



**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
DE TELECOMUNICACIÓN**

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

SISTEMA DE COMUNICACIÓN PERSISTENTE BASADO EN
WEBSOCKETS, OPTIMIZACIÓN Y ADAPTACIÓN A
DISPOSITIVOS MÓVILES DE LA HERRAMIENTA DE
E-LEARNING FLEQ

AUTOR: JORGE GONZÁLEZ NAVARRO

TUTOR: GREGORIO ROBLES MARTÍNEZ

Curso Académico 2012/2013

Proyecto Fin de Carrera

SISTEMA DE COMUNICACIÓN PERSISTENTE BASADO EN WEBSOCKETS, OPTIMIZACIÓN Y ADAPTACIÓN A DISPOSITIVOS MÓVILES DE LA HERRAMIENTA DE E-LEARNING FLEQ

AUTOR

JORGE GONZÁLEZ NAVARRO

TUTOR

GREGORIO ROBLES MARTÍNEZ

La defensa del presente Proyecto Fin de Carrera se realizó el día de
de , siendo calificada por el siguiente tribunal:

PRESIDENTE:

SECRETARIO:

VOCAL:

y habiendo obtenido la siguiente calificación:

CALIFICACIÓN:

Fuenlabrada, a de de .

Copyright © 2012 Jorge González Navarro

Este documento se publica bajo la licencia

Creative Commons Reconocimiento-CompartirIgual 3.0 España

<http://creativecommons.org/licenses/by-sa/3.0/es>

(Ver Apéndices)

A mi familia

*Felicidad no es hacer lo que uno
quiere sino querer lo que uno hace.*

Jean Paul Sartre

Agradecimientos

Podría escribir varias páginas con poco que hiciera memoria. Muchas han sido las personas que han hecho posible que haya llegado hasta aquí, tanto en el ámbito personal como en el académico, incluso en el profesional, en mi hasta ahora corta carrera.

Desde mis comienzos en el colegio, pasando por el instituto, hasta llegar a los años universitarios. Pero por muy importantes que hayan podido ser estas personas, ninguna alcanza el grado de importancia que tienen mis padres, Pedro y Elena, en que haya llegado a poder escribir estas líneas.

Ellos son los verdaderos artífices de que hoy sea quien soy, y de que mañana llegue a ser quien pueda llegar a ser. Ni sumando todo mi esfuerzo durante estos años se podría comparar al sacrificio que ambos han realizado incansablemente para darme todo y poder conseguir este hito que sé, les llena de orgullo. El mismo orgullo que siento yo de poder escribir todo esto con la certeza de que es completamente cierto. En realidad, esto es vuestro, de los dos. Gracias por regalármelo.

Otra persona que me gustaría mencionar aunque ya no esté es mi abuelo Pablo. En realidad, toda mi familia debería tener un pequeño apartado aquí, pero probablemente él sea de quien más aprendí gran parte de los valores que hoy poseo, porque desde bien pequeño pasé gran parte de mi vida junto a él y junto a mi adorada abuela Sagrario, en aquellos interminables veranos en el pueblo. Desde el cielo él y desde un poco más cerca ella, sé que estarán orgullosos de mí.

En el ámbito académico, gracias a todas las personas que me han apoyado en los momentos difíciles y a las que han confiado en mí, como mi tutor del proyecto, Gregorio, que siempre me tendió una mano cuando la necesité durante estos años y no dudó ni un instante en aceptarme como uno de sus proyectandos para hacer crecer una de sus criaturas más queridas: FLEQ.

También me gustaría expresar mi agradecimiento a todas aquellas personas que no apostaban ni un céntimo porque yo llegara hasta aquí, gracias sinceramente, porque probablemente, ese estímulo fue en ocasiones más grande que el que me aportaban otros.

Dejo lo mejor para el final... Sin ella, probablemente este día no sería así de especial. Porque ella es quien le ha dado sentido a toda mi vida, y quien se la seguirá dando de aquí en adelante. Muchas gracias por todo el apoyo que me has dado durante estos maravillosos dos años. Te amo Cris.

Gracias también a mi hermana Nieves por todo lo que hemos pasado y pasaremos juntos, y a todo el resto de mi familia, amigos, compañeros y profesores.

Gracias de todo corazón. Siempre os llevaré conmigo.

Jorge

Resumen

En este proyecto se pretende extender y hacer accesible la aplicación e-learning FLEQ desde dispositivos móviles de última generación.

Para ello, el primer paso que se dará será desligar la aplicación del actual sistema de comunicación utilizado para la realización de partidas, el sistema de chat IRC, lo que además servirá para que la aplicación quede integrada de manera completa, sin dependencia de sistemas de terceros.

La primera acción a llevar a cabo será el estudio de las diferentes tecnologías candidatas a sustituir al IRC, haciendo un análisis exhaustivo de las ventajas e inconvenientes que conlleva cada una de las alternativas, estudio en el que entrarán en juego factores tan importantes como las actuales redes inalámbricas.

Este estudio permitirá evaluar qué sistema será el más adecuado para desarrollar una aplicación viable desde el punto de vista inalámbrico, así como desde el punto de vista de desarrollo.

Se deberá tener en cuenta la fragmentación del mercado de dispositivos móviles actual, en el que conviven terminales muy diversos, con lo que se planteará una de las cuestiones más candentes en la actualidad tras la llegada de HTML5: aplicaciones móviles nativas vs. aplicaciones móviles multiplataforma.

Todo esto tendrá como objetivo final desarrollar un ecosistema en el que tengan cabida el mayor número posible de dispositivos del mercado actual, y lo que es más importante, que esta cantidad pueda aumentar en el futuro. Por tanto, se ha de desarrollar un sistema que haga uso de tecnologías jóvenes y potentes que permitan que el ciclo de vida del producto sea lo más extenso posible.

Índice general

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	3
1.3. Estructura de la memoria	5
2. Estado de la Ciencia	7
2.1. Aplicaciones cliente-servidor	7
2.1.1. Servidor	8
2.1.2. Cliente	9
2.2. Real Time Web	9
2.2.1. Tecnologías PULL	10
2.2.2. Tecnologías PUSH	11
2.3. Tecnologías web	16
2.3.1. HTML5	17
2.3.2. CSS3	18
2.3.3. JavaScript	20
2.3.4. jQuery	21
2.4. Python, Django y Tornado	22
2.4.1. El lenguaje Python	22

2.4.2.	Django	24
2.4.3.	Tornado	26
3.	Análisis de tecnologías	27
3.1.	Comparativa de tecnologías del sistema de comunicación	28
3.1.1.	PUSH vs PULL	28
3.1.2.	Comparativa técnicas PUSH	29
3.1.3.	Selección final: Socket IO	31
3.2.	Comparativa de tecnologías para el diseño de clientes móviles	35
3.2.1.	Aplicaciones móviles nativas vs HTML5	35
3.2.2.	Selección final: HTML5	37
3.3.	Comparativa de lenguajes y frameworks para la implementación del servidor	38
3.3.1.	Selección final: Python	39
4.	Diseño e implementación	43
4.1.	Introducción	43
4.2.	Diseño e implementación del servidor	44
4.2.1.	Modelo de datos	44
4.2.2.	Arquitectura REST	46
4.2.3.	Aplicación web	48
4.2.4.	Servidor web	59
4.3.	Diseño e implementación del cliente	63
4.3.1.	Introducción	63
4.3.2.	Diseño del cliente	64
4.3.3.	Implementación cliente móvil	69

4.3.4. Implementación cliente de escritorio	76
5. Despliegue y resultados	79
5.1. Introducción	79
5.2. Pruebas internas	80
5.3. Prueba real: SATpeonato	82
6. Conclusiones	91
6.1. Análisis de objetivos	91
6.2. Estimación del trabajo realizado	95
6.3. Futuras líneas de trabajo	96
6.4. Publicación del código	99
6.5. Valoración final	100
Bibliografía	103
Apéndices	104
A. Planificación del proyecto	107
B. Resultados de las encuestas	111
C. Glosario	119
D. Licencia Creative Commons	131
E. Licencia Affero GPL	143

Índice de figuras

2.1. Polling	11
2.2. AJAX Multi-part Streaming	13
2.3. HTTP Long Polling	14
4.1. Arquitectura FLEQ	44
4.2. Diagrama UML	46
4.3. Arquitectura interna FLEQ app	49
4.4. Arquitectura interna FLEQbot	57
4.5. Diagrama de flujo de FLEQbot	58
4.6. Servidor web	61
4.7. Diseño esquemático de la versión de escritorio	65
4.8. Diseño esquemático de la versión móvil (I)	66
4.9. Diseño esquemático de la versión móvil (II)	68
4.10. Acceso y registro en la interfaz móvil	70
4.11. Ventana principal de la interfaz móvil	71
4.12. Gestión de torneos desde la interfaz móvil	72
4.13. Estadísticas de un torneo en la interfaz móvil	73
4.14. Configuración de partidas desde la interfaz móvil	74
4.15. Ventana de juego de la interfaz móvil	75

4.16. Página de inicio de la interfaz de escritorio	77
4.17. Ventana de juego de la interfaz de escritorio	77
4.18. Formulario para la creación de un nuevo torneo	78
5.1. Evolución del porcentaje de aciertos en el SATpeonato	84

Índice de cuadros

2.1. Soporte WebSockets	16
3.1. Soporte Socket IO	34
5.1. Evolución de la participación en el SATpeonato	83

Capítulo 1

Introducción

*Se puede tener por compañera la fantasía,
pero se debe tener como guía a la razón.*

Samuel Johnson

1.1. Motivación

Quizá uno de los sectores menos explotados en la actualidad en Internet sea la educación. Numerosos son los entornos que disfrutan de las características de la red de redes, que se ha convertido en la última década en un elemento imprescindible para la gran mayoría de la humanidad. Porcentaje que no hará sino crecer en los años venideros.

Puede que esta explotación se deba a la franja de edad de los actuales docentes, personas en su mayoría que han crecido ajenos a Internet, ya que en la juventud de muchos de ellos, lo que hoy es una de nuestras mayores ventanas al mundo, para ellos no dejaba de ser una fantasía que fluía en la cabeza de unos cuantos locos. Locos que hoy en día son considerados genios. Genios que nos han abierto un mundo de posibilidades, un mundo de alternativas, un mundo de deseos y proyectos que muchos de nosotros hemos tratado, tratamos o trataremos de llevar a cabo para lograr nuestros sueños.

Evidentemente, la actual generación web crecerá. Los sistemas docentes serán inundados por personas que, como yo, no entenderíamos este mundo sin un PC cerca. Personas que, como mi tutor del presente proyecto, no dejan de intentar que este momento se adelante lo máximo posible, e intentan incansablemente expresar todo lo que pueden sus conocimientos para mejorar labores de su día a día como docentes.

Por todo esto, creo que un buen día, nació en su cabeza el proyecto FLEQ. Estas siglas hacen referencia a lo que en sí misma es la esencia del proyecto: *Free Libresoft Educational Quizbowl*. Un proyecto orientado a la educación que hace uso de las nuevas tecnologías, como forma de hacerlo accesible a cualquier persona que tenga a mano un PC con conexión a la red.

FLEQ en sí mismo no es muy complicado. Su función es muy clara. Organiza torneos con contenido didáctico. A simple vista, no hay más. Pero detrás, observándolo detenidamente, se comprueba que no es así.

FLEQ es un sistema cuyo lema lo resume todo: *A synchronous online competition to motivate and improve learning*. Motivar y mejorar el aprendizaje. Nada es más motivante que la propia competitividad. Esa es la carta que juega FLEQ para lograr sus metas.

La competitividad, sumada a un sistema de torneos adecuado para fomentar el aprendizaje progresivo, provocará que el estudio se acentúe, con la meta de conseguir escalar posiciones en la clasificación del torneo. Esto en el fondo se traduce en un aprendizaje de los conceptos destacados en el lote de preguntas que el docente introdujo al crear el torneo, que recogerán la esencia de lo que se quiere transmitir al alumno.

En la actualidad, el uso de la red de redes desde dispositivos inalámbricos aumenta trepidantemente, como así recoge un reciente estudio de StatCounter, que desvela que el 8'5 % del tráfico generado en Internet proviene de estos dispositivos. Esta cifra, que puede parecer pequeña, se viene duplicando desde 2009 año tras año.

Esta evolución hace suponer que la tendencia será creciente en los próximos años, y más si tenemos en cuenta la llegada de las redes móviles de cuarta generación, las redes LTE. Este salto evolutivo no supondrá una disrupción tecnológica como supuso el anterior paso de GSM a UMTS (2G y 3G respectivamente), pero sin duda alguna, dará un salto sustancial en cuanto a experiencia de usuario, ya que las características de acceso que se preveen para estas redes pueden llegar a ser mayores a las de un acceso vía WiFi actualmente de una conexión con fibra óptica.

FLEQ no da soporte a usuarios móviles. No dispone de clientes móviles que cubran a este porcentaje creciente de usuarios. Si bien es cierto que existen clientes IRC disponibles para sistemas operativos como Android, la propia aplicación FLEQ no está adaptada a

la pequeña pantalla, y la experiencia de usuario actualmente hacen dificultoso su uso e inviable su implantación en entornos educativos.

Este proyecto tratará de cubrir todas estas carencias, y hacer así que FLEQ sea un sistema abierto, usable y accesible por cualquier persona, y desde cualquier medio.

1.2. Objetivos

Los objetivos de este proyecto quedan enmarcados dentro de la evolución del proyecto FLEQ, que pretende conseguir implantar esta aplicación como una potente herramienta de e-learning dentro del ámbito académico.

La responsabilidad de este proyecto es grande, porque puede marcar el futuro de FLEQ como herramienta, y la posibilidad de abrirla al público general, fuera del uso interno que se pueda hacer en la propia Universidad en la que se ha desarrollado, o el departamento que la ha promovido.

El objetivo principal no es en absoluto crear una herramienta comercializable, pero los resultados de este proyecto sí pueden marcar futuros planteamientos de expansión comercial, objetivo mucho más ambicioso, y que requerirá de un plazo más largo y de nuevos estudios y medios.

El objetivo principal de este proyecto es, por tanto, hacer accesible FLEQ desde dispositivos móviles con una experiencia de usuario excelente, que propicie que al usuario no le suponga un *handicap* usar el sistema desde su smartphone o tablet.

Esta meta requiere tener en cuenta diversas consideraciones, ya que se debe considerar que la esencia de FLEQ son las partidas entre los jugadores, y que en éstas es extremadamente importante el tiempo de respuesta de los usuarios, que determinará los aciertos en cada pregunta del juego.

Por tanto, un jugador móvil puede estar en desventaja con respecto a un jugador de escritorio, ya que la velocidad de escritura en un teclado convencional es mayor que la velocidad con la que se puede escribir en un teclado de un dispositivo móvil, por regla general.

Por todo esto, las metas de este proyecto se podrían resumir en la siguiente enumeración:

- Integrar el módulo de comunicación de FLEQ en la propia aplicación, de modo que quede desligado de sistemas terceros como podían ser los servidores IRC de los que hace uso el actual sistema.
- Desarrollar una aplicación móvil que permita a los usuarios hacer uso de la aplicación desde cualquier dispositivo inalámbrico de última generación.
- Optimizar la interfaz de usuario de la aplicación de escritorio, tanto en contenido como en formato, adaptada a las nuevas herramientas de diseño disponibles y estableciendo un flujo mucho más accesible para los usuarios.

Estos serían los objetivos principales del proyecto, que como se puede observar, pretenden dar un salto de calidad tanto en el backend como en el frontend de FLEQ.

Estas dos partes de cualquier proyecto de software deben andar de la mano: de muy poco servirá tener una arquitectura excepcional en una herramienta software, si la UI (User Interface) no está a la altura.

De igual modo que de poco servirá tener una UI que haga uso de todo el potencial que ofrece hoy en día la web, si la estructura interna de la aplicación no es todo lo fiable y/o fluida que debiera.

Se ha de buscar un equilibrio entre ambas, teniendo claro que el pilar fundamental será desarrollar una magnífica estructura interna, pero sin olvidar el lado del usuario, que al final, será el que consiga fidelizar al usuario con el entorno FLEQ.

Así, dentro de los hitos principales enumerados anteriormente, podemos distinguir una serie de objetivos secundarios sobre los que se sentarán las bases de esos objetivos generales:

- Evaluar las redes inalámbricas: contextualizar el entorno en el que va a trabajar la aplicación será clave para desestimar posibles candidatos a sustituir al actual IRC, ya que éste debe emplear el mínimo ancho de banda posible, de cara sobretodo a los clientes móviles.

- Reestructurar el modelo de datos de la aplicación: FLEQ cuenta con un modelo de datos ya definido, pero será necesario evaluarlo y estudiar qué mejoras podrían ser introducidas para evitar consultas innecesarias, reducir el uso de memoria y optimizar el tiempo de acceso a los datos.
- Arquitectura escalable: se deberá realizar un diseño modular, con el objetivo de que futuras actualizaciones, mejoras o adaptaciones sean fácilmente integrables y no requieran de una reestructuración del actual modelo de FLEQ.
- Experiencia de usuario: la aplicación ha de ser accesible por usuarios no expertos. Por tanto, se diseñará de modo que su uso sea fácil para cualquier tipo de usuario.

1.3. Estructura de la memoria

Con el fin de facilitar la lectura de este documento, se cree necesario explicar brevemente su estructura, indicando los objetivos de cada uno de los capítulos que lo componen:

1. *Introducción.* Tiene como objetivo enmarcar el contexto de este proyecto, para que el lector pueda entender qué era FLEQ antes y qué va a ser FLEQ una vez concluido el mismo y alcanzados los objetivos marcados.
2. *Estado de la Ciencia.* Presentación de las principales tecnologías que han sido utilizadas en la elaboración de este proyecto y sus principales alternativas, de modo que se pueda contextualizar el marco tecnológico en el que se sitúa y poder realizar posteriormente comparaciones objetivas sobre una base firme.
3. *Análisis de tecnologías.* En este capítulo se acometerán comparaciones entre las principales alternativas que han sido estudiadas para cada uno de los módulos que componen FLEQ, tratando de esclarecer el motivo de las decisiones tomadas con argumentos de valor y objetivos. Estas comparaciones se han centrado en los principales campos de FLEQ: sistema de comunicación para las partidas, implementación de la interfaz móvil y selección de lenguaje y framework de desarrollo.
4. *Diseño e implementación.* Este capítulo compone el núcleo del proyecto. En él se recoge la arquitectura completa de la aplicación desarrollada, tratando de dar res-

puesta a todas las preguntas que puedan surgir derivadas de la implementación de este complejo sistema.

5. *Despliegue y resultados.* Presentación de la evolución de las pruebas realizadas con la herramienta y evaluación de los resultados obtenidos. Este capítulo servirá para conocer qué acogida ha tenido FLEQ en un entorno real y qué valor añade al mundo de la educación.
6. *Conclusiones.* Capítulo final, destinado a la evaluación global del proyecto. En esta evaluación se analizará el grado alcanzado en los objetivos marcados inicialmente, se planteará un resumen de tiempo de trabajo empleado y finalmente se propondrán futuras líneas de trabajo, basadas en ideas surgidas durante el desarrollo y en comentarios obtenidos de los probadores de FLEQ.

Seguidamente, se encuentra el resumen de la bibliografía básica que ha sido consultada y un apartado de apéndices que dan soporte a esta memoria.

Estos apéndices comprenden una planificación resumida de las fases que han compuesto este Proyecto Fin de Carrera, los resultados completos de las encuestas realizadas a los participantes de las pruebas, un glosario con explicaciones detalladas de terminología empleada a lo largo de la memoria, como apoyo a la lectura, y finalmente, las licencias completas bajo las que se distribuye este documento y el propio código de la aplicación.

Esperamos que la lectura de esta memoria en la que se ha puesto tanto esfuerzo, sea del agrado del lector y pueda disfrutar con ella, tanto como se ha disfrutado escribiéndola.

Capítulo 2

Estado de la Ciencia

*Vale más saber alguna cosa de todo,
que saberlo todo de una sola cosa.*

Blaise Pascal

En este capítulo se introducirá de manera general el conjunto de técnicas, metodologías y herramientas tecnológicas que se han tenido en cuenta a la hora de realizar este proyecto.

Durante el mismo, se hará referencia a términos variados que pueden ser desconocidos para el lector. En dicho caso, se anima a completar el contenido de este capítulo en el glosario de términos que aparece al final de la presente memoria.

2.1. Aplicaciones cliente-servidor

El modelo cliente-servidor es el más extendido dentro de las aplicaciones distribuidas en una red de computadores, en el que la ejecución se distribuye entre ambos actores. La esencia de su funcionamiento es muy sencilla: el cliente realizará peticiones al servidor, que se encargará de dar respuesta a éstas.

Este modelo se plantea como un sistema centralizado, en el que toda la información reside en el servidor, lo cual facilita el diseño y la administración de la aplicación.

Por tanto, el paradigma cliente-servidor está compuesto por una serie de clientes que se conectan a un servidor, en el que se centralizan los recursos de la aplicación, y que pone estos a disposición de los clientes cada vez que estos los solicitan, en función de su rol dentro del sistema.

Todas las gestiones, por tanto, son manejadas por el servidor, que decidirá qué recursos son accesibles para cada cliente, y en función de ello, procesará la respuesta adecuada.

Existen otro tipo de arquitecturas en redes de computadores, alternativas al paradigma cliente-servidor. Entre ellas, la que más relevancia ha tenido en los últimos años ha sido la arquitectura *peer-to-peer*, en la que no existen clientes ni servidores como tales, si no que todos los componentes de la red se comportan como nodos iguales entre sí.

De este modo, actúan simultáneamente como cliente y servidor respecto al resto, permitiendo el intercambio de paquetes de forma directa.

2.1.1. Servidor

A grandes rasgos, el rol del servidor en las aplicaciones cliente-servidor o C/S ha quedado recogido en los párrafos anteriores: es el sistema encargado de controlar y almacenar la información de la aplicación, y el acceso de los clientes a ésta.

Atendiendo a su modo de procesar las peticiones de los clientes, se pueden diferenciar dos tipos:

- Servidor iterativo: las peticiones son atendidas de forma secuencial, de modo que cada una de ellas debe esperar su turno en una cola de peticiones.
- Servidor concurrente: las peticiones son atendidas en paralelo, mediante la creación de procesos o hilos.

Los servidores iterativos son adecuados para aplicaciones C/S en los que los clientes no generen peticiones demasiado pesadas, tales como manejo de archivos. Al ser mono-proceso, una petición de este tipo podría hacer demasiado larga la espera de respuesta de otros clientes que hayan tratado de comunicarse con el servidor después.

Del mismo modo, un sistema que cuente con un gran tráfico de peticiones, es inviable con esta configuración, ya que las esperas serían igualmente largas, como en la casuística anterior.

Sin embargo, este tipo de servidor es mucho más fácil de implementar que el servidor concurrente, y puede prestar un buen rendimiento en aplicaciones con poco tráfico y con

reducidas operaciones de I/O de archivos.

Los servidores concurrentes resuelven todos los problemas que podían generarse del uso de un servidor iterativo. Cada petición genera un hilo de ejecución en el servidor, por lo que la petición es procesada en ese nuevo hilo, y el servidor puede atender a futuras conexiones de clientes aunque la petición del cliente anterior todavía esté en curso.

Evidentemente, este tipo de sistema requiere de una implementación más costosa, ya que requiere de la creación de procesos, a los que habrá que proteger de problemas derivados de la concurrencia.

Este tipo de servidores se caracteriza por tanto, por su velocidad de respuesta y su capacidad de atender a múltiples clientes de forma simultánea, lo que los convierte en el tipo predominante en las aplicaciones C/S de la actualidad, en especial en las encapsuladas dentro del paradigma de Real Time Web, como es el caso de este proyecto.

2.1.2. Cliente

El cliente es el remitente de las solicitudes que llegan al servidor, por lo que su papel dentro de la arquitectura C/S es activo, y el punto de acceso de los usuarios a la aplicación.

Este término comenzó a ser utilizado ante terminales que únicamente contaban con conexión a una red de telecomunicación, la cual empleaban para comunicarse con un equipo remoto.

Estos dispositivos no disponían de ningún mecanismo de computación, por lo que todas las operaciones se realizaban en este otro equipo, que comunicaba los resultados, y se mostraban al usuario.

El navegador web es el cliente por excelencia. A través de él, se puede acceder a multitud de servicios que proporcionan contenido variado a los usuarios.

2.2. Real Time Web

En sistemas web de tiempo real, como servicios de mensajería instantánea, juegos online multijugador o cualquier aplicación que pueda necesitar establecer comunicaciones

asíncronas entre servidor y cliente, podemos diferenciar dos tipos de tecnologías atendiendo al origen del establecimiento de las comunicaciones.

De este modo, se denomina tecnología *pull* aquella en la que el cliente solicita periódicamente actualizaciones de estado al servidor, y tecnología *push*, en la que el servidor es capaz de establecer conexiones con el cliente ante la generación de nuevos eventos, sobre un canal establecido al inicio de la comunicación entre ambos lados.

Esta diferenciación puede también catalogarse como conexiones no persistentes (*pull*) y persistentes (*push*).

2.2.1. Tecnologías PULL

Las tecnologías *pull* son aquellas en las que el origen de las comunicaciones es establecido por el cliente.

Este tipo de transacciones supone que el cliente debe preguntar periódicamente al servidor para mantener su estado actualizado. En aplicaciones de tiempo real, como pueda ser un sistema de mensajería, el slot temporal entre cada *refresh* del cliente debe ser reducido, no superior al segundo, para conseguir un sistema en tiempo real.

Entre este tipo de tecnologías se encuentra el famoso *polling*, y otras variantes que de éste se han derivado, optimizaciones que reducen el consumo de ancho de banda de las aplicaciones, como trata de conseguir el denominado *long polling*.

HTTP Polling

La técnica de *polling* fue la primera en ser utilizada para conseguir comunicaciones en tiempo real en aplicaciones cliente-servidor.

Como se avanzó anteriormente, en este modelo el cliente inicia conexiones de forma periódica con el servidor, solicitando actualización de estado. Si en ese intervalo de tiempo que va desde una petición hasta la siguiente, ha ocurrido algún evento en el servidor, éste se lo notificará, lo que supone que el cliente puede recibir estas actualizaciones con un retraso máximo igual al *delay* establecido entre peticiones.

Para evitar que este *delay* pueda llegar a ser muy elevado, el intervalo de actualización

del cliente debe ser corto. Esto propicia que la mayor parte de las solicitudes que se realizan sean innecesarias, ya que el servidor no tiene nada que notificar al cliente, por lo que se produce un consumo de ancho de banda que podría ser evitado.

JSONP Polling

JSONP polling es básicamente el mismo concepto que HTTP polling. La diferencia entre ambos radica en que mediante esta técnica se puede establecer peticiones *cross-domain*, es decir, peticiones fuera del dominio de la página desde la que se genera.

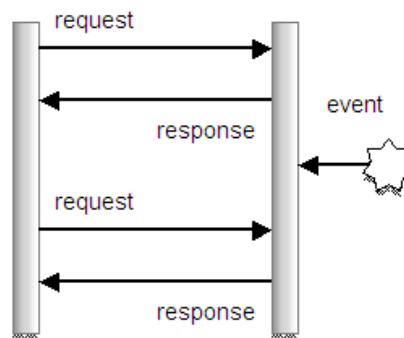


Figura 2.1: Cliente (izqda) realizando polling contra el servidor (dcha)

2.2.2. Tecnologías PUSH

En el lado opuesto a las técnicas *pull*, encontramos las tecnologías *push*, aquellas mediante las cuales el servidor es capaz de comunicarse con el cliente de forma asíncrona, porque éste previamente ha establecido una conexión persistente entre ambos.

Ese es el punto clave de diferencia entre ambas técnicas. Las tecnologías *push* se basan en el uso de canales de comunicación persistentes, es decir, las respuestas que facilita el servidor son parciales, no terminan la comunicación como sí ocurre en las técnicas *pull*. De este modo, todo el flujo de información durante la sesión del cliente, se realiza habiéndose producido una única petición desde este lado.

Esto supone grandes beneficios para el funcionamiento de las aplicaciones en tiempo real: minimiza consumo de ancho de banda del cliente, fundamental en conexiones desde dispositivos móviles, y reduce al mínimo el *delay* en la recepción de las actualizaciones.

Las tecnologías *push* son implementadas mediante diferentes técnicas, la mayoría de las cuales han quedado bajo el paraguas del término Comet, con el que se las identifica. Otras más novedosas, han nacido en el hábitat de HTML5, como es el caso de los WebSockets.

COMET

El término Comet es un neologismo empleado para definir un modelo de aplicaciones web que permiten que el servidor sea capaz de enviar asíncronamente información al cliente, sin la petición explícita de éste.

Este término engloba múltiples técnicas y supone una ruptura con el modelo original de la web, en el cual el servidor era el elemento reactivo de la dupla cliente-servidor, mientras que el cliente era la parte proactiva.

El conjunto de tecnologías *push* que engloba Comet también es referido por otros términos como Ajax Push, Reverse Ajax, HTTP Streaming o HTTP server push.

Forever iframe

Esta fue una de las primeras técnicas en ser utilizadas para conseguir comunicación persistente entre cliente y servidor. Se basa en la inclusión de un *iframe* dentro de la página web.

Este *iframe* carga otra página web sin contenido inicial. Por tanto, es invisible desde el recurso original. A medida que se genera contenido en el servidor, éste lo incluye en el recurso cargado por el *iframe* entre etiquetas *script*, de modo que el navegador pueda interpretarlas y actualizar el contenido con la nueva información recibida.

Entre las ventajas de esta técnica, destaca su simplicidad de implementación, y su cobertura en la práctica totalidad de los navegadores del mercado. Entre sus desventajas, cabe mencionar que mediante este modelo de comunicación, no hay manera de implementar control de errores o conocer el estado de la conexión, ya que toda la comunicación es manejada mediante etiquetas HTML.

AJAX Multi-part Streaming

Esta técnica es más fiable que la basada en *iframes*. Se basa en el uso del flag multi-part en el objeto XHR.

El cliente envía una petición al servidor, que se mantiene abierta. Cada vez que se produce un evento, el servidor genera una respuesta *multi-part* y es enviada a través de la conexión previamente establecida.

Entre las ventajas de esta técnica, cabe destacar que esta es la técnica Comet que mejor uso hace del ancho de banda. Entre sus desventajas, no todos los navegadores soportan el flag de multi-part para objetos XHR. Además, genera problemas en relación al buffering, ya que elementos extensos pueden ser fraccionados para su envío, lo que puede generar cierta latencia ya que el envío no se realiza hasta que el buffer está completo.

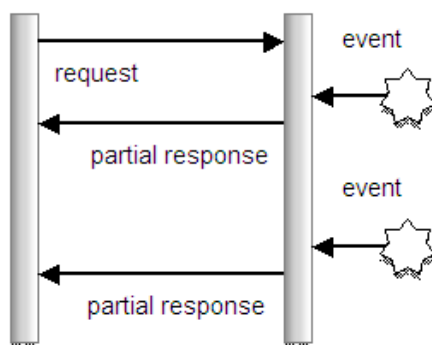


Figura 2.2: Conexión persistente entre cliente (izqda) y servidor (dcha)

HTTP Long Polling

La técnica de *long polling* es una optimización del *polling* tradicional que trata de minimizar el consumo de ancho de banda en aplicaciones cliente-servidor, simulando una conexión persistente.

La diferencia respecto al *polling* tradicional reside en que el servidor puede retrasar su respuesta si no dispone de ninguna información para el cliente, hasta que se produzca algún evento, o hasta que se alcance un *timeout*, establecido por el propio cliente. Esto lo convierte en una tecnología *push*, ya que el servidor decide cuando generar la respuesta, en función de las necesidades de cada instante.

Cuando el cliente recibe la respuesta del servidor, relanza la petición, de modo que siempre hay una conexión establecida entre ambos actores de la comunicación.

Con *long polling*, se reduce considerablemente el consumo de ancho de banda respecto al *polling* tradicional, así como el retraso que sufre el cliente en recibir nuevos eventos.

Esta técnica puede ser implementada de dos diferentes formas:

- Script tags: similar al modelo de *iframes*, la respuesta del servidor consiste en etiquetas *script*, cuyo contenido es ejecutado por el navegador.
- XHR: utiliza objetos XHR en las comunicaciones cliente-servidor.

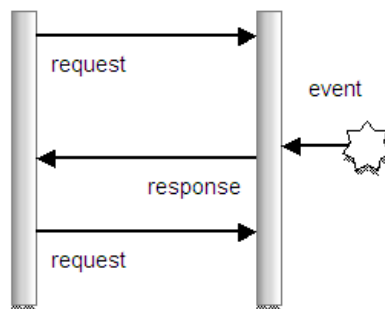


Figura 2.3: Cliente (izqda) realizando long polling contra el servidor (dcha)

XMPP

El protocolo XMPP (eXtensible Messaging and Presence Protocol, anteriormente denominado Jabber) es un estándar abierto y extensible basado en XML, diseñado para mensajería instantánea.

Este protocolo surgió como alternativa a otros protocolos propietarios que utilizaban plataformas como el famoso Windows Live Messenger. Al ser un protocolo abierto, existen diversos clientes y servidores libres que dan soporte a este sistema de mensajería instantánea.

La primera liberación de software se produjo en mayo del año 2000. Es utilizado por los grandes sistemas de mensajería de la actualidad, como Google Talk o WhatsApp.

Entre las ventajas de este protocolo destacan su descentralización, está basado en estándares abiertos, su robusta seguridad (SASL y TLS), y su flexibilidad.

Como cualquier tecnología, cuenta con desventajas. Una de las principales es la sobrecarga de la red producida por los datos de presencia de los usuarios, lo que supone una media del 70 % del tráfico derivado de su uso (además de ser información redundante en la mayoría de los casos).

Este problema está enfrentando en la actualidad a las grandes compañías de telefonía móvil como AT&T, Vodafone o Telefónica entre otros, contra las compañías que generan este tráfico en sus servicios (principalmente Google y WhatsApp), ya que el aumento del número de dispositivos móviles que hacen uso masivo de estos servicios, está produciendo saturaciones en las redes de datos, lo que conlleva que sea necesario invertir en infraestructuras para garantizar calidades de servicio (QoS) adecuadas a todos los usuarios.

Para realizar estos despliegues, solicitan la aportación económica de estas compañías que, como Google o Facebook, hacen uso de la red móvil sin ningún coste.

Otros puntos débiles del protocolo se derivan de su escasa escalabilidad, y de la imposibilidad de realizar transferencias de información binarias. Como se indicó anteriormente, XMPP está basado en XML, por lo que las transferencias se realizan en este formato, aumentando el tamaño del paquete a transmitir, lo que supone que, lógicamente, las comunicaciones presenten *delays* superiores a las comunicaciones con información en crudo.

El protocolo XMPP cuenta con una organización encargada de su estandarización, la XSF (XMPP Standards Foundation), anteriormente llamada JSF (Jabber Software Foundation).

WebSockets

La tecnología WebSocket ha sido introducida como una de las nuevas especificaciones de HTML5, definiendo conexiones full-dúplex sobre un único socket TCP sobre el cual, tanto el cliente como el servidor pueden realizar envíos de información de forma totalmente asíncrona.

Esta nueva implementación representa un salto evolutivo tanto para las obsoletas técnicas *pull* como para las optimizadas técnicas Comet, en los sistemas web de comunicación

	IE	Firefox	Chrome	Safari	Opera	iOS	Android	BB
Actual	9.0	16.0	23.0	6.0	12.1	6.0	4.1	7.0
Futura	10.0	17.0	24.0		12.5			10.0

Cuadro 2.1: Soporte de WebSocket en los navegadores actuales (verde: si, rojo: no)

en tiempo real. Con Websockets, se reduce enormemente el tráfico de red innecesario y la latencia de las comunicaciones. Esto supone grandes beneficios en el lado del servidor, que verá aliviadas sus tareas ya que tendrá que atender menos peticiones, de menor carga y evitará las innecesarias.

Se basa en el lenguaje de scripting JavaScript, y es soportado por la gran mayoría de navegadores actuales, tanto móviles como de escritorio.

Una de las principales características que incorpora esta nueva especificación es la posibilidad de atravesar firewalls y proxies, lo cual supone un problema para otras tecnologías, que establecen sus conexiones en puertos diferentes al 80, que pueden estar bloqueados por motivos de seguridad.

En el caso de técnicas Comet, se debe hacer uso de long-polling para evitar este tipo de problema. Esta técnica es una solución válida, pero indudablemente, produce un considerable aumento del tráfico de red, además de un incremento de la latencia.

Con esta nueva especificación, esto deja de ser un problema, ya que cuando un WebSocket detecta un proxy, establece automáticamente un túnel para sortearlo.

2.3. Tecnologías web

Las nuevas tecnologías de desarrollo web facilitan la implementación de aplicaciones, así como optimizan el uso de recursos, tanto locales como de la propia red de comunicaciones.

En los últimos años, la llegada de las nuevas versiones de HTML y CSS han incrementado notoriamente la calidad visual de los sitios web, y no sólo eso.

HTML5, la nueva especificación de HTML, pretende revolucionar el concepto que hasta ahora se tenía de la web estándar, proporcionando APIs tales como los mencionados

WebSockets u otras como la API de geoposicionamiento, presentando a este lenguaje como una seria alternativa a las aplicaciones móviles nativas, tan extendidas actualmente.

Todo esto, unido a la evolución de las técnicas AJAX, la aparición y desarrollo de librerías de JavaScript como jQuery, y los numerosos frameworks existentes para facilitar el diseño de interfaces, se está traduciendo en un aumento considerable de la calidad de los sitios web actuales.

Estas nuevas tecnologías que se introducirán a continuación, presentan un gran salto evolutivo para el desarrollo web, orientado tanto a optimizar el uso de los recursos disponibles ante una masa de usuarios cada vez mayor, como a la mejora sustancial de los contenidos ofrecidos, traduciéndose todo ello en un crecimiento palpable de la experiencia de usuario.

2.3.1. HTML5

HTML5 es la quinta versión del lenguaje de marcado HTML de la World Wide Web, sobre el que se basan todas las páginas web.

Esta nueva especificación define dos variantes de sintaxis: HTML (clásica) y XHTML (basada en XML), que por primera vez están siendo desarrollados en paralelo.

Aunque todavía no es un estándar definitivo y se encuentra en modo experimental, la gran mayoría de navegadores actuales están incluyendo progresivamente las nuevas características de este lenguaje, debido al gran abanico de posibilidades que incorpora.

En cuanto al propio lenguaje, se han introducido numerosos cambios, entre los que destacan los siguientes:

- Nuevas etiquetas: se han incorporado etiquetas que responden a la demanda actual de la web, tales como *footer*, *nav*, *audio*, *vídeo*, etc.
- Mejoras en el elemento *canvas*: permite la proyección de objetos en 3D.
- Mejoras en formularios: inclusión de elementos para introducir email, número de teléfono, URL, calendario para seleccionar fechas, y validación del contenido por medio de JavaScript.

- Visores para representar fórmulas matemáticas, vectores o gráficas en 2D y 3D.
- Eliminación de etiquetas obsoletas: tags como *font* y *center*, cuya funcionalidad es ampliamente cubierta en las hojas de estilo, han sido suprimidas.
- Soporte para CSS3.

Pero sin duda alguna, sus puntos fuertes vienen por las APIs que introduce:

- Drag & Drop: permite arrastrar y soltar ficheros directamente desde un directorio local al navegador.
- Local & Global Storage: almacenamiento persistente en local mediante bases de datos SQLite o almacenamiento de objetos por aplicación (global).
- Offline mode: permite la descarga de recursos en modo local para trabajar desconectado posteriormente.
- WebSockets: comunicación bidireccional entre cliente-servidor mediante un socket TCP.
- WebWorkers: *threads* para la realización de tareas en paralelo.

Todas estas funcionalidades dejan patente que el abanico de posibilidades que se abre con la llegada de HTML5 es inmenso, y APIs como la de geoposicionamiento dejan claro que a través del navegador se puede acceder a hardware de bajo nivel, independientemente de cual sea el dispositivo en el que se esté trabajando.

Sin duda, esto supone un punto de inflexión en el ámbito del desarrollo de aplicaciones para dispositivos móviles, debate que será abordado en la siguiente sección.

2.3.2. CSS3

Las hojas de estilo (Cascading Style Sheets) surgieron con el propósito de separar el contenido de la forma en las páginas web, y conseguir así que su código fuera fácilmente interpretable. Esto se consiguió desde las primeras versiones de CSS.

El deseo inicial pretende que estas propiedades de formato sean recogidas en un documento aparte del documento HTML, aunque también puede ser incluido en éste. El objetivo fue alcanzado desde las primeras versiones de CSS, sin embargo, el objetivo de ofrecer un control absoluto sobre los elementos ha sido mucho más costoso.

Estas primeras versiones no cubrían aspectos que los diseñadores web pretendían cubrir, y para conseguirlos se utilizaban argucias que suponían un gran esfuerzo y resultados no estandarizados a los distintos navegadores.

En muchas ocasiones, para conseguir estos efectos se alteraba el contenido del HTML, con lo que se estaba generando un efecto contrario al que se pretendía conseguir inicialmente.

CSS3 surge para contrarrestar estos efectos negativos generados por el avance de la calidad en el diseño web. Incorpora nuevos mecanismos para mantener un mayor control sobre el estilo de los elementos, sin tener que recurrir a complicadas extensiones de código.

Entre las principales novedades que introdujo CSS3 destaca el soporte proporcionado a los siguientes elementos:

- Bordes: radio, sombras, imágenes y color.
- Fondos: capas con múltiples imágenes.
- Color: HSL, HSLA, RGBA y ajuste de opacidad.
- Texto: sombras, separación de palabras, web fonts, etc.
- Degradados: lineales y radiales con y sin repetición.
- Ajustes de interfaz, selectores por atributos.
- Layout multicolumna.
- Animaciones.
- Transiciones.
- Media Queries.

De entre todas estas características, especial mención requiere este último módulo, CSS3 Media Queries, en el contexto del desarrollo web para dispositivos móviles, que proporciona la funcionalidad necesaria para establecer qué tipo de pantalla está siendo utilizada para visualizar un documento web determinado, en lo que se ha denominado *responsive design*.

2.3.3. JavaScript

Lenguaje de programación orientado a objetos, interpretado, basado en prototipos, débilmente tipado y dinámico, que fue estandarizado como ECMAScript.

Su desarrollo fue iniciado por Netscape en la primera mitad de los años 90. Su diseño se basa en una sintaxis similar al lenguaje C, aunque adopta parte de la funcionalidad del lenguaje Java. En la actualidad, todos los navegadores web tienen capacidad de interpretar código JavaScript, el cual es utilizado para acceder y modificar los objetos del árbol DOM del documento HTML.

Su uso es masivo en aplicaciones web en la actualidad, en el lado del cliente, aunque también puede ser utilizado en el lado del servidor, como es el caso del moderno NodeJS, servidor orientado a eventos basado en este lenguaje, cuyo uso se está extendiendo en la actualidad como una de las alternativas para aplicaciones web de tiempo real.

También es utilizado en aplicaciones externas al mundo web, como en documentos PDF, para la generación de hiperenlaces internos, o en aplicaciones de escritorio, como widgets.

Pero sin duda alguna, JavaScript debe su éxito a la técnica AJAX (Asynchronous JavaScript And XML), mediante la cual se pueden establecer conexiones HTTP de forma asíncrona haciendo uso de este lenguaje, con lo que se puede actualizar el contenido de las páginas web de forma dinámica sin necesidad de recargar el recurso al completo.

Este método ha supuesto toda una revolución en el modelo web que existía anteriormente, llegando a convertirse en el factor común de las aplicaciones web. Existen múltiples variantes de posibilidades de implementación del lado del servidor, pero el lado del cliente está monopolizado por este lenguaje, posición que se ha visto reforzada con la llegada del HTML5, que basa todas sus APIs en el uso de este lenguaje.

Las aplicaciones web en tiempo real, como la que nos ocupa en este proyecto, así como todas las técnicas explicadas en la sección anterior, hacen uso de este lenguaje, por lo que se puede decir que JavaScript es el principal actor que hace posible el concepto de Real Time Web.

2.3.4. jQuery

Biblioteca de JavaScript creada con el objetivo de simplificar la sintaxis de este lenguaje, y facilitar así la manipulación del árbol DOM de los documentos HTML, la interacción con el servidor o el manejo de eventos, además de proporcionar técnicas para modificar la interfaz de usuario de forma sencilla.

Esta librería apareció en el año 2006, y desde ese momento no ha dejado de crecer. Al tratarse de software libre, la evolución de su desarrollo es muy rápida, habiéndose producido numerosas actualizaciones desde su aparición, que además de resolver errores, han servido para extender sus funcionalidades.

Tal ha sido el éxito de esta biblioteca de JavaScript, que grandes empresas del sector tecnológico como Nokia o Microsoft la han incluido en sus plataformas.

La biblioteca consiste en un único fichero JavaScript, en el que quedan recogidos una gran variedad de métodos que permiten conseguir las funcionalidades indicadas anteriormente.

A raíz de esta librería, han sido desarrollados frameworks de desarrollo de interfaces que facilitan la implementación del front-end de aplicaciones web, tanto para equipos de escritorio (jQueryUI) como para dispositivos móviles (jQueryMobile).

jQueryMobile

Este framework, que se basa tanto en jQuery como en jQueryUI, está orientado exclusivamente al desarrollo de interfaces de usuario de sitios web adaptados a dispositivos móviles. Destaca por su gran utilidad, así como su facilidad de uso.

Es soportado por un gran porcentaje de dispositivos móviles actuales, cubriendo tanto los más populares como iOS, Android, Windows Phone o BlackBerry como los que menos

cuota de mercado poseen, tales como Palm WebOS, MeeGO o Symbian. Esto supone que sea una de las herramientas que está empujando con fuerza en el desarrollo de aplicaciones móviles multiplataforma en detrimento de las aplicaciones nativas.

Al ser una librería orientada al uso en webs móviles (aunque también puede ser utilizada en aplicaciones de escritorio), ha sido optimizada para que su peso sea mínimo, consiguiendo ser empaquetada en un fichero de unos 20 kB.

En la misma línea de optimización de recursos para reducir tiempos de carga de recursos web, no hace uso de imágenes de forma predeterminada, aunque obviamente, pueden ser incluidas si así se desea.

Hace uso del marcado de HTML5 para estructurar las páginas y elementos de éstas, aprovechando al máximo la funcionalidad de éste, para lanzar widgets o aplicar modificaciones sin necesidad de código adicional.

Sus componentes son fácilmente accesibles, ya que han sido desarrollados basándose en la especificación WAI-ARIA y hacen uso de los *roles* de HTML5. Aunque el framework proporciona varias temáticas por defecto, se proporciona una potente herramienta para crear, modificar y personalizar el *look and feel* de la web, llamado ThemeRoller.

Otras de las posibilidades que da jQueryMobile es su adaptación a herramientas de conversión como PhoneGap, lo que permitirá hacer uso de APIs nativas de los dispositivos móviles, como el GPS, acelerómetro, notificaciones, etc. consiguiendo aplicaciones de calidad para una amplia gama de dispositivos en mucho menos tiempo que si se desarrollaran de forma nativa.

2.4. Python, Django y Tornado

2.4.1. El lenguaje Python

El lenguaje Python fue desarrollado a finales de los años 80 por el holandés Guido Van Rossum como sucesor del lenguaje ABC. Fue bautizado con este nombre en homenaje a los humoristas británicos *Monty Python*, de los que Van Rossum era aficionado.

Python es un lenguaje multiparadigma: implementa orientación a objetos, programa-

ción imperativa y ciertos aspectos de programación funcional y reflexiva. Esto lo convierte en un lenguaje muy versátil, ya que puede ser utilizado en diferentes áreas o campos obteniendo resultados óptimos. Adicionalmente, puede soportar paradigmas alternativos mediante el uso de extensiones.

Además, es un lenguaje interpretado, ya que es ejecutado por medio de un intérprete, lo que conlleva que sea más flexible que los lenguajes compilados, en detrimento de que la ejecución sea menos eficiente que en estos.

Su tipado dinámico le permite que una misma variable pueda tomar distintos valores a lo largo del flujo de ejecución del programa, ya que la tipificación de los tipos de datos se realiza en tiempo de ejecución, a diferencia de otros lenguajes como C o Java que realizan este proceso en tiempo de compilación.

Por último, Python es un lenguaje multiplataforma, lo que permite que sea empleado indistintamente en todos los sistemas operativos. Es distribuido bajo una licencia de código abierto, Python Software Foundation License (sucesora de la Python License), la cual es compatible con la licencia GPL.

Alrededor de este lenguaje, y gracias a su carácter modular, se ha creado una comunidad de cooperación que ha permitido que se desarrollen extensiones de Python que cubren multitud de áreas, lo que permite que sea óptimo para resolver multitud de problemas. Entre los módulos más destacados se encuentran:

- Bases de datos: MySQLdb, PyGreSQL, pySQLite.
- Interfaz gráfica: pyGtk, wsPython.
- Imágenes: Python Image Library (PIL).
- Numéricos: NumPy, matplotlib.
- Juegos: PyGame.
- Mensajería: jabberpy.
- Web: pythonweb, mechanize, Selenium.
- Puertos: pySerial, pyUSB, pyParallel.

- Integración: libgmail, pyGoogle, pyWordNet.
- Sonido: pySoundic, pyMedia.
- ...

Estos paquetes se encuentran disponibles en PyPI (Python Package Index), repositorio oficial de Python para módulos de terceros escritos en código abierto. Para acceder a este repositorio fácilmente desde la terminal se dispone de *pip*, herramienta que permite instalar y administrar los módulos Python disponibles en PyPI, que apareció como sustituto del anterior *easy install*.

Como se ha mencionado, Python ha creado una gran comunidad de desarrollo a su alrededor, lo que permite que el lenguaje ofrezca respuesta a multitud de situaciones que se puedan plantear al desarrollador. Su sitio web deja constancia de esto, con un portal para la comunidad Python, wikis, foros, documentación, repositorios, etc.

En resumen, Python es flexible y abierto, lo que ayuda a su rápida y constante evolución, convirtiéndolo en un lenguaje de programación adecuado para cubrir ampliamente los requisitos que pueda necesitar una aplicación web, ya que además cuenta con un conjunto muy variado y potente de *web frameworks*, entre los que destacan Django y Tornado.

2.4.2. Django

Django es un web framework de código abierto escrito en el lenguaje Python, desarrollado en su origen para gestionar páginas de noticias de la World Company Lawrence (Kansas, EEUU). Fue publicado en 2005 bajo una licencia BSD.

Sus orígenes, en los que daba soporte a sitios web de noticias, son evidentes ya que proporciona herramientas que facilitan el desarrollo de páginas orientadas a contenidos.

Este *framework* de alto nivel facilita al máximo la tarea al desarrollador puesto que proporciona un nivel alto de abstracción que le permite despreocuparse de la configuración de protocolos o hilos de ejecución.

Django sigue las premisas marcadas por el acrónimo DRY, *Don't repeat yourself*, por

lo que se pone énfasis en la reutilización máxima de sus componentes.

Su arquitectura está basada principalmente en el paradigma MVC (Modelo Vista Controlador), que adaptada a sus características lo convierten en realidad en un MPV (Modelo Plantilla Vista).

Dispone de un mecanismo de generación de plantillas que permite generar y extender fácilmente los documentos HTML que son presentados al usuario, permitiendo modificar su contenido fácilmente con sentencias de pseudocódigo integradas en los propios documentos durante el procesado de las peticiones.

Django soporta diversos sistemas de bases de datos para el manejo y almacenamiento de la información, siendo PostgreSQL el recomendado. También destacan el empleo de MySQL y SQLite.

El *framework* integra un servidor web ligero que es utilizado para acciones de desarrollo. A la hora de desplegar la aplicación web en un entorno de producción, se recomienda el uso de Apache, aunque el soporte que Django tiene de la especificación WSGI permite que puedan ser utilizados una gran variedad de servidores web.

Entre las características principales de Django se encuentran:

- Mapeador relacional de objetos: definición de objetos en lenguaje Python y acceso dinámico a los mismos mediante APIs que permiten interactuar con bases de datos PostgreSQL, SQL o SQLite.
- Interfaz de administración: gestión de usuarios, páginas y bases de datos.
- Sistema de plantillas: proporciona un lenguaje de plantillas que permite separar el diseño, el contenido y el código de la aplicación.
- Seguridad (XSS, SQL Injection, etc.).
- Sistema de caché.
- Formato libre de URLs.
- Internacionalización: soporte multilinguaje.

Diversos sitios web hacen uso de Django como framework de desarrollo web, entre los que destacan las redes sociales *Instagram* y *Pinterest*, o el sistema de comentarios *Disqus*.

2.4.3. Tornado

Tornado está compuesto de un servidor web no bloqueante de código abierto y un conjunto de módulos escritos en Python que fueron desarrollados para dar vida al proyecto FriendFeed, una aplicación web en tiempo real que permite ver publicaciones de redes sociales, medios, blogs, etc. en un único sitio.

FriendFeed fue adquirida por Facebook en el año 2009, tras lo cual se liberó el código de Tornado bajo Licencia Apache.

Este framework destacada por ser no-bloqueante y extremadamente rápido. Al ser no-bloqueante, puede manejar miles de conexiones simultáneamente lo que le hacen muy adecuado para aplicaciones web de tiempo real, objetivo, por otra parte, para el que fue creado inicialmente.

Capítulo 3

Análisis de tecnologías

*Los problemas son oportunidades
para demostrar lo que se sabe.*

Duke Ellington

Una vez realizada la introducción de las diferentes tecnologías que han sido utilizadas en el desarrollo de este proyecto, en este capítulo se va a abordar un análisis comparativo en los diferentes campos, para esclarecer qué motivos han llevado a la elección de unas u otras en la implementación final.

Cabe decir que estos análisis se van a centrar en los ámbitos del proyecto que han sido susceptibles de utilizar diferentes alternativas, ya que, por ejemplo, el proyecto deja patente desde el principio que se trata de una aplicación cliente-servidor, por lo que no será necesario entrar a explicar por qué no se ha optado por una aplicación *peer-to-peer*.

De este modo, el primer análisis que se va a llevar a cabo tratará de argumentar la decisión tomada en cuanto a la tecnología que se ha utilizado como mecanismo de comunicación en las partidas de FLEQ.

Al igual que en el capítulo anterior, se anima al lector a acudir al glosario final para profundizar en ciertos términos que puedan ser empleados durante el desarrollo de esta sección.

3.1. Comparativa de tecnologías del sistema de comunicación

En esta sección se va a realizar un estudio para determinar qué tecnologías serán las más adecuadas para el sistema de juego de FLEQ.

Se comenzará con un estudio de las tecnologías *pull* y *push*, de forma general, para continuar por sus diferentes ramas en función de la que resulte más beneficiosa.

3.1.1. PUSH vs PULL

Como se explicó en el capítulo anterior, las tecnologías *pull*, como el tradicional método de polling, son desde el punto de vista de desarrollo, mucho más fáciles de implementar que cualquier técnica *push*. Este es el único punto a favor de estas técnicas.

Su mecanismo es sencillo: el cliente implementa un bucle infinito que se ejecuta periódicamente en función de un intervalo determinado. Este mecanismo provoca efectos negativos como un consumo innecesario de ancho de banda, ya que la mayor parte de las peticiones que se realizan no devuelven ninguna información porque no se ha producido ningún evento, y un retraso que puede ser tan grande como lo sea el propio intervalo de ejecución del bucle.

Sin embargo, las técnicas *push* evitan estos problemas, ya que el servidor es proactivo en vez de ser reactivo, y puede enviar información a los clientes de forma asíncrona, por lo que estos no deben estar realizando peticiones periódicas para actualizar su estado.

De este modo, el ancho de banda consumido por los clientes es mucho menor y estrictamente el necesario, y el retraso en la recepción de los nuevos eventos es únicamente debido al que presente la red de comunicaciones que esté siendo empleada por el cliente.

Por todo esto, parece obvio que la aplicación debe orientarse hacia técnicas *push*, ya que el empleo de polling puede hacer inutilizable el sistema desde dispositivos móviles.

Una técnica *pull* desde un terminal móvil puede hacer que la aplicación no cumpla con los requisitos buscados. Un intervalo de ejecución del bucle demasiado corto, rápidamente colapsará la red, si se está empleando red de telefonía, y un intervalo más extenso, hará

que la percepción de sistema de tiempo real quede muy perjudicada.

En un equipo de escritorio, esta solución no tendría estos problemas (los achacables a terminales móviles), pero sí los primeros. Además, está lejos de ser una solución óptima, y que debe ser evitada en la medida de lo posible, más si se tiene en cuenta las diversas opciones existentes en la actualidad.

3.1.2. Comparativa técnicas PUSH

Una vez se ha argumentado los beneficios del uso de tecnologías *push* respecto a las *pull*, se va a proceder a analizar las diferentes técnicas *push* descritas en el capítulo anterior.

En este apartado, se ha diferenciado entre técnicas englobadas bajo el término Comet, el protocolo XMPP y los WebSockets de HTML5.

Las técnicas Comet (iframes, streaming y long polling) son a priori, las que más facilidades presentan desde el punto de vista de la implementación. Sin embargo, estas técnicas pueden encontrar problemas derivados del soporte en ciertos navegadores. Por ejemplo, la técnica de *iframes* requiere que el navegador soporte la etiqueta correspondiente, la técnica de *streaming* requiere el soporte del flag multi-part para objetos XHR y puede provocar incrementos en la latencia debido al buffering.

Además, con *iframes* no es posible conocer el estado de las conexiones, por lo que una caída del sistema de juego podría traer efectos muy negativos para los usuarios, puesto que el sistema no sería capaz de restablecer la conexión por sí mismo.

Por su parte, HTTP Long Polling, sólo necesita que el navegador soporte JavaScript, lo cual no es un impedimento, pero no deja de ser una técnica que bien pudiera ser considerada híbrida entre *push* y *pull*, ya que mantiene un canal de comunicación persistente durante un intervalo determinado de tiempo, pero no indefinido, por lo que no es la solución ideal si se trata de reducir el ancho de banda consumido, tan a tener en cuenta en el campo de dispositivos móviles.

El protocolo XMPP sería el indicado a priori para este sistema, ya que es un protocolo de mensajería instantánea. Sin embargo, presenta inconvenientes que pueden provocar

que no sea una buena alternativa.

XMPP presenta un sistema de control de presencia de los usuarios, que satura la red con mensajes relativos al estado de su conexión. FLEQ es un sistema de juego en el que la presencia de los usuarios no es estrictamente relevante.

El usuario no necesita interactuar directamente con su adversario ni conocer su estado para poder jugar, sino que necesita hacerlo con FLEQbot, el robot encargado de gestionar el juego, que se conecta a horas establecidas e introduce mensajes constantemente desde el momento en que entra en una *gameroom* indicando el tiempo restante para el comienzo del juego.

El mayor intercambio de mensajes supone un incremento de transacciones de los usuarios, que indican periódicamente su estado. Teniendo en cuenta que el objetivo es conseguir que la experiencia de usuario de FLEQ desde dispositivos móviles sea lo mejor posible, XMPP puede perjudicar a ello, ya que un mayor flujo de datos puede ocasionar mayores retardos en la transmisión de mensajes, provocando una percepción negativa que el usuario pueda achacar al uso de FLEQ desde un terminal móvil.

Por otro lado, la implementación de este protocolo requiere de un servidor especial. La implementación de estos servidores es laboriosa, mucho más de lo que pueda ser la configuración de cualquier servidor web estándar tipo Apache o Nginx.

Por último, los WebSockets de HTML5 presentan al igual que el resto, ventajas y desventajas. Su principal desventaja viene derivada de su juventud, lo que provoca que aún no sea soportado por todos los navegadores.

Si bien es cierto que la gran mayoría de nuevas versiones de los navegadores están incorporando esta tecnología, hay que tener en cuenta que el grueso de usuarios utiliza versiones anteriores a las más recientes, y entre sus prioridades, no suele tener cabida el actualizar el navegador de su PC o móvil, salvo que esta tarea se realice de forma automática.

La tecnología de WebSocket ha sido desarrollada específicamente para realizar comunicaciones bidireccionales, mientras que las técnicas basadas en AJAX que permiten esta funcionalidad (parte de las técnicas Comet), son simples "parches" que tratan de simular este tipo de conexión para el que inicialmente no fueron diseñadas.

Mediante WebSockets, las transacciones entre cliente y servidor son las más ligeras y rápidas de todas las técnicas analizadas. Además, esta técnica es la que mejor uso hace del ancho de banda. Todo esto hace pensar que es la tecnología adecuada para el sistema de juego de FLEQ, si no contara con el *handicap* de su todavía reducida implantación en los navegadores.

3.1.3. Selección final: Socket IO

Tras haber evaluado las características más relevantes de las tecnologías *push*, llega el momento de decidir qué sistema será el implementado para dar vida al motor de juego de FLEQ.

Ninguna de las anteriores técnicas nos proporciona una garantía total respecto a soporte, velocidad, facilidad de implementación y optimización del uso del ancho de banda.

Si bien es cierto que la API de WebSockets de HTML5 cumple con la práctica totalidad de estos requisitos ideales, como se ha dicho, actualmente su implantación puede suponer un problema debido a que gran parte de los usuarios utilizan navegadores que aún no lo soportan.

Existen sitios web que hacen uso de ciertas innovaciones técnicas, e impiden a usuarios con navegadores obsoletos su uso, instando a su actualización. FLEQ no pretende causar ese primer impacto, que experiencias previas sugieren que es una práctica que se debe evitar en la manera de lo posible.

El éxito de una aplicación web, especialmente al inicio de su existencia, se basa fuertemente en su facilidad de uso. El hecho de forzar a los usuarios a actualizar sus navegadores puede suponer malas sensaciones desde un principio, que lleguen incluso al no acceso a este sistema.

Probablemente, en un periodo aproximado de un año, el soporte de WebSockets en los navegadores implantados sea muy cercano al 100 %, pero actualmente, éste no es el caso. Por todo esto, se debía buscar una solución intermedia a medio plazo, con vistas a la utilización de WebSockets en el largo.

Tras descartar el tradicional *polling* por los problemas que genera su uso en un cliente

móvil, ya explicados en secciones anteriores, se antojaba necesario adoptar como mecanismo de comunicación alguna de las técnicas Comet, que, si bien no eran las ideales, su soporte en navegadores actuales es mucho mayor que el de WebSockets. Especialmente en el caso de HTTP Long Polling, que únicamente necesita que el cliente soporte el lenguaje JavaScript.

Sin embargo, entre las técnicas Comet, HTTP Multi-part Streaming se antojaba la mejor de todas, debido a su menor consumo de ancho de banda.

Esta solución sería una solución intermedia, con la vista puesta en la implantación de WebSockets como sistema definitivo en el medio-largo plazo.

La solución no terminaba de convencer, ya que es difícil como desarrollador, optar por implementar técnicas que de antemano se conoce no son las óptimas por el hecho que se está argumentando.

Por esto, se trató de agotar todas las vías antes de comenzar la implementación de HTTP Multi-part Streaming como núcleo del sistema de juego.

En esta tarea de búsqueda de alternativas, se halló una librería de JavaScript que podría resolver todos los conflictos que durante estas líneas se han planteado: la librería Socket IO.

Socket IO

Esta librería de JavaScript permite resolver los problemas derivados del soporte de las diferentes técnicas *push* en los navegadores, de manera que implementa diferentes sistemas de comunicación y emplea el mejor de ellos en función de las posibilidades del cliente en el que esté trabajando.

Socket IO es justo lo que se estaba buscando. Un mecanismo que permite que cada cliente emplee el mejor mecanismo de transmisión en función de sus posibilidades individuales, sin acotar el uso de la aplicación a usuarios con navegadores modernos, ni optar por una técnica intermedia.

De este modo, se dará soporte a un porcentaje muy cercano al 100 % de los clientes, se optimizará el funcionamiento de FLEQ dependiendo del navegador en el que se use,

no se causará trastorno a los usuarios con una necesaria actualización del navegador, y lo que es más importante, es una solución válida para el medio y largo plazo.

El funcionamiento de Socket IO parece bastante claro, por lo que se ha podido deducir de las líneas anteriores. Implementa distintos mecanismos de comunicación bidireccional entre cliente y servidor, y trata de utilizar el mejor de todos en función del navegador.

Las técnicas de comunicación que implementa, y que trata de utilizar, serían las siguientes, por este orden:

1. WebSockets.
2. Adobe Flash Socket.
3. AJAX Long Polling.
4. AJAX Multi-part Streaming.
5. Forever Iframe.
6. JSONP Polling.

Como se puede observar, Socket IO implementa desde la técnica más moderna como es WebSocket, hasta el tradicional *polling* como último recurso, pasando por las diferentes técnicas Comet (basadas en AJAX) explicadas a lo largo de esta memoria.

Como segunda opción aparece una solución basada en Flash, la cual es una buena alternativa a WebSocket, pero que puede causar muchos problemas en dispositivos que no soportan esta tecnología, como pueden ser los terminales de Apple, por lo que se pondrá interés en evitar que sea utilizada.

En este *ranking* de mecanismos que Guillermo Rauch, creador de la librería, estableció, se observa como HTTP Long Polling es, a su juicio, una técnica preferible a HTTP Multi-part Streaming.

Anteriormente se indicó que de no haber encontrado esta alternativa, se hubiera optado por implementar una de las técnicas Comet, preferiblemente la basada en Multi-part Streaming, argumentando como razón la optimización del uso de ancho de banda.

Esto nos lleva a pensar que esta decisión podría no haber sido la correcta, ya que probablemente los problemas derivados del uso de Multi-part Streaming (soporte del flag multi-part, buffering...) puedan ser más perjudiciales que el propio uso de *long polling*, que penaliza el uso de ancho de banda.

La librería Socket IO es además razonablemente fácil de implementar en el lado del cliente, ya que su API es similar a la que utiliza WebSocket, con métodos que permiten la conexión, reconexión y desconexión, entre otros.

En el lado del servidor, tampoco se antoja demasiado complicada, puesto que existen módulos ya implementados para frameworks como Django o Tornado, los cuales se presentan como los principales candidatos para el desarrollo de FLEQ.

El soporte que Socket IO proporciona en los diferentes navegadores, queda resumido en la siguiente tabla:

	IE	Firefox	Chrome	Safari	Opera
Soporte	5.5+	3+	4+	3+	10.61+
Actual	9.0	16.0	23.0	6.0	12.1

Cuadro 3.1: Soporte de Socket IO en las diferentes familias de navegadores

Como se puede observar en la relación anterior, en la que aparecen las principales familias de navegadores de escritorio, Socket IO es soportado desde versiones muy antiguas, ya que en el peor de los casos, es necesario únicamente que estos soporten lenguaje JavaScript, para realizar AJAX Long Polling o JSONP Polling.

En cuanto a navegadores de dispositivos móviles, como Android Webkit, iOS Safari, etc. soportan Socket IO desde su primera versión, ya que desde su lanzamiento soportaban JavaScript.

Por todos estos motivos, Socket IO se presenta como una solución que cumple todos los requisitos que se exigían al planteamiento de este análisis:

- Soporte casi total en navegadores implantados actualmente.
- Solución óptima a corto, medio y largo plazo.
- Implementación de WebSockets como primera alternativa.

- Optimización de la tecnología de comunicación en función del cliente.
- Implementación asequible tanto en el servidor como en el cliente.

Esto conlleva a que esta librería sea la solución definitiva para dar vida al nuevo motor de juego de FLEQ, que sustituirá a IRC.

3.2. Comparativa de tecnologías para el diseño de clientes móviles

El diseño de aplicaciones para dispositivos móviles está sufriendo cambios en los últimos tiempos debido a la aparición de nuevas técnicas que hacen cuestionarse si merece la pena el desarrollo de *apps* nativas. Estas técnicas tienen nombre y apellidos comunes: HTML5.

3.2.1. Aplicaciones móviles nativas vs HTML5

El desarrollo de aplicaciones está cambiando, eso es evidente. Tras el *boom* que supuso la llegada y expansión de los *smartphones*, se desató entre los desarrolladores una nueva vía sobre la que extender y plasmar sus ideas.

Cada una de las principales plataformas móviles creó a su alrededor una comunidad de desarrolladores que nutrían y nutren sus respectivas tiendas de aplicaciones.

Cada una de estas plataformas proporciona entornos de desarrollo basados en diferentes lenguajes, y APIs mediante las cuales acceder a todas las características de los terminales y crear aplicaciones sobre un mundo de posibilidades mucho más amplio que el que podía abrir un equipo convencional de escritorio: GPS, sensores, mensajería *everywhere*, etc.

Esta diversidad de SDKs (Software Development Kit) hace que el desarrollo de una misma app tenga que ser abordado para varias plataformas de manera independiente, siendo generalmente tres: Android, iOS y BlackBerry. Recientemente, se empieza a añadir un cuarto: Windows Phone. Todo esto implica cuatro desarrollos distintos de una misma aplicación.

Esto se traduce en la necesidad de un mayor número de recursos económicos y temporales.

La llegada e implantación progresiva de HTML5 junto con CSS3, ha abierto un nuevo mundo de posibilidades en el desarrollo de aplicaciones para dispositivos móviles. Además de las posibilidades que estas tecnologías aportan en cuanto al desarrollo de interfaces de usuario, hay que añadir las variadas APIs que incorpora HTML5, y que se irán aumentando en el futuro: WebSockets, geolocalización, almacenamiento, etc.

Esto supone que mediante HTML5 se pueda cubrir gran parte de la funcionalidad que proporcionan los desarrollos nativos.

Por todo esto, cabe hacer un análisis exhaustivo de los requerimientos de la aplicación que se quiere desarrollar, para evaluar si un desarrollo en HTML5 puede cubrir los requisitos necesarios, o si por el contrario, se hace necesario recurrir a desarrollos nativos, con el aumento de inversión que ello implica.

La evolución de este nuevo panorama parece apuntar a que HTML5 acabará por imponerse a los desarrollos nativos, por todo lo argumentado anteriormente. Prueba de ello es el próximo sistema operativo que RIM lanzará en enero de 2013 para sus dispositivos BlackBerry, cuyo SDK de desarrollo de aplicaciones estará basado en HTML5.

Esta apuesta de BlackBerry puede ser el termómetro que determine si finalmente, las aplicaciones nativas quedarán en el recuerdo en favor de las aplicaciones multiplataforma basadas en las nuevas tecnologías web, que facilitan enormemente el desarrollo.

Actualmente, han proliferado plataformas como PhoneGap, propiedad de Adobe, que permiten la traducción de aplicaciones desarrolladas en HTML5, CSS3 y JavaScript a aplicaciones nativas, incluso permiten al desarrollador acceder a APIs de los dispositivos de forma nativa: cámara, contactos, ficheros, notificaciones, acelerómetro, red, GPS, etc.

De este modo, en casos que requieran manejo de funcionalidades que no puedan ser accesibles desde HTML5, se puede realizar un desarrollo genérico de la interfaz de usuario, y posteriormente, a través de PhoneGap, agregar estas funcionalidades para cada plataforma.

3.2.2. Selección final: HTML5

Inicialmente el proyecto que está siendo recogido en esta memoria comprendía el desarrollo de un cliente Android nativo para la aplicación FLEQ.

Así fue planteado inicialmente. Además de eliminar el sistema de comunicación basado en IRC, se pretendía hacer accesible el servicio desde el sistema operativo de Google.

El desarrollo del proyecto comienza obviamente, por desligar el sistema de IRC, para después, con la aplicación terminada, desarrollar el cliente móvil.

Como se explicó en la sección anterior, el sistema de comunicación seleccionado tras un profundo estudio, está basado en WebSockets y en técnicas AJAX.

El SDK de Android no da soporte nativo a WebSockets, lo cual complicaba sobremedida este punto, tan determinante para el proyecto.

Además de esto, desarrollar una aplicación exclusiva para clientes con dispositivos Android no parecía ser lo más adecuado para una plataforma que pretende expandirse y no caer en el olvido cuando este Proyecto Fin de Carrera acabe.

Para conseguir esto, o al menos intentarlo, se deben jugar todas las cartas posibles, y dar soporte únicamente a móviles Android sería contraproducente desde las primeras pruebas, ya que no todos los usuarios (alumnos en este caso) podrían hacer uso de FLEQ.

Todo lo expuesto supuso que la opción de realizar una aplicación web adaptada a dispositivos móviles fuera cogiendo fuerza respecto al desarrollo nativo de un cliente para Android.

Además, FLEQ no requiere la utilización de ninguna API especial, como pudiera ser, por ejemplo, el acceso a sensores del teléfono, por lo que un desarrollo nativo no sería estrictamente necesario.

Por tanto, en el caso particular que nos ocupa, un desarrollo en HTML5 supone las siguientes ventajas:

- Soporte multiplataforma: FLEQ será accesible desde cualquier *smartphone*.
- Facilidad de implementación: el desarrollo de una aplicación en HTML5 es más

sencillo que el desarrollo de una *app* nativa de Android, tanto el front-end (interfaz de usuario) como el back-end (lógica de la aplicación).

- Los navegadores móviles soportan Socket IO, y un alto porcentaje de ellos, también soportan WebSockets, mientras que el SDK de Android no lo hace.
- Mayor facilidad en corrección de bugs: cualquier error puede ser subsanado por el desarrollador sin que los usuarios tengan que actualizar la versión de su aplicación, como ocurre con las nativas.
- Mayor facilidad de distribución: la aplicación es accesible a través de cualquier navegador, no es necesario publicarla en tiendas de aplicaciones, las cuales pueden requerir tasas de publicación, como en el caso de la App Store de Apple.
- Soporte para tabletas: no será necesario adaptar la UI (User Interface) a este tipo de dispositivos, ya que los navegadores redimensionan el contenido gracias a las propiedades de HTML5 y CSS3 Media Queries.

Tras esta enumeración de ventajas, parece obvio pensar que la decisión final adoptada es realizar una web móvil en HTML5 para dar acceso a FLEQ al mayor número de usuarios sea cual sea el dispositivo que estén utilizando.

Sin duda alguna, esta universalidad es un elemento diferenciador para el sistema, y que aporta mucho valor para convertir a FLEQ en una plataforma sólida, moderna y abierta.

3.3. Comparativa de lenguajes y frameworks para la implementación del servidor

Diversos son los lenguajes disponibles para la implementación de aplicaciones web. De entre todos ellos, se debe elegir cuidadosamente cuál emplear, en función de los requerimientos del sistema y de las posibilidades que éste ofrezca, en cuanto a módulos, APIs disponibles o *frameworks* para el desarrollo.

Entre las diferentes posibilidades que existen para el desarrollo de una aplicación web de las características de FLEQ, destacan tres lenguajes por encima del resto: PHP, Python y Ruby.

3.3.1. Selección final: Python

Para realizar este análisis y argumentar la selección del lenguaje y framework empleado en el desarrollo de la aplicación, se necesita desgranar a grandes rasgos la arquitectura interna de FLEQ, aunque esto vaya a ser tratado ampliamente en el siguiente capítulo.

FLEQ está compuesto por una aplicación web estándar, con sistema de usuarios, recursos, una base de datos donde se almacena la información, etc. Lo habitual que compone cualquier web al uso. Sin embargo, FLEQ también contará con un sistema gestor de torneos, el cual está basado en la ejecución de determinadas tareas periódicas por medio de *scripts*.

Por otro lado, FLEQ debe dar soporte a un sistema de comunicación, que como se analizó anteriormente, utilizará la librería Socket IO, por lo que habrá que implementar un módulo que dé soporte a WebSockets.

Todas estas funcionalidades que debe implementar FLEQ, hacen que sea preferible la utilización de un lenguaje multiparadigma que permita acometer todas ellas, sin necesidad de realizar cada módulo con lenguajes alternativos. De esta forma, se conseguirá un entorno integrado, que permitirá realizar un desarrollo más efectivo, debido a la especialización que se adquirirá en un sólo lenguaje.

Entre los lenguajes multiparadigma disponibles para el desarrollo de aplicaciones web, podemos destacar PHP, Ruby y Python.

El lenguaje Java ha sido descartado como alternativa, debido a que no es un lenguaje multiparadigma, y a que sus frameworks de desarrollo web no son tan potentes como lo puedan ser Ruby On Rails o Django. Además es un lenguaje bastante pesado en el lado del servidor, lo que puede ralentizar el flujo de ejecución de la aplicación.

A su vez, la familia de lenguajes C, es empleada para el desarrollo de otro tipo de sistemas no orientados a la web, como puedan ser *kernels* de servidores, sistemas operativos, etc., más a bajo nivel.

Centrando el análisis en los tres anteriores, PHP es el siguiente lenguaje en ser descartado.

La primera razón para realizar este descarte se deriva de su anatomía. Pese a ser

un lenguaje orientado totalmente al desarrollo web, su código se integra en los propios documentos HTML que cargan los navegadores, algo que no parece ser lo más adecuado para FLEQ.

Además, pruebas de rendimiento realizadas han determinado que es el más lento de los tres, debido a que su interpretación se realiza en tiempo de ejecución. Esto podría ser especialmente delicado para la implementación del sistema de juego de FLEQ, en el que la velocidad es algo primordial.

En cuanto a sus frameworks, a pesar de que cuenta con diversos entornos, ninguno alcanza el nivel de funcionalidad que pueden aportar Django o Ruby On Rails.

Por tanto, la lucha final se dirime entre Python y Ruby, de lo que gran parte de culpa tiene la potencia de los frameworks citados anteriormente.

Python vs Ruby

Ambos lenguajes son multiparadigma, lo cual permite que puedan ser empleados con diversos fines. Sin embargo, a pesar de esto, Ruby es fuertemente orientado a objetos, lo que deja patente que aunque soporte otros paradigmas, su desarrollo puso especial interés en la orientación a objetos.

Python y Ruby son lenguajes orientados claramente al desarrollo de aplicaciones web, y prueba de ello es la cantidad de web frameworks que han sido desarrollados para ambos.

En cuanto a velocidad de ejecución, Python es más rápido que Ruby, algo muy a tener en cuenta para fortalecer la experiencia de usuario.

Ambos lenguajes cuentan con comunidades de desarrollo sólidas, pero el éxito que ha tenido Python, superior al de Ruby, ha favorecido junto a su facilidad de aprendizaje, a que hayan proliferado multitud de módulos y extensiones que permiten abordar el desarrollo de una aplicación web reutilizando o incluyendo estas extensiones que ya han sido implementadas.

Analizando las posibilidades de ambos lenguajes para desarrollar y dar soporte al sistema de comunicación de FLEQ, basado en Socket IO, hayamos un web framework de Python creado específicamente para dar soporte a este tipo de aplicaciones en tiempo

real, Tornado, analizado en el capítulo anterior.

Tornado, junto con la mayor versatilidad de Python y su mayor rapidez, hacen que este lenguaje tome ventaja sobre Ruby, en lo referente a la implementación de FLEQ.

El último punto de este análisis, y el que determinará definitivamente por qué vertiente se inclina el proyecto FLEQ, vendrá dado por una rápida comparativa entre Ruby On Rails y Django, los dos principales web frameworks de estos dos potentes lenguajes.

Cabe aclarar que esta comparativa se plantea ya que Tornado será empleado como herramienta que dé soporte al módulo de comunicación, por sus características, pero no albergará la lógica de lo que será FLEQ en sí mismo, debido a que los otros dos son mucho más potentes en este sentido.

Uno de los puntos fuertes de Django, destacado por multitud de programadores a lo largo y ancho de la web, es su documentación, algo muy a tener en cuenta a la hora de decidir por qué web framework decantarse.

Además, Django permite crear proyectos web rápidamente y extenderlos mediante aplicaciones o módulos de terceros de forma trivial, mediante un gestor de aplicaciones. La escalabilidad de Django es otro punto muy a tener en cuenta sobre Ruby On Rails, que no puede presumir de ser demasiado escalable.

Por último, el interfaz gráfico de administración que proporciona Django es un elemento diferenciador clave respecto al resto de web frameworks, que posibilita una mejor gestión de la aplicación y sus componentes, desde la base de datos hasta los usuarios y los recursos web.

La siguiente lista de ventajas de Django sobre Ruby On Rails, resume las razones que determinarán la decisión final:

- Velocidad: Python es más rápido que Ruby.
- Escalabilidad: Django integra aplicaciones de terceros de forma trivial.
- Arquitectura: basada en la MVC, el modelo MPV aporta mayor flexibilidad.
- Documentación: Django cuenta con una documentación excelente, muy por encima de la de Ruby On Rails.

- Administración: interfaz gráfica para la gestión de los contenidos del proyecto.
- Facilidades: el framework de Python es más accesible que Ruby On Rails para programadores sin experiencia en estos lenguajes.

Por todas estas razones, Python será el lenguaje seleccionado para dar vida a FLEQ, utilizando el web framework Django para dar soporte al grueso de la aplicación, y por otro lado, Tornado, para dar soporte a las comunicaciones del sistema de juego.

Capítulo 4

Diseño e implementación

*La inspiración existe, pero tiene
que encontrarte trabajando.*

Pablo Ruiz Picasso

En este capítulo se presentan las características más relevantes que componen la arquitectura y el diseño de FLEQ.

Se realizará un análisis de los componentes que darán vida a la plataforma, estudiando en profundidad el diseño y la implementación de sus dos pilares principales: el servidor y el cliente.

4.1. Introducción

El esquema principal de la arquitectura de la aplicación queda recogido en la figura 4.1, en la cual se puede observar como el sistema está compuesto por dos grandes módulos, que a su vez se dividen en otros dos secundarios:

- El servidor, el cual estará alojado en una máquina física que albergará tanto la lógica de la aplicación, como la base de datos que contiene la información de la plataforma.
- El cliente, compuesto por dos interfaces de usuario completamente independientes, una adaptada a equipos de escritorio y a otra a dispositivos móviles, las cuales permitirán a los usuarios interactuar con el sistema.

Una vez realizada una primera aproximación que ayude a entender a grandes rasgos la estructura de FLEQ, se va a proceder a desgranar cada uno de los módulos que componen los dos puntos enumerados anteriormente, de forma individual.

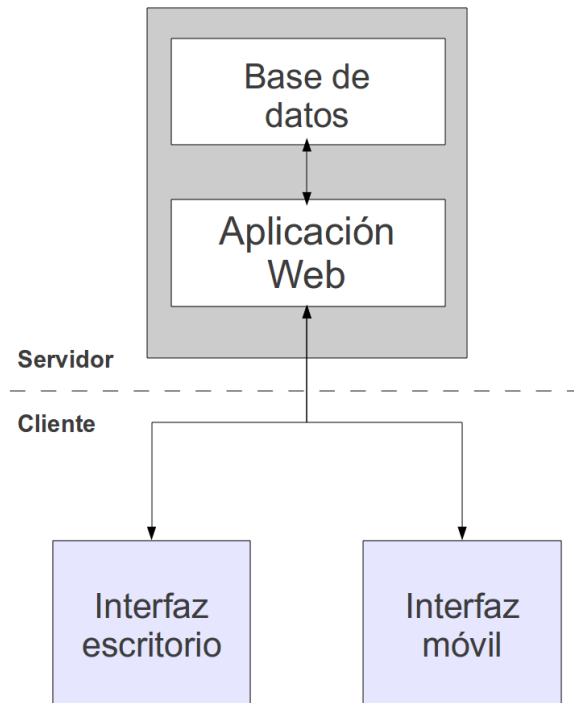


Figura 4.1: Arquitectura básica de FLEQ

4.2. Diseño e implementación del servidor

4.2.1. Modelo de datos

El modelo de datos de FLEQ está organizado siguiendo una estructura relacional, en la cual los elementos que la componen establecen interconexiones entre sí. Este paradigma de modelo de base de datos fue postulado en 1970 por el informático inglés Edgar Frank Codd, quedando recogidos sus fundamentos en los siguientes puntos:

- Tablas interconexionadas entre sí creando relaciones de dependencia.
- Cada tabla tiene un identificador (nombre) único.
- Cada tabla está compuesta por un conjunto de registros.

- Las relaciones entre tablas se establecen por medio de claves primarias (tabla padre) y claves foráneas (tabla hija).
- Las claves foráneas contienen el valor de la clave primaria de la tabla padre, estableciendo así la relación entre ambas.

El modelo de datos relacional que implementa el presente Proyecto Fin de Carrera se basa en el modelo heredado de la anterior plataforma, desarrollada por Arturo Moral y Félix Redondo Sierra.

Partiendo de esta herencia, se han realizado algunas simplificaciones en algunos elementos que se consideraban innecesarios, con lo que se ha conseguido reducir el volumen de la información almacenada, y simplificar la extracción de datos.

Cabe decir que estos cambios han sido poco significativos, y el *kernel* del modelo de datos heredado sigue siendo el mismo, ya que se consideró que era una buena aproximación, y lo más importante, ya había sido testeado previamente obteniendo resultados más que satisfactorios.

Por tanto, y teniendo siempre en mente que el objetivo principal del presente proyecto no pasaba por reestructurar la base de datos, las correcciones que se han llevado a cabo han tenido como propósito facilitar ciertos aspectos durante la elaboración del mismo.

Este modelo de datos definitivo queda recogido en el diagrama UML (Unified Modeling Language) de la figura 4.2, elaborado con el propósito de que el lector pueda explorar el funcionamiento interno del gestor de torneos de FLEQ.

Del diagrama UML se deducen los siguientes puntos:

- La tabla de torneos puede ser considerada el modelo padre del sistema de datos.
- La tabla de usuarios contendrá los datos de estos, y estará relacionada con los distintos elementos del modelo, determinando su propietario.
- Cada torneo (Tournament) se compone de rondas (Rounds), y éstas a su vez se componen de juegos (Games).
- Las preguntas (Questions) están catalogadas por categorías (Categories). A su vez

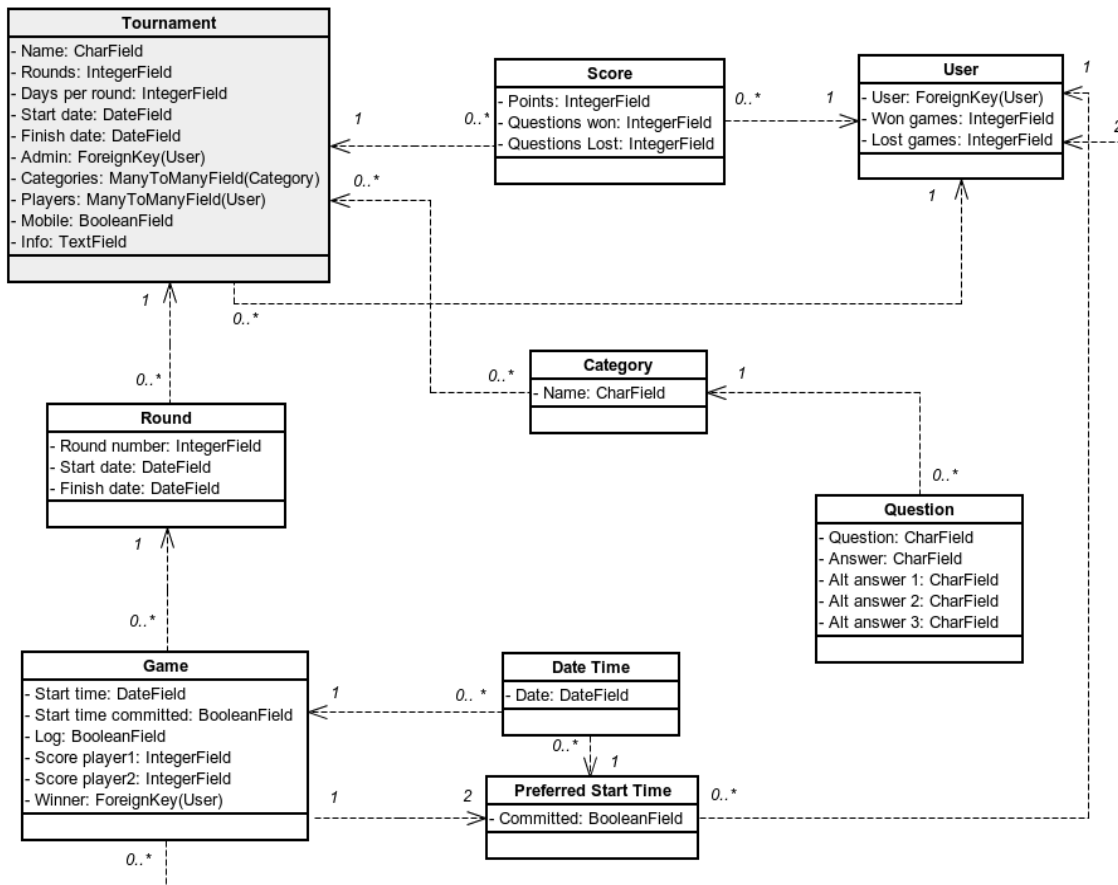


Figura 4.2: Diagrama UML del modelo de datos de FLEQ

un torneo puede estar compuesto de varias categorías de preguntas, y una categoría puede ser empleada en diferentes torneos.

- Cada juego tiene asociada una hora de comienzo, que puede ser configurada por los usuarios. Para ello, se dispone de los modelos *Date Time*, que corresponderá a cada una de las horas seleccionadas, y *Preferred Start Time*, que servirá para determinar si cada usuario ha realizado ya su solicitud, y permitirá establecer la hora definitiva de la partida cuando ambos jugadores hayan seleccionado sus horas de preferencia.

4.2.2. Arquitectura REST

REST (Representational State Transfer) es un estilo arquitectural para sistemas distribuidos ampliamente extendido en la web estática (ficheros) y parcialmente en la web dinámica (aplicaciones web).

Esta arquitectura está basada en un conjunto de normas, recogidas en la siguiente enumeración:

- Cada recurso debe tener una URL propia y única, entendiendo por recurso cualquier tipo de información que quiera hacerse accesible a través de la red.
- Los recursos contienen hiperenlaces a otros recursos.
- Conjunto limitado y estándar de métodos HTTP (GET, PUT, POST y DELETE).
- Múltiples representaciones por recurso.
- Comunicación sin estado: las peticiones contienen toda la información necesaria para ser procesada, ya que el cliente guarda su estado, lo que permite que la lógica del servidor sea más simple.

Estas prácticas proporcionarán ciertas ventajas a la aplicación en particular, y a la red en general. De entre ellas, cabe mencionar las siguientes:

- Aumento de la escalabilidad.
- Maximización de la reutilización, ya que cada recurso cuenta con un identificador único.
- Minimización de acoplamiento.
- Permite la evolución.
- Eliminación de fallos parciales.

FLEQ ha seguido las pautas más significativas y restrictivas de esta arquitectura, por lo que puede considerarse una aplicación REST:

- Cada recurso en FLEQ cuenta con una URL única.
- Los recursos contienen hiperenlaces que permiten navegar por la aplicación desde el raíz sin necesidad de recurrir a la barra de direcciones del navegador, tanto en la versión de escritorio como en la versión móvil.

- La aplicación hace uso únicamente de los métodos GET y POST para la solicitud y el envío de datos respectivamente. Los métodos PUT y DELETE no han sido implementados, ya que un gran porcentaje de los navegadores actuales no los soportan, junto a que suelen ser rechazados por *firewalls* para prevenir problemas de seguridad y usos malintencionados.
- FLEQ permite comunicación sin estado: el cliente mantiene el estado de su sesión por medio de *cookies*, que son almacenadas y enviadas en cada petición que es realizada al servidor.

Como se ha podido comprobar, FLEQ cumple cuatro de los cinco requisitos que propone la arquitectura REST. El único que no se ha seguido es la disposición de todos los recursos en múltiples representaciones (HTML, JSON, XML, texto plano, etc.), ya que las necesidades de la aplicación no requerían la implementación de estas extensiones.

De las cinco características de REST, ésta puede ser considerada como la menos estricta, ya que su no cumplimiento no implica que la aplicación no pueda ser catalogada como *REST application*.

Por lo tanto, FLEQ sí puede ser considerada una aplicación web que sigue la arquitectura REST.

4.2.3. Aplicación web

La aplicación web en que se basa el proyecto FLEQ está compuesta por varios módulos con funcionalidades independientes, quedando recogida su estructura en la figura 4.3.

Estos módulos son los siguientes:

- Django: lógica de la aplicación, control de usuarios, recursos y administración del sistema.
- Tornado: sistema de comunicación basado en Socket IO.
- Gestor de torneos: scripts escritos en Python encargados de la generación y administración de torneos.

- FLEQbot: robot encargado de la ejecución de partidas en FLEQ.

A continuación se procederá a explicar con detalle estos componentes de forma individual.



Figura 4.3: Arquitectura interna de la aplicación web

Django

La aplicación web cuenta con un módulo desarrollado sobre el framework Django, lo que compondrá la parte principal de la misma.

Django será el encargado de interconectar FLEQ con sus usuarios, ya que implementa toda la lógica para la gestión de recursos, por medio de los cuales, los usuarios podrán hacer uso de esta herramienta de e-learning, basándose en el patrón de diseño MVC que será explicado con profundidad a la conclusión de esta introducción.

Además de esto, Django proporcionará el módulo de administración de la aplicación, que permitirá al administrador gestionar mediante una interfaz gráfica todos los recursos de la base de datos asociada al proyecto.

Por medio de un *middleware*, Django será el encargado de discriminar el tipo de dispositivo con el que los usuarios interactúan con FLEQ, con el objetivo de facilitar la interfaz de escritorio o la interfaz adaptada a dispositivos móviles, según corresponda.

Patrón de diseño MPV

En la introducción que se realizó de Django en capítulos anteriores, se esbozó la idea de que este framework seguía la arquitectura MVC (Modelo Vista Controlador), pero con ciertas modificaciones que en realidad la convertían en una arquitectura MPV (Modelo Plantilla Vista).

El MVC es un patrón de diseño de software que separa los datos, la lógica y la presentación de la aplicación en tres componentes independientes e intercomunicados.

El proceso de ejecución de esta arquitectura se resume en los siguientes pasos:

1. El usuario interactúa con la interfaz (vista o *view*).
2. El controlador o *controller* recibe la información que el usuario envía.
3. Según la información recibida, el controlador consulta o aplica los cambios en el modelo correspondiente.
4. El controlador actualiza el contenido de la vista.
5. La interfaz espera una nueva acción del usuario.

En el caso de Django, este patrón de diseño sufre modificaciones, principalmente en cuanto a la nomenclatura de los componentes:

- La filosofía de Django define que el controlador es en realidad la vista, ya que es el módulo que describe que información es presentada al usuario. Por este motivo, el fichero en el que se definen los métodos que en otros frameworks como Ruby On Rails serían controladores, en Django es nombrado *views.py*, lo que da muestras de esta variación de nomenclatura.
- Las vistas del patrón MVC pasan a ser plantillas o *templates* en Django, ya que ésta es la verdadera forma en la que este framework presenta la información e interactúa con el usuario.
- El modelo permanece inalterable, siendo atribuido en ambos casos a los objetos almacenados en bases de datos.

Bien MVC o MPV, el objetivo de ambos patrones es común: separar la lógica de la aplicación, el modelo de datos y la interfaz de usuario en tres bloques completamente independientes pero interrelacionados, con lo que se pretende obtener los siguientes beneficios:

- Mejor escalabilidad del sistema.
- El diseño e implementación de los componentes puede realizarse en paralelo.
- Los modelos pueden ser representados en diferentes vistas y manejados por diferentes controladores.
- Los modelos no necesitan sufrir modificaciones para adaptarse a diferentes tipos de vistas.
- Notificación automática en las vistas de cambios en los modelos.

Aplicaciones

Django es un web framework modular, lo que permite añadir funcionalidades adicionales al proyecto por medio de aplicaciones independientes. Todo proyecto Django cuenta inicialmente con aplicaciones preinstaladas, tales como *django.contrib.auth* o *django.contrib.sessions*, relacionadas con la gestión de usuarios (acceso, registro, sesiones, etc.).

Adicionalmente, Django cuenta con aplicaciones o extensiones tales como el módulo de administración ya explicado anteriormente, que facilita la gestión de los recursos del proyecto de forma rápida y ágil por medio de una interfaz gráfica accesible a través del navegador.

Este módulo ha sido obviamente utilizado en FLEQ, ya que su integración es trivial y sus beneficios considerables.

La aplicación en la que se encuentra la lógica del proyecto ha sido denominada *Quiz-bowl*. En ella, se encuentra la declaración del modelo de datos de FLEQ, así como las vistas (*views*), distribuidas en diferentes ficheros para facilitar tanto la implementación como la depuración. Los más destacados son los siguientes:

- *views_admin.py*: en este fichero se incluyen los métodos relacionados con funcionalidad accesible únicamente para usuarios con privilegios de administración (creación de torneos y subida de preguntas), así como la generación de los respectivos formularios.
- *views_connect.py*: métodos relacionados con la gestión de cuentas de usuario (login, logout, sign in) y sus formularios asociados.
- *views_games.py*: gestión de partidas (selección de horario, acceso a ventana de juego, histórico de juegos ganados y perdidos).
- *views_tournaments.py*: métodos relacionados con la gestión de torneos (registro/cancelación de suscripciones, estadísticas, rondas, visualización de torneos próximos, activos o finalizados, etc.).

Middlewares

La definición estándar de *middleware* establece que es un componente de software que permite la comunicación entre dos aplicaciones independientes.

En Django, su comportamiento no es exactamente ese. Un *middleware* en Django es un plugin ligero de bajo nivel que facilita el procesado de peticiones HTTP.

Cada *middleware* es responsable de realizar una función específica. Django contiene varios de estos componentes predefinidos, aunque el desarrollador puede crear los suyos propios para cubrir necesidades particulares.

Esta especie de filtros es aplicada de forma gradual a las peticiones HTTP antes de que sean procesadas por las vistas, de modo que pueden verse como capas que protegen la entrada a las vistas, como si se tratara de una cebolla.

Son los encargados de la comprobación de la sesión de los usuarios, autenticación, etc.

Adicionalmente, FLEQ incluye un *middleware* de terceros para la comprobación de la naturaleza de la petición llamado *minidetector*, el cual, basándose en las cabeceras de las peticiones HTTP y un listado de fabricantes de dispositivos móviles, determina de forma adecuada si la petición proviene de un equipo de escritorio o si por el contrario, lo hace desde un dispositivo móvil.

De este modo, la vista será capaz de distinguir entre ambas casuísticas, proveyendo la interfaz correspondiente según cada caso.

Tornado

Tornado será el web framework utilizado para dar soporte al módulo de comunicación de FLEQ. Como se explicó anteriormente, Tornado está compuesto por un conjunto de módulos implementados para dar soporte a comunicaciones en tiempo real, y además, cuenta con un potente servidor web que dará soporte a estas conexiones.

Este web framework será el encargado de atender las conexiones establecidas mediante la librería Socket IO que implementan los clientes. Para este propósito, Tornado cuenta con un módulo abierto, denominado TornadoIO2, y que puede ser instalado por medio del gestor de paquetes de Python, PyPI (Python Package Index).

Este módulo dará soporte a las conexiones establecidas mediante Socket IO, cualquiera que sea el sistema de comunicación establecido por el navegador del cliente (WebSockets, XHR Polling, Multi-part Streaming, etc.).

TornadoIO2 implementa una extensión del servidor base de Tornado que habilitará la funcionalidad necesaria para atender estas conexiones, abstrayendo toda esta implementación, lo que facilitará la labor de desarrollo de este módulo de FLEQ.

Una vez se ha dado soporte a las conexiones en el lado del servidor, se debe establecer el protocolo de comunicación mediante el cual se van a comunicar los clientes con el servidor, tratando de que éste sea lo más escueto posible para reducir el flujo de datos y agilizar las comunicaciones.

Para este fin, se ha decidido implementar un mini-protocolo propio basado en el formato JSON, ya que este sistema es fácilmente parseable, tanto en el lado del servidor (Python) como en el lado del cliente (JavaScript).

Además, mediante JSON se optimiza la cantidad de información enviada, cuidando de este modo el volumen de datos necesario para las comunicaciones.

Este mini-protocolo se basa en un sistema de códigos, al estilo del protocolo HTTP. Para este proyecto, únicamente han sido implementados dos códigos, suficiente para cubrir

las necesidades del sistema:

- Código 1: este código hace referencia a peticiones de conexión por parte de los clientes. Cuando un mensaje JSON es recibido con este identificador, Tornado percibe que un usuario está tratando de conectarse a la plataforma. Su estructura es la siguiente:

$$\{“code”: “1”, “room”: “id”, “user”: “username”\}$$

- Código 2: este segundo código es el empleado para el envío de mensajes, después de haberse realizado la conexión. Su estructura es similar a la anterior, agregando el campo de mensaje:

$$\{“code”: “2”, “room”: “id”, “user”: “username”, “message”: “message”\}$$

Este sistema tiene la ventaja de ser fácilmente escalable, lo que quiere decir que de manera sencilla se podría ampliar el protocolo añadiendo nuevos códigos que permitirían funcionalidades adicionales, como por ejemplo, controlar la presencia de los usuarios.

El funcionamiento interno de este módulo se puede deducir sencillo, por la topología de los mensajes JSON presentados anteriormente:

- El servidor Tornado espera conexiones escuchando en un puerto del servidor.
- Cuando recibe un mensaje en formato JSON de un cliente, lo parsea:
 - Si el código del mensaje es 1, Tornado almacena al usuario en un *set* de Python (si ya existía, lo redefine).
 - Si el código del mensaje es 2, Tornado envía el contenido de ese mensaje a los usuarios almacenados en el *set* que están conectados a la misma *chatroom*.
- Si Tornado recibe orden de desconexión por parte de la librería Socket IO (este método es ejecutado automáticamente cuando el usuario abandona la ventana de juego), elimina al usuario del *set* de usuarios.

Gestor de torneos

FLEQ cuenta con un módulo destinado a la gestión de torneos compuesto por un conjunto de *scripts* que son ejecutados periódicamente.

Este sistema es el encargado de generar los torneos, las rondas que los componen, el sistema de partidas, los mails recordatorios, etc. y es controlado mediante el planificador de tareas Crontab, presente en todos los sistemas Linux.

Mediante Crontab, se establece el *scheduling* de este gestor de torneos, de forma que se garantice la ejecución de estas tareas en el momento adecuado.

Este módulo de FLEQ está compuesto por los siguientes *scripts* escritos, como no podía ser de otra forma, en lenguaje Python:

- *tournament_creator.py*: este *script* es el encargado de crear los torneos el día de su comienzo. Se ejecuta a las 00.01h todos los días, comprueba si hay algún torneo cuyo comienzo esté previsto ese día y lo crea, siempre y cuando haya al menos dos participantes inscritos. Tras esto, notifica por medio de un correo electrónico el comienzo/cancelación del torneo a los jugadores y al administrador.
- *round_creator.py*: de manera análoga al anterior, genera las rondas que están planificadas para cada día e informa a los usuarios. El proceso de generar una ronda comprende el establecimiento de las partidas, para lo cual sigue el sistema suizo de competición, emparejando a los jugadores de acuerdo a su clasificación actual en el torneo. Es ejecutado a las 00.05h todos los días.
- *player_reminder.py*: este *script* se encarga de recordar a los usuarios el comienzo de las partidas. Con media hora de antelación, envía un email a los jugadores cuyas partidas comiencen en el siguiente turno. Se ejecuta todas las horas del día en el minuto 30 de las mismas.
- *game_launcher.py*: encargado de poner en marcha a FLEQbot, el robot encargado del sistema de juego. Este *script* se ejecuta tres minutos antes del comienzo de las partidas, es decir, en el minuto 57 de todas las horas del día.

FLEQbot

El robot encargado de supervisar y gestionar las partidas en FLEQ ha sido denominado FLEQbot, en un alarde de ingenio.

Este robot se compone básicamente de un *script* escrito en Python, que incorpora una librería que le permite interactuar con la aplicación como si se tratara de un usuario real.

Esta librería se llama Selenium, y como no podía ser de otra forma, fue instalada por medio del gestor de paquetes de Python, PyPI.

La potencia de este módulo es realmente asombrosa. Permite interactuar de forma programática con un navegador tal como si lo estuviera haciendo una persona, por medio de sentencias en código Python muy sencillas.

Permite interactuar con varias familias de navegadores, tales como Mozilla Firefox, Google Chrome, Opera, Internet Explorer o Safari.

De esta forma, se pudo resolver de manera exitosa, aunque no sencilla, uno de los puntos de acción que más preocupaban cuando se comenzó la planificación de este proyecto.

Selenium es un software desarrollado para el testeo de aplicaciones web, y está disponible para más lenguajes de programación, no únicamente para Python: Java, Ruby, C# o Perl son otros lenguajes que permiten el uso esta librería.

Mediante la librería Selenium, FLEQbot es capaz de lanzar un navegador e interactuar con él siguiendo este esquema:

1. Abre una ventana de Mozilla Firefox.
2. Se conecta a la URL donde se encuentra alojado el proyecto (*pfj-jgonzalez.libresoft.es*).
3. Analiza el documento HTML en busca del formulario de login.
4. Completa dicho formulario con su usuario y contraseña, y accede a FLEQ.
5. Una vez *logueado*, se dirige a la URL donde se va a disputar la partida haciendo uso del ID de ésta, que le ha pasado como parámetro *game_launcher.py*.

6. Ya en la ventana de juego, examina el código HTML para detectar la ventana de chat, y el formulario de envío de mensajes.
7. Completa el formulario y envía el primer mensaje, avisando de que el comienzo de la partida está próximo.

Este sistema es análogo al que FLEQ utilizaba en su versión anterior, en la que IRC era el sistema de comunicación empleado. En aquella plataforma, FLEQbot lanzaba un cliente IRC para interactuar con el sistema, de igual forma que ahora se lanza un navegador, ya que este es el medio que hemos utilizado para dar soporte al módulo de comunicación.

Como bien es sabido, un navegador consume muchos recursos en una máquina, sobre todo debido a su interfaz gráfica. Por esto, se hizo uso de otro módulo de Python llamado PyVirtualDisplay, que permite crear un entorno gráfico virtual y que la ventana del navegador no exista físicamente, por lo que se consigue que Firefox se ejecute en segundo plano y se reduzcan los recursos consumidos en la máquina.

La figura 4.4 representa de forma gráfica la arquitectura interna de FLEQbot, lo que puede servir para aclarar al lector esta atípica estructura.

Anteriormente se ha explicado y enumerado el proceso de conexión de FLEQbot con la aplicación. A continuación, se va a presentar el diagrama de flujo que sigue este robot ante una partida (figura 4.5).

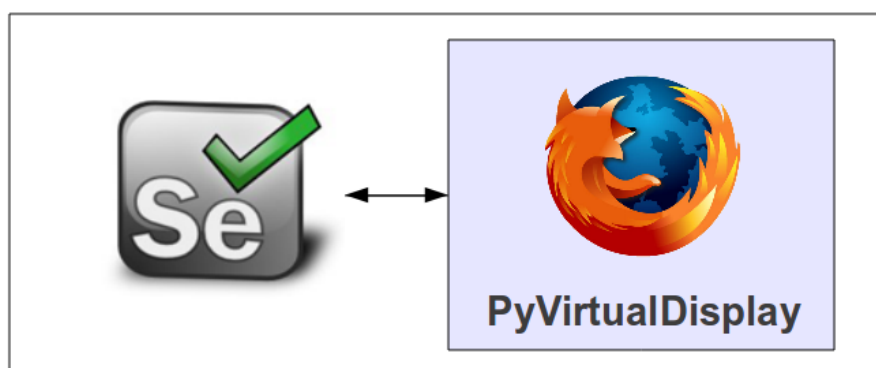


Figura 4.4: Arquitectura interna de FLEQbot

Como se puede observar, FLEQbot sigue un esquema sencillo. Implementa un bucle que se ejecuta con una frecuencia de 1 Hz. En este bucle, se analiza el código HTML de la ventana de juego en busca de nuevos mensajes.

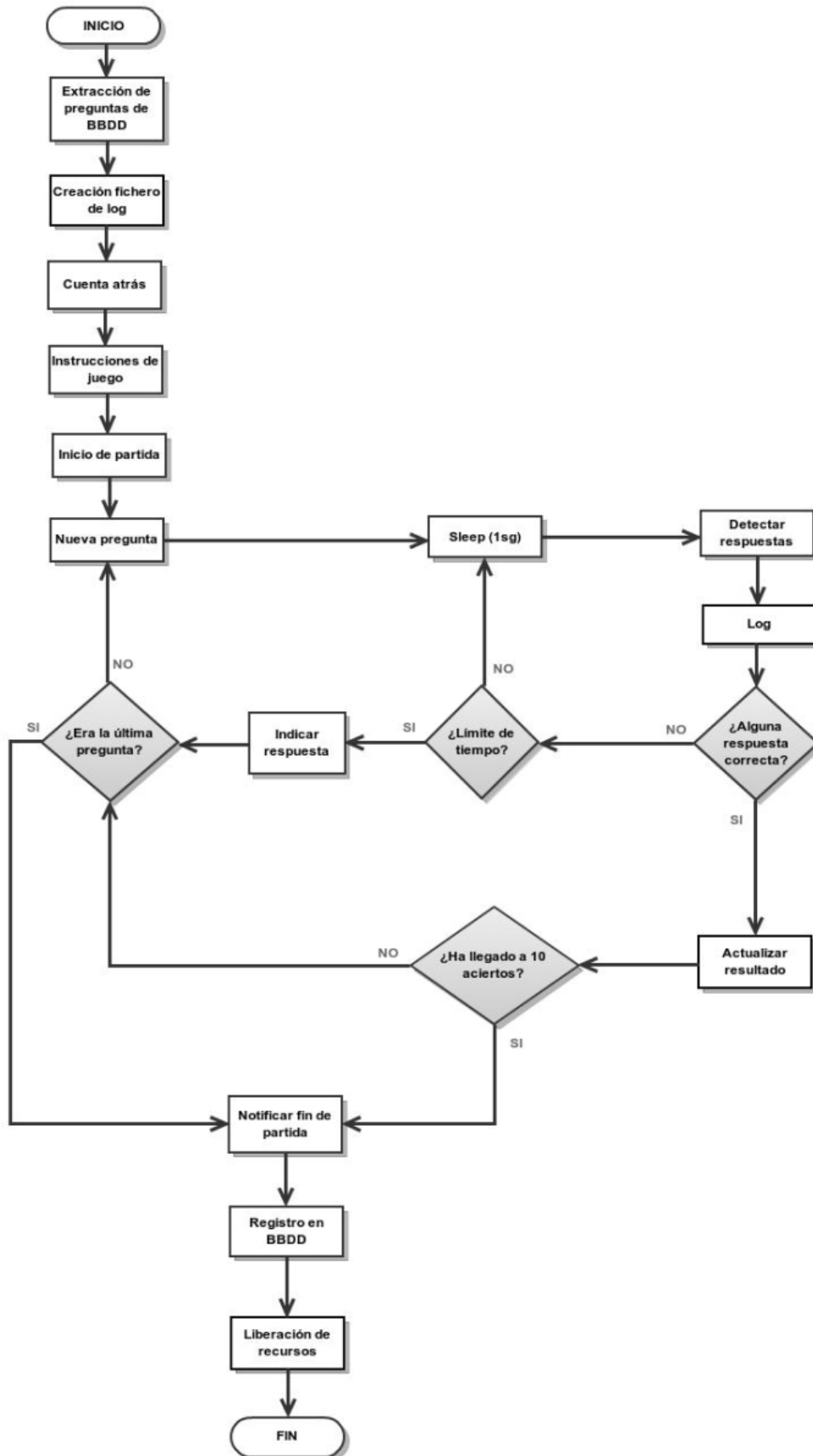


Figura 4.5: Diagrama de flujo de FLEQbot

Estos mensajes serán las respuestas de los usuarios ante la pregunta que se ha lanzado. Se ha establecido un límite de tiempo para que los usuarios respondan, con el fin de evitar partidas sin fin, bien sea porque los usuarios no estén jugando o porque no hallen la respuesta correcta.

Inicialmente, FLEQ implementa un modelo de juego que consiste en que el primer jugador que alcance 10 aciertos gana la partida. De forma análoga a lo anterior, se ha establecido un límite de 30 preguntas, de forma que si se consumen todas sin que ningún jugador haya alcanzado los aciertos establecidos, ganará el que más haya acertado hasta ese momento.

De esta forma quedan controladas todas las posibles casuísticas, y en cualquiera de los casos, las partidas tienen una duración máxima de 45 minutos, con lo que es imposible que se solapen con las de la hora siguiente, y lo que es más importante, se evita el problema que ocasionaría que una partida se extendiera más allá de las 00.00h en un día de inicio de ronda.

Como se aprecia en el diagrama de flujo, FLEQbot implementa un sistema de registro de partidas:

- Una vez realizada la conexión a la ventana de juego, FLEQbot crea un fichero de log de la partida, con el ID del juego como nombre, en la carpeta *logs* del servidor.
- En cada ejecución del bucle, FLEQbot guarda en el fichero de log los nuevos mensajes, añadiéndoles su *timestamp*.
- Cuando acaba la partida, el fichero es cerrado adecuadamente.

Este registro de la partida es accesible por los jugadores una vez que ésta ha concluido, de modo que puedan revisar el flujo del juego y sus errores.

4.2.4. Servidor web

FLEQ se encuentra alojado en una máquina virtual con sistema operativo Debian (Linux), la cual se encuentra en una máquina física que el departamento de Sistemas

Telemáticos y Computación (GSyC) de la URJC tiene en sus instalaciones de Fuenlabrada (Madrid).

Esta máquina virtual fue creada para la puesta en producción del presente proyecto, y por lo cual, hubo que configurar desde cero.

Como parte de esta configuración, se ha tenido que configurar un servidor web para que la aplicación sea accesible a terceros, con el principal propósito de la realización de las pruebas de test que quedarán recogidas en el capítulo siguiente.

Aunque la puesta en producción del servicio no debería ser un hito relevante en la memoria de un proyecto de este tipo, en este caso sí se antoja necesario debido a sus características, ya que como se ha ido tratando en secciones anteriores, consta de un módulo de comunicación con tecnologías *push* que pueden requerir tratamiento especial.

Como se avanzó en el capítulo anterior, el web framework Tornado sería el empleado para dar soporte al sistema de juego de FLEQ, mientras que Django sería empleado para la implementación de la lógica de la aplicación.

Tornado, además de ser un framework de desarrollo web, también incorpora un potente servidor web que puede ser utilizado en producción. Este servidor, como es lógico, acepta las conexiones que establece la API de WebSockets, a diferencia de otros servidores web, o módulos como el *mod_wsgi* de Apache, que impiden el uso de esta tecnología.

Tornado incorpora un módulo WSGI (Web Server Gateway Interface), por lo que podría ser puesto en producción tras un servidor Apache funcionando como proxy, pero el inconveniente indicado anteriormente lo descarta como opción real para ser utilizado en FLEQ.

Este módulo WSGI además permite la interoperabilidad entre Tornado y otros web frameworks escritos en Python, como por ejemplo Django. Por esto, podríamos plantear la posibilidad de conectar ambos frameworks, y utilizar el servidor web de Tornado para poner el servicio en producción.

Sin embargo, se ha preferido optar por diseñar una arquitectura en el servidor más distribuida y estructurada, por lo que Django y Tornado serán lanzados por servidores independientes situados por detrás de un proxy, que será la puerta de acceso de las peti-

ciones de los clientes al servidor.

La decisión de establecer una arquitectura distribuida tiene muchas ventajas. Repartir la carga de trabajo entre diversos módulos internos redundará en un mejor servicio del sistema, ya que el trabajo será repartido de forma que no será soportado por un único módulo, lo que provocará mayor velocidad en las respuestas y menor colapso.

En línea con lo anterior, las peticiones de ficheros estáticos tales como imágenes, hojas de estilo, ficheros de scripting, etc. también serán tratados por otro servidor individual, permitiendo que estas peticiones, que pueden requerir de mayor tiempo de respuesta, no supongan un inconveniente para otro tipo de peticiones como las generadas en el sistema de juego, en las que la velocidad de respuesta es fundamental.

De este modo, la arquitectura final del servidor web seguirá el siguiente esquema:

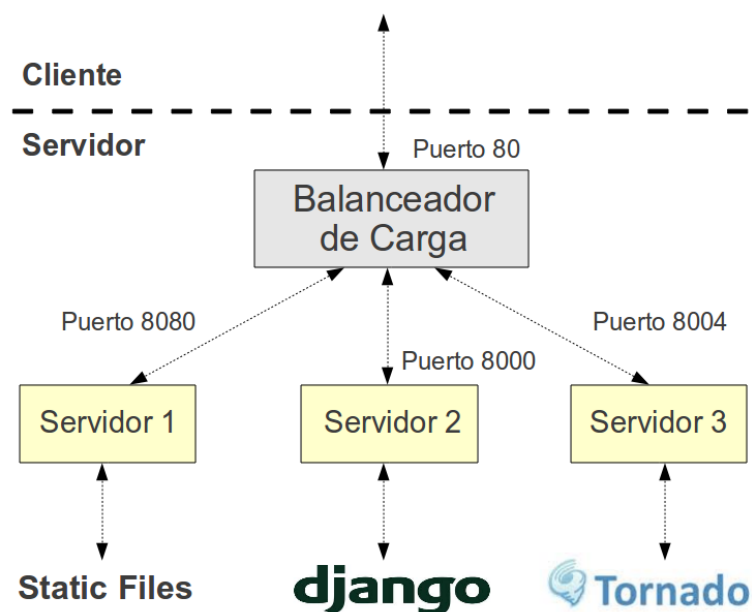


Figura 4.6: Arquitectura del servidor web

Este diseño permitirá que el servidor trabaje de forma más liviana y con mayor eficacia, ya que cada familia de recursos será servida por un módulo especializado en ese servicio. De este modo, los servidores elegidos para cada una de las ramas establecidas en el esquema anterior son:

- Servidor 1: este servidor debe atender peticiones de ficheros estáticos. Para este propósito, destaca por encima de todos el servidor Nginx, que será el elegido para

dar cobertura a este tipo de conexiones por su eficacia y su capacidad.

- Servidor 2: para servir la aplicación Django se va a utilizar el servidor Gunicorn, creado específicamente para aplicaciones desarrolladas en Python, que destaca por su rapidez, ligereza y un escaso consumo de recursos, en vez de utilizar el tan tradicional como pesado servidor Apache.
- Servidor 3: como se avanzó anteriormente, Tornado cuenta con su propio servidor web, el cual se va a emplear para dar servicio al módulo de comunicación de FLEQ.

Por último, queda por desvelar que aplicación será la elegida para trabajar como balanceador de carga en el servidor.

Este módulo, que estará tras el puerto 80 del servidor y dará acceso a las peticiones de los clientes a la plataforma, será el encargado de encaminar hacia el servidor interno adecuado cada una de las peticiones recibidas en función de su naturaleza.

Para este fin se ha escogido el servidor HAProxy, un proxy que es utilizado principalmente para esto, y que puede repartir carga entre varias máquinas físicas distintas de forma dinámica. Destaca por su rapidez, lo que ha sido tenido en cuenta para realizar esta elección.

Se ha configurado HAProxy para establecer los parámetros que le permitan diferenciar y enrutar las distintas conexiones, basándose en la detección de subcadenas y protocolos. De esta forma, HAProxy procede ante una petición de la siguiente manera:

- La primera regla establecida detecta si el protocolo de la URL no es HTTP (`http://`) sino WebSocket (`ws://`), o si la ruta del recurso comienza por `/socket.io`. Si se detectan cualquiera de estas dos situaciones, la petición que se está analizando es una conexión relativa al sistema de mensajes, por lo que es encaminada al puerto 8004, donde se encuentra escuchando el servidor de Tornado.
- La segunda regla determina, de manera análoga a la anterior, si la petición hace referencia a algún fichero estático (`/images`, `/css`, `/js`, `/admin/media`), encaminando estas peticiones hacia el puerto 8080 donde se encuentra el servidor Nginx.

- El resto de peticiones son encaminadas al puerto 8000, donde Gunicorn espera atender las relativas a la aplicación Django.

Tanto Nginx como HAProxy cuentan con un demonio o *daemon* que permite mantener activos los procesos relativos a su funcionamiento. Sin embargo, Gunicorn y Tornado no disponen de esta funcionalidad, por lo que se ha tenido que utilizar un sistema de control de procesos en Linux. Para este fin se ha empleado Supervisor, un *daemon* que permite controlar el estado de procesos.

De esta forma, ambos servidores permanecerán activos en segundo plano, de la misma forma en que lo hacen Nginx y HAProxy por medio de sus *daemons*.

Se ha configurado Supervisor para que mantenga ambos procesos siempre activos, por lo que ante cualquier problema que pueda ocurrir durante la ejecución de estos, se tratará de restablecer el servicio, garantizándose así un funcionamiento automático del sistema.

4.3. Diseño e implementación del cliente

Tras haber realizado un análisis en profundidad del lado del servidor que compone esta aplicación web, se va a proceder ahora a desgranar la arquitectura e implementación de la interfaz de usuario, mediante la cual se permitirá a los usuarios interactuar con el sistema.

4.3.1. Introducción

El cliente de la aplicación que está siendo abordada en el presente Proyecto Fin de Carrera está compuesto por dos implementaciones independientes, tal como se ha ido explicando a lo largo de esta memoria.

El objetivo principal del proyecto era hacer accesible la aplicación desde dispositivos móviles, para lo cual parte de las adaptaciones que ya se han abordado se antojaban necesarias.

Estas dos variantes del cliente pretenden cubrir las necesidades de dos grupos de usuarios distintos, o mejor dicho, dos situaciones de uso diferentes: el uso de FLEQ desde

un equipo de escritorio y desde un dispositivo móvil.

Para la primera de ellas FLEQ ya contaba con una interfaz, creada para el anterior sistema de juego y que contaba con un módulo de IRC empotrado que trataba de simular una integración de la aplicación que en realidad no existía.

Por otro lado, el desarrollo de un cliente móvil partirá desde cero ya que no existía ninguna implementación previa, lo cual es uno de los principales motivos que han llevado a que esté pudiendo escribir estas líneas.

4.3.2. Diseño del cliente

Tras realizar un análisis intenso de la anterior interfaz de escritorio de FLEQ, se decidió acometer una reestructuración tanto de forma como de contenido.

En un principio, ésta no era una de las prioridades, ni siquiera estaba en el libro de ruta del proyecto inicialmente, pero las primeras implementaciones del sistema de comunicación tuvieron como marco de prueba esta interfaz previa, lo cual descubrió varias necesidades:

- Interfaz poco atractiva y no adaptada a los nuevos mecanismos de diseño.
- Información dispersa y redundante.
- Dificultades de uso.
- Sistema de navegación complicado.
- Desorganización.
- Estructura creciente, que podía provocar desproporciones en el tamaño de ciertas secciones.

Todo esto provocó que se planificará una reestructuración del sitio, teniendo en cuenta además que si se quería obtener una aplicación web sólida, las distintas interfaces debían estar relacionadas, tanto en formato, como en estructuración, para que los usuarios no tuvieran que acometer dos aprendizajes distintos para usar el mismo sistema.

Este proceso de implementación se abordaría una vez terminadas las prioridades del proyecto, por lo que pasó a ser la última etapa del mismo, previa a la puesta en producción.

Sin embargo, su diseño si se consideró previamente puesto que como se ha dicho, éste debía estar relacionando con el diseño de la interfaz móvil que se desarrollaría previamente.

Teniendo como principal objetivo el desarrollo de una interfaz simple e intuitiva, se optó por una estructura básica que mantuviera partes de forma estática en los diferentes recursos, de forma que se permitiera una navegación sencilla por los principales *sites* de la aplicación, la cual se puede apreciar claramente en el siguiente esquema:

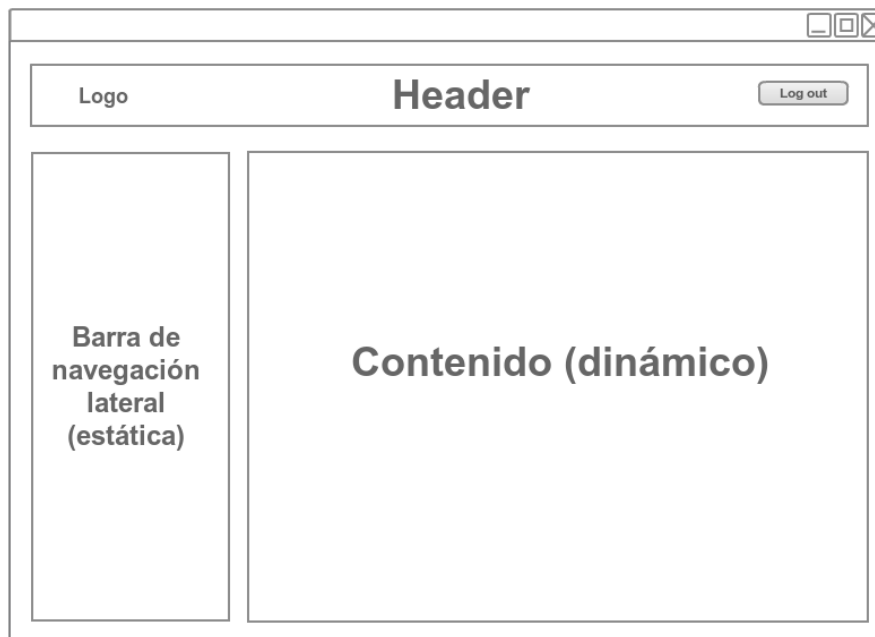


Figura 4.7: Diseño esquemático de la versión de escritorio

De esta forma, todos los recursos fundamentales están a un sólo *click*, sea cual sea el recurso actual.

El objetivo principal en el diseño de la interfaz móvil era mantener esa filosofía, siendo conscientes de las características de las pantallas de los dispositivos móviles.

Éstas son más reducidas, por lo que sería imposible mantener el diseño anterior y conseguir una aplicación que destaque por su usabilidad.

Por todo esto, se ha optado por una pantalla inicial formada únicamente por lo que sería la barra de navegación lateral en un equipo de escritorio, de modo que a través de cada uno de los enlaces se acceda a la información respectiva.

Los patrones básicos del diseño de ambas interfaces han quedado establecidos. Queda



Figura 4.8: Diseño esquemático de la versión móvil (I)

por tanto, determinar cuáles serán las opciones a las que podrá acceder un usuario de FLEQ a través de estos menús de navegación.

Vistas principales

Analizando las características del sistema, se han establecido dos grupos diferentes de opciones:

- *Games*: relacionadas con las partidas, sus configuraciones, estadísticas y ventanas de juego.
 - *Next games*: lista de próximas partidas programadas. Cada elemento de la lista es un enlace al *site* de la partida, que variará según su estado. Inicialmente, se mostrará la configuración de horarios. Cuando el horario haya sido fijado, este enlace dirige a la *chatroom* en la que se desarrolla el juego. Tras éste, se podrá consultar el *log* de la partida (solo desde la versión de escritorio).
 - *Won games*: lista de partidas ganadas por el usuario. Igualmente cada elemento

de la lista es un enlace, que dirige al log de la partida.

- *Lost games*: ídem a la anterior, pero con la lista de partidas perdidas por el jugador.
- *Tournaments*: gestión de torneos, visualización de estadísticas de los mismos, etc.
 - *My tournaments*: lista de torneos en los que participa, ha participado o participará el usuario. Cada elemento es un enlace a la página de estadísticas del torneo, en la cual se puede ver la clasificación, los resultados de las partidas clasificados por rondas y la información del torneo.
 - *Active tournaments*: lista de torneos actualmente en curso. Enlaces a los *sites* de estadísticas de los mismos.
 - *Next tournaments*: lista de próximos torneos programados. El usuario puede registrarse en estos torneos accediendo a la página de estadísticas del mismo. Igualmente, puede anular su suscripción. Además, se puede consultar la lista de jugadores apuntados y la información del torneo.
 - *Finished tournaments*: ídem a *Active Tournaments*, pero de torneos ya finalizados.

De los dos grupos de opciones anteriores, *Games* y *Tournaments*, se deducen siete diferentes alternativas dentro de la aplicación.

Como se indicó anteriormente, en un dispositivo móvil no se podrá mostrar en paralelo la barra de navegación y el contenido del recurso, por lo que se dotará a todos los recursos de esta web móvil de un enlace de retorno a la ventana principal en la cabecera de la página.

Además, se incluirá una pequeña barra horizontal de pestañas que serán enlaces directos a recursos relacionados. Por recursos relacionados se entienden los del mismo grupo al que pertenece éste, es decir, si se está navegando por *Next Games*, la barra de navegación mostrará enlaces a los sitios del grupo *Games*, y de forma análoga para los recursos de la familia *Tournaments* (ver figura 4.9).



Figura 4.9: Diseño esquemático de la versión móvil (II)

Usuarios

En FLEQ existen dos tipos de usuarios, según los permisos que posea: usuario básico y usuario administrador.

- Usuario básico: puede unirse a torneos y participar en ellos.
- Usuario administrador: además de los permisos básicos, puede organizar torneos y subir preguntas al sistema. Para conseguir un usuario con este tipo de privilegios, habría que contactar con los encargados de mantenimiento del proyecto, ya que no existe posibilidad de crear una cuenta de administrador sin su autorización.

Un usuario administrador tiene permisos especiales en FLEQ que le permiten, básicamente, crear nuevos campeonatos. Para ello, disponen de dos recursos adicionales:

- *New tournament*: mediante esta opción, se accede a un formulario en el que configurar un nuevo torneo. El administrador debe determinar el número de rondas, el

número de días por rondas, las categorías de preguntas que van a utilizarse en las partidas, una breve descripción del torneo y el tipo de acceso que se va a permitir para las partidas, pudiendo restringirse a dispositivos móviles seleccionando la opción *Only mobile devices*.

- *Load questions*: el administrador podrá subir al sistema nuevas preguntas clasificadas por categorías, que posteriormente serán utilizadas en los torneos. El mecanismo para introducir preguntas es mediante un fichero de texto plano, cuyas líneas contienen la información de las preguntas con el siguiente formato:

```
categoría;pregunta;respuesta;[[alternativa_1];[alternativa_2];[alternativa_3];]
```

Estas opciones son accesibles mediante el menú de navegación lateral, que incorpora dos nuevos enlaces para usuarios administradores.

Mediante la aplicación móvil un usuario básico puede realizar las mismas acciones que desde un equipo de escritorio, desde registrarse hasta seleccionar las horas de preferencia de las partidas, y por supuesto, participar en ellas.

Sin embargo, las funciones especiales de las que dispone un usuario con permisos de administración no son accesibles desde esta interfaz, ya que se ha considerado que no aportaba suficiente valor a la aplicación como para invertir tiempo en su implementación.

4.3.3. Implementación cliente móvil

La interfaz móvil ha sido desarrollada haciendo uso de *jQueryMobile*, un potente *framework* para el desarrollo de web móviles basado en HTML5, y que ya fue analizado ampliamente en el segundo capítulo de esta memoria.

Mediante *jQueryMobile* se conseguirá crear una web móvil con un *look and feel* extraordinario, que nada tiene que envidiar al diseño que se podría haber conseguido mediante desarrollos nativos.

El uso de este *framework* ha permitido el ahorro de grandes tiempos de desarrollo, puesto que el diseño de la interfaz se ha realizado una única vez, es decir, una implementación es válida para cualquier dispositivo móvil, sean cuales sean las dimensiones de las pantallas.

De haber realizado desarrollos nativos, conseguir estos resultados habría supuesto un gran aumento del tiempo de desarrollo, lo que hubiera hecho el proyecto inviable tanto por recursos humanos como por recursos temporales.

En las siguientes representaciones se podrá observar como la interfaz destaca por su orden y simplicidad, permitiendo un uso sencillo incluso para los usuarios más inexpertos.

Acceso

La ilustración (figura 4.10 (izqda)) recoge el aspecto que presenta la pantalla inicial de FLEQ, compuesta por un formulario de acceso y un botón que permite al usuario crear una nueva cuenta en la aplicación (figura 4.10 (dcha)).

The figure consists of two side-by-side screenshots of the FLEQ application interface. The left screenshot shows the login screen. At the top, the 'fleg' logo is displayed in blue. Below it is the tagline: 'A synchronous online competition to motivate and improve learning'. There are two input fields: 'username' and 'password'. Below these is a blue button with a star icon and the text 'Log In'. Underneath is the text 'Create a new account?' followed by another blue button with a star icon and the text 'Sign In'. At the bottom, it says '© FLEQ - 2012'. The right screenshot shows the registration screen. At the top, it says 'Complete the form below and get a new user account to enjoy fleg'. There are five input fields: 'username', 'first name', 'last name', 'password', and 'repeat password'. Below these is the text 'Introduce an email account that you usually use, here you will receive useful FLEQ notifications' followed by an 'email' input field. At the bottom is a blue button with a star icon and the text 'Register'.

Figura 4.10: Acceso (izqda) y registro (dcha)

Ventana principal

Basada en las premisas de diseño que fueron establecidas en la sección anterior, la ventana principal de la aplicación está compuesta por una lista de enlaces directos a los principales *sites* de la aplicación.

Estos han sido divididos en base a los dos grupos que fueron analizados en la sección de diseño, formando dos listas individuales: *Games* y *Tournaments*.

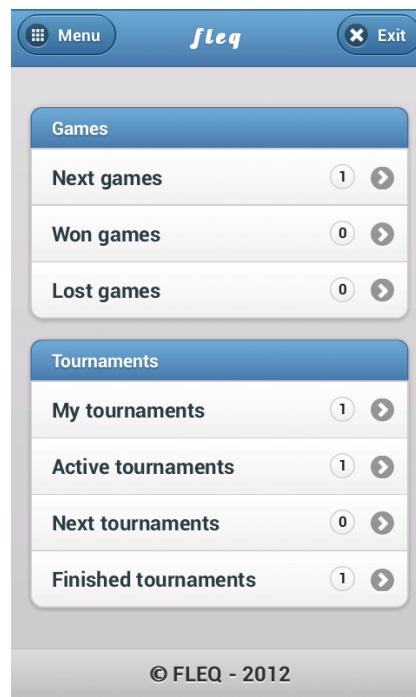


Figura 4.11: Menú principal

La apariencia de esta ventana principal así como del resto de páginas demuestran que mediante *jQueryMobile* se pueden obtener resultados muy profesionales.

Tournaments

La presentación de los torneos disponibles en la aplicación se realiza por medio de listas *customizadas*, como se puede observar en la figura 4.12 (izqda).

En ella se puede apreciar como cada ítem de la lista presenta la información más relevante de cada torneo: fechas de inicio y fin, número de rondas, días por ronda y temática de las preguntas (categorías).

La participación en torneos puede ser gestionada completamente desde la versión móvil de FLEQ.

Los usuarios pueden registrarse/borrarse en los torneos (figura 4.12 (dcha)), conocer la tabla de clasificación (figura 4.13 (izqda)), los resultados de las partidas de todos los jugadores distribuidos por rondas (figura 4.13 (dcha)) y consultar la información que el profesor (administrador FLEQ) haya introducido sobre el torneo.

Observando la tabla de clasificación de un torneo, se puede apreciar como no aparecen

todos los datos de cada usuario. Se han suprimido los datos menos importantes (*Days of Reflection, Questions Won, Questions Lost*), para no recargar demasiado la tabla, representándose únicamente los datos fundamentales (*User, Games Played, Games Won, Games Lost, Points*).

Como se explicó en la sección dedicada al diseño de los clientes de FLEQ, la funcionalidad adicional de la que un administrador dispone para la creación de torneos y la subida de preguntas al sistema, no es accesible desde la interfaz móvil.

Este es el único punto donde existe diferencia entre ambas interfaces en cuanto a torneos, lo cual no se considera una carencia en la implementación ya que como se argumentó adecuadamente, no aporta demasiado valor. Especialmente la subida de preguntas mediante un fichero de texto, lo cual puede incluso resultar engorroso desde un terminal móvil.

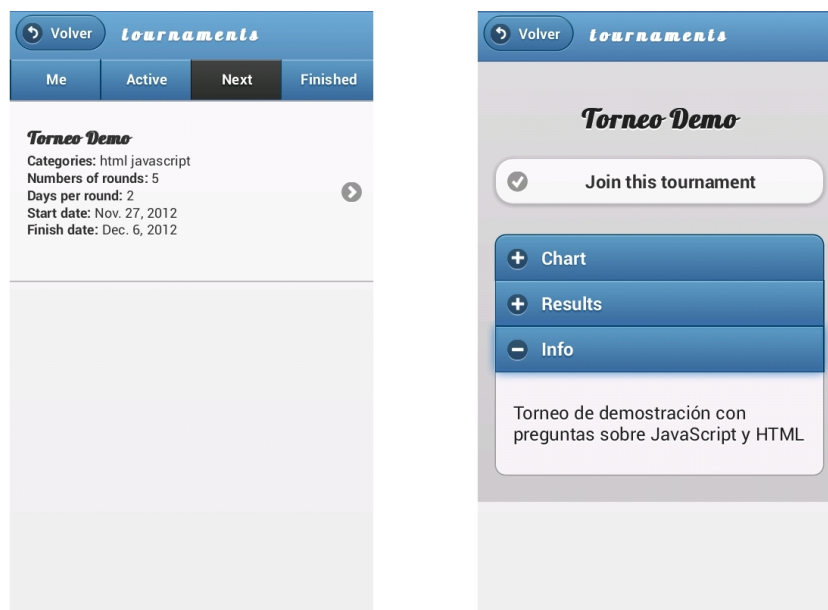


Figura 4.12: Gestión de torneos

Games

Al igual que la gestión de torneos, el manejo de los eventos relacionados con las partidas de los diferentes torneos pueden ser manejadas por completo desde la interfaz móvil de FLEQ.

Desde esta interfaz, el usuario puede igualmente acceder a su listado de partidas

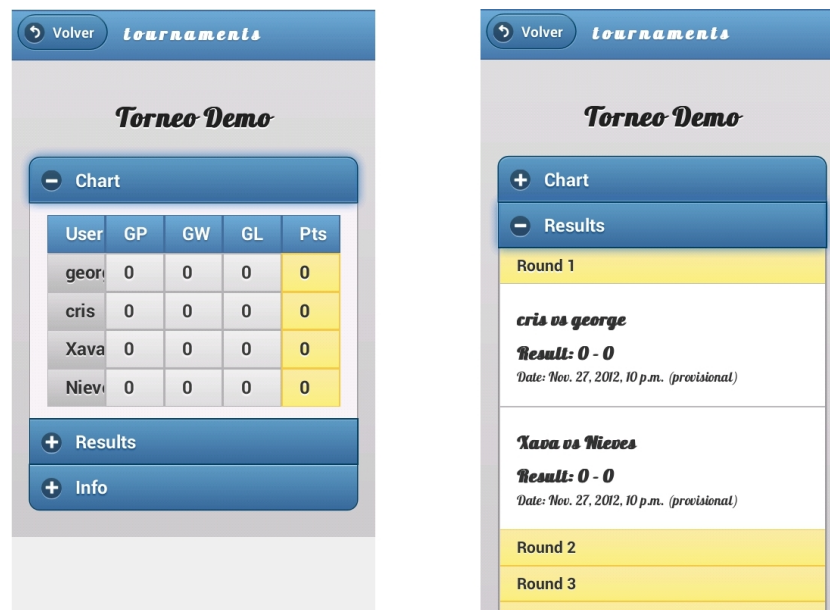


Figura 4.13: Clasificación (izqda) y resultados (dcha) de un torneo

programadas o consultar su histórico de *won/lost games* (figura 4.4 (izqda)).

En cuanto a la configuración de las partidas, el usuario puede seleccionar sus horas de preferencia tal y como se puede comprobar en la figura 4.14, anular preferencias previas o confirmar horarios propuestos por el adversario.

Como se puede observar, estas opciones se han implementado por medio de listas desplegables, de modo que la primera vista de la página deje claro al usuario la funcionalidad de la misma sin tener que recorrer largas listas, lo que habría ocurrido en el caso de que estos listados no se presentaran inicialmente ocultos.

Como en el caso de los torneos, el sistema de juegos también carece de una funcionalidad que sí se presta en la versión de escritorio. El log de las partidas no puede ser consultado desde un dispositivo móvil.

Las razones para ello son varias. La principal razón se deriva del flujo de información. Aunque estos ficheros de log no son muy grandes, su carga puede ser demasiado lenta si el usuario móvil no dispone de una buena conexión, lo que no daría una buena impresión de la aplicación a ojos del usuario.

Además, la representación de esta información en una pantalla de dimensiones reducidas puede no ser la mejor, ya que hay que tener en cuenta que además del propio mensaje

y el nombre de usuario, también se debe incluir la marca de tiempo para poder seguir la evolución de la partida.

Por todo esto, se optó por no incluir esta funcionalidad desde la interfaz móvil, aunque como se puede deducir, no habría supuesto ningún problema desde el punto de vista de desarrollo, ni siquiera en cuanto a tiempo.

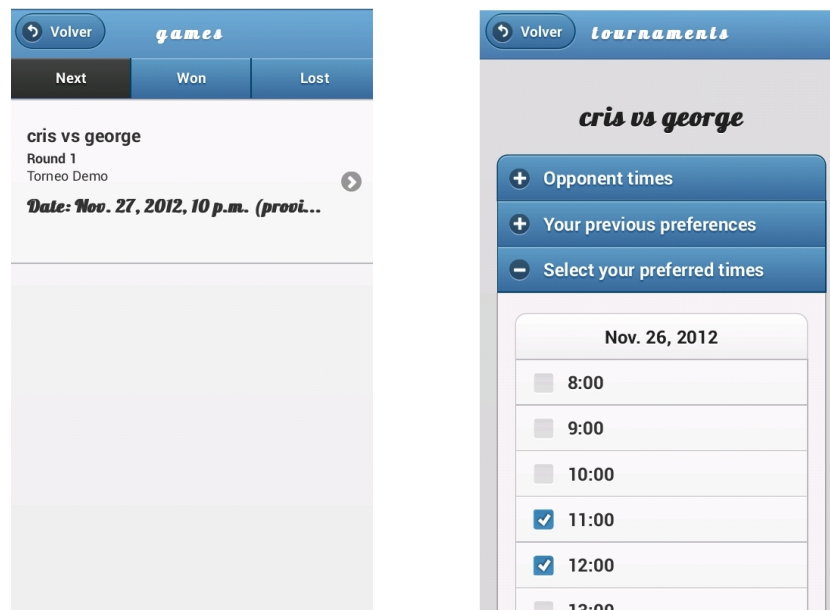


Figura 4.14: Configuración de partidas

Ventana de juego

Esta ha sido la parte en la que más cuidado se ha tenido que poner en la implementación, ya que conseguir una buena experiencia de usuario sería la clave del éxito de la aplicación.

Si jugar desde la aplicación móvil resultaba dificultoso, todo el esfuerzo empleado en el desarrollo de la interfaz móvil podría haber sido en vano.

Por eso, se han cuidado todos los detalles posibles, teniendo en cuenta que algunos de ellos dependerían de las características del navegador que utilice el cliente.

Como se puede apreciar en la figura 4.15, la ventana de diálogo o de juego ha sido dimensionada para que quede perfectamente acoplada con el teclado táctil de los dispositivos y evitar que al extenderlo se produzcan desplazamientos de ésta.

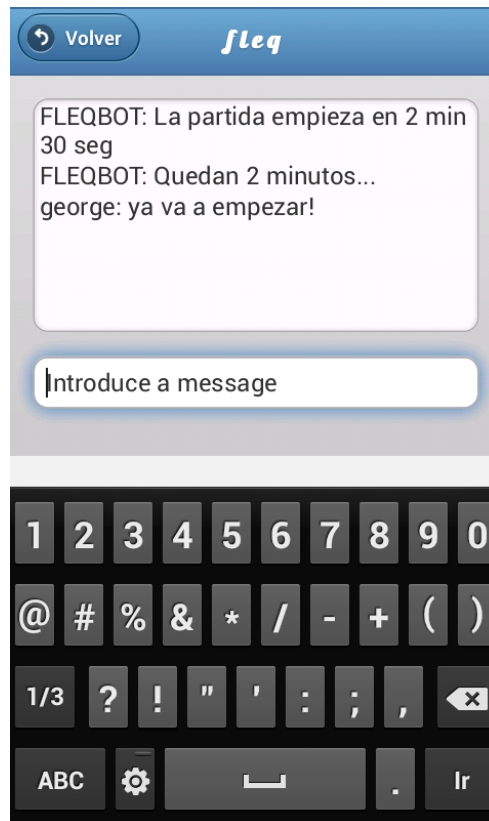


Figura 4.15: Ventana de juego

Las dimensiones de esta ventana de chat son fijas, es decir, no pueden crecer ni en altura ni en anchura. Su *scroll* está fijado para permanecer siempre al pie de la misma, de modo que los nuevos mensajes aparecen por la parte inferior y siempre están visibles, evitando que el usuario se tenga que preocupar de desplazar hacia abajo el visor de la ventana de juego.

El estado de la conexión viene indicado por el contenido del cuadro de texto. Si la conexión está establecida, el fondo de este cuadro será blanco y el mensaje impreso será *Introduce a message*.

Sin embargo, si la conexión no está establecida, bien porque se está procesando como ocurre al acceder a esta página, o bien porque se ha producido una desconexión, el fondo del cuadro de texto será rojo, y el mensaje impreso indicará el estado concreto (*Connecting* o *Reconnecting*).

De esta forma, el usuario es consciente del estado de su conexión, lo cual evita posibles confusiones que pueden resultar muy negativas en el transcurso de una partida.

4.3.4. Implementación cliente de escritorio

La implementación de la interfaz de cliente para equipos de escritorio ha supuesto el último paso en el desarrollo de este proyecto, previo a la puesta en producción y a la elaboración de la documentación.

En su implementación se ha optado por un diseño simple pero moderno, que permita al usuario una navegación cómoda y agradable por la aplicación.

Para ello se han utilizado hojas de estilo basadas en la última versión de CSS, es decir, CSS3. A su vez, para la elaboración de listas se ha utilizado parte de la funcionalidad de *jQueryMobile*, a pesar de que este framework esté orientado al desarrollo de webs móviles.

Este uso se debe a varios motivos. El primero de ellos es su innegable calidad en cuanto al diseño, que permite el desarrollo de listas con un *look and feel* muy profesional.

Y el segundo, viene dado por la sencillez, ya que se contaba con experiencia previa, puesto que anteriormente se habría desarrollado la interfaz móvil haciendo uso de este framework de desarrollo.

Obviamente, hubo que introducir modificaciones en algunos paquetes para evitar comportamientos extraños de la página web.

Como ejemplo de esto, *jQueryMobile* bloquea el uso del *scroll* horizontal, es decir, no es posible desplazarse por una página horizontalmente, ya que se busca que las páginas estén ajustadas al ancho de la pantalla y sean estáticas en este sentido.

Si haciendo uso de este *framework*, se minimizaba el navegador desde un equipo de escritorio, era imposible desplazarse horizontalmente, por lo que parte del contenido de la página era inaccesible sin redimensionar la pantalla.

Una vez realizados éste y otros cambios que permitieran que *jQueryMobile* se adaptara a los requisitos buscados, se pudo realizar el desarrollo de la nueva interfaz de escritorio de FLEQ.

El resultado de este trabajo se puede apreciar en la figura 4.16, *splash* de la aplicación web en el que aparece un mosaico de dispositivos móviles con la aplicación en pantalla, lo que deja constancia del salto evolutivo que FLEQ ha dado.

Las funcionalidades de esta interfaz son las mismas que desde la interfaz móvil para un usuario básico, por lo que no merece la pena saturar la memoria con explicaciones reiterativas.

Sin embargo, el usuario administrador si merece una explicación más profunda, ya que es en esta interfaz en la que puede configurar y crear nuevos torneos, como ya se ha adelantado.

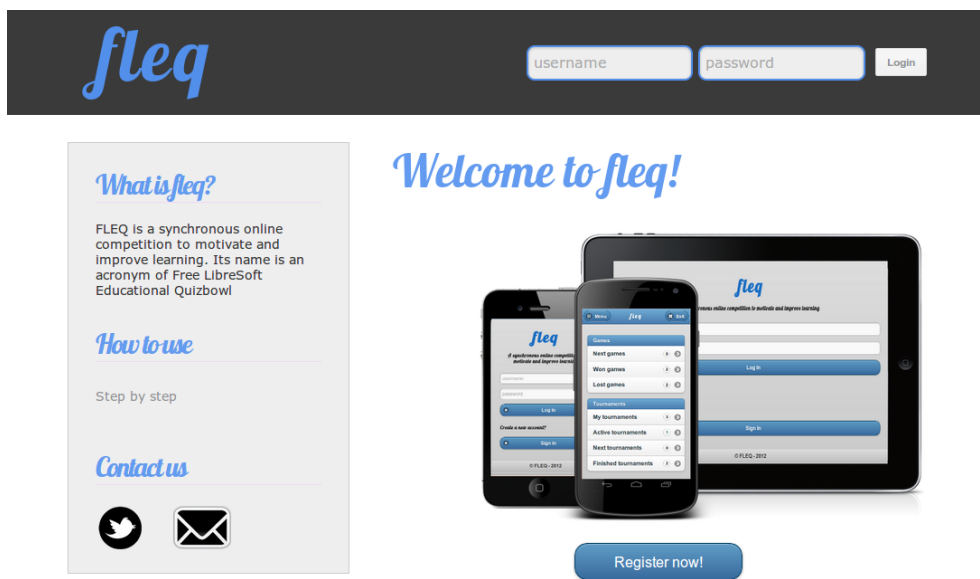


Figura 4.16: Página de inicio

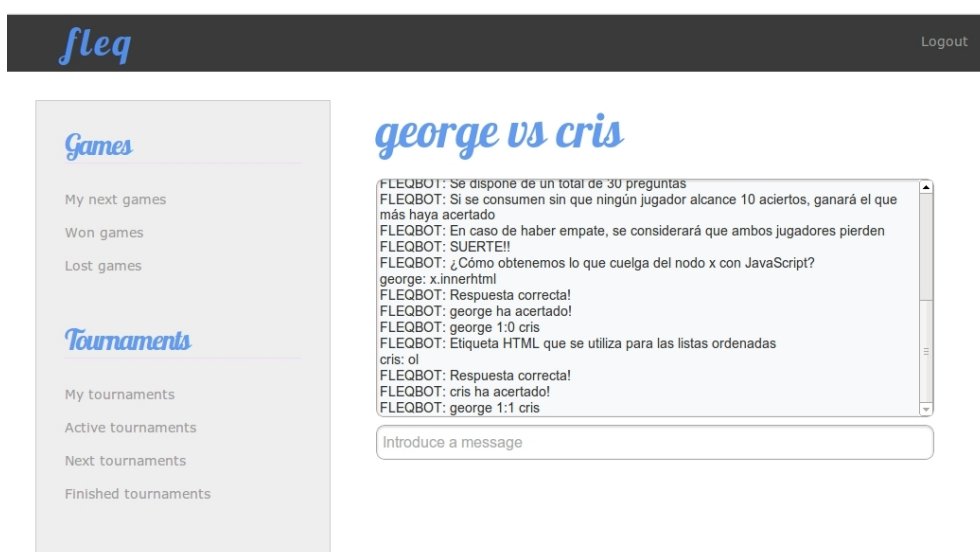
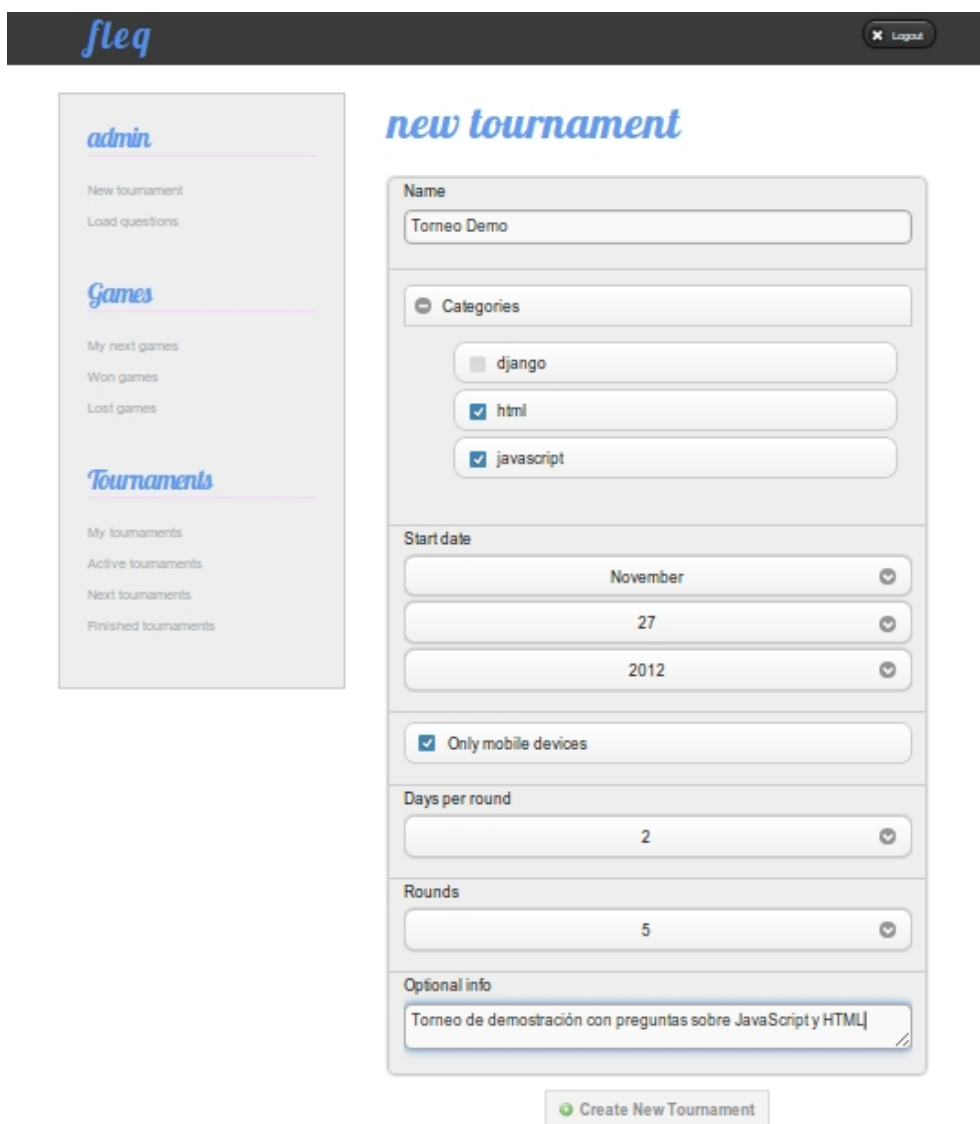


Figura 4.17: Ventana de juego

Administración

La interfaz de un usuario administrador presenta el aspecto que se puede ver en la figura 4.18, donde se observa el formulario al que se enfrenta este tipo de usuarios para configurar un nuevo torneo.

Así mismo, atendiendo a la barra de navegación lateral, se comprueba como ésta presenta dos nuevas opciones: creación de torneos (figura 4.18) y subida de preguntas (compuesta de un simple formulario para cargar el fichero correspondiente).



The screenshot displays the 'fleq' administration interface. At the top, there is a dark header with the 'fleq' logo on the left and a 'Logout' button on the right. A left sidebar contains navigation links under three sections: 'admin' (New tournament, Load questions), 'Games' (My next games, Won games, Lost games), and 'Tournaments' (My tournaments, Active tournaments, Next tournaments, Finished tournaments). The main content area is titled 'new tournament' and contains a form with the following fields:

- Name:** A text input field containing 'Torneo Demo'.
- Categories:** A section with a minus sign icon and three checkboxes: 'django' (unchecked), 'html' (checked), and 'javascript' (checked).
- Start date:** Three dropdown menus for month (November), day (27), and year (2012).
- Only mobile devices:** A checkbox that is checked.
- Days per round:** A dropdown menu set to 2.
- Rounds:** A dropdown menu set to 5.
- Optional info:** A text area containing 'Torneo de demostración con preguntas sobre JavaScript y HTML'.

At the bottom of the form is a 'Create New Tournament' button with a green plus icon.

Figura 4.18: Formulario para la creación de un nuevo torneo

Capítulo 5

Despliegue y resultados

El mundo exige resultados. No le cuentes a otros tus dolores del parto. Muéstrales al niño.

Indira Gandhi

Una vez realizado todo el desarrollo, llegó el momento de poner en funcionamiento FLEQ para poder testear el rendimiento del sistema al que se ha dado vida con tanto esmero y tantas horas de esfuerzo durante los últimos meses.

5.1. Introducción

Indudablemente, antes de llegar a este punto en el que FLEQ ha podido ser utilizado en un escenario real con usuarios externos al proyecto, se han realizado innumerables pruebas tanto de los módulos individuales como de todo el conjunto.

Desde aquel primer test en el que se montó un simple sistema de chat basado en el tradicional método de *polling* y que rápidamente denotó la necesidad de algo mejor, hasta las últimas pruebas que se realizaron con personas cercanas o con el propio tutor en un mini-torneo.

Todo ha servido para detectar puntos de acción y convertir FLEQ en lo que los alumnos de la asignatura Servicios y Aplicaciones Telemáticas (SAT) de la ETSIT han podido disfrutar durante diez días, en un apasionante torneo relacionado con esta asignatura.

5.2. Pruebas internas

Entre las pruebas internas que se llevaron a cabo, especial mención requiere la que se realizó previamente al campeonato final en SAT.

En este campeonato se convocó a personas de mi entorno familiar (hasta un total de cuatro participantes), para testear la herramienta en un torneo con contenido relacionado con parentescos familiares y preguntas sobre temas de actualidad.

Este torneo demostró que FLEQ puede ser utilizado con fines de ocio, ya que se consiguió pasar un rato agradable y fue realmente entretenido.

Supuso un hecho muy gratificante, ya que además de corroborar que la herramienta funcionaba correctamente, se pudo comprobar como algo que ha monopolizado tu vida durante varios meses puede aportar un valor añadido, y ha servido al menos para pasar un buen rato (independientemente de los resultados del torneo de SAT, que vendrían posteriormente).

Los comentarios vertidos por los participantes sirvieron para retocar pequeños detalles antes del campeonato en la Universidad, relacionados principalmente con visualización de la información en la pantalla de los dispositivos móviles y con aspectos de diseño.

El sistema de juego no dio ningún problema, únicamente alguna desconexión, normal teniendo en cuenta que la conexión utilizada por todos fue red móvil en vez de WiFi, y ante lo que poco se puede hacer por nuestra parte.

Aún así, el sistema de reconexión que Socket IO implementa permite que este problema reduzca su efecto considerablemente, con lo que la participación del usuario en el juego apenas se ve afectada.

Este sistema de juego si había provocado algún comentario en la prueba que se realizó con el tutor, que sirvió para descubrir problemas más significativos:

- Desconocimiento del estado de la conexión: este punto fue solventado mediante los indicadores de color en el cuadro de texto, como se comentó en el capítulo anterior. De esta forma el usuario es consciente del estado de su conexión, deduciendo que no está conectado a través del sombreado rojo del cuadro de texto.

- Problemas de conexión en Android 2.1 Eclair: el dispositivo móvil que utilizó el tutor del proyecto para testear la herramienta utilizaba este sistema operativo, lo que dejó patente problemas con la interfaz móvil. No podía siquiera *loguearse* en la aplicación desde el navegador, algo realmente extraño. El dispositivo que yo utilizaba era Android 4.0 ICS y no daba ningún problema. Para resolver el enigma, se crearon emuladores de Android, de las versiones 2.1 y 2.2. En esta última no pasaba, con lo que se añadía incertidumbre al problema. Y desde 2.1, accediendo mediante la interfaz de escritorio por el navegador del emulador, este problema tampoco ocurría. Con lo cual, estaba claro: el problema estaba provocado por alguna mala configuración de *jQueryMobile*. Tras un intenso análisis, se comprobó que había un conflicto entre dos paquetes de este framework, que impedían que el navegador por defecto de esta versión de Android fuera capaz de procesar el envío del formulario de login.

Aunque esta versión de Android está prácticamente en desuso (menos del 4% según datos de Google Play), se consideró necesario invertir tiempo en su corrección ya que el error venía de nuestra parte, no del SO o del *browser*.

De las líneas anteriores se extrae que los dos terminales utilizados en este mini-torneo eran Android. Sin embargo, en el torneo familiar con el que se comenzó esta sección, se contó con la participación de un iPhone, lo que sirvió para comprobar que desde estos dispositivos el comportamiento de FLEQ era el esperado.

Los problemas surgidos con la versión 2.1 de Android nos pusieron sobre aviso, por lo que se decidió realizar pruebas de forma general, con emuladores de los principales sistemas operativos que podían ser utilizados en siguientes pruebas para evitar posibles problemas como el que nos ocupa.

Por esto, otro emulador de Android fue utilizado para comprobar el correcto funcionamiento de los sistemas operativos destinados anteriormente a las tablets (Android 3, actualmente tanto tablets como smartphones utilizan Android 4).

El funcionamiento de FLEQ en BlackBerry y Windows Phone fue testeado también con emuladores de estos sistemas, desde sus respectivos SDKs en el sistema operativo Windows de escritorio.

Después de todas estas pruebas, en las que se ha testeado adecuadamente la aplicación en mini-torneos y se ha comprobado su funcionamiento en dispositivos de las cuatro principales familias de *smartphones*, se estaba en disposición de avanzar a la siguiente fase en la cual FLEQ sería utilizado en el entorno real ya mencionado anteriormente.

5.3. Prueba real: SATpeonato

El torneo de la asignatura SAT fue bautizado como SATpeonato por el tutor del presente Proyecto Fin de Carrera y profesor de dicha asignatura.

Las modificaciones que se tuvieron que acometer después de las pruebas internas analizadas anteriormente, estuvieron terminadas a fecha de 31 de octubre de 2012, por lo que el SATpeonato podría haber empezado durante la primera semana del mes de noviembre.

Sin embargo, problemas derivados de las ocupaciones del tutor provocaron que su organización se tuviera que retrasar hasta la segunda quincena de ese mismo mes.

Finalmente, el torneo pudo ser creado el domingo 11 de noviembre, tras lo que se sucedieron los siguientes hitos:

- 11-Noviembre: creación del SATpeonato en FLEQ.
- 11-16 Noviembre: periodo de inscripción.
- 16-25 Noviembre: desarrollo del SATpeonato.
- 25-27 Noviembre: periodo de evaluación mediante encuestas online.
- 28-29 Noviembre: análisis de resultados.

Las características del torneo fueron:

- Número de rondas: 5.
- Días por rondas: 2.
- Categorías: HTML, JavaScript y Django.
- Acceso limitado a dispositivos móviles.

Participantes

Éramos conscientes de que las fechas para la realización de una actividad extra en la asignatura no eran las más adecuadas, ya que los exámenes del primer cuatrimestre del curso estaban muy próximos.

Aún así, la acogida del torneo fue buena y finalmente se registraron 10 participantes.

Esta cifra supone aproximadamente un 25 % del total de alumnos de SAT, basándose este porcentaje en el número de usuarios registrados en la web de la asignatura en Moodle (51, de los cuales 2 son profesores, otro es el propio autor del proyecto, y algunos otros son alumnos de años anteriores que no se han borrado del curso), lo cual consideramos que es una cifra de aceptación muy buena, teniendo el problema derivado de las fechas que ya se ha comentado anteriormente.

	R1	R2	R3	R4	R5
Participantes	9	8	8	7	8

Cuadro 5.1: Evolución de la participación en el torneo

Como se puede observar en el cuadro 5.1, la participación de los alumnos en el campeonato ha sido razonablemente buena, contando todas las rondas con al menos 7 participantes. De este modo, la media de participación global ha sido del 80 %.

Contenido

El torneo fue definido con una batería de 90 preguntas elaboradas por el tutor y ajustadas al contenido de la asignatura (HTML, JavaScript y Django).

Entre estas preguntas, cabe destacar que su nivel de dificultad era medio-alto, con el objetivo de que el alumno percibiera en las primeras rondas sus carencias en estos ámbitos.

La figura 5.2 muestra la evolución de aciertos que ha habido en las diferentes rondas del torneo de forma global, considerando como pregunta no acertada aquella en la que se ha consumido el tiempo disponible (90 segundos) sin que ningún jugador haya sido capaz de dar una solución correcta.

Por tanto, los aciertos serán aquellas preguntas que sí han sido respondidas adecuada-

mente por alguno de los dos contrincantes, siendo el porcentaje representado el cociente de las preguntas acertadas y el total de preguntas lanzadas por ronda.

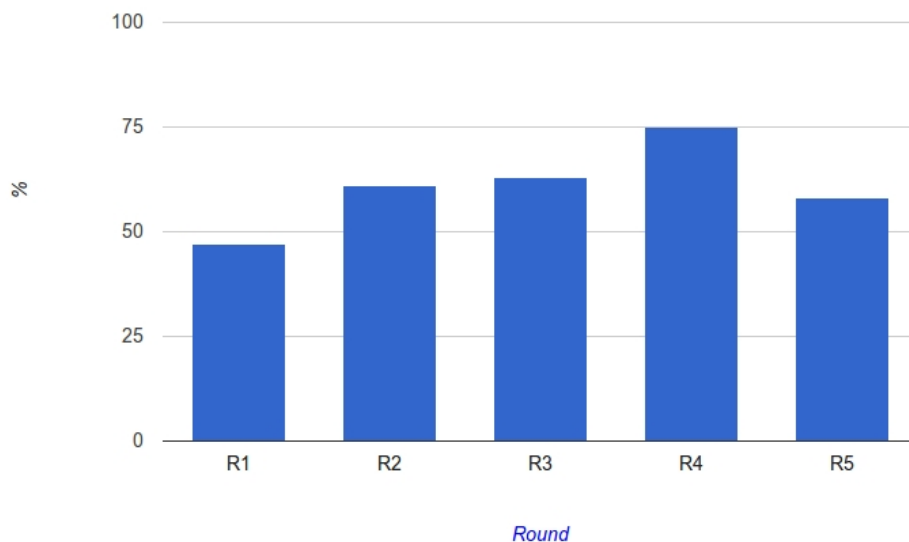


Figura 5.1: Evolución del porcentaje de aciertos en el torneo

Se puede observar como esta evolución ha sido creciente en las cuatro primeras rondas, cayendo en la última.

La explicación a este hecho es sencilla. Se quería comprobar si el torneo había conseguido que los alumnos mejoraran sus conocimientos sobre estos temas, fundamentales para la asignatura.

Considerando que cada partida podía lanzar un máximo de 30 preguntas, que se disponía de 90 en la base de datos y que ya habían transcurrido cuatro rondas, muchas preguntas comenzaban a ser repetidas, por lo que la comparativa se podía ver algo distorsionada.

Para realizar una comparativa real, se reemplazó este lote inicial de 90 preguntas por un lote final de 30 cuestiones nuevas, escritas en este caso por el autor del proyecto y con un grado mayor de dificultad que el primer lote.

De esta forma se podría establecer una comparación real entre los resultados de la primera y la última ronda, que habrían contado con preguntas totalmente independientes.

Como se puede apreciar, el porcentaje de aciertos en la última ronda fue mayor que en

la primera, por lo que se puede concluir que el torneo ha cubierto las expectativas iniciales y FLEQ ha cumplido con su cometido, aumentando este valor de un 47 % a un 58 %.

Encuestas de evaluación final

A la finalización del torneo, los participantes han completado una encuesta online vía Google Docs para evaluar el campeonato y FLEQ de forma completamente anónima.

Todos los participantes completaron este breve cuestionario compuesto por 21 preguntas en los dos días posteriores a la finalización del torneo.

Estas preguntas fueron redactadas concienzudamente para extraer la máxima información posible de los alumnos, y así detectar puntos de mejora que abordar en futuros proyectos.

Obviamente, estas preguntas se centraron mayoritariamente en asuntos relacionados con las novedades que ha incorporado este proyecto, como el nuevo sistema de juego y las interfaces.

Además, se incluyeron algunas otras relacionadas con el formato del sistema, las cuales han aportado mucha información útil para el futuro de la aplicación y han sugerido algunos de los puntos de acción que serán recogidos en el próximo capítulo.

A continuación se listan los resultados más frecuentes de cada una de las preguntas de esta encuesta:

- La interfaz móvil es fácil e intuitiva (escala Likert de 5 niveles): De acuerdo.
- Valoración del diseño de la interfaz móvil (de 0 a 5): 4.
- Sistema de cambio de hora adecuado (escala Likert de 5 niveles): De acuerdo.
- Sistema de alertas por mail adecuado (escala Likert de 5 niveles): Muy de acuerdo.
- Ritmo de juego adecuado (escala Likert de 5 niveles): De acuerdo.
- Sistema actual basado en WebSockets vs Sistema anterior basado en IRC: Sistema actual basado en WebSockets.
- Dispositivo móvil: Android.

- Conexión: WiFi.
- Navegador: Android Browser.
- FLEQ ha estimulado el estudio: SI.
- Mejora de conocimientos sobre HTML, Django y JavaScript: SI.
- Conocimiento previo de la API de WebSockets: NO.
- Participación anónima: NO.
- Incomodidad por competitividad: NO.
- Grado de satisfacción global (de 0 a 5): 4.

Estos han sido a grandes rasgos, los resultados de las preguntas más importantes de la encuesta.

Adicionalmente, se incluía un cuadro de texto donde los participantes pudieran incluir las sugerencias o comentarios que consideraran oportunos, de forma voluntaria. Estos comentarios han sido variados, y por supuesto, muy enriquecedores para el presente y futuro del proyecto:

- Alumno 1: “Algunas preguntas tenían una respuesta muy larga y que contenían caracteres especiales, y con el teclado del móvil resultaba complicado escribirla.”
“Empecé utilizando Android Browser que no soporta la API de WebSockets de HTML5 y luego usé Opera Mobile que si la soporta, la verdad es que funcionaba bastante mejor, era más rápido y se desconectaba menos.”
- Alumno 2: “La hora por defecto debería de ajustarse si uno de los 2 concursantes selecciona sus preferencias y el otro no.”
- Alumno 3: “Creo que sería interesante añadir la funcionalidad de que la última pregunta aparezca en el caso de quedarte sin conexión y volver a los pocos instantes”
- Alumno 4: “Los mails llegaban con poca antelación, y creo que sería conveniente avisar a los jugadores por mail cuando se establezca la hora del juego, no solo cuando esté a punto de comenzar.”

- Alumno 5: “Es un poco incómodo que sea una aplicación a través del navegador y en tiempo real, en mi opinión prefiero las aplicaciones por turnos que te avisan en la barra de notificaciones, porque al avisar por email media hora antes se me ha olvidado varias veces participar a mi hora.”
- Alumno 6: “El SATpeonato ha estado muy bien en general. El único inconveniente que le he visto es que, al ser sólo para dispositivos móviles, me ha sido más difícil escribir correctamente algunas respuestas, porque es fácil pulsar en una letra distinta que esté cerca de la que se quiere pulsar, y es más complicado corregir algo que ya se ha escrito. De todas formas tengo poca práctica escribiendo en móviles táctiles por lo que sería adecuado comprobar si sucede también con gente con más práctica.”

Estos comentarios han servido para detectar flaquezas de la aplicación, que quedarán recogidos en la sección del próximo capítulo dedicada a posibles mejoras.

Los resultados completos de estas encuestas pueden ser consultados en la sección de Apéndices.

Análisis de resultados

A la vista de todo lo expuesto anteriormente, los resultados arrojados tras el SATpeonato han sido muy satisfactorios, por lo que se puede afirmar que estamos muy contentos del trabajo realizado.

Como se ha podido comprobar anteriormente, la evolución del porcentaje de aciertos en el torneo ha sido creciente, lo cual nos ha permitido descubrir que el principal objetivo para el que se ha desarrollado esta herramienta se ha cumplido.

La comparación real entre los resultados de la primera y última ronda, en la que se contó con nuevas preguntas de mayor dificultad, así lo demuestra. Se ha pasado de un 47 % de aciertos con un lote de cuestiones de nivel medio-alto a un 58 % con preguntas de un nivel alto.

La evolución es evidente, por lo que creo que FLEQ ha cumplido su objetivo claramente, fuera de las características técnicas de su implementación y desarrollo.

No hay que olvidar que ese es el propósito de esta herramienta de e-learning, y en lo

que se basa su lema: *A synchronous online competition to improve and motivate learning.*

En ese eslogan queda reflejado claramente que el objetivo de FLEQ es mejorar y motivar el aprendizaje, lo que han confirmado los propios alumnos con sus respuestas mayoritarias a las preguntas correspondientes:

- FLEQ ha estimulado el estudio: SI.
- Mejora de conocimientos sobre HTML, Django y JavaScript: SI.

Por tanto, en el apartado de objetivos didácticos, se puede afirmar que FLEQ ha cumplido ampliamente.

Centrándonos ahora en los objetivos tecnológicos, considerando como tales la satisfacción global con la herramienta, con la interfaz móvil y con el uso en general, también se han recibido buenas críticas como recogen las respectivas respuestas:

- La interfaz móvil es fácil e intuitiva (escala Likert de 5 niveles): De acuerdo.
- Valoración del diseño de la interfaz móvil (de 0 a 5): 4.
- Grado de satisfacción global (de 0 a 5): 4.

Estos resultados, cercanos todos ellos a la máxima puntuación posible, demuestran que el apartado tecnológico de la aplicación ha cubierto las expectativas y se ha conseguido poner en producción un sistema fiable, amigable y, lo más importante, usable.

La usabilidad de FLEQ desde dispositivos móviles era uno de los puntos críticos a los que se debía enfrentar la herramienta, y creemos que los resultados han sido satisfactorios.

Es cierto que uno de los alumnos reclama en su *feedback* que algunas preguntas eran demasiado largas y con caracteres especiales, lo que dificultaba su respuesta.

Esto no fue al azar. En el primer lote de preguntas se incluyeron respuestas difíciles para determinar los límites en este tipo de torneos en los que sólo se permite acceso desde dispositivos móviles.

Se ha comprobado que el usuario puede sentirse incómodo ante este tipo de respuestas tan difíciles de escribir desde un teclado táctil, por lo que se recomendará incluir respuestas más comedidas en torneos para móviles como lo era el SATpeonato.

Fuera de estos pequeños apuntes, FLEQ no ha dado ningún problema serio. Todas las partidas han transcurrido de forma adecuada, sin ningún contratiempo, lo que demuestra que FLEQbot es un sistema sólido con garantías totales en su funcionamiento y el sistema de comunicación implementado es reconocido por todos los participantes del SATpeonato como un gran avance respecto al anterior sistema basado en IRC.

Por todo esto, podemos concluir que los resultados del torneo han sido muy positivos, tanto para este Proyecto Fin de Carrera en particular como para FLEQ de forma general.

Capítulo 6

Conclusiones

*Solamente aquel que construye el futuro
tiene derecho a juzgar el pasado.*

Friedrich Nietzsche

Este capítulo cierra esta memoria, y con ello un proyecto que ha supuesto un gran salto tecnológico para FLEQ.

Llegados a este punto es necesario hacer balance de los hitos alcanzados, por lo que este apartado comenzará con una evaluación de los objetivos marcados al comienzo del proyecto, continuando con una estimación del tiempo empleado para completarlo y un listado de posibles líneas de desarrollo que se han detectado durante el mismo.

Esta memoria se cerrará con una valoración final de carácter personal acerca del trabajo realizado y se aprovechará para indicar el proceso seguido en la liberación del código de la aplicación.

6.1. Análisis de objetivos

En el primer capítulo de esta memoria se presentaron las líneas maestras de este proyecto, entre cuyos objetivos principales se encontraban los siguientes:

1. Integrar el módulo de comunicación de FLEQ en la propia aplicación.
2. Desarrollar una aplicación móvil.
3. Optimizar la interfaz de usuario de la aplicación de escritorio.

Como se ha podido ir leyendo a lo largo de esta memoria, los tres objetivos claves enumerados en esta lista han sido cumplidos en este proyecto.

Analicemos individualmente cada uno de ellos para saber hasta qué punto se ha logrado alcanzar estos objetivos.

Objetivo 1. Integración del sistema de comunicación

El primero de ellos cita la integración del sistema de juego de FLEQ dentro de la propia aplicación, eliminando la dependencia de IRC.

Tanto por la importancia que supone para una aplicación su integración y no dependencia de sistemas terceros, como por el nivel de dificultad que ha supuesto este hito, es sin duda el objetivo más importante del presente PFC.

Así se ha demostrado en esta memoria, en la que ha adquirido un papel protagonista desde el principio.

Como se ha argumentado en varias ocasiones, la solución tecnológica adoptada para la resolución de este hito es la mejor posible.

Una solución que cubre los requisitos deseables para cualquier sistema o aplicación: universalidad, futuro y eficiencia. Estamos hablando obviamente de la librería Socket IO:

- Es universal porque implementa un conjunto de sistemas de comunicación bidireccional que permite dar cobertura a la práctica totalidad de navegadores instalados en todo el mundo.
- Es una herramienta de largo recorrido puesto que implementa WebSockets, la mejor y más moderna tecnología, que ha sido creada específicamente para cubrir este tipo de conexiones.
- Y por último, es eficiente precisamente por la utilización de WebSockets, un sistema de comunicación bidireccional que optimiza el consumo de ancho de banda en las comunicaciones y potencia la velocidad de las mismas.

Por todo esto y teniendo en cuenta las opiniones vertidas por los participantes del SATpeonato, que aplaudían la decisión de abandonar IRC y mostraban su satisfacción

con el nuevo sistema, se puede afirmar que el objetivo ha quedado cubierto de manera satisfactoria.

Objetivo 2. Desarrollo de una aplicación móvil

Este objetivo inicialmente nació con vistas al desarrollo de una aplicación nativa para dispositivos de la familia Android, y evolucionó a lo largo del proyecto como ya se ha explicado, hasta convertirse en el desarrollo de una web móvil, que permitiera que FLEQ pudiera ser accesible desde cualquier *smartphone* o *tablet*, fuera cual fuera su SO.

Para su desarrollo se utilizó el framework *jQueryMobile* que permitió obtener una web móvil con un *look and feel* de gran calidad, y un diseño muy intuitivo como así reflejan los resultados de las encuestas del SATpeonato:

- La interfaz móvil es fácil e intuitiva (escala Likert de 5 niveles): *De acuerdo*.
- Valoración del diseño de la interfaz móvil (de 0 a 5): *4*.

Una valoración de 4 sobre 5 en cuanto a diseño y *De acuerdo* (segunda mejor valoración en una escala de 5 opciones) en cuanto a facilidad de uso, por lo que se puede considerar que el objetivo ha sido cumplido de manera óptima.

Además, con esta solución se ha logrado cubrir la práctica totalidad del mercado actual de *smartphones* y *tablets*, con lo que se ha superado con creces el objetivo inicial del proyecto con el que únicamente se hubiera cubierto el sector de usuarios de Android.

Objetivo 3. Optimización de la interfaz de escritorio

La interfaz de escritorio ha sido reconstruida por completo, tanto su formato como su contenido y estructura.

De esta forma se ha conseguido una interfaz mucho más modernizada, para lo que se han utilizado las últimas técnicas de diseño web basadas en HTML5 y CSS3.

Igualmente, se ha establecido un flujo de navegación más sencillo que permite al usuario un uso de la aplicación web menos engorroso. La información es presentada de forma clara y concisa, por lo que la satisfacción del usuario ha crecido notablemente.

Por tanto, se puede concluir que este objetivo principal del proyecto también ha sido cubierto de forma satisfactoria.

Objetivos secundarios

Además de estos objetivos principales ya analizados, el proyecto contaba con una serie de objetivos secundarios que también merece la pena analizar, principalmente porque alguno de ellos puede haber sido descuidado como consecuencia de las necesidades de los anteriores.

Entre los objetivos secundarios marcados al inicio de la memoria, destacan:

- Reestructurar el modelos de datos.
- Crear una arquitectura escalable.
- Mejorar la experiencia de usuario.

La reestructuración del modelo de datos fue llevada a cabo con prioridad 2, lo que se ha traducido en que no haya sido todo lo profunda que podría haber sido.

Aún así, esta reestructuración ha existido, pero siendo autocríticos, se podría haber optimizado más, por lo que este objetivo secundario lo debemos catalogar como parcialmente cubierto.

En cuanto a la escalabilidad de la arquitectura, cabe decir que este otro objetivo sí ha sido ampliamente cubierto, ya que el simple uso de Django como *framework* de desarrollo ha dotado al sistema de una escalabilidad excelente.

En otro aspecto del sistema, el servidor web, la arquitectura obtenida también destaca por su escalabilidad, por lo que el sistema cumple con este objetivo de forma adecuada.

Por último, la mejora de la experiencia de usuario en el uso de FLEQ, que quizá debía considerarse como un objetivo principal, también ha sido ampliamente cubierta como así demuestran las opiniones vertidas por los participantes del SATpeonato (ver apéndices).

6.2. Estimación del trabajo realizado

Una vez alcanzado este punto se considera necesario realizar un balance del tiempo empleado en el desarrollo de este proyecto, para ayudar al lector a comprender el volumen de trabajo que ha supuesto la implementación de todo este engranaje.

Cabe mencionar que durante los meses de desarrollo se ha combinado esta actividad con el trabajo como becario que desde el mes de junio de este mismo año 2012, desarrollo durante 6 horas al día.

Por esto, el hito de poder presentar este proyecto en el mes de diciembre supone una gran satisfacción, ya que compaginar ambas tareas y haber podido desarrollar este proyecto es algo que me llena de orgullo, pues deja patente mi esfuerzo y sacrificio durante los últimos meses.

En función a la planificación expuesta en los apéndices, se va a cuantificar el trabajo realizado en número de horas de forma aproximada.

1. Estudio de objetivos (Julio-Agosto 2012): en esta fase de estudio, documentación y primeras pruebas, se empleó una media de 3 horas diarias, por lo que considerando 50 días de trabajo real, se pueden computar 150 horas de trabajo en esta primera fase.
2. Desarrollo (Septiembre-October 2012): esta ha sido la fase que más dedicación ha requerido, y supuso que el tiempo de descanso diario se viera reducido a la mínima expresión. En este periodo se ha estimado un trabajo diario de 6 horas de media, considerando que durante los fines de semana esta cifra era muy superior, ya que eran los días en los que se podía dedicar más tiempo. La cuenta de horas asciende por tanto, hasta 300 (considerando igualmente 50 días en los dos meses).
3. Pruebas reales y escritura de la memoria (Noviembre 2012): la realización de pruebas, en especial el SATpeonato, ha supuesto bastantes horas de trabajo, ya que se estuvo pendiente de que no hubiera problemas en todo momento, y se monitorizaron todas las partidas (25 en total) para comprobar el correcto funcionamiento de FLEQbot. Todo esto, junto con demás gestiones que hubo que realizar con los participantes, unido a la elaboración y análisis de las encuestas, supone que se puedan

imputar 20 horas de trabajo. Por otro lado, se estima en unas 80 horas (4 horas al día durante 20 días) la elaboración de esta memoria, lo que completa la cifra de 100 horas durante el mes de noviembre.

4. Últimos detalles (Diciembre 2012): esta fase compone la elaboración de la presentación, la preparación del código para su liberación y la conversión de esta memoria a formato de libro electrónico *epub*, lo que se estima que supondrá un total de 50 horas de trabajo aproximadamente.

Estas valoraciones son aproximadas, ya que no ha existido ningún control exhaustivo del tiempo.

Aún así queda reflejado como el periodo clave del proyecto ha sido el trimestre formado por septiembre, octubre y noviembre, especialmente los dos primeros meses.

El resultado de estas estimaciones realizadas por cada fase individualmente, hace un total de 600 horas de trabajo (aproximadamente) para el desarrollo de este Proyecto Fin de Carrera.

6.3. Futuras líneas de trabajo

El propio desarrollo del proyecto ha revelado posibles líneas de mejora de FLEQ, las cuales han sido completadas con la ayuda de las críticas vertidas por los participantes del SATpeonato.

Esto supone que FLEQ es una herramienta con un largo camino por recorrer, y que este proyecto no es más que uno de los pasos que se debía dar para mantener viva la plataforma.

Entre todos los puntos que pueden mejorar y complementar FLEQ, se destacan los siguientes, que pueden ser acometidos en futuros Proyectos Fin de Carrera:

- **Minería de datos:** la aplicación ha sido desarrollada teniendo en cuenta esta más que posible extensión futura. Por ello, todas las partidas que se disputan en FLEQ almacenan su desarrollo en un fichero de texto, como ya se explicó anteriormente.

Estos ficheros serían la base de un módulo estadístico que presente datos de forma gráfica, con el objetivo de que los organizadores de torneos puedan comprobar la eficacia del sistema, de forma similar a como se ha hecho en la evaluación de aciertos del SATpeonato. De este modo, se podrían obtener estadísticas tanto individuales como colectivas, que irían por ejemplo, desde el tiempo medio empleado para responder hasta el número de respuestas vertidas hasta conseguir el acierto. Para el desarrollo de este módulo se recomienda utilizar la librería *Google Chart Tools*, potente herramienta que permite la construcción de diagramas y tablas de forma sencilla y eficaz en HTML5.

Este sistema permitiría un comportamiento inteligente de los torneos, ya que mediante estas mediciones se podría descartar las preguntas más acertadas para posibles rondas, o seleccionar las preguntas con más dificultades para cada participante de forma individual.

- **Internacionalización:** FLEQ es una herramienta abierta y la idea inicial es que pueda ser utilizado indistintamente por cualquier usuario, desde cualquier parte del mundo. Actualmente, el contenido de FLEQ está escrito en inglés por su universalidad, pero sería conveniente que el usuario pudiera seleccionar su lengua de preferencia. Django cuenta con un módulo que facilita este proceso, *django.utils.translation*, del que hemos recibido buenas impresiones, por lo que instamos a futuros desarrolladores a su uso. En líneas con esta internacionalización de FLEQ, se antoja necesario revisar el sistema horario, de modo que la herramienta pueda mostrar a los clientes su hora local, realizando luego las correspondientes conversiones en el servidor. Para este hito se recomienda la utilización del módulo de Python *pytz*.
- **PhoneGap:** esta herramienta permite la conversión de web móviles a aplicaciones nativas añadiendo soporte a funciones que no son accesibles desde HTML5 (al menos hasta la fecha) como las famosas *Notifications*. De este modo, los usuarios de FLEQ desde dispositivos móviles podrían sustituir el sistema de aviso por mail por notificaciones en su *smartphone*, como sugería uno de los participantes del SATpeonato. Este tipo de notificaciones son menos molestas que un email, por lo que se podría pensar incluso en extender su uso, pudiendo por ejemplo notificar las selecciones de hora de los rivales en tiempo real. El desarrollo de aplicaciones nativas colocaría a

FLEQ en las principales tiendas de aplicaciones lo que sería una buena herramienta de marketing, al tiempo que se pueden conocer métricas sobre su uso (descargas, instalaciones activas, etc.). Sin embargo, esta mejora debe ser analizada con detenimiento, ya que requerirá de adaptaciones en el servidor que pueden suponer una cantidad de tiempo y complejidad elevadas.

- **Foros:** con el fin de aumentar la interacción y colaboración entre los participantes de los torneos, se plantea la inclusión de un tablón de comentarios sobre el que los usuarios pudieran plantear cuestiones, dudas y compartir conocimiento relacionado con el torneo en general o cada una de las categorías de preguntas de modo que habría, por ejemplo, un foro para discusiones sobre HTML, otro para Python, etc. Para este propósito se recomienda el uso de *Disqus*, un servicio web destinado a estos propósitos, empotrable en *sites* de terceros, y por cierto, desarrollado en Django.
- **Administración:** actualmente, un usuario administrador tiene acceso al módulo de administración de Django, mediante el cual puede manipular toda la información de la aplicación. Se debería generar una interfaz de gestión de torneos, que permitiera a los administradores seleccionar lotes de preguntas específicos para cada ronda, eliminar cuestiones, o corregir su contenido, todo esto, de forma externa al Admin de Django, que debería ser sólo utilizado por los responsables de mantenimiento de FLEQ. Este hecho permitiría abrir la herramienta más fácilmente al público general, ya que ahora mismo, un usuario administrador puede eliminar cualquier información en FLEQ, incluidas las propias cuentas de usuarios.
- **Integración en Moodle:** puede plantearse la integración de FLEQ en Moodle como un módulo adicional, permitiendo que los campeonatos online formen parte del desarrollo de una asignatura en esta plataforma educativa.
- **Nuevas modalidades de juego:** atendiendo a las peticiones de los participantes del SATpeonato, se insta a desarrolladores futuros al estudio de nuevas alternativas de juego: sistemas de torneo distintos al suizo, partidas por equipos, sistema de puntuación por *sets*, etc.

Como se ha podido comprobar, existen bastantes líneas de extensión abiertas muy interesantes lo cual es un buen síntoma de que la herramienta no ha tocado techo y puede

seguir creciendo.

Todas las ideas expuestas anteriormente son a nuestro juicio muy potentes, pero sin lugar a dudas la más interesante de todas es la primera.

La elaboración de un módulo estadístico potente será la clave del éxito de esta plataforma en el ámbito educativo, ya que será una forma objetiva de demostrar que este sistema cumple los objetivos que se ha propuesto, los cuales son muy interesantes desde el punto de vista académico.

6.4. Publicación del código

FLEQ es un proyecto de software libre, por lo que una vez finalizado el desarrollo se ha liberado el código de la aplicación.

Antes de realizar esta publicación se ha anexionado una licencia para evitar usos malintencionados. La licencia elegida para este propósito ha sido la Affero General Public License (Affero GPL), tal y como recomienda la Free Software Foundation (su creadora) para aplicaciones que se usan en servidores.

Es una licencia *copyleft* derivada de la licencia GPL a la que se añade una nueva cláusula para obligar a distribuir el software si éste se ejecuta para ofrecer servicios a través de una red de computadores.

De este modo, Affero GPL es la licencia adecuada para proteger el código de aplicaciones web como la que nos ocupa, ya que en caso de que alguien modifique el código y ponga la plataforma en producción, deberá hacer accesible el código modificado, permitiendo a los usuarios descargarlo.

Una vez elegida e incluida la licencia, el código ha sido publicado en el repositorio Google Code, bajo el sistema de control de versiones Subversion (SVN). La URL donde se encuentra alojado el proyecto FLEQ es la siguiente:

<http://code.google.com/p/fleq>

Cualquier persona puede así acceder al código de la aplicación (tanto del servidor

como del cliente), modificarlo y distribuirlo (el texto de la licencia Affero GPL puede consultarse en los Apéndices).

6.5. Valoración final

Una vez se alcanza el final de la presente memoria, se antoja necesario evaluar el desarrollo de este Proyecto Fin de Carrera de forma personal, transmitiendo cuáles han sido las sensaciones durante su desarrollo y las conclusiones que del mismo se deducen.

Lo primero de todo, decir que esta actividad académica con la que se cierra un ciclo, ha cubierto sobradamente las expectativas que se tenían en un principio.

El tutor del mismo, Gregorio, suele clasificar a los alumnos en dos grupos en función de su predisposición ante un proyecto de este tipo. Por un lado, alumnos que se lo plantean como una actividad en la que profundizar en una determinada materia poniendo broche a los conocimientos adquiridos durante la carrera, y por otro, alumnos que lo ven como un mero trámite que es necesario cumplir cuanto antes con el menor esfuerzo y dedicación posible.

Yo creo que su sensación inicial conmigo fue que pertenecía al primer grupo, aunque no deja de ser una percepción personal. Lo que sí me siento seguro de afirmar es que una vez concluido, no tiene dudas de que sí pertenezco a ese primer grupo, lo que me llena de satisfacción, ya que mi situación actual bien podría haberme empujado inconscientemente al segundo grupo.

Con mi situación actual me refiero a que este proyecto ha sido compaginado con mis labores como becario, que se iniciaron en el mes de junio de este mismo año, por lo que todo el desarrollo ha tenido que ser realizado en paralelo con esta otra actividad como ya he comentado antes.

Por eso mismo me siento muy orgulloso del trabajo realizado, porque con un esfuerzo y sacrificio bastante elevado, pocas horas de sueño y menos de ocio, se ha conseguido desarrollar un trabajo que a priori me costaba imaginar, y creo que a Gregorio también (al menos en cuanto a tiempo de ejecución).

En cuanto al desarrollo del proyecto, como he comentado al inicio de esta sección, estoy

tremendamente satisfecho porque ha servido para que desarrolle una forma de plantear las cosas que no se enseña durante la carrera.

Estamos acostumbrados a realizar actividades prácticas de forma guiada, sin plantearnos en la mayoría de los casos el por qué de las cosas. Simplemente se siguen unas pautas establecidas por los docentes. Pero en el Proyecto Fin de Carrera esto no es así.

Con este proyecto he aprendido a pensar de una manera diferente. Por primera vez en toda mi vida he tenido que plantear todo un desarrollo desde cero. Se ha tenido que trabajar proactivamente, a diferencia del resto de actividades académicas en las que el alumno es reactivo en la mayoría de los casos.

No creo que haya mejorado demasiado mis habilidades como programador con este desarrollo, tampoco creo que fuera ese el objetivo ni mucho menos, pero si estoy convencido de que he aprendido a resolver cuestiones por mí mismo, a plantear soluciones a situaciones reales.

El hecho de poder haber ido moldeando los objetivos del proyecto, desde los iniciales hasta los que finalmente han sido, me ha parecido una práctica interesantísima, porque he comprobado como las alternativas que he podido plantear de acuerdo a mis investigaciones y pensamientos han sido aprobadas por el tutor, porque he sido capaz de discutir sobre ello y demostrar las ventajas de las alternativas.

Además de esto, destacar también que el volumen de conocimiento adquirido durante su elaboración, en especial en la fase inicial de investigación, ha sido enorme por lo que la satisfacción es doble en este sentido.

Otra excepcional experiencia ha sido la experimentada durante la fase de pruebas de la herramienta, momento que afrontas con temores ya que tu trabajo va a ser expuesto a usuarios exigentes. Temores que posteriormente se transforman en una inmensa satisfacción cuando todo se desarrolla de la manera adecuada, ya que puedes comprobar como tu trabajo de meses está aportando un valor añadido y la gente disfruta utilizándolo.

Aunque no todo han sido alegrías. Momentos duros se han pasado, decir lo contrario sería mentir demasiado, pero creo que ha merecido la pena, como espero que haya merecido la pena el tiempo que el lector ha dedicado a leer esta memoria y haya disfrutado de la misma forma que yo he disfrutado con su escritura.

Bibliografía

- [1] FIRTMAN, M. *jQuery Mobile: Up and Running*. O'Reilly Media, 2012
- [2] KLINFELD, S. *HTML5 for Publishers*. O'Reilly Media, 2011
- [3] MORAL SANTIAGO, A. *Campeonatos online de preguntas para motivar y mejorar el aprendizaje*. ETSII (URJC), 2011
- [4] RICHARDSON, L., RUBY, S. *RESTful Web Services*, O'Reilly Media, 2007
- [5] SAINT-ANDRE, P. , SMITH, K., TRONÇON, R. *XMPP: The Definitive Guide*, O'Reilly Media, 2009
- [6] Django Documentation: <https://docs.djangoproject.com>
- [7] HTML5 Documentation, <http://dev.w3.org/html5/html-author/>
- [8] IBM Developer Works Site: <https://www.ibm.com/developerworks>
- [9] jQueryMobile Documentation: <http://jquerymobile.com/docs/>
- [10] Mozilla Developer Network (MDN), <https://developer.mozilla.org/es/>
- [11] Python Documentation: <http://python.org/doc/>
- [12] Tornado Documentation: <http://www.tornadoweb.org/documentation>
- [13] Selenium Documentation: <http://seleniumhq.org/docs/>
- [14] Socket IO Documentation, <https://github.com/learnboost/socket.io/wiki>
- [15] WebSocket Site: <http://www.websocket.org>

Apéndices

Apéndice A

Planificación del proyecto

Este proyecto ha sido desarrollado en el segundo semestre del año 2012, es decir, desde el mes de julio hasta diciembre de ese año, cuando es presentado.

Tras la finalización de la carrera en el mes de junio de 2012 se emprendió la elaboración del mismo, la cual ha comprendido varias fases que quedan resumidas en las siguientes secciones.

Estudio de objetivos (Julio-Agosto 2012)

Durante los primeros meses del proyecto, el trabajo se centró en determinar cuáles iban a ser los objetivos a conseguir durante su desarrollo, los cuales quedaron recogidos en el primer capítulo de la memoria.

Con el objetivo de optimizar el tiempo de desarrollo y los resultados, este apartado se antojaba fundamental, ya que una buena decisión a priori podría evitar desarrollos innecesarios.

De este modo, en esta fase se detectaron las posibles alternativas de que se disponían para la consecución de los distintos objetivos, y tras esto, se analizaron exhaustivamente para determinar cuáles serían las decisiones que provocarían la mejor consecución de los objetivos.

A la conclusión de esta fase, las líneas del proyecto quedaron claramente definidas lo que permitió un desarrollo con las máximas garantías posibles.

Desarrollo (Septiembre-Octubre 2012)

Una vez establecidas las *guidelines* del proyecto, llega la hora de la implementación del sistema.

Esta fase comprendió los dos meses siguientes, periodo en el que se desarrolló todo el grueso del sistema, en el siguiente orden:

1. Servidor (5 semanas)
2. Interfaz móvil (2 semanas)
3. Interfaz de escritorio (1 semana)

Este periodo del proyecto lleva implícitas las primeras pruebas de *testing*, necesarias durante el desarrollo de cualquier software.

Especial relevancia tomaron estas pruebas en la parte final de esta fase, en la que se realizaron las primeras pruebas reales de manera interna y/o privada.

Pruebas reales y escritura de la memoria (Noviembre 2012)

El mes de noviembre será destinado a la puesta en producción de FLEQ y a su utilización en un entorno real.

De forma paralela, se produce la elaboración de la presente memoria.

Últimos detalles (Diciembre 2012)

Durante el mes de diciembre, ya con la memoria concluida y la prueba real realizada, quedarían ciertos aspectos que cerrar antes de la presentación.

Estos aspectos comprenderían:

- Revisión del código para su posterior publicación en Google Code, ya que se trata de una plataforma de software libre
- Conversión de la presente memoria al formato de libro electrónico *epub*

- Elaboración de la presentación

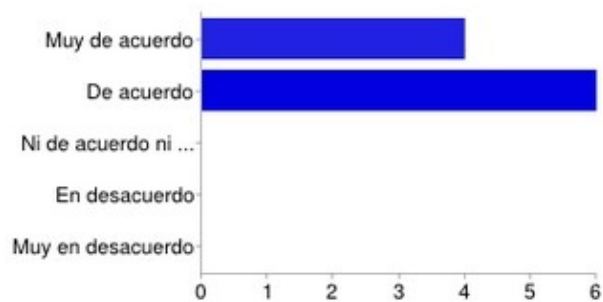
Con esto, el proyecto quedaría terminado y todo estará listo para el día de la presentación ante el tribunal, prevista para la última semana lectiva de ese mismo mes.

Apéndice B

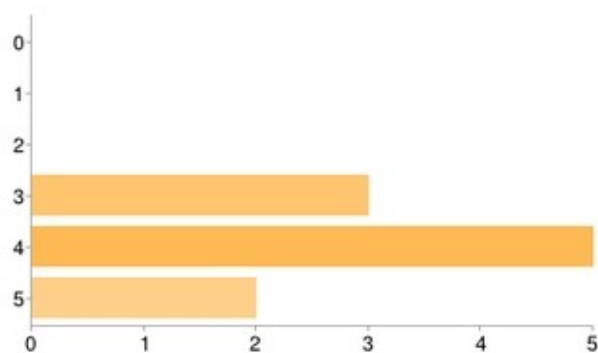
Resultados de las encuestas

En este apéndice se incluyen los resultados completos de las encuestas que rellenaron los participantes tras el SATpeonato.

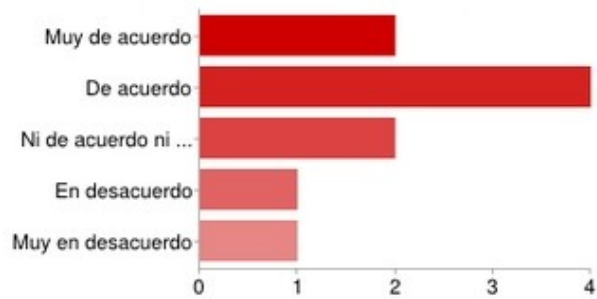
Pregunta 1. La interfaz móvil de FLEQ me parece fácil e intuitiva



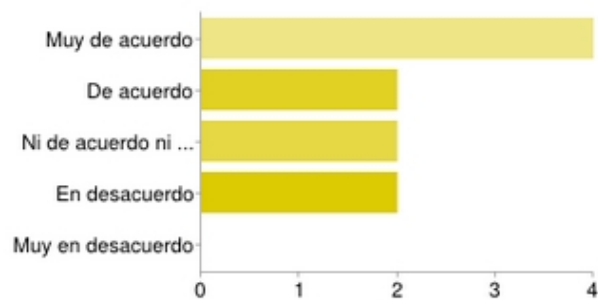
Pregunta 2. De 0 a 5, valora el diseño de la interfaz móvil de FLEQ



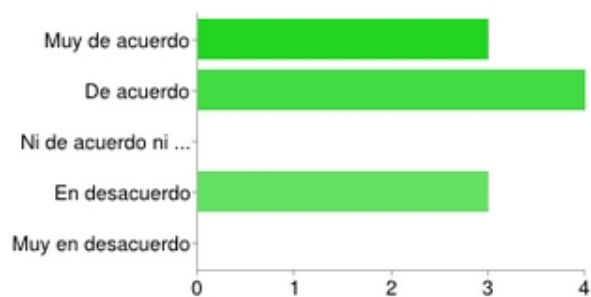
Pregunta 3. El sistema de cambio de hora consensuada me ha parecido muy adecuado ya que nos ha permitido jugar en el mejor horario



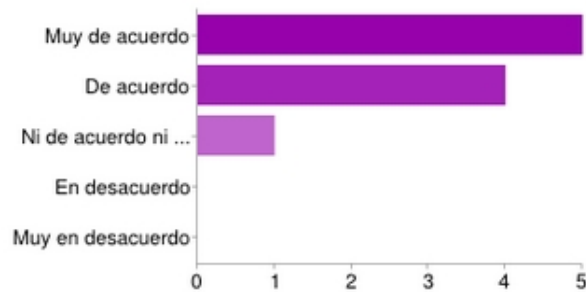
Pregunta 4. Los avisos a través de e-mail me han parecido acertados y necesarios



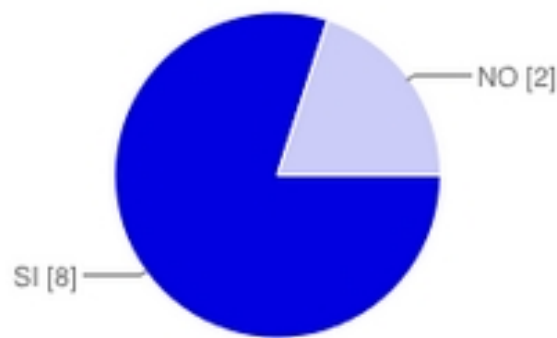
Pregunta 5. El ritmo de juego me parece adecuado: tiempo dado para responder, tiempo entre preguntas, duración de la partida, etc.



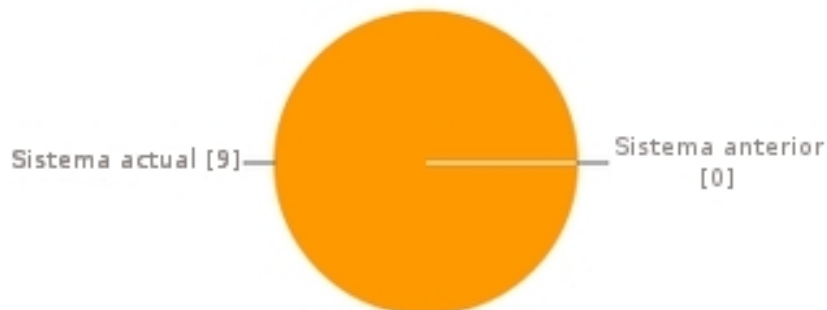
Pregunta 6. Me gustaría que se añadiesen otras modalidades de juego (e.g., por equipos) y tipos de torneo (e.g., eliminatorias directas, liguilla y eliminatorias)



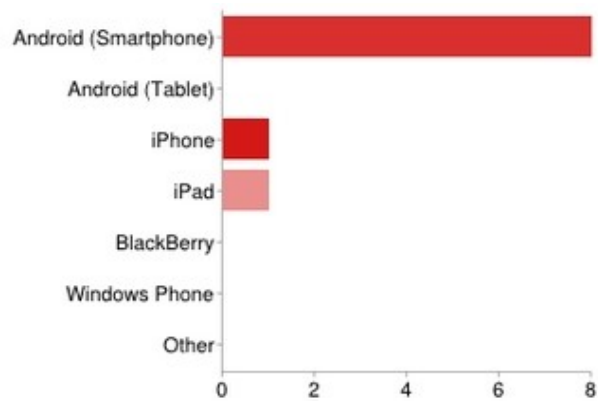
Pregunta 7. Indica si ha sido la primera vez que has jugado en FLEQ



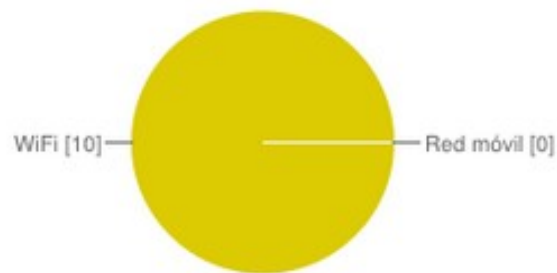
Pregunta 8 (opcional). En caso de que la respuesta anterior fuera negativa, indica qué sistema te parece mejor



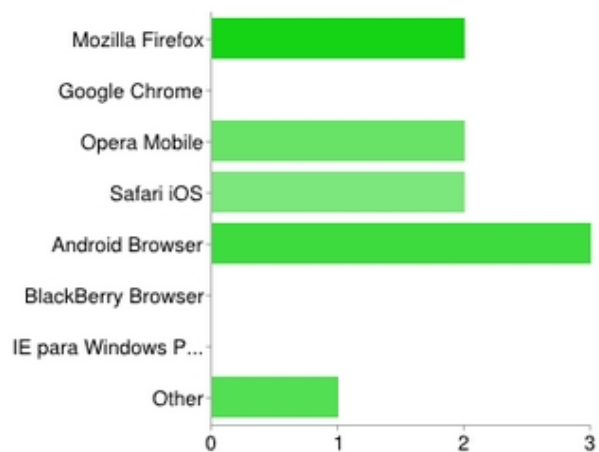
Pregunta 9. Indica el tipo de dispositivo que has utilizado para el campeonato



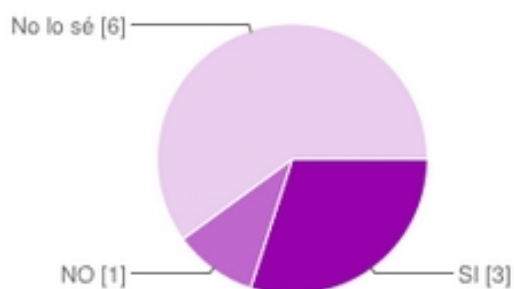
Pregunta 10. Indica qué tipo de conexión has utilizado habitualmente para jugar en FLEQ



Pregunta 11. Indica el navegador que has utilizado desde tu dispositivo móvil para participar en las partidas



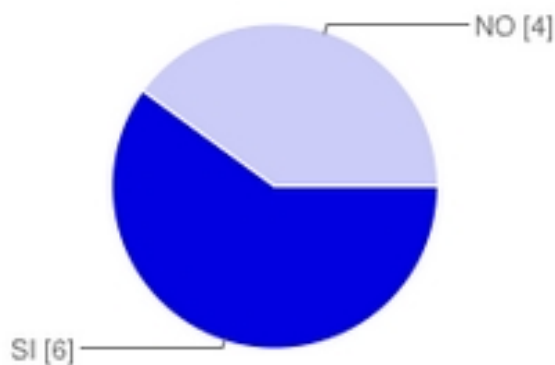
Pregunta 12. Indica si el navegador que utilizabas soportaba la API de WebSockets de HTML5



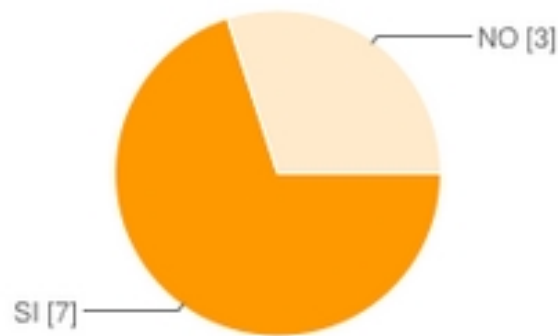
Pregunta 13 (opcional). Si en la respuesta anterior indicaste NO, explica si has notado algún tipo de problema, como por ejemplo, mayor latencia, desconexiones, etc.

1. Alumno 1: “Empecé utilizando Android Browser que no soporta la API de WebSockets de HTML5 y luego usé Opera Mobile que si la soporta, la verdad es que funcionaba bastante mejor, era más rápido y se desconectaba menos.”

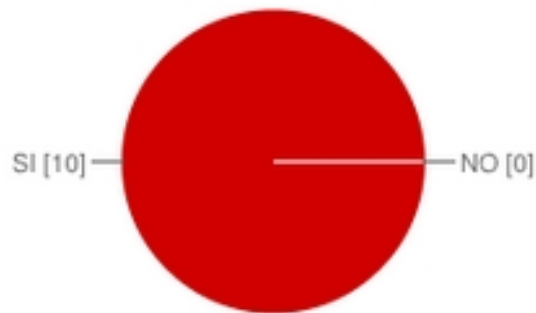
Pregunta 14. Indica si el campeonato ha estimulado tu estudio



Pregunta 15. Indica si el campeonato te ha permitido detectar puntos de la asignatura en los que debes poner mayor atención



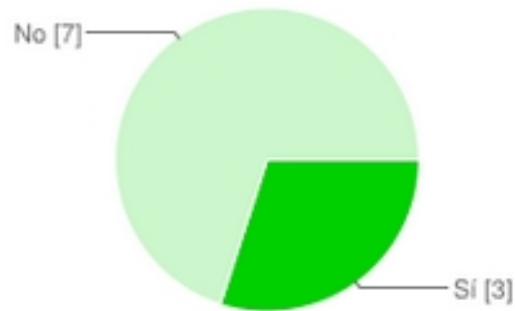
Pregunta 16. Indica si el campeonato ha logrado que tengas mayores conocimientos sobre HTML, Django y JavaScript



Pregunta 17. Indica si conocías la API de WebSocket de HTML5 antes de comenzar este campeonato



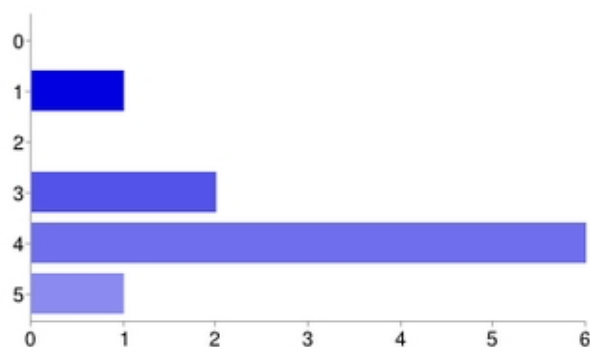
Pregunta 18. ¿Has participado en el campeonato de manera anónima? (o sea, los demás participantes no sabían quién eras tú a partir del nick)



Pregunta 19. ¿Te has sentido incómodo porque el campeonato sea competitivo?



Pregunta 20. Finalmente, indica tu grado de satisfacción con el campeonato y con FLEQ, de 0 a 5



Pregunta 21 (opcional). Añade libremente cualquier comentario sobre las cuestiones anteriores o cualquier tema relacionado con FLEQ, para ayudarnos a mejorar la herramienta

- Alumno 1: “Algunas preguntas tenían una respuesta muy larga y que contenían caracteres especiales, y con el teclado del móvil resultaba complicado escribirla.”
- Alumno 2: “La hora por defecto debería de ajustarse si uno de los 2 concursantes selecciona sus preferencias y el otro no.”
- Alumno 3: “Creo que sería interesante añadir la funcionalidad de que la última pregunta aparezca en el caso de quedarte sin conexión y volver a los pocos instantes.”
- Alumno 4: “Esta práctica o juego me ha ayudado a entender qué es un smartphone y como se navega con él más o menos.”
- Alumno 5: “Los mails llegaban con poca antelación, y creo que sería conveniente avisar a los jugadores por mail cuando se establezca la hora del juego, no solo cuando esté a punto de comenzar.”
- Alumno 6: “Es un poco incómodo que sea una aplicación a través del navegador y en tiempo real, en mi opinión prefiero las aplicaciones por turnos que te avisan en la barra de notificaciones, porque al avisar por email media hora antes se me ha olvidado varias veces participar a mi hora.”
- Alumno 7: “El SATpeonato ha estado muy bien en general. El único inconveniente que le he visto es que, al ser sólo para dispositivos móviles, me ha sido más difícil escribir correctamente algunas respuestas, porque es fácil pulsar en una letra distinta que esté cerca de la que se quiere pulsar, y es más complicado corregir algo que ya se ha escrito. De todas formas tengo poca práctica escribiendo en móviles táctiles por lo que sería adecuado comprobar si sucede también con gente con más práctica.”

Apéndice C

Glosario

Android

Android es un sistema operativo móvil basado en el kernel de Linux desarrollado por la OHA (Open Handset Alliance), liderada por la multinacional Google.

El proyecto Android nació en Android Inc, compañía que fue adquirida por Google en 2005. Su núcleo está desarrollado en el lenguaje C, con algunas bibliotecas escritas en C++. Sin embargo, el desarrollo de aplicaciones para esta plataforma se realiza principalmente en lenguaje Java, aunque existe la posibilidad de desarrollar aplicaciones nativas en C, práctica poco habitual por otra parte.

Apache

Apache es un servidor web de código abierto, escrito en C y multiplataforma, basado durante su desarrollo en el antiguo servidor NCSA HTTPd, aunque posteriormente fue reestructurado por completo.

Destaca por su modularidad. Apache consta de un *core* al que se pueden agregar diversos módulos para aumentar su funcionalidad. Entre estos módulos se encuentran ampliaciones orientadas a la seguridad, como el *mod_ssl*, u otros que permiten a Apache servir aplicaciones desarrolladas en diversos lenguajes como *mod_php*, *mod_perl*, *mod_ruby* o *mod_python*, aunque este último ya no es utilizado, tras la aparición de *mod_wsgi*.

Apache es el servidor más utilizado en la web, aunque su uso está decreciendo desde

2005, cuando alcanzó su mayor cuota de mercado con un 70% de uso.

API: Application Programming Interface

Conjunto de funciones y métodos empaquetados en bibliotecas o librerías que proporciona un alto nivel de abstracción y puede ser accedido desde otro programa sin necesidad de conocer su comportamiento interno.

AJAX: Asynchronous JavaScript And XML

Técnica de desarrollo web ejecutada en el lado del cliente, que hace uso del lenguaje interpretado JavaScript para establecer conexiones de forma asíncrona con el servidor, y XMLHttpRequest para el acceso a los datos, con el objetivo de actualizar una parte del contenido de la página sin tener que recargar el recurso al completo. Esto supone grandes beneficios en cuanto a velocidad, interactividad y usabilidad de la web.

CSS: Cascade Style Sheets

Lenguaje utilizado para definir la presentación de un documento estructurado en HTML, XML o XHTML. Nació con el objetivo de separar la presentación y la estructura en una página web. Puede ser integrado en el propio documento HTML o en un fichero individual.

DOM: Document Object Model

API que proporciona un conjunto estándar de objetos, un modelo sobre cómo pueden relacionarse y una interfaz para acceso y manipulación de los mismos. Mediante el DOM se puede acceder y modificar el contenido, la estructura y el estilo de documentos HTML y XML.

GSM: Global System for Mobile

Sistema estándar de comunicación empleado por los terminales móviles de segunda generación, antecesor de UMTS.

Gunicorn

Gunicorn (Green Unicorn) es un servidor web para aplicaciones desarrolladas en lenguaje Python. Surge a raíz del proyecto Unicorn para aplicaciones en Ruby.

Es compatible con diversos frameworks de Python, Django entre ellos. Destaca por su ligereza, rapidez y un escaso consumo de recursos.

Es utilizado en aplicaciones de tiempo real debido a que soporta hilos de ejecución asíncronos, lo que le permite trabajar con tecnologías como Comet o Websockets.

Este servidor web debe trabajar detrás de un proxy para evitar ataques DoS (Denial of Service). Entre los diferentes proxies a utilizar como *front-end* de Gunicorn, se recomienda el uso de Nginx.

HAProxy

Servidor proxy utilizado como balanceador de carga entre distintos servidores web. Destaca por su velocidad y su capacidad de distribución de carga de forma dinámica en función del estado de los servidores a los que administra.

HAProxy es utilizado como *front-end* de varios servidores web. Su uso permite optimizar el volumen de trabajo de un servicio, puesto que balancea la carga entre distintas máquinas, lo que evita sobrecargas en ciertos equipos.

Las peticiones que recibe HAProxy pueden ser enrutadas en función de otros parámetros configurables, como cookies, subdominios en las URL's, dominios IP de los clientes, etc. lo que permite distribuir las peticiones entre distintos servidores web especializados en diferentes tareas, como pueden ser el servicio de ficheros estáticos, peticiones asíncronas,

bloqueantes, websockets, etc.

HTML: HyperText Markup Language

Lenguaje de marcado que se emplea para describir y traducir la estructura e información de las páginas web en forma de texto, así como para introducir objetos como imágenes o vídeos. Es estructurado mediante etiquetas. Fue desarrollado inicialmente por Tim Berners-Lee en el Centro Europeo de Investigación Nuclear (CERN) a principios de los años 90, como un subconjunto del conocido SGML (Standard Generalized Markup Language). Fue creado con el objetivo de que científicos de distintas universidades e institutos de investigación pudieran compartir de forma fácil información acerca de sus investigaciones. Posteriormente, Berners-Lee creó el World Wide Web y el primer navegador web, el ViolaWWW.

IDE: Integrated Development Environment

Programa informático compuesto por un conjunto de herramientas de programación que facilitan el desarrollo de aplicaciones. Básicamente se compone de un editor de código, un compilador, un depurador y una interfaz gráfica de usuario (GUI).

IETF: Internet Engineering Task Force

Organización internacional de normalización creada en 1986, conocida por ser la entidad que regula las propuestas y los estándares de Internet, conocidos como RFC (Request For Comments).

iOS

Sistema operativo móvil desarrollado por Apple para dar vida a sus dispositivos inalámbricos, iPhone, iPad y iPod. Es un sistema propietario, por lo que no puede ser

utilizado en dispositivos externos a *la marca de la manzana*.

Este sistema operativo es un derivado de Mac OS X, que a su vez deriva de la distribución Unix denominada Darwin BSD. Está escrito en C y derivados de éste, como C++ u Objective-C. Fue anunciado en enero de 2007 aunque no se lanzó hasta el mes de junio de ese mismo año.

IRC: Internet Relay Chat

Protocolo de comunicación en tiempo real basado en texto, que permite comunicarse entre sí a dos o más personas en un sistema basado en canales, de modo que todos los usuarios que se encuentren en un canal podrán comunicarse aunque no hayan tenido ningún contacto previamente. Creado en 1988 por Jarkko Oikarinen, fue uno de los sistemas de comunicación más populares de la red en la década de los 90. Su uso ha decaído durante la última década debido a la aparición de las redes sociales o la mensajería instantánea.

JSON: JavaScript Object Notation

Formato para el intercambio de datos que se caracteriza por su simplicidad y ligereza, lo que le ha convertido en una alternativa a XML en las comunicaciones establecidas mediante técnicas AJAX.

LAMP: Linux Apache MySQL Python/Perl/PHP

Acrónimo utilizado para describir un servicio web que se encuentra alojado en una máquina Linux y utiliza Apache como servidor web, MySQL como gestor de bases de datos y, por último, Python, Perl o PHP como lenguajes de implementación de la aplicación web.

LTE: Long Term Evolution

Evolución de la tecnología UMTS que permite aumentar considerablemente las velocidades de acceso a la red, con el fin de abastecer el continuo crecimiento de la demanda de los usuarios, y permitir así el acceso a nuevos contenidos y aplicaciones como la televisión móvil, juegos en línea, contenidos en tiempo real o aplicaciones que cada vez requieren un flujo de datos más elevado.

Nginx

Servidor web y proxy inverso ligero de alto rendimiento, de código abierto, escrito en C y multiplataforma, creado en el año 2004.

Destaca por su eficacia como servidor de ficheros estáticos, balanceo de carga, la posibilidad de ser usado como proxy inverso, sistemas de comprensión y su capacidad de soportar más de 10k conexiones simultáneamente.

Además, puede ser utilizado como proxy para protocolos de correo electrónico como SMTP, POP3 e IMAP.

Su uso se está extendiendo en los últimos años, llegando a ser actualmente el tercer servidor web más utilizado con una cuota de mercado cercana al 12%.

OHA: Open Handset Alliance

Alianza comercial de 84 compañías dedicada al desarrollo de estándares abiertos para dispositivos móviles. Entre sus miembros destacados se encuentran corporaciones de renombre como Google (líder), Samsung, HTC, Dell, Intel, Qualcomm o Telefónica, entre otros. Fue constituida en 2007, fecha en la cual presentó su principal desarrollo, el sistema operativo para dispositivos móviles Android.

Paradigma de programación

Un paradigma de programación determina los métodos y procedimientos que un desarrollador empleará en la construcción de un software específico.

Un lenguaje de programación está basado en un determinado paradigma, habiendo sido creado para un propósito determinado, aunque existen lenguajes más versátiles que pueden dar respuesta a diversas funcionalidades, convirtiéndose en los denominados lenguajes multiparadigma.

Esto permite al desarrollador emplear el más conveniente en función del problema que se pretenda resolver. Ningún paradigma es capaz de resolver todos los problemas de forma sencilla y eficiente, por lo que es fundamental estudiar previamente el problema para determinar cuál será el método adecuado para resolverlo.

Entre los paradigmas principales se encuentran: imperativo, declarativo (dividido a su vez en funcional y lógico), orientado a objetos y programación dirigida por eventos.

PHP

Lenguaje de programación diseñado para el desarrollo de aplicaciones web con contenido dinámico.

Es un lenguaje multiparadigma ya que soporta orientación a objetos, programación imperativa, procedural y reflexiva.

PHP es modular, por lo que se puede expandir su potencial por medio de módulos o extensiones (ext's).

Sus mayores inconvenientes vienen derivados de su interpretación, que se realiza en tiempo de ejecución, y que su código fuente no puede ser ocultado ya que se encuentra embebido en los documentos HTML que cargan los navegadores.

Ruby

Lenguaje de programación multiparadigma creado por el japonés Yukihiro Matsumoto que vio la luz en el año 1995.

Su carácter multiparadigma le permite soportar orientación a objetos y programación funcional. Además, soporta programación reflexiva al igual que PHP.

Es multiplataforma y se distribuye bajo una licencia de software libre.

A pesar de que permite programación reflexiva y funcional, Ruby está claramente enfocado a la orientación a objetos. Todos los tipos de datos son objetos, incluso las clases y tipos que otros lenguajes de OO (Orientación a objetos) definen como primitivas.

Ruby on Rails (RoR)

Web framework escrito en Ruby sigue la arquitectura MVC (Modelo Vista Controlador). Su propósito principal es conseguir que el desarrollador pueda implementar aplicaciones web de forma simplista, con necesidad de menor cantidad de código y configuraciones.

Ruby On Rails hace uso de la metaprogramación que soporta Ruby, por lo que el código resultante es muy legible.

Además de los módulos que sirven para manejar los componentes de la arquitectura MVC (ActiveRecord o ActionPack), RoR dispone de una serie de extensiones para diversas funciones: Active Resource (creación de recursos siguiendo el paradigma REST), Action Mailer (envío de correo electrónico), AJAX on Rails (utilización de AJAX para procesar peticiones de un navegador) o Gemas (plugins que permiten extender la funcionalidad de RoR con funciones pre-escritas para el login de usuarios, desarrollo de plantillas, etc.).

SDK: Software Development Kit

Conjunto de herramientas de software que permiten el desarrollo de aplicaciones para un sistema concreto.

Smartphone

Un *smartphone* o teléfono inteligente es un nuevo concepto de terminal móvil con una mayor capacidad de computación y conectividad que funciona sobre un sistema operativo propio de última generación y que aprovecha las nuevas capacidades que proporcionan las redes móviles actuales, lo que les permite el acceso a contenido multimedia en tiempo real, así como multitud de funcionalidades adicionales que anteriormente no podían ser soportadas.

Entre sus principales características destacan: correo electrónico, acceso a Internet (WiFi o red móvil), GPS (Global Positioning System), acelerómetro, visor de documentos, aplicaciones, etc.

Tablet

Dispositivos móviles que se caracterizan por una pantalla de mayores dimensiones que un *smartphone*, entre 7 y 11 pulgadas. Su propósito es mejorar la experiencia de usuario en aplicaciones multimedia, y por regla general no incorporan la posibilidad de realizar llamadas.

TCP: Transfer Control Protocol

Protocolo de comunicación orientado a conexión y fiable del nivel de transporte del modelo OSI. Es la capa intermedia entre el protocolo de Internet (IP) y la aplicación. Permite que la comunicación entre dos sistemas se efectúe libre de errores, sin pérdidas y de forma segura.

UML: Unified Modeling Language

Lenguaje de modelado utilizado para visualizar, especificar y documentar sistemas, métodos o procesos de forma gráfica.

UMTS: Universal Mobile Telecommunications System

Tecnología empleada por los terminales móviles de tercera generación, sucesora de GSM, orientada a proporcionar servicios multimedia. Se caracteriza por sus capacidades multimedia, su velocidad de acceso a la red y una calidad de voz similar a la de las redes fijas. Desarrollada por el 3GPP, grupo de colaboración que une a los organismos internacionales de telecomunicaciones más importantes del mundo, es el sistema que ha producido el cambio radical de lo que es un teléfono móvil, pasando de ser un mero instrumento de comunicación para convertirse en un terminal multimedia con múltiples capacidades debido a la gran cantidad de servicios ofertados.

WAI-ARIA: Web Accessibility Initiative - Accesible Rich Internet Applications

Especificación publicada por el W3C que define cómo aumentar la accesibilidad en páginas web con contenido dinámico y componentes desarrollados en HTML, JavaScript y AJAX.

Web Framework

Conjunto de paquetes y módulos para el desarrollo de aplicaciones web que proporciona un alto nivel de abstracción, lo que evita que el desarrollador tenga que preocuparse del manejo de protocolos de comunicación, conexiones o hilos de ejecución a bajo nivel.

Esta abstracción permite que el desarrollo se centre en la propia lógica de la aplicación web, ya que el funcionamiento básico interno de ésta ya está soportado por los módulos que componen el web framework.

WHATWG: Web HyperText Application Technology Working Group

Comunidad interesada en la evolución de HTML y tecnologías relacionadas, fundada por compañías de la talla de Apple, Opera Software y la Fundación Mozilla.

Windows Phone

Sistema operativo móvil desarrollado por Microsoft (por lo que es un sistema propietario), que nació como reemplazo del anterior Windows Mobile a finales del año 2011.

En el momento de su lanzamiento incorporaba el navegador Internet Explorer Mobile 9, basado en el navegador de escritorio Internet Explorer 9. Esto supone que el sistema operativo móvil de Microsoft soporta en su navegador de serie la API de Websockets, a diferencia del navegador de serie de Android, que no lo soporta ni en su versión más actual.

WSGI: Web Server Gateway Interface

Interfaz que permite conectar aplicaciones web escritas en lenguaje Python con servidores web para su puesta en producción.

W3C: World Wide Web Consortium

Comunidad internacional que desarrolla estándares que aseguran el crecimiento de la World Wide Web a largo plazo.

XML: eXtensible Markup Language

Lenguaje de marcado desarrollado por el W3C que deriva del lenguaje SGML. Al ser extensible, permite añadir nuevas etiquetas en cualquier momento. Es utilizado en diversos ámbitos fuera de Internet, como en bases de datos, editores de texto, hojas de cálculo,

etc. ya que permite un intercambio de información de forma estructurada entre diferentes plataformas.

XHTML: eXtensible HyperText Markup Language

HTML expresado como XML válido. De esta forma es más estricto en cuanto a nivel técnico, lo que supone que su interpretación y detección de errores sea más rápida.

XHR: XMLHttpRequest

API disponible en lenguajes web de scripting como JavaScript. Es utilizado para enviar peticiones HTTP directamente y cargar la respuesta de forma inmediata. Los datos recibidos pueden ser interpretados en diferentes formatos, como JSON, XML, HTML o incluso texto plano. El contenido de estas respuestas puede ser utilizado para modificar el árbol DOM del documento HTML.

Apéndice D

Licencia Creative Commons

CREATIVE COMMONS CORPORATION NO ES UN DESPACHO DE ABOGADOS Y NO PROPORCIONA SERVICIOS JURÍDICOS. LA DISTRIBUCIÓN DE ESTA LICENCIA NO CREA UNA RELACIÓN ABOGADO-CLIENTE. CREATIVE COMMONS PROPORCIONA ESTA INFORMACIÓN TAL CUAL (ON AN 'AS-IS' BASIS). CREATIVE COMMONS NO OFRECE GARANTÍA ALGUNA RESPECTO DE LA INFORMACIÓN PROPORCIONADA, NI ASUME RESPONSABILIDAD ALGUNA POR DAÑOS PRODUCIDOS A CONSECUENCIA DE SU USO.

Licencia

LA OBRA O LA PRESTACIÓN (SEGÚN SE DEFINEN MÁS ADELANTE) SE PROPORCIONA BAJO LOS TÉRMINOS DE ESTA LICENCIA PÚBLICA DE CREATIVE COMMONS (CCPL O LICENCIA). LA OBRA O LA PRESTACIÓN SE ENCUENTRA PROTEGIDA POR LA LEY ESPAÑOLA DE PROPIEDAD INTELECTUAL Y/O CUALESQUIERA OTRAS NORMAS QUE RESULTEN DE APLICACIÓN. QUEDA PROHIBIDO CUALQUIER USO DE LA OBRA O PRESTACIÓN DIFERENTE A LO AUTORIZADO BAJO ESTA LICENCIA O LO DISPUESTO EN LA LEY DE PROPIEDAD INTELECTUAL.

MEDIANTE EL EJERCICIO DE CUALQUIER DERECHO SOBRE LA OBRA O LA PRESTACIÓN, USTED ACEPTA Y CONSIENTE LAS LIMITACIONES Y OBLIGACIONES DE ESTA LICENCIA, SIN PERJUICIO DE LA NECESIDAD DE CON-

SENTIMIENTO EXPRESO EN CASO DE VIOLACIÓN PREVIA DE LOS TÉRMINOS DE LA MISMA. EL LICENCIADOR LE CONCEDE LOS DERECHOS CONTENIDOS EN ESTA LICENCIA, SIEMPRE QUE USTED ACEPTÉ LOS PRESENTES TÉRMINOS Y CONDICIONES.

1. Definiciones

a. La obra es la creación literaria, artística o científica ofrecida bajo los términos de esta licencia.

b. En esta licencia se considera una prestación cualquier interpretación, ejecución, fonograma, grabación audiovisual, emisión o transmisión, mera fotografía u otros objetos protegidos por la legislación de propiedad intelectual vigente aplicable.

c. La aplicación de esta licencia a una colección (definida más adelante) afectará únicamente a su estructura en cuanto forma de expresión de la selección o disposición de sus contenidos, no siendo extensiva a estos. En este caso la colección tendrá la consideración de obra a efectos de esta licencia.

d. El titular originario es:

i. En el caso de una obra literaria, artística o científica, la persona natural o grupo de personas que creó la obra.

ii. En el caso de una obra colectiva, la persona que la edite y divulgue bajo su nombre, salvo pacto contrario.

iii. En el caso de una interpretación o ejecución, el actor, cantante, músico, o cualquier otra persona que represente, cante, lea, recite, interprete o ejecute en cualquier forma una obra.

iv. En el caso de un fonograma, el productor fonográfico, es decir, la persona natural o jurídica bajo cuya iniciativa y responsabilidad se realiza por primera vez una fijación exclusivamente sonora de la ejecución de una obra o de otros sonidos.

v. En el caso de una grabación audiovisual, el productor de la grabación, es decir, la persona natural o jurídica que tenga la iniciativa y asuma la responsabilidad de las

fijaciones de un plano o secuencia de imágenes, con o sin sonido.

vi. En el caso de una emisión o una transmisión, la entidad de radiodifusión.

vii. En el caso de una mera fotografía, aquella persona que la haya realizado.

viii. En el caso de otros objetos protegidos por la legislación de propiedad intelectual vigente, la persona que ésta señale.

e. Se considerarán obras derivadas aquellas obras creadas a partir de la licenciada, como por ejemplo: las traducciones y adaptaciones; las revisiones, actualizaciones y anotaciones; los compendios, resúmenes y extractos; los arreglos musicales y, en general, cualesquiera transformaciones de una obra literaria, artística o científica. Para evitar la duda, si la obra consiste en una composición musical o grabación de sonidos, la sincronización temporal de la obra con una imagen en movimiento (synching) será considerada como una obra derivada a efectos de esta licencia.

f. Tendrán la consideración de colecciones la recopilación de obras ajenas, de datos o de otros elementos independientes como las antologías y las bases de datos que por la selección o disposición de sus contenidos constituyan creaciones intelectuales. La mera incorporación de una obra en una colección no dará lugar a una derivada a efectos de esta licencia.

g. El licenciador es la persona o la entidad que ofrece la obra o prestación bajo los términos de esta licencia y le concede los derechos de explotación de la misma conforme a lo dispuesto en ella.

h. Usted es la persona o la entidad que ejercita los derechos concedidos mediante esta licencia y que no ha violado previamente los términos de la misma con respecto a la obra o la prestación, o que ha recibido el permiso expreso del licenciador de ejercitar los derechos concedidos mediante esta licencia a pesar de una violación anterior.

i. La transformación de una obra comprende su traducción, adaptación y cualquier otra modificación en su forma de la que se derive una obra diferente. La creación resultante de la transformación de una obra tendrá la consideración de obra derivada.

j. Se entiende por reproducción la fijación directa o indirecta, provisional o permanente, por cualquier medio y en cualquier forma, de toda la obra o la prestación o de parte de

ella, que permita su comunicación o la obtención de copias.

k. Se entiende por distribución la puesta a disposición del público del original o de las copias de la obra o la prestación, en un soporte tangible, mediante su venta, alquiler, préstamo o de cualquier otra forma.

l. Se entiende por comunicación pública todo acto por el cual una pluralidad de personas, que no pertenezcan al ámbito doméstico de quien la lleva a cabo, pueda tener acceso a la obra o la prestación sin previa distribución de ejemplares a cada una de ellas. Se considera comunicación pública la puesta a disposición del público de obras o prestaciones por procedimientos alámbricos o inalámbricos, de tal forma que cualquier persona pueda acceder a ellas desde el lugar y en el momento que elija.

m. La explotación de la obra o la prestación comprende la reproducción, la distribución, la comunicación pública y, en su caso, la transformación.

n. Los elementos de la licencia son las características principales de la licencia según la selección efectuada por el licenciador e indicadas en el título de esta licencia: Reconocimiento, CompartirIgual.

o. Una licencia equivalente es:

i. Una versión posterior de esta licencia de Creative Commons con los mismos elementos de licencia.

ii. La misma versión o una versión posterior de esta licencia de cualquier otra jurisdicción reconocida por Creative Commons con los mismos elementos de la licencia (ejemplo: Reconocimiento-CompartirIgual 3.0 Japón).

iii. La misma versión o una versión posterior de la licencia de Creative Commons no adaptada a ninguna jurisdicción (Unported) con los mismos elementos de la licencia.

iv. Una de las licencias compatibles que aparece en <http://creativecommons.org.../compatiblelicenses> y que ha sido aprobada por Creative Commons como esencialmente equivalente a esta licencia porque, como mínimo:

a. Contiene términos con el mismo propósito, el mismo significado y el mismo efecto que los elementos de esta licencia.

b. Permite explícitamente que las obras derivadas de obras sujetas a ella puedan ser distribuidas mediante esta licencia, la licencia de Creative Commons no adaptada a ninguna jurisdicción (Unported) o una licencia de cualquier otra jurisdicción reconocida por Creative Commons, con sus mismos elementos de licencia.

2. Límites de los derechos

Nada en esta licencia pretende reducir o restringir cualesquiera límites legales de los derechos exclusivos del titular de los derechos de propiedad intelectual de acuerdo con la Ley de propiedad intelectual o cualesquiera otras leyes aplicables, ya sean derivados de usos legítimos, tales como la copia privada o la cita, u otras limitaciones como la resultante de la primera venta de ejemplares (agotamiento).

3. Concesión de licencia

Conforme a los términos y a las condiciones de esta licencia, el licenciador concede, por el plazo de protección de los derechos de propiedad intelectual y a título gratuito, una licencia de ámbito mundial no exclusiva que incluye los derechos siguientes:

- a. Derecho de reproducción, distribución y comunicación pública de la obra o la prestación.
- b. Derecho a incorporar la obra o la prestación en una o más colecciones.
- c. Derecho de reproducción, distribución y comunicación pública de la obra o la prestación lícitamente incorporada en una colección.
- d. Derecho de transformación de la obra para crear una obra derivada siempre y cuando se incluya en ésta una indicación de la transformación o modificación efectuada.
- e. Derecho de reproducción, distribución y comunicación pública de obras derivadas creadas a partir de la obra licenciada.
- f. Derecho a extraer y reutilizar la obra o la prestación de una base de datos.
- g. Para evitar cualquier duda, el titular originario:

i. Conserva el derecho a percibir las remuneraciones o compensaciones previstas por actos de explotación de la obra o prestación, calificadas por la ley como irrenunciables e inalienables y sujetas a gestión colectiva obligatoria.

ii. Renuncia al derecho exclusivo a percibir, tanto individualmente como mediante una entidad de gestión colectiva de derechos, cualquier remuneración derivada de actos de explotación de la obra o prestación que usted realice.

Estos derechos se pueden ejercitar en todos los medios y formatos, tangibles o intangibles, conocidos en el momento de la concesión de esta licencia. Los derechos mencionados incluyen el derecho a efectuar las modificaciones que sean precisas técnicamente para el ejercicio de los derechos en otros medios y formatos. Todos los derechos no concedidos expresamente por el licenciador quedan reservados, incluyendo, a título enunciativo pero no limitativo, los derechos morales irrenunciables reconocidos por la ley aplicable. En la medida en que el licenciador ostente derechos exclusivos previstos por la ley nacional vigente que implementa la directiva europea en materia de derecho sui generis sobre bases de datos, renuncia expresamente a dichos derechos exclusivos.

4. Restricciones

La concesión de derechos que supone esta licencia se encuentra sujeta y limitada a las restricciones siguientes:

a. Usted puede reproducir, distribuir o comunicar públicamente la obra o prestación solamente bajo los términos de esta licencia y debe incluir una copia de la misma, o su Identificador Uniforme de Recurso (URI). Usted no puede ofrecer o imponer ninguna condición sobre la obra o prestación que altere o restrinja los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Usted no puede sublicenciar la obra o prestación. Usted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías. Usted no puede reproducir, distribuir o comunicar públicamente la obra o prestación con medidas tecnológicas que controlen el acceso o el uso de una manera contraria a los términos de esta licencia. Esta sección 4.a también afecta a la obra o prestación incorporada en una colección, pero ello no implica que ésta en su conjunto quede automáticamente o deba quedar sujeta a los términos de

la misma. En el caso que le sea requerido, previa comunicación del licenciador, si usted incorpora la obra en una colección y/o crea una obra derivada, deberá quitar cualquier crédito requerido en el apartado 4.c, en la medida de lo posible.

b. Usted puede distribuir o comunicar públicamente una obra derivada en el sentido de esta licencia solamente bajo los términos de la misma u otra licencia equivalente. Si usted utiliza esta misma licencia debe incluir una copia o bien su URI, con cada obra derivada que usted distribuya o comunique públicamente. Usted no puede ofrecer o imponer ningún término respecto a la obra derivada que altere o restrinja los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Usted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías cuando distribuya o comunique públicamente la obra derivada. Usted no puede ofrecer o imponer ningún término respecto de las obras derivadas o sus transformaciones que alteren o restrinjan los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Usted no puede reproducir, distribuir o comunicar públicamente la obra derivada con medidas tecnológicas que controlen el acceso o uso de la obra de una manera contraria a los términos de esta licencia. Si utiliza una licencia equivalente debe cumplir con los requisitos que ésta establezca cuando distribuya o comunique públicamente la obra derivada. Todas estas condiciones se aplican a una obra derivada en tanto que incorporada a una colección, pero no implica que ésta tenga que estar sujeta a los términos de esta licencia.

c. Si usted reproduce, distribuye o comunica públicamente la obra o la prestación, una colección que la incorpore o cualquier obra derivada, debe mantener intactos todos los avisos sobre la propiedad intelectual e indicar, de manera razonable conforme al medio o a los medios que usted esté utilizando:

i. El nombre del autor original, o el seudónimo si es el caso, así como el del titular originario, si le es facilitado.

ii. El nombre de aquellas partes (por ejemplo: institución, publicación, revista) que el titular originario y/o el licenciador designen para ser reconocidos en el aviso legal, las condiciones de uso, o de cualquier otra manera razonable.

iii. El título de la obra o la prestación si le es facilitado.

iv. El URI, si existe, que el licenciador especifique para ser vinculado a la obra o la prestación, a menos que tal URI no se refiera al aviso legal o a la información sobre la licencia de la obra o la prestación.

v. En el caso de una obra derivada, un aviso que identifique la transformación de la obra en la obra derivada (p. ej., "traducción castellana de la obra de Autor Original," "guión basado en obra original de Autor Original").

d. Este reconocimiento debe hacerse de manera razonable. En el caso de una obra derivada o incorporación en una colección estos créditos deberán aparecer como mínimo en el mismo lugar donde se hallen los correspondientes a otros autores o titulares y de forma comparable a los mismos. Para evitar la duda, los créditos requeridos en esta sección sólo serán utilizados a efectos de atribución de la obra o la prestación en la manera especificada anteriormente. Sin un permiso previo por escrito, usted no puede afirmar ni dar a entender implícitamente ni explícitamente ninguna conexión, patrocinio o aprobación por parte del titular originario, el licenciador y/o las partes reconocidas hacia usted o hacia el uso que hace de la obra o la prestación. Para evitar cualquier duda, debe hacerse notar que las restricciones anteriores (párrafos 4.a, 4.b y 4.c) no son de aplicación a aquellas partes de la obra o la prestación objeto de esta licencia que únicamente puedan ser protegidas mediante el derecho sui generis sobre bases de datos recogido por la ley nacional vigente implementando la directiva europea de bases de datos.

5. Exoneración de responsabilidad

A MENOS QUE SE ACUERDE MUTUAMENTE ENTRE LAS PARTES, EL LICENCIADOR OFRECE LA OBRA O LA PRESTACIÓN TAL CUAL (ON AN "AS-IS" BASIS) Y NO CONFIERE NINGUNA GARANTÍA DE CUALQUIER TIPO RESPECTO DE LA OBRA O LA PRESTACIÓN O DE LA PRESENCIA O AUSENCIA DE ERRORES QUE PUEDAN O NO SER DESCUBIERTOS. ALGUNAS JURISDICCIONES NO PERMITEN LA EXCLUSIÓN DE TALES GARANTÍAS, POR LO QUE TAL EXCLUSIÓN PUEDE NO SER DE APLICACIÓN A USTED.

6. Limitación de responsabilidad

SALVO QUE LO DISPONGA EXPRESA E IMPERATIVAMENTE LA LEY APLICABLE, EN NINGÚN CASO EL LICENCIADOR SERÁ RESPONSABLE ANTE USTED POR CUALESQUIERA DAÑOS RESULTANTES, GENERALES O ESPECIALES (INCLUIDO EL DAÑO EMERGENTE Y EL LUCRO CESANTE), FORTUITOS O CAUSALES, DIRECTOS O INDIRECTOS, PRODUCIDOS EN CONEXIÓN CON ESTA LICENCIA O EL USO DE LA OBRA O LA PRESTACIÓN, INCLUSO SI EL LICENCIADOR HUBIERA SIDO INFORMADO DE LA POSIBILIDAD DE TALES DAÑOS.

7. Finalización de la licencia

a. Esta licencia y la concesión de los derechos que contiene terminarán automáticamente en caso de cualquier incumplimiento de los términos de la misma. Las personas o entidades que hayan recibido de usted obras derivadas o colecciones bajo esta licencia, sin embargo, no verán sus licencias finalizadas, siempre que tales personas o entidades se mantengan en el cumplimiento íntegro de esta licencia. Las secciones 1, 2, 5, 6, 7 y 8 permanecerán vigentes pese a cualquier finalización de esta licencia.

b. Conforme a las condiciones y términos anteriores, la concesión de derechos de esta licencia es vigente por todo el plazo de protección de los derechos de propiedad intelectual según la ley aplicable. A pesar de lo anterior, el licenciador se reserva el derecho a divulgar o publicar la obra o la prestación en condiciones distintas a las presentes, o de retirar la obra o la prestación en cualquier momento. No obstante, ello no supondrá dar por concluida esta licencia (o cualquier otra licencia que haya sido concedida, o sea necesario ser concedida, bajo los términos de esta licencia), que continuará vigente y con efectos completos a no ser que haya finalizado conforme a lo establecido anteriormente, sin perjuicio del derecho moral de arrepentimiento en los términos reconocidos por la ley de propiedad intelectual aplicable.

8. Miscelánea

a. Cada vez que usted realice cualquier tipo de explotación de la obra o la prestación, o de una colección que la incorpore, el licenciador ofrece a los terceros y sucesivos licenciarios la concesión de derechos sobre la obra o la prestación en las mismas condiciones y términos que la licencia concedida a usted.

b. Cada vez que usted realice cualquier tipo de explotación de una obra derivada, el licenciador ofrece a los terceros y sucesivos licenciarios la concesión de derechos sobre la obra objeto de esta licencia en las mismas condiciones y términos que la licencia concedida a usted.

c. Si alguna disposición de esta licencia resulta inválida o inaplicable según la Ley vigente, ello no afectará la validez o aplicabilidad del resto de los términos de esta licencia y, sin ninguna acción adicional por cualquiera de las partes de este acuerdo, tal disposición se entenderá reformada en lo estrictamente necesario para hacer que tal disposición sea válida y ejecutiva.

d. No se entenderá que existe renuncia respecto de algún término o disposición de esta licencia, ni que se consiente violación alguna de la misma, a menos que tal renuncia o consentimiento figure por escrito y lleve la firma de la parte que renuncie o consienta.

e. Esta licencia constituye el acuerdo pleno entre las partes con respecto a la obra o la prestación objeto de la licencia. No caben interpretaciones, acuerdos o condiciones con respecto a la obra o la prestación que no se encuentren expresamente especificados en la presente licencia. El licenciador no estará obligado por ninguna disposición complementaria que pueda aparecer en cualquier comunicación que le haga llegar usted. Esta licencia no se puede modificar sin el mutuo acuerdo por escrito entre el licenciador y usted.

Aviso de Creative Commons

Creative Commons no es parte de esta licencia, y no ofrece ninguna garantía en relación con la obra o la prestación. Creative Commons no será responsable frente a usted o a cualquier parte, por cualesquiera daños resultantes, incluyendo, pero no limitado, daños generales o especiales (incluido el daño emergente y el lucro cesante), fortuitos o causales,

en conexión con esta licencia. A pesar de las dos (2) oraciones anteriores, si Creative Commons se ha identificado expresamente como el licenciador, tendrá todos los derechos y obligaciones del licenciador.

Salvo para el propósito limitado de indicar al público que la obra o la prestación está licenciada bajo la CCPL, ninguna parte utilizará la marca registrada Creative Commons.º cualquier marca registrada o insignia relacionada con Creative Commons"sin su consentimiento por escrito. Cualquier uso permitido se hará de conformidad con las pautas vigentes en cada momento sobre el uso de la marca registrada por Creative Commons", en tanto que sean publicadas su sitio web (website) o sean proporcionadas a petición previa. Para evitar cualquier duda, estas restricciones en el uso de la marca no forman parte de esta licencia.

Puede contactar con Creative Commons en: <http://creativecommons.org/>

Apéndice E

Licencia Affero GPL

GNU AFFERO GENERAL PUBLIC LICENSE

Version 3, 19 June 2007

Copyright © 2007 Free Software Foundation, Inc.

<http://fsf.org>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs,

and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions

'This License' refers to version 3 of the GNU Affero General Public License.

'Copyright' also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

'The Program' refers to any copyrightable work licensed under this License. Each licensee is addressed as 'you'. 'Licensees' and 'recipients' may be individuals or organizations.

To 'modify' a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a 'modified version' of the earlier work or a work 'based on' the earlier work.

A 'covered work' means either the unmodified Program or a work based on the Program.

To 'propagate' a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To 'convey' a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays 'Appropriate Legal Notices' to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code

The 'source code' for a work means the preferred form of the work for making modifications to it. 'Object code' means any non-source form of a work.

A 'Standard Interface' means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The 'System Libraries' of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A 'Major Component', in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The 'Corresponding Source' for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without

conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to 'keep intact all notices'.

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an 'aggregate' if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and non-commercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to

the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A 'User Product' is either (1) a 'consumer product', which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, 'normally used' refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

'Installation Information' for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been

modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms

'Additional permissions' are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered 'further restrictions' within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An 'entity transaction' is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents

A 'contributor' is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's 'contributor version'.

A contributor's 'essential patent claims' are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, 'control' includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a 'patent license' is any express agreement or com-

mitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To 'grant' such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. 'Knowingly relying' means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is 'discriminatory' if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license

or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License

The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License or any later version applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM 'AS IS' WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

