1、技术选型表

项目	web app	备注
1、终端支持	pc pad phone	
1.1、开发语言框架	前端： HTML5 CSS3 JavaScript	
1.2、响应式布局框架	Bootstrap Vue.js Google MaterialDesign	
1.3、传感器	无	
2 服务端支持		
2.1 语言	Java	
2.2 web 框架	struts+spring	
2.3 ORM 框架	Hibernate	
2.4 关系数据库	MySQL	
2.5 数据缓存	localstorage redis	分为终端缓存以及服务端缓存两类
2.6 负载均衡	nginx	
2.7 信息中间件	ZeroMQ	
2.8 其他第三方组件	聚合数据-影视影讯检索 API 微信扫码支付 API（待定）	
3 开发平台与工具		
3.1 IDE	前端： sublime text	

	服务端 idea eclispe	
3.2 集成与测试	bugclose	
3.3 源代码管理	Github	

2、技术原型开发内容

2.1 项目技术风险元素。

- 实时聊天
- 本地缓存
- 推荐电影
- 支付方式

2.2 验证性技术原理

● 实时聊天

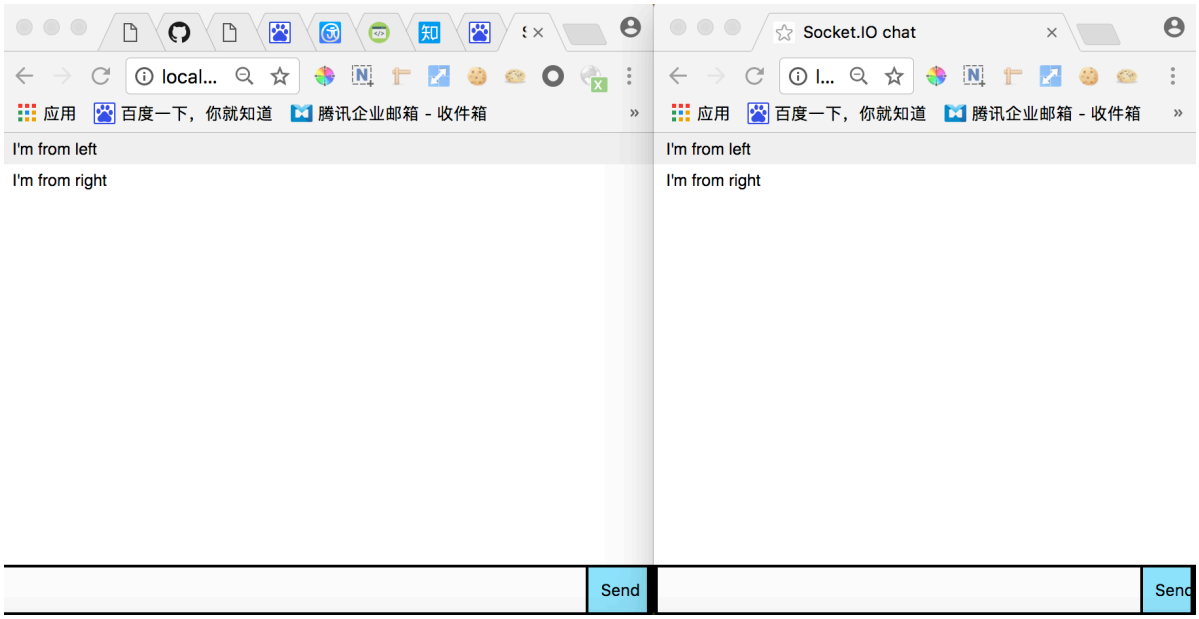
在 CS 基本框架中。要实现用户的实时聊天需要途径 Client-Server-Client。http 协议是单双工的，而这里我们需要使用 websocket 实现全双工的通信。我们这里给出一个使用 nodejs 实现的实时聊天 demo，通过这一机制，我们可以让服务器广播到特定用户，从而实现聊天功能。

```
io.on('connection', function(socket){
  socket.on('chat message', function(msg){
    socket.broadcast.emit('chat message', msg);
  });
});
```

服务端打开信息监听接口，获取到信息后广播

```
var socket = io();
$('form').submit(function(){
  socket.emit('chat message', $('#m').val());
  $('#m').val('');
  return false;
});
socket.on('chat message', function(msg){
  $('#messages').append($('<li>').text(msg));
});
```

终端 js 广播信息，并且监听服务端发送的 socket 信息



nodejs 原生支持并发事件，所以我们只需要监听不同的事件名就可以建立多个聊天窗口。而使用 java 的话，需要使用多线程开发。

- 本地缓存

传统的 web app 中，要在本地保存信息的话一般会使用到 cookie 或者文件机制，但是 cookie 存储的大小有限，而 file 的话又过于麻烦，而且会污染用户终端。得益于 html5，我们现在可以使用诸如 localstorage、indexedDB 等新的技术去实现本地缓存。

html 的缓存机制类似于 noSQL，是以对象的形式将信息存储起来，方便下次获取的。使用方式特别简单：

```
// Store
localStorage.setItem("lastname", "Gates");
// Retrieve
document.getElementById("result").innerHTML = localStorage.getItem("lastname");
```

- 支付方式

web app 中使用微信的扫码支付是一项比较快捷易用的选择，使用上用微

信官方的文档支持，比较方便。

场景介绍

用户扫描商户展示在各种场景的二维码进行支付。

步骤1：商户根据微信支付的规则，为不同商品生成不同的二维码（如图6.1），展示在各种场景，用于用户扫描购买。

步骤2：用户使用微信“扫一扫”（如图6.2）扫描二维码后，获取商品支付信息，引导用户完成支付（如图6.3）。

图6.1 支付二维码

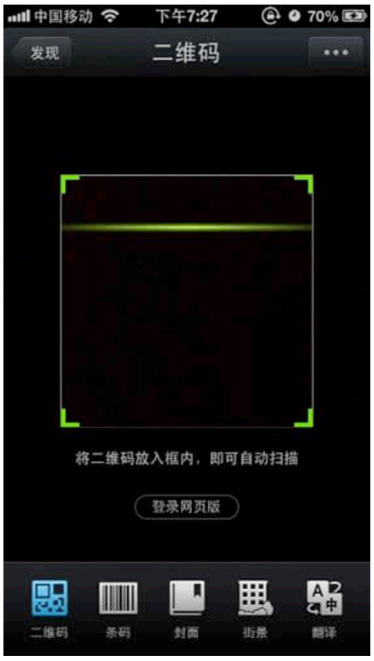


图6.2 打开微信扫一扫二维码

图6.3 确认支付页面

不过需要注意的是要完成支付步骤，需要申请微信商户，这可能成为项目中一个比较麻烦的地方。

- 推荐电影

推荐算法大致可以分为三类：基于内容的推荐算法、协同过滤推荐算法和基于知识的推荐算法。

基于内容的推荐算法，原理是用户喜欢和自己关注过的 Item 在内容上类似的 Item，比如你看了哈利波特 I，基于内容的推荐算法发现哈利波特 II-VI，与你以前观看的在内容上面（共有很多关键词）有很大关联性，就把后者推荐给你，这种方法可以避免 Item 的冷启动问题（冷启动：如果一个 Item 从没有被关注过，其他推荐算法则很少会去推荐，但是基于内容的推荐算法可以分析 Item 之间的关系，实现推荐），弊端在于推荐的 Item 可能会重复，典型的就是新闻推荐，如果你看了一则关于 MH370 的新闻，很可能推荐的新闻和你浏览过的，内容一致；另外一个弊端则是对于一些多媒体的推荐（比如音乐、电影、图片等）由于很难提内容特征，则很难进行推荐，一种解决方式则是人工给这些 Item 打标签。

协同过滤算法，原理是用户喜欢那些具有相似兴趣的用户喜欢过的商

品，比如你的朋友喜欢电影哈利波特 I，那么就会推荐给你，这是最简单的基于用户的协同过滤算法（user-based collaboratIve filtering），还有一种是基于 Item 的协同过滤算法（item-based collaborative filtering），这两种方法都是将用户的所有数据读入到内存中进行运算的，因此成为 Memory-based Collaborative Filtering，另一种则是 Model-based collaborative filtering，包括 Aspect Model，pLSA，LDA，聚类，SVD，Matrix Factorization 等，这种方法训练过程比较长，但是训练完成后，推荐过程比较快。

最后一种方法是基于知识的推荐算法，也有人将这种方法归为基于内容的推荐，这种方法比较典型的是构建领域本体，或者是建立一定的规则，进行推荐。

混合推荐算法，则会融合以上方法，以加权或者串联、并联等方式尽心融合。

我们的电影信息主要来源于用户历史查看记录和购买信息，以及电影信息我们有演出人员，制作人员，年份类型。综合来看，我们在产品初期应该使用基于内容的推荐算法，因为此时用户较少，无法训练出有效的模型。而因为电影内容过大，只能根据简略的基本信息推荐，始终还是不准确的，所以在用户数达到一定规模以后，还需要用协同过滤的办法进行推荐。