



# Benemérita Universidad Autónoma de Puebla

Materia: *Sistemas Operativos II*

Profesor: *Gilberto López Poblano*

Alumno: *Christian Amauri Amador Ortega - 201927821*

**- Archivos Json -**

*Para esta serie de prácticas con código utilizamos el lenguaje C# en el IDE visual studio (la versión más reciente en septiembre de 2022). Puede haber variaciones en el funcionamiento de exactamente los mismos códigos pero con otra versión de visual studio u otro IDE. De todas formas estas variaciones deben de ser mínimas e insignificantes.*

*!!! Documentaremos el funcionamiento de las aplicaciones (en su ejecución) y haremos algunos comentarios / notas / consideraciones / advertencias al respecto. Para ahorrar tiempo y esfuerzo, trataremos de no ahondar demasiado en la estructura del código, sino simplemente explicar brevemente cómo funciona **(se asume que el lector tiene ya conocimientos suficientes sobre conceptos varios de programación estructurada, ambiente gráfico, variables, datos, y demás...)***

*Finalmente hay que tomar en cuenta que C# y Visual studio son herramientas pesadas y caprichosas. Y que el equipo que vayamos a utilizar debe tener cierta capacidad de rendimiento superior a sólo básica.*

Primero, el objetivo del proyecto es crear un programa en C# con el que podamos manejar archivos Json. Para eso hemos creado **dos** proyectos. Uno es el funcionamiento básico de un Jsonweb (sólo lee datos) , el otro es un poco más elaborado, es un pequeño chat, (lee y escribe) pero sigue siendo básico

En cualquier caso, el repositorio en el que estos códigos se encuentran (y otros códigos en C# del curso de Sistemas operativos 2, BUAP FCC, Otoño 2022) está disponible haciendo click en el siguiente enlace:

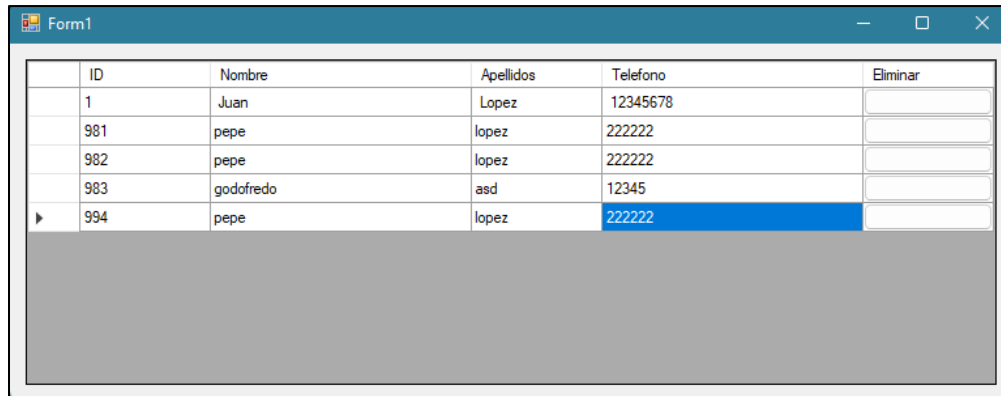
<https://drive.google.com/drive/folders/1yZujMI51XAnEZBMj-4ykUWMfHXBC9uYp?usp=sharing>

Y estará disponible al menos durante el resto de 2022. Gracias. 😊

## **Primer programa:**

### **Ejemplo rápido del funcionamiento general del proyecto, para resumir:**

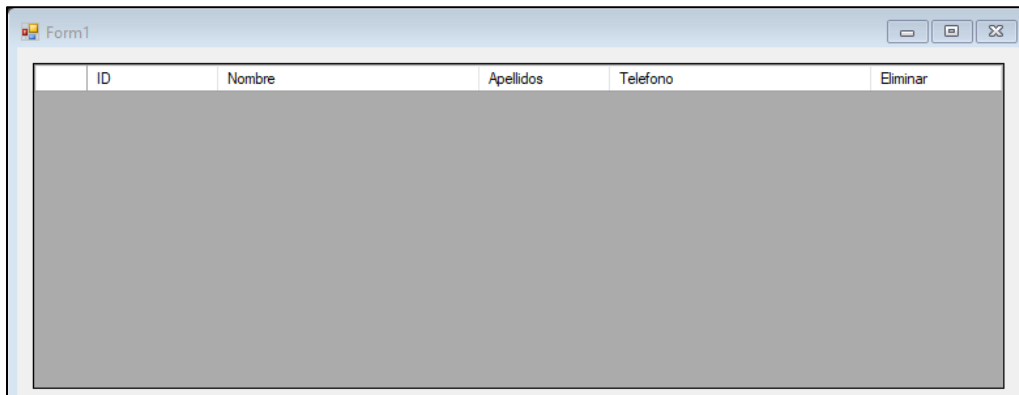
- Podemos leer un archivo Json con un formato definido (que conocemos)



	ID	Nombre	Apellidos	Telefono	Eliminar
	1	Juan	Lopez	12345678	
	981	pepe	lopez	222222	
	982	pepe	lopez	222222	
	983	godofredo	asd	12345	
▶	994	pepe	lopez	222222	

Ahora, los archivos .cs del proyecto (que nosotros creamos) corresponden a un diseño y un script. Comencemos por el diseño...

## ***- Diseño -***



**Form1** (Form main)

**Contiene el dataGridView en el que recibiremos los datos tipo Json cuyo formato conocemos.**  
(Estos datos son: ID, Nombre, Apellidos y teléfono)

## - Script -

### Form1.cs

La estructura inicial de este script consta de:

- 11 librerías
- namespace definido como: webJsonOnline
- public partial class Form1 : Form

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Net;
8  using System.Text;
9  using System.Threading.Tasks;
10 using System.Windows.Forms;
11 using Newtonsoft.Json;
12
13 namespace webJsonOnline
14 {
15     public partial class Form1 : Form
16     {
```

Dentro de namespace webJsonOnline tenemos las clases **Dato** y **Root**:

La clase **Dato** define la estructura del archivo Json que vamos a leer de la url (en este caso leeremos: ID, Nombre, Apellido, teléfono y clave):

```
48     1 referencia
49     public class Dato
50     {
51         1 referencia
52         public string id { get; set; }
53         1 referencia
54         public string nom { get; set; }
55         1 referencia
56         public string app { get; set; }
57         1 referencia
58         public string tel { get; set; }
59         1 referencia
60         public string clave { get; set; }
61     }
```

La clase **Root** la definimos como una clase cuyo único atributo es una lista de objetos tipo **Dato**:

```
57 2 referencias
58 public class Root
59 {
60     6 referencias
    public List<Dato> dato { get; set; }
61 }
```

Ahora, dentro de **Form1 : Form**, se encuentran los siguientes componentes:

```
15 3 referencias
16 public partial class Form1 : Form
17 {
21     1 referencia
    public Form1() ...
22     1 referencia
    private void Form1_Load(object sender, EventArgs e) ...
26
27     1 referencia
    public void load() ...
46 }
```

- un constructor
- 2 métodos

La función **Form1\_Load()** simplemente manda a llamar a la función **Load()**....

```
22 private void Form1_Load(object sender, EventArgs e)
23 {
24     load();
25 }
```

La función `load()` limpia la tabla de la ventana (en caso de que haya habido consultas previas), luego consulta la url indicada donde vamos a sacar los datos, la url completa es:

<https://serviciosdigitalesplus.com/servicio/servicio.php?tipo=1&clave=2304>

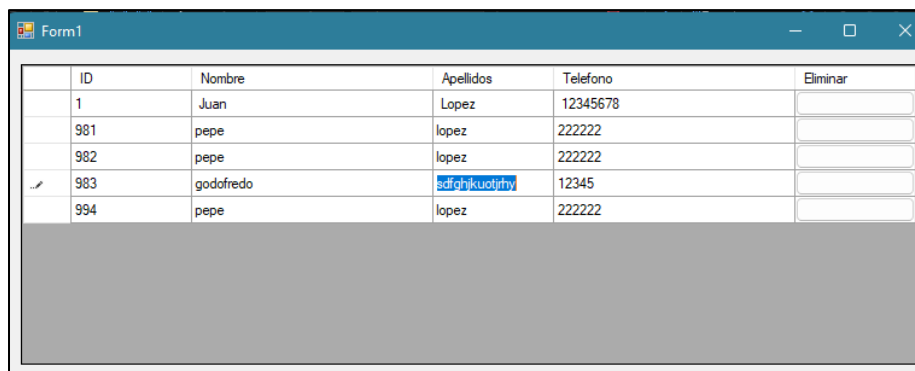
luego guarda en un string los datos de la url. Posteriormente crea el objeto `r` de clase `root` para separar individualmente los datos obtenidos.

Finalmente deposita los datos de `r` en la tabla de la ventana `main` mediante un `for`.

```
1 referencia
27 public void load()
28 {
29     dataGridView1.Rows.Clear();
30     String url = "https://serviciosdigitalesplus.com/servicio/servicio.php?";
31     String texto = (new WebClient()).DownloadString(url);
32     Root r = JsonConvert.DeserializeObject<Root>(texto);
33
34     for (int i = 0; i < r.dato.Count ; i++)
35     {
36         int j = this.dataGridView1.Rows.Add();
37         dataGridView1.Rows[j].Cells[0].Value = r.dato[i].id.ToString();
38         dataGridView1.Rows[j].Cells[1].Value = r.dato[i].nom.ToString();
39         dataGridView1.Rows[j].Cells[2].Value = r.dato[i].app.ToString();
40         dataGridView1.Rows[j].Cells[3].Value = r.dato[i].tel.ToString();
41         dataGridView1.Rows[j].Cells[4].Value = r.dato[i].clave.ToString();
42     }
43
44 }
45
```

De hecho ese es todo el programa, una vez ejecutado, no tiene más funcionalidad que simplemente leer una vez los datos de la url indicada, ni siquiera tenemos que presionar ningún botón o tecla para que haga esto. Aquí termina.

**Nota:** el programa te permite modificar la tabla leída, pero esto no modifica el Json original. Simplemente dejemos pasar este detalle despreciable...

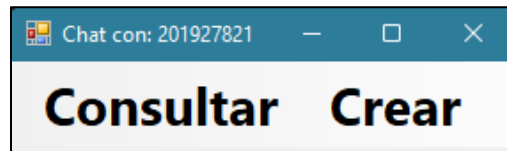


	ID	Nombre	Apellidos	Telefono	Eliminar
	1	Juan	Lopez	12345678	<input type="button" value="Eliminar"/>
	981	pepe	lopez	222222	<input type="button" value="Eliminar"/>
	982	pepe	lopez	222222	<input type="button" value="Eliminar"/>
	983	godofredo	sdjghjkuotjthv	12345	<input type="button" value="Eliminar"/>
	994	pepe	lopez	222222	<input type="button" value="Eliminar"/>

## Segundo programa:

### Ejemplo rápido del funcionamiento general del proyecto, para resumir:

- Podemos escoger entre consultar o crear mensajes:



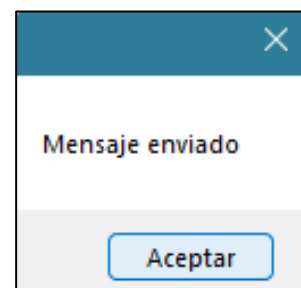
Consultar:

A screenshot of the 'Consultar' window. It displays a table with the following data:

	Ciente	Fecha	Mensaje
▶	Amauri	2022-10-24 22:08:58	Realmente espero que esto funcione
	Amauri_otra_vez	2022-10-24 22:10:10	Si_funciono
	xdxdxd	2022-10-24 22:25:08	swswswswswsws
	[REDACTED]	2022-10-24 22:56:26	[REDACTED]
	babe_you_re_too_controlling	2022-10-26 22:16:49	Imma feed you to the wolves

Crear:

A screenshot of the 'Crear' window. It has two input fields: 'Nombre del cliente:' with the value 'Amauri3000' and 'Mensaje a entregar:' with the value 'DocumentandoAUltimaHoraEh'. There is a blue button labeled 'Enviar'.

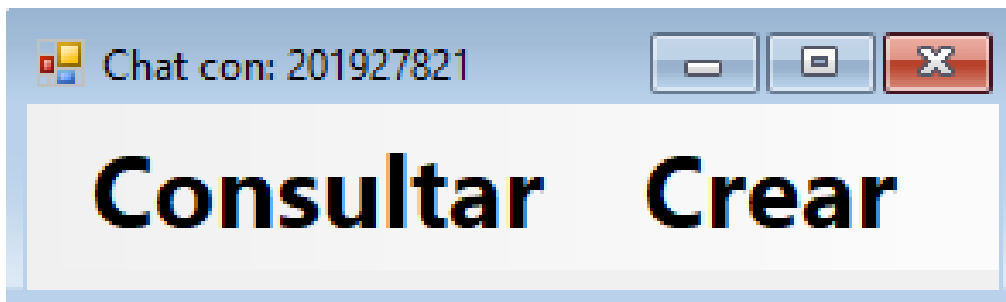


A screenshot of the 'Consultar' window after creating a new message. The table now includes the new entry:

	Ciente	Fecha	Mensaje
	Amauri	2022-10-24 22:08:58	Realmente espero que esto funcione
	Amauri_otra_vez	2022-10-24 22:10:10	Si_funciono
	xdxdxd	2022-10-24 22:25:08	swswswswswsws
	[REDACTED]	2022-10-24 22:56:26	[REDACTED]
	babe_you_re_too_controlling	2022-10-26 22:16:49	Imma feed you to the wolves
▶	Amauri3000	2022-11-21 23:27:53	DocumentandoAUltimaHoraEh

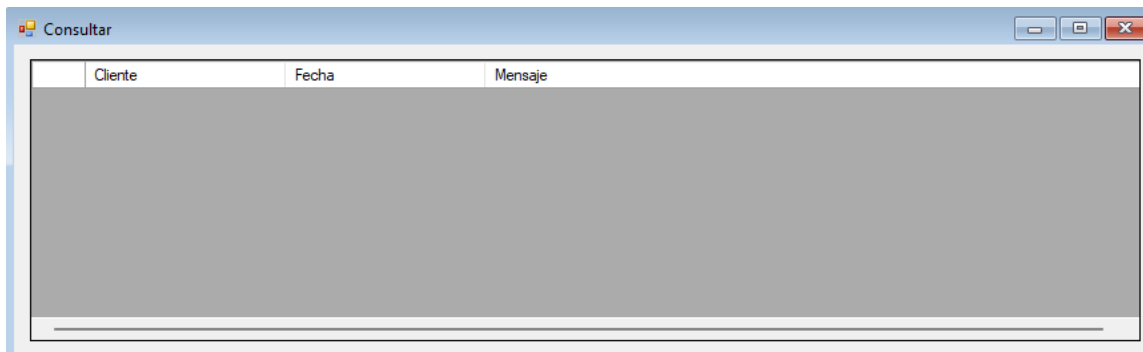
Ahora, los archivos .cs del proyecto (que nosotros creamos) corresponden a 3 diseños y 3 scripts. Comencemos por los diseños...

## *- Diseños -*



#1 **Form1** (ventana main)

Contiene el menú strip con las dos opciones de las que disponemos (Consultar y crear) Su tamaño es constante.



#2 **Consultar**

Contiene el dataGridView en el que depositaremos los datos de la consulta a la url. No podemos modificar los valores de sus celdas, sólo están para leer. No tiene funcionalidad aparte de eso. Su tamaño es variable. Cuenta con scrollBars.





#3 **Crear** (ventana main)

**Contiene lo necesario para enviar un mensaje a nuestra url** Con el formato Json adecuado (definido por nosotros) Si ambas cajas de texto (nombre y mensaje) no están llenas; el botón de enviar no tiene funcionalidad. Este se activa sólo si hay valores válidos en ambos campos (hay algunos caracteres especiales que no son admitidos). Su tamaño es constante.

## *- Scripts -*

### #1 **Form1.cs**

La estructura inicial de este script consta de:

- 11 librerías
- namespace definido como: `_08_Chat_Tarea_`
- `public partial class Form1 : Form`

```
1  using System;
2      using System.Collections.Generic;
3      using System.ComponentModel;
4      using System.Data;
5      using System.Drawing;
6      using System.Linq;
7      using System.Text;
8      using System.Threading.Tasks;
9      using System.Windows.Forms;
10     using Newtonsoft.Json;
11     using System.Net;
12
13     namespace _08_Chat_Tarea_
14     {
15         3 referencias
16         public partial class Form1 : Form
17         {
```

Dentro de `namespace _08_Chat_Tarea_` tenemos las clases `Chat` y `Root`:

La clase `Chat` define los componentes de los que va a constar un mensaje (con formato Json).

```
40
41 1 referencia
42  public class Chat
43  {
44      1 referencia
45      public string cliente { get; set; }
46      0 referencias
47      public string usuario { get; set; }
48      1 referencia
49      public string descripcion { get; set; }
50      1 referencia
51      public string fecha { get; set; }
52  }
```

La clase `Root` Se define como una clase cuyo único atributo es una lista de `Chats`.

```
50 2 referencias
51  public class Root
52  {
53      4 referencias
54      public List<Chat> chat { get; set; }
55  }
```

Ahora, dentro de `Form1 : Form.` se encuentran los siguientes componentes:

```
15 3 referencias
16  public partial class Form1 : Form
17  {
18      1 referencia
19      public Form1() ...
20
21      1 referencia
22      private void Form1_Load(object sender, EventArgs e) ...
23
24      1 referencia
25      private void consultarToolStripMenuItem_Click(object sender, EventArgs e) ...
26
27      1 referencia
28      private void crearToolStripMenuItem_Click(object sender, EventArgs e) ...
29  }
```

- un constructor
- 3 métodos

El método `Form1_Load()` está vacío...

```
23 1 referencia
    private void Form1_Load(object sender, EventArgs e)
24  {
25
26  }
```

El método `consultarToolStripMenuItem_Click()` crea un objeto de tipo `Consultar`, y lo muestra:

```
28 1 referencia
    private void consultarToolStripMenuItem_Click(object sender, EventArgs e)
29  {
30      Consultar C = new Consultar();
31      C.Show();
32  }
```

El método `crearToolStripMenuItem_Click()` crea un objeto de tipo `Crear` y lo muestra:

```
34 1 referencia
    private void crearToolStripMenuItem_Click(object sender, EventArgs e)
35  {
36      Crear C = new Crear();
37      C.Show();
38  }
```

## #2 *Form1.cs*

La estructura inicial de este script consta de:

- 11 librerías
- namespace definido como: `_08_Chat_Tarea_`
- `public partial class Consultar : Form`

```
1  using Newtonsoft.Json;
2  using System;
3  using System.Collections.Generic;
4  using System.ComponentModel;
5  using System.Data;
6  using System.Drawing;
7  using System.Linq;
8  using System.Net;
9  using System.Text;
10 using System.Threading.Tasks;
11 using System.Windows.Forms;
12
13 namespace _08_Chat_Tarea_
14 {
15     4 referencias
16     public partial class Consultar : Form
17     {
```

Ahora, dentro de Consultar : Form. se encuentran los siguientes componentes:

```
15 4 referencias
16 public partial class Consultar : Form
17 {
18     1 referencia
19     public Consultar() ...
20
21     0 referencias
22     private void textBox1_TextChanged(object sender, EventArgs e) ...
23
24     1 referencia
25     private void Consultar_Load(object sender, EventArgs e) ...
26
27 }
28
```

- un constructor
- 2 métodos

El método `textBox1_TextChanged()` está vacío...

```
22 0 referencias
23 private void textBox1_TextChanged(object sender, EventArgs e)
24 {
25 }
26
```

El método `Consultar_Load()` sigue exactamente el mismo método de lectura de archivos Json por url que el método `Form1.cs > namespace webJsonOnline > load()` del código del primer proyecto. Pero en este caso, la url y los datos, son diferentes.

Url: <http://serviciosdigitalesplus.com/chat/?tipo=2&usuario=201927821>

(201927821) corresponde a la matrícula universitaria del autor del código.

```
27 1 referencia
28 private void Consultar_Load(object sender, EventArgs e)
29 {
30     String url = "http://serviciosdigitalesplus.com/chat/?tipo=2&usuario=201927821";
31     String texto = (new WebClient().DownloadString(url));
32     Root r = JsonConvert.DeserializeObject<Root>(texto);
33     for (int i = 0; i < r.chat.Count; i++)
34     {
35         this.dataGridView1.Rows.Add();
36         this.dataGridView1.Rows[i].Cells[0].Value = r.chat[i].cliente;
37         this.dataGridView1.Rows[i].Cells[1].Value = r.chat[i].fecha;
38         this.dataGridView1.Rows[i].Cells[2].Value = r.chat[i].descripcion;
39     }
40 }
41
```

### #3 Crear.cs

La estructura inicial de este script consta de:

- 10 librerías
- namespace definido como: `_08_Chat_Tarea_`
- `public partial class Crear : Form`

```
1  using System;
2      using System.Collections.Generic;
3      using System.ComponentModel;
4      using System.Data;
5      using System.Drawing;
6      using System.Linq;
7      using System.Net;
8      using System.Text;
9      using System.Threading.Tasks;
10     using System.Windows.Forms;
11
12     namespace _08_Chat_Tarea_
13     {
14         4 referencias
15         public partial class Crear : Form
16         {
```

Ahora, dentro de `Crear : Form`, se encuentran los siguientes componentes:

```
14     4 referencias
15     public partial class Crear : Form
16     {
17         1 referencia
18         public Crear() ...
19
20
21         1 referencia
22         private void Crear_Load(object sender, EventArgs e) ...
23
24
25         1 referencia
26         private void button1_Click(object sender, EventArgs e) ...
27
28     }
```

- un constructor
- 2 métodos

El método `Crear_Load()` está vacío...

```
21 1 referencia
22 private void Crear_Load(object sender, EventArgs e)
23 {
24 }
25
```

El método `button1_Click()` Corresponde a la acción ejecutada al presionar el botón “Enviar” en el diseño “Crear”. Primero verifica que ambas cajas de texto ( nombre del cliente y mensaje ) estén llenas con algo (caso contrario se te notifica). Si ambas cajas están llenas, definimos la `String url` como el link:

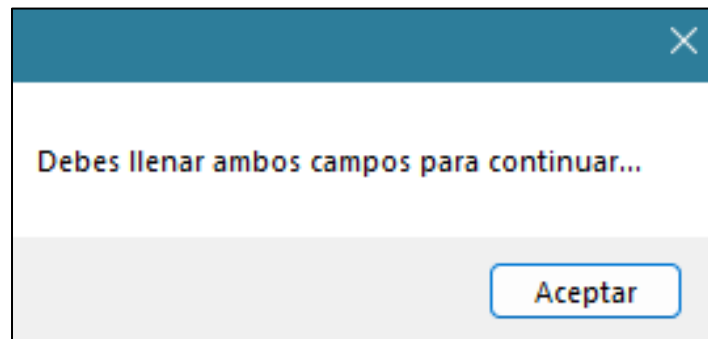
<http://serviciosdigitalesplus.com/chat/?tipo=1&cliente=> + cliente( nombre ingresado en la caja de texto) + [&descripcion=](#) + descripción(El mensaje que se quiere enviar) + [&usuario=](#) + usuario(La matrícula universitaria del autor. En este caso, definida en la línea 36 como: 201927821);

Inmediatamente después de eso, mandamos la string “url” al url indicado mediante la línea 44: `String texto = (new WebClient()).DownloadString(url)`; Finalmente si todo salió bien, se muestra al usuario de nuestro programa el mensaje “Mensaje enviado” mediante un `MessageBox`:

Queda entonces de la siguiente manera:

```
26 1 referencia
27 private void button1_Click(object sender, EventArgs e)
28 {
29     if (this.textBox1.Text == "" || this.textBox2.Text == "")
30     {
31         MessageBox.Show("Debes llenar ambos campos para continuar...");
32         return;
33     }
34     else
35     {
36         String cliente = "";
37         String usuario = "201927821"; // CAMBIAR DE MATRICULAAAAA
38         String descripcion = "";
39
40         cliente = this.textBox1.Text;
41         descripcion = this.textBox2.Text;
42
43         String url = "http://serviciosdigitalesplus.com/chat/?tipo=1&cliente="
44             + cliente + "&descripcion=" + descripcion + "&usuario=" + usuario;
45         String texto = (new WebClient()).DownloadString(url);
46
47         MessageBox.Show("Mensaje enviado");
48     }
49 }
```

Ejemplo del mensaje que se muestra en caso de que uno de los dos campos a llenar se encuentre vacío:



### **!!!! Nota final:**

Dada la estructura tan sencilla y fácil de este proyecto, y dado que estos mismos pasos fueron implementados por muchos estudiantes; Resulta muy fácil entre nosotros irrumpir en el chat de otro. Simplemente hay que cambiar nuestra matrícula por la de otro compañero en las líneas correspondientes del código (son 2 o 3). Eso es un fallo de seguridad fatal. Pero como este es un proyecto pequeño y didáctico... lo despreciamos.