江西理工大學

# 本科毕业设计(论文)

题　目：基于 Hybrid 技术的多人在线摇一摇游戏

专题题目：

学　院：软件学院

专　业：软件工程　　班　级：12 软件嵌入式 1 班

学　号：12221124　　学　生：黄睿

指导教师：　　职称：

指导教师：　　职称：

时间：2016 年 4 月 30 日

# 江 西 理 工 大 学

# 本 科 毕 业 设 计（论文）任 务 书

软件**学院** 软件工程 **专业** 2012 **级**（2016 **届**）嵌入式 1 **班** 12221124 **黄睿**

**题　目：基于 Hybrid 技术的多人在线摇一摇游戏**

**专题题目**（若无专题则不填）：

**原始依据**(包括设计（论文）的工作基础、研究条件、应用环境、工作目的等)：

移动互联网的热潮刮起后，众多公司前赴后继的进入。但是很快发现移动应用的开发人员太少，所以导致疯狂的人才争夺。市场机制下移动应用开发人才的待遇扶摇直上，最终变成众多企业无法负担养一个具备跨平台开发能力的专业移动应用开发团队。而 HTML5 的出现让 Web App 露出曙光，HTML5 开发移动应用的跨平台和廉价优势让众多想进入移动互联网领域的公司开始心动。可是当下基于 HTML5 的 Web App 更是雾里看花，在用户入口习惯、分发渠道和应用体验这三个核心问题没解决之前，Web App 也很难得以爆发。正是在这样是机缘巧合下，基于 HTML5 低成本跨平台开发优势又兼具 Native App 特质的 Hybrid App 技术杀入混战，并且很快吸引了众人的目光。大幅的降低了移动应用的开发成本，可以通过现有应用商店模式发行，在用户桌面形成独立入口等等这些，让 Hybrid App 成为解决移动应用开发困境不错的选择，也成为现阶段 Web App 的代言人。Hybrid App 像刺客一样，在 Native App 和 Web App 混战之时，偶然间的在移动应用开发领域占有了一席之地。

另一方面，用户在使用 App 时的参与感需求越来越强，App 需要大量的活动运营来与用户形成互动。

目前，Hybrid 技术在国内还属于新兴技术，在应用和实践上都多以探索为主。

**主要内容和要求：**（包括设计（研究）内容、主要指标与技术参数，并根据课题性质对学生提出具体要求）：

使用 Native 的技术去实现一个逻辑，往往需要写两份代码，并且难以满足灵活多变的需求。同时通过客户端发布新版本，由于无法强制用户升级，用户往往会分布在多个版本，碎片化使得代码更加难以维护。而使用 HTML5 相关技术具备极强动态性，可以在服务端随时更新代码，但是无法充分利用移动终端的特性打造真正无线化的应用，且在很多场景下无法提供顺畅的体验。

本课题的研究目标就是致力于解决上述问题，并实现一个具有代表性的互动游戏。

整个游戏包含：

1. 客户端 Hybrid 容器，输出传感器等客户端特有能力。

2. HTML5 游戏逻辑代码。
3. Node 游戏服务端，为多终端游戏互动提供支持，主要工作是信息转发、同步。
4. Node 游戏管理后台。

主要指标：

1. 游戏流程正常
2. 游戏后台正常
3. 服务端支持的并发访问量（QPS）
4. 客户端兼容性

**日程安排：**

| 序号 | 各阶段工作内容 | 时间安排 |
|---|---|---|
| 1 | 毕业设计开题报告 | 2016 年 3 月 24 日前完成 |
| 2 | 需求分析 | 第一、二周 |
| 3 | 概要设计 | 第三、四周 |
| 4 | 详细设计 | 第五、六周 |
| 5 | 系统设计 | 第七、八周 |
| 6 | 系统分发 | 第九、十周 |
| 7 | 毕业设计论文整理 | 第十一、十二周 |
| 8 | 提交毕业论文 | 2016 年 5 月 1 日前 |

**指导教师**（签字）：

年　　月　　日

注：本表可自主延伸，各专业根据需要调整。

# 江 西 理 工 大 学

# 本 科 毕 业 设 计（论文）开 题 报 告

软件**学院** 软件工程 **专业** 2012 **级**（2016 **届**）嵌入式 1 **班** 12221124 **黄睿**

**题　目：基于 Hybrid 技术的多人在线摇一摇游戏**

**专题题目**（若无专题则不填）：

**本课题来源及研究现状：**

- **课题来源：**

移动互联网的热潮刮起后，众多公司前赴后继的进入。但是很快发现移动应用的开发人员太少，所以导致疯狂的人才争夺。市场机制下移动应用开发人才的待遇扶摇直上，最终变成众多企业无法负担养一个具备跨平台开发能力的专业移动应用开发团队。而 HTML5 的出现让 Web App 露出曙光，HTML5 开发移动应用的跨平台和廉价优势让众多想进入移动互联网领域的公司开始心动。可是当下基于 HTML5 的 Web App 更是雾里看花，在用户入口习惯、分发渠道和应用体验这三个核心问题没解决之前，Web App 也很难得以爆发。

另一方面，用户越来越重视使用 App 过程中的参与感。目前已经有大量使用互动项目取得巨大成功的案例。例如，2015 年双十一期间手机淘宝的红包雨和搜索“秋裤”等密令出现的红包都是属于互动项目；2015 年春节期间支付宝的集五福咻一咻也是属于互动项目。而且此类互动还不仅仅局限于线上，线下也能起到不错的效果。但是互动项目往往对灵活性、时效性具有较高的要求，传统的移动端开发技术难以解决此类问题。

正是在这样是机缘巧合下，基于 HTML5 低成本跨平台开发优势又兼具 Native App 特质的 Hybrid 解决方案杀入混战，并且很快吸引了众人的目光。大幅的降低了移动应用的开发成本，可以通过现有应用商店模式发行，在用户桌面形成独立入口等等这些，让 Hybrid App 成为解决移动应用开发困境不错的选择。Hybrid App 像刺客一样，在 Native App 和 Web App 混战之时，偶然间的在移动应用开发领域占有了一席之地。

- **研究现状：**

目前，Hybrid 相关技术在国外内仍处于高速发展状态，且主要以 Apache Cordova 进行可视化改造，利用页面可视化快速搭建来降低开发成本。而 Cordova 却没有针对用户体验给出一个较好的解决方案。

可视化开发虽然降低了一定开发门槛，但是面对较为复杂的单页面应用时，会发现难以利用现有的前端技术去解决问题。

**课题研究目标、内容、方法和手段：**

- **研究目标：**

本课题主要针对一个典型的互动场景“摇一摇”给出一个轻量、高度可拓展的Hybrid方案，并保留现有的前端开发技术，其开发门槛通过前端脚手架、自动化构建工具来降低开发门槛。

- **研究内容：**

以Android平台实现高度结构化、可重用、高性能的Hybrid容器，此容器以SDK方式发布。SDK提供HTML5页面嵌入基础能力、JS-Native双向通道、Native插件机制、页面资源缓存机制、拦截器机制和页面跳转工具集。

基于Web技术体系开发多人在线的“摇一摇”互动游戏。

同时为游戏建立相应的后台管理系统，提供创建游戏、修改游戏等操作。

- **研究方法：**

以Android作为移动终端的研究平台。

Hybrid容器直接基于Android Framework中的WebKit改造，除了要支持基本JSBridge，还需要针对性解决网络传输性能低下、页面体验差问题。

前端部分利用现有技术体系完成游戏业务流程、以及界面展示等问题。但最终需要运行在Hybrid容器中，因为其中会使用到JSBridge，常规的浏览器或者其他App的Webkit无法支持此种方式的调用。

基于Node.js实现高性能的实时服务器为游戏提供消息传递、逻辑处理。随后，通过其Express搭建管理后台来控制实时服务器的行为。

## 设计（论文）提纲及进度安排：

- **设计（论文）提纲：**

第一章　绪论

第二章　相关技术与方法

第三章　系统分析

第四章　系统设计

第五章 系统实现

第六章　系统测试

第七章　总结

- **进度安排：**

1、毕业设计开题2015年12月底之前前完成

2、第一、二周 需求分析

3、第三、四周 概要设计

4、第五、六周 详细设计

5、第七、八、九周 系统开发

6、第十周 系统测试

7、第十一、十二周 毕业设计论文整理

## 主要参考文献和书目：


[1] JavaScript 使用详解[M],机械工业出版社,7-111-06897-1,TP312JA

[2] Web 专家工作室系列-JavaScript 实战解析[M],机械工业出社 ,TP312JA

[3] JavaScript 程序设计及应用[M],西北工业大学出版社, ,TP312JA

[4] JavaScript 1.1 开发指南[M],Tatters,清华大学出版社,7-302-02856-7,TP312JA

[5] JavaScript 编程指南[M],王炜,电子工业出版社,7-5053-5187-7,TP312JA

[6] 萨师煊，王珊.数据库系统概论(第三版)[M].北京：高等教育版,1998.

[7] 冯明.基于混合模式(Hybrid App)移动终端设计的方法[J].数字技术与应用,2015,第 4 期:148-149

[8] 李刚.疯狂 Android 讲义[M].北京：电子工业出版社，2013.

[9] 杨丰盛.Android 技术内幕[M].北京：计协工业出版社，2011.

[10] 杨云君.Android 的设计与实现[M].北京：机械工业出版社，2013.


## 指导教师审核意见：

**指导教师**（签字）：　　年　　月　　日

注：本表可自主延伸

# 摘　要

混合型应用（Hybrid App）是指以移动客户端为主体，搭载移动端的HTML5页面，充分发挥客户端的可定制性和HTML5的灵活性。针对像“摇一摇”这样的互动性项目，往往具有时效性需要快速发布的要求，采用Hybrid技术恰好能解决其痛点。本文研究的就是将Hybrid技术应用于此类互动性项目的解决方案。

首先以Android为平台研发了一个轻量Hybrid容器，打通了JavaScript和Native的运行环境，使得业务可以用HTML5描述却能利用Native的能力。接着围绕它，在HTML5的构建上采用了内联等方式优化了连接数，在Native环境通过预下载和缓存优化了资源下载过程，大幅度改善了HTML5页面在客户端页面的体验。最后将这些优化与Hybrid容器放在一起以SDK形式发布。

接着在Hybrid容器上实现了一个极具代表性的互动项目多人在线的“摇一摇”游戏，并为控制游戏建立了其管理后台。

**关键词：**

Hybrid；互动游戏；JavaScript；Android；

# ABSTRACT

Hybrid applications (Hybrid App) refers to the mobile client as the main body, equipped with mobile side HTML5 page, give full play to the client customizability and flexibility of HTML5. For like "Shake" This interactive projects, often with the timeliness required rapid release requirements, using Hybrid technology just to solve their pain points. This study is to Hybrid technology applied to such solutions interactive projects.

First Android-platform Hybrid developed a lightweight container, opened the JavaScript and Native operating environment, so that services can be able to take advantage of the ability to use HTML5 Native description. Then around it, build HTML5 on the use of inline and other ways to optimize the number of connections in the Native environment by downloading and caching pre-optimized resource download process, greatly improving the HTML5 page on the client side of the page experience. Finally, the optimization of these containers together with the Hybrid released in SDK form.

Then on Hybrid container implements a highly representative of interactive projects multiplayer online "Shake" game, control the game and for the establishment of its management background.

**Key words:**

Hybrid；Interactive Game；JavaScript；Android；

# 目 录

# 第一章　绪论

## 1.1 研究背景和意义

移动互联网的热潮刮起后，众多公司前赴后继的进入。但是很快发现移动应用的开发人员太少，所以导致疯狂的人才争夺。市场机制下移动应用开发人才的待遇扶摇直上，最终变成众多企业无法负担养一个具备跨平台开发能力的专业移动应用开发团队。而HTML5的出现让Web App露出曙光，HTML5开发移动应用的跨平台和廉价优势让众多想进入移动互联网领域的公司开始心动。可是当下基于HTML5的Web App更是雾里看花，在用户入口习惯、分发渠道和应用体验这三个核心问题没解决之前，Web App也很难得以爆发。

另一方面，用户越来越重视使用App过程中的参与感。目前已经有大量使用互动项目取得巨大成功的案例。例如，2015年双十一期间手机淘宝的红包雨和搜索“秋裤”等密令出现的红包都是属于互动项目；2015年春节期间支付宝的集五福咻一咻也是属于互动项目。而且此类互动还不仅仅局限于线上，线下也能起到不错的效果。但是互动项目往往对灵活性、时效性具有较高的要求，传统的移动端开发技术难以解决此类问题。

正是在这样是机缘巧合下，基于HTML5低成本跨平台开发优势又兼具Native App特质的Hybrid解决方案杀入混战，并且很快吸引了众人的目光。大幅的降低了移动应用的开发成本，可以通过现有应用商店模式发行，在用户桌面形成独立入口等等这些，让Hybrid App成为解决移动应用开发困境不错的选择。Hybrid App像刺客一样，在Native App和Web App混战之时，偶然间的在移动应用开发领域占有了一席之地。

目前，Hybrid相关技术在国外内仍处于高速发展状态，且主要以Apache Cordova进行可视化改造，利用页面可视化快速搭建来降低开发成本。而Cordova却没有针对用户体验给出一个较好的解决方案。

可视化开发虽然降低了一定开发门槛，但是面对较为复杂的单页面应用时，会发现难以利用现有的前端技术去解决问题。

## 1.2 研究目标

本课题主要针对一个典型的互动场景“摇一摇”给出一个轻量、高度可拓展的Hybrid方案，并保留现有的前端开发技术，其开发门槛通过前端脚手架、自动化构建工具来降低开发门槛。

以 Android 平台实现高度结构化、可重用、高性能的 Hybrid 容器，此容器以 SDK 方式发布。SDK 提供 HTML5 页面嵌入基础能力、JS-Native 双向通道、Native 插件机制、页面资源缓存机制、拦截器机制和页面跳转工具集。

最后基于 Web 技术体系开发多人在线的“摇一摇”互动游戏，使其运行在 Hybrid 容器中，同时为游戏建立相应的后台管理系统，提供创建游戏、修改游戏等操作。

## 1.3 论文结构

第一章，主要介绍 Hybrid 技术的现状以及移动互联网对互动项目的需求。

第二章，介绍本文研究过程中使用到的其他技术和工具。

第三章，对系统的可行性以及需求进行了介绍。

第四章，描述了系统的整体架构以及关键业务流程。

第五章，对具体业务的技术实现方式进行了介绍。

第六章，对系统整体进行了测试并给出了测试报告。

第七章，对整个项目研究进行了总结。

## 1.4 本章小结

Hybrid 是一个在市场有着强烈需求下诞生的方案，Hybrid 技术的发展能有效的解决目前移动端存在的痛点。

# 第二章　相关技术与方法

## 2.1 架构概述

系统整体采用 C/S 架构。C 为用户终端，提供一个类似 Browser 的容器，但不影响客户端本身的业务逻辑。S 为服务端，主要由基础后台和消息中心构成。

## 2.2 关键技术简介

### 2.2.1 HTML5/CSS3/JavaScript ES2015

HTML5 是顺应移动互联网的发展而在 HTML 上拓展的新标准。新的标准对移动终端显得更加友好，其灵活性、信息传递能力都得到了极大的增强。例如，对语义的增强赋予了网页更好的意义和结构，本地存储特性能使有效的驻留数据在本地。

CSS3 则提供了更多的风格，使得构建一个接近于 Native 的页面变得容易。加上现在主流终端的设计风格向扁平化转移，很多 HTML5 页面和 Native 已经几乎一样。

JavaScript ES2015 提供了很多能提升开发效率的语法糖，对工程的模块化构建也给出了解决方案，回调函数无限嵌套的问题也得益于 Promise 的出现而被解决了。

但是事实上，在移动终端上这些标准都没有完全被实现，但已经被实现的大部分特性已经足够用来构建一个体验不错的页面了。

### 2.2.2 lib-flexible

阿里巴巴针对移动端基于 rem 单位的轻量自适应方案，目前在 github 开源。能够解决移动端屏幕的适配问题，并提供了相应的栅格布局机制。

### 2.2.3 Java

Java 是极为流行的面向对象设计语言，为业务逻辑抽象提供了有效的支持，同时具备卓越的通用性和安全性。Google 将其作为 Android SDK 的一等语言，目前 Android 上的两个虚拟机分别是 Dalvik 和 ART 虚拟机对 Java 提供了良好的支持，并且针对移动终端的特性做了大量优化。

目前，人才市场对 Java 相关储备也是相对充足的，这方面无形之间降低了移动端开发的成本。

### 2.2.4 Android Framework

Android Framework 是 Android 系统对开发者暴露的组件集合，其意义在于屏蔽系统底层的复杂流程同时提供良好的兼容性。Framework 针对开发者提供了对应的 SDK 和 NDK，既可以通过 SDK 使用 Java 编写复杂业务逻辑，也可以通过 NDK 使用 C/C++定制高效的底层机制。

框架提供了电话服务、视图管理、资源管理、活动管理、包管理、网络管理等支持。

#### 2.2.4.1 WebKit

WebKit 是 Android Framework 提供的网页展示套件，可以在简单的在原生页面嵌入一个浏览器页面。

但是嵌入的浏览器页面，获得的是浏览器的体验，而不是客户端的原生体验。

同时，在网页中也难以利用客户端的特性。

#### 2.2.4.2 SensorManager

SensorManager 是 Android Framew 提供的访问硬件传感器的服务。

使用 SensorManager 可以设置硬件传感器的值。

### 2.2.5 Node.js

Node.js 是一个基于 Google V8 引擎的运行时平台，可以方便的搭建网络应用，并且拥有着非常活跃的社区。其单线程执行、事件驱动的异步执行模型是一大亮点。这种执行模型大幅度减少了大量线程造成的内存开销和上文切换开销。

JavaScript 运行于服务端同时也打破了目前前端和后端的划分，使得前端开发工程师也能介入到后端呈现层，大幅提升开发效率。

### 2.2.6 Socket.io

Socket.io 是一个实现了基于事件的实时双向通信网络库。

在支持 WebSocket 的情况下会直接使用 WebSocket 作为长连接通信。如果无法使用 WebSocket 则会降级为轮询的方式保证可用性。

### 2.2.7 Express

基于 Node.js 的 web 快速、极简开发框架。在 Node.js 基础上提供了一系列 Web 所需的基本功能，如 Session、路由、中间件机制、模板引擎支持等。

## 2.3 开发工具

### 2.3.1 Android Studio

Android Studio 是 Google 推出的最新 Android 集成开发环境。

Android Studio 基于 Intellij IDEA，提供了丰富的 Android 开发插件。能有效的支持基于 Gradle 构建、Android 源码查看、代码静态扫描分析（包括 Lint、常见漏洞分析、依赖分析）、快速重构、ProGuard 和应用签名、可视化布局编辑器、性能分析等。

相比于老牌的 Eclipse+ADT 开发组合，Android Studio 更加智能、也为 Android 开发做了更多的定制，极大的提高了开发效率。

### 2.3.2 Gradle

Gradle 是基于 Groovy 的特定领域语言（DSL）的自动化构建工具。

使用 android-gradle-plugin 可以对 Android 打包流程提供完整的构建支持。其中主要包括依赖管理、Manifest 配置及合并、aapt 打包资源生成 R 文件、aidl 生成 java 文件、编译源文件、生成 Dex 文件、混淆、签名、生成 Apk 文件。

相比其他自动化构建，更加优雅的处理了 Android 复杂的打包流程，在多模块开发时具备极大的优势。（大多数外部依赖，可以通过 Gradle 依赖配置在中心仓库中获取。

### 2.3.3 Sublime Text

Sublime Text 能对 Node.js 进行语法高亮、自动完成、多窗口，同时拥有代码片段功能，可以将常用的代码片段保存，在需要时快速生成。其轻量、高效的特性非常适合前端开发。

### 2.3.4 NPM

NPM(Node Package Manager)是针对 Node.js 的包管理工具。

能针对 Node.js 项目进行依赖管理。大多数模块可以通过 NPM 安装，从而减少开发环境配置的工作量。

### 2.3.5 Gulp

Gulp 是针对 Node.js 的自动化构建工具，其丰富的插件可以快速创建出高质量的工程。本次有使用到 gulp-concat 文件合并，gulp-uglify 代码混淆。

### 2.3.6 Git

Git 是目前最流行的版本管理系统，能有效的支持项目开发。
开发过程中可以利用 git 快速迭代，在发生问题时及时回滚。同时提供高效的历史查看以及变更记录。

## 2.4 本章小结

目前有大量的工具和中间件、框架支持此类项目的开发，能将项目的关注点大幅度缩小，做到高度关注业务以及为业务定制。

# 第三章　系统分析

## 3.1 可行性分析

### 3.1.1 技术可行性

Android Framework 提供的 WebKit 组件可以提供基础的支持，通过拦截 js 事件可以传输字符串，从而能够利用自定义协议定制规范的接口。

HTML5 新特性也已经被主流移动终端支持。同时，HTML5 主要来自于移动网络下的传输耗时，能够通过合并请求、缓存资源等能有效的改善体验。

Socket.io 是 Node.js 实现的的全双工实时通信网络库，能以长连接的形式与游戏服务器进行高效通信，在 HTML5 中通过相关接口获取手机状态后，即可通过此通道向服务器传输信息。

Express 是基于 Node.js 的 Web 框架，基于它能快速搭建管理游戏服务器的后台页面。

以上均为自由或开放源码软件，因此用于完成游戏系统是完全可行的。

### 3.1.2 经济可行性

互动性质项目的特点是具有时效性，运营时主要需要 CDN 用于分发 HTML5 页面，云服务器 ECS 用于游戏逻辑处理。例如选择阿里云按量计费，目前价格为 0.36 元/GB，云服务器 ECS 按量计费为 0.395 元/每小时。

作为一场活动，从经济上看是完全可行的。

### 3.1.3 操作可行性

项目完成后，对于植入 SDK 的客户端应用能够直接扫码开始游戏，游戏可以独立于客户端发布。并且 HTML5 中的通用资源文件会提前下载到客户端中，从用户体验上看页面打开速度是非常快的。

进入游戏后，通过获取客户端的登陆信息能够快速的进入游戏。游戏过程的主要交互即使在给出游戏开始后，用户通过摇动手机，待时间到达后，给出排名作为结果。

整个过程交互简单，对用户没有过强的认知要求，因此从操作上看是完全可行的。

## 3.2 需求分析

### 3.2.1 系统总体需求

表 3-1　质量要求

| 主要质量属性 | 详细要求 |
| --- | --- |
| 正确性 | 给定输入，要给出合法的输出，即能够正确完成预期功能。 |
| 健壮性 | 具有较高的容错能力和恢复能力。 |
| 性能效率 | 帧率应在 50 以上，wifi 和 4G 网络环境下 2s 内完成页面首屏渲染，3G 网络下 2s 内收到响应首包，2G 网络下 2s 内完成连接建立。 |
| 易用性 | 易理解性：软件研制过程中形成的所有文档语言简练、前后一致、也易于理解。 |
| 安全性 | 防止软件受到意外或蓄意的存取、使用、修改、毁坏或泄密的软件属性。 |
| 可扩展性 | 能方便的进行二次开发，满足对功能的扩展或提高并能提高相应的安全机制 |
| 兼容性 | 不易与其他软件起冲突 |
| 动态性 | 业务代码不依赖于客户端，可动态变更 |

### 3.2.2 用例图分析

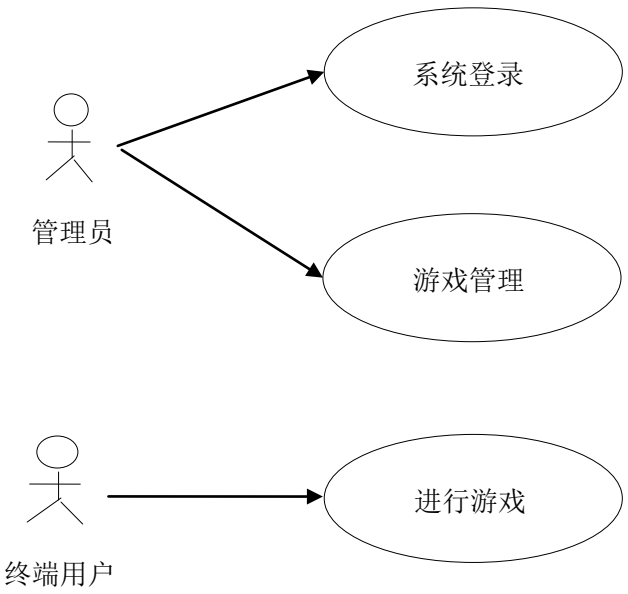


图 3-1　系统用例图

系统整体用例有系统登录、游戏管理、进行游戏，具体说明见表 3-2 至表 3-4：

表 3-2　系统登录

| 用例编号 | UC01 | |
|---|---|---|
| 用例名称 | 系统登录 | |
| 用例概述 | 管理员通过此用例登录系统 | |
| 主参与者 | 管理员 | |
| 前置条件 | 无 | |
| 基本事件流 | 步骤 | 活动 |
| | A1 | 输入正确账户名密码登录 |
| 扩展事件流 | 1a | 用户名不存在，返回提示信息 |
| | 1b | 密码错误，返回提示信息 |

表 3-3　游戏管理

| 用例编号 | UC02 | |
|---|---|---|
| 用例名称 | 游戏管理 | |
| 用例概述 | 管理员通过此用例管理系统中的游戏 | |
| 主参与者 | 管理员 | |
| 前置条件 | 管理员身份正确登录 | |
| 基本事件流 | 步骤 | 活动 |
| | A1 | 选择需要的操作，增加、修改、删除游戏 |
| | A2 | 输入游戏信息 |
| | A3 | 保存输入的游戏信息 |
| 扩展事件流 | 1a | 游戏不存在，返回提示信息 |
| | 1b | 游戏信息非法，返回提示信息 |

表 3-4　进行游戏

| 用例编号 | UC03 | |
|---|---|---|
| 用例名称 | 游戏管理 | |
| 用例概述 | 管理员通过此用例管理系统中的游戏 | |
| 主参与者 | 终端用户 | |
| 前置条件 | 客户端已登录，且能取得用户名 | |
| 基本事件流 | 步骤 | 活动 |
| | A1 | 在规定时间内不断摇动手机 |
| | A2 | 给出排名作为结果 |
| | 1b | 无网络，返回提示信息 |

## 3.3 本章小结

系统从技术、经济和操作三个角度都被证实是可行。目标用户明确，并从用户出发列出了需求，同时为系统需求给出了具体指标。

# 第四章　系统设计

## 4.1 系统全局架构

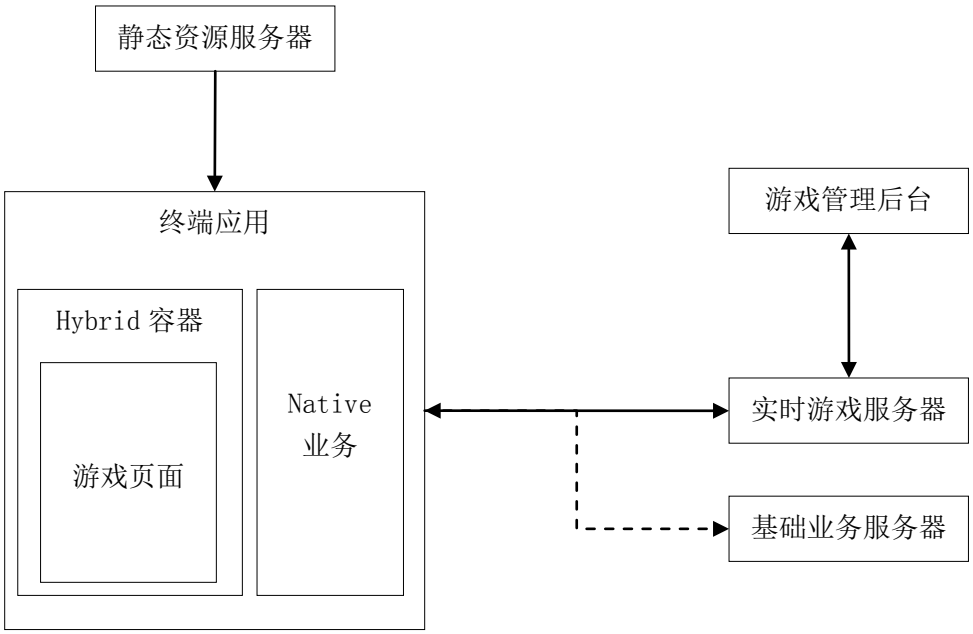


图 4-1　系统全局架构

在客户端中集成 Hybrid 容器 SDK，游戏页面发布在静态资源服务器，客户端可以通过配置来开放入口或者让用户以扫码访问访问游戏页面，从而使游戏逻辑脱离客户端基础业务逻辑存在。

游戏页面与实时游戏服务器以高速全双工通道通信保证游戏实时性。

游戏后台则可以控制实时游戏服务器的各项参数，从而达到控制游戏的目的。

基础业务服务器是客户端原有的其他业务服务器，主要为其他业务提供支持，同时客户端的登陆也发生在这里。

## 4.2 Hybrid 容器

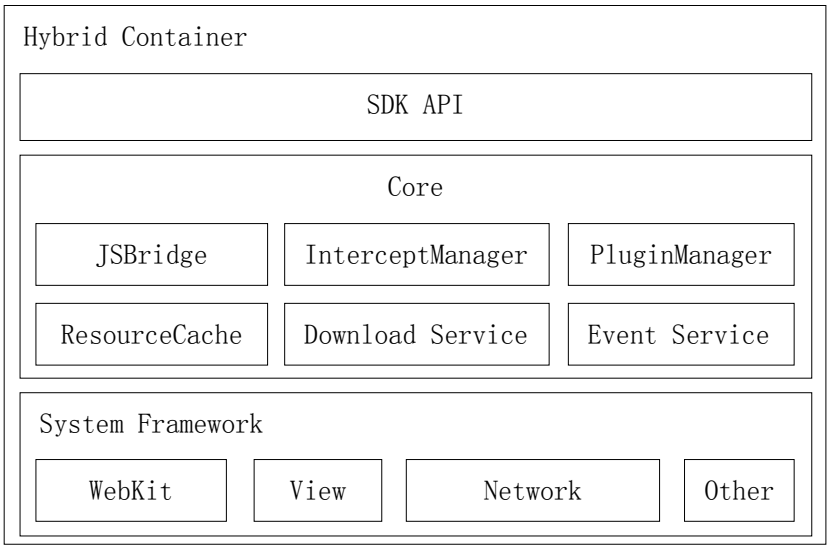


图 4-2　Hybrid 容器架构

SDK API 层主要提供对容器的配置，以及功能定制，支持自定义插件、替换 Download Module 实现等功能。

Core 层是容器内部的必备机制。其中 JSBridge 提供基础的双向通道，InterceptManager 用于拦截请求，PluginManager 提供业务暴露自身逻辑接口的能力，ResourceCache 通过预下载缓存静态资源来加速页面访问，Download Service 提供一个可支持替换实现的下载功能，Event Service 提供一套事件机制。

System Framework 层是外部依赖，利用系统的基础组件能快速实现 Core 层。

## 4.3 管理后台

### 4.3.1 系统登录

User 类是一个 POJO 类，用于记录系统的用户信息。

表 4-1　User 类功能表

| 类功能描述 | 处理和用户相关的业务操作 | | |
|---|---|---|---|
| 类属性 | | | |
| 类型 | 名称 | 描述 | 备注 |
| String | ID | 用户 ID | 系统中用户的唯一标示，由系统自动生成 |
| String | Name | 姓名 | 长度限制 20 位 |
| String | Password | 密码 | 长度限制 16 位 |

UserModel 类用于操纵 User，控制用户的登陆、登出。

表 4-2　UserModel 类功能表

| 类功能描述 | 配置游戏 | | |
|---|---|---|---|
| 类属性 | | | |
| 类型 | 名称 | 描述 | 备注 |
| 主要实现方法 | | | |
| 方法名 | 输入参数 | 输出参数 | 方法功能描述 |
| login | String name<br>String pwd | User | 登陆系统，用户输入 Name 和 PASSWORD 后进行验证，验证通过返回 User 对象，验证失败返回 null。 |
| logout | Void | Void | 注销系统，注销成功。 |

### 4.3.2 系统登录

Game 类是一个 POJO 类，用于记录系统的用户信息。

表 4-3　Game 类功能表

| 类功能描述 | 游戏信息 | | |
|---|---|---|---|
| 类属性 | | | |
| 类型 | 名称 | 描述 | 备注 |
| String | ID | 游戏 ID | 系统中游戏的唯一标示，由系统自动生成 |
| String | AdminID | 游戏所有者 ID | 游戏所有者 ID |
| String | Name | 游戏名字 | 游戏名字，长度限制 10 位 |
| Long | StartTime | 游戏开始时间（Unix 时间戳） | 游戏真正开始的时间 |
| Long | Duration | 游戏持续时间（秒） | 游戏开始后持续时间，必须在 10-180s 内 |

GameModel 类用于操纵 Game，控制游戏的创建、修改和删除。

表 4-4　GameModel 类功能表

| 类功能描述 | 配置游戏 | | |
|---|---|---|---|
| 类属性 | | | |
| 类型 | 名称 | 描述 | 备注 |
| 主要实现方法 | | | |
| 方法名 | 输入参数 | 输出参数 | 方法功能描述 |
| create | String name<br>Long createTime<br>Long duration | Game | 创建一个指定名字的游戏。 |
| save | Game game | Boolean | 如果游戏存在则按当前的信息保存并返回 true 表示修改成功，如果游戏不存在或因为其他原因修改失败返回 false |
| delete | Game game | Boolean | 如果游戏存在则删除并返回 true 表示修改成功，如果游戏不存在或因为其他原因删除失败返回 false |

## 4.4 关键业务设计

### 4.4.1 系统登录

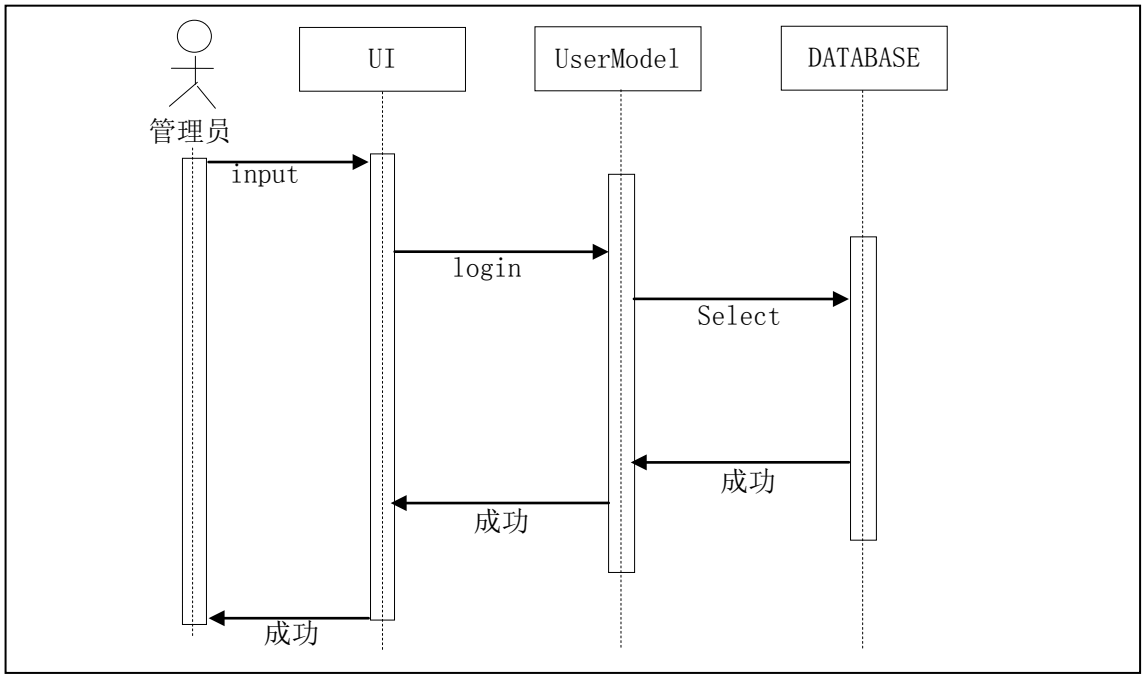


图 4-3　系统登录顺序图

管理员通过 UI 界面输入用户名，密码，调用 UserModel 类 login 方法，在 login 方法中调用 DATABASE 的 select 方法，如果一切正常则返回成功。

### 4.4.2 创建游戏

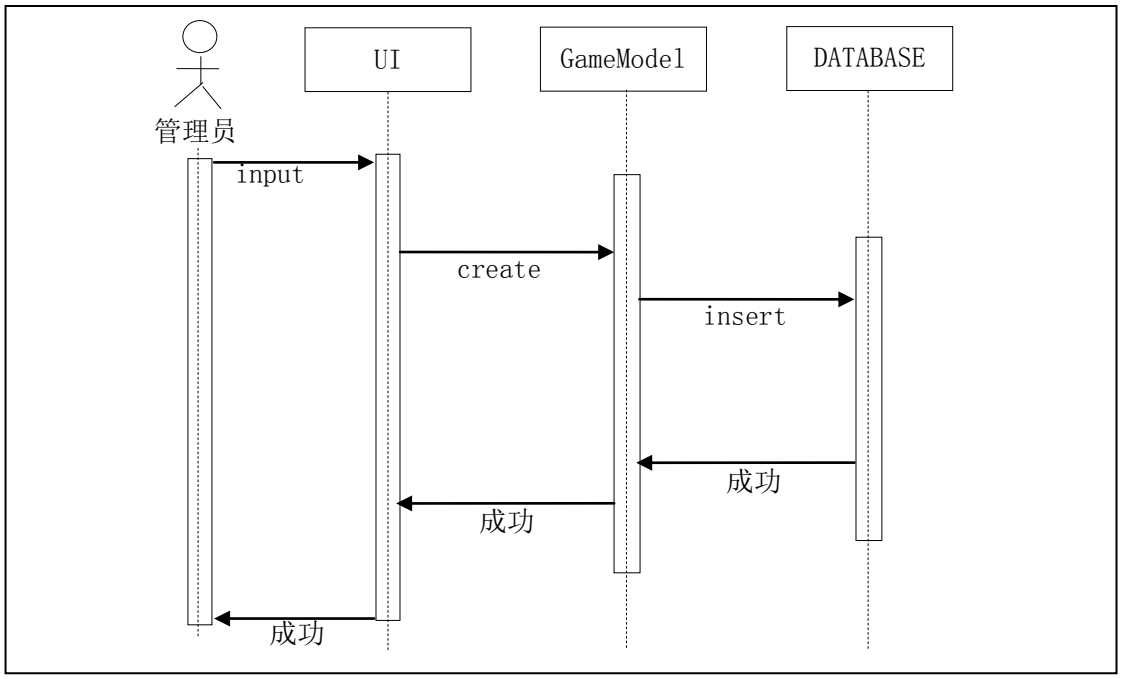


图 4-4　创建也游戏顺序图

管理员通过 UI 界面输入游戏名字，开始时间，持续时间，调用 UserModel 类 create 方法，在 create 方法中调用 DATABASE 的 insert 方法，如果一切正常则返回成功。

从开始时间到持续时间结束前都为游戏时间即状态为运行中，如果当前系统时间早于开始时间则状态为等待状态，如果当前系统时间晚于结束时间则为已结束。

### 4.4.3 数据交互

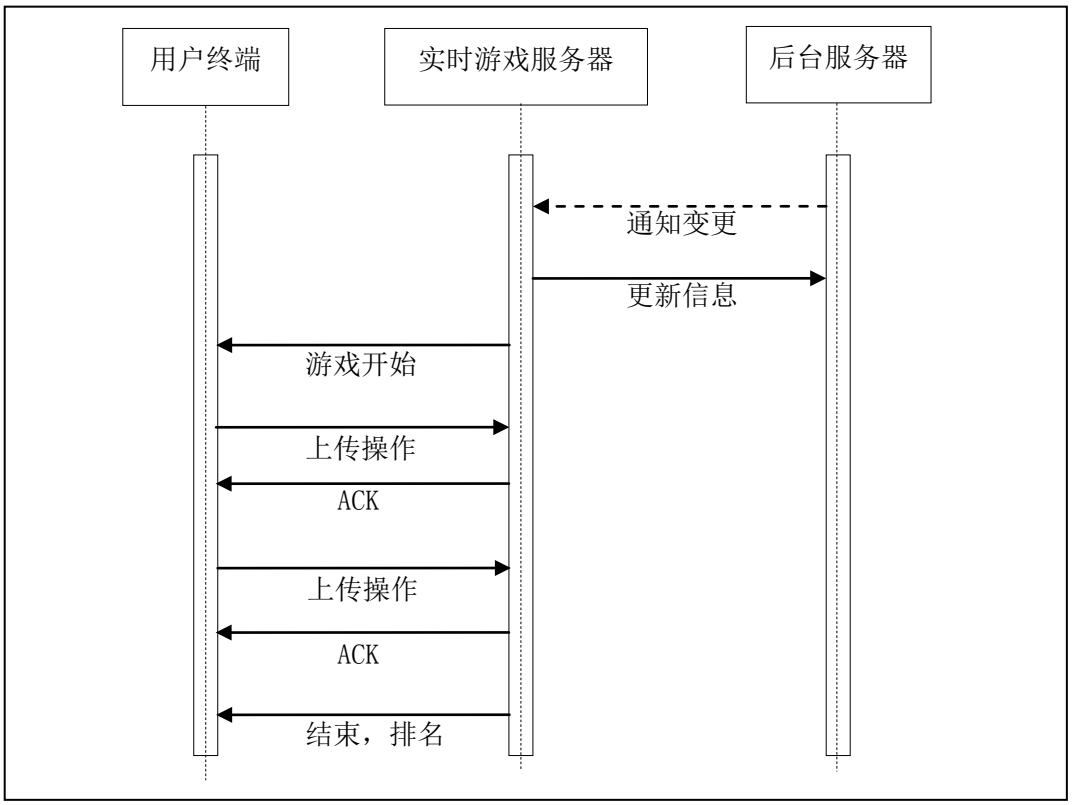


图 4-5　多端数据交互图

实时游戏服务器在启动时或者收到管理后台的通知时会向后台管理服务器拉取正在进行中的游戏。

游戏中的用户在游戏开始后会不断上传游戏操作数据到游戏实时服务器，游戏服务器根据操作进行简单的累加计算得分，并向终端给出收到的确认信息。

当时间结束时，对所有用户进行排名后，向终端发送结果。终端根据收到的结果展示相应的排名。

## 4.5 数据库设计

### 4.5.1 概述

本系统一共涉及到 User 和 Game 两张表，分别用于保存用户和游戏的信息。

### 4.5.2 概念设计

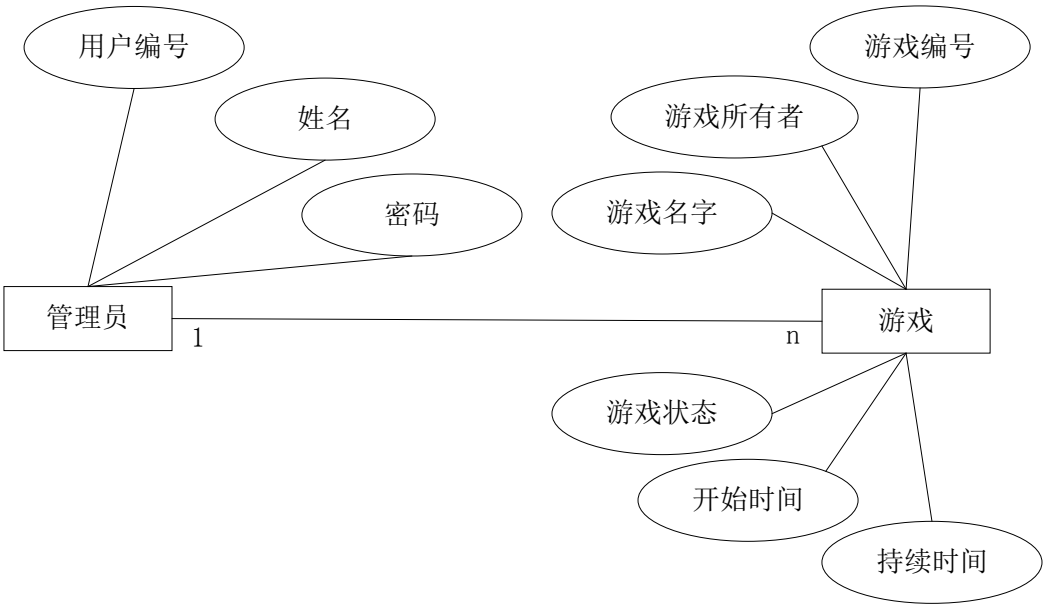


图 4-6　系统 ER 图

### 4.5.3 数据库表

系统中使用到数据库基本如表 4-1 所示。

表 4-5　系统数据库表

| 编号 | 表名 | 描述 |
|---|---|---|
| 01 | User | 用户表 |
| 02 | Game | 游戏表 |

数据库中的用户表用来存储各类用户的相关数据，每个用户都要在用户表中有相相应的记录。

数据库中的游戏表用来存储每一场游戏的具体信息，每一场游戏都要在游戏表中有相应几率。

- 用户表

系统用户表用于登录系统使用，用户表的具体字段如表 4-6 所示。

表 4-6　用户表

| 字段名 | 字段描述 | 类型/长度 | 约束 | 备注 |
|---|---|---|---|---|
| id | 用户编号 | Int | PK | 自增 |
| name | 用户名称 | Varchar(20) | NOT NULL | |
| password | 用户密码 | Varchar(40) | NOT NULL | （32 位 MD5 码） |

字段用户编号为本表的主键用来标识一个用户，用户帐号和密码是两个非空字段，用户使用用户名和密码作为登录系统的凭证，密码字段是使用所 MD5 摘要码有效保证了用户密码不被数据库管理人员看到，可以有效保障系统的安全性。

- 游戏表

表 4-7　游戏表

| 字段名 | 字段描述 | 类型/长度 | 约束 | 备注 |
|---|---|---|---|---|
| id | 游戏编号 | Int | PK | 自增 |
| name | 游戏名称 | Varchar(20) | NOT NULL | |
| admin_id | 游戏所有者 ID | Varchar(20) | FK User(id) | |
| start_time | 游戏开始时间 | Long | NOT NULL | Unix 时间戳 |
| duration | 游戏持续时间 | Long | NOT NULL | 单位：秒 |

## 4.6 本章小结

本章给出了系统的全局架构，并对整个游戏的各端的职责需要有明确的划分，同时明确了关键业务的流程。

# 第五章 系统实现

## 5.1 Hybrid 容器

在客户端方面实现一个 Hybrid 的容器，其 SDK 命名为 BreezeSDK。为了给前端开发提供一个友好有开发方式，提供配套的 lib-breeze 前端库。

### 5.1.1 JSBridge

- 功能描述

JavaScript JVM 和 Android JVM 之间的桥梁，使得能在 Java 中执行 JavaScript 代码，在 JavaScript 中调用 Java 暴露的接口。

如果要在 Java 环境中调用 JavaScript 代码，可以使用 BreezeSDK 提供的 BRWebView. evaluateJavaScript(String)方法以同步的方式执行一段 JavaScript 代码。

如果要在 JavaScript 环境调用 Java 接口，则需要 lib-breeze，之后可以使用 lib.Breeze.call(plugin, method, params, succCallback, failCallbak)方法以异步的方式调用 Java 代码，Java 代码执行完后以回调的同时通知 JavaScript 环境。

- 技术方案

对于 Java 环境中调用 JavaScript 代码，Android WebView 有提供一个原生的 evaluateJavaScript(String,ValueCallback<String>)方法，但是该方法要求系统版本高于 4.4 才能使用，而国内市场仍然有大量用户不满足此要求，因此需要另辟蹊径。

Android WebView 还有提供一个 loadUrl(String)方法，该方法一般用于页面的跳转，但网页自身技术有要求要以执行 JavaScript 的方式处理“javascript:”作为前缀的 url，因此可以基于 loadUrl(String)方法来封装一个无系统版本要求的 evaluateJavaScript(String)方法。

```
public void evaluateJavascript(final String script) {
    new Handler(getContext().getMainLooper()).post(new Runnable() {
        @Override
        public void run() {
            loadUrl("javascript:" + script);
        }
    });
}
```

对于 JavaScript 环境中调用 Java 暴露的接口，Android WebView 有提供一个 addJavaScriptInterface(Object, String)方法来添加能在 JavaScript 环境中调用的同步接口，但是这个接口存在重大缺陷。在系统版本低于 4.2 的设备上存在 XSS 漏洞，攻击方可以在 JavaScript 中利用 Java 的反射特性使用系统 API。而且同步的执行方式会阻塞 JavaScript 代码的执行，可能会造成 UI 上的卡顿，同时这种同步的执行方式也不符合前端开发的编码规范。

为了提供一个高效、安全、符合前端编码规范的方法，只能使用自定义协议的方式来实现。只要能将在 JavaScript 的信息以字符串的方式在客户端取到，就能够根据自定义的协议来处理逻辑。

Android WebChromeClient 允许通过重写 OnJsPrompt(WebView, String, String, JsPromptResult)、OnJsAlert(WebView, String, String, JsResult)和 OnJsConfirm(WebView, String, String, JsResult)三个方法来自定义提示对话框。其中 OnJsPrompt 方式是用于拦截 JavaScript 中 window.prompt(msg, default)的，但是此方法的体验问题在移动端已经几乎没有人使用了，正适合用来实现我们的通信过程。

自定义的数据传输协议采用 Json 的方式，如下：

```
{
  "plugin": "pluginName",
  "method": "methodName",
  "params": {},
  "info": {}
}
```

plugin 字段表示将要调用的接口所在的模块名，method 字段表示将要调用的方法名，params 字段则表示将要传给调用方法的业务参数，info 字段则是系统添加上下文信息。

同时，为了避免 prompt 的冲突，使用 prompt 的 default 参数作为协议头校验，仅对其值为“hybrid://protocol/breeze”的才进行处理。

由于在 Java 环境无法直接调用到任意命名空间内的方法，甚至回调方法还是匿名的，因此我们需要在 JavaScript 为这些回调方法编号并放入词典中，然后将回调方法的编号放入 info 字段中作为上下文信息传给 Java 环境，同时提供一个 lib.Breeze.callback 方法供客户端通过编号回调 JavaScript 方法。

为了避免添加在 JavaScript 中的 Callback 方法因为一直没有被调用而造成 JavaScript 虚拟机的内存泄露，添加了一个定时的垃圾清理机制来解决，如果一个 Callback 方法在 120 秒内没有被调用，则将会被清除，之后即便从 Native 调用也不会发生任何事。

### 5.1.2 插件机制

- 功能描述

插件机制提供了在 JSBridge 基础上快速拓展功能的能力，SDK 中对常用功能有一个默认实现，业务方接入 SDK 后也能通过 PluginManager 快速定制自己的业务插件。

- 技术方案

插件的生命周期：

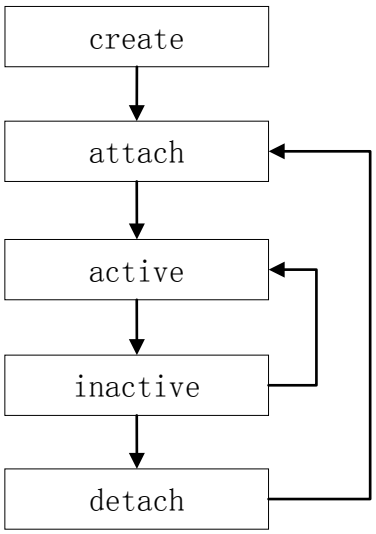


图 5-1　插件生命周期图

create 是一个插件刚刚被创建还没有绑定到 WebView 上的状态。

attach 是一个插件被绑定到 WebView 上后的状态，绑定之后可以取得对应的上下文信息和 WebView 的 JSBridge。

active 是 WebView 在前台工作时的状态。

inactive 是 WebView 切换到后台工作时的状态。

detach 是从 WebView 解除绑定后的状态，此时已经无法再使用上下文信息和 JSBridge 了。

PluginManager 的主要职责就是提供插件的配置，以及生命周期的控制。

插件的启用可以通过 PluginManager 的 registerPlugin(String, HybridgePlugin)方法来注册。当插件不在需要时则可以通过 PluginManager 的 unregisterPlugin(String)方法来反注册。

如果要自定义一个插件，只需要继承抽象类 HybridPlugin 实现 execute(String, String, CallMethodContext)方法即可，该方法即使对传入的方法名和参数处理，处理完成后根据业务结果调用 CallMethodContext 的 success 方法或 fail 方法。

SDK 提供一系列基础 Hybrid 插件，包括系统通知、页面跳转、传感器使用等。例如系统通知插件：

```
public class NotifcationPlugin extends HybridPlugin
{
    public void showToast(final String text)
    {
        Toast.makeText(getContext(), text, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void execute(String method, String params,
CallMethodContext jsContext) {
        if("showToast".equals(method)) {
            if(params != null) {
                try {
                    JSONObject jsonObject = new JSONObject(params);
                    if (jsonObject.has("text")) {
                        String text = jsonObject.getString("text");
                        showToast(text);
                        jsContext.success();
                    }
                } catch (JSONException e) {
                    jsContext.fail();
                }
            }
            jsContext.fail();
        }
    }
}
```

### 5.1.3 拦截器机制

- 功能描述

拦截器机制实现了对资源的面向切面控制。对于所有的资源请求，都需要经过拦截器的检查。SDK 提供业务方快速加入拦截器定制的能力。

- 技术方案

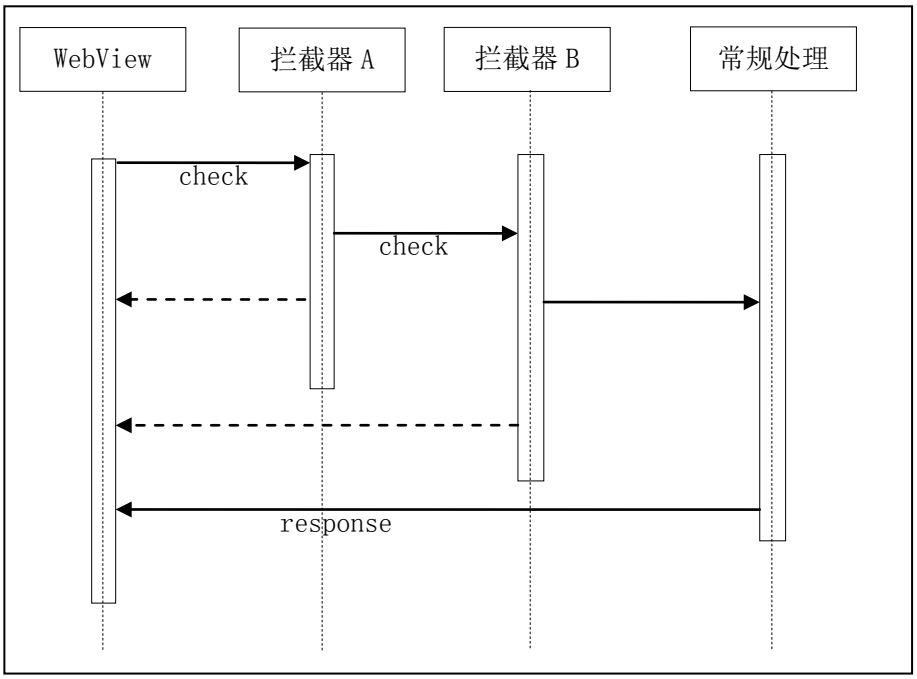


图 5-2　拦截器工作时序图

从 WebView 发出的每一个请求都会经过 InterceptManager 中的拦截器检查，如果命中了一个拦截器，则由拦截器给出请求的响应。

请求的源的拦截可以在 Android WebViewClient 的 shouldInterceptRequest(WebView, String)方法中获取，只需要重新该方法即可轻易拿到每一个请求的 Url，接着将这个 Url 交给 InterceptManager 来检查是否有拦截器要处理。

如果要定制一个拦截器，只需要实现 Interceptor 接口即可。

### 5.1.4 下载机制

- 功能描述

下载机制是一个高度可定制的模块，旨在优化资源在移动网络下的传输，同时为 Hybrid 的缓存机制提供强有力支持。可以轻易的使用第三方网络库来定制整个 SDK 的下载。

- 技术方案

通过规范下载接口来提供外部的高度可定制化，下载过程以观察者模式实现，其中主要的接口有：

DownloadInterface 下载过程的实际处理

DownloadListener 下载成功或失败的监听

每次使用 DownloadService 会为其分配一个 Task ID，接着调用 DownloadInterface 实现类的 download(int, url, String, Object,

DownloadListener)方法来完成下载。整个过程都通过 Task ID 来实现针对每一项任务的处理。如果需要在 Listener 中拿到调用时的逻辑，可以在调用时放置一个对象，该对象在会一直传到 DownloadListener 中。

SDK 内部实现了一个基于系统网络库的基础版本，如果需要使用第三方只需要实现 DownloadInterface 接口，然后在 SDK 中配置 DownloadHandler 即可。

### 5.1.4 缓存机制

- 功能描述

对于常用资源进行预下载缓存，随后对 HTML5 页面中使用相关资源时拦截，如果命中缓存直接返回缓存资源，从而减少了网络传输耗时，加快页面打开速度。

- 技术方案

通过 Map 来管理需要缓存的 url 和本地文件信息。在 SDK 初始化完后且 CPU 空闲时在后台静默下载，默认使用增量下载方式。

同时继承 Interceptor 实现一个资源缓存拦截器，对所有资源进行拦截，如果能在缓存的 Map 到相关信息并且已经下载到本地则视为命中缓存，由缓存来响应。

### 5.1.5 事件机制

- 功能描述

允许 Plugin 以 Service 方式运行，如果 Plugin 处于 Active 状态则可以向 EventService 发送事件。

- 技术方案

EventService 主要通过 JSBridge 调用 lib.Breeze.fire(event)来分发事件。其中传输协议为：

```
{
   "code": "eventCode",
   "info": {}
}
```

协议中的 code 字段是事件编码，info 字段则是事件附带的信息，由业务方自己定义。在 lib.Breeze.fire(event)中将根据事件编码将事件交给对应的 Listener 处理。Lisrener 可由 lib.Breeze.addEventListener(code, listener)加入，在其内部使用一个以 code 为主键词典来管理，如此可以快速找到能响应当前事件的所有 Listener。

## 5.2 摇一摇游戏前端页面


图 5-3　游戏页面

- 功能描述

游戏的业务逻辑实现，能够上报自己设备的摇动信息，以及展示排名。

- 技术方案

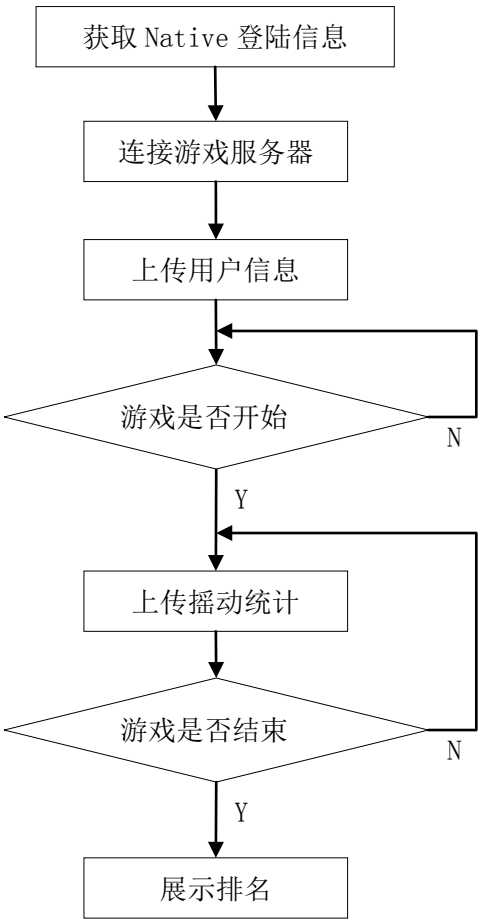


图 5-4　游戏前端工作流程图

其中 Native 登陆信息，由客户端实现一个 UserPlugin 注册到 Hybrid 容器的 PluginManager 中，HTML5 中通过 JSBridge 来获取。相关代码如下：

```
lib.Breeze.call('user', 'getUserNick', {}, function(nick) {
    next(nick);
},
function(e) {
});
```

游戏的开始由游戏实时服务器下发的"gameStart"消息作为信号。

游戏开始后通过 JSBridge 打开时 ShakePlugin，随后通过事件监听就能获取到手机的摇动信息,同时上报给游戏实时服务器。相关代码如下：

```
lib.Breeze.addEventListener('shake', function() {
    uploadShake(count);
},
```

```
function(e) {
});
```

因为前端的首屏展示速度和网络传输非常相关，在移动网络下如果需要加载多个资源则会多次建立连接，从而导致性能较差。所以我们需要对 CSS 和 JS 进行内联处理来减少网络请求数，同时一些 Hybrid 已经缓存的仍然保留其外部链接形式。对 CSS 和 JS 的内联可以使用 Gulp 和 WebPack 来配置。最后在通过 Gulp 的插件 gulp-uglify 和 gulp-concat 进行混淆输出。

## 5.3 实时游戏服务

- 功能描述

  处理多个用户之间的游戏数据同步。

- 技术方案

  每一个终端在进入游戏后，通常都以 socket.io 建立到实时游戏服务端的长连接，之后再次收发消息时将直接投递。除非 WebView 不支持 WebSocket,但是目前主流的机器都已经支持。

  Socket.io 发送消息是以一个 code 和 msg 组成的，所以我们只需要为不同的事件定义 code 即可达到控制游戏的过程。

表 5-1　消息表

| 消息 Code | 消息 Msg | 备注 |
| --- | --- | --- |
| started | None | 游戏已经开始 |
| uploadNick | Nick | 上报当前用户 Nick |
| uploadShake | None | 上报一次手机摇动 |
| ended | RankList | 游戏已经结束，Msg 中传递排行榜，Nick 之间以逗号作为分隔符 |
| killed | None | 游戏被强制终止 |

## 5.4 游戏管理后台


图 5-5　系统后台查看

### 5.4.1 系统登录


图 5-6　系统登陆

- 功能描述

  登陆体系用于验证管理员的合法信息，防止后台被恶意使用。
- 技术方案

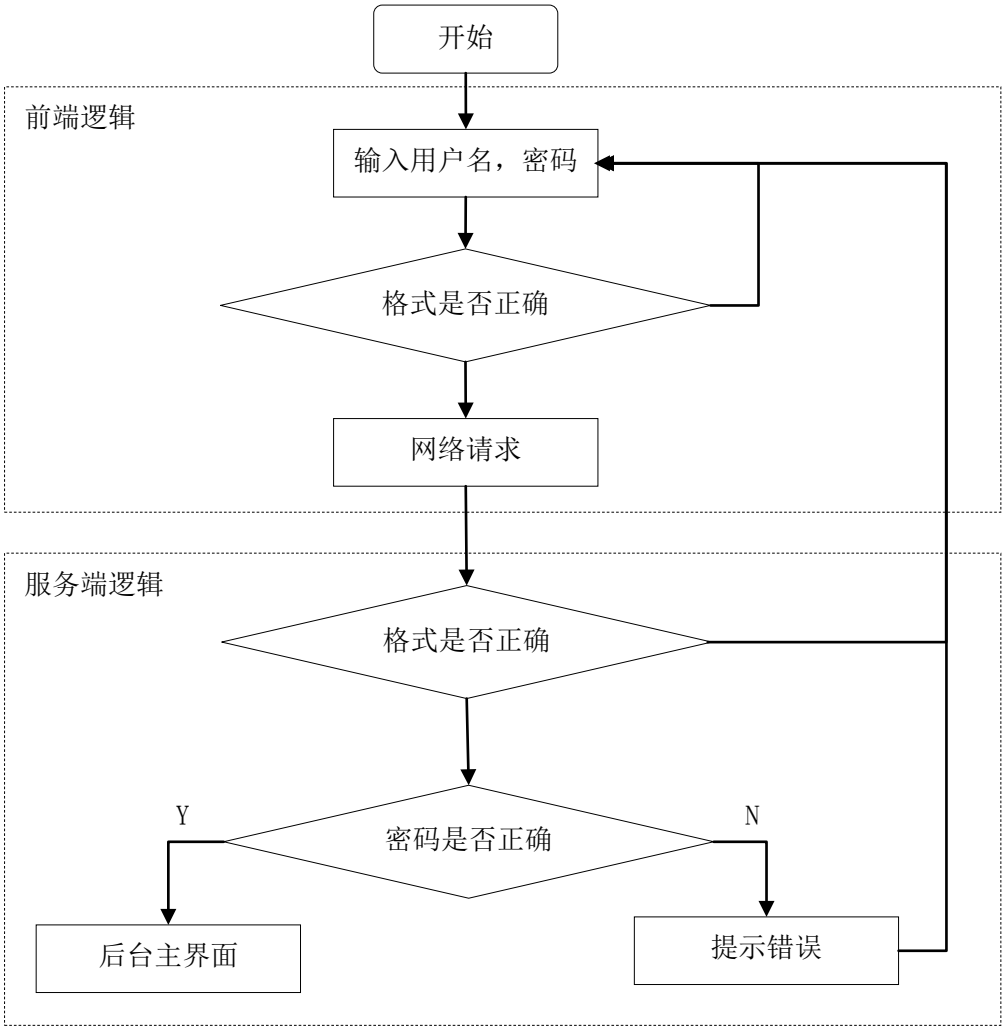


图 5-5　登陆流程图

在输入登陆信息后，前端做一次格式验证后将使用用户名和 MD5 后的密码发起一次 HTTP 请求。服务器端收到数据后，再验证一次格式，之后根据用户名从数据库的 User 表中获取密码的 MD5 值与前端传来的参数做比较，如果一致则被认为是登陆成功。登陆成功时，会将登陆成功的信息放入 Session 中，SessionID 则被自动被放入 Cookie。

后续的登陆验证则可以通过 Session 中的信息来确定当前是否是登陆。

### 5.4.2 创建游戏


图 5-7　创建游戏

- 功能描述

使得管理员能自由的配置游戏的开始和结束，并且对游戏的当前状况有所了解。

- 技术方案

在输入登陆信息后，前端将表单信息发送给服务端，服务端验证数据合法性后，写入 DB，同时通知当前连接到的实时游戏服务器游戏信息有变更。

## 5.5 本章小结

本章主要针对 Hybrid 的各个机制给出了详细设计，游戏的多端协作给出了具体的解决方案。

值得注意的是，客户端 Hybrid 容器实现需要较多的自定义协议，但此种方案获得最大的灵活性。同时多端之间信息传输也使用到了自定义协议，此时更多考虑的是协议的轻便，以减少网络传输开销。同时设计了后台来控制实时游戏服务器。

# 第六章　系统测试

## 6.1 系统测试综述

测试需要保证系统功能正常。

系统整体采用黑盒测试方法，关注点在于用户做出输入后，能否得到预期的输出结果。

## 6.2 测试用例

### 6.2.1 用户登录模块

表 6-1　用户登录

| 功能测试 | | | | | |
|---|---|---|---|---|---|
| 概述 | | | | | |
| 测试编号 | GN001 | | | | |
| 功能描述 | 用户登录 | | | | |
| 功能 URL | | | | | |
| 用例目的 | 测试用户是否能登录成功 | | | | |
| 前提条件 | 进入用户登录界面 | | | | |
| 测试操作 | | | | | |
| 编号 | 输入/动作 | 期望的输出响应 | 实际情况 | 是否正确 | 错误编号 |
| 1 | 输入不存在的用户名，不填写密码 | 系统提示用户名或密码错误 | 系统提示登录名或密码错误 | 正确 | |
| 2 | 输入正确的用户名输入错误的密码 | 系统提示用户名或密码错误 | 系统提示登录名或密码错误 | 正确 | |
| 3 | 输入错误的用户名输入正确的密码 | 系统提示用户名或密码错误 | 系统提示登录名或密码错误 | 正确 | |
| 4 | 不填写用户名及密码 | 系统提示用户名或密码错误 | 系统提示用户名或密码错误 | 正确 | |

### 6.2.2 游戏定时开始

表 6-2　游戏定时开始

| 功能测试 | | | | | |
|---|---|---|---|---|---|
| 概述 | | | | | |
| 测试编号 | GN002 | | | | |
| 功能描述 | 游戏定时开始 | | | | |
| 用例目的 | 测试游戏能否在配置的时间开始 | | | | |
| 前提条件 | 服务端已经配置了一场即将启动的游戏 | | | | |
| 测试操作 | | | | | |
| 编号 | 输入/动作 | 期望的输出响应 | 实际情况 | 是否正确 | 错误编号 |
| 1 | 在后台配置一场 1 分钟后开始的游戏 | 时间到后游戏开始 | 游戏在 1 分钟后开始 | 正确 | |
| 2 | 在后台配置一场 5 分钟后开始的游戏 | 时间到后游戏开始 | 游戏在 5 分钟后开始 | 正确 | |
| 3 | 在后台配置一场 1 分钟前开始的游戏 | 系统提示非法的时间 | 系统提示非法的时间 | 正确 | |

### 6.2.3 游戏排名

表 6-3　游戏排名

| 功能测试 | | | | | |
|---|---|---|---|---|---|
| 概述 | | | | | |
| 测试编号 | GN002 | | | | |
| 功能描述 | 游戏排名 | | | | |
| 用例目的 | 测试游戏能否在结束时给出排名 | | | | |
| 前提条件 | 游戏进行中 | | | | |
| 测试操作 | | | | | |
| 编号 | 输入/动作 | 期望的输出响应 | 实际情况 | 是否正确 | 错误编号 |
| 1 | 两个终端，其中一个放置于桌面，另一个不断摇动 | 放置于桌面的终端排名低于不断摇动的 | 放置于桌面的终端排名低于不断摇动的 | 正确 | |
| 2 | 两个终端，以较低频率摇动，另一个以较高频率摇动 | 以较低频率摇动的终端排名低于以较高频率的 | 以较低频率摇动的终端排名低于以较高频率的 | 正确 | |

## 6.3 测试分析

测试结果符合预期，游戏的流程正确，对于异常的输入能给出错误提示，系统质量良好。

## 6.4 本章小结

采用黑盒测试的方法针对系统的主要模块做了功能测试，并分析。测试结果符合预期，游戏流程正确，对于异常能给出错误提示，系统质量良好。

# 第七章　总结

采用 Hybrid 来实现业务是在性能和灵活性、开发成本的平衡上做的较好的方案。灵活性上其发布后更新时间，基本只依赖于静态服务器的缓存回源时间。开发成本则是可以充分利用之前 Web 前端开发人员来开发大量移动端页面。

在互动游戏上，Hybrid 容器基本只需要提供几个传感器即可满足互动需求，设计人员在有 Idea 后项目可以快速落地，也不再需要老版本用户升级到最新版本客户端才能参与，大幅度降低了参与了互动的门槛。

Hybrid 容器部分作为 SDK 实现，使得该容器可以快速集成到任何 App，使其具备同等能力。SDK 的插件机制能使得业务方快速定制自己的功能，并且插件设计同时使得整个 SDK 非常轻量，没有引入过多的不需要的功能。

但是，目前 Hybrid 各大模块都还有可以改进的空间。JSBridge 模块目前缺乏权限控制，而 HTML5 页面则是任何人都可以发布的，其安全性有待加强。ResourceCache 目前只具备全量更新和增量更新两种更新方式，许多资源下载后可能不会再被用到，随着缓存数量越来越多会造成查询效率低下，且大量磁盘空间浪费，随后可以引入优先级和缓存淘汰机制。Download Service 目前默认使用系统的实现，然而移动网络还有大量可以优化的地方。虽然有提供替换网络实现的机制，但是目前还缺乏对主流的第三方网络库的支持。

# 参考文献

[1] JavaScript 使用详解[M],机械工业出版社,7-111-06897-1,TP312JA
[2] Web 专家工作室系列-JavaScript 实战解析[M],机械工业出版社, ,TP312JA
[3] JavaScript 程序设计及应用[M],西北工业大学出版社, ,TP312JA
[4] JavaScript 1.1 开发指南[M],Tatters,清华大学出版社,7-302-02856-7,TP312JA
[5] JavaScript 编程指南[M],王炜,电子工业出版社,7-5053-5187-7,TP312JA
[6] 萨师煊，王珊.数据库系统概论(第三版)[M].北京：高等教育出版社，1998.
[7] 冯明.基于混合模式(Hybrid App)移动终端设计的方法[J].数字技术与应用,2015,第 4 期:148-149
[8] 李刚.疯狂 Android 讲义[M].北京：电子工业出版社，2013.
[9] 杨丰盛.Android 技术内幕[M].北京：计协工业出版社，2011.
[10] 杨云君.Android 的设计与实现[M].北京：机械工业出版社，2013.
[11] 程湘.下一代 Web 界面技术研究[J].华商,2008(5):20-24.
[12] 马越.Android 的架构与应用[D].北京：中国地质大学，2008:330-357
[13] 柯元旦.Android 内核剖析[M].北京：机械工业出版社，2013.
[14] 钟茂生，王明文.软件设计模式及其使用[J].计算机应用，2002.
[15] 姚昱旻, 刘卫国. Android 的架构与应用开发研究[J]. 计算机系统应用, 2008, 17(11):110-112.

# 致　谢

在我的论文设计期间，首先我要感谢我的指导老师刘冰老师，他经常第一时间来关注我论文的进展，使我能够及时、顺利的完成此次的论文。同时我还要感谢设计期间一直支持我的工作室朋友们，他们经常与我进行头脑风暴，这是我灵感的源泉。最后，我要感谢学校对我长期以来的栽培，才能使我完成这篇论文。

# 基于 Hybrid 技术的多人在线摇一摇游戏

黄睿

（江西理工大学软件学院，信息工程系，江西 南昌 330013）

**摘　要：**混合型应用（Hybrid App）是指以移动客户端为主体，搭载移动端的 HTML5 页面，充分发挥客户端的可定制性和 HTML5 的灵活性。针对像“摇一摇”这样的互动性项目，往往具有时效性需要快速发布的要求，采用 Hybrid 技术恰好能解决其痛点。本文研究的就是将 Hybrid 技术应用于此类互动性项目的解决方案。首先以 Android 为平台研发了一个轻量 Hybrid 容器，接着在 Hybrid 容器上实现了一个极具代表性的互动项目多人在线的“摇一摇”游戏，并为控制游戏建立了其管理后台。

**关键字：** Hybrid，互动游戏，JavaScript，Android

**中图分类号：**G642　　**文献标志码：**A

# Hybrid technology-based multiplayer online game Shake

Huangrui

(Jiangxi university of science and technology institute of software, jiangxi nanchang 330000)

**Abstract:** Hybrid applications (Hybrid App) refers to the mobile client as the main body, equipped with mobile side HTML5 page, give full play to the client customizability and flexibility of HTML5. For like "Shake" This interactive projects, often with the timeliness required rapid release requirements, using Hybrid technology just to solve their pain points. This study is to Hybrid technology applied to such solutions interactive projects. First Android-platform Hybrid developed a lightweight container, then the container Hybrid implements a highly representative of interactive projects multiplayer online "Shake" game, control the game and for the establishment of its management background.

**Key words:** Hybrid, Interactive Game, JavaScript, Android；

## 1 基于 Hybrid 方案的游戏设计

### 1.1 全局架构

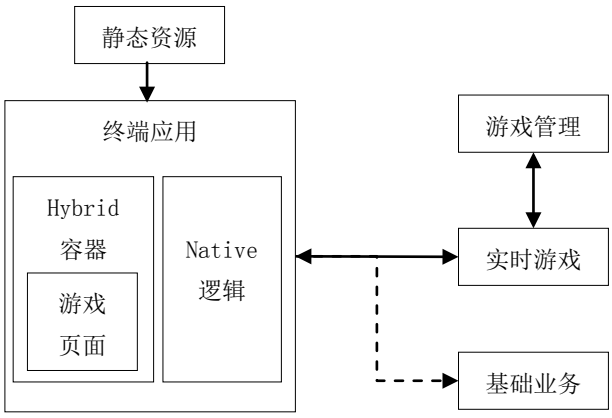


图 1　全局架构图

在客户端中集成 Hybrid 容器 SDK，游戏页面发布在静态资源服务器，客户端可以通过配置来开放入口或者让用户以扫码访问访问游戏页面，从而使游戏逻辑脱离客户端基础业务逻辑存在。游戏页面与实时游戏服务器以高速全双工通道通信保证游戏实时性。游戏后台则可以控制实时游戏服务器的各项参数，从而达到控制游戏的目的。基础业务服务器是客户端原有的其他业务服务器，主要为其他业务提供支持，同时客户端的登陆也发生在这里。

### 1.2 Hybrid 容器

Hybrid 容器的核心由多部分组成。其中 JSBridge 提供基础的双向通道，InterceptManager 用于拦截请求，PluginManager 提供业务暴露自身逻

辑接口的能力，ResourceCache 通过预下载缓存静态资源来加速页面访问，Download Service 提供一个可支持替换实现的下载功能，Event Service 提供一套事件机制。

### 1.3 游戏业务

管理员可以登陆后通过 UI 界面输入游戏名字、开始时间和持续时间来创建游戏。随后，管理后台会通知实时游戏服务器数据有变更，接着实时服务器向后台管理服务器拉取正在进行中的游戏。

游戏的业务以 HTML5 页面的形式放置在静态资源服务上，当玩家访问页面时即可访问到最新的游戏业务逻辑。玩家在游戏开始后会不断上传游戏操作数据到游戏实时服务器，游戏服务器根据操作进行简单的累加计算得分，并向终端给出收到的确认信息。

当时间结束时，对所有用户进行排名后，向终端发送结果。终端根据收到的结果展示相应的排名。

## 2 Hybrid 技术实现方案

### 2.1 JSBridge

Android WebView 提供一个 loadUrl 方法，该方法一般用于页面的跳转，但网页自身技术有要求要以执行 JavaScript 的方式处理“javascript:”作为前缀的 url，因此可以基于 loadUrl 方法来封装一个无系统版本要求的 evaluateJavaScript 方法。

Android WebChromeClient 允许通过重写 OnJsPrompt 拦截 JavaScript 中 window.prompt，并且因为 window.prompt 的体验问题在移动端已经几乎没有人使用了，正适合用来实现我们的通信过程。

自定义的数据传输协议采用 Json 的方式，如下：

```
{
   "plugin": "pluginName",
   "method": "methodName",
   "params": {},
   "info": {}
}
```

plugin 字段表示将要调用的接口所在的模块名，method 字段表示将要调用的方法名，params 字段则表示将要传给调用方法的业务参数，info 字段则是系统添加上下文信息。

同时，为了避免 prompt 的冲突，使用 prompt 的 default 参数作为协议头校验，仅对其值为“hybrid://protocol/breeze”的才进行处理。

由于在 Java 环境无法直接调用到任意命名空间内的方法，甚至回调方法还是匿名的，因此我们需要在 JavaScript 为这些回调方法编号并放入词典中，然后将回调方法的编号放入 info 字段中作为上下文信息传给 Java 环境，同时提供一个 lib.Breeze.callback 方法供客户端通过编号回调 JavaScript 方法。

### 2.2 插件机制

PluginManager 的主要职责就是提供插件的配置，以及生命周期的控制。

其中一个 HybridPlugin 具有 5 个生命周期，分别是是 create、attach、active、inactive 和 detach。create 是一个插件刚刚被创建还没有绑定到 WebView 上的状态。attach 是一个插件被绑定到 WebView 上后的状态，绑定之后可以取得对应的上下文信息和 WebView 的 JSBridge。active 是 WebView 在前台工作时的状态。inactive 是 WebView 切换到后台工作时的状态。detach 是从 WebView 解除绑定后的状态，此时已经无法再使用上下文信息和 JSBridge 了。

插件的启用可以通过 PluginManager 的 registerPlugin 方法来注册。当插件不在需要时则可以通过 PluginManager 的 unregisterPlugin 方法来反注册。

如果要自定义一个插件，只需要继承抽象类 HybridPlugin 实现 execute 方法即可，该方法即使对传入的方法名和参数处理，处理完成后根据业务结果调用 CallMethodContext 的 success 方法或 fail 方法来回调给 JavaScript 环境。

### 2.3 拦截器机制

从 WebView 发出的每一个请求都会经过 InterceptManager 中的拦截器检查，如果命中了一个拦截器，则由拦截器给出请求的响应。

请求的源的拦截可以在 Android WebViewClient 的 shouldInterceptRequest(WebView, String)方法中获取，只需要重新该方法即可轻易拿到每一个请求的 Url，接着将这个 Url 交给 InterceptManager 来检查是否有拦截器要处理。

如果要定制一个拦截器，只需要实现 Interceptor 接口即可。

### 2.4 下载机制

通过规范下载接口来提供外部的高度可定制化，下载过程以观察者模式实现，其中主要的接口有：

DownloadInterface 下载过程的实际处理

DownloadListener 下载成功或失败的监听

每次使用 DownloadService 会为其分配一个 Task ID，接着调用 DownloadInterface 实现类的 download 方法来完成下载。整个过程都通过 Task ID 来实现针对每一项任务的处理。如果需要在 Listener 中拿到调用时的逻辑，可以在调用时放置一个对象，该对象在会一直传到 DownloadListener 中。

Hybrid 容器默认实现了一个基于系统网络库的基础版本，如果需要使用第三方只需要实现 DownloadInterface 接口，然后在初始化 Hybrid 时配置为 DownloadHandler 即可。

### 2.5 缓存机制

通过 Map 来管理需要缓存的 url 和本地文件信息。在 SDK 初始化完后且 CPU 空闲时在后台静默下载，默认使用增量下载方式。

同时继承 Interceptor 实现一个资源缓存拦截器，对所有资源进行拦截，如果能在缓存的 Map 到相关信息并且已经下载到本地则视为命中缓存，由缓存来响应。

### 2.6 事件机制

EventService 主要通过 JSBridge 调用 lib.Breeze.fire 来分发事件。其中传输协议为：

```
{
   "code": "eventCode",
   "info": {}
}
```

协议中的 code 字段是事件编码，info 字段则是事件附带的信息，由业务方自己定义。在 lib.Breeze.fire 中将根据事件编码将事件交给对应的 Listener 处理。Lisrener 可由 lib.Breeze.addEventListener 加入，在其内部使用一个以 code 为主键词典来管理，如此可以快速找到能响应当前事件的所有 Listener。

## 3 游戏业务逻辑


图 2　游戏页面

### 3.1 游戏主流程实现

游戏的开始由游戏实时服务器下发的"gameStart"消息作为信号。

游戏开始后通过 JSBridge 打开时 ShakePlugin，随后通过事件监听就能获取到手机的摇动信息,同时上报给游戏实时服务器。

因为前端的首屏展示速度和网络传输非常相关，在移动网络下如果需要加载多个资源则会多次建立连接，从而导致性能较差。所以我们需要对 CSS 和 JS 进行内联处理来减少网络请求数，同时一些 Hybrid 已经缓存的仍然保留其外部链接形式。对 CSS 和 JS 的内联可以使用 Gulp 和 WebPack 来配置。最后在通过 Gulp 的插件 gulp-uglify 和 gulp-concat 进行混淆输出。

### 3.2 后台登陆实现

在输入登陆信息后，前端做一次格式验证后将使用用户名和 MD5 后的密码发起一次 HTTP 请求。服务器端收到数据后，再验证一次格式，之后根据用户名从数据库的 User 表中获取密码的 MD5 值与前端传来的参数做比较，如果一致则被认为是登陆成功。登陆成功时，会将登陆成功的信息放入 Session 中，SessionID 则被自动被放入 Cookie。

后续的登陆验证则可以通过 Session 中的信息来确定当前是否是登陆。

### 3.3 游戏创建实现

在输入登陆信息后，前端将表单信息发送给服务端，服务端验证数据合法性后，写入 DB，同时通知当前连接到的实时游戏服务器游戏信息有变更。

## 参考文献

[1] JavaScript 编程指南[M],王炜,电子工业出版社,7-5053-5187-7,TP312JA

[2] 萨师煊，王珊.数据库系统概论(第三版)[M].北京：高等教育出版社，1998.

[3] 冯明.基于混合模式(Hybrid App)移动终端设计的方法[J].数字技术与应用,2015,第 4 期:148-149

[4] 李刚.疯狂 Android 讲义[M].北京：电子工业出版社，2013.

[5] 杨丰盛.Android 技术内幕[M].北京：计协工业出版社，2011.

[6] 杨云君.Android 的设计与实现[M].北京：机械工业出版社，2013.