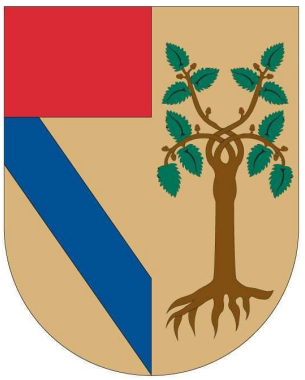


UNIVERSIDAD
PANAMERICANA

Alumno:

Sebastian Astiazarán Lopez - 0226403

Pablo Armando Uscanga Camacho - 0225717

Rolando Adrián Palacios Guzmán - 0223424

Trabajo:

Proyecto Final - Chats Grupales

Materia:

Cómputo Distribuido

Profesor:

Dr. Juan Carlos López Pimentel

INTRODUCCIÓN

REQUERIMIENTOS DEL PROYECTO

Repositorio de proyecto: https://github.com/Sastiazaran/chatRooms

Programas instalados:
- Software de visualización de codigo: Visual Studio (PREFERENCIAL)
- Software para creacion de archivos de Qt Designer(.ui)
- Docker Desktop

Librerias de Python necesarios: **PyQt5**

ORGANIZACION

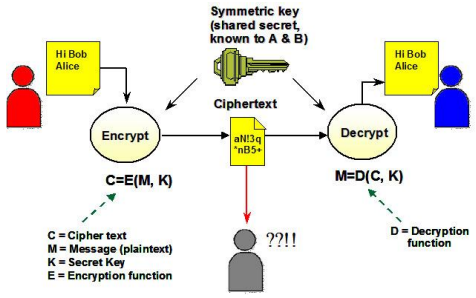
Desarrollo de Front End: Sebastián Astiazarán
Desarrolladores de Back End: Pablo Uscanga, Rolando Palacios

Aúnque cada uno de los integrantes tenía una tarea preestablecida, nuestro trabajo se centro en el trabajo en conjunto en la plataforma de Discord, en donde mientras uno estaba trabajando los demás daban feedback y ayudaban a resolver dudas en conjunto. De esa manera todos estabamos involucrados en el proyecto y conociamos como es que trabajaba el código en su totalidad.

Pensamos en que ibamos a tener mejor compresión cada uno de nosotros si todas las partes las haciamos en conjunto que separadas.

DESARROLLO

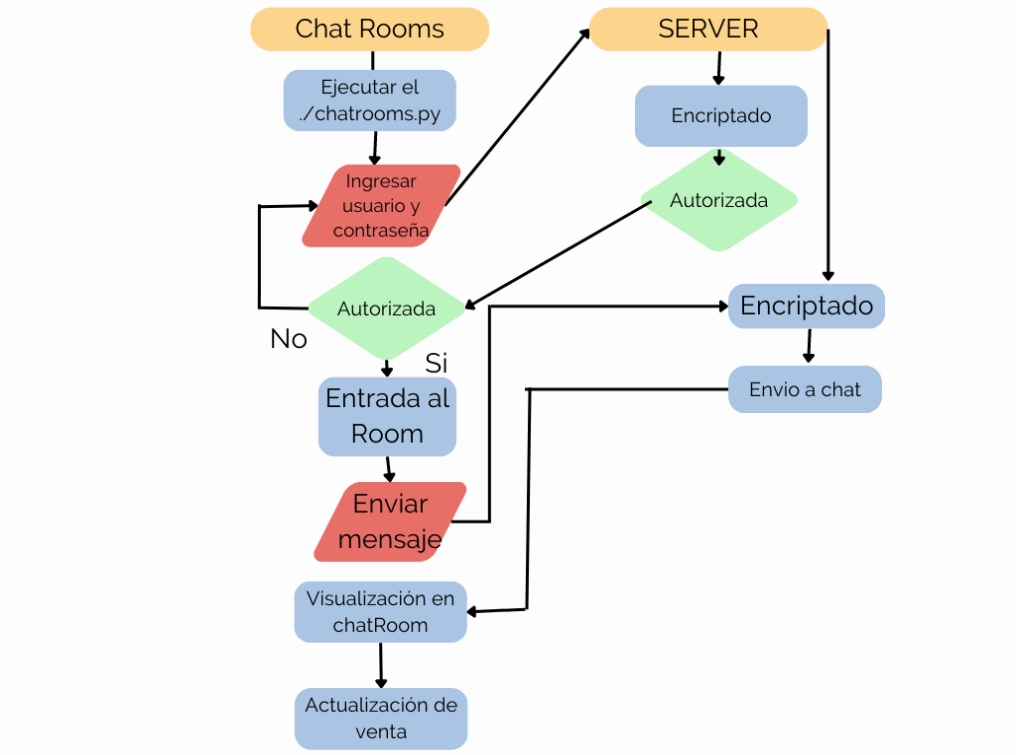
PROTOCOLO ALICE AND BOB



Nuestro enfoque para la encriptación de los mensajes es que tenemos nuestro código de encriptación dentro del servidor, que todo lo que recibe del cliente lo cifra y lo pasa en una verificación para desencriptar con una clave que nosotros especificamos como “k”.

Eso al momento de enviar las credenciales para iniciar en el servidor pasan por el cifrado para verificar las claves “usuario” y “contraseña” se encripta y desencriptan dentro del servidor para enviar respuesta a los usuarios y proceguir con la instrucción.

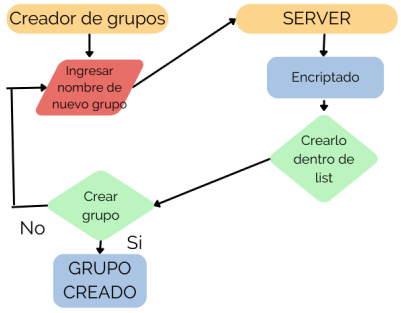
DIAGRAMA



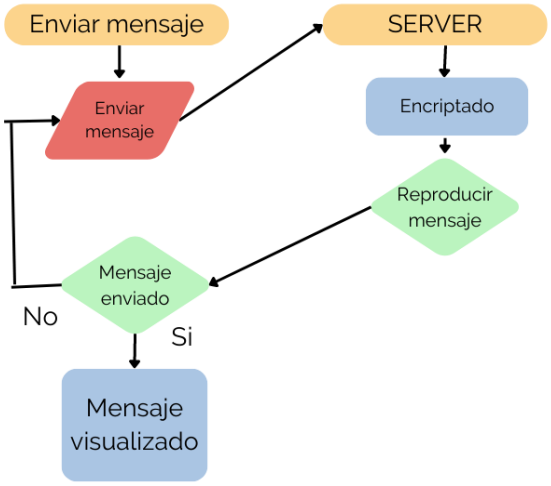
Lo que se muestra en el diagrama es la forma en la que hacemos todas las conexiones pasando primero por el cliente que es el que ejecuta el problema que se conecta al servidor y despues pasa por el servidor para hacer el Protocolo de encriptación que despues hace que las funciones conectadas se reproduzcan en la aplicación.

La primera siendo el ingreso hacia el servidor, despues la creación de chats de grupo con su respectivo nombre, despues con el envío de mensajes al grupo para que se puedan visualizar en el UI, y despues cambiarse de chat de grupo, ya sea en la pestañita que tenemos en la parte de abajo donde se visualizan todos los grupos.

PROTOCOLO DE CREAR GRUPO



PROTOCOLO DE ENVIAR MENSAJE



CONCLUSION

EVIDENCIA

```python
import sys
import random
import utilities
import socket
from PyQt5 import QtWidgets, uic
from PyQt5.QtWidgets import (
    QMainWindow,
    QApplication,
    QWidget,
    QLabel,
    QLineEdit,
    QPushButton,
    QVBoxLayout,
)

HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 5000  # The port used by the server

#lists

users = []
userChatrooms = []
```

```
allChatrooms = []
getAdminChat = []
labelList = []
buttonList = []
button1List = []
button2List = []
getChat = ""

app = QtWidgets.QApplication([])

loginWindow = uic.loadUi("Login.ui")
mainWindow = uic.loadUi("MainWindow.ui")
errorWindow = uic.loadUi("error.ui")
createChatRoom = uic.loadUi("createChatRoom.ui")
mainWindowAdmin = uic.loadUi("MainWindowAdmin.ui")
registerWindow = uic.loadUi("Register.ui")

username = ""

def getDataUsers():
    global users
    #   Get Users
    msgUser = "getUsers|"
    msgUser = utilities.cifrar(msgUser)
    s.send(msgUser.encode())

    userData = s.recv(1024)
    userData = utilities.cifrar(userData.decode())
    users = userData.split("|")

def getUserChatRoooms():
    global userChatrooms
    #   Get User ChatRooms
    msgUserChatrooms = "getUserChatrooms|"
    msgUserChatrooms = utilities.cifrar(msgUserChatrooms)
    s.send(msgUserChatrooms.encode())

    userChatroomsData = s.recv(1024)
    userChatroomsData = utilities.cifrar(userChatroomsData.decode())
    userChatrooms = userChatroomsData.split("|")

def getAllChatRooms():
```

```
    global allChatrooms
     #   Get All ChatRooms
    msgAllChatrooms = "getAllChatrooms|"
    msgAllChatrooms = utilities.cifrar(msgAllChatrooms)
    s.send(msgAllChatrooms.encode())

    allChatroomsData = s.recv(1024)
    allChatroomsData = utilities.cifrar(allChatroomsData.decode())
    allChatrooms = allChatroomsData.split("|")

def getAdminChats():
    global getAdminChat
    #   Get AdminChat
    msgGetAdminChat = "getAdminChat|"
    msgGetAdminChat = utilities.cifrar(msgGetAdminChat)
    s.send(msgGetAdminChat.encode())

    getAdminChatData = s.recv(1024)
    getAdminChatData = utilities.cifrar(getAdminChatData.decode())
    getAdminChat = getAdminChatData.split("|")

def gui_login():
    global username
    registerWindow.hide()
    username = loginWindow.lineEdit.text()
    password = loginWindow.lineEdit_2.text()

    print("username: " + username + " password: " + password)

    msg = "auth|"
    msg = msg + username + "|" + password + "\0"
    msg = utilities.cifrar(msg)

    print("Enviando: %s " % (msg))
    s.send(msg.encode())

    data = s.recv(1024)
    data = utilities.cifrar(data.decode())
    print(data)

    if len(username) == 0 or len(password) == 0:
        loginWindow.label_5.setText("Please enter data on all fields")
```

```
    elif "Denied" in data:
        gui_error()
    else:
        gui_entrar()

def gui_entrar():
    global username, getChat
    loginWindow.hide()
    mainWindow.show()

    mainWindow.label_Nickname.setText("Welcome " + username + " !")

    ##  Area 1 (Users)
    # Acceder al área de desplazamiento y al widget contenedor
    scroll_area = mainWindow.scrollArea
    scroll_content_widget = mainWindow.scrollAreaWidgetContents

    # Crear un layout vertical para el widget contenedor
    layout = QVBoxLayout(scroll_content_widget)

    # Ejemplo de agregar elementos dinámicamente
    getDataUsers()
    #getUserChatRoooms()
    getAllChatRooms()
    for i in users:
        label = QLabel(f"User {i}")
        layout.addWidget(label)

    ##  Area 2 (Current chatrooms)
    scroll_area_2 = mainWindow.scrollArea_2
    scroll_content_widget_2 = mainWindow.scrollAreaWidgetContents_2

    # Crear un layout vertical para el widget contenedor
    layout = QVBoxLayout(scroll_content_widget_2)

    # Ejemplo de agregar elementos dinámicamente

    for i in userChatrooms:
        label = QLabel(f"Chat Room {i}")
        button = QPushButton(f"Join {i}")
        layout.addWidget(label)
        layout.addWidget(button)
```

```
    ##  Area 3 (Current chatrooms)
    scroll_area_3 = mainWindow.scrollArea_3
    scroll_content_widget_3 = mainWindow.scrollAreaWidgetContents_3

    # Crear un layout vertical para el widget contenedor
    layout = QVBoxLayout(scroll_content_widget_3)

    # Ejemplo de agregar elementos dinámicamente
    for i in allChatrooms:
        label = QLabel(f"Chat Room {i}")
        button = QPushButton(f"Join to {i}")
        layout.addWidget(label)
        layout.addWidget(button)

    ##

def write_to_msgBrowser():
    global getChat
    msggetChat = "getChat|"
    msggetChat = utilities.cifrar(msggetChat)
    s.send(msggetChat.encode())

    getChats = s.recv(1024)
    getChats = utilities.cifrar(getChats.decode())
    getChat = getChats.split("|")
    print(getChat)
    mainWindow.msgEdit.append(getChat)
    #gui_entrar()

def send_message():
    global username, getChat
    msgSend = mainWindow.lineEdit.text()
    msgSend = "messageSent|"  + username + '->' + msgSend + '|' + getChat
    msgSend = utilities.cifrar(msgSend)
    print("Enviando: %s" + msgSend)
    s.send(msgSend.encode())

    getMsgSent = s.recv(1024)
    getMsgSent = utilities.cifrar(getMsgSent.decode())
    getChat = getMsgSent.split("|")
    mainWindow.label_2.setText(getChat)
```

```
    # write_to_msgBrowser()
    mainWindow.hide()
    #gui_entrar()

def adminView():
       global username, users, userChatrooms, allChatrooms, label, button,
button1, button2, getAdminChat
    mainWindow.hide()
    mainWindowAdmin.show()

    mainWindowAdmin.label_Nickname.setText("Welcome admin " + username + " !")

    ##  Area 1 (Users)
    # Acceder al área de desplazamiento y al widget contenedor
    scroll_area = mainWindowAdmin.scrollArea
    scroll_content_widget = mainWindowAdmin.scrollAreaWidgetContents

    # Crear un layout vertical para el widget contenedor
    layout = QVBoxLayout(scroll_content_widget)
    getDataUsers()
    getUserChatRoooms()
    getAllChatRooms()
    getAdminChats()
    # Ejemplo de agregar elementos dinámicamente
    for i in users:
        label = QLabel(f"User {i}")
        layout.addWidget(label)


    ##  Area 2 (Current chatrooms)
    scroll_area_2 = mainWindowAdmin.scrollArea_2
    scroll_content_widget_2 = mainWindowAdmin.scrollAreaWidgetContents_2

    # Crear un layout vertical para el widget contenedor
    layout = QVBoxLayout(scroll_content_widget_2)


    # Ejemplo de agregar elementos dinámicamente
    for i in userChatrooms:
        label = QLabel(f"Chat Room {i}")
        button = QPushButton(f"Join {i}")
```

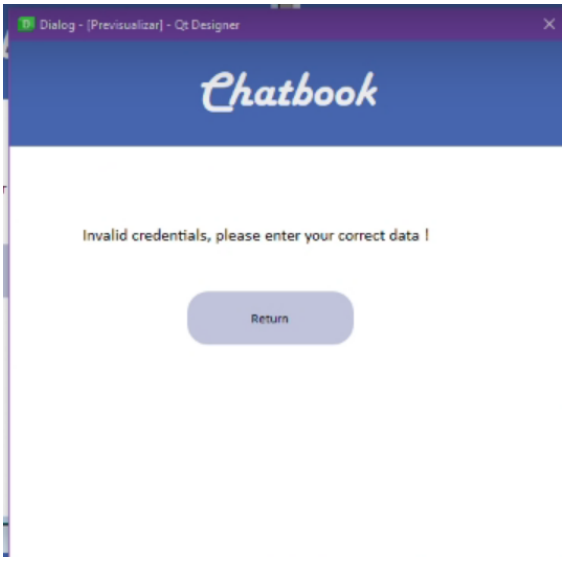
```
        layout.addWidget(label)
        layout.addWidget(button)


    ##  Area 3 (Current chatrooms)
    scroll_area_3 = mainWindowAdmin.scrollArea_3
    scroll_content_widget_3 = mainWindowAdmin.scrollAreaWidgetContents_3

    # Crear un layout vertical para el widget contenedor
    layout = QVBoxLayout(scroll_content_widget_3)

    # Ejemplo de agregar elementos dinámicamente
    for i in allChatrooms:
        label = QLabel(f"Chat Room {i}")
        button = QPushButton(f"Join to {i}")

        layout.addWidget(label)
        layout.addWidget(button)

    ##

    ##  Area 3 (Current chatrooms)
    scroll_area_4 = mainWindowAdmin.scrollArea_4
    scroll_content_widget_4 = mainWindowAdmin.scrollAreaWidgetContents_4

    # Crear un layout vertical para el widget contenedor
    layout = QVBoxLayout(scroll_content_widget_4)

    # Ejemplo de agregar elementos dinámicamente
    for i in getAdminChat:
        labeladmin = QLabel(f"User {i}")
        add = QPushButton(f"add {i}")
        kick = QPushButton(f"delete {i}")
        layout.addWidget(labeladmin)
        layout.addWidget(add)
        layout.addWidget(kick)

    #   add

    msgAdd = "addUser|" + username + '|' + labeladmin
    msgAdd = utilities.cifrar(msgAdd)
    print("Add %s " % (msgAdd))
```

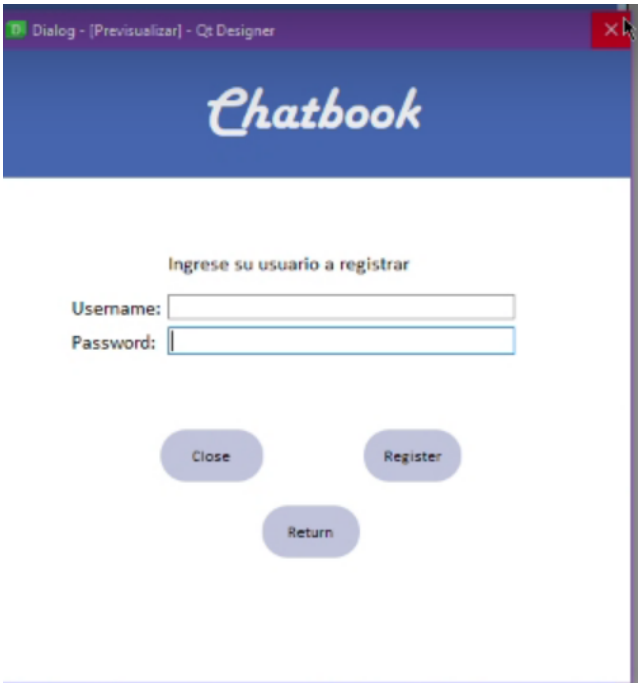
```
    s.send(msgAdd.encode())

    #   delete

    msgDelete = "deleteUser|" + username + '|' + labeladmin
    msgDelete = utilities.cifrar(msgDelete)
    print("Add %s " % (msgDelete))
    s.send(msgAdd.encode())

def gui_error():
    loginWindow.hide()
    errorWindow.show()

def return_to_login():
    mainWindow.hide()
    mainWindowAdmin.hide()
    loginWindow.show()

def return_to_login_from_error():
    errorWindow.hide()
    loginWindow.show()

def createChatRoom_to_mainWindow():
    createChatRoom.hide()
    mainWindowAdmin.hide()
    mainWindow.show()

def mainWindow_to_createChatRoom():
    mainWindow.hide()
    mainWindowAdmin.hide()
    createChatRoom.show()

def CreateChatRoom():
    global username
    chatRoomName = createChatRoom.lineEdit.text()

    msgChatRoomName = "createChatRoom|"
    msgChatRoomName =  msgChatRoomName + username + '|' + chatRoomName + '\0'

    msgChatRoomName = utilities.cifrar(msgChatRoomName)
    print("Enviando: %s " % (msgChatRoomName))
    s.send(msgChatRoomName.encode())
```

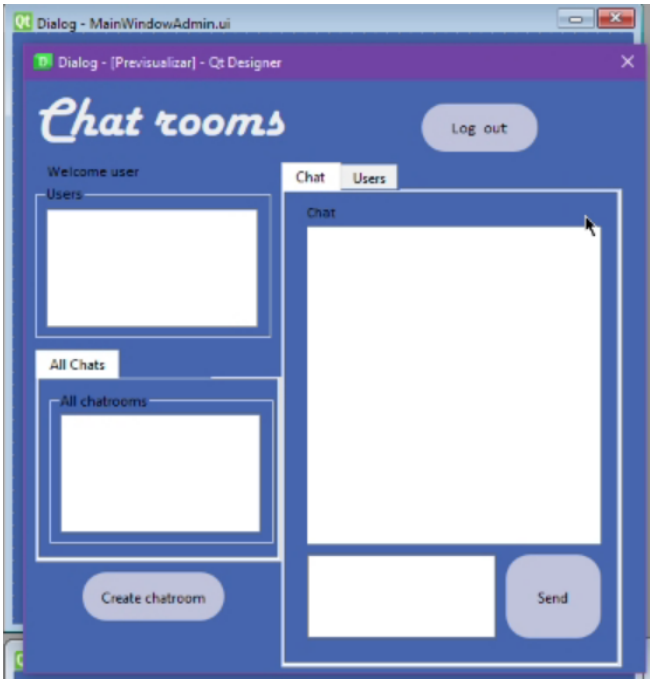
```
    if len(chatRoomName) == 0:
        createChatRoom.creationMessage_2.setText("Please enter data on field")
    else:
        createChatRoom.creationMessage.setText("Chat room " + chatRoomName + "
created successfuly!!")

def close():
    app.exit()

def register_user():
    global username
    loginWindow.hide()
    registerWindow.show()
    username = loginWindow.lineEdit.text()
    password = loginWindow.lineEdit_2.text()

    print("username: " + username + " password: " + password)

    msg = "registrarUsuario|"
    msg = msg + username + "|" + password + "\0"
    msg = utilities.cifrar(msg)

    print("Enviando: %s " % (msg))
    s.send(msg.encode())

    data = s.recv(1024)
    data = utilities.cifrar(data.decode())
    print(data)

    if len(username) == 0 or len(password) == 0:
        loginWindow.label_5.setText("Please enter data on all fields")
    else:
        gui_login()

def register_to_login():
    registerWindow.hide()
    loginWindow.show()
#   Funcionalidad botones

loginWindow.pushButton.clicked.connect(gui_login)
loginWindow.pushButton_2.clicked.connect(close)
```

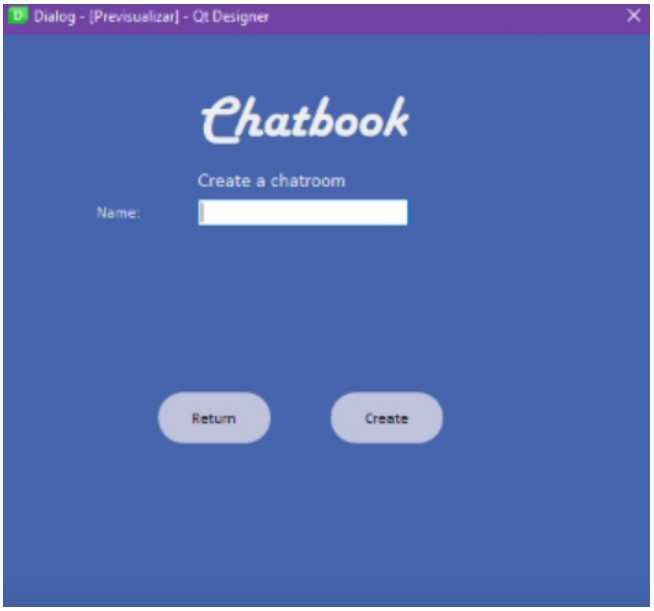
```
loginWindow.pushButton_3.clicked.connect(register_user)
mainWindow.button_createChatRoom.clicked.connect(mainWindow_to_createChatRoom)
mainWindow.button_logOut.clicked.connect(return_to_login)
mainWindow.button_send.clicked.connect(send_message)
errorWindow.pushButton.clicked.connect(return_to_login_from_error)
createChatRoom.button_returnButton.clicked.connect(createChatRoom_to_mainWindo
w)
createChatRoom.button_createButton.clicked.connect(CreateChatRoom)
mainWindow.button_Admin.clicked.connect(adminView)
mainWindowAdmin.button_createChatRoom.clicked.connect(mainWindow_to_createChat
Room)
mainWindowAdmin.button_logOut.clicked.connect(return_to_login)
mainWindow.button_send.clicked.connect(send_message)
registerWindow.pushButton_3.clicked.connect(register_user)
registerWindow.pushButton.clicked.connect(register_to_login)
registerWindow.pushButton_2.clicked.connect(close)



with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    loginWindow.show()
    app.exec()
```

IMAGENES DE UI









DIFICULTADES

Lo que se dificulto pero logró tener solución es la forma de cambiar de grupo, ya que cuando se envía la solucitud lo que regresaba era el apartado sin encriptar pero no regresaba el cambio en la lista del grupo, pero ahora lo que hace es que hicimos una función aparte que solventaba el problema de listado y ya se tenía la libertad de entrar y salir.

Otra dificultad fue el desarrollo de la UI, ya que se estaba teniendo una constante actualización para tener un desarrollo amigable a la vista y dejará de ser el estilo rústico de los primeros momentos.

APRENDIZAJES

Nuestra conclusión final es que logramos aplicar el protocolo de Alice and Bob para que todas las funciones dentro del servidor fueran capaces de poder desencriptar las solicitudes mandadas por el cliente y que de esa manera se tuviera un poco de “control” y principios de seguridad en servidores.

También se reaprendio la complementación de la interfaz gráfica de UI que habíamos visto previamente, pero que nos fue de ayuda para darle más color y más vida a nuestras interfaces y el cómo de estas para tener un tono amigable pero compatibles a la funcionalidad de este.