Modelado y Programación
Proyecto 1

César Hernández Cruz

Resumen

El Proyecto 1 de la clase Modelado y Programación consiste en un chat, implementado como una aplicación de escritorio. Este chat consta de dos partes, un servidor, y un cliente, que se comunican mediante un protocolo de red determinado en el curso.

El objetivo de este documento es describir mi implementación de dicha aplicación.

1. El proyecto

La aplicación a desarrollarse consta de dos partes, un servidor y un cliente. El servidor recibe conexiones de red de de diversos clientes y atiende las solicitude que estos realizan, e.g., identificarse, enviar mensajes a una sala de chat general, crear salas de chat, invitar a otros usuarios a dichas salas, enviar mensajes directos a otros usuarios, cambiar de estado, solicitar la lista de usuarios conectados, etc. Existe un protocolo previamente determinado para que el cliente pueda comunicarse con el servidor, y a su vez, pueda procesar los mensajes que recibe, ya sea de otros usuarios, o en respuesta a sus solicitudes.

El servidor está implementado como una aplicación que se ejecuta en la terminal, mientras que el cliente es una aplicación de escritorio con una interfaz gráfica.

2. Lenguaje de implementación

El lenguaje que elegí para implementar mi proyecto es Go. Con la restricción de tiempo para aprender un nuevo lenguaje de programación, al mismo tiempo desarrollar por primera vez un proyecto desde cero y hacer por primera vez una interfaz gráfica, me interesaba utilizar un lenguaje que fuera nuevo para mi, y que cumpiera con algunas restricciones: que tuviera manejo automático de memoria, que tuviera una sintaxis sencilla, pero que fuera robusto para llevar a cabo un proyecto mediano. Descarté Python porque, aunque cumple con las dos primeras propiedades, en el pasado he trabajado con él y había notado que al no ser compilado, hay errores que son muy difíciles de detectar (en mi limitada experiencia), lo que lo hace no cumplir con mi tercer requerimiento.

Me encontré con que Go cuenta con una comunidad de usuarios donde es fácil encontrar apoyo para aquellos que empezamos con el lenguaje. Además, tiene una sintaxis amigable, y tiene algunas estructuras interesante, como los canales, diseñados para facilitar la concurrencia. También, sus herramientas para compilar, correr pruebas y ejecutar los programas, me parecieron atractivas para facilitar dichas tareas y poder concentrarme en la implementación. Tras realizar el tour de go, quedé convencido de utilizar este lenguaje.

En retrospectiva, creo que lo que menos me convence del lenguaje son las bibliotecas para las interfaces gráficas. Elegí gotk, un "binding" para gtk, cuya documentación (y ejemplos) deja mucho que desear. Elegí esta biblioteca pues es ampliamente utilizada, y es fácil encontrar ejemplos, que aunque están en otros lenguajes (principalmente C, Python y Ruby), es relativamente fácil traducirlos a Go.

Utilicé dos versiones de Go:

- go version go1.10.3 darwin/amd64 (MacOS)
- go version go1.10.1 linux/amd64 (Ubuntu)

3. Cómo obtener y ejecutar el programa

El código de mi proyecto se encuentra en mi repositorio de github, mismo que puede ser clonado con el comando

```
$ git clone https://github.com/Japodrilo/MyP-Proyecto1.git
```

El lenguaje Go tiene algunas restricciones respecto a la organización de las carpetas, y aunque no todas son estándar, la ruta principal para todos los proyectos con Go debe de ser la misma para que las herramientas de compilación y ejecución funcionen correctamente, esta ruta es conocida como `GOPATH` y está asociada a una variable que especifica el lugar de nuestro "workspace".

En particular, la ruta para este proyecto debe de ser

```
GOPATH/src/github.com/Japodrilo/MyP-Proyecto1/
```

La única dependencia para este proyecto es la biblioteca para interfaces gráficas gotk3, que puede instalarse en Ubuntu con el comando

```
$ go get github.com/gotk3/gotk3/gtk
```

y que a su vez tiene las siguientes dependencias

- GTK 3.6-3.16
- GLib 2.36-2.40
- Cairo 1.10 o 1.12

que pueden instalarse en Ubuntu con el comando

```
$ sudo apt-get install libgtk-3-dev libcairo2-dev libglib2.0-dev
```

En caso de requerir instalarlo en otro sistema operativo, la documentación oficial puede consultarse aquí. La ruta de gotk3 debe de ser

```
GOPATH/src/github.com/gotk3/gotk3
```

y la variable `GOBIN` debe de estar definida para que los archivos binarios se guarden automáticamente en dicha ruta.

Una vez con ambas variables definidas, los archivos binarios pueden generarse con el comando

```
GOPATH/src/github.com/Japodrilo/MyP-Proyecto1/go install ./...
```

mismos que se guardarn en la ruta dada por `GOBIN`. Tambin, el cliente puede correrse con el comando

```
GOPATH/src/github.com/Japodrilo/MyP-Proyecto1/cmd/cliente/go run
                                cliente.go
```

y el servidor con el comando

```
GOPATH/src/github.com/Japodrilo/MyP-Proyecto1/cmd/servidor/go run
                            servidor.go puerto
```

Las clases en el modelo contienen pruebas unitarias que corren con el comando

```
GOPATH/src/github.com/Japodrilo/MyP-Proyecto1/go test ./...
```

sin embargo, todas las pruebas fallan por omisión.

Cabe señalar que hay problemas conocidos con gotk3, y en particular, hubo una computadora con Ubuntu en la que no pude reproducir la instalación, y la interfaz gráfica no funcionó como se esperaba. Algunos problemas comunes se discuten aquí.

4. Interfaz gráfica

La interfaz gráfica es muy sencilla. Cuenta con un menú principal con cuatro submenús. El primero cuenta con las opciones para conectarse y desconectarse. Todas las opciones están apagadas cuando el cliente no está conectado al servidor, excepto “Conectarse” y “Cerrar pestaña actual”. Hay un submenú llamado “Estado”, en el que las opciones simplemente cambian el estado en el que el usuario se encuentra. El submenú pestañas sólo tiene la opción de cerrar la pestaña actual. Finalmente, el menú “Sala” tiene las opciones autodescriptivas “Crear” e “Invitar”, además de la opción “Mis salas”, donde se pueden abrir pestañas para las conversaciones de las salas a las que el usuario pertenece.

5. Problemas conocidos

Tuve que implementar la actualización de la lista de usuarios con "polling", lo que (creo) provoca un problema al desconectar el cliente y volverlo a conectar en la misma ventana. A veces, el cliente indica que el nombre de usuario que se quiere utilizar está ocupado, aunque no sea así.

Es posible cerrar el diálogo con el que el usuario se identifica con el servidor. No es recomendable hacerlo, pero en caso de hacerlo, puede volverse a abrir al volver a introducir los datos de conexión.