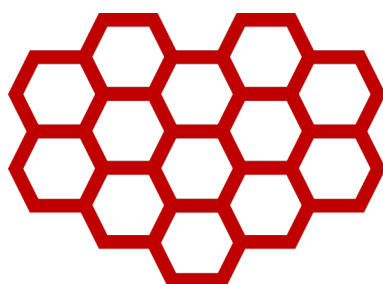


至简网格客户端使用与 UI 开发指导



作者

李国勇

日期

2024.3.1

修订记录

日期	内容	作者
2024.3.1	创建文档	李国勇
2024.3.22	增加内置组件	李国勇

摘要

至简网格是一款 HTTP 服务器，用于开发基于数据库的端云结合的服务程序，可以运行在资源极其有限的设备上，比如安卓手机、树莓派等，使得服务器可以尽量前移到生产端，可以运用于边缘计算、企业生产信息化、办公自动化等，

为了方便用户使用服务端的能力，必须要有与之配套的客户端。本文介绍至简网格客户端的安装使用与 UI 开发，UI 开发中，需要开发人员具备基本的 js、vue 的技术积累。

目录

至简网格客户端使用与 UI 开发指导	1
修订记录	2
摘要	3
1. 安装与使用	6
1.1. 安装	6
1.1.1. windows 客户端	6
1.1.2. android 客户端	6
1.2. 端侧设置	8
1.2.1. 个人帐号	9
1.2.2. 公司帐号	10
1.2.3. 登录	12
1.3. 应用市场	14
1.3.1. 应用列表	14
1.3.2. 应用详情	15
1.4. 打开应用	16
1.5. 公司帐号管理	18
1.5.1. 帐号管理	18
1.5.2. 服务授权	20
1.5.3. 群组管理	21
1.6. 个人帐号管理	22
2. UI 开发	23
2.1. 概述	23
2.1.1. 服务目录结构	23
2.1.2. 服务中的网络请求	24
2.1.3. 服务起始页样例	24
2.1.4. 交互页面样例	30
2.2. Http 请求	32
2.2.1. request	32
2.2.1.1. 请求参数	33
2.2.1.2. 响应处理	34
2.2.2. download	36
2.2.3. getExternal	37
2.3. 内置函数	37
2.3.1. 函数列表	38
2.3.2. 扫码案例	43
2.3.3. 生成二维码	44
2.4. 内置组件	45
2.4.1. addr_dialog	46

2.4.2. addr_input	46
2.4.3. addr_select	47
2.4.4. alert_dialog	47
2.4.5. confirm_dialog	49
2.4.6. date_input	49
2.4.7. grp_selector	50
2.4.8. user_selector	51
2.4.9. login_dialog	51
2.4.10. service_selector	52
2.4.11. process_dialog	52
2.5. 报表开发	54
3. 服务举例	55
3.1. 极简 CRM	55
3.2. 极简会员	56
3.3. ClassHour	58

1. 安装与使用

1.1. 安装

在做端侧 UI 开发之前，首先需要安装客户端。当前支持 windows、android 两种客户端，两种界面很接近，操作也一样。但是有部分安卓中的功能，在 windows 客户端中没有，比如扫码、横竖屏等。

1.1.1. windows 客户端

从网站下载后，双击安装即可。

windows 客户端需依赖系统的 Edge 浏览器。此浏览器在 windows7 及以上版本都可以安装，windows 10、11 中已默认安装。如果未安装，则需要手动下载安装 [Edge 浏览器](#)。

1.1.2. android 客户端

至简网格客户端至少需要在安卓 8.0 中安装（2017 年 8 月 22 日发布）。

至简网格客户端没有上传到各大应用市场，需要使用安卓手机的浏览器扫码下载，然后再安装。

有些情况下，浏览器没有安装应用的权限，会提示确认是否赋予浏览器安装应用的权限，此时必须给予授权。



是否允许「浏览器」安装应用？

您的设备和个人数据更容易受到未知应用的攻击。安装来自该来源的应用即表示，您同意对因使用这些应用可能导致的设备损坏或数据丢失问题承担责任。



浏览器
安装来源

记住我的选择

允许

禁止

安装时，会有安全提醒，请选中“我已充分了解风险，并继续安装”。因为安卓手机品牌众多、版本众多，提示不尽相同，总之需要容许安装才可以。

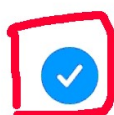


至简网格

版本: 1.0 | 大小:13.4MB | 权限 >

i 该应用未经过小米安全审核, 请您仔细甄别, 谨慎安装!

如果您是开发者可[联系我们](#)上架该应用及获取服务支持



已了解此安装包未经检测, 可能存在风险

查找类似应用

继续安装

取消安装

通过以上步骤, 安装就完成了, 与普通 App 安装是一样的。

1.2. 端侧设置

设置中可以进行个人帐号注册与登录, 或者公司帐号的登录, 登录完成后, 可以设置个人信息。

帐号分为两类: 个人帐号、公司帐号, 用户可以登录一个个人账号, 同时可以登录一个或多个公司的公司账号。

在服务中使用账号时，如果当前打开的服务是一个公司服务，则自动使用当前选中的公司的账号；如果是一个个人服务，无论当前选中的是哪个公司的账号，都使用个人账号。

1.2.1. 个人帐号

在使用一些个人服务时，比如密码箱、专注力等服务，它们不属于任何一家公司服务，所以必须使用个人帐号。



个人帐号需要自己注册，当前只支持自建帐号，没有使用 QQ、微信等第三方帐号。

注册

账号

密码

确认密码

验证码

确定

取消

1.2.2. 公司帐号

公司帐号及其初始密码是公司的超级管理员创建的, 创建方法请参照 [用户服务](#)中的描述。

在登录公司帐号之前, 需要先点击左上角的图标添加公司, 待添加的公司必须已经注册过。



公司 ID：公司或组织注册时获得的 ID；

接入码：接入密码，需注意，它相当于 WIFI 密码，不可随意透露给公司外不相关的人员。

增加公司

公司ID

接入码

确定

取消

如果工作环境不能访问外网，点击“确定”后，端侧无法知道连接哪个服务器，这时会要求输入“内网地址”。

增加公司

公司ID
1084

接入码
111111

内网地址
192.168.0.1:8523

确定

取消

公司 ID、接入码、内网地址，在[服务端设置](#) 都可以找到，请联系服务器维护人员获得这些信息。

1.2.3. 登录

个人帐号与公司帐号的登录与退出是一样的。

因为公司帐号是超级管理员添加的，系统默认的公司超级管理员帐号是 admin，密码是 123456。强烈建议在第一次登录时修改 admin 密码。

[←](#) 账号信息 -

昵称

邮箱

手机号

修改密码

>

系统支持同时登录一个个人帐号与多个公司帐号，登录时需要点击左上角的图标，选择个人或者某个公司。



然后再点击右上角的登录，在弹出窗口中输入帐号、密码，点击“登录”即可。

登录

账号

密码



登录

取消

在使用服务时，如果是个人服务，则自动使用个人帐号身份。

如果是公司级服务，则使用当前选中的公司帐号，在左上角可以切换当前的公司。如果当前帐号是个人帐号，且有多多个公司帐号，则无法访问，因为个人帐号无法在公司级服务中使用，也无法自动从多个公司帐号中自动挑选一个，所以只能失败。

此特性可以满足一个员工同时为多家企业服务的场景。

1.3. 应用市场








应用分成两类，一类是公司级应用，一类是个人应用。如 CRM、会员等都是公司级应用；密码箱、专注力等是个人应用。


公司级应用需要单独部署公司服务器才可以使用，服务可以部署在一部安卓手机上，也可以部署在服务器上，或者部署在云端；个人应用为个人服务器，为生活提供便利，比如记密码、练习专注力、记单词、算账、杂记等，这些功能不需要部署服务器，安装即可使用。


1.3.1. 应用列表


点击应用就可以进入详情界面，进行安装或卸载。


应用


	服务调测助手/console flyinmind@zhijian.net.cn
	服务器UI/serverui flyinmind@zhijian.net.cn
	UI基础库/assets flyinmind@zhijian.net.cn
	极简会员/member flyinmind@zhijian.net.cn
	端侧UI/clientui flyinmind@zhijian.net.cn
	企业用户服务/user flyinmind@zhijian.net.cn
	极简CRM/crm flyinmind@zhijian.net.cn


公司服务


个人服务


首页


应用


设置

1.3.2. 应用详情

在详情中有关于应用的详细介绍。如果没有安装，则下方显示“安装”按钮，否则显示“卸载”按钮；当检测到新版本时，会多一个“升级”按钮。



满足小微企业客户关系管理所需，实现客户、联系人、订单、服务、回款管理，支持按 workflow 方式推动销售、服务、回款等工作。

可自定义 workflow 步骤，每个步骤可以单个负责人签字，也可以多个参与人共同决策，方便企业实现 workflow 规范化。

实时数据统计，并呈现在主界面中，普通成员可以查看一月内的报表，企业主与财务可以查看年度与月度报表。

精准的数据授权，只将数据授权给参与 workflow 的成员，其他成员通过数据分享查看数据，数据分享可设置期限。

具备客户、联系人、订单管理功能



注：以上图例只作为样例，并不代表实际情况。

1.4. 打开应用

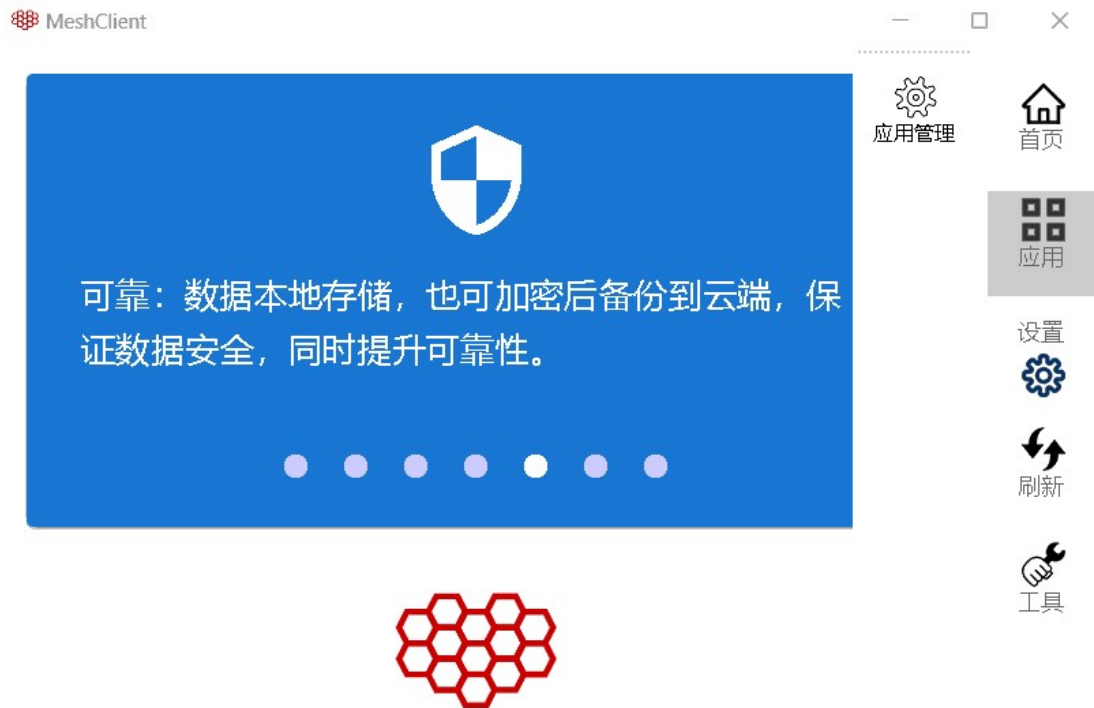
在应用主界面的最下方，点击“应用”按钮，出现一个应用栏，里面显示了所有已安装的应用，比如 CRM、会员等。 点击一个应用，就可以进入应用的主界面。

每种应用的主界面不同，下图为 CRM 的主界面。



打开一个应用，使用一段时间后，再次点击“应用”按钮，可以切换到其他应用。如果退出程序，下次打开程序时，会自动进入上次打开程序时进入的应用。

Windows 版本的客户端与此类似，应用栏显示在屏幕的右侧。



1.5. 公司帐号管理

公司级服务大多依赖公司帐号服务，它记录了公司内所有员工的帐号、密码、授权等信息。主要功能有员工帐号管理、服务授权与群组管理。系统实现中，强依赖帐号管理与服务授权。群组管理在每个业务中根据需要使用，在 CRM、会员服务中都未使用群组功能。

1.5.1. 帐号管理

公司级员工帐号只有超级管理员可以增加、删除，在界面的右上角有“服务授权”与“群组管理”两个图标。

帐号管理



帐号

昵称

电话

admin

超级管理员

lgy

lgy

lhc

lhc



点击某个员工帐号，进入帐号详情界面，在此可以修改员工的邮箱、电话号码等信息。忘记密码时，可以在此重置密码。

注意：重置的密码是随机生成的 6 个字符，在使用此密码登录后，需及时更改。

← 员工管理

帐号

lgy

昵称

lgy

电话

邮箱

登录时间

1970/1/1 08:00:00

密码

QbOK7F

激活/禁用

重置密码

任职部门

部门名称

#/开发部

职位

开发部经理

服务权限

服务名	角色	扩展权限
极简会员	admin	公网访问

在此还可以禁用帐号，禁用的帐号无法登录系统，也可以重新启用。在此还可以查看帐号从属于哪些组织、在服务中拥有的授权。 如果需要调整，可以删除从属关系与授权。

1.5.2. 服务授权

员工拥有帐号后，还不能在任何服务中进行操作，只有经过超级管理员授权后，才可以使用相应的服务。 授权时指定的角色，需要服务的开发人员在角色定义接口中定义。

帐号	昵称	角色	扩展权限
admin	超级管理员	企业主	公网访问
lgy	lgy	企业主	公网访问
lhc	lhc	服务	公网访问

×

🔍

+

授权时可以指定是否可以在公网访问内网的服务，如果未授权公网访问，则只能在内网访问服务。

1.5.3. 群组管理

管理公司的各类群组，最为常用的是部门划分。作为一个为中小企业服务的开发框架，我们推荐使用弱矩阵方式进行管理， 有明确的组织结构，同时按项目方式推动公司的运作，还可以存在一些虚拟“群组”作为补充。



点击组织，进入下一层组织，比如进入“部门”，可以配置下一层组织结构，也可以在组织中增删成员。



1.6. 个人帐号管理

个人帐号不可以用在公司服务中，它需要用户自己注册产生。

2. UI 开发

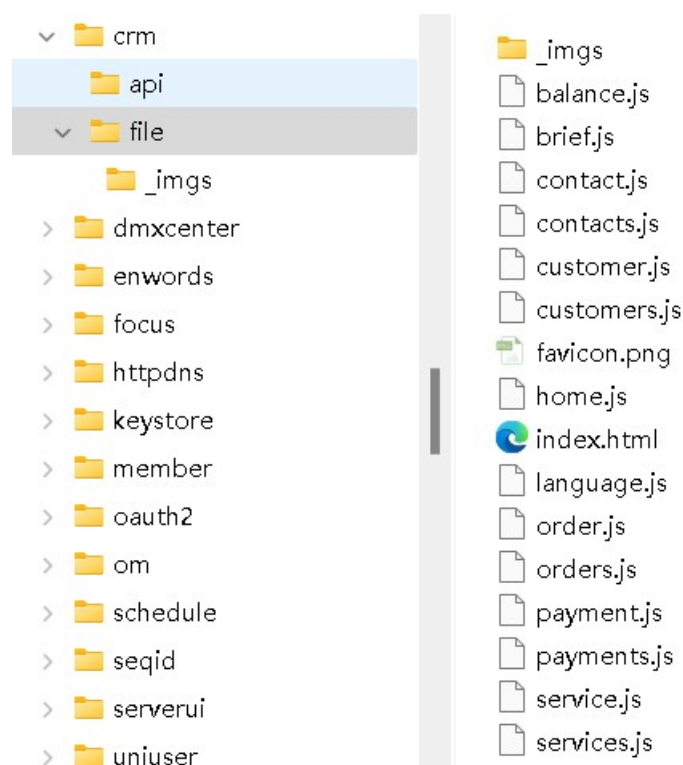
2.1. 概述

至简网格客户端是一个轻应用开发的平台，端侧的开发，本质就是使用 html+js+vue+quasar 开发网页，如果需要输出报表，可以使用 echarts。至简网格在客户端中提供了一些原生接口，使得与至简网格的服务端对接变得非常方便。

至简网格并没有限制使用什么类型的 js 前端框架，但是推荐使用 vue+quasar，并且内置了 vue 与 quasar，服务开发时，可以直接引用它们。

2.1.1. 服务目录结构

端侧 UI 开发是放在服务的 file 子目录中的，比如下图是 CRM 服务的结构，其中 file 目录就是端侧 UI 实现。服务在启动时会自动生成一个 app.cfg 文件，并将它们一起打包成一个 zip 文件。客户端在安装、升级时，用的就是这个 zip 文件。



端侧 ui 在安装时，下载此 zip 文件，并且解压到本地目录，然后加载其中的 index.html 文件，显示服务的 UI。所以，如果服务需要在客户端显示内容，则，file 目录下必须有一个 index.html 文件。在 index.html 文件中，完成 vue、quasar 的初始化，如果使用了报表，还需要做 echarts 的加载工作。

至简网格服务客户端开发本质是网页开发，所以在 index.html 中，完全与普通网页开发是一致的。

2.1.2. 服务中的网络请求

在端侧开发中，对网络的请求不可以使用任何一种 ajax 框架，比如 axios、jquery 等，因为为了实现至简网格服务端的灵活部署，内置 webview 时，禁用了它的网络访问能力。如果需要通过网络请求，必须使用原生类 Http 的函数，request 是用来请求至简网格服务端接口的，download 是用来下载文件的，getExternal 是用来访问非至简网格服务的。

2.1.3. 服务起始页样例

```
<!DOCTYPE html><-- DOCTYPE 不可省略 -->
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8" />
```

```
<meta name="content-type" content="text/html; charset=utf-8" />
```



```
<meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
```

```
<-- 避免加载 favicon -->
```

```
<link rel="icon" href="data:image/ico;base64,aWNv">
```

```
<-- 如果使用 quasar, 则必须加载以下两个 css -->
```

```
<link href="/assets/v3/quasar_font.css" rel="stylesheet" type="text/css">
```

```
<link href="/assets/v3/quasar.css" rel="stylesheet" type="text/css">
```

```
<title>CRM</title>
```

```
</head>
```

```
<body style="overflow:hidden">
```

```
<-- app.mount('#app')用到 div 的 id, v-cloak 使得 vue 在没有得到值之前,
不显示变量名 -->
```

```
<div id="app" v-cloak>
```

```
<-- router-view, 应用的页面都在此加载 -->
```

```
<router-view></router-view>
```

```
</div>
```

```
</body>
```

<-- 必须加载的 js -->

<script src="/assets/v3/vue.js"></script>

<script src="/assets/v3/vue-router.js"></script>

<script src="/assets/v3/quasar.js"></script>

<script src="/assets/v3/osadapter.js"></script>

<-- 根据需要选择是否加载 -->

<script src="/assets/v3/echarts.js"></script>

<script src="/assets/v3/qrcode.js"></script>

<script type="module">

import Language from "./language.js"

//延迟加载, 引入公共组件, 这样 import 可以兼容安卓 7

//如果不考虑兼容 android 7, 则可以按需加载, 在 VueRouter.createRouter
的 routes 中直接 import

import DateInput from "/assets/v3/components/date_input.js"

import UserSelector from "/assets/v3/components/user_selector.js"

import AlertDialog from "/assets/v3/components/alert_dialog.js"

import ConfirmDialog from "/assets/v3/components/confirm_dialog.js"

```
const l=(typeof os)=='undefined' ? navigator.language : os.language();
```

```
const tags = l.indexOf("zh") == 0 ? Language.cn : Language.en;
```

```
//router 定义
```

```
const router = VueRouter.createRouter({"history":
```

```
VueRouter.createMemoryHistory(),
```

```
routes:[
```

```
//定义 router, 安卓 7 不支持按需加载 import
```

```
{path:'/home', component:()=>import('./home.js')},
```

```
...
```

```
]]);
```

```
//service 定义
```

```
const service = {
```

```
go_back() { //返回, 在页面中通过 service.go_back 调用
```

```
router.back();
```

```
},
```

```
jumpTo(url) {
```

```
router.push(url);
```

```
}
```

```
};
```

```
//app 主体定义
```

```
const app = Vue.createApp({
```

```
  provide:{tags:tags, service:service, icons:icons},
```

```
  created(){
```

```
    service.baseInfo();
```

```
    this.$router.push('/home').catch(err => {err}) //避免报
```

NavigationDuplicated, 此错误不影响功能

```
  },
```

```
  mounted() {
```

```
    window.sys_go_back = this.sysGoBack;//给 webview 调用
```

```
  },
```

```
  methods:{
```

```
    sysGoBack() {
```

```
      //声明全局函数，在 webview 中调用，
```

```
      //实现按回退按钮回退到历史页面，如果无历史，则退出 activity 或  
      应用
```

```
      if(this.$router.currentRoute.value.path === "/home") {
```

```
        return false;
    }
    this.$router.back();
    return true;
}
}

});
```

app.use(Quasar);//必须： 设置 Quasar

app.use(router);//必须： 设置 route

//注册全局组件， 按需选择， 另外还有 address 控件

```
app.component('component-user-selector', UserSelector);
```

```
app.component('component-alert-dialog', AlertDialog);
```

```
app.component('component-confirm-dialog', ConfirmDialog);
```

```
app.component('component-date-input', DateInput);
```

app.mount('#app');//必须： 启动 APP

```
</script>
```

```
</html>
```

2.1.4. 交互页面样例

以下为一个删减来所有细节的首页实现，具体的实现可以参照已在 gitee、csdn、github 开源的服务实现。

```
export default {  
  
  inject:['service', 'tags'], //引用全局对象，页面中可以像使用 data 中变量一样使用  
  
  data(){return{  
  
    name:"test"//数据定义，在页面中{{xxx}}括起的部分，在此都必须定义  
  
  }},  
  
  created(){  
  
    this.init(); //初始化加载，在 mounted  
  
  },  
  
  methods:{  
  
    init(){  
  
      //处理逻辑  
  
    }  
  
  },  
}
```

//注意""不是单引号，是键盘左上角的反单引号(backquote)

template:`

<q-layout view="lHh lpr lFf" container style="height:100vh">

<q-header elevated>

<!-- 非必须，页眉内容 -->

</q-header>

<q-footer elevated>

<!-- 非必须，页脚内容 -->

</q-footer>

<q-page-container>

<q-page class="q-pa-md">

<!-- 中间内容 -->

{{name}}

</q-page>

</q-page-container>

</q-layout>

`//用反单引号结尾

```
}
```

2.2. Http 请求

在内置浏览器中禁用了所有网络访问能力，即使使用 axios 也不能访问，必须通过 request(opts, service)、download(opts, service)、getExternal(url)三个接口实现。

2.2.1. request

request 函数中不可以传入完整的 url，只需传入服务名、服务接口，request 内部根据网络分布情况，会选择合适的服务器，自动拼接出完整的请求 url。

```
request({method:"POST", url:"/api/customer/create", data:dta},
"crm").then(resp => {

    if(resp.code != RetCode.OK) {

        this.$refs.errMsg.showErr(resp.code, resp.info);

        return;

    }

    //如果有响应数据，在这里处理 resp.data

})
```


2.2.1.1. 请求参数

request、download 的 service 参数为被请求的服务名称, opts 为请求选项, 包括 method、url、data、private 四项, file_name 是 download 特有的。

- 1) method: 支持 GET/POST/PUT/DELETE 方法, 如果是 GET/DELETE;
- 2) url: 请求 URL, 可以在"?"后面带参数;
- 3) data: 请求参数, 必须是 json 对象, 如果 method 是 GET/DELETE, 则无需传 opts.data 参数;
- 4) isCloud: 表示无论当前选中的是哪个公司, 请求都会发到根公司的云上服务中;
- 5) private: 可以不传递, 默认为 true, 表示需要做用户鉴权, 如果访问 public 接口, 将 private 设为 false 即可; 如果客户端已登录, 则会自动使用用户 token 获取服务 token, 然后用服务 token 访问服务接口; 如果用户未登录, 则操作失败, 建议在收到 NO_RIGHT 错误码时, 跳出提醒登录的窗口。
- 6) timeout: 单位毫秒, 不设置或设成小于或等于 8000 的值, 则使用默认的 HttpClient, 超时为 8 秒, 否则创建一个临时 HttpClient, 使用此 timeout 值; 不推荐使用此设置, 除非万不得已, 比如安装服务、备份数据等请求, 因为每次都会新建一个 HttpClient, 既耗时又耗资源

2.2.1.2. 响应处理

Http.request 是用来请求接口的，返回都是 json 格式。如果响应中需要携带数据内容，必须放在 data 字段中，没有响应数据内容，data 可以省略。

响应处理中，首先判断 code 是否为 RetCode.OK，只有 OK 是正常处理，其他错误码则根据情况处理，比如 EXISTS，在某些情况下是正常的响应码，这需要业务实现时判断。响应数据在 resp.data 中，data 是一个 Map 对象。

2.2.1.2.1. 响应整体结构

所有响应的顶层结构都是一样的，包括返回码 code、信息 info，如果是查询类的请求，会包括数据 data 字段，每个查询类接口的 data 都不相同。

```
{  
  code:0,  
  info:"Success",  
  data:{  
    a:1,  
    b:"xxx",  
    c:{...},  
    d:[...]  
  }  
}
```

2.2.1.2.2. 返回码

响应体中的 code 为返回码, 客户端的返回码与服务端完全一致。如果无错误则为 OK(0),

返回码在 js 脚本中用 RetCode.xx 直接引用, code 定义如下:

名称	值	含义
OK	0	成功
DEPRECATED	1	接口即将废弃
INTERNAL_ERROR	100	内部错误
INVALID_TOKEN	102	无效 token
EMPTY_BODY	103	请求体错误, 用在 POST 请求中
DB_ERROR	104	数据库错误
INVALID_SESSION	105	无效的 session
SERVICE_NOT_FOUND	106	服务不存在
TOO_BUSY	107	系统太忙
SYSTEM_TIMEOUT	108	系统超时
NOT_SUPPORTED_FUNCTION	109	API 存在, 但是所需的功能不支持
API_NOTFOUND	110	API 不存在
NO_RIGHT	111	无权调用
NO_NODE	112	找不到可用的节点提供服务
INVALID_NODE	113	无效的节点, 比如数据库分片的情况下, 请求发到错误的 webdb 实例上
THIRD_PARTY_ERR	114	调用第三方服务失败
UNKNOWN_ERROR	150	未知错误
EXISTS	2000	已经存在
NOT_EXISTS	2001	不存在
API_ERROR	3000	API 错误
WRONG_JSON_FORMAT	3001	JSON 体解析失败

2.2.2. download

request 用来请求服务端接口，返回都是 json 格式的，download 是用来下载文件的，返回内容是二进制格式。

opts 中，除了支持 request 的所有选项外，还需要增加一个 file_name 参数，用来指定被下载的文件在存到本地时的名称，如果不指定，就用 url 中的 uri 作为文件名。

如果不是通过接口访问获得文件，还可以增加一个 file 参数，设为 true 时，使用的 url 中将不会携带 api。其他参数与 request 完全相同。

```
download({file_name: fn,/*attatchment*/ url:'/downloadlog?n=' +
encodeURIComponent(f)}, "crm").then(resp => {

  Console.debug(JSON.stringify(resp));

  if(resp.code == RetCode.OK) {

    this.dlList.splice(0, 0, {file:resp.data.saveAs, size:resp.data.size,
bg:'#00000000'})

  } else {

    this.dlList.splice(0, 0, {file:f, size:0, bg:'#884444'})

  }
})
```

```
this.dlDlg=true;
```

```
});
```

2.2.3. getExternal

用于请求访问非至简网格服务的 http 资源, 它的响应内容全部当作普通文本处理, 服务在处理响应结果时, 必须自己理解它的格式定义。

```
getExternal({url:'https://domain/pathsources...',headers:{...}}).then(txt=> {
```

```
    var resp = JSON.parse(txt);
```

```
    if(resp.code != 0) {
```

```
        ...
```

```
    }
```

```
});
```

2.3. 内置函数

在端侧, 除了内置浏览器本身 js 支持的能力外, 至简网格还提供了一些内置的函数, 随着系统的完善, 会有更多的内置能力通过 js 函数方式开放出来。

2.3.1. 函数列表

在客户端 UI 编程中，有些功能 js 不能或不易实现，比如加解密、文件读写等，这些功能在平台中已经通过原生的方式实现，并提供了相应的 js 接口。

函数	备注
公共函数	
request(opts, service)	向 service 指定的服务发起请求，详细内容请参考 Http 中的描述
download(opts, service)	下载文件，请求参数与 request 完全相同，响应的内容是一个文件，并且存到端侧指定的目录中，此目录是端侧工作目录下的 download 子目录
getExternal(opts)	请求其他网站的 URL，opts 与 request 中定义相同，只是 URL 需要传递完整的值，比如 https://www.gitee.com/xxxx
Platform 类中的函数	
height()	以像素为单位的浏览器可见区域高度，如果使用了 quasar，建议使用 \$q.screen.height
width()	以像素为单位的浏览器可见区域宽度，如果使用了 quasar，建议使用 \$q.screen.width
portrait()	变成竖屏
landscape()	变成横屏
undefineOrientation()	不定义横竖屏，恢复成原来的样子
language()	当前系统选择的语言，比如 zh-CN
isSupported(feature)	判断一个功能是否被支持，当前只有 scancode(识别二维码、条形码)、orientation(改变屏幕显示方向)可选
showTools()	显示工具栏
hideTools()	隐藏工具栏，注意，此功能在 windows 客户端不起作用

函数	备注
scanCode(jsCbld)	扫描二维码、条码, jsCbld 请参照 扫码案例 , 通过 __regsiterCallback(callback)注册回调时获得
Console 类中的函数 日志输出到端侧的日志文件中, 而不是浏览器的控制台上; 输出到控制台请使用小写的 console	
debug(s)	输出 debug 级别的日志
info(s)	输出 info 级别的日志
warn(s)	输出 warn 级别的日志
error(s)	输出 error 级别的日志
File 类中的函数	
init()	初始化, 使用之前必须初始化, 为服务创建私有目录, 所以以下函数中的 fileName 都不必提供完整路径, 会自动加上服务对应的根目录
append(fileName, s)	向 fileName 指定的文件末尾追加内容,
write(fileName,s)	向 fileName 指定的文件写入内容, 如果已存在该文件, 则会覆盖掉。服务只可以读写服务根目录或其子目录下的文件。
read(fileName)	从 fileName 指定的文件读出内容, 返回一个字符串。此功能不宜用于读取超大文件
JStr 类中的函数	
uuid()	产生 uuid 字符串, 使用 base64 编码
replaceChars(str,ch,replaceWith)	在 str 中寻找 ch, 并替换成 replaceWith
chkIdNo(s)	判断是否为合法的身份证号码
chkCreditCode(s)	判断是否为合法的统一信用码
base64CharCode(c)	返回一个字符的 base64 编码, c 必须是"a-z,A-Z,0-9,_,-"中的一个

函数	备注
base64Char(v)	将 0-63 数值，转为"a-z,A-Z,0-9,_,-"中的一个
intHash(s)	返回字符串的整型 hash 值
longHash(s)	返回字符串的长整型 hash 值
absHash(s)	返回字符串的长整型 hash 的绝对值
isLanIP(v)	判断是否为局域网 IP，支持 IPv4 与 IPv6 判断
isIPv4(v)	是否为一个合法的 IPv4 地址
isIPv6(v)	是否为一个合法的 IPv6 地址
Secure 类中的函数	
pbkdf2(pwd, iterationCount)	使用 pbkdf2 算法，将 pwd 迭代 iterationCount 次
pbkdf2Check(pwd, savedPwd)	检查输入的 pwd 与 savedPwd 是否一致，savedPwd 由 pbkdf2 函数生成
cbcEncrypt(plain, key, keyLen)	使用 AES-CBC 算法加密，plain 为明文，key 为密钥，keyLen 可以选择 16/24/32。IV 为随机产生，并记录在密文的前面 16 字节中。
cbcDecrypt(cipher, key, keyLen)	使用 AES-CBC 算法解密，cipher 为密文，其中包括了随机 IV
gcmEncrypt(plain, key, keyLen)	使用 AES-GCM 算法加密，plain 为明文，key 为密钥，keyLen 可以选择 16/24/32。IV 为随机产生，并记录在密文的前面 16 字节中。
gcmDecrypt(cipher, key, keyLen)	使用 AES-GCM 算法解密，cipher 为密文，其中包括了随机 IV
keyPair(pwd)	产生 ECC 密钥对，pwd 为加密密钥对的密钥；返回内容为一个字符串
publicKey(kp)	keyPair 产生密钥后，通过此函数导出公钥
privateKey(kp, pwd)	keyPair 产生密钥后，通过此函数导出私钥，因为私钥是加密的，所以必须提供密码
eccEncrypt(plain, pubKey)	使用 ECC 公钥加密，plain 为明文，pubKey 为 publicKey

函数	备注
pubKey)	从密钥对中导出的公钥；返回加密后的字符串密文
eccDecrypt(cipher, prvKey)	使用 ECC 私钥解密, cipher 为密文, prvKey 为 privateKey 从密钥对中导出的私钥（未加密，需注意安全）；返回解密后的字符串明文
keyPairEncrypt(kp, plain)	使用 ECC 密钥对中公钥加密 plain 字符串；返回加密后的密文字符串
keyPairDecrypt(kp, pwd, cipher)	使用 ECC 密钥对中私钥解密 cipher 字符串, pwd 为调用 keyPair 产生密钥对时的密码；返回解密后的明文字符串
saveItem(k, v)	使用系统根密钥加密存储服务信息；使用此函数时需注意，系统根密钥加密的内容只在当前系统可以解密，离开当前系统则无法解密
readItem(k)	使用系统根密钥解密存储的服务信息，如果不存在，则返回空字符串
md5(str)	使用 MD5 算法对 str 进行不可逆运算
sha1(str)	使用 SHA1 算法对 str 进行不可逆运算
sha256(s)	使用 SHA256 算法对 s1,s2,s3...进行不可逆运算，在它们之间会增加分隔符“-”
hmacSHA256(str)	使用 SHA256 算法对 str 进行不可逆运算。随机生成 16 字节 key，并记录在结果的前面
hmacSHA256Check(str, saved)	验证 str 与 saved 是否一致，saved 是 hmacSHA256 算法生成的
hmacSHA1(str, key)	使用 HMAC-SHA1 算法对 str 进行不可逆运算，key 可以是一个随机字符串
isPwdStrong(acc, pwd, min, max, charTypeNum, diffCharNum)	判断密码强度是否足够。 acc：帐号，用于判断密码是否与帐号接近 pwd：密码 min：最小长度 charTypeNum：不同字符的数量 diffCharNum：不同类型字符数量，0-9 a-z A-Z 其他，共四类

函数	备注
<p>Database 类中的函数</p> <p>虽然浏览器都内置了数据库实现，但是，浏览器内置数据库标准已废弃，考虑到未来的兼容性，所以提供此类。</p> <p>除了 open 函数，所有函数异步返回，结果的形式为：</p> <pre>{code:ressult_code,info:error_infomation,data:{...}}</pre> <pre>var db='test_db'; if(Database.open(db)>0) { alert(xxx);return;} Database.initialize(db,"create table if not exists testtab(...);create index if not exists ...").then(res=>{ if(res.code!=RetCode.OK){...,return;} ... }); Database.execute(db, "insert into testtab(...),values(...)").then(res=> if(res.code!=RetCode.OK){...,return;} if(res.data.lineNum>0){...} }); Database.queryMaps(db, "select c1,c2 from testtab where ...").then(res=> if(res.code!=RetCode.OK){...,return;} for(var row of res.data.rows) { Console.info(row.c1 + "," + row.c2);... } });</pre>	
open(db)	打开一个本地的数据库，此处以及后面函数中出现的 db 参数都是指是数据库名称；返回 0 表示失败，1 表示已打开过了，2 表示新建成功
initialize(db,sqls)	初始化数据库，sqls 是一个字符串，包括一条或多条建表

函数	备注
	语句，多条时，用分号分隔
execute(db,sql)	执行一条增、删、改类的 sql 语句；返回 data:{lineNum:xxx}, lineNum 为受影响的行数
executes(db,sqls)	执行一条或多条增、删、改类的 sql 语句，多条时，用分号分隔；返回内容与 execute 相同
queryArrays(db,sql)	执行一条查询 sql，结果以数组方式返回，不包括列名；比如，data:{rows:[[a,b,c],[d,e,f]...]}, 每行记录的列顺序与 sql 中的列顺序一致
queryMaps(db,sql)	执行一条查询 sql，结果以对象数组方式返回，包括列名，比如，data[{c1:a,c2:b,c3:c},{c1:d,c2:e,c3:f}...]
queryMap(db,sql)	执行一条查询 sql，结果以对象方式返回，比如，data:{c1:a,c2:b,c3:c}

2.3.2. 扫码案例

```

var jsCbId=__regsiteCallback(resp => {

    if(resp.code!=RetCode.OK) {

        this.$refs.alertDlg.showError(resp.code, resp.info);

        return;

    }

    var data = JSON.parse(resp.data.value);

    .....

});

```

```
Platform.scanCode(jsCbId);
```

2.3.3. 生成二维码

如果需要生成二维码，必须在起始页 index.html 中包含 qrcode，已内置到客户端版本中。

```
<script src="/assets/v3/qrcode.js"></script>
```

```
showQrCode() {
```

```
    var txt = JSON.stringify({data...}); //待生成的内容必须为一个字符串
```

```
    new QRCode(this.$refs.qrCodeArea, {
```

```
        text: txt,
```

```
        width: width, //必须像素为单位
```

```
        height: width,
```

```
        colorDark: '#000000',
```

```
        colorLight: '#ffffff',
```

```
        correctLevel: QRCode.CorrectLevel.H
```

```
    });
```

```
}
```

如果是在一个 dialog 中显示，需要在 dialog 的 show 事件中处理显示二维码的工作，否则无法显示。

```
<q-dialog v-model="qrCodeDlg" @show="showQrCode">  
  
<q-card :style="{min-width: width+'px'}" bordered> <q-card-section>  
  
<div ref="qrCodeArea" :style="{width:width+'px',height:width+'px'}"> </div>  
  
</q-card-section> </q-card>  
  
</q-dialog>
```

2.4. 内置组件

端侧内置了一些常用组件，有地址选择、alert 对话框、确认对话框等。

首先，在 UI 实现时先 import 它们，比如：

```
import DateInput from "/assets/v3/components/date_input.js"  
import UserSelector from "/assets/v3/components/user_selector.js"  
import AlertDialog from "/assets/v3/components/alert_dialog.js"  
import ConfirmDialog from "/assets/v3/components/confirm_dialog.js"
```

其次，在 app.mount('#app')之前将这些组件都注册进去

```
app.component('component-user-selector', UserSelector);  
app.component('component-alert-dialog', AlertDialog);  
app.component('component-confirm-dialog', ConfirmDialog);  
app.component('component-date-input', DateInput);
```

最后，在使用时当作普通的标签使用

```
<component-user-selector :label="tags.signers" :accounts="newCust.nextSigners"></component-user-selector>
```

2.4.1. addr_dialog

以一个对话框的形式，提供三级（省、市、县/区）的地址选择，这些数据都是从云上查询接口得到的。它支持以下属性：

属性名称	备注
country	国家码，字符串类型，默认为 156（中国）
label	标签，字符串类型，显示在地址选择框上方，默认为空
ok	确定按钮的标签，字符串类型，默认为"确定"，点击此按钮，会触发 confirm:modelValue
cancel	取消按钮的标签，字符串类型，默认为"取消"

2.4.2. addr_input

以一个输入框的形式，提供多级的地址选择，输入任意关键字，会模糊搜索一个完整的地址。

此组件在安卓 7 中无法正常显示，并且，oppo 的安卓 8 中也会无法显示。如果考虑兼容，请使用 addr_dialog。

支持以下属性：

属性名称	备注
------	----

属性名称	备注
label	标签，字符串类型，显示在地址选择框上方，默认为空

2.4.3. addr_select

用三个输入框选择地址，每一级选择后，后面的级会自动更新。支持以下属性：

属性名称	备注
label	标签，字符串类型，显示在地址选择框上方，默认为空

2.4.4. alert_dialog

信息提示对话框，此对话框在点击空白处时会消失。支持以下方法：

方法名	备注
show(msg)	显示 msg
showErr(code,info)	显示错误码及其对应的错误信息，如果 errMsgs 中包括了错误码信息，则显示 errMsgs，否则显示默认的错误信息

支持以下属性：

属性名称	备注
errMsgs	错误码与错误信息的对应关系，对象类型，比如{4000:"参数错误"}
title	标题，字符串类型，默认为“警告”
close	关闭按钮的标签，字符串类型，默认为“关闭”

使用时先引入组件，注意要增加 ref 属性，比如 errMsg。

```
<component-alert-dialog ref="errMsg"> </component-alert-dialog>
```

需要显示错误信息时，引用 ref，调用 showErr 函数：

```
this.$refs.errMsg.showErr(errCode, errInfo)
```

或者直接显示信息：

```
this.$refs.errMsg.show(info)
```

如果是在其他组件中引用本组件，可以参照以下方法，首先引入本组件：

```
import AlertDialog from "/assets/v3/components/alert_dialog.js"
```

然后，在组件中注册组件：

```
export default {  
  
  ...  
  
  components:{  
  
    "alert-dialog":AlertDialog  
  
  },  
  
  ...  
  
}
```

并在 template 中申明组件：


```
<alert-dialog :title="failToCall" :close="close"
ref="errMsg"></alert-dialog>
```

最后，就可以调用组件的函数了：

```
this.$refs.errMsg.showError(errCode, errInfo);
```

2.4.5. confirm_dialog

确认对话框，是否方法与 alert_dialog 类似。支持以下方法：

方法名	备注
show(msg,callback)	显示 msg，并设置点击确认时的回调函数

支持以下属性：

属性名称	备注
title	标题，字符串类型，默认为“警告”
ok	确定按钮的标签，字符串类型，默认为“确定”，点击时会调用回调函数
close	关闭按钮的标签，字符串类型，默认为“关闭”

2.4.6. date_input

日期输入框。quasar 本身的日期组件已经很棒，提供此组件，是用于提供一些默认属性，降低代码量。支持以下属性：

属性名称	备注
label	输入框的标题，字符串类型
dateFormat	日期的显示格式，字符串类型，默认为“YYYY/MM/DD”
min	最小的日期，字符串类型，格式需要与 dateFormat 一致
max	最大的日期，字符串类型，格式需要与 dateFormat 一致
weekDays	从星期天到星期六，每天的名称，字符串数组类型，默认为["日","一","二","三","四","五","六"]
months	从 1 月到 12 月，每月的名称，字符串数组类型，默认为["一月","二月",...]
close	关闭按钮的标签，字符串类型，默认为“关闭”

2.4.7. grp_selector

群组选择输入框。调用企业用户服务中的群组接口，选择一个群组，支持对群组名称模糊搜索，并列出生搜索结果供选择。支持以下属性：

属性名称	备注
label	输入框的标题，字符串类型
useid	是否返回群组 id，布尔类型，默认为 false，返回群组名称，否则返回群组 id
grp	群组，字符串类型

2.4.8. user_selector

用户选择输入框。调用企业用户服务中接口获得帐号信息，选择一个帐号。

输入帐号、电话号码的部分或全部对帐号进行模糊搜索，并列出现搜索结果供选择。

支持以下属性：

属性名称	备注
label	输入框的标题，字符串类型
useid	是否返回群组 id，布尔类型，默认为 false，返回帐号，否则返回帐号 id
accounts	选中的帐号或帐号 id，数组类型，如果是单选，则只有一个成员
multi	是否多选，布尔类型，默认为 true，表示可选择多个帐号，在 accounts 中返回
service	限定的服务，字符串类型，如果不为空，则只返回在这个服务中获得授权的帐号，默认为空，表示不限制
roles	限定的角色，字符串数组类型，必须与 service 属性一起设置。如果不为空，则只返回在指定服务中获得相应角色的帐号，默认为空，表示不限制

2.4.9. login_dialog

用户登录对话框。输入帐号、密码，调用企业用户服务中接口进行用户登录。

支持以下属性：

属性名称	备注
label	对话框的标题，字符串类型，默认为“登录”
accType	帐号类型，字符串类型，默认为"N"，表示为普通帐号，公司帐号时只能为 N
account	帐号，字符串类型
pwd	密码，字符串类型

cancel	取消登录按钮标签，字符串类型，默认为“取消”
close	关闭按钮标签，字符串类型，默认为“关闭”，此按钮在登录失败时出现在错误提示框中
failToCall	错误提示信息，字符串类型，默认为“关闭”，此提示在登录失败时出现在错误提示框中

2.4.10. service_selector

服务选择输入框。输入服务名或描述信息模糊搜索服务列表供选择。支持以下属性：

属性名称	备注
label	对话框的标题，字符串类型，默认为“登录”
services	选中的服务列表，字符串数组类型
type	服务类型，字符串类型，默认为“enterprise”，表示选中公司类服务，如果为 personal，表示选择个人服务
useid	是否返回服务 id，布尔类型，默认为 false，返回服务名，否则返回服务 id
multi	是否多选，布尔类型，默认为 true，表示可选择多个服务，在 services 中返回，否则只返回一个服务

2.4.11. process_dialog

进度提示框，用于显示一个环状进度条，此进度条并不表示当前确定的进度，只是一个表示任务仍在继续的状态。它支持以下方法：

方法名	备注
-----	----

方法名	备注
show(title, info, icon, action, actionDone)	title: 对话框的标题 info: 进度提示信息 icon: 圆形进度条中间的图标 action: 点击确定按钮时需要执行的任务，必须为异步函数，传入参数为进度对话框本身，可以调用它的函数，比如 setInfo。action 可以为空 actionDone: 任务完成时的回调，会传递两个参数，第一个为进度对话框本身，第二个为 action 执行之后的返回值。可以为空
setInfo(info)	info 参数为进度提示信息，可以在 action、actionDone 中调用，显示与进度有关的信息

支持以下属性：

属性名称	备注
width	对话框的宽度，单位可以是 px、vw、vh、em、rem 等，字符串类型，默认为"80vw"
ok	确定按钮的标签，字符串类型，默认为"执行"，点击时会调用 action 函数
close	关闭按钮的标签，字符串类型，默认为"关闭"

使用时先引入组件，注意要增加 ref 属性，比如 procDlg。

```
<component-process-dialog
  ref="procDlg"> </component-process-dialog>
```

需要显示进度条时，引用 ref，调用 show 函数：

```
this.$refs.procDlg.show('数据备份', '确定要执行备份吗？',
  'cloud_download',
```

```
(dlg)=> {  
  
  dlg.setInfo("");  
  return this.service.command({cmd:"restore"}, 100000); //必须为异步函数  
  
},  
(dlg,resp)=> {  
  if(resp.code!=RetCode.OK) {  
  
    dlg.setInfo(formatErr(resp.code, resp.info));  
  
  } else {  
    dlg.setInfo(this.tags.restoreSuccess);  
  
  }  
})
```

2.5. 报表开发

至简网格内置了 echarts，并默认使用 echarts 生成图表。

为了使用 echarts，必须在起始页 index.html 中引用。echarts 已集成到客户端版本中，业务无需自己下载。除折线图、柱状图等基本图表外，还包括 tree/relation/scatter/sunburst 四个高级图表。

```
<script src="/assets/v3/echarts.js"></script>
```

echarts 详细使用方法，请参考 [echarts 官方文档](#)

与公司帐号不同，多个公司帐号，会使用当前选中的公司帐号；但是在个人服务中，端侧会自动选择个人帐号，用户不会感知。

员工授权

帐号

lzx lzx ×

角色

财务



公网访问

确定

关闭

3. 服务举例

3.1. 极简 CRM

客户关系管理系统，为小微企业管理客户信息，实现客户信息、联系人信息、订单信息、回款信息、售后服务信息记录，实现了基本的业财一体化。

提供了实时的简报，让每个销售人员能够掌握自己当前的销售进展。服务报表中，提供了公司范围的收支报表与按产品维度的收支报表。

3.2. 极简会员

会员管理系统，为服务行业提供的客户会员软件，与 CRM 不同，这类服务直接面向个人，所以其中不包括企业信息，只有会员信息，完成会员登记、订单管理、消费记录，能够输出实时的图形化报表，针对每个会员都可以一键导出所有服务记录存入 word 文档中，便于新员工学习，提升服务水平。





会员详情页，可以创建订单、增加消费记录，当会员对消费记录有不同意见时，可以输入密码校验订单是否被恶意修改过。

← 王二-骨盆修复

创建时间	服务量	余额	备注
2024-03-19 17:46	2	10	111
2024-03-19 19:51	1	9	1233

点此可以修改上课记录的备注

将消费记录导出到word文档中



一键导出消费记录到本地的一个 word 文档。

3.3. ClassHour

课时管理系统，为课外辅导班、兴趣班开发的学员课时管理服务。能够实现学员登记、订单管理、课时管理等，并且能够输出实时的报表。结合一些辅导中心的激励级制，提供了简单的积分奖励机制。

12			
姓名	性别	积分	最近上课时间
张柳	女	2	2024/3/19
胡思想	男	59	2024/3/19
李大地	男	25	2024/3/19
陈四五	女	23	2024/3/19
李二三	女	62	2024/3/19
李三	男	1	业务报表 3/
王五	男	0	系统设置 3/
王三	男	0	增加上课记录 3/
李四四	女	1	增加学员 3/
王二	男	1	2024/3/
输入手机号、姓名模糊搜索学员			
搜索(1~)			

学员信息可以通过模糊搜索查找，也可以在这里为一个或多个学员创建课时记录，课时会自动扣减。



点击学员可以看到学员详情，在这个界面可以创建订单，在订单上记录课时等。

← 胡思想-素描班

建档时间	课时	余额	备注	
2024-03-19 09:37	2	8	答弟弟	
2024-03-15 21:11	60	10	10000	
2024-03-15 20:17	2	70	222	
2024-03-15 20:12	2	72	22222	
2024-03-15 17:34	2	74	2	
2024-03-15 17:34	2	76	2	
2024-03-13 21:24	2	78	特殊t	

点此可以修改上课记录的备注

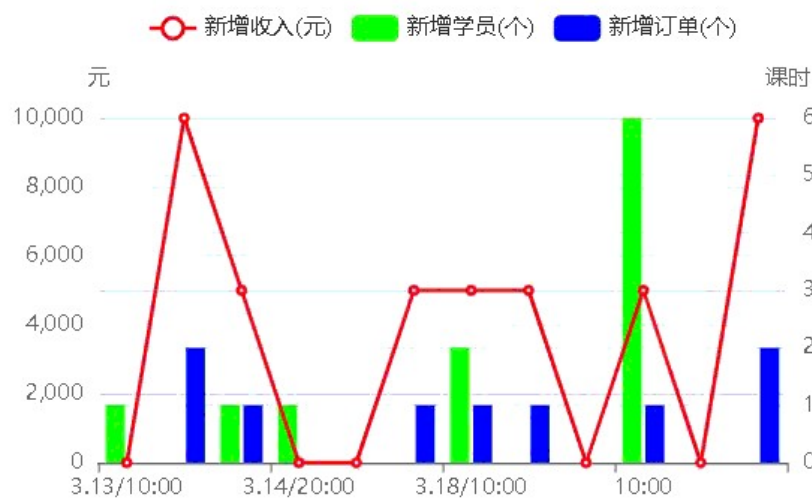
将上课记录导出到word文档中



在课时记录中，可以了解学员的学习进度。与会员管理系统类似，这里也可以一键导出学员的上课记录到一个 word 文档中，可以将这个文档转给学员家长。

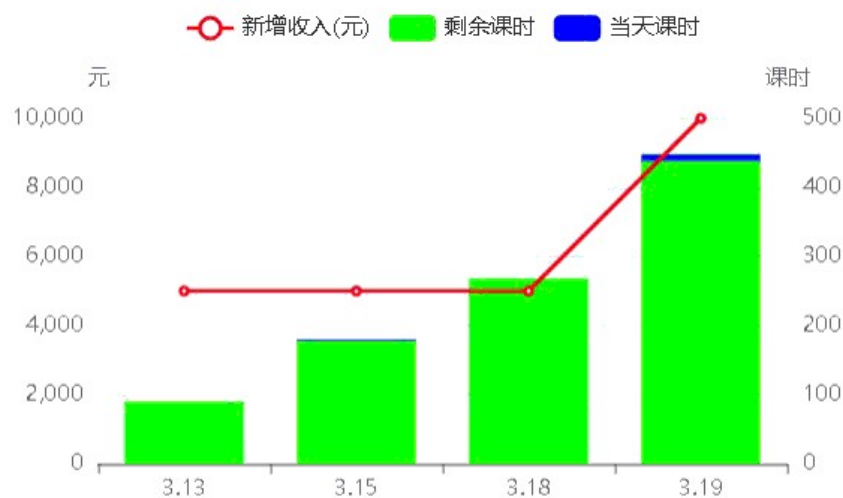
← 业务报表

简报



动漫班(5000,90课时)

套餐报表



报表中有总体的报表，也有单个套餐的报表，点击报表上面蓝色的图标就可以查看生成报表的原始详细数据，那里包括更多的信息。