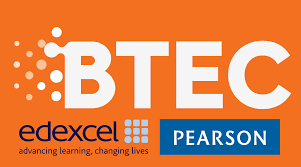
**BTEC Level 5 HND Computing and Systems Development**



A Project Distributed Software Applications Presented

by

Nguyễn Đình Tùng(GC00952)



Submitted to the Graduate School of the

University of Greenwich Vietnam in partial fulfillment

Wednesday, 20th June 2018

Lecturer: Đỗ Quốc Bình

**Table of Contents**

[**Introduction** 5](#_Toc520235498)

[**Task 1: Discuss the principles, characteristics and features of distributed computing** 6](#_Toc520235499)

[1. Definition of distributed software applications 6](#_Toc520235500)

[2. Comparison between distributed systems vs local systems 8](#_Toc520235501)

[3. Challenges in Distributed Systems 9](#_Toc520235502)

[**Task 2: Critically evaluate the impact of distributed software applications delivered on distributed platforms** 12](#_Toc520235503)

[1. Advantages 12](#_Toc520235504)

[2. Disadvantages 13](#_Toc520235505)

[**Task 3: Be able to design distributed software applications** 14](#_Toc520235506)

[3.1: Design a distributed software application for a given problem 14](#_Toc520235507)

[3.2: Explain the components and data and file structures required to implement a given design 15](#_Toc520235508)

[**References** 21](#_Toc520235509)

**Table of figures**

[Figure 1. Model distributed system 7](#_Toc520235393)

[Figure 2. Distributed software development 8](#_Toc520235394)

[Figure 3. Challenges in distributed systems 10](#_Toc520235395)

[Figure 4. UML use-case diagram 14](#_Toc520235396)

[Figure 5. ERD diagram 15](#_Toc520235397)

[Figure 6. Connect database 16](#_Toc520235398)

[Figure 7. Create Entity Classes from database 16](#_Toc520235399)

[Figure 8. Create session bean for entity classes 17](#_Toc520235400)

[Figure 9. Create Web services SOAP 17](#_Toc520235401)

[Figure 10. Web client with SOAP 18](#_Toc520235402)

[Figure 11. UML Deployment detail 18](#_Toc520235403)

[Figure 12. Connect web services from WSDL 19](#_Toc520235404)

[Figure 13. UML conceptual class diagram 20](#_Toc520235405)

**Table of tables**

[Table 1. Compare between distributed and local system 9](#_Toc520235433)

# **Introduction**

This report I will write about discuss the principles, characteristics and features of distributed computing and critically evaluate the impact of distributed software applications delivered on distributed platforms, design a distributed software application for a given problem and explain the components and data and file structures required to implement a given design.

# **Task 1: Discuss the principles, characteristics and features of distributed computing**

## Definition of distributed software applications

* **History:** In the 1960s and 1969s, the IBM mainframe computer was born. It was only for research and learning, and the connectivity between the machines was quite poor. In 1969 introduced ARPANET - predecessor of the Internet.
* In 1970, the concept of distributed applications emerged, and it was a hot trend at the time. Most successful applications were based on ARPANET and began to appear local-area network-Ethernet. In 1973 DCS (Distributed computer system) was released to allow users create system that interconnect minicomputers, provide resource sharing, fault tolerant.
* In 1980 inherner, fidonet, usenet started explode. The universities an researcher were research about DCS and given definition about principles of distributed computing in 1982 and International symposium on distributed computing in 1985. Beside they also given definition about parallel architectures and message passing interface.
* Since 1990, it has grown rapidly and has been rapidly upgraded as Web-services, cloud computing.
* The rugged development of the Internet system has made the computer more close and more effective for people. However, a major challenge for application developers is that network programming is becoming increasingly complex. The traditional single programming model has been greatly altered. Today, you no longer simply write applications to run on a single machine. Application requires user interaction, resource sharing, remote invocation, transaction linkage, data distribution, and so on. With client / server model requests) has been around for a long time (even today). Under the client / server model, all complex processing operations are transferred to the processing server. The client only needs to send the request and display the data
* Multi-tier application development model has been born. Your processing applications are not installed on the client anymore, but installed on another server. The goal is to make the client a lightweight, easy-to-configure, easy-to-change server. If you need to change the source code of the application you just need to change it on a server. All clients that connect to the application server will always be hosted by the latest version of the server.
* “A distributed system is a network that consists of autonomous computers that are connected using a distribution middleware. They help in sharing different resources and capabilities to provide users with a single and integrated coherent network.” [1] this is definition about distributed system. Below is model distributed system

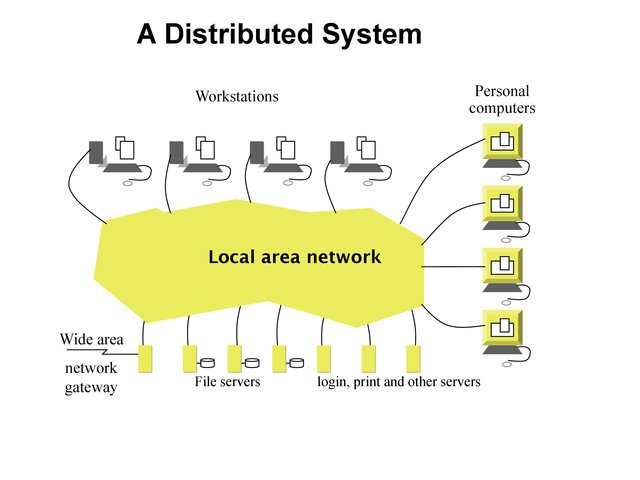


Figure 1. Model distributed system

* “Distributed applications are software that is executed or runs on multiple computers in a network. These applications interact to achieve specific goals or tasks. Traditional applications rely on a single system to run them. Even in the client-server model, the application software must run on the client or server the client is accessing. However, distributed applications run on both at the same time. With distributed applications, if one node is running a specific application is broken, another node can resume the task.” [2] This is definition about distributed software application.

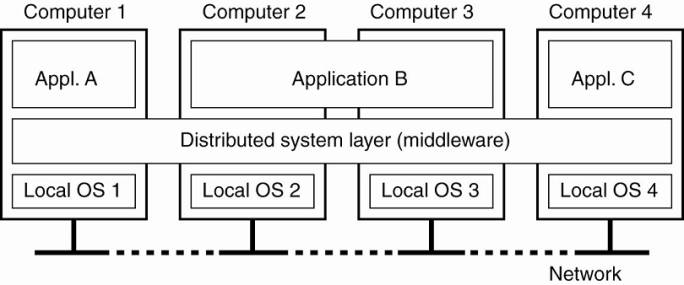


Figure 2. Distributed software development

## Comparison between distributed systems vs local systems

* Below is table compare between distributed system and local system about advantages and disadvantages of both system.

|  |  |  |
| --- | --- | --- |
|  | Distributed system | Local system |
| Advantages | - Resource sharing: A user A can use the user's laser printer B and user B can access the files of A. Generally, resource sharing in distributed systems Provides a mechanism for sharing files in remote locations, processing information in a distributed database, printing at a remote location, using remote devices to perform operations.  - Accelerate calculation: A computational operation is divided into several parts at the same time. The dispersion system allows the division of the calculation into multiple locations for parallel computing.  - Safe: If a location in the scattered system is broken, other locations continue to work.  - Scalability: the ability of a system, network, or process to handle an increasing amount of work, or its potential to be expanded to accommodate that growth.  - Replication: increase data availability and durability on errors.  - Simultaneous: The ability of different parts or units of a program, algorithm, or issue to be performed out-of-order or in partial order. | - The benefit of running locally is that you can work without impacting others. The benefit of the dev central server is that you can check how your changes affect and interact with the changes that other people are making when they are being made.  - Ideally you want to have both. Develop and run your unit tests locally so that you can isolate changes until you are comfortable with them, then integrate them on the team server so that you can test for integration. The change before everything into the test system.  - No internet connection required  - Easily work with most development tools  - Does not affect other developers |
| Disadvantages | - Distributed systems are spread over many other machines. The main difference from the system design perspective is the extent of the errors that can occur in a much more complex distributed system.  - Often the goal in a distributed system is to have a system that can operate while one or more components fail or fail. This adds to the complexity of the first problem.  - Distribution is more complex and costly than on the server and also on the client side when it does not pretend to be centralized, eg without strong consistency.  - Stragglers: a process that takes too much time to produce desired results. | - Depends on the underlying operating system  - May need to run all types of server services on your local machine  - The environment may differ from production.  - Need more resources (mainly RAM, CPU is not a problem with modern hardware virtualization support)  - Some local debugging tools that will attach directly to the server will not work |

Table 1. Compare between distributed and local system

* In different cases, you should choose the appropriate system such as a local system suitable for small management models such as schools or a small company. With distributed systems, you should design for large companies a broad management model that requires the flexibility and scalability of data storage.

## Challenges in Distributed Systems

* Designing a distributed system is not easy, sometimes it raises many challenges for the programmer. Some challenges need to be overcome to get the ideal system. The main challenges in the distribution systems are listed below:

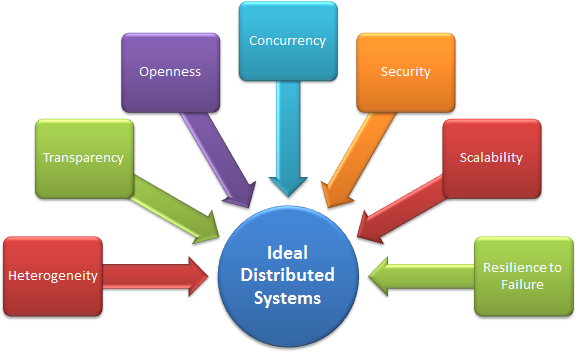


Figure 3. Challenges in distributed systems

* **Heterogeneity:** The Internet allows users to access services and run applications on a heterogeneous collection of computers and networks. Heterogeneity (ie, diversity and differentiation) applies to all of the following:
* Hardware devices: computer, tablet, mobile phone, Embedded device, etc.
* Operating Systems: Ms Windows, Linux, Mac, Unix, etc.
* Network: Local Area Network, Internet, wireless network, satellite link, etc.
* Programming language: Java, C / C ++, Python, PHP, etc.
* Different roles of software developers, designers, system managers.
* Different programming languages use different representations for characters and data structures such as arrays and records. These differences must be resolved if programs written in different languages can communicate with each other. Programs written by different developers cannot communicate with each other unless they use common standards. Below are some terms to help implement the common standards.
* **Middleware:** The term "middleware" applies to a software layer that provides a programming abstraction as well as conceals the heterogeneity of the underlying network, hardware, operating system, and programming language. Much of the middleware is made through Internet protocols, which itself hides the differences of the underlying networks, but all middleware has a difference in operating systems and hardware.
* **Heterogeneous and mobile code:** The term mobile code used to refer to program code can be passed from one computer to another and runs at the destination - the Java applet is an example. The code that is suitable for running on a computer is not necessarily appropriate to run on another computer because executable programs are usually specific to both the instruction set and the host operating system.
* **Transparent:** Transparency is defined as the concealment of users and application developers about the separation of components in a distributed system, so that the system is considered a whole rather than a set of independent components. . In other words, distributed system designers must hide the complexity of the systems as much as possible. Some provisions on transparency in distributed systems are:
* Access Hide differences in data representations and how to access resources
* Location Hide where resources are located
* Moving Hidden resources can move to another location
* Repositioning Hide resources that can be moved to another location using
* Replication of hidden resources can be copied in many places
* Also Hides that a resource can be shared by some competing users
* Failure Hide trouble and restore resources
* Persistence Hides resources (software) in memory or disk space.
* **Openness:** The openness of a computer system is a feature that determines whether the system can be extended and redeployed in a variety of ways. The openness of distributed systems is largely determined by the extent to which new resource-sharing services can be added and provided to a variety of client programs. If the interfaces are clearly defined for a published system, developers will easily add new features or replace future subsystems. For example, Twitter and Facebook have APIs that allow developers to develop their own software interactively.
* **Security:** Many of the information resources that are made available and maintained in distributed systems have high intrinsic value for their users. Therefore, their security is considerable. Information resource security has three components:
* security (protection against disclosure to unauthorized individuals)
* integrity (protection against change or corruption),
* available to the attorney (protection against interference with the means of accessing resources).
* **Failure Handling:** Sometimes the computer system is faulty. When an error occurs in hardware or software, the program may produce incorrect results or may stop before they complete the intended calculation. The handling of the failure is especially difficult.
* **Concurrency:** Both services and applications provide resources that can be shared by clients in a distributed system. Therefore, it is likely that some clients will try to access a shared resource at the same time. For an object to be safe in the environment at the same time, its operations must be synchronized in such a way that its data remains consistent. This can be achieved by standard techniques such as semaphores, which are used in most operating systems.
* **Scalability:** The distribution system must be scalable when the number of users increases. Scalability is defined by B. Clifford Neuman as “A system is considered scalable if it can handle the addition of users and resources without losing noticeable performance or increasing complexity of administration.”
* **Size:** The number of users and resources to process. The incident is overloaded
* **Geography:** Distance between users and resources. Problems related to communication reliability
* **Administration:** As the size of distributed systems increases, many systems need to be controlled. Incidents related to disturbing behavior.
* **Conclusions:** This task I written about history development of distributed system and definition about it. I also compare between two systems this is distributed and local about advantages and disadvantages of each system to help you can choice reasonable for your management system. Beside I also written about challenges in distributed system to help programmer understand more about distributed system.

# **Task 2: Critically evaluate the impact of distributed software applications delivered on distributed platforms**

* As described above, designing a distributed software has many challenges, including the challenges of disadvantages and challenging advantages. Below I will highlight the beneficial challenges and the challenges that lead to the disadvantage of a distributed system.

## Advantages

* **Scalability:** Because distributed systems operate on many different machines, they are scalable. That is, the distributed system can adjust how much system resources it is using in light of what kind of system requirements. If a system has a high demand, it can have all machines running at full capacity. However, if the load on the system is relatively low, it may lose the various components of the offline dispersion system to save power and carry on the system. As demand on the system increases again, these components can be returned online.
* A distributed system is called scalable if it adapts to the size of the system, as shown in the following terms:
* easy to add users and system resources
* when the system changes geographically, resulting in a change in the geographic location of users and resources
* the system has changed the scale of governance
* If the distributed system is scalable, it often affects the performance of the system (the performance of the system is the performance of the object).
* **Openness:** The openness of a computer system is the ease of extending hardware (peripherals, memory, communication interfaces, etc.) and software (OS models, communication protocols, resource sharing services ...). In other words, the openness of distributed systems means the ease of configuring both hardware and software.
* **Performances:** Distributed systems allow greater overall service performance than systems with centralized, single-site functionality. By transmitting the calculated load through different nodes, each location will suffer less stress. This allows each node to perform more efficiently, increasing the overall service performance. An example of how the messaging service works is in high demand. Rather than dump every existing user transaction into a single server, transactions are spread across several different servers. In this way, the demand on each individual node is reduced, and the data that each node receives gets infiltrated into the other nodes in the background.
* **Transparency:** Transparency is the essence of the dispersion system. Transparency of the distributed system is understood as the obscuration of individual components of the computer system (hardware and software) for the user and application programmer. The user has access to the data located at a remote data point automatically by the system, regardless of the dispersion of all data on the network. The system gives the user the impression that the data is stored locally. Transparency is reflected in many aspects, below are some of the most typical aspects:
* **Throughput:** Accesses the local / global object in the same way. The physical separation of system objects is obscured to the user.
* **Through positioning (also known as name transparency**): The user does not recognize the location of the object. Objects are positioned and guided by logical names in a unified system.
* **Transparent immigration (also known as positioning independence)** is a complementarity throughout positioning in the sense that objects not only are indicated by logical names but are also moved to other physical locations. without renaming.
* **Transparent at the same time:** allows the sharing of shared objects without dispute. It is similar to the concept of time division in the general sense.
* **Transparent Mirror:** Provides consistency of multiple instances (or regions) of files and data. This property is closely related to transparency at the same time but is more specific because files and data are special object types.
* **Transparent Parallelism:** Allows for parallel operations where the user does not need to know how that parallel operation occurs, where and when. Parallelism may not be user-specific.
* **Throughout the error:** Providing the fault tolerance of a system is understood as a fault in the system that can be transformed into a reduction in system performance in a more flexible manner, not only minimizing the crash and risk. dangerous to the user.
* **Performance throughout:** try to achieve consistency and assert (not necessarily equal) performance level even when changing system structure or load distribution. Moreover, users are not subject to delays or over-variations when performing remote operations. Performance also demonstrates that system performance is not reduced over time.
* **Transparent size:** related to flexibility and latency. It allows the growth of the system to be obscured by the user. System size does not have an impact on user awareness.
* **During review:** Indicates that vertical system growth is inversely proportional to systemic growth. Software re-observation is obscured by the user. Throughout reviewing is also understood as throughout the segment.
* **Reliability:** When calculations are concentrated around a single machine, the health of the machine is the health of the whole service --- if it is broken, the whole service is the same. However, distributed systems can continue to operate if a node is down. While performance requirements on other nodes will increase, as the stress of each machine is under, other nodes will remain active. However, failure in an important threshold of nodes can still put the service down.

## Disadvantages

* The added complexity needed to ensure proper coordination between web pages is a major disadvantage. This complexity has many different forms:
* **Software development costs:** Difficult to deploy distributed database system; so it is more expensive.
* **Greater potential for errors:** Since the sites constitute a distributed database system operating in parallel, it is difficult to ensure the accuracy of the algorithm, especially in the process of failure of the system and restore from error. The potential exists for extremely delicate errors.
* **Increase in processing costs:** The exchange of information and additional calculations needed to achieve cross-fertilization is a form of cost that does not arise in a centralized system.
* **Security:** Because it uses internet and source open, so it’s easy to hacker attack by other away.
* **Conclusions:** This task I written about advantages and disadvantages of distributed software application. With requirement in scenario I think challenges advantages for system will are scalability, openness and reliability because system will contain many data from users also managers and it will need characteristics advantages as scalability and openness. Disadvantages with this scenario is software development cost and security.

# **Task 3: Be able to design distributed software applications**

## 3.1: Design a distributed software application for a given problem

**1. UML Use-case diagram for scenario**

- With requirements of ABC company, I will design two main roles this is one role intended for staffs to they can management events of company and one role intended for customers to they can hire cameras or lens of company.

- With role intended for staffs they can add new products to system store online and they will have responsibility about this products, they can management accounts of customers. For example when customers register account and send request to staffs and staffs will accept for create account of customers. Outside staffs can view report follow month or in duration they want.

- With role intended for customers they can create a new account and login to system. They can view all products of company on store online, they can reserve products follow quantity and in duration they want, thay can hire products of company with duration default is 10 days. They can view products they reserved and hire and they can hire or return products. Below is UML use-case diagram for system.

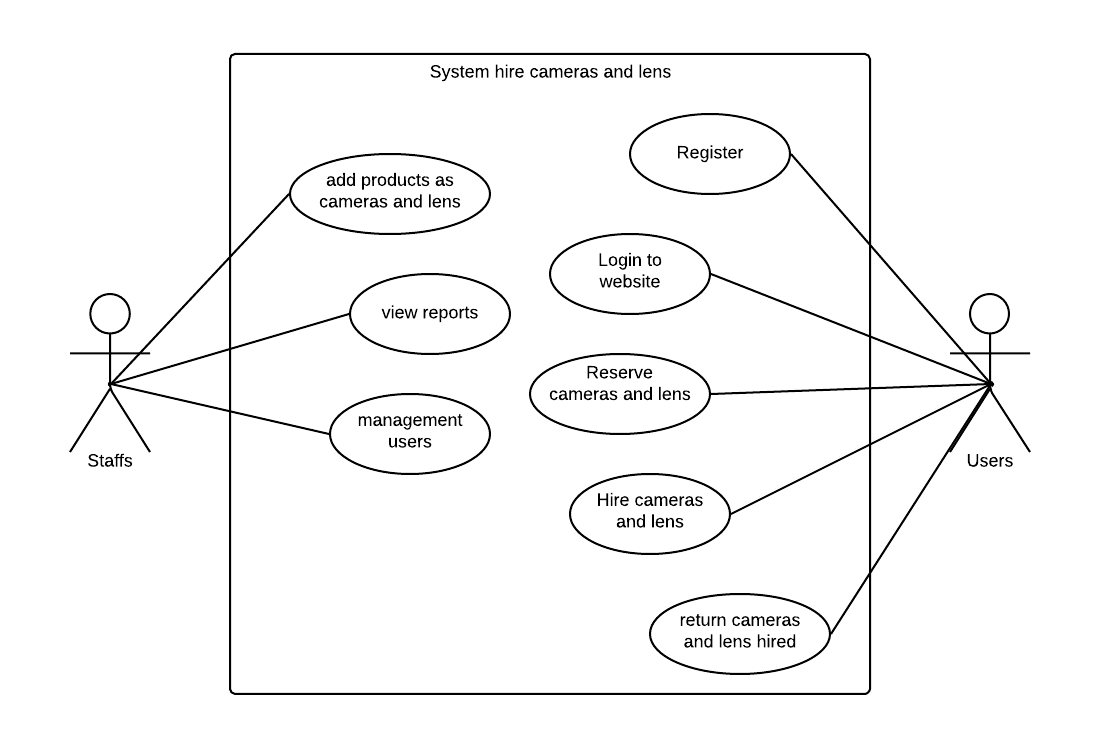


Figure 4. UML use-case diagram

**2. ERD diagram**

- After then determined roles for system I designed database for system with six tables this is: Staffs; Customers, Categories, Manufacturers, Products and Orders, they are related to each other through the foreign keys.

- Table Staffs have relationship 1 – n with table Products because one staff can add new many products to system store online

- Table Customers have relationship 1 – n with table Orders because one customer can order many times

- Table Categories and Manufacturers are relationship 1 – n with table Products. For example same a category is Camera but they split become many products with different quality and durability and they will have different names and different prices. Or when brand can manufacturing many different products…

- Table Products have relationship n – n with tables Orders because one order can have many products or opposite one products can be ordered many time through quantity have in warehouse of store. Below is ERD diagram of database.

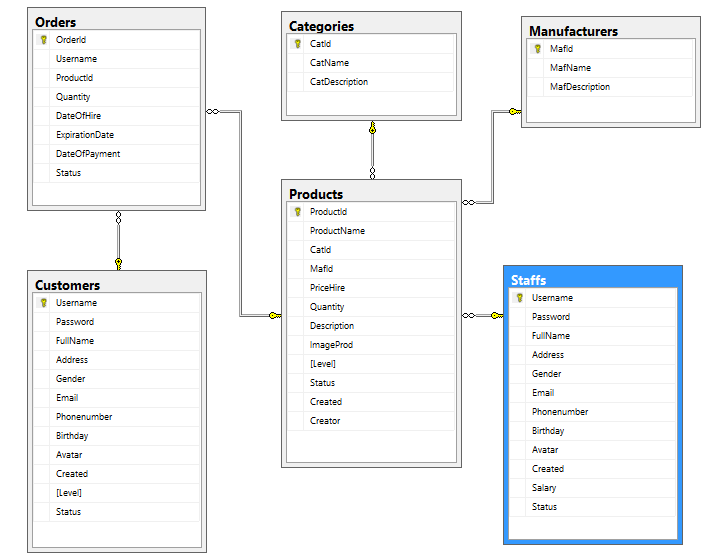


Figure 5. ERD diagram

## 3.2: Explain the components and data and file structures required to implement a given design

**1. UML Deployment diagram detailing my system architecture**

- I designed follow style distributed mean is I designed program with a web services at this program I used services SOAP to create server.

- After then I finished design database I designed Data Model for program by IDE netbeans and used tool create Entity Classes from Database after connected to database and registered JNDI on glassfish server and used tool Session Beans for Entity Classes to create interfaces with this tools I just write code for manipulate the database.

* Below is image to connect to database. This program I used MS SQL Server to design database and I connected to MS SQL Server, with each different tools we will must connect to them by other ways

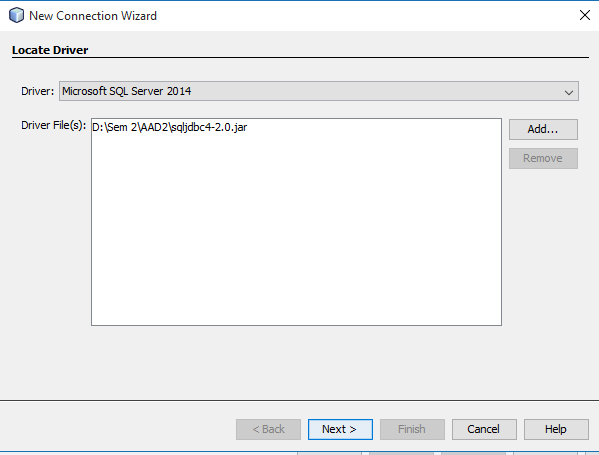


Figure 6. Connect database

* Continue is image to create an Entity Classes from database

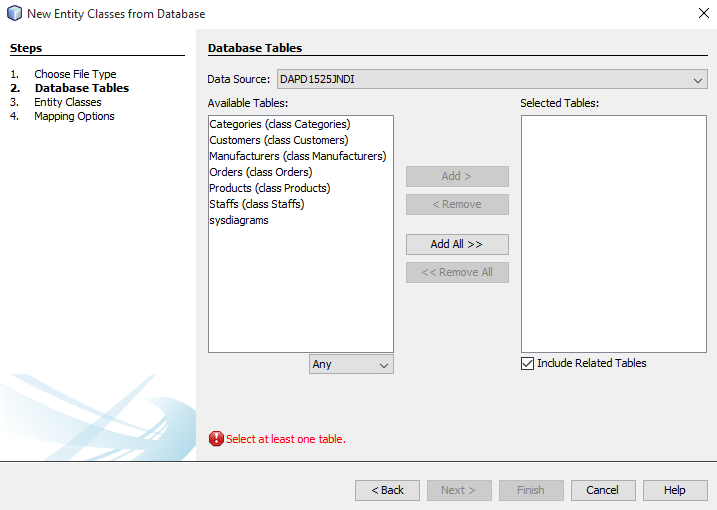


Figure 7. Create Entity Classes from database

* Continue is image to create a Session bean for entity classes

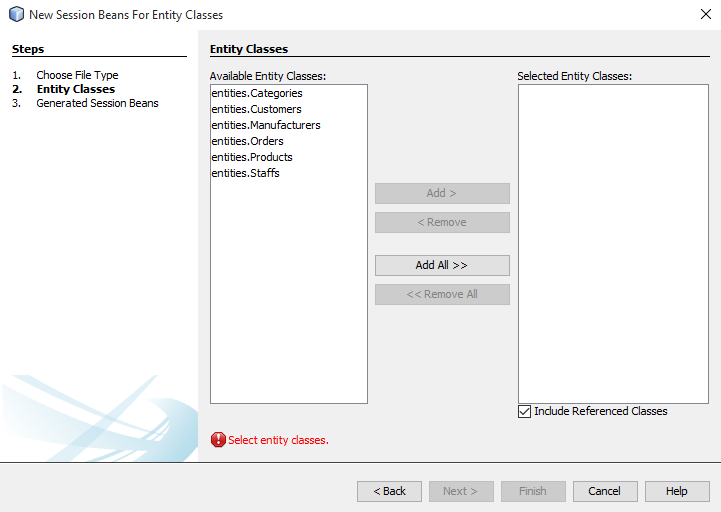


Figure 8. Create session bean for entity classes

- After then finished create data model I created a web services SOAP as image below

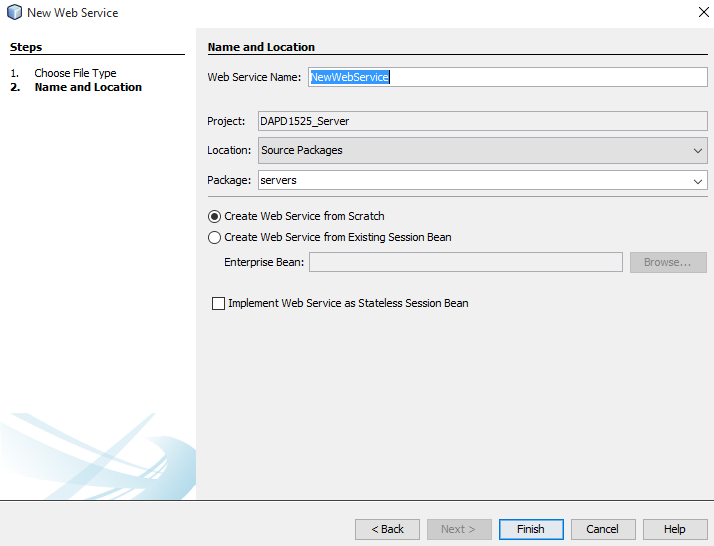


Figure 9. Create Web services SOAP

- Continue I created a web client with WSDL from web services as image below

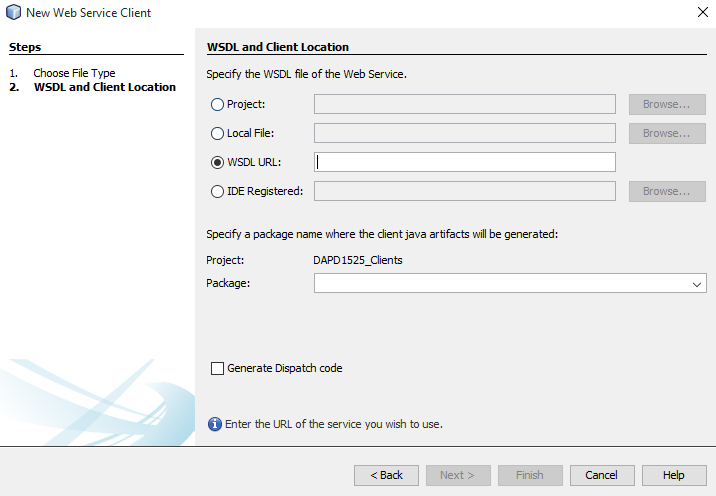
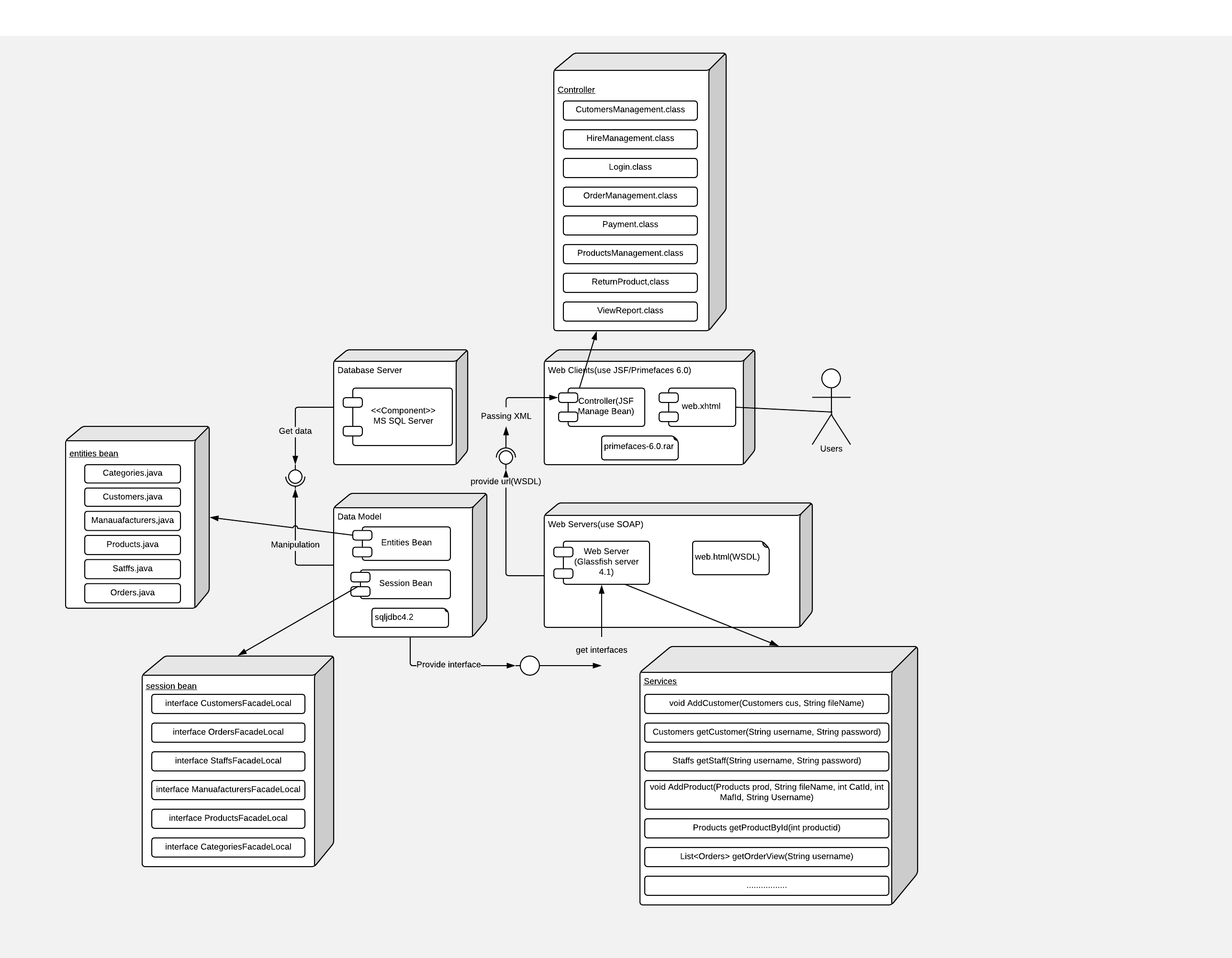


Figure 10. Web client with SOAP

- Continue I designed website to customers can interactive with system. I used JSF / PrimeFaces 6.0 to design website.

- After the finished design program, program will deployment as image below is model process deployment of system.

Figure 11. UML Deployment detail

* **Below I will describe process operation of my program**
* First, when I connected with database by tool netbeans I will create Entity Bean for database netbeans will auto create class corresponding to each table in database and have attribute same with database, continue I will create Session Bean from Entity Classes, netbeans will auto create two classes for each entity, one class is Façade and one class to contain interface. For example with entity Customers, netbeans will create one class is CustomersFacade.java and one class is CustomersFacadeLocal to contain interfaces. Below is code snippet example, one function to manipulate with database in my program.

@Override

public void updateStatus**(**String username**,** int status**)** **{**

Query query **=** em**.**createQuery**(**"UPDATE Customers cus SET cus.status="**+**status**+**" WHERE cus.username='"**+**username**+**"'"**);**

query**.**executeUpdate**();**

**}**

* Continue, at classes Façade I will write code manipulate with database as functions fillAll(), updateStatus() and to overrite to can create interface in classes FacadeLocal.
* Continue, at class services I will must refer to FacadeLocal by away call enterprise bean mean is classes FacadeLocal, after then complete call enterprise bean I will be use functions have in classes Façade as fillAll() or updatStatus() and create functions return for each function. Below is code snippet to call an enterprise bean

@EJB

private CustomersFacadeLocal customersFacade**;**

* After the have web server. I will create client with link WSDL from web server as below image

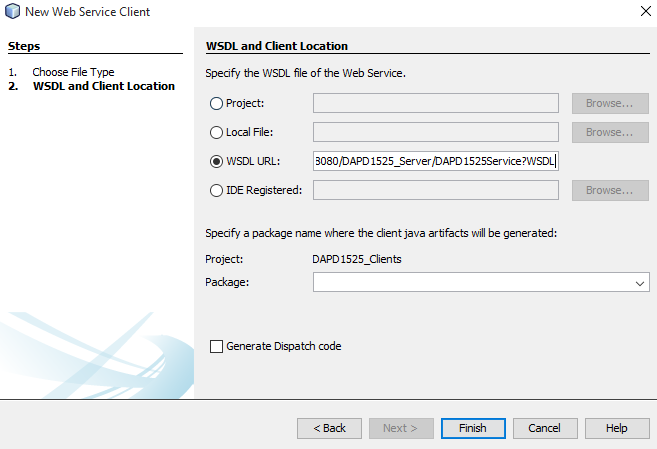


Figure 12. Connect web services from WSDL

* At web client I will create JSF Manage Bean (controllers) from class here I will call web service operation and can call any functions have in web server. Below is code snippet to call web services in controller in my program

@WebServiceRef**(**wsdlLocation **=** "WEB-INF/wsdl/localhost\_8080/DAPD1525\_Server/DAPD1525Service.wsdl"**)**

private DAPD1525Service\_Service service**;**

* Finally, I will create page jsf and get passing data from controllers to page jsf to display to users.

**2. UML Conceptual Class Diagram**

- After then describe process deployment of program I designed a UML conceptual class diagram

- It include detail type of attributed as attributed Username with type is varchar(32)…

- It also include functions need implement of each class as function addCustomer(), AddProduct()…

- Outside it also describe relation of each class with together as relation class Manuafacturers an class Products is relation 1 – 1..\*

- It will help you can understand more how to design model data model

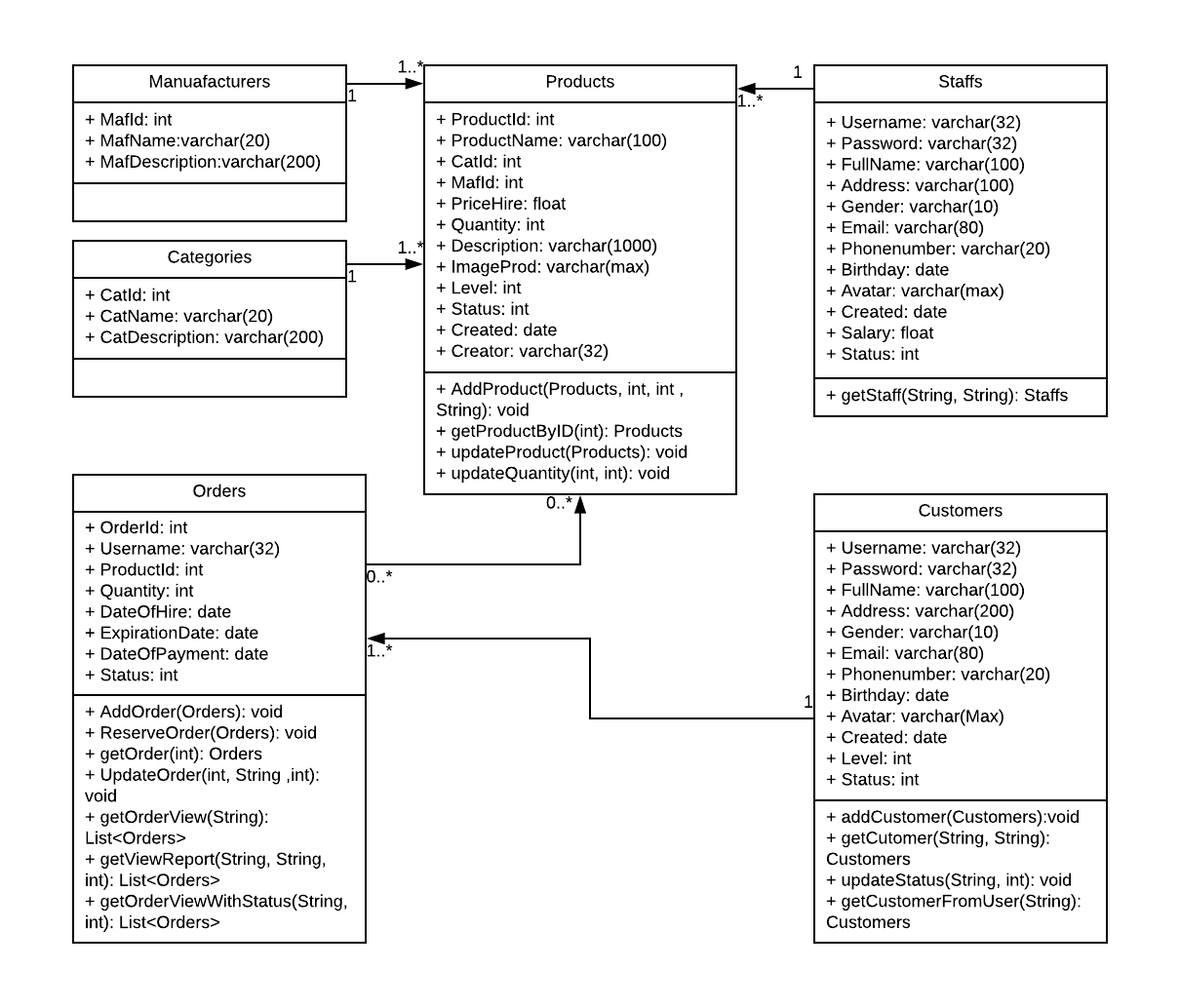


Figure 13. UML conceptual class diagram

* **Conclusions:** This task I written about model design my program by describe process design and give image to you can understand more process I designed. Give for you UML use-case diagram describe roles in program, give UML ERD diagram to you know model database of my program, give UML deployment to you understand process system enforcement, give UML class diagram help you understand about model class in program.

# **References**

[1] Accessed date: June 29th 2018; Title: Definition about distributed system; Link: <https://www.techopedia.com/definition/18909/distributed-system>

[2] Accessed date: June 29th 2018; Title: What is distributed software application?; Link: <https://www.techopedia.com/definition/23971/distributed-application>