在集群中各个节点的分片中，会影响ElasticSearch的性能、安全和稳定性。

1. 副本分片

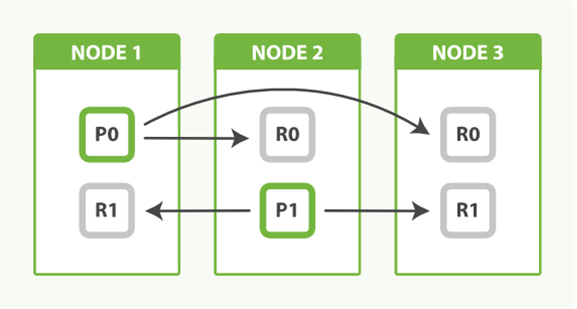


图 5.9 多节点副本分片工作原理

副本分片的主要目的就是为了故障转移，正如在 集群内的原理 中讨论的：如果持有主分片的Node（在部署中指Pod）宕机，一个副本分片就会晋升为主分片的角色。

在索引写入时，副本分片做着与主分片相同的工作。新文档首先被索引进主分片然后再同步到其它所有的副本分片。增加副本数并不会增加索引容量。

无论如何，副本分片可以服务于读请求，如果你的索引也如常见的那样是偏向查询使用的，那你可以通过增加副本的数目来提升查询性能。

分片并不是越多越好，分片过多会带来如下问题：

1. 小的分片会造成小的分段，从而会增加开销。查询大量小的分片将使每个分片的处理速度更快，但是需要按顺序排队和处理更多的任务，它不一定比查询较少数量的较大分片更快。
2. ElasticSearch使用词频统计来计算相关性. 当然这些统计也会分配到各个分片上。如果在大量分片上只维护了很少的数据, 则将导致最终的文档相关性较差。

根据一些经验，最终决定假设将会存在60GB的数据，设置60/30=2个分片，并对每个分片设置1个副本分片。

以上有关数据库和ElasticSearch文件使用Kubernetes的PV(Persistent Volume) / PVC (Persistent Volume Claim)实现，将相关文件资料挂载到宿主机中。

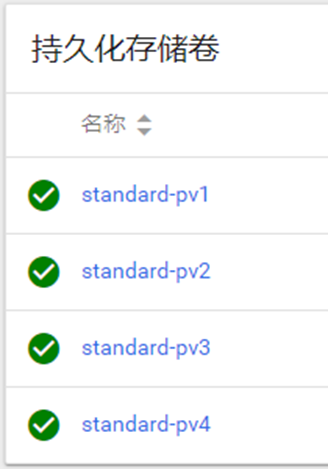


图 5.10 Temage集群中的Persistent Volume

### 数据结构与程序的关系

表 5.30 数据结构与程序关系表

|  |  |  |  |
| --- | --- | --- | --- |
| 模块 | 程序 | 描述 | 涉及的数据结构 |
| 图文排版模块 | get\_homepage\_data | 获取将要显示在主页中的信息。 | Auth\_user, Profile, Card, Product, Collection, Theme |
| get\_gallery\_data | 根据用户兴趣信息获取游廊页面显示的信息 | Auth\_user, Profile, Product, Card, Theme |
| get\_gallery\_more\_data | 同上，用于翻页或下拉获取更多的情况 | Auth\_user, Profile, Product, Card, Theme |
| get\_collection\_data | 获取用户收藏夹信息 | Auth\_user, Profile, Product, Card |
| get\_recent\_data | 获取用户最近编辑的作品信息 | Auth\_user, Profile, Product |
| get\_product | 给出某篇作品的详细信息 | Auth\_user, Profile, Product, Card, Theme, Style |
| post\_collect | 用于用户收藏自己感兴趣的内容 | Auth\_user, Profile, Product, Card, Collection |
| cancel\_collect | 用于用户取消收藏 | Auth\_user, Profile, Product, Card, Collection |
| delete\_product | 用于用户删除作品 | Auth\_user, Profile, Product，Card |
| post\_search | 用于用户搜索 | Auth\_user, Profile, Product, Card, Collection |
| 用户模块 | login\_submit | 用于用户登录 | Auth\_user, Profile |
| JWTauthenticate | 用于认证用户登录状态 | Auth\_user, Profile |
| register | 用户注册 | Auth\_user, Profile |
| 用户社区模块 | post\_picture | 用于前端提交图片到模型进行运算 | Auth\_user, Profile, Cache |
| post\_text | 用于前端提交文章到模型进行运算 | Auth\_user, Profile, Cache |
| push\_match\_event | 调用模型进行图文匹配，返回表示图文匹配结果的list | Auth\_user, Profile, Cache |
| confirm\_style | 以字符串形式返回模型运算后生成的html | Theme，Style，Auth\_user, Profile, Cache |
| store\_passage | 初步存储作品 | Auth\_user, Profile, Product, Theme, Style |
| finished\_work | 用于用户微调结束后的作品整体效果预览 | Product |
| confirm\_store | 用户最后根据满意度打分并发布 | Auth\_user, Profile, Product, Card |
| download\_picture | 用于用户最后作品生成长图并下载 | Product |

## 运行设计

### 运行模块关系

1. 用户社区模块组合

a) 用户注册、用户习惯初始化，用户习惯预测结合。

b) 用户登录、优秀作品浏览、优秀作品收藏、优秀版式克隆结合。

c) 用户登录、优质作品收藏、用户习惯记录，用户习惯预测结合。

2. 智能图文排版模块组合

a) 用户登录、图片预处理、文本预处理、图文匹配结合。

b) 用户登录、文本主题分类、版式选择、图文匹配、排版美化结合。

c) 用户登录、文本主题分类、用户习惯记录、用户习惯预测结合

d) 用户登录、用户反馈记录、模型自我提升结合

### 运行控制

1. 用户社区运行控制

a) 用户必须登录后才能进行相关操作。

b) 用户必须进入到具体作品后才能进行收藏和克隆的相关操作。

c) 用户必须在注册之后才能进行习惯记录和习惯预测。

2. 智能图文排版运行控制

a) 用户必须在登录之后才能进行作品生成的相关操作。

b) 用户必须在上传文本之后才能进行文本分类。

c) 用户必须在上传图片和文本后才能进行图文匹配。

d) 用户必须在选定风格之后才能进行排版美化。

e) 用户必须在风格融合完成后才能进行的用户反馈记录。

### 运行时间

1. 用户社区运行时间

a) 耗时较少的功能模块

用户注册、用户登录、优秀作品浏览、优秀作品收藏、优秀作品克隆

b) 耗时较多的功能模块

产品关键词搜索、用户习惯搜索。

2. 智能图文排版运行时间

a) 耗时较少的功能模块

用户反馈记录\版式选择、排版美化、

b) 耗时较多的功能模块

模型自我提升、图片预处理、文本预处理、图文匹配、文本主题分类。

## 错误/异常处理设计

### 错误/异常输出信息

图 5.11 Sentry的使用

Sentry 是一个开源的实时错误报告工具，支持 web 前后端、移动应用以及游戏。Sentry支持多用户、多团队、多应用管理 Temage使用Sentry对运行时错误进行日志记录。

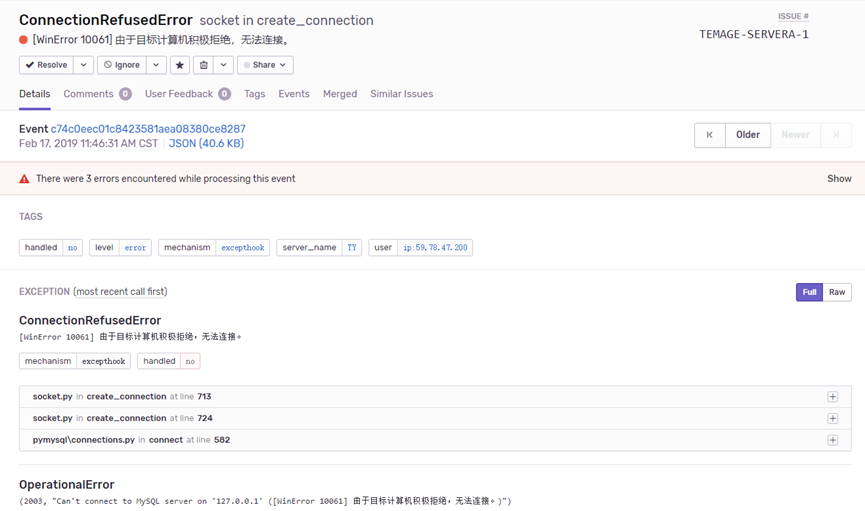


图 5.12 Sentry错误日志

### 错误/异常处理对策

Temage计划使用Docker + Kubernetes + Istio实现Microservice架构，能够性能降级、恢复再启动。

1. 性能降级

由于ServerB提供深度学习模型的inference，会消耗大量的内存和CPU资源，为防止服务器过载，本项目在服务器压力达到阈值后用Istio实现性能降级，不允许用户进行图文匹配等有关深度学习模型相关API的调用，但仍支持用户对原有作品进行收藏修改等一系列操作。

1. 恢复再启动

Kubernetes使用Deployment对Pod进行管理。无响应的Pod进行删除，并启用副本集进行替代，Deployment会自动维持Pod的数量。当Pod里的某个容器一段时间未响应的情况下，Kubernetes会主动kill该容器并创建新的容器，如果Pod所在的Node宕机，则会将这个Node上的所有Pod重新调度到其他节点上，用以维持Deployment中规定的数量。

1. 设置后备

Temage将会对数据库进行定时主动备份，当数据库数据损毁时，可立即启用备份资源，将损失降到最小。

1. ES副本分片恢复机制

ES副本分片的主要目的就是为了故障转移，如果持有主分片的节点宕机，一个副本分片就会晋升为主分片的角色，同时会再生成一个副本分片，维持副本分片的数量要求，增强集群的稳定性。

5. 实时监控

 利用Prometheus + Grafana搭建监控平台，对项目进行实时的监控，做出及时的调整，保证系统的安全。

图 5.13 使用Prometheus + Grafana监控集群

## 系统配置策略



图 5.14 Kubernetes + Istio联合与分工

1. 负载均衡

Temage选择使用Istio实现负载均衡，并且使用权重最小策略。

1. 断路器

Istio中设置断路器，对Pod进行被动的健康检查，设定不健康规则，并且将不健康Pod 移除负载均衡池。

1. 设置请求超时

对于ServerB的请求，超时时间可适当延长，为30s，对于ServerA的请求，超时时间 要适当缩短，设置为2s。

有关Kubernetes和Istio的相关配置文件详见cluster\_config文件夹。

## 系统部署方案

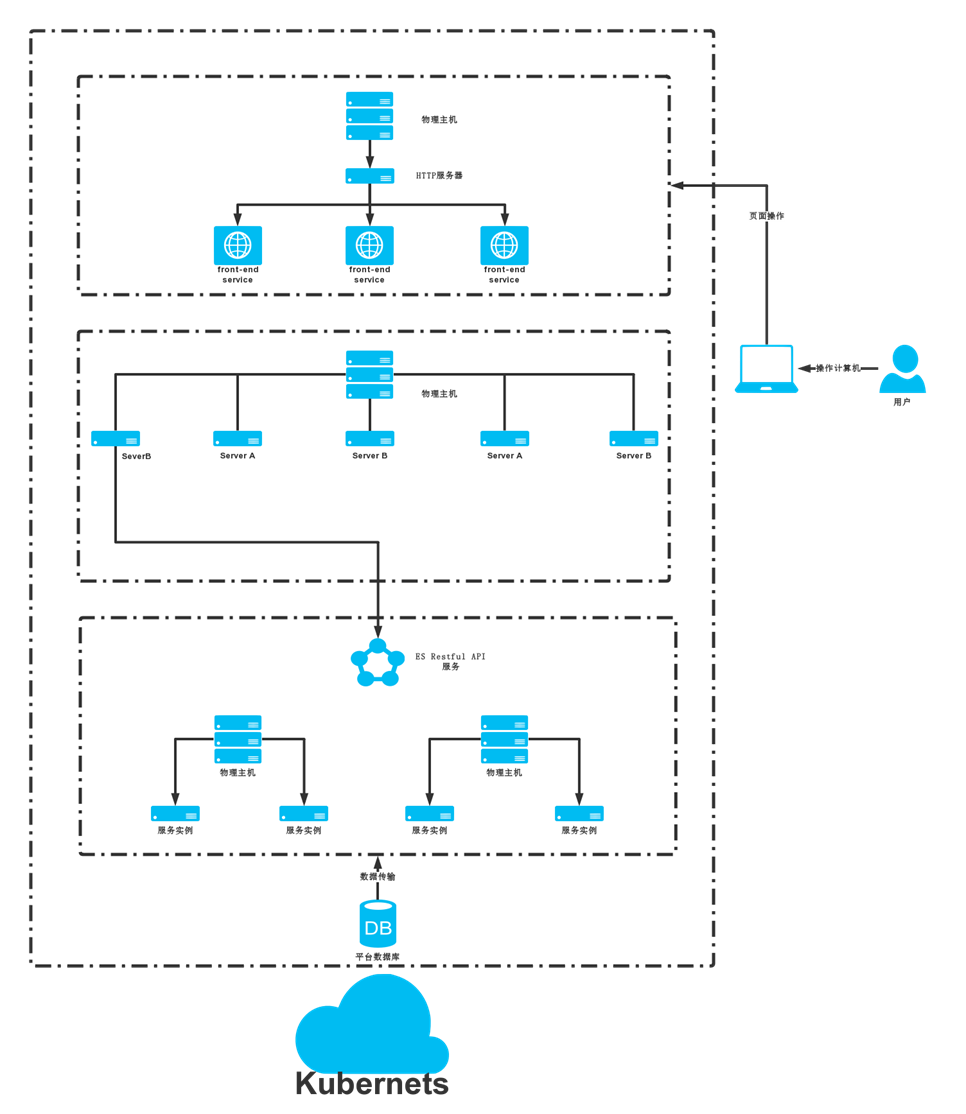


图 5.15 Temage部署结构图

Temage计划使用Kubernetes + Istio在集群上进行部署。Temage的集群资源为4核8G虚拟机，数量为3，以一台为Master，其余为Node。

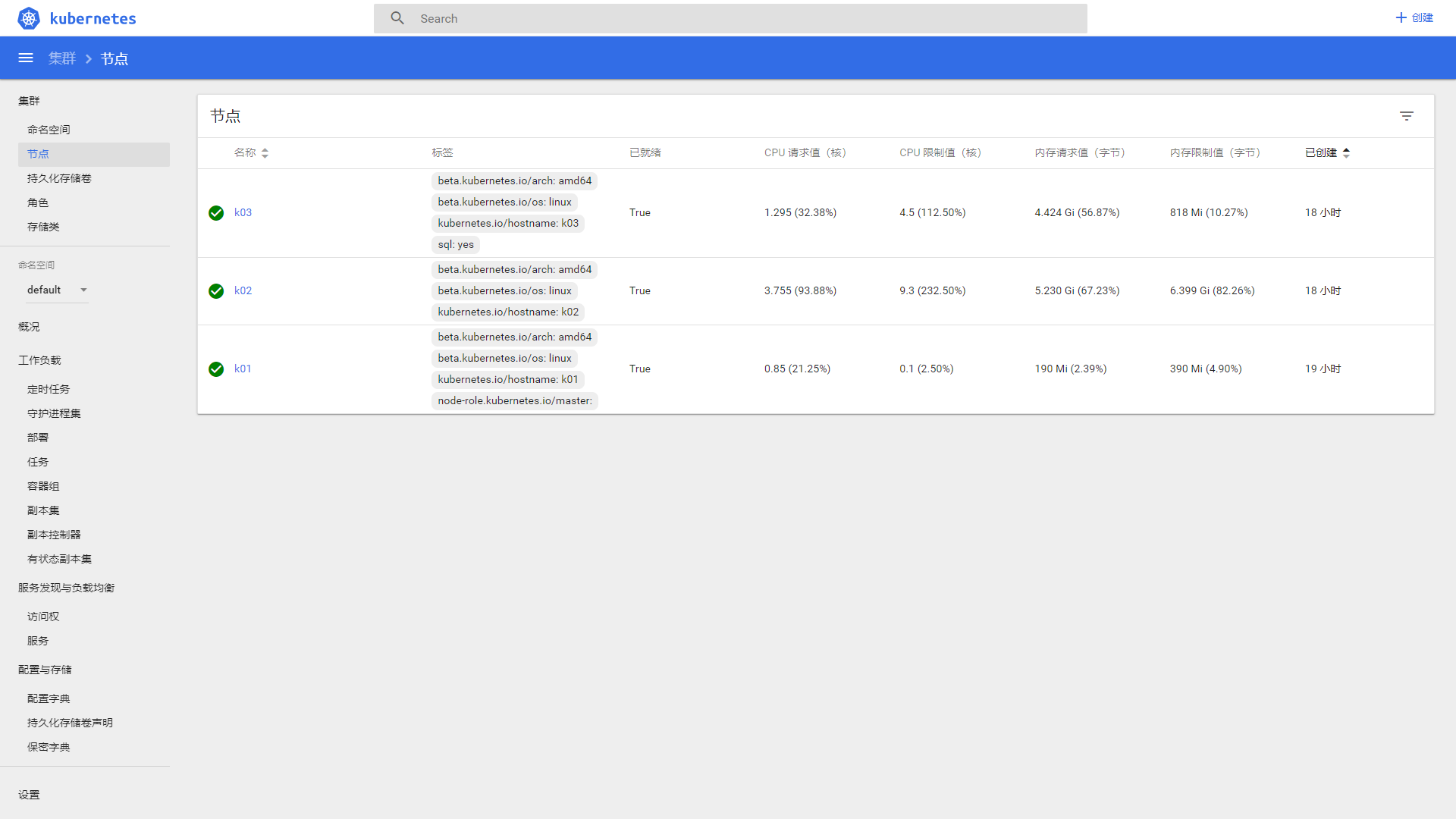


图 5.16 kubedashboard上查看节点信息

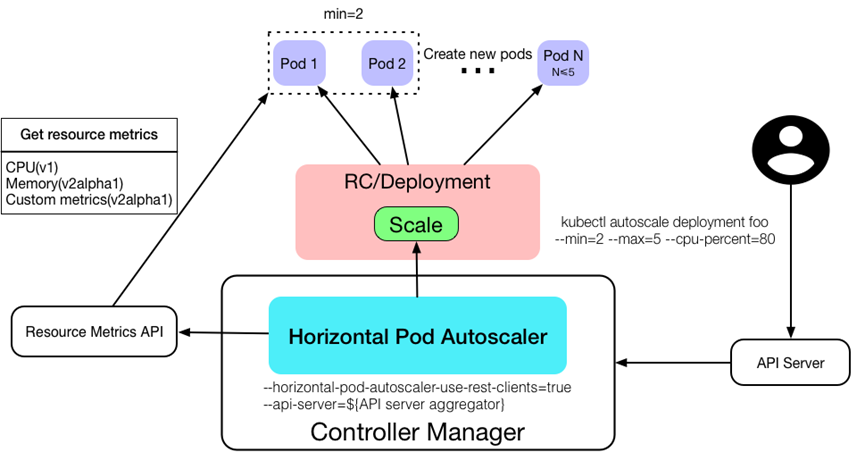
Kubernetes可以使用Deployment + HPA(Horizontal Pod Autoscaling)维护Pod的总数，对无响应的Pod进行删除，并创建新的Pod。Metrics-server将会监察每个Pod的情况，当CPU负载达到一定数值时，将会触发水平扩展创建新的Pod，保证服务的稳定。

图 5.17 HPA & Deployment

具体细节见cluster\_config文件夹中的配置文件。

## 代码规范

1. 后端代码规范：

遵守官方代码规范：

* 1. Tornado风格：
     1. 遵守PEP8风格指南
     2. 使用4个空格进行缩进
     3. 使用下划线而不是小驼峰来命名变量、函数名和方法名
     4. 使用首字母大写的方式来命名类
     5. 每个python脚本文件开头为#-\*-coding:utf-8-\*-，默认保存为utf-8格式
     6. Import语句分行书写
     7. 在二元运算符两边各空一格 [=,-,+=,==,>,in,is not, and]，函数的参数列表中，,之后要有空格，左括号之后，右括号之前不要加多余的空格
  2. Django风格：
     1. 模块级函数和类定义之间空两行，类成员函数之间空一行，可以使用多个空行分隔多组相关的函数
     2. 每个方法都有文档字符串进行描述
     3. 文档字符串中，使用动词（action words）
     4. 在视图中，视图函数的第一个参数应该被命名为 request。
     5. class Meta 应该在字段被定义后才出现，使用一个空行分隔。
     6. model 内的类和方法的定义顺序应该遵循如下顺序（不是所有项都是必需的）：
        + 所有的数据库字段
        + 自定义管理器属性（manager attributes）
        + class Meta
        + def \_\_unicode\_\_()
        + def \_\_str\_\_()
        + def save()
        + def get\_absolute\_url()
        + 其他自定义方法

1. 前端代码规范

前端ESLint recommended规范，详见https://cn.eslint.org/docs/rules/?tdsourcetag=s\_pcqq\_aiomsg

## 其他相关技术与方案

无

# 数据库设计

数据库的设计一定程度上影响着项目的效率和服务质量。数据库设计的一项基本原则就是既能满足应用需求又具有良好的数据库性能。根据用户的需求，设计出的ER图如图6.1所示。

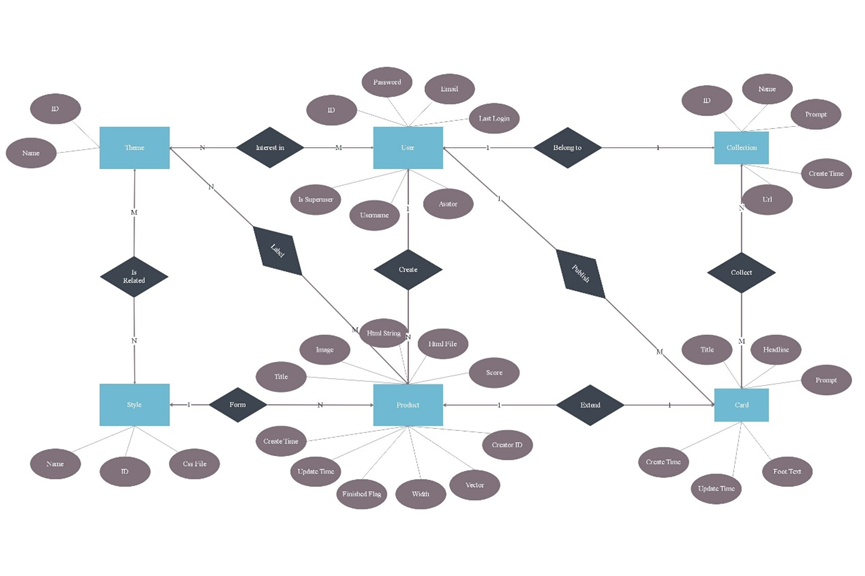


图 6.1 数据库设计ER图

下面给出数据库的部分设计原则：

1. 总体设计：
   1. 数据库字符集严格设置为UTF-8。
   2. 数据库引擎使用支持更多事务处理的InnoDB
   3. 数据库表结构设计满足第三范式（Third Normal Form,3rd NF），即表中所有数据元素不仅能唯一被主关键字所标识，而且它们之间还必须独立。
2. 命名规范：
   1. 表名根据实体直接命名，便于理解与维护。
   2. 字段命名小写加下划线，简洁易懂，便于维护。
3. 字段数据类型设计规范：
   1. 时间类型的字段使用datetime类型，根据时间记录目的的不同，设置为“创建时”和“更新时”两类。
   2. 对于已知长度有限的字符串，使用varchar类型，而对于长度未知的字符串使用longtext类型。
4. 主键设计规范：

类型为INT的主键一般设置为自增（AUTO\_INCREMENT）, 而有遇到类型为非INT的主键，一般使用服务器查重无误后再创建。

1. 特殊说明：

由于用户和作品跟主题都是多对多的关系，构成一个多态多对多的关系，在实现过程中用户和作品都继承自一个名为Commonthemes（共有主题）的父类，它们的id命名为commonthemes\_ptr\_id。

如下图是数据库表结构图：

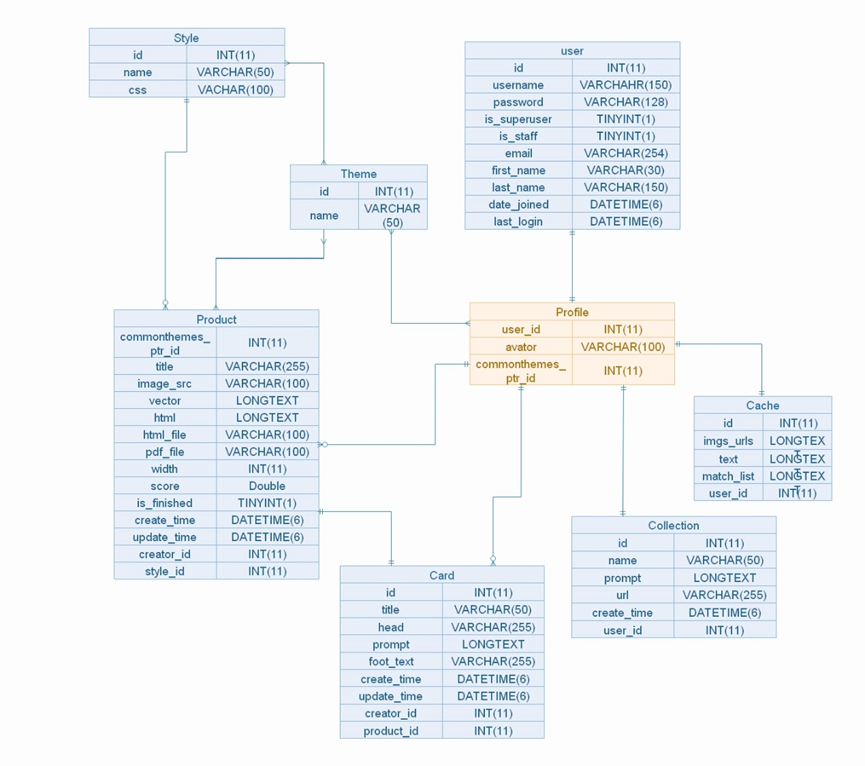


图 6.2 数据库结构图

表字段描述如下：

表 6.1 Profile

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 备注 |
| user\_id | INT(11) | N |  | FK | 对应的auth\_user的ID编号 |
| avator | VARCHAR(100) | N |  |  | 用户头像的相对存储路径 |
| commonthemes\_ptr\_id | INT(11) | N |  | PK | 多态关系继承产生的id |

表 6.2 auth\_user

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 备注 |
| id | INT(11) | N |  | PK | ID编号 |
| username | VARCHAR(150) | N |  |  | 用户名 |
| password | VARCHAR(128) | N |  |  | 密码 |
| is\_superuser | TINYINT(1) | N |  |  | 是否为超级用户 |
| is\_staff | TINYINT(1) | N |  |  | 是否为员工 |
| email | VARCHAR(254) | N |  |  | 邮箱地址 |
| first\_name | VARCHAR(30) | Y |  |  | 名 |
| last\_name | VARCHAR(250) | Y |  |  | 姓 |
| date\_joined | DATETIME(6) | N |  |  | 用户创建日期 |
| last\_login | DATETIME(6) | N |  |  | 上次登录日期 |

表 6.3 Theme

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 备注 |
| id | INT(11) | N |  | PK | ID编号 |
| name | VARCHAR(50) | N |  |  | 主题的名称 |

表 6.4 Style

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 备注 |
| id | INT(11) | N |  | PK | ID编号 |
| name | VARCHAR(50) | N |  |  | 样式名称 |
| css | VARCHAR(100) | N |  |  | 样式表的相对路径 |

表 6.5 Product

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 备注 |
| commontheme\_ptr\_id | INT(11) | N |  | PK | 多态关系产生的id |
| title | VARCHAR(255) | N |  |  | 作品标题 |
| image\_src | VARCHAR(100) | N |  |  | 作品长图的相对路径 |
| vector | LONGTEXT | N |  |  | 模型处理作品产生的向量 |
| html | LONGTEXT | Y |  |  | 生成结果的html字符串 |
| html\_file | VARCHAR(100) | N |  |  | 生成结果的html文件的相对路径 |
| pdf\_file | VARCHAR(100) | Y |  |  | 根据html生成的pdf文件 |
| width | INT(11) | N |  |  | 长图设置的宽度 |
| score | DOUBLE | Y |  |  | 用户评分 |
| is\_finished | TINYINT(1) | N | 0 |  | 是否完成并发布 |
| create\_time | DATETIME(6) | N |  |  | 创建时间 |
| update\_time | DATETIME(6) | N |  |  | 修改时间 |
| creator\_id | INT(11) | N |  | FK | 创建者的id |
| style\_id | INT(11) | N |  | FK | 使用样式的id |

表 6.6 Card

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 备注 |
| id | INT(11) | N |  | PK | ID编号 |
| title | VARCHAR(50) | N |  |  | 卡片标题 |
| head | VARCHAR(255) | Y |  |  | 卡片描述字段1 |
| prompt | LONGTEXT | Y |  |  | 卡片描述字段2 |
| foot\_text | VARCHAR(255) | Y |  |  | 卡片描述字段3 |
| create\_time | DATETIME(6) | N |  |  | 创建时间 |
| update\_time | DATETIME(6) | N |  |  | 更新时间 |
| creator\_id | INT(11) | N |  | FK | 创建者id |
| prduct\_id | INT(11) | N |  | FK | 对应作品id |

表 6.7 Collection

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 备注 |
| id | INT(11) | N |  | PK | ID编号 |
| name | VARCHAR(50) | N |  |  | 收藏夹名称 |
| prompt | LONGTEXT | Y |  |  | 收藏夹描述字段 |
| url | VARCHAR(255) | N |  |  | 收藏夹链接 |
| create\_time | DATETIME(6) | N |  |  | 创建时间 |
| user\_id | INT(11) | N |  | FK | 对应用户id |

表 6.8 Cache

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 备注 |
| id | INT(11) | N |  | PK | ID编号 |
| img\_urls | LONGTEXT | Y |  |  | 缓存用户上传的图片的路径列表 |
| text | LONGTEXT | Y |  |  | 缓存用户上传的文章 |
| match\_list | LONGTEXT | Y |  |  | 图文匹配模型返回的匹配结果 |
| user\_id | INT(11) | N |  | FK | 对应用户id |

# 详细设计

Temage设计总览如下表：

表 7.1 详细设计表

|  |  |  |  |
| --- | --- | --- | --- |
| **功能模块** | **功能** | **功能描述** | **优先级** |
| **图文排版模块** | 文本主题分类 | 使用TextCNN模型对用户上传文本进行分析，对文本进行分类，为作品提供关键字，影响后期美化 | 10 |
| 图文匹配 | 使用双分支模型进行图文匹配，形成初排版 | 10 |
| 风格融合 | 将用户选择的风格融合到作品中，进行美化 | 10 |
| **用户社区模块** | 优秀产品推荐 | 以LSTM为基础建立模型，对用户的习惯进行追踪，推断的风格选择 | 10 |
| 用户反馈学习 | 记录用户最终选择的风格于用户习惯中，进一步更新用户习惯模型和文本分类模型 | 8 |
| 用户搜索推荐 | 用户使用关键词搜索提供相应作品的显示，供用户参考 | 8 |
| 优质版式分享 | 用户可以对平台中优质的作品进行二次创作 | 7 |

## 图文排版功能模块

### 功能描述

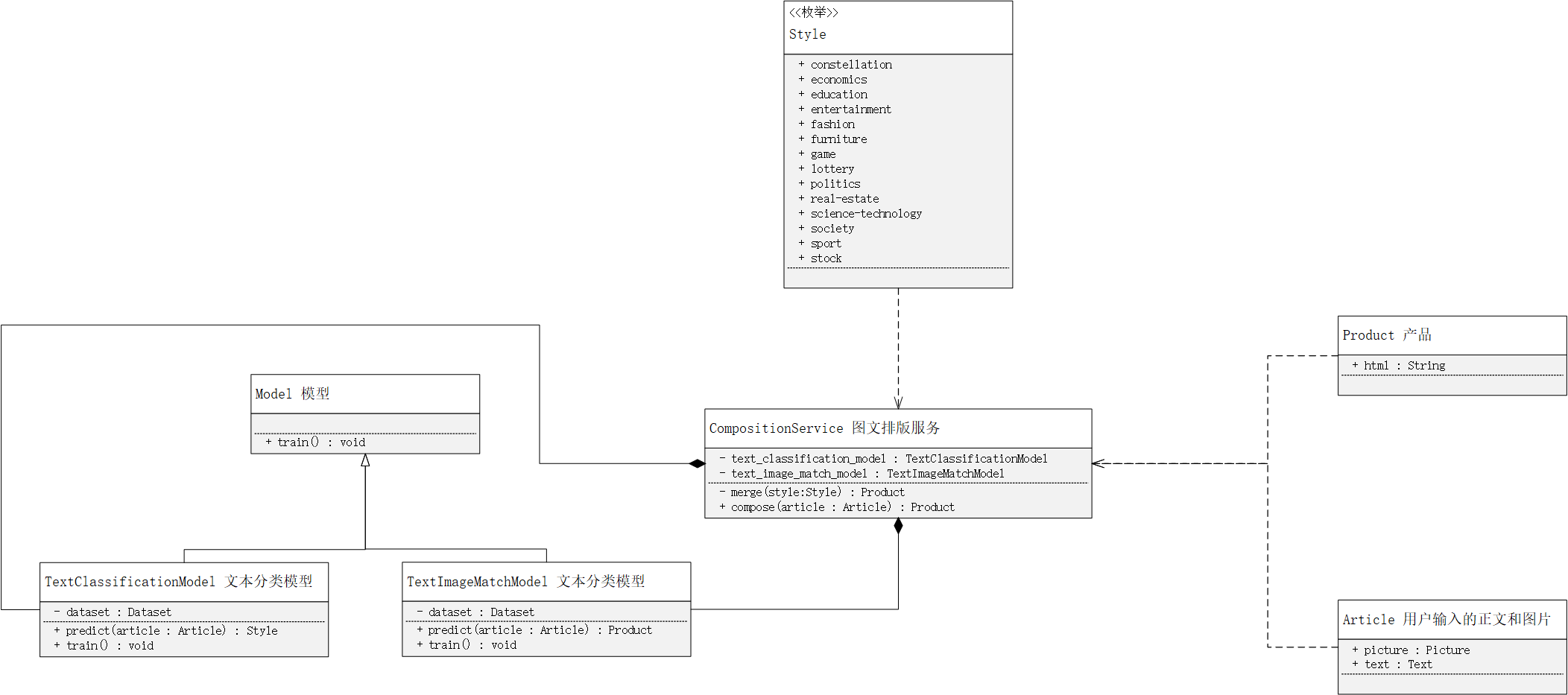
用户依次上传文本和图片、进行文本主题分类和图文匹配，在选定风格之后最终进行风格融合，最终用户给出对作品的反馈。该功能模块中的三个功能为强内聚（顺序内聚）。

图 7.1 图文排版功能模块的UML类图

### 性能描述

图文排版模块的性能瓶颈在图文匹配和文本主题分类中使用到深度学习模型进行。对这两个子模块进行基础性能测试后，结果如下：

表 7.2 图文排版功能模块性能描述表

|  |  |  |  |
| --- | --- | --- | --- |
| 图文排版功能模块性能描述 | | | |
| 服务端硬件描述 | | CPU：4核，  内存：8GB，  带宽：1Mbps | |
| 客户端硬件描述 | | CPU：2核，  Intel：酷睿i5  主频：2.7GHz  内存：双通道8G | |
| 子模块名称 | 模型/应用名称 | | 平均响应时间/s |
| 文本主题分类 | 文本分类模型（TextCNN） | | 856ms（前端推断）+208ms（数据传输）+5522ms（后端词嵌入） |
| 图文匹配 | 双分支匹配模型 | | 408ms（数据传输）+9927ms（后端推断） |

### 输入

表 7.3 图文排版功能模块输入表

|  |  |  |
| --- | --- | --- |
| 图文排版功能模块输入 | | |
| 功能 | 输入名称 | 输入类型 |
| 文本主题分类 | -- | -- |
|  | 文本 | String |
| 图文匹配 | -- | -- |
|  | 文本 | String |
|  | 图片 | File |
| 风格融合 | -- | -- |
|  | 作品ID | Int |
|  | 作品最终风格 | Int |
| 用户反馈 |  |  |
|  | 作品ID | Int |
|  | 作品评价 | Int |

### 输出

表 7.4 图文排版功能模块输出表

|  |  |  |
| --- | --- | --- |
| 图文排版功能模块输出 | | |
| 功能 | 输出名称 | 输出类型 |
| 文本主题分类 | -- | -- |
|  | 各主题概率 | Array |
| 图文匹配 | -- | -- |
|  | 图片文本初排版 | String |
|  |  | File |
| 风格融合 | -- | -- |
|  | 作品ID | Int |
|  | 作品最终风格 | Int |
| 用户反馈 | -- | -- |
|  | 无 | -- |

### 算法

N张图片和M个段落在通过双分支模型推断之后，最终将得到一个m\*n的矩阵，本项目需要提出一种策略来得到最终的最佳匹配模式。

本项目使用贪心算法来实现最佳匹配：

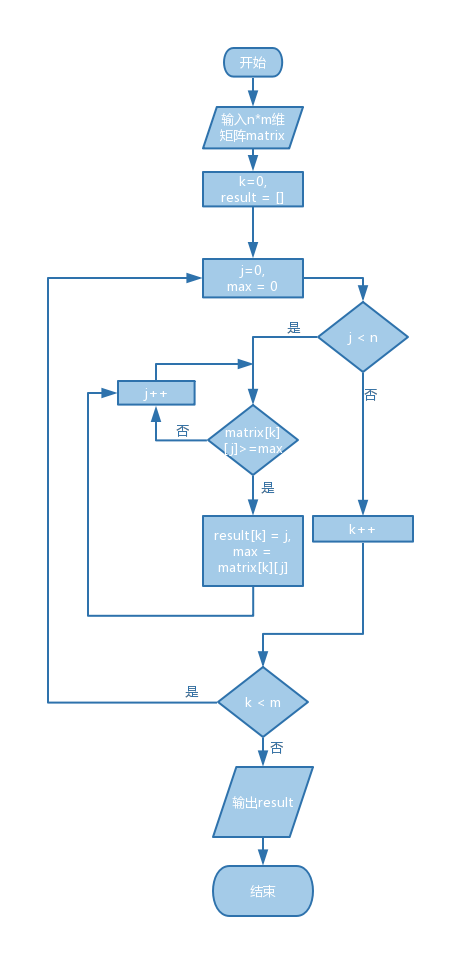


图 7.2 图文排版功能模块的算法流程图

### 程序逻辑

用户依次进行文本主题分类、图文智能匹配、风格融合和作品反馈这四个步骤，最终生成作品公开发表。

1. 在用户进入工作区开始创作时，要求用户输入扁平化的图文信息；

2. 在用户触发下一步按钮后，客户端向ServerB请求embedding信息，同时向ServerA缓存图文信息，ServerA向ServerB发送图文缓存信息，进行图文匹配；

3. 客户端得到词嵌入后，客户端立即进行推断文本分类，供用户选择；

4. 用户确定风格后，返回智能图文排版结果，进行微调；

5. 生成产品，得到用户反馈，助力模型发展

### 接口

表 7.5 图文排版功能模块接口表

|  |  |
| --- | --- |
| 图文排版功能模块接口 | |
| 接口名称 | 接口功能 |
| post\_text | 接收用户上传的文本 |
| post\_picture | 接收用户上传的图片 |
| push\_match\_event | 通知后端进行推断 |
| post\_confirmed\_style | 返回对应产品的html的最终稿 |
| store\_passage | 储存对应产品的评分和反馈 |
| finished\_work | 返回对应产品的html的最终稿 |
| Download | 下载对应产品的html |
| confirm\_store | 确定最终储存成品，发布 |
| Embedding | 返回文段对应的embedding |
| image\_match | 通过给定的文本和图片，提供文本和图片的最佳匹配 |
| history\_predict | 返回更新过的用户习惯 |

### 存储分配

explore模块的存储空间分为两类，文件存储空间和数据存储空间。文件存储包括用户的头像jpg文件，已完成作品的长图文件、html文件、pdf文件等，样式表css文件。数据存储包括其他非文件内容，如用户的基本信息，作品和收藏夹的创建信息、描述信息，主题和样式基本信息等，完整列表请查看数据库设计部分。

用于用户习惯追踪模型大小约为100MB。

每个用户的除作品以外的信息，包括用户个人信息、头像、收藏夹、兴趣等，大小大约为1MB。

在explore模块上为每位用户分配大约2MB的存储空间。

### 限制条件

无。

## 用户社区模块

### 功能描述

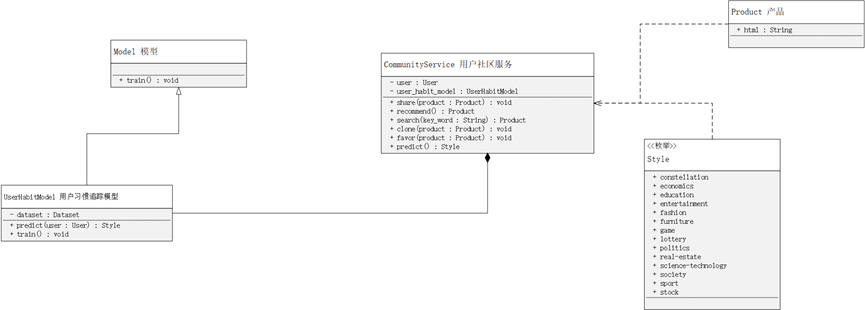


图 7.3用户社区模块UML类图

### 性能描述

用户社区模块的性能瓶颈在于关键字搜索和用户习惯追踪，性能测试结果如下：

表 7.6 用户社区模块性能描述表

|  |  |  |  |
| --- | --- | --- | --- |
| 用户社区模块性能描述 | | | |
| 服务端硬件描述 | | CPU：4核，  内存：8GB，  带宽：1Mbps | |
| 客户端硬件描述 | | CPU：2核，  Intel：酷睿i5  主频：2.7GHz  内存：双通道8G | |
| 子模块名称 | 模型/应用名称 | | 平均响应时间/s |
| 用户习惯追踪 | 习惯追踪模型（TextCNN） | | 188ms（数据传输）+3218ms（后端推断） |
| 关键字搜索 | 双分支匹配模型 | | 83ms（ElasticSearch 检索）+245ms（后端返回数据） |

### 输入

表 7.7 用户社区功能模块输入表

|  |  |  |
| --- | --- | --- |
| 用户社区功能模块输入 | | |
| 功能 | 输入名称 | 输入类型 |
| 优质模板分享 | -- | -- |
|  | 用户信息 | Int |
| 用户习惯追踪 | -- | -- |
|  | 用户信息 | Int |
| 优质产品推荐 | -- | -- |
|  | 用户信息 | Int |
|  | 关键字 | String |

### 输出

表 7.8 用户社区功能模块输出表

|  |  |  |
| --- | --- | --- |
| 用户社区功能模块输出 | | |
| 功能 | 输出名称 | 输出类型 |
| 优质模板分享 | -- | -- |
|  | 作品详情 | Cards |
| 用户习惯追踪 | -- | -- |
|  | 用户习惯 | Vector |
| 优质产品推荐 | -- | -- |
|  | 作品详情 | Cards |

### 算法

用户社区模块重点在于提升用户体验感，重点优化前端UI的舒适性，没有特殊算法的使用。

### 程序逻辑

用户可以随意在游廊中进行游览，可以进行下拉、搜索、收藏和克隆等一系列操作。

### 接口

表 7.9 用户社区功能模块接口表

|  |  |
| --- | --- |
| 用户社区功能模块接口 | |
| 接口名称 | 接口功能 |
| get\_homepage\_data | 获取首页相关信息 |
| post\_register | 用户注册信息收集 |
| post\_login\_submit | 用户登录 |
| post\_jwt\_authenticate | 使用jwt\_code换取用户信息 |
| get\_gallery\_data | 获取游廊信息 |
| get\_gallery\_more\_cards | 更新游廊信息 |
| post\_search | 通过关键字搜索 |
| get\_collection\_data | 获取收藏夹信息 |
| get\_recent\_data | 获取近期操作过的产品信息 |
| get\_product | 通过ID获取产品信息 |
| post\_collect | 登记某用户收藏某作品 |
| delete\_product | 删除作品 |
| cancel\_collect | 取消某用户收藏某作品 |

### 存储分配

workflow模块的存储空间分为两类，文件存储空间和数据存储空间。文件存储包括用用户创建作品时缓存的图片文件、html文件、pdf文件、长图文件，以及作品使用的样式表CSS文件。数据存储包括其他非文件内容，如作品的其他详细信息，主题和样式基本信息等， 完整列表请查看数据库设计部分。  
 图文匹配的模型大小为203MB，文本分类模型大小为9MB。每个用户在工作区的缓存文件大小约为3MB。每个非会员用户最多可在数据库中保留五篇作品的数据和文件，预计大小为25MB。在workflow模块上为每位用户分配大约30MB的存储空间。预计一年内本项目用户数达到1k，需要的存储空间约为30GB。根据分配给用户的存储空间和分配给软件的存储空间总和，总共分配约31GB的存储空间。为了保持项目的弹性，预计分配40GB的存储给初期项目。

### 限制条件

无。