Biopedia

Online Bioinformatics Data Center

Developed By **Nerdy Buddy**

Biopedia - 设计文档

Nerdy Buddy

组长：宋佳铭

组员：袁源，肖迪，杨煜，滕爽

2014 年 12 月 3 日

目录

1 引言

1.1 编写目的

这个文档的适用对象为本网站及数据库的用户和该项目的开发者。

对于用户来说，他们可以通过这个文档了解这个网站的使用方法，从而方便地使用到他们需要的功能。

对于开发者来说，本文档能让他们对整个项目有一个大体了解，从而在对项目存在疑惑的时候，参考该文档，解决问题，继续开发。

1.2 背景

本设计文档针对生物信息宏基因组数据展示平台项目，这个项目的具体说明详见需求文档，所有的设计都基于需求文档完成，并在此基础上根据迭代需求进行一定的细微调整。

2 程序系统的体系结构

2.1 UML序列图

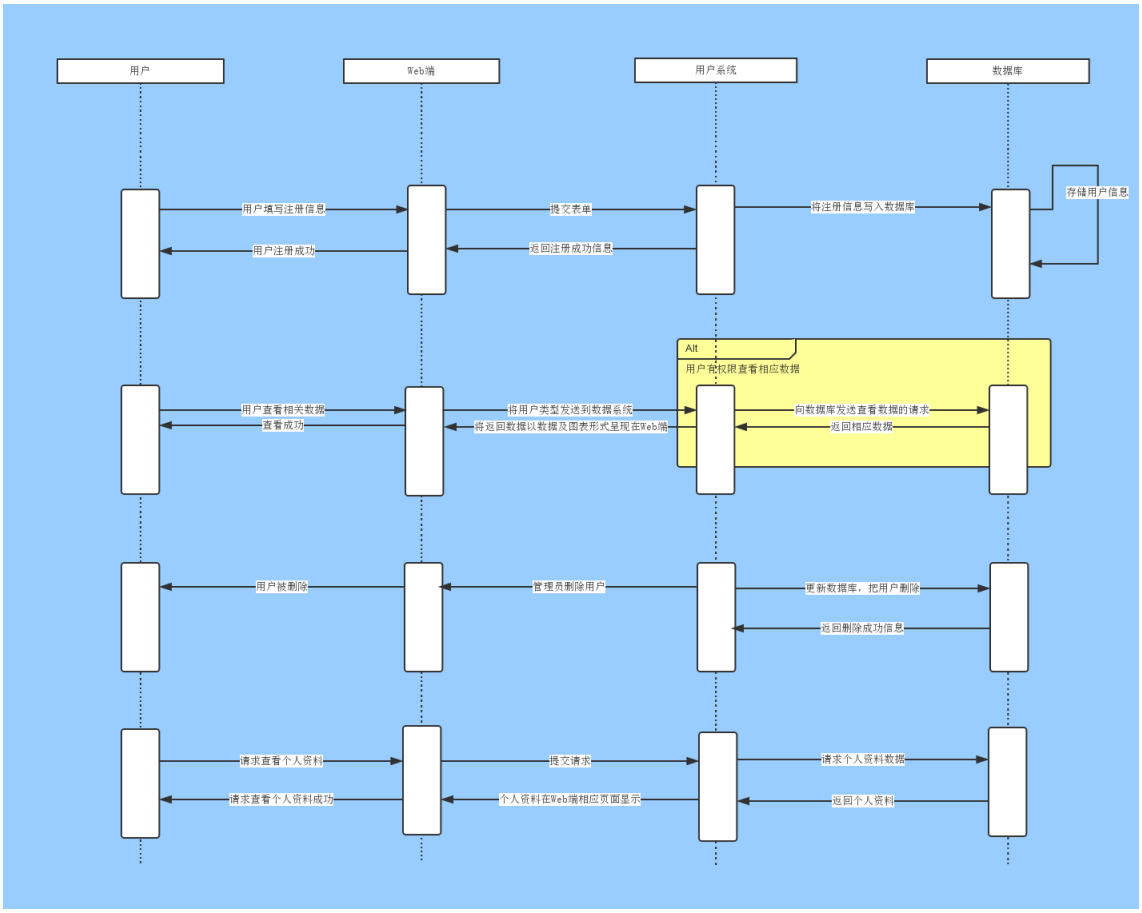

图 1: UML序列图

2.2 基本框架

本次开发我们使用么MVC框架，即Model View Controller，模型(model)－视图(view)－控制器(controller)框架，使用Python语言进行开发，使用Flask Web应用框架。

视图为Web端，用户通过Web端与后端进行交互。Web实现部分使用Jinja2框架。整体网站风格使用Google Material Design作为模板实现，同时使用Bootstrap包。画图部分使用Charts.js、Highcharts、d3.实现。

控制器为中间层，连接前端和后端。

模型为后端，数据库部分采用MongoDB搭建。

3 基本设计概述

3.1 主页面

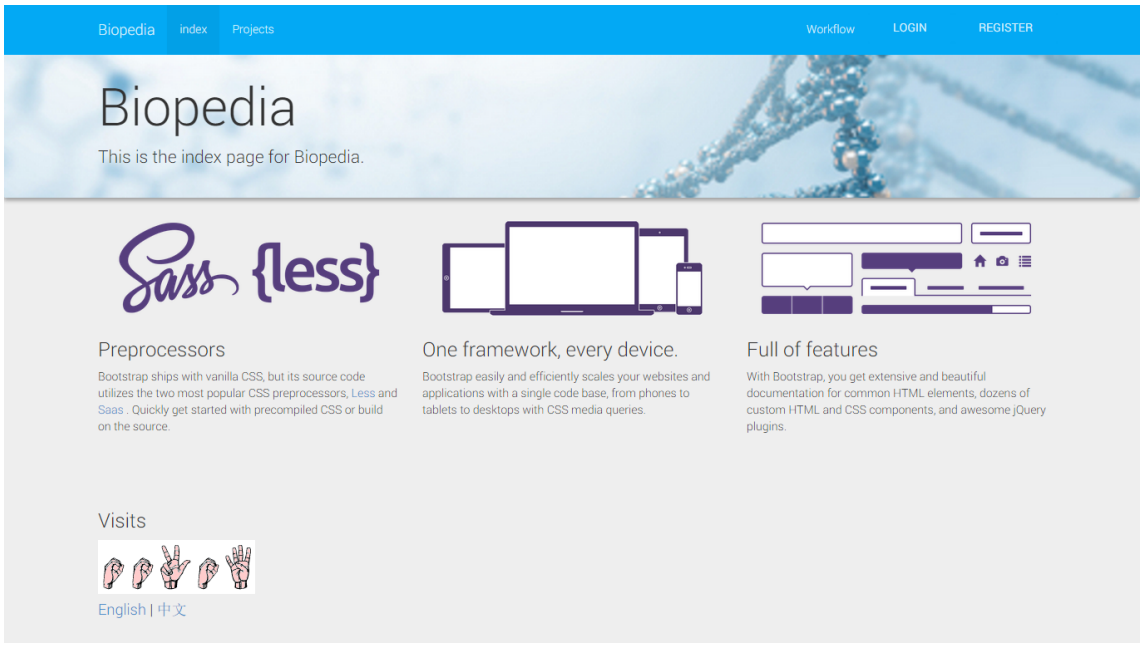

图 2: 首页

3.2 用户注册流程

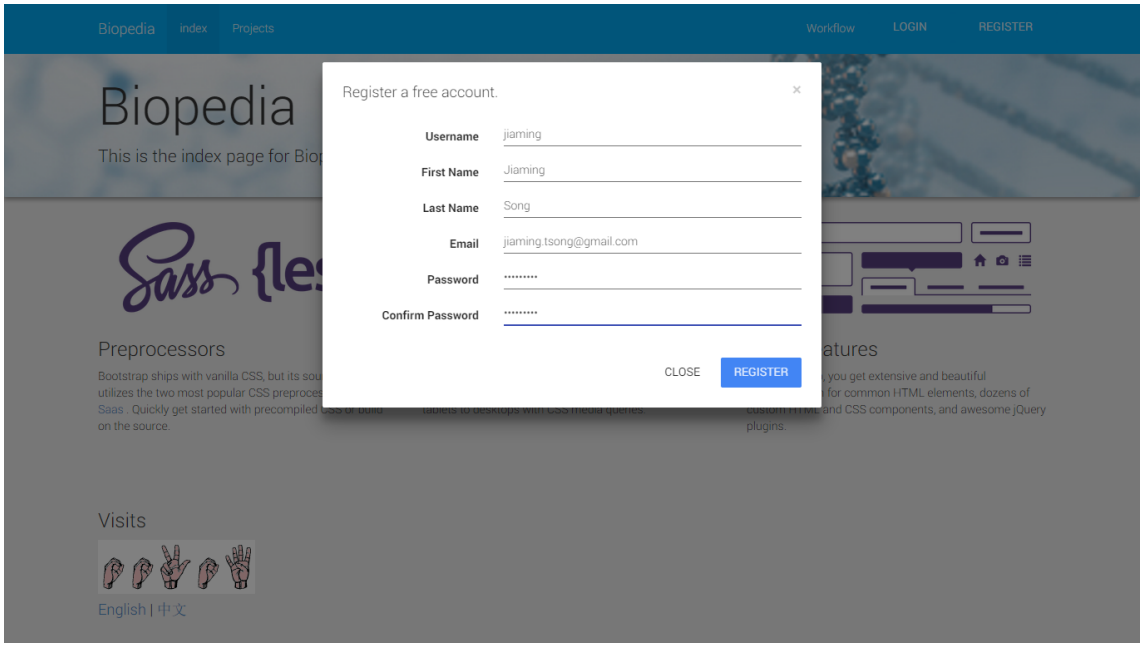

图 3: 注册

3.3 用户登录流程

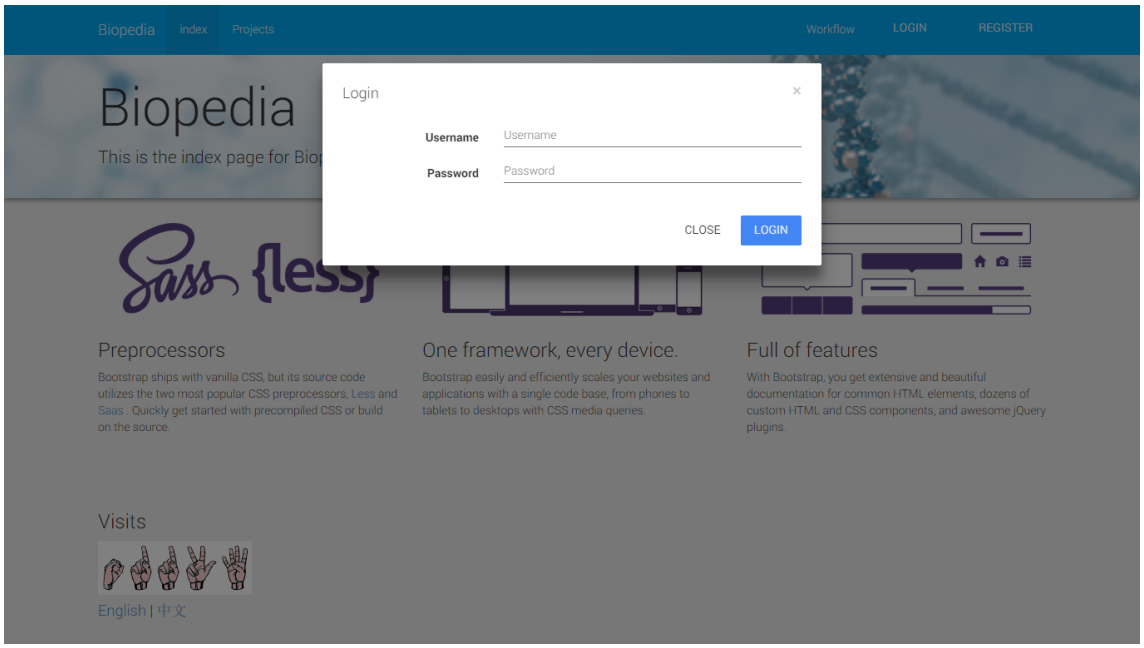

图 4: 登录

- 用户通过点击登录按钮，在弹出的登录界面输入已注册的用户名和密码，进行登录。

- 登录成功后可在右上角看到用户名
- 登录失败将在登录页面显示错误信息。

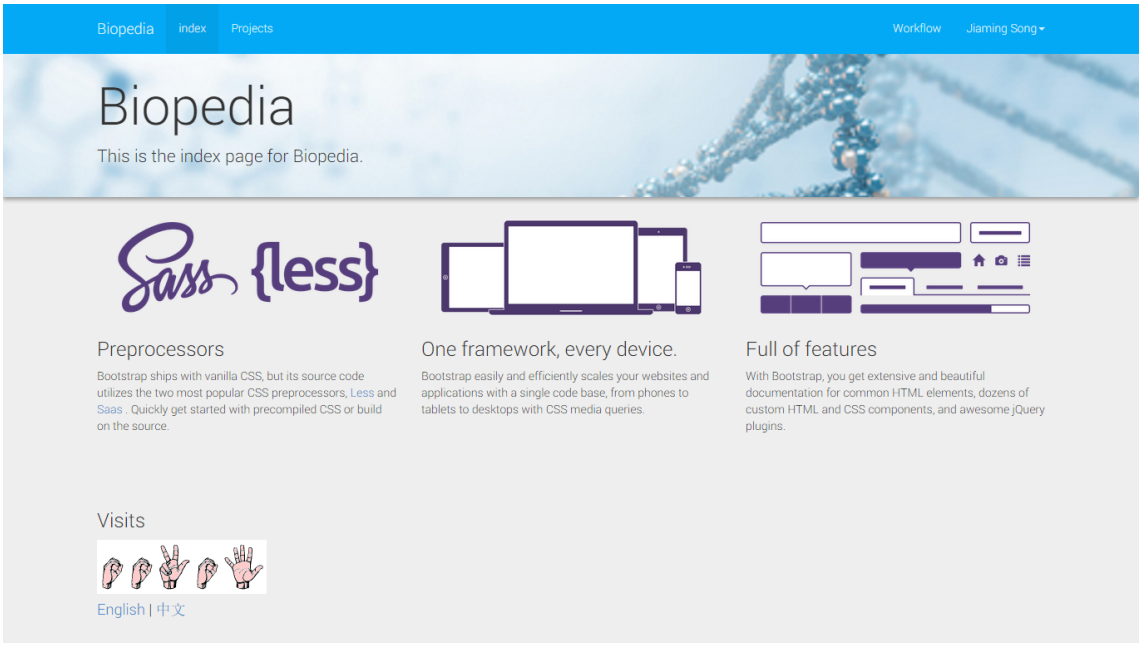

图 5: 用户已登录界面

3.4 用户登出流程

- 用户通过点击右上角的用户名，弹出下拉菜单。
- 用户通过点击“Logout”按钮，进行登出操作。

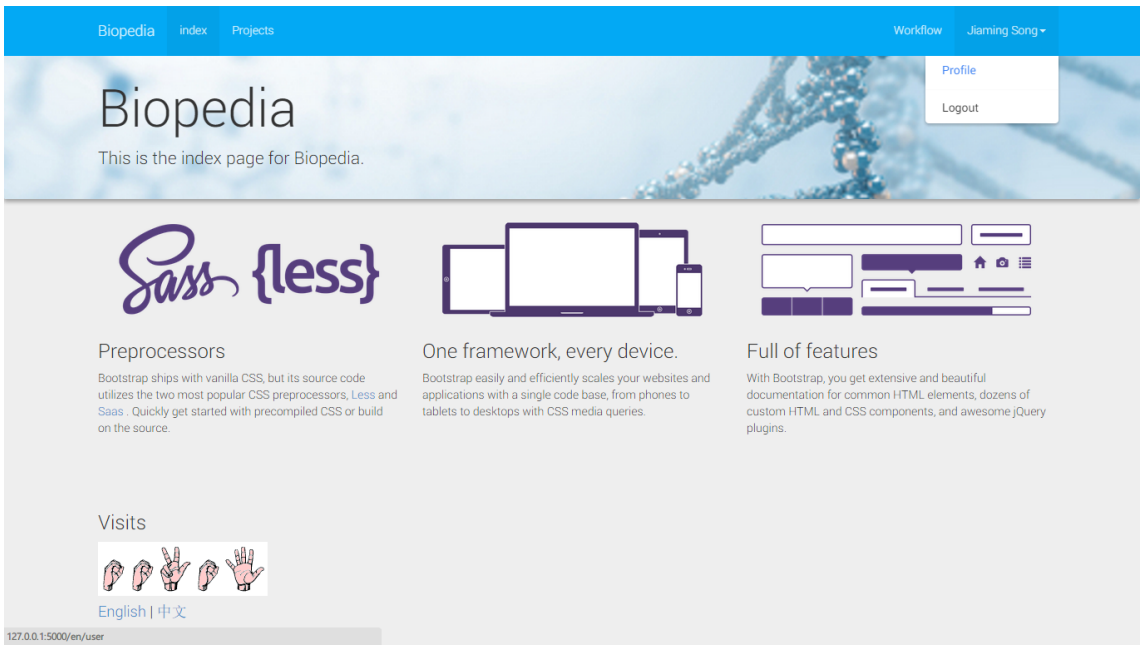

图 6: 点击profile

3.5 查看个人资料页面

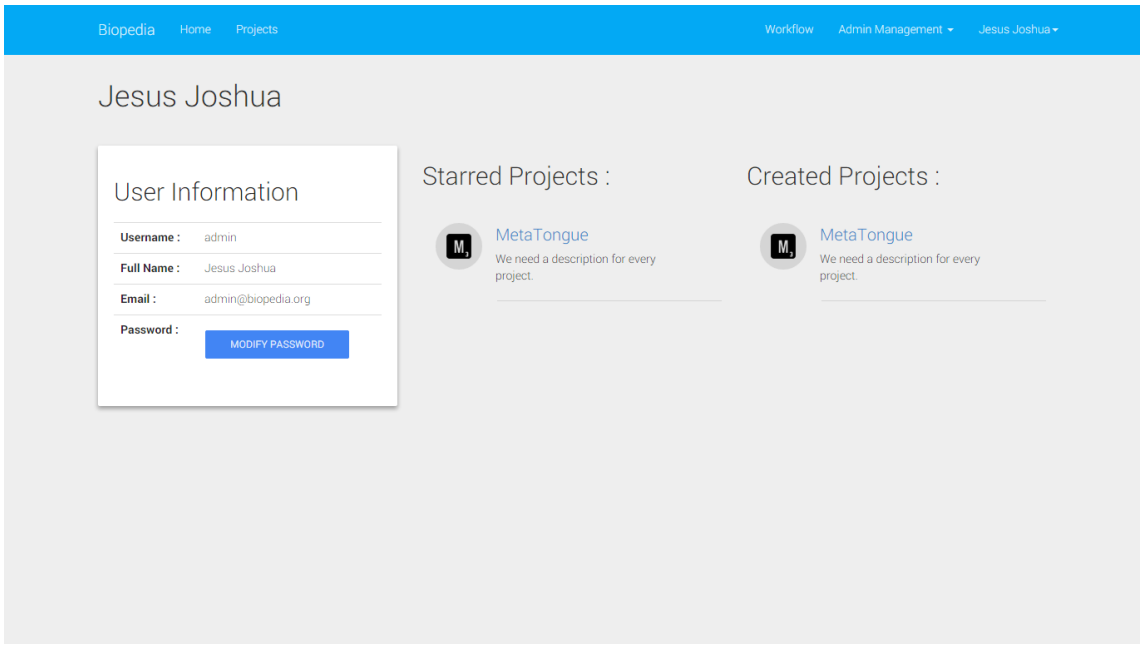

图 7: 用户个人界面

- 在用户成功登陆后本站的各个页面，点击右上角用户名。

- 在下拉菜单中点击“Profile”，进入查看个人资料页面。 用户在该页面能进行更改密码等操作。
- 除此之外，用户还可以在这里看到该用户的“Starred Project”以及“Created Projects”等信息。

3.6 修改密码页面

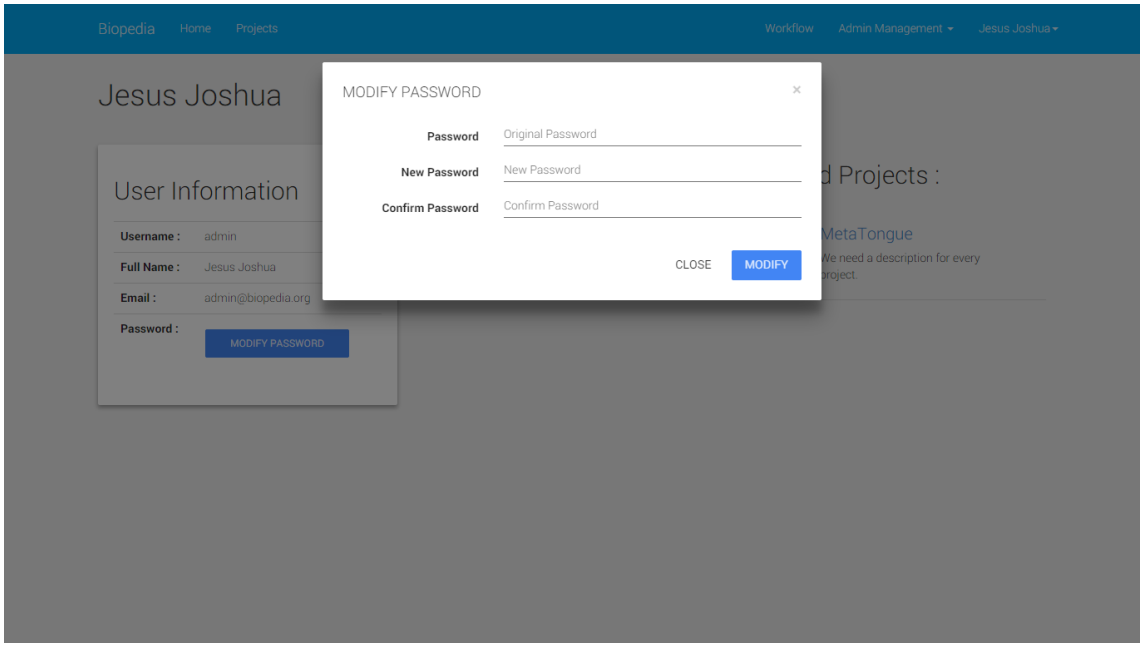

图 8: 修改密码

- 用户在查看个人资料页面，通过点击“MODIFY PASSWORD”按钮进行密码修改。
- 输入旧密码、新密码，并确认一次新密码。
- 在确认旧密码正确，新密码和第二次输入的确认密码一致后修改密码成功

3.7 查看项目页面

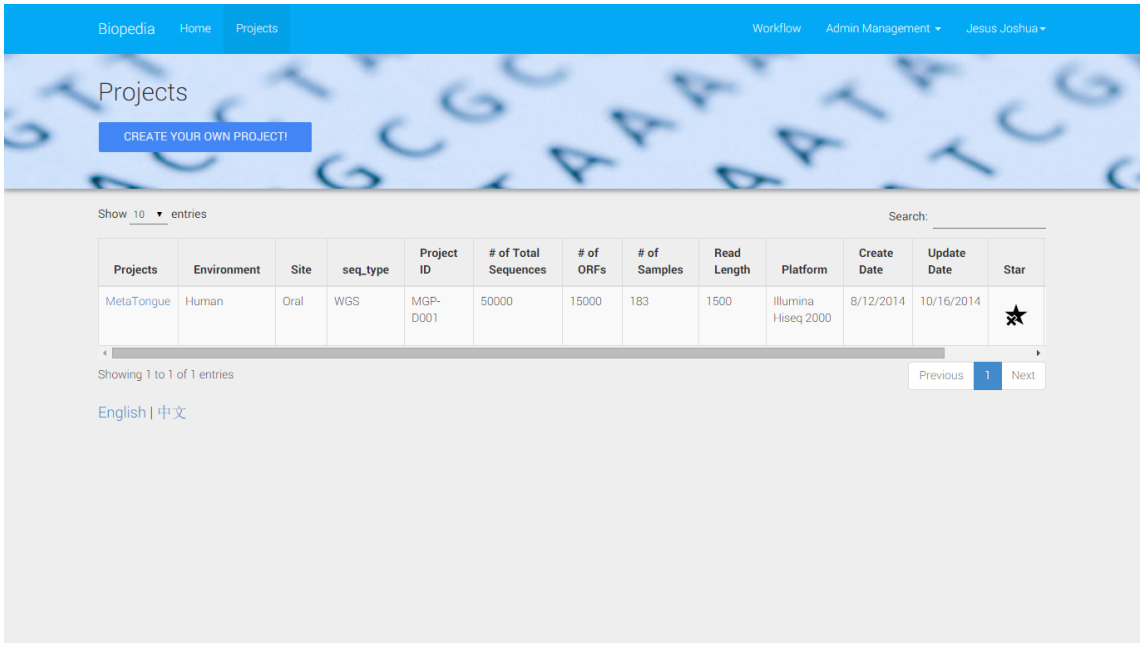

图 9: 项目界面

- 用户通过在主页面点击导航栏左上角的“项目”按钮，进入查看项目页面。
- 用户通过点击“Create your own project”按钮创建自己的项目。
- 用户可以在Show entries处通过下拉菜单选择每页显示的项目数量。
- 用户可以在Search处输入要查找的内容，找到相应的数据。

3.8 查看项目样本页面

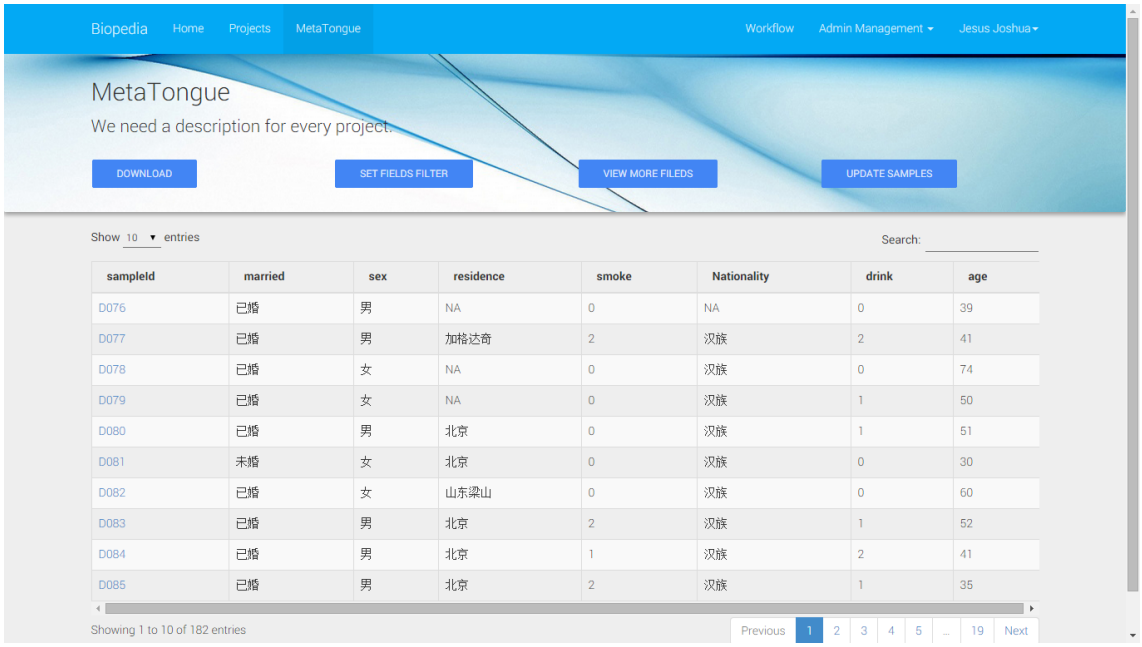

图 10: 样本

- 用户在项目中点击项目名字，进入项目样本查看页面。
- 用户通过点击“Download”按钮可以进行数据下载。
- 用户通过点击“SET FIELDS FILTER”按钮可以设置筛选选项。
- 用户通过点击“VIEW MORE FILEDS”按钮可以查看更多筛选选项。
- 用户通过点击“UPDATE SAMPLES”进行样本的更新。
- 用户可以在Show entries处通过下拉菜单选择每页显示的样本数量。
- 用户可以在Search处输入要查找的内容，找到相应的数据。

3.9 查看数据图表页面

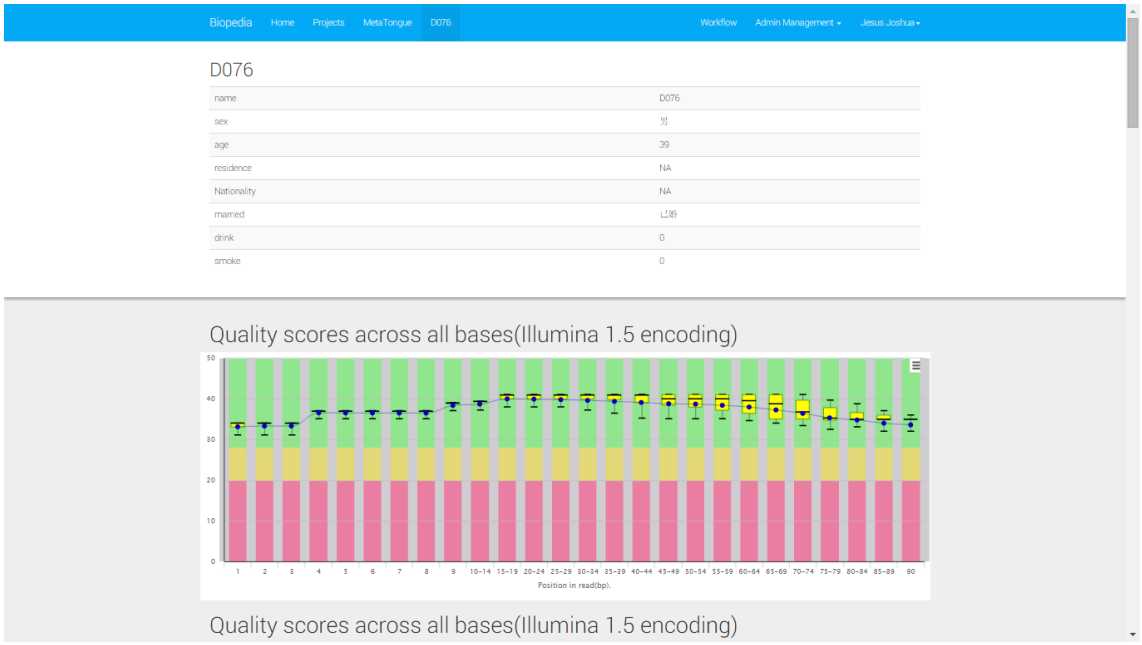

图 11: 首页

用户通过点击项目中的样本，查看数据图表。

4 模块之间的接口设计

4.1 前端渲染接口

4.1.1 Index

index.html是项目的首页，展示项目的一些基本信息。

这个页面需要两个接口参数（见表1）：

名称	类型	含义
language	enum	选择中文('cn')或者英文('en')
count	int	显示计数器访问数量

表 1: index.html 接口参数

在jinja2语法中，我们可以通过{{ language }}来调用language的值。下面就是一个支持中英文切换的宏：

Listing 1: 支持中英文切换的宏(使用jinja2语法)

```
{% macro switch_lang(en, cn) %}
    {% if language ==’en’ %}
        {{ en }}
    {% else %}
        {{ cn }}
    {% endif %}
{% endmacro %}
```

借助language参数和宏的特性，我们将中英文的切换变成一句话switch_lang('English', '中文')，大大增加了我们的工作效率。

4.1.2 Login & Register

登录和注册的功能，我们放入了include/login_register_modals.html的文件中，在每一个模板中，我们通过引用这个文件来完成登录的渲染，大大减小了我们的工作量。同时，我们在这里会判断用户是否已经登录，这一功能通过传递user这一个值来完成，其中user包含以下几项（见表3.1.2）：

名称	类型	含义
username	string	用户名
email	string	邮箱
firstname	string	名
lastname	string	姓
admin	bool	是否为管理员

表 2: login_register_modals.html接口

出于安全考虑，我们不会传递密码信息。这样，我们不需要修改前端，只需要在后端增加一个参数user，就可以让前端渲染出用户希望看到的界面。

4.1.3 Projects

projects.html 包含了所有和项目相关的代码。我们通过调用后端的projects_backend对其进行渲染，用到的接口如下（见表3）：

名称	类型	含义
language	enum	选择中文（'cn'）或者英文（'en'）
projects_list	list	包含所有项目信息，其中每个元素是一个dict，包含项目信息，用来渲染表格

表 3: projects.html接口

projects_list包含所有项目信息，其中的每一个元素project包含以下几项(见表4)：

名称	类型	含义
project_name	string	项目名称
project_id	string	项目ID
environment	enum	环境
site	enum	地点
num_of_sequences	int	序列数量
num_of_orfs	int	ORF数量
num_of_samples	int	样本数量
sequence_type	enum	序列种类
platform	enum	平台
read_length	int	舌苔长度
create_date	date	创建日期
update_date	date	更新日期
star	bool	如果有user, 判断是否关注
delete	bool	如果有user, 判断是否有删除权限

表 4: projects.html中每一个项目信息的接口

4.1.4 Samples

samples.html 包含了所有和观看项目相关的代码，我们通过调用samples_backend对其进行渲染，用到的接口如下（见表5）：

名称	类型	含义
language	enum	选择中文（'cn'）或者英文（'en'）
project_name	string	项目名称
sample_list	list	样本列表
project_fields_name	list	显示的项目域名称
all_fields_name	list	全部项目域名称
fields_string_type		
string_field_element		
mapping	dict	项目映射文件的
created_project	bool	如果有user, 判断是否有更新权限

表 5: samples.html接口

其中sample_list包括样本的列表，其中每一个元素sample包含下列接口（见表6）

名称	类型	含义
project_name	string	项目名称
{ field }	-	和项目相关，可能有多个值；对于特定的project，其值不同

表 6: samples_list中samples的接口定义

这些接口和我们需要渲染项目页时的动作，以及用户的需求完全吻合：

1. 显示给用户需要显示的项目
2. 给用户选择项目显示的选项
3. 控制用户权限，如果用户没有权限，不显示相应的按钮
4. 给用户选择中文/英文显示的权利

4.1.5 Profile

profile.html包含了所有和观看一个样本相关的代码，我们通过调用projects.profile_backend对其进行渲染，用到的接口如下（见表7）：

名称	类型	含义
language	enum	
sample_name	string	样本名称
project_name	string	项目名称
sample_detail	dict	项目的细节
selected_details_name	list	显示的项目细节

表 7: profile.html渲染接口

其中sample_detail和selected_details_name是用来显示样本基本信息的字典，用户可以通过sample_detail['field']来查看项目的'field'域的值。

4.1.6 User

user.html包含了用户基本信息的代码，通过调用user.profile_backend来进行渲染，用到的接口如下（见表8）：

名称	类型	含义
language	enum	
user	User	显示在导航栏上的用户信息
disp_user	User	显示在页面中间的用户信息，主要是在管理员模式下使用
starred_projects	list	user关注的项目
created_projects	list	user创建的项目

表 8: user.html渲染接口

其中User信息和2)中一样，而starred_projects和created_projects和3)中一样。

4.1.7 User Management

user-admin.html 包含了管理员管理用户的代码，通过调用user_admin_backend来进行渲染。用到的接口有：

名称	类型	含义
language	enum	
user	User	显示在导航栏上的用户信息
disp_user	User	显示在页面中间的用户信息，主要是在管理员模式下使用

表 9: user-admin.html接口

其中users_list包含User的username, email, firstname, lastname, 和admin五个域，如果一个user是admin，那么他不可以被删除，否则他能够被删除。

4.1.8 Workflow

workflow.html包含了Workflow的代码，由于该模块根据用户的需求比较独立，我们没有加入过多的渲染接口。

4.2 服务器接口

我们将我们的Flask架构分拆为几个部分，这些部分在Flask的语言中叫蓝图（Blueprint），意义和Django的app比较类似。我们的蓝图架构如下面的ER图（图12）所示：

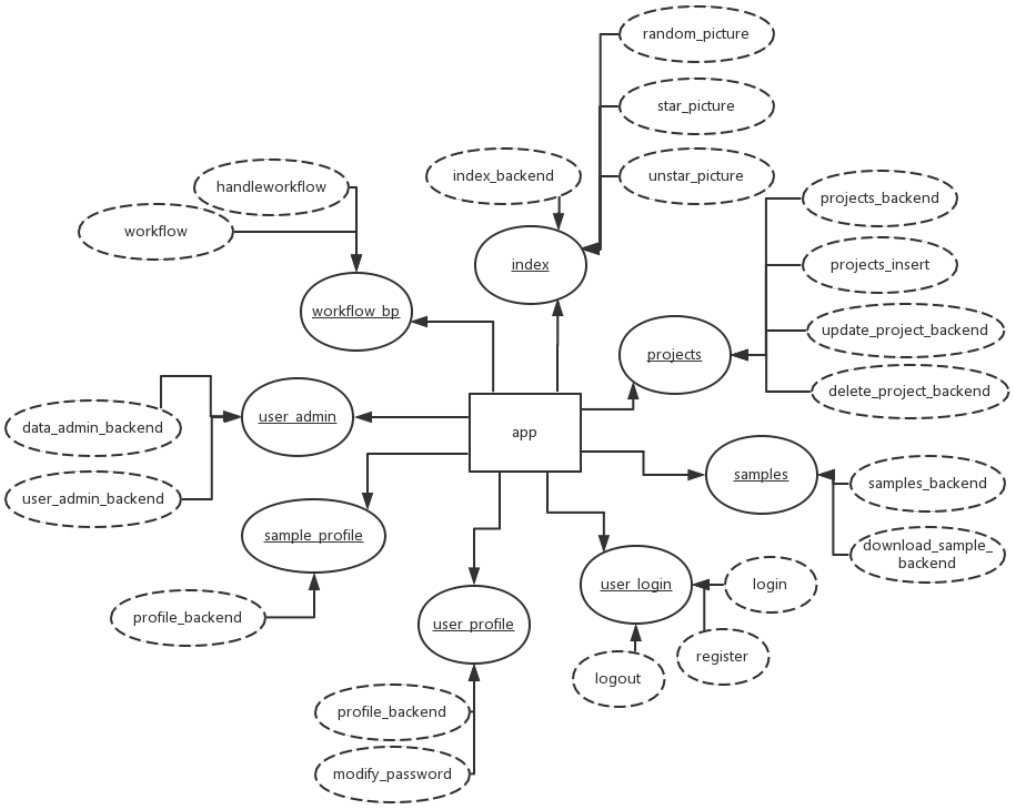

图 12: 数据库ER图

其中方形为Flask的应用核心（app），圆形为蓝图，虚线圆形为和url对应的函数方法。

函数方法和对应的url关系如表10所示：

蓝图	函数名	参数	URL	含义
index	index_backend	language	/index /<language>/index	首页
	random_picture	-	/random-picture	根据给定的英文名称，显示对应的大写字符图片。
	star_picture	-	/star-picture	显示带勾的星形
	unstar_picture	-	/unstar-picture	显示带叉的星形
projects	projects_backend	language	/projects /<language>/projects	显示项目
	projects_insert	language	/projects_insert /<language>/projects_insert	插入项目
	update_project_backend	language	/update-project /<language>/update-project	更新项目数据
	delete_project_backend	language	/delete-project /<language>/delete-project	删除项目
samples	samples_backend	language	/samples /<language>/samples	显示项目中的样本
	download_sample_backend	-	/download-sample	下载样本
user_login	login	language	/login /<language>/login	登录
	register	language	/register /<language>/register	注册
	logout	language	/logout /<language>/logout	登出
user_profile	profile_backend	language	/user /<language>/user	用户个人界面
	star_backend	-	/star	点关注
	modify_password	language	/modify-password /<language>/modify-password	修改密码
sample_profile	profile_backend	language	/profile /<language>/profile	单个样本详细信息
user_admin	user_admin_backend	language	/user-admin /<language>/user-admin	管理员用户管理页面
	delete_user	-	/delete-user	删除一个用户
	data_admin_backend	language	/data-admin	管理员项目管理页面

表 10: 函数方法和url对应关系

如果将Blueprint看做类，Flask App看做package，那么对应的包图如图13所示：

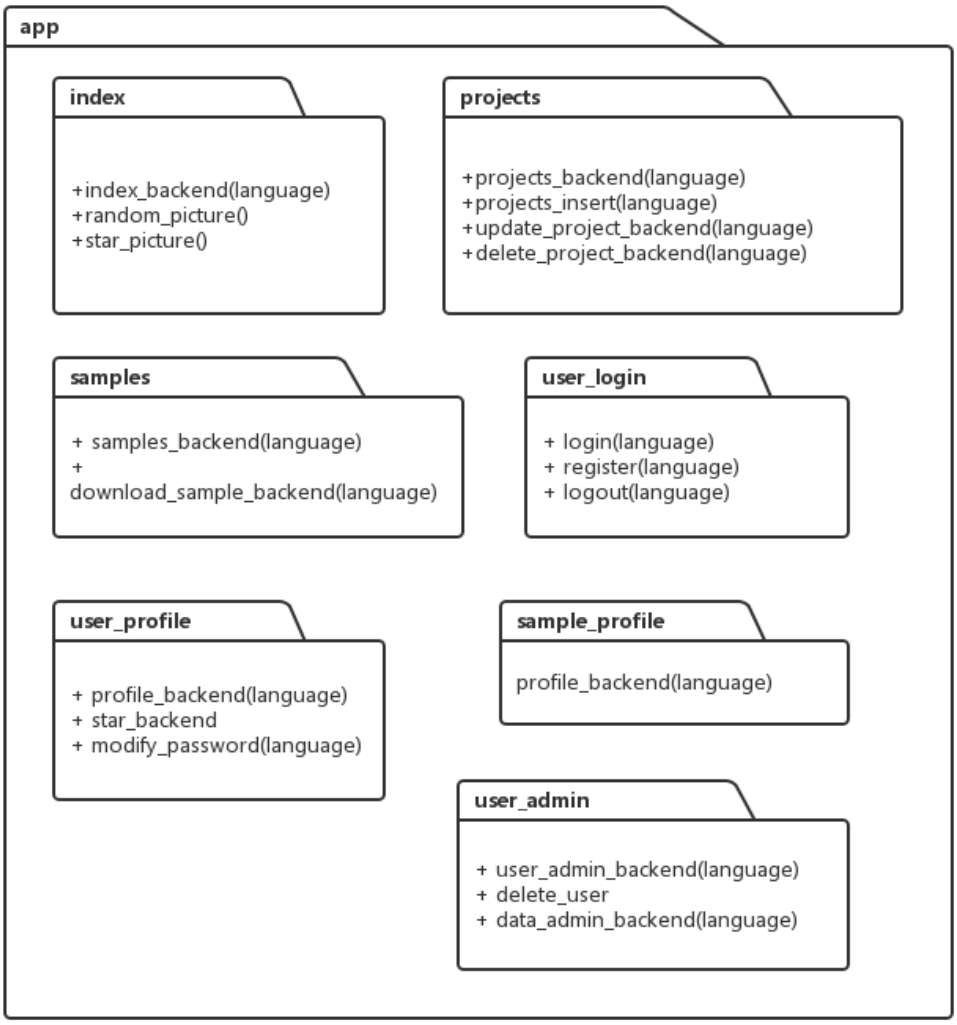

图 13: 数据库包图

作为一款Web应用，不仅需要输入函数参数作为指令，也需要依靠URL指令和Session的支持获取一些参数，这些参数包括：

URL模板参数 在定义URL的时候自动定义为参数，这种参数为函数必须存在的参数，可以通过设定默认值来缩短长度；

URL请求参数 在URL后以'?arg=value'的形式出现，这种参数会显示在URL上，但并非必需；

Session 不随URL出现，经常用于控制用户登录管理；

表单提交 用于在POST方法中提交表单信息。

下面几个部分会详细介绍参数的使用。

4.2.1 index

index包括首页和一些基本应用的URL，主要提供主页，以及为其他蓝图跳转提供支持。

函数名称	参数名	参数类型	参数含义
index_backend	language	URL模板参数	切换语言
	username	Session	如果存在，为已经登录的用户名；否则不存在
	alert_message	Session	警告信息，用来处理注册/登录不正确的警告
	alert_place	Session	警告位置，用来分别显示警告的位置（注册/登录）
random_picture	无		
star_picture	无		
unstar_picture	无		

表 11: index蓝图的url接口

4.2.2 projects

projects包括项目显示和处理的URL，主要提供项目显示，增加删除的功能。

关于详细的projects接口，请参考表12。

函数名称	参数名	参数类型	参数含义
projects_backend	language	URL模板参数	切换语言
	username	Session	如果存在，为已经登录的用户名；否则不存在
projects_insert	language	URL模板参数	切换语言
	name	表单提交	输入姓名
	environment		输入环境
	site		输入地址
	sequence_type	表单提交	输入序列种类
	project_id	表单提交	输入项目ID
	num_of_total_sequences	表单提交	输入序列总数
	num_of_orfs	表单提交	输入ORF总数
	num_of_samples	表单提交	输入样本总数
	read_length	表单提交	输入舌苔长度
	platform	表单提交	输入平台
	create_date	表单提交	输入创建日期
	update_date	表单提交	输入更新日期
	mapping	表单提交	输入中英文对照文件
	samples	表单提交	输入样本文件
update_project_backend	language	URL模板参数	切换语言
	update	表单提交	上传的样本文件
	project_name	表单提交	上传的项目名称
	username	Session	用户名
delete_project_backend	language	URL模板参数	切换语言
	username	Session	用户名
	projectname	URL请求参数	要删除的项目名称

表 12: projects蓝图的url接口

4.2.3 samples

samples包括样本的显示，下载，更新等功能，见表13

函数名称	参数名	参数类型	参数含义
samples_backend	language	URL参数模板	切换语言
	name	URL请求参数	项目名称
	fields	URL请求参数（长度不定）	选择的键值类型
	username	Session	用户名
download_sample_backend	project_name	URL请求参数	要下载的项目名称

表 13: samples蓝图的api接口

4.2.4 user_login

user_login包含用户的登录，注册，登出等功能，见表14

函数名称	参数名	参数类型	参数含义
login	username	表单提交	登录所需用户名
	password	表单提交	登录所需密码
register	username	表单提交	注册所需用户名
	password	表单提交	注册所需密码
	email	表单提交	注册所需邮箱
	firstname	表单提交	注册所需名
	lastname	表单提交	注册所需姓
logout	username	Session	登出的用户名

表 14: user_login蓝图的api接口

4.2.5 user_profile

user_profile包含用户的个人信息显示等功能，见表15

函数名称	参数名	参数类型	参数含义
profile_backend	language	URL模板参数	切换语言
	username	Session	用户名
star_backend	username	Session	用户名
	project_name	URL请求参数	关注/取关的项目名
modify_password	username	Session	用户名
	password	表单提交	旧密码
	newpassword	表单提交	新密码

表 15: user_profile蓝图的api接口

4.2.6 sample_profile

sample_profile包含样本的信息显示功能，见表16

函数名称	参数名	参数类型	参数含义
profile_backend	language	URL模板参数	切换语言
	project	URL请求参数	项目名
	name	URL请求参数	样本名

表 16: sample_profile蓝图的api接口

4.2.7 user_admin

user_admin包含和管理员操作相关的功能，例如用户管理，数据库管理，用户删除，项目删除等。

函数名称	参数名	参数类型	参数含义
user_admin_backend	username	Session	用户名，只有是管理员方能有效
	language	URL模板参数	切换语言
delete_user	username	Session	用户名，只有是管理员方能有效
	username	URL请求参数	被删除的用户名

表 17: user_admin_backend蓝图的api接口

4.3 数据库接口

我们采用MongoDB作为我们的数据库，因此对于一般的操作，就是使用MongoDB的接口进行操作，我们使用的具体框架是Flask-PyMongo，MongoEngine和MongoDB原生接口，如图14所示。

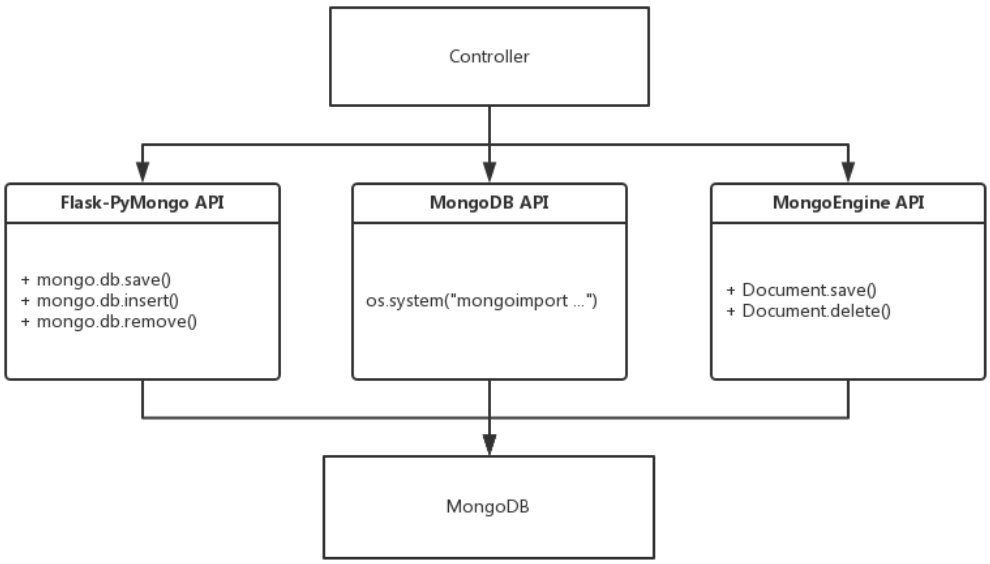

图 14: 数据库关系图

例如在保存操作中，我们利用这两种框架提供了以下几种接口，见表18:

接口	类型	操作
mongo.db.save(info)	Flask-PyMongo	将info存入MongoDB中
os.system("mongoimport ...")	MongoDB接口	将文件导入MongoDB中
Document.save()	MongoEngine	将MongoEngine的Document存入MongoDB中

表 18: 数据库接口

4.4 用户接口

用户接口是通过html模板上的按钮来实现的，它可以像服务器端发送GET或者POST请求，使得用户可以很方便地对网站进行导航，对数据库进行操作。下面我们以nav_bar_right.html提供的接口为例，用图示的形式展示用户接口（见图15）。

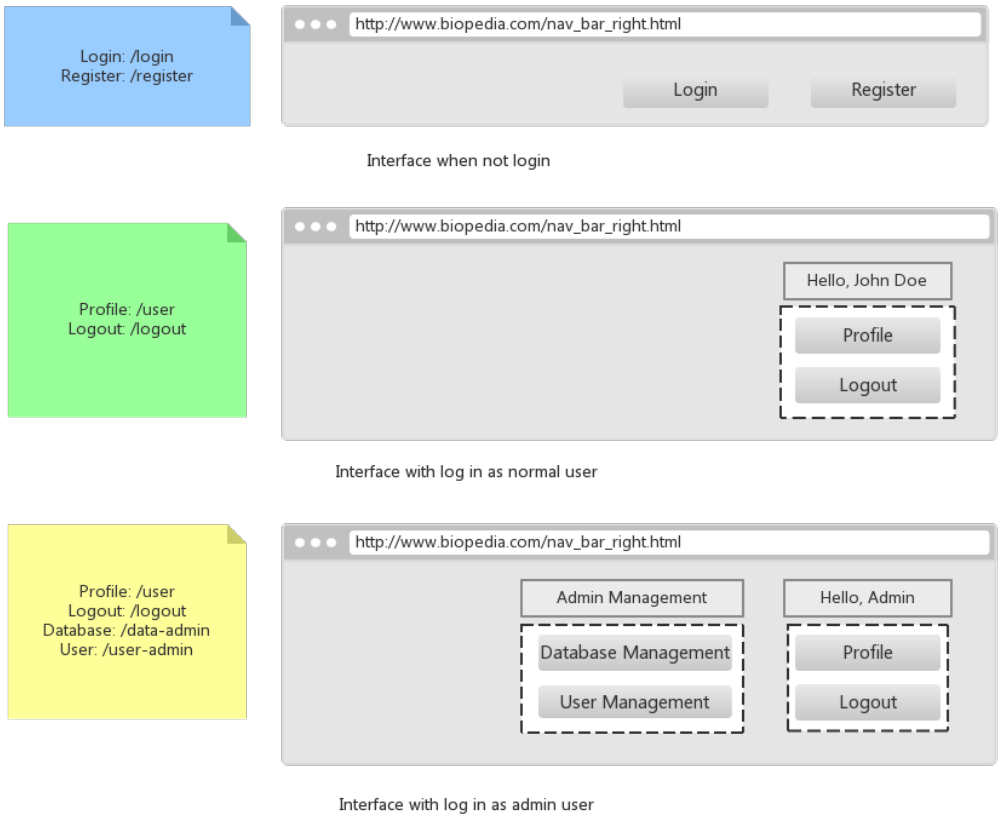

图 15: 右侧导航栏界面演示

nav_bar_right为导航栏右侧的接口，分为以下3种情况，显示3个不同的界面：

1. 用户未登录，可以点击的按钮包括登录和注册，弹出对话框后，分别对应/login和/register
2. 用户已登录，权限为普通用户，可以点击的按钮包括资料和登出，分别对应/user和/logout
3. 用户已登录，权限为管理员用户，可以点击的按钮包括普通用户的按钮，还有数据库管理以及用户管理，分别对应/data-admin和/user-admin

下面一个表格（表19）概括了用户可以看到的重要的按钮位置，按钮文字，和其对应的url：

位置	文字（按钮）	url	条件
导航栏	登录	/login	用户未登录，点开弹出对话框，选确定
	注册	/register	用户未登录，点开弹出对话框，选确定
	个人信息	/profile	用户已登录
	登出	/logout	用户已登录
	数据库管理	/data-admin	用户是管理员
	用户管理	/user-admin	用户是管理员
项目	删除	/delete-project	用户创建了这个项目/用户是管理员
	关注	/star-project	用户已登录
	创建	/create-project	用户已登录
	项目	/samples?project_name=...	无
样本	下载	/download	用户已登录
	更新	/upload	用户创建了这个项目/用户是管理员
	选择范围和键值	/samples?key=value	无
个人信息	查看项目	/sample?project_name=...	用户已登录
	修改密码	/modify-password	用户已登录
用户管理	查看用户	/user?name=...	用户是管理员
	删除用户	/delete-user	用户是管理员

表 19: 用户交互API接口和对应的url

5 数据库设计

5.1 基本信息

根据客户的需求，我们在应用中采用了MongoDB（http://www.mongodb.org），这是目前在IT行业中非常流行的一种非关系型数据库（NoSql），它具有灵活的数据存储方式，可以满足我们对于不同客户存储不同种类项目的需求。

MongoDB中很好的实现了面向对象（Objected-Oriented）的思想。在MongoDB中，数据库（Database）包含若干个集合（Collection），集合又包含若干个记录，这里的每一个记录都是一个档案（Document）对象。MongoDB最大的优势在于所有的数据操作，都无需开发人员调用SQL语句，也不需要考虑数据库的类型，可以通过直接调用指令，对数据库进行操作。

由于我们开发的是Python应用，因此我们采用了PyMongo来对MongoDB直接进行操作。针对键值固定的集合，我们采用了MongoEngine框架，这是一个在保持MongoDB高度有效性前提下，能够对MongoDB中的数据种类和格式进行限制的框架，我们利用MongoEngine来存储用户等信息。

5.2 设计结构

我们的设计如下ER（Entity Relationship）图所示：（见图16）

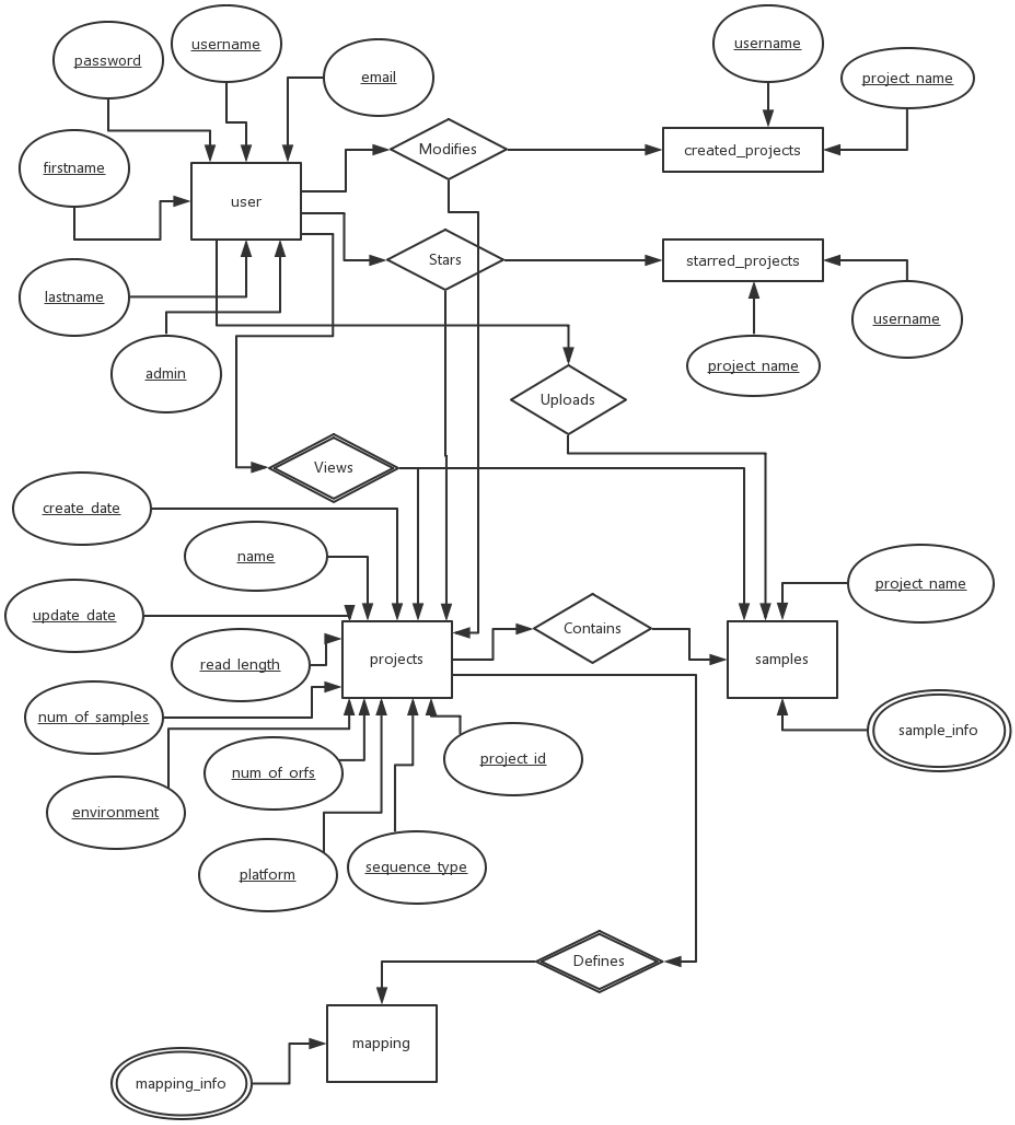

图 16: 数据库实体关系图

其中方形表示集合，圆形表示集合中的键值，双圆形表示集合中不确定数量的键值，菱形表示关系，双菱形表示弱关系其中各个集合的含义如下：

user 记录用户的信息，包括用户名，密码，邮箱，姓名等。

projects 记录项目的基本信息，包括项目名称，创建时间，修改时间等。

samples 记录项目中样本的全部信息，其中样本有所属项目名称信息。

mapping 记录项目类别名称中，中英文对照的信息。

created_projects 记录用户和项目之间的上传关系。

starred_projects 记录用户和项目之间的关注关系。

5.3 说明

1. 实体集User（主键为username）和Projects（主键为name）通过Modify和Star相互联系。在Modify过程中，User可以创建/删除一个项目，并修改created_projects实体集中的元素；在Star过程中，用户可以对一个项目加关注，并修改starred_projects实体集中的元素。

2. 在View动作中，实体集User还可以对项目实体集Projects和样本实体集Samples进行查看。User也可以查看created_projects和starred_projects

3. 在Upload动作中，实体集User可以对Samples实体集进行修改。

4. 在Define动作中，实体集Projects定义了项目键值的中英文转换，并将其存放在Mapping实体集中。

5.4 表结构

5.4.1 Biopedia.user

Biopedia.user对应ER图中的实体集user，记录用户的基本信息。

名称	类型	限制	意义
username	string	必须存在，长度不超过16	用户名
admin	bool	必须存在	是否为管理员
email	string	必须存在	邮箱
firstname	string	必须存在，长度不超过50	名
lastname	string	必须存在，长度不超过50	姓
password	string	必须存在，长度不超过50	密码

表 20: Biopedia.user键值

5.4.2 Biopedia.projects

Biopedia.projects对应ER图中的实体集projects，记录项目的基本信息。

名称	类型	限制	意义
name	string	必须存在	项目名称
project_id	string	必须存在	项目ID，目前由用户自己输入
environment	enum	必须存在	实验环境，目前由用户自己输入
site	enum	必须存在	实验地点，目前由用户自己输入
num_of_total_sequences	int	必须存在	总序列长度
num_of_orfs	int	必须存在	ORF总数
num_of_samples	int	必须存在	样本总数
sequence_type	enum	必须存在	序列类型
platform	enum	必须存在	平台
create_date	date	必须存在	创建日期，有输入格式限制
update_date	date	必须存在	更新日期，有输入格式限制
read_length	int	必须存在	舌苔长度

表 21: Biopedia.projects键值

5.4.3 Biopedia.samples

Biopedia.samples对应ER图中的实体集samples，记录样本的基本信息。

名称	类型	限制	意义
sample_id	string	必须存在	样本ID
project_name	string	必须存在	所归属的样本名称
{key: value}	-	-	样本的键值信息（数量不定），随着项目的不同而不同

表 22: Biopedia.samples键值

5.4.4 Biopedia.mapping

Biopedia.mapping对应ER图中的实体集mapping，记录项目样本键值（key）的中英文对照信息。

名称	类型	限制	意义
project_name	string	必须存在	所归属的项目名称
cn	string	必须存在	key的中文信息
en	string	必须存在	key的英文信息，和samples中的key一致

表 23: Biopedia.mapping键值

5.4.5 Biopedia.starred_projects

Biopedia.starred_projects对应ER图中的实体集starred_projects，记录用户关注项目的信息。

名称	类型	限制	意义
username	string	必须存在，长度不超过16	关注项目的用户名
project_name	string	必须存在	被关注的项目名

表 24: Biopedia.starred_projects键值

5.4.6 Biopedia.created_projects

Biopedia.created_projects对应ER图的实体集created_projects，记录用户创建项目的信息。

名称	类型	限制	意义
username	string	必须存在，长度不超过16	关注项目的用户名
project_name	string	必须存在	被关注的项目名

表 25: Biopedia.created_projects键值

6 设计模式

在这款系统的设计中，我们采用了多种设计模式，这些设计模式极大的增加了代码的可维护性，减少编码时间。

6.1 MVC框架

MVC模式，也就是Model-View-Controller模式，使得我们能够将前端和后端分开，仅保留合适的接口进行交互，可以减少维护上的许多问题。

例如，对于项目管理的模块，我们的Model在MongoDB中，Controller在projects.py中，而View在projects.html中。这三者的关系可以用下图所示：

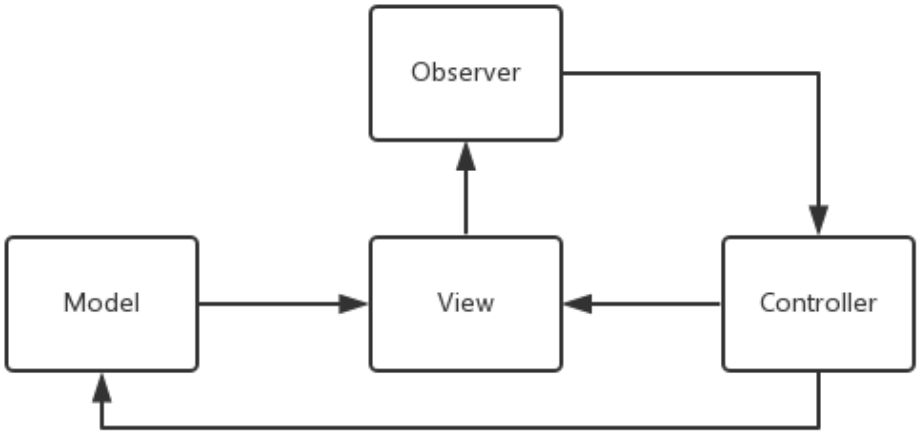

图 17: MVC概念图

其中用户通过View的接口（或者URL）访问Controller，Controller处理用户请求后对Model对应的数据进行调整，之后使用Model中的数据渲染View，三个模块可以顺利地协同工作不互相影响。

6.2 单例（Singleton）模式

Singleton模式，是指在程序的运行过程中，我们对某一个类只可以创建一次，这里Flask类的app就只能够被创建一次，其他文件对app的调用都是通过import来实现的。

我们使用这一模式的目的主要有：

1. 保证数据库操作方唯一；
2. 维护访问计数；
3. 避免访问调用的冲突

6.2.1 开闭原则

“开闭原则”，即“对扩展开放，对修改关闭”，指的是在软件设计当中，当对某一软件实体进行扩展时，尽量不要求修改原有的软件实体。这种设计模式其实是为了更好的可维护性。

我们在很多地方使用了这个原则，下面是几个例子：

导航栏右端显示 我们的网站上方的导航栏是一样的，这样就给我们留下了代码重用的空间。因此，我们在写好了login_and_register_modals.html和nav_bar_right.html之后，就可以通过直接调用include，来自动生成导航栏和登录/注册的界面，而不需要了解其内部细节。

重定向 当用户给定的输入出现非法情况时，我们需要对这些情况进行处理，这时我们使用准备好的方法index.index，然后，如果遇到这些情况，自动跳转到index.index，这时我们也不需要了解index的细节，仅仅通过一个借口就可以进行跳转。

6.2.2 模板方法

模板方法定义了一个演算法的步骤，并且允许类别为一个或者多个步骤提供实践方式。让使用者对一个方法进行重复定义。这种方法往往会提高代码的开发效率，使得代码可读性更强，便于维护。

例如，我们在解决中英文的切换时，经历过以下几个步骤：

1. 分别实现中文页面和英文页面

2. 直接在后端定义需要输出的中文和英文的字符，然后通过传递参数的形式进行中英文切换的实现；

3. 中英文字符在前端定义，后端传递一个参数language，然后通过jinja2语法中的if语句实现切换；

4. 定义一个宏switch_lang('中文', 'English')，效果等价if。

可以发现，最后一种方法，使用了模板方法和MVC方法相结合的方式，使得代码可维护性最好，而且代码量大大缩短，增加了开发速度。

图片列表

图表列表