

## Self Interpreter

Generated by Doxygen 1.8.12



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	Acceptor Class Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Constructor & Destructor Documentation . . . . .	6
3.1.2.1	Acceptor() [1/3] . . . . .	6
3.1.2.2	Acceptor() [2/3] . . . . .	6
3.1.2.3	Acceptor() [3/3] . . . . .	6
3.1.3	Member Function Documentation . . . . .	6
3.1.3.1	operator=() [1/2] . . . . .	6
3.1.3.2	operator=() [2/2] . . . . .	6
3.1.3.3	interrupt() . . . . .	7
3.1.3.4	run() . . . . .	7
3.1.3.5	collectClosedClients() . . . . .	7
3.2	ColumnRecord Class Reference . . . . .	7
3.2.1	Detailed Description . . . . .	7
3.3	ColumnRecordWk Class Reference . . . . .	7
3.3.1	Detailed Description . . . . .	8
3.4	Message Class Reference . . . . .	8

3.4.1	Member Function Documentation	8
3.4.1.1	toString()	8
3.4.1.2	getLength()	8
3.4.1.3	setLength()	8
3.4.1.4	getMessage()	9
3.4.1.5	setMessage()	9
3.4.1.6	setCommand()	9
3.4.1.7	getCommand()	9
3.5	ModeSelector Class Reference	9
3.5.1	Detailed Description	10
3.5.2	Constructor & Destructor Documentation	10
3.5.2.1	ModeSelector() [1/4]	10
3.5.2.2	ModeSelector() [2/4]	10
3.5.2.3	ModeSelector() [3/4]	10
3.5.2.4	ModeSelector() [4/4]	10
3.5.3	Member Function Documentation	11
3.5.3.1	operator=() [1/2]	11
3.5.3.2	operator=() [2/2]	11
3.5.3.3	exitRoutine()	11
3.6	Morph Class Reference	11
3.6.1	Detailed Description	12
3.6.2	Member Typedef Documentation	12
3.6.2.1	slot_morph	12
3.6.3	Constructor & Destructor Documentation	12
3.6.3.1	Morph() [1/3]	12
3.6.3.2	Morph() [2/3]	13
3.6.3.3	Morph() [3/3]	13
3.6.4	Member Function Documentation	13
3.6.4.1	operator=() [1/2]	13
3.6.4.2	operator=() [2/2]	13

3.6.4.3	<a href="#">clear()</a>	13
3.6.4.4	<a href="#">getObjName()</a>	13
3.6.4.5	<a href="#">getCodeSegment()</a>	13
3.6.4.6	<a href="#">getSlotsSize()</a>	14
3.6.4.7	<a href="#">getSlotName()</a>	14
3.6.4.8	<a href="#">isNativeMethodSlot()</a>	15
3.6.4.9	<a href="#">isMutableSlot()</a>	15
3.6.4.10	<a href="#">isArgumentSlot()</a>	15
3.6.4.11	<a href="#">isParentSlot()</a>	15
3.6.4.12	<a href="#">getSlotObjName()</a>	16
3.6.4.13	<a href="#">getSlotObjPreview()</a>	16
3.6.4.14	<a href="#">setObjName()</a>	16
3.6.4.15	<a href="#">setCodeSegment()</a>	16
3.6.4.16	<a href="#">addSlot()</a>	17
3.6.4.17	<a href="#">mostrar()</a>	17
3.7	<a href="#">MorphWindow Class Reference</a>	17
3.7.1	<a href="#">Detailed Description</a>	18
3.7.2	<a href="#">Constructor &amp; Destructor Documentation</a>	19
3.7.2.1	<a href="#">MorphWindow()</a>	19
3.7.2.2	<a href="#">~MorphWindow()</a>	19
3.7.3	<a href="#">Member Function Documentation</a>	19
3.7.3.1	<a href="#">addWidgets()</a>	19
3.7.3.2	<a href="#">configureTreeView()</a>	19
3.7.3.3	<a href="#">drawMorph()</a>	19
3.7.3.4	<a href="#">doAction()</a>	20
3.7.3.5	<a href="#">getWindow()</a>	20
3.8	<a href="#">Object Class Reference</a>	20
3.8.1	<a href="#">Detailed Description</a>	21
3.8.2	<a href="#">Member Typedef Documentation</a>	22
3.8.2.1	<a href="#">delegate</a>	22

3.8.2.2	<a href="#">slot_t</a>	22
3.8.2.3	<a href="#">slot_map</a>	22
3.8.2.4	<a href="#">fpointTuple</a>	22
3.8.3	<a href="#">Constructor &amp; Destructor Documentation</a>	22
3.8.3.1	<a href="#">Object()</a>	22
3.8.4	<a href="#">Member Function Documentation</a>	22
3.8.4.1	<a href="#">getParentSlots() [1/2]</a>	22
3.8.4.2	<a href="#">getParentSlots() [2/2]</a>	22
3.8.4.3	<a href="#">findObject()</a>	23
3.8.4.4	<a href="#">configureNativeMethods()</a>	23
3.8.4.5	<a href="#">collect_internal()</a>	23
3.8.4.6	<a href="#">_AddSlots()</a>	23
3.8.4.7	<a href="#">_RemoveSlots()</a>	24
3.8.4.8	<a href="#">clone()</a>	24
3.8.4.9	<a href="#">print()</a>	24
3.8.4.10	<a href="#">printObj()</a>	24
3.8.4.11	<a href="#">operator*()</a>	25
3.8.4.12	<a href="#">operator+()</a>	25
3.8.4.13	<a href="#">operator-()</a>	25
3.8.4.14	<a href="#">operator/()</a>	26
3.8.4.15	<a href="#">operator==(())</a>	26
3.8.4.16	<a href="#">operator!=(())</a>	26
3.8.4.17	<a href="#">operator=() [1/2]</a>	27
3.8.4.18	<a href="#">operator=() [2/2]</a>	27
3.8.4.19	<a href="#">getSlots()</a>	27
3.8.4.20	<a href="#">getNativeMethods()</a>	27
3.8.4.21	<a href="#">addSlot()</a>	27
3.8.4.22	<a href="#">removeSlot()</a>	28
3.8.4.23	<a href="#">setCodeSegment()</a>	28
3.8.4.24	<a href="#">getCodeSegment()</a>	28

3.8.4.25	<a href="#">setName()</a>	28
3.8.4.26	<a href="#">getName()</a>	28
3.8.4.27	<a href="#">isObject()</a> [1/2]	29
3.8.4.28	<a href="#">isObject()</a> [2/2]	29
3.8.4.29	<a href="#">isNativeMethod()</a>	29
3.8.4.30	<a href="#">recvMessage()</a>	30
3.8.4.31	<a href="#">collect()</a>	30
3.8.4.32	<a href="#">enableNativeMethod()</a>	30
3.8.4.33	<a href="#">disableNativeMethod()</a>	31
3.8.4.34	<a href="#">addCreatedObject()</a>	31
3.8.4.35	<a href="#">findObjectById()</a>	31
3.8.4.36	<a href="#">getId()</a>	31
3.8.4.37	<a href="#">swapSlotMutability()</a>	31
3.8.4.38	<a href="#">setPrimitive()</a>	32
3.8.4.39	<a href="#">getPrimitive()</a>	32
3.8.5	<a href="#">Member Data Documentation</a>	32
3.8.5.1	<a href="#">nativeMethods</a>	32
3.9	<a href="#">Parser Class Reference</a>	32
3.9.1	<a href="#">Detailed Description</a>	34
3.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	34
3.9.2.1	<a href="#">Parser()</a> [1/3]	34
3.9.2.2	<a href="#">Parser()</a> [2/3]	35
3.9.2.3	<a href="#">Parser()</a> [3/3]	35
3.9.3	<a href="#">Member Function Documentation</a>	35
3.9.3.1	<a href="#">operator=()</a> [1/2]	35
3.9.3.2	<a href="#">operator=()</a> [2/2]	35
3.9.3.3	<a href="#">parse()</a>	35
3.9.3.4	<a href="#">receiveMessage()</a>	35
3.9.3.5	<a href="#">nilObj()</a>	36
3.9.3.6	<a href="#">boolObj()</a>	36

3.9.3.7	<a href="#">stringObj()</a>	36
3.9.3.8	<a href="#">numberObj()</a>	36
3.9.3.9	<a href="#">objectObj()</a>	36
3.9.3.10	<a href="#">nameObj()</a>	36
3.9.4	<a href="#">Member Data Documentation</a>	37
3.9.4.1	<a href="#">flagExecute</a>	37
3.10	<a href="#">ParserProtocoloMorph Class Reference</a>	37
3.10.1	<a href="#">Detailed Description</a>	37
3.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	37
3.10.2.1	<a href="#">ParserProtocoloMorph() [1/3]</a>	37
3.10.2.2	<a href="#">ParserProtocoloMorph() [2/3]</a>	38
3.10.2.3	<a href="#">ParserProtocoloMorph() [3/3]</a>	38
3.10.3	<a href="#">Member Function Documentation</a>	38
3.10.3.1	<a href="#">operator=() [1/2]</a>	38
3.10.3.2	<a href="#">operator=() [2/2]</a>	38
3.10.3.3	<a href="#">getCampo()</a>	38
3.11	<a href="#">ParserProtocoloServidor Class Reference</a>	38
3.11.1	<a href="#">Detailed Description</a>	39
3.11.2	<a href="#">Constructor &amp; Destructor Documentation</a>	39
3.11.2.1	<a href="#">ParserProtocoloServidor() [1/3]</a>	39
3.11.2.2	<a href="#">ParserProtocoloServidor() [2/3]</a>	39
3.11.2.3	<a href="#">ParserProtocoloServidor() [3/3]</a>	39
3.11.3	<a href="#">Member Function Documentation</a>	39
3.11.3.1	<a href="#">operator=() [1/2]</a>	39
3.11.3.2	<a href="#">operator=() [2/2]</a>	40
3.11.3.3	<a href="#">getString()</a>	40
3.12	<a href="#">ParserProtocoloWorkspaces Class Reference</a>	40
3.12.1	<a href="#">Detailed Description</a>	40
3.12.2	<a href="#">Constructor &amp; Destructor Documentation</a>	40
3.12.2.1	<a href="#">ParserProtocoloWorkspaces() [1/3]</a>	40



3.12.2.2	<a href="#">ParserProtocolWorkspaces()</a> [2/3]	41
3.12.2.3	<a href="#">ParserProtocolWorkspaces()</a> [3/3]	41
3.12.3	Member Function Documentation	41
3.12.3.1	<a href="#">operator=()</a> [1/2]	41
3.12.3.2	<a href="#">operator=()</a> [2/2]	41
3.12.3.3	<a href="#">getCampo()</a>	41
3.13	Proxy Class Reference	41
3.13.1	Detailed Description	42
3.13.2	Constructor & Destructor Documentation	42
3.13.2.1	<a href="#">Proxy()</a> [1/3]	42
3.13.2.2	<a href="#">Proxy()</a> [2/3]	43
3.13.2.3	<a href="#">Proxy()</a> [3/3]	43
3.13.3	Member Function Documentation	43
3.13.3.1	<a href="#">operator=()</a> [1/2]	43
3.13.3.2	<a href="#">operator=()</a> [2/2]	43
3.13.3.3	<a href="#">is_finished()</a>	43
3.13.3.4	<a href="#">interrupt()</a>	43
3.13.3.5	<a href="#">send()</a>	43
3.13.3.6	<a href="#">receive()</a>	44
3.13.3.7	<a href="#">sendError()</a>	44
3.13.3.8	<a href="#">sendOK()</a>	44
3.13.3.9	<a href="#">sendOKWks()</a>	44
3.14	ProxyClient Class Reference	44
3.14.1	Detailed Description	45
3.14.2	Constructor & Destructor Documentation	45
3.14.2.1	<a href="#">ProxyClient()</a> [1/3]	45
3.14.2.2	<a href="#">ProxyClient()</a> [2/3]	46
3.14.2.3	<a href="#">ProxyClient()</a> [3/3]	46
3.14.2.4	<a href="#">~ProxyClient()</a>	46
3.14.3	Member Function Documentation	46

3.14.3.1	<a href="#">operator=()</a> [1/2]	46
3.14.3.2	<a href="#">operator=()</a> [2/2]	46
3.14.3.3	<a href="#">run()</a>	46
3.14.3.4	<a href="#">execLobbyCMD()</a>	46
3.14.3.5	<a href="#">execLocalCMD()</a>	47
3.14.3.6	<a href="#">showLobby()</a>	47
3.14.3.7	<a href="#">execRefresh()</a>	47
3.14.3.8	<a href="#">setObjName()</a>	47
3.14.3.9	<a href="#">setCodeSegment()</a>	47
3.14.3.10	<a href="#">getSlotObj()</a>	48
3.14.3.11	<a href="#">swapMutability()</a>	48
3.14.3.12	<a href="#">goBack()</a>	48
3.14.3.13	<a href="#">availableWks()</a>	48
3.14.3.14	<a href="#">loadWks()</a>	48
3.14.3.15	<a href="#">newWks()</a>	49
3.14.3.16	<a href="#">deleteWks()</a>	49
3.14.3.17	<a href="#">closeWks()</a>	49
3.14.3.18	<a href="#">topObj()</a>	49
3.15	<a href="#">ProxyServer Class Reference</a>	49
3.15.1	<a href="#">Detailed Description</a>	50
3.15.2	<a href="#">Constructor &amp; Destructor Documentation</a>	50
3.15.2.1	<a href="#">ProxyServer()</a>	50
3.15.3	<a href="#">Member Function Documentation</a>	51
3.15.3.1	<a href="#">sendCMDMessage()</a>	51
3.15.3.2	<a href="#">sendCmdMessage()</a>	51
3.15.3.3	<a href="#">run()</a>	51
3.15.3.4	<a href="#">getFlag()</a>	51
3.15.3.5	<a href="#">areThereErrors()</a>	51
3.15.3.6	<a href="#">getErrors()</a>	51
3.15.3.7	<a href="#">setFlag()</a>	51

3.16 SelectWkWindow Class Reference . . . . .	52
3.16.1 Detailed Description . . . . .	52
3.16.2 Constructor & Destructor Documentation . . . . .	53
3.16.2.1 SelectWkWindow() . . . . .	53
3.16.3 Member Function Documentation . . . . .	53
3.16.3.1 addWidgets() . . . . .	53
3.16.3.2 configureTreeView() . . . . .	53
3.16.3.3 drawWorkspaces() . . . . .	53
3.16.3.4 updateList() . . . . .	53
3.16.3.5 getWindow() . . . . .	54
3.17 Server Class Reference . . . . .	54
3.17.1 Detailed Description . . . . .	54
3.17.2 Constructor & Destructor Documentation . . . . .	55
3.17.2.1 Server() . . . . .	55
3.17.2.2 ~Server() . . . . .	55
3.17.3 Member Function Documentation . . . . .	55
3.17.3.1 getWorkspace() . . . . .	55
3.17.3.2 loadWorkspace() . . . . .	55
3.17.3.3 availableWorkspace() . . . . .	55
3.17.3.4 newWorkspace() . . . . .	56
3.17.3.5 closeWorkspace() . . . . .	56
3.17.3.6 deleteWorkspace() . . . . .	56
3.17.3.7 receiveCode() . . . . .	56
3.17.3.8 getLobby() . . . . .	57
3.17.3.9 getObj() . . . . .	57
3.17.3.10 setObjName() . . . . .	57
3.17.3.11 setCodeSegment() . . . . .	57
3.17.3.12 getSlotObj() . . . . .	59
3.17.3.13 swapMutability() . . . . .	59
3.17.4 Member Data Documentation . . . . .	59

3.17.4.1	workspaces	59
3.18	Socket Class Reference	60
3.18.1	Detailed Description	60
3.18.2	Constructor & Destructor Documentation	60
3.18.2.1	Socket() [1/2]	60
3.18.2.2	Socket() [2/2]	61
3.18.3	Member Function Documentation	61
3.18.3.1	send()	61
3.18.3.2	receive()	61
3.19	Thread Class Reference	62
3.19.1	Detailed Description	62
3.20	VirtualMachine Class Reference	62
3.20.1	Detailed Description	63
3.20.2	Constructor & Destructor Documentation	63
3.20.2.1	VirtualMachine()	63
3.20.3	Member Function Documentation	63
3.20.3.1	createNil()	63
3.20.3.2	createString()	63
3.20.3.3	createNumber()	63
3.20.3.4	createBoolean()	64
3.20.3.5	createEmptyObject()	64
3.20.3.6	findObjectById()	64
3.20.3.7	setLobby()	64
3.21	Workspace Class Reference	64
3.21.1	Detailed Description	65
3.21.2	Member Function Documentation	65
3.21.2.1	receive()	65
3.21.2.2	findObjectById()	65

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ColumnRecord	
ColumnRecord . . . . .	7
ColumnRecordWk . . . . .	7
Message . . . . .	8
ModeSelector . . . . .	9
Morph . . . . .	11
Object . . . . .	20
Parser . . . . .	32
ParserProtocoloMorph . . . . .	37
ParserProtocoloServidor . . . . .	38
ParserProtocoloWorkspaces . . . . .	40
Server . . . . .	54
Socket . . . . .	60
Thread . . . . .	62
Acceptor . . . . .	5
Proxy . . . . .	41
ProxyClient . . . . .	44
ProxyServer . . . . .	49
VirtualMachine . . . . .	62
Window	
MorphWindow . . . . .	17
SelectWkWindow . . . . .	52
Workspace . . . . .	64



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Accepter	5
ColumnRecord	7
ColumnRecordWk	7
Message	8
ModeSelector	9
Morph	11
MorphWindow	17
Object	20
Parser	32
ParserProtocolMorph	37
ParserProtocolServidor	38
ParserProtocolWorkspaces	40
Proxy	41
ProxyClient	44
ProxyServer	49
SelectWkWindow	52
Server	54
Socket	60
Thread	62
VirtualMachine	62
Workspace	64





## Chapter 3

# Class Documentation

### 3.1 Acceptor Class Reference

Inherits [Thread](#).

#### Public Member Functions

- [Acceptor](#) (uint32\_t port, [Server](#) &server)  
*Constructor.*
- [Acceptor](#) (const [Acceptor](#) &)=delete
- [Acceptor](#) ([Acceptor](#) &&)=delete
- [Acceptor](#) & operator= (const [Acceptor](#) &)=delete
- [Acceptor](#) & operator= ([Acceptor](#) &&)=delete
- [~Acceptor](#) ()  
*Destructor.*
- void [interrupt](#) ()
- virtual void [run](#) ()

#### Private Member Functions

- void [collectClosedClients](#) ()

#### Private Attributes

- std::vector< [ProxyClient](#) \* > **programThreads**
- [Server](#) & **server**
- [Socket](#) **socket**
- bool **interrupt\_task**

#### 3.1.1 Detailed Description

Es el encargado de aceptar nuevos clientes abriendo proxys en nuevos hilos. Un hilo por cada cliente conectado.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 `Acceptor()` [1/3]

```
Acceptor::Acceptor (
    uint32_t port,
    Server & server )
```

Constructor.

Parameters

<i>port</i>	puerto donde escuchar
<i>server</i>	referencia al <a href="#">Server</a>
<i>workspace</i>	puntero a <a href="#">Workspace</a> .

#### 3.1.2.2 `Acceptor()` [2/3]

```
Acceptor::Acceptor (
    const Acceptor & ) [delete]
```

Constructor por copia deshabilitado

#### 3.1.2.3 `Acceptor()` [3/3]

```
Acceptor::Acceptor (
    Acceptor && ) [delete]
```

Constructor por movimiento deshabilitado

### 3.1.3 Member Function Documentation

#### 3.1.3.1 `operator=()` [1/2]

```
Acceptor& Acceptor::operator= (
    const Acceptor & ) [delete]
```

Operador asignacion deshabilitado

#### 3.1.3.2 `operator=()` [2/2]

```
Acceptor& Acceptor::operator= (
    Acceptor && ) [delete]
```

Operador asignacion por movimiento deshabilitado

### 3.1.3.3 interrupt()

```
void Acceptor::interrupt ( )
```

Metodo que sirve para la interrupcion del proceso.

### 3.1.3.4 run()

```
void Acceptor::run ( ) [virtual]
```

Metodo principal de la clase. Hace los procesamientos

Implements [Thread](#).

### 3.1.3.5 collectClosedClients()

```
void Acceptor::collectClosedClients ( ) [private]
```

Revisa si los clientes que tiene conectados terminaron su ejecucion, luego limpia los recursos utilizados

## 3.2 ColumnRecord Class Reference

Inherits ColumnRecord.

### Public Attributes

- Gtk::TreeModelColumn< bool > **m\_col\_delete**
- Gtk::TreeModelColumn< Glib::ustring > **m\_col\_slotName**
- Gtk::TreeModelColumn< bool > **m\_col\_mutable**
- Gtk::TreeModelColumn< Glib::ustring > **m\_col\_objType**
- Gtk::TreeModelColumn< Glib::ustring > **m\_col\_preview**

### 3.2.1 Detailed Description

Esta clase representa el modelo de columnas que se va a utilizar en el TreeView de los slots.

## 3.3 ColumnRecordWk Class Reference

Inherits ColumnRecord.

### Public Attributes

- Gtk::TreeModelColumn< bool > **m\_col\_delete**
- Gtk::TreeModelColumn< Glib::ustring > **m\_col\_wkName**

### 3.3.1 Detailed Description

Esta clase representa el modelo de columnas que se va a utilizar en el TreeView que enumera los Workspaces.

## 3.4 Message Class Reference

### Public Member Functions

- **Message** (size\_t length, char command, std::string message)
- std::string [toString](#) ()
- size\_t [getLength](#) () const
- void [setLength](#) (const size\_t len)
- std::string [getMessage](#) () const
- void [setMessage](#) (const char \*str)
- void [setCommand](#) (const char cmd)
- char [getCommand](#) () const

### Private Attributes

- size\_t **length**
- char **instr**
- std::string **message**

### 3.4.1 Member Function Documentation

#### 3.4.1.1 toString()

```
std::string Message::toString ( )
```

Este metodo devuelve el objeto convertido a std::string.

#### 3.4.1.2 getLength()

```
size_t Message::getLength ( ) const
```

Devuelve la longitud del mensaje

#### 3.4.1.3 setLength()

```
void Message::setLength (
    const size_t len )
```

Fija la longitud del mensaje

## Parameters

<i>len</i>	nueva longitud
------------	----------------

## 3.4.1.4 getMessage()

```
std::string Message::getMessage ( ) const
```

Devuelve el texto del mensaje-

## 3.4.1.5 setMessage()

```
void Message::setMessage (
    const char * str )
```

Fija el text del mensaje.

## Parameters

<i>str</i>	nuevo mensaje
------------	---------------

## 3.4.1.6 setCommand()

```
void Message::setCommand (
    const char cmd )
```

Fija el comando del mensaje.

## Parameters

<i>cmd</i>	nuevo comando.
------------	----------------

## 3.4.1.7 getCommand()

```
char Message::getCommand ( ) const
```

Devuelve el comando del mensaje.

## 3.5 ModeSelector Class Reference

## Public Member Functions

- [ModeSelector](#) (int port)
- [ModeSelector](#) (std::string filename)
- [ModeSelector](#) (const [ModeSelector](#) &)=delete
- [ModeSelector](#) ([ModeSelector](#) &&)=delete
- [ModeSelector](#) & operator= (const [ModeSelector](#) &)=delete
- [ModeSelector](#) & operator= ([ModeSelector](#) &&)=delete

## Private Member Functions

- void `exitRoutine` (`Acceptor` \*accepter)

### 3.5.1 Detailed Description

Es la clase selectora para los dos modos del servidor. El modo servidor propiamente dicho y el modo para levantar archivos locales con código self.

### 3.5.2 Constructor & Destructor Documentation

#### 3.5.2.1 `ModeSelector()` [1/4]

```
ModeSelector::ModeSelector (
    int port )
```

Constructor del modo server

##### Parameters

<i>port</i>	puerto de escucha del servidor para aceptar conexiones.
-------------	---------------------------------------------------------

#### 3.5.2.2 `ModeSelector()` [2/4]

```
ModeSelector::ModeSelector (
    std::string filename )
```

Constructor del modo archivo

##### Parameters

<i>filename</i>	nombre del archivo con el script en código self que se desea ejecutar.
-----------------	------------------------------------------------------------------------

#### 3.5.2.3 `ModeSelector()` [3/4]

```
ModeSelector::ModeSelector (
    const ModeSelector & ) [delete]
```

Constructor por copia deshabilitado

#### 3.5.2.4 `ModeSelector()` [4/4]

```
ModeSelector::ModeSelector (
    ModeSelector && ) [delete]
```

Constructor por movimiento deshabilitado

### 3.5.3 Member Function Documentation

#### 3.5.3.1 operator=() [1/2]

```
ModeSelector& ModeSelector::operator= (
    const ModeSelector & ) [delete]
```

Operador de asignacion deshabilitado

#### 3.5.3.2 operator=() [2/2]

```
ModeSelector& ModeSelector::operator= (
    ModeSelector && ) [delete]
```

Operador de asignacion por movimiento deshabilitado

#### 3.5.3.3 exitRoutine()

```
void ModeSelector::exitRoutine (
    Acceptor * acceptor ) [private]
```

Metodo que interrumpe al aceptador y joina el hilo que se abrio

##### Parameters

<i>acceptor</i>	Objeto aceptador de conexiones.
-----------------	---------------------------------

## 3.6 Morph Class Reference

### Public Types

- typedef std::tuple< std::string, bool, bool, bool, bool, std::string, std::string > [slot\\_morph](#)

### Public Member Functions

- [Morph](#) ()
- [Morph](#) (const [Morph](#) &)=delete
- [Morph](#) ([Morph](#) &&)=delete
- [Morph](#) & [operator=](#) (const [Morph](#) &)=delete
- [Morph](#) & [operator=](#) ([Morph](#) &&)=delete
- void [clear](#) ()
- std::string [getObjName](#) () const
- std::string [getCodeSegment](#) () const
- int [getSlotsSize](#) () const
- std::string [getSlotName](#) (int nSlot) const
- bool [isNativeMethodSlot](#) (int nSlot) const

- bool [isMutableSlot](#) (int nSlot) const
- bool [isArgumentSlot](#) (int nSlot) const
- bool [isParentSlot](#) (int nSlot) const
- std::string [getSlotObjName](#) (int nSlot) const
- std::string [getSlotObjPreview](#) (int nSlot) const
- void [setObjName](#) (std::string &cad)
- void [setCodeSegment](#) (std::string &cad)
- void [addSlot](#) (std::string &slotName, bool isNativeMethod, bool isMutable, bool isArgument, bool isParent, std::string &objSlotName, std::string &objSlotPreview)
- void [mostrar](#) ()

### Private Attributes

- std::string **objName**
- std::string **codeSegment**
- std::vector< [slot\\_morph](#) > **slots**

### 3.6.1 Detailed Description

Este objeto contiene la información que ve el cliente desde la GUI y representa al objeto que está visualizando.

### 3.6.2 Member Typedef Documentation

#### 3.6.2.1 slot\_morph

```
typedef std::tuple<std::string, bool, bool, bool, bool, std::string, std::string> Morph↔  
::slot_morph
```

Slot del objeto [Morph](#). Elementos de slot\_t pos tipo variable

```
0 std::string nombreSlot  
1 bool esMetodoNativo  
2 bool esMutable  
3 bool esArgument  
4 bool esParent  
5 std::string nombreObjSlot  
6 std::string previewObjSlot
```

### 3.6.3 Constructor & Destructor Documentation

#### 3.6.3.1 Morph() [1/3]

```
Morph::Morph ( )
```

Constructor



### 3.6.3.2 Morph() [2/3]

```
Morph::Morph (
    const Morph & ) [delete]
```

Constructor por copia deshabilitado

### 3.6.3.3 Morph() [3/3]

```
Morph::Morph (
    Morph && ) [delete]
```

Constructor por movimiento deshabilitado

## 3.6.4 Member Function Documentation

### 3.6.4.1 operator=() [1/2]

```
Morph& Morph::operator= (
    const Morph & ) [delete]
```

Operador de asignacion deshabilitado

### 3.6.4.2 operator=() [2/2]

```
Morph& Morph::operator= (
    Morph && ) [delete]
```

Operador de asignacion por movimiento deshabilitado

### 3.6.4.3 clear()

```
void Morph::clear ( )
```

Limpia el contenido del morph

### 3.6.4.4 getObjName()

```
std::string Morph::getObjName ( ) const
```

Retorna el nombre del objeto.

### 3.6.4.5 getCodeSegment()

```
std::string Morph::getCodeSegment ( ) const
```

Retorna el bloque de código del objeto.

#### 3.6.4.6 getSlotsSize()

```
int Morph::getSlotsSize ( ) const
```

Retorna la cantidad de slots del objeto.

#### 3.6.4.7 getSlotName()

```
std::string Morph::getSlotName (
    int nSlot ) const
```

Retorna el nombre del slot indicado.

**Parameters**

<i>nSlot</i>	número de slot del que se solicita información.
--------------	-------------------------------------------------

**3.6.4.8 isNativeMethodSlot()**

```
bool Morph::isNativeMethodSlot (
    int nSlot ) const
```

Retorna si el slot indicado es un método nativo.

**Parameters**

<i>nSlot</i>	número de slot del que se solicita información.
--------------	-------------------------------------------------

**3.6.4.9 isMutableSlot()**

```
bool Morph::isMutableSlot (
    int nSlot ) const
```

Retorna si el slot indicado es mutable.

**Parameters**

<i>nSlot</i>	número de slot del que se solicita información.
--------------	-------------------------------------------------

**3.6.4.10 isArgumentSlot()**

```
bool Morph::isArgumentSlot (
    int nSlot ) const
```

Retorna si el slot indicado es de tipo argumento.

**Parameters**

<i>nSlot</i>	número de slot del que se solicita información.
--------------	-------------------------------------------------

**3.6.4.11 isParentSlot()**

```
bool Morph::isParentSlot (
    int nSlot ) const
```

Retorna si el slot indicado es de tipo parent.

**Parameters**

<i>nSlot</i>	número de slot del que se solicita información.
--------------	-------------------------------------------------

**3.6.4.12 getSlotObjName()**

```
std::string Morph::getSlotObjName (
    int nSlot ) const
```

Retorna el nombre del objeto contenido en el slot indicado.

**Parameters**

<i>nSlot</i>	número de slot del que se solicita información.
--------------	-------------------------------------------------

**3.6.4.13 getSlotObjPreview()**

```
std::string Morph::getSlotObjPreview (
    int nSlot ) const
```

Retorna una cadena con la vista previa del objeto contenido en el slot indicado.

**Parameters**

<i>nSlot</i>	número de slot del que se solicita información.
--------------	-------------------------------------------------

**3.6.4.14 setObjName()**

```
void Morph::setObjName (
    std::string & cad )
```

Setea el nombre del objeto

**Parameters**

<i>cad</i>	nuevo nombre del objeto.
------------	--------------------------

**3.6.4.15 setCodeSegment()**

```
void Morph::setCodeSegment (
    std::string & cad )
```

Setea el bloque de código del objeto

## Parameters

<i>cad</i>	nuevo bloque de código del objeto.
------------	------------------------------------

## 3.6.4.16 addSlot()

```
void Morph::addSlot (
    std::string & slotName,
    bool isNativeMethod,
    bool isMutable,
    bool isArgument,
    bool isParent,
    std::string & objSlotName,
    std::string & objSlotPreview )
```

Agrega un slot al Objeto [Morph](#)

## Parameters

<i>slotName</i>	nombre del slot.
<i>isNativeMethod</i>	indica si es un método nativo.
<i>isMutable</i>	indica si el slot es mutable.
<i>isArgument</i>	indica si el slot es de tipo argumento.
<i>isParent</i>	indica si el slot es de tipo parent.
<i>objSlotName</i>	nombre del objeto contenido en el slot.
<i>objSlotPreview</i>	vista previa del objeto contenido en el slot.

## 3.6.4.17 mostrar()

```
void Morph::mostrar ( )
```

Imprime por pantalla los campos del [Morph](#)

## 3.7 MorphWindow Class Reference

Inherits Window.

## Public Member Functions

- [MorphWindow](#) ([Morph](#) &morph, [ProxyServer](#) &proxyServer, std::mutex &m)
- [~MorphWindow](#) ()
- Gtk::Window \* [getWindow](#) ()
- **MorphWindow** ([MorphWindow](#) &&)=delete
- **MorphWindow** (const [MorphWindow](#) &)=delete
- [MorphWindow](#) & **operator=** ([MorphWindow](#) &&)=delete
- [MorphWindow](#) & **operator=** (const [MorphWindow](#) &)=delete

## Private Member Functions

- void [addWidgets](#) ()
- void [configureTreeView](#) ()
- void [drawMorph](#) ()
- void [doAction](#) (char action, std::string text)
- void [btnLobby\\_clicked](#) ()
- void [btnGoBack\\_clicked](#) ()
- void [btnRefresh\\_clicked](#) ()
- void [btnEnviar\\_clicked](#) ()
- void [btnApply\\_clicked](#) ()
- void [btnSetSlot\\_clicked](#) ()
- void [btnSetCodeSegment\\_clicked](#) ()
- bool [window\\_deleted](#) (GdkEventAny \*any\_event)
- void [on\\_row\\_activated](#) (const Gtk::TreeModel::Path &path, Gtk::TreeViewColumn \*column)
- void [on\\_Open\\_selected](#) ()
- void [on\\_CloseWorkspace\\_selected](#) ()
- void [cellMutable\\_toggled](#) (const Glib::ustring &path)
- void [cellDelete\\_toggled](#) (const Glib::ustring &path)

## Private Attributes

- [Morph](#) & [morph](#)
- [ProxyServer](#) & [proxyServer](#)
- std::mutex & [m](#)
- Glib::RefPtr< Gtk::Builder > [refBuilder](#)
- Gtk::Window \* [pWindow](#) = nullptr
- Gtk::Button \* [pBtnLobby](#) = nullptr
- Gtk::Button \* [pBtnGoBack](#) = nullptr
- Gtk::Button \* [pBtnRefresh](#) = nullptr
- Gtk::Button \* [pBtnEnviar](#) = nullptr
- Gtk::Button \* [pBtnApply](#) = nullptr
- Gtk::Button \* [pBtnSetSlot](#) = nullptr
- Gtk::Button \* [pBtnSetCodeSegment](#) = nullptr
- Gtk::TextView \* [pTxtEntrada](#) = nullptr
- Gtk::TextView \* [pTxtCodeSegment](#) = nullptr
- Gtk::Entry \* [pTxtObjName](#) = nullptr
- Gtk::Entry \* [pTxtSlot](#) = nullptr
- Gtk::TreeView \* [pTreeView](#) = nullptr
- Glib::RefPtr< Gtk::TreeStore > [m\\_refTreeModel](#)
- [ColumnRecord](#) [m\\_Columns](#)
- Gtk::MenuItem \* [pMenuItemOpen](#) = nullptr
- Gtk::MenuItem \* [pMenuItemCloseWorkspace](#) = nullptr

### 3.7.1 Detailed Description

Se encarga de dibujar la ventana que representa al Morphic de Self. Permite visualizar la información de un objeto de por vez.

## 3.7.2 Constructor & Destructor Documentation

### 3.7.2.1 MorphWindow()

```
MorphWindow::MorphWindow (
    Morph & morph,
    ProxyServer & proxyServer,
    std::mutex & m )
```

Constructor de la clase

#### Parameters

<i>morph</i>	referencia al objeto <a href="#">Morph</a> .
<i>proxyServer</i>	referencia al proxy
<i>m</i>	referencia al mutex

### 3.7.2.2 ~MorphWindow()

```
MorphWindow::~MorphWindow ( )
```

Destructor de la clase

## 3.7.3 Member Function Documentation

### 3.7.3.1 addWidgets()

```
void MorphWindow::addWidgets ( ) [private]
```

Este metodo agrega los widgets a la ventana

### 3.7.3.2 configureTreeView()

```
void MorphWindow::configureTreeView ( ) [private]
```

Este metodo configura las columnas los eventos del TreeView.

### 3.7.3.3 drawMorph()

```
void MorphWindow::drawMorph ( ) [private]
```

Este metodo dibuja los datos recibidos del servidor en el TreeView.

### 3.7.3.4 doAction()

```
void MorphWindow::doAction (
    char action,
    std::string text ) [private]
```

Este metodo sirve para enviar las solicitudes al [ProxyServer](#)

### 3.7.3.5 getWindow()

```
Gtk::Window * MorphWindow::getWindow ( )
```

Devuelve un puntero al objeto Gtk::Window.

## 3.8 Object Class Reference

### Public Types

- typedef [Object](#) \*([Object](#)::[delegate](#)) (const std::vector< [Object](#) \*> &)
- typedef std::tuple< [Object](#) \*, bool, bool, bool > [slot\\_t](#)
- typedef std::map< std::string, [slot\\_t](#) > [slot\\_map](#)
- typedef std::tuple< [delegate](#), bool > [fpointTuple](#)

### Public Member Functions

- [Object](#) ()  
*Constructor que sirve para construir el objeto lobby.*
- [Object](#) ([Object](#) \*lobby)  
*Constructor de los objetos que no son Lobby.*
- [~Object](#) ()  
*Destructor.*
- [Object](#) (const [Object](#) &\_\_object)  
*Constructor copia.*
- [Object](#) ([Object](#) &&obj)=delete
- [Object](#) & operator= (const [Object](#) &\_\_object)=delete
- [Object](#) & operator= ([Object](#) &&)=delete
- [slot\\_map](#) getSlots () const
- std::map< std::string, [fpointTuple](#) > [getNativeMethods](#) () const
- [Object](#) \* [addSlot](#) (std::string &name, [Object](#) \*obj, bool \_mutable, bool isParentSlot, bool isArgument)
- [Object](#) \* [removeSlot](#) (std::string &name)
- void [setCodeSegment](#) (const std::string &code)
- std::string [getCodeSegment](#) () const
- void [setName](#) (const std::string &name)
- std::string [getName](#) () const
- bool [isDataObject](#) (std::string &messageName)
- bool [isDataObject](#) ()
- bool [isNativeMethod](#) (std::string &messageName)
- [Object](#) \* [recvMessage](#) (std::string &messageName, std::vector< [Object](#) \*> args, bool clone)
- [Object](#) \* [collect](#) (const std::vector< [Object](#) \*> &args)
- void [enableNativeMethod](#) (std::string methodName)
- void [disableNativeMethod](#) (std::string methodName)
- void [addCreatedObject](#) ([Object](#) \*obj)
- [Object](#) \* [findObjectById](#) (uint32\_t id)
- uint32\_t [getId](#) () const
- void [swapSlotMutability](#) (const std::string &slotName)
- void [setPrimitive](#) (const bool newValue)
- bool [getPrimitive](#) () const



## Private Types

- typedef std::tuple< [Object](#) \*, bool > **tuple\_createdObjects**

## Private Member Functions

- [slot\\_map](#) getParentSlots () const
- [Object::slot\\_map](#) getParentSlots ([Object](#) \*pointer) const
- bool [findObject](#) (std::string name, [Object](#) \*&returnValue, [delegate](#) &function) const
- void [configureNativeMethods](#) ()
- void [collect\\_internal](#) ()
- [Object](#) \* [\\_AddSlots](#) (const std::vector< [Object](#) \*> &args)
- [Object](#) \* [\\_RemoveSlots](#) (const std::vector< [Object](#) \*> &args)
- [Object](#) \* [clone](#) (const std::vector< [Object](#) \*> &args)
- [Object](#) \* [print](#) (const std::vector< [Object](#) \*> &args)
- [Object](#) \* [printObj](#) (const std::vector< [Object](#) \*> &args)
- [Object](#) \* [operator\\*](#) (const std::vector< [Object](#) \*> &args)
- [Object](#) \* [operator+](#) (const std::vector< [Object](#) \*> &args)
- [Object](#) \* [operator-](#) (const std::vector< [Object](#) \*> &args)
- [Object](#) \* [operator/](#) (const std::vector< [Object](#) \*> &args)
- [Object](#) \* [operator==](#) (const std::vector< [Object](#) \*> &args)
- [Object](#) \* [operator!=](#) (const std::vector< [Object](#) \*> &args)

## Private Attributes

- [slot\\_map](#) slots  
*Representa los slots del objeto.*
- std::string name  
*Nombre del objeto.*
- std::string codeSegment  
*Representa el code segment del objeto.*
- std::map< std::string, [fpointTuple](#) > [nativeMethods](#)
- [Object](#) \* [lobby](#)  
*Puntero al objeto lobby.*
- uint32\_t id = 0  
*id numerico (y unico) del objeto.*
- uint32\_t idCounter = 1  
*Contador de objetos.*
- std::map< uint32\_t, tuple\_createdObjects > [createdObjects](#)  
*Diccionario con todos los objetos creados.*
- bool isPrimitive = false  
*Indica si el objeto es de un tipo primitivo.*

### 3.8.1 Detailed Description

Representa un objeto del lenguaje Self.

### 3.8.2 Member Typedef Documentation

#### 3.8.2.1 delegate

```
typedef Object\*(Object::\* Object::delegate) (const std::vector< Object * > &)
```

Definición del tipo delegate que es un puntero a función que tiene la forma `Object* función (const std::vector<↔Object*>&)`

#### 3.8.2.2 slot\_t

```
typedef std::tuple<Object\*, bool, bool, bool> Object::slot\_t
```

Formato que tiene el slot.

puntero a [Object](#) con la referencia al objeto

booleano que indica si es mutable o no

booleano que indica si el objeto apuntado es un parent slot

booleano que indica si está implementado nativamente

booleano que indica si es un argumento (:) )

#### 3.8.2.3 slot\_map

```
typedef std::map<std::string, slot\_t> Object::slot\_map
```

Formato que representa el diccionario interno. Con string como clave que representa el nombre del slot, y `slot_t` que representa el valor.

#### 3.8.2.4 fpointTuple

```
typedef std::tuple<delegate, bool> Object::fpointTuple
```

Formato que representa las funciones nativas que están habilitadas. En la primera parte de la tupla está el puntero a función. En la segunda un booleano que indica si está habilitado o no.

### 3.8.3 Constructor & Destructor Documentation

#### 3.8.3.1 Object()

```
Object::Object (  
    Object && obj ) [delete]
```

Constructor por movimiento deshabilitado

### 3.8.4 Member Function Documentation

#### 3.8.4.1 getParentSlots() [1/2]

```
Object::slot\_map Object::getParentSlots ( ) const [private]
```

Devuelve los parent slots que tiene el objeto.

#### 3.8.4.2 getParentSlots() [2/2]

```
Object::slot\_map Object::getParentSlots (  
    Object * pointer ) const [private]
```

Devuelve los parent slots del objeto pasado como argumento

## Parameters

<i>pointer</i>	objeto sobre el cual devolver los parent slot.
----------------	------------------------------------------------

## 3.8.4.3 findObject()

```
bool Object::findObject (
    std::string name,
    Object *& returnValue,
    delegate & function ) const [private]
```

Este metodo sirve para buscar el objeto de un slot.

## Parameters

<i>name</i>	nombre del slot a buscar
<i>returnValue</i>	puntero a <a href="#">Object</a> devuelto con el puntero al slot.
<i>function</i>	puntero a funcion devuelto si es un metodo nativo.

## Return values

<i>true</i>	si encontro el objeto seteando el puntero.
<i>false, los</i>	punteros estan en nullptr.

## 3.8.4.4 configureNativeMethods()

```
void Object::configureNativeMethods ( ) [private]
```

Este metodo habilita los metodos nativos comunes a todos los objetos. Estos son: `_AddSlots`, `_RemoveSlots`, `clone` y `printObj`.

## 3.8.4.5 collect\_internal()

```
void Object::collect_internal ( ) [private]
```

Recorre todos los slots del objeto y va marcando que es accesible

## 3.8.4.6 \_AddSlots()

```
Object * Object::_AddSlots (
    const std::vector< Object *> & args ) [private]
```

Funcion nativa para agregar un slot.

**Parameters**

<i>args</i>	vector con Object* con los objetos a agregar al slot.
-------------	-------------------------------------------------------

**3.8.4.7 \_RemoveSlots()**

```
Object * Object::_RemoveSlots (
    const std::vector< Object *> & args ) [private]
```

Funcion nativa para borrar un slot.

**Parameters**

<i>args</i>	vector con Object* con los objetos para borrar.
-------------	-------------------------------------------------

**3.8.4.8 clone()**

```
Object * Object::clone (
    const std::vector< Object *> & args ) [private]
```

Metodo nativo para clonar el objeto.

**Parameters**

<i>args</i>	vector de Object* vacio.
-------------	--------------------------

**3.8.4.9 print()**

```
Object * Object::print (
    const std::vector< Object *> & args ) [private]
```

Metodo nativo para imprimir por pantalla.

**Parameters**

<i>args</i>	vector de Object* vacio.
-------------	--------------------------

**3.8.4.10 printObj()**

```
Object * Object::printObj (
    const std::vector< Object *> & args ) [private]
```

Metodo nativo para imprimir por pantalla datos de debugging.

## Parameters

<i>args</i>	vector de Object* vacio.
-------------	--------------------------

## 3.8.4.11 operator\*()

```
Object * Object::operator* (
    const std::vector< Object *> & args ) [private]
```

Metodo nativo para efectuar la multiplicacion

## Parameters

<i>args</i>	vector de Object* vacio.
-------------	--------------------------

## 3.8.4.12 operator+()

```
Object * Object::operator+ (
    const std::vector< Object *> & args ) [private]
```

Metodo nativo para efectuar la suma

## Parameters

<i>args</i>	vector de Object* vacio.
-------------	--------------------------

## Return values

<i>Object</i>	con el resultado de la operacion. Se opera sobre el primer operando, es decir, sobre el llamante del metodo.
---------------	--------------------------------------------------------------------------------------------------------------

## 3.8.4.13 operator-()

```
Object * Object::operator- (
    const std::vector< Object *> & args ) [private]
```

Metodo nativo para efectuar la resta

## Parameters

<i>args</i>	vector de Object* vacio.
-------------	--------------------------

## Return values

<i>Object</i>	con el resultado de la operacion. Se opera sobre el primer operando, es decir, sobre el llamante del metodo.
---------------	--------------------------------------------------------------------------------------------------------------

#### 3.8.4.14 operator/()

```
Object * Object::operator/ (
    const std::vector< Object *> & args ) [private]
```

Metodo nativo para efectuar la division

##### Parameters

<i>args</i>	vector de Object* vacio.
-------------	--------------------------

##### Return values

<i>Object</i>	con el resultado de la operacion. Se opera sobre el primer operando, es decir, sobre el llamante del metodo.
---------------	--------------------------------------------------------------------------------------------------------------

#### 3.8.4.15 operator==( )

```
Object * Object::operator==(
    const std::vector< Object *> & args ) [private]
```

Metodo nativo para chequear igualdad

##### Parameters

<i>args</i>	vector de Object* vacio.
-------------	--------------------------

##### Return values

<i>Object</i>	con el resultado de la operacion. Se opera sobre el primer operando, es decir, sobre el llamante del metodo.
---------------	--------------------------------------------------------------------------------------------------------------

#### 3.8.4.16 operator!=( )

```
Object * Object::operator!=(
    const std::vector< Object *> & args ) [private]
```

Metodo nativo para chequear desigualdad

##### Parameters

<i>args</i>	vector de Object* vacio.
-------------	--------------------------

##### Return values

<i>Object</i>	con el resultado de la operacion. Se opera sobre el primer operando, es decir, sobre el llamante del metodo.
---------------	--------------------------------------------------------------------------------------------------------------

**3.8.4.17 operator=()** [1/2]

```
Object& Object::operator= (
    const Object & _object ) [delete]
```

Operador de asignacion deshabilitado

**3.8.4.18 operator=()** [2/2]

```
Object& Object::operator= (
    Object && ) [delete]
```

Operador de asignacion por movimiento deshabilitado

**3.8.4.19 getSlots()**

```
Object::slot_map Object::getSlots ( ) const
```

Devuelve todos los slots que tiene el objeto en forma de diccionario.

**3.8.4.20 getNativeMethods()**

```
std::map< std::string, Object::fpointTuple > Object::getNativeMethods ( ) const
```

Devuelve todo los metodos nativos que tiene el objeto en forma de diccionario.

**3.8.4.21 addSlot()**

```
Object * Object::addSlot (
    std::string & name,
    Object * obj,
    bool _mutable,
    bool isParentSlot,
    bool isArgument )
```

Agrega el slot con los datos pasados como parametros.

**Parameters**

<i>name</i>	nombre del slot.
<i>obj</i>	puntero a Object* para guardar en el slot.
<i>_mutable</i>	booleano que indica si es mutable o no.
<i>isParentSlot</i>	booleano que indica si es parent slot o no.
<i>isArgument</i>	booleano que indica si es argumento o no.

#### 3.8.4.22 removeSlot()

```
Object * Object::removeSlot (
    std::string & name )
```

Borra el slot especificado

##### Parameters

<i>name</i>	indica el nombre del slot a borrar.
-------------	-------------------------------------

#### 3.8.4.23 setCodeSegment()

```
void Object::setCodeSegment (
    const std::string & code )
```

Agrega como code segment del objeto la cadena que se la pasa

##### Parameters

<i>code</i>	nuevo codigo para reemplazar en el code segment.
-------------	--------------------------------------------------

#### 3.8.4.24 getCodeSegment()

```
std::string Object::getCodeSegment ( ) const
```

Devuelve el code segment del objeto.

#### 3.8.4.25 setName()

```
void Object::setName (
    const std::string & name )
```

Setea el nombre del objeto

##### Parameters

<i>name</i>	con el nombre del objeto para cambiar.
-------------	----------------------------------------

#### 3.8.4.26 getName()

```
std::string Object::getName ( ) const
```

Devuelve el nombre del objeto.



**3.8.4.27 isDataObject()** [1/2]

```
bool Object::isDataObject (
    std::string & messageName )
```

Determina si el slot buscado del objeto es un DataObject o un MethodObject.

**Parameters**

<i>messageName</i>	nombre del slot.
--------------------	------------------

**Return values**

<i>true</i>	si es data object
<i>false</i>	si no lo es.

**3.8.4.28 isDataObject()** [2/2]

```
bool Object::isDataObject ( )
```

Determina si es un DataObject o un MethodObject.

**Parameters**

<i>messageName</i>	nombre del slot.
--------------------	------------------

**Return values**

<i>true</i>	si es data object
<i>false</i>	si no lo es.

**3.8.4.29 isNativeMethod()**

```
bool Object::isNativeMethod (
    std::string & messageName )
```

Determina si es un metodo nativo.

**Parameters**

<i>messageName</i>	nombre del slot.
--------------------	------------------

**Return values**

<i>true</i>	si es un metodo nativo
<i>false</i>	si no lo es.

### 3.8.4.30 recvMessage()

```
Object * Object::recvMessage (
    std::string & messageName,
    std::vector< Object *> args,
    bool clone )
```

Metodo principal que sirve para recibir mensajes. Devuelve un Object\*

#### Parameters

<i>messageName</i>	nombre del slot que va a recibir el mensaje.
<i>args</i>	vector con Object* con los argumentos para pasar.
<i>clone</i>	indica si hay que clonar o no.

#### Return values

<i>Object</i>	con el slot solicitado con los argumentos procesados.
---------------	-------------------------------------------------------

### 3.8.4.31 collect()

```
Object * Object::collect (
    const std::vector< Object *> & args )
```

Metodo nativo para invocar el garbage collector.

#### Parameters

<i>args</i>	vector de Object* vacio.
-------------	--------------------------

### 3.8.4.32 enableNativeMethod()

```
void Object::enableNativeMethod (
    std::string methodName )
```

Habilita el metodo nativo.

#### Parameters

<i>methodName</i>	nombre del metodo a habilitar.
-------------------	--------------------------------

#### Return values

<i>Object</i>	con el resultado de la operacion. Se opera sobre el primer operando, es decir, sobre el llamante del metodo.
---------------	--------------------------------------------------------------------------------------------------------------

#### 3.8.4.33 disableNativeMethod()

```
void Object::disableNativeMethod (
    std::string methodName )
```

Deshabilita el metodo nativo.

##### Parameters

<i>methodName</i>	nombre del metodo a habilitar.
-------------------	--------------------------------

#### 3.8.4.34 addCreatedObject()

```
void Object::addCreatedObject (
    Object * obj )
```

Agrega un objeto a la lista de objetos creados. Solo debe invocarse desde lobby.

##### Parameters

<i>obj</i>	objeto a agregar.
------------	-------------------

#### 3.8.4.35 findObjectById()

```
Object * Object::findObjectById (
    uint32_t id )
```

Devuelve un objeto. Solo debe invocarse desde Lobby.

##### Parameters

<i>id</i>	id del objeto a buscar.
-----------	-------------------------

#### 3.8.4.36 getId()

```
uint32_t Object::getId ( ) const
```

Devuelve el id del objeto.

#### 3.8.4.37 swapSlotMutability()

```
void Object::swapSlotMutability (
    const std::string & slotName )
```

Cambia el atributo de mutabilidad de un slot.

**Parameters**

<i>slotName</i>	nombre del slot a modificar
-----------------	-----------------------------

**3.8.4.38 setPrimitive()**

```
void Object::setPrimitive (
    const bool newValue )
```

Cambia el atributo de objeto primitivo.

**Parameters**

<i>newValue</i>	nuevo valor.
-----------------	--------------

**3.8.4.39 getPrimitive()**

```
bool Object::getPrimitive ( ) const
```

Devuelve un booleano que indica si el objeto es primitivo o no.

**3.8.5 Member Data Documentation****3.8.5.1 nativeMethods**

```
std::map<std::string, fpointTuple> Object::nativeMethods [private]
```

Diccionario con los metodos nativos. La clave es el nombre y el valor el puntero a la funcion.

**3.9 Parser Class Reference****Public Member Functions**

- [Parser](#) ([VirtualMachine](#) &vm, [Object](#) \*context)
- [Parser](#) (const [Parser](#) &)=delete
- [Parser](#) ([Parser](#) &&)=delete
- [Parser](#) & [operator=](#) (const [Parser](#) &)=delete
- [Parser](#) & [operator=](#) ([Parser](#) &&)=delete
- std::vector< [Object](#) \* > [parse](#) (std::string &cad)

## Private Member Functions

- **Object \* receiveMessage** (Object \*obj, std::string &strName, std::vector< Object \*> &args)
- std::vector< Object \* > **script** ()  
*Valida si en la cadena continua un script.*
- **Object \* expression** ()  
*Valida si en la cadena continua una expression.*
- **Object \* expressionCP** ()  
*Valida si en la cadena continua una expressionCP.*
- **Object \* expressionP** ()  
*Valida si en la cadena continua una expressionP.*
- **Object \* keywordMessage** ()  
*Valida si en la cadena continua un keyword message.*
- **Object \* binaryMessage** ()  
*Valida si en la cadena continua un binary message.*
- **Object \* unaryMessage** ()  
*Valida si en la cadena continua un unary message.*
- **Object \* receiver** ()  
*Valida si en la cadena continua un receiver.*
- bool **slotList** (Object \*objContenedor)  
*Valida si en la cadena continua un slotList.*
- bool **slotNameExtended** (int &tipoSlot, std::string &strName)  
*Valida si en la cadena continua un slotNameExtended.*
- **Object \* constant** ()  
*Valida si en la cadena continua un constant.*
- bool **operador** (std::string &strOperador)  
*Valida si en la cadena continua un operador.*
- bool **operadorSlot** (std::string &strOperadorSlot)  
*Valida si en la cadena continua un operadorSlot.*
- bool **lowerKeyword** (std::string &strLowerKeyword)  
*Valida si en la cadena continua un lowerKeyword.*
- bool **capKeyword** (std::string &strCapKeyword)  
*Valida si en la cadena continua un capKeyword.*
- void **skipSpaces** ()  
*Saltea los espacios.*
- bool **isString** (const std::string strMatch)  
*Valida si el string a continuación de la cadena es el buscado.*
- bool **isLowercaseLetter** ()  
*Valida si en la cadena continua una letra minuscula.*
- bool **isUppercaseLetter** ()  
*Valida si en la cadena continua una letra mayuscula.*
- bool **isLetter** ()  
*Valida si en la cadena continua una letra.*
- bool **isSign** ()  
*Valida si en la cadena continua un signo + -.*
- bool **isDigit** ()  
*Valida si en la cadena continua un digito.*
- bool **isAlpha** ()  
*Valida si en la cadena continua un digito o una letra.*
- **Object \* nilObj** ()
- **Object \* boolObj** ()

- `Object * stringObj ()`
- `Object * numberObj ()`
- `Object * objectObj ()`
- `Object * nameObj (Object *&context)`
- `bool nil ()`  
*Valida si en la cadena continua un nil.*
- `bool isTrue ()`  
*Valida si en la cadena continua un true.*
- `bool isFalse ()`  
*Valida si en la cadena continua un false.*
- `bool name (std::string &strName)`  
*Valida si en la cadena continua un name.*
- `std::string string ()`  
*Valida si en la cadena continua un string.*
- `bool number (float &number)`  
*Valida si en la cadena continua un number.*

## Private Attributes

- `Object * context`  
*Objeto sobre el que se ejecuta el script.*
- `std::string cad`  
*String del script a ejecutar.*
- `uint32_t pCad`  
*Posicion de la cadena cad sobre la que está analizando el parser.*
- `bool debug = false`  
*Modo debug.*
- `int flagExecute`
- `VirtualMachine & vm`  
*Máquina virtual del workspace.*

### 3.9.1 Detailed Description

Es la clase encargada de parsear scripts en código self y de indicarle a la maquina virtual (VM) que objetos y mensajes se deben emitir en consecuencia.

### 3.9.2 Constructor & Destructor Documentation

#### 3.9.2.1 Parser() [1/3]

```
Parser::Parser (
    VirtualMachine & vm,
    Object * context )
```

Constructor

Parameters

<code>vm</code>	Máquina virtual
<code>context</code>	Objeto sobre el que se ejecuta el script.

**3.9.2.2 Parser()** [2/3]

```
Parser::Parser (
    const Parser & ) [delete]
```

Constructor por copia deshabilitado

**3.9.2.3 Parser()** [3/3]

```
Parser::Parser (
    Parser && ) [delete]
```

Constructor por movimiento deshabilitado

**3.9.3 Member Function Documentation****3.9.3.1 operator=()** [1/2]

```
Parser& Parser::operator= (
    const Parser & ) [delete]
```

Operador de asignacion deshabilitado

**3.9.3.2 operator=()** [2/2]

```
Parser& Parser::operator= (
    Parser && ) [delete]
```

Operador de asignacion por movimiento deshabilitado

**3.9.3.3 parse()**

```
std::vector< Object * > Parser::parse (
    std::string & cad )
```

Inicia la secuencia de parseo del script en código self

**Parameters**

<i>cad</i>	código self
------------	-------------

**3.9.3.4 receiveMessage()**

```
Object * Parser::receiveMessage (
    Object * obj,
```

```
std::string & strName,
std::vector< Object *> & args ) [private]
```

Esta es la función más importante del parser, se encarga de decidir que objetos deben ser clonados antes de aplicarles un `recv_message`. Para ello valida si son: data objects: (objetos primitivos o que tengan `codeSegment` vacío). method objects: son aquellos que no son data objects e incluye a los métodos nativos. Los métodos nativos son un caso especial ya que algunos requieren de la clonación como los operadores binarios y hay otros que no la requieren.

### 3.9.3.5 nilObj()

```
Object * Parser::nilObj ( ) [private]
```

Valida si en la cadena continua un nil obj y si es así le pide a la máquina virtual que lo cree.

### 3.9.3.6 boolObj()

```
Object * Parser::boolObj ( ) [private]
```

Valida si en la cadena continua un bool obj y si es así le pide a la máquina virtual que lo cree.

### 3.9.3.7 stringObj()

```
Object * Parser::stringObj ( ) [private]
```

Valida si en la cadena continua un string obj y si es así le pide a la máquina virtual que lo cree.

### 3.9.3.8 numberObj()

```
Object * Parser::numberObj ( ) [private]
```

Valida si en la cadena continua un number obj y si es así le pide a la máquina virtual que lo cree.

### 3.9.3.9 objectObj()

```
Object * Parser::objectObj ( ) [private]
```

Valida si en la cadena continua un objeto y si es así le pide a la máquina virtual que cree un objeto vacío al cual luego se le cargarán los slots.

### 3.9.3.10 nameObj()

```
Object * Parser::nameObj (
    Object *& context ) [private]
```

Valida si en la cadena continua un name y si es así se pide un lookup de ese objeto para recuperarlo y devolverlo.



### 3.9.4 Member Data Documentation

#### 3.9.4.1 flagExecute

```
int Parser::flagExecute [private]
```

Cuando vale 1 se ejecuta el metodo receiveMessage Cada vez que se ingresa a un nuevo script incrementa en 1.

## 3.10 ParserProtocoloMorph Class Reference

### Public Member Functions

- [ParserProtocoloMorph](#) ([Morph](#) &morph, std::string &cad)
- [ParserProtocoloMorph](#) (const [ParserProtocoloMorph](#) &)=delete
- [ParserProtocoloMorph](#) ([ParserProtocoloMorph](#) &&)=delete
- [ParserProtocoloMorph](#) & operator= (const [ParserProtocoloMorph](#) &)=delete
- [ParserProtocoloMorph](#) & operator= ([ParserProtocoloMorph](#) &&)=delete

### Private Member Functions

- std::string [getCampo](#) ()

### Private Attributes

- [Morph](#) & **morph**
- std::string & **cad**
- int **pCad** = 0

#### 3.10.1 Detailed Description

Esta clase se encarga de parsear los mensajes que lleguen al cliente con el comando OK\_MSG\_MORPH y cargar el objeto [Morph](#) con la información obtenida.

#### 3.10.2 Constructor & Destructor Documentation

##### 3.10.2.1 ParserProtocoloMorph() [1/3]

```
ParserProtocoloMorph::ParserProtocoloMorph (
    Morph & morph,
    std::string & cad )
```

### Constructor

#### Parameters

<i>morph</i>	Objeto que contiene la información para dibujar en la ventana <a href="#">MorphWindow</a> .
<i>cad</i>	cadena que se va a parsear según protocolo

### 3.10.2.2 ParserProtocoloMorph() [2/3]

```
ParserProtocoloMorph::ParserProtocoloMorph (
    const ParserProtocoloMorph & ) [delete]
```

Constructor por copia deshabilitado

### 3.10.2.3 ParserProtocoloMorph() [3/3]

```
ParserProtocoloMorph::ParserProtocoloMorph (
    ParserProtocoloMorph && ) [delete]
```

Constructor por movimiento deshabilitado

## 3.10.3 Member Function Documentation

### 3.10.3.1 operator=() [1/2]

```
ParserProtocoloMorph& ParserProtocoloMorph::operator= (
    const ParserProtocoloMorph & ) [delete]
```

Operador de asignacion deshabilitado

### 3.10.3.2 operator=() [2/2]

```
ParserProtocoloMorph& ParserProtocoloMorph::operator= (
    ParserProtocoloMorph && ) [delete]
```

Operador de asignacion por movimiento deshabilitado

### 3.10.3.3 getCampo()

```
std::string ParserProtocoloMorph::getCampo ( ) [private]
```

Captura el siguiente campo de la cadena cad utilizando como separador el caracter especial de protocolo CHAR↔  
\_SEPARADOR

## 3.11 ParserProtocoloServidor Class Reference

### Public Member Functions

- [ParserProtocoloServidor](#) (Object \*obj)
- [ParserProtocoloServidor](#) (const [ParserProtocoloServidor](#) &)=delete
- [ParserProtocoloServidor](#) ([ParserProtocoloServidor](#) &&)=delete
- [ParserProtocoloServidor](#) & operator= (const [ParserProtocoloServidor](#) &)=delete
- [ParserProtocoloServidor](#) & operator= ([ParserProtocoloServidor](#) &&)=delete
- std::string [getString](#) ()

## Private Attributes

- [Object \\* obj](#)

*Objeto con el que se genera la cadena.*

### 3.11.1 Detailed Description

Esta clase se encarga de generar una cadena con el formato especificado por protocolo en funcion de un objeto dado por el servidor para que la misma sea enviada por el [ProxyClient](#) al cliente.

### 3.11.2 Constructor & Destructor Documentation

#### 3.11.2.1 ParserProtocoloServidor() [1/3]

```
ParserProtocoloServidor::ParserProtocoloServidor (
    Object \* obj )
```

Constructor

Parameters

<a href="#">obj</a>	Objeto con el que se genera la cadena
---------------------	---------------------------------------

#### 3.11.2.2 ParserProtocoloServidor() [2/3]

```
ParserProtocoloServidor::ParserProtocoloServidor (
    const ParserProtocoloServidor & ) [delete]
```

Constructor por copia deshabilitado

#### 3.11.2.3 ParserProtocoloServidor() [3/3]

```
ParserProtocoloServidor::ParserProtocoloServidor (
    ParserProtocoloServidor && ) [delete]
```

Constructor por movimiento deshabilitado

### 3.11.3 Member Function Documentation

#### 3.11.3.1 operator=() [1/2]

```
ParserProtocoloServidor& ParserProtocoloServidor::operator= (
    const ParserProtocoloServidor & ) [delete]
```

Operador de asignacion deshabilitado

### 3.11.3.2 operator=() [2/2]

```
ParserProtocoloServidor& ParserProtocoloServidor::operator= (
    ParserProtocoloServidor && ) [delete]
```

Operador de asignacion por moviemento deshabilitado

### 3.11.3.3 getString()

```
std::string ParserProtocoloServidor::getString ( )
```

Genera la cadena por protocolo en funcion del objeto obj y la retorna.

## 3.12 ParserProtocoloWorkspaces Class Reference

### Public Member Functions

- [ParserProtocoloWorkspaces](#) (std::vector< std::string > &workspaces, std::string &cad)
- [ParserProtocoloWorkspaces](#) (const [ParserProtocoloWorkspaces](#) &)=delete
- [ParserProtocoloWorkspaces](#) ([ParserProtocoloWorkspaces](#) &&)=delete
- [ParserProtocoloWorkspaces](#) & operator= (const [ParserProtocoloWorkspaces](#) &)=delete
- [ParserProtocoloWorkspaces](#) & operator= ([ParserProtocoloWorkspaces](#) &&)=delete

### Private Member Functions

- std::string [getCampo](#) ()

### Private Attributes

- std::vector< std::string > & **workspaces**
- std::string & **cad**
- int **pCad** = 0

### 3.12.1 Detailed Description

Esta clase se encarga de parsear los mensajes que lleguen al cliente con el comando OK\_MSG\_SELECT\_WKS y cargar el vector de workspaces con la información obtenida.

### 3.12.2 Constructor & Destructor Documentation

#### 3.12.2.1 ParserProtocoloWorkspaces() [1/3]

```
ParserProtocoloWorkspaces::ParserProtocoloWorkspaces (
    std::vector< std::string > & workspaces,
    std::string & cad )
```

Constructor

## Parameters

<i>workspaces</i>	lista de nombres de workspaces.
<i>cad</i>	cadena que se va a parsear según protocolo

## 3.12.2.2 ParserProtocoloWorkspaces() [2/3]

```
ParserProtocoloWorkspaces::ParserProtocoloWorkspaces (
    const ParserProtocoloWorkspaces & ) [delete]
```

Constructor por copia deshabilitado

## 3.12.2.3 ParserProtocoloWorkspaces() [3/3]

```
ParserProtocoloWorkspaces::ParserProtocoloWorkspaces (
    ParserProtocoloWorkspaces && ) [delete]
```

Constructor por movimiento deshabilitado

## 3.12.3 Member Function Documentation

## 3.12.3.1 operator=() [1/2]

```
ParserProtocoloWorkspaces& ParserProtocoloWorkspaces::operator= (
    const ParserProtocoloWorkspaces & ) [delete]
```

Operador de asignacion deshabilitado

## 3.12.3.2 operator=() [2/2]

```
ParserProtocoloWorkspaces& ParserProtocoloWorkspaces::operator= (
    ParserProtocoloWorkspaces && ) [delete]
```

Operador de asignacion por movimiento deshabilitado

## 3.12.3.3 getCampo()

```
std::string ParserProtocoloWorkspaces::getCampo ( ) [private]
```

Captura el siguiente campo de la cadena cad utilizando como separador el caracter especial de protocolo CHAR↔  
\_SEPARADOR

## 3.13 Proxy Class Reference

Inherits [Thread](#).

Inherited by [ProxyClient](#), and [ProxyServer](#).

## Public Member Functions

- [Proxy](#) ([Socket](#) &socket)
- [Proxy](#) (const [Proxy](#) &)=delete
- [Proxy](#) ([Proxy](#) &&)=delete
- [Proxy](#) & [operator=](#) (const [Proxy](#) &)=delete
- [Proxy](#) & [operator=](#) ([Proxy](#) &&)=delete
- bool [is\\_finished](#) ()
- void [interrupt](#) ()

## Protected Member Functions

- void [send](#) ([Message](#) &message)
- virtual int [receive](#) ()
- void [sendError](#) (std::string msg)
- void [sendOK](#) (std::string msg)
- void [sendOKWks](#) (std::string msg)

## Protected Attributes

- [Socket](#) & [serverSocket](#)  
*[Socket](#) interno.*
- bool [\\_interrupt](#)  
*Flag que indica si hay que interrumpir la recepcion de mensajes.*
- bool [finished](#)  
*Flag que indica si finalizo.*
- [Message](#) [message](#)  
*Mensaje recibido.*

### 3.13.1 Detailed Description

Esta clase es la base para las clases [ProxyClient](#) y [ProxyServer](#). Contiene los metodos y atributos comunes a ambos.

### 3.13.2 Constructor & Destructor Documentation

#### 3.13.2.1 [Proxy\(\)](#) [1/3]

```
Proxy::Proxy (
    Socket & socket )
```

Constructor.

#### Parameters

<a href="#">socket</a>	socket sobre el cual trabajar
------------------------	-------------------------------

### 3.13.2.2 Proxy() [2/3]

```
Proxy::Proxy (
    const Proxy & ) [delete]
```

Constructor por copia eliminado

### 3.13.2.3 Proxy() [3/3]

```
Proxy::Proxy (
    Proxy && ) [delete]
```

Constructor por movimiento eliminado

## 3.13.3 Member Function Documentation

### 3.13.3.1 operator=() [1/2]

```
Proxy& Proxy::operator= (
    const Proxy & ) [delete]
```

Operador asignacion eliminado

### 3.13.3.2 operator=() [2/2]

```
Proxy& Proxy::operator= (
    Proxy && ) [delete]
```

Operador asignacion por movimiento eliminado

### 3.13.3.3 is\_finished()

```
bool Proxy::is_finished ( )
```

Indica si la conexion finalizo.

### 3.13.3.4 interrupt()

```
void Proxy::interrupt ( )
```

Interumpe la ejecucion del proxy.

### 3.13.3.5 send()

```
void Proxy::send (
    Message & message ) [protected]
```

Envia un mensaje del tipo command\_t con el mensaje

#### 3.13.3.6 receive()

```
int Proxy::receive ( ) [protected], [virtual]
```

Recibe datos. Una vez que recibe los datos los guarda internamente.

#### 3.13.3.7 sendError()

```
void Proxy::sendError (
    std::string msg ) [protected]
```

Envia un mensaje de error

##### Parameters

<i>msg</i>	std::string con el mensaje a enviar con la descripcion del error
------------	------------------------------------------------------------------

#### 3.13.3.8 sendOK()

```
void Proxy::sendOK (
    std::string msg ) [protected]
```

Envia un mensaje de OK para el [Morph](#)

##### Parameters

<i>msg</i>	con el resultado de la operacion.
------------	-----------------------------------

#### 3.13.3.9 sendOKWks()

```
void Proxy::sendOKWks (
    std::string msg ) [protected]
```

Envia un mensaje de OK para la ventana selectora de Workspaces

##### Parameters

<i>msg</i>	con el resultado de la operacion.
------------	-----------------------------------

## 3.14 ProxyClient Class Reference

Inherits [Proxy](#).



## Public Member Functions

- [ProxyClient](#) ([Socket](#) &socket, [Server](#) &server)
- [ProxyClient](#) (const [ProxyClient](#) &)=delete
- [ProxyClient](#) ([ProxyClient](#) &&)=delete
- [ProxyClient](#) & [operator=](#) (const [ProxyClient](#) &)=delete
- [ProxyClient](#) & [operator=](#) ([ProxyClient](#) &&)=delete
- [~ProxyClient](#) ()
- virtual void [run](#) ()

## Private Member Functions

- void [execLobbyCMD](#) (std::string &cad)
- void [execLocalCMD](#) (std::string &cad)
- void [showLobby](#) ()
- void [execRefresh](#) ()
- void [setObjName](#) (const std::string &cad)
- void [setCodeSegment](#) (const std::string &cad)
- void [getSlotObj](#) (const std::string &cad)
- void [swapMutability](#) (const std::string &cad)
- void [goBack](#) ()
- void [availableWks](#) ()
- void [loadWks](#) (const std::string &cad)
- void [newWks](#) (const std::string &cad)
- void [deleteWks](#) (const std::string &cad)
- void [closeWks](#) ()
- uint32\_t [topObj](#) ()

## Private Attributes

- [Server](#) & [server](#)  
*Servidor al que el proxy le hace las consultas.*
- std::string [idWorkspace](#)  
*Workspace en el que se encuentra el cliente.*
- std::stack< uint32\_t > [seenObj](#)  
*Pila de IDs de objetos vistos por el cliente.*
- [Socket](#) \* [sckptr](#) = nullptr  
*Socket asociado al proxyClient.*

## Additional Inherited Members

### 3.14.1 Detailed Description

El [ProxyClient](#) es el encargado de responder las peticiones del cliente ([ProxyServer](#)). Para resolver las peticiones delega las consultas al servidor(Server/Modelo de Negocio). El [ProxyClient](#) solo conoce los IDs del [Workspace](#) y de los objetos con los que trabaja.

### 3.14.2 Constructor & Destructor Documentation

#### 3.14.2.1 [ProxyClient](#)() [1/3]

```
ProxyClient::ProxyClient (
    Socket & socket,
    Server & server )
```

Constructor.

## Parameters

<i>socket</i>	socket sobre el cual trabajar
<i>server</i>	referencia al modelo de negocio

**3.14.2.2 ProxyClient()** [2/3]

```
ProxyClient::ProxyClient (
    const ProxyClient & ) [delete]
```

Constructor por copia eliminado

**3.14.2.3 ProxyClient()** [3/3]

```
ProxyClient::ProxyClient (
    ProxyClient && ) [delete]
```

Constructor por movimiento eliminado

**3.14.2.4 ~ProxyClient()**

```
ProxyClient::~~ProxyClient ( )
```

Destructor.

**3.14.3 Member Function Documentation****3.14.3.1 operator=()** [1/2]

```
ProxyClient& ProxyClient::operator= (
    const ProxyClient & ) [delete]
```

Operador asignacion eliminado

**3.14.3.2 operator=()** [2/2]

```
ProxyClient& ProxyClient::operator= (
    ProxyClient && ) [delete]
```

Operador asignacion por movimiento eliminado

**3.14.3.3 run()**

```
void ProxyClient::run ( ) [virtual]
```

Metodo que sirve para procesar la solicitud que le envia el cliente.

Implements [Thread](#).

**3.14.3.4 execLobbyCMD()**

```
void ProxyClient::execLobbyCMD (
    std::string & cad ) [private]
```

Le pide al servidor que ejecute un script de código self con el entorno/contexto de lobby y le retorna al cliente la respuesta.

## Parameters

<i>cad</i>	script a procesar por el servidor.
------------	------------------------------------

## 3.14.3.5 execLocalCMD()

```
void ProxyClient::execLocalCMD (
    std::string & cad ) [private]
```

Le pide al servidor que ejecute un script de código self con el entorno/contexto del objeto que ve el cliente y le retorna al cliente la respuesta.

## Parameters

<i>cad</i>	script a procesar por el servidor.
------------	------------------------------------

## 3.14.3.6 showLobby()

```
void ProxyClient::showLobby ( ) [private]
```

Le pide al servidor la cadena que representa por protocolo a lobby y le retorna al cliente la respuesta.

## 3.14.3.7 execRefresh()

```
void ProxyClient::execRefresh ( ) [private]
```

Le pide al servidor la cadena que representa por protocolo al objeto que esta viendo el cliente para actualizar las novedades y le retorna al cliente la respuesta.

## 3.14.3.8 setObjName()

```
void ProxyClient::setObjName (
    const std::string & cad ) [private]
```

Le pide al servidor que le setee el nombre al objeto que ve el cliente. Retorna al cliente la respuesta con el objeto modificado.

## Parameters

<i>cad</i>	nuevo nombre.
------------	---------------

## 3.14.3.9 setCodeSegment()

```
void ProxyClient::setCodeSegment (
    const std::string & cad ) [private]
```

Le pide al servidor que le setee el bloque de código al objeto que ve el cliente y retorna al cliente la respuesta con el objeto modificado.

#### Parameters

<i>cad</i>	nuevo bloque de código.
------------	-------------------------

#### 3.14.3.10 getSlotObj()

```
void ProxyClient::getSlotObj (
    const std::string & cad ) [private]
```

Le pide al servidor el objeto contenido en el slot del objeto que el cliente está viendo.

#### Parameters

<i>cad</i>	nombre del slot en el objeto que ve el cliente.
------------	-------------------------------------------------

#### 3.14.3.11 swapMutability()

```
void ProxyClient::swapMutability (
    const std::string & cad ) [private]
```

Le pide al servidor cambiar la mutabilidad del slot del objeto que el cliente está viendo.

#### Parameters

<i>cad</i>	nombre del slot en el objeto que ve el cliente.
------------	-------------------------------------------------

#### 3.14.3.12 goBack()

```
void ProxyClient::goBack ( ) [private]
```

Le pide al servidor el objeto anterior de la pila seenObj y se lo devuelve al cliente.

#### 3.14.3.13 availableWks()

```
void ProxyClient::availableWks ( ) [private]
```

Le pide al servidor una lista de workspaces existentes y le devuelve esa lista al cliente.

#### 3.14.3.14 loadWks()

```
void ProxyClient::loadWks (
    const std::string & cad ) [private]
```

Le indica al servidor que un cliente va a entrar a un workspace por lo que el servidor le debe retornar el lobby de ese workspace para devolvérselo al cliente.

## Parameters

<i>cad</i>	nombre del workspace a cargar.
------------	--------------------------------

## 3.14.3.15 newWks()

```
void ProxyClient::newWks (
    const std::string & cad ) [private]
```

Le indica al servidor que se debe crear un nuevo workspace y le retorna al cliente el lobby que le devolvi el server.

## Parameters

<i>cad</i>	nombre del nuevo workspace.
------------	-----------------------------

## 3.14.3.16 deleteWks()

```
void ProxyClient::deleteWks (
    const std::string & cad ) [private]
```

Le indica al servidor que se debe eliminar un workspace y le retorna al cliente la nueva lista de workspace disponibles.

## Parameters

<i>cad</i>	nombre del workspace a eliminar.
------------	----------------------------------

## 3.14.3.17 closeWks()

```
void ProxyClient::closeWks ( ) [private]
```

Le indica al servidor que el cliente se desconecta del workspace actual.

## 3.14.3.18 topObj()

```
uint32_t ProxyClient::topObj ( ) [private]
```

Devuelve el ID del objeto que está mas arriba en la pila de seenObj. Es decir de los objetos vistos por el cliente hasta el momento.

## 3.15 ProxyServer Class Reference

Inherits [Proxy](#).

## Public Member Functions

- [ProxyServer](#) ([Socket](#) &socket, [Morph](#) &morph, std::vector< std::string > &[workspaces](#), std::mutex &m)
- bool [sendCmdMessage](#) (char command, std::string &strMessage)
- void [run](#) ()
- bool [getFlag](#) () const
- bool [areThereErrors](#) () const
- std::string [getErrors](#) ()
- void [setFlag](#) (const bool newValue)

## Private Member Functions

- virtual void [sendCMDMessage](#) ()

## Private Attributes

- [Morph](#) & [morph](#)  
*Morph interno de la clase.*
- std::vector< std::string > & [workspaces](#)  
*Lista de nombres de workspaces.*
- std::mutex & [m](#)  
*Mutex pasado por referencia.*
- bool [flag](#)  
*Este flag le indica al proxy que debe ejecutar un comando.*
- std::string [errorMsg](#)  
*Mensaje de error si es que hay.*

## Additional Inherited Members

### 3.15.1 Detailed Description

Es la encargada de enviar las peticiones generadas desde la GUI al servidor.

### 3.15.2 Constructor & Destructor Documentation

#### 3.15.2.1 ProxyServer()

```
ProxyServer::ProxyServer (
    Socket & socket,
    Morph & morph,
    std::vector< std::string > & workspaces,
    std::mutex & m )
```

#### Constructor

##### Parameters

<i>socket</i>	<i>morph</i>
<i>workspaces</i>	<i>lista de nombres de los workspaces</i>
<i>m</i>	<i>mutex</i>

### 3.15.3 Member Function Documentation

#### 3.15.3.1 sendCMDMessage()

```
void ProxyServer::sendCMDMessage ( ) [private], [virtual]
```

Envia el mensaje que esta guardado en message al servidor

#### 3.15.3.2 sendCmdMessage()

```
bool ProxyServer::sendCmdMessage (
    char command,
    std::string & strMessage )
```

Envia un mensaje para ejecutar codigo self

##### Parameters

<i>command</i>	comando a enviar
<i>strMessage</i>	mensaje a enviar

#### 3.15.3.3 run()

```
void ProxyServer::run ( ) [virtual]
```

Metodo que sirve para procesar la respuesta que le envia el servidor.

Implements [Thread](#).

#### 3.15.3.4 getFlag()

```
bool ProxyServer::getFlag ( ) const
```

Obtiene el flag que determina que envio el comando y recibio la respuesta por parte del servidor. Indica que ya se concretaron las operaciones y que se puede volver a enviar nuevamente.

#### 3.15.3.5 areThereErrors()

```
bool ProxyServer::areThereErrors ( ) const
```

Indica si hubo o no errores informados por el servidor.

#### 3.15.3.6 getErrors()

```
std::string ProxyServer::getErrors ( )
```

Obtiene los errores. Si no hubo ninguno devuelve una cadena vacio, si hay devuelve el error.

#### 3.15.3.7 setFlag()

```
void ProxyServer::setFlag (
    const bool newValue )
```

Setea el flag que indica si se esta esperando una respuesta del servidor.

## Parameters

<i>newValue</i>	nuevo valor del flag
-----------------	----------------------

### 3.16 SelectWkWindow Class Reference

Inherits Window.

#### Public Member Functions

- [SelectWkWindow](#) ([Morph](#) &morph, std::vector< std::string > &workspaces, [ProxyServer](#) &proxyServer, std::mutex &m)
- Gtk::Window \* [getWindow](#) ()
- **SelectWkWindow** (const [SelectWkWindow](#) &)=delete
- **SelectWkWindow** ([SelectWkWindow](#) &&)=delete
- [SelectWkWindow](#) & **operator=** (const [SelectWkWindow](#) &)=delete
- [SelectWkWindow](#) & **operator=** ([SelectWkWindow](#) &&)=delete

#### Private Member Functions

- void **btnRefreshWk\_clicked** ()
- void **btnNewWk\_clicked** ()
- void **treeView\_toggled** (const Glib::ustring &path)
- void **treeView\_on\_row\_activated** (const Gtk::TreeModel::Path &path, Gtk::TreeViewColumn \*column)
- void [addWidget](#) ()
- void [configureTreeView](#) ()
- void [drawWorkspaces](#) ()
- void [updateList](#) ()

#### Private Attributes

- [Morph](#) & **morph**
- std::vector< std::string > & **workspaces**
- [ProxyServer](#) & **proxyServer**
- std::mutex & **m**
- Glib::RefPtr< Gtk::Builder > **refBuilder**
- Gtk::Window \* **pWindow** = nullptr
- Gtk::Button \* **pBtnRefreshWk** = nullptr
- Gtk::Button \* **pBtnNewWk** = nullptr
- Gtk::Entry \* **pTxtNewWk** = nullptr
- Gtk::TreeView \* **pTreeView** = nullptr
- Glib::RefPtr< Gtk::TreeStore > **m\_refTreeModel**
- [ColumnRecordWk](#) **m\_Columns**

#### 3.16.1 Detailed Description

Se encarga de dibujar la ventana selectora de workspaces.



## 3.16.2 Constructor & Destructor Documentation

### 3.16.2.1 SelectWkWindow()

```
SelectWkWindow::SelectWkWindow (
    Morph & morph,
    std::vector< std::string > & workspaces,
    ProxyServer & proxyServer,
    std::mutex & m )
```

Constructor de la clase

#### Parameters

<i>morph</i>	referencia al objeto <a href="#">Morph</a> .
<i>workspaces</i>	referencia al vector con los nombres de los workspaces
<i>proxyServer</i>	referencia al proxy
<i>m</i>	referencia al mutex

## 3.16.3 Member Function Documentation

### 3.16.3.1 addWidgets()

```
void SelectWkWindow::addWidgets ( ) [private]
```

Este metodo agrega los widgets a la ventana

### 3.16.3.2 configureTreeView()

```
void SelectWkWindow::configureTreeView ( ) [private]
```

Este metodo configura las columnas los eventos del TreeView.

### 3.16.3.3 drawWorkspaces()

```
void SelectWkWindow::drawWorkspaces ( ) [private]
```

Este metodo dibuja los datos recibidos sobre los workspaces provenientes del servidor en el TreeView. Previamente hay que llamar al metodo [updateList\(\)](#) para actualizar los datos en memoria. Este metodo no los actualiza, solo dibuja lo que esta cargado.

### 3.16.3.4 updateList()

```
void SelectWkWindow::updateList ( ) [private]
```

Manda la solicitud al servidor para actualizar la lista de workspaces disponibles que esta guardada en memoria. Para redibujar la lista, llamar a [drawWorkspaces\(\)](#).

### 3.16.3.5 getWindow()

```
Gtk::Window * SelectWkWindow::getWindow ( )
```

Devuelve un puntero al objeto Gtk::Window.

## 3.17 Server Class Reference

### Public Member Functions

- [Server](#) ()
- [~Server](#) ()
- **Server** (const [Server](#) &)=delete
- **Server** ([Server](#) &&)=delete
- [Server](#) & **operator=** (const [Server](#) &)=delete
- [Server](#) & **operator=** ([Server](#) &&)=delete
- void [loadWorkspace](#) (std::string name)
- std::vector< std::string > [availableWorkspace](#) ()
- void [newWorkspace](#) (std::string name)
- void [closeWorkspace](#) (std::string name)
- void [deleteWorkspace](#) (std::string name)
- std::string [receiveCode](#) (const std::string &idWk, uint32\_t &idObj, std::string &code)
- std::string [getLobby](#) (const std::string &idWk, uint32\_t &idObj)
- std::string [getObj](#) (const std::string &idWk, uint32\_t &idObj)
- std::string [setObjName](#) (const std::string &idWk, uint32\_t &idObj, const std::string &cad)
- std::string [setCodeSegment](#) (const std::string &idWk, uint32\_t &idObj, const std::string &cad)
- std::string [getSlotObj](#) (const std::string &idWk, uint32\_t &idObj, const std::string &cad)
- std::string [swapMutability](#) (const std::string &idWk, uint32\_t &idObj, const std::string &cad)

### Private Types

- typedef std::tuple< [Workspace](#) \*, uint32\_t > **workspace\_tuple**

### Private Member Functions

- [Workspace](#) \* [getWorkspace](#) (const std::string &idWk)

### Private Attributes

- std::mutex **m**
- std::map< std::string, workspace\_tuple > [workspaces](#)

### 3.17.1 Detailed Description

Representa el modelo de negocio. Resuelve las peticiones de los [ProxyClient](#)'s y administra los recursos que se deben proteger.

## 3.17.2 Constructor & Destructor Documentation

### 3.17.2.1 Server()

```
Server::Server ( )
```

Constructor

### 3.17.2.2 ~Server()

```
Server::~~Server ( )
```

Destructor

## 3.17.3 Member Function Documentation

### 3.17.3.1 getWorkspace()

```
Workspace * Server::getWorkspace (
    const std::string & idWk ) [private]
```

Retorna el workspaces en función del id solicitado

Parameters

<i>idWk</i>	id del workspaces
-------------	-------------------

### 3.17.3.2 loadWorkspace()

```
void Server::loadWorkspace (
    std::string name )
```

Acumula en el contador de clientes del workspace en el mapa workspaces

Parameters

<i>name</i>	id del workspace al que se conecta el cliente
-------------	-----------------------------------------------

### 3.17.3.3 availableWorkspace()

```
std::vector< std::string > Server::availableWorkspace ( )
```

Retorna una cadena por formato de protocolo con la lista de workspaces disponibles.

#### 3.17.3.4 newWorkspace()

```
void Server::newWorkspace (
    std::string name )
```

Inicializa un nuevo workspace con el nombre pasado por parametro.

##### Parameters

<i>name</i>	id del nuevo workspace
-------------	------------------------

#### 3.17.3.5 closeWorkspace()

```
void Server::closeWorkspace (
    std::string name )
```

Desacumula en el contador de clientes del workspace en el mapa workspaces

##### Parameters

<i>name</i>	id del workspace del que se desconecta el cliente
-------------	---------------------------------------------------

#### 3.17.3.6 deleteWorkspace()

```
void Server::deleteWorkspace (
    std::string name )
```

Elimina el workspace con el nombre pasado por parametro.

##### Parameters

<i>name</i>	id del workspace a eliminar
-------------	-----------------------------

#### 3.17.3.7 receiveCode()

```
std::string Server::receiveCode (
    const std::string & idWk,
    uint32_t & idObj,
    std::string & code )
```

Le pide al workspace que ejecute codigo self con el contexto dado.

##### Parameters

<i>idWk</i>	id del workspace
<i>idObj</i>	id del objeto de ese workspace
<i>code</i>	script de código self

### 3.17.3.8 getLobby()

```
std::string Server::getLobby (
    const std::string & idWk,
    uint32_t & idObj )
```

Le pide el lobby al workspace y genera la cadena del objeto por protocolo

#### Parameters

<i>idWk</i>	id del workspace
<i>idObj</i>	id del objeto de ese workspace

### 3.17.3.9 getObj()

```
std::string Server::getObj (
    const std::string & idWk,
    uint32_t & idObj )
```

Le pide el objeto al workspace y genera la cadena del objeto por protocolo

#### Parameters

<i>idWk</i>	id del workspace
<i>idObj</i>	id del objeto de ese workspace

### 3.17.3.10 setObjName()

```
std::string Server::setObjName (
    const std::string & idWk,
    uint32_t & idObj,
    const std::string & cad )
```

Setea el nombre del objeto y genera la cadena del objeto por protocolo

#### Parameters

<i>idWk</i>	id del workspace
<i>idObj</i>	id del objeto de ese workspace
<i>cad</i>	nuevo nombre del objeto

### 3.17.3.11 setCodeSegment()

```
std::string Server::setCodeSegment (
    const std::string & idWk,
    uint32_t & idObj,
    const std::string & cad )
```

Setea el bloque de código del objeto y genera la cadena del objeto por protocolo

## Parameters

<i>idWk</i>	id del workspace
<i>idObj</i>	id del objeto de ese workspace
<i>cad</i>	nuevo bloque de código

3.17.3.12 `getSlotObj()`

```
std::string Server::getSlotObj (
    const std::string & idWk,
    uint32_t & idObj,
    const std::string & cad )
```

Genera la cadena por protocolo del objeto contenido en el slot

## Parameters

<i>idWk</i>	id del workspace
<i>idObj</i>	id del objeto de ese workspace
<i>cad</i>	nombre del slot

3.17.3.13 `swapMutability()`

```
std::string Server::swapMutability (
    const std::string & idWk,
    uint32_t & idObj,
    const std::string & cad )
```

Cambia la mutabilidad del slot y genera la cadena por protocolo del objeto

## Parameters

<i>idWk</i>	id del workspace
<i>idObj</i>	id del objeto de ese workspace
<i>cad</i>	nombre del slot

## 3.17.4 Member Data Documentation

3.17.4.1 `workspaces`

```
std::map<std::string, workspace_tuple> Server::workspaces [private]
```

Mapa con clave ID workspaces y valor una tupla con un puntero al workspace y la cantidad de clientes activos en el mismo.

## 3.18 Socket Class Reference

### Public Member Functions

- [Socket](#) (std::string hostname, uint32\_t port)
- [Socket](#) (uint32\_t port)
- [Socket](#) (const [Socket](#) &)
- Constructor por copia.*
- [~Socket](#) ()
- Destructor.*
- void [bind\\_and\\_listen](#) ()
- Metodo que sirve para escuchar en un determinado puerto.*
- void [connect](#) ()
- Metodo que se usa para conectar un cliente a un servidor.*
- [Socket](#) \* [accept](#) ()
- void [send](#) (const char \*buffer, uint32\_t length)
- int [receive](#) (char \*buffer, uint32\_t length)
- void [shutdown](#) ()
- Cierra el socket y libera los recursos.*
- [Socket](#) & [operator=](#) (const [Socket](#) &)=delete
- Operador de asignacion deshabilitado.*

### Private Member Functions

- void [initialize](#) (uint32\_t flags)

### Private Attributes

- std::string **hostname**
- uint32\_t **port**
- int **socket\_fd**
- int **accepted\_socket\_fd**
- struct addrinfo **hints**
- struct addrinfo **addr**
- struct addrinfo \* **ptr**
- bool **\_shutdown**

#### 3.18.1 Detailed Description

Representa una encapsulacion de los sockets provistos por el sistema operativo que estan en las librerias de Unix.

#### 3.18.2 Constructor & Destructor Documentation

##### 3.18.2.1 [Socket\(\)](#) [1/2]

```
Socket::Socket (
    std::string hostname,
    uint32_t port )
```

Constructor



## Parameters

<i>hostname</i>	IP del servidor
<i>port</i>	puerto a escuchar

## 3.18.2.2 Socket() [2/2]

```
Socket::Socket (
    uint32_t port ) [explicit]
```

## Constructor

## Parameters

<i>port</i>	puerto a escuchar
-------------	-------------------

## 3.18.3 Member Function Documentation

## 3.18.3.1 send()

```
void Socket::send (
    const char * buffer,
    uint32_t length )
```

Metodo que sirve para enviar un mensaje

## Parameters

<i>buffer</i>	desde donde leer los datos a enviar
<i>length</i>	tamano en bytes de los datos a enviar

## 3.18.3.2 receive()

```
int Socket::receive (
    char * buffer,
    uint32_t length )
```

Metodo que sirve para recibir un mensaje

## Parameters

<i>buffer</i>	donde guardar los datos recibidos
<i>length</i>	tamano en bytes recibidos

### 3.19 Thread Class Reference

Inherited by [Acceptor](#), and [Proxy](#).

#### Public Member Functions

- void **start** ()
- void **join** ()
- virtual void **run** ()=0

#### Private Member Functions

- **Thread** (const [Thread](#) &)=delete
- [Thread](#) & **operator=** (const [Thread](#) &)=delete
- **Thread** ([Thread](#) &&other)
- [Thread](#) & **operator=** ([Thread](#) &&other)

#### Private Attributes

- std::thread **thread**

#### 3.19.1 Detailed Description

Encapsula los metodos para iniciar, correr y joinear hilos.

### 3.20 VirtualMachine Class Reference

#### Public Member Functions

- **VirtualMachine** (const [VirtualMachine](#) &)=delete
- **VirtualMachine** ([VirtualMachine](#) &&)=delete
- [VirtualMachine](#) & **operator=** (const [VirtualMachine](#) &)=delete
- [VirtualMachine](#) & **operator=** ([VirtualMachine](#) &&)=delete
- [VirtualMachine](#) ()
- [Object](#) \* **createNil** ()
- [Object](#) \* **createString** (std::string &strString)
- [Object](#) \* **createNumber** (float number)
- [Object](#) \* **createBoolean** (bool value)
- [Object](#) \* **createEmptyObject** ()
- [Object](#) \* **findObjectById** (uint32\_t id)
- void **setLobby** ([Object](#) \*lobby)

#### Private Attributes

- [Object](#) \* **lobby** = nullptr

### 3.20.1 Detailed Description

Es la encargada de crear y almacenarlos en una pila en el objeto lobby. Tanto la maquina virtual como el objeto lobby son unicos por workspace.

### 3.20.2 Constructor & Destructor Documentation

#### 3.20.2.1 VirtualMachine()

```
VirtualMachine::VirtualMachine ( )
```

Constructor

### 3.20.3 Member Function Documentation

#### 3.20.3.1 createNil()

```
Object * VirtualMachine::createNil ( )
```

Crea un objeto primitivo nil

#### 3.20.3.2 createString()

```
Object * VirtualMachine::createString (
    std::string & strString )
```

Crea un objeto primitivo string

Parameters

<i>strString</i>	cadena del objeto
------------------	-------------------

#### 3.20.3.3 createNumber()

```
Object * VirtualMachine::createNumber (
    float number )
```

Crea un objeto primitivo number

Parameters

<i>number</i>	número del objeto
---------------	-------------------

#### 3.20.3.4 createBoolean()

```
Object * VirtualMachine::createBoolean (
    bool value )
```

Crea un objeto primitivo booleano

##### Parameters

<i>value</i>	estado del objeto booleano
--------------	----------------------------

#### 3.20.3.5 createEmptyObject()

```
Object * VirtualMachine::createEmptyObject ( )
```

Crea un objeto no primitivo vacio

#### 3.20.3.6 findObjectById()

```
Object * VirtualMachine::findObjectById (
    uint32_t id )
```

Busca un objeto por su ID y lo retorna

##### Parameters

<i>id</i>	id del objeto buscado
-----------	-----------------------

#### 3.20.3.7 setLobby()

```
void VirtualMachine::setLobby (
    Object * lobby )
```

Setea el lobby de la maquina virtual creado por el workspace

##### Parameters

<i>lobby</i>	objeto lobby creado por el workspace
--------------	--------------------------------------

## 3.21 Workspace Class Reference

### Public Member Functions

- [Workspace](#) ()

*Constructor.*

- **Workspace** (const [Workspace](#) &)=delete
- **Workspace** ([Workspace](#) &&)=delete
- [Workspace](#) & **operator=** (const [Workspace](#) &)=delete
- [Workspace](#) & **operator=** ([Workspace](#) &&)=delete
- [~Workspace](#) ()

*Destructor.*

- uint32\_t **receive** ([Object](#) \*context, std::string &code)
- [Object](#) \* **getLobby** ()

*Devuelve el objeto lobby.*

- [Object](#) \* **findObjectById** (uint32\_t id)

## Private Attributes

- [Object](#) \* **lobby**
- [VirtualMachine](#) **vm**

### 3.21.1 Detailed Description

Representa un ambiente de trabajo (workspace) y es el encargado de crear el lobby y de llamar al parser para ejecutar los scripts de código self.

### 3.21.2 Member Function Documentation

#### 3.21.2.1 receive()

```
uint32_t Workspace::receive (
    Object * context,
    std::string & code )
```

Recibe codigo self y lo ejecuta.

#### Parameters

<i>code</i>	[std::string] codigo que recibe
-------------	---------------------------------

#### Returns

Devuelve un [Object](#)\* con el resultado de la ejecucion

#### 3.21.2.2 findObjectById()

```
Object * Workspace::findObjectById (
    uint32_t id )
```

Busca el objeto por ID y retorna el objeto real.

**Parameters**

<i>id</i>	id del objeto.
-----------	----------------

# Index

- \_AddSlots
    - Object, [23](#)
  - \_RemoveSlots
    - Object, [24](#)
  - ~MorphWindow
    - MorphWindow, [19](#)
  - ~ProxyClient
    - ProxyClient, [46](#)
  - ~Server
    - Server, [55](#)
- Acceptor, [5](#)
  - Acceptor, [6](#)
  - collectClosedClients, [7](#)
  - interrupt, [6](#)
  - operator=, [6](#)
  - run, [7](#)
- addCreatedObject
  - Object, [31](#)
- addSlot
  - Morph, [17](#)
  - Object, [27](#)
- addWidgets
  - MorphWindow, [19](#)
  - SelectWkWindow, [53](#)
- areThereErrors
  - ProxyServer, [51](#)
- availableWks
  - ProxyClient, [48](#)
- availableWorkspace
  - Server, [55](#)
- boolObj
  - Parser, [36](#)
- clear
  - Morph, [13](#)
- clone
  - Object, [24](#)
- closeWks
  - ProxyClient, [49](#)
- closeWorkspace
  - Server, [56](#)
- collect
  - Object, [30](#)
- collect\_internal
  - Object, [23](#)
- collectClosedClients
  - Acceptor, [7](#)
- ColumnRecord, [7](#)
- ColumnRecordWk, [7](#)
- configureNativeMethods
  - Object, [23](#)
- configureTreeView
  - MorphWindow, [19](#)
  - SelectWkWindow, [53](#)
- createBoolean
  - VirtualMachine, [63](#)
- createEmptyObject
  - VirtualMachine, [64](#)
- createNil
  - VirtualMachine, [63](#)
- createNumber
  - VirtualMachine, [63](#)
- createString
  - VirtualMachine, [63](#)
- delegate
  - Object, [22](#)
- deleteWks
  - ProxyClient, [49](#)
- deleteWorkspace
  - Server, [56](#)
- disableNativeMethod
  - Object, [30](#)
- doAction
  - MorphWindow, [19](#)
- drawMorph
  - MorphWindow, [19](#)
- drawWorkspaces
  - SelectWkWindow, [53](#)
- enableNativeMethod
  - Object, [30](#)
- execLobbyCMD
  - ProxyClient, [46](#)
- execLocalCMD
  - ProxyClient, [47](#)
- execRefresh
  - ProxyClient, [47](#)
- exitRoutine
  - ModeSelector, [11](#)
- findObject
  - Object, [23](#)
- findObjectById
  - Object, [31](#)
  - VirtualMachine, [64](#)
  - Workspace, [65](#)
- flagExecute

- Parser, 37
- fpointTuple
  - Object, 22
- getCampo
  - ParserProtocoloMorph, 38
  - ParserProtocoloWorkspaces, 41
- getCodeSegment
  - Morph, 13
  - Object, 28
- getCommand
  - Message, 9
- getErrors
  - ProxyServer, 51
- getFlag
  - ProxyServer, 51
- getId
  - Object, 31
- getLength
  - Message, 8
- getLobby
  - Server, 57
- getMessage
  - Message, 9
- getName
  - Object, 28
- getNativeMethods
  - Object, 27
- getObj
  - Server, 57
- getObjName
  - Morph, 13
- getParentSlots
  - Object, 22
- getPrimitive
  - Object, 32
- getSlotName
  - Morph, 14
- getSlotObj
  - ProxyClient, 48
  - Server, 59
- getSlotObjName
  - Morph, 16
- getSlotObjPreview
  - Morph, 16
- getSlots
  - Object, 27
- getSlotsSize
  - Morph, 13
- getString
  - ParserProtocoloServidor, 40
- getWindow
  - MorphWindow, 20
  - SelectWkWindow, 53
- getWorkspace
  - Server, 55
- goBack
  - ProxyClient, 48
- interrupt
  - Acceptor, 6
  - Proxy, 43
- is\_finished
  - Proxy, 43
- isArgumentSlot
  - Morph, 15
- isDataObject
  - Object, 28, 29
- isMutableSlot
  - Morph, 15
- isNativeMethod
  - Object, 29
- isNativeMethodSlot
  - Morph, 15
- isParentSlot
  - Morph, 15
- loadWks
  - ProxyClient, 48
- loadWorkspace
  - Server, 55
- Message, 8
  - getCommand, 9
  - getLength, 8
  - getMessage, 9
  - setCommand, 9
  - setLength, 8
  - setMessage, 9
  - toString, 8
- ModeSelector, 9
  - exitRoutine, 11
  - ModeSelector, 10
  - operator=, 11
- Morph, 11
  - addSlot, 17
  - clear, 13
  - getCodeSegment, 13
  - getObjName, 13
  - getSlotName, 14
  - getSlotObjName, 16
  - getSlotObjPreview, 16
  - getSlotsSize, 13
  - isArgumentSlot, 15
  - isMutableSlot, 15
  - isNativeMethodSlot, 15
  - isParentSlot, 15
  - Morph, 12, 13
  - mostrar, 17
  - operator=, 13
  - setCodeSegment, 16
  - setObjName, 16
  - slot\_morph, 12
- MorphWindow, 17
  - ~MorphWindow, 19
  - addWidgets, 19
  - configureTreeView, 19
  - doAction, 19



- drawMorph, 19
- getWindow, 20
- MorphWindow, 19
- mostrar
  - Morph, 17
- nameObj
  - Parser, 36
- nativeMethods
  - Object, 32
- newWks
  - ProxyClient, 49
- newWorkspace
  - Server, 55
- nilObj
  - Parser, 36
- numberObj
  - Parser, 36
- Object, 20
  - \_AddSlots, 23
  - \_RemoveSlots, 24
  - addCreatedObject, 31
  - addSlot, 27
  - clone, 24
  - collect, 30
  - collect\_internal, 23
  - configureNativeMethods, 23
  - delegate, 22
  - disableNativeMethod, 30
  - enableNativeMethod, 30
  - findObject, 23
  - findObjectById, 31
  - fpointTuple, 22
  - getCodeSegment, 28
  - getId, 31
  - getName, 28
  - getNativeMethods, 27
  - getParentSlots, 22
  - getPrimitive, 32
  - getSlots, 27
  - isDataObject, 28, 29
  - isNativeMethod, 29
  - nativeMethods, 32
  - Object, 22
  - operator!=, 26
  - operator\*, 25
  - operator+, 25
  - operator-, 25
  - operator/, 26
  - operator=, 27
  - operator==, 26
  - print, 24
  - printObj, 24
  - recvMessage, 29
  - removeSlot, 27
  - setCodeSegment, 28
  - setName, 28
  - setPrimitive, 32
  - slot\_map, 22
  - slot\_t, 22
  - swapSlotMutability, 31
- objectObj
  - Parser, 36
- operator!=
  - Object, 26
- operator\*
  - Object, 25
- operator+
  - Object, 25
- operator-
  - Object, 25
- operator/
  - Object, 26
- operator=
  - Accepter, 6
  - ModeSelector, 11
  - Morph, 13
  - Object, 27
  - Parser, 35
  - ParserProtocoloMorph, 38
  - ParserProtocoloServidor, 39
  - ParserProtocoloWorkspaces, 41
  - Proxy, 43
  - ProxyClient, 46
- operator==
  - Object, 26
- parse
  - Parser, 35
- Parser, 32
  - boolObj, 36
  - flagExecute, 37
  - nameObj, 36
  - nilObj, 36
  - numberObj, 36
  - objectObj, 36
  - operator=, 35
  - parse, 35
  - Parser, 34, 35
  - receiveMessage, 35
  - stringObj, 36
- ParserProtocoloMorph, 37
  - getCampo, 38
  - operator=, 38
  - ParserProtocoloMorph, 37, 38
- ParserProtocoloServidor, 38
  - getString, 40
  - operator=, 39
  - ParserProtocoloServidor, 39
- ParserProtocoloWorkspaces, 40
  - getCampo, 41
  - operator=, 41
  - ParserProtocoloWorkspaces, 40, 41
- print
  - Object, 24
- printObj
  - Object, 24

- Proxy, 41
  - interrupt, 43
  - is\_finished, 43
  - operator=, 43
  - Proxy, 42, 43
  - receive, 43
  - send, 43
  - sendError, 44
  - sendOKWks, 44
  - sendOK, 44
- ProxyClient, 44
  - ~ProxyClient, 46
  - availableWks, 48
  - closeWks, 49
  - deleteWks, 49
  - execLobbyCMD, 46
  - execLocalCMD, 47
  - execRefresh, 47
  - getSlotObj, 48
  - goBack, 48
  - loadWks, 48
  - newWks, 49
  - operator=, 46
  - ProxyClient, 45, 46
  - run, 46
  - setCodeSegment, 47
  - setObjName, 47
  - showLobby, 47
  - swapMutability, 48
  - topObj, 49
- ProxyServer, 49
  - areThereErrors, 51
  - getErrors, 51
  - getFlag, 51
  - ProxyServer, 50
  - run, 51
  - sendCMDMessage, 51
  - sendCmdMessage, 51
  - setFlag, 51
- receive
  - Proxy, 43
  - Socket, 61
  - Workspace, 65
- receiveCode
  - Server, 56
- receiveMessage
  - Parser, 35
- recvMessage
  - Object, 29
- removeSlot
  - Object, 27
- run
  - Acceptor, 7
  - ProxyClient, 46
  - ProxyServer, 51
- SelectWkWindow, 52
  - addWidget, 53
  - configureTreeView, 53
  - drawWorkspaces, 53
  - getWindow, 53
  - SelectWkWindow, 53
  - updateList, 53
- send
  - Proxy, 43
  - Socket, 61
- sendCMDMessage
  - ProxyServer, 51
- sendCmdMessage
  - ProxyServer, 51
- sendError
  - Proxy, 44
- sendOKWks
  - Proxy, 44
- sendOK
  - Proxy, 44
- Server, 54
  - ~Server, 55
  - availableWorkspace, 55
  - closeWorkspace, 56
  - deleteWorkspace, 56
  - getLobby, 57
  - getObj, 57
  - getSlotObj, 59
  - getWorkspace, 55
  - loadWorkspace, 55
  - newWorkspace, 55
  - receiveCode, 56
  - Server, 55
  - setCodeSegment, 57
  - setObjName, 57
  - swapMutability, 59
  - workspaces, 59
- setCodeSegment
  - Morph, 16
  - Object, 28
  - ProxyClient, 47
  - Server, 57
- setCommand
  - Message, 9
- setFlag
  - ProxyServer, 51
- setLength
  - Message, 8
- setLobby
  - VirtualMachine, 64
- setMessage
  - Message, 9
- setName
  - Object, 28
- setObjName
  - Morph, 16
  - ProxyClient, 47
  - Server, 57
- setPrimitive
  - Object, 32

- showLobby
  - ProxyClient, [47](#)
- slot\_map
  - Object, [22](#)
- slot\_morph
  - Morph, [12](#)
- slot\_t
  - Object, [22](#)
- Socket, [60](#)
  - receive, [61](#)
  - send, [61](#)
  - Socket, [60](#), [61](#)
- stringObj
  - Parser, [36](#)
- swapMutability
  - ProxyClient, [48](#)
  - Server, [59](#)
- swapSlotMutability
  - Object, [31](#)
- Thread, [62](#)
- toString
  - Message, [8](#)
- topObj
  - ProxyClient, [49](#)
- updateList
  - SelectWkWindow, [53](#)
- VirtualMachine, [62](#)
  - createBoolean, [63](#)
  - createEmptyObject, [64](#)
  - createNil, [63](#)
  - createNumber, [63](#)
  - createString, [63](#)
  - findObjectById, [64](#)
  - setLobby, [64](#)
  - VirtualMachine, [63](#)
- Workspace, [64](#)
  - findObjectById, [65](#)
  - receive, [65](#)
- workspaces
  - Server, [59](#)