青岛大学
QINGDAO UNIVERSITY

本科毕业设计

题　　目：学生组织管理系统微信小程序

学　　院：计算机科学技术学院

专　　业：信息安全

姓　　名：李磊

学　　号：2017201918

指导教师：张云红

2021 年　6 月 9 日

学生组织管理系统微信小程序

Student Organization Management System WeChat Mini Program

郑 重 声 明

本人呈交的学位论文（设计），是在指导教师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。除文中已经注明引用的内容外，本学位论文（设计）的研究成果不包含他人享有著作权的内容。对本论文（设计）所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本学位论文（设计）的知识产权归属于青岛大学。

本人签名：_____　　日期：_____

摘　要

在现在的高校中，学生的活动越来越丰富，伴随着活动的丰富学生除了班级和年级这个组织关系也产生了越来越多的其他组织关系，比如：某社团成员、学生会成员等。

为了方便学生高效管理多种组织关系，以多维、可视化的角度展示出学生的组织关系，建立一个学生组织管理系统。

微信作为数十亿计月活的国民 APP，其微信小程序开发已经比较成熟，作为组织系统的开发平台比较便捷，学生组织管理系统开发平台选用微信小程序，最大程度上实现便捷。

关键词：组织管理、高效、便捷、微信小程序

Abstract

In today's colleges and universities, student activities are becoming more and more abundant. With the enrichment of activities, students have more and more other organizational relationships in addition to the organizational relationship of class and grade, such as members of a certain club, student union, etc.

In order to facilitate students to efficiently manage a variety of organizational relationships, show students' organizational relationships from a multidimensional and visual perspective, and establish a student organization management system.

As a national APP with billions of monthly activities, WeChat mini-program development has been relatively mature, and it is relatively convenient as a development platform for the organization system. The student organization management system development platform uses WeChat mini-programs to achieve maximum convenience.

Keywords： Organization Management、 Efficient、 Convenient、 WeChat Mini Program

目录

第 1 章 绪论

1.1 背景现状

随着科技进步、人民生活水平提高和时代的发展，人们的娱乐方式也呈现多样化，出现了丰富多彩的兴趣活动。大学生作为年轻群体的主力，兴趣活动更是层出不穷，越来越多的兴趣活动出现在学校中，社团数量不断增加，每年新生入学参加社团更是出现“百团大战”的景象。随着社团数量增加，学生间的组织关系逐渐复杂起来，由最开始的年级班级组织增加了各种社团成员组织，又或者学生会组织等等，同学间的组织关系管理也变得复杂起来。

1.2 解决思路

高效且便捷地处理组织关系。

1.2.1 如何高效？

将所有学生作为集合 X，所有组织作为集合 Y，则 X 中会含有 a=张三、b=李四等等学生，Y 中会含有 a1=班级、b1=学生会等等，定义二元关系 R 为属于关系。如图 1-1 所示：

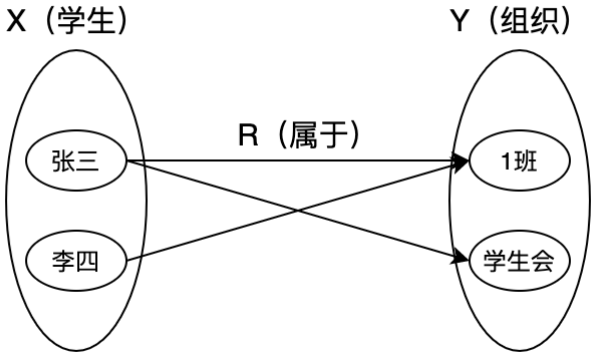

图 1-1 学生组织二元关系

建立 R=(X,Y,G(R))关系型数据库来高效地管理学生组织数据。

1.2.2 如何便捷？

20 世纪 30 年代的我们已经完全步入移动端时代，手机或其他移动设备成为生活的不可或缺的一部分，主要原因之一就是其便捷。微信 APP 作为移动端主流应用之一，小程序更是便捷，无需安装其他应用在移动端即可使用。选择微信小程序平台则可直接享受其便捷性。

1.3 章节安排

第一章为绪论，介绍学生组织系统提出的背景现状，针对弊端简单提出思路，设计组织系统；

第二章为微信小程序，介绍应用微信小程序相关知识；

第三章为系统设计，介绍组织系统功能设计、数据库设计和小程序逻辑功能的详细设计；

第四章为小程序展示，展示学生组织管理系统小程序具体使用和数据库可视化操作。

第 2 章 微信小程序

2.1 小程序介绍与开发环境

2.1.1 小程序技术发展历史

随着微信 APP 的高速发展，其 DAU 不断增加，用户数量越来越庞大，其 WebView 逐渐成为了移动端 Web 的一个重要入口，微信官方为了方便自己 Web 的开发就封装了一些 JS-API 。

在 2015 年，微信官方打包了一整套关于 Web 开发的 SDK，这套 SDK 功能十分强大，可以调用很多微信 Native 原生能力，比如：调用系统硬件能力（录音、拍摄）、微信 APP 功能（分享、支付、卡券）等等很多; 然后微信将这套 SDK 公开发布，称为 JS-SDK，这下给微信的 Web 开发者打开了一扇大门，让之前很难做到或是做不到的事情都可以轻而易举的实现。

JS-SDK 是对微信原先 Native 与 WebView 通信工具 WeixinJSBrige 的一层封装并扩充了更多调用微信 Native 原生的能力，在微信宣布对其开放之后，SDK 的使用量快速上升。随着使用数量的增加，表面上解决了很多 Web 能力不足的问题，但是移动网页性能的问题并没有改善，特别是 Web 网页的通病：白屏问题。白屏是很影响用户体验的一个问题，并且受设备性能和网速影响大，当设备低端和网络环境差时，白屏的时长会更明显。

为了解决白屏等性能问题，微信升级了 JS-SDK，推出了升级版本的 SDK，其中添加了微信 Web 资源离线存储技术，这是面向 Web 开发者提供的基于微信内的 Web 加速方案。

这个加速方案是使用微信 Native 资源存储能力，不需要每次加载网页时向服务器发送请求 Web 资源，如果本地有离线的 Web 资源，则直接从本地读取资源渲染页面即可，省掉了网络请求的耗时，可以缩短白屏时间，优化用户浏览 Web 页面。

但是这个离线存储方案仍不是完美的，它虽然可以解决一些白屏问题，但是对于那些使用 Web 资源较多的页面，即使不需要从服务端拉取 Web 资源，直接从本地读取，依然需要花费较多的时间去加载资源，比如：页面内有很多的 CSS 或者 JavaScript 脚本，这些文件就会占用大量的 UI 线程资源，一定程度上会 b 阻塞本地资源的读取。也会产生白屏现象，而且这样分文件的本地缓存在代码迭代时成本也是不低的，操作繁琐。

此外，除了白屏这一指标，还有其他 JS-SDK 没有解决的体验问题，比如一

些操作的反馈：页面切换时比较生硬、点击操作的迟滞感等。虽然这些问题

除了白屏，影响 Web 体验的问题还有缺少操作的反馈，主要表现在两个方面: 页面切换的生硬和点击的迟滞感。不过这对于一些有经验的开发者来说不是难题，可以其他优秀的框架或者逻辑处理解决掉，不过这样会增加大量的开发成本，产生的 KPI 是不够的。

所以微信团队为了使其 Web 功能不再受限于以上问题，设计了一个全新的系统——微信小程序，它具有 JS-SDK 所不具备的功能：

- 快速的加载
- 更强大的能力
- 原生的体验
- 易用且安全的微信数据开放
- 高效和简单的开发

至此，微信团队开始推广其小程序，小程序开始大放异彩，出现了各种丰富的小程序。

2.1.2 开发环境

每个小程序都一个 AppID，AppID 与小程序是一一对应的。AppID 需要使用微信账号申请，申请的账号可以管理小程序，是小程序管理员，可以添加协同开发者，只有协同开发者和管理员才可以开发与 AppID 对应的小程序。

小程序的开发语言层面与传统的 Web 开发区别不大，但是由于小程序渲染和逻辑分离的运行机制与传统 Web 开发是不同的，所有无法使用传统网页开发工具。

因此微信推出了小程序开发一站式 IDE——微信开发者工具，每个 AppID 对应小程序的管理员或开发者在微信开发者工具上扫码即可进行开发。

开发者可以借助微信开发者工具完成小程序的代码开发、编译运行、界面和逻辑调试、真机预览和提交发布版本等功能[1]。

2.2 小程序代码组成

小程序由配置代码 JSON 文件、模板代码 WXML 文件、样式代码 WXSS 文件以及逻辑代码 JavaScript 文件组成[1]。

2.2.1 JSON 配置

JSON 是一种数据格式，并不是编程语言。在小程序中，JSON 扮演的静态

配置的角色[1]。

2.2.2 WXML 模板

WXML 全称是 WeiXin Markup Language，是小程序框架设计的一套标签语言，结合小程序的基础组件、事件系统，可以构建出页面的结构[1]。

2.2.3 WXSS 样式

WXSS（WeiXin Style Sheets）是一套用于小程序的样式语言，用于描述 WXML 的组件样式，也就是视觉上的效果[1]。

WXSS 与 Web 开发中的 CSS 类似。为了更适合小程序开发，WXSS 对 CSS 做了一些补充以及修改。

2.2.4 JavaScript 脚本

小程序的主要开发语言是 JavaScript ，开发者使用 JavaScript 来开发业务逻辑以及调用小程序的 API 来完成业务需求[1]。

2.3 小程序发布

小程序提供了两种发布模式：全量发布和分阶段发布[1]。全量发布是指当点击发布之后，所有用户访问小程序时都会使用当前最新的发布版本。分阶段发布是指分不同时间段来控制部分用户使用最新的发布版本，分阶段发布我们也称为灰度发布。一般来说，普通小程序发布时采用全量发布即可，当小程序承载的功能越来越多，使用的用户数越来越多时，采用分阶段发布是一个非常好的控制风险的办法。因为随着程序的复杂度提高以及影响面的扩大，新版本的代码改动或多或少会带来 Bug，作为服务方当然不希望异常的服务状态一下子扩散到整个用户群体，此时应该通过分阶段发布来逐步观察服务的稳定性，再决定是否进行全量发布。

2.4 本章小结

在本章中我们介绍了小程序的发展历史和开发环境，都是基于微信 web 开发经验逐渐积累演变出现的产物；又介绍了小程序的代码组成，由 WXML 文件提供 UI 组件、WXSS 提供 UI 组件样式、JS 文件提供数据逻辑支持和 JSON 文

件提供静态配置能力；最后介绍了官方提供了两种小程序发布上线的模式：直接全量和分阶段发布两种。

第 3 章 系统设计

3.1 功能模块概览

从表面上看，学生组织管理系统共由 9 个页面组成，分别是：登陆界面、组织列表、组织主页、公告列表、成员列表、聊天室、新建组织、新·1 建公告、学生主页，不同的页面隶属不同的功能模块，下面看一下不同页面隶属哪个功能模块。

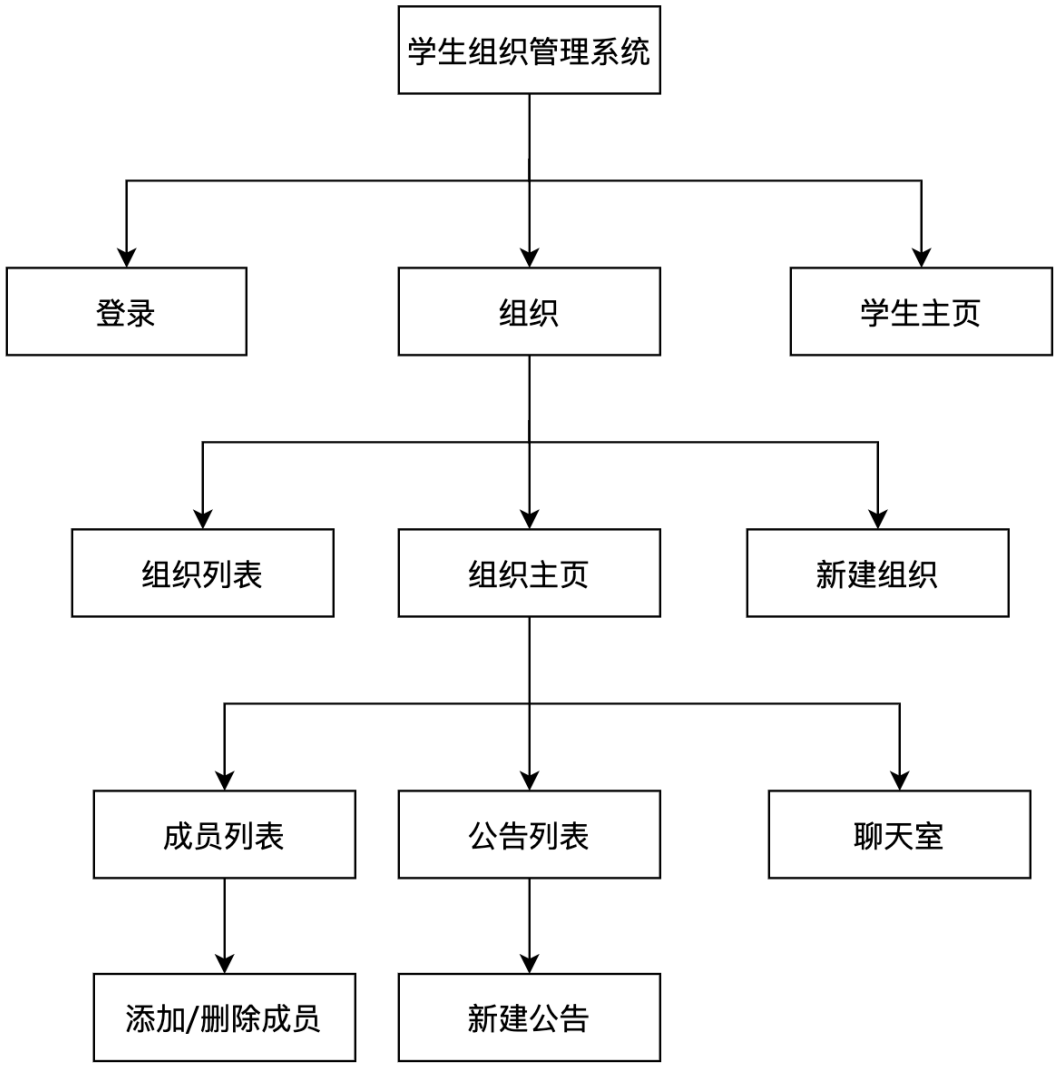

图 3-1 功能模块概览图

3.1.1 登录模块

登录模块是学生组织管理系统的入口，是用户进入系统时首先看到的页面，也是进入组织模块和学生主页模块的入口。

登录模块仅包含一个登陆界面，比较简单。输入正确的学号和密码即可进入系统其他模块。

3.1.2 组织模块

组织模块是系统的核心模块，有多个页面组成：

1）组织列表页

列表页面展示登录学生所加入的所有组织，每个组织作为列表的一个单元，类似消息列表，还有新建组织的入口。

2）新建组织

用户在新建组织页面可以添加一个新的组织，输入组织名称和组织名称必要信息，同时会默认将创建者作为管理员并加入组织。

3）组织主页

组织主页展示组织的信息：组织头像、组织名称、组织最新公告、组织负责人、组织成员数量，进入聊天室的入口和退出组织按钮。

点击公告栏、成员数量栏也会相应进入其二级页。

4）成员列表

成员列表是类似通讯录的一个页面，会将成员按字典序分组并排列。在顶部会有成员管理的两个功能：添加成员和移除成员，当然这是属于管理员的权限。

5）公告列表

公告列表记录由公告单元组成，每个公告单元展示着公告内容、发布者以及发布时间；右下角的发布按钮是进入发布页面的入口。

6）新建公告

新建公告页面比较简单，只有输入框和发布按钮，输入框可以多行输入想要发布的新公告，点击发布即可实时发布。这是属于所有组织成员的权利。

7）聊天室

聊天室也是所有组织成员可参与的，是即时通讯，类似微信等通讯类软件，可以实现图片消息和文字消息的实时发送。

3.1.3 学生主页模块

学生主页模块仅有学生主页一个页面组成，和消息列表构成组织系统的两个底 tab。页面内展示了学生的个人信息：姓名、学号、年级、班级、学院和专业等内容。

3.2 数据库设计

数据库是学生组织管理系统用来存储、管理数据的优秀数据结构。

3.2.1 微信云开发的数据库

微信云开发提供的能力中包含数据库能力，但提供的数据库不是关系型数据，是新型的 JSON 文档型数据库，我们可以用它来代替关系型数据库。一个数据库可以有多个集合（相当于关系型数据中的表），集合可看做一个 JSON 数组，数组中的每个对象就是一条记录，记录的格式是 JSON 对象。关系型数据库和 JSON 数据库的概念对应关系如下表：

表格 3-1 关系型数据库与 JSON 数据库对应关系

关系型数据库	JSON 数据库
表 table	集合 collection
行 row	记录 record / doc
列 column	字段 field

3.2.2 集合设计

1) 将所有学生作为一个集合（表）来存储学生信息，作为学生单元

表格 3-2 student 集合（表）设计

Student（collection / table）	
字段 field	含义
_id	记录唯一 id
_createTime	记录创建时间
_updateTime	记录更新时间

avatar	头像地址
name	姓名
password	密码
student_id	学号
gender	性别
grade	年级
class	班级
specialty	专业
college	学院

2）创建一个组织集合，包含所有组织

表格 3-3 组织集合（表）设计

Organization（collection / table）	
字段 field	含义
_id	记录唯一 id
_createTime	记录创建时间
_updateTime	记录更新时间
organization_name	组织名称
organization_avatar	组织头像 url
organization_members（JSON 数组）	组织成员，关联 Student._id
announcements（JSON 数组）	组织公告列表
admin	管理员，关联 Student._id

3）聊天室集合，用于存储聊天室的每条消息

表格 3-4 聊天室集合（表）设计

Chatroom（collection / table）	
字段 field	含义
_id	记录唯一 id

group_id	组织 ID，关联 organization._id
msg_type	消息类型（text/image）
avatar	发送者头像 url
name	发送者姓名
sendTime	发送时间
sendTimeTS	发送时间戳
textContent	文本内容
imgFileID	图片地址
user_id	发送者 id，关联 Student._id

3.2.3 数据模型 E-R 图

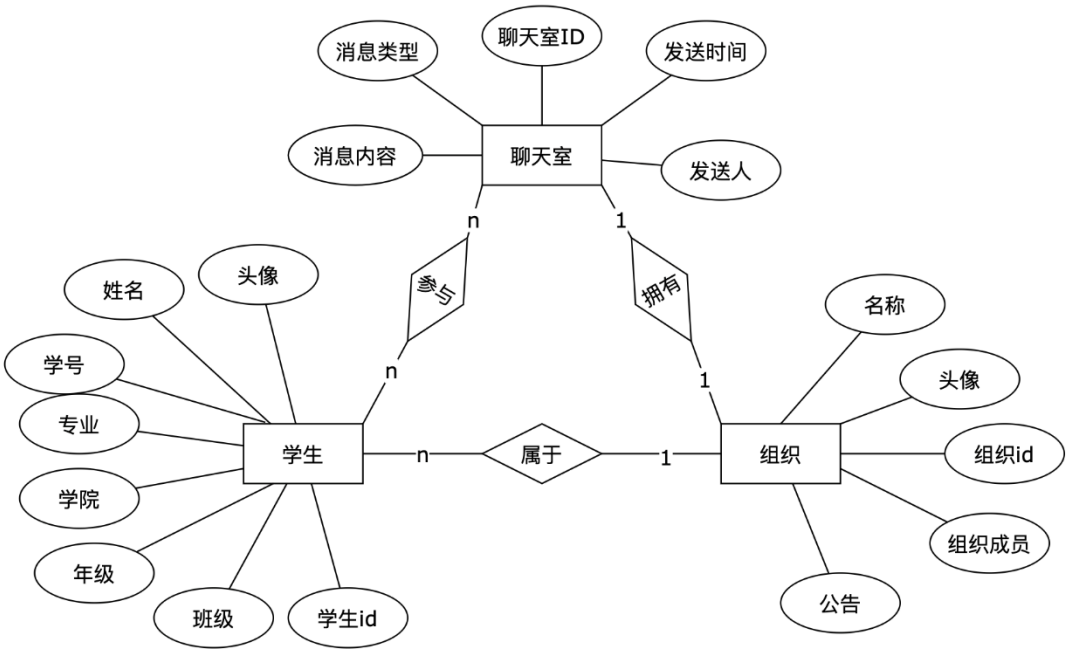

图 3-2 组织数据模型 E-R 图

图中实体有 3 个：学生、组织和聊天室

主要说一下实体之间的联系：

学生与组织之间存在隶属的联系，1 个组织可能有 n 个学生成员；

学生与聊天室之间存在参与关系，n 个学生使用 n 个聊天室功能进行聊天；

聊天室与组织之间存在拥有关系，1 个组织拥有 1 个聊天室。

3.3 小程序逻辑设计

按功能模块部分讲述不同页面的数据和逻辑流程。

3.3.1 登录页面

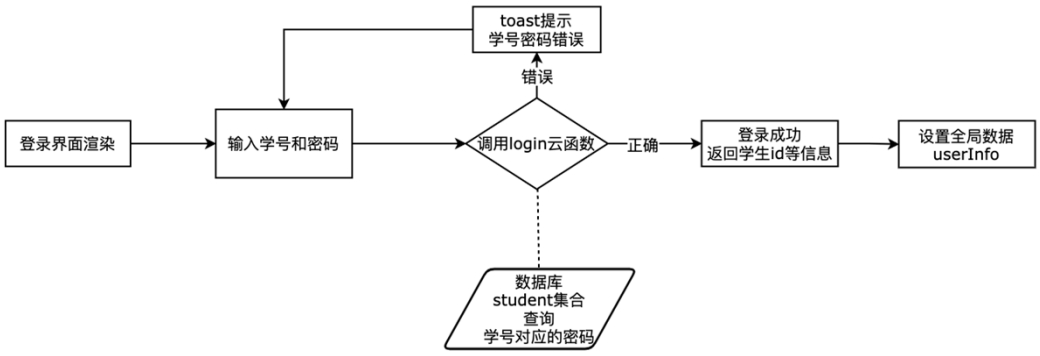

图 3-3 登录流程图

学生在登录界面输入学号和密码即可请求登录，会调用腾讯云服务的 login 云函数[1]（无需自己部署服务器，只需编写业务代码），login 云函数会在 student 集合中查询学号等于用户输入的，对比学号对应的密码是否与登录者相同，相同则登录成功。login 云函数代码见附录。

3.3.2 组织列表

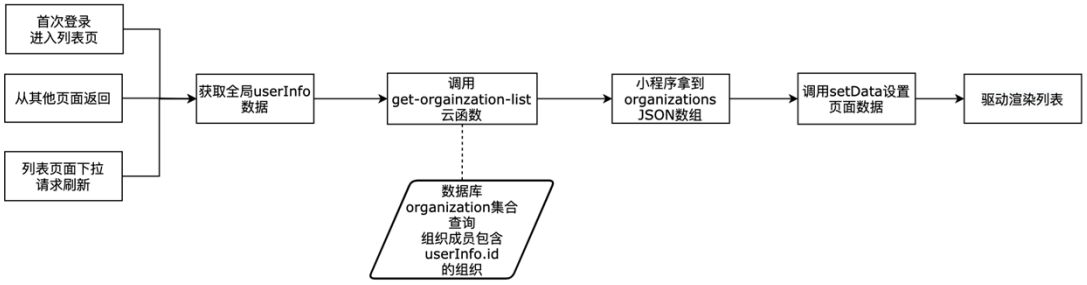

图 3-4 组织列表渲染流程

组织列表页面属于数据驱动渲染页面，需要拿到数据后才会进行页面渲染，没有数据时会处于骨架屏 loading 状态。

学生在首次登录进入组织列表、从其他页面返回或者在列表页面内下拉请求刷新时，都会获取全局 userInfo 用户数据，然后调用云函数 get-organization-list 并把 userInfo 作为函数的参数传入，云函数会在 organization 集合中查询组织成

员中包含 userInfo.id 的组织，然后返回所有符合查询条件的组织列表；小程序拿到列表（JSON 数组）后调用 setData 给页面设置数据。因为页面是数据驱动的方式，所以在设置数据后会驱动页面刷新 UI 重新渲染，结束骨架屏 loading 状态。

3.3.3 新建组织

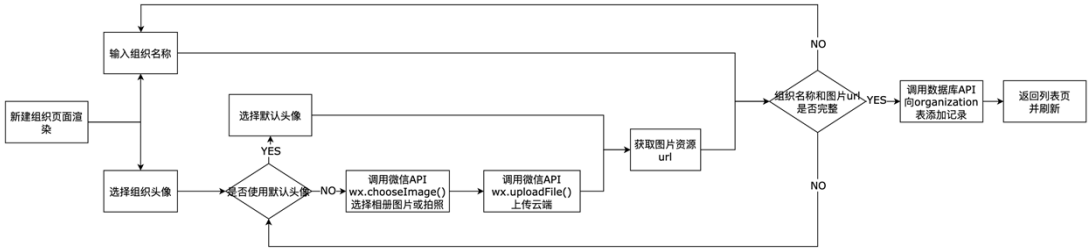

图 3-5 新建组织流程

新建组织页面无需数据驱动渲染，进入这个页面后输入组织的名称和选择组织的头像既可以创建新的组织。选择组织头像分两种情况：一是选择系统默认头像，这样的话图片资源的 url 是系统已知的，另一种情况就是用户选择自己上传系统头像，这样需要先调用微信的接口 wx.chooseImage()上传设备相册图片或者拍照，再通过 wx.uploadFile()接口上传图片至云端获取 url。当组织名称和图片 url 输入完整后就可以向数据库 organization 集合中添加一条新的记录作为一个新的组织；添加成功后会返回组织列表页自动刷新即可看见新建的组织。

3.3.4 组织主页

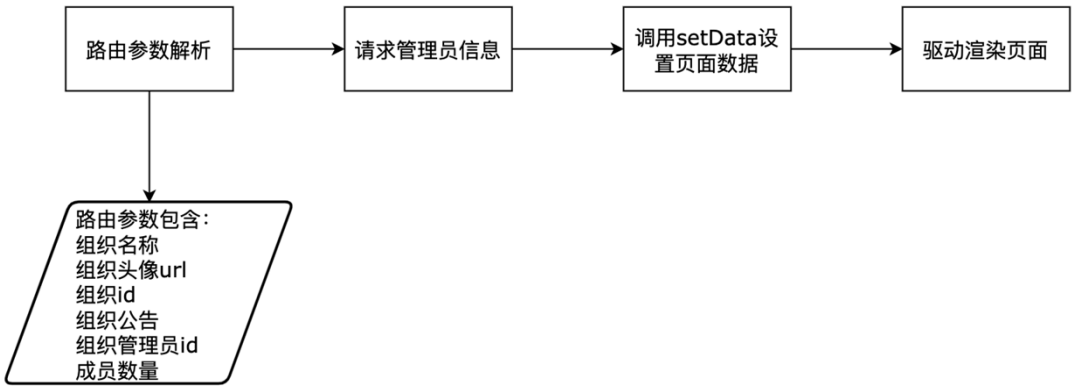

图 3-6 组织页面加载流程

组织主页也属于数据驱动渲染页面,所以页面渲染需要数据，没有数据时会处于骨架屏 loading 状态。

页面的大部分数据是通过路由参数带入的，比如：组织名称、组织头像 url、

组织 id、组织公告、组织管理员 id 和成员的数量，页面需要显示的是管理员姓名不是 id，所以做一次 student 表的查询，拿到管理员姓名后设置页面数据即可进行渲染，结束 loading 状态。

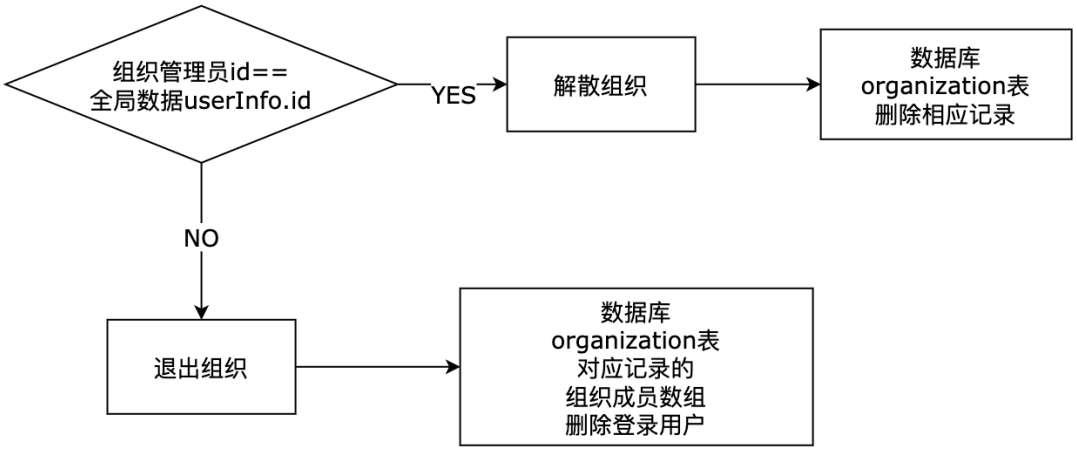

图 3-7 退出组织流程

退出组织是在组织主页的一个按钮，分两种情况：一是组织管理员 id 等于全局 userInfo.id，意思是登录用户为当前组织的管理员，退出的话就会解散组织，再点击退出就会解散组织，从 organization 表中删除当前组织的记录；另一种情况就是登录用户只是组织的普通成员，退出组织操作仅会将其 student 集合的记录 id 从组织的成员数组中删掉。

3.3.5 公告列表

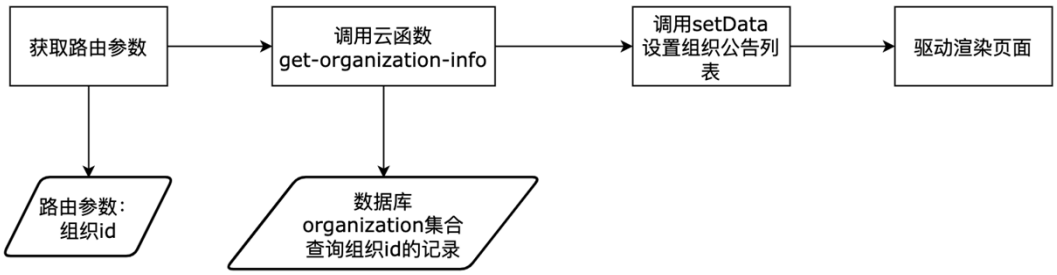

图 3-8 组织列表加载流程

公告列表页面也属于数据驱动渲染页面，但是未使用骨架屏，默认样式是暂无公告，加载数据后再展示公告列表。

公告列表进入后从路由参数中取组织 id，然后调用云函数 get-organization-info 来在数据库 organization 集合中查询组织 id 等于路由参数的记录并返回组织信息，从组织信息中取公告列表数据再调用 setData 设置页面数据进行页面渲染。

3.3.6 新建公告

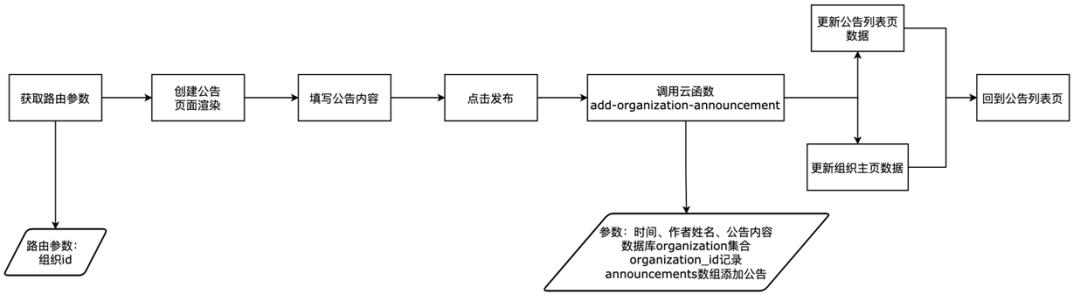

图 3-9 创建公告流程图

创建公告页面无需数据驱动渲染，获取路由参数后即渲染公告输入框和发布按钮。用户在完成输入公告内容后点击发布，就会调用云函数 add-organization-announcement，在数据库 organization 集合的 organization_id 记录中的 announcements 数组添加公告。

云函数执行完成后更新视图栈中的上一级页面公告列表页数据，重新渲染；再更新上上级页面组织主页公告栏数据，重新渲染。发布完成，跳转回到公告列表页面。

3.3.7 成员列表

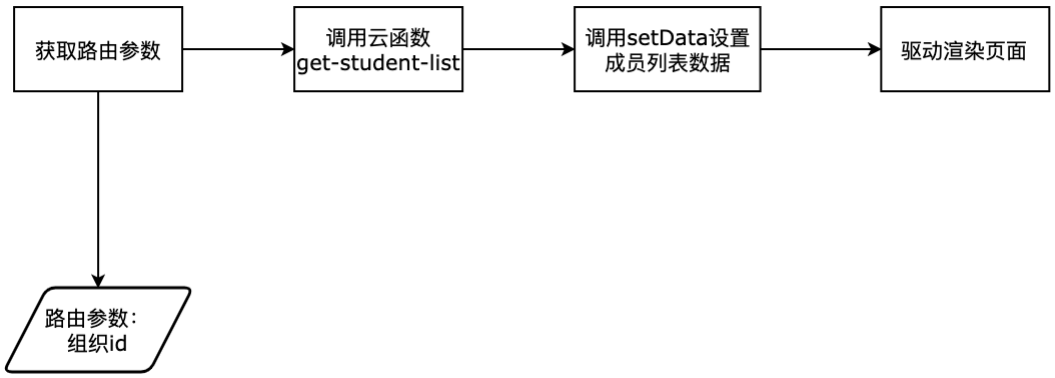

图 3-10 成员列表加载流程

成员列表页面也是数据驱动渲染页面，需要在获取到组织成员列表后才能渲染，在拿到数据之前展示骨架屏 loading 态。

成员列表页面从路由参数中获取组织 id，根据 id 请求 get-student-list 云函数，云函数会返回按字典序分组并排列好的成员数据列表方便小程序进行渲染。

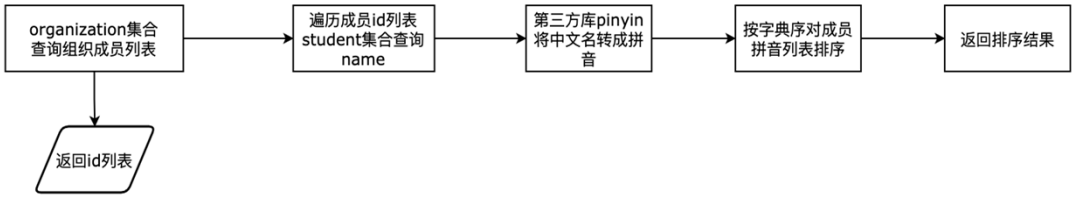

图 3-11 get-student-list 云函数流程

get-student-list 云函数比其他云函数复杂一些，不只是简单的数据库查询操作：

1）首先它是根据参数组织 id 查询 organization 集合的记录，取出组织成员列表；

2）这时的成员只是一堆 student 集合的记录 id，而且是无序的，需要再在 student 集合中查询每个 id 对应的 name 值；

3）做完 id 到 name 转换后，需要将 name 按字典序分组排列，这时用到了第三方库《汉字拼音转换工具》，通过此组件可以将中文转换成拼音；

4）再将拼音列表按照字典序排序并分组就是比较简单的了。

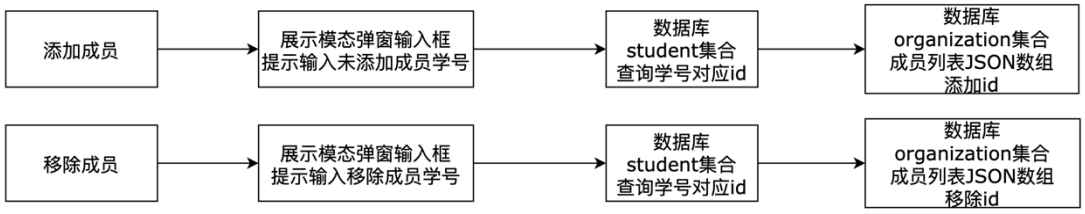

图 3-12 成员管理流程图

成员管理功能分为两种：添加成员和移除成员。

添加成员功能和移除成员流程类似，点击后都会提示微信模态弹窗，提醒输入待添加或移除的成员的学号，再在 student 集合中查询学号对应的 id，然后在 organization 集合中对于当前记录的组织成员数组添加或移除之前查询的 student 集合 id。

3.3.8 聊天室

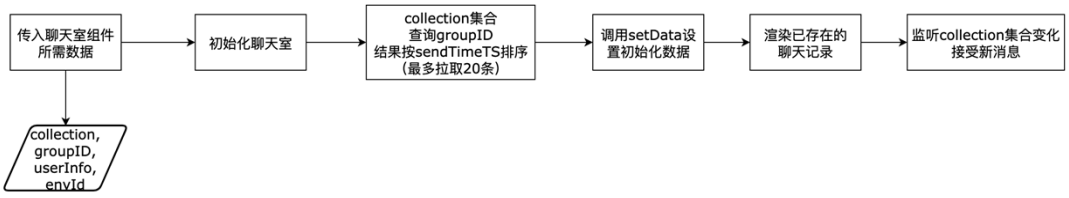

图 3-13 聊天室初始化流程

传入聊天室所需的数据：聊天室集合、聊天室 id（groupID）、用户信息和

环境 id，来进行聊天室模块的初始化。

在数据库传入的集合中查询传入 groupID 的记录，并取出按时间顺序排序的 20 条消息，设置给页面数据并渲染消息，再继续监听集合数据变化。

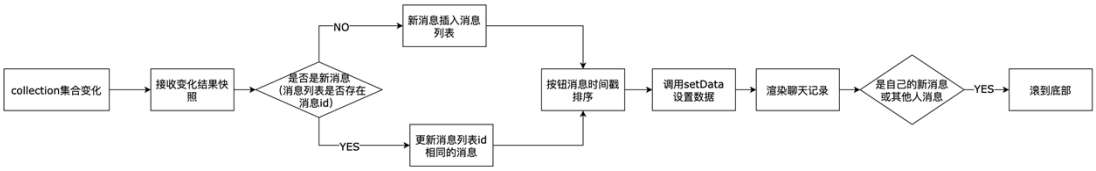

图 3-14 新消息接收流程

新消息接受基于微信数据库提供的实时消息推送能力：给定查询条件，每当数据库更新而导致查询条件对应的查询结果发生变更时，小程序可收到一个更新事件，其中可获取更新内容和更新后的查询结果快照。

接受变化的快照后，在当前消息列表中查询是否存在相同 id，如果有的话，替换消息；如果没有的话说明是新消息插入到消息列表；再将数组按时间戳顺序排序，设置页面数据渲染聊天页面。

如果是自己新发的消息或者其他人的消息将聊天列表滚动到底部方便用户使用。

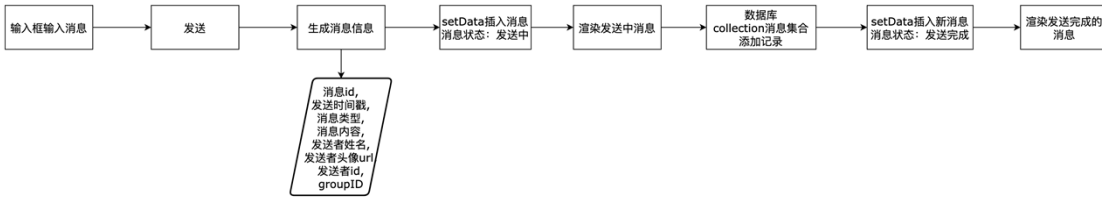

图 3-15 发送消息流程

用户在输入框中输入消息，点击发送就会生成消息信息，包含：消息 id、发送时间戳、消息类型、消息内容、发送者姓名、发送者头像 url、发送者 id、聊天室 groupID；这时将消息状态置为发送中，再在数据库 collection 消息集合中添加记录，完成后的回调将消息类型置为发送完成，页面将发送中的消息渲染为发送完成的消息，一条消息就发送完成了。

3.3.9 学生主页

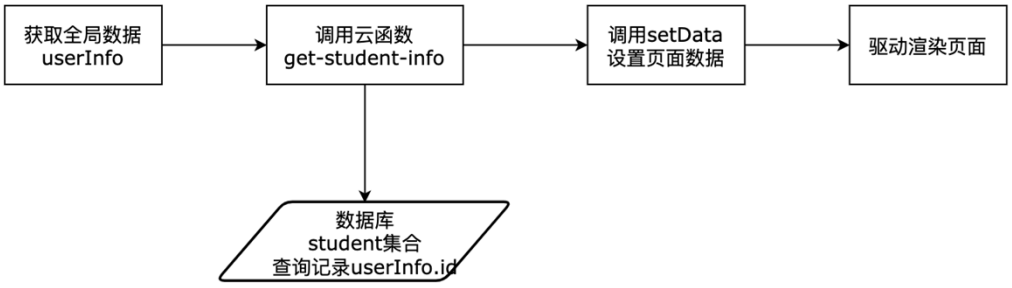

图 3-16 学生主页加载流程图

学生主页属于数据驱动渲染页面，会在获取全局数据 userInfo 后请求云函数 get-student-info 返回学生个人信息，再调用 setData 设置页面数据，驱动页面进行渲染。

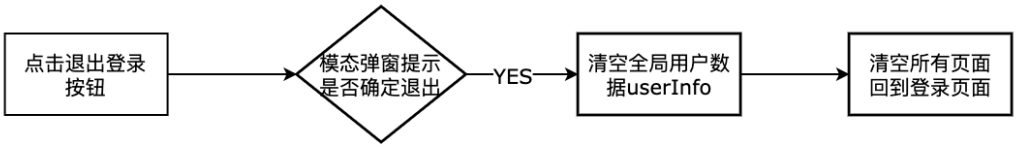

图 3-17 退出登录流程

学生主页包含一个退出登录按钮，点击可以退出组织管理系统。其流程比较简单：点击退出后会弹一个确认视图，点击确认后会清空全局用户数据 userInfo 和所有页面，然后回到登录页面。

3.3.10 逻辑测试

系统测试账号：2017201918

系统测试密码：123456

测试环境：微信开发者工具打开组织系统小程序项目，编译即可。

3.4 本章小结

在本章中我们介绍了学生组织管理系统的功能设计、数据库设计和逻辑设计。功能设计介绍了组织管理系统的三大模块：登录、组织和学生主页模块，就每个模块所包含的页面以及各个模块页面包含的功能作了陈述；数据库设计则是介绍了采用的微信官方提供的 JSON 数据库，它与关系型数据库的对应关系，并用

它实现了关系型数据库的功能，以及学生组织管理系统的三个集合（表）设计，通过 E-R 图来更直观地展示了数据模型；最后逻辑设计则是按功能设计模块的划分，分别介绍了各个功能的逻辑数据流程，通过流程图来直观地展现，还有测试环境，将项目使用开发者工具打开编译即可运行测试。

第 4 章 小程序展示

4.1 小程序码

图 4-1 小程序码

图 4-2 小程序体验二维码

发布后，微信扫一扫小程序码即可进入小程序。未发布时体验组成员可以扫描体验二维码进入。

4.2 登录

图 4-3 登录页面

输入学号和密码即可登录系统。

4.3 组织列表页

图 4-4 组织列表

组织列表页展示登录用户已加入的所有组织，下拉可以刷新页面。每个组织 cell 展示组织头像、组织名称和最新公告。

4.4 学生主页

图 4-5 退出登录

图 4-6 学生主页

学生主页展示学生个人信息，并且有退出系统按钮，点击可以退出登录。退出登录，返回未登录状态。

4.4 创建组织

图 4-7 创建组织

输入组织名称和选择组织头像后即可创建一个新的组织。头像可以选择默认的或者自定义从相册和拍照选择。

4.5 组织主页

4.5.1 组织主页列表

图 4-8 组织主页

组织主页展示组织信息：头像、名称、最新公告、负责人、成员数量。公告栏可以跳转至公告页面，成员栏可以跳转至成员列表页，退出按钮可以退出组织。

4.5.2 退出组织

图 4-9 管理员退出组织

图 4-10 非管理员退出组织

区分用户是否是管理员，是管理员会解散组织，不是管理员则仅退出组织。

4.6 公告

4.6.1 公告列表

图 4-11 公告列表

公告列表展示组织所有发布过的公告，以及创建者和时间。

4.6.2 创建公告

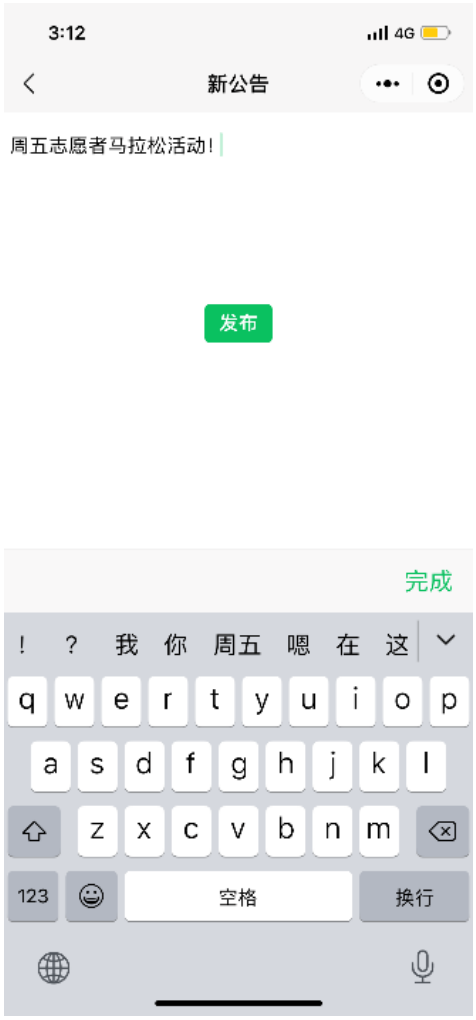

图 4-12 创建公告

在输入框输入公告内容即可发布，作为组织最新公告。

4.7 组织成员

4.7.1 成员列表

图 4-13 成员列表

成员列表展示组织内所有成员，并按字典序分组排了，而且有成员管理入口。

4.7.2 成员管理

图 4-14 移除成员

图 4-15 添加成员

添加、移除成员在弹窗内输入要操作的学号就可以添加或移除了。

4.8 聊天室

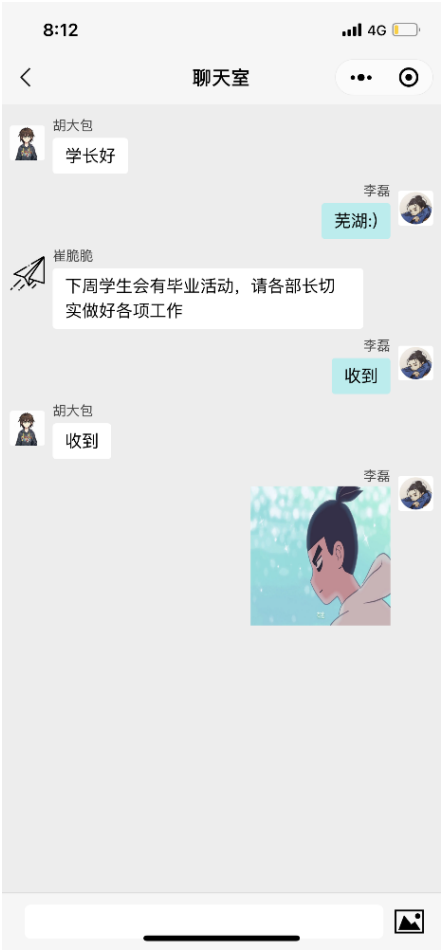

图 4-16 聊天室

聊天室界面，可以与组织内成员进行即时通信；而且支持图片消息类型。

4.9 数据库管理

微信提供了可视化内容管理平台方便我们管理数据库模型，无需编码即可使用，方便使用。组织系统使用内容平台管理 student 集合和 organization 集合。

可以实现批量导入学生数据。

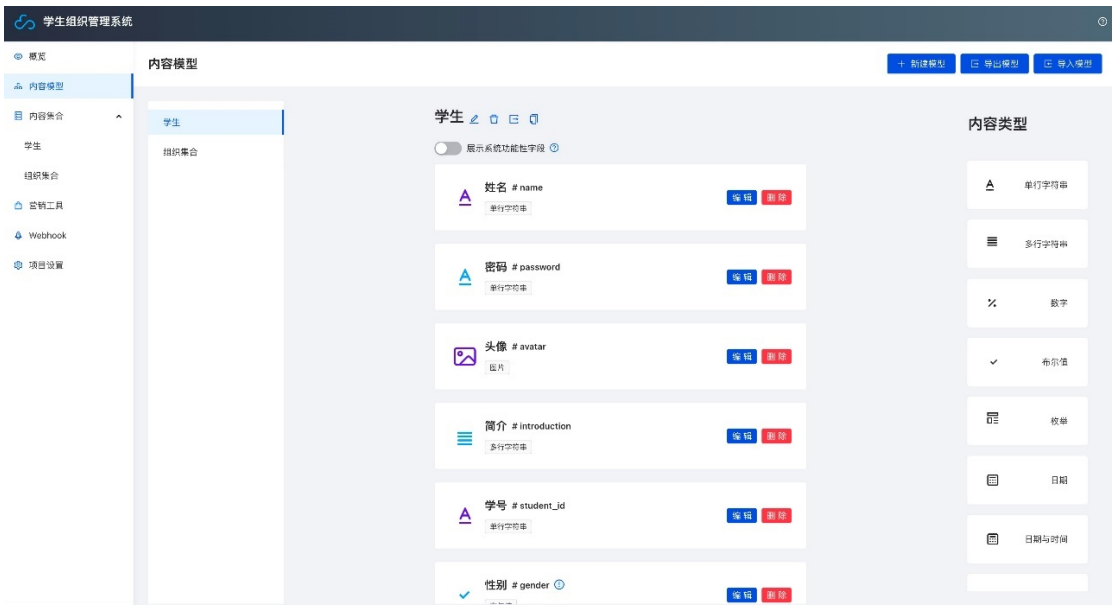

图 4-17 可视化数据库 student 模型

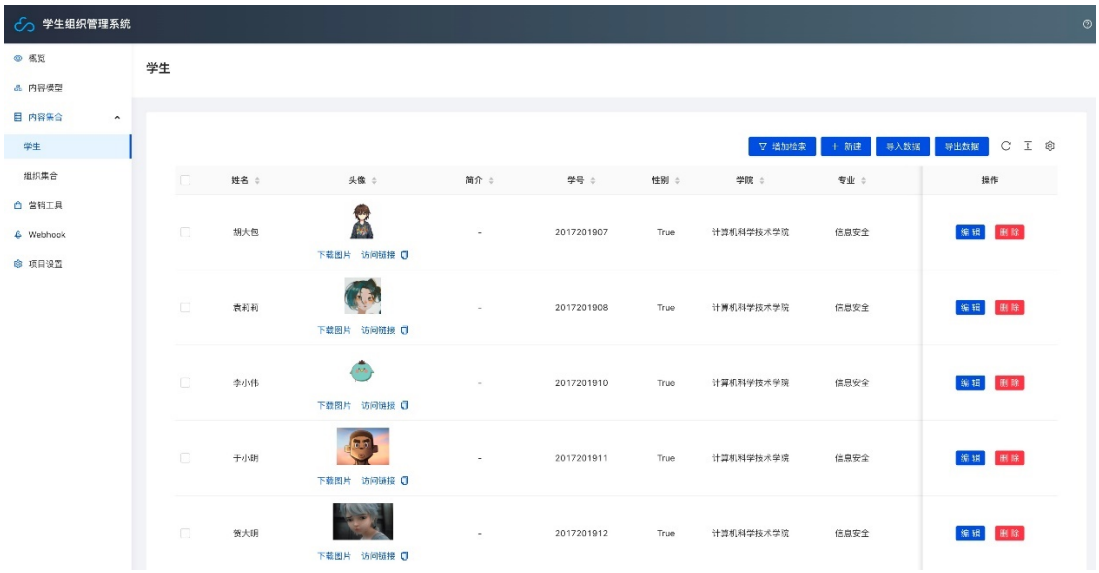

图 4-18 可视化数据库 student 集合

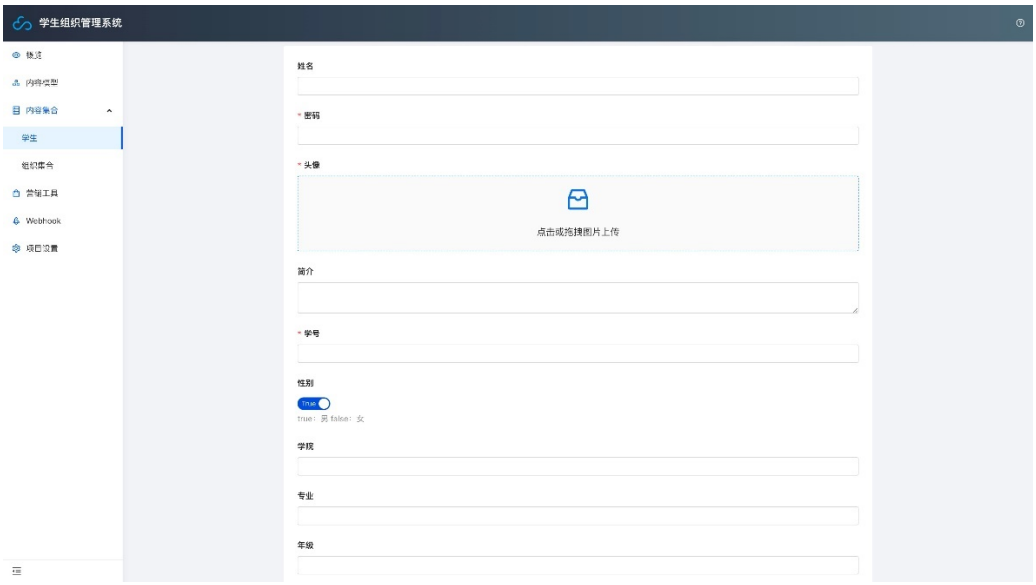

图 4-19 可视化数据库 student 集合添加记录

4.10 本章小结

本章先给出了学生组织系统微信小程序的小程序码，待发布后即可扫码进入系统；然后详细地展示了系统的各个页面，简单地对页面内功能进行了阐述；最后展示了如何方便地向数据库导入学生信息。

结束语

本文就繁琐难以处理的学生组织关系，提出了学生组织管理系统微信小程序设计，用来高效便捷处理组织关系；然后介绍微信小程序开发的相关知识，阐述了小程序的诞生、发展以及小程序开发和发布的相关知识；再介绍了小程序的系统设计，包括功能设计、数据库设计和逻辑设计；最后演示了小程序的各个功能的页面和数据库可视化操作。

小程序体验版功能测试正常，可以正常实现功能设计模块中的功能。

谢辞

本论文是我的第一篇论文，从论文的选题、研读和实现，每一步都是新的挑战，最终选中组织系统小程序命题，也是出于对前后端知识的渴望。在完成论文期间我不仅对前后端知识有了更深的了解，也更加佩服那些推动技术进步，乐意把知识分享，共同进步的科研工作者，也更加坚定了我向着更深方向学习发展的决心。

在这期间，首先感谢我的指导老师张云红。从选题、开题报告、中期检查到定稿，每个阶段老师都给予了我很大的帮助，抽身于百忙之中对我悉心指导，无微不至，很荣幸成为老师指导的学生，使我受益匪浅。

其次感谢在大学期间我的所有授课老师，是他们的悉心教导，才成就了今天的我。他们的言传身教，深刻地感染了我，激励着我不断向前，继续在学术的浩海中前行！ 最后，感谢我的父母及家人。你们对我持续不断的严格教育，使我学会勇敢地面对困难，更加坚强，在磨砺中得到成长。你们始终如一的支持和关爱也给予了我最温暖的心灵港湾。

参考文献

[1] 小程序开发指南 - 微信官方文档. Available at https://developers.weixin.qq.com/ebook?action=get_post_info&docid=0008aeea9a8978ab0086a685851c0a

[2] 杜磊. PaaS 平台后端管理系统的设计与实现[D]. 北京交通大学, 2017.

[3] 黄庆祥. 微信实习管理平台的建设[D]. 江门职业技术学院, 2018.

[4] 姬彦雪. 基于行为数据的儿童健康管理应用设计与实现[D]. 北京邮电大学, 2018.

[5] 孙月玲. 微信小程序的设计与开发[D]. 科技创新导报, 2018.

[6] 邹明荣 刘小玲 黄琨 高鑫. 基于 WXSS/WXML 技术的景区微信小程序的开发——以西岭雪[D]. 四川旅游学院, 2020.

[7] 王元刚 李冠宇 丁亚飞. 面向 Agent 的信念修正的系统模型[D]. 大连海事大学信息科学技术学院, 2016.

[8] 周扬华 高颖. 互联网沟通协同平台在高速公路运营养护业务领域中的应用[D]. 中国交通信息化, 2019.

[9] 王雯心. 基于微信的英语口语翻转课堂模式研究[D]. 河南广播电视大学, 2019.

[10] 张胜南. 实验教学管理系统设计与实现[D]. 西安电子科技大学, 2015.

[11] 吴彦德. 基于微信的网络借贷信息中介子系统的设计与实现[D]. 西安电子科技大学, 2017.

[12] 王芊溪 盛姣 王岩. 微信小程序“停车我帮您”探讨[D]. 湖北农机化, 2019.

[13] 张守朋. 基于 Web 新闻发布论坛的系统的建立[D]. 吉林大学, 2008.

[14] 金峰. 基于微信小程序的家用物联网系统开发[D]. 浙江大学, 2019.

[15] 许燕. 基于小程序的在线少儿英语学习平台的研究与设计[D]. 软件, 2019.

[16] 谢根甲. 基于 NOSQL 的国际贸易与投资信息系统开发[D]. 湖南大学, 2015.

[17] 李哲 周灵. 微信小程序的架构与开发浅析[D]. 佛山科学技术学院, 2019.

附录（代码）

代码地址：https://github.com/dawn-LL/Student-Organization-Management-System

```
// 云函数 add-organization-announcement
const cloud = require('wx-server-sdk')

cloud.init({
  env: 'dawn-0gwq7rxef71fdb3f'
})

exports.main = async (event, context) => {
  const db = cloud.database()
  db.collection('organization').doc(event.organization_id).update({
    data:{
      announcements:db.command.push({
        each:[event.announcement],
        position:0
      })
    }
  }).then(res => {
    console.log("[添加公告]",res)
  })
}
```

```
// 云函数 get-organization-info
const cloud = require('wx-server-sdk')
cloud.init({

  env:'dawn-0gwq7rxef71fdb3f'
})

/**
 *
 * @param { id 组织 doc_id option 返回信息} event
 * @param {*} context
 */
exports.main = async (event, context) => {
  const db = cloud.database()
  let res = await db.collection('organization').doc(event.id).get()
  console.log("[get-organization-info] id=", event.id, "res=", res)
  return res
```

```
}

// 云函数 get-organization-list
const cloud = require('wx-server-sdk')

cloud.init({
  env: 'dawn-0gwq7rxef71fdb3f'
})

// 云函数入口函数
exports.main = async (event, context) => {
  const db = cloud.database()
  let res = await db.collection('organization').where({
    organization_member:event.userInfo.id
  }).get()
  console.log("返回结果:", res)
  return res;
}



// 云函数 get-student-info
const cloud = require('wx-server-sdk')

cloud.init({
  env: 'dawn-0gwq7rxef71fdb3f'
})

// 云函数入口函数
exports.main = async (event, context) => {
  const db = cloud.database()
  let res = {
    data:{}
  }
  if (event.id != undefined) {
    res = await db.collection('student').doc(event.id).get()
  } else if (event.student_id != undefined) {
    let tmp = await db.collection('student').where({
      student_id:event.student_id
    }).get()
    res.data = tmp.data[0]
  }
  console.log("[get-student-info-event]", event)
  console.log("[get-student-info-res]", res)
```

```
  return res
}
// 云函数 get-student-list
const cloud = require('wx-server-sdk')
const pinyin = require('pinyin')

cloud.init({
  env:'dawn-0gwq7rxef71fdb3f'
})

// 云函数入口函数
/**
 *
 * @param {id} event
 * @param {*} context
 * 根据组织 id 返回组织成员字典序列表
 * [{
 *  alpha:'a',
 *  list:[]
 * }, {
 *  alpha:'b',
 *  list:[]
 * }]
 */
exports.main = async (event, context) => {
  const db = cloud.database()
  let org_info = await db.collection('organization').doc(event.organization_id).get()
  console.log("id=", event.organization_id, "org_info:", org_info)
  var array = org_info.data.organization_member//组织成员 id 列表

  var list = []//学生姓名和首字母列表（用于排序）
  for(let i = 0; i < array.length; i++) {
    let stu_item = await db.collection('student').doc(array[i]).get()
    console.log("[stu_item]", stu_item)
    //排序数组元素
    let list_cell = {
      name:"",
      alpha:"",
      avatar:"",
    }
    list_cell.avatar = stu_item.data.avatar
    list_cell.name = stu_item.data.name
    let alpha = pinyin(stu_item.data.name, {
      style:pinyin.STYLE_FIRST_LETTER
```

```
    })
    list_cell.alpha = alpha[0]
    console.log("[list_cell]", list_cell)
    list[i] = list_cell
  }
  list = list.sort((a, b) => {
    return pinyin.compare(a.name, b.name)
  })
  console.log("[sorted list]", list)

  var res = []//返回结果
  var res_length = 0
  var res_cell = {
    alpha:"",
    list:[]
  }
  for(let i = 0; i < list.length; i++) {
    if(i == 0) {
      res_cell.alpha = list[i].alpha
    } else if(list[i].alpha > list[i-1].alpha) {
      res[res_length++] = Object.assign({}, res_cell)
      res_cell.alpha = list[i].alpha
      res_cell.list = []
    }
    res_cell.list.push(list[i])
  }
  res[res_length++] = Object.assign({}, res_cell)
  console.log("[返回结果]", res)
  return res
}



//云函数 login
const cloud = require('wx-server-sdk')

// 初始化 cloud
cloud.init({
  // API 调用都保持和云函数当前所在环境一致
  env: 'dawn-0gwq7rxef71fdb3f'
})

/**
 * event 参数包含小程序端调用传入的 data
```

```
 */

exports.main = async (event, context) => {

  const db = cloud.database()

  let res =  await db.collection("student").where({
    student_id:event.account
  }).get()
  console.log("rse:", res)
  console.log("event:", event)

  if (event.password == res.data[0].password) {
    return {
      msg:"success",
      name:res.data[0].name,
      id:res.data[0]._id,
      avatar:res.data[0].avatar,
    }
  } else {
    return {msg:"密码错误"}
  }
}
```