BEGINNING

Distributed object programming is a hot topic of software distributed technology these days. Java is the pioneer language with RMI (Remote Method Invocation), an extremely efficient and flexible implementation of distributed objects. Therefore, we have chosen the topic "Building an application to simulate movie ticket booking for cinemas" to help manage this cinema effectively.

Theoretically, the sections that will be covered include:

I.Overview of the topic

II.Theoretical Basis

III.Building an application to simulate movie ticket booking for cinemas

IV.Conclusion, development direction.

In practical terms, when implementing, the topic helps us gain more knowledge about Java programming, understand how a basic mobile application works. This enhanced our knowledge when writing our next RMI application.

Although, we have tried our best in the implementation process, but there will definitely be some shortcomings and limitations. We look forward to your valuable comments and suggestions.

# **TABLE OF CONTENTS**

[ACKNOWLEDGMENTS 1](#_Toc58100707)

[PREFACE 2](#_Toc58100708)

[TABLE OF CONTENTS 3](#_Toc58100709)

[LIST OF DRAWINGS 5](#_Toc58100710)

[Chapter I: OVERVIEW OF THE TOPIC 7](#_Toc58100711)

[1.1. Research Purpose 7](#_Toc58100712)

[1.2.Research Mission 8](#_Toc58100713)

[1.3.Research Methods 8](#_Toc58100714)

[Chapter II: THEORETICAL BASIS 9](#_Toc58100715)

[2.1. JAVA 9](#_Toc58100716)

[2.1.1. Overview of Java 9](#_Toc58100717)

[2.1.2. Why use Java? 10](#_Toc58100718)

[2.2. Overview of RMI 11](#_Toc58100719)

[2.2.1. Overview of RMI 11](#_Toc58100720)

[2.2.2. What is RMI ? 11](#_Toc58100721)

[2.3. How does RMI work? 11](#_Toc58100722)

[2.4. Architecture of the RMI Program 13](#_Toc58100723)

[2.5. Mechanisms involved in a RMI Application 14](#_Toc58100724)

[2.6. The execution mechanism of an RMI Application 15](#_Toc58100725)

[Chapter III: BUILDING THE MOVIE TICKET SIMULATION APP FOR CINEMAS 16](#_Toc58100726)

[3.1. System analysis and design 16](#_Toc58100727)

[3.1.1. App Description 16](#_Toc58100728)

[3.1.2. ERD Diagram 17](#_Toc58100729)

[3.1.3. Database Diagram 17](#_Toc58100730)

[3.4. Application interface 18](#_Toc58100731)

[3.4.1. Interface“Server” 18](#_Toc58100732)

[3.3.2. Interface “Client” 18](#_Toc58100733)

[Chapter IV:CONCLUSIONS AND DEVELOPMENT DIRECTIONS 25](#_Toc58100734)

[4.1.Conclude 25](#_Toc58100735)

[4.1.1.Reach 25](#_Toc58100736)

[4.1.2.Restriction 25](#_Toc58100737)

[4.2.Recommend 25](#_Toc58100738)

[4.2.1.Development orientation 25](#_Toc58100739)

[4.2.2.Orientation for expansion and improvement 26](#_Toc58100740)

[REFERENCES 27](#_Toc58100741)

LIST OF DRAWINGS

|  |  |  |
| --- | --- | --- |
| **Number** | **Name** | **Page** |
| 2.1 | Remote method calls executed on remote Java virtual machine | 11 |
| 2.2 | Multiple subscription services with one subscription | 12 |
| 2.3 | Stub object calls the framework object | 12 |
| 2.4 | RMI type of program architecture | 13 |
| 2.5 | Role of the name mapping service | 14 |
| 2.6 | RMI Application execution process | 15 |
| 3.1 | ERD Chart | 17 |
| 3.2 | Database Diagram | 17 |
| 3.3 | Admin login interface | 18 |
| 3.4 | Movie ticket management interface | 18 |
| 3.5 | User login interface | 19 |
| 3.6 | Update personal information | 19 |
| 3.7 | Forgot the interface password | 20 |
| 3.8 | Movie ticket management interface | 20 |
| 3.9 | Choose a movie | 21 |
| 3.10 | Ticket booking | 21 |
| 3.11 | Choosing a theater | 22 |
| 3.12 | Choose a movice time | 22 |
| 3.13 | Choose a seat | 23 |
| 3.14 | Paying for movie tickets | 23 |
| 3.15 | Tickets purchased | 24 |

Chapter I: TOPIC OVERVIEW

1.1. Research purposes

Today, the application of information technology and computerization is considered as one of the decisive factors for the activities of governments, organizations as well as companies, it plays an extremely important role. can make powerful breakthroughs.

Along with the continuous development of computer technology and electronic networks, information technology also has high-class technologies and in turn conquers one peak after another. The Internet is one of the products of great value and is increasingly becoming an indispensable tool, the main platform for the transmission and exchange of information globally.

Now, the work of managing, organizing and booking tickets at cinemas is a matter of concern. The manager always faces difficulties in the operation of each department such as movies, showtimes, equipment and especially the ticket buying and selling process. When you want to buy movie tickets, buyers have to go to the theater, wait in line to see if there are tickets available, if there are good seats, if there is a movie they like or not, ... very passive. and annoying and time consuming. time. Therefore, RMI applications that help learners are very popular and widely used because of its convenience and ease of management.

Help save development costs later for software developers deploying software. Improve programming thinking ability, catch up with new technology trends.

For that reason, we have carried out the topic "Building an application to simulate movie ticket booking for cinemas". We will write an application where the manager can generate multiple movie titles along with movie information and allow the customer to make a transaction, the client side can book tickets, buy tickets, watch movies and pay for tickets as required. We carry out this topic with the main purpose of learning and our application builds a part to make cinema management more convenient and faster.

1.2. Research mission

Compared to other methods, ticket management applications are relatively efficient and cost-effective. Moreover, with the connection to the MySQL database, storing information is faster and more convenient.

Knowing that need, we ourselves as information technology students always push ourselves to do something. So we built an application, a simple, user-friendly, easy-to-use system that allows users to quickly access it.

1.3. Research Methods

Methods: Studying lectures in class combined with some documents of Education Publishing House.

Use research methods:

- Method of reading documents.

- Sample analysis method.

- Experimental method

Chapter 2: THEORETICAL BASIS

2.1. JAVA

2.1.1. Java Overview

The Java programming language was originally developed by Sun Microsystems initiated by James Gosling and released in 1995 (Java 1.0 [J2SE]). Up to this point (February 2015) the latest version of Java Standard Edition (JSE) is 8. With its multi-platform advantage, Java is increasingly being applied widely on many devices from PC to mobile and many other hardware devices...

Java is an object-oriented programming language, so it also has 4 common characteristics of object-oriented languages:

• Abstraction: is the process of identifying and grouping properties and actions related to a particular entity, related to the application under development.

• Polymorphism: allows a method to have different effects on different types of objects. With polymorphism, if the same method applies to objects of different classes, it will give different results. The essence of the problem is that this method includes the same number of parameters.

• Inheritance: This allows objects to share or extend existing properties without having to redefine them.

• Encapsulation: is the process of hiding the implementation details of an object from the users of that object.

Besides, Java has some other features:

• Platform Independent (Write Once, Run Anywhere): Unlike many other programming languages ​​like C and C++, when Java is compiled, it is not compiled into machine-specific code, but rather instead bytecode that runs on top of a Java virtual machine (JVM). This means that any device with a JVM installed will be able to execute Java programs.

• Simple: learning Java is actually much easier than C / C ++, if you are familiar with object-oriented programming languages, learning Java will be easier. Java becomes simpler than C/C++ by removing multiple inheritance and pointer math from C/C++.

• Security: Java supports very good security by encryption algorithms such as one way hashing or public key...

• Multithreading: With multithreading Java can write programs that can execute multiple tasks at the same time. This feature is often used a lot in game programming.

• High performance thanks to the garbage collection (garbage collection), which frees memory for unused objects.

• Flexibility: Java is considered more flexible than C/C++ because it is designed to adapt to many development environments.

2.1.2. Why use Java?

The Java language has a very similar syntax to the C / C ++ language (a very powerful language widely used today), but it has been changed quite a lot to meet the ability to be independent of the operating system. onion. The foundation of the Java language are classes. The classes act as objects, the programmer when building the application will use some standard classes of the system, and at the same time can build another class to meet the work requirements.

Java is an easy language to learn, it removes the redundant and cumbersome commands of C/C++ so that programmers can focus on writing programs. At the same time, it also limits programmers to interfere too deeply in the system.

One of the strengths of Java is that Java is highly secure. Another point is that Java is free, Java's programming tools are often many, Java community is very large, plugin creation, open source code is very rich for you to use.

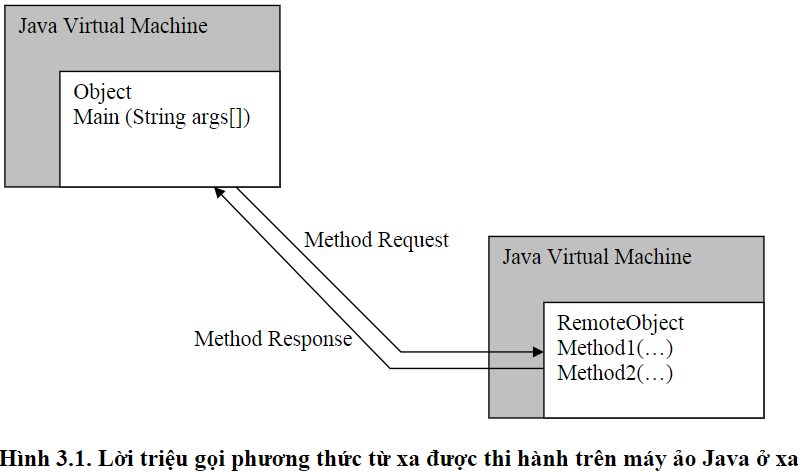
2.2. Overview of RMI

2.2.1. Overview of RMI

RMI (Remote Method Invocation) is a distributed system technology that allows a Java virtual machine (JVM) to call methods of objects located on other Java virtual machines in the same network.

2.2.2. What is RMI?

RMI is a Java technology that allows one JVM to communicate with another JVM and execute methods of the object residing on that JVM.

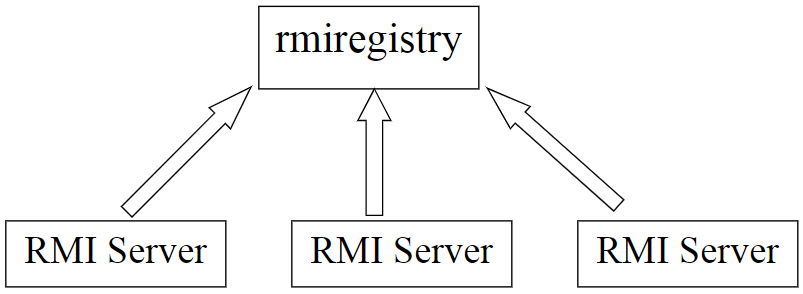


2.1. The remote method call is executed on the remote Java virtual machine

**2.3. How does RMI work?**

Systems that use RMI to communicate generally fall into two categories: Client and Server. The server provides the RMI service and the client invokes the methods provided by the server. The RMI server must subscribe to a search service, which allows clients to find information the server provides, or they can refer to another service.

A background application for RMI is called rmiregistry. This application runs and processes independently of RMI programs, it allows objects on the Server to register their names. Every time an object is registered, it waits and then makes a call from the client side.

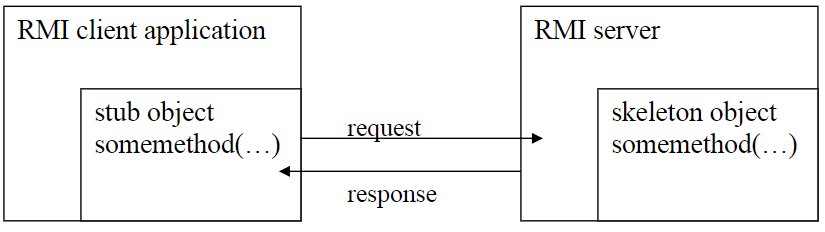


2.2. Multiple subscription services with one subscription

Objects on the Client send messages to remote methods. Before a remote method can be executed, the client must have its reference on the server. That is done by the search service in the RMI registry.

The object on the Client requests the service name and will receive a URL. Remember that for non-HTTP URLs, most protocols can represent using the URL's syntax. The format used by RMI to represent a remote reference object is as follows: rmi://hostname:port/servicename

Where hostname is the name of the server or its IP address, port number is the port number of the service provider, and service name is a string describing the service. The details of network operations are transparent to the application developer when working with remote objects, making it as simple as working with objects at the local machine. This is made possible by the intelligent division of the RMI system into two components, one stub and the other, the framework.

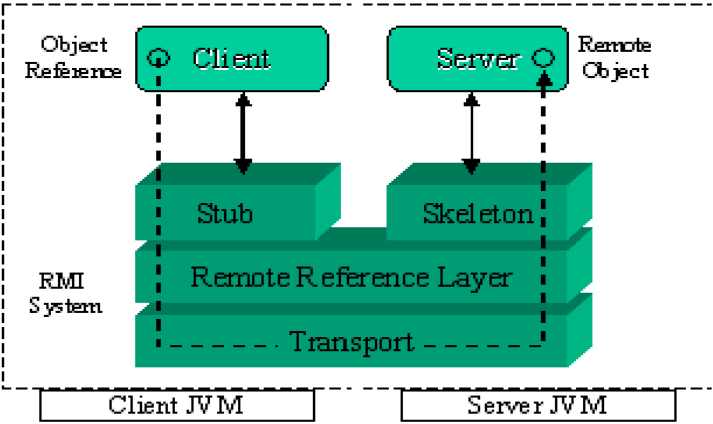


2.3. The stub object calls the skeleton object.

At the RMI Server, the skeleton object is responsible for listening for requests and passing those requests to the RMI service. After a developer creates an RMI interface, he or she must also specify the interface. This defined object will be called a skeleton object.

## **2.4. Architecture of the RMI program**

The architecture of the RMI program mechanism is described as follows:



2.4. RMI type program architecture

In there:

A server is a program that provides objects that can be called remotely.

A client is a program that has references to the methods of remote objects on the server.

The stub contains references to remote methods on the server.

+ Skeleton receives references from Stub to trigger the corresponding method on the Server.

The remote reference layer is the communication system of RMI.

Transport is a transport layer based on TCP/IP protocol between machines in the network. By using the above layered architecture, each layer can be decentralized or replaced without affecting the rest of the system.

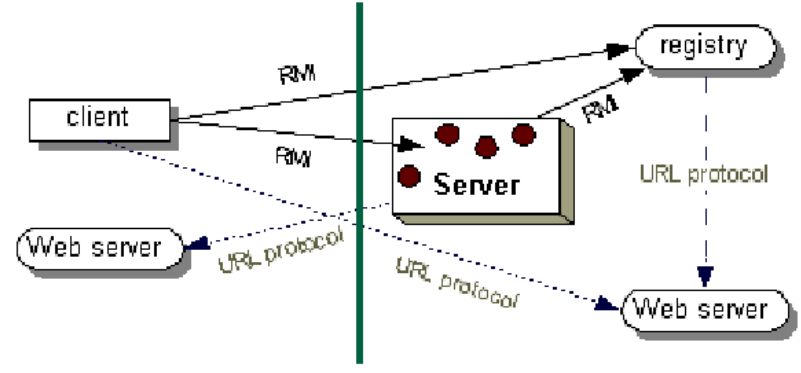
## **2.5. Mechanisms involved in an RMI application**

In a distributed application, the following mechanisms are required:

+ Remote object positioning mechanism (Remote object positioning)

+ Mechanism of communication with remote objects (Communicating with remote objects)

+ Load classes that list bytecodes for classes to be passed between virtual machines (Load class bytecodes for objects passed around)



2.5. Role of the name mapping service

In there:

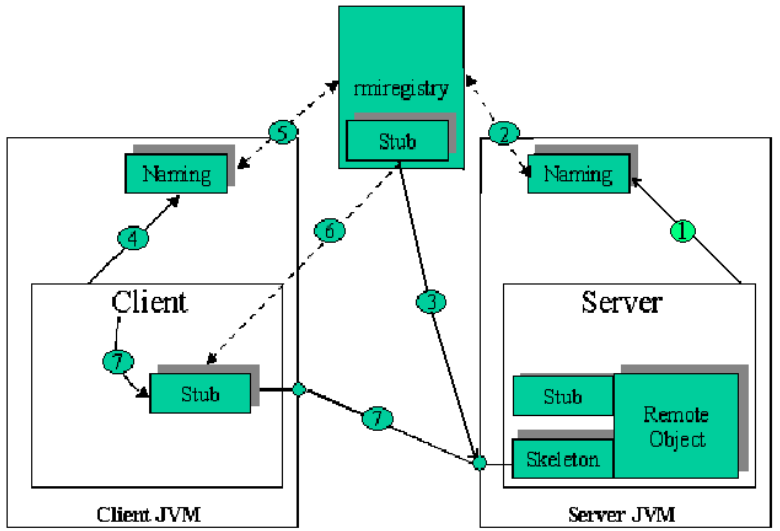
The server registers the name for its remote callable object with the name mapping service (Registry Server).

The client finds the remote object through the name registered on the Registry Server (lookup) and then calls the remote methods.

+ Figure 2.5 also shows how the RMI system uses an existing Web Server to transfer the bytecodes of the layers back and forth between the Client and the Server.

## **2.6. The execution mechanism of an RMI application**

The execution of an RMI application goes like this:



2.6. RMI application execution

+ Step 1: Server creates remote callable objects along with their Stubs and Skeletons.

+ Step 2: The server uses the Naming class to register a name for a remote object (1).

+ Step 3: Set the remote object's stub registry name with Registry Server (2).

+ Step 4: The Registry Server is ready to provide a reference to the requested remote object (3).

+ The client requests Naming to locate the remote object through the registered name (lookup method) with the name service (4).

+ Set the remote object's stub download name from the name service that the object has registered for Client (5).

+ Implement the Stub object and return the remote object reference to the Client (6).

The client makes a remote method call through the Stub(7) object.

# **Chapter 3: BUILDING THE MOVIE TICKET SIMULATION APP FOR CINEMAS**

3.1. System analysis and design

### **3.1.1. App Description**

Identify the actors that use the site as follows:

• Admin: Manage the system, including managing theaters, screening rooms, seats, movies, movie showtimes.

• Customer: The person who uses the software to book movie tickets.

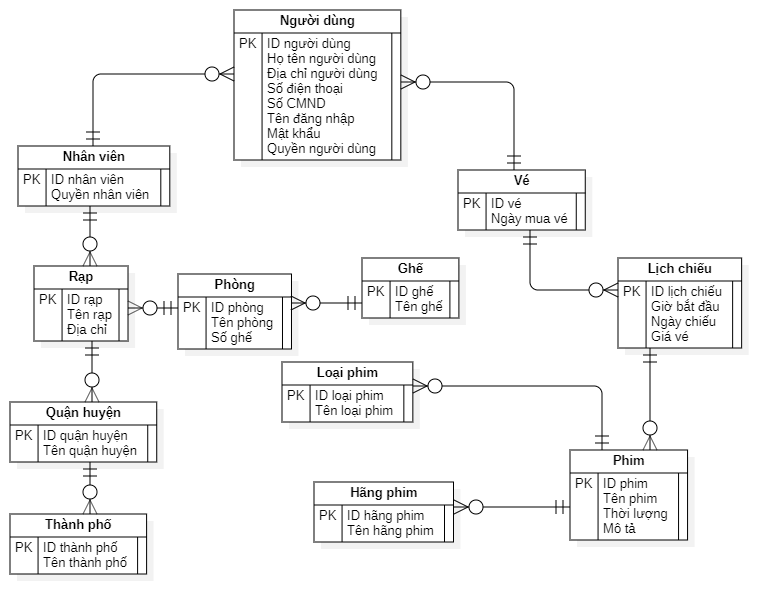
Based on the above factors, our website includes 2 main modules:

1 module for managers, 1 module for customers.

### **Module for admin:**

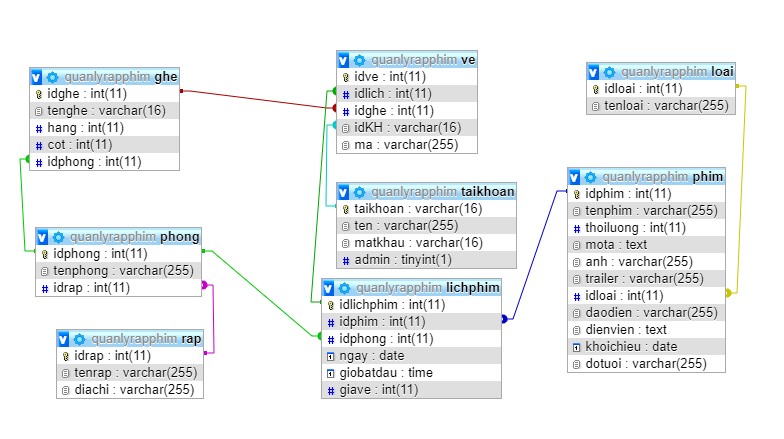
* Displays a list of movies.
* Show rooms list Show theaters by cinema.
* Show sitting position.
* Display the list of movies, detailed information of the movie.
* Show programs according to the schedule of each movie
* Save the ticket and show it to each user.
* **Module for customer:**
* View movie information and select a movie.
* Select the cinema and showtime.
* Choose a seat.
* Book movie tickets and view booked tickets.
* Personal information management.

### **3.1.2. ERD Diagram**



3.1 ERD Diagram

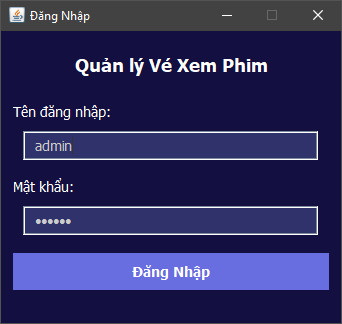
### **3.1.3. Database Diagram**



3.2 Database Diagram

3.4. Application interface

3.4.1. Interface “Server”



3.3 Admin login interface



3.4 Movie ticket management interface

3.3.2. Interface “Client”

Graphical user interface, application

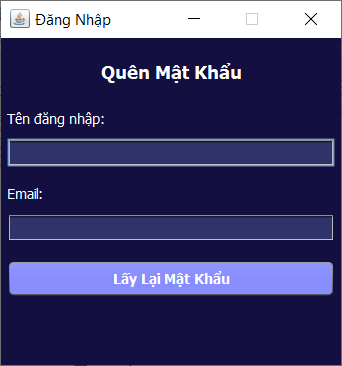
Description automatically generated

3.5 User login interface

Graphical user interface, application

Description automatically generated

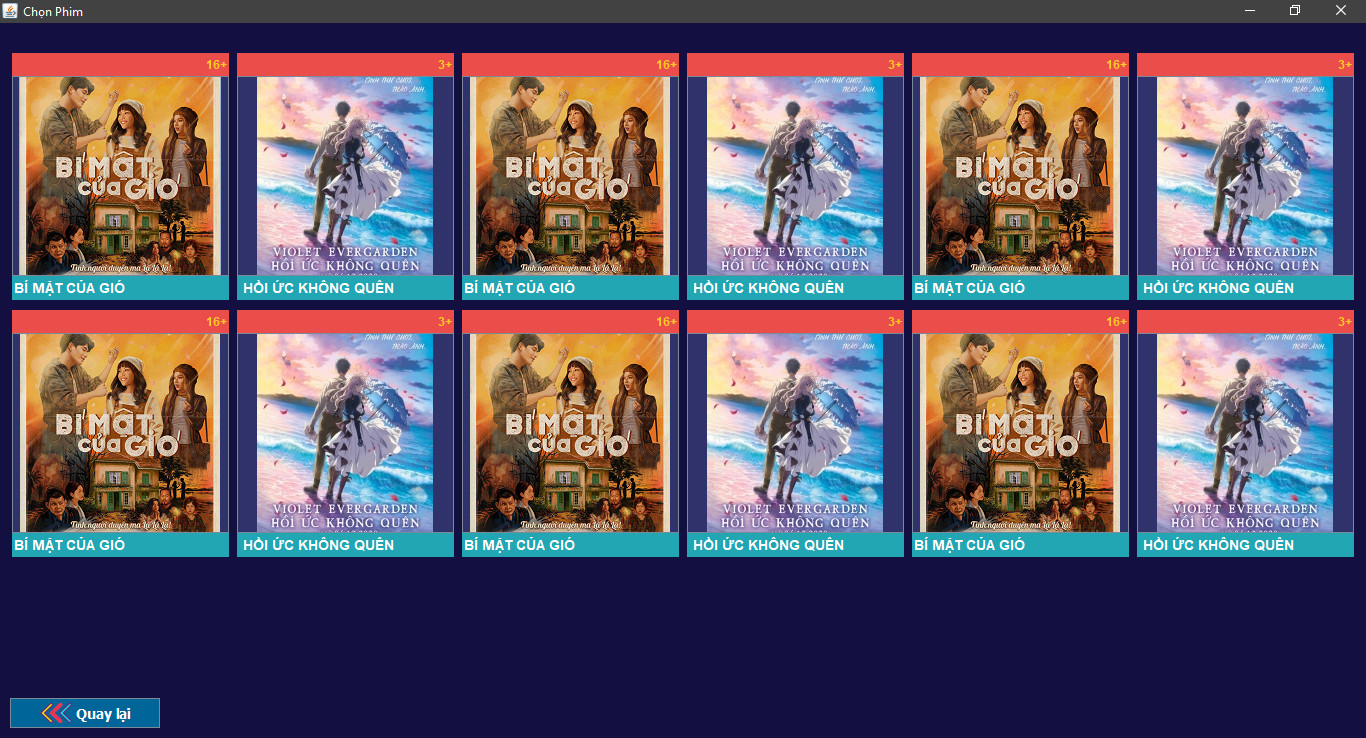
3.6 Update personal information



3.7 Forgot password interface



3.8 Movie ticket booking management interface



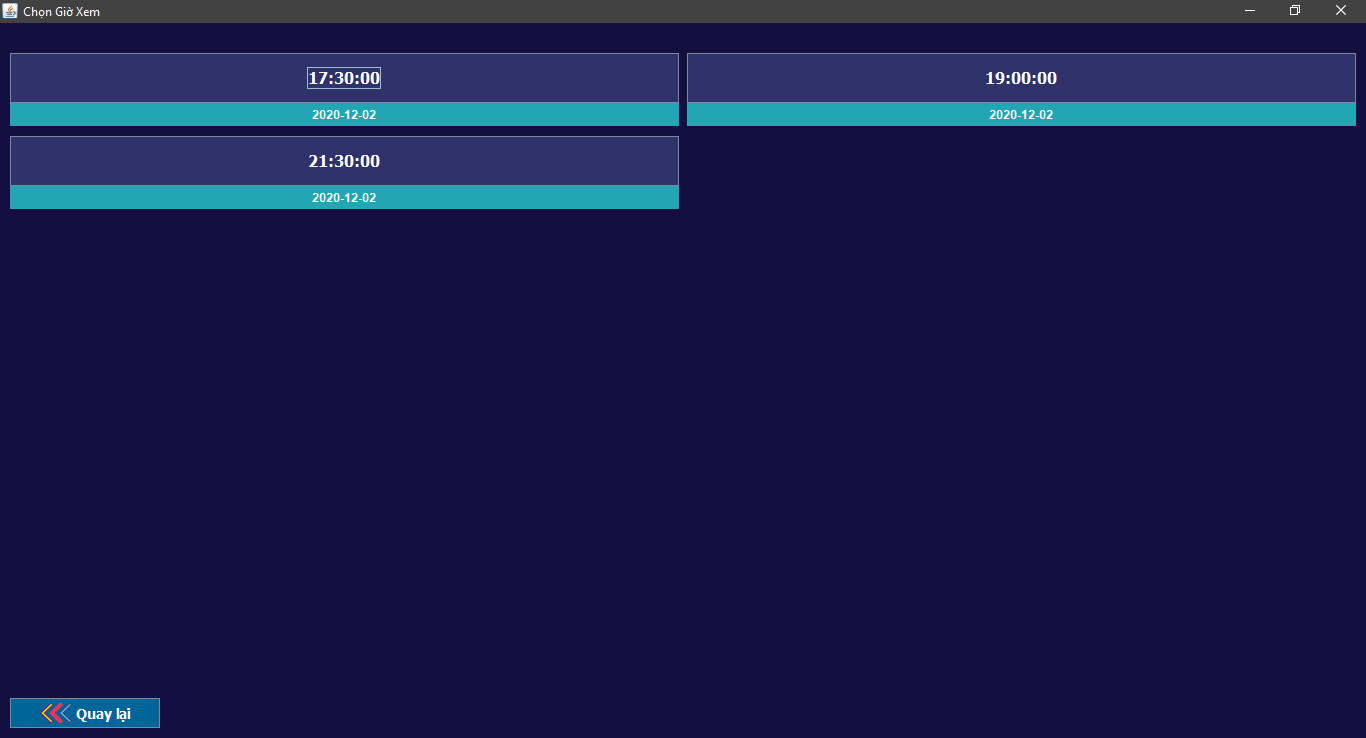
3.9 Choose a movie



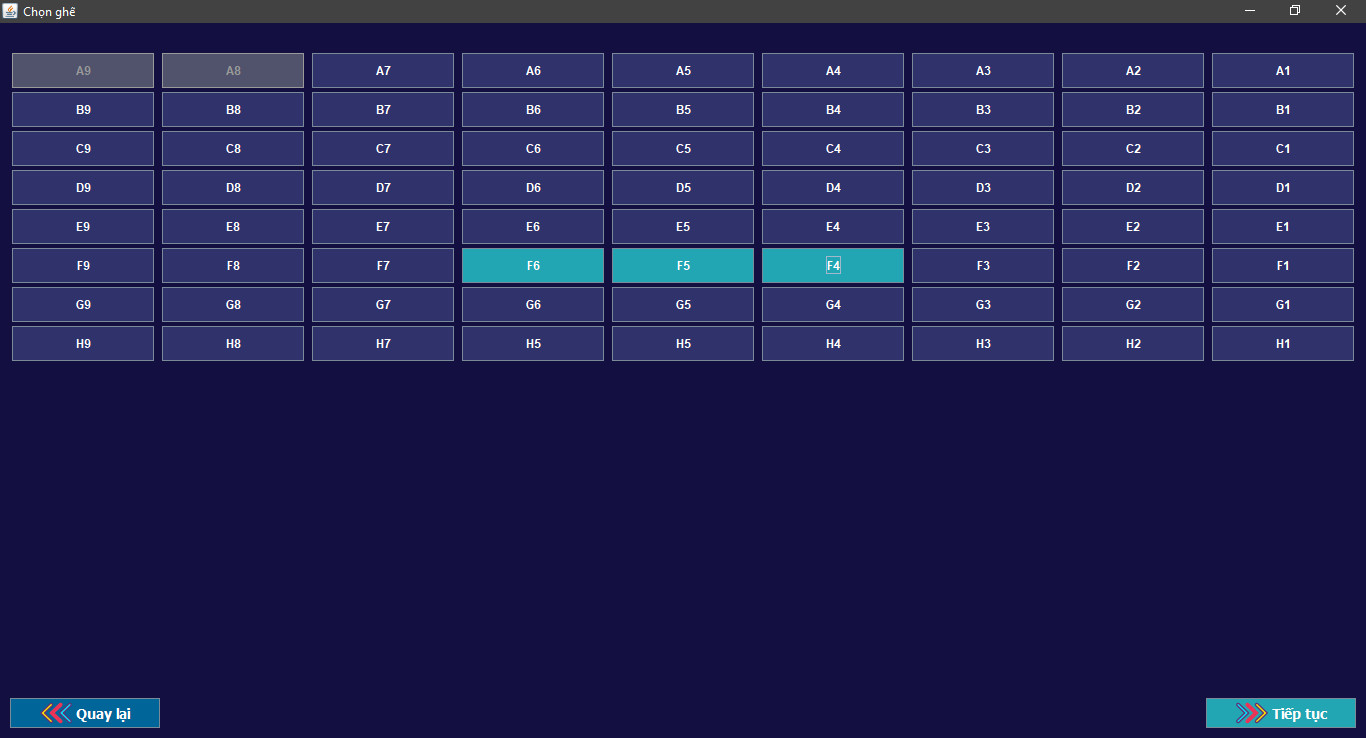
3.10 Book tickets



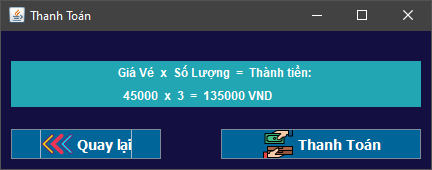
3.11 Choose a theater



3.12 Choose movie time



3.13 Choose a seat



3.14 Ticket payment

Graphical user interface

Description automatically generated

3.15 Tickets purchased

Chapter 4: CONCLUSIONS AND DEVELOPMENT ORIENTATIONS

4.1. Conclude

4.1.1. Reach

* Basically, the project has completed the most necessary and important functions in the project
* Complete software with user-requested functions

4.1.2. Restriction

Besides the advantages that have been achieved, there are also some limitations that we face. Due to the limited amount of knowledge and limited time to implement the topic, we could not achieve the set goal to the maximum, the following are our limitations:

* The interface is not good.
* The functions are still limited, not really creating more functions suitable for users.
* The system model has not been optimized much.

4.2. Recommend

4.2.1. Oriented development

With a basic understanding of the language and tools, we will continue to hone and learn to improve our skills and accumulate the knowledge to build larger scale applications. From there, complete and full-featured applications will be born to serve specific individuals and objects with a better purpose. When writing the application, we also have the opportunity to express ourselves more, we believe that after doing this application well, the next job applications will not be too difficult for us.

As for our app, we will continue to perfect it with the hope that in the not too distant future our app will be complete and marketable.

4.2.2. Orientation to expand and improve the system

In the coming time, we will solve the above limitations and improve more features, design the application page more beautiful, vivid and intuitive, from which the applicability of the application will be improved. high. maximum and efficient.

REFERENCES

|  |  |
| --- | --- |
| [1] | Trần Thu Mai, Lectures on DISTRIBUTION SYSTEM, 2017. |
| [2] | Ngo Dinh Ngoc, Introduction to Java RMI(Remote Method Invocation), <https://viblo.asia/p/gioi-thieu-ve-java-rmiremote-method-invocation-XogBG2xrRxnL,> 2015. |
| [3] | Gia Sư Tin Học, Distributed Programming with Java RMI, <https://giasutinhoc.vn/lap-trinh/lap-trinh-giao-dien/lap-trinh-phan-tan-voi-java-rmi-bai-7-2/,> 2019. |
| [4] | GP Coder, Intructions to connect to the database using Java JDBC, <https://gpcoder.com/5188-huong-dan-ket-noi-co-so-du-lieu-voi-java-jdbc,> 2019. |