项目开发总结报告

# 引言

**软件项目的任何文档都包含引言部分，需介绍文档的编写目的、背景、预期读者、以及参考资料，还可包含文档历史演化版本。在课程实践中，引言部分可不必填写。**

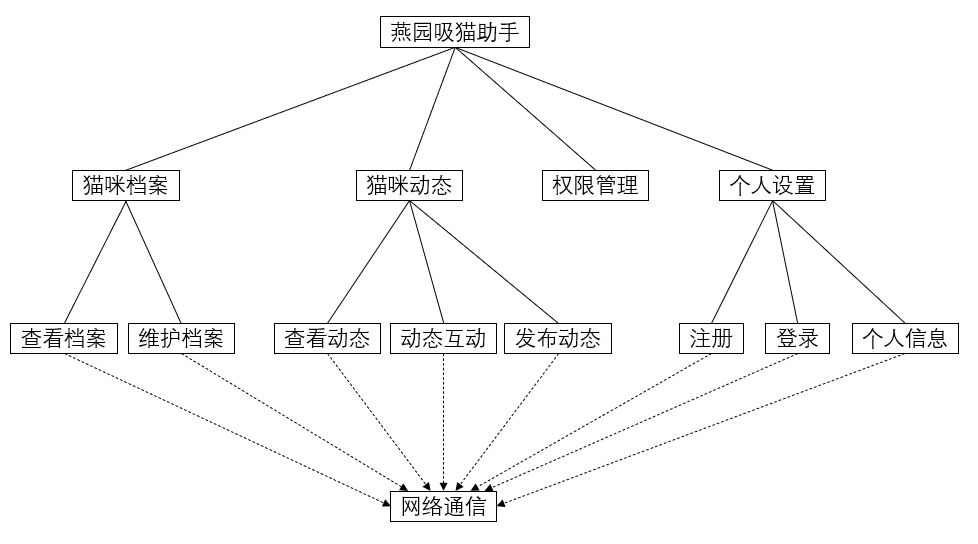
# 实际开发结果

## 产品

说明最终制成的产品，包括：

1. 程序系统中各个程序的名字，它们之间的层次关系；

程序系统主要由“猫咪档案”“猫咪动态”“权限管理”“个人设置”“网络通信”等模块构成，其中每个模块下面还包括了一些子模块。它们之间的层次关系如下：



1. 程序系统共有哪几个版本，各自的版本号及它们之间的区别；

目前只有一个版本，版本号为0.9。

1. 每个文件的名称；

见软件包。

1. 所建立的每个数据库。

系统建立的数据库共有10个数据表，如下：

用户类：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用户ID | 用户名 | 密码 | 邮箱 | 是否为管理员 | 头像 | 个性签名 |

验证码类：

|  |  |  |  |
| --- | --- | --- | --- |
| 验证码ID | 邮箱 | 验证码 | 上次请求时间 |

猫咪类：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 猫咪ID | 猫咪名称 | 照片 | 介绍 | 相关猫咪 |

猫咪关系类：

|  |  |  |
| --- | --- | --- |
| 猫咪1ID | 猫咪2ID | 猫咪关系 |

动态类：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 动态ID | 发布者ID | 发布时间 | 文本 | 是否为视频 | 多媒体资源 | 点赞数量 |

评论类：

|  |  |  |  |
| --- | --- | --- | --- |
| 评论ID | 动态ID | 评论用户ID | 评论内容 |

点赞类：

|  |  |  |
| --- | --- | --- |
| 点赞ID | 动态ID | 点赞用户ID |

关键词类:

|  |  |  |
| --- | --- | --- |
| 关键词ID | 动态ID | 关键词内容 |

饲养员类：

|  |  |  |
| --- | --- | --- |
| 饲养员ID | 用户ID | 猫咪ID |

饲养员申请类：

|  |  |  |
| --- | --- | --- |
| 申请ID | 用户ID | 猫咪ID |

## 主要功能和性能

逐项列出本软件产品所实际具有的主要功能和性能，对照项目开发计划、功能需求说明书的有关内容，说明原定的开发目标是达到了、未完全达到、或超过了，并分析原因。

|  |  |  |  |
| --- | --- | --- | --- |
| **立项目标** | **实际情况** | **偏差有无** | **原因分析** |
| 用户注册，登录 | 达到了 | 无 |  |
| 发布、浏览、检索猫咪信息 | 达到了 | 无 |  |
| 发布、浏览、检索、评论或点赞猫咪社区动态 | 达到了 | 无 |  |
| 给猫咪社区动态添加tag | 未达到 | 有 | 开发时间不足 |
| 查看猫咪关系图谱 | 达到了 | 无 |  |
| 用户申请成为猫咪管理员，审查猫咪信息的更新更正 | 达到了 | 无 |  |
| 【扩展需求】根据照片自动识别校内猫咪 | 未达到 | 有 | 开发时间不足 |

## 进度

列出原定计划进度与实际进度的对比，明确说明，实际进度是提前了、还是延迟了，分析主要原因。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **里程碑** | **预定日期** | **实际日期** | **偏差有无** | **原因分析** |
| 选题及开发计划制定 | 2020.3.14 | 2020.3.14 | 无 |  |
| 结构化需求分析 | 2020.3.19 | 2020.3.19 | 无 |  |
| 面向对象分析 | 2020.4.12 | 2020.4.12 | 无 |  |
| 面向对象设计 | 2020.4.27 | 2020.4.27 | 无 |  |
| API设计 | 2020.4.5 | 2020.4.5 | 无 |  |
| 第一阶段编码 | 2020.5.18 | 2020.6.3 | 有 | 开发难度超出预期 |
| 第一阶段单元测试 | 2020.5.24 | 2020.6.3 | 有 | 开发难度超出预期 |
| 集成测试、确认测试与系统测试等 | 2020.6.3 | 2020.6.3 | 无 |  |

# 开发工作评价

## 对生产效率的评价

并列出原订计划数作为对比。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **项目** | **实际生产效率** | **计划生产效率** | **比较结果** | **原因分析** |
| 程序 | 1200行 | 2500行 | 实际生产效率低于原计划计划 | 小组成员对于开发工具不熟练 |
| 文件 | 3000字 | 3000字 | 基本持平 | 组内在文档写作方面分工明确 |
| 测试 | 30 | 50 | 实际生产效率低于原计划计划 | 小组成员缺乏软件测试经验 |

## 对产品质量的评价

说明在测试中检查出来的程序编制中的错误发生率为（每千条指令语句中的错误语句数）。

## 对技术方法的评价

**Django**

* 后端开发使用python的Django框架。Django基于MVC模型，拥有MVC设计模式的所有优点。和上手比较简单的Flask框架相比，Django作为一种面向相对大型应用的框架，提供模块更加全面，开发也仍然方便。虽然我们的应用并没有很“大型”，但有机会可以接触这样一个全面的框架，从而对应用开发框架有更深刻全面的理解，是非常宝贵的经历。
* Django自带ORM关系对象映射，自带数据库操作功能，能够根据对接的数据库引擎翻译成对应的sql语句，无需过多关心使用的究竟是哪种数据库引擎，数据库迁移也只需要更换Django的数据库引擎，比如我们使用的数据库引擎是MySQL，只要完成在settings.py等地方完成设置，具体的实现也不需要太关注各种数据库引擎的不同。
* Django有非常方便的自助管理后台，不需要写多少代码就可以拥有一个完整的后台管理界面。
* Django的URL模块设计很简洁便利，我们按照RESTful API接口设计标准规范访问资源，只需要设置project和app中的urls.py文件。
* Django的报错信息非常详细易懂，对新手友好。
* Django自带Django.test，可以直接使用它来进行单元测试。

**MySQL**

* 我们选择使用MySQL数据库，MySQL是一种关系型数据库，采用基于关系表的存储方式，关系表易于理解和维护。MySQL支持多线程、多处理器，能够充分利用CPU资源；支持多种平台、多种语言，包括我们所使用的python Django；SQL查询算法优秀，查询速度快。

**Uwsgi**

* 我们使用python的一个模块uwsgi来实现基于uwsgi+Nginx下django项目生产环境的部署。uwsgi安装只需要简单的pip命令，编辑uwsgi.ini文件即可完成对django项目的配置和启动。
* uWSGI实现了uwsgi协议、WSGI协议、http等协议，具有占用内存少、性能优速度快、多app管理、日志功能详尽等特点。uWSGI服务器自己实现了基于uwsgi协议的server部分，我们只需要在uwsgi的配置文件中指定application的地址，uWSGI就能直接和应用框架中的WSGI application通信，非常便捷。

**Nginx**

* 我们使用Nginx这款轻量级且高性能的HTTP和反向代理Web服务器，具有占用内存少、启动极快、高并发能力强等优点。Nginx中的HttpUwsgiModule与uWSGI服务器进行交换。

**Android Studio**

* 我们前端实现的是Android开发，使用Android Studio集成开发环境
* 采用 Java 语言来开发。开发完成后，Android 应用程序可以容易的打包。
* 可以基于模板的向导来生成常用的Android应用设计和组件
* 可以基于[Gradle](https://zh.wikipedia.org/wiki/Gradle)支持构建
* 内置[Android SDK](https://zh.wikipedia.org/w/index.php?title=Android_SDK&action=edit&redlink=1" \o "Android SDK（页面不存在）)和[AVD](https://zh.wikipedia.org/w/index.php?title=AVD&action=edit&redlink=1" \o "AVD（页面不存在）)管理器

**Postman**

* 我们在部分单元测试（后端）和性能测试中使用了Postman。Postman有图形界面，门槛低上手快，只要创建测试集，根据接口测试文档填写请求参数和断言；也有命令行版本方便在服务器上运行；支持各种请求类型；支持自定义HTTP Request；支持Pretty Response；跨平台；支持读取数据文件；支持抓包，保存历史记录，支持在线存储数据，通过账号迁移数据，多终端同步用例，支持分类收藏和共享。

**Apache Benchmark**

* 我们使用Apache Benchmark来进行压力测试，这是Apache附带的web压力测试工具，使用方便，统计功能强大，能够在本地环境本地命令行发起测试请求来验证服务器的处理性能，不参杂网络传输时间的影响。ab命令会创建很多的并发访问线程，模拟多个访问者同时对某一URL进行访问，可用来测试Nginx等Web服务器的负载压力。

## 出错原因的分析

错误：发送POST请求返回403，csrf验证失败。

原因：django中的POST默认开启了csrf验证，在使用POST方法发送数据时需要先向服务器获取一个csrf\_token，在POST时一起发送到服务器。如果不发送，就会返回403。这里我们直接在django中关闭了csrf。

错误：使用https访问服务器失败。

原因：https需要验证服务器的证书，这里我们跳过验证来解决这个问题。

错误：django在非DEBUG模式下无法访问static资源

原因：django默认static资源只能在DEBUG模式下访问，因此通过nginx来提供static资源。

错误：django访问数据库出错

原因：一个人更新了自己的models并migrate到数据库但未commit到github上，其他人的代码中的models的字段就与数据库中的字段不符，在查询数据库时就会出错。因此要及时同步代码到github上。

错误：在申请饲养员通过、发布动态之后，饲养员、动态列表没有显示新的内容。

原因：新增加的信息没有加到本地变量中，新增每次提交信息后会刷新信息。

错误：使用前端传过来的整数值时出错

原因：后端获得的参数是str类型，获取int值时需要类型转换。

# 经验与教训

列出从这项开发工作中所得到的最主要的经验与教训及对今后的项目开发工作的建议。

## 进度控制

小组严格按照甘特图规划进行项目管理，一定程度上保障了项目的进度要求。但是甘特图的规划不够细致，也造成了一些不利影响，比如没有考虑扩展功能需求。

## 定期交流

小组成立以来，基本维持每周一次的小组讨论会，前后端内部也进行过多次讨论，良好的交流是软件能够顺利开发的重要因素。

## git应用

git是帮助软件开发团队进行协作开发的重要工具，应该充分利用git

进行团队开发。

## 及时测试

软件测试的目的在于发现错误，及时测试并改正，可以避免该错误对后续功能或性能要求的影响。小组虽然在前后端分别完成了软件测试过程，但没有及时完成前后端交互这一内容。

## 非功能需求

要认识到非功能需求的重要性，对于非功能需求要有准确的界定，并且认真设计并实现非功能需求。我们小组针对软件的非功能需求也有特别的处理，比如简化功能逻辑提高性能，对于软件的安全性和易用性也进行了特别的处理。

## 小组成员感想

**王昱斐**

第一次参与一个完整的应用软件开发，最大的障碍就是“万事开头难”，无论是github的使用还是Django和MySQL等等的实现，都是几乎从零开始摸索，经常会有不知所措的时候，关于这点的经验教训是除了自己闷头查解决方法以外，一定要及时求助和与队友沟通，团队合作才能事半功倍。

这次不仅仅是写代码，而是真正体验了一个软件被开发出来的全过程，和大家一起讨论、确定软件需求，一起进行结构化分析与设计、面向对象分析与设计等等，体验了以后去企业工作会用到的严谨的软件开发。

Github的使用经验也是一大收获，因为之前使用得不多需要适应，我可能会没有及时提交代码的习惯，甚至对于把自己的分支merge到后端分支或者总分支有一些恐惧，生怕有什么错误会影响到其他组员。逐渐地才明白及时提交、及时merge，有问题及时沟通才是真正高效的，以后我会注意这些养成好的习惯。

抱着学习的心态加入后端开发小组，“后端”真正在我眼中具象了起来：了解了Django的MTV设计模式，了解了如何创建模型和配置数据库，了解了如何用不同视图响应不同URL地址的请求，了解了如何测试请求与响应。这些都是我非常珍视的收获。

**程轶**

通过软件工程课程实践，我比较完整地体验了一个软件开发的全过程，学习掌握了集中常用的软件开发工具。软件开发过程比我原来想象的要更加复杂困难。要想开发一个好的软件，需要一整个团队的通力协作。并不是考一个能力极强的人全程carry就能完成的。通过团队协作，达到1+1>2的效果，也使得整个开发效率提高，节省时间。

我的另一大感受就是软件开发远远不只是写代码那么简单，这与我原来的认识也很不一样。应该说写代码只是软件开发的较小一部分。除了代码，我们还要花大量的时间确认需求、书写文档、进行测试。软件工程也绝不只需要一个人码力强劲，更需要有良好的沟通能力、团队意识。如果要做一个team leader，则要求更高，需要有很强的组织能力。

**叶文沁**

在软工课程学习和项目参与中，我收获了很多，不仅是知识，还有开发经验、技巧，以及团队协作能力，还有看到项目一点点的搭建起来的成就感。在这一过程中，对于软件开发这一概念也有了更切实的理解和体验，跳出了软件开发等同于代码实现的浅显认知；与此相对的，也让我认识到除了技术水平，开发经验、团队合作能力对于一个软件开发人员有着同等重要的地位，明白了软件开发中的各环节和负责各项工作的人员都是不可或缺的存在。

**余卓**

这次是我大学期间第一次参与一个团队合作的大项目。这次软件工程的实践，从最开始的选题设计，到之后的结构化分析与面向对象分析设计，再到之后开始编写软件，对软件进行测试，包括我在内的每个小组成员都全程参与，贡献自己的力量。这是我之前从未有过的经历，是我的学习生活中难得的一次实战演练。并且在这个过程中，我学习到了很多有用的软件工程思想，也学习到了很多实用技能，比如github的基本使用，基于Android Studio的软件开发，前端界面设计，前后端数据通信，后端数据管理等，可谓是收获满满。感谢组内的各位小组成员和组长的辛勤付出，也感谢老师和助教们的指点和帮助。

**张石然**

软件工程这门课程真的让我收获很大。第一次参与一个完整的大项目，虽然最后的实现成果与一开始预期出入比较大，但这过程中我成长了很多。

首先是第一次做项目组长，第一次意识到一个团队的领导对整个团队的氛围和进度影响都很大。我自己有很多地方做得不充分，通知不到位，沟通不及时，导致大家之间的理解有错位。协作最关键的是交流，尤其是大部分成员都第一次参与，更要沟通及时。而组长要协调好，计划好进展。

其次是从零开始学习技术并实践运用。前端是第一次接触，Android Studio也是第一次用，之前也没有写过Java语言的大项目，因此这次难度几乎是地狱级的。而我们前端的四个人，大家都是第一次接触安卓开发。我作为组长，先学习了技术，搭了前端的整个框架。一开始上手确实很难，但是过程中对安卓开发都有了更全面详细的了解，也学习了很多常见错误的处理。

最后是有关自主学习和协作开发。疫情在家，线上交流开发，相当不易。第一次全面使用GitHub协作开发，困难重重。要处理不同开发者之间merge的冲突，功能实现的落差等等。但一个学期下来，对git的操作熟悉了不少，相信之后再做类似的工作会更加熟练和顺利。