**项目总结报告**

日期：2021/1/8

|  |  |  |  |
| --- | --- | --- | --- |
| 组号 | 11 | 项目名称 | HiTime |
| 编程语言 | Java | 开发平台和框架 | Android Studio、IDEA(Springboot框架) |

|  |  |
| --- | --- |
| **软件需求特性** | |
| 1.是否实现了项目立项时的所有需求？列出实现的新增需求和未实现的需求。  实现了立项时的所有需求  ·用户注册和登录(格式、强度校验、密码和验证码登录)  ·修改密码  ·设定邮箱  ·趣味闹钟(可以设置小游戏)  ·语音设定闹钟  ·远离手机（可以设置白名单）  ·添加删除好友  ·为好友设定闹钟  ·智能推荐相同作息时间的好友  ·修改主题  ·分享功能  ·积分功能  ·新手指引  ·统计报表  2.采用哪种架构风格？哪些设计模式？  **前端：**  前端采用client/server架构，前端作为client向后端发送请求，收到回应后通过ui展示给用户。  前端的设计模式如下：  ①template method：由于前后端的交互主要是由websocket和http请求进行实现的，只有传输的参数形式不同，故将交互抽象成了一个template类，并定义传输的接口。在数据传输时，在此template的基础上，构建一个concrete class即可，从而极大的提高了代码的复用性。  ②bridge method：后端将实现和接口分开，所有的代码实现均面向接口进行编程，降低了前后端耦合性，实现了应用内的高内聚，低耦合，同时也提高了代码的可读性。  **后端：**  后端采用分层架构风格，将后端分为Controller，Service，Dao等层，分别负责向前端用户传递数据、执行业务逻辑处理用户请求、执行数据持久性操作等工作，有利于后端整个系统的维护和复用，各层逻辑分工明确。  后端的设计模式如下：  ①职责链模式，利用Springboot提供的Interceptor功能对HTTP请求进行拦截，阻拦不安全的API调用，达到认证与鉴权的目的。  ②装饰器模式，为后端处理产生的各种结果以一个统一的Msg对象装饰并返回给前端APP，实现后端返回信息的用户友好。  ③外观(Façade)模式，为一组接口提供更高层的统一接口，易于操作使用。  3.技术方案有哪些亮点？  ①核心算法：在推荐好友功能处，使用k均值聚类算法，根据趣味闹钟、远离手机的使用习惯，对用户进行智能分类，为用户推荐与之使用习惯类似的用户，实现智能的好友推荐，提高用户粘性。  ②用户友好:首次下载时，有核心功能的简要介绍，在用户可能误解的地方加入了提示，帮助用户自定义自己的远离手机计划。  ③激励机制：灵感来源于forest软件，通过向日葵的生长情况来督促用户完成自己制定的任务。  ④由于HTTP的无状态性，后端采用了JWT对HTTP请求进行认证，达到了拦截未授权的API调用的效果，保证了用户隐私安全。  ⑤采用了Springboot提供的Validator，以annotation的形式将参数校验与实际业务逻辑解耦，使得代码更加简明清晰。  ⑥通过Jenkins服务器实现了自动化测试+部署的功能，减轻运维人员的工作量。  4.是否做了单元测试？是否做了系统功能测试？是否做了性能测试？是否做了兼容性等其他非功能测试？  后端采用Junit完成了单元测试，核心业务逻辑测试基本覆盖了所有代码，代码行覆盖率达到了93%。  对一些后端的API做了性能测试，考虑到实际的硬件水平，性能效果在可接受范围内。  按照产品需求文档对后端进行了完整的系统功能测试，每一个API接口均正常工作。 | |
|  | |
| **项目组成员对项目的贡献度（%）** | |
| 崔绍杰：25%  刘遇时：25%  杨弈骋：25%  周洹羽：25% | |
| **软件度量** | |
| 软件代码行数（不包括注解行、空行和复用代码）： | 147217 |
| 复用他人代码行数： | 600 |
| 类的个数： | 209 |

|  |
| --- |
| **经验、教训和建议** |
| 经验：  此次项目中，拥有细化的commit和code review流程，规范的CI/CD流程能够大大加快项目的开发速度。  一开始的进行适当合理的架构设计、数据库设计、接口设计，能极大的提高代码的复用性，免于返工。  教训：  在项目开始就应该统一命名规范与ui风格，防止返工。 |