

RunScheme 项目报告

活力智启 健康随行

Vitality and wisdom are enlightened
health accompanies you

目录

1 项目名称	3
2 摘要	3
3 软件分类	3
4 应用领域	3
5 开源协议	3
6 作品概述	3
6.1 背景	4
6.2 创新交互	5
6.3 用户定位	6
6.4 设计思路	7
7 体系结构	8
8 功能描述	9
8.1 首页	9
8.2 课程详细页面	9
8.3 饮食推荐	10
8.4 天气	11
8.5 训练计划	12
8.6 训练详细内容	12
8.7 具体训练科目	13
8.8 运动历史	14
8.9 新闻资讯	15
9 关键技术	16
9.1 获取时间	16
9.2 天气 API 的接收与请求	18
9.3 根据时间跳转	20
9.4 数据库	23
9.5 新闻 API 的接收与请求	34
10 过程中遇到的问题	35
10.1 天气获取	35
10.2 数据库存储	36
10.3 真机与模拟器	36
11 组织架构及职责	37
12 比赛收获	38
13 总结	39

1 项目名称

RunScheme

2 摘要

党的十八大以来，我国加快推进科技自立自强。满足人民群众日益增长的健康需求，全面推进健康中国建设，需要进一步加强科技与健康的融合发展。为了摆脱以国外操作系统为支撑的运动健康软件长期占领健康需求市场的局面，Vivo 推出了 Rust 语言编写的蓝河操作系统，为我们自主开发运动健康软件提供必要条件。RunScheme 手表应用软件为跑步爱好者提供了更好的交互体验，不仅可以提供实时的信息咨询，也能为用户量身定制训练方案，还可以显示适配的饮食与穿搭建议，实现“活动有方，五脏自和”的理想生活状态。

3 软件分类

健康生活类

4 应用领域

运动与健康

5 开源协议

"license": "Apache-2.0"

6 作品概述

6.1 背景

目前，国外操作系统为支撑的运动健康软件长期占领健康需求市场，国人运动的行为数据、运动轨迹、国民的身体基本状况等基础数据存在着一定的安全隐患。为落实整体国家安全观，Vivo 推出了 Rust 语言编写的蓝河操作系统，为我们自主开发运动健康软件提供必要条件。同时，《健康中国行动（2019—2030 年）》明确，到 2022 年，建立覆盖经济社会各领域的健康促进政策体系，全民健康素养水平稳步提高，重大慢性病发病率上升趋势得到遏制。到 2030 年，全民健康素养水平大幅提升，健康生活方式基本普及，居民主要健康影响因素得到有效控制，人均健康预期寿命显著提高。

截至 2022 年底，我国运动健身 APP 行业市场规模达到 130.67 亿元，同比增长 20.3%，运动健身 APP 行业在近年来保持了快速增长的态势。在运动健身 APP 行业，Keep、咕咚、乐动力、小米运动是主要的参与者。其中，Keep 以 32.5% 的市场份额位居行业第一，咕咚以 22.8% 的市场份额位居第二。线上健身市场积累了大量活跃用户，人们对健康生活方式不断追求，预计未来我国运动健身 APP 行业市场规模将继续保持快速增长态势。

为了落实好以人民健康为核心的任务，加快推进自主科技与健康的安全和发展，Vivo 作为一家 28 年专注通信行业的科技公司，创新性地使用 Rust 语言编写了面向通用人工智能时代自研的下一代智慧操作系统，蓝河操作系统。通过不断地实践和科技创新，进行 Vivo 生态的建设，打造了一系列具有行业里程碑意义的产品，比如 Hi-Fi 手机，屏幕指纹手机，以及位列全球多个权威榜单的蓝心 AI 大模型等。还为不断推进 Vivo 生态的科技与健康领域融合发展，Vivo 已开发的手表端应用软件，主要功能包括运动指导、运动数据记录、科学的健身计划。在实际使用中，这些应用反映出在运动一体化的相关运用中存在不能提供实时资讯，用户对话还有提升空间，没有对应的饮食建议和穿搭建议，用户需要打开其它应用软件才能获取上述信息。这为我们进一步开发一体化的信息咨询、饮食与穿搭建议等功能提供了必然。

6.2 创新交互

当今运动健身 APP 行业蓬勃发展，市场上运动健身软件数量丰富、种类繁多，以下是几款主流的运动健身软件的主要功能分析：

软件 功能	Keep	咕咚	乐动力	小米运动	RunScheme
多样化健身计划	✓	✓	✓	✓	✓
运动数据记录	✓	✓	✓	✓	✓
运动动作指导	✓	✓	-	✓	✓
训练计划定制	✓ (会员)	✓	-	✓	✓
运动饮食规划	✓ (较少)	-	-	-	✓
穿搭建议(天气变化)	-	-	-	-	✓
新闻	-	-	-	✓	✓

由上表可知，主流的运动健身软件如 Keep、咕咚、乐动力和小米运动，虽然都提供了丰富的功能和个性化的服务，但在实际使用中仍有一些不足，特别是在饮食规划和天气变化对穿搭的影响方面。

RunScheme 作为一款智能手表应用软件，在**继承主流软件优点**的同时，更加注重用户的**实际需求和体验**。

俗话说想要提高运动成绩“三分靠练，七分靠吃”，RunScheme 加入了基于用户自身身体情况，并且与训练计划相匹配的**饮食推荐功能**。该功能使用户在使用软件训练的过程中，运动恢复及运动补给方面更加便利，指导用户如何准备饮食，帮助用户更全面地提升身体素质、运动能力，提高运动成绩。

以及根据温度、天气、湿度等天气情况进行穿搭推荐的功能，使用户在使用软件训练的过程中，在运动准备和运动恢复方面更加便利、高效，帮助用户更全面地提升运动能力。

其次，RunScheme 还考虑到**天气变化对运动穿搭**的影响，根据温度、天气、

湿度等天气情况为用户提供基于天气情况的**穿搭推荐**，使用户能够轻松应对多变的天气，帮用户解决退出应用去查找天气信息并为穿搭发愁的烦恼，大大增加用户使用的便利性和舒适度。

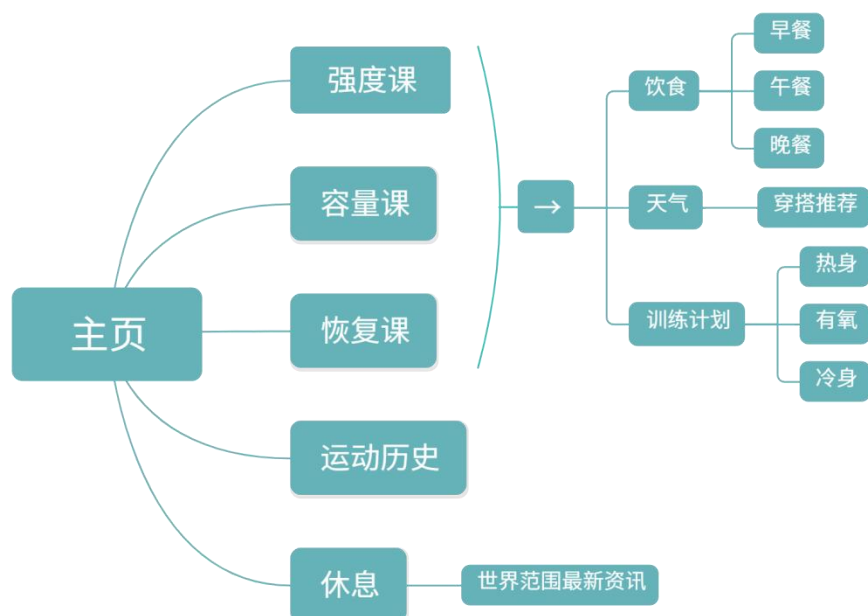
同时，RunScheme 还能给用户随时提供**世界范围内的最新资讯**，这些资讯内容覆盖**体育赛事信息、健康养生知识、科学运动方法、前沿科技动态、以及时尚生活指南**等多个方面，方便用户运动之余也能了解到各个方面的最新信息。

综上所述，RunScheme 吸收市场上主流软件优点以及改进其不足进行了功能的优化设计，大大增加了用户使用的便利性和舒适度，让运动、健康、生活变得更加简单高效。

6.3 用户定位

在设计 RunScheme 应用时，针对 vivo watch 的用户人群进行了精准定位。主要包括健康意识增强，愿意投入时间精力锻炼的人群；工作压力大、时间紧张的城市白领；面临学业压力但热爱运动的高校学生；有一定跑步基础、追求更高成绩和体验的业余跑步爱好者；喜欢尝试新科技和智能设备的爱好者；以及注重个性化体验，期望获得个性化运动建议和训练计划的人群。RunScheme 旨在为这些用户提供科学、智能、个性化的跑步训练体验，助其在忙碌生活中提升身体素质，享受健康生活。

6.4 设计思路



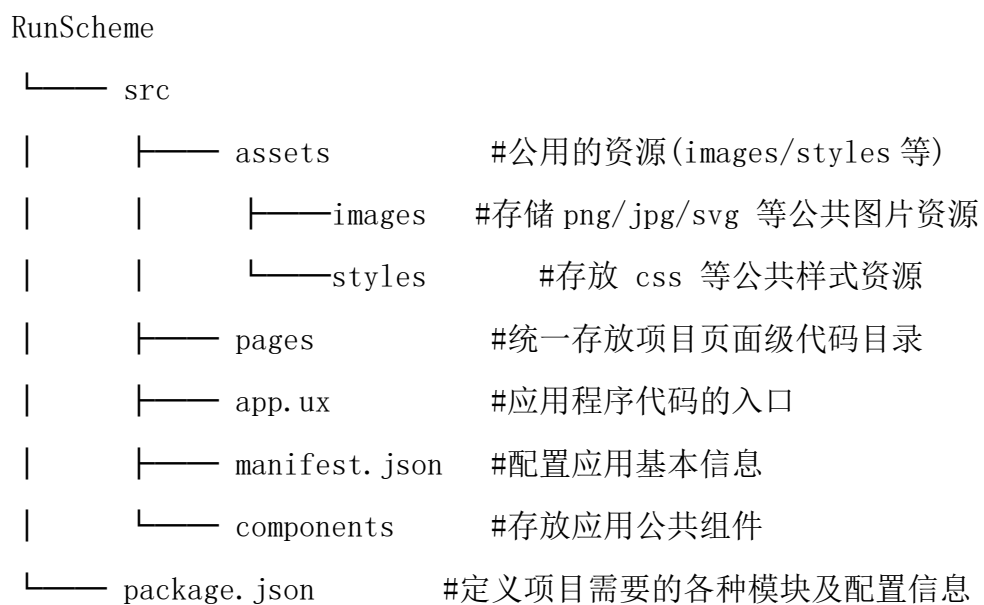
RunScheme 专注于为业余跑者提供一个简洁、便利的户外跑步训练导航平台。首先，进入软件用户会看到一个主页，主页列出了可选的课程类别，如“强度课”、“容量课”等。用户选择其中某一个课程，便可进入该课程的详细页面。

在课程的详细页面上，用户可以查看天气及穿搭推荐、饮食建议和训练计划。天气信息对于户外跑步来说至关重要，用户可以通过软件查看户外天气情况，并且会有适合对应天气的时尚运动穿搭推荐，用户可以根据推荐结合自身情况进行合理的穿衣搭配，防止因为穿着导致人体中暑或者失温。饮食建议详细说明了在训练前后应该摄入的食物种类和营养配比，用户可根据饮食建议获取足够的能量和营养，使用户更好的提升自己的运动能力，促进身体恢复。训练计划根据用户的自身跑步能力进行了科学地训练安排，点击便可进入训练计划的详细页面。

在训练计划的详细页面中，对具体的训练科目进行了分类，可以看到具体训练科目的训练项目大类，如强度课中的热身、有氧、冷身。用户进入任意一个训练项目大类，可以看到具体的训练科目，点击可进入训练界面，用户便可以开始训练。具体的训练科目界面有详细的动作指导，用户可以方便、快速的掌握动作要领，进行高质量的训练。

RunScheme 希望通过上述功能的设计实现使业余跑者能够得到高效便捷的
运动训练服务，帮助业余跑者全面提升跑步能力，提高跑步成绩，达到预期的训
练目标。

7 体系结构



8 功能描述

8.1 首页

在首页的表盘上，顶部通过调用函数获取系统时间并且格式化，从而显示当天的日期和时间。页面主体展示一系列课程选项，包括“强度课”、“容量课”、“恢复课”和“休息课”等。用户通过点击便可进入该课程的详细页面，从而为用户提供更深入的课程信息。



8.2 课程详细页面

进入某一课程详细页面后，如果此时恰好是计划的用餐时间，则会直接跳转到饮食推荐页面，该功能旨在提醒用户及时补充营养、摄入食物。



否则，进入课程详细页面，以强度课为例，页面顶部显示实时日期，左上角的饮食标志点击可进入专业饮食推荐，右上角的天气标志点击可展示最新天气信息及穿搭推荐。页面正中央有励志名言，当前参加赛事的倒计时。页面底部是训练计划入口，点击可进入软件提供的定制训练方案。



8.3 饮食推荐

以早餐为例，进入饮食推荐页面后，页面顶部有实时日期，下方有软件给出的建议用餐时间，确保用户能够科学的用餐补给。页面的主体部分则详细列出了针对用户身体情况及训练计划的具体饮食建议，旨在为用户提供均衡、营养的饮食方案。在页面的左侧和右侧，设有快捷切换按钮，用户可以通过点击查看不同时间段的饮食计划。在页面的最底部，设有一个返回首页的按钮，用户点击即可返回到课程详情页面。



8.4 天气

在天气页面中，我们通过调用天气 API 接口在页面上显示当前所在地区、天气状况、温度、湿度以及风向等，为用户提供一个针对日常生活的全面天气信息。在页面的下方，通过对天气数据的深入分析，为用户提供了定制的穿搭推荐帮助用户选择适合的装备，以应对多变的天气情况。



8.5 训练计划

进入训练计划页面后，多样化的训练项目大类将逐一呈现。这些训练项目大类包括热身、有氧训练、无氧运动、力量训练以及冷身等多个关键阶段。每个阶段都有了详细具体的训练科目，确保用户在合适的时长内高效而系统的训练。



8.6 训练详细内容

以热身为例，页面主体部分详细列出了当前的具体训练内容，用户可以点击进入，开始训练。在页面的右侧，设置了一个切换按钮，点击后用户可以切换训练项目大类，开启下一阶段的训练。页面的左上角，设有一个返回按钮，用户点击可返回到训练项目大类的页面。



8.7 具体训练科目

(1) 以弓箭步为例，动作详情页面的顶端显示着动作名称；页面中部会向用户展示动作训练要求，如持续时长；文字部分详细地阐述动作的关键要领；页面底部设有两个按钮，点击“开始”，开始训练，点击“暂停”，暂停训练，当课程倒计时 30s 结束之后，自动结束训练，训练记录计入历史运动数据，方便用户日后查看，了解自己过去的训练情况。



(2) 以有氧长跑为例，在动作详情页面的顶部显示动作名称；紧接着，页面中部有一个计时器，用于记录跑步时长；页面底部设有两个按钮，点击“开始”，开始跑步，点击“结束”，结束跑步，将跑步数据计入历史运动数据中，方便用户日后查看，了解自己过去的训练情况。



(3) 以慢速步行为例，在动作详情页面的顶端显示动作名称；页面中部会向用户显示建议的动作持续时间；文字部分则详细地阐述了动作要领；页面底部设有一个“已完成”按钮，点击完成本次训练，并将本次训练课程记录到历史运动数据中，方便用户日后查看，了解自己过去的训练情况。



8.8 运动历史

在运动历史页面可以看到，已完成的所有运动历史，其内容包括该动作的完成时间，名称，以及每个动作需要坚持的时间。





8.9 新闻资讯

点击“休息”，会出现新闻内容，页面中展示许多新闻模块，为用户提供丰富多彩的每日新闻内容。点击任意新闻模块，即可深入阅读更加详细的内容，若想要获得更加全面的信息需要访问手机 app。





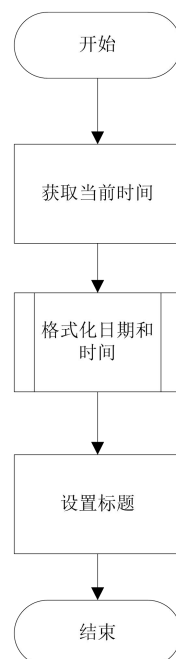
9 关键技术

9.1 获取时间

该功能用于获取当前系统的时间，确保时间信息的准确性和即时性。通过库函数进行操作显示实时时间，如首页的表盘或任何需要显示当前时间的页面。同时，用于计算时间差等与时间相关的操作。

(1) 获取系统时间

1. 流程图如下：



2. 代码如下：

```
// 获取当前系统时间
const currentDate = new Date();
// 格式化日期和时间，例如：'23Feb 14:54'
const formattedDate = this.formatDate(currentDate);
// 设置 title 为格式化后的当前时间
this.date = formattedDate;
```

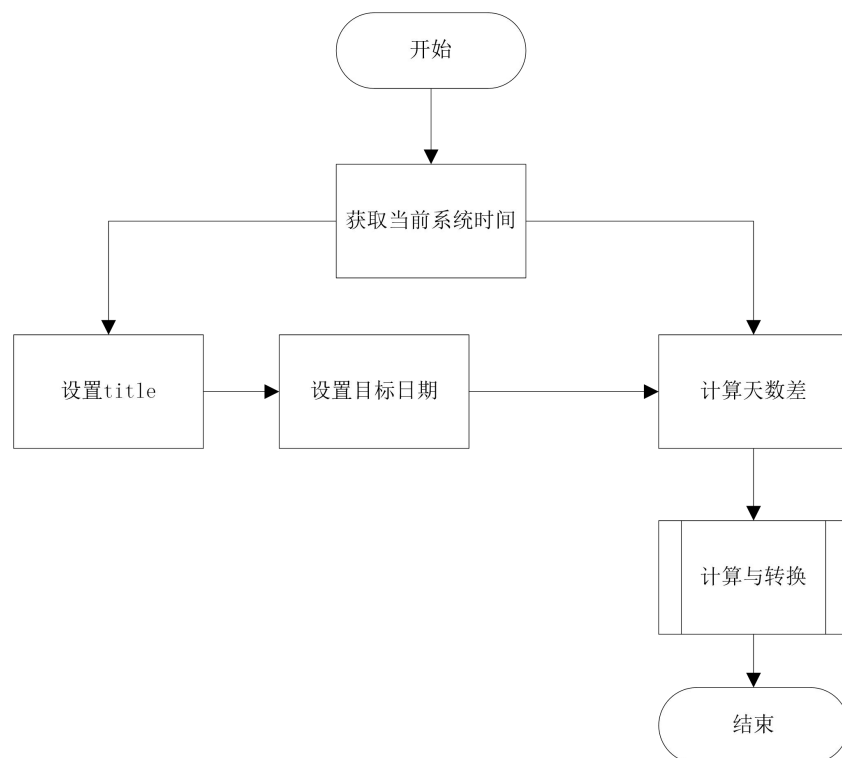
```
formatDate(date) {
  // 格式化日期和时间的函数，根据您的需要调整格式
  const monthNames = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'];
  const day = date.getDate();
  const monthIndex = date.getMonth();
  //const year = date.getFullYear();
  const hours = date.getHours();
  const minutes = date.getMinutes();

  // 确保分钟是两位数
  const formattedMinutes = minutes < 10 ? `0${minutes}` : minutes;

  // 返回格式化后的日期和时间字符串
  return `${day} ${monthNames[monthIndex]} ${hours}:${formattedMinutes}`;
},
```

(2) 计算当前日期与现在日期的时间差

1. 流程图如下：



2. 代码如下：

```
// 设置目标日期为2024年10月19日
const targetDate = new Date(2024, 10, 19); // 注意月份是从0开始的, 所以10月是9

// 计算两个日期之间的天数差
this.days = this.daysBetweenDates(currentDate, targetDate);
```

```
daysBetweenDates(date1, date2) {
  // 获取时间差的毫秒数
  const diff = Math.abs(date2 - date1);

  // 转换成天数
  const days = Math.ceil(diff / (1000 * 60 * 60 * 24));

  return days;
},
```

9.2 天气 API 的接收与请求

通过使用蓝河操作系统中的 fetch 模块进行 HTTP 请求,调用高德的天气 API 接口,获取天气的相关数据,根据返回数据的结构,将需要的数据存放到对应的变量中,以便于在页面上进行显示。

```
<script>
import fetch from '@blueos.network.fetch'; // 引入fetch模块,用于进行HTTP请求

export default {
  data() {
    return {
      tem: '', // 存储温度
      wind: '', // 存储风向
      wetness: '', // 存储湿度
      pro: '', // 存储省份
      cit: '', // 存储城市
      clothes1: '轻薄速干T恤', // 根据温度推荐的上衣
      clothes2: '短裤', // 根据温度推荐的裤子
      we: '', // 存储天气情况
    };
  },

  onShow() {
    // 在组件显示时调用
    fetch.fetch({
      url: 'https://restapi.amap.com/v3/weather/weatherInfo?key=d478911f1c9aea4c3ff9651e0b46e525&city=110107', // 天气API的URL
      method: 'get', // HTTP请求方法为GET
      responseType: 'json', // 返回的数据类型为JSON
      success: (response) => {
        let responseData;
        if (typeof response === 'string') {
          responseData = JSON.parse(response); // 如果响应是字符串,则解析为JSON对象
        } else {
          responseData = response; // 否则直接使用响应数据
        }
        const temperature = responseData.data.lives[0].temperature; // 获取温度
      }
    });
  }
};
```

```

    // 获取并存储其他天气信息
    const winddirection = responseData.data.lives[0].winddirection; // 风向
    const humidity = responseData.data.lives[0].humidity; // 湿度
    const province = responseData.data.lives[0].province; // 省份
    const city = responseData.data.lives[0].city; // 城市
    const weather = responseData.data.lives[0].weather; // 天气情况

    // 更新组件的数据属性
    this.tem = `${temperature}°C`;
    this.wind = `${winddirection}风`;
    this.wetness = `${humidity}%`;
    this.pro = `${province}`;
    this.cit = `${city}`;
    this.we = `${weather}`;

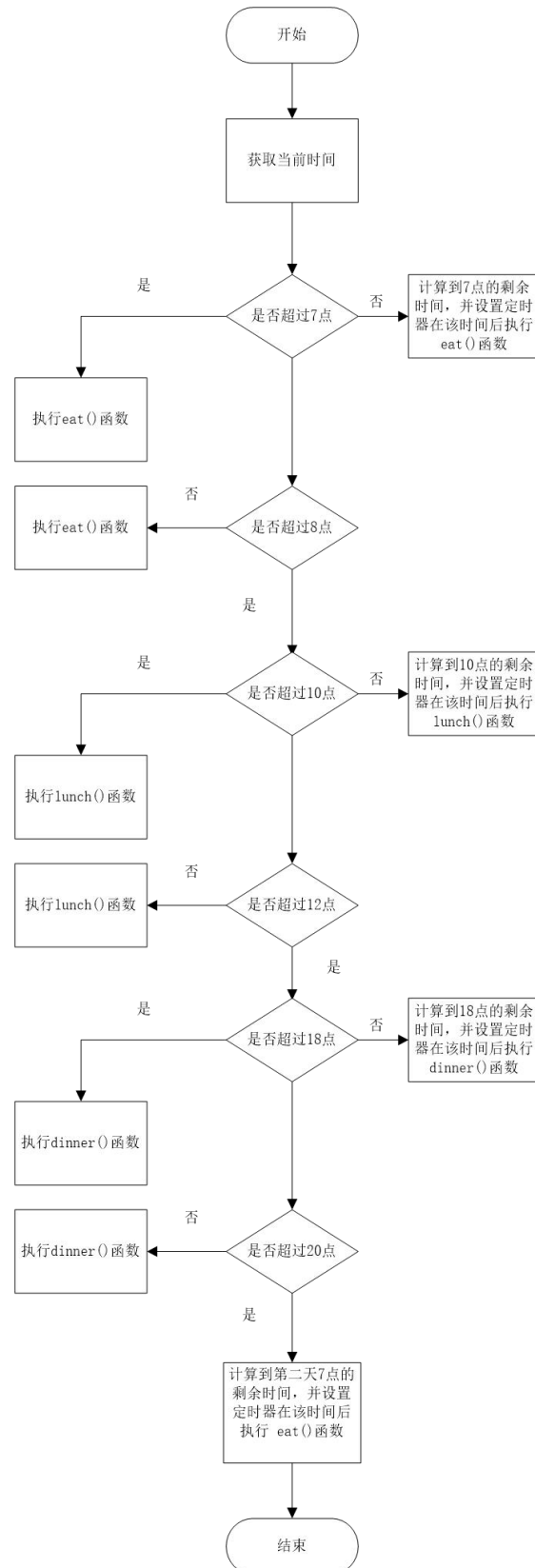
    console.log(responseData); // 打印响应数据到控制台
  },
  fail: (error) => {
    console.error("Fetch error: ", error); // 请求失败时打印错误信息
  }
})
},
}
</script>

```

9.3 根据时间跳转

根据实际时间与设定的饮食建议时间做对比，实现直接跳转到饮食页面。

1. 流程图如下：



2. 代码如下:

```
        this.jump_eat();
    },
    jump_eat(){
        const currentDate = new Date();//获取当前时间
        const hou = currentDate.getHours();//取此时的小时
        const miu = currentDate.getMinutes();//取此时的分钟
        /*以下两个变量的作用为，例如计算17:45距离18:00还有多少分钟
        记录距离下一个整点差多少分钟: tm=60-45=15 分钟
        记录时间间隔: tim= (18-1-17) + tm=15 分钟
        */
        // 测试:
        //const hou = 7;
        //const miu = 0;
        let tm = 0;
        let tim = 0;

        if(hou < 7)
        {
            if(miu > 0)
            {
                tm = 60 - miu;

                tim = ((7 - 1 - hou) * 60 + tm) * 60 * 1000;
            }
            else{
                tim = ((7 - hou) * 60) * 60 * 1000;
            }
            setTimeout(this.eat , tim);
        }
        else if(hou >= 7 && hou <= 8)
        {
            if(hou == 7||(hou == 8&&miu == 0))
                this.eat();
            else{
                tm = 60 - miu;
                tim = ((10 - 1 - hou) * 60 + tm + 30) * 60 * 1000;
                setTimeout(this.lunch , tim);
            }
        }
        else if(hou > 8 && hou < 10)
        {
            if(miu > 0)
```

```

    {
        tm = 60 - miu;
        tim = ((10 - 1 - hou) * 60 + tm + 30) * 60 * 1000;
    }
    else{
        tim = ((10 - hou) * 60 + 30) * 60 * 1000;
    }
    setTimeout(this.lunch , tim);
}
else if(hou >= 10 && hou <= 12)
{
    if(hou == 10 && miu < 30)
    {
        tim = (30 - miu) * 60 * 1000;
        setTimeout(this.lunch , tim);
    }
    else if(hou == 12 && miu != 0)

```

```

    tm = 60 - miu;
    tim = ((18 - 1 - hou) * 60 + tm + 30) * 60 * 1000;
    setTimeout(this.dinner, tim);
}
else {
    this.lunch();
}
}
else if (hou > 12 && hou < 18) {
    if (miu > 0) {
        tm = 60 - miu;
        tim = ((18 - 1 - hou) * 60 + tm + 30) * 60 * 1000;
    }
    else {
        tim = ((18 - hou) * 60 + 30) * 60 * 1000;
    }
    setTimeout(this.dinner, tim);
}
}

```

```

}
else if(hou >= 18 && hou <= 20)
{
    if(hou == 18 && miu < 30)
    {
        tim = (30 - miu) * 60 * 1000;
        setTimeout(this.dinner , tim);
    }
    else if(hou == 20 && miu != 0)
    {
        tm = 60 - miu;
        tim = ((24 - 1 - hou) * 60 + tm + 7 * 60) * 60 * 1000;
        setTimeout(this.eat , tim);
    }
    else{
        this.dinner();
    }
}
else{

```

```

    if(miu > 0)
    {
        tm = 60 - miu;
        tim = ((24 - 1 - hou) * 60 + tm + 7 * 60) * 60 * 1000;
    }
    else{
        tim = ((24 - hou) * 60 + 7 * 60) * 60 * 1000;
    }
    setTimeout(this.eat , tim);
}

```

9.4 数据库

一、数据库的后端代码主要由四个部分构成

1. TrainController

● 概述:

TrainController 类是一个 Spring Boot RESTful 控制器，提供了对 Nexuscore 对象的管理功能。通过 HTTP 请求，客户端可以获取所有记录或添加新记录。


```

8  import com.fasterxml.jackson.databind.ObjectMapper;
9  import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.web.bind.annotation.*;
11
12 import java.util.Date;
13 import java.util.Map;
14
15 @RestController
16 @RequestMapping("/train")
17 @Slf4j
18 public class TrainController {
19
20     @Autowired
21     private TrainMapper trainMapper;
22
23     @GetMapping("/getAll")
24     public Result getAll() {
25
26         return Result.success(trainMapper.getAll());
27     }
28
29     @PostMapping("/add")
30     public Result add(@RequestBody Nexuscore nexuscore) {
31
32         log.info("{} ", nexuscore);
33
34         trainMapper.insert(nexuscore);
35
36         return Result.success();
37     }
38 }
39

```

- 类定义和注解

```

@RestController
@RequestMapping("/train")
@Slf4j

```

- ① @RestController: 声明这是一个 RESTful 控制器。
- ② @RequestMapping("/train"): 基础 URL 路径。
- ③ @Slf4j: 生成日志记录器。

- 获取所有记录的 API

```

@GetMapping("/getAll")
public Result getAll() {

    return Result.success(trainMapper.getAll());
}

```


- ① @GetMapping("/getAll")：映射 HTTP GET 请求，路径为/train/getAll。
 - ② 调用 trainMapper.getAll() 方法，获取所有 Nexuscore 记录。
- 添加记录的 API

```
@PostMapping("/add")
public Result add(@RequestBody Nexuscore nexuscore) {

    log.info("{} ", nexuscore);

    trainMapper.insert(nexuscore);

    return Result.success();
}
```

- ① @PostMapping("/add")：映射 HTTP POST 请求，路径为/train/add。
- ② 使用@RequestBody 将请求体中的 JSON 数据转换为 Nexuscore 对象。
- ③ 记录传入的 Nexuscore 对象信息。
- ④ 调用 trainMapper.insert(nexuscore) 方法，将 Nexuscore 对象插入数据库。

- ⑤ 返回成功结果。

2. TrainMapper

- 概述：

TrainMapper 接口是一个 MyBatis 的 Mapper 接口，定义了用于操作 Nexuscore 表的 SQL 语句。通过这个接口，可以获取所有 Nexuscore 记录或插入新的记录。

```
1 package com.ncut.mapper;
2
3 > import ...
4
5
6
7
8
9
10 @Mapper 2 usages
11 public interface TrainMapper {
12
13     @Select("select * from nexuscore") 1 usage
14     List<Nexuscore> getAll();
15
16     @Insert("insert into nexuscore (date, train_item, duration) values (#{date}, #{trainItem}, #{duration})") 1 usage
17     void insert(Nexuscore nexuscore);
18 }
19
```

- 获取所有记录的方法

```
@Select("select * from nexuscore") 1 usage
List<Nexuscore> getAll();
```

① @Select("select * from nexuscore"): 指定 SQL 查询语句，用于获取 nexuscore 表中的所有记录。

② 方法 getAll(): 返回包含所有 Nexuscore 对象的列表。

● 插入新记录的方法

```
@Insert("insert into nexuscore (date, train_item, duration) values ({date}, #{trainItem}, #{duration}) ") 1 usage
void insert(Nexuscore nexuscore);
```

① @Insert("insert into nexuscore (date, train_item, duration) values ({date}, #{trainItem}, #{duration})"): 指定 SQL 插入语句，将 Nexuscore 对象的属性值 插入到 nexuscore 表中。

② 方法 insert(Nexuscore nexuscore): 接受一个 Nexuscore 对象作为参数， 并将其插入到数据库中。

3. Nexuscore

● 概述

Nexuscore 类是一个数据对象类，用于表示训练记录。它包含训练的日期、项目和持续时间等信息，并提供了序列化和反序列化的支持。

```
package com.ncut.pojo;

import ...

@Data 5 usages
@NoArgsConstructor
@AllArgsConstructor
public class Nexuscore {

    private static final long serialVersionUID = 1L;

    private Long id;

    @JsonFormat(pattern="yyyy-MM-dd",timezone="GMT+8")
    private Date date;

    private String trainItem;

    private Integer duration;
```

● 类定义和注解

```

@Data 5 usages
@NoArgsConstructor
@AllArgsConstructor
public class Nexuscore {

```

① @Data: 自动生成 getter、setter、toString、equals、hashCode 等方法。

② @NoArgsConstructor: 自动生成无参构造函数。

③ @AllArgsConstructor: 自动生成包含所有字段的构造函数。

● 类字段定义

```
private static final long serialVersionUID = 1L;
```

定义一个静态常量 serialVersionUID，用于序列化。

```
private Long id;
```

定义 id 字段，类型为 Long，用于唯一标识 Nexuscore 对象。

```

@JsonFormat(pattern="yyyy-MM-dd", timezone="GMT+8")
private Date date;

```

① @JsonFormat(pattern="yyyy-MM-dd", timezone="GMT+8"): 指定日期格式化模式和时区。

② date 字段，类型为 Date，表示训练日期。

```
private String trainItem;
```

定义 trainItem 字段，类型为 String，表示训练项目的名称。

```
private Integer duration;
```

定义 duration 字段，类型为 Integer，表示训练持续时间，以分钟为单位。

4. RunPulseApplication

● 概述

RunPulseApplication 类是 Spring Boot 应用程序的主类，用于启动应用程序。该类包含应用程序的入口点，并配置了必要的 Spring Boot 注解。

```
package com.ncut;

import lombok.extern.slf4j.Slf4j;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@Slf4j
public class RunPulseApplication {

    public static void main(String[] args) {
        SpringApplication.run(RunPulseApplication.class, args);
    }
}
```

● 类定义和注解

```
@SpringBootApplication
@Slf4j
public class RunPulseApplication {
```

① @SpringBootApplication: 标记这是一个 Spring Boot 应用程序的主类，包含配置、自动配置和组件扫描的功能。

② @Slf4j: 使用 Lombok 注解生成一个名为 log 的日志记录器。

● 主函数

```
public static void main(String[] args) {
    SpringApplication.run(RunPulseApplication.class, args);
}
```

① public static void main(String[] args): Java 应用程序的入口方法。

② SpringApplication.run(RunPulseApplication.class, args): 启动 Spring Boot 应用程序，初始化 Spring 上下文和内嵌服务器。

二、在前端中，通过编写的 JS 函数将训练数据写入数据库，以及将数据库

中的历史训练数据的读出

1. 训练数据的写入:



● 概述:

以弓步压腿的动作详情页面为例，上图为手表页面，将写数据的 JS 函数与已完成按钮绑定，点击此按钮，调用函数，实现将该动作的完成时间、名称和训练时间写入数据库中。

```
<input class="btn1" type="button" value="已完成" @click="transform"/>

transform() {
  // 获取当前系统时间
  const currentDate = new Date();
  // 设置title为格式化后的当前时间
  this.year = currentDate.getFullYear();
  this.month = currentDate.getMonth()+1;
  this.day = currentDate.getDate();

  const data1 = {
    date: `${this.year}-${this.month.toString().padStart(2, '0')}-${this.day.toString().padStart(2, '0')}`,
    trainItem: `${this.title}`,
    duration: `${this.time.replace('s', '')}`
  };

  fetch.fetch({
    url: 'http://localhost:8080/train/add',
    method: 'POST',
    header: {
      'Content-Type': 'application/json;charset=UTF-8',
      // 'Data-Type': 'json'
    },
    data: JSON.stringify(data1),
    success: function(data) {
      console.log('Data sent successfully:', data);
    },
    fail: function(data, code) {
      console.error('Failed to send data:', data, 'Error code:', code);
    }
  })
}
```

● 绑定:

```
<input class="btn1" type="button" value="已完成" @click="transform"/>
```

将 JS 中定义的 transform 函数，绑定到按钮上。

- 定义函数名：

```
transform() {
```

- 获取当前时间：

```
// 获取当前系统时间
const currentDate = new Date();
// 设置title为格式化后的当前时间
this.year = currentDate.getFullYear();
this.month = currentDate.getMonth()+1;
this.day = currentDate.getDate();
```

- ① 通过 Date 函数获取当前的系统时间存放在 currentDate 中。

- ② 对 currentDate 调用 Date 类的方法，获取年月日存放在 data 中定义好的 year、month 和 day 中

- 将获取的时间信息转化为 JS 对象

```
const data1 = {
  date: `${this.year}-${this.month.toString().padStart(2, '0')}-${this.day.toString().padStart(2, '0')}`,
  trainItem: `${this.title}`,
  duration: `${this.time.replace('s', '')}`}
```

- ① 对月份和日，原本是数字，使用 toString 方法转化为字符串，再使用 padStart 方法规定字符串长度 2，若长度不够，则用 ‘0’ 填充。

- ② 对存放持续时间的变量 time 使用 replace 方法，将末尾的 s 替换为 ‘ ’。

- ③ 定义一个 data1 的 JS 对象，date 属性存放格式化好的年月日的字符串，trainItem 属性存放训练项目名称，duration 属性存放格式化好的持续时间。

- 发送 POST 请求


```

fetch.fetch({
  url: 'http://localhost:8080/train/add',
  method: 'POST',
  header: {
    'Content-Type': 'application/json;charset=UTF-8',
    // 'Data-Type': 'json'
  },
  data: JSON.stringify(data1),
  success: function(data) {
    console.log('Data sent successfully:', data);
  },
  fail: function(data, code) {
    console.error('Failed to send data:', data, 'Error code:', code);
  }
})

```

- ① 使用 `fetch.fetch` 方法发送一个 POST 请求到 `http://localhost:8080/train/add`
- ② `url` 指定请求的服务器的地址。
- ③ `method` 指定使用 POST 方法发送数据。
- ④ `header` 指定请求的内容类型为 JSON 格式，并且字符编码为 UTF-8。
- ⑤ 使用 `JSON.stringify` 方法将 JS 对象 `data1` 转换成 JSON 字符串，便于在网络请求中发送。
- ⑥ `data` 指定在网络请求中发送的数据是转换成 JSON 字符串的 `data1`。
- ⑦ `success` 是一个回调函数，请求成功输出打印出传送的数据。
- ⑧ `fail` 是一个回调函数，请求失败打印出错误信息。

2. 历史训练数据的读出



● 概述：

上图为手表页面，将读数据的 JS 函数放到该页面的 OnShow 函数中，打开该页面，执行 OnShow 函数，便能将从数据库中读到的运动的历史数据显示在页面中。

```
<list-item type="item" for="{ { newList } }">
  <text class="text_date">{{ $item.date.substring(0,10) }}</text>
  <text class="text_item">{{ $item.trainItem }}</text>
  <text class="text_seconds">{{ $item.duration + 's' }}</text>
</list-item>
```

```
onShow() {
  fetch.fetch({
    url: 'http://localhost:8080/train/getAll',
    method: 'get',
    responseType: "json",
    success: (response) => {
      let responseData;
      if (typeof response === 'string') {
        responseData = JSON.parse(response);
      } else {
        responseData = response;
      }
      console.log(response)
      this.newList = responseData.data.data;
    },
    fail: (error) => {
      console.error("Fetch error: ", error);
    }
  })
},
```

● 页面显示：

```
<list-item type="item" for="{ { newList } }">
  <text class="text_date">{{ $item.date.substring(0,10) }}</text>
  <text class="text_item">{{ $item.trainItem }}</text>
  <text class="text_seconds">{{ $item.duration + 's' }}</text>
</list-item>
```

① 使用 for 循环，遍历 newList，每一个遍历到的元素都放到 item 中，然后通过 item 的属性来展示数据。

② 从数据库中读出的日期格式与页面期望显示的不同，于是对 item 的

date 使用 substring 方法，对获取到的日期进行切片，使其在页面中显示的是年月 日

③ 从数据库中读出的 duration 它表示的是时间，但是读出来只有数字，没有对应的单位，于是显示的时候在后面加上单位。

● 发送 GET 请求

```
onShow() {  
  fetch.fetch({  
    url: 'http://localhost:8080/train/getAll',  
    method: 'get',  
    responseType: "json",  
    success: (response) => {  
      let responseData;  
      if (typeof response === 'string') {  
        responseData = JSON.parse(response);  
      } else {  
        responseData = response;  
      }  
      console.log(response)  
      this.newList = responseData.data.data;  
    },  
    fail: (error) => {  
      console.error("Fetch error: ", error);  
    }  
  })  
},
```

④ 使用 fetch.fetch 方法发送一个 GET 请求到

<http://localhost:8080/train/getAll>

⑤ url 指定请求的服务器的地址。

⑥ method 指定使用 GET 方法请求数据。

⑦ responseType 指定期望的响应数据类型为 JSON。

⑧ success 是一个回调函数，在请求成功时执行。

⑨ 首先检查响应是否为字符串类型（某些情况下，响应可能会以字符串形式返回）。

⑩ 如果是字符串类型，则使用 JSON.parse(response) 将其解析为 JSON 对

象；否则，直接使用 response。

⑪ 打印 response 数据到控制台以便调试。

⑫ 解析后的数据中 response 的数据结构是 {... data: { ...data: [...]}... }, 于是将 response.data.data 赋值给 newList。

⑬ fail 是一个回调函数，在请求失败时执行。

打印错误信息到控制台，便于调试和错误处理。

9.5 新闻 API 的接收与请求

通过使用蓝河操作系统中的 fetch 模块进行 HTTP 请求，调用综合新闻 API 接口，获取当日综合新闻的相关信息，根据返回数据的结构，将需要的数据存放到对应的变量中，以便于在页面上进行显示。

```
import fetch from '@blueos.network.fetch'; // 引入fetch模块，用于进行HTTP请求
import router from '@blueos.app.appmanager.router'; // 引入router模块，用于页面跳转

export default {
  data() {
    return {
      direction: ['up'], // 方向数组，默认为'up'
      newList: [
        {
          source: 'IT家科学探索', // 新闻来源
          title: '国产化率达 85% 以上，全球首款江海直达型 LNG 加注运输船“淮河能源启航”号在上海交付', // 新闻标题
          description: '据“浦东发布”消息，7月19日，全球首款、中国首制江海直达型14000立方米液化天然气（LNG）加注运输船“淮河能源启航”号提前2个月在上海命名交',
          ctime: '2024-07-20 10:00:10', // 新闻发布时间
          url: 'https://www.ithome.com/0/783/097.htm', // 新闻链接
          path: '/pages/breakdetail' // 页面路径
        },
        {
          source: '中华国内',
          title: '新疆巴音郭楞州尉犁县发生4.5级地震',
          description: '', // 新闻描述
          ctime: '2024-07-20 00:00:00',
          url: 'https://news.china.com/domestic/945/20240720/46884384.html',
          path: '/pages/breakdetail'
        },
        {
          source: '南方社会',
          title: '温州一医生遭男子持刀伤害致多处严重损伤，经抢救无效不幸去世',
          description: '目前，相关善后工作正在有序开展。',
          ctime: '2024-07-20 09:54:00',
          url: 'https://news.southcn.com/node_179d29f1ce/b0b661d70d.shtml',
          path: '/pages/breakdetail'
        }
      ]
    }
  }
}
```

```

    },
    {
      source: '南方社会',
      title: '温州一医生遭男子持刀伤害致多处严重损伤，经抢救无效不幸去世',
      description: '目前，相关善后工作正在有序开展。',
      ctime: '2024-07-20 09:54:00',
      url: 'https://news.southcn.com/node_179d29f1ce/b0b661d70d.shtml',
      path: '/pages/breakdetail'
    },
    {
      source: '南方社会',
      title: '中央广播电视总台传达学习党的二十届三中全会精神',
      description: '7月19日，中央广播电视总台分别召开党组（扩大）会议和全台中层以上干部视频会议，传达学习党的二十届三中全会精神，部署总台学习宣传贯彻全会精神。',
      ctime: '2024-07-20 10:25:00',
      url: 'https://news.southcn.com/node_179d29f1ce/b6a155a693.shtml',
      path: '/pages/breakdetail'
    },
    {
      source: '南方社会',
      title: '彭丽媛出席“爱在阳光下”——中非儿童手拉手夏令营活动',
      description: '7月19日下午，国家主席习近平夫人、世界卫生组织结核病和艾滋病防治亲善大使、联合国教科文组织促进女童和妇女教育特使彭丽媛在北京城市图书馆出席“爱在阳光下”——中非儿童手拉手夏令营活动。',
      ctime: '2024-07-20 10:27:00',
      url: 'https://news.southcn.com/node_179d29f1ce/88eeab8b7a.shtml',
      path: '/pages/breakdetail'
    },
  ], // 新闻列表，包含若干新闻条目，每个条目包含新闻的来源、标题、描述、发布时间、链接和页面路径
},
onShow() {
  // 在组件显示时调用
  fetch.fetch({
    url: 'https://whyta.cn/api/tx/generalnews?key=150ff6d47b76', // API的URL
    method: 'get', // HTTP请求方法为GET
    responseType: "json", // 返回的数据类型为JSON
    success: (response) => {
      let responseData;

```

```

      let responseData;
      if (typeof response === 'string') {
        responseData = JSON.parse(response); // 如果响应是字符串，则解析为JSON对象
      } else {
        responseData = response; // 否则直接使用响应数据
      }

      // 检查responseData.data是否存在
      if (!responseData.data) {
        console.error("responseData.data is undefined");
        return;
      }

      // 提取新闻列表
      this.newsList = responseData.data.result.newslist;

      // 检查newsList是否存在
      if (!this.newsList) {
        console.error("newsList is undefined");
        return;
      }

      // 对新闻列表进行遍历和处理
      this.newsList.forEach(news => {
        console.log(`新闻标题: ${news.title}`);
        console.log(`新闻描述: ${news.description}`);
        console.log(`新闻来源: ${news.source}`);
        console.log(`发布时间: ${news.ctime}`);
        console.log(`新闻链接: ${news.url}`);
        console.log('-----');
      });

    },
    fail: (error) => {
      console.error("Fetch error: ", error); // 请求失败时打印错误信息
    }
  })
},

```

10 过程中遇到的问题

10.1 天气获取

构建天气应用的过程中，我们期望从开发文档中获取到现成的天气接口，以便能够轻松地集成实时天气数据。然而，开发文档未能提供所需的天气接口。

解决办法：

为集成天气信息，我们选用了高德地图天气 API，它提供全面且易于集成的数据服务。首先，在高德地图官网注册并获取 API 密钥。获得密钥后，我们通过编程发送 HTTP 请求到 API，附带必要的请求参数，以获取 JSON 或 XML 格式的天气数据。

10.2 数据库存储

在开发过程中，我们最初可能使用沙箱来存储数据，以快速验证应用的功能。然而，随着开发的深入和数据的增多，使用沙箱来存储数据变得不再合适。

解决办法：

为保障数据持久性和可用性，我们将数据存储于稳定数据库。首先，在开发环境中设置数据库服务器，创建数据库及所需表结构，依据应用需求设计表结构。随后，编写代码连接数据库，实现数据增删改查操作，利用数据库连接库简化操作。最终，通过访问权限控制、数据加密、定期备份等措施，确保数据库安全，防止数据泄露或未授权访问。

10.3 真机与模拟器

在 BlueOS Studio 集成开发环境上编写的程序，无法使用 adb，通过 USB 连接手机，手机通过蓝牙连接手表，将程序导入到手表上调试。

解决办法：

在确保手机打开了开发者模式的条件下，使用 USB 连接手机后，手机传输选项将仅充电改为仅传输文件即可。

11 组织架构及职责

为确保项目的顺利进行，成立了李俊杰、谢邹健、张诗语的三人工作小组。在使用 git 和及时设计协同开发的工作中，为了人尽其才，保质保量地完成好工作，现作分工如下：

李俊杰：

1. 将设计的 UI 页面转化为实际可运行的代码。熟悉前端技术栈以及开发文档，如 HTML、CSS 和 JavaScript，以确保页面正常显示和交互。

2. 实现页面内容的同时，通过 js 语言的编写来实现获取系统时间的功能。这个功能将帮助用户实时了解当前的时间信息，提升应用的实用性。使用 Git 进行版本控制，确保代码的可靠性和可维护性。

谢邹健：

主要任务是连接天气 API 和编写数据库。

1. 在网上搜索并筛选出合适的天气 API，了解 API 的接口文档和调用方式，以便在应用中获取实时的天气信息。在连接 API 的过程中，处理可能出现的网络错误和数据异常，确保数据的准确性和稳定性。

2. 进行数据库的编写和连接工作，为应用提供数据存储和查询的功能。这包括设计数据库结构、编写 SQL 语句以及实现数据的增删改查等操作。使用 Git 进行版本控制，确保代码的稳定性和可维护性。

张诗语：

主要负责开展用户调研的工作，旨在深入了解目标用户群体的需求与偏好，通过收集和分析用户数据，形成对产品设计的依据；基于调研结果，将定义 app 的交互逻辑，确保用户在使用过程中能够顺畅、自然地完成任务；还将负责制定用户操作流程，简化操作步骤，以提升整体的用户体验；界面 UI 设计，通过运用专业的设计理念和工具，使用户在使用过程中获得愉悦的视觉体验。

12 比赛收获

通过项目的实施，加强和锻炼了团队协作创新的能力；每位组员在边学边干中，不仅掌握了新的知识，还提升了系统思考的能力，也加强了实际操作能力的锻炼。具体如下：

李俊杰：

在参加这个项目之前，我已经对 ml、css 以及 js 等语言有一些基础知识。但是，项目的要求我们进行创新型开发交互，这让我觉得我可以参与进来进一步熟悉这些知识。

在项目开始后的前几周，我们花了很多时间来理解项目的需求和，并构思项目应该从哪些方面去进行创新交互设计。

随着项目的推进，我们遇到了一些挑战，但在团队的共同努力下，我们克服了这些问题，最终成功地完成了项目。

通过这个项目让我学到了很多，我深入了解到蓝河操作系统手表应用开发，进一步熟练使用 ml、css 以及 js 等语言，同时通过及时沟通和查阅文档学习到一些之前没有接触过的内容。我很高兴能够参加这样的项目，我期待未来能继续在应用开发方面不断进步。

谢邹健：

在参加这个项目之前，我已经掌握了一些前端 api 调用、后端 SpringBoot 框架的一些基础知识以及简单的 SQL 语句。完成这个项让我可以更加熟悉这些知识，增强我的实战能力。

在项目开始后的前几周，我们花了很多时间来理解项目的需求并构思项目应该从哪些方面去进行创新交互设计，以及我负责的部分该如何实现

随着项目的推进，我由于不熟悉遇到了一些挑战，但在团队的共同努力下，以及我不断地学习钻研，最终配合其他两位同志，成功地完成了项目。

通过这个项目让我学到了很多，我深入了解到蓝河操作系统手表应用开发，进一步熟练了 api 调用、SpringBoot 框架和 SQL 的基础知识及语句，同时通过及时沟通、网络搜索资料、查阅文档学习到了许多新的知识。我认为参加这样的项目是非常有意义的，我期待未来能继续在应用开发方面不断进步。

张诗语：

在比赛中，我主要负责界面设计相关工作。此次比赛让我更好地实践之前学习的交互设计、页面设计相关的专业知识与理论，收获颇丰。

首先，在交互设计方面，我运用了以用户为中心的设计理念，关注用户在使用过程中的体验。我通过了解用户的需求和痛点，以简单、直观的操作方式来设计交互流程。同时，我运用了交互设计中的可用性原则，如一致性、反馈、简洁性等，确保用户在使用过程中能够快速上手，提高使用效率。

其次，在页面设计方面，我注重信息的布局与呈现。我运用了格式塔原理，通过对比、重复、对齐和亲近性等原则，使界面元素具有良好的视觉层次，提高用户的阅读体验。同时，我还关注色彩的运用，通过色彩搭配来突出重点，引导用户的注意力，增强界面的可读性。

此外，在团队协作方面，我与编程同学保持密切沟通，确保设计能够顺利实现。我学会了如何将设计稿精确还原为实际界面，并在过程中不断调整和优化，以达到最佳的用户体验。

最后，也十分荣幸可以参加本次比赛，本次比赛不仅提升了我的专业技能，也锻炼了我的解决问题和创新思维的能力，让我更加明白了在以后的学习和工作中怎样去更好地关注用户需求，运用专业知识与理论，为用户创造价值！

13 总结

为了贯彻落实好以人民健康为核心的任务，加快推进自主科技与健康的安全和发展，基于 Vivo 使用 Rust 语言编写的蓝河操作系统，我们开发的 RunScheme 软件，在实现了 Vivo 已开发软件所涵盖的运动指导、运动数据记录、科学的健身计划功能的基础上，加入了根据天气情况作出的穿搭推荐、根据用户自身情况和训练计划作出的饮食推荐以及天气和新闻的实时资讯功能，解决了原有应用在运动一体化的相关运用中存在的问题，使 Vivo Watch3 的运动健康应用，功能更加丰富，用户的可选择性更高，手表在市场上的竞争力更强，实现健康与科技的融合，助推了健康中国、数字中国、总体国家安全观的建设。