

{:toc}

一、开发

此部分将介绍项目的开发人员、计划、开发环境和工具

开发人员

学号	昵称	Github	岗位角色
16340213	阿里武*	wangjiwu	项目经理、产品经理，架构师
16340204	汤小纪*	13326651141	客户经理、UI/UX设计师，QA 工程师
16340205	小鹏*	Walikrence	Python工程师， 数据库DBA
16340111	李冰	Lyrix28	Python工程师，数据库DBA
15353005	范瑞潮*	Sefaice	Js工程师， DevOps 工程师

开发计划

本项目采用敏捷开发方式，项目共分为4个迭代，分别是 项目启动、问卷系统迭代、小任务系统迭代、测试和文档完善， 以下是每个迭代的具体计划。

- **inceptions 1 项目启动（ 2019-03-27 ~ 2019-04-19）**
 - 目标:
 - 开会讨论项目课题， 架构， 分工
 - 完成市场调研， 项目愿景， 项目规格说明书
 - 注册登录实现
- **Iteration 2 问卷系统迭代 (2019-04-20 ~ 2019-05-10)**
 - 目标:
 - 完成API文档设计
 - 完成数据库设计
 - 完成问卷系统前端逻辑
 - 完成问卷系统后端逻辑
- **Iteration 3 小任务系统迭代 (4 周 2019-05-10 ~ 2019-06-10)**
 - 目标:
 - 完成个人博客
 - 项目文档整理
 - 小任务系统实现
 - 完成问卷系统后端逻辑
- **Iteration 4 测试和文档完善 (2周 2019-06-15 ~ 2019-06-25)**

- 目标:
 - 完成技术报告
 - 后端测试: 李冰 & 汤万鹏
- 项目测试: 汤浩林
 - 前端测试: 范瑞潮
- 项目文档完善: 王继武

开发环境和工具

- 前端: Windows 10 + sublime
- 后端: Windows 7 + pycharm
- UI: Windows10 + AXURE RP
- 项目管理 windows + TAPD、Umlet、

二、软件总体设计

软件总体设计分为 5 个部分

- 产品设计: 介绍项目产品的设计愿景, 方法, 成果
- 界面设计: 说明各个界面, 已经使用的UI和交互设计方法
- 前端设计: 说明前端的技术选型及缘由、模块方法、软件设计技术
- 后端设计: 说明后端的技术选型及缘由、模块方法、软件设计技术和数据库设计
- 架构设计: 说明项目的前后端架构

1. 产品设计

1.1 设计方向

设计目的

我们的初衷是做一款让大学生能够轻松使用的多功能网页。众所周知, 大学生要时常发布问卷进行问卷调查, 并且有许多的闲置物品不方便交易, 我们这个网站旨在让大学生能够利用闲暇时光, 完成一些小任务, 填写问卷, 赚取闲钱, 以及交易物品

市场现状

现在在市面上有不少这样的问卷调查软件, 也有发布小任务小兼职的网站, 但是它并不是面向大学生群体, 而且鱼龙混杂不利于大学生的安全交易, 我们设计的目的就是面向我们大学生, 让大学生可以在一个相对熟悉的小圈子里面安全交易。

面向人群

同校以及相邻学校, 熟悉手机APP使用, 网站应用的在校大学生。

1.2 设计参考/竞品分析

市场趋势

和以前大不相同，现在的软件针对性更强，面向更加特殊的人群。换言之，每种产品都有它不同的客户目标，并在此基础上最大程度满足用户的产品需求，因此如果A群体想要产品让他能够调研社会上所有的用户，而B群体想要产品针对性的为大学生而设计，那么我们的产品就不可能完全满足这两组用户。因此我们设计的产品是面向大学生的，我们有信心我们的产品能在这块做到最好

确定竞品

- 目的：进一步了解产品所属相关领域的发展程度和前景，对竞品进行多方面分析，探索产品的竞争优势，并对产品今后的设计提供参考方向和建议。
- 过程：观察身边使用率较多的交易APP和问卷系统
- 结果：我们最终选择了闲鱼APP和问卷星APP

竞品对比

问卷星

- 目标人群：进行定量研究收集数据的用户
- 产品定位：简易的问卷发布和填写系统
- 设计特点：UI简单清晰，问卷发布和填写方式都很简单
- 产品优劣： 优点：面向全社会的人群，可以收集到不同人群的数据，操作简单 缺点：针对性不强，交互性不强

闲鱼

- 目标人群：有闲置物品想交易的人群
- 产品定位：发布信息网站，让用户自主交易
- 设计特点：使用简单，交易信息有归类
- 产品优劣： 优点：信息归类得很好，用户自主性强 缺点：交易方信息不透明，无法估计安全性

1.3 设计过程

（1）用户调研

调研目的

确定我们的设计方向，明确用户需求，并建立可用的用户模型，在我们模拟的场景中设计我们的产品。已求产品最大程度符合我们设计目的。

调研方法

- 采取定性研究的方法，选取10名左右的大学生进行采访，定性研究其需求
- 利用上面得到的数据，建立我们的用户模型
- 在建立用户模型之后，在特定的场景里面进行模拟
- 模拟完的数据汇总最后完成设计方案

调研过程

定性研究环节

采访大学生1:

Q: 请问你平时会使用像问卷星这样的系统吗?

A: 会啊, 之前上管理课就要用这个做数据收集

Q: 会不会觉得用问卷星收集数据全靠朋友圈转发, 很多人不愿意填呢?

A: 是这个道理, 我们经常是发个红包让别人填哈哈哈

Q: 要是能在问卷里面设置填问卷有悬赏这样会不会好一点?

A: 嗯嗯确实挺好的, 而且我希望有更多的大学生能看到我们的问卷然后填写。

采访大学生2:

Q: 同学你平时会上网卖东西吗?

A: 哦哦我懂, 你问的是闲鱼吗, 会啊

Q: 对的对的, 一般卖些什么东西呀

A: 一般都是二手书呀, 二手单车呀这些

Q: 有没有什么更具体的想法?

A: 我觉得让我能和学校里面的人交易就好了, 闲鱼太杂我有点不放心

(2) 用户模型的建立

用户特点

- 更倾向于在学校的大范围内发布问卷
- 更倾向于和认识的人, 一个学校的人交易
- 希望功能齐全, 能够满足大学生生活的许多要求

(3) 场景假设

学生王建同学有一天在管理课上面接到任务, 需要做定量的调查收集数据, 然后他打开我们的网站, 设置好问卷问题, 输入悬赏金额而后发布出来。很快因为有悬赏金额, 因此有人就马上来答题了, 收集完了数据, 王建同学很快就完成了作业, 然后他很开心的打开游戏开始玩耍。这时候他的舍友来了, 跟他说: 你的冬天被子把箱子塞满了, 我们没地方放东西。王建感到一丝尴尬, 他想了想要怎么处理这些不用的东西, 这时候他想到了我们的网站, 于是他在网上发布了这个被子的交易信息, 由于在一个学校里面, 而且学号公开, 所以信息很透明, 大家放心, 于是很快就有人出来交易了。到晚上, 王建同学肚子饿了想点外卖, 这时候他看到通知里面有帮拿外卖的, 他顺便下楼拿自己的同时接了这个任务, 于是一碗饭的钱又到手了

1.4 最终设计

最后我们决定设计一款软件, 功能包括

- 问卷发布和填写
- 任务发布和接收
- 校园内交易

2. 界面设计

2.1 组成部分

我们的界面主要分为3个

- 主界面
- 问卷系统界面
- 小任务系统界面

2.2 各个界面介绍

主界面

预览



界面元素

- 主界面包括导航栏，登录按钮和问卷，小任务系统按钮
- 导航栏包括主界面导航，问卷系统导航，和社区导航
- 登录按钮跳转到登录或是注册界面
- 问卷按钮跳转到问卷系统，小任务系统跳转到任务系统

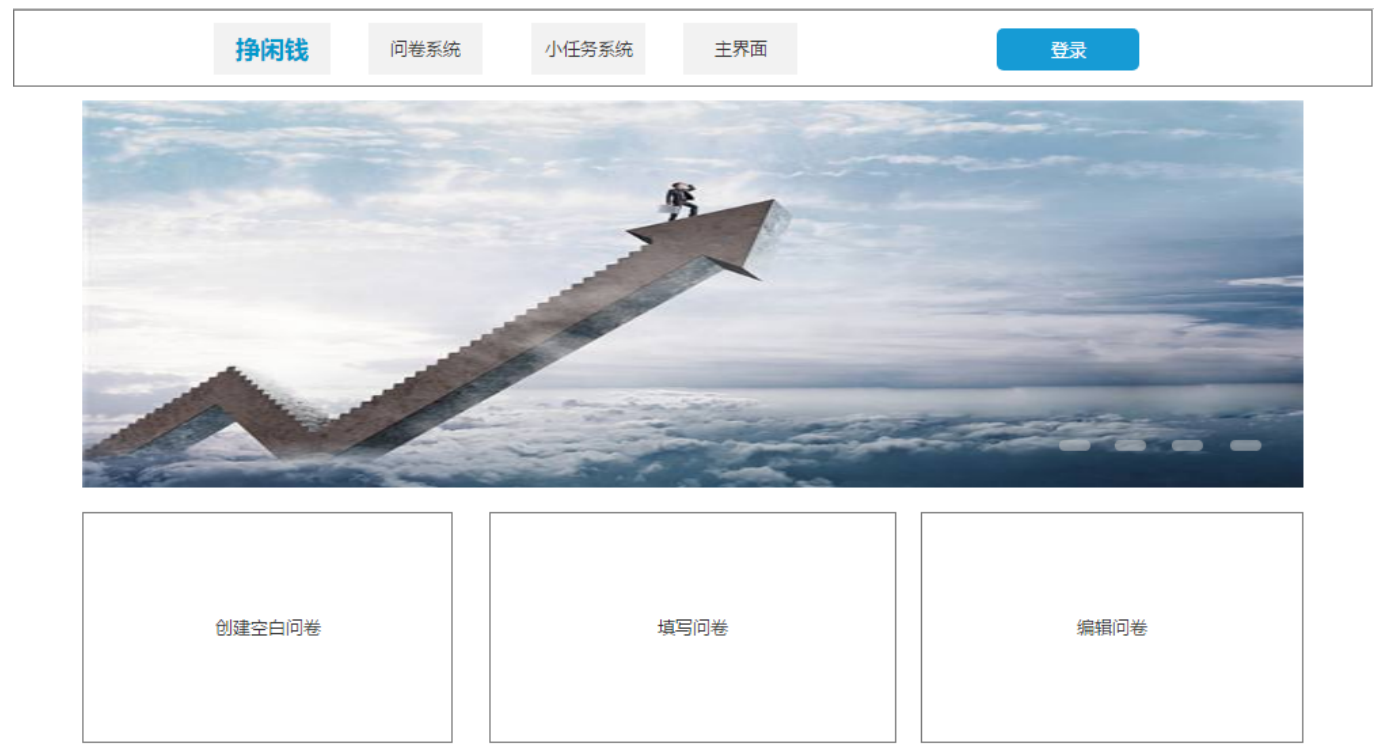
视觉效果

- 控件有明显的视觉提示，使用的时候就很清晰的知道什么地方可以做什么
- 暗色调的底部图片很好的和我们的主题“不只是问卷调查”形成鲜明的对比，上方问卷系统，小任务系统，社区系统控件用白色字也很好的突显，用户找起来不麻烦

- 挣闲钱三个大字用浅蓝色，突出主题的同时也契合背景，问卷调查和小任务系统用浅色调，弥补整个图片只有深色调的空白，显得不那么单调。
- 留有空白，同时该有的都有，设计简约而美观

问卷系统界面

预览



界面元素

- 问卷系统界面包括上方导航栏，中间跑马灯广告，下方功能跳转区域
- 导航栏包括主界面导航，问卷系统导航，和社区导航
- 登录按钮跳转到登录或是注册界面
- 下方的跳转区块跳转到我们问卷系统的主体

视觉效果

- 控件色调以浅色系为主，不会给用户带来审美疲劳
- 下方的跳转区域简单明确，麻雀虽小五脏俱全
- 有留白，看起来舒适

小任务界面

预览



界面元素

- 小任务系统界面包括上方导航栏，中间跑马灯广告，下方功能跳转区域
- 导航栏包括主界面导航，问卷系统导航，和社区导航
- 登录按钮跳转到登录或是注册界面
- 下方的跳转区块跳转到我们小任务系统的主体

视觉效果

- 控件色调以深色浅色搭配，对比明确美观而清晰
- 下方的跳转区域简单明确，这只是我们的初稿，在成品时该部分还会加上装饰
- 有留白，看起来舒适

2.3 总体评价

可以看到我们在主界面下还加了其他内容（用侧滑栏划下就可看到）这部分是对我们功能的简单介绍，同时配有直观的图片表述主题，在下方还有跳转按钮，可以说增加的这个部分让我们的主界面更加的丰富，用户使用我们产品的第一感受就能了解到我们能做什么，我们具有什么样的特点

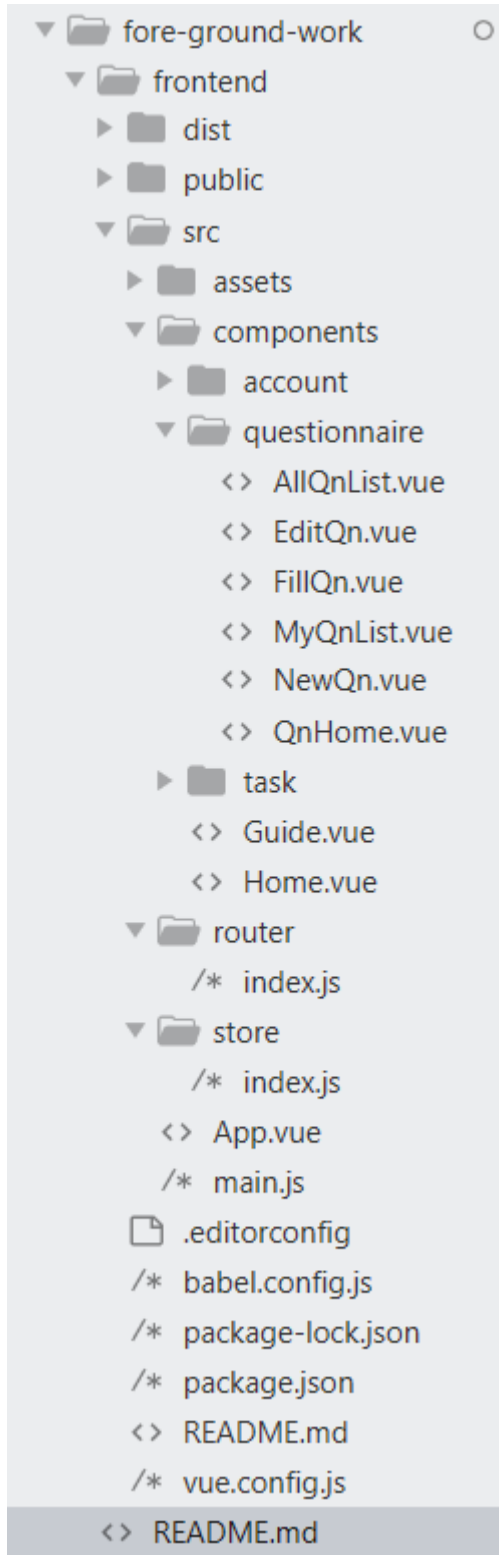
界面依旧保持我们原来的简约的想法，同时我们增加了很多的视觉元素，这让我们的界面看上去简单而美观，可以说是一次全方位的优化，如果说还有不足，那么我觉得就是字体上面，控件的交互方面可以做得更好，比如说问卷系统，小任务系统和社区系统可以增加一个很浅的有立体感小边框，这样看上去交互性就更强，我们字体可以设置渐变，这样让我们的界面看上去更加生动而不会像现在这样呆板。

3. 前端设计

模块划分

vuejs的MVVM架构让开发过程专注于页面的编写，我们的前端部分共由以下几部分页面组成：

- 注册登录界面
- 主导航界面
- 问卷调查的系列页面
- 小任务系统的系列页面



前端代码树形图(用工具[treer](#)生成):


```
├─babel.config.js
├─package-lock.json
├─package.json
├─README.md
├─vue.config.js // vue配置文件
├─src
|   ├─App.vue // 主组件
|   ├─main.js // 项目入口
|   ├─store
|   |   └─index.js // vuex文件
|   ├─router
|   |   └─index.js // vue-router配置文件
|   ├─components // 所有子组件
|   |   ├─Guide.vue // 主导航界面
|   |   ├─Home.vue // 根界面
|   |   ├─task // 小任务系统组件
|   |   |   ├─NewTask.vue
|   |   |   ├─TaskDetail.vue
|   |   |   ├─TaskHome.vue
|   |   |   └─TaskList.vue
|   |   ├─questionnaire 问卷调查组件
|   |   |   ├─AllQnList.vue // 全部问卷列表页面
|   |   |   ├─EditQn.vue // 编辑问卷页面
|   |   |   ├─FillQn.vue // 填写问卷页面
|   |   |   ├─MyQnList.vue
|   |   |   ├─NewQn.vue // 新建问卷页面
|   |   |   └─QnHome.vue // 问卷根页面
|   |   ├─account
|   |   |   ├─Signin.vue // 登陆页面
|   |   |   └─Signup.vue // 注册页面
|   └─assets
|       ├─avatar.jpg
|       ├─favicon.ico
|       ├─home-carousel-1.jpg
|       ├─home-carousel-1.png
|       ├─home-carousel-2.jpg
|       ├─home-carousel-3.jpg
|       ├─home-carousel-4.jpg
|       ├─home-detail-1.jpg
|       ├─home-detail-2.jpg
|       └─qncover.jpg
├─public
|   ├─index.html // html文件
|   └─static
|       └─favicon.ico
├─dist // 打包目录
|   ├─index.html
|   └─static
|       ├─favicon.ico
|       └─js
|           ├─app.c7868cb2.js
|           ├─app.c7868cb2.js.map
|           └─chunk-vendors.552d5829.js
```

```

| | | | chunk-vendors.552d5829.js.map
| | | | img
| | | | | avatar.f5009b8c.jpg
| | | | | home-carousel-1.70735838.png
| | | | | home-carousel-2.8b347fd3.jpg
| | | | | home-carousel-3.d201e704.jpg
| | | | | home-carousel-4.4d24e122.jpg
| | | | | home-detail-1.bbc5b2ce.jpg
| | | | | home-detail-2.541c6dfc.jpg
| | | | css
| | | | | app.1408651e.css
| | | | | chunk-vendors.53600669.css

```

关键实现

- **vuex全局状态存储。**vuex可以存储用户的登录状态及账户信息，而刷新有可能丢失数据，因此我采用了localStorage存储vuex数据，监听浏览器刷新事件，在刷新前后读写localStorage，实现了刷新和关闭浏览器都不会丢失用户信息。

```

created: function () {
  // 在页面加载时读取localStorage里的状态信息
  if (localStorage.getItem('store')) {
    this.$store.replaceState(Object.assign({}, this.$store.state,
    JSON.parse(localStorage.getItem('store'))))
  }
  // 在beforeunload页面刷新时将vuex里的信息保存到localStorage里
  window.addEventListener('beforeunload', () => {
    localStorage.setItem('store', JSON.stringify(this.$store.state))
  })
}

```

- **页面访问权限。**只有登录过的用户才能访问应用界面，否则会跳转到登录界面，采用vue-router全局守卫监听路由跳转，检查vuex中存储的登录状态，对未登录的情况进行跳转。

```

router.beforeEach((to, from, next) => {
  if (to.path !== '/signin' && to.path !== '/signup') { // if jumping to ~sign
  page, check if login
    if (store.state.userInfo == null) {
      next('/signin')
    }
  }
  next()
})

```

- **与后端对接。**我们项目的后端采用的是django框架，由于前后端分离的项目特性，前后端对接的大部分工作都是数据交互，axios可以设置一个全局请求url的置换，在开发时将数据转发到后端服务器端口，在开发完成打包时取消这个转发。

4. 后端设计

技术选型及理由

django

Django 是以 Python 编写的高级，MVC 风格的开源库。Django 也被称为“完美主义者的最后框架”，它最初是为新闻网站设计的，并且允许开发人员编写数据库驱动Web应用程序，因此无需从头开始编码。

除了更快完成常见的 Web 开发任务，Django 还能使设计过程干净务实。Django 是新的 Python Web 开发人员的最佳选择，其官方文档和教程是软件开发中最好的。

在技术市场充斥着一系列网络框架，但 Django 一直是最受欢迎的服务器端 Web 框架。设计 Django 最初的原因是：不要重复造轮子。Django 是用 Python 编写的，通过最小化编写代码来提高效率，再加上云平台的支持，使Django 成为 Web 开发者最受欢迎的选择。

Django的主要特点

1. Django配有“Batteries-Included” Django 基于 “Batteries-Included” 的理念，可不必使用单独的库来实现常见功能，例如身份验证，URL 路由，模板系统，对象关系映射器（ORM）和数据库模式迁移。如果您正在使用或使用 Flask，您必须注意到它要求一个单独的库，如用 Flask-Login 来执行用户身份验证，而 Django 不必这样做。
2. 免费 API 使用 Django，可以根据您的模型生成 Python API，不需要额外的编码就能够生成 API 了。
3. 独特的管理界面 即使在网站完全构建之前，您也可以从外部贡献者的网站上获取有关信息，这就是 Django 的优点。该框架使您能够快速轻松地从应用程序的模型中创建一个管理站点。
4. 代码布局 与大多数 Web 框架相反，Django 通过使用称为 application 的东西，更容易地将新功能插入到产品中。因此，开发人员一直被鼓励编写自包含的代码。
5. Django 的 ORM 专注于数据库 Django 的对象关系映射器（ORM）负责处理数据库,所以没有处理结构化查询语言（SQL）那样的麻烦，它主要用于查询数据库所需的数据。与许多通过SQL 直接在数据库上工作的 Python 框架不同，Django 开发人员有一个独特的工具来操纵相应的 Python 模型对象。Django 通过使用 PostgreSQL，MySQL，SQLite 和 Oracle 等关系数据库管理系统来实现开箱即用的功能。

模块划分

```
├─ api.md # api文档
├─ db.sqlite3
├─ dist # 静态文件夹
│   ├── index.html
│   └── static
│       ├── css
│       │   ├── app.88022460.css
│       │   └── chunk-vendors.3b2d6e5b.css
│       ├── favicon.ico
│       └── img
│           ├── avatar.f5009b8c.jpg
│           ├── home-carousel-1.70735838.png
│           ├── home-carousel-2.8b347fd3.jpg
│           ├── home-carousel-3.d201e704.jpg
│           └── home-carousel-4.4d24e122.jpg
```

```

├── home-detail-1.bbc5b2ce.jpg
├── home-detail-2.541c6dfc.jpg
├── js
│   ├── app.135b342f.js
│   ├── app.135b342f.js.map
│   ├── chunk-vendors.55b965b0.js
│   └── chunk-vendors.55b965b0.js.map
├── makefile
├── manage.py # 管理Django 程序
├── mission # 任务模块
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-37.pyc
│   │   ├── admin.cpython-37.pyc
│   │   ├── models.cpython-37.pyc
│   │   ├── urls.cpython-37.pyc
│   │   └── views.cpython-37.pyc
│   ├── admin.py
│   ├── apps.py # 模块主入口
│   ├── migrations # 数据库操作记录->相应表结构发生变化
│   │   ├── __init__.py
│   │   └── __pycache__
│   │       └── __init__.cpython-37.pyc
│   ├── models.py #任务模块的小模块
│   ├── tests.py #任务模块的测试模块
│   ├── urls.py#任务模块的路由定义
│   └── views.py #任务模块的视图函数
├── mypro
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-37.pyc
│   │   ├── settings.cpython-37.pyc
│   │   ├── urls.cpython-37.pyc
│   │   └── wsgi.cpython-37.pyc
│   ├── settings.py # 配置文件
│   ├── urls.py
│   └── wsgi.py
├── paper
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-37.pyc
│   │   ├── admin.cpython-37.pyc
│   │   ├── models.cpython-37.pyc
│   │   ├── tests.cpython-37.pyc
│   │   ├── urls.cpython-37.pyc
│   │   └── views.cpython-37.pyc
│   ├── admin.py # 用于注册数据库中数据表
│   ├── apps.py # 配置当前APP
│   ├── migrations #数据库操作记录->相应表结构发生变化
│   │   ├── 0001_initial.py
│   │   ├── __init__.py
│   │   └── __pycache__
│   │       └── __init__.cpython-37.pyc
│   └── models.py # 文件系统的小模块

```

```

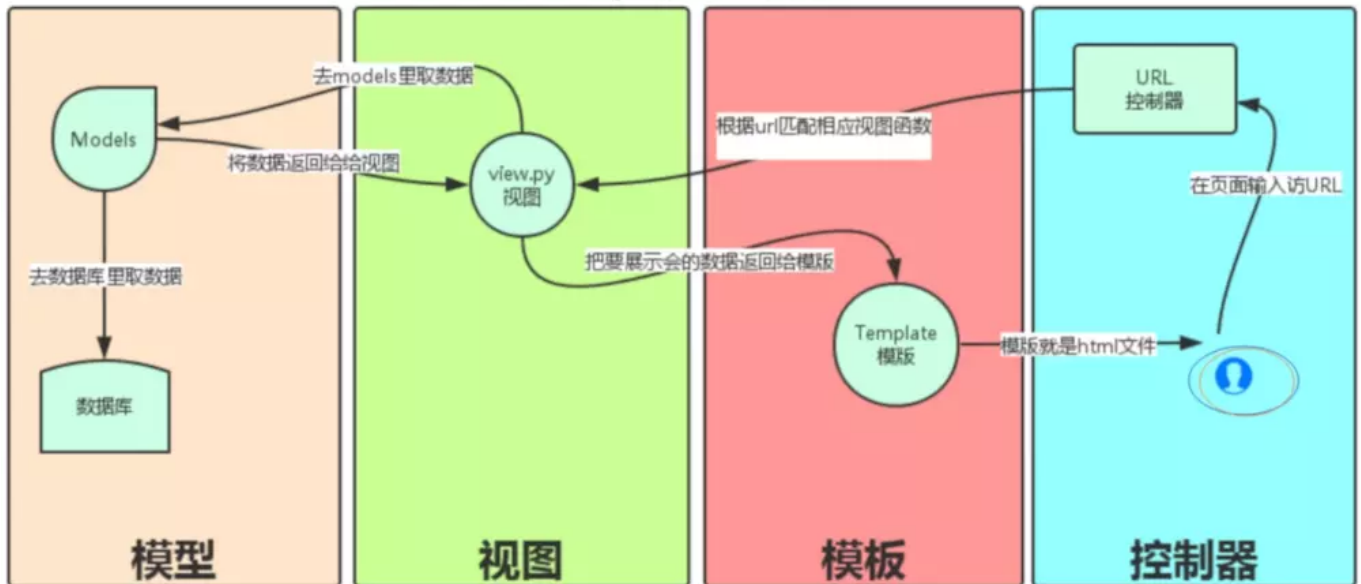
├── tests.py # 测试模块
├── urls.py # 路由定义
├── views.py # 视图定义
├── users
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-37.pyc
│   │   ├── admin.cpython-37.pyc
│   │   ├── models.cpython-37.pyc
│   │   ├── test.cpython-37.pyc
│   │   ├── tests.cpython-37.pyc
│   │   ├── urls.cpython-37.pyc
│   │   └── views.cpython-37.pyc
│   ├── admin.py # 用于注册数据库中数据表
│   ├── apps.py # 子程序入口
│   ├── migrations # 数据库操作记录->相应表结构发生变化
│   │   ├── 0001_initial.py
│   │   ├── 0002_auto_20190623_0502.py
│   │   ├── 0003_auto_20190625_0448.py
│   │   ├── 0004_auto_20190625_1359.py
│   │   ├── __init__.py
│   │   └── __pycache__
│   │       ├── 0001_initial.cpython-37.pyc
│   │       ├── 0002_auto_20190623_0502.cpython-37.pyc
│   │       ├── 0003_auto_20190625_0448.cpython-37.pyc
│   │       ├── 0004_auto_20190625_1359.cpython-37.pyc
│   │       └── __init__.cpython-37.pyc
│   ├── models.py # 用户模块
│   ├── templates # 模版
│   │   └── users
│   │       └── index.html
│   ├── tests.py # 测试模块
│   ├── urls.py # 路由模块
│   └── views.py # 视图模块

```

软件设计技术

C/S Client Server Model

CS架构：本质上django程序就是一个socket服务端，浏览器其实就是一个socket客户端。django自带的wsgi模块处理浏览器的请求信息，用户只需要实现路由和视图函数、模板等代码部分。



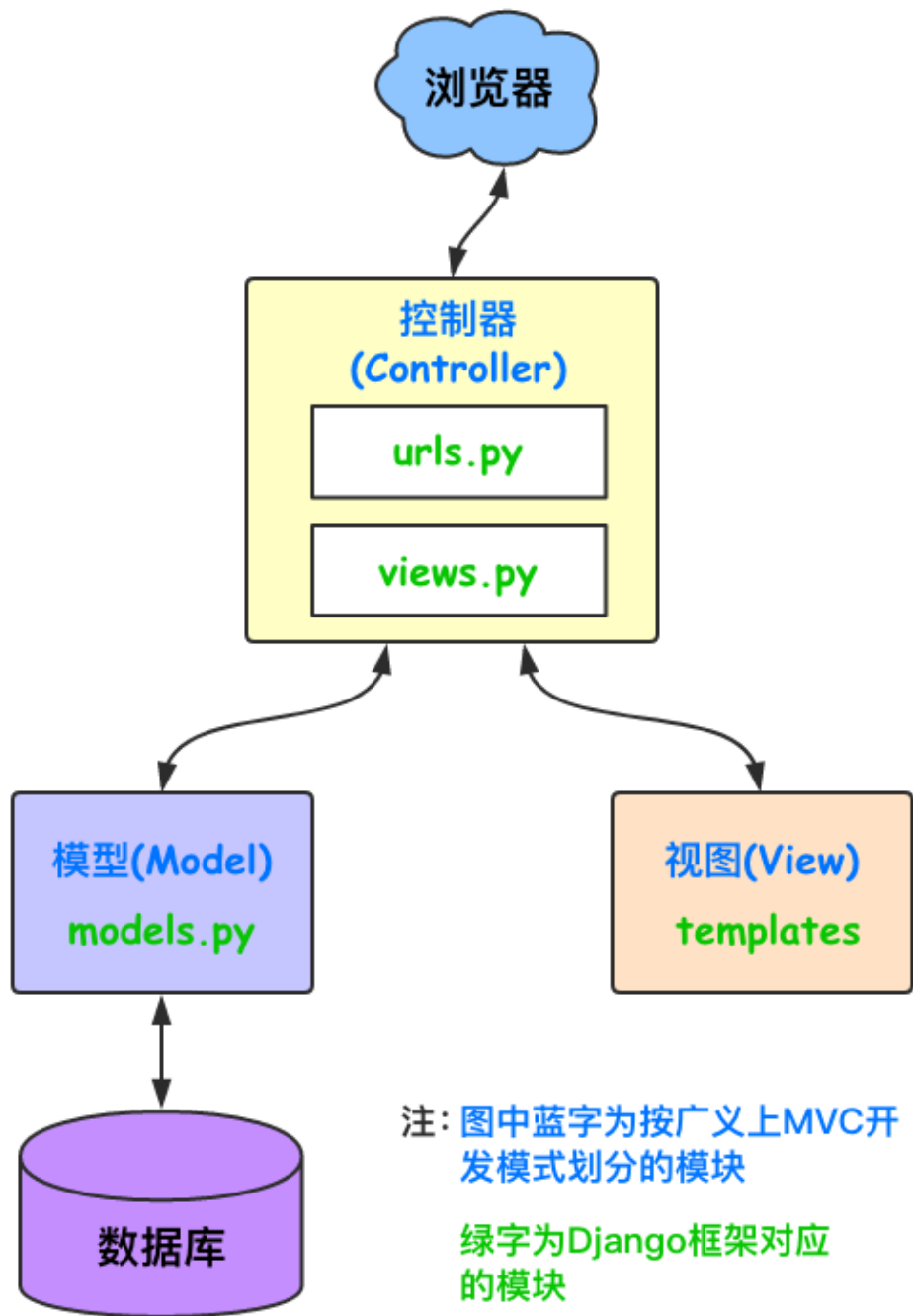
MTV

分为三个基本部分

Model(模型): 负责业务对象与数据库的对象(ORM)

Template(模版): 负责如何把页面展示给用户

View(视图): 负责业务逻辑，并在适当的时候调用Model和Template



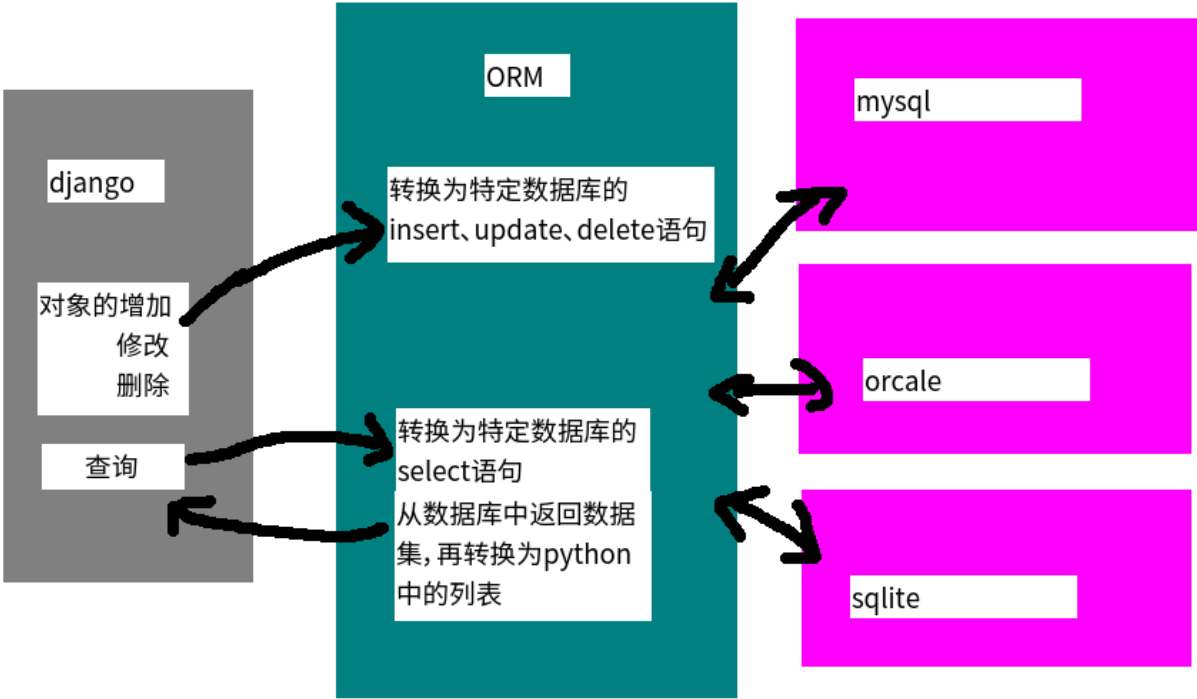
其中我们每个模块：用户、问卷、任务 都有着三个部件

名称	修改日期	类型	大小
__pycache__	2019/6/27 15:34	文件夹	
migrations	2019/6/27 15:32	文件夹	
templates	2019/6/27 15:32	文件夹	
__init__	2019/6/27 15:32	Python File	0 KB
admin	2019/6/27 15:32	Python File	1 KB
apps	2019/6/27 15:32	Python File	1 KB
models	2019/6/27 15:32	Python File	1 KB
tests	2019/6/27 15:32	Python File	5 KB
urls	2019/6/27 15:32	Python File	1 KB
views	2019/6/27 15:32	Python File	4 KB

ORM

ORM是“对象-关系-映射”的简称，主要任务是：

- 根据对象的类型生成表结构
- 将对象、列表的操作，转换为sql语句
- 将sql查询到的结果转换为对象、列表



数据库设计

user表

属性	数据类型	是否为主键
用户ID	bigint,key	是
邮箱	varchar,key	是
手机	varchar,key	是
学号	varchar,key	是
姓名	varchar	否
年龄	int	否
性别	varchar	否
年级	varchar	否
专业信息	varchar	否
昵称	varchar,key	是

属性	数据类型	是否为主键
学生头像	blob	否
身份	varchar	否

问卷表

问卷表保存问卷的相关信息， 题目是通过题目表的题目的问卷编号属性相同来关联的

属性	数据类型	是否为主键
问卷ID	bigint	是
问卷名称	varchar	否
问卷描述	varchar	否
问卷类型	varchar	否
问卷奖励	varchar	否
问卷发布者	bigint	否
问卷状态	varchar	否
问卷截止时间	varchar	否

题目表

保存所有题目的基本信息， 不同的问卷编号表示在不同的问卷中

属性	数据类型	是否为主键
题目ID	bigint,key	是
题目标题	varchar	否
题目类型	varchar	否
题目信息	varchar	否
是否必答	bool	否
问卷编号	bigint	否

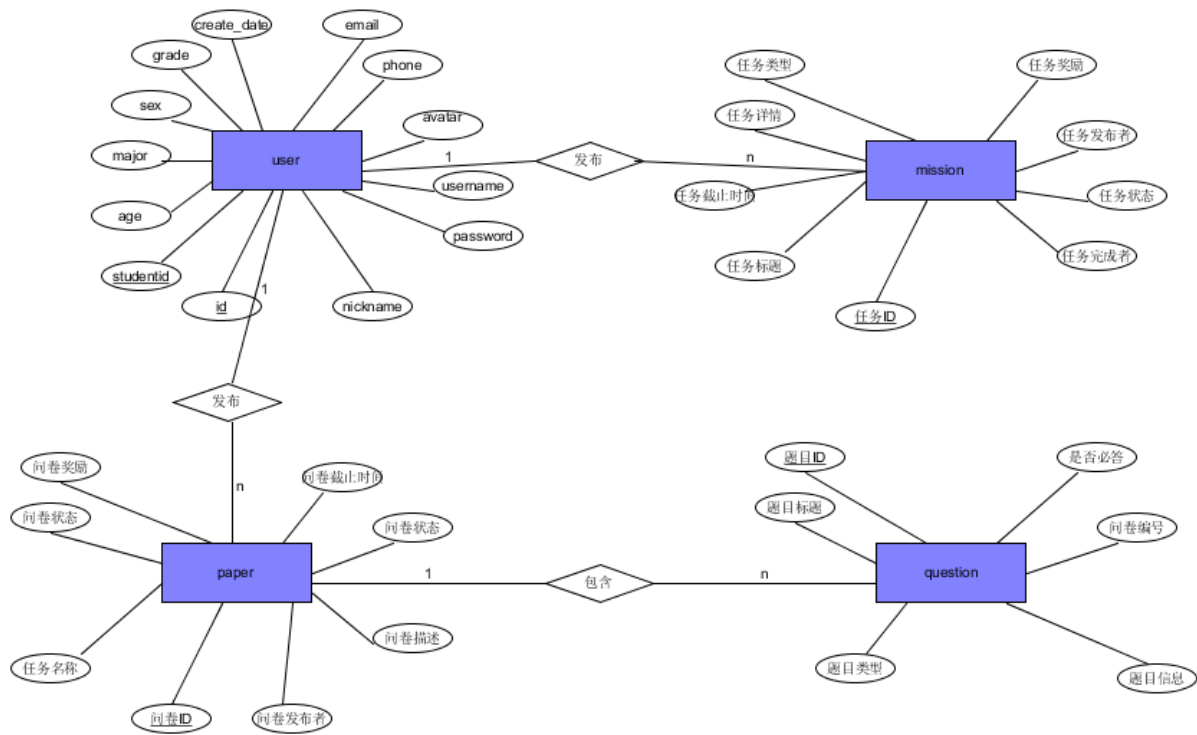
任务表

属性	数据类型	是否为主键
任务ID	bigint,key	是
任务标题	varchar	否
任务详情	varchar	否

属性	数据类型	是否为主键
任务类型	varchar	否
任务奖励	varchar	否
任务发布者	bigint	否
任务状态	varchar	否
任务完成者	bigint	否
任务截止时间	varchar	否

ER图

ER模型中包含3钟相互关联的信息：数据对象、数据对象的属性及数据对象彼此之间相互连接的关系。我们一共有4个实体题目之间的关系如图。



接口API设计规范

1、协议

使用https协议

2、域名

应该尽量将API部署在专用域名之下。

<https://api.bangbangbao.com>

3、路径

由于REST API是面向资源的，所以路径中只能出现名词，不能出现动词，所用名词尽量参考数据库表的设计,例如：<https://bangbangbao.com/user/{username}>

4、http请求方式

1. GET（SELECT）：从服务器取出资源（一项或多项）。
2. POST（CREATE）：在服务器新建一个资源。
3. PUT（UPDATE）：在服务器更新资源（客户端提供改变后的完整资源）。

使用例子：

GET /paper: 列出所有问卷

POST /paper: 创建问卷

GET /user /{username}: 获取某个指定电影的信息

5、过滤信息

如果对资源的需求不是全部，那么需要提供过滤的参数，例如：

<https://bangbangbao.com/user/name=abc> 返回名字为abc的用户

6、数据

使用json数据格式进行数据传递。

7、状态码

200 OK - [GET]: 服务器成功返回用户请求的数据，该操作是幂等的（Idempotent）。

201 CREATED - [POST/PUT/PATCH]: 用户新建或修改数据成功。

202 Accepted - [*]: 表示一个请求已经进入后台排队（异步任务）

204 NO CONTENT - [DELETE]: 用户删除数据成功。

400 INVALID REQUEST - [POST/PUT/PATCH]: 用户发出的请求有错误，服务器没有进行新建或修改数据的操作，该操作是幂等的。

401 Unauthorized - [*]: 表示用户没有权限（令牌、用户名、密码错误）。

403 Forbidden - [*] 表示用户得到授权（与401错误相对），但是访问是被禁止的。

404 NOT FOUND - [*]: 用户发出的请求针对的是不存在的记录，服务器没有进行操作，该操作是幂等的。

406 Not Acceptable - [GET]: 用户请求的格式不可得（比如用户请求JSON格式，但是只有XML格式）。

410 Gone -[GET]: 用户请求的资源被永久删除，且不会再得到的。

422 Unprocesable entity - [POST/PUT/PATCH] 当创建一个对象时，发生一个验证错误。

500 INTERNAL SERVER ERROR - [*]: 服务器发生错误，用户将无法判断发出的请求是否成功。

8、错误处理

如果状态码是4xx，就应该向用户返回出错信息。一般来说，返回的信息中将error作为键名，出错信息作为键值即可。使用详细的错误包装错误：

```
{
  "errors": [
    {
      "userMessage": "Sorry, the requested resource does not exist",
      "internalMessage": "No car found in the database",
      "code": 34,
      "more info": "http://dev.mwaysolutions.com/blog/api/v1/errors/12345"
    }
  ]
}
```

5. 架构设计

5.1 前端架构设计

技术栈

vuejs + webpack + ant-design

开发环境

nodejs 8.12.0 + vue-cli 2.9.6

5.2 后端架构设计

开发环境 mac os + sublime

技术栈 django 2.1.7 + SQLite + apache

部署环境 Ubuntu 18.04.2 LTS

5.3 架构问题

可靠性和可恢复性

- 因素：web应用使用过程中出现访问服务端中断问题并进行恢复
- 度量和质量场景：当web应用访问服务端中断时，在正常的工作环境下，如果在5s内检测到其恢复，则重新建立连接，否则提示用户无法连接
- 可变性：在能够重新建立连接之前，可以在客户端进行一些操作，存在在web缓存中
- 该因素对涉众、架构以及其他因素的影响：web用户提交的数据可能会发生丢失，无法传递给服务端
- 对于成功的优先级：高

- 困难或风险：中等

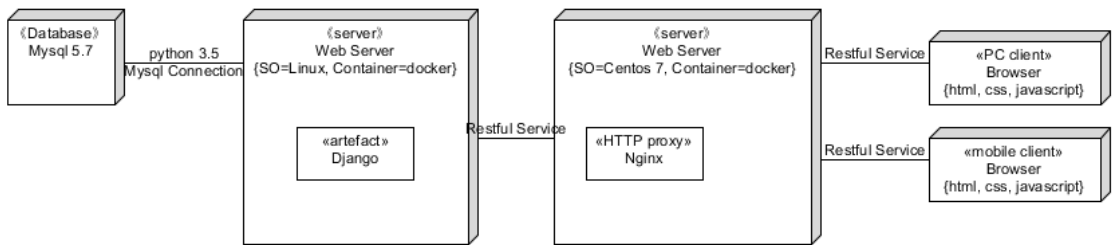
5.4 解决方案说明

可靠性和可恢复性解决方案

当出现访问服务端中断问题时，

- 首先进行重新连接，
- 如果失败则进入离线模式， 其中和服务器的发送接收操作不可用，
- 但是可以进行问卷创建等操作， 这些离线模式下的数据将存储在浏览器缓存中， 不会

5.5 物理视图



5.6 逻辑视图

