# 软件工程综合实验项目验收报告

# 研发报告

## 项目名称： 企业内部沟通APP

项目指导老师： 曹春

项目小组成员：

焦点 141220049

廖祥森 141160033

陈潢 141270004

1. **项目任务**
   1. 项目背景

移动应用（App）方兴未艾，特别是聊天沟通类的app满足了人们即时沟通交流的需求。不过这些App也将生活与工作混在一起，很多职场人士是无法处理得当。使用这类App进行工作交流的话，可能前一秒还在交流工作的事情，后一秒就和朋友闲聊，刷朋友圈去了。

此外许多企业内部信息泄露或被骗的新闻不断传出，也让企业内部沟通app的市场需求越来越大。使用企业内部沟通app可开展企业内部管理，加强企业成员间更好更快地交流和互动，有效提高企业管理效率。

* 1. 项目目标

本产品基于项目背景描述，希望构建专用于企业内部沟通交流的软件。关于软件分别有网页版和移动设备版，使用范围广泛。

为了更好的开发产品，针对企业内部将用户分为管理员、部长、部员三个等级，从而使权限以及功能简单分明。

* 1. 产品功能

|  |  |  |  |
| --- | --- | --- | --- |
|  | **功能名称** | **功能说明** | **备注** |
| WEB | 组织结构管理 | 定义企业树形结构，添加、删除及修改部门信息，修改部门部长 | 管理员 |
| WEB | 任务管理 | 添加符合格式要求的Activiti流程定义文件并部署相应流程，修改流程名，删除已部署流程 | 管理员 |
| APP | 聊天 | 和企业内任意成员进行私聊，部长可开启子部门所组成的群聊 | 所有成员 |
| APP | 查看通知 | 查看已读或者未读通知，未读通知通知栏会推送 | 所有成员 |

* 1. 团队分工

廖祥森：小组组长，担任移动端开发与WEB端开发

焦点：担任WEB端开发

陈潢：担任后端开发与服务器部署

1. **项目进度及里程碑**
2. 研发进度

3月，需求分析、功能设计与UI原型设计。

4月，技术探索，设计API格式。

5月到6月中旬，编写APP与WEB静态界面，测试完成极光聊天DEMO，后端与Activiti的集成，运用Docker与DaoCloud实现持续继承

6月中旬到7月中旬，前后端对接，测试，文档编写。

1. 里程碑

蒲公英快速迭代，发布新版本。



1. **项目技术及辅助工具**
2. 工具

IDE：WebStorm + Sublime Text 3 + Intellij idea

服务器：Tomcat + Nginx + Docker

集成与发布： DaoCloud

测试： Karma + Jasmine + JUnit + 百度云测 + Testin + 阿里云PTS

代码管理：Git + Github

1. 技术

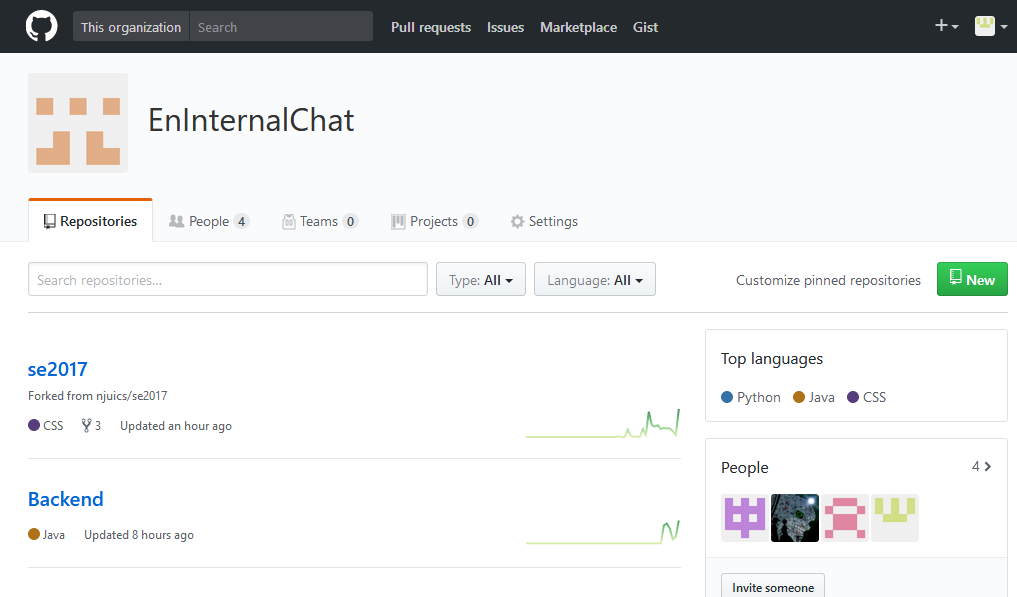
APP借助Ionic与Cordova开发框架，实现跨平台开发，聊天与推送部分借助名极光推送第三方推送服务。

WEB的UI设计借助Bootstrap，整体采用Angular1作为开发框架。

后台基于Java的Spring框架并集成第三方项目，项目使用maven管理依赖。

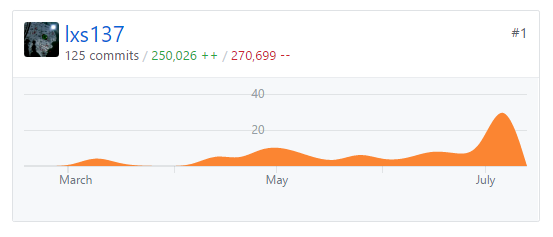
服务端部署在Linux环境下，基于docker-compose实现服务的便携迁移和一键部署。

1. **项目工作量数据**
2. 项目主页[https://github.com/EnInternalChat](https://github.com/EnInternalChat)

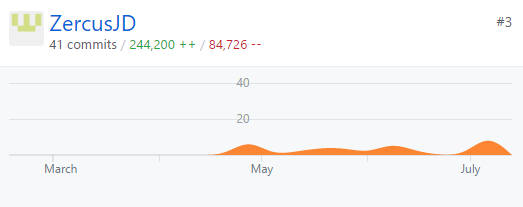


1. 代码commit数据

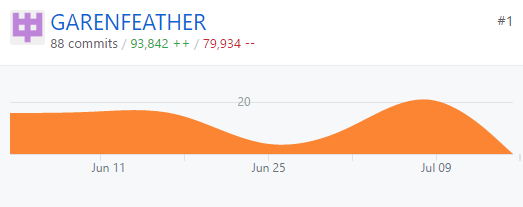
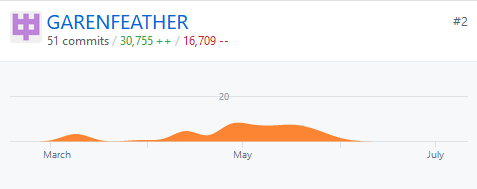
廖祥森：



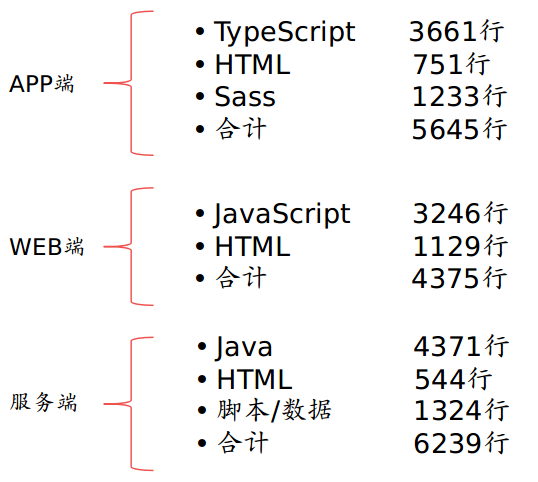
焦点：



陈潢：



1. 代码行数



1. **项目设计变更及应对情况**
2. WEB端页面设计变更

变更原因：第一版WEB页面没有使用WEB开发框架，并且没有公司部门树形结构的展示，为了之后开发的便利性因此弃用。

应对：首先，切换WEB页面模板，添加树形结构支持控件，添加Markdown在线编辑器。其次，对纯静态网页添加angularJS框架代码，为与后端连接交互做准备。

结果：WEB端企业部门树形结构采AngularBootstrap NavTree，实现部门节点的动态添加与删除。SimpleMDE在线markdown编辑器，前后端通知统一采用marked.js进行渲。Bootstrap相应式UI框架，适配多种尺寸的屏幕分辨率。

1. APP端设计变更

变更原因：最早采用ionic1作为开发框架，UI不够美观。

一开始打算使用WebSocket开发聊天功能，开发难度较高。

应对：首先，切换为ionic2开发框架，借助Angular2作为底层的开发框架。其次，调研世面上常用的第三方通信服务的使用，确定使用激光推送JMessage作为通信模块的实现方式。

结果： Ionic2开发的APP界面更加美观，并且采用TypeScript作为开发语言，具有类型检查机制，提高了开发效率。极光推送JMessage使用简单，并且支持Ioinc框架，减少了通信模块的开发难度。

1. 服务端设计变更

变更原因：最初服务端代码完全依赖于部署的Linux环境，mongo、mysql、redis、tomcat等组件都靠手动安装，切换服务器环境需要从头来过，操作繁复耗时，且启动速度极不稳定。

应对：通过咨询经验者并结合当前开发状况以及持续集成部署的要求，选用了docker容器技术，服务本身需要依赖的组件较多，不易集成到一个docker容器中，于是进一步使用docker-compose容器编排技术，并改变本地打包上传部署的方式，直接在服务端完成打包部署和启动。

结果：服务部署和启动效率大幅提升，操作大幅简化，对Linux服务器环境的要求大大下降，因此还成功的在生产开发环境的基础上部署了第二个测试环境，同时服务器数据被破坏后重新初始化变得极为容易，对接效率上升。

1. 后台架构设计变更

变更原因：后台架构设计开始大家都对rest认知模糊，因此初始API设计大范围的违反了restful的设计风格。同时由于对非关系型数据库的认知不足，数据模型设计也冗余且不符合场景需求。

应对：在老师的相关技术指导之下，controller层（此处即API）大范围整改，最终改成了70%以上偏向restful风格的API，且遵守。了更多的restful设计原则。Model层大规模整改，在具有一定关联关系的数据模型之间建立级联引用关系，尽可能去除冗余数据。

结果：整体设计更加合理，获取数据量基本不变的情况下减少了与数据库之间的交互，理论查询效率上升，且数据模型更加符合场景需求。

1. 人员变更

由于余仁杰的提前退出，因此原本由他负责的一部分APP的开发工作完全由廖祥森接手，有他负责的Activiti引擎部分完全由陈潢接手。

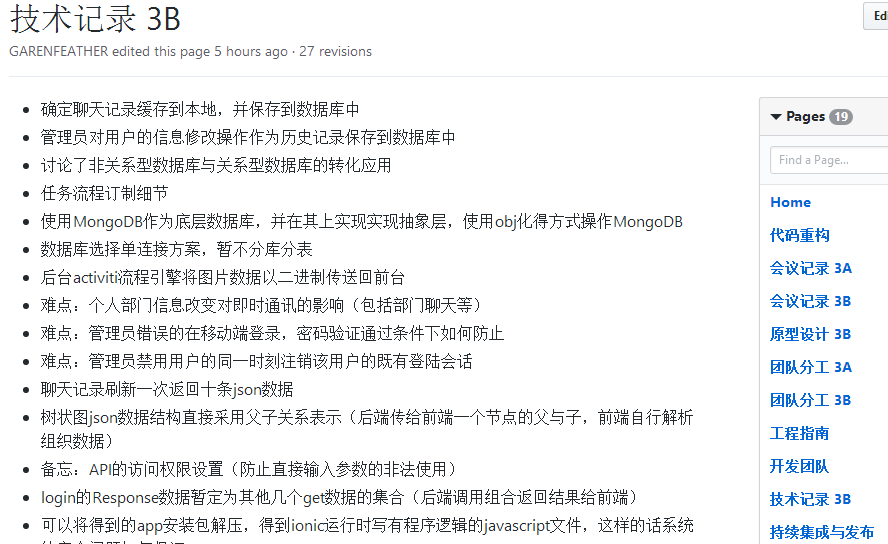
1. **项目相关活动结果**



会议记录



每周团队分工安排



技术记录