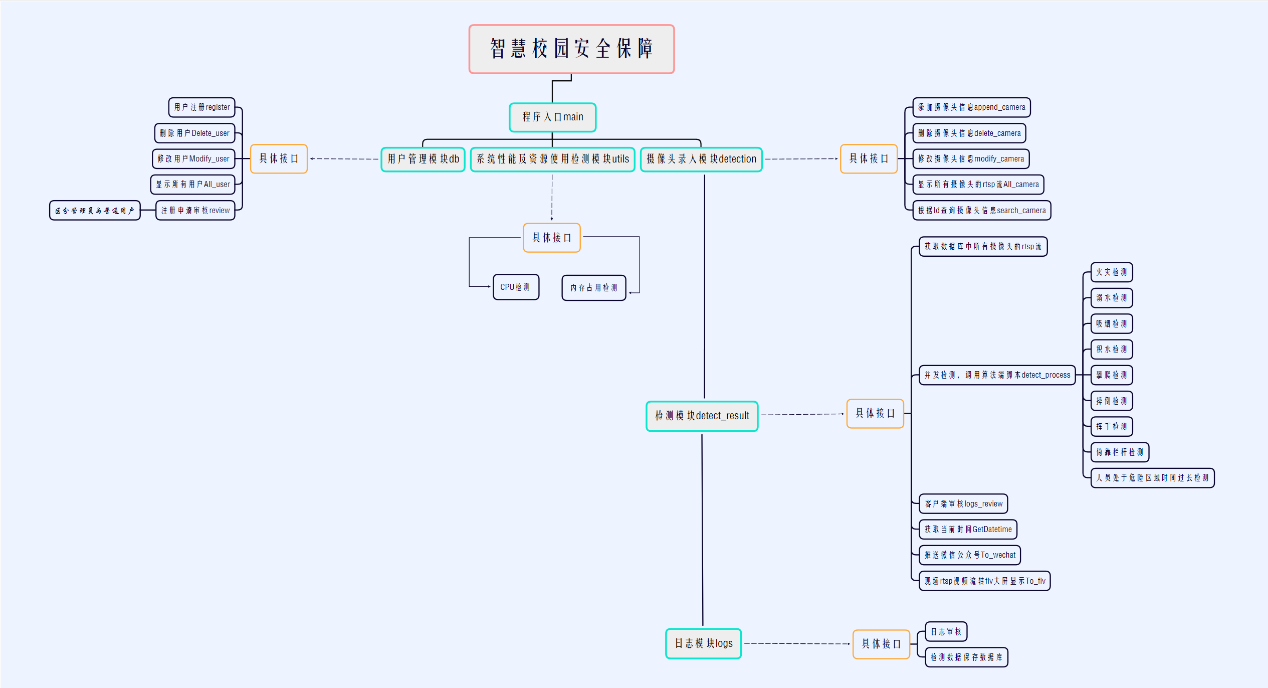
# 系统设计文档

1. 系统架构



1. 该项目共有六个主要模块，分别是：程序入口main、用户管理模块db、系统性能及资源使用检测模块utils、摄像头录入模块detection、检测模块detect\_result、日志模块logs
2. 程序入口main：
3. 使用Go语言的Gin框架实现HTTP服务器，用于处理客户端的请求。在响应中设置允许客户端以不同的协议请求的CORS相关的HTTP头，指定哪些HTTP方法和头信息是被允许的。
4. 创建名为“/stream\_data”的API接口，使用流式传输方式返回一个非常大的字符串，可以避免在服务器端将整个响应数据都缓存到内存中，为其它模块对于处理大量数据和处理时间较长的操作奠定基础。
5. 通过使用Gin框架创建的HTTP服务器，配置不同方法的路由，主要响应客户端的GET和POST请求。将不同请求路径映射到不同的处理函数上。
6. 用户管理模块：
7. 用户注册：借由Gin框架创建的服务器来接受前端返回的用户注册数据，进行严格过滤后进行解析，解析为Re\_User结构的对象。使用原生SQL语句对解析出的对象进行插入数据库的操作，插入到user\_login数据库的register\_users表中，为后面的注册申请审核奠定基础。
8. 删除用户：借由Gin框架创建的服务器来接受前端返回的JSON数据，其中包含了待删除的用户的Id字段值，执行对应的SQL语句将user\_login数据库的users表中Id字段值对应的用户删除，如果发生错误将使用Go语言的回显机制，返回一个非空的error对象，表示解析失败的原因，并将错误信息打印到控制台上。
9. 修改用户：借由Gin框架创建的服务器来接受前端返回的JSON数据，其中包含了用户的Id、用户名、密码、邮箱、身份（管理员/普通用户）、用户对应的编号（学号），将其解析后执行对应的更新SQL语句，如果执行错误会产生回显。
10. 显示所有用户：前端进行对应的显示所有用户的请求后，调用查询的SQL语句，将在user\_login数据库的users表中查询到的每一条记录对应的用户Id、用户名、密码、身份（管理员/普通用户）、用户对应的编号（学号），添加到All\_user\_struct结构体数组中。最后将这个结构体数组作为一个JSON数组返回给前端。
11. 注册申请审核：后端借由Gin框架创建的服务器，将在查询到user\_login数据库的register\_users表中的每一条数据后作为一个JSON数组返回给前端。前端的管理员审核管理界面对每一条注册用户记录进行审核，审核后发送对应的JSON数据。如果审核通过，则将数据添加到user\_login数据库的uses表中，并区分管理员和普通用户，并删除register\_users表中的对应记录。如果审核没有通过，也会调用删除的SQL语句将register\_users表中的注册用户进行删除操作。
12. 系统性能及资源使用检测模块utils
13. CPU检测：使用Go语言的Gin框架和runtime包，实现快速、高性能的CPU监测功能，将其以JSON格式返回给客户端。通过runtime包中的函数获取当前的CPU核心数与运行情况，然后代码将结果封装成CPUUsage结构体，将其转换为JSON格式返回给前端。
14. 内存占用检测：使用Go语言的Gin框架和runtime包，实时的获取当前程序的内存使用情况功能，封装为MemUsage结构体，将其转换为JSON格式返回给前端。
15. 摄像头录入模块detection
16. 添加摄像头：借由Gin框架创建的服务器来接受前端返回的JSON数据，其中包含了摄像头的Id、编号、位置、检测任务、检测报警概率、检测帧率信息，解析后调用对应的SQL语句存入数据库中。
17. 删除摄像头信息：借由Gin框架创建的服务器来接受前端返回的JSON数据，其中包含了待删除摄像头的Id值信息。解析为对应的Delete\_camera\_struct结构体后，调用对应的DELETE SQL语句删除对应的摄像头信息。
18. 修改摄像头信息：借由Gin框架创建的服务器来接受前端返回的JSON数据，解析为Camera结构体后，调用对应的UPDATE SQL语句来更新摄像头信息。
19. 显示所有摄像头的rtsp流：使用对应的SELECT SQL语句，查询detection数据库的cameras表中所有的摄像头记录。将每一条记录添加到JSON数组all\_cameras\_to\_front中。请求成功返回此JSON数组到前端。
20. 检测模块detect\_result
21. 获取数据库中所有摄像头的rtsp流：使用GORM框架来快速地查找detection数据库的cameras表中的每一条记录，将查找记录中的检测任务信息缓存，为下一步并并发调用算法端脚本奠定基础。
22. 并发检测，调用算法端脚本：充分发挥Go语言特性，轻量级并发调用算法端的脚本进行检测。通过os依赖包中的os.Getwd()获取当前工作目录，并用filepath.Join()拼接出Python算法端脚本的完整路径。将函数传入上一步获取到的摄像头的位置、检测任务、检测报警概率和检测帧率，通过exec包执行。通过cmd.StdoutPipe()获取脚本输出的管道，并用bufio创建扫描器，通过循环扫描脚本输出，将输出解析为map[string]interface{}类型的变量，解析其中的字段，将解析后的字段插入到MySQL数据库中。最后，通过Gin框架创建的服务器将报警信息传递给前端。
23. 客户端审核：在并发调用算法端脚本，生成记录返回给前端后，客户端管理员进行审核。审核后将detection数据库中的results表中的记录未审核状态修改为已审核。
24. 获取当前时间：调用time包来获取当前时间，将时间格式化为“2006-01-02 15:04:05”的形式，即年-月-日 时:分:秒。最后将格式化后的字符串作为String字段的值，Valid字段设置为true，转换为sql.NullString。
25. 推送微信公众号：后端使用gin-web框架通过处理HTTP请求，实现推送信息至公众号。首先，To\_weixin函数接收前端处理的POST请求，调用微信API向微信公众号发送消息。然后，调用Get\_token和Get\_openid函数通过发送Get请求，请求API来获得访问令牌和关注公众号的用户ID列表。最后，解析前端传来的JSON，并将其解析为Weixin\_Results结构，构造消息发送到WeChat组，使用WeChat官方API将消息发送给所有关注公众号的用户。
26. 现场rtsp视频流转flv大屏显示：后端使用Go语言的Gin框架和GStreamer，通过gst-launch-1.0工具，实现了从接入模拟现场的测试RTSP视频流中提取H.264编码的视频数据，将其解码后，使用flvmux元素封装为FLV格式的视频文件，并将其保存在磁盘内，并将生成的FLV文件作为HTTP响应的内容类型，设置HTTP响应头为“video/x-flv”，将其传输给客户端。通过使用FFmpeg和GStreamer这两个多媒体框架，低资源占用率的条件下实现了视频的解码、编码、转换等操作；在同一个函数中并发地实现客户选中的8个重点地区的rtsp视频流到flv格式文件的转换。
27. 日志模块logs
28. 日志审核：在并发调用算法端脚本，生成记录返回给前端后，客户端管理员进行审核。审核后将detection数据库中的results表中的记录未审核状态修改为已审核。
29. 检测数据保存数据库：在算法端产生数据后，使用GORM框架连接数据库，并将算法端生成的数据快速地插入到MySQL数据库中。使用GORM另一方面是为了更好的可维护性。
30. 技术选型

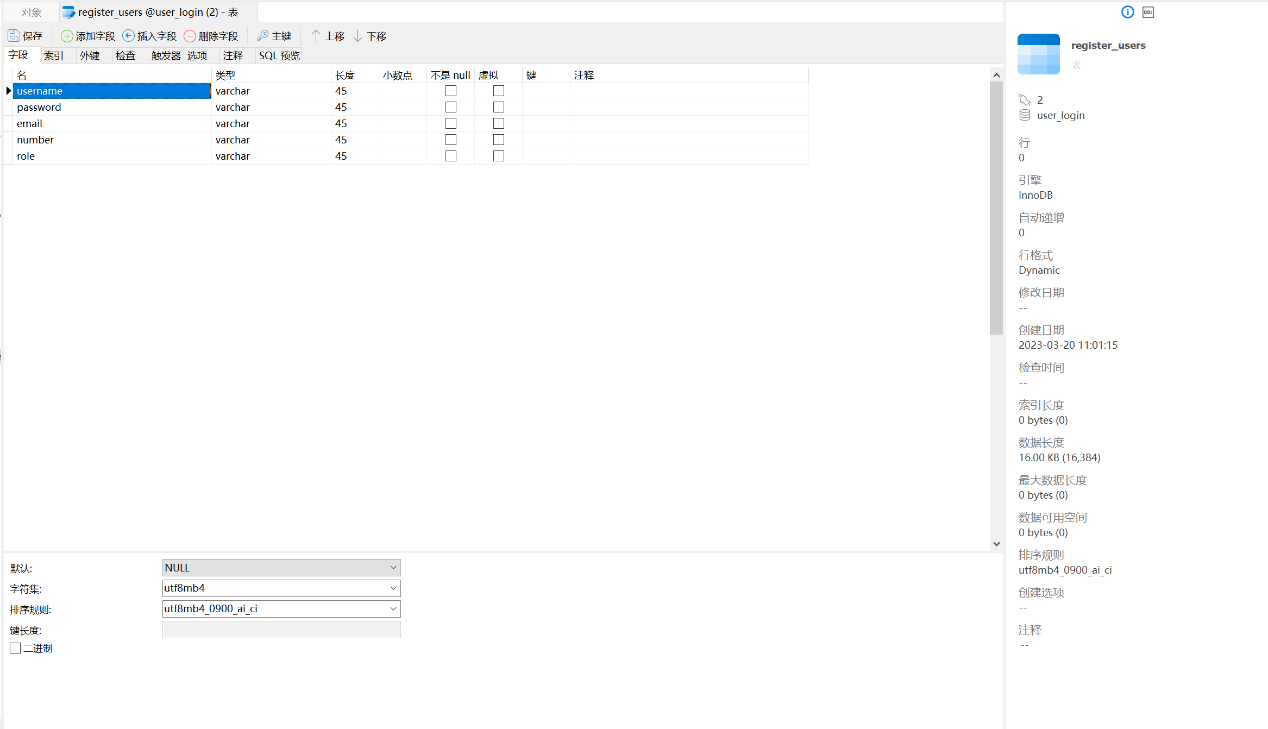
后端采用Golang作为开发语言，使用了Gin作为前后端交互框架，GORM作为与数据库交互框架，使用MySQL作为开发使用数据库。

为了实现项目中的轻量级并发要求，项目采用Golang作为开发开发语言。充分利用goroutine轻量级线程，高效地创建和管理线程。利用channel实现线程安全的通信和同步。

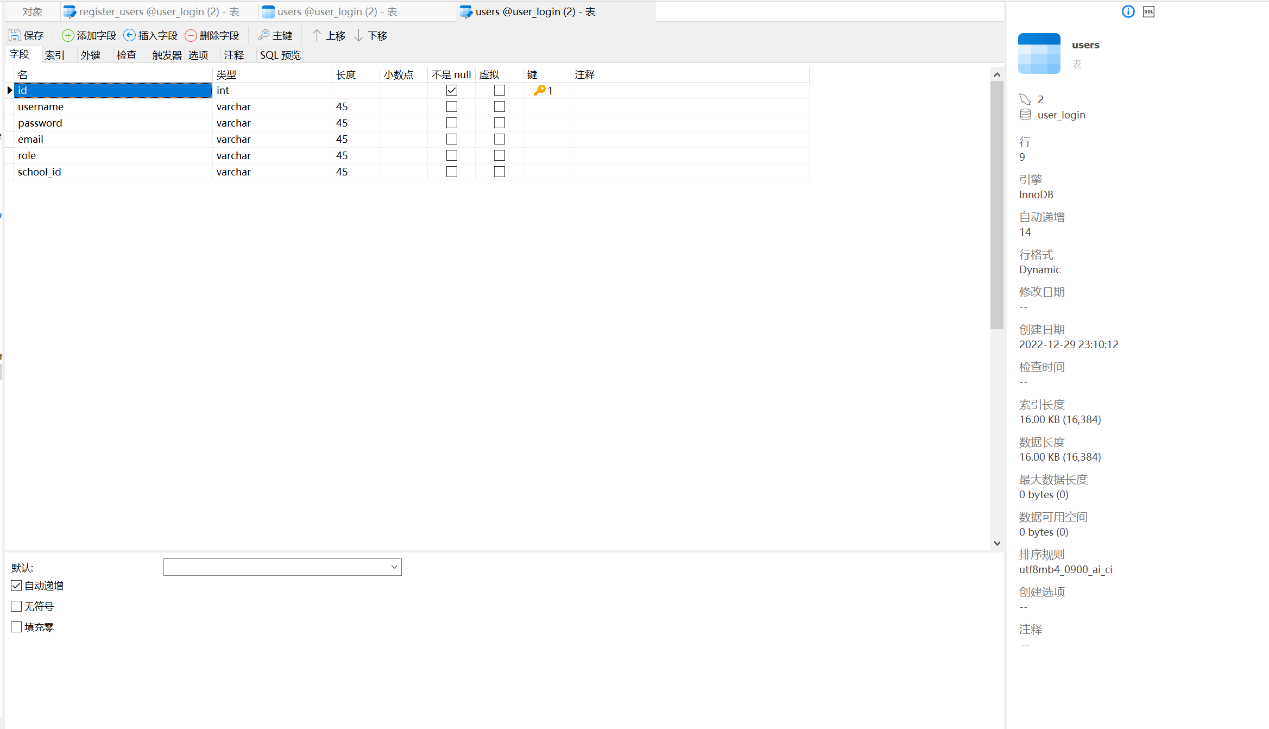
为了满足前后端大量数据的交互，采用高性能的Gin框架，充分发挥其协程和非阻塞I/O等技术，实现高性能的路由和中间件处理，更快地响应请求，满足更多的并发请求的要求。

为了提高开发效率，适当地简化数据库操作、提高程序的可读性、充分利用数据库的功能、提高数据库进行CRUD操作的速度，采用GORM与原生SQL混合开发。

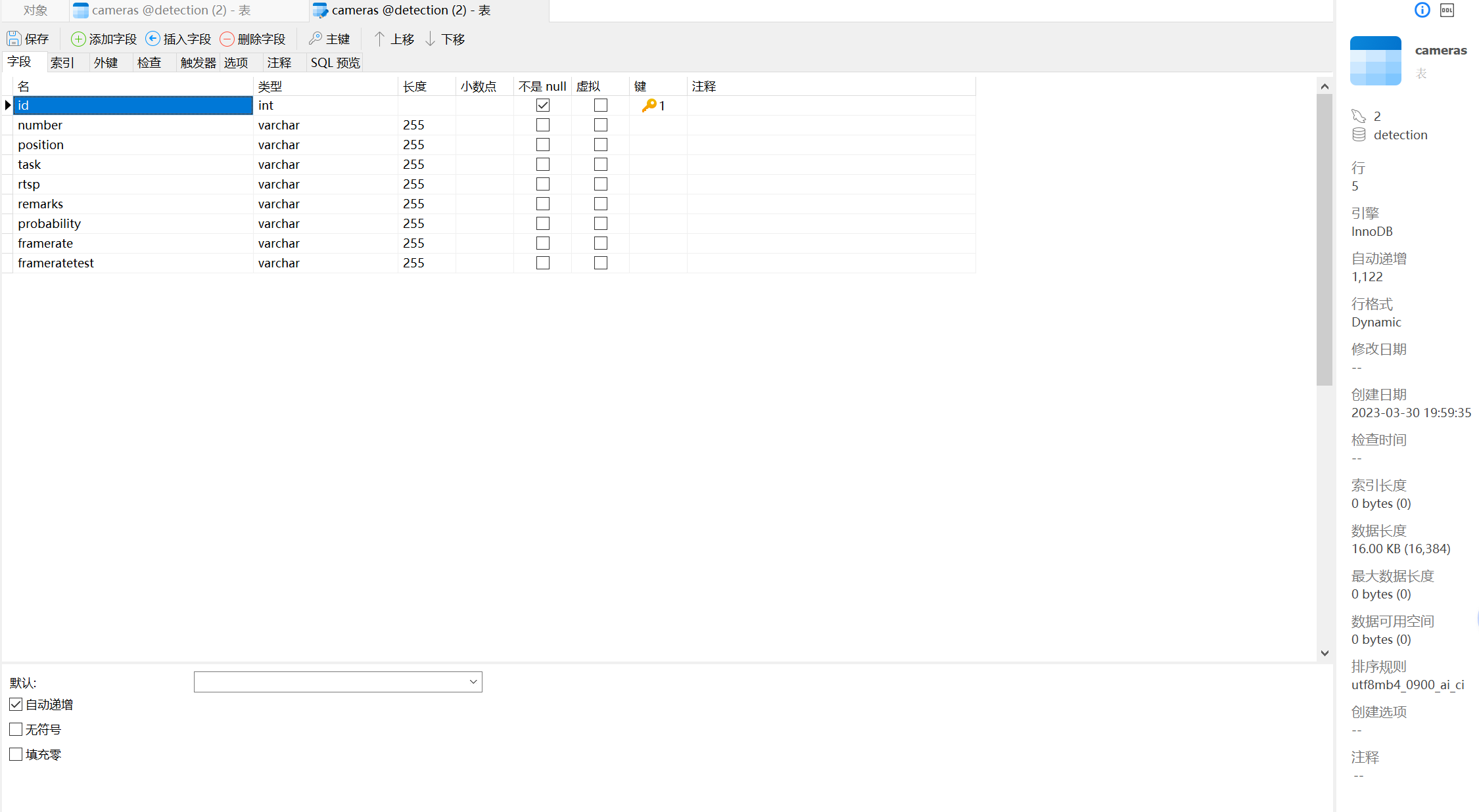
1. 数据库设计
2. 该项目共设计并使用两个数据库，分别是user\_login和detection。数据库运行在3306端口。
3. user\_login数据库中存放用户信息，共设计两张表。
4. register\_users表用来存放注册用户的信息，保存注册用户的用户名、密码、邮箱、编号和身份（管理员/普通用户）信息,均采用varchar(45)的格式进行保存，数据库引擎采用InnoDB,字符集采用utf8mb4。



1. users表用来存放通过注册审核的用户信息，保存注册用户的Id值、用户名、密码、邮箱、身份（管理员/普通用户）、编号（学号），其中Id值采用int自增类型，并将其设置为UNIQUE，索引方法使用BTREE，其余采用varchar(45)的数据类型。数据库引擎采用InnoDB,字符集采用utf8mb4。



1. detection数据库用于存放摄像头录入信息与检测报警记录日志信息，共设计两张表。
2. cameras表用来存放录入摄像头信息，保存摄像头的Id值、编号、位置。任务、rtsp流、备注、检测概率、检测帧率的信息。其中Id值设置为自增的int类型，其余为varchar(255)类型。数据库引擎采用InnoDB。



1. results表用来存放算法端产生的报警信息的id值、图片相对位置、检测概率、任务、位置、检测时间、审核状态的信息。其中id值为int自增类型，图片位置为longtext类型，检测概率为float类型，审核状态为tinyint类型，其余为varchar(255)类型。数据库引擎采用InnoDB。

