基于华为 appcube 的口罩预约与配送系统的开发与设计

张鼎仁　任　娟　韩英夫
（61287 部队，四川 成都 610036）

摘　要　随着“互联网 +”的快速发展，用户的需求也日益增加，敏捷开发、快速跟进用户需求变得更加重要。华为云 appcube 是华为自主研发的 aPaaS 服务平台，开发者可以迅捷高效的整合自己的实际需求，集成云计算、大数据、视频、人工智能、5G 等多种新技术的平台，将这些新技术以组件化的方式接入和融合至开发者数据，本文以口罩预约与配送系统的开发与设计为例，通过华为 appcube 平台快速开发构建。最后总结分析本项目的成功经验，以及项目存在的不足和改进措施。

关键词　appcube　aPaaS　JavaScript　预约配送
中图分类号：TN919　　**文献标识码**：A　　**文章编号**：1007-0745（2021）09-0005-05

1 绪论

在新冠疫情影响下，口罩成为人们日常生活中必不可缺的日用品，如何分配和调度口罩成为人们面临的一项难题，即使是同一地区，不同社区和乡镇也存在需求不均衡的现象，手动通过填报表格汇总统计，这项工作本身耗时耗力，不同地区的表格格式还可能不尽相同，更会影响发放口罩的效率。因此，亟需开发一个能快速适应各种需求、能够快速构建的口罩预约与配送系统，引入的华为 appcube 开发平台可安装行业服务，平台自身拥有丰富的工具集，它是支持在云上开发、测试、部署、运维的 aPaaS 平台，针对不同水平的开发者，可采用对应程度的开发模式，促进了开发的高效应用和实现，降低了开发门槛。

2 口罩预约与配送系统总体设计方案

第一步，注册登录。开发者通过注册华为开发者中心进入应用开发；第二步，创建应用。首先定义命名空间，创建 APP 及目录并定义业务对象，之后组装前端页面，定义市政报表管理页面，配置物业人员无需登录即可预约和市政人员使用的菜单最后编译发布应用，其中组装前端页面包括定义市政人员管理页面、定义市政人员修改页面、定义物业人员预约页面、定义“根据 ID 查询”逻辑和定义“新增与编辑”逻辑；第三步，应用独立部署并对外开放。独立部署后的应用包括三个方面，分别是物业人员扫码预约、市政人员管理信息和市政人员报表统计。

3 口罩预约与配送系统详细设计方案

3.1 定义业务对象

通过 Appcube 创建 App 以及相应目录，App 名称为 MaskMgtApp，开发者支持多种数据格式的字段，这样就能灵活应对各种用户需求，同时平台可以通过增删改自定义对象，同时每个字段需要定义是否需要索引、是否必需、创建人、最后修改人、最后修改时间。

对应口罩预约的申请信息，我们同时需要自定义一个对象以表示预约信息，对象名称为 MaskMgtInfo，华为 Appcube 会自动将这些信息写入数据库表结构，开发者无需再写相关的数据库增删查改的接口，省去了大量数据交互的工作量，这种可视化建立表结构的方法也为用户更快的了解项目提供了基础（口罩预约信息属性表如表 1 所示）。

3.2 组装前端页面

3.2.1 定义“新增与编辑”逻辑

针对业务场景的复杂内容，Appcube 平台提供了脚本 Script，Appcube 的脚本支持 Javascript。加入脚本的具体方法是通过页面的 Logic 目录添加脚本，创建一个名称为 EditMaskMgtInfo 的脚本名称。对于新创建的脚本默认是锁定的状态，如果要编译并执行该脚本，需要解锁并输入测试对应的脚本。脚本代码如下所示：

```
// 本脚本用于新增或者修改信息
import * as db from ‘db’;// 导入处理 object 相关的标准库
import * as context from ‘context’;// 导入上下文相关的标准库
import * as date from ‘date’;
import * as buffer from ‘buffer’;
// 定义入参结构，入参包含 1 个参数：业务对象，为必填字段
@action.object({ type: “param” })
export class ActionInput {
    @action.param({ type: ‘Struct’, required: true, label: ‘object’ })
  maskMgtInfo: object;
}
// 定义出参结构，出参包含 1 个参数，记录业务对象的 id
@action.object({ type: “param” })
export class ActionOutput {
  @action.param({ type: ‘String’ })
  maskMgtInfoId: string;
```

表 1 口罩预约信息属性表

预约对象属性	字段标签	字段名称	数据类型	是否唯一	是否必填	建议长度	默认值
联系人姓名	联系人姓名	name（复用平台预置的标准字段）	Name	否	是	NA	
小区名称	小区名称	residence	文本	否	是	64	
联系人手机	联系人手机	phoneNumber	电话	否	是	NA	
联系人地址	联系人地址	address	文本区	否	是	NA	
期望到货日期	期望到货日期	expectArriveDate	日期	否	是	NA	
预约数量	预约数量	orderCount	数字	否	是	3	
配送状态	配送状态	sendStatus	选项列表 已预约：ordered 已分配：assigned 已接单：delivered 已完成：finished	否	是	NA	ordered

```
}
// 使用数据对象 lgj__maskMgtInfo__CST
@useObject([‘lgj__MaskMgtInfo__CST’])
@action.object({ type: “method” })
export class editMaskMgtInfo {  // 定义接口类，接口的入参为 ActionInput，出参为 ActionOutput
    @action.method({ input: ‘ActionInput’, output: ‘ActionOutput’ })
    public editMaskMgtInfo(input: ActionInput): ActionOutput {
        let out = new ActionOutput();  // 新建出参 ActionOutput 类型的实例，作为返回值
        let error = new Error();  // 新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let maskMgtInfo = input.maskMgtInfo;  // 将入参赋值给 maskMgtInfo 变量，方便后面使用
            let s = db.object(‘lgj__MaskMgtInfo__CST’);  // 获取 lgj__MaskMgtInfo__CST 这个 Object 的操作实例
            // 新增口罩预约信息
            if (!maskMgtInfo[‘id’]) {
                // 合法性效验
                this.doValidate(input);
                this.checklegality(input);
                // 根据预约小区名称与联系人手机号码，查询近 8 天内是否有预约记录，如果有预约记录，则不允许再次预约
                let maskMgtInfosByCondition = s.queryByCondition({
                    conjunction: db.Conjunction.AND,
                    conditions: [
                        { field: “lgj__expectArriveDate__CST”, operator: db.Operator.gt, value: date.format(date.now(), ‘yyyy-MM-dd’, context.getTimeZone()) },
                        { field: “lgj__residence__CST”, operator: db.Operator.eq, value: maskMgtInfo[‘lgj__residence__CST’] },
                        { field: “lgj__phoneNumber__CST”, operator: db.Operator.eq, value: maskMgtInfo[‘lgj__phoneNumber__CST’] }]
                });
                console.log(“maskMgtInfosByCondition is: “ + JSON.stringify(maskMgtInfosByCondition));
                if ((maskMgtInfosByCondition || []).length) {
                    let expectArriveday = maskMgtInfosByCondition[0].lgj__expectArriveDate__CST;
                    // 效验 8 天后的日期
                    let now = date.now();
                    now.setDate(now.getDate() + 8);
                    if (expectArriveday <= date.format(now, ‘yyyy-MM-dd’, context.getTimeZone())) {
                        error.name = “EM”;
                        error.message = " 您近 8 天内有预约，不能再预约 .";
                        throw error;
                    }
                }
                else {
                    this.removeSpace(maskMgtInfo);
                    let maskMgtInfoId = s.insert(maskMgtInfo);  // 向 lgj__MaskMgtInfo__CST 插入一条数据，返回数据的唯一标识即口罩预约信息 ID
```

```
            if (maskMgtInfoId && maskMgtInfoId != “”) {
              out.maskMgtInfoId = maskMgtInfoId;
            }
            else {
              error.name = “EM”;
                error.message = “maskMgtInfo Cannot Be Added.”;
              throw error;
            }
          }
        }
        // 编辑修改口罩预约信息
        else {
          this.removeSpace(maskMgtInfo);
          let id = maskMgtInfo[‘id’];
          delete maskMgtInfo[‘id’];
           let count = s.update(id, maskMgtInfo);  // 根据口罩预约信息 ID，编辑更新 lgj__maskMgtInfo__CST 的一条数据
          if (count && count == 1) {
            out.maskMgtInfoId = id;
          }
          else {
            error.name = “EM”;
                error.message = “maskMgtInfo Cannot Be Updated.”;
            throw error;
          }
        }
      } catch (error) {
        console.error(error.name, error.message);
        context.setError(error.name, error.message);
      }
      return out;
    }
    // 去除空格
    private removeSpace(maskMgtInfo) {
      // 防止前后空格入库
        maskMgtInfo[‘lgj__phoneNumber__CST’] = maskMgtInfo[‘lgj__phoneNumber__CST’].replace(/\s+/g, “”);
        maskMgtInfo[‘lgj__residence__CST’] = maskMgtInfo[‘lgj__residence__CST’].replace(/\s+/g, “”);
      maskMgtInfo[‘name’] = maskMgtInfo[‘name’].replace(/\s+/g, “”);
      if (maskMgtInfo[‘lgj__address__CST’]) {
          maskMgtInfo[‘lgj__address__CST’] = maskMgtInfo[‘lgj__address__CST’].replace(/\s+/g, “”);
      }
    }
    // 检查具体的合法性
    private checklegality(input) {
      console.log(“checklegality”);
      let error = new Error();
      if (JSON.stringify(input.maskMgtInfo) == “{}”) {
        error.name = “EM”;
        error.message = “”;
        error.name = “EM”;
        error.message = " 信息有误 ";
        throw error;
      }
        console.log(input.maskMgtInfo[‘lgj__residence__CST’]);
      //residence 不能为空
       let residence = input.maskMgtInfo[‘lgj__residence__CST’];
      if (!residence || residence.replace(/\s+/g, “”) == “”) {
        error.name = “EM”;
        error.message = " 请填写 – 预约小区名称 .";
        throw error;
      }
        if (/[~!@#$%^&*()/\|,.<>?”’();:_+\-=\[\]{}a-zA-Z]/.test(residence.replace(/\s+/g, “”))) {
          error.name = “EM”;
            error.message = " 请填写 – 无特殊符号及数字与字母的小区名称 .";
          throw error;
        }
        console.log(input.maskMgtInfo[‘lgj__orderCount__CST’]);
      //orderCount 不能为空
        let orderCount = input.maskMgtInfo[‘lgj__orderCount__CST’];
      if (!orderCount) {
        error.name = “EM”;
        error.message = " 请填写口罩预约数量 .";
        throw error;
      }
      let numberReg = /^([1-9][0-9]{0,2})$/;
      console.log(!numberReg.test(orderCount));
      if (!numberReg.test(orderCount) || orderCount > 500) {
        error.name = “EM”;
        error.message = " 请正确填写口罩预约数量 , 最多预约 500 个 .";
        throw error;
      }
      //expectArriveDate 不能为空
        let expectArriveDate = input.maskMgtInfo[‘lgj__expectArriveDate__CST’];
      console.log(expectArriveDate);
```

```
        if (!expectArriveDate || expectArriveDate == "") {
            error.name = "EM";
            error.message = " 请填写期望到达日期 .";
            throw error;
        }
        // 效验 3 天后的日期
        let now = date.now();
        now.setDate(now.getDate() + 2);
         if (expectArriveDate <= date.format(now, 'yyyy-MM-dd', context.getTimeZone())) {
            error.name = "EM";
              error.message = " 期望到货日期，请至少填写 3 天后日期 .";
            throw error;
        }
         console.log(input.maskMgtInfo['lgj__phoneNumber__CST']);
        //phoneNumber 不能为空
          let phoneNumber = input.maskMgtInfo['lgj__phoneNumber__CST'];
         if (!phoneNumber || phoneNumber.replace(/\s+/g, "") == "") {
            error.name = "EM";
            error.message = " 请填写联系人手机号码 .";
            throw error;
        }
        // 手机号码合法
        let reg = /^1[3|4|5|7|8][0-9]{9}$/;
        console.log(reg.test(phoneNumber.replace(/\s+/g, "")));
        if (!reg.test(phoneNumber.replace(/\s+/g, ""))) {
            error.name = "EM";
            error.message = " 请填写合法的联系人手机号码 .";
            throw error;
        }
        console.log(input.maskMgtInfo['name']);
        //personName 不能为空
        let personName = input.maskMgtInfo['name'];
        if (!personName || personName.replace(/\s+/g, "") == "") {
            error.name = "EM";
            error.message = " 请填写 - 联系人姓名 .";
            throw error;
        }
          if ((/[~!@#$%^&*()/\|,.<>?"'();:_+-=\[\]{}0-9]/.test(personName.replace(/\s+/g, "")))) {
            error.name = "EM";
              error.message = " 请填写 - 无特殊符号的联系人姓名 .";
            throw error;
        }
    }
    // 长度基础效验
    private doValidate(input) {
        console.log("doValidate");
        for (let property in input) {
            console.log(property);
            if (Array.isArray(input[property])) {
                console.log('aaa' + input[property]);
                for (let tmp of input[property]) {
                    this.doValidate(tmp);
                }
            } else {
                let buf = buffer.from(input[property] || '');
                console.log('bbb' + input[property]);
                if (typeof (input[property]) == "string") {
                    console.log('ccc' + input[property]);
                    if (property != "url") {
                        if (buf.size() > 255) {
                            context.throwError('请检查 -- 某个属性是否填的长度 >255', property);
                        }
                    }
                } else if (typeof (input[property]) == "object") {
                    console.log('ddd' + input[property]);
                    for (let i in input[property]) {
                        buf = buffer.from(input[property][i] || '');
                        if (i == "lgj__residence__CST") {
                            if (buf.size() > 64) {
                                context.throwError('请检查 -- 预约小区名称是否填的长度 >64', i);
                            }
                        }
                        else if (i != "url" && i != "content") {
                            console.log('eee' + buf);
                            if (buf.size() > 255) {
                                context.throwError('请检查 -- 某个属性是否填的长度 >255', i);
                            }
                        }
                    }
                }
            }
        }
    }
```

Appcube 平台同时支持脚本验证功能，可以通过编辑器测试新增逻辑能否正常执行，测试的结果以 json 字符串格式显示。同理，可以采用同样的方法定义“根据 Id 查询”逻辑，脚本名称定义为 queryMaskMgtDetail。

脚本代码如下所示：

```
/*****************************
 * 本脚本用于按记录 ID 查询信息记录
 * ***************************/
import * as db from ‘db’;// 导入处理 object 相关的标准库
import * as context from ‘context’;// 导入上下文相关的标准库
// 定义入参结构
@action.object({ type: “param” })
export class ActionInput {
  @action.param({ type: ‘String’, required: true })
  maskMgtInfoId: string;// 口罩预约 ID
}
// 定义出参结构
@action.object({ type: “param” })
export class ActionOutput {
  @action.param({ type: ‘Struct’, label: ‘object’ })
  maskMgtInfo: object;// 口罩预约对象
}
@useObject([‘lgj__MaskMgtInfo__CST’])// 使用数据库对象 lgj__MaskMgtInfo__CST
@action.object({ type: “method” })
export class QueryMaskMgtInfoDetail {
    @action.method({ input: ‘ActionInput’, output: ‘ActionOutput’ })
    public queryMaskMgtInfoDetail(input: ActionInput): ActionOutput {
        let out = new ActionOutput();   // 新建出参 ActionOutput 类型的实例，作为返回值
        let error = new Error();   // 新建错误类型的实例，用于在发生错误时保存错误信息
        try {
          // 必填校验
          if (!input.maskMgtInfoId || input.maskMgtInfoId == “”) {
            error.name = “TM”;
            error.message = " 请传入口罩预约记录的 Id.";
            throw error;
          }
          // 获取 lgj__MaskMgtInfo__CST 这个 Object 的操作实例
          let s = db.object(‘lgj__MaskMgtInfo__CST’);
          // 查询字段 ( 全部 )
          let option = {};
          // 查询条件
          let condition = {
            “conjunction”: “AND”,
            “conditions”: [{
              “field”: “id”,
              “operator”: “eq”,
              “value”: input.maskMgtInfoId
            }]
          };
          // 调用按条件查询 lgj__MaskMgtInfo__CST 的接口
          let record = s.queryByCondition(condition, option);
          // 如果查询到数据
          if (record && record[0]) {
            // 将结果挂入输出对象中
            out.maskMgtInfo = record[0]
          }
        } catch (error) {
          console.error(error.name, error.message);
          context.setError(error.name, error.message);
        }
        return out;
    }
}
```

3.2.2 组装页面

Appcube 中的 UI 构建器支持对所需前端页面进行组合编排，并为相关的组件属性，触发器进行配置，满足用户的业务需求，这种可视化的前端开发工具大大缩短了开发进程，开发者无需掌握太多的 Html、Css、Javascript 知识，只需进行必要的属性、事件配置，就可以快速生成所需要的前端页面。

3.3 定义市政管理报表

Appcube 报表关联的对象需要设置为“允许报表使用”，定义的报表分为按预约日统计各小区预约量报表和按期望到货日统计各小区待配发量报表。同时，为了方便物业人员的使用，能够将人们的口罩需求数量及时快速统计，我们将申请设置为通过二维码扫码预约的形式免登陆登记。

3.4 独立部署并对外开放

项目开发成功后，我们还可以导入对应的资产包给其他用户，以便其他用户直接使用和测试该项目。这种软件包式的管理方式大大提高了项目部署效率，免去了部署开发环境的重复操作，同时云端部署还省去了运维成本，提高了项目运行的稳定性。

4 结语

本文主要介绍了基于华为 Appcube 平台开发的口罩预约与配送系统，它可以满足新冠疫情防控期间口罩需求的统计汇总的现实需求，解决了基层繁杂的表格填报难题，对其他行业的数据统计和汇总同样具有借鉴意义，更加值得学习的地方是该 aPaaS 平台云开发、云测试、云部署、云运维的特点。它不再依赖于复杂的现场网络环境和硬件局限性，同时大大缩短了项目开发周期，提高了工作效率。同时在 Appcube 平台开发过程中也发现了一些问题，比如开发的旧模块无法适配新版本的问题。在后续的学习和工作中，我将不断充电学习，开发具有高适配性、高安全性的口罩预约与配送系统。