XXXX 学院

实验教学报告书

学　　院　XXXXXXXXXXXXXXXXXXXXXXX

专业班级　XXXXXXXXXXXXXXXXXXXXXXX

学生学号　XXXXXXXXXXXX

学生姓名　XXXXXX

实践课程　《网络应用创新设计》

实 验 报 告

实践项目	聊天室小程序的开发和实践	指导教师	XXX
实践时间	第 1-16 周 64 课时	实践地点	新闻楼 429

实践目的：

结合本学期所学微信小程序的知识，运用云服务，api 的调用，代码逻辑实现，完成一个聊天室小程序

实践内容：

本小程序一共分为六个功能，

第一、最新版本的获取微信头像昵称功能，最新版用户登录的功能必须将头像上传到云存储中，才能与别人完成交互。

第二、可以在首页的公屏上和别人互动聊天，

第三、可以创建房间，并可以直接将房间号发到首页的公屏上

第四、房间内聊天

第五、首页公平聊天的复制功能，首页公平的复制功能分为两种，公平聊天的复制功能与房主发房间号信息的复制功能是分开的

第六、搜索房间功能

本小程序一共有三个页面，分别是首页，登录页和房间页。

本小程序应用到了云开发中的实时数据推送功能。

实践过程：

一、项目概要

(一)小程序简介

随着智能手机的兴起，人们的交流越来越依靠手机，同时也越来越封闭，这款小程序就是为了满足日常交友，陌生人之间聊天而准备的，为的就是打破封闭，丰富人们日常生活。

(二)项目说明

该小程序的模块并不多，功能也比较单一，相对于微信来说更是不值一提，但我看中的是实时数据推送的前景，因为我想在毕业设计中应用更多的，更丰富的关于实时数据推送的功能，所以我才会在结课设计中进行尝试，了解它具体应该如何实现。

(三)目标群体

主要用户群体：使用智能手机的人群以年轻人为主

(四)设计背景

实时数据推送是小程序云开发已经发布的一个云服务，可以监听云数据库的数据变更，实时推送到小程序端，省去了开发者搭建 WebSocket 的成本，是小程序中实时推送的高效实践方案。

实时数据推送有广泛应用场景，此处是一些示例：

1．聊天/即时通信：小游戏内聊天、大厅广播、区服广播等；企业内部小程序中的即时通信能力等

2．多人小游戏：使用状态同步的小游戏，如棋牌类等回合制游戏

3．协作工具：如在线协作文档、团队任务管理等

4．实时应用状态同步：以信息流为例，可以实时获取最新文章、以及最新评论、点赞、通知等内容，让交互更顺畅自然

本次结课设计，我将运用课程内所学的关于调用云端数据库、云存储等功能，结合最新版微信头像昵称填写能力以及实时数据推送开发一款聊天室小程序

（五）微信小程序简介

微信小程序，小程序的一种，英文名 Wechat Mini Program，是几乎感知不到下载安装即可使用的应用，它实现了应用“触手可及”的梦想，用户扫一扫或搜一下即可打开应用。

全面开放申请后，主体类型为企业、政府、媒体、其他组织或个人的开发者，均可申请注册小程序。微信小程序、微信订阅号、微信服务号、微信企业号是并行的体系。

小程序使用的是 MINA 框架，目的是通过简单、高效的方式让开发者可以在微信中开发具有原生 App 体验的服务。

MINA 的核心是一个响应的数据绑定系统。

整个系统分为两块：视图层（view,描述语言 wxml 和 wxss）和逻辑层（App Serice，基于 JavaScript）。这可以让数据与视图非常简单的保持同步。当做数据修改时，只需要在逻辑层改数据，视图层就会做响应的更新。开发者只需要将页面路由、方法、生命周期函数注册进框架，其他的一切复杂的操作都将由框架处理。

1.逻辑层

小程序的逻辑层就是所有.js 脚本文件的集合。小程序在逻辑层处理数据并发送至视图层，同时接受视图层发回的事件请求。

MINA 框架的逻辑层是由 JavaScript 编写，在此基础上，微信团队做出了一些优化，以便更高效的开发小程序，这些优化包括：

1、增加 app 方法用来注册程序，增加 page 方法用来注册页面；

2、提供丰富的 API 接口；

3、页面的作用域相对独立，并拥有了模块化的能力；

简单概括，逻辑层就是各个页面的.js 脚本文件。

需要注意的是，小程序的逻辑层由 js 编写，但并不是在浏览器中运行的，所以 JavaScript 在 Web 中的一些能力都不能使用，比如 dom、window 等，这也是我们开发过程中要克服的阻碍。

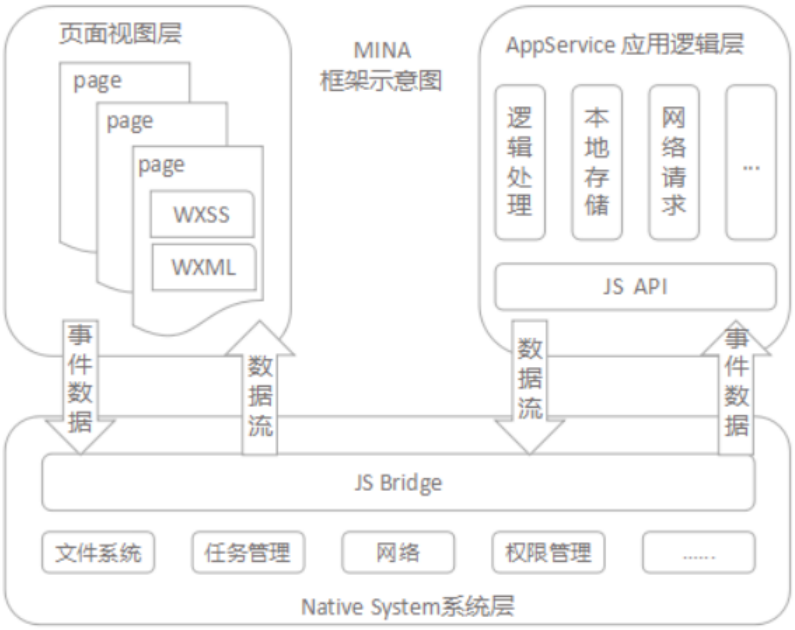
2.视图层

对于微信小程序而言，视图层就是所有的.wxml（WeiXin Markup language）文件与.wxss（WeiXin Style Sheet）文件的集合：.wxml 用于描述页面结构而.wxss 用于描述页面样式。

视图层以给定的样式来展现数据并反馈事件给逻辑层，而数据展现是以组件来进行的。组件（Component）是视图的基本组成单元。

目录结构

框架示意图



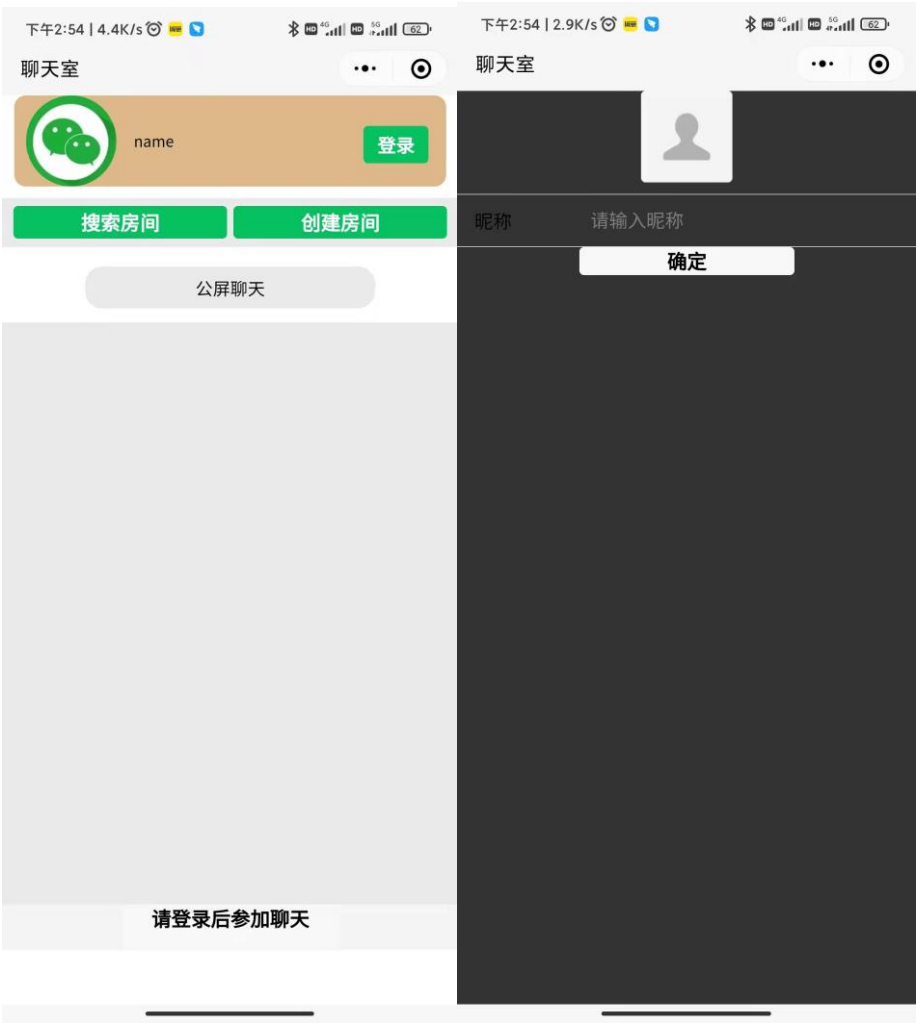
二、系统各功能模块详细设计与实现

（一）登录功能

这是主页未登录的情况下不能聊天，点击登录可以看到是新版的获取昵称头像。

```
<button size="mini" type="primary" bindtap="login" wx:if="{{login}}">登录</button>
```

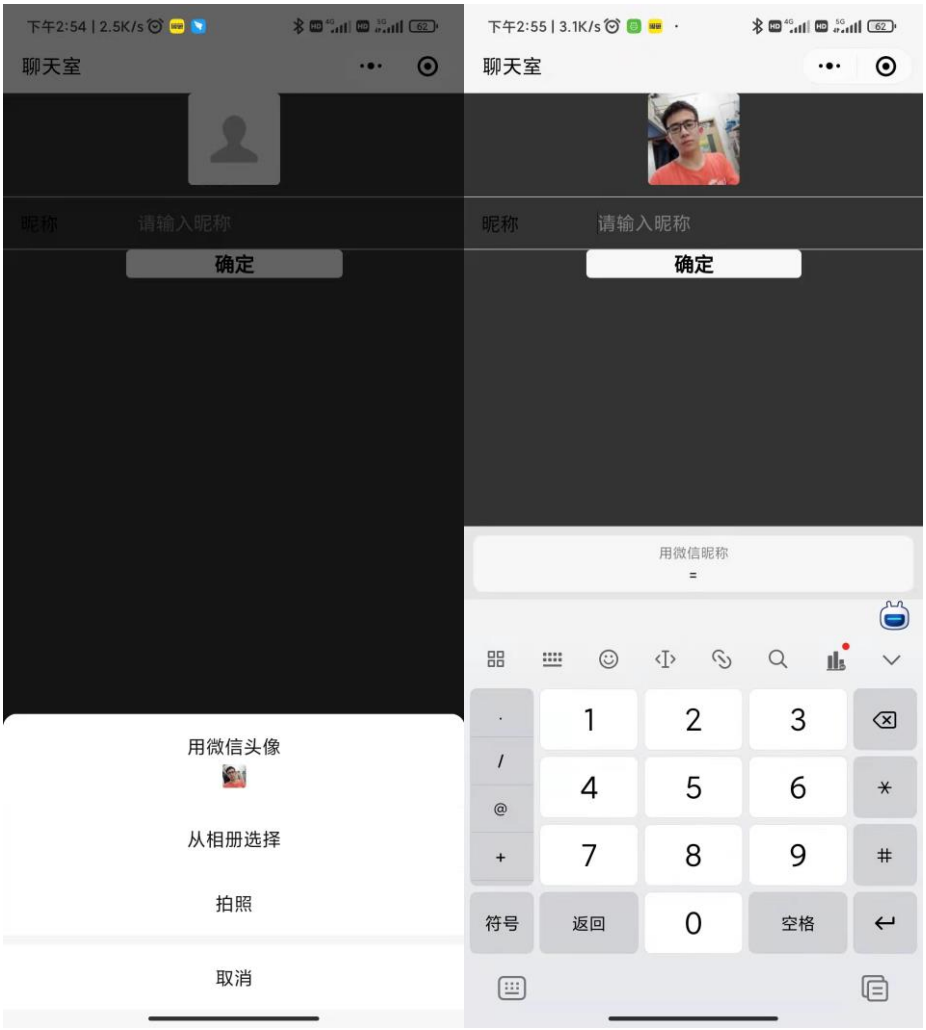
```
login(e) {
  this.setData({
    login:false
  })
  wx.reLaunch({
    url: '../login/login'
  })
},
```



点击头像和昵称可以获取微信头像和微信昵称

```
<button class="avatar-wrapper" open-type="chooseAvatar" bind:chooseavatar="onChooseAvatar">
  <image class="avatar" src="{{avatarUrl}}" mode='widthFix'></image>
</button>
```

```
<view class="input">
  <text>昵称</text>
  <input bindinput="getName" type="nickname" class="weui-input" placeholder="请输入昵称" />
</view>
<button bindtap="upload">确定</button>
```



注意：这里获取到的头像地址是本地地址，只能自己看到别人看不到

所以在点击确定的时候

```
onChooseAvatar(e) {
  let avatarUrl = e.detail.avatarUrl
  //console.log(e.detail.avatarUrl)
  app.globalData.avatarUrl = e.detail.avatarUrl
  this.setData({
    avatarUrl: avatarUrl,
  })
},
getName: function (e) {
  //console.log(e.detail.value)
  this.setData({ name: e.detail.value })
},
```

我将头像上传到云存储中，这样别人就也可以看到你的头像

```
upload: function (e) {
  const filePath = app.globalData.avatarUrl
  const cloudPath = app.globalData.openid
  wx.cloud.uploadFile({
    cloudPath,
    filePath,
    success: res => {
      wx.showToast({
        title: '上传成功',
        duration: 3000
      })
      console.log(res.fileID)
```

本地地址

上传到云端的云地址

必须填写头像和昵称不然无法登录

```
console.log(openid)
if (avatarUrl == undefined || name == undefined || name == '') {
  avatarUrl = undefined
  name = undefined

  wx.showToast({
    title: '头像或昵称未填写',
    icon: 'none',
    duration: 3000
  })
} else {
```

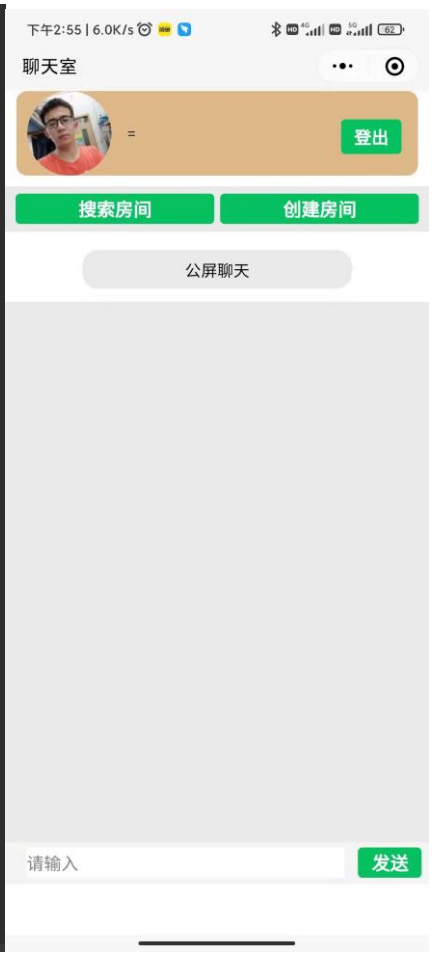
user_name_photo 为用户数据库，获得到信息后要先去用户数据库中验证，查看是否存在该用户的用户信息，如果不存在直接用 .add({}) 往数据库中添加用户信息。（user_login 为用户登录标志位，已登录置 1，未登录置 0，后面讲登出会讲到）

```
} else {
  user_name_photo.where({
    _openid: openid
  }).get({
    success: res => {
      //console.log(res)
      console.log(res.data)
      let user = res.data
      if (user == '') {
        user_name_photo.add({
          data: {
            avatarUrl: avatarUrl,
            avatar: filename,
            nickName: name,
            addDate: today,
            user_login: 1
          },
        })
```

如果存在用户信息就用.update({})更新用户信息，因为用户可能会变更头像或昵称

之后会通过 api wx.reLaunch({}) 跳转回首页

```
    } else if (user[0]._openid == openid) {
      user_name_photo.where({
        _openid: openid
      })
        .update({
          data: {
            avatarUrl: avatarUrl,
            avatar: filename,
            nickName: name,
            addDate: today,
            user_login: 1
          },
        })
    } else {
      wx.showToast({
        title: '联系作者',
        icon: 'none',
        duration: 3000
      })
    }
  }
})
wx.reLaunch({
  url: '../index/index?id=' + app.globalData.openid
})
```



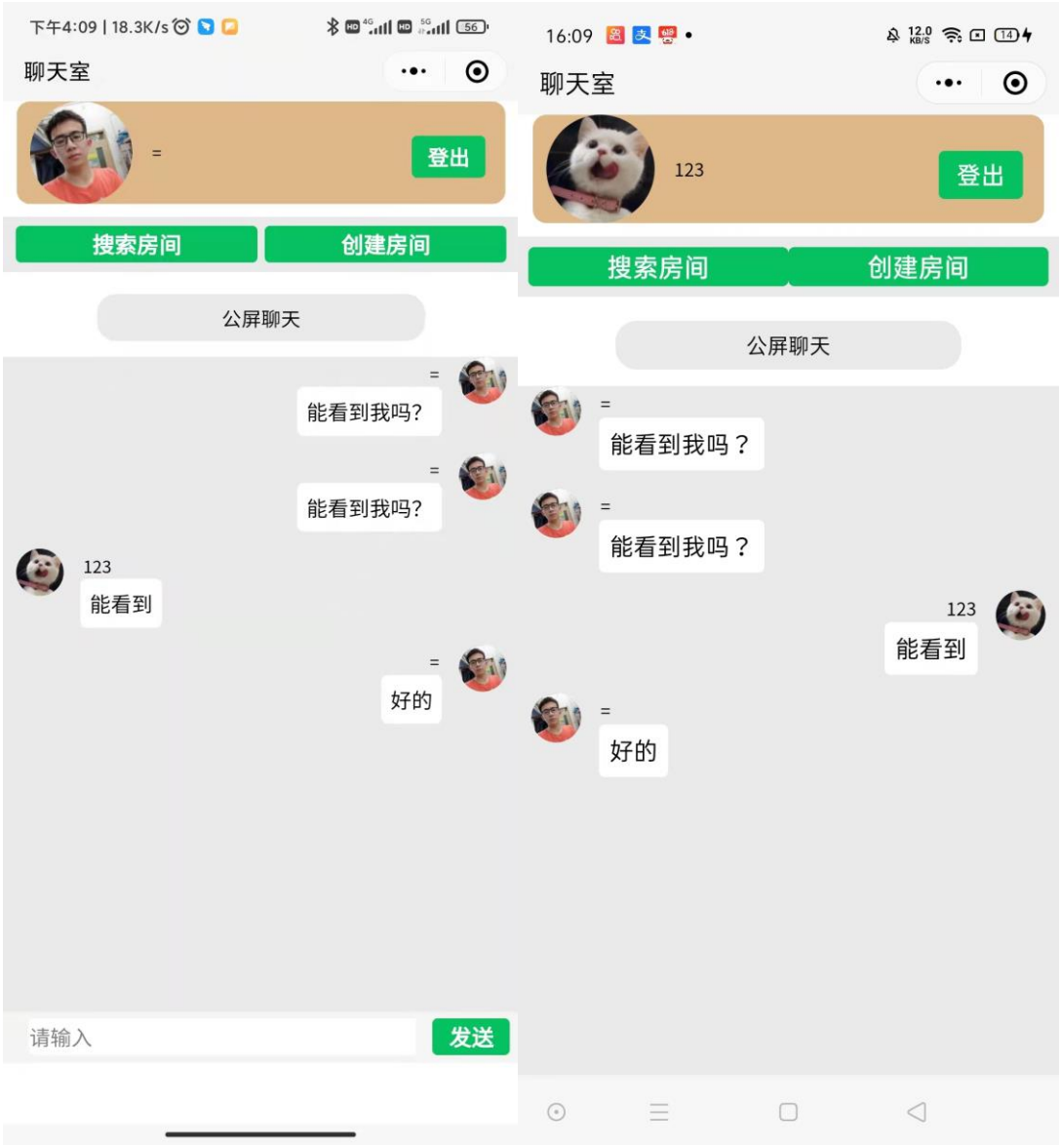
跳转回首页后我们发现永辉已登录

点击登出会将头像昵称恢复默认，会将登录标志位 user_login 置 0

```
exit() {
  this.setData({
    avatarUrl: '../../images/logo.png',
    nickName: 'name',
    login: true
  })
  let openid = this.data.openid
  user_name_photo.where({
    _openid: openid
  }).update({
    data: {
      user_login: 0
    },
  })
},
```

（二）公屏聊天功能

接下来是主页的公屏聊天部分，这个聊天记录是所有人都可以看到的




```
<input class="message-sender-input" type="text-input" placeholder="请输入"
bindinput="onTextInput" value="{{talkabout}}"></input>
<button class="send-btn" type="primary" bindtap="onSend">
```

Input 输入框中输入内容

按发送会触发 onsend 事件

```
onTextInput: function (e) {
  this.setData({ talkabout: e.detail.value })
  //console.log(e.detail.value)
},
onSend() {
  let openid = this.data.openid
  //console.log('index',openid)
  //this.pageScrollToBottom()
  let talkabout = this.data.talkabout;
  //console.log(talkabout)
  if (!talkabout) {
    return
  }
  let sendtime = formatDate()
  user_name_photo.where({
    _openid: openid
  }).get({
    success: res => {
```

如果输入框为空则返回

在用户数据库中查找当前用户并获取其信息

获取到信息后将信息上传到世界聊天记录数据库中

```
let sendtime = formatDate()
user_name_photo.where({
  _openid: openid
}).get({
  success: res => {
    //console.log(res.data)
    const doc = {
      copy: true,
      avatarUrl: res.data[0].avatar,
      nickName: res.data[0].nickName,
      msgText: 'text',
      talkabout: talkabout,
      date: sendtime,
      sendTime: new Date(),
      sendTimeTS: Date.now(),
    }
    talk.add({
      data: doc,
    })
```

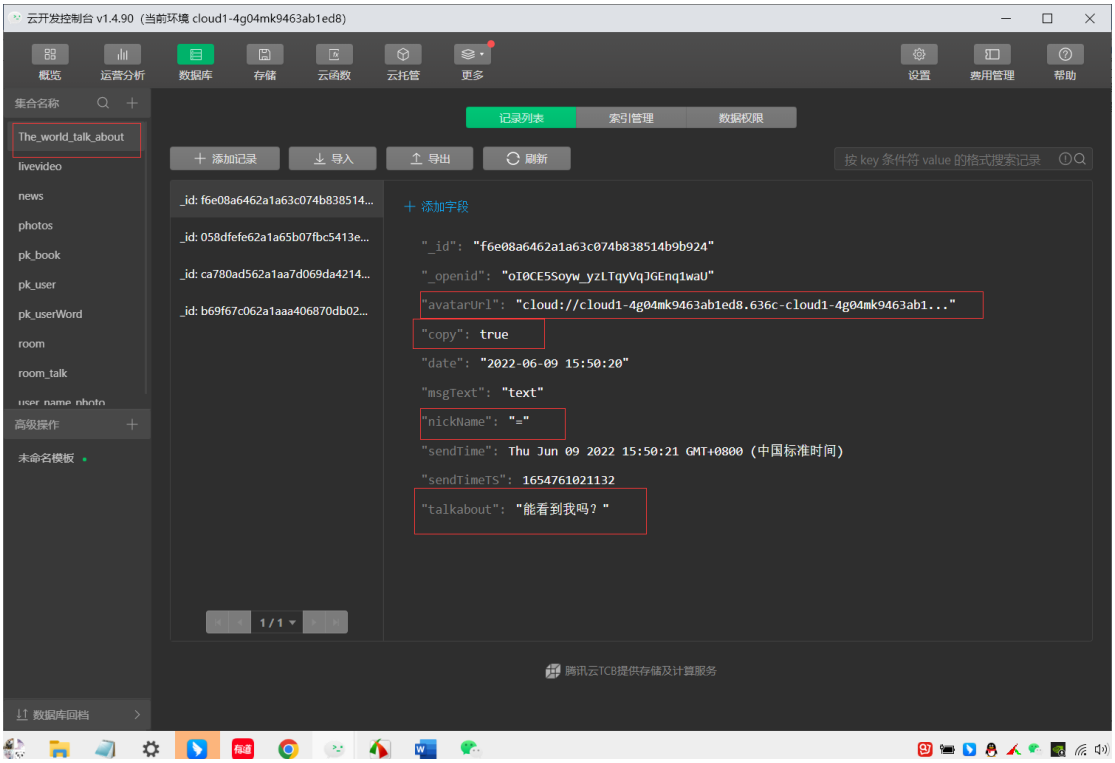
复制标志位

用户头像

用户昵称

说的话

```
const talk = db.collection('The_world_talk_about')
```



之后再实时数据推送

.watch({onChange:function(e){}})

将数据实时显示在页面中

```
const that = this
talk.orderBy('sendTimeTS', 'asc').watch({
  onChange: function (snapshot) {
    // console.log('item',snapshot.docs.length - 1)
    // let toLast= 'item'+(snapshot.docs.length - 1)
    // console.log(toLast)
    that.setData({
      stars: snapshot.docs,
      toLast: 'item' + (snapshot.docs.length - 1)
    })
    //that.pageScrollToBottom()
```

实时显示的数据

滚动论定位用的

scroll-into-view 后接标签的 id，作用是将滚轮定位到相关 id 的标签上。我将 stars 上循环出的每一个标签都设置上 id=”item{{index}}”,这样假设有 14 条消息记录，那么第 14 条消息记录的 id 就是 item13，发送第 15 条消息时，toLast 会回传 item14，而发送的第 15 条消息的标签 id 正好就是 item14，这样滚轮就会定位

到第 15 条消息的标签上

```
<view class="world"><text>公屏聊天</text></view>
<view class="chatroom">
  <scroll-view scroll-y="true" class="lit_body" id="scroll_view" scroll-into-view="{
  {toLast}}">
    <view class="card" wx:for="{{stars}}" wx:key="index">
      <view class="left" wx:if="{{openid!=item._openid}}" id="item{{index}}">
        <view class="left-card-head">
          <view class="left-img">
            <image class="avatar" src='{{item.avatarUrl}}' mode='widthFix'></image>
          </view>
          <view class="left-title">
            <text class="nickName">{{item.nickName}}</text>
          </view>
        </view>
        <view class="card-body">
          <text class="left-text" user-select="{{item.copy}}"bindlongpress ='copy'
          data-copy='{{item.talkabout}}'>{{item.talkabout}}</text>
        </view>
      </view>
      <view class="right" wx:if="{{openid==item._openid}}" id="item{{index}}">
        <view class="right-card-head">
          <view class="right-img">
            <image class="avatar" src='{{item.avatarUrl}}' mode='widthFix'></image>
          </view>
          <view class="right-title">
            <text class="nickName">{{item.nickName}}</text>
          </view>

        </view>
        <view class="right-card-body">
          <text class="right-text" user-select="{{item.copy}}"bindlongpress ='copy'
          data-copy='{{item.talkabout}}'>{{item.talkabout}}</text>
        </view>

      </view>

    </view>

  </scroll-view>
```

我写了两遍消息的循环标签分别对应左和右，如果消息记录的 openid 和当前用户的 openid 一样那就会在右边显示，反之就会在左边显示。

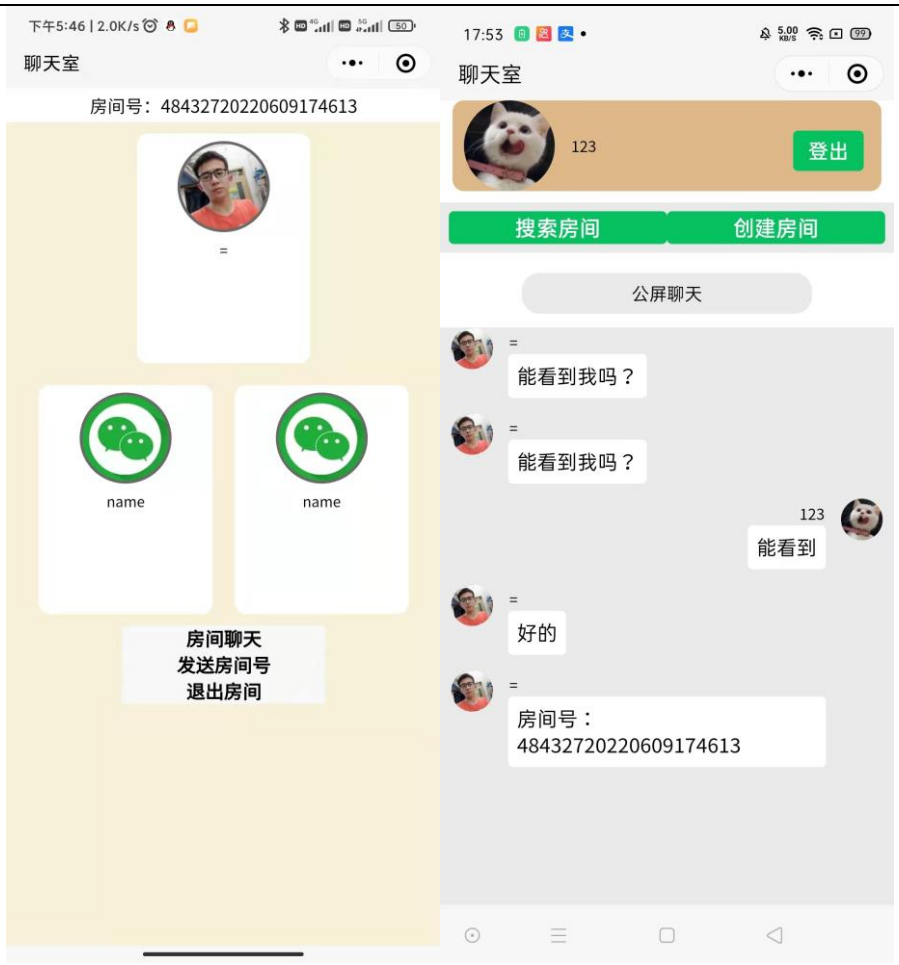
（三）创建房间功能

点击创建房间会触发 buildroom 事件

```
<button type="primary" bindtap="buildroom">创建房间</button>
```

首先从 user_name_photo 用户数据库中获取创建者的信息，然后再随机生成一个房间号，接着把用户信息和房间号一同写入 room 房间数据库中，然后再跳转到房间页面

```
buildroom: function () {
  let openid = this.data.openid
  //console.log('buildroom',openid);
  user_name_photo.where({ _openid: openid }).get({
    success: function (res) {
      //console.log(res.data)
      // console.log(Math.random() * 1000000)
      // console.log(res.data[0].addDate)
      // let a = res.data[0].addDate
      // console.log(a.match(/\d+/g).join(''))
      const roomnumber = Math.floor(Math.random() * 1000000) + formatDate().match(/\d
      +/g).join('')
      //console.log(roomnumber)
      room.add({
        data: {
          _room_number: roomnumber,
          users: {
            0: {
              avartr: res.data[0].avatar,
              nickName: res.data[0].nickName,
              userid: openid,
            },
            1: null,
            2: null,
          }
        },
      })
      wx.reLaunch({
        url: '../room/room?id=' + roomnumber,
      })
    }
  })

},
onRoomInput: function (e) {…
```



点击发送房间号会直接将消息发送到主页面的公屏聊天里

点击 sendnumber 会触发 sendnumber 事件

```
<button bindtap="sendnumber">发送房间号</button>
```

room.js 页面里的 sendnumber 与 index.js 里面的 onsend 是差不多的，只是 room.js 页面里的 sendnumber 只能发送固定的内容且复制标志位是 false。无论 room.js 页面里的 sendnumber 还是 index.js 里面的 onsend，他们都是将消息写入了 The_world_talk_about 数据库当中

```
sendnumber: function () {
  let openid = app.globalData.openid
  console.log('index', openid)
  //this.pageScrollToBottom()
  let talkabout = '房间号: ' + this.data.number;
  //console.log(talkabout)
  if (!talkabout) {
    return
  }
  let sendtime = formatDate()
  user_name_photo.where({
    _openid: openid
  }).get({
    success: res => {
      //console.log(res.data)
      const doc = {
        copy: false,
        avatarUrl: res.data[0].avatar,
        nickName: res.data[0].nickName,
        msgText: 'text',
        talkabout: talkabout,
        date: sendtime,
        sendTime: new Date(),
        sendTimeTS: Date.now(),
      }
      talk.add({
        data: doc,
      })
    }
  })
},
```

（四）房间聊天功能

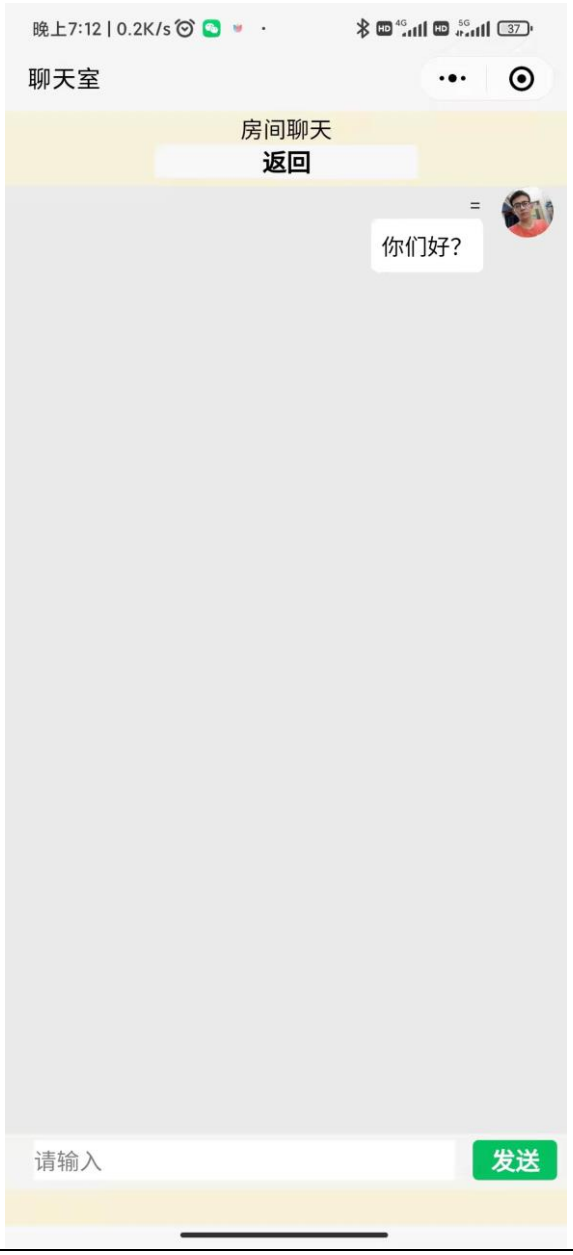
点击房间聊天会切换到另一个另一个标签页

```
<button bindtap="say">房间聊天</button>
```

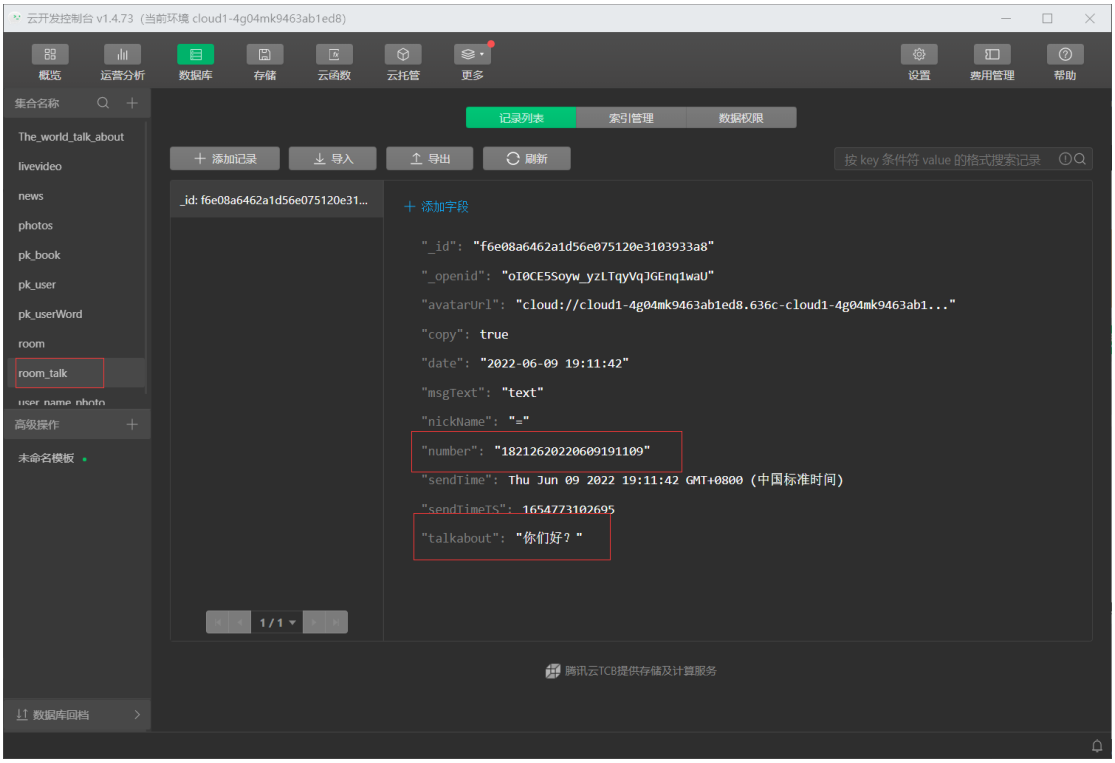
```
say: function () {
  this.setData({ about: false })

},
```

```
<view class="container" wx:if="{{about}}">
```

```
<view class="container"wx:if="{{!about}}">
```



房间聊天与公屏聊天的逻辑实现大同小异，不同的地方在于房间聊天的聊天记录存储在一个叫 room_talk 的数据库里，而且每条记录都标注了房间号，用户在房间聊天里只能看到在同一房间里的人所发送的消息，不同房间的人看不到彼此的消息



筛选同一房间号的消息记录

```
room_talk.orderBy('sendTimeTS', 'asc').watch({
  onChange: function (snapshot) {
    //console.log(snapshot.docs.length);
    let j = 0
    for (let i = 0; i < snapshot.docs.length; i++) {
      //console.log(snapshot.docs[i].number);
      //console.log(number);
      if (snapshot.docs[i].number == number) {
        room_say[j] = snapshot.docs[i]
        j = j + 1
      }
    }
    that.setData({
      star: room_say,
      toLast: 'item' + (snapshot.docs.length - 1)
    })
    //that.pageScrollToBottom()
```

回到房间页，点击退出房间会触发 quit 事件

```
<button bindtap="quit">退出房间</button>
```

首先会获取 room 数据库中该房间号的房间信息，通过 for 循环和 if 语句获取该房间剩余的人数

```
quit: function (e) {
  //console.log(app.globalData.openid);
  let user_openid = app.globalData.openid
  let number = this.data.number
  room.where({ _room_number: number }).get({
    success: function (res) {
      let arr = new Array(3)
      //console.log(res.data[0].users);
      let a = 0
      for (let i = 0; i < 3; i++) {
        if (res.data[0].users[i] != null) {
          a = a + 1
        }
      }
      console.log(a);
      if (a <= 1) {
        room.where({ _room_number: number }).remove({
          success: function (res) {
            //              //that.show()
```

如果剩余人数是 1 人退出房间的话会直接 remove 删除这个房间，同时也会删除该房间的聊天消息记录

```
for (let i = 0; i < 3; i++) {
  if (res.data[0].users[i] != null) {
    a = a + 1
  }
}
console.log(a);
if (a <= 1) {
  room.where({ _room_number: number }).remove({
    success: function (res) {
      //              //that.show()
      //console.log(res)
    }
  })
  room_talk.where({ number: number }).remove({
    success: function (res) {
      //              //that.show()
      console.log(res)
    }
  })
} else {
```

首先用将该房间内的所有用户信息传递到一个 arr 数组内，然后用 for 循环从 0 位置开始判断是否有人，如果没人（null）位置加一开始判断下一位置，如果有人则判断这个位置上的人是不是你，如果不是你开始下一次循环，如果是你，则在对应位置上的 arr 数组置 null，然后.update({})将 arr 数组更新到数据库上

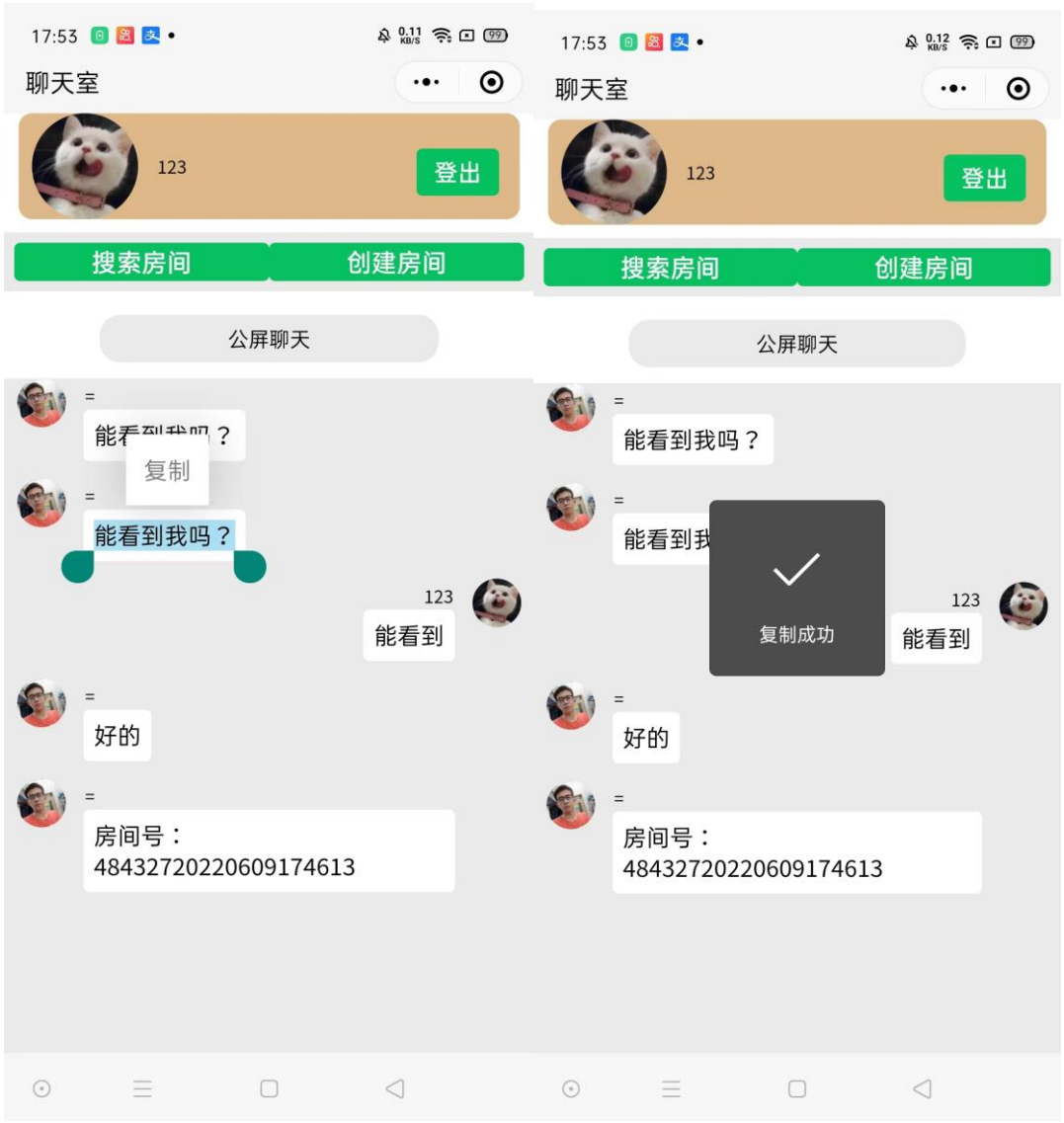
```
} else {
  for (let i = 0; i < 3; i++) {
    arr[i] = res.data[0].users[i]
  }
  console.log(arr);
  console.log(user_openid);
  for (let i = 0; i < 3; i++) {
    console.log(res.data[0].users[i]);
    if (res.data[0].users[i] == null) {
      continue
    } else if (res.data[0].users[i].userid == user_openid) {
      arr[i] = null
      console.log('dafdaf', i);
      room.where({ _room_number: number }).update({
        data: {
          users: arr
        },
        success: function (res) {
          //console.log(res.data)
        }
      })
```

然后返回到主界面

```
wx.reLaunch({
  url: '../index/index'
})
```

（五）消息复制功能

在主界面中，长按消息会触发复制选项，可以看到复制分为两种，一种就是普通的标签复制，而另一种则是 API 提供的复制功能，二者由前面提到的复制标志位控制。



公屏聊天发送的消息会被置为 ture


```
"_id": "f6e08a6462a1a63c074b838514b9b924"
"_openid": "oI0CE5Soyw_yzLTqyVqJGEnq1waU"
"avatarUrl": "cloud://cloud1-4g04mk9463ab1ed8.636c-cloud1-4g04mk9463ab1..."
"copy": true
"date": "2022-06-09 15:50:20"
"msgText": "text"
"nickName": "="
"sendTime": Thu Jun 09 2022 15:50:21 GMT+0800 (中国标准时间)
"sendTimeTS": 1654761021132
"talkabout": "能看到我吗？"
```

而从房间发出来的房间号会被置为 false

```
"_id": "0ab5303b62a1c2b50887609631be100e"
"_openid": "oI0CE5Soyw_yzLTqyVqJGEnq1waU"
"avatarUrl": "cloud://cloud1-4g04mk9463ab1ed8.636c-cloud1-4g04mk9463ab1..."
"copy": false
"date": "2022-06-09 17:51:49"
"msgText": "text"
"nickName": "="
"sendTime": Thu Jun 09 2022 17:51:49 GMT+0800 (中国标准时间)
"sendTimeTS": 1654768309687
"talkabout": "房间号：48432720220609174613"
```

他们由 user-select 标签和 copy 事件联合控制，如果是 true 则会触发 user-select 标签，如果是 false 则会触发 copy 事件

```
<text class="right-text" user-select="{{item.copy}}"bindlongpress ='copy'
data-copy='{{item.talkabout}}'>{{item.talkabout}}</text>
```

```
var code = e.currentTarget.dataset.copy;
talk.orderBy('sendTimeTS', 'desc').where({ copy: false }).get({
  success: function (res) {
    //console.log(res.data.length);
    for (let i = 0; i < res.data.length; i++) {
      if (code == res.data[i].talkabout) {
        code = code.match(/\d+/g).join('')
        //console.log(code);
        wx.setClipboardData({
          data: code,
          success: function (res) {
            //console.log(res)
            wx.showToast({
              title: '复制成功',
            });
          },
          fail: function (res) {
```

（六）房间搜索功能

复制完房间号后，我们粘贴房间号点击加入会触发两个事件 onRoomInput 和 joinroom


```
<input type="text-input" placeholder="请输入" bindinput="onRoomInput" value=""></
input>
<button type="primary" size="mini" bindtap="joinroom">加入</button>
```

onRoomInput 是获取房间号

```
onRoomInput: function (e) {
  this.setData({ room_number: e.detail.value })
},
```

```
joinroom: function () {
  let openid = this.data.openid
  //console.log('joinroom',openid);
  let number = this.data.room_number
  let c = 1
  var b;
  for (var i = 0; i < number.length; i++) {
    b = number % 10;
    if (b != 0 && b != 1 && b != 2 && b != 3 && b != 4 && b != 5 && b != 6 && b != 7 &
    & b != 8 && b != 9) {
      c = 0;
    }
  }
  if (c == 0) {
    wx.showToast({
      title: '不能为字母,特殊符号',
      icon: 'none'
    });
  } else if (c != 0) {
```

如上图首先 joinroom 会先判断你输入的房间号是否全为数字，如果不是，则返回提示信息。

如下图你输入的是一串纯数字，然后它会先获取你的个人信息，因为你已经登陆了所以是从数据库里。然后它用你输入的房间号在 room 数据库中寻找对应房间号的房间，如果没有找到则会提示房间不存在

```
} else if (c != 0) {
  user_name_photo.where({ _openid: app.globalData.openid }).get({
    success: function (res) {
      //console.log(res.data)
      let avartr = res.data[0].avatar
      let nickname = res.data[0].nickName
      room.where({ _room_number: number }).get({
        success: function (res) {
          //console.log(res.data[0].users)
          if (res.data.length == 0) {
            //console.log('该房间不存在');
            wx.showToast({
              title: '该房间不存在',
              icon: 'none'
            });
          } else if (res.data[0]._room_number == number) {
```

```
} else if (res.data[0]._room_number == number) {
  let list = res.data[0].users
  let user = new Array(3)
  let person = 0
  for (let i = 0; i < 3; i++) {
    if (list[i] != null && list[i].avartr != "" && list[i].nickName !=
    "") {
      user[i] = {
        avartr: list[i].avartr,
        nickName: list[i].nickName,
        userid: list[i].userid,
      }
      person += 1
    }
  }
```

如上图，如果通过你输入的房间号找到了对应的房间，则会用 for 循环把房间内的用户信息记录到 user 数组里，同时记录人数 person

如下图如果人数已达上限则会提示该房间人数已满，如果还有空位则会把你的信息记录到 user 数组里，然后 user 数组再把值传递给 room 数据库，同时会跳转到房间页

```
      if (person >= 3) {
        wx.showToast({
          title: '该房间人数已满',
          icon: 'none'
        });
      } else {
        wx.reLaunch({
          url: '../room/room?id=' + number,
        })
        //console.log(list)
        for (let i = 0; i < 3; i++) {
          //console.log(list[i])
          if (list[i] == null || list[i].avartr == "" && list[i].nickName ==
          "") {
            user[i] = {
              avartr: avartr,
              nickName: nickname,
              userid: openid,
            }
            //console.log(user)
            room.where({ _room_number: number }).update({
              data: {
                users: user
              }
            })
            break;
          }
        }
      }
    }
```

实践总结与体会：

时光飞逝转眼间一学期又过去了《微信小程序》这门课也迎来了尾声，在做结课设计的时候遇到了很多困难，最后也解决了问题。作品里还有一些 bug，有的找到了有的没找到，比如在未登录的条件下可以创建房间，这是刚刚发现的。还有就是页面不太美观，不是很上档次。功能呢相对于微信来说更是显得非常鸡肋，但是它用到的技术却很有前景。这次的设计是一场考试，同时也是毕设的练习，在后面的开发中我将应用更过的关于和实时数据推送有关的案例。

在这次开发过程中遇到了很多困难，让我印象最深刻的就是，新版获取头像的链接是本地链接，只能在自己这里显示，只有上传到云存储中，读取 fileID 才能让别人也看见你的头像。其他困难更多的是在逻辑上如何实现，就拿加入房间来说，我是把 room 数据库中一条记录当成了一个房间，那么加入房间就是把用户的信息塞入到这一条记录中，当然这样做起来非常麻烦。可当我完成这些逻辑实现后又有了新的思路，不用 room 数据库，在 user_name_photo 数据库中的用户数据上多加一条 roomnumber 房间，需要在房间里显示成员时直接用.where({roomnumber:房间号})检索相同房间号的人就行了。

还有一些我现在也没有学会的知识，希望在以后的学习和生活中我可以掌握这些知识。这个项目也培养了我的自学能力，做作品的时候出现的问题，很多都是通过百度，查书来解决的。很感谢老师的教导，同学的帮助。

实践指导教师评语：

实践成绩：优秀□ 良好□ 一般□ 及格□ 不及格□

签名：

年 月 日