研发报告

目录

引言

- 编写目的

> 该文档为详细介绍SchoolInapp的开发过程、在开发过程中遇到的难点以及如何解决。通过阅读该文档，可以使读者对SchoolInApp清楚的了解SchoolInApp的研发流程、人员分工、项目进展、项目关键技术等，了解SchoolInApp的开发过程中遇到的困难以及如何解决这些困难。

- 背景
    - 软件名称

> 软件名称为：ScholInApp

    - 任务提出者

> 南京大学计算机系张天老师作为项目的发起人，项目需求来源于实验课程，由张天老师扮演项目发起人，提出用户需求。

    - 开发者

> 在葛笑飞同学的指导下，由李聪，冯伟赞，满磊，路萍，朱修羽五位大三同学组成的开发团队，共同完成整个项目的开发,其中李聪作为team leader。

    - 用户

> SchoolInApp的使用用户分为两类，包括所有南京大学的本科生以及老师。注册用户时需要使用南京大学邮箱，非南京大学人员没有权限使用SchoolInApp。

- 专业术语
    - SchoolInApp：软件的名称
    - studio:工作室
    - essay:工作室发的文章
    - question: 用户或者工作室提出的问题
    - REST: Representational state transfer
    - API: Application Programming Interface
    - spring: 应用程序框架
- 参考资料
    - 客户端开发平台
    - 服务器端开发平台
    - Android开发规范
    - 后端java开发规范
    - 开发者单位
    - REST
    - spring
    - API

研发过程

- 研发流程
    - 需求分析
    - UI & API 设计
    - 编码 & 测试
    - 基于文本的App（60%）

    - Https/Cookie和通知机制（5%）
    - 基于富文本的App（25%）
    - 语音功能（5%）
    - 界面调整与优化（5%）
    - 上线 & 维护
- 项目任务
    - 项目任务
        - 我们需要完成一个安卓端的以兴趣和问题驱动的知识咨询共享服务平台。
    - 参与方
        - 该App由个人用户和工作室成员参与互动，从而让工作室可以为个人用户提供知识咨询和共享服务。
    - 用户
        - 注册进入该App的用户为南大在校校职工，凭南大个人邮箱进行注册。注册后的用户可以
            - 浏览热门问题/文章，并对问题/文章进行点赞，对文章进行评论。
            - 向某个工作室咨询问题。
            - 通过资历证明材料，提出组建工作室的申请，并在通过后组建工作室，从而对外提供知识咨询和共享服务。
    - 工作室
        - 工作室的成员可以为个人用户做出解答。工作室管理员可以发表文章进行知识共享。
- 项目组及成员分工安排
    - 页面设计：李聪负责设计，并开会与小组其他成员讨论修改
        - API 设计：小组讨论 API 设计样例
        - 冯伟赞：用户 API
        - 李聪：工作室 API
        - 路萍：问题/文章 API
        - 满磊：回答 API
        - 朱修羽：评论 API
        - API 维护：李聪
    - 前端代码编写：页面划分为主要依据，功能为次要依据
        - 李聪：登录注册页面；个人信息页面(包括好多个子页面，收藏、已提问收藏回答问题等)；团队信息页面；同时兼做通知、语音、安全功能；
        - 满磊：提问(定向、不定向)页面；写文章页面；搜索页面；兼做富文本；
        - 路萍：文章展示页面，包括对文章的评论展示页面；问题页面，包括对问题的评论展示页面；回答展示页面，包括对回答的评论展示页面；兼做富文本显示组件；
    - 后端代码编写：冯伟赞、朱修羽
        - 冯伟赞：用户、问题、回答；搜索；富文本；语音
        - 朱修羽：工作室、文章、评论；通知机制
    - 文档整理：全体成员

        - 路萍、朱修羽：需求文档、设计文档、demo录制
        - 满磊：工作日志、研发报告、用户手册
        - 冯伟赞：后端测试计划、后端测试报告、前后端测试报告和计划的整合
        - 李聪：前端测试计划、前端测试报告、项目报告
- 项目进度
    - 已完成
        - 需求分析
        - UI & API 设计
        - 编码 & 测试
        - 基于文本的App（60%）
        - Https/Cookie和通知机制（5%）
        - 基于富文本的App（25%）
        - 语音功能（5%）
        - 界面调整与优化（5%）
    - 正在进行
        - 上线 & 维护
- 项目里程碑
    - 需求文档
    - 设计文档
        - UI 设计稿
        - API文档
    - 代码
        - 客户端 | SchooledInApp
        - 服务器 | server
        - 测试服务器 | pseudo-server
- 关键技术
    - 贯穿：Distributed Version Control
    - 设计：Material Design，REST，HTTP
    - 基于文本的 App ：Android Base，CS/MVC/ORM
    - 开启通知机制的 App ：Long live connection
    - 基于文本的安全 App ：HTTPS/SSL/TLS，IP-Based-Self-Signed-Certificate
    - 基于富文本的 App ：HTTP Multipart，Inverted Index
    - 基于语音的 App ：Android Media，Data Persistence
- 辅助工具
    - 协同开发：git（down to gitlab）

- 界面设计：PhotoShop
- 前端
  - 开发工具
    > Android Studio 3.1
  - 辅助类库
    > Gson; CircleImageView; Glide; ButterKnife; Okhttp3; Retrofit; Otto; BRVAH; JPush; RichEditor; FloatingActionButton; Espresso;
- 后端
  - 开发工具
    > IntelliJ IDEA 2017.1
  - 辅助类库
    > tomcat; hibernate; spirngframework; commone-fileupload; jackson; log4j; jsoup; lucene
  - 部署
    > Docker

- 项目统计数据
  - 需求功能点：16个（详见需求文档）
  - 文档规模数：
    - 需求文档：15页（路萍、朱修羽整理）
    - 设计文档：73页（路萍、朱修羽整理）
  - 业务代码行： $ find . -name "*.java" | xargs cat | grep -v ^$ | wc -l
    - 前端：15k +（粗略估计：lc 6k; lp 4k; ml 5k）
    - 后端：10k +（粗略估计：fwz 6k; zxy 4k）
  - 项目分支合并次数：
    - 前端 36/48 次
    - 后端 20/27 次
- 需求变更&变化
  - 时间
    - 我们组正处于API设计完成阶段
  - 应对策略
    - 校园问答集体需求讨论
    - 确定会议召开后
    - 由于相对于我们本来的设计的需求和API来说有所缩小，因此我们也将API缩小
    - 放到dev分支下，原API在master分支下
- 规范与合作
  - 开发过程中严格遵循如下代码规范：
    - AndrJava oid 开发规范： http://www.androidchina.net/2141.html
    - 后端开发规范： http://www.hawstein.com/posts/google-java-style.html

- 合作
  - 组会：每周周五晚 19:00 在 211 集中开会。
    - 会议内容主要包括：本周任务完成情况、个人本周进度汇报、组内Code Review、分支合并、下周任务布置等
    - 重要会议的纪要在 gitlab 上记录。
  - 集中开发：每周二晚19:00 – 22:00 集中在 208 开发。
  - Git 协同合作方式：为保证代码的可追溯性，我们对 Git 做了如下要求：
    - Git 分支主要分为 master, dev, feature_xxx, fix_xxx。
    - master 分支保存大版本迭代（目前仅有一个大版本），受保护，任何人不可合并、推送，当且仅当所有人同意后在所有人注视下进行分之合并。
    - dev 分支保存小版本迭代（分支合并），受保护，任何人不可合并、推送，当且仅当所有人同意后在所有人注视下进行分之合并。
    - 所有新功能一律以 feature_xxx 或者 dev_xxx 分支命名。
    - 所有 debug 一律以 fix_xxx 分支命名。
    - 为保证代码可追溯性，分支合并一律用 git merge --no-ff ，不可用 git rebase ，不可省略 --no-ff

关键技术

- 前后端解耦
  - 背景
    > 使用传统软件设计方案，由于软件功能繁多，会使前端和后端需要经常沟通，是软件开发效率降低，开发成本上升，如何设计SchoolInApp使得前后端沟通减少成为一个难题。
  - 解决方案
    > 摒弃传统软件基于功能的设计方案，我们使用面向资源的设计方案。SchoolInApp的每个功能是对各个资源的操作，只要具有相应的资源，自然可以实现相应的功能，由面向资源带来的好处是，前后端可以根据资源进行交互，只要定义好SchoolInApp具有的资源以及对资源进行的操作，前后端只需要实现对资源的操作即可，因此可以大大减少前后端的沟通，rest风格的API可以满足这一需求。
- UI设计
  - 背景
    > 由于没有设计基础，SchoolInApp的前期界面设计很不美观，如何设计出美观的界面成为一个难题。
  - 解决方案
    > Android官方推荐的界面设计风格是material，设计并使用满足material风格的的控件并使用既为满足material风格的设计，SchoolInApp的界面外观得到大幅度提升。
- 模块划分
  - 背景
    > 前端由三人合作，后端由两人合作，如何划分模块使得每个人的任务均匀而且模块之间的耦合度最小。
  - 解决方案
    > 前端依照activity划分，由于Android每个页面是一个activity，依据activity来划分是最适合的。后端将资源划分为五部分，各个资源之间的耦合度最小，每个人负责完成其中的部分资源。
- 网络框架
  - 背景
    > Android开发避免不了使用网络，但是我们需要选择一种既适合rest又适合HTTPS的网络框架。

- 解决方案

> 针对网络框架，展开调研，Android常用的网络框架有原生框架，OkHttp，retrofit,其中retrofit既适合rest也适合HTTPS。

- 数据安全
  - 背景

> 网络安全是一个很大的问题，如何使用户数据有最大限度的安全保障是一个迫切需要解决的问题。

  - 解决方案

> 展开调研，对于用户名和密码，使用cookie能够解决，但是对于用户数据，不能使用cookie解决，需要使用散列等，比较麻烦。使用HTTPS既能解决用户名密码安全也能解决用户数据安全问题。

- 代码规范
  - 背景

> 每个人的代码风格都不一样，代码没有统一规范，无法阅读，代码量比较大，自己写的代码自己也会记不住，调试等无法进行。

  - 解决方案

> 收集网上的代码规范，整理出一套自己的代码规范，所有人员代码需要按照这个规范，代码提交的时候需要其他的人员审核，审核不通过则不能提交。

- 生成证书
  - 背景

> 由于使用由专门机构颁发的HTTPS证书需要花钱，我们决定使用自签名证书，但是基于IP的自签名无法直接使用。

  - 解决方案

> 设法给证书添加多个域名，使其能够正常使用，生成证书的脚本见脚本附录。

- 通知
  - 背景

> 谷歌官方有提供原生的推送通知，但是在中国境内，所有谷歌提供的服务都会被屏蔽，选择哪种第三方服务可以完成消息推送功能。

  - 解决方案

> 调研国内最流行的消息推送服务，最后选择极光推送，成功完成消息推送。

- 富文本编辑器
  - 背景

> 普通文本显示太单一，文字不美观，用户体验差，如何能够编辑向markdown一样格式的文本，提升用户体验。

  - 解决方案

> 通过调研，返发现可以使用第三方提供的开源项目，我们在github上找到使用最多的richeditor编辑器，使用richeditor进行编辑文本，极大的提升了用户体验，同时也带来了问题，文本显示也需要富文本显示器，我们在github上也找了一个显示富文本的显示器，解决显示和编辑问题。

- 语音
  - 背景

> 在需求说明里有明确提到需要支持语音，但是Android原生的语音效果很差，不能使用。

  - 解决方案

> 通过调研，找到github上的一些开源项目，但是由于版本问题，不能直接使用，我们不得不自己实现用户使用语音发送文章和问题的功能，实现语音点击播放，自己实现下载语音并播放。

脚本附件

```
#!/bin/bash -e

if [[ -z "$1" ]]; then cat <<EOF
certify - Creates and self-signs X.509 SSL/TLS certificates
          with the "subjectAltName" extension.
Usage: certify IP:114.212.130.21 [IP:114.212.130.21] [DNS:mail.example.com] [...]

EOF
exit; fi

umask 066
KEYSIZE=2048
DAYS=3650
PRIVKEY="privkey.pem"
CSR="csr.pem"
CERT="cert.pem"
CERT_P12="cert.p12"
CERT_JKS="cert.keystore"
P12_PASS="123456"
JKS_PASS="123456"
CN="$1"
altnames="$1"

while shift && (($#)); do altnames="$altnames,$1"; done
echo -e "basicConstraints=critical,CA:true,pathlen:0\nsubjectAltName=$altnames" > extensions.ini

# generate private key
openssl genrsa -out "$PRIVKEY" $KEYSIZE

# generate certificate signing request
openssl req \
        -new \
        -key "$PRIVKEY" \
        -out "$CSR" \
        -subj "/C=CN/ST=Uxample/L=Vxample/O=Wxample Tech Inc/CN=schooledin.pseudoserver"

# generate the self-signed certificate with SAN enabled
openssl x509 \
        -req -signkey "$PRIVKEY" \
        -days $DAYS \
        -in "$CSR" \
        -out "$CERT" \
        -extfile extensions.ini

# show certification info
openssl x509 -in "$CERT" -noout -text

# convert to p12
openssl pkcs12 \
        -export -in "$CERT" \
        -inkey "$PRIVKEY" \
        -out "$CERT_P12" \
        -name "server" \
        -passout "pass:$P12_PASS"

# convert to keystore
/usr/java/default/bin/keytool -importkeystore -v \
        -srckeystore "$CERT_P12" \
        -srcstoretype pkcs12 \
        -srcstorepass "$P12_PASS" \
        -destkeystore "$CERT_JKS" \
        -deststoretype jks \
        -deststorepass "$JKS_PASS"

# mv to ssl/
mkdir -p /usr/local/tomcat-ssl/
mv $PRIVKEY /usr/local/tomcat-ssl/ && \
        mv $CERT /usr/local/tomcat-ssl/ && \
        mv $CERT_P12 /usr/local/tomcat-ssl/ && \
        mv $CERT_JKS /usr/local/tomcat-ssl/

# remove redundant files
rm extensions.ini "$CSR"
```

```
cat <<EOF
`tput bold`
Success!
Your brand new certificate has been written to "ssl/$CERT", "ssl/$CERT_P12", "ssl/$CERT_JKS".
Your private key has been written to "ssl/$PRIVKEY".
Share the certificate with everyone, and the key with no one!
`tput sgr0`
EOF
```