

## 一、 实验目的

- 1 设计并实现一个精简的图书管理系统要求具有图书入库、查询、借书、还书、借书证管理等功能。
- 2 通过该图书馆系统的设计与实现，提高学生的系统编程能力，加深对数据库系统原理及应用的理解。

## 二、 系统需求

### 1 基本数据对象

对象名称	包含属性
书	书号, 类别, 书名, 出版社, 年份, 作者, 价格, 总藏书量, 库存
借书证	卡号, 姓名, 单位, 类别 (教师 学生等)
管理员	管理员 ID, 密码, 姓名, 联系方式
借书记录	书号, 借书证号 ,借期, 还期, 经手人 (管理员 ID)

### 2 基本功能模块

模块名称	功能描述
管理员登陆	输入管理员 ID, 密码; 登入系统 或 返回 ID/密码 错误.
图书入库	<p>1. 单本入库</p> <p>2. 批量入库 (方便最后测试)</p> <p>图书信息存放在文件中, 每条图书信息为一行. 一行中的内容如下 ( 书号, 类别, 书名, 出版社, 年份, 作者, 价格, 数量 )</p> <p>Note: 其中 年份、数量是整数类型; 价格是两位小数类型; 其余为字符串类型</p> <p>Sample: ( book_no_1, Computer Science, Computer Architecture, xxx, 2004, xxx, 90.00, 2 )</p>
图书查询	<p>要求可以对书的 类别, 书名, 出版社, 年份(年份区间), 作者, 价格(区间) 进行查询. 每条图书信息包括以下内容: ( 书号, 类别, 书名, 出版社, 年份, 作者, 价格, 总藏书量, 库存 )</p> <p>可选要求: 可以按用户指定属性对图书信息进行排序. (默认是书名)</p>
借书	<p>1.输入借书证卡号</p> <p>显示该借书证所有已借书籍 (返回, 格式同查询模块)</p> <p>2.输入书号</p> <p>如果该书还有库存, 则借书成功, 同时库存数减一。 否则输出该书无库存, 且输出最近归还的时间。</p>
还书	<p>1.输入借书证卡号</p> <p>显示该借书证所有已借书籍 (返回, 格式同查询模块)</p> <p>2.输入书号</p> <p>如果该书在已借书籍列表内, 则还书成功, 同时库存加一.</p>

	否则输出出错信息.
借书证管理	增加或删除一个借书证.

### 三、 实验环境

#### 1 数据库平台

MySQL5.7

#### 2 开发工具

Python3.8 + PyCharm Community Edition 2022.1

#### 3 数据库连接

Python 中的 PyMySQL 模块

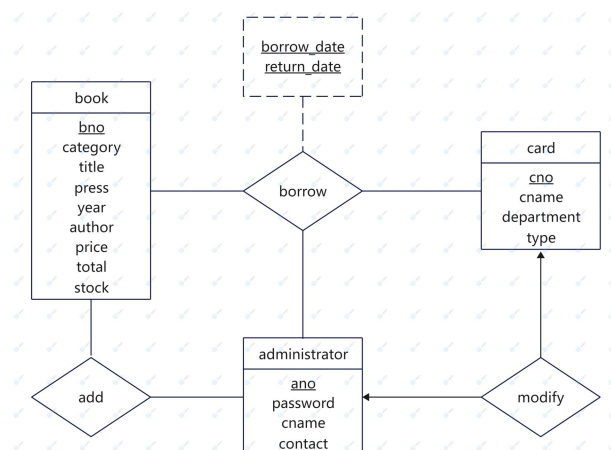
#### 4 用户图形界面

Pyside2 + Qt designer

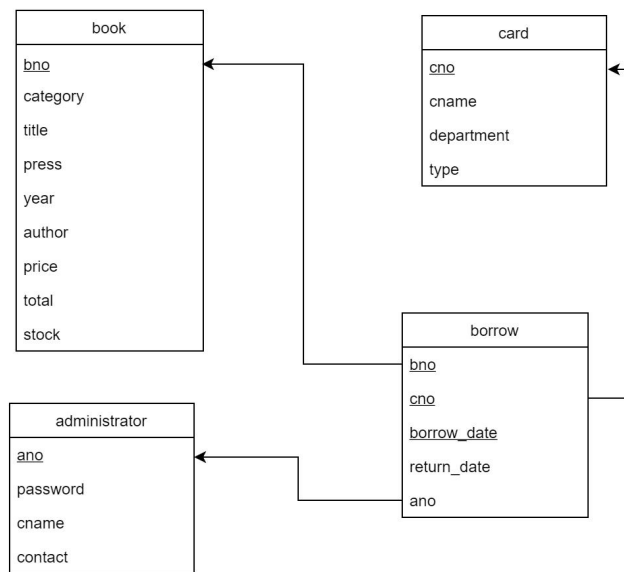
### 四、 程序源码与运行结果

#### 1 建表

##### a) E-R 模型



##### b) 数据库表结构



### c) 代码

```

● create table book
  (bno varchar(15) primary key,
  category varchar(15),
  title varchar(40),
  press varchar(30),
  year int(11),
  author varchar(20),
  price decimal(7,2),
  total int(11),
  stock int(11))engine=innodb default charset=utf8;

● create table card
  (cno varchar(7) Primary key,
  cname varchar(20),
  department varchar(40) ,
  type int(1),
  check(type in(0,1)))engine=innodb default charset=utf8;

● create table administrator
  (ano varchar(7) primary key,
  password varchar(6),
  aname varchar(20),
  contact varchar(20))engine=innodb default charset=utf8;

● create table borrow
  (bno varchar(15),
  cno varchar(7),
  borrow_date datetime,
  return_date datetime,
  ano varchar(7),
  primary key(bno,cno,borrow_date),
  foreign key(bno) references book(bno),
  foreign key(cno) references card(cno),
  foreign key(ano) references administrator(ano))engine=innodb default charset=utf8;
  
```

## 2 数据库连接

该部分采用的是 python 中的 pymysql 包，代码包含在 connect.py 中，返回一个连接后的数据库对象。

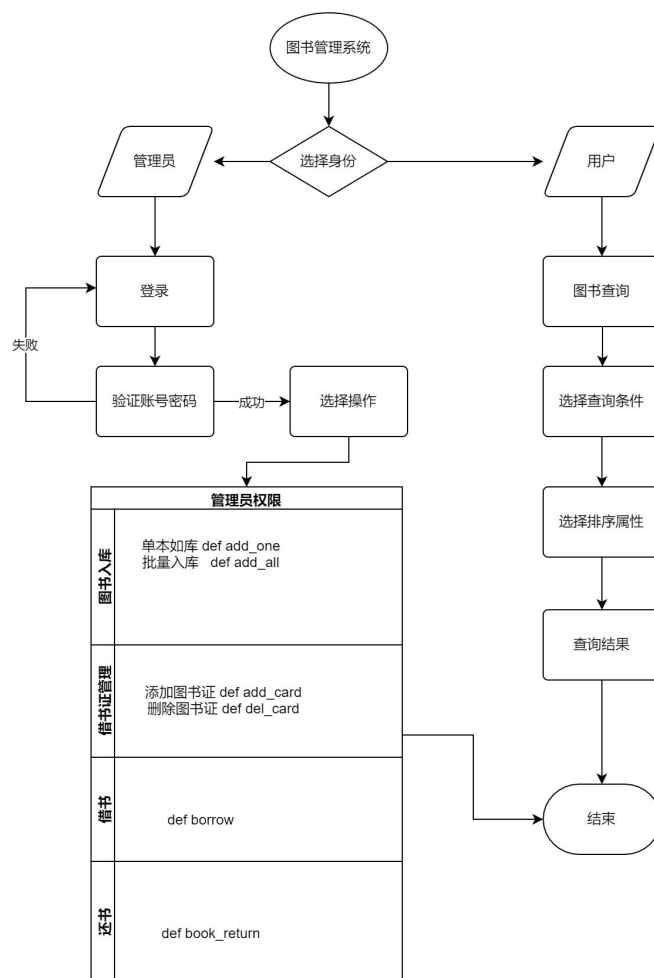
```
import pymysql #导入模块
def conn():
    db = pymysql.connect(host='localhost',
                        user='root',
                        password='3312531',
                        database='lib',
                        charset='utf8')
    return db
```

## 3 系统实现

该部分主要在 python 内实现，系统运行时只需要运行 main.py 即可。

建立 Admin 和 user 两个类，包含管理员和用户可操作的权限。

### a) 实现流程



### b) 管理员类

在 admin.py 中建立了一个 Admin 的类，定义了管理员可操作的所有方法，如登录、图书入库、借还书、借书证管理。

#### i. 登录

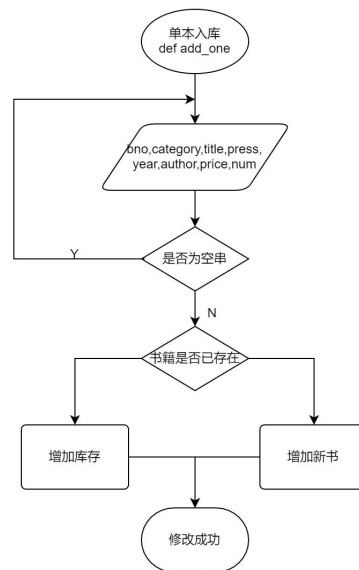
将输入的账号和密码与 administrator 中的 ano, password 相比较，相同方可进行之后的操作，跳转

至操作选择界面

```
# 登录~
def login(self, name, password):
    cursor = db.cursor() # 使用cursor方法创建一个游标对象cursor
    sql = "select * from administrator where ano = '%s' and password = '%s'" % (name, password)
    cursor.execute(sql) # 执行语句
    results = cursor.fetchall() # 获得所有记录的列表
    if len(results) == 0: # 判断数组是否为空
        return "false"
    else:
        return "true"
```

## ii. 单本入库

定义 add\_one 函数实现单本入库功能：若书籍已存在，则修改库存；若书籍不存在，则添加新书。



```
def add_one(self, new_bno, new_cate, new_title, new_press, new_year, new_author, new_price, num):
    # 类型转换
    if new_year != '' and new_price != '' and num != '' and new_bno != '' and new_title != '':
        new_year = int(new_year)
        new_price = float(new_price)
        num = int(num)
        cursor = db.cursor() # 使用cursor方法创建一个游标对象cursor
        sql = 'SELECT bno, category, title, press, year, author, price FROM lib.book;'
        cursor.execute(sql)
        result = cursor.fetchall()
        flag = 0
        # 判断该书是否已在库中
        for x in result:
            if (new_bno, new_cate, new_title, new_press, new_year, new_author, new_price) == x:
                flag = 1 # 在库中
                try:
                    # 更新库存和藏书
                    sql1 = "update lib.book set stock = stock + %d where bno = '%s';" % (num, new_bno)
                    sql2 = "update lib.book set total = total + %d where bno = '%s';" % (num, new_bno)
                    cursor.execute(sql1)
                    cursor.execute(sql2)
                    db.commit()
                    return '库存修改成功!'
                except Exception as e:
                    db.rollback()
                    return e
```

```

if flag == 0: # 不在库中
    try:
        sql = 'insert into book values("%s","%s","%s","%s","%d","%s","%f","%d","%d")' \
            % (new_bno, new_cate, new_title, new_press, new_year, new_author, new_price, num, num)
        cursor.execute(sql)
        db.commit()
        return '入库成功!'
    except Exception as e:
        db.rollback()
        return e

```

### iii. 多本入库

定义 add\_all 函数：用两次 split 方法提取书籍信息后重复调用 add\_one

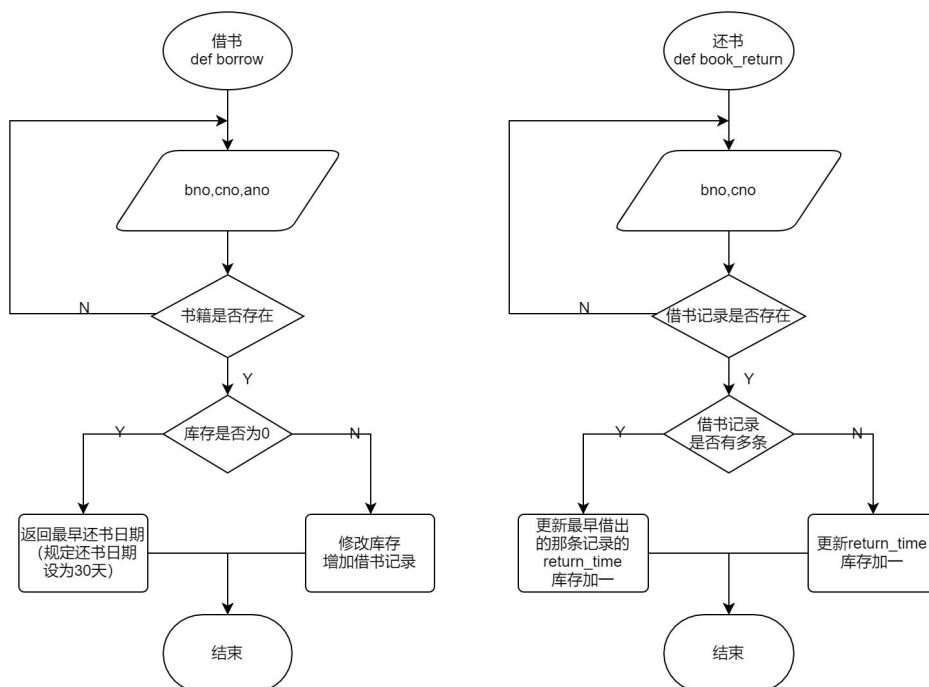
```

# 批量入库
def add_all(self, str):
    if str != '':
        try:
            # 用回车分隔两条记录
            str_tup = str.split(sep="\n")
            for rec in str_tup:
                if rec != '':
                    rec_list = rec.split(sep=",")
                    # 首尾去掉括号
                    rel = self.add_one(rec_list[0][1:], rec_list[1], rec_list[2], rec_list[3], rec_list[4],
                                      rec_list[5], rec_list[6], rec_list[7][0:-1])
                    # print(rel)
            return 1
        except Exception as e:
            return e

```

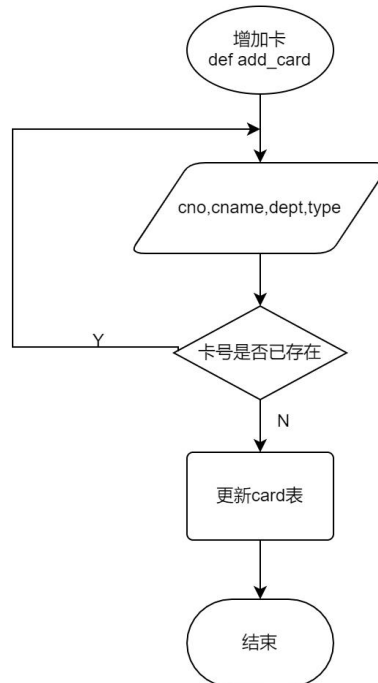
### iv. 借还书

定义 borrow 和 book\_return 函数：由于借还书时要显示该借书证所有已借书籍（本人的理解是已借未还的书籍，不然好像没什么意思），定义了一个 query 函数用来返回已借书籍，其中已借未还的书籍是指数据库中 bno=“输入的卡号”且 return\_date 为 null 值的记录。之后定义了 borrow 和 book\_return 两个函数来实现借还书记录的更新。其中借还书的时间采用 datetime 包的 datetime.now() 函数获取现在时间。还书时若借了多本同样的书未还则归还最早借出的那本。



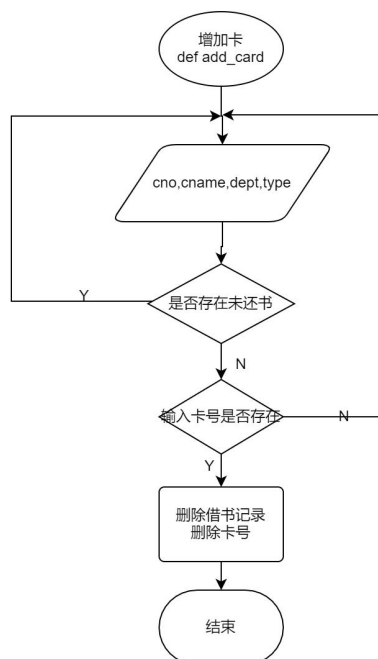
v. 增加借书证

定义 add\_card 函数：判断输入的卡号是否违反了主键约束，若不违反，则在 card 表中增加一条记录。



vi. 删除借书证

定义 del\_card 函数：首先在 borrow 中查询该卡号是否存在未还书籍，即 return\_date 为 null 值，存在则无法删除；之后查询卡号是否存在，存在的话先删除 borrow 中的借书记录，之后再删除 card 表中的卡号信息，因为 borrow 中的 cno 外键引用了 card 表中的 cno。



### c) 用户类——图书查询和筛选

#### i. 图书查询

search 函数可以实现对任意属性的查询，由于 sql 语句的执行语句是字符串，所以可以通过字符串的连接操作实现个别属性的查询，若输入值非空，则在语句中加入查询条件。

```
def search(self, bno, cate, title, press, ymin, ymax, author, pmin, pmax):
    try:
        cursor = db.cursor() # 使用cursor方法创建一个游标对象cursor
        sq = "select bno,category,title,press,year,author,price,total,stock from lib.book where 1=1 "
        if bno != '':
            sq = sq + " and bno='%s'" % bno
        if cate != '':
            sq = sq + " and category='%s'" % cate
        if title != '':
            sq = sq + " and title='%s'" % title
        if press != '':
            sq = sq + " and press='%s'" % press
```

#### ii. 图书筛选

rerange 函数可以实现用户所选择属性的升序或降序排序，用 sorted 方法实现对元组的排序比在 mysql 中实现排序要方便得多。

```
def rerange(self, arrange2, arrange1, result):
    if len(result) == 0:
        pass
    else:
        if arrange2 == '升序':
            if arrange1 == '书号':
                result = sorted(result, key=lambda tup: tup[0])
            elif arrange1 == '类型':
                result = sorted(result, key=lambda tup: tup[1])
            elif arrange1 == '书名':
                result = sorted(result, key=lambda tup: tup[2])
            elif arrange1 == '出版社':
                result = sorted(result, key=lambda tup: tup[3])
            elif arrange1 == '年份':
```

## 4 前端设计和展示

### a) 前端设计

前端设计采用的是 pyside2 并结合其自带的 Qt designer。在 python 中实现了对 Qt 生成的 ui 后缀文件的动态调用。对每一个窗口创建一个类，此时窗口的所有控件都变成该类的属性，通过窗口控件的 clicked 方法实现不同窗口之间的跳转以及操作函数的链接。

```
# 主界面~
class Hello(QWidget, Ui_Form):

    def __init__(self):
        super().__init__()
        # 从文件中加载UI定义
        qfile_stats = QFile("ui/hello.ui")
        qfile_stats.open(QFile.ReadOnly)
        qfile_stats.close()
        # 从 UI 定义中动态 创建一个相应的窗口对象
        # 注意：里面的控件对象也成为窗口对象的属性了
        # 比如 self.ui.button , self.ui.textEdit
        self.ui = QUiLoader().load(qfile_stats)

        self.ui.admin.clicked.connect(self.jumpad)
```

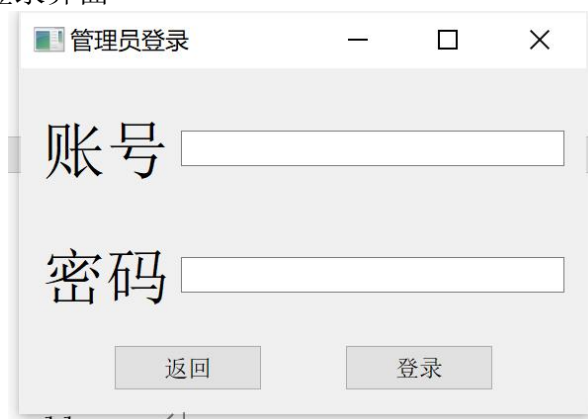
### b) 前端界面展示



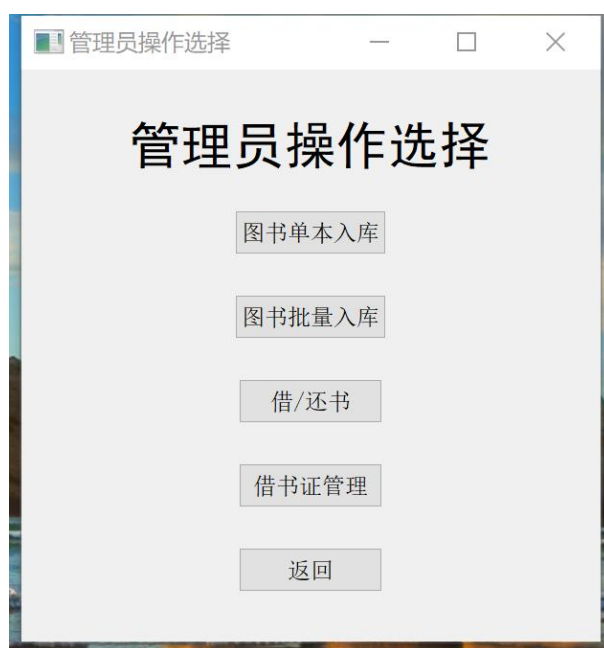
i. 登录界面



ii. 管理员登录界面



iii. 管理员操作选择



iv. 图书单本入库



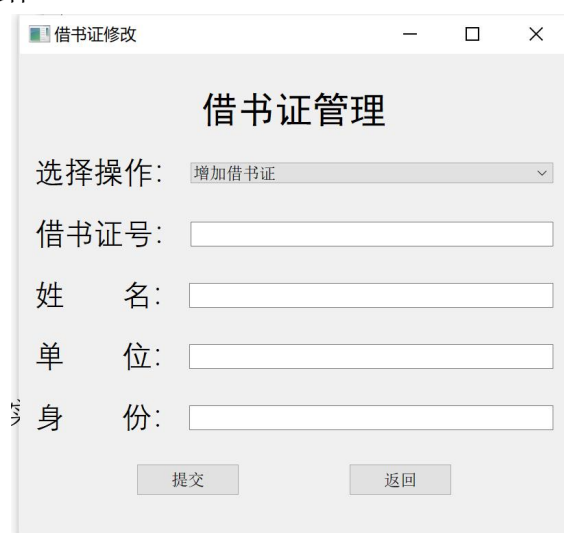
A screenshot of a software window titled "单本图书入库" (Single Book Entry). The window contains a form with the following fields: "书号" (Book Number), "类别" (Category), "书名" (Book Name), "出版社" (Publisher), "年份" (Year), "作者" (Author), "价格" (Price), and "数量" (Quantity). Each field has a corresponding text input box. At the bottom of the form, there are two buttons: "提交" (Submit) and "返回" (Return).

v. 借还书



A screenshot of a software window titled "借/还书" (Borrow/Return Book). The window contains a form with the following fields: "借/还书" (Borrow/Return Book) with a dropdown menu set to "借书" (Borrow), "借书证号" (Borrower ID), "书号" (Book Number), and "管理员账号" (Admin Account). There are "查询" (Query) and "确认" (Confirm) buttons next to the "借书证号" and "书号" fields respectively. Below the form, there is a large empty box labeled "已借书籍" (Borrowed Books). At the bottom, there is a "返回" (Return) button.

vi. 借书证操作



A screenshot of a software window titled "借书证管理" (Borrower ID Management). The window contains a form with the following fields: "选择操作" (Select Operation) with a dropdown menu set to "增加借书证" (Add Borrower ID), "借书证号" (Borrower ID), "姓名" (Name), "单位" (Unit), and "身份" (Identity). There are "提交" (Submit) and "返回" (Return) buttons at the bottom.

## vii. 图书查询



## 五、 实验总结

- 1 在数据库的创建中，由于 borrow 表是由 book, card, administrator 的联系生成的弱实体，其表中还需要包含这 3 个表的主键 bno, cno, ano。
- 2 在删除 card 表中数据时注意 borrow 表的外键约束，要先删掉 borrow 表中的相关数据，再删 card 表中数据，或者采用级联删除的方式。
- 3 数据库连接中返回的是一个数据库对象，要实现对数据库的操作还需要用数据库方法创建一个游标对象 cursor。用 Python 连接数据库后，数据库的数据处理功能相对弱化了，在读取数据库的数据后，很多操作可以在 Python 中完成，比如排序、字符串的处理等，在 Python 中处理要方便得多。
- 4 在前端设计的过程中，我尝试了两种方法，一种是利用安装 pyside2 时自带 pyuic 将 qt designer 做好的 ui 图形界面转为 py 后缀的代码文件，然后在代码上添加按钮的 clicked 槽函数，链接其他自定义函数，比如实现页面的跳转等（网上的教程也大多是采取这种方式）。但采用这种方式的话，一旦我修改图形界面，只能将 ui 文件重新转换一次，原有的编辑就不会被保留，非常麻烦。因此我采用了第二种方式——ui 文件的动态调用：将每个窗口都定义成一个类，在类的初始化函数中打开相应 ui 文件，将 ui 文件中的所有控件都变成了对象的一个属性。这样可以实现前端和后端设计上的分离，将前端的修改实时反映到后端的调用上。