研 究 生 毕 业 论 文

（申请工程硕士学位）

论 文 题 目	微信公众号图文系统的设计与实现
作 者 姓 名	
学科 专业名称	工 程 硕 士（软 件 工 程 领 域）
研 究 方 向	软件工程
指 导 教 师	

2019 年 4 月 10 日

学　　号：

论文答辩日期：　　年　月　日

指 导 教 师：　　（签字）

微信公众号图文系统的设计与实现

作　　者:

指导教师:

XXX **大学研究生毕业论文**

（申请工程硕士学位）

XXX **大学** XXX **学院**

2019 **年** 4 **月**

The Design and Implementation of WeChat official account graphic system

(Author Name)

Submitted in partial fulfillment of the requirements for the degree of Master of Engineering

Supervised by

(Supervisor's position) (Supervisor's Name)

XXX Institute

XXX UNIVERSITY

Nanjing, China

April, 2019

摘　要

随着 web2.0 技术的发展，国内外涌现许多新兴的自媒体平台。早期，人们通过订阅报纸杂志、浏览博客等方式获取信息。自媒体平台的出现，改变了原来单向的信息传播方式，越来越多人利用自媒体平台发声。微信公众号作为自媒体平台，倡导再小的个体，也有自己的品牌。网页端公众平台系统作为信息发送端，移动端微信公众号图文系统作为信息接收端，共同构成微信公众平台项目。

本文的主要工作是描述微信公众号图文系统的设计与实现。该系统使用混合式应用的开发模式，使用 SeaJS 框架作为前端模块化管理工具，用 RPC 服务框架 Svrkit 作为后台服务框架，应用 QQMail 模板引擎实现数据层与表现层分离。微信公众号图文系统作为面向订阅者的项目，实现了图文消息推送机制。该推送机制使多 IDC 机器间能够通信，能及时、准确的推送公众号发布的文章，并将文章以 timeline 的形式向订阅者展示。系统用预加载、数据离线化、分段渲染等手段实现图文秒开模块，在微信客户端场景下，读者无需再等待进度条加载，能立刻看到想要阅读的文章。为了提供给用户更佳的阅读体验，在图文信息展示方面，本项目对图片采用 HEVC 编码方式以节省带宽流量，应用超分辨率重建技术还原图片的清晰度。系统不仅实现图片、音频等基本展示功能，还提供视频、卡券、小程序等更加丰富的信息表现形式。在提供阅读功能的基础上，系统基于 geta8key 技术获得微信用户登录态信息，提供点赞、留言、在看、投诉、分享等互动功能，增加阅读的趣味性和社交性。为了提供给写作者更优质的内容产出环境，公众平台推出洗稿合议机制，打击洗稿等违规行为。

微信公众号图文系统为了提供更好的用户体验和构建更优质的自媒体平台环境，在推送形式、展示与互动方式上做了重新设计与改进。不仅为公众平台带来了收益，也使公众号能够应对各种“号”形式的自媒体风潮。

关键词：微信公众号、自媒体、SeaJS 框架、QQMail 模板、HEVC

Abstract

With the development of web2.0 technology, many We-media platforms have emerged at home and abroad. In the early days, people obtained information by subscribing to newspapers and magazines, browsing blogs, and so on. The emergence of the We-media platform has changed the way of one-way information dissemination. More and more people use the We-media platform to express themselves. The WeChat public account, as a We-media platform, advocates small individual also has its own brand. The web-side official platform system serves as the information sending end, and the mobile-side official account graphic system serves as the information receiving end to jointly form the WeChat official platform project.

The main work of the article is to describe the design and implementation of the WeChat official account graphic system. The project uses the development way of the Hybrid App. It uses the SeaJS framework as the front-end module management tool. The RPC service framework Svrkit is used as the back-end service framework. The QQMail template engine is used in the project to separate the data layer from the presentation layer. As a subscriber-oriented project, the official account grahic system implements a message push mechanism capable of communication between multiple IDC machines. The subscriber can receive the article pushed by official account in the form of timeline timely and accurately. The pre-loading, data offline, segmentation rendering, and other technologies are used to implement the graphic second opening module. In the WeChat App, the reader can immediately see the article selected for reading without waiting for the progress bar to load. In order to provide users with a better reading experience, the image is HEVC encoded to save bandwidth traffic, and the super-resolution reconstruction technology is applied to restore the clarity of the image. It not only realizes the basic display function of pictures and audio, but also provides abundant information expressions such as videos, card coupons and mini programs. Besides providing reading function, the system also provides interactive functions such as praise, comment, wow, report, share and so on, based on geta8key technology to obtain the information of WeChat users, to

increase the interest and sociality of reading. In order to provide writers with a better-quality content production environment, a cleansing collegiate mechanism was launched to combat violations which use unauthorized content or misuse of original content.

In order to provide a better user experience and build a better We-media platform environment, WeChat official account graphic system has been redesigned and improved in push, display and interaction way. Not only does it bring benefits to the public platform, but it also enables the WeChat Official Account Platform to cope with the various “Account” forms of We-media.

Keywords： WeChat Official Account, We-media, SeaJS, QQMail template, HEVC

目　录

图目录

表目录

第一章　引言

1.1 项目背景与意义

随着互联网技术的发展，诞生了“自媒体”这一概念，自媒体包括微信、微博等各种网络传播媒介。自媒体的诞生改变了传统单向发声的信息传播模式，用户可以用极其简单的方式创作出优质的内容。现在，越来越多的人利用自媒体平台传播和获取信息，传播者与被传播者的界限不断消解[张弥弭, 2014]。

微信公众号作为一个自媒体平台，是公众号主持者与订阅者互动、对话的平台，同时也是用户在移动端的信息接入口。区别于微博、微信等平台字数局限、信息琐碎的特点，与微博裂变的传播模式不同，公众号主要是依托信息推送和用户互动的传播形式[黄楚新等, 2015]。公众号的出现，打破了像论坛、博客等基于传统互联网的内容生产和信息传播形式[陆高峰, 2016]。

微信公众号有订阅号、服务号两种类型，分别为个人、媒体、企业等提供服务。个人或媒体可以注册订阅号，向订阅者传递最新的消息。企业可以注册服务号，进行用户管理和服务交互等操作[汤燕, 2016]。公众号主持方与订阅者之间是订阅与被订阅的关系，每个人都可以成为内容的生成者。公众号可以通过用户订阅、用户打赏、允许广告主投放广告等方式获得收益，需要提供优质的内容来吸引用户订阅。公众号图文系统是在公众号端编写文章并推送的基础上，提供给读者阅读环境的系统。准确无误的信息推送，快速便捷的消息打开，丰富多样的阅读形式，友好的交互方式，优质的平台环境都是图文系统需要实现以及提供的。图文系统需要考虑的方面，大到整体消息流的推送形式，小到文章内嵌图片的编码方式。

微信公众平台分为网页端微信公众平台项目和移动端公众号图文系统项目，微信公众号图文系统是本人在微信公众平台部门实习的过程中，参与设计、研发和优化的系统。该项目相对成熟化，在近期进行了改版并做了很多方面的改进，以适应自媒体平台环境的变化。系统中订阅号消息的展示形式从列表形式，改版为消息流形式。以前读者在阅读文章时需要等待进度条加载完毕，页面的打开速度缓慢，现在文章可以在秒级单位内完成。原先系统中仅订阅者与公众号主持方

可见的点赞互动行为，升级为在看，提供更丰富的互动能力。系统将用户在看的文章推送到看一看，用户可与好友交流。公众号图文系统从图文的推送，到图文秒级打开到展示，到提供图文互动能力，每个环节都进行了多重考虑和严格的设计。系统在构建平台环境，实现消息推送机制，提供互动能力，优化阅读体验等方面，对业界具有借鉴和指导意义。

1.2 国内外相关研究现状及分析

“自媒体”的概念最早出现在 2002 年，在 Dan • Gillmor 提出的新闻媒体 3.0 的概念里，3.0 指自媒体，1.0、2.0 分别指旧媒体和新媒体[周晓虹, 2011]。自媒体的载体多种多样，包括微博、微信、今日头条、抖音短视频等。自媒体使信息传播方式从单向传播，转换为一种互播方式。微信公众平台简称公众号作为一个自媒体平台，在 2013 年产生这一概念，距离现在大约 6 年的时间，如今该项目体系已相对成熟，在早期相似的自媒体平台很少。

在国内，近几年有几款类似形式的自媒体平台产品，如百度百家号、今日头条头条号、网易新闻网易号、搜狐媒体搜狐号等。许多“号”形式的自媒体平台的出现，打破了原来只有公众号的垄断地位。百家号、网易号、搜狐号、企鹅号等自媒体平台号的出现，有利于促进内容生产平台的优化完善。

在图文推送的机制上，公众号与其他号平台存在不同。今日头条、搜狐号、百家号等平台号发布资讯的宿主是新闻类的 App，而公众号的宿主是社交聊天类 App。新闻类 App 依靠大数据，根据用户的浏览习惯、关注类型等，分析用户的需求和喜好，向用户推送机器分析认为用户喜欢的类型。微信公众平台是单纯的依赖于社交传播的机制，仅仅在用户关注公众号后才会获得消息推送。系统鼓励用户通过传播优质的内容，使得他人可以在朋友圈、与朋友会话等场景看到公众号的文章，增加文章的传播度。单纯依赖机器的行为，缺少专业的信息审核人员，有时存在不足。有些情况下本来是作者的原创内容，被他人抄袭或者洗稿(通过对原创文章进行同义词变换、语句转换、段落更换等方式来生产内容的行为)先行发布在其他平台上而被认为申明原创的证据不足[陆高峰，2016]。现在是算法时代，通过标题、标签等特性满足了算法的推荐要求，但是不注重生产内容，最后用户还是会流失。因此每个号平台都在努力维护优质的内容生产环境。在建设

阶段，都会用各种方式吸引优质的作者，比如今日头条推出“千人万元计划”，公众平台的作者赞赏机制等[刘胜男, 2017]。有些平台对每日推送的文章数没有限制，在公众平台中一般账号每日最多群发一条文章，经过认证的账号每日可以群发多条，这种方式也有利于公众号选出更加优质的内容发布。图文展示方面，每个平台都支持在文章中插入图片、视频等媒介，微信公众号图文系统中还提供小程序、卡券等与微信相关媒介在文章中嵌入，同样每个平台也都会嵌入与自身特性相关的媒介。每个平台除了提供展示文章功能，还需要提供互动能力，基本都提供点赞、评价、分享等互动功能。

国外类似微信公众号的有 Facebook 的 Instant Articles。Instant Articles 被定义为提供给内容创建者在 Facebook 上创建快速、交互式的文章，对用户而言实际上是内容整合的平台。由于人们喜欢在 Facebook 中分享文章，但是网页的打开速度比较慢，因此推出了即时文章项目。与其他国内的自媒体平台相比，Instant Articles 与微信公众号更加相似，因为 Instant Articles 也是基于 Facebook 的社交平台，但是 Instant Articles 与国内的自媒体平台有几点不同。第一：在内容发布者的注册条件上，头条号和公众号等任何企业、媒体、甚至个体都可以注册，头条号甚至只需手机号即可注册，门槛较低。Instant Articles 虽然没有明确的规定，但是合作的基本上是拥有良好声誉的媒体。第二：在 Instant Articles 中创建的文章都为在平台内直接创建文章，不存在转载、第三方链接等形式。第三：公开透明化的广告收益分配机制[宋黎等, 2018]。Instant Articles 项目集成于 Facebook 应用内，将发布者发布的文章直接存储在 Facebook 的应用中。当有用户点击链接时，不会引导到网站上，而是直接链接到 Instant Articles 的页面中，缩短了文章的加载时间和下载网页内容的时间。因为在通常情况下，当用户在 Facebook 中点击一个文章的时候，无论如何优化网站都需要等待加载，才能读到想要的内容，这种方式提升了用户的阅读体验。除了体验上的优化，Instant Artciles 还提供一些互动功能，比如倾斜手机查看高清图片、滚屏时自动播放视频、提供点赞和评论等。同时 Facebook 也提供专属的 App，内容发布者可查看最新的阅读量、访问量、转发量等数据，可以基于数据对自身发布的内容和接收信息的群体进行调整。

国内外都有类似于公众号的自媒体平台，但是由于依托的产品特性和传播的理念有差异，使得每个平台在文章推送机制、文章展示方式、互动形式上都有所区分，并且在成熟度、受众和使用人数上有明显的区别。总而言之，无论是像今日头条、百度 app 这种基于数据内容推荐的产品，还是像微信公众号、百家号等依赖于内容分发渠道的产品，争夺的核心都是优质的内容。对于无论是已经成熟的自媒体平台，还是新兴的自媒体平台而言，既要培养优质的内容生产者，打造高质量的媒介产品，同时还要洞察用户的心理，通过数据研究用户的行为，对文章媒介产品多种层级的开发，满足不同类型用户的需求。自媒体平台不仅需要研究新的技术，应用于内容生成环境和内容展现环境，也要坚守自己的品牌，形成自己的品牌优势[黄馨茹, 2017]。

1.3 本文的主要工作

本文主要描述微信公众号图文系统的设计与实现，公众号图文系统主要目的是提供给用户良好的阅读环境，在每个环节都做到提供给用户最佳的体验。当读者关注了公众号成为该公众号的订阅者后，将会收到公众号方群发或者定时群发的消息。图文系统的推送功能及时将消息推送给订阅者，将图文消息以时间流的形式展示。在用户点击阅读文章时，文章做到在秒级单位内展现在用户面前，无需用户等待，区别于以往用户需要等待进度条加载完毕的场景。图文系统支持以图片、视频、音频、小程序卡片、卡券、投票等丰富的媒介形式展示文章，并且在为了节省带宽和提高阅读时图片的清晰度，改变图文编码方式并进行图像超分辨率重构。图文系统在保证用户信息安全的基础上，在获得用户微信信息后，提供评论、点赞、分享等互动能力。系统同时兼容无登录态的场景，可脱离宿主微信 App，支持在浏览器中阅读。在为了提供优质的内容生产环境，公众平台鼓励原创，抵制抄袭、洗稿等违法违规行为，在文章发布层阻止违规文章的推送，在文章阅读层提供给用户投诉违规行为的渠道。同时公众号图文系统也提供给订阅者便捷的方式对公众号关注和取关。

总结而言，本文主要描述公众号图文系统的设计与实现，包括图片、视频、留言、点赞、投诉等功能。系统通过安全性、可交互时间、首屏时间、交互友好性等非功能性需求的实现，满足订阅者的阅读需求和提升读者的阅读体验。

1.4 本文的组织结构

作为一个自媒体平台，本文将主要描述微信公众号图文系统是如何设计和实现的，并且说明如何通过重新设计和优化来提升用户体验以吸引更多的用户和构建优质的内容产出平台的。

本文的组织结构如下：

第一章 引言部分。对微信公众号图文系统项目的背景和意义、国内外类似的自媒体平台的发展和图文系统的功能做了简单的介绍。

第二章 技术概述。描述微信公众号图文系统中所参考的和用到的技术。

第三章 图文系统需求分析和总体设计。分析并说明微信公众号图文系统的功能性需求，并以用例图的形式列举说明，同时分析并描述图文系统的非功能性需求。对项目进行了概要设计，用架构图的形式展示图文系统的整体架构，并且对主要的模块进行说明。

第四章 公众号图文系统实现。描述微信公众号图文系统中图文推送、图文展示、图文互动、图文秒开模块的具体实现，以代码和算法图的形式说明关键部分的实现。

第五章 总结和展望。总结论文期间所做的工作，并对微信公众号图文系统的未来发展做进一步展望。

第二章　技术综述

2.1 Hybrid App

App(mobile application)称作移动应用程序，常见的 App 大致可以分为三类：Web App、Native App、Hybrid App。Web 移动应用是指使用 Html5 编写的，不需要下载安装，寄生在浏览器中的应用。Web App 的开发与维护成本低，更新也无需通知用户手动升级，能够跨多个平台与终端，但是无法获取系统级别的通知、提醒，设计方面会受到网络环境、系统级别权限问题、浏览器的兼容性等方面的限制，体验较差。而 Native App 即指原生 App，是一种基于智能手机本地操作系统如 iOS、Andriod，使用原生方式编写的应用程序。这种应用程序一般依赖于操作系统，有很强的交互能力，用户体验较好。与 Web App 相比，Native App 开发成本较高，后期的维护也比较困难，但是能够获取到手机底层的功能，如：打开摄像头、获取相册、获取地理位置等。

随着移动浪潮的兴起，移动应用开发的需求量激增，业务扩展比较迅速，对开发效率要求较高。使用 iOS 或者 Andriod 开发 App 成本过高，而 HTML5 的低成本、高效率、跨平台的特性可以与客户端结合，形成一种新的开发模式叫 Hybrid App[keep789, 2018]。这是一种混合的、半 Native 半 Web 的开发模式，既具有原生应用的优点比如好的用户体验，也具有 Web 应用可以跨平台开发的优势[周齐飞, 2014]。在许多第三方框架的帮助下，Hybrid App 可以像原生应用一样访问本地的资源[sun et al., 2016]。混合式应用底层依赖于原生 App 开发提供的 webview 组件，在 iOS7 以下有 UIWebview，7 以上有 WKWebview，Andriod 中是 webview，这个组件可以加载 HTML 文件。混合式应用的上层部分使用 HTML&CSS&JavaScript 来做业务开发，底层透明化，上层多样化，这种方式可以满足业务快速迭代的要求。目前这种开发模式越来越火，很大的原因是可以进行“热更新”，不受限于移动应用的审核机制，并且上线过程更加简单。微信公众号图文系统也是一种混合式的应用，底层依赖于微信提供的 webview，大部分业务逻辑和功能由 Html5 页面实现，分享到朋友圈、发送给朋友等与微信应用相

关功能由客户端实现。图文系统的业务能够快速迭代，不用跟随微信版本审核，能够随时快速上线功能与修复问题。

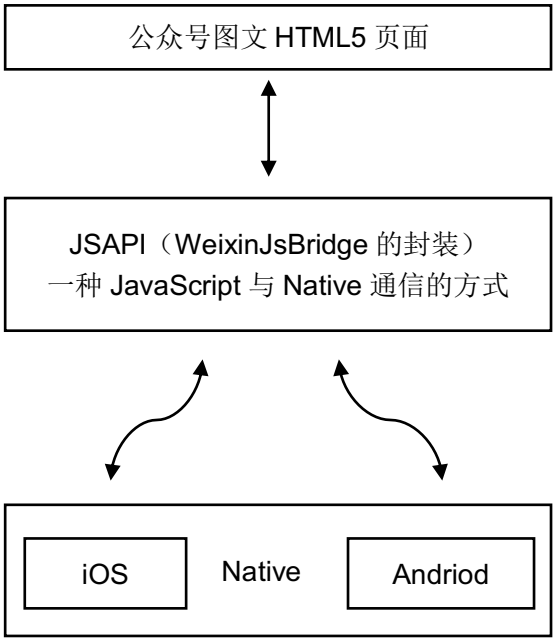

图 2.1 公众号图文系统 Hybrid 架构

如图 2.1 所示，图中描述了微信公众号图文系统中 H5 页面和底层 Native 是如何交互的。H5 页面通过微信客户端提供的 WeixinJsBridge 获得客户端的原生功能，比如发送给朋友、在 safari 中打开等，大部分客户端提供的功能都隐藏在公众号文章右上角的点点点按钮中。WeixinJsBridge 根据名称可理解为桥梁的意思，是连接 JS 与 Native 之间的桥梁，促使能够进行双向通信。Javascript 向 Native 发送信息，是通过 API 注入或者 URL 拦截的方式，进行功能调用。Native 向 Javascript 发送消息，可以通过直接执行拼接好的 Javascript 代码，通知调用的结果和进行消息推送。WeixinJsBridge 对外开放了许多公共的 API，一些为业务特性而开发的 API 需要针对特定的域名添加白名单才能使用。图文系统中将 WeixinJsBridge 封装成 JSAPI，在 JSAPI 中增加与业务相关的特性，比如：上报 WeixinJsBridge 调用出错信息等。

2.2 SeaJS

随着 javascript 在 web 应用中扮演着越来越重要的角色，但在浏览器中没有指定检测脚本间的依赖关系的方式。比如在加载页面上<script>标签时，按顺

序进行评估、加载和执行，会首先需要加载公共库和函数，然后再加载实际执行脚本[Amber, 2015]。为了降低系统之间的耦合度，减少资源的循环依赖，增加系统的内聚性，诞生了模块化的概念。每个模块之间独立存在，内部可以独立完成子功能。模块化管理通过依赖加载机制，将小文件组成大文件，为整个项目服务。

SeaJS 是符合 CMD 规范的 javascript 模块加载框架，初衷是帮助前端开发者从繁重的 javascript 文件中解放。CMD 规范称为通用模块定义，描述如何定义模块以便在浏览器环境中交互，定义模块需要有的最小特征。SeaJS 将大文件拆分成小文件，保持小颗粒度的模块化开发，同时不需要考虑依赖关系，可以实现模块化的开发和加载机制，按照先后不同的依赖关系对 js 文件进行加载，可以理解为 js 文件加载器[adeyi, 2013]。SeaJS 可以确保在加载完所有依赖的文件后再加载当前文件，在有大量的 javascript 文件的项目中可以确保文件的加载顺序，能够避免出现加载快的文件依赖于加载慢的文件而出现由于依赖的某些库或者函数没有加载完成，导致的浏览器报错问题[古兰今, 2017]。SeaJS 提供一些公共的 api，如：define 函数、require 函数等，用于模块定义、处理模块间的依赖。SeaJS 本身是用 javascript 实现的框架，并没有改写 javascript 语言，能够与其他框架如 jQuery 等共同使用。这种模块化管理框架可以提高前端代码的可读性，解决在 javascript 领域文件依赖关系混乱和代码纠缠等问题，有利于代码的编写和维护[张洋, 2013]。

在公众号图文系统中使用 SeaJS 框架进行模块化管理，没有使用另一款框架 RequireJS。RequireJS 框架是符合 AMD 规范的，AMD 规范称为异步模块定义，异步加载模块以及其依赖项。在外部表现上，RequireJS 与 SeaJS 都既可以运行在浏览器端，又可以运行在服务端作 node 等环境的模块加载器，但 SeaJS 提供的 api 更加简洁易用。在社区维护上，SeaJS 采用自我封装的形式，支持与其他类库协作，而 RequireJS 尝试让第三方类库修改来支持，因此在社区维护上，SeaJS 更好。SeaJS 的插件机制和 Node 一致，支持让插件开发者访问和修改，十分灵活，而 RequireJS 是在源码内部预留接口的形式[玉伯, 2012]。在内部的实现机制上，两者的不同在于：

1. 模块加载后执行时间。SeaJS 是加载后不执行，按需执行真正用到的代码，顶层的业务逻辑加载完到最终的子模块加载后再回调顶层的业务逻

辑。RequireJS 是加载后马上执行，绑定顶层模块的定义事件，等待底层依赖模块回溯该事件到顶层来触发业务逻辑。

2. 分析依赖耗时。在不用构建工具时，RequireJS 的性能要高于 SeaJS。
3. 执行中内存释放。RequireJS 在回溯的过程中会产生许多模块实例的闭包，直到底层的依赖回溯到上层后才释放，SeaJS 在模块加载完成后就删除生成的实例，从这个层面上看 SeaJS 更适合复杂的应用。

在对比了 SeaJS 和 RequireJS 的 api 简易性，可扩展性等要素，并且考虑到图文系统项目的复杂性和构建过程，最终选择使用 SeaJS 作为模块化管理工具。

2.3 QQMail 模板引擎

2.3.1 模板引擎

随着 web 应用开发的发展，产生了一种设计模式 MVC 模式。MVC 分别代表 model、view、controller，表示数据层、表现层和控制层。这种模式将逻辑层与表现层分离，提升了 web 应用开发的效率。Web 设计中应当遵守的原则是数据层与表现层分离，数据层是原始、相对固定、与表现无关的，而表现层多变的，决定数据的外在表现。

有一个公式，数据+模板=表现，中间加号代表转换引擎[姬一文等, 2011]。模板引擎的作用是为了将显示与数据分离，模板文件与模板数据由模板引擎处理生成最终的 HTML 页面。模板引擎需要在模板系统的上下文中执行，对模板数据进行处理、过滤，对模板文件进行处理，是模板的驱动器。设计者在建立出模板的基础上，对模板表示的行为进行解释。设计时，在选择适合语言的基础上，设计模板编译器和模板解释器，模板编译器又分为词法分析器和语法分析器[刘佳等, 2006]。模板引擎也是一种微型语言，需要有条件判断、变量赋值、循环等语言特性和其他功能，使得在模板系统中可以方便地修改数据层的外在表现，避免在业务逻辑层做适配。模板引擎的语法在使用上简单易学，编写者需要在模板页面中嵌入语法表达式。模板引擎在 HTML 文件中创建出占位符，处理数据并

进行替换，最终返回带有数据的页面。返回的页面可以直接展示在浏览器中，不需要另外做处理。

模板引擎是基于后台语言实现的，如基于 PHP 的 Smarty、FastTemplate 模板引擎，用 java 实现的 FreeMaker 等，每种模板引擎的实现方式都有所不同。腾讯在 2010 年时推出了 QQMail 模板引擎，该引擎是用 c++语言编写。相对于 javascript 处理速度更快，可以在 windows 和 linux 下编译使用。国内的各大公司都有自己的模板引擎，如阿里的 kissy template，百度的 baidu template 等，QQMail 模板与业界其他模板引擎从渲染速度上、安全性、大小、错误处理、调试等方面相比毫不逊色，可以应用于各类应用的开发[itprobile, 2014]。

2.3.2 QQMail 引擎处理流程

表格中的数据可以以柱状图的形式展示，也可以以饼状图的形式展示。类似于表格中的数据被转换成图表的过程，模板引擎也是做相似的工作，将编写的 HTML 模板与后台接口返回的 XML 数据结合成最终的 HTML 页面。

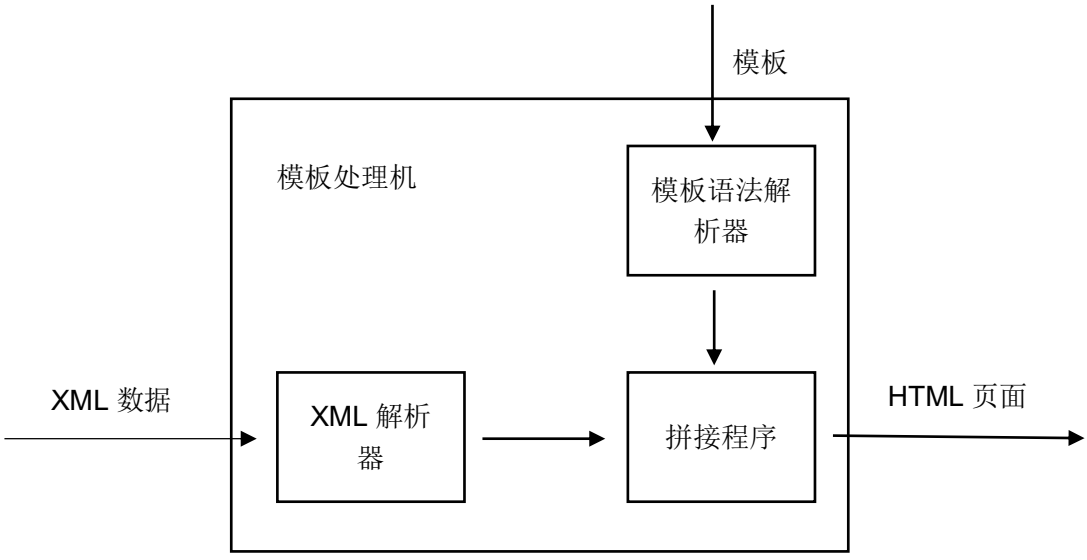

图 2.2 QQMail 模板引擎处理流程

如图 2.2 所示，后台进行逻辑处理后，以 XML 的形式返回数据，模板处理机内部的 XML 解析器根据节点标签对 XML 数据进行解析。模板语法解析器对前端生成的模板进行语法解析和词法解析，拼接程序根据模板的语法表达式，对 XML 内的数据运算，进行替换返回最终的 HTML 页面。

QQMail 模板引擎的语法是<%%>的形式，当解析器解析到<%%>标志时进行替换。同时在<%%>内也支持插入 javascript 函数，支持变量替换，比如$name$形式，将 XML 数据中 name 标签对应的值替换显示出来。支持节点的层次选择，由于 XML 的文档结构也是一颗树，解析器可以获得父节点的子节点，比如$parent.child1.name$的形式，即获得父节点的第一个孩子节点的名字。支持 while 循环，以<%@while(...)%>…<%@endwhile%>的形式。支持条件判断，用<%@if%><%@elseif%><%@endif%>的形式包裹函数体，除此之外，还支持模板函数、include 语法等。对于复杂数据和模板而言，比如 32K 的数据和 90K 的模板，模板解析小于 1ms，XML 解析也小于 1ms，拼接大约需要 22ms，在性能上 QQMail 模板引擎是非常优秀的。QQMail 模板引擎的学习成本也比较低，对于后台人员而言，直接输出 XML，无需学习。对于前端人员而言，只需学习模板语法，在有 XML 知识的基础上，可快速掌握[李明强, 2010]。

2.4 Svrkit 框架

RPC(Remote Procedure Call)服务架构风格将服务器看作一些过程的组成，客户端调用过程来完成任务的执行。服务以细粒度的方式定义，每个服务有不同的接口，每个接口有自己的语义和专有的参数，调用方需了解接口的定义和数据格式等[冯新扬等, 2010]。Google 的 Protobuf Buffers 是用于结构化和序列化数据的技术，用于网络间的传输。是一种高效可扩展的数据结构，相比 XML 数据结构而言，具有更少的歧义、更快、更方便的数据存取、节省空间等特点。在发送数据时，系统需要对发送的信息进行序列化操作，在接收数据时，系统需要进行反序列化和解包操作[李纪欣等, 2013]。

Svrkit 框架是用微信内部用 C++编写的高性能的 RPC 服务框架，数据传输基于 Protobuf 格式。最初由广州研发部研发出来，后来逐步被广泛应用在微信相关项目中，微信公众号图文系统也是使用该框架进行前后端管理。Svrkit 框架请求处理的过程和 MVC 模式类似，与 MVC 模式类比，CGI 起的是 Controller 的作用。CGI 接口作为程序逻辑入口，接收逻辑请求，调用 Svrkit 服务端来处理对应的业务逻辑请求，并且返回处理结果。Svrkit 框架有很多个 CGI 接口，通过配置使得每个请求通过一对一映射的 url 找到对应的 CGI 进行处理。Server 像

MVC 模式中的 Model 层，负责逻辑的具体实现还有数据库操作[jieniyimiao, 2016]。

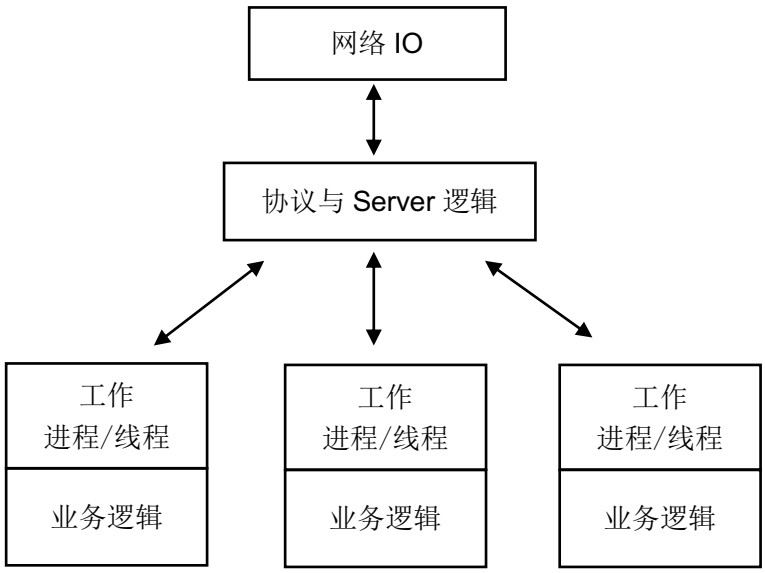

图 2.3 Svrkit 框架模型

如上图 2.3 所示，Svrkit 处理框架使用多进程或多线程的模型。主进程通过 Fork 创建进程池，进程池动态调整进程数，接受进程加入事件驱动。进程池轮询抢占管道，通过信号量抢占互斥，有数据则进行处理，没有数据时则进行空转。该框架涉及许多技术，包括进程间通信、连接管理、事件管理、进程模型等，代码很复杂并且庞大。模块较多，实现的功能特性较多，包括负载保护、快速拒绝等。RPC 服务架构允许像调用本地服务一样调用远程的服务，使得调用远程服务更加简单。开发人员不需要了解通信的机制和底层的实现细节，只需要以定义的方式提供接口和调用接口即可[ChrisMurphy, 2017]。

2.5 超分辨率重建算法

高分辨率的图像是指图像的像素密度较高，能够看出细节信息，对场景的描述更加准确。随着科技发展，高分辨率的图像在各个领域需求都特别大，比如：视频监控、医疗数字图像、卫星图像领域等[张明, 2010]。为了获得高分辨率的图像，仅仅依赖如芯片、光学部件等硬件是不可行的。不仅由于硬件的成本会非常高，而且不断增长的更高分辨率的需求，会最终超过光学部件的极限。近些年，对于低质量图像重建技术获得许多人的关注和研究。超分辨率重建是指将低分辨

率的图通过图像处理方法，重建出高分辨率图像的技术。能够在光学不见的极限范围之上获得超分辨率的图像，并且对比而言实现的成本要低很多[江静等, 2012]。

超分辨率重建算法是在低分辨率图片的基础上，还原图片的清晰度，重建出高分辨率的图片。传统的超分辨率算法需要基于大量的同一场景、有相同亚像素值的低分辨率的图片。基于假设或者关于从高分辨率图像到低分辨率图像生成模型的先验知识，生成一系列的低质量的图片，但是许多信息会在从高到低生成图片的过程中丢失[Yang et al., 2008]。目前超分算法分为两大类，基于学习的和基于重建约束的。基于学习的超分算法是通过在从高分辨率的图片到低分辨率图文生成过程中，建立起一个共生模型的基础上，为低分辨率的图片增加高频的细节。基于重建约束的超分算法，要求重建出的图像要像原先的图片，为了获得更好的结果，对重建好的高分辨图片应用好的先验或者约束[廖秀秀等, 2012]。

类似于计算机视觉其他领域比如物体分类、检测等方法，超分算法也经历了手工的设计阶段，主要是插值算法。插值算法是利用数学函数，用图像中已知的若干个点，使用计算得到的函数值作为边缘的值。中层表达的学习阶段，主要是字典学习方法。现在是端到端的阶段，使用深度学习方法。插值超分算法主要有几类，传统的插值算法、基于边缘的图像插值算法、基于区域的图像插值算法。传统插值算法比较简单，容易实现，但是缺点是容易产生锯齿和马赛克，图像的细节会变得模糊。字典学习的超分算法，是利用低分辨率的图片和高分辨率图片之间的映射关系，这种方式也有很快的速度，但是依然会有清晰度的问题[程衍华, 2018]，深度学习的超分辨率算法是目前最热门也是性能最高的超分辨率算法。

单图像超分技术指在基于单个低分辨率的图像生成高分辨率的图像[Kim et al. ,2015]，利用卷积神经网络 CNN 的学习能力，低质量的图片作为输入，基于低质量与高质量图片的对应关系，深度卷积神经网络处理后，直接输出高质量的图片。不同于传统的超分算法，每个部分分块处理，将所有层一起优化[Dong et al., 2014]。深度学习超分算法可以分为几类：CNN 网络、基于残差结构的跨层连接、基于密集连接跨层连接、基于残差结构和密集连接的跨层连接，每种模型都有一定的适合场景，但是也有相应的缺点，在速度、收敛难易度、显存耗能等

方面都会有所区别[Tai et al., 2017]。许多研究提出了深层次的神经网络进行超分，但是一般图像超分辨率重建有很高的计算复杂度，针对大尺寸的图片而言，算法的处理流程需要处理上亿级的像素数据，越深层次需要的时间可能会越长。在微信公众平台中每天有几百万的图片上传量，但是由于带宽的限制图片通常会被压缩，或者上传的图片经过多次转发等因素，会造成用户看到的图片模糊不清。为了用户的阅读体验，需要还原图片的清晰度，并且不能造成带宽的浪费，因此在公众平台中应用深度学习的超分算法，使读者在阅读时看到清晰的图片。对于像微信公众平台之类的项目，每日需要处理的图片达到上百万，算法处理的效率也非常重要。如何在超分效果和性能之间找到一个平衡点，是图文系统中超分算法的关键[刘琚等, 2009]。

2.6 图片压缩格式

2.6.1 WebP 格式

WebP 是由谷歌提出的一种新的图片压缩格式，和目前主流的图片格式如 PNG、JPG 相比，在同等图片质量的前提下，WebP 的大小要小很多[Si et al., 2016]。它来自于视频 V8 的编码格式，拥有很高的压缩比和质量。V8 采用虚拟的参考帧高级预计编码、基于宏块级的多线程策略、增加复杂度的上下文编码、自适应的回路滤波等技术，以较小的数据获得较高质量的视频[Lian et al., 2012]。WebP 同时提供有损压缩和无损压缩两种格式，无损压缩后 WebP 比 PNG 小了约 45%的图片大小，即使图片是经过第三方工具压缩后，WebP 仍可以减少约 28%的图片大小[又拍云, 2017a]。浏览器端，Chrome、Opera 等浏览器支持 WebP，并不是所有浏览器都支持。包括谷歌在内的许多网站，为了节省网络带宽等资源，都将网站上的图片批量替换成 WebP 格式的图片[阿立, 2016]。

WebP 有损压缩通过将图片分成 8*8 和 16*16 的的两个色度像素宏块，每个宏块内，编码处理器基于之前的宏块，预测颜色信息和冗余动作。通过关键帧技术，对宏块中已解码的像素进行关键帧运算，绘制图像中未知的部分，去除冗余的像素信息，实现高效的压缩。WebP 的无损压缩采用颜色变换、预测变换、减去绿色变换等技术对图片进行处理，对变换后的图像的数据和编码进行熵编码。

颜色变换是去除每个像素的 R、G、B 的值，彩色变换时先保持绿色值，根据绿色值变化红色值，再根据绿色值转换蓝色值，最后根据红色的值进行转换。预测变换是通过临近像素的相关性等数据减少熵，有向左、上、右等像素预测模式，还有平均值预测模式[又拍云, 2017b]。

WebP 格式的图片能够在保障图片质量的前提下，图片的大小比一般格式的图片小很多。对于需要展示较多图片并对图片的质量有要求的项目而言，是一种节省带宽的选择。

2.6.2 HEVC 编码

视频压缩是实现视频服务的前提，通过编码器对视频进行压缩，降低存储和传输的成本，压缩后的视频通过解码器转换成原始的视频[王建富, 2015]。随着人们对观看高清视频的需求的增长，对高分辨率、高帧率和高压缩率的要求越来越高，原先的视频编码 H.264/AVC 标准已不能满足需求。推出了在相似视频质量的条件下，比原来 H.264/AVC 标准能节省 50%左右码率的 HEVC 标准[徐荣飞, 2013]。HEVC 是 High Efficiency Video Coding 的缩写，即高效率的视频编码。在 2013 年发布的 HEVC 第一个版本，成为 ITU-T 视频编码专家组和 ISO/IEC 运动图像专家组的最新的视频编码标准。HEVC 标准也在不断扩展，适应于其他应用场景，如：提升精度和颜色格式化处理、可扩展的视频编码、3D 等场景视频编[Sullivan et al., 2013]。

HEVC 编码技术可以说是对 H.264/AVC 技术的升级，包括帧内预测、帧间预测、转换、量化、熵编码等模块。HEVC 将这些模块整合成 3 个不同的单元，分别为编码单元、预测单元和转换单元。HEVC 编码单元更为灵活，最小可以 8*8 像素，最大可以 64*64 像素，与 H.264/AVC 只可以为 16*16 像素不同。该技术能够大幅度的提高视频的压缩比，能够支持 4k/8k 的高清视频的播放。能够有效的减少带宽，为用户在观看视频或浏览图片时节省网络流量[徐振, 2016]。

HEIF 是 High Efficiency Image File Format 的简称，HEIF 格式的图片文件是以 HEVC 为编码器实现，和 GIF 格式相比，有更小的文件大小，并且拥有更多颜色数据。和 JPEG 格式图片相比，HEIF 在图像压缩后不会在边缘出现瑕疵，

处理上更加平滑，并且压缩比是原来的 2 倍左右。iOS11 采用 HEIF 格式取代原来的 JPG 格式，提升图片的质量的同时也减少存储占用的空间大小。

2.7 React Fiber 架构

浏览器的渲染引擎是一种单线程的方式，解析和渲染 DOM Tree 和 CSS Tree。由于这种一次只能做一件事的单线程方式，当遇到复杂的 DOM 操作页面时就会出现卡顿、崩溃的情况。页面上 DOM 的操作会引起页面的重绘或回流，然而复杂的网站不可避免会有很多用户交互，因此需要考虑如何降低 DOM 操作引起的性能损耗。

React 是近几年比较火一款前端框架，第一个提出了 Virtual DOM 概念。React 改变了前端开发的方式，改变了大家对前端框架的认识，可以与其他框架、类库搭配使用[卓越开发者联盟, 2015]。当页面的元素发生变化时，React 会构建一个新的虚拟 DOM，进行新旧 DOM 对比，通知浏览器需要更新的元素。浏览器会计算 DOM Tree，重新绘制页面[刘杰凤, 2017]。React 的更新过程包括两个阶段，调度阶段和渲染阶段。调度阶段是用更新后的数据生成新的虚拟 DOM，遍历虚拟 DOM，和原来的 DOM 进行对比，找出需要更新的元素，放入更新队列中。渲染阶段是通过遍历更新队列，将对应的元素更新[傅崇琛, 2018]。然而在遇到复杂的操作或者频繁更新数据时，React 会有性能问题。React 核心的调度策略为递归遍历，从顶层组件开始，一直到底层的子组件，之后再回溯到父组件，这种组件渲染方式在对于层次很深的组件而言，需要更长的时间。在更新组件的时间里，用户的行为得不到反馈，包括点击、鼠标移动等操作，处理用户事件需要等到 React 执行完后才会回调。

为了解决 React 的渲染性能问题，React 16 中提出了 React Fiber 架构。Fiber 架构将组件的更新分为两个时期，绘制前的生命周期与绘制后的生命周期。绘制前的生命周期可以被打断，每隔一段时间都会确认是否有其他任务。绘制后的生命周期不可以被打断，将一次性应用所有的 DOM 变更。Fiber 架构将原先的单线程任务分为单个的小任务，并且每个任务有各自的优先级。优先级低的任务可以被优先级高的任务阻断，用户的输入响应的任务优先级大于 DOM 更新的渲染任务，因此用户的输入能够得到反馈。该调用机制下能够暂停及恢复任务，

分配给任务不同的优先级，可以重复使用之前已经完成的任务或者终止不再需要的任务[sebmarkbage, 2016]。Fiber 架构只是内部的调度和渲染机制改变了，提供给用户的 API 并没有发生改变，因此已有的项目不需要做变更。

图文系统中借鉴 React Fiber 分割任务的调用机制和将原先的大任务分成一个一个小任务的理念，在 HTML 中渲染 DOM 结构时，分段执行。先解析渲染首屏节点，执行 javascript 函数，在闲时继续渲染其他节点，实现在微信公众号图文系统的首屏渲染机制。

2.8 本章小结

本章主要介绍了微信公众号图文系统中使用和参考的技术。首先介绍了图文系统的混合式应用开发模式，之后介绍了前端模块化管理所用到的 SeaJS 框架、后台直出的模板引擎 QQMail 和 Svrkit 后台服务框架，然后介绍了图文系统中对图片处理相关的超分辨率重建技术和 WebP、HEVC 图片编码方式，最后介绍了在图文系统中实现文章秒级打开所参考的 React Fiber 架构。

第三章　图文系统需求分析与总体设计

3.1 微信公众号图文系统概述

微信公众平台作为一个自媒体平台，网页端公众平台系统为每个用户提供适合的注册账号类型，是为公众号主持方提供发布文章、管理订阅者的平台。公众号可以通过提供自动回复、自定义菜单等功能与用户互动，通过开启赞赏、允许插入广告等方式获得收益，通过图表等形式查看每篇文章的发布效果。微信公众号图文系统是为订阅者提供服务的系统，用户订阅公众号后，成为公众号的订阅者。订阅者可以通过便捷的方式添加关注、取消关注并且可以查看所有关注的公众号。公众号通过公众平台向订阅者推送消息，当订阅者在群发对象列表中并且没有拒绝接收消息时，将会收到公众号发布的最新消息。用户会按照时间的先后顺序接收到公众号推送的文章，最新的文章显示在最上，可以不断下滑以查看历史推送的消息。为了给用户提供最佳的阅读体验，用户可以在秒级单位内打开并阅读。图文系统为公众号发布的文章提供丰富的媒介展示形式，包括图片、文字、视频、音频、小程序、投票、卡券、商品、代码块等。图文系统页面以统一的样式风格，并且提供优质的阅读体验，同时还提供一种新型的社交方式。阅读不再是一个人的阅读，当阅读文章时，读者可以留下个人的阅读感受，或者与他人分享。同时用户可以通过收藏等方式，供以后去阅读。图文系统作为自媒体平台，需要维护优质的内容生产环境，提供优质文章写作氛围。公众平台鼓励原创文章，抵制抄袭，并且严厉打击单纯的抄袭和换种方式的洗稿等违规行为。当发现违规行为时，系统对公众号进行发文封禁原创封禁等操作，同时提供给文章作者或者其他人员投诉举报的渠道。公众号图文系统是一个较完备的自媒体文章推送、展示、提供互动的平台，既可以在移动端微信应用内使用，也可以在任一浏览器中打开，但是功能上会有所区别。

微信公众号图文系统并不是一个独立的系统，微信公众平台有订阅号、服务号、小程序三种类型账号。除了公众号图文系统密切相关的微信公众平台系统，还有订阅号助手 App，赞赏助手小程序，公众号数据助手小程序等。订阅号助手 App，是一款移动应用，帮助订阅号管理已发表的文章，新建文章，管理素材。

公众号管理者及运营者能够及时看到用户的留言和赞赏，比起网页端更方便。赞赏助手小程序，是为了作者而开发的小程序。往往一个作者可能为多个公众号写稿，使得作者能够更好的管理收益，这种赞赏方式能够鼓励作者写出更加优秀的原创的文章。内部的公众号数据助手小程序，以图表形式展示公众号相关数据。内部人员通过观察数据变化，对公众号图文系统进行调整和优化。

3.2 公众号图文系统需求分析

3.2.1 图文系统功能性需求

新闻类的消息一般都是即时性的。若系统延迟推送消息，信息就成为了过期消息。因此无论是哪种类型的消息，公众号图文系统都需要向订阅者及时准确的推送公众号发布的最新消息。推送方式按照时间线的顺序，最新的消息显示在最前。用户可以不断下滑查看以往收到的推送消息，并且可以查看公众号推送的所有历史消息。公众号需要有自己的主页,主页是公众号与订阅者最直接的沟通渠道，提供自定义回复的方式与订阅者进行交流。公众号可以通过自定义菜单更好的管理公众号功能，如提供热点事件菜单、合作版块等菜单。公众号还需要有小主页，相当于个人的名片。小主页中展示公众号的以往发布的热点文章，用户可以将小主页分享给朋友，让朋友关注该公众号。

在自媒体行业，一篇文章最基本的需求是展示文字，对于不同的用户，文字的大小有不同的要求。文章发布者即公众号的作者在编辑文章时，可以在编辑器中设置字体的字号、格式、粗细等。图文系统需要支持在手机端阅读时可以调整字体的大小，满足不同用户的阅读需求。图文系统除了显示文字，还需要显示图片。发布者编辑文章时可以插入图片，当订阅者在手机端阅读文章中的图片，需要能够清晰无延迟的显示。文章中的图片还可以点击放大查看，读者可以将图片保存到手机、收藏、发送给朋友等操作。在报刊杂志时代，只需要有文字和图片就足以满足一篇文章的展示需求，而在多媒体时代除了文字和图片之外，还需要支持视频、音频等媒介。系统支持视频播放、暂停等基本操作，对于纯视频分享的文章需要支持打开后自动播放。因为公众号图文系统是嵌入在微信应用内的系统，还需要支持微信特有的媒体插入，比如：小程序、卡券等。在图文系统中可

以插入小程序的卡片，当点击小程序卡片时可以自动调起微信内的小程序。在图文系统中插入的卡券，可以保存至微信中。除了文字、图片、视频、小程序、卡券等，文章中还可以显示外部商品、广告、投票等。是否允许在发表的文章中嵌入广告主的广告对于公众号而言是一个自选项，在微信公众平台网页端登陆公众平台账号后可以选择打开或关闭。广告的形式有许多种，顶部广告、文中广告和底部广告。广告是由广告主投放，图文系统的广告处理逻辑将广告嵌入在公众号的文章中。为了更高的广告收益，使得广告能有更大的曝光量，当用户滑动到广告时，系统会自动播放广告。每个用户看到的广告数量是有限的，并且所有广告都是经过微信公众平台严格筛选的。图文内需要显示的内容如图 3.1 所示：

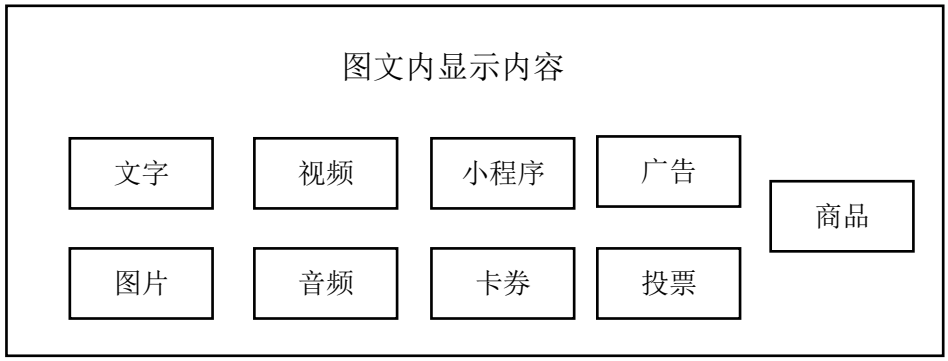

图 3.1 图文内显示内容

文章中除了需要展示标题、公众号名称这些基本信息，还需要标识文章是否是原创。当为原创文时，标识该文章的作者；若是转载文，显示转载文章的推荐语，这些推荐语可在公众平台浏览器端转载文章时编辑输入。

以上是图文系统的展示内容需求，单纯个人阅读文章可能只需要展示文章内容，然而在阅读文章时还需要与他人交流，分享喜欢的内容或者收藏以供下次阅读。在微信中系统能够支持发送至朋友，分享至朋友圈、QQ、腾讯微博、QQ 空间，在微信读书中阅读，在 safari 中打开等操作，并且可以通过复制链接直接发送给别人。微信公众号图文系统通过以上的途径增加文章的传播度，没有设计推荐算法向用户推荐认为用户喜欢的文章。

微信公众号图文系统以微信作为宿主系统，需要获得微信用户的信息。在获得用户登录信息的情况下，向用户提供互动能力。系统根据用户 id 记录文章的阅读数，订阅者可以对喜欢的文章进行点赞并且可以对文章留言。为了用户的体验和防止恶意或者不符合规范的留言，只有被公众号采纳的留言才会显示在留言列表中，留言的条数也有限制。公众号可以选择某些留言将其置顶，有时候优质

的点评会带给用户更好的体验感。在之前的需求中文章支持点赞，点赞行为是对公众号文章的肯定，但是对于其他用户是不可见的，只有用户自己知道是否对文章点赞。为了给微信看一看的引流，并且使微信内的好友之间有更强的互动，公众号推出在看功能。用户可以通过点击在看，并且添加在看评论，在微信发现中看一看应用里即可以看到自己和朋友在看的文章。用户可以对朋友在看的文章点赞和评论，使以前个人的点赞行为有更强的互动性。订阅者可以对自己喜欢的作者写的特别好的文章进行赞赏，给作者更多写文章的动力。同时订阅者可以将有趣、在看等文章分享给朋友或者分享到其他平台，交流自己的看法。当读者阅读到不符合规范的文章比如抄袭、诱导分享等，可以进行投诉并佐以证据，帮助微信公众平台构建更好的写作氛围。系统会给支持插入广告的公众号一定的收益，系统中广告的形式又分为顶部、文中和底部。

公众平台作为一个自媒体平台，需要有良好的文章发表环境，鼓励优秀的作者写文章，并且坚决反对不符合规范的行为，如：欺诈、色情、诱导行为、不实信息、违法犯罪、骚扰、内容未经授权/滥用原创、侵权行为等。欺诈行为又包括多级分销、网络借贷、高额反利等；诱导行为包括诱导分享、诱导关注；不实信息包括政治类的、医疗类的、社会事件内的不实信息；内容未经授权/滥用原创包括抄袭/洗稿，滥用原创。对于抄袭洗稿行为，完全的清除是不可能的，只能通过一系列的手段来阻止此行为。除了在公众平台方面检测出文章是否存在抄袭，从第一层面发现这种行为，还为作者提供了维权的手段、为用户提供了举报的方式。图文系统提供给用户方便的投诉渠道，并且细分投诉的分类，使得投诉的类目更加明确。系统将投诉分配给有专业知识的人员处理，使得投诉可以尽快得到处理。用户可以填写投诉的描述和上传证据截图，辅佐平台进行判定。文章有许多种类型，原创文、转载文等，当为转载文时不仅需要有标识，还需要给用户阅读到原创文章的途径，因此需要提供阅读原文的入口。

图文系统向订阅者提供推送文章、图文展示、图文互动、管理已关注的公众号等功能，具体的公众号图文系统的用例图如下图 3.2 所示：

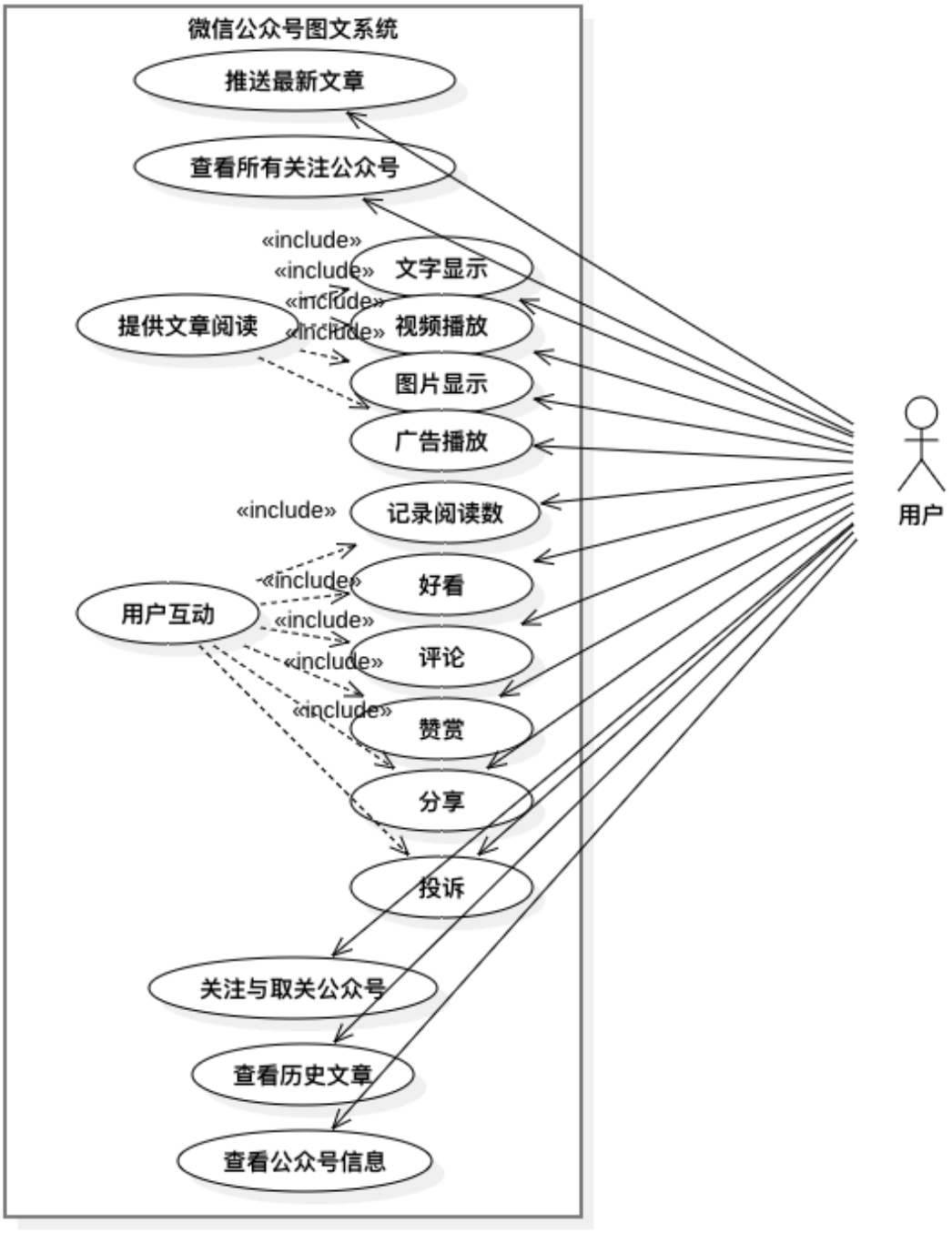

图 3.2 微信公众号图文系统用例图

3.2.2 图文系统非功能性需求

公众号推送的文章，依靠标题、封面或者简要概述吸引了用户点击打开，需要能够快速的展现在用户面前，并且能够与用户交互。如果当用户打开某篇文章需要等到好几秒甚至几十秒的时间，用户在等待的过程中会关闭该文章。用户就会拒绝甚至反感继续使用这一平台，到其他的自媒体平台阅读文章，最终导致用

户流失。因此系统快速响应、页面打开时间短、可交互时间短是非常重要的非功能性需求。

当公众号图文系统获得用户信息后，用户才可与该系统交互，获得关注和取关公众号、分享文章、投诉文章、评论、在看等交互功能。在无用户信息时，用户只可以获得文章的阅读功能，并且文章中是没有广告的。图文系统嵌入在微信应用中获得微信的登陆状态信息，但是微信作为一款聊天类型的软件，需要注重用户的隐私，微信中还有微信支付，涉及到个人财产问题，所以保护用户信息不被盗用非常重要。

订阅号文章的日均打开量达到近 20 亿，热门的文章会有几百万的转发量，某一时刻同时访问系统的人数会达到将近 1 亿的数量，因此服务器端需要有足够大的负载能够支持访问量。服务器几秒的崩溃，可能就会导致几千万的广告损失。并且系统需要有容错性，当某几台服务器崩溃时，剩余的服务器能够继续提供服务，不会影响正常的使用，需要具有很高的稳定性。

公众号图文系统需要不仅需要可以在微信应用内使用，还需要兼容 iOS、Andriod 各大品牌移动设备，也需要支持在 IE、谷歌、QQ、Safari 等浏览器中显示。因此系统需要具备良好的兼容性。

公众号图文系统需要与其他系统交互，与微信公众平台网页端系统、订阅号 App、赞赏助手小程序等系统联系紧密。因此，系统保存的数据需要能够互通，其互操作性如下图 3.3 所示：

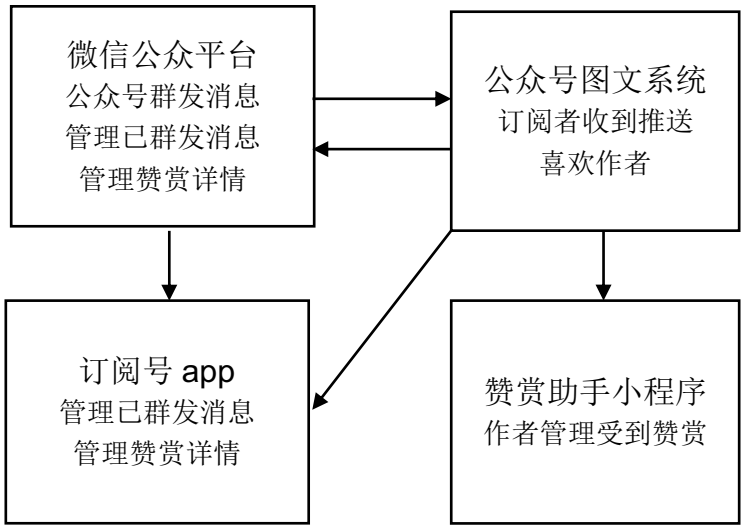

图 3.3 微信公众号图文系统与其他系统关联图

当公众号在微信公众平台网页端群发消息，公众号主持者能在订阅号 app 中看到已发表的文章，订阅者能在图文系统中收到推送。当读者阅读到喜欢的文章，并对作者赞赏后，在微信公众平台、订阅号 App、赞赏助手小程序中都能查询到相关信息。

图文系统在设计和实现上需要考虑时间特性、负载、安全性、兼容性、互操作性等非功能性需求，总结如表 3.1 所示：

表 3.1 公众号图文系统非功能需求列表

时间特性	公众号图文页面打开时间不超过 2s
	公众号图文系统可交互时间不超过 5s
负载	能够同时支持超过 1 亿人同时访问
安全性	保证微信用户的信息安全，不会出现信息泄露，和盗用用户信息的情况。
兼容性	微信端兼容 iOS、Andriod 设备，浏览器端兼容谷歌、Safari、QQ 等各大浏览器。
互操作性	能够与微信公众平台系统、订阅号 app、赞赏助手小程序等互相联系，互相访问数据信息。

3.3 公众号图文系统总体设计与模块设计

3.3.1 公众号图文系统总体设计

公众号图文系统的整体架构分为展示层、逻辑层和数据存储层。展示层由前端和客户端人员负责，逻辑层由后台负责，数据存储层设计由数据处理人员负责，每个层的互通由运维人员负责，各个部门的人员各司其职。展示层分为在微信客户端内展示和在浏览器中展示，在微信客户端展示需要与客户端进行协作，并且提供给用户播放广告、分享、评论、投诉等交互性的功能，在浏览器中显示则不具备以上功能。客户端中展示是用客户端提供的 webview 中打开 HTML 链接的方式实现的，是混合式的开发，既具有 web 开发迭代快的特点能够响应快速变化的需求，同时能够通过客户端实现与微信应用相关的功能。

接入层是面向用户的层面，通常是监听用户的 ip。CGI 是指服务器根据 URL 开启进程的启动程序，是前端与后台联调的接口，动态生成 HTML 页面，通常做一些参数校验、登录态的校验等。CGI 是用 C/C++编写的，在接入层之后，后台

server 之前，起到一个网关的作用，放在 webserver 上，转发给后台 server 的二进制的程序，有时候会做多个后端的数据汇聚并承担部分简单的逻辑处理。后台 server 做私有协议处理，是真正的业务逻辑处理处理前端的页面请求，与数据层之间交互，做数据存储、查询等操作。在浏览器中可以直接通过访问 CGI 链接展示图文信息，公众号图文系统大致的架构如图 3.4 所示：

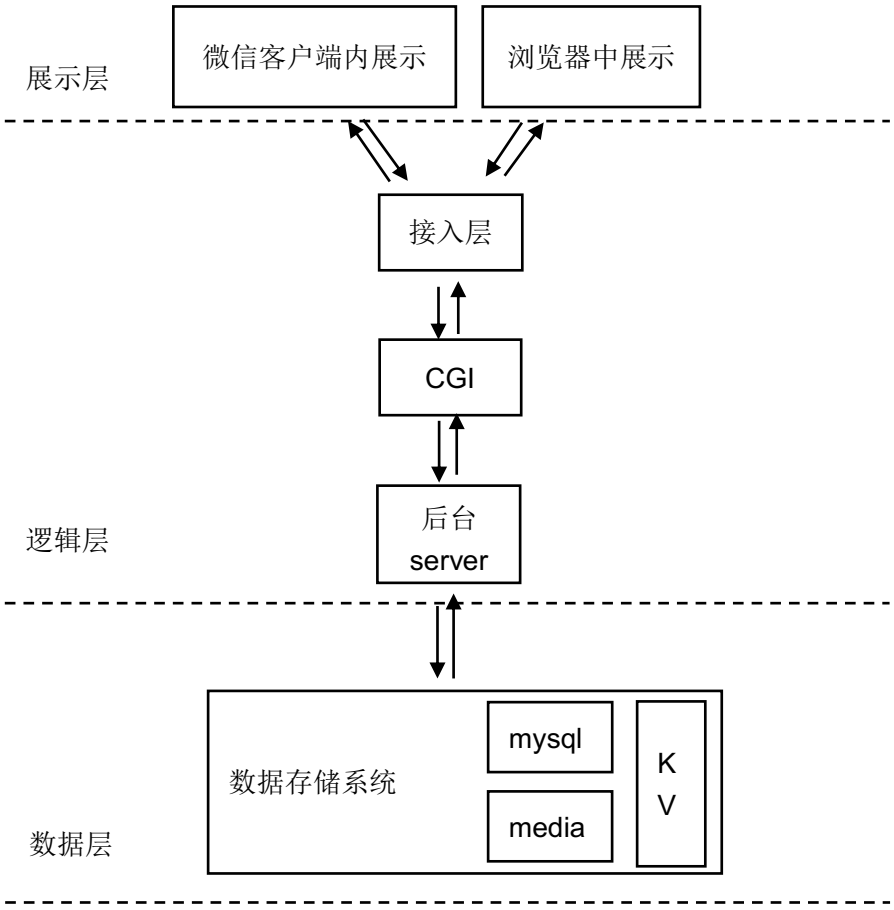

图 3.4 图文系统整体架构图

一般而言，通过浏览器端查看网络请求，看到请求的链接中有 cgi-bin 这个参数即为 cgi 请求。称为 cgi-bin 的原因是，一般在服务器端用 cgi-bin 目录存放脚本，也可以通过查看请求的返回数据是否带有 HTML 标签判断是否是 CGI 请求。系统请求流程如图 3.5 所示:

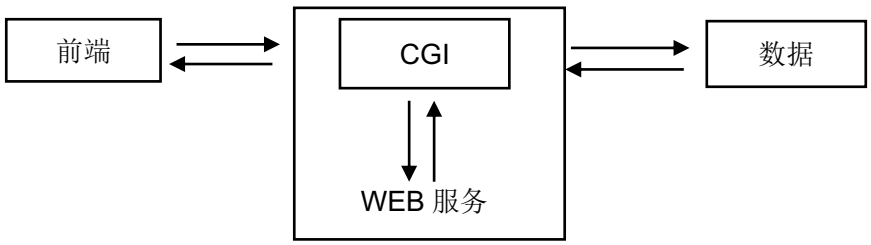

图 3.5 图文系统前后端请求流程图

前端将 CGI 请求通过 HTTP 发送给服务器，服务器通过配置环境变量或者管道的方式将请求交给 CGI 来执行，CGI 处理完毕后将结果传给 Web 服务器，服务器通过 HTTP 将数据返回，前端将数据展示出来。公众号图文系统使用 svrkit 框架作为后台服务框架，服务间发送同步 RPC 请求进行相互调用。

数据层由多种数据存储介质实现，根据业务数据的需要存储在相应的介质中。后台逻辑与数据层的交互并不是直接调用，而是通过访问数据层提供的数据服务，数据服务包括数据访问服务和数据存储服务。将图文系统中固定访问的资源存放在 media（专门存放媒体资源的服务器）中，运维将链接地址与在机器上的路径对应起来。因为固定资源访问的次数会更加频繁，这种方式可以直接获取，而不需要查找，节省了时间。MySQL 用来存储计算量并不是那么大的数据，内部的 KV 存储系统是名为 SDB 的一个 key-value 存储系统。数据访问服务能够根据需要将数据访问或操作请求处理后转发到相应的数据存储服务，逻辑处理和数据存储分离，维护数据的稳定与安全。系统的每层职责分明，通过提供服务的方式相互访问。这种设计方式满足抽象的原则，对外暴露的信息较少。

3.3.2 公众号图文系统模块设计

公众号图文系统大致可以分为五个大的模块，图文消息推送模块、图文展示模块、图文互动模块、图文秒开模块、公众号管理模块。图文消息推送模块负责文章的推送，当公众号在微信公众平台编辑文章并确认后，立即或定时推送给在群发列表的用户。微信中订阅号消息聊天条目将会被置顶，并且有红点提示给用户。当用户进入订阅号消息后，关注的公众号新发送的文章以信息流的方式展现。订阅者通过不断下拉的方式，浏览近期发布的文章，当遇到感兴趣的标题时，即可点击进入浏览，看到的即是公众号图文系统中最关键的模块：图文展示模块。当想要阅读一篇文章，打开却需要十几秒，在等待的过程中用户就会流失；文章中的图片很小，想要看清图片，换个大屏的手机，图片依旧很模糊，便会被认为是一个不好的阅读体验。自媒体平台最核心是内容，当有丰富的内容并且文章也具有足够的深度，那么文章的展示就成为自媒体应用最重要的一项。在图文展示模块，每个小的细节的不关注，功能的不够全面，都会造成新关注用户不增长或用户流失。公众号管理模块负责公众号相关功能，比如：关注与取关公众号、查

看公众号相关信息、查看公众号发表的历史文章、公众号主页自定义菜单等功能。当用户得到很好的阅读体验，并且对文章很认可，便会想要分享文章，发表自己的观点，图文系统中需要提供互动能力。为了营造良好的阅读环境，使得优质的而不是同类相互抄袭的文章被传播。提供给用户投诉的手段，这些功能属于图文互动模块。公众号图文系统的这五个模块共同构成公众号图文系统，模块间的关系如图 3.6 所示：

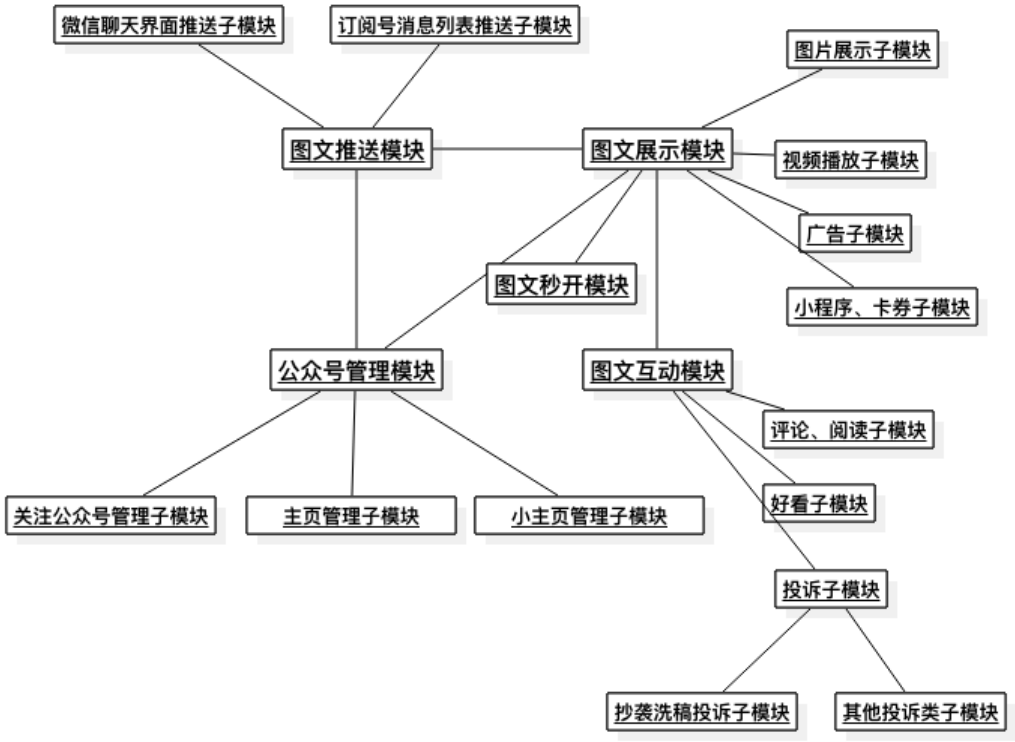

图 3.6 图文系统模块间关系图

每个模块由子模块构成，子模块由更小的子模块构成。比如图文互动模块，由评论、好看、投诉子模块构成；投诉子模块由抄袭洗稿投诉子模块、其他类型投诉子模块构成。每个模块负责各自的功能，模块之间相互协作。

当发布者在微信公众平台发布一篇文章后，无论是定时发送还是立刻发送，文章通过后台发送到图文系统，图文推送模块向用户推送该文章。当用户对文章感兴趣时，选择查看阅读文章，图文系统展示模块将用户感兴趣的文章展示给用户。用户可以在此基础上，选择播放暂停视频等图文中的操作，也可以评论该文章，当被发布者筛选后，会展示在文章主体内容的下面。订阅者对文章选择在看，并且确认后。喜欢的文章被推送到看一看系统，展示在看一看系统中，朋友也会

看到。订阅者可以投诉文章，比如文章抄袭，该项投诉会被推送到审核系统。明显的抄袭会被审核系统直接处理并返回结果，其他遗留的投诉项会被运营审核人员处理，审核系统将结果推送到图文系统，图文系统再推送给用户。当无法辨识的抄袭行为，图文系统会询问用户是否补充材料，并在最终处理完成后，同时告诉投诉者和被投诉者处理的结果，投诉成立将会对被投诉者进行惩罚。图文系统整体时序图如图 3.7 所示：

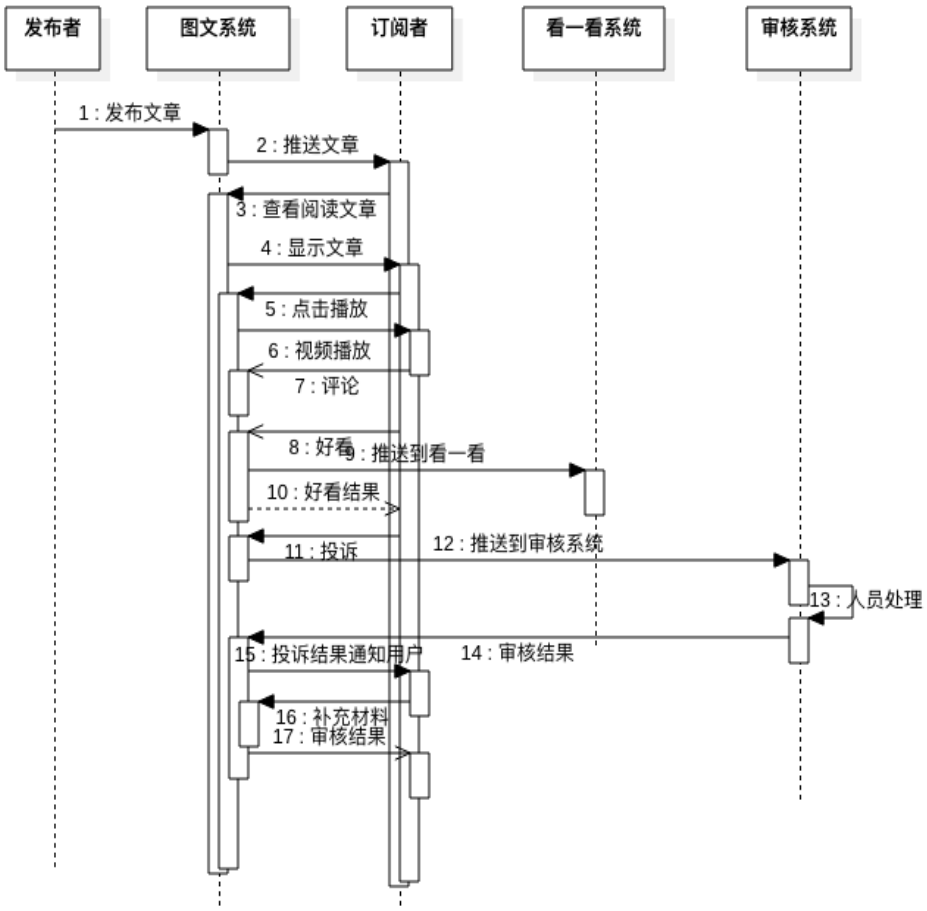

图 3.7 图文系统整体时序图

图文系统中每个模块有各自的功能，又相互联系通信。下文将重点说明图文推送、图文展示、图文秒开、图文互动这四个模块的设计。

3.3.3 图文推送模块设计

当用户关注某订阅号后，即成为该订阅号的订阅者，订阅号可以在公众平台系统中查看所有订阅的用户，并可以对用户分组管理。订阅号在微信公众平台网页端编辑文章或者直接从素材库选择素材后，发送文章给合适的人群。订阅号在

群发文章时，可以根据标签或者性别等选择对象，可以设定时间选择定时发送或者立即发送，后台会触发推送图文的逻辑。

图文推送模块包括新消息提醒功能、订阅号 timeline 功能、星标功能、推送设置等功能。在图文系统中推送一条图文消息，首先订阅号消息这一条目会显示有几条新的消息，消息内容显示为最新图文消息的标题，前缀为新的消息的条目，同时有红点提示。从形式上看，和普通的朋友聊天或群聊的消息是类似的，实现方式上也是相似的。在收到推送消息后，用户打开显示的内容是与聊天条目不同的。聊天条目是显示与朋友的聊天界面，而订阅号消息显示的是订阅号消息列表。订阅号 timeline 功能以信息流的形式展示最新发布的公众号文章，其中未读的星标公众号发布的文章会显示在最前。主页面以标题加文章中首图的形式展示，不显示摘要信息，因此标题会起比较大的作用。但是图文系统在主页面增加了对公众号更便捷的取关功能，因此公众号需要在标题党和内容深度上做出权衡。

图文推送模块的流程为当发布者推送一条消息时，数据存储到图文推送系统，图文推送系统发送给订阅者。消息推送的过程可以分为两个环节，消息到服务器和服务器推送给接收者。当发布者发送消息到接入层 connectSvr，接入层收到消息交给逻辑层 sendSvr 处理，逻辑层会进行一些比如反垃圾、黑名单的处理逻辑，若不丢弃则存储到存储层 msgStore 中。服务器将图文消息推送给订阅者，pushSvr 负责消息的推送逻辑，sendSvr 逻辑处理完将图文消息转发给 pushSvr。pushSvr 逻辑查询到订阅者属于的接入层的 ConnectSvr，将消息发送给 connectSvr，最终发送给接收者。各个模块之间的通信是一种长连接，这种依赖长连接的方式会有缺陷，因为在微信 app 退出到后台一段时间后，会清空微信拥有的所有资源比如 cpu，内存等，这个时候 connectSvr 对应的长链接将会失效，并不会推送成功。因此需要通过手机第三方的推送系统，比如苹果 ApnsPush、微软的 WpPush 等。这种方式下，PushSvr 发送一个推送提示给 ApnsSvr 服务器，ApnsSvr 服务器发送给订阅者，发布架构如下图 3.8 所示：

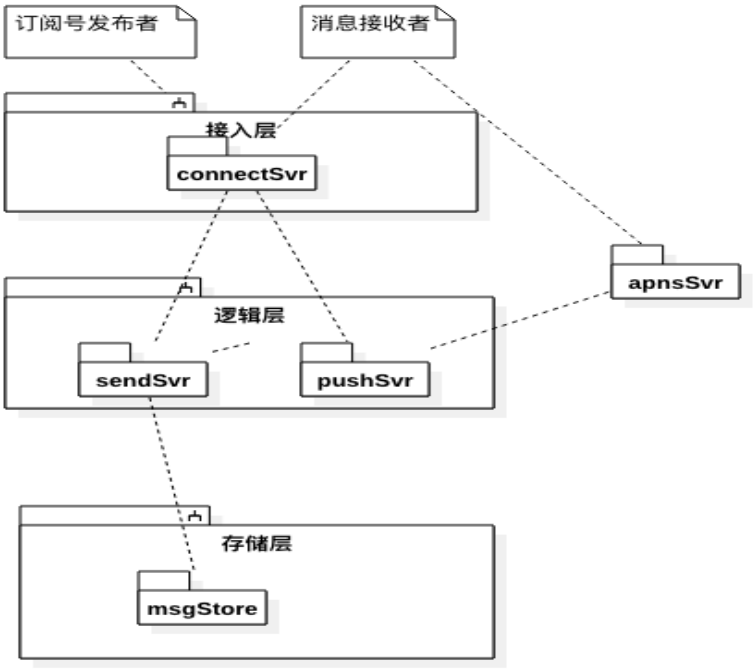

图 3.8 图文消息发布架构

图文消息的接收阶段：当订阅者打开微信应用时，客户端发送消息请求到 connectSvr 中间层，connectSvr 处理该消息请求。connectSvr 将该消息请求转发到 ReceiveSvr 实际的逻辑处理模块，ReceiveSvr 从数据存储 msgStore 库中查询，获得客户端未接收到的图文消息，按照此流程将消息返回到客户端，订阅者即能收到最新的消息。消息的接收框架图如图 3.9 所示：

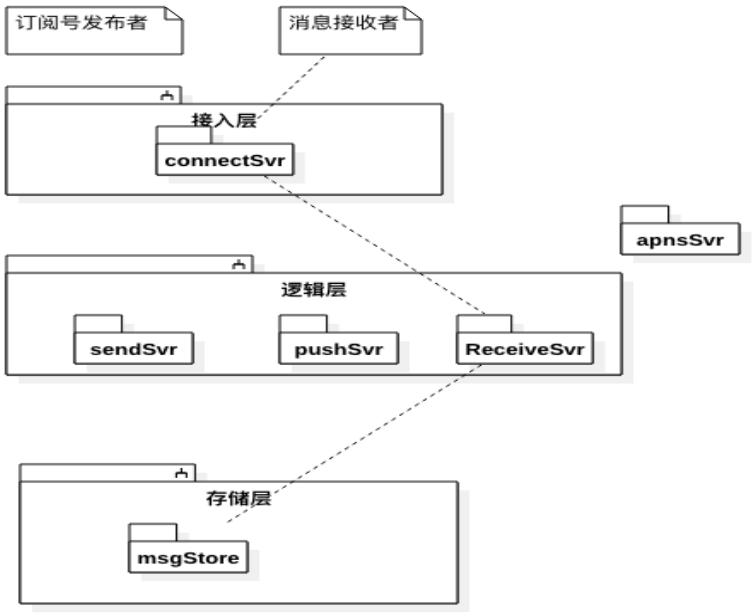

图 3.9 图文消息收取架构

图文消息的发送和收取架构能够基本满足在微信 app 中，用户能在 100ms 以内收到消息。即使用户退出微信到后台，第三方的 ApnsPush 服务也能在秒级单位内通知到用户登录微信收取消息。

3.3.4 图文展示模块设计

图文展示的入口场景有许多种，公众号会话（即订阅号消息主页面）、朋友圈、好友转发、看一看、搜一搜、历史消息页、腾讯微博和其他应用比如 qq，还有搜狗浏览器中也可以查找并阅读公众号文章。其中主页面、朋友圈、好友转发、看一看、搜一搜、历史消息页都是在微信应用中，因此在这些场景下打开的图文都具有互动功能。公众号会话的占比最高，大部分的场景是经过图文推送系统推送消息到用户，用户打开阅读文章。第二大占比的来自于朋友圈转发，属于图文互动模块的分享功能。在微信中的场景实现方式是一致的，即通过在 webview 中嵌入 Html5 页面方式展现。图文展示除了在 webview 中展示，凭借其 H5 页面的特性，可以直接在浏览器中展示，展示功能没有差异。

图文展示模块包括文字、图片、视频、音频、卡券、商品、广告、小程序、阅读量、原文显示，其中文字展示包括：标题、日期、公众号、作者、主体文字。每个人的阅读习惯不同，喜欢阅读时字体的大小也不同，因此图文展示模块提供调整字体的大小，分为 5 档。图片的展示功能需要支持原图显示、图片懒加载、图片编码处理等功能。视频和音频展示，基本需要支持播放、暂停，特殊的需求是视频需要支持自动播放。卡券、商品、小程序功能需要与微信应用交互，能够将卡券保存到微信支付中。当用户打开商品链接时，自动调起小程序。公众号最主要的收益来源来自于广告，广告主投放广告后，订阅号文章中能够以丰富的样式展示广告。广告同时也有几种形式，视频、图片、文字等多种类型广告，广告的位置可以在顶部、文中或底部。广告的显示是一个可选项，订阅号可以设置不允许显示广告，同时不会获得收益。订阅号在编辑文章时，可以选择自动插入广告，公众平台系统会在合适插入的位置插入广告，一篇文章中最多两个广告。图文展示模块的功能组成总结如图 3.10 所示：

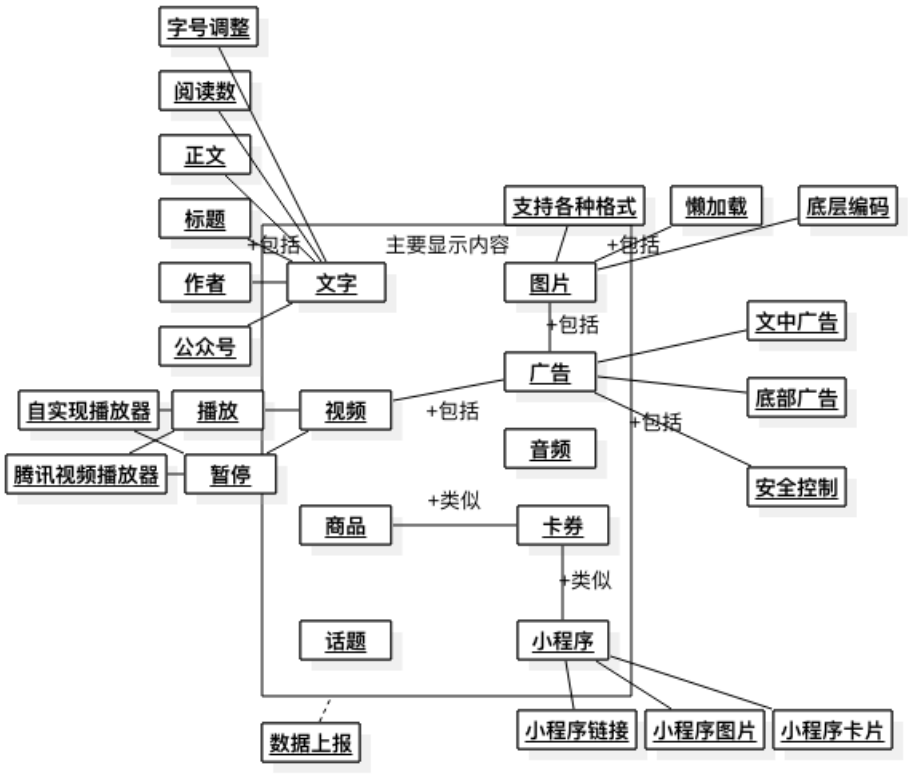

图 3.10 图文展示模块功能组成

商品、卡券与小程序有类似的地方，都是以链接的方式嵌入，需要与其他系统交互，比如：京东、小程序、微信支付等。因此在设计时使用公共的基础类，各自在组合该公共类的基础上，各自实现类来实现自身的特性。

视频播放功能在最初实现时是使用腾讯视频的播放器，直接调用其提供的接口。但是后来考虑到业务特性扩展(支持在视频中插入广告)、样式统一、减少带宽浪费等需求，无法对原来腾讯视频的代码或者其他播放器的代码进行修改，因此系统内部实现了视频播放器。当特殊情况下，原生视频播放器无法播放时，向下兼容使用腾讯视频播放器。在微信应用中，优先使用原生播放器，在浏览器中因为不具有客户端内核支持，使用腾讯视频播放器。视频中需要插入广告，同时广告也有视频广告类型，还有图片广告类型，广告中组合视频与图片的特性，在视频播放器中加入支持广告的扩展插件。广告是面向用户的，浏览器中无法获取用户的微信信息，因此单纯在浏览器中展示的图文是没有广告的。

音频类的也是自主实现了一套声音播放器，在插入的音频来自于 qq 音乐或酷狗音乐时，获得链接，使用 qq 或者酷狗的接口校验插入的音频是否具有版权，再播放。

整个图文展示系统是公众号图文系统中最核心的模块，对页面中的 js 错误进行记录监控。图文打开的速度、图片加载的速度、图片加载是否失败、视频播放、视频暂停、视频获取的连接、小程序跳转情况等，每个环节的数据都会进行上报，通过数据来监控整个系统。将数据接入到监控系统中，以折线图、饼状图等图形化界面展示，并且在异常时有告警和提示。

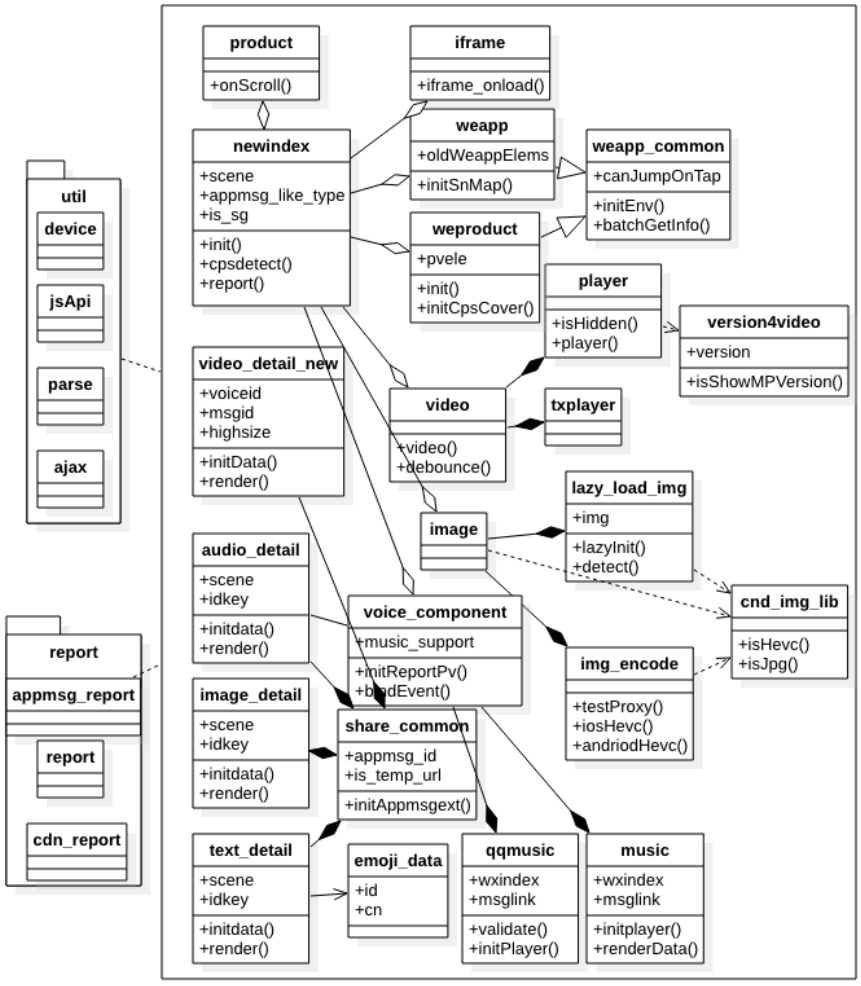

图 3.11 图文展示模块类图

如上图 3.11 所示，公众号图文系统图文展示页面包括图文页、分享页，分享页又包括视频分享页、音频分享页、图片分享页、文字分享页。总的页面入口有以下几种：newindex，video_detail_new，audio_detail、image_detail、text_detail。Newindex 类是图文类，组合了视频 video_player、图片 image、小程序 weapp、数据上报 appmsg_report 等图文展示模块中所有类。需要与微信客户端交互的功能封装为 jsapi 类，该类是对客户端提供的 WeixinJsBridge 类的再次封装。比如告诉客户端当前订阅号的名称和图标，当用户使用浮窗功能时，

可以将图文暂时收进浮窗内，浮窗显示订阅号的图标。用户可以点击浮标，回到之前阅读的位置。图片处理 Image 类，具有处理图片的显示样式，响应用户的 tap 事件等功能。Image 类组合对图片懒加载进行处理的 lazyload_img 类。为了节省带宽，对图片使用 WebP 或者 HEVC 编码方式，设置图片链接的后缀。将图片发送到图片资源服务器请求图片资源时，调用不同内核的编码库，返回对应编码的图片。cnd_img_lib 类是图片格式判断的接口函数，比如是否为 gif、png 等。Appmsg_monitor 类是对图文进行监控类，包括：图片尺寸监控、防劫持监控、xss 监控。Sogou_index 类处理针对于需要在搜狗浏览器中显示所需要的特性，比如：针对搜狗浏览器的需求，对图片进行放大。投票和卡券的实现方式类似，因此设计时将其封装为同一个类 appmsg_iframe。该类对 iframe 的宽度自适应处理，上报曝光信息，处理跳转。小程序商品 weproduct 类、小程序卡片 weapp 类等有关小程序的显示，提取出它们之间公共的部分封装为 weapp_common 类，比如获得链接、跳转链接、获得商标等，再在各自的类中实现各自特有的功能。广告 ad 类是广告的总入口类，其中又包括广告卡片 card 类、品牌介绍 profile 类、赞助广告 sponor 类等。系统的公共函数库放在 util 包中，util 包中包括处理 dom 事件的 domEvent 类、获得用户设备类型的 device 类、获得用户微信版本的 mmversion 类等，约 50 几个公共类。因为这些公共函数库是每个页面都会使用的功能，因此拆分成小类，具有可扩展性和可重用性。虽然图文展示的入口是一个大类，但是内部细分成了许多小类，前端使用 SeaJS 框架对各个功能进行模块化管理，每个小类的功能划分明确，整体的结构比较清晰，易于维护。

3.3.5 图文互动模块设计

图文系统在满足图文阅读功能的基础上，提供丰富的互动能力。在读纸质书籍的时候，许多人会写书评。自媒体时代，诞生了文章评论人这一职业，在微信订阅号的文章中同样提供评论的功能。订阅者的评论会被提交到后台，推送给订阅号管理者。但订阅者发表评论不一定会被采纳，比如一些侮辱言论、毫无意义言论等。被采纳的评论，会显示在文章下方的评论区域。为了文章的阅读体验，系统控制最多显示 50 条精选评论。阅读文章的其他用户可以对评论点赞，评论

的展示顺序按照点赞的排行，从高到低展示。用户可以取消对评论的点赞，评论者可以自主选择是否删除评论。

许多自媒体平台中，都提供点赞和评论功能。在之前版本中，系统提供对文章的点赞功能，用户可以随意点赞或者取消赞。赞的行为并不会被好友看到，只是表示对订阅号文章的认可。在微信中推出看一看功能后，增加阅读文章时的互动性。用户可以认为文章好看，同时可以添加评论，系统会将该文章推送到看一看中。好友可以看到该用户在看的文章，可以阅读，点赞以及评论。是一种为看一看引流的方式，也使优秀的原创能够被传播，被更多的人看到，这种方式促进构建更优质的文章创作平台。另一种提供文章传播的方式是分享，用户可以将文章分享到朋友圈、发送给朋友、分享到微博等应用，也可以单纯复制链接发送给朋友。为了维护良好的自媒体创作环境，提供投诉的手段给阅读者、创作者、订阅号。投诉的类型分为许多种，每个一级投诉分类又会有二级投诉子分类，对投诉做到细分，便于定义投诉的范围和对投诉进行处理。用户提供一些辅助材料，供系统和处理人员处理，系统及时将投诉的结果告诉用户。

所有图文互动都是基于用户信息的，在订阅者图文系统中是使用微信用户的信息。在无用户信息的场景，不提供评论、在看、分享、投诉等交互功能。当用户打开图文不需要提示用户登录，图文系统自动获得微信用户的登陆状态。为了安全考虑，用户的登录状态只缓存一定的时间，并不会永久有效。

图文互动模块中各个交互功能相互独立，又都依赖用户信息的模块。为了用户的阅读体验，主体的图文部分首先加载，评论、点赞和广告等信息异步加载。对公共的异步加载逻辑封装为 async 类，获取异步加载的数据，比如阅读量，评论等信息，因为评论、在看等信息一般都在图文的底部，这种方式下不会影响用户首次打开的体验。Newindex 类组合 async 类，处理图文中异步加载的部分，async 类聚合阅读在看类 like、广告类 ad。投诉类的入口在客户端图文页面右上角的点点点标志中，由客户端提供。投诉类 complain 是一个总入口，提供各种类型的投诉。违规原创、抄袭、洗稿等投诉，有各自的特性，比如：违规原创投诉提供补充证据材料，洗稿有另外的推送给评委邀请评委进行洗稿评定的流程，有推送给被投诉者的二次确认。图文互动模块的类图如图 3.12 所示：

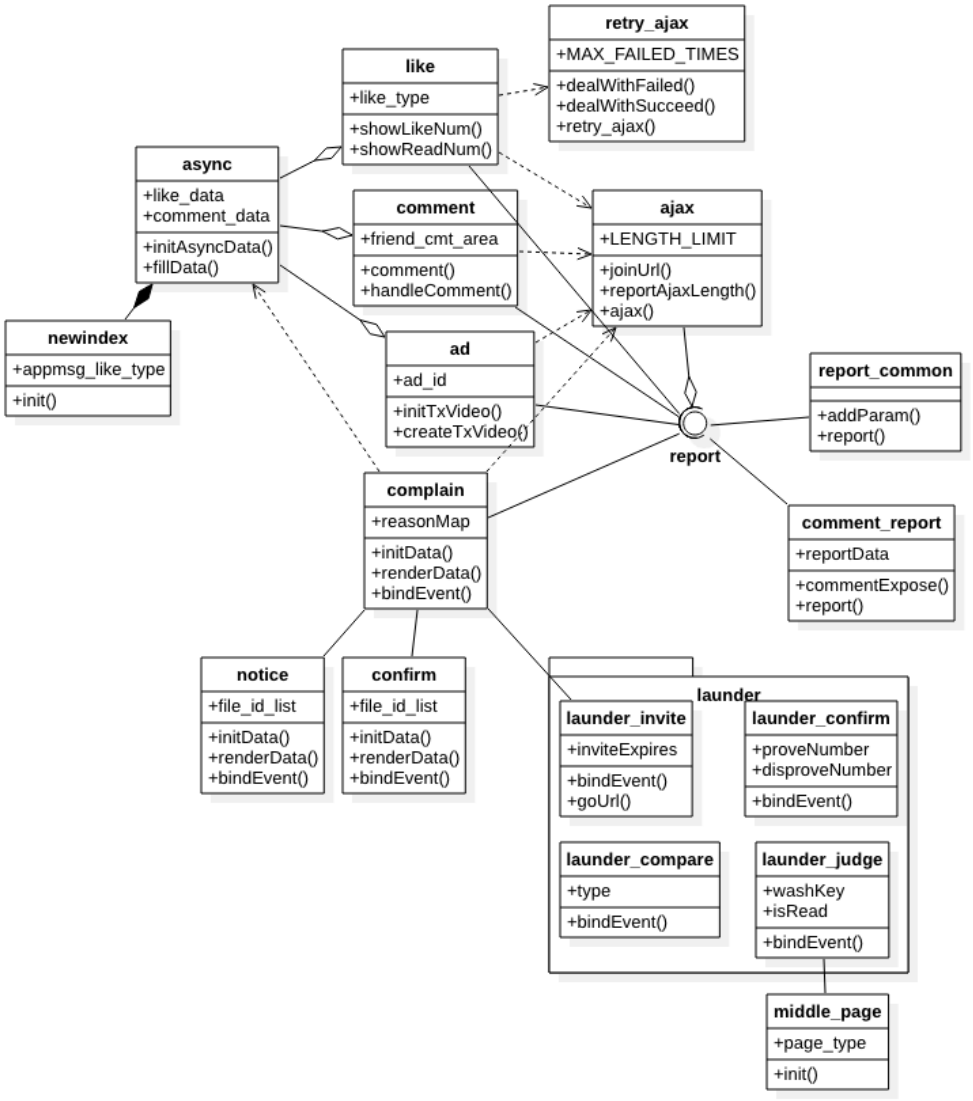

图 3.12 图文互动模块类图

上图中，launder 包负责洗稿相关的功能。其中有洗稿合议邀请 launder_invite 类，当公众平台筛选出比较优质的评论人员、订阅号作者时，向相关人员发出邀请，收到邀请的人员可以选择接受或拒绝。launder_compare 类，提供给洗稿合议人员对比文章的功能。评论结果提交类 launder_judge，洗稿合议人员可以选择文章为洗稿或不洗稿，评论可以添加自己评定结果的理由。当收回的有效合议结果与发出的合议邀请的占比达到一定的值时，系统对结果进行评定，否则重新发起洗稿合议。launder_confirm 类负责通知包括洗稿文章作者、原创文章作者、参与洗稿合议人员评定结果。当有判定结果后，系统对洗稿文处理，删除该文章，将访问该文章的链接替换为中间页，在页面中说明该文章是洗稿文章，并引导用户阅读真正的原创文章。

3.3.6 图文秒开模块设计

图文秒开模块是对整个公众号图文系统的优化模块，从首屏时间(用户点击图文到用户看到图文所用的时间)和可交互时间(用户可以进行留言、点赞等操作需要等待的时间)两个方面对订阅号图文系统进行优化。在图文展示和图文互动模块已经充分考虑了各种前端优化方案并进行了设计，包括：图片懒加载、资源 cdn 获取、缓存、页面 js/javascript 布局结构、模块设计、QQMail 模板页面直出等。根据上面的设计方案，从用户点击图文到图文展示的时序图如图 3.13 所示：

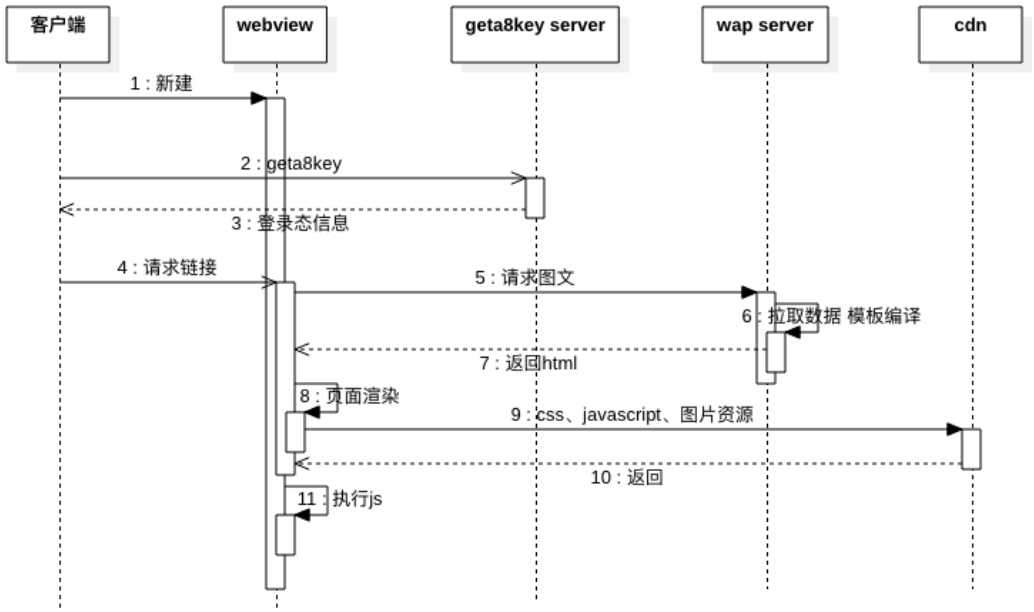

图 3.13 图文系统渲染时序图

大致流程为，在用户点击图文时，客户端会新建一个 webview，然后客户端会发送一个 geta8key 请求，这个请求是用来获取用户的登录态信息。当客户端得到回包后，客户端发送一个带有登录态的 url 给 webview，webview 根据 url 请求 wap 的后台。图文系统的后台去请求图文的数据，使用 QQMail 模板引擎，将数据和模板文件编译成页面返回。当 webview 接收到 html 后进行页面的渲染，渲染完毕后即时首屏时间。Webview 在解析的同事会向 cdn 请求 css、js、图片等资源，接收到返回后并且 javascript 执行完成后，即是用户可交互时间，以上是以前设计的整个系统交互流程。但是一篇图文需要加载的资源非常多，仅仅靠前端和模板渲染等是无法做到在秒级单位内打开的。

图文秒开模块从每个阶段对图文系统进行优化，geta8key 技术是用来获得用户的登录态，以此方式实现图文互动模块内的功能，用户更关注的是图文能够被立刻看到，因此以前等待 geta8key 返回后再请求图文，可以改成两者并行，能够节省许多时间。第二个主要耗时是网络请求，网络请求的耗时主要包括三个部分：网络环境、回包的大小、通信协议，网络环境是外在条件，无法改变。一篇图文系统中的文章，它的回包主要包含两个部分，HTML 代码和与文章相关数据。HTML 代码这部分对于每个图文都是一样的，不一样的是页面的数据。既然对于用户都是一样的，那可以把 HTML 的代码放到客户端，从而达到减小网络回包大小的目的。将 HTML 代码离线化，所谓离线化，就是把代码放在客户端。HTML 离线化后，当 webview 接收到文章数据后，需要在前端把 HTML 代码和数据合并起来，进行前端编译。使用这个方案之后，每次请求页面的数据，这种方案可以不走网络 https 请求协议，而走客户端私有协议。网络请求 cdn 资源也是一大耗时，因此考虑将 css、图片等资源离线化并且进行预加载。流程优化后时序图如图 3.14 所示：

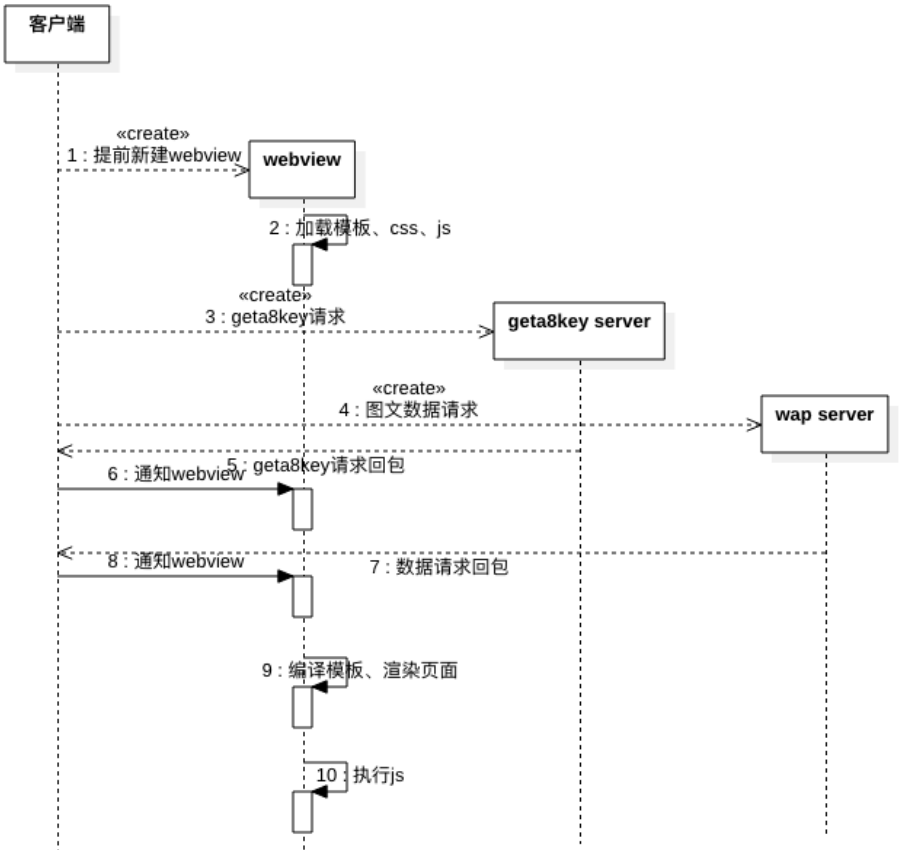

图 3.14 优化后图文系统渲染时序图

当用户点击订阅号消息时，客户端预加载 webview 的同时加载前端代码。当用户点击文章时向后台请求数据，向 geta8key 服务器请求用户登录态。当 webview 获得数据后，渲染页面。当用户再次点击文章时，同样获取登录态信息，直接获得缓存的数据，渲染页面。HTML5 页面与秒开页面总体的对比如表 3.2 所示：

表 3.2 HTML5 与秒开页面对比表

HTML5	图文秒开
Geta8key 请求同步	Geta8key 请求异步
网络请求拉取 HTML	HTML 离线化，网络请求拉数据
后台模板编译	前端模板编译
每次新建 webview	Webview 预加载
网络请求拉取 JS/CSS	JS/CSS 离线化

图文秒开优化是依赖于微信客户端的，在浏览器中打开仍是走以前的流程，目前只有在从订阅号消息列表进入图文时会使用图文秒开模块。图文秒开模块对系统进行了优化，提升了页面的打开速度，极大地缩短了首屏时间和用户可交互时间。

3.4 本章小结

本章首先对整个公众号图文系统和相关联的系统进行了概述，分别对功能需求和非功能需求进行了分析，以用例图和表格的形式展示了系统需求。之后对公众号图文系统进行总体设计，以架构图的形式描述了整体的架构和以时序图的形式描述整个系统的通信流程。将整个系统分成图文推送、图文展示、图文互动、图文秒开、公众号管理模块，对模块之间的关系进行论述。细分描述了主要模块的功能，以类图、时序图、内容组成图等形式对每个模块进行详细的设计，为之后实现做准备。

第四章　公众号图文系统实现

4.1 图文推送模块的实现

4.1.1 消息推送与收取实现

在第三章图文推送模块设计中，以时序图的形式描述了图文推送模块内各个模块间的交互。整个消息推送与收取功能的实现，是基于邮箱存储转发的拉取机制，流程上是服务器通知新消息到达，客户端主动上服务器去拉取的过程。在实现上还需要考虑在弱网等特殊情况下消息丢失的情况。最简单的实现方法是消息收取者每次都发送一个 ack 确认请求给服务器，但是在特殊网络请求下，ack 请求也会丢失，并且这种方式在服务器与消息收取者之间交互太多。在图文系统实现时使用 Sequence 队列作为消息收取的确认机制，给每个用户分配最大为 42 亿的队列，每收取一条消息都会记录下来。消息收取者维护当前已收取消息的队列值最大值，服务器记录为该用户已经分配到的最大值，将两者的最大值进行对比，从而确保信息的不丢失。Sequence 队列消息确认流程如图 4.1 所示：

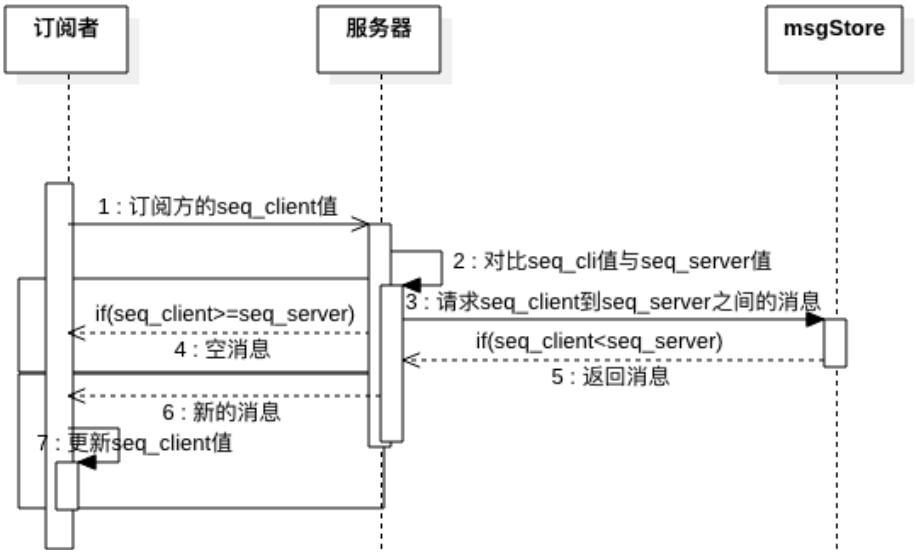

图 4.1 Sequence 队列消息确认流程

客户端发送已接收图文消息的最大值 seq_client 到服务器端，服务器端将接收到的 seq_client 的值和在服务器端维护的已发送消息的最大值 seq_server 的

值对比，当 seq_client<seq_server 时，说明手机端存在消息没有接收到，因此向服务器请求 seq_client 到 seq_server 之间的消息，返回给订阅者。客户端接收到消息后，将自身的 seq_client 值更新为 seq_server。如值不小于，那么则返回一个空的消息。这种方式能够保证用户接收到所有未接收到的消息，比如用户在某个时间点断网了，他目前收到的图文消息数为 100，此刻服务器端维护的该用户应收取的消息数为 115，当用户连上网络后，手机端微信发送自身的 100，服务器对比后，将 100 到 115 之间的图文消息返回给用户。当用户用两台手机登陆的时候，即使之前收取的消息维护的消息的 seq_client 值不一致，也不会造成消息的重复收取。

在实现中，对客户端主动上服务器拉取的流程也进行了优化。在 pushSvr 提示用户客户端去服务器接收消息时，pushSvr 已经可以计算未发送给用户的消息数，可以直接向用户推送未收到的消息。将客户端向服务器请求新的消息，服务器返回新的消息的流程简化了，极大提升了收发消息的速度。消息推送与收取的流程变更为，当公众号下发的图文消息 sendMsgRequest 发送到 connectSvr，connectSvr 将该推送图文的请求告诉 pushSvr 后，pushSvr 需要将消息通知给订阅者，connectSvr 向 receiveSvr 请求需要推送的图文数据后，告知订阅者有新的消息的同时将新的消息推送给订阅者。

以上模式是对订阅号和接收订阅号的用户的文章存储机器在同一个 IDC 机器下的设计，在整个消息推送和收取的交互过程中，connector 模块作为中间模块，与其他 pushSvr、sendSvr、msgStore 等模块交互。connectSvr 类是其中较重要的一个类，具有通知客户端有新消息、返回客户端未接收的消息、接收客户端的已经接收到消息的确认信号等关键功能。ConnectSvr 向客户端发送带有图文消息数据的 notifyData，需要先获取到客户端的 sequence 序列的值，才能使用客户端的 sequence 值向 ReceiveSvr 增量获取图文消息。因此在进入 NotifyData 状态之前，需要等待客户端主动做一次收取数据的请求，获取客户端的 sequence 值，并且存在 connectSvr 中。在 NotifyData 状态下，每次都需要对用户是否收到消息进行确认。如果不进行确认，更新了用户客户端的 sequence，然而消息并没有收取到，未收到的消息不会进行二次发送。在 connectSvr 中，

如没有接收到 ack 确认消息，并不会直接返回数据，会依旧走通知客户端的流程，让客户端主动拉取图文消息。ConnectSvr 类的定义如下图 4.2 所示：

```
class connectSvr{
    private: int clientSequence; // 客户端序列
        boolean ack; // 客户端消息接收确认
        int serverSequence; // 服务端序列
        int times; // 传输数据次数
    public: void notify() { };
        string getClientSequence(string uin) {
            if (this.validateUin(uin)) {return;}// 查询 msgStore 获得 client_seq
            return client_seq;
        }
        booolean validateUin(string uin) {
            // 判断 uin 是否被反垃圾，禁用
            boolean ifPermit = true;
            if (!this.isBeenAntiSpam() || !this.isBeenBanned()) {ifPermit = false;}
            if (!this.isInBizList(uin)) {ifPermit = true;}return ifPermit;
        }
        void getMsgRequest(string uin) {
            clientSequence = this.getClientSequence();
            // 查询服务器端 server_seq
            serverSequence = getSeverSequence();
            if (clientSequence >= severSequence) {// 返回空消息 } else
            {// 获得从 clientSequence 到 serverSequence 的数据} ack = false;
        }
        void notifyData() {
            if (times === 1) {
               notify();
            } else {
               getMsgRequest(); }
        }
        void ackConfirm() {
            ack = true;
            clientSequence = serverSequence;
            times++;
        }
     }
```

图 4.2 ConnectSvr 类代码

整个图文系统的推送流程在手机端的体验十分快速，能够让用户快速的收到消息。

4.1.2 多个 IDC 机器消息收发

多个 IDC(Internet Data Center)机器消息收发，指的是当用户属于不同的数据中心时，图文系统进行图文消息的推送和消息。目前微信共有 4 个 IDC 机器，香港、加拿大、上海和深圳。每个用户只能属于一个 IDC，并且从属关系不能随意切换，用户只能在自己所属的 IDC 机器进行消息的收发操作。下面将以当公众号在上海附近区域发送图文文章，位于深圳的订阅者接收到图文为例子。

图文系统使用 proxy 来实现多个 IDC 机器之间收发消息，proxy 位于逻辑层和数据层之间。当订阅号发送一篇文章时，sendSvr 将消息转发给 proxy，proxy 将接收者在其他 IDC 机器的用户消息转发到 proxyQueue，在同一 IDC 机器的消息存到 msgStore 中。proxyQueue 将消息存在队列存储器中，深圳的用户客户端通过 proxy 判断消息不是来自于同一个 IDC，通过 proxyProcessor 向上海的 proxyQueue 请求属于深圳的消息，proxyQueue 向数据存储的队列获取消息返回给深圳的用户。图文系统是用异步队列的方式进行多个 IDC 之间的消息传输，为了确保异步队列的稳定性和可靠性，proxyQueue、QueueStore 都设计为多个机器。proxyQueue 使用一致性哈希算法进行容灾部署处理，当收到消息需要存储时，使用一致性哈希算法选取一台机器进行存储。QueueStore 的设计是以组为单位，每个组之间没有关联，每个组有三台存储机器。使用 NRW 策略进行管理，总共有 3 台机器，有存储需求时同时写三台机器，有两台写成功即为成功，有读取需求时有两台读成功也认为成功，保持每台机器之间的一致性。

4.1.3 图文推送模块页面截图

订阅号图文系统在微信中的固定条目，提示新的图文消息条数和最新的推送文章的名称，在展开的订阅号消息中，按照时间顺序，标题加头图的形式展示，能够不断下拉收取较早的消息。可以查看经常访问的公众号，并且对置顶的公众号星标显示。图文推送模块中订阅号消息的界面截图如下图 4.3 所示：

图 4.3 订阅号消息页面截图

4.2 图文展示模块的实现

4.2.1 图片展示的实现

在微信公众号图文系统中，有些公众号的文章中可能会嵌入几十张甚至几百张图片。如果系统在加载文章页面时立刻去拉取所有的图片资源，会导致耗费许多时间，影响用户阅读体验。因此在展示图片时，系统实现懒加载机制，待图片快滑动到用户可视区域范围内再去 cdn 中拉取图片资源，可以缩短页面的打开时间，优化用户的体验。

懒加载的实现机制是初始时不加载全部图片，等需要的时候再加载。若是用户没有浏览到的图片即可以不加载，这种方式可以节省带宽。实现时，在编写 HTML 中图片元素时，对图片 IMG 标签不设置 src 属性，即不会发生图片请求。在后台返回模板数据时将链接保存在 IMG 标签的数据属性 data 中，即以 data-url='xxx'形式。系统监听读者的滚动行为，当图片即将滑动到可视区域范围时，批量设置 IMG 的 src 属性，请求图片资源。同时在 IMG 标签上记录图文的长宽比率和宽度，在初始绘制所有图片标签时计算出图片的占位，设置图片的宽高，避免在加载图片时出现跳动的情况。图片加载的 detect 方法代码如图 4.4 所示：

```
function detect() {
        images <- this.images
        if imagesIsNull return
        scrollTop <-this.getScrollTop() //先获取滚动条距离顶部的距离
        innerHeight <-this.getClientHeight() //屏幕可视区域高度
        bottom_offset <-this.offset || 60 //  图片距离可视区域 60 像素开始加载
        var top_offset = 0// 默认图片距离可视区域 0 像素  开始加载
        if window.networkType == "wifi" then
            loadNextScreenImgs
        endif
        initSomeNeededParams
        for i= 0 - images.length do
            imgRect <- img.el.getBoundingClientRect()
            src <-img.src;
            if !img.show and inVisibleArea() then //没有展示过并要到达可视区域
              if !!that.changeSrc then
                  src <-that.changeSrc(img.el, src, gifIdx);
              endif
              function onload() {
                  forceHeight <- t.getAttribute("data-forceheight");
                  if forceHeight then
                      setHeightAttributeForce()
                  else
                      removeHeightAttribute()
                  endif
                   if t.getAttribute('_width') then
                       setPropertyWidth()
                   else
                       removePropertyWidth()
                   endif
                   !!that.onload && (that.onload(img.el.src, t))
                   attr(img.el, 'src', src)
                   loadList.push(src)
                   img.show <- true
                   setProperty(img.el, "visibility", "visible", "important")
             }
        end for
        //回调，让使用者知道在什么时间段加载的图片列表
        if loadList.length > 0 and !!this.detect) then
            this.detect({ time: time, loadList: loadList, scrollTop: scrollTop })
        endif
}
```

图 4.4 图文展示模块 detect 方法定义

一张普通 jpg、png 格式的图片大小大概几 MB，动图大概要几十 MB，带有图片的文章每次请求量都会达到几百甚至几千兆，对带宽的要求很高。为了减少图片 cdn 带宽，系统采用 HEVC 编码对图片进行处理，据统计图片请求的带宽会相对于原先减少 30%-40%左右。整个项目组所做的工作是：视频组升级 hevc 的编码解析库，cdn 更新 hevc 编码库，x5 内核更新 hevc 的解码库，安卓客户端更新 x5 的内核，iOS 更新图片代理中的 hevc 解析库。前端对图片请求链接处理，最终在 iOS 和 Andriod 的图文系统图片请求中都能走 hevc 解码库，达到减少带宽的目的。由于手机系统、客户端版本、系统内核版本的不同，在设置图片编码方式时需要做兼容性处理。处理的流程如下图 4.5 所示：

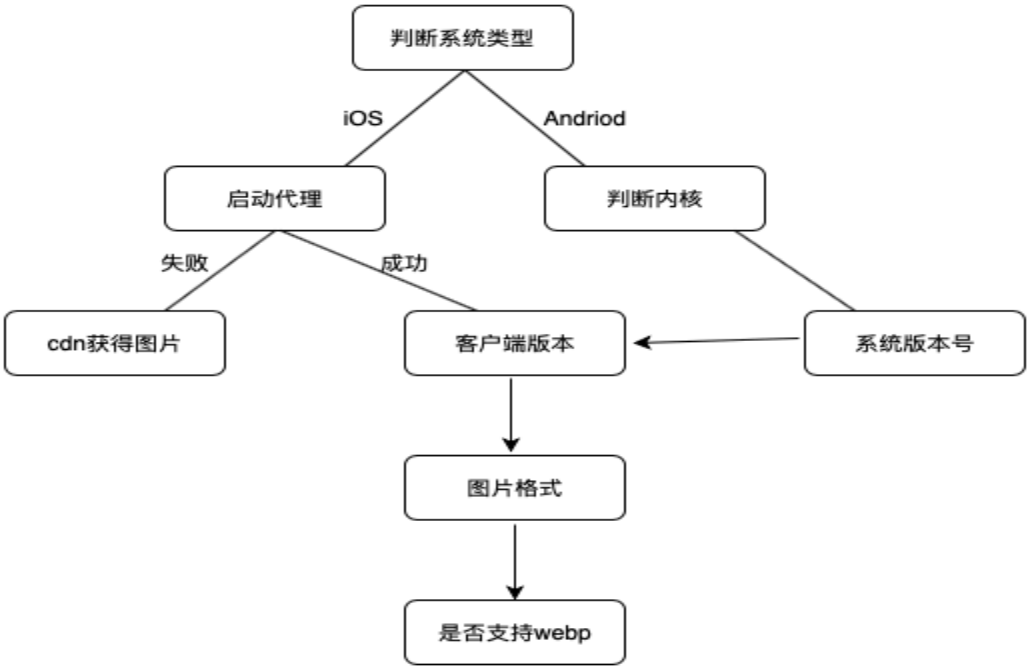

图 4.5 图片编码处理算法流程图

首先根据 Navigator.userAgent 判断手机的系统类型，区分是 iOS 或 Andriod 系统，用不同的方式对图片进行编码处理。当为 iOS 系统时，判断是否可以走客户端代理访问图片链接。当获得客户端服务器的地址成功时，图片链接前缀使用服务端的地址，失败时，直接从 cdn 获取图片。判断客户端版本，当客户端版本小于 6.5.13 时，系统未实现 HEVC 编码，通过图片的 base64 编码，通过判断 lossy、lossless、alpha、animation 等编码参数判断是否支持 WebP 编码，不支持时，走普通编码格式。当客户端的版本在 6.5.13 到 6.8.0 时，系统实现了 gif 格式图片支持 HEVC 编码，其他格式图片按是否支持 webp 分别处理。在大于

6.8.0 客户端版本中，jpg、png、gif 等格式图片均支持 HEVC 编码。当为 Andriod 系统时，微信客户端的内核有 xweb 和 x5 两种，目前在安卓客户端的覆盖率大约是 xweb 占有 10%，其余为 x5 内核。xweb 版本大于 359 以上时，支持 jpg、png、gif 格式的图文进行 HEVC 编码。xweb 版本大于 16 以及 x5 的内核的版本大于 43305 时，对于 gif 格式图片支持 HEVC 编码。前端的处理方式是在发送图片请求时在图片的链接加上相应的参数，WebP 编码加上 wx=webp 的参数，HEVC 加上 wx=wxpic 的参数。当客户端接收到带有该表示位的请求，从客户端到内核到底层视频编码库依次对 WebP 和 HEVC 编码进行处理，最终返回编码后的图片。

在微信公众平台中每天都有都几百万的图片上传量，考虑到节省带宽和流量一般会对图片进行压缩，而且有些订阅号文章的图片来自于第三方网站，本身就不是很清晰，为了用户的阅读体验，使用户能够看到清晰的图片，使用超分算法对图片进行超分，还原图片的清晰度。在第二章的超分技术概述中对业界的超分技术进行的论述，在图文系统中使用 PyramidNet 模型实现超分算法。一般的超分算法直接处理的是低分辨率的图片进行超分辨放大，默认图片是没有经过压缩的，然而在公众平台上传的图片如上节描述大部分都是经过压缩的。PyramidNet 模型将超分分为 2 个步骤，去模糊和超分辨率放大，超分部分使用主流的基于神经网络的 CNN 超分算法。在模型训练的过程中，实现了超分小工具项目，用于比较不同模型的超分效果。该工具支持上传文件或者文件夹图片，选择处理的模型和相应的超分放大倍数，到服务器上训练模型，以图表的形式展示模型的超分效果，支持查看原图。超分小工具对促进超分算法的实现并且应用到公众号图文系统中起了很大的促进作用。

4.2.2 视频播放器的实现

在开始实现图文展示模块时，直接使用腾讯视频播放器。然而公众号图文系统对于视频播放有特殊的业务需求，系统需要支持在视频的片头插入广告主的广告。并且在之前视频播放器的网络带宽耗费非常大，因此实现了一套视频播放器，而拉取视频封面、时长、链接等信息还是用的腾讯视频播放器的接口。流程大致是首先用腾讯视频的接口获取封面、时长等信息，并且在文章展示时显示出封面

和时长，提供开始播放的按钮。当用户点击播放时，加载视频头，当网络失败、接口调用失败、视频头加载失败时，提示加载失败，可重新加载，播放中的视频可暂时、终止和继续播放。

视频播放带宽浪费是由于在浏览器中对视频的加载没有限速策略，当在页面中嵌入 video 标签后，会一直加载视频直到视频加载结束。但是据公众平台统计，大部分的用户会看到 60%的进度就会关闭视频，因此有 40%的带宽是被浪费的。在实现时，使用 proxy 策略控制视频流量。在本地启用一个 proxy，视频的连接请求会经过该 proxy，向外网请求资源。开始 proxy 只会请求最多 60s 的视频，之后通过客户端 JSAPI 方法同步当前视频的播放进度来控制视频流量的获取，以缓慢的速度请求视频流量。

因为业务需求需要在视频播放器中插入视频广告，视频广告的播放和普通视频的播放流程一致。通过插件管理机制，将视频广告和 proxy 作为插件，不影响视频播放器的主流程，通过调度器，对视频广告、proxy 进行调度，控制各个插件之间的状态流转。当用户点击开始播放视频，需要播放片头广告时，插件队列第一个插件接收到开始消息，中断状态避免继续流转，系统去拉取广告的内容并播放，直到获得广告内容再进行状态的流转，通知播放器播放。当广告正常播放到结束时，浏览器的内核触发结束状态，广告插件中断后，重新派发一个 begin 的通知告知播放器播放主视频。这种插件机制，不影响主播放器的流程，可以扩展增加比如片尾广告、弹幕等功能，并不会影响主播放器的代码实现。

图文系统中视频播放器最关键的部分是主播放器的实现。在图文中的视频初始化先对视频进行占位，实现方式和图片占位一致，在获得实际宽度和高度后对视频进行替换。系统用状态机对视频的状态进行管理，状态机共有 6 种状态，init 初始态、play 播放中、pause 暂停、loading 加载中、end 结束、结束状态。状态机除了这 6 种父状态，还有更细分的 8 种子状态，init 初始态(和父状态一致)、play 状态(视频播放的时候触发)、playing 状态(视频播放的时候触发此时父状态为 play)、waiting|stalled|seeking|seeked 状态(父状态都为 loading 状态)、preload 状态是(在用户第一次点击视频播放时 play 或 playing 之前的触发状态，父状态为 loading)，使用状态机对视频播放器的状态扭转进行控制。视频播放的实现的流程如下：

1. 初始化视频所在的 Dom，当存在多个视频时获得第一个 video。
2. 初始化 video 的数据属性，hasBeginPlay 属性标识是否已经开始播放、canPlay 属性标识是否可以播放、hasScroll 属性标识页面是否有滑动、replaySec 属性标识重复播放的时间。
3. 初始化 player，监听 contextMenu 事件，阻止冒泡和原生的右键事件。去除进度条，设置视频的大小适应于移动设备。根据初始化播放器的参数，设置视频是否重复播放或者静音。
4. 获取视频链接，当视频链接为空，触发错误状态并提示。
5. 初始化片头广告、本地代理等插件，触发 loading 状态。
6. Loading 状态展示菊花旋转按钮，触发 ready 状态。
7. Ready 状态真正设置视频连链接，重新进行初始化，展示封面信息，并且设置进度条的信息，触发 loaded 状态。
8. 如果视频设置了自动播放，触发 readyBeginPlay 方法，如果没有，隐藏加载图标。
9. 调用 readyBeginPlay 方法，若没有设置为 src 则进行设置，并且触发 beginPlay 方法，展示第一次播放的菊花按钮，设置 hasBeginPlay 状态为 true，调用 video 的 play 方法。
10. 绑定按钮和播放条事件，绑定 video 的各种事件，监听 postMessage 的消息。

当图文系统在微信客户端内使用时提供浮窗功能，当用户将文章收进浮窗内时，系统可以暂存正在阅读的文章，并且将正在播放的视频暂停播放，当用户再点击浮窗打开文章时，系统将文章滚动到之前的位置，继续播放原先视频。该功能的实现是通过监听页面的 onVisibilityChange 事件，当页面隐藏时，遍历页面中所有视频判断它们所处的状态，利用 hasBeginPlay 和 isPlay 方法，判断是否已经开始播放并且正在播放，暂停视频，缓存当前正在播放的视频。当页面显示时，获得之前缓存的暂停播放的视频，如果之前的视频已经开始播放并且没有播放完成，调用 play 方法继续播放该视频，并清除缓存的正在播放的视频。

4.2.3 小程序展示实现

公众号作者在编辑文章时可以插入小程序，小程序应用可以以文字、图片、卡片、小程序码的形式展示在文章中。当订阅者点击时文章中的小程序，系统可以调起小程序并给用户使用。系统通过 html5 和客户端协同实现这一功能。

微信公众号图文系统中小程序应用和小程序商品展示的实现都基于 weapp_common 这一公共基础类。weapp_common 类首先根据 Navigator.usergent 判断用户是否在微信客户端中，若不在微信客户端中则提示用户在微信中打开小程序。系统同时可以根据 useragent 获得微信的版本，6.5.3 以上的微信版本，是支持调起小程序的，以下的不支持，会提示用户更新微信版本。之后 weapp_common 类获取文章中所有嵌入的小程序的 appid(appid 是小程序的唯一标识)，在请求的过程中分别记录下失败、成功但回包解析失败、成功回包解析也成功的场景。系统使用重试策略对请求 appid 失败的小程序做重试，最多重试三次。然后该公共类使用 batch_get_weapp 接口获取小程序的信息，在调用该接口前，需要对之前保存的文中的所有小程序的 appid 做去重处理，防止同一页面中请求的 appid 过多，导致 URL 长度超过 HTTPS 协议的上限，最终导致页面产生 414 错误。系统调用接口的过程是将所有的 appid 拼接到请求的 URL 上，发送 ajax 请求到服务器端获取每个请求的小程序信息。客户端方面提供给外界获得小程序的跳转链接、封面、图片等信息的 JSAPI 为 openWeApp，因此图文系统使用该接口对小程序链接执行跳转，当关闭小程序时调用 waerrpage 接口返回公众号文章页。

Weapp 类是小程序的真正实现类，组合了 weapp_common 类。weapp_common 类负责获取小程序 appid、当用户点击小程序时执行跳转等功能。Weapp 类中在初始状态下，使用 init 函数对页面进行渲染操作，在函数中创建元素、设置类名调整样式、插入元素等。初始化时插入小程序并获得小程序相关信息，对小程序取消和确认添加监听事件。同时监听用户的鼠标悬停事件，当鼠标悬停时显示小程序的二维码，设置显示二维码的位置和需要悬停的时间。在实现时对图片类的小程序的点击事件，改成监听 tap 动作，防止访问触发图片的点击事件，显示图片的原图。图文系统中小程序显示类 weapp 的实现代码如图 4.6 所示：

```
Class weapp {
    // 初始化属性
    render()
    initSnMap()
    function render() {
        for i <- 0-newWeappElems.length+oldWeappElems do
            isNewElem <- i < newWeappElems.length
            // 获取元素、appId、path、imageURL、title 等信息
            var displayElem = document.createElement('span')
            // 设置元素样式
            … 加入元素数组，将 dom 元素插入到页面中
        end for
    }
   function initSnMap() { // 根据相关信息拉取小程序信息
        … 监听点击和取消事件
        // 不支持 Tap 跳转或不在微信应用中时显示小程序二维码
        If isCodeShow then
            … 监听鼠标对于小程序二维码的移入与移出事件
        End if
        WeappCommon.getAppidInfo({
            onSuccess: function (opt) {
                var appidInfoMap = opt.data.infoMap
                for i = 0-weappDisplayElemArr.length do
                    //定义 appid、path、imageURL、title、info、imgLink、tag 等变量
                    if tag != 'a' then
                        displayElem.innerHTML = setImage、title、nickname、avatar
                    else if !isImgLink then
                        ClassList.addClass(displayElem, 'weapp_text_link')
                    else
                        setImgLinkStyle()
                        displayElem.setAttribute('href', '')
                    end if
                    …改成 tap 跳转，可以避免跟客户端长按图片产生冲突
                })
        }
    }
}
```

图 4.6 weapp 类代码

Weapp 类主体是两个方法，render 和 initSnMap 方法，render 方法负责初始化页面中的小程序 appid、路径 path、图片链接 imageUrl、标题 title 等属性，在页面中创建 span 标签，设置类 class 调整显示小程序的位置，页面全局缓存需要展示的小程序元素和当页面滚动时需要调整的元素。initSnMap 负责拉取小

程序相关信息,初始化需要上报的数组。当小程序窗口确认时，调用成功的回调函数。当微信不支持小程序点击时或者不在微信客户端中时，显示小程序二维码。主方法放在 weapp_comom 类的获得小程序信息 getAppidInfo 的回调成功的函数中，区分图片、文字，分别加上对应的样式。在安卓中显示时，将元素的 href 设置为空，安卓长按图片时才会出现原生菜单。安卓下 a 标签包含图片时，A 标签 href 要么是空，要么有链接，长按图片功能才生效。对小程序增加 tap 事件而不是 click 事件以防与图片的 click 事件冲突。调用 weapp_common 的 jumpURL 方法，jumpURL 方法中调用 JSAPI 提供的打开小程序应用的方法 openWeApp。同时 initSnMap 方法也针对当文章在浏览器中显示时，对小程序二维码的位置做特殊处理。

4.2.4 图文展示模块页面截图

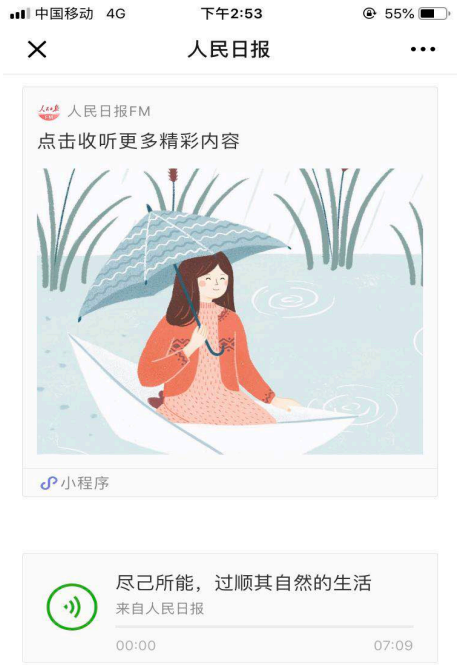

图 4.7 图文展示模块页面截图

如上图 4.7 所示，主要截取的是图文展示模块中小程序卡片和音频的展示界面。小程序卡片以图片和说明文字的形式展示，当用户点击卡片时，会自动调起人民日报 FM 小程序，当用户关闭时，回到公众号的文章。小程序卡片下面是音频，显示音频的标题、说明和音频总时长，具有暂停和继续播放的功能。文章中还可以展示图片、视频、商品、卡券、投票等，不一一截图说明，在订阅号消息系统内可体验。

4.3 图文互动模块的实现

4.3.1 互动模块登录态实现

在图文系统中，图文的展示属于主流程，用户首先关心的是文章能否正常展示。系统在提供图文展示的基础上，提供互动功能比如点赞\在看、评论、赞赏、广告等。图文系统寄生于微信应用内，在使用图文系统的相关功能时，无需用户再次登录，直接使用已在微信内登陆用户的信息。在实现时，系统是基于geta8key 技术，来获取用微信用户的登录态信息的。

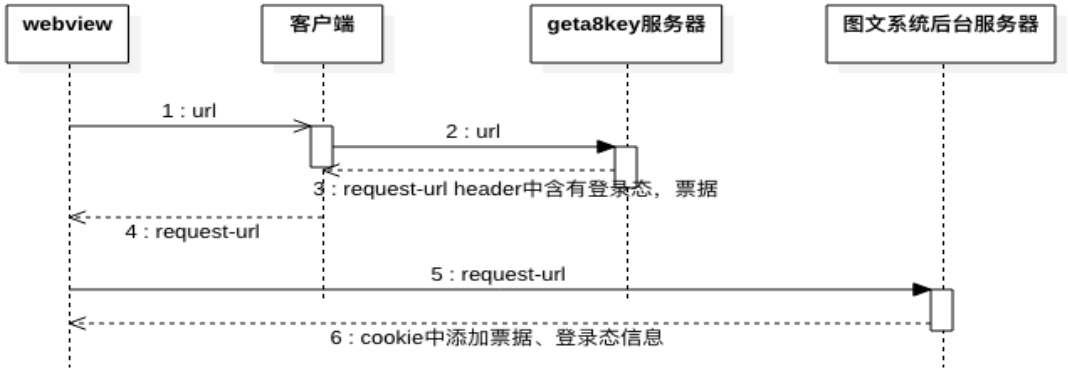

图 4.8 图文互动模块获取登录态流程

如上图 4.8 所示，以时序图的形式描述了系统获取登录态的大致流程。当webview 向客户端发送加载 url 的请求后，客户端转发该请求到后台 geta8key服务器。Geta8key 服务器判断是否下发登录态信息，在这一层可以拦截一些比如淘宝链接的 url。geta8key 微信后台，根据请求的域名判断调用哪一业务的geta8key 服务器，比如微信公众平台域名为 mp.weixin.qq.com，这种类型的链接会被转发给开放平台的 geta8key 服务器。服务器处理完成后返回用于请求的request-url，request-url 用于真正向后台发送请求，带有用户客户端版本、语言、还有 ticket。Ticket 是个人微信号的票据，后端对其校验后，可以获得用户信息，如：用户昵称、头像等。客户端向后台发送请求的连接中，会带有登录态信息，服务器端收到请求，处理完业务逻辑后，会下发票据和登录态信息在返回的cookie 中。在后续的页面请求中，cookie 中都会带有票据和登录态信息，后台会进行校验。

互动模块的加载通过 ajax 请求 getappmsgext 方法异步拉取，需要传递 mid、sn、idx 标识唯一图文的三元组，version、devicetype、pass_ticket 等 20 多个必须参数。每次 getappmsgext 的请求会根据 id 进行上报，用于监控图文互动模块的正常显示。当异步请求图文信息成功回调后，系统根据请求获得的阅读数、在看数、评论消息数，对需要显示的信息初始化操作比如初始化投票、卡券，加载广告。图文系统基于获取的用户的登录态信息，向用户提供互动功能。其中分享、投诉、在看功能的截图如图 4.9 所示：

图 4.9 部分互动功能截图

4.3.2 图文赞/在看功能的实现

公众号图文系统改版之前用户可对订阅号发布的文章进行点赞，点赞的状态只有用户自己可见，表示对文章的认可。在实现时，在发送的请求的同时改变用户所见的文章的点赞状态，因为在这种情况下用户不会特别关心实际的点赞效果，而关心目前的状态是否变化。为了尽可能保证请求发送成功，对点赞或取消赞的请求增加重试策略，将原先的 ajax 类封装为 retry_ajax 类。对未发送成功或由于网络等特殊原因失败的请求放入重新请求的队列中，对队列中的失败请求进行重试，最多重试 3 次，3 次后失败即不进行处理。若在之后的请求发送成功，则覆盖原先失败同类的请求。

图文系统为了提供更强的社交性，将点赞功能升级为在看，系统推送用户在看的文章到看一看应用中，朋友可以在看一看中看到，并且用户可以评论和为好友点赞。当用户在图文系统中点了在看时，后台异步队列将用户的点赞请求记录下来。在看一看中需要同步了解用户发送的请求。在实现时有两种选择，一种方式是看一看方主动拉取，一种是客户端对看一看发送状态提醒。最终考虑认为发送状态提醒这种方式更适合，因为这种方式下，看一看不需要一直轮询查看是否有新消息，每一次的主动拉取也会影响性能。最终实现的流程是，使用微信客户端提供新的 api 为 handleHaokanAction，当用户在文章中点击在看或者补充相应的评论时，调用 jsapi 相应的方法传递是否在看的标志位和评论数据，还有透传给看一看的数据，包括文章的链接和标题。客户端扮演的是透传的角色，将图文系统的数据透传给看一看。客户端通过事件通知看一看，看一看监听已定义的事件，接收透传的数据并刷新模板，当有新的消息时在客户端红点提醒。

在最初设计时，项目组无法区分是提供点在看加评论同时进行还是直接点在看再评论哪种方式对用户更友好。在看功能实现时，为了对比不同方式的效果，根据用户 uin 进行区分，进行了 A/B test。当用户的版本低于 7.0 时，看一看中没有好友在看的功能，系统不提供在看的功能，仍使用旧版本的点赞功能。后台处理用户 uin，用户 uin%100 为奇数时，走 A 方案，用户可以直接推荐文章在看，稍后再对文章添加推荐理由或者不补充评论。当 uin%100 为偶数时，走 B 方案，在推荐在看的时候必须补充推荐的看法，但是看法可以为空。在异步请求图文信息的接口 getappmsgext 中增加 mmversion 参数，该参数提供给后台判断用户是否是旧版本的微信，走旧方案。在 getappmsgext 接口增加返回，是否出现首次教育弹窗 prompted、用户方案 style、是否对用户禁用在看功能 like_disabled 等属性。在初始时，每个用户的 prompted 参数都是 0，当用户选择文章在看后，会提醒用户，推荐的文章将在看一看中被好友看到。当用户确认推荐后，向后台发送在看请求，并将 promted 置为 1。用户同样可以选择取消，点击的在看并不会被发送到后台，下次再点击时仍会出现教育页。

在实现在看功能展示的过程遇到的有两个主要问题：1.滚动透传问题。2.评论框光标错位问题。滚动透传指的是在移动端，当在页面上弹出评论窗口，在评论窗口内滚动时，会导致背后的文章也跟随着滚动。这是移动端本身的问题，为

了解决这一问题，对文章中的滚动事件分别处理。在弹出评论窗口后，监听页面的 touchmove 事件，当来源为除了 textarea 和 button 以外的元素时，阻止默认的滚动事件。监听评论 textarea 区域的 touchstart 和 touchmove 事件，记录手指滑动的初始位置，在手机滑动到评论框最顶端或最底端时，阻止默认事件，在滚动区域内时不做处理。滚动透传问题的解决方式代码如图 4.10 所示：

```
var disableMove = function() {
        document.addEventListener('touchmove', preventMove, { passive: false });
        el_blikeCommentTextSecond.addEventListener('touchstart',getTouchStart,
          { passive: false});
        el_blikeCommentTextSecond.addEventListener('touchmove', preventText, false);
}; // 监听整个文章和评论区域
var enableMove = function() { // 取消监听}
var preventMove = function(e) {
        var target = e.target;
        if (target.tagName === 'TEXTAREA' || target.tagName === 'BUTTON') {
              return;
        } else {
               … 阻止默认事件
        }
}
var getTouchStart = function(e) {
        var targetTouches = e.targetTouches || [];
        if (targetTouches.length > 0) {
              var touch = targetTouches[0] || {};
              startY = touch.clientY;
        }
}
var preventText = function(e) {
         var canMove = false;
         … 获得当前 scrollHeight、scrollTop 等参数
         if (changedTouches.length > 0) {
            var touch = changedTouches[0] || {};
            var moveY = touch.clientY;
            if (moveY > startY && scrollTop <= 0) {
               canMove = false;
            } else if (moveY < startY && scrollTop + offsetHeight >= scrollHeight) {
               canMove = false;
            } else {
               canMove = true;
            }…}}
```

图 4.10 滚动透传问题解决代码

为了给用户更好的体验，在用户选择填写观点时，利用输入框的自动获得焦点 focus 方法自动调起手机键盘。在调起的过程中，如果是 fixed 定位的元素，在 iOS 的某一版本下会存在光标错位的问题。因此在设计样式的时候，在正常状态下为 fixed 定位元素，在调起键盘的时候，更改输入框的定位为绝对定位，并且设置其他的样式来解决这一问题。

为了对比 A/B 方案，对每个环节增加上报，在处理在看请求时，增加 action_type 参数，区分是单纯点赞、在看、评论、在看+评论其中哪种行为，通过量级数据观察对比哪种方案的使用人数更多，最终各项数据都表明 A 方案更友好，最终保留 A 方案。在看功能是在公众号新推出的互动功能，还在需要不断的优化改进。

4.4 图文秒开模块的实现

4.4.1 前端编译实现

图文秒开模块设计中最关键的是前端编译、首屏渲染、客户端离线化处理。在一般场景下，订阅号图文系统的实现方案是向后台请求已经编译完的页面，是一种后台直出模式。后台将获取到文章的页面模板和请求文章的数据，通过 QQMail 模板引擎编译返回一个完整的 html 页面。因为每篇图文都是请求的同一份页面模板，只是每篇文章的数据不同，因此在秒开场景下，在客户端缓存页面模板，向后台获取数据后，在前端编译，这种方式能够减少每次请求模板的时间。前端编译简单而言，就是用前端的语言处理原先在 html 代码中 QQMail 的模板语法，和后台请求的数据结合起来。不可否认 QQMail 模板引擎具有很强的性能优势，但是这会导致前端开发依赖于后台的模板渲染。在秒开中使用 QmTpl 模板引擎，实际上是将后台的模板引擎搬到前端。

QmTpl 的解析过程如下，首先构造 QmTpl 类对象，扫一遍模板，匹配其中的<%..%>和$..$两种类型的标签。语法<%..%>主要有三种类型，分别是控制语句<%@..%>、注释或块级标签<%##..##%><%#..%><%#/..%>、作用域语句<%xxx%> <%/xxx%>。控制语句又分成两类，分别是过程控制、内置函数，其中过程控制包括 if、elseif、endif、while、endwhile，内置函数包括加、减、乘、

除、日期等。模板解析时可能存在嵌套语句和表达式的情况，所以需要递归解析，当遇到注释和块级标签直接去掉，不加以处理。作用域语句主要用于限定作用域，故 QmTpl 维持了一个作用域栈，使用闭包形成相应的作用域，并通过取值函数获得作用域对应数据对象，其中作用域保留字 top 和 parent 表示返回顶层和上一层作用域。表达式$...$，只有一种，作用就是获取相应的值，在 QQMail 模板语法规则里，数据类型只有一种，那就是字符串。解析完成后，会形成 function string，通过 new Function 构造渲染函数，并缓存起来，当传入数据时渲染出 HTML，从而做到一次解析、多次渲染。

相比于之前的模板引擎，由于多次进行全局正则匹配，效率很低，而且没有生成渲染函数，都是在模板和数据同时传入时才进行解析渲染，对于二次渲染性能没有任何提升。对于图文消息模板，平均解析渲染耗时约为 80ms 左右。而 QmTpl 由于只扫描一遍模板，所以解析时效率会高一些。同时图文秒开中存在预加载的情况，在预加载时先对模板进行解析，等数据传入时才会进行渲染。由于主要耗时都在模板解析上，所以渲染时间大大减少，平均耗时仅为 12ms。

4.4.2 首屏渲染实现

文章页面渲染到用户看到整个过程，会发生前端拿到数据、解析 json 字符串、将数据和 HTML 进行合并等操作，在到浏览器显示模板的时间内会发生白屏。从上报的测速数据表示，前端拿到数据耗费 2ms、解析 json 字符串大概需要 12ms，前端页面渲染耗费大概 418ms，可以发现最耗时的是前端页面的渲染流程。因为用户最先看到的是文章首屏的内容，可以先渲染首屏的内容，非首屏的内容可以延后渲染。

HTML 的渲染是以 DOM 树即页面的标签节点树为模型渲染，浏览器有专门的 HTML 解析器解析 HTML 文档，在解析的过程中创建 DOM 树，创建完毕后进行渲染。公众号文章的 HTML 的结构是固定，文章的具体内容会放置在固定 id 的 dom 元素中，后台返回带有标签的文章数据 content_noencode 中。文档顶层是 html、body 标签，内部为 div、p、span 等元素，展示在首屏的可能只是 html 文档中的前一部分的节点。渲染过程从父节点开始，不断向下渲染，直到最内层子节点。因此可以将前一部分的元素先渲染，先渲染首屏的内容，用户可以

先看到一部分，再渲染其他的内容，这种机制称为分段渲染。系统挟持后台返回未编码的文章内容数据，先不渲染在页面中，对文章内容的 dom 元素进行处理。通过创建 documentFragement(文档片段)，因为其是存储在内存里，在所有操作完成后被插入到真正的页面 DOM 树中，当其内部的元素发生变化时，不会引起回流，能够提升性能。具体实现是对页面节点采用前序链式遍历，封装为 Appender 添加类，实现两个原型方法 moveFirst、moveRest，分别负责首次节点插入和剩余节点插入。通过多次尝试，暂时定为首次渲染 20 个深度轮次是局部最优方案，在系统中添加上报信息后观察再进行调整具体的数值，一个深度轮次是一次前序遍历到叶子节点。moveFirst 方法如图 4.11 所示：

```
Appender.prototype.moveFirst = function () {
  … 声明变量
  while (true) {
    if (!this._start) {
      this._start = origin;}
    if (!origin.visited) {
      origin.visited = true;
      if (!root) {
        root = document.createDocumentFragment();
        origin.twin = root;
      } else {
        origin.twin = origin.node.cloneNode();
        frag = origin.par.twin;
        frag.appendChild(origin.twin);
      }
      if (this._isHidden(origin.node)) {
        …节点不可见，不遍历子节点，直接添加
      } else { … 遍历子节点}
   }
    … 遍历兄弟节点
    origin = origin.par;   // 回溯父节点
    if (origin === this._start || !origin) {
      this._finish = true;
      break;
    }
  }
  … 添加节点到页面中
};
```

图 4.11 moveFirst 方法代码

moveFirst 方法主体是一个 while 循环，循环的终止条件是到达叶节点的次数大于 20 次。初始时将开始节点赋值为 origin 元素，判断开始节点是否访问过，未访问过时将 visited 标志位设为 true。当节点不可见时，不遍历子节点，直接添加整个节点，否则遍历子节点，并且判断到达叶节点的次数是否到限制，再遍历兄弟节点。回溯父节点，如果已经全部遍历完成后，则不需要再添加剩下的节点，执行 moveRest 方法了。

为了提高用户的阅读体验，系统还提供记录用户上次阅读位置的功能，因此首屏渲染也需要针对回到上次阅读位置进行优化。每渲染一部分 DOM，判断是否可以设置到上一次的位置，一直渲染直到可以设置到上一次阅读位置。设置上一次阅读位置后，遍历图片，加载已经曝光过的图片，这种方式同时可以解决图片加载慢的问题。

4.4.3 秒开通用化

在对图文系统改造时，只针对从订阅号消息进入的阅读场景进行了图文秒开改造。微信公众号图文系统需要展示的除了图文文章，还有视频分享页、音频分享页、图片分享页、文字分享页等，页面体验上需要做到一致，因此系统针对图文页的秒开设计进行通用化。

用框架对秒开方案进行管理，页面改造的流程包括：打包页面所有 HTML 代码，打包页面所有 JS\CSS 代码，对逻辑代码进行改造(数据回调逻辑、前端编译逻辑、登录态回调逻辑、预加载 JS 代码调用等)。页面前端文件的打包流程可以通过提供编译构建工具来自动处理，使用 Json 配置文件，将页面中的 HTML\JS\CSS 和依赖的文件打包成目标文件。

```
function SkeletonTpl(lifecycles) {
  // 如果需要更多插入点，可以添加
  this.initSkeleton = lifecycles.initSkeleton; // 模板初始化逻辑插入点
  this.receivePageData = lifecycles.receivePageData; // 获取到数据处理逻辑插入点
  this.getPageAuth = lifecycles.getPageAuth; // 获取到登录态处理逻辑插入点
  // 暴露外部方法
  this.init = this.init.bind(this);
  this.runMainJs = this.runMainJs.bind(this);
  this.notifyApp = this.notifyApp.bind(this);}
```

图 4.12 skelonTpl 秒开框架代码

如上图 4.12 所示，在改造逻辑代码时，提供一个小型的框架 skelonTpl。skelonTpl 框架提供数据回调、登录态回调、数据代码执行逻辑、上报函数逻辑、错误处理逻辑和其他逻辑。对需要接入秒开的页面，首先进行全局配置，再接入底层逻辑处理框架即可。

4.4.4 秒开方案对比及效果

将公众号图文系统的秒开与百度 app 的秒开方案进行了对比，除了前端常用的优化方案如图片 base64 编码处理、缓存、减少页面请求数之外，共同特点是都做了离线化处理，将页面文件离线存储在客户端，并且将不在首屏的内容异步化处理。但是公众号图文系统是一个依赖于微信的项目，和百度 app 有一些区别，因此再将该方案与同样在微信内的微信游戏系统做对比。

相比于微信游戏方案，公众号图文系统的秒开方案有几个优势。第一：图文系统将所有文件打包成一个文件，所以页面内是没有任何请求的，而微信游戏的方案，他们是使用了一个 SDK 来拦截页面内的全部请求，比较耗时。第二：图文系统是在点击文章的时候可以拉取数据，而微信游戏的方案依赖于 js 做异步请求拉取数据，也会耗费时间。第三：图文系统的秒开页面是和 H5 页面共用一套发布系统部署页面的，微信游戏需要和 H5 分开部署，增加了部署和运营的成本。方案对比表下表 4.1 所示：

表 4.1 秒开方案对比表

公众号图文系统	微信游戏
Webview 内没有拉取文件请求	Webview 内请求用 sdk 拦截
点击文章时拉取数据	根据拦截的 JS 再异步请求
与 HTML 共同一套发布系统	与 HTML5 分开发布

将公众号图文 H5 页面与秒开耗时进行对比，秒开平均耗时约 450ms，HTML5 的首屏耗时约 950ms，秒开的首屏时间提升了一倍左右。图文 HTML5 首屏耗时和秒开首屏耗时的对比图如图 4.13 和 4.14 所示：

图 4.13 公众号图文 HTML5 首屏时间

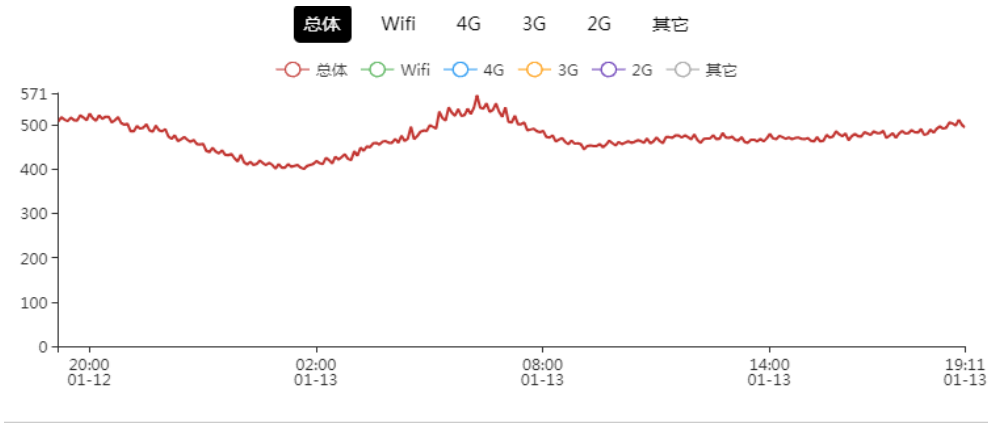

图 4.14 公众号图文秒开首屏时间

图文秒开所耗费可交互时间也减少了很多，图文 H5 的可交互时间耗时约 2500ms，秒开的可交互耗时约 1250ms。无论从首屏时间、可交互时间，还是加载首张图片的时间，图文秒开的设计对提升图文页面的打开速度做出了很大的提升，并且从数据上看，秒开对用户打开 pv 提升了约 4.5%，文章的阅读数上涨了 4.5%。图文 HTML5 用户可交互耗时与秒开可交互耗时的对比图如图 4.15 和 4.16 所示：

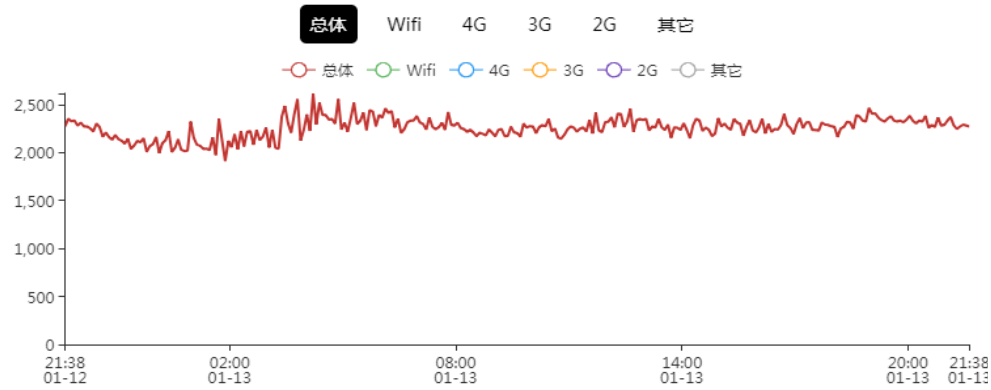

图 4.15 公众号图文 H5 可交互时间

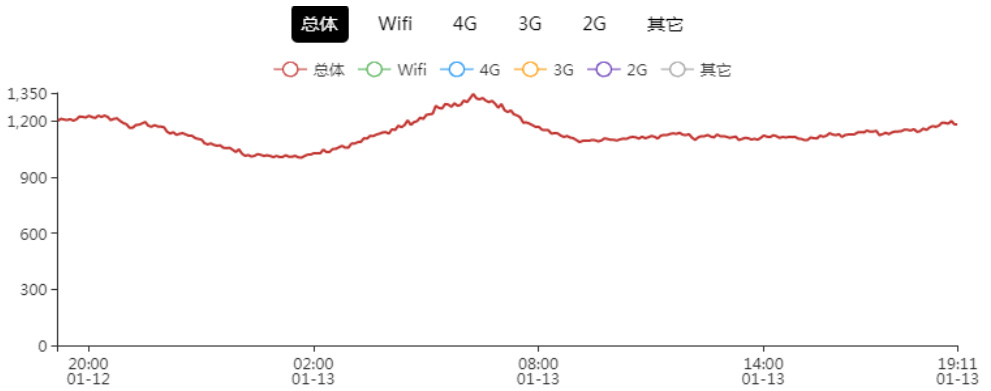

图 4.16 公众号图文秒开可交互时间

4.5 本章小结

文章主要描述了公众号图文系统中图文推送模块、图文展示模块、图文互动模块、图文秒开模块的实现，通过代码图、算法图对各个模块进行了详细说明，通过界面截图说明每个模块所实现的功能，通过上报系统的测速图说明模块对系统带来的优化效果。

第五章 总结和展望

5.1 总结

微信公众号图文系统是一个自媒体平台，在满足基本的阅读功能的基础上，通过图文秒开等方式优化读者的阅读体验。系统通过提供投诉维权手段，维护良好的自媒体环境，鼓励原创作者写原创文章，因而能够推送给读者更优质的文章。系统提供互动能力，读者能与朋友分享交流，使阅读更加有趣。本文首先介绍了微信公众号图文系统的项目背景，分析了类似于公众号的自媒体平台的研究现状。

然后，介绍了图文系统混合式开发技术，图文系统前端部分所用到的 SeaJS 模块化管理框架和用 C++语言编写的后台直出模板引擎 QQMail。阐述了图文系统中为了优化图片阅读体验所用到的超分辨率重建技术和新型的编码方式 WebP、HEVC。介绍了在实现图文秒开时参考的 React Fiber 架构的思想，同时还介绍了后台服务框架 svrkit。

随后，对微信公众号图文系统进行的概述，分析了公众号图文系统的需求，以用例图和表格的形式展示需求。在需求分析的基础上，对系统进行了总体设计和模块设计，通过架构图和模块图的形式说明整个系统的总体设计。对系统分成了图文推送、图文展示、图文互动、图文秒开模块，阐述了每个模块，以类图、时序图等形式对主要模块进行了详细设计和说明。

最后，在需求分析和设计的基础上，用详细代码和算法图的形式说明了图文系统中重要功能的实现，以界面截图和测速上报数据截图的方式说明系统中采用的实现方式带来的效果。描述了图文推送模块中的消息推送功能的实现，使用户能够及时、不丢包的收到最新的图文消息。重点说明了图文展示模块中展示图片、视频、小程序形式的实现方式，以多样化媒介展示文章，极大程序提升用户阅读的体验。并且介绍了系统如何提供给用户与他人互动的能力，阐述了新特性在看功能是如何实现的。说明了图文系统中统图文秒开的实现，提升文章的打开速度，促进的阅读文章数和打开人数的增加。

5.2 工作展望

微信公众号图文系统是一个比较成熟的项目，面对近些年不断涌现出的新兴自媒体平台，公众号图文系统在上一年度做了改版和许多改进，从推送文章方式上，到文章的阅读效果，到提供给读者的互动能力，到最大能力的维护原创作者的权益。目前图文秒开只是应用在从订阅号消息流中打开的场景，秒开通用化的框架已实现，需要继续普及到每个场景和每种类型的文章，提高每个场景的打开速度。文章互动在看功能推出已有一定的时间，需要不断观察上报的用户相关数据，继续改进和调整，以找到最能满足用户的互动形式，图文展示中各种类型的媒介展示方式也需要继续优化。刚推出的洗稿合议功能需要继续推行，鼓励更多优质的写作者参与洗稿合议团的评审工作，严格抵制洗稿合议行为。在技术上，为了提升前后端协同开发的效率，需要在小范围内试行 nodeLogicsvr 框架的使用，使用 nodejs 搭建逻辑层，再逐步推行到整个公众号图文系统。

参 考 文 献

[Amber, 2015] T. Amber, N. Cloud, *RequireJS*, Springer, 2015.

[Dong et al., 2014] C. Dong, C. C. Loy and K. He, Learning a deep convolutional network for image super-resolution[C], *European Conference on Computer Vision*. Springer, Cham, 2014: 184-199.

[Kim et al., 2015] J. Kim, J. K. Lee and K.M. Lee, *Accurate Image Super-Resolution Using Very Deep Convolutional Networks*[J]. 2015.

[Lian et al., 2012] L. Lian, W. Shilei, *Webp: A New Image Compression Format Based on VP8 Encoding*[J]. Microcontrollers & Embedded Systems, 2012.

[Sebmarkbage, 2016] Sebmarkbage, *Fiber Principles: Contributing To Fiber*, github, 2016.

[Si et al., 2016] Z. Si, K. Shen, *Research on the WebP Image Format*[J]. 2016.

[Sullivan et al., 2013] G. J. Sullivan, Fellow, *IEEE Overview of the High Efficiency Video Coding (HEVC) Standard*[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2013, 22(12):1649-1668.

[Sun et al., 2016] D. Sun, C. Guo and D. Zhu, Secure HybridApp: A detection method on the risk of privacy leakage in HTML5 hybrid applications based on dynamic taint tracking[C], *2016 2nd IEEE International Conference on Computer and Communications (ICCC). IEEE*, 2016.

[Tai et al., 2017] Y. Tai, J. Yang and X. Liu, Image super-resolution via deep recursive residual network[C], T*he IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 1(4).

[Yang et al., 2008] J. Yang, J. Wright and T. Huang, *Image super-resolution as sparse representation of raw image patches[J]*, 2008.

[阿立, 2016] 阿立，*WebP 图片文件从何而来[J]*，电脑爱好者，2016(15):48-49.

[adeyi, 2013] adeyi，*前端 js 框架学习*，csdn，2013.

[程衍华, 2018] 程衍华，*PyramidNet-公众平台低质量图片超分技术解析*，技术报告，腾讯，2018.

[ChrisMurphy, 2017] https://www.cnblogs.com/ChrisMurphy/p/6550184.html, RPC 架构简单理解，ChrisMurphy，2017.

[冯新扬等, 2010] 冯新扬，沈建京，*REST 和 RPC:两种 Web 服务架构风格比较分析[J]*，小型微型计算机系统，2010，31(7).

[傅崇琛, 2018] 傅崇琛，https://mp.weixin.qq.com/s/uDIknJ-WeUJnPR8S-HnTww，React Fiber 初探，2018.

[古兰今, 2017] https://www.cnblogs.com/goloving/p/7712310.html ，seaJS 简介和完整实例，古兰今，2017.

[黄楚新等, 2015] 黄楚新，王丹，*微信公众号的现状、类型及发展趋势，新闻与写作*，2015(7):5-9.

[黄馨茹, 2017] 黄馨茹，*内容生产之变*，青年记者，2017(21):26-26.

[itprobile, 2014] https://www.cnblogs.com/guohu/p/3870677.html，各种 JS 模板引擎对比数据，cnblogs, 2014.

[江静等, 2012] 江静，张雪松，*图像超分辨率重建算法综述[J]*，红外技术，2012，34(1).

[姬一文等, 2011] 姬一文，吴庆波，杨沙洲，*一种服务器端模板引擎的改进与实现[J]*，计算机应用研究，2011，28(3).

[jieniyimiao, 2016] jieniyimiao，*微信与朋友圈后台架构*，技术报告，腾讯，2016.

[Keep789, 2018] Keep789，*Web App*、*Hybrid App*、*Native App 之间的差异*，技术报告，csdn，2018.

[廖秀秀等, 2012] 廖秀秀，韩国强，沃焱，*基于流形学习和梯度约束的图像超分辨率重建[J]*，华南理工大学学报（自然科学版），2012，40(4).

[李纪欣等, 2013] 李纪欣，王康，周立发，*Google Protobuf 在 Linux Socket 通讯中的应用[J]*，电脑开发与应用，2013(4).

[李明强, 2010] 李明强，*QQMail 模板引擎介绍*，技术报告，腾讯，2010.

[刘杰凤, 2017] https://infoq.cn/article/what-the-new-engine-of-react ，React 的新引擎—React Fiber 是什么？，刘杰凤，2017.

[刘佳等, 2006] 刘佳，卢显良，*小型高效模板引擎的设计与实现[J]*，计算机应用研究，2006，23(4).

[刘琚等, 2009] 刘琚，乔建苹，*基于学习的超分辨率重建技术[J]*，智能系统学报，2009，4(03):199-207.

[刘胜男, 2017] 刘胜男，*算法时代“好内容”的定义*，新闻与写作，2017(6).

[陆高峰, 2016] 陆高峰，*自媒体进入众“号”喧哗时代*，青年记者，2016(36):118-118.

[宋黎等, 2018] 宋黎，葛岩，*从“报网互动”到 Instant Articles——“新媒体与新媒体合作”取代“传统媒体与新媒体融合”*，西南民族大学学报(人文社科版)，2018，v.39；No.322(06):158-164.

[汤燕, 2016] 汤燕，*微信公众平台在高校就业服务工作中的现状及对策研究[J]*，经营管理者，2016(27).

[王建富, 2015] 王建富，*H.265/HEVC 编码加速算法研究[D]*，中国科学技术大学，博士论文，2015.

[徐荣飞, 2013] 徐荣飞，*HEVC 编解码算法的 CUDA 优化[D]*，北京邮电大学，硕士论文，2013.

[徐振, 2016] 徐振，http://blog.sina.com.cn/cnxuzhigang，H.265/HEVC

编码技术及画质对比，网易，2016.

[玉伯, 2012] https://www.zhihu.com/question/20342350 ， LABjs 、RequireJS、SeaJS，知乎，2012.

[又拍云, 2017a] https://www.cnblogs.com/upyun/p/7813319.html ，cnblogs，话说 webp，cnblogs，2017.

[又拍云, 2017b] https://www.cnblogs.com/upyun/p/6844690.html ，cnblogs，如何通过 WebP 自适应方案减少图片资源大小，cnblogs，2017.

[张明, 2010] 张明，*图像超分辨率重建和插值算法研究[D]*，中国科学技术大学，硕士论文，2010.

[张弥弭, 2014] 张弥弭，*基于网络自媒体平台的品牌传播模式研究*，厦门大学，2014.

[张洋, 2013] http://blog.codinglabs.org/articles/modularized-javascript-with-seajs.html，使用 SeaJS 实现模块化 JavaScript 开发，codingLabs，2013.

[周齐飞, 2014] 周齐飞，*基于 Android 平台的 Hybrid App 开发[J]*，电脑编程技巧与维护，2014(15).

[卓越开发者联盟, 2015] 卓越开发者联盟，*React：引领未来的用户界面开发框架[M]*，电子工业出版社，2015.

[周晓虹, 2011] 周晓虹，*自媒体时代:从传播到互播的转变*，新闻界，2011(4):20-22.