## App端测试

**课程大纲**

**一、App端系统架构**

**二、App发布策略**

**三、搭建App端测试环境**

**四、App端专项测试**

**五、App端性能测试**

### 一、**App端系统架构**

**APP端与Web端的系统架构区别**

**APP端：**C/S结构(客户端/服务器)，需要下载安装客户端。

**Web端：**B/S结构(浏览器/服务器)，基于浏览器访问。

**APP端测试范围:**



### 二、APP发布策略

将开发完成的移动应用程序通过特定的渠道和流程，向公众发布，使得用户可以下载、安装并使用应用程序。

**分类:**内部发布渠道，线上发布渠道。

#### 1、内部发布

在实际测试工作中，为了方便测试程序包的安装和管理，可以使用一些应用内测分发平台。如：蒲公英、Testlink等。

**步骤:**

1、开发将应用测试包上传到这些平台上

2、平台可以生成对应的二维码

3、测试直接扫码进行应用安装



#### 2、线上发布

产品测试完成后，将APP发布到应用各种平台上。

**安卓应用：**豌豆荚、应用宝、各类手机品牌商城等

**IOS应用：**主要有App store、iTools

**步骤：**

1、开发者账号注册，申请在发布平台(各种应用商店)上架

2、针对不同的发布平台，在软件包中加入对应的平台ID，上传到发布平台

3、平台审核通过后，用户即可在应用商店中下载

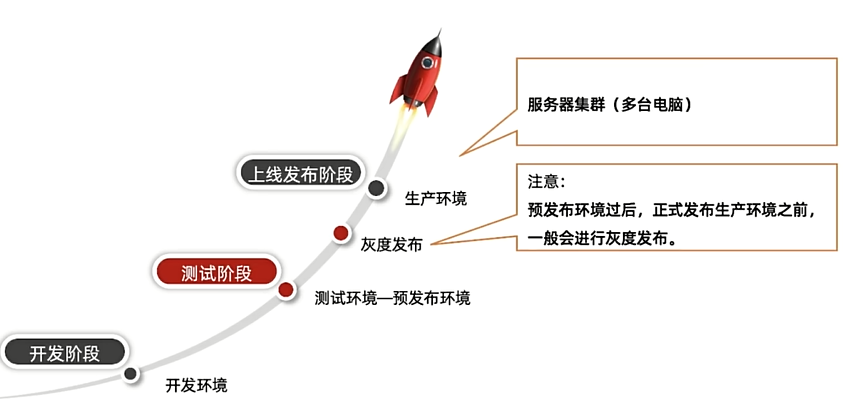
**注意：**

一般线上发布过程，由开发人员负责。

在软件包加入平台ID后，上传到发布平台时，需要测试人员验证核心的业务功能。

#### 3、发布策略

项目发布时采用的一种策略，先发布少数（1-3台）服务器，待运行稳定后再发布到所有服务器。



**总结：**

**1、APP发布方式**

内部发布：蒲公英、Testlink等

线上发布：

安卓应用：豌豆荚、应用宝、360手机助手、应用市场

IOS应用：主要有App store、iTools

**2、上线发布策略**

开发环境-->测试环境-->（灰度发布)-->生产环境

灰度发布：部分用户可用，若有异常则回滚

线上发布：所有用户可用

### 三、搭建App端测试环境

App应用运行所依赖的软、硬件环境。

**依赖：**

移动设备（手机、平板），模拟器（mumu，雷电）

App安装包（通过apk安装包进行安装或通过应用平台进行安装。）



#### 1、模拟器安装

MuMu模拟器官网：https://mumu.163.com/

雷电模拟器官网：https://www.ldmnq.com/

**步骤：**

官网下载安装包

运行安装包完成安装

异常处理：表示BIOS设计没有开启虚拟机。

解决方案：进入BIOS打开虚拟机设置即可。

进入BIOS的方式根据电脑品牌的不同，方式也不同。

以联想电脑为例：开机时连续按F12键即可进入。



#### 2、App安装

将App的apk安装包拖入模拟器中即可完成App的安装。



**MuMu模拟器中的App相关操作：**

打开APP ： 鼠标左键点击App图标

左/右滑动：鼠标左键单击后，往左/右拖拽

上/下滑动：鼠标左键单击后，往上/下拖拽

**搭建环境总结**

1、环境：apk安装包，模拟器

2、apk安装：将apk安装包拖入模拟器

### 四、App端专项测试

测试App程序在不同的移动设备上是否能持久、稳定的运行。



**App专项测试目的：**

保障主流移动设备能正常使用App应用；

不同的网络环境App应用正常使用；

不同App版本正常使用。

**App专项测试内容：**



#### 1、安装测试

**正常场景:**

在不同的操作系统版本上安装

从不同的安装渠道安装，如：App商城、手机助手、直接下载apk文件安装

不同的安装路径，如：安装到手机上、安装到SD卡上

卸载后安装

正在运行时覆盖安装

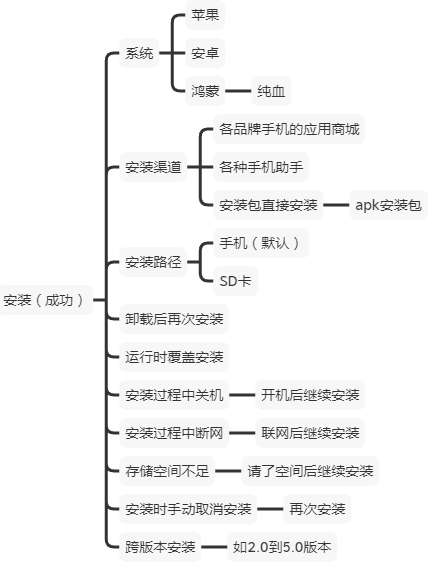
**异常场景:**

安装时出现异常（关机、断网），恢复后能否继续安装

安装时存储空间不足

安装时手动取消后再次安装

低版本覆盖安装高版本



#### 2、升级测试

**关注点:**

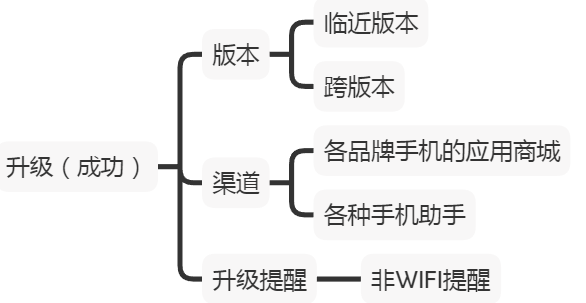
从临近版本升级

跨版本升级

不同渠道升级，如：应用商场、手机助手

升级提醒 成功，如：可不提醒、可以提示升级、强制升级

应用内升级时非WIFI提醒



#### 3、卸载测试

**关注点:**

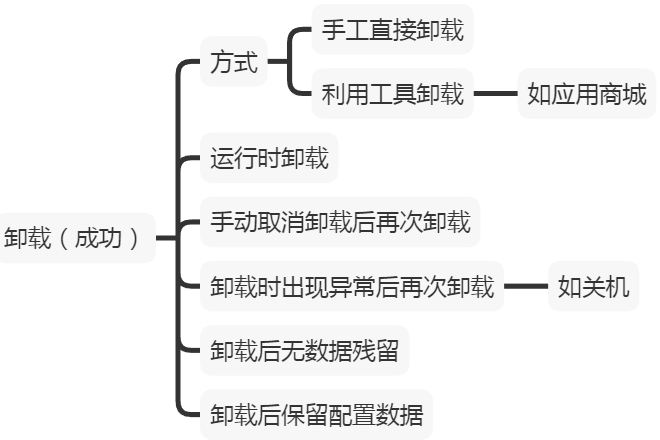
正常卸载，如：APP手动卸载、工具卸载

运行时卸载

取消卸载

卸载异常中断后卸载

卸载后无数据残留



#### 4、兼容性测试

程序能在不同的设备上运行正常。

**关注点：**

品牌型号，如：品牌、系统版本、分辨率

网络，如：2G/3G/4G/5G/WIFI

软件兼容

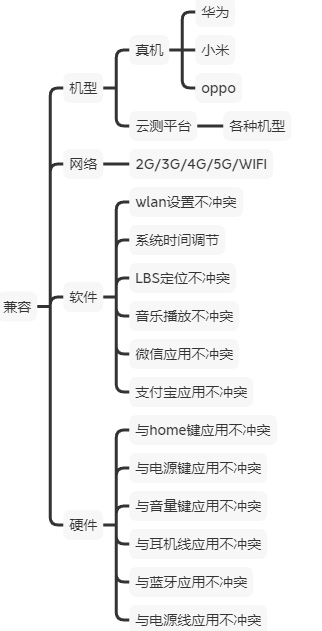
硬件兼容

**兼容性测试方式:**

1、使用公司已有的真机进行兼容性测试。

2、使用第三方的兼容性平台进行测试。

如：线上云测平台testin（收费）https://www.testin.cn/



#### 5、Push消息

Push消息是APP推送的各种通知。如∶点赞、评论、关注



**Push消息推送方式**

**Pull(拉）客户端主动获取∶**

客户端固定时间主动向服务器获取消息。

Pull方式消耗客户端和服务器资源。

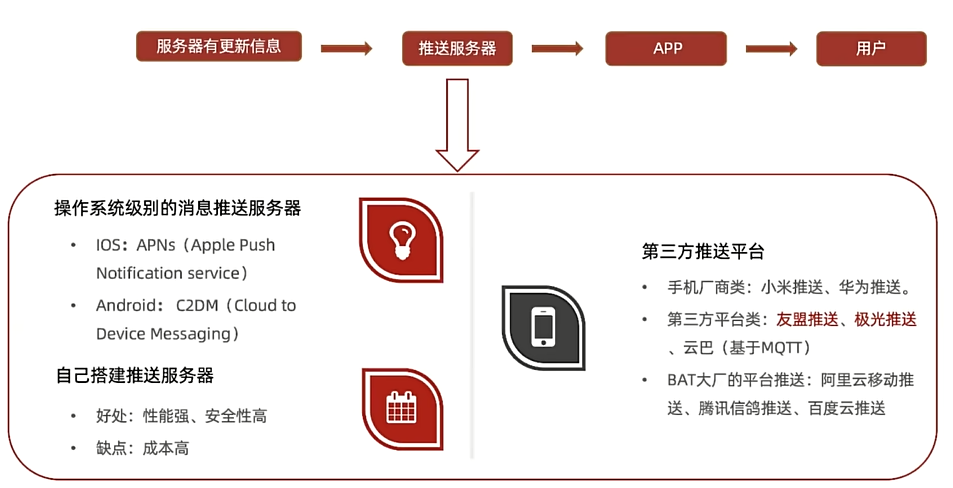
**Push(推）客户端被动接受:**

当服务器有更新消息时，主动发送到客户端。

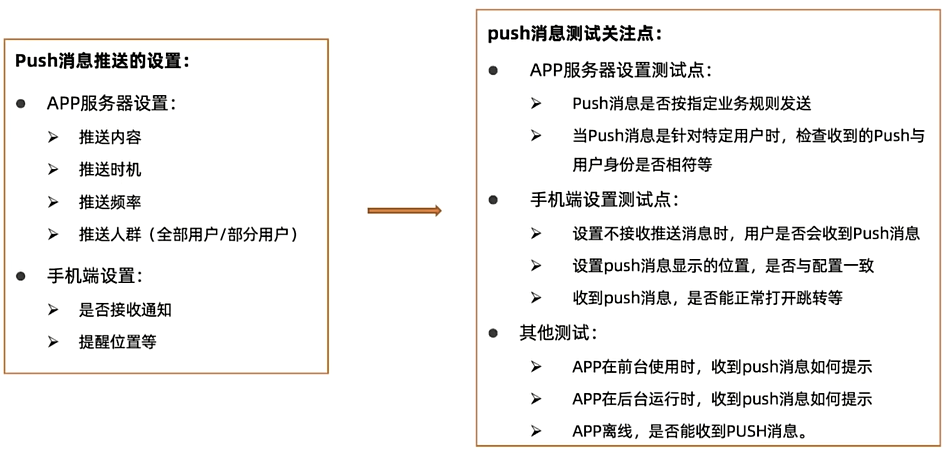
Push方式节省客户端和服务器资源。

在APP项目中，基于手机电量与流量的考虑，使用的都是push方式进行消息推送，因此又叫Push消息。

**Push消息推送流程：**



**Push消息关注点：**





#### 6、交叉测试

又叫（冲突、干扰）测试，是指一个功能正在执行过程中，另外一个事件或操作对该过程进行干扰的测试。

如：在App前台/后台运行同时接听来电或者下载文件等。

**交叉事件测试关注点:**

APP运行时接打电话;

APP运行时收发信息;

APP运行时紊看应用推送APP运行接上蓝牙设备

APP运行时接收文件弹窗提醒APP运行时旋转屏幕

APP运行时切换网络（4G、Wi-Fi) ;

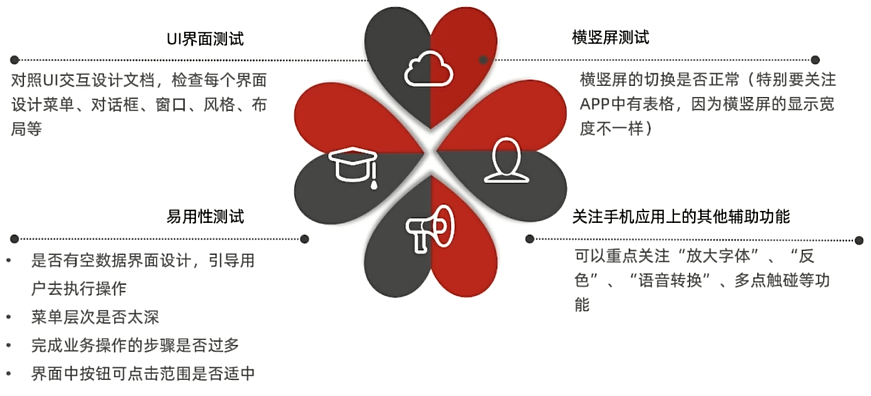
App运行时使用相机、计算器等手机自带应用;

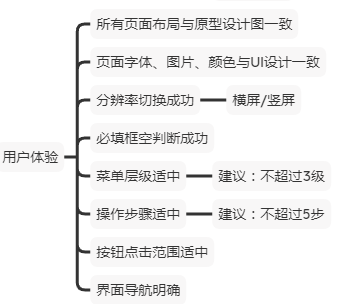
App运行时电量告警、插拔充电器。



#### 7、用户体验测试

以主观的角度去感知产品或服务的舒适、易用、友好亲切程度。





### 五、App端性能测试

测试App使用期间占用硬件资源（CPU、内存、流量、电量)使用情况。

**App性能测试关注点：**

App使用时对CPU、内存的占用情况;

App使用时流量、电量的消耗情况;

App使用时是否流畅;

App的启动时间是否过长;

App是否能长时间稳定运行等。

#### 1、SoloPi工具下载/安装

SoloPi是一个无线的Android自动化工具，具备录制回放、性能测试等功能。

**性能测试:**

能够对CPU、内存与网络环境进行限制，复现应用在性能较差、网络环境不佳场景下的表现。

**录制回放:**

能够将用户的操作记录下来，支持在各个设备上进行回放。

**一机多控:**

操作一台主机设备来控制多台从机设备，进行重复冗杂的兼容性测试，能够极大提升兼容性测试的效率。

**下载：**

https:/ /www.pgyer.com/solopi

提示：若打不开网址可输入 pgyer.com/solopi

**安装：**

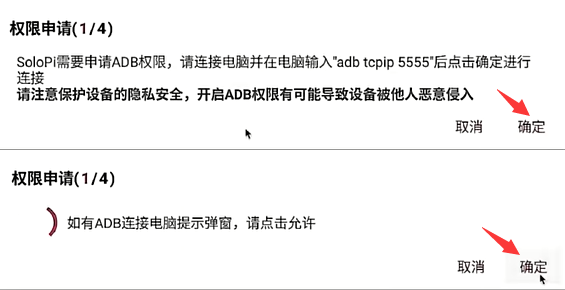
将下载好的SoloPi (APK)文件拖最至MuMu模拟器即可完成安装。

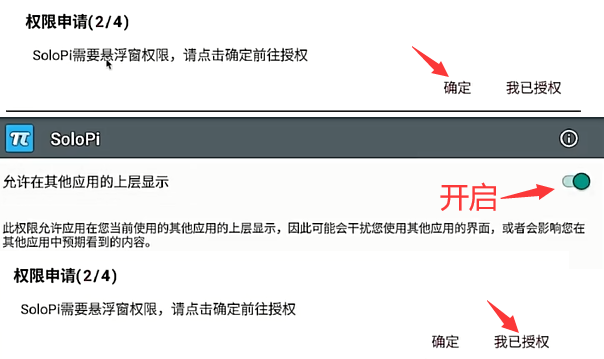
**SoloPi使用**

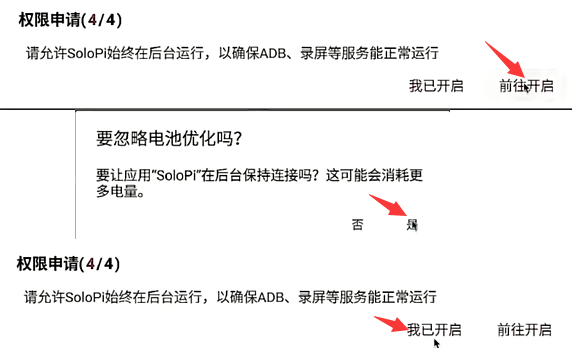
1、打开SoloPi，选择性能测试。



**注意**：第一次使用要按照提示步骤完成授权设置。







#### 2、CPU监控指标

**监控指标：CPU（全局占用 CPU** ， **应用进程 CPU** **）**

**判断标准：**不长时间占用在90%以上即可。

**CPU消耗引起的现象：**

CPU使用长时间处于90%以上：

手机发热、耗电量增加。

响应变慢

无响应(ANR)

**全局占用CPU：**

整机的CPU使用水平，即当前手机的CPU整体使用率。

Linux系统中CPU利用率分为**用户态、系统态**和**空闲态。**

用户态：表示CPU处于应用程序执行的时间；

系统态：表示系统内核执行的时间；

空闲态：表示空闲系统进程执行的时间。

**应用进程CPU：**

表示自开机以来，应用程序消耗的CPU时间的总数。

**例：性能CPU测试**

需求：测试滑动首页CPU使用率

**步骤:**

1.打开SoloPi工具，勾选CPU监控指标。

2.进入tpshop，操作上述业务。

3.查看CPU运行结果。

检查APP运行时CPU是否长时间处于90%以上。



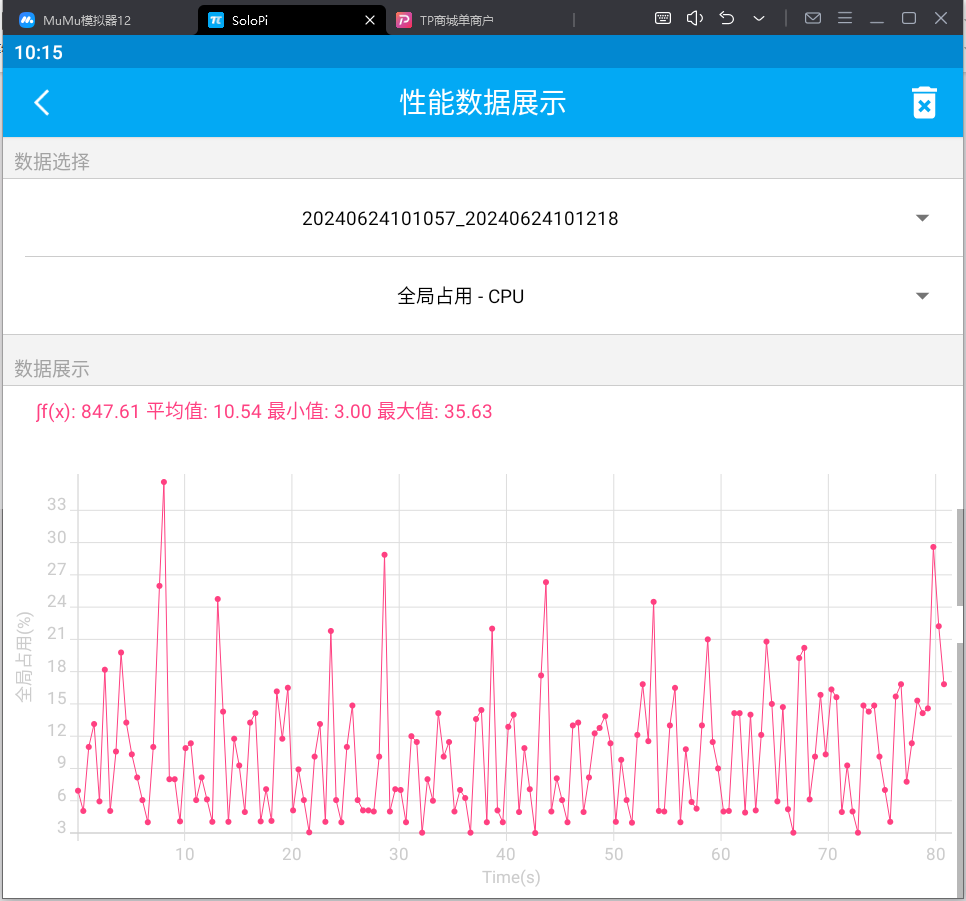










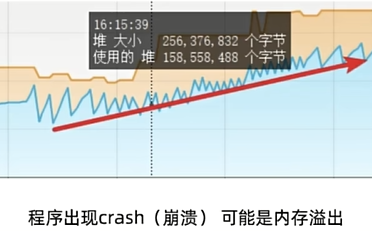


#### 3、内存监控指标

每个程序运行时都需要将代码和数据放入内存中，内存不足则程序无法正常运行。

**监控指标：内存（PSS）**

**判断标准：**PSS指标不持续（大幅）增长。



**Private dirty(私有内存)∶**

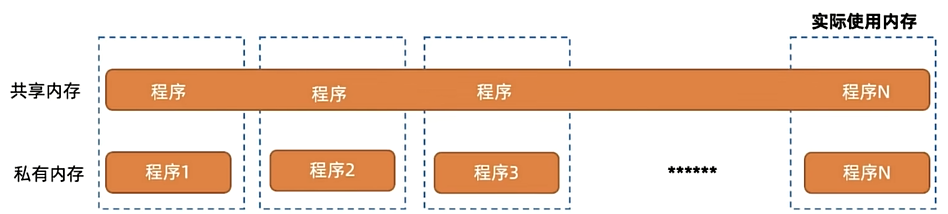
进程独占内存，也就是进程销毁时可以回收的内存容量。

**PSS（实际使用内存):**

将跨进程共享页也加入进来，进行按比例计算PSS。

这样能够比较准确的表示进程占用的实际物理内存。

程序使用时是私有内存+共享内存，所以我们测试时更多的是**关注实际使用内存（PSS）**。



**内存问题常见的现象**

**内存泄漏:**

程序在申请内存后，无法释放已申请的内存空间，一次内存泄漏危害可以忽略，但内存泄漏堆积后果很严重，无论多少内存，迟早会被占光。

内存泄漏会最终会导致内存溢出。

**内存溢出:**

指程序在申请内存时，没有足够的内存空间供其使用，出现内存溢出。

**例：性能内存测试**

需求：浏览tpshop首页平均内存消耗情况。

**步骤:**

1.打开SoloPi工具，配置内存监控；

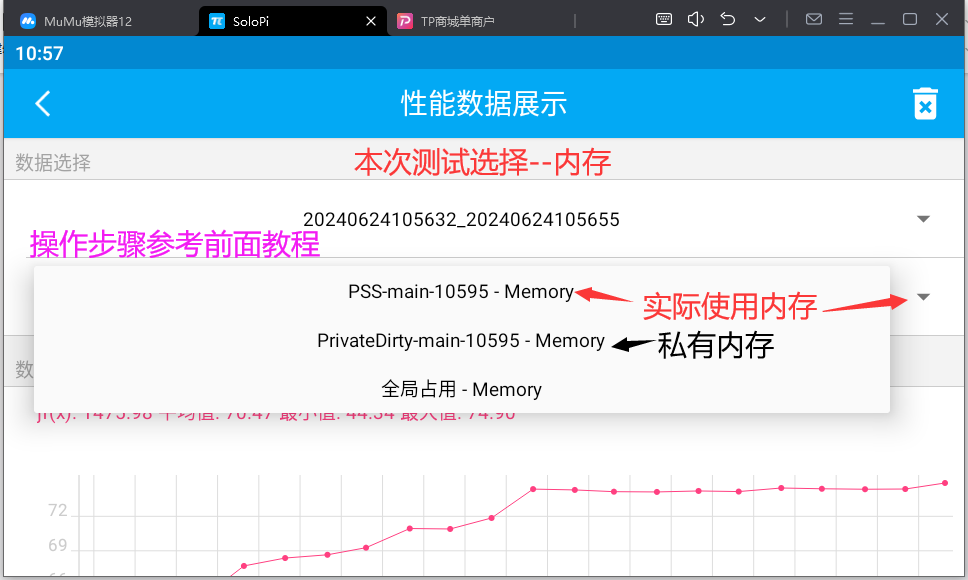
2.进入APP，操作上述业务；

**工作中建议操作2分钟以上**

3.查看内存运行结果。

检查程序实际使用的内存PSS值。

PSS数据不是大幅持续增加即可。



#### 4、流量

操作App会与服务器交换数据，流量就是指这些交互数据的总大小。

**监控指标：网络(进程上行流量-main，进程下行流量-main)**

**判断标准：**

1.与基准数据对比（基准数据来自于产品经理，或者以往数据积累）

2.横向对比，参考竞品数据。



**例：性能流量测试**

需求:打开tpshop首页，上下滑动动态20秒，获取消耗的网络流量。

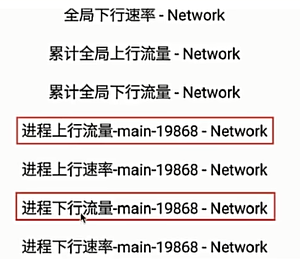
**步骤:**

1.打开SoloPi工具，勾选流量监控指标网络；

2.进入APP，操作上述业务；

3.查看流量统计结果。

注意：在模拟器中无法统计电脑流量使用情况，看进程使用流量即可。







**流量优化策略:**

数据的压缩

不同数据格式的采用

控制访问的频次

只获取必要的数据

缓存机制

针对不同的网络类型设置不同的访问策略

#### 5、电量

APP应用使用时对电池电量的平均消耗。

**监控指标：电池（平均功率）**

**判断标准：**

1.与基准数据对比（基准数据来自于产品经理，或者以往数据积累）

2.横向对比，参考竞品数据。

**常见的耗电量大的场景:**定位，网络传输，屏幕亮度，......

**电量的监控方法**

**系统自带接口：**

最新的IOS和Android系统内置的Setting里可以查看各个APP的电池消耗；但该方案不能检测固定某一时间段内的电池精准消耗。

**硬件检测：**

通过硬件可以精准地获得应用的电量消耗；但该方案测试时需要拆机，成本太高比较麻烦。

**软件工具检测：**

通过第三方的软件来获取应用的电量消耗。如: 360省电王、SoloPi

该方案取决于第三方软件的计算准确性。

例：性能电量测试

需求:打开Tpshop，进入首页，上下滑动动态2分钟，获取消耗的电量。

步骤:

1.打开SoloPi工具，勾选电量监控指标:电池

2.进入APP，操作上述业务

3.查看电量统计结果。

提示：模拟器没有电池，无法获取电量数据；但不妨碍我们学习如何进行电量测试的手段。



#### 6、流畅度

动画播放或图片切换的流畅性。

动画其实是由一张张画出来的图片连贯执行产生的效果，当一张张独立的图片切换速度足够快的时候，会欺骗我们的眼睛，以为这是连续的动作。

反之，当图片切换不够快时就会被人眼看穿，反馈给用户的就是卡顿现象。



**监控指标：帧率（FPS）**

**判断标准：**建议24帧以上（或者公司规定）

**FPS:** GPU在一秒内绘制的帧数。(简单理解为一秒内呈现给用户的图片数)，**FPS值越高画面越流畅**。

**流畅度问题产生的影响:**

想要让大脑觉得动作是连续的，至少需要每秒**10-12帧**；

想达到流畅的效果，至少需要每秒**24帧**；

60帧每秒的流畅度是最佳的，我们的目标就是让程序的流畅度能接近**60帧**每秒。

例：性能流畅度测试案例

需求:打开tpshop，进入首页，上下滑动动态2分钟，记录FPS值。

步骤:

1.打开SoloPi工具，勾选：帧率；

2.进入APP，操作上述业务；

3.查看流畅度运行结果。

**注意：**

当页面多为**静态时**，FPS值很**小**是正常的；

页面数据多为**动态加载时**，FPS值比**较大**。



#### 7、启动速度

从启动App到主页面加载完成的速度。

**监控指标：启动耗时计算**

**判断标准：**

1.与基准数据对比（基准数据来自于产品经理，或者以往数据积累）

2.横向对比，参考竞品数据。

**App启动分类：**

**冷**启动：启动App进程，这种启动方式叫做冷启动。

**热**启动：将App从后台置于前台。

**例：性能启动时间测试**

需求:分别获取tpshop冷启动和热启动时间

**步骤:**

1.打开SoloPi工具，勾选：启动耗时计算

2.进入APP，操作上述业务

3.查看启动耗时计算运行结果。

注意：第一次使用需要下载相关插件。











**小结：**

**1、App端系统架构是什么样的？与Web有什么区别？**

**2、App端通常都是怎么发布的？**

**3、什么是灰度发布？**

**4、App专项测试都测什么内容？**

**5、App性能测试用什么工具做的？都测什么内容？**

**6、App端与Web端测试的区别是什么？**