项目文档

1. 项目名称：AbLoG
2. 开发环境：开发语言：Python3.5.3，数据库管理系统：MySQL5.7.18 Community，前端模板引擎：jinja2，异步web框架：aiohttp。
3. 开发过程：

* 编写web基本骨架：我们的web app是建立在asyncio这个支持异步IO的模型上的，因此在一开始，我们打算先编写一个web app的基本骨架。在这个骨架中，我们确定了将要使用的端口号，并且使用异步IO的策略，使得多个浏览器可以同时访问我们的web app。（如果使用tomcat的话，这是轻而易举的事，但是我们并不是使用tomcat这个成熟的服务器，所以web服务是由自己编写的）。
* 编写ORM：在编写ORM之前，我们首先封装了基本的数据库操作，如SELECT、EXECUTE等，另外还封装了创建、销毁连接池的函数（这样做的好处是显而易见的，可以避免在访问数据库时频繁的连接数据库）。ORM（Object Relational Mapping）是为了避免日后的代码中出现大量重复晦涩的数据库操作。我们使用一个对象映射数据库中的一张表，操作对象的函数封装了对数据库的操作，以后在使用数据库时只要调用这些对象的函数就可以了。
* 编写Model：编写好了ORM之后，我们从数据库的表所对应的对象模板中建立三个对象并初始化，这三个对象分别是user(存储用户信息)、blogs（博客）、comments（评论）。
* 编写web框架：因为aiohttp框架对于url的处理（从request中获取传入参数，自己构造response）比较底层，这势必会造成后续的大量重复性工作，因为，我们在aiohttp的基础上，将对于url的处理进一步封装，使得重复的工作由特定的函数完成。另外，我们还构造了一个middleware，在对url处理之前先进行一次拦截，将返回给服务器的值转换为reaponse对象，这样我们自己编写的web框架才符合aiohttp框架的要求。
* 编写配置文件：因为Python的语法非常简单简洁，所以我们的配置文件同样是由Python编写的。配置文件中包含了数据库的名称、密码和端口号等信息。因为工程部署到服务器上很有可能需要修改部分信息，所以我们还写了另一个不同的配置文件，这样既可以在本地开发，又可以随时将工程部署到服务器上，而不需要不断地修改配置文件。
* 编写MVC：利用先前写好的Web框架和ORM框架中的部分函数，我们编写MVC的业务逻辑，MVC的用户界面和业务逻辑分离开来，这些业务逻辑处理URL，然后将数据传到网页上，我们的网页模板在template文件夹中。
* 构建前端：我们的html网页使用了uikit这个css框架。为了避免每个html网页中都出现大量重复的代码，我们先编写了一个“父模板”，“父模板”包括了各个html网页中相同的样式，各个网页作为“子模板”去继承这个“父模板，就可以拥有父模板中的样式”。
* 编写用户登录、注册页面（前端与后端）：注册和登陆页面的前端样式还是使用uikit这个前端css框架；后端部分，用户注册很简单，只需要获得输入的信息然后存储到数据库中就可以了；用户登录比较麻烦，我们没有使用Session，而是直接读取cookie中的信息来验证登录。并且我们在之前的middleware中添加了解析cookie的代码，这样就不需要在每一个url处理函数中都解析一遍cookie。
* 编写日志创建页：编写日志创建页的主要难点在于如何将负责排版、样式的html代码和负责交互的js代码分割清楚，以方便后期的维护。这里我们使用了MVVM模型，首先用ViewModel将数据和显示关联起来，我们使用的ViewModel框架是Vue框架。通过将Model和View双向绑定在一起，可以将我们在Form表单中输入的信息立刻同步到Model中，也可以使得添加到Model中的信息即刻显示在View上。这样做到了数据和显示的分离，使得后期对于双方的维护更加方便。
* 编写日志列表页：在日志列表页（管理界面）中，我们同样沿用前面所提到的MVVM模型，只需要在处理对应url的函数中添加上对于MVVM的支持，就可以很容易的编写好一个简单易用的日志列表页。
* 代码优化、用户界面美化：然后是我们的收尾工作，在这一步我们编写了日志显示页、用户管理也和评论管理页，因为所用到的知识在前面已经用过，所以在写这些页面的时候并没有遇到什么大问题，有点小问题我们都慢慢哦解决了。同时因为每次对前端或者是后端的代码修改之后，都要手动重启一次服务器，这样频繁的手动重启很麻烦，所以我们还用Python的watchdog模块编写了一个监控代码变化的脚本，这样我们每次保存代码的时候，服务都会自动重启一次，很方便有没有。