

信息安全工程课题研究

|  |  |
| --- | --- |
| 题目： | Slide CPATCH结题报告 |

|  |  |
| --- | --- |
| 成 员： | 龙东恒 宋文浩 许佳 谢贤彬 |
| 院 系： | 软件与微电子学院 |
| 导师姓名： | 孙惠平 |

二〇一七年 十二月

# **目录**

[**目录** 1](#_Toc502262317)

[**一、课题目标回顾** 2](#_Toc502262318)

[**二、** **项目过程概述** 3](#_Toc502262319)

[2.1 Slide CPATCHA 项目结构 3](#_Toc502262320)

[2.2 拼图处理概述 3](#_Toc502262321)

[2.3 前端Web技术概述 3](#_Toc502262322)

[2.3. 后台服务器技术概述 3](#_Toc502262323)

[2.3.1 服务器搭建 3](#_Toc502262324)

[2.3.2 数据库建立 3](#_Toc502262325)

[**三、项目过程细节** 4](#_Toc502262326)

[3.1 拼图处理 4](#_Toc502262327)

[3.2 前端页面 4](#_Toc502262328)

[3.3 后台服务器 4](#_Toc502262329)

[3.4 数据库 4](#_Toc502262330)

[**四、用户手册** 5](#_Toc502262331)

[4.1 源码获取及配置 5](#_Toc502262332)

[4.2 启动程序 5](#_Toc502262333)

[4.3 关闭程序 5](#_Toc502262334)

[4.4 生成拼图 5](#_Toc502262335)

[4.5 录入信息 5](#_Toc502262336)

[**五、解决方案评价** 6](#_Toc502262337)

[5.1 解决方案优势 6](#_Toc502262338)

[5.2 解决方案劣势 6](#_Toc502262339)

[**六 项目体会** 7](#_Toc502262340)

# **一、课题目标回顾**

随着互联网的不断发展和演进，以及社会对个人隐私的重视程度不断的加大，企业公司受到多方面的压力，纷纷使用各类办法保护内部数据，其中包含用户个人数据以及商业数据。当前市场出现了多种保护隐私数据的解决方案，最有效且低成本的方案是CPATCHA验证的使用。CPATCHA以其“Easy for human, hard for machine”的概念被广泛应用在区分人类和机器人行为的识别上，并且在多年的发展过程中，衍生出各类验证方案，如文本、汉字、图片选择及滑块等各式各样的验证码。在对国内市场中专做验证码市场的多个公司的产品进行比对后，小组提出自己的解决方案，该方案具有以下特点：

1. 更多用户交互。用户需要进行简单的三次判断、一次选择和一次拖动。
2. 降低识别风险。无知识型验证码本身不知识来挑战机器，所以机器无法通过学习以破解验证码（目前的方法是图像边界识别），此外新增的多个判断和选择较小的增大用户破解难度，但会急剧增大机器人破解的难度。从而降低了被机器人识别的风险。
3. 更好的兼容性。组内提出的解决方案及其落地的代码结果允许使用者可以通过简单的代码配置完成对此方案的动态配置，如确定子拼图选择个数，确定原图背景中出现的用于模糊视觉的假子拼图个数（0-1）。

综上，此次课题中，我们遵循软件开发中的敏捷开发模型，轻文档多交流，使用代码版本管理工具管理组员贡献的源码，并用市面中相对流行的前端、服务器、图像处理框架完成整个项目的源码编写。

# **项目过程概述**

## Slide CPATCHA 项目结构

在

## 拼图处理概述

正

## 2.3 前端Web技术概述

滑

## 2.3. 后台服务器技术概述

后台的服务器中涉及简易服务器的搭建和数据库的建立，下面将对两者所采用的技术进行简要的说明。

### 2.3.1 服务器搭建

开

### 2.3.2 数据库建立

本项目采用了操作相对比较熟悉的mysql数据库来对项目涉及到的数据进行分类存储，为使python可以操作mysql，下载安装了pymysql模块。如果写程序用pymysql和程序交互，就需要写原生sql语句。如果进行复杂的查询，那sql语句就要进行一点一点拼接，而且不太有重用性，扩展不方便。而且写的sql语句可能不高效，导致程序运行也变慢。

因此为了避免把sql语句写死在代码里，使用ORM框架直接把原生sql封装好了并且以熟悉的方式操作，像面向对象那样。ORM（object relational mapping）,就是对象映射关系程序，简单来说我们类似python这种面向对象的程序来说一切皆对象，但是我们使用的数据库却都是关系型的，为了保证一致的使用习惯，通过orm将编程语言的对象模型和数据库的关系模型建立映射关系，这样我们在使用编程语言对数据库进行操作的时候可以直接使用编程语言的对象模型进行操作就可以了，而不用直接使用sql语言。

orm相当于把数据库也给你实例化了，在代码操作mysql中级又加了orm这一层。

orm的优点：

1隐藏了数据访问细节，“封闭”的通用数据库交互，ORM的核心。他使得我们的通用数据库交互变得简单易行，并且完全不用考虑该死的SQL语句。快速开发，由此而来。

2.ORM使我们构造固化数据结构变得简单易行。

缺点：

无可避免的，自动化意味着映射和关联管理，代价是牺牲性能（早期，这是所有不喜欢ORM人的共同点）。现在的各种ORM框架都在尝试使用各种方法来减轻这块（LazyLoad，Cache），效果还是很显著的。

在Python中，最有名的ORM框架是SQLAlchemy。本次项目采用SQLAlchemy来完成。

SQLAlchemy

优点：

1. 企业级 API，使得代码有健壮性和适应性
2. 灵活的设计，使得能轻松写复杂查询

缺点：

1. 工作单元概念不常见
2. 重量级 API，导致长学习曲线

项目需要的数据库CRUD相对简单，因此选用SQLAlchemy。

同时，Navicat是一套快速、可靠并价格相当便宜的数据库管理工具，专为简化数据库的管理及降低系统管理成本而设。它的设计符合数据库管理员、开发人员及中小企业的需要。Navicat 是以直觉化的图形用户界面而建的，让你可以以安全并且简单的方式创建、组织、访问并共用信息。因此使用Navicat来管理数据库。

# **三、项目过程细节**

开

## 3.1 拼图处理

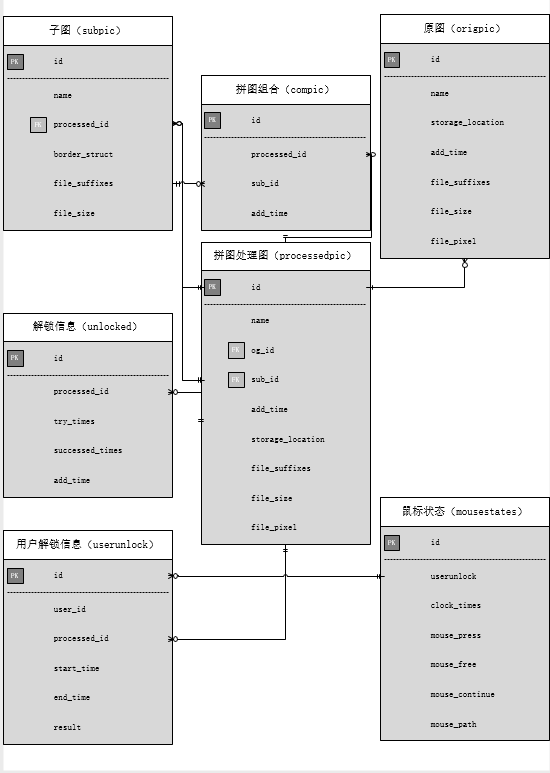
## 3.2 前端页面

## 3.3 后台服务器

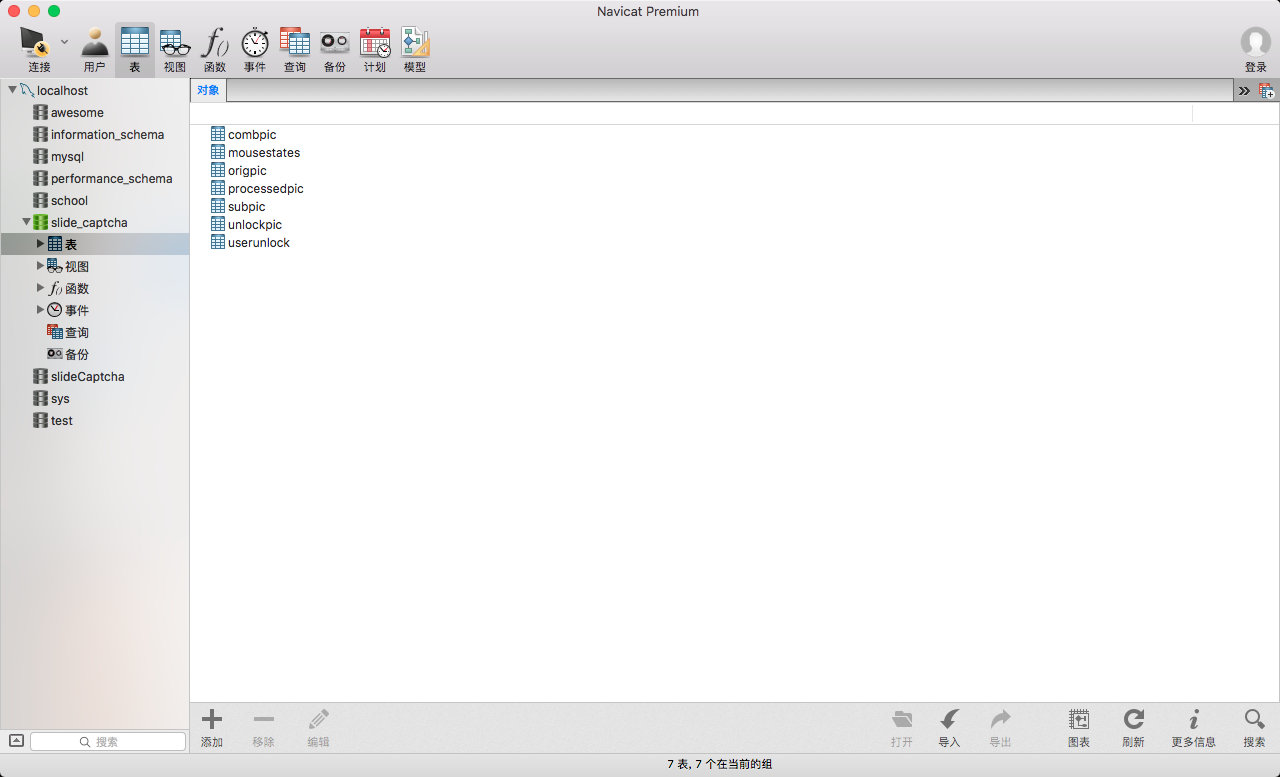
## 3.4 数据库

表结构如下：

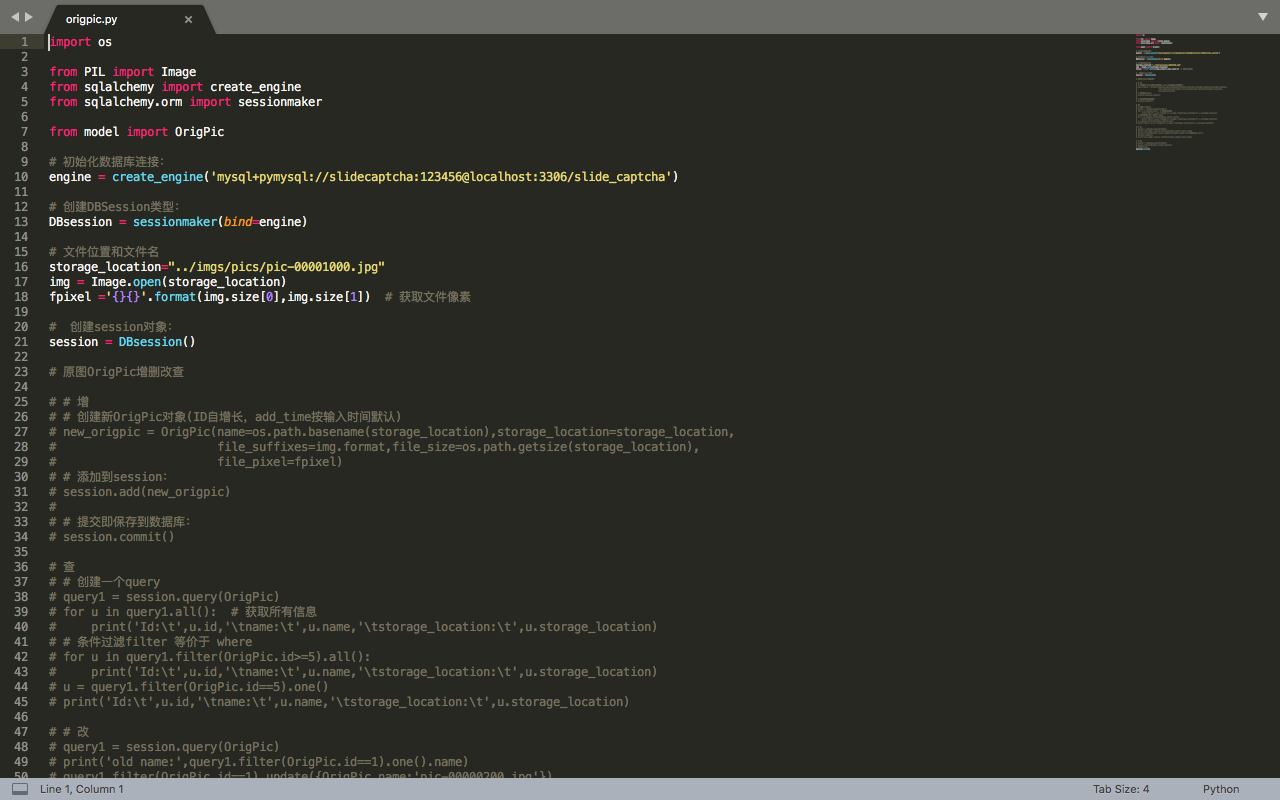
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | 主键 | 属性 | 属性 | 属性 | 属性 |
| origpic | id | name | storage\_location | add\_time | file\_suffixes |
| file\_size | file\_pixel |  |  |
| processedpic | id | name | storage\_location | og\_id | sub\_id |
| file\_suffixes | file\_size | file\_pixel | add\_time |
| subpic | id | name | processed\_id | border\_struct | file\_suffixes |
| file\_size |  |  |  |
| compic | id | processed\_id | sub\_id | add\_time |  |
| unlockpic | id | processed\_id | try\_times | success\_times | avg\_time |
| userunlock | id | user\_id | processed\_id | start\_time | end\_time |
| result |  |  |  |
| mousestates | id | userunlock | clork\_times | mouse\_press | mouse\_free |
| mouse\_continue | mouse\_path |  |  |

表关系如下：

Navicat来管理数据库。如下图所示：



数据库增删改查语句，如下图所示：



# **四、用户手册**

## 4.1 源码获取及配置

## 4.2 启动程序

## 4.3 关闭程序

功。

## 4.4 生成拼图

此

## 4.5 录入信息

# **五、解决方案评价**

项

## 5.1 解决方案优势

## 5.2 解决方案劣势

# **六 项目体会**