① 产品功能介绍

产品概述

这是一个基于UniApp开发的跨平台日历应用程序，支持Android、iOS以及Web等多个平台。该应用实现了完整的日程管理功能，用户可以通过直观的界面进行日程的增删改查，并支持多种视图模式和智能提醒功能。

核心功能

1. 日历视图展示

- **月视图**：以月为单位展示日程安排，清晰显示整个月份的日程分布情况
- **周视图**：以周为单位展示日程，详细呈现一周内的每日安排
- **日视图**：以天为单位展示日程，按照时间顺序排列当天的所有事件

2. 日程管理功能

- **创建日程**：用户可以添加新的日程事件，包括标题、描述、时间、地点等详细信息
- **编辑日程**：支持修改已有日程的所有信息
- **查看日程**：点击任意日程可查看详情信息
- **删除日程**：支持删除不再需要的日程事件

3. 日程提醒功能

- **自定义提醒时间**：用户可以为每个日程设置提醒，支持设置提前分钟、小时或天数提醒
- **智能提醒推送**：系统会在日程开始前按照设定的时间自动推送提醒通知
- **多平台通知**：在不同平台上均能收到及时的提醒通知

② 程序概要设计

整体架构

系统采用前后端分离架构，前端使用UniApp框架开发，后端使用Node.js + Express构建RESTful API，数据存储采用MongoDB数据库。

后端设计

- **技术栈**：Node.js + Express + MongoDB
- 主要模块：
    1. 数据库连接模块：负责连接MongoDB数据库
    2. 路由模块：定义RESTful API接口路由
    3. 控制器模块：处理业务逻辑
    4. 模型模块：定义数据模型结构

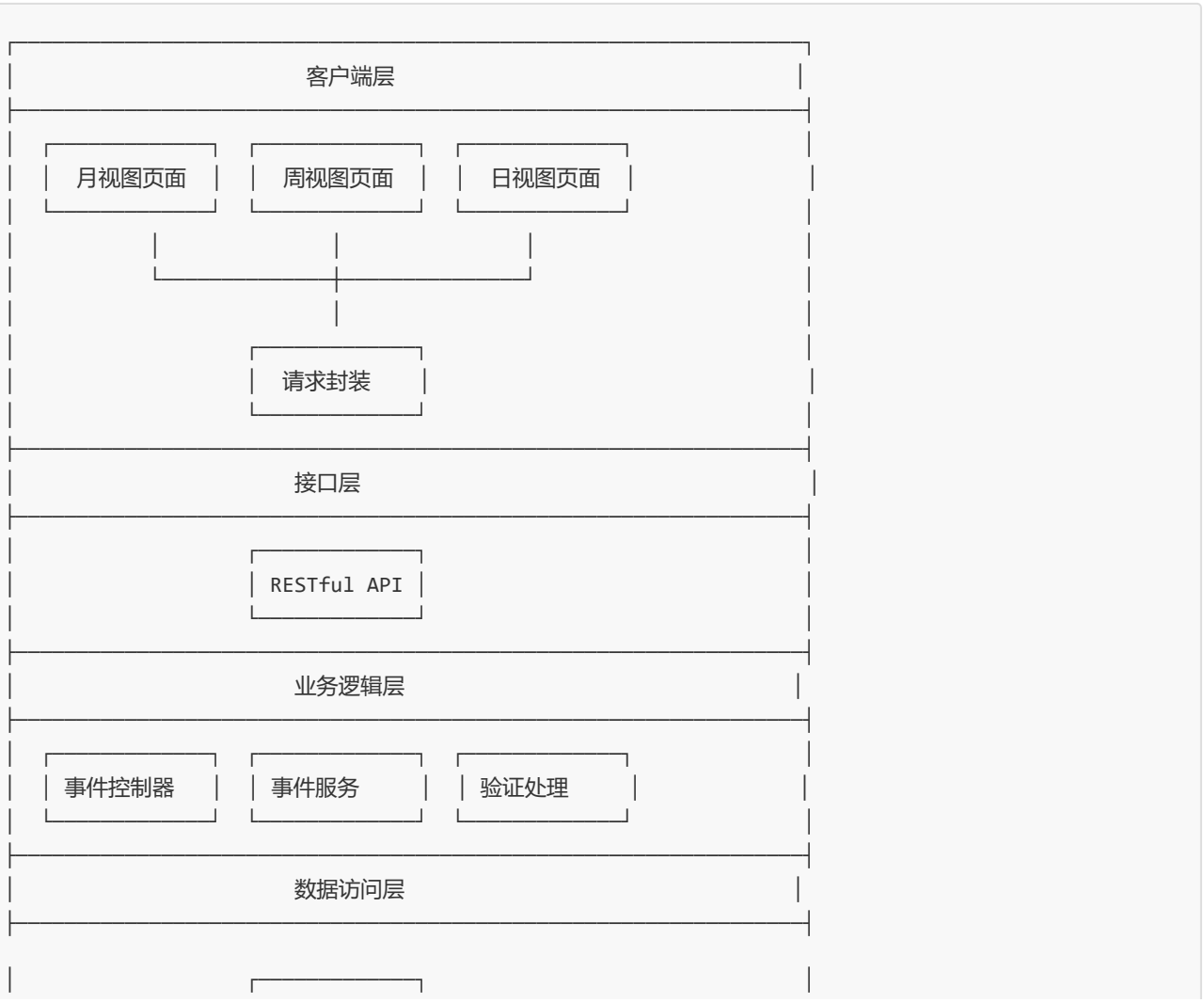
前端设计

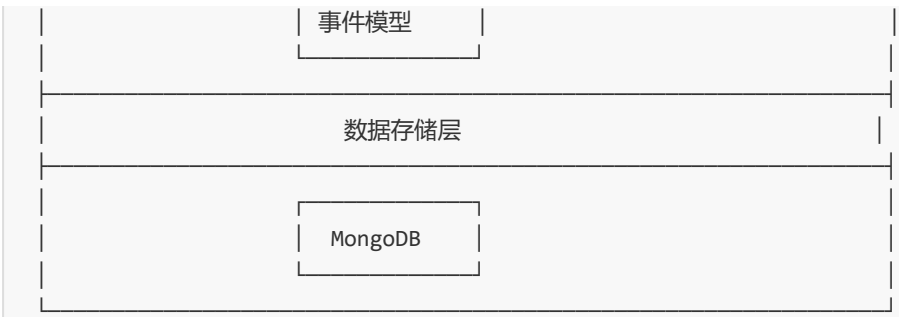
- **技术栈**：UniApp + Vue2
- 主要页面：
    1. 月视图页面：展示月度日程安排
    2. 周视图页面：展示周度日程安排
    3. 日视图页面：展示每日详细日程
- 核心组件：
    1. 月视图组件(MonthView)
    2. 周视图组件(WeekView)
    3. 日视图组件(DayView)

数据流设计

1. 用户在前端界面操作（增删改查日程）
2. 前端通过HTTP请求调用后端API
3. 后端接收请求，处理业务逻辑并与数据库交互
4. 数据库存储或检索相关数据
5. 后端返回处理结果给前端
6. 前端更新界面展示最新数据

③ 软件架构图





④ 技术亮点及其实现原理

1. 跨平台兼容性

技术亮点：使用UniApp框架实现一套代码多端运行，同时支持Android、iOS和Web平台。

实现原理：

- UniApp编译器将Vue代码转换为各平台原生代码
- 通过条件编译处理平台特定逻辑
- 统一的API封装屏蔽平台差异

2. 响应式日程提醒

技术亮点：实现前端定时提醒功能，能够在事件即将开始时主动通知用户。

实现原理：

- 利用JavaScript的setTimeout函数实现定时任务
- 在组件加载和数据更新时重新计算提醒时间
- 通过uni.showToast在不同平台推送提醒通知
- 提供分钟、小时、天三种时间单位供用户选择

3. 多维度日程视图

技术亮点：提供月、周、日三种不同维度的视图展示，满足用户在不同场景下的查看需求。

实现原理：

- 月视图：计算当月所有日期并展示对应事件
- 周视图：计算当前周的所有日期并展示事件
- 日视图：展示指定日期的所有事件并按时序排序
- 三个视图共享同一套数据源和服务接口

4. RESTful API 设计

技术亮点：遵循RESTful规范设计API，结构清晰，易于维护和扩展。

实现原理：

- 使用Express框架构建RESTful API
- 采用标准HTTP方法（GET、POST、PUT、DELETE）对应CRUD操作

- 统一JSON格式的数据交换
- 规范的状态码和错误处理机制

5. 前后端分离架构

技术亮点：完全的前后端分离设计，提高系统的可维护性和扩展性。

实现原理：

- 前端专注于UI展示和用户交互
- 后端专注于数据处理和业务逻辑
- 通过HTTP协议进行数据传输
- 松耦合设计便于团队分工协作

6. Vue 2 组件化架构

技术亮点：采用 Vue 2 经典选项式 API（Options API），通过组件化开发实现代码复用和关注点分离。

实现原理：

- 使用经典的选项式 API（data、methods、computed、watch 等）
- 通过组件封装实现月视图、周视图、日视图等可复用组件
- 利用 Vue 的响应式系统实现数据绑定和状态管理
- 通过 props 和 events 实现父子组件通信
- 使用生命周期钩子处理组件初始化和销毁逻辑