### **1. 数据结构**

#### **事件（Event）对象**

每个事件对象包含以下字段：

* id: 唯一标识符（数字）
* title: 事件标题（字符串）
* location: 事件地点（字符串）
* startTime: 开始时间（字符串，格式为"HH:MM"）
* endTime: 结束时间（字符串，格式为"HH:MM"）
* date: 日期（数字，表示该月的第几天，例如9表示6月9日）
* description: 事件描述（字符串，可选）
* type: 事件类型（字符串，如'默认'、'重要日'等，在编辑表单中出现）
* allDay: 是否全天事件（布尔值，在编辑表单中出现）

#### **事件存储结构**

事件以嵌套对象的形式存储：

{

[date]: {

[startTime]: eventObject,

[startTime]: eventObject,

...

},

...

}

例如：

{

9: {

'09:00': { id: 1, title: '采购会议', ... },

'14:00': { id: 2, title: '客户访谈', ... }

},

10: {

'10:00': { id: 3, title: '项目评审', ... }

},

...

}

#### **日期元数据（用于渲染顶部星期栏）**

const weekDays = [

{ id: 1, day: '一', date: 9, lunar: '十四' },

{ id: 2, day: '二', date: 10, lunar: '十五' },

...

];

#### **时间槽（用于渲染时间网格）**

const timeSlots = [

{ id: 1, time: '08:00' },

{ id: 2, time: '09:00' },

...

];

#### **全天事件**

const allDayEvents = [

{ id: 1, title: '父亲节' }

];

### **2. 组件接口**

#### **WeekCalendar组件**

#### ****props****:

onAddEvent: 当点击添加按钮时调用的函数

onEditEvent: 当在事件详情中点击编辑按钮时调用的函数，参数为当前事件对象

#### **EventFormScreen组件**

#### ****props****:

mode: 模式，'create'（新建）或'edit'（编辑）

eventData: 编辑模式下传入的事件对象（可选）

onClose: 关闭表单的回调

onSave: 保存事件的回调，参数为事件对象和模式（'create'或'edit'）

#### **App主组件**

管理两个视图的切换：

currentView: 当前视图，'calendar'（周视图）或'form'（表单视图）

formMode: 表单模式，'create'或'edit'

selectedEvent: 当前选中的事件（用于编辑）

### **3. 事件处理流程**

#### **添加事件**

在App组件中，点击添加按钮设置formMode='create'，切换视图到表单。

在EventFormScreen中填写信息，点击保存。

调用onSave（即App中的handleSaveEvent），这里可以处理事件保存逻辑（如更新状态或发送到后端）。

关闭表单，回到周视图。

#### **编辑事件**

1.在周视图中点击一个事件，弹出详情弹窗。

2.点击编辑按钮，调用onEditEvent（即App中的handleEditEvent），设置formMode='edit'并传入当前事件，切换到表单视图。

3.在表单中修改事件，点击保存，同样调用onSave。

4.关闭表单，回到周视图。

#### **删除事件**

在周视图的事件详情弹窗中点击删除按钮，执行删除操作（删除本地状态中的事件）。

1. 效果展示

周界面



可以上下滚动查看日程

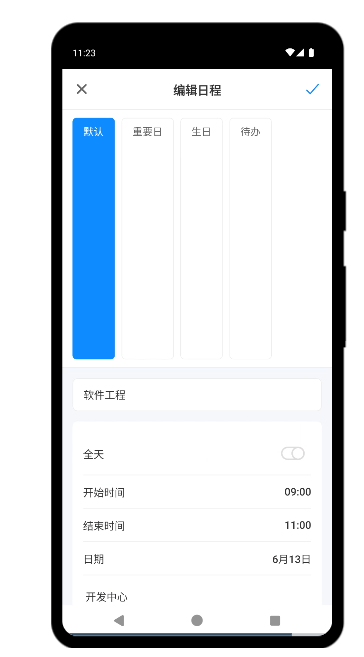
点击日程块可以查看详细信息



点击删除按钮可以删除对应日程



点击编辑可以跳转到编辑界面



点击右下角加号可以跳转到新建日程界面

