移动APP总体架构设计

**修订说明**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **编号** | **日期** | **描述** | **版本** | **作者** | **审核** | **发布日期** |
|  | 2017.05.13 | 创建 | 0.1 | 丁豪 |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

目录

[1 总体设计说明 4](#_Toc482631542)

[1.1 概述 4](#_Toc482631543)

[1.2 适用范围 4](#_Toc482631544)

[1.3 术语说明 4](#_Toc482631545)

[2 业务流程 4](#_Toc482631546)

[3 总体技术架构设计 4](#_Toc482631547)

[3.1 总体架构图 4](#_Toc482631548)

[3.2 技术栈说明 4](#_Toc482631549)

[3.3 网络通信 4](#_Toc482631550)

[3.4 图片缓存 4](#_Toc482631551)

[3.5 异常情况技术方案 4](#_Toc482631552)

[3.6 web内嵌设计 5](#_Toc482631553)

[3.7 消息推送方案 5](#_Toc482631554)

[3.8 设计模式（MVC、MVVM、MVP） 5](#_Toc482631555)

[3.9 日志设计方案 5](#_Toc482631556)

[3.10 分享技术方案 5](#_Toc482631557)

[4 API接口设计 5](#_Toc482631558)

[5 数据采集方案 5](#_Toc482631559)

[6 安全设计 5](#_Toc482631560)

[6.1 总体安全设计 5](#_Toc482631561)

[6.2 网络通信安全设计 5](#_Toc482631562)

[6.3 android安全加固方案 5](#_Toc482631563)

[6.4 密码键盘、手势密码、指纹登录设计方案 6](#_Toc482631564)

[6.5 用户token设计方案 6](#_Toc482631565)

[7 app发布方案 6](#_Toc482631566)

[7.1 版本升级技术方案 6](#_Toc482631567)

[7.2 灰度发布技术方案 6](#_Toc482631568)

[8 数据存储方案 6](#_Toc482631569)

# 总体设计说明

## 概述

描述此技术文档的目的

## 适用范围

描述此技术文档面向的对象

## 术语说明

重点技术术语说明

# 业务流程

描述重点业务流程，从整体上认知业务

# 总体技术架构设计

## 总体架构图

项目结构

|——project

|——build //代码发布目录，用于存放webpack相关配置和脚本

|——config //项目配置目录，区分开发环境、测试环境和线上环境

|——node\_modules //项目开发依赖的一些模块

|——src //开发目录，重点工作目录

|——components //组件目录，内部又可分为多个组件目录

|——costComponent //费用（各种缴费）组件

|——slideSwitchComponent //滑动开关组件

|——...

|——view //组件目录

|——router //路由目录

|——index.js //首页路由

|——... //分页路由

|——App.vue //根组件

|——main.js //入口文件

|——static //静态资源存放目录

|——css

|——js

|——img

|——test //存放测试文件及日志文件

|——.babelrc //语法配置文件

|——.editorconfig //编辑配置文件

|——.gitignore //git配置文件

|——.postcssrc.js //postcss配置文件

|——index.html //首页

|——package.json //项目依赖配置文件

|——README.md //说明文件

## 技术栈说明

Vue版本非断崖式更新，最近的版本一直在强化服务端渲染，建议采用最新版本，便于后期维护。

Vue版本：2.2.6

Vue-router版本：2.3.1

Axios版本：0.16.1

Vuex版本：2.3.1

Node版本：>= 4.0.0

Npm版本：>= 3.0.0

Webpack版本：2.3.3

1. 网络通信层：原生js使用ajax进行浏览器端与服务端通信。Vue2.0使用axios进行网络通信。
2. 项目骨架：vue提供一个官方命令行工具vue-cli，可用于快速搭建一个基于webpack模板的项目。提供开箱即用的构建工具配置，带热重载、保存时静态检查以及可用于生产环境的构建配置的项目。
3. Ui层

（1）根组件（缴费首页）app.vue，组件划分。http请求数据统一放在api文件夹统一管理，父子组件通信用props属性传递和emit发送事件。

（2）水费、电费、煤气费、固话费、宽带费、油卡充值、有线电视、交通违章、手机费9个小模块，写一个公共组件，v-for循环遍历出来，需要服务端提供数据接口。

（3）确定、查询、滑动开关等按钮做成公共组件并export，在有需要的页面import。

（4）组件涉及到路由用vue-router，嵌套路由，使用 children 配置项。

（5）涉及到state的全局管理用vuex，根据具体需求确定是否使用vuex。

（6）表单获得焦点弹出键盘，需要协调服务端、移动端提供接口。

（7）移动端1像素要用css3的transform：scale处理。

（8）若后期页面内容过多，页面更完美，不希望出现滚动条，可使用兼容性比较好的better-scroll或者vue的插件vue-scroll。

（9）移动端布局使用弹性布局flex。考虑flex标准版本display：flex的兼容性问题，兼容安卓4.4，ios7.1。

（10）vue插件：

数据可视化：若后期有需要柱状图、饼图，可使用vue插件Vue-ECharts。

懒加载：Vue懒加载插件vue-lazyload。

轮播插件：vue-slide或者vue-swiper。

（11）页面动画实现可以用vue的transition过渡和css动画animation。注意使用animate.css。

## 网络通信

浏览器和Web服务器之间一问一答的交互过程遵守HTTP协议。HTTP协议是TCP/IP协议集中的一个应用层协议，定义了浏览器和Web服务器之间交换数据过程和数据本身的格式。通信协议使用http，技术使用ajax。ajax指代了浏览器端通过javascript发起http请求的一系列技术。主要目的是解决之前[web网页](https://www.baidu.com/s?wd=web%E7%BD%91%E9%A1%B5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkP1K-rAcdnHc3rAD3uADd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjnzP16dnWczrjDYn1nLnj6Y)无法实现网页局部刷新的问题。AJAX的核心就是通过javascrit程序调用而发起的一次http请求。

在vue里面，http请求是以插件的形式来完成的，vue1.0官方推荐使用vue-resource，vue升级到2.0以后，官方不再更新vue-resource，转而支持axios。目前主流的 Vue 项目，都选择 [axios](https://github.com/mzabriskie/axios) 来完成 ajax 请求，而大型项目都会使用 Vuex 来管理数据。

## 图片缓存

缴费模块大多只是些小图标，图片可利用CSS Sprite。背景图或可根据设备dpi的不同，分别切换到2x或者3X图即可。浏览器自带缓存功能即可，不需要设置缓存。

## 异常情况技术方案

代码异常：  
  模块异常：引入webpack的代码分割功能，出错的模块会停止加载并在控制台打印出错文件索引。  
     javascript语法异常：针对代码块使用try,catch包装，出错代码块会在catch里被捕捉；通过window.addEventListener(“error”, function(evt){})，在回调函数中进行处理；对于压缩后的代码，启用source map以获取行数及错误位置。

网络异常：  
 使用promise进行异步请求，在reject参数中写入回调函数，根据xmlHttpReques对象的status属性弹出dialog弹框，按照需求要求，提供错误信息和错误代码。  
     网络中断导致错误，提示“网络连接中断或暂不可用，请检查网络”。   
 通讯或页面加载时长超过10秒，提示“请求超时，请重新操作”，关闭提示框然后返回上一页面。

## web内嵌设计

描述h5内嵌技术方案，包括具体技术方案、提供的h5接口格式、接口范围

## 消息推送方案

描述消息推送技术方案，是否使用第三方推送技术

## 设计模式（MVC、MVVM、MVP）

对于传统的MVC框架，Model 频繁发生变化时，开发者需要手动更新到View，而对于View层的操作导致 的Model 变化，开发者也需要将变化的数据同步到Model 中，这导致视图层和数据层之间的处理越来越复杂，数据状态很难设计和维护。  
 在MVVM架构下，View 和 Model 之间并没有直接的联系，而是通过ViewModel进行交互，Model和ViewModel之间的交互是双向的， 因此View 数据的变化会同步到Model中，而Model 数据的变化也会立即反应到View上。  
 这使得View 和 Model 之间的同步工作完全自动化，无需人为干涉，因此开发者只需关注业务逻辑，不需要手动操作DOM,不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理。  
 Vue.js是MVVM架构的最佳实践，专注于 MVVM 中的 ViewModel，不仅做到了数据双向绑定，而且也是一款轻量级的JS库。  
 所以，本模块决定使用MVVM模式。

## 日志设计方案

描述日志的详细设计方案，日志等级分类，存储、采集等

## 分享技术方案

描述分享的技术方案，包括使用的第三方技术

## 第三方登录

描述第三方登录的详细技术方案

## 二维码、人脸、银行卡扫描

描述使用的第三方技术

# API接口设计

（1）英文单词不要拼错，时态也不要错。

返回booleans的判断函数，单数要用 is 复数要用are，这样你的命名就和文档中的描述保持了一致性。

（2）属性命名最好是定语+名词

比如 userName, maxSize, textColor

（3）保持方法的对称性，有些方法一旦出现就应该是成对的

比如 有open就要有close，有alloc就要有free，有add就要有remove，这些单词基本是固定搭配的，使用者就很容易理解。

（4）良好的接口说明文档

接口文档要清晰、明了，包含多少个接口，每个接口的地址、参数、请求方式、数据交换格式、参数是否必填、编码格式UTF8，返回值等都要写清楚。

（5）接口数据、状态

接口必须提供明确的数据状态信息，不管是成功的，还是失败的，都必须返回给APP客户端。

（6）简洁

（7）空字段

接口遵循“输入宽容，输出严格”原则，输出的数据结构中空字段的值一律为 null

（8）分页

请求某个资源集合时，可以通过指定 count 参数来指定每页的资源数量，通过 page 参数指定页码。

ios，android提供键盘接口，缴费首页，缴费查询，缴费支付。

1、登录接口

请求说明

url http://域名/baixin/user/stateLogin

格式 Json

请求方式 Post

参数说明

userID String 用户ID

userName String 用户名

2、注册接口

请求说明

url http://域名/baixin/user/login

格式 Json

请求方式 Post

参数说明

Phone String 用户电话号码

User String 用户名

3、缴费订单

请求说明

url http://域名/baixin/user/pay

格式 Json

请求方式 Post

参数说明

payMoney String 提供缴费的金额

User String 用户名

code String 缴费的验证码

# 4.1 vue开发规范

1、开发尽量使用ES6。

2、操作数据，不是万不得已，尽量不要操作DOM。

3、使用vue中的语法糖,比如可以用:style代替v-bind:style；用@click代替v-on:click

4、不要在js里绑定业务相关的事件，业务事件与逻辑，绑定在HTML上，如果有定时器，组件销毁时，定时器也要销毁。

5、命名。组件名称（包括路由组件）使用“-”分割，比如abc-def，不要使用驼峰命名

6、事件。在派发和广播事件时，事件的名称也使用"-"来分割，而且前缀为该组件的名称，不能使用驼峰 ，例如当前组件为open-layer.vue，则事件名称为 open-layer-close

7、不要将HTML的Attributes和Vue的Model混用

比如最终要实现的代码是：

<img src=123.jpg>

<a href="index.html?id=123"></a>

<div id="id-123"></div>

比如Vue实例为：

new Vue({

el: 'body',

data: {

id: 123

}

})

错误 的使用是：

<img src={{ id }}.jpg>

<a href="index.html?id={{ id }}"></a>

<div id="id-{{ id }}"></div>

正确 的使用是：

<img :src=id + '.jpg'>

<a :href="'index.html?id=' + id"></a>

<div :id="'id-' + id "></div>

8、资源的高度可复用（组件、指令、过滤器）

为了使组件、自定义指令、自定义过滤器高度可复用，需要尽可能地将可复用的内 容单独拆离，将组件放在components目录中（如果需要可以在components目录创建子目录）。 将自定义指令放在directives目录中（如果需要可以在directives目录创建 子目录目录创建子 目录）。将自定义过滤器放在filters目录内（如果需要可以在以在filters目录创建子目录）

9、css的使用

可将业务型的CSS代码单独写一个css文件，比如首页，index.css，放置在 styles 目录 内（如果需要，可以在styles目录内再创建目录），在使用index.css的组件内，使用@import 引入css，对于局部的css，可以在style上增加scoped，在style中lang属性添加用什么预 处理器（less，sass）

# 4.2 webpack开发规范

1、entry配置入口js文件。

2、output中定义构建文件的输出位置，以及文件名，其中包含path和filename

3、module中加载相关模块，通过正则判断文件的后缀名，根据定义的不同引用不同的加载 器。

4、配置loader，将资源代码作为参数，返回新的代码，配置loader加载那些文件，不加载那 些文件

5、使用babel-loader解析es6语法，然后我们编写的JS文件可以被编译成可被低版本浏览器 处理的常规代码

# 数据采集方案

网站数据统计分析，根据用户浏览目标网站的行为（如打开某网页、点击某按钮、将商品加入购物车等）及行为附加数据（如某下单行为产生的订单金额等）。主要实现流量分析、来源分析、内容分析及访客属性等常用分析视角。将采集的数据，请求一个后端接口，将信息返回给后台。

# 安全设计

## 总体安全设计

描述app总体安全架构方案

## 网络通信安全设计

描述网络通信安全加密方案、防篡改方案

## android安全加固方案

描述Android安全加固技术方案

## 密码键盘、手势密码、指纹登录设计方案

描述密码键盘设计方案、手势密码、指纹登录详细设计方案

## 用户token设计方案

不再使用session方案，使用token方案的话，如何和服务器交互，如何保存token，token的结构，token的失效期等

# app发布方案

## 版本升级技术方案

版本如何做到无感知升级，如何做到？

## 灰度发布技术方案

如何做到灰度发布，灰度发送的维度

# 数据存储方案

是否需要存储数据、存储数据类型、数据的安全性等