ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN LẬP TRÌNH

ỨNG DỤNG QUẢN LÝ THƯ VIỆN

GIÁO VIÊN HƯỚNG DẪN:

ThS Phan Thanh Tao

ThS Phan Chí Tùng

SINH VIÊN THỰC HIỆN:

Họ và tên: Nguyễn Huy Tưởng; Lớp: 19TCLC_DT4; Nhóm: 14A

Họ và tên: Hồ văn Vy; Lớp: 19TCLC_DT4; Nhóm: 14A

Đà Nẵng, Ngày 26 tháng 12 năm 2020

LỜI MỞ ĐẦU

Lời nói đầu tiên em xin phép được cảm ơn quý thầy cô hướng dẫn: Ths Phan Thanh Tao, ThS Phan Chí Tùng đã hỗ trợ chúng em rất nhiều trong đồ án PBL2 này. Xin được cảm ơn những người bạn đã cùng hỗ trợ nhau trong lúc cùng làm đồ án. Cảm ơn những tác giả của các bài viết mà chúng em đã tham khảo trên internet.

Đồ án PBL2 là đồ án đầu tiên giúp chúng em tiếp xúc với cách thức tạo ra một phần mềm hoàn thiện, đầy đủ các tính năng. Vì vậy trong đồ án lần này đã giúp chúng em có thêm được rất nhiều kĩ năng về cách thức lên ý tưởng, lập trình, thiết kế Hướng đối tượng, thiết kế Cơ sở dữ liệu và thiết kế thuật toán. Tuy vậy, do đây là đồ án lập trình thứ 2 tiếp xúc với những lĩnh vực mới nên có thể còn nhiều sai sót, Kính mong quý Thầy/Cô châm chước bỏ qua. Chúng em xin cảm ơn!

MỤC LỤC

DANH MỤC HÌNH VẼ

1. GIỚI THIỆU ĐỀ TÀI

1.1. Tên đề tài

ỨNG DỤNG QUẢN LÝ THƯ VIỆN

1.2. Lý do chọn đề tài

Hiện nay, đọc sách đang là một thói quen tốt nhưng đã dần mai một ở thế hệ trẻ hiện nay. Theo Sở Thông tin - Truyền thông TPHCM, trung bình mỗi người Việt Nam đọc 1 cuốn sách mỗi năm. Các trường học cũng đang khuyến khích học sinh đọc sách bằng việc xây dựng những thư viện sách cho học sinh tự do vào đọc và mượn sách. Đi kèm với việc đó là vấn đề cần quản lý việc mượn, trả sách sao cho hiệu quả, tránh mất mát cho thư viện cũng như cho nhà trường. Chính vì vậy, trong đồ án PBL2 lần này chúng em đã chọn đề tài **Ứng dụng quản lý thư viện**, với mong muốn có cơ hội tìm hiểu, học hỏi, áp dụng những kiến thức đã học để xây dựng một ứng dụng giúp nhà trường có thể quản lý việc mượn trả sách cũng như quản lý kho sách một cách hiệu quả.

1.3. Mục đích của đề tài

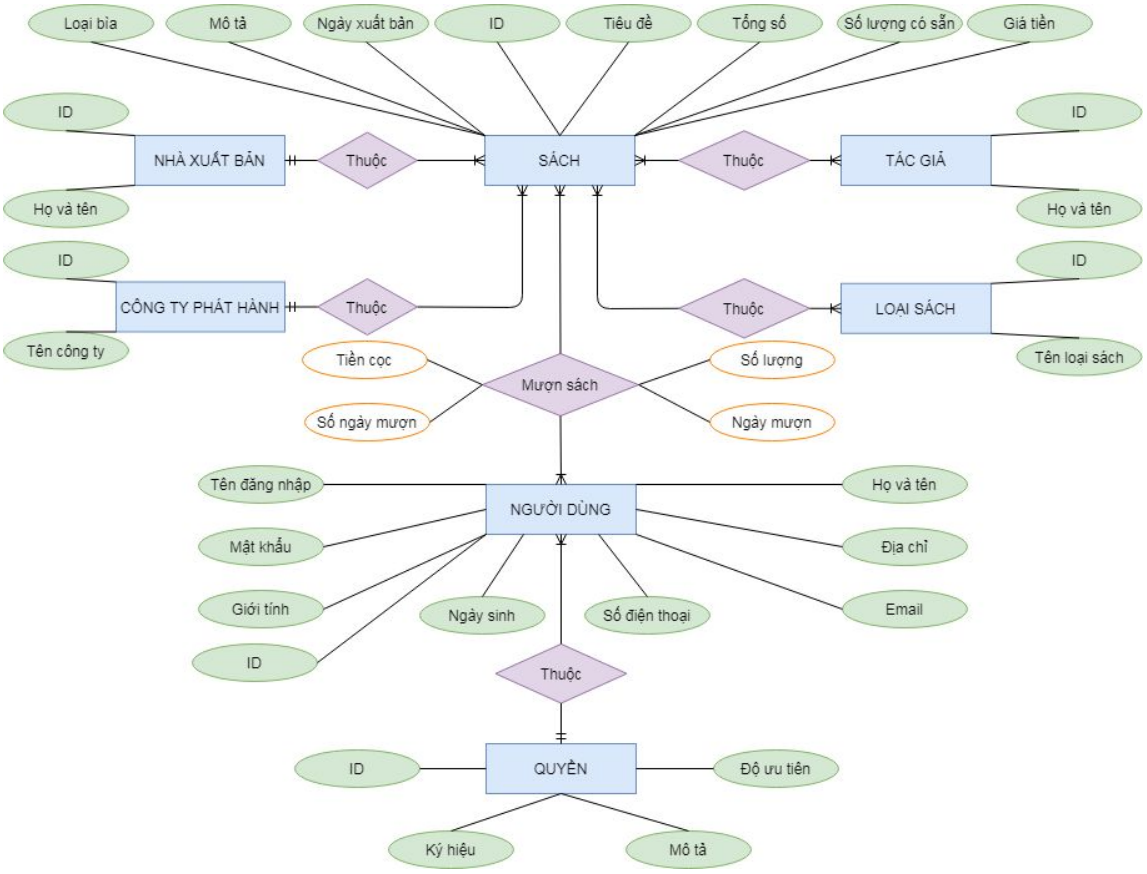
Với đề tài **Ứng dụng quản lý thư viện**, chúng em mong muốn xây dựng một ứng dụng giúp nhà trường quản lý sách, mượn trả sách tại thư viện một cách hiệu quả. Đảm bảo tính thuận tiện, dễ sử dụng cho người quản lý và tính tiện lợi cho những người mượn sách. Xây dựng ứng dụng hoạt động trơn tru, không phát sinh lỗi trong quá trình hoạt động thực tế, không bị mất mát về dữ liệu.

2. THIẾT KẾ CƠ SỞ DỮ LIỆU

Trong đồ án lần này chúng em sử dụng hệ quản trị cơ sở dữ liệu SQL Server. Là hệ quản trị cơ sở dữ liệu ổn định, tính bảo mật cao, thuận tiện để áp dụng sau quá trình học tập.

2.1. Thiết kế mô hình ER

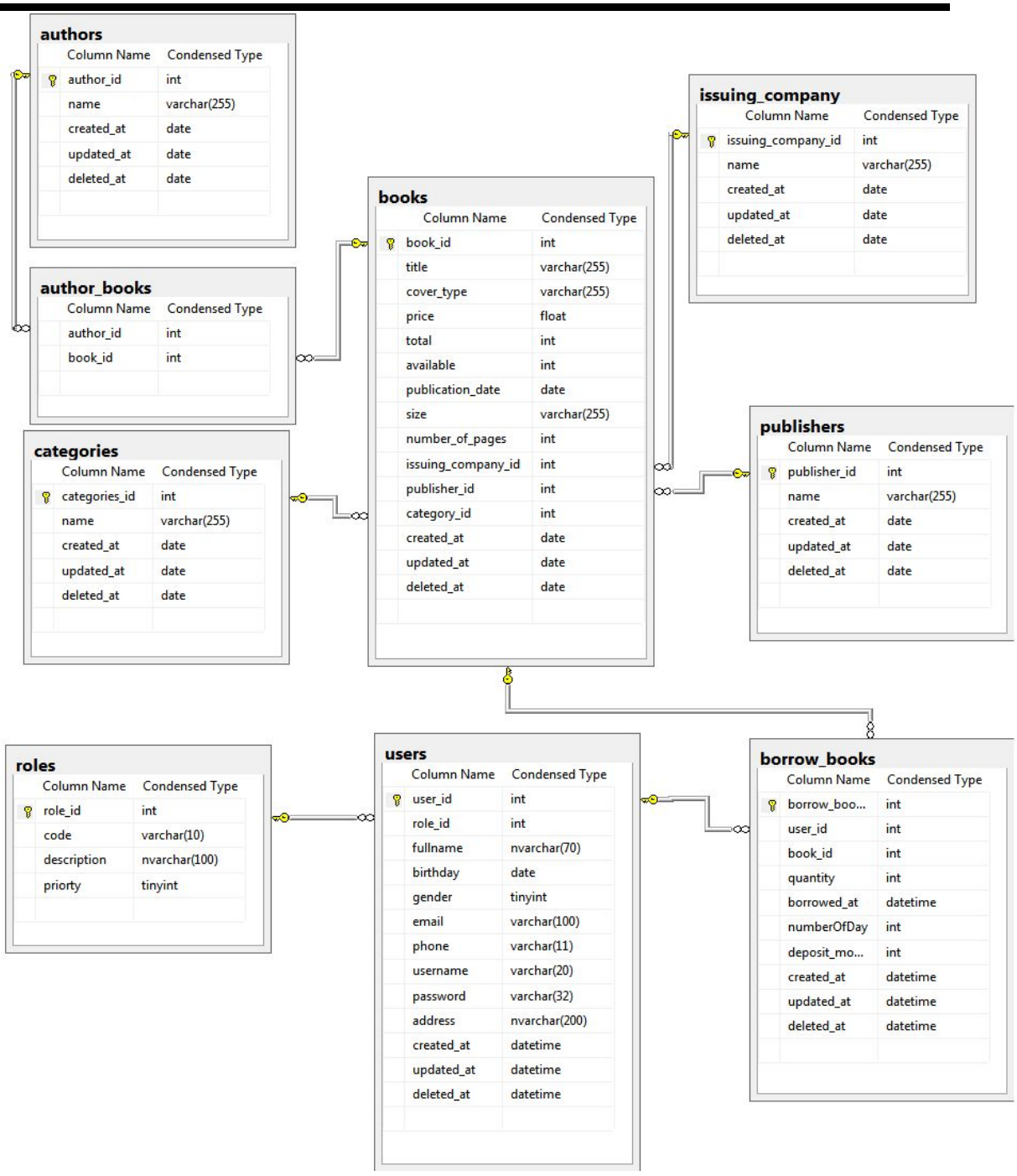
Bước thiết kế mô hình ER (Entity - Relationship) hết sức quan trọng. Mục tiêu của mô hình ER trong quá trình thiết kế cơ sở dữ liệu đó là phân tích dữ liệu, xác định các đơn vị thông tin cơ bản cần thiết của tổ chức, mô tả cấu trúc và mối liên hệ giữa chúng. Mô hình ER giúp chúng ta hình dung được bao quát cơ sở dữ liệu gồm những thực thể nào và quan hệ giữa những thực thể đó. Trước hết chúng em đã xác định được cơ sở dữ liệu gồm những thực thể **Sách, Tác giả, Nhà xuất bản, Mượn trả sách, Người sử dụng, Quyền, Lịch sử.** Sau đó tiến hành xây dựng các thuộc tính cho thực thể và quan hệ giữa các thực thể đó, được thể hiện qua mô hình sau:



Hình 1. Mô hình ER

2.2. Thiết kế các bảng chính

Sau khi thiết kế mô hình ER, chúng ta đã dần hình dung được những thực thể chính và quan hệ cấu thành cơ sở dữ liệu. Từ đó chúng ta có thể thiết kế ra những bảng thể hiện chi tiết từng thực thể:



Hình 2. Thiết kế chi tiết cơ sở dữ liệu

2.4. Thiết kế Trigger

Để cơ sở dữ liệu hoạt động trơn tru, đảm bảo tính đúng đắn và không bị mất mát dữ liệu, ngoài việc xử lý ở chương trình, chúng ta cũng cần phải xây dựng những Trigger cho cơ sở dữ liệu ứng với các sự kiện Insert, Update, Delete.

2.4.1. Sự kiện Insert:

Đối với bảng users, khi tạo một bảng ghi mới, cơ sở dữ liệu sẽ tự động kiểm tra các trường đã ràng buộc trong lúc tạo bảng với từ khóa check. Một số trường được kiểm tra:

- **fullname** NVARCHAR(70) NOT NULL => Ràng buộc trường fullname không được là NULL
- **gender** TINYINT check(gender = 0 OR gender = 1 OR gender = 2) => Ràng buộc giới tính chỉ có ba trạng thái 0, 1, 2 ứng với Nam, Nữ, Không xác định
- **username** VARCHAR(20) check(LEN(username) > 5) => Ràng buộc username luôn dài hơn 5 ký tự
- **password** VARCHAR(32) check(LEN(password) > 10) => Ràng buộc mật khẩu luôn dài hơn 10 ký tự

Đối với bảng borrow_books cần tạo trigger để kiểm tra một số ràng buộc sau:

- Kiểm tra số sách có sẵn có đủ để mượn hay không
- Nếu có đủ sách thì trừ số sách tại trường books.available số sách mượn tương ứng

2.4.2. Sự kiện Update

Hầu hết các bảng đều có trường updated_at được cập nhật vào thời điểm hiện tại nhờ Trigger Update

2.4.3. Sự kiện Delete

Hầu hết các bảng dữ liệu quan trọng đều được xóa mềm chứ không xóa ngay khỏi cơ sở dữ liệu để tránh mất mát dữ liệu quan trọng (Sẽ được giải thích kĩ hơn tại mục sau)

Tại bảng Roles, sau khi xóa đi một quyền bất kỳ, Trigger sẽ thực hiện công việc:

- Tìm kiếm quyền thấp nhất trong bảng roles
- Chỉnh sửa những tài khoản thuộc quyền vừa bị xóa: Chỉnh sửa thành quyền thấp nhất

Công việc này giúp người dùng không bị mất tài khoản khi bị xóa đi một quyền.

Tại bảng borrow_book, sau khi xóa đi một quyền bất kỳ, Trigger sẽ thực hiện công việc cập nhật lại số lượng sách có sẵn tại bảng Books.

3. THIẾT KẾ THUẬT TOÁN

3.1. Phát biểu bài toán

Trong việc xây dựng ứng dụng phát sinh bài toán **Cần phải sắp xếp lại dữ liệu** để thuận tiện cho việc quản lý dữ liệu.

- **Input: Danh sách các đối tượng, Tiêu chí sắp xếp**
- **Output: Danh sách các đối tượng đã được sắp xếp theo tiêu chí sắp xếp**

3.2. Phân tích thuật toán

3.2.1. Giới thiệu

Hiện nay, sắp xếp là một bài toán phổ biến và cũng đã có rất nhiều giải thuật sắp xếp được công bố. Tuy nhiên, Quick sort (Sắp xếp nhanh) vẫn là thuật toán có độ phổ biến cao nhờ vào tốc độ xử lý và độ phức tạp trung bình thuộc hàng thấp.

3.2.2. Ý tưởng

Giả sử chúng ta đang cần sắp xếp tăng dần. Ý tưởng của thuật toán Quick sort là thực hiện việc chọn một phần tử làm chuẩn, sau đó thực hiện biến đổi trên danh sách sao cho các phần tử bên trái phần tử đó nhỏ hơn phần tử chuẩn và các phần tử bên phải phần tử đó lớn hơn phần tử chuẩn. Bây giờ chúng ta có một phần tử đã nằm đúng vị trí bởi vì nó lớn hơn các phần tử bên trái và nhỏ hơn các phần tử bên phải. Tiếp tục lặp quá trình trên đối với danh sách bên trái và bên phải phần tử chuẩn cho đến khi danh sách còn một phần tử.

Chúng ta có nhiều cách để chọn phần tử làm chuẩn: Chọn phần tử đầu danh sách, cuối danh sách, giữa sách sách hoặc chọn ngẫu nhiên. Việc chọn phần tử làm chuẩn củng ảnh hưởng rất nhiều đến tốc độ thực hiện, tuy nhiên máy tính không thể biết được vai trò của từng phần tử, nên chúng ta có thể chọn một cách cố định, trong dự án này thuật toán được triển khai với việc chọn phần tử cuối cùng làm chuẩn.

3.3. Triển khai thuật toán

```
QuickSort(arr[], low, high, compare(a, b)):
Parameter:
 arr[]: Danh sách cần sắp xếp
 low: Chỉ số đầu tiên đoạn cần sắp xếp
 high: Chỉ số cuối cùng đoạn cần sắp xếp
  compare(a, b): Hàm thể hiện tiêu cho sắp xếp, trả về true
nếu a đứng trước b
Return:
 Không có giá trị trả về
Result:
 Danh sách arr[] đã được sắp xếp theo tiêu chí đầu vào
BEGIN
 If (low < high):
   pivot = arr[high]
   left = low
   right = high - 1
   While (True):
     While (left <= right and compare(arr[left], pivot)):
left++
     Endwhile
     While (left <= right and not compare(arr[right], pivot)):
       right--
     Endwhile
     If (left >= right):
       break
     Endif
     Swap(arr[left], arr[right])
     left++
     right--
   Endwhile
   Swap(arr[left], arr[high])
   p = left
   QuickSort(arr, low, p - 1)
   QuickSort(arr, p + 1, high)
 Endif
END
```

3.4. Độ phức tạp

Giả sử trường hợp trung bình, cứ mỗi lần chọn ta chọn được phần tử mà vị trí đúng của nó là giữa danh sách cần sắp xếp. Theo như thuật toán ở trên ta có phép biến đổi cơ bản là số lần dịch chuyển biến **left** và **right** để tìm vị trí hoán đổi. Nếu phần tử chuẩn là chính giữa thì tổng cộng tại bước đó **left** và **right** cần dịch chuyển n lần. Nên ta có hệ thức truy hồi:

$T(1) = 0$

$T(n) = 2*T(n/2) + n$

$T(n) = 2*[2*T(n/4) + n/2] + n = 4*T(n/4) + 2*n$

Có dạng $T(n) = 2k*T(n/2k) + k*n$

Cho đến khi $n = 1 => n = 2k$

Suy ra $T(n) = n*\log(n)$

4. CẤU TRÚC DỮ LIỆU

Chương trình sử dụng chủ yếu 2 cấu trúc dữ liệu: danh sách đặc và danh sách liên kết đôi (2 cấu trúc này được sử dụng trong các lớp Listt, ArrayListt, LinkedListt sẽ được trình bày và giải thích rõ ở mục thiết kế hướng đối tượng)

5. CHƯƠNG TRÌNH VÀ KẾT QUẢ

5.1. Tổ chức chương trình

5.1.1. Công cụ sử dụng

Ngôn ngữ lập trình: C++

Thư viện hỗ trợ giao diện: Qt

Phần mềm lập trình: Qt Creator

Phần mềm hỗ trợ xây dựng cơ sở dữ liệu: SQL Server 2014 Management Studio

Công cụ thiết kế: DrawIO (draw.io)

Công cụ quản lý phiên bản: Git & Github

5.1.2. Thiết kế hướng đối tượng

Việc thiết kế hướng đối tượng gồm ba việc chính bao gồm: Xây dựng các lớp để ánh xạ dữ liệu từ cơ sở dữ liệu (Entity Class), xây dựng các lớp hỗ trợ (Listt, LinkedListt, ArrayListt, Sort, DotEnv,...) và các lớp điều khiển (View, Service, Repository)

Các lớp ánh xạ cơ sở dữ liệu (Entity Class) gồm các lớp User, Book, Author, Publisher, IssuingCompany, Category, Roles, … Dưới đây là minh họa một số lớp vừa kể ở trên:

Class User
private: int user_id; String fullname; Date birthday int gender String email String phone String username String password Role role // has one String address Date created_at Date updated_at Listt<BorrowBook> *borrowList // has many **public:** User(); User(int, String, Date, int, String, String, String, String, Role, String, Date, Date) ~User()

class Roles	class BorrowBook
private: int role_id int priorty String code String description **public**: Role(int = -1, int = -1, String = "", String = "") ~Role()	**private**: Date borrowed_at; int quantity; Book book // has one int num_of_day; int deposit_money; QDate updated_at; int id; **public**: BorrowBook(); BorrowBook(int, QDate, int, Book, int, int, QDate); ~BorrowBook();

Các lớp cơ bản:

class Sort<T>
Sắp xếp danh sách
public: Sort() ~Sort() static bool compare(T, T) static int partition (Listt<T>*, int, int, bool func(T, T)) static void quickSort(Listt<T>*, int, int, bool func(T, T) = nullptr) static void swap(T&, T&)

class Password
Cung cấp phương thức để Hash mật khẩu và so sánh mật khẩu, hỗ trợ thêm tham số salt
private: String password **public:** Password(String = "") ~Password() String hashMd5(String = "") bool compare(String, String = "")

class DotEnv	class DatabaseConection
Lấy các biến môi trường, hoặc đọc từ file nếu đang trong quá trình phát triển phần mềm	Là lớp phụ trách việc kết nối, duy trì luôn chỉ có 1 kết nối với cơ sở dữ liệu thông qua Singleton Pattern
private: static DotEnv* _instance; String *name; String *value; int numOfVar; bool haveLoadFromFile; DotEnv() **public:** static DotEnv* initDotEnv() DotEnv(bool=true, String="./.env") ~DotEnv(); String operator[](const std::string&)	**private:** static DatabaseConnection* _instance; QSqlDatabase conn; QSqlQuery *query; DatabaseConnection() **public:** ~DatabaseConnection() static DatabaseConnection* initDatabaseConnection() QSqlQuery* getQuery()

Mô tả lớp Listt cũng như các lớp triển khai LinkedListt và ArrayListt, cũng như lớp phụ thuộc NodeLinkedListt

class Listt<Entity>	class LinkedListt<Entity> : public Listt<Entity>
Là một lớp chỉ bao gồm các hàm thuần ảo (interface) cung cấp giao diện cho các lớp LinkedListt và ArrayListt implement	Là lớp kế thừa, implement từ lớp thuần ảo Listt. Lớp này sử dụng cấu trúc dữ liệu danh sách liên kết đơn

public:
Listt();
virtual ~*Listt*();
virtual int *indexOf*(const E) = 0;
virtual int *lastIndexOf*(const E) = 0;
virtual bool *add*(const E) = 0;
virtual bool *add*(const int &, const E) = 0;
virtual int *getSize*() = 0;
virtual E *get*(const int &) = 0;
virtual bool *remove*(const E) = 0;
virtual bool *removeAt*(const int &) = 0;
virtual bool *isEmpty*() = 0;
virtual void *clear*() = 0;
virtual bool *contains*(const E) = 0;
virtual void *show*() = 0;
virtual const E *set*(const int &, const E) = 0;
virtual void *sort*(bool (*compare)(const E, const E)) = 0;
	private:
	NodeLinkedListt<E> *data;
	int size;
	
	public:
	LinkedListt();
	~*LinkedListt*();
	virtual int *indexOf*(const E);
	virtual int *lastIndexOf*(const E);
	virtual bool *add*(const E);
	virtual bool *add*(const int &, const E);
	virtual bool *addFirst*(const E);
	virtual bool *removeAt*(const int &);
	virtual bool *remove*(const E);
	virtual int *getSize*();
	virtual bool *isEmpty*();
	virtual void *clear*();
	virtual bool *contains*(const E);
	virtual E *get*(const int &);
	virtual const E *set*(const int &, const E);
	virtual void *show*();
	virtual void *sort*(bool (*compare)(const E, const E));
	static bool compareASC(const E, const E);
	static bool compareDESC(const E, const E);
	NodeLinkedListt<E> *listIterator();

class ArrayListt<E> : public Listt<E>	class NodeLinkedListt<E>
Là lớp kế thừa, implement từ lớp thuần ảo Listt. Lớp này sử dụng cấu trúc dữ liệu danh sách đặc	Là lớp được sử dụng trong lớp LinkedListt với vai trò như là một node của danh sách liên kết (quan hệ composition với lớp LinkedListt)

private:
E *data;
int size;
 	
public:
ArrayListt();
~ArrayListt();
int *indexOf*(const E);
int *lastIndexOf*(const E);
virtual bool *add*(const E);
bool *add*(const int &, const E);
virtual int *getSize*();
virtual E *get*(const int &);
virtual bool *removeAt*(const int &);
virtual bool *remove*(const E);
virtual bool *isEmpty*();
virtual void *clear*();
virtual bool *contains*(const E);
virtual const E *set*(const int &, const E);
virtual void *sort*(bool (*compare)(const E, const E));
static bool compareASC(const E, const E);
static bool compareDESC(const E, const E);
void *show*();
template <class T> friend std::ostream &operator<<(std::ostream &, ArrayListt<T> &);
template <class T> friend std::ostream &operator<<(std::ostream &, ArrayListt<T> *);
	private:
	E data;
	NodeLinkedListt *previous;
	NodeLinkedListt *next;
	
	public:
	NodeLinkedListt(const E);
	~NodeLinkedListt();
	bool hasNext();
	bool hasPrevious();
	NodeLinkedListt *getNext();
	NodeLinkedListt *getPrevious();
	bool setNextNode(NodeLinkedListt *);
	bool setPreviousNode(NodeLinkedListt *);
	const E getData();
	const E setData(const E);
	
	template<typename T>
	struct is_pointer { static const bool value = false; };
	
	template<typename T>
	struct is_pointer<T*> { static const bool value = true; };

Các lớp điều khiển do Qt hỗ trợ:

- MainWindow: Điều khiển giao diện chính của chương trình
- ManageUser: Điều khiển giao diện quản lý tài khoản
- ManageRole: Điều khiển giao diện quản lý quyền
- ManageBook: Điều khiển giao diện quản lý Sách
- ManageCateogry: Điều khiển giao diện quản lý chuyên mục
- ManagePublisher: Điều khiển giao diện quản lý nhà xuất bản
- ManageIssuingCompany: Điều khiển giao diện quản lý công ty phát hành
- Manage_Author: Điều khiển giao diện quản lý tác giả

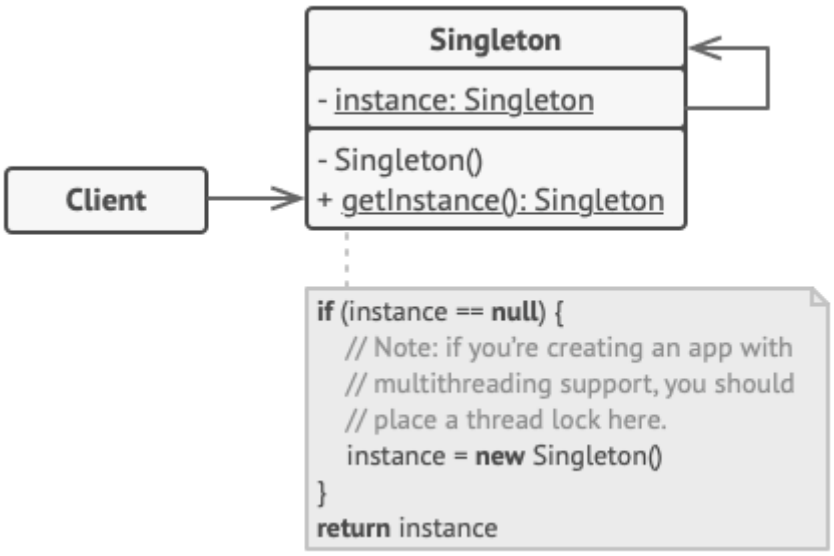
Xây dựng lớp Service và Repository theo cấu trúc sau, từ đó mỗi thực thế sẽ kế thừa và có những hàm riêng để xử lý phần cơ sở dữ liệu:

class Service	class Repository
Thực hiện nhận các thao tác đối với một thực thể và thực hiện (Có mối quan hệ composition với lớp Repository)	Thực hiện giao tiếp với cơ sở dữ liệu (Có mối quan hệ composition với lớp DatabaseConnection)

public:
Service();
virtual ~Service();
virtual Listt<T>* *findAll*() = 0;
	public:
	Repository();
	virtual ~*Repository*();
	virtual T *parse*(QSqlQuery*) = 0;
	virtual Listt<T>* *findAll*() = 0;

5.1.3. Mẫu Singleton

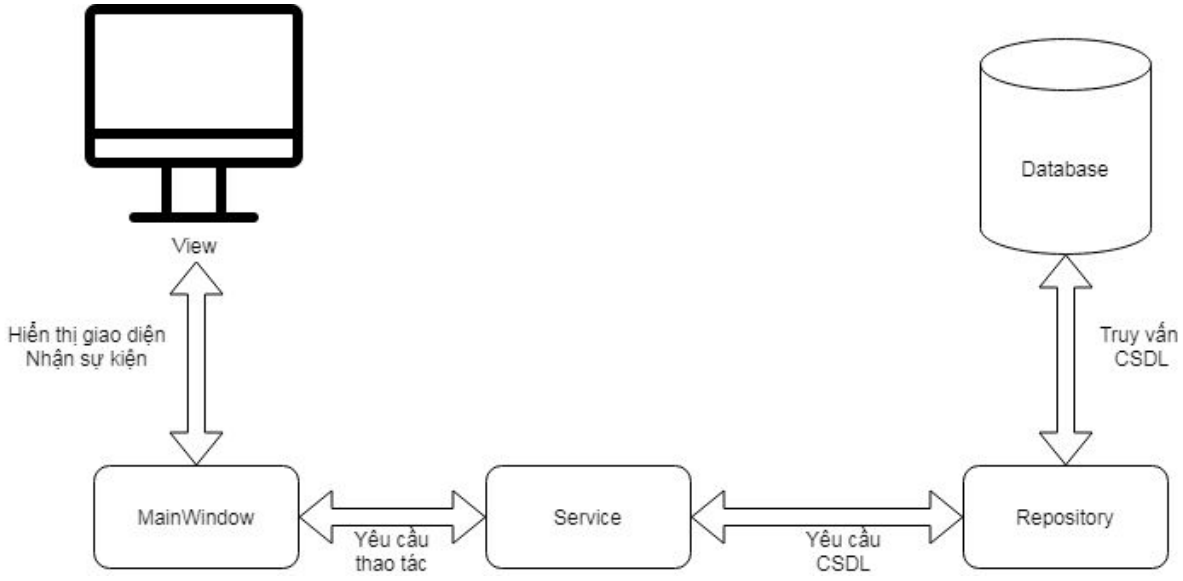
Mẫu Singleton được áp dụng trong nhiều chương trình hướng đối tượng, với mục đích hạn chế việc khởi tạo đối tượng, sử dụng một đối tượng duy nhất để điều phối chương trình, tránh việc xung đột dữ liệu. Lớp áp dụng mẫu Singleton sẽ có hàm dựng thuộc phạm vi Private. Chính vì vậy để yêu cầu một đối tượng, chương trình sẽ yêu cầu thông qua hàm tĩnh của Lớp đó và hàm tĩnh có chức năng duy trì một đối tượng duy nhất trong suốt chương trình.



Hình 3. Mẫu Singleton

Trong dự án này, mẫu Singleton được áp dụng cho hầu hết các lớp có chức năng điều khiển chương trình như các lớp **Service, Repository, DotEnv**

5.1.4. Luồng chạy của chương trình



Hình 4. Luồng chạy của chương trình

5.1.5. Xây dựng chức năng chính

5.1.5.1 Chức năng đọc biến môi trường

Khi vận hành thực tế, phần mềm sẽ cần một số thông số mà đòi hỏi cần bảo mật như HOST ,USERNAME và PASSWORD của cơ sở dữ liệu. Chính vì vậy lớp DotEnv được xây dựng với mục đích đọc các biến môi trường ấy từ file nếu như ứng dụng đang trong quá trình phát triển hoặc đọc các biến môi trường ấy từ hệ điều hành nếu như trong môi trường sản phẩm. Cách sử dụng lớp DotEnv:

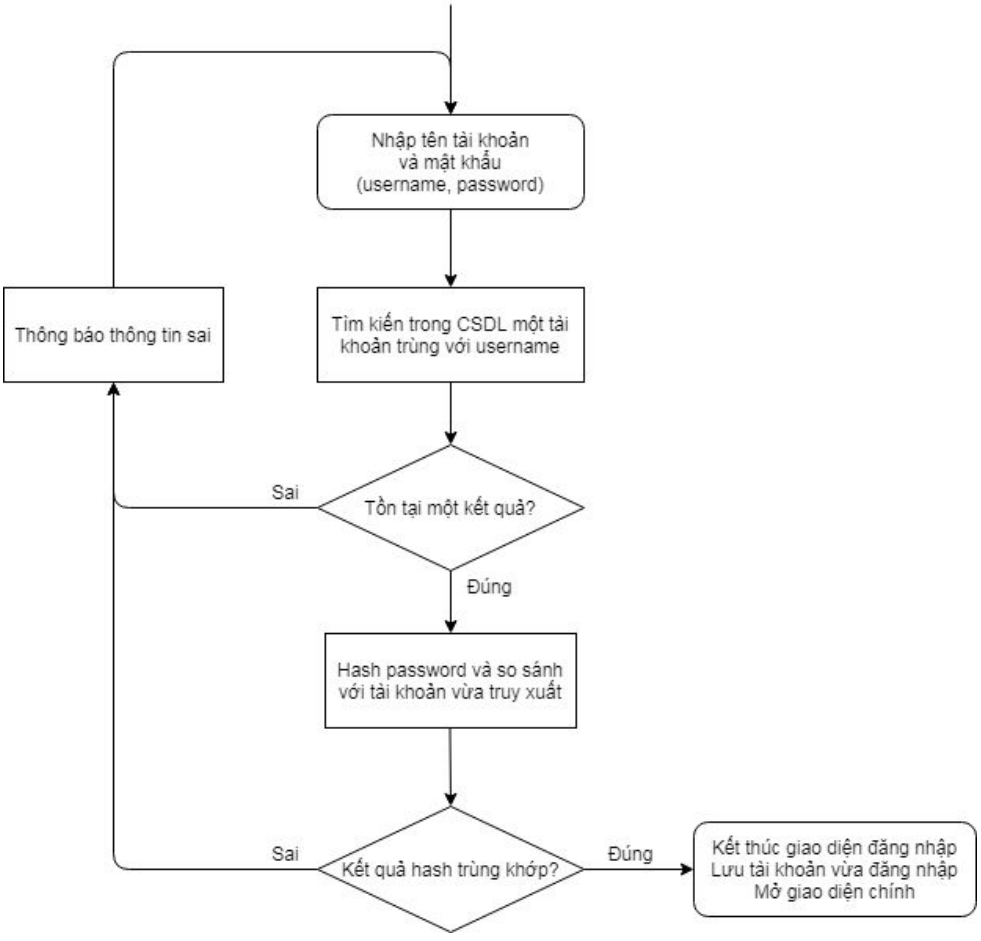
```
DotEnv* dotEnv = DotEnv::initDotEnv();
String DB_NAME = (*dotEnv)["DB_NAME"];
String DB_SERVER = (*dotEnv)["DB_SERVER"];
```

Cấu trúc file giả lập biến môi trường trong môi trường phát triển ứng dụng:

```
DB_NAME=TenCSDL
DB_SERVER=ServerSqlServer
```

5.1.5.2 Chức năng đăng nhập - đăng xuất

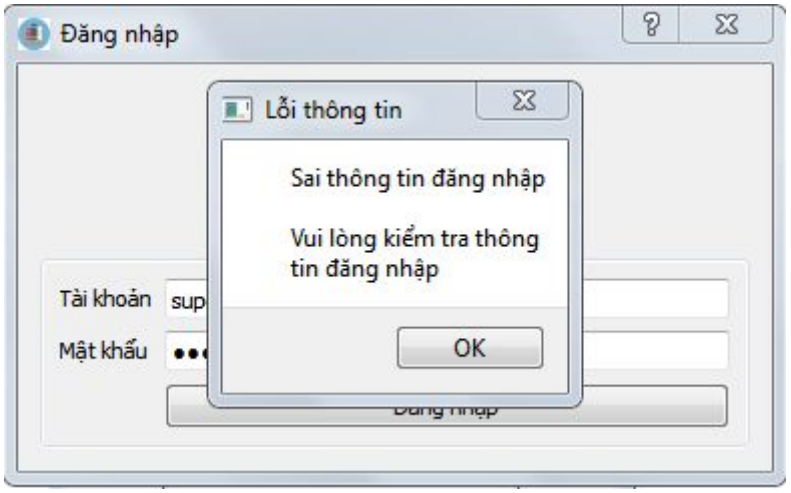
Để đảm bảo bảo mật, mật khẩu do người dùng tạo sẽ được Hash bằng thuật toán MD5 để lưu vào cơ sở dữ liệu. Việc lưu mật khẩu gốc vào cơ sở dữ liệu trong thực tế sẽ không hợp lý, khi hacker tấn công vào cơ sở dữ liệu, lấy được mật khẩu gốc thì hoàn toàn có thể đăng nhập được hệ thống. Thay vào đó MD5 là thuật toán mã hóa một chiều, tức là chuỗi ký tự mật khẩu sẽ được mã hóa thành chuỗi hex 32 ký tự, cho dù có được chuỗi mã hóa cũng không thể suy ngược thành mật khẩu. Chính vì vậy sau khi nhập tên đăng nhập và mật khẩu, thứ tự thao tác sẽ như sau:



Hình 5. Sơ đồ thực hiện đăng nhập

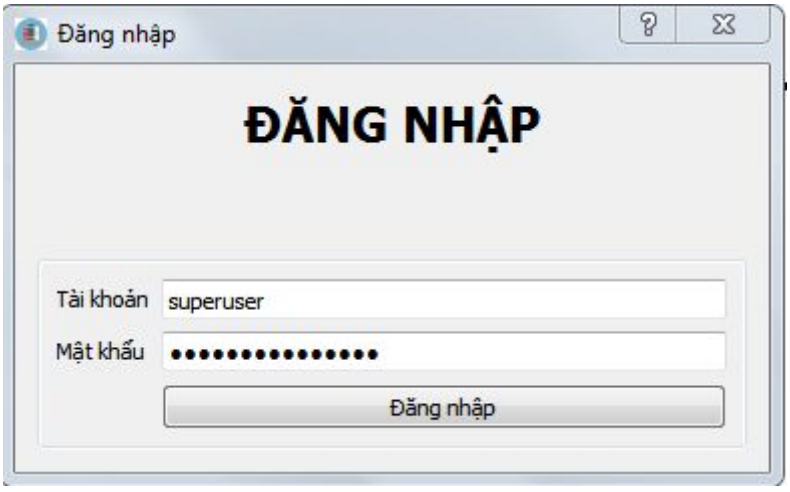
Sau khi đăng nhập, trường User *sessionUser thuộc lớp MainWindow được gán User vừa đăng nhập, cũng chính là trường lưu tài khoản đang làm

việc. Mọi việc truy xuất thông tin tài khoản đang làm việc đều thông qua trường này.



Hình 6. Thông báo lỗi đăng nhập

Khi đăng xuất, trường sessionUser được gán giá trị NULL, đồng thời đóng giao diện chính hiện tại, hiển thị giao diện đăng nhập.



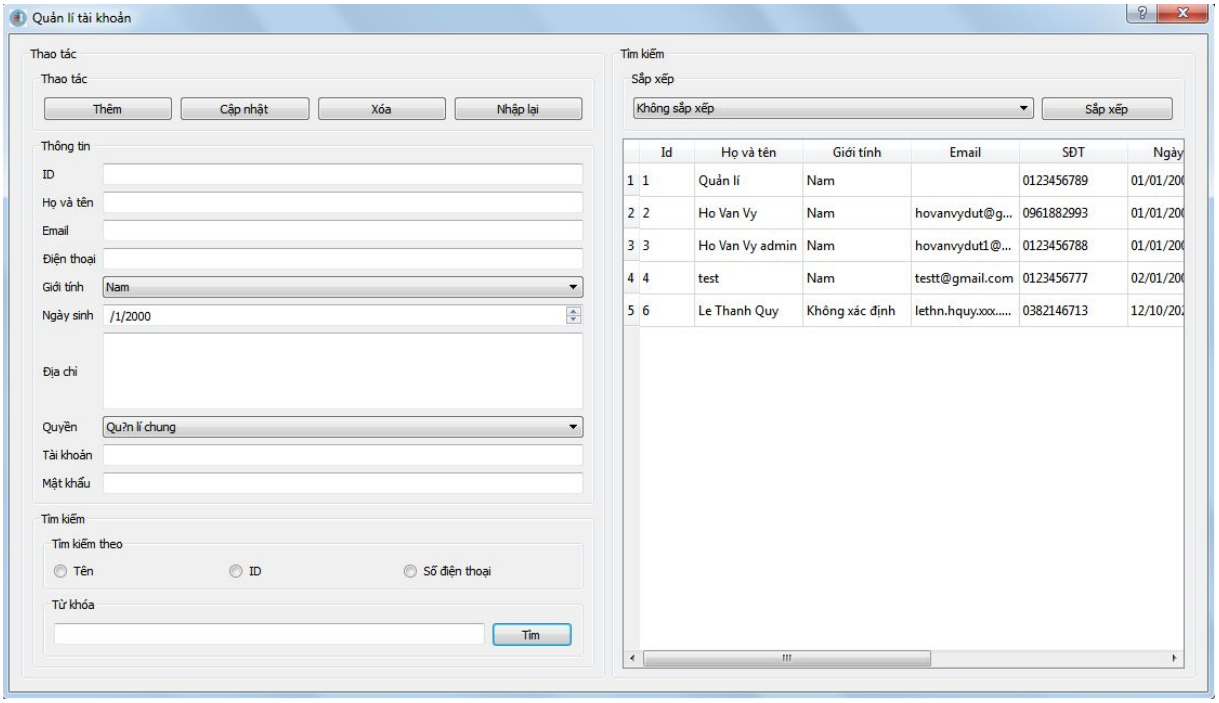
Hình 7. Giao diện đăng nhập

5.1.5.3 Chức năng Đọc - Xem - Sửa - Xóa tài khoản:

Khi vào menu -> Quản lý -> Quản lý tài khoản. Giao diện quản lý tài khoản được mở ra

Xem thông tin tài khoản: Chức năng tìm kiếm giúp người dùng có thể tìm ngay tài khoản cần truy xuất. Có ba lựa chọn bao gồm tìm kiếm theo tên, theo ID và theo số điện thoại. Sau khi nhấn nút tìm kiếm, dữ liệu sẽ hiển thị tại bảng bên trái và người dùng có thể nháy đúp lên tài khoản, thông tin chi tiết sẽ được hiển thị lên những ô bên trái. Người dùng củng có các tùy chọn sắp xếp để dễ dàng quản lý nhiều tài khoản. Tất nhiên trường mật khẩu sẽ không thể

hiển thị vì mật khẩu đã được mã hóa một chiều và không thể khôi phục mật khẩu gốc.



Hình 8. Thông tin hiển thị chi tiết sau khi được lựa chọn

Đồng thời nếu tài khoản được chọn có quyền ưu tiên cao hơn tài khoản hiện thời, các nút Cập nhật và Xóa sẽ bị ẩn đi. Đây chính là tính năng phân quyền trong hệ thống.

Đối với việc tìm kiếm theo tên và tên số điện thoại, ta dựng toán tử LIKE trong SQL để việc tìm kiếm thuận tiện hơn, chỉ cần nhập một phần của từ khóa để tìm kiếm. Ví dụ:

```
SELECT * FROM users WHERE deleted_at IS NULL AND fullname LIKE '%TuKhoa%'
```

Thêm tài khoản: Sau khi người dùng nhập đầy đủ thông tin, nhấn nút Thêm để thêm mới một tài khoản. Sau khi nhận yêu cầu thêm tài khoản, chương trình bắt đầu lấy dữ liệu từ các ô nhập trên giao diện, ánh xạ vào một đối tượng User và thực hiện xác thực dữ liệu:

- Kiểm tra các trường bắt buộc gồm họ tên, số điện thoại đã điền chưa.
- Kiểm tra số điện thoại và email có hợp lệ không (sử dụng biểu thức chính quy), kiểm tra tên có chưa ký tự đặc biệt hoặc chữ số không.
    - Ví dụ biểu thức chính quy (regular expression):
    - phone = /[0-9]{5, 11}/ // Gồm 5 đến 10 chữ số
    - email = /[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,4}/

- Kiểm tra xem tên đăng nhập có dài hơn 5 ký tự và mật khẩu dài hơn 10 ký tự không.
- Kiểm tra quyền: Chỉ cho phép thêm mới một người có độ ưu tiên của quyền thấp hơn mình.
- Nếu chọn quyền không phải Khách hàng thì kiểm tra đã nhập tên đăng nhập và mật khẩu chưa (Nếu tại mục Quyền chọn Khách hàng thì tại ô nhập tên tài khoản và mật khẩu sẽ ẩn đi). Nếu chọn quyền là Khách hàng thì tài khoản này không dùng để đăng nhập, do đó trường username sẽ được đặt là số điện thoại và mật khẩu được đặt mặc định.
- Kiểm tra xem số điện thoại và tên tài khoản đã được sử dụng chưa.

Nếu thông tin không đúng thì thông báo cho người dùng dùng sữa bằng cách sử dụng ngoại lệ, ném ngoại lệ ra để ở giao diện chính nắm bắt và xử lý. Ngược lại thì gọi phương thức Insert để thêm vào cơ sở dữ liệu.

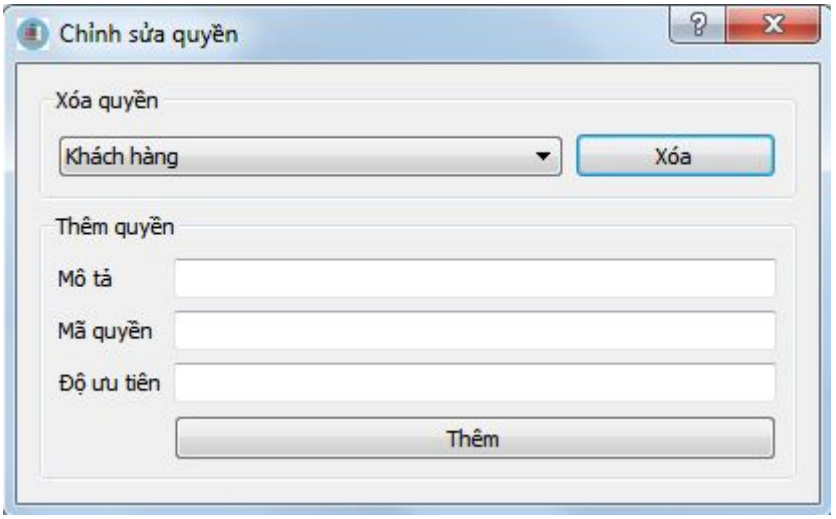
Xóa tài khoản: Như đã đề cập đến ở trên, đối với những dữ liệu quan trọng thì việc xóa dữ liệu thường là xóa mềm (Không xóa mất khỏi cơ sở dữ liệu). Chính vì vậy trong bảng users có trường deleted_at, lúc khởi tạo nhật giá trị NULL. Khi nhận được yêu cầu xóa thì chương trình sẽ kiểm tra người dùng đã chọn tài khoản cần xóa. Kiểm tra mức độ ưu tiên của quyền người bị xóa và cuối cùng kiểm tra tài khoản đó đã trả đủ sách chưa. Nếu không hợp lệ sẽ thông báo, ngược lại thì gọi phương thức delete, phương thức này sẽ cập nhật người được xóa đó với trường deleted_at thành thời gian hiện tại, với ý nghĩa người này đã bị xóa đi. Ví dụ truy vấn:

```
UPDATE users SET deleted_at = GETDATE() WHERE user_id = 1
```

Cập nhật tài khoản: Sau khi chọn tài khoản, thông tin được hiển thị ở bên trái. Người dùng có thể chỉnh sửa thông tin đó sau đó nhấn nút Cập nhật để cập nhật thông tin. Chương trình củng sẽ đi kiểm tra thông tin tương tự như lúc Insert. Lưu ý rằng như đã đề cập ở trên, nếu chọn tài khoản có độ ưu tiên cao hơn thì sẽ không có quyền cập nhật.

5.1.5.4 Chức năng Thêm - Xóa quyền

Chỉ tài khoản có độ ưu tiên cao nhất có quyền thêm hoặc xóa quyền.



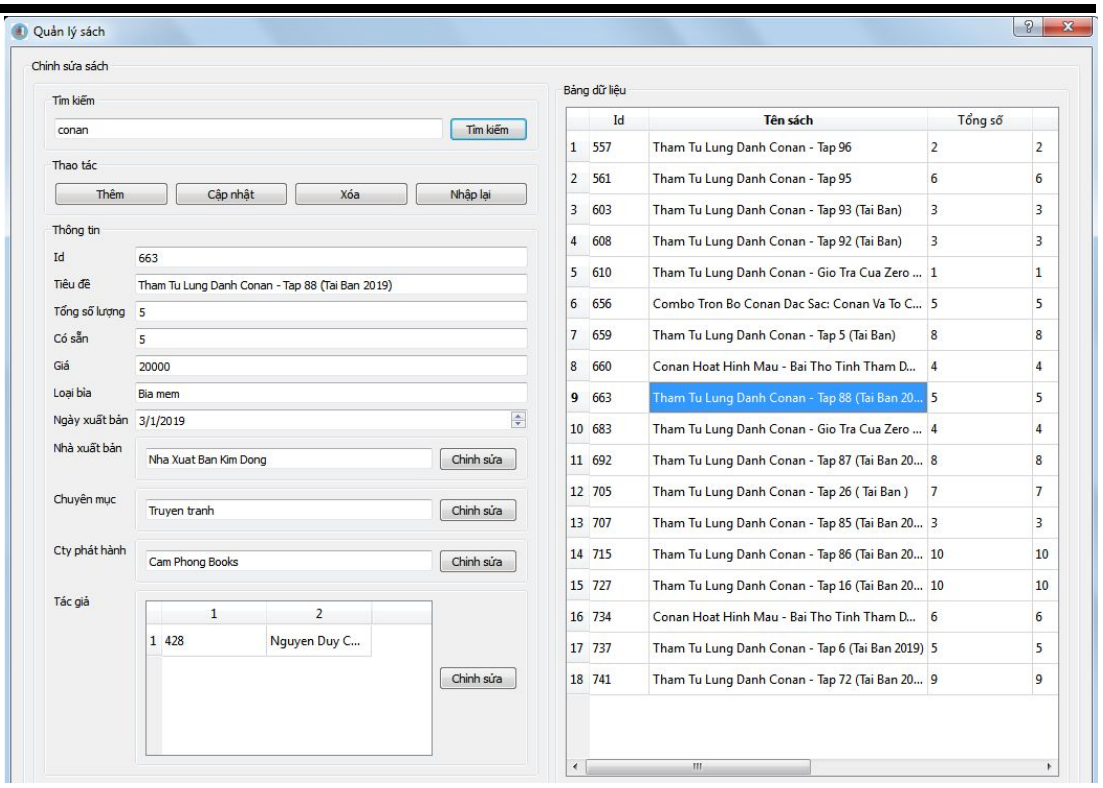
Hình 9. Giao diện thêm - xóa quyền

Người dùng có thể chọn quyền và nhấn nút xóa, quyền superuser và quyền guest là hai quyền không được phép xóa (Quyền cao nhất và thấp nhất). Ngoài ra khi xóa một quyền bất kỳ, để tránh mất mát dữ liệu, những tài khoản thuộc quyền vừa xóa sẽ không bị xóa theo mà sẽ được gán quyền thấp nhất. Việc này được thực hiện hoàn toàn tự động nhờ trigger.

Người dùng cũng có thể nhập đầy đủ thông tin, sau đó nhấn Thêm để thêm quyền mới, để dễ dàng kiểm soát thì dữ liệu được đảm bảo là độ ưu tiên sẽ từ 1 đến 99, độ ưu tiên 0 sẽ là quyền superuser - quyền cao nhất và độ ưu tiên 100 là khách hàng - độ ưu tiên thấp nhất.

5.1.5.5 Chức năng quản lý sách

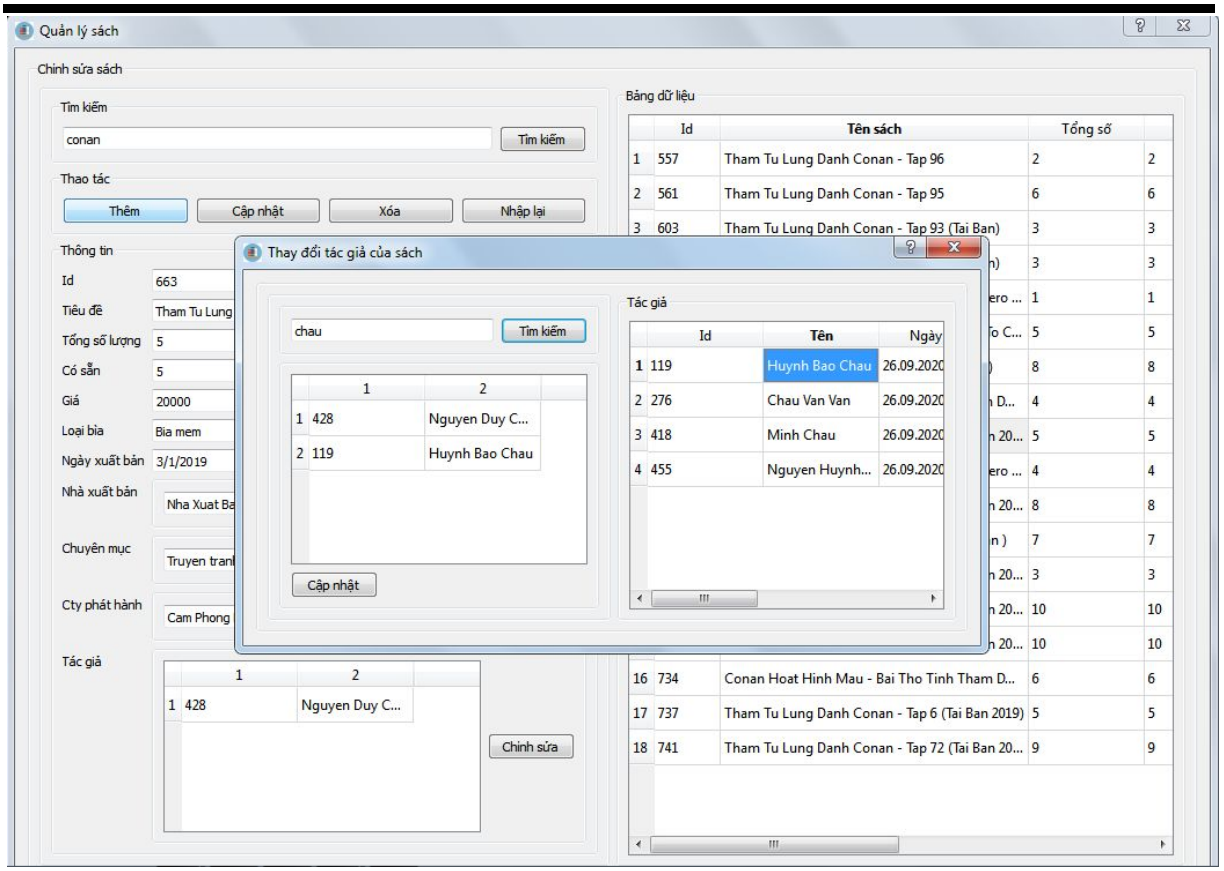
Chức năng quản lý sách: Tại giao diện này, người dùng có thể tự do tìm kiếm sách theo tên, nháy đúp vào sách cần chỉnh sửa ở bảng kết quả tìm kiếm (bảng nằm phía bên phải giao diện) thì dữ liệu sẽ đổ sang form để chỉnh sửa ở bên trái.



Hình 10. Giao diện quản lý sách

Tại form này, người dùng có thể:

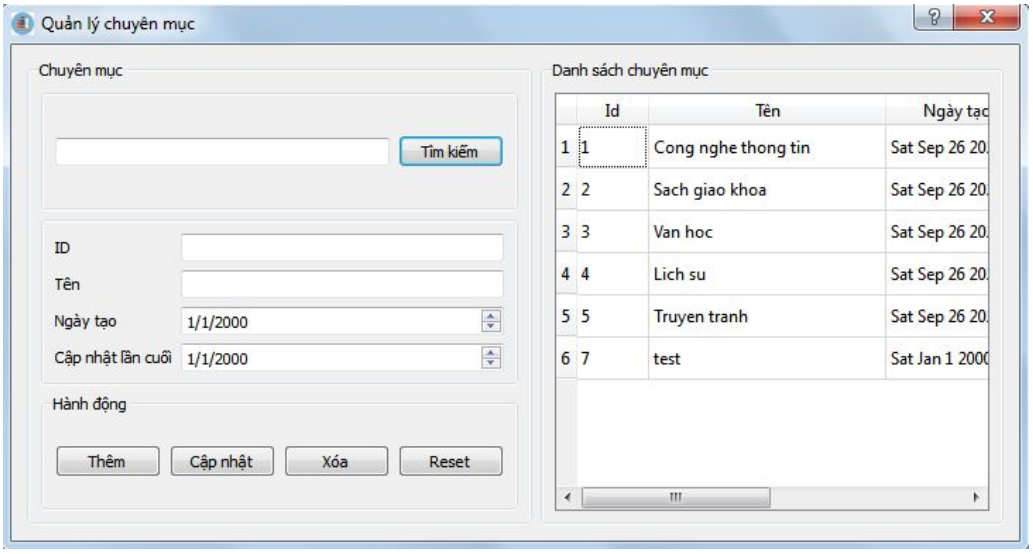
- Điều chỉnh tên sách
- Điều chỉnh số lượng có sẵn, tổng số lượng sách
- Điều chỉnh ngày xuất bản
- Thay đổi nhà xuất bản, chuyên mục sách, công ty phát hành đã có trong cơ sở dữ liệu (có giao diện hỗ trợ tìm kiếm, thay đổi các nhà xuất bản, chuyên mục, công ty phát hành trong cơ sở dữ liệu)
- Thêm hoặc xóa các tác giả của sách hiện tại (có giao diện hỗ trợ tìm kiếm, thêm, xóa các tác giả của sách hiện tại trong cơ sở dữ liệu)



Hình 11. Giao diện chỉnh sửa tác giả của sách được chọn

5.1.5.6 Chức năng quản lý chuyên mục sách

Tại giao diện này, chương trình hỗ trợ người dùng các tính năng tìm kiếm theo tên, tạo mới chuyên mục sách, cập nhật chuyên mục đã tồn tại cũng như xóa chuyên mục



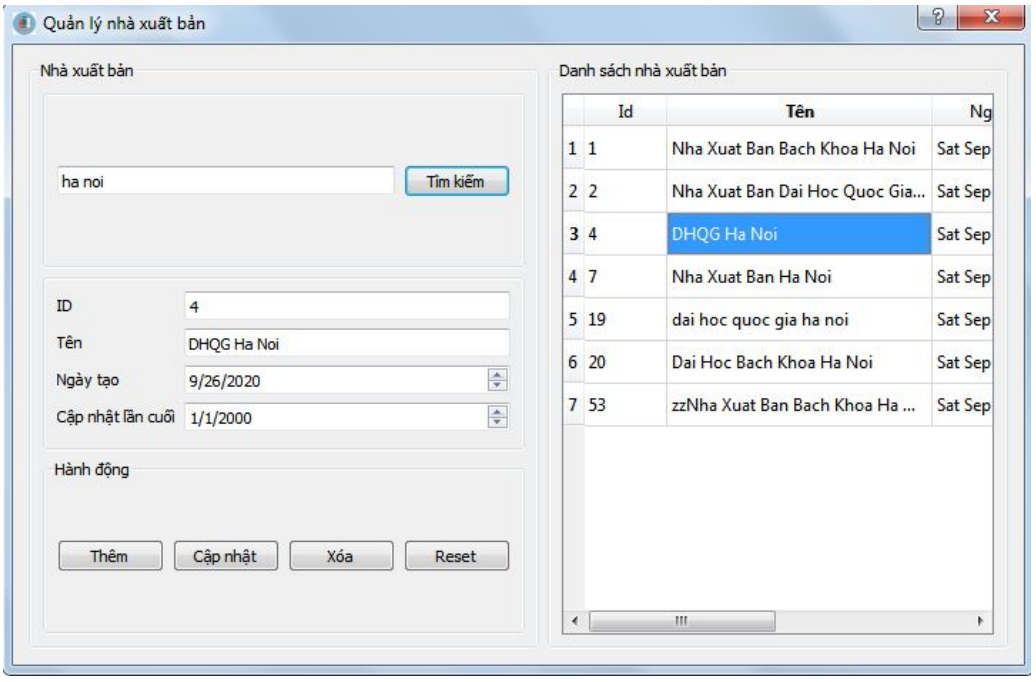
Hình 12. Giao diện quản lý chuyên mục sách

5.1.5.7 *Chức năng quản lý tác giả*

Tương tự như chức năng quản lý chuyên mục sách, cũng hỗ trợ các chức năng tìm kiếm, tạo mới, cập nhật, xóa tác giả

5.1.5.8 *Chức năng quản lý nhà xuất bản*

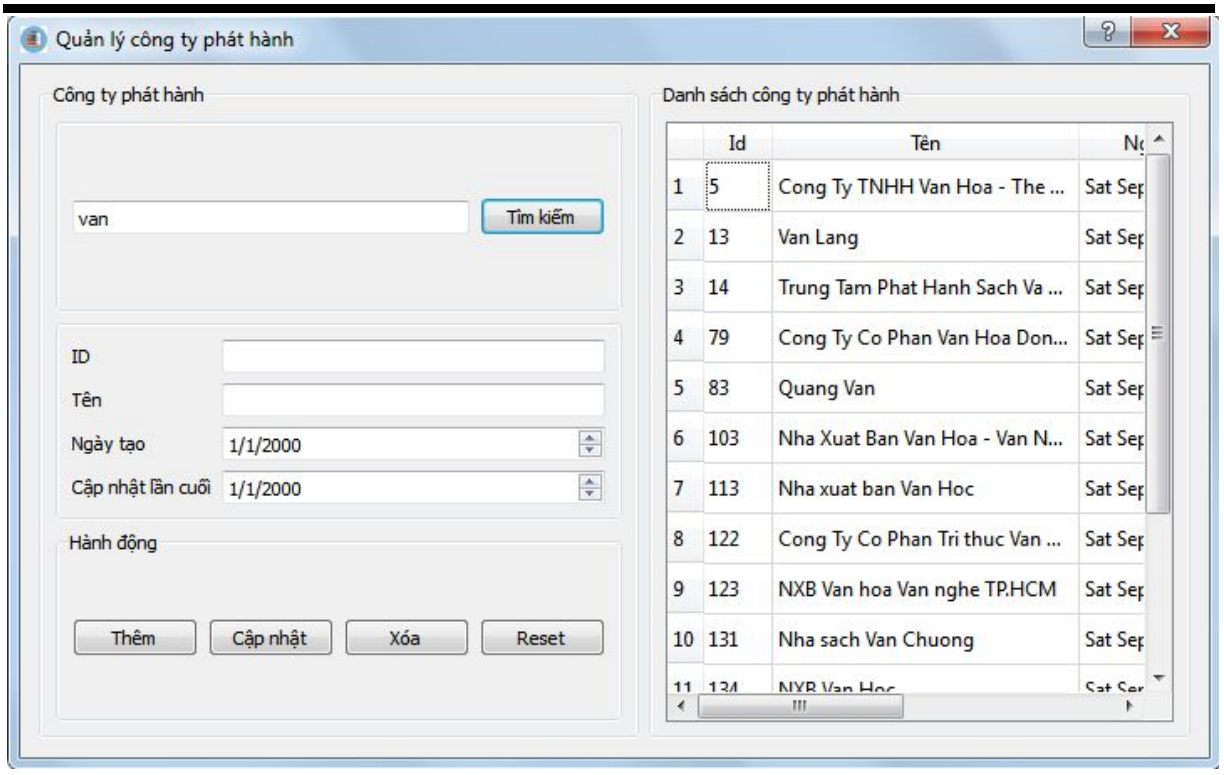
Tương tự như chức năng quản lý chuyên mục sách, cũng cung cấp, hỗ trợ các chức năng tìm kiếm, tạo mới, cập nhật, xóa nhà xuất bản



Hình 13. Giao diện quản lý nhà xuất bản

5.1.5.9 *Chức năng quản lý công ty phát hành*

Tương tự như chức năng quản lý chuyên mục sách, giao diện này cũng cung cấp, hỗ trợ các chức năng tìm kiếm, tạo mới, cập nhật, xóa công ty phát hành



Hình 14. Giao diện quản lý công ty phát hành

5.1.5.10 Chức năng mượn - trả sách

Chức năng mượn sách: Tại góc bên trái dưới cùng của giao diện chính,

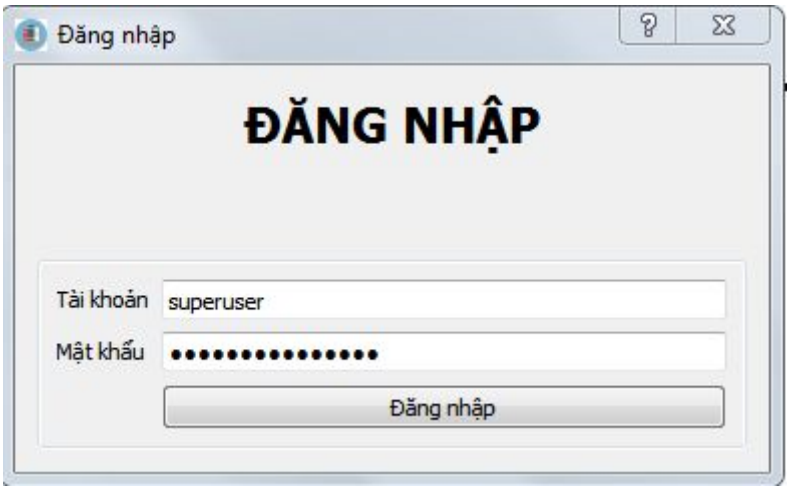
người dùng tiến hành tìm kiếm người cần mượn sách theo tên, số điện thoại hoặc id. Kết quả tìm kiếm sẽ được hiển thị thành một danh sách ngay bên dưới ô tìm kiếm. Người dùng bắt buộc phải nháy đúp chuột để chọn người dùng mượn sách, khi chọn thành công, dữ liệu người dùng được chọn sẽ được đổ dữ liệu sang ô bên phải ngay cạnh danh sách. Tiếp theo đó, người dùng phải cần chọn những sách cần mượn bằng cách sử dụng chức năng tìm kiếm sách theo tên sách, tên tác giả, mã id của sách tại ô tìm kiếm sách, kết quả tìm kiếm sách cũng được hiển thị thành một danh sách ngay bên dưới ô tìm kiếm. Người dùng chỉ việc nháy đúp vào các sách muốn mượn, những sách này sẽ được đổ vào danh sách phía bên trái trên cùng. Cuối cùng người dùng chọn nút mượn sách là hoàn tất công việc.

Chức năng trả sách: Tại giao diện chính cũng có chức năng tìm kiếm tài khoản tương tự như giao diện quản lý tài khoản. Người dùng có thể tìm kiếm và chọn tài khoản bằng cách nháy đúp vào tài khoản cần trả sách, thông tin cá nhân sẽ hiện sang một bên. Tiếp theo người dùng nhấn Nút sách đã mượn, chương trình sẽ truy vấn và hiển thị ra những sách và số lượng sách người đó đã mượn. Người dùng có thể nháy đúp vào những cuốn sách muốn trả, những cuốn sách đó sẽ được chuyển sang bảng bên trái. Sau đó nhấn nút trả sách để trả những cuốn sách đã chọn. Tương tự, bảng mượn sách cũng là

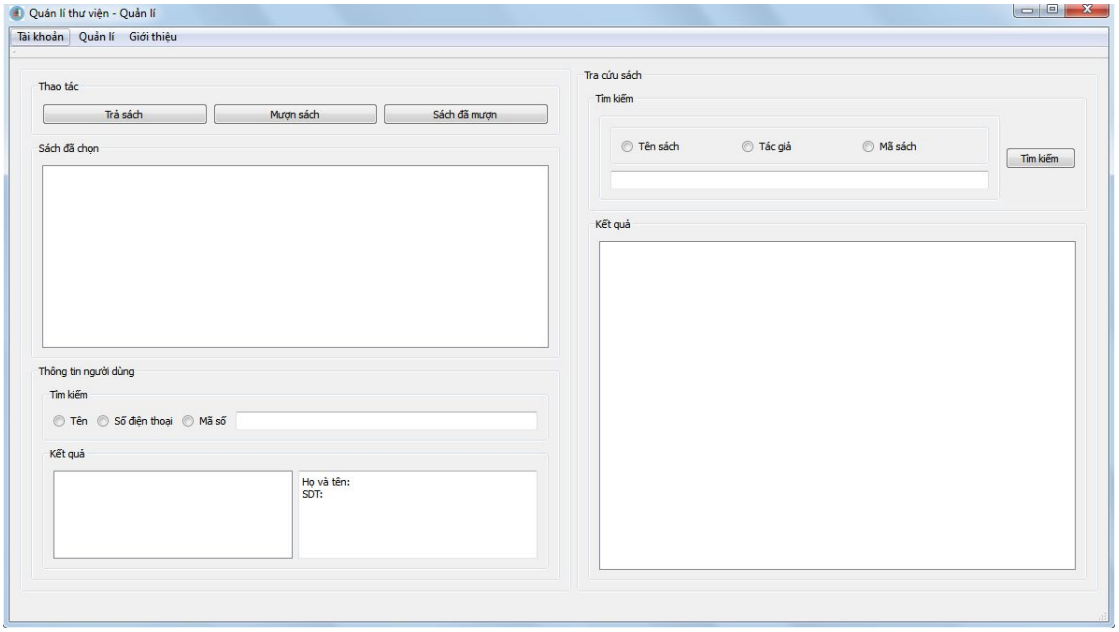
một dữ liệu quan trọng, do đó khi người dùng trả sách, chương trình sẽ không xóa hẳn mà sẽ cập nhật trường deleted_at với ý nghĩa trường này đã được xóa.

5.2. Kết quả

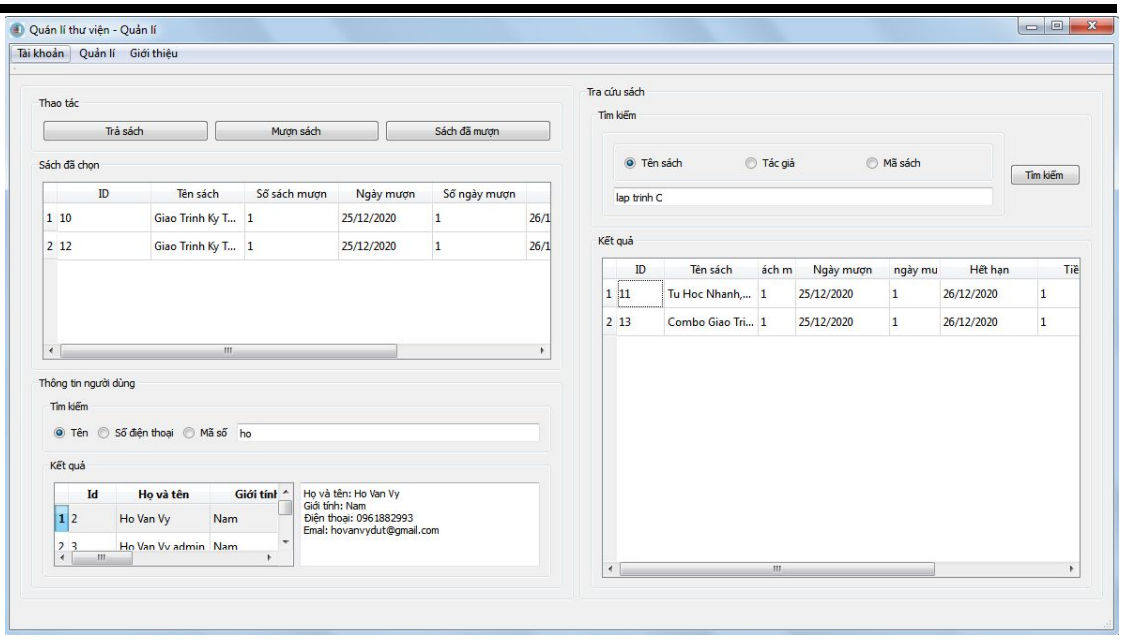
5.2.1. Giao diện chính của chương trình



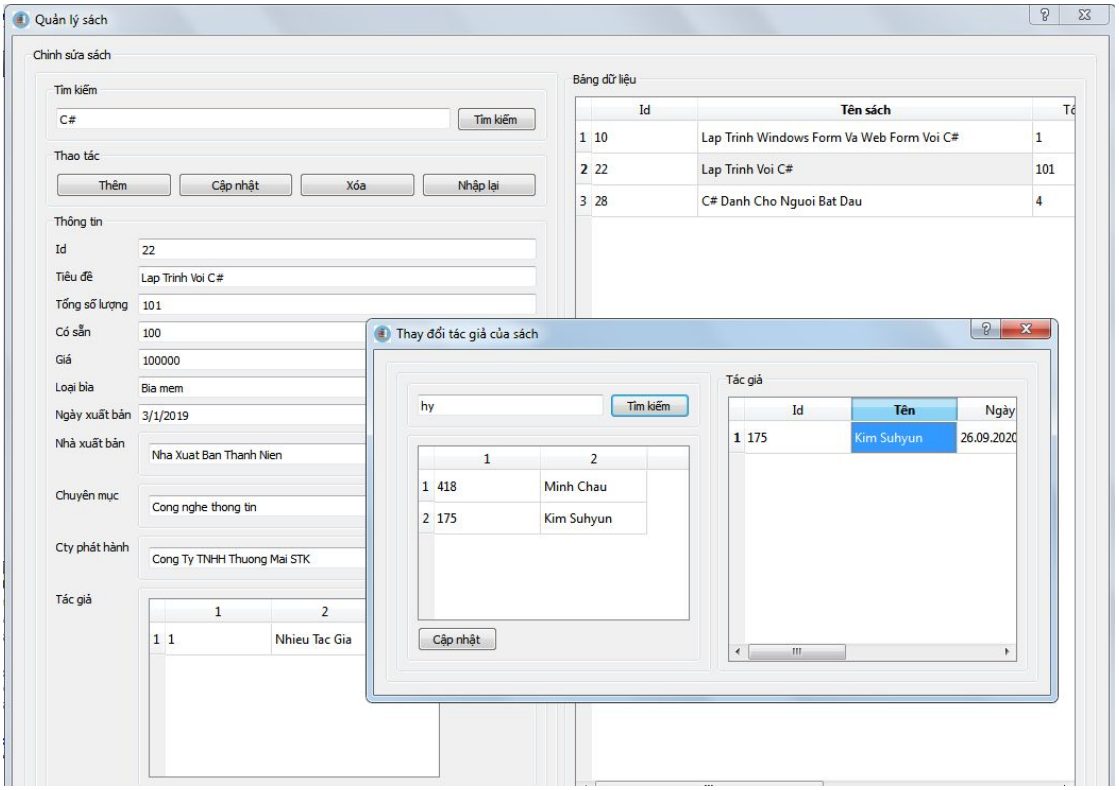
Hình 15. Giao diện đăng nhập



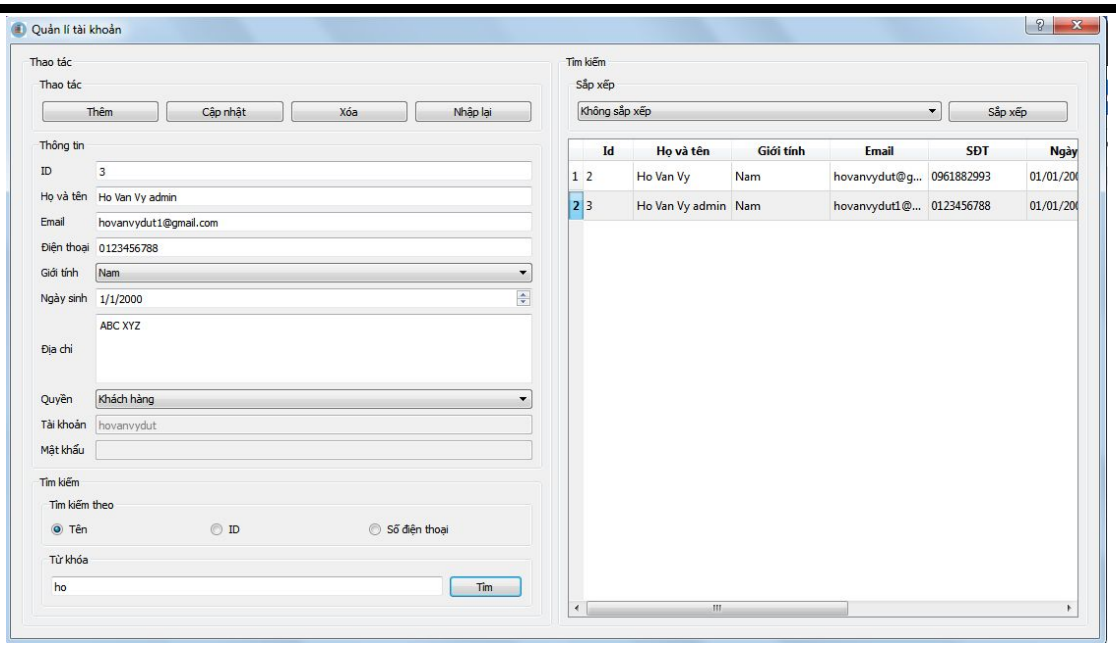
Hình 16. Giao diện ban đầu quản lý mượn trả sách



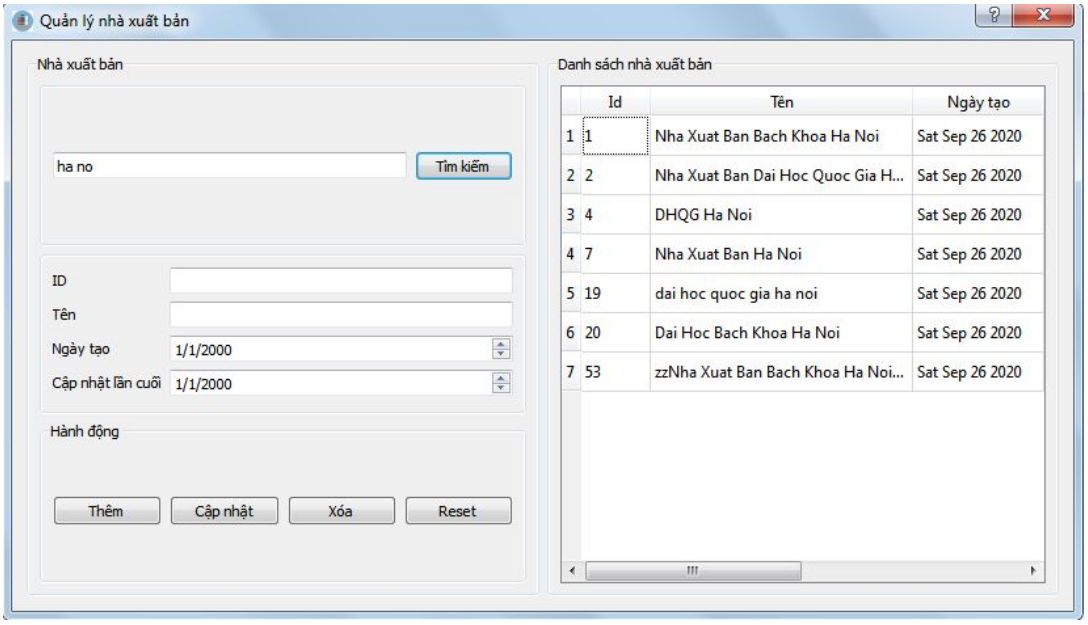
Hình 17. Giao diện thực hiện mượn - trả sách



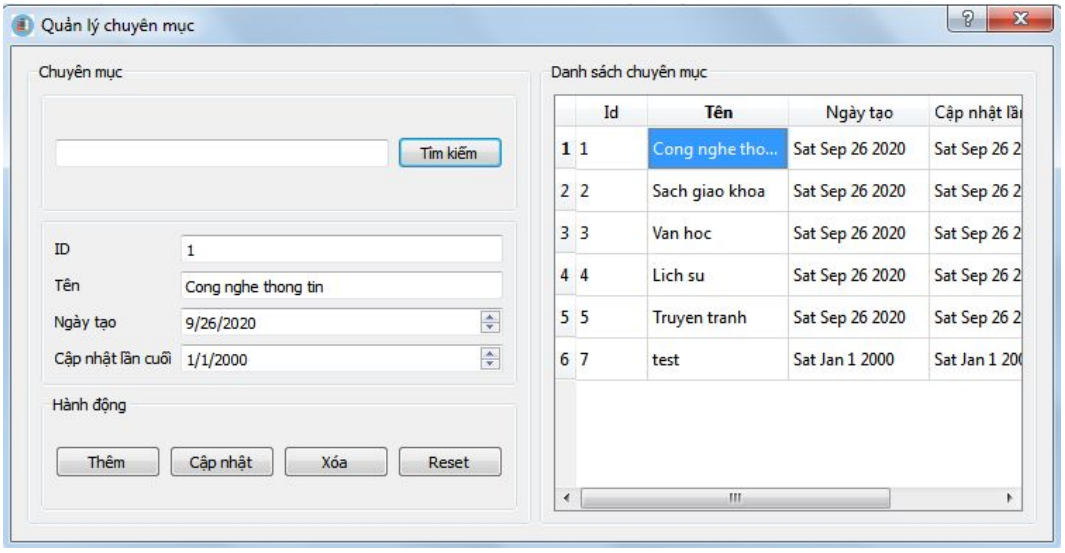
Hình 18. Giao diện quản lý sách



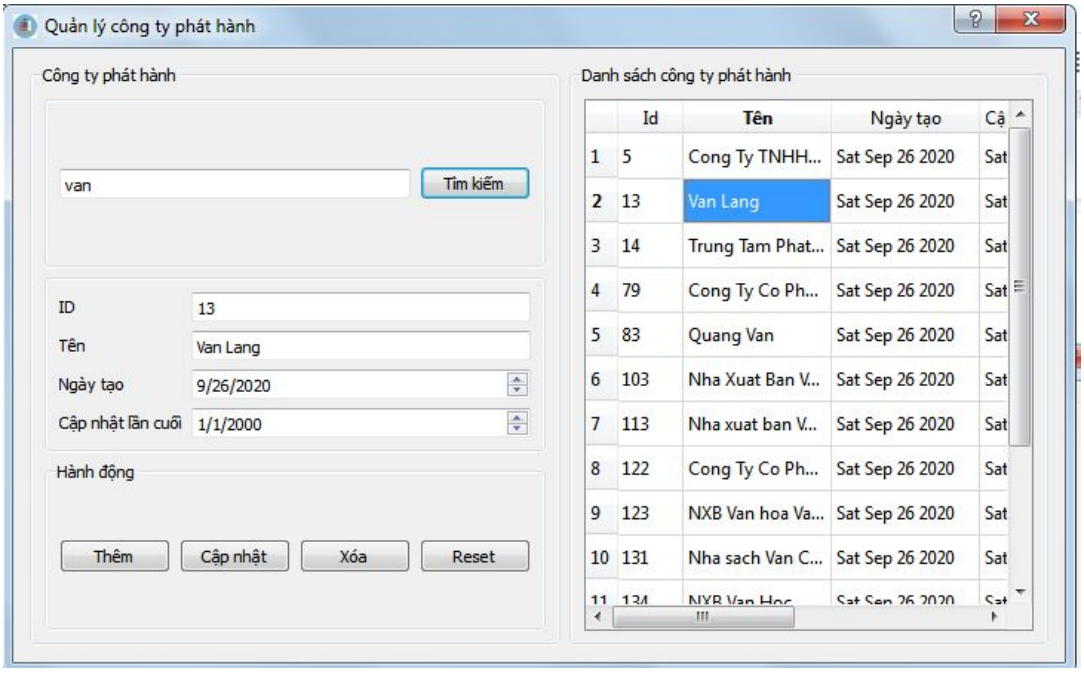
Hình 19. Giao diện quản lý tài khoản



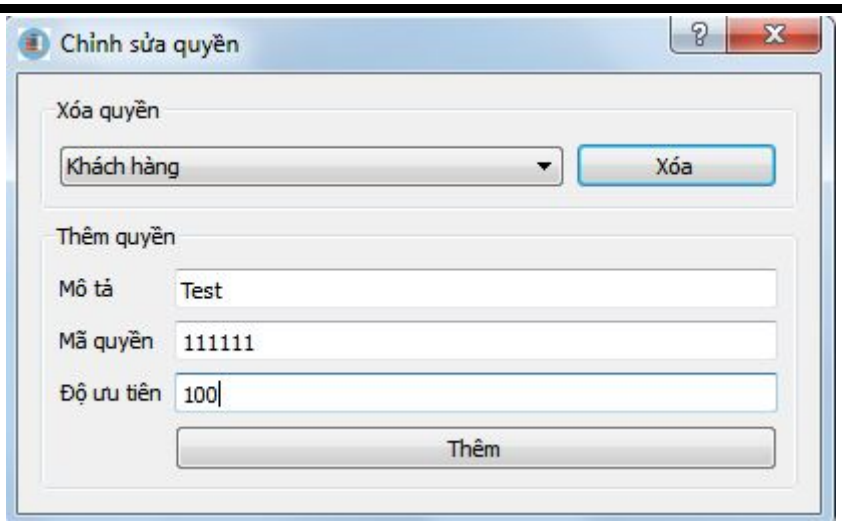
Hình 20. Giao diện quản lý nhà xuất bản



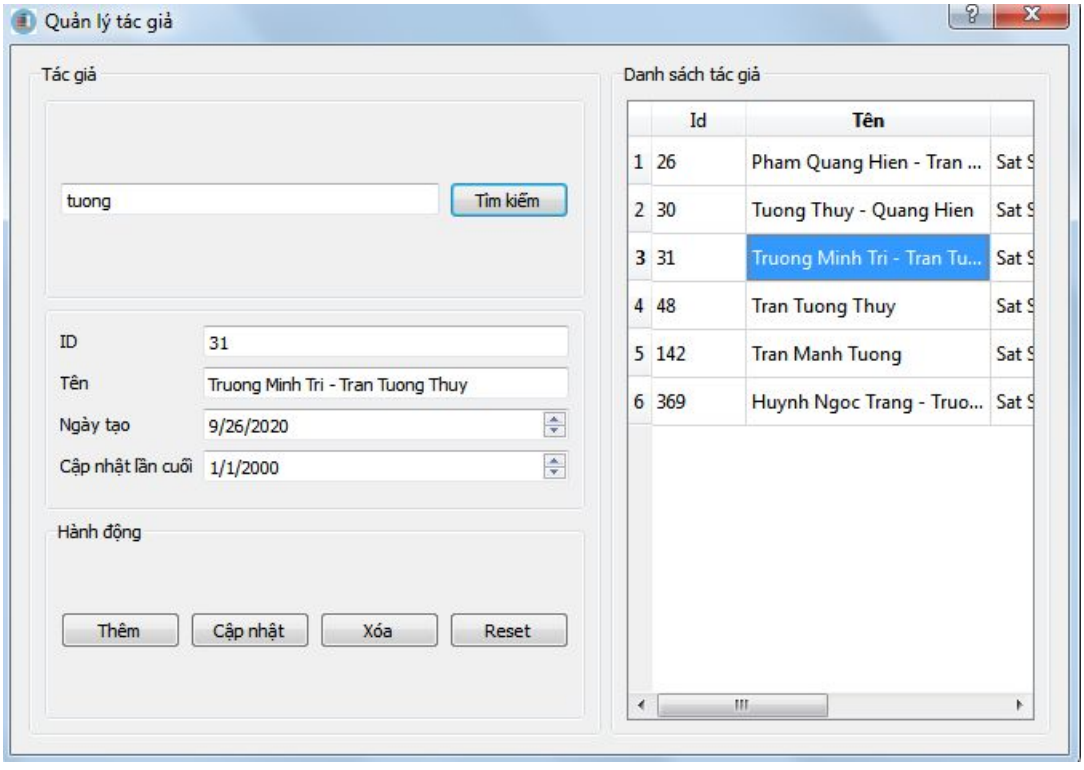
Hình 21. Giao diện quản lý chuyên mục



Hình 22. Giao diện quản lý công ty phát hành



Hình 23. Giao diện quản lý quyền



Hình 24. Giao diện quản lý tác giả

5.2.2. Kết quả thực thi của chương trình

Sau khi hoàn thiện, chương trình chạy ổn định, hiệu quả. Trong quá trình thử nghiệm không gây mất mát dữ liệu. Không bị lỗi crash chương trình khi chạy. Tốc độ phản hồi tương đối nhanh, tạo cảm giác thuận tiện cho người sử dụng.

5.2.3. Nhận xét

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

a. Kết luận

Sau khi hoàn thành, ứng dụng hoạt động ổn định, đảm bảo tính thuận tiện và bảo mật cho người sử dụng.

Tuy nhiên để hoàn thiện một sản phẩm đưa vào sử dụng được thì cần phải cải thiện và phát triển thêm nhiều tính năng hơn nữa.

b. Hướng phát triển

- Phát triển hệ thống quét mã vạch cho sách thay vì phải tìm kiếm thử công gây mất thời gian.
- Phát triển hệ thống thẻ từ để dễ dàng kiểm soát người mượn sách.
- Phát triển thêm giao diện để lọc ra những cuốn sách chưa được trả hoặc những người dùng đã trễ hẹn giao sách.
- Nâng cấp thuật toán mã hóa mật khẩu.
- Phát triển hệ thống nhắc nhở trả sách thông qua tin nhắn hoặc email.

TÀI LIỆU THAM KHẢO

[1] Refactoring, **Singleton**, *https://refactoring.guru/design-patterns/singleton, 5/12/2020*

[2] Tran Viet Ha, **Mô hình quan hệ - thực thể (Entity – Relationship Model)**, *https://viblo.asia/p/mo-hinh-quan-he-thuc-the-entity-relationship-model-oOVlYEenl8W, 15/12/2020*

[3] Programiz, **Quicksort Algorithm**, *https://www.programiz.com/dsa/quick-sort, 20/12/2020*

[4] Nguyen Tien Manh, **Lập trình phần mềm với Qt**, *https://www.youtube.com/playlist?list=PLVLXkgfvvxZWMFAuB8RjZ6Ex6Hc4hWtGq, 25/11/2020*