软件设计说明书

——Popush2

Version 1.0

编写者

Prepared by

团队：**扬帆启程**

Group Name: Sailon

陈华榕 **Huarong Chen**	**2011013236**	**chenhuarongzp@gmail.com**
文庆福 **Qingfu Wen**	**2011013239**	**thssvince@163.com**
庄晨帆 **Chenfan Zhuang**	**2011013246**	**zhuangchenfan@gmail.com**
杨　磊 **Lei Yang**	**2011013256**	**yl93528@gmail.com**
林维妮 **Wheini Lin**	**2013400888**	**winnie180690@gmx.de**

Instructor: 刘强，刘璘

Qiang Liu, Lin Liu

Course: 软件工程（3）

Software Engineering (3)

Teaching Assistant: 钱康来，司徒静弘，龚云飞，王得希

Kanglai Qian, Jinghong Situ, Yunfei Gong, Dexi Wang

Date: **2013/11/09**

目录

文档修订

版本号 Version	主要作者 Primary Author(s)	简述 Description of Version	完成时间 Date Completed
1.0	文庆福、杨磊、庄晨帆、陈华榕	详细阐述 Popush2 的详细开发、设计、实现情况。	2013/11/09

1 简述

1.1 文档目的

Popush 是清华大学 2012~2013 学年春季学期《软件工程》课程中，由清华大学软件学院 2010 级 Popush 团队开发的在线协作编程平台（1.0 版本）。2013~2014 学年秋季学期《软件工程（3）》课程中，在原开发团队许可下，由清华大学软件学院 2011 级同学对 1.0 版本进行升级，重点优化其前端代码及少量后端代码，Sailon 团队将原项目升级成为 Popush2。

本文档将对 Popush2 进行尽可能详尽的描述，希望读者能够通过本文档深入地了解我们的开发目的、开发过程与开发结果，特别是我们重构的思路以及重构结果。

1.2 产品概述

Popush 是一个在线协作编程平台，通过线上的多人协同编程、聊天、代码运行、调试等，解决实际开发中多人协同编程时容易遇到的物理距离、代码冲突等问题。

通过本平台，能促进更多优质代码的产生，而且有助于大大缩短开发时间，整体提升多人开发效率，克服开发环境的依赖，真正实现随时、随地编程。

1.3 文档概述与受众

本说明书中第 2 部分简单介绍我们的开发团队及开发管理一些情况，第 3 部分是系统设计与实现的详细阐述，第 4 部分重点关注系统的交付并对未来维护提出建议。

本说明书受众包括开发人员、维护人员、课程教师、助教等。

受众	说明	推荐重点阅读的章节
开发人员	非项目原班开发人员需要先对本系统的设计进行较深入的了解，之后才能进行扩展，本说明书是了解项目的首要参考资料。	若对项目已有一定了解，推荐重点阅读第 3 部分。若对项目没有任何了解，推荐阅读所有章节。
维护人员	除了对平台进行扩展的开发人员，还有对平台进行维护的维护人员，他们同样要对项目有深入的了解，推荐阅读本说明书。	推荐重点阅读第 3、4 部分。
课程教师和助教	通过此说明书主要了解项目一的开发、设计情况。	推荐重点阅读第 3 部分和所有加星号（*）的章节。进行功能验收可重点参考 4.1 章节。

1.4 格式约定

本说明书遵从一定的格式要求，**标题中文使用黑体并逐级减小字号**，***四级标题开始不减小字号并使用斜黑体***，正文中文使用宋体，英文统一使用 Arial 字体，正文字号 11px，单倍行距，段落首行缩进 2 字符，*斜宋体正文作为评注*，**加粗宋体正文作为强调**，**带星号（*）的标题**表示与项目一检查要求中的加分项有关。

本说明书中涉及的文件路径，若是 js 文件，如无特殊说明，默认其存放于 static/app/js 中，其他文件则默认存放于 static/app 中。

1.5 参考文献与致谢

[1] Popush 设计文档 V1.0

[2] 03_SEProject1.pdf

[3] 软件需求规格说明书——Popush Version 1.1 by Sailon

[4] 面向对象技术 UML 教程，王少峰，清华大学出版社

在此特别感谢 Popush 1.0 版本开发团队的付出与无私，没有他们的优质工作作为铺垫，我们就不可能将重构注意力几乎完全集中在前端页面。

我们还特别感谢刘璘老师，她的教学和课外指导帮助我们较好地将面向对象建模技术应用在项目的分析与构建中。

特别感谢司徒静弘、钱康来、龚云飞、王得希四位编外编内助教，他们对软件质量的严格把控促进我们提高要求，做出更好的项目，也在平时对我们进行了耐心的指导。

当然还要特别感谢刘强老师，她不仅十分关心我们的项目进展、学习情况等，还细致入微关心我们的日常生活，带给我们的有欢乐、有紧张、有鼓舞、有霸气、有感动，是良师更是益友！~~（而且还是土豪）~~

最后感谢所有关注关心本项目以及所有为本项目付出辛勤劳动的人们！

2 开发规范

2.1 开发人员

扬帆启程（Sailon）团队成立于 2013 年 9 月，共有五名成员，他们是：

姓名	简介	分工	备注
陈华榕	码农	数据封装与控制逻辑	组长
文庆福	大叔	页面实现	
庄晨帆	女神	数据封装与控制逻辑	
杨磊	男神	页面实现	
林维妮	华裔	页面设计	德国交换生

Happy studying, happy working, happy living.

一个多月以来，团队五位成员同心协力，乘风破浪，为 Popush2 的开发投入了大量精力，正是由于大家的努力才得以让项目顺利按时交付。

2.2 开发管理*

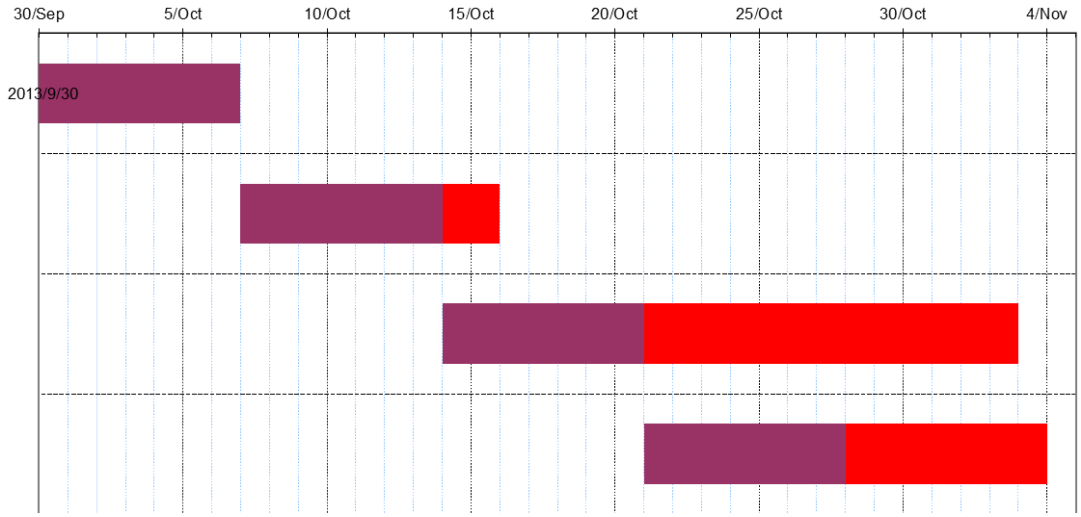
我们使用了合理有效的团队管理和项目管理方式，具体包括：

- 资料共享：ColorWork
- 版本控制：Github
- 交流通讯：QQ，飞信

我们每周都进行三次集中开发，分别是周一下午、周二上午、周六全天，累计集中开发时间超过 70 小时。

2.3 开发进度

序号	任务名称	计划工时	开始时间	计划完成时间	实际完成时间	实际工时	提前	滞后
1	确定并熟悉框架，细化分工	7.0	2013/9/30	2013/10/6	2013/10/6	7.0	**0.0**	**0.0**
2	确定页面结构，完成用户管理	7.0	2013/10/7	2013/10/13	2013/10/15	9.0		**2.0**
3	文件管理与协同编辑页面与功能	7.0	2013/10/14	2013/10/20	2013/11/2	20.0		**13.0**
4	完成其余功能，项目整体的功能测试,性能测试与优化	7.0	2013/10/21	2013/10/27	2013/11/3	14.0		**7.0**



2.4 开发环境与工具

Linux 平台

软件	版本	软件	版本	软件	版本
Ubuntu Server	13.04	GCC	4.7	Ruby	2.0
Nginx	1.5	GDB	7.5	Lua	5.2
Node	0.10	Python	2.7	JDK	7.0
MongoDB	2.2	Perl	5.14		

Windows 平台

软件	版本	软件	版本
Windows 8	Enterprise	MinGW32-gdb	7.6.1-1
Nginx	1.5.5	Python	3.3.2
Node	0.10.17	Perl	5.16.3
MongoDB	2.4.6	Ruby	1.8.6
MinGW32-gcc	4.8.1-3	MinGW32-lua	5.2.0-1
MinGW32-gcc-g++	4.8.1-3	JDK	1.7.0_25

开发工具包括 Sublime Text 2、Chrome 开发者工具等。

3 设计与实现

3.1 设计思路

我们已经有一款比较成熟的在线协同编程平台 Popush 了，但经过几个月的使用，该产品暴露出一些问题，既有功能方面的也有非功能方面的。此次项目的项目重构，重点关注原平台的代码质量，通过重构期望能得到一个易于维护、易于扩展、更加稳定的系统。

3.1.1 基础成果

原版 Popush 已经能够高效稳定地进行文件目录管理、文件共享、代码协同编辑、协同运行、调试等，是一款能投入实际使用的在线编程协作平台。

从功能上看，原版 Popush 是成功的，但从工程上看，原版 Popush 是以敏捷开发的方式快速构建的，主要在可维护性、可扩展性方面存在不足，在前端方面尤为明显。

3.1.2 重构思路

此次前端代码重构的主要目的是希望将前端代码模块化，使得更加易于维护。为使得前端重构更加便捷，我们决定采用一款开源的前端 MVC 框架，从设计模式上实现前端代码的数据层、逻辑层以及表现层的分离。

首先我们将页面上的所有数据都抽象成模型进行封装，然后利用控制器来建立起表现层与数据层之间的联系，解耦表现层和数据层。此外，我们按照不同的功能将前端进行功能模块的划分，按照不同的功能模块来组织代码。希望通过以上两点使得重构能够实现高内聚低耦合的效果。

3.1.3 框架选择

Angular.js 是 Google 公司开源的一套前端 MVC 框架，本系统选用 Angular.js 基于如下原因：

- 数据双向绑定。数据模型简单，可同步更新 DOM 元素和 Model 中的变量，减少甚至避免严重影响表现层与数据层解耦的 DOM 操作。
- 模块清晰。清晰的功能划分，在 angular.module.config 里进行 router 的配置；angular.module.controller 里写业务逻辑；在 angular.module.directive 里写视图行为；在 angular.module.filter 中对页面数据格式化。
- 依赖注入。通过添加简单的服务，Angular.js 可侦测得到核心的服务，注入到需要的控制器中，完美实现多个控制器的数据共享。
- 兼容良好。支持主流的高级浏览器，对 IE 方面，它兼容 IE8 及以上的版本。
- 可测试。Angular 是一个可测试框架，支持点对点的单元测试。

3.1.4 创新设计*

由于选用了 Angular.js 框架，原前端的代码很难直接使用。这既是坏事也是好事，坏事是这样大大增加了我们进行前端重构的工作量，而好处是我们可以摆脱原前端设计的束缚，完全独立地设计、开发出全新的 Popush2 前端页面并大胆地增加用户急需的一些功能支持。

以下列举我们进行的一些创新设计，每一项都从用户角度出发，有用户需求支撑。各个创新设计的具体功能、界面流程等渗透在后文的系统描述中，以下只对相应设计进行简单表述并阐述如此设计的原因。

3.1.4.1 注册后自动登录

这是一个小改进，但意义不小。

用户注册结束后立刻自动登录，即刻就能使用，这样的设计在很多成熟的产品中被采用了。我们希望用户能先使用系统，觉得好用再成为我们的持久用户，因此我们采取了这样的设计。确实，注册时已经输过一遍用户名和两遍密码了，没必要让用户再输一遍。

3.1.4.2 集成的消息中心

随处可能出现的提示条、提示文字等，影响布局，影响整体美感，还不易管理。受Linux桌面的启发，我们实现了集成的消息中心（在页面的右上角以浮动方式出现消息）。这样一定程度上增进了视觉感受，而且大大简化了消息的管理方式，我们通过封装，能通过简单的一句调用进行消息的增删。

3.1.4.3 集成的设置中心

保持交互界面的整体性，我们尽可能地不使用菜单等容易影响其他交互的交互方式。为此设计了集成的设置中心，通过点击用户头像可进入，之后可进行所有关于用户的设置。

3.1.4.4 目录结构文件树

在原版 Popush 中，如何很快地找到某个文件？只要知道文件的路径，很快就能找到。那如何很快地找到两个文件？受 Windows 资源管理器文件树的启发，我们实现了 Popush 目录结构的文件树。

现在看这个功能似乎不那么重要，但长远来看，未来 Popush 开始以项目进行代码组织了以后，文件树是必不可少的，否则会给用户带来诸多不便，影响体验。

3.1.4.5 多标签支持

在原版 Popush 中，很难在两个文件或目录间进行切换，而未来 Popush 开始以项目进行代码组织后，这样的功能会很常用，因此我们设计了多标签的工作区，每个标签可以是目录或文件，也可以是设置中心，用户可以自由地进行标签的管理。

特别地，原版 Popush 中一个用户只能同时进入一个 room，应用多标签后，如果在两个文件之间切换，实际只有当前激活的文件有效，这样显然是不好的。因此我们对后端代码进行了简单修改使之支持多个文件能同时保持有效连接。

3.1.4.6 全新设计的交互界面

因为没有原版 Popush 设计的束缚，因为有了以上精彩的新功能，我们对交互界面进行了全新的设计，更简洁大方高效。

3.2 模块划分

有了 MVC 三层划分，我们又按功能进行了模块划分。以下的模块划分说明与实际代码目录结构有细微差异（比如代码目录结构中将多标签、文件共享、消息中心等归入 WorkspaceModule，而在这部分的说明将它们分在不同的模块中，也就是说目录结构的

模块与此处说明的模块不是一一对应的），做这些调整是为了更好地说明各个数据结构、接口和功能。

3.2.1 通信模块

3.2.1.1 设计概述

在 Popush2 中，将通信模块进行了封装，使得任何需要进行 socket 通信的地方只需要注入通信模块就能顺畅地使用。

通信模块中的文件只有一个，即 socket/SocketModule.js。

3.2.1.2 数据结构

通信模块的数据结构很简单，创建 SocketModel（注入名简化为 socket），在其中创建 Popush2 中唯一的 socket 对象，仅此而已。

3.2.1.3 接口规范

对外提供五个重要的接口方法，分别是：

接口定义	说明
onScope(scope, events)	将 socket 事件与 Angular.js 中 Controller 的作用域 scope 进行批量绑定。其中 events 支持多个事件，events 是个对象，其中有多组 key-value 对应，每组 key 作为绑定的事件名，value 作为绑定的事件处理函数。当对应 Controller 被销毁时，会自动解绑相关事件处理函数。
on(eventName, callback)	将 scoket 事件 eventName 与 callback 绑定，若该事件已绑定有处理函数，将不进行绑定。
forceOn(eventName, callback)	将 scoket 事件 eventName 与 callback 绑定，若该事件已绑定有处理函数，将覆盖已有绑定。
emit(eventName, data, callback)	使用 socket 发出 eventName 事件，数据为 data，回调函数为 callback。
removeAllListeners(e)	将调用 socket 的该方法。

3.2.1.4 界面体现

通信模块没有直接的界面体现，但其他各个模块的界面体现都离不开通信模块的支持。

3.2.2 用户模块

3.2.2.1 设计概述

用户模块包括用户信息管理相关、设置中心、消息中心等。涉及的文件包括：

文件	说明
partials/avatarMgt.html	头像管理相关页面呈现。
partials/languageMgt.html	语言切换相关页面呈现。
partials/message.html	消息中心相关页面呈现。
partials/passwordMgt.html	修改密码相关页面呈现。
partials/signIn.html	登录相关页面呈现。
partials/signUp.html	注册相关页面呈现。

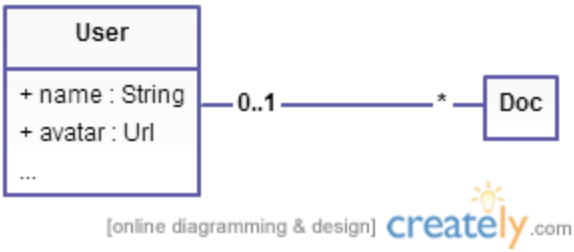
partials/usersettings.html	设置中心相关页面呈现。
user/AvatarController.js	头像管理页面与数据的控制。
user/LanguageController.js	语言切换页面与数据的控制。
user/PasswordController.js	修改密码页面与数据的控制。
user/SettingsController.js	设置中心页面与数据的控制。
user/SignInController.js	登录页面与数据的控制。
user/SignUpController.js	注册页面与数据的控制。
user/SwitchController.js	登录注册功能切换的页面控制。
workspace/MessageController.js	消息中心页面与数据的控制。
user/UserModel.js	定义并实现 UserModel 数据模型。
workspace/MessageModel.js	定义并实现 MessageModel 数据模型。
user/UserModule.js	Angular 模块 userModule 的定义。

3.2.2.2 数据结构

为用户信息管理设计的数据模型是 UserModel（注入名为 userModel），其中涉及的数据包括：

成员定义	说明
+language : String	表示当前语言的字符串。
+userLock : Dict	封装各种锁，类似字典的使用方式。 具体有：connected, signIn, signed, relogin。
+currentUser : User	当前用户对象。
+logoutCallbacks[] : Function	登出的回调函数。

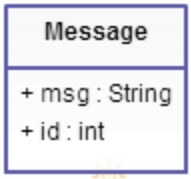
其中 User 和 Doc 的关系可描述为：



为消息中心设计的数据模型是 MessageModel（注入名为 messageModel），其中涉及的数据包括：

成员定义	说明
+msgs[] : Message	Message 类型的消息数组，存放当前所有消息。
-curId : int	消息的 ID，在增加新消息时加一并存入 Message 中，保证每个 Message 对象有唯一的 ID，类似数据库中数据项的 ID。

Message 类定义为：



3.2.2.3 接口规范

UserModel 对外定义了 6 个接口，如下：

接口定义	说明
getLanguage() : String	得到当前的语言。
setLanguage(newLang)	设置语言为newLang，若期望设置的语言不被支持，则重新载入上次的设置。
lock : Dict	userLock。
user : User	currentUser。
logout()	登出的准备、执行与后续。
logoutCallbacks[] : Function	logoutCallbacks。

MessageModel 对外定义了 4 个接口，如下：

接口定义	说明
msgs[] : Message	msgs。
append(msg, tout)	增加一个消息，内容为 msg，tout 毫秒后自动关闭（若未指定，默认为 8000）。
remove(msgid)	根据消息的 id 移除消息。
clear()	清空消息中心。

3.2.2.4 界面体现

事件	界面响应
注册	若注册成功，将自动登录并给出注册成功和登录成功的消息提示。若失败，以红色提示条的方式给出错误提示。
登录	若登录成功，将切换至工作空间并给出登录成功的消息提示。若失败，以红色提示条的方式给出错误提示。
登录后点击用户头像	工作空间切换至用户设置中心。
切换界面语言	登录前可在显眼位置进行语言设置，登录后可在设置中心中进行设置，都是即时生效的。
上传新头像	在设置中心中切换至更换头像功能。点击当前大头像可上传新头像，修改成功与失败均有相关提示。
修改密码	在设置中心中切换至修改密码功能，填写表单后无论成功失败均有相关提示。
登出	点击左上角登出链接即可，将自动清空当前用户数据并登出。

3.2.3 多标签模块*

3.2.3.1 设计概述

我们在工作区设计了对多标签的支持，支持三种类型的标签：用户设置页、文件目录页和编程页。为此，我们需要设计负责标签管理的数据模型，并将它注入到需要的模块去。多标签模块涉及的文件有：

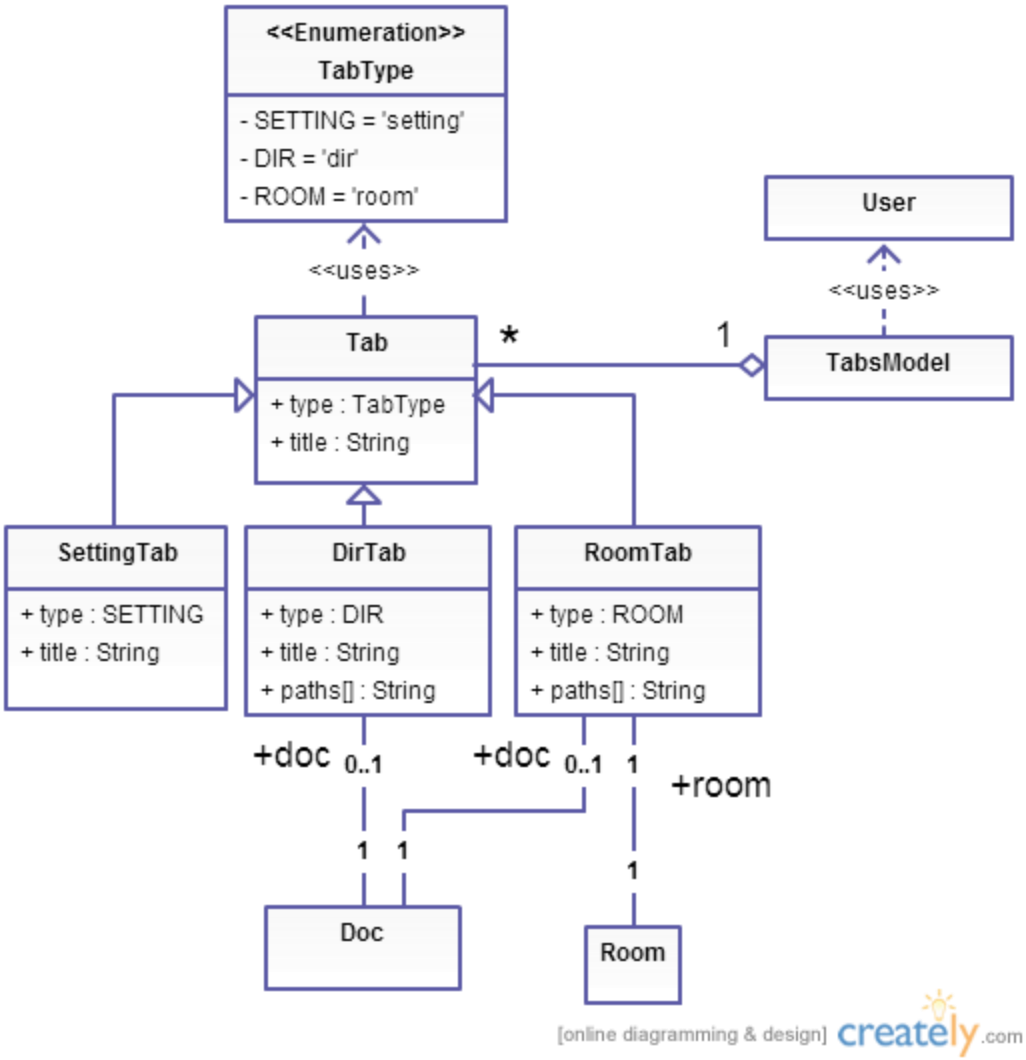
文件	说明
partials/tabs.html	Tab 数据的页面呈现。
workspace/TabsController.js	负责页面与数据的控制。
workspace/TabsModel.js	定义并实现 TabsModel 数据模型。

3.2.3.2 *数据结构*

多标签模块设计的数据模型是 TabsModel（注入名为 tabsModel），其中涉及的数据包括：

成员定义	说明
+tabs[] : Tab	存放所有的 Tab 对象，在创建、关闭 Tab 等操作中需要用到该数组。Tab 对象有三种，都由 type、title、其他个性化数据这三部分组成。具体见表格后的类图。
+current : Tab	当前界面激活 Tab 对象的引用。
+currentMembers[] : User	当前位置的共享用户列表。在 setting 中只有当前用户，在 dir 和 room 中为当前目录或文件的所有共享用户。**该对象不可以更改引用**，只能通过 Array 的成员方法进行更新。
+destDoc : Doc	引用一个 Doc 对象，在创建 Room 对象时传递。
-roomSetCallback : Function	创建 Room 对象完成后的回调函数，在创建 Room 对象前指定，创建完成后自动回调，因此是私有成员。

本模块的数据类可描述为（为更好地说明上述数据结构，采用了 UML 类图但并未完全按照类图的规范）：



3.2.3.3 *接口规范*

对外定义了 17 个接口，其中有 1 个接口已被替代。如下：

接口定义	说明
tabs : Array	对外公开的 tabs 对象。
current : Tab	对外公开的 current 对象。 **该接口已被 getCurrent 替代。**
members : Array	对外公开的 currentMembers 对象。
updateMembers()	通过 current 对象读取当前应呈现的共享成员列表。
showSettings()	创建用户设置标签并激活，若已存在用户设置标签则不创建直接激活。
addFolder(doc)	通过给定的 Doc 对象 doc 创建 dir 类型的标签并激活。若待创建的标签已存在，将直接激活不创建。
setCurrent(index)	用于更新 current 并做一些切换期间的其他处理，设置当前标签 tabs[index]。
changeDoc(newDoc)	在 dir 和 room 类型的标签中使用，进行路径沿途其他目录的切换，将切换至 newDoc 对应的目录。考虑这是用在标签的路径导航栏的，因此切换后一定会变成 dir 类型的标签。
getPath() : String	用于获取当前标签的路径信息。
getCurrent() : Tab	获取 current 对象。
clear()	清空所有标签。在用户登出后进行清理活动。
enterRoom(doc, tabToClose)	创建 room 类型的标签，打开 doc 对应的代码文件。如果指定 tabToClose，还会将对应的 Tab 对象移除。
getDestDoc() : Doc	destDoc 的 get 方法。
setDestDoc(doc)	destDoc 的 set 方法。
changePath(tab, index)	设置 Tab 对象 tab 的当前路径为 tab.paths[index]。
runRoomSetCallback(room)	回调 roomSetCallback(room)。
changeTabPath(oldPath, newPath)	将 oldPath 对应标签的 Doc 对象更新为 newPath 对应的 Doc 对象。

3.2.3.4 *界面体现*

事件	界面响应
打开新的标签	在需要打开新的标签时，会在工作区创建新的标签并传递相关数据。
关闭标签	当只剩一个标签时，不可关闭。当有多个标签时可根据需要关闭当前激活标签或未激活标签，关闭标签将释放相关数据。
标签之间进行切换	切换标签将更新界面显示。

3.2.4 文件目录模块

3.2.4.1 设计概述

为方便地进行文件管理，我们设计了文件树和在标签页的文件目录管理。在文件树中，点击文件夹图标，可以打开或关闭文件夹；点击文件夹名或文件名，可以激活该文件夹或文件对应的标签。当多标签中打开了某文件或文件夹，处于激活状态会在文件树中有蓝底标识，处于未激活状态会在文件树中有绿底标识。

为此，我们需要设计管理文件目录结构的数据模型，并分别为文件树和文件目录的标签页创建 Controller。文件目录模块涉及的文件包括：

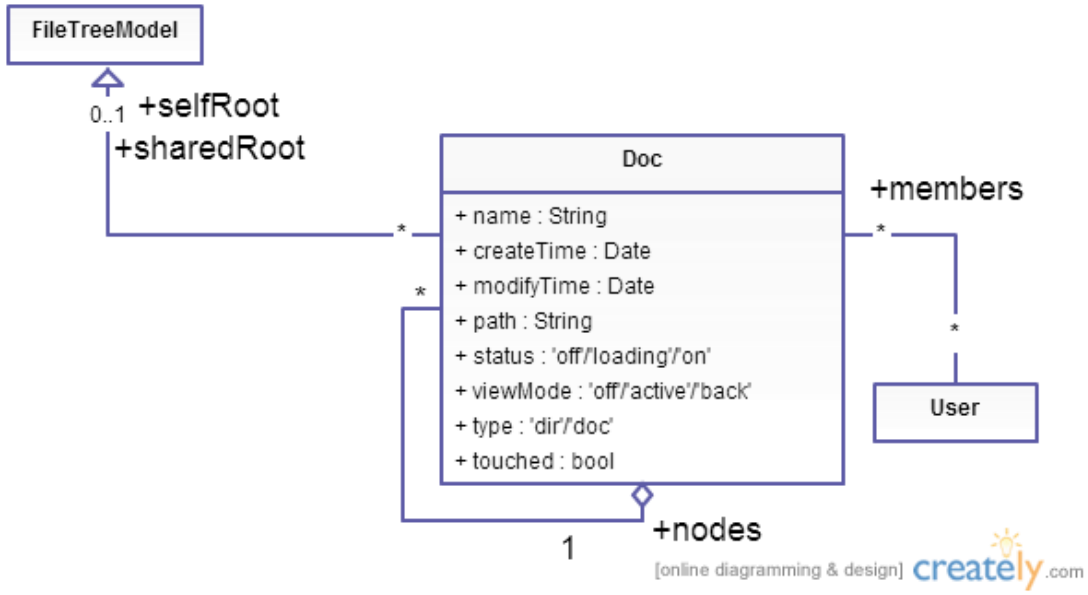
文件	说明
partials/catalogue.html	文件树的页面呈现。
partials/filelist.html	标签页中的文件目录管理页面呈现。
file/FileListController.js	标签页中文件目录管理的控制。
file/FileTreeController.js	文件树的控制。
file/FileTreeModel.js	定义并实现 FileTreeModel 数据模型。
file/FileModule.js	Angular 模块 fileModule 的定义。

3.2.4.2 数据结构

文件目录模块设计的数据模型是 FileTreeModel（注入名为 fileTreeModel），其中涉及的数据包括：

成员定义	说明
+selfRoot[] : Doc	存放属于当前用户的 Doc 对象，在进行个人目录结构更新时需要用到该数组。Doc 类可参考本表后附的类图。
+sharedRoot[] : Doc	存放属于其他用户且与当前用户共享的 Doc 对象，在进行共享目录结构更新时需要用到该数组。
+rootStatus : Dict	作为字典使用，表示文件树中个人目录和共享目录的状态。
+tabsFn : Dict	作为字典使用，存放 TabsModel 的一些回调函数。

Doc 类可描述为（为更好地说明上述数据结构，采用了 UML 类图但并未完全按照类图的规范）：



3.2.4.3 *接口规范*

对外定义了 13 个接口。如下：

接口定义	说明
delete(root)	从文件树中删去 Doc 对象 root 的所有子节点。
select(path) : Doc	根据 path 得到文件树中对应 Doc 对象。
update(docPack)	直接将 socket 的 doc 事件得到的 data 对象作为 docPack 更新文件树。
updateByObj(obj)	由 Doc 对象发送相应 socket 消息更新文件树。
updateByPath(path)	由 path 发送相应 socket 消息更新文件树。
updateRoot()	发送 socket 消息更新个人根目录与共享根目录。
changeViewDirByObj(obj, nmode)	更新 Doc 对象 obj 的 viewMode 为 nmode。
changeViewDirByPath(path, nmode)	通过 path 更新对应 Doc 对象的 viewMode 为 nmode。
self : Array	selfRoot。
shared : Array	sharedRoot。
rootStatus : Dict	rootStatus。
closeChildrent(obj)	设置 Doc 对象 obj 的所有子节点（包括所有层次）的 status 为 off。
tabsFn : Dict	tabsFn。

3.2.4.4 *界面体现*

事件	界面响应
点击文件树文件夹图标	若当前文件夹已展开，将其收起。若当前文件夹未展开，将读取其中内容并展开。
点击文件树文件夹名称	在工作区创建该文件夹的标签(若已存在则直接激活不创建）。
点击文件树文件名称	在工作区创建该文件的标签（编程模式，若已存在则直接激活不创建）。
标签页文件目录点击下一级文件夹	当前标签页直接切换至下一级文件夹。
标签页文件目录点击下一级文件	当前标签页直接切换至下一级文件（编程模式）。
标签页文件目录点击上一级文件夹	当前标签页直接切换至上一级文件夹。
标签页文件目录新建文件/新建文件夹	在文件列表中出现输入框和确定、取消按钮。输入后即可创建，成功失败均有消息提示。
标签页文件目录重命名	进入重命名模式，点击文件名后的铅笔图标即可重命名该文件，成功失败均有消息提示。
标签页文件目录删除	进入删除模式，点击文件名后的垃圾桶图标即可删除该文件，会有进一步确认，成功失败均有消息提示。

标签页文件目录共享	进入共享模式，点击文件或文件夹状态后的共享图标即可设置该文件的共享者，成功失败均有消息提示。

3.2.5 编程模块

3.2.5.1 设计概述

每一个文件被设计成一个房间，不同房间用字段 roomid 区分，在每个房间中独立管理代码编辑、协同编辑、聊天、系统消息、运行、调试等操作。

由于采取了多标签的新设计，因此应保持能同时处于多个房间中，为此对后端代码进行不多的改写（主要是路由部分的一些逻辑），以支持这样的特性。

本模块涉及的文件如下：

文件	说明
partials/room.html	标签页编程模式的页面呈现。
room/RoomController.js	编程模式的页面和数据控制。
room/ChatController.js	编程模式中聊天的页面和数据控制。
room/ConsoleController.js	编程模式中控制台的页面和数据控制。
room/RunController.js	编程模式中运行调试的页面和数据控制。
room/VoiceController.js	编程模式中语音聊天的页面和数据控制。
room/RoomModel.js	编程模式的数据模型。
room/RoomModule.js	编程模块 roomModule 的定义。

3.2.5.2 数据结构

表格 1 数据结构：Room

成员定义	说明
+ doc : Doc	包含当前房间文档信息的对象
+ expressionList[] : String	监视列表
+ data : Object	服务器端传回的 data 结构体
+ id : int	当前房间的 id
+ type : String	当前标签页的类型
+ editor : CodeMirror	当前房间的 CodeMirror 对象
+ state : int = 0	编辑框状态（0 表示 editing, 1 表示 running, 2 表示 debugging）
+ saving : bool = false	代码保存的状态
+ lock : bool = false	当前房间锁的状态
+ locks : Dict	当前房间的锁，类似字典的使用
+ savetimestamp : int = 1	代码保存的已用时间
+ savetimeout : int = 500	代码保存的超时时间
+ timer : timer = null	代码保存的计时器
+ runable : bool = false	当前运行状态
+ debugable : bool = false	当前调试状态
+ q[] : req	文档修改的信息
+ bq[] : req	断点修改的信息
+ bps : String	记录断点位置的 01 字串
+ runningline : int = -1	当前运行行数

+ waiting : bool = false	当前房间的等待信息，如果此时锁的状态为 true，则房间处于等待状态。
+ oldText : String	修改前的文本内容
+ oldBps : String	修改前的断点信息
+ cursors[] : Cursor	代码编辑框中所有用户的光标信息
+consoleOpen : bool = false	控制台的显示状态
+ consoleOutput[] : String	控制台的输出数据
+ consoleState : String	控制台的状态
+ chatOpen : bool = false	聊天窗口的显示状态
+ chat[] : Msg	聊天窗口消息的发送者，类型，时间和内容
+ voiceOn : bool = false	语音聊天模式的状态
+ buffertext : String	放置于缓冲区中的代码
+ bufferfrom : int = -1	修改字符串的起始位置
+ bufferto : int = -1	修改字符串的终止位置
+ buffertimeout : int = 1000	代码缓冲区的超时时间

表格 2 数据结构：locks：Dict

字段	说明
+ run : bool	当前运行状态
+ debug : bool	当前调试状态
+ operation : bool	当前操作状态
+ chat : bool	当前文本聊天状态
+ voice : bool	当前语音聊天状态

表格 3 数据结构: Msg

字段	含义
+ name : String	消息发送者名称
+ type : String	消息类型（system, self, other）
+ content : String	消息内容
+ time : String	消息发送时间

表格 4 数据结构：roomGlobal（静态数据）

字段	含义
+ languagemap[] : String	存储所有 Codemirror 代码编辑框支持语法高亮的语言
+ modemap[] : String	存储不同编程语言对应的 CodeMirror 的 mode 以及 MIME type
+ runableext[] : String	支持“运行”操作的语言
+ debugableext[] : String	支持“调试”操作的语言

表格 5 数据结构：Cursor

字段	含义
+ element[] : String	光标的 div 层信息
+ pos : int	光标的位置

表格 6 数据结构：req

字段	含义
+ roomid : int	当前房间的 id
+ version : int	当前文档的版本号
+ from : int	修改文本的起始位置
+ to : int	修改文本的终止位置
+ text : String	最新版本的内容

3.2.5.3 接口规范

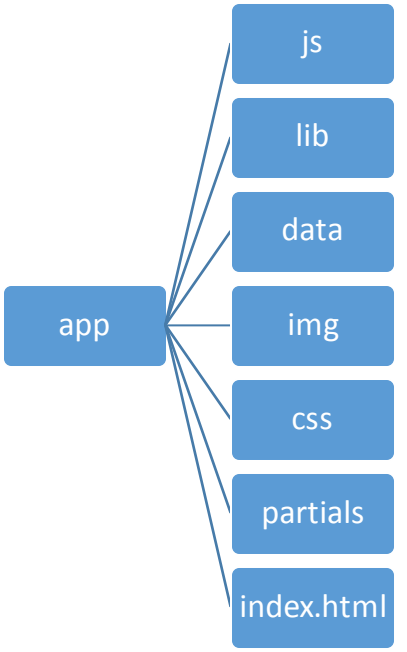
对外定义了 10 个接口，具体信息如下：

接口名称	说明
getRoomById(roomid) : Room	根据该房间 id 获得房间信息
leaveRoom(room)	离开该房间
registerEditorEvent(room)	监听该房间代码编辑框的 change 事件
saveevent(room, cm)	重置该房间的保存状态
initbreakpoints(room, bpsstr)	维护该房间断点字串，初始化新断点
removebreakpointat(room, cm, n)	去除该房间代码编辑框指定行数的断点
addbreakpointat(room, cm, n)	为该房间代码编辑框指定行数增加断点
runtoline(room, n)	运行该房间代码到第 n 行
toggleConsole(room)	弹出该房间的控制台
newcursor(name) : Element	根据用户名创建一个显示光标的 div 层

3.2.5.4 界面体现

事件	界面响应
用户进入房间	当前房间的聊天窗口给出该用户进入的深红色的信息提示。
用户离开房间	当前房间的聊天窗口给出该用户离开的深红色的信息提示。
用户运行程序	当前房间的聊天窗口给出该用户运行程序的深红色的信息提示。
用户调试程序	当前房间的聊天窗口给出该用户调试程序的深红色的信息提示。
用户增加断点	当前房间代码编辑框的相应位置出现断点。
用户运行到某一行	当前房间代码编辑框的运行当前行有箭头标示。
用户增加监视变量	当前房间监视列表出现新添加的监视变量名和变量值。
用户编写代码	当前房间的协同者看到该用户所做的代码修改，该用户光标为红色。
用户发送文本聊天信息	当前房间的聊天窗口显示用户文本信息，发送的信息为绿色，接收的信息为蓝色。
用户发送语音聊天信息	若浏览器支持，当前房间的语音按钮被按下，语音聊天开启；否则提示支持该功能的浏览器。

3.3 目录结构



各个功能模块，包括对应的 Controller 和 Model

引用的第三方库文件

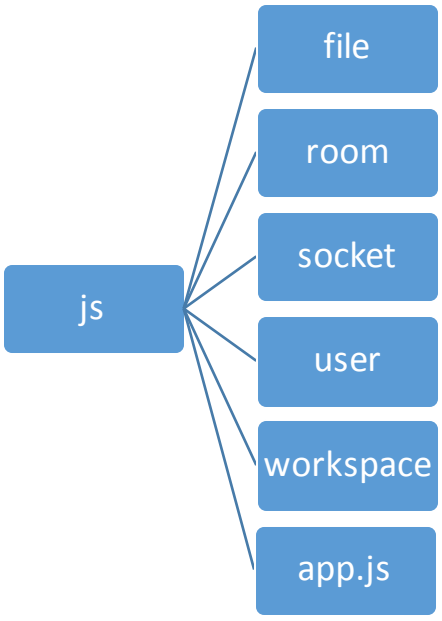
翻译的语言

图片

样式文件

各个 html 页面

主页面



文件功能模块，包括文件树和文件列表等

房间功能模块，包括编辑框、控制台和聊天等

socket 通信封装模块

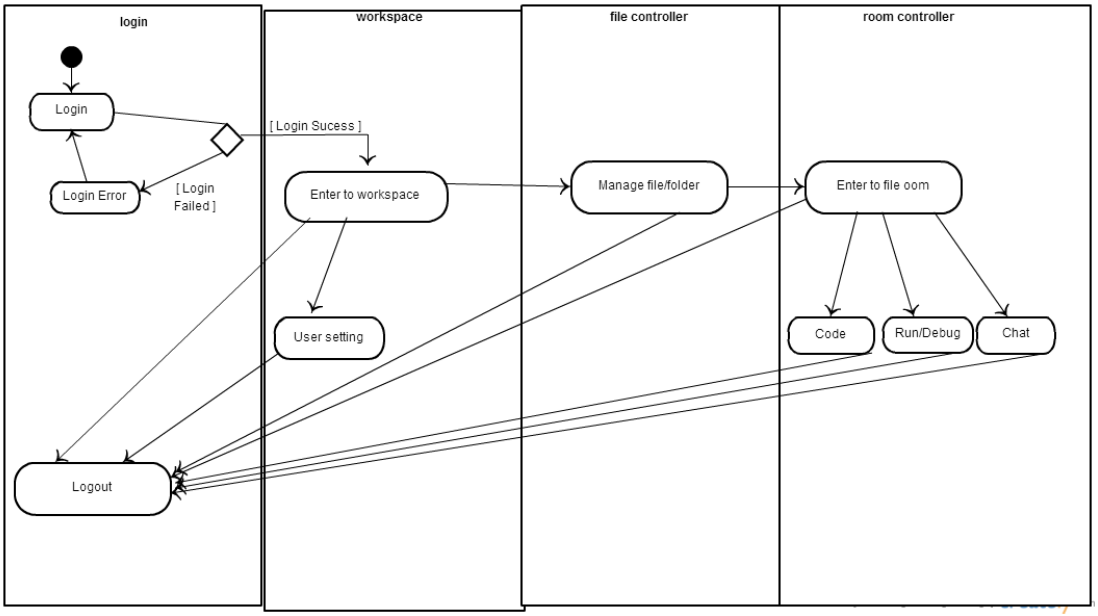
用户管理模块，包括登录、注册、更改头像等

工作区模块，包括多标签、消息机制、和共享等

主模块，前端的控制入口，页面路由以及模块注入。

3.4 系统描述

3.4.1 主要功能流程

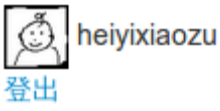


3.4.2 主要界面流程

3.4.2.1 首页（登录、注册界面）

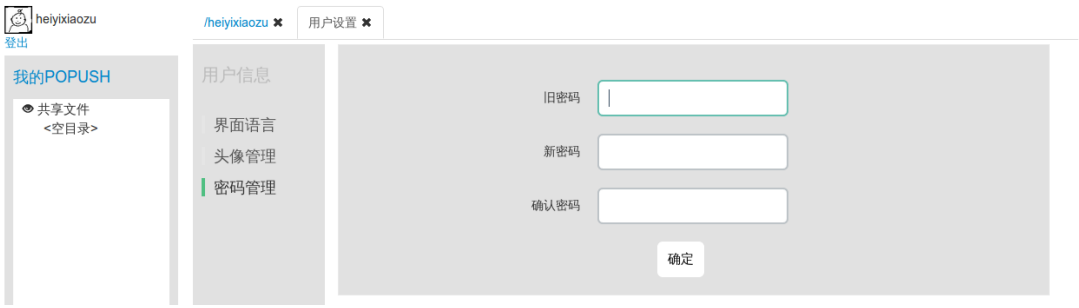
登录成功或注册成功都将自动跳转进入工作区。

3.4.2.2 工作区：用户信息



3.4.2.3 工作区：用户设置中心

点击个人头像进入用户设置中心。



3.4.2.4 工作区：左侧文件树

3.4.2.5 工作区：文件目录管理

3.4.2.6 文件目录管理：新建文件/文件夹

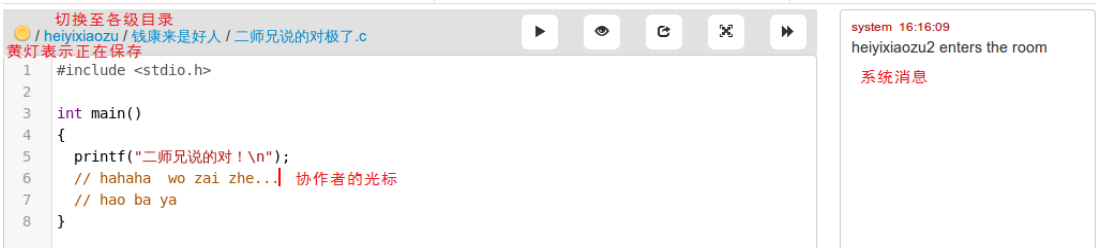
完成文件/文件夹创建后会立刻更新当前目录。

3.4.2.7 文件目录管理：重命名文件

3.4.2.8 文件目录管理：删除文件

3.4.2.9 文件目录管理：共享文件

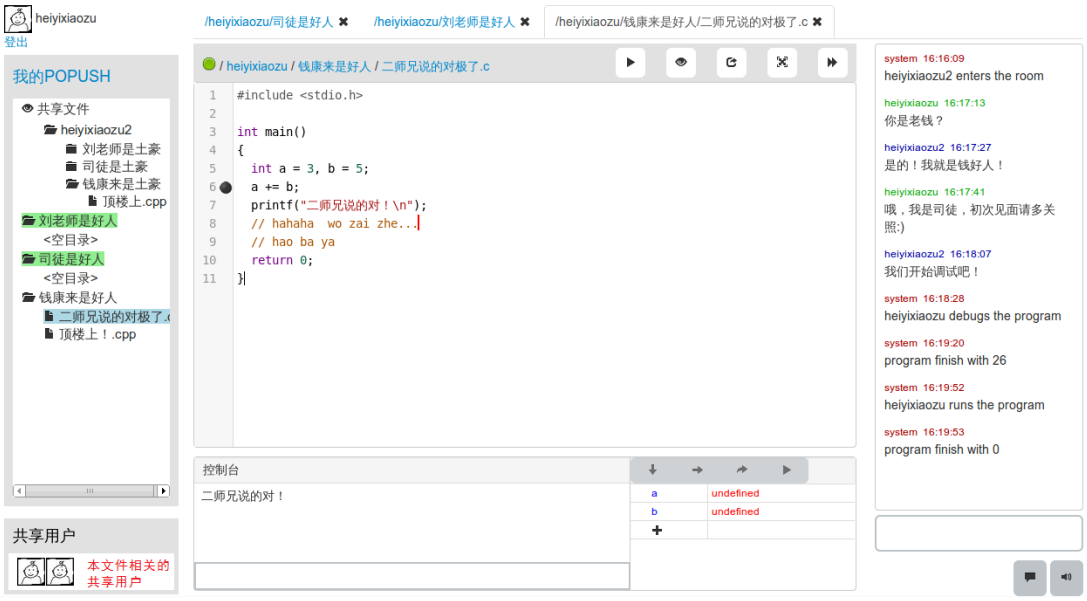
3.4.2.10 编程模式：进入房间与编辑代码



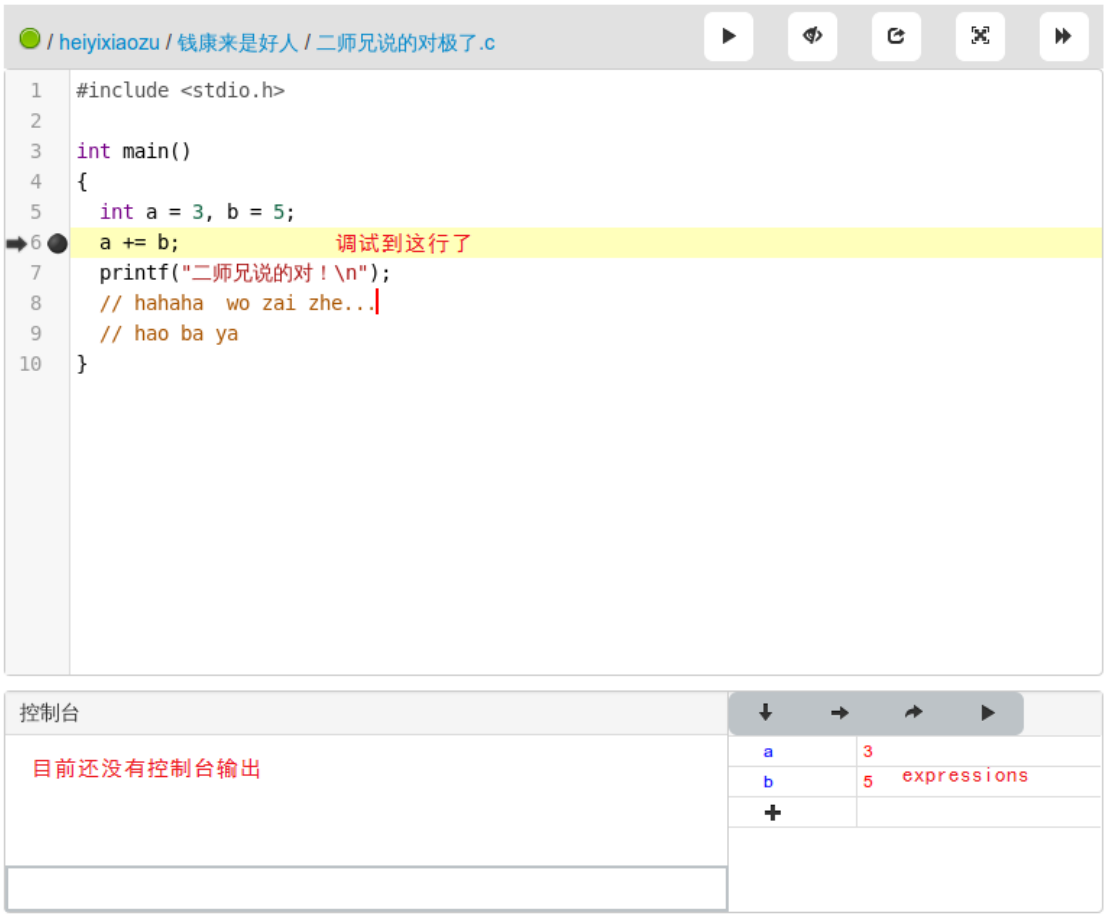
3.4.2.11 编程模式：聊天

3.4.2.12 编程模式：调试、运行

运行的结果如下。



调试的话则是这样（其他区域与上图基本一致）：



4 软件交付

4.1 功能验收

主要有 22 项功能需进行验收，详见下表：

功能	预期结果	操作步骤
注册	若注册成功，将自动登录并有相关提示；若注册失败，在输入框上方，对非法输入给出红色的错误提示。	1.运行 popush 进入主界面，点击“注册”按钮，切换到注册页面。 2.填写用户名，密码和密码确认。 3.点击“注册”按钮。
登录	若登录成功，跳转进入主界面并有相关提示。	1.输入用户名和密码。 2.点击“登录”按钮。
登出	登出成功，回到登录界面。	点击页面左上角的“登出”按钮
语言切换	选择某语言，页面所有说明性文字变为该语言，且消息提示也变为该语言。	方式一：在首页下拉框选择页面语言。 方式二： 1.进入主页面，点击个人头像，打开“用户设置”标签。 2.点击“语言切换”按钮。 3.选择页面语言。
修改头像	若头像修改成功，用户头像变为新上传的图片；若头像修改失败，消息提示失败原因。	1.进入主页面，点击个人头像，打开“用户设置”标签。 2.点击“更改头像”按钮。 3.点击当前头像，选择本地头像上传。
修改密码	若密码修改成功，消失提示密码修改成功；否者提示失败原因。	1.进入主页面，点击个人头像，打开“用户设置”标签。 2.点击“修改密码”按钮。 3.输入原有密码和新密码。
新增文件（夹）	文件（夹）创建成功在右上角给出悬浮提示；名称中包含非法字符（@/\）时输入框变红，触发创建操作时在输入框上方给出错误提示；其他创建失败原因在右上角给出悬浮提示。	1.进入主页面，打开一个文件列表的标签页。 2.点击“新建文件（夹）”按钮。 3.输入文件（夹）名称，键盘回车或鼠标点击“创建”触发创建操作。
查看文件	点击文件，对应文件被打开，左边文件树中被打开的文件被标记为浅绿色，当前	方式一： 在左侧文件树中点击文件。 方式二：

	正在编辑的文件被标记为浅蓝色。	在标签页中点击文件图标。
查看文件夹	点击文件夹，进入相应目录，左边文件树中被打开的文件夹显示文件夹的内容。	方式一： 在左侧文件树中点击文件夹，进入所选目录。 方式二： 在文件列表中点击文件夹，进入该文件夹下级目录。 方式三： 在文件列表上方的路径中点击目录，回退到对应的上级目录。
重命名文件（夹）	文件（夹）名修改成功，右上角悬浮提示修改成功，修改后的名称在文件列表中自动按字母表顺序排序；否则提示在右上角悬浮提示修改失败的原因。	1.进入主页面，打开一个文件列表的标签页。 2.点击“重命名文件（夹）”按钮。 3.点击对应文件（夹）后面的重命名图标。 4.输入新名字。 5.点击“确认”修改，点击“取消”放弃修改。
删除文件（夹）	文件（夹）删除成功，右上角悬浮提示删除成功，删除后的文件从文件列表中消失；否则提示在右上角悬浮提示删除失败的原因。	1.进入主页面，打开一个文件列表的标签页。 2.点击“删除文件（夹）”按钮。 3.点击对应文件（夹）后面的删除图标。 4.点击“确认”删除文件（夹），点击“取消”放弃删除。
修改文件（夹）的协作者	添加协作者成功，右上角悬浮提示添加协作者成功，且协同列表中显示协同者的用户名；删除协作者成功，右上角悬浮提示删除协作者成功，协同列表中用户被删除。两种操作执行成功后，文件列表的共享状态和页面左下角的共享用户会相应修改。 被删除的协作者将被直接强制踢出房间。	1.进入主页面，打开一个文件列表的标签页。 2.点击“共享文件（夹）”按钮。 3.点击对应文件（夹）后面的共享图标。 4.在共享用户的窗口中输入用户名，点击“添加”按钮，添加协作者。 5.选中已存在的协作者，点击“删除”按钮，删除协作者。
进入房间	用户进入房间，当前房间聊天窗口用深红色字体给出	与查看文件的操作一致。

	该用户进入房间的消息提示。	
离开房间	用户离开房间，当前房间聊天窗口用深红色字体给出该用户进入房间的消息提示。	方式一： 关闭文件编辑的标签页。 方式二： 从文件编辑退回到上级目录。
协同编辑代码	用户可以看到其他协作者的光标（红色），不同用户所做的修改会实时同步到各协作者的页面。	1.多个协同编程人员进入同一文件。 2.多人同时进行代码编写。
代码框最大化	代码框覆盖整个屏幕，按下 Esc 退出最大化	1.进入文件。 2.在代码编辑框上方点击“最大化”按钮。
协同运行	所有协作者都可以看到当前代码运行的结果。	1.多个协同编程人员进入同一文件。 2.点击“运行按钮”。
协同设置断点	所有协作者都可以看到所设置断点的位置。	1.多个协同编程人员进入同一文件。 2.在代码编辑框出现行号的部分，在对应行处添加断点。
协同修改变量监视列表	所有协作者都可以看到变量监视列表的变化。	1.多个协同编程人员进入同一文件。 2.在控制台的变量监视列表处点击添加变量或表达式。
协同调试	所有协作者都可以看到当前断点，变量监视列表，以及程序运行到某一行的状态。	1.多个协同编程人员进入同一文件 2.在代码编辑框上方点击“调试”按钮。 3.在控制台中选择“逐语句”，“逐过程”，“继续”，“跳出过程”进行调试。
文本聊天	当前房间的聊天窗口显示用户文本信息，当前用户发送的信息为绿色，协作者发送的信息为蓝色。	1.多个协同编程人员进入同一文件。 2.在代码编辑框上方点击“展开”按钮。 3.在聊天窗口的输入框中输入消息。 4.点击“发送”或者敲击回车发送。
语音聊天	若浏览器支持，不同协作者可以听到彼此的声音；否则	1.多个协同编程人员进入同一文件。

	提示用户选择支持此功能的浏览器。	2.在代码编辑框上方点击“展开”按钮。 3.在聊天窗口中选择语音聊天。

4.2 维护指南*

4.2.1 软件质量特性评估

本部分阐明各个软件质量特性对于 Popush2 的评估方式，维护时应重点关注项目较薄弱的质量特性。

4.2.1.1 正确性

Popush2 使用前端 MVC 框架 Angular.js，作为多页面 SPA 进行开发，当前端代码中出现错误时，将很容易看到有局部页面甚至整个页面显示异常。

4.2.1.2 可靠性

对于平常的使用，Popush2 能稳定可靠地使用，但由于未进行测试，不能涵盖各种极端情形，**后期维护应逐步提高系统的可靠性**。

4.2.1.3 易用性

与原版 Popush 相比，Popush2 进行了一些创新设计，大大提高了系统的易用性。

4.2.1.4 可维护性

通过前端 MVC 框架 Angular.js，与原版 Popush 相比，Popush2 的前端层次划分合理，可维护性得到明显提升。

4.2.1.5 可测试性

Popush2 尚未进行专门的测试，但 Angular.js 内置测试模块，可通过编写相关测试代码进行合理的测试，**之后的维护中要特别关注这一点**。

4.2.1.6 可移植性

系统依赖的软件都是跨平台的，能较方便进行移植。

4.2.2 尚存缺陷与问题*

我们知道 Popush2 不是完美的，即使有充足的时间也很难做到完美无瑕，更何况实际时间并不充裕。

我们正视 Popush2 中存在的问题，罗列如下，供后续维护时优先解决。

4.2.2.1 尚存 BUG

详见**附录 B.2（Popush2 的 BUG）**。

4.2.2.2 设计缺陷：UserModel 完全关注于当前用户

目前 UserModel 设计成了完全用来管理当前用户信息，而实际上 UserModel 用来管理与当前用户有关联的所有用户信息会更好，这样可以省去一些功能实现上的麻烦，还能减少重复信息的传输以优化前后端通信性能，减少一些带宽流量消耗。

4.2.2.3 *维护缺陷：RoomModel 代码混乱*

由于时间关系，没能完全按 Angular 的思想重写 RoomModel，这是 Popush2 的一大遗憾。现在的做法是将原版 Popush 中 room 的相关代码几乎原封不动地移植到 RoomModel 中，最多这是改了一些变量封装和函数名等。这样造成 RoomModel 十分臃肿且代码混乱，不利于维护。

未来维护应考虑将 RoomModel、RoomController 与 room.html 一起重写。

4.2.2.4 *维护缺陷：个别函数不易理解*

Popush2 代码的编写中尽可能地考虑让代码更易读，因此我们的整体思想是让代码直接能读懂，不到迫不得已不用注释，我们相信不用注释也能读懂的代码才是很棒的代码。但由于时间的限制，我们没能把所有代码都写得很漂亮，更糟的是个别被写得不易理解的函数没有一丁点注释，这不利于维护。

未来维护应考虑将晦涩难懂的函数重写，与我们的整体思想一致，我们建议**通过最少的注释能让代码变得易懂**。

4.2.2.5 *文档缺陷：不同对象之间的交互需要更清晰的表述*

本说明书中，对不同对象之间的交互缺少清晰的表达，未来维护应考虑将这部分补全，如为频繁进行跨对象交互的模块增加一些**顺序图和协作图**等。

附录 A　第三方组件

名称	用途	相关文件	来源
Angular	前端 MVC 框架	lib/angular	http://angularjs.org/
Angular-translate	翻译页面语言	lib/angular-translate	https://github.com/PascalPrecht/angular-translate
Angular-UI	与 Angular 配套的 UI 组件	lib/ui-codemirror lib/ui-bootstrap lib/angular-ui	http://angular-ui.github.io/
UI-CodeMirror	Angular-UI 组件	lib/ui-codemirror	https://github.com/angular-ui/ui-codemirror
UI-Bootstrap	支持在 Angular 中使用 Bootstrap	lib/ui-bootstrap 以及 css 中的 bootstrap.min.css	https://github.com/angular-ui/bootstrap
UI-Router	支持局部页面的路由	lib/angular-ui	https://github.com/angular-ui/ui-router
CodeMirror	代码编辑框	lib/codemirror	http://codemirror.net/
Firebase	实时数据库	lib/firebase	https://www.firebase.com
Socket.io	WebSocket 的一种实现	lib/socket-io	http://socket.io

附录 B　BUG 跟踪

附录 B.1　Popush 的 BUG*

标题	具体表现	复现方式	解决状态	解决思路
预料之外的 socket 消息	页面一直连不上后端服务器，后端日志经常出现 Exception，必须重启服务器才能恢复。	emit login -> success emit register -> success emit login -> success emit login -> bug refresh -> success emit login -> bug again	待解决	检查登录、注册的路由逻辑，应该是 login 与 register 有起冲突的操作导致的。
缺少对接收 null 数据的处理	服务器挂掉，后端日志出现 Exception，必须重启服务器才能恢复。	emit 任何一个需要 data 的消息，只要附带 data 为 null 就会出现 BUG。	已解决	增加相关判断。

附录 B.2　Popush2 的 BUG

标题	具体表现	复现方式	解决状态	解决思路
翻译异常	界面翻译不完全	耐心寻找。	待解决	仔细找出所有的待翻译串，将其翻译后的映射写入 data/languages 中相关语言的 json 文件即可。
房间状态异常	房间“状态灯”在特定情况下不能正确地切换	打开一个文件，进行一次键入，会发现状态灯保持绿色，约半秒后转为黄色并很快转回绿色。但如果一次性进行两次及两次以上键入，状态灯会立刻转为黄色。	待解决	这应该是在 CodeMirror 键入事件处理中缺少一种状态灯切换的判断，使得连续两次键入才会发生状态改变。可重点关注那部分的代码。
离开房间异常	无法捕捉离开房间的消息	多个协同编辑者进入同一个房间进行代码编辑，其中一个协同编辑者离开，系统不会给出该用户离开的消息提示，且代表该用户的光标依然存在。	待解决	可能是后端 leave 事件的代码有问题，也可能是前端 leave 事件的处理函数有问题。

代码编辑框光标功能异常	鼠标移到光标上无法显示该光标代表的用户	多个协同编辑者进入同一个房间进行代码编辑，其中一个用户将鼠标移到其他用户的光标上，不会给出该光标的所属用户。	待解决	设置 Codemirror 和 ui-bootstrap 的兼容关系，使得 popover 或 tooltip 生效。
共享状态显示异常	第三级以上文件（夹）的共享状态和根目录的文件夹不同步	选择一个第三级以上的共享文件夹，点击进入，在该目录中查看文件的共享状态，该共享状态只会显示“私有”。	待解决	对共享状态的判断应该与所属第二级文件目录相同，进行这样的更改就行了。
共享用户列表显示异常	第三级以上文件（夹）共享用户列表为空	选择一个第三级以上的共享文件夹进入，会发现左下角共享用户列表为空。	待解决	共享用户列表应该与所属第二级文件目录相同，进行这样的更改就行了。

附录 C　文档编写日志

- 2013/11/09，文档 V1.0 编写完成，主要参与者：文庆福、杨磊、庄晨帆、陈华榕。