

## 《移动应用软件开发》期末软件设计报告

软件名称	日历 app
<p>说明：</p> <ol style="list-style-type: none"><li>1. 每人独立完成移动应用软件设计开发，并完成设计报告。</li><li>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。</li><li>3. 软件开发平台选择 <b>HarmonyOS</b>，优先使用 <b>ArkUI</b> 开发。</li><li>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。</li><li>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。</li><li>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。</li></ol>	
<p><b>报告正文</b></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>1.1 软件设计意图</p> <p>日历 app 是我们每个人都离不开的手机 app，它能帮我实现日程添加、日期计算、农历查看等功能。所以，本次期末软件设计我选择开发实现日历 app，复现现有 app 的基本功能。</p> <p>1.2 功能介绍</p> <ul style="list-style-type: none"><li>● 日期展示功能（公历、农历）：主界面显示当月所有日期的数字以及对应的农历。</li><li>● 日历切换：左滑右滑日历进行日历切换。</li></ul>	

- 日期跳转：点击“三角”图标弹出日期选择器，选择对应年月后进行日期跳转。
- 当日标注：显示时，如果是今天，日期会用不同颜色显示。
- 日期间隔计算：选中日期后界面自动显示当前日期与选中日期的间隔。
- 日程新增：选中相应的日期后，点击“+”图标，弹出对话框新建日程。
- 日程提醒：后台在设置日期提醒用户设置的日程。
- 编辑、删除日程：侧滑显示的日程可以对其进行编辑和删除。
- 模糊搜索日程：点击日期会显示当日所有日程；点击“搜索”图标，通过键入关键字搜索日程，并跳转页面进行显示。
- 黄历查询：点击“详情”图标，跳转页面显示日期详情，包括：干支、生肖、节气等内容。

### 1.3 特色

- 加载动画：界面跳转时，可能会由于数据请求等原因无法及时加载内容，添加加载动画使得页面跳转更自然。
- 黄历显示：目前部分日历 app 没有黄历功能，例如：苹果。
- 日程添加、搜索、删除、编辑功能。
- 日程提醒：添加日程后会设置后台系统提醒。

## 二、开发和运行环境（可能包括开发平台、运行平台、开发工具、第三方组件使用情况、其他支撑软件运行的条件等）

1. **开发平台**：ArkTS 编程语言。
2. **开发工具**：DevEco Studio 64 位。
3. **运行平台**：原则上可在支持鸿蒙操作系统的所有终端上使用。
4. **第三方组件**：公历转农历算法来自 chinese-lunar 库。
5. **支持软件运行的条件**：部分功能需要用户给予网络权限。

三、设计说明（写明设计思想、程序的结构、功能设计、界面设计、模型设计、程序主要执行流程图，最后是核心源代码，截图等）

### 3.1 设计思想

#### 3.1.1 功能设计部分

- 日期展示功能（公历、农历）：主界面显示当月所有日期的数字以及对应的农历。
- 日历切换：左滑右滑日历进行日历切换。
- 日期跳转：点击“三角”图标弹出日期选择器，选择对应年月后进行日期跳转。
- 当日标注：显示时，如果是今天，日期会用不同颜色显示。
- 日期间隔计算：选中日期后界面自动显示当前日期与选中日期的间隔。
- 日程提醒：后台在设置日期提醒用户设置的日程。
- 日程新增：选中相应的日期后，点击“+”图标，弹出对话框新建日程。
- 编辑、删除日程：侧滑显示的日程可以对其进行编辑和删除。
- 模糊搜索日程：点击日期会显示当日所有日程；点击“搜索”图标，通过键入关键字搜索日程，并跳转页面进行显示。
- 黄历查询：点击“详情”图标，跳转页面显示日期详情，包括：干支、生肖、节气等内容。

#### 3.1.2 界面设计部分

- 日历显示主界面：日历显示部分采用格栅布局，添加点击时间；日程显示部分使用 List 以及 ForEach 进行渲染，使得界面美观简洁。
- 搜索结果展示页面：使用 List 对结果数组进行渲染，界面简洁美观。
- 日期详情界面：使用格栅布局、List 容器等显示内容，使界面美观简洁。

### 3.2 程序的结构

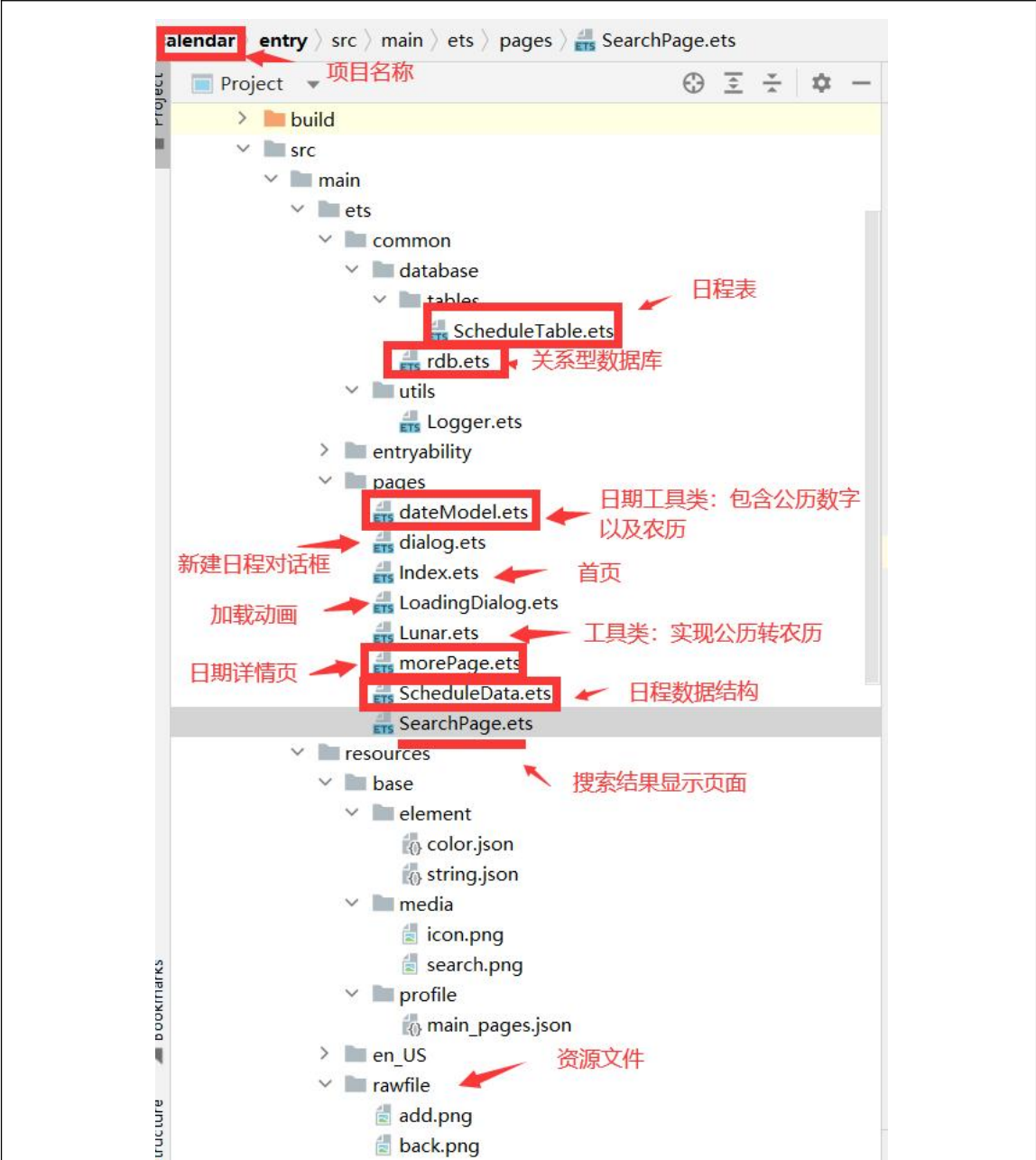


图 1. 程序的结构

3.3 功能设计（功能关系图）

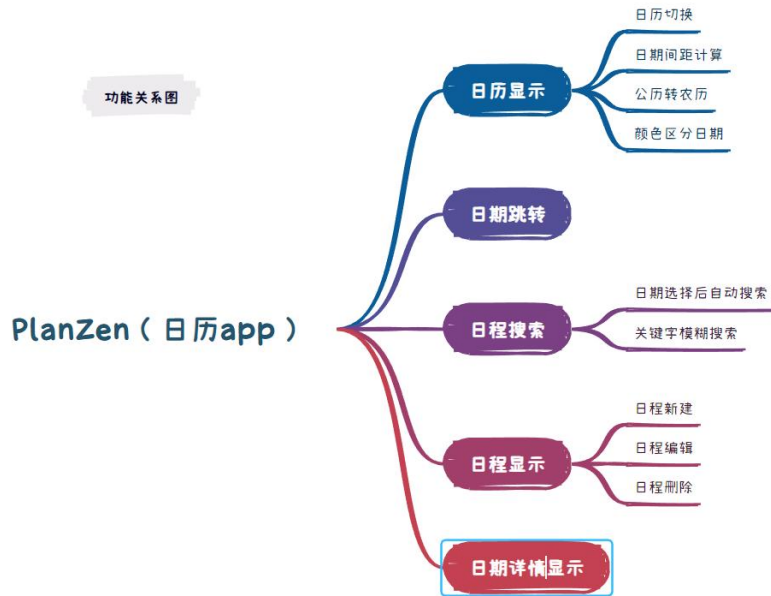


图 2. 功能关系图

### 3.4 界面设计

#### 3.4.1 日历显示页面（首页）、搜索结果页面、黄历页面，如图：



图 3. 首页、搜索结果页面、日期详情页

### 3.5 模型设计：基于华为鸿蒙开发的 Stage 模型。

### 3.6 主要执行流程图

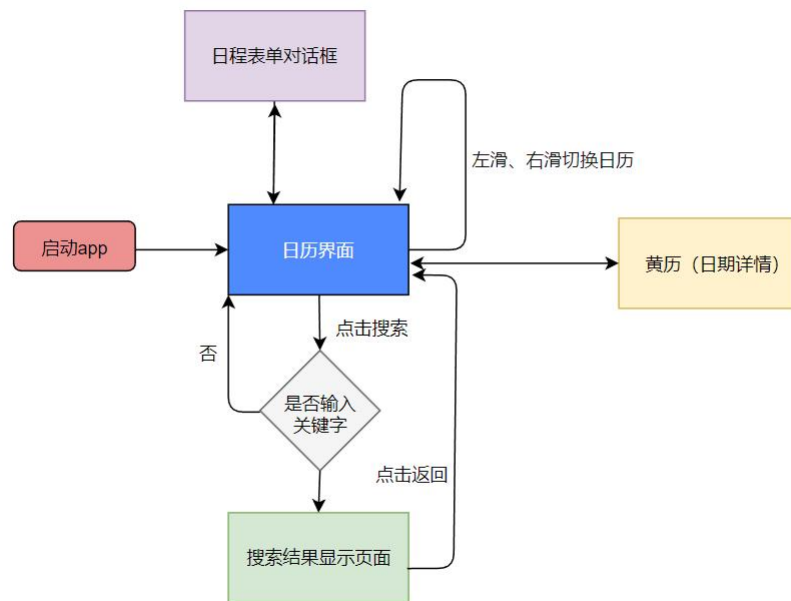


图 4. 程序主要执行流程图

### 3.7 核心源代码

**3.7.1 Lunar.ets:** 该类主要通过公历日期，如：2023-12-05 转换成农历日期，即：十月廿三。除此之外，还有获得传统干支纪年，如癸卯年、生肖，如 2023 年为兔年。下面展示主要源代码：

```

1  solarToLunar(solar:Date):any{
2    let year = solar.getFullYear()
3    let offset = year - 1900; //获得偏移量
4    if(offset < 0 || offset >= MAPPING.length){
5      throw new Error('Specified date range is invalid.')
6    }
7    //查找范围内的农历数据
8    let data = this.findLunar(solar, offset, 0, 13, false)
9    //如果没有找到，则找前一年的，因为农历在公历之前，并且不会超过一年，查一年就可以了
10   data = data || this.findLunar(solar, offset-1, 12, 99, true)
11   //还是没有找到，表示超出范围
12   if(!data) return false;
13   let firstDay = new Date(data.year, data.month-1, data.day)
14   let day = this.solarDiff(solar, firstDay, "d")+1;
15   let month = data.leapMonth > 0 && data.lunarMonth > data.leapMonth ? data.lunarMonth - 1
16   : data.lunarMonth;
17   let leap = data.leapMonth > 0 && data.leapMonth + 1 == data.lunarMonth;
18   let result = {
19     leap: leap,
20     year: data.lunarYear,
21     month: month,
22     day: day,
23     leapMonth: data.leapMonth
24   };
25   return result;
26 }

```

图 5. 公历转农历函数

```

1 //找到农历, isPerYear,是否为农历前一年的对应数据
2 findLunar(solar: Date, index, minMonth, maxMonth, isPerYear): any{
3     let mapping = MAPPING[index] ;
4     if(!mapping) return false;
5     let year = solar.getFullYear();
6     let month = solar.getMonth() + 1;
7     let day = solar.getDate();
8     let lunarYear = year;
9     let lunarMonth, find, solarMonth;
10    let segMonth, segDay;
11    //查找农历
12    for(let i=mapping.length-1; i>0;i--){
13        lunarMonth = i;
14        // @ts-ignore
15        segMonth = Number(mapping[i].substring(0, 2))
16        // @ts-ignore
17        segDay = Number(mapping[i].substring(2, 4))
18        solarMonth = isPerYear && segMonth > 12 ? segMonth-12: segMonth
19        find = solarMonth < month || (solarMonth == month && segDay <= day) ||
20        ((segMonth <= minMonth || segMonth >= maxMonth) && isPerYear);
21        if ((solarMonth == 12 && solarMonth > month && i == 1)) {
22            find = true;
23            year--;
24        };
25        if (find) break;
26    }
27    if(!find) return false; //如果找到, 则赋值
28    if (isPerYear && segMonth == 12) year = year - 1; //取前一年
29    lunarYear = isPerYear ? lunarYear - 1 : lunarYear;
30    return {
31        year: year,
32        month: solarMonth,
33        day: segDay,
34        lunarYear: lunarYear,
35        lunarMonth: lunarMonth,
36        leapMonth: mapping[0] //闰月
37    };
38 }

```

图 6. 辅助函数, 找到农历对应位置

**3.7.2 dateModel.lets:** 该类主要是定义了 date 数据结构, 包含 isSelected (是否被选中)、isToday (是否是今天)、date (Date 类型, 日期)、weekDay (一周的第几天)、lunarMonth (农历月)、lunarDay (农历日) 成员属性等, 为了加快页面显示速度, 这里只设置了少量的成员属性, 其他信息等到需要时在进行计算。源代码展示:

```

1  export class date {
2      isSelected: boolean = false;
3      isToday: boolean = false;
4      date: Date | null ;
5      weekDay: number = 0; //星期几?
6      // yearTips: string = ""; // 干支描述
7      // chineseZodiac: string = ""; //生肖
8      // lunaryear: string = "";
9      lunarmonth:string = "";
10     lunarday:string = "";
11     result: any ;
12     // solarTerms: ""; //节气描述
13     // avoid: ""; // 不宜事项
14     // lunarCalendar: string = ""; //农历日期
15     // suit: ""; //宜事项
16     // dayOfYear: number = 0; //一年的第几天
17     // weekOfYear: number = 0; //一年的第几周
18     // constellation: string = ""; //星座
19     // indexWorkDayOfMonth: number = 0; //当月的第几个工作日
20     lunar = new Lunar()
21     constructor(isToday: boolean, date: Date | null) {
22         this.isToday = isToday;
23         this.date = date;
24         if(this.date !== null){
25             this.weekDay = this.date.getDay()
26             this.result = this.lunar.solarToLunar(date)
27             // this.chineseZodiac = result.shengxiao;
28             // this.yearTips = result.Ganzhi;
29             // this.lunaryear = result.lunaryear;
30             this.lunarmonth = this.lunar.lunarFormat(this.result, "M");
31             this.lunarday = this.lunar.lunarFormat(this.result, "D");
32         }
33     }
34 }

```


图 7. date 数据结构

**3.7.3 ScheduleData.ets:** 该类主要是对日程 schedule 数据结构进行定义，包括：id、title、date、year、day、note（备注）成员属性。

### 3.7.4 LoadingDialog.ets & dialog.ets

- **LoadingDialog:** 由于日期详情页是使用 request 进行数据请求，页面显示不及时，添加加载动画使页面衔接更自然。





```

1  @CustomDialog
2  export struct LoadingDialog{
3    controller: CustomDialogController
4    build(){
5      Row(){
6        LoadingProgress().width(40).height(40).color(0x317aff)
7        Text("Loading...").fontSize(20).fontColor(Color.Gray)
8      }.width(100).height(100)
9    }
10 }


```

图 8. 加载动效对话框

- **dialog:** 该类是用于用户新建日程时弹出的表单对话框，对话框显示表单比页面直接显示更自然。这里不作代码展示。

### 3.7.5 index.ets: 日历首页。（由于篇幅限制，这里介绍部分功能。）

- **使用手势实现 tab 切换:** 由于 tab 自带的切换模式不支持循环滑动，使用手势结合其 `changeIndex` 接口实现循环 tab 切换。



```

1  .gesture(
2    PanGesture()
3    .onActionStart(() => {
4      console.log("拖动手势开始!! ")
5    })
6    .onActionUpdate((event: GestureEvent) => {
7      this.offsetx = event.offsetX;
8      console.log("offset: " + event.offsetX)
9    })
10   .onActionEnd(() => {
11     if (this.offsetx > 150) {
12       //月份减小
13       if (this.month === 1) {
14         this.year = this.year - 1;
15       }
16       this.month = this.months[(this.month + 10)%12];
17       this.tabController.changeIndex(this.month - 1);
18       this.selectedindex = Math.floor(Math.random()*20) + 7;
19       //this.getDaysOfMonth(this.year, this.month)
20       this.aboutToAppear()
21       this.offsetx = 0;
22     } else if (this.offsetx < -150) {
23       //月份增大
24       if (this.month === 12) {
25         this.year = this.year + 1;
26       }
27       this.month = this.months[(this.month)%12];
28       this.tabController.changeIndex(this.month - 1);
29       this.selectedindex = Math.floor(Math.random()*20) + 7;
30       this.aboutToAppear()
31       this.offsetx = 0;
32     }
33   })
34 )

```

图 9. 日历切换原理代码

- **日期间隔计算：**通过 Date 的 getTime 接口实现计算，最后通过双目运算符实现几天前、几天后、今天、明天、昨天显示。

```

1 //日期间隔计算
2 ComputeDays(date1:Date, date2:Date): number{
3   let diff = date1.getTime() - date2.getTime();
4   return Math.floor(diff / (1000 * 60 * 60 * 24)) +1;
5 }
6 //双目运算符实现显示
7 Text(this.ComputeDays(this.selectedDate, new Date()) === 0? "今天" :
8   (this.ComputeDays(this.selectedDate, new Date()) < 0?
9   (this.ComputeDays(this.selectedDate, new Date()) === -1? "昨天":
10    (-1*this.ComputeDays(this.selectedDate, new Date()).toString() + "天前")
11    : (this.ComputeDays(this.selectedDate, new Date()) === 1? "明天":
12     (this.ComputeDays(this.selectedDate, new Date()).toString() + "天后"))) )
13   .fontWeight(FontWeight.Bold)

```

图 10. 日期间隔计算

- **添加日程提醒：**通过添加后台代理提醒，在特定时刻通知用户。

```

1 let datetime: reminderAgentManager.LocalDateTime = {
2   year: newSchedule.year,
3   month: newSchedule.month,
4   day: (new Date(newSchedule.date)).getDate(),
5   hour: 9,
6   minute: 0,
7 }
8 let remindercalendar: reminderAgentManager.ReminderRequestCalendar = {
9   dateTime: datetime,
10  reminderType: reminderAgentManager.ReminderType.REMINDER_TYPE_CALENDAR,
11  actionButton:[
12    {
13      title: '延迟通知',
14      type: reminderAgentManager.ActionButtonType.ACTION_BUTTON_TYPE_SNOOZE
15    }
16  ],
17  wantAgent: {
18    pkgName: "com.example.calendar",
19    abilityName:"EntryAbility"
20  },
21  timeInterval: 3600,
22  title: '日程提醒',
23  content: newSchedule.title + '(点击查看详情)',
24 }
25 try{
26   reminderAgentManager.publishReminder(remindercalendar, (err, reminderId) =>{
27     //console.log("添加提醒成功, 日程: ", newSchedule.title + (newSchedule.note === '' ? '' : '(, 详情: ' + newSchedule.note)))
28   })
29 }catch (error){
30   //console.log("提醒添加失败, 错误信息为: ", error.code + error.message)
31 }

```

图 11. 日程提醒

- **日程表、日程数据插入、删除、更新不作展示，原理同实验三。**

### 3.7.6 SearchPage.ets & morePage.ets

- **SearchPage.ets:** 该界面实现搜索结果显示。
- **morePage.ets:** 该界面实现详情展示，通过查询到的获取黄历 api，根据页面跳转时传过来的日期，使用 request 进行数据请

求，并在请求失败后返回原页面。

```

1  aboutToAppear() {
2    let httpRequest = http.createHttp();
3    let result = router.getParams()['date'].replace(/-/g, '');
4    let url = 'https://www.mxnzp.com/api/holiday/single/' + result +
5    '?ignoreHoliday=false&app_id=xljlk9pds1pnjq19&app_secret=JDSTVBDRBwBN09LXa5622S3dSxL1
6    1z2Z';
7    httpRequest.request(url, (err, res) => {
8      if (!err) {
9        // @ts-ignore
10       let tmp = JSON.parse(res.result).data;
11       this.days = tmp;
12       // @ts-ignore
13       console.log("请求的数据为: " + JSON.stringify(JSON.parse(res.result).data))
14       //console.log("请求的数据为: " + JSON.stringify(this.days))
15       // @ts-ignore
16       this.suit = tmp.suit.split('.')
17       console.log("请求数据区成功, suit数组长度" + this.suit.length.toString())
18       // @ts-ignore
19       this.avoid = tmp.avoid.split('.')
20       // @ts-ignore
21       let tmp_str = "农历" + tmp.lunarCalendar
22       this.words = []
23       for (let i = 0; i < tmp_str.length; i++) {
24         this.words.push(tmp_str[i])
25       }
26       this.weekDay = tmp.weekDay;
27       this.typeDes = tmp.typeDes;
28       this.yearTips = tmp.yearTips;
29       this.chineseZodiac = tmp.chineseZodiac;
30       this.solarTerms = tmp.solarTerms;
31       this.dayOfYear = tmp.dayOfYear;
32       this.weekOfYear = tmp.weekOfYear;
33       this.constellation = tmp.constellation;
34       this.LoadingdialogController.close()
35       // @ts-ignore
36       console.log("请求数据成功, 详细日期: " + tmp.date)
37     } else {
38       console.log("请求数据失败!")
39       this.LoadingdialogController.close()
40       prompt.showToast({ message: '加载失败, 请检查网络后重试!', bottom: 70 });
41       setTimeout(function () {
42         router.back()
43       }, 500);
44     }
45   }
46 }

```

图 12. request 数据请求

#### 四、源程序调试过程（运行、调试截图和文字）

##### 1. 日程新增、显示、编辑。

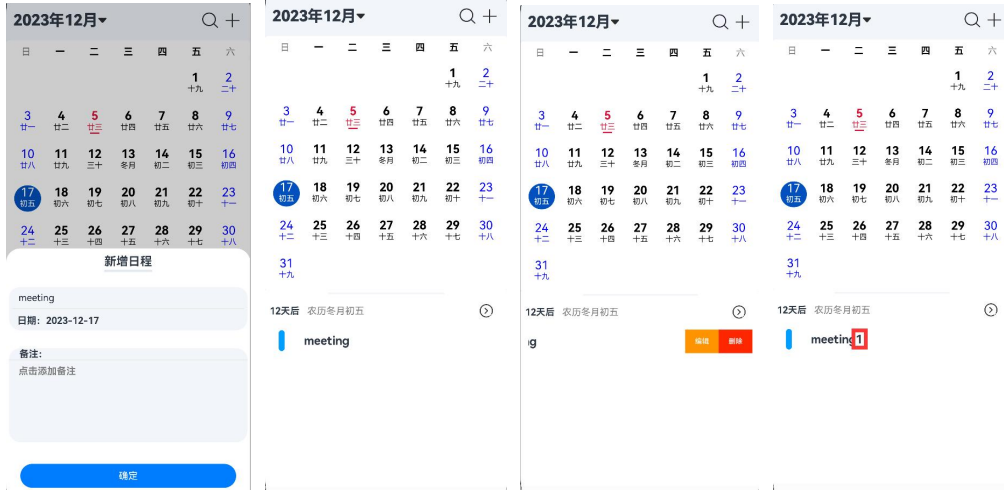


图 13. 新增日程、日程显示、日程编辑

## 2. 日程模糊搜索、日程删除。

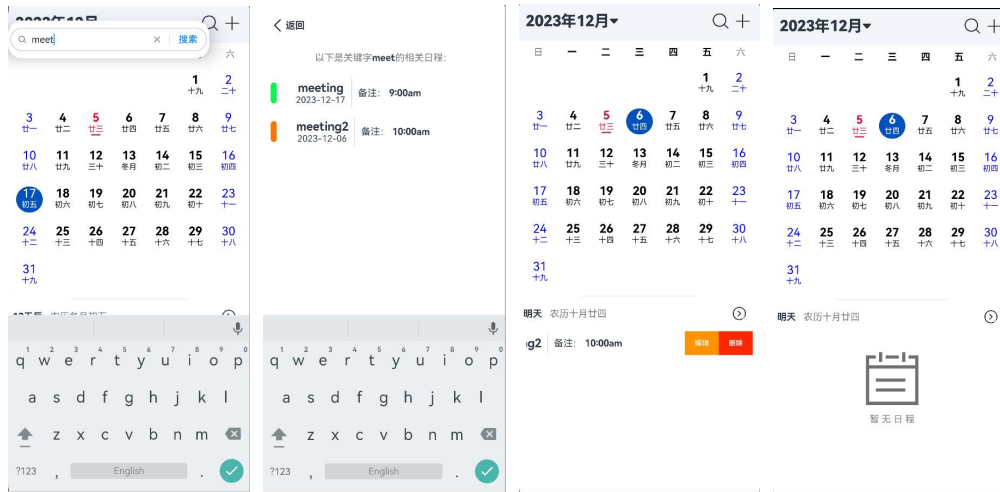


图 14.日程模糊搜索、日程删除

## 3. 日期跳转、日历侧滑切换。



图 15. 日期跳转、日历侧滑切换

4. 日期详情页、加载动画、网络请求失败提示并自动返回。



图 16. 日期详情页、加载动画

5. 日程提醒、点击提醒打开页面。

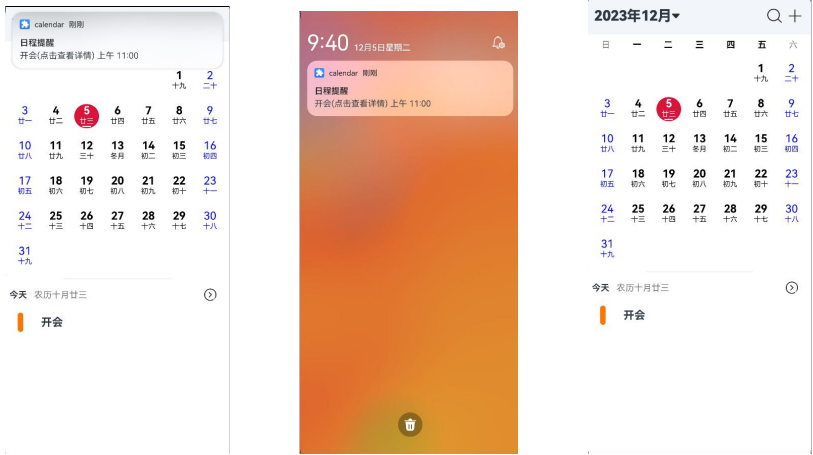


图 17. 日程提醒效果