中文标题: 宋体 二号字 加粗 居中

英文标题: 大写 小四 加粗 居中

其他文本 宋体 小三

|  |
| --- |
| [公司名称] |
| [文档标题] |
| [文档副标题] |

|  |
| --- |
| Smith feili  [日期] |

**摘要**

内容字体: 小四 宋体字

[关键字:中英文内容摘要基本一致,均要有3-8个关键字各关键字之间要有1个空格分号]

**SUMMARY**

Content font format: sitka banner 小二

**目录**

内容字体: 小四 宋体字

小三号字黑体居中之间空四格,与内容空一行

格式:

第一章:\*\*\*\*\*\*\*\* ^^^^^^1 四号 黑体

第一节:\*\*\*\*\*\*\*\* ^^^^^^^ \* 四号 宋体

第二章: \*\*\*\*\*\*\* ^^^^^^^ \*四号 黑体

--------------- - - - - - - - - - - - - - - - -- - -

毕业设计说明书

1. **绪论**

**1.1目的及意义**

**学生信息管理系统是一个教育单位不可缺失的部分,一个功能齐全,简单易用的信息管理系统不但能有效地减轻学校和相关工作人员的工作负担,它的内容对于学校的决策者和管理者来说都至关重要.**

**作为计算机应用的一部分,使用计算机对学校的各类信息进行管理,具有手工管理所无法比拟的优点,例如:检索迅速,查询方便,效率高,可靠性好,存储量大,保密性好,寿命长,成本低等,这些优点能够极大的提高信息管理系统的效率.**

* 1. **项目开发的目标**

**本项目旨在开发一款学生信息管理系统,以软件方式在一定程度上帮助学校的管理行为.主要有三类功能,教师学生信息管理,课程管理,成绩管理.**

1. **开发工具的选择**

**2.1开发工具选择**

**1.IDE: Qtcreator 6.0.2 community**

**2.语言:c++**

**3.数据库: mysql 8.0.19**

**4.软件生命周期开发模型: 瀑布模型**

**5.qt库版本:qt5.1.29**

**3.需求分析**

**3.1软件需求分析**

**学生信息管理系统的操作角色为教师,学生,管理员,主要分析如下:**

**教师注册教师信息之后,登陆系统可以修改教师信息,查询执教课程信息,执教班级信息;查询学生成绩信息;查看学生信息,查看其他教师信息,与其他教师沟通,与执教班级学生沟通,与管理员沟通.**

**学生通常情况下,注册学生信息之后,登陆系统可以查看自己所学课程信息,查看授课教师信息,班级其他学生信息;查询考试成绩信息,可以与教师沟通,可以与本班级其他学生沟通,与管理员沟通.**

**管理员负责审核教师信息,学生信息;教师入校执教后,管理员负责分配教师执教班级,执教科目;学生入校上学后,管理员负责分配学生班级信息,所学课程信息.管理教师信息,管理学生信息.与教师学生沟通.,与其他管理员沟通**

**综上所述,可得如下需求:**

**教师功能有:教师信息注册,教师信息修改,教师信息查询,学生信息查询,与其他教师沟通,与执教班级学生沟通,查询成绩信息,查询课程信息,与管理员沟通**

**学生功能有:学生信息注册,学生信息修改,学生信息查询,查询课程信息, 查询成绩信息,与课程老师沟通,与本班级其他学生沟通,与管理员沟通**

**管理员功能:分配学生班级,分配教师授课班级,分配学生所学课程,分配教师授课课程,管理学生信息,管理教师信息.与教师,学生,其他管理员沟通**

**用例图如下图所示:**

**3.2数据库需求分析**

**由3.1软件需求知,软件的数据库需要存储教师信息,学生信息,管理员信息,考试成绩信息,班级信息,课程信息,即需要根据以上信息建立数据库表和数据库表之间的关系.**

**ER模型如图所示**

**4.总体设计**

**软件总体来看分为服务器/客户端架构,即需要编写服务器和客户端两个部分.为了减轻客户端开发负担,易于测试,所以在这个软件中,先开发服务器端,并将软件的核心功能都集中在服务器端.**

**教师,学生使用客户端向服务器端发送功能请求,服务器端根据客户端请求进行功能选择或交由管理员进行处理,并将功能的执行结果返回给客户端.服务器端和客户端根据消息类格式通信.**

**4.1服务器端总体设计(不包含ui界面)**

**服务器端功能主要包括登陆,注册,数据库操作.**

**根据3.1的用例图及3.2的ER图分析得服务器端类图主要结构如下所示:**

**4.2客户端总体设计**

**客户端登陆时向服务器发送账号密码验证消息,主要功能为向服务器请求数据以及处理服务器返回的请求数据,按照一定规则处理服务器返回的数据展示给用户.**

**客户端类图主要结构如下图所示:**

**4.3数据库逻辑设计**

**由3.2得出的ER图分析得学生信息管理系统数据库各个表的逻辑结构如下列表格所示:**

**教师表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **工号** | **integer** | **主键** | **非空** | **累加** |
| **姓名** | **varchar(10)** |  | **空值** | **空值** |
| **性别** | **varchar(2)** |  | **非空** | **‘M’** |
| **年龄** | **varchar(2)** |  | **空值** | **空值** |
| **电话** | **varchar(11)** |  |  |  |
| **职务** | **varchar(10)** |  |  |  |
| **课号** | **integer** | **外键** |  |  |

**学生表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **学号** | **integer** | **主键** | **非空** | **累加** |
| **姓名** | **varchar(10)** |  | **空值** | **空值** |
| **性别** | **varchar(2)** |  | **非空** | **‘M’** |
| **年龄** | **varchar(2)** |  | **空值** | **空值** |
| **电话** | **varchar(11)** |  |  |  |
| **班号** | **integer** | **外键** |  |  |

**教师学生表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **工号** | **integer** | **外键** |  |  |
| **学号** | **Integer** | **外键** |  |  |

**班级表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **班号** | **integer** | **主键** | **非空** | **累加** |
| **人数** | **integer** |  | **空值** | **空值** |
| **老班** | **integer** | **外键** |  |  |
| **班长** | **Integer** | **外键** |  |  |

**教师班级表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **工号** | **integer** | **外键** |  |  |
| **班号** | **integer** | **外键** |  |  |

**课程表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **课号** | **integer** | **主键** | **非空** | **累加** |
| **课名** | **varchar(10)** |  | **空值** | **空值** |

**考试表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **考试** | **integer** | **主键** | **非空** | **累加** |
| **名称** | **varchar(10)** |  | **空值** | **空值** |
| **时间** | **varchar(2)** |  | **非空** |  |

**学习表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **学号** | **integer** | **外键** |  |  |
| **课号** | **integer** | **外键** |  |  |
| **考试** | **integer** | **外键** |  |  |
| **成绩** | **integer** |  | **非空** | **0** |

**管理员登陆表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **id** | **Integer** | **主键** | **非空** | **累加** |
| **姓名** | **varchar(10)** |  | **空值** | **空值** |
| **密码** | **varchar(16)** |  | **非空** | **‘123’** |

**教师学生登陆表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **账号** | **integer** | **主键** | **非空** | **累加** |
| **密码** | **varchar(16)** |  | **非空** | **‘123’** |

**教师图片表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **编号** | **integer** | **主键** | **非空** | **累加** |
| **工号** | **integer** | **外键** |  |  |
| **路径** | **varchar(100)** |  |  |  |

**学生图片表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **键** | **空值** | **默认** |
| **编号** | **integer** | **主键** | **非空** | **累加** |
| **学号** | **integer** | **外键** |  |  |
| **路径** | **varchar(100)** |  |  |  |

**5.详细设计**

**5.1 服务器端详细设计(不包含ui界面):**

**(1).建立消息类,作为服务器端和客户端交流通信得数据格式.**

**消息类属性:**

**1.消息种类**

**2.发送者账号**

**3.发送者类型**

**4.接收者账号**

**5.接收者类型**

**6.内容**

**消息类消息种类属性各值含义:**

|  |  |
| --- | --- |
| **消息的种类** | **内容的含义** |
| **0** | **查询登陆密码** |
| **1** | **查询** |
| **2** | **修改数据库请求** |
| **3** | **注册账号信息** |
| **4** | **用户之间的沟通信息** |
| **5** | **传输的图片数据** |
| **6** | **服务器与客户端连接成功** |
| **7** | **将内容交给管理员查看** |
| **10** | **客户端的查询语句,查询教师信息.** |
| **11** | **客户端的修改语句,无需经过管理员同意** |
| **12** | **客户端登陆请求的结果** |
| **13** | **客户端登陆请求的结果** |
| **14** | **注册请求的结果** |
| **15** | **注册请求得结果** |
| **16** | **客户端的查询语句,查询学生信息** |
| **17** | **客户端的查询语句,查询班级信息** |
| **18** | **客户端的查询语句,查询成绩信息** |
| **19** | **客户端初始化查询语句,查询教师,学生,班级** |

**=**

**同一行的值以\t作分隔符换行以\n标识.**

**服务器与客户端通信时序列化message类为字符串,每个属性之间用\r**

**隔开.**

**备注:如果消息只有种类和内容属性值不为0,则为该类型的通知消息.**

**以上每个类型的消息都有一个种类和内容属性不为0的通知消息.**

**发送者类型和接收者类型如下表所示**

|  |  |
| --- | --- |
| **数值** | **含义** |
| **0** | **管理员** |
| **1** | **教师** |
| **2** | **学生** |

**(2).建立消息操作类,用于建立客户端和服务器端的socket连接.**

**服务器端为每个连接成功的客户端建立一个消息操作类对象.**

**通过消息操作类,服务器端收到来自客户端消息,经过消息分析类的处理之后,将消息的操作结果通过消息操作类返回给客户端.**

**消息操作类内部维护一个消息类队列,用于存储客户端发送来的消息.**

**消息操作类属性:**

1. **消息类队列 --- 存储客户端发送的消息**
2. **消息分析类对象 --- 用于处理消息**
3. **socket连接表 --- 以哈希表存储客户端的连接对象**

**消息操作类方法:**

1. **读取消息 --- 读取客户端发送来的消息放到消息类队列.**
2. **消息处理 --- 从消息类队列中取消息交给消息分析类对象处理.**
3. **发送消息 --- 将消息处理结果返回给客户端**

**4. 建立连接 --- 在设定好的端口监听客户端连接,每收到一个合法连接开启一个线程与之通信,之后继续监听.**

**(3).建立消息分析类,用于对消息操作类接收到的消息进行解析.**

**通过消息分析类,对消息操作传递的消息按照消息种类解析,根据消息种类调用其内部适当的处理函数,将解析后的消息作为处理函数的参数进行处理,处理函数将消息的处理结果重新封装为消息类返回给消息分析类.**

**消息分析类属性:**

1. **登陆类对象**
2. **查询类对象**
3. **修改类对象**

**4.注册类对象**

**5.沟通类对象**

**消息分析类方法:**

**1.解析消息 --- 将传递来的消息按照消息种类进行解析.**

**2.登陆处理 --- 登陆类对象调用内部方法进行处理.**

**3.查询处理 --- 查询类对象调用内部方法进行处理**

**4.修改处理 --- 修改类对象调用其内部方法进行处理**

**5.注册处理 --- 注册类对象调用其内部方法进行处理**

**6.沟通处理 --- 根据socket连接表转发消息给接收对象**

**(4).建立服务器类,用于初始化服务器的后台处理对象及ui界面.**

**服务器类属性:**

**1.登陆管理员队列**

**2.待审核消息队列**

**3.消息操作类对象**

**服务器类方法:**

**1.管理员登陆审核**

**2.管理员注册审核**

**3.启动服务器**

**4.关闭服务器**

**5.修改数据库信息**

**(5).建立登陆处理类,用于验证客户端用户登陆,保存已登录的客户端列表.**

**登陆处理类属性:**

**1.登陆教师队列**

**2.登陆学生队列**

**登陆处理类方法:**

**1.检查账号 --- 临时建立一个数据库操作类对象查询数据库**

**以检查账号合法性.**

**2.登陆 --- 封装登陆合法性消息返回**

**(6).建立查询类**

**查询类方法:**

**1.查询 --- 将消息进一步生成为sql语句及查询目标,得到具 体的查询请求,并调用以下方法的一个或多个.**

**2.查询教师 --- 临时建立一个数据库操作类对象查询数据库将数据封装为消息类格式返回**

**3.查询学生 --- 同上**

**4.查询学习 --- 同上**

**5.查询班级 --- 同上**

**6.查询课程 --- 同上**

**7.查询教班 --- 同上**

**(7).建立修改类**

**修改类方法:**

**1.修改 --- 生成sql语句并调用以下方法的一个或多个**

**2.修改教师 --- 临时建立一个数据库操作类对象修改数据库**

**将数据封装为消息类格式返回**

**3.修改学生 --- 同上**

**4.修改学习 --- 同上**

**5.修改班级 --- 同上**

**6.修改课程 --- 同上**

**7.修改教班 --- 同上**

**8.审核 --- 将sql语句提交给服务器待审核消息队列,之后直**

**接返回.默认教师和学生对自己的个人信息修改不需经过审核.**

**(8).建立注册处理类**

**注册处理类方法:**

**1.审核 --- 将消息传递给待审核消息队列,之后直接返回.**

**2.注册 --- 用于注册账号.**

**(9).数据库操作类**

**数据库操作类方法:**

**1.查询 --- 连接数据库进行查询操作**

**2.修改 --- 连接数据库进行修改操作**

**(10).管理员类**

**管理员类属性:**

**1.id**

**2.姓名**

**3.服务器对象**

**5.2客户端详细设计**

**(1)建立消息类**

**消息类属性和方法与服务器端的消息类完全一致.**

**(2)建立客户端类**

**客户端类属性:**

**1.消息交互类对象**

**2.班级信息 --- 用于保存信息便于下次查阅.**

**3.教师信息 --- 同上.**

**4.学生信息 --- 同上**

**5.课程信息 --- 同上**

**6.沟通信息 --- 同上**

**7.成绩信息 --- 同上**

**客户端类方法:**

1. **ui对象初始化 --- 用于初始化客户端界面所需要的资源**

**(3)建立消息交互类**

**消息交互类属性:**

1. **消息类队列**
2. **消息处理类**

**消息交互类方法:**

**1.接收消息 --- 接收来自服务器的消息.**

**2.消息处理 --- 调用分析消息方法分析消息.**

**3.发送消息 --- 将消息发送给服务器.**

**4.连接服务器 --- 向服务器发送连接请求**

**(4).建立消息处理类**

**消息处理类属性:**

**1.展示数据类对象**

**消息处理类方法:**

**1.分析消息 --- 对消息依据其标准格式进行分析.**

**2.展示数据 --- 调用展示数据类对象的相应展示数据方法.**

**(5).建立展示数据类**

**展示数据类方法:**

**1.班级成绩 --- 按照班级成绩数据展示格式进行展示**

**2.学生信息 --- 展示学生信息.**

**3.教师信息 --- 展示教师信息.**

**4.班级信息 --- 展示班级信息**

**5.课程信息 --- 展示课程信息**

**6.沟通信息 --- 展示沟通信息**

**(6).沟通交流类**

**沟通交流类方法:**

**1.沟通 --- 发送客户端要发送的信息,显示发送信息,保存沟通信息.**

**(7).注册类**

**注册类方法:**

**1.发送消息 --- 向服务器端发送注册信息.**

**(8).登陆类**

**登陆类方法:**

**1.发送消息 --- 向服务器端发送登陆消息.**

**(9).数据操作类**

**数据操作类方法:**

**1.查询班级信息 --- 获取要查询的班级信息,封装为消息类格式**

**发给服务器端.**

**2.查询教师信息 --- 获取要查询的教师信息,封装为消息类格式**

**发给服务器**

**3.查询学生信息 --- 获取要查询的学生信息,封装为消息类格式**

**发给服务器**

**4.查询课程信息 --- 获取要查询的课程信息,封装为消息类格式**

**发给服务器**

1. **查询成绩信息 --- 获取要查询的成绩信息,封装为消息类格式**

**(10).由于客户端和服务器端相互通信是面向字节流的通信,因此在传输过程中为message类的序列化提供了一个工具类message\_serialization用于在服务器和客户端之间传输消息的过程中序列化和反序列化消息.**

**6.编码**

**6.1编码标准**

**1.软件程序使用utf-8字符集编码**

**2.数据库使用utf-8字符集编码**

**3.客户端服务器端通信时用utf-8字符集编码传送**

**设置msvc编译器编译utf-8文件出错问题**

msvc{

QMAKE\_CXXFLAGS +=/utf-8

QMAKE\_CFLAGS += /utf-8

}

**6.2 数据库代码:**

**创建数据库:**

**create database stu\_mgr DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;**

**创建课程表:**

**create table courses(**

**course\_id int PRIMARY KEY,**

**course\_name varchar(10));**

**创建教师表:**

**create table teachers(**

**teacher\_id int PRIMARY KEY,**

**name varchar(10),**

**gender varchar(2),**

**age int,**

**tel varchar(11),**

**duties varchar(10),**

**course\_id int,**

**FOREIGN KEY(course\_id) REFERENCES courses(course\_id)**

**on delete cascade \**

**on update cascade);**

**创建学生表:**

**create table students(**

**stu\_id int PRIMARY KEY,**

**stu\_name varchar(10),**

**gender varchar(2),**

**age int,**

**tel varchar(11),**

**class\_id int);**

**创建班级表:**

**create table classes(**

**class\_id int PRIMARY KEY,**

**stu\_num int,**

**head\_teacher\_id int,**

**head\_stu\_id int,**

**FOREIGN KEY(head\_teahcer\_id)**

**REFERENCES teachers(teacher\_id)**

**on delete cascade \**

**on update cascade,**

**FOREIGN KEY(head\_stu\_id) REFERENCES**

**students(stu\_id)**

**on delete cascade \**

**on update cascade);**

**ALTER TABLE students ADD CONSTRAINT FK\_stu\_cls**

**FOREIGN KEY (class\_id)**

**REFERENCES classes(class\_id)**

**on delete cascade \**

**on update cascade;**

**创建考试表:**

**create table examination(**

**exam\_id int PRIMARY KEY,**

**exam\_name varchar(10) NOT NULL,**

**exam\_time datetime NOT NULL);**

**创建教师班级表:**

**create table teacher\_class(**

**teacher\_id int,**

**class\_id int,**

**FOREIGN KEY(teacher\_id) REFERENCES teachers(teacher\_id)**

**on delete cascade \**

**on update cascade,**

**FOERIGN KEY(class\_id) REFERENCES classes(class\_id)**

**on delete cascade \**

**on update cascade,**

**PRIMARY KEY(teacher\_id, class\_id));**

**创建学习表:**

**create table study\_record(**

**stu\_id int,**

**course\_id int,**

**exam\_id int,**

**FOREIGN KEY(stu\_id) REFERENCES students(stu\_id)**

**on update cascade \**

**on delete cascade,**

**FOREIGN KEY(course\_id) REFERENCES courses(course\_id)**

**on update cascade \**

**on delete cascade,**

**FOREIGN KEY(exam\_id) REFERENCES examination(exam\_id)**

**on update cascade \**

**on delete cascade,**

**grade int,**

**PRIMARY KEY(stu\_id, course\_id, exam\_id));**

**创建教师学生表:**

**create table teacher\_student(**

**teacher\_id int,**

**stu\_id int,**

**FOREIGN KEY(teacher\_id) REFERENCES teachers(teacher\_id)**

**on update cascade \**

**on delete cascade,**

**FOREIGN KEY(stu\_id) REFERENCES students(stu\_id)**

**on update cascade \**

**on delete cascade,**

**PRIMARY KEY(teacher\_id,stu\_id));**

**创建管理员账户表:**

**create table administrator(**

**admin\_id int PRIMARY KEY,**

**password varchar(16) NOT NULL);**

**创建用户表:**

**create table users(**

**user\_id int PRIMARY\_KEY,**

**password varchar(16) NOT NULL);**

**创建教师图片路径表:**

**create table teacher\_image(**

**id int primary key,**

**teacher\_id int,**

**img\_path varchar(100),**

**FOREIGN KEY(teacher\_id) references teachers(teacher\_id)**

**on update cascade \**

**on delete cascade);**

**创建学生图片路径表:**

**create table stu\_image(**

**id int primary key,**

**stu\_id int,**

**img\_path varchar(100),**

**FOREIGN KEY(stu\_id) references students(stu\_id)**

**on update cascade \**

**on delete cascade);**

**修改教师表存储过程代码:**

**delimiter //**

**create procedure update\_teachers\_info(in n varchar(10), in g varchar(2), in a int, in t varchar(11), in d varchar(10), in c int, in t\_id int, in c\_n varchar(10))**

**begin**

**update teachers**

**set name = n ,gender = g, age = a, tel = t, duties = d, course\_id =**

**(select course\_id from courses where course\_name = c\_n)**

**where teacher\_id = t\_id;**

**end//**

**delimiter ;**

**修改学生表存储过程代码:**

**6.3服务器端编码过程**

**服务器和客户端初始化规则:**

**客户端登陆时向服务器发送自己的身份消息,服务器接收之后根据消息标识客户端,并将其存储到服务器内的hash表中,之后向客户端发送ready消息.客户端接收到ready消息之后与服务器端开始正式通信.**

**1.ui前端控件与后台数据的交互使用qt的信号槽机制实现.**

**(1).本系统中应用的核心代码结构:**

1. **信号发送者 – btn(QPushButton\*对象)**
2. **自定义信号 – signals: void btn\_clicked(int value);**
3. **接收者- 主窗口对象(MainWindow\* mainwindow)**
4. **槽函数 – slots: btn\_process(int value)**
5. **连接:**

**Qobject::connect(btn,SIGNAL(btn\_clicked(int),mainwindow,SLOT(btn\_process(int)));**

**(2).信号槽调用结构**

**1.接收信号对象收到信号时,发送信号**

**emit btn\_clicked(value);**

**2.收到信号后调用主窗口对应槽函数btn\_clicked(int).**

**备注:在使用槽信号机制的时候,为了区分按钮,为每个按钮对象**

**添加了一个索引,登陆界面按钮索引表如下:**

|  |  |
| --- | --- |
| **索引** | **含义** |
| **0** | **登陆** |
| **1** | **注册** |
| **2** | **退出程序** |

**为了后期方便统一美观优化,在服务器主界面不再使用按钮,改用标签作为按钮.**

**索引表如下:**

|  |  |
| --- | --- |
| **索引** | **含义** |
| **8** | **消息内容主要页面** |
| **9** | **消息列表上一页** |
| **10** | **消息列表下一页** |
| **11** | **同意消息请求** |
| **12** | **不同意消息请求** |
| **13** | **教师学生信息管理开关** |
| **14** | **教师信息查看** |
| **15** | **学生信息查看** |

|  |  |
| --- | --- |
| **索引** | **含义** |
| **0** | **服务器功能选择开关** |
| **1** | **未处理消息查看开关** |
| **2** | **关闭服务器** |
| **3** | **开启服务器开关** |
| **4** | **关闭服务器开关** |
| **5** | **服务器记录页面** |
| **6** | **未处理消息显示列表** |
| **7** | **消息内容顶端页面** |

|  |  |
| --- | --- |
| **索引** | **含义** |
| **16** | **信息查看列表上一页** |
| **17** | **信息查看列表下一页** |
| **18** | **信息标题栏** |
| **19** | **搜索** |
| **20** | **筛选** |
| **21** | **个人信息页面** |
| **22** | **确认信息修改** |
| **23** | **取消消息修改** |
| **24** | **返回选择页面** |

|  |  |
| --- | --- |
| **索引** | **含义** |
| **25** | **添加信息** |
| **26** | **显示班级列表** |
| **27** | **显示课程列表** |
| **28** | **确定** |
| **29** | **取消** |
| **30** | **添加** |
|  |  |
|  |  |

**2.服务器登陆界面如图所示:**

**3.服务器主界面如图所示:**

**4.连接数据库代码:**

QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");

db.setHostName("localhost");

db.setDatabaseName("person");

db.setUserName("root");

db.setPassword("root");

bool ok = db.open();

QSqlQuery query;

query.prepare("SELECT id,name,balance "

"FROM account");

query.exec();

qDebug()<<"the num of queried row: "<<query.size();

if(query.isActive())

{

while(query.nextResult())

{

QString id = query.value(0).toString();

QString name = query.value(1).toString();

QString balance = query.value(2).toString();

qDebug()<< id << ", " << name << ", " << balance << " .";

}

}

db.close();

**select函数返回数据格式: 查询内容 + 查询内容行数, 执行失败返回空串**

**update函数返回数据格式: 表的被修改行数,执行失败返回-1**

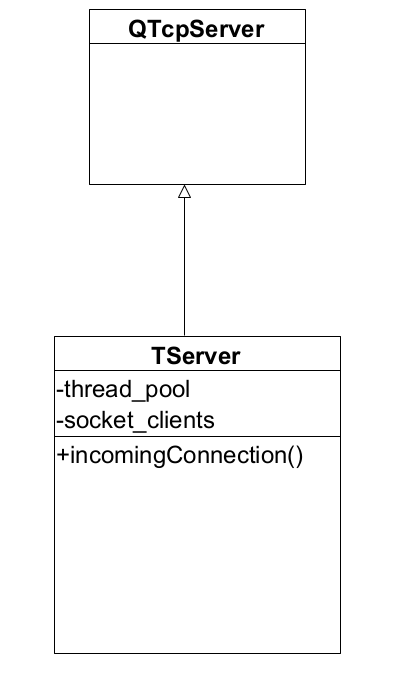
**备注:由于服务器会接受多个线程访问,所以存在多个线程同时访问数据库的现象.对此我做出的处理是使用互斥锁每次仅允许一个线程获得数据库连接得使用权限.**

**5.显示消息列表:**

**设索引大于20为消息索引,调用消息列表处理函数.每次显示10 条记录**

**6.搭建服务器框架:**

**创建QTcpServer子类TServer,TServer重写了QTcpServerde incomingConnection方法,类如如图所示:**

****

**通过重写incomingConnection方法,每当有新连接的客户端socket出现时,我们都可以为其新开一个线程,在线程里通过其socket描述符获取socket对象,由于跨线程调用会出现socket notifier不能正常工作,导致异常退出.**

**最终,得到了一个简易的多线程服务器框架.**

**7.显示服务器控制页面信息.**

**收到一个新的连接socket时,发送show\_new\_connection(int)信号给主线程,并将消息加入显示服务器控制页面信息缓存队列,调用update函数,重新绘制所有消息.**

**8.服务器与客户端交互方式:**

**客户端向服务端发出功能请求时,服务端获取客户端的消息开始处理消息,此时客户端不能再次向服务器端发送功能请求,所有功能请求由客户端阻拦直到服务器返回请求结果结束阻拦.**

**9.服务器处理消息方式:**

**每个客户端线程各自拥有一个消息处理对象用于处理消息.客户端线程将消息传递给message\_operation对象,message\_operation对象根据消息的type属性确定消息请求功能,将消息传递给message\_analysis对象进而确定客户端使用的功能.或不处理消息将消息放入服务器的未处理消息队列,由管理员决定请求是否执行.最终消息请求处理完成后,由message\_operation对象将结果返回给调用者.**

**10.管理员在服务器端审核注册的教师及学生**

**教师在客户端界面注册账号时,点击教师注册,客户端生成对应的sql语句并封装为message消息格式发送给服务端,学生注册账号同理.管理员收到消息后审核通过即可注册成功.**

**11.管理员在服务器端为已注册的教师及学生填入个人信息.**

**管理员在人员查看页面修改教师及学生的个人信息.后台具体流程:**

**服务器按照指定数量从数据库中获取人员信息,在信息列表中显示查询出的人员账号,用户点击对应的账号控件,数据库则查询该用户账号对应的全部信息并展示在页面内容中.**

**管理员可以对查询出的内容作出修改,点击确定按钮修改生效,点击取消按钮恢复未修改前的状态.**

**12.软件服务器端和数据库交互详情:**

**用户在使用服务器检索数据时,服务器将检索出的数据按照一定组织格式呈现给用户,用户可以查看数据和修改数据,修改数据时需要双击要修改的数据字段.**

**在用户双击鼠标修改数据时,服务器将修改的数据的值保存到修改数据队列中,用户修改完字段值后将该字段值保存到待修改数据队列中,同时显示用户的修改信息.**

**用户点击确定按钮后,服务器依据待修改数据队列中的值对数据库作出相应的修改.点击取消按钮时,服务器依据修改数据队列的数据显示原来未修改的数据.**

**12.添加和修改班级课程信息.**

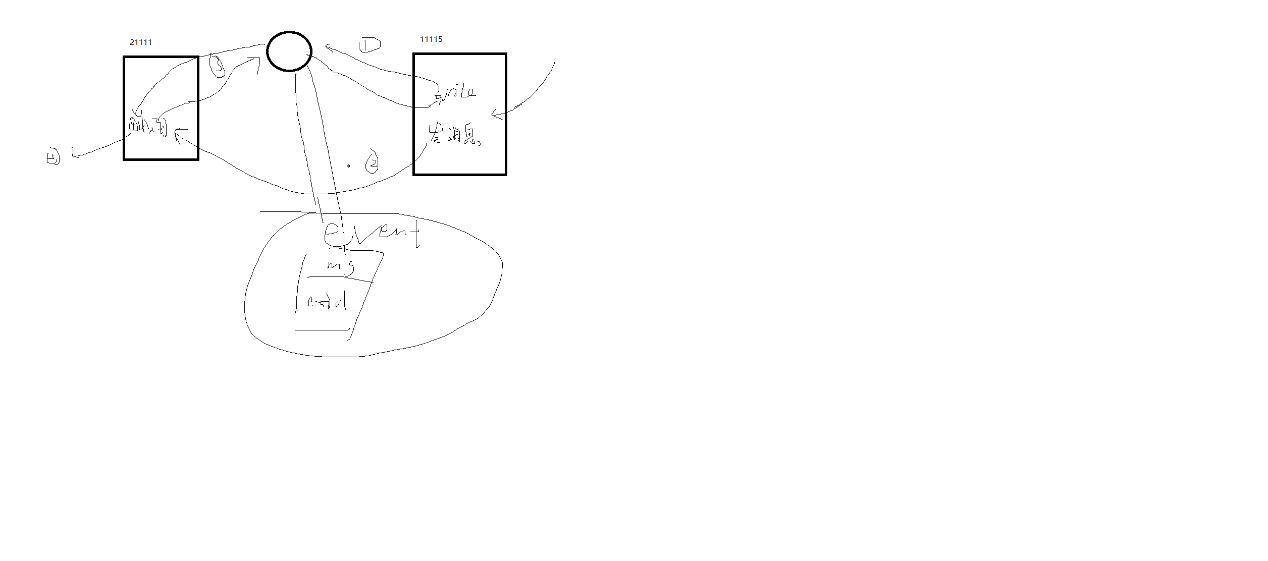
**用户可以查看班级信息,课程信息并对此做出修改.**

**13.转发客户端消息**

**由于QTcpSocket只能在本线程写消息,跨线程会导致程序崩溃,具体处理流程如图所示.**

**服务器收到客户端发送来的消息时,向目标socket发送转发消息事件通知,目标socket通过event函数获取发送来的事件内包含的消息并将其转发.**

**当一个线程需要通过另一个线程内的socket转发消息时,向另一个线程得事件队列中发送事件通知另一个线程得socket进行相关得操作.**



**6.4客户端编码过程**

**客户端图形界面主要由登陆界面,注册界面,沟通界面,成绩界面,个人信息界面组成.采用多线程方式进行数据交互和数据展示.**

**登陆界面和注册界面主要负责登陆服务器和注册账户.**

**沟通界面主要包含与教师沟通,与学生沟通界面.在沟通时只要程序没有关闭始终保留与已交流用户的聊天记录.**

**成绩界面包含考试选择界面,考试选择界面负责展示每次考试的各科成绩.**

**个人信息界面主要负责展示用户的个人信息查看以及让用户修改自己的个人信息.**

1. **登陆界面如图所示:**
   1. **用户登陆时,提供两个登陆选项,教师登陆和学生登陆.登陆成功后,客户端保存用户信息及显示个人信息.**
2. **注册界面:**
   1. **注册界面大致和登陆界面相同,只有教师登陆和学生登陆变为教师注册和学生注册.**
3. **沟通界面:**
   1. **沟通界面包含与学生沟通界面和与教师沟通界面.**
   2. **用户通过选择沟通界面的可沟通人员列表进行沟通.鼠标点击相应的列表人员会展示该人员的个 人信息,和沟通按钮.**
   3. **进行沟通的时候数据提交给服务器,由服务器进行转发.**
4. **成绩界面:**
   1. **成绩界面由学生成绩界面和老师成绩界面组成.老师只能查看老师成绩界面,学生只能查看学生成绩界面.**
   2. **学生成绩界面包含考试种类的选择,根据用户点击不同的考试种类展示该同学的全部科目成绩信息.**
   3. **老师成绩界面包含考试种类的选择,根据老师点击不同的考试种类,展示老师所教课程的所有班级.老师可以查看每个班级所有同学的成绩信息.**
5. **个人信息界面:**
   1. **老师和学生点击该按钮会展示老师和学生的个人信息,包含电话等.**

**备注:各个页面按钮索引表如下:**

|  |  |
| --- | --- |
| **索引** | **含义** |
| **0** | **退出** |
| **1** | **消息界面** |
| **2** | **联系人界面** |
| **3** | **成绩界面** |
| **4** | **个人信息界面** |
| **5** | **查看教师按钮** |
| **6** | **查看班级学生(教师端)** |
| **6** | **查看本班学生(学生端)** |
| **7** |  |

**6.沟通界面用户列表显示**

**a)每个列表为user类,user类内部包含一个QLabel对象,QImage对象,及其他必要user属性,如id,姓名等.在显示用户列表时,主窗口调用user类的show\_user方法通过user类内的坐标信息将qlabel和qimage画在主窗口的正确位置.**

* 1. **每个列表元素的标识为IDENTIFIER\_CONTACT,在点击时调用主窗口的get\_class\_students函数获取该按钮包含的信息并动态生成坐标显示在窗口中.**

**7.测试**

**7.1 数据库测试**

**(1).测试管理员登陆功能**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试用例** | **账号** | **密码** | **操作** | **结果** |
| **1** | **10001** | **beagoodstudent** | **登陆** | **失败** |
| **2** | **10001** | **beagoodstudent** | **注册** | **成功** |
| **3** | **10001** | **beagoodstudent** | **登陆** | **成功** |
| **4** | **10001** | **beagoodstuden** | **登陆** | **失败** |
| **5** | **10002** | **beagoodstudent** | **注册** | **成功** |

**7.2服务器测试**

**(1).测试服务器控制模块**

**点击开始服务器按钮调用启动服务器函数**

**点击关闭服务器按钮调用关闭服务器函数**

**(2).测试消息列表模块**

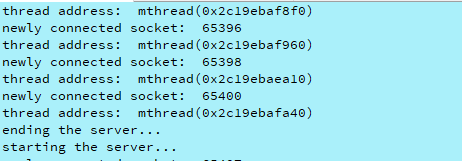
**点击消息按钮正确获取全部未处理消息,并正确设置完显示位置,按照每页10行的格式进行显示,同时根据消息类别进行分类.**

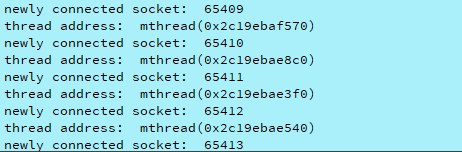
**点击消息查看消息内容.**

**(3).测试退出功能**

**点击按钮按照正确步骤关闭程序.**

**(4).测试服务器启动和关闭功能:**

****

****

**客户端测试程序:**

QApplication a(*argc*, *argv*);

MainWindow w;

thread\_m th;

th.start();

/\*connect\_to\_server cts(new QTimer());

cts.connect();\*/

while(1);

return a.exec();

**(5).服务器与客户端通信测试:**

**a). 多客户端同服务器单次收发通信:**

**多客户端测试程序连接机制:**

**客户端使用QTimer发出超时信号,每次间隔一秒,收到信号生成一个新的客户端socket线程同服务器收发消息**

**b).多客户端同服务器多次收发消息通信测试**

**(6).管理员管理数据库测试:**

**1,添加测试:**

**添加信息测试:**

**在信息内容显示页面查看信息,点击添加按钮添加新行,对信息作出修改后,点击确认修改按钮将数据提交到数据库修改.**

**2.修改测试:**

**在文本框中将所有要修改的信息修改好以后,点击确认修改按钮,向服务器发出请求.**

**3.删除测试:**

**管理员只能删除学生的课程成绩信息,教师的授课班级信息,课程信息,班级信息.**

**一次性选中所有要删除的行,之后点击删除按钮,向数据库提交该请求.**

**7.3客户端测试**

**(1).客户端登陆功能测试:**

**用户登陆账号时,服务器端收到用户登陆消息后,将用户信息打印在服务器信息页面.当服务器关闭时,客户端断开连接.服务器再次打开时,客户端再次和服务器建立连接.客户端注册账号,服务器端出现消息提示,管理员同意注册请求后,客户端可以直接进行登陆.**