

**BREAKBRAIN: RED SOCIAL PARA LA MEJORA DE HABILIDADES
MENTALES A TRAVÉS DE JUEGOS**



**UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA**

**INGENIERÍA
EN INFORMÁTICA**

PROYECTO FIN DE CARRERA

BreakBrain: Red social para la mejora de habilidades mentales
a través de juegos

Sergio García Mondaray

Septiembre, 2013



**UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA**

Departamento de Tecnologías y Sistemas de Información

PROYECTO FIN DE CARRERA

BreakBrain: Red social para la mejora de habilidades mentales
a través de juegos

Autor: Sergio García Mondaray
Director: Jesús Serrano Guerrero

Septiembre, 2013

Sergio García Mondaray

Ciudad Real – Spain

E-mail: sgmonda@gmail.com

Teléfono: 661 778 347

Web site: <http://www.sgmonda.com>

© 2013 Sergio García Mondaray

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Se permite la copia, distribución y/o modificación de este documento bajo los términos de la Licencia de Documentación Libre GNU, versión 1.3 o cualquier versión posterior publicada por la *Free Software Foundation*; sin secciones invariantes. Una copia de esta licencia esta incluida en el apéndice titulado «GNU Free Documentation License».

Muchos de los nombres usados por las compañías para diferenciar sus productos y servicios son reclamados como marcas registradas. Allí donde estos nombres aparezcan en este documento, y cuando el autor haya sido informado de esas marcas registradas, los nombres estarán escritos en mayúsculas o como nombres propios.

TRIBUNAL:

Presidente:

Vocal 1:

Vocal 2:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE

VOCAL 1

VOCAL 2

SECRETARIO

Fdo.:

Fdo.:

Fdo.:

Fdo.:

Resumen

El presente documento es un ejemplo de memoria del Proyecto Fin de Carrera o Trabajo de Fin de Máster según el formato y criterios de la Escuela Superior de Informática de Ciudad Real. La intención es que este texto sirva además como una serie de consejos sobre tipografía, L^AT_EX, redacción y estructura de la memoria que podrían resultar de ayuda. Por este motivo, se aconseja al lector consultar también el código fuente de este documento.

Este documento utiliza la clase L^AT_EX *arco-pfc*, disponible como paquete Debian/Ubuntu, consulta:

https://bitbucket.org/arco_group/arco-pfc.

Si encuentra cualquier error o tiene alguna sugerencia, por favor, utilice el *issue tracker* del proyecto *arco-pfc* en:

https://bitbucket.org/arco_group/arco-pfc/issues

El resumen debería estar formado por dos o tres párrafos resaltando lo más destacable del documento. No es una introducción al problema, es decir, debería incluir los logros más importantes del proyecto. Suele ser más sencillo escribirlo cuando la memoria está prácticamente terminada. Debería caber en esta página (es decir, esta cara).

Índice general

Resumen	XI
Índice general	XIII
Índice de cuadros	XIX
Índice de figuras	XXI
Índice de listados	XXV
Listado de acrónimos	XXVII
Agradecimientos	XXIX
1. Introducción	1
1.1. Redes sociales	1
1.2. El cerebro humano	2
1.2.1. La neuroplasticidad	3
1.3. El entrenamiento cerebral mediante juegos	3
1.4. Estructura del documento	4
2. Motivación y objetivos	7
2.1. Motivación	7
2.1.1. Enfermedad mental	8
2.1.2. Prevención de la degeneración neuronal	8
2.1.3. Mejora activa	8
2.2. Objetivos	8
2.2.1. Objetivo general	9
2.2.2. Objetivos específicos	9
3. Antecedentes	11
3.1. Sitios web de juegos online	11

3.2. Entrenamiento cerebral mediante mini juegos	15
3.3. Desarrollo web moderno	17
3.3.1. Servidor web	18
3.3.2. Cliente web	21
3.4. Anatomía y fisiología del cerebro humano	24
3.4.1. Estructura cerebral	24
3.4.2. El desarrollo cerebral	28
3.4.3. Estructuras primarias del cerebro	31
3.4.4. Los hemisferios cerebrales	33
3.4.5. La corteza cerebral	35
3.4.6. Estructuras subcorticales. El estrés, las emociones y la enfermedad mental	45
3.5. Neuroplasticidad	50
3.5.1. Memoria (Memory)	52
3.5.2. Resolución de problemas (Problem solving)	54
3.5.3. Atención (Attention)	55
3.5.4. Velocidad (Speed)	55
3.5.5. Flexibilidad (Flexibility)	56
4. Método de trabajo	59
4.1. Proceso Unificado de Desarrollo	59
4.1.1. Visión global del proceso	59
4.1.2. Etapas e iteraciones (dimensión dinámica)	60
4.1.3. Artefactos (dimensión estática)	61
4.2. Iteraciones	61
4.3. Análisis de requisitos	61
4.3.1. Primera aproximación a los requisitos	62
4.3.2. Requisitos funcionales	64
4.3.3. Requisitos de evolución	72
4.3.4. Requisitos de soporte	73
4.3.5. Requisitos de calidad	73
4.4. Diagrama de despliegue	75
4.5. Análisis de casos de uso	76
4.5.1. Identificación de actores	76
4.5.2. Identificación de casos de uso	77
4.5.3. Especificación de casos de uso	78

4.6. Diseño	101
4.7. Implementación	102
4.7.1. Tecnología	102
4.7.2. Estructura del código fuente	103
4.8. Estadísticas del código fuente	103
4.9. Pruebas	105
4.9.1. Pruebas unitarias	105
4.9.2. Pruebas funcionales	106
4.9.3. Pruebas de aceptación	106
4.9.4. Pruebas de estrés del sistema	107
5. Arquitectura	115
5.1. Composición del sistema	115
5.2. Arquitectura del sitio web	116
5.2.1. Patrón MVC	116
5.3. Arquitectura del servidor	117
5.3.1. Patrón MVC	117
5.4. Diseño modular	118
6. Resultados	121
6.1. Especificación del entrenamiento cerebral	121
6.1.1. Memoria	121
6.1.2. Resolución de problemas	123
6.1.3. Atención	124
6.1.4. Velocidad de procesamiento	125
6.1.5. Flexibilidad	126
6.2. Medición cuantitativa de la evolución cerebral	129
6.2.1. Juegos, partidas y actividades puntuables	129
6.2.2. Puntuación de partidas	130
6.2.3. Variables de clasificación	131
6.3. Sistema de recomendación	134
6.3.1. Algoritmo de recomendación de usuarios	134
6.3.2. Algoritmo de recomendación de juegos	134
7. Conclusiones y propuestas	135
7.1. Objetivos alcanzados	135

7.2. Propuestas de trabajo futuro	136
7.3. Opinión personal	138
A. Iteraciones del PUD	143
A.1. Iteraciones de inicio	143
A.2. Iteraciones de elaboración	144
A.3. Iteraciones de construcción	145
A.4. Etapa de transición	148
A.5. Visión gráfica	148
B. Guía de instalación	149
B.1. Requisitos previos	149
B.1.1. NodeJS	149
B.1.2. MongoDB	149
B.2. Instalación de BreakBrain	150
C. Guía para el desarrollador: creación de juegos para BreakBrain	153
C.1. Requisitos	153
C.1.1. Conocimientos del desarrollador	153
C.1.2. Herramientas de desarrollo	154
C.2. Tipos de juegos	154
C.2.1. Juegos monojugador	154
C.2.2. Juegos multijugador	155
C.3. El sistema de juegos de BreakBrain	155
C.3.1. Componentes del sistema de juegos	155
C.3.2. Componentes de un juego	155
C.4. Creación de un juego desde cero	156
C.4.1. Eligiendo una categoría	156
C.4.2. Eligiendo el tipo de juego	158
C.4.3. Creando la estructura básica del juego	158
C.4.4. Preparando los recursos	159
C.4.5. Desarrollando la parte servidora	161
C.4.6. Desarrollando la parte cliente	167
C.5. Publicación del juego en BreakBrain	176
D. Manual de usuario	177
D.1. Registro	177

D.1.1. Activación de la cuenta	177
D.1.2. Recuperación de la contraseña	177
D.2. Primer acceso	177
D.2.1. Completando el perfil de usuario	177
D.2.2. Definiendo las preferencias de entrenamiento	177
D.3. Siguiendo a otros usuarios	177
D.4. Entrenando el cerebro con juegos	177
D.4.1. Juegos monojugador	177
D.4.2. Jugando contra otros usuarios	177
E. Licencia de uso del software	179
Bibliografía	197
Índice alfabético	203

Índice de cuadros

4.1. Evolución del proceso de desarrollo. Los porcentajes se han estimado teniendo en cuenta las líneas de código (para el caso de la implementación), la cobertura para las pruebas, y el número de elementos especificados (para el caso de los requisitos y los casos de uso)	62
4.2. Archivos de código fuente	104
4.3. Estadísticas del código fuente de BreakBrain	104
4.4. Estadísticas de las pruebas de estrés en un servidor Debian 64-bits Linux 3.2.0, CPU Intel Core i7 Q720 1.60GHz x 4, RAM 6GB	110
6.1. Ficha de entrenamiento de capacidades mentales	133
7.1. Grado de consecución de objetivos específicos	136

Índice de figuras

1.1. Lóbulos cerebrales	2
3.1. Página principal de http://www.minijuegos.com	12
3.2. Categorías de juegos de http://www.minijuegos.com	13
3.3. Juegos multijugador en http://www.minijuegos.com	14
3.4. http://www.miniclip.com	15
3.5. http://www.minigamesfreak.com	16
3.6. http://www.juegosfan.com	17
3.7. http://www.minigameonline.net	18
3.8. http://www.lumosity.com	19
3.9. http://www.brainarena.com	20
3.10. Muestra de las posibilidades gráficas de HTML5 con WebGL	23
3.11. Diagrama sencillo de una neurona	25
3.12. Surcos y giros del cerebro humano	26
3.13. Tubo neural. A la izquierda el aspecto del tubo en la etapa más temprana de desarrollo, donde pueden apreciarse las 3 partes principales: prosencéfalo, mesencéfalo y rombencéfalo. A la derecha, en una etapa posterior, el prosencéfalo ha pasado a formar el telencéfalo y el diencéfalo.	30
3.14. Subsistemas simpático y parasimpático del SNA	33
3.15. Localización del cerebelo	34
3.16. Hemisferios cerebrales (vista frontal)	34
3.17. Corteza del cíngulo anterior	36
3.18. Lóbulos cerebrales	37
3.19. Neurona piramidal	39
3.20. Corteza orbital-frontal	40
3.21. Áreas de Brodmann	41
3.22. Hipotálamo. Tálamo. Hipocampo. Cerebelo. Cuerpo calloso. Neocortex. Tallo cerebral	47
3.23. Ejemplo de partitura de piano compleja, extraída de una pieza de Kaikhosru Shapurji Sorabji	52

4.1. Proceso Unificado de Desarrollo: etapas, entregables e iteraciones	60
4.2. Diagrama de despliegue del sistema completo	76
4.3. Diagrama de casos de uso	77
4.4. Diagrama de interacción del caso de uso UC-1 (flujo normal)	80
4.5. Diagrama de interacción del caso de uso UC-3 (flujo normal)	83
4.6. Diagrama de interacción del caso de uso UC-4 (flujo normal)	85
4.8. Estadísticas del código fuente: los lenguajes de BreakBrain	105
4.7. Diagrama de interacción del caso de uso UC-12 (flujo normal)	111
4.9. Resultado parcial de tests unitarios	112
4.10. Resultados de la batería de tests funcionales	113
4.11. Diseño del test de estrés	114
4.12. Resultados de las pruebas de estrés	114
 5.2. Patrón MVC en el cliente de BreakBrain	116
5.3. Patrón MVC en el servidor de BreakBrain	117
5.4. Diseño modular del sistema	118
5.1. Arquitectura detallada del sistema	120
 6.1. Decisiones de ejemplo para juegos de razonamiento cuantitativo: ¿Qué cuadrado de cada pareja tiene un valor mayor?	124
6.2. Situación de ejemplo para un juego de concentración. El jugador indica (por ejemplo mediante las teclas de dirección), la dirección en la que el elemento central apunta.	125
6.3. Escenario de ejemplo para un juego multijugador de orientación espacial . .	126
6.4. Ejemplo de prueba para ejercitarse el control de impulsos	127
6.5. Ejemplo de prueba para entrenar la comutación de tareas	127
6.6. Diagrama de capacidades y habilidades mentales	128
6.7. Juegos, partidas y actividades puntuables	130
 A.1. Iteración I1	144
A.2. Iteración E1	144
A.3. Iteración E2	145
A.4. Iteración C1	146
A.5. Iteración C2	146
A.6. Iteración C3	147
A.7. Iteración C4	147
A.8. Evolución iterativa del desarrollo	148

B.1. Página de autenticación de BreakBrain	152
C.1. Logo del juego “Careto”	159
C.2. Fondo del juego “Careto”	159
C.3. Imágenes de juego para “Careto”	160
C.4. Jerarquía de directorios para “Careto”	161

Índice de listados

3.1. Ejemplo de inclusión de código PHP en HTML	19
3.2. Servidor web básico implementado con NodeJS	21
C.1. Esquema de la parte servidora de un juego para BreakBrain	162
C.2. Script servidor de “Careto”	163
C.3. Esquema de la parte cliente de un juego para BreakBrain	167
C.4. Script servidor de “Careto”	168

Listado de acrónimos

SNA	Sistema Nervioso Autónomo
REM	Rapid Eye Movement (Fase del sueño)
TOC	Trastorno obsesivo compulsivo
TEPT	Trastorno de estrés post-traumático
SFC	Síndrome de Fatiga Crónica
IRM	Imagen por Resonancia Magnética
VBM	Morfometría Basada en Vóxel
PUD	Proceso Unificado de Desarrollo
RUP	Rational Unified Process
MVC	Modelo-Vista-Controlador
SGBD	Sistema Gestor de Base de Datos
PAAS	Platform As A Service
SAAS	Software As A Service
KDD	Knowledge Discovery in Databases
SIP	Surco Intra-Parietal
API	Application Programming Interface (Interfaz de Programación de Aplicaciones)
OOTB	Out-Of-The-Box (que funciona directamente, sin necesidad de instalaciones adicionales)
WWW	World Wide Web
SMTP	Simple Mail Transfer Protocol (Protocolo de Transferencia Simple de Correo electrónico)
FTP	File Transfer Protocol (Protocolo de Transferencia de Archivos)
P2P	Peer-To-Peer (Entre pares)
IRC	Internet Relay Chat (Retransmisión de Chat por Internet)
IIS	Internet Information Services
JSP	Java Server Pages

ASP	Active Server Pages
VoIP	Voice over IP (Voz sobre IP)
SSH	Secure SHell
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
SSL	Secure Socket Layer
HTML	HyperText Markup Language (Lenguaje de Marcado de Hipertexto)
CSS	Cascade Style Sheet
GET	Petición HTTP de recuperación datos de un recurso
POST	Petición HTTP de envío de datos para ser procesados por el servidor
DOM	Document Object Model
NPM	Node Package Manager
AWS	Amazon Web Services
WORE	Write Once, Run Everywhere

Agradecimientos

Escribe aquí algunos chascarrillos simpáticos. Haz buen uso de todos tus recursos literarios porque probablemente será la única página que lean tus amigos y familiares. Debería caber en esta página (esta cara de la hoja).

Sergio

A alguien muy querido y/o respetado

Capítulo 1

Introducción

ESTE primer capítulo pretende ofrecer una visión global de lo que el lector encontrará en este texto. El objetivo del presente es la construcción de un sistema informático completo, extensible y centrado en la mejora de diferentes habilidades mentales. No obstante, de ningún modo se centra exclusivamente en aspectos de la Ingeniería Informática, sino que ofrece un componente importante de estudio biológico centrado en comprender el funcionamiento del cerebro humano. Por supuesto, ese componente neurológico se plantea desde un punto de vista no experto en el tema, lo que facilita su comprensión por parte de personas ajena al campo de la biología o la medicina.

Dado que de cara al usuario final del sistema BreakBrain es una red social, en primer lugar se realizará una introducción al aspecto social del Internet de hoy en día. Posteriormente se darán algunas pinceladas sobre el funcionamiento del cerebro humano, introduciendo conceptos elementales que más adelante serán desarrollados en profundidad. Por último, se ofrecerá una aproximación breve al entrenamiento cerebral mediante juegos.

1.1. Redes sociales

Si hay algo que caracteriza, por encima de cualquier otra cualidad, a los sitios web más importantes y con más visitas actualmente, es sin duda alguna el componente social que poseen. La compartición de contenido entre internautas ha cobrado gran protagonismo en los últimos años, enmarcada en webs cada vez más dinámicas que mejoran la experiencia de usuario.

Las redes sociales constituyen el máximo exponente de esa posibilidad de compartición, permitiendo mantener diferentes tipos de relaciones sociales entre los integrantes de las mismas. En la actualidad existen numerosas redes sociales, más o menos importantes. Las más conocidas y de mayor éxito son redes sociales dedicadas al ocio, en las que los usuarios comparten fotografías y comentarios, crean eventos, etc. No obstante, existen redes sociales de muy diversa índole en Internet, dedicadas al deporte, al cine, al desarrollo software, etc. No debemos ver esta abundancia como una sobresaturación de mercado, sino sencillamente entender que el aspecto social es la esencia de la web hoy en día.

1.2. El cerebro humano

El cerebro humano tiene una estructura similar al del resto de animales, pero destaca por su gran tamaño (en comparación con el resto del cuerpo). Este mayor tamaño proviene especialmente de lo que se conoce como *corteza cerebral* —o *córtex*—. El córtex es una cubierta que envuelve al cerebro, con una textura repleta de pliegues que posibilita que quepa en el cráneo.

La corteza cerebral se divide físicamente en dos hemisferios similares, que se encuentran, a su vez, divididos funcionalmente en cuatro lóbulos (figura 1.1):

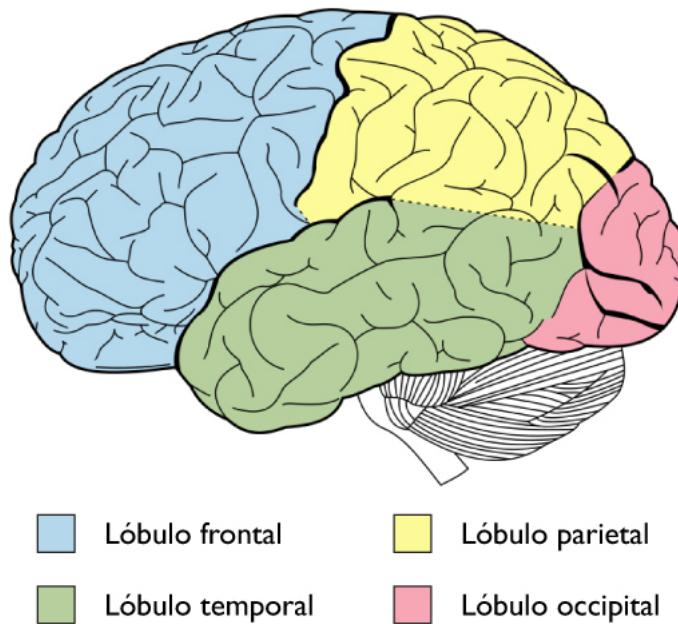


Figura 1.1: Lóbulos cerebrales

■ Lóbulo frontal

Controla la capacidad de movimiento, razonamiento, resolución de problemas, lenguaje y emociones.

■ Lóbulo parietal

Se encarga de las percepciones sensoriales externas (como manos y pies): sensibilidad, tacto, percepción, presión, temperatura y dolor.

■ Lóbulo temporal

Se encarga de la audición, equilibrio y coordinación. Recibe y procesa información de los oídos y regula las emociones y motivaciones, como ansiedad, placer e ira.

■ Lóbulo occipital

Se encarga del procesamiento de la visión y la producción de imágenes.

A lo largo de este texto se estudia la composición del cerebro humano en detalle, atendiendo tanto a aspectos arquitecturales como funcionales. El lector experto en el tema encontrará el análisis bastante superficial, dado que ha sido realizado desde el biológicamente inexperto punto de vista de la ingeniería. No debemos olvidar que el objetivo del presente escrito es la elaboración de un sistema informático. No obstante, se han utilizado fuentes ampliamente reconocidas en el campo de la medicina.

El lector se verá sorprendido con pequeñas curiosidades históricas sobre las que se apoyan los conceptos explicados durante el análisis cerebral. El objetivo de las mismas no es otro que amenizar su lectura, ayudando a entender cómo se ha llegado a deducir el funcionamiento de cada área del cerebro.

1.2.1. La neuroplasticidad

El cerebro humano es un sistema dinámico de redes neuronales con gran potencial de evolución. La plasticidad cerebral, o *neuroplasticidad*, es precisamente esa capacidad dinámica que posee el cerebro, que le permite mejorar con el aprendizaje y repetición de ciertos ejercicios, y en la que se basa este proyecto. En la sección 3.5 se estudiará en profundidad este concepto, mencionando la evolución de su sensibilidad en base a la evolución del cuerpo humano y comentando algunos estudios interesantes sobre el tema.

Se relacionará la neuroplasticidad con el dolor y estrés crónicos, y se introducirá el interesante concepto del *cambio cualitativo*, comparándolo con el entrenamiento por repetición e introduciendo así la idea del entrenamiento cerebral mediante juegos.

1.3. El entrenamiento cerebral mediante juegos

Si hay algo que resulta realmente atractivo para el ser humano es la posibilidad de superar sus limitaciones, ya sea de forma física —mediante el deporte— o de forma mental —mediante el entrenamiento cerebral—. En este sentido no existe una forma más amena de hacerlo que mediante la práctica de pequeños juegos.

Existen diversos artículos dedicados, total o parcialmente, al estudio del entrenamiento cerebral mediante la práctica de juegos. Algunos artículos, como [Hac11], destacan el éxito de los juegos dedicados al entrenamiento cerebral, como el “Dr. Kawashima’s Brain Training”, de Nintendo. Otros utilizan la mejora obtenida mediante juegos como punto de partida para realizar otros estudios. Un buen ejemplo es [Hir04], donde se estudia el efecto de la práctica de juegos sobre la percepción espacial de los seres humanos, comparando las mejoras obtenidas en función de la edad y el sexo.

El capítulo 6 se dedica al estudio del cerebro y desglose del conjunto de habilidades mentales estimulables. Concretamente, en la sección 6.1 se detalla el procedimiento del entrenamiento de las distintas habilidades mentales mediante los juegos de BreakBrain.

1.4. Estructura del documento

A continuación se ofrece una pequeña descripción del contenido de cada capítulo de este documento:

Capítulo 1: Introducción

Este capítulo sirve como presentación del proyecto, explicando el contexto en el que se enmarca y dando unas primeras pinceladas sobre el desarrollo del mismo.

Capítulo 2: Motivación y objetivos

En este capítulo se presentan las motivaciones que desembocaron en el desarrollo de Breakbrain, así como los objetivos que persigue el mismo, tanto el objetivo general como los diferentes objetivos específicos en los que se desglosa.

Capítulo 3: Antecedentes

En este capítulo se presenta un análisis sobre el estado del arte en el que se enmarca el proyecto. Se analizarán las tecnologías y herramientas en las que se basa su desarrollo, así como algunos sistemas software precedentes que han servido de inspiración u ofrecen cierta semejanza con el proyecto que nos ocupa.

Capítulo 4: Método de trabajo

En este capítulo se explica la metodología seguida durante el desarrollo del sistema informático que supone este proyecto. Se hará un recorrido detallado por todas las etapas del mismo, desde el primer estudio de requisitos hasta la realización de pruebas de diferentes tipos, presentando cada artefacto generado durante el proceso.

Capítulo 5: Arquitectura

En este capítulo se ofrece una visión detallada de la composición de BreakBrain. Se hará especial incapié en el despliegue del sistema, los patrones utilizados y la extensibilidad de la plataforma.

Capítulo 6: Resultados

Este capítulo supone un pilar fundamental en el desarrollo del sistema. En él se presenta un estudio general sobre la anatomía y fisiología cerebral humana, desglosando la capacidad mental en las cinco áreas fundamentales: memoria, resolución de problemas, atención, velocidad y flexibilidad. Así mismo, se ofrecerá una especificación precisa sobre el entrenamiento cerebral mediante juegos —para cada una de las áreas mencionadas— diseñado para BreakBrain.

Capítulo 7: Conclusiones y propuestas

En este capítulo se realiza una valoración de los objetivos alcanzados, así como propuestas de futuro y una opinión personal sobre el desarrollo del proyecto.

Anexo A: Iteraciones del PUD

Este anexo contiene el desglose iterativo del desarrollo de BreakBrain, obtenido a

partir de la aplicación del Proceso Unificado de Desarrollo.

Anexo C: Guía para el desarrollador: creación de juegos para BreakBrain

Este anexo ofrece una guía detallada para el desarrollo de juegos para la plataforma.

Anexo D: Manual de usuario

Este anexo ofrece un manual de usuario para aprender a manejar BreakBrain de una forma sencilla.

Anexo B: Guía de instalación

Este anexo muestra el proceso necesario para instalar y ejecutar BreakBrain.

Capítulo 2

Motivación y objetivos

EN estos últimos años el ser humano ha ido adquiriendo una cierta preocupación sobre su salud física, lo que está provocando el aumento de la práctica de actividades deportivas de todo tipo con mayor frecuencia que años atrás. Sin duda alguna esto supone un beneficio importante para las personas. Pero la salud física no lo es todo.

En muchas ocasiones las personas no son conscientes de la importancia que tiene para su vida el hecho de mantener una cierta salud mental. Los medios de comunicación de masas no aportan nada a favor de esa concienciación mencionada, de forma que en muchos casos el cerebro es descuidado en cierta manera. Del mismo modo que el entrenamiento físico suele resultar divertido, el entrenamiento mental no tiene por qué no serlo también, a pesar de lo que muchas personas puedan pensar al principio.

¿Y si existe la posibilidad de realizar ejercicio mental de forma amena? Quizá se podría ejercitar la mente inconscientemente o de forma colateral mientras se centra la atención en la práctica de un video juego divertido. ¿Y si puede hacerse de forma social, en colaboración con más personas?

2.1. Motivación

Dada la importancia de la salud —tanto física como mental— en la vida de una persona, resulta evidente la necesidad del ejercicio y mantenimiento de la misma. Si la práctica de alguna tarea divertida puede contribuir al objetivo propuesto, obviamente dicha tarea debería ser practicada con frecuencia. Por otro lado, dicha tarea deberá resultar entretenida para que las personas sientan atracción por la práctica de la misma.

Los juegos ocasionales son una buena oportunidad de entretenimiento al alcance de cualquier persona, ya sean ejecutados sobre smartphones, videoconsolas u ordenadores personales. Más concretamente, los juegos multijugador ofrecen una experiencia social adicional. Si existe la posibilidad de combinar el entretenimiento de los juegos ocasionales (tanto monojugador como multijugador) con un cierto entrenamiento cerebral, sin duda será una oportunidad genial para obtener esa salud mental mencionada anteriormente de esa forma divertida de la que se ha hablado.

2.1.1. Enfermedad mental

Con el paso de los años, la ciencia ha permitido prolongar la existencia humana hasta límites que hace sólo un siglo eran insospechados. En España, la esperanza de vida se encuentra alrededor de los ochenta años para los hombres y de los ochenta y cinco para las mujeres. Pero ese incremento de la esperanza de vida también ha destapado enfermedades degenerativas que hace años eran desconocidas —sirvan como ejemplo el *Alzheimer* o la *demencia senil*—. Concretamente en España existen en la actualidad un total de ochocientas mil personas a las que se les ha diagnosticado Alzheimer.

Ninguna de las dos enfermedades mencionadas tienen cura a día de hoy, pero eso no es un obstáculo para que se pueda planear una cierta prevención ante las mismas.

2.1.2. Prevención de la degeneración neuronal

Si existe una forma de prevenir ciertas enfermedades mentales propias del envejecimiento humano, y además es susceptible de adaptarse a algún tipo de entretenimiento, nos encontraremos ante una oportunidad expléndida para la mejora de la calidad de vida de las personas de la tercera edad. Quizá exista la posibilidad de reducir considerablemente la probabilidad de padecer una enfermedad mental, ¿por qué no explorarla?

2.1.3. Mejora activa

¿Y si la práctica de una actividad reiterada sirviera de ayuda a esa prevención? Los juegos son un buen ejemplo de actividad a la que las personas se someten voluntariamente con el único propósito de divertirse. Quizá pueda mantenerse ese propósito pero permitiendo además que esos juegos mejoren de alguna forma la capacidad mental del jugador. Si pudiéramos medir esa mejora, cada jugador dispondría de estadísticas evolutivas comparables con las de otros jugadores, y de esta forma se podría someter toda esa recopilación de datos a algún procedimiento de Knowledge Discovery in Databases (KDD) para extraer conocimiento, por ejemplo.

2.2. Objetivos

Con la motivación expresada como punto de partida, BreakBrain se presenta con una serie de objetivos bien definidos. A continuación se detallan dichos objetivos, agrupándolos en dos conjuntos principales:

- **Objetivo general:** A grandes rasgos, qué se pretende conseguir con la realización del sistema informático que supone el proyecto.
- **Objetivos específicos:** Se trata de una serie de *objetivos académicos* —metas de carácter educativo, destinadas a complementar y ampliar los conocimientos adquiridos

durante el estudio de la Ingeniería Informática— y de unos *objetivos funcionales* — metas específicas de la funcionalidad del producto—.

2.2.1. Objetivo general

El objetivo general y detonante motivador para el desarrollo de BreakBrain es la construcción de una plataforma social destinada al entrenamiento de habilidades mentales mediante la práctica de juegos, tanto individualmente como de forma colaborativa y competitiva. La plataforma ofrecerá estadísticas de evolución y permitirá el seguimiento (o suscripción a novedades) de otros usuarios. Así mismo, la plataforma deberá ser extensible, de forma que cualquier desarrollador pueda desarrollar sus propios juegos e integrarlos en el sistema, para que otros usuarios puedan jugar con ellos.

2.2.2. Objetivos específicos

Desde el punto de vista académico, un proyecto tan ambicioso como BreakBrain supone el enfrentamiento del alumno a un problema de gran embergadura, mucho mayor que cualquier otro proyecto que se haya afrontado durante el estudio de la licenciatura de Ingeniería Informática.

Como dice el proverbio árabe:

Quien se empeña en pegarle una pedrada a la luna no lo conseguirá, pero terminará sabiendo manejar la honda.

Con lo anterior no se pretende insinuar que el desarrollo del presente proyecto se inicie pensando en el fracaso, nada más lejos de la realidad. Lo que se tiene claro desde el primer momento es que el aprendizaje adquirido durante el análisis, diseño e implementación de la plataforma será suficiente motivo como para que merezca la pena un gran esfuerzo, independientemente del grado de éxito que alcance BreakBrain.

Así pues, algunos objetivos académicos y funcionales que se persiguen con este proyecto son los siguientes:

- Afrontar el desarrollo de una red social desde cero, sin el uso de frameworks especializados, sino contemplando la creación de toda la plataforma paso a paso.
- Investigar y prever relaciones entre usuarios. Crear un sistema de recomendación basado en las expectativas de entrenamiento cerebral y resultados de cada usuario.
- Crear una plataforma extensible mediante un pequeño framework de desarrollo de videojuegos sencillos. El objetivo es que cualquier desarrollador pueda extender la funcionalidad de BreakBrain mediante sus propios juegos.

- Liderar un proyecto de software libre ambicioso, gestionando la integración de juegos de terceros.
- Aprender a utilizar correctamente los nuevos estándares de HTML5, CSS3.
- Afrontar el desarrollo de aplicaciones web de tiempo real mediante el uso de WebSockets.
- Familiarizarse y ganar experiencia con el lenguaje JavaScript y su uso en el lado del servidor mediante NodeJS.
- Aprender a utilizar bases de datos documentales no basadas en el lenguaje SQL.

Capítulo 3

Antecedentes

A lo largo de este capítulo se presenta el estudio previo realizado como primer paso a la hora de abordar el proyecto que nos ocupa. En primer lugar se ofrece un análisis de los sistemas software que comparten cierta similitud con BreakBrain (secciones 3.1 y 3.2), tanto desde el punto de vista de un usuario común como mediante una visión técnica de los mismos. Posteriormente se ofrece un repaso acerca de las tecnologías web más utilizadas y extendidas en la actualidad (sección 3.3) para, por último, centrar la atención en el desarrollo de mini-juegos preparados para su ejecución web (sección 3.3.2), estudiando las diferentes posibilidades de implementación de los mismos. Como colofón se realiza un estudio de las modernas posibilidades para el desarrollo web de tiempo real (sección ??).

Mientras que los primeros análisis tratan de resultar amenos a cualquier tipo de lector, las últimas secciones están orientadas a un público con cierto nivel de conocimientos técnicos, dado que supone un paso importante en las decisiones tecnológicas tomadas para abordar el proyecto.

3.1. Sitios web de juegos online

En la actualidad son cada vez más numerosos los sitios web orientados al ocio. De entre ellos, algunos han ganado popularidad por centrarse en ofrecer multitud de pequeños juegos casuales, sin ninguna finalidad más allá de ofrecer un breve tiempo de diversión. Estos juegos normalmente no constan de una historia larga a la que enfrentarse en varias sesiones de juego, sino que se trata de juegos con unas pocas pantallas o una historia de apenas unos minutos.

Dada la considerable afluencia de público a este tipo de webs, resulta muy común el emplearlas como negocio mediante el uso de la publicidad para proporcionar ingresos, lo que en muchas ocasiones ofrece una experiencia de usuario que deja bastante que desear.

A continuación se ofrece un pequeño análisis sobre algunas de estas webs:

Minijuegos.com

Con un *Pagerank*¹ de 6/10, Minijuegos [Min] se sitúa a la cabeza de las webs de este tipo. Después de llevar online casi una década, se trata de una muestra clara del crecimiento de Internet y las tendencias que han ido evolucionando a lo largo del tiempo con respecto a las tecnologías utilizadas para la implementación de los juegos (que serán estudiadas en la sección 3.3.2).

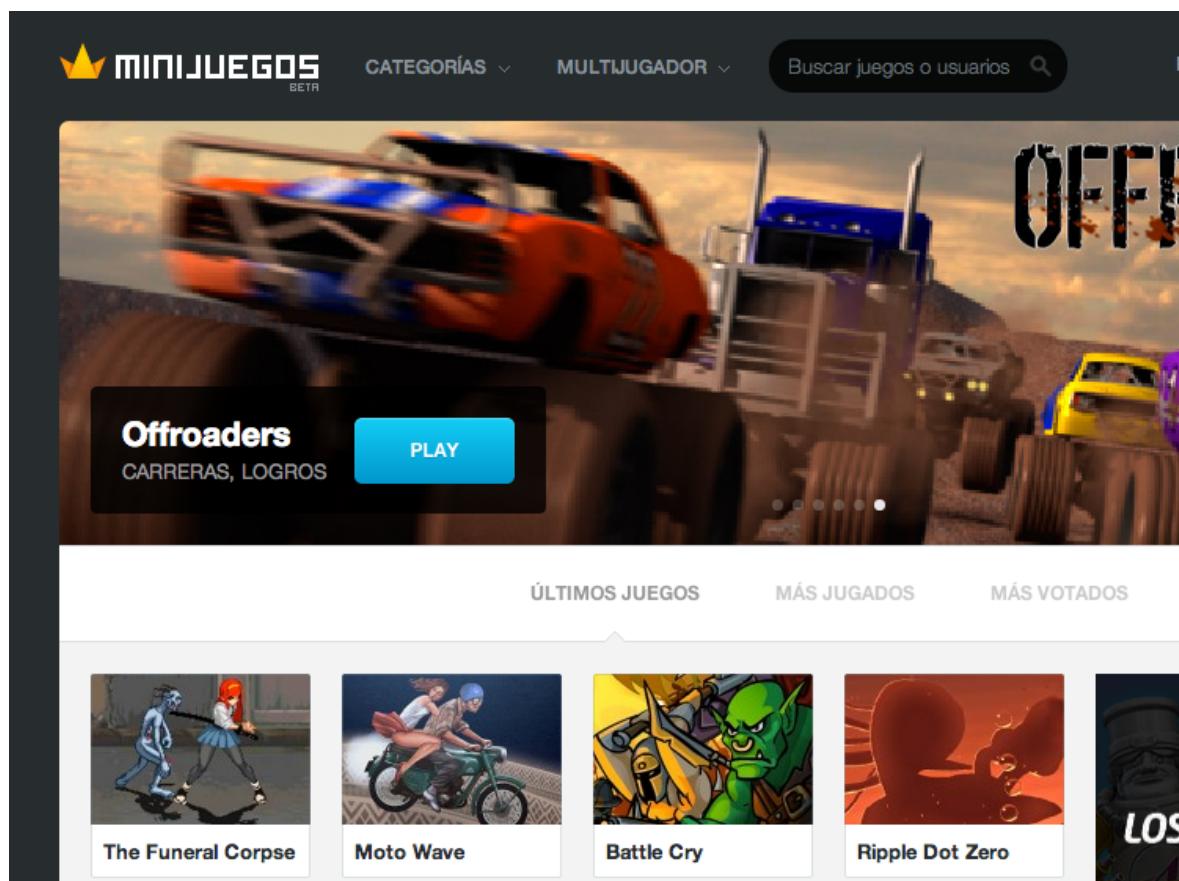


Figura 3.1: Página principal de <http://www.minijuegos.com>

Su continua renovación, así como su apuesta por un modelo de publicidad poco abusivo, han garantizado su éxito durante años. En la actualidad, ofrecen recursos para que los desarrolladores tengan aún más facilidades para crear sus juegos para la plataforma, soportando varias tecnologías diferentes.

Los juegos de este servicio se clasifican en función de 10 categorías (ver figura 3.2): Acción, Aventuras, Carreras, Clásicos, Deportes, Estrategia, Gestión, Habilidad, Mesa y Colecciones.

¹El *Pagerank*, definido oficialmente como un “método de asignación de importancia a nodos de una base de datos enlazada” por [Pag07], es un algoritmo mediante el cual Google puntuá a cada sitio web según su importancia o influencia en Internet.

Cada una de las categorías contiene entre 5 y 10 subcategorías diferentes, lo que abarca un amplio surtido de juegos de todo tipo. Como último detalle, Minijuegos ofrece además algunos juegos multijugador (ver figura 3.3);



Figura 3.2: Categorías de juegos de <http://www.minijuegos.com>

Un punto fuerte de Minijuegos es el conjunto de herramientas y recursos que ofrecen para el desarrollo de juegos por parte de terceros. Ponen APIs de diferentes tecnologías a disposición de los desarrolladores. Esto garantiza un éxito prolongado de la plataforma, dado que la sencillez de programación de juegos para la misma motiva a los desarrolladores a ampliar su catálogo cada día.

Los juegos están desarrollados en Flash (requieren la instalación de un plugin) y HTML5 (funcionan OOTB con cualquier navegador moderno).

Miniclip.com

Con el mismo pagerank, 6/10, Miniclip ofrece también una gran cantidad de juegos, clasificados en más de 70 categorías —lo cual puede resultar algo abrumador para el usuario—. En la figura 3.4 se muestra la parte de la interfaz de esta plataforma.

Visualmente, y en cuanto a usabilidad se refiere, Miniclip está muy por detrás de Minijuegos. Además se aprecian pequeños defectos, como el hecho de que exista una categoría

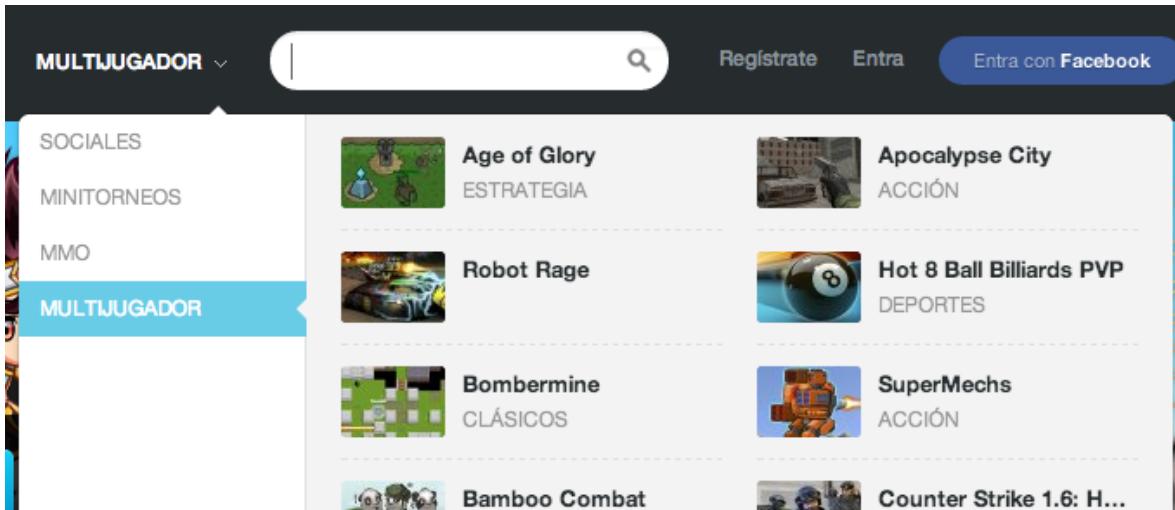


Figura 3.3: Juegos multijugador en <http://www.minijuegos.com>

llamada *Educación* y otra llamada *Educativos*, que dan una sensación de una menor profesionalidad.

Los juegos, tanto multijugador como individuales, están desarrollados en Flash y Unity, por lo que requieren la instalación de plugins para poder ser ejecutados.

Minigamesfreak.com

Bajando bastante el nivel, tanto en cuanto a calidad de la plataforma como a popularidad, se encuentra Minigamesfreak, con un pagerank de 2/10. Se trata de un sitio que recopila juegos individuales de otras plataformas, como Minijuegos. La usabilidad deja mucho que desear, teniendo que pasar por múltiples páginas desde que se hace click sobre un juego hasta llegar a él y poder jugar.

Minigamesfreak tiene un enfoque lucrativo mediante publicidad, algo abusiva, que produce una experiencia de usuario desagradable. Sin prestar ninguna atención al diseño ofrecen una apariencia deficiente y poco atractiva.

Juegofan.com

Al igual que Minigamesfreak, Juegofan ofrece una recopilación de juegos (individuales o monojugador) de otras webs populares, como Minijuegos. No obstante, en este caso se trata de un sitio web más profesional y agradable para el usuario. Con un pagerank de 4/10 denota una popularidad importante en Internet.

La interfaz que ofrece está un poco saturada de publicidad, por lo que la experiencia es mejorable.

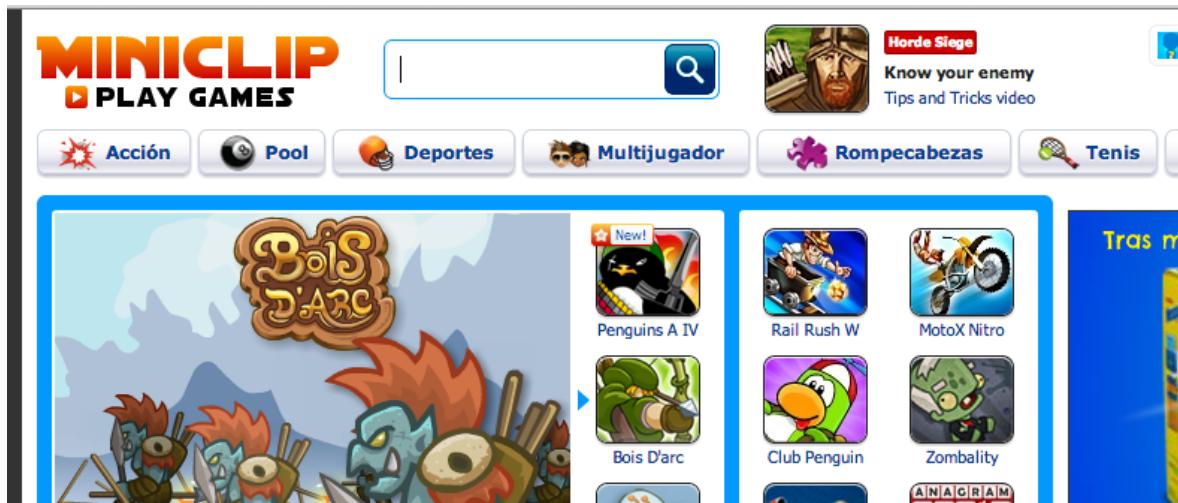


Figura 3.4: <http://www.miniclip.com>

Minigameonline.net

De nuevo una web de recopilación de juegos de otras páginas, de un nivel de madurez y popularidad similar al de Minigamesfreak (pagerank de 2/10). Con una sencillez mayor, la experiencia de juego mejora frente al primero, dado que cargar cualquier juego es inmediato y no te obliga a pasar por páginas intermedias. Además la publicidad no resulta molesta y queda relegada a un segundo plano.

Los juegos que integra están implementados en Flash, y clasificados en 8 categorías entre las que se incluyen, a parte de las clásicas (Carreras, Acción, Barcos, Disparos, Puzzle...), categorías específicas para las webs de las que se han extraído los juegos, como Miniclip.

3.2. Entrenamiento cerebral mediante mini juegos

En las secciones anteriores han sido expuestas algunas de las plataformas de minijuegos más conocidas de Internet. Se trata de webs recopilatorias de juegos casuales de multitud de temáticas, donde los juegos enfocados al ingenio, la memoria o la educación juegan un papel poco importante —indistinguible del de juegos de plataformas, carreras, acción, etc—.

En esta sección se analizarán otras plataformas de juegos, esta vez centradas en juegos de estimulación de habilidades mentales.

Lumosity.com

Con un pagerank de 6/10, se trata de una de las webs más populares de minijuegos orientados a la estimulación de habilidades mentales. Muestra una interfaz sencilla, con un diseño muy agradable y sin publicidad. La ausencia de anuncios se agradece al utilizar la platafor-

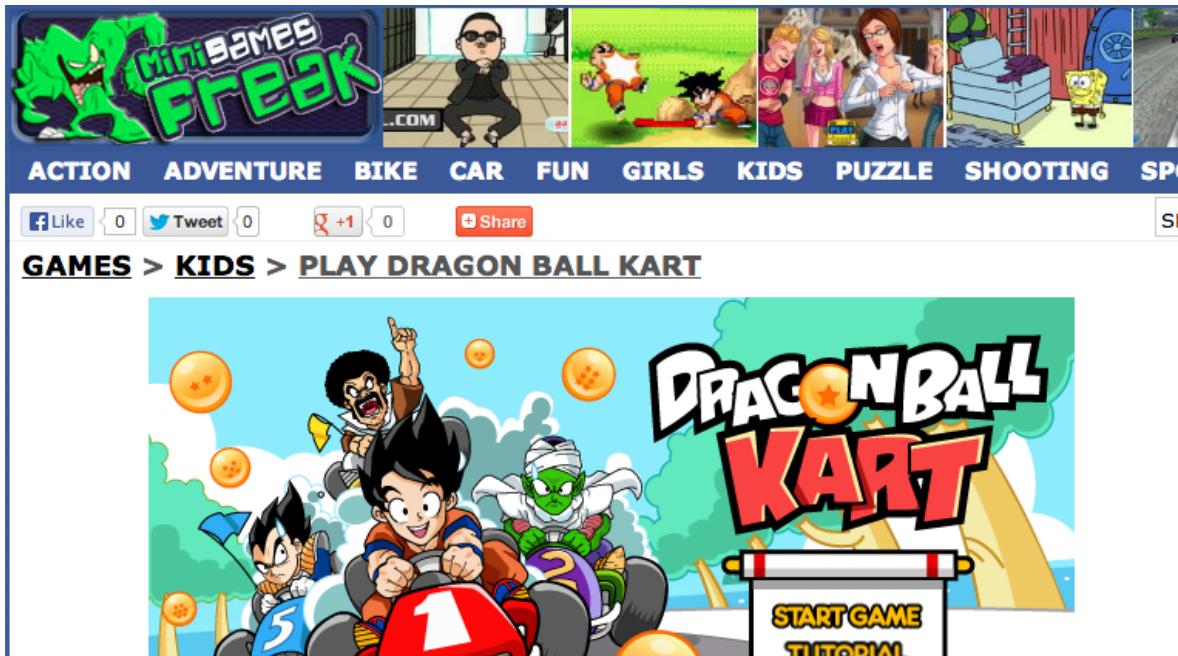


Figura 3.5: <http://www.minigamesfreak.com>

ma, pero tiene un gran inconveniente: el servicio ofrecido es de pago (ver figura 3.8). Las tarifas no son demasiado económicas, por lo que emplear la plataforma como entretenimiento casual queda totalmente descartado para la mayoría de usuarios. De forma gratuita no es posible probar ningún juego.

Lumosity está enfocada sobre todo al entrenamiento rutinario, ofreciendo un seguimiento personalizado para cada usuario.

En esta plataforma los juegos son clasificados en 5 categorías: Velocidad, Memoria, Atención, Flexibilidad y Resolución de Problemas. Todos los juegos son individuales —quedando así anulado cualquier atisbo “social”— y están implementados en Flash, con el inconveniente que ello supone: es necesario tener instalado un plugin para esta tecnología.

Brainarena.com

Con menor popularidad (pagerank de 3/10), Brainarena cambia el enfoque lucrativo hacia un modelo basado en la publicidad, ofreciendo sus servicios de forma totalmente gratuita (como ocurría con las plataformas estudiadas en la sección 3.1) a costa de empeorar considerablemente la experiencia de usuario.

Son sólo 16 juegos los que están al alcance del usuario. La ausencia de categorías para clasificarlos resulta un tanto extraña.

Por otro lado, la jugabilidad es estupenda: todos los juegos muestran una imagen y dan 4 opciones que pueden ser elegidas con las teclas 1, 2, 3 y 4 del teclado.

Figura 3.6: <http://www.juegosenfan.com>

3.3. Desarrollo web moderno

La imparable popularización de Internet ha sido propulsora de una gran evolución tecnológica y social desde que la “*Red de redes*” fue concebida hasta nuestros días. Al principio de la década de los 90 tener un computador personal en casa era un lujo al alcance de pocos, y el acceso a Internet un lujo todavía mayor. Los teléfonos móviles no se empezarían a extender hasta 5 o 10 años después. A día de hoy cualquier persona tiene un ordenador en casa y acceso a Internet. Es más, según [Acc12] el 69 % de los accesos a Internet en la actualidad se realizan desde teléfonos móviles.

Internet es una red descentralizada de servicios como WWW, SMTP, FTP, P2P, IRC, VoIP, SSH, etc. De entre ellos el de mayor éxito es el primero: la World Wide Web (WWW), conocida informalmente como “*La Web*”. WWW es un conjunto de protocolos que permite la consulta de archivos de hipertexto².

La explicación más sencilla del funcionamiento de Internet consiste en la comprensión de dos conceptos: servidor y cliente. Los clientes envían peticiones al servidor, y éste construye y devuelve una respuesta. En el caso de la navegación web, la petición puede ser la solicitud HTTP GET de una página web, y la respuesta el código HTML de la web, que será interpretado y renderizado por el navegador del cliente. Otro ejemplo puede ser una petición HTTP POST de envío de un formulario, y una respuesta de redirección.

²El hipertexto es una herramienta de software con estructura no secuencial que permite crear, agregar, enlazar y compartir información de diversas fuentes por medio de enlaces asociativos. El hipertexto es una de las formas de la *hipermedia* —manipulación de información interactiva en diversos formatos—.

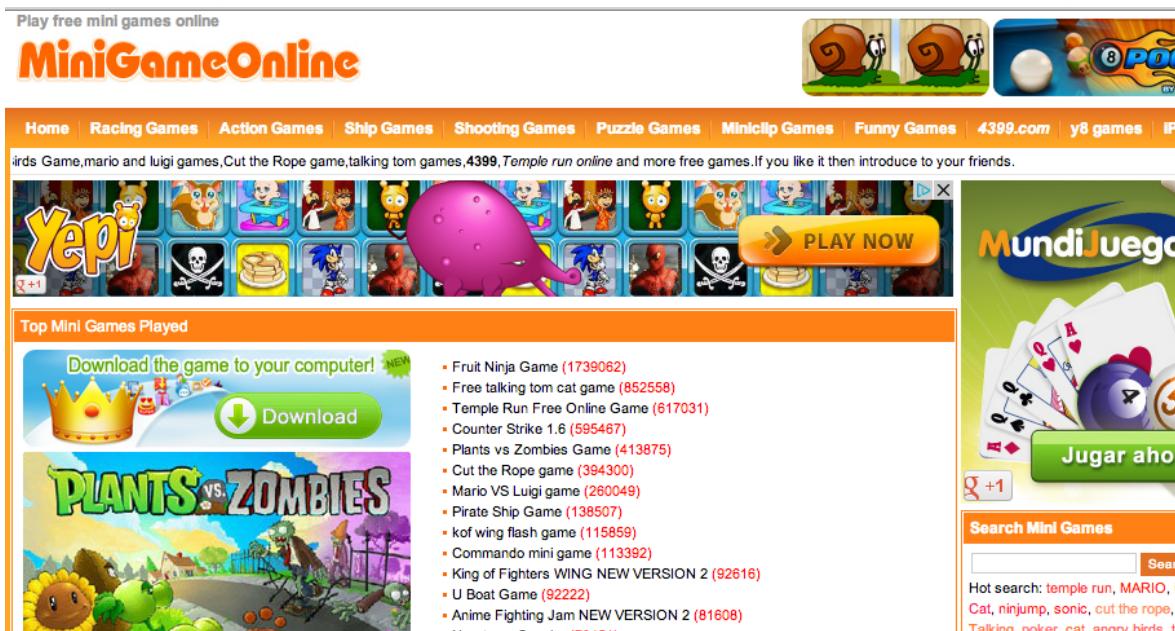


Figura 3.7: <http://www.minigameonline.net>

A continuación se presentan algunas de las tecnologías web más populares, tanto a nivel del servidor como del cliente.

3.3.1. Servidor web

Un servidor web es un computador que realiza, siguiendo un orden riguroso, las siguientes tareas:

1. Recibe una petición de un cliente
2. Procesa la petición
3. Realiza la tarea interna correspondiente a esa petición
4. Construye una respuesta
5. Envía la respuesta al cliente que hizo la solicitud

En la actualidad existen numerosas tecnologías y entornos de ejecución para la construcción y despliegue de servidores web, en casi cualquier lenguaje de programación existente. Cada opción tiene sus ventajas e inconvenientes y es más o menos adecuada dependiendo de la situación y finalidad del servidor.

En el caso concreto de las páginas web, el servidor recibe peticiones GET y POST del navegador web de un cliente y genera código HTML y de *scripting* (JavaScript) que dicho navegador es capaz de interpretar.

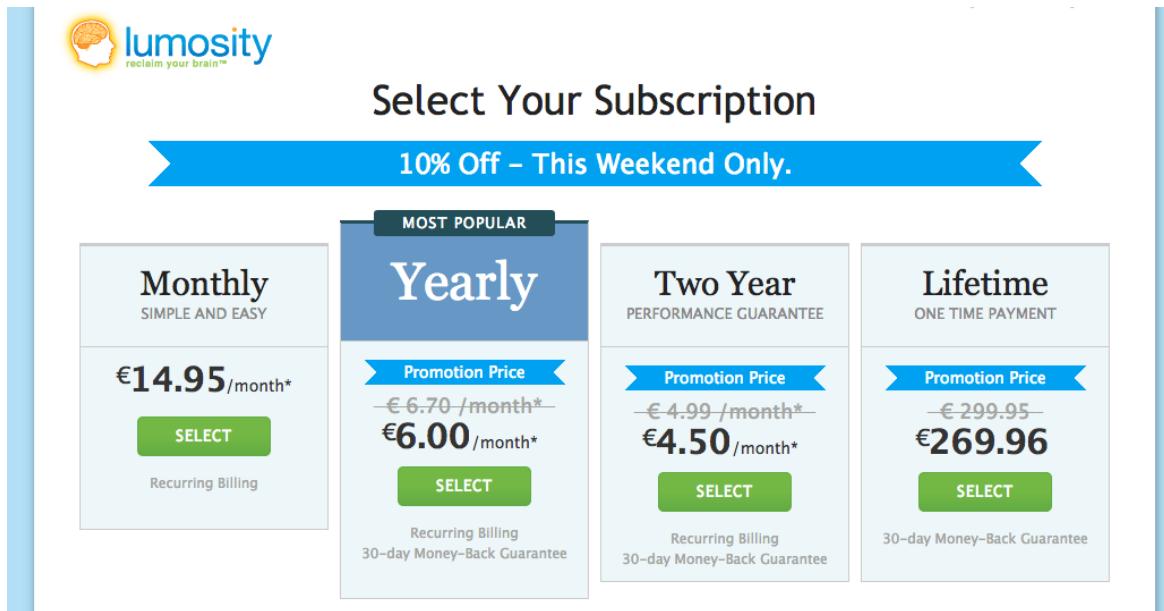


Figura 3.8: <http://www.lumosity.com>

A continuación se exponen algunas de las tecnologías más utilizadas en la creación de servidores web.

PHP

PHP es un lenguaje de *scripts* —scripts del lado del servidor, por lo que no son interpretables por un navegador web cliente— ampliamente utilizado para todo tipo de propósitos. Está orientado especialmente al desarrollo web, y puede ser empotrado en código HTML (ver listado 3.1).

```

1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <meta charset="UTF-8" />
5     <title> Ejemplo basico PHP</title>
6   </head>
7   <body>
8     <?php
9       echo 'Esto es codigo PHP';
10      ?>
11   </body>
12 </html>
```

Listado 3.1: Ejemplo de inclusión de código PHP en HTML

Este lenguaje es software libre, y además de ser interpretado por un servidor web también permite su ejecución en un intérprete de comandos común. En [PHP] se ofrecen recursos útiles para el aprendizaje y primeros pasos del lenguaje.

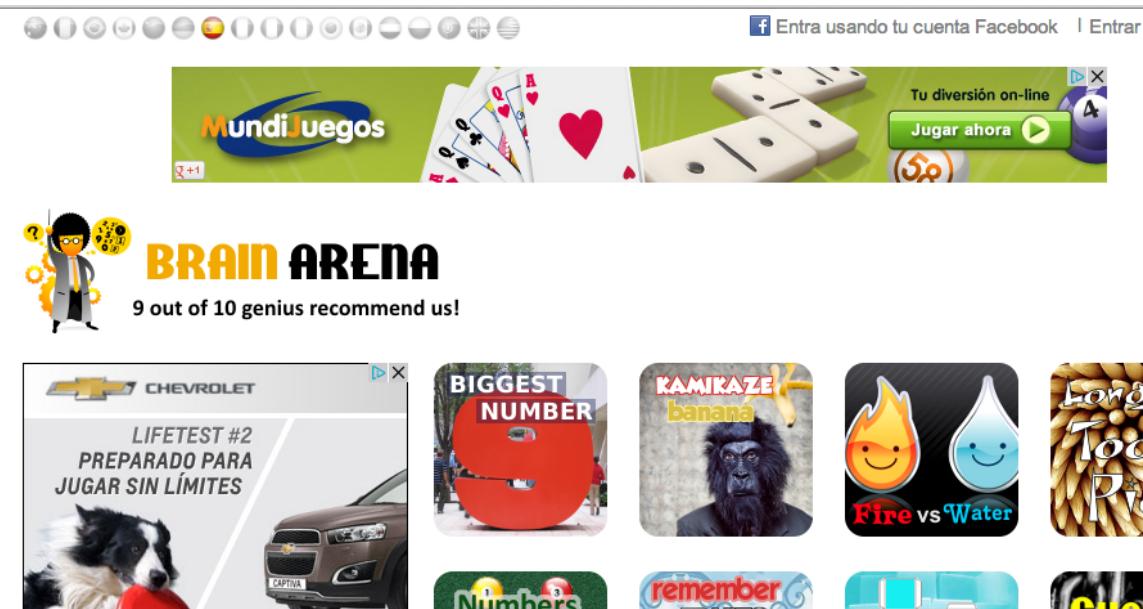


Figura 3.9: <http://www.brainarena.com>

ASP.NET

A diferencia del anterior, éste es un framework diseñado exclusivamente para el desarrollo de páginas web dinámicas y servicios web. Es el sucesor del antiguo ASP, lanzado por Microsoft en 1996. Permite su utilización con cualquier lenguaje soportado por el framework .NET (Java, C#, etc.), y puede ser utilizado únicamente en servidores IIS de Microsoft.

En comparación a otras alternativas interpretadas (como PHP), ASP.NET ofrece la ventaja de trabajar con código compilado, lo que le otorga mejor rendimiento.

JSP (Java Server Pages)

Aunque Java es un lenguaje de propósito general, no es hasta 1999 cuando Sun Microsystems lanza JSP para competir con otros lenguajes y frameworks, PHP y ASP.

Esta tecnología requiere una máquina virtual Java, capaz de interpretar su *bytecode*, y soporta lo que se conocen como anotaciones —comentarios HTML con una sintaxis especial, que permiten la modificación de ciertas partes del código en tiempo de ejecución—.

NodeJS

En los últimos años la popularización de JavaScript como lenguaje de programación web (del lado del cliente) ha crecido de forma considerable, ayudada en parte por el desarrollo del motor de ejecución V8 (creado por Google para el navegador Chrome). Lo que no se pensó hasta 2009 era en la posibilidad de utilizarlo en el lado del servidor. Fue entonces cuando Ryan Dahl lanzó NodeJS, un entorno de ejecución de JavaScript nativo, basado en el motor

V8, centrado en la entrada salida y basado en eventos.

NodeJS es software libre, está enfocado a la escalabilidad y permite la creación de servidores web de manera realmente sencilla. El listado 3.2 muestra la implementación de un servidor HTTP básico que responde “Hello world”.

```
1 var http = require('http');
2
3 http.createServer(function (request, response) {
4     response.writeHead(200, {'Content-Type': 'text/plain'});
5     response.end('Hello World\n');
6 }).listen(8000);
7
8 console.log('Server running at http://127.0.0.1:8000/');
```

Listado 3.2: Servidor web básico implementado con NodeJS

La popularidad de NodeJS resulta asombrosa, teniendo en cuenta que se trata de una tecnología muy nueva. En los cuatro años que lleva en desarrollo cuenta ya con más de 37000 módulos registrados en NPM, lo que da una idea de la calurosa acogida que ha tenido por parte de la comunidad de desarrolladores.

3.3.2. Cliente web

El cliente web es sencillamente un navegador corriente que, principalmente, accede a páginas web y realiza peticiones de formularios. Cuando un usuario accede a una página web, el navegador la solicita al servidor, éste la construye y la devuelve, y el navegador la interpreta y se la muestra al usuario.

Como veremos más adelante, la labor del navegador web va más allá de la mera petición e interpretación de páginas web. Concretamente cuando nos centramos en el desarrollo de videojuegos multijugador, el navegador es también el encargado de mantener la comunicación con el servidor de forma transparente al usuario.

Tecnologías web para desarrollo de videojuegos

Existen diversas tecnologías disponibles para la inserción de contenido gráfico animado e interactivo en una página web. A continuación se expondrán las más importantes, absolutas protagonistas del desarrollo de videojuegos web hoy en día.

■ JApplets

Durante un largo tiempo, los applets de Java (o JApplets) fueron la fuerza dominante en el desarrollo de contenido interactivo web en el lado del cliente. La popularización de Java como lenguaje de propósito general fue en gran parte la responsable. No obstante, con la modernización de HTML y los navegadores web, muchas de las funcionalidades que antes se hacían mediante Java pasaron a estar soportadas OOTB, por lo que esta

tecnología de Java fue perdiendo popularidad. Nótese que los applets de Java requieren de un complemento adicional o plug-in instalado en el navegador para ser ejecutados.

En la actualidad se utiliza bastante poco, habiendo quedado prácticamente en desuso en el ámbito de los videojuegos (en parte por el bajo rendimiento gráfico que ofrece).

■ Flash y Shockwave

Las tecnologías Flash y Shockwave, ambas desarrolladas por Macromedia (empresa que más adelante sería absorbida por Adobe), supusieron una gran evolución en el desarrollo de contenido interactivo web. Hoy en día son ampliamente utilizadas, y ofrecen un rendimiento muy bueno en la reproducción de contenido multimedia —Shockwave soporta aceleración gráfica—.

Se trata de dos tecnologías complementarias, siendo Shockwave algo más amplia (es capaz de reproducir contenido Flash, lo que no ocurre a la inversa). Requieren de un plug-in para el navegador web. En el caso de Flash, el plug-in se encuentra en el 95 % de los computadores de escritorio, mientras que en el caso de Shockwave sólo en el 40 %, según [W3T].

En la actualidad está ocurriendo algo similar a lo que pasó con los applets de Java, y es que HTML y los navegadores modernos son capaces de llevar a cabo tareas que antes sólo podía hacer Flash, por lo que cada vez se está empezando a utilizar menos para algunas aplicaciones, debido en parte a la ventaja de no necesitar de plug-ins adicionales.

■ Unity web player

Unity es un “*ecosistema de desarrollo de videojuegos*”, según [Uni]. Bajo la filosofía Write Once, Run Everywhere (WORE) permite el desarrollo de videojuegos compilables de forma automática para múltiples plataformas. El gran problema de Unity es que es de pago —con un precio que oscila entre los 1500 y los 6500 euros—, por lo que no se encuentra al alcance de todos los bolsillos.

Entre las ventajas que ofrece frente a otras alternativas, las más importantes son:

- Rendimiento
- Calidad gráfica
- Soporte de plug-ins

Unity web player es el componente que permite la ejecución en un navegador de videojuegos creados con Unity. Tal y como ocurría con Flash y Shockwave, si el navegador web no cuenta con el plugin instalado no será capaz de ejecutar el componente.

■ HTML5

El lenguaje de marcado HTML ha ido adaptándose a las necesidades surgidas a causa de la evolución de Internet. En este sentido, la especificación de la última versión (aún

no cerrada por completo) incluye funcionalidades que antes eran propias de Flash o los applets de Java, y que ahora cualquier navegador web moderno es capaz de desempeñar. Buen ejemplo de estas funcionalidades son la inclusión de audio y vídeo, así como el objeto canvas, que posibilita la creación de videojuegos complejos con un rendimiento excelente.

La tecnología WebGL, parte de la especificación de HTML5, permite incluso utilizar la aceleración gráfica para reproducir contenido 3D de alta calidad. No es de extrañar que HTML5 esté acabando poco a poco con otras tecnologías (que requieren complementos adicionales para ser ejecutadas).

En la figura 3.10 se muestra un ejemplo de la calidad gráfica ofrecida por HTML5. En [Gooa] existe un gran surtido de ejemplos para experimentar con este tipo de contenidos web.

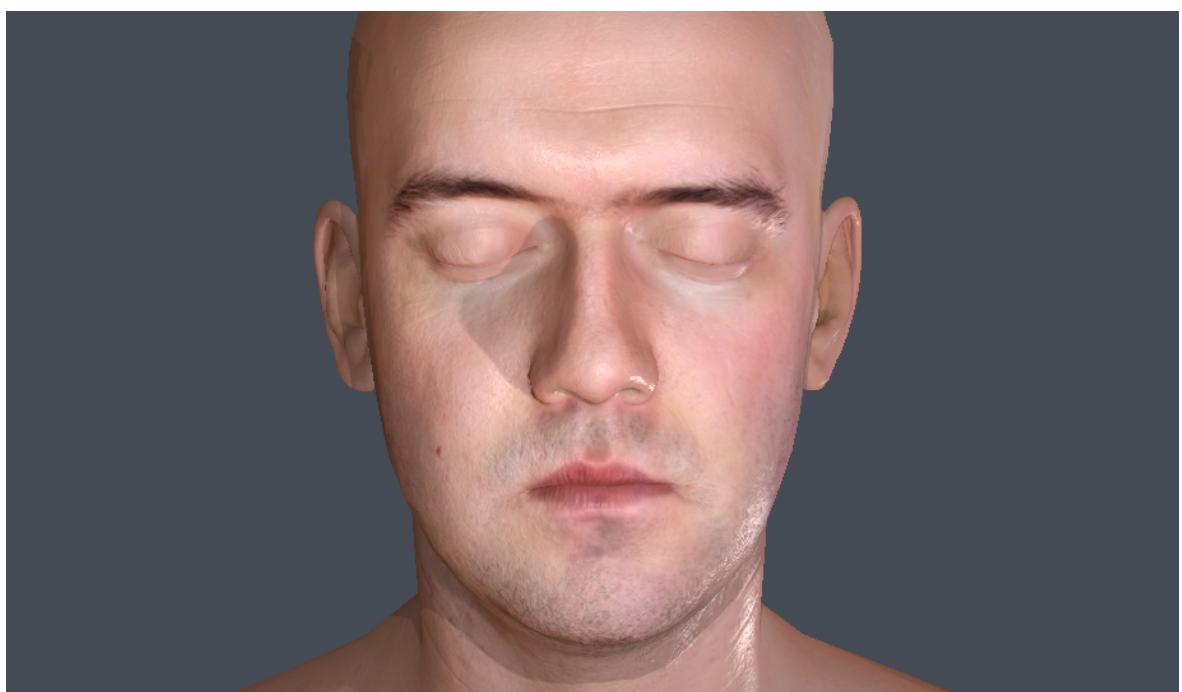


Figura 3.10: Muestra de las posibilidades gráficas de HTML5 con WebGL

Comunicación web ininterrumpida: websockets

WebSocket es una tecnología que implementa sockets de comunicación entre servidores y navegadores web (aunque pueden utilizarse por cualquier tipo de aplicación cliente/servidor). Proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP.

Se trata de una tecnología que aún se encuentra en desarrollo. La API de WebSocket está siendo normalizada por el W3C, y el protocolo WebSocket, a su vez, está siendo normalizado

por el IETF.

Como las conexiones TCP ordinarias sobre puertos diferentes al 80 suelen ser bloqueadas por los administradores de redes, el uso de esta tecnología proporcionaría una solución mediante la multiplexación de diferentes servicios WebSocket sobre un único puerto TCP.

La especificación del protocolo WebSocket define dos nuevos esquemas de URI:

- **wss**: para conexiones cifradas
- **ws**: para conexiones no cifradas

El resto de componentes del URI se definen con la sintaxis genérica de URI.

3.4. Anatomía y fisiología del cerebro humano

El primer paso antes de tratar de emplear conocimientos técnicos para mejorar las habilidades mentales de los seres humanos es, evidentemente, la familiarización con la fisiología³ del cerebro, pues de nada sirve tratar de mejorar algo sin conocerlo previamente. En este sentido se ha realizado un esfuerzo adicional por ofrecer esta base científica de una forma amena para el lector. Así pues, en las siguientes secciones se realizará un recorrido por los aspectos más relevantes en cuanto a la estructura y funcionamiento del “órgano más importante de todos” [Jac09].

Una vez adquiridos esos conocimientos previos sobre la materia, nos encontraremos en disposición de diseñar e implementar los mecanismos y métricas más adecuados para la mejora de las funciones cerebrales humanas.

3.4.1. Estructura cerebral

El cerebro de una persona viva es de color rosa, algo pálido. “*Conservado en formol, el cerebro endurece y pierde su color*”, afirma [Cze01]. En esta sección hablaremos de las células que constituyen la estructura el cerebro. Estas células se organizan en dos grupos:

- núcleo
- corteza (o córtex⁴)

Materia gris, materia blanca y células gliales

El cerebro es en gran parte de color gris, y es por ello que se emplea la denominación *materia gris*. La materia gris está formada por miles de millones de neuronas. Tiene la “*consistencia blanda de harina de avena cocida*”, afirma [Cze01]. En la figura 3.11 puede apreciarse la forma general de una neurona.

³Ciencia biológica que estudia las funciones de los seres orgánicos.

⁴A lo largo de este texto se emplearán indistintamente los términos *corteza* y *córtex*, dado que así se hace en la literatura médica en nuestro idioma.

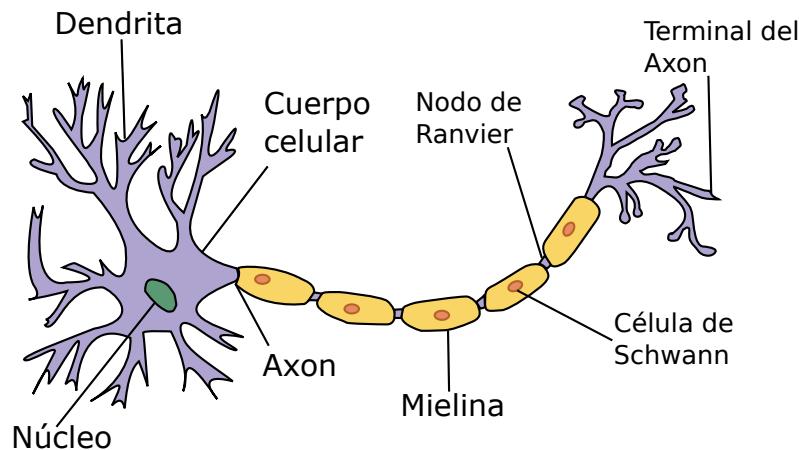


Figura 3.11: Diagrama sencillo de una neurona

Puesto que hay tantas neuronas, es tentador considerarlas diminutas en términos de su alcance, pero en realidad algunas neuronas tienen axones muy largos. Cuando una multitud de axones se agrupan, aparecen como *materia blanca*. En [Cze01] se describe la materia blanca como “*una colección de axones largos, aislados, ramas salientes de neuronas que conectan las diversas partes del cerebro entre sí*”, debiendo el color blanco a “*la mielina, un reluciente material aislante que envuelve los largos axones*”.

[Dam94] proporciona una descripción sucinta de las materias blanca y gris: “*La materia gris corresponde en gran parte a las colecciones de cuerpos de células nerviosas, mientras que la materia blanca corresponde en gran parte a los axones, o fibras nerviosas, queemanan de los cuerpos celulares en la materia gris*”. Las neuronas se encuentran “*apoyadas por las células gliales*”, que son “*esenciales para la actividad del cerebro*”

Por otro lado, en [Cze01] encontramos una buena descripción de las células gliales: “*Cada neurona está rodeado por entre diez y cincuenta células gliales, que constituyen la mayor parte del cerebro. Las células gliales apoyan a la neurona de varias maneras: proporcionan el soporte estructural, aíslan las ramas neuronales con mielina para acelerar la transmisión de señales eléctricas por su axones, mantienen la nutrición de la neurona, ayudan en la eliminación de residuos, e incluso establecen los marcadores que conducen las ramas de una neurona en desarrollo a su destino correcto*”. Como detalle curioso, en el mismo libro se afirma que “*un estudio del cerebro de Albert Einstein demostró que es posible que tuviera un exceso de células gliales*”.

Cortezas y núcleos cerebrales

“*La materia gris se presenta en dos variedades*”, afirma [Dam94]. “*La primera variedad de materia gris es la formada neuronas que se colocan en capas, como en una torta, para formar una corteza. Ejemplos son la corteza cerebral que cubre los hemisferios cerebrales y la corteza cerebelosa que envuelve el cerebelo. En la segunda variedad de materia gris, las*

neuronas no están estratificadas, sino que se organizan como cacahuetes dentro de un tarro de cristal. Esta formación constituye un núcleo”.

El mejor ejemplo de neuronas organizadas en forma de corteza es la *neocorteza* o *neocórtex*, a menudo llamada *corteza cerebral*. El neocórtex se pliega de una manera que permite a una superficie más grande encajar dentro de los confines del cráneo humano. Los anatomistas llaman *surcos* a los pliegues corticales, y *giro* a la zona lisa entre dichos pliegues. En la figura 3.12 pueden apreciarse tanto los surcos como los giros de la corteza cerebral.

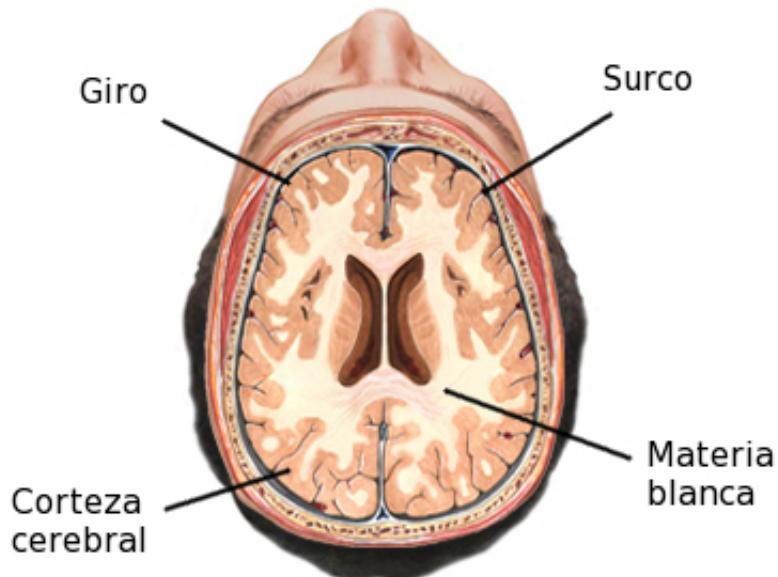


Figura 3.12: Surcos y giros del cerebro humano

Las neuronas

Los componentes principales de una neurona se detallan en la figura 3.11. La longitud del axón puede ser mucho más larga de como se muestra en la ilustración, tal y como ya se ha dicho cuando se han introducido los conceptos de *materia gris* y *materia blanca*. Asimismo, en [Sap] se resalta la complejidad de una sola neurona, explicando que puede tener una media de 10.000 dendritas y 10.000 terminales de axón. La comunicación entre las neuronas es tanto química como eléctrica, así que el término electroquímica se utiliza para describir el proceso global.

Simplificando mucho, las moléculas químicas llamadas *neurotransmisores* son las encargadas de llevar mensajes de una neurona a otra consecutiva. Las vesículas en las terminales del axón de una neurona liberan moléculas de neurotransmisores. Desde un punto de vista químico, una molécula de neurotransmisor “*se ajusta como una llave en una cerradura*”, dice [Sap], cuando hace contacto con un receptor en las dendritas de una segunda neurona.

En [Res95] se explica que “*Los receptores son grandes moléculas de proteínas dinámicos que se encuentran a lo largo y dentro de la membrana celular*”. Los receptores pueden, explica el texto, “*verse aumentadas en número y avidez por su neurotransmisor según las circunstancias. Grandes y prolongadas ingestas de ciertas sustancias, por ejemplo, llevar a un aumento en el número de receptores para estas sustancias —esta es la base de la adicción a las drogas—... Más tarde, si la persona adicta se mantiene alejada de las drogas, los receptores pueden morir*”.

La unión del neurotransmisor y receptor es inhibidora o excitadora. “*Cada neurotransmisor influye en su propio receptor independiente de la acción de los receptores de otros*”, dice [Res95]. “*Si bien algunos neurotransmisores pueden disminuir la tensión entre el interior y el exterior de la célula nerviosa y así estimular la célula en acción (neurotransmisor excitador), otras lo aumentan y por lo tanto inhiben la célula de cocción (neurotransmisor inhibidor)*”.

Una vez que el neurotransmisor y receptor se unen entre sí, [Res95] describe su acción como “dinámica” y “exquisitamente sensible”. Hay dos familias de receptores:

■ Receptores de canal iónico

Provocan cambios en la permeabilidad de la membrana y, por lo tanto, regulan el flujo de iones a través de su canal. Cuando un neurotransmisor reacciona con su receptor, produce como resultado un cambio en la forma del receptor, de modo que los iones pueden entonces fluir a través de la membrana desde el punto de alta concentración hacia el punto de menor concentración.

■ Receptores sin canales iónicos

Actúan a través de intermediarios, las proteínas G, ubicadas en el interior de la célula nerviosa. Básicamente, las proteínas G funcionan como factores de acoplamiento que sirven como enlaces entre el receptor de la superficie exterior de la membrana de la célula nerviosa y un gran número de procesos celulares relacionados entre sí dentro de la célula.

Una excitación suficiente en los receptores de los canales iónicos de las espinas dendríticas provoca una onda de cambio iónico que lleva al axón de la neurona a alcanzar su máximo potencial de acción. Una señal eléctrica así desplaza hacia abajo el axón de la neurona y los terminales de los axones para provocar que las vesículas liberen un neurotransmisor químico. En otras palabras, la neurona “dispara”. Como las vesículas liberan moléculas neurotransmisoras, cada molécula queda “a la deriva”, con el potencial de hacer contacto con un receptor. Las neuronas, comunicándose entre sí en una dirección, producen un circuito. A menudo existen circuitos alternativos entre las mismas estructuras cerebrales.

Los circuitos neuronales locales en el neocórtex constituyen regiones corticales. Las regiones corticales no solo pueden conectarse unas con otras, sino también con los núcleos

subcorticales.

Neuronas, proyecciones y vías

[Dam94] explica que “*un haz de axones provenientes de fuente conocida y hacia un objetivo determinado se conoce a menudo como proyección, porque proyecta axones a una colección de neuronas. Una secuencia de proyecciones a través de varias estaciones de destino se conoce como vía*”. A menudo se puede ver una vía (materia blanca) a simple vista. Como se ha mencionado antes, [Cze01] describe la materia blanca como “*una colección de axones largos, aislados, ramas salientes de neuronas que conectan las diversas partes del cerebro entre sí*”.

3.4.2. El desarrollo cerebral

Es momento de estudiar el proceso de formación del cerebro humano.

Introducción a la embriología

En [Shu09] se ofrece una genial introducción a lo que se conoce como *capas germinales* y a la embriología:

“*En la década de 1800, algunos filósofos naturales estudiaron a los embriones para tratar de encontrar el plan común para la vida en la tierra. Uno de estos observadores fue Karl Ernst von Baer, nacido en una familia noble, que inicialmente se formó para ser médico. Su mentor académico sugirió que para tratar de entender cómo funcionan los órganos de un pollo desarrollado, debía estudiar y entender el desarrollo del pollo.*

Desafortunadamente, von Baer no podía permitirse el lujo de trabajar en incubadoras de pollos, ni podía permitirse muchos huevos. Por suerte para él, tenía un amigo rico, Christian Pander, que podía darse el lujo de hacer los experimentos. Mientras se fijaban en los embriones, se encontraron con algo fundamental: todos los órganos del pollo se puede remontar a una de las tres capas de tejido del embrión en desarrollo. Estas tres capas se conocían como las capas germinales. Lograron un estatus casi legendario, que conservan aún hoy en día.

El descubrimiento de las tres capas de von Baer dio los medios para hacer preguntas importantes: ¿Todos los animales comparten este patrón? Son el corazón, los pulmones y los músculos de los animales derivados de estas capas? Y, sobre todo, ¿las mismas capas se convierten en los mismos órganos en especies diferentes?

Von Baer comparó las tres capas de embriones de pollo con todo lo demás que pasara por sus manos: peces, reptiles y mamíferos. Efectivamente, pudo con-

cluir que todos y cada uno de los órganos animales se habían originado en una de estas tres capas. Significativamente, las tres capas se formaron con las mismas estructuras en todas las especies. Cada corazón de cada especie estaba formado a partir de la misma capa. Otra capa daba lugar al cerebro de los animales. Y así sucesivamente. No importa lo diferentes que sean las especies, tanto adultos como pequeños embriones, todos pasamos por las mismas etapas de desarrollo”.

Tubo neural

El cerebro comienza a formarse a partir de una estructura en forma de tubo que se conoce como *tubo neural*. [Lau01] afirma que el comienzo de todo el sistema nervioso central puede ser concebido como un globo inflado, alargado como el que usan los payasos para hacer globoflexia. El tubo neural se flexiona en dos puntos: la *flexión cervical* y la *flexión cefálica*.

En las primeras etapas del desarrollo humano, sólo hay tres bultos o vesículas en el tubo neural. A medida que se desarrolla el embrión, estas vesículas comienzan a diferenciarse en subdivisiones que se denominan comúnmente *prosencéfalo*, *mesencéfalo* y *rombencéfalo*. Los seres humanos comparten estas características de desarrollo en el cerebro con todos los demás vertebrados, incluidos los peces vertebrados, los anfibios, los reptiles, las aves y, por supuesto, otros mamíferos.

La figura 3.13 muestra el comienzo de la división del tubo neural. [Lau01] señala que, debido a que todo se coloca inicialmente en la línea media, los ventrículos laterales —una cavidad interna de los hemisferios cerebrales, que serán estudiados en la sección 3.4.4— tienen que ser divisiones laterales en los que el tubo neural se vuelve hacia el exterior. En la figura 3.13 se puede apreciar cómo esas divisiones del prosencéfalo comienzan a tomar forma para producir el *telencéfalo* (que después evoluciona y forma los hemisferios cerebrales) y el *diencéfalo* (que pasa a formar el *tálamo* y el *hipotálamo*).

En [Lau01] se explica que el telencéfalo es “el ventrículo más evolucionado”, una especie de “cerebro bien formado”. El prefijo *tele*, del griego antiguo, significa “a distancia”. Por otro lado, el prefijo *di* de *diencéfalo*, según [Lau01], se explica porque el diencéfalo es una especie de “cerebro de en medio” o “inter-cerebro”.

Si bien el *mesencéfalo* permanece relativamente indiferenciado, el *rombencéfalo* pasa a formar dos vesículas: la *metencéfalo* (que después pasa a formar el *puente troncoencefálico* (o protuberancia anular) y el *cerebelo*) y el *mielencéfalo* (que después se convierte en el *bulbo raquídeo*).

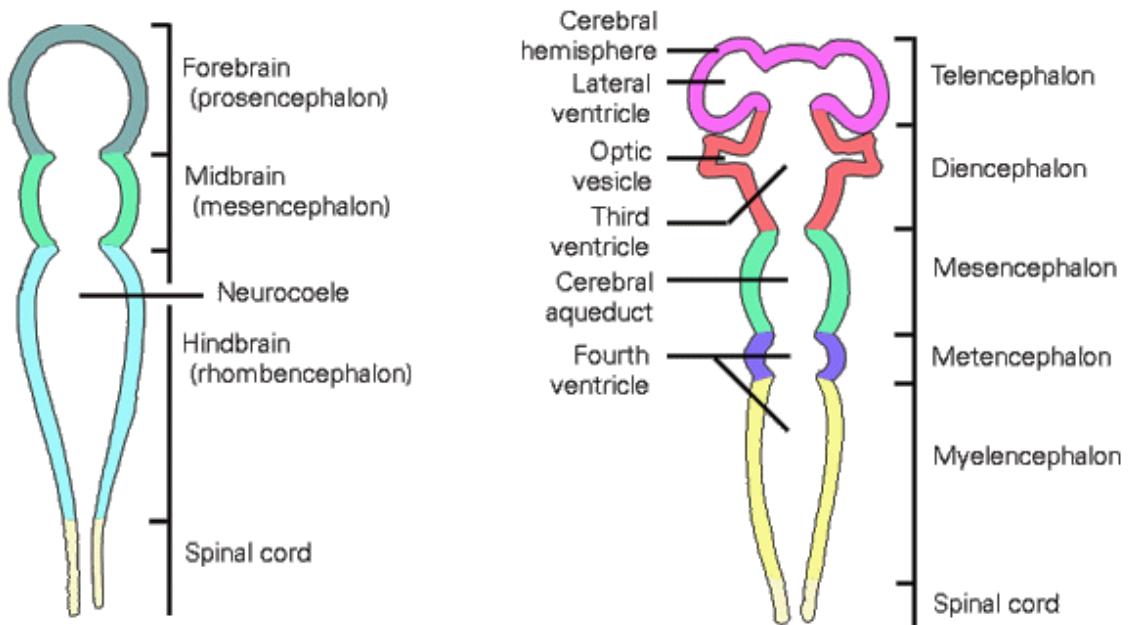


Figura 3.13: Tubo neural. A la izquierda el aspecto del tubo en la etapa más temprana de desarrollo, donde pueden apreciarse las 3 partes principales: prosencéfalo, mesencéfalo y rombencéfalo. A la derecha, en una etapa posterior, el prosencéfalo ha pasado a formar el telencéfalo y el diencéfalo.

Integración de neurocircuitos

A medida que el cerebro se desarrolla, los circuitos neuronales son creados para integrar el cerebro en un todo funcional. [Dam94] deja esto claro: “*Hay miles de millones de neuronas en los circuitos de un cerebro humano, el número de sinapsis que se forman entre ellas es al menos de 10 billones, y la longitud de los axones de las neuronas que forman el cableado de los circuitos asciende al orden de cientos de miles de millas. La escala de tiempo para la transferencia de información es muy pequeña, del orden de decenas de milisegundos, lo que significa que en un segundo en la vida de la mente, el cerebro produce millones de patrones de activación sobre una gran variedad de circuitos distribuidos en varias regiones del cerebro*”. Más tarde, el autor añade: “*Los secretos elementales de la mente residen, local y globalmente, en el momento de la interacción neuronal al disparar patrones generados por los circuitos neuronales del cerebro de un organismo vivo*”.

Los desequilibrios en la neurotransmisión de los circuitos cerebrales causan síntomas tales como obsesiones, compulsiones, tics nerviosos y déficit de atención. Algunos factores que influyen en estos desequilibrios son las infecciones virales o bacterianas, incessantes formas de estrés, lesiones físicas, la vulnerabilidad genética, o una combinación de factores. [Dam94] afirma que “*La distinción entre enfermedades del cerebro (mente), problemas neurológicos, y problemas psicológicos o psiquiátricos es una desafortunada herencia cultural que permanece en la sociedad, y la medicina refleja una ignorancia básica de la relación*

entre el cerebro y la mente”.

3.4.3. Estructuras primarias del cerebro

“Debajo de la corteza cerebral, el cerebro humano tiene un parecido sorprendente con los de otras especies mucho más antiguas”, explica [Cze01]. Así como lo hacen en la mayoría de los animales, las neuronas del bulbo raquídeo y el puente troncoencefálico situado en la base del cerebro, donde nace la médula espinal, de manera constante y fiable regulan sus funciones vegetativas, automatizando su actividad corporal como los latidos del corazón y la respiración.

El tronco encefálico. La formación reticular

El denominado *tronco encefálico* (o *tallo cerebral*) incluye el *mesencéfalo* o cerebro medio, el *puente troncoencefálico* y el *bulbo raquídeo*. El núcleo del tallo cerebral se llama *formación reticular*. La palabra “reticular” significa “red como”, por lo que dicho núcleo describe el aspecto estructural del tejido del tallo cerebral. Por otro lado, el *sistema de activación reticular*, como puede leerse en [Lin], es “*una parte de la formación reticular que se extiende desde el tronco cerebral hasta el mesencéfalo y el tálamo, con conexiones distribuidas por toda la corteza cerebral y que controla el grado de actividad del sistema nervioso central —por ejemplo, mantiene el sueño y la vigilia, así como la transición entre los dos estados*”. Existen diversos neurotransmisores implicados en la función reticular del sistema de activación, como la *acetilcolina* y la *norepinefrina*.

Las neuronas encargadas de la segregación de *serotonina*, llamadas *núcleos del rafé*, forman una cresta o costura en el centro de la formación reticular. [Pan98] explica que estas neuronas están “*situadas en la línea media del cerebro, lo que indica que son muy antiguas en la evolución del órgano*”.

En [Joh00] se señala que “*la red de neuronas serotoninérgicas en el tronco cerebral estaba presente en los primeros vertebrados y ha mantenido una posición anatómica notablemente constante durante la evolución de vertebrados. Las neuronas serotoninérgicas se llaman así porque segregan, desde sus terminales de axón, el neurotransmisor llamado serotonina*”. Más adelante continúa: “*los cuerpos celulares de las neuronas serotoninérgicas ocupan prácticamente la misma ubicación en el interior de cada cerebro de los vertebrados y se encuentran incluso en el mismo lugar en el sistema nervioso central. Este sistema ha sido increíblemente conservado durante la evolución, sin embargo, participa vitalmente en los aspectos más complejos de nuestro pensamiento y de las emociones*”.

Generación de patrones

Aunque rara vez nos detengamos a pensar en ello, nuestro cerebro constantemente genera patrones de actividad, aparentemente sin esfuerzo. Muchos de estos patrones puede conti-

nuar incluso después de que “se apague” la conciencia. El *tallo cerebral* es el ejemplo más obvio de un generador de patrones. En [Shu09] se explica lo siguiente: “*Nuestro cerebro puede controlar nuestra respiración sin ningún esfuerzo consciente por nuestra parte. La mayor parte de ese trabajo se lleva a cabo en el tronco cerebral, en el límite entre el cerebro y la médula espinal. El cerebro envía impulsos nerviosos a los músculos respiratorios principales. La respiración sigue un patrón en el que están implicados los músculos del pecho, el diafragma y la garganta, trabajando en un orden bien definido. Por consiguiente, esta parte del tallo cerebral se considera un generador central de patrones. Esta región puede producir patrones rítmicos nerviosos y, en consecuencia, la activación de los músculos correspondientes. Un número de generadores de este tipo en el cerebro y la médula espinal se encargan del control de otros comportamientos rítmicos, como la ingesta y el caminar*”.

El sistema nervioso autónomo (SNA)

Según [Sap04] “la manera principal en que el cerebro puede decirle al resto del cuerpo lo que debe hacer es el envío de mensajes a través de los nervios que se ramifican desde el cerebro hasta la columna vertebral y hacia la periferia de su cuerpo”. Conocemos mucho el sistema nervioso voluntario, donde “*tú decides mover un músculo y lo mueves*”, dice el autor. Él explica que existe otra parte del sistema nervioso, el *sistema nervioso autónomo* (SNA), que controla los eventos más espontáneos e involuntarios, tales como el rubor, la *piel de gallina*, y el orgasmo.

[Lin] define el SNA como “*una parte del sistema nervioso de los vertebrados que regula las acciones involuntarias*”. Por otro lado, según [Mer] el sistema nervioso autónomo “*funciona automáticamente (de manera autónoma), sin esfuerzo consciente de una persona*” y “*controla los órganos internos, incluyendo los vasos sanguíneos, el estómago, el intestino, el hígado, los riñones, la vejiga, los genitales, los pulmones, los músculos de los ojos, el corazón y el sudor, la saliva y las glándulas digestivas*”.

El SNA se compone de dos subsistemas: el *sistema nervioso parasimpático* y el *sistema nervioso simpático*. La función del sistema nervioso parasimpático es promover la calma, con funciones tales como el descanso y la digestión, mientras que la función del sistema nervioso simpático es la de preparar al individuo para luchar o huir. En la figura 3.14 se muestra una panorámica fácil de recordar de esta división funcional.

El cerebelo

[Lin] destaca que el *cerebelo* (del latín “pequeño cerebro”) está “*situado entre el tronco cerebral y la parte posterior del cerebro en los seres humanos, y formado por dos lóbulos laterales y un lóbulo medio*”. En la figura 3.15 puede apreciarse la localización del mismo.

La función principal del cerebelo es la de integrar las vías sensitivas y las vías motoras. Existe una gran cantidad de haces nerviosos que conectan el cerebelo con otras estructuras

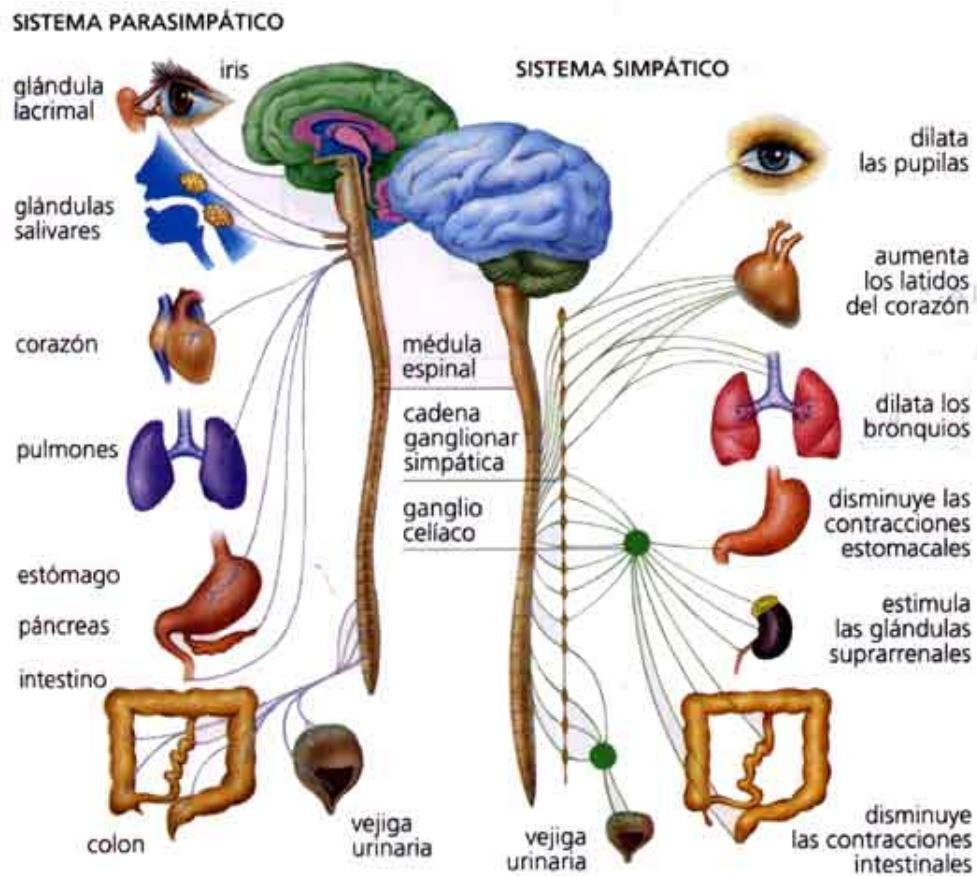


Figura 3.14: Subsistemas simpático y parasimpático del SNA

encefálicas y con la médula espinal. El cerebelo integra toda la información recibida para precisar y controlar las órdenes que la corteza cerebral manda al aparato locomotor a través de las vías motoras. Por ello, lesiones a nivel del cerebelo no suelen causar parálisis pero sí desordenes relacionados con la ejecución de movimientos precisos, mantenimiento del equilibrio, la postura y aprendizaje motor.

Como último detalle, de acuerdo con [Mer] “*hay un creciente consenso en que, además de la coordinación, el cerebelo controla algunos aspectos de la memoria, el aprendizaje y la cognición*”. El texto explica que, aunque la *ataxia* — una enfermedad que se caracteriza por provocar la descoordinación en el movimiento de las partes del cuerpo— es el síntoma más típico de disfunción cerebelosa, muchas otras anomalías motoras también pueden ser fruto de esta disfunción.

3.4.4. Los hemisferios cerebrales

El cerebro se divide en dos mitades prácticamente simétricas denominadas *hemisferios cerebrales* (figura 3.16). En [Res94] se analiza cómo los hemisferios izquierdo y derecho responden ante diferentes situaciones, para determinar la funcionalidad de cada uno:

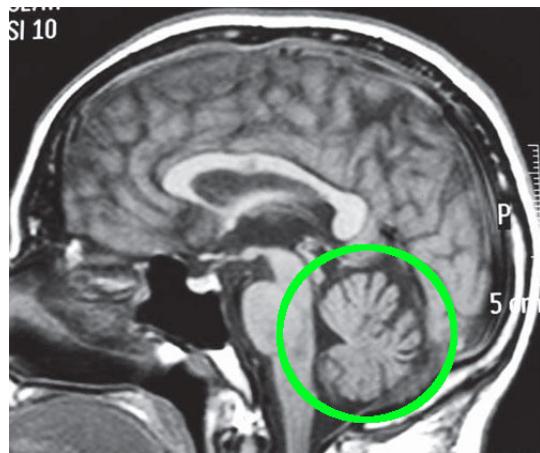


Figura 3.15: Localización del cerebelo

“El hemisferio derecho es dominante para cuestiones no verbales, tareas espaciales, interpretación de gestos faciales y emociones, visualización y transformación mental, y para la apreciación e intuición de figuras geométricas. Por otro lado, el hemisferio izquierdo se centra en el habla, la escritura y la comprensión oral, así como el cálculo.

Pero aún más importante que la división funcional es el hecho de que cada hemisferio tiene su propia conciencia.”

Ambos hemisferios funcionan de forma diferente al procesar emociones. Mientras que el hemisferio derecho responde ante emociones negativas, el izquierdo lo hace ante emociones positivas. Cuando el hemisferio derecho sufre daños, en [Pan98] se explica que “*con frecuencia los pacientes permanecen alegres, pese a la gravedad de sus problemas*”. Por otro lado, un daño similar en el hemisferio izquierdo “*puede causar estrés emocional catastrófico, predisponiendo a los pacientes a sufrir depresión*”.

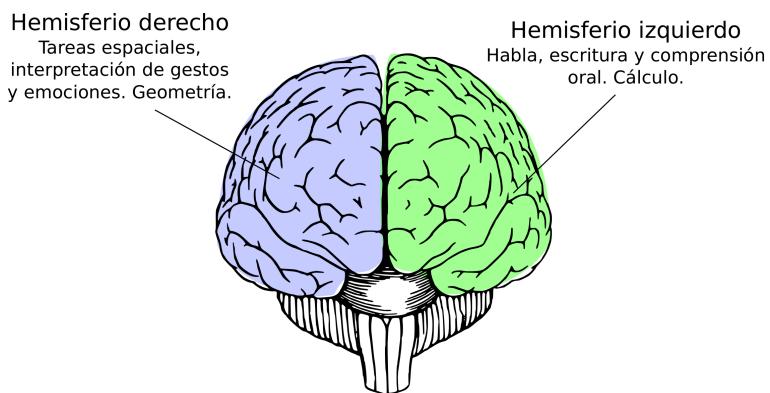


Figura 3.16: Hemisferios cerebrales (vista frontal)

3.4.5. La corteza cerebral

[Joh00] define “córtez” como “*la cubierta externa o corteza de un objeto*”.

El neocórtex

Como se mencionó en la sección 3.4.2 cuando se describía el desarrollo del tubo neural, el *prosencéfalo* (cerebro anterior), el *mesencéfalo* (cerebro medio), y el *rombencéfalo* (cerebro posterior) son las partes del cerebro humano durante el comienzo del desarrollo del sistema nervioso central. Durante el desarrollo embrionario el prosencéfalo se divide en *diencéfalo* (tálamo e hipotálamo), y *telencéfalo* (hemisferios cerebrales).

“*El neocórtex*⁵ es una especialización en el telencéfalo que es paralelo a la formación de la cresta dorsal ventricular en reptiles y aves. El neocórtex es tanto una característica única de mamíferos como son las glándulas mamarias o el martillo y el yunque en el oído medio”, según [Joh00], y “*almacena información acerca de la estructura del medio ambiente a fin de que el mamífero puede fácilmente encontrar alimentos y otros recursos necesarios para su supervivencia*”.

Por otro lado, [Sap] se refiere al *neocórtex* como una “*especialización primate*”, terminología que corresponde con la descripción anterior.

La corteza del cíngulo anterior: emoción, atención y memoria a corto plazo

Antes de discutir las diferenciaciones anatómicas denominadas lóbulos conviene hablar de la corteza del cíngulo anterior, una zona importante de la corteza cerebral que tradicionalmente se ha considerado parte del sistema límbico⁶. Desde el punto de vista evolutivo, cabe destacar que la corteza del cíngulo anterior es más antigua que las zonas exteriores de la corteza.

La corteza del cíngulo anterior se amolda a la curva exterior del cuerpo calloso, la gruesa banda de fibras nerviosas que conecta los hemisferios izquierdo y derecho del cerebro (ver hemisferios en figura 3.16). El término “cíngulo” proviene del latín, y significa cinturón o faja. Este área también es denominada *circunvolución cingulada*. [Lin] define “anterior” como “*en relación con o que se encuentra cerca o hacia la cabeza...*”. En la figura 3.17 la corteza del cíngulo (tanto anterior como posterior) se muestra resaltada, mientras que la parte anterior se encuentra, además, rodeada.

En [Dam94] el autor escribe: “*Me gustaría proponer que existe una región particular del cerebro humano donde los sistemas afectados por la emoción / sentimiento, la atención, y la memoria a corto plazo interactúan tan íntimamente que constituyen la fuente de la energía de*

⁵También conocido como *neocorteza* en la literatura médica española.

⁶Sistema formado por varias estructuras cerebrales que gestionan respuestas fisiológicas ante estímulos emocionales. Está relacionado con la memoria, la atención, los instintos sexuales, las emociones —placer, miedo, agresividad—, la personalidad y la conducta.

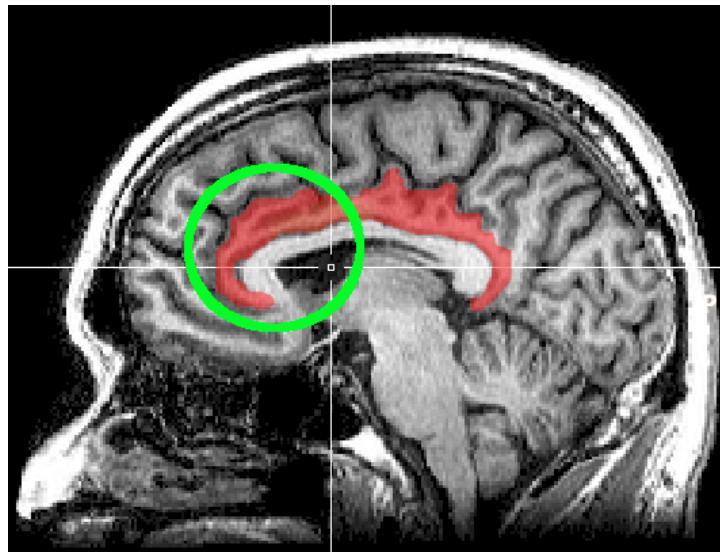


Figura 3.17: Corteza del cíngulo anterior

la acción exterior (movimiento) y la acción interna (pensamiento animación, razonamiento). Esta región manantial es la corteza cingulada anterior, otra pieza del rompecabezas que constituye el sistema límbico”. Después continúa diciendo: “*Mi idea acerca de esta región proviene de la observación de un grupo de pacientes con daño en y alrededor de ella. La enfermedad se describe mejor como animación suspendida, mental y externa —la variedad extrema de un deterioro del razonamiento y de expresión emocional—.*”.

Lo anterior llama la atención sobre un fenómeno interesante que ilustra la función de la corteza cingulada anterior. “*Cuando una apoplejía destruye la corteza motora en el hemisferio izquierdo del cerebro y, como resultado, el paciente tiene parálisis en el lado derecho de la cara*”, comenta [Dam94], “*los músculos no pueden actuar y la boca tiende a ser arrastrada hacia un lado al pedirle al paciente que la abra — los dientes revelan un aumento de la asimetría—. Sin embargo, cuando el paciente sonríe o se ríe espontáneamente, en respuesta a una observación humorística, sucede algo completamente diferente: la sonrisa es normal, los dos lados de la cara se mueven de forma correcta, y la expresión es natural, nada diferente de la sonrisa habitual pre-parálisis de esa persona. Esto pone de manifiesto que el control motor para una secuencia de movimientos relacionada con las emociones no se encuentra en la misma ubicación que el control de un acto voluntario. El movimiento relacionado con las emociones se desencadena en otras zonas del cerebro, incluso si los músculos objetivos del movimiento son los mismos.*”

Para aclarar que la expresión emocional tiene su origen, [Dam94] escribe: “*Si usted estudia a un paciente en el que un golpe ha dañado la corteza cingulada anterior en el hemisferio izquierdo, verá exactamente el resultado opuesto en reposo o en movimiento relacionado con las emociones, la cara es asimétrica, menos móvil en el lado derecho que en el izquierdo. Pero si el paciente trata de contraer los músculos de la cara deliberadamente, los movimien-*

tos se llevan a cabo normalmente y vuelve la simetría. La emoción está relacionada con el movimiento, entonces, se controla desde la región cingulada anterior.”

La corteza cingulada anterior puede desempeñar un papel importante en los lazos entre padres e hijos.

Los lóbulos cerebrales

La diferenciación cerebral en “lóbulos” se utiliza para indicar la posición relativa de las estructuras cerebrales. Por ejemplo, la amígdala se encuentra dentro de los lóbulos temporales. Es importante recordar que un lóbulo no funciona independientemente, no son unidades autónomas, sino que son meramente una referencia anatómica. [Dam94] señala que “en un sólo segundo el cerebro produce millones de patrones de activación a través de una gran variedad de circuitos distribuidos en varias regiones del cerebro”.

Si nos centramos en cualquiera de los hemisferios cerebrales por separado, encontramos una nueva división de funcionalidad: los lóbulos cerebrales (figura 3.18):

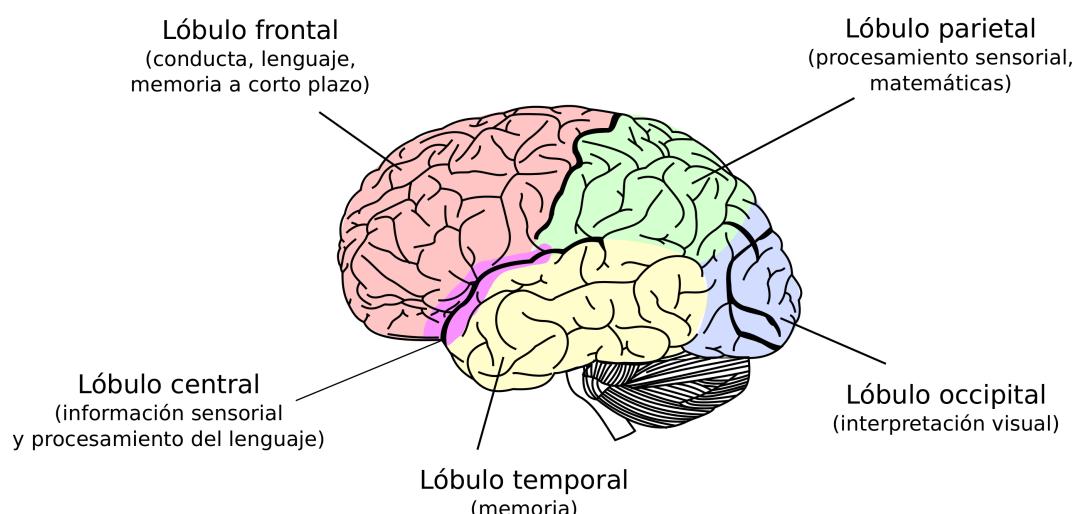


Figura 3.18: Lóbulos cerebrales

■ Lóbulo frontal

Está involucrado en la planificación y priorización de tareas. Según [Res95]: “El lóbulo frontal constituye el 50 % de cada hemisferio cerebral de los seres humanos. Es el encargado de iniciar cualquier actividad motriz, como el habla. Se subdivide en el lóbulo prefrontal —que integra personalidad con emociones— y la corteza motora —que transforma los pensamientos en acciones—.

■ Lóbulo temporal

Contiene un área sensorial relacionada con la audición. Ubicado dentro de cada lóbulo temporal se encuentran la amígdala y el hipocampo, que están, tal como se describe en

[Res95], “*implicados en el aprendizaje, la memoria, la experiencia y la expresión de la emoción*”. Una porción del lóbulo temporal, llamada la *corteza entorrial*, conecta las señales corticales de entrada con el hipocampo. “*Las fibras de los cuatro lóbulos, junto con fibras de asociación que unen éstas entre sí, convergen en la región del hipocampo*”, dice [Res95]. En [Joh00] se señala que la amígdala —tan importante para el procesamiento emocional— también recibe la entrada de un área cortical en el lóbulo temporal llamada *corteza visual inferotemporal*, que es “*más amplia en gran medida en los primates superiores*”. [Joh00] comenta, además, que “*Charles Goss, Robert Desimone, Rolls Edmund y David Perrett entre otros, han demostrado que las neuronas de la corteza inferotemporal son especialmente sensibles a las imágenes de los rostros*”.

■ Lóbulo occipital

Incluye sofisticados mapas topográficos con complejas interconexiones, necesarias para el procesamiento visual y, acorde con [Lin], tiene “*la forma de una pirámide de 3 lados*”. [Mer] explica que, además de procesar e interpretar la visión, las áreas corticales en el lóbulo occipital permiten al ser humano mantener memorias visuales e integrar percepciones visuales con la información espacial procedente de los lóbulos parietales adyacentes. Si una lesión en la cabeza daña los lóbulos occipitales, en ocasiones se produce lo que se conoce como *agnosia visual* —definida por [Mer] como la “*pérdida de la capacidad de asociar objetos con su habitual papel o función*”—. Con suficiente daño en el lóbulo occipital, “*la gente no puede reconocer caras familiares u objetos comunes, como una cuchara o un lápiz, a pesar de que puede ver estas cosas*”, según se explica en [Mer].

■ Lóbulo parietal

Contiene un área que procesa las sensaciones corporales, llamada *corteza somatosensorial*. “*El lóbulo parietal es la estación receptora de la información sensorial desde el lado opuesto del cuerpo y es responsable de la integración de lo que se ve con lo que se siente a través de una red de fibras de asociación*”, explica [Res95].

■ Lóbulo central o ínsula

Se encuentra situado “*en el centro del hemisferio cerebral, enterrado entre la unión de los lóbulos frontal y temporal*”, acorde con [Lin]. Según [Mer], además, la ínsula integra el procesamiento de la información sensorial y autonómica. Involucrado en ciertas funciones del lenguaje, su deterioro puede derivar en la *afasia*, enfermedad que supone la pérdida de capacidad de producir o comprender el lenguaje, tanto hablado como escrito.

La corteza frontal

La figura 3.19 muestra una neurona piramidal, que es común en la corteza prefrontal. La corteza frontal ayuda a centrarse en lo que una tarea es ahora. Los pacientes que sufren demencia en las primeras etapas son incapaces de mantenerse en una tarea concreta, y tienden a volver a tareas anteriores. Sapolsky demostró esta idea con un sencillo experimento consistente en solicitar a pacientes con esta enfermedad que dijeran los meses del año, y después pedirles que contaran hacia atrás a partir 20. El resultado fue que durante la cuenta atrás, los pacientes pasaban a enumerar meses del año, mostrando una imposibilidad para mantener la tarea en la que se encontraban. Esta disfunción se conoce como *perseverancia e intrusión* (reversión a una tarea anterior).

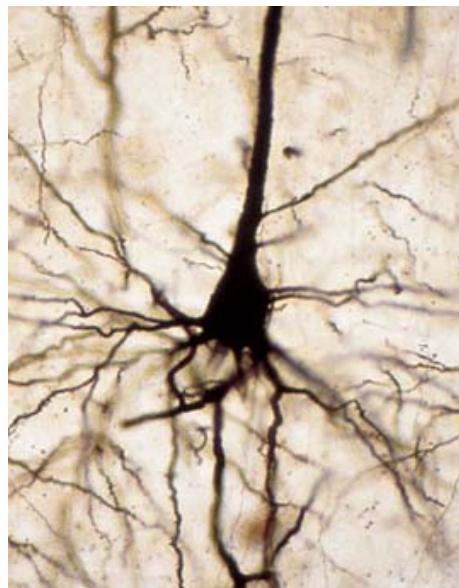


Figura 3.19: Neurona piramidal

“*La corteza frontal está involucrada en el control ejecutivo, sensación de gratificación y planificación a largo plazo*”, explica [Sap04]. “*Lo hace enviando proyecciones inhibitorias en el sistema límbico, sistema cerebral involucrado en la emoción y la impulsividad. Además, la corteza frontal resiste entradas estimulantes del sistema límbico, ignorando tentadores susurros límbicas como «No estudie para el examen, salga a dar una vuelta»*”. En el mismo texto se añade que cuando la corteza frontal es destruida, los pacientes sufren desinhibición sexual, agresividad, etc.

Como apunte curioso, en [JC05] se explica cómo si su cerebro se ve dañado en un accidente o derrame cerebral, puede parecer que tiene dañados los lóbulos frontales, incluso cuando los lóbulos frontales permanecen intactos. “*La gente siempre piensa en los lóbulos frontales debido a que son la última estructura en evolucionar, y por lo tanto la más delicada, mientras que las estructuras más antiguas son increíblemente robustas*”. Pero el neuropsicólogo Elkhonon Goldberg, sigue explicando el texto, presentó una teoría diferente: “*Él piensa que,*

si bien los lóbulos frontales pueden ser más frágiles, hay otro factor en juego, y es que todas las otras partes del cerebro están conectadas a ellos. Al dañar cualquier parte del cerebro, la entrada a los lóbulos frontales se ve alterada, y un cambio en la entrada produce un cambio en la salida. Si los lóbulos frontales no están recibiendo la entrada correcta, es lógico que no produzcan el resultado correcto, aunque estructuralmente estén en perfecto estado. Así que todo el daño cerebral termina pareciéndose a daño en el lóbulo frontal.”.

“En los primates, el lóbulo frontal tiene un papel importante en el establecimiento de prioridades y la planificación”, explica [Joh00], “En particular, la superficie inferior del lóbulo frontal, la denominada corteza orbital-frontal, es especialmente importante para estas funciones”. La ubicación de la corteza orbital-frontal puede apreciarse en la figura 3.20. Cabe destacar que en la literatura médica también se suele utilizar el término *región ventromedial del lóbulo frontal* para referirse a la región orbital-frontal.

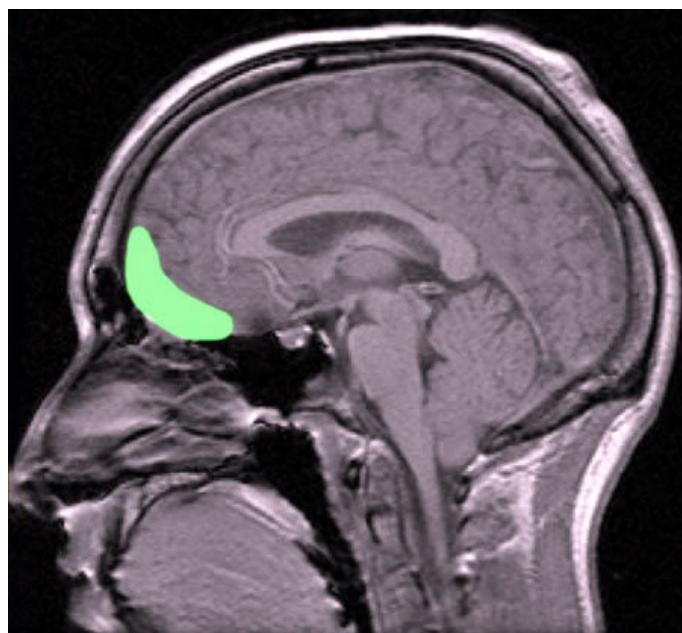


Figura 3.20: Corteza orbital-frontal

Las áreas de Brodmann

Según se afirma en [II] “La arquitectura celular difiere suficientemente de una parte del neocórtex a otra como para ser utilizada como un criterio para definir las áreas corticales funcionalmente diferentes”. Con esa observación, el anatomista alemán Korbinian Brodmann elaboró un mapa de el cerebro basado en las diferencias en la arquitectura celular de las diversas partes de la corteza, asignando a cada parte de la corteza un número del 1 al 52 (ver figura 3.21). Respecto a esta división, [II] dice: “La intuición de Brodmann, cuya exactitud ha sido confirmada en muchas ocasiones, afirma que una estructura anatómica particular corresponde a una función en particular. Por ejemplo, el área de Brodmann 17

(que recibe información de un núcleo del tálamo que se conecta a la retina) resulta que corresponde exactamente con la corteza visual primaria.”.

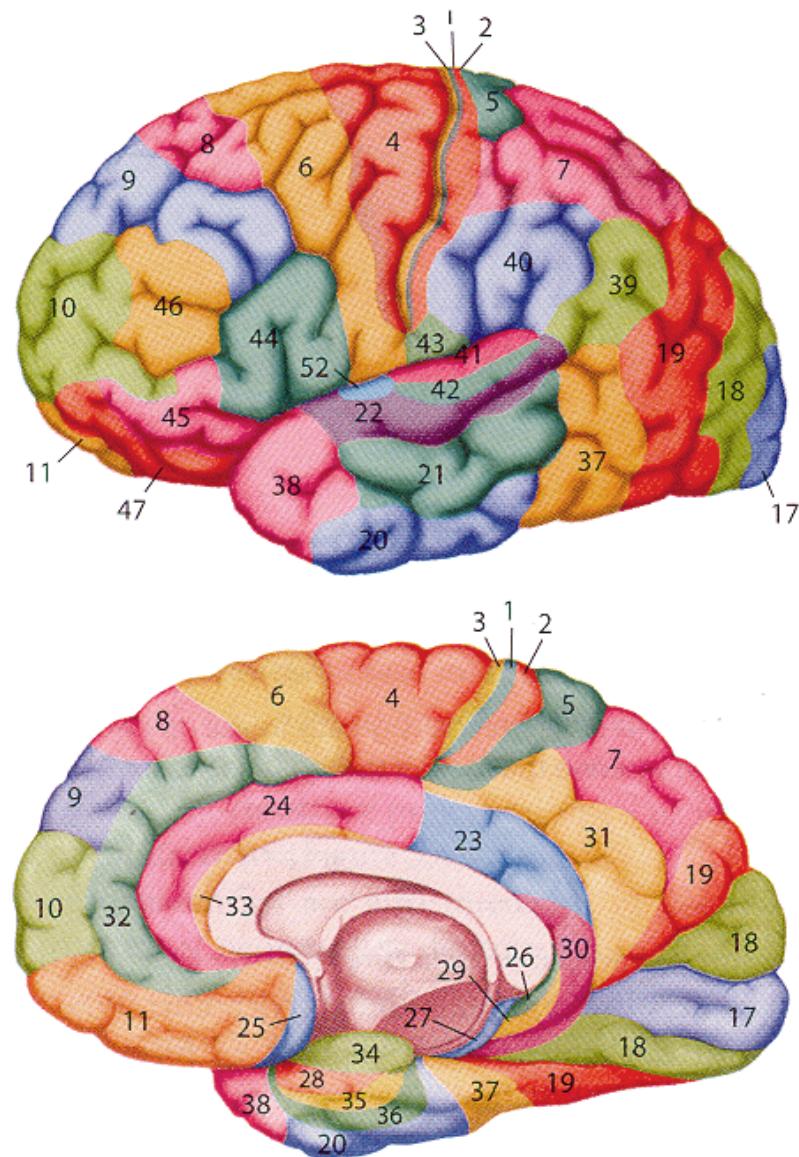


Figura 3.21: Áreas de Brodmann

En la figura 3.21, la corteza motora primaria se corresponde con el área 4. El neurocirujano Wilder Penfield Graves (1891-1976) exploró esta región mientras trataba a pacientes con epilepsia severa en Montreal. El objetivo de Penfield era destruir las células nerviosas responsables de la crisis. Mientras el paciente se encontraba bajo el efecto de anestesia local, Penfield estimuló el cerebro con sondas eléctricas, observando las respuestas de los pacientes. Lo hizo para identificar las áreas que requieren cirugía y evitar las áreas vitales que no deben ser destruidas. Al hacer esto, se observó que la estimulación de ciertas áreas de la corteza cerebral desencadenaba contracciones musculares muy localizadas en el lado opuesto del

cuerpo. La conclusión fue que las áreas de la corteza asignadas a las diferentes partes del cuerpo “*no son proporcionales a su tamaño, sino más bien a la complejidad de los movimientos que puede realizar. Por lo tanto, las áreas de la mano y cara son especialmente grandes en comparación con las del resto del cuerpo, lo cual no es sorprendente, ya que la velocidad y la destreza de la mano humana y los movimientos de la boca son precisamente los que nos dan dos de nuestras facultades más distintivamente humanas: la capacidad de utilizar las herramientas y la capacidad de hablar.*”.

En [Joh00] se cita el trabajo de Hughlings Jackson en la determinación de cómo el cerebro controla el movimiento muscular. En la observación de que los pacientes epilépticos en ocasiones sufren crisis convulsivas centradas en un lugar particular en el cuerpo, Jackson “*llegó a la conclusión de que los músculos se representan en el cerebro en un lugar determinado, que dedujo encontrarse en algún lugar de la corteza cerebral o en una estructura cercana llamada cuerpo estriado. Esta teoría fue un cambio radical de la opinión clínica predominante de la época, que era que las crisis epilépticas eran causadas por una alteración en el nivel más bajo del tronco cerebral*”. En el mismo texto puede leerse: “*En 1870, la predicción topográfica de Hughlings Jackson fue confirmada por los médicos alemanes Fritsch y Gustav Eduard Hitzig, quienes descubrieron la corteza motora mediante la estimulación de la superficie del cerebro en perros con débiles corrientes eléctricas y la observación de los movimientos discretos del cuerpo*”.

[Joh00] explica que las observaciones de Jackson “*se refieren a tres propiedades fundamentales del neocortez. La primera es que contiene mapas topográficos, la segunda es que las partes más usadas de esos mapas tienen una representación mayor, y la tercera que el neocortez tiene un papel clave en la génesis de la epilepsia*”. Después continúa diciendo: “*El circuito cortical es muy plástico, ya que puede cambiar su organización funcional en respuesta a la experiencia, y es crucial para la formación de la memoria*”. Esta cualidad plástica del cerebro es la que la literatura médica denomina *neuroplasticidad*, concepto que se verá más adelante.

En la figura 3.21, las áreas 1, 2 y 3 representan la *corteza somatosensorial*, otra especie de mapa topográfico. Si un amigo le hace cosquillas, hay unas zonas muy concretas de la corteza cerebral que se ven activadas. El mapeo somatosensorial permite conocer, monitorizando la actividad cerebral, qué lugar de su cuerpo recibió las cosquillas. En [Joh00] se explica cómo se descubrieron esos mapas topográficos: “*Con el desarrollo de los amplificadores electrónicos y los osciloscopios de la década de 1930 fue posible registrar la actividad eléctrica de la corteza cerebral. Edgar Douglas Adrian, Woolsey Clinton, entre otros, descubrieron que la región adyacente a la corteza motora se activa eléctricamente por la estimulación mecánica de la superficie del cuerpo; a esa región la llamaron corteza somatosensorial, del griego soma, que significa cuerpo. Al registrar actividad en un sitio particular de la corteza somatosensorial, fueron capaces de trazar un campo receptivo en la superficie del cuerpo*

que activó ese sitio. Con un proceso sistemático encontraron una correspondencia punto a punto en la superficie cortical, por lo que fueron capaces de determinar la representación de la superficie del cuerpo en la corteza somatosensorial”.

[Joh00] desarrolla la idea de cómo mediante el uso de grabaciones de microelectrodos, Michael Merzenich, Jon Kaas, y sus colaboradores fueron capaces de confirmar que hay por lo menos cuatro mapas de la superficie del cuerpo en la corteza somatosensorial de los monos. Al igual que los mapas de la corteza motora, los de la corteza somatosensorial en los primates muestran un fuerte énfasis en la mano y la cara, lo que indica la existencia de mayor sensibilidad en las zonas de la mano, los labios y la lengua, que están conectadas a áreas mucho más grandes de la corteza que las zonas menos sensibles del cuerpo. Al igual que con la corteza motora, los mapas corticales somatosensoriales son plásticos y la representación cortical está más expandida para las partes del cuerpo que son muy usadas. La distinción entre la corteza somatosensorial y la corteza motora no es absoluta. La corteza motora tiene algunas funciones sensoriales y viceversa.

Con respecto al papel de la experiencia en la determinación de las representaciones corticales, [Joh00] escribe:

“Experimentos funcionales realizados en sujetos humanos han demostrado que la representación cerebral de la mano tiene mayor expansión, como resultado de la realización de movimientos complejos de los dedos. La expansión de la representación de la mano puede ser observada tras un corto plazo de capacitación, pero es más notable en los lectores de Braille y en los músicos que tocan instrumentos de cuerda. Estos resultados demuestran el papel de la experiencia: cuanto más fino es el grado de control y el uso de un músculo, mayor será la expansión de su representación en la corteza cerebral”.

De nuevo estamos aproximándonos al concepto de *neuroplasticidad*, que será desarrollado en la sección 3.5.

Mapas retinotópicos

En las áreas del cerebro dedicadas a la vista, los mapas cerebrales son denominados *retinotópicos*, puesto que reflejan la topografía de la retina —la capa de neuronas activadas por la luz que recubren la parte posterior del ojo—. La circuitería visual en la corteza cerebral humana contiene varias docenas de mapas retinotópicos diferentes, cada uno dedicado a analizar el flujo de entrada visual de una forma concreta. La corteza visual primaria —área 17 de Brodmann (figura 3.21)— es el principal receptor de impulsos provenientes de la parte visual del tálamo. Cabe destacar que las áreas visuales además extraen características tales como el color, el movimiento y la forma.

Gran parte del procesamiento visual del cerebro tiene lugar en el lóbulo occipital. Atendiendo a la figura 3.21, las áreas dedicadas a esta tarea son la 17, 18, 19, 20 y 21. El área 17 de Brodmann es la corteza visual primaria (V1). El área 18 representa la corteza visual secundaria (V2) y el área 19 la corteza visual asociativa (V3). Cada una de estas áreas constituye un mapa retinotópico. Como se mencionó en el apartado 3.18, las áreas de Brodmann 20 y 21 forman parte del lóbulo temporal, y representan la corteza visual inferotemporal, que incluye neuronas sensibles a las imágenes de rostros y envía información visual a la amígdala.

En [Joh00] se detalla cómo la corteza visual primaria (V1) se asignó inicialmente:

“En la guerra entre Rusia y Japón de 1905, muchos soldados japoneses sufrieron heridas de bala en la parte posterior de su cerebro. Debido a la velocidad de salida más alta y el tamaño más pequeño de las balas desarrolladas en el siglo XIX, estas armas tendían a producir lesiones cerebrales más localizadas que las que fueron infringidas en guerras anteriores, lo que mejoró la atención de los heridos y dio lugar a mayores tasas de supervivencia. Muchos de los soldados heridos fueron cegados en parte por estas lesiones, por lo que el oftalmólogo Tatsuji Inouye trabajó para el gobierno japonés para evaluar el alcance de su ceguera como un medio para determinar sus beneficios de pensión. Inouye descubrió que la parte del campo visual en el que estos soldados habían quedado ciegos correspondía a la localización de las lesiones cerebrales producidas por los orificios de entrada y salida de la bala. Mediante la combinación del déficit visual de diferentes soldados pudo deducir la organización topográfica de la corteza visual primaria. Inouye reveló que había mucha más corteza dedicada a la representación de la parte central de la retina que a la periferia. Esta es la parte de la retina con mayor agudeza, y es nuestro medio más importante para obtener información del entorno —y la parte que el lector está empleando en este instante al leer este libro—. El mapa de la corteza visual primaria de Inouye ha sido confirmado por las técnicas modernas de extracción de imágenes cerebrales.”

En [Res94] se proporciona un ejemplo de cómo el daño a un área del cerebro, en sólo un hemisferio, puede cambiar dramáticamente la forma en que percibimos el mundo. El autor discute cómo Michael Gazzaniga llevó a cabo un experimento que involucró a un sujeto que había perdido la parte izquierda del campo visual a causa de daño cerebral en el área visual del lado derecho de su cerebro: “Se les pidió que se imaginara a sí mismo mirando hacia California desde Nueva York, y que nombrara los estados que encontraba en medio. Nombró sólo diez estados, todos ellos situados a la derecha de su punto de vista imaginario. Omitió a los estados a la izquierda, que corresponden al campo visual dañado por su lesión cerebral”

derecha. Lo que no podía ver en el mundo real como resultado de su lesión cerebral, tampoco era capaz de verlo en la imagen espacial de su imaginación”.

Con el fin de diferenciar la complejidad de las áreas de la corteza visual, en [Joh00] se citan las diferencias en la corteza visual que distinguen a los primeros mamíferos de los primates y los seres humanos: “*Las zarigüeyas y los erizos, que en muchos aspectos se parecen a los primeros mamíferos que vivieron hace más de 60 millones de años, han limitado su capacidad visual a un pequeño número de las áreas corticales visuales. En los mamíferos, los mapas corticales de la retina son relativamente uniformes en cuanto a la cantidad de espacio cortical dedicado a la parte más central del campo visual, en general en estos animales no es mucho mayor que la corteza dedicada a las partes más periféricas del campo visual. Por el contrario, los primates tienen extremadamente bien desarrolladas las capacidades visuales y tienen un gran número de mapas corticales dedicados a la percepción visual y la memoria. Dentro de la mayoría de estos mapas hay un fuerte énfasis de la representación de la parte central del campo visual y una representación mucho más pequeña de las partes periféricas de la campo visual*”.

3.4.6. Estructuras subcorticales. El estrés, las emociones y la enfermedad mental

En esta sección se estudiarán las principales estructuras subcorticales del cerebro de los mamíferos. En [Pan98], en referencia al procesamiento emocional instintivo que tiene lugar en nuestras estructuras cerebrales subcorticales, se señala la probabilidad de que, “*en un sentido evolutivo, gran parte del potencial de procesamiento de información en la corteza cerebral es de servicio (se trata de servidores automatizados, a menudo inconscientes) a las instrucciones de los impulsos afectivos que gobernaban el comportamiento previo a la evolución cortical*”.

En [Mor09] se ofrece una gran descripción de la similitud entre los humanos y los demás mamíferos. Nosotros, los humanos, compartimos las mismas estructuras cerebrales subcorticales con todos los demás mamíferos. Textualmente:

Tanto mi gato Buster como yo nos sobresaltamos y gritamos de dolor ante un estímulo doloroso repentino, y nuestras piernas se contraen en respuesta a un golpe en la rótula de la rodilla. La organización neuronal espinal responsable de estas actividades es la misma, tanto en gatos como en seres humanos.

Profundizando en la parte más baja del cerebro, tanto en Buster como en mí, las mismas neuronas controlan las funciones corporales básicas, tales como la regulación de la respiración, la frecuencia cardíaca y los vómitos. Más hacia adelante residen las células nerviosas que regulan los comportamientos de sueño y la vigilia, que son idénticos en los seres humanos y otros mamíferos, y los

resultados de una disfunción en dichas zonas provoca problemas similares, como la narcolepsia y el trastorno de sueño REM. En esta región del cerebro, en todos los mamíferos se encuentran las neuronas que contienen el neurotransmisor dopamina, que degeneran en la enfermedad de Parkinson.

En la base de los hemisferios cerebrales se encuentra la amígdala, en forma de almendra, que lidiá con el temor y la ansiedad, tanto en personas como animales. Los monos y las ratas han contribuido mucho a nuestra comprensión de la amígdala. La corteza cerebral suprayacente es donde todos nosotros los mamíferos analizamos las sensaciones que provienen de la piel, los músculos y las articulaciones a través de la médula espinal, o los ojos y oídos en el caso de la visión y la audición.

Nos alejamos de otros animales, sin embargo, en el gran desarrollo de la parte delantera de nuestra corteza cerebral, los lóbulos frontales, y la mayor proporción de tejido cerebral, llamadas “áreas de asociación”, que integran la información obtenida de las regiones que reciben directamente la información sensorial. Estas últimas regiones se llaman las “áreas primarias sensoriales y motoras”, ya que reciben sensaciones puras y dirigen el movimiento del cuerpo. Se encuentran dentro de los lóbulos frontales, a través de los que nosotros los humanos reflexionamos sobre el pasado y nos preparamos para el futuro. Los animales no tienen esta capacidad última. Los animales se prepararse para el futuro de una forma limitada —por ejemplo, la ardilla piensa en recolectar y enterrar nueces para el invierno—.

El cerebro animal

En [JC05] se relata el primer encuentro del autor con con el tejido cerebral real. “*El cerebro de cerdo fue un gran shock para mí porque cuando comparamos las estructuras de nivel inferior, como la amígdala, a las mismas estructuras en el cerebro humano no podía ver ninguna diferencia en absoluto. El cerebro de cerdo y el cerebro humano se veían exactamente idénticos. Pero cuando miré a la neocorteza, la diferencia fue enorme. el neocortex humano es visiblemente más grande y doblado hacia arriba, y cualquiera puede verlo sin necesidad de un microscopio*”.

En la figura 3.22 se muestra la *amígdala*, el *hipocampo* y el *cuerpo calloso*. Debe recordarse que los hemisferios izquierdo y derecho contienen cada una de estas estructuras, que son imágenes especulares el uno de otro.

La amígdala, el estrés, el TOC y el TEPT

[Cze01] describe la *amígdala* como una estructura “almendrada”. La amígdala se encuentra protegida dentro de cada lóbulo temporal. El lóbulo temporal se encuentra detrás de los

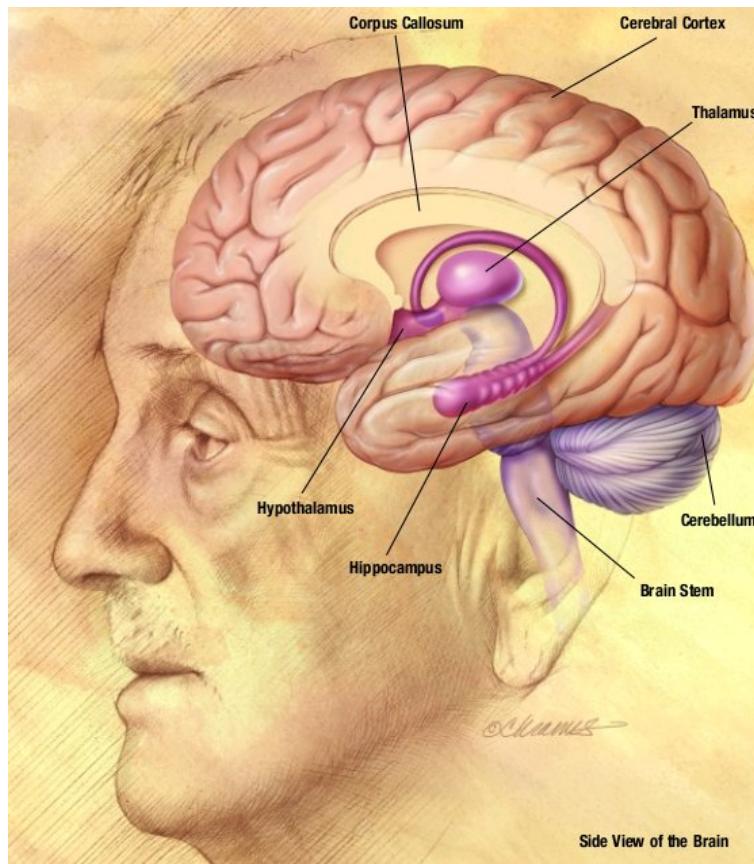


Figura 3.22: Hipotálamo. Tálamo. Hipocampo. Cerebelo. Cuerpo calloso. Neocortex. Tallo cerebral

templos, de ahí su nombre.

En [Dam94] se explica que “*el primer indicio de la relación existente entre la amígdala y las emociones se pueden encontrar en la obra de Heinrich Klüver y Paul Bucy, quienes demostraron que la extirpación quirúrgica de la parte del lóbulo temporal que contiene a la amígdala provoca indiferencia afectiva, entre otros síntomas*”.

Diversas investigaciones demuestran que si la *amígdala* está dañada, el ser humano pierde la capacidad de discernir las emociones, especialmente el miedo y la ira. Según [Joh00], el papel de la amígdala en la percepción de las expresiones faciales “*se mostró claramente en los trabajos de Ralph Adolphs*”. Ralph Adolphs estudió a un paciente que había sufrido un daño bilateral amigdalar sin lesiones significativas en otras partes del cerebro. Aunque este paciente tenía una visión normal y podía percibir rostros, no era capaz de discriminar el contenido emocional de las expresiones faciales negativas del miedo y la ira. Así, todos los rostros le parecían estar sonriendo o mostrando una expresión neutral, incluso los que mostraban un gesto exagerado. El daño a la amígdala también afecta a la capacidad para discernir la emoción en el habla. “*Sophie Scott y sus colegas encontraron que la lesión amigdalar también afecta a la capacidad de percibir el contenido emocional de la entonación del habla,*

a pesar de que su paciente tenía una audición normal. Al igual que con las expresiones faciales en pacientes de Adolphs, las expresiones auditivas del miedo y la ira fueron las más confundidas”.

El hipocampo y la corteza prefrontal se ven reducidos por el estrés, según afirma [PD08], mientras que la amígdala ve incrementado su tamaño y potenciado su funcionamiento, lo que le permite ser más poderosa con el tiempo —incluso llegando a ejercer control sobre nuestro razonamiento—. Dicha actividad potenciada puede exacerbar los síntomas de la enfermedad mental, incluyendo el TOC.

En [Sap04] se hace hincapié en que si bien los *glucocorticoides*⁷ liberados durante episodios de estrés pueden alterar la función del hipocampo y la memoria, los *glucocorticoides* mismos incrementan la excitabilidad de la *amígdala*, permitiendo que las neuronas crezcan más allá de la simple unión entre células.

Especialmente en relación con el TEPT, la experiencia es un factor clave. Los neurocientíficos han descubierto que la experiencia moldea la *amígdala*. Se podría decir que la *amígdala* “aprende”, con el tiempo, el nivel de peligro que debe ser asociado a cada estímulo en particular. En la definición de relevancia incentivo, los autores Vilayanur S. Ramachandran y Lindsay M. Oberman expertos describen el proceso por el cual amígdala puede predecir el peligro. Según [OM06]:

“Cuando una persona ve el mundo, él o ella se enfrenta a una enorme cantidad de información sensorial —sonidos, olores, etc.—. Después de ser procesada en las áreas sensoriales del cerebro, la información es transmitida a la amígdala, que actúa como un portal para el sistema límbico. La amígdala determina la forma en que la persona debe responder emocionalmente: con el miedo (a la vista de un ladrón), la lujuria (al ver a un amante) o indiferencia (cuando se enfrentan a algo trivial). La amígdala envía mensajes al resto del sistema límbico y, finalmente, alcanza el sistema nervioso autónomo, que prepara al cuerpo para la acción. Si la persona se enfrenta a un ladrón, por ejemplo, su ritmo cardíaco aumenta y su cuerpo comienza a sudar para disipar el calor generado por el esfuerzo muscular. La activación autonómica, a su vez, alimenta al cerebro con información de vuelta, amplificando la respuesta emocional. Con el tiempo, la amígdala crea un mapa que detalla el significado emocional de todo en el ambiente del individuo”.

En [LeD96] se ofrece un extracto de un informe de Heinrich Klüver y Paul Bucy sobre el comportamiento de un mono tras la extirpación de sus lóbulos temporales —donde se

⁷Los *glucocorticoides* son hormonas de acción contraria a la de la *insulina* en sangre. También actúan sobre el metabolismo intermedio de grasas y proteínas. Los *glucocorticoides* producidos por el cuerpo humano son el *cortisol*, la *cortisona* y la *corticosterona*.

encuentra la *amígdala*—. Dicho informe explica cómo el mono no muestra ira ni miedo, y parece incapaz de reconocer objetos:

“El animal, estando hambriento y siendo enfrentado a una variedad de objetos —un peine, una perilla de baquelita, una semilla de girasol, un tornillo, un palo, un trozo de manzana, una serpiente viva, una pieza de plátano, y una rata viva— , los recoge indiscriminadamente. Lleva cada objeto a la boca, y lo desecha si no es comestible”.

Klüver y Bucy se refieren a este conjunto de síntomas como “*ceguera psíquica*”, dado que los animales tenían una agudeza visual perfecta, pero eran ciegos a la importancia psicológica de los estímulos. Además, [LeD96] explica cómo después de la eliminación de los lóbulos temporales, los monos se convirtieron en “*hipersexuales*”, tratando de copular con otros monos del mismo sexo o con los miembros de otras especies (actividad sexuales rara vez practicada por “monos normales”).

Cómo la experiencia afecta al cerebro

¿Es posible que el entrenamiento pueda crear conexiones neuronales capaces de automatizar comportamientos instintivos? En [Sap] se examina cómo la comunicación entre las neuronas se ve fortalecida como resultado de la experiencia. Sin duda alguna, este punto es clave en el desarrollo de este proyecto, puesto que su pilar fundamental es la posibilidad de mejorar las habilidades mentales mediante cierto tipo de entrenamiento.

Cuando los terminales de las dendritas de las neuronas se estimulan de forma rápida, la comunicación entre las neuronas se ve potenciada “*potenciada*” por ciertos cambios químicos producidos en el contexto neural. Posteriormente, se necesita una menor estimulación para provocar ese mismo nivel de activación neuronal. Esto es así debido a que en el momento en que se percibe una señal eléctrica a través del axón de la neurona, se produce la liberación de los mensajeros químicos llamados *neurotransmisores* en la sinapsis entre las neuronas, lo que a menudo aumenta la probabilidad de que otras neuronas se disparen en un especie de reacción en cadena. En otras palabras, el autor afirma que la potencia de activación neuronal se incrementa. Esta potenciación en un tiempo largo es lo que se conoce como “*potenciación a largo plazo*”, y constituye la base para el aprendizaje y la memoria, posiblemente incluso de las formas perjudiciales de aprendizaje, tales como el trastorno de estrés post-traumático (TEPT).

En [Mac49] el autor teoriza sobre el proceso de activación neuronal: “*Es posible que si un cierto patrón eléctrico de información fuera repetido durante un período prolongado de tiempo, o a intervalos repetidos, en el circuito neuronal, las células nerviosas se sensibilizan de forma permanente a responder a este patrón particular en algún momento futuro. Tal mecanismo proporciona una variedad de memoria perdurable.*”.

Citando textualmente a [Gal]:

Con el tiempo, repetidas experiencias estresantes pueden, literalmente, y no sólo en sentido figurado, alterar el sistema nervioso de las personas. La investigación en animales ha demostrado que cuando a una rata se le da una pequeña descarga no muestra ninguna reacción notable, pero al ser expuesta a factores de estrés tales como la repetición de descargas durante cinco días consecutivos, muestra signos de respuesta al estrés, cuando se expone durante siete u ocho días, la rata tiene un ataque, y a partir de entonces se mantiene alterada y ataca con poca o ninguna provocación. Experimentos de este tipo, por supuesto, no se llevan a cabo con personas, pero Philip Gold y otros neurocientíficos piensan ahora que en los seres humanos, también mediante la activación de una cascada de reacciones químicas, puede producirse un estrés crónico grave, sobre todo en los primeros años de vida, y causar cambios en los genes de manera que se altere la funcionalidad de las células del cerebro de forma permanente. Debido a que las neuronas requieren un tipo particular de entrada para activarse o desactivarse, sólo algunos miles de neuronas, cada una de las cuales está implicado en algún aspecto de la estructura celular o de comunicación, se activan en un momento dado. Cuando una persona de temperamento vulnerable es constantemente bombardeados con estímulos perturbadores, dice Gold, los genes que se activan son los que participan en los componentes celulares de la respuesta al estrés.

La neurotransmisión en la amígdala a veces es activada para generar comportamientos impulsados por la *dopamina*, como conductas orientadas a resolver problemas de control corporal y de confianza. En circunstancias normales, esto puede ser interpretado como un instinto de supervivencia, pero en situaciones de estrés extremo, y especialmente cuando no hay disponible una salida para la energía, estos comportamientos pueden convertirse en obsesiones o compulsiones. [Sap05] describe cómo los monos liberan dopamina anticipándose a una recompensa alimenticia. Su excitación no alcanza el máximo cuando la comida por fin les llega, sino antes de ese instante. “*Se trata de la anticipación de la recompensa. Se trata de la expectativa y la confianza*”.

3.5. Neuroplasticidad

Como se explica en [SBS⁺12], diversas investigaciones en el campo de la neurología concluyen que el cerebro humano, en lugar de tratarse de una “*masa de células estática*”, se trata de un sistema dinámico de redes neuronales con gran potencial de crecimiento bajo unas circunstancias favorables. La publicación diferencia esas circunstancias en dos períodos:

- **Periodo crítico:** Cuando el cerebro del individuo es más sensible a las influencias del entorno.
- **Periodo sensible:** Cuando el cerebro es menos susceptible a la influencia exterior.

Sin embargo, el fin del periodo crítico no implica que el cerebro “se estanque” y deje de ser susceptible de verse mejorado, sino que el nivel de neuroplasticidad de esa etapa no será alcanzado fuera de ella. Por tanto, durante el periodo sensible es necesario un mayor esfuerzo para mejorar algún aspecto cerebral que para obtener la misma mejora durante el periodo crítico.

En [Fac06] se introduce la idea de cómo el Síndrome de Fatiga Crónica (SFC) se relaciona con una reducción de la materia gris cerebral, sirviéndose de diversas técnicas no invasivas de obtención de información cerebral mediante imágenes —como la Imagen por Resonancia Magnética (IRM) o la Morfometría Basada en Vóxel (VBM)—, para explicar después que la plasticidad cerebral permite que algunos pacientes con este problema recuperen cierta funcionalidad de procesamiento cerebral perdida.

El cerebro de un músico es un objetivo de estudio perfecto para analizar la neuroplasticidad: los pianistas, por ejemplo, deben coordinar la producción de hasta 1800 notas por minuto con ambas manos (ver figura 3.23); la composición musical, además, requiere de la integración de información sensorial y motora, junto a una precisa observación de la ejecución. Estos argumentos son los que presenta [MAJ02] para justificar cómo el cerebro de un músico es un buen ejemplo de neuroplasticidad. En dicho artículo se explica cómo varios años de experiencia musical —sobre todo cuando los individuos empiezan sus estudios musicales a una edad temprana— pueden estar relacionados con un aumento en las materias gris y blanca, así como en el volumen, de las regiones 31, 32, 36, 39 y 41 del cerebro (ver figura 3.21).

[MCKV01] relaciona el dolor crónico y patológico con la neuroplasticidad. Para ello estudia el *síndrome del miembro fantasma* —percepción de sensaciones de que un miembro amputado todavía forma parte del cuerpo y funciona correctamente—. Antiguamente se solía pensar que estas sensaciones se debían a que el cerebro seguía recibiendo mensajes de los nervios que originalmente llevaban impulsos desde el miembro amputado. Sin embargo, la creencia actual se basa en que el cerebro sigue conservando un área dedicada a dicho miembro, por lo que el individuo sigue “sintiéndolo”. Los estímulos sensoriales actúan sobre sistemas neuronales que fueron modificados por entradas anteriores, y su comportamiento y salida están influenciados por la “memoria” de esos eventos del pasado. [MCKV01] relaciona esto con la neuroplasticidad en el sentido en que en la mayoría de casos la sensación del miembro fantasma desaparece, al menos de forma temporal, sometiendo a los individuos a visiones que revelen la no correspondencia entre el miembro real y el percibido.

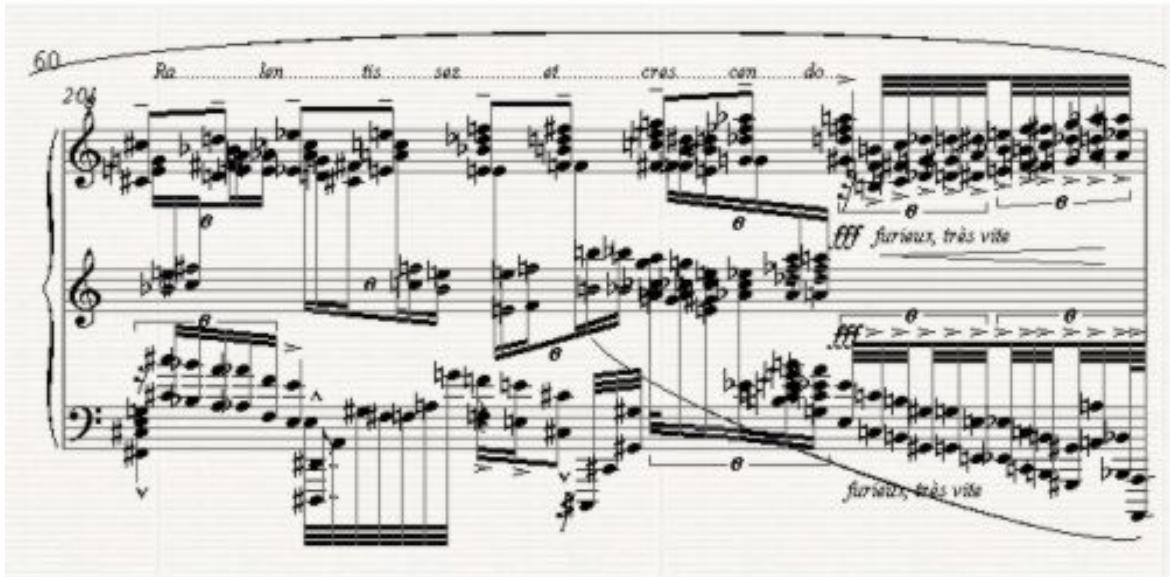


Figura 3.23: Ejemplo de partitura de piano compleja, extraída de una pieza de Kaikhosru Shapurji Sorabji

El estrés crónico —el cual puede derivar en depresión— interrumpe la neuroplasticidad de forma temporal, explica [PD08], mientras que un tratamiento antidepresivo ofrece el resultado opuesto.

Yendo más lejos, en [MDB⁺08] se desarrolla la idea del *cambio cualitativo* —aprendizaje de nuevas tareas— como principal impulsor de la modificación estructural del cerebro, por encima de la práctica reiterada de un ejercicio conocido. Para ello se estudia el efecto del aprendizaje de juegos malabares por parte de 20 individuos, observando cómo este ejercicio produce una alteración de la materia gris del *córtex occipitotemporal* con tan sólo una semana de entrenamiento.

3.5.1. Memoria (Memory)

Memoria de trabajo

En [JBJP08] se estudia la forma de estimular la memoria de trabajo a lo largo de cuatro experimentos, empleados para mejorar lo que denomina “*inteligencia fluida*”. Ésta es una habilidad compleja humana que posibilita la adaptación del pensamiento a nuevas situaciones o problemas a los que nunca antes nos habíamos enfrentado.

El hecho de mejorar la memoria de trabajo como un medio para conseguir un fin mayor, como el mencionado, pone de manifiesto la importancia de este tipo de memoria, que es empleada en la resolución de problemas o realización de tareas complejas.

La memoria de trabajo también es empleada como medio para conseguir un objetivo diferente —y más ambicioso si cabe— en [OWK04]. El texto relata como, mediante la estimulación de la memoria de trabajo a través de diversos experimentos, se consigue incrementar la actividad prefrontal y parietal del cerebro. Si recordamos la diferenciación de los lóbulos cerebrales expuesta en la sección 3.4.5, los lóbulos prefrontal y parietal albergan el procesamiento de emociones y de la información sensorial, respectivamente.

Recuerdo espacial (Spatial recall)

Este aspecto funcional del cerebro se encarga del almacenamiento a largo plazo de representaciones visuales. Una forma elegante de explicar su comportamiento es mediante la creación de un modelo del mismo, como hace [BB], y la posterior simulación de su comportamiento en entornos complejos. La lectura relata el efecto de daños cerebrales artificiales producidos sobre el modelo, concretamente en lóbulo parietal, y utilizan sus resultados para la realización de predicciones sobre la representación neural que tiene lugar en el cerebro.

En líneas generales, el modelo consiste en lo siguiente: un *parahipocampo*⁸ genera una representación alocéntrica de los objetos a partir de un *flujo visual ventral* —qué es el objeto—, mientras que el lóbulo parietal interpreta un *flujo visual dorsal* para generar una representación egocéntrica —dónde se encuentra el objeto—. Una conexión bidireccional recurrente entre las regiones egocéntrica y alocéntrica produce la memoria a largo plazo.

Atendiendo al objetivo del presente proyecto, resulta de aún más interés el estudio de [Hir04], donde se analiza el efecto que los videojuegos pueden ejercer sobre las habilidades espaciales. Una de las hipótesis más interesantes del estudio es la existencia de un comportamiento diferente entre sexos ante la estimulación ejercida por los videojuegos. Mediante un experimento en el que algunos individuos son sometidos a la práctica de un juego 3D durante 5 horas seguidas se puede observar, entre otras cosas, que las mujeres presentan una mejora de las habilidades espaciales ligeramente mayor que los hombres.

Recuerdo cara-nombre (Face-name recall)

Denominamos *recuerdo cara-nombre* a la habilidad cerebral para mantener una correspondencia entre nombres de personas y sus caras. En [HHG00] se propone un modelo estructural que explica el sistema neural encargado de esta habilidad, una de las más desarrolladas en los seres humanos frente a otros animales. No obstante, se concluye que la diferenciación fisiológica de los elementos participantes en este comportamiento es algo difusa, dando pie a la realización de otros posibles modelos que lo expliquen.

Por otro lado, [TKD⁺09] muestra algunos resultados experimentales sobre el reconocimiento facial por parte de algunos voluntarios: los individuos son sometidos a la visua-

⁸Estructura compleja que recubre el hipocampo por su parte inferior. También es denominado *circonvolución parahipocámpica*

lización de un conjunto de rostros —de forma secuencial— durante unos instantes, para mostrarles otro conjunto diferente (con ciertos rostros en común con el primero) después. Los individuos deben reconocer las caras que han aparecido en ambos conjuntos y las que aparecen en el segundo conjunto y no formaban parte del primero.

El hipocampo es el absoluto protagonista en la asociación cara-nombre.

3.5.2. Resolución de problemas (Problem solving)

Aritmética (Arithmetic)

Centrándonos en los efectos neuroplásticos relacionados con la aritmética, en [DMCW04] se explica cómo la realización de cálculos por parte de los seres humanos se ve reflejada en la activación del Surco Intra-Parietal (SIP)⁹. Un estudio adicional que corrobora este hecho descubre que ciertas patologías de esa zona cerebral producen la denominada *acalculia* —imposibilidad para la realización de cálculos matemáticos, sencillos o complejos—.

Las propuestas futuras que se realizan en [DMCW04] abarcan la comprensión de las homologías y diferencias entre la actividad cerebral de determinados simios y los seres humanos.

Razonamiento lógico (Logical reasoning)

El razonamiento lógico tiene lugar principalmente en el lóbulo frontal, y consiste en la habilidad para combinar procesos cognitivos y la flexibilidad para reconocimiento de patrones, extracción de conclusiones y toma de decisiones. Como recurso de interés para adentrarse en el mundo de la lógica, [Jul04] ofrece un manual completo sobre las lógicas de proposiciones y predicados, enfocado al aprendizaje por parte de personas del ámbito de la Ingeniería Informática.

El razonamiento lógico permite, partiendo de uno o más juicios, derivar la validez o falsedad de otro juicio distinto. Para [DBM08] “*el sistema de procesamiento consciente y reflexivo es vital para el razonamiento lógico*”. Por tanto, se podría afirmar que el razonamiento lógico es un proceso consciente.

Razonamiento cuantitativo (Quantitative reasoning)

Los seres humanos empleamos el razonamiento cuantitativo para realizar estimaciones. Resulta de utilidad para comparar precios de productos y cuotas, valor de impuestos, etc. Especialmente en el mundo empresarial, el razonamiento cuantitativo es empleado para la toma de decisiones de negocio y selección de estrategias.

⁹Hendidura irregular en la superficie convexa del lóbulo parietal, que marca la división entre los lóbulos parietales superior e inferior.

3.5.3. Atención (Attention)

Concentración (Focus)

La concentración no es más que la capacidad para prestar atención a la información pertinente, ignorando distracciones irrelevantes. Resulta de gran importancia en situaciones que impliquen realizar una tarea como la lectura en un entorno ruidoso como una cafetería. La densidad de información que llega a nuestros sentidos es inmensa, por lo que la concentración para desechar cualquier información no relevante es de alta importancia.

En [BKP⁺06] se exponen los resultados obtenidos tras un experimento en el que personas jóvenes y mayores son sometidas a un cierto programa de entrenamiento centrado en el control de la atención. Los resultados indican una mejora sustancial en la atención —gracias a la *neuroplasticidad*—, curiosamente equivalente en ambos grupos de edad.

Campo visual (Visual field)

El campo visual consiste en el espacio que un individuo puede ver en un momento dado sin mover sus ojos. Implica a la visión periférica y resulta de gran importancia cuando se conduce un vehículo o se practican ciertos deportes. La mejora del campo visual, por tanto, supone un mecanismo muy efectivo para mejorar también nuestro rendimiento llevando a cabo esas actividades.

El campo visual disminuye con la edad, como expone [KBD⁺88]. Sin embargo, mediante cierto tipo de videojuegos es posible realizar mejoras importantes en esta habilidad, entre otras, tal y como expresa [C.S03] en anteposición a lo que comúnmente es denominado “aprendizaje perceptivo”¹⁰.

3.5.4. Velocidad (Speed)

Procesamiento de información (Information processing)

Esta habilidad mental consiste en la captura e interpretación de la información sensorial que recibimos. Realizar esta tarea de forma rápida nos permite mejores tiempos de respuesta y capacidad de adaptación frente a cambios en nuestro entorno. Una de las tareas cotidianas en las que la velocidad de procesamiento de información adquiere gran importancia es la conducción.

“A medida que el ser humano envejece”, tal y como explica [KJO⁺05], “el rendimiento de nuestras habilidades funcionales decrece de forma significativa”, lo que afecta de forma directa a nuestra velocidad de procesamiento de información. No obstante, es en [BER07] donde se muestran resultados claros —mediante seis estudios diferentes— del impacto que el entrenamiento cognitivo y produce sobre la velocidad de procesamiento. Curiosamente, además, concluye que “la edad y la educación tienen un impacto muy pequeño o nulo sobre

¹⁰Mejora de la respuesta a estímulos mediante una exposición duradera a los mismos

la ganancia producida por el entrenamiento”.

Orientación espacial (Spatial orientation)

La orientación espacial consiste en encontrar la dirección deseada en movimiento en un entorno físico. Se pone de manifiesto cuando nos desplazamos por una ciudad que no conocemos, así como durante la lectura e interpretación de un mapa. Se trata de una capacidad innata en gran parte, pero que puede mejorarse mediante entrenamiento.

Esta habilidad presenta diferencias entre hombres y mujeres. Sin embargo, [FSP07] afirma que dichas diferencias desaparecen parcialmente tras la práctica de ciertos ejercicios mediante videojuegos. Por otro lado, [OF94] documenta los efectos del entrenamiento de la orientación espacial mediante videojuegos, así como posibles formas de medir esa mejora —de gran interés para el proyecto que nos ocupa—.

3.5.5. Flexibilidad (Flexibility)

Control de impulsos (Response inhibition)

El control de los impulsos es una habilidad mental humana que nos diferencia del resto de animales. Nos permite priorizar acciones teniendo en mente los objetivos a los que nos enfrentamos, controlando nuestros impulsos animales. En [DTHS] se habla del control de impulsos como la “*la función cerebral más evolucionada del ser humano*”.

“*Conocer cómo está implementado este control de impulsos en el cerebro es uno de los mayores problemas de la neurociencia contemporánea*”, afirma [NRLP⁺].

Planificación (Planning)

La planificación es la capacidad cerebral que posee el ser humano para priorizar diferentes tareas a lo largo de un periodo de tiempo. Resulta de gran importancia, en combinación con la capacidad para intercambiar el foco de atención entre esas tareas (habilidad que se estudiará a continuación).

Commutación de tareas (Task switching)

La commutación de tareas supone la adaptación al cambio en el entorno, detectando cuándo y en qué situaciones resulta más beneficioso pasar a realizar otra tarea en lugar de continuar con la que estaba en curso, así como en qué momento volver a la primera tarea. Resulta inmediata la relación de esta habilidad con lo que en computación se conoce como *multitarea*¹¹. Gracias a la commutación de tareas el ser humano es capaz de cambiar su foco de atención en base a cambios ajenos a él en su contexto.

En relación con la memoria de trabajo (vista en la sección 3.5.1), en [OK04] se realiza una

¹¹Característica de los sistemas operativos modernos de permitir que varios procesos sean ejecutados (en apariencia) al mismo tiempo, compartiendo uno o más procesadores.

prueba de significación de la hipótesis de que, con una práctica adecuada, el ser humano es capaz de emplear la memoria de trabajo para ejecutar dos operaciones simultáneas.

Fluidez verbal (Verbal fluency)

La fluidez verbal supone el acceso veloz al vocabulario conocido durante una conversación o tarea de escritura. Consiste en el ingenio y la capacidad de intercambio y conexión de palabras, y resulta crucial para la comunicación.

Esa conocida sensación de “*tener una palabra en la punta de la lengua*” está relacionada estrechamente con la fluidez verbal. Depende de multitud de factores, como la edad, cuyo efecto se analiza en [BS04].

Capítulo 4

Método de trabajo

A lo largo de este capítulo se presenta la metodología de desarrollo empleada durante el ciclo de vida del proyecto, analizando las diferentes etapas e iteraciones por las que ha pasado el mismo. Además se detallan los diferentes entregables producidos durante el proceso de desarrollo.

En primer lugar se presenta una visión global sobre el proceso de desarrollo, así como algunos detalles sobre las etapas e iteraciones por las que ha pasado el desarrollo del proyecto. En segundo lugar se estudia el análisis de requisitos del sistema, haciendo énfasis en aspectos como la calidad y la evolución, y no sólo en la funcionalidad. Después se ofrece un estudio detallado sobre los casos de uso del sistema, para dar paso al diseño e implementación del mismo. En la sección 4.7 se hará también un repaso por las tecnologías empleadas para implementar BreakBrain. Por último se presentará la documentación de pruebas.

4.1. Proceso Unificado de Desarrollo

Para este proyecto se ha seguido una metodología de desarrollo iterativa, basada en el Proceso Unificado de Desarrollo (PUD) —también conocido como Rational Unified Process (RUP)—. Se trata de un “*Proceso de Ingeniería del Software*”, según [Rat98]. Su meta es asegurar la producción de software de alta calidad, satisfaciendo las necesidades del usuario final y acotando la planificación y el presupuesto de una forma predecible.

4.1.1. Visión global del proceso

PUD hace una clara distinción entre las dos dimensiones sobre las que se fundamenta el proceso, una dimensión estática y una dimensión dinámica:

- El eje horizontal representa el tiempo, y hace referencia al aspecto dinámico del proceso, en términos de *etapas e iteraciones*.
- El eje vertical representa el aspecto estático del proceso, descrito en forma de *actividades, flujos de trabajo o artefactos*.

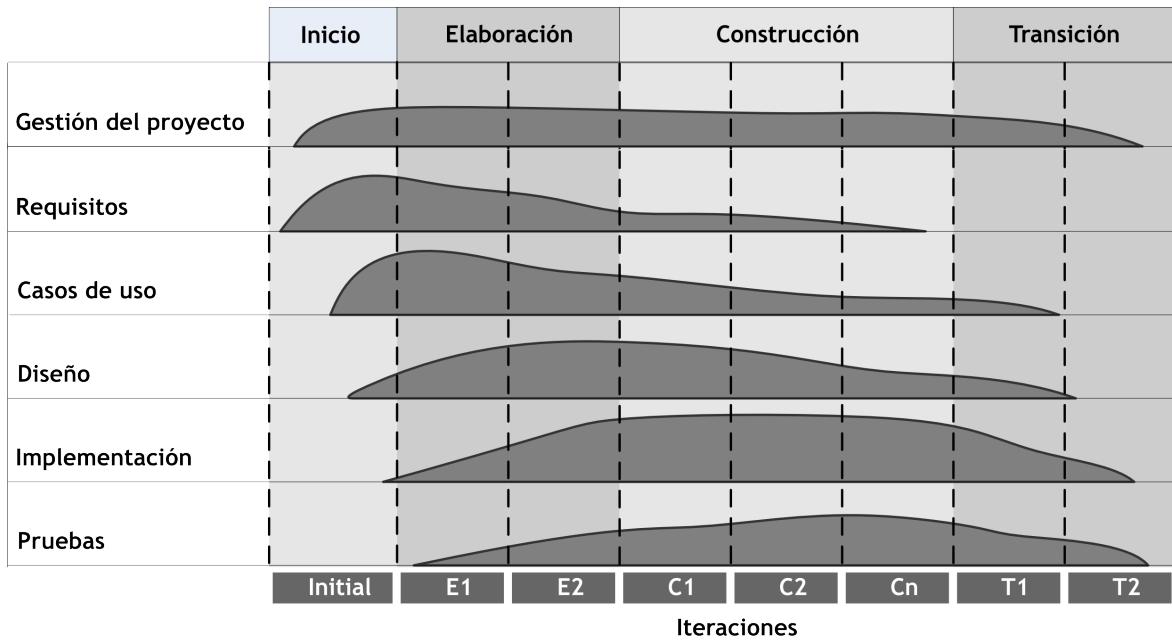


Figura 4.1: Proceso Unificado de Desarrollo: etapas, entregables e iteraciones

En la figura 4.1 se presenta gráficamente la estructuración del Proceso Unificado de Desarrollo a lo largo de las dos dimensiones mencionadas.

4.1.2. Etapas e iteraciones (dimensión dinámica)

Tal y como se detalla en [Rat98], el ciclo de vida del software según el Proceso Unificado de Desarrollo se divide en 4 etapas o fases:

1. Fase de inicio
2. Fase de elaboración
3. Fase de construcción
4. Fase de transición

La primera etapa, como inicio del desarrollo, tiene como finalidad el establecimiento de los requisitos del proyecto, los aspectos clave del mismo y las limitaciones conocidas para su desarrollo. En esta etapa se realiza la primera aproximación del modelo de casos de uso, y se presenta un plan de gestión del proyecto, detallando sus fases e iteraciones. Esta etapa puede concluir con la generación de un sencillo prototipo.

La fase de elaboración es la dedicada al análisis del dominio del problema, la aproximación de la arquitectura del sistema y la eliminación de los riesgos más críticos. Esta etapa es la más crítica, puesto que al finalizarla se puede considerar por terminada la *ingeniería dura*. El diseño del sistema es el protagonista de esta fase. El análisis de requisitos y el modelo de

casos de uso se dejan prácticamente terminados, y se da comienzo a la implementación.

La etapa de construcción es dedicada a desarrollar todos los componentes restantes y a integrarlos en el producto. Todas las características del software son probadas. Los esfuerzos se centran en la gestión de recursos y el control de operaciones para optimizar los costes, planificación y calidad. En esta etapa se crean los manuales de usuario.

El propósito de la fase de transición es la liberación del producto a la comunidad de usuarios. Una vez que el producto se entrega al usuario final, es posible que surjan problemas que corregir o se generen nuevos requisitos que deban ser implementados.

4.1.3. Artefactos (dimensión estática)

Un *artefacto* es una pieza de información que es producida, modificada o utilizada por un proceso. Los artefactos son los productos tangibles del proyecto, los elementos generados durante el desarrollo hacia el producto final. Son la salida de una actividad y entrada de otra, y pueden ser modelos, documentos, archivos de código fuente, archivos ejecutables, etc.

Los artefactos generados durante el desarrollo de este proyecto, los cuales serán presentados más adelante, son los siguientes:

- Especificación de requisitos
- Despliegue del sistema
- Modelo de casos de uso
- Diseño de la arquitectura software
- Implementación
- Documentación de pruebas

4.2. Iteraciones

En el apéndice A se ofrece un análisis detallado de la evolución iteracional del proceso de desarrollo del proyecto. En total se han realizado 7 iteraciones, cubriendo las etapas de Inicio, Elaboración y Construcción.

En el cuadro 4.1 se ofrece un resumen de dichas estadísticas, mostrándose la evolución de cada artefacto con la conclusión de cada nueva iteración.

4.3. Análisis de requisitos

Llegado este punto, procede estudiar los requisitos que debe satisfacer el sistema completo. En primer lugar se aproximará un listado de requisitos definidos de forma sencilla y

-	-	Inicio	Elaboración		Construcción			
-	-	I1	E1	E2	C1	C2	C3	C4
-	Duración	75h	50h	50h	75h	125h	80h	70h
Artefactos	Requisitos	53.6 %	71.4 %	85.7 %	89.3 %	96.4 %	100 %	100 %
	Casos de uso	27.8 %	50 %	77.8 %	94.4 %	100 %	100 %	100 %
	Diseño	10 %	25 %	75 %	80 %	85 %	90 %	95 %
	Implementación	0 %	5.2 %	17.3 %	20.2 %	50.8 %	63.1 %	75 %
	Pruebas	0 %	0 %	5 %	10 %	33.4 %	59.1 %	71 %

Cuadro 4.1: Evolución del proceso de desarrollo. Los porcentajes se han estimado teniendo en cuenta las líneas de código (para el caso de la implementación), la cobertura para las pruebas, y el número de elementos especificados (para el caso de los requisitos y los casos de uso)

superficial, para después realizar un análisis detallado de los mismos, siguiendo las directrices del estándar IEEE 830.

4.3.1. Primera aproximación a los requisitos

Para facilitar la visualización y posterior análisis de los mismos, se realiza la siguiente división:

- **Requisitos de servidor**

Requerimientos específicos del funcionamiento interno del sistema servidor, tanto del servidor web como del almacenamiento en base de datos y de comunicación.

- **Requisitos de usuario**

Requerimientos asociados a la utilización por parte de un usuario. Están limitados a los aspectos referidos a los clientes del sistema.

- **Requisitos no funcionales**

Requerimientos referidos a la seguridad, la evolución del software, al soporte, etc.

A continuación se enumeran los requisitos de cada grupo. Se trata de un análisis superficial. Con posterioridad se procederá a detallar esos requisitos.

Funcionalidad referida al servidor

La primera aproximación a los requisitos asociados al funcionamiento del sistema interno es la siguiente:

- El sistema debe permitir la comunicación con usuarios.
- El sistema debe permitir almacenar y recuperar la información personal de los usuarios.

- El sistema debe permitir almacenar y recuperar la evolución cerebral de los usuarios.
- El sistema debe permitir recuperar listados de usuarios.
- El sistema debe poder recomendar usuarios afines a uno dado.
- El sistema debe poder recomendar juegos a los usuarios en base a su estado actual y evolución temporal.
- El sistema debe ofrecer un listado con los juegos disponibles.
- El sistema debe soportar la comunicación entre usuarios mediante mensajes escritos.

Funcionalidad referida al usuario

La primera aproximación a los requisitos asociados al usuario es la siguiente:

- Los usuarios podrán hacer uso de toda la web con cualquier navegador moderno, sin necesidad de plugins o componentes adicionales.
- Los usuarios deben poder autenticarse en el sistema.
- Los usuarios podrán cerrar su sesión.
- Los usuarios no autenticados no deben tener acceso al resto del sitio web.
- Los usuarios podrán registrarse.
- El email de registro debe ser auténtico.
- Los usuarios podrán recuperar su contraseña si no la recuerdan.
- Los usuarios podrán seleccionar los parámetros cerebrales que deseen ejercitarse.
- Los usuarios podrán editar su información personal.
- Los usuarios podrán seleccionar una imagen de avatar que les represente.
- Los usuarios podrán seguir a otros usuarios.
- Los usuarios podrán dejar de seguir a otros usuarios.
- Los usuarios deben tener a su disposición un resumen de su evolución.
- Los usuarios deben tener a su disposición un listado con noticias sobre los usuarios a los que sigue.
- Los usuarios deben poder ver su perfil cerebral completo, con el estado actual y la evolución semanal, mensual y anual.
- Los usuarios podrán hacer búsquedas de usuarios.
- Los usuarios recibirán recomendaciones de juegos y de usuarios a los que seguir.
- Los usuarios podrán visualizar un listado completo de juegos disponibles, diferenciando los juegos multijugador de los juegos de un solo jugador.
- Los usuarios podrán ejecutar los juegos.

Aspectos ajenos a la funcionalidad

- El acceso a la base de datos debe ser rápido, para evitar esperas por parte del cliente.
- La lógica de los juegos multijugador debe ser rápida, por lo que el servidor de juegos multijugador deberá tener muy buen rendimiento.
- El rendimiento de los juegos deberá ser óptimo, por lo que el navegador de los clientes deberán ejecutar los juegos de forma eficiente.
- El sistema debe soportar cientos de miles de peticiones simultáneas y responder de forma rápida y robusta.
- El acceso al sistema debe ser controlado mediante email y contraseña.
- Las personas no autenticadas no tendrán acceso a ninguna información ni juego.

4.3.2. Requisitos funcionales

Partiendo del análisis preliminar de requerimientos, los requisitos funcionales de los que parte el desarrollo del proyecto, basándose en el estándar IEEE 830, son los detallados a continuación.

Para cada requisito se especifica una descripción sencilla del mismo, una valoración de su importancia (esencial, condicional u opcional), y una descripción de su validez, que incluye los mecanismos para medir su nivel de consecución, los medios para cumplir dicho requisito y una descripción de la relevancia que tiene dicho requisito en el desarrollo completo del sistema.

- RF-1**
- Descripción: El sistema permitirá que los clientes se conecten al servidor e interactúen con la web.
 - Importancia: Esencial.
 - Validez:
 - *Medible:* El navegador web podrá conectarse al servidor web.
 - *Alcanzable:* El servidor ofrecerá una interfaz de peticiones GET y POST común.
 - *Relevante:* Este requisito es vital para la utilización del sistema por parte de los usuarios.
- RF-2**
- Descripción: El sistema permitirá que los clientes intercambien mensajes con el servidor.
 - Importancia: Esencial.
 - Validez:
 - *Medible:* El navegador web y el servidor deben intercambiar mensajes.
 - *Alcanzable:* El servidor ofrecerá una interfaz de websockets para el intercambio de mensajes.

- *Relevante*: Este requisito es vital para la utilización del sistema por parte de los usuarios.

RF-3 ■ Descripción: El sistema permitirá almacenar y recuperar información personal de los usuarios.

■ Importancia: Esencial.

■ Validez:

- *Medible*: El sistema recuperará la información personal de cualquier usuario dados su identificador y, por cuestiones de seguridad, su contraseña.
- *Alcanzable*: El servidor estará conectado a una base de datos donde se almacenen y de donde se recuperen todos los datos.
- *Relevante*: Este requisito es vital para que los usuarios puedan interactuar con el sistema.

RF-4 ■ Descripción: El sistema permitirá almacenar y recuperar la evolución del entrenamiento cerebral de los usuarios.

■ Importancia: Esencial.

■ Validez:

- *Medible*: El sistema recuperará el perfil cerebral de cualquier usuario dados su identificador y, por cuestiones de seguridad, su contraseña.
- *Alcanzable*: El servidor estará conectado a una base de datos donde se almacenen y de donde se recuperen todos los datos.
- *Relevante*: Este requisito es vital para que los usuarios puedan tener acceso a su evolución.

RF-5 ■ Descripción: El sistema permitirá recuperar listados de usuarios.

■ Importancia: Esencial.

■ Validez:

- *Medible*: El sistema realizará búsquedas en base a consultas textuales sencillas, obteniendo un listado con los resultados de la misma.
- *Alcanzable*: El servidor estará conectado a una base de datos donde se almacenen todos los usuarios, y lanzará sobre ella consultas específicas.
- *Relevante*: Este requisito es vital para que los usuarios puedan buscar a otros usuarios y empezar a seguirles, retarles a algún juego, etc.

RF-6 ■ Descripción: El sistema recomendará juegos a los usuarios, en base a su evolución y características.

■ Importancia: Condicional

■ Validez:

- *Medible*: Los usuarios tendrán en la sección de juegos un apartado con los juegos a los que el sistema les recomienda jugar, en base a su evolución

cerebral.

- *Alcanzable*: Se desarrollará un algoritmo de recomendación de juegos a usuarios basado en los parámetros cerebrales elegidos por ellos en su perfil y en la evolución de dichos parámetros a lo largo del tiempo.
- *Relevante*: Este requisito es muy recomendable para mejorar la experiencia de usuario, además de para ayudar a mejorar las capacidades mentales en las que el usuario se vea menos hábil o cuya mejora le cueste más.

RF-7 ■ Descripción: El sistema recomendará usuarios a los que seguir y retar, en base a sus preferencias y habilidades mentales.

■ Importancia: Condicional

■ Validez:

- *Medible*: En la sección de búsqueda de usuarios de la web se ofrecerá un apartado con los usuarios más recomendables para el usuario actual.
- *Alcanzable*: Se desarrollará un algoritmo de recomendación de personas a personas basado en los parámetros cerebrales elegidos por ellos en su perfil y en la evolución de dichos parámetros a lo largo del tiempo.
- *Relevante*: Este requisito es muy recomendable para mejorar la experiencia de usuario, además de para ayudar a mejorar el aspecto social de BreakBrain.

RF-8 ■ Descripción: El sistema ofrecerá un listado de juegos disponibles, permitiendo jugar a cualquiera de ellos.

■ Importancia: Esencial

■ Validez:

- *Medible*: Los usuarios tendrán a su disposición un apartado de la web con todos los juegos disponibles.
- *Alcanzable*: La web se dotará con una sección de juegos en la que se listen todos y cada uno de los juegos disponibles, categorizados por las capacidades mentales que ayudan a estimular.
- *Relevante*: Este requisito es vital para conseguir que BreakBrain sea lo que se espera.

RF-9 ■ Descripción: El sistema permitirá la comunicación entre usuarios mediante mensajes con el servidor.

■ Importancia: Esencial

■ Validez:

- *Medible*: Los navegadores cliente de dos usuarios podrán comunicarse entre sí, de tal forma que puedan ejecutar juegos multijugador o comunicarse por medio de chat.
- *Alcanzable*: Se emplearán websockets (un módulo en el cliente y otro en el

servidor), para mantener una comunicación TCP full-duplex entre ambas partes. El servidor hará de intermediario entre dos clientes, simulando una comunicación directa entre ellos.

- **Relevante:** Este requisito es vital para poder ofrecer juegos multijugador en la red social.

RF-10 ■ Descripción: Los usuarios podrán hacer uso de la web completa —incluyendo los juegos— con cualquier navegador moderno, sin necesidad de plugins o extensiones adicionales.

■ Importancia: Condicional

■ Validez:

- **Medible**: La web funcionará correctamente en diversos navegadores actuales, como las últimas versiones de Firefox, Chrome, Opera, etc.
- **Alcanzable**: Se emplearán estándares web HTML5 y CSS3 para el desarrollo completo del sitio web, incluidos los juegos y cualquier otro componente de la red social.
- **Relevante**: Se trata de un requisito importante, para imponer el mínimo número de restricciones al usuario final.

RF-11 ■ Descripción: Los usuarios podrán autenticarse en el sistema para tener acceso a todo el sitio web.

■ Importancia: Esencial

■ Validez:

- **Medible**: Al cargar BreakBrain se mostrará una pantalla de inicio de sesión, y ésta será la única visible hasta que el visitante inicie una sesión válida.
- **Alcanzable**: El sitio web mantendrá un objeto de sesión con la información de usuario. Al iniciar sesión se creará dicho objeto, y al cerrarla se eliminará.
- **Relevante**: Este es un requisito indispensable para que la web quede restringida a usuarios registrados y para que estos puedan consultar su evolución, seguir a otros usuarios, etc.

RF-12 ■ Descripción: Los usuarios autenticados podrán cerrar su sesión.

■ Importancia: Condicional.

■ Validez:

- **Medible**: En la parte superior derecha de la web (con una sesión iniciada) se mostrará un botón para cerrar sesión.
- **Alcanzable**: El sitio web mantendrá un objeto de sesión con la información de usuario. Al iniciar sesión se creará dicho objeto, y al cerrarla se eliminará.
- **Relevante**: Este requisito es importante para que diferentes usuarios puedan utilizar el mismo navegador.

- RF-13**
- Descripción: Cualquier persona podrá registrarse para obtener una cuenta de usuario.
 - Importancia: Esencial
 - Validez:
 - *Medible:* Al acceder a BreakBrain sin una sesión iniciada, junto al formulario de login, aparecerá un formulario de registro para crear una cuenta de usuario.
 - *Alcanzable:* El sistema permitirá la creación de las estructuras de datos necesarias para la creación de un nuevo usuario, a petición de cualquier visitante desde la página de inicio.
 - *Relevante:* Es un requisito básico para que los usuarios puedan obtener una cuenta y utilizar la red social.
- RF-14**
- Descripción: El email de registro debe ser auténtico.
 - Importancia: Condicional
 - Validez:
 - *Medible:* La red social no debe ser accesible a una cuenta de usuario hasta que el email haya sido correctamente comprobado.
 - *Alcanzable:* Durante el proceso de registro, el sistema debe asegurar que el email proporcionado es correcto. Para ello enviará a dicha dirección un enlace de activación. El usuario hará clic sobre el enlace para activar la cuenta y poder empezar a utilizarla.
 - *Relevante:* En BreakBrain, el identificador de los usuarios es su email. Por tanto, este requisito es importante para asegurar el contacto con cualquier usuario.
- RF-15**
- Descripción: Los usuarios que no recuerden su contraseña podrán crear una nueva.
 - Importancia: Condicional
 - Validez:
 - *Medible:* La pantalla de inicio mostrará la posibilidad de solicitar un cambio de contraseña. Éste enviará un email al usuario, con un enlace de uno sólo uso que permita crear una nueva contraseña.
 - *Alcanzable:* Existirá una pantalla de establecimiento de contraseña, cuyo uso está restringido al acceso mediante enlaces generados por BreakBrain para los usuarios que lo soliciten. La base de datos mantendrá un hash por usuario, que será el utilizado para identificar un cambio de contraseña, generado cada vez que se utilice el anterior, para asegurar un único uso.
 - *Relevante:* Se trata de un requisito opcional, pero indispensable en caso de olvido de la contraseña.

- RF-16**
- Descripción: Los usuarios podrán seleccionar qué parámetros cerebrales desean ejercitarse y monitorizar.
 - Importancia: Condicional
 - Validez:
 - *Medible:* El perfil de usuario permitirá seleccionar las capacidades mentales que el usuario desee.
 - *Alcanzable:* La persistencia mantendrá un perfil cerebral con las capacidades mentales que deben ser consideradas por los algoritmos de recomendación y demás componentes del proyecto. Sólo los parámetros cerebrales que el usuario desee trabajar deberán ser tenidos en cuenta.
 - *Relevante:* Este requisito es importante para personalizar el entrenamiento cerebral. Si no se satisface, los usuarios no podrán priorizar las capacidades mentales.
- RF-17**
- Descripción: Los usuarios podrán editar su información personal.
 - Importancia: Opcional
 - Validez:
 - *Medible:* El perfil le permitirá al usuario modificar su información personal.
 - *Alcanzable:* BreakBrain permitirá modificar todos los datos personales, menos el email, puesto que es el identificador de usuario y su autenticidad fue comprobada durante el proceso de registro.
 - *Relevante:* Es un requisito interesante para mejorar la experiencia de usuario.
- RF-18**
- Descripción: Los usuarios podrán seleccionar una imagen de avatar para que les represente en la red.
 - Importancia: Opcional
 - Validez:
 - *Medible:* El perfil de usuario permitirá visualizar y modificar la imagen representativa del mismo.
 - *Alcanzable:* BreakBrain permitirá modificar los datos personales, incluida la imagen de avatar.
 - *Relevante:* Este requisito es poco importante, pero ayudará al usuario a mantener una imagen deseada frente al resto de usuarios, por lo que mejorará la experiencia de uso.
- RF-19**
- Descripción: Los usuarios podrán comenzar a seguir a otros usuarios.
 - Importancia: Esencial
 - Validez:
 - *Medible:* La pantalla de exploración de usuarios permitirá comenzar a seguir a cualquier usuario.

- *Alcanzable*: La persistencia mantendrá las estructuras de datos necesarias para almacenar qué usuarios siguen a qué otros usuarios.
- *Relevante*: Este requisito es fundamental para mantener la componente social del proyecto.

RF-20 ■ Descripción: Los usuarios podrán dejar de seguir a los usuarios a los que siguen.
■ Importancia: Esencial
■ Validez:

- *Medible*: La pantalla de exploración de usuarios permitirá dejar de seguir a cualquier usuario.
- *Alcanzable*: La persistencia mantendrá las estructuras de datos necesarias para almacenar qué usuarios siguen a qué otros usuarios.
- *Relevante*: Este requisito es fundamental para mantener la componente social del proyecto.

RF-21 ■ Descripción: Los usuarios podrán consultar un resumen de su evolución en la página de inicio.
■ Importancia: Condicional
■ Validez:

- *Medible*: La pantalla principal mostrará un pequeño resumen de la evolución del perfil cerebral.
- *Alcanzable*: El sistema deberá dar acceso a sólo parte del perfil cerebral y su evolución. La página principal mostrará, entre otras cosas, esa información resumida.
- *Relevante*: La satisfacción de este requisito mejorará la experiencia de usuario, evitando que el mismo tenga que acceder al perfil completo para ver cómo va evolucionando.

RF-22 ■ Descripción: Los usuarios podrán consultar un listado de novedades sobre los usuarios a los que siguen en la página de inicio.
■ Importancia: Condicional
■ Validez:

- *Medible*: La página principal del sitio web mostrará las novedades de los usuarios a los que se sigue.
- *Alcanzable*: El perfil de cada usuario incluirá un pequeño listado de novedades. Los usuarios que sigan a uno dado, y sólo ellos, tendrán acceso a dicho listado.
- *Relevante*: La satisfacción de este requisito mejorará la experiencia de usuario, sobre todo en el aspecto social.

- RF-23** ■ Descripción: Los usuarios podrán ver su perfil cerebral completo, con el estado actual y la evolución semanal, mensual y anual.
- Importancia: Esencial
- Validez:
- *Medible:* La web contará con una página dedicada al perfil y evolución cerebral.
 - *Alcanzable:* El sistema manejará una estructura de datos compleja con toda la información relevante referida al cerebro. Esta información incluirá el estado actual y la evolución del mismo a lo largo del tiempo.
 - *Relevante:* Se trata de un requisito vital para alcanzar los objetivos del proyecto.
- RF-24** ■ Descripción: Los usuarios podrán hacer búsquedas de usuarios.
- Importancia: Esencial
- Validez:
- *Medible:* Existirá una página de exploración de usuarios que permitirá realizar búsquedas filtradas sobre todo el directorio de usuarios.
 - *Alcanzable:* Se desarrollará un algoritmo de búsqueda de usuarios, y será puesto a disposición de todos los usuarios mediante un sencillo formulario de búsqueda.
 - *Relevante:* Este requisito es de obligado cumplimiento para mantener un aspecto social completo en el proyecto, permitiendo que los usuarios puedan encontrar a otros usuarios y comenzar a seguirlos, dejar de seguirlos, retarles a juegos multijugador, etc.
- RF-25** ■ Descripción: Los usuarios recibirán recomendaciones de juegos.
- Importancia: Condicional
- Validez:
- *Medible:* La web mostrará un apartado con juegos recomendados para el usuario actual.
 - *Alcanzable:* Se desarrollará un algoritmo de recomendación de juegos completo, basado en la evolución cerebral de los usuarios y su afinidad a cada tipo de juegos.
 - *Relevante:* Se trata de un requerimiento importante para mejorar la experiencia de usuario y asegurar una mejora cerebral equilibrada, reforzando las capacidades en los que el usuario flaquee más.
- RF-26** ■ Descripción: Los usuarios recibirán recomendaciones de usuarios a los que seguir.
- Importancia: Condicional.

- Validez:
 - *Medible:* La web mostrará un apartado con usuarios recomendados para el usuario actual.
 - *Alcanzable:* Se desarrollará un algoritmo de recomendación de juegos completo, basado en la evolución cerebral de los usuarios y su afinidad a cada tipo de juegos.
 - *Relevante:* Es un requisito importante para mejorar la experiencia de usuario.

RF-27 ■ Descripción: Los usuarios podrán visualizar un listado completo de juegos disponibles, diferenciando los juegos multijugador de los juegos de un sólo jugador.

- Importancia: Esencial

- Validez:

- *Medible:* La web ofrecerá un listado con todos los juegos disponibles, permitiendo que el usuario ejecute el que desee.
- *Alcanzable:* El sistema mantendrá un catálogo categorizado con todos los juegos disponibles.
- *Relevante:* Este requisito es vital para satisfacer los objetivos de BreakBrain.

RF-28 ■ Descripción: Los usuarios podrán retar a otros usuarios y ser retados para jugar a juegos multijugador.

- Importancia: Condicional

- Validez:

- *Medible:* La web ofrecerá un listado con todos los juegos disponibles, permitiendo que el usuario ejecute el que desee. En caso de ser un juego multijugador permitirá elegir a otro usuario contra el que jugar.
- *Alcanzable:* El sistema permitirá seleccionar un usuario rival para jugar a un juego multijugador. Se enviará un mensaje de alerta al mismo para avisarle del reto y permitirle aceptarlo o rechazarlo.
- *Relevante:* Este requisito es importante para satisfacer el objetivo social de BreakBrain.

4.3.3. Requisitos de evolución

Partiendo del análisis preliminar de requerimientos, los requisitos de evolución de los que parte el desarrollo del proyecto, basándose en el estándar IEEE 830, son los siguientes:

- EV-1** ■ Descripción: El sistema debe soportar la integración de juegos desarrollados por terceros.
- Importancia: Condicional
 - Validez:
 - *Medible:* Cualquier persona podrá crear juegos para la red social.

- *Alcanzable*: Existirá un cargador de juegos genérico, así como un pequeño framework de construcción de los mismos, para posibilitar la compatibilidad con el sistema de comunicación cliente-servidor del sistema.
- *Relevante*: Es un requisito interesante para facilitar la expansión de la red social, permitiendo tener contenidos nuevos de forma frecuente.

4.3.4. Requisitos de soporte

Partiendo del análisis preliminar de requerimientos, los requisitos de soporte de los que parte el desarrollo del proyecto, basándose en el estándar IEEE 830, son los siguientes:

- SO-1**
- Descripción: El acceso a la web debe realizarse desde un navegador moderno.
 - Importancia: Esencial
 - Validez:
 - *Medible*: El rendimiento en navegadores obsoletos dejará mucho que desear, siendo estos incluso incompatibles con ciertos componentes de la última especificación del estándar HTML5, como el canvas.
 - *Alcanzable*: Se utilizarán elementos compatibles con Firefox 9+, Chrome/- Chromium 15+, Safari 5+ y Opera 12+.
 - *Relevante*: Este requisito es de vital cumplimiento para que la experiencia de usuario sea satisfactoria.

4.3.5. Requisitos de calidad

Partiendo del análisis preliminar de requerimientos, los requisitos de soporte de los que parte el desarrollo del proyecto, basándose en el estándar IEEE 830, son los siguientes:

- CA-1**
- Descripción: Los usuarios no autenticados no podrán acceder al contenido del sitio web.
 - Importancia: Esencial
 - Validez:
 - *Medible*: Cuando un visitante no autenticado acceda a cualquier página de la web, éste será redireccionado a la página de inicio (login).
 - *Alcanzable*: La descarga de cualquier contenido de la web será posible únicamente desde una cuenta de usuario válida, por lo que ningún visitante no autenticado tendrá acceso a ninguna información.
 - *Relevante*: Este requisito es importante para mantener la privacidad dentro de la red social.
- CA-2**
- Descripción: La base de datos almacenará la información de los usuarios de forma segura.
 - Importancia: Esencial

- Validez:
 - *Medible:* Toda la información vital de los usuarios será almacenada de forma encriptada.
 - *Alcanzable:* Los datos sensibles de los usuarios se almacenarán de forma encriptada.
 - *Relevante:* Requisito importante para mantener la seguridad en el almacenamiento de la información.
- CA-3** ■ Descripción: La comunicación entre el cliente y el servidor será segura.
 - Importancia: Esencial
 - Validez:
 - *Medible:* Toda la información vital de los usuarios será enviada al servidor de forma encriptada.
 - *Alcanzable:* Los datos sensibles de los usuarios se encriptarán localmente antes de ser enviados al servidor.
 - *Relevante:* Requisito muy importante para garantizar la seguridad de la transmisión de información.
- CA-4** ■ Descripción: La información de sesión local se almacenará de forma segura.
 - Importancia: Esencial
 - Validez:
 - *Medible:* La información sensible que se permanezca guardada localmente será ilegible.
 - *Alcanzable:* Los datos sensibles serán encriptados antes de almacenarse de forma local.
 - *Relevante:* Requisito importante para mantener la seguridad del software.
- CA-5** ■ Descripción: El servidor debe ofrecer tiempos de respuesta muy bajos, para evitar retardos en el uso de la web o los juegos.
 - Importancia: Esencial
 - Validez:
 - *Medible:* La utilización de la web debe ser fluida.
 - *Alcanzable:* Se desarrollará un servidor con buen rendimiento y tiempos de respuesta bajos. Se utilizará MongoDB como base de datos, para asegurar un rendimiento óptimo en la recuperación de información.
 - *Relevante:* Cumplir este requisito será esencial para que la experiencia de usuario sea buena.
- CA-6** ■ Descripción: Los juegos deben ofrecer una interacción sencilla.
 - Importancia: Condicional
 - Validez:

- *Medible*: Cualquier persona no debería tener dificultad para jugar a los juegos de la plataforma.
- *Alcanzable*: Los controles de los juegos serán preferiblemente mediante el uso del ratón o de un número bajo de teclas. Serán intuitivos, como el uso de las flechas del teclado.
- *Relevante*: Requisito altamente recomendable para asegurar una buena jugabilidad.

- CA-7**
- Descripción: El sistema debe ser escalable, es decir, debe permitir aumentar la capacidad de trabajo sin comprometer su funcionamiento y calidad normales.
 - Importancia: Esencial
 - Validez:
 - *Medible*: Podrá conectarse un gran número de usuarios de forma simultánea y realizar varias peticiones de información, juegos, etc. sin que el sistema sufra pérdida de rendimiento.
 - *Alcanzable*: Se empleará NodeJS para la construcción del servidor, así como MongoDB para la persistencia.
 - *Relevante*: Este requisito es vital para el proyecto en construcción, dado que una red social puede sufrir un gran número de conexiones simultáneas.

4.4. Diagrama de despliegue

En base a la caracterización del sistema producida por el análisis de requisitos desarrollado a lo largo de la sección 4.3 se ha elaborado el diagrama de despliegue de la figura 4.2. Nótese que se trata de una arquitectura cliente-servidor típica.

Como puede apreciarse en la figura 4.2 se emplean servicios externos que satisfacen las necesidades del sistema:

- Por un lado se hace uso del servicio **MongoLab** [Monb], que proporciona un Sistema Gestor de Base de Datos (SGBD) funcionando como lo que se conoce como Software As A Service (SAAS), con MongoDB [Mona] como base de datos. La persistencia de la base de datos tiene lugar en los servicios de almacenamiento web de Amazon [Ama].
- Por otro lado, el servidor del sistema es alojado en **Nodester** [Nodb], una plataforma/hosting de aplicaciones web construidas con NodeJS [Noda]. Este servicio pertenece a lo que se conoce como Platform As A Service (PAAS), y es ofrecido por AppFog [App].

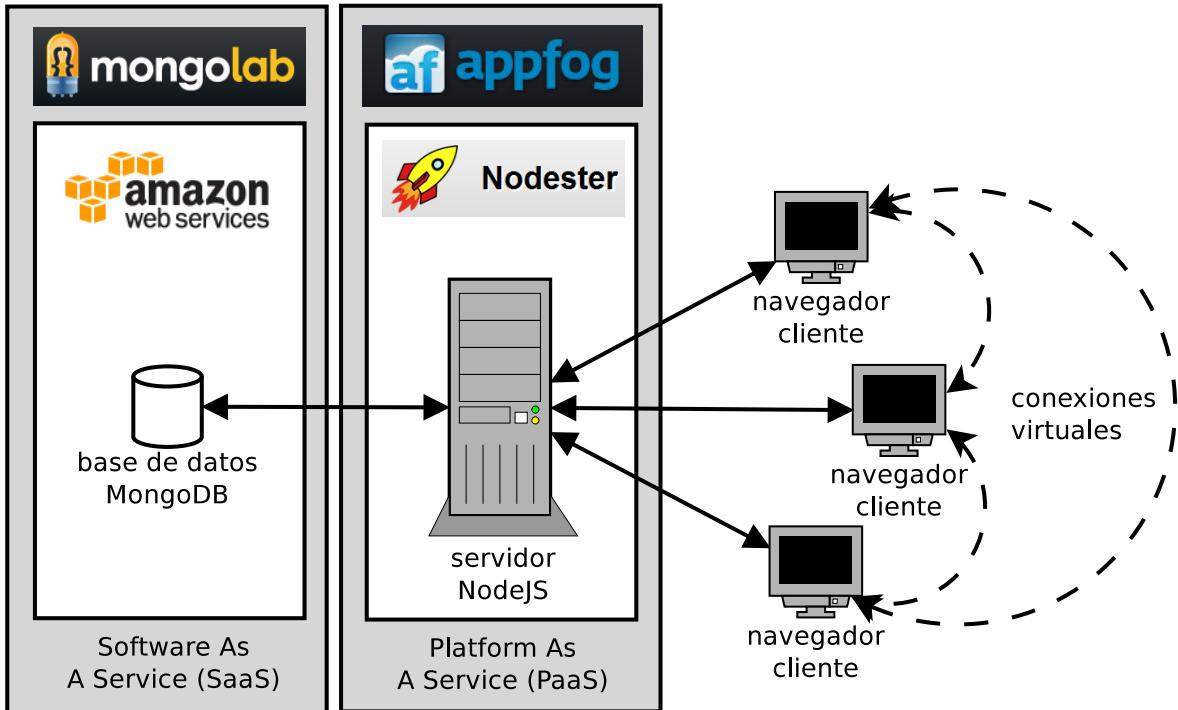


Figura 4.2: Diagrama de despliegue del sistema completo

4.5. Análisis de casos de uso

En esta sección se detalla el proceso seguido para la generación del artefacto correspondiente al análisis de casos de uso del sistema. En primer lugar se realiza una identificación de los actores que intervienen, para después dar paso a la identificación de los casos de uso. Por último éstos son especificados mediante una descripción, el listado de requisitos asociados, los actores que intervienen, las precondiciones y postcondiciones, los flujos de eventos y los posibles casos de uso asociados.

4.5.1. Identificación de actores

Considerando el sistema en construcción como un único bloque, los actores que interactúan con él son los visitantes (no autenticados), los usuarios del mismo (personas autenticadas) y el sistema gestor de base de datos, al tratarse de un servicio externo. Así pues, los actores que interactuarán con los casos de uso son los siguientes:

- Visitante
- Usuario
- Sistema Gestor de Base de Datos (SGBD)

4.5.2. Identificación de casos de uso

En base al estudio de requisitos realizado en la sección 4.3, los casos de uso que se han identificado y pasarán a ser detallados como base de las etapas posteriores del desarrollo son los mostrados en la figura 4.3.

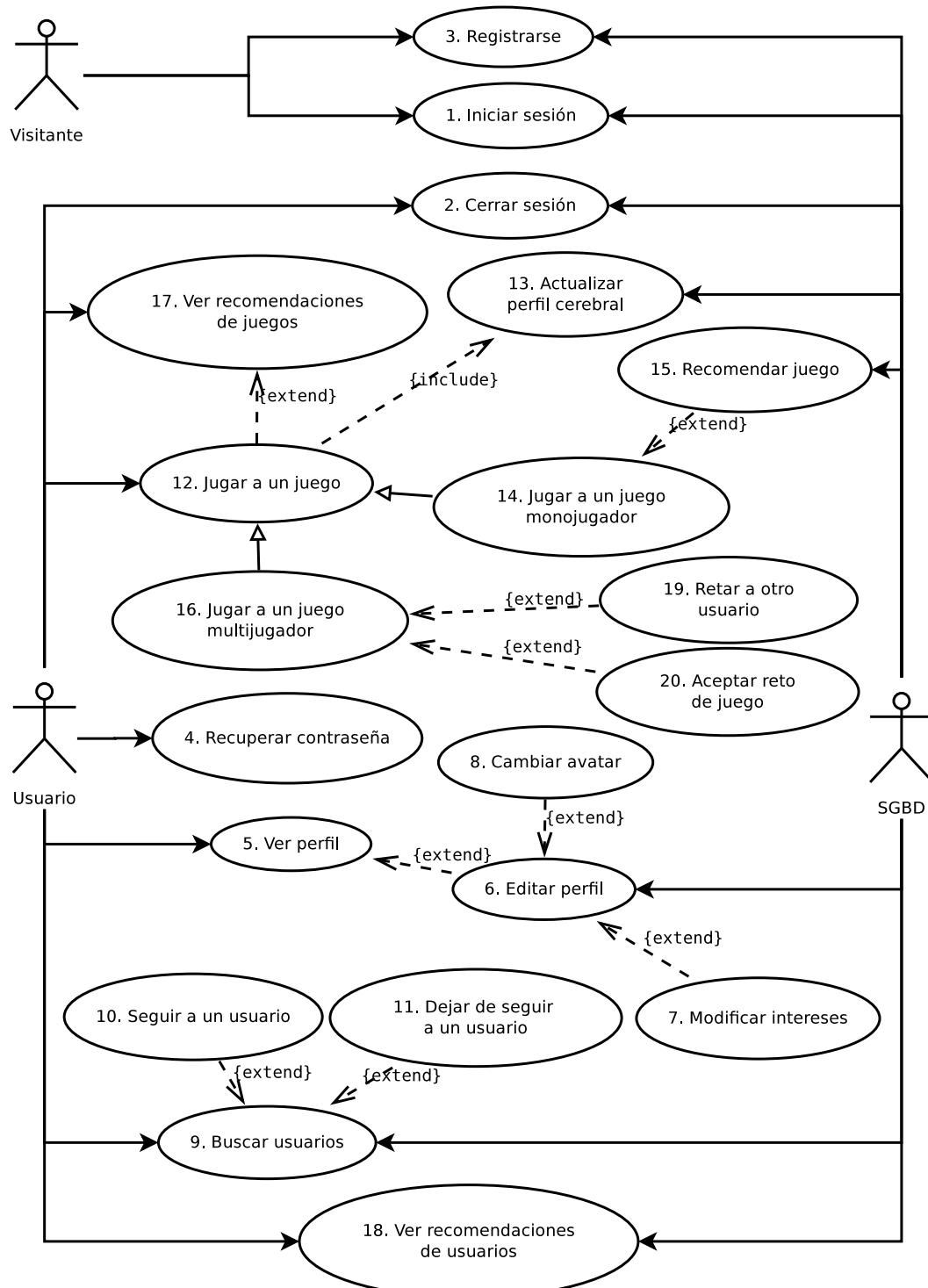


Figura 4.3: Diagrama de casos de uso

4.5.3. Especificación de casos de uso

A continuación se detallan todos los casos de uso identificados en la etapa anterior. Los más identificativos se acompañan, además, de un diagrama de secuencia para facilitar la comprensión del funcionamiento interno del sistema.

- UC-1**
- **Caso de uso:** Iniciar sesión
 - **Requisito funcional asociado:** RF-11
 - **Actores**
 - Visitante
 - SGBD
 - **Descripción:** Un visitante de la web inicia sesión.
 - **Precondiciones**
 - La persona no está autenticada.
 - **Postcondiciones**
 - La persona queda autenticada (es un usuario de la red social), y tiene acceso al contenido de la web.
 - **Flujo normal:** autenticación exitosa (figura 4.4)
 1. El usuario rellena el formulario de login (email + contraseña) de la web.
 2. La web realiza la petición de envío al módulo cliente de websockets.
 3. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
 4. El módulo servidor de websockets analiza la petición y se la pasa a la lógica del servidor.
 5. El módulo de lógica de servidor transforma la petición y solicita al módulo de persistencia la comprobación de los datos.
 6. El módulo de persistencia solicita al SGBD el usuario asociado al par email/contraseña.
 7. El SGBD devuelve el objeto que representa al usuario al módulo de persistencia.
 8. El módulo de persistencia transforma el objeto de usuario a la representación de dominio de un usuario y devuelve el resultado a la lógica del sistema.
 9. La lógica comprueba que el usuario es correcto y solicita su envío hacia el cliente al módulo servidor de websockets.
 10. El módulo servidor de websockets envía el objeto de usuario al módulo cliente de websockets.
 11. El módulo cliente de websockets pasa el objeto a la web.
 12. La web se actualiza con la información básica del usuario y muestra el contenido interno de la red social.

■ Flujo alternativo 1: contraseña incorrecta o usuario inexistente

1. El usuario rellena el formulario de login (email + contraseña) de la web.
2. La web realiza la petición de envío al módulo cliente de websockets.
3. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
4. El módulo servidor de websockets analiza la petición y se la pasa a la lógica del servidor.
5. El módulo de lógica de servidor transforma la petición y solicita al módulo de persistencia la comprobación de los datos.
6. El módulo de persistencia solicita al SGBD el usuario asociado al par email/contraseña.
7. El SGBD devuelve un error de correspondencia email/contraseña o de contraseña incorrecta.
8. El módulo de persistencia transforma el error a la representación de dominio y lo devuelve al módulo de lógica del sistema.
9. La lógica solicita el envío del error hacia el cliente al módulo servidor de websockets.
10. El módulo servidor de websockets envía el error al módulo cliente de websockets.
11. El módulo cliente de websockets pasa el error a la web.
12. La web muestra un error de “contraseña incorrecta” o “usuario inexistente”, según proceda.

■ Includes

- Ninguno

■ Extensiones

- Ninguna

■ Diagrama de interacción: Figura 4.4**UC-2**

- Caso de uso:** Cerrar sesión
- Requisito funcional asociado:** RF-12
- Actores**
 - Usuario
 - SGBD
- Descripción:** Un usuario autenticado cierra su sesión.
- Precondiciones**
 - La persona debe tener una sesión iniciada.
- Postcondiciones**

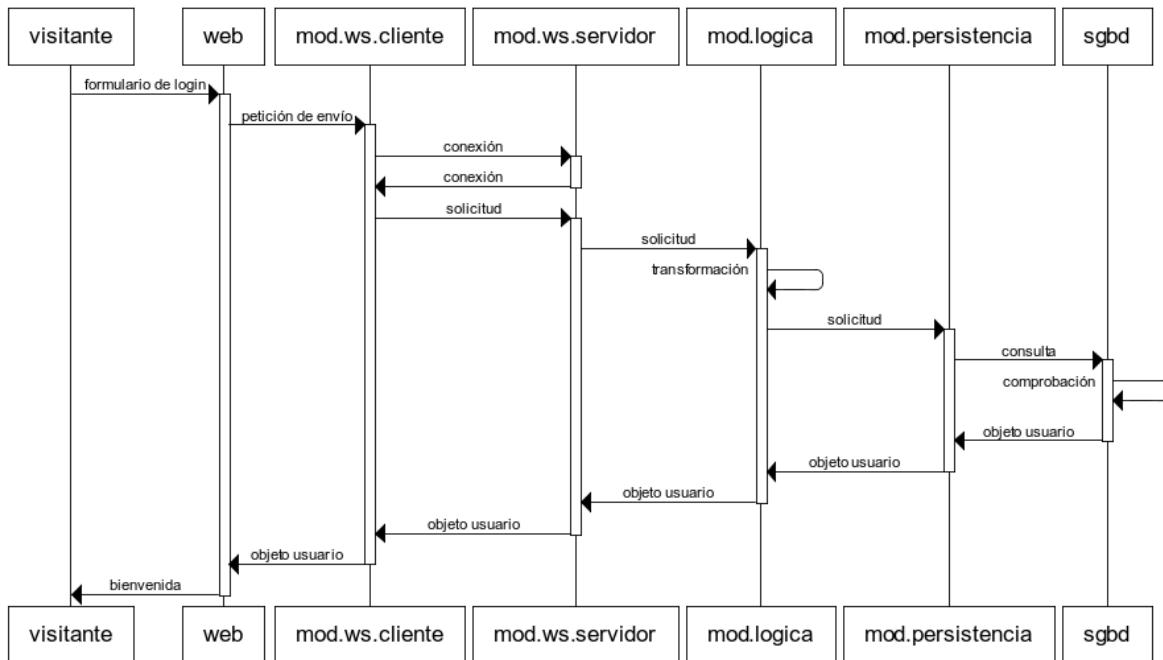


Figura 4.4: Diagrama de interacción del caso de uso UC-1 (flujo normal)

- El acceso al contenido de la web se cierra y el usuario pasa a ser considerado como un visitante.

■ Flujo normal

1. El usuario pulsa el botón de cerrar sesión.
2. El navegador elimina la información de sesión, y muestra la pantalla de inicio de sesión, evitando el acceso al contenido de la web.

■ Includes

- Ninguno

■ Extensiones

- Ninguna

UC-3

■ Caso de uso: Registrarse

■ Requisito funcional asociado: RF-13

■ Actores

- Visitante
- SGBD

■ Descripción: Un visitante de la web se registra para obtener una cuenta de usuario.

■ Precondiciones

- La persona no está autenticada.

- **Postcondiciones**

- Existe una nueva cuenta de usuario.

- **Flujo normal:** registro exitoso

1. El usuario rellena el formulario de registro.
2. La web realiza la petición de envío al módulo cliente de websockets.
3. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
4. El módulo servidor de websockets analiza la petición y se la pasa a la lógica del servidor.
5. El módulo de lógica de servidor transforma la petición y solicita al módulo de persistencia la comprobación de los datos.
6. El módulo de persistencia solicita al SGBD la creación del usuario.
7. El SGBD devuelve el objeto que representa al usuario al módulo de persistencia.
8. El módulo de persistencia transforma el objeto de usuario a la representación de dominio de un usuario y devuelve el resultado a la lógica del sistema.
9. La lógica envía una petición de activación al módulo de email.
10. El módulo de email envía un correo electrónico de activación al email proporcionado por el usuario.
11. El usuario utiliza el link de activación.
12. La web solicita el envío de activación al módulo cliente de websockets.
13. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
14. El módulo servidor de websockets pasa la información de activación a la lógica del servidor.
15. El módulo de lógica pasa la información de activación al módulo de persistencia.
16. El módulo de persistencia ejecuta la consulta de activación sobre el SGBD.
17. El SGBD activa la cuenta y devuelve una confirmación.
18. El módulo de persistencia devuelve la confirmación al módulo de lógica.
19. El módulo de lógica pasa la confirmación al módulo servidor de websockets.
20. El módulo servidor de websockets envía al módulo cliente de websockets la confirmación de activación.
21. El módulo cliente de websockets pasa la confirmación a la web.
22. La web muestra un mensaje de activación satisfactoria.

- **Flujo alternativo 1:** email de registro ya en uso

1. El usuario rellena el formulario de registro.

2. La web realiza la petición de envío al módulo cliente de websockets.
3. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
4. El módulo servidor de websockets analiza la petición y se la pasa a la lógica del servidor.
5. El módulo de lógica de servidor transforma la petición y solicita al módulo de persistencia la comprobación de los datos.
6. El módulo de persistencia solicita al SGBD la creación del usuario.
7. El SGBD devuelve un error de email en uso.
8. El módulo de persistencia devuelve el error a la lógica del sistema.
9. El módulo de lógica pasa el error al módulo servidor de websockets.
10. El módulo servidor de websockets envía al módulo cliente de websockets el error de email en uso.
11. El módulo cliente de websockets pasa la el error a la web.
12. La web muestra un mensaje de error por email en uso.

■ **Includes**

- Ninguno

■ **Extensiones**

- Ninguna

■ **Diagrama de interacción:** Figura 4.5

UC-4

- **Caso de uso:** Recuperar contraseña
- **Requisito funcional asociado:** RF-15
- **Actores**
 - Usuario
 - SGBD
- **Descripción:** El usuario que olvidó su contraseña genera una nueva, sustituyendo ésta a la anterior.
- **Precondiciones**
 - La persona no debe tener una sesión iniciada.
- **Postcondiciones**
 - La contraseña anterior del usuario queda sustituida por la nueva.
- **Flujo normal**
 1. El usuario hace clic sobre el enlace de recuperación de contraseña.
 2. La web le ofrece un diálogo para que introduzca el email con el que se registró.

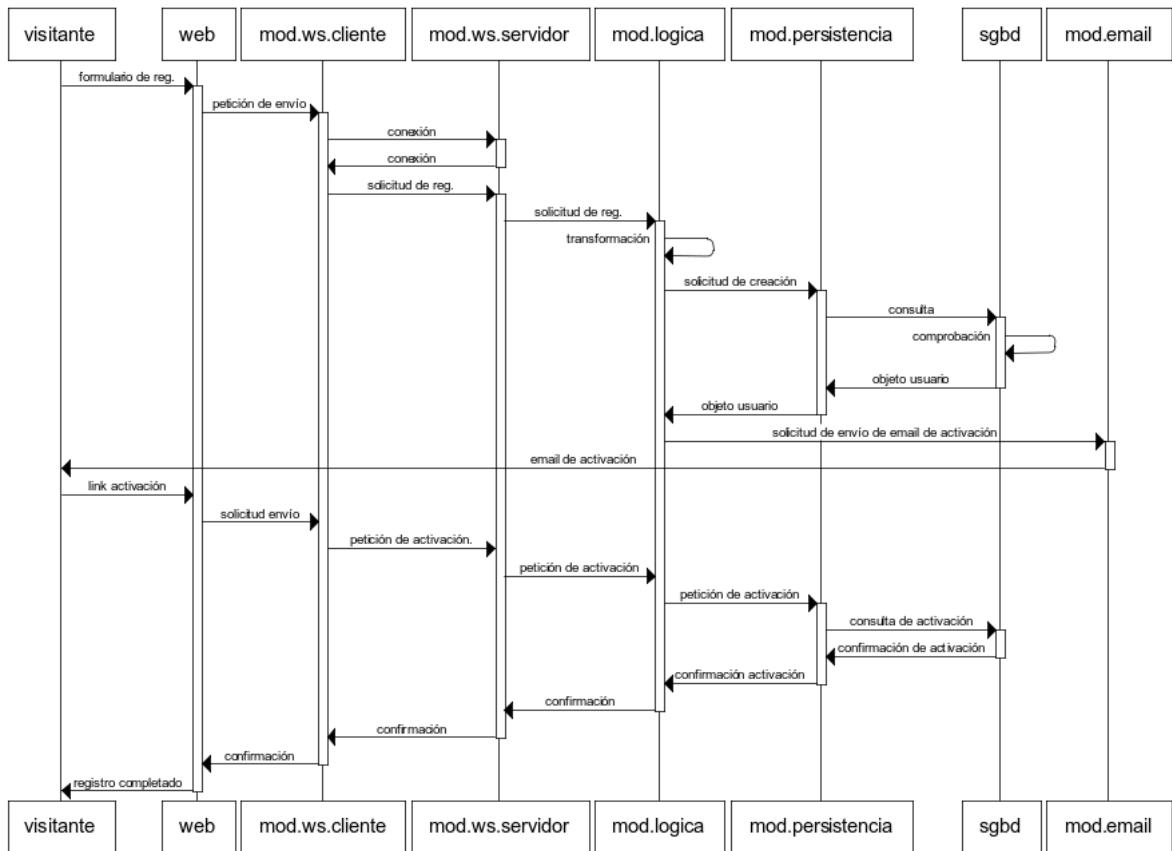


Figura 4.5: Diagrama de interacción del caso de uso UC-3 (flujo normal)

3. El usuario escribe su email y acepta el envío de un enlace para generar la contraseña.
4. La web envía la solicitud al módulo de websockets cliente.
5. El módulo cliente de websockets envía la solicitud al módulo servidor de websockets.
6. El módulo servidor de websockets recibe la solicitud y la pasa al módulo de lógica del servidor.
7. El módulo de lógica pasa la solicitud al módulo de persistencia.
8. El módulo de persistencia pide el hash del objeto usuario asociado al email de la petición al SGBD.
9. El SGBD devuelve el hash del usuario asociado al email.
10. El módulo de persistencia devuelve el hash al módulo de lógica.
11. El módulo de lógica genera un link y solicita al módulo de email el envío del mismo.
12. El módulo de email envía el link a la dirección proporcionada por el usuario.
13. El usuario sigue el enlace recibido por email y accede a la pantalla de generación de contraseña. Crea una nueva contraseña y acepta su envío.

14. La web envía los nuevos datos al módulo cliente de websockets.
15. El módulo cliente de websockets recibe la información y envía la solicitud de cambio de contraseña al módulo servidor de websockets.
16. El módulo servidor de websockets pasa la petición al módulo de lógica del servidor.
17. El módulo de lógica solicita la sobreescritura de la contraseña antigua al módulo de persistencia, y genera un hash nuevo para sobreescribir al antiguo y evitar que el link de regeneración de contraseña vuelva a usarse.
18. El módulo de persistencia realiza las consultas pertinentes sobre el SGBD.
19. El SGBD modifica los datos y devuelve una confirmación de que los cambios se han realizado satisfactoriamente.
20. El módulo de persistencia devuelve la confirmación al módulo de lógica.
21. El módulo de lógica solicita el envío de la confirmación al módulo servidor de websockets.
22. El módulo servidor de websockets envía la confirmación al módulo cliente de websockets.
23. El módulo cliente de websockets pasa la confirmación a la web.
24. La web muestra un mensaje de éxito.

■ **Includes**

- Ninguno

■ **Extensiones**

- Ninguna

■ **Diagrama de interacción:** Figura 4.6

UC-5

- **Caso de uso:** Ver perfil
- **Requisitos funcionales asociados:** RF-16, RF-17
- **Actores**
- Usuario
 - SGBD
- **Descripción:** Un usuario solicita ver su perfil completo.
- **Precondiciones**
- El usuario está autenticado en el sistema
- **Postcondiciones**
- Ninguna
- **Flujo normal**
1. El usuario accede a la página de perfil.

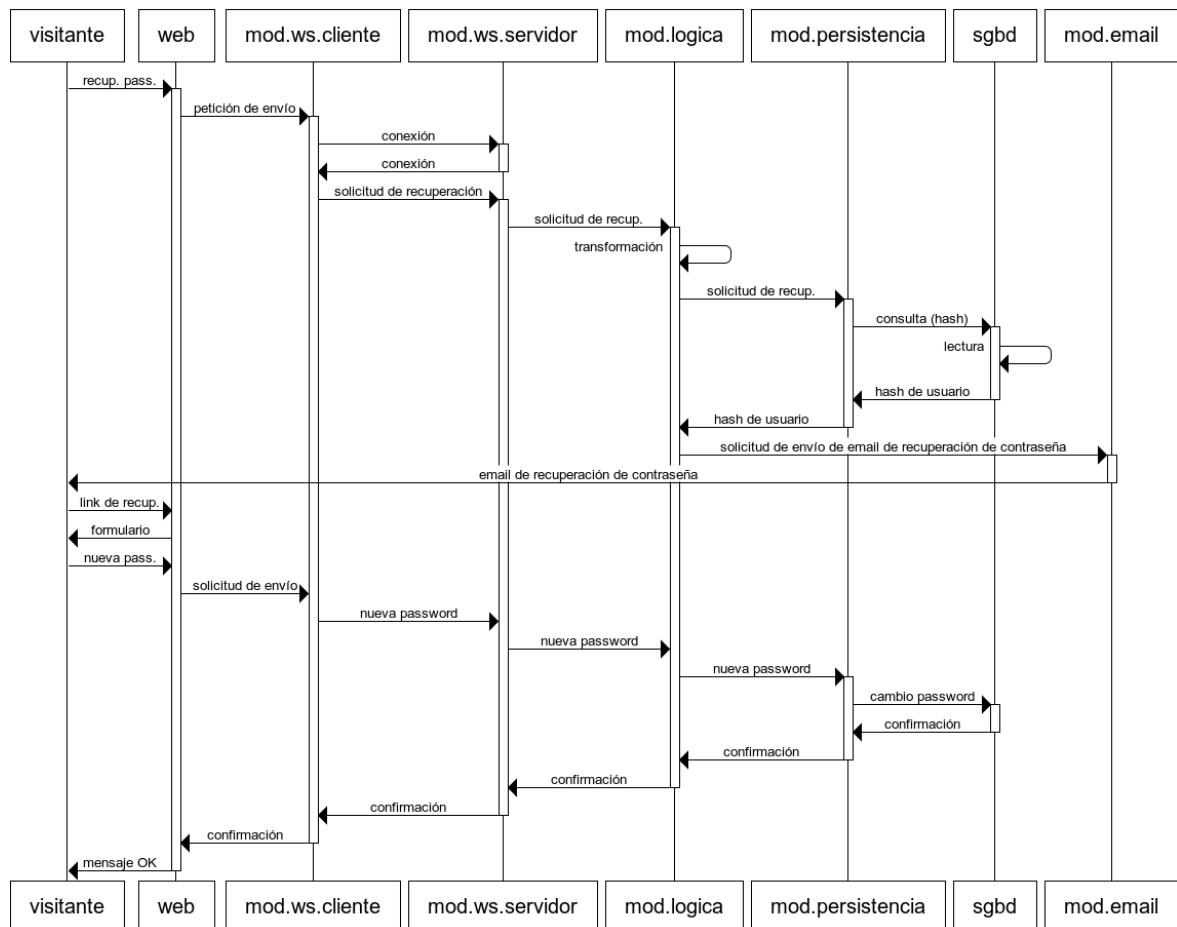


Figura 4.6: Diagrama de interacción del caso de uso UC-4 (flujo normal)

2. La web envía la petición de información al módulo cliente de websockets.
3. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
4. El módulo servidor de websockets pasa la solicitud del perfil al módulo de lógica.
5. El módulo de lógica del servidor pasa la solicitud al módulo de persistencia.
6. El módulo de persistencia realiza una consulta sobre SGBD para recuperar el perfil de usuario.
7. El SGBD recupera la información y se la envía de vuelta al módulo de persistencia.
8. El módulo de persistencia devuelve el objeto de perfil al módulo de lógica.
9. El módulo de lógica solicita el envío del perfil al módulo servidor de websockets.
10. El módulo servidor de websockets envía la información al módulo cliente de websockets.

11. El módulo cliente de websockets pasa toda la información de perfil a la web.
12. La web se actualiza con la información del perfil completo.

■ **Includes**

- Ninguno

■ **Extensiones**

- UC-6

UC-6 ■ **Caso de uso:** Editar perfil

■ **Requisitos funcionales asociados:** RF-17

■ **Actores**

- Usuario
- SGBD

■ **Descripción:**

■ **Precondiciones**

- El usuario está autenticado en el sistema.

■ **Postcondiciones**

- El perfil del usuario es modificado en la base de datos.

■ **Flujo normal**

1. El usuario accede a la página de perfil.
2. La web envía la petición de información al módulo cliente de websockets.
3. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
4. El módulo servidor de websockets pasa la solicitud del perfil al módulo de lógica.
5. El módulo de lógica del servidor pasa la solicitud al módulo de persistencia.
6. El módulo de persistencia realiza una consulta sobre SGBD para recuperar el perfil de usuario.
7. El SGBD recupera la información y se la envía de vuelta al módulo de persistencia.
8. El módulo de persistencia devuelve el objeto de perfil al módulo de lógica.
9. El módulo de lógica solicita el envío del perfil al módulo servidor de websockets.
10. El módulo servidor de websockets envía la información al módulo cliente de websockets.
11. El módulo cliente de websockets pasa toda la información de perfil a la web.
12. La web se actualiza con la información del perfil completo.

13. El usuario modifica la información personal y pulsa el botón de guardar cambios.
14. La web envía la solicitud de modificación al módulo cliente de websockets.
15. El módulo cliente de websockets envía la solicitud al módulo servidor de websockets.
16. El módulo servidor de websockets pasa la solicitud de modificación de perfil al módulo de lógica.
17. El módulo de lógica pasa la solicitud al módulo de persistencia.
18. El módulo de persistencia ejecuta las consultas de modificación sobre el SGBD.
19. El SGBD devuelve una confirmación.
20. El módulo de persistencia devuelve la confirmación al módulo de lógica.
21. El módulo de lógica devuelve la confirmación al módulo servidor de websockets.
22. El módulo servidor de websockets envía la confirmación al módulo cliente de websockets.
23. El módulo cliente de websockets devuelve la confirmación a la web.
24. La web se actualiza mostrando un mensaje de éxito al guardar los cambios.

■ **Includes**

- Ninguno

■ **Extensiones**

- UC-8
- UC-7

- UC-7**
- **Caso de uso:** Modificar intereses
 - **Requisitos funcionales asociados:** RF-16
 - **Actores**
 - Usuario
 - SGBD
 - **Descripción:** El usuario modifica las habilidades mentales que quiere trabajar.
 - **Precondiciones**
 - El usuario debe estar autenticado en el sistema.
 - El usuario debe estar visualizando su perfil.
 - **Postcondiciones**
 - Los parámetros de entrenamiento cerebral (intereses mentales del usuario) se ven modificados de forma temporal (no permanente)
 - **Flujo normal**

1. El usuario hace clic sobre un botón para ver sus intereses actuales.
2. La web solicita la información de intereses al módulo cliente de websockets.
3. El módulo cliente de websockets envía la solicitud al módulo servidor de websockets.
4. El módulo servidor de websockets recibe la petición y se la pasa al módulo de lógica del servidor.
5. El módulo de lógica solicita la información al módulo de persistencia.
6. El módulo de persistencia ejecuta la consulta de recuperación de intereses sobre el SGBD.
7. El SGBD obtiene la información requerida y se la envía de vuelta al módulo de persistencia.
8. El módulo de persistencia pasa la información de intereses al módulo de lógica.
9. El módulo de lógica devuelve los datos al módulo servidor de websockets para que los envíe al cliente.
10. El módulo servidor de websockets envía de vuelta la información al módulo cliente de websockets.
11. El módulo cliente de websockets pasa la información a la web.
12. La web muestra los intereses del usuario.
13. El usuario modifica esos intereses.

■ **Includes**

- Ninguno

■ **Extensiones**

- Ninguna

UC-8

■ **Caso de uso:** Cambiar avatar

■ **Requisitos funcionales asociados:** RF-18

■ **Actores**

- Usuario

■ **Descripción:** El usuario modifica su avatar (imagen representativa).

■ **Precondiciones**

- El usuario debe estar autenticado.
- El usuario debe encontrarse en la página de perfil.

■ **Postcondiciones**

- El avatar de usuario es cambiado de forma temporal (no permanente).

■ **Flujo normal**

1. El usuario hace clic sobre un botón para “cambiar su avatar”.

2. La web le muestra un mensaje de diálogo para que seleccione una imagen nueva desde su ordenador.
3. El usuario selecciona una imagen y acepta.
4. La web recupera la imagen elegida y sustituye al avatar anterior.

- **Includes**

- Ninguno

- **Extensiones**

- Ninguna

- UC-9**
- **Caso de uso:** Buscar usuarios
 - **Requisitos funcionales asociados:** RF-24
 - **Actores**
 - Usuario
 - SGBD
 - **Descripción:** El usuario ejecuta una búsqueda sobre el sistema, que le devuelve un listado de usuarios coincidentes con dicha consulta.
 - **Precondiciones**
 - El usuario debe estar autenticado en la web.
 - El usuario se encuentra visualizando la página de exploración de usuarios.
 - **Postcondiciones**
 - Ninguna
 - **Flujo normal**
 1. El usuario rellena el formulario de búsqueda y hace clic sobre el botón de buscar.
 2. La web construye la estructura de datos adecuada para almacenar la información del formulario y solicita su envío al módulo cliente de websockets.
 3. El módulo cliente de websockets envía la información al módulo servidor de websockets.
 4. El módulo servidor de websockets pasa la información al módulo de lógica del servidor.
 5. El módulo de lógica pasa la información de búsqueda al módulo de persistencia.
 6. El módulo de persistencia ejecuta el algoritmo de búsqueda sobre el SGBD.
 7. El SGBD devuelve un listado de resultados coincidentes con la búsqueda.
 8. El módulo de persistencia devuelve los resultados al módulo de lógica del servidor.

9. El módulo de lógica solicita el envío de los resultados al módulo servidor de websockets.
10. El módulo servidor de websockets envía los resultados al módulo cliente de websockets.
11. El módulo cliente de websockets recibe los resultados y se los pasa a la web.
12. La web se actualiza y muestra los resultados de la búsqueda.

■ **Includes**

- Ninguno

■ **Extensiones**

- UC-10
- UC-11

UC-10 ■ **Caso de uso:** Seguir a un usuario

■ **Requisitos funcionales asociados:** RF-19

■ **Actores**

- Usuario
- SGBD

■ **Descripción:** El usuario actual se convierte en seguidor de otro usuario.

■ **Precondiciones**

- El usuario está autenticado en la web.
- El usuario se encuentra en la página de exploración de usuarios.
- El usuario ha realizado una búsqueda de usuarios cuyo resultado incluye al usuario objetivo.
- El usuario actual no es seguidor del usuario al que se va a empezar a seguir.

■ **Postcondiciones**

- El usuario actual es seguidor del otro usuario.

■ **Flujo normal**

1. El usuario hace clic sobre el resultado de búsqueda asociado al usuario objetivo.
2. La web muestra el perfil del usuario objetivo.
3. El usuario actual hace clic sobre el botón “seguir”.
4. La web envía la solicitud de seguimiento al módulo cliente de websockets.
5. El módulo cliente de websockets envía la solicitud de seguimiento al módulo servidor de websockets.
6. El módulo de lógica pasa la solicitud al módulo de persistencia.
7. El módulo de persistencia ejecuta la consulta de establecimiento de un nuevo seguidor sobre el SGBD.

8. El SGBD actualiza la base de datos y devuelve una confirmación.
9. El módulo de persistencia devuelve la confirmación al módulo de lógica del servidor.
10. El módulo de lógica solicita el envío de la confirmación al módulo servidor de websockets.
11. El módulo servidor de websockets envía la confirmación al módulo cliente de websockets.
12. El módulo cliente de websockets recibe la confirmación y se los pasa a la web.
13. La web se actualiza y muestra un mensaje de éxito.

■ **Includes**

- Ninguno

■ **Extensiones**

- Ninguna

UC-11

■ **Caso de uso:** Dejar de seguir a un usuario

■ **Requisitos funcionales asociados:** RF-20

■ **Actores**

- Usuario
- SGBD

■ **Descripción:** El usuario actual deja de ser seguidor de otro usuario.

■ **Precondiciones**

- El usuario está autenticado en la web.
- El usuario se encuentra en la página de exploración de usuarios.
- El usuario ha realizado una búsqueda de usuarios cuyo resultado incluye al usuario objetivo.
- El usuario actual es seguidor del usuario al que se va a dejar de seguir.

■ **Postcondiciones**

- El usuario actual no es seguidor del otro usuario.

■ **Flujo normal**

1. El usuario hace clic sobre el resultado de búsqueda asociado al usuario objetivo.
2. La web muestra el perfil del usuario objetivo.
3. El usuario actual hace clic sobre el botón “dejar de seguir”.
4. La web envía la solicitud de anulación de seguimiento al módulo cliente de websockets.

5. El módulo cliente de websockets envía la solicitud de anulación de seguimiento al módulo servidor de websockets.
6. El módulo de lógica pasa la solicitud al módulo de persistencia.
7. El módulo de persistencia ejecuta la consulta de eliminación de un seguidor sobre el SGBD.
8. El SGBD actualiza la base de datos y devuelve una confirmación.
9. El módulo de persistencia devuelve la confirmación al módulo de lógica del servidor.
10. El módulo de lógica solicita el envío de la confirmación al módulo servidor de websockets.
11. El módulo servidor de websockets envía la confirmación al módulo cliente de websockets.
12. El módulo cliente de websockets recibe la confirmación y se los pasa a la web.
13. La web se actualiza y muestra un mensaje de éxito.

■ **Includes**

- Ninguno

■ **Extensiones**

- Ninguna

UC-12

■ **Caso de uso:** Jugar a un juego

■ **Requisitos funcionales asociados:** RF-8, RF-27

■ **Actores**

- Usuario
- SGBD

■ **Descripción:** El usuario lanza un juego para jugar a él.

■ **Precondiciones**

- El usuario debe estar autenticado en el sistema.

■ **Postcondiciones**

- El perfil cerebral es actualizado con los resultados de la partida.

■ **Flujo normal**

1. El usuario accede a la página de juegos.
2. La web solicita la petición de los listados de juegos al módulo cliente de websockets.
3. El módulo cliente de websockets envía la solicitud al módulo servidor de websockets.
4. El módulo servidor de websockets pasa la solicitud al módulo de lógica.

5. El módulo de lógica realiza la petición al módulo de juegos.
6. El módulo de juegos devuelve un listado con todos los juegos disponibles.
7. El módulo de lógica solicita el envío del listado al módulo servidor de web-sockets.
8. El módulo servidor de websockets envía el listado de juegos al módulo cliente de websockets.
9. El módulo cliente de websockets devuelve el listado de juegos a la web.
10. La web carga la página solicitada con los juegos disponibles.
11. El usuario selecciona un juego al que jugar.
12. La web solicita el juego al módulo cliente de websockets.
13. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
14. El módulo servidor de websockets pasa la petición al módulo de lógica.
15. El módulo de lógica solicita el juego pedido al módulo de juegos.
16. El módulo de juegos devuelve el juego solicitado completo.
17. El módulo de lógica solicita el envío del juego al módulo servidor de websockets.
18. El módulo servidor de websockets envía el juego al módulo cliente de websockets.
19. El módulo cliente de websockets pasa el juego a la web.
20. La web carga el juego recibido y lo ejecuta.

- **Includes**

- UC-13

- **Extensiones**

- Ninguna

- **Diagrama de interacción:** Figura 4.7

UC-13

- **Caso de uso:** Actualizar perfil cerebral
- **Requisitos funcionales asociados:** RF-8, RF-27
- **Actores**
 - SGBD
- **Descripción:** Tras una partida, el perfil cerebral es actualizado.
- **Precondiciones**
 - El usuario está autenticado en el sistema.
 - El usuario acaba de terminar una partida.
- **Postcondiciones**
 - El perfil cerebral del usuario queda modificado.

■ Flujo normal

1. La web solicita el envío del resultado al módulo cliente de websockets.
2. El módulo cliente de websockets envía el resultado al módulo servidor de websockets.
3. El módulo servidor de websockets pasa el resultado al módulo de lógica del servidor.
4. El módulo de lógica solicita la actualización del perfil cerebral al módulo de persistencia.
5. El módulo de persistencia ejecuta las consultas de actualización sobre el SGBD.
6. El SGBD realiza la actualización del perfil cerebral y devuelve una confirmación al módulo de persistencia.
7. El módulo de persistencia devuelve la confirmación al módulo de lógica.
8. El módulo de lógica solicita el envío de la confirmación al módulo servidor de websockets.
9. El módulo servidor de websockets envía la confirmación al módulo cliente de websockets.
10. El módulo cliente de websockets pasa la confirmación a la web.
11. La web muestra una notificación confirmando que el perfil cerebral ha sido actualizado.

■ Includes

- Ninguno

■ Extensiones

- Ninguna

UC-14

- **Caso de uso:** Jugar a un juego monojugador
- **Requisitos funcionales asociados:** RF-8, RF-27
- **Actores**
 - Usuario
 - SGBD
- **Descripción:**
- **Precondiciones**
 - El usuario debe estar autenticado en el sistema
 - El usuario debe estar situado en la página de juegos, con el listado cargado.
- **Postcondiciones**
 - Ninguna
- **Flujo normal**

1. El usuario selecciona un juego monojugador al que jugar.
2. La web solicita el juego al módulo cliente de websockets.
3. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
4. El módulo servidor de websockets pasa la petición al módulo de lógica.
5. El módulo de lógica solicita el juego pedido al módulo de juegos.
6. El módulo de juegos devuelve el juego solicitado completo.
7. El módulo de lógica solicita el envío del juego al módulo servidor de websockets.
8. El módulo servidor de websockets envía el juego al módulo cliente de websockets.
9. El módulo cliente de websockets pasa el juego a la web.
10. La web carga el juego recibido y lo ejecuta.

■ Includes

- Ninguno

■ Extensiones

- UC-15

■ Generalización de: UC-12**UC-15****■ Caso de uso:** Recomendar juego**■ Requisitos funcionales asociados:** RF-8, RF-27, RF-25**■ Actores**

- Usuario
- SGBD

■ Descripción: El usuario recomienda un juego monojugador a otro usuario.**■ Precondiciones**

- El usuario está autenticado en el sistema.
- El usuario ha cargado un juego.

■ Postcondiciones

- El usuario destino de la recomendación recibe una notificación por email.

■ Flujo normal

1. El usuario pulsa el botón de “recomendar” junto a un juego cargado, y selecciona un usuario al que enviar dicha recomendación.
2. La web solicita el envío de la recomendación al módulo cliente de websockets.
3. El módulo cliente de websockets envía la petición de recomendación al módulo servidor de websockets.

4. El módulo servidor de websockets pasa la petición de recomendación al módulo de lógica.
5. El módulo de lógica solicita los datos de contacto del usuario destino de la recomendación al módulo de persistencia.
6. El módulo de persistencia realiza una consulta para recuperar la información del usuario destino de la recomendación al SGBD.
7. El SGBD recupera la información y la envía de vuelta al módulo de persistencia.
8. El módulo de persistencia pasa los datos de contacto al módulo de lógica.
9. El módulo de lógica solicita el envío de la recomendación al módulo de email.
10. El módulo de email envía la recomendación al usuario correspondiente, y devuelve una confirmación al módulo de lógica.
11. El módulo de lógica solicita el envío de la confirmación de vuelta al cliente al módulo servidor de websockets.
12. El módulo servidor de websockets envía la confirmación de recomendación al módulo cliente de websockets.
13. El módulo cliente de websockets pasa la confirmación a la web.
14. La web muestra una notificación informando de que la recomendación ha sido enviada correctamente.

■ **Includes**

- Ninguno

■ **Extensiones**

- Ninguna

UC-16

■ **Caso de uso:** Jugar a un juego multijugador

■ **Requisitos funcionales asociados:** RF-8, RF-27

■ **Actores**

- Usuario
- SGBD

■ **Descripción:**

■ **Precondiciones**

- El usuario debe estar autenticado en el sistema.
- El usuario debe estar situado en la página de juegos, con el listado cargado.

■ **Postcondiciones**

- El perfil cerebral de ambos jugadores se ve modificado.

■ **Flujo normal**

1. El usuario selecciona un juego multijugador al que jugar y un usuario contra el que jugar.
2. La web solicita el juego y el envío de la invitación al módulo cliente de websockets.
3. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
4. El módulo servidor de websockets pasa la petición al módulo de lógica.
5. El módulo de lógica solicita el envío de la invitación al módulo servidor de websockets.
6. El módulo servidor de websockets envía la invitación al módulo cliente de websockets del otro usuario (el retado).
7. El módulo cliente de websockets del usuario retado pasa la invitación a la web.
8. La web muestra la invitación y permite aceptarla o rechazarla.
9. El usuario retado acepta la invitación.
10. La web envía el mensaje de aceptación al módulo cliente de websockets.
11. El módulo cliente de websockets del usuario retado envía el mensaje de aceptación al módulo servidor de websockets.
12. El módulo servidor de websockets pasa el mensaje de aceptación al módulo de lógica.
13. El módulo de lógica solicita el juego pedido al módulo de juegos.
14. El módulo de juegos devuelve el juego solicitado completo.
15. El módulo de lógica solicita el envío del juego al módulo servidor de websockets.
16. El módulo servidor de websockets envía el juego al módulo cliente de websockets de cada usuario.
17. El módulo cliente de websockets pasa el juego a la web.
18. La web carga el juego recibido y lo ejecuta.

■ **Flujo alternativo 1:** El usuario retado rechaza la invitación a jugar

1. El usuario selecciona un juego multijugador al que jugar y un usuario contra el que jugar.
2. La web solicita el juego y el envío de la invitación al módulo cliente de websockets.
3. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
4. El módulo servidor de websockets pasa la petición al módulo de lógica.
5. El módulo de lógica solicita el envío de la invitación al módulo servidor de websockets.
6. El módulo servidor de websockets envía la invitación al módulo cliente de

websockets del otro usuario (el retado).

7. El módulo cliente de websockets del usuario retado pasa la invitación a la web.
8. La web muestra la invitación y permite aceptarla o rechazarla.
9. El usuario retado rechaza la invitación.
10. La web envía el mensaje de rechazo al módulo cliente de websockets.
11. El módulo cliente de websockets del usuario retado envía el mensaje de rechazo al módulo servidor de websockets.
12. El módulo servidor de websockets pasa el mensaje de rechazo al módulo de lógica.
13. El módulo de lógica solicita el envío de rechazo al módulo servidor de websockets.
14. El módulo servidor de websockets envía el rechazo al módulo cliente de websockets del usuario original.
15. El módulo cliente de websockets pasa el mensaje de rechazo a la web.
16. La web notifica el rechazo del reto.

■ **Includes**

- Ninguno

■ **Extensiones**

- UC-18
- UC-19

■ **Generalización de:** UC-12

UC-17

- **Caso de uso:** Ver recomendaciones de juegos
- **Requisitos funcionales asociados:** RF-8, RF-27, RF-25
- **Actores**
 - Usuario
 - SGBD
- **Descripción:** El usuario recibe recomendaciones automatizadas de juegos por parte del sistema de recomendación.
- **Precondiciones**
 - El usuario está autenticado en el sistema.
- **Postcondiciones**
 - Ninguna
- **Flujo normal**
 1. El usuario accede a la página de juegos.

2. La web solicita la recomendación de juegos al módulo cliente de websockets.
3. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
4. El módulo servidor de websockets pasa la petición de recomendación al módulo de lógica.
5. El módulo de lógica solicita el listado de juegos al módulo de juegos.
6. El módulo de juegos devuelve al módulo de lógica el listado de juegos disponibles.
7. El módulo de lógica solicita la información completa del usuario al módulo de persistencia.
8. El módulo de persistencia lanza una consulta al SGBD para recuperar la información del usuario.
9. El SGBD devuelve la información completa del usuario al módulo de persistencia.
10. El módulo de persistencia pasa los datos del usuario al módulo de lógica.
11. El módulo de lógica solicita la recomendación de juegos al módulo de recomendación, adjuntándole el listado de juegos y la información completa del usuario.
12. El módulo de recomendación devuelve el resultado del algoritmo de recomendación de juegos al módulo de lógica.
13. El módulo de lógica solicita el envío de la recomendación al módulo servidor de websockets.
14. El módulo servidor de websockets envía la recomendación al módulo cliente de websockets.
15. El módulo cliente de websockets pasa la recomendación a la web.
16. La web muestra los resultados de la recomendación en la página de juegos.

■ **Includes**

- Ninguno

■ **Extensiones**

- Ninguna

UC-18

- **Caso de uso:** Ver recomendaciones de usuarios
- **Requisitos funcionales asociados:** RF-26
- **Actores**
 - Usuario
 - SGBD
- **Descripción:** El usuario recibe recomendaciones automatizadas de usuarios si-

milares a él por parte del sistema de recomendación.

■ **Precondiciones**

- El usuario está autenticado en el sistema.

■ **Postcondiciones**

- Ninguna

■ **Flujo normal**

1. El usuario accede a la página de exploración de usuarios.
2. La web solicita la recomendación de usuarios al módulo cliente de websockets.
3. El módulo cliente de websockets envía la petición al módulo servidor de websockets.
4. El módulo servidor de websockets pasa la petición de recomendación al módulo de lógica.
5. El módulo de lógica solicita el listado de usuarios al módulo de persistencia.
6. El módulo de persistencia lanza una consulta de usuarios al SGBD.
7. El SGBD devuelve un listado acotado de usuarios.
8. El módulo de persistencia devuelve al módulo de lógica el listado de usuarios.
9. El módulo de lógica solicita la información completa del usuario al módulo de persistencia.
10. El módulo de persistencia lanza una consulta al SGBD para recuperar la información del usuario.
11. El SGBD devuelve la información completa del usuario al módulo de persistencia.
12. El módulo de persistencia pasa los datos del usuario al módulo de lógica.
13. El módulo de lógica solicita la recomendación de usuarios al módulo de recomendación, adjuntándole el listado de usuarios y la información completa del usuario.
14. El módulo de recomendación devuelve el resultado del algoritmo de recomendación de usuarios al módulo de lógica.
15. El módulo de lógica solicita el envío de la recomendación al módulo servidor de websockets.
16. El módulo servidor de websockets envía la recomendación al módulo cliente de websockets.
17. El módulo cliente de websockets pasa la recomendación a la web.
18. La web muestra los resultados de la recomendación en la página de usuarios.

■ **Includes**

- Ninguno

- **Extensiones**

- Ninguna

4.6. Diseño

Dada la naturaleza de la tecnología empleada para el desarrollo del sistema de BreakBrain, la implementación se lleva a cabo de forma modular sobre el núcleo que supone NodeJS —en el lado del servidor—. Tanto a ese nivel como en el lado del cliente, se sigue el patrón MVC para la construcción del software.

En cuanto al servidor, las tres capas MVC son las siguientes:

- **Servidor web / Servidor de websockets:** Se encarga de la comunicación con el sistema cliente. Actúa como intermediario entre éste y el núcleo del servidor, en el que se implementa la lógica principal del sistema.
- **Núcleo del servidor:** Se trata de la lógica de dominio, donde se codifica el comportamiento central de BreakBrain, en torno al cual se presentan el resto de componentes software.
- **Subsistema de persistencia:** Es el responsable del almacenamiento y recuperación de la información referida a los usuarios y las partidas, los perfiles de evolución cerebral, etc. Se comunica con el núcleo del servidor y actúa de intermediario entre éste y el Sistema Gestor de Base de Datos (SGBD), que es proporcionado por un servicio remoto.

Atendiendo al cliente, las capas MVC en las que se divide su implementación son las enumeradas a continuación:

- **Web (GUI):** Se trata de la capa de presentación, a través de la cual el usuario interactúa con el sistema. Éste es el único componente software del cual el usuario final tiene constancia.
- **Lógica del cliente:** Esta capa manipula la información del usuario y actúa de intermediaria entre la interfaz gráfica (web) y la capa de persistencia temporal del cliente.
- **Persistencia temporal:** Se encarga de mantener la sesión temporal del usuario.

Cada módulo, así como cada capa, ofrece un acoplamiento mínimo con el resto, de tal forma que los cambios efectuados en cualquiera de ellos no afectan directamente a los demás. Por tanto, resulta sencilla la sustitución de módulos por nuevas versiones, por ejemplo, siempre y cuando se respete la interfaz de comunicación.

La figura 5.4 muestra de forma sencilla los componentes que constituyen el sistema, así como la conexión entre ellos.

4.7. Implementación

La implementación de BreakBrain se ha llevado a cabo completamente desde cero, construyendo una red social completa sin emplear frameworks externos u otras facilidades. Las razones para hacerlo así son diversas, pero la más importante ha sido la voluntad de aprender cómo funciona hasta el más mínimo aspecto de un sistema tan grande y de estas características. Además se decidió apostar por tecnologías muy modernas y con buenos resultados.

En esta sección se presentan brevemente las tecnologías empleadas para implementar el sistema. Posteriormente se analiza la estructura del código fuente escrito, así como pequeñas estadísticas sobre el mismo.

4.7.1. Tecnología

A la hora de analizar las tecnologías empleadas en la construcción de BreakBrain, podemos realizar una sencilla clasificación por el ámbito en el cual han sido empleadas. Así pues, a continuación se detallan los lenguajes de programación, frameworks y entornos de ejecución utilizados por el sistema, y clasificados en tres grupos: servidor, cliente y comunicación.

Tecnologías del servidor

- **JavaScript:** Es el lenguaje empleado para la implementación de todos los componentes del servidor.
- **NodeJS:** Entorno de ejecución de JavaScript basado en el motor V8 desarrollado por Google. Supone cierta revolución en el ámbito del desarrollo de aplicaciones web de tiempo real.
- **Express.js [Vis]:** framework MVC para el desarrollo de aplicaciones web con NodeJS.
- **MongoDB:** Se trata de un motor de base de datos NoSQL de gran auge en la actualidad. Ofrece muy buenas estadísticas de rendimiento y escalabilidad.

Tecnologías del cliente

- **HTML5:** Tecnología empleada para la construcción del documento estructurado que constituye la interfaz gráfica del cliente: la web.
- **CSS3:** Hojas de estilos para mejorar la apariencia y comportamiento de los documentos HTML.
- **JavaScript:** Lenguaje empleado para la implementación del comportamiento de la interfaz gráfica y de la lógica del cliente.

Tecnologías de comunicación

- **Websockets:** Tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web.

4.7.2. Estructura del código fuente

El código fuente está categorizado en dos grandes grupos: el código fuente del servidor y el código fuente del cliente. Dado que todos los lenguajes empleados en el desarrollo son interpretados, no es necesario pensar en procesos de compilación ni archivos binarios.

Para hacer uso de la web cliente, el usuario accede mediante su navegador web a la dirección donde se aloja BreakBrain. Además, las tecnologías empleadas para desarrollar el sistema son estándares soportados por todos los navegadores web modernos. Por tanto, no debemos preocuparnos por ningún tipo de proceso de instalación por parte del usuario. Así mismo, todos los ficheros fuente de BreakBrain se encuentran centralizados en el servidor, por lo que la actualización, corrección de errores, etc. resulta muy sencilla.

El código fuente se encuentra clasificado en la siguiente jerarquía de directorios:

```
/                                // Raíz del código. Servidor principal y tests
/node_modules/                  // Módulos de terceros para NodeJS
/server/                         // Módulos de BreakBrain para NodeJS
/server/games/                   // Juegos de BreakBrain (parte del servidor)
/server/public/                  // Contenido estático servido al cliente
/server/public/games/            // Juegos de BreakBrain (parte publica)
/server/public/js/                // Archivos JavaScript para la web del cliente
/server/public/css/              // Hojas de estilos para la web del cliente
/server/public/img/              // Imágenes servidas para la web del cliente
```

En el cuadro 4.2 se ofrece una lista con los diferentes archivos fuente del sistema, junto a una descripción y una muestra de la extensión de cada uno.

4.8. Estadísticas del código fuente

En el cuadro 4.3 se muestra el nivel de utilización de cada lenguaje —de programación o maquetación— en la implementación de BreakBrain. Nótese la diferenciación entre JavaScript del lado del cliente y JavaScript del lado del servidor. Se ha realizado esta separación por la naturaleza clásica del desarrollo web de emplear dos lenguajes diferentes para el cliente y el servidor. En este caso se trata del mismo lenguaje, pero resulta de interés realizar una comparativa de la cantidad de código escrito en ambos extremos de la comunicación.

Cuadro 4.2: Archivos de código fuente

Fichero fuente	Tipo	Num. de líneas	Descripción
/generator.js	JavaScript	67	Generador de datos aleatorios
/server.js	JavaScript	739	Servidor principal
/test.js	JavaScript	21	Módulo de test unitarios
/test-functionality.js	JavaScript	502	Batería de tests funcionales
/test-stress.js	JavaScript	63	Batería de tests de estrés
/server/constants.js	JavaScript	10	Constantes del programa
/server/database.js	JavaScript	56	Módulo de persistencia
/server/email.js	JavaScript	34	Módulo de email
/server/Game.js	JavaScript	65	“Clase” padre de los juegos
/server/games.js	JavaScript	43	Módulo servidor de juegos
/server/games/*	JavaScript	305	Módulos servidores de los juegos
/server/util.js	JavaScript	59	Módulo de utilidades
/public/games.html	HTML	56	Vista de la página de juegos
/public/header.html	HTML	18	Vista de la cabecera
/public/home.html	HTML	77	Vista de la página de inicio
/public/login.html	HTML	95	Vista de la página de login
/public/logout.html	HTML	13	Vista de la página de logout
/public/password.html	HTML	46	Página de recuperación de contraseña
/public/people.html	HTML	93	Vista de la página “gente”
/public/profile.html	HTML	171	Vista de la página de perfil
/public/css/*	CSS	1244	Hojas de estilos del cliente
/public/js/*	JavaScript	1646	Archivos JavaScript del cliente

Cuadro 4.3: Estadísticas del código fuente de BreakBrain

Lenguaje	Nº de archivos	Líneas de código	Protagonismo
JavaScript (cliente)	15	1646	31.3 %
JavaScript (servidor)	13	1964	37.4 %
HTML5	8	398	7.6 %
CSS3	12	1244	23.7 %

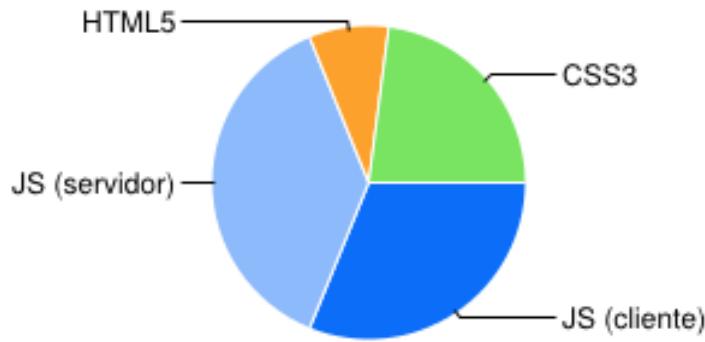


Figura 4.8: Estadísticas del código fuente: los lenguajes de BreakBrain

4.9. Pruebas

Con el fin de asegurar la calidad del software construido se han llevado a cabo diferentes tipos de pruebas sobre el mismo.

4.9.1. Pruebas unitarias

Para la realización de las pruebas unitarias se ha desarrollado un pequeño módulo (`test.js`), que permite la realización de tests sencillos. Estos tests se han repartido por el código fuente en los puntos donde semánticamente tienen sentido, y sólo se lanzan si al ejecutar el servidor se le añade el argumento `-t`. Al ejecutar el servidor de esta forma, el log muestra los casos de éxito y fracaso de todos los tests unitarios. Al finalizar los mismos se ofrece una pequeña estadística del resultado obtenido.

El subsistema de test tiene la particularidad de que posibilita su uso desde los nuevos juegos que se creen, ofreciendo así una manera sencilla de realizar tests unitarios desde la implementación del nuevo juego sin tener que utilizar otros módulos externos. La forma de llevar a cabo tests es muy sencilla:

```
test('descripción del test',
    expresion_que_debe_ser_true, // por ejemplo: instancia != null
    'mensaje de error');
```

El resultado de la ejecución de los tests unitarios básicos del sistema es la mostrada en la figura 4.9. Nótese que dicho resultado es el correspondiente al caso de ejecución más básico, en el que no se carga ningún juego, por lo que a medida que se carguen diferentes juegos dotados de tests, esta salida será mayor.

4.9.2. Pruebas funcionales

Se han llevado a cabo una serie de pruebas funcionales automatizadas, que cubren los aspectos más importantes de la funcionalidad especificada en los apartados 4.3 y 4.5. Dada la naturaleza asíncrona y no bloqueante de la tecnología con la que ha sido desarrollado el servidor del proyecto, estas pruebas se lanzan y analizan de forma asíncrona. Para ello se ha utilizado un módulo de NodeJS llamado *Vows* [Mal].

Estas pruebas se encuentran en el archivo `tests-functionality.js`, y pueden lanzarse sobre el intérprete de NodeJS una vez que el servidor está en ejecución. Los resultados de estas pruebas se muestran en la figura 4.10.

4.9.3. Pruebas de aceptación

Durante el desarrollo del sistema se han realizado pequeñas y repetidas pruebas de aceptación por parte de un subconjunto variado de usuarios finales, recogiendo las opiniones de los mismos y empleando éstas, junto con las estadísticas obtenidas, para la corrección de errores y mejora de aspectos varios.

La elección del subconjunto poblacional sometido a las pruebas de aceptación se ha llevado a cabo mediante un muestreo aleatorio estratificado, con el objetivo de obtener opiniones y observaciones de uso de usuarios de diferentes edades. Los estratos definidos para la realización del muestreo son los siguientes:

- Joven (entre 15 y 30 años de edad)
- Adulto 1 (entre 30 y 45 años de edad)
- Adulto 2 (entre 45 y 60 años de edad)

Durante la planificación de las pruebas se seleccionó aleatoriamente un individuo de cada subconjunto —dichas personas se comprometieron a someterse a las pruebas de forma periódica—. Los individuos representantes de cada estrato son los siguientes:

- Joven - Rodrigo Testillano
- Adulto 1 - Néstor Ceballos
- Adulto 2 - Rafael García

En total se han llevado a cabo 3 sesiones de pruebas de aceptación. A continuación se enumeran las actividades en las que consistió cada una de ellas:

PA-1 La primera sesión consistió en:

- Crearse una cuenta en la red social.
- Autenticarse en el sistema.
- Cerrar la sesión.

- Tratar de acceder al sistema con datos erróneos.
- Simular el olvido de la contraseña para recuperar el acceso a su cuenta.

PA-2 La segunda sesión comprendió el uso de la red y la gestión de perfiles:

1. Explorar las estadísticas del perfil de usuario.
2. Modificar la información personal del perfil.
3. Modificar las preferencias de entrenamiento cerebral.
4. Búsqueda de usuarios.
5. Seguir y dejar de seguir a otros usuarios.

PA-3 En la tercera sesión los individuos utilizaron el subsistema de juegos:

1. Exploración de los juegos individuales.
2. Exploración de los juegos multijugador.
3. Jugar a un juego monojugador.
4. Retar a otro usuario a un juego multijugador.
5. Ser retado por otro usuario y aceptar. Jugar partida multijugador.
6. Ser retado por otro usuario y rechazar la invitación.

Los resultados obtenidos durante estas pruebas de aceptación fueron de gran utilidad, no sólo para solucionar ciertos problemas que no habían sido localizados hasta el momento de la utilización del sistema por parte del usuario final, sino también para mejorar la *usabilidad* ante problemas de diseño detectados durante la observación del uso por parte de los individuos. En resumen, algunos de los beneficios de estas pruebas fueron los siguientes:

- Corrección de pequeños errores de programación (*bugs*).
- Corrección de problemas de distribución de elementos en la web (problemas de usabilidad).
- Aceptación de críticas sobre los esquemas de color y rediseño de los mismos.
- Mejora de la interacción hombre-máquina ante la observación de dificultades de uso de algunos componentes por parte de ciertos individuos.

4.9.4. Pruebas de estrés del sistema

Esta batería de pruebas permite observar el comportamiento y escalabilidad del sistema, observando el comportamiento del mismo en situaciones de poca carga y en situaciones extremas, con gran cantidad de peticiones costosas simultáneas.

Cabe añadir que la probabilidad de que el sistema se vea sometido a estas situaciones extremas en un entorno real es despreciable. Sin embargo, estas pruebas resultan de gran utilidad para estudiar la respuesta del servidor y la base de datos.

Para realizar las pruebas de estrés se ha diseñado un pequeño programa que lanza peticiones simultáneas contra el servidor. Las peticiones lanzadas son de registro y borrado de usuarios. Se ha seleccionado la operación de registro de un usuario —con generación aleatoria de datos— porque es la que más coste tiene para la base de datos, y así nos permite abarcar toda la amplitud del sistema. Por otro lado, las operaciones de eliminación se han incluido en el test simplemente para eliminar los usuarios y perfiles cerebrales creados en la etapa anterior, y así servir también de contraste, al tratarse de la operación de menor coste de la base de datos.

Los resultados obtenidos son los siguientes:

■ 1 única petición (RAM: 13MB)

```
1 STRESS-TEST: Starting stress test
2 STRESS-TEST: 1 false users registered
3 Time spent in 1 false users registration: 648ms
4 STRESS-TEST: False users deleted
5 STRESS-TEST: Stress test finished in
6 Time spent in 1 false users deletion: 270ms
7 Time spent in whole stress test: 918ms
```

■ 10 peticiones simultáneas (RAM: 15.7MB)

```
1 STRESS-TEST: Starting stress test
2 STRESS-TEST: 10 false users registered
3 Time spent in 10 false users registration: 964ms
4 STRESS-TEST: False users deleted
5 STRESS-TEST: Stress test finished in
6 Time spent in 10 false users deletion: 272ms
7 Time spent in whole stress test: 1236ms
```

■ 20 peticiones simultáneas (RAM: 17MB)

```
1 STRESS-TEST: Starting stress test
2 STRESS-TEST: 20 false users registered
3 Time spent in 20 false users registration: 469ms
4 STRESS-TEST: False users deleted
5 STRESS-TEST: Stress test finished in
6 Time spent in 20 false users deletion: 277ms
7 Time spent in whole stress test: 748ms
```

■ 50 peticiones simultáneas (RAM: 19MB)

```
1 STRESS-TEST: Starting stress test
2 STRESS-TEST: 50 false users registered
3 Time spent in 50 false users registration: 864ms
4 STRESS-TEST: False users deleted
5 STRESS-TEST: Stress test finished in
```

```
6 Time spent in 50 false users deletion: 320ms
7 Time spent in whole stress test: 1186ms
```

- 100 peticiones simultáneas (RAM: 21.4MB)

```
1 STRESS-TEST: Starting stress test
2 STRESS-TEST: 100 false users registered
3 Time spent in 100 false users registration: 6955ms
4 STRESS-TEST: False users deleted
5 STRESS-TEST: Stress test finished in
6 Time spent in 100 false users deletion: 1387ms
7 Time spent in whole stress test: 8342ms
```

- 200 peticiones simultáneas (RAM: 28.5MB)

```
1 STRESS-TEST: Starting stress test
2 STRESS-TEST: 200 false users registered
3 Time spent in 200 false users registration: 14460ms
4 STRESS-TEST: False users deleted
5 STRESS-TEST: Stress test finished in
6 Time spent in 200 false users deletion: 1870ms
7 Time spent in whole stress test: 16330ms
```

- 500 peticiones simultáneas (RAM: 59MB)

```
1 STRESS-TEST: Starting stress test
2 STRESS-TEST: 500 false users registered
3 Time spent in 500 false users registration: 34901ms
4 STRESS-TEST: False users deleted
5 STRESS-TEST: Stress test finished in
6 Time spent in 500 false users deletion: 5654ms
7 Time spent in whole stress test: 40555ms
```

- 1000 peticiones simultáneas (RAM: 128MB)

```
1 STRESS-TEST: Starting stress test
2 STRESS-TEST: 1000 false users registered
3 Time spent in 1000 false users registration: 65909ms
4 STRESS-TEST: False users deleted
5 STRESS-TEST: Stress test finished in
6 Time spent in 1000 false users deletion: 14254ms
7 Time spent in whole stress test: 80163ms
```

- 5000 peticiones simultáneas (RAM: 340MB)

```
1 STRESS-TEST: Starting stress test
2 STRESS-TEST: 5000 false users registered
3 Time spent in 5000 false users registration: 322281ms
```

Cuadro 4.4: Estadísticas de las pruebas de estrés en un servidor Debian 64-bits Linux 3.2.0, CPU Intel Core i7 Q720 1.60GHz x 4, RAM 6GB

Peticiones simultáneas	Tiempo total de inserciones	Tiempo total de eliminaciones	Tiempo total del test	Memoria RAM (server)
1	669 ms	271 ms	940 ms	13 MB
10	964 ms	272	1236 ms	15.7 MB
20	469 ms	277 ms	748 ms	17 MB
50	864 ms	320 ms	1186 ms	19 MB
100	6955 ms	1387 ms	8342 ms	21.4 MB
200	14460 ms	1870 ms	16330 ms	28.5 MB
500	34901 ms	5654 ms	40555 ms	50 MB
1000	65909 ms	14254 ms	80163 ms	128 MB
5000	322281 ms	75101 ms	397382 ms	340 MB
10000	547662 ms	229148 ms	776810 ms	520 MB

```

4 STRESS-TEST: False users deleted
5 STRESS-TEST: Stress test finished in
6 Time spent in 5000 false users deletion: 75101ms
7 Time spent in whole stress test: 397382ms

```

- 10000 peticiones simultáneas (RAM: 520MB)

```

1 STRESS-TEST: Starting stress test
2 STRESS-TEST: 10000 false users registered
3 Time spent in 10000 false users registration: 547662ms
4 STRESS-TEST: False users deleted
5 STRESS-TEST: Stress test finished in
6 Time spent in 10000 false users deletion: 229148ms
7 Time spent in whole stress test: 776810ms

```

A la vista de los resultados puede observarse el buen comportamiento del servidor para situaciones normales y extremas. Por supuesto, el caso de recibir gran cantidad de peticiones de registro simultáneas es muy poco probable, pero nos da una idea del rendimiento del sistema. El registro es la operación más costosa (para el subsistema de base de datos), y por ello ha sido elegida como *operación estresante* para someter al sistema a ella.

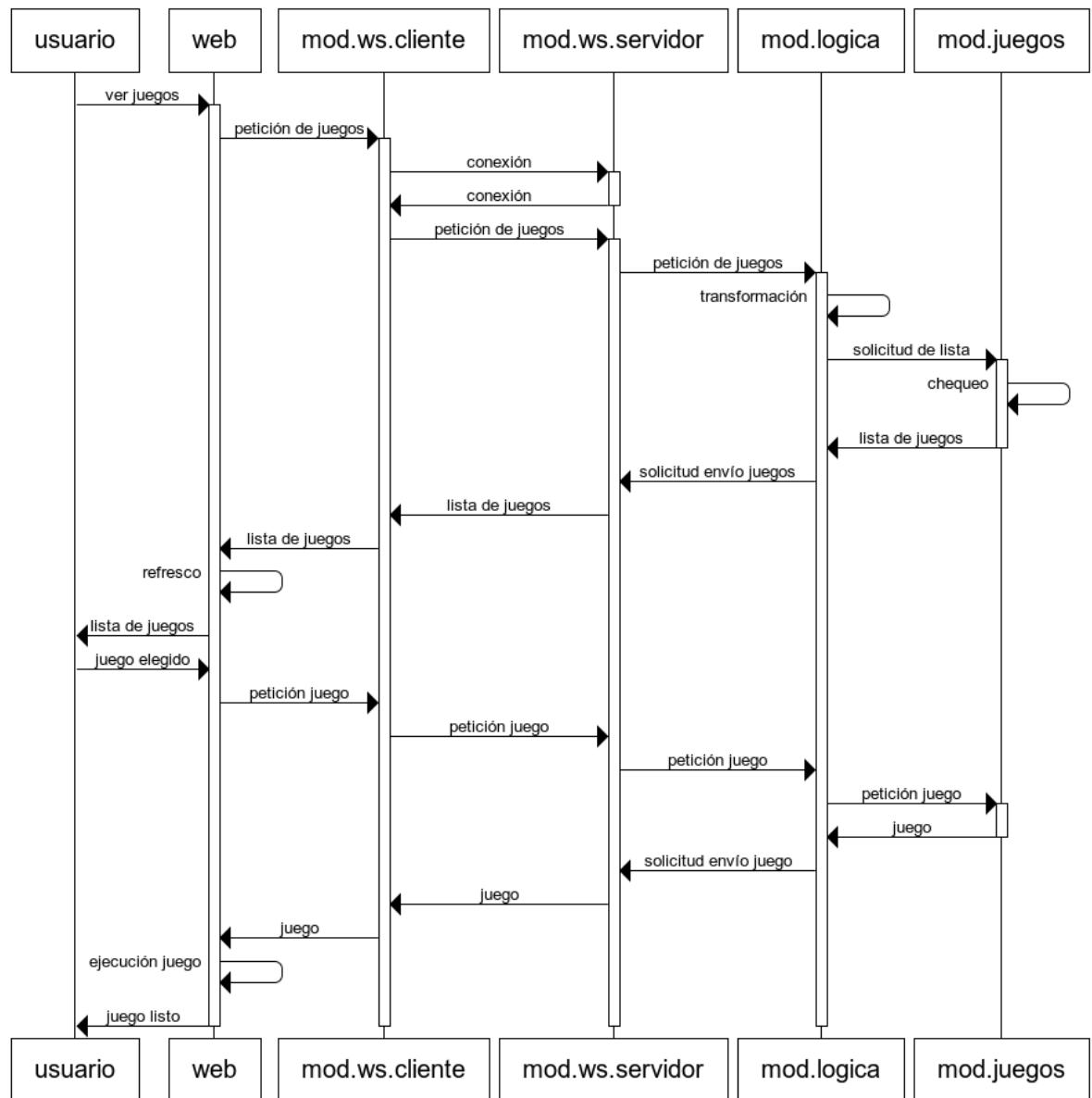


Figura 4.7: Diagrama de interacción del caso de uso UC-12 (flujo normal)

```
sgmonda@envy:~/projects/breakbrain $ node server.js -t
27 Nov 2012 19:24:30::UNIT TEST 1 CORRECT: constants module load
27 Nov 2012 19:24:30::DATABASE: Connecting to MongoDB database...
27 Nov 2012 19:24:30::UNIT TEST 2 CORRECT: mongodb module load
27 Nov 2012 19:24:30::UNIT TEST 3 CORRECT: database server creation
27 Nov 2012 19:24:30::UNIT TEST 4 CORRECT: database object creation
27 Nov 2012 19:24:30::UNIT TEST 5 CORRECT: database collection retrieval
27 Nov 2012 19:24:30::UNIT TEST 6 CORRECT: connection to mongo server
27 Nov 2012 19:24:30::UNIT TEST 7 CORRECT: connection to mongo db
27 Nov 2012 19:24:31::UNIT TEST 8 CORRECT: database connection opening
27 Nov 2012 19:24:32::UNIT TEST 9 CORRECT: database authentication
27 Nov 2012 19:24:32::DATABASE: Connected and authenticated successfully
27 Nov 2012 19:24:32::UNIT TEST 10 CORRECT: database object instantiation
27 Nov 2012 19:24:32::UNIT TEST 11 CORRECT: fs module load
27 Nov 2012 19:24:32::UNIT TEST 12 CORRECT: express module load
27 Nov 2012 19:24:32::UNIT TEST 13 CORRECT: http module load
27 Nov 2012 19:24:32::UNIT TEST 14 CORRECT: web app creation
27 Nov 2012 19:24:32::EMAIL-MODULE: Email subsystem ready to send emails
27 Nov 2012 19:24:32::UNIT TEST 15 CORRECT: email module load
27 Nov 2012 19:24:32::GAMES-MODULE: Loading games server...
27 Nov 2012 19:24:32::WEB SERVER: running on port 20661
27 Nov 2012 19:24:32::UNIT TEST 16 CORRECT: games directory reading
27 Nov 2012 19:24:32::UNIT TEST 17 CORRECT: games count
27 Nov 2012 19:24:32::GAMES-MODULE: Loading game "Un juego"...
27 Nov 2012 19:24:32::UNIT TEST 18 CORRECT: particular game load
27 Nov 2012 19:24:32::UNIT TEST 19 CORRECT: games module load
27 Nov 2012 19:24:32::UNIT TEST 20 CORRECT: websockets module load
27 Nov 2012 19:24:32::WEBSOCKETS SERVER: running on port 20661
27 Nov 2012 19:24:32::MAIN SERVER: The whole server is ready!

27 Nov 2012 19:24:32::CORRECT UNIT TESTS: 20 (100%)
^Csgmonda@envy:~/projects/breakbrain $
```

Figura 4.9: Resultado parcial de tests unitarios

```
sgmonda@envy:~/projects/breakbrain $ vows tests-functionality.js --spec
  ◇ Pruebas funcionales de BreakBrain
    Websockets connection
      ✓ Ping test
      ✓ Message communication
    Users and brains registration
      ✓ Users count
      ✓ Brain profiles count
      ✓ Register user
      ✓ Users/Brains correspondence
    User management
      ✓ Login
      ✓ Retrieve user info
      ✓ Updating user info
      ✓ User account activation
    Miscellaneous
      ✓ Hash checking process
    Social behavoir
      ✓ Following status checking
      ✓ Start following
      ✓ Stop following
      ✓ Users search
    Games module
      ✓ Get games list
      ✓ Games recommendation
      ✓ Users recommendation
    User and brain profile deletion
      ✓ Delete user
      ✓ Delete brain profile
    ✓ OK » 20 honored (5.430s)
sgmonda@envy:~/projects/breakbrain $
```

Figura 4.10: Resultados de la batería de tests funcionales

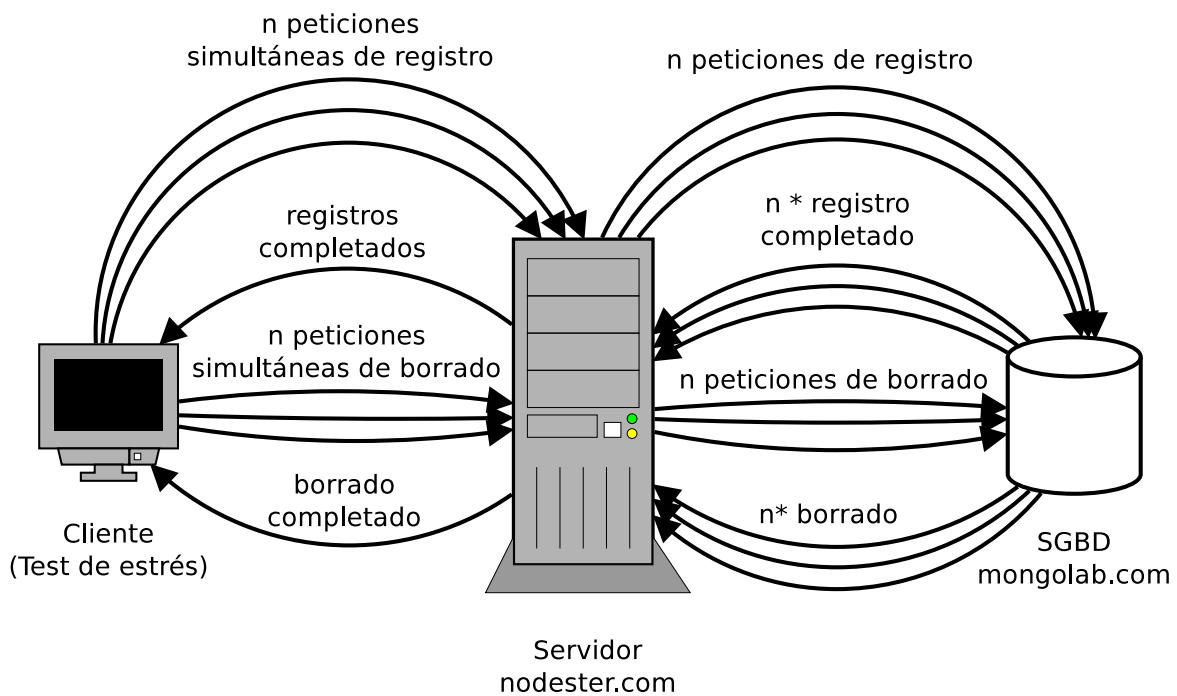


Figura 4.11: Diseño del test de estrés

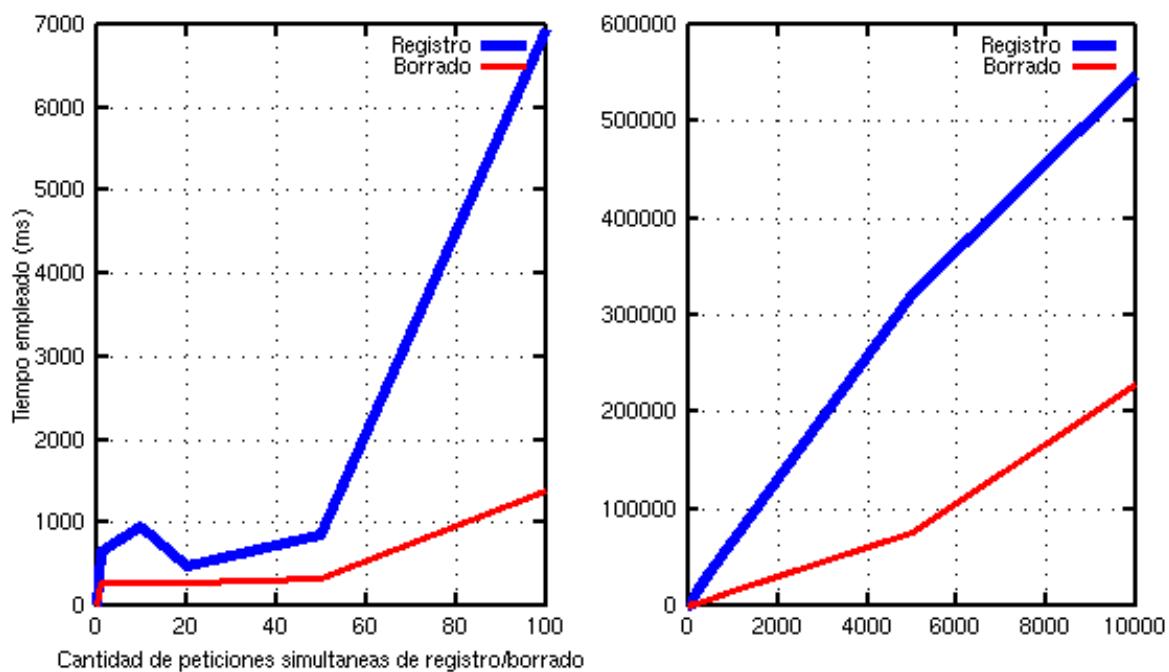


Figura 4.12: Resultados de las pruebas de estrés

Capítulo 5

Arquitectura

ESTE capítulo pretende ofrecer una visión completa de la composición del sistema informático que constituye BreakBrain, profundizando en los diferentes componentes que lo forman y haciendo hincapié en los patrones de diseño utilizados y otros detalles de interés para el lector con conocimientos técnicos.

A lo largo de las siguientes secciones, se presentará la arquitectura detallada del sistema completo, para después estudiar a más bajo nivel los componentes principales por separado, centrando la atención en el desarrollo de software.

5.1. Composición del sistema

En la figura 5.1 se ofrece una visión detallada de los componentes que forman BreakBrain. Se trata de un enfoque de medio nivel que se adentra en la composición de los diferentes elementos de despliegue del sistema vistos en la sección 4.4. Como puede apreciarse, los dos protagonistas del enfoque *cliente-servidor* que sigue BreakBrain son tan complejos que se hace patente la necesidad de estudiarlos de forma detallada por separado.

A la vista de la figura 5.1 se perciben cuatro elementos esenciales en la composición del sistema:

- **Cliente**

Sitio web ejecutado sobre un navegador corriente. El uso de JavaScript que hace BreakBrain es intenso, tanto por la ejecución de los juegos como por el subsistema de comunicación permanente (necesario para los juegos multijugador), por lo que un navegador con buen soporte de JavaScript (como Chrome/Chromium [Goob]) otorgará siempre una mejor experiencia de usuario.

- **Servidor principal**

Se trata del corazón del sistema, compuesto principalmente por un servidor web corriente, al que se le ha añadido un servidor de comunicaciones mediante sockets TCP (con mensajes encapsulados en HTTP) para mantener una comunicación ininterrumpida con los clientes, y ofrecer así la base para la ejecución de juegos multijugador. El servidor se ejecuta en Nodester [Nodb], Platform As A Service (PAAS) de Software

Libre que ofrece un buen servicio de forma totalmente gratuita.

■ Servidor de juegos

La piedra angular del servicio de juegos monojugador y multijugador. Construido de forma extensible, permite la integración de juegos de terceros en la plataforma.

■ Sistema gestor de base de datos

Se ha optado por una base de datos no relacional MongoDB, alojada en los servicios de Amazon WS [Ama] y gestionada por MongoLab [Monb]. Este último Software As A Service (SAAS) se encarga del mantenimiento de la base de datos, asegurando la alta disponibilidad y escalabilidad.

5.2. Arquitectura del sitio web

La aplicación web de BreakBrain ha sido desarrollada utilizando NodeJS y ExpressJS, lo que fuerza la arquitectura de la misma hacia un enfoque Modelo-Vista-Controlador (MVC). A su vez, la parte cliente (página web) ha sido desarrollada siguiendo este patrón.

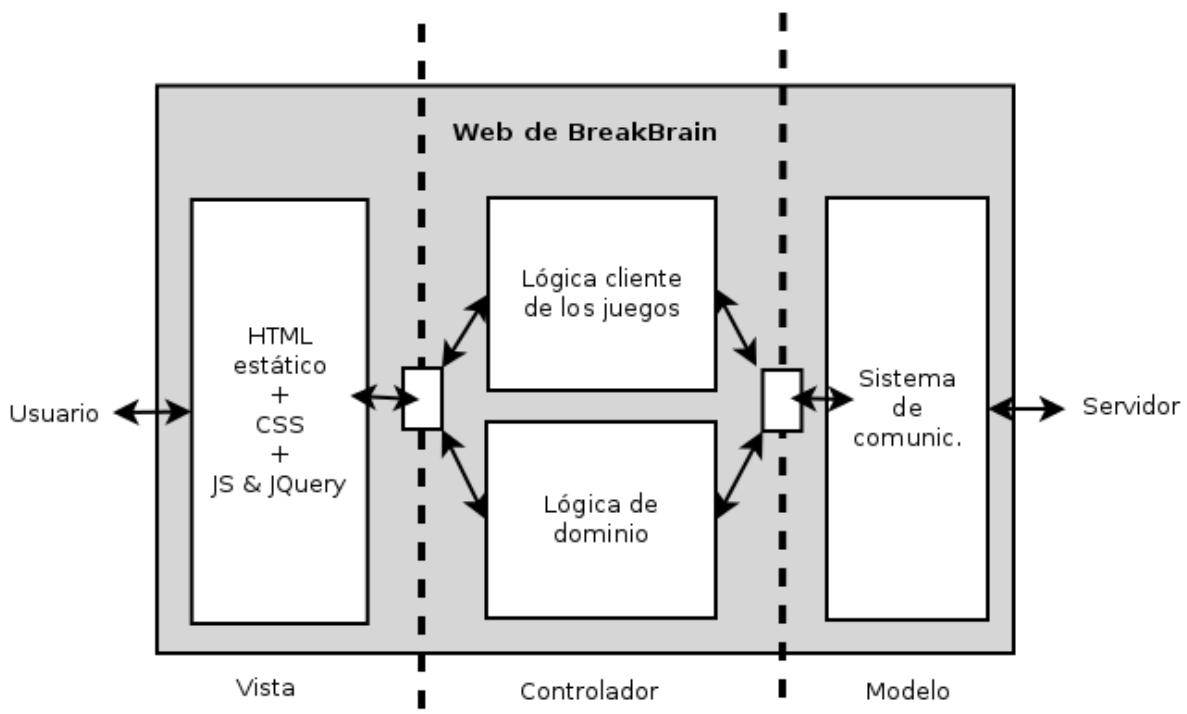


Figura 5.2: Patrón MVC en el cliente de BreakBrain

5.2.1. Patrón MVC

La arquitectura Modelo-Vista-Controlador en el cliente de la plataforma se distingue atendiendo a la división de los componentes que la constituyen en los siguientes tres grupos:

- **Modelo:** Sistema de comunicación con el servidor.
- **Vista:** Interfaz gráfica con la que el usuario interactúa. Se trata de código HTML embellido con estilos CSS y manipulado mediante JavaScript, haciendo uso de JQuery.
- **Controlador:** Lógica principal del cliente. Aquí pueden distinguirse dos componentes principales:
 - Lógica de dominio a nivel de cliente
 - Lógica de la parte cliente de los juegos

En la figura 5.2 se aprecia el esquema expuesto.

5.3. Arquitectura del servidor

La aplicación web de BreakBrain ha sido desarrollada utilizando NodeJS y ExpressJS, lo que fuerza la arquitectura de la misma hacia un enfoque Modelo-Vista-Controlador (MVC). A su vez, la parte servidora ha sido desarrollada siguiendo este patrón.

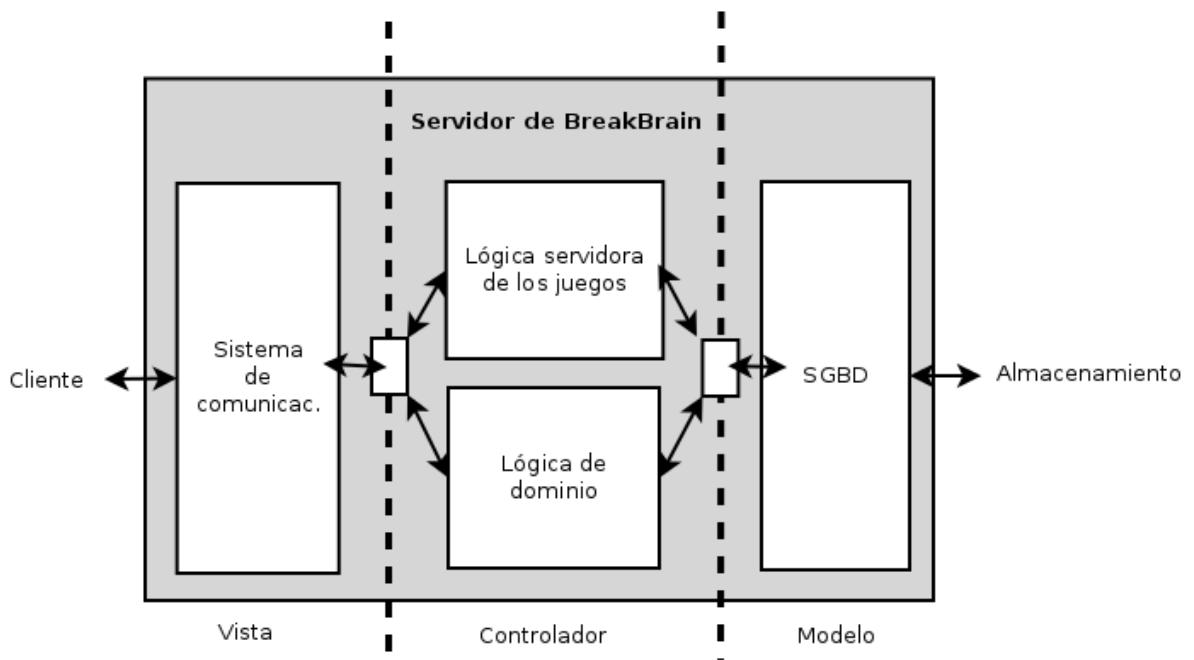


Figura 5.3: Patrón MVC en el servidor de BreakBrain

5.3.1. Patrón MVC

La arquitectura Modelo-Vista-Controlador en el servidor de la plataforma se distingue atendiendo a la división de los componentes que la constituyen en los siguientes tres grupos:

- **Modelo:** Subsistema de persistencia, encargado de la manipulación de la base de datos.
- **Vista:** Interfaz de comunicación con el cliente mediante websockets.
- **Controlador:** Lógica central del servidor, compuesta por dos subsistemas:
 - Lógica de dominio a nivel de servidor
 - Lógica de servidor de los juegos

En la figura 5.3 se aprecia el esquema expuesto.

5.4. Diseño modular

La propia esencia de NodeJS, la tecnología con la que BreakBrain ha sido implementado, guía al desarrollador hacia una estructura basada en módulos independientes.

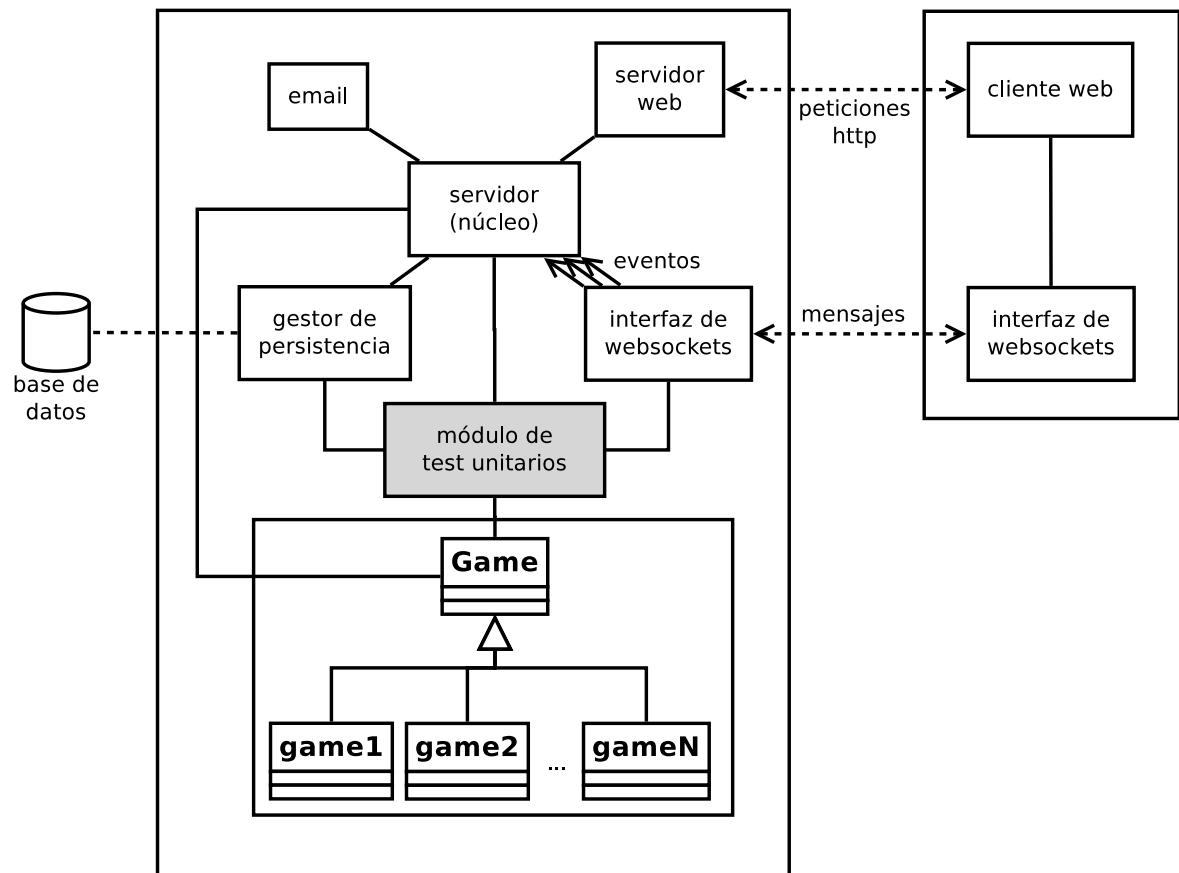


Figura 5.4: Diseño modular del sistema

En la figura 5.4 puede apreciarse la estructura modular de la plataforma, donde pueden distinguirse los siguientes componentes, agrupados según su funcionalidad:

■ Gestor de persistencia

Módulo encargado de la configuración y comunicación con el Sistema Gestor de Base de Datos (SGBD) utilizado (MongoDB).

■ Subsistema de email

Módulo encargado del envío automático de emails.

■ Servidor web

Módulo de manejo de peticiones y respuestas HTTP, así como de la generación de contenido web dinámico para el cliente.

■ Servidor de websockets

Módulo de conexión con el cliente mediante websockets. Mantiene (y pone a disposición de los juegos) una comunicación ininterrumpida con el navegador web de los usuarios.

■ Núcleo del servidor

Módulo central, con la lógica de interconexión del resto de módulos y la creación de los propios servidores web y de websockets.

■ Módulo de test

Módulo de pruebas de la plataforma.

■ Servidor de juegos

Módulo encargado de la gestión de los juegos.

■ Cliente web

El sitio web con el que el usuario interactúa.

■ Cliente de websockets

Puente de comunicación entre el cliente y el servidor.

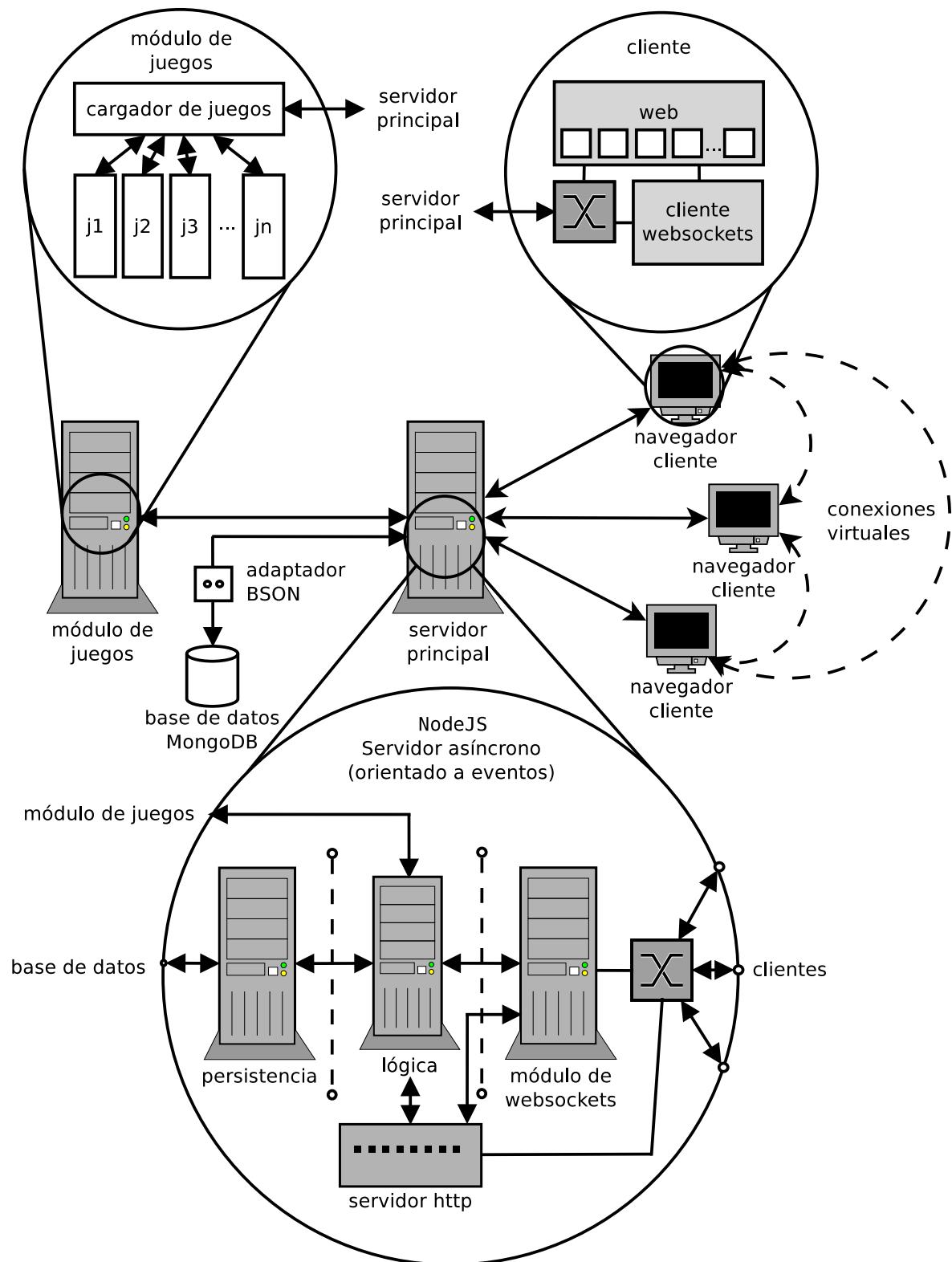


Figura 5.1: Arquitectura detallada del sistema

Capítulo 6

Resultados

UNA vez adquiridos los conocimientos básicos sobre la estructura y funcionamiento del cerebro humano (sección 3.4), así como sobre la neuroplasticidad (sección 3.5) y el carácter maleable que ésta otorga al cerebro, en este capítulo se hace uso de esos conocimientos para la creación de un sistema de entrenamiento cerebral complejo. Así mismo se ofrece una especificación detallada sobre cómo realizar una medición cuantitativa para poder valorar la evolución de dicho entrenamiento.

Por último se expondrán algunos algoritmos de recomendación que, haciendo uso de las métricas previamente definidas, permitirán encontrar usuarios afines a uno dado, así como juegos adecuados a un usuario en un instante concreto.

6.1. Especificación del entrenamiento cerebral

En base a la etapa de investigación documentada en las secciones 3.4 y 3.5, las consideraciones que se han llevado a cabo para reconocer y diferenciar las capacidades mentales estimulables mediante el ejercicio agrupan estas capacidades en 5 grandes conjuntos:

1. Memoria
2. Resolución de problemas
3. Atención
4. Velocidad de procesamiento
5. Flexibilidad

A continuación se presenta una jerarquía completa de las capacidades mentales consideradas, así como de las habilidades en que estas se han considerado subdivididas, detallándose el significado de cada una y cómo debe ser estimulada. En la figura 6.6 se presenta un diagrama esquemático de dicha jerarquía.

6.1.1. Memoria

La mejora de la memoria se centra en tres aspectos realmente relevantes en la vida cotidiana: la memoria de trabajo, la memoria espacial y la memoria de asociación nombre-cara.

■ Memoria de trabajo

Procesos y estructuras de información empleados para el almacenamiento temporal y la manipulación de información durante el desarrollo de una tarea.

La memoria de trabajo requiere la activación de un circuito de neuronas, el cual activa en sí la memoria propiamente dicha. Esta memoria, si bien es activada desde la *corteza prefrontal*, requiere a su vez la activación del resto de estructuras neuroanatómicas implicadas, como el *lóbulo temporal* para el significado o el *lóbulo occipital* para la imagen visual.

En un contexto informático, la memoria de trabajo se asemeja a la memoria RAM de un computador.

Los juegos para mejorar esta capacidad deben ofrecer pequeñas cantidades de información —cada vez mayores según se aumente el nivel de dificultad— durante un periodo corto de tiempo. El jugador debe asimilar esas porciones de información y recordarlas, para utilizarlas transcurrido un periodo de tiempo relativamente corto. El clásico juego de las parejas de cartas es un buen ejemplo. Otra posibilidad es ofrecer lecturas de párrafos cortos, y después jugar con esa información. Del mismo modo se puede jugar con otros sentidos, ofreciendo información auditiva, por ejemplo.

■ Memoria espacial

Memoria responsable de registrar la información sobre el entorno y la orientación en el espacio.

Por ejemplo, en la vida real, la memoria espacial es la que nos permite recordar la ubicación del cuarto de baño de una casa ajena. De forma aún más sencilla, la memoria espacial nos permite trabajar en un escritorio, recordando dónde se encuentran los papeles, los lapiceros y bolígrafos, etc.

Los juegos dedicados al ejercicio y estimulación de esta capacidad mental deberán ofrecer un paisaje de objetos —de complejidad variable, dependiendo de la dificultad de juego— y dar un tiempo —limitado o no— para que el jugador memorice la localización de todos ellos. Después el jugador deberá reconstruir ese paisaje, de forma completa o parcial.

■ Memoria de asociación nombre-cara

Se trata de los procesos relacionados con el registro y recuperación de la información que relaciona objetos visuales con objetos verbales. El caso más comprensible es el de la asociación entre las caras de las personas y sus nombres. Estimular esta capacidad incrementa la habilidad para recordar el nombre de las personas que nos presentan y vemos por primera vez.

La estimulación de esta capacidad debe llevarse a cabo mediante el ejercicio de asociación de relacionar cada objeto de un conjunto con su único correspondiente de otro (u otros). Los juegos dedicados al trabajo de este tipo de memoria deberán mostrar la relación existente entre ciertos objetos y sus etiquetas, para permitir que el jugador pueda analizar y memorizar esas relaciones y reconstruirlas con posterioridad.

6.1.2. Resolución de problemas

Este conjunto se compone de las tres capacidades mentales que entran en juego a la hora de enfrentarnos y resolver cualquier tipo de problema: la aritmética, el razonamiento lógico y el razonamiento cuantitativo:

■ Aritmética

Esta capacidad mental consiste en la realización de operaciones matemáticas de forma correcta y rápida. Comprende la técnica de cálculo para las operaciones de suma, resta, multiplicación y división.

La estimulación de esta capacidad se llevará a cabo con la exposición del jugador a repetidas operaciones matemáticas sencillas. La resolución de muchas operaciones de pequeño tamaño obtiene mejores resultados que la resolución de pocas operaciones gigantescas.

■ Razonamiento lógico

Es el proceso de lógica mediante el cual, partiendo de uno o más juicios, se deriva la validez o falsedad de otro juicio distinto. Tiene que ver con el reconocimiento de patrones y toma de decisiones.

Los juegos destinados a estimular esta habilidad consistirán en encontrar reglas ocultas clasificando objetos. Se mostrarán secuencialmente una serie de objetos, y el jugador debe indicar si cree que cada uno cumple o no una regla desconocida. A partir de sus aciertos y fallos deberá ir desentrañando la regla.

Por ejemplo, una partida puede consistir en la regla “*polígonos con más de 4 lados y color rojo, verde o azul*”. El juego irá mostrando polígonos de diferentes colores y número de lados, y el jugador deberá indicar si cree que cada figura cumple o no la regla. Si una figura la cumple, el jugador sabrá que ese color y número de lados cumplen la regla, por lo que en un caso similar indicará que sí se cumple. Si la misma figura pero con un color diferente no cumple la regla, entonces el jugador aprenderá que el nuevo color no es aceptado por esa regla.

■ Razonamiento cuantitativo

Como ya se dijo en la sección 3.5.2, el razonamiento cuantitativo es utilizado por los seres humanos para realizar estimaciones.

Los juegos de razonamiento cuantitativo consistirán en la toma de decisiones simples en un corto espacio de tiempo. Dichas decisiones supondrán la elección de una opción entre varias, pudiendo ser el número mayor de entre un conjunto de números o resultados de operaciones, el lado que pesa más de una balanza con diferentes pesos a ambos lados, etc.



Figura 6.1: Decisiones de ejemplo para juegos de razonamiento cuantitativo: ¿Qué cuadrado de cada pareja tiene un valor mayor?

6.1.3. Atención

Este subconjunto de habilidades mentales se subdivide en 2 aspectos: Concentración y Campo visual.

■ Concentración

La concentración es la habilidad mental consistente en fijar los sentidos y pensamientos en una única tarea, objeto, etc. discriminando al resto. Es el filtrado de información sensorial, atendiendo una única fuente y discriminando el resto.

Los juegos de concentración consistirán en el enfrentamiento del jugador a una situación en la que tenga que centrarse en un único elemento de un conjunto homogéneo. Un posible juego de este tipo puede ser aquel que muestre una serie de objetos iguales apuntando en diferentes direcciones, obligando al jugador a indicar la dirección en la que apunta uno sólo de ellos. Cada grupo debe ser diferente, pero el objetivo debe seguir un patrón sencillo para que el usuario lo localice rápido y pueda indicar su respuesta. La figura 6.2 muestra una posible aproximación, en la que los elementos son flechas y el jugador debe indicar dónde apunta la del medio.

■ Campo visual

El campo visual es el espacio que un individuo puede ver en un instante sin mover sus ojos. El diseño de juegos destinados a la estimulación de esta habilidad es sencillo: basta con mostrar varios elementos repartidos por el espacio de juego de forma simultánea, y luego, tras ocultarlos, solicitar al jugador que realice algún tipo de tarea con los objetos que se mostraron y ya no aparecen. Por ejemplo, se pueden mostrar varios números y, después de ocultarlos, pedirle al usuario que haga click en las ubicaciones aproximadas de los números por orden creciente o decreciente de valor.



Figura 6.2: Situación de ejemplo para un juego de concentración. El jugador indica (por ejemplo mediante las teclas de dirección), la dirección en la que el elemento central apunta.

6.1.4. Velocidad de procesamiento

Grupo mental centrado en la minimización del tiempo de respuesta ante estímulos y situaciones problemáticas. Se divide en las siguientes habilidades: Procesamiento de información y Orientación espacial:

- **Procesamiento de información**

Habilidad consistente en la interpretación de la información sensorial. Tiene una relación directa con la capacidad de adaptación a los cambios del entorno.

Los juegos dedicados al entrenamiento del procesamiento de información consistirán en una serie de elementos mostrados secuencialmente, de uno en uno, de tal forma que algunos se repitan de forma consecutiva. Cuando el jugador ve cada elemento, indica si éste es exactamente igual que el anterior mostrado o no. Al responder se pasa al siguiente objeto, y así sucesivamente. El objetivo es responder de forma correcta el mayor número de veces en un tiempo limitado.

- **Orientación espacial**

La orientación espacial consiste en la conciencia tridimensional de la situación de un individuo en un entorno físico.

Los juegos dedicados a entrenar esta habilidad consistirán en un espacio bidimensional sobre el que se situarán elementos cuya posición memorizar. Después el mapa sufrirá algún tipo de transformación de *efecto espejo* o rotación, y el jugador tendrá que localizar dichas posiciones sobre el mapa modificado. Alternativamente, para el caso concreto de los juegos multijugador, otra posibilidad sería la de resolver un laberinto simétrico, empezando cada jugador por un extremo, y compitiendo por llegar antes que el otro al centro del laberinto. Dicho laberinto deberá sufrir transformaciones como las mencionadas, de tal manera que el desplazamiento por el mismo suponga la inversión de direcciones —si el laberinto es rotado 90 grados en sentido horario, en el nuevo laberinto resultante las direcciones se verían alteradas: para desplazarse hacia arriba

habría que usar la tecla izquierda, para moverse hacia la derecha habría que utilizar la tecla de dirección hacia arriba, y así sucesivamente—.

La figura 6.3 muestra un ejemplo de laberinto multijugador para un juego de estimulación de la orientación espacial.



Figura 6.3: Escenario de ejemplo para un juego multijugador de orientación espacial

6.1.5. Flexibilidad

La flexibilidad es el subconjunto más amplio de habilidades mentales: *control de impulsos, planificación, conmutación de tareas y fluidez verbal* constituyen los aspectos relacionados con esta capacidad:

- **Control de impulsos**

Como principal aspecto diferenciador entre el ser humano y el resto de animales, el control de impulsos permite priorizar unos objetivos por encima de posibles impulsos naturales.

Existen algunos juegos típicos relacionados con el control de impulsos que son muy buenos candidatos para ser implementados para BreakBrain. Son los juegos en los que se muestran palabras que representan a un color escritas en color (el mismo u otro). Por ejemplo se puede repetir de forma aleatoria, alternando los colores en los que las palabras aparecen escritas. El jugador debe indicar cuándo coincide el nombre del color representado por la palabra y el color en el que aparece.

- **Planificación**

La planificación es la ordenación de tareas por prioridad a lo largo de un periodo temporal. Los juegos destinados a entrenar esta habilidad deben centrarse en la preparación de rutas sobre un escenario mediante pequeños cambios que vayan construyendo el camino. Esos cambios deben estar limitados en número o, alternativamente, utilizar los movimientos como penalización temporal para calcular la puntuación.



Figura 6.4: Ejemplo de prueba para ejercitarse el control de impulsos

■ Comutación de tareas

La comutación de tareas es la adaptación de la atención intercambiando entre diferentes ejercicios. Dependiendo de la situación (tarea actual) será conveniente actuar de una forma y no de otra.

Los juegos destinados a estimular y entrenar la habilidad de comutación de tareas deben definir situaciones diferentes y asociar a cada una de ellas una forma de proceder. Una vez expuestas al jugador, dichas situaciones se alternarán de forma continuada, y el jugador tendrá que proceder como se espera en cada una de ellas. La figura 6.5 muestra un ejemplo sencillo en el que varias flechas se desplazan en la misma dirección (diferente en cada actividad puntuable). Existen dos situaciones:

- Las flechas son rojas: el jugador debe indicar en qué dirección se mueven
- Las flechas son verdes: el jugador debe indicar a qué dirección apuntan

Los dos escenarios se repiten, alternándose de forma aleatoria. El jugador debe responder, indicando la dirección que corresponda según la situación, en el menor tiempo posible.

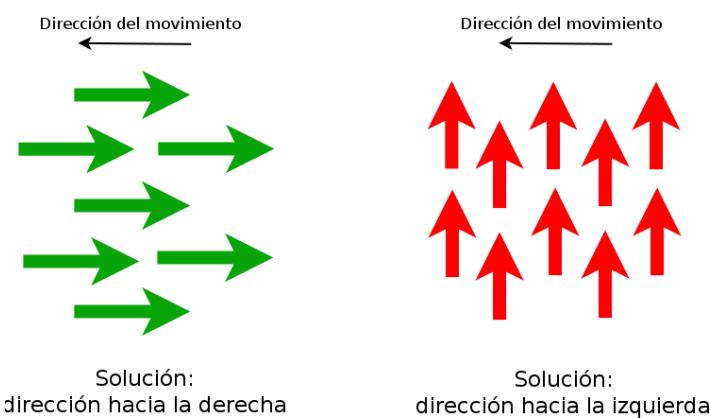


Figura 6.5: Ejemplo de prueba para entrenar la comutación de tareas

■ Fluidez verbal

Esta habilidad consiste en el acceso rápido al vocabulario conocido durante una tarea de escritura o una conversación. Resulta de vital importancia en la comunicación.

Los juegos dedicados a entrenar la fluidez verbal deben ofrecer partes de palabras (por ejemplo las dos o tres primeras letras) y permitir que el jugador vaya introduciendo palabras que cumplan ese patrón. Cada palabra correcta sumará una puntuación directamente proporcional a la longitud de la misma.

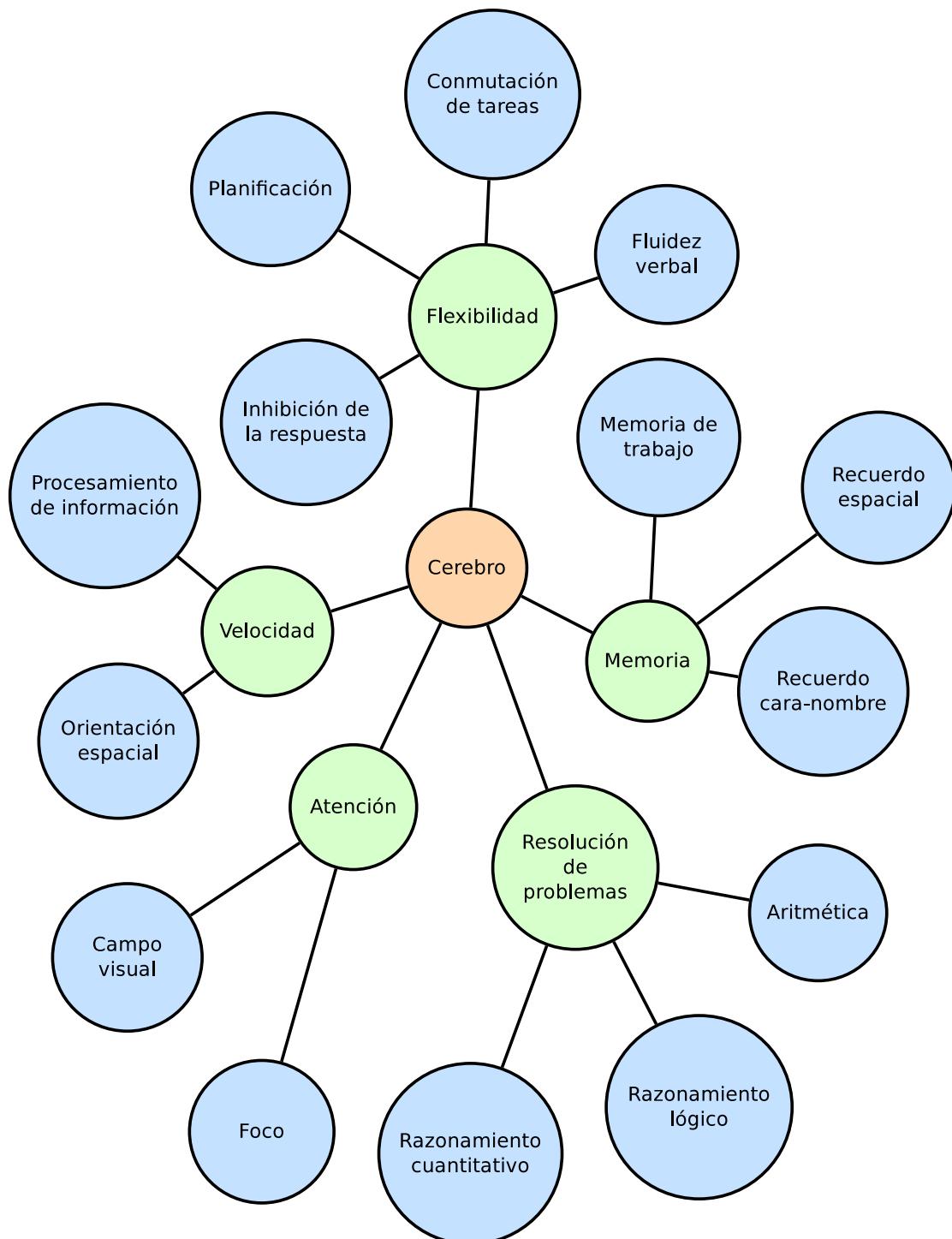


Figura 6.6: Diagrama de capacidades y habilidades mentales

6.2. Medición cuantitativa de la evolución cerebral

A lo largo de esta sección se expondrán los conceptos básicos del diseño de la plataforma, para después desarrollar el sistema de puntuación de partidas y el conjunto de variables cerebrales que posibilitarán, por un lado la medición de la evolución de entrenamiento, y por otro la clasificación y recomendación de usuarios y juegos.

6.2.1. Juegos, partidas y actividades puntuables

El diseño de BreakBrain hace distinción entre tres conceptos principales: juegos, actividades puntuables y partidas. En la figura 6.7 se ofrece una visión gráfica de la relación entre estos términos.

Resulta de vital importancia comprender bien estos conceptos para poder comprender el sistema de medición de la evolución de entrenamiento que ha sido creado en torno a ellos.

Juego

Un juego es una pequeña aplicación interactiva, centrada en una única habilidad mental (perteneciente a una capacidad), integrada en la plataforma y que permite su utilización por parte de uno o más usuarios de forma simultánea.

En base a la cantidad de jugadores que puedan enfrentarse entre sí en un juego, distinguimos entre juegos monojugador y juegos multijugador:

■ Juegos monojugador

Se trata de juegos diseñados para ser utilizados por un único usuario. No supone ningún tipo de interacción con otros usuarios. En este tipo de juegos no se gana o se pierde, sino que el resultado de los mismos es un valor cuantitativo que indica el grado de éxito de la partida (término que se estudia más adelante).

■ Juego multijugador

En este caso se trata de juegos diseñados para ser utilizados por dos jugadores a la vez, suponiendo un enfrentamiento entre ellos. Sólo un jugador de los dos puede ganar cada ronda o partida (término que se estudia más adelante).

Partida

Una partida es cada una de las instancias jugables de un juego. Para ser más claros, cuando dos jugadores se encuentran jugando al mismo juego individual, cada uno en su máquina y sin realizar interacción alguna con el otro, cada uno de esos jugadores está jugando realmente una partida distinta del mismo juego.

Por otro lado, cuando dos jugadores se enfrentan cara a cara a un juego, compitiendo el uno contra el otro, ambos se encuentran en una misma partida de dicho juego. Obviamente

en este caso siempre se tratará de juegos multijugador.

Actividad puntuable

La actividad puntuable es la unidad mínima jugable y, por lo tanto, la unidad mínima de entrenamiento y puntuación. Cada partida está compuesta por varias actividades puntuables. Por ejemplo, dado un juego de memoria en el que el usuario tiene que recordar la localización de grupos de objetos, cada actividad puntuable sería cada distribución de objetos a la que el usuario debe enfrentarse y resolver.

En base al número de actividades puntuables que las compongan, distinguimos entre los siguientes tipos de partidas:

- **Partidas de corta duración:** Partida formada por 10 actividades puntuables
- **Partidas de duración media:** Partida formada por 30 actividades puntuables
- **Partidas de larga duración:** Partida formada por 50 actividades puntuables

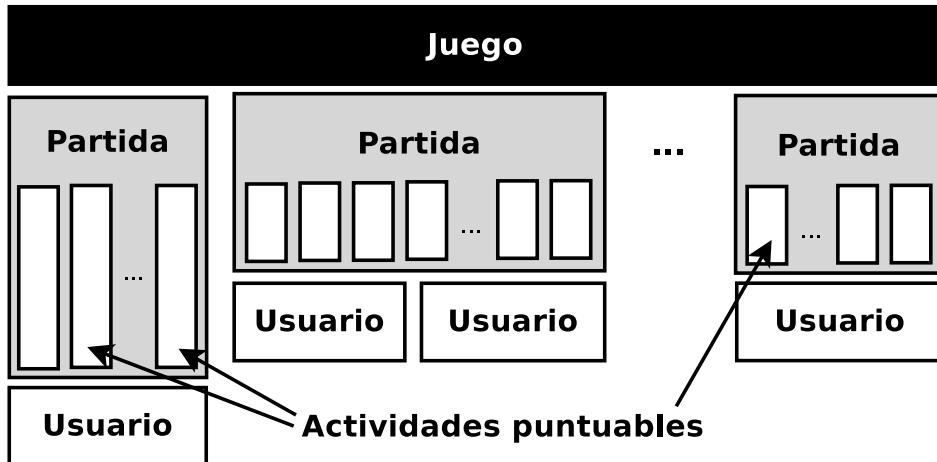


Figura 6.7: Juegos, partidas y actividades puntuables

6.2.2. Puntuación de partidas

Tal y como se ha explicado en la sección 6.2.1, una partida p está compuesta por varias actividades puntuables a_i . La puntuación de una partida, s_p , es la suma de las puntuaciones obtenidas en cada actividad puntuable:

$$s_p = \sum_{i=0}^A s_i \quad (6.1)$$

siendo A el número total de actividades puntuables completadas en la partida p , y s_i la puntuación de cada una de esas actividades.

La puntuación de cada actividad, s_i , se basa en dos aspectos: el tiempo empleado para completarla, t_i , y la valoración de éxito de la misma, e_i . Esta valoración de éxito puede ser satisfactoria o no satisfactoria, lo que matemáticamente se traduce en un 1 o un 0. Por otro lado, el tiempo t_i debe encontrarse normalizado entre 0 y 1, para lo cual basta con dividir el tiempo empleado en realizar la actividad i entre el tiempo empleado en realizar la actividad que más duración haya tenido. Teniendo en cuenta ambos aspectos, la puntuación de una actividad puntuable a_i es la siguiente:

$$s_i = \frac{1}{t_i} \cdot e_i \quad (6.2)$$

Reemplazando esta expresión en la ecuación 6.1, tenemos:

$$s_p = \sum_{i=0}^A \frac{1}{t_i} \cdot e_i \quad (6.3)$$

Las actividades puntuables que no hayan sido completadas reciben una puntuación $s_i = 0$. Dependiendo del tipo de partida, será o no posible darla por finalizada sin haber completado todas las actividades puntuables que la constituyen:

- **Puntuación en partidas monojugador:** el jugador recibe los puntos obtenidos en todas las actividades puntuables que constituyen la partida al finalizar la misma.
- **Puntuación en partidas multijugador:** una vez finalizadas todas las actividades puntuables de la partida por parte de uno de los jugadores, éste es considerado vencedor y la partida termina. Ambos jugadores reciben los puntos obtenidos durante la misma. En el caso del jugador perdedor, recibirá la puntuación de las actividades puntuables que haya completado satisfactoriamente antes de que el otro jugador haya terminado.

Teniendo en cuenta todo lo anterior, una partida p formada por A actividades puntuables tendrá una puntuación s_p tal que $0 \leq s_p \leq A$.

6.2.3. Variables de clasificación

El perfil de cada usuario contiene la puntuación de cada habilidad cerebral, calculada mediante la suma de las puntuaciones obtenidas en cada partida de cada juego destinado a estimular esa habilidad. Además de ello contiene otras variables, como la cantidad de partidas ganadas, el sexo del individuo, etc. Todas estas variables se clasifican en 3 grupos:

Variables estáticas o de agrupación

- **sex:** Sexo (hombre o mujer)
- **age:** Edad
- **keywords:** Palabras clave

- **training-interests:** Lista de habilidades mentales que el usuario está interesado en entrenar

Variables dinámicas mentales

- **memory:** puntuación en el área de Memoria. Se trata de la suma de las siguientes tres puntuaciones:
 - **working-memory:** puntuación en memoria de trabajo.
 - **spatial-memory:** puntuación en memoria espacial
 - **face-name:** puntuación en memoria de asociación nombre-cara
- **problem-solving:** puntuación en el área de Resolución de problemas. Se calcula mediante la suma de las siguientes tres puntuaciones:
 - **arithmetic:** puntuación en aritmética.
 - **logical-reasoning:** puntuación en razonamiento lógico.
 - **quantitative-reasoning:** puntuación en razonamiento cuantitativo.
- **attention:** puntuación en la capacidad de Atención. Su valor es la suma de los valores de las siguientes dos variables:
 - **focus:** puntuación en la habilidad de concentración
 - **visual-field:** puntuación en campo visual
- **speed:** puntuación en el área de Velocidad de procesamiento. Se trata de la suma de las siguientes dos variables:
 - **information-processing:** puntuación en procesamiento de la información
 - **spatial-orientation:** puntuación en orientación espacial
- **flexibility:** puntuación en la capacidad de Flexibilidad. Se computa mediante la suma de las siguientes cuatro variables:
 - **response-inhibition:** puntuación en control de impulsos
 - **planning:** puntuación en planificación
 - **task-switching:** puntuación en commutación de tareas
 - **verbal-fluency:** puntuación en fluidez verbal

Variables dinámicas sociales

- **followers:** cantidad de personas que siguen al usuario.
- **followees:** cantidad de personas a las que el usuario sigue.

Grupo	Habilidad	Descripción	Entrenamiento mediante juegos
Memoria	Memoria de trabajo	Almacenamiento temporal de información durante el desarrollo de una tarea.	Mostrar pequeñas cantidades de información para memorizar y después requerirlas de algún modo.
	Memoria espacial	Información del entorno, localización de objetos y orientación en el espacio.	Ofrecer un paisaje de objetos para su memorización y solicitar la reconstrucción del mismo.
	Memoria nombre-cara	Relación entre objetos y etiquetas asociadas a dichos objetos.	Mostrar la relación existente entre varios objetos y etiquetas. El jugador deberá memorizar esas relaciones y reconstruirlas.
Res. prob.	Aritmética	Realización de operaciones matemáticas sencillas	Enfrentar al jugador a gran cantidad de operaciones sencillas.
	Razonamiento lógico	Inducción de reglas lógicas a partir de casos concretos.	Mostrar elementos que cumplen o no una regla lógica para que el jugador la advine y la aplique a otros elementos.
Atención	Razonamiento cuantitativo	Realización de estimaciones y toma de decisiones en base a ellas.	Ofrecer grupos de elementos cuantificables para que el jugador elija rápidamente el de mayor o menor valor.
	Concentración	Fijación y dedicación de las actividades sensoriales a una única tarea.	Ofrecer grupos de elementos en los que el jugador tenga que prestar atención a alguna propiedad de uno sólo de ellos.
Velocidad	Campo visual	Espacio tridimensional visible por un individuo sin realizar movimientos oculares.	Mostrar y ocultar elementos en el espacio de juego. El jugador deberá realizar alguna tarea con las ubicaciones donde aparecían y su valor.
	Procesamiento de Información	Interpretación de la información sensorial. Adaptación al cambio.	Ofrecer una secuencia de elementos, algunos idénticos consecutivamente. El jugador debe decir si cada uno es igual al anterior o no.
Flexibilidad	Orientación espacial	Consciencia tridimensional de la situación de uno mismo en el espacio.	Mostrar elementos distribuidos en un espacio. Tras una transformación del mismo, el jugador deberá localizar dichas posiciones.
	Control de impulsos	Priorización de objetivos personales por encima de los impulsos animales.	Jugar con los colores y su representación textual. El jugador deberá poder prestar atención a un sólo aspecto: el color o el texto.
	Planificación	Priorización alterna de unas tareas sobre otras en un periodo temporal.	Ofrecer un espacio bidimensional maleable mediante cambios. El jugador construirá un camino usando el efecto de esos cambios.
Fluidez	Commutación de tareas	Intercambio de la atención entre ejercicios de forma arbitraria e intermitente.	Ofrecer situaciones diferentes, cada una con una misión. Las situaciones van cambiando, y en cada una el jugador debe proceder.
	Fluidez verbal	Recuperación rápida del vocabulario durante una conversación fluida.	Ofrecer las primeras o últimas letras de una palabra. El jugador deberá enumerar palabras que cumplen ese patrón.

Cuadro 6.1: Ficha de entrenamiento de capacidades mentales

6.3. Sistema de recomendación

6.3.1. Algoritmo de recomendación de usuarios

6.3.2. Algoritmo de recomendación de juegos

Capítulo 7

Conclusiones y propuestas

ESTE capítulo pretende ofrecer un análisis de los objetivos perseguidos durante el desarrollo del proyecto, estudiando en qué medida han sido alcanzados. Así mismo, pretende ofrecer una serie de propuestas futuras para completar el desarrollo y mejorar en todo lo posible algunos aspectos importantes, como la usabilidad, el rendimiento y la seguridad.

Finalmente se expone una valoración personal del autor sobre la experiencia que ha supuesto trabajar en la construcción de BreakBrain.

7.1. Objetivos alcanzados

En 2.2 se expusieron los objetivos, tanto generales como específicos, que se perseguían al embarcarse en el proyecto que supone el desarrollo de BreakBrain.

De forma muy general, la meta era la construcción de una plataforma social destinada al entrenamiento de habilidades mentales mediante la práctica de juegos, tanto individualmente como de forma colaborativa y competitiva. Se pretendía que la plataforma ofreciera estadísticas de evolución y permitiera el seguimiento (o suscripción a novedades) de otros usuarios, y que fuera extensible, de forma que cualquier desarrollador pudiera desarrollar sus propios juegos e integrarlos en el sistema.

También de forma muy general, el objetivo ha sido conseguido con creces, habiendo obtenido como resultado un proyecto maduro de software libre, disponible ya a día de hoy en la plataforma GitHub [Git], y cuyo desarrollo sigue en proceso.

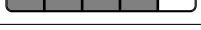
Realizando un análisis más detallado, los objetivos específicos en los que se desglosaba el objetivo general eran los siguientes:

- Afrontar el desarrollo de una red social desde cero, sin el uso de frameworks especializados, sino contemplando la creación de toda la plataforma paso a paso.
- Investigar y prever relaciones entre usuarios. Crear un sistema de recomendación basado en las expectativas de entrenamiento cerebral y resultados de cada usuario.
- Crear una plataforma extensible mediante un pequeño framework de desarrollo de videojuegos sencillos. El objetivo es que cualquier desarrollador pueda extender la

funcionalidad de BreakBrain mediante sus propios juegos.

- Liderar un proyecto de software libre ambicioso, gestionando la integración de juegos de terceros.
- Aprender a utilizar correctamente los nuevos estándares de HTML5, CSS3.
- Afrontar el desarrollo de aplicaciones web de tiempo real mediante el uso de WebSockets.
- Familiarizarse y ganar experiencia con el lenguaje JavaScript y su uso en el lado del servidor mediante NodeJS.
- Aprender a utilizar bases de datos documentales no basadas en el lenguaje SQL.

A continuación se valora el grado de consecución de todos estos objetivos. Para ello se especificará una valoración de entre 0 y 5 para cada uno (donde 0 significa que el objetivo no se ha alcanzado de ninguna manera y 5 que el objetivo ha sido alcanzado completamente). Un valor intermedio de 3, por ejemplo, implica que el objetivo no ha sido alcanzado del todo, pero que el desarrollo está muy cerca de hacerlo. El valor 4 implica una consecución prácticamente completa del objetivo (a falta de pequeños detalles mejorables).

Objetivo específico	Consecución
Desarrollar una red social desde cero	
Soportar el desarrollo e integración de juegos de terceros	
Crear un sistema de recomendación de usuarios y juegos	
Liderar un proyecto de software libre ambicioso	
Aprender a utilizar los nuevos estándares HTML5 y CSS3	
Tiempo real en aplicaciones web mediante WebSockets	
Aprender JavaScript y NodeJS	
Aprender a utilizar bases de datos documentales NoSQL	

Cuadro 7.1: Grado de consecución de objetivos específicos

7.2. Propuestas de trabajo futuro

Pese a todo el esfuerzo dedicado al desarrollo de BreakBrain, obviamente una única persona en un tiempo limitado no puede afrontar todos y cada uno de los objetivos al 100 %. Por ello desde el primer momento se pretendió que del desarrollo del mismo se obtuviera un proyecto de software libre ambicioso, con vistas a continuar su desarrollo tanto por parte del autor del presente documento como por la comunidad de desarrolladores de software libre. Así pues, la primera y más importante propuesta de futuro es asegurar el cumplimiento riguroso de todos los requisitos que no hayan sido finalizados en su totalidad.

Yendo algo más lejos, a continuación se ofrecen algunas posibles mejoras en diferentes ámbitos (como la seguridad, la usabilidad o el rendimiento). Las siguientes propuestas tienen como objetivo la maduración de BreakBrain como plataforma social y de entrenamiento cerebral.

■ **Ampliar el catálogo de juegos**

El presente documento, así como el estado del proyecto en el momento de su publicación, pretende mostrar el funcionamiento del framework de integración de juegos para la plataforma. Es por ello que, aunque todos los tipos de juegos a desarrollar han sido especificados detalladamente, sólo una pequeña cantidad de juegos han sido implementados. En un futuro cercano, una de las primeras metas es ofrecer al menos un juego para cada habilidad mental.

■ **Enriquecer las relaciones sociales en BreakBrain**

En el momento de publicar este documento, las relaciones sociales entre usuarios de la red social se limitan a la suscripción a las novedades de otros usuarios (lo que se ha denominado *seguir* a usuarios) y la participación conjunta en el entrenamiento cerebral, mediante el uso de juegos multijugador. Sería muy positivo en un futuro enriquecer estas relaciones, de forma que pueda haber grupos o listas de usuarios a los que se sigue, por ejemplo. Esto daría pie a un aumento de la información útil para los algoritmos de recomendación integrados.

■ **Mejorar el aspecto de la web y adaptar el diseño a dispositivos móviles**

Aunque BreakBrain hace un uso intensivo de JavaScript mediante la implementación de los juegos —por lo que en principio es más adecuado para su utilización en computadores de escritorio—, son cada vez mayores los avances en cuanto al rendimiento de HTML5 y JavaScript en las plataformas móviles más conocidas: iOS y Android.

En un futuro será conveniente que BreakBrain ajuste su aspecto a todo tipo de tamaños y resoluciones de pantalla, por ejemplo mediante el uso de frameworks que faciliten el *responsive design*, como Bootstrap.

■ **Construir una app móvil**

En las plataformas móviles más populares existe un cierto capado en el rendimiento web, que dificulta la construcción de aplicaciones basadas únicamente en componentes y código que sea ejecutado en el navegador. El objetivo de esta limitación es llevar a los desarrolladores a decantarse por el código nativo siempre, de forma que el control de las apps siga estando en manos de los gestores de las tiendas de aplicaciones (Google, Apple y Amazon, entre otros).

En algún momento resultaría conveniente crear una app de BreakBrain naiva para las principales plataformas, iOS y Android. Obviamente tendrán que tomarse decisiones

importantes de diseño, y los juegos tendrán que ser reescritos o convertidos mediante alguna herramienta generadora de código nativo.

■ **Ampliación del conjunto de pruebas**

Como en todo software de calidad que se precie, la existencia de más pruebas debe ser siempre un objetivo de futuro.

La elaboración de tests de integración, posiblemente acompañada de un sistema automático de despliegue, sería muy beneficioso cuando haya más de un individuo encargado de la liberación de *releases*.

■ **Utilizar HyperText Transfer Protocol Secure (HTTPS)**

Aunque en estos momentos BreakBrain no maneja datos sensibles, resultará conveniente empezar a utilizar un protocolo de transferencia seguro, HTTPS. Más que un protocolo diferente de HTTP, en realidad es lo mismo pero añadiendo una capa de cifrado Secure Socket Layer (SSL) por encima. Con HTTPS los mensajes HTTP son cifrados antes de la transmisión y descifrados después de la recepción para ser interpretados.

Ahora que BreakBrain es un proyecto de software libre, el cumplimiento de las anteriores propuestas —así como el surgimiento de nuevas— es mucho más que un mero deseo o idea, ya que toda una comunidad de desarrolladores está en disposición de aportar su granito de arena para conseguirlo.

7.3. Opinión personal

Tras muchos meses de trabajo invertido en el desarrollo de BreakBrain, no puedo sentir otra cosa que satisfacción. Son muy numerosas las dificultades encontradas al afrontar un proyecto de tal embergadura, pero dichas dificultades han servido —y sirven— para aprender mucho sobre muy diversos aspectos del desarrollo de software. Pero en este caso concreto, además, el hecho de perseguir un objetivo tan ambicioso como lo es el entrenamiento y mejora de las habilidades mentales del ser humano ha servido como motivante para estudiar la fisiología del cerebro y comprender así, siempre a un nivel muy básico, cómo funciona nuestro órgano más importante.

Desde un punto de vista académico, desarrollar BreakBrain ha contribuido de una forma sorprendente a mejorar mis competencias técnicas. He explorado campos que durante la carrera no se estudian, como el desarrollo web de tiempo real o el manejo de bases de datos documentales, al tiempo que he asimilado en profundidad otros aspectos que durante la carrera sólo se estudian de forma superficial. Me ha servido para aprender tecnologías nuevas que están ganando popularidad cada día, como NodeJS o MongoDB, y para familiarizarme con servicios como Amazon Web Services (AWS), cuyo conocimiento es valorado

por muchas empresas actualmente. En resumen, gracias a BreakBrain soy una persona más competente profesionalmente.

Este Proyecto Fin de Carrera sólo es la cima de una gran montaña, la guinda de un pastel elaborado con esfuerzo y sacrificio: la carrera de Ingeniería Informática. Después de todo lo aprendido no puedo sentir otra cosa que orgullo por todo el trabajo desarrollado, así como por el resultado obtenido tras estos años. Jamás he sentido una satisfacción mayor que aprendiendo.

ANEXOS

Anexo A

Iteraciones del PUD

Este anexo pretende ofrecer un análisis descriptivo de las diferentes iteraciones por las que ha pasado el presente proyecto durante su desarrollo, siguiendo para ello el PUD, como se comentó en la sección 4.1.

A continuación se hará un recorrido por cada iteración, explicando en qué han consistido y detallando la repercusión global de cada una. Las iteraciones están agrupadas en tres fases: Inicio, Elaboración y Construcción.

Cada iteración es detallada desde 2 puntos de vista:

■ Local

Con este enfoque se ofrece una descripción de en qué ha consistido la iteración. Además se detalla la proporción del desarrollo general que ha supuesto. Es importante comprender que las cifras de cada iteración no representan la repartición del esfuerzo de esa tarea, sino el porcentaje de cada artefacto que ha sido completado en esa iteración.

■ Global

Con este enfoque se muestran los porcentajes completado y restante del desarrollo global, desde el inicio de la primera iteración hasta la finalización de la iteración en cuestión.

Al finalizar este anexo se ofrecerá una descripción gráfica de la evolución seguida durante el proceso.

A.1. Iteraciones de inicio

Iteración I1

■ Descripción:

Esta primera iteración se ha basado en la elaboración del análisis de un conjunto inicial de requisitos —los más importantes— así como el estudio de casos de uso de una parte de los mismos.

Al mismo tiempo que parte de los requisitos quedaban especificados por completo, una pequeña parte del diseño del software a construir quedaba definido.

Los artefactos que se han visto involucrados en esta iteración son, por lo tanto, el análisis de requisitos, en análisis de casos de uso y el diseño.

Esta iteración ha tenido una duración de 75 horas aproximadamente.

■ **Desarrollo local:**

- Requisitos: 53.6 %.
- Casos de uso: 27.8 %.
- Diseño: 10 %.
- Implementación: 0 %.
- Pruebas: 0 %.

■ **Desarrollo global: 18.28 %**

■ **Desarrollo restante: 81.72 %**

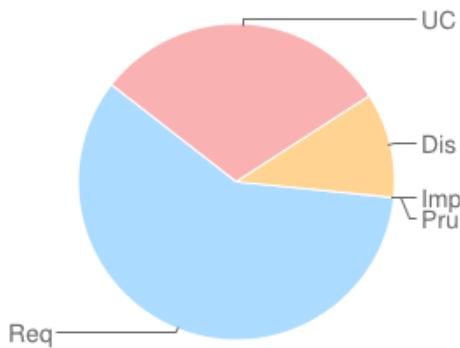


Figura A.1: Iteración II

A.2. Iteraciones de elaboración

Iteración E1

■ **Descripción:**

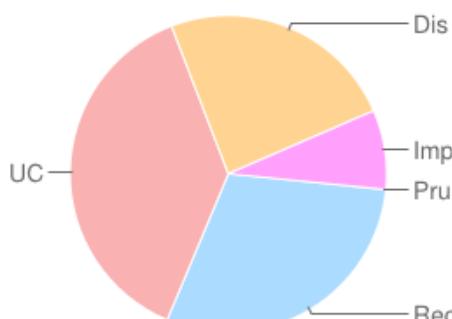
Esta primera iteración de la fase de elaboración ha supuesto los primeros pasos de la implementación —en una proporción modesta—, al tiempo que los análisis de requisitos y de casos de uso han sido ampliados.

Los artefactos que se han visto involucrados en esta iteración son, por lo tanto, el análisis de requisitos, en análisis de casos de uso, el diseño y la implementación.

Esta iteración ha tenido una duración de 50 horas aproximadamente.

■ **Desarrollo local:**

- Requisitos: 17.8 %.
- Casos de uso: 22.2 %.
- Diseño: 15 %.
- Implementación: 5.2 %.
- Pruebas: 0 %.



■ **Desarrollo global: 30.3 %**

■ **Desarrollo restante: 69.7 %**

Figura A.2: Iteración E1

Iteración E2

■ Descripción:

Esta última etapa de la fase de Elaboración ha supuesto un pequeño avance generalizado. La mayor parte de los esfuerzos se han dedicado al diseño del software. Además se han creado las primeras pruebas unitarias.

Todos los artefactos se han visto involucrados en esta iteración: el análisis de requisitos, en análisis de casos de uso, el diseño, la implementación y las pruebas.

Esta iteración ha tenido una duración de 50 horas aproximadamente.

■ Desarrollo local:

- Requisitos: 14.3 %.
- Casos de uso: 27.8 %.
- Diseño: 50 %.
- Implementación: 12.1 %.
- Pruebas: 5 %.

■ Desarrollo global: 52.2 %

■ Desarrollo restante: 47.8 %

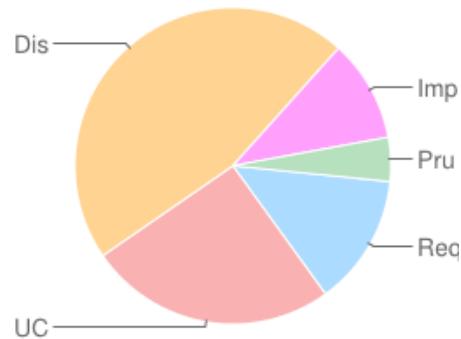


Figura A.3: Iteración E2

A.3. Iteraciones de construcción

Iteración C1

■ Descripción:

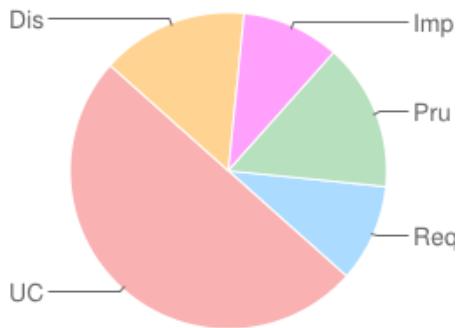
En esta iteración se ha realizado un pequeño esfuerzo por prácticamente terminar de definir los casos de uso. Dado que en la etapa de construcción el objetivo es centrarse en la implementación, en esta primera iteración se ha pretendido dejar prácticamente finalizados los artefactos previos a la misma.

En esta iteración están implicados todos los artefactos: análisis de requisitos, en análisis de casos de uso, diseño, implementación y pruebas.

Esta iteración ha tenido una duración de 75 horas aproximadamente.

■ **Desarrollo local:**

- Requisitos: 3.6 %.
- Casos de uso: 16.6 %.
- Diseño: 5 %.
- Implementación: 2.9 %.
- Pruebas: 5 %.



■ **Desarrollo global: 58.8 %**

■ **Desarrollo restante: 41.2 %**

Figura A.4: Iteración C1

Iteración C2

■ **Descripción:**

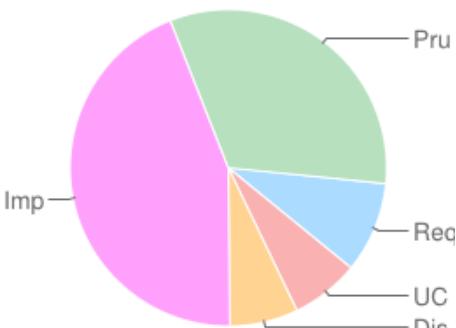
Una vez el proyecto se ha centrado en la etapa de construcción, en esta iteración se realiza un gran avance en la implementación del sistema software. El análisis de requisitos queda prácticamente terminado, a excepción de algún pequeño detalle —requisitos sencillos que surgieron a medida que el desarrollo avanzaba—, y el análisis de casos de uso se finaliza por completo.

En esta iteración están implicados todos los artefactos: análisis de requisitos, en análisis de casos de uso, diseño, implementación y pruebas, con un peso muy importante para la implementación.

Esta iteración ha tenido una duración de 125 horas aproximadamente.

■ **Desarrollo local:**

- Requisitos: 7.1 %.
- Casos de uso: 5.6 %.
- Diseño: 5 %.
- Implementación: 30.6 %.
- Pruebas: 23.4 %.



■ **Desarrollo global: 73.1 %**

■ **Desarrollo restante: 26.9 %**

Figura A.5: Iteración C2

Iteración C3

■ **Descripción:**

Continuando con un esfuerzo mayor para la implementación, esta iteración preten-

de además avanzar de forma importante con el diseño. Las pruebas son mejoradas y ampliadas.

En esta iteración están implicados los artefactos de análisis de requisitos, diseño, implementación y pruebas, con un peso muy importante, como ya se ha mencionado, para las pruebas.

Esta iteración ha tenido una duración de 80 horas aproximadamente.

■ Desarrollo local:

- Requisitos: 3.6 %.
- Casos de uso: 0 %.
- Diseño: 15 %.
- Implementación: 12.3 %.
- Pruebas: 25.7 %.

■ Desarrollo global: 82.4 %

■ Desarrollo restante: 17.6 %

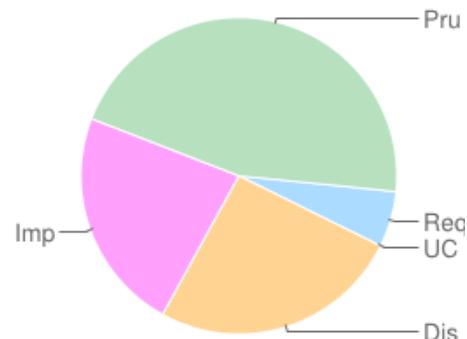


Figura A.6: Iteración C3

Iteración C4

■ Descripción:

En esta última iteración de la fase de construcción los artefactos protagonistas son la implementación y las pruebas. Los análisis de requisitos y de casos de uso quedaron finalizados. El diseño queda en una fase muy avanzada, a falta de pulir pequeños detalles.

En esta iteración están implicados los artefactos de diseño, implementación y pruebas.

Esta iteración ha tenido una duración de 70 horas aproximadamente.

■ Desarrollo local:

- Requisitos: 0 %.
- Casos de uso: 0 %.
- Diseño: 5 %.
- Implementación: 11.9 %.
- Pruebas: 11.9 %.

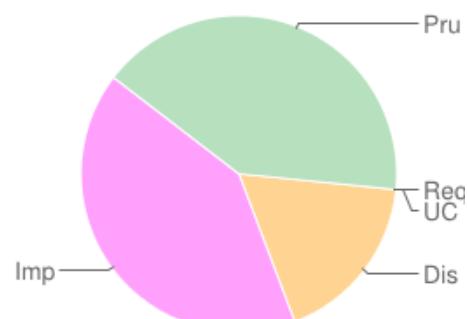


Figura A.7: Iteración C4

A.4. Etapa de transición

La etapa de transición del desarrollo de BreakBrain no ha sido abarcada en este documento. Se espera completarla como plan de futuro. Actualmente el proyecto se encuentra en una fase avanzada, pero a falta de terminar algunos detalles antes de poder ponerlo a disposición de los usuarios.

A.5. Visión gráfica

A continuación se representa gráficamente la proporción de desarrollo de cada artefacto software en cada una de las iteraciones. Nótese la clara relación con la figura 4.1, en la que se mostraba la evolución teórica de dichos artefactos según el PUD.

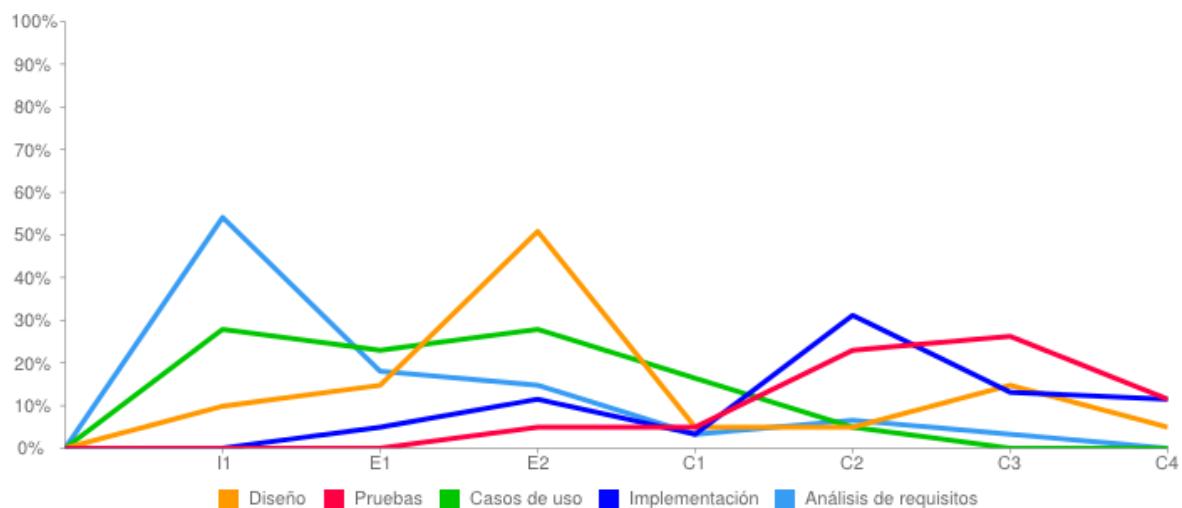


Figura A.8: Evolución iterativa del desarrollo

Anexo B

Guía de instalación

B.1. Requisitos previos

Para poder ejecutar BreakBrain el primer paso es instalar (si no se encuentran en nuestro sistema) el entorno de ejecución NodeJS y el Sistema Gestor de Base de Datos (SGBD) MongoDB.

B.1.1. NodeJS

La instalación de NodeJS es realmente sencilla. En la web oficial (<http://nodejs.org>) hay disponibles para descarga multitud de instaladores para diferentes sistemas operativos. Encuentre el adecuado para su sistema y siga los pasos de la instalación.

Alternativamente, y gracias a que NodeJS es software libre, es posible descargar el código fuente y compilarlo de forma manual. Por último, en caso de que nuestro sistema sea GNU/Linux, es posible simplificar al máximo la tarea mediante la utilización de un gestor de paquetes; por ejemplo, en el caso de Ubuntu:

```
# apt-get install nodejs
```

Una vez instalado NodeJS, desde un intérprete de comandos será posible comprobar que todo ha ido bien, por ejemplo viendo qué versión del binario de NodeJS está disponible en el PATH:

```
# node -v  
v0.8.5
```

B.1.2. MongoDB

Una vez que NodeJS se encuentre correctamente instalado en nuestro sistema, el siguiente paso es instalar el SGBD.

Al igual que en el caso anterior, la tarea es realmente sencilla siguiendo los pasos de la documentación oficial de MongoDB y usando alguno de los instaladores disponibles en

```
http://www.mongodb.org/
```

Como siempre, si disponemos de GNU/Linux las cosas son aún más sencillas, gracias a los gestores de paquetes. En el caso de Ubuntu:

```
# apt-get install mongodb
```

Una vez instalado, es posible comprobar que el binario se encuentra en el PATH del sistema utilizando un intérprete de comandos para comprobar qué versión se encuentra instalada:

```
$ mongo -v
MongoDB shell version: 2.2.3
```

B.2. Instalación de BreakBrain

Una vez que NodeJS y MongoDB se encuentran perfectamente instalados en el sistema, se estará en disposición de instalar y ejecutar BreakBrain. Para ello se deben seguir los siguientes pasos:

1. Descargar el código fuente del repositorio oficial (<https://github.com/sgmonda/breakbrain>). Si se dispone de GIT instalado, se puede ejecutar el siguiente comando:

```
$ git clone https://github.com/sgmonda/breakbrain
```

En caso contrario bastará con descargar el siguiente ZIP y descomprimirlo a mano:

```
https://github.com/sgmonda/breakbrain/archive/master.zip
```

2. Abrir un intérprete de comandos y desplazarse al directorio extraído tras la descompresión:

```
$ cd breakbrain
```

3. Instalar las dependencias de BreakBrain, utilizando el gestor de paquetes NPM:

```
$ npm install
```

4. Si todo ha ido bien, ejecutar BreakBrain ahora es realmente sencillo:

```
$ node server.js
Tue, 30 Jul 2013 22:48:45 (GMT) :: DATABASE: Connecting to MongoDB...
Tue, 30 Jul 2013 22:48:46 (GMT) :: DATABASE: Connected and authent...
Tue, 30 Jul 2013 22:48:46 (GMT) :: EMAIL-MODULE: Email subsystem r...
Tue, 30 Jul 2013 22:48:46 (GMT) :: GAMES-MODULE: Loading games ser...
Tue, 30 Jul 2013 22:48:46 (GMT) :: WEB SERVER: running on port 206...
Tue, 30 Jul 2013 22:48:46 (GMT) :: GAMES-MODULE: Loading game "Un ...
Tue, 30 Jul 2013 22:48:47 (GMT) :: WEBSOCKETS SERVER: running on p...
Tue, 30 Jul 2013 22:48:47 (GMT) :: MAIN SERVER: The whole server i...
```

Para ejecutar en modo *test*, basta con añadir el flag --test:

```
node server.js --test
```

5. En este punto ya es posible utilizar un navegador web para acceder a la copia local de BreakBrain en ejecución, accediendo a <http://localhost:20661>. La página de login se mostrará como en la figura B.1.

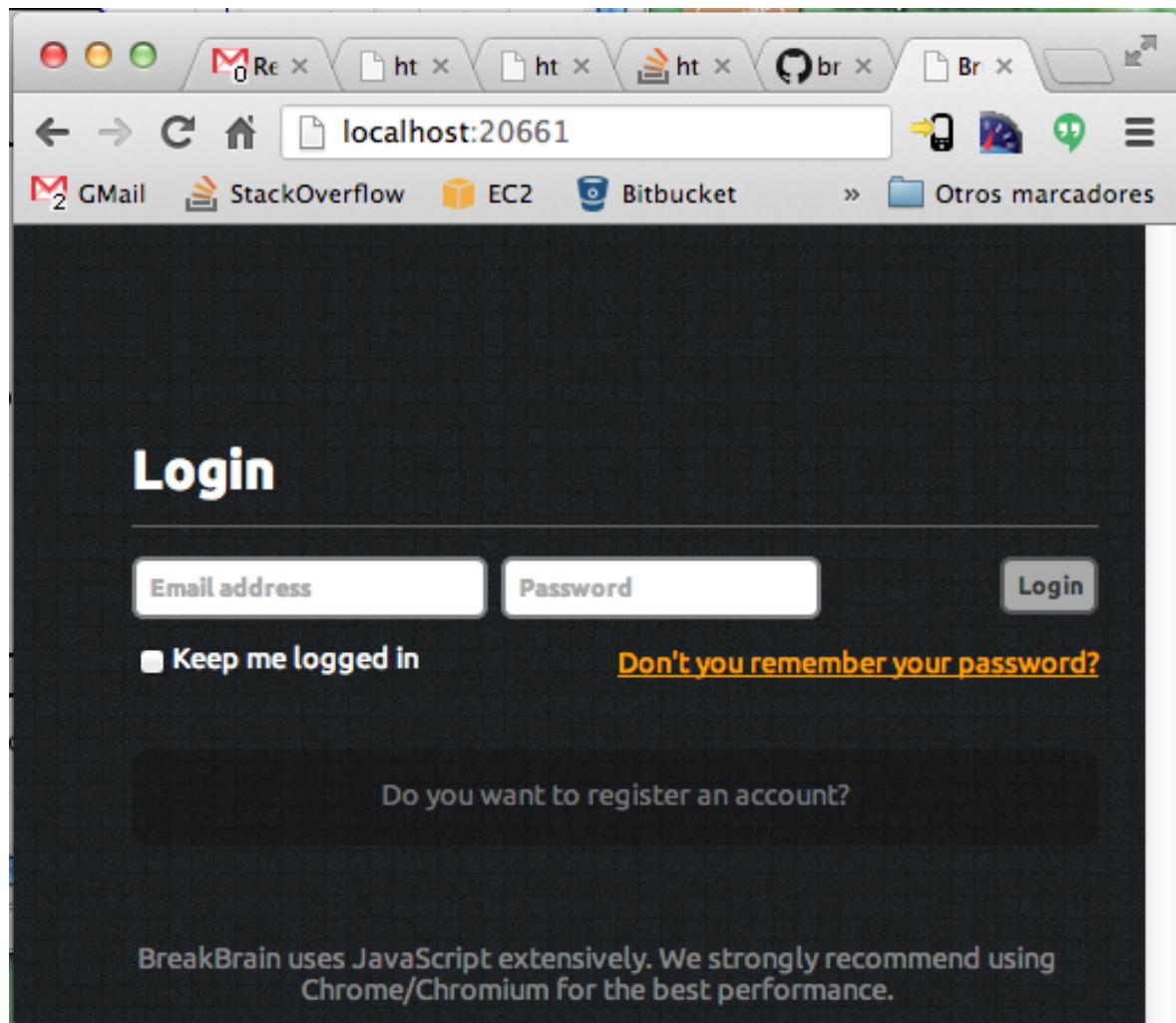


Figura B.1: Página de autenticación de BreakBrain

Anexo C

Guía para el desarrollador: creación de juegos para BreakBrain

ESTE apéndice pretende ser una guía completa para desarrolladores. A lo largo del mismo se detallará el funcionamiento interno del subsistema de juegos de BreakBrain y se explicará, de forma clara y utilizando ejemplos prácticos, el procedimiento de creación e integración de juegos de terceros en BreakBrain.

Tras la lectura de esta guía, el desarrollador estará en disposición de extender la funcionalidad de BreakBrain mediante sus propios juegos, ya sean monojugador o multijugador, posibilitando que cualquier usuario de la red social haga uso de ellos.

C.1. Requisitos

Antes de poder desarrollar un juego para BreakBrain existen algunos requisitos que el desarrollador debe satisfacer. Suponen la posesión de una serie de conocimientos, habilidades y herramientas necesarias para desarrollar juegos HTML5 2D compatibles con la plataforma.

C.1.1. Conocimientos del desarrollador

Para desarrollar juegos para BreakBrain es necesario:

- Conocimiento avanzado del lenguaje JavaScript.
- Experiencia desarrollando con NodeJS.
- Experiencia en desarrollo de gráficos 2D.
- Conocimiento avanzado del canvas de HTML y manejo fluido del DOM.

Además, resulta recomendable contar con:

- Experiencia utilizando el framework de juegos 2D CreateJS.
- Experiencia en el desarrollo de aplicaciones web de tiempo real.
- Experiencia en el manejo de websockets.

C.1.2. Herramientas de desarrollo

En cuanto a las herramientas necesarias para desarrollar juegos para BreakBrain, encontramos las siguientes:

- Editor de texto plano: puede servir cualquier editor de texto, aunque es recomendable contar con algún editor avanzado que ofrezca resaltado de sintaxis y facilidades para la programación. Se recomienda GNU Emacs, pero puede utilizar otras soluciones, como VIM, Sublime Text, Gedit, etc.
- Navegador web: necesario para depurar y probar los juegos en desarrollo. Resulta muy importante que el navegador que utilicemos disponga de buenas herramientas para facilitar la depuración, así como de un buen motor de JavaScript que permita sacarle todo el potencial gráfico a la máquina. Se recomienda utilizar Chromium/Chrome o Mozilla Firefox.
- Terminal de línea de comandos para ejecutar NodeJS: este *runtime* es el utilizado para ejecutar la parte servidora de los juegos, por lo que resulta esencial disponer de él para asegurar el correcto funcionamiento del juego.

Aunque no es necesario, además se recomienda disponer de:

- Sistema Operativo de tipo UNIX: para el desarrollo y la ejecución de los juegos es altamente recomendable un entorno UNIX, como GNU/Linux, BSD o Mac OS X.

C.2. Tipos de juegos

Cada juego de la plataforma pertenece y está diseñado para estimular a una única habilidad mental (ver tabla 6.2.3). Por ello, antes de crear un juego el primer paso es decidir a qué habilidad mental estará dirigido. Independientemente de ello, el segundo paso será decidir el tipo de juego: en BreakBrain hay dos tipos de juegos, en función del número de jugadores que puedan participar en una misma partida: juegos monojugador y juegos multijugador.

C.2.1. Juegos monojugador

Se trata de juegos diseñados para ser utilizados por un único usuario, por lo que no supone ningún tipo de interacción social. En este tipo de juegos no se gana o se pierde, sino que el resultado es un valor cuantitativo que indica el grado de éxito de la partida, basado en la puntuación obtenida en cada actividad puntuable de la misma.

Para comprender mejor la relación entre los conceptos de juego, partida y actividad puntuable, ver sección 6.2.1.

C.2.2. Juegos multijugador

En este caso se trata de juegos diseñados para ser utilizados por dos jugadores a la vez, suponiendo un enfrentamiento entre ellos. Sólo un jugador de los dos puede ganar cada ronda o partida. No obstante, el progreso del entrenamiento cerebral de ambos se verá afectado por el resultado que han obtenido en las actividades puntuables completadas durante la partida, independientemente de quién gane.

Para comprender mejor la relación entre los conceptos de juego, partida y actividad puntuable, ver sección 6.2.1.

C.3. El sistema de juegos de BreakBrain

Llegados a este punto resulta conveniente comprender la composición del subsistema de juegos de la plataforma, así como de cada juego de la misma.

C.3.1. Componentes del sistema de juegos

El sistema de juegos de BreakBrain se compone de tres elementos, o conjuntos de elementos, bien diferenciadas:

- Cargador de juegos
- Partes servidoras de los juegos
- Partes cliente de los juegos

El cargador de juegos es la piedra angular del subsistema, ya que se encarga de chequear los juegos disponibles y ponerlos a disposición del usuario. Una vez instalado correctamente un nuevo juego en una instancia de BreakBrain, el cargador se encargará de analizar sus componentes, arrancar la parte servidora del mismo y poner a disposición de la web la parte cliente, para que cualquier usuario pueda usarlo. Todo ello tiene lugar de forma automática.

Las partes servidoras de los juegos se encuentran en el lado del servidor. El cargador de juegos arranca dichas partes servidoras en el proceso de inicio del servidor, para que los juegos estén disponibles y las partes cliente puedan comunicarse con las partes servidoras.

Las partes cliente de los juegos se cargan de forma dinámica al abrir la web de BreakBrain. Son más flexibles, en el sentido en el que no requieren de un reinicio del servidor para funcionar o manifestar cambios.

C.3.2. Componentes de un juego

Los juegos de BreakBrain se componen de 2 partes principales: un pequeño servidor (un único script JS) y un cliente (un directorio con un script JS y recursos).

Parte servidora

La parte servidora de los juegos es ejecutada en un servidor NodeJS. Se trata de la espina dorsal sobre la que el juego se construye. En el caso de juegos monojugador, este componente es muy sencillo, y se limita a almacenar el resultado de las partidas mediante llamadas al núcleo del servidor de BreakBrain. En el caso de partidas multijugador, el papel es aún más importante, encargándose de la lógica del juego, la comunicación con la parte cliente de los jugadores y manteniendo la sincronización constante.

Tal y como BreakBrain está concebido, las tareas mencionadas resultan realmente sencillas: la conexión con los clientes se gestiona de forma automática, sin que el desarrollador tenga que preocuparse por ello. La comunicación resulta trivial. La lógica principal del juego dependerá de qué clase de juego se implemente.

En la sección C.4 se detalla el proceso de creación de un juego desde cero, detallando el comportamiento de la parte servidora de un juego.

Parte cliente

La parte cliente de los juegos es ejecutada por el navegador web. La experiencia de usuario puede variar, por lo tanto, dependiendo del navegador web del usuario de BreakBrain. El papel de la parte cliente de los juegos es gestionar la representación gráfica del estado y evolución de los mismos. En juegos monojugador, además, la lógica del juego puede estar implementada en este componente (dado que no se comparte con otros usuarios en tiempo de ejecución).

En la sección C.4 se detalla el proceso de creación de un juego desde cero, detallando el comportamiento de la parte cliente de un juego.

Recursos

Los recursos multimedia requeridos por los juegos (sprites, archivos de audio, etc.) se localizan en la parte cliente —dado que es la que los utiliza—. Como mínimo un fondo o background y un ícono son obligatorios por juego.

C.4. Creación de un juego desde cero

En esta sección se realizará un recorrido práctico por el proceso de creación de un juego. Para ello se creará un juego completo desde cero.

C.4.1. Eligiendo una categoría

El primer paso a la hora de crear un juego para BreakBrain es elegir la categoría a la que se quiere que pertenezca, es decir, la capacidad y habilidad mental que se quiere entrenar. En BreakBrain cada juego se orienta a la estimulación de una única habilidad mental

(perteneciente a una única categoría).

A continuación se listan las capacidades en las que el entrenamiento cerebral se divide, así como las habilidades mentales que componen cada una de ellas. Cuando durante el desarrollo se especifiquen categorías o habilidades, se hará siempre mediante los identificadores asociados a cada una de ellas (indicados entre paréntesis en la siguiente lista):

- Memoria (MEM)
 - Memoria de trabajo (MEM-WORK)
 - Memoria espacial (MEM-SPACE)
 - Memoria nombre-cara (MEM-FN)
- Resolución de problemas (PRO)
 - Aritmética (PRO-ARIT)
 - Razonamiento lógico (PRO-LOG)
 - Razonamiento cuantitativo (PRO-QUAN)
- Atención (ATT)
 - Concentración (ATT-FOCUS)
 - Campo visual (ATT-VFIELD)
- Velocidad (SPD)
 - Procesamiento de información (SPD-INF)
 - Orientación espacial (SPD-SPA)
- Flexibilidad (FLE)
 - Control de impulsos (FLE-INRES)
 - Planificación (FLE-PLAN)
 - Conmutación de tareas (FLE-SWT)
 - Fluidez verbal (FLE-VER)

En la tabla 6.2.3 se ofrece una sencilla descripción de cada habilidad mental, junto a una explicación del funcionamiento de un juego basado en ella.

En esta guía se seguirán los pasos necesarios para crear un juego orientado a mejorar la habilidad **Memoria nombre-cara** de la capacidad **Memoria**. El juego se llamará, por ejemplo, “Careto”.

	de trabajo	durante el desarrollo de una tarea.	y después requeridas de algún modo.
Memoria	Memoria espacial	Información del entorno, localización de objetos y orientación en el espacio.	Ofrecer un paisaje de objetos para su memorización y solicitar la reconstrucción del mismo.
	Memoria nombre-cara	Relación entre objetos y etiquetas asociadas a dichos objetos.	Mostrar la relación existente entre varios objetos y etiquetas. El jugador deberá memorizar esas relaciones y reconstruirlas.
	Aritmética	Realización de operaciones matemáticas sencillas de forma rápida y correcta.	Enfrentar al jugador a gran cantidad de operaciones sencillas de suma, resta, multiplicación y división.

El juego ofrecerá al jugador una serie de objetos con una etiqueta asociada durante unos segundos. Después se mostrarán los objetos sin etiquetas, y el jugador deberá especificar la etiqueta de cada uno de ellos. Las etiquetas que serán asociadas a cada objeto no tendrán relación semántica alguna con él, para evitar que el conocimiento previo a la partida pueda influir en el resultado de la misma.

C.4.2. Eligiendo el tipo de juego

Los juegos para BreakBrain pueden ser monojugador o multijugador. Ambos tipos de juegos servirán del mismo modo al entrenamiento cerebral, pero en el caso de los juegos multijugador entrará en juego, además, el aspecto social y motivante del enfrentamiento a otro usuario.

En este caso se optará por la creación de un juego multijugador. Ambos jugadores serán enfrentados a la misma escena, para asegurar la igualdad de condiciones, y cuando uno la consiga resolver ambos pasarán a la siguiente. Nótese que este tipo de decisiones dependen del desarrollador, y en este mismo caso podría decidirse enfrentar a los dos jugadores a diferentes escenas, independizando el progreso de cada uno en la partida.

C.4.3. Creando la estructura básica del juego

¡Manos a la obra! Es hora de empezar a crear el juego. En este paso es totalmente recomendable tener instalado BreakBrain en local, para poder depurar el juego durante el desarrollo, sin tener que esperar a terminarlo. Así pues, suponiendo que tenemos una copia funcionando en el directorio \$BREAKBRAIN, tendremos que seguir los siguientes pasos para crear la estructura del juego:

1. Crear el script de servidor. Por seguir la nomenclatura de la plataforma habrá que llamarlo de la forma “<nombreEnCamelCase>-server.js”. Dicho script debe situarse en el directorio de partes servidoras de juegos (\$BREAKBRAIN/server/games). En nuestro caso:

```
$BREAKBRAIN/server/games/careto-server.js
```

2. Ahora se debe crear el directorio de la parte cliente del juego. Se sigue la misma nomenclatura: “<nombreEnCamelCase>-client”. La ubicación adecuada es el directorio donde se encuentran las partes clientes de los juegos: \$BREAKBRAIN/public/games. En nuestro caso, el directorio a crear es el siguiente:

```
$BREAKBRAIN/public/games/careto-client
```

3. Crear el script cliente del juego (llamado `main.js`) en su directorio cliente. En nuestro caso, el script se sitúa en:

`$BREAKBRAIN/public/games/careto-client/main.js`

4. Especificar un ícono y una imagen de fondo para el juego. Estas dos imágenes deben ubicarse en el directorio de la parte cliente del juego, y se llamarán `logo.png` y `background.png`, respectivamente. El logo debe tener un tamaño de 125x125 pixels, y el fondo de 600x480 pixels. En nuestro caso, ambos archivos son los siguientes:

`$BREAKBRAIN/public/games/careto-client/logo.png`



Figura C.1: Logo del juego “Careto”

`$BREAKBRAIN/public/games/careto-client/background.png`

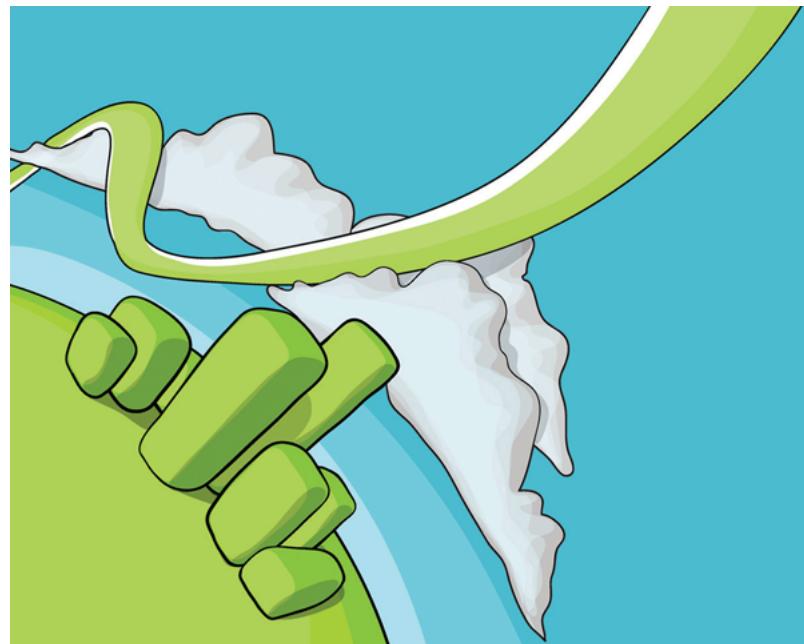


Figura C.2: Fondo del juego “Careto”

C.4.4. Preparando los recursos

Cualquier recurso multimedia empleado por el juego deberá situarse en el directorio de su parte cliente. En nuestro caso:

`$BREAKBRAIN/public/games/careto-client`

Para “Careto” utilizaremos una serie de imágenes (a las que asociaremos una etiqueta durante la ejecución). La etiqueta será textual, por lo que no necesita ningún recurso externo. Las imágenes, por otro lado, deberán ser incluidas como recurso externo. En este caso se emplearán las letras del alfabeto Braille como imágenes a las que asociar las etiquetas (ver figura C.3).

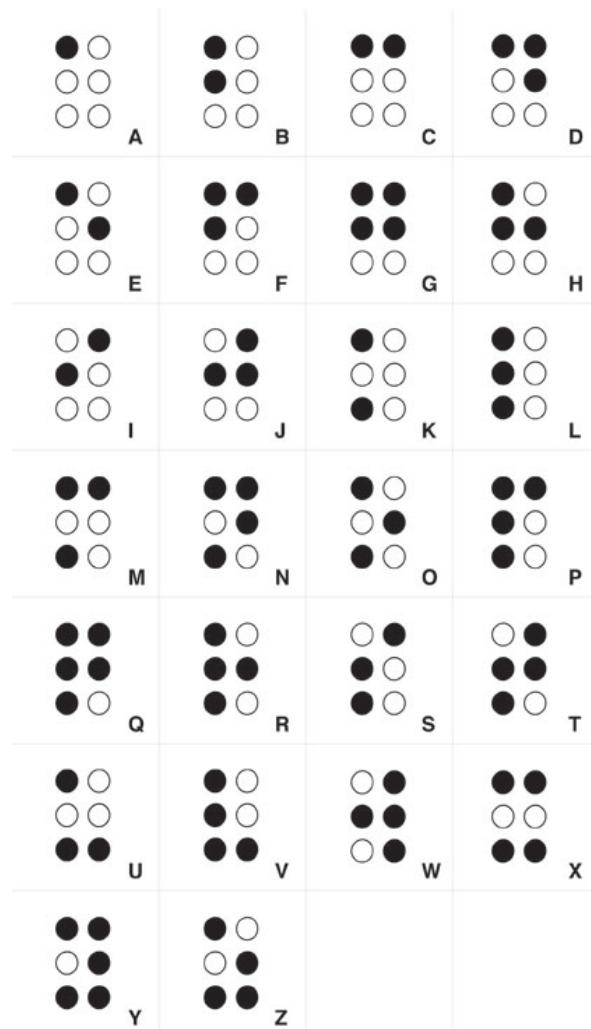


Figura C.3: Imágenes de juego para “Careto”

Llegados a este punto, la estructura y recursos de “Careto” están perfectamente definidas y ubicadas en su localización correcta:

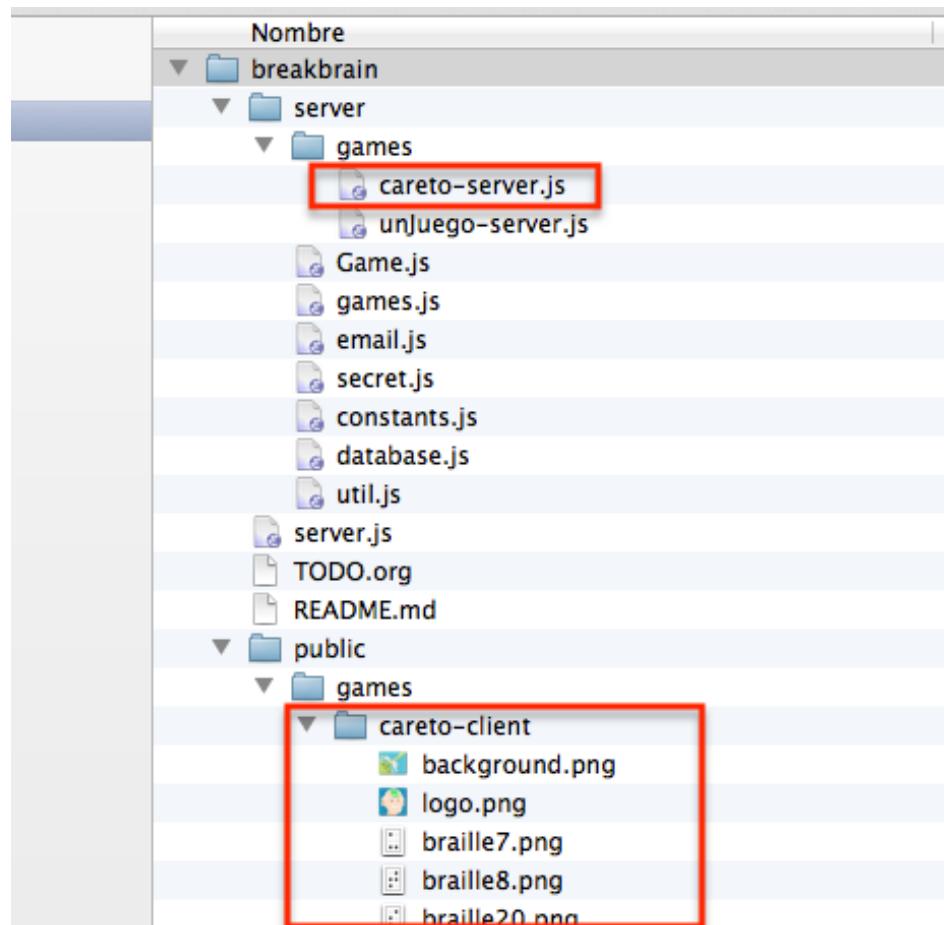


Figura C.4: Jerarquía de directorios para “Caret”

C.4.5. Desarrollando la parte servidora

El esquema de la parte servidora de un juego de BreakBrain es realmente sencillo, y es el mostrado en el listado C.1. Basta con importar la clase Game y exportar una función que devuelva una instancia de dicha clase. La función puede recibir 2 argumentos que BreakBrain pasa automáticamente:

- Una función de test que se puede utilizar para realizar tests unitarios del juego.
- La URL esperada de la parte cliente (esta debe pasarse a su vez a la instancia de Game).

La recepción de estos dos argumentos es opcional, pero recomendable. La función de test es realmente útil para ejecutar tests unitarios. Por otro lado, la ruta de la parte cliente puede establecerse a mano, pero no hay razón para ello si se siguen las directivas establecidas de ubicación y nomenclatura.

```

1 // Clase Game, de la que todos los juegos heredan
2 var Game = require('../Game.js');
3
4 // Objeto exportado segun la especificacion de modulos para NodeJS
5 module.exports = function (test, clientURL) {
6
7     var g = new Game(
8         '<nombre>', // Nombre del juego
9         '<capacidad>', // Capacidad a estimular
10        '<habilidad>', // Habilidad mental
11        clientURL // URL de la parte (auto)
12    );
13
14    // Puede utilizar el framework de pruebas incluido
15    // en BreakBrain para realizar test unitarios:
16    //
17    // test('Nombre del test', <condicion>, 'mensaje de fallo');
18
19    setInterval(function(){
20
21        // Aqui la logica periodica:
22        // - Actualizaciones del estado que dependan del tiempo
23        // - Comprobacion de si el juego se ha terminado
24
25    }, 100);
26
27 /**
28 * Manejador/Receptor de mensajes del cliente.
29 * @param {object} msg Mensaje a recibir
30 *                      (su contenido es especificado en el cliente)
31 */
32 g.on = function (msg) {
33
34     // Actualizar el estado del juego en funcion del mensaje
35     // que llegue.
36     //
37     // El formato del mensaje se define en el cliente, pero
38     // todos los mensajes incluyen un campo fijo: "from"
39     // (el ID de usuario que envia el mensaje)
40
41 };
42
43 return g;
44 };

```

Listado C.1: Esquema de la parte servidora de un juego para BreakBrain

Siguiendo el esquema anterior, el código desarrollado para el servidor de “Careto” (el juego de ejemplo que está siendo creado), explicado mediante los comentarios, es el del listado C.2:

```
1 /*jshint node: true*/
2 /*global module*/
3
4 // Dependencias
5
6 var Game = require('../Game.js');
7 var util = require(' ../../server/util.js');
8
9 // Constantes del juego
10
11 var LABELS = [
12     'perro', 'gato', 'almohada', 'rata', 'rueda', 'cepillo',
13     'terminal', 'ornitorrinco', 'cocina', 'silla', 'edificio',
14     'corteza', 'sexto', 'televisor', 'piedra', 'interruptor',
15     'tela', 'diamante', 'proceso', 'dentista', 'centro',
16     'centeno', 'saltamontes', 'molino', 'coche', 'hormiga'
17 ];
18
19 module.exports = function (test, clientURL) {
20
21     // Instanciacion del juego =====
22
23     var g = new Game(
24         'Careto', // Nombre del juego
25         'MEM', // Capacidad a estimular
26         'MEM-FN', // Habilidad mental
27         clientURL // URL de la parte (auto)
28     );
29
30     test('CARETO: Instanciacion del juego', g, 'El juego Careto no es
31         instanciado correctamente');
32
33     // Definicion de los de los componentes del juego =====
34
35     /**
36      * Genera aleatoriamente un nivel de juego, compuesto
37      * por un conjunto de pares (imagen, etiqueta)
38      *
39      * @param {number} level Nivel de dificultad
40      */
41
42     var genLevel = function (level) {
43         var used = [], index, newLevel = [], label;
```

```
42     for (var i = 0, len = level + 1; i < len; i++) {
43         do {
44             index = Math.floor(Math.random() * 20);
45         } while(used[index]);
46         used[index] = true;
47         newLevel.push({
48             index: index,
49             label: LABELS[Math.floor(Math.random() * LABELS.length)]
50         });
51     }
52     return newLevel;
53 };
54
55 /**
56  * Representacion de los jugadores
57 */
58 var player1, player2;
59
60 /**
61  * Nivel actual del juego (dificultad) y representacion del nivel (
62  * pares imagen-etiqueta)
63 */
64
65 /**
66  * Estado del juego
67 */
68 var running = false;
69
70 /**
71  * Inicializacion de una partida nueva
72 */
73 var init = function () {
74     level = 0;
75     currentLevel = genLevel(0);
76     player1 = {ready: false, points: 0};
77     player2 = {ready: false, points: 0};
78 };
79
80 init();
81
82 // Logica periodica del juego =====
83
84 setInterval(function(){
85     if (running) {
86         g.emit({
87             key: 'update',
```

```
88         level: level,
89         player1: {
90             points: player1.points
91         },
92         player2: {
93             points: player2.points
94         }
95     });
96 }
97 }, 1000 * 0.5);
98
99 // Receptor de mensajes del cliente =====
100
101 g.on = function (msg) {
102
103     switch(msg.key) {
104
105         case 'ready': // Usuario preparado
106
107             // Actualizamos el jugador al que se corresponda
108             var player = player1.ready ? player2 : player1;
109             player.ready = true;
110             player.id = msg.from;
111             player.name = msg.name;
112
113             if (player1.ready && player2.ready) {
114                 // Ambos jugadores estan preparados => el juego empieza
115                 running = true;
116                 g.emit({
117                     key: 'start',
118                     player1: player1.name,
119                     player2: player2.name
120                 });
121                 g.emit({
122                     key: 'change-level',
123                     currentLevel: currentLevel
124                 });
125             } else {
126                 // Algun jugador aun no esta listo. El otro espera
127                 g.emit({
128                     key: 'waiting'
129                 });
130             }
131
132             break;
133
134         case 'exit': // El jugador ha cerrado el juego (el rival gana)
```

```
135
136     // Actualizamos el jugador correspondiente
137     if (player1.id == msg.from) {
138         player1 = {ready: false};
139     } else if (player2.id == msg.from) {
140         player2 = {ready: false};
141     }
142
143     // Si los dos jugadores han cerrado el juego, lo finalizamos
144     if (!player1.id && !player2.id) {
145         running = false;
146         init();
147         break;
148     }
149
150     // Si solo queda un jugador en pie, este gana
151     g.emit({
152         key: 'win'
153     });
154
155     break;
156
157 case 'level-completed': // Un jugador ha completado el nivel
158     actual
159
160     level++;
161     currentLevel = genLevel(level);
162     g.emit({
163         key: 'change-level',
164         currentLevel: currentLevel
165     });
166
167     break;
168
169 case 'skip': // Un jugador decide saltar una ronda
170
171     if (!running) {
172         break;
173     }
174     currentLevel = genLevel(level);
175     init();
176     running = false;
177     g.emit({
178         key: 'change-level',
179         currentLevel: currentLevel
180     });
181
```

```

181     break;
182
183     default:
184         util.error('CARETO', 'Wrong msg type received: ' + msg.key);
185     }
186
187 };
188
189 return g;
190 };

```

Listado C.2: Script servidor de “Careto”

C.4.6. Desarrollando la parte cliente

La parte cliente de los juegos supone básicamente la especificación de un conjunto de manejadores para distintos eventos: pulsación de una tecla, click de ratón, movimiento de ratón, etc.

Para la manipulación de gráficos es recomendable utilizar CreateJS, una potente biblioteca gráfica que ha sido integrada cuidadosamente en BreakBrain, y que se encuentra disponible directamente desde los scripts de lógica cliente. En [Cre] se puede encontrar toda la documentación necesaria para utilizarla. El cliente de “Careto” la utilizará, por lo que será un buen punto de partida para manejarla, en caso de que el lector no esté familiarizado con ella.

```

1 // Este es el esquema de la parte cliente de un juego para BreakBrain.
2 // Puede usarse la funcion sendMessage() para enviar mensajes a la
3 // parte servidora del juego.
4 // La variable "user", con informacion del usuario, esta disponible
5 // globalmente.
6 // La biblioteca CreateJS esta incluida en BreakBrain, por lo que
7 // puede utilizarse (y es recomendable hacerlo) para la creacion
8 // de los juegos.
9
10 window.game = (function() {
11
12     // Aqui se declaran los componentes del juego
13
14     return {
15
16         start: function() {
17             // Esta funcion es llamada al cargar el juego
18         },

```

```

19
20     tick: function() {
21         // Esta funcion es llamada en cada refresco periodico
22     },
23
24     keydown: function(key) {
25         // Esta funcion es llamada cuando una tecla es pulsada
26     },
27
28     keyup: function(key) {
29         // Esta funcion es llamada cuando una tecla es liberada
30     },
31
32     mousedown: function(x, y){
33         // Esta funcion es llamada cuando el raton es pulsado
34     },
35
36     mouseup: function(x, y){
37         // Esta funcion es llamada cuando el raton es soltado
38     },
39
40     mousemove: function(x, y){
41         // Esta funcion es llamada cuando el raton es movido
42     },
43
44     stop: function(){
45         // Esta funcion es llamada cuando el cliente cierra el
46         // juego
47     },
48
49     onMessage: function(msg) {
50         // Esta funcion recibe los mensajes de la parte servidora
51     }
52 };
53 })();
54 
```

Listado C.3: Esquema de la parte cliente de un juego para BreakBrain

Siguiendo el esquema anterior, el código desarrollado para el cliente de “Careto”, haciendo uso de CreateJS y explicado mediante los comentarios, es el del listado C.4:

```

1 window.game = (function() {
2
3     // Mensaje central del juego
4
5     var message = new Text('hola', "20px Arial", "black");
6 
```

```
6     message.textAlign = "right";
7     message.x = 580;
8     message.y = 450;
9     message.maxLength = 500;
10    stage.addChild(message);

11
12 // Informacion del jugador 1

13
14 var player1 = {
15     name: new Text(' ', "20px Arial", "black"),
16     level: new Text(' ', "15px Arial", "black")
17 };
18 player1.name.textAlign = "center";
19 player1.name.x = 150;
20 player1.name.y = 10;
21 player1.name.maxLength=250;
22 stage.addChild(player1.name);

23
24 player1.level.textAlign = "center";
25 player1.level.x = 150;
26 player1.level.y = 40;
27 player1.level.maxLength=250;
28 stage.addChild(player1.level);

29
30 // Informacion del jugador 2

31
32 var player2 = {
33     name: new Text(' ', "20px Arial", "black"),
34     level: new Text(' ', "15px Arial", "black")
35 };
36 player2.name.textAlign = "center";
37 player2.name.x = 450;
38 player2.name.y = 10;
39 player2.name.maxLength=250;
40 stage.addChild(player2.name);

41
42 player2.level.textAlign = "center";
43 player2.level.x = 450;
44 player2.level.y = 40;
45 player2.level.maxLength=250;
46 stage.addChild(player2.level);

47
48 // Nivel de juego

49
50 var level = new Text(' ', '20px Arial', 'black');
51 level.textAlign = 'left';
52 level.x = 10;
```

```
53     level.y = 450;
54     level maxWidth = 300;
55     stage.addChild(level);
56
57 /**
58  * Pinta un par (imagen, etiqueta) en una posicion determinada del
59  * canvas
60  *
61  * @param {number} index Imagen a pintar
62  * @param {number} x Posicion horizontal
63  * @param {number} y Posicion vertical
64  * @param {string} label Etiqueta asociada a la imagen
65  */
66
67 var paintImage = function (index, x, y, label) {
68
69     var aux = new Image();
70     aux.src = "/games/careto-client/braille" + index + ".png";
71
72     var img = new createjs.Bitmap(aux);
73     img.x = x;
74     img.y = y;
75     stage.addChild(img);
76
77     if (label) {
78         label = new Text(label, '15px Arial', 'black');
79         label.textAlign = 'center';
80         label.x = x + 30;
81         label.y = y + 95;
82         stage.addChild(label);
83     }
84
85
86 /**
87  * Pinta un conjunto de pares (imagen, etiqueta)
88  *
89  * @param {array} images Pares (imagen, etiqueta) a pintar
90  */
91
92 var paintImages = function (images) {
93     var x = 70, y = 80;
94     images.forEach(function (img) {
95         img.stageChild = paintImage(img.index, x, y, img.label);
96         x += 80;
97         if (x > 500) {
98             x = 70;
```

```
98             y += 120;
99         }
100     });
101 };
102
103 /**
104  * Limpia el stage de CreativeJS, eliminando los pares (imagen,
105  * etiqueta)
106  * asociados al nivel de juego actual
107  */
108 var removeLevelFromStage = function () {
109     currentLevel.forEach(function (elem) {
110         var pair = elem.stageChild;
111         stage.removeChild(pair.image);
112         stage.removeChild(pair.label);
113     });
114
115 /**
116  * Randomiza un vector de elementos, asignando a cada uno una
117  * posicion aleatoria del mismo.
118  *
119  * @param {array} o Vector a revolver
120  */
121 var shuffle = function (o) {
122     for (var j, x, i = o.length; i; j = Math.floor(Math.random() * i
123         ), x = o[--i], o[i] = o[j], o[j] = x);
124     return o;
125 }
126
127 /**
128  * Nivel actual
129  */
130 var currentLevel = [];
131
132 /**
133  * Estado del juego
134  */
135 var playing = false;
136
137 /**
138  * Intervalo periodico de cuenta atras
139  */
140 var interval = null;
141
142 /**
143  * Establece el nivel actual de juego
```

```
143     *
144     * @param {array} newLevel Vector de pares (imagen, etiqueta)
145     * asociados
146     * al nuevo nivel de juego.
147     */
148
149     var setCurrentLevel = function (newLevel) {
150
151         // Eliminamos pares anteriores
152         removeLevelFromStage();
153         // Establecemos el nuevo nivel como nivel de juego
154         currentLevel = newLevel;
155         // Pintamos el nuevo nivel
156         paintImages(newLevel);
157
158         // Asignamos un valor a la cuenta atras de la memorizacion
159         var countdown = currentLevel.length * 5;
160         playing = false;
161
162         if (interval) {
163             clearTimeout(interval);
164         }
165
166         // Iniciamos la cuenta atras
167         interval = setInterval(function () {
168             if (countdown <= 0) {
169                 if (playing) {
170                     sendMessage({key: 'skip'});
171                     return;
172                 }
173                 countdown = -1;
174                 message.text = '';
175                 playing = true;
176
177                 // Eliminamos los pares actuales, para repintarlos
178                 // aleatoriamente
179                 removeLevelFromStage();
180                 // Pintamos de nuevo los pares, de forma aleatoria y
181                 // semitransparente
182                 paintImages(shuffle(newLevel));
183                 currentLevel.forEach(function (elem) {
184                     elem.stageChild.image.alpha = 0.5;
185                     elem.stageChild.label.text = '';
186                 });
187
188                 // Establecemos la cuenta atras para resolver el nivel
189                 countdown = currentLevel.length * 10;
190                 message.text = 'Quedan ' + countdown + ' segundos';
```

```
187         return;
188     }
189     countdown--;
190     message.text = 'Quedan ' + countdown + ' segundos';
191 }, 1000);
192 };
193
194 /**
195 * Comprueba a que elemento le corresponde una posicion absoluta en
196 * pixels
197 *
198 * @param {number} x Posicion horizontal
199 * @param {number} y Posicion vertical
200 * @returns Elemento par (imagen, etiqueta) en el que el pixel (x,y)
201 *          cae
202 */
203 var whichElement = function (x, y) {
204     for (var i = 0, length = currentLevel.length; i < length; i++) {
205         var image = currentLevel[i].stageChild;
206         if (x > image.x && x < (image.x + image.width) && y > image.y
207             && y < (image.y + image.height)) {
208             return currentLevel[i];
209         }
210     }
211 };
212
213 return {
214     start: function() {
215         // Esta funcion es llamada al cargar el juego
216         sendMessage({key: 'ready', name: user.realname});
217     },
218
219     tick: function() {
220         // Esta funcion es llamada en cada refresco periodico
221     },
222
223     keydown: function(key) {
224         // Esta funcion es llamada cuando una tecla es pulsada
225     },
226
227     keyup: function(key) {
228         // Esta funcion es llamada cuando una tecla es liberada
229     },
230
231     mousedown: function(x, y) {
232         if (!playing) {
```

```
231     return;
232 }
233
234 // Comprobamos sobre que elemnto se ha hecho click
235 var elem = whichElement(x, y);
236 if (elem) {
237     // Solicitamos la etiqueta sociada a ese elemento
238     var label = prompt('Etiqueta para esta imagen');
239     if (label) {
240         // Resaltamos el elemento como resuelto y le asignamos
241         // la etiqueta
242         elem.stageChild.label.text = label;
243         elem.stageChild.image.alpha = 1;
244         // Si la etiqueta es correcta, la pintamos de verde, si
245         // no de rojo
246         if (elem.label == label) {
247             elem.stageChild.label.color = 'green';
248         } else {
249             elem.stageChild.label.color = 'red';
250         }
251         // Comprobamos si ya ha resuelto todos los pares (
252         // imagen, etiqueta)
253         for(var i = 0, len = currentLevel.length; i < len; i++) {
254             {
255                 var aux = currentLevel[i].stageChild.label;
256                 if (!aux.text || aux.color != 'green') {
257                     return;
258                 }
259             }
260         },
261
262         mouseup: function(x, y){
263             // Esta funcion es llamada cuando el raton es soltado
264         },
265
266         mousemove: function(x, y){
267             var elem = whichElement(x, y);
268             if (elem) {
269                 $('#game')[0].style.cursor = 'pointer';
270             } else {
271                 $('#game')[0].style.cursor = 'auto';
272             }
273         },
274 }
```

```
274
275     stop: function(){
276         // Esta funcion es llamada si el cliente cierra el juego
277         sendMessage({key: 'exit'});
278     },
279
280     /**
281      * Recibe un mensaje del servidor
282      */
283     onMessage: function(msg) {
284
285         switch(msg.key) {
286
287             case 'waiting': // Se debe esperar al rival
288                 message.text = 'Esperando al rival';
289                 break;
290
291             case 'start': // El juego empieza
292                 player1.name.text = msg.player1;
293                 player2.name.text = msg.player2;
294                 message.text = '';
295                 break;
296
297             case 'win': // El juego ha terminado y el usuario es el
298                 ganador
299                 message.text = 'Eres el ganador!';
300                 clearTimeout(interval);
301                 break;
302
303             case 'update': // El juego ha sufrido una actualizacion
304                 player1.level.text = msg.player1.points + ' puntos';
305                 player2.level.text = msg.player2.points + ' puntos';
306                 level.text = 'Nivel ' + msg.level;
307                 break;
308
309             case 'change-level': // El nivel de juego ha cambiado
310                 setCurrentLevel(msg.currentLevel);
311                 break;
312             }
313         };
314     })();
315 }
```

Listado C.4: Script servidor de “Careto”

C.5. Publicación del juego en BreakBrain

Para publicar un juego en BreakBrain basta con desarrollarlo sobre el código completo de la plataforma y generar un parche con el que solicitar un *pull request*¹ a través de GitHub. La petición será revisada y, si todo es correcto, aceptada. El juego quedará integrado completamente en el repositorio oficial de BreakBrain y puesto a disposición de todos los usuarios en el siguiente despliegue.

El repositorio oficial del proyecto se encuentra en la siguiente URL, a disposición de la comunidad de software libre:

<https://github.com/sgmonda/breakbrain>

Cualquiera puede colaborar en el proyecto. El software libre lo hacemos todos.

¹Solicitud de integración de código personal en un proyecto de software libre

Anexo D

Manual de usuario

ljk sdf fs df

D.1. Registro

D.1.1. Activación de la cuenta

D.1.2. Recuperación de la contraseña

D.2. Primer acceso

D.2.1. Completando el perfil de usuario

D.2.2. Definiendo las preferencias de entrenamiento

D.3. Siguiendo a otros usuarios

D.4. Entrenando el cerebro con juegos

D.4.1. Juegos monojugador

D.4.2. Jugando contra otros usuarios

Anexo E

Licencia de uso del software

BreakBrain se mantiene en desarrollo bajo licencia GPL-3, cuya descripción y particularidades se encuentran definidas a continuación. Para más información se recomienda visitar el sitio web oficial de GNU, concretamente el apartado de licencias:

[http://www.gnu.org/licenses/.](http://www.gnu.org/licenses/)



El siguiente texto es una traducción al español del original, disponible en la URL especificada arriba.

***** LICENCIA PÚBLICA GENERAL DE GNU *****

**** Versión 3, 29 de junio de 2007 ****

Esta es una traducción no oficial al español de la GNU General Public License. No ha sido publicada por la Free Software Foundation, y no establece legalmente las condiciones de distribución para el software que usa la GNU GPL -estas condiciones se establecen solamente por el texto original, en inglés, de la GNU GPL. Sin embargo, esperamos que esta traducción ayude a los hispanohablantes a entender mejor la GNU GPL.

This is an unofficial translation of the GNU General Public License into Spanish. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL -only the original English text of the GNU GPL does that. However, we hope that this translation will help Spanish speakers understand the GNU GPL better.

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Se permite la copia y distribución de copias literales de este documento, pero no se permite su modificación.

Preámbulo: La Licencia Pública General de GNU es una licencia libre, bajo "copyleft", para software y otro tipo de obras.

Las licencias para la mayoría del software y otras obras de carácter práctico están diseñadas para privarle de la libertad de compartir y modificar las obras. Por el contrario, la Licencia Pública General de GNU pretende garantizar su libertad de compartir y modificar todas las versiones de un programa -para cerciorar que permanece como software libre para todos sus usuarios. Nosotros, la Free Software Foundation, usamos la Licencia Pública General de GNU para la mayoría de nuestro software; la cual se aplica también a cualquier otra obra publicada de esta forma por parte de sus autores. Usted también puede aplicarla a sus programas.

Cuando hablamos de software libre (free software), nos referimos a libertad, no a precio. Nuestras Licencias Públicas Generales están diseñadas para garantizar su libertad de distribuir copias de software libre (y cobrar por ellas si lo desea), recibir el código fuente o poder obtenerlo si quiere, modificar el software o usar fragmentos de él en sus nuevos programas, y que sepa que puede hacer esas cosas.

Para proteger sus derechos, necesitamos impedir que otros le denieguen esos derechos o que le pidan que renuncie a ellos. Por ello, tiene ciertas responsabilidades si distribuye copias del software, o si lo modifica: la responsabilidad de respetar la libertad de otros.

Por ejemplo, si distribuye copias de un programa, bien sea gratis o por una tasa, debe transferirles a los que lo reciban las mismas libertades que usted recibió. Debe asegurarse que ellos, también, reciben o pueden obtener el código fuente. Y debe mostrarles estos términos para que ellos puedan conocer sus derechos.

Los desarrolladores que usan la GNU GPL protegen tus derechos con dos pasos: (1) haciendo valer el derecho de propiedad intelectual en el software, y (2) ofreciéndole esta Licencia que le da el permiso legal para copiarlo, distribuirlo y/o modificarlo.

Para la protección de autores y desarrolladores, la GPL explica claramente que no hay garantía para este software libre. Por el bien tanto de usuarios como de autores, la GPL requiere que las versiones modificadas sean marcadas como con cambios, de forma que sus

problemas no puedan ser atribuidos de forma errónea a autores de versiones previas.

Algunos dispositivos están diseñados para denegar a los usuarios el acceso para instalar o ejecutar versiones modificadas del software en su interior, a pesar de que el fabricante puede hacerlo. Esto es fundamentalmente incompatible con el objetivo de proteger la libertad de los usuarios de modificar el software. El modelo sistemático de este abuso ocurre en el ámbito de los productos de uso personal, lo cual es precisamente donde es más inaceptable. Por consiguiente, hemos diseñado esta versión de la GPL para prohibir la práctica de estos productos. Si estos problemas surgen de forma substancial en otro dominios, estamos preparados para extender esta disposición a esos dominios en futuras versiones de la GPL, así como sea necesario para proteger la libertad de los usuarios.

Por último, todo programa es amenazado constantemente por las patentes de software. Los Estados no deberían permitir patentes que restringen el desarrollo y el uso de software en ordenadores de propósito general, pero en aquellos que lo hacen, deseamos evitar el peligro particular de que las patentes aplicadas a un programa libre podrían convertirlo de forma efectiva en propietario. Para prevenir esto, la GPL garantiza que las patentes no pueden ser utilizadas para hacer que el programa no sea libre.

Los términos exactos y las condiciones para la copia, distribución y modificación se exponen a continuación.

Términos y Condiciones

1. Definiciones.

"Esta Licencia" se refiere a la versión 3 de la Licencia Pública General de GNU.

"Derechos de Autor ("Copyright")" también incluye a las leyes similares a la de derechos de autor ("copyright") que se apliquen a otro tipo de obras, tales como las máscaras usadas en la fabricación de semiconductores.

"El Programa" se refiere a cualquier obra con derechos de autor ("copyright") bajo esta Licencia. Cada licenciatario es tratado como "usted". Los "Licenciatarios" y los "destinatarios" pueden ser individuos u organizaciones.

"Modificar" una obra quiere decir copiar de ella o adaptar parte o la totalidad de la obra de una forma que se requieran permisos de derechos de autor ("copyright"), distintos de los de hacer una copia exacta. La obra resultante es llamada "versión modificada" de la obra previa o una

obra "basada en" la obra previa.

Una "obra amparada" significa o el Programa sin modificar o una obra basada en el Programa.

"Difundir" una obra significa hacer cualquier cosa con ella que, sin permiso, le haría responsable de forma directa o indirecta de infringir la ley correspondiente de derechos de autor ("copyright"), excepto ejecutarla en un ordenador o modificar una copia privada. La difusión incluye copiar, la distribución (con o sin modificación), hacerla disponible para el público, y en algunos países también otras actividades.

"Transmitir" una obra quiere decir cualquier tipo de difusión que permita a otras partes hacer o recibir copias. La mera interacción con un usuario a través de una red informática, sin la transferencia de una copia, no es transmitir.

Una interfaz interactiva de usuario muestra "Avisos Legales Apropiados" en la medida que incluye una característica visible práctica y destacable que (1) muestra un aviso apropiado de derechos de autor ("copyright"), e (2) informa al usuario de que no hay garantía para la obra (excepto las garantías proporcionadas), que los licenciatarios pueden transmitir la obra bajo esta Licencia, y cómo ver una copia de esta Licencia. Si la interfaz presenta una lista de comandos de usuario u opciones, como un menú, un elemento destacado en la lista satisface este criterio.

2. Código Fuente.

El "código fuente" de una obra significa la forma preferida de trabajo para hacerle modificaciones. "Código objeto" es cualquier forma no-fuente de una obra.

Una "Interfaz Estándar" significa una interfaz que es un estándar oficial definido por un cuerpo de estándares reconocido o, en el caso de interfaces especificadas para un lenguaje de programación en particular, una que es extensamente utilizada entre los desarrolladores que trabajan en ese lenguaje.

Las "Bibliotecas del Sistema" de una obra ejecutable incluyen cualquier cosa, diferente de la obra como un todo, que (a) están incluidas en la forma normal de paquetizado de un Componente Importante, y (b) sirve solo para habilitar el uso de la obra con ese Componente Importante, o para implementar una Interfaz Estándar para la cual la implementación está disponible para el público en forma de código fuente. Un "Componente Importante", en este contexto, significa un componente esencial importante (kernel, sistema de ventanas, etcétera) del sistema operativo en concreto (si hubiese) en el cual el ejecutable funciona, o un compilador utilizado para producir la obra, o un intérprete de código

objeto utilizado para hacerlo funcionar.

La "Fuente Correspondiente" de una obra en forma de código objeto significa todo el código fuente necesario para generar, instalar, y (para una obra ejecutable) hacer funcionar el código objeto y modificar la obra, incluyendo scripts para controlar dichas actividades. Sin embargo, ello no incluye la obra de las Bibliotecas del Sistema, o herramientas de propósito general o programas de libre disponibilidad general los cuales son usados sin modificaciones para la realización de dichas actividades, pero que no son parte de la obra. Por ejemplo, la Fuente Correspondiente incluye ficheros de definición de interfaces asociados a los ficheros fuente para la obra, y el código fuente para bibliotecas compartidas y subprogramas enlazados dinámicamente para los que la obra está específicamente diseñado para requerir, tales como comunicación de datos intrínseca o flujo de control entre aquellos subprogramas y otras partes de la obra.

La Fuente Correspondiente es necesario que no incluya nada que los usuarios puedan regenerar automáticamente desde otras partes de la Fuente Correspondiente.

La Fuente Correspondiente de una obra en forma de código fuente es la obra en sí.

3. Permisos básicos.

Todos los derechos concedidos bajo esta Licencia se conceden durante la duración de los derechos de autor ("copyright") del Programa, y son irrevocables siempre que se cumplan las condiciones establecidas. Esta Licencia afirma explícitamente su ilimitado permiso para ejecutar el Programa sin modificar. La salida de la ejecución de una obra amparada está amparada por esta Licencia solo si la salida, dado su contenido, constituye una obra amparada. Esta Licencia reconoce sus derechos de uso razonable u otro equivalente, según lo establecido por la ley de derechos de autor ("copyright").

Usted podrá realizar, ejecutar y difundir obras amparadas que usted no transmita, sin condición alguna, siempre y cuando no tenga otra licencia vigente. Podrá distribuir obras amparadas a terceros con el único propósito de que ellos hagan modificaciones exclusivamente para usted, o proporcionarle ayuda para ejecutar estas obras, siempre y cuando cumpla con los términos de esta Licencia en la transmisión de todo el material del cual usted no controle los derechos de autor ("copyright"). Aquellos que realicen o ejecuten las obras amparadas por usted, deben hacerlo exclusivamente en su nombre, bajo su dirección y control, en los términos que le prohiban realizar ninguna copia de su trabajo con derechos de autor ("copyright") fuera de su relación con usted.

La transmisión bajo otras circunstancias se permite únicamente bajo las condiciones expuestas a continuación. No está permitido sublicenciar, la sección 10 hace que sea innecesario.

4. Protección de los Derechos Legales de los Usuarios frente a la Ley Antievasión.

Ninguna obra amparada debe considerarse parte de una medida tecnológica efectiva, a tenor de lo establecido en cualquier ley aplicable que cumpla las obligaciones expresas en el artículo 11 del tratado de derechos de autor ("copyright") de WIPO adoptado el 20 de diciembre de 1996, o leyes similares que prohiban o restrinjan la evasión de tales medidas.

Cuando transmita una obra amparada, renuncia a cualquier poder legal para prohibir la evasión de medidas tecnológicas mientras tales evasiones se realicen en ejercicio de derechos amparados por esta Licencia respecto a la obra amparada; además, usted renunciará a cualquier intención de limitar el uso o modificación del trabajo con el objetivo de imponer, contra el trabajo de los usuarios, sus derechos legales o los de terceros para prohibir la evasión de medidas tecnológicas.

5. Transmisión de copias literales.

Usted podrá distribuir copias literales del código fuente del Programa tal cual lo ha recibido, por cualquier medio, siempre que publique visible y apropiadamente en cada copia el correspondiente aviso de derechos de autor ("copyright"); mantenga intactos todos los avisos que establezcan que esta Licencia y cualquier cláusula no-permisiva añadida acorde con la cláusula 7 son aplicables al código; mantenga intactos todos los avisos de ausencia de garantía; y proporcione a todos los destinatarios una copia de esta Licencia junto con el Programa.

Usted podrá cobrar cualquier importe o no cobrar nada por cada copia que distribuya, y podrá ofrecer soporte o protección de garantía mediante un pago.

6. Transmisión de Versiones Modificadas de la Fuente.

Usted puede transmitir una obra basada en el Programa, o las modificaciones para generarla a partir del Programa, en la forma de código fuente bajo los términos de la sección 4, suponiendo que además cumpla las siguientes condiciones:

- a. La obra debe incluir avisos destacados indicando que usted la ha modificado y dando una fecha pertinente.
- b. La obra debe incluir avisos destacados indicando que está liberada bajo esta Licencia y cualquier otra condición añadida bajo la sección 7. Este requerimiento modifica los requerimientos de la sección 4 de "mantener intactos todos los avisos".
- c. Usted debe licenciar la obra entera, como una unidad, bajo esta

Licencia para cualquier persona que esté en posesión de una copia. Esta Licencia se aplicará por consiguiente, junto con cualquier término aplicable adicional de la sección 7, a la totalidad de la obra, y a todos sus componentes, independientemente de como estén empaquetados. Esta Licencia no da permiso para licenciar la obra de otra forma, pero no invalida esos permisos si usted los ha recibido de forma separada.

- d. Si la obra tiene interfaces de usuario interactivas, cada una debe mostrar los Avisos Legales Apropriados; sin embargo, si el Programa tiene interfaces interactivas que no muestren los Avisos Legales Apropriados, tampoco es necesario que su obra lo haga.

Una recopilación de una obra amparada con otras obras separadas e independientes, que no son por su naturaleza extensiones de la obra amparada, y que no se combinan con ella con el fin de formar un programa más grande, en o sobre un volumen de un medio de almacenamiento o distribución, es llamado un "agregado" si la recopilación y su resultante derechos de autor ("copyright") no son usados para limitar el acceso o los derechos legales de los usuarios de la recopilación más allá de lo que las obras individuales permitan. La inclusión de una obra amparada en un agregado no provoca que esta Licencia se aplique a los otros componentes del agregado.

7. Transmisión en Forma de No-Fuente.

Usted puede transmitir una obra amparada en forma de código objeto bajo los términos de las secciones 4 y 5, siempre que también transmite la Fuente Correspondiente legible por una máquina bajo los términos de esta Licencia, de una de las siguientes formas:

- a. Transmitir el código objeto en, o embebido en, un producto físico (incluyendo medios de distribución físicos), acompañado de la Fuente Correspondiente en un medio físico duradero habitual para el intercambio de software.
- b. Transmitir el código objeto en, o embebido en, un producto físico (incluyendo medios de distribución físicos), acompañado de un ofrecimiento escrito, válido durante al menos tres años y válido mientras usted ofrezca recambios o soporte para ese modelo de producto, de dar a cualquiera que posea el código objeto o (1) una copia de la Fuente Correspondiente de todo el software en el producto amparado por esta Licencia, en un medio físico duradero habitual para el intercambio de software, por un precio no más elevado que el coste razonable de la realización física de la transmisión de la fuente, o (2) acceso para copiar la Fuente Correspondiente de un servidor de red sin costo alguno.

- c. Transmitir copias individuales del código objeto con una copia del ofrecimiento escrito de proveer la Fuente Correspondiente. Esta alternativa está permitida solo ocasionalmente sin fines comerciales, y solo si usted ha recibido el código objeto con ese ofrecimiento, de acuerdo con la subsección 6b.
- d. Transmitir el código objeto ofreciendo acceso desde un lugar determinado (gratuitamente o mediante pago), y ofrecer acceso equivalente a la Fuente Correspondiente de la misma manera en el mismo lugar sin cargo adicional. No es necesario exigir a los destinatarios que copien la Fuente Correspondiente junto con el código objeto. Si el lugar para copiar el código objeto es un servidor de red, la Fuente Correspondiente puede estar en un servidor diferente (gestionado por usted o un tercero) que soporte facilidades de copia equivalentes, siempre que mantenga instrucciones claras junto al código objeto especificando dónde encontrar la Fuente Correspondiente. Independientemente de qué servidor albergue la Fuente Correspondiente, usted seguirá estando obligado a asegurar que está disponible durante el tiempo que sea necesario para satisfacer estos requisitos.
- e. Transmitir el código objeto usando una transmisión peer-to-peer, siempre que informe a los otros usuarios donde se ofrece el código objeto y la Fuente Correspondiente de la obra al público general de forma gratuita bajo la subsección 6d.

Una porción separable del código objeto, cuyo código fuente está excluido de la Fuente Correspondiente, como una Biblioteca del Sistema, no necesita ser incluida en la distribución del código objeto de la obra. Un "Producto de Usuario" es o (1) un "producto de consumo", lo que significa cualquier propiedad tangible personal que es usada habitualmente con fines personales, familiares o domésticos, o (2) cualquier cosa diseñada o vendida para ser incorporada en una vivienda. A la hora de determinar cuando un producto es un producto de consumo, los casos dudosos serán resueltos en favor de la cobertura. Para un producto concreto recibido por un usuario concreto, "uso habitual" se refiere a un uso típico y común de esa clase de producto, sin tener en cuenta el estado del usuario concreto o la forma en la que el usuario concreto realmente use, o espera o se espera que use, el producto. Un producto es un producto de consumo independientemente de si el producto tiene usos esencialmente comerciales, industriales o no comerciales, a menos que dicho uso constituya el único modo de uso significativo del producto. La "Información de Instalación" de un Producto de Usuario quiere decir cualquier método, procedimiento, clave de autorización, u otra

información requerida para instalar y ejecutar versiones modificadas de la obra amparada en ese Producto de Usuario a partir de una versión modificada de su Fuente Correspondiente. La información debe ser suficiente para garantizar que el funcionamiento continuado del código fuente modificado no es prevenido o interferido por el simple hecho de que ha sido modificado.

Si usted transmite una obra en código objeto bajo esta sección en, o con, o específicamente para usar en, un Producto de Usuario, y la transmisión tiene lugar como parte de una transacción en la cual el derecho de posesión y uso de un Producto de Usuario es transferido a un destinatario en perpetuidad o por un periodo establecido (independientemente de cómo se caracterice la operación), la Fuente Correspondiente transmitida bajo esta sección debe estar acompañada de la Información de Instalación. Pero este requisito no se aplica si ni usted ni ningún tercero tiene la capacidad de instalar código objeto modificado en el Producto de Usuario (por ejemplo, la obra ha sido instalada en la ROM).

El requisito de proveer de la Información de Instalación no incluye el requisito de continuar proporcionando asistencia, garantía, o actualizaciones para una obra que ha sido modificada o instalada por el destinatario, o para un Producto de Usuario en el cual ha sido modificada o instalada. El acceso a una red puede ser denegado cuando la modificación en sí afecta materialmente y adversamente el funcionamiento de la red o viola las reglas y protocolos de comunicación de la red.

La Fuente Correspondiente transmitida, y la Información de Instalación proporcionada, de acuerdo con esta sección debe estar en un formato que sea documentado públicamente (y con una implementación disponible para el público en formato de código fuente), y no deben necesitar contraseñas o claves particulares para la extracción, lectura o copia.

8. Términos adicionales.

Los "Permisos adicionales" son términos que se añaden a los términos de esta Licencia haciendo excepciones de una o más de una de sus condiciones. Los permisos adicionales que son aplicables al Programa entero deberán ser tratados como si estuvieran incluidos en esta Licencia, en la medida bajo la ley aplicable. Si los permisos adicionales solo son aplicables a parte del Programa, esa parte debe ser usada separadamente bajo esos permisos, pero el Programa completo queda bajo la autoridad de esta Licencia sin considerar los permisos adicionales.

Cuando se transmite una copia de una obra derivada, se puede opcionalmente quitar cualesquiera permisos adicionales de esa copia, o de cualquier parte de ella. Los permisos adicionales pueden ser escritos para requerir su propia eliminación bajo ciertos casos cuando se modifica

la obra. Se pueden colocar permisos adicionales en material, añadidos a una obra derivada, para los cuales se establecen o se pueden establecer los permisos de derechos de autor ("copyright") apropiados.

No obstante cualquier otra disposición de esta Licencia, para el material que se añada a una obra derivada, se puede (si está autorizado por los titulares de los derechos de autor ("copyright") del material) añadir los términos de esta Licencia con los siguientes términos:

- a. Ausencia de garantía o limitación de responsabilidad diferente de los términos de las secciones 15 y 16 de esta Licencia; o
- b. Exigir la preservación de determinados avisos legales razonables o atribuciones de autor en ese material o en los Avisos Legales Apropiados mostrados por los obras que lo contengan; o
- c. Prohibir la tergiversación del origen de ese material, o requerir que las versiones modificadas del material se marquen de maneras razonables como diferentes de la versión original; o
- d. Limitar el uso con fines publicitarios de los nombres de los licenciantes o autores del material; o
- e. Negarse a ofrecer derechos concedidos por leyes de registro para el uso de alguno nombres comerciales, marcas registradas o marcas de servicio; o
- f. Exigir la compensación de los licenciantes y autores de ese material por cualquiera que distribuya el material (o versiones modificadas del mismo) estableciendo obligaciones contractuales de responsabilidad sobre el destinatario, por cualquier responsabilidad que estas obligaciones contractuales impongan directamente sobre los licenciantes y autores.

Todos los demás términos adicionales no permisivos son consideradas "restricciones extra" en el sentido de la sección 10. Si el Programa, tal cual se recibió, o cualquier parte del mismo, contiene un aviso indicando que se encuentra cubierto por esta Licencia junto con un término que es otra restricción, se puede quitar ese término. Si un documento de licencia contiene una restricción adicional, pero permite relicenciar o redistribuir bajo esta Licencia, se puede añadir a un material de la obra derivada bajo los términos de ese documento de licencia, a condición de que dicha restricción no sobreviva el relicenciamiento o redistribución. Si se añaden términos a una obra derivada de acuerdo con esta sección, se debe colocar, en los archivos fuente involucrados, una declaración de los términos adicionales aplicables a esos archivos, o un aviso indicando donde encontrar los términos aplicables.

Las términos adicionales, permisivos o no permisivos, pueden aparecer en forma de una licencia escrita por separado, o figurar como excepciones;

los requisitos anteriores son aplicables en cualquier forma.

9. Conclusiones.

Usted no podrá propagar o modificar una obra amparada salvo lo expresamente permitido por esta Licencia. Cualquier intento diferente de propagación o modificación será considerado nulo y automáticamente se anularán sus derechos bajo esta Licencia (incluyendo las licencias de patentes concedidas bajo el tercer párrafo de la sección 11).

Sin embargo, si usted deja de violar esta Licencia, entonces su licencia de un titular de los derechos de autor ("copyright") correspondiente será restituida (a) provisionalmente, a menos que y hasta que el titular de los derechos de autor ("copyright") explícita y finalmente termine su licencia, y (b) permanentemente, si el titular del copyright no le ha notificado su violación por algún medio razonable antes de los 60 días siguientes a la cesación.

Además, su licencia de un titular de los derechos de autor ("copyright") correspondiente será restituida permanentemente si el titular de los derechos de autor ("copyright") le notifica la violación por algún medio razonable, siendo ésta la primera vez que recibe la notificación de violación de esta Licencia (para cualquier obra) de ese titular de los derechos de autor ("copyright"), y usted subsana la violación antes de 30 días después de la recepción de la notificación.

La cancelación de sus derechos bajo esta sección no da por canceladas las licencias de terceros que hayan recibido copias o derechos de usted bajo esta Licencia. Si sus derechos han sido cancelados y no fueran renovados de manera permanente, usted no cumple los requisitos para recibir nuevas licencias para el mismo material bajo la sección 10.

10. Aceptación No Obligatoria por Tenencia de Copias.

Usted no está obligado a aceptar esta Licencia por recibir o ejecutar una copia del Programa. La propagación adicional de una obra amparada surgida únicamente como consecuencia de usar una transmisión peer-to-peer para recibir una copia tampoco requiere aceptación. Sin embargo, esta Licencia solo le otorga permiso para propagar o modificar cualquier obra amparada. Estas acciones infringen los derechos de autor ("copyright") si usted no acepta esta Licencia. Por lo tanto, al modificar o distribuir una obra amparada, usted indica que acepta esta Licencia para poder hacerlo.

11. Herencia Automática de Licencia para Destinatarios.

Cada vez que transmita una obra amparada, el destinatario recibirá automáticamente una licencia de los licenciadores originales, para ejecutar, modificar y distribuir esa obra, sujeto a esa Licencia. Usted no será responsable de asegurar el cumplimiento de esta Licencia por terceros.

Una "transacción de entidad" es una transacción que transfiere el control de una organización, o sustancialmente todos los bienes de una, o subdivide una organización, o fusiona organizaciones. Si la propagación de una obra amparada surge de una transacción de entidad, cada parte en esa transacción que reciba una copia de la obra también recibe todas las licencias de la obra que la parte interesada tuviese o pudiese ofrecer según el párrafo anterior, además del derecho a tomar posesión de las Fuentes Correspondientes de la obra a través del predecesor interesado, si el predecesor tiene o puede conseguirla con un esfuerzo razonable. Usted no podrá imponer ninguna restricción posterior en el ejercicio de los derechos otorgados o concedidos bajo esta Licencia. Por ejemplo, usted no puede imponer un pago por licencia, derechos u otros cargos por el ejercicio de los derechos otorgados bajo esta Licencia, y no puede iniciar litigios (incluyendo demandas o contrademandas en pleitos) alegando cualquier reclamación de violación de patentes por cambiar, usar, vender, ofrecer en venta o importar el Programa o alguna parte del mismo.

12. Patentes.

Un "colaborador" es un titular de los derechos de autor ("copyright") que autoriza, bajo los términos de la presente Licencia, el uso del Programa o una obra en la que se base el Programa. La obra así licenciada se denomina "versión en colaboración" del colaborador.

Las "demandas de patente esenciales" del colaborador son todas las reivindicaciones de patentes poseídas o controladas por el colaborador, ya se encuentren adquiridas o hayan sido adquiridas con posterioridad, que sean infringidas de alguna manera, permitidas por esta Licencia, al hacer, usar o vender la versión en colaboración, pero sin incluir demandas que solo sean infringidas como consecuencia de modificaciones posteriores de la versión en colaboración. Para los propósitos de esta definición, "control" incluye el derecho de conceder sublicencias de patente de forma consistente con los requisitos establecidos en la presente Licencia.

Cada colaborador le concede una licencia de la patente no-exclusiva, global y libre de regalías bajo las demandas de patente esenciales del colaborador, para hacer, usar, modificar, vender, ofrecer para venta, importar y otras formas de ejecución, modificación y difusión del contenido de la versión en colaboración.

En los siguientes tres párrafos, una "licencia de patente" se define como cualquier acuerdo o compromiso expreso, cualquiera que sea su denominación, que no imponga una patente (como el permiso expreso para ejecutar una patente o acuerdos para no imponer demandas por infracción

de patente). "Conceder" una licencias de patente de este tipo a un tercero significa hacer tal tipo de acuerdo o compromiso que no imponga una patente al tercero.

Si usted transmite una obra amparada, conociendo que está amparada por una licencia de patente, y las Fuentes Correspondientes no se encuentran disponibles de forma pública para su copia, sin cargo alguno y bajo los términos de esta Licencia, ya sea a través de un servidor público o mediante cualquier otro medio, entonces usted deberá (1) hacer que las Fuentes Correspondientes sean públicas, o (2) tratar de eliminar los beneficios de la licencia de patente para esta obra en particular, o (3) tratar de extender, de manera compatible con los requisitos de esta Licencia, la licencia de patente a terceros. "Conocer que está afectado" significa que usted tiene conocimiento real de que, para la licencia de patente, la distribución de la obra amparada en un país, o el uso de la obra amparada por sus destinatarios en un país, infringiría una o más patentes existentes en ese país que usted considera válidas por algún motivo.

Si en virtud de o en conexión con alguna transacción o acuerdo, usted transmite, o difunde con fines de distribución, una obra amparada, y concede una licencia de patente para algún tercero que reciba la obra amparada, y les autorice a usar, transmitir, modificar o difundir una copia específica de la obra amparada, entonces la licencia de patente que usted otorgue se extiende automáticamente a todos los receptores de la obra amparada y cualquier obra basada en ella.

Una licencia de patente es "discriminatoria" si no incluye dentro de su ámbito de cobertura, prohíbe el ejercicio de, o está condicionada a no ejercitar uno o más de los derechos que están específicamente otorgados por esta Licencia. Usted no debe transmitir una obra amparada si está implicado en un acuerdo con terceros que esté relacionado con el negocio de la distribución de software, en el que usted haga pagos a terceros relacionados con su actividad de distribución de la obra, bajo el que terceros conceden, a cualquier receptor de la obra amparada, una licencia de patente discriminatoria (a) en relación con las copias de la obra amparada transmitidas por usted (o copias hechas a partir de estas), o (b) principalmente para y en relación con productos específicos o compilaciones que contengan la obra amparada, a menos que usted forme parte del acuerdo, o que esa licencia de patente fuese concedida antes del 28 de marzo de 2007.

Ninguna cláusula de esta Licencia debe ser considerada como excluyente o limitante de cualquier otra licencia implicada u otras defensas legales a que pudiera tener derecho bajo la ley de propiedad intelectual vigente.

13. No Abandonar la Libertad de Otros.

Si se le imponen condiciones (bien sea por orden judicial, acuerdo o de otra manera) que contradicen las condiciones de esta Licencia, estas no le eximen de las condiciones de esta Licencia. Si usted no puede transmitir una obra amparada de forma que pueda satisfacer simultáneamente sus obligaciones bajo esta Licencia y cualesquiera otras obligaciones pertinentes, entonces, como consecuencia, usted no puede transmitirla. Por ejemplo, si usted está de acuerdo con los términos que le obligan a cobrar una regalía por la transmisión a aquellos a los que transmite el Programa, la única forma en la que usted podría satisfacer tanto esos términos como esta Licencia sería abstenerse completamente de transmitir el Programa.

14. Utilización con la Licencia Pública General Afferro de GNU.

A pesar de cualquier otra disposición de esta Licencia, usted tiene permiso para enlazar o combinar cualquier obra amparada con una obra licenciada bajo la Licencia Pública General Afferro de GNU en una única obra combinada, y para transmitir la obra resultante. Los términos de esta Licencia continuarán aplicándose a la parte que es la obra amparada, pero los requisitos particulares de la Licencia Pública General Afferro de GNU, sección 13, concernientes a la interacción a través de una red se aplicarán a la combinación como tal.

15. Versiones Revisadas de esta Licencia.

La Free Software Foundation puede publicar versiones revisadas y/o nuevas de la Licencia General Pública de GNU de vez en cuando. Cada nueva versión será similar en espíritu a la versión actual, pero puede diferir en detalles para abordar nuevos problemas o preocupaciones.

Cada versión recibe un número de versión distintivo. Si el Programa especifica que cierta versión numerada de la Licencia General Pública de GNU "o cualquier versión posterior" se aplica a él, usted tiene la opción de seguir los términos y condiciones de esa versión numerada o de cualquier versión posterior publicada por la Free Software Foundation. Si el Programa no especifica un número de versión de la Licencia General Pública de GNU, usted puede escoger cualquier versión publicada por la Free Software Foundation.

Si el Programa especifica que un representante puede decidir que versiones futuras de la Licencia General Pública de GNU pueden ser utilizadas, la declaración pública del representante de aceptar una versión permanentemente le autoriza a usted a elegir esa versión para el Programa.

Las versiones posteriores de la licencia pueden darle permisos adicionales o diferentes. No obstante, no se impone a ningún autor o

titular de los derechos de autor obligaciones adicionales como resultado de su elección de seguir una versión posterior.

16. Descargo de Responsabilidad de Garantía.

NO HAY GARANTÍA PARA EL PROGRAMA, PARA LA EXTENSIÓN PERMITIDA POR LA LEY APLICABLE. EXCEPTO CUANDO SE INDIQUE LO CONTRARIO POR ESCRITO, LOS TITULARES DE LOS DERECHOS DE AUTOR ("COPYRIGHT") Y/O TERCEROS PROPORCIONAN EL PROGRAMA "TAL CUAL" SIN GARANTÍAS DE NINGÚN TIPO, BIEN SEAN EXPLÍCITAS O IMPLÍCITAS, INCLUYENDO, PERO NO LIMITADO A, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN Y APTITUD PARA UN PROPÓSITO PARTICULAR. EL RIESGO TOTAL EN CUANTO A CALIDAD Y RENDIMIENTO DEL PROGRAMA ES CON USTED. SI EL PROGRAMA PRESENTA ALGÚN DEFECTO, USTED ASUME EL COSTO DE TODAS LAS REVISIONES NECESARIAS, REPARACIONES O CORRECCIONES.

17. Limitación de la responsabilidad.

EN NINGÚN CASO A MENOS QUE SEA REQUERIDO POR UNA LEY APLICABLE O ACUERDO ESCRITO NINGÚN TITULAR DE LOS DERECHOS DE AUTOR ("COPYRIGHT"), O NINGÚN TERCERO QUE MODIFIQUE Y/O TRANSMITA EL PROGRAMA COMO SE PERMITE ANTERIORMENTE, SERÁ RESPONSABLE ANTE USTED POR DAÑOS, INCLUYENDO CUALESQUIERA DAÑOS GENERALES, PARTICULARES, IMPREVISTOS O DERIVADOS DEL USO O IMPOSIBILIDAD DE USO DEL PROGRAMA (INCLUYENDO, PERO NO LIMITADO A, LA PÉRDIDA DE DATOS, DATOS GENERADOS INCORRECTOS, PÉRDIDAS SUFRIDAS POR USTED O POR TERCERAS PERSONAS, O LOS FALLOS DEL PROGRAMA PARA OPERAR CON OTROS PROGRAMAS), INCLUSO SI DICHO TITULAR O UN TERCERO HA SIDO ADVERTIDO DE LA POSIBILIDAD DE TALES DAÑOS.

18. Interpretación de las Secciones 15 y 16.

Si el descargo de responsabilidad de garantía y el límite de responsabilidad proporcionado anteriormente no tiene efectos legales de acuerdo a sus términos, los juzgados deberán aplicar la ley local que más se asemeje a una renuncia absoluta de la responsabilidad civil concerniente al Programa, a menos que una garantía o una asunción de responsabilidad acompañe a la copia del Programa como resultado del pago de una tasa.

Fin de los términos y condiciones

Cómo Aplicar Estos Términos a Sus Nuevos Programas

Si desarrolla un nuevo programa, y quiere que sea lo más usado posible por el público, la mejor manera de conseguirlo es hacerlo software libre para que cualquiera pueda redistribuirlo y modificarlo bajo estos términos.

Para ello, añada la siguiente nota al programa. Lo más seguro es añadirla al principio de cada fichero fuente para declarar más efectivamente la exclusión de garantía; y cada fichero debe tener al menos la línea de "derechos de autor ("copyright")" y un puntero a donde se pueda encontrar la anotación completa.

<una línea para dar el nombre del programa y una breve idea de lo que hace>

Copyright (C) <año> <nombre del autor>

Este programa es software libre: puede redistribuirlo y/o modificarlo bajo los términos de la Licencia General Pública de GNU publicada por la Free Software Foundation, ya sea la versión 3 de la Licencia, o (a su elección) cualquier versión posterior.

Este programa se distribuye con la esperanza de que sea útil pero SIN NINGUNA GARANTÍA; incluso sin la garantía implícita de MERCANTIBILIDAD o CALIFICADA PARA UN PROPÓSITO EN PARTICULAR. Vea la Licencia General Pública de GNU para más detalles.

Usted ha debido de recibir una copia de la Licencia General Pública de GNU junto con este programa. Si no, vea <<http://www.gnu.org/licenses/>>.

También añada información sobre cómo contactarle por correo electrónico u ordinario.

Si el programa es interactivo, haga que muestre un breve aviso como el siguiente cuando se inicie en modo interactivo:

<programa> Copyright (C) <año> <nombre del autor>

Este programa se ofrece SIN GARANTÍA ALGUNA; escriba 'show w' para consultar los detalles. Este programa es software libre, y usted puede redistribuirlo bajo ciertas condiciones; escriba 'show c' para más información.

Los hipotéticos comandos show w y show w deberán mostrar las partes correspondientes de la Licencia General Pública. Por supuesto, los comandos en su programa pueden ser diferentes; para una interfaz gráfica de usuario, puede usar un mensaje del tipo "Acerca de".

También debería conseguir que su empresa (si trabaja como programador) o escuela, en su caso, firme una "renuncia de derechos de autor ("copyright")" sobre el programa, si fuese necesario. Para más información a este respecto, y saber cómo aplicar y cumplir la licencia GNU GPL, consulte <http://www.gnu.org/licenses/>.

La Licencia General Pública de GNU no permite incorporar sus programas

como parte de programas propietarios. Si su programa es una subrutina en una biblioteca, podría considerar mucho más útil permitir el enlace de aplicaciones propietarias con la biblioteca. Si esto es lo que quiere hacer, utilice la GNU Lesser General Public License en vez de esta Licencia. Pero primero, por favor consulte <http://www.gnu.org/philosophy/why-not-lgpl.html>.

La anterior traducción ha sido extraída de [Abe].

Bibliografía

- [Abe] Gonzalo Abella. Traducción no oficial de la licencia GPL v3 al español. URL <http://goo.gl/E1y7od>.
- [Acc12] Accenture. Mobile Web Watch. 2012.
- [Ama] Amazon. Amazon Web Services. URL <http://aws.amazon.com/es/>.
- [App] AppFog. AppFog. URL <http://www.appfog.com/>.
- [BB] Suzanna Becker and Neil Burgess. Modelling spatial recall, mental imagery and neglect.
- [BER07] Karlene Ball, Jerri D. Edwards, and Lesley A. Ross. The Impact of Speed of Processing Training on Cognitive and Everyday Functions. 2007.
- [BKP⁺06] Louis Bherer, Arthur F. Kramer, Matthew S. Peterson, Stanley Colcombe, Kirk Erickson, and Ensar Becic. Testing the limits of cognitive plasticity in older adults: Application to attentional control. *Elsevier B.V.*, 2006.
- [BS04] Deborah M. Burke and Meredith A. Shafto. *Aging and Language Production*. Current Directions In Psychological Science, 2004.
- [Cre] CreateJS. CreateJS Suite for HTML5 interactive experiences. URL <http://www.createjs.com/>.
- [C.S03] Green C.S. Action video game modifies visual selective attention. *PubMed*, 2003.
- [Cze01] Thomas B. Czerner. *What Makes You Tick? The Brain in Plain English*. John Wiley & Sons, Inc., New York, 2001.
- [Dam94] Antonio R. Damasio. *Descartes' Error: Emotion, Reason, and the Human Brain*. Penguin Books, New York, 1994.
- [DBM08] C.N. DeWall, R.F. Baumeister, and E.J. Masicampo. *Evidence that logical reasoning depends on conscious processing*. Conscious Cogn, 2008.

- [DMCW04] Stanislas Dehaene, Nicolas Molko, Laurent Cohen, and Anna J Wilson. Arithmetic and the brain. *Current opinion in neurobiology*, 14(2):218–24, April 2004. ISSN 0959-4388. URL <http://www.ncbi.nlm.nih.gov/pubmed/15082328>.
- [DTHS] Zheng D., Oka T., Bokura H., and Yamaguchi S. The key locus of common response inhibition network for no-go and stop signals.
- [Fac06] Fast Facts. Matters of the Brain. pages 16–19, 2006.
- [FSP07] Jing Feng, Ian Spence, and Jay Pratt. Playing an Action Video Game Reduces Gender Differences in Spatial Cognition. 2007.
- [Gal] Winifred Gallagher. The Atlantic Monthly. URL <http://www.theatlantic.com/magazine/archive/1994/09/how-we-become-what-we-are/303534/>.
- [Git] Inc. GitHub. GitHub. Build software better, together. URL <http://www.github.com/>.
- [Gooa] Google. Chrome Experiments. URL <http://www.chromeexperiments.com/>.
- [Goob] Google. Chrome web browser. URL <http://chrome.google.com>.
- [Hac11] Dan Hackley. Coach your cortex. 24(8):586–589, 2011.
- [HHG00] Jv Haxby, Ea Hoffman, and Mi Gobbini. The distributed human neural system for face perception. *Trends in cognitive sciences*, 4(6):223–233, June 2000. ISSN 1879-307X. URL <http://www.ncbi.nlm.nih.gov/pubmed/10827445>.
- [Hir04] Mika Hirvasoja. Effects of computer game playing on spatial skills. 2004.
- [II] Mental Health Institute of Neurosciences and Addiction (INMHA). The brain from top to bottom. An interactive website about the human brain and behavoir. URL <http://thebrain.mcgill.ca/>.
- [Jac09] Stephen Jacyna. *The most important of all the organs: Darwin on the brain*. BRAIN. A journal of Neurology., 2009.
- [JBJP08] Susanne M Jaeggi, Martin Buschkuhl, John Jonides, and Walter J Peirig. Improving fluid intelligence with training on working memory. *Proceedings of the National Academy of Sciences of the United States of America*, 105(19):6829–33, 2008. ISBN 0801268105. ISSN 10916490. URL <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2383929&tool=pmcentrez&rendertype=abstract>.

- [JC05] Temple Grandin Johnson and Catherine. *Animals in Translation: Using the Mysteries of Autism to Decode Animal Behavior*. A Harvest Book, Harcourt, Inc., Orlando, 2005.
- [Joh00] John Allman. *Evolving Brains*. W.H. Freeman, 2000.
- [Jul04] Pascual Julián Iranzo. *Lógica simbólica para informáticos*. RA-MA, 2004.
- [KBD⁺88] Ball K.K., Beard B.L., Roenker D.L., Miller R.L., and Griggs D.S. Age and visual search: expanding the useful field of view. *PubMed*, 1988.
- [KJO⁺05] Wood K.M., Edwards J.D., Clay O.J., Wadley V.G., Roenker D.L., and Ball K.K. Sensory and cognitive factors influencing functional ability in older adults. 2005.
- [Lau01] Andrew Lautin. *The Limbic Brain*. Plenum Publishers, New York, 2001.
- [LeD96] Joseph LeDoux. *The Emotional Brain: The Mysterious Underpinnings of Emotional Life*. Simon & Schuster, 1996.
- [Lin] Dr. Donald A.B. Lindberg. Medlineplus. Trusted Health Information for you.
- [Mac49] PD MacLean. Psychosomatic disease and the Visceral Brain. *Recent developments bearing on the Papez theory of Emotion*, 1949. URL [http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Psychosomatic+Disease+and+the+"Visceral+Brain"\#4](http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Psychosomatic+Disease+and+the+).
- [MAJ02] Thomas F Münte, Eckart Altenmüller, and Lutz Jäncke. The musician's brain as a model of neuroplasticity. *Nature reviews. Neuroscience*, 3(6):473–8, June 2002. ISSN 1471-003X. URL <http://www.ncbi.nlm.nih.gov/pubmed/21566538>.
- [Mal] Maciej Malecki. Vows. Asynchronous BDD & continuous integration for node.js. URL <https://github.com/cloudhead/vows>.
- [MCKV01] R Melzack, T JCoderre, J Katz, and a L Vaccarino. Central neuroplasticity and pathological pain. *Annals of the New York Academy of Sciences*, 933:157–74, March 2001. ISSN 0077-8923. URL <http://www.ncbi.nlm.nih.gov/pubmed/12000018>.
- [MDB⁺08] Arne May, Joenna Driemeyer, Janina Boyke, Christian Gaser, and Christian Bu. Changes in Gray Matter Induced by Learning. Technical report, 2008.
- [Mer] Merck Co. The Merck Manuals. Trusted Medical and Scientific Information.
- [Min] Miniplay. Minijuegos. URL <http://www.minijuegos.com>.

- [Mona] MongoDB. MongoDB. URL <http://www.mongodb.org/>.
- [Monb] Mongolab. MongoLab. URL <https://mongolab.com/>.
- [Mor09] Adrian R. Morrison. *An Odyssey with Animals: A Veterinarian's Reflections on the Animal Rights & Welfare Debate*. Oxford University Press, 2009.
- [Noda] NodeJS. NodeJS official website. URL <http://nodejs.org/>.
- [Nodb] Nodester. Nodester. URL <http://www.nodester.com>.
- [NRLP⁺] James K. Nelson, Patricia A. Reuter-Lorenz, Jonas Persson, Ching-Yue C. Sylvester, and John Jonides. *Mapping interference resolution across task domains: A shared control process in left inferior frontal gyrus*.
- [OF94] Lynn Okagaki and Peter A. Frensch. *Effects of Video Game Playing on Measures of Spatial Performance: Gender Effects in Late Adolescence*. Journal of Applied Developmental Psychology, 1994.
- [OK04] Klaus Oberauer and Reinhold Kliegl. *Simultaneous Cognitive Operations in Working Memory After Dual-Tasks Practice*. Journal of Experimental Psychology, 2004.
- [OM06] Vilayanur S. Ramachandran Oberman and Lindsay M. Broken Mirrors: A Theory of Autism. Technical report, 2006.
- [OWK04] Pernille J Olesen, Helena Westerberg, and Torkel Klingberg. Increased pre-frontal and parietal activity after training of working memory. *Nature neuroscience*, 7(1):75–9, January 2004. ISSN 1097-6256. URL <http://www.ncbi.nlm.nih.gov/pubmed/14699419>.
- [Pag07] Lawrence Page. Scoring documents in a linked database, 2007. URL <http://goo.gl/0Eq0K>.
- [Pan98] Jaak Panksepp. *Affective Neuroscience: The Foundations of Human and Animal Emotions*. Oxford University Press, New York, 1998.
- [PD08] Christopher Pittenger and Ronald S Duman. Stress, depression, and neuroplasticity: a convergence of mechanisms. *Neuropsychopharmacology : official publication of the American College of Neuropsychopharmacology*, 33(1):88–109, January 2008. ISSN 0893-133X. URL <http://www.ncbi.nlm.nih.gov/pubmed/17851537>.
- [PHP] PHP. PHP, a widely-used general-purpose scripting language. URL <http://php.net>.

- [Rat98] Rational. *Rational Unified Process. Best Practices for Software Development Teams*, 1998.
- [Res94] Richard M. Restak. *The Modular Brain: How New Discoveries in Neuroscience are Answering Age-Old Questions about Memory, Free Will, Consciousness, and Personal Identity*. Scribner, 1994.
- [Res95] Richard M. Restak. *Brainscapes: An Introduction to What Neuroscience Has Learned about the Structure, Function, and Abilities of the Brain*. Hyperion Books, 1995.
- [Sap] Robert M. Sapolsky. Biology and Human Behavior: The Neurological Origins of Individuality.
- [Sap04] Robert M. Sapolsky. *Why Zebras Don't Get Ulcers: The Acclaimed Guide to Stress, Stress-Related Diseases, and Coping*. Henry Holt and Company, New York, third edit edition, 2004.
- [Sap05] Robert M. Sapolsky. *Monkeyluv and Other Essays on Our lives as Animals*. Scribner, New York, 2005.
- [SBS⁺12] Goal Statement, Editorial Board, Credit Statement, Sponsorship Statement, and Disclosure Statement. *Neuroplasticity: Teaching an Old Brain New Tricks*. 2012.
- [Shu09] Neil Shubin. *Your Inner Fish: A Journey Into the 3.5-Billion-Year History of the Human Body*. Vintage, 2009.
- [TKD⁺09] Iris Trinkler, John a King, Christian F Doeller, Michael D Rugg, and Neil Burgess. Neural bases of autobiographical support for episodic recollection of faces. *Hippocampus*, 19(8):718–30, August 2009. ISSN 1098-1063. URL <http://www.ncbi.nlm.nih.gov/pubmed/19173228>.
- [Uni] Unity. Unity3D Engine. URL <http://unity3d.com/>.
- [Vis] Visionmedia. Express. URL <http://expressjs.com/>.
- [W3T] W3Techs. World Wide Web Technology Surveys. URL <http://w3techs.com/>.

Índice alfabético

- acalculia, 54
- acetilcolina, 31
- afasia, 38
- amígdala, 46–49
- artefacto, 59, 61
- ataxia, 33
- bug, 107
- bulbo raquídeo, 29
- bytecode, 20
- córtex, 2, 24
- córtex/corteza
 - inferotemporal, 38, 44
 - occipitotemporal, 52
 - prefrontal, 48, 122
 - visual asociativa, 44
 - visual primaria, 44
 - visual secundaria, 44
- cambio cualitativo, 52
- capas germinales, 28
- cerebelo, 32
- corticosterona, 48
- cortisol, 48
- cortisona, 48
- cuerpo calloso, 46
- cuerpo estriado, 42
- diencéfalo, 29
- dopamina, 50
- embriología, 28
- express.js, 102
- GET, 17
- glucorticoides, 48
- hipermedia, 17
- hipertexto, 17
- hipocampo, 46, 54
- hipotálamo, 29
- HTML, 17, 19
- HTTP, 17
- insulina, 48
- lóbulo
 - frontal, 53, 54
 - occipital, 122
 - parietal, 53, 54
 - temporal, 122
- lumosity, 15
- materia
 - blanca, 51
 - gris, 51
- mesencéfalo, 29, 31, 35
- metencéfalo, 29
- mielencéfalo, 29
- mongodb, 75
- núcleo, 24
- núcleos del rafé, 31
- narcolepsia, 46
- neocórtex, 26, 35, 41, 42
- neurona
 - piramidal, 39
- neuroplasticidad, 3, 42, 43, 51, 55, 121

neurotransmisor, 26, 27, 49
nodejs, 75, 101, 102, 106
norepinefrina, 31

pagerank, 12–15
PHP, 19
POST, 17
prosencéfalo, 29, 35
protuberancia anular, 29
puente troncoencefálico, 29, 31

rombencéfalo, 29, 35

sistema límbico, 35, 36, 39, 48

tálamo, 29
telencéfalo, 29
templo, 47

usabilidad, 107

vows, 106

websockets, 103

Este documento fue editado y tipografiado con L^AT_EX
empleando la clase **arco-pfc** que se puede encontrar en:
https://bitbucket.org/arco_group/arco-pfc

[Respetá esta atribución al autor]

