



UNIVERSIDAD CARLOS III DE MADRID
Ingeniería Informática

Diseño y desarrollo de un prototipo de Framework para laboratorios remotos

Implementación de las primeras herramientas y
funcionalidades



PROYECTO DE FIN DE CARRERA

Madrid, Octubre 2011

Autor: *Carlos A. Rodríguez Mecha*
Tutores: *Ramón I. Barber Castaño*
Javier Fernández Muñoz

Diseño y desarrollo de un prototipo de Framework para laboratorios remotos

Implementación de las primeras herramientas y funcionalidades

REMOTE LABORATORY FRAMEWORK

PROTOTYPE

Carlos A. Rodríguez Mecha

Remote Laboratory Framework por Carlos A. Rodríguez Mecha se encuentra bajo una Licencia Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.



*A mis padres, pues sin ellos nunca lo hubiera conseguido.
A Aitor, Antonio, Diego, Jose, Juan y Sergio, por aguantarme.
A Mayu, por estar siempre ahí.
A mis tutores, Ramón y Javier, por haberme guiado.
A Anabel, por contestar cada una de mis incisantes preguntas.
y al resto de mis amigos y compañeros por alegrarme estos seis años.*

“Si tú crees que no puedes conseguirlo, no te preocunes, no lo harás.”
Vancouver, 2010

Resumen

En el último medio siglo, la sociedad se ha visto abrumada por el avance de tecnologías que ya forman parte de la vida cotidiana. Es por esto que es imposible dar cabida a largo plazo a los modelos de trabajo e industria tradicionales. El cambio se está produciendo en la actualidad y por ese motivo surgen nuevas ideas para intentar adecuar estos modelos a la “sociedad de la información”. Las empresas y centros de investigación intentan dar “pasos de gigante” para innovar y mantenerse en la cima.

La evolución de estas tecnologías también implica que los problemas a los que se enfrentan estos modelos se modifiquen, desaparezcan e incluso se creen nuevos. Algunos como la capacidad de las herramientas han desaparecido por completo para dar paso a soluciones que aporten velocidad y potencia. Otros siguen aún sin resolverse, como las limitaciones físicas de localización que se presentan en la industria contemporanea.

El proyecto que se detalla a continuación pretende dar una solución concreta a este problema aplicado a la forma de enseñanza en centros tecnológicos y universidades, uniendo y utilizando los avances punteros que son accesibles para cualquier estudiante, desarrollador o ingeniero. Se trata de una solución versátil que permite salvar otros tipos de problemas derivados, así como ser utilizada para varios usos no contenidos en este documento, como es la domótica o investigación. En el caso que se ocupa se centrará en desarrollar una plataforma *online* que permita acceder y administrar de forma remota a un conjunto de recursos *hardware* y *software* distribuidos de forma transparente para el usuario final. Para ello, se manejarán conceptos como *comunicación*, *virtualización* y *bases de datos*.

A lo largo de las siguientes páginas se dará un repaso a las tecnologías que se han usado así como la evolución histórica de las mismas. También se hará incapié en los requisitos para el desarrollo y diseño del mismo.

Abstract

In the last half century, the society has been overwhelmed by the progress of technologies that are already part of everyday life. This is why it is impossible to accommodate long-term work patterns and traditional industry. The change is happening now and that is why new ideas to try to adapt these models to the “information society”. The companies and research centers are trying to “giant steps” to innovate and stay on top.

The evolution of these technologies also means that the problems faced by these models change, disappear and even create new. Some, like the ability of the tools have altogether disappeared to make way for solutions that provide speed and power. Others aren’t completely settled yet, as the physical limitations of location presented in contemporary industry.

The project described below is to provide a concrete solution to this problem applied to the form of teaching in universities and technological centers, connecting and using the pointer advances that are accessible to any student, developer or engineer. It is a versatile solution that can save other types of problems and be used for various purposes not contained in this document, such as home automation or research. In the present case will focus on developing a platform *online* to allow access and remotely manage a set of *hardware* and *software* resources distributed transparently to the end user. To this end, concepts such as *communication*, *virtualization* and *databases* will be handled.

Throughout the following pages will give an overview of the technologies used, as well as the historical evolution of the same. Emphasis will also be on the requirements for the design and development.

Índice general

1. Introducción	1
1.1. Introducción	2
1.2. Motivación del proyecto	2
1.3. Objetivos	3
1.4. Planificación	4
1.5. Contenido	5
2. Estado del arte	7
2.1. El pasado	8
2.1.1. Hasta 1930: La prehistoria	8
2.1.2. 1930 - 1945: El inicio	8
2.1.3. 1945 - 1960: La agencia <i>ARPA</i>	8
2.1.4. 1960 - 1970: Creación de <i>ARPANET</i>	9
2.1.5. 1970 - 1980: Evolución de la comunicación	9
2.1.6. 1980 - 1990: La era de los ordenadores personales	10
2.1.7. 1990 - 2000: La aparición de las <i>puntocom</i>	11
2.1.8. 2000 - 2010: El crecimiento exponencial	12
2.2. El presente	13
2.3. El futuro	14
2.4. Los sistemas y modelos de trabajo	15
2.4.1. Sistema individual	15

2.4.2. Sistemas centralizados	17
2.4.3. Sistemas distribuidos	18
2.4.4. Internet y la “nube”	18
2.4.5. Sistemas <i>clusters</i> y <i>grid</i>	19
2.5. Las plataformas virtuales	19
2.5.1. Java y su máquina virtual	20
2.6. El almacenamiento	22
2.6.1. Sistemas Gestores de Bases de Datos	22
2.6.2. SQLite	22
2.6.3. MySQL	22
2.7. La comunicación	24
2.7.1. El modelo TCP/IP	24
2.7.2. Sockets	26
2.7.3. Llamadas a procedimientos remotos	26
2.7.4. Java RMI	26
2.7.5. CORBA	27
2.7.6. Framework .NET	28
2.7.7. Servicios Web	28
2.8. Las limitaciones	29
2.8.1. La interoperabilidad	30
2.8.2. Acceso remoto	30
3. Análisis	31
3.1. Introducción	32
3.2. Descripción general	32
3.2.1. Capacidades generales	32
3.2.2. Restricciones generales	33
3.2.3. Características de los usuarios	33

3.2.4. Entorno operacional	33
3.3. Requisitos del usuario	34
3.3.1. Requisitos de capacidad	35
3.3.2. Requisitos de restricción	46
3.4. Casos de uso	48
3.5. Requisitos del <i>software</i>	52
3.5.1. Requisitos funcionales	52
3.5.2. Requisitos no funcionales	62
4. Diseño	67
4.1. Arquitectura del sistema	68
4.2. Diseño de componentes	70
4.2.1. Base de datos central	70
4.2.2. Servidor Web	70
4.2.3. Laboratorio	73
4.2.4. Datos de la herramienta	75
4.2.5. Herramienta	75
4.2.6. Gestión de laboratorios	77
4.2.7. Cliente	78
4.2.8. Comunicaciones	79
4.3. Despliegue de componentes	80
5. Pruebas	83
5.1. Configuraciones	84
5.2. Pruebas de verificación de funcionalidad	84
5.3. Pruebas de recuperación de errores	89
5.4. Pruebas de estrés	92
6. Problemas y mejoras	95

6.1.	Problemas encontrados en el desarrollo	96
6.1.1.	La sincronización	96
6.1.2.	La portabilidad de Java	96
6.1.3.	<i>Streaming</i> de vídeo y Java	97
6.1.4.	La tarjeta PCI-1711-BE	98
6.2.	Próximos pasos	99
6.2.1.	Próximas versiones de RLF	99
7.	Conclusiones	101
A.	Presupuesto	105
B.	Guía de despliegue	111
C.	Guía de mantenimiento	119
D.	Guía de desarrollo de herramientas	123
E.	Guía de usuario	131
F.	Material entregado	143
	Bibliografía	149
	Índice alfabético	150

Índice de figuras

1.1. Calendario de tareas	6
2.1. El ordenador “K3”	9
2.2. Esquema relacional de una base de datos	10
2.3. Velocidad de conexión	11
2.4. Entradas en los DNS	12
2.5. Evolución de la “nube”	14
2.6. OpenNebula	15
2.7. Componente domótico	16
2.8. Sistema individual	16
2.9. Android	17
2.10. Sistema centralizado	17
2.11. Sistema jerárquico	18
2.12. La “nube”	19
2.13. <i>Cluster</i>	20
2.14. Estructura de SQLite	23
2.15. Modelo TCP/IP	25
2.16. Estructura de Java RMI	27
2.17. Estructura de CORBA	28
2.18. Estructura de .NET	29
2.19. Servicios web	29

3.1. Casos de uso: Administrador y desarrollador	51
3.2. Casos de uso: Cliente	51
4.1. Arquitectura de RLF	69
4.2. E-R central	71
4.3. Servidor Web y Base de datos central.	72
4.4. El laboratorio y las herramientas.	73
4.5. E-R local	75
4.6. E-R de la herramienta	76
4.7. Gestor de laboratorios por capas	77
4.8. Gestor de laboratorios	77
4.9. Comunicaciones RLF	78
4.10. Cliente de escritorio	79
4.11. Protocolo RLF	79
4.12. Despliegue de RLF	81
5.1. RLF en Mac	91
6.1. Java.io VS Java.nio	97
6.2. Túnel SSH	99
6.3. Próximas líneas RLF	100

Índice de tablas

1.1. Planificación del proyecto	4
3.1. Entorno operacional del administrador.	34
3.2. Entorno operacional del desarrollador.	34
3.3. Entorno operacional del cliente.	34
5.1. Configuraciones de las pruebas.	84
A.1. Coste de personal	106
A.2. Coste material para el desarrollo	106
A.3. Coste total de desarrollo	107
A.4. Costes generales	107
A.5. Coste del proveedor y monitor	108
A.6. Coste por laboratorio	108
A.7. Costes de herramientas	109
A.8. Presupuesto de un sistema estándar	109

Capítulo 1

Introducción

La finalidad de este capítulo es dar a conocer el contenido de este libro, así como un primer encuentro con el proyecto. Tratado como una pequeña introducción, los conceptos novedosos serán explicados en capítulos posteriores.

1.1. Introducción

Este proyecto se inicia como parte de una solución a problemas concretos en la evolución tecnológica. Desde hace años, el mundo está inmerso en un constante cambio impulsado por el desarrollo de una parte de la ciencia, la cual sirve como base a innumerables campos. Tan amplio es el efecto, que no se concibe, en la actualidad, la vida sin ella. Ésta es la ciencia de la información.

Todo lo que implica una revolución como ésta se ve reflejado en el pensamiento de los nuevos científicos, ingenieros, estudiosos y empresarios, los cuales han tenido que adaptar sus ideas a esta nueva situación. Éstas son las que ahora, y en el futuro, resolverán los problemas que surgen como consecuencia directa del cambio sometido.

La idea (o proyecto), que en las posteriores páginas de este documento es explicada, no es más que una de las miles que intentan ayudar a esa evolución sin retorno, asegurando que será la base para otras soluciones. Se dará como nombre *Remote Laboratory Framework* (versión *Prototype*) a este proyecto.

Se empieza, entonces, con los motivos y características que han gestado esta idea, así como su análisis y diseño, y, por último, su implementación en la vida real.

1.2. Motivación del proyecto

La ciencia de la información ha contribuido al desarrollo de otros campos (no sólo en el ambiente tecnológico) y al aumento de necesidades concretas. Yendo a las partes más básicas de la vida cotidiana, la educación y el trabajo se han visto afectados directamente, siendo parte del problema y de la solución.

De tal modo es así que como al cabo de los años de estudio, un alumno se enfrenta a numerosos problemas relacionados con ambos campos, independientemente del tipo de estudios cursados. Las innumerables horas dedicadas al desarrollo y aplicación de los conocimientos adquiridos hacen que las ideas surjan como una respuesta lógica a estos mismos problemas.

Pero este proyecto se centrará en dos problemas concretos que hasta hace unos años, no tenían aparente solución:

El primero es la **dependencia** que se tiene en los modelos de educación y de trabajo actuales respecto a los lugares donde se desarrollan estas actividades. Los traslados pueden llegar a copar gran parte del tiempo en un estudiante o trabajador, el cual, la mayor parte es improductivo. Cabe decir que se realizan esfuerzos por parte de las organizaciones involucradas para intentar eliminar esta gran dependencia creando herramientas de comunicación y sistemas automatizados remotos. Siendo soluciones válidas, aún quedan muchos campos por tratar, como es el caso de las herramientas de trabajo electrónicas, desde ordenadores, máquinas y componentes mecánicos entre otros, dispuestos en un lugar concreto, donde el personal debe desplazarse obligatoriamente para poder usarlos. *El problema de la dependencia física de las herramientas de trabajo.*

El segundo problema se centra más en la **variedad** de estas herramientas. Cada empresa vende sus productos a estos centros y organizaciones cerrando el paso a la competencia, lo que conlleva a que cada uno de estos sea incompatible (en mayor o menor medida) con los desarrollados por otras empresas. Esto obliga a los estudiantes y trabajadores a aprender como se usa cada producto, incluso cuando su funcionalidad es la misma, con el gasto de tiempo y esfuerzo que esto requiere. Se

acentúa aún más en las herramientas *software*, donde, sin seguir ningún estándar, fuerzan al usuario a cambiar su forma de trabajo. *El problema de la heterogeneidad de los componentes tecnológicos.*

Cualquier intento de solventar el primer problema se enfrenta con el segundo de manera irremediable, obligando a que cada solución sea específica para un producto en concreto.

Paralelamente, el avance tecnológico desarrolla el concepto de “la gran red”, Internet, que se muestra como aliciente para solventar dichos problemas. Esta gran tecnología permite lidiar con el problema de la dependencia, y las herramientas surgidas de ella, como las grandes plataformas de trabajo apuntan al problema de la variedad.

A partir de aquí, se puede extender el concepto de este proyecto a otros campos sin aparente relación. En la ingeniería, con sistemas complejos compuestos por multitud de componentes electrónicos y tecnologías, es donde se invierten grandes fortunas en solventar estos dos mismos problemas. Incluso, al aportar los avances tecnológicos a campos donde anteriormente no se encontraban, como es el caso de la arquitectura y construcción, aparecen dichos problemas.

1.3. Objetivos

Apuntando a unos objetivos concretos y bien definidos, el proyecto tiene como finalidad solventar los dos problemas anteriormente descritos. Como se verá a lo largo de este documento, se considerará crear un sistema que permita principalmente:

- Acceder a un conjunto de recursos heterogéneos de manera remota, es decir, sin necesidad de estar físicamente donde se encuentran.
- Conseguir un sistema distribuido que sea visto como un único componente por parte del usuario.
- Incluir la posibilidad de ampliar el sistema de forma escalable sin que suponga el desarrollo desde cero.

Si se ponen en práctica estos objetivos, se obtendrá una *plataforma* que permita la eliminación de muchos sistemas dedicados, así como su ahorro, en ambientes de trabajo donde se requiera utilizar recursos físicos electrónicos, aplicándose también a recursos didácticos en universidades o escuelas. De este modo se minimizará el tiempo que se requiere para su uso y el tiempo que estas herramientas permanecen *ociosas*. Para cumplir estos objetivos, es necesario atacar un conjunto metas más desgranadas que atienden a objetivos sencundarios.

- Permitir al usuario manejarlos de igual manera que si lo hiciera de forma local.
- Dar cabida a recursos de diferentes tipos, transformando esa heterogeneidad en una aparente homogeneidad.
- Eliminar las dependencias de los recursos en cuanto a requisitos propios de la plataforma, es decir, reducir al mínimo las especificaciones de funcionamiento de cada recurso.
- Automatizar todo el proceso de la utilización de los recursos, de forma que no requiera de intervención externa para el funcionamiento normal del sistema, lo que hará que se pueda utilizar en cualquier momento, incluso fuera de horarios de trabajo o lectivos.
- Ayudar a que las posteriores modificaciones del sistema sean de forma controlada y manteniendo, en la medida de lo posible, la coherencia del proyecto original.

Desde el punto de vista de la creación de nuevos recursos, se establecieron unas normas y requisitos para poder salvar esa heterogeneidad.

Como conclusión, se dirá que se ha creado un sistema distribuido manejado por un *middleware*¹ que a la vez se comporte como un *framework*² para la creación, distribución y uso de nuevos recursos tecnológicos.

1.4. Planificación

Debido a la gran cantidad de trabajo que supone el desarrollo de este proyecto, es necesario establecer una planificación *a priori* que defina cómo abordar tales problemas.

Siendo la parte más importante el análisis y diseño, se realizará en primer lugar la recolección de requisitos definidos en primera instancia por el cliente y los usuarios, con su posterior tratamiento y desarrollo.

En segundo lugar el diseño completo del sistema, a partir de dichos requisitos dará lugar a las especificaciones necesarias para empezar el desarrollo del mismo.

Tarea	Etapa	Horas estimadas
Planteamiento inicial	Análisis y diseño	16
Aceptación del proyecto	Análisis y diseño	4
Primera toma de requisitos	Análisis y diseño	8
Revisión de requisitos	Análisis y diseño	8
Toma de requisitos	Análisis y diseño	20
Aceptación de requisitos	Análisis y diseño	8
Investigación y Diseño	Análisis y diseño	120
Aceptación del diseño	Análisis y diseño	4
Desarrollo	Desarrollo	1100
Verificación y pruebas	Pruebas	40
Aceptación del sistema	Pruebas	4
Documentación	Documentación	116
Presentación del proyecto	Presentación	20
Total		1468

Tabla 1.1: Planificación de las tareas.

A continuación, se procederá a poner en práctica el diseño obtenido, siendo fieles a lo anteriormente establecido. Esto comprenderá desde el desarrollo del código necesario hasta las posteriores remodelaciones.

Después, con las primeras versiones del sistema, se procederá a realizar los *test* necesarios que verifiquen el buen funcionamiento del mismo, en diversas situaciones y sobre múltiples estados. Debido a que en esta etapa puede surgir la necesidad de modificar el desarrollo del mismo, es

¹Capa de *software* que se ejecuta sobre el sistema operativo que permite manejar una red de computadoras como un sistema único [24]

²(Plataforma, entorno, marco de trabajo). Desde el punto de vista del desarrollo de *software*, es una estructura de soporte definida, en la cual otro proyecto de *software* puede ser organizado y desarrollado. [2]

posible que se deba retroceder de forma cíclica para poder solventar los problemas y errores que puedan aparecer.

Además, realizarán múltiples revisiones sobre la documentación creada durante todo el proceso para establecer su versión final.

Por último, y con la versión final, tanto del sistema como de la documentación, se expondrá el conjunto de los productos en una presentación donde asistirá el cliente, los tutores que supervisarán el desarrollo de todo el proyecto y un consejo de evaluación.

El recuento de horas se muestra en la tabla 1.1 y calendario propuesto para estas tareas, representado en un diagrama de Gantt se encuentra en la figura 1.1 al final del capítulo.

1.5. Contenido

La documentación que aquí se presenta viene estructurada de forma que siga los mismos pasos de la planificación anteriormente descrita:

Capítulo 1 - Introducción: Sirve como ayuda para poder comprender la motivación de este proyecto, así como los objetivos que se desean alcanzar.

Capítulo 2 - Estado arte: Este capítulo de especial importancia da una visión general de las tecnologías usadas, así como la evolución que ha permitido el desarrollo del sistema. En las diferentes secciones se explicará cada concepto de importancia utilizado en este.

Capítulo 3 - Análisis: Mostrará el conjunto de los requisitos y especificaciones concretas que son necesarias para poder llevar a cabo correctamente esta idea.

Capítulo 4 - Diseño: Agrupa todas las soluciones propuestas a las anteriores especificaciones, así como la forma de actuar para desarrollar el proyecto.

Capítulo 5 - Pruebas: Especifica las diferentes situaciones controladas a las que se someterá el sistema para comprobar el correcto funcionamiento, así como los resultados obtenidos.

Capítulo 6 - Problemas y conclusión: Se especificarán los problemas más importantes que se han encontrado en el desarrollo del proyecto y una breve conclusión.

Además, se incluyen unos anexos para la correcta utilización y comprensión del sistema.

Anexo A - Presupuesto: Lista detallada de los costes que supone implantar y mantener el sistema.

Anexo B - Guía de despliegue: Instrucciones que se deberán llevar a cabo para poder desplegar completamente el sistema en un ambiente de trabajo.

Anexo C - Guía de mantenimiento: Conjunto de directrices de mantenimiento para administraciones de la plataforma.

Anexo D - Guía de desarrollo: Primeras pautas para crear o modificar módulos del propio sistema así como la creación de nuevas herramientas.

Anexo E - Guía del usuario: Destinado al usuario final de la aplicación de escritorio y *web*.

Anexo F - Material entregado: Lista exhaustiva de todos los componentes, guías y herramientas entregados a la finalización del proyecto en su primer prototipo.

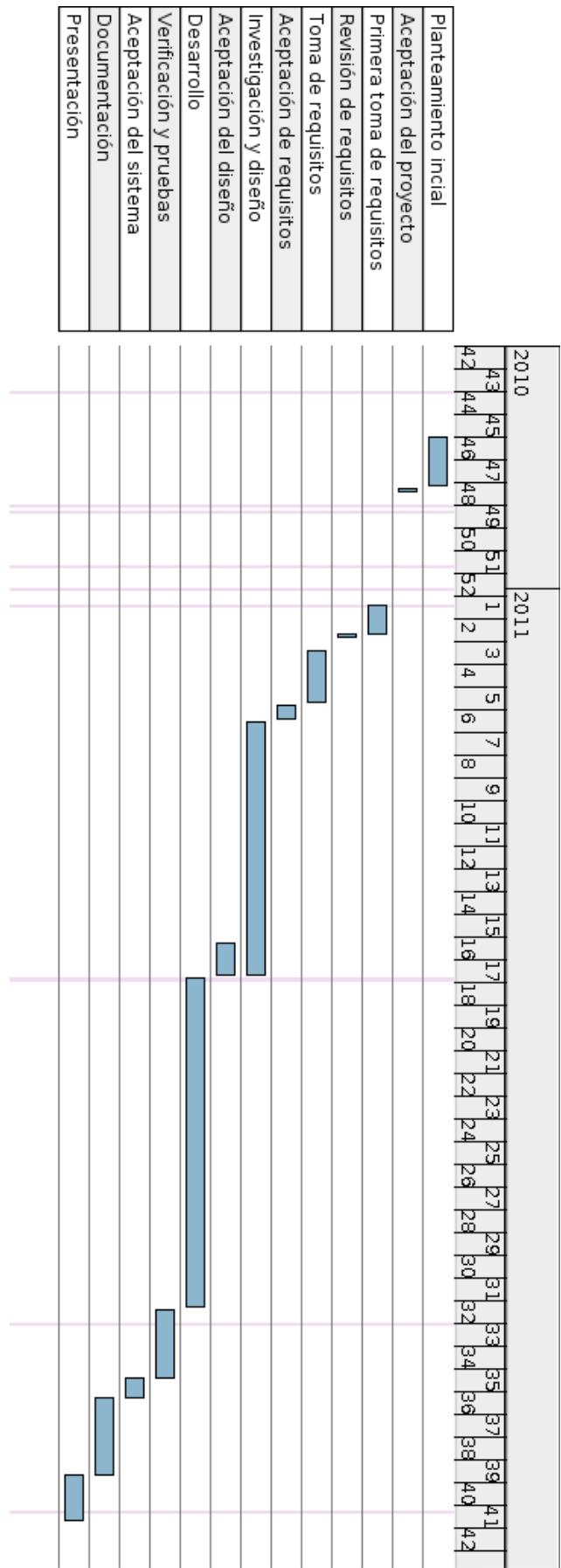


Figura 1.1: Calendario por semanas de las tareas realizadas.

Capítulo 2

Estado del arte

Para entender las técnicas y diseño que se van a utilizar en todo el proyecto, se incluye un repaso por la historia de las redes de ordenadores y las bases de datos, así como la evolución de los sistemas de trabajo y de información. También se mostrarán algunas de las características más importantes de todos los recursos utilizados para el desarrollo.

2.1. El pasado

Desde que Alan Turing (1912-1954) definiera la máquina con su propio nombre [25] en 1936, el mundo de la informática o ciencia de la información, ha sufrido un crecimiento a marchas forzadas de una forma súbita comparada con el resto de las ciencias. En el momento que se concreta la definición de algoritmo¹ comienza la base de la era de la denominada “sociedad de la información”.

2.1.1. Hasta 1930: La prehistoria

Sin que se definiera aún esta ciencia, toda la atención estaba centrada en resolver sencillos problemas matemáticos de forma automática. Mediante válvulas y tubos de vacío, se realizaban experimentos con ondas de radio que emulaban sistemas [4].

Operaciones tan básicas como las puertas lógicas que sirven como base a los paradigmas informáticos se crearon en 1924². Con estas herramientas empezó a surgir el concepto de lenguajes de programación (aún no con ese nombre) que eran necesarios para traducir un lenguaje formal matemático a un conjunto de instrucciones o componentes que definieran una *estructura* electrónica que desempeñara una determinada función. Los estudios se centraban en solventar los costes de crear máquinas especializadas en una única tarea. Se registran los laboratorios Bell, con la idea de buscar soluciones electrónicas para cálculos básicos matemáticos. Gracias a esto, Vannervar Bush pudo completar su máquina que resolvió por primera vez ecuaciones diferenciales de manera autónoma.

2.1.2. 1930 - 1945: El inicio

Kurt Gödel se adentró en los lenguajes formales anteriormente comentados [9], dando paso a que el propio Turing inventara la primera máquina programable capaz de realizar diferentes acciones según las instrucciones administradas. Con ello, el matemático pudo realizar el primer juego de ajedrez donde el adversario era una máquina completamente autónoma. El cálculo de cada jugada podía durar varias horas.

En 1941, se inventa, por parte de Konrad Zuse, la computadora programable y completamente automática. Se llamó “K3” (véase figura 2.1 y los ordenadores actuales tienen muchas similitudes con ella). A partir de ese año, las sucesiones de computadoras creadas por parte de grupos de científicos y matemáticos aumentaron en capacidad y redujeron sus limitaciones, con lo que se encontraron nuevos problemas que solventar, aunque seguían siendo máquinas que no se podían comunicar unas con otras.

2.1.3. 1945 - 1960: La agencia ARPA

En estos años Turing realiza un estudio sobre la comunicación en las computadoras, pero en este caso, atendiendo a la interacción con los asistentes humanos. Fue uno de los primeros pasos para entender el avance que ocurrió como consecuencia de la guerra fría, la creación de ARPA³, la

¹Conjunto de reglas que expresa la manera de resolver un problema o un tipo específico de problemas, en un número finito de pasos o cálculos de expresiones [19].

²Walther Bothe construyó una puerta lógica AND para usarla en experimentos físicos, por lo cual recibió el premio Nobel de Física en 1954[4].

³Agencia de Proyectos de Investigación Avanzada. Organización creada para investigación de defensa que originalmente sólo disponía de una pequeña oficina, sin investigadores ni científicos [24].

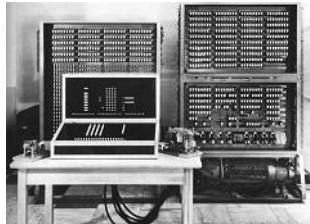


Figura 2.1: El equipo de Konrad Zuse podía almacenar hasta 64 palabras.

organización que más adelante diera a conocer uno de los inventos más importantes del siglo XX, la red de ordenadores.

La empresa IBM empezó a crear las computadoras de forma industrial, ya programables, y estandarizó el lenguaje ensamblador [18] como forma de programación para sus máquinas. A la vez, lenguajes como Fortran y Cobol, que aún se pueden utilizar, salieron a la luz como sustitutos para este engorroso lenguaje, que no era escalable y mucho menos portable, ya que cada *software* estaba realizado para una máquina con unas especificaciones físicas muy concretas. Estos lenguajes utilizan un compilador⁴ desarrollado generalmente por la misma empresa que provee el conjunto de herramientas para el desarrollo de aplicaciones.

2.1.4. 1960 - 1970: Creación de ARPANET

Los inventos fueron sucediéndose de año en año, aumentando la capacidad de cada máquina y reduciendo sus costes. Aún era pronto para que aparecieran los ordenadores personales, pero el número de computadoras creció, y con ello algunas necesidades como la interacción entre las mismas. Es aquí cuando por parte de la organización ARPA se aglutinaron algunos de los primeros conceptos que iban apareciendo sobre las redes de ordenadores. También se aplica, en 1969, el primer protocolo de comunicación en redes, complejo conocido como NCP⁵. Al final de la década, la fibra óptica se empieza a utilizar para comunicaciones en redes militares que aumenta de forma considerable la velocidad de las mismas.

Por otro lado, se dan las primeras pinceladas a los conceptos de programación estructurada, en la que actualmente se basan los lenguajes más modernos y potentes. El mismo grupo que creó años antes el lenguaje Cobol, introduce en el mercado la primera solución al almacenamiento masivo de datos en formato electrónico, el IDS (Almacén de Datos Integrado [13]). Desde ese momento, las bases de datos y las redes estuvieron estrechamente unidas.

2.1.5. 1970 - 1980: Evolución de la comunicación

Una vez la base de las comunicaciones fue creada, los distintos grupos de investigación aplicaron todas las técnicas aprendidas a los sistemas en red. Esto suponía que los datos en bruto no era lo único que se compartía, sino conjuntos de datos coherentes y estructurados. Empezaron a enviarse ficheros enteros con protocolos como FTP (*File Transfer Protocol*) y se crearon almacenes de datos a los que se accedía por medio de una red. Modelos tan importantes actualmente como TCP/IP⁶ en los que se basa Internet aparecieron como respuesta al problema que se presentaba a los

⁴Programa especial que convierte instrucciones a alto nivel en código máquina o ensamblador [18].

⁵*Net Control Protocol*. Conjunto de primitivas que permiten la realización de tareas de comunicación a alto nivel [24].

⁶Arquitectura que se cimenta en los dos protocolos TCP e IP, que hace posible la conexión entre múltiples redes heterogéneas [24]. (Ver página 24).

desarrolladores para implementar comunicaciones entre ordenadores. El primer correo electrónico llegó a su destino gracias a Ray Tomlinson en 1971.

Por parte de IBM, en esta década se crearon las bases de datos relacionales, que, al contrario que sus predecesoras, permiten la navegación por medio de enlaces entre los mismos, como se muestra en la figura 2.2. Después se estandarizó el lenguaje SQL,⁷ que se conserva hasta nuestros días con ligeros cambios.

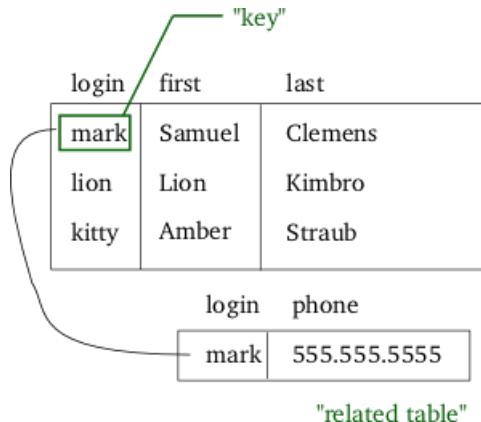


Figura 2.2: Los esquemas relacionales permiten enlazar unos datos con otros.

Los ordenadores se podía comunicar sin dificultades gracias a la aparición de *Ethernet* que utilizaba un cable único para unir varias computadoras de forma local. También, Microsoft y Apple se crearon con el fin de proveer los primeros ordenadores personales. El número de máquinas creció de forma imprevista hasta para estas empresas [17].

2.1.6. 1980 - 1990: La era de los ordenadores personales

Desde el punto de vista de las redes, esta década fue la época del afianzamiento de las tecnologías anteriormente desarrolladas. Aparecieron sistemas que posibilitaban una mayor transparencia en las comunicaciones, como los servidores DNS⁸ que permitían localizar otras máquinas a partir de un identificador más “amigable” que un número de 128 bits.

En 1983, ARPANET se separa de los fines militares para pasar las redes al ámbito civil. Para muchos, esta fue la creación de Internet. La aparición de lenguajes y tecnologías sufrió un crecimiento exponencial, apareciendo organizaciones como GNU⁹, Cisco Systems, Adobe, etc. Surgen, también, los estándares que servirán para el desarrollo de las interfaces de comunicación actuales, como XML (*eXtended Markup Language*).

Con el auge de la nueva forma de programación orientada a objetos, se crean las nuevas bases de datos, que establecieron todo su contenido como entidades coherentes. Esto sirvió también para una optimización en la utilización de los enlaces o claves entre datos.

⁷Structured Query Language. Lenguaje creado para el manejo y la búsqueda de datos en los gestores de bases de datos relacionales.

⁸Sistema de Nombres de Dominio. Organiza las máquinas dentro de dominios y hace una correspondencia de un nombre con la dirección de un *host*[24].

⁹GNU is Not Unix. Proyecto para crear un sistema operativo completamente libre bajo licencia GPL [22].

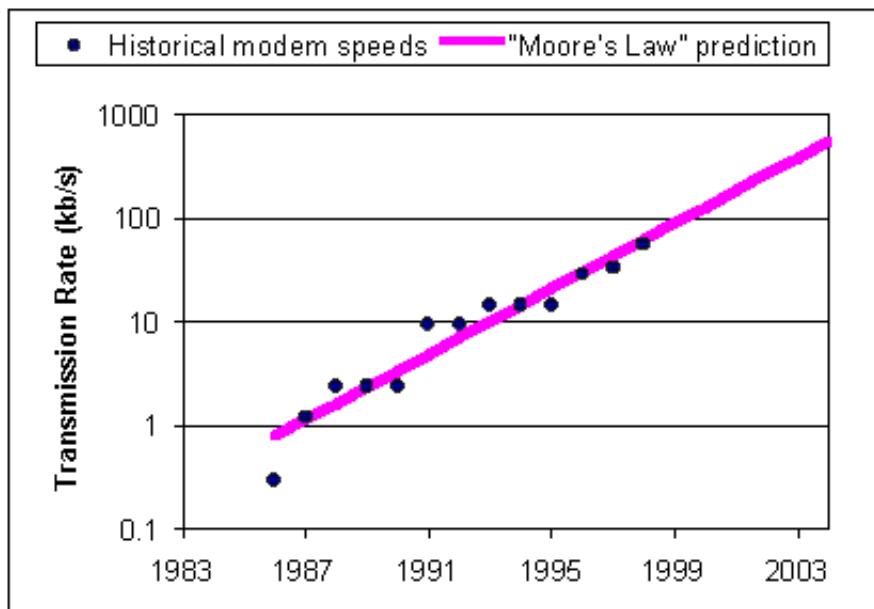


Figura 2.3: Velocidad de conexión comparada con la ley de Moore [10].

2.1.7. 1990 - 2000: La aparición de las *puntocom*

La inversión necesaria para poder tener un ordenador personal se fue reduciendo al cabo del tiempo, haciendo posible que las familias pudieran disponer de una o varias máquinas en su propio hogar.

El *Word Wide Web* (www) se crea como una nueva forma de entender , donde usuarios y empresas pueden publicar su información de manera sencilla y accesible desde cualquier parte del mundo con acceso a una línea telefónica. Es aquí cuando el modelo de trabajo de las empresas y la forma de enseñanza empieza a modificarse. Aparecen por primera vez (aunque de forma rudimentaria) las plataformas *online*, que permiten acceder a información y procesos de forma remota desde un ordenador personal, como son *RPC*, y *CORBA* de las que se hablará en la sección 2.7.

Muchas empresas aprovechan este “boom” para modificar sus modelos de negocio y aparecer en el escaparate mundial. Se publican portales donde compartir información de forma anónima, con soporte para realizar preguntas y contestarlas sobre todos los aspectos del conocimiento (web social). La información pasa a ser parte de la sociedad de manera pública y donde se acuñarán términos como “la nube” y los “sistemas distribuidos”. La cantidad de información contenida en Internet empieza a ser imposible de tratar con el formato actual y se plantea hacer cambios estructurales para incorporar la Web Semántica, y la Web 2.0. Sun Microsystems crea su máquina virtual en 1995.

Durante este periodo, las limitaciones de velocidad y capacidad se van reduciendo de manera exponencial (como se muestra en la figura 2.3), lo que permite que la información que se puede compartir aumente y varíe, en forma de vídeos, música, etc. Las bases de datos se convierten en almacenes gestionados automáticamente, llamados *Data Warehouse* que supone otro escalón más para el negocio y las transacciones monetarias por Internet.

Además, aparecen nuevos dispositivos de comunicación inalámbrica, como el *Bluetooth*¹⁰, *Wifi*¹¹ y GPRS,¹² que permiten ampliar aún más la difusión de Internet y solventar más limitaciones físicas. Podemos ver en la figura 2.4 la evolución en la década de los noventa del número de entradas DNS que era accesible a través de Internet.

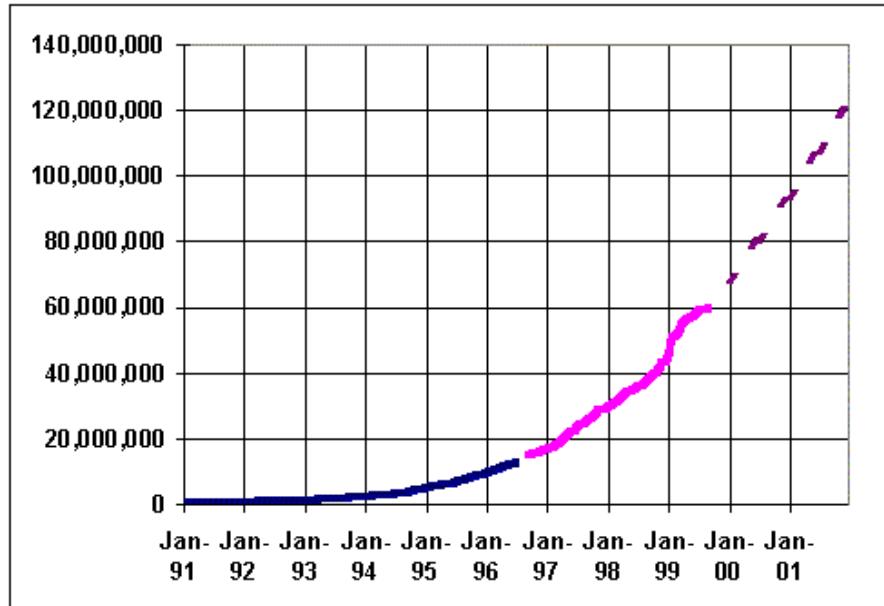


Figura 2.4: Número de entradas en los DNS públicos [10].

2.1.8. 2000 - 2010: El crecimiento exponencial

No sólo ha crecido el número de nodos de red en Internet, sino también la forma de concebir los sistemas de computación y almacenamiento de datos. Es cuando se crean las grandes granjas de ordenadores, se empiezan a sustituir los supercomputadores por múltiples ordenadores personales conectados en red que abaratán los costes de mantenimiento y escalabilidad. Aparece la nueva estructura de bases de datos en la “nube” para compartir información remotamente y de forma transparente. Con ellas se desarrollan *APIs*¹³ de código libre para que las distintas plataformas puedan acceder a esas bases de datos desde cualquier parte del mundo.

Además, evolucionan los pequeños dispositivos portátiles, con gran capacidad de cálculo y almacenamiento, que, mediante baterías, permiten una autonomía de muchas horas, sin necesidad de cables ni otras limitaciones físicas. Para muchos, estos dispositivos podrán llegar a sustituir a los ordenadores como se conocen en la actualidad, reduciendo su tamaño y aumentando su velocidad y potencia.

Una nueva forma de trabajo a distancia se empieza a poner en práctica en algunas administraciones y empresas, el “teletrabajo”. Para ello se utilizan sistemas de acceso remoto, de los que se ha hablado antes. En el *Valle del Silicio*, en California (Estados Unidos), muchas empresas tecnológicas apuestan por dar a sus empleados un día a la semana, en el que pueden trabajar a distancia desde sus casas, aumentando la satisfacción de los trabajadores.

¹⁰Red inalámbrica de corto alcance creada por el consorcio de L.M. Ericsson, IBM, Intel, Nokia y Toshiba [24].

¹¹Red inalámbrica LAN que utiliza protocolos derivados del 802.11 [24].

¹²Servicio de Radio de Paquetes Generales. Es la generación 2.5 de los sistemas inalámbricos en móviles. Los paquetes van por encima de las redes GSM [24].

¹³Interfaz de Programación de Aplicaciones. Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizada por otro *software* [24].

Como una consecuencia lógica del aumento de máquinas, tanto portátiles como fijas, de diferentes empresas y con diferentes soluciones, se crea un nuevo problema, la heterogeneidad de la forma de trabajar de estos dispositivos.

2.2. El presente

Todas las miradas están centradas en la infraestructura del *cloud computing* o “nube” . Ningún organismo oficial, negocio, centro de enseñanza o de investigación se puede concebir sin la ayuda de esta tecnología, que ahorra cantidades muy importantes de capital en personal y recursos físicos. Por otro lado, se invierten grandes fortunas en el desarrollo e implantación de nuevas soluciones empleando este sistema.

Aparecen lo que se conoce como los sistemas SOI (Infraestructuras Orientadas a Servicio) que solucionan las nuevas exigencias de flexibilidad y cambio que necesitan los modelos de negocio. Implementan arquitecturas orientadas a servicio (SOA) que contienen componentes (tanto físicos como lógicos) fácilmente reemplazables y que aportan homogeneización y escalabilidad. Partiendo de una plataforma rígida, como se encuentran en los sistemas tradicionales, se debe conseguir una respuesta “on demand” situando el servicio como cliente.

La desventaja más importante de estos sistemas es la definición de un mapa claro de dependencias entre componentes (debido a su complejidad), lo que aporta la necesidad de predecir de forma más concreta la carga que tiene cada componente. Es por esto que entra en juego el reparto de carga dinámico que viene siendo uno de los grandes problemas a los que se enfrentan los desarrolladores e investigadores en esta época. Como solución parcial, se añaden sistemas de monitorización para obtener información en tiempo real del conjunto del sistema. Se verá más adelante que en este mismo proyecto, se han utilizado estas herramientas.

Es por esto que casi todas las grandes compañías de informática están poniendo cada vez más empeño en ofrecer más y mejores recursos *cloud computing*: SalesForge.com (Forge.com) Amazon (Amazon EC2 y S3), Microsoft(Windows Azure), Google (Google Apps Engine), IBM (Smart-Cloud)... Se puede ver en la figura 2.5 la tendencia por adaptar las nuevas tecnologías surgidas de la “nube” .

Es aquí donde toma importancia el concepto de *virtualización* . En los sistemas tradicionales, el *hardware* limitaba los componentes *software* a utilizar, pero gracias a la virtualización, la parte física de los sistemas pasa a un segundo plano, siendo invisible para los desarrolladores de soluciones. Aunque a primera vista suponga una gran carga para estos sistemas, ya que *aplana* las características de cada *hardware* , a la larga aporta una gran ventaja a la hora de modificarlos. Siendo un concepto muy importante en este proyecto, sólo se podrá tratar por encima, ya que el sistema de virtualización usado en el mismo es sencillo debido a la finalidad del mismo.

El dinero invertido en estas nuevas tecnologías suele venir de capitales privados con el apoyo de grandes empresas, pero se está viendo el aumento de los productos *open source*¹⁴ que son mantenidos en parte por universidades y desarrolladores anónimos. Es decir, la “nube” es accesible incluso a empresas y comunidades de poco capital. Muchos de los servicios se pueden obtener (en versión reducida generalmente) gratis, como es el caso del almacenamiento en *Dropbox*, la gestión de proyectos en *Github*, publicación web en *Godaddy*, etc. Por tanto, los servicios son tanto B2C (de negocio para clientes) y B2B (de negocio para negocios, término referido también al *outsourcing*).

Uno de los proyectos más ambiciosos *open source* para la creación de una plataforma *cloud computing* completamente gratis se llama OpenNebula , desarrollado en parte por investigadores de

¹⁴ “Código abierto es el término con el que se conoce al *software* distribuido y desarrollado libremente.” [22]

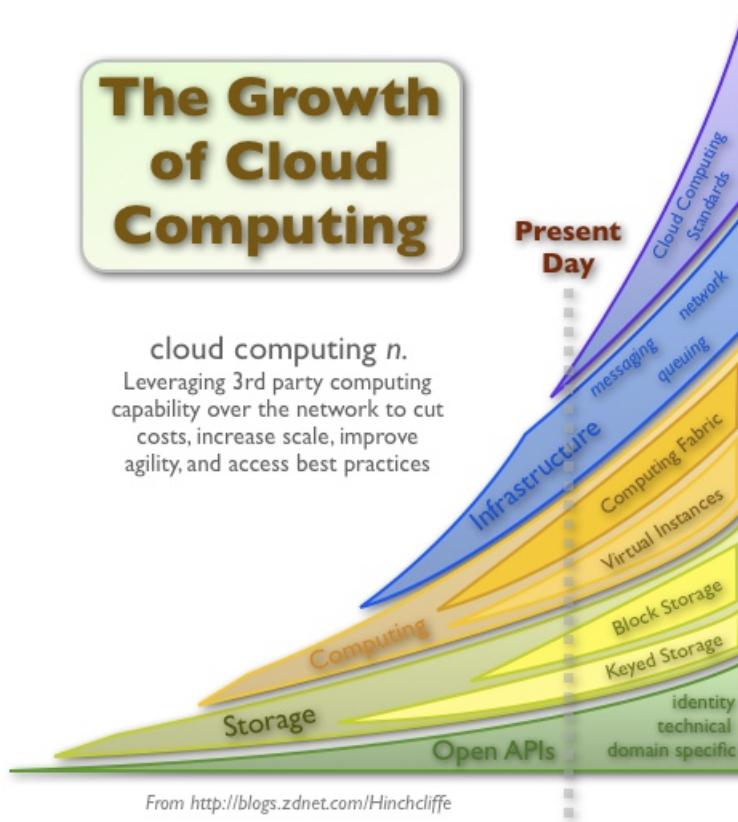


Figura 2.5: El desarrollo de la “nube” es la gran apuesta de esta década.

la Universidad Complutense de Madrid [5]. Este sistema provee al usuario, mediante un conjunto de herramientas (tanto comerciales como libres) de su propia “nube” accesible desde cualquier parte del mundo (véase figura 2.6).

Haciendo referencia a este proyecto, en la actualidad no existe ningún producto que atienda todos los requisitos y objetivos propuestos en este documento. Las plataformas más parecidas, desarrolladas por empresas para su propio uso, como es el caso de la plataforma de programas de trabajo de Telefónica sólo proveen un entorno *software*.

2.3. El futuro

Es difícil predecir el futuro teniendo en cuenta la velocidad que ha tomado la evolución de la tecnología en este último medio siglo, aunque está claro que el desarrollo de Internet será la pieza clave que mueva la sociedad de la información.

Las limitaciones físicas que ahora existen serán un mero recuerdo y los nuevos tipos de computadores, como los biológicos o los cuánticos sustituirán a los ordenadores personales actuales, reducidos en tamaño y aumentados en capacidad y potencia. La cantidad de información que se pueda almacenar no será más un problema y los esfuerzos se centrarán en la velocidad y optimización.

Ya ha comenzado la era de la domótica, donde las propias casas son elementos integrados en sistemas complejos conectados a Internet, automáticos e independientes. En realidad, la domótica

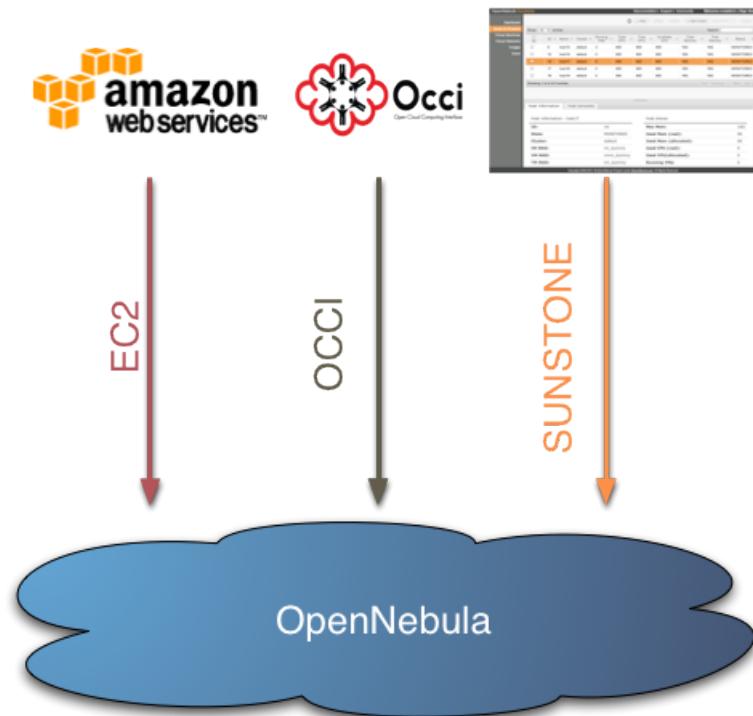


Figura 2.6: OpenNebula utiliza servicios comerciales como soporte.

no es más que manejar *hardware* de forma remota. En la figura 2.7 se muestra un dispositivo de control de *hardware* acoplado a una casa.

2.4. Los sistemas y modelos de trabajo

Al igual que las arquitecturas de las redes de ordenadores, las plataformas de trabajo convencional se pueden clasificar dependiendo de su distribución y sus tipos de comunicación. A continuación, se establecen los tipos básicos de sistemas que serán clave para el entendimiento de este proyecto.

2.4.1. Sistema individual

Se puede considerar como una red de un solo nodo como se puede ver en la figura 2.8. Es aquel sistema donde no existe comunicación y todo depende de un mismo elemento. Es así como se ha trabajado hasta antes de la revolución de la sociedad de la información de forma tecnológica.

Los dispositivos portátiles son un caso especial de un sistema individual, sólo que permite el desplazamiento del propio nodo entre otras redes, por lo que puede salir de esta categoría y aplicarse a las siguientes. Actualmente se consideran sistemas portátiles también a los *Smartphones*.



Figura 2.7: La domótica puede ser un componente más en la construcción inmobiliaria.



Figura 2.8: Máquinas en un sistema individual.

¹⁵, *Tablets* ¹⁶ y *Netbooks* ¹⁷. En la última década, las empresas que tradicionalmente se dedicaban a los ordenadores personales, han cambiado su modelo de producción a estos dispositivos portátiles, desarrollando plataformas completas y *software* específico para ellas.

Las grandes empresas se han abierto camino en estos dispositivos gracias a los nuevos sistemas operativos Android (figura 2.9), Windows Phone 7 e iOS, de Google, Microsoft y Windows respectivamente. El resto se ha ido desplazando para ocupar una cuota mínima en el mercado actual, debido a su falta de apoyos por la comunidad de desarrolladores, como son Symbian de Nokia, Palm de ahora HP y el Sistema RIM de los dispositivos Blackberry.

¹⁵Teléfono móvil con funciones avanzadas de gestión de programas y acceso a Internet.

¹⁶Dispositivo portátil de mayor capacidad y tamaño que un móvil, generalmente para uso multimedia e Internet, aunque cada vez más para acciones más complejas.

¹⁷Ordenadores de tamaño reducido con una gran autonomía debido a la batería que poseen, pensados para trabajos ligeros y el acceso a Internet



Figura 2.9: El sistema operativo Android está disponible para *tablets* y *smartphones*.

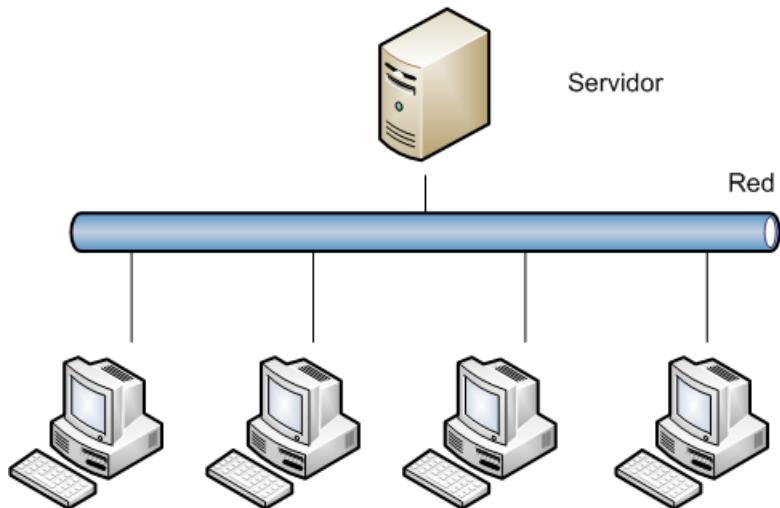


Figura 2.10: Esquema de un sistema centralizado.

2.4.2. Sistemas centralizados

Son aquellos en los que los nodos dependen de otros especiales que administran y manejan toda la comunicación. En terminología técnica se consideran como los paradigmas de Cliente/Servidor (figura 2.10, varios nodos o clientes se conectan a un servidor para solicitar servicios) [24]. Generalmente, los nodos centrales son de mayor tamaño o capacidad, y también tienen una mayor responsabilidad. Sin ellos, la comunicación no existe y, por lo tanto, el desarrollo es imposible. Es aquí donde se permite compartir, además de datos, *hardware* y recursos que se ponen a disposición de los nodos del sistema.

También es donde aparecen los sistemas por niveles, que son la evolución propia de los sistemas centralizados, donde un nodo central de un sistema es un nodo cliente más en un nivel superior, como podemos ver en la figura 2.11.

El problema más típico de estos sistemas es su falta de robustez, siendo dependiente de un sistema central. Además, la escalabilidad está limitada por la potencia de estos nodos centrales, que tienen que atender todas las peticiones.

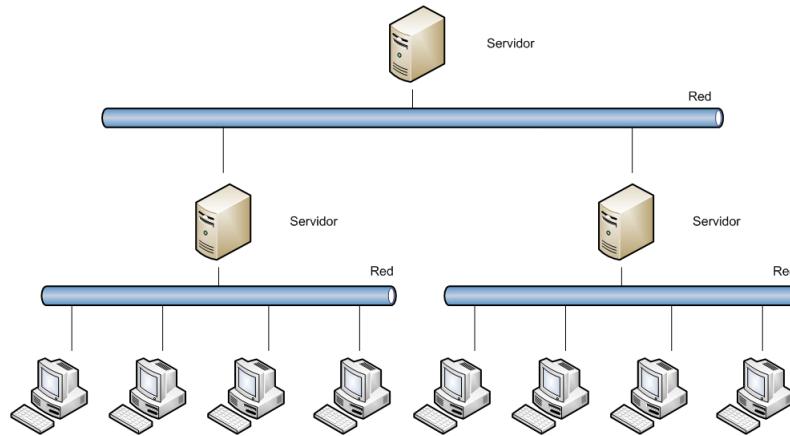


Figura 2.11: Representación de un sistema centralizado por niveles.

2.4.3. Sistemas distribuidos

Suponen la evolución a los sistemas centralizados. Son un conjunto de nodos donde, de forma transparente, funcionan como uno único [24]. Su escalabilidad es mucho mayor que los sistemas centralizados, debido a la homogeneidad de roles. Para poder controlar estas arquitecturas, aparecen el . Estas plataformas no pueden concebirse sin los términos *virtualización* y *comunicación*.

Estos sistemas ofrecen unas características que se listan a continuación [14].

Ventajas:

- Reducción del coste del computador y del acceso a la red.
- Compartición de recursos.
- Escalabilidad.
- Tolerancia a fallos. Al no depender de un sólo sistema central, existe la posibilidad que después de que ocurra un fallo, el conjunto del sistema no se vea afectado.

Inconvenientes:

- Múltiples puntos de fallo.
- Seguridad. Considerado como el gran inconveniente en estos sistemas debido a los innumerables puntos de acceso.

2.4.4. Internet y la “nube”

Comprende múltiples sistemas centralizados y distribuidos, organizados de manera jerárquica que ofrecen servicios a los nodos de la red. Desde la descarga de información, hasta la compartición, uso y publicación de la misma. El término “nube” aparece como referencia a los servicios (generalmente almacenaje de información) en Internet, que se proveen, siendo transparente al usuario donde se encuentran, aunque disponible para su acceso en todo momento. La limitación de capacidad pasa a un segundo plano ya que entran en juego las granjas de ordenadores anteriormente mencionadas [10].

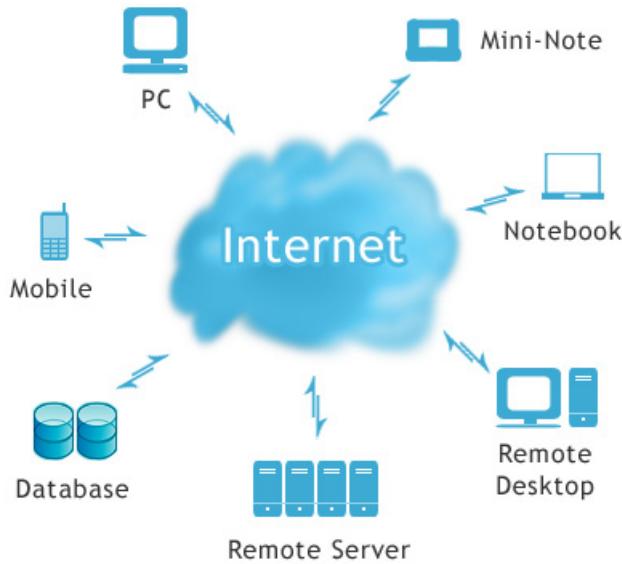


Figura 2.12: La “nube” provee servicios a multitud de dispositivos.

Esta “nube” es accesible desde cualquier dispositivo con conexión a Internet (figura 2.12, lo que hace que sea necesario disponer de plataformas que acepten varios tipos de formatos, tanto estándar como privativos. Esto ha supuesto un gran avance para homogeneizarlas y poder obtener servicios sin preocuparse de restricciones y limitaciones. Está compuesta por capas que son intermediarios para acceder a las máquinas finales.

2.4.5. Sistemas *clusters* y *grid*

Son casos especiales de sistemas centralizados, en los que el servicio principal es la capacidad de cálculo. Están formados por nodos genéricos baratos, fácilmente reemplazables. Los *grids* son agrupaciones de *clusters* (como los de la figura 2.13) accesibles a través de Internet, poniéndolos a disposición para que realicen una tarea por nodo.

Actualmente muchas empresas ofrecen servicios de *clusters* y *grid* para otras empresas, que suponen un gran ahorro. Está estimado que los negocios que contratan estos servicios pueden ahorrarse hasta un millón de dólares al año [1].

2.5. Las plataformas virtuales

Debido a la extensión que supondría comentar todas las plataformas de desarrollo actuales (o las usadas en el proyecto), sólo se pondrá atención a aquellas que formen parte del sistema a emplear en el proyecto, es decir, las plataformas virtuales.

La estrategia de la virtualización está compuesta por múltiples campos. Desde la virtualización de un sistema operativo completo, es decir, emularlo en otro sistema, por lo que se podría tener varios sistemas operativos en ejecución a la vez en una única máquina, hasta la emulación de almacenamiento, que conforma un grupo de “contenedores” de datos, incompatibles entre ellos y



Figura 2.13: Varios ordenadores personales forman un *cluster*.

heterogéneos, formando una única unidad. En los servidores actuales se utiliza esta técnica para conseguir una serie de ventajas explicadas a continuación [3]:

- Rápida incorporación de nuevos recursos para los servidores virtualizados.
- Reducción de los costes de espacio, de *hardware* y consumo necesario.
- Administración global centralizada y simplificada.
- Permite gestionar los centros de cálculo como un *pool* de recursos o agrupación de toda la capacidad de procesamiento, memoria, red y almacenamiento disponible en nuestra infraestructura.
- Mejora en los procesos de réplica de sistemas. Tanto para pruebas como para *backups*.
- Aislamiento: Un fallo general de sistema de una máquina virtual no afecta al resto de máquinas virtuales.
- Migración en caliente de máquinas virtuales (sin pérdida de servicio) de un servidor físico a otro, eliminando la necesidad de paradas planificadas por mantenimiento de los servidores físicos.
- Balanceo dinámico de máquinas virtuales entre los servidores físicos, como se ha comentado en 2.2.

Existe un caso específico en el que sólo se virtualiza una parte concreta del sistema. Esto también supone muchas ventajas en cuanto al desarrollo de aplicaciones y el uso de determinados servicios en cualquier tipo de máquina.

2.5.1. Java y su máquina virtual

Java se puede denominar como un conjunto de herramientas (bastante amplio) que ha obtenido mucha fama desde su creación. Aparte de ser un lenguaje de programación orientado a objetos, tiene por detrás muchos servicios que lo hacen el lenguaje más utilizado del mundo en la creación de aplicaciones. Consta de [23]:

1. El lenguaje de programación orientado a objetos.
2. Bibliotecas estándar para todas las plataformas.
3. Compilador a *bytecode*, entendible por la máquina virtual.
4. La máquina virtual de Java que ejecuta ese código. Hace que cualquier programa escrito en este lenguaje funcione en cualquier plataforma independientemente del sistema instalado.

Es una plataforma que compila e interpreta código. Pero se pueden incluir también todas las otras herramientas que permiten añadir versatilidad a este conjunto:

1. La edición estándar de Java contiene bibliotecas para la realización de interfaces gráficas.
2. También dispone de *applets* que permiten añadir aplicaciones a entornos web de manera sencilla.
3. La edición de empresa (o J2EE) dispone de herramientas web para la creación de páginas y servicios web con la sintaxis normal. Estas herramientas son JSP, Servlets y Filtros.
4. Compilador a *bytecode*, entendible por la máquina virtual.
5. Dispone de conectores para cualquier base de datos del mercado, por lo que es sencillo utilizar estos productos en las aplicaciones.
6. Aunque no es propio de la plataforma de Java, Google ha desarrollado un SDK¹⁸ a disposición de los usuarios, para poder crear aplicaciones para su sistema operativo Android en este lenguaje.

Todo no son ventajas. Java tiene mala fama porque al ser un sistema virtualizado, utiliza más recursos y es más lento que los lenguajes completamente compilados, como lo son C o C++. Aún así, es indiscutible que debida a la gran versatilidad de la que dispone, es una herramienta indispensable para un programador.

J2EE: Java 2 Enterprise Edition

Esta versión más amplia, como ya se ha comentado, dispone de varias herramientas para la implementación de aplicaciones con un alto grado de comunicaciones, desde páginas web simples a servicios web sustentados en servidores. También se ha desarrollado gracias a Java programas como GlassFish y Tomcat, que permiten incluir estos servicios mediante un único archivo y son accesibles por red. La versión de empresas contiene todo lo necesario para crear servidores de aplicaciones, en diversas capas distribuidas, flexible y escalable.

A partir de ahí, se han creado multitud de *frameworks* y otras herramientas para un desarrollo más sencillo y más estructurado.

Actualmente el futuro de Java es un poco incierto debido a que la empresa Oracle compró Sun Microsystem, y ha optado por políticas que impiden el desarrollo libre de esta plataforma.

¹⁸Software Development Kit. Conjunto de bibliotecas para el desarrollo de aplicaciones en una determinada plataforma.

2.6. El almacenamiento

Se obtiene una correspondencia entre las plataformas de almacenaje masivo de información con los distintos tipos de sistemas ya que están basadas en ellos. Se puede decir que el almacenamiento es la segunda parte más importante de la informática, después del tratamiento de esos datos. Es por ello que la vía de las bases de datos ha ido en constante evolución de forma paralela a las comunicaciones entre ordenadores.

No se explicará la capa más baja del almacenaje de la información, como son los sistemas de ficheros ya que no compete a este proyecto. Los sistemas de almacenaje pasan a ser grandes administradores de información que no sólo se preocupan por contener los datos, sino también por ofrecer soluciones para la búsqueda y el mantenimiento de los mismos.

2.6.1. Sistemas Gestores de Bases de Datos

Son conocidos como el producto final de la base de datos en sí. Engloban al conjunto de herramientas necesarias para utilizar todo lo referente a esas bases de datos. Desde el lenguaje de consulta SQL (ver página 9) hasta los motores de optimización y búsqueda, administración de usuarios y seguridad, comunicaciones, incluso algunas proveen herramientas específicas para acceder a la información desde otras plataformas mediante conectores.

Existen en la actualidad multitud de soluciones para almacenamiento, adecuadas al uso que se les vaya hacer, con gran cantidad de datos o plataformas especialmente ligeras para una rápida utilización.

En este proyecto se van a centrar en usar dos específicas para dos necesidades distintas, como son MySQL y SQLite.

2.6.2. SQLite

Este gestor tiene una característica muy importante y es que es extremadamente ligero. Tan ligero que se usa en dispositivos de baja capacidad como los anteriormente mencionados *smartphones*. No está preparado para manejar grandes cantidades de información, ni de tratamiento avanzado de datos, como, por ejemplo, las claves ajenas de las tablas. Dispone de una interfaz muy sencilla y accesible desde cualquier lenguaje de programación, sacando el máximo rendimiento desde C. A costa de su flexibilidad, se han sacrificado las tareas de mantenimiento y de optimización, por lo que no cuenta con procesos que ayuden a obtener la información de manera más rápida. No dispone de acceso remoto a la información y se comporta como un fichero binario en el sistema operativo.

Se puede ver en la figura 2.14 que dispone de una estructura mucho más sencilla que el resto de soluciones que existen en la actualidad. SQLite es código libre y se puede obtener de manera totalmente gratuita.

2.6.3. MySQL

Muy popularizado en Internet, este gestor gratuito pero con licencia doble (comercial y libre) es una solución media para la mayoría de los proyectos existentes. Contiene todas las opciones de un

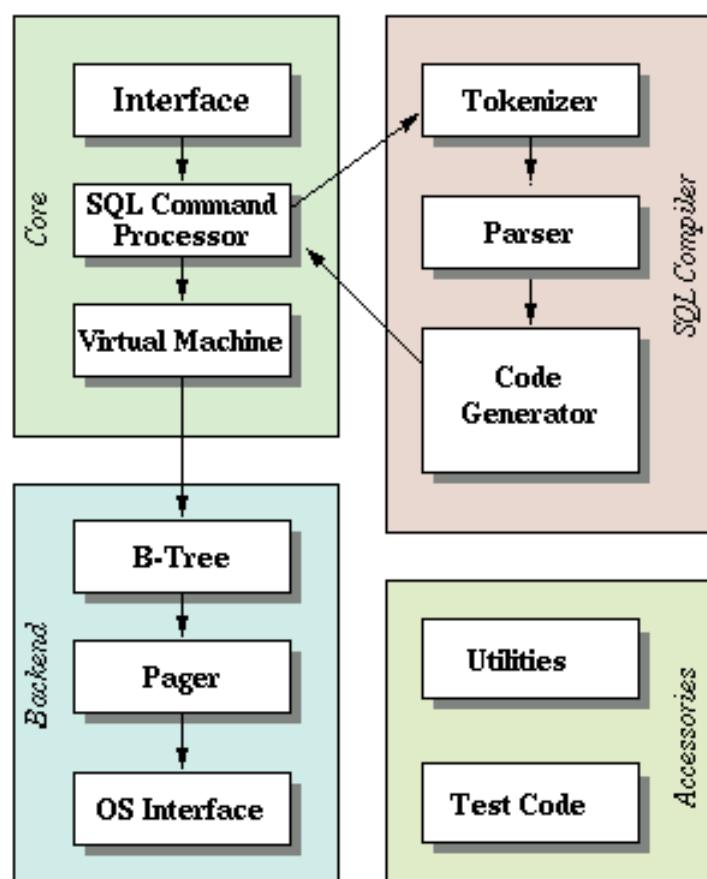


Figura 2.14: SQLite no dispone de procesos de optimización.

gran gestor empresarial, pero con un tamaño más reducido y menos optimizado. Es perfecto para cantidades moderadas de información (como las utilizadas en pequeñas y medianas empresas), y se han realizado multitud de versiones para optimizar sus capacidades, como Google, que la modificó para aumentar su capacidad de mantenimiento (por motivos estructurales), o Facebook, que actualmente sigue utilizando los servidores con MySQL.

Posee múltiples procesos de mantenimiento que necesitan estar en constante ejecución, además les da acceso de forma remota, por lo que es idóneo para sistemas con un alto grado de comunicación.

Existen multitud más de soluciones, como son Oracle Database, la *open source* por excelencia, PostgreSQL, también empresariales como las proporcionadas por IBM y los grandes almacenes de datos.

2.7. La comunicación

Todas las comunicaciones en los ordenadores actuales poseen una misma estructura definida por los protocolos que intervienen en ellas. Aunque cada protocolo tenga una función, debido a que el sistema está definido por capas, comparten muchos de los elementos. Para empezar esta sección, se definirá el modelo actual que se aplica a Internet, ya que es el que se utilizará en este proyecto. Después, se dará un repaso a las distintas tecnologías que están por encima de ese modelo, y las utilidades y características más importantes de cada una.

2.7.1. El modelo TCP/IP

Es un modelo de referencia que se usó para ARPANET, a pesar de existir otro estándar definido en 1983 desarrollado por la ISO (Organización Internacional de Estándares), que es el que se ha mantenido hasta nuestros días. Se basa en una pila de capas, cada cual usa sus capas inferiores y les proporciona más funcionalidad. Así, la capa de menor nivel es la más básica.

Estas capas tienen unos protocolos que actúan con la información que le llega de una capa superior, como se ve en la figura 2.15.

Capa de red

Es la capa de menor nivel y se encarga de las tareas más básicas de comunicación. Realmente, localiza una dirección MAC¹⁹ correspondiente a una máquina y envía bit a bit toda la información suministrada por capas superiores. Además añade comprobaciones de errores a nivel de bit. Esta es dependiente de la estructura de red, y existen versiones para redes LAN, ARPANET, Radio de paquete y SATNET.

Capa de interred

Permite mantener la transparencia entre distintas redes, por lo que es el centro de unión entre toda la arquitectura. Su trabajo es permitir que los equipos inyecten datos dentro de cualquier red y que éstos viajen a su destino de manera independiente.

¹⁹Dirección única asignada a cada interfaz de red, correspondiente con la subcapa de Control de Acceso al Medio [24].

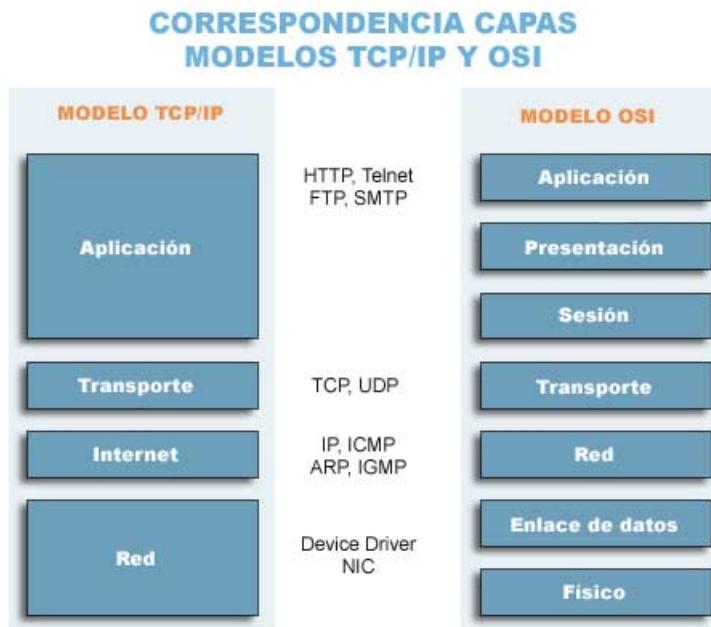


Figura 2.15: Modelo TCP/IP y modelo OSI desarrollado en 1983.

Su protocolo principal es IP (Protocolo de Internet) y da nombre al modelo.

Capa de transporte

Permite a una máquina origen y destino mantener una conversación completa, independientemente de cuantos “saltos entre redes” tengan que realizar los datos.

Los protocolos de esta capa son TCP (Protocolo de Control de Transporte), orientado a conexión y seguro que también da nombre al modelo, y UDP (Protocolo de Datagrama de Usuario) para aplicaciones que no deseen control de flujo.

Capa de aplicación

Es la última capa que está en contacto con el usuario. Es sobre ella donde el *software* de las máquinas puede crear sus propios protocolos, teniendo por debajo todos los anteriormente descritos. Los protocolos más famosos comprenden:

HTTP: Protocolo de Transferencia de Hipertexto [20]. Es base para la WWW (2.1.7). Puede transferir cualquier tipo de archivo a través de Internet. Está estructurado de forma Cliente/Servidor orientado a transacciones.

FTP: Es el protocolo estándar de Internet para envío de archivos. Tiene la característica de usar dos tipos de conexión, una de control y otra de datos [21].

SMTP: Protocolo Simple de Transferencia de Correos. Es parecido al protocolo FTP pero su finalidad es enviar y recibir mensajes enviados de un usuario a otro a través de la red [21].

DNS: Sistema de Nombres de Dominio. Se compone de una base de datos distribuida que hace la correspondencia entre nombres de máquinas y sus direcciones.

Los protocolos deben ser implementados sobre un sistema de envío, desde los más simples, como los *sockets*, hasta plataformas complejas como .NET. A continuación se describirán los más importantes.

2.7.2. Sockets

Son la abstracción básica que contienen los diferentes sistemas operativos para enviar datos mediante los distintos protocolos de nivel de transporte anteriormente detallados. A bajo nivel, funcionan como posiciones de memoria donde se escriben y se leen datos, conectan dos procesos de forma bidireccional entre dos máquinas [11].

Su funcionamiento es sencillo:

1. Se crea el *socket* asignándole un protocolo de transporte (generalmente TCP o UDP).
2. Al *socket* se le asigna un puerto de la máquina al que conectarse o al que va a escuchar (dependiendo si es del lado del cliente o del servidor). Desde ese momento, este está a la espera y preparado para la comunicación.
3. Mediante operaciones de lectura y escritura (similares a la escritura y lectura de ficheros) se envían y reciben datos entre los dos extremos de la comunicación.
4. Se cierran los *sockets* de conexión.

Con este mecanismo se han creado los protocolos complejos y las plataformas que se describen a continuación.

2.7.3. Llamadas a procedimientos remotos

Se trata de una implementación por medio de *sockets* que permite invocaciones remotas de funciones. De forma transparente al programador, se realiza la comunicación, se procesa en la otra máquina y se devuelve el resultado. Son la base de las plataformas distribuidas, basadas en objetos [11]. Su evolución directa son las *Invocaciones a Procedimientos Remotos*.

2.7.4. Java RMI

Compone el conjunto de herramientas implementadas para la máquina virtual de Java (ver sección 2.5.1), que permiten realizar aplicaciones distribuidas, ocultando las comunicaciones para que el desarrollador programe como si todo el contenido estuviera en la misma máquina.

Se basa en una estructura de servicios a los que, mediante un registro de nombres, los clientes acceden y obtienen la localización del servicio requerido [15]. Estos pueden invocar métodos que no se encuentran en su propia máquina. Cada aplicación contiene una interfaz donde se especifican los métodos remotos y los parámetros de los mismos.

Como se puede ver en la figura 2.16, el *RMI Registry* debe ser accesible desde todos los nodos de la red. Estos, mediante una etiqueta, obtienen la localización del objeto dentro de la red.

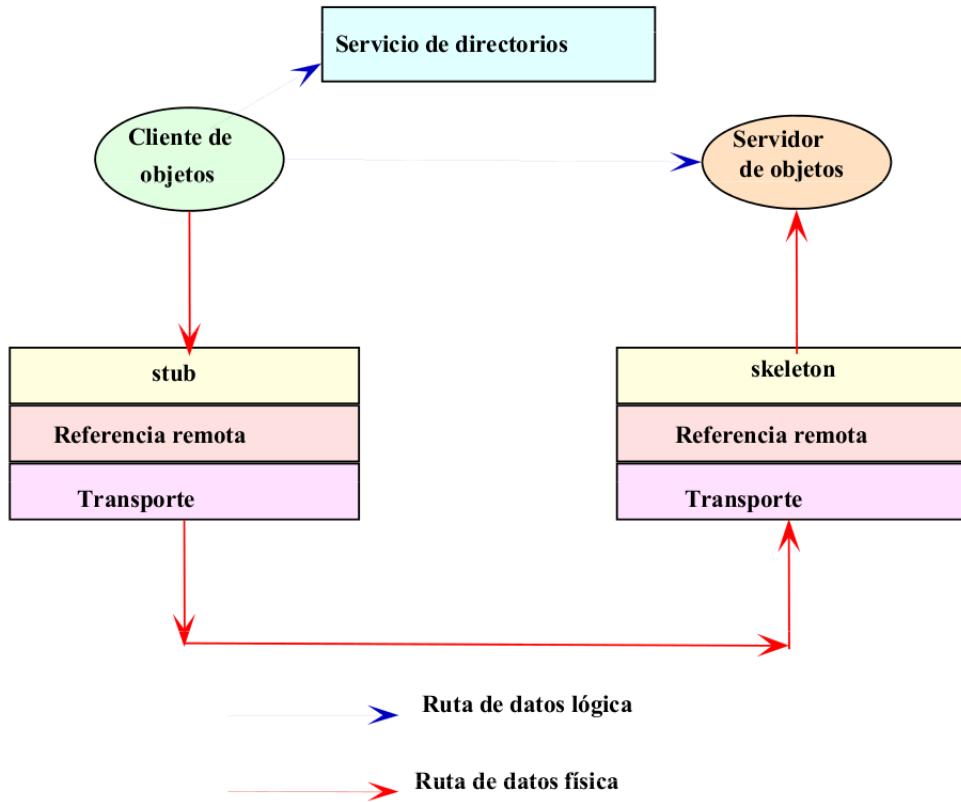


Figura 2.16: Arquitectura de Java RMI [14].

Después son conectados a la máquina que lo posee y los objetos son enviados serializados²⁰ para su posterior uso.

El inconveniente más importante es que esta plataforma sólo se puede utilizar en *software* para se ejecute en la de Java, limitando el número de programas que pueden interactuar con ella.

2.7.5. CORBA

Con una arquitectura muy parecida a Java RMI, es una plataforma más compleja que permite interactuar independientemente del lenguaje de programación utilizado en el *software*. Esto hace posible que se realicen comunicaciones entre distintas arquitecturas de forma transparente al usuario y al desarrollador. A pesar de los lenguajes utilizados, CORBA está basado en el paradigma orientado a objetos, por lo que, aunque se estén utilizando lenguajes como C o Matlab, se realizan invocaciones a métodos.

Se caracteriza por definir las interfaces de acceso con un lenguaje llamado IDL (Lenguaje de Definición de Interfaces) que, al igual que en Java RMI, todos los nodos deben poseer. Está considerada como una de las arquitecturas más difíciles de utilizar debido a su gran complejidad y al número de configuraciones posibles. Actualmente está siendo utilizada en proyectos de gran envergadura donde se requiere una alta interoperabilidad.

²⁰Un objeto serializado es su representación íntegra de forma textual, que se puede escribir o leer de un fichero (al igual que enviar como un dato más). Generalmente es usado para guardar objetos en el estado actual.

En la figura 2.17 se puede apreciar las distintas capas de la arquitectura.

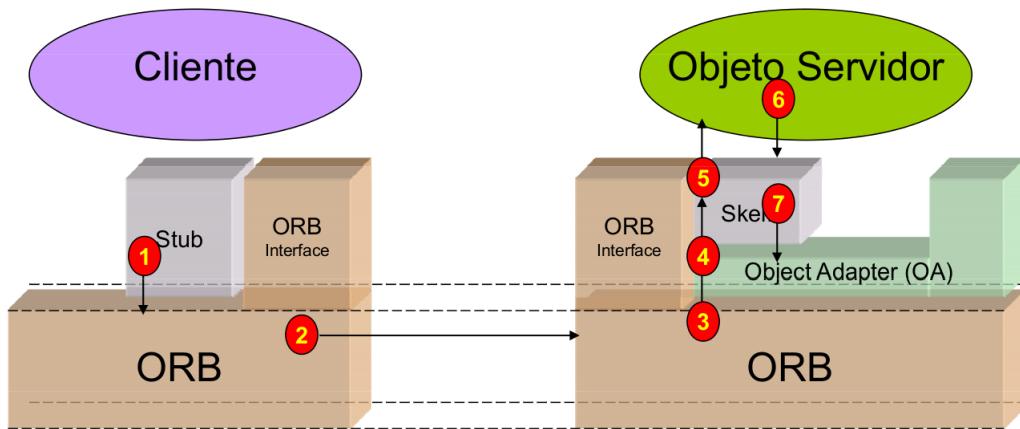


Figura 2.17: Componentes de un sistema CORBA [14].

2.7.6. Framework .NET

Esta plataforma fue desarrollada por Microsoft para dar salida a los lenguajes en los que se puede desarrollar *software* en Windows, aunque también dispone de plataforma en sistemas Unix. Al igual que Java, utiliza una “máquina virtual” para ejecutar el código compilado (CLR o *Common Language Runtime*, figura 2.18 [12]). Gracias a ello, se aglutinaron todos los lenguajes de la plataforma (C#, J#, Visual C++, Visual Basic y .NET) creando una solución muy versátil. Además, permite la comunicación, al igual que Java RMI, transparente al usuario entre estos programas. La potencia es mayor cuando se pueden incluir servicios web nativamente.

Para desarrollar en esta plataforma se dispone de un entorno especializado, Visual Studio, que añade más posibilidades a .NET.

En los últimos años, ha surgido un auge ya que Microsoft permitió añadir compatibilidad con .NET a su sistema operativo para dispositivos móviles, *Windows Phone 7*.

2.7.7. Servicios Web

Como ya se comentó en la página 13, la tendencia pasa a ser la creación de servicios destinados tanto a otras empresas como a clientes finales. Estos servicios se pueden considerar como aplicaciones completas a las que se accede remotamente.

Por motivos de seguridad, se aceptó que la forma de comunicar estos servicios fuera a través de los métodos tradicionales de Internet, es decir, mediante los protocolos ya establecidos, como es HTTP. Este protocolo se quedó pequeño pronto y se plantearon añadir más funcionalidades por encima sin perder esa base. Es así como se crean los protocolos de intercambio de objetos (más complejos que simple información), como SOAP, que utiliza un formato XML para el envío e identificación de los objetos (véase figura 2.19).

Como en las anteriores plataformas descritas, los servicios web se definen mediante una interfaz común llamada WSDL (*Web Service Description Language*), donde los clientes pueden obtener toda

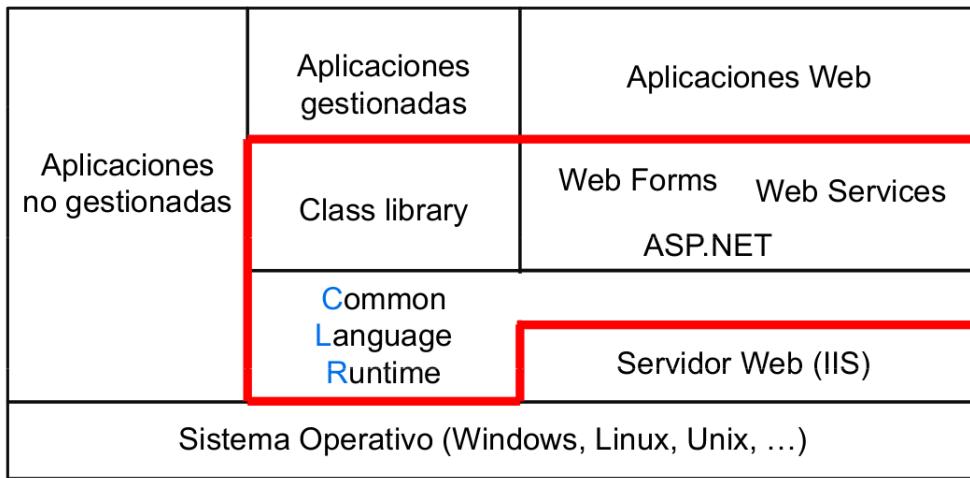


Figura 2.18: Arquitectura de .NET [14].

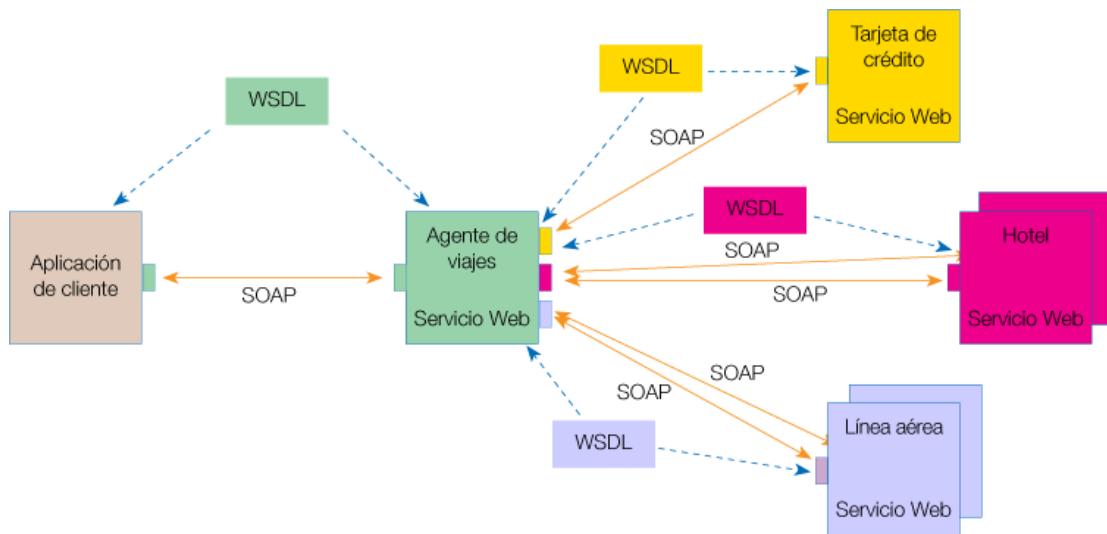


Figura 2.19: Estructura típica de un sistema con varios servicios web.

la información de los métodos a utilizar, sus parámetros y sus excepciones. Surgen vías alternativas que aprovechan los métodos de comunicación actuales de los navegadores de Internet, para obtener el máximo beneficio de estos servicios. Un ejemplo es REST, cada vez más extendido, el cual realiza las peticiones mediante la URL de la propia página, introduciendo como parámetros el método a ejecutar. Este sistema es usado en sitios web como Facebook y son de gran ayuda a los desarrolladores ya que no necesitan ningún tipo de librería externa para poder utilizar por completo los distintos servicios que provee la empresa.

2.8. Las limitaciones

A lo largo de este capítulo se ha demostrado cómo se han ido superponiendo unas soluciones a otras, al igual que aparecían nuevos problemas y nuevas necesidades surgidas a partir de las propias soluciones. Aún queda mucho camino para resolver las grandes limitaciones de potencia

y capacidad, pero otras pueden ser solventadas mediante las ideas surgidas de los investigadores, estudiantes e ingenieros.

2.8.1. La interoperabilidad

Desde que se creó el “K3”, cada empresa, investigador o científico ha implantado sus soluciones sin preocuparse de la interacción con su entorno, pero en estos tiempos, ha sido una de las necesidades más urgentes debido a la gran cantidad de dispositivos y productos que existen en el mercado. Las empresas que sólo miran por su producto se condenan al fracaso a largo plazo. La competencia es mayor y las guerras de productos incomodan cada vez más al cliente, que, por lo general, no tiene una compañía como exclusiva.

Gracias a que el desarrollo de las distintas soluciones en cuanto a comunicación y almacenaje se ha mantenido al margen de productos muy concretos, parece ser que la solución al problema de interoperabilidad pasa obligatoriamente por estos campos. Con la aparición de la “nube” y los sistemas virtuales, se ha salvado este gran obstáculo (en mayor o menor medida). Pero sigue faltando soluciones más genéricas para *software*. Aparte de Java y .NET, no existen otras plataformas que permitan ejecutar programas independientemente de la máquina ni del sistema operativo. Es lógico entonces que si se desea solventar un problema se necesite utilizar cualquiera de esos dos sistemas.

2.8.2. Acceso remoto

Se ha avanzado mucho desde que cada máquina era independiente una de otra, y todo lo necesario estaba contenido en la misma. Ahora la sustitución de una de estas en una red o un sistema complejo es casi inmediato, sin tener que realizar *backups* concretos ni alterando la estructura de dicha red. Cada nuevo producto que se crea en la actualidad deja de verse como algo instalado en una máquina, sino más bien como un conjunto de programas que necesitan estar comunicados en servidores, clientes, terminales, etc. Por supuesto, sigue habiendo *hardware* que aún es imposible su acceso remoto, y es este proyecto el que pretende solucionar en gran medida este problema, usando las tecnologías anteriormente mencionadas, y aprendiendo de la evolución de los sistemas.

Capítulo 3

Análisis

Este capítulo está centrado en la especificación de la plataforma RLF, cómo se concibió y todas las funcionalidades que ofrece. Además de los requisitos establecidos que debe cumplir.

3.1. Introducción

El análisis de este producto ha sido guiado por los estándares recogidos por la Agencia Espacial Europea para los proyectos de Ingeniería del Software [8]. La estructura de contenidos se ha mantenido aunque se han modificado algunos de los apartados considerados innecesarios ya que se especifican para un conjunto de documentos que describan un producto, y no para uno solo, como ocurre en este caso.

Para el desarrollo de RLF Prototype se ha contado con dos clientes del ámbito educativo, cada uno especializado en un área que han determinado las funciones que contiene el sistema. Así, se cuenta con un cliente ingeniero industrial (**Cliente A**) y con un cliente ingeniero informático (**Cliente B**).

En primera instancia, se especifican las condiciones generales del proyecto, así como sus capacidades más notables. A continuación, se listan todos los requisitos que fueron impuestos para la creación del producto, así como en conjunto de acciones a realizar por los distintos usuarios. Por último, se tratarán los requisitos desde el punto de vista del diseño.

Una vez todos los aspectos del análisis estén recogidos y establecidos, se pasará a diseñar desde el punto de vista técnico el sistema RLF.

3.2. Descripción general

3.2.1. Capacidades generales

El producto RLF cumple la finalidad de ofrecer servicios a los usuarios. La lista de capacidades que se muestra a continuación es el resultado de un primer análisis, que posteriormente serán desarrollados para una mayor comprensión:

- RLF provee de herramientas, generalmente con componentes *hardware*, a los usuarios, los cuales pueden utilizarlas de forma exclusiva durante un determinado tiempo.
- Las herramientas ofrecen un conjunto de acciones a realizar. Estas acciones son configuradas por parte del usuario para adecuarse a sus necesidades.
- El usuario está informado en todo momento del estado del sistema y de las acciones comandadas, así como de las herramientas a las que tiene acceso.
- Cada acción tiene la libertad de utilizar todo su potencial mediante servicios externos a la plataforma RLF.
- Los usuarios pueden acceder al sistema en tiempo real sin limitaciones horarias, siendo posible utilizarlo de manera remota.
- Se establecen un conjunto de normas, o maneras de trabajar, para desarrollar nuevas herramientas, que se comportan de manera estándar para su inmediato uso.
- Proporciona mecanismos para asegurar la integridad tanto del *software* como *hardware* que se ofrece mediante la plataforma RLF.

3.2.2. Restricciones generales

Como cada producto *software*, RLF cumple unas determinadas restricciones que se aclararon también en el análisis inicial. Son las que han condicionado el diseño de la plataforma:

- Ningún componente de la RLF está limitado por el sistema que lo contiene, ya que forma parte de un sistema multiplataforma, siendo las herramientas la única excepción.
- La plataforma RLF responde de igual manera independientemente de dónde se encuentren los distintos componentes, siendo posible contenerla en una misma máquina, o en una red de computadores.
- Se asegura la integridad de cada componente por separado, sin permitir que por un mal uso o por congestión aparezcan errores inesperados.

3.2.3. Características de los usuarios

Se identifican principalmente tres tipos de usuario para los que va destinada el sistema. Cada uno desempeña un rol distinto, y por lo tanto, se determinan distintas características y responsabilidades para cada uno. Cabe esclarecer que dentro de cada categoría de usuario, puede haber distintos niveles, pero que estos no son determinantes en el uso de la plataforma. Los roles que se especifican a continuación no son jerárquicos ni comparten acciones:

- **Administradores:** Personal técnico que gestiona y mantiene la plataforma RLF en constante funcionamiento. Realizan tareas necesarias para asegurar la disponibilidad completa del producto, así como gestionar todos los datos que componen el conjunto de información manejado por el sistema. Es por ello que deben poseer conocimientos medios de utilización de bases de datos y diversos sistemas operativos.
- **Desarrolladores:** Profesionales encargados de crear las herramientas para su posterior utilización por parte de los clientes. Deben cumplir un conjunto de normas aplicadas al *framework* para desarrollar el *software* necesario que la plataforma RLF pueda manejar. Tienen una interacción muy limitada con el sistema.
- **Clientes:** Son los usuarios finales de la plataforma, que mediante las distintas aplicaciones cliente (aquellas que dan acceso a RLF) pueden utilizar las herramientas asignadas.

Se entiende que una misma persona puede poseer varios roles. No se tienen en cuenta los usuarios indirectos de la plataforma, que no acceden a ella, como son los profesores en un entorno educativo que definen el conjunto de herramientas a la cual un usuario tiene acceso, o los administradores de red que se encargan del buen funcionamiento de la misma. Estas acciones recaerán directamente en el usuario administrador.

3.2.4. Entorno operacional

En este punto se analizan las necesidades tecnológicas del sistema desde el punto de vista de los usuarios. Se debe distinguir entre las necesidades de cada uno de los roles anteriormente comentadas:

ADMINISTRADOR
Ordenador portatil con acceso a redes <i>Ethernet</i> e inalámbricas. No es necesario que disponga de interfaz gráfica. Se requiere Java JRE y MySQL Query Browser (véase el apéndice B).

Tabla 3.1: Entorno operacional del administrador.

DESARROLLADOR
Ordenador con acceso a redes. Requiere los elementos para poder desarrollar las herramientas en el lenguaje deseado. Dependiendo de la configuración de la plataforma puede ser necesario un cliente FTP.

Tabla 3.2: Entorno operacional del desarrollador.

CLIENTE
Cualquier tipo de máquina que contenga una acceso a la red y la máquina virtual Java.
Dispositivo portatil tipo <i>smartphone</i> o <i>tablet</i> con acceso a redes inalámbricas y navegador de Internet compatible con Javascript.

Tabla 3.3: Entorno operacional del cliente.

3.3. Requisitos del usuario

A continuación se listan todos los requisitos impuestos por los clientes A y B (ver sección 3.1) para el diseño de la plataforma RLF. Están clasificados dependiendo de su tipo, si aportan funcionalidad o si registran, y contienen una subclasiﬁcación dependiendo de a qué componente se refieran. Las características de un requisito vienen dadas por su identificador, prioridad en cuanto a implantación, fuente, necesidad y descripción.

3.3.1. Requisitos de capacidad

General: Hardware remoto			
IDENTIFICADOR	RUC-1	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
El hardware será controlado desde la plataforma sin necesidad de configurarlo en el lugar físico donde se encuentra.			

General: Hardware distribuido			
IDENTIFICADOR	RUC-2	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
El hardware que se pondrá a disposición de los clientes estará distribuido en varios ordenadores.			

General: Hardware exclusivo			
IDENTIFICADOR	RUC-3	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Todo hardware sólo podrá ser usado por un único cliente, sin que el uso de otros pueda influirle.			

General: Hardware multidisciplinar			
IDENTIFICADOR	RUC-4	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
La plataforma RLF contendrá una interfaz de acceso genérica para cualquier tipo de hardware.			

General: Plataforma accesible			
IDENTIFICADOR	RUC-5	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
La plataforma será accesible para los clientes desde cualquier ordenador conectado a Internet, siendo posible la configuración para limitar este acceso.			

General: Acceso central			
IDENTIFICADOR	RUC-6	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Independientemente de dónde se encuentre el <i>hardware</i> el que el cliente quiere utilizar, se accederá desde una misma dirección para todo el sistema.			

General: Plataforma modular			
IDENTIFICADOR	RUC-7	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
La plataforma RLF estará compuesta por módulos que interactuarán entre ellos, al menos dos, los servidores locales o laboratorios y el servidor central o proveedor .			

General: Cliente del la plataforma			
IDENTIFICADOR	RUC-8	PRIORIDAD	Baja
FUENTE	Cliente B	NECESIDAD	Opcional
DESCRIPCIÓN			
Para acceder a la plataforma se dispondrá de un cliente que no limitado por el sistema operativo donde se encuentre.			

General: Servicios como hardware			
IDENTIFICADOR	RUC-9	PRIORIDAD	Media
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
El <i>hardware</i> deberá ser encapsulado como un tipo de servicio genérico, llamado herramienta .			

Proveedor: Base de datos central			
IDENTIFICADOR	RUC-10	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
El proveedor contendrá la información en una base de datos central a la que accederá cada vez que se requiera utilizar dicha información.			

Proveedor: Usuarios clientes			
IDENTIFICADOR	RUC-11	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Los clientes poseerán un nombre de usuario, una contraseña, un email y un rol específico para acceder a las herramientas.			

Proveedor: Usuarios administradores			
IDENTIFICADOR	RUC-12	PRIORIDAD	Media
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Los administradores serán autenticados en la base de datos del proveedor para controlar el acceso a sistemas críticos.			

Proveedor: Servicio Web			
IDENTIFICADOR	RUC-13	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
El proveedor contendrá una interfaz de acceso para la conexión a través de Internet y así permitirá a los clientes utilizar los servicios de la plataforma.			

Proveedor: Información en tiempo real			
IDENTIFICADOR	RUC-14	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Los clientes obtendrán la información del proveedor actualizada, tanto del estado de las herramientas como el de los laboratorios.			

Proveedor: Información de las herramientas			
IDENTIFICADOR	RUC-15	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
La información de las herramientas estará contenida en la base de datos del proveedor para obtener una mayor respuesta cuando los clientes la soliciten.			

Proveedor: Información de los laboratorios			
IDENTIFICADOR	RUC-16	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Al ser un sistema con un componente central, la información de cada laboratorio, es decir, dónde se localiza y cómo puede contarse a él, se encontrará en la base de datos del proveedor.			

Proveedor: Acciones			
IDENTIFICADOR	RUC-17	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Las acciones que se podrán llevar a cabo mediante la interfaz de acceso al proveedor serán conexión o desconexión de un cliente, obtención de la información de las herramientas y su estado, y reserva de las mismas.			

Proveedor: Monitor			
IDENTIFICADOR	RUC-18	PRIORIDAD	Baja
FUENTE	Cliente B	NECESIDAD	Opcional
DESCRIPCIÓN			
El proveedor tendrá otra interfaz de acceso limitada únicamente para conectar y desconectar del sistema, y además obtener el estado actual de las herramientas.			

Proveedor: Seguridad en el acceso			
IDENTIFICADOR	RUC-19	PRIORIDAD	Media
FUENTE	Cliente B	NECESIDAD	Opcional
DESCRIPCIÓN			
Todas las comunicaciones del proveedor con los clientes estarán debidamente cifradas ya que pueden atravesar redes inseguras.			

Proveedor: Registro de herramientas			
IDENTIFICADOR	RUC-20	PRIORIDAD	Media
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
El proveedor dará acceso a los laboratorios para que registren herramientas, generando un identificador único para cada herramienta y una clave de uso.			

Proveedor: Tiempo de acceso máximo			
IDENTIFICADOR	RUC-21	PRIORIDAD	Media
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
A cada usuario se le asignará un tiempo de acceso máximo contado en minutos, que una vez sobrepasado se expulsa del sistema.			

Proveedor: Reserva de herramientas			
IDENTIFICADOR	RUC-22	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
El proveedor se encargará de bloquear las herramientas reservadas por los clientes para que no puedan ser usadas.			

Laboratorios: Comunicación			
IDENTIFICADOR	RUC-23	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Toda la comunicación entrante y saliente de un laboratorio se realizará mediante red.			

Laboratorios: Base de datos			
IDENTIFICADOR	RUC-24	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada laboratorio contendrá su propia base de datos, que elimina el tráfico en la red y permite hacer operaciones locales más rápidas. En ellas se almacenará la información de las herramientas.			

Laboratorios: Administración			
IDENTIFICADOR	RUC-25	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Los laboratorios contarán con una interfaz de acceso para poder administrarlos sin necesidad de una parada total del sistema.			

Laboratorios: Registro de herramientas			
IDENTIFICADOR	RUC-26	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada laboratorio estará al cargo de varias herramientas, que deberá gestionar y administrar.			

Laboratorios: Acceso a las herramientas			
IDENTIFICADOR	RUC-27	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Para eliminar el tráfico del proveedor, los laboratorios aceptarán peticiones de los clientes que previamente se hayan conectado al sistema y hayan reservado las herramientas necesarias.			

Laboratorios: Eliminación de herramientas			
IDENTIFICADOR	RUC-28	PRIORIDAD	Baja
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Se podrá eliminar una herramienta concreta de un laboratorio si así se requiere.			

Laboratorios: Mantenimiento			
IDENTIFICADOR	RUC-29	PRIORIDAD	Baja
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Para aportar seguridad, los laboratorios tendrán un estado de “mantenimiento” donde no se permite la comunicación con los clientes, en el cual se podrá realizar tareas de administración del mismo.			

Laboratorios: Ejecuciones de las herramientas			
IDENTIFICADOR	RUC-30	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Los laboratorios sólo permitirán una instancia de ejecución por herramienta. Esto conlleva la creación de una cola de peticiones para la ejecución de estas.			

Laboratorios: Comunicación con las herramientas			
IDENTIFICADOR	RUC-31	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Un laboratorio se comunicará con las herramientas que tiene asignadas mediante un mecanismo asíncrono para que una herramienta con fallos no bloquee la ejecución.			

Laboratorios: Parada de emergencia			
IDENTIFICADOR	RUC-32	PRIORIDAD	Baja
FUENTE	Cliente A	NECESIDAD	Opcional
DESCRIPCIÓN			
Cada laboratorio podrá ser parado de manera urgente por parte de un administrador, que cierra todas las ejecuciones activas y elimina las pendientes. Después, el laboratorio se desactivará.			

Laboratorios: Logs			
IDENTIFICADOR	RUC-33	PRIORIDAD	Baja
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada laboratorio poseerá su propio mecanismo de registro de eventos y errores, que puede ser consultado por un administrador.			

Laboratorios: Información en tiempo real			
IDENTIFICADOR	RUC-34	PRIORIDAD	Media
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada vez que un laboratorio cambie su estado, el proveedor será informado, incluso cuando se realizan paradas de emergencia.			

Herramientas: Acciones			
IDENTIFICADOR	RUC-35	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada herramienta podrá realizar varias acciones, todas relacionadas con el mismo <i>hardware</i> . El usuario elegirá una y la configurará. No será posible ejecutar a la vez dos acciones de la misma herramienta.			

Herramientas: Configuración de las acciones			
IDENTIFICADOR	RUC-36	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada acción tendrá asociada unos parámetros de salida y de entrada, con un tipo de datos establecido y una descripción. Mediante la definición de los valores de estos parámetros la acción se configura para su ejecución. Los parámetros de salida sólo se leerán por parte del cliente cuando la acción ha terminado.			

Herramientas: Excepciones y estado final de la acción			
IDENTIFICADOR	RUC-37	PRIORIDAD	Media
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Las acciones, durante su ejecución, podrán lanzar excepciones que informan al usuario de un error, además, cada finalización tendrá asociada un estado incluido por el desarrollador.			

Herramientas: Constantes			
IDENTIFICADOR	RUC-38	PRIORIDAD	Baja
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Las herramientas podrán contener varias constantes con un tipo de datos concreto y un valor que no varía desde que se registra la herramienta.			

Herramientas: Atributos			
IDENTIFICADOR	RUC-39	PRIORIDAD	Baja
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Los laboratorios definirán los atributos propios de cada herramienta, donde se incluyen el identificador, la clave, nombre, descripción, rol, versión y administrador responsable de la herramienta.			

Herramientas: Entrada y salida			
IDENTIFICADOR	RUC-40	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Las herramientas dispondrán de una entrada y salida textual para poderse comunicar con el usuario.			

Herramientas: Servicios externos			
IDENTIFICADOR	RUC-41	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Para dar funcionalidad extra a cada herramienta, las acciones podrán tener asociadas servicios externos, que se activarán en la ejecución y que pueden ser usadas por los clientes.			

Herramientas: Archivo de descripción			
IDENTIFICADOR	RUC-42	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Para definir una herramienta por completo y registrarla en la plataforma RLF, se dispondrá de un fichero con un formato establecido que será leído en el momento del registro en el laboratorio.			

Herramientas: Librerías <i>libtool</i>			
IDENTIFICADOR	RUC-43	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Opcional
DESCRIPCIÓN			
Como forma de abstracción, se incluirán librerías para la comunicación de la herramienta con el laboratorio correspondiente. Estas librerías proveerán las funciones de acceso y desconexión, lectura y escritura de parámetros, lectura de constantes y atributos, lanzamiento y lanzamiento de excepciones.			

Herramientas: Herramientas de datos			
IDENTIFICADOR	RUC-44	PRIORIDAD	Media
FUENTE	Cliente A	NECESIDAD	Opcional
DESCRIPCIÓN			
Esta categoría especial de herramientas permitirá acceder a varios clientes a la vez, pero sin interactuar con el <i>hardware</i> . No disponen de entrada textual, ni de parámetros de configuración. Sólo tendrán una acción asignada.			

Herramientas: Limpiadores			
IDENTIFICADOR	RUC-45	PRIORIDAD	Media
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Existirá por cada herramienta una acción especial que establece el <i>hardware</i> de la herramienta a su estado original. No podrá ser utilizada por los clientes y de forma automática se lanzará cuando se detecta errores en alguna acción.			

Herramientas: Tiempo máximo de ejecución			
IDENTIFICADOR	RUC-46	PRIORIDAD	Media
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada acción poseerá un tiempo máximo de ejecución que el laboratorio monitoriza. Cuando ese tiempo es sobrepasado, se parará la ejecución por completo y se establece el estado de acción fallida.			

Cliente: Visualización simultánea			
IDENTIFICADOR	RUC-47	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
La interfaz del cliente será capaz de visualizar a la vez varias acciones de distintas herramientas.			

Cliente: Uso de las herramientas			
IDENTIFICADOR	RUC-48	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Se proveerá al usuario de una “consola” (con entrada y salida textual) para el manejo de cada una de las acciones.			

Cliente: Acciones			
IDENTIFICADOR	RUC-49	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
La interfaz permitirá al cliente conectarse y desconectarse del sistema, reservar herramientas, comprobar la información de las mismas y su estado en tiempo real, y la ejecución de acciones.			

Cliente: Reserva múltiple			
IDENTIFICADOR	RUC-50	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Optativa
DESCRIPCIÓN			
Un cliente podrá reservar de manera simultánea varias herramientas a las que tiene acceso.			

Cliente: Monitor			
IDENTIFICADOR	RUC-51	PRIORIDAD	Baja
FUENTE	Cliente A	NECESIDAD	Optativa
DESCRIPCIÓN			
La aplicación cliente dispondrá con una aplicación auxiliar por la que acceder la monitor central.			

Cliente: Información			
IDENTIFICADOR	RUC-52	PRIORIDAD	Media
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Todos los datos públicos de las herramientas serán mostrados al cliente por medio de la interfaz, añadiendo las descripciones a los parámetros.			

3.3.2. Requisitos de restricción

General: Cantidad de comunicación			
IDENTIFICADOR	RUR-1	PRIORIDAD	Media
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
La información trasmisita en las comunicaciones, así como el número de estas deberán ser reducidas al mínimo para no saturar cada componente.			

General: Sistema escalable			
IDENTIFICADOR	RUR-2	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Todo el sistema será escalable, es decir, que no influya en su funcionamiento el hecho de aumentar o disminuir el número de laboratorios o clientes.			

Proveedor: Control de seguridad			
IDENTIFICADOR	RUR-3	PRIORIDAD	Media
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Los clientes que no se desconecten de forma válida en el sistema o que sufren errores, el proveedor no les permitirá volver a conectarse hasta que un administrador dé el visto bueno.			

Proveedor: Bloqueos de peticiones			
IDENTIFICADOR	RUR-4	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
El proveedor contará con mecanismos para atender varias peticiones a la vez sin dejar ninguna a la espera.			

Laboratorios: Portables			
IDENTIFICADOR	RUR-5	PRIORIDAD	Alta
FUENTE	Cliente B	NECESIDAD	Esencial
DESCRIPCIÓN			
Los laboratorios funcionarán sin problemas en diversos sistemas operativos con una única implementación.			

Laboratorios: Acciones asíncronas			
IDENTIFICADOR	RUR-6	PRIORIDAD	Media
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Todas las acciones que se llevan a cabo en los laboratorios serán asíncronas, para no detener ni bloquear ningún componente que interfiera con ellos.			

Laboratorios: Ejecuciones externas			
IDENTIFICADOR	RUR-7	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Las acciones se ejecutarán en los laboratorios como programas externos para no añadir carga a la plataforma.			

Laboratorios: Máximo número de ejecuciones			
IDENTIFICADOR	RUR-8	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada laboratorio poseerá un número máximo de acciones en ejecución (configurado por el propio administrador). Cuando ocurre esto, las acciones que están pendientes tomarán el estado de espera.			

Herramientas: Parada			
IDENTIFICADOR	RUR-9	PRIORIDAD	Alta
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
Las herramientas deberán estar preparadas para sufrir paradas sin previo aviso, aunque no es necesario que devuelvan el estado original al <i>hardware</i> .			

Cliente: Control de seguridad web			
IDENTIFICADOR	RUR-10	PRIORIDAD	Media
FUENTE	Cliente A	NECESIDAD	Esencial
DESCRIPCIÓN			
El control de seguridad aplicado a los usuarios para el cliente de escritorio no bloqueará la cuenta cuando ocurran errores en el cliente web.			

3.4. Casos de uso

Un caso de uso define una serie de pasos o actividades que deben realizarse para llevar a cabo un proceso. Las entidades o personajes que participan en un caso de uso se denominan actores.

Este apartado se separa en dos partes. La primera es una representación gráfica de los distintos casos de uso ordenados por actores (administrador, desarrollador y cliente). En la segunda parte se explicarán textualmente algunos de los más influyentes, que representan las funcionalidades más importantes; *Publicar herramienta* y *Ejecutar acción*.

Como se aprecia en la figura 3.1 el administrador tiene responsabilidades de mantenimiento del sistema, y todas sus funcionalidades están relacionadas con esta tarea. El desarrollador en cambio, dentro del sistema sólo podrá realizar parte de la función *Publicar herramienta*. Se explicará más adelante este caso particular, el cual requiere de dos actores para poder completarse. En la figura 3.2 se comprueba que el usuario necesita haber iniciado sesión para realizar el resto de funciones, además de que para ejecutar una acción sea necesario reservar la herramienta que la contiene.

A continuación se muestra la información de los dos casos de uso anteriormente señalados, donde se incluye los requisitos para llevarlos a cabo, los posibles escenarios secundarios y su descripción detallada.

CU-1: Publicar herramienta	
ACTORES	Administrador, desarrollador
PRECONDICIONES	<ul style="list-style-type: none"> ■ El desarrollador debe haber implementado la herramienta de acuerdo con las normas del <i>framework</i>. ■ El laboratorio debe estar desarmado.
POSTCONDICIONES	<ul style="list-style-type: none"> ■ La herramienta estará a disposición de los usuarios para poder utilizarla.
DESCRIPCIÓN	
<p>El desarrollador crea una herramienta para incluirla en la plataforma. El administrador obtiene los datos necesarios para registrarla y se lo indica al desarrollador.</p>	
ESCENARIO PRINCIPAL	
<ol style="list-style-type: none"> 1. El administrador introduce el fichero de configuración en el laboratorio mediante la aplicación de mantenimiento. 2. El laboratorio pide una nueva clave e identificador al proveedor. 3. El laboratorio valida la configuración y crea las estructuras en su base de datos y la generada para la propia herramienta. 4. El laboratorio envía al proveedor la información sobre la nueva herramienta y establece su estado como “no disponible”. 5. El administrador recibe la nueva clave y el identificador. 6. El desarrollador modifica el código para incluir la nueva clave. 7. El administradorarma el laboratorio. 	
ESCENARIO SECUNDARIO	
<ol style="list-style-type: none"> 1. El administrador introduce el fichero de configuración en el laboratorio mediante la aplicación de mantenimiento. 2. El laboratorio pide una nueva clave e identificador al proveedor. 3. El laboratorio determina que la configuración no está bien construida y tiene errores. 4. El administrador recibe el error encontrado. 	

CU-2: Ejecutar acción	
ACTORES	Cliente
PRECONDICIONES	<ul style="list-style-type: none"> ■ La herramienta debe haber sido reservada por el usuario.
POSTCONDICIONES	<ul style="list-style-type: none"> ■ La acción se ejecuta en el laboratorio e interactua con el cliente.
DESCRIPCIÓN	
El cliente desea ejecutar una acción de una herramienta concreta.	
ESCENARIO PRINCIPAL	
<ol style="list-style-type: none"> 1. El cliente selecciona la acción a ejecutar y pulsa el botón. 2. La interfaz muestra los parámetros de entrada y los servicios externos de los que dispone la herramienta. 3. El cliente introduce el valor de esos parámetros y pulsa “Execute”. 4. El cliente abre los servicios externos indicados con anterioridad. 5. La aplicación cliente envía al laboratorio la petición de ejecución y se pone a la espera. 6. Cuando el laboratorio lo dispone, envía a la aplicación cliente la confirmación de ejecución y la información de conexión para el envío y recepción de datos. 7. La aplicación cliente se conecta con los nuevos parámetros al laboratorio y empieza a recibir los datos de la ejecución. 8. El cliente interacciona con la ejecución. 	
ESCENARIO SECUNDARIO	
<ol style="list-style-type: none"> 1. El cliente selecciona la acción a ejecutar y pulsa el botón. 2. La interfaz muestra los parámetros de entrada y los servicios externos de los que dispone la herramienta. 3. El cliente introduce el valor de esos parámetros y pulsa “Execute”. 4. El cliente abre los servicios externos indicados con anterioridad. 5. La aplicación cliente envía al laboratorio la petición de ejecución y se pone a la espera. 6. El laboratorio devuelve un fallo por no poder ejecutar la herramienta. 7. La aplicación cliente muestra el error por pantalla. 	

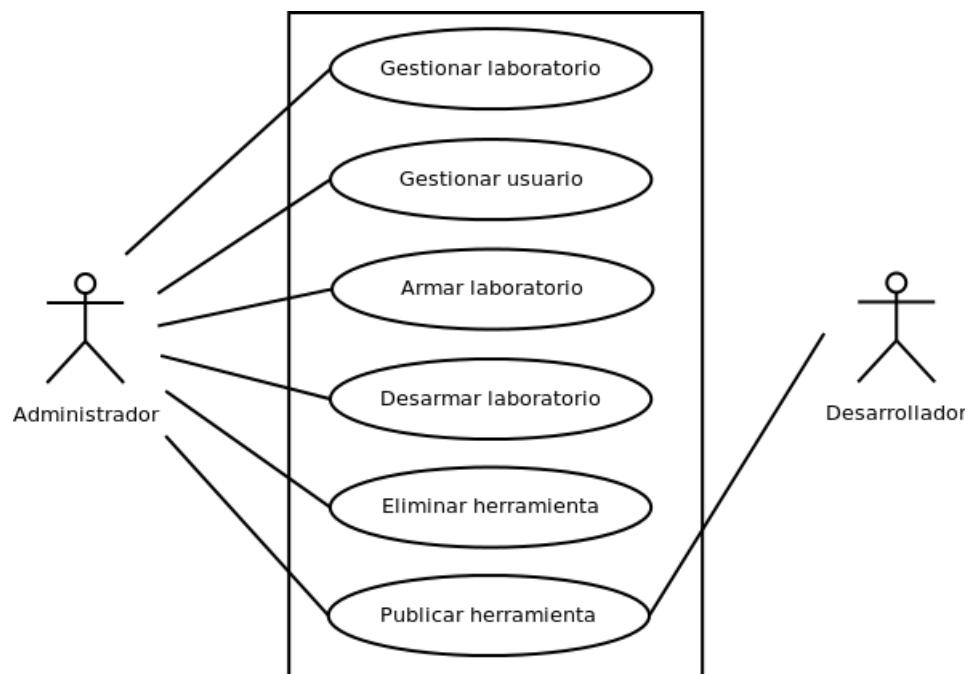


Figura 3.1: Casos de uso: Administrador y desarrollador

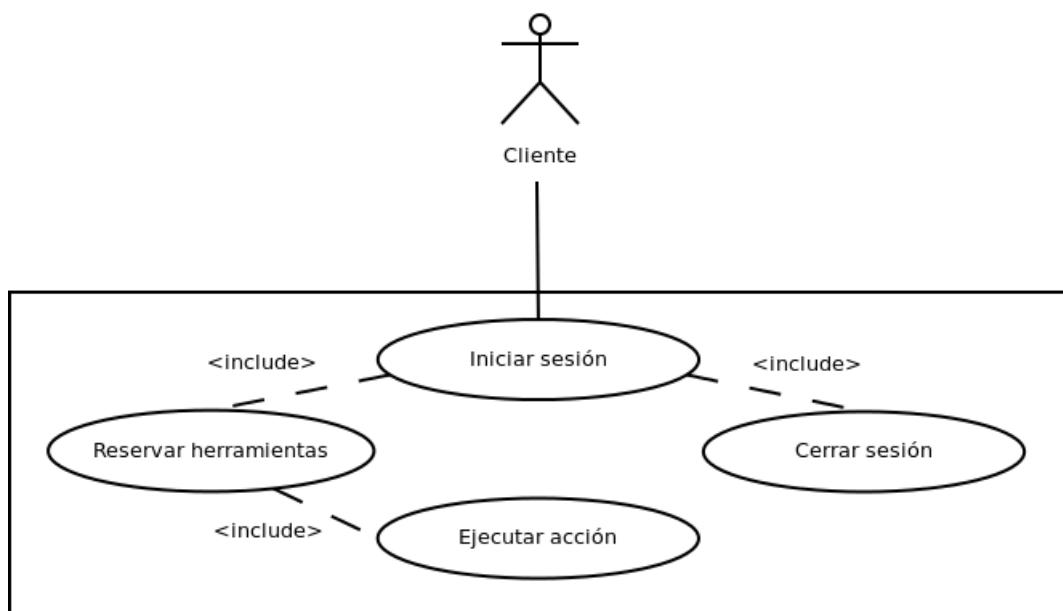


Figura 3.2: Casos de uso: Cliente

3.5. Requisitos del software

Los requisitos que aquí se muestran son la respuesta desde el punto de vista del diseño del *software* a los requisitos de usuario (tanto de capacidad como de restricción). La estructura es idéntica a los anteriores excepto que el tipo cambia a funcional y no funcional.

3.5.1. Requisitos funcionales

General: Aplicaciones			
IDENTIFICADOR	RUR-1	PRIORIDAD	Alta
FUENTE	RUC-1, RUC-9	NECESIDAD	Esencial
DESCRIPCIÓN			
El <i>hardware</i> será controlado completamente por aplicaciones que formarán parte de las herramientas y serán ejecutadas mediante peticiones de las aplicaciones cliente.			

Framework: Desarrollo versátil			
IDENTIFICADOR	RUR-2	PRIORIDAD	Alta
FUENTE	RUC-4	NECESIDAD	Esencial
DESCRIPCIÓN			
El <i>framework</i> de diseño de herramientas no impondrá restricciones en cuanto a lenguaje de programación, plataforma ni estructura del código.			

Framework: Desarrollo cómodo			
IDENTIFICADOR	RUR-3	PRIORIDAD	Alta
FUENTE	RUC-41	NECESIDAD	Esencial
DESCRIPCIÓN			
Se permitirán utilizar las funciones del sistema de entrada y salida por teclado para la comunicación de las herramientas con el cliente. Aunque la información que pasa a través de ellas sea enviada por la red.			

Comunicaciones: Modelo TCP/IP			
IDENTIFICADOR	RUR-4	PRIORIDAD	Alta
FUENTE	RUC-5, RUC-24	NECESIDAD	Esencial
DESCRIPCIÓN			
Todas las comunicaciones serán basadas en el modelo estándar TCP/IP y utilizarán un protocolo especialmente diseñado para RLF. Cada nodo contendrá una IP de acceso, y la plataforma se identificará con la IP del proveedor.			

Comunicaciones: Dirección de la plataforma			
IDENTIFICADOR	RUR-5	PRIORIDAD	Alta
FUENTE	RUC-6, RUC-7	NECESIDAD	Esencial
DESCRIPCIÓN			
La plataforma se identificará con la IP del proveedor, que será por la que se podrá acceder.			

Comunicaciones: Protocolo de comunicaciones			
IDENTIFICADOR	RUR-6	PRIORIDAD	Media
FUENTE	RUC-5, RUC-19	NECESIDAD	Esencial
DESCRIPCIÓN			
El protocolo de comunicaciones será utilizado por todos los módulos y no variará su implementación.			

Comunicaciones: Protocolo síncrono			
IDENTIFICADOR	RUR-7	PRIORIDAD	Media
FUENTE	RUC-24	NECESIDAD	Esencial
DESCRIPCIÓN			
El protocolo de comunicaciones será un modelo de petición/respuesta síncrono, aunque las acciones globales no lo sean.			

General: Vía de acceso			
IDENTIFICADOR	RUR-8	PRIORIDAD	Alta
FUENTE	RUC-6	NECESIDAD	Esencial
DESCRIPCIÓN			
El proveedor será el encargado de indicar a las aplicaciones cliente dónde se encuentran los laboratorios.			

Proveedor: Acceso a la base de datos			
IDENTIFICADOR	RUR-9	PRIORIDAD	Alta
FUENTE	RUC-10	NECESIDAD	Esencial
DESCRIPCIÓN			
El administrador podrá insertar, eliminar y modificar datos de la base de datos central. Será gestionada mediante la aplicación que el propio SGBD provea. El proveedor accederá mediante el driver correspondiente en la plataforma de desarrollo.			

Proveedor: Autenticación			
IDENTIFICADOR	RUR-10	PRIORIDAD	Media
FUENTE	RUC-11, RUC-12	NECESIDAD	Esencial
DESCRIPCIÓN			
La autenticación de cualquier usuario, independientemente del tipo de rol, será validada por el proveedor.			

Proveedor: Interfaz de acceso			
IDENTIFICADOR	RUR-11	PRIORIDAD	Alta
FUENTE	RUC-13	NECESIDAD	Esencial
DESCRIPCIÓN			
Las aplicaciones cliente usarán las funciones contenidas en el proveedor, las cuales, serán divididas en dos interfaces, con una estructura de servicio web definido por los ficheros WSDL y así dar acceso a las aplicaciones cliente.			

Proveedor: Actualizaciones de datos			
IDENTIFICADOR	RUR-12	PRIORIDAD	Alta
FUENTE	RUC-14	NECESIDAD	Esencial
DESCRIPCIÓN			
Los datos almacenados en el proveedor serán los últimos en actualizar. Por cada actualización correcta de datos realizadas por los distintos módulos se deberá considerar como un cambio permanente.			

Proveedor: Información comprimida			
IDENTIFICADOR	RUR-13	PRIORIDAD	Baja
FUENTE	RUC-15	NECESIDAD	Opcional
DESCRIPCIÓN			
Cuando un cliente solicite información sobre una herramienta, se entregará un conjunto de datos previamente comprimidos y calculados para una mayor velocidad de respuesta.			

Proveedor: Información de cada laboratorio			
IDENTIFICADOR	RUR-14	PRIORIDAD	Alta
FUENTE	RUC-16	NECESIDAD	Esencial
DESCRIPCIÓN			
Cuando un cliente reserve una herramienta, se le entregará la información de conexión del laboratorio que la contiene, como es la IP y el puerto.			

Proveedor: Funciones web			
IDENTIFICADOR	RUR-15	PRIORIDAD	Alta
FUENTE	RUC-17, RUC-18	NECESIDAD	Esencial
DESCRIPCIÓN			
Las funciones de las dos interfaces de acceso proveerán parámetros de entrada y salida, así como excepciones. Estos objetos serán capaces de ser serializados, para poder ser enviados mediante el protocolo HTTP y SOAP.			

Proveedor: Seguridad en las herramientas			
IDENTIFICADOR	RUR-16	PRIORIDAD	Alta
FUENTE	RUC-20	NECESIDAD	Esencial
DESCRIPCIÓN			
Las claves y los identificadores de las herramientas serán únicos y no reutilizables. Las claves se generarán mediante funciones <i>hash</i> a partir de la fecha y el identificador obtenido.			

Proveedor: Tiempo máximo asignado			
IDENTIFICADOR	RUR-17	PRIORIDAD	Alta
FUENTE	RUC-21	NECESIDAD	Esencial
DESCRIPCIÓN			
Los clientes tendrán preasignado un tiempo almacenado en la base de datos, que será cronometrado por los propios laboratorios en los que estén registrados.			

Proveedor: Bloqueo de herramientas			
IDENTIFICADOR	RUR-18	PRIORIDAD	Alta
FUENTE	RUC-3, RUC-22	NECESIDAD	Esencial
DESCRIPCIÓN			
La base de datos contendrá información de quién tiene reservada cada herramienta, impi- diendo que herramientas que no sean de datos se encuentren accesibles por dos clientes, por lo se requerirá un bloqueo de escritura y lectura en la información cada vez que se requiera reservar.			

Laboratorios: Localización de las aplicaciones			
IDENTIFICADOR	RUR-19	PRIORIDAD	Alta
FUENTE	RUC-2	NECESIDAD	Esencial
DESCRIPCIÓN			
En cada laboratorio se almacenarán, de forma local, todas las aplicaciones implicadas en las ejecuciones de las herramientas que tienen asignadas.			

Laboratorios: Base de datos			
IDENTIFICADOR	RUR-20	PRIORIDAD	Alta
FUENTE	RUC-25	NECESIDAD	Esencial
DESCRIPCIÓN			
La base de datos de cada laboratorio contendrá la ruta de acceso a la herramienta, su clave y su identificador. Será de baja capacidad y portable, por lo tanto, el SGBD será SQLite.			

Laboratorios: Comunicación con las herramientas			
IDENTIFICADOR	RUR-21	PRIORIDAD	Alta
FUENTE	RUC-32	NECESIDAD	Esencial
DESCRIPCIÓN			
El laboratorio creará una base de datos ligera (SQLite) para ofrecer comunicaciones con la herramienta. De manera asíncrona obtendrá los datos que se originen en la ejecución de la misma.			

Laboratorios: Kernel			
IDENTIFICADOR	RUR-22	PRIORIDAD	Alta
FUENTE	RUC-26	NECESIDAD	Esencial
DESCRIPCIÓN			
Un laboratorio ofrecerá una capa de gestión llamada <i>Kernel</i> que será independiente de las comunicaciones con los clientes y las ejecuciones de herramientas.			

Laboratorios: Funciones del Kernel			
IDENTIFICADOR	RUR-23	PRIORIDAD	Alta
FUENTE	RUC-30, RUC-34	NECESIDAD	Esencial
DESCRIPCIÓN			
El <i>Kernel</i> de cada laboratorio será el encargado de enviar y recibir comunicaciones con el proveedor, además de añadir y eliminar herramientas. También armarán y realizarán las acciones necesarias para desarmar el laboratorio.			

Laboratorios: Parada de emergencia en el Kernel			
IDENTIFICADOR	RUR-24	PRIORIDAD	Alta
FUENTE	RUC-30	NECESIDAD	Esencial
DESCRIPCIÓN			
Cuando el <i>Kernel</i> reciba una parada de emergencia detendrá los gestores anexos inmediatamente, lo que permitirá al administrador acceder al <i>hardware</i> de forma más rápida.			

Laboratorios: Gestor de comunicaciones			
IDENTIFICADOR	RUR-25	PRIORIDAD	Alta
FUENTE	RUC-28	NECESIDAD	Esencial
DESCRIPCIÓN			
Capa superior que administrará todas las peticiones recibidas por los clientes así como las notificaciones de las ejecuciones.			

Laboratorios: Tiempo máximo de un cliente			
IDENTIFICADOR	RUR-26	PRIORIDAD	Media
FUENTE	RUC-21	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada gestor de comunicaciones controlará el tiempo que pasa el cliente registrado en el laboratorio. Cuando se sobrepasa, informa al proveedor y deniega cualquier intento posterior de comunicación.			

Laboratorios: Gestor de ejecución			
IDENTIFICADOR	RUR-27	PRIORIDAD	Alta
FUENTE	RUC-31, RUC-36	NECESIDAD	Esencial
DESCRIPCIÓN			
Capa superior que administrará todas las ejecuciones de las acciones y de los limpiadores.			

Laboratorios: Tiempo máximo de una acción			
IDENTIFICADOR	RUR-28	PRIORIDAD	Media
FUENTE	RUC-47	NECESIDAD	Esencial
DESCRIPCIÓN			
El gestor de ejecución se encargará de controlar el tiempo máximo de cada acción, que será interrumpida en el instante que lo sobrepase.			

Herramientas: Aplicaciones			
IDENTIFICADOR	RUR-29	PRIORIDAD	Alta
FUENTE	RUC-9, RUC-36	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada herramienta contendrá un conjunto de aplicaciones de escritorio que ayudarán a manejar y configurar el hardware.			

Herramientas: Base de datos			
IDENTIFICADOR	RUR-30	PRIORIDAD	Alta
FUENTE	RUC-37	NECESIDAD	Esencial
DESCRIPCIÓN			
Toda la información de las acciones estarán disponible en la base de datos creada por el laboratorio en cada herramienta. Y podrá ser modificada por el usuario a través del mismo para la configuración de las mismas herramientas.			

Herramientas: Base de datos			
IDENTIFICADOR	RUR-31	PRIORIDAD	Alta
FUENTE	RUC-37, RUC-39	NECESIDAD	Esencial
DESCRIPCIÓN			
Toda la información de las acciones estarán disponible en la base de datos creada por el laboratorio en cada herramienta. Y podrá ser modificada por el usuario a través del mismo para la configuración de las mismas herramientas.			

Herramientas: Generación de datos			
IDENTIFICADOR	RUR-32	PRIORIDAD	Media
FUENTE	RUC-38, RUC-40	NECESIDAD	Esencial
DESCRIPCIÓN			
Los datos generados en la ejecución de un acción serán guardados en la base de datos para su posterior consulta.			

Herramientas: Servicios externos			
IDENTIFICADOR	RUR-33	PRIORIDAD	Alta
FUENTE	RUC-42	NECESIDAD	Esencial
DESCRIPCIÓN			
La abstracción mediante <i>sockets</i> de servicios externos de cada herramienta permitirá al usuario utilizarlos como servicios de la propia plataforma, con diferentes orígenes.			

Herramientas: Archivo de descripción			
IDENTIFICADOR	RUR-34	PRIORIDAD	Alta
FUENTE	RUC-43	NECESIDAD	Esencial
DESCRIPCIÓN			
Los archivos de descripción de herramientas seguirán dos esquemas, dependiendo del tipo de herramienta, con un formato XML.			

Herramientas: Herramientas de datos			
IDENTIFICADOR	RUR-35	PRIORIDAD	Media
FUENTE	RUC-45	NECESIDAD	Opcional
DESCRIPCIÓN			
Para los clientes, las herramientas de datos siempre estarán disponibles, independientemente del número de usuarios usándolas, y dispondrán de un mecanismo autónomo de <i>broadcast</i> de información.			

Herramientas: Limpiadores			
IDENTIFICADOR	RUR-36	PRIORIDAD	Media
FUENTE	RUC-46	NECESIDAD	Esencial
DESCRIPCIÓN			
Estas acciones serán especificadas en el archivo de descripción y tendrán preferencia en cuanto a las peticiones de ejecución normales. Una herramienta no se podrá ejecutar si ha ocurrido un error y no se ha ejecutado con anterioridad el limpiador.			

Libtool: Base de datos			
IDENTIFICADOR	RUR-37	PRIORIDAD	Alta
FUENTE	RUC-44	NECESIDAD	Esencial
DESCRIPCIÓN			
Esta abstracción permitirá la conexión a la base de datos de la herramienta para la lectura y escritura de información. El lenguaje de programación dependerá de la implementación de la propia herramienta.			

Libtool: Base de datos			
IDENTIFICADOR	RUR-38	PRIORIDAD	Alta
FUENTE	RUC-44	NECESIDAD	Esencial
DESCRIPCIÓN			
Esta abstracción permitirá la conexión a la base de datos de la herramienta para la lectura y escritura de información. El lenguaje de programación dependerá de la implementación de la propia herramienta.			

Cliente de escritorio: Función			
IDENTIFICADOR	RUR-39	PRIORIDAD	Media
FUENTE	RUC-8	NECESIDAD	Esencial
DESCRIPCIÓN			
La aplicación cliente principal será ejecutada como aplicación de escritorio que utilizará el servicio web proveedor para la interacción con la plataforma RLF.			

Cliente de escritorio: Ventanas de ejecución			
IDENTIFICADOR	RUR-40	PRIORIDAD	Alta
FUENTE	RUC-48, RUC-49	NECESIDAD	Esencial
DESCRIPCIÓN			
Por cada acción en ejecución, el cliente tendrá activa una ventana independiente con acceso de entrada y salida por teclado.			

Cliente de escritorio: Herramientas			
IDENTIFICADOR	RUR-41	PRIORIDAD	Alta
FUENTE	RUC-51, RUC-53	NECESIDAD	Esencial
DESCRIPCIÓN			
Por cada herramienta, la interfaz del cliente poseerá una pestaña que contendrá toda su información. Se podrán seleccionar de forma independiente para su posterior reserva.			

Cliente de escritorio: Conexión			
IDENTIFICADOR	RUR-42	PRIORIDAD	Alta
FUENTE	RUC-50	NECESIDAD	Esencial
DESCRIPCIÓN			
El cliente conservará la información de conexión durante toda la ejecución, sin ser necesario volver a conectarse para realizar múltiples reservas.			

Cliente de escritorio: Actualización			
IDENTIFICADOR	RUR-43	PRIORIDAD	Baja
FUENTE	RUC-50	NECESIDAD	Esencial
DESCRIPCIÓN			
Se ofrecerá al cliente la posibilidad de actualizar la información del número de herramientas como de su estado sin necesidad de cerrar la aplicación.			

Cliente web: Función			
IDENTIFICADOR	RUR-44	PRIORIDAD	Baja
FUENTE	RUC-18, RUC-52	NECESIDAD	Opcional
DESCRIPCIÓN			
Los clientes podrán acceder a la página web de la plataforma para comprobar el estado actual de las herramientas, si están disponibles, ocupadas o no conectadas. El formato deberá ser compatible para dispositivos con pantalla pequeña.			

Gestor de laboratorios: Función			
IDENTIFICADOR	RUR-45	PRIORIDAD	Baja
FUENTE	RUC-26	NECESIDAD	Esencial
DESCRIPCIÓN			
Para gestionar los laboratorios se proveerá de una aplicación de consulta con acceso por red con las acciones de armar, desarmar, parar, registrar y eliminar herramientas, y comprobar el estado del laboratorio.			

Registro: Logs			
IDENTIFICADOR	RUR-46	PRIORIDAD	Baja
FUENTE	RUC-34	NECESIDAD	Esencial
DESCRIPCIÓN			
Todo componente en la plataforma se servirá de un conjunto de registros para indicar errores y funciones realizadas.			

3.5.2. Requisitos no funcionales

Comunicaciones: Sistema escalable			
IDENTIFICADOR	RUR-1	PRIORIDAD	Media
FUENTE	RUR-2	NECESIDAD	Esencial
DESCRIPCIÓN			
La plataforma utilizará su propio sistema de comunicaciones y de altas de laboratorios, sin utilizar soluciones ya implementadas como Java RMI, CORBA o RPC.			

Comunicaciones: Protocolo JSON			
IDENTIFICADOR	RUR-2	PRIORIDAD	Media
FUENTE	RUR-1	NECESIDAD	Esencial
DESCRIPCIÓN			
El protocolo de comunicaciones estará basado en el formato JSON, previamente cifrado, que reduce de forma considerable los datos a enviar de los objetos a enviar.			

Comunicaciones: Protocolo JSON avanzado			
IDENTIFICADOR	RUR-3	PRIORIDAD	Media
FUENTE	RUR-1	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada objeto enviado entre módulos puede contener otros objetos como atributos también en formato JSON.			

Comunicaciones: Peticiones y respuestas			
IDENTIFICADOR	RUR-4	PRIORIDAD	Media
FUENTE	RUR-1	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada petición o respuesta tendrá asociado un número de operación que se incluirá en la cabecera del mensaje.			

Proveedor: Token de acceso y uso			
IDENTIFICADOR	RUR-5	PRIORIDAD	Media
FUENTE	RUR-3	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada cliente tiene asociado un atributo con un límite de tiempo que será generado cada vez que se conecte para reducir el tráfico de comunicaciones y sólo tener que identificarse mediante esa cadena de caracteres. Cuando se desconecta será invalidado.			

Proveedor: Servicio Web J2EE			
IDENTIFICADOR	RUR-6	PRIORIDAD	Alta
FUENTE	RUR-4	NECESIDAD	Esencial
DESCRIPCIÓN			
El proveedor será un conjunto de dos servicios web publicados en una plataforma Tomcat, además de la base de datos MySQL. El servicio web principal llamado RLF_Provider y el secundario, RLF_Monitor.			

Laboratorios: Máquina virtual de Java			
IDENTIFICADOR	RUR-7	PRIORIDAD	Alta
FUENTE	RUR-5	NECESIDAD	Esencial
DESCRIPCIÓN			
Los laboratorios estarán implementados en Java y utilizarán drivers de conexión a las bases de datos SQLite.			

Laboratorios: Archivos ejecutables			
IDENTIFICADOR	RUR-8	PRIORIDAD	Alta
FUENTE	RUR-5	NECESIDAD	Esencial
DESCRIPCIÓN			
Para configurar un laboratorio se proveerá de un fichero de configuración junto con el archivo ejecutable.			

Laboratorios: Ficheros fuente			
IDENTIFICADOR	RUR-9	PRIORIDAD	Alta
FUENTE	RUR-5	NECESIDAD	Esencial
DESCRIPCIÓN			
Todo recurso necesario para la ejecución de los laboratorios será contenida en un directorio que deberá acompañarse con el ejecutable.			

Laboratorios: Conexiones			
IDENTIFICADOR	RUR-10	PRIORIDAD	Alta
FUENTE	RUR-6	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada laboratorio posee un puerto de conexión con el proveedor, otro de notificaciones para el cliente, uno de recepción de peticiones y por último, el puerto de control de mantenimiento.			

Laboratorios: Asincrónos			
IDENTIFICADOR	RUR-11	PRIORIDAD	Alta
FUENTE	RUR-6	NECESIDAD	Esencial
DESCRIPCIÓN			
Por cada petición de ejecución se enviará en qué instante empezará a funcionar, ya que es posible que no sea instantáneo si hay sobrecarga en el laboratorio.			

Laboratorios: Ejecuciones			
IDENTIFICADOR	RUR-12	PRIORIDAD	Alta
FUENTE	RUR-7	NECESIDAD	Esencial
DESCRIPCIÓN			
El laboratorio se servirá de peticiones al sistema y señales para ejecutar las acciones, así como de un servicio de <i>pipeling</i> para redirigir la entrada y la salida de la propia acción.			

Laboratorios: Colas de ejecución			
IDENTIFICADOR	RUR-13	PRIORIDAD	Alta
FUENTE	RUR-8	NECESIDAD	Esencial
DESCRIPCIÓN			
Cada laboratorio poseerá un contenedor de acciones en ejecución, donde está limitado por su configuración inicial. Todas las peticiones de ejecución, si los contenedores están completos, serán retrasadas.			

Framework: Parada			
IDENTIFICADOR	RUR-14	PRIORIDAD	Alta
FUENTE	RUR-9	NECESIDAD	Esencial
DESCRIPCIÓN			
Aunque no se impondrá ninguna norma a los desarrolladores en cuestión de bloqueo de ficheros o de programas por el motivo de las paradas inesperadas, se instará en la manera de no mantener ficheros abiertos ni componentes susceptibles de ocasionar errores en el futuro.			

Cliente de escritorio: Máquina virtual de Java			
IDENTIFICADOR	RUR-15	PRIORIDAD	Alta
FUENTE	RUR-4	NECESIDAD	Esencial
DESCRIPCIÓN			
La aplicación de escritorio será implementada en Java para su portabilidad y se requerirá configurar mediante un archivo adjuntado con el ejecutable.			

Cliente web: Múltiple conexión			
IDENTIFICADOR	RUR-16	PRIORIDAD	Alta
FUENTE	RUR-8	NECESIDAD	Esencial
DESCRIPCIÓN			
Mediante el uso de <i>tokens</i> almacenados en la base de datos del proveedor una desconexión no es obligatoria en el cliente web.			

Capítulo 4

Diseño

En el siguiente capítulo se tratará el diseño de la plataforma RLF, desde los primeros conceptos hasta la implementación. También se describirán cada una de las partes involucradas en el funcionamiento del sistema.

4.1. Arquitectura del sistema

El primer paso del diseño es, mediante la recopilación de información para la anterior parte de análisis, la definición de la arquitectura del sistema, es decir, sus componentes principales y la forma de interacción entre ellos. Para ello se ha valido de un esquema (figura 4.1) que los identifica. Estos componentes forman una estructura general de cliente/servidor o servidores jerárquicos, aunque cada componente implementa en su interior otro tipo de arquitecturas que se definirán más adelante.

Siguiendo el orden de la parte superior a la inferior de dicha figura, obtenemos los siguientes componentes:

La plataforma se asienta sobre una base de datos central que contiene la información general del sistema. Con ella, los diferentes componentes obtienen la información actualizada de las acciones tomadas y del estado global. Los laboratorios y el servidor web son los componentes directamente relacionados con esta base.

El servidor web actúa como la puerta de acceso a la plataforma RLF. Está compuesto por dos servicios web (servicio proveedor y servicio monitor) y una página web para el acceso a uno de ellos. El servicio proveedor funciona como componente principal, y se encarga de tratar las peticiones por parte de los clientes y darles acceso a los distintos laboratorios.

Los laboratorios conforman el esqueleto de la plataforma y funcionan como gestores de peticiones de ejecución para las herramientas. Contienen una base de datos local propia de cada laboratorio, pero también se sirven de la base de datos central para informar del estado al resto de componentes. Primero, el servidor web indica al laboratorio los clientes válidos permitidos en el sistema, después son los propios clientes los que contactan con él para que atienda sus peticiones y empieza el intercambio de mensajes.

También, laboratorios son gestionados externamente por los administradores mediante un componente remoto (gestor de laboratorios) que no tiene interacción con otros componentes de la plataforma. Con él se puede arrancar los laboratorios, registrar herramientas y otras tareas de mantenimiento.

Al otro lado del sistema se encuentra el cliente de escritorio, que permite acceder al usuario cliente a la plataforma. Provee de la interfaz necesaria para manejar cada herramienta y realizar las peticiones. Está relacionado con la interfaz que implementa el servicio proveedor ya que realiza el acceso a través de ella.

Como casos especiales, se encuentran las aplicaciones externas y los servicios externos. No están directamente relacionados con la plataforma ni implementados en la misma, aunque son parte importante para el desarrollo de herramientas. Contienen su propias comunicaciones y es el usuario cliente el encargado de acceder a los servicios mediante estas aplicaciones y son las herramientas las encargadas de activarlos.

Por último se tienen las herramientas y sus datos, que funcionan como dos componentes separados, ya que en realidad, esos datos (contenidos en bases de datos locales) son los intermediarios entre los laboratorios y las herramientas. Mediante funciones del sistema, los laboratorios arrancan las aplicaciones de las herramientas, pero es, mediante dichas bases de datos, donde intercambian información de forma asíncrona.

En el siguiente apartado se explicarán con detalle cada uno de los componentes, indicando sus relaciones y sus funciones.

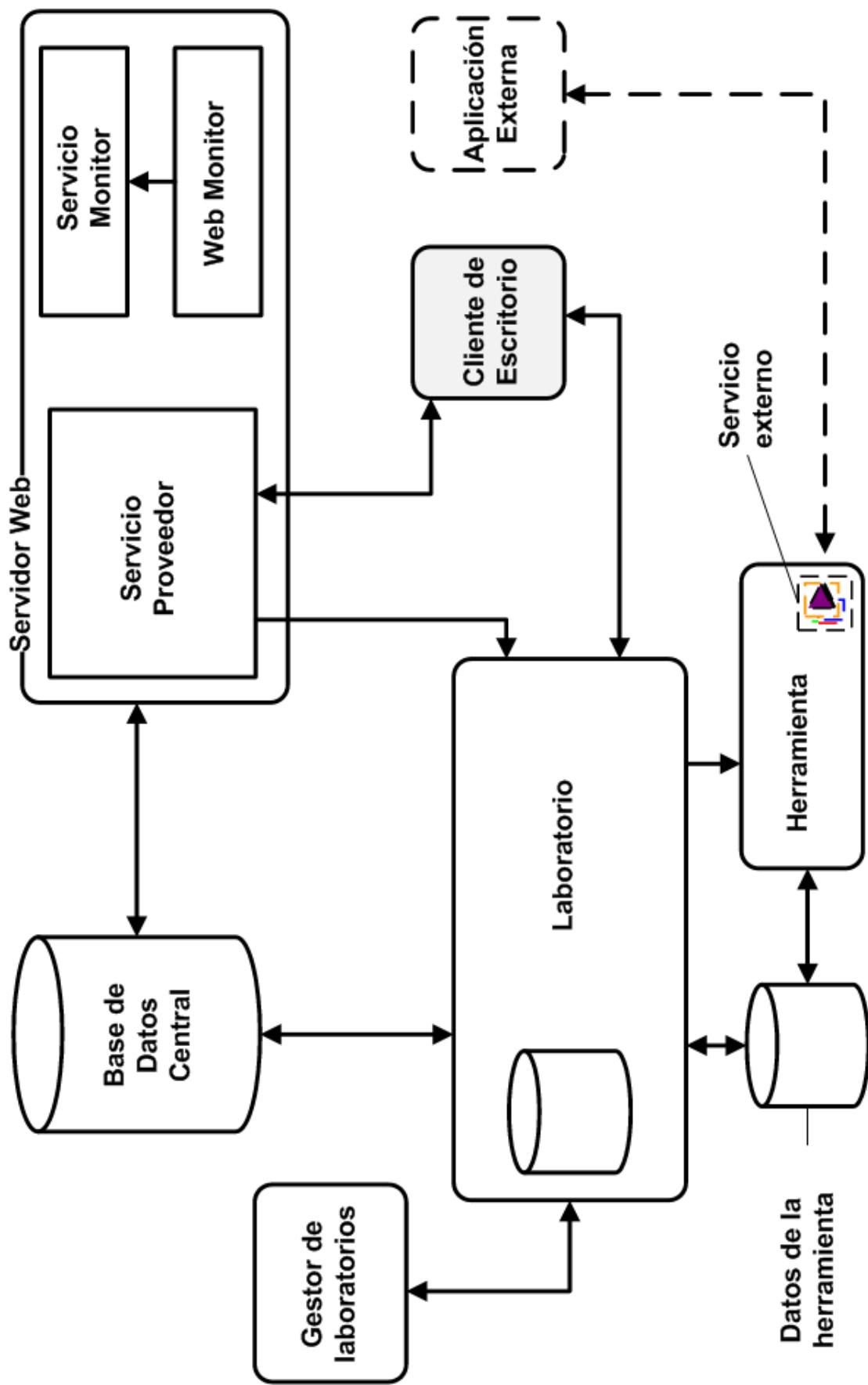


Figura 4.1: Arquitectura de RLF según sus componentes.

4.2. Diseño de componentes

Para comprender el conjunto de la plataforma, se dispone a explicar cada componente por separado (y sus subcomponentes), incluyendo las relaciones que tiene con otros. Las definiciones que se muestran a continuación responden al estándar *UML 1.1*. Se establece el mismo orden que se siguió en el apartado anterior.

4.2.1. Base de datos central

Compone el principal almacén de datos de toda la plataforma RLF. Contiene un conjunto de tablas que organizan la información para que sea accesible mediante el servicio proveedor y los distintos laboratorios. Como se puede ver en la figura 4.2 (modelo entidad-relación), contiene las siguientes estructuras:

- Una tabla para la información de cada laboratorio. Incluye todos los elementos necesarios para la conexión y su estado principal.
- Tablas de información de usuario, diferenciando entre los usuarios clientes y los usuarios administradores.
- La información principal de cada herramienta está contenida en la tabla *tool* junto con su estado y en qué laboratorio se encuentra.
- Un registro temporal por cada herramienta reservada, relacionando el cliente con la herramienta, y la fecha de la propia reserva.
- La descripción de cada herramienta, formada por sus constantes, parámetros, atributos, acciones y servicios externos están almacenados en sus propias tablas.

Cada cambio que se realiza en esta base de datos de forma automatizada, como son procesos de conexión, desconexión, reserva, activación de laboratorios, etc. están agrupados en procedimientos almacenados (interfaz *iBase*)¹ para mejorar la seguridad y la fiabilidad del sistema. El resto de cambios concretos, como la insercción de una nueva herramienta, se realizan mediante el procedimiento normal. Se añaden también un conjunto de disparadores e índices para mejorar la velocidad de la base de datos.

Como se ve en la figura 4.3 la base de datos interactúa directamente con el servidor web, y además con los laboratorios.

4.2.2. Servidor Web

Está compuesto por dos servicios y una página web de acceso a la plataforma (figura 4.3) y el componente de comunicaciones.

NOTA: El componente de comunicaciones es descrito al final de este sección, ya que hay multitud de otros componentes que lo usan a pesar de tener la misma implementación (véase sección ??). Aunque sea este componente el que se encargue de la recepción y envío de datos a nivel interno, las interfaces las proveen el resto de componentes, por lo que sólo se considera una capa a bajo nivel en la comunicación y será representado como tal.

¹Conjunto de instrucciones en una base de datos donde no hay valor de retorno.

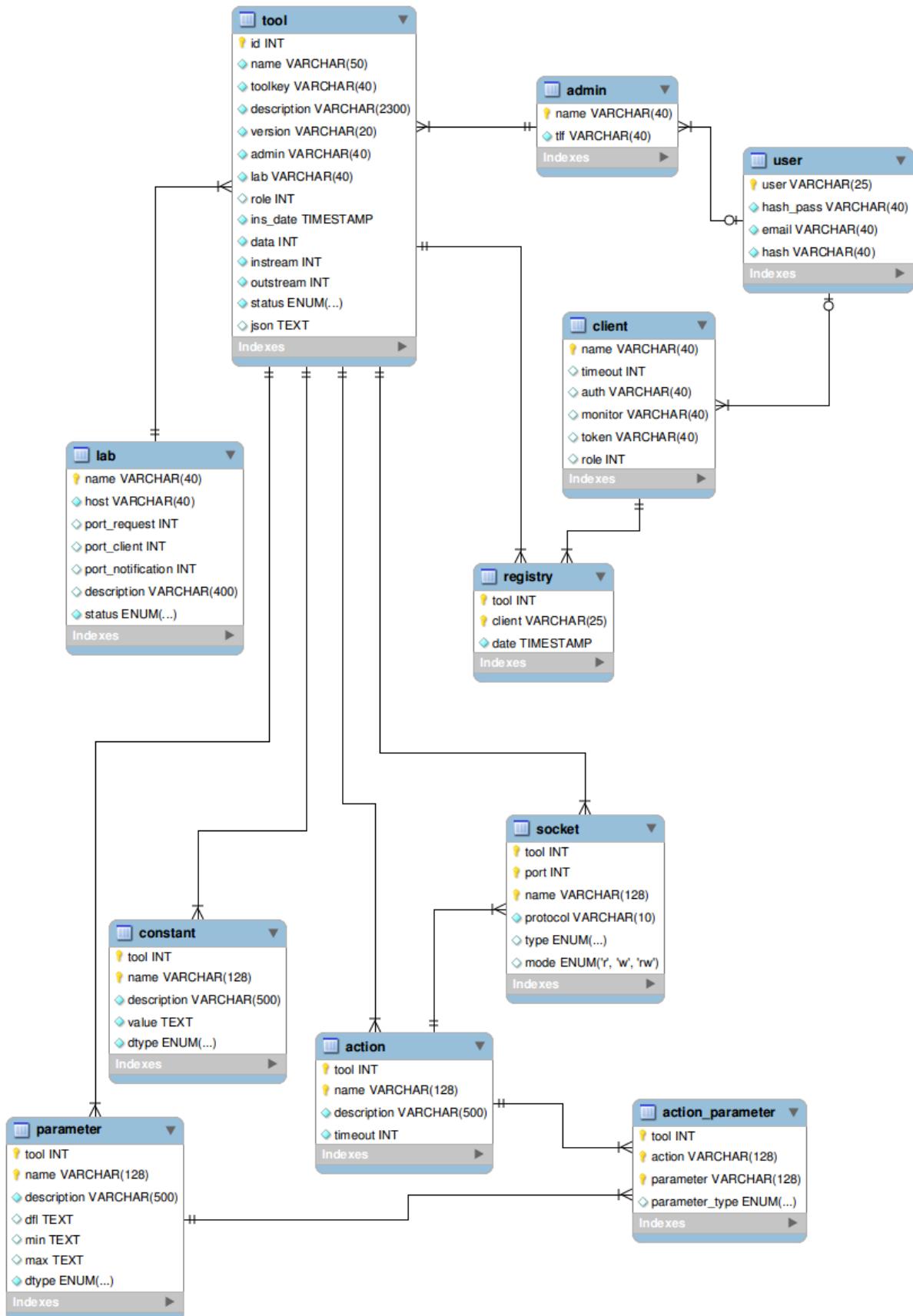


Figura 4.2: Estructura de la base de datos central.

Servicio Monitor

Utiliza la información contenida en la base de datos central para que mediante la interfaz de acceso *iMonitor* se obtenga el estado de las herramientas. Los métodos de esta interfaz son:

- Autentificación al monitor.
- Desconexión del monitor.
- Obtención del estado en forma de lista con el identificador de la herramienta, su nombre y su estado.

Página Web

Presenta una interfaz accesible a través de un navegador para utilizar el servicio del monitor. Está preparada para mantener la conexión abierta aunque se deje de atender la página. Se compone de una sección para la autentificación y una lista de herramientas y sus estados.

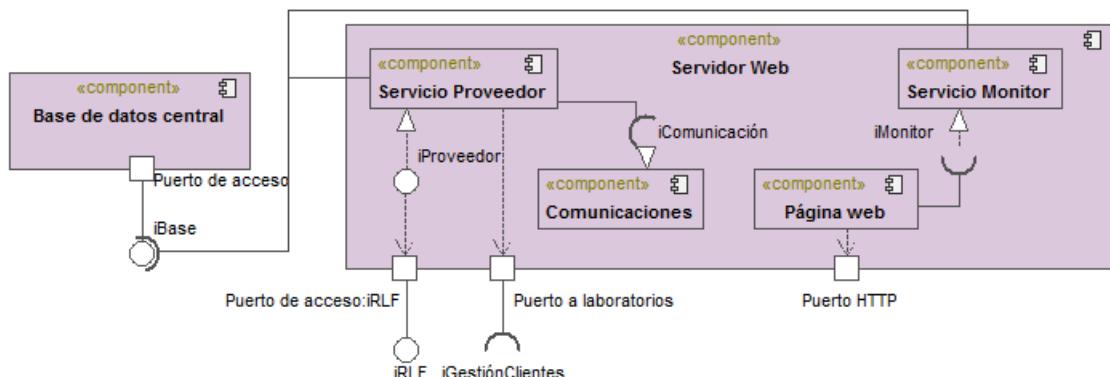


Figura 4.3: Servidor Web y Base de datos central.

Servicio Proveedor

Representa la interfaz abstracta de toda la plataforma para el acceso. Se puede decir que es el coordinador de todo el sistema, que permite a los clientes localizar (con la interfaz *iRLF*) los laboratorios e iniciar las conexiones. Se sirve de los datos de la base central mediante la interfaz *iDatos* y comunica a los laboratorios cuándo serán accedidos por los clientes.

Las acciones de la interfaz *iRLF* se listan a continuación:

- Conexión al sistema. Cuando un cliente se conecta, genera el *token* correspondiente y se lo envía para la identificación de todas las operaciones siguientes.
- Desconexión. Invalida dicho *token* de acceso y libera todas las herramientas que pudiera tener reservadas.
- Obtiene el estado actual de las herramientas a las cuales el cliente tiene acceso.

- Describe las herramientas las cuales pueden ser accedidas por el cliente. La descripción es un mensaje codificado en JSON precalculado con todos los atributos, constantes, acciones y parámetros de las herramientas.
- Reserva las herramientas seleccionadas. Avisa además a los laboratorios para que inicien el contador de tiempo del cliente y devuelve a este la información de dónde se encuentra cada herramienta.

4.2.3. Laboratorio

Este complejo componente gestiona todas las peticiones a las herramientas por parte de los clientes. Está compuesto por tres subcomponentes principales (*Kernel*, gestor de comunicaciones y gestor de ejecución) y por otros tres secundarios (comunicaciones, gestor de herramientas y base de datos) como se puede ver en la figura 4.4.

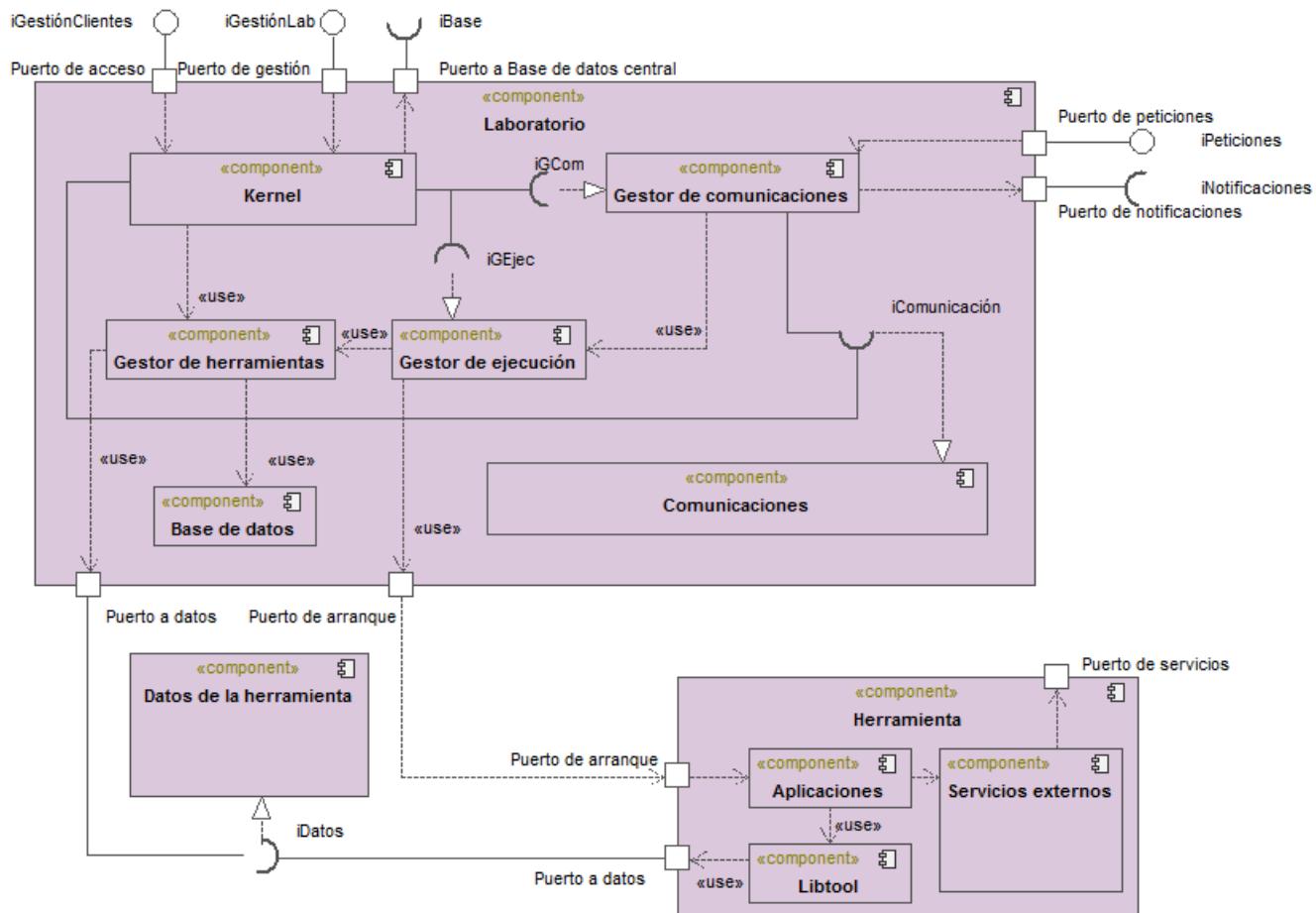


Figura 4.4: El laboratorio y las herramientas.

Kernel

Es el núcleo del laboratorio, que atiende las peticiones del proveedor (recibidas por la interfaz *iGestiónClientes*) y las de gestión (interfaz *iGestiónLab*) desde el programa administrador. Tam-

bién controla la ejecución y el arranque de los otros dos subcomponentes principales. Es el único componente que tiene acceso a la base de datos central para poder cambiar su estado.

Cuando el laboratorio es armado, el *Kernel* activa a los gestores de comunicaciones y de ejecución que se componen de hilos independientes. Usa al gestor de herramientas cuando recibe una petición de registro o de parada.

Las principales acciones que se concentran en este componente son:

- Iniciar el laboratorio. Avisa al proveedor que ha sido arrancado.
- Parar el laboratorio. Sólo se realiza bajo petición de un administrador y cuando no está armado.
- Armar y desarmar. En el estado de armado, se pueden recibir peticiones de los clientes y ejecutar las diferentes acciones.
- Obtener el estado global del laboratorio y sus herramientas. Sólo es utilizado por los administradores.
- Avisar al proveedor (mediante la base de datos) que el tiempo de un cliente ha terminado.
- Iniciar el proceso de registrar o eliminar una herramienta. Sólo en el caso de no estar armado.
- Realizar una parada de emergencia completa.

Gestor de ejecución

Contiene todos los elementos necesarios para organizar las peticiones (enviadas por el gestor de comunicaciones) de ejecución de las diferentes acciones de las herramientas. Controla el tiempo máximo de cada ejecución y realiza el envío de datos entrantes y salientes a modo de *stream* a los clientes, sin formato establecido. Cuando una acción es terminada o interrumpida, genera el mensaje a enviar al cliente. También indica al gestor de herramientas qué cambios se deben hacer en la base de datos de la herramienta antes de ejecutar una determinada acción, y recoger estos cuando la acción haya terminado.

Gestor de comunicaciones

Es el enlace con los clientes. Recibe las peticiones de ejecución por la interfaz *iPeticiones* y las encola en el gestor de ejecución. Cuando ocurre cualquier evento con esas acciones, se lo comunica al cliente mediante el puerto de notificaciones. Con esto se consigue un tratamiento asíncrono. Por último, todos los cronómetros de cada cliente los contiene este gestor, así como la información de cada usuario válido en el sistema.

Gestor de herramientas

Componente encargado en la gestión de las herramientas activas en el laboratorio, así como el manejo de su información mediante el subcomponente de la base de datos propia. Valida todas las configuraciones XML introducidas por los administradores y lee, cuando el laboratorio es arrancado, las ya existentes. No funciona como un hilo a parte, si no como una biblioteca de métodos.

Además, toda la información en tiempo real es insertada o leída de la base de datos de la herramienta como si fuese la propia aplicación.

Base de datos

Funciona como un almacén de los datos de acceso de las herramientas. Conforma una estructura simple de una única tabla con los elementos más importantes, siendo la mostrada en la figura 4.5

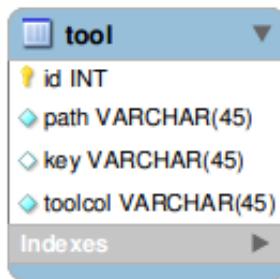


Figura 4.5: Estructura de la base de datos del laboratorio.

4.2.4. Datos de la herramienta

Ha sido concebido como una interfaz entre la aplicación y el laboratorio, pero de forma asíncrona. De esta forma, la herramienta es tratada como un objeto externo. El contenido es la estructura directa de la herramienta (figura 4.6).

La correspondencia de cada tabla es similar a la base de datos central, que contiene la misma información, a excepción de los atributos, que conforman las características propias de cada herramienta (en el servidor central están representadas como columnas de la tabla principal), y las excepciones y estados de las diferentes acciones.

Está implementada de forma sencilla para obtener una mayor velocidad de escritura y lectura. Es por ello que todas las claves ajenas y otras restricciones han sido desactivadas, y son comprobadas de forma externa.

4.2.5. Herramienta

Este componente contiene la arquitectura del *framework* de RLF. No se definirán las diferentes herramientas entregadas si no la forma que deben tener las aplicaciones.

Aplicaciones

Son el conjunto de acciones a ejecutar. Están implementadas según las normas del *framework* (ver Manual de desarrollo) y sólo son dependientes del otro componente *Libtool* y de los servicios externos si se requieren.

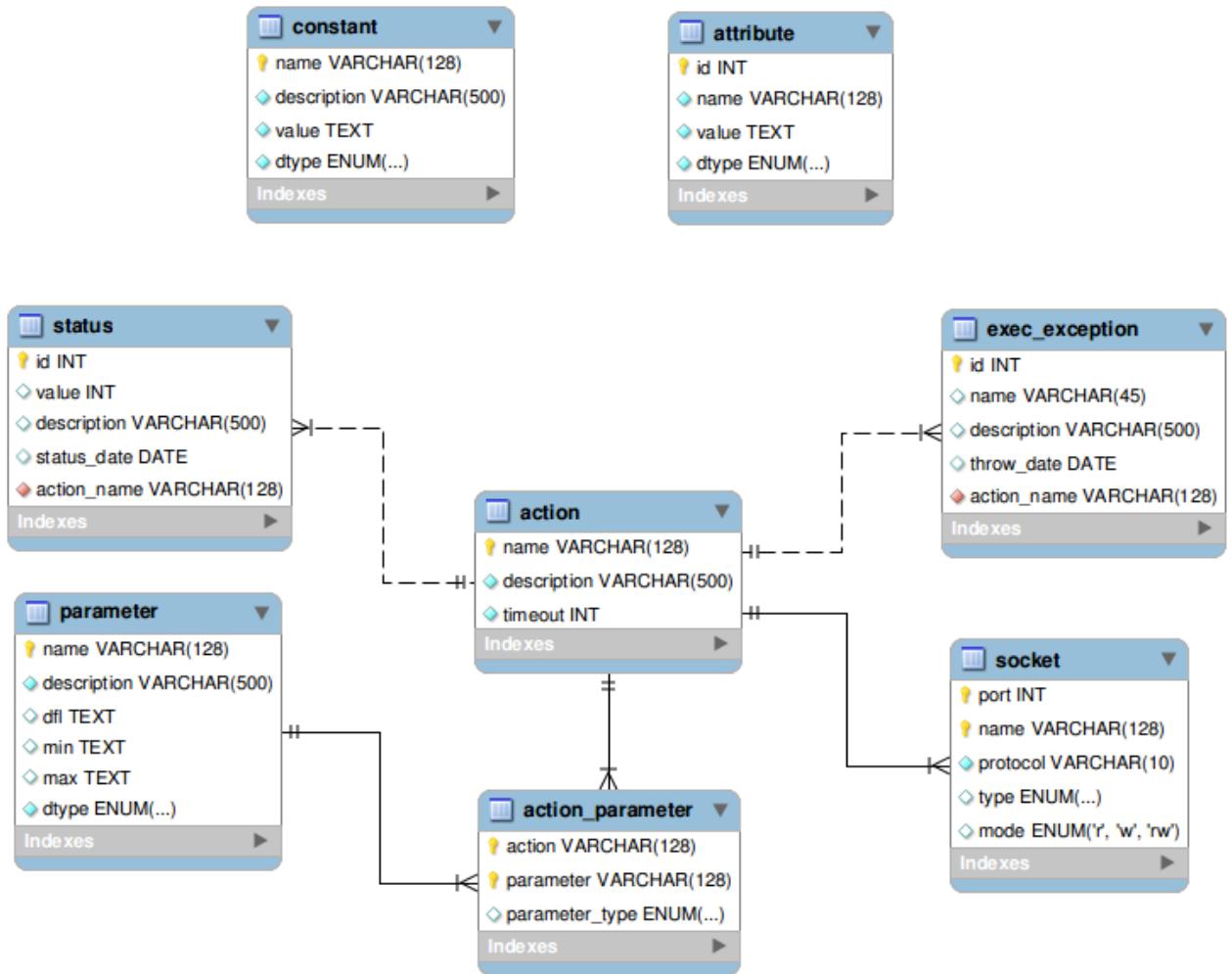


Figura 4.6: Estructura de la base de datos de la herramienta.

Servicios externos

Poseen un acceso como puerto de comunicaciones donde los clientes pueden conectarse, pero no es gestionado por el laboratorio ni por el proveedor central.

Libtool

Funciona como una biblioteca de funciones para la transformación en objeto. Así se pueden obtener los datos escritos por el laboratorio. Las funciones que provee este componente son las siguientes:

- Conexión y desconexión con el laboratorio. Si una aplicación requiere lectura o escritura de sus valores, es necesario estar conectado.
- Lectura y escritura de parámetros.
- Lectura de atributos. Las herramientas no pueden escribir sus propios atributos, ya que estos son definidos por el laboratorio poseedor.

- Lectura de constantes.
- Generación de excepciones.
- Escritura del estado de la acción.

4.2.6. Gestión de laboratorios

Este componente forma una aplicación usada por los administradores de la plataforma RLF y forma una arquitectura de capas. Cada capa se sirve de la interfaz de la anterior, como se puede ver en la figura 4.7.



Figura 4.7: Estructura de capas del gestor de laboratorios.

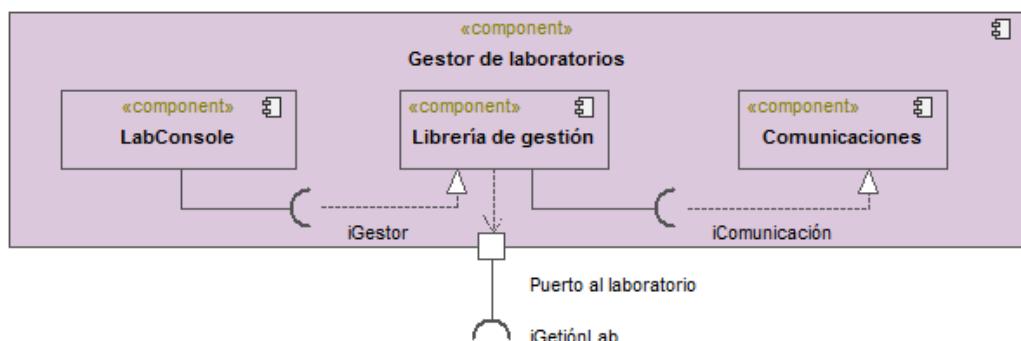


Figura 4.8: Componentes del gestor de laboratorios.

LabConsole

Actúa como interfaz textual para enviar peticiones a los laboratorios. Utiliza el conjunto de métodos proveídos por la librería de gestión (interfaz *iGestor* en la figura 4.8).

Librería de gestión

Incluye los parámetros necesarios introducidos mediante la interfaz textual en los mensajes para las peticiones de mantenimiento. Después serán enviadas a los laboratorios, obteniendo una respuesta y descodificándola.

4.2.7. Cliente

El cliente se muestra como una aplicación de escritorio en Java con una interfaz implementada en Swing. Debe estar configurada para acceder al servidor central de RLF. Al igual que los laboratorios tiene diferentes subcomponentes como hilos independientes (figura 4.9).

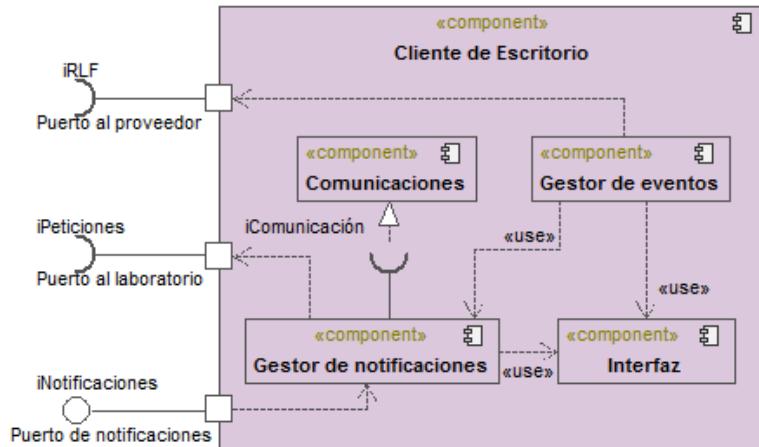


Figura 4.9: Componentes del sistema de comunicaciones de la plataforma RLF.

Gestor de eventos

Es el componente encargado de “escuchar” los eventos de la interfaz generados por el usuario para llevar a cabo acciones con el proveedor o las herramientas. Establece una conexión directa con la interfaz *iRLF* generada por el servicio proveedor. Los mensajes enviados con este componente no tienen el formato de dato interno, si no que utiliza el propio formato HTTP + SOAP que se usa en los servicios web.

Gestor de notificaciones

Todas las peticiones que se envían a los laboratorios y las notificaciones que se reciben de estos pasan a través de este componente, que usa el protocolo definido en el componente Comunicaciones. Se sirve de la interfaz *iPeticiones* e implementa *iNotificaciones* con una función de inicio de la acción y otra notificación de tiempo excedido. Actúa como puerto de escucha y posee ejecución paralela al gestor de eventos.

Interfaz

Provee al usuario cliente de todos los mecanismos necesarios para la realizar el acceso completo a la plataforma.

4.2.8. Comunicaciones

La implementación de este componente, usado en los otros, forma un protocolo de comunicación añadido al modelo TCP/IP (véase sección 2.7.1) que es la base de las comunicaciones internas de la plataforma en forma de petición/respuesta. Además también tiene la función de librería para enviar y recibir datos, obteniendo directamente la información importante y *parseada*.

En la figura 4.10 se puede ver la disposición de sus subcomponentes.

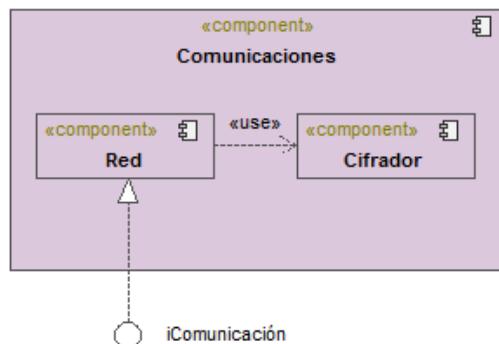


Figura 4.10: Componentes del cliente de escritorio.

Cifrador

Biblioteca con los métodos necesarios para cifrar datos textuales con el algoritmo *base64*² y la obtención de valores mediante el algoritmo resumen *SHA-1*.

Red

Implementa el protocolo de mensajes RLF. Se basa en un sistema de capas que convierte objetos JSON en mensajes reconocibles por el sistema. Los mensajes están formados por un identificador que corresponde a la operación a realizar y un conjunto de atributos. En la primera capa (capa de objeto) se obtienen esos atributos y se “aplanan” (un atributo puede ser a la vez otro objeto JSON). En la capa de mensaje se cifra según el algoritmo *base64* para no perder caracteres del mensaje y se añade la cabecera, que es el tamaño total del mensaje (ver figura 4.11). El proceso a la inversa también ha sido implementado.

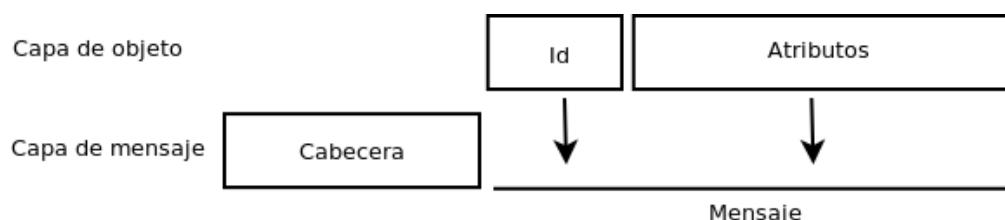


Figura 4.11: Protocolo RLF.

NOTA: Para obtener más información sobre los identificadores de las peticiones, consultar el código adjuntado donde se encuentra la lista exhaustiva de los mismos.

²Base 64 es un sistema de numeración posicional que usa 64 como base. Es la mayor potencia de dos que puede ser representada usando únicamente los caracteres imprimibles de ASCII.[24]

Esos mensajes son enviados y recibidos mediante *sockets* que provee el sistema operativo. Los métodos son los siguientes:

- Conexión y desconexión de *sockets* con direcciones remotas.
- Envío de una petición/respuesta con un *socket* conectado.
- Recepción de una petición/respuesta por un *socket* a la escucha.

4.3. Despliegue de componentes

Como última sección se presenta el despliegue de los componentes en los distintos dispositivos o máquinas que son necesarios para el funcionamiento del sistema, también se incluyen los entornos de ejecución necesarios. Se puede ver en la figura 4.12, siguiendo el estándar *UML 1.1* el conjunto de componentes.

Este esquema es orientativo ya que debido a la versatilidad de la plataforma, todos los componentes pueden estar en el mismo nodo, o combinaciones de ellos. En cambio, la multiplicidad de cada relación entre los nodos sí que es obligatoria, haciendo imposible tener en una misma plataforma RLF dos servidores centrales, o que un mismo cliente esté conectado a dos plataformas con la misma aplicación cliente. Para más información consultar el Manual de despliegue contenido en esta misma documentación.

NOTA: Los elementos “artifact” representan las aplicaciones finales que contienen a los componentes, referenciados mediante la relación “manifest”.

En primera instancia se define el servidor central con la base de datos de RLF sobre el entorno MySQL y los servicios en la plataforma web Tomcat que soporta J2EE (ver sección 2.5.1). A él se pueden conectar dispositivos portátiles con navegadores (como entorno de ejecución) para acceder a la página web de monitorización que provee el servidor central.

Después se encuentran los nodos que contienen los laboratorios. Mediante la máquina virtual de java se ejecuta la aplicación *Lab* que contiene el componente principal del laboratorio. Con SQLite se gestionan los datos de las herramientas. Y, dependiendo del tipo de aplicación, se ejecutan las herramientas en el sistema operativo, al igual que los servicios externos. Todos los elementos necesarios para la ejecución de las acciones de las herramientas tienen que estar contenidos en estos mismos nodos.

El nodo cliente contiene todas las aplicaciones necesarias para poder utilizar los servicios externos, además de la máquina virtual Java para poder ejecutar el cliente de escritorio, llamado “RLF_Client”. Este nodo debe tener conexión con la red del servidor central así como acceso a los laboratorios.

Por último, el gestor de laboratorios se encuentra en el terminal de acceso a la red de los laboratorios. Al ser una aplicación Java, necesitará la máquina virtual para que funcione.

Los siguientes pasos en el desarrollo de la plataforma incluyen la implementación y las pruebas. No se ha dedicado un capítulo exclusivo para la implementación debido a que en el material entregado se encuentran todas las referencias y definiciones necesarias para comprenderla. Se podrá encontrar en el código, además, los algoritmos y estructuras necesarias para el funcionamiento de la plataforma RLF.

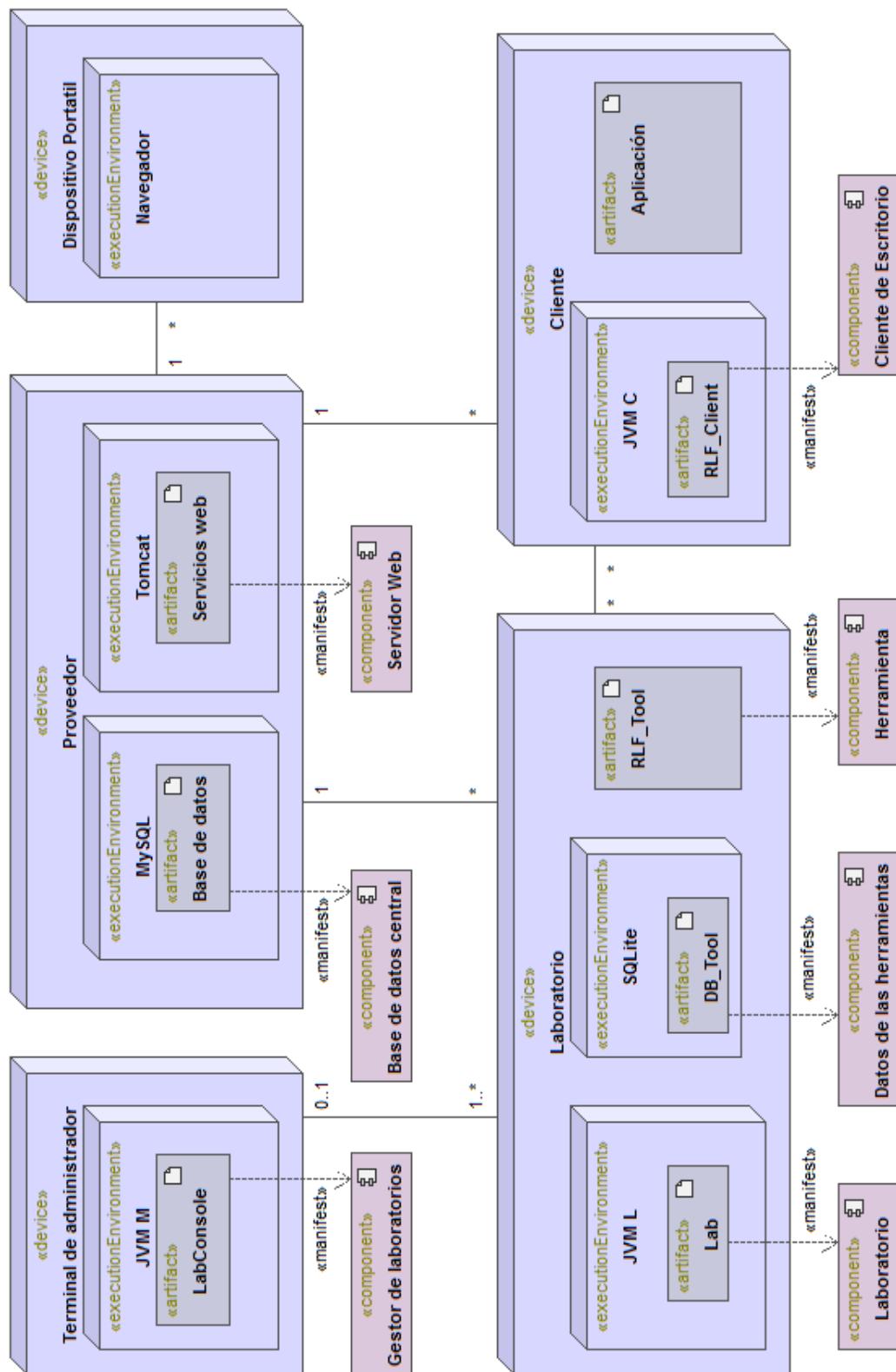


Figura 4.12: Despliegue de RLF por componentes.

Capítulo 5

Pruebas

Las pruebas de verificación del sistema implementado es una parte importante del desarrollo *software*. En este capítulo se incluyen las más destacadas que validan el conjunto de la plataforma RLF.

Las pruebas que se muestran a continuación han sido seleccionadas del conjunto global que se realizaron en la implementación del proyecto. Debido al gran número de estas no se ha podido incluir todas.

5.1. Configuraciones

Se han establecido un conjunto de configuraciones de la arquitectura para las distintas pruebas, con lo que se consigue verificar que independientemente de la estructura del sistema, los resultados sean los mismos. Estas configuraciones tienen un nombre asociado que se especificará en cada una de las pruebas. A continuación se muestra cada configuración, con el número de componentes que intervienen:

Nombre	L	CE	CW	Disposición
Básica	1	2	1	Todos los componentes en la misma máquina, excepto un CE y el CW que está en otra máquina en red.
Común	2	1	1	El P y un L se disponen en una máquina, el otro L en un sistema Windows en red local con el CE. El CW en un <i>smartphone</i> con 3G.
Remota	2	2	2	Cada componente está en una máquina distinta, pero el P y los L están conectados en red local, mientras que los CE y CW se conectan mediante Internet.
Educativa	5	2	1	P se encuentra en una red externa, los L están conectados en red local, y los CE y CW en otra red externa. Algunas herramientas han sido duplicadas.
Industrial	5	5	2	Cada componente se encuentra en un nodo distinto, distribuidos entre redes locales e Internet. Algunas herramientas han sido duplicadas.

Tabla 5.1: Configuraciones de las pruebas.

NOTA 1: Los *laboratorios* (L) son numerados secuencialmente de forma 1, 2, 3... Los *clientes de escritorio* (CE) se corresponden con el alfabeto en mayúsculas: A, B, C... Y por último, los *clientes web* (CW) corresponden con el alfabeto en minúsculas: a, b, c...

NOTA 2: Todas las configuraciones se componen de un solo proveedor y monitor (P), y de las cinco herramientas entregadas con el proyecto, además de un número concreto de laboratorio, clientes de escritorio y clientes web.

5.2. Pruebas de verificación de funcionalidad

Estas pruebas han sido concebidas para verificar todas las funciones y opciones que aporta la plataforma RLF. Han sido realizadas en un entorno controlado y comprobando que la respuesta del sistema a determinadas acciones era la esperada.

Durante el desarrollo del proyecto, estas pruebas se repitieron obteniendo diferentes errores que fueron corregidos hasta obtener los resultados esperados.

PRUEBA 1					
ID	PF-A	Configuración	Básica	Resultado	Válido
Descripción					
Se accede desde la red externa mediante el CE para reservar todos las herramientas disponibles, ejecutando a la vez dos de ellas. Después se comprueban el estado de las herramientas mediante el CW, donde todas están ocupadas.					
Acciones					
<ol style="list-style-type: none"> 1. Reserva de todas las herramientas. 2. Ejecución de dos de ellas. 3. Comprobación por la web. 4. Obtención de las herramientas ocupadas. 					

PRUEBA 2					
ID	PF-B	Configuración	Común	Resultado	Válido
Descripción					
Se comprueba ejecución de herramientas en diferentes L desde el mismo CE.					
Acciones					
<ol style="list-style-type: none"> 1. Reserva de una herramienta de cada laboratorio. 2. Ejecución de las acciones de las herramientas. 3. Desconexión. 					

PRUEBA 3				
ID	PF-C	Configuración	Resultado	Válido
Descripción				
El objetivo es la comprobación de los limpiadores de cada herramienta. Para ello es requerido un cierre forzado de una acción.				
Acciones				
<ol style="list-style-type: none"> 1. Reserva de una herramienta. 2. Ejecución de una acción. 3. Forzado de cierre desde el CE. 4. Comprobación del <i>log</i> del laboratorio responsable. 5. El limpiador se ejecuta a los pocos segundos de cerrar la acción. 				

PRUEBA 4				
ID	PF-D	Configuración	Resultado	Válido
Descripción				
Se verifica que las herramientas de datos se ejecuten por varios usuarios y que sólo se ciernen cuando todos se han desconectado.				
Acciones				
<ol style="list-style-type: none"> 1. Reserva de una herramienta con el CE A. 2. Reserva de la misma herramienta con el CE B. 3. El CE B ejecuta la acción de la herramienta. 4. Al cabo de un tiempo, el CE A también. 5. El CE B cierra la ejecución y se desconecta. 6. Por último, CE A para la acción. 7. Se comprueba el <i>log</i> de L y se verifica que la acción se ha detenido correctamente cuando el CE A se desconectó. 				

PRUEBA 5				
ID	PF-E	Configuración Básica	Resultado	Válido
Descripción				
Comprobación de la liberación de herramientas una vez que un usuario se ha desconectado, y su posterior uso por parte de otro usuario.				
Acciones				
<ol style="list-style-type: none"> 1. Reserva de un conjunto de herramientas con el CE A. 2. Conexión del CE B. 3. El CE B comprueba el estado de las herramientas en uso por CE A. 4. CE A cierra algunas acciones forzándolas y otras con la finalización normal, después se desconecta. 5. El CE B reserva las herramientas que CE A ha liberado. 6. Ejecuta las acciones de esas herramientas. 7. Debe esperar para el inicio de algunas acciones, correspondientes con las que se ha forzado su cierre por CE A, ya que se están ejecutando sus limpiadores. 8. Se comprueba el estado de las herramientas en el CW. 9. CE B libera sus herramientas después de utilizarlas. 10. Se vuelve a comprobar el estado de las herramientas mediante el CW y se verifica que todas están libres. 				

PRUEBA 6				
ID	PF-F	Configuración Básica	Resultado	Válido
Descripción				
Verificación de los roles establecidos para los distintos usuarios y las herramientas que pueden utilizar.				
Acciones				
<ol style="list-style-type: none"> 1. Se registran las herramientas con distintos roles. 2. Se crea un usuario con un rol específico. 3. El usuario, mediante el CE y el CW comprueba que sólo tiene acceso a las herramientas con su rol o inferior. 				

PRUEBA 7				
ID	PF-G	Configuración	Resultado	Válido
Descripción				
Se pretende comprobar la correcta liberación de herramientas después de que el usuario haya sido expulsado del sistema por exceso de tiempo.				
Acciones				
<ol style="list-style-type: none"> 1. El CE reserva varias herramientas. 2. Ejecuta una acción hasta que se le acaba el tiempo. 3. El sistema expulsa al usuario, cerrando las acciones en ejecución y liberando las herramientas. 4. Por último, se ejecutan las acciones. 				

PRUEBA 8				
ID	PF-H	Configuración	Resultado	Válido
Descripción				
Esta prueba fue diseñada para utilizar los servicios externos de cada una de las herramientas.				
Acciones				
<ol style="list-style-type: none"> 1. Los CE A y B reservan varias herramientas, una de ellas de datos para acceder a su servicio externo. 2. Ambos ejecutan las herramientas utilizando a la vez programas externos para el manejo de los servicios. 3. Todos los servicios y las acciones en ejecución responden correctamente, tanto en el envío de datos como en la recepción. 				

PRUEBA 9					
ID	PF-I	Configuración	Común	Resultado	Válido
Descripción					
Prueba la capacidad de insertar herramientas <i>en caliente</i> , es decir, mientras el sistema está en pleno funcionamiento, atendiendo peticiones de los usuarios.					
Acciones					
<ol style="list-style-type: none"> 1. El CE <i>A</i> reserva algunas herramientas. 2. El CE <i>B</i> se conecta al sistema pero aún no reserva nada. 3. Se inserta en el L <i>1</i> una herramienta y se arma. 4. El CE <i>B</i> puede reservar desde ese momento la herramienta. 					

PRUEBA 10					
ID	PF-J	Configuración	Remoto	Resultado	Válido
Descripción					
La última prueba de funciones corresponde a la respuesta de un L cuando está sobrecargado de acciones en ejecución.					
Acciones					
<ol style="list-style-type: none"> 1. El L <i>1</i> se configura para que sólo acepte 4 procesos en ejecución. 2. Los CE <i>A</i> y <i>B</i> reservan varias herramientas que pertenecen a L <i>1</i> y ejecutan todas las acciones correspondientes. 3. Las ejecuciones se ven bloqueadas con el estado de espera hasta que alguna con estado de ejecución termine. 4. Las peticiones de ejecución se ejecutan con el orden de llegada. 					

5.3. Pruebas de recuperación de errores

Las siguientes pruebas comprobaron que bajo ciertas situaciones de error o incontroladas, el sistema respondía tal y como corresponde a la recuperación de errores implementada.

Muchos de los errores que pueden surgir en una plataforma RLF son externos a la misma, por lo que su control se resume en una parada controlada del componente o componentes afectados.

PRUEBA 11					
ID	PE-A	Configuración	Remoto	Resultado	Válido
Descripción del error					
La herramienta en ejecución sufre un fallo bloqueante.					
Respuesta					
<ol style="list-style-type: none"> 1. El L encargado de la herramienta espera hasta que se agote el tiempo de la acción. Dado ese caso, se considera un error bloqueante. 2. L establece el estado de error para esa acción, para por completo la ejecución de la acción y añade una petición para ejecutar el limpiador. 3. El CE recibe la notificación de que la acción actual ha sido parada por un error surgido. 4. Se muestra las excepciones enviadas por la herramienta y el estado final de la misma. 					

PRUEBA 12					
ID	PE-B	Configuración	Remoto	Resultado	Válido
Descripción del error					
Uno de los laboratorios es desarmado por el administrador o sufre una parada por un error.					
Respuesta					
<ol style="list-style-type: none"> 1. El CE recibe el mensaje de error y comprueba las herramientas reservadas de ese laboratorio. 2. Se muestra el diálogo de aviso por cierre inesperado de L. 					

PRUEBA 13					
ID	PE-C	Configuración	Remoto	Resultado	Válido
Descripción del error					
El P sufre un error o su nodo es desconectado de la red.					
Respuesta					
<ol style="list-style-type: none"> 1. Cuando los CE o CW se intentan conectar al P, o realizar una acción si ya lo estaban, se muestra el diálogo de problema con el servidor central. 					

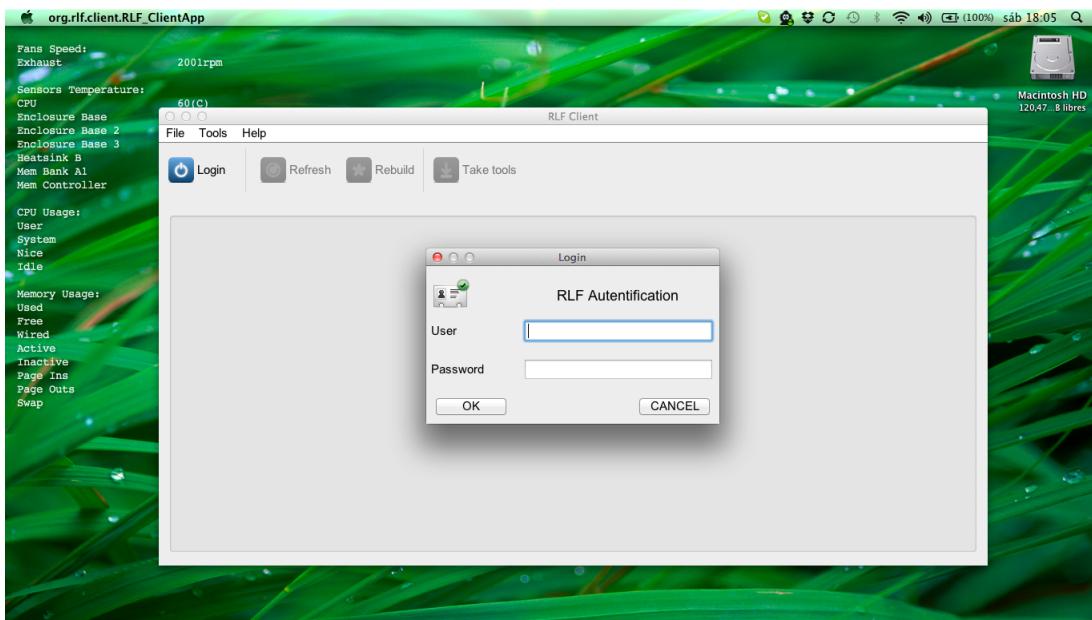


Figura 5.1: En estas pruebas también se han incluido nodos con el sistema Macintosh.

PRUEBA 14				
ID	PE-D	Configuración Remoto	Resultado	Válido
Descripción del error				
El CE sufre un error o una desconexión del sistema.				
Respuesta				
<ol style="list-style-type: none"> 1. El CE no consigue desconectarse del sistema de manera normal. 2. El P bloquea la cuenta de usuario hasta que se ponga en contacto con el administrador para que la desbloquee. 				

PRUEBA 15				
ID	PE-E	Configuración Remoto	Resultado	Válido
Descripción del error				
El CW sufre un error o una desconexión del sistema.				
Respuesta				
<ol style="list-style-type: none"> 1. El CW no consigue desconectarse del monitor de manera normal. 2. P no bloquea la cuenta de usuario, por lo que se puede volver a conectar desde un CW o un CE. 				

5.4. Pruebas de estrés

Por último, estas pruebas están recogidas para verificar el correcto funcionamiento en ambientes con gran carga o baja potencia. El número de componentes y clientes es aumentado.

PRUEBA 16				
ID	PX-A	Configuración	Básico	Resultado
Descripción del conjunto				
Todos los componentes se encuentran en un <i>netbook</i> , con un procesador Atom 450 y 1 GB de memoria RAM. Acceden dos CE a él, uno de ellos ejecuta la herramienta de vídeo por <i>streaming</i> . Acceden otros dos CW mediante dispositivos <i>Android</i> .				
Respuesta				
El sistema responde correctamente ocupando el 95 % del procesador y 150 MBs de RAM. El vídeo se recibe de manera fluida y la interfaz del CE A funciona correctamente.				

PRUEBA 17				
ID	PX-B	Configuración	Remoto	Resultado
Descripción del conjunto				
En un entorno donde las conexiones entre las redes son de muy baja velocidad, se prueba el acceso a las diferentes herramientas. El P y L 1 están conectados a la red mediante <i>tethering</i> con un teléfono móvil con acceso 3G (alrededor de 200 kB/s de bajada y 64 kB/s de subida en Madrid [7]), así como los dos CE y otro CW conectados a redes normales.				
Respuesta				
Los accesos al proveedor se ven ralentizados. Cuando se ejecuta una acción, a pesar de que la interfaz no se bloquea, el flujo de datos es demasiado lento como para poder tener la continuidad necesaria para un correcto funcionamiento.				

PRUEBA 18				
ID	PX-C	Configuración	Educativa	Resultado
Descripción del conjunto				
Se accede desde 15 terminales en los laboratorios informáticos de la universidad hacia el sistema que se encuentra en el taller de Automática. Las redes de conexión son las propias de la universidad.				
Respuesta				
La plataforma RLF funciona sin ningún retardo.				

PRUEBA 19				
ID	PX-D	Configuración Industrial	Resultado	Válido
Descripción del conjunto				
Los componentes se encuentran distribuidos en distintas localidades separadas por varios kilómetros de distancia. Además, uno de los laboratorios y un CE se situaron en Osaka, Japón.				
Respuesta				
La plataforma RLF funciona sin ningún retardo.				

Como se ha comprobado en las numerosas pruebas, la plataforma RLF *Prototype 1* ha superado los objetivos de estabilidad y seguridad. Por supuesto, para obtener una versión completa para el mercado, ha de someterse a pruebas de estrés mucho más severas. Al estar tratando con un sistema crítico donde los errores pueden suponer un gasto en reparaciones bastante importante, es necesario un control de calidad exhaustivo y de gran precisión.

Capítulo 6

Problemas y mejoras

Se recoge en este capítulo los problemas derivados de la implementación de la plataforma RLF así como los trabajos futuros que se deberán tener en cuenta para posteriores versiones.

6.1. Problemas encontrados en el desarrollo

Se listan a continuación los problemas referentes a la implementación y a las tecnologías usadas. Sólo han sido incluidos aquellos más importantes que han determinado el desarrollo de la plataforma RLF.

6.1.1. La sincronización

Siendo un problema que arrastran muchas de las plataformas que existen actualmente en el mercado, fue arrastrado desde el principio de la implementación. Se ha invertido mucho tiempo en conseguir un sistema que es asíncrono a partir de muchos componentes síncronos.

Los sistemas síncronos [11] son aquellos que se bloquean a la espera de una comunicación concreta por parte del otro interlocutor. En cambio, los sistemas asíncronos, pueden realizar otras tareas mientras el otro interlocutor genera la información. Los distintos componentes síncronos que se aprecian, como la entrada y salida estándar de todas las herramientas, han sido modificados para poder permitir no bloquear al resto de componentes.

Esto se ha conseguido con la sustitución de los mecanismos de lectura y escritura (de *sockets*, ficheros y teclado) tradicionales, que se presentan en la máquina virtual de Java como *streams* pertenecientes a las librerías “java.io”. A partir de la versión 1.4.2 de Java, Sun Microsystems añadió las librerías “java.nio” [16] que aportaban una nueva forma de ver la entrada y salida para la máquina virtual. El conjunto fue llamado Java New I/O, en referencia al sistema antiguo, Java I/O. La estructura de estas librerías es muy parecida a los sistemas Unix, pudiendo utilizar funciones no bloqueantes, y algunas herramientas muy útiles para no malgastar tiempo de cálculo. Además, incluyen *buffers* gestionados por el propio sistema operativo anfitrión que dotan de una mayor velocidad a la, de por si lenta, máquina de Java.

En la figura 6.1 se puede comprobar como con las distintas versiones de estos dos conjuntos de librerías se mejora la velocidad de lectura y escritura. Se realizó el estudio con varios ficheros de distinto tamaño (coordenada X) y la velocidad de lectura y posterior escritura en KB/s (coordenada Y). Las series corresponden a dos experimentos, A y B, con la versión 2.2 de “java.io” y la versión 2.4 de “java.nio” [6].

6.1.2. La portabilidad de Java

Aunque es una plataforma que puede ejecutarse en múltiples sistemas, Java necesita “una ayuda” para poder realizar bien su cometido en Linux y Windows por igual. El primer problema que se encuentra es en el acceso a archivos del sistema de ficheros. El árbol de rutas es distinto para *ext4*¹. que para *ntfs*². Esto conlleva a que el código de implementación debe ser lo suficientemente genérico como para que no surjan problemas a la hora de portar los distintos componentes.

Aunque en el desarrollo de RLF no se ha optado en ningún momento por la división de código, es decir, incluir un código para cada sistema operativo, se han realizado múltiples cambios en el diseño para adaptarlo a un único código. Se puede ver a continuación, un ejemplo de división de código atendiendo al sistema operativo [23]:

¹Sistema de ficheros que las distribuciones Linux usan en la actualidad

²Sistema de ficheros moderno para las últimas versiones de Windows, como XP, Vista y Windows 7.

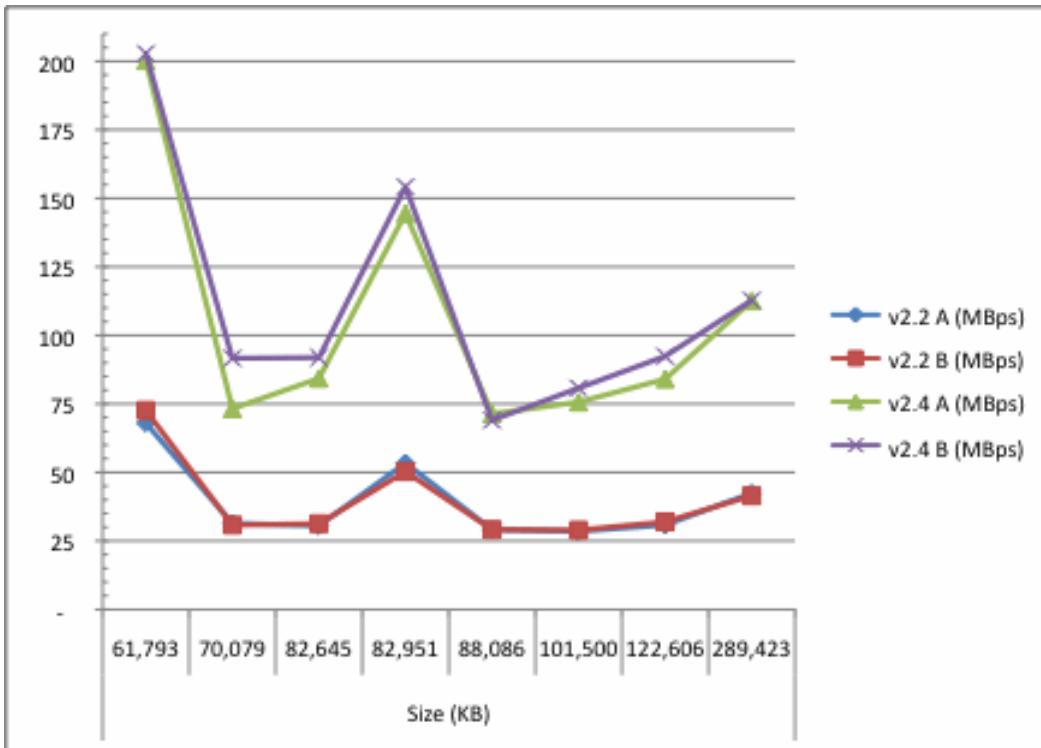


Figura 6.1: Comparación de las librerías “java.io” y su nueva versión “java.nio”.

```

String osName = System.getProperty("os.name");
if ((osName.equals("Windows NT")
    || osName.equals("Windows 7")
    || osName.equals("Windows XP")) {
    // Do something...
} else if ((osName.equals("Linux")
    || osName.equals("Mac")) {
    // Do another thing...
} else {
    // Cry.
}

```

6.1.3. Streaming de vídeo y Java

Fueron muchos días los que se intentó, sin éxito, utilizar la cámara web de la herramienta RLF_Video a través de Java. Para ello se utilizó el *framework* aportado por Sun Microsystems para el control de sistemas multimedia llamado JMF (*Java Media Framework*).

A pesar de que el nuevo dueño de las tecnologías Java, Oracle, indique en su página que sigue en activo y se está desarrollando actualmente aplicaciones con él, la última versión data de 2001, con la versión de Java anterior a la 1.4.2 (la primera considerada “moderna”). Además, la documentación para el desarrollo ya no es accesible desde las páginas oficiales.

Se optó por utilizar el servidor de VLC, incluido en las distribuciones Linux y adecuarlo mediante BASH para poder ejecutarlo desde la plataforma RLF.

6.1.4. La tarjeta PCI-1711-BE

Uno de los elementos más importante que se ha aportado a las herramientas presentadas, es la interactividad con un *hardware* donde su acceso era en el mismo lugar donde se encontraba. Se utilizó una tarjeta que provee de entradas y salidas electrónicas, llamada Advantech PCI-1711-BE que disponía de un conjunto de librerías para interactuar por medio del *software*, que podían ser utilizadas en sistemas Windows y Linux.

Todo intento por utilizar la documentación (escrita en 1996) y las herramientas aportadas fue un fracaso hasta que se consiguieron unos ejemplos que se podían utilizar en el IDE Visual Studio 2005. Siendo aún incompatibles con los sistemas actuales, mediante sustitución de código antiguo y de librerías que ya no existen en Windows, se pudo adecuar a la plataforma .NET, y con ello, a la plataforma RLF.

Se puede ver a continuación el código original de algunos ejemplos de dicha tarjeta:

```
/*
(...)

 * Revision      : 1.00
 * Date          : 7/1/2003
 (...)

 */

(...)

// Estos tipos de datos no son compatibles con .NET
DWORD dwErrCde;
ULONG l1DevNum;
long l1DriverHandle;
USHORT usChan;
(...)

// Funciones no soportadas por Windows
getch();
(...)
```

6.2. Próximos pasos

No cabe duda que la plataforma RLF aquí presentada necesita más desarrollo para poder afianzarla como un producto comercial. Estando aún en la versión *Prototype*, requiere de determinadas tareas para poder ser implantada en entornos de trabajo. Se recopilan a continuación los próximos trabajos propuestos:

- Se pueden aplicar pruebas de estrés a la plataforma mayores de las que se incluyen en la sección 5.4, contando con varias decenas de laboratorios, y varios cientos de usuarios.
- Acoplamiento del sistema de cifrado de comunicaciones con túneles SSH que aportarían blindaje a la plataforma, de la misma forma que se muestra en la figura 6.2.
- Diseño de tareas automatizadas para la base de datos del proveedor, como comprobación de nodos de la red o de usuarios con estados erróneos.
- Creación de “paquetes” de aplicaciones para alumnos, que permitan obtener todo el *software* necesario en un solo instalador (como por ejemplo, reproductores de video, clientes FTP, etc) pudiendo además reservar las herramientas en conjunto con anterioridad.

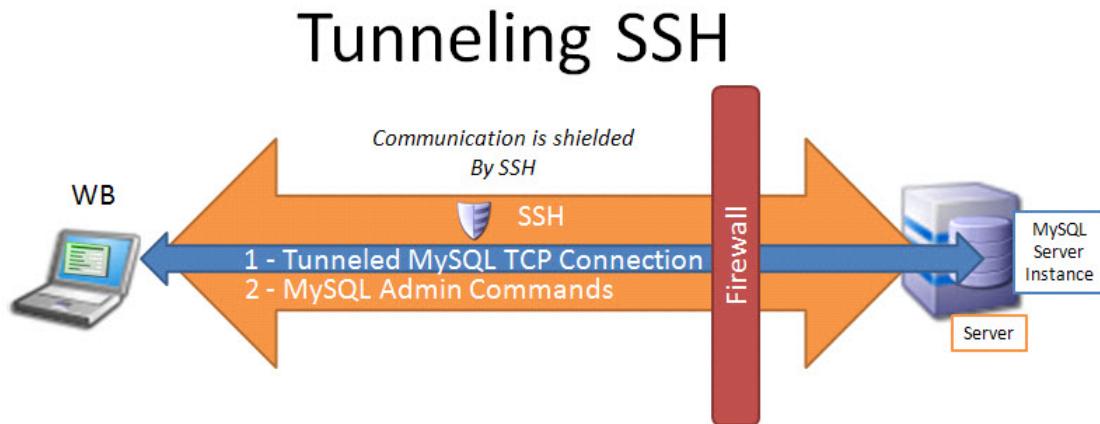


Figura 6.2: Ejemplo de la arquitectura de un túnel SSH.

6.2.1. Próximas versiones de RLF

Dada la base de RLF Prototype, cabe proponer otras funciones que, con unos determinados cambios, pueden llevarse a cabo, aprovechando las características principales del mismo. Las líneas de desarrollo pueden variarse e incluso crear nuevas, todo depende de las necesidades del centro que utilice RLF. Se muestra en la figura 6.3 continuación una idea de las próximas versiones y sus cambios.

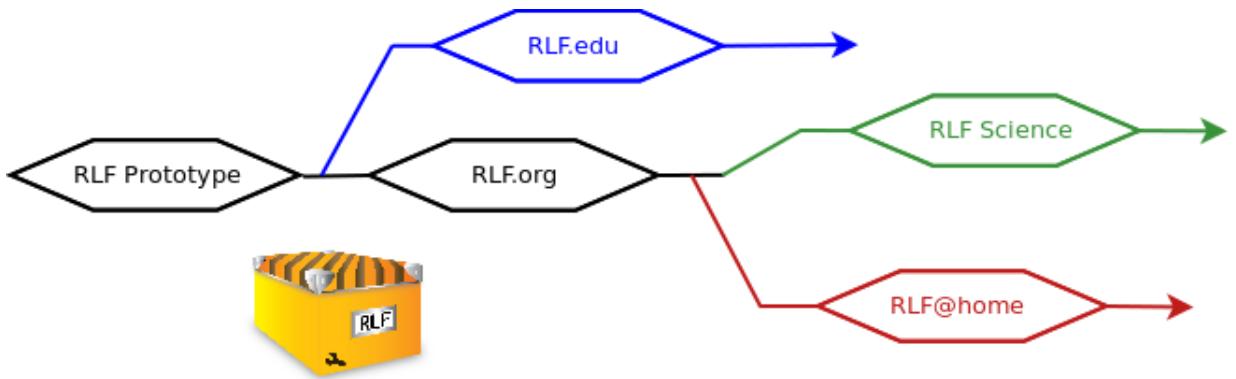


Figura 6.3: Próximas posibles líneas RLF.

RLF.org Corresponde a la evolución natural del proyecto, con mejoras de estabilidad y seguridad, y una fase completa de *tests* de estrés. Sería la candidata para salir al mercado y adecuarse a varias líneas de trabajo.

RLF.edu Línea especializada en entornos educativos, donde se pueden establecer configuraciones estándar para la realización de prácticas, como por ejemplo, que todos los laboratorios contengan una herramienta de vídeo, y que obligue a la interfaz a contener un reproductor de *streaming* embebido.

RLF@home Cambiando el sistema de reserva de herramientas, y liberando de carga a los laboratorios, se puede convertir en una plataforma para edificios o casas inteligentes. Así se podrían gestionar distintos dispositivos desde el cliente.

RLF Science Al igual que algunos centros de investigación, se pueden alquilar por determinado tiempo elementos *hardware* a usuarios de Internet. Como por ejemplo, en el observatorio astronómico de Chile, se puede utilizar el telescopio si se paga una cuota. También puede servir como plataforma para sistemas *grid*, proporcionando capacidad de cálculo y de almacenamiento sustituyendo las herramientas *hardware* por terminales de acceso a distintos nodos de una red de computación.

Capítulo 7

Conclusiones

“Sólo es posible avanzar cuando se mira lejos. Solo cabe progresar cuando se piensa en grande.” *José Ortega y Gasset*



El avance de las nuevas tecnologías significa el avance de la sociedad, y de todos sus aspectos, desde la forma de relacionarse hasta la forma de trabajar. Es indudable que Internet revoluciona los sistemas actuales, sin que se pueda ignorar. Las empresas se han visto obligadas a cambiar su modelo de negocio para adecuarse a estos tiempos. Y por supuesto los centros de enseñanza. Se pueden encontrar universidades donde no es necesario ir a clase, ya que mediante plataformas colaborativas disponibles en Internet, así como la llegada de la emisión de sonido y vídeo a través de red el alumno puede aprender casi de la misma manera que si se desplazada hasta el centro educativo. Pero poco a poco, ese “casi” irá desapareciendo, aumentando a su vez la posibilidad de acceso a enseñanzas superiores.

La mayor parte de las retribuciones positivas de este proyecto provienen de haber desarrollado una plataforma desde la capa más baja hasta la que está en contacto con el usuario. El uso de las múltiples tecnologías y su unión como un único sistema ha demostrado que es necesario evolucionar la tecnología desde diferentes puntos de vista, y sin ser guiados por un único objetivo.

Este proyecto no sólo ha sido la culminación de seis años de estudio, si no también un aspecto importante en la formación para ser ingeniero informático, y más aún si la especialidad son los sistemas distribuidos, que ahora tanta importancia han adquirido desde la aparición de la “nube” y los sistemas portátiles.

Así pues, habiendo cumplido con los objetivos marcados al inicio del proyecto, la satisfacción de haber realizado este proyecto es plena, ya que no sólo se ha conseguido crear una plataforma distribuida totalmente funcional, sino que se ha realizado con las mismas tecnologías que utilizan las grandes empresas en el mundo industrial y educativo. Muchas de las decisiones tomadas durante el desarrollo de este proyecto han venido influenciadas por conceptos que actualmente los grandes proyectos también se plantean, como es el tema de la seguridad. Se han tenido que sortear dificultades muy presentes a la hora de desarrollar *software*, que cada vez se hacen mayores por la cantidad de dispositivos y sistemas, creados por diferentes empresas y organizaciones, que se encuentran en los hogares y centros.

El que en un proyecto de fin de carrera sea utilizado en un escenario práctico, real y donde se espera continuar es algo muy poco común, y desde luego inmensamente satisfactorio. Se espera que este trabajo sirva para futuras ideas que ayuden a la comunidad educativa a formar a mejores profesionales e investigar nuevas tecnologías. El esfuerzo se ha visto recompensado en forma de nuevas ideas.

Madrid, 1 de Octubre de 2011.

Fdo: Carlos A. Rodríguez Mecha

Apéndice A

Presupuesto

La versatilidad aportada a este proyecto reduce en gran medida los gastos necesarios para el despliegue del mismo. Se adjunta a continuación un presupuesto aproximado dependiendo de las necesidades actuales del lugar de trabajo o enseñanza. Todos los precios aquí recogidos datan de Septiembre de 2011.

Costes de desarrollo

No son necesarios para el despliegue pero cuantifican la cantidad de trabajo empleada para la creación de la plataforma, así con su diseño y documentación.

Coste de personal

Este cálculo se realiza atendiendo a la planificación que se incluyó en la tabla 1.1 del capítulo introductorio. El precio por hora está estimado según los sueldos en Septiembre de 2011 en Madrid:

Concepto	Coste en €
Ing. Informático por hora	30,00
Corrector de documentación por hora	17,00
TOTAL	44.210,00

Tabla A.1: Coste de personal

Costes de material para el desarrollo

Algunos de estos precios no son por la compra del producto (señalados con #), si no por el gasto aproximado en el uso del material.

Concepto	Coste en €
Servidor primario #	180,00
Servidor secundario #	120,00
Ordenador portátil #	120,00
Dispositivo móvil	169,00
Sistema operativo Windows 7 Professional	0,00
Sistema operativo Windows XP Professional	0,00
Sistema operativo Unix	0,00
Base de datos MySQL	0,00
Base de datos SQLite	0,00
Servidor de aplicaciones Tomcat o JBoss	0,00
Cámara web	13,50
Micrófono	6,99
Advantech PCI-1711-BE #	60,00
Altavoces doble canal	29,50
TOTAL	698,99

Tabla A.2: Coste material para el desarrollo

Deduciéndose como coste total del desarrollo:

Concepto	Coste en €
Coste personal	44.210,00
Coste material	698,99
TOTAL	44.908,99

Tabla A.3: Coste total de desarrollo

Costes generales de mantenimiento

Estos costes están asumidos para el despliegue de la plataforma, contando que su distribución se encuentre en la misma red local, aunque el número de nodos es indiferente.

Concepto	Coste anual en €
Línea de alta velocidad empresarial, <i>Movistar</i>	840,00
Suministro eléctrico, <i>Endesa</i>	1.200,00
Seguridad física y de datos, <i>Movistar</i>	600,00
Coste estimado en reparaciones	400,00
TOTAL	3.040,00

Tabla A.4: Costes aproximados de servicios requeridos.

Debido a que los componentes *hardware* tienen más uso que aquellos con un limitado tiempo de acceso diario que se pueden encontrar en las universidades o puestos de trabajo, se asume un incremento en el coste de reparación anual.

Coste material por componentes

A pesar de que todos los componentes RLF pueden estar contenidos en la misma máquina, se especifican los costes de cada uno por separado, después se incluyen configuraciones estándar con su inversión correspondiente.

Proveedor y monitor

Servidor central de toda la plataforma, es la máquina con más prestaciones de todo el conjunto. Sus sistema es Unix y no requiere de ningún programa de pago debido a su configuración estándar. Puede desdoblarse si se requiere para obtener una mayor velocidad y potencia, quedando un servidor con mayor capacidad para el proveedor y uno con menos para el monitor.

Concepto	Coste en €
PowerEdge T410 Tower Server, <i>Dell</i>	1.217,00
{Latitude 13 (Terminal portatil de acceso), <i>Dell</i> }	{619,00}
Router (Incluido en el contrato de la línea)	0,00
Sistema de alimentación ininterrumpida (SAI)	79,90
Sistema operativo Unix	0,00
Base de datos MySQL	0,00
Servidor de aplicaciones Tomcat o JBoss	0,00
TOTAL	1.296,90
TOTAL {con terminal de acceso}	1.915,90

Tabla A.5: Costes aproximados de un proveedor con monitor estándar.

Si se desea cobrar por los servicios de esta plataforma, se deberá pagar la licencia de MySQL y J2EE a Oracle. La terminal de acceso es un ordenador portatil de prestaciones medias con sistema operativo Unix. Es opcional y sólo se requiere uno por sistema RLF.

Laboratorio

Por cada laboratorio en la plataforma se tiene que aplicar este coste. Dependiendo del sistema se deberá pagar la licencia de Windows.

Concepto	Coste en €
PowerEdge T110, <i>Dell</i>	378,00
Sistema operativo Unix	0,00
{Sistema operativo Windows 7 Professional}	{149,99}
Base de datos SQLite	0,00
TOTAL	378,00
TOTAL {con Windows}	527,99

Tabla A.6: Costes aproximados por cada laboratorio.

Herramientas

Concepto	Coste en €
eLight HD 720p Webcam, <i>Trust</i>	79,00
Advantech PCI-1711-BE	499,00
Altavoces doble canal	29,50
TOTAL	607,50

Tabla A.7: Costes aproximados de las herramientas entregadas.

NOTA: Aquí sólo se recogen los gastos de las herramientas entregadas junto con este proyecto.

Ejemplo de un sistema estándar

Se compone de un proveedor central, un laboratorio de Unix y otro de Windows. Todos en diferentes máquinas y con todas las herramientas entregadas en uso. No es necesario una terminal de acceso, ya que el proveedor cuenta con ella.

Concepto	Coste en €
Gastos generales (al año)	3.040,00
Proveedor	1.296,90
Laboratorio Windows	527,99
Laboratorio Linux	378,00
Herramientas	607,50
TOTAL	5850,39

Tabla A.8: Presupuesto aproximado de un sistema estándar.

Apéndice B

Guía de despliegue

RLF está compuesto de varios módulos que trabajan conjuntamente como una única entidad, pero para eso, se requieren un conjunto de pasos que los *orquestan* al inicio.

Plataforma Java

Debido a que es un requisito en **cualquier** máquina de la plataforma RLF, es necesario instalarla antes que cualquier otro componente que se describe a continuación. Se pueden obtener de los siguientes enlaces:

Windows, Java 6 JRE

- Descarga: <http://java.com/es/download/>
- Manual: http://java.com/es/download/help/index_installing.xml

Linux, Java 6 JDK

- Descarga: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- Manual: <http://www.guias-ubuntu.org/index.php?title=Java>

Proveedor y monitor

Estos dos módulos están originalmente creados para entornos Linux, aunque debido a las tecnologías usadas, pueden ser instalados en Windows. Sólo se explicará cómo ponerlos en funcionamiento en una distribución Linux.

Se componen de una base de datos principal en MySQL y de dos servicios web contenidos en el mismo paquete para su acceso externo. A continuación se muestran los pasos necesarios para incluirlos en el sistema.

Base de datos

Se requerirá instalar los paquetes que proveen la plataforma MySQL completa. Para ello se echará mano del gestor de aplicaciones incluidos en las distintas distribuciones Linux. Los paquetes a instalar son los siguientes:

MySQL 5.1

- **mysql-common** Paquete general de la plataforma MySQL.
- **mysql-server** Contiene todo lo necesario para contener un servidor de base de datos.
- **mysql-admin** Aplicación para la cómoda creación de usuarios y realización de *backups* entre otros.
- **mysql-client** Herramientas necesarias para el acceso a bases de datos MySQL.
- **mysql-query-browser** La solución más cómoda que aporta MySQL para el manejo de los datos y creación de tablas en la base de datos.

NOTA: Es recomendable que la base de datos esté en la misma máquina que el servidor web ya que están configurados por defecto para acceder a una base de datos local, aunque se incluye en la sección de este mismo manual cómo configurar los servicios web para poder acceder a la base de datos de forma desacoplada.

Una vez instalados estos paquetes, será necesario configurar el servidor para atender las peticiones, en ello se asignará su IP de escucha y un puerto que será necesario para que cualquier parte de RLF acceda al proveedor. Véase para realizar esta configuración inicial:

<http://dev.mysql.com/doc/refman/5.0/es/unix-post-installation.html>

Una vez el servidor esté completo con la base de datos activa, tocará el turno de aportarle información. Mediante *MySQL Query Browser* se creará un esquema nuevo llamado “rlf”. Después se desplegarán los *scripts* que crean todas las tablas necesarias, se encuentran en la carpeta *rlf/src/sql* y son el conjunto de archivos indicados en el fichero *makedb.sql*.

Creación de usuarios.

Cuando se pueda disponer de la base de datos, se recomienda crear los usuarios de acceso a la misma. Mínimo se requiere un administrador general, uno para el servicio web y uno por cada laboratorio que se vaya a disponer. Se podrá realizar mediante el programa instalado anteriormente *MySQL Administrator*. Los permisos necesarios para el usuario del proveedor y de los laboratorios son los siguientes:

En el esquema “mysql”:

- *SELECT*

En el esquema “rlf”:

- *SELECT*
- *INSERT*
- *UPDATE*
- *DELETE*
- *REFERENCES*
- *LOCK_TABLES*
- *EXECUTE*

Mientras que el usuario administrador tendrá todos los permisos en todos los esquemas disponibles.

Los servicios web

Una vez que la base de datos está creada, es necesario instalar el servidor que permite ejecutar aplicaciones web. Se ha seleccionado la plataforma Tomcat que viene disponible con el IDE NetBeans de desarrollo en J2EE.

Antes de poder desplegar la aplicación, se requiere modificar el código de acceso a la base de datos del servicio web. Para ello, se utilizará el IDE NetBeans que se puede descargar de <http://netbeans.org/>, y se importará el proyecto contenido en *rlf/projects/netbeans/RLF_Provider*. A continuación se modifica la clase “Database.java”:

```
/** Localización de la base de datos del proveedor. */
public final static String DATABASE = "jdbc:mysql://<IP>:<PUERTO>/rlf";
/** Usuario de acceso a la base de datos. */
private static String USER = <USUARIO PROVEEDOR>;
/** Contraseña del usuario. */
private static String PASS = <CONTRASEÑA>;
```

Una vez modificado esto, se compilará el proyecto obteniendo como resultado un archivo “Provider.war”. Después se instalará el programa Tomcat mediante los siguientes paquetes:

Tomcat 6.0

- **tomcat6** Paquete general de la plataforma Tomcat.
- **tomcat6-admin** Aplicación para navegador que permite la configuración del sistema de manera muy sencilla.

Es necesario asignarle una IP y un puerto de escucha. Serán estos parámetros los que luego se configurarán en los clientes. Se puede obtener información de como configurar Tomcat en el siguiente enlace oficial:

<http://tomcat.apache.org/tomcat-6.0-doc/setup.html>

Por último se copiará el archivo generado anteriormente en la carpeta *webapps* donde se haya instalado Tomcat. A partir de ese momento se puede acceder al monitor y al proveedor por medio de web o de los clientes, aunque no habrá aún usuarios RLF introducidos. Las rutas típicas de acceso a estos dos servicios pueden ser:

<http://<IP>:<PUERTO>/RLF/Provider>
<http://<IP>:<PUERTO>/RLF/Monitor>

Laboratorios

Una vez instalada la máquina virtual de Java, no es necesario instalar ningún componente más. Conviene copiar todos los archivos contenidos en *rlf/bin/lab* (*rlf\bin\lab* en Windows) a otra carpeta y configurar el laboratorio modificando el fichero *res/lab.conf* (o *res\lab.conf* en Windows) de la siguiente forma:

```

lab_name=<NOMBRE ÚNICO DEL LABORATORIO>
user=<USUARIO DE LA BASE DE DATOS>
pass=<CONTRASEÑA>
provider_host=<IP DE LA BASE DE DATOS DEL PROVEEDOR>
provider_port=<PUERTO DE LA BASE DE DATOS DEL PROVEEDOR>
labmanager_request_port=<PUERTO GENERAL DEL LABORATORIO. DFL=6400>
client_request_port=<PUERTO DE COMUNICACIONES 1. DFL=6401>
client_notification_port=<PUERTO DE COMUNICACIONES 2. DFL=6402>
provider_request_port=<PUERTO DE COMUNICACIONES 3. DFL=6403>
max_process=<NÚMERO MÁXIMO DE PROCESOS EN EJECUCIÓN. DFL=4>

```

Es importante apuntar el puerto general del laboratorio ya que es necesario para su configuración mediante LabConsole (véase Manual de mantenimiento).

Insertar el laboratorio en el proveedor

Para que los laboratorios puedan utilizar correctamente el proveedor, deben ser introducidos en la base de datos de este. Para ello es necesario obtener la IP de la máquina donde se ejecuta. Por cada laboratorio, se insertará en la base de datos central la siguiente sentencia SQL:

```
INSERT INTO lab (name, host, description) VALUES ('nombre', 'ip', 'descripción');
```

Ejecutar el laboratorio

Para iniciar el laboratorio sólo es necesario acceder a la carpeta contenedora y ejecutar el fichero “lab.jar” aunque es recomendable utilizar la terminal que disponga el sistema operativo para obtener posibles errores (aparte de los que aparezcan en el *log*). Se podrá realizar esto mediante el siguiente comando, válido tanto en la terminal de Linux como en el CMD de Windows:

```
java -jar <ruta>/lab.jar
```

Herramientas

El despliegue de las herramientas sólo es referido a las que han sido entregadas con este proyecto. Para crear nuevas consultar el Manual de desarrollo y el Manual de mantenimiento.

Configuración inicial

Para que las herramientas funcionen se ha de modificar las rutas contenidas en sus ficheros de configuración XML. También puede ser necesario cambiar algún parámetro de estas que dependa del sistema (como por ejemplo, en el caso de RLF_Video la IP de acceso). Después, se podrán registrar (ver Manual de administrador) y obtener la clave única, que se incluirá en el código y posteriormente se compilarán. Se ha añadido los proyectos para los distintos IDEs de las herramientas para poder ser compilados (y arregladas las dependencias) sin dificultad.

Componentes necesarios

Algunas herramientas necesitan de componentes instalados externos que se citan a continuación.

- **RLF_DummyTool:** Ninguno.

- **RLF_Music:**

- Es necesario el programa para linux MPG123 que se puede obtener del paquete con el mismo nombre.
- El paquete de desarrollo *libsdlite-dev*.

- **RLF_Video:**

- Utiliza el programa VLC que se obtiene de los repositorios oficiales de la distribución bajo el paquete con el mismo nombre.
- El driver necesario para utilizar la cámara web en Linux.

- **RLF_FreeMem:**

- Requiere la librería System.Data.SQLite instalada en el sistema, que se puede obtener de <http://sqlite.phxsoftware.com/>.

- **RLF_Board:**

- También requiere la librería System.Data.SQLite.
- Necesita los drivers para la tarjeta asociada (<http://www.advantech.com/>). Es recomendable que se recompile resolviendo las dependencias de estas librerías ya que no pueden incluirse en el propio ejecutable.

Usuarios de RLF

Para que se pueda usar y administrar la plataforma, es necesario dar de alta a los usuarios de la misma.

Administradores

Serán los encargados de mantener las herramientas, así como eliminarlas y registrarlas. Además utilizan el programa LabConsole para manejar los distintos laboratorios. Para dar de alta a un administrador en el sistema es requerido insertar esta sentencia SQL en la base de datos de proveedor:

```
INSERT INTO user (user, hash_pass, email)
    VALUES ('nombre', SHA1('contraseña'), 'email');
INSERT INTO admin (name, tlf)
    VALUES ('nombre', 'teléfono');
```

A partir de aquí ya podrá realizar dichas tareas.

Clients

Podrán usar el monitor web y el cliente de escritorio. Son los usuarios finales de RLF. Tienen asociado un tiempo máximo de reserva de herramientas (a discreción de los administradores, generalmente una hora) y un rol que les permite acceder a herramientas de mayor nivel o menos (el rol 0 es el más básico y según aumenta es más restrictivo). Se deberá añadir estas sentencias SQL al proveedor:

```
INSERT INTO user (user, hash_pass, email)
    VALUES ('nombre', SHA1('contraseña'), 'email');
INSERT INTO client (name, timeout, role)
    VALUES ('nombre', tiempo, rol);
```

NOTA: Un administrador no incluye el rol de cliente, por lo que si se requiere un usuario con ambos perfiles, habrá que darlo de alta en cada perfil por separado de esta forma:

```
INSERT INTO user (user, hash_pass, email)
    VALUES ('nombre', SHA1('contraseña'), 'email');
INSERT INTO admin (name, tlf)
    VALUES ('nombre', 'teléfono');
INSERT INTO client (name, timeout, role)
    VALUES ('nombre', tiempo, rol);
```


Apéndice C

Guía de mantenimiento

Este manual está dedicado a los administradores para realizar un correcto mantenimiento de la plataforma RLF mientras esta esté en ejecución. Además se incluyen algunos errores externos para su tratamiento.

Proveedor y monitor

La única parte del proveedor y monitor que necesita mantenimiento es la base de datos global. Aunque la mayoría del tratamiento de datos es automático, en la versión *Prototipo 1* entregada hay que realizar algunas tareas a mano.

Usuarios

Debido a motivos de seguridad, si un usuario no se desconecta correctamente del sistema (en el cliente, en el monitor no ocurre esto) su cuenta se queda bloqueada para su revisión. Para permitir otra vez el uso de esta cuenta, hay que eliminar el *token* de acceso del usuario, es decir, establecer su valor a *NULL*. Se puede utilizar un programa de gestión de la base de datos como MySQL Browser (descargable en <http://dev.mysql.com/doc/query-browser/es/index.html>).

Backups

El proceso de duplicar la base de datos es necesario cada cierto tiempo (a discreción de los administradores) para salvaguardar los datos de los usuarios. Realmente, este proceso es requerido sólo para estos datos, ya que la forma de desplegar el sistema hace que no sea muy complejo volver a añadir los datos. Para obtener información de cómo hacer un *backup* en una base de datos MySQL consultar <http://dev.mysql.com/doc/refman/5.1/en/backup-methods.html>.

Logs

Es recomendable que se revisen los *logs* del proveedor y monitor por si ha habido algún problema durante las ejecuciones. Se encuentran en la carpeta del mismo nombre y pueden ser consultados mediante un visor de texto genérico. También puede ser necesario borrarlos cada cierto tiempo para no saturar el disco.

Laboratorios

El mantenimiento de los laboratorios se realiza mediante el programa *LabConsole* incluido en este proyecto. Los laboratorios tienen tres estados principales:

- **Iniciado:** Es el estado por defecto de un laboratorio. En él se pueden realizar la mayoría de las tareas de mantenimiento, como registrar una nueva herramienta y eliminarla, y también armar el propio laboratorio. Cuando se desarma el laboratorio o se arranca por primera vez se alcanza este estado.
- **Armado:** En este estado el laboratorio está preparado para escuchar las peticiones de los usuarios y comunicarse con el proveedor. No se puede realizar tareas de mantenimiento, sólo obtener el estado de cada una de las herramientas en tiempo real.
- **Parado:** La ejecución del laboratorio se cierra. Para una parada controlada es necesario desarmarlo antes. También se puede llegar a este estado con una parada de emergencia.

LabConsole

Esta herramienta permite acceder a cualquier laboratorio de la red. Para utilizarlo se requiere un nombre de administrador y su contraseña. Las acciones que se pueden realizar son las siguientes:

```
java -jar LabConsole.jar [-h <IP> -p <Puerto>] -user <Usuario>
                           -pass <Contraseña> <Comando> <Parámetros>
```

- **Armar:** Arma el laboratorio al que se accede. Antes de escuchar peticiones, se ejecutarán todos los limpiadores de las herramientas.

<Comando>: arm
 <Parámetros>: ninguno

- **Desarmar:** Desarma el laboratorio. Es recomendable que se compruebe el estado antes de las herramientas, ya que si hay usuarios usándolas se desconectarán.

<Comando>: disarm
 <Parámetros>: ninguno

- **Parar:** Para por completo el laboratorio. Si está armado lanzará un error.

<Comando>: stop
 <Parámetros>: ninguno

- **Emergencia:** Para por completo el laboratorio y dejará las herramientas en el estado actual. Todos los usuarios conectados serán expulsados y no se aceptarán nuevas conexiones. La clave de emergencia por defecto es “Emergency!”.

<Comando>: emergency
 <Parámetros>: <Clave de emergencia>

- **Estado:** Obtiene el estado de las herramientas (en ejecución o parada) y del propio laboratorio. Esta acción se puede realizar cuando el laboratorio está iniciado o armado.

<Comando>: status
 <Parámetros>: ninguno

- **Registrar:** Registra una herramienta. Es necesario indicar dónde se encuentra el fichero .xml de configuración (ruta completa local). Cuando se registre se obtendrá el identificador único de la herramienta y su clave. Esta acción sólo puede llevarse a cabo cuando el laboratorio no está armado. Véase el Manual de Despliegue.

<Comando>: registry
 <Parámetros>: <Ruta local del fichero XML>

- **Eliminar:** Elimina una herramienta. Es necesario indicar el identificador de la herramienta y su clave. También será borrada de la base de datos. Esta acción sólo puede llevarse a cabo cuando el laboratorio no está armado.

<Comando>: drop
 <Parámetros>: <ID de la herramienta> <Clave de la herramienta>

Limiadores

Cada herramienta posee una acción especial para *resetear* el *hardware* asociado. Esta acción se ejecuta automáticamente cuando se arma el laboratorio y cuando ha ocurrido fallo de ejecución. Es recomendable que se realice cada día un desarmado y armado de todos los laboratorios para que se ejecuten los limiadores al menos una vez cada 24 horas.

Logs

Al igual que el proveedor y el monitor, los laboratorios tienen su propio registro de sucesos, que es recomendable observar. También se pueden eliminar cada cierto tiempo.

Herramientas

No necesitan un mantenimiento concreto, aunque por motivos de seguridad, puede ser necesario registrar y eliminar las herramientas cada mes o periodo similar y así puedan cambiar de clave.

Hardware

A pesar de que el mantenimiento del *hardware* sea automático, el administrador debe realizar revisiones periódicas al mismo, independientemente del uso que se le ha dado.

Apéndice D

Guía de desarrollo de herramientas

Este proyecto basa su versatilidad en la gran cantidad de soluciones que se pueden aportar en forma de conjuntos de aplicaciones (llamadas herramientas). Aquí se explican los métodos necesarios para realizar dichas herramientas.

Estructura de una herramienta

Las herramientas son conjuntos de aplicaciones sin interfaz gráfica que se comunican con la plataforma RLF mediante una librería llamada *libtool*. Se caracterizan por:

- Cada herramienta puede tener una o varias acciones a realizar (es decir, una o varias aplicaciones) que sólo se podrá ejecutar una instancia de ellas, siendo imposible ejecutar dos o más a la vez.
- La herramienta puede tener una salida estándar (textual) y una entrada también textual que serán las que el usuario visualice.
- Generalmente estas están asociadas a un *hardware* específico, como puede ser una cámara de vídeo.
- Pueden ofrecer uno o varios servicios externos, los cuales requieren que el usuario se conecte a ellos de forma independiente a la plataforma, llamados *sockets*. Por ejemplo, pueden ser servicios externos un servidor FTP o HTML, *streaming* de archivos multimedia, conexión a una base de datos, etc.
- Las herramientas y sus componentes se definen en un fichero XML con un formato determinado, y se usará para darlas de alta en la plataforma RLF.
- Aquellas que poseen el tipo “Herramienta de datos” no disponen de entrada estándar y sólo contienen una acción (o aplicación). Además esa acción no tiene parámetros de salida ni de entrada.
- Cada acción tiene un número determinado de parámetros de entrada, salida o ambos que serán definidos antes de ejecutarla, así como unas constantes que no variarán su valor. Cuando se termine la ejecución, se almacena un resultado, y las excepciones que puedan ocurrir.

En cuanto a los ficheros que debe contener una herramienta, posee una carpeta principal (llamada *root*) donde será la ruta base para acceder a todas las aplicaciones.

Desarrollar una nueva herramienta

Lo primero es realizar el fichero de configuración que defina el comportamiento de la herramienta. Se deberá decidir si la herramienta pasa a ser “de Datos”. Dependiendo del tipo, se tendrá que llenar un fichero XML u otro. Se muestran a continuación dos ejemplos de ambos tipos:

Configuración

```
<!-- EJEMPLO DE UNA HERRAMIENTA -->
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tool SYSTEM
  "/home/rodriguezmecha/Universidad/Proyecto/r1f/tools/tool.dtd"
>
<tool
  path="/home/rodriguezmecha/Universidad/Proyecto/r1f/tools/linux/RLF_DummyTool"
```

```

    data="false"
  >
    <in-stream/>
    <out-stream/>
    <attributes>
      <name>RLF DummyTool</name>
      <version>0.1</version>
      <description>
        Ejecuciones básicas para realizar pruebas en Linux.
        Comprende dos acciones que ejecutan operaciones de entrada
        y salida por teclado y acceso a ficheros locales.
      </description>
      <admin>carlos</admin>
      <role>0</role>
    </attributes>
    <constants>
    </constants>
    <parameters>
      <parameter name="exit_word" data-type="string">
        <description>Palabra con la que salir del programa.</description>
      </parameter>
      <parameter name="nechos" data-type="int">
        <description>Número de echos realizados.</description>
      </parameter>
    </parameters>
    <actions>
      <action name="echo" timeout="5">
        <value>rlfdummytool --echo</value>
        <description>Repite la entrada por teclado.</description>
        <action_parameter name="exit_word" type="in"/>
        <action_parameter name="nechos" type="out"/>
      </action>
      <action name="cpu-info" timeout="7">
        <value>rlfdummytool --cpu</value>
        <description>
          Obtiene información básica sobre el sistema (nombre de
          la máquina y memoria libre) bajo petición del usuario.
        </description>
      </action>
      <resetter>rlfdummytool --clean</resetter>
    </actions>
  </tool>

```

Son pocas las diferencias que hay entre las dos configuraciones, sólo denotar el atributo *data* y la forma de declarar las acciones. Cada tipo de herramienta tiene un DTD para comprobar la estructura de la configuración (llamados *tool.dtd* y *data-tool.dtd*).

```

<!-- EJEMPLO DE UNA HERRAMIENTA DE DATOS -->
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tool SYSTEM
  "/home/rodriguezmecha/Universidad/Proyecto/rdf/tools/data-tool.dtd">
<tool
  path="/home/rodriguezmecha/Universidad/Proyecto/rdf/tools/linux/RLF_Video"

```

```
    data="true"
  >
    <out-stream/>
    <attributes>
      <name>RLF Video</name>
      <version>0.1</version>
      <description>
        Streaming de video y sonido. Incluye 3 segundos de delay
        en el envío. Es necesario utilizar programas externos para
        poder obtener el flujo de datos.
      </description>
      <admin>carlos</admin>
      <role>0</role>
    </attributes>
    <constants>
      <constant name="devices" data-type="string">
        <value>
          v4l://dev/video0:input-slave=alsa://:v4l-norm=0
          :v4l-frequency=0:file-caching=300
        </value>
        <description>Dispositivos de captura.</description>
      </constant>
      <constant name="http_ip" data-type="string">
        <value>192.168.1.128</value>
        <description>IP del host.</description>
      </constant>
      <constant name="http_port" data-type="int">
        <value>64000</value>
        <description>Puerto de acceso.</description>
      </constant>
      <constant name="user" data-type="string">
        <value>video</value>
        <description>Usuario para la conexión.</description>
      </constant>
      <constant name="pass" data-type="string">
        <value>video</value>
        <description>Contraseña del usuario.</description>
      </constant>
    </constants>
    <action name="http-streaming" timeout="60">
      <value>rlfvideo --http</value>
      <description>
        Streaming por video mediante el protocolo HMTL. Utilizar un
        programa como VLC (con recepción de imágenes y sonido por
        volcado de red) para conectarse al servidor (http://nombre
        del servidor:puerto/)
      </description>
      <socket port="64000" protocol="http" type="media" mode="r"/>
    </action>
    <resetter>rlfvideo --clean</resetter>
  </tool>
```

En primer lugar se encuentra la cabecera de la configuración, donde se indica el tipo de herramienta, su esquema DTD correspondiente (incluido en el proyecto) y su carpeta *root* bajo el

nombre de *path*. A continuación se especifica si se dispone de entrada y salida estándar (*in-stream* y *out-stream*).

Después están los atributos de cada herramienta, donde se definen su nombre, descripción, administrador (debe estar registrado en la base de datos), versión y rol (número donde el 0 es la base, y es más restrictivo según se va añadiendo valores).

Las constantes definen valores fijos que no pueden ser cambiados, pero que todas las acciones pueden utilizar y obtener. Se componen de un tipo de datos, nombre, valor y descripción.

Los parámetros son definidos sólo en las herramientas normales, y se componen de un nombre, descripción, un valor máximo, mínimo y por defecto (opcional). Más adelante se les relaciona con las acciones.

Las acciones son diferentes comandos con un nombre, una descripción y un tiempo máximo de ejecución. A partir de ahí, se definen los parámetros que tienen asociados (indicando si son de entrada, salida o ambos). También tienen asociadas los *sockets* que utilizará el comando para proveer el servicio externo, indicando su puerto, el protocolo, el tipo (media, datos o gráficos) y si es de lectura, escritura o ambos. Hay que recordar que esos comandos serán llamados a partir del directorio root, es decir, en el primer caso, la acción “cpu-info” se llamará de la forma <path>/rlfdummytool --cpu.

Como acción especial está el *resetter* o limpiador, que no podrá ser ejecutado por el usuario y sirve para establecer a un estado seguro el *hardware* asociado a la herramienta. No tiene parámetros ni servicios externos.

Uso de la librería *libtool*

Como ya se ha comentado, esta librería sirve para poner en contacto a la herramienta con la plataforma, y es necesario utilizarla para poder registrarla. Se ha añadido a este proyecto tres librerías escritas en C/C++, Bash y .NET, compartiendo todas la misma estructura de funciones:

1. **RLF_Init:** Inicia la conexión con la plataforma. Es necesario introducir la clave que se proporcionó al registrar la herramienta. Sin esta función no se puede acceder a ninguna otra. Si se está ejecutando una acción la cual no ha sido ordenada por la plataforma, no se permitirá el acceso a la misma.
2. **RLF_Finalize:** Desconecta la herramienta, indicando un código de finalización (establecido por el administrador) y una descripción. Es obligatorio realizarlo antes de cerrar la aplicación o antes de que no haya una finalización concreta (por ejemplo, antes de un bucle infinito). Hay que tener en cuenta que la aplicación puede ser detenida en cualquier momento por varios motivos, por lo que debe estar preparada para ello.
3. **RLF_GetConst:** Obtiene la constante indicada por el nombre y es almacenada en la estructura correspondiente (ver código para más información).
4. **RLF_GetAttribute:** Obtiene el atributo correspondiente con el nombre introducido. Estos atributos son gestionados por la plataforma.
5. **RLF_GetParameter:** Obtiene el valor y la información de cualquier tipo de parámetro relacionado con la acción en ejecución actual. Estos valores han sido establecidos por el usuario.
6. **RLF_SetParameter:** Establece el valor textual de un parámetro de salida que esté asociado a la misma acción.

7. **RLF_ThrowException:** Lanza una excepción al usuario con un nombre y una descripción. No interrumpe el ciclo de ejecución.

NOTA: Para obtener más información sobre las librerías específicas en cada lenguaje consultar el código entregado.

A continuación se muestra un ejemplo de una acción programada en Visual C++ que obtiene la memoria libre del sistema y la envía a la salida estándar cada cierto tiempo. Es parte de una herramienta de datos:

```
// 1. Inicio RLF
RLF_Manager ^ manager = gcnew RLF_Manager();
try {
    manager->init(TOOLKEY);
} catch (RLF_Exception ^ e){
    Console::WriteLine("Error con la base de datos al iniciar. " + e->getMsg());
    stream->Flush();
    return 1;
}

// 2. Obtención de la constante de tiempo.
try {
    time = Convert::ToInt32(manager->getConst("time")->getValue());
} catch (RLF_Exception ^ e){
    Console::WriteLine("Error con la base de datos al obtener datos." + e->getMsg());
    try {
        manager->finalize(1, "Error.");
    } catch (...){
    }
    return 1;
}

try {
    manager->finalize(1, "Error.");
} catch (...){
    Console::WriteLine("Error con la base de datos al finalizar.");
    return 1;
}

// 3. Ejecución.
while(true){
    GlobalMemoryStatusEx (&statex);
    Console::Write("Memory in use: {0:G}% ({1:D} / {2:D} Kbytes)",
                  statex.dwMemoryLoad, statex.ullAvailPhys/DIV,
                  statex.ullTotalPhys/DIV);
    Thread::Sleep(1000 * time);
}
```

Buenas prácticas

Debido a que las aplicaciones que se programen deben seguir unos estándares, es recomendable seguir estos pasos:

- Las librerías que se utilicen para ejecutar las acciones, incluso la librería *libtool* deben estar en la carpeta *lib* dentro del directorio *root* de la propia herramienta.
- Los usuarios sólo verán los valores de los parámetros de salida cuando se termine de ejecutar la acción, por lo que escribir varias veces un valor no tiene sentido.
- Los *buffers* de salida son tratados como en los ficheros, por lo que si se quiere mostrar en tiempo real al usuario, será necesario vaciarlos o reducir su tamaño a cero. (Esto se puede conseguir mediante la función *flush* contenida en muchos lenguajes de programación.)
- Si se usa en una misma herramienta distinto *hardware* dependiendo de la acción a tomar, es mejor separarlas y dividirlas en varias herramientas, para que los usuarios puedan disponerlas de manera más efectiva.

Crear una nueva *libtool*

Puede ser necesario crear nuevas librerías para otros lenguajes, como Java, Python e incluso el *script* Bat de Windows. Para ello es necesario tener instalado la librería de acceso a la base de datos SQLite ya que la comunicación se realiza mediante ese formato (se puede obtener más información sobre las APIs de SQLite en la siguiente dirección <http://www.sqlite.org/>).

En el desarrollo de la librería hay que tener en cuenta las consultas a la base de datos, y la protección de escritura y lectura. Como explicar esto puede conllevar un aumento de la documentación bastante grande, se puede consultar las librerías ya hechas que están profusamente comentadas para ver cómo desarrollar una nueva, ya que la estructura es esencialmente la misma y su aplicación es sencilla.

La figura ?? muestra el modelo que sigue la base de datos asociada a cada herramienta.

Apéndice E

Guía de usuario

Este manual está destinado al usuario final de la plataforma RLF. Contiene las acciones a realizar con el cliente de escritorio así como la utilización del monitor mediante web.

RLF_ Client

Este programa da acceso a todas las herramientas incluidas en el sistema RLF al que se va a conectar. Funciona tanto en Windows como en plataformas Linux. Aunque oficialmente no está soportado, también funciona en plataformas Macintosh que sean compatibles con la máquina virtual Java.

Instalación

Para poder ejecutar el cliente, es necesario tener con anterioridad unas herramientas instaladas en el ordenador, independientemente del sistema operativo y son gratuitas:

JRE 1.6 de Java Sun: Contiene la máquina virtual y las librerías necesarias para ejecutar programas en el entorno Java. Se puede descargar de <http://java.com/es/download/> aunque para sistemas Linux se puede encontrar en los repositorios oficiales con el nombre de `sun-java6-jre`.

(Opcional) Reproductor media VLC: Sólo es necesario si se requieren usar herramientas de vídeo. Se puede encontrar en la página oficial (<http://www.videolan.org/vlc/>) para cualquier plataforma. Al igual que antes, los repositorios de la mayoría de las distribuciones Linux bajo el nombre de `vlc`.

(Opcional) Navegador de Internet: Sólo es necesario para algunas herramientas concretas que utilicen servicios como FTP. Debido a que existen multitud de navegadores, se puede escoger el que quiera, incluso el que venga instalado por defecto en los sistemas operativos.

La instalación de estos programas se puede encontrar en Internet sin complicación, por lo que no se dedicará tiempo a explicarlo en este manual.

Una vez que esto esté listo, se puede pasar a la instalación del cliente. Para ello, es necesario obtener los archivos de la carpeta `rlf/bin/client` (`rlf\bin\client` en Windows) y copiarlos donde se desee. Es importante que las carpetas `res` y `lib` se copien en la misma ubicación donde está el archivo ejecutable `RLF_ Client.jar`.

Configuración

Para que el cliente se conecte a una dirección específica (que será dada por el administrador) donde se encuentra el proveedor es necesario modificar el archivo de configuración `res/client.conf` (`res\client.conf` en sistemas Windows). Este fichero contiene la siguiente línea:

```
provider_url=http://[dirección] : [puerto] / [ruta]
```

Para configurarlo sólo es necesario modificar los parámetros *dirección*, *puerto* y *ruta* (eliminando los corchetes). Como por ejemplo:

```
provider_url=http://163.117.150.85:8084/RLF/Provider
```

Ejecución

Para ejecutar el programa sólo es necesario hacer “doble click” en el archivo RLF_Client.jar en cualquier sistema. Si como consecuencia se obtiene un mensaje de “No se reconoce el archivo ejecutable”, la instrucción para la línea de comandos es:

```
java -jar RLF_Client.jar
```

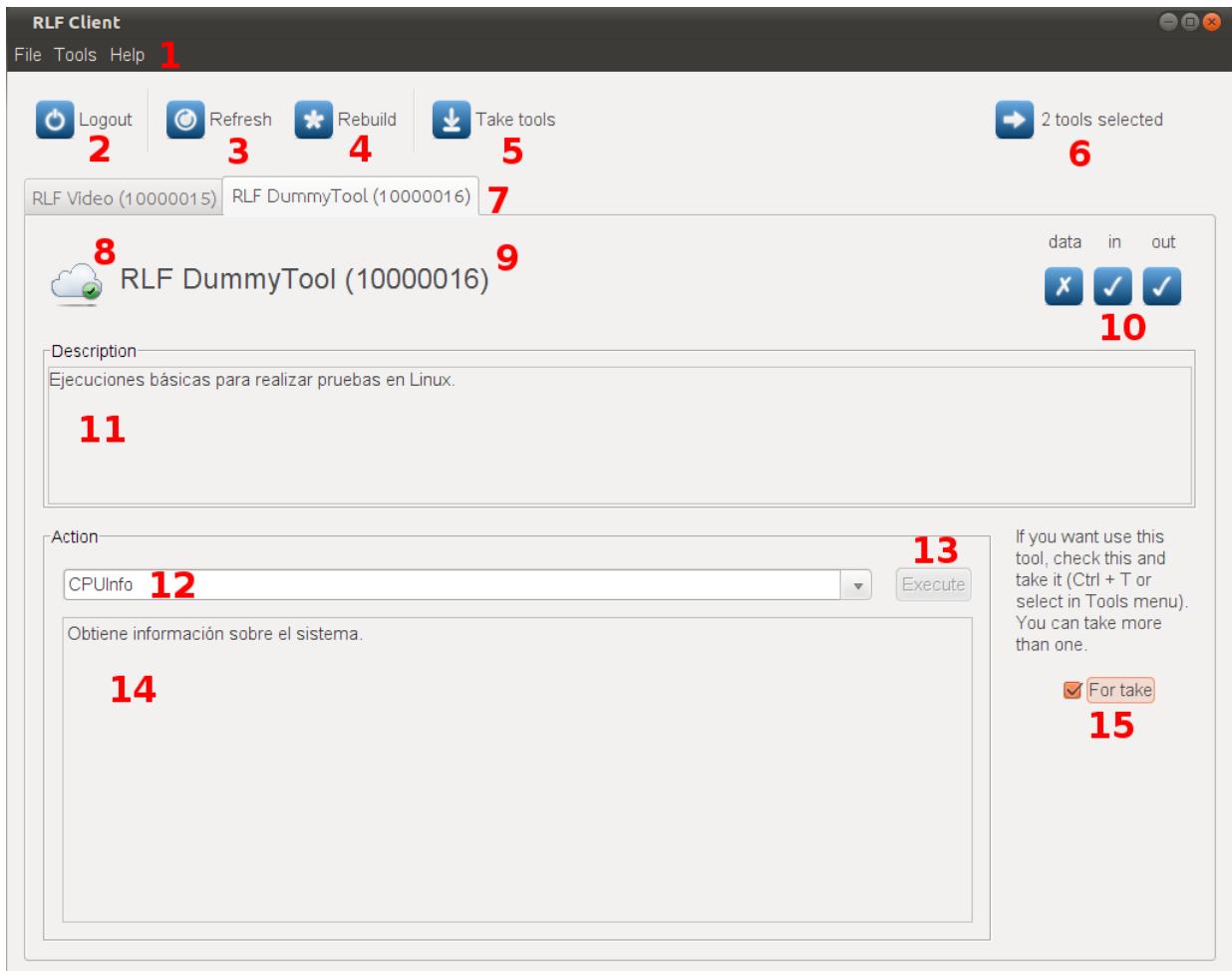
NOTA: En sistemas Linux puede haber errores con fallos de conexión o de la interfaz gráfica, son externos al programa y vienen dados por la máquina virtual:

- Hay que asegurar que no se está utilizando la versión OpenJDK de Java, que es la que viene por defecto en Linux. Esta versión tiene problemas (reconocidos oficialmente) con las interfaces gráficas Swing. Para arreglar este problema, seguir la guía que se puede obtener en esta dirección:

<http://www.e-capy.com/reemplazar-openjdk-por-el-jdk-de-java-en-ubuntu/>

- Siempre es recomendable utilizar el comando anteriormente descrito en vez de ejecutar el programa con “doble click” ya que Java establece sus rutas dependiendo de dónde haya sido ejecutado. Es por esto que puede que no encuentre las librerías o el fichero de recursos.

Interfaz



Aquí se listan todos los elementos que componen la interfaz del cliente mostrados en la anterior figura. Puede que en determinados momentos algunas de estas opciones no estén activas.

1. Es la barra de menús donde poder acceder a todas las acciones, como ruta alternativa a los otros botones. Además se indican los atajos de teclado de estas acciones.
2. Con este botón se puede conectar o desconectar el usuario al sistema.
3. Sirve para obtener el estado de cada herramienta antes de haberlas reservado.
4. Reconstruye la información de las herramientas por si ha habido cambios estructurales en el servidor.
5. Reserva las herramientas seleccionadas.
6. En esta zona se indican las herramientas que se han seleccionado, o el tiempo restante de reserva de las mismas. Si se posiciona el puntero del ratón encima se pueden obtener el nombre e identificador de cada herramienta seleccionada o reservada.
7. Cada pestaña corresponde a una herramienta distinta controlada por el sistema.
8. Estado de la herramienta, ver la sección “Estados de una herramienta” para más información.

9. Nombre e identificador único de cada herramienta.
10. Atributos de cada herramienta, ver la sección “Atributos de una herramienta” para más información.
11. Descripción de la herramienta.
12. Acción seleccionada de la herramienta. Sólo se pueden ejecutar una vez la herramienta haya sido reservada. Si se despliega la lista se podrán obtener todas las acciones asignadas.
13. Cuando la herramienta está reservada, con este botón se ejecuta la acción seleccionada.
14. Descripción de la acción seleccionada.
15. Sirve para seleccionar la herramienta de la pestaña actual para su posterior reserva.

Estados de una herramienta

Cada herramienta tiene asociado un estado con cuatro valores posibles que se corresponden a las cuatro imágenes de la siguiente figura:

- **Disponible:** Esta herramienta no está siendo usada por nadie y puede reservarse.
- **No disponible:** Esta herramienta está siendo usada por otro usuario o está desconectada del sistema.
- **Reservada:** La herramienta ya está reservada y lista para su ejecución.
- **En ejecución:** Alguna acción de esta herramienta está actualmente en ejecución.



Atributos de una herramienta

Cada herramienta posee tres atributos característicos que determinan su comportamiento. Se encuentran indicados en la zona 10 explicada en el apartado anterior:

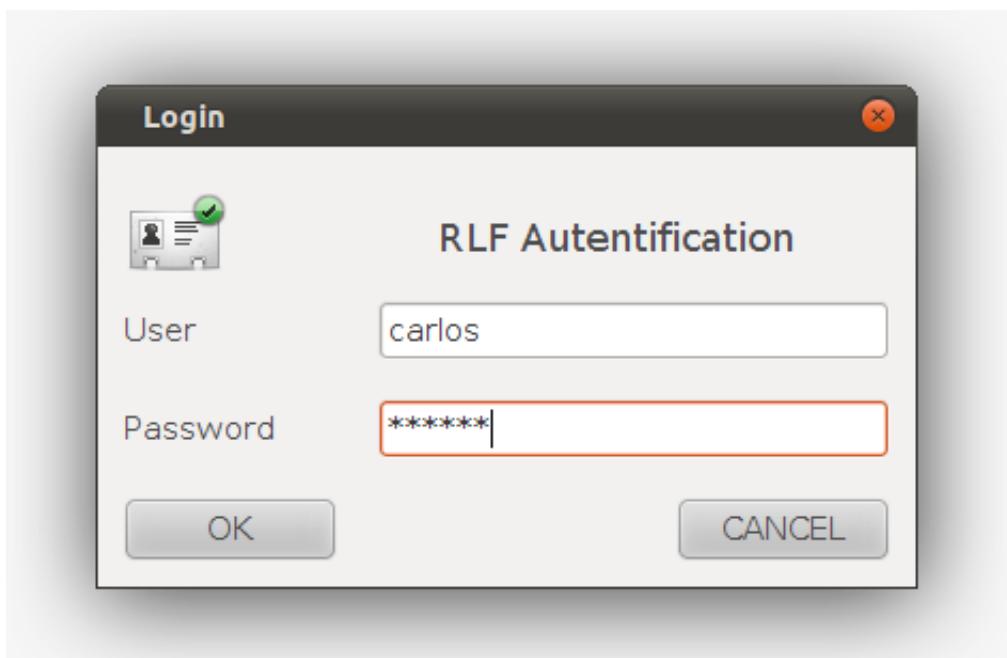
- **In:** Si está activo, la herramienta acepta interacción con el usuario mediante la entrada por teclado. Cuando se ejecute alguna acción, se podrán enviar comandos en el visor de ejecuciones.
- **Out:** Si está activo indica que la herramienta dará información textual al usuario cuando se ejecute una acción.
- **Data:** Indica si la herramienta tiene la categoría de “Herramienta de datos”. Este tipo de herramientas no tienen entrada por teclado, sólo contienen una acción y permiten que varios usuarios la utilicen a la vez, por lo tanto, esta herramienta siempre estará libre para su uso.

Cómo utilizar las herramientas

El proceso para obtener unas herramientas y realizar acciones con ellas se describe a continuación.

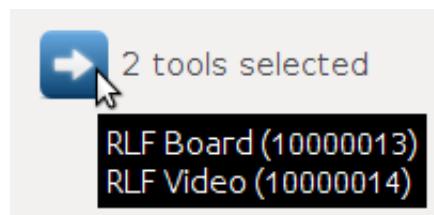
Conexión

En primer lugar se ha de conectar con el servidor, para ello se pulsa el botón de *Login* y se introduce el usuario y la contraseña asignada en el diálogo que aparecerá como se ve en la figura a continuación. A partir de ese momento, el resto de las opciones estarán disponibles.



Selección de las herramientas

Cuando las herramientas aparezcan, se podrá seleccionar aquellas que dispongan del estado **Disponible** mediante el *checkbox* propio de cada pestaña. Debido a que puede haber varios usuarios realizando peticiones, es recomendable actualizar (con el botón *Refresh*) los estados de las herramientas antes de seleccionarlas, ya que no se actualizan en tiempo real. Cuando se hayan seleccionado las herramientas, aparecerán en el indicador mostrado en la siguiente figura.



Reserva

Para poder utilizar unas es necesario reservarlas antes. Cuando todas las herramientas que se deseen estén seleccionadas, se procederá a reservarlas con el botón *Take Tools*. A partir de aquí, el cronómetro se pondrá en marcha, ya que cada usuario tiene un tiempo máximo de uso de las herramientas.

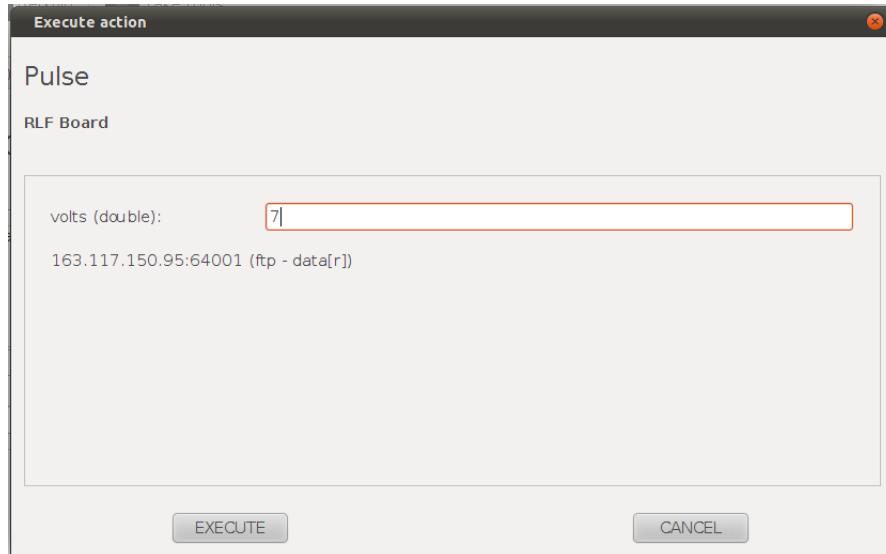


ATENCIÓN: Cuando el cronómetro está cerca del límite de tiempo lanza un aviso mediante un diálogo, después, cuando se ha terminado cierra todas las acciones en ejecución y libera las herramientas.

Ejecutar una acción

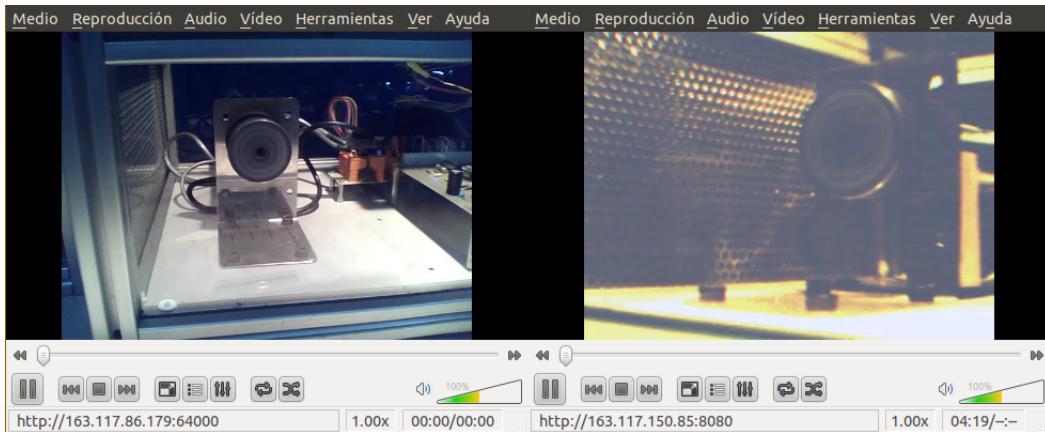
Cada herramienta puede ejecutar a la vez una sola acción, pero se pueden tener varias acciones de distintas herramientas ejecutándose en el mismo instante. Para ello, selecciona la acción correspondiente y se pulsa el botón *Execute*.

Se mostrará un diálogo con los parámetros de entrada de la acción y con los *sockets* para servicios externos. En los parámetros se indica el tipo y si se pasa el ratón por encima del nombre, se podrá obtener la descripción del mismo.



En el caso de esta figura, hay un parámetro de entrada llamado “volts” de tipo *double* y un servicio externo de FTP de lectura al que se puede acceder mediante la URL `ftp://163.117.150.95:64001/`. Para ello se utiliza un navegador o un cliente FTP.

ATENCIÓN: Los *sockets* representan servicios que provee la herramienta pero que deben ser atendidos fuera de la interfaz de RLF Client. Esto significa que es necesario utilizar un programa externo, como puede ser un navegador o un reproductor de vídeo. La información que se muestra es la localización o *URL* de ese mismo servicio, así como su protocolo y el tipo de datos que obtiene.

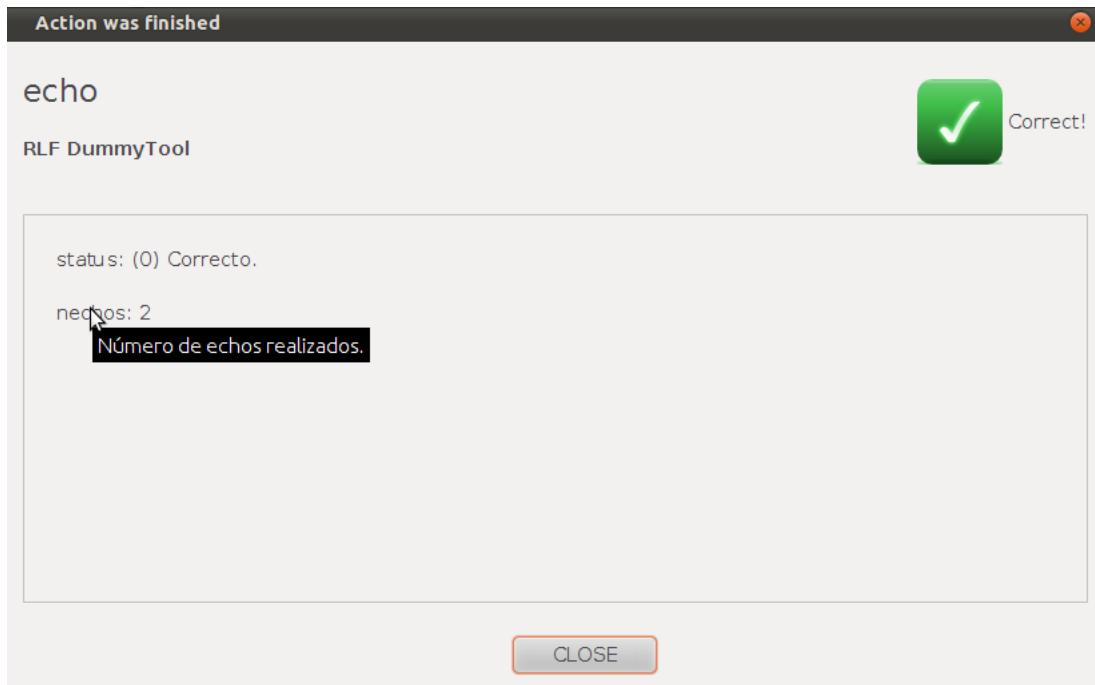


Cuando se tengan configurados los parámetros, se podrá proceder a ejecutar la acción. Si la máquina donde se encuentra la herramienta tiene mucha carga, puede ser necesario esperar hasta que se pueda ejecutar. Se sabe que la ejecución está en marcha cuando el ícono muestra el título “Running...”.



Interactuar con una acción

Dependiendo del tipo de herramienta, en el diálogo de ejecución se podrá obtener la información de salida de la herramienta e introducir comandos mediante el campo de texto y el botón *Send*. Se podrá terminar la ejecución abortándola (cerrando la ventana) o completando la acción, la cual lanzará otra ventana con los resultados de la aplicación, que conforman el estado final, los parámetros de salida y las excepciones lanzadas, como se ve en la siguiente figura.



Liberar las herramientas

Tanto si se ha terminado de utilizar las herramientas como si se ha acabado el tiempo, se ha de desconectar mediante el botón *Logout* o cerrando la aplicación.

Errores comunes

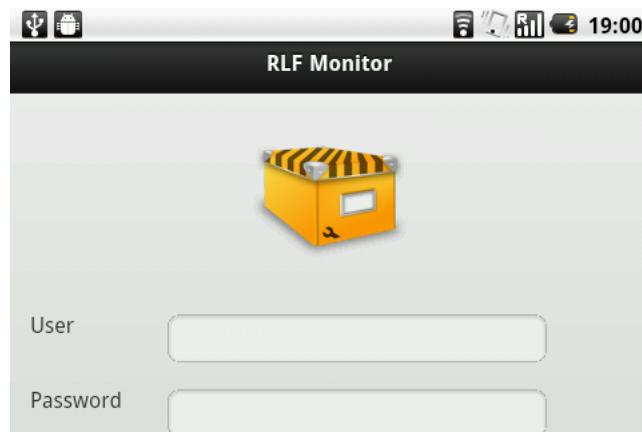
Por motivos de seguridad, si ocurriera algún error grave en la aplicación RLF_ Client y el usuario no se pudiera desconectar con normalidad, no podrá volverse a conectar hasta que el administrador haya dado su aprobación, para ello, póngase en contacto con él e indique el motivo de su error.

RLF - Monitor

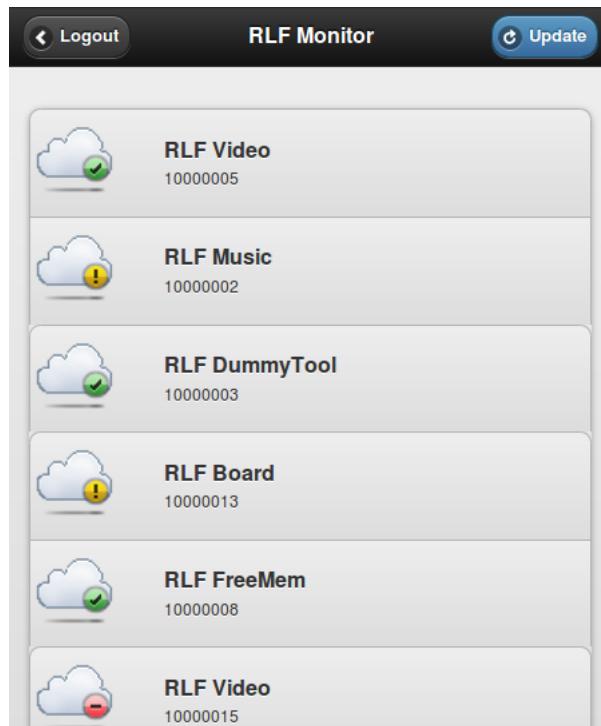
Este servicio está disponible para poder comprobar en tiempo real y desde cualquier dispositivo el estado de las herramientas. Aunque la interfaz esté diseñada especialmente para dispositivos móviles, puede ser consultado desde un navegador de un ordenador personal sin problemas. Con ello se pretende ahorrar tiempo al usuario para informarse si las herramientas que quiere utilizar están en ese momento libres.

Para acceder al servicio, se debe ir a la dirección proporcionada por el administrador mediante cualquier tipo de navegador de Internet. Como por ejemplo:

<http://163.117.150.85:8080/RLF/Monitor/>



Esa página pedirá el nombre y contraseña del usuario que se le ha asignado, una vez esté conectado, podrá obtener información como esta:



Estados de una herramienta en el monitor

La lista de estados que se pueden obtener mediante el servicio del monitor es distinta a la que se puede observar con la aplicación cliente RLF- Client:

- **Disponible:** Esta herramienta no está siendo usada por nadie y puede reservarse.
- **No disponible:** Esta herramienta está desconectada del sistema.
- **En uso:** Algún usuario tiene reservada esta herramienta.



Apéndice F

Material entregado

Debido a que este producto contiene múltiples módulos y componentes, se listan a continuación todo lo se ha entregado junto con este documento, y su disposición.

Componentes

Proveedor

Compone el servicio web de acceso a la plataforma y los *scripts* de la base de datos para poder crearla. Este servicio web debe ser desplegado en alguna plataforma como Tomcat o JBoss que permita aplicaciones Java. Los *scripts* están realizados para el SGBD MySQL.

Monitor

Sirve para obtener información en tiempo real de las herramientas registradas en el sistema. Está compuesto por un servicio y página web con formato para dispositivos móviles o pantallas pequeñas.

Laboratorio

Módulo de gestión de los distintos servidores locales que contienen herramientas. Puede instalarse en cualquier plataforma y toma la función de un proceso *demonio*.

Gestor de laboratorios

Es la herramienta para poder administrar todos los laboratorios ejecutados en la red. Aplicación por consola que se recomienda usar en sistemas Unix.

Cliente

Permite acceder a la plataforma RLF. Puede ser utilizado en cualquier sistema operativo con soporte para Java. Utiliza el servicio web propio del proveedor.

Componentes RLF

Son un conjunto de herramientas que utilizan todos los módulos. Dispone de un cifrador de datos y la capa de comunicación de toda la plataforma mediante sockets.

Bibliotecas *libtool*

Conjunto de funciones y métodos específicos para conectar la herramienta con el laboratorio. Están hechas en varios lenguajes de programación.

Herramienta 1: RLF_ Video

Emite video y sonido mediante un servidor HTTP. Está destinada para sistemas Unix fue desarrollada en Shell Bash. Su capa base es el programa/servidor VLC. Es una herramienta multiusuario.

Herramienta 2: RLF_ Music

Reproduce canciones seleccionadas por el usuario en el sistema remoto, por lo que no pueden ser oídas desde el cliente. Pensado para sistemas Unix con el programa MPG123.

Herramienta 3: RLF_ DummyTool

Conjunto de pruebas básicas en consola Shell Bash para Unix. Comprueba la información del sistema en tiempo real.

Herramienta 4: RLF_ FreeMem

Obtiene la memoria en uso de un sistema Windows. Esta herramienta es multiusuario y la información también es en tiempo real.

Herramienta 5: RLF_ Board

Acciones para utilizar la tarjeta Advantec PCI1711BE. Desarrollado para sistemas Windows. Pensado para realizar prácticas de laboratorio de electrónica y automática. Permite enviar un pulso de 5 segundos con el voltaje introducido, obteniendo la respuesta mediante un servicio FTP.

Documentación

Todo el código entregado está profusamente comentado para ayudar a desarrollar nuevos módulos a partir de código antiguo. Además, para el código escrito en Java, se ha entregado la documentación del API completa (realizada mediante *javadoc*).

Distribución

La disposición de directorios entregados es la siguiente:

```
|-- bin
|   |-- client
|   |   |-- lib
|   |   |-- log
|   |   '-- res
|   |-- lab
|   |   |-- log
|   |   '-- res
|   |-- labconsole
|   '-- provider
|-- doc
|   |-- api
|   |   |-- index-files
|   |   |-- org
|   |   '-- resources
|   |-- images
|   '-- user
|   |-- include
|   |-- lib
|   '-- template
|-- img
|-- lib
|-- projects
|   |-- eclipse
|   |   |-- RLF_Lab
|   |   |-- RLF_LabConsole
|   |   |-- RLF_LabManager
|   |   |-- RLF_Log
|   |   '-- RLF_Utils
|   |-- netbeans
|   |   |-- RLF_Client
|   |   '-- RLF_Provider
|   '-- visual studio
|       |-- LibTool
|       |-- RLF_Board
|       '-- RLF_FreeMem
|-- src
|   '-- sql
`-- tools
    |-- lib
    |   |-- Bash
    |   |-- C
    |   '-- NET
    |-- linux
    |   |-- RLF_DummyTool
    |   |-- RLF_Music
    |   '-- RLF_Video
    '-- windows
        |-- RLF_Board
        '-- RLF_FreeMem
```

Directorio *bin*

Contiene todos los archivos ejecutables de los distintos componentes organizados por carpetas. Si se desea ejecutar esos archivos binarios en otra localización, es necesario copiar todo lo que contienen sus carpetas, ya que es necesario para un correcto funcionamiento.

Directorio *doc*

En este directorio está toda la documentación aportada para el proyecto, desde la descripción de los APIs hasta este propio documento, en *latex* con todos los recursos necesarios para la maquetación del mismo.

Directorio *img*

Se agrupan todas las imágenes que se han utilizado para el desarrollo del proyecto. Son las usadas en los componentes aunque no son necesarias para su ejecución, ya que han sido introducidas en el mismo código.

Directorio *lib*

Aquí están todas las librerías requeridas para la compilación de los distintos módulos. Cada proyecto tiene su referencia en este directorio. Además, se puede encontrar la propia librería de RLF.

Directorio *projects*

Se ha considerado que es más sencillo realizar nuevos módulos o modificar los entregados mediante los proyectos originales de los distintos IDEs, como son Eclipse, NetBeans y Visual Studio. Aquí está contenido de forma organizada haciendo referencia al tipo de IDE.

Directorio *src*

Se encuentran los archivos fuente de los componentes que no tienen establecido su propio proyecto. Es el caso de los *scripts* en SQL de las distintas bases de datos.

Directorio *tools*

Contiene todas las herramientas entregadas (sus archivos ejecutables) separadas por plataforma y las propias *libtools* en los distintos lenguajes. Antes de mover estos archivos consultese el Manual del Desarrollador.

Bibliografía

- [1] Características del servicio amazon ec2. *Amazon Elastic Compute Cloud*. Consultado el 15 de Septiembre de 2011.
<http://aws.amazon.com/es/ec2/>.
- [2] Diccionario de informática alegsa. Consultado el 11 de Septiembre de 2011.
<http://www.alegsa.com.ar/Dic/>.
- [3] Documentación de vmware. *VMWare*. Consultado el 20 de Septiembre de 2011.
<http://www.vmware.com/es/virtualization/virtualization-basics/what-is-virtualization.html>.
- [4] Ieee history center. Consultado el 9 de Septiembre de 2011.
http://www.ieee.org/about/history_center/index.html.
- [5] Opennebula documentation. *OpenNebula.org*. Consultado el 19 de Septiembre de 2011.
<http://www.opennebula.org/documentation:knowledge>.
- [6] Real world performance metrics: java.io vs. java.nio. *Geekomatic*. Consultado el 24 de Septiembre de 2011.
<http://geekomatic.ch/2009/01/16/1232134440000.html>.
- [7] Velocidad del 3g por comunidades. *ADSL Zone*. Consultado el 26 de Septiembre de 2011.
<http://www.adslzone.net/estudio-velocidad-bandaancha-movil-3g.html>.
- [8] Guide to applying the esa software engineering standards to small software projects. *ESA Board for Software Standardisation and Control*, Mayo 1996. Documentos BSSC(96)2 Issue 1 y ESA PSS-05-0 Issue 2.
- [9] *The Oxford Companion to Philosophy*. Oxford University Press, 2005.
- [10] Ludvigson, Erik Bernstein, David. Protocols and formats for cloud computing interoperability. *Computer.org*, Mayo 2009. Consultado el 11 de Septiembre de 2011.
<http://www.computer.org/portal/web/csdl/doi?doc=doi/10.1109/ICIW.2009.55>.
- [11] García Carballeira, Félix Carretero Pérez, Jesús. *Sistemas Operativos, Una visión aplicada*. Mc Graw Hill, 2007.
- [12] Chapman, Davis. *Visual C++ .NET*. Sams, 2001.
- [13] Codd, E.F. *A Relational Model of Data for Large Shared Data Banks*. ACM, 1970.
- [14] Expósito Singh, David. *Curso de Desarrollo de Aplicaciones Distribuidas*. Grupo ARCOS, Universidad Carlos III de Madrid, 2010.
- [15] Grosso, William. *Java RMI*. O'Reilly, 2002.
- [16] Hitchens, Ron. *JavaNIO*. O'Reilly Media, 2002.

- [17] Huitema, Christian. Challenges of the next generation networks. *Keynote for Internet'99*, Octubre 1999. Consultado el 13 de Septiembre de 2011.
<http://huitema.net/papers/challenges/challenges99.html>.
- [18] Jones, D.S. *80x86 Assembly Programming*. Oxford Science Publications, 1991.
- [19] Martínez-Val, José M^a. *Diccionario Enciclopédico de Tecnología*. Editorial Síntesis, 2000.
- [20] Stallings, William. *Comunicaciones y Redes de Computadores*. Pearson Prentice Hall, 2004.
- [21] Stevens, W. Richard. *TCP/IP Illustrated, Volume 1*. Addison-Wesley Publishing Company, 1994.
- [22] Rocío Sánchez, FernandoArango. *El Kernel 2.4 de Linux*. Prentice Hall, 2002.
- [23] Huecas Fernández-Toribio, Gabriel Sánchez Allende, Jesús. *Java 2*. McGraw-Hill, 2005.
- [24] Tanenbaum, Andrew S. *Redes de computadoras, Cuarta edición*. Pearson, Prentice Hall, 2003.
- [25] O'Connor, John y Robertson, Edmund F. *Biografía de Alan Mathison Turing*. MacTutor History of Mathematics archive.

Índice alfabético

- .NET, 26, 28
 - ASP.NET, 28
 - interoperabilidad, 30
- libtool*, 44, 144
- Kernel*, 73
- LabConsole*, 77
- backup*, 120
- libtool*, 76
- log*, 120
- middleware*, 4, 18
- ARPA, *véase ARPANET*
- base de datos, 22
 - MySQL, 22
 - SQL, 10
 - SQLite, 22
- cifrador, 144
- cliente, 36, 144
- cloud computing, *véase “nube”*
- CORBA, 11, 27
- demonio, 144
- domótica, 14
- gestión de laboratorios, 77
- gestor de comunicaciones, 74
- gestor de ejecución, 74
- gestor de herramientas, 74
- herramienta, 36
- Internet, 10, 11
 - ARPANET*, 9
 - navegador, 29
 - OpenNebula, 13
 - www, 11
- Java, 11, 20, 112
 - interoperabilidad, 30
 - J2EE, 21
 - Java RMI, 28
 - máquina virtual, 26, 27
 - nio, 96
- laboratorio, 36, 73, 144
- monitor, 144
- PCI-1711-BE, 98
- proveedor, 36, 144
- rol, 117, 127
- servicio web, 28
 - WSDL, 28
- servidor, 107
- sistema distribuido, 18
 - “nube”, 13, 18
- socket, 26, 144
- TCP/IP, 24
- Tomcat, 114
- virtualización, 13, 19