前端技术要求：  
框架：react-native 测试开发使用：expo  
语言：typescript  
Api请求：GraphQL  
状态管理：zustand

**项目整体格式（附录1）**

1. 整体采用expo的文件路由方式；  
2．组件的命名方式为‘\_’+小写字母的驼峰命名，例如：’\_componentDemo1.tsx’  
将公共组件单独放在项目下的文件夹中，公共组件的命名为大写字母开头的驼峰命名，例如’ComponentDemo1.tsx’;  
3．将公共字体常量和颜色常量抽离，分别保存在自己所属的文件中；  
4．每个页面保存一个文件夹，隶属在上一层下，文件夹中有Components文件夹（用来存放页面组件）、connect文件（用来存放GraphQL请求代码）、zustand文件（用来存放当前页面下的状态管理）、page文件（页面的具体样式）  
5．页面之间的跳转使用router进行跳转；

**整体开发流程**

编写前端页面，从时间周期来讲，会经历 组件编写--->页面编写--->状态管理--->接口对接--->接入sdk 阶段

**组件阶段**

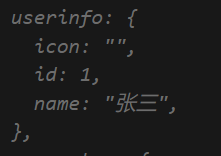
**编写前**

首先要根据后台的设计图，需要对整体项目有一个框架，要知晓具体的实现流程。然后就要对组件进行拆分，公共组件放在公共组件文件夹中进行编写，页面私有组件放在具体页面文件夹下的组件文件夹中进行编写。

**编写时**

编写时，在组件层级下，需要对数据接口数量、类型有一个具体的判断，为数据传递时留有接口，测试时可以选择使用ai根据具体的接口格式生成一些mockdata来进行测试。

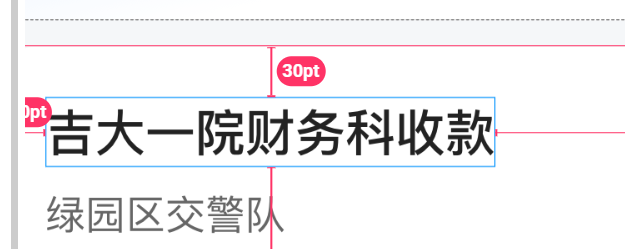
例如



**问题及解决办法**

**margin和pending会导致样式改变**

例如



外pendingTop（30）= 内marginTop（30）= 外pendingTop（15）+ 内marginTop（15）

有时候会导致参数与设计图不对，所以我们可以采取的方法是只使用margin 或者 只是用pending即可

**关于view的用法**

一般来讲我们应该采用view包裹的格式，如果有字体或者对应组件的调用，都使用view包裹如果需要调整位置 只需要调整view的对应参数即可

**view嵌套**

在写view，考虑下面的情况



一些组件在上面，一些组件在下面，这样我们在更改适配时，会很麻烦，我们可以采取的方法是

<view> //设置参数上下填充

<view>

//上部组件

<view>

//component\_1

</view>

<view>

//component\_2

</view>

</view>

<view>

//下部组件

</view>

</view>

**......**

**页面阶段**

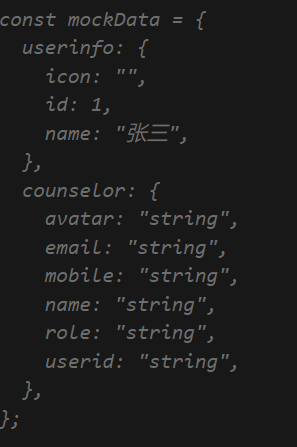
**编写前**

检查页面中所有用到的组件，查看是否所有已经编写完毕。

**编写时**

调整所有组件在页面中的格式，确定页面整体的数据格式，编写接口和mockdata进行测试。

例如



**问题及解决办法**

**格式**

有一些组件可能在编写时便添加了在页面的格式，会导致在页面中添加参数后的格式不对，可以根据实际情况，选择组件内设置还是页面中设置即可。

**......**

**状态管理**

状态是用来管理对应组件的显示逻辑，我们会遇到点击后产生弹窗的情况，例如



但是他只是一个组件，都是编写在这个页面中的，所以，我们便把他编写在同一个页面下，用一个布尔参数他的显示与否，这种参数就是状态管理。

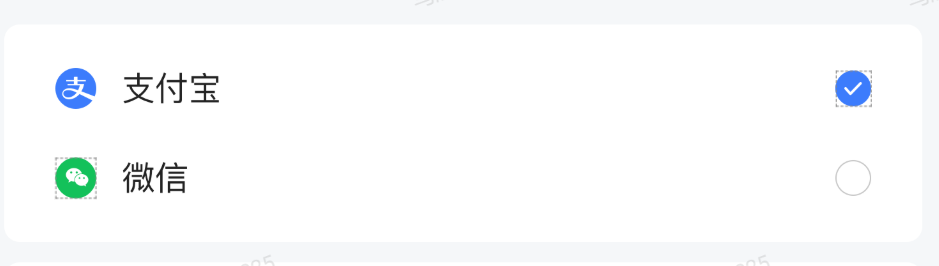
PS：数据请求后，也可以设置一个数据格式的参数来存储请求到的数据，进而控制前端组件的展示，这何尝不是一种状态管理呢😀？

**编写前**

确定页面中，状态的数量和格式，将组件中的参数和状态管理文件中的参数进行绑定，具体格式可以根据选用的状态管理工具来实现。

**编写时**

编写时，注意组件数据之间的联动性，并且需要注意，各个状态之间的联动关系，例如



付款方式的选择只能选择一种，选择支付宝时，便要将微信的是否选中参数调整为false，其他的还有很多，需要注意。

**问题及解决办法**

**状态刷新**

状态的改变是随时生效的，不会存在异步的问题。理解方式是 我们将具体的页面参数绑定到了状态管理中的参数，因此我们改变页面中的参数就是改变了状态管理中的参数，二者是共同进行的。

**状态改变连带函数的执行**

这种复杂的状态改变后的副作用的执行，我们可以使用useEffect（react中，具体情况视语言而定）监听状态的改变，来执行比如刷新等操作。

**......**

**接口对接**

由于前端和后端的开发是并行进行的，因此会出现前端和后端对一个功能模块理解不一致的情况，尽管在项目开始前已经进行了约定，但进行到这一阶段还会出现很多问题，这也是最麻烦的阶段。

**编写前**

从后端所给的api格式中，使用postman确定正确性

**编写时**

根据postman中正确的语句进行编写

**问题及解决办法**

**异步问题**

数据请求的过程中涉及到从服务器中读取，在这个过程会存在时间的问题。假如在第一次加载时可以设置一个加载页，判断数据为空时显示正在加载，当加载时需要将函数设置成async，等待数据成功传递之后才执行下一步加载。

**后端接口增添**

不要回答！！！ 不要回答！！！ 不要回答！！！

========================================================

当真的不得不需要更改时（以graphQL举例），需要在postman中重新确认传递的数据格式，再在对应的接口网址下载结构文件（schema.graphql)，然后根据其中的类型结构修改已经编写完成到graphQL语句，之后重新运行npm run generate（用以生成请求函数的指令，可搜索codegen），接着再根据请求到的数据格式重新在页面中接受数据并渲染.....

========================================================

所以 不要回答！！！ 不要回答！！！ 不要回答！！！

**......**

**sdk接入**

从这里开始基本已经完成了，下面就要接入所需的第三方的sdk，例如微信支付和登录、支付宝支付等等，具体要求查找对应的官方文档即可。

**问题及解决办法**

**适配语言**

一些第三方库可能所支持的语言不同，例如react-native，就需要等编译成原生的代码（java，swift）才能接入。

**最后生成对应的apk和ipa上架到应用商店便完成了所有的app开发的内容啦！**

**结语**

恭喜你，学会了我用两个月的时间所学习的所有内容，应该可以让你少走一些弯路！！

马渤皓 2025-2-21 17：08

**附录一**

**project/ # 项目根目录**

**├── app/ # 应用主目录**

**│ ├── (auth)/ # 需要登录认证的路由组**

**│ │ ├── page1/ # 页面1**

**│ │ │ ├── components/ # 页面1私有组件**

**│ │ │ │ ├── \_componentDemo1.tsx**

**│ │ │ │ └── \_componentDemo2.tsx**

**│ │ │ ├── connect.tsx # API 接口请求封装**

**│ │ │ ├── page.tsx # 页面1主组件**

**│ │ │ └── zustand.tsx # 页面1状态管理**

**│ │ └── page2/ # 页面2（结构同page1）**

**│ │**

**│ ├── (unauth)/ # 无需登录认证的路由组**

**│ │ ├── \_layout.tsx # 布局组件**

**│ │ ├── +not-found.tsx # 404页面**

**│ │ ├── page.tsx # 主页面**

**│ │ └── zustand.tsx # 状态管理**

**│ │**

**│ ├── asset/ # 静态资源**

**│ │ ├── image/ # 图片资源**

**│ │ │ └── image1.png**

**│ │ └── sound/ # 音频资源**

**│ │ └── sound1.mp3**

**│ │**

**│ ├── components/ # 全局共享组件**

**│ │ └── ComponentDemo1.tsx**

**│ │**

**│ ├── constants/ # 全局常量配置**

**│ │ ├── Colors.ts # 颜色常量**

**│ │ ├── Enum.ts # 枚举定义**

**│ │ └── Text.ts # 文本常量**

**│ │**

**│ └── sdk/ # 第三方服务集成**

**│ └── wechat.ts # 微信相关功能封装**