



**FACULTAD  
DE INGENIERIA**

---

Universidad de Buenos Aires

75.42 – Taller de Programación I

Grupo 2

2do Cuatrimestre 2016

---

**Self Interpreter**

---

# **Manual de Usuario**



# Índice

<b>Índice</b>	<b>3</b>
<b>Instalación</b>	<b>4</b>
Requerimientos de software	4
Requerimientos de hardware	4
Proceso de Instalación	4
<b>Configuración</b>	<b>5</b>
<b>Forma de uso</b>	<b>6</b>
Servidor	6
Modo servidor	6
Modo ejecución de archivos locales	6
Cliente	7
Modo cliente	7

# Instalación

## Requerimientos de software

Como requisitos de software para poder instalar el programa es necesario contar con:

OS:

- Alguna distribución de linux que soporte GTK+3

Bibliotecas:

- GTK+ 3.22
- gtkmm versión 3.22
- glibmm versión 2.50
- glib versión 2.50.2
- GNU make  $\geq 3.81$
- GNU g++  $\geq 4.8.2$

## Requerimientos de hardware

Como requerimientos mínimos de hardware se consideran los necesarios para poder correr como mínimo una interfaz gráfica que soporte GTK3 y un entorno de red mínimo.

Esta aplicación se desarrolló en un entorno Debian, corriendo sobre una arquitectura Intel x86\_64 con un mínimo de 2GB de memoria RAM. A modo orientativo, Debian requiere un mínimo de velocidad de CPU y memoria RAM, que se detalla a continuación.

Tipo de instalación	RAM (mínimo)	RAM (recomendado)	Disco duro
Sin escritorio	128 Megabytes	512 Megabytes	2 Gigabytes
Con escritorio	256 Megabytes	1 Gigabyte	10 Gigabyte

<sup>1</sup>

## Proceso de Instalación

Para poder instalar el aplicativo es necesario en primer lugar compilar los fuentes. Esto se lleva a cabo a través de la herramienta *make*.

```
$ make -f Makefile_client_server -B
```

Una vez que haya terminado el procedimiento de generación de los ejecutables, se procede a su instalación. Por default ambas aplicaciones se instalan en */usr/local/bin*.

Para poder instalarlas es necesario contar con privilegios, esto es logueándose como administrador o a través del comando *sudo*.

---

<sup>1</sup> <https://www.debian.org/releases/jessie/amd64/ch03s04.html.es>

```
# make -f Makefile_client_server install
```

Para desinstalar el aplicativo, también con usuario con privilegios, escribir:

```
# make -f Makefile_client_server uninstall
```

## Configuración

El aplicativo no requiere archivos de configuración.

## Forma de uso

### Servidor

Una vez instalado el servidor tiene dos formas de uso.

#### Modo servidor

El servidor en este modo actúa como un verdadero servidor. Es decir, una vez iniciado espera conexiones entrantes de clientes y es el encargado de asegurarse de que no haya problemas de concurrencia entre los diferentes clientes al interactuar con los *workspaces*. En el servidor se guarda toda la información de los *workspaces* (ambientes de trabajo) y es el encargado de responder las peticiones de los clientes para interactuar con los mismos.

Se ejecuta con la siguiente línea de comandos:

```
$ server s <puerto>
```

#### Modo ejecución de archivos locales

El servidor en este modo no va a escuchar conexiones entrantes, sino que leerá un archivo local con código self y lo ejecutará.

Se ejecuta con la siguiente línea de comandos:

```
$ server f <nombre de archivo>
```

## Cliente

Una vez instalado, el cliente tiene un único modo de uso.

### Modo cliente

El cliente muestra una interfaz gráfica para permitir interactuar con el servidor, quien es el encargado de manejar todos los workspaces (ambientes de trabajo).

Se ejecuta con la siguiente línea de comandos:

```
$ client <ip servidor> <puerto>
```

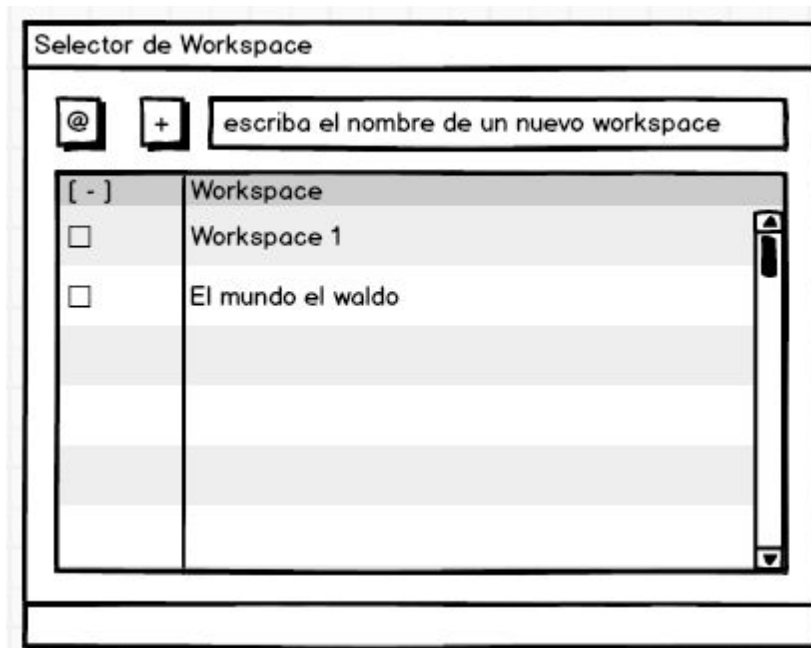
El usuario tiene varias maneras de interactuar con la GUI.

Se realizó un *mockup* con la herramienta Balsamiq para representarlo.

La interfaz tiene dos ventanas principales, el selector de *workspaces* y el visor de *workspace* (también llamada Morph view).

### Selector de workspaces

Cuando iniciamos el programa se va a desplegar esta ventana (la ventana selectora de *workspaces*).

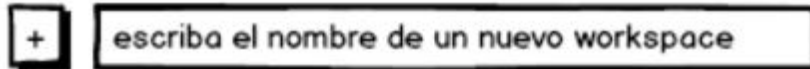


### Botón @ (Refresh)



Este botón permite al usuario hacer un refresh de la lista de workspaces disponibles en el servidor.

### Nuevo workspace



Tenemos una barra de texto en donde podemos ingresar el nombre de un nuevo *workspace*.

Al pulsar el botón [+] a la izquierda de la barra de texto hacemos efectiva la creación del *workspace* y el mismo se agrega a la lista de *workspaces*.

### Lista de workspaces

[ - ]	Workspace
<input type="checkbox"/>	Workspace 1
<input type="checkbox"/>	El mundo el waldo

En el *Treeview* podemos visualizar la lista de *workspaces*.

1er columna:

Tenemos un *checkbox* para eliminar *workspaces*.

2da columna:

Se visualiza el nombre del *Workspace* y si hacemos doble clic en el mismo ingresaremos al *workspace* para interactuar con él.

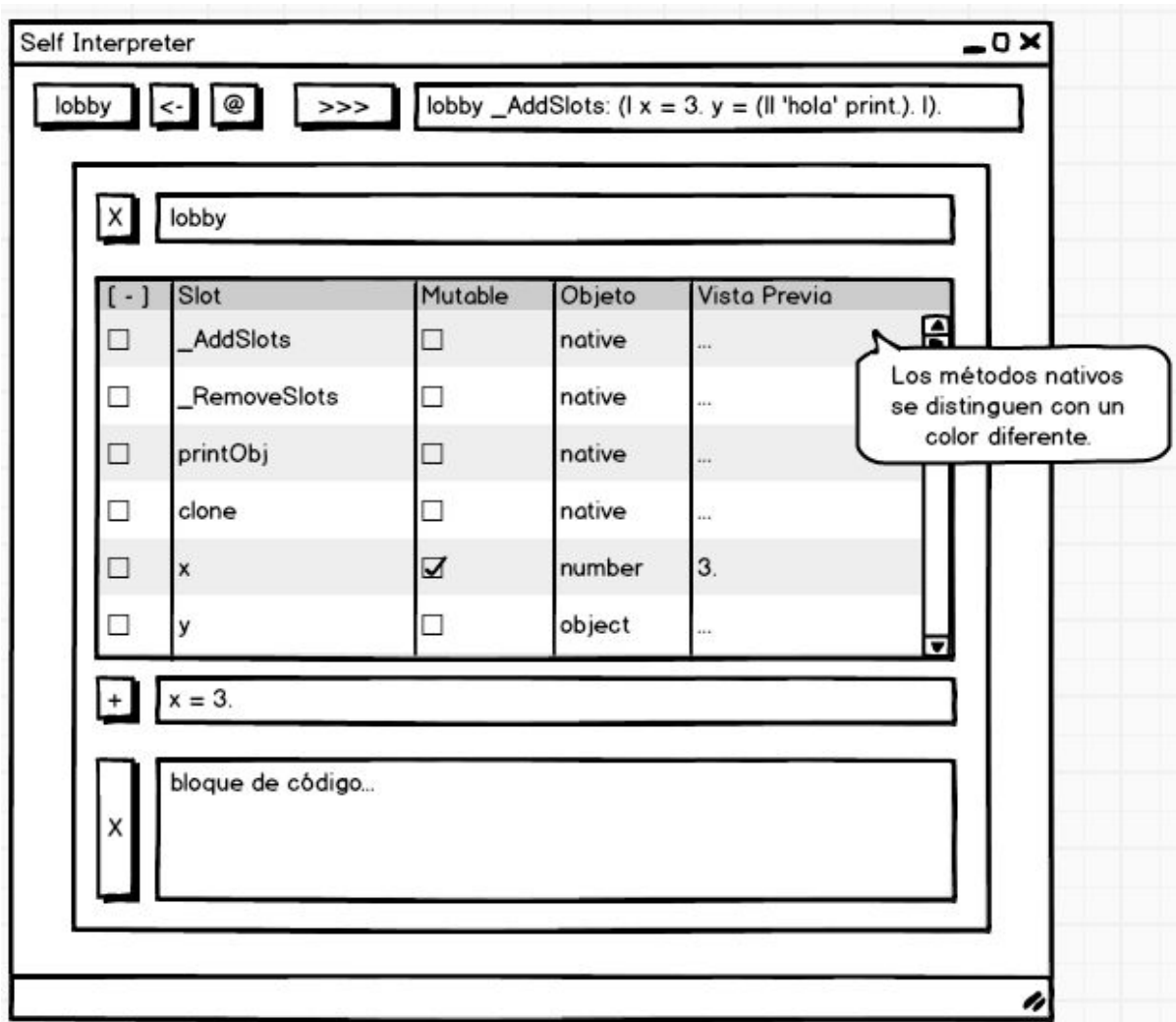


## Visor de workspace / Morph view

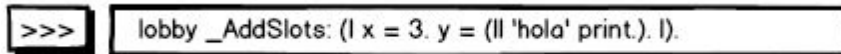
Esta ventana representa el *workspace* seleccionado en la ventana selector de *workspaces*.

Si se quiere salir del *workspace* ingresar al menú desplegable de la barra superior (no se ve en el *mockup*) y seleccionar '**Abandonar Workspace**'.

Si se desea cargar un archivo en la barra de comandos ingresar al menú desplegable de la barra superior (no se ve en el *mockup*) y seleccionar '**Ejecutar código self desde archivo**'.

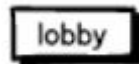


### Barra para ejecutar código self



Tenemos un botón ">>>" que envía al servidor el código self que se encuentra en el *textbox* de su derecha.

### Botón lobby



Este botón nos enviará directamente a la raíz de todos los objetos (el objeto lobby).

### Botón <-



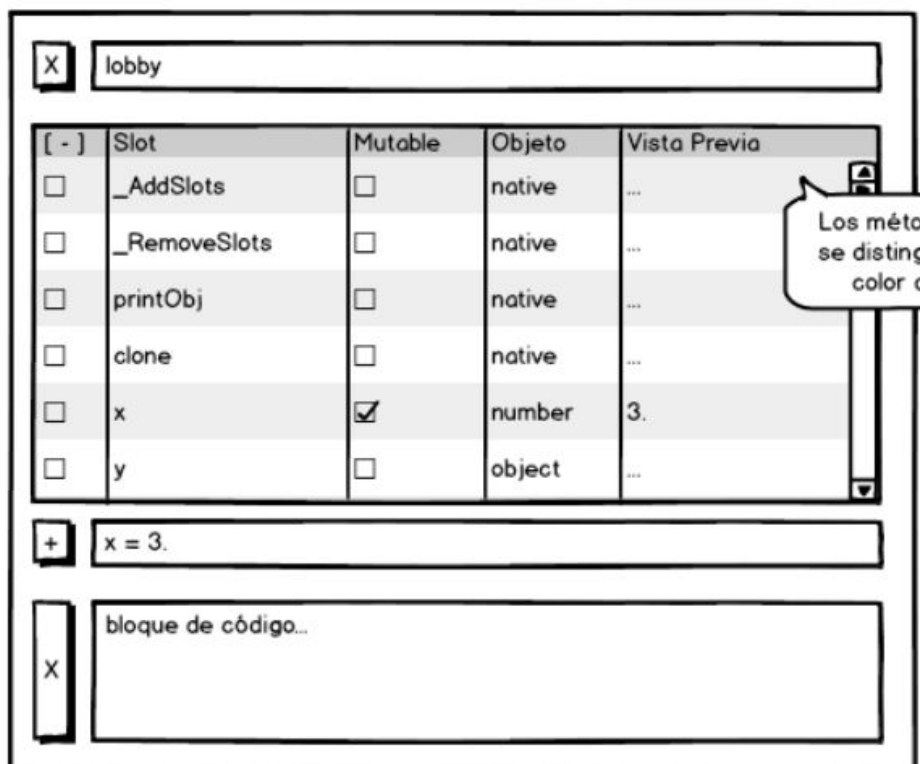
Este botón nos enviará al último objeto visto.

### Botón @ (Refresh)



Este botón permite al usuario hacer una actualización del objeto que está visualizando. Esto le permite tener la versión más reciente del objeto antes de realizar una edición del mismo.

### Morph del Objeto



Representa al objeto que estamos viendo. Recordemos que esta GUI muestra un único objeto en simultáneo.

La vista se compone de 3 partes.

- La sección del seteo del nombre.
- La sección de slots.
- La sección del Bloque de Código.

### Sección nombre del objeto



Tenemos una barra de texto editable en donde se visualizará del nombre del objeto.

En caso de desear cambiar el nombre al objeto, se debe editar la barra de texto y pulsar el botón para guardar el cambio.

### Sección Slots (treeview o datagrid)

[ - ]	Slot	Mutable	Objeto	Vista Previa
<input type="checkbox"/>	_AddSlots	<input type="checkbox"/>	native	...
<input type="checkbox"/>	_RemoveSlots	<input type="checkbox"/>	native	...
<input type="checkbox"/>	printObj	<input type="checkbox"/>	native	...
<input type="checkbox"/>	clone	<input type="checkbox"/>	native	...
<input type="checkbox"/>	x	<input checked="" type="checkbox"/>	number	3.
<input type="checkbox"/>	y	<input type="checkbox"/>	object	...

Los métodos nativos se distinguen con un color diferente.

+ x = 3.

En el *Treeview* podemos visualizar los slots con el respectivo objeto que tengan asociado e interactuar con el mismo.

1er columna:

Tenemos un *checkbox* para eliminar slots.

2da columna:

Se visualiza el nombre del *Slot* en el cuál se verá reflejado si es un slot común, un *argument slot* o un *parent slot*. Ya sea como <nombreSlot>, :<nombreSlot>, <nombreSlot>\* respectivamente.

3ra columna:

Tenemos un *checkbox* para indicar el estado del slot, si es mutable o no.

4ta columna:

Se indica el nombre del objeto.

Los objetos por default se inicializan con un nombre según como fue creado.

Si es un método nativo (por ejemplo el `_AddSlots`) se crea con el nombre **'native'**.

Si es un objeto primitivo se crea con un nombre según el tipo de primitivo.

- Primitivo Null/nil se llama por default **'nil'**
- Primitivo Booleano se llama por default **'boolean'**
- Primitivo Número se llama por default **'number'**
- Primitivo String/cadena de texto se llama por default **'string'**

Si es otro tipo de objeto se llama por default **'string'**.

Todos los objetos pueden ser renombrados (excepto los métodos nativos).

5ta columna:

Se observa una vista previa del objeto.

Si se hace doble clic sobre una fila de un *slot* se accede al objeto que guarda el *slot*.

Se debe hacer doble clic en la fila asociada al slot para traer a pantalla el objeto contenido en el mismo. De esta manera se puede analizar y modificar el objeto contenido a detalle.

Abajo del *treeview* tenemos una barra de texto en donde se debe escribir el *slot* (o los slots ya que soporta multislots) que se desea agregar reutilizando la misma nomenclatura que maneja self. A la izquierda tenemos un botón [ + ] que nos permite generar el/los slot/s.

### Bloque de código



Tenemos una barra de texto editable en donde se visualizará el bloque de código del objeto.

En caso de desear cambiar bloque de código, se debe editar la barra de texto y pulsar el botón para guardar el cambio.