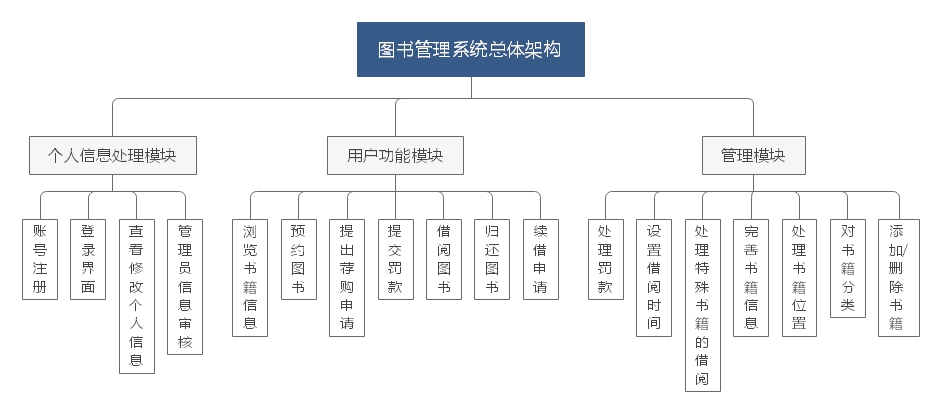
# 3.1 总体系统功能设计

本系统为图书管理系统，新用户通过填写个人信息登陆界面之后给管理员审核（老用户直接验证身份后可以进入）。完成认证后可以进行一系列操作，包括预约借书，申请续借，查看书籍信息，提交罚款等。管理员则对于用户的借阅情况对用户的信息、操作以及图书的信息进行管理。

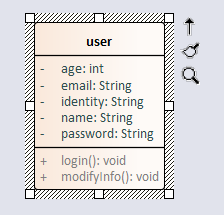


如图，大致可将系统分为三大块：个人信息处理模块包括用户的注册与登录，个人信息的完善等等以及管理员对用户的信息处理；功能模块包括用户的借阅功能（预约，续借，提交罚款等）以及管理员的处理罚款等操作；书籍管理模块主要包括管理员对于书籍信息的维护。

1. 信息处理模块

（1）这里为用户提供一个进入系统的界面用户完成信息的验证后进入。如果用户第一次登录时，需要完成基本的信息填写方可进入。验证失败的用户不得进入。

（2）用户进入系统后，可以修改个人信息。修改后的信息经由管理员审核后存入数据库。



1. 用户功能模块

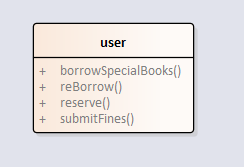
用户登录成功后会进入功能界面，这是系统的主要界面。

用户功能模块是图书管理系统的主要模块，主要内容为用户的功能：申请预约、借阅、续借、归还、提交罚款等。

（a）用户预约书籍：用户首先检查自己有没有尚未提交的罚款，如果有先提交罚款；之后选择时间预约书籍，没有与当前数据冲突（已经存在的预约时间）则提示预约成功，否则预约失败。

（b）用户申请珍贵书籍的借阅：首先会发送请求进行身份的核实，核实标准（identity）为是否为大学生或是老师，管理员处理信息后返回结果；其次检查信用情况，同时如果有罚款未提交应先提交罚款；身份满足要求后，选择预约的时间，时间如果没有冲突的话借阅成功。

（c）用户续借图书：用户发送续借的请求；管理员首先检查是否超期，超期则提醒先提交罚款；设置当前时间为新的借阅时间，重新计算应归还日期。



1. 书籍管理模块

该模块主要是由管理员用来实现功能：对用户请求的操作响应与对书籍进行的管理。对于用户的身份进行了限制（管理员）。

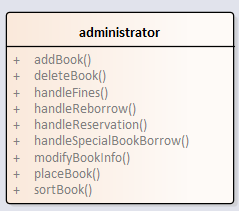
1. 处理用户借阅珍贵图书的请求：核实标准（identity）为是否为大学生或是老师，管理员处理信息后返回结果；其次检查信用情况，同时如果有罚款未提交提示先交罚款；身份满足要求后，处理申请预约的时间，时间如果没有冲突的话借阅成功。

（b）处理续借的请求：管理员首先检查是否超期，超期则提醒先提交罚款；设置当前时间为新的借阅时间，重新计算应归还日期。

（c）修改书籍的信息：管理员对书籍的信息（分类、位置、编号）进行修改。

（d）处理书籍的位置：改变书籍的位置（房间号+书架号）。

（e）对于书库中的书籍，进行增加/删除等工作。



3.2 体系结构设计

3.2.1基本体系结构设计

该系统的适用场景为一个校园的图书馆，对象为该校园的师生及其他工作人员。所需要的功能包括借还书、荐购、预约等一般的图书馆所具有功能，方便图书馆管理员的管理以及师生对图书馆的利用。

本系统主要采用分层次结构进行架构。该结构有以下特点：升级和维护方式简单，对功能的修改便捷，减少维护成本。支持扩展与重用。

因为系统同时使用人数较少，信息库中存储的信息都为纯文本形式。考虑成本问题以及前期对需求的分析，决定基于Qt架构与SQLite来构建该分层次图书管理系统。

从上到下依次为视图层、控制器层、模型层和数据库层。如下图所示：

视图层

控制层

模型层

数据库层

视图层：提供图书馆系统的基本视图显示，与与用户的交互界面，将用户对应用程序的操作请求下传给其他层处理。为用户显示模型信息。

控制层：按预定的业务逻辑处理视图层提交的请求。

模型层：将模型的状态转交视图层或者控制层，以更新视图或控制信息，提供简化的SQL语句，及时更新修改数据库数据，提供数据持久化。

数据库层：用户信息数据库的建立与管理。

3.2.2质量属性的满足

|  |  |  |
| --- | --- | --- |
| 目标 | 实现方式 | 所采用战术 |
| 安全性 | 系统必须限制用户对数据的访问范围，对于没有权限的用户防止其对非自己可见的信息的访问，系统还应提供必须的身份检测与授权手段。 | 身份验证  身份授权  数据机密性 |
| 可用性 | 对于用户的非法访问与操作，需要及时制止防止对系统的损害，系统还应具有一定的故障恢复能力。 | 异常检测及程序终止  故障恢复性 |
| 可维护性 | 系统实现时分层实现，控制不同层次之间的交流，将错误一定程度限制在本层中。 | 分层次 |
| 可移植性 | 采用跨平台框架Qt对系统进行架构，减少了系统移植的成本。 | 跨平台框架 |

3.2.3 模型层架构分析

因为本系统所采用的功能都与中心数据库相关，所以在模型层中通过Qt DB接口将所有功能模块与数据库之间采用仓库模式连接。以此方便模块的添加修改以及部分模块之间的数据共享。

模型层架构如下：

系统用户信息库、

图书信息库

个人信息处理模块

管理模块

用户功能模块

Qt DB接口及数据持久化模块

3.2.4业务逻辑层架构分析

业务逻辑层决定了系统的实用性，根据Qt框架的特点，我们决定对业务逻辑层采用MVC风格进行架构。

及当用户使用该系统的桌面应用程序时，在用户在GUI上发出的相关信号时，采用Qt默认的Delegate作控制器（为Qt预定义组件，实现时可以不用关注），QModel等为主的功能类作模型层，封装核心数据与部分功能。

业务逻辑层架构如下：

桌面应用程序

Qtui

Delegate

QModel等

数据库

3.2.5整体架构分析

由上述分析，整体架构如下：

数据库

用户请求

桌面应用程序

Qt ui

Delegate

个人信息处理模块

管理模块

用户功能模块

QModel及数据持久化模块

Qt框架

Model