**可视化平台**

**(内部资料 请勿外传)**

**修订记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **时间** | **修订内容** | **修订人** | **审批人** |
| 1 | | 2019.12.7 | 初建 | 王龙 |  |
| 2 | | 2020.02.11 | 细化 | 王龙 |  |
| 3 | | 2020.03.06 | 调整文档结构，增加进展与状态和详细接口 | 王龙 |  |
| 4 | | 2020.03.09 | 增加接口明细 | 王龙 |  |
| 5 | | 2020.03.13 | 更新项目状态  更新模板目录结构规范 | 王龙 |  |

进展

**当前状态:**

1. 操作（PC端）前端开发中（80%）
2. ‘插座’模板的开发中（40%）
3. 服务器端开发还未启动

**进展记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **时间** | **工作内容** | **作者** |
| NA | | 2020.03.13 | 模板：  1.放弃使用以前scaffolding环境，重新实现脚手架采用**react-app-rewired**方案  2.完成mock数据在模板中的运行  3.设计并开发通用HOC，让普通组件变为可配置组件  PC端：  H5样式预计已基本完成,部分细节有待完善；  使用vw方案开发H5已完成； | 王龙 |
| NA | | 2020.03.06 | 1.模板中模块分割与定义  2.模块开发环境与框架搭建  3.‘插座’模板中首页模块定义  4. 操作（PC端）前端中‘产品列表’、‘产品创建’、‘产品配置’样式添加 | 王龙 |
| NA | | 2020.02.29 | 1.产品配置与添加界面完成  2.开发mock API 支持调试 | 王龙 |

目录

目录

[进展 3](#_Toc34408170)

[目录 4](#_Toc34408171)

[1 背景 6](#_Toc34408172)

[2 系统目标 6](#_Toc34408173)

[3 名词解释 7](#_Toc34408174)

[4 系统需求 8](#_Toc34408175)

[4.1.1 上下文图 8](#_Toc34408176)

[4.1.2 用例图 9](#_Toc34408177)

[4.1.3 配置逻辑 9](#_Toc34408178)

[5 系统设计 10](#_Toc34408179)

[5.1 系统框图 10](#_Toc34408180)

[5.2 物理视图 11](#_Toc34408181)

[5.2.1 系统组件关系 12](#_Toc34408182)

[5.3 模板概念 12](#_Toc34408183)

[5.4 组件概念 13](#_Toc34408184)

[5.4.1 组件配置界面 13](#_Toc34408185)

[5.5 自定义功能 14](#_Toc34408186)

[6 详细设计 14](#_Toc34408187)

[6.1 服务器功能 14](#_Toc34408188)

[6.1.1 设备类型列表接口 14](#_Toc34408189)

[6.1.2 存储当前模板配置接口 14](#_Toc34408190)

[6.1.3 打包接口 14](#_Toc34408191)

[6.2 模板规范 14](#_Toc34408192)

[6.2.1 目录结构 15](#_Toc34408193)

[6.2.2 文件说明 15](#_Toc34408194)

[6.2.3 preview界面 16](#_Toc34408195)

[6.2.4 打包命令 17](#_Toc34408196)

[6.2.5 模板开发（react） 18](#_Toc34408197)

[6.3 Profile规范 18](#_Toc34408198)

[6.4 国际化 20](#_Toc34408199)

[6.5 互斥条件 20](#_Toc34408200)

[6.6 系统开发环境 21](#_Toc34408201)

[7 UI设计 21](#_Toc34408202)

[7.1 平台前端 21](#_Toc34408203)

[7.2 插座模板 22](#_Toc34408204)

# 背景

目前的业务中，经常需要开发设备控制界面。大部分**简单**的控制界面有一下几个特点:

* 功能不同，但业务逻辑类似
* 需求较多
* 开发时间短，但上线时间紧

目前的开发流程：

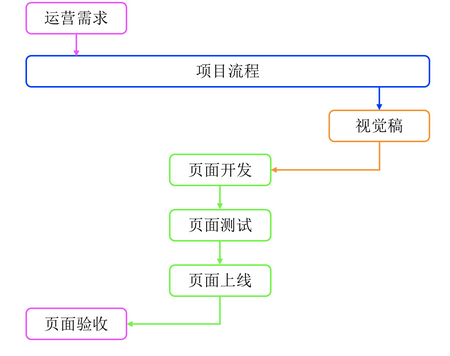


图1.目前的开流程

# 系统目标

本系统对高频和重复的控制界面，抽象并且标准化为模板，系统同时提供对模板的可视化配置的支持。

产品/运维等需求方可以直接通过系统生成出UI包，开发和设计人员只需要提供模板与维护该系统即可。

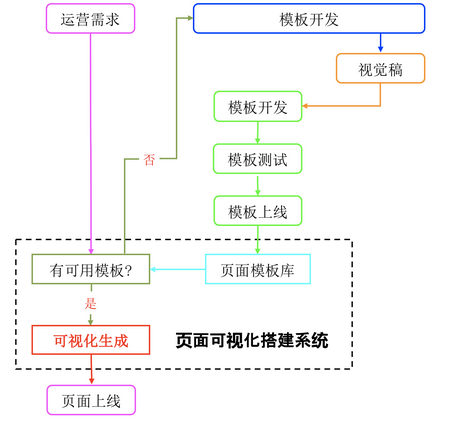


图2：改进后开发流程

系统为B/S结构，但以可执行程序的形式发布，运行在单机上。作为kit开发者平台的辅助工具，不提供账号、存储、权限等功能。

# 名词解释

**组件：**

界面组成的最小单位，同时也是可配置的最小单位。组件依赖外部给其输入**参数，**才能完成完整的渲染逻辑。

系统会对其参数提供可视化的编辑界面

**模板：**

模板是一个包含了完整业务逻辑的界面，不同的页面模板适用于不同的业务功能, 从模板库中选择合适的页面模板并派生出默认业务页面, 再对默认页面进行可视化编辑, 从而生成目标业务页面.



图3：从模板派生具体界面

# 系统需求

### 上下文图

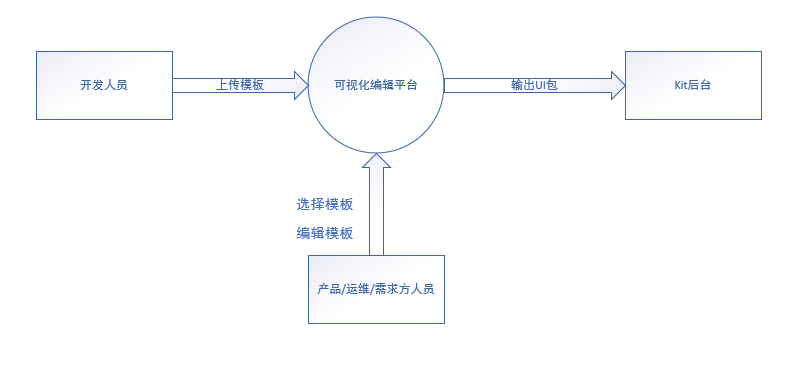


图4：系统上限文图

### 用例图

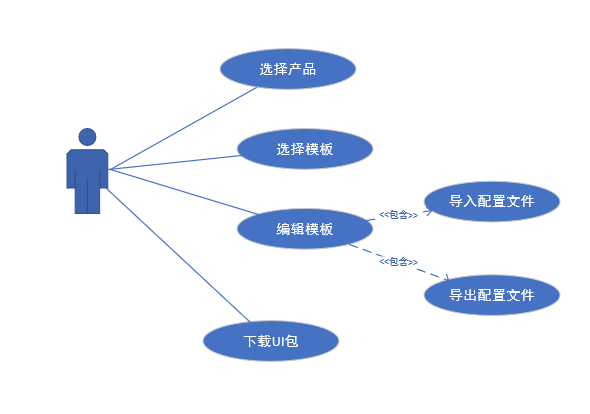


图5：系统用例图

### 配置逻辑

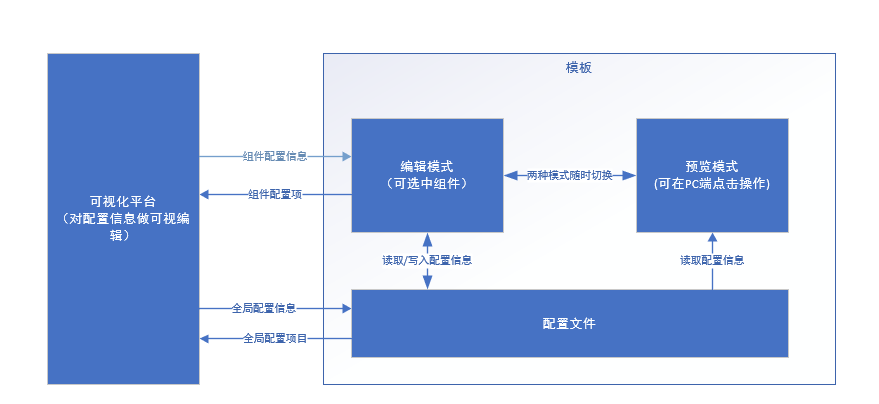


图6：平台对模板中的配置项进行可视化编辑

# 系统设计

## 系统框图



图7：系统框图

配置信息持久化：

平台应该提供保存当前配置的功能，但为了减少复杂度，减轻对云端的依赖。系统不提供账号、登陆等功能。

用户可导出当前的配置文件，以作为下次修改或者复用之基础。

## 物理视图

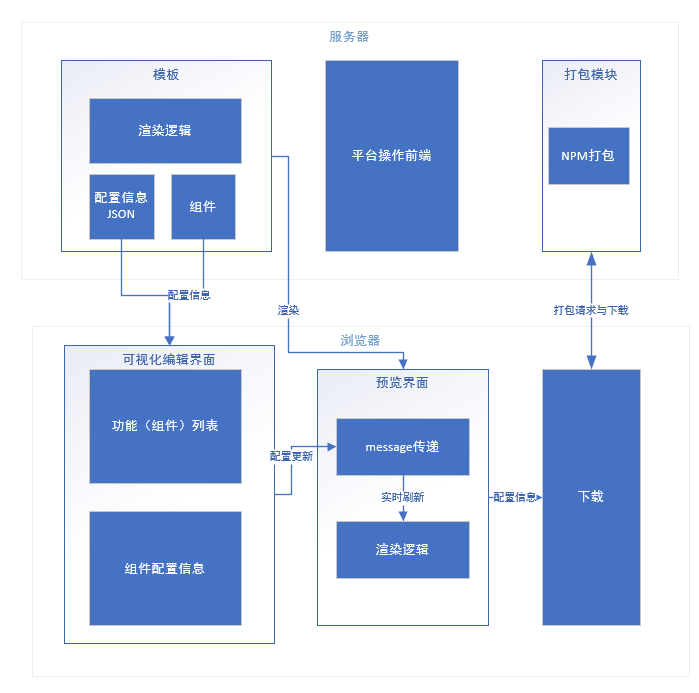


图8：物理架构图

### 系统组件关系

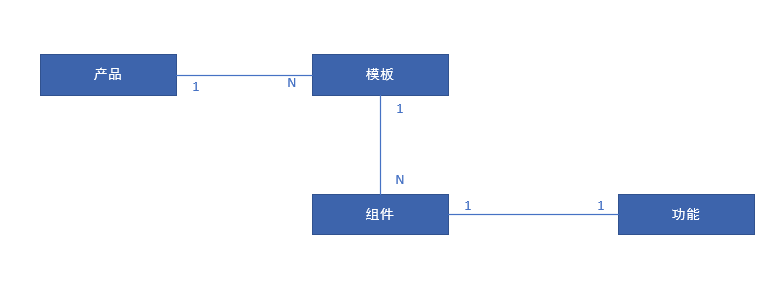


图9:系统组件关系图

由图中可以看到：

1. 一种产品可以对于多个模板，如插座可以有多个风格迥异的模板
2. 一个模板本质上是多个组件的组合
3. 一个组件对应一个设备功能

## 模板概念

模板本质上是组件与组件默认属性的组合，如上所述在一个模板中，一个功能即是一个UI上的组件，同时也是配置的最小维度

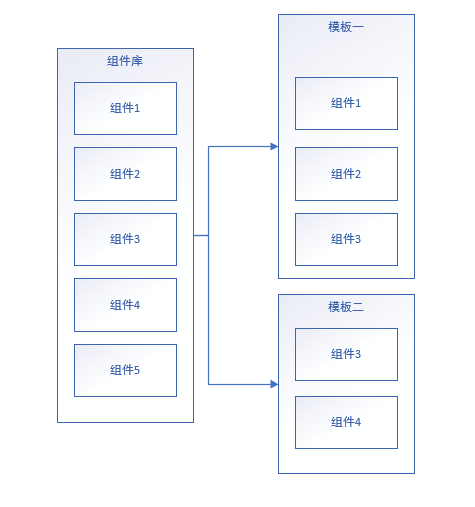


图10：组件库与模板图

因此模板可以用一个JSON数据来表示，即组件（功能）列表+默认数据。

**模板 = 组件（功能）列表+默认数据 + 渲染逻辑**

特别注意:

**模板需要标明是否支持自定义功能，以及自定义功能在界面中的展示（排列）逻辑**

## 组件概念

组件是业务内容的呈现载体, 不同的功能, 封装在不同的组件中. 所以页面模板中的组件是差异化的, 如插座的开关、和空调的模式调节组件在props(参数)和功能上是截然不同的。在编辑不同组件内容时, 组件配置数据的数据结构是也是差异化的.

### 组件配置界面

系统中的组件必须提供一个JSON Schema文件来对其参数进行说明（类似于xml的xsd标准），按照 JSON Schema 规范对 JSON 数据进行描述, 可以动态渲染出配置表单; 且 JSON Schema 可以对编辑后的数据做格式校验, 避免编辑错误. 这比编写一个表单页面更加简单和高效.

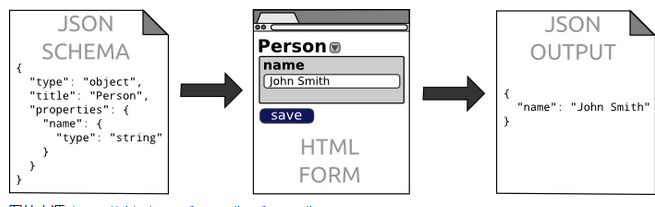


图11：使用JSON Schema实现可视化编辑

## 自定义功能

一个模板中难以包括所有的功能，因此提供扩展点至关重要（但不是必需的），系统组件库提供3类组件，作为自定义功能的实现方式

1. on off类型组件
2. 枚举类型组件
3. 连续类型组件

# 详细设计

## 服务器功能

服务器端功能设计，以尽量简单为目标。不涉及数据库的使用。

### 模板列表接口

入参:

无

出参：

{  
 **error**: **null**,  
 **data**: {  
 **list**: [  
 {  
 **id**: 123,  
 **title**: **'模板1'**, *//string 模本的名称* **editable**: **true**,*//boolean 是否支持编辑* **extendable**: **true**,*//boolean 是否支持新增自定义参数* **category**: **'电工'**,  
 **supports**: [**'开光'**, **'定时'**, **'延时'**, **'童锁'**, **'电流'**],  
 **snapshot**: **'https://ss2.baidu.com/6ONYsjip0QIZ8tyhnq/it/u=1924313363,3424253386&fm=173&s=F004DA149C203F0FF29950C20300A090&w=600&h=339&img.JPG'**,  
  
 **products**: [  
 {  
 **model**: **'JS-23'**,  
 **name**: **'SP MINI'**,  
 },{  
 **model**: **'JS-27'**,  
 **name**: **'SP MINI'**,  
 }  
 ]  
 },  
 ]  
 }  
}

### 添加模板

入参：

templateId, {model, name}

出参：

同6.1.1

### 设备模板列表接口

未定

### 打包接口

在服务器环境中切换进如当前模板模拟，进行打包

## 模板规范

模板的本质是一个有特定目录结构和实现了预定接口的UI包。对于技术框架没有要求。

### 目录结构

|-- .bowerrc

|-- dist 存放正式版zip，供用户下载

|-- preview

|-- index.html 用户在PC上进行编辑预览的HTML

|-- desc.json 对于该模板的说明

|-- dependents 代码目录

|-- settings.json 平台生成的配置信息

|-- profile.json 该设备的功能

|-- intl 国际化目录

|-- zh.json

|-- en.json

|-- errors errors文案国际化目录

|-- zh.json

|-- en.json

|-- profileStrings

|-- zh.json

|-- en.json

|-- custom-scene 开发自定义场景界面的目录,参考该目录下readme

### 文件说明

* desc.json

定义该模板的基本信息

title： string 模本的名称

editable： boolean 是否支持编辑

extendable： boolean 是否支持新增自定义参数

category: enum string 所属的种类 （如socket(插座)、ac(空调)等）

### preview界面

preview中的html是用于在PC 浏览器中iframe里面开发，可以给操作者预览，并且选择编辑。

该html需要提供message接口与平台对进行通信，并根据接收到的message动态界面。

#### 接口定义

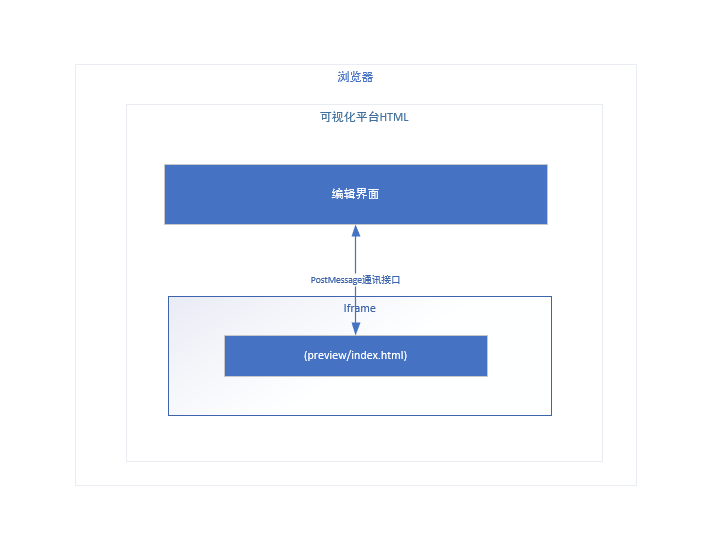


图12：平台与模板之间通过PostMessag接口进行交互

## Message 类型

*//GLOBAL\_SCHEMA 模板=>平台的  
//全局配置的SCHEMA*{  
 type:**'GLOBAL\_SCHEMA'**,  
 ***payload***:*schema*}  
  
  
*//GLOBAL\_JSON 平台的=>模板  
//全局配置data*{  
 type:**'GLOBAL\_JSON'**,  
 ***payload***:***json***}  
  
*//COMPONENT\_SCHEMA 模板=>平台的  
//选中组件的SCHEMA*{  
 type:**'COMPONENT\_SCHEMA'**,  
 ***payload***:*schema*}  
  
  
*//COMPONENT\_JSON 平台的=>模板  
//选中组件的配置数据*{  
 type:**'COMPONENT\_JSON'**,  
 ***payload***:***json***}  
  
  
  
*//TOGGLE\_INTF 平台的=>模板  
//启用/停用功能项*{  
 type:**'TOGGLE\_INTF'**,  
 ***payload***:{  
 intf:**'pwr'**,  
 enable:**true** }  
}  
  
  
*//ADD\_INTF 平台的=>模板  
//增加功能功能项*{  
 type:**'ADD\_INTF'**,  
 ***payload***:{  
 **"key"**:**'pwr'**,  
 **"intf"**: [{**"act"**: 3, **"idx"**: 1, **"ifttt"**: 0, **"in"**: [4]}],  
 **"pwr"**: {**"name"**: **"电源开关"**, **"values"**: {**"0"**: **"关"**, **"1"**: **"开"**}},  
 **"addToTimer"**:**true**,  
 **"location"**:1  
 }  
}

## 模板开发（react）



图13：模板的系统框图

使用框架技术如react、vue可以的实现，一次开发同时支持preview与编辑功能的UI。

关键点：

1. 所有的组件属性都应该集成在全局的stroe中
2. 为可编辑的组件赋予一个全局唯一的ID
3. UI内部需要维护一个组件树的结构，以便明确选中的UI在组件树种的为位置，这样才能对布局进行修改

## Profile规范

平台依赖于profile规范来得知当前模板的功能，但目前的协议无法完全满足系统的需求因此做出了少量的扩展。

入参类型

由数组构成，第一个元素表示参数格式：

1 表示枚举型

2 表示连续型

3 表示简单类型

枚举型格式

[1, V1,V2,V3]

连续型格式

[2, 最小值，最大值，步长，倍数], 倍数可以是1,10,100,1000。

简单类型

[3]

布尔类型【扩展】

[4]

云类型，如定时、延时、历史统计等复杂的功能【扩展】

[5]

注：连续型数据上报云端，云端会处理倍数，从云端获取的历史记录中，其值为真实值。

动作类型act

Bit0 置1 get

Bit1 置1set

两个bit都是1 读写都支持

使能类型enable【扩展】

0 不启用

1 启用

是否为必选required【扩展】

0 非必选

1 必选

显示位置 location 【扩展】

1 首页

2 功能设置页

3 both

## 国际化

系统读取UI包中的静态资源文件，生成界面，让其添加或者补全新的国际化文案，在进行打包的时候上传；

国际化分为3个部分：

1. profile strings
2. 错误信息
3. 程序内部文案

## 互斥条件

互斥用一个以json方式提供出设备的互斥条件。格式如下：

{

param1: {

val1: {expression}, //在expression运算值为true的情况下，val1值不可用

val2: {expression},

},

param2: {

val1: {expression},

val2: {expression},

},

}

以格力空调的互斥条件举例：

{

WdSpd: {

'6': {"Mod==='3'||Mod==='0'||SwhSlp==='1'"},

//自动、送风模式或者睡眠开启，风速中的“强劲”不可用

},

AssHt: {

'1': {"Mod!=='4'"},

//除制热模式，辅热开启功能都不可用

},

}

平台提供其可视化的编辑方式。

互斥的提出增加的模板开发的难度

## 系统开发环境

Nodejs express

react

# UI设计

## 平台前端



## 插座模板

