

年级、专业、班级	2021 级计算机科学与技术卓越 2 班			姓名	文红兵	学号	20214590
提交时间	2023. 12. 6	学年学期	2023-2024（1）	指导教师		杨瑞龙	
软件名称	天天小说						
<p>说明：</p> <ol style="list-style-type: none">1. 每人独立完成移动应用软件设计开发，并完成设计报告。2. 自拟软件题目。不准抄袭，应当是自己独立编写的软件。抄袭计 0 分。3. 软件开发平台选择 HarmonyOS，优先使用 ArkUI 开发。4. 从界面、技术、功能等方面对软件进行评价。程序有必要的注释。5. 从文档条理性、文档规范性、内容完整性等方面对设计报告进行评价。6. 所开发的程序源代码压缩成 ZIP 格式，命名为：学号姓名-期末软件设计.zip。报告命名为：学号姓名-期末软件设计报告.docx。演示视频清晰，MP4 格式，大小不超过 20M，命名为：学号姓名-期末软件设计演示.mp4。请在谷歌或者搜狗浏览器预览视频是否能够正常播放。三个文件分别提交到云班课。							
<p style="text-align: center;">报告正文</p> <p><i>注：格式要规范，正文汉字一律使用宋体小四，文字行间距使用 1.5 倍行距。段首缩进 2 个汉字，图片居中，每个图片在底部有标题和编号。表格上面有表格标题和编号。版面漂亮、整洁、统一。不超过 20 页。</i></p>							
<p>一、软件设计意图、功能介绍和特色</p> <p>（一） 软件设计意图</p> <p>该灵感来源于一位朋友，说现在小说收费啥的，然后我就有了设计该软件的想法，写一个完全免费的，包含全网几乎所有小说的 app。</p> <p>（二） 特色</p> <ul style="list-style-type: none">● 前后端分离架构：采用前端框架 arkts 和 arkui 设计，后端使用 golang 的 iris、xorm 和 viper 等框架。这种架构使得前后端开发可以独立进行，							

提高了开发效率和代码的可维护性。

- **数据库和数据来源：**使用 MySQL 数据库存储小说数据。而且你提到你自己编写了爬虫程序来采集数据，这意味着我的 APP 能够从互联网上获取全网几乎所有小说的免费阅读内容，这是一个很大的特色，为用户提供了广泛的阅读选择。
- 具有非常重要的现实性和实用性。

（三） 功能介绍

- 书城功能

随机推荐，加载提示，搜索功能，书籍详细信息

部分截图：

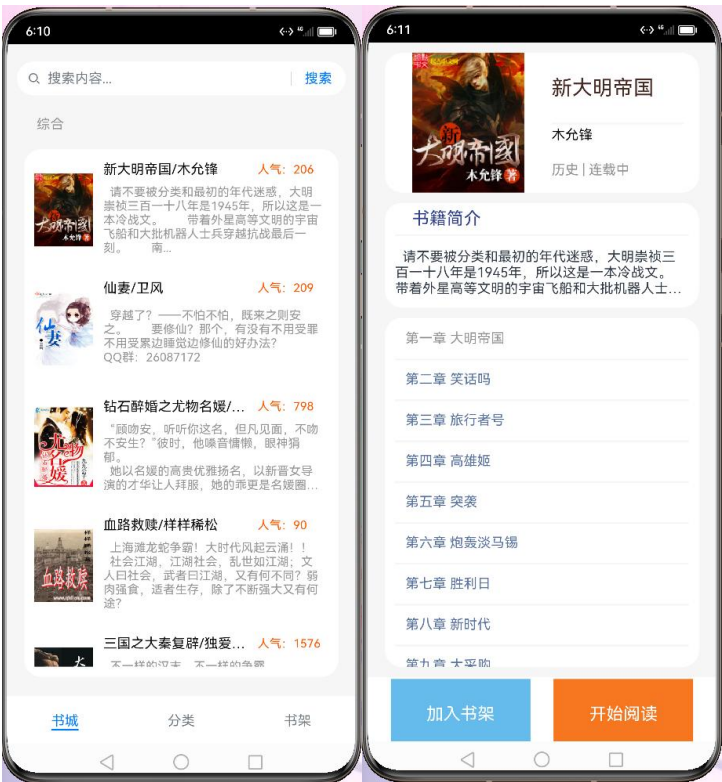


图 1 书城功能

- 分类功能：

七大类型与个性化浏览

部分截图：



图 2 分类功能

3. 书架功能：

我的书架与历史记录

部分截图：

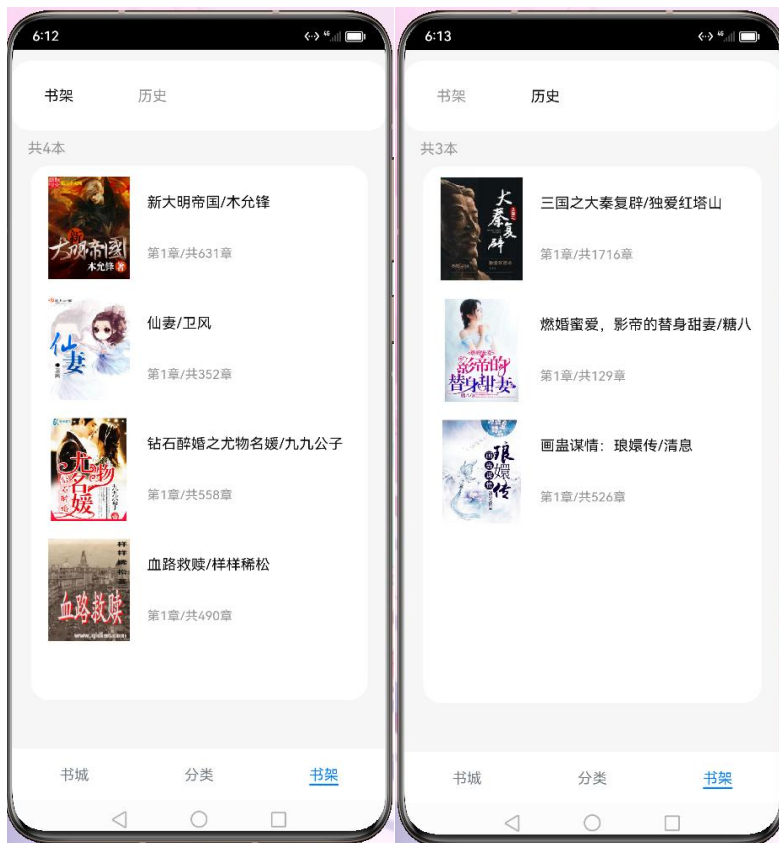
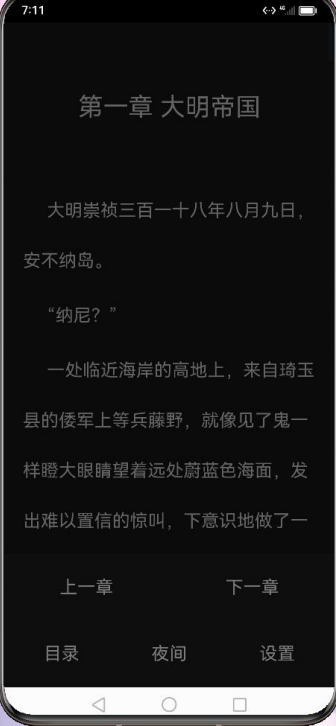
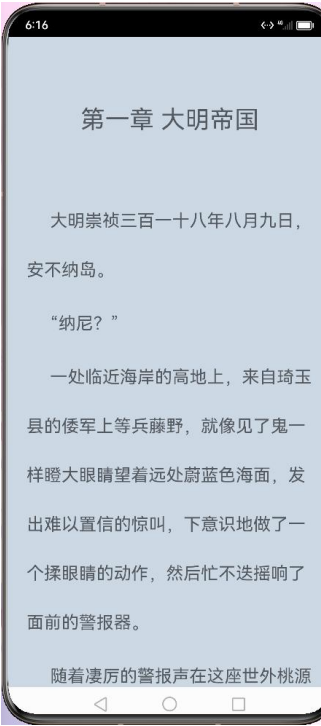
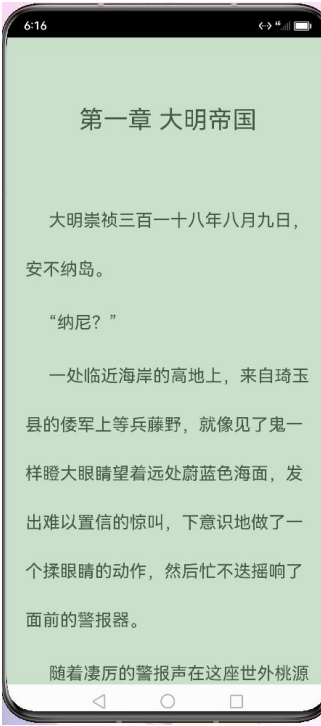
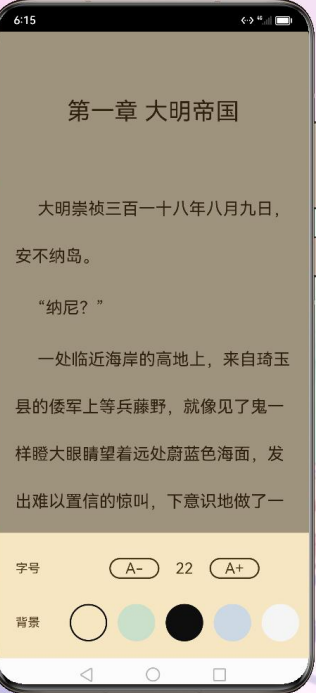
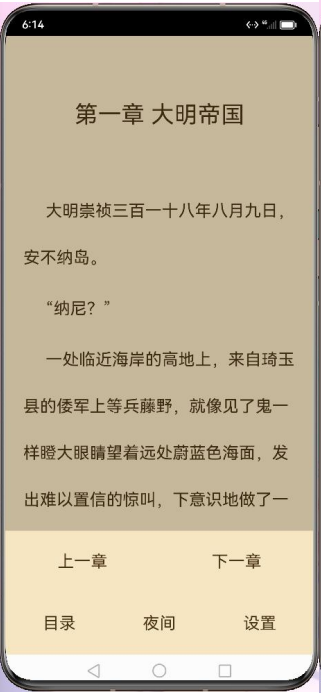
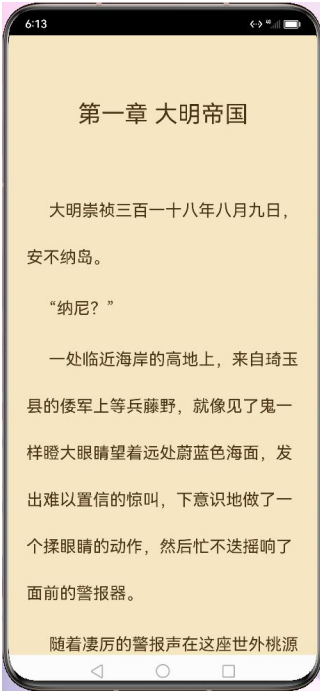


图 3 书架功能

4. 阅读功能：

多种入口，阅读设置，章节跳转

部分截图：



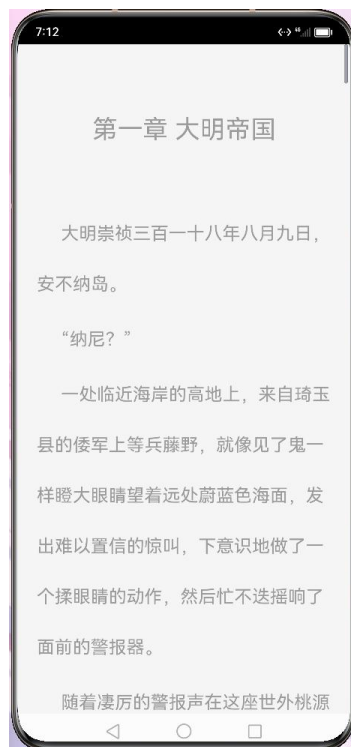


图 4 阅读功能

二、开发和运行环境

（一）后端运行环境

1. golang 1.21 版本
2. log 包版本为 1.21 版本
3. iris 框架 v12 版本
4. swagger 框架 iris-v12 版本
5. golang mysql 驱动最新版
6. xorm 最新版本
7. viper 最新版本
8. Mysql 8.0
9. GoLand 2022.2.4
10. DataGrip 2021.2.2

（二）前端运行环境

11. DevEco Studio

12. 项目版本 API9 以上

三、设计说明

（一）设计思想

前后端分离架构：采用前端框架 **arkts** 和 **arkui** 设计，后端使用 **golang** 的 **iris**、**xorm** 和 **viper** 等框架。这种架构使得前后端开发可以独立进行，提高了开发效率和代码的可维护性。

（二）界面设计

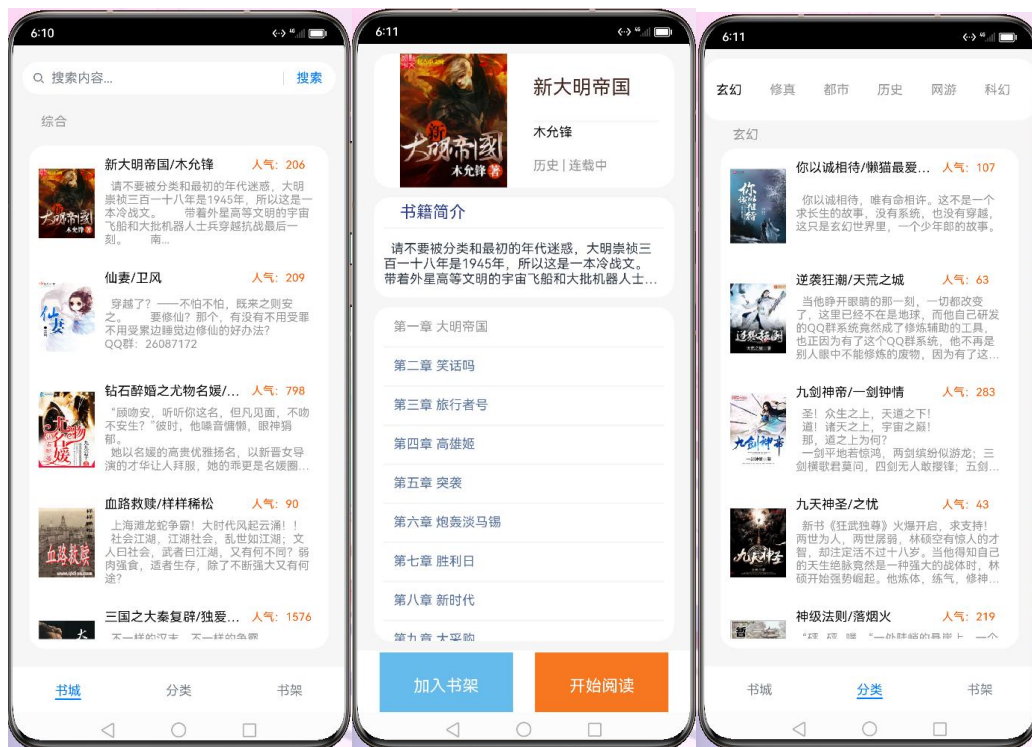






图 5 界面设计

(三) 程序结构于功能关系图

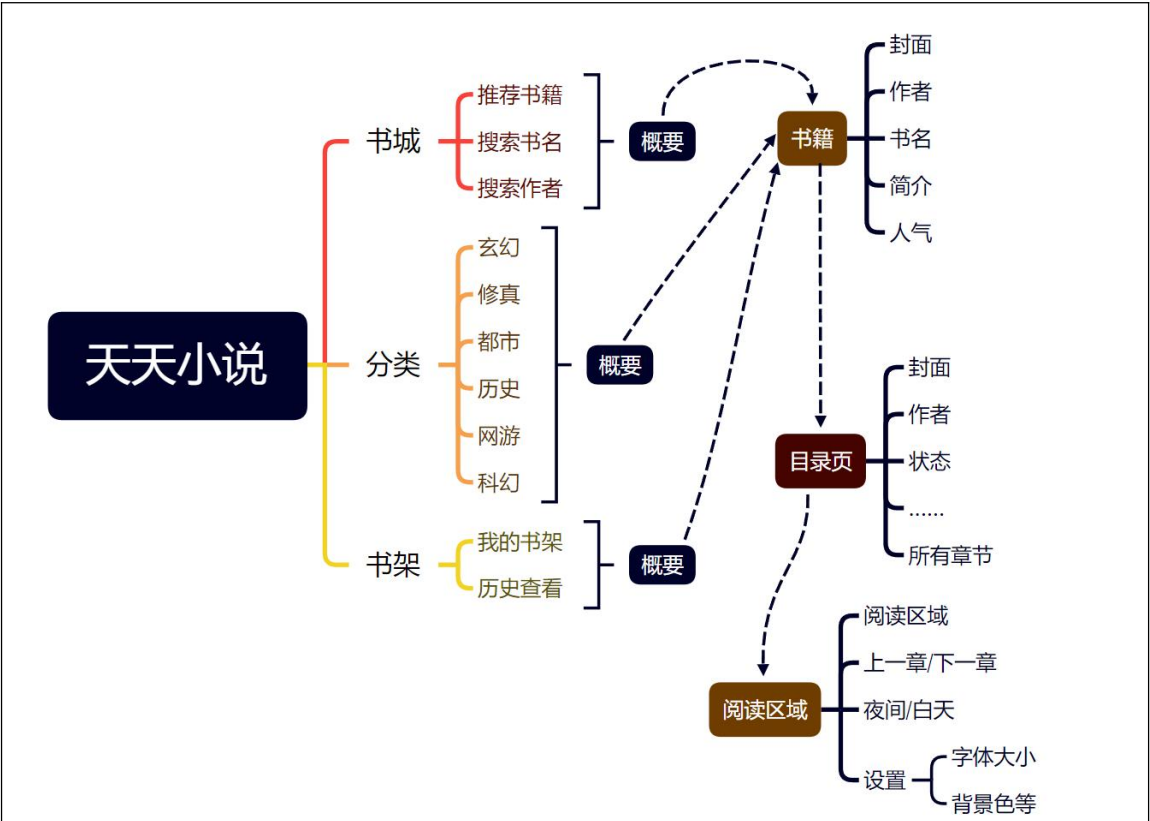


图 6 功能关系图

（四） 核心源代码

● 爬虫代码

```

15 func main() {
16     endNum := 1391
17     host := "https://www.biqugecn.com/"
18     const layout = "2006-01-02 15:04:05"
19     file, err := os.OpenFile( name: "logfile.txt", os.O_APPEND|os.O_CREATE|os.O_WRONLY, perm: 0644)
20     if err != nil {...}
21     defer file.Close()
22     log.SetOutput(file) // 将日志输出目标设置为文件
23     group := &sync.WaitGroup{} // 控制主线程和协程一起结束
24     semaphore := make(chan struct{}, 3) // 并向数，不能设置太大了，服务器容易崩
25     for i := 1; i <= endNum; i++ {
26         group.Add( delta: 1)
27         semaphore <- struct{}{}
28         go func(index int) {
29             defer func() { <-semaphore }()
30             defer group.Done()
31             addr := host + "toptoptime/" + cast.ToString(index) + ".html"
32             html := R.Get(addr)
33             html = R.ConvertEncodingToUTF8(html)
34             matches := R.FindAll(html, regex: `<tr><td class="hidden-xs">(.*?)</td><td>`+
35                 `<a href="(.*?)" target="_blank">(.*?)</a></td><td class="hidden-xs"><a `+
36                 `href="(.*?)" target="_blank">(.*?)</a></td><td>(.*?)</td><td>(.*?)</td></tr>`)
37             for _, match := range matches {
38                 func() {
39                     defer func() {...}()
40                     category := match[1]
41                     addr = match[2]
42                     name := match[3]
43                     latest := match[4]
44                     author := match[5]
45                     updateTime := match[6]
46                     updatetime, _ := time.Parse(layout, updateTime)

```

```

54     if dao.BookQuery(name) : 0
55         html = R.Get(addr)
56         html = R.ConvertEncodingToUTF8(html)
57         imageUrl := R.FindSignal(html, regex: `[1] // imageUrl
58         info2 := R.FindSignal(html, regex: `人气: (.*?)</span><.*?>(.*?)</span>`) // renqi, status
59         detail := R.FindSignal(html, regex: `<p class="text-muted" id="bookIntro".*?>(.*?)</p>`)[1]
60         detail = strings.ReplaceAll(detail, old: " ", new: "")
61         detail = strings.ReplaceAll(detail, old: "&nbsp;", new: " ")
62         detail = strings.ReplaceAll(detail, old: "<br/>", new: "\n")
63         html = R.FindSignal(html, regex: `<div class="panel panel-default" id="list-chapterAll">(.*?)</html>`)[1]
64         chapters := R.FindAll(html, regex: `<dd class="col-md-3"><a href="(.*?)".*?</a></dd>`)
65         book := &model.Book{
66             BookName:      name,
67             BookAuthor:     author,
68             BookDetail:     detail,
69             BookImage:      imageUrl,
70             BookLatestChapter: latest,
71             BookRenqi:       info2[1],
72             BookStatus:     info2[2],
73             BookTags:       category,
74             BookUpdateTime: updatetime,
75             BookPage:       addr,
76             BookAllChapterNum: len(chapters),
77         }
78         dao.BookInsert(book)
79     }()
80 }
81 }
82 }
83 }
84 }
85 group.Wait()
86 }

```

图 7 爬虫代码

● 获取章节内容

```
36 app.Post( relativePath: "/book/chapter/content", func(ctx *context.Context) {
37     log.Printf( format: "API:/book/chapter/content Be Called\n")
38     defer func(ctx *context.Context) {...}(ctx)
45     book := struct {
46         BookName      string
47         BookPage        string
48         BookReadChapter int
49     }{}
50     _ = ctx.ReadJSON(&book)
51     addr := book.BookPage
52     index := book.BookReadChapter
53     html := R.Get(addr)
54     html = R.ConvertEncodingToUTF8(html)
55     html = R.FindSignal(html, regex: '<div class="panel panel-default" id="list-chapterAll">(.*)</html>')[1]
56     chapters := R.FindAll(html, regex: '<dd class="col-md-3"><a href="(.*)" title="(.*)">(.*)</a></dd>')
57     addr = book.BookPage + chapters[index-1][1]
58     title := chapters[index-1][2]
59     html = R.Get(addr)
60     html = R.ConvertEncodingToUTF8(html)
61     content := ""
62     ret := R.FindSignal(html, regex: '<div.*?id="htmlContent".*>(.*)</div>')
63     for true {
64         content += ret[1]
65         if !strings.Contains(ret[1], substr: "本章未完，点击下一页继续阅读") {...}
66         ret = R.FindSignal(html, regex: '<a id="linkNext" class="btn btn-default" href="(.*)">下一页</a>')
67         addr = book.BookPage + ret[1]
68         html = R.Get(addr)
69         html = R.ConvertEncodingToUTF8(html)
70         ret = R.FindSignal(html, regex: '<div.*?id="htmlContent".*>(.*)</div>')
71     }
72     content = strings.ReplaceAll(content, old: '<p class="text-danger text-center bg0">本章未完，点击下一页继续阅读</p>', new: "")
73     content = strings.ReplaceAll(content, old: '笔趣阁 www.biqugecn.com 最快更新<a href=".">+book.BookPage+'>+book.BookName+'</a>最新章节!', new: "")
74     content = strings.ReplaceAll(content, old: '<br>', new: "\n")
75     content = strings.ReplaceAll(content, old: '<br/>', new: "\n")
76     content = strings.ReplaceAll(content, old: '&nbsp;', new: " ")
77     err := ctx.JSON(result.DataResult(struct {
78         Title string `json:"title"`
79         Content string `json:"content"`
80     }){
81         Title: title,
82         Content: content,
83     })
84     if err != nil {
85     }
86 })
```

图 8 获取章节内容代码

● 获取所有章节

```
91 app.Post( relativePath: "/book/chapter/all", func(ctx *context.Context) {
92     log.Printf( format: "API:/book/chapter/all Be Called\n")
93     defer func(ctx *context.Context) {
94         if err := recover(); err != nil {
95             log.Printf("API:/book/chapter/all ERROR: #{err}\n")
96             ctx.JSON(result.FailedResult())
97             return
98         }
99     }(ctx)
100     book := struct {
101         BookName string
102         BookPage string
103     }{}
104     _ = ctx.ReadJSON(&book)
105     addr := book.BookPage
106     html := R.Get(addr)
107     html = R.ConvertEncodingToUTF8(html)
108     html = R.FindSignal(html, regex: '<div class="panel panel-default" id="list-chapterAll">(.*)</html>')[1]
109     chapters := R.FindAll(html, regex: '<dd class="col-md-3"><a href="(.*)" title="(.*)">(.*)</a></dd>')
110     ret := []string{}
111     for i := range chapters {
112         ret = append(ret, chapters[i][1])
113     }
114     ctx.JSON(result.DataResult(ret))
115 })
```

图 9 获取所有章节

- 前端主页面设计

```
98 build() {
99   Row() {
100     Column() {
101       Tabs({ barPosition: BarPosition.End }) {
102         TabContent() {BookCity({msg: $msg_city, books: $books_city})}
103         .tabBar('书城')
104
105         TabContent() {Category({msg: $msg_category, books: $books_category})}
106         .tabBar('分类')
107
108         TabContent() {BookShelf({books:$books_shelf, type: $type_shelf})}
109         .tabBar('书架')
110       }
111       .onChange((index: number)=>{...})
112     }
113     .width(FULL)
114   }
115   .height(FULL)
116 }
117 }
```

图 10 前端主页面代码

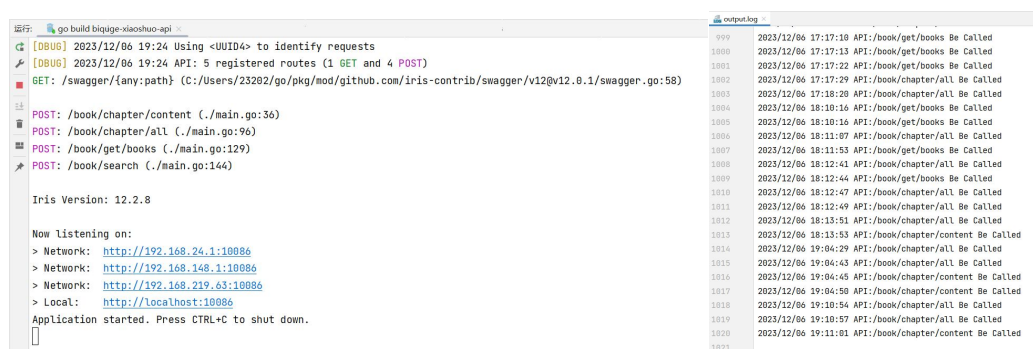
- 前端发起 Http 请求

```
6 export function RequestPost(url: string, obj: Object, callback?:Function) {
7   let httpRequest = http.createHttp();
8   httpRequest.on('headersReceive', (header) => {
9   });
10  httpRequest.request(
11    url, {
12      method: http.RequestMethod.POST,
13      header: {
14        'Content-Type': 'application/json'
15      },
16      extraData: obj,
17      expectDataType: http.HttpDataType.STRING,
18      usingCache: true,
19      priority: 1,
20      connectTimeout: 10000,
21      readTimeout: 10000,
22      usingProtocol: http.HttpProtocol.HTTP1_1,
23    }, (err, data) => {
24      if (!err) {
25        callback(JSON.parse(data.result as string))
26      } else {
27        httpRequest.off('headersReceive');
28        httpRequest.destroy();
29      }
30    }
31  );
32 }
```

图 11 前端发起 http 请求代码

四、源程序调试过程（运行、调试截图和文字）

（一） 后端运行截图



```
运行 go build biguge-xiaoshuo-api
[DBG] 2023/12/06 19:24 Using <UUID4> to identify requests
[DBG] 2023/12/06 19:24 API: 5 registered routes (1 GET and 4 POST)
GET: /swagger/{any:path} (C:/Users/23202/go/pkg/mod/github.com/iris-contrib/swagger/v12@v12.0.1/swagger.go:58)
POST: /book/chapter/content (./main.go:36)
POST: /book/chapter/all (./main.go:96)
POST: /book/get/books (./main.go:129)
POST: /book/search (./main.go:144)

Iris Version: 12.2.8

Now listening on:
> Network: http://192.168.24.1:10086
> Network: http://192.168.148.1:10086
> Network: http://192.168.219.63:10086
> Local: http://localhost:10086
Application started. Press CTRL+C to shut down.

output.log
999 2023/12/06 17:17:18 API:/book/get/books Be Called
1000 2023/12/06 17:17:13 API:/book/get/books Be Called
1001 2023/12/06 17:17:22 API:/book/get/books Be Called
1002 2023/12/06 17:17:29 API:/book/chapter/all Be Called
1003 2023/12/06 17:18:28 API:/book/chapter/all Be Called
1004 2023/12/06 18:10:16 API:/book/get/books Be Called
1005 2023/12/06 18:10:16 API:/book/get/books Be Called
1006 2023/12/06 18:11:07 API:/book/chapter/all Be Called
1007 2023/12/06 18:11:53 API:/book/get/books Be Called
1008 2023/12/06 18:12:41 API:/book/chapter/all Be Called
1009 2023/12/06 18:12:44 API:/book/get/books Be Called
1010 2023/12/06 18:12:47 API:/book/chapter/all Be Called
1011 2023/12/06 18:12:49 API:/book/chapter/all Be Called
1012 2023/12/06 18:13:51 API:/book/chapter/all Be Called
1013 2023/12/06 18:13:53 API:/book/chapter/content Be Called
1014 2023/12/06 19:04:29 API:/book/chapter/all Be Called
1015 2023/12/06 19:04:43 API:/book/chapter/all Be Called
1016 2023/12/06 19:04:45 API:/book/chapter/content Be Called
1017 2023/12/06 19:04:50 API:/book/chapter/content Be Called
1018 2023/12/06 19:10:54 API:/book/chapter/all Be Called
1019 2023/12/06 19:10:57 API:/book/chapter/all Be Called
1020 2023/12/06 19:11:01 API:/book/chapter/content Be Called
1021
```

图 12 后端运行

（二） 前端运行截图

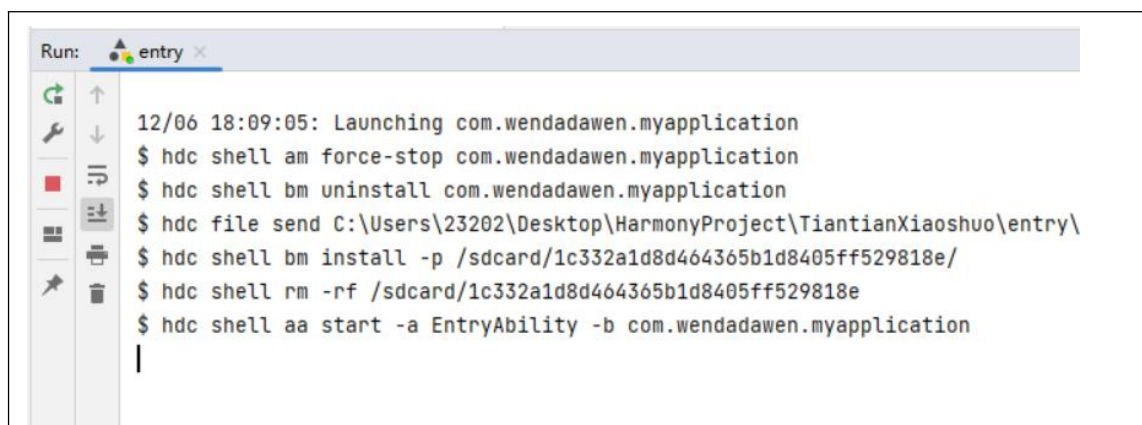


图 13 前端运行

五、总结及分析（完成过程中的心得体会，成功与失败之处，经验，收获，建议、将来计划和展望）

在这次的小小说 APP 的期末项目中，我尝试了不同的技术和工具，学到了很多知识，并且也遇到了一些挑战和问题。通过这次实践，我意识到了一些成功和失败之处，同时也有了一些宝贵的经验和建议

首先，我感到最成功的一点是实现了前后端分离架构，使用了 arkts 和 arkui 进行页面设计，后端采用 golang 的 iris, xorm 等架构。我成功地将自己爬虫程序采集而来的数据存储在 mysql 数据库中，并通过网络请求获取到了数据。通过这个过程，我深刻理解了前后端分离架构的优点，可以更好地管理和维护代码。

不过，在这个项目中，我遇到了一些问题和挑战。首先，由于爬取的数据量过大，时常导致网站崩溃的情况发生。为了解决这个问题，我需要对爬虫程序进行优化，限制爬取速度及其并发量，以免给网站带来太大的压力。

此外，懒加载也是一个需要处理的问题。因为 arkts 的懒加载代码量实在不少，因此我采取了手动实现懒加载的方式，并改进了搜索懒加载的功能，不必一次把所有的结果展示出来造成性能不够。

还有一个比较棘手的问题是，为了进行网络请求，我需要开启系统权限。

而且在请求本机的 API 时，不能使用 localhost，而必须使用 WLAN 的 ipv4 局域网 IP 才行。这可能是因为 APP 的 localhost 无法识别为我的电脑的网络地址，而被识别为手机的地址。通过将地址改为具体电脑的局域网 IP，问题得以解决。

另外，我还遇到了一些其他的技术问题。例如，鸿蒙系统只支持小于 5MB 的传送，导致传送不过去；Console.Log 无法打印很长的字符串，误以为是代码出了问题；无法实现自动化测试，让我感到很困扰，调试代码很不方便。还有 Arkts 中的 as 关键词，虽然可以进行类型强转，但却不能同时赋值类型的函数和变量，这一点坑了我很久。

尽管遇到了一些问题，但在这个项目中，我也有一些成功和收获。我成功地实现了一个前后端分离的小说阅读 APP，用户可以在书城浏览、搜索、阅读小说，并将喜欢的书籍加入书架。意识到了前后端分离架构的优势，能够更好地解耦前后端的开发过程，提高开发效率和灵活性。同时，我也深刻理解了爬虫程序的重要性，以及如何处理大数据量对网站的压力问题。

总的来说，这次期末项目是一个充满挑战和收获的过程。我学到了很多知识和经验，也意识到自己在某些技术方面还需要进一步提升。