Table of Contents

交通巡检平台解决方案

如何打印文档?

1. 点击左侧菜单拦中的”下载PDF版本“
2. 打开pdf后，点击做上角的打印按钮

> 手机上查看PDF版本，请先下拉页面，然后点击击文档正文区域左上角图标来展开菜单

版本

序号	修改内容	撰稿人	修订时间	版本号
1	撰写 技术方案	尹彬	2018-11-21	V0.1
2	增加 方案报价	尹彬	2018-11-22	V0.3
3	增加 PDF版本的下载	尹彬	2018-11-22	V0.4
4	修改 方案1主要模块和功能	尹彬	2018-11-24	V0.5

需求概述

关于需求背景: 应济南市交通设施运维需求，需要尽快上线一款运维软件，达到统一管理运维日常、巡检、维修过程中产生的数据及图片、影像资料，以及巡检轨迹等数据。最终产出一系列的统计数据报表、图表，及综合数据信息。并提交相关部门使用。

---

1. 硬件部分： gps定位和车辆行驶轨迹的显示，需要采购gps硬件或接入已有GPS接口。
2. 软件部分：
    - 《交通设施巡检管理系统》
    - 《交通设施巡检APP》
3. 主要系统角色：
    - 系统管理员
    - 软件开发管理员
    - 公司文员（一些非维修人员录入信息之外的数据维护）
    - 设施维保人员
    - 其他角色待定（可灵活配置）...

> 说明：
>
> 1. 角色就是一组功能的合集，并起了一个名字，某人设置了这个（可以设置多个）角色，则就拥有了这些功能访问的权限
> 2. 角色和权限的维护：系统管理员可以增加和修改其他角色，并设置相关可访问菜单和按钮，然后分配给某些用户即可
> 3. 系统账号，由系统管理员统一分配。个人账号的密码，在分配后可以自己修改。

解决方案1

方案一是交通巡检管理系统的解决方案，仅包括pc端，不包括APP端。

一、 主要模块和功能：

1. 设施维修人员，每天维修的数据，自己录入系统，包括:
    - 护栏
    - 标志

    > 后续可增加模块和属性
2. 维保数据的统计分析，支持全文和关键词检索：
    - 月道路设施维护信息分析
    - 月道路撞损设施信息分析
    - 月巡检维护设施信息分析
    - 现有交通设施的信息建立与档案管理
    - ...
3. 维修车辆的监控
    - 出车，随车gps设备每间隔1秒-60秒把位置发送服务器
    - 管理系统可地图中查看车辆定位
    - 管理系统可地图中查看车辆行驶轨迹
4. 其他模块
    - 基础数据维护（地区、故障类型等）
    - 用户管理
    - 菜单管理
    - 角色管理
    - 权限管理
    - 机构管理
    - ...

> 注：详细的模块功能设计和UI设计，在签订合同付设计款项后才能设计和制作，见：实施阶段和运维。

二、 后台服务端技术方案

1.1. 后台项目主要采用技术：

- 核心框架：springcloud 全家桶
- 安全框架：Spring Cloud Oauth2、Spring Security
- 分布式任务调度框架：elastic-job
- 持久层框架和类库：MyBatis、Mybatis_PageHelper、通用Mapper4
- 数据库连接池：阿里巴巴 Druid
- 日志管理：Logback
- 三方服务： 短信、支付等

1.2. 后台项目结构规划：

```
├─villealife-master----------------------------父项目，公共依赖
|  |
```

```
|  ├─villealife-eureka-------------------------微服务注册中心
|  |
|  ├─villealife-discovery----------------------微服务配置中心
|  |
|  ├─villealife-monitor------------------------微服务监控中心
|  |
|  ├─villealife-zipkin-------------------------微服务日志采集中心
|  |
|  ├─villealife-gateway------------------------微服务网关中心
|  |
|  ├─villealife-provider
|  |  |
|  |  ├─villealife-provider-mdc-----------------数据服务中心
|  |  |
|  |  ├─villealife-provider-omc-----------------订单服务中心
|  |  |
|  |  ├─villealife-provider-opc-----------------对接服务中心
|  |  |
|  |  ├─villealife-provider-tpc-----------------任务服务中心
|  |  |
|  |  └─villealife-provider-uac-----------------用户服务中心
|  |
|  ├─villealife-provider-api
|  |  |
|  |  ├─villealife-provider-mdc-api-------------数据服务中心API
|  |  |
|  |  ├─villealife-provider-omc-api-------------订单服务中心API
|  |  |
|  |  ├─villealife-provider-opc-api-------------对接服务中心API
|  |  |
|  |  ├─villealife-provider-tpc-api-------------任务服务中心API
|  |  |
|  |  ├─villealife-provider-sdk-api-------------可靠消息服务API
|  |  |
|  |  └─villealife-provider-uac-api-------------用户服务中心API
|  |
|  ├─villealife-common
|  |  |
|  |  ├─villealife-common-base------------------公共POJO基础包
|  |  |
|  |  ├─villealife-common-config----------------公共配置包
|  |  |
|  |  ├─villealife-common-core------------------微服务核心依赖包
|  |  |
|  |  ├─villealife-common-util------------------公共工具包
|  |  |
|  |  ├─villealife-common-zk--------------------zookeeper配置
|  |  |
|  |  ├─villealife-security-app-----------------公共无状态安全认证
|  |  |
|  |  ├─villealife-security-core----------------安全服务核心包
|  |  |
|  |  └─villealife-security-feign---------------基于auth2的feign配置
|  |
|  ├─villealife-generator
|  |  |
|  |  ├─villealife-generator-mdc----------------数据服务中心Mybatis Generator
|  |  |
|  |  ├─villealife-generator-omc----------------数据服务中心Mybatis Generator
|  |  |
|  |  ├─villealife-generator-opc----------------数据服务中心Mybatis Generator
|  |  |
|  |  ├─villealife-generator-tpc----------------数据服务中心Mybatis Generator
|  |  |
|  |  └─villealife-generator-uac----------------数据服务中心Mybatis Generator
```

2.1. 后台的前端主要采用技术：

技术名称	作用
Nodejs	运行引擎
Vuejs	框架
Vue-Router	Vue路由
Vuex	Vue状态管理
Axios	交互处理
Element UI	UI框架
Pug	Node模版
Sass	样式预处理
Webpack	模块打包
Electron	桌面软件制作
Yarn	依赖管理
Google Material-Icons	图标字体

2.2. 后台的前端项目结构规划：

```
├── README.md                    项目介绍
├── .electron-vue                    构建脚本目录
|   ├── webpack.base.conf.js         webpack基础配置,开发环境,生产环境都依赖
|   ├── webpack.dev.conf.js          webpack开发环境配置
|   ├── webpack.prod.conf.js         webpack生产环境配置
|   ├── build.js                     生产环境构建脚本
|   ├── dev-server.js                开发服务器热重载脚本，主要用来实现开发阶段的页面自动刷新
|   ├── utils.js                     构建相关工具方法
├── build                        桌面安装包目录
|   ├── linux-unpacked               linux系统可执行程序
|   ├── mac                          mac系统可执行程序
|   ├── win-unpacked                 windows系统可执行程序
├── config                       项目配置
|   ├── dev.env.js                   开发环境变量
|   ├── index.js                     项目配置文件
|   ├── prod.env.js                  生产环境变量
  ├── dist                       源码“编译”输出目录
|   ├── electron                     用于桌面打包的编译目录
|   ├── web                          用于部署服务器通过pc浏览器访问的编译目录
├── src                          源码目录
|   ├── main.js                      入口文件
|   ├── config                       入口相关配置文件
|   ├── app.vue                      根组件
|   ├── components                   公共组件目录
|   |   └── base                       基础组件
|   |   └── layouts                    布局组件
|   |       └──header                      头部组件
|   |           └──index.vue
|   |           └──index.less
|   ├── styles                       样式资源
|   |   └── index.less                 样式入口
|   |   └── var.less                   变量
|   |   └── reset.less                 重置样式
```

```
|   |   └── common.less                公共样式
|   ├── images                       图片资源
|   |   └── auth                       验证模块图片
|   ├── pages                        页面目录
|   |   └── auth                       验证模块
|   |   |   └── login                    登录文件
|   |   |       └── index.vue              登录页
|   |   |       └── index.less             登录页样式
|   |   └── other module1              其他模块1... ...
|   |   └── other module2              其他模块2... ...
|   |   └── other module3              其他模块3... ...
|   ├── routes                       路由目录
|   |   └── auth                       验证模块
|   |       └── index.js                 验证模块入口
|   |   └── index                      所有模块汇总
|   ├── store                        应用级数据（state）
|   |   └── index.js                   所有模块数据汇总
|   |   └── auth                       验证相关数据模块
|   |       └── index.js                 验证模块入口
|   |       └── mutation-types.js        类型
|   |       └── actions.js               actions
|   |       └── mutations.js             mutations
|   |       └── getters.js               getters
|   |       └── state.js                 默认状态
|   ├── services                     接口api定义
├── .eslintrc.js                     eslint规则配置
├── package.json                     nodejs版本、依赖、开发测试打包等配置
```

> 如果界面是一个软件的皮囊，那后台就是这个软件的灵魂。 我公司将采用海尔、海信在广泛使用的微服务、前后端分离架构，为您搭建高可维护、高负载、高性能的服务端和管理软件。

解决方案2

一、 技术方案

在方案一基础上，增加App端： 现有效果较好的app开发技术方案对比：

比较内容	Flutter	Weex	React Native	Android Native
平台实现	通过Dart虚拟机编译成机器码	1. Vue编写的Web页面编译成JS bundle； 2. Native端解析DOM，生成真实的Native控件； 3. Android平台通过ART虚拟机编译成机器码。	1.React编写JS文件，如果是UI界面，会映射到Virtual DOM； 2.通过C++编写的Bridge调用原生的API，控件则是根据DOM映射到原生的View； 3. Android平台通过ART虚拟机编译成机器码。	通过ART虚拟机编译成机器码
绘制引擎	Skia engine	1. 2D绘制：JSCore + Skia engine 2. 3D绘制：JSCore + OpenGL ES	1. 2D绘制：JS V8 + Skia engine 2. 3D绘制：JS V8 + OpenGL ES	1. 2D绘制：Skia engine 2. 3D绘制：OpenGL ES
核心语言	Dart	Vue	React	Java/Kotlin
APK大小	16M	18M	15M	10M
上手难度	一般	容易	较难	/
框架程度	重	较轻	较重	一般
社区	丰富，谷歌团队支持	较小、阿里支持（目前托管Apache）	活跃、Facebook支持	庞大、谷歌团队支持
支持	Android、IOS、Web	Android、IOS、Web	Android、IOS	只支持Android
软件发布	支持热更新	支持热更新	支持热更新	支持热更新
未来前景	1. 谷歌打算推出的新系统Fuchsia采用Flutter； 2. 社区活跃，文档完善； 3. 版本更新迭代很快，基本两天更新一次； 4. 有上线的手机App。	1. 前景不明，阿里自己的产品正在逐渐去Weex化，采用Hybird或ReactNative代替； 2. 后期维护力度小，文档混乱； 3. 版本更新较慢。	1. 成熟度高，已经有很多成功案例； 2. 版本有持续更新，文档完善； 3. 受限于本身跨平台机制，上限不高。	由于目前各种跨平台框架都有缺陷、不够成熟，加上性能方面的优势，Native开发暂时还是主流方式。
性能	自带绘制引擎，号称接近原生，但目前就实际效果看，还有一些距离	因为多了一层JS解析，渲染慢一些	跟Weex差不多	性能表现最佳
TV交互支持	不太友好，需要开发者做大量的焦点处理和按键分发工作，有文档但资料较少	不太友好，需要开发者做大量的焦点处理和按键分发工作，没有文档	明确支持TV交互，有相关文档	本身支持TV平台，5.0版本开始，谷歌大力支持android tv，开发者处理焦点问题和按键事件比较容易
动画	满足现有的动画需求	缺少多页面跳转时的过场动画	满足现有的动画需求	满足现有的动画需求
适用场景	适用跟系统交互少，页面不太复杂的场景	适用于电商类App、业务场景不复杂、需要动态更新的场景	适合业务不太复杂、页面简单的小项目	适用跟系统高度耦合，追求高性能的场景
免安装	不支持	可以支持，需要实现类似快应用的框架（国内只有小程序和快应用支持免安装）	可以支持，需要实现类似快应用的框架（国内只有小程序和快应用支持免安装）	Android的Instant App支持免安装，但是国内用不了
初步结论	综合上面对比数据及开发Demo实践得出如下结论： 1. TV应用由于焦点及按键事件等处理较特殊，基本所有跨平台方案都没有对其支持，并且业务高不开跟底层系统的大量交互，涉及到各个平台兼容问题，开发人员技术栈限制，目前使用原生开发效率更高，效果更好； 2. 谷歌强推Flutter，即将推出的Fuchsia新系统打算统一桌面端、移动端和网页端，并且会采用Flutter框架，以此来推广新系统，值得关注； 3. Weex最大的优势是入门简单，前端和移动端开发都容易上手，代码结构比较清晰，但目前版本迭代较慢，已贡献给Apache，其原班开发人员已不再维护，使用该框架，存在一定风险； 4. ReactNative成熟度高，明确支持TV端交互，体系已经完善，但是上手门槛高，开发难度大，目前在经历大的重构，跟Native控件耦合度高，天花板低，可以小范围使用； 基于以上考虑，TV应用建议使用原生开发，手机应用建议关注Flutter。			

so，最终决定采用flutter作为app UI开发的技术，dart作为主要app开发语言。

> 参考资料：flutter介绍
>
> 说明： 采用目前最先进的APP开发技术，最终实现的APP，从性能、流畅度、动画特效等各个方面来讲都会比其他h5、混合开发方式制作的APP要好太多。

二、 FAQ

我公司技术有限公司有什么优势，为什么我们是你最佳的APP软件制作商?

APP项目失败的情况主要有3种：

1. 项目一直拖到，一直不能交付，并且很可能继续拖下去
2. 项目在开发期间不断上涨成本
3. 项目完成，BUG多多，不能使用

这三种状况会把你困在了一个尴尬的境地，你一方面心疼你已经花费的时间和精力，希望这个项目完成下去;另一方面你看着无限拖延的时间和无限增加的成本，其实心里清楚这个项目是完成不了了。

至于为什么你现在会经历目前的状况?只有复盘项目流程才能清楚。

情景一： 初次和APP开发公司见面，对方异常热情。你刚刚把项目简要的和他说了一遍，他就立马告诉你：“这个项目交给我们做，X万，X个月包你上线!”你惊讶于他迅速的判断，心想有可能是因为对方经验丰富。

纠正： APP开发的周期和报价是按照客户需求所判定的，没有做过详细的APP需求梳理和评估，任何报价都是瞎 扯 淡 !

我公司： 免费提供1对1的APP需求评估服务，专业客服梳理APP需求，最终形成评估报告。有凭有据的提供靠谱App开发周期和报价。

情景二： 项目合同一签完，之前每天打电话给你的APP开发公司一下变得安静了。你开始主动打电话去了解项目的进度和情况，可是对方爱答不理，或是敷衍两句：“项目很正常，很快就做完的。”明明是自己的项目，可你一点掌控力都没有。

纠正： 作为负责任的APP开发公司，在项目正式开始后，理应指派专人负责双方的沟通。客户应该及时了解到项目的状况和进程，以便对整体项目作出反馈或调整。

我公司： CTO亲自指派专属监理，全程跟踪项目进展。

情景三： 项目开发进展到一半，开发团队突然说原先的功能做不了了，留下你原地懵逼，因为你根本不知道项目做到了哪一步，你更不知道项目卡在了哪一步。

纠正： 一个APP的开发应该是分阶段性的，大致分为需求梳理、产品原型、UI设计、前端、后端、后台和测试。因此，客户首先应该知道APP开发的每一阶段所需周期，其次应该知道APP开发进展到哪一个阶段，有什么问题，有什么成果。

我公司： 定制服务将APP开发分为需求预评估、产品原型设计、UI设计、APP端开发、服务端开发、接口联调合和测试及验收共7个阶段。每一个阶段的开发都有独立的周期，按步交付。每一阶段完成后，需要客户和开发团队双方进行确认，确认无误后，再继续进行下一阶段的开发。

情景四： 项目交付的时间从春暖花开一直拖到了寒冬腊月，拖着拖着最后还是完成了。你怀着激动的心情打开时才发现，说好的开发一个APP，拿到一看却是个手机网站??

纠正： 客户和开发团队在前期时并没有沟通清楚开发模式，甚至都没有沟通清楚产品质量标准是什么，项目就启动了。开发团队也可能钻了“客户不懂技术”的空子，拿模板、拿嵌套的方式忽悠客户。

我公司： 定制服务采用标准化的我公司开发技术，并且拥有10年的技术积累。许多质量低、体验差的烂尾项目，使用我公司技术后实现快速重建，从而获得更高的质量保证和更快速的迭代能力。

情景五： 开发团队告诉你项目完成并向你扔了一个APP安装包，你接住APP安装包后大吼：说好的交付文档呢?说好的上线应用商店呢??

纠正： 在合同中应规定交付规则和交付项目，一个APP开发的完成，不仅仅是交付一个软件，普遍上，APP前端页面代码、后端和管理后台源代码等内容都应该交付给客户。

我公司： 在项目收尾时，按照标准化验收体系，将APP安装包、前后端源代码、需求文档等十几项交付物完整递交，这将方便客户进行项目的更新和迭代。

同时，我公司与客户直接签约，官方保证每一个项目上线至各大应用市场。

情景六： 当你满怀憧憬，第一次打开你的APP时，却发现……登录页面无“忘记密码”，页面滑动经常卡死，还有那个社交对话框，时不时的就闪退。看着满眼的小BUG简直欲哭无泪，今后的升级和维护可怎么办呐?

纠正： 交付了但无法使用的APP，依旧是个烂尾货

我公司： 使用标准WEB技术开发iOS、Android原生App，不仅开发快、技术难度低，成本低，支持在线升级。flutter开发难度大，但制作的APP效果是最好的。weex等vue技术栈混合开发APP处于中间，在工期成本和用户使用体验上比较均衡。我公司会综合考虑APP用途、资金预算、工期等因素，为您选定最优的技术方案。

我公司拥有雄厚的技术研发实力，致力于为客户提供完美的原生APP开发解决方案。

问题和技术难点

目前为止的问题和技术重点：

1. gps设备基础软件的研发

- gps各种参数设置接口，比如：ip、端口、发送位置频率、波特率等
- 服务端需研发gps上传位置的”接口“来接收位置信息，数据分析后，存入数据库中

2. 行驶轨迹难点

- 行驶轨迹功能依赖于第1条gps基础软件的研发成功。
- 集成百度地图或高德地图开发接口，在此基础上开发车辆轨迹显示功能

3. 流程梳理、表单梳理、报表设计，等信息交流问题

- 需求提出方和软件制作方，需在项目开工编码前，建立密切沟通、快速沟通的方式，才有助于软件的成功。

4. APP的界面设计

- 设计在考虑成本的同时，需达到界面美观、简洁，功能划分合理、界面布局合理、操作边界等。

5. 原生APP的开发

- 因采用最先件的Flutter技术开发，技术上有一定难度
- APP开发人员招聘成本、APP开发人员培训比较高

建议

系统分二期来做

1. 一期：

- 做好基础数据的录入和维护，做好《方案1》中服务端的建设

2. 二期：

- 实现APP端，更方便用户使用
- 其他可以扩展的业务。。。

实施阶段和运维

1. 实施阶段

阶段序号	主要任务
[v] 1	需求调研，提供技术方案
[x] 2	报价，签订开发合同、付设计款
[x] 3	确定需求细节，编写详细设计文档
[x] 4	框架设计、架构设计
[x] 5	开发环境配置、测试环境搭建、编码规范
[x] 6	程序逻辑流程详细设计、数据库详细设计和建库
[x] 7	UI设计和客户确认
[x] 8	付开发款
[x] 9	开发分工、编码、调试
[x] 10	上线试运营、BUG修复、功能完善
[x] 11	项目代码和文档的交付、项目的验收
[x] 12	付尾款
[x] 13	运维，技术支持

2. 运维

系统自上线之日起，12个月内提供免费技术维护服务。免费无维护服务期满后，需协商签订运维合同。 年运维费用一般为合同额的15%之间。

方案1报价

费用构成汇总表：

序号	构成	费用
1	设计费用	5500¥
2	后台软件开发费用	30000¥
3	硬件配套软件研发费用	12000¥
4	阿里云服务器年租赁费用	2500¥
5	12个月免费维护	0¥
6	开票费用（3%）	1500¥
7	总费用	51500¥

> 租赁服务器的费用：
>
> 1. 每年租用阿里云服务器费用预计：2500元，包括：
>   - web服务器费用
>   - 对象存储服务或文件存储服务器费用
> 2. 服务器租用年限越长折扣越大

方案2报价

收费项目

一、苹果企业开发者账号申请工作内容：

1. 每年固定上交苹果费用： 99美元（按当前汇率计算）
2. 第一次申请苹果企业开发者账号人工费用
3. 邓白氏编码申请费用
4. 资料准备，上传等的人工费用

申请账号时间为5个工作日。

二、APP上架的工作内容：

1. app上架App Store 上架审核周期：5个工作日
2. app上架安卓应用市场工作 上架的应用市场有：
    - 腾讯开发者平台
    - 360开发者平台
    - 小米账号
    - 华为开发者平台
    - 百度开发者平台
    - 91手机助手开发者平台

三、软件开发收费内容：

1. 管理后台 (此费用包括在方案1中)
2. 派单员端APP（安卓、苹果）
3. 维修员段APP（安卓、苹果）
4. 产品设计和12个月免费维护

费用构成汇总表：

具体需求待定，根据工作量和技术难度，开发公司付出的代价不同则报价也不同，价格只是预计价格：

序号	构成	费用
1	APP UI界面设计费用	10000¥
2	苹果企业开发者账号申请	800¥
3	APP上架人工费用	1000¥
4	APP原生开发费用（ios、android）	6万 - 8万?
5	12个月免费维护	0¥
6	开票费用（3%）	2150¥?
7	总费用预计	7万以上

> 方案2的APP开发，采用的是纯原生技术来开发，开发成本会比H5技术或weex混合开发要高很多，但制作的APP精美程度和操作体验也会高很多。
>
> 注意：之后每年需要续交的费用： 固定每年上交苹果费用：99美元