清华帮帮忙项目报告

项目简介

项目名称为 ***TsingHelper*** ，中文名为 ***清华帮帮忙*** ，旨在解决校内学生带餐、二手交易、学习互助等需求问题。

项目功能

截至目前，本项目所有已实现的功能包括：

个人信息有关

- 支持修改用户头像
- 支持显示个人主页，并可自行设置主页背景图片
- 支持修改并显示其他基本信息，包括但不限于：真实姓名、微信、邮件、宿舍地址、所属院系和年级等

任务（需求）有关

- 发布任务
  共四种任务可供用户选择：社区互助、代餐跑腿、学习解惑、个人问卷。四种任务大体信息相同，同时也有各自特有的一些信息
- 查看其他人发布的任务
  首页会显示20个其他用户发布的任务，用户可进入底部导航栏中的 **任务** 页面查看所有任务
- 任务筛选及排序
  首页和任务页面均可根据关键词对任务进行搜索和筛选，在首页还可查看指定类型的任务，任务页面可对所有任务按时间及报酬进行排序，其中时间按照发布时间从早到晚显示，报酬按照从高到低进行显示
- 接取并提交任务
  用户点击任何一个任务进入详情页面后可接取该任务，接取之后可进行任务的提交，提交之后需要等待发布者审核是否已正确完成任务
- 任务的统一管理
  在个人页面中可以统一管理与我有关的所有任务，处于不同阶段（如待提交、待审核等）的任务会分开显示，同时可在 **我发布的任务** 页面对我发布的任务进行管理或者快捷进入发布任务界面。针对每个我发布的任务，可以查看这个任务目前的完成情况，包括正在做的人、待审核的人等，其中对每个已提交该任务的用户，可在这里选择审核通过或审核不通过。

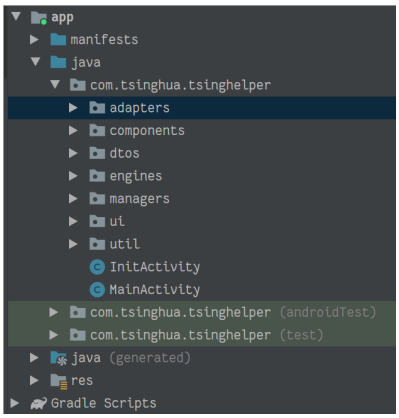
用户间互动有关

- 消息发送与接收
  用户可通过进入其他人的个人主页或者消息页面，发起与他人的聊天。在接收到来自其他用户的消息后，客户端能够将其实时的更新在消息页面的对话框列表中。进入聊天页面后会自动显示与该用户的所有历史聊天记录以及每条消息的发送/接收时间
- 关注与取消关注
  个人主页中会显示这个用户已关注的用户数量和被关注的数量，用户可在他人主页中选择关注或者取消关注此人（如果已经关注了的话）。同时还可通过个人页面里的 **我的关系** 按钮进入关系页面，此页面会显示用户当前已关

注的和被关注的所有用户列表

项目基本架构

项目的代码目录结构如下图：



下面分别对每个包进行相关说明：

adapters

包括各个界面需要的一些常见的数据适配器，如 `RecyclerView` 的适配器等。

components

包括几个自定义的UI组件。

dtos

DateTransferObject: 数据传输对象，该对象定义了与后端交接数据的规范，如字段名称等。
此包中对几个常用的数据传输对象进行了封装，主要包括用户对象、任务对象以及消息对象等。

engines

用户从本机选择图片进行上传的图片加载引擎包，负责展示本地照片供用户选择。

managers

包括一个 `ImageGridLayoutManager` 类，用户在创建任务页面上传的任务图片就是由该类负责进行排版的。

ui

绝大部分程序界面都定义在这个包中，此包又有以下几个子包：

- home 包括首页部分的几个页面。
- login 登录页面和忘记密码页面。
- messages 消息页面和聊天页面。
- mine 个人页面，以及以下两个子包：
  - profile 个人资料显示页面和资料编辑页面。
  - settings 设置页面。

- search 任务的搜索页面。
- task 创建新任务的页面、任务详情页面以及任务审核页面。

util

整个程序通用的一些工具类。
如提示框显示工具 `ToastUtil` 、定义后端接口地址的网络工具 `HttpUtil` 以及图片缓存工具 `GlideCacheUtil` 等。

成员分工

本项目由邱圆辉、杨松霖、那森共同开发，具体成员分工如下：

- 杨松霖
  - 前端界面的设计、实现与美化
- 那森
  - 后端框架的搭建及部分接口的实现与测试
- 邱圆辉
  - 前端界面的设计、实现与美化
  - 后端部分接口的实现与测试

设计亮点

项目的几个设计亮点如下：

- 程序整体架构清晰，不同包之间分工明确
  上文已对项目的架构进行了较为详细的说明，在此不再赘述。总之项目总体遵循了数据与界面分离的原则，页面在更新时只需调用相应的数据类进行数据更新即可。
- 与后端交互的统一管理
  本项目的后端有将近三十个不同接口，每个接口又都有不一样的数据接收与返回格式。为了统一规范管理前后端的数据交互，我们分别针对这两个问题采取了如下的应对策略：
  - ***HttpUtil*** 类的设计与实现，此类主要负责：
    - 统一定义后端的所有接口地址，这样当后端接口地址有修改时前端只需要在此类中修改对应地址即可。
    - 基于 ***okHttp3*** ，对常用的 ***get post*** 请求进行了一定的封装，减少了代码的冗余量。
  - ***dto*** 包的设计与实现，此包中定义了三个与后端数据交互有关的 ***DTO*** 对象：
    - ***MessageDTO*** ：消息传输对象
    - ***TaskDTO*** ：任务传输对象
    - ***UserDTo*** ：用户传输对象

    封装后，前端可直接根据后端返回的 *json* 数据快速构造一个对应的 ***DTO*** 对象，而不必每次都进行繁琐的解析操作。
- 消息的发送与接收采用了 ***WebSocket*** 协议进行传输
  程序运行后会与服务器建立一个 ***WebSocket*** 长连接，负责消息的发送与接收。这样避免了 *http* 协议无法让服务器主动给客户端发送数据的缺陷，同时也保证了客户端在接收到来自其他用户的消息后能够及时的刷新在消息页面。

项目感想

这个项目是我第一个参与开发的安卓项目，经过几个月的开发后，总结几点收获如下：

- 首先极大程度上让自己熟悉了安卓前端界面开发的流程，包括各种组件的布局、各种 margin和padding。
- 除了前端开发之外，还增进了对java开发以及用nodejs作为后端进行开发的熟悉程度。

当然，还有几个小瑕疵：

- 前端界面除了消息和聊天页面外，没有用任何第三方UI框架，所有界面均由原生安卓组件或自定义的组合原生组件组成。虽然能够更底层地接触安卓开发，但是在一定程度上降低了项目的开发效率。
- 消息页面实现不完善。虽然后端支持文本和图像消息的传输，但由于时间关系，前端目前只支持传输文本消息。同时没有针对未读消息的通知（即类似小红点的显示）。除此之外，我们虽然已经实现了历史消息的缓存类（见 ***util/ChatHistoryCache*** ），但是因未能克服缓存带来的诸多难题（如保证不重复缓存、保证从缓存获取的消息不会再从后端请求等），我们没有将这个缓存机制应用起来，而只是简单的将所有消息都放在内存里，希望下次碰到类似问题时能够采取更好的策略。

其他说明

为了方便老师和助教进行测试，我们将后端部署在了自己的服务器上，同时还创建了多个用户和多个任务以增进项目的用户体验，在此将目前所有已存在的用户列举如下，供老师助教测试用：（注：本项目登录时只验证手机号和密码，因此登录时务必输入正确的手机号）

用户名	密码	手机号
Chinwer	227515	18573843049
Lymntrix	227515	15873818124
Niklaous	227515	13034663500
Elijah	227515	18724653394
留白	227515	13695723454
养乐多	227515	18922336677
山海	227515	13622226983
Queen	227515	16845692769
nasen	114514	17610779859
Candy	227515	13549872365

Created: 2020-06-22 Mon 22:48