

# Baco: *streaming* de vídeo y VoIP sobre TCP

Carlos Clement Bellido  
carlos.clement@iesdoctorbalmis.com

Alicante, 26 de mayo de 2020

Está página ha sido dejada en blanco intencionadamente.

# Índice general

<b>Acrónimos</b>	<b>7</b>
<b>1. Introducción</b>	<b>9</b>
1.1. Objetivo . . . . .	9
1.2. Justificación . . . . .	10
1.3. Análisis de lo existente . . . . .	10
<b>2. Necesidades empresariales para el desarrollo</b>	<b>13</b>
2.1. Recursos humanos . . . . .	13
2.1.1. Trabajadores . . . . .	14
2.1.2. Contratos . . . . .	14
2.1.3. Seguridad Social . . . . .	16
2.2. Prevención de riesgos laborales . . . . .	16
2.2.1. Riesgos laborales y medidas preventivas . . . . .	17
2.3. Iniciativa emprendedora . . . . .	20
2.3.1. Forma jurídica . . . . .	20
2.3.2. Tramites administrativos . . . . .	20
2.3.2.1. Constitución . . . . .	20
2.3.2.2. Puesta en marcha . . . . .	21

## *ÍNDICE GENERAL*

<b>3. Requisitos funcionales de la aplicación</b>	<b>23</b>
<b>4. Análisis y Diseño</b>	<b>25</b>
4.1. Diagrama de la arquitectura . . . . .	25
4.2. Diagrama de casos de uso . . . . .	28
4.3. Diagrama de clases . . . . .	30
4.3.1. Cliente . . . . .	30
4.3.2. Servidor . . . . .	34
4.4. Diseño de datos . . . . .	34
4.4.1. Amistades . . . . .	37
<b>5. Codificación</b>	<b>39</b>
5.1. Lenguajes de programación y marcado . . . . .	39
5.1.1. C# . . . . .	39
5.1.2. XAML . . . . .	39
5.1.3. Transact-SQL (T-SQL) . . . . .	40
5.2. Entornos de programación (IDE) . . . . .	40
5.2.1. Visual Studio . . . . .	40
5.2.1.1. NuGet . . . . .	41
5.2.2. SQL Server Management Studio (SSMS) . . . . .	42
5.3. Herramientas . . . . .	42
5.3.1. Mozilla Firefox . . . . .	42
5.3.2. Postman . . . . .	42
5.4. Aspectos relevantes de la implementación . . . . .	43
5.4.1. API . . . . .	43
5.4.1.1. ApiBaco . . . . .	43

5.4.1.2. Service . . . . .	43
5.4.2. Cliente – Baco . . . . .	43
5.4.2.1. ScreenRecorder.cs . . . . .	44
5.4.2.2. VoiceRecorder.cs . . . . .	50
5.4.3. Servidor – BacoServer . . . . .	52
5.4.4. Base de datos . . . . .	52
<b>6. Manuales de usuario</b>	<b>59</b>
6.1. Baco . . . . .	59
6.2. BacoServer . . . . .	77
<b>7. Ficheros entregados</b>	<b>89</b>
<b>8. Instalación y requisitos</b>	<b>91</b>
8.1. Instalación . . . . .	91
8.1.1. Intalación de Baco . . . . .	91
8.1.2. Intalación del servidor de Baco . . . . .	92
8.1.3. Árbol de directorios de la carpeta «Baco SL» . . . . .	92
8.1.4. Puesta en marcha del API REST . . . . .	92
8.2. Requisitos . . . . .	93
<b>9. Conclusiones</b>	<b>95</b>
9.1. Conclusiones sobre el trabajo realizado . . . . .	95
9.2. Posibles ampliaciones y mejoras . . . . .	95
<b>Lista de Códigos</b>	<b>97</b>
<b>Bibliografía</b>	<b>99</b>

## *ÍNDICE GENERAL*

Esta página ha sido dejada en blanco intencionadamente.

# Acrónimos

**CLI** Del inglés, *command-line interface*, es una interfaz basada únicamente en líneas de texto simple..

**FPS** Del inglés, *Frames Per Second*, indica la frecuencia de muestreo de imágenes por cada segundo..

**GUI** Del inglés, *graphical user interface*, es una interfaz basada imágenes y objetos gráficos para expresar información y acciones al usuario..

**LINQ** Del inglés, *Language Integrated Query*, permite, a los lenguajes .NET, capacidades de consulta de datos de forma nativa. Pertenece a la compañía Microsoft..

**VoIP** Del inglés, *Voice Over Internet Protocol*, es un conjunto de recursos que permiten el envío de señales de audio por Internet empleando el protocolo IP.

## *ACRÓNIMOS*

Esta página ha sido dejada en blanco intencionadamente.

# Capítulo 1

## Introducción

Baco es una aplicación de escritorio orientada a sistemas Windows y enfocada a la comunicación entre usuarios en Internet.

El repositorio se encuentra en GitHub (clonar proyecto) bajo licencia *GNU General Public License v3.0*.

### 1.1. Objetivo

Se busca como objetivo suplir las carencias que las actuales VoIP poseen: un control más técnico sobre la comunicación con los usuarios, facilidades para las tareas ordinarias, optimización de flujo de datos... todo ello acompañado de una mentalidad de software libre: los fuentes del programa están a disposición de los usuarios para que puedan compilar y usar su propia copia del programa. La idea es que cada usuario pueda aportar sus modificaciones a la aplicación oficial y, los trabajadores de Baco contemplen si dicha modificación propuesta pueda ser incluida en la aplicación oficial (en forma de nueva versión).

## CAPÍTULO 1. INTRODUCCIÓN

### 1.2. Justificación

Debido a que las principales aplicaciones de comunicación por Internet no daban la posibilidad de manejar los aspectos complejos de las comunicaciones se ha desarrollado esta aplicación. Dichos controles sobre el flujo de datos sobre Internet se entrará en detalle más adelante.

Debido a que, actualmente, predomina el software privativo, se ha optado por liberar el código para que la comunidad haga de él lo que le plazca.

### 1.3. Análisis de lo existente

En el mercado, existen multitud de aplicaciones VoIP, pero ninguna de ellas da las opciones que Baco proporciona. Si se continúa el desarrollo puede llegar a ser una competencia feroz ante los principales competidores: Discord, Skype, TeamSpeak... una de las características que no poseen estos competidores es la opción de suscripción a canales RSS. La pantalla principal de Baco nos muestra los canales RSS a los que el usuario está suscrito e implementa un visor de HTML cuyo motor de renderizado es Gecko e incorpora bibliotecas de Mozilla Firefox.

Baco no solo posee un visor de RSS, también nos proporciona un sistema de llamadas por Internet. Estas llamadas son entre amigos y pueden ser de cuantas personas se quiera. Además se da la opción de compartir en tiempo real la pantalla manejando parámetros de envío: tasa de capturas por segundo (FPS) y calidad de captura (menor calidad aumenta la velocidad de transmisión y reduce el peso de la imagen).

Mientras se está en llamada no se transmite continuamente la voz del usuario: solamente se envía en el caso de pasar un umbral mínimo y no rebasar un umbral máximo. Estos umbrales pueden modificarse en las opciones de Baco, dentro de las cuales podremos encontrar más campos a modificar en el apartado de audio (todas las opciones modificables se encuentran con un botón de restablecimiento al valor por defecto).

Por otra parte, dejando de lado la aplicación de cliente, tenemos un servi-

### *1.3. ANÁLISIS DE LO EXISTENTE*

dor que es el encargado de gestionar el funcionamiento de la aplicación. El servidor es una aplicación de consola y su función es controlar todo cuanto sucede en los clientes y en el propio servidor. Mientras el cliente esté dentro de la aplicación estará conectado al servidor.

Debido a que es CLI se ha implementado un sistema básico de comandos para el manejo y control en tiempo real del servidor.

La API es otra CLI con la diferencia que acepta peticiones HTTP y envía datos por red, es decir, actúa a modo de servidor. Ello es capaz de hacerlo gracias a Microsoft.EntityFrameworkCore.

Las peticiones HTML se resuelven y se retorna el JSON especificado. Los datos del JSON salen del proyecto Service, el cual es una biblioteca de clases que se encarga de la conexión con la base de datos mediante un contexto. Se hablará de ello mas adelante, en la subsección 5.4.4. Service hace las consultas al contexto de la base de datos con LINQ.

## *CAPÍTULO 1. INTRODUCCIÓN*

Esta página ha sido dejada en blanco intencionadamente.

# Capítulo 2

## Necesidades empresariales para el desarrollo

Dado que Baco es un programa de código abierto, los ingresos vienen por parte de las empresas que quieran utilizar este software. Los usuarios particulares podrán obtener una copia del programa de forma gratuita mientras que las empresas deberán pagar para obtenerlo. Puesto que la empresa tiene que pagar por el software se le ofrecen funcionalidades especiales:

- Soporte a los posibles problemas o errores que tengan con la aplicación
- Certificado de comunicaciones entre usuarios
- Cifrado de datos
- Posibilidad de un servidor privado

### 2.1. Recursos humanos

Al ser una empresa de software y con el panorama en la actualidad (la crisis de la COVID-19) se ha demostrado que el trabajo desde las casas de los trabajadores no es sino una ventaja para la misma empresa. Es por ello que

## *CAPÍTULO 2. NECESIDADES EMPRESARIALES PARA EL DESARROLLO*

Baco opta por una empresa descentralizada, sin ningún sitio fijo de trabajo. Con esto abaratamos los costes que suponga la compra o el alquiler del sitio de trabajo. En caso de reunión se dispondrá de un lugar para ello (comprado o alquilado por la empresa). No obstante se facilitarán los medios y materiales necesarios para los trabajadores. Entiéndase uno de los medios principales la comunicación entre los compañeros; para lograr esta comunicación se usará Baco, de este modo los mismos programadores estarían probando el programa todo el tiempo.

### **2.1.1. Trabajadores**

Debido a que Baco no es gratuito con fines empresariales habremos de tener una sección de comerciales que se encarguen de la contratación del software por parte de las empresas y otra de marketing para darse a conocer a ellas. Y por último tendríamos la parte imprescindible de la empresa, los programadores.

El salario de los trabajadores se ajustará a los establecido por el convenio del sector.

### **2.1.2. Contratos**

Debido a que se va a utilizar la línea de Internet por parte de los trabajadores desde su domicilio se pagará una cantidad determinada al trabajador en nómina en concepto de gastos por uso de Internet, telefónicos y electricidad. Todos los trabajadores tendrán un sistema de cobro complementario al salario por objetivos a modo de incentivo, así como un contrato a tiempo completo ordinario.

Los tipos de trabajadores podrían estructurarse de las siguiente forma:

- Programadores: la empresa proveerá los programas necesarios a los trabajadores para la realización de los trabajos. En esta categoría los programadores se enfrentarían a: desarrollo de nuevas funcionalidades de

## *2.1. RECURSOS HUMANOS*

Baco, implementación de funcionalidades cedidas por la comunidad (como bien se explicó en la sección 1.1) y mantenimiento de la página web, la cuál será el medio de descarga tanto como para empresas como para particulares.

- Comerciales: en el supuesto de visitas presenciales se pagará un plus de vestuario y transporte, ademas de los medios necesarios para el desplazamiento al cliente (si es requerido, la empresa alquilará a una empresa un vehículo para el empleado). La función de los comerciales será visitar empresas con el fin de lograr contratos con ellas para la cesión de software.
- Publicistas: estos serán los encargados de la parte que el público verá de Baco. La publicidad estará enfocada en empresas y particulares, centrándose principalmente en la publicidad en Internet. Al igual que los programadores, se les proporcionará acceso a cualquier tipo de software que requieran.
- Relaciones públicas (RR.PP.): será la primera persona con quien los particulares y las empresas hablarán si quieren contactar con Baco. Será quien redirija a los comerciales las empresas que se hayan interesado por el software, reenvíe las ideas que haya tenido la comunidad para implementar a Baco a los programadores, lleve las redes sociales e informe a los publicistas del impacto que haya tenido la publicidad de la empresa en el público.

El hecho de que haya varias categorías no implica que una persona no pueda realizar más de una, por ejemplo, un publicista puede también encargarse de la parte de RR.PP., o un comercial puede ser publicista. Si esto se diera supondría un incremento en el salario del trabajador. Este incremento junto con el complemento por trabajo realizado supone flexibilizar las capacidades de los trabajadores y no centrarlos solo en una materia dentro de la empresa, ya que esto podría suponer a la larga un decremento del interés. La empresa buscará por encima de todo unos trabajadores felices aunque ello suponga una pérdida de capital.

## *CAPÍTULO 2. NECESIDADES EMPRESARIALES PARA EL DESARROLLO*

### **2.1.3. Seguridad Social**

Las obligaciones en materia de seguridad social se dan a la hora de su puesta en marcha. Podemos dividirla en dos apartados:

- Trámites generales:
  - Alta de los socios y administradores en los regímenes de la Seguridad Social: debido a que la empresa está creada por una única persona y puesto que, evidentemente, se posee más del 50 % del capital el régimen será el de autónomos.
- Trámites en caso de contratar trabajadores:
  - Inscripción de la empresa: cuando, por vez primera, se quiera contratar trabajadores se habrá de solicitar la inscripción como empresario en la Tesorería General de la Seguridad Social (TGSS).
  - Afiliación de trabajadores (en el supuesto de que no estén afiliados): aquellos trabajadores que vayan a ser contratados y no tengan número de filiación (NAF) habrán de solicitarlo. Debe solicitarse a instancia del empresario.
  - Alta de los trabajadores en el Régimen de la Seguridad Social: cada trabajador habrá comunicarse su alta en el Régimen de la Seguridad Social correspondiente.

## **2.2. Prevención de riesgos laborales**

Dado que es una empresa de software no posee demasiados riesgos laborales graves. Los principales riesgos son obvios, la salud visual de los trabajadores, así como la física que se da a causa de malos hábitos por parte del trabajador. También cabe mencionar que los comerciales, al tener la opción del desplazamiento con vehículo tengan que ser tenidos en cuenta por posibles percances durante los trayectos a las empresas.

## 2.2. PREVENCIÓN DE RIESGOS LABORALES

### 2.2.1. Riesgos laborales y medidas preventivas

A continuación se enumerarán los principales riesgos laborales junto con los motivos del riesgo y las medidas preventivas correspondientes:

- Contactos eléctricos: puesto que los trabajadores trabajan con instrumentos conectados a la corriente queda constancia de que pueda suponer un riesgo a considerar.
  - No manipular ni intentar reparar equipos de trabajo conectados a la electricidad.
  - No desconectar aparatos tirando directamente del cable de corriente.
  - No usa regletas (comúnmente conocidas como «ladrones») que no garanticen la continuidad del conductor a tierra ni las conecte en serie.
  - Asegurarse de que los cables no se encuentran aplastados o dañados.
  - Desconectar los equipos cuando no se usen.
  - Evitar limpiar con líquidos cualquier equipo conectado a la red eléctrica.
  - No manipular instalaciones eléctricas con las manos húmedas.
- Condiciones ambientales generales: exposición a iluminación inadecuadas que puedan dar lugar a problemas visuales.
  - Adecuar la iluminación a las exigencias visuales.
  - Iluminar correctamente las zonas oscuras.
  - Para evitar reflejos o fuertes contrastes ninguna ventana se encontrará ni detrás ni delante de una pantalla.
  - Las ventanas han de tener elementos que amortigüen la entrada de luz para que, en el caso de molestias, pueda regularse la luz.

## *CAPÍTULO 2. NECESIDADES EMPRESARIALES PARA EL DESARROLLO*

- La iluminación artificial habría de ser difusa y proveniente de fuentes de luz de gran superficie.
- Reparar de manera inmediata aquellas fuentes de luz que pudieran parpadear.
- Para la prevención de la sequedad de ojos y mucosas se recomienda una elevada humedad relativa y la ventilación diaria del área de trabajo.
- La temperatura habría de estar a un nivel con el que el trabajador esté a gusto.
- Aislamiento de fuentes de ruido para evitar la exposición prolongada a ruidos elevados.
- Condiciones del puesto de trabajo: a consecuencia del entorno, organización, diseño del puesto de trabajo, carga mental, factores psicosociales y desempeño pueden producirse daños para la salud. Es por ello que habrán de ser analizados antes de que se produzcan.
  - Una correcta posición a la hora de sentarse:
    - Espalda apoyada en el respaldo y en ángulo recto con respecto a las piernas.
    - La altura de la silla ha de ser aquella sobre la cual los codos puedan descansar sobre la mesa formando un ángulo de 90º.
    - Si los pies no llegan al suelo se requerirá de un reposapiés.
  - Distribuir los elementos de trabajo de forma que estén más cerca los que más utilizados.
  - El espacio permitirá cambios de postura y movimientos.
  - Al realizarse en el domicilio el trabajo deberá estar aislado/separado para que permita cierta intimidad.
  - Planificación de las actividades a realizar y los descansos a realizar.
  - Evitar trabajar más de cuatro horas seguidas y cada dos realizar una pausa para relajar las musculatura.

## 2.2. PREVENCIÓN DE RIESGOS LABORALES

- Sobre esfuerzos: dado que el manejo del teclado y el ratón es imprescindible para la realización del trabajo, ello se considera como posible causa de lesiones como el síndrome del túnel carpiano (causado por el uso continuado del ratón).
  - Evitar apoyar el teléfono a la hora de hablar entre en hombro y la cabeza.
  - Los giros sobre la silla habrían de hacerse haciendo uso de los pies y no de giros bruscos de tronco.
  - No forzar la postura para alcanzar objetos que se encuentren lejos: levantarse para alcanzarlos.
  - No permanecer estático durante un tiempo excesivo.
  - Sentado no se habrían de manipular cargas.
- Usuarios de P.V.D. (Pantalla de Visualización de Datos): la empresa deberá de analizar la posibilidad de la aparición en sus empleados de daños relacionados con el manejo continuo de pantallas alfanuméricas o gráficas (recogido en el Real Decreto 488/1997 del 14 de Abril).
  - Colocar la pantalla lo más centrada para evitar molestias posturales del tronco y cervicales.
  - La pantalla deberá de estar más baja que el nivel de los ojos y de 60 a 80 cm. Nunca habrá de estar a menos de 40 cm o superior a 90 cm.
  - Parpadear frecuentemente para evitar la sequedad ocular y realizar pausas periódicas.
  - Desviar la vista periódicamente a objetos a media y larga distancia.
  - Mantener los antebrazos apoyados en la silla o en la mesa para evitar la suspensión de las muñecas en el uso del teclado y el ratón.

## *CAPÍTULO 2. NECESIDADES EMPRESARIALES PARA EL DESARROLLO*

### **2.3. Iniciativa emprendedora**

Para la creación de la empresa se deben de especificar ciertos detalles de ella, como la forma jurídica, cuál será el proceso de constitución y trámites para la puesta en marcha.

#### **2.3.1. Forma jurídica**

Debido a que los trabajadores realizarían su trabajo desde casa y la dirección de la empresa residiría en una oficina cuyo coste mensual no sería elevado se ha elegido la forma jurídica de Sociedad de Responsabilidad Limitada. El teletrabajo está regulado por el Acuerdo Marco Europeo sobre Teletrabajo [Ángel Belzunegui Eraso, 2008], [Prueba piloto de teletrabajo para el colectivo «dentro de convenio», 2005].

#### **2.3.2. Tramites administrativos**

Consideraremos trámites administrativos únicamente los de constitución y puesta en marcha de la sociedad.

##### **2.3.2.1. Constitución**

- Registro Mercantil Central: Certificación negativa del nombre de la sociedad
- Agencia Tributaria (AEAT): Número de identificación fiscal
- Notario: Escritura pública (la cual deberá presentarse a inscripción en el Registro Mercantil Provincial)
- Consejerías de Hacienda de las CC.AA: Impuesto sobre transmisiones patrimoniales y actos jurídicos documentados
- Registro Mercantil Provincial: Inscripción de la empresa en el Registro

## 2.3. INICIATIVA EMPRENDEDORA

### 2.3.2.2. Puesta en marcha

- AEAT:
  - Alta en el Censo de empresarios, profesionales y retenedores
  - Impuesto sobre Actividades Económicas (exentas las empresas de nueva creación durante los dos primeros ejercicios)
- TGSS:
  - Alta de los socios y administradores en los regímenes de la Seguridad Social
  - Inscripción de la empresa
  - Afiliación de trabajadores (en el supuesto de que no estén afiliados)
  - Alta de los trabajadores en el Régimen de la Seguridad Social
- Registro Mercantil Provincial:
  - Legalización del Libro de actas, del Libro registro de socios, del Libro-registro de acciones nominativas y del Libro registro de contratos entre el socio único y la sociedad
  - Legalización del Libro Diario y del Libro de Inventarios y Cuentas Anuales
- Autoridades de certificación: Obtención de un certificado electrónico
- Ayuntamientos: Licencia de actividad
- Otros organismos oficiales y/o registros: Inscripción en otros organismos oficiales y/o registros
- Servicio Público de Empleo Estatal: Alta de los contratos de trabajo
- Consejería de Trabajo de la CCAA: Comunicación de apertura del centro de trabajo

## *CAPÍTULO 2. NECESIDADES EMPRESARIALES PARA EL DESARROLLO*

- Inspección Provincial de Trabajo: Obtención del calendario laboral
- Oficina Española de Patentes y Marcas: Registro de signos distintivos (opcional)

# Capítulo 3

## Requisitos funcionales de la aplicación

Como bien se ha comentado en la sección 1.1, Baco ofrece manejar a niveles técnicos distintos aspectos, como la tasa de bits, el número de canales de audio, el búfer de sonido, la tasa de fotogramas que se envía, la calidad con la que se transmite vídeo... debido a esto, Baco sería ideal para aquellas personas que quieran modificar aspectos técnicos dentro de la aplicación (mediante las opciones que esta da), y las que quieran modificar los fuentes del programa para realizar acciones específicas (*open source*). También, debido a su simplicidad, puede ser usado por personas sin conocimientos informáticos relacionados con los tocados por la aplicación.

En general: Baco es una alternativa sencilla, simple y enfocada a todos los públicos.

### *CAPÍTULO 3. REQUISITOS FUNCIONALES DE LA APLICACIÓN*

Esta página ha sido dejada en blanco intencionadamente.

# Capítulo 4

## Análisis y Diseño

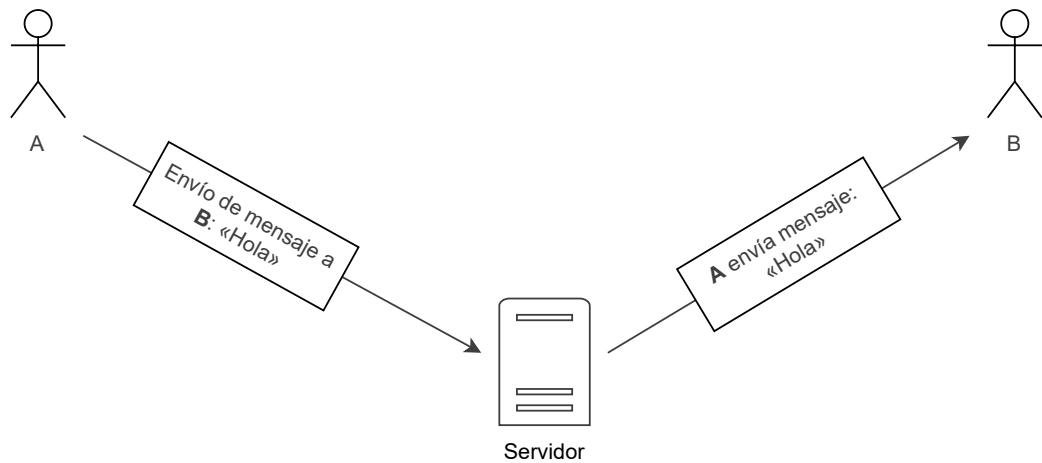
Para que la comunicación entre los usuarios sea posible ha de hacerse mediante un servidor pasarela. Dicho servidor es el encargado de registrar los usuarios conectados, mandar y almacenar mensajes de texto entre usuarios (almacenar aquellos que no puedan llegar al usuario destino), establecer llamadas entre usuarios (envío de audio y vídeo) y acceder al API REST. Los clientes acceden a la base de datos por medio del servidor, de este modo únicamente han de hacer una conexión por Internet; si el cliente está conectado al servidor, está en contacto con todos sus amigos y con la base de datos.

### 4.1. Diagrama de la arquitectura

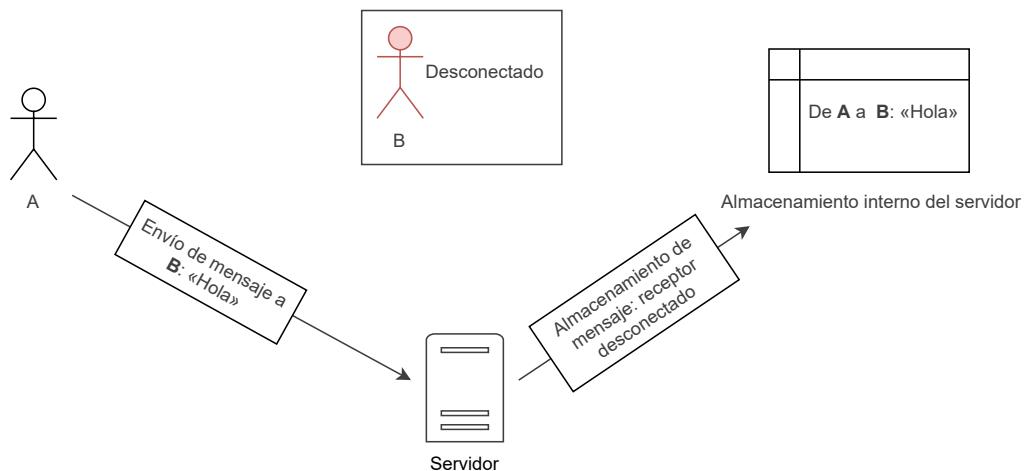
La arquitectura de Baco es relativamente sencilla: un usuario tiene amigos y puede comunicarse con ellos. Las comunicaciones se establecen mediante el servidor de Baco, quien se encarga de establecer las comunicaciones entre los usuarios.

Supongamos que un usuario (A) quiere enviar un mensaje de texto a un amigo (B) que está conectado:

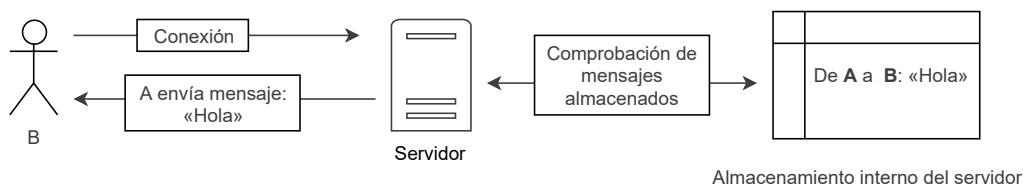
## CAPÍTULO 4. ANÁLISIS Y DISEÑO



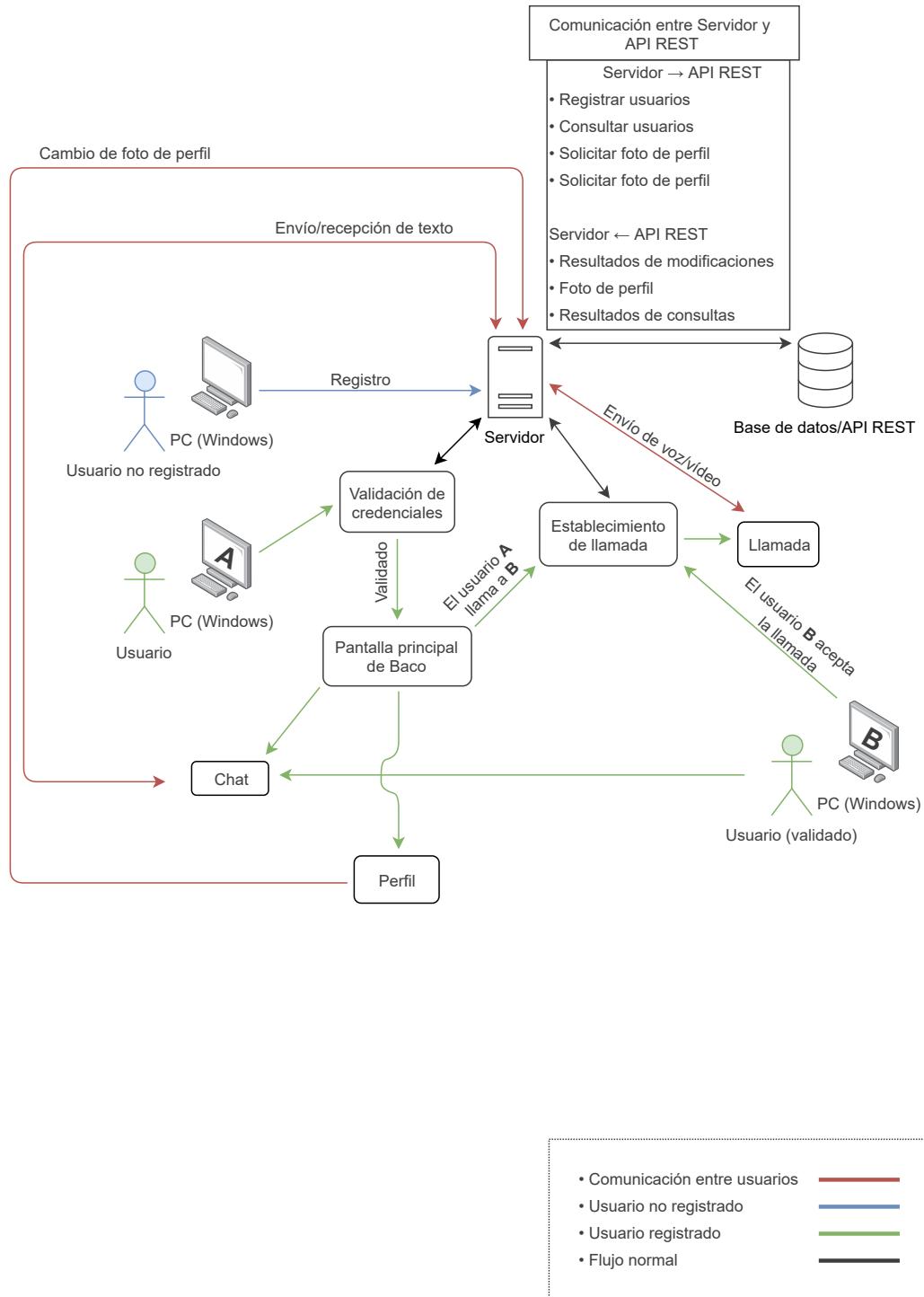
El funcionamiento es simple. En el supuesto de que B no esté conectado, cuando lo esté el mensaje le llegaría de igual manera.



Usuario B se conecta:



#### 4.1. DIAGRAMA DE LA ARQUITECTURA



Baco no solo es capaz de enviar a un único destinatario, es capaz de manejar grupos, es decir, un usuario envía un mensaje a múltiples usuarios y estos lo

## *CAPÍTULO 4. ANÁLISIS Y DISEÑO*

reciben en la misma conversación.

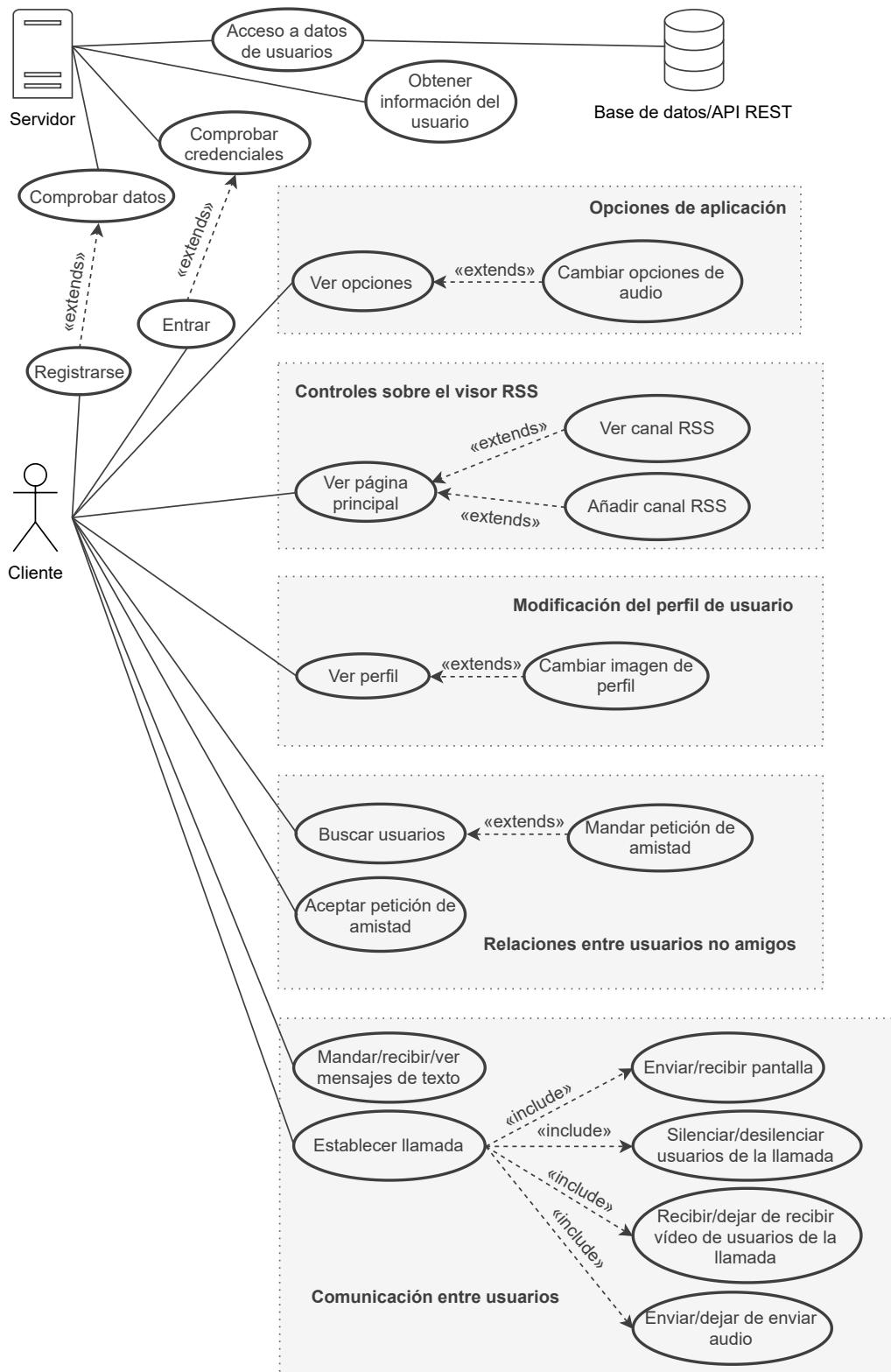
Si sustituimos los mensajes de texto por vídeo y audio obtenemos el sistema de llamada de Baco. Esto implica que las comunicaciones VoIP no han de ser exclusivamente de dos personas, sino que pueden ser de cuantas se quiera.

### **4.2. Diagrama de casos de uso**

A continuación se muestra el diagrama simplificado de casos de uso. En él podremos apreciar las diversas acciones que puede realizar el cliente (una vez registrado):

- Llamar a amigos
- Mandar y recibir mensajes de texto
- Ver y actualizar la imagen de perfil
- Ver los canales RSS que el usuario está suscrito y suscribirse a más
- Modificar opciones de audio

## 4.2. DIAGRAMA DE CASOS DE USO



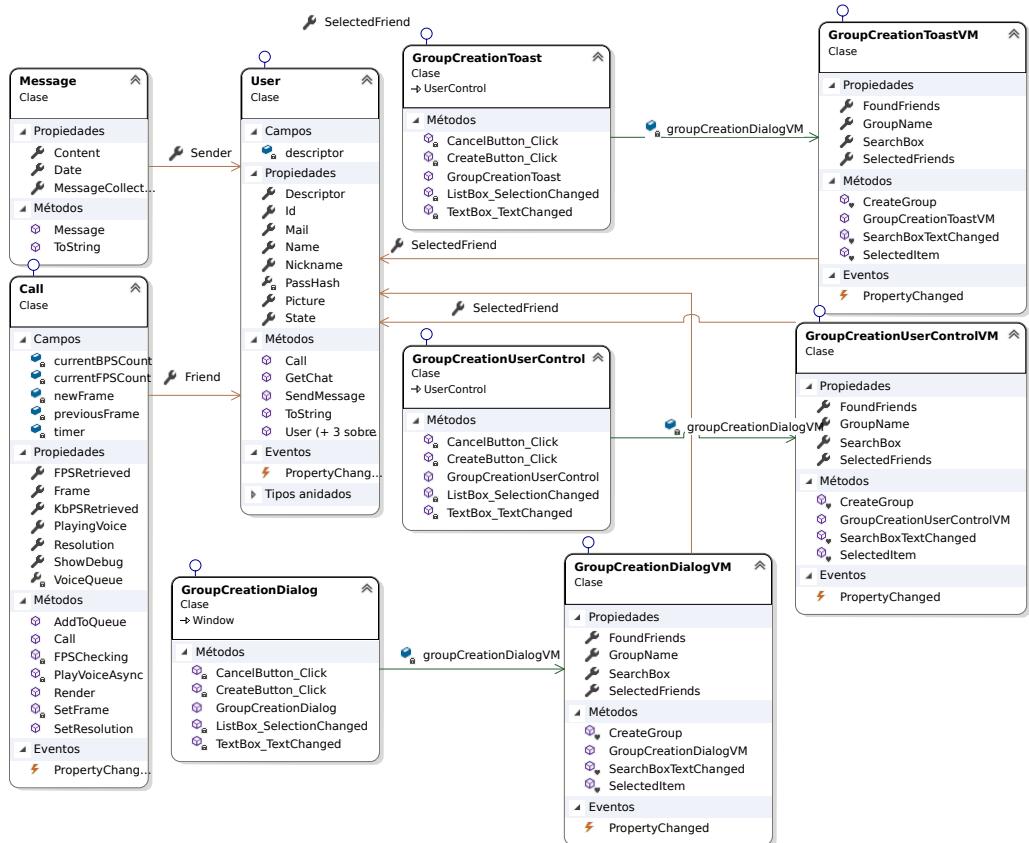
## CAPÍTULO 4. ANÁLISIS Y DISEÑO

### 4.3. Diagrama de clases

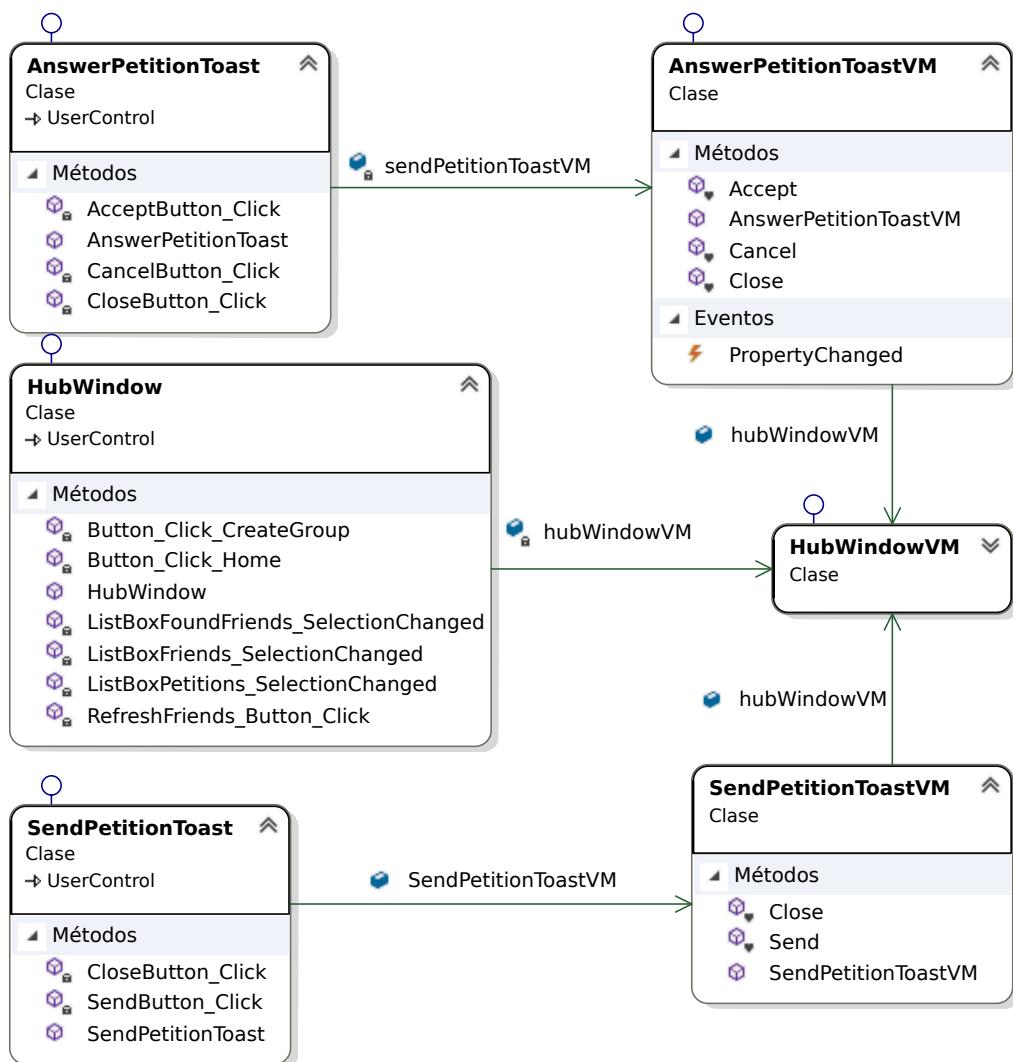
Debido a la complejidad de las clases en la solución de proyectos, se van a simplificar las relaciones. Se mostrarán los diagramas de clases del cliente y del servidor dado que el proyecto que maneja la API no posee complejidad alguna en este aspecto.

#### 4.3.1. Cliente

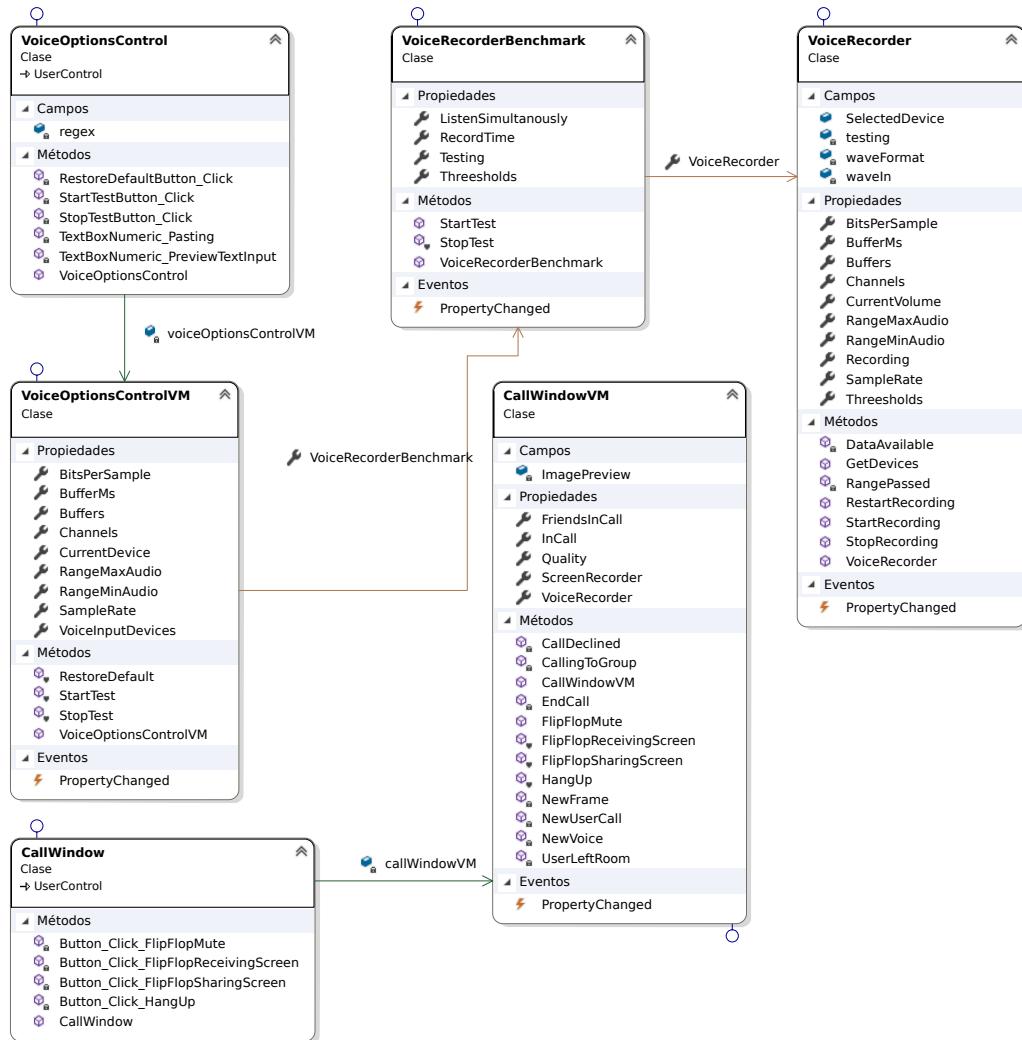
Las páginas siguientes mostrarán el diagrama de clases de Baco (aplicación de cliente).



### 4.3. DIAGRAMA DE CLASES

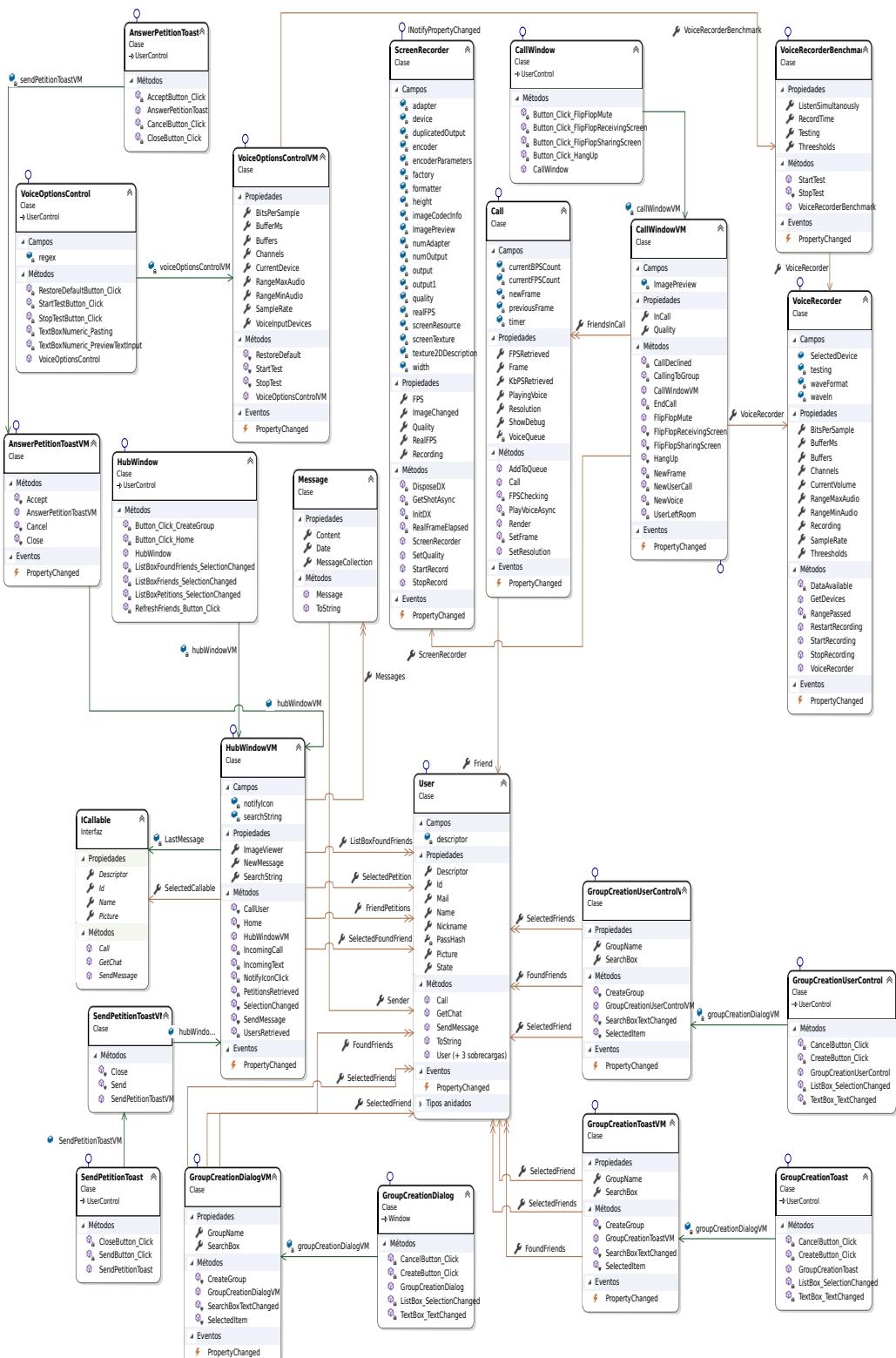


## CAPÍTULO 4. ANÁLISIS Y DISEÑO



Debido a que el diagrama es muy extenso no se han incluido las relaciones de colecciones; en el diagrama de a continuación se mostrara completo y con las colecciones, por ello se recomienda ver este documento con un visor de PDF.

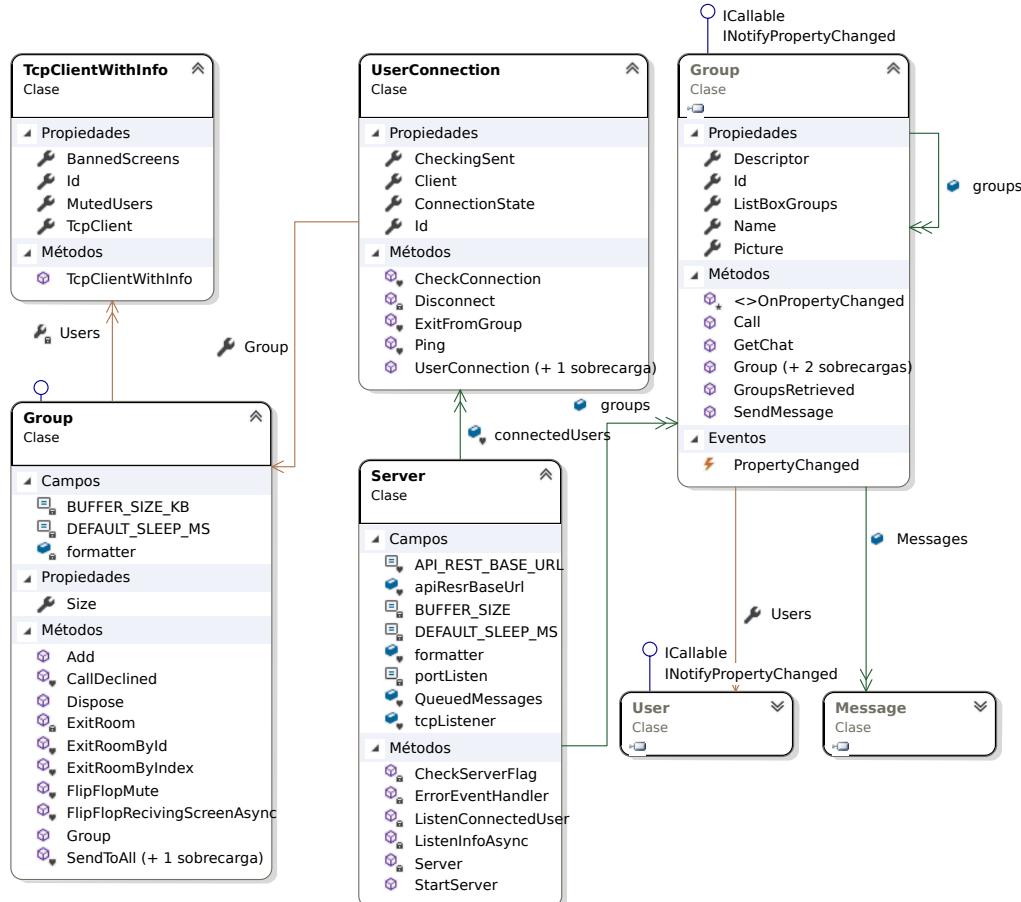
### 4.3. DIAGRAMA DE CLASES



## CAPÍTULO 4. ANÁLISIS Y DISEÑO

### 4.3.2. Servidor

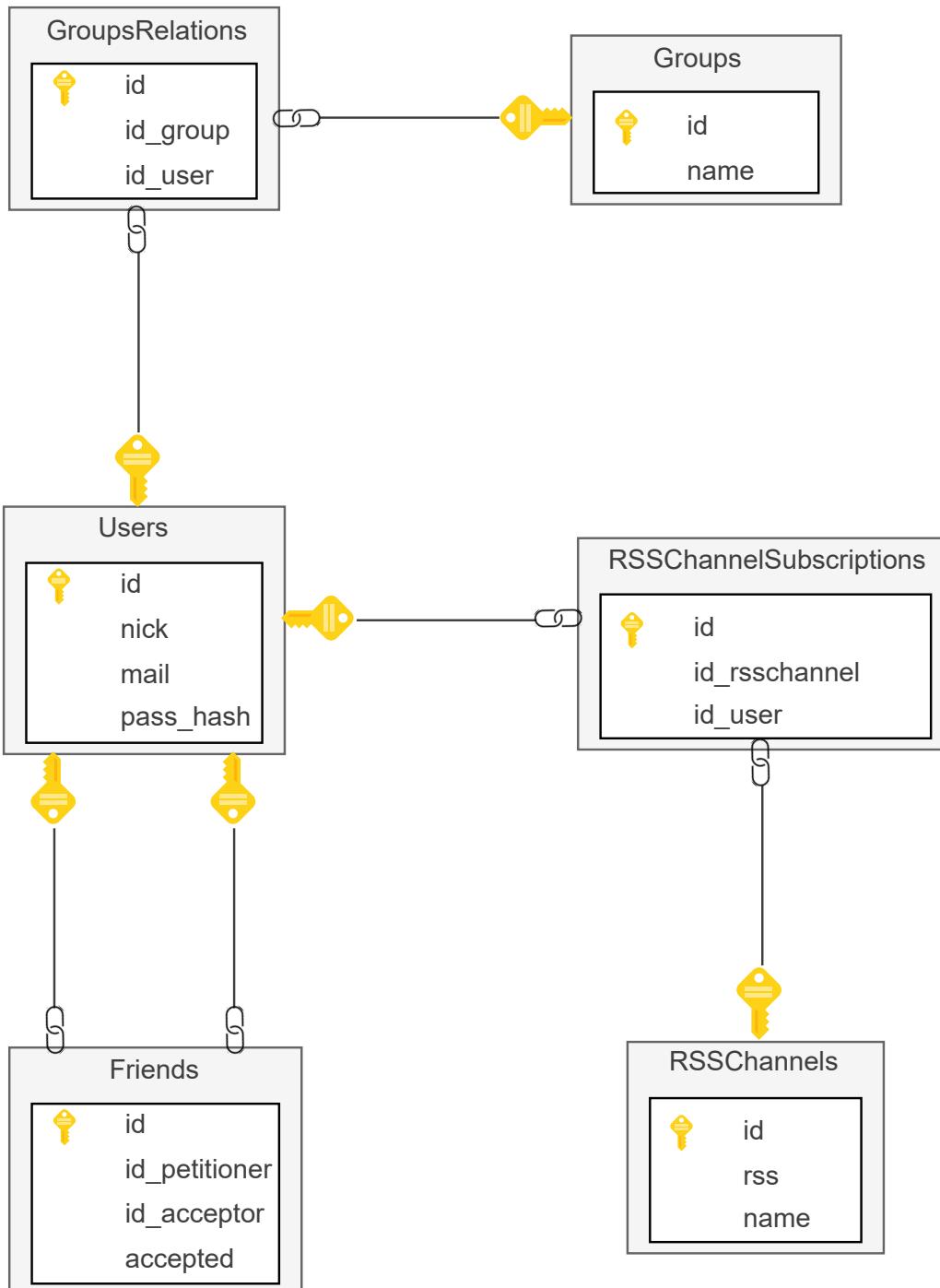
A continuación se muestra el diagrama del servidor.



## 4.4. Diseño de datos

Los datos se almacenan en base de datos relacional que se muestra a continuación:

#### 4.4. DISEÑO DE DATOS



A partir de aquí se hará referencia a las tablas de la base de datos con la siguiente terminología:

## CAPÍTULO 4. ANÁLISIS Y DISEÑO

- GroupsRelations: relación de grupos
  - id: identificador
  - id\_group: identificador del grupo
  - id\_user: identificador del usuario
- Groups: grupos
  - id: identificador
  - name: nombre del grupo
- Users: usuarios
  - id: identificador
  - nick: nombre de usuario
  - mail: correo del usuario
  - pass\_hash: hash de la contraseña del usuario
- Friends: amigos
  - id: identificador
  - id\_petitioner: identificador del usuario emisor de la petición
  - id\_acceptor: identificador del usuario receptor de la petición
  - accepted: estado de la petición
- RSSChannelSubscriptions: suscripciones de canales RSS
  - id: identificador
  - id\_rsschannel: identificador del canal RSS
  - id\_user: identificador del usuario
- RSSChannels: canales RSS
  - id: identificador
  - rss: dirección del canal

## 4.4. DISEÑO DE DATOS

- name: nombre del canal

Apréciese en el diagrama la estructura principal de datos donde:

- Un usuario puede tener muchos amigos
- Una relación de amistad solo puede tener dos amigos (emisor y receptor)
- Un usuario puede tener muchos grupos
- Un grupo puede tener muchos usuarios
- Un usuario puede tener muchas suscripciones a canales RSS

### 4.4.1. Amistades

Las relaciones de amistad entre usuarios se establecen gracias a la tabla en la base de datos de «amigos». En dicha tabla tenemos dos referencias a los usuarios: quien envía la petición y quien la recibe. El campo que indica el estado de la petición toma tres posibles valores: 1, 0 y nulo, donde:

- 1: aceptada - ambos usuarios son amigos
- 0: rechazada - el usuario receptor a rechazado la petición de amistad del emisor
- Nulo: sin respuesta - el usuario receptor aún no ha respondido a la petición de amistad

Debido a este establecimiento de estados nos resulta muy sencillo obtener las peticiones de amistad de cada usuario: si el identificador del usuario en la tabla de amigos tiene como estado «nulo» (siendo el receptor de la petición) sabemos que tiene una petición de amistad (del usuario emisor). Del mismo modo, obtener los amigos de un usuario no es más que buscar tanto en la columna de emisores como la de receptores y obtener el identificador del otro usuario.

Se hablará más adelante de la creación de la base de datos (en la subsección 5.4.4).

## *CAPÍTULO 4. ANÁLISIS Y DISEÑO*

Esta página ha sido dejada en blanco intencionadamente.

# Capítulo 5

## Codificación

Para la realización de Baco se han requerido diversos entornos, herramientas y lenguajes; todo ello se especificará en las secciones siguientes (sección 5.1, sección 5.2, sección 5.3).

### 5.1. Lenguajes de programación y marcado

Comentaremos los lenguajes que se han usado en los puntos siguientes.

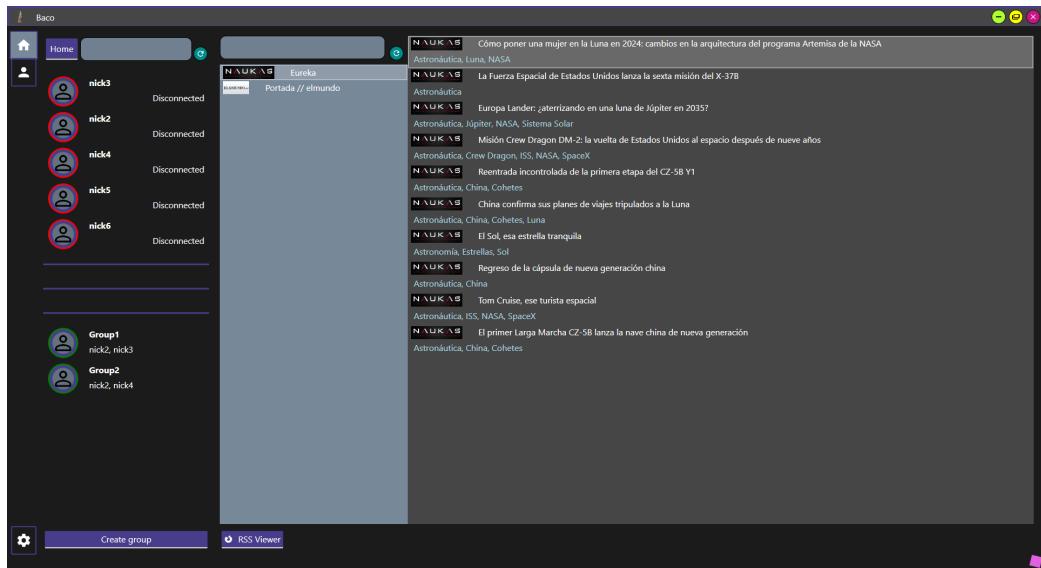
#### 5.1.1. C#

Se ha utilizado C# para el desarrollo de Baco debido a la familiaridad con este lenguaje.

#### 5.1.2. XAML

La interfaz gráfica (GUI) de Baco se ha realizado en WPF, por ello se ha hecho uso de XAML; es un lenguaje de marcado basado en XML y cuya finalidad es definir la GUI de nuestra aplicación.

## CAPÍTULO 5. CODIFICACIÓN



### 5.1.3. Transact-SQL (T-SQL)

T-SQL es un tipo de lenguaje de consultas estructurado desarrollado por Microsoft; es una extensión de SQL y lo enriquece para incluir diversos aspectos. La principal ventaja es la simplicidad de la exportación de bases de datos a Azure. El por qué de la elección de este lenguaje se especifica en la subsección 5.4.4.

## 5.2. Entornos de programación (IDE)

Los entornos han sido elegidos por la cercanía que se tenía a ellos: se habían usado con anterioridad y se tenía un manejo cómodo y con soltura. También se expresarán opiniones personales acerca de ellos.

### 5.2.1. Visual Studio

Visual Studio proporciona herramientas y facilidades (como su *IntelliSense*) que son superiores a las de sus competidores. Una gran ventaja es la admis-

## 5.2. ENTORNOS DE PROGRAMACIÓN (IDE)

nistración de paquetes que este posee (NuGet) el cual comentaremos en el siguiente punto.

### 5.2.1.1. NuGet

A continuación se enumeran los paquetes usados en la solución:

Nombre	Versión	Baco	BacoServer	ApiBaco	Service
Fody	6.1.1	Sí	No	No	No
Gecko	45.0.34	Sí	No	No	No
Microsoft.AspNetCore	2.2.0	No	No	Sí	No
Microsoft.AspNetCore.Mvc	2.2.0	No	No	Sí	No
Microsoft.AspNetCore.Mvc.Abstractions	2.2.0	No	No	No	Sí
Microsoft.Bcl	1.1.10	Sí	No	No	No
Microsoft.Bcl.Build	1.0.14	Sí	No	No	No
Microsoft.EntityFrameworkCore	3.1.2	No	No	Sí	No
Microsoft.EntityFrameworkCore.Design	3.1.2	No	No	No	Sí
Microsoft.EntityFrameworkCore.SqlServer	3.1.2	No	No	Sí	Sí
Microsoft.EntityFrameworkCore.Tools	3.1.2	No	No	Sí	Sí
Microsoft.Toolkit	6.0.0	Sí	No	No	No
Microsoft.Toolkit.Parsers	6.0.0	Sí	No	No	No
NAudio	1.10.0	Sí	No	No	No
Newtonsoft.Json	12.0.3	Sí	Sí	No	No
PropertyChanged.Fody	2.6.1	Sí	No	No	No
SharpDX	4.2.0	Sí	No	No	No
SharpDX.Direct3D11	4.2.0	Sí	No	No	No
SharpDX.DXGI	4.2.0	Sí	No	No	No
System.ServiceModel.Syndication	4.7.0	No	Sí	No	No

## CAPÍTULO 5. CODIFICACIÓN

### 5.2.2. SQL Server Management Studio (SSMS)

Debido a que T-SQL está desarrollado por Microsoft, el mejor IDE para crear y manejar T-SQL es el SSMS. Provee características para el manejo rápido y sencillo de las bases de datos del sistema: creación y modificación de tablas con una interfaz gráfica, edición de datos de tablas, exportación a Azure sencilla...

Cabe aclarar que no se ha hecho uso de las interfaces gráficas para la creación de las tablas necesarias por Baco, como bien se puede apreciar en la subsección 5.4.4.

## 5.3. Herramientas

Debido a que no solo hacen falta IDE para desarrollar un programa, también cabe mencionar otras herramientas que han sido de utilidad para pruebas. Estos programas se han utilizado exclusivamente para ver y modificar la base de datos.

### 5.3.1. Mozilla Firefox

Se ha hecho uso de Mozilla Firefox por su visor de JSON y para emular la comunicación que Baco tendría con el API REST. A diferencia de Chrome, Firefox viene con un visor JSON incorporado y no requiere de complementos de terceros para ello.

### 5.3.2. Postman

Debido a que los navegadores solo pueden realizar una acción (*GET*), Postman nos ofrece la posibilidad de enviar datos a nuestro API REST con acciones como *UPDATE*, *PUT* o *POST*. Si bien es cierto que Postman podría hacer el trabajo de Mozilla Firefox, este es más cómodo de manejar debido

## *5.4. ASPECTOS RELEVANTES DE LA IMPLEMENTACIÓN*

a que es un navegador web (es decir, no te da tantas opciones para enviar al servidor como Postman).

### **5.4. Aspectos relevantes de la implementación**

A continuación, se expondrán y comentarán fragmentos de código de difícil comprensión. Entiéndase por «difícil comprensión» aquellos fragmentos que queden fuera de los conocimientos adquiridos.

#### **5.4.1. API**

El API REST esta realizado en C#, .NET Core 3.1. Consta de dos proyectos: ApiBaco (apartado 5.4.1.1) y Service (apartado 5.4.1.2).

##### **5.4.1.1. ApiBaco**

Es una aplicación de consola que recibe las peticiones HTTP y envía los datos solicitados. Para la realización de este se ha tenido que estudiar como se comporta en C# [Learn Entity Framework Core, 2020].

##### **5.4.1.2. Service**

Es una librería de clases; aquí es donde reside la comunicación con la base de datos. Este proyecto es quien posee la lógica respecto al acceso a datos. El proyecto ApiBaco (apartado 5.4.1.1) llama a las funciones que este posee.

### **5.4.2. Cliente – Baco**

Pese a la magnitud del proyecto Baco, su composición es relativamente simple; se quieren matizar las siguientes clases debido a que su nivel de complejidad puede dar lugar a la no comprensión de la finalidad. También se explicará cómo se envía la pantalla por llamada.

## CAPÍTULO 5. CODIFICACIÓN

### 5.4.2.1. ScreenRecorder.cs

*ScreenRecorder* es la clase encargada de capturar y enviar la pantalla. Dicha acción se realiza haciendo uso de DirectX mediante el paquete SharpDX.

Vamos a explicar de forma rápida cómo se graba y envía el vídeo. Para comenzar, empezaremos por la función encargada de empezar a grabar:

```
278     public void StartRecord(Image imagePreview, int limitFPS = 0)
279     {
280         if (!Recording)
281         {
282             System.Timers.Timer timerRealFPS = new System.Timers.Timer
283             {
284                 Interval = 1000
285             };
286             timerRealFPS.Elapsed += RealFrameElapsed;
287             timerRealFPS.Start();
288             ImagePreview = imagePreview;
289             Recording = true;
290             FPS = limitFPS;
291             Task.Run(async () =>
292             {
293                 while (Recording)
294                 {
295                     await GetShotAsync();
296                     await Task.Delay(1000 / FPS);
297                     realFPS++;
298                 }
299             });
300         }
301     }
```

Código 5.1: Inicio de grabación de pantalla

Nada más iniciar la función, la línea 280 comprueba si se está grabando ya la pantalla; en caso contrario se procederá a comenzar a grabar. Desde la línea 282 hasta la 287 se configura el «temporizador» que va a ser nuestro contador de FPS reales. El temporizador ejecutará cada segundo la función que actualizará los FPS reales. Establecemos cuál será el control encargado de la previsualización de la captura y dejamos corriendo en un hilo la grabación. En el bucle de la línea 293 tenemos tres líneas; la primera accede a la función encargada de grabar y enviar la captura (), la segunda se espera el tiempo necesario entre captura y captura y por último añadimos uno al contador de

## 5.4. ASPECTOS RELEVANTES DE LA IMPLEMENTACIÓN

FPS reales.

La captura de pantalla se lleva a cabo con la función «*GetShotAsync*»:

```
156     private async Task GetShotAsync()
157     {
158         try
159         {
160             OutputDuplicateFrameInformation duplicateFrameInformation;
161             // Try to get duplicated frame within given time
162             duplicatedOutput.AcquireNextFrame(5000, out duplicateFrameInformation, out
163             screenResource);
164
165             ImageChanged = duplicateFrameInformation.LastPresentTime == 0;
166
167             // copy resource into memory that can be accessed by the CPU
168             using (Texture2D screenTexture2D = screenResource.QueryInterface<Texture2D>())
169             {
170                 device.ImmediateContext.CopyResource(screenTexture2D, screenTexture);
171             }
172
173             // Get the desktop capture texture
174             DataBox mapSource = device.ImmediateContext.MapSubresource(screenTexture, 0,
175                 MapMode.Read, MapFlags.None);
176             Rectangle boundsRect = new Rectangle(0, 0, width, height);
177
178             await Task.Run(() =>
179             {
180                 using (Bitmap bitmap = new Bitmap(width, height,
181                     System.Drawing.Imaging.PixelFormat.Format32bppArgb))
182                 {
183                     // Copy pixels from screen capture Texture to GDI bitmap
184                     BitmapData bitmapData = bitmap.LockBits(boundsRect,
185                         ImageLockMode.WriteOnly, bitmap.PixelFormat);
186                     IntPtr sourcePtr = mapSource.DataPointer;
187                     IntPtr destinationPtr = bitmapData.Scan0;
188                     for (int y = 0; y < height; y++)
189                     {
190                         // Copy a single line
191                         Utilities.CopyMemory(destinationPtr, sourcePtr, width * 4);
192
193                         // Advance pointers
194                         sourcePtr = IntPtr.Add(sourcePtr, mapSource.RowPitch);
195                         destinationPtr = IntPtr.Add(destinationPtr, bitmapData.Stride);
196                     }
197
198                     // Release source and dest locks
199                     bitmap.UnlockBits(bitmapData);
200
201                     device.ImmediateContext.UnmapSubresource(screenTexture, 0);
202
203             
```

## CAPÍTULO 5. CODIFICACIÓN

```
199         using (MemoryStream ms = new MemoryStream())
200     {
201         bitmap.Save(ms, imageCodecInfo, encoderParameters);
202         byte[] bitmapToSend = ms.ToArray();
203         Bitmap bmp;
204         using (MemoryStream msBmp = new MemoryStream(bitmapToSend))
205     {
206             bmp = new Bitmap(msBmp);
207         }
208
209         ImagePreview.Dispatcher.Invoke(new Action(() =>
210     {
211         ImagePreview.Source = ImageSourceFromBitmap(bmp);
212     }));
213
214         // Split and check changes
215         List<VideoFrame.FrameMapping> updatedFrames =
216             VideoFrame.UpdateFrameParts(bmp, imageCodecInfo,
217             encoderParameters);
218
219         if (updatedFrames != null) // Check if there is any changes
220             using (MemoryStream ms2 = new MemoryStream())
221         {
222             // Compress updatedFrames into byte[]
223             formatter.Serialize(ms2, new object[] { updatedFrames, new
224                 Point(width, height) });
225             Task.Run(() => Client.SendToServer(new
226                 ServerObject(ServerFlag.SendingData, new
227                     SenderObject(SenderFlags.Image, ms2.ToArray()))));
228         }
229     }
230     catch (SharpDXException e)
231     {
232         if (e.ResultCode.Code == SharpDX.DXGI.ResultCode.AccessLost.Result.Code)
233         {
234             Debug.WriteLine("Error GetShot ACCESS LOST – relaunch in 2s!");
235             Thread.Sleep(2000);
236             DisposeDX();
237             GC.Collect();
238             InitDX();
239         }
240         else if (e.ResultCode.Code != SharpDX.DXGI.ResultCode.WaitTimeout.Result.Code)
241         {
242             Debug.WriteLine(e.Message);
243             throw;
244         }
245     }
246 }
```

#### 5.4. ASPECTOS RELEVANTES DE LA IMPLEMENTACIÓN

```
244     }
245     finally
246     {
247         try
248         {
249             // Dispose manually
250             if (screenResource != null)
251             {
252                 screenResource.Dispose();
253                 screenResource = null;
254                 duplicatedOutput.ReleaseFrame();
255             }
256
257             // Force the Garbage Collector to cleanup memory to prevent memory leaks
258             await Task.Factory.StartNew(() => { GC.Collect(); });
259         }
260         catch (Exception e)
261         {
262             Debug.WriteLine("Error GetShot finnaly – relaunch in 2s!");
263             Thread.Sleep(2000);
264             DisposeDX();
265             GC.Collect();
266             InitDX();
267         }
268     }
269 }
270 }
```

Código 5.2: Grabación de pantalla y envío de vídeo

La función de grabación y emisión de vídeo tiene lugar de la línea 160 a la 226, y en ellas se aprecian las siguientes partes:

- 160 – 174: definición de cómo será la imagen capturada. Se obtienen las dimensiones de la pantalla para la captura y se copia la textura de la pantalla a memoria; la textura se extrae del objeto «*device*» (que aparece por primera vez en la línea 169) y se copia a otro objeto el cuál sabremos en todo momento dónde está (el objeto «*device*» podría cambiar su valor en cualquier momento). Y las últimas dos líneas dejan predefinido cómo tendrán que ser los datos más básicos de la imagen.
- 180 – 197: Creación de la imagen y liberación de la GPU. La primera instrucción especifica los atributos de la imagen resultante de la captura de pantallas y las dos siguientes nos harán falta para copiar en memoria

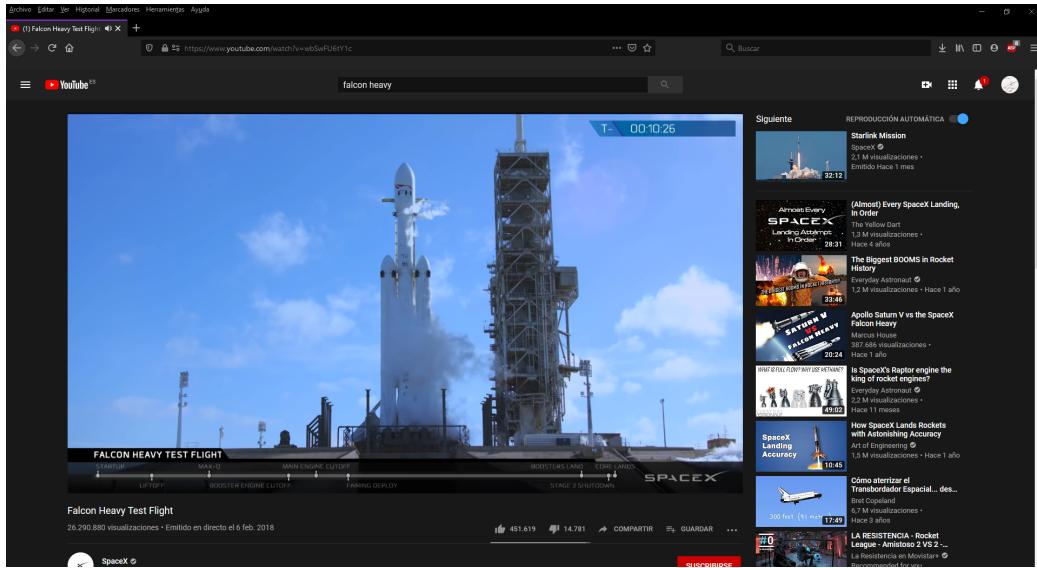
## CAPÍTULO 5. CODIFICACIÓN

de forma rápida la captura en forma de mapa de bits. El bucle que sigue será el encargado de ir de fila en fila de la imagen capturada e irá copiándose en memoria las filas que lea. Fura del bucle, en la línea 195, se desbloquea el mapa de bits con los atributos especificados y la línea siguiente libera el recurso que accede a la captura de pantalla por parte de la GPU; por ello, en el punto superior se comentaba que «*device*» puede cambiar de valor en cualquier momento. Dado que esto está sucediendo en un hilo a parte, otro hilo puede ahora acceder al objeto «*device*» y realizar, a la vez, las mismas acciones que estamos en este momento realizando.

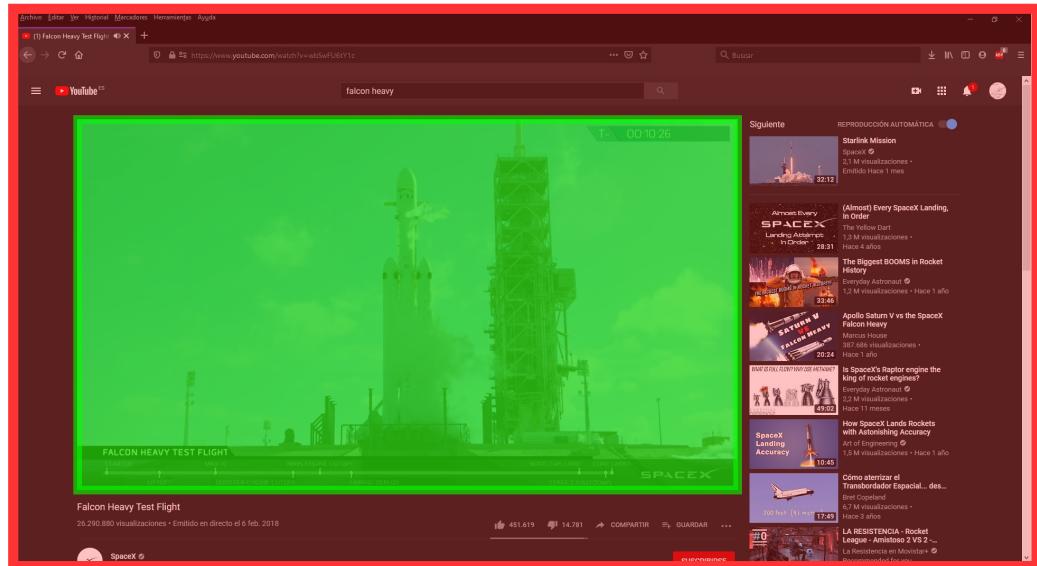
- 201 – 223: compresión de imagen, previsualización y envío. La primera línea almacena en un *stream* de memoria el mapa de bits resultante de las acciones anteriores y lo comprime con los códecs y los parámetros de envío. La imagen comprimida es mostrada en la línea 211. La siguiente instrucción (concretamente la función de la clase «*VideoFrame*») se encarga de:
  1. Cortar la imagen en porciones iguales.
  2. Comparar las porciones con las porciones de la imagen anterior (si no existe imagen anterior no se compararán).
  3. Determinar qué porciones son las que han cambiado.
  4. Devolver una lista con, únicamente, las porciones de la imagen que han cambiado.

Al enviar únicamente las porciones que cambian reducimos considerablemente el uso de red cuando se envía la pantalla, llegando a darse casos de 0 Kb/s en momentos en los que la pantalla no se mueve. La función que realiza las particiones emplea una algoritmia basada en probabilidades, se estima la probabilidad de que una fragmento haya cambiado con respecto al anterior, por ejemplo: dado el caso de una retransmisión de vídeo supongamos que el emisor del vídeo retransmite un vídeo; la pantalla que se transmite es la siguiente:

## 5.4. ASPECTOS RELEVANTES DE LA IMPLEMENTACIÓN



Como podremos suponer, en la zona del vídeo es dónde, principalmente, sucederán los cambios. En la siguiente imagen se resalta:



Véase la zona coloreada en verde como los fragmentos que más probabilidad tienen de cambiar y los rojos los que menos. En este supuesto,

## CAPÍTULO 5. CODIFICACIÓN

si el usuario no se desplaza por la página, solamente se enviará la zona sombreada en verde, que es la zona que va a cambiar, el resto de las zonas (la parte sombreada en rojo) no se van a comprobar todas para ver si hay cambios, solamente algunas dado que la probabilidad de que haya algún cambio en ella es baja.

En el supuesto de que no haya ningún cambio se retorna un valor nulo. Dicho valor nulo nos servirá para, en la línea 217, enviar o no la pantalla; evidentemente, si no hay cambios no se envía nada. En el caso de haberlos se comprime la lista en memoria junto con la especificación de las dimensiones de la pantalla (línea 221), de este modo sabremos cuál es la posición de cada bloque que enviamos con respecto a la resolución. Por último se envía el objeto serializado en memoria al servidor para que él lo retransmita a los usuarios de la llamada (solo a aquellos que estén viendo la retransmisión).

### 5.4.2.2. VoiceRecorder.cs

La clase *VoiceRecorder* detecta los dispositivos de entrada de audio y captura el sonido para ser enviado por red o para hacer pruebas de sonido (en el menú de opciones). El paquete que hace esto posible es NAudio.

Funciona de un modo similar a la clase ScreenRecorder.cs, solo que en vez de capturar la pantalla lee el voltaje del dispositivo de entrada de audio. Esto sucede en la siguiente función:

```
59     public void StartRecording()
60     {
61         waveFormat = new WaveFormat(SampleRate, BitsPerSample, Channels);
62         waveIn = new WaveInEvent
63         {
64             BufferMilliseconds = BufferMs,
65             DeviceNumber = SelectedDevice,
66             WaveFormat = waveFormat,
67             NumberOfBuffers = Buffers
68         };
69         waveIn.DataAvailable += DataAvailable;
70         waveIn.StartRecording();
71         Recording = true;
```

#### 5.4. ASPECTOS RELEVANTES DE LA IMPLEMENTACIÓN

72 }

Código 5.3: Inicio de captura de micrófono

La primera línea nos permite determinar cómo vamos a transformar la señal analógica del micrófono a su señal digital. A continuación, en las líneas que van de la 62 a la 68, se establecen los parámetros que indican como se ha de capturar la información del micrófono:

1. El búfer: cuánto tiempo se va guardar en el búfer.
2. El dispositivo de entrada: indicará el índice del dispositivo a leer.
3. Formato de la modulación por impulsos codificados: lo comentado anteriormente, cómo se va a transformar la señal analógica en digital.
4. Número de búferes: cuántos búferes se usarán.

Antes de comenzar a leer en la línea 70, la instrucción anterior es la suscripción al evento «*DataAvailable*». Este evento se lanza cada vez que se llene el búfer que hayamos especificado y será donde se envíe el audio. La función suscrita es la siguiente:

```
98     private void DataAvailable(object sender, WaveInEventArgs e)
99     {
100         if (RangePassed(e))
101             Task.Run(() =>
102             {
103                 if (waveIn.DeviceNumber != SelectedDevice)
104                 {
105                     RestartRecording();
106                     return;
107                 }
108
109                 using (MemoryStream ms = new MemoryStream())
110                 {
111                     using (WaveFileWriter writer = new WaveFileWriter(ms, waveFormat))
112                     {
113                         writer.Write(e.Buffer, 0, e.BytesRecorded);
114                     }
115                     if (!testing)
116                         Client.SendToServer(new ServerObject(ServerFlag.SendingData, new
117                                         SenderObject(SenderFlags.Voice, ms.ToArray())));
else
```

## CAPÍTULO 5. CODIFICACIÓN

```
118             AudioUtils.PlaySound(ms.ToArray());  
119         }  
120     });  
121 }
```

Código 5.4: Búfer de lectura lleno

Primeramente nos encontramos una condición para enviar el audio; dicha condición supondrá si el audio disponible en el búfer está en el rango de volumen para ser enviado (el sonido no es ni muy alto ni muy bajo). En el caso de estarlo se comprueba si el dispositivo de entrada ha cambiado, si no, se comprime (serializando) el audio. Antes de ser enviado el audio comprimido se comprueba si se está en llamada o se está haciendo pruebas en el menú de opciones. Si se están haciendo pruebas, el sonido se reproduce por el dispositivo de salida del propio ordenador, si no se están haciendo pruebas se envía al servidor para su retransmisión a los que estén escuchando al usuario en la llamada.

### 5.4.3. Servidor – BacoServer

Debido a que el servidor actúa principalmente a modo de pasarela, no se ve necesario ninguna explicación del código. También es el encargado de tener un registro de los usuarios que están conectados en cada momento y manejar las llamadas.

### 5.4.4. Base de datos

Dada la elección de base de datos de Microsoft SQL se ha tenido que implementar la base de datos en su lenguaje nativo: T-SQL. Microsoft SQL nos permite la exportación a objetos desde la base de datos a nuestra solución por medio de la consola del administrador de paquetes NuGet. Hacen falta los paquetes de Microsoft.EntityFrameworkCore, Microsoft.EntityFrameworkCore.Design, Microsoft.EntityFrameworkCore.SqlServer y Microsoft.EntityFrameworkCore.Tools. Tan solo es necesario un comando tanto como para crear como para actualizar:

#### 5.4. ASPECTOS RELEVANTES DE LA IMPLEMENTACIÓN

```
Scaffold-DbContext "Server=DirecciónInstancia; Database=NombreDB;  
TrustedConnection=True;" Microsoft.EntityFrameworkCore.SqlServer  
-OutputDir CarpetaSalida -Context "NombreContexto" -DataAnnotations
```

En mi caso, dado:

- **DirecciónInstancia:** localhost/SQLEXPRESS05
- **NombreDB:** baco
- **CarpetaSalida:** Model
- **NombreContexto:** BacoContext

tendríamos el comando de la siguiente manera:

```
Scaffold-DbContext "Server=localhost/SQLEXPRESS05; Database=baco;  
Trusted_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer  
-OutputDir Model -Context "BacoContext" -DataAnnotations
```

En el caso de querer actualizar el proyecto con la base de datos solo tendríamos que añadir el parámetro “-Force”:

```
Scaffold-DbContext "Server=localhost/SQLEXPRESS05; Database=baco;  
Trusted_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer  
-OutputDir Model -Context "BacoContext" -DataAnnotations -Force
```

Una vez realizado el comando sobre un proyecto (en nuestro caso «Service») nos quedaría una estructura de directorios como esta:

```
Service  
└ Model ..... Carpeta autogenerada por el comando de mapeo  
    └ BacoContext.cs .... Contexto sobre el cual se accede a los datos  
    └ Friends.cs  
    └ Groups.cs  
    └ GroupsRelations.cs  
    └ Rsschannels.cs
```

## CAPÍTULO 5. CODIFICACIÓN



donde BacoContext.cs es el contexto con el cual vamos a acceder a la base de datos. El resto de objetos son los que nos ha mapeado automáticamente Microsoft.EntityFrameworkCore y los cuales se usarán para recibir los datos del contexto.

Para el acceso a los datos en Baco se ha decidido que el proyecto Service sea una biblioteca de clases. A dicha biblioteca accede el proyecto ApiBaco. ApiBaco es el servicio REST de nuestro proyecto; haciendo uso de la librería Service es capaz de acceder a la base de datos.

La disposición de datos (cliente y API REST) nos permite tener todo en una misma máquina (no obstante podría soportar estar en máquinas separadas el API REST y el servidor).

Puesto que no se ha impartido el lenguaje se va a mostrar el código necesario para la creación de la base de datos con valores de pruebas:

```
1 CREATE DATABASE baco;
2 USE baco;
3
4 CREATE TABLE Users (
5     id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
6     nick VARCHAR(16) NOT NULL,
7     mail VARCHAR(255) NOT NULL,
8     pass_hash VARCHAR(60) NOT NULL
9 );
10
11 CREATE TABLE Friends (
12     id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
13     id_petitioner INT NOT NULL,
14     id_acceptor INT NOT NULL,
15     accepted BIT DEFAULT NULL,
16     FOREIGN KEY (id_petitioner)
17     REFERENCES Users(id),
18     FOREIGN KEY (id_acceptor)
19     REFERENCES Users(id)
20 );
21
22 CREATE TABLE RSSChannels (
23     id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
24     rss VARCHAR(64) NOT NULL,
25     name VARCHAR(24) NOT NULL
26 );
27
```

#### 5.4. ASPECTOS RELEVANTES DE LA IMPLEMENTACIÓN

```
28 CREATE TABLE RSSChannelSubscriptions (
29     id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
30     id_rsschannel INT NOT NULL,
31     id_user INT NOT NULL,
32     FOREIGN KEY (id_rsschannel)
33     REFERENCES RSSChannels(id),
34     FOREIGN KEY (id_user)
35     REFERENCES Users(id)
36 );
37
38 CREATE TABLE Groups (
39     id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
40     name VARCHAR(24) NOT NULL
41 );
42
43 CREATE TABLE GroupsRelations (
44     id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
45     id_group INT NOT NULL,
46     id_user INT NOT NULL,
47     FOREIGN KEY (id_group)
48     REFERENCES Groups(id),
49     FOREIGN KEY (id_user)
50     REFERENCES Users(id)
51 );
52
53 INSERT INTO Users (nick, mail, pass_hash)
54 VALUES
55 ('nick1', 'mail1', 'nIdNeuo3T99bvMKcHSdKOedQ9rPcNAptEErOt26PuxE='),
56 ('nick2', 'mail2', 'nIdNeuo3T99bvMKcHSdKOedQ9rPcNAptEErOt26PuxE='),
57 ('nick3', 'mail3', 'nIdNeuo3T99bvMKcHSdKOedQ9rPcNAptEErOt26PuxE='),
58 ('nick4', 'mail4', 'nIdNeuo3T99bvMKcHSdKOedQ9rPcNAptEErOt26PuxE='),
59 ('nick5', 'mail5', 'nIdNeuo3T99bvMKcHSdKOedQ9rPcNAptEErOt26PuxE='),
60 ('nick6', 'mail6', 'nIdNeuo3T99bvMKcHSdKOedQ9rPcNAptEErOt26PuxE='),
61 ('nick7', 'mail7', 'nIdNeuo3T99bvMKcHSdKOedQ9rPcNAptEErOt26PuxE=');
62
63 INSERT INTO Friends (id_petitioner, id_acceptor, accepted)
64 VALUES
65 (1, 3, 1),
66 (2, 1, 1),
67 (1, 4, 1);
68
69 INSERT INTO RSSChannels (rss, name)
70 VALUES
71 ('http://feeds.feedburner.com/naukas/danielmarin', 'Eureka'),
72 ('https://e00-elmundo.uecdn.es/elmundo/rss/portada.xml', 'Portada // elmundo');
73
74 INSERT INTO RSSChannelSubscriptions (id_rsschannel, id_user)
75 VALUES
76 (1, 1),
77 (2, 1),
```

## CAPÍTULO 5. CODIFICACIÓN

```
78 (2, 2);  
79  
80 INSERT INTO Groups (name)  
81 VALUES  
82 ('Group1'),  
83 ('Group2');  
84  
85 INSERT INTO GroupsRelations (id_group, id_user)  
86 VALUES  
87 (1, 1),  
88 (1, 2),  
89 (1, 3),  
90 (2, 2);  
91 (2, 1),  
92 (2, 4),  
93 (2, 5);
```

Código 5.5: Script T-SQL de creación de la base de datos de Baco

Hasta la línea 51 el script se encarga de la creación de las tablas base de datos. Véase que T-SQL es igual de SQL. De ahí hasta el final es la inserción de datos a las tablas creadas.

Se ha querido plasmar el código T-SQL para demostrar que no posee diferencias notables respecto a SQL.

Puesto que la base de datos está alojada en el ordenador personal, la versión final habría de ser exportada a una base de datos como Azure. Para ello se hará uso de Microsoft Data Migration Assistant. Una vez se ha hecho la migración, para conectarla con nuestro servicio tan solo habría que actualizar como se hizo en la cadena de actualización de servicio pero con ligeros cambios:

```
Scaffold-DbContext "Server=URLAzure;Database=NombreAzureDB;User  
Id=Usuario;Password=Contraseña"  
Microsoft.EntityFrameworkCore.SqlServer -OutputDir CarpetaSalida -Context  
"NombreContexto" -DataAnnotations -Force
```

En mi caso seria la siguiente cadena:

```
Scaffold-DbContext  
"Server=dbcarlosclement.database.windows.net;Database=BacoDB;User
```

#### *5.4. ASPECTOS RELEVANTES DE LA IMPLEMENTACIÓN*

```
Id=administrador;Password=#admin123"  
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Model -Context  
"BacoContext" -DataAnnotations -Force
```

## *CAPÍTULO 5. CODIFICACIÓN*

Esta página ha sido dejada en blanco intencionadamente.

# Capítulo 6

## Manuales de usuario

Debido a que se han entregado dos proyectos con instaladores se ha procedido a elaborar dos manuales de usuario: el de cliente y el del servidor. El servidor también tiene un manual dado que las empresas pueden interesarse por tener un servidor privado.

Los manuales se anexan a escala 0.75; si se quieren ver a escala 1 estos manuales están incluidos en los ficheros entregados.

### 6.1. Baco

Anexado en esta sección se encuentra el manual de usuario de Baco.

## *CAPÍTULO 6. MANUALES DE USUARIO*

Esta página ha sido dejada en blanco intencionadamente.

*6.1. BACO*

Manual de usuario: Baco

Baco SL

26 de mayo de 2020

*CAPÍTULO 6. MANUALES DE USUARIO*

Esta página ha sido dejada en blanco intencionadamente.

# Índice general

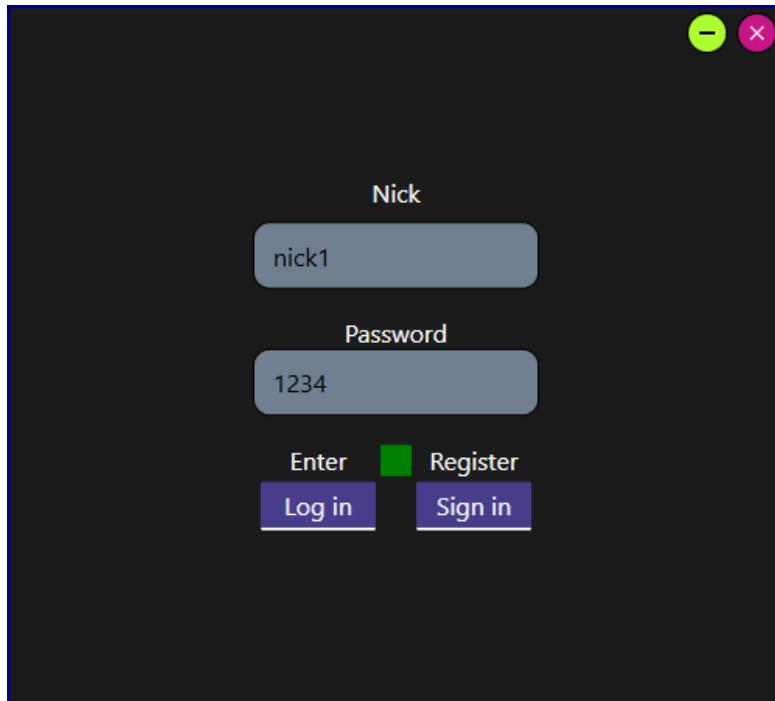
<b>Primeros pasos</b>	<b>5</b>
Especificación de la dirección del servidor (avanzado) . . . . .	8
<b>Entrada al sistema e interfaz</b>	<b>9</b>
Visor RSS . . . . .	10
Chat . . . . .	12
Llamada . . . . .	13
Perfil . . . . .	14
Opciones . . . . .	15
<b>Amigos</b>	<b>15</b>

## *CAPÍTULO 6. MANUALES DE USUARIO*

Esta página ha sido dejada en blanco intencionadamente.

## Primeros pasos

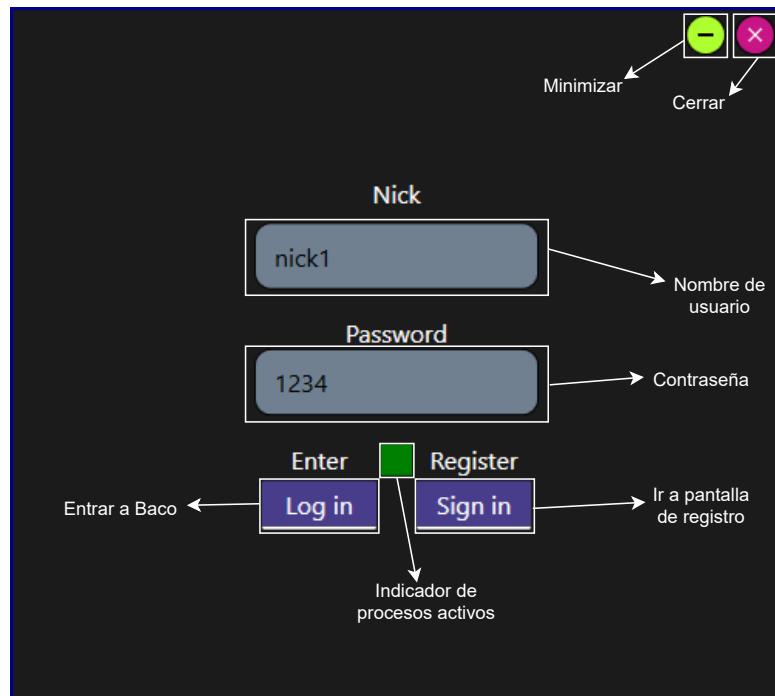
El sistema de registro de Baco es muy simple y rápido. Tan solo tienes que llenar los campos que se muestran en el botón *sign in*. La interfaz del lanzador de Baco:



Donde los controles son los siguientes:

## CAPÍTULO 6. MANUALES DE USUARIO

### ÍNDICE GENERAL



En el caso de tener cuenta introduciríamos las credenciales (en nuestro caso «nick1» y como contraseña «1234») y presionaríamos el botón «*Log in*». En este momento empezará a enviar los datos al servidor para confirmarlos. En caso de confirmación negativa se mantendría esta pantalla. Si la confirmación es positiva entrará a la aplicación.

Si se pulsa el botón «*Sign in*» se mostraría la siguiente pantalla:

### 6.1. BACO

The screenshot shows a registration form on a dark-themed website. At the top right are a green minus sign and a red X button. On the left is a blue back arrow labeled "Ir a pantalla de login". The form fields are:

- Nick**: A red-bordered input field.
- Password**: A red-bordered input field containing a single character.
- Password confirmation**: A red-bordered input field.
- Mail**: A red-bordered input field.

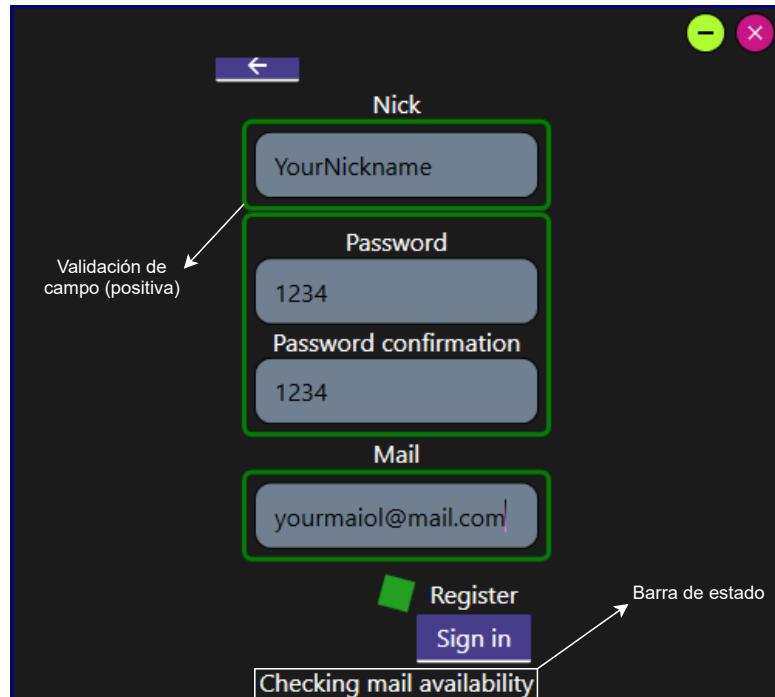
Below the form is a button bar with a green square icon labeled "Register" and a blue square icon labeled "Sign in". Arrows point from the text labels to their corresponding elements:

- "Ir a pantalla de login" points to the back arrow.
- "Validación de campo (negativa)" points to the Password field.
- "Confirmación de contraseña" points to the Password confirmation field.
- "Correo electrónico de la cuenta" points to the Mail field.
- "Envío de solicitud de registro" points to the "Sign in" button.

Respecto a la vista anterior se observa que hay dos campos más: confirmación de contraseña y correo electrónico. Conforme se vayan rellenando los campos con los datos que se quiera tener se irá comprobando con el servidor la disponibilidad de ellos. En la vista siguiente se muestran todos los campos validados con el servidor:

## CAPÍTULO 6. MANUALES DE USUARIO

### ÍNDICE GENERAL



Esto quiere decir que:

- No hay ningún usuario con ese nombre
- Los datos de los campos de contraseña y confirmación de contraseña son idénticos
- El correo electrónico no consta como el correo de ningún usuario registrado

### Especificación de la dirección del servidor

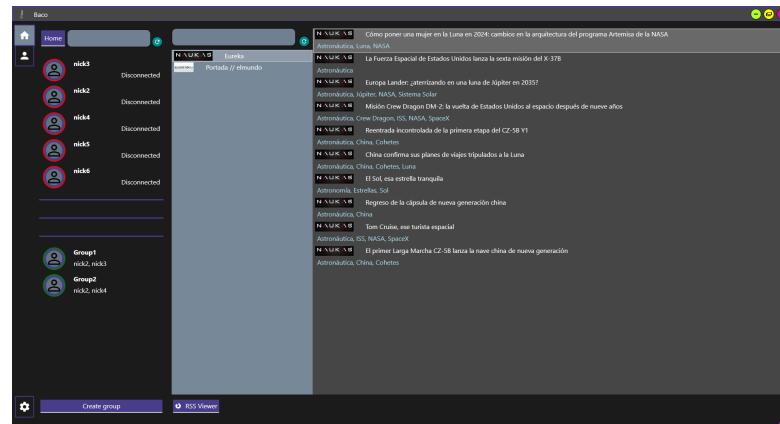
Si se quiere cambiar la dirección a la que se quiere acceder, en el lanzador de Baco tiene que pulsarse el indicador de procesos activos; este abrirá un

## 6.1. BACO

diálogo donde se nos permitirá ingresar una dirección alternativa a la predeterminada.

### Entrada al sistema e interfaz

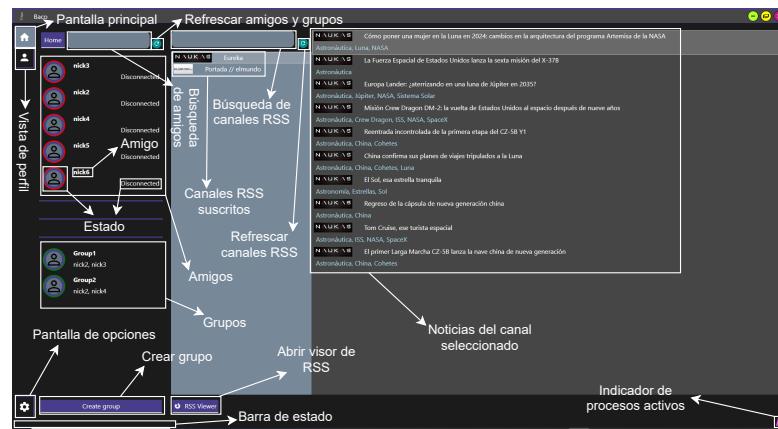
Realizada la confirmación de datos con el servidor se accede a la siguiente pantalla:



La cual sus controles son los que se exponen a continuación:

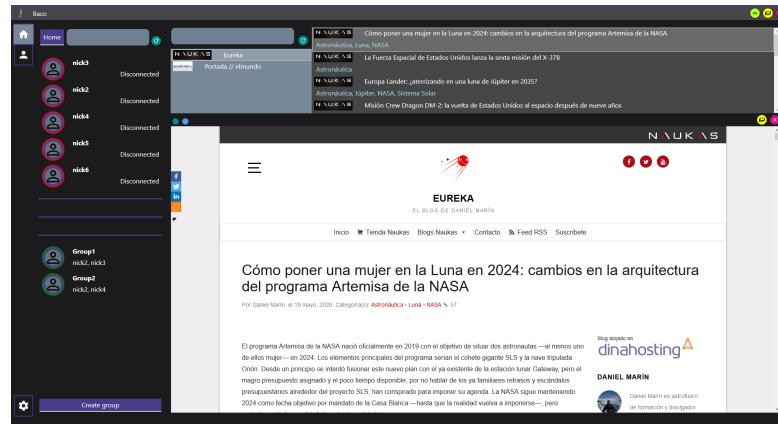
## CAPÍTULO 6. MANUALES DE USUARIO

### ÍNDICE GENERAL



### Visor RSS

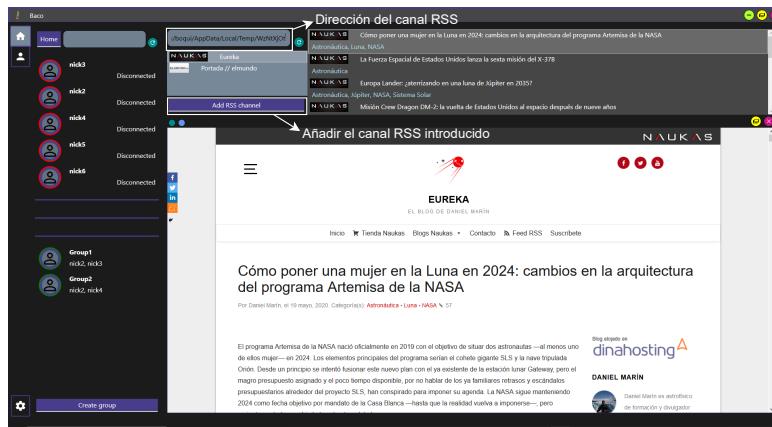
Cuando se abre el visor RSS (*«RSS viewer»*) sobre la misma pantalla se mostrará un cliente web:



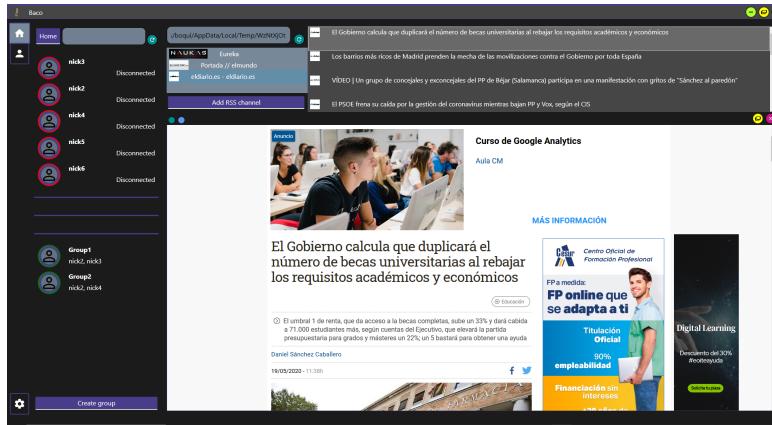
En la barra de búsqueda de canales podemos introducir la dirección de un canal de RSS para suscribirse a él. En cuanto la barra de búsqueda tenga

## 6.1. BACO

carácteres se mostrará un botón en la parte inferior para añadir el canal:



Y una vez añadido tendremos un nuevo proveedor:

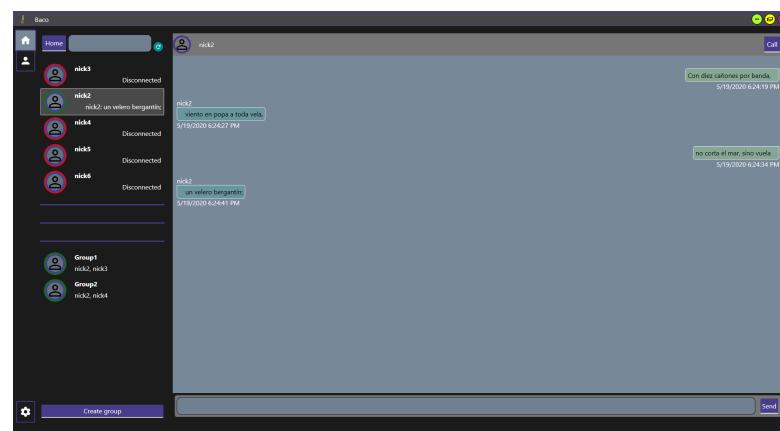


## CAPÍTULO 6. MANUALES DE USUARIO

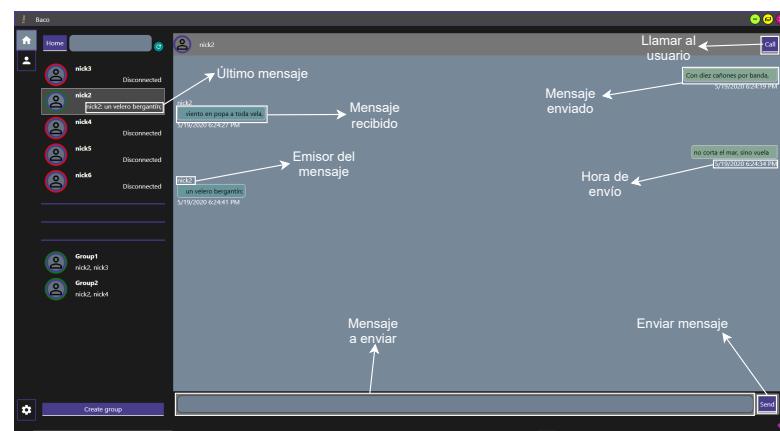
### ÍNDICE GENERAL

#### Chat

La comunicación entre amigos se realiza pulsando el amigo con el que se quiera comunicar. Al pulsar se abrirá la ventana de chat:



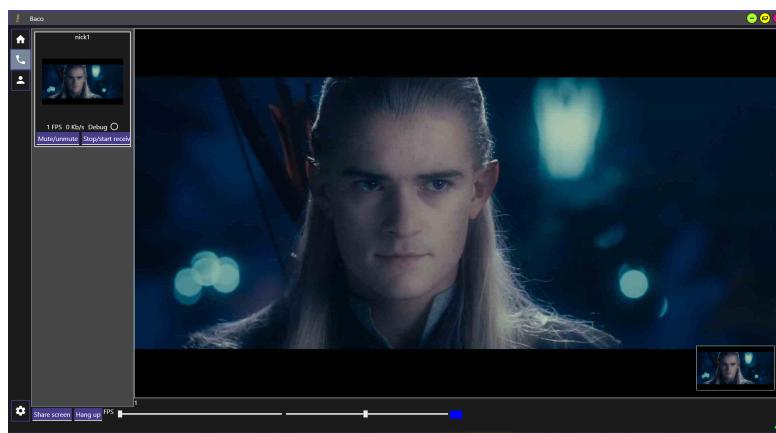
Dentro de esta ventana, sus controles son:



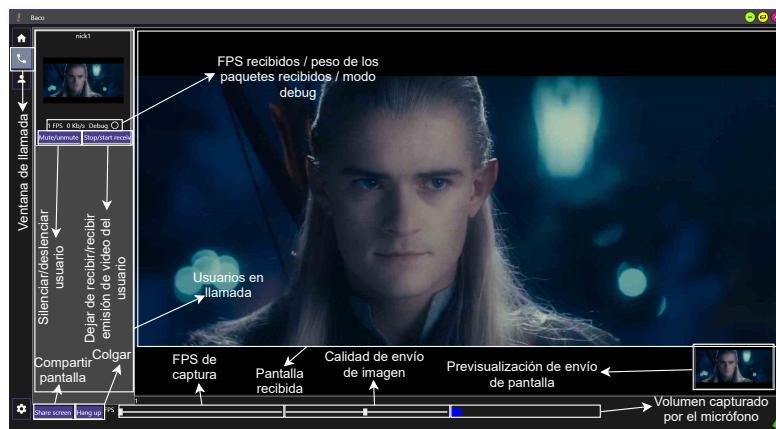
## 6.1. BACO

### Llamada

Cuando se quiere llamar al usuario, si está conectado, le saldrá una ventana emergente informándole que quieres establecer una llamada. Si se acepta se accedería a una nueva ventana (la cual no está disponible en el menú lateral):



Los controles dentro de la llamada son los que siguen:

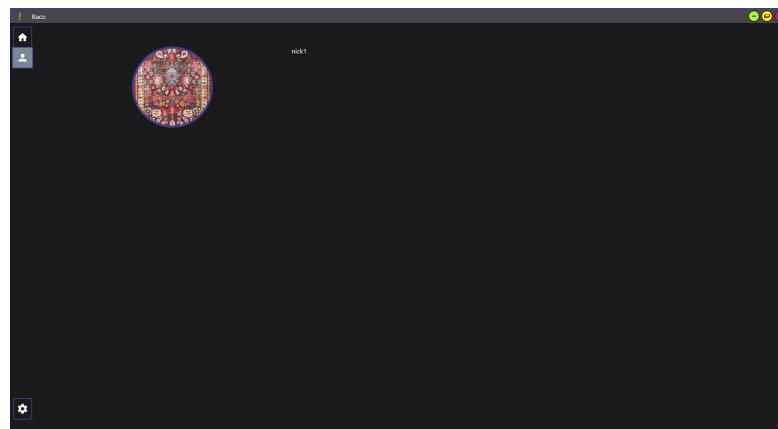


## CAPÍTULO 6. MANUALES DE USUARIO

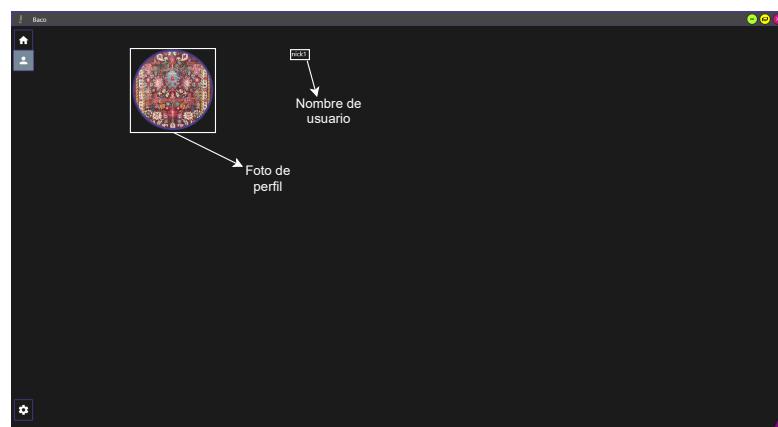
### ÍNDICE GENERAL

#### Perfil

Si entramos a la ventana de perfil veremos nuestra foto de perfil y nuestro nombre de usuario:



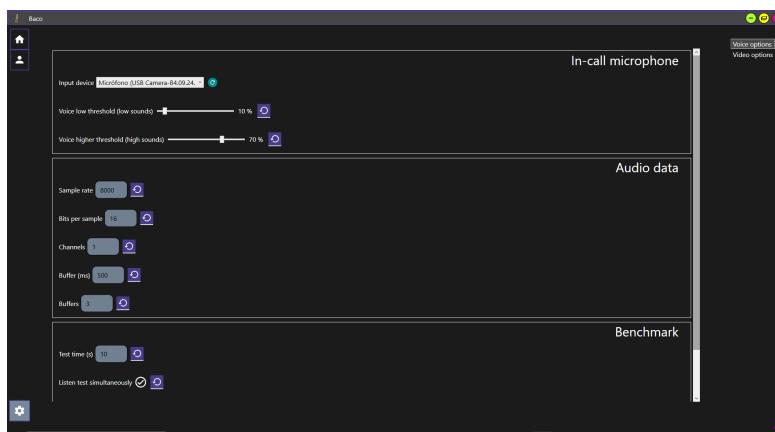
En esta ventana podremos ver nuestro nombre de perfil y cambiar la foto presionándola:



## 6.1. BACO

### Opciones

Y la última ventana, las opciones, donde se pueden cambiar las configuraciones disponibles:



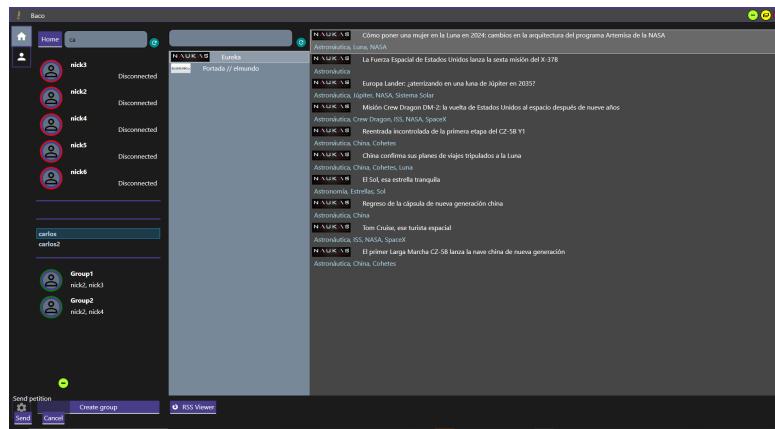
### Amigos

Las peticiones de amistad se llevan a cabo de la siguiente forma:

1. En el campo de búsqueda de amigos se introduce el nombre de usuario del amigo
2. Conforme se vaya escribiendo se irán mostrando los usuarios encontrados que contengan la cadena que se ha escrito:

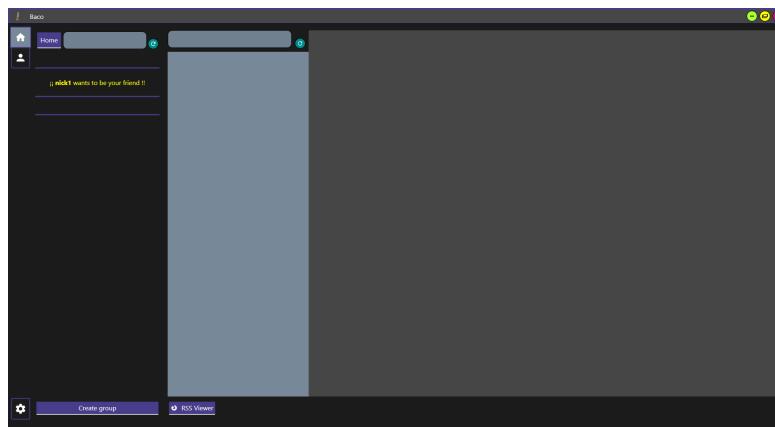
## CAPÍTULO 6. MANUALES DE USUARIO

### ÍNDICE GENERAL



3. Una vez se encuentre al amigo se presiona sobre su nombre de usuario
4. En la parte inferior izquierda aparecerá un diálogo para la enviar la petición
5. Seleccionamos la acción positiva y ya le llegará una notificación

Las peticiones de amistad se reciben en la pantalla principal de la siguiente forma:



## *6.2. BACOSERVER*

### **6.2. BacoServer**

Anexado en esta sección se encuentra el manual de usuario de BacoServer. Véase que la sintaxis empleada en él no es la misma que en el manual de usuario de Baco, sino más técnica.

## *CAPÍTULO 6. MANUALES DE USUARIO*

Esta página ha sido dejada en blanco intencionadamente.

## *6.2. BACOSERVER*

Manual de usuario: BacoServer y API REST

Baco SL

26 de mayo de 2020

*CAPÍTULO 6. MANUALES DE USUARIO*

Esta página ha sido dejada en blanco intencionadamente.

# Índice general

<b>Puesta en marcha</b>	<b>5</b>
Localización del ejecutable (API REST) . . . . .	5
Localización del ejecutable (BacoServer) . . . . .	5
Configuración del adaptador . . . . .	5
Primera ejecución . . . . .	6
Dirección del API REST . . . . .	6
Interfaz . . . . .	8
Resaltado de mensajes . . . . .	8
Información . . . . .	8
Aviso ( <i>warning</i> ) . . . . .	9
Error . . . . .	9
Error crítico . . . . .	9
Comandos . . . . .	10

*CAPÍTULO 6. MANUALES DE USUARIO*

Esta página ha sido dejada en blanco intencionadamente.

## **Puesta en marcha**

BacoServer requiere de una configuración previa a su puesta en marcha. Dado que su función es la de hospedar a los clientes, controlar los mensajes, las llamadas y las conexiones al API REST, BacoServer tiene que tener un puerto accesible y conocido por sus usuarios.

### **Localización del ejecutable (API REST)**

Para empezar tenemos el fichero «API REST» localizado en la carpeta comprimida con diversas librerías. Nuestro ejecutable es «ApiBaco.exe». Tenemos también una carpeta llamada «Images»; en ella se guardarán las imágenes de perfil de los usuarios con una estructura de nombre de usuario + extensión, por ejemplo: un usuario se llama «nick1»; si este usuario tiene foto de perfil con extensión \*.png se guardará como «nick1.png». Si no tuviera foto de perfil se cogería la imagen por defecto: «default.png». Cambiándose «default.png» por otra imagen, esta imagen será la predeterminada para aquellos usuarios sin imagen de perfil.

### **Localización del ejecutable (BacoServer)**

El instalador BacoInstaller nos instalará por defecto en archivos de programa, dentro de la carpeta «Baco SL», un ejecutable del servidor junto con una copia del cliente. El ejecutable se denomina «BacoServer.exe» y es una aplicación con interfaz CLI; esta se explicará más adelante.

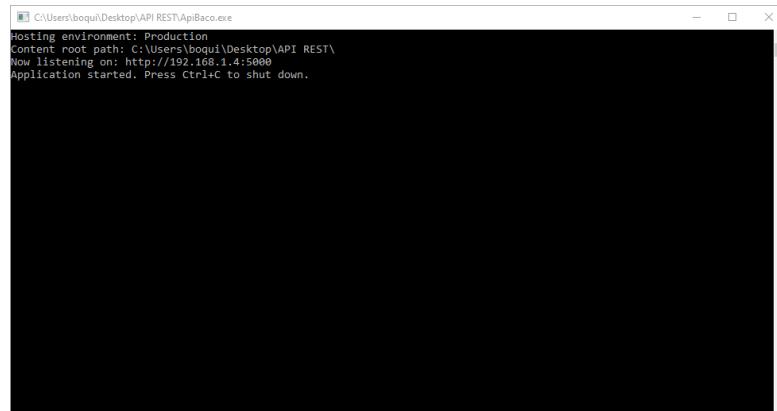
### **Configuración del adaptador**

BacoServer se comunica con protocolo TCP y el puerto que utiliza es el 7854. El API REST utiliza el 5000 y su protocolo es TCP también. Sabidos estos datos puede optarse por una red privada para la comunicación interna en la empresa. Al cambiarse la dirección del servidor y/o del API REST, los

## CAPÍTULO 6. MANUALES DE USUARIO

### ÍNDICE GENERAL

clientes han de cambiar también la dirección a la que se conectan. Véase esto en el manual de usuario de Baco. La dirección del API REST nos la da el mismo cuando se inicia:



```
C:\Users\boqui\Desktop\API REST\ApiBaco.exe
Hosting environment: Production
Content root path: C:\Users\boqui\Desktop\API REST\
Now listening on: http://192.168.1.4:5000
Application started. Press Ctrl+C to shut down.
```

En la tercera línea se ve dónde está escuchando las peticiones:

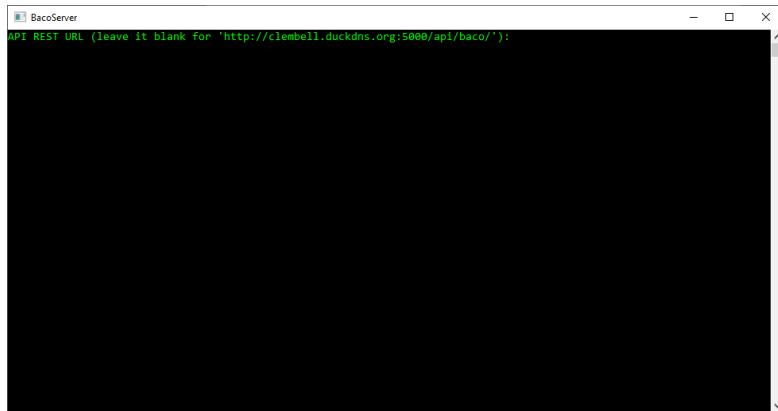
<http://192.168.1.4:5000>

### Primera ejecución

#### Dirección del API REST

Debido a que puede tenerse un API REST propio ajeno al predeterminado, al inicio del servidor se ofrece la opción de especificar la dirección del API REST. Si se deja en blanco se pondrá la dirección por defecto, si se introduce otra se comprobará la validez de esta haciendo una petición al API REST. Si la petición no es posible se volverá a pedir la dirección del API REST junto con un mensaje de error.

## 6.2. BACOSERVER



En nuestro caso no se ha introducido ninguna dirección, por lo tanto nos estaremos conectando al API REST de la dirección:

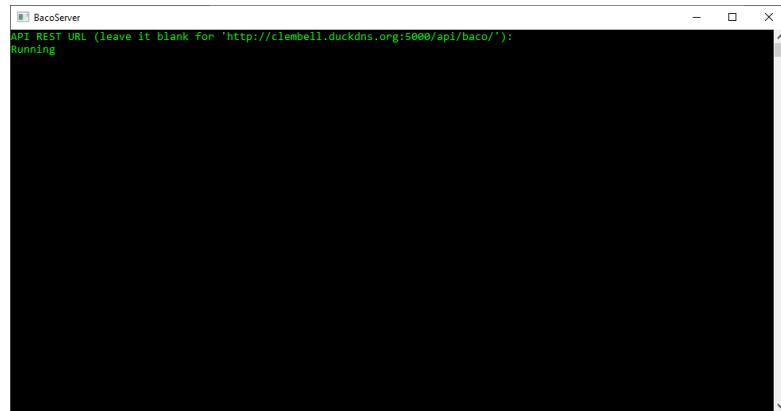
<http://clembell.duckdns.org:5000/api/baco/>

Accediendo a el enlace deberían verse valores de prueba para confirmar la comunicación satisfactoria.

Cuando se realicen las comprobaciones con respecto al API REST y hayan sido satisfactorias aparecerá el mensaje «*running*», el cual será indicativo de que el servidor está operativo.

## CAPÍTULO 6. MANUALES DE USUARIO

### ÍNDICE GENERAL



## Interfaz

Al ser CLI la interfaz no va más allá del resaltado del texto y el uso de comandos. Se van a explicar los colores que se van a ver en la aplicación y los comandos que se pueden usar en ella.

### Resaltado de mensajes

Será la principal interfaz a la que nos enfrentaremos; tenemos tres tipos de colores, el verde, el amarillo, el rojo y el blanco. A continuación se verá para qué son usados cada uno.

Junto con el resaltado de los avisos y los errores, va un contador cuya función es registrar el número de avisos o errores por segundo; esto es a modo informativo y se explicará su porqué más adelante.

### Información

El flujo normal del servidor será resaltado de color verde. Entiéndase como flujo normal todos aquellos mensajes que vayan orientados a la visualización

## 6.2. BACOSERVER

de la acción que esté haciendo el servidor en cada momento.

### Aviso

Los avisos los entendemos como flujos de programa que no suponen la ruptura de este pero que podrían deberse a factores no deseados, como por ejemplo un mal funcionamiento por parte del cliente. Serán resaltados de color amarillo. Supongamos que un cliente tiene un error y no deja de enviar peticiones erróneas al servidor. Dichas peticiones no suponen la ruptura de este, tan solo son desechadas junto con el mensaje en pantalla del error informativo (no olvidemos que los avisos vienen dados por errores controlados). Si las peticiones no cesaran y fueran repetitivas, el contador de avisos y errores por segundo nos informaría de cuántos avisos por segundo están sucediendo, y ello podría suponer un error.

En líneas generales y como conclusión: un aviso no es un error del funcionamiento normal del servidor, pero muchos avisos por segundo sí que puede suponerlo; las causas pueden ser variadas, pero las más comunes son la falta de memoria o un manejo excesivo de peticiones.

### Error

Con diferencia de los avisos, estos suponen errores no deseados y, por lo general, producidos por el servidor. Pueden no suponer un error crítico pero, no obstante, son indicativos de que el servidor no está funcionando correctamente. El mensaje que se mostrará por pantalla será el del error.

### Error crítico

Supone la ruptura total del programa. Se resalta con el color blanco y es indicativo de un error no controlado.

## CAPÍTULO 6. MANUALES DE USUARIO

### ÍNDICE GENERAL

#### Comandos

Para poder manejar el servidor existen una serie de comandos que nos facilitan tareas informativas. Dichos comandos son los que siguen:

- **help**: muestra todos los comandos junto con una breve descripción.
- **status**: muestra los usuarios conectados junto con su tiempo de respuesta. El tiempo de respuesta viene dado por un ping que se realiza en el momento de la ejecución del comando.
- **cls**: borra todos los datos mostrados por pantalla.
- **restart**: expulsa del servidor a todos los usuarios y reinicia el servicio de escucha.

# Capítulo 7

## Ficheros entregados

En los ficheros que se hace entrega se encuentra:

- Una solución de Visual Studio con todos los fuentes necesarios para compilar la API, el cliente y el servidor.
- Las instrucciones necesarias en T-SQL para la creación de las tablas en SSMS (incluidas en la subsección 5.4.4).
- Un instalador (wizard) de la aplicación y del servidor.
- Un fichero comprimido con los ficheros necesarios para la ejecución del API REST. Dado que es una aplicación .NET Core 3.1 el instalador daría como resultado una biblioteca de clases; esto es debido a que puede ser ejecutado desde cualquier plataforma, no solo Windows. Por ello se han incluido los resultados de la compilación en el fichero comprimido en ZIP (dentro de la carpeta de los instaladores).
- Manuales de usuario (incluido en el capítulo 6).
- Memoria del trabajo (este documento).

## *CAPÍTULO 7. FICHEROS ENTREGADOS*

Esta página ha sido dejada en blanco intencionadamente.

# Capítulo 8

## Instalación y requisitos

### 8.1. Instalación

En esencia, los proyectos entregados son tres: el cliente, el servidor y el API REST. En los tres siguientes puntos se explicará cómo se instalan o ejecutan. Los instaladores son los de la aplicación de cliente (subsección 8.1.1) y el servidor (subsección 8.1.2), y el ejecutable viene en la subsección 8.1.4 por los motivos explicados en el punto cuatro del capítulo 7. Los instaladores, por defecto, crearán accesos directos en el escritorio del usuario.

#### 8.1.1. Instalación de Baco

La ruta por defecto será en los archivos de programa y en la carpeta Baco SL. Dentro se generará una carpeta con el nombre de Baco y en su interior se encontrarán todos los ficheros necesarios para el funcionamiento de la aplicación del cliente.

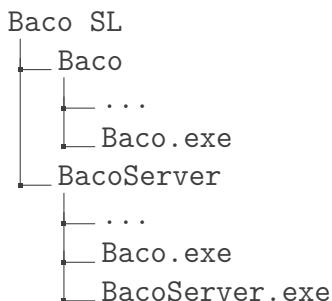
## CAPÍTULO 8. INSTALACIÓN Y REQUISITOS

### 8.1.2. Instalación del servidor de Baco

El servidor posee el mismo instalador que la aplicación del cliente y se instala en la misma localización, con la diferencia de que la carpeta que se crea se llamará BacoServer. En su interior podremos apreciar que se incluye la aplicación de cliente. Esto es debido a que para que el servidor funcione requiere de características del cliente, pero el cliente no requiere de características del servidor. En cierta manera, instalando el servidor se instala en cliente. También apréciase que lo usual no será que el usuario instale el servidor en su ordenador, si cabe quizás una empresa pueda requerir de un servidor privado.

### 8.1.3. Árbol de directorios de la carpeta «Baco SL»

Dentro de la carpeta donde, por defecto, se alojan los programas, encontramos la siguiente estructura de carpetas:



Donde los ficheros \*.exe son los que podremos encontrar como accesos directos en el escritorio.

### 8.1.4. Puesta en marcha del API REST

Se ha entregado un fichero comprimido dentro de la carpeta de instaladores que contiene los ficheros fuente del API REST. Si se descomprime se verá una carpeta denominada «API REST» y dentro se podrá encontrar el ejecutable del API REST. Dicho ejecutable se denomina «ApiBaco.exe».

## 8.2. REQUISITOS

### 8.2. Requisitos

Los requisitos no difieren de los de cualquier otra aplicación; a continuación se enumerarán y explicará el por qué de ellos:

- Conexión a Internet: dado que los datos están almacenados en Azure, para que el API REST acceda a ellos debe de tener conexión a Internet.
- Tarjeta gráfica de gama baja: no se requiere mucha potencia gráfica, pero para la captura de pantalla y el renderizado de vídeo se requiere de computación gráfica. En los casos de pruebas con una Gigabyte GeForce GTX 550 Ti 1GB GDDR5 los clientes no superaban el 5 % de uso de la GPU al enviar y recibir vídeo a la vez.
- 2GB de RAM: si bien es cierto que hoy en día lo normal es encontrar memorias de 4GB para arriba, Baco no requiere de gran uso de memoria, menos de 500MB y suponiendo que el programa está al 100 % de sus funcionalidades.
- Procesador de gama baja: dado que mucha de la carga potencial de procesamiento de datos se la lleva la GPU, la CPU no recibe una carga muy grande. Durante las pruebas en un Intel Core i5-3570 de 3.80 GHz no superaba el 10 % de CPU al 100 % de sus funcionalidades.
- Espacio disponible de 20MB: puesto que es un programa muy simple requiere de una capacidad muy reducida.

## *CAPÍTULO 8. INSTALACIÓN Y REQUISITOS*

Esta página ha sido dejada en blanco intencionadamente.

# Capítulo 9

## Conclusiones

Pese al poco tiempo que se han hecho prácticas presenciales, gracias a ello se ha podido realizar el trabajo en SSML con T-SQL e implementarlo en Visual Studio para realizar el API REST en C#. Durante el tiempo que se estuvo en la empresa se enseñó también un control más amplio de LINQ y que, sin duda, queda plasmado en el proyecto.

### 9.1. Conclusiones sobre el trabajo realizado

Se seguiría ampliando y añadiendo funcionalidades, todo cuanto hay en la aplicación ha hecho que se comprendan a fondo multitud de aspectos de programación: manejo de grandes cantidades hilos, la importancia de un código óptimo, la transmisión de datos por Internet a bajo nivel, importancia y control de eventos, y un largo etcétera.

### 9.2. Posibles ampliaciones y mejoras

La mentalidad de Baco es el mantenimiento por parte de su comunidad y el canal oficial, como bien se expresó en la sección 1.1. De todos modos, se tienen prestaciones pensadas para añadir a la aplicación (orden de prioridad):

## CAPÍTULO 9. CONCLUSIONES

- Control de los periféricos (teclado y ratón) de usuarios en llamada.
- Confirmación de creación de cuenta con el correo electrónico.
- Un apartado de opciones más amplio.
- Mejora de envío de mensajes de texto: la adición de Markdown daría lugar a un texto más enriquecido.
- Optimización de la recepción de video: el vídeo se envía con una tasa de fotogramas alta, pero queda limitada por la velocidad de renderizado del receptor. Una mejor transmisión de los paquetes de vídeo facilitaría la tasa de fotogramas por segundo.
- Transmisión de vídeo grabado por la cámara de web.
- Una correcta separación de clases: si se separan las clases que se encargan de modificar los aspectos visuales de la aplicación y se encapsulan en una librería las clases «cerebro» (aquellas que realizan las tareas imprescindibles de la aplicación, como el envío y la recepción de datos, el manejo de estos...) se podría, fácilmente, exportar Baco a plataformas móviles (Xamarin) y entornos web (ASP.NET): tan solo habría que diseñar su GUI.

# Listado de Códigos

5.1.	Inicio de grabación de pantalla . . . . .	44
5.2.	Grabación de pantalla y envío de vídeo . . . . .	45
5.3.	Inicio de captura de micrófono . . . . .	50
5.4.	Búfer de lectura lleno . . . . .	51
5.5.	Script T-SQL de creación de la base de datos de Baco . . . . .	54

## *LISTA DE CÓDIGOS*

Esta página ha sido dejada en blanco intencionadamente.

# Bibliografía

[Learn Entity Framework Core, 2020] Learn Entity Framework Core (2020).

Disponible en línea: <https://www.learnentityframeworkcore.com/>. Accedido: febrero 2020. Citada en la página 43.

[Ángel Belzunegui Eraso, 2008] Ángel Belzunegui Eraso (2008). Teletrabajo en España, acuerdo marco y administración pública. *Universidad Rovira i Virgili*. Citada en la página 20.

[Prueba piloto de teletrabajo para el colectivo «dentro de convenio», 2005]

Telefónica de España, S.A.U. (2005). *Acuerdo desarrollo cláusula 12.1 convenio colectivo 2003/2005*. Citada en la página 20.

## *BIBLIOGRAFÍA*

Esta página ha sido dejada en blanco intencionadamente.

