数据科学与计算机学院

# 软件设计综合实验

# 订你所想

## 软件设计文档

小组成员

颜鹏翔，钟洪桢，时福源，王禹尊

# 文档管理信息表

| | |
|---|---|
| 主题 | **软件设计文档** |
| 版本 | V3.0 |
| 内容 | 对软件设计以及使用相关技术的规划与说明 |
| 关键字 | 设计说明书 订票网站 最终版本 |
| 参考文档 | |
| 创建时间 | 2017.6.20 |
| 创建人 | 王禹尊 |
| 最新发布时期 | 2017.6.24 |

# 文档变更记录

| 版本 | 日期 | 描述 | 作者 |
|---|---|---|---|
| **V0.5** | 2017.3.20 | 第一草案，在细化阶段精化 | 王禹尊 |
| **V1.0** | 2017.4.5 | 确定了后端以及前端的基本框架 | 王禹尊 |
| **V1.5** | 2017.4.9 | 在前端框架中加入 Veu<br>前端框架最终确定 | 王禹尊 |
| **V1.8** | 2017.4.11 | 产品需求确定基本，数据库结构确定 | 王禹尊 |
| **V2.0** | 2017.4.20 | 需求最终修改完成，软件设计书修改完成 | 王禹尊 |
| **V2.1** | 2017.6.5 | 数据库代码完成，对文档中相关技术的落实进行审查，并更新对照 | 王禹尊 |
| **V2.2** | 2017.6.9 | 后端代码完成，对代码中是否落实设计书内容进行审查，并更新对照 | 王禹尊 |
| **V2.3** | 2017.6.11 | 前端代码完成，对其进行审查，并更新对照 | 王禹尊 |
| **V3,0** | 2017.6.20 | 封档 | 王禹尊 |

# 文档主要评审意见

产品组

| 评审人员 | 日期 | 意见 |
|---|---|---|
| 时福源 | 2017.06.23 | |

QA 组

| 评审人员 | 日期 | 意见 |
|---|---|---|
| 颜鹏翔 | 2017.06.23 | |

# 目录

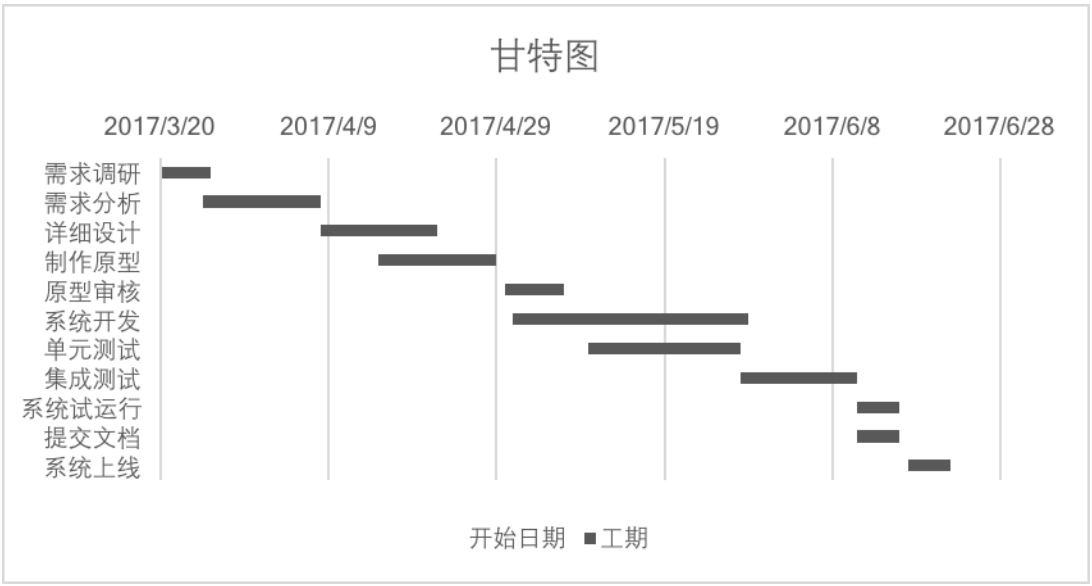
# 一、 开发规划：

## (一) 开发人员

| 学号 | 姓名 | 身份 | 邮箱 | 专业方向 |
|---|---|---|---|---|
| 14331272 | 王禹尊 | 组长 | 1411958927@qq.com | 数媒 |
| 14331327 | 颜鹏翔 | 组员 | kinpzzz@gmail.com | 计应 |
| 14331235 | 时福源 | 组员 | 1974242865@qq.com | 计应 |
| 14331389 | 钟洪桢 | 组员 | zhzdeng@qq.com | 计应 |

## (二) 开发计划

1. 甘特图




2. 开发计划简述

如图所示在项目初期阶段，首先开始需求调研。

需求调研阶段，我们将首先根据初期的会议内容考虑市场需求以及基本的市场现状，并根据以上的内容设计问卷来寻找痛点。我们准备使用第三方问卷工具，以电子问卷的方式来进行调查。初步预计会收到 200 份问卷。

在需求调查阶段，同时开展对同类型的网站的评估工作。进入初步的需求分析阶段。目标是取得现有电影购票网站的基本购票流程，并对其交互等方面作出评估。找出冗余的功能以及不必要的跳转等。

在需求调查阶段结束后，正式进入需求分析阶段。首先分析调查问卷的结果，并得出所需的结论。进一步确定用户的用例等等。结合前期进行的现有购

票网站分析，确定最终的用例以及需求。

在需求分析结束之后，技术小组根据需求小组的分析结果（UML 图，用例图等等）进行系统设计，包括架构设计、数据结构选择、确定数据库、确定代码风格、建立 GitHub 仓库等等。

完成设计工作之后，即开始初步的原型程序制作。按照初步的设计方案，进行原型程序的编码。在原型程序编码结束之后，立刻对此程序进行评估。主要的目标是确定需求设计是否有较大偏差，以及软件架构设计是否有不恰当的地方。

评审结束后需要对需求文档以及设计文档作出修订，形成最终版的需求文档以及设计文档。

确定了最终的需求以及软件设计架构之后，就进入了正式的编码阶段暨系统开发阶段。在编码的同时要求完成单元测试的内容。目标是，提交的每个子模块代码都在工程师的手中保证完成了单元测试。

在系统开发完成之后，由测试工程师以及质量保证工程师牵头进行集成测试。目标是核对系统是否有影响使用的 bug。

在完成集成测试后，进入系统试运行阶段，此阶段的负责人是项目经理、客户经理、质量保证经理。在此阶段需要完成的目标是核对整个项目是否符合预期，在功能上是否能满足要求。

以上完成后即进入发布阶段，之后将进行运维工作。

## (三) 开发环境和工具

1. 终端支持：PC
   (1) 开发语言框架：HTML5，CSS3，JavaScript
2. 服务器端支持
   (1) 语言：Java，Python
   (2) Web 框架：Struts MVC + Spring Boot，Hibernate
   (3) 关系数据库：MySQL
   (4) 负载均衡机制：Nginx
3. 开发平台与工具
   (1) IDE：eclipse
   (2) 集成与测试：Travis
   (3) 源代码管理：Github
   (4) 项目管理与自动构建：maven

### (四) 开发规范

1. WEB 前端

   语言：Javascript，html，CSS
   代码风格：JS ES5 代码风格；ES6 代码风格；CSS 代码风格；HTML/CSS 代码风格
   自动化检测工具：ESLint

2. WEB 后端

   语言：Java
   代码风格 Google Java Style（科学上网），中文翻译
   自动化检测工具：Checkstyle(Ecplise 插件安装教程)

3. 爬虫脚本：

   语言：Python3.6+
   代码风格：Python 风格规范

# 二、 总体设计

## (一) 概念术语描述（后端）

1. Java

   (1) java 是纯面向对象编程的语言；
   (2) 平台无关性 （一次编译，到处运行；Write Once，Run Anywhere）；
   (3) java 提供了许多内置的类库，通过这些类库，简化了开发人员的设计工作，同时缩短了项目开发时间；
   (4) 提供了对 Web 应用开发的支持，例如，Applet，Servlet，和 JSP 可以用来开发 Web 应用程序，，Socket，RMI 可以用来开发分布式应用程序的类库；
   (5) 去除了 c++中难以理解，容易混淆的特性（如 c++中的多继承，头文件，指针，结构，单元，运算符重载，虚拟基础类，使得程序更加严谨，整洁；
   (6) 具有较好的安全性和健壮性。java 语言经常会被用在网络环境中，为了增强程序的安全性

2. Spring

   Spring Framework(简称 Spring)是根据 Rod Johnson 著名的《Expert One-on-One J2EE Design and Development》而开发的 J2EE 应用程序框架。目

前主要根据 Rod Johnson 和 Juergen Hoeller 而进行开发的，目前发布的最新版为 1.1.4。 Spring 是 J2EE 应用程序框架，不过，更严格地讲它是针对 Bean 的生命周期进行管理的轻量级容器(Lightweight container)，可以单独利用 Spring 构筑应用程序，也可以和 Struts，Webwork，Tapestry 等众多 Web 应用程序框架组合使用，并且可以与 Swing 等桌面应用程序 API 组合。所以 Spring 并不仅仅只能应用在 J2EE 中，也可以应用在桌面应用及小应用程序中。针对 Spring 开发的组件不需要任何外部库。

优点：

(1) Spring 能有效地组织你的中间层对象。
(2) Spring 能消除在许多工程中常见的对 Singleton 的过多使用。
(3) Spring 能消除各种各样自定义格式的属性文件的需要，使配置信息一元化。
(4) Spring 能够帮助我们真正意义上实现针对接口编程。
(5) 在 Spring 应用中的大多数业务对象没有依赖于 Spring。
(6) 使用 Spring 构建的应用程序易于单元测试。
(7) Spring 支持 JDBC 和 O/R Mapping 产品(Hibernate)
(8) MVC Web 框架，提供一种清晰，无侵略性的 MVC 实现方式。
(9) JNDI 抽象层，便于改变实现细节，可以方便地在远程服务和本地服务间切换。
(10) 简化访问数据库时的例外处理。
(11) Spring 能使用 AOP 提供声明性事务管理，可以不直接操作 JTA 也能够对事务进行管理。
(12) 提供了 JavaMail 或其他邮件系统的支持。

## (二) 概念术语描述（前端）

1. Vue.js

Vue.js（读音 /vju:/，类似于 view）是一个构建数据驱动的 web 界面的库。Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

优点：

(1) 响应式编程：mvvm 框架，实现数据的双向绑定。
(2) 组件化：一切都是组件，组件可以套其他组件，增强了可复用性。
(3) 模块化：我们用一个模块打包工具来配合 Vue.js，比如 Webpack 或者 Browserify，然后再加上 ES2015。每一个 Vue 组件都可以看做一个独立的模块。
(4) 动画：Vue 自带简洁易用的过渡动画系统。有很多获奖的互动类网站

是用 Vue 开发的。Vue 的反应式系统也使得它可以用来开发高效的数据驱动的逐帧动画。

(5) 路由：Vue 本身是不带路由功能的。但是，有 vue-router 这个可选的库来配合。vue-router 可以将嵌套的路径映射到嵌套的组件，并且提供了细致的路径跳转控制。

(6) 文档和配套设施：文档和配套设施完善，社区活跃，生态系统完备，易于上手。

2. ES6（ECMAScript 6）

新一代的 javascript 也被称为 ECMAScript 6(也称为 ES6 or Harmony)。优点：

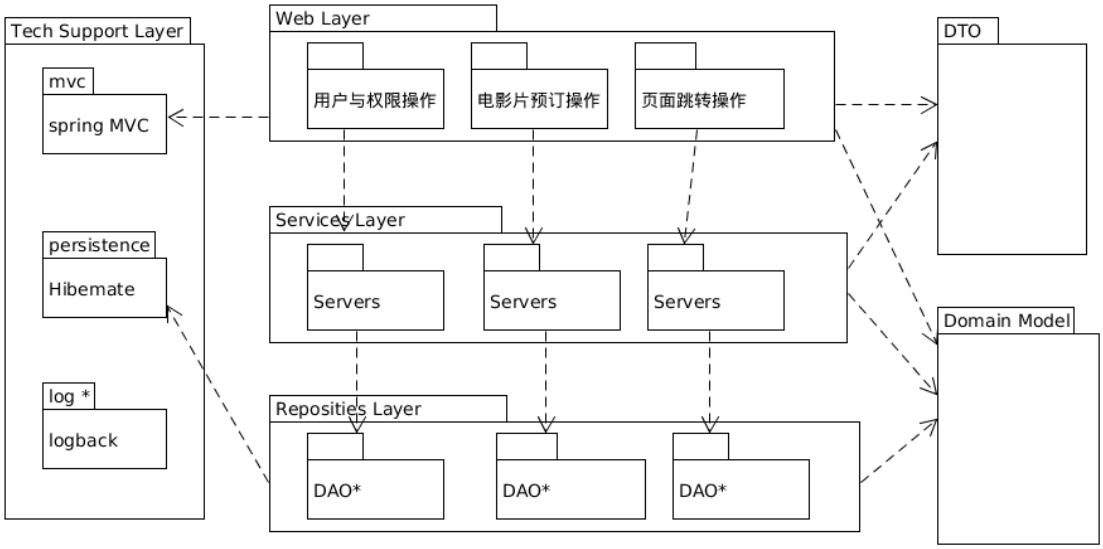
(1) 糖语法：首先，语法糖是一种语法，使得语言更容易理解和更具有可读性，它使语言相对我们来说变得更"甜"。这也意味着 ES6 的一些"新"的特点并不是真的新，只是试图简化语法而已，让我们编程更容易。这样就无需使用老式的取巧的方法编写你的代码，而是可以一种更简单的方式来编写代码，那就是使用糖语法。

(2) 模块 Module：如果你想将所有 js 放在一个文件中，或者你要在应用的不同地方使用相同的功能，你就要使用模块，关键词是 export。

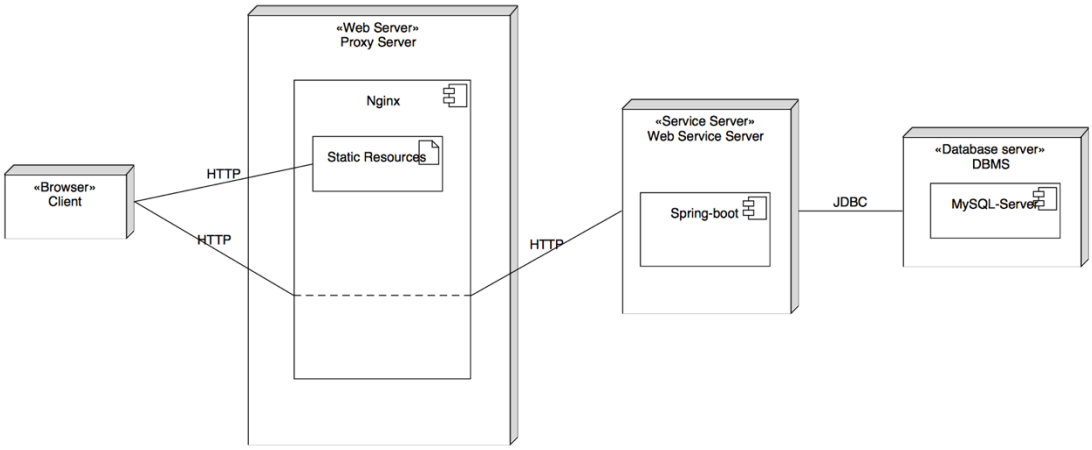
(3) let 和 const：在一段代码块中用 let 或者 const 声明的变量会限制它们只在这个块中可见。这叫做块级作用域。

## (三) 基本设计描述

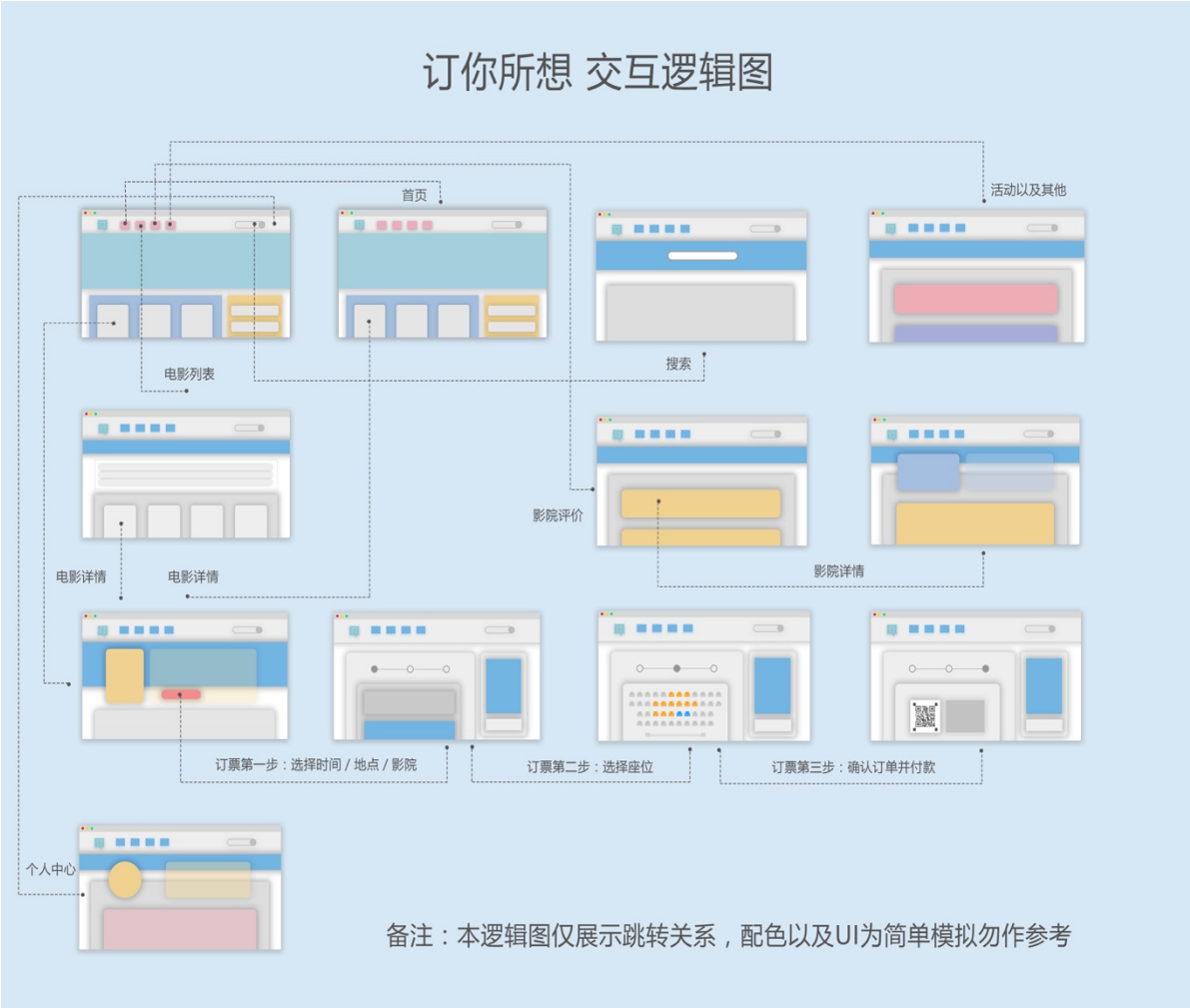
1. 系统总体逻辑结构图



2. 系统部署结构图

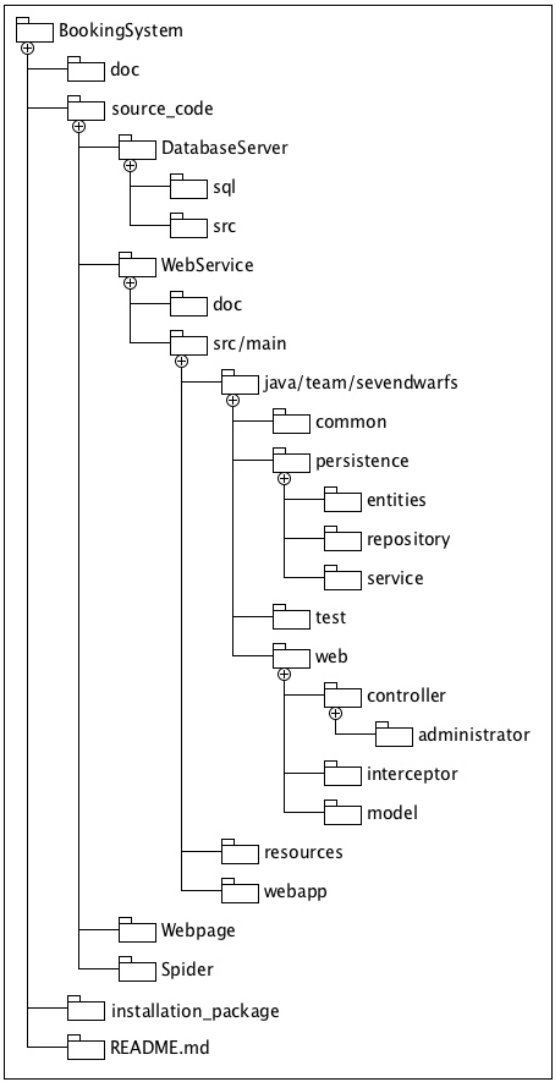



## (四) 主要界面流程描述

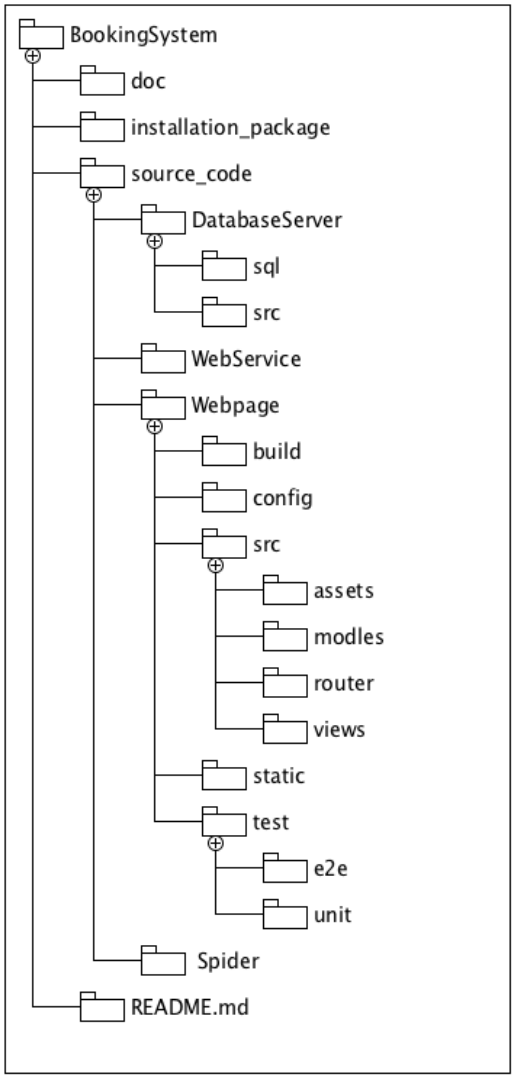


## (五) 模块列表

1. 模块划分

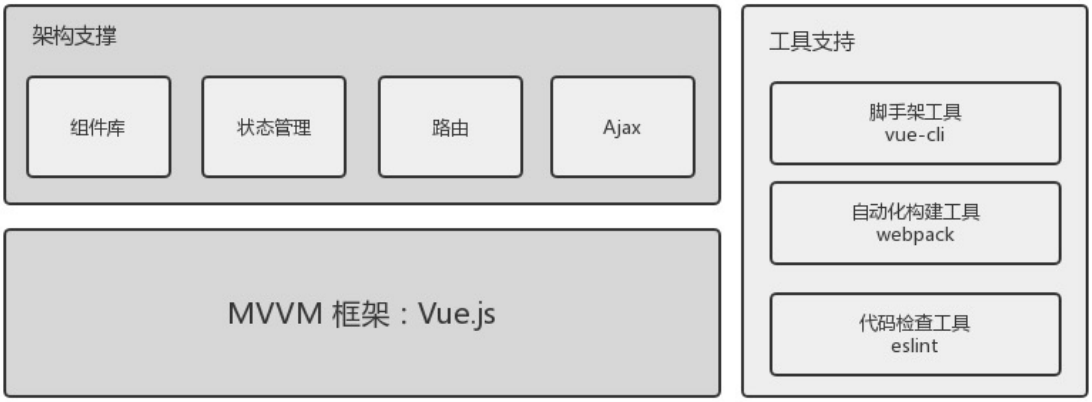



后端




前端

2. 前端结构



# 三、 接口规范

## (一) Web 服务器

1. 返回的状态码

| 类型 | stateCode | info |
| --- | --- | --- |
| 成功 | 200 | NULL |
| 错误 | 500 | 错误信息 |

2. 用户登录/注册

| 路由 | 方法 | 说明 | 提交格式 | 测试 |
| --- | --- | --- | --- | --- |
| /api/login | POST | 提交用户登录表单 username, password, 允许邮箱/手机/用户名登录 | | OK |
| /api/signup | POST | 提交用户注册表单 username, password, email, phone | | OK |
| /api/logout | PUT | 登出 | | OK |
| /api/user | GET | 获取当前用户信息 | | OK |
| /api/user | PUT | 修改当前用户信息，填写需要修改的项, username, email, phone, oldPassword, newPassword | | OK |
| /api/user/order | GET | 查看该用户的所有订单 | OrderModel | OK |
| /api/user/screen/{id} | PUT | 锁定/购买座位，需要上传 需要用户登录后 | seat={88 长字符串，锁定的位置用 1 表示，购买位置用 2 表示，其他用 0 填充} | |

购票例子:
表单格式:

```
seat=11000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
// 表示锁定第 1，2 个位置
seat=00220000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

```
// 表示购买第 1，2 个位置
// 购买前需要先锁定
public class OrderModel {
        private List<FilmOrder> filmOrderModelList;
}
public class FilmOrder {
    private Integer id;
    private User user;
    private Integer screenId;
    private String seat;
}
```

3. 管理员账号

| 路由 | 方法 | 说明 | 提交格式 |
| --- | --- | --- | --- |
| /api/admin/login | POST | adminname，password 登录 | |
| /api/admin/logout | PUT | 登出 | |
| /api/admin/newMovie | POST | 需要填写的域 chineseName,englishName,pictureUrl,type,length,releaseDate,introduction | |
| /api/admin/{id} | DELETE | 当初对应 id 的电影 | |
| /api/admin/{id} | PUT | 更新电影信息，只需要填写需要更新的域，和创建电影的域名字相同 | |
| /api/admin/cinema/{id} | DELETE | 参数对应 id 的影院 | |
| /api/admin/cinema/create | POST | 创建一个影院 | CinemaModel |
| /api/admin/cinema/{id} | PUT | 修改一个影院信息 | CinemaModel |

```
CinemaModel {
    private String name;
    private String address;
    private String phone;
    private List<Screen> screens;
}
```

4. 获取电影信息

| 路由 | 方法 | 说明 | 返回值 | 测试 |
| --- | --- | --- | --- | --- |
| /api/movie/name/{查询电影名} | GET | 返回电影名对应信息，允许查询中英文电影名，返回一条记录或空 | SimpMovie | OK |
| /api/movie/type/{type}?id=ID | GET | 返回电影类型列表，数目为从 id 开始往后 20 条，默认 id = 0 | List | OK |
| /api/movie/date/day/20170501 | GET | 返回 2017-05-01 上映的电影列表，如果输入非法日期，返回当天上映列表 | List | OK |
| /api/movie/date/month/201705 | GET | 返回 2017-05 上映的电影列表，如果输入非法日期，返回当月上映列表 | List | OK |
| /api/movie/date/year/2017 | GET | 返回 2017 上映的电影列表，如果输入非法日期，返回当年上映列表 | List | OK |
| /api/movie/{id} | GET | 返回 ID=id 的电影详细信息 | Movie | OK |
| /api/movie/showing/{number} | GET | 返回最近一个月上映的电影列表,number 条 | List | OK |
| /api/movie/query/count?type={}&area={}&year={} | GET | year=2007，允许 type,area,year 字段为"all" | Integer | OK |
| /api/movie/query?type={}&area={}&year={}&page={}&step={} | GET | 返回 [page*stap, page*step+step]的数据,允许 type,area,year 字段为"all" | List | OK |

```
SimpMovie {
    private String name;
    private Integer id;
    private String url;
}
```

5. 获取演员/导演信息

| 路由 | 方法 | 说明 | 返回值 | 测试 |
| --- | --- | --- | --- | --- |
| /api/person/{id} | GET | 通过演员/导演的 ID 获取 | Person | OK |
| /api/person/movie/{id} | GET | 获取电影 ID 的演员/导演名单 | List | OK |

```
Person {
    private Integer id;
    // 名字
    private String name;
    // 照片的 URL
    private String url;
    // 表示是导演还是演员
    private String type; // "actor", "director"
}
```

6. 获取影院信息

没有说明默认返回影院简要信息：影院 id，影院名字 name | 路由 | 方法 | 说明 | 接受内容 | 返回值 | 测试 | | :-----------------------------: | :--: | :-------------------: | ---- | :-------------: | :--: | | /api/cinema?number={}&address={} | GET | number 选填默认 10，address 必填 | | List | | | /api/cinema/{id} | GET | 返回影院详细信息 | | Cinema | OK | | /api/cinema/showing?id={id} | GET | 返回正在该影院上映的电影简要信息列表 | | List | OK |

```
SimpCinema {
    private Integer id;
    private String name;
}

Cinema {
    private Integer id;
    private String name;
    private String address;
    private String phone;
    private List<Screen> screens;
}
```

7. 获取排片信息

| 路由 | 方法 | 说明 | 接受内容 | 返回值 | 测试 |
| --- | --- | --- | --- | --- | --- |
| /api/screen?cinemaid={}&mov | GET | 获取对应影院对应电影的排片情况列表 | | List | OK |

| | | | | | |
|---|---|---|---|---|---|
| ieid={}&date={}&time={} | | | | | |
| /api/screen/{id} | GET | 获取对应 id 的排片情况 | | Screen | OK |

```
Seat {
    private List<Integer> vacancy;
    private List<Integer> soldOut;
    private List<Integer> locking;
}

Screen {
    private Integer id;
    private Date time;
    private String language;
    private String room;
    private Double price;
    private Cinema cinema;
    private String movieName;
    private String seats; // '0'->空位,'1'->被锁定,'2'->已售出 8x11 列优先，比如 2 行 1 列下标为 8
}
```

8. 搜索功能

| 路由 | 方法 | 说明 | 返回值 | 测试 |
|---|---|---|---|---|
| /api/search?query={} | | | | |

## (二) 数据库

1. 配置与安装

- Requirements
  - docker
- Build docker image

```
$ docker build -t db-server .
```

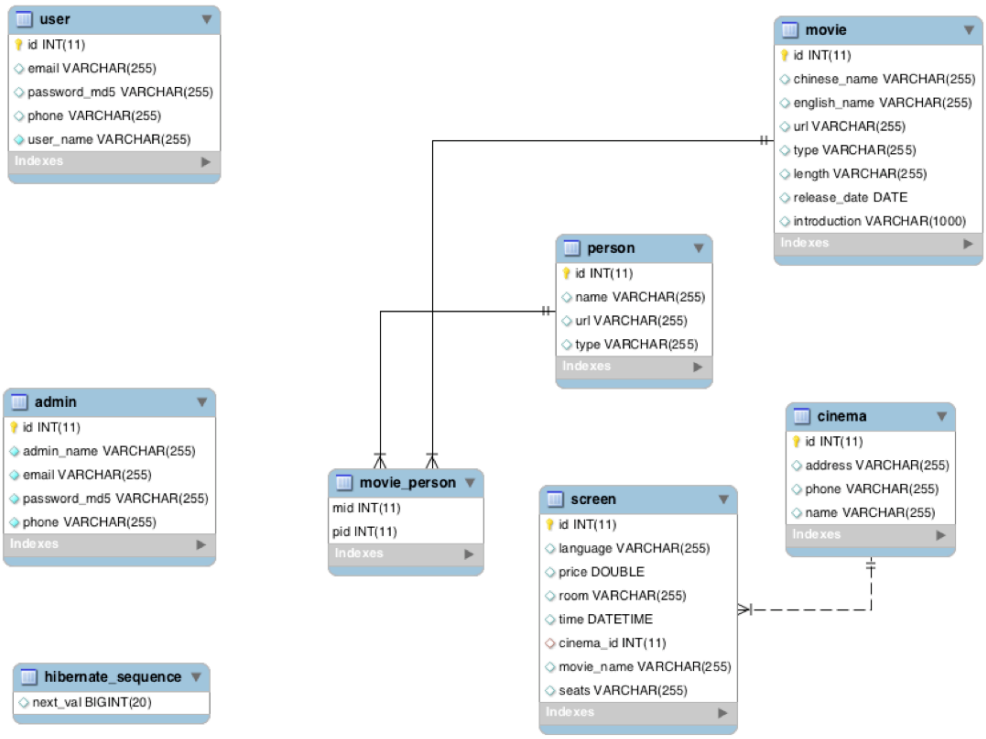
- Run docker container

```
$ docker run -d --name db db-server
```

- need to stop web-service server container before building and build after building

```
docker rm restful-server
docker rmi kinpzz/restful-server
docker build -t kinpzz/restful-server ../WebService
docker run -d -p 127.0.0.1:8082:8082 --name restful-server --link db:db-server kinpzz/restful-server
```

## 2. Model




user 表：用户表，记录用户的信息，用户名，密码的 MD5，电话，邮箱

movie 表：记录电影的信息，包括中文名，英文名，电影类型，电影时长，上映日期，电影简介， 电影海报的 URL，参演人员名单

person 表：记录电影人的信息，通过 type 列区别是导演还是演员，包括名字，照片 URL，type 电 影人的类型（导演/演员）

cinema 表：订单编号，电影 id、影院 id、场次 id、座位 id screen 表：荧屏 id，语言，价格，房间 id，时间，影院 id，电影名字，座位 id

admin 表：id，名字，密码，email，电话号码

movie 表和 person 表是一对多的关联映射关系

# 四、 软件设计技术

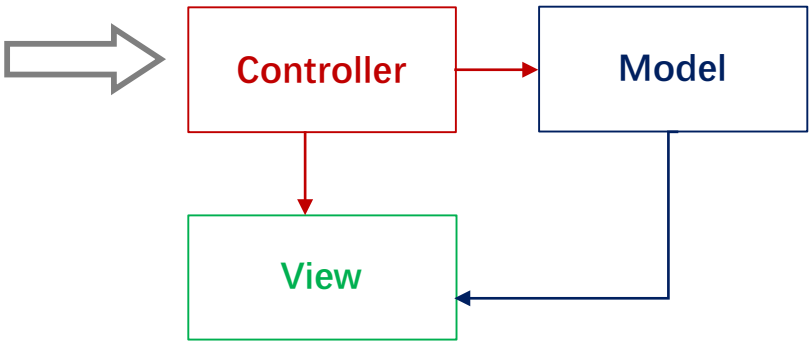
## (一) 前后端分离

1. 理解 MVC

MVC 是一种经典的设计模式，全名为 Model-View-Controller，即模型-视图-控制器。

其中，模型是用于封装数据的载体，例如，在 Java 中一般通过一个简单的 POJO（Plain Ordinary Java Object）来表示，其本质是一个普通的 java Bean，包含一系列的成员变量及其 getter/setter 方法。对于视图而言，它更加偏重于展现，也就是说，视图决定了界面到底长什么样子，在 Java 中可通过 JSP 来充当视图，或者通过纯 HTML 的方式进行展现，而后者才是目前的主流。模型和视图需要通过控制器来进行粘合，例如，用户发送一个 HTTP 请求，此时该请求首先会进入控制器，然后控制器去获取数据并将其封装为模型，最后将模型传递到视图中进行展现。

综上所述，MVC 的交互过程如图 1 所示。




2. MVC 模式的优点与不足

MVC 模式早在上个世纪 70 年代就诞生了，直到今天它依然存在，可见生命力相当之强。MVC 模式最早用于 Smalltalk 语言中，最后在其它许多开发语言中都得到了很好的应用，例如，Java 中的 Struts、spring MVC 等框架。正是因为这些 MVC 框架的出现，才让 MVC 模式真正落地，让开发更加高效，让代码耦合度尽量减小，让应用程序各部分的职责更加清晰。

既然 MVC 模式这么好，难道它就没有不足的地方吗？我认为 MVC 至少有以下三点不足：

每次请求必须经过“控制器->模型->视图”这个流程，用户才能看到最终的展现的界面，这个过程似乎有些复杂。

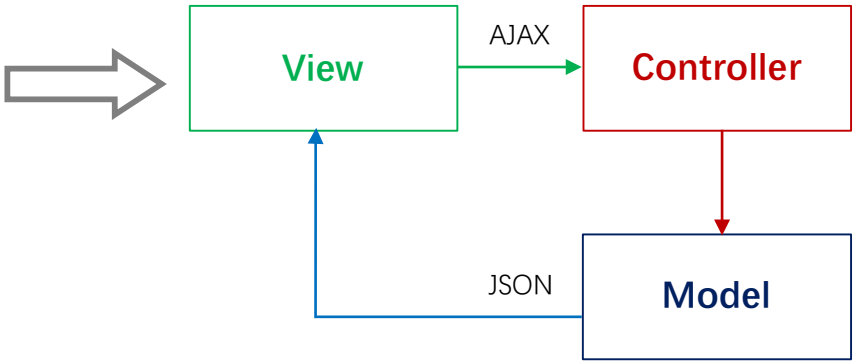
实际上视图是依赖于模型的，换句话说，如果没有模型，视图也无法呈现

出最终的效果。

渲染视图的过程是在服务端来完成的，最终呈现给浏览器的是带有模型的视图页面，性能无法得到很好的优化。

为了使数据展现过程更加直接，并且提供更好的用户体验，我们有必要对 MVC 模式进行改进。不妨这样来尝试，首先从浏览器发送 AJAX 请求，然后服务端接受该请求并返回 JSON 数据返回给浏览器，最后在浏览器中进行界面渲染。

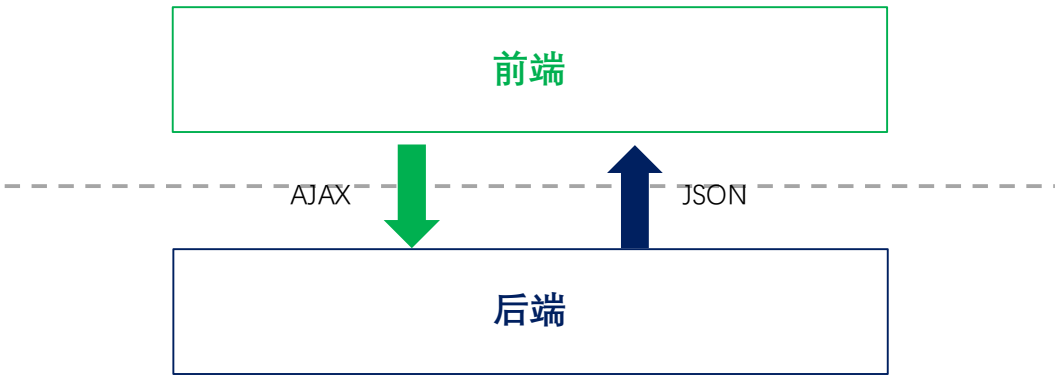
改进后的 MVC 模式如图 2 所示。




也就是说，我们输入的是 AJAX 请求，输出的是 JSON 数据，市面上有这样的技术来实现这个功能吗？答案是 REST。

REST 全称是 Representational State Transfer（表述性状态转移），它是 Roy Fielding 博士在 2000 年写的一篇关于软件架构风格的论文，此文一出，威震四方！国内外许多知名互联网公司纷纷开始采用这种轻量级的 Web 服务，大家习惯将其称为 RESTful Web Services，或简称 REST 服务。]

如果将浏览器这一端视为前端，而服务器那一端视为后端的话，可以将以上改进后的 MVC 模式简化为以下前后端分离模式，如图 3 所示。



可见，有了 REST 服务，前端关注界面展现，后端关注业务逻辑，分工明确，职责清晰。那么，如何使用 REST 服务将应用程序进行前后端分离呢？我们接下来继续探讨，首先我们需要认识 REST。

3. 认识 REST

REST 本质上是使用 URL 来访问资源种方式。众所周知，URL 就是我们平常使用的请求地址了，其中包括两部分：请求方式与请求路径，比较常见的请求方式是 GET 与 POST，但在 REST 中又提出了几种其它类型的请求方式，汇总起来有六种：GET、POST、PUT、DELETE、HEAD、OPTIONS。尤其是前四种，正好与 CRUD（Create-Retrieve-Update-Delete，增删改查）四种操作相对应，例如，GET（查）、POST（增）、PUT（改）、DELETE（删），这正是 REST 与 CRUD 的异曲同工之妙！需要强调的是，REST 是“面向资源”的，这里提到的资源，实际上就是我们常说的领域对象，在系统设计过程中，我们经常通过领域对象来进行数据建模。

REST 是一个“无状态”的架构模式，因为在任何时候都可以由客户端发出请求到服务端，最终返回自己想要的数据，当前请求不会受到上次请求的影响。也就是说，服务端将内部资源发布 REST 服务，客户端通过 URL 来访问这些资源，这不就是 SOA 所提倡的“面向服务”的思想吗？所以，REST 也被人们看做是一种“轻量级”的 SOA 实现技术，因此在企业级应用与互联网应用中都得到了广泛应用。

下面我们举几个例子对 REST 请求进行简单描述：可以查看 API 来更好地理解。

可见，请求路径相同，但请求方式不同，所代表的业务操作也不同，例如，/advertiser/1 这个请求，带有 GET、PUT、DELETE 三种不同的请求方式，对应三种不同的业务操作。

虽然 REST 看起来还是很简单的，实际上我们往往需要提供一个 REST 框架，让其实现前后端分离架构，让开发人员将精力集中在业务上，而并非那些具体的技术细节。下面我们将使用 Java 技术来实现这个 REST 框架，整体框架会基于 Spring 进行开发。

## (二) Vue 渐进式框架

1. 为什么要有框架

(1) 框架的存在是为了帮助我们应对复杂度

前端框架特别多，那么为什么要有框架呢？框架的存在是为了帮助我们应对复杂度 。当我们需要解决一些前端上工程问题的时候，这些问题会有不同的

复杂度。如果你用太简陋的工具应对非常复杂的需求，就会极大地影响你的生产力。所以，框架本身是帮我们把一些重复的并且已经受过验证的模式，抽象到一个已经帮你设计好的 API 封装当中，帮助我们去应对这些复杂的问题。

(2) 框架自身也有 复杂度

框架本身也会带来复杂度。相信大家在调研各种框架或学习各种框架时，会遇到学习曲线问题——有些框架会让人一时不知如何上手。

(3) 工具复杂度是为了处理内在复杂度所做的投资

工具的复杂度是可以理解为是我们为了处理问题内在复杂度所做的投资。为什么叫投资？那是因为如果投的太少，就起不到规模的效应，不会有合理的回报。这就像创业公司拿风投，投多少是很重要的问题。如果要解决的问题本身是非常复杂的，那么你用一个过于简陋的工具应付它，就会遇到工具太弱而使得生产力受影响的问题。

反之，是如果所要解决的问题并不复杂，但你却用了很复杂的框架，那么就相当于杀鸡用牛刀，会遇到工具复杂度所带来的副作用，不仅会失去工具本身所带来优势，还会增加各种问题，例如培训成本、上手成本，以及实际开发效率等。

(4) Pick the right tool for the job

“Pick the right tool for the job”——在国外，跟开发者讨论一些框架选型问题时，大家都会说这句话——一切都要看场景。因为，前端开发原生开发或者桌面开发模式相比，有自己的独特之处，它跟其实并不那么固定。在 Web 上面，应用可以有非常多的形态，不同形态的 Web 应用可能有完全不同程度的复杂度。这也是为什么要谈工具复杂度和所要做的应用复杂度的问题。

(5) 怎么看前端框架的复杂度

目前的前端开发已经越来越工程化，而我们需要解决的实际问题也是不同的。我们就下图进行分析。


我们可能在任何情况下都需要 **声明式的渲染功能** ，并希望尽可能避免手动操作，或者说是可变的 **命令式操作** ，希望尽可能地让 DOM 的更新操作是自动的，状态变化的时候它就应该自动更新到正确的状态；我们需要**组件系统** ，将一个大型的界面切分成一个一个更小的可控单元； **客户端路由** ——这是针对单页应用而言，不做就不需要，如果需要做单页应用，那么就需要有一个

URL 对应到一个应用的状态，就需要有路由解决方案； **大规模的状态管理** ——当应用简单的时候，可能一个很基础的状态和界面映射可以解决问题，但是当应用变得很大，涉及多人协作的时候，就会涉及多个组件之间的共享、多个组件需要去改动同一份状态，以及如何使得这样大规模应用依然能够高效运行，这就涉及大规模状态管理的问题，当然也涉及到可维护性，还有**构建工具** 。现在，如果放眼前端的未来，当 HTTP2 普及后，可能会带来构建工具的一次革命。但就目前而言，尤其是在中国的网络环境下，打包和工程构建依然是非常重要且不可避免的一个环节。

2. 渐进式框架 Vue.js

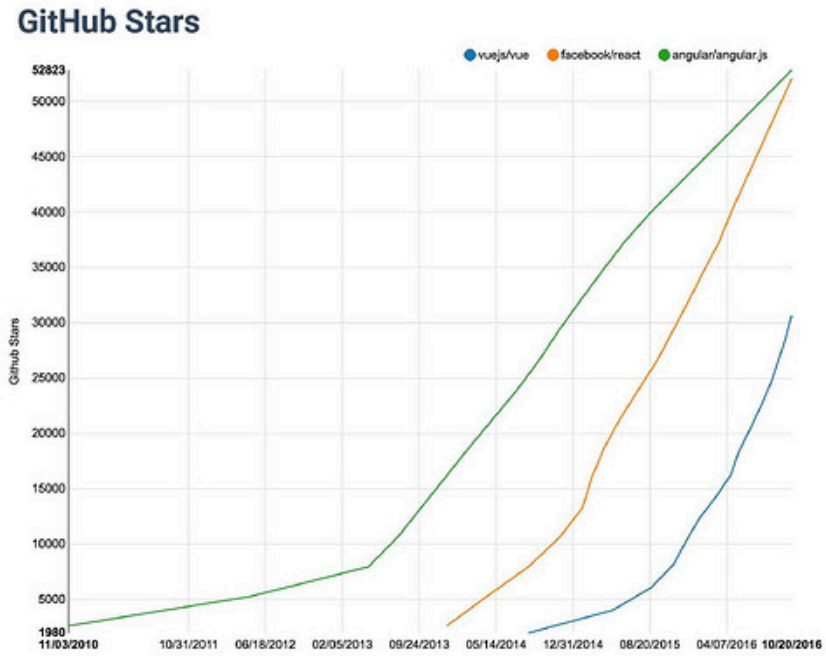
(1) Vue.js 现状

以下数据可以体现出 Vue. js 的现状。

**现状**

- ~30k stars on GitHub
- ~185k/mo downloads on NPM
- ~264k/mo unique visitors to vuejs.org
- ~55k weekly active Chrome extension users

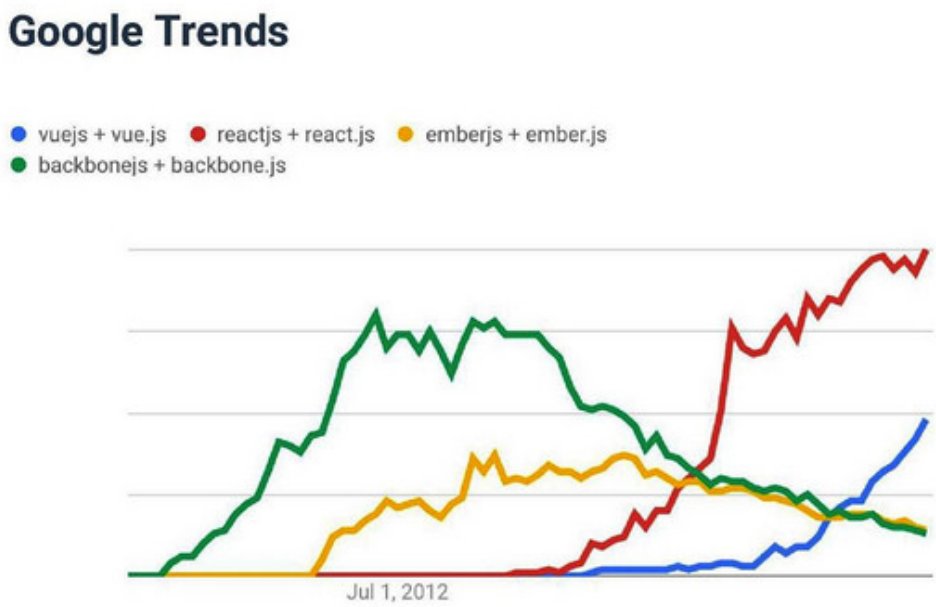
前一段时间突破了三万星（如下图所示），总下载量过百万。




官网上每个月的用户量为 26 万，这个应该是不包含中国区数据。官方开发者插件的周活跃用户数在 5 万 5 左右。这个数据是我觉得最有说服力的数据。

安装并且使用开发者插件的 Vue 用户，应该会在实际生产中真正频繁使用 Vue。

Google 搜索趋势的相关数据如下图所示。图中，绿色的是 Backbone 的数据，黄色是 Ember，红色是 React，蓝色是 Vue。可以看出 React 和 Vue 近两年发展势头都比较迅猛。可以看出，Vue 的曲线开始的是很早，2013 年已经开始，但是有很长一段时间的增长是比较低的。因为在那一段时间我还在谷歌工作，Vue 基本上是作为个人项目在运营。在过去一两年中，Vue 获得了非常大的突破性发展。这个图里没有 Angular，因为 Angular 的量还是非常大的，如果放进去就破表了。




这些数据并不能绝对地代表框架当前的热度，但有一定的参考价值。可以看到 React 的势头很足。而由 Vue 的曲线还可以看出它的增长速度还在不停上扬。

(2) Vue 的定位

它与其他框架的区别就是渐进式的想法，也就是“Progressive”——这个词在英文中定义是渐进，一步一步，不是说你必须一竿子把所有的东西都用上。

(3) Vue 的设计

接下来我们回到之前看的图：


Vue 从设计角度来讲， 虽然能够涵盖这张图上所有的东西，但是并不需要一上手就把所有东西全用上 ，因为没有必要。无论从学习角度，还是实际情况，这都是可选的。声明式渲染和组建系统是 Vue 的核心库所包含内容，而客户端路由、状态管理、构建工具都有专门解决方案。这些解决方案相互独立，你可以在核心的基础上任意选用其他的部件，不一定要全部整合在一起。

# 五、 附录

## (一) 参考资料


1. https://cn.vuejs.org
2. http://blog.csdn.net/shaobingj126/article/details/49420145
3. https://github.com/SevenDwarfs/Deployment
4. https://github.com/SevenDwarfs/Dashboard


## (二) 附加文档

1. 表 1《小组分工与贡献率》

| 学号 | 姓名 | 分工 | 贡献率 |
|---|---|---|---|
| 14331272 | 王禹尊 | 项目管理、软件设计、UI 设计、交互设计、项目文档 | 25% |
| 14331327 | 颜鹏翔 | 项目部署、持续集成、前端开发、项目文档 | 26% |
| 14331235 | 时福源 | 需求分析、爬虫设计与开发、项目文档 | 23% |
| 14331389 | 钟洪桢 | 后端开发、数据库设计 | 26% |

2. 表 2 《制品与贡献率》

| | 制品 | 王禹尊 | 颜鹏翔 | 时福源 | 钟洪桢 |
|---|---|---|---|---|---|
| 源码 | 后端：WebService | - | - | - | 100% |
| | 前端：Webpage | 15% (UI) | 85% | - | - |
| | 数据库：DatabaseServer | - | - | - | 100% |
| | 爬虫：Spider | - | - | 100% | - |
| 文档 | 软件需求规格说明书(SRS) | - | - | 100% | - |
| | 软件设计文档(SD) | 100% | - | - | - |
| | 安装部署说明 | - | 100% | - | - |