ITESM CAMPUS QUERÉTARO

CARLOS ALFONSO SÁNCHEZ ROSALES

A01703280

**FINAL PROJECT: CHAT SERVER**

PROGRAMMING LANGUAGES

ISC11

AUGUST – DECEMBER 2021

**CONTEXT**

Currently, we are trying to live a new life with the COVID-19 coronavirus and accepting that we are changing and the way we communicate, too.

So, during this pandemic a lot of people are in their houses and can’t visit their family, so they have changed its way of communication for example phone and videocalls, chat by WhatsApp, Facebook, Instagram, etc.

But what happen when we are in a zone rural? Maybe that is not something that involve us, but we must think that if we are using this way of communication is because we do not want to get infected.

I found from an article on internet that they interviewed some people that live in rural zone are worried about the impact of the COVID-19, these people are from Hidalgo, Chiapas, Puebla, Chihuahua, Nuevo Leon, Oaxaca, Estado de Mexico, and Jalisco. [1]

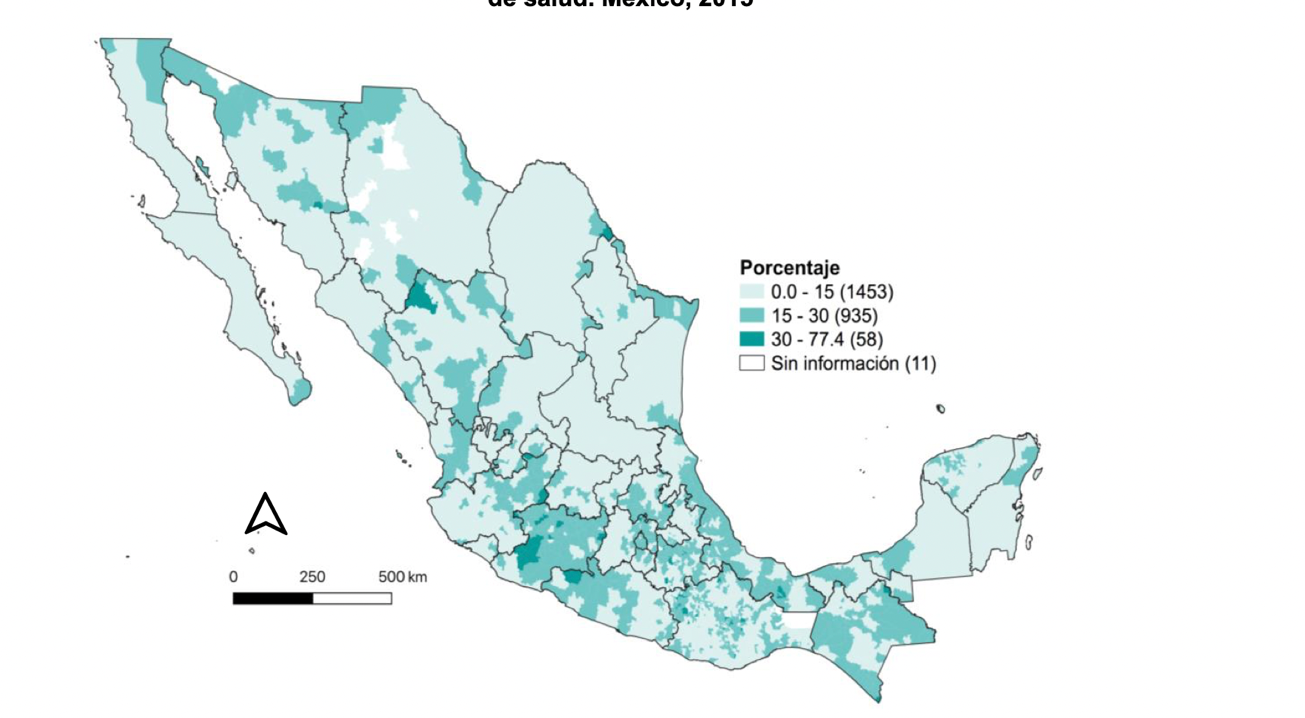


Figure 1: FROM: CONEVAL, BASED ON THE STADISTIC MODEL OF 2015.

In this map (figure 1) we can observe about the percentage of citizens that do not have access to medical services, so if somebody of the rural zone gets infected, he must be in his home and not get out of there.

But we add a new problem, Internet is no available over all Mexico, so many places where we do not have internet are in rural zones, and this is a problem that has existed since many years ago.

According with the Instituto Nacional de Estadistica y Geografia (INEGI) in Mexico the mobile service coverage was of 83.2% and in Instituto Federal Telecomunicaciones (IFT) in the rural zone 50.4% of the rural population is user of internet (figure 2).

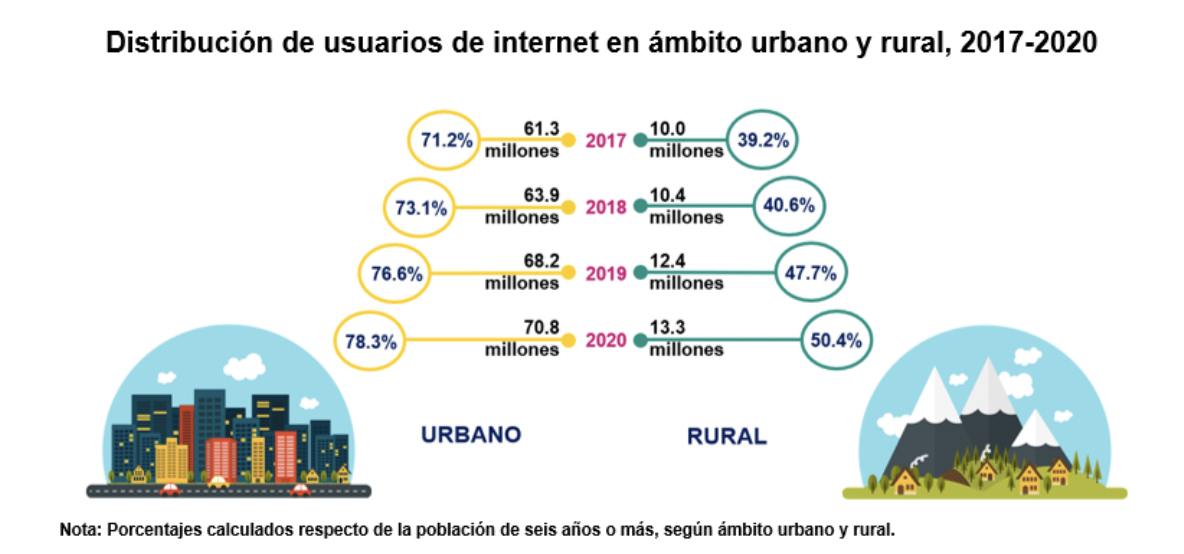


Figure 2: Distribution of the internet users on urban zone and rural zone

In this project what I’m trying to solve is the communication without the necessity to be close with many people to avoid the COVID-19 generating a project functional and easy to use.

**SOLUTION**

Nowadays is easy to get an electronic device like laptop, cellphone, and tablet even we do not have access to internet. For this reason, I did a chat server that can be used in a rural zone and the people can communicate without the necessity of Internet, for example, in Guanajuato capital have many rural zones, one of them that I visited is San Nicolas del Chapin and, in that zone, does not have internet connection and have already of 500 habitants.

We could apply this chat to communication between them and we avoid they can get infected. Also, if the community is big, they do not have to walk and save time.

**HOW WAS DEVELOPED THE SOLUTION?**

In first instance, I will explain some concepts that are important. I used the paradigm of concurrency in Java, but what is concurrency?

Well, concurrency refers to run several programs at the same time, for example if you are writing a text from a paper in computer you are reading and writing at the same time, in computer some examples are:

1. Banking systems, many people are making depositing, or retiring on his accounts at the same time.
2. When you play videogames, all the gamers are making movements, attacking, or defending.
3. Machines in a factory, they must make the productions and do many things to the same time to allow all the production is completed at time.

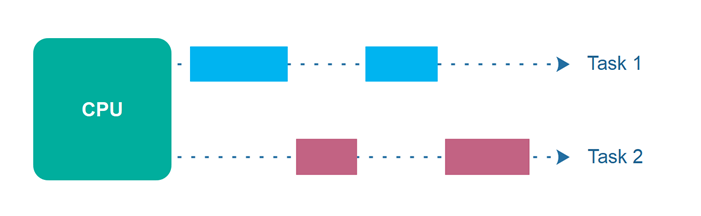


Figure 3: Example of threads.

For a better visualization we can in the image above observe how in the computer are running two tasks on the same CPU, for this we use Threads (figure 3).

A thread, in the context of Java, is the path followed when executing a program. All have at least one thread, mainly known as the main thread, it is created automatically by the JVM. These threads allow us to run some code, that later we will see.

We have **process** that in few words, a process is a series of actions or steps to achieve a particular goal (figure 4).

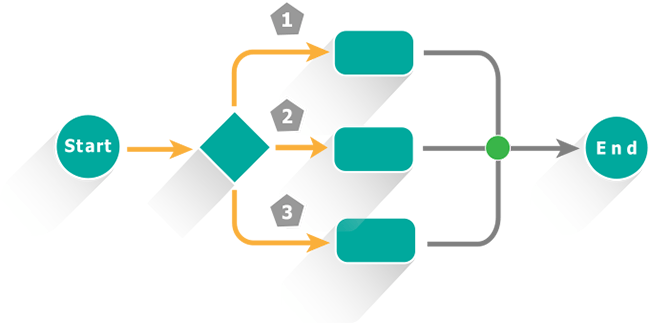


Figure 4: Example of a series of steps(process)

Also, to allow communication we use sockets and server sockets.

A **socket** is an endpoint for communication between two programs running on the network. This receives the port number so it can identify the application that data is destined to be sent to.

A **server socket** helps us for providing system-independent implementation of the server side of a client/server. So, the server listens the port and with this we can stablish communications with the clients.

Now, we have understood some concepts that are relevant to develop the solution, we can continue to explain the development of the solution.

I used Eclipse and as I mentioned before I did it in the programming language of Java to show the used of concurrency paradigm, since this is the best way to solve the problem because it allows the connections between many users and the internet is not required.

**GUI**

For the view of the user, I used **JFrame and JPanel** from the library swing, this is a basic GUI, and it will allow most of the computers can use the chat service. So JFrame basically helps us to represent all the things that should go on the window, and the configuration of the window like borders, sizes, etc. In the other hand, we have JPanel that is a generic class that join all the elements like, labels, buttons, text areas, etc. To show us on the JFrame.



Figure 4: Code to generate the view of the user.

In the image above show the code of the view of the user , where we as mentioned before the JFrame to develop and easy and understandable interface, so we use labels to show the name of the client, the combox where the user will see the connections, and in few words we have our field to see all the conversation, making possible to scroll over the window, the input to receive the message, adding a listener to know when the user wants to send the message.

The code is generated in files of the type of java, so everything is putting on the location according with the order we indicate in the code, a clear example is in a photo of the view (figure 5) seeing the name of the user is the first thing we are possible to see.

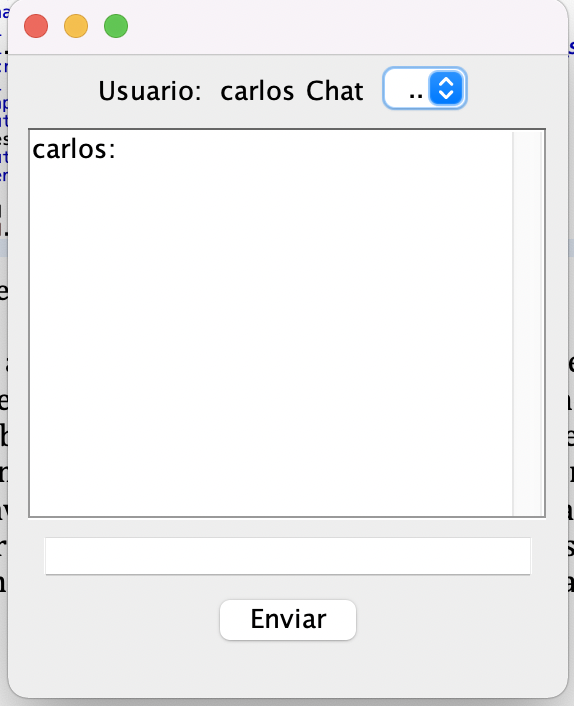


Figure 5: View of the chat interface.

To see the window, we have the class **UserView** (figure 6) that extends of **JFrame**, basically we assign the size of the window, and it makes we can see all the elements like, text areas, inputs, etc. that are because we add the layout that we constructed before (figure 4), we make visible to the user and we add some listeners to know when an client is online.

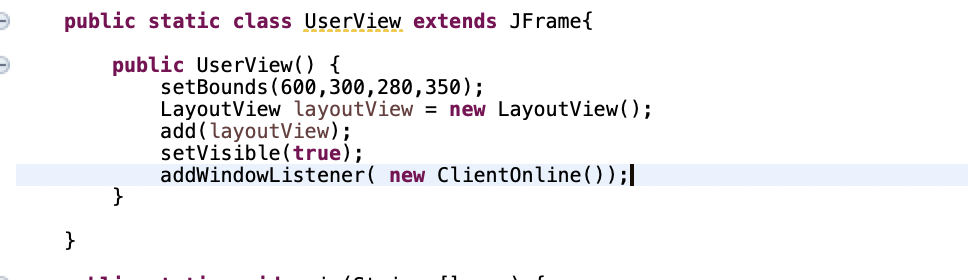


Figure 6: UserView class

**How to make the chat server functional?**

**Side of the Client**

In this the part important is that we use **sockets**, and as I explained it allows two make a connection between two machines to send a message, so the first thing that the system would do when we click on “enviar” is to put the message we sent on our view. It adds to the existed conversation, getting the text with the method getText (figure 7).

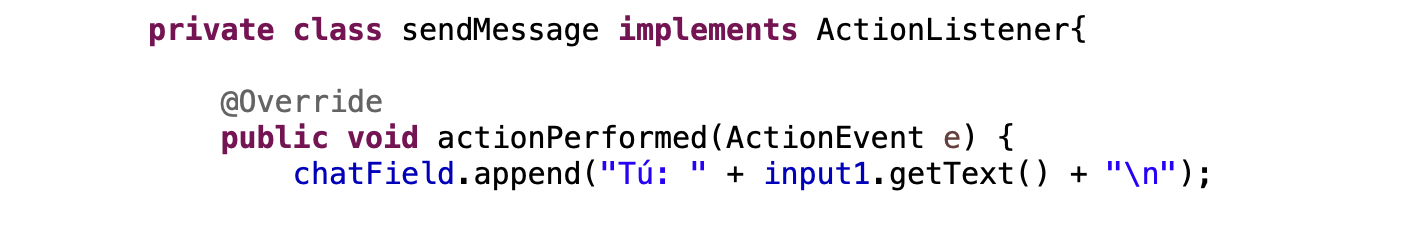


Figure 7: Sending the message

Then this is one of the most important functions (figure 8), where we make the connection to the server using the socket and the port that is listening, so according with the documentation of java this can generate an error, so we handle it with the try-catch to do not crash the problem, this can be from an error of the ip address, or the ports that is not listening, there is a const of using sockets.



Figure 8: Code of the use of sockets.

Then, we know we have to send to an ip address from my user a message, so that have to be in a package to send the information to the server, so we create the information of type **PackageSent** (it will be explain later), and we set the information that I mentioned before, then create the ObjectOutputStream that writes primitive data types that can be read on an ObjectInputStream, so we write the information to the pack and is sent, finally we clear the input and close the socket.

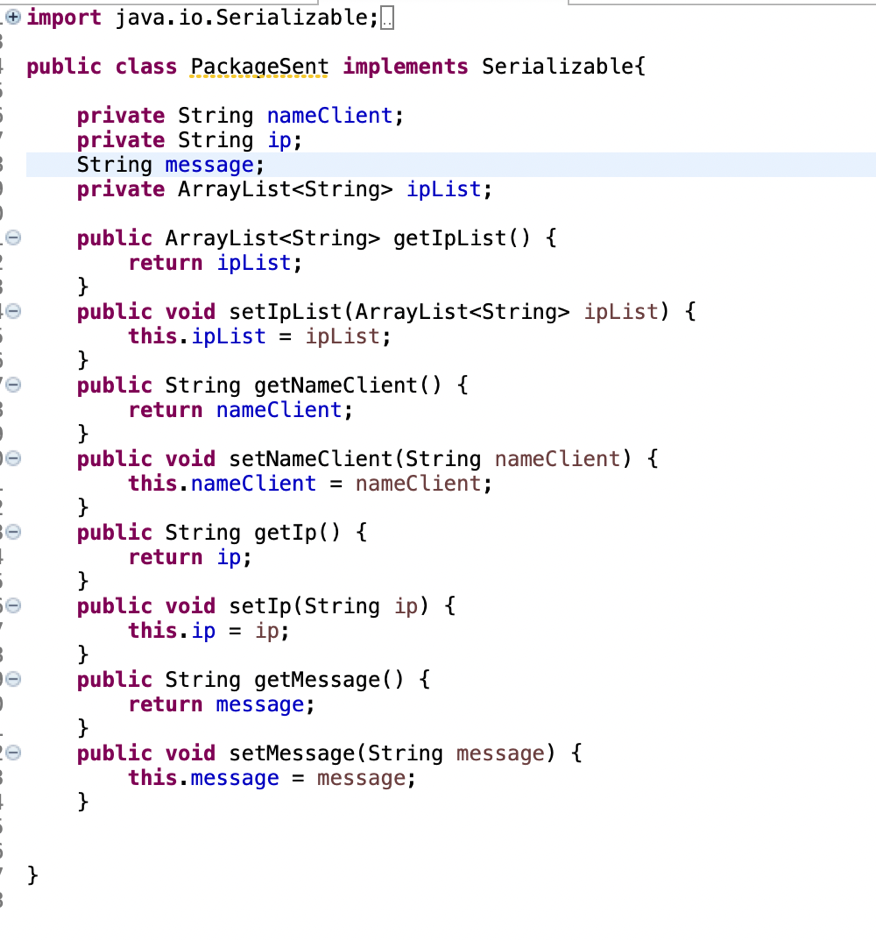


Figure 9: Integration of the pack that is sent.

Now, we are going to talk about **PackageSent** (figure 9), that is a class of java that I created with the purpose of be able to send the information in only one pack, so if you can see we have de name of the user, the ip, message and an array list of ip, this last is because we need to update when someone new connects, so in this way we update the list of users that are online and we can send messages.

But if you can see at the top where we create the class, we use **serializable,** why? Well, almost we know that we use binary system on a network, in our own computer, so it only can read bytes and we do not have to do anything because the method defaultWriterObject of the class ObjectOutputStream makes the serialization so in this way the pack can travel on the network and be receive for the server and sent to the addressee.

To listen the messages receives, we have to always listening here is where coming concurrency that we will explain later.



Figure 10: Listening the messages.

If you can see in the image above (figure 10), we are listening with the server socket, so the first step is always to listen and accept the connection if the message received is online, then we know that have a new user and update our list of ip that are able to receive messages.

When we know that is connected then get the input, so as we know that comes in a binary type is necessary to cast to the type of PackageSent that we explained before, and we get the message and from what user was sent, so we append to the area text, and we can see the message.

**SERVER:**

Well, to allow the communication between machines we have to have a server that handle the packs received and sent to the final user, for this we have our class server in java.

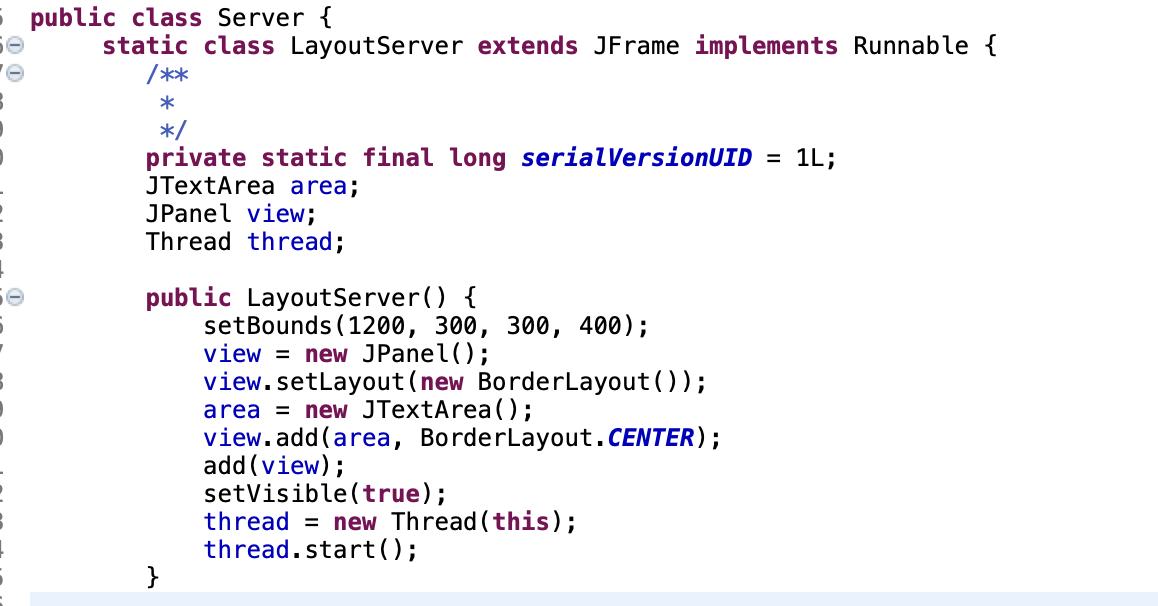


Figure 11: Frame to have an interface of the server.

First, we did an interface (figure 11), that is why we extends from JFrame again because we can see in this what it receives from the user.

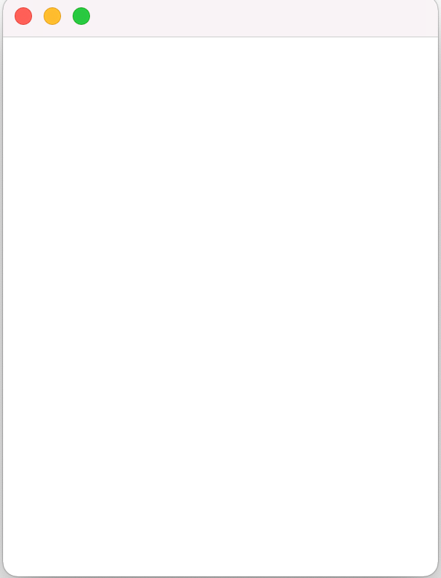


Figure 12: Frame of the server



Figure 13: Listening the server to the users.

In second way, we have the override method **run** (figure 13) that I will explain later, in this first part we focus on the server socket, that has the functionality of wait the packages that comes from the network, so the server is always listening we always receive a message from the user that is online, to update our list of ip that are available to all the users.

In this case we localize the ip of the user using **getInetAddress** to get the direction, in that way we know the ip has to be saved on string format, so when we get that, it uses the method **getHostAddress** that allow to ip on string format, so we save on the array and send a pack to all the user because they need to know someone new is connected and update their lists.

It does for every user connected, and it is constructed like before we mentioned is the package made.

When it knows that the users are already connected then we generate a socket to make the communication and be able to send the message, so in and object of the type **ObjectInputStream** read the message received from the user and to know the ip to the addressee user, so in this way we create the connection with the socket and add the parameters that is the ip direction and the port.

Then it writes the pack and send the information, and finally close the connections.

**CONCURRENCY**

Well, making a brief review I said that I used concurrency paradigm, this because we have to allow a connection and a service of the chat that is always listening and sending too the information we want.

So, to achieve this, we did use of **Threads**, so every time we open a chat, we run a new thread, that allow to always run, if you remember we saw it in the figure 13, where we override the methods and that’s why we implement runnable.

Then, as I many times mentioned before this allows always the program run, and this allow us to chat with other users. For this we create a thread and we always run that class, where we have the methods to be able to chat.

So, to make it runs we use the methods starts, and a clear example we have in the figure 14.

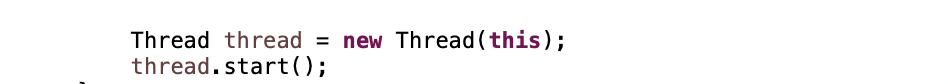


Figure 14: Creating a thread.

The same is to the server if you can observe in the figure 15, we make the same to the thread so it allows always to run and be able to make the connections between machines.

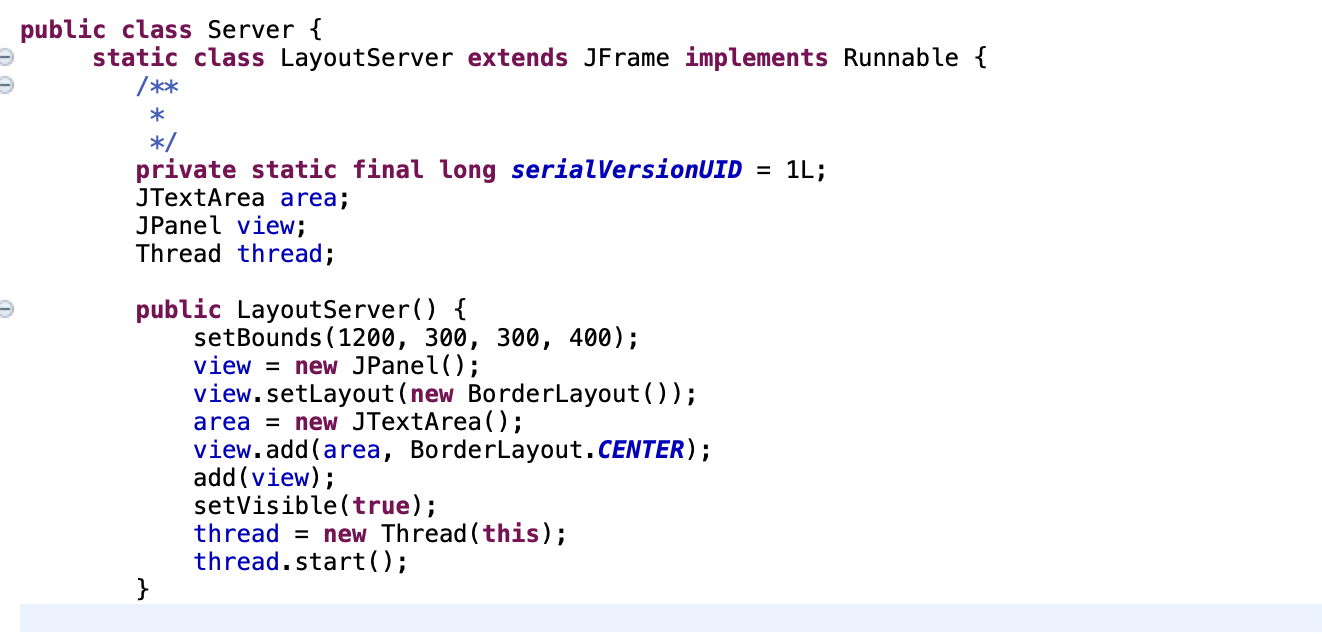


Figure 15: Creating thread on the server.



Figure 16: Diagram of the threads and sockets

Deploy and Test

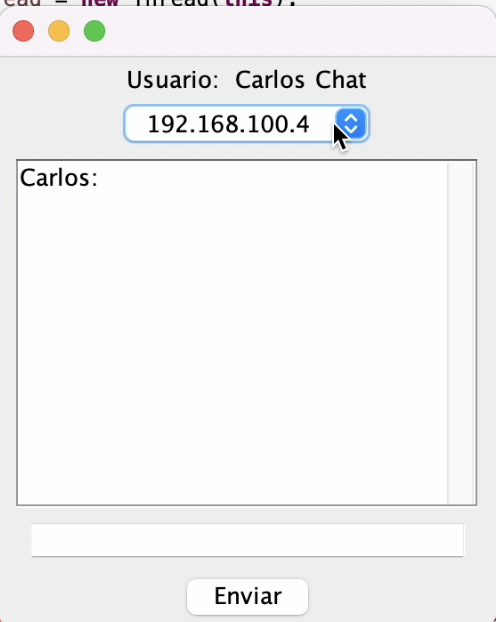
In first instance to be able to run the chat, we have made a runnable jar file, it means that only requires the JDK to run the chat without the necessity of Eclipse.

We have to write our name and click on Ok, to start the chat.

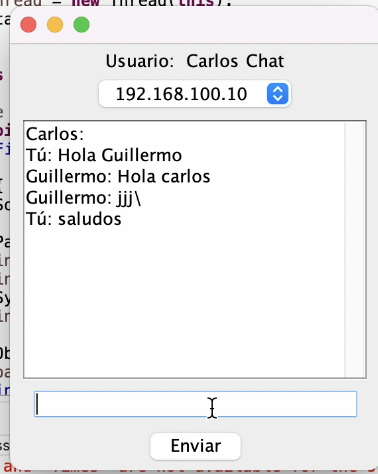


Figure 17: Writing username.

Then, we will see the interface of the chat, an every time someone new connects to the chat, its ip will be available on the list, and you click over it, you will start to chat.

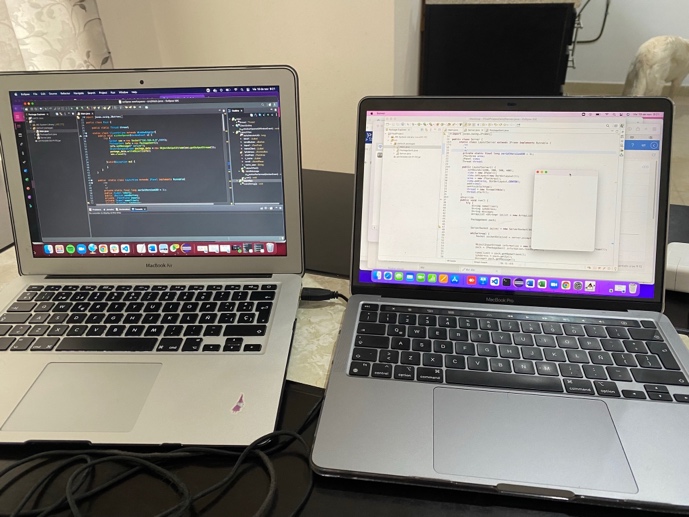


I did some tests between my computer and roommate’s computer, and we have a good result, and we can chat.

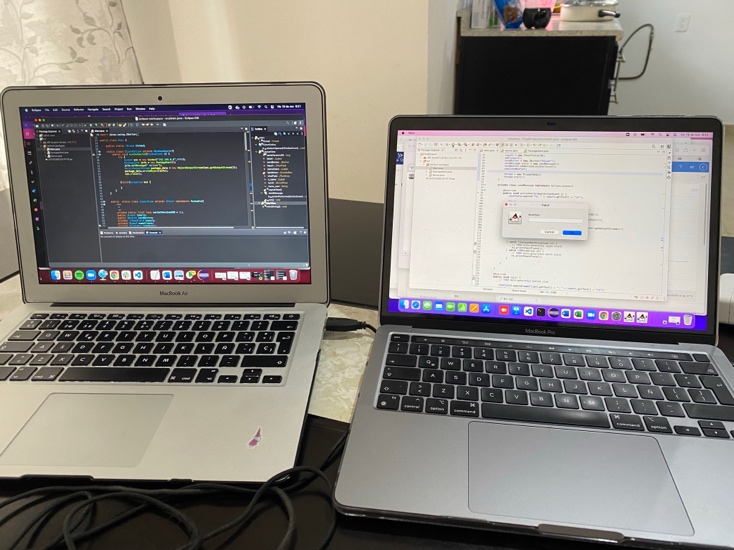


Here, we can see the chat, and it worked without errors.

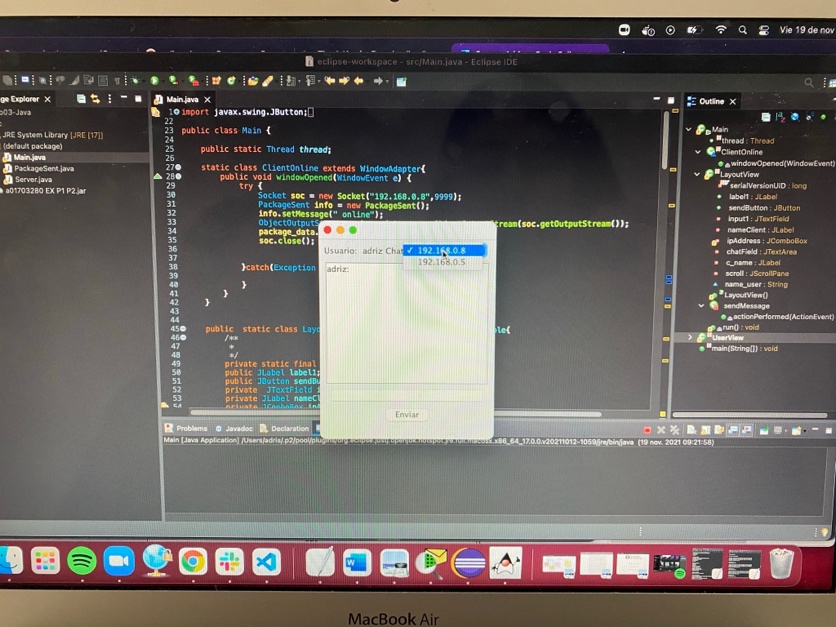
The I did another test with my friend’s computer and different ips and it worked fine, too.



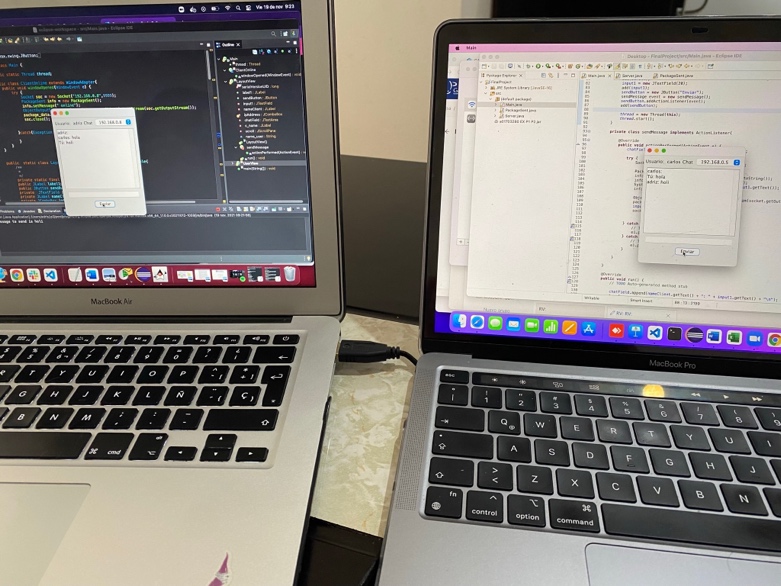
I started the server.



I opened the first chat.



Then my friend opened the chat, and we can see our ips.



And finally we can chat.

**CONCLUSION:**

With the development of this application, we can notice that we can do multiple things with different paradigms, in this case, as we know we use the concurrency in Java and give us a great result to all the project, adding that we can apply to solve a social problem.

Finally, a communication nowadays is important, and we have to achieve or at least to try that all the people can be communicated, so in this way can help us to the development of our society. Java was a good option because it is something that all the computers can run, so since have a simple interface the users will not have problems.

REFERENCES

Unknown. (2018). Runnable interface in Java. Noviembre 17, 2021, de GeeksforGeeks Sitio web: https://www.geeksforgeeks.org/runnable-interface-in-java/

Uknown. (NA). Leer y escribir objetos. November 17, 2021, de EHU Sitio web: http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/archivos/objetos.htm

Oracle. (NA). ObjectOutputStream. November 17, 2021, de Oracle Sitio web: https://docs.oracle.com/javase/7/docs/api/java/io/ObjectOutputStream.html

Cervigon, C. (NA). Interfaces gráficas. November 17, 2021, de FDI Sitio web: <https://www.fdi.ucm.es/profesor/jpavon/poo/tema6resumido.pdf>

Unknown. (NA). Java Socket Programming. November 17, 2021, de Javat point Sitio web: https://www.javatpoint.com/socket-programming

Unknown. (2021). November 17, 2021. Introduction to Concurrency, de LMU Sitio web: https://cs.lmu.edu/~ray/notes/introconcurrency/

Jenkov, J. (2020). Concurrency vs. Parallelism. November 17, 2021, de Jenkov Sitio web: http://tutorials.jenkov.com/java-concurrency/concurrency-vs-parallelism.html

How to do in java. (2020). Java Concurrency Tutorial. November 17, 2021, de How to do in java Sitio web: https://howtodoinjava.com/java-concurrency-tutorial/

Vazquez, R. (2021). Así podría México lograr que toda su población tenga cobertura de internet. November 17, 2021, de Forbes Sitio web: https://www.forbes.com.mx/negocios-mexico-lograr-que-toda-su-poblacion-tenga-cobertura-de-internet/

Instituto Federal de Telecomunicaciones. (2021). En México hay 84.1 millones de usuarios de internet y 88.2 millones de usuarios de teléfonos celulares: ENDUTIH 2020. (Comunicado de prensa) 22 de junio. November 17, 2021, de Instituto Federal de Telecomunicaciones Sitio web: <http://www.ift.org.mx/comunicacion-y-medios/comunicados-ift/es/en-mexico-hay-841-millones-de-usuarios-de-internet-y-882-millones-de-usuarios-de-telefonos-celulares>

Navarrete,T. (2021). la política social en el contexto de la pandemia por el virus sars-cov-2 (covid-19) en méxico. November 17, 2021, de CONEVAL Sitio web: <https://www.coneval.org.mx/Evaluacion/iepsm/Documents/Politica_Social_covid-19.pdf>

Carrasco, H. (2020). COVID-19 en el México rural. November 17, 2021, de Nexos Sitio web: https://redaccion.nexos.com.mx/covid-19-en-el-mexico-rural/

Wen, T.. (2020). Our relationship with voice and video chat is changing in the Covid-19 era. How should those who dislike phone and video chat handle the influx of calls?. November 17, 2021, de BBC Sitio web: https://www.bbc.com/worklife/article/20200408-coronavirus-how-lockdown-helps-those-who-fear-the-phone

Ha, N. (2019). How to Create a Chat Console Application in Java using Socket. November 10, 2021, de CodeJava Sitio web: https://mail.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket