ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO DE TELECOMUNICACIÓN

Título del proyecto:

SNAPER! HERRAMIENTA DE ANÁLISIS ESTÁTICO

DOCUMENTO

Lidia Marín Franco

Oihane Usunariz

Pamplona, 27 de Julio del 2016

Cuando quieres realmente una cosa, todo el Universo conspira para ayudarte a conseguirla.
Paulo Coelho

Agradecimientos

A mis padres, por darme la oportunidad de estudiar y permitirme la libertad de terminar el proyecto cuando yo lo sentía.

A Oihane, mi tutora, por la confianza y las facilidades que me ha dado para realizar este proyecto después de tantos años.

A Javi, que a pesar de todo sigue confiando en mí y me ha guiado y animado (a su manera) a que termine la carrera.

CONTENIDO

CAPÍTULO 1

INTRODUCCIÓN

1. INTRODUCCIÓN

1.1. PRESENTACIÓN DEL PROYECTO

El desarrollo de la tecnología está cambiando nuestros hábitos de aprender, de relacionarnos con las personas. El conocimiento gracias a Internet esta accesible a todo el mundo que tenga un dispositivo móvil o un ordenador.

El alumnado aprende de manera intuitiva como buscar información o manejar aplicaciones. A la programación en las aulas actualmente se le ha dado cada vez más importancia.

Para el profesor o la profesora es difícil valorar los conocimientos de programación del alumnado. Existen varias formas de programar lo mismo y obtener el mismo resultado, aparece para esto un nuevo concepto, el pensamiento computacional y plataformas que ayudan al alumnado a desarrollar su capacidad de análisis computacional como son Scratch, App Inventor, Snap!, etc…

El pensamiento computacional implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática.
En el ámbito de aplicaciones los conceptos del pensamiento computacional más utilizados son:

- Secuencia
- Iteración
- Paralelismo
- Eventos
- Condicionales
- Operadores
- Datos

Herramientas como Scratch y Snap! utilizan bloques gráfico que se arrastran y se anidan. Los bloques gráficos permiten a los usuarios un aprendizaje eficaz, y su entorno amigable permite la potenciación de la creatividad en el diseño y desarrollo de los proyectos.

Este proyecto permite al profesorado y alumnado analizar y valorar sus capacidades mediante las características del pensamiento computacional y utilizando la herramienta Snap!.

1.2. OBJETO DEL PROYECTO

Este proyecto pretende crear una herramienta dirigida al profesorado y alumnado donde analizar proyectos generados en Snap!, hacer un seguimiento de ejercicios propuestos y valorar sus capacidades mediante los conceptos del pensamiento computacional.

Esta herramienta tiene que ser accesible y usable.

1.2.1. OBJETIVOS ESPECÍFICOS

1. Comprender el concepto actual del pensamiento computacional. Características y su uso.

2. Conocer las herramientas que existe para ayudar al alumnado, tales como Scratch o Snap!, a entender la programación y los diferentes tipos de análisis de código que existen actualmente.

3. Crear una aplicación web que permita a usuarios y profesorado evaluar los distintos ejercicios propuestos.

4. Crear un servicio para la aplicación web, el cual implementa una interfaz REST para evitar la sobrecarga del sistema

5. Construir el modelo de datos necesarios para almacenar la distinta información de los usuarios, ejercicios y notas requerida

6. Diseñar una interfaz gráfica accesible y usable mediante Bootstrap.

7. Permitir un seguimiento de la evolución de los alumnos y alumnas en términos del pensamiento computacional.

CAPÍTULO 2

MARCO TEÓRICO

2. MARCO TEÓRICO

Actualmente la sociedad se ha visto transformada por las TIC. Los ordenadores, móviles y otros dispositivos digitales son una herramienta fundamental hoy en día. Es la era de la computación. Aparecen nuevas profesiones, nuevos servicios etc… Permiten realizar cualquier tarea que pueda hacer un programa, pero además, cualquier individuo puede instalar cualquier programa, construir cualquier programa, compartir cualquier programa, combinar programas a distancia.

En el siglo XXI, comprender las TIC permite acceder a un mundo nuevo. Este mundo se está redefiniendo: adaptarse o morir.

En esta casuística aparecen nuevos conceptos, entre ellos el **Pensamiento computacional**.

2.1. PENSAMIENTO COMPUTACIONAL

Los estudiantes y profesionales tienen la necesidad de aprender y practicar las habilidades del pensamiento computacional para poder utilizar las nuevas tecnologías y confrontar los desafíos del siglo XXI.

Jeannette Wing, la principal percusora, define así el pensamiento computacional: "el pensamiento computacional implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática" en su artículo Computational thinking(2).

A partir de esta definición han ido apareciendo otras sobre el pensamiento computacional, así la Sociedad Internacional para la Tecnología en Educación (ISTE) y la Asociación de Docentes en Ciencias de la Computación (CSTA) proponen como definición, el pensamiento computacional es un proceso de resolución de problemas y sus características principales son:

- Formular problemas de forma que se permita el uso de un ordenador y otras herramientas para ayudar a resolverlos.
- Organizar y analizar lógicamente la información.
- Representar la información a través de abstracciones como los modelos y las simulaciones.
- Automatizar soluciones haciendo uso del pensamiento algorítmico (estableciendo una serie de pasos ordenados para llegar a la solución).
- Identificar, analizar e implementar posibles soluciones con el objetivo de lograr la combinación más efectiva y eficiente de pasos y recursos.
- Generalizar y transferir este proceso de resolución de problemas para ser capaz de resolver una gran variedad de familias de problemas

Estas habilidades se apoyan mediante una serie de o actitudes que son esenciales en el pensamiento computacional. Estas actitudes incluyen:

- Confianza en el manejo de la complejidad.
- Persistencia al trabajar con problemas difíciles.
- Tolerancia a la ambigüedad.
- Habilidad para lidiar con problemas no estructurados.

- Habilidad para comunicarse y trabajar con otros para alcanzar una meta o solución común.

También Google da su propia definición de pensamiento computacional como un conjunto de habilidades y técnicas de solución de problemas, que los ingenieros de software usan para escribir los programas informáticos que subyacen a las aplicaciones que usamos a diario.

El pensamiento computacional tiene como objetivo desarrollar sistemáticamente las habilidades de pensamiento crítico y resolución de problemas con base en los conceptos de la computación.

El **pensamiento crítico** se define como el modo de pensar en el cual se mejora la calidad de su pensamiento al apoderarse de las estructuras inherentes del acto de pensar y al someterlas a estándares intelectuales.

Cuando una persona utiliza el **pensamiento computacional** piensa críticamente: utiliza de forma constructiva la información, las conclusiones y los puntos de vista; se empeña en ser claro, exacto, preciso y relevante.

Por medio de los conceptos de la computación es posible entender que aspectos de un problema pueden resolverse aprovechando el procesamiento de los ordenadores actuales.

2.1.1. USO DEL PENSAMIENTO COMPUTACIONAL

La mayoría de los usuarios actualmente solo utilizan los ordenadores para hacer trabajos de oficina, redes sociales, chats, etc... pero no aportan el cambio en la manera de confrontarlos problemas de este siglo.

Las habilidades de una persona que utiliza el pensamiento computacional utiliza diferentes niveles de abstracción para entender y resolver un problema lo más eficazmente posible. Jeannette Wing sostiene que la esencia del pensamiento computacional es la abstracción.

Hoy en día los usuarios están invadidos por la masificación de información. Se tiene rápido acceso a ella a través de Internet. Pero mucha de esta información es mala, falsa o es errónea. De ahí que sea tan importante el pensamiento computacional, teniendo una buena base en el pensamiento computacional se puede discernir más fácilmente que información es correcta o no.

El ordenador pasa a ser un instrumento a disposición de estudiantes, del desarrollo humano, científico y tecnológico. Así la finalidad del pensamiento computacional es desarrollar el pensamiento crítico junto con los conceptos de la computación como son la abstracción, programación, algoritmos, etc... Estos conceptos son enseñados a cualquier persona, no solo a ingenieros o programadores.

El pensamiento computacional es independiente de la profesión o estudios de una persona, el pensamiento computacional puede ser usado en la vida personal o profesional.

A modo de ejemplo, Inglaterra, a partir del año académico 2014-15, incluyo formalmente el estudio del Pensamiento Computacional y programación de ordenadores como parte del plan de estudios de la educación primaria y secundaria, como se describe en el

"currículo nacional en Inglaterra: Estudio de Programa de Computación" (Department for Education England, 2013).

Así que un estudiante que aprenda el pensamiento computacional no quiere decir que su carrera profesional vaya dirigida a ser informático o cualquier otra profesión relacionada con la ciencia.

Por ejemplo, un escritor aplica el pensamiento computacional cuando empieza a escribir su libro, piensa en el argumento, sigue un flujo de acontecimientos, que repasa varias veces.

Otro ejemplo sería aquel comercial que hace un estudio de mercado, diseña un nuevo producto y hace un cálculo de los gastos que conlleva llevarlo a cabo.

Pero, ¿qué se debe valorar cuando se estudia el pensamiento computacional?

2.1.2. VOCABULARIO UTILIZADO PENSAMIENTO COMPUTACIONAL

En el artículo de Computational Thinking leader ship toolkit (Pensamiento Computacional caja de herramientas para lideres) se presenta el vocabulario utilizado en el pensamiento computacional:

- **Recopilar datos**: El proceso de reunir la información apropiada.
- **Analizar datos**: Darle sentido a los datos, hallar o establecer patrones y sacar conclusiones.
- **Representar datos**: Representar y organizar los datos en gráficas, cuadros, palabras o imágenes apropiadas.
- **Descomponer problemas**: Dividir una tarea en partes más pequeñas y más manejables.
- **Abstraer**: Reducir la complejidad para definir o establecer la idea principal.
- **Algoritmos y procedimientos**: Serie de pasos ordenados que se siguen para resolver un problema o lograr un objetivo.
- **Automatización**: Hacer que los computadores o las máquinas realicen tareas tediosas o repetitivas.
- **Simulación**: Representar o modelar un proceso.
- **Paralelismo**: Organizar los recursos para que simultáneamente realicen tareas con el fin de alcanzar una meta y objetivo común.

En el documento Computational Thinking Teacher Resouces se presentan nueve EAPC, "Experiencias de Aprendizaje del Pensamiento Computacional".

Las EAPC pretende mostrar actividades del pensamiento computacional de forma divertida y amigable a los maestros y resaltando los conceptos del pensamiento computacional.

Las EAPC pueden utilizar o no un ordenador. Éstas son solo una guía para que cada maestro o docente diseñe sus propias experiencias.

Las actividades sin el uso del ordenador se hacen a través de puzzles, dibujos, etc. Se pueden ver ejemplos para ser utilizados en la siguiente página web: http://csunplugged.org/ Estas actividades sin ordenador pretenden:

- Separar las distracciones y problemas técnicos con los ordenadores.
- Se exploran nuevas formas para resolver problemas con los conceptos de computación.
- Comprender el problema y solo usar la máquina para resolverlo.

En cuanto a las actividades con los ordenadores han surgido aplicaciones que apoyan el aprendizaje del pensamiento computacional en el aula como por ejemplo, Scratch, Snap!...

2.2. SCRATCH

Existen varios entornos de programación diseñados para usuarios noveles en programación. Se pueden destacar Alice (Alice, 2015), Greenfoot (Greenfoot, 2015)

Una de las herramientas más populares en Estados Unidos y en Europa y que se está utilizando en varios colegios es Scratch (principalmente niños de 8 a 16 años). Scratch permite una programación orientada a objetos en un entorno atractivo y amigable para niños, lo que permite desarrollar el pensamiento algorítmico y potenciación de la creatividad en el diseño y desarrollo de los proyectos.

Scratch se construye sobre las ideas del constructivismo de Logo (Kafai and Resnick, 1996), (Papert, 1980) y Etoys (Kay, 2010).

A diferencia de los lenguajes de programación tales como C o Java, con Scratch no se necesita aprender una sintaxis, se utilizan bloques que se arrastran y se anidan de forma intuitiva y divertida. Ante esta situación el usuario se concentra en la lógica del programa. Así Scratch permite la creación de video juegos, animaciones, música y producciones artísticas.

Las principales características de estas aplicaciones son:

- Desarrollar el pensamiento lógico y algorítmico
- Desarrollar métodos para solucionar problemas de forma ordenada
- Tener la posibilidad de obtener resultados complejos a partir de ideas simples
- Aprender y asumir conceptos matemáticos: coordenadas, variables, algoritmos, aleatoriedad
- Aprender los fundamentos de la programación
- Usar distintos medios: sonido, imagen, texto, gráficos
- Posibilitar el aprendizaje colaborativo a través del intercambio de conocimiento

Scratch es un entorno mediante el cual se puede aprender lenguaje de programación de manera sencilla y eficiente, sin necesidad de tener conocimientos sintácticos de programación. Su lema: *programa, crea y juega* nos da una clara idea de cuál es su primera intención.

Los estudiantes al programar y compartir proyectos de Scratch, comienzan a desarrollarse como pensadores computacionales: aprenden conceptos básicos de computación y matemáticas, y a la vez también aprenden estrategias de diseño, resolución de problemas, y otras formas de colaboración.

En este ámbito de aplicaciones los conceptos del pensamiento computacional que son usados son:

- Secuencia
- Iteración
- Paralelismo
- Eventos
- Condicionales
- Operadores
- Datos

Las personas que desarrollan estas técnicas basadas en el ordenador están en disposición de resolver problemas complejos no sólo por sacar provecho de la potencia computacional de los ordenadores sino también por la capacidad de los lenguajes de ordenador en describir sistemáticamente un problema en varias capas de abstracción y de describir el interface entre dichas capas sin ambigüedad. Esta habilidad aumenta la complejidad de los problemas reales para los cuales hay que encontrar una solución buena y eficiente.

El uso de estos bloques no solo facilita su entendimiento sino que además permite, de manera intuitiva, escribir de forma sintácticamente correcta. Esto se debe a que, los denominados bloques, poseen una forma física que los hace distinguibles unos de otros y del mismo modo indican el tipo de construcción que se permite hacer con ese tipo de órdenes. Se trata de un lenguaje que usa Scripts y Sprites.

La siguiente tabla hace una comparativa entre el trabajo en las aulas con Scratch y la manera tradicional. Muestra las ventajas que ofrece Scratch.

El trabajo con Scratch	El trabajo en un aula tradicional
El estudiante es activo	El estudiante es pasivo
Comunicación e intercambio de ideas entre estudiantes	Trabajo individual con pocas posibilidades de compartir
El estudiante planifica actividades	El estudiante responde a las actividades planificadas por otros
Cada estudiante trabaja en proyectos de su interés	Los estudiantes trabajan en el proyecto asignado por el profesor
El conflicto y el error son necesarios para aprender	El conflicto y el error tienen un carácter negativo, hay que evitarlos
Cada estudiante avanza a su propio ritmo	Todos los estudiantes deben conseguir resultados uniformes
El docente no es depositario de todo el saber, simplemente es guía del proceso de enseñanza/ aprendizaje	El docente es el que sabe y dirige la clase
El estudiante es cada vez más autónomo	El estudiante es totalmente dependiente

Tabla 1. Trabajo con Scratch y aula tradicional

Scratch es un programa con el que se puede trabajar on-line a través de la web http://scratch.mit.edu

A continuación se muestra el interfaz gráfico de Scratch y los conceptos básicos para comprender su uso.

Interfaz de Scratch

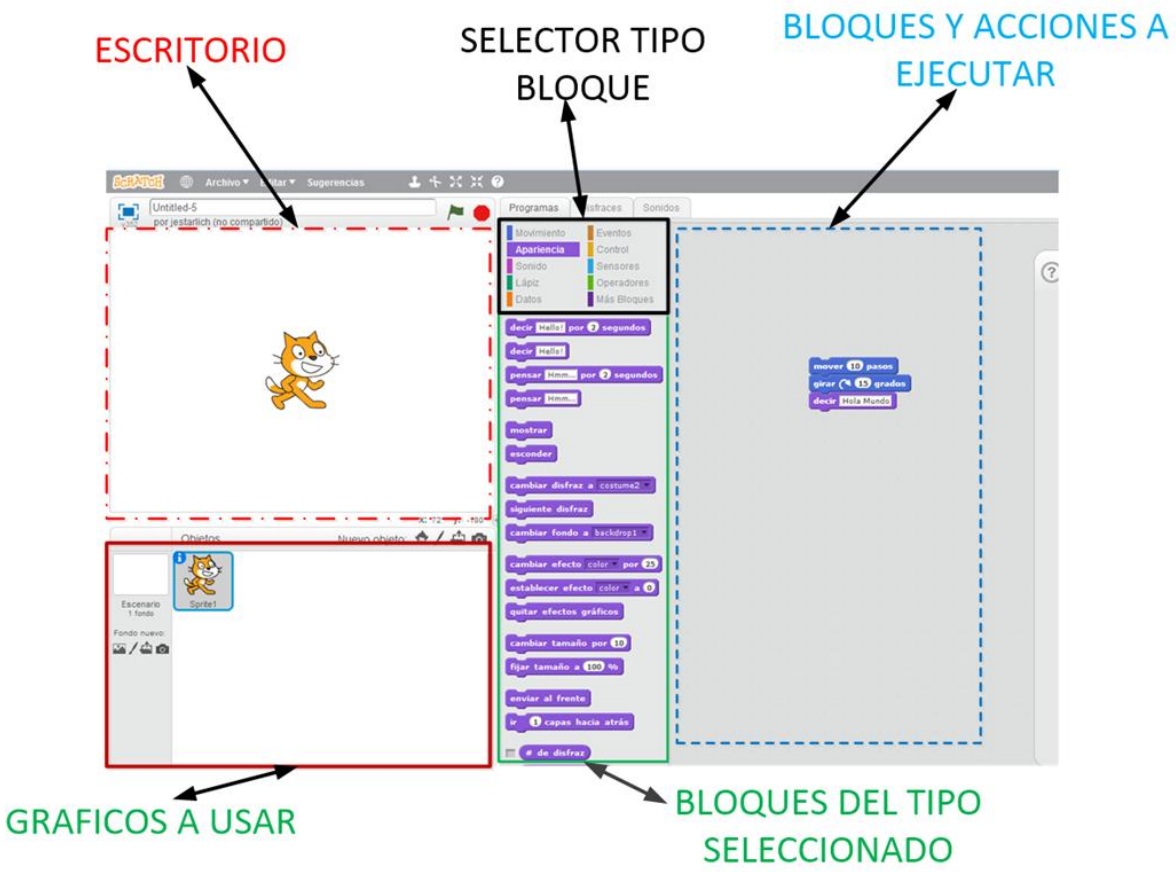

Figura 1 Interfaz de Scratch

Escritorio / Escenario: Es el lugar donde se ve el resultado de nuestras acciones. Por defecto es un fondo blanco y un gato, pero el aspecto gráfico se puede modificar a nuestro gusto.

Objetos / Sprites: Personajes/ gráficos que vamos a usar en nuestro programa o juego.

Propiedades del escenario (*Sprite* del escenario): al seleccionarlo se puede tratar como un *Sprite* más.

Disfraces: distintos gráficos para un *Sprite*.

Sonidos: sonidos de los que dispone el *Sprite*.

Bloques de programación: bloques disponibles dentro de una categoría.

Bandera verde/roja: la bandera verde indica al sistema que comience a ejecutar la programación. El punto rojo detiene la ejecución en cualquier momento.

Scripts (programas): zona donde se preparan los programas de cada *Sprite*.

Bloques en Scratch

Los bloques de Scratch se agrupan en diferentes categorías, se describen a continuación:

- Movimiento: Instrucciones del control de movimiento de un objeto.

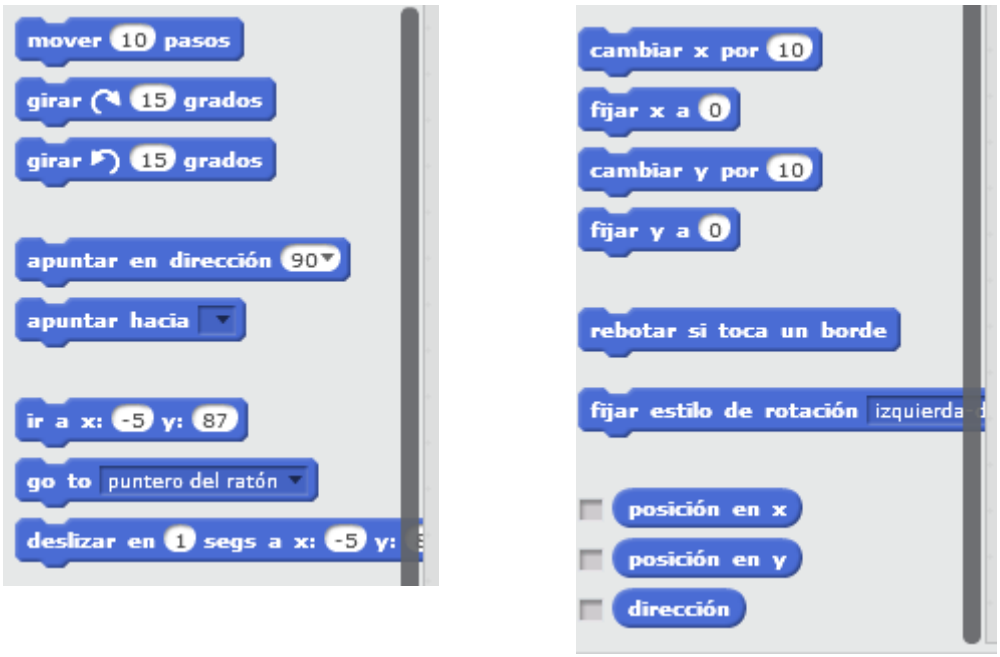

Figura 2 Instrucciones movimiento

- Apariencia: Estos bloques se refieren a los cambios de aspecto del objeto. También tiene bloques de dialogo.

Figura 3Instrucciones apariencia

- Sonido: Bloques relacionados con el sonido del objeto.

Figura 4 Instrucciones de sonido

- Lápiz: Usar el escenario como un lienzo

Figura 5 Instrucciones de lápiz

- Datos: En esta categoría se pueden crear variables globales o locales para un objeto.

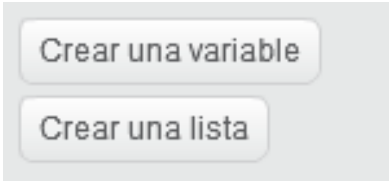

Figura 6 Instrucciones de Datos

- Eventos: Es el grupo más importante de órdenes. Determinan los eventos o acciones del programa o del *Sprite*

Figura 7 Eventos

- Control: En esta categoría se encuentran las instrucciones de control, bucles y condicionales de un objeto.

Figura 8 Instrucciones de control

- Sensores: En esta categoría existen los bloques necesarios para interactuar con hardware y software exterior.

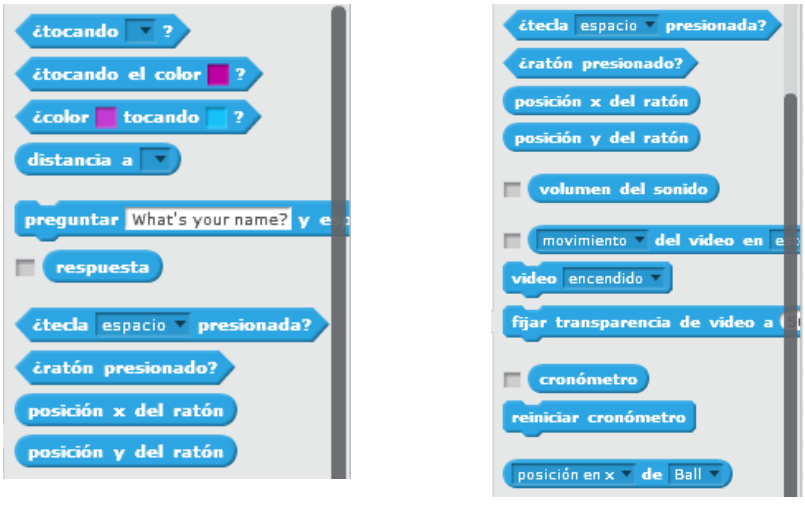

Figura 9 Sensores

- Operadores: Apartado en el que están los bloques de aritmética.

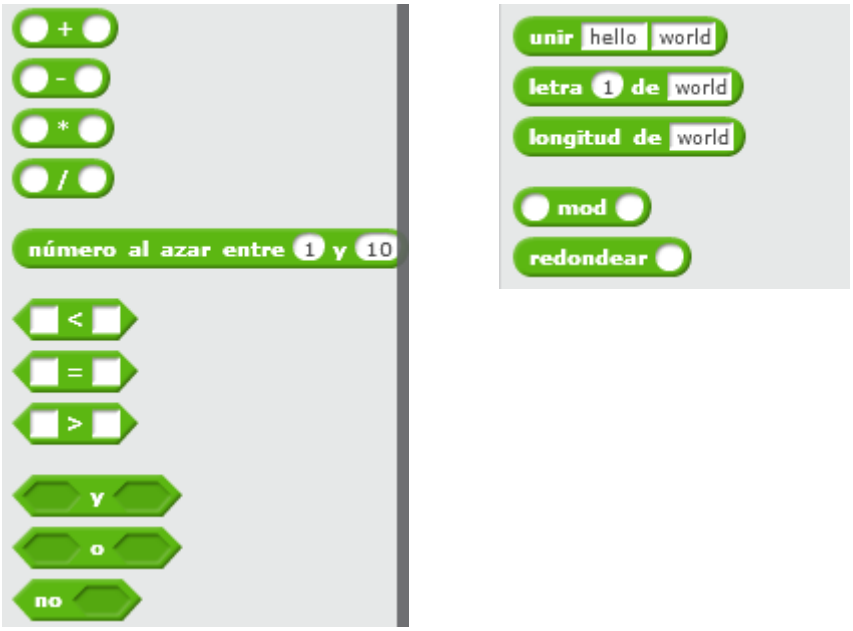

Figura 10 Operadores

2.2.1. ¿CÓMO USAR SCRATCH?

Hay dos elementos fundamentales en Scratch que definir:

- *Sprite* / Objeto: Se trata del personaje u objeto que se sitúa en el escenario y sobre el que se aplican eventos y acciones.
- *Scripts*: Un script es una secuencia de instrucciones que se aplican sobre un objeto.

Lo primero que hay que hacer para empezar con un proyecto, es tener un *Sprite* en el escenario. Para ello se accede a la opción Nuevo objeto, dentro de esta existen varias formas de crear el objeto.

Figura 11 Nuevo Objeto

- Se selecciona un objeto de la biblioteca del proyecto
- La segunda opción permite dibujar un *Sprite* en el editor de Scratch
- La tercera opción permite utilizar un objeto desde un archivo.
- La cuarta opción permite utilizar una foto como objeto.

Una vez que se sitúa un objeto (*Sprite*) en el escenario se añade algún evento, por ejemplo "al presionar bandera verde"

Figura 12 Evento presionar bandera verde

Después de este se puede añadir algún bloque, por ejemplo, de movimiento "mover 10 pasos":

Figura 13 Movimiento10 pasos

Si se pulsa sobre la bandera verde, el objeto se mueve 10 unidades en el eje x.

También se puede dibujar en el escenario para crear una escena donde se encuentra un objeto. Para ello se selecciona la categoría de Lápiz. Dentro de esta categoría existen dos bloques importantes:

Figura 14 Bajar lápiz

Bajar lápiz simula la acción de comenzar a escribir en el papel mientras que subir lápiz simula la acción de levantar la mano del papel. Las demás opciones permiten personalizar el estilo de escritura, desde el color hasta la dureza del trazo.

2.3. SNAP!

Inspirado en Scratch aparece una nueva herramienta Snap!. Se dirige a los estudiantes principiantes y más avanzados mediante la inclusión y la ampliación de las funciones Scratch.

Snap! es una versión extendida de Scratch. Incluye todas las funciones de Scratch y además permite la creación de bloques propios, es decir, se pueden agrupar un conjunto de instrucciones que se vayan a usar de manera periódica en un único bloque para futuros usos e incluso dejarla como parte de nuestra librería. Anteriormente Snap! fue conocido por BYOB (build your own blocks)

En Snap! existe un tipo de bloques que permite crear funciones, lo que da una visión más clara de una programación orientada a objetos. También cuenta con listas y los procedimientos. Estos bloques se muestran a continuación:

Figura 15 Programación orientada a objetos

Los bloques “correr” y “llamar” permiten la ejecución y creación de funciones que, o bien definir dentro del mismo bloque, o bien tenemos definidas aparte gracias a la creación de bloques propios.

El bloque “iniciar” permite la ejecución de scripts independientes al del propio *Sprite*. Gracias a esto se puede simular el funcionamiento de hilos de ejecución, es decir, poner en funcionamiento programas de manera paralela en vez de forma secuencial. Esto permite ejecuciones mucho más eficientes.

Las flechitas que se encuentran en los bloques descritos anteriormente permiten pasar parámetros a las llamadas de la función, por ejemplo en el caso del *for* se le pasa el valor de la *i*.

2.4. ANÁLISIS DE CÓDIGO

El análisis del código permite mejorar la calidad del software. Los analizadores de código fuente ayudan a detectar problemas en el código sin necesidad de ejecutar los programas.

El análisis del código ayuda a detectar incidencias y validar reglas metodológicas que durante el desarrollo no han sido percibidas.

Al contrario que el análisis manual que tiene gran coste, el análisis de código requiere poca inversión de tiempo para localizar problemas.

El análisis automático, puede ser programado y repetido tantas veces como se considere necesario, por lo que se realiza con mayor periodicidad.

Existen dos tipos de análisis de código:

- **Análisis estático**
  Es una inspección directa del código fuente de forma semántica. No se ejecuta el programa.
- **Análisis dinámico**
  Se crean un conjunto de entradas de prueba variada para concluir si el comportamiento y resultado de la prueba es el correcto. Se realiza mediante programas en un procesador real o virtual.

Este proyecto se centra en el análisis estático de código fuente.

Análisis estático de código

"El análisis estático del código es el proceso de evaluar el software sin ejecutarlo"

Esta técnica de análisis estático de código, como se ha comentado anteriormente se ejecuta directamente sobre el código fuente. Procesa si el código es correcto e intenta darnos sugerencias para poder mejorar el código.

Este tipo de análisis se ha desarrollado ampliamente y sobre todo se utiliza en detectar problemas de seguridad.

Por ejemplo, los compiladores de lenguajes de programación (c#, java) son analizadores de código fuente.

Figura 16 Analizadores de código fuente

En 1979 Bell Labs desarrolló Lint, el cual se utilizaba para detectar código C sospechoso con el objetivo de poder detectar errores semánticos en programas, como uso de variables no inicializadas, condiciones invariables y operaciones cuyo resultado quedara fuera de rango.

Hoy en día estas herramientas tienen analizadores léxicos y sintácticos que procesan el código fuente y gracias a un conjunto de reglas que aplican sobre determinadas estructuras si nuestro código fuente posee una estructura concreta que el analizador considere como "mejorable" en base a estas reglas lo indicará y sugerirá una mejora.

Esto sirve para ganar fiabilidad del código, tener una mejor facilidad de mantenimiento y desarrollo de código, así mismo la lectura del código será mejor.

El analizador pretende:

- reducir el rendimiento,
- provocar errores en el software,
- complicar el flujo de datos,
- tener una excesiva complejidad,
- suponer un problema en la seguridad.

Una de las principales preocupaciones de este análisis son la integridad y solidez del código.

Se puede decir que existen dos tipos de análisis estáticos de código.

- El análisis automático que lo realiza un programa que detecta los problemas de código en base a unas reglas que tiene.
- El análisis manual, lo realiza una persona, se centra en la aplicación y considera si las librerías y métodos se utilizan correctamente.

Por lo tanto estos análisis son complementarios, mientras que el automático se centra en la semántica y sintaxis del código, pudiendo ser utilizado en cualquier aplicación, el análisis manual comprueba la estructura de la aplicación y su forma de relacionarse con librerías externas.

La gran diferencia entre el análisis manual y el automático es el tiempo que se tarda en realizar.

Un análisis manual debería ejecutarse cada vez que se añade una nueva funcionalidad a nuestro programa. También sería interesante ejecutar un análisis manual si a la hora de desarrollar una nueva funcionalidad o modificar una existente se requiere más esfuerzo de lo normal.

Figura 17 Análisis manual

Sin embargo, el análisis automático puede ser realizado con mayor periodicidad porque al no ser una persona quien lo realiza, se pude programar y repetirlo.
Además este análisis es un análisis objetivo y siempre devuelve la misma respuesta para un mismo código.

Ventajas del análisis estático automático:

- Se analiza sin ejecutar el software. Necesita menos tiempo.
- Consistente y objetivo. La herramienta no ideas preconcebidas (que normalmente tienen los desarrolladores o revisores).
- Apuntan a la causa raíz, no a los síntomas. Una prueba de penetración puede establecer que hay un problema, pero no su causa final ni cómo corregirlo.
- Detección precoz. La aplicación no tiene que estar integrada ni necesita ejecutarse.
- Su ejecución es barata. Un sistema puede re-analizarse cuando se aplican cambios, o cuando se descubre una nueva vulnerabilidad de aplicación.

Aun así, no se debe creer todo lo que expone el analizador estático puesto que una condición concreta en nuestro código sea correcta. Estos son los llamados falsos positivos.

Desventajas del análisis estático automático:

- Este tipo de análisis no comprueba si el programa responde como se ha programado.
- Puede devolver falsos positivos. Es posible que detecte un error que nosotros hemos programado así conscientemente.
- Como llegamos a ejecutar el programa, necesitamos hacer pruebas de testeo para comprobar si el resultado es el esperado.
- Este tipo de análisis están muy ligados al lenguaje de programación.

¿Qué posibles fallos se pueden detectar con este tipo de análisis?

Fallos de datos:

- Se comprueba si las variables son inicializadas antes de que se utilicen.

- Si todas las constantes tienen nombre.
- Si el tamaño de las *arrays* es el necesario.
- Las cadenas de caracteres tienen delimitadores explícitos.
- Comprueba si los buffer pueden ser desbordados.
- Posibles violaciones de los límites de los vectores
- Variables declaradas pero nunca utilizadas
- Variables asignadas dos veces pero nunca utilizadas entre asignaciones

Fallos de control:

- Comprueba si las condiciones de las instrucciones de condicionales.
- Se comprueba si los bucles terminan
- Se comprueba si instrucciones de tipo case, se valoran todos los casos
- Se comprueba si existe código inalcanzable.

Fallos de entrada/salida:

- Se comprueba si todas las entradas se utilizan
- Si todas las variables de salida se les asignan un valor.

Fallos de interfaz:

- Si todas las llamadas a funciones y a métodos tienen el número correcto de parámetros.
- Si los tipos de parámetros reales concuerdan con los formales.
- Si están en el orden correcto los parámetros.
- Inconsistencias en el tipo de parámetros.
- Los resultados de las funciones no se utilizan.

Defectos de gestión de almacenamiento:

- Si una estructura con punteros se modifica, ¿Se reasignan correctamente todos los punteros?
- Si se asigna correctamente el espacio de memoria
- Se desasigna explícitamente el espacio de memoria cuando ya no se necesita?

Defectos de manejo de excepciones

- Se toman en cuenta todas las condiciones de errores posibles?

Etapas del análisis estático

- Análisis del flujo de control. Comprueba los bucles con múltiples puntos de entrada o salida, encuentra códigos inalcanzables.
- Análisis de uso de los datos. Detecta variables no inicializadas, variables escritas dos veces sin que intervenga una asignación, variables que se declaran pero nunca se usan, etc.
- Análisis de interfaz. Comprueba la consistencia de una rutina, las declaraciones del procedimiento y su uso.

- Análisis de flujo de información. Identifica las dependencias de las variables de salida. No detecta anomalías en sí pero resalta información para la inspección o revisión del código.
- Análisis de caminos. Identifica los caminos del programa y arregla las sentencias ejecutadas en el camino. Nuevamente, es potencialmente útil en el proceso de revisión.
- Ambas etapas generan grandes cantidades de información. Deben utilizarse con cuidado

CAPÍTULO 3
DESCRIPCIÓN TECNOLÓGICA

3. DESCRIPCIÓN TECNOLÓGICA

En esta sección se estudia las tecnologías que se usan y el porqué de su utilización en el diseño del proyecto.

Hoy en día se mandan 3000 millones de correos o 50 millones de tuits. Esto puede parecer impactante pero poco a poco el usuario da por sentado los servicios que usa y cada vez es más exigente con la rapidez y la calidad de los mismos. Pero para el desarrollador la cosa cambia mucho. El desarrollador empieza a entender el back-end de las aplicaciones que está en proceso contantes y acelerado.

Tradicionalmente el modelo de Internet era contar con servidores potentes que hacían todo el trabajo dinámico y generaban todos los archivos HTML que servían al navegador del usuario.

Esto fue así porque los ordenadores de sobremesa no tenían ni memoria ni el procesador suficiente para llevar a cabo las operaciones que hacia el servidor. Pero esto esta cambiando en los últimos años. Se ha incrementado el uso de la red y la capacidad de los ordenadores y además han aparecido nuevos dispositivos, móviles, tablets...

El desarrollo web no puede mantenerse ajeno a este cambio y de ahí que aparezcan los nuevos sistemas como angular.js, ember.js o backbone.js.

Estos sistemas tiene como objetivo mover gran parte del trabajo que realizaba el servidor al cliente. Son aplicaciones construidas completamente en Javascript, lo que implica que el código se ejecuta en el navegador.

Los navegadores ya no son simple marcos que muestran páginas estáticas. Son plataformas capaces de ejecutar aplicaciones complejas. Todo esto es posible gracias a HTML5 y sobre todo Javascript. Este tipo de desarrollos consiste en reducir lo máximo posible las llamadas con el servidor.

Estas aplicaciones son conocidas como *Single- page applications*. Aplicaciones de una sola página que responden rápidamente a las interacciones del usuario. Un buen ejemplo de este tipo de aplicaciones es *Gmail*.

El navegador es donde se ejecuta la aplicación web. Esta aplicación necesita recibir información desde fuera y des un servidor. Un cliente pide información mediante una petición HTTP. El servidor recibe la petición y mediante un lenguaje del lado del servidor (PHP, C# ...) procesa la mayor parte de la lógica y envía la respuesta. Este tipo de arquitectura, solo la vista reside en el cliente y tanto el modelo como el controlador residen en el servidor.

En los frameworks de Javascript MVC del lado del cliente los modelos y los controladores pasan a residir al cliente. La relación entre vista y las otras capas por tanto va a estar en el propio cliente mediante una API.

Las ventajas de este tipo de arquitectura reside en la rapidez hacia el usuario final. Además el desarrollador no necesita conocer otros lenguajes de servidor.

Aun así el servidor sigue desempeñando un papel importante en el desarrollo de aplicaciones y por este motivo aparece NodeJS, un servidor construido íntegramente en Javascript.

3.1. CLIENTE

Cliente es una aplicación informática o un ordenador que consume un servicio remoto en otro ordenador conocido como servidor, normalmente a través de una red de telecomunicaciones.
En este caso, el cliente es un interfaz web. Toda interfaz web se construye en base a tres componentes: HTML, CSS y Javacript.

3.1.1. HTML

HTML es el lenguaje que se emplea para el desarrollo de páginas de Internet. Está compuesto por una serie de etiquetas que el navegador interpreta y da forma en la pantalla. HTML dispone de etiquetas para imágenes, hipervínculos que nos permiten dirigirnos a otras páginas, saltos de línea, listas, tablas, etc...

3.1.2. CSS

CSS o hoja de estilos en cascada, es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML.
Este archivo se utiliza para separar el estilo de la pagina web en otro archivo independiente.

Algunas ventajas de utilizar CSS son:

- Al mantener la presentación de un sitio web en un archivo diferente, la actualización del sitio es más rápida y con mayor control.
- Optimización del ancho de banda de la conexión puesto que se pueden definir varios estilos para un mismo elemento de la web.
- Mejora en la accesibilidad del documento, con el uso de CSS ya no es necesario utilizar tablas.

3.1.3. Bootstrap

Hasta hace poco solo se accedía a las páginas web a través de un ordenador de sobremesa, poco a poco han ido apareciendo nuevos dispositivos, móviles, tabletas con distintas resoluciones lo que hace que las páginas web no se vieran correctamente, así nace Bootstrap.

Bootstrap es un framework desarrollado por Twitter para crear interfaces y diseños web responsive basados en HTML5 y CSS3. Bootstrap adapta la interfaz de la aplicación web al dispositivo que la pide.

Boostrap contiene plantillas de menús, formularios, botones... etc.

Es de código abierto y se puede descargar compilado o a través del código fuente original.

3.1.4. Javascript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos.
En ocasiones necesario utilizar algún framework para que facilite el desarrollo de la misma.
En este proyecto se utiliza como framework AngularJS.

3.1.5. AngularJS

Figura 18 AngularJs

Recientemente ha aparecido una oleada de sistemas que han situado a Javascript en otro nivel. AngularJs es uno de ellos pero están otros muchos como BackboneJs o EmberJS. Son frameworks que viene a aportar herramientas y patrones de diseño con lo que Javascript se convierte en un lenguaje capaz de servir como motor de grandes aplicaciones.

Esto tiene sentido ya que los ordenadores modernos son capaces de procesar con velocidad las cosas. Antes el servidor era el que tenía la responsabilidad de enviar el código HTML completo al cliente, ahora la tendencia es que solo envíe los datos y que el cliente (navegador) sea quien los trate y los muestre.

Angular.js fue creado por Miško Hevery en 2009, quien trabaja para Google.

Según Wikipedia, *AngularJS, o simplemente Angular, es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.*

¿Por qué AngularJs?

Aunque existen más de un framework, AngularJS objetivamente es mejor en muchos aspectos:

- Con Angular se requiere escribir menos código. Con respecto a BackboneJs hay muchas mejoras como por ejemplo el "doble binding"
- AngularJS tiene el apoyo de Google y una gran comunidad detrás. Las búsquedas de AngularJs se han disparado mientras que las de otros frameworks no ha mejorado. Esto quiere decir que se puede encontrar mayor documentación y más componentes.

Principales características de AngularJS

- AngularJS mejora HTML. El objetivo es producir un código HTML más fácil de entender a la hora de leerlo. AngularJS viene cargado con herramientas para ofrecer a los desarrolladores la capacidad de hacer HTML enriquecido. Esto es gracias a lo que se conoce como "directiva".
- Promueve patrones de diseño adecuados para aplicaciones web. Implementa el modelo MVC en su variante MVVM. Estos patrones marcan la separación del código en capas.

Anatomía de Angular JS

AngularJS es uno de los frameworks más populares para desarrollar aplicaciones del lado cliente (Frontend) con JavaScript.

El patrón que se usa en Angular, es el conocido Modelo, Vista, Controlador.

- Vistas: Será el HTML y todo lo que represente datos o información.
- Controladores: Se encargarán de la lógica de la aplicación y sobre todo de las llamadas "Factorías" y "Servicios" para mover datos contra servidores o memoria local en HTML5.
- Modelo de la vista: En Angular el "Modelo" es algo más de aquello que se entiende habitualmente cuando te hablan del MVC tradicional, osea, las vistas son algo más que el modelo de datos. En modo de ejemplo, en aplicaciones de negocio donde tienes que manejar la contabilidad de una empresa, el modelo serían los movimientos contables. Pero en una pantalla concreta de tu aplicación es posible que tengas que ver otras cosas, además del movimiento contable, como el nombre de los usuarios, los permisos que tienen, si pueden ver los datos, editarlos, etc. Toda esa información, que es útil para el programador pero que no forma parte del modelo del negocio, es a lo que llamamos el "Scope" que es el modelo en Angular.

Además de este patrón Angular incluye otro elemento interesante. Los módulos.

Los módulos hacen referencia a la manera que nos va a proponer Angular para que los desarrolladores sean más ordenados.

Por otra parte se puede dividir en dos áreas este framework.

- Parte de HTML. Es la parte declarativa, con las vistas y directivas.
- Parte de Javascript: Que son los controladores, factorías y servicios.

En medio tenemos el denominado Scope que representa al modelo de AngularJs. El *Scope* hace la conexión entre las vistas y los controladores.

Es el enlace que traslada los datos de un lugar a otro sin necesidad de programar nada. Para utilizar AngularJS hay que incluir la librería en la aplicación. Ver Apéndice 1 AngularJS.

3.2. SERVIDOR WEB

Para el servidor web se ha elegido NodeJS por sus prestaciones y porque es muy fácil de llamar desde AngularJS.

Ver Apéndice 1. Instalación NodeJS

3.2.1. Node JS

Figura 19 NodeJS

Node es un programa de servidor.

Según Wikipedia, *Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMA Script, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.*

Node.js es un entorno Javascript del lado del servidor, basado en eventos. Node.js permite correr código JavaScript en el backend, fuera del cliente. Para ejecutar el código JavaScript en el backend, este necesita ser interpretado y ejecutado, Esto es lo que Node.js realiza, haciendo uso de la Maquina Virtual V8 de Google, el mismo entorno de ejecución para JavaScript que Google Chrome utiliza.

Además, para Node.js existen una gran variedad de módulos útiles, entre ellos Express que se utiliza en este proyecto.

3.2.2. Express

Para ayudar a crear aplicaciones web más fácilmente nació Express. Este *framework* está escrito en JavaScript para Node.js. Su objetivo es que no tengamos que reinventar la rueda cada vez que queramos crear una aplicación web, ofreciéndonos soporte para las principales necesidades en este tipo de aplicaciones: gestión de peticiones y respuestas, cabeceras, rutas, vistas...

3.2.3. REST

La Transferencia de Estado Representacional (Representational State Transfer) o REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

3.3. BASE DE DATOS

Para la base de datos se ha elegido MongoDb una base de datos NoSQL por sus prestaciones y porque es muy fácil de llamar con NodeJS mediante Moongose.

3.3.1. MongoDB

Figura 20 MongoDB

MongoDB es una base de datos NoSQL, este tipo de bases de datos difieren de las bases relacionales clásicas. NoSQL persigue almacenar y administrar grandes cantidades de información de una manera eficaz. Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que **no es necesario seguir un esquema**

MongoDB es una base de datos orientada a documentos. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

MongoDB es especialmente útil en entornos que requieran escalabilidad. Con sus opciones de replicación y sharding, que son muy sencillas de configurar, podemos conseguir un sistema que escale horizontalmente sin demasiados problemas.

MongoDB ha sido creado para brindar escalabilidad, rendimiento y gran disponibilidad, escalando de una implantación de servidor único a grandes arquitecturas complejas de centros multidatos.

Características principales son:

- Consultas Ad hoc
- Indexación
- Replicación
- Balanceo de carga
- Almacenamiento de archivos
- Agregación
- Ejecución de JavaScript del lado del servidor

Ver Apéndice 1. Instalación MongoDB

3.3.2. Mongoose

MongooseJs se utiliza para modelar los datos de las base de datos NoSQL.Con mongoose creamos Schema (Esquemas). Estos esquemas, están dados en Json.

Ejemplo de esquema de mongoose:

```
var esquemaUsuarios = new Schema({
Nombre : String,
        Apellidos :String,
        Nick : String,
        Contrasena :Number,
        IdRol :String,
        IdClase:[{type: Schema.Types.ObjectId, ref: 'Clase' }],
        IdGrupo :[{type: Schema.Types.ObjectId, ref: 'Grupo' }],
        NotaUltimoEjercicio : String ,
        NotaFinal:String,
        Administrador:Boolean
},{collection:'Usuarios'});
var Usuario = mongoose.model('Usuario', esquemaUsuarios);
module.exports = Usuario;
```

Esto permite una mayor modularidad en el código.

CAPÍTULO 4

DISEÑO E IMPLEMENTACIÓN

4. DISEÑO E IMPLEMENTACIÓN

El proyecto consiste en una aplicación web orientada a los profesorado de la ESO para la validación de programas creados en Snap! bajo la visión de las características del pensamiento computacional.

4.1. OBJETIVOS DE LA HERRAMIENTA

La herramienta creada valora los siguientes factores dentro del proyecto de Snap!:

- Comprobar si los personajes del proyecto han sido renombrados con un nombre descriptivo. Nombre inadecuado es todo aquel que empieza por "*Sprite*", ejemplo, *Sprite*1, objeto1, etc..

Figura 21 Objetos

- Comprobar si en el programa creado existe **código muerto**, es decir, se trata de código fuente que se ejecuta pero sus resultados nunca se usan. Por ejemplo, bloques que no comienzan por un bloque sombrero.
  Código muerto:

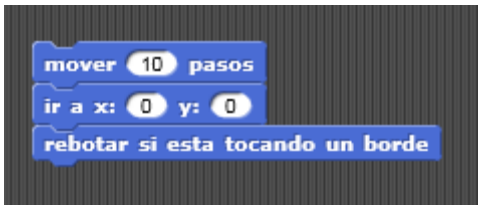

Figura 22 Código muerto

- Comprobar si existe **código duplicado** cuando puede ser creado en un método aparte reutilizable. Al crear un método separado el mantenimiento del código es más fácil.

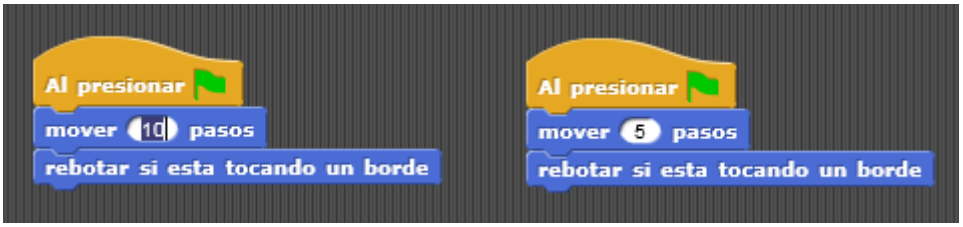

Figura 23 Código duplicado

- Comprobar el **paralelismo**, es decir, la posibilidad de que varias cosas ocurran al mismo tiempo. Por ejemplo, que dos personajes realicen una acción al mismo tiempo, o que un personaje haga varias cosas a la vez.
  Se divide en tres niveles de puntuación según las instrucciones que se utilicen en el programa.

1 punto: Cuando en el programa no existen bloques que comiencen a ejecutarse en el mismo momento. Por ejemplo, si existen al menos dos bloques de "presionar bandera verde"

Figura 24 Paralelismo 1

2 puntos: Cuando en el programa existen al menos dos bloques del tipo "presionar una tecla" , dos scripts en ‘when I am’

Figura 25 Paralelismo 2

3 puntos: Cuando en el programa existen al menos dos bloques del tipo "cuando se reciba un mensaje"

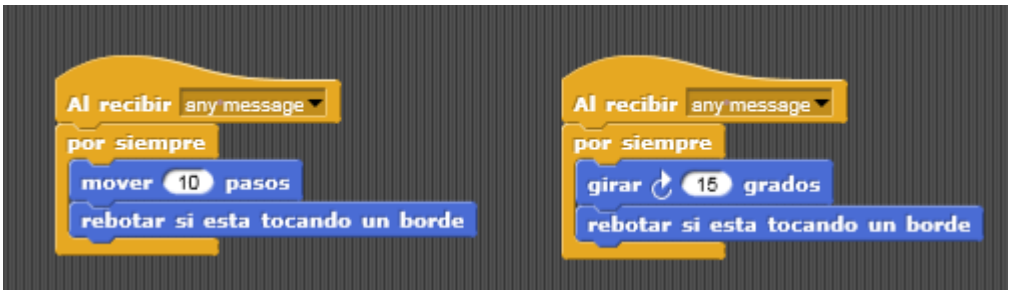

Figura 26 Paralelismo 3

- Se valora el **pensamiento lógico**, esto es, si hay instrucciones que el personaje actué de diferente manera según una condición. Ejemplo de estas instrucciones es el *"si"*
  Se divide en tres niveles de puntuación según la complejidad de las instrucciones:
  1 punto: Si utiliza la sentencia *"si"*

Figura 27 Pensamiento lógico 1

2 puntos: Si el programa utiliza la sentencia *"si/sino"*. Este bloque evalúa la condición y según el resultado, si es verdadera ejecuta una instrucción y sino otra.

Figura 28 Pensamiento lógico 2

3 puntos: En ocasiones es necesario evaluar más de una condición a la vez, en este caso, se utilizan las operaciones lógicas

Figura 29 Pensamiento lógico 3

- Comprobar el nivel de **control de flujo**, si existen instrucciones que repitan ciertos bloques, por ejemplo, instrucción *"repetir hasta"*
  Se divide en tres niveles según la complejidad del programa
  1 punto: Valida cuando el programa ejecuta bloques uno a uno

Figura 30 Control de flujo 1

2 puntos: Si utiliza en el programa instrucciones de repetición

Figura 31 Control de flujo 2

3 puntos: Cuando utiliza la instrucción *repetir hasta que*

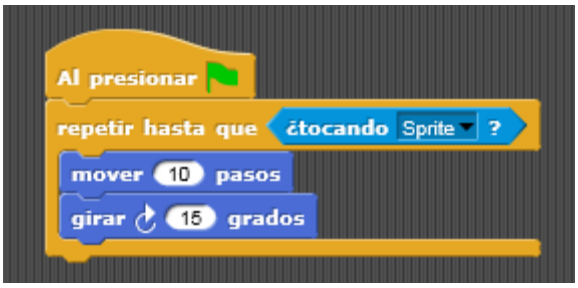

Figura 31 Control de flujo 3

- Comprobar la **interactividad** con el usuario, es decir, el usuario que lo está ejecutando el programa puede interactuar , por ejemplo, moviendo el personaje, respondiendo preguntas, etc..
  Se divide en tres niveles según la complejidad del programa:
  1 punto: Cuando el programa comienza con un bloque sombrero del tipo "Al presionar bandera verde"

Figura 32 Interatividad 1

2 puntos: Cuando el programa utiliza bloques del tipo "Al presionar la tecla"

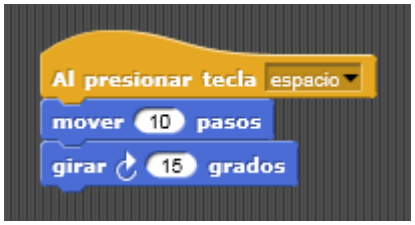

Figura 33 Interatividad 2

3 puntos: Cuando el programa utiliza bloques de sonido

Figura 34 Interatividad 3

- **Representación de datos**, se valora las instrucciones tipo cambio de disfraz, de tamaño...
  Se divide en tres niveles según la complejidad del programa:

1 punto: Cuando no utiliza el programa ningún tipo de variables

Figura 35 Representación de datos 1

2 puntos: Cuando el programa define variables para almacenar valores

Figura 36 Representación de datos 2

3 puntos: Cuando el programa define variables de tipo lista

Figura 37 Representación de datos 3

- Comprobar el nivel de **abstracción**, la abstracción consiste en dividir un problema en partes más pequeñas. Lo ideal es que el comportamiento del personaje sea controlado por diferentes programas y que cada uno de estos programas se ocupe de una cuestión concreta
  Se divide en tres niveles según la complejidad del programa:
  1 punto: Cuando utiliza un único programa

Figura 38 Representación de datos 1

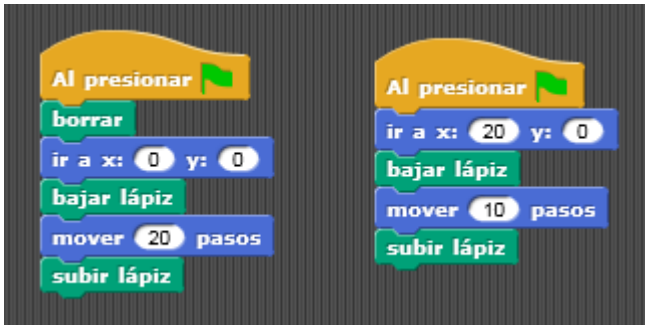

Figura 39 Representación de datos 2

2 puntos: Snap! permite crear bloques personalizados que se pueden llamar desde cualquier parte del programa.
Se define un bloque para posteriormente utilizarlo en el programa.

Figura 40 Representación de datos 3

3 puntos: Comprueba si el programa utiliza clones. Se crea un objeto y a partir de él se pueden crear otros con los mismos atributos.

Figura 41 Representación de datos 4

- Comprobar el nivel de instrucciones de **sincronización**, permiten organizar a nuestros personajes para que las cosas ocurran en el orden que nosotros queremos.
  Se divide en tres niveles según la complejidad del programa:
  1 punto: Cuando el programa contiene el bloque "esperar x segundos". Es la forma más fácil de sincronización.

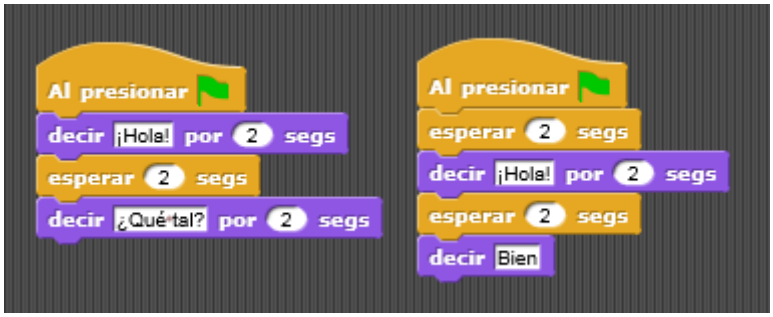

Figura 42 Sincronización 1

2 puntos: Cuando se hace uso del envío de mensajes en el programa

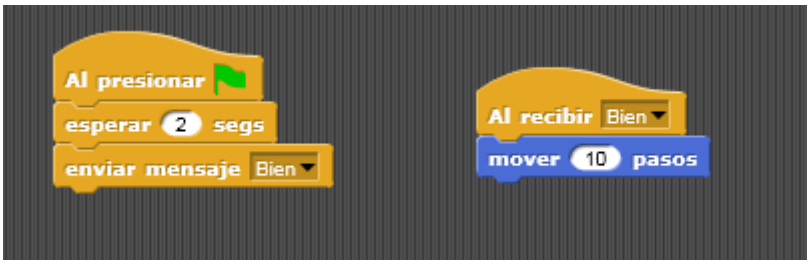

Figura 43 Sincronización 2

3 puntos: Cuando el programa utiliza bloques del tipo “esperar hasta que” o “cuando el fondo cambie a...”.

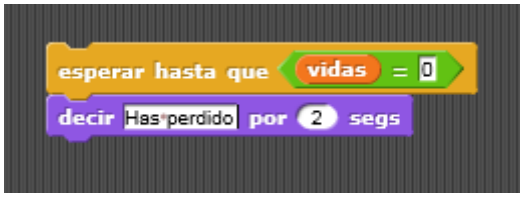

Figura 44 Sincronización 3

En la siguiente tabla se pude ver un resumen de los objetivos segun el pensamiento computacional

Pensamiento computacional	1 punto	2 puntos	3 puntos
Paralelismo	Dos scripts con bandera verde	Dos scripts en 'tecla presionada', dos scripts en 'when I am' el mismo objeto	Dos scripts en 'cuando reciba mensaje'
Pensamiento lógico	Sí	Si – sino	Más de una condición en el Sí
Control de flujo	Secuencia de bloques	Repetir, por siempre	Repetir hasta
Interactividad con el usuario	Bandera verde	Tecla presionada, bloques con ratón	Sonido
Sincronización	Esperar	Enviar, cuando reciba mensaje, parar todos	Esperar hasta
Abstracción	Más de un programa	Definición de bloques propios	Uso de clones
Representación de datos	Modificadores de propiedades de objetos	Operaciones con variables	Operaciones con listas

Tabla 2 Resumen

4.2. PERFILES DE USUARIO

Se han establecido cuatro perfiles para el acceso a la herramienta que se definen a continuación:

- **Usuario anónimo:** Usuario general que tiene acceso a la página inicial. Puede subir un archivo de Snap! y validar su programa según los criterios. No guarda ningún histórico de los ejercicios validados.

- **Alumno/a:** Este usuario posee un nick y una contraseña para acceder a la aplicación. Este alumno o alumna está relacionado con una clase y unos ejercicios determinados. Guarda el histórico de la puntuación de cada ejercicio. Además los usuarios pueden formar grupos. Dos usuarios forman un grupo.

- **Profesor/a:** Este usuario accede a la herramienta a través de un nick y contraseña. Este rol puede crear clases, ejercicios, alumnos y alumnas. Pueden acceder a los resultados de los alumnos y alumnas.

- **Administrador/a:** Este usuario accede a la herramienta a través de un nick y contraseña. Este rol puede hacer los mismo que el Profesor/a. Además este rol puede dar de alta profesores y profesoras.

4.3. DIAGRAMA DE CASOS DE USO

Los diagramas de casos de uso explican gráficamente un conjunto de casos de uso de un sistema, los actores y la relación entre estos y los casos de usos.

Se diferencia en varios subsistemas para mayor compresión de los requisitos

4.3.1 Interfaz gráfica

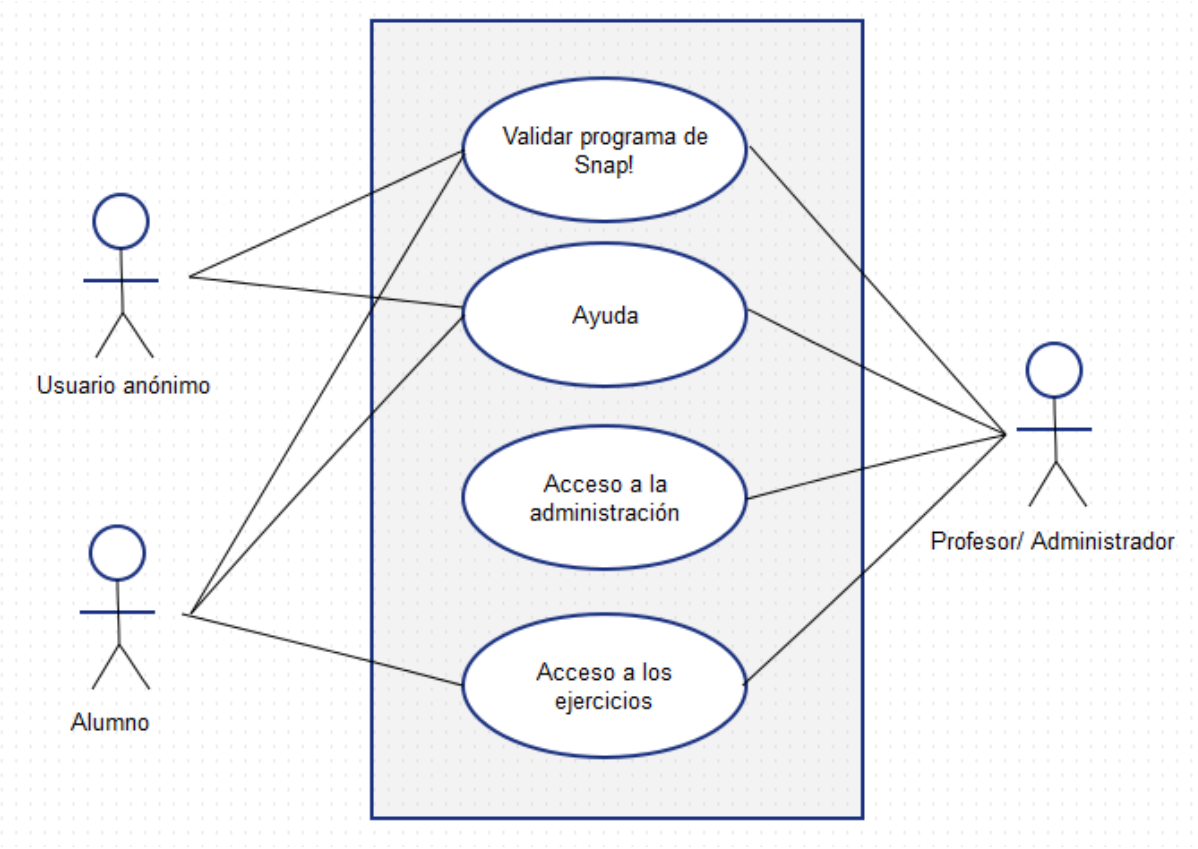

Figura 45 Diagrama caso de uso Interfaz gráfica

4.3.2. Administración - Gestión niveles

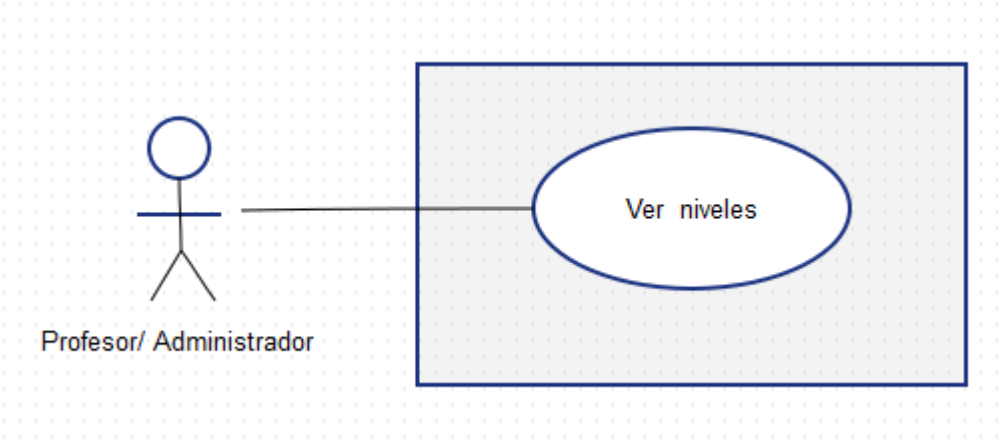

Figura 46 Diagrama caso de uso Niveles

4.3.3. Administración - Gestión de clases

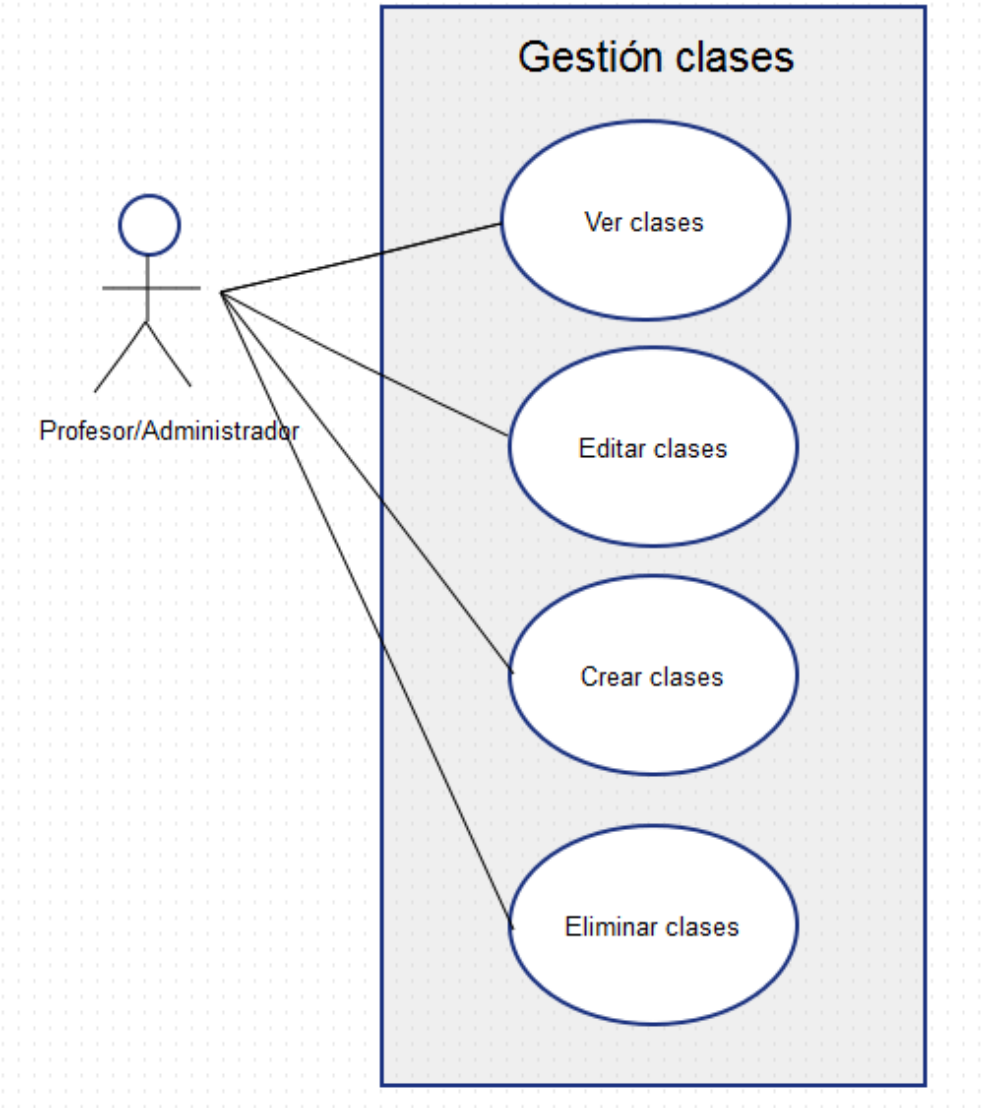

Figura 47 Diagrama caso de uso Gestión clases

4.3.4. Administración - Gestión de grupos

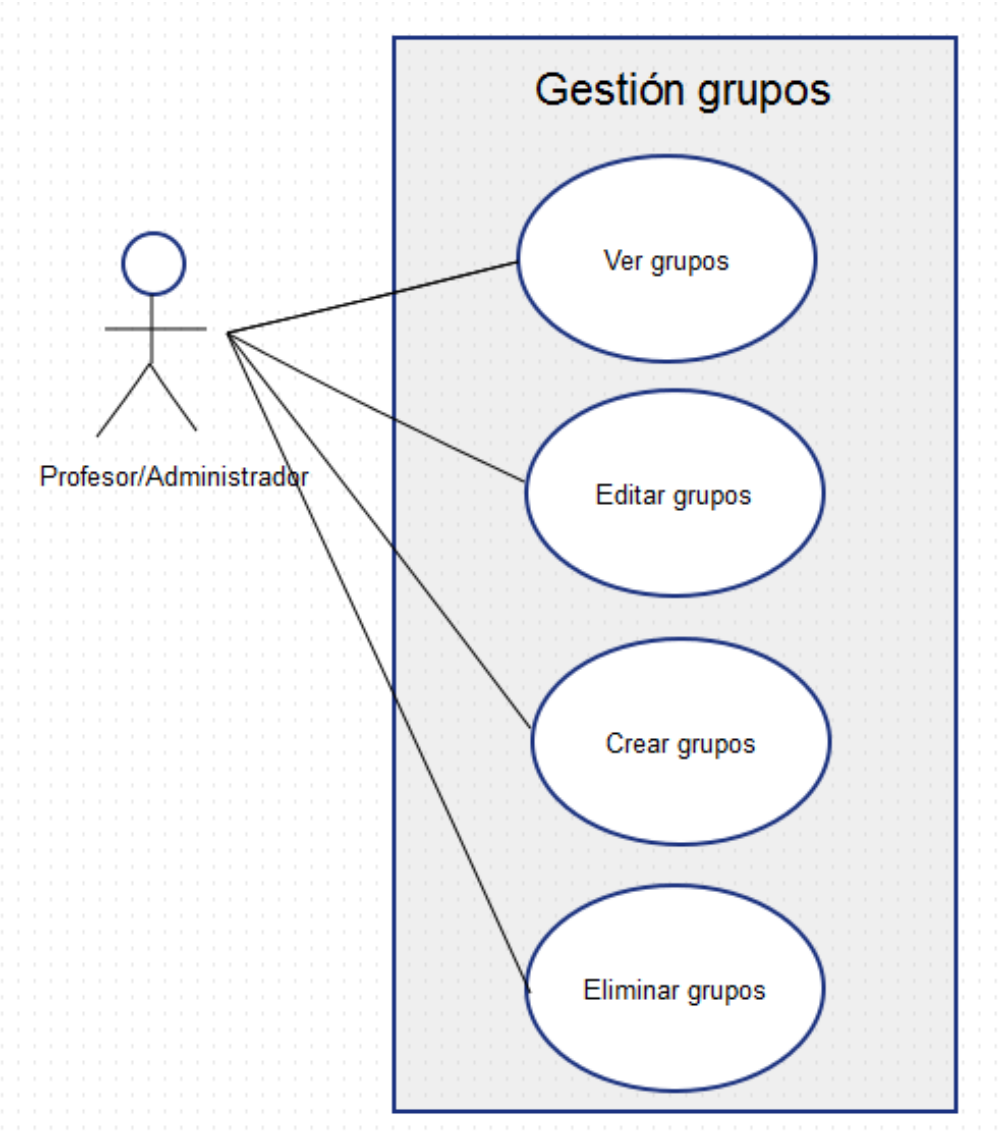

Figura 48 Diagrama caso de uso Gestión grupos

4.3.5. Administración - Gestión de usuarios

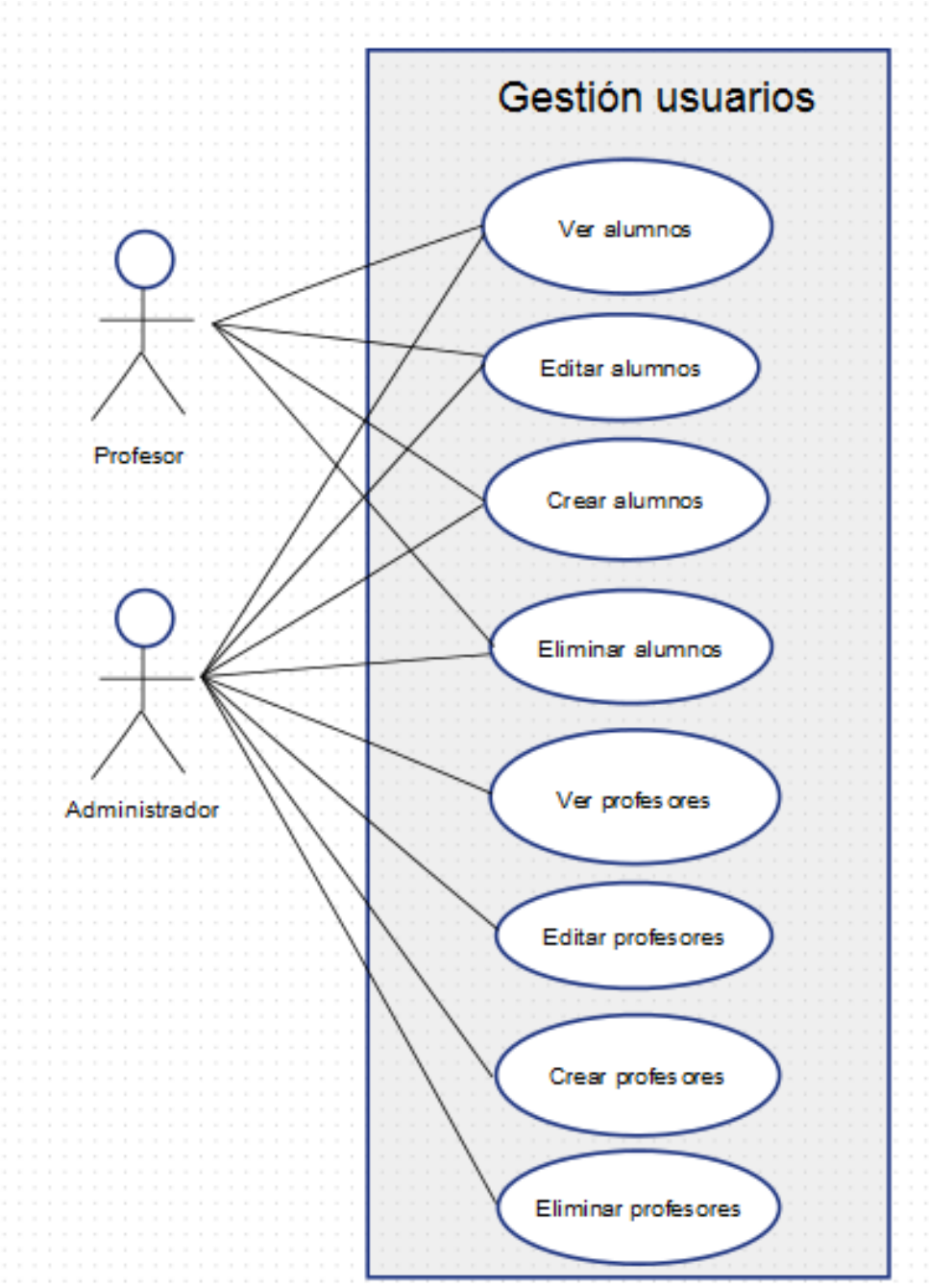

Figura 49 Diagrama caso de uso Gestión usuarios

4.3.6. Administración - Gestión ejercicios

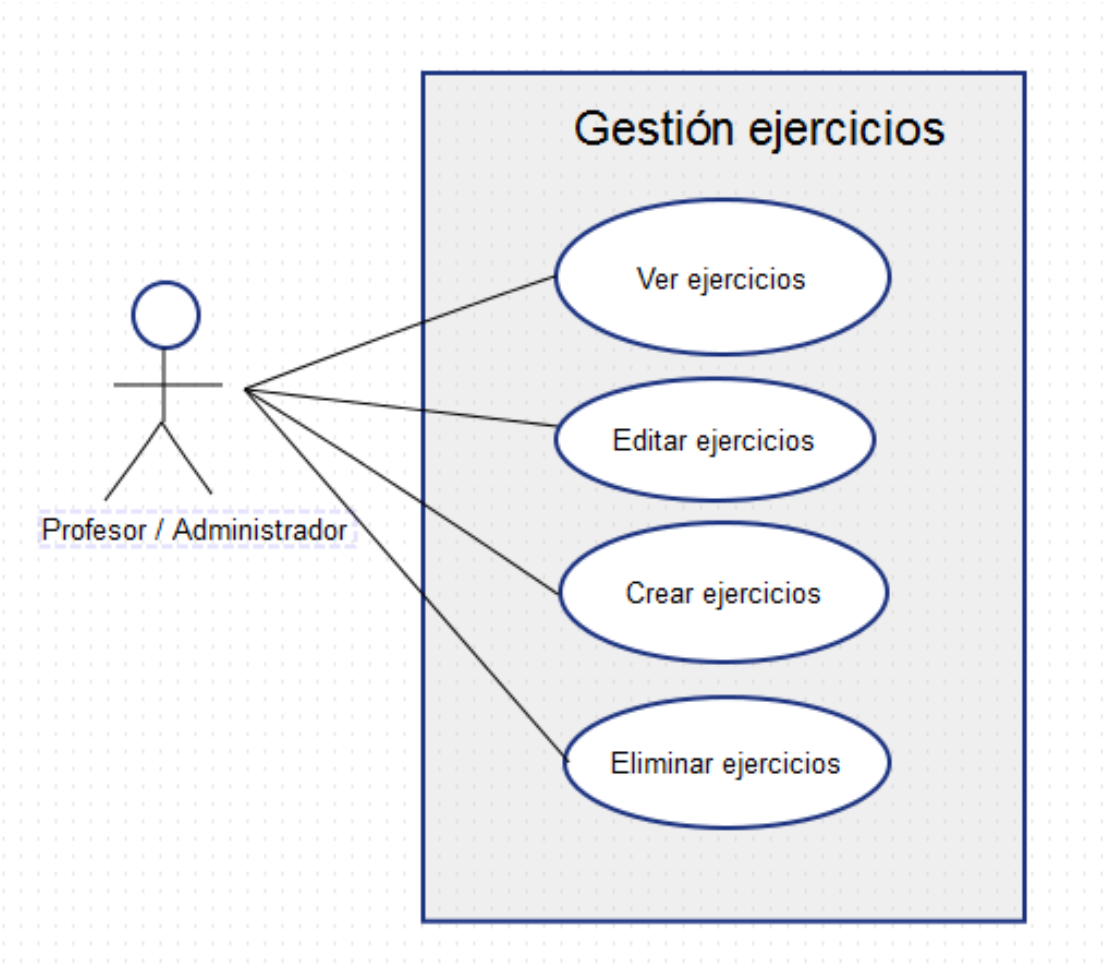

Figura 50 Diagrama caso de uso Gestión ejercicios

4.3.7. Administración - Puntuación

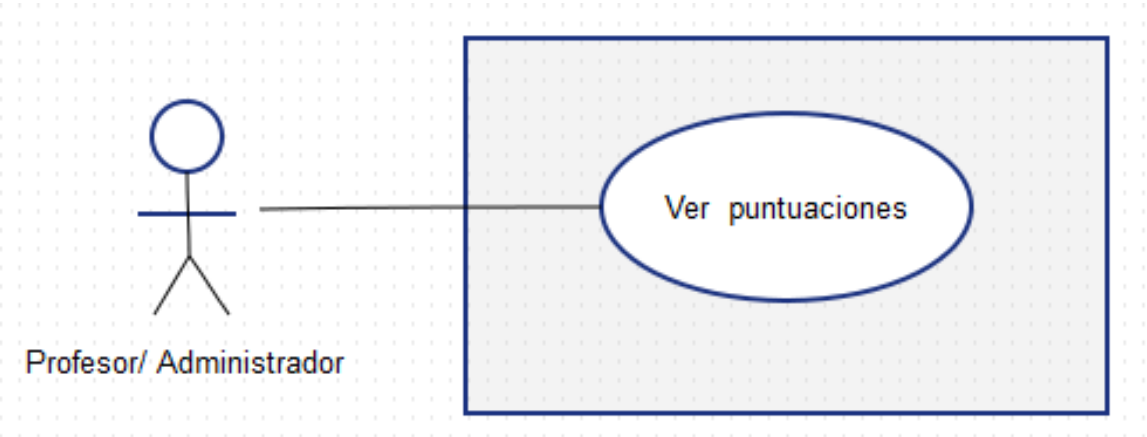

Figura 51 Diagrama caso de uso Gestión puntuaciones

4.4. CASOS DE USO

Un caso de uso es un documento narrativo que describe el comportamiento de un sistema desde el punto de vista de un actor. El actor es una entidad externa del sistema que participa en el caso de uso, en este caso será el usuario o profesor.

A continuación se describen los casos de uso identificados para esta herramienta:

Caso de uso	**Validar programa Snap!**
Actores	Todos los usuarios
Propósito	Evaluar el programa de Snap!
Descripción	El usuario desde la pantalla inicial, sube su programa Snap! con extensión .xml. La herramienta lee el XML y evalúa según los criterios correspondientes.
Curso normal de los eventos	1. El usuario accede a la aplicación y sube un archivo .xml de Snap! 2. La herramienta evalua el programa segun los criterios establecidos en los requirimientos del pensamiento computacional 3. La herramienta muestra al usuario el resultado obtenido de la evaluación
Comentarios	Ninguna

Tabla 3 Caso uso Validar programa Snap!

Caso de uso	**Acceso a la aplicación**
Actores	Usuarios con nick y password
Propósito	Acceder a la aplicación un usuario registrado
Descripción	El usuario registrado accede a la aplicación según el rol que tenga
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña 2. La herramienta muestra los ejercicios según el rol y el usuario
Comentarios	Ninguna

Tabla 4 Caso uso Acceso a la aplicación

Caso de uso	**Acceso a los ejercicios**
Actores	Usuarios con nick y password
Propósito	Acceder a la aplicación un usuario registrado
Descripción	El usuario registrado accede a la aplicación y muestra el siguiente ejercicio para evaluar
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña 2. La herramienta muestra el ejercicio actual a programar y los evaluados anteriormente con la puntuación obtenida. 3. El usuario sube el archivo .xml de Snap! con el ejercicio resuelto 4. La herramienta evalúa el programa y muestra el nuevo ejercicio a realizar

Comentarios	Los ejercicios se muestran según el nivel. Los ejercicios pueden hacerse varias veces. Los ejercicios se muestran según la fecha de visualización y la clase a la que pertenece el alumno y alumna. Al aumentar de nivel se aumenta la dificultad de los ejercicios. El siguiente ejercicio solo se muestra si se ha superado el anterior

Tabla 5 Caso uso Acceso a los ejercicios

Gestión de Niveles

Caso de uso	Gestión Niveles
Actores	Usuarios con rol profesor y administrador
Propósito	Ver los niveles de la aplicación
Descripción	Ver los niveles de la aplicación existentes
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú de Niveles 2. La herramienta muestra los niveles con el número de ejercicios necesarios para pasar de nivel
Comentarios	Existen tres niveles: Bronce, Plata y Oro

Tabla 6 Caso uso Gestión Niveles

Gestión de Clases

Caso de uso	Ver Clases
Actores	Usuarios con rol profesor y administrador
Propósito	Ver las clases de la aplicación
Descripción	El profesor ver sus clases
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú de Clases 2. La herramienta muestra las clases dadas de alta en ese momento

Tabla 7 Caso uso Ver Clases

Caso de uso	Crear Clases
Actores	Usuarios con rol profesor y administrador
Propósito	Crear las clases de la aplicación
Descripción	El profesor crea sus clases
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú de Clases 2. La herramienta muestra las clases dadas de altas en ese momento 3. La herramienta pide los datos para dar de alta la clase (nombre, activo) 4. La herramienta registra la clase si los datos son correctos

Tabla 8 Caso uso Crear Clases

Caso de uso	Editar Clases
Actores	Usuarios con rol profesor y administrador
Propósito	Editar las clases de la aplicación
Descripción	El profesor/a edita sus clases
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú de Clases 2. La herramienta muestra las clases dadas de alta en ese momento 3. El usuario selecciona la clase a editar y modifica 4. La herramienta modifica la clase si los datos son correctos

Tabla 9 Caso uso Editar Clases

Caso de uso	Eliminar Clases
Actores	Usuarios con rol profesor y administrador
Propósito	Eliminar las clases de la aplicación
Descripción	El profesor/a elimina sus clases
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú de Clases 2. La herramienta muestra las clases dadas de alta en ese momento 3. El usuario selecciona la clase a eliminar 4. La herramienta elimina la clase

Tabla 10 Caso uso Eliminar Clases

Gestión de Grupos

Caso de uso	Ver Grupos
Actores	Usuarios con rol profesor y administrador
Propósito	Ver los grupos de la aplicación
Descripción	La herramienta muestra los grupos
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú de Grupos 2. El usuario selecciona una de las clases para ver los grupos asociados 3. La herramienta muestra los grupos de la clase seleccionada

Tabla 11 Caso uso Ver Grupos

Caso de uso	Crear Grupos
Actores	Usuarios con rol profesor y administrador
Propósito	Crear grupos de la aplicación
Descripción	El profesor/a crea los grupos de la aplicación
Curso normal de los eventos	1. El usuario accede al menú de Grupos 2. El usuario selecciona la clase 3. La herramienta muestra los grupos de la clase 4. El usuario añade los datos para dar de alta el grupo 5. La herramienta da de alta el grupo en la clase seleccionada

Comentarios	Un grupo está formado por dos alumnos o alumnas

Tabla 12 Caso uso Crear Grupos

Caso de uso	**Editar Grupos**
Actores	Usuarios con rol profesor y administrador
Propósito	Editar grupos de la aplicación
Descripción	El profesor/a edita los grupos de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú de Grupos 2. El usuario selecciona la clase 3. La herramienta muestra los grupos de la clase seleccionada 4. El usuario selecciona el grupo a editar y modifica sus datos 5. La herramienta edita el grupo

Tabla 13 Caso uso Editar Grupos

Caso de uso	**Eliminar Grupos**
Actores	Usuarios con rol profesor y administrador
Propósito	Eliminar grupos de la aplicación
Descripción	El profesor/a eliminar los grupos de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú de Grupos 2. El usuario selecciona la clase 3. La herramienta muestra los grupos de la clase seleccionada 4. El usuario selecciona el grupo eliminar 5. La herramienta elimina el grupo

Tabla 14 Caso uso Eliminar Grupos

Gestión de Alumnos y alumnas

Caso de uso	**Ver alumnos y alumnas**
Actores	Usuarios con rol profesor y administrador
Propósito	Ver los alumnos y alumnas de la aplicación
Descripción	La herramienta muestra los alumnos y alumnas
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al submenú Usuarios - Alumnos y alumnas 2. El usuario selecciona una de las clases para ver los alumnos y alumnas de esa clase 3. La herramienta muestra los alumnos y alumnas de la clase seleccionada

Tabla 15 Caso uso Ver Alumnos y alumnas

Caso de uso	Crear alumnos y alumnas
Actores	Usuarios con rol profesor y administrador
Propósito	Crear alumnos y alumnas de la aplicación
Descripción	El profesor/a crea los alumnos y alumnas de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al submenú Usuarios - Alumnos y alumnas 2. El usuario selecciona la clase 3. La herramienta muestra los alumnos y alumnas de la clase seleccionada 4. El usuario añade los datos para dar de alta el alumno (Nombre, Apellidos, Nick, Clase, Grupo) 5. La herramienta da de alta el alumno en la clase seleccionada

Tabla 16 Caso uso Crear Alumnos y alumnas

Caso de uso	Editar alumnos y alumnas
Actores	Usuarios con rol profesor y administrador
Propósito	Editar alumnos y alumnas de la aplicación
Descripción	El profesor/a editar los alumnos y alumnas de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al submenú Usuarios - Alumnos y alumnas 2. El usuario selecciona la clase 3. La herramienta muestra los alumnos y alumnas de la clase seleccionada 4. El usuario selecciona y modifica los datos del alumno (Nombre, Apellidos, Nick, Contraseña, Clase, Grupo) 5. La herramienta edita el alumno en la clase seleccionada

Tabla 17 Caso uso Editar Alumnos y alumnas

Caso de uso	Eliminar alumnos y alumnas
Actores	Usuarios con rol profesor y administrador
Propósito	Eliminar alumnos y alumnas de la aplicación
Descripción	El profesor/a elimina los alumnos y alumnas de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al submenú Usuarios - Alumnos y alumnas 2. El usuario selecciona la clase 3. La herramienta muestra los alumnos y alumnas de la clase seleccionada 4. El usuario selecciona el alumno para borrar 5. La herramienta elimina el alumno en la clase seleccionada

Tabla 18 Caso uso Eliminar Alumnos y alumnas

Gestión de Profesores

Caso de uso	Ver Profesores
Actores	Usuarios con rol administrador
Propósito	Ver los profesores de la aplicación
Descripción	La herramienta muestra los profesores
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al submenú Usuarios - Profesores 2. La herramienta muestra los profesores de la aplicación

Tabla 19 Caso uso Ver Profesores

Caso de uso	Crear Profesores / Administradores
Actores	Usuarios con rol administrador
Propósito	Crear profesores de la aplicación
Descripción	El administrador crea profesores y administradores de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al submenú Usuarios - Profesores 2. La herramienta muestra los profesores de la aplicación 3. El usuario añade los datos para dar de alta un profesor(Nombre, Apellidos, Nick, Administrador) 4. La herramienta da de alta el profesor o administrador

Tabla 20 Caso uso Crear Profesores

Caso de uso	Editar Profesores / Administradores
Actores	Usuarios con rol administrador
Propósito	Editar profesores de la aplicación
Descripción	El administrador edita profesores y administradores de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al submenú Usuarios - Profesores 2. La herramienta muestra los profesores de la aplicación 3. El usuario selecciona al profesor y modifica los datos (Nombre, Apellidos, Nick, Administrador, Contraseña) 4. La herramienta edita el profesor o administrador

Tabla 21 Caso uso Editar Profesores

Caso de uso	Eliminar Profesores / Administradores
Actores	Usuarios con rol administrador
Propósito	Eliminar profesores de la aplicación
Descripción	El administrador elimina profesores y administradores de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al submenú Usuarios - Profesores 2. La herramienta muestra los profesores de la aplicación 3. El usuario selecciona al profesor 4. La herramienta elimina el profesor o administrador

Tabla 22 Caso uso Eliminar Profesores

Gestión de Ejercicios

Caso de uso	Ver Ejercicios
Actores	Usuarios con rol profesor y administrador
Propósito	Ver los ejercicios de la aplicación
Descripción	La herramienta muestra los ejercicios de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú Ejercicios 2. El usuario selecciona el idioma 3. La herramienta muestra los ejercicios de la aplicación 4. El usuario selecciona un ejercicio 5. La herramienta muestra las clases a las que pertenece el ejercicio

Tabla 23 Caso uso Ver Ejercicios

Caso de uso	Crear Ejercicios
Actores	Usuarios con rol profesor y administrador
Propósito	Crear los ejercicios de la aplicación
Descripción	El usuario crea los ejercicios de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú Ejercicios 2. El usuario selecciona el idioma 3. La herramienta muestra los ejercicios de la aplicación 4. El usuario añade los datos para dar de alta un ejercicio (Titulo, Ejercicio, FechaFinal, EsEjercicioFinal) 5. La herramienta da de alta el ejercicio 6. El usuario selecciona el ejercicio y lo añade a una nueva clase rellenando los datos necesarios (Clase, Nivel, Orden, FechaVisualización, Fecha entrega) 7. La herramienta asocia el ejercicio a la clase

Tabla 24 Caso uso Crear Ejercicios

Caso de uso	Editar Ejercicios
Actores	Usuarios con rol profesor y administrador
Propósito	Editar los ejercicios de la aplicación
Descripción	El usuario edita los ejercicios de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú Ejercicios 2. El usuario selecciona el idioma 3. La herramienta muestra los ejercicios de la aplicación 4. El usuario selecciona el ejercicio a editar y modifica los datos. 5. La herramienta actualiza la información del ejercicio

Tabla 25 Caso uso Editar Ejercicios

Caso de uso	Eliminar Ejercicios
Actores	Usuarios con rol profesor y administrador
Propósito	Eliminar los ejercicios de la aplicación
Descripción	El usuario elimina los ejercicios de la aplicación
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú Ejercicios 2. El usuario selecciona el idioma 3. La herramienta muestra los ejercicios de la aplicación 4. El usuario selecciona el ejercicio a eliminar 5. La herramienta elimina el ejercicio

Tabla 26 Caso uso Eliminar Ejercicios

Caso de uso	Añadir Clases a un ejercicio
Actores	Usuarios con rol profesor y administrador
Propósito	Añadir clase a un ejercicio
Descripción	Se asocia un ejercicio a una clase
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú Ejercicios 2. El usuario selecciona un idioma 3. La herramienta muestra los ejercicios 4. El usuario selecciona el ejercicio 5. La herramienta muestra las clases a las que está asignado el ejercicio 6. El usuario rellena los datos para dar de alta la clase (Clase, Nivel, Orden, Fecha visualización, Fecha entrega) 7. La herramienta asocia el ejercicio a la clase
Comentarios	Un ejercicio puede ser asignado a varias clases

Tabla 27 Caso uso Añadir Clases a un ejercicio

Caso de uso	Eliminar Clases a un ejercicio
Actores	Usuarios con rol profesor y administrador
Propósito	Eliminar clase a un ejercicio
Descripción	Se asocia un ejercicio a una clase
Curso normal de los eventos	1. El usuario introduce su nick y la contraseña y accede al menú Ejercicios 2. El usuario selecciona un idioma 3. La herramienta muestra los ejercicios 4. El usuario selecciona el ejercicio 5. La herramienta muestra las clases a las que está asignado el ejercicio 6. El usuario selecciona la clase a eliminar

Tabla 28 Caso uso Eliminar Clases a un ejercicio

4.5. ARQUITECTURA

El usuario accede a la herramienta a través de un dispositivo móvil o ordenador personal.

El usuario anónimo a través de la página inicial sube el programa de Snap!. Este archivo es recogido por AngularJS y se llama al método indicado para obtener los resultados obtenidos de evaluar los parámetros del pensamiento computacional.

Para un alumno se procede igualmente que el usuario anónimo con la diferencia que en este caso el método accede a la base de datos para guardar los datos del ejercicio para ese alumno.

Una vez calculados los datos se devuelve a la pantalla inicial que se renderiza según Bootstrap.

El modelo de arquitectura utilizado es el de Cliente - Servidor.

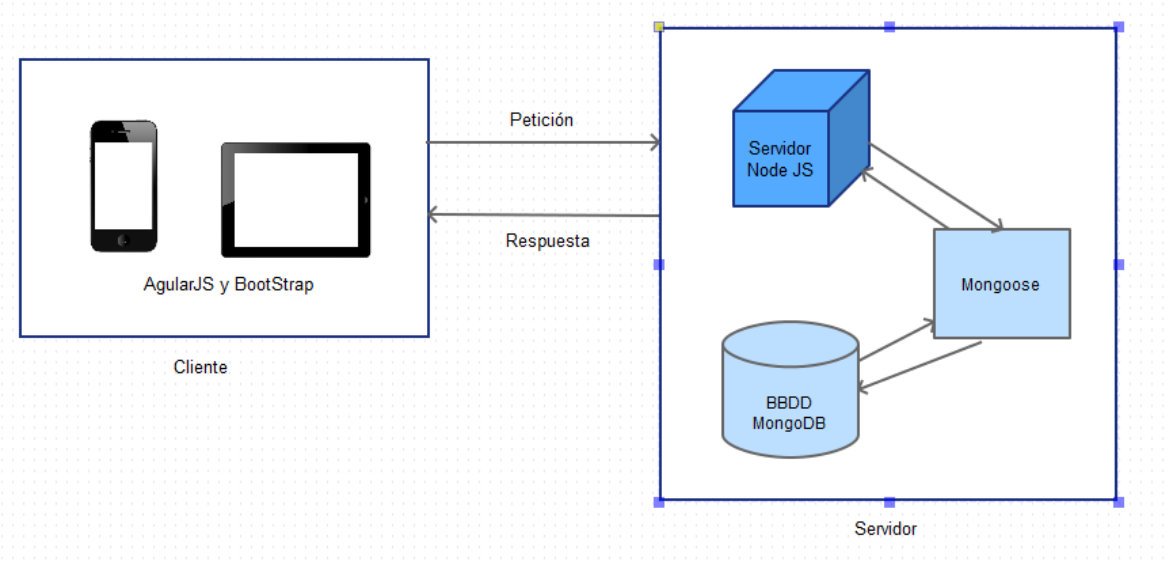

Figura 52 Arquitectura

Se describe cada componente:

- **Cliente**: El acceso a la herramienta es a través de un navegador, Internet Explorer, Chrome, etc… El desarrollo de Front-End se realiza en AngularJS y Bootstrap que permite que el diseño sea responsive y se vea correctamente en los diferentes dispositivos.
- **Servidor**: El servidor se basa en NodeJS, permite la abstracción de la base de datos, el modelo vista controlador y la interacción con el componente Front End.
- **Base de Datos**: La base de datos utilizada es MongoDB. Una base de datos NoSQL.

Abajo se muestra el diagrama visual del flujo que va a seguir aplicación con las tecnologías que emplea en cada parte:

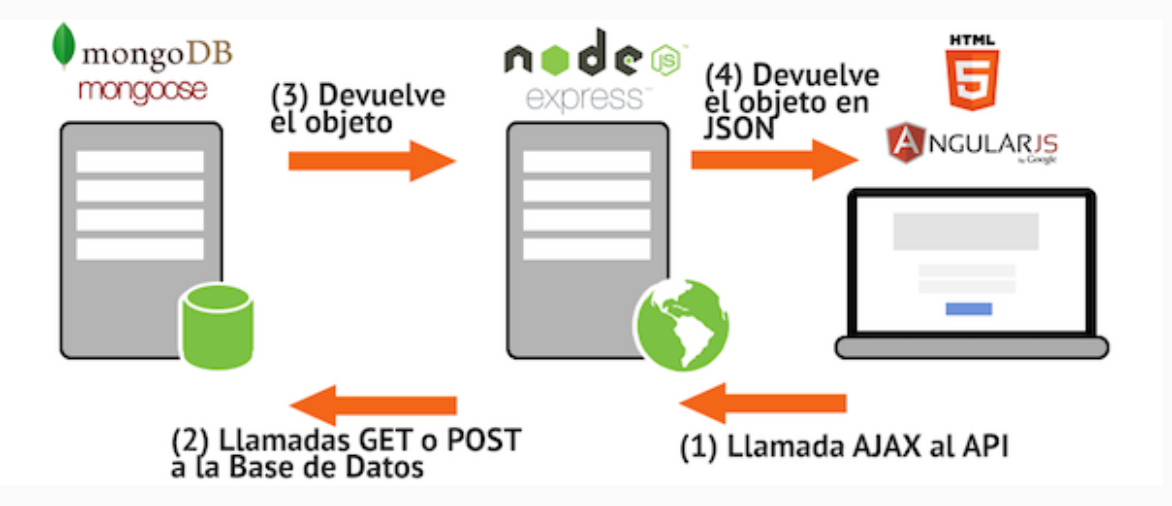

Figura 53 Arquitectura MEAN

Desde el frontend, con Angular se hacen llamadas AJAX a nuestra API en el servidor Node. Este consulta a la base de datos (Mongo) dependiendo de la llamada realizada. La BD devuelve el objeto como respuesta a Node y este lo sirve como JSON a Angular que lo muestra en el frontend sin necesidad de recargar la página, creando así una *Single Page Application*.

A este concepto y las tecnologías que lo posibilitan se les ha bautizado con el nombre de MEAN, acrónimo formado por las iniciales de las cuatro tecnologías principales que entran en juego: MongoDB, Express, AngularJS y Node.js

4.6. DISEÑO E IMPLEMENTACIÓN DEL SERVIDOR

4.6.1 Modelo de datos

A continuación se muestran las colecciones que se utilizan en este proyecto y los scripts necesarios para crearlo:

Colección Idiomas

Idiomas	Tipo
_ id	ObjectID
Idioma	String
Activo	Number

Tabla 29 Idiomas

Script:

```
var esquemaIdioma = new Schema({
        Idioma: String,
        Activo: String
},{collection:'Idiomas'});
```

Inicialmente esta colección tiene los idiomas euskera y castellano.

```
db.Idiomas.insert({
        "Idiomas":"Euskera",
        "Activo":"1"
        });

db.Idiomas.insert({
        "Idiomas":"Castellano",
        "Activo":"1"
        });
```

Colección Roles

Roles	Tipo
_ id	ObjectID
Rol	String

Tabla 30 Roles

Script:

```
var esquemaRoles = new Schema({
        _id:Number,
        Rol: String
},{collection:'Roles'});
```

Inicialmente esta colección contiene tres roles: Alumno, Profesor y Administrador

```
db.Roles.insert({
        "Roles ":"Alumnno"
        });

db.Roles.insert({
        "Rol":"Profesor"
        });

db.Roles.insert({
        "Rol":"Administrador"
        });
```

Colección Niveles

Niveles	Tipo
_ id	ObjectID
Nivel	String
NumeroEjercicios	Number
Activo	Number

Tabla 31 Niveles

Script:

```
var esquemaNiveles = new Schema({
        Nivel: String,
        NumeroEjercicios:Number,
        Activo:Number
},{collection:'Niveles'});
```

Inicialmente esta colección contiene tres niveles: Bronce, Plata y Oro

```
db.Niveles.insert({
        "Nivel":"Bronce",
        "NumeroEjercicios":0,
        "Activo":0
        });

db.Niveles.insert({
        "Nivel":"Plata",
        "NumeroEjercicios":0,
        "Activo":0
        });

db.Niveles.insert({
        "Nivel":"Oro",
        "NumeroEjercicios":0,
        "Activo":0
        });
```

NumeroEjercicios: Se actualiza cada vez que se añade un nuevo ejercicio al nivel

Colección Clases

Clases	Tipo
_ id	ObjectID
Clase	String
Activo	Number

Tabla 32 Clases

Script:

```
var esquemaClases = new Schema({
        Clase: String,
        Activo:Number,
        UsuarioAlta:Number,
        UsuarioModif:Number
},{collection:'Clases'});
```

Colección Ejercicios

Ejercicios	Tipo
_ id	ObjectID
PuntuacionMinima	Number
Final	String
FechaFinal	String
Activo	Number

Tabla 33 Ejercicios

PuntuacionMinima: Puntuación necesaria para pasar al siguiente ejercicio. Inicialmente se establece de un 60%.
Final y FechaFinal: Indica si es el ejercicio de la evaluación final y la fecha en la que se activa.
Script:

```
var esquemaEjercicios = new Schema({
        PuntuacionMinima: Number,
        Final: String,
        FechaFinal: String,
        Activo: Number
},{collection:'Ejercicios'});
```

Colección EjerciciosClases

Este schema contiene una referencia al ejercicio, al nivel y a la clase que está asociada.

EjerciciosClases	Tipo
_ id	ObjectID
IdEjercicio	ObjectID
IdClase	ObjectID
IdNivel	ObjectID
Orden	Number
FechaVisualizacion	String
FechaEntrega	String
Activo	Number

Tabla 34 EjerciciosClases

Un ejercicio se asignará varias clases y a diferentes niveles
Orden: Orden en el que se muestran los ejercicios.
Script:

```
var esquemaEjerciciosClases = new Schema({
        IdEjercicio:[{type: Schema.Types.ObjectId, ref: 'Ejercicio' }],
        IdClase:[{type: Schema.Types.ObjectId, ref: 'Clase' }],
        IdNivel:[{type: Schema.Types.ObjectId, ref: 'Nivel' }],
        Orden:Number,
        FechaVisualizacion: String,
        FechaEntrega:String,
        Activo: Number
},{collection:'EjerciciosClases'});
```

Colección EjerciciosIdiomas

Este schema contiene una referencia al ejercicio, al idioma y a la clase que está asociada. Se guardan ejercicios traducidos en los diferentes idiomas.

EjerciciosIdiomas	Tipo
_ id	ObjectID
IdEjercicio	ObjectID
IdClase	ObjectID
IdEjercicioClase	ObjectID
IdIdioma	ObjectID
Titulo	String
Ejercicio	String

Tabla 35 EjerciciosIdiomas

Script:

```
var esquemaEjerciciosIdiomas = new Schema({
        IdEjercicio:[{type: Schema.Types.ObjectId, ref: 'Ejercicio' }],
        IdEjercicioClase:[{type: Schema.Types.ObjectId, ref: 'EjercicioClase' }],
        IdClase:[{type: Schema.Types.ObjectId, ref: 'Clase' }],
        IdIdioma:[{type: Schema.Types.ObjectId, ref: 'Idioma' }],
        Titulo: String,
        Ejercicio: String
},{collection:'EjerciciosIdiomas'});
```

Colección Grupos

Este schema contiene una referencia la clase que está asociada.

Grupos	Tipo
_ id	ObjectID
IdClase	String
PuntosActuales	Number
Contrasena	String
Activo	Number

Tabla 36 Grupos

Script:

```
var esquemaGrupos = new Schema({
        Grupo:String,
        IdClase: String,
        PuntosActuales: Number,
        Contrasena:Number,
        Activo:Number
},{collection:'Grupos'});
```

Colección Usuarios

Este schema contiene una referencia al grupo, al rol y a la clase que está asociada.

Usuarios	Tipo
_ id	ObjectID
Nombre	String
Apellidos	String
Nick	String
Contrasena	Number
IdRol	String
IdClase	ObjectID
IdGrupo	ObjectID
NotaUltimoEjercicio	String
NotaFinal	String
Administrador	Boolean

Tabla 37 Usuarios

Script:

```
var esquemaUsuarios = new Schema({
        Nombre : String,
        Apellidos :String,
        Nick : String,
        Contrasena :Number,
        IdRol :String,
        IdClase:[{type: Schema.Types.ObjectId, ref: 'Clase' }],
        IdGrupo :[{type: Schema.Types.ObjectId, ref: 'Grupo' }],
        NotaUltimoEjercicio : String ,
        NotaFinal:String,
        Administrador:Boolean
},{collection:'Usuarios'});
```

Colección Puntuación

Este schema contiene una referencia al ejercicioIdioma, al ejercicio, al ejercicioClase y al grupo.

Puntuacion	Tipo
_ id	ObjectID
IdEjercicioIdioma	ObjectID
IdEjercicio	ObjectID
IdEjercicioClase	ObjectID
IdGrupo	ObjectID
Orden	Number
PtosConvencion	Number
PtosScriptsDuplicados	Number
PtosCodigoMuerto	Number
PtosParalelismo	Number
PtosPensamientoLogico	Number

PtosControlFlujo	Number
PtosInteractividad	Number
PtosSincronizacion	Number
PtosAbstraccion	Number
PtosDatos	Number
PtosTotales	Number
Aprobado	Number
FechaEjercicio	String
Xml	String

Tabla 38 Puntuación

Script:

```
var esquemaPuntuacion = new Schema({
        IdEjercicioIdioma:[{type: Schema.Types.ObjectId, ref: 'EjercicioIdioma' }],
        IdEjercicio:[{type: Schema.Types.ObjectId, ref: 'Ejercicio' }],
        IdEjercicioClase:[{type: Schema.Types.ObjectId, ref: 'EjercicioClase' }],
        IdGrupo :[{type: Schema.Types.ObjectId, ref: 'Grupo' }],
        Orden: Number,
        PtosConvencion : String,
        PtosScriptsDuplicados :String,
        PtosCodigoMuerto:String,
        PtosParalelismo :Number,
        PtosPensamientoLogico : Number ,
        PtosControlFlujo:Number,
        PtosInteractividad:Number,
        PtosSincronizacion:Number,
        PtosAbstraccion:Number,
        PtosDatos:Number,
        PtosTotales:Number,
        Aprobado:Number,
        FechaEjercicio:String,
        Xml: String
},{collection:'Puntuacion'});
```

Colección PuntuacionHistorico

Este schema contiene una referencia al ejercicioIdioma, al ejercicio, al ejercicioClase y al grupo.

PuntuacionHistorico	Tipo
_ id	ObjectID
IdEjercicioIdioma	ObjectID
IdEjercicio	ObjectID
IdEjercicioClase	ObjectID
IdGrupo	ObjectID
Orden	Number
PtosConvencion	Number
PtosScriptsDuplicados	Number
PtosCodigoMuerto	Number
PtosParalelismo	Number

PtosPensamientoLogico	Number
PtosControlFlujo	Number
PtosInteractividad	Number
PtosSincronizacion	Number
PtosAbstraccion	Number
PtosDatos	Number
PtosTotales	Number
Aprobado	Number
FechaEjercicio	String
Xml	String

Tabla 39 Puntuación Histórico

Script:

```
var esquemaPuntuacionHistorico = new Schema({
        IdEjercicioIdioma:[{type: Schema.Types.ObjectId, ref: 'EjercicioIdioma' }],
        IdEjercicio:[{type: Schema.Types.ObjectId, ref: 'Ejercicio' }],
        IdEjercicioClase:[{type: Schema.Types.ObjectId, ref: 'EjercicioClase' }],
        IdGrupo :[{type: Schema.Types.ObjectId, ref: 'Grupo' }],
        PtosConvencion : String,
        PtosScriptsDuplicados :String,
        PtosCodigoMuerto:String,
        PtosParalelismo :Number,
        PtosPensamientoLogico : Number ,
        PtosControlFlujo:Number,
        PtosInteractividad:Number,
        PtosSincronizacion:Number,
        PtosAbstraccion:Number,
        PtosDatos:Number,
        PtosTotales:Number,
        FechaEjercicio:String,
        Xml: String
},{collection:'PuntuacionHistorico'});
```

4.6.2 Diagrama de la base de datos

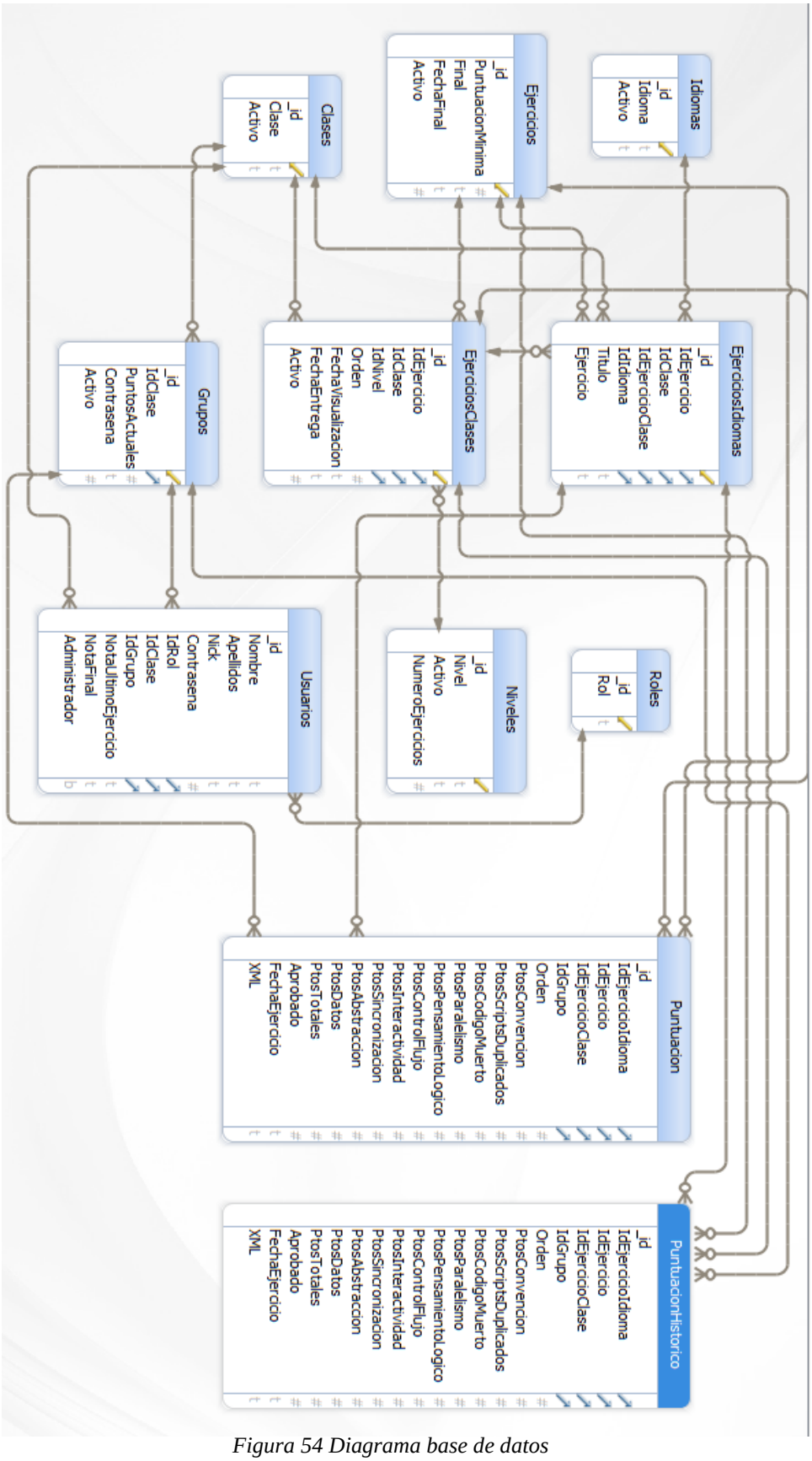

Figura 54 Diagrama base de datos

4.6.3 Comunicación cliente - servidor. API REST

Para llevar a cabo la comunicación entre el cliente y el servidor se ha implementado una API REST.

REST, Representational State Transfer, es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP.

Existen tres niveles de calidad a la hora de aplicar REST en el desarrollo de una aplicación web:

- Uso correcto de URIs
- Uso correcto de HTTP.
- Implementar Hypermedia.

1. Uso correcto de URIs

Las URLs nos permiten acceder a cada una de las páginas, secciones o documentos del sitio web.

Se llama recurso a la información a la que se accede o se quiere modificar o borrar, independientemente de su formato.

Las URL, Uniform Resource Locator , son un tipo de URI, Uniform Resource Identifier, que permite identificar de forma única el recurso y permite localizarlo para poder acceder a él o compartir su ubicación.

La URL tiene el siguiente formato: {protocolo}://{dominio o hostname}[:puerto (opcional)]/{ruta del recurso}?{consulta de filtrado}

Las reglas básicas a tener en cuenta para poner nombre a una URI son las siguientes:

- Los nombres de URI no deben implicar una acción, por lo tanto debe evitarse usar verbos en ellos.
  Por ejemplo, una dirección del tipo http://localhost/usuario/1/editar es incorrecta puesto que tiene el verbo editar en ella.
- Deben ser únicas, no deben tener más de una URI para identificar un mismo recurso.
- Deben ser independiente de formato. Por ejemplo, una URI tipo http://localhost/usuario/1.pdf no es correcta porque identifica el formato .pdf
- Deben mantener una jerarquía lógica. Por ejemplo, una URI tipo http://localhost/usuario/1/grupo/10 no es correcta porque no sigue la jerarquía lógica
- Los filtrados de información de un recurso no se hacen en la URI.Para filtrar, ordenar, paginar o buscar información en un recurso, debemos hacer una consulta sobre la URI, utilizando parámetros HTTP en lugar de incluirlos en la misma.

2. Uso correcto HTTP

HTTP tiene varios métodos que se usan para manipular recursos:

- GET: Para consultar y leer recursos

- POST: Para crear recursos
- PUT: Para editar recursos
- DELETE: Para eliminar recursos.
- PATCH: Para editar partes concretas de un recurso

Generalmente se utilizan los métodos GET y POST.

3.Implementar Hypermedia

Hypermedia permite conectar mediante vínculos las aplicaciones clientes con las APIs, permitiendo a dichos clientes despreocuparse por conocer de antemano del cómo acceder a los recursos.

A continuación se muestran las URI utilizadas en el proyecto para su correcto funcionamiento:

Login

URL	/login
Método	POST
Parámetros	Nick y Contraseña
Descripción	Devuelve si el usuario esta registrado y puede acceder a la aplicación

Tabla 40 Url Login

Niveles

URL	/niveles
Método	POST
Parámetros	Ninguno
Descripción	Devuelve el listado de los niveles de la apliccación

Tabla 41 Url Niveles

Clases

URL	/clases
Método	GET
Parámetros	Ninguno
Descripción	Devuelve el listado de las clases

Tabla 42Url Get Clases

URL	/clases
Método	PUT
Parámetros	Clase, usuarioAlta y activo
Descripción	Agrega una nueva clase

Tabla 43 Url Put Clases

URL	/clases
Método	POST
Parámetros	Id, clase, usuarioModif y activo
Descripción	Modifica una clase determinada

Tabla 44 Url Post Clases

URL	/clasesactivas
Método	GET
Parámetros	Ninguno
Descripción	Devuelve el listado de las clases activas

Tabla 45 Url Get ClasesActivas

URL	/clasesusuario
Método	GET
Parámetros	Usuario y rol
Descripción	Devuelve el listado de las clases de un usuario

Tabla 46 Url Get ClasesUsuario

URL	/clases/:id
Método	DELETE
Parámetros	Id
Descripción	Elimina la clase con el Id enviado

Tabla 47 Url Delete Clases

Grupos

URL	/grupos
Método	GET
Parámetros	IdClase
Descripción	Devuelve el listado de los grupos de la clase enviada

Tabla 48 Url Get Grupos

URL	/grupos
Método	PUT
Parámetros	Grupo, IdClase, PuntosActuales, Contrasena, UsuarioAlta, Activo
Descripción	Inserta un nuevo grupo en una clase

Tabla 49 Url Put Grupos

URL	/grupos
Método	POST
Parámetros	Id, Grupo, IdClase, Contrasena, UsuarioModif, Activo
Descripción	Modificamos el grupo con el id enviado

Tabla 50 Url Post Grupos

URL	/grupo
Método	POST
Parámetros	Id
Descripción	Obtiene los datos del grupo con el id enviado

Tabla 51 Url Post Grupo

URL	/grupos/:id
Método	DELETE
Parámetros	Id
Descripción	Elimina el grupo con el id enviado

Tabla 52 Url Delete Grupos

Usuarios

URL	/alumno/:id
Método	POST
Parámetros	Id
Descripción	Devuelve los datos del alumno enviado

Tabla 53 Url Post Alumno

URL	/alumnos
Método	GET
Parámetros	IdClase
Descripción	Devuelve los alumnos y alumnas de la clase enviada

Tabla 54 Url Get Alumnos y alumnas

URL	/usuario
Método	PUT
Parámetros	Nombre, Apellidos, Nick, Contrasena, idRol, idClase, idGrupo, Administrador
Descripción	Añadir un alumno

Tabla 55 Url Put Usuario

URL	/usuario
Método	POST
Parámetros	Id, Nombre, Apellidos, Nick, Contrasena, idClase, idGrupo, Administrador
Descripción	Modificar un alumno

Tabla 56 Url Post Usuario

URL	/usuario/:id
Método	DELETE
Parámetros	Id
Descripción	Elimina el alumno enviado

Tabla 57 Url Delete Alumno

URL	/profesor
Método	PUT
Parámetros	Nombre, Apellidos, Nick, Contrasena, idRol, Administrador
Descripción	Inserta un nuevo profesor

Tabla 58 Url Put Profesor

URL	/profesor
Método	POST
Parámetros	Id, Nombre, Apellidos, Nick, Contrasena, Administrador
Descripción	Modificar un profesor

Tabla 59 Url Post Profesor

URL	/profesores
Método	GET
Parámetros	Ninguno
Descripción	Devuelve la lista de profesores

Tabla 60 Url Get Profesores

Roles

URL	/roles
Método	GET
Parámetros	Ninguno
Descripción	Devuelve la lista de roles

Tabla 61 Url Get Roles

Idiomas

URL	/idiomas
Método	GET
Parámetros	Ninguno
Descripción	Devuelve la lista de idiomas

Tabla 62 Url Get Idiomas

Ejercicios

URL	/ejerciciosClaseIdioma
Método	GET
Parámetros	idIdioma
Descripción	Devuelve la lista de los ejercicios según el idioma seleccionado

Tabla 63 Url Get EjerciciosClaseIdioma

URL	/ejerciciosClase
Método	POST
Parámetros	idEjercicio
Descripción	Devuelve el ejercicio del idEjercicio enviado

Tabla 64 Url Post EjerciciosClase

URL	/ejercicios_Clases
Método	POST
Parámetros	idEjercicio,IdClase,IdNivel, Orden, FechaVisualizacionEjer, FechaEntregaEjer, IdEjercicioIdioma
Descripción	Insertar un nuevo ejercicio

Tabla 65 Url Post Ejercicios_Clases

URL	/ejercicio_Clase/:id
Método	POST
Parámetros	idEjercicioClase
Descripción	Elimina un ejercicio

Tabla 66 Url Post Ejercicios_Clases_id

Puntuación

URL	/puntuacion
Método	POST
Parámetros	idGrupo
Descripción	Obtiene la puntuación obtenida por el grupo

Tabla 67Url Post Puntuacion

URL	/puntuacionEjercicio
Método	PUT
Parámetros	IdEjercicioIdioma,IdEjercicio,IdEjercicioClase, IdGrupo, Orden, PtosParalelismo, PtosPensamientoLogico, PtosControlFlujo, PtosInteractividad, PtosSincronizacion,PtosAbstraccion, PtosDatos,PtosTotales,FechaEjercicio, xml
Descripción	Inserta la puntuación obtenida en el ejercicio y actualiza los puntos actuales del grupo

Tabla 68 Url Post PuntuacionEjercicio

4.7. DISEÑO E IMPLEMENTACIÓN DEL INTERFAZ

El diseño de la interfaz perseguía una serie de objetivos tales como que permitiera cierta compatibilidad entre los navegadores, especialmente para Mozilla Firefox e Internet Explorer. Además se buscaba un diseño sencillo para facilitar el uso de la aplicación al usuario.

Las vistas han quedado divididas en tres partes:

- **Cabecera**: donde se muestra el nombre de la aplicación
- **Menú**: Dependiendo de si el usuario tiene un rol básico o de administrador, la aplicación cargará un menú de usuario u otro.
- **Contenido**: Muestra el contenido específico para cada funcionalidad de la aplicación.

A continuación se muestran las pantallas que utiliza la aplicación separadas por la visualización que tienen los alumnos y alumnas y profesorado. (Ver Apéndice 2. Manual de usuario)

4.7.1 Usuario no registrado

Inicio

Página inicial que es accesible por cualquier usuario sin registrarse.

Página desde la que se carga un ejercicio/ programa de Snap! con extensión XML y desde la que se analiza el proyecto.

Figura 55 Inicio

Análisis

Al pulsar sobre el botón Analizar proyecto, este se analiza y se muestra la siguiente pantalla.

En esta pantalla se utiliza el componente de Bootstrap llamado *Progress bars*. Se añaden clases diferentes para cada puntuación obtenida.

Figura 56 Análisis

Login

Desde esta pantalla se accede a la aplicación con el usuario registrado, profesores y alumnos y alumnas.

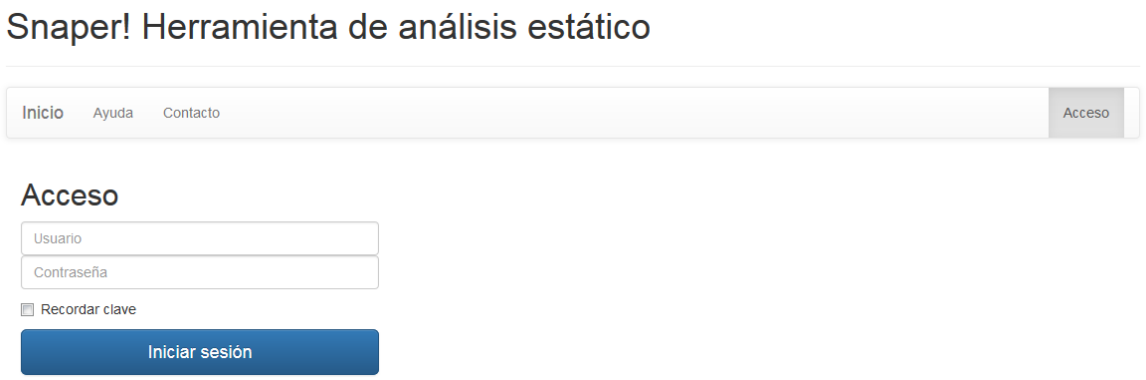

Figura 57 Login

4.7.2 Alumno y alumna

Al acceder como usuario registrado la aplicación muestra los ejercicios ya realizados con su correspondiente puntuación y el siguiente ejercicio a realizar.

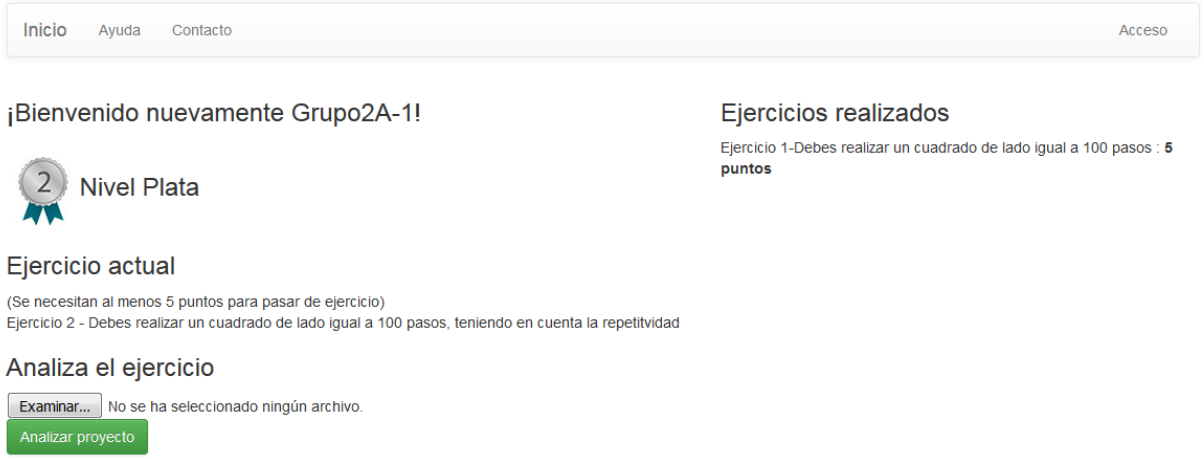

Figura 58 Alumno y alumna

4.7.3 Profesor / Administrador

Niveles

Esta pantalla muestra los niveles de la aplicación y el número de ejercicios que tiene cada nivel para ser superados.

En esta pantalla se muestran los datos mediante el control Angular UI Grid.

Este formulario no es editable. Se pueden añadir más niveles pero desde la base de datos

Figura 59 Niveles

Clases

En la parte izquierda de la pantalla se muestran las clases y su estado. En la parte derecha es la zona donde se pueden realizar las acciones sobre las clases (editar, eliminar, agregar)

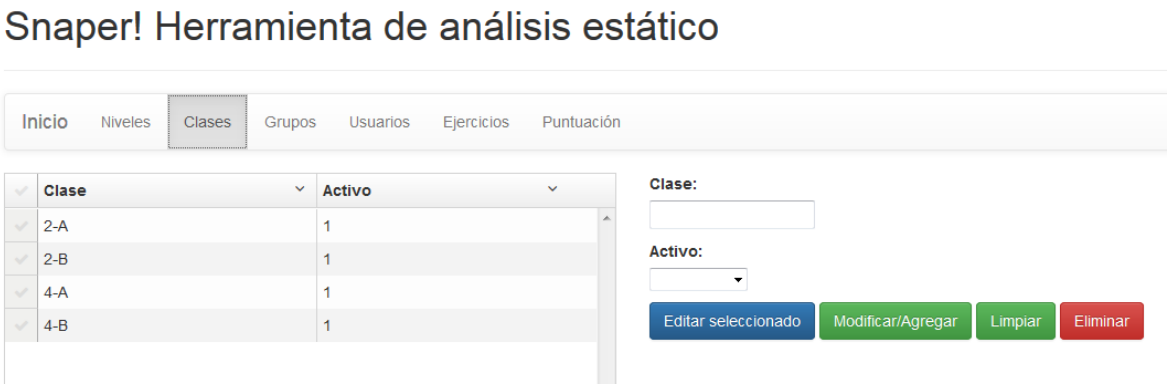

Figura 60 Clases

Grupos

En la parte izquierda de la pantalla se muestran los grupos, junto con sus contraseñas, los puntos actuales que tiene y el estado. En la parte derecha es la zona donde se pueden realizar las acciones sobre los grupos(editar, eliminar, agregar)

En el alta de un grupo la contraseña se genera automáticamente y el valor de los puntos es de cero puesto que todavía no ha realizado ningún ejercicio.

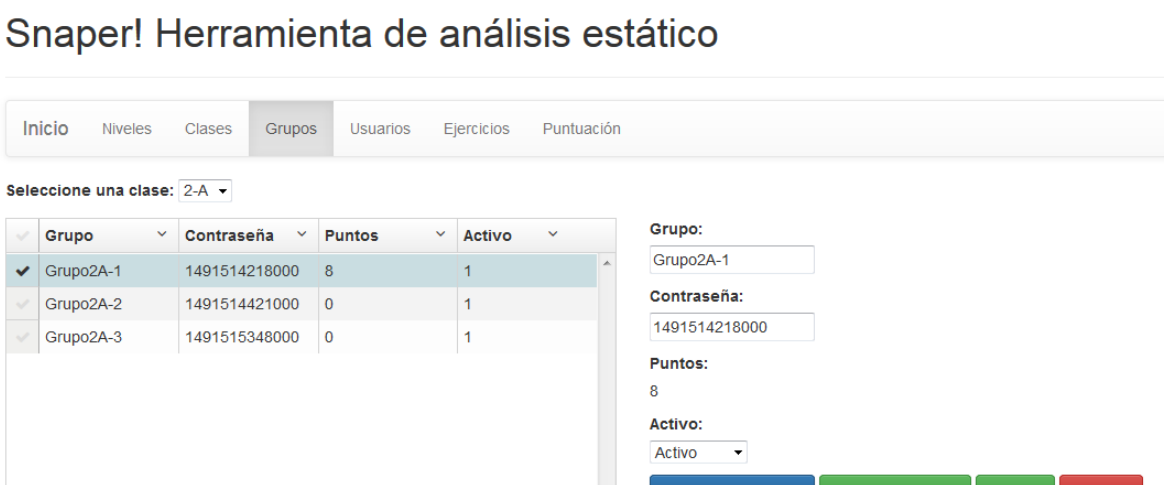

Figura 61 Grupos

Usuarios

Si somos administradores muestra la visibilidad de alumnos y alumnas y profesores

Alumnos y alumnas

La pantalla muestra los alumnos y alumnas y sus propiedades.

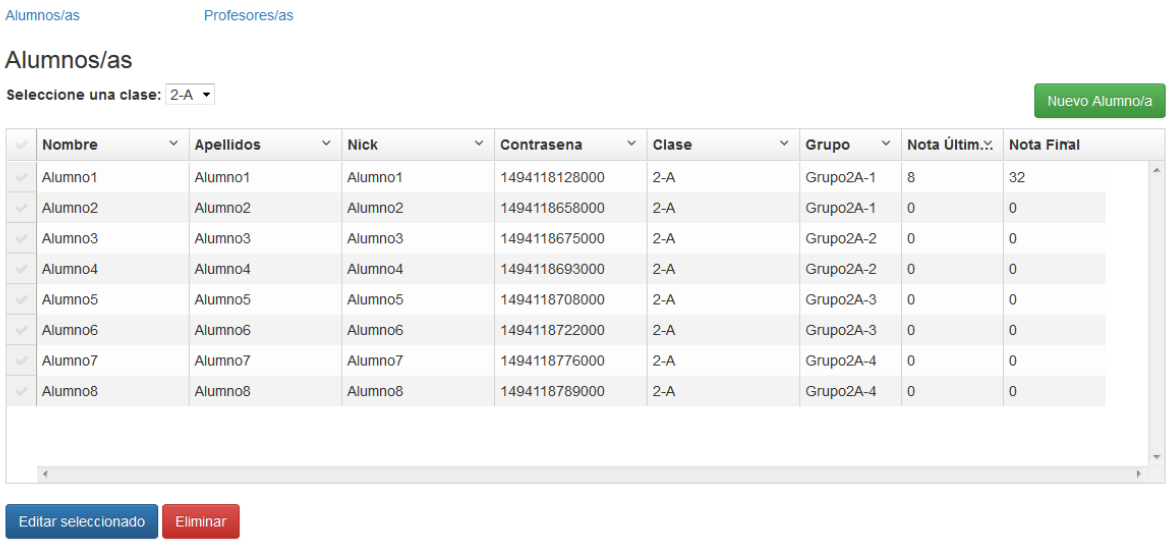

Figura 61 Usuarios

Al pulsar sobre Nuevo Alumno/a aparece la siguiente ventana modal

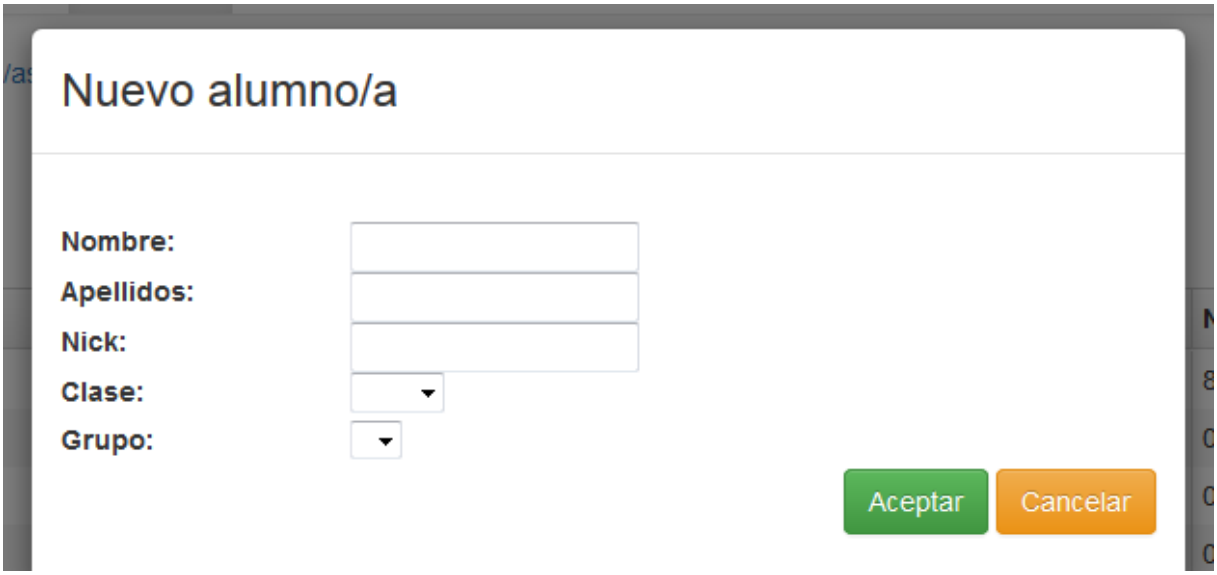

Figura 62 Nuevo alumn/a

Profesor

La pantalla muestra los profesores y sus propiedades. Solo el administrador puede dar de alta otros administradores.

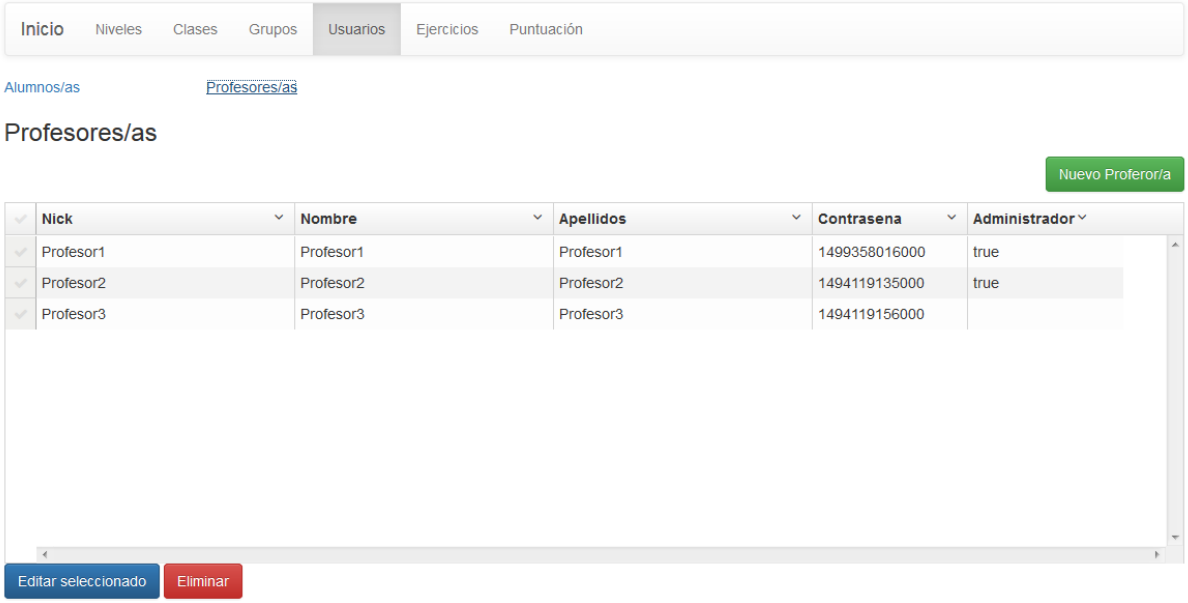

Figura 63 Profesores

Al pulsar sobre Nuevo Profesor aparece la siguiente ventana modal

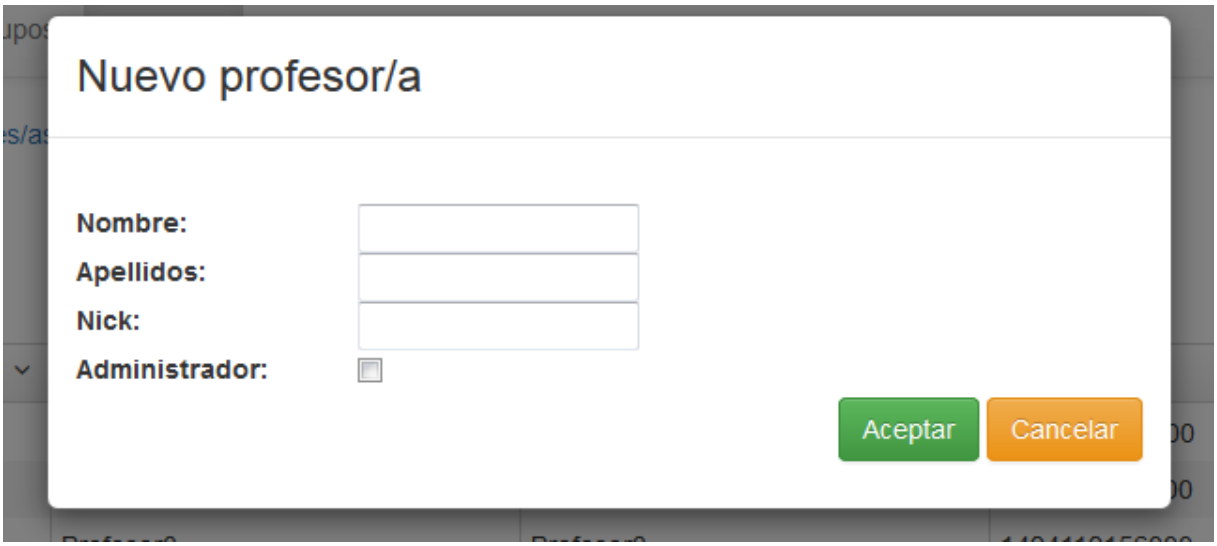

Figura 64 Nuevo profesor/a

Ejercicios

Figura 65 Ejercicios

Al seleccionar un ejercicio se cargan las clases a las que está asignado.

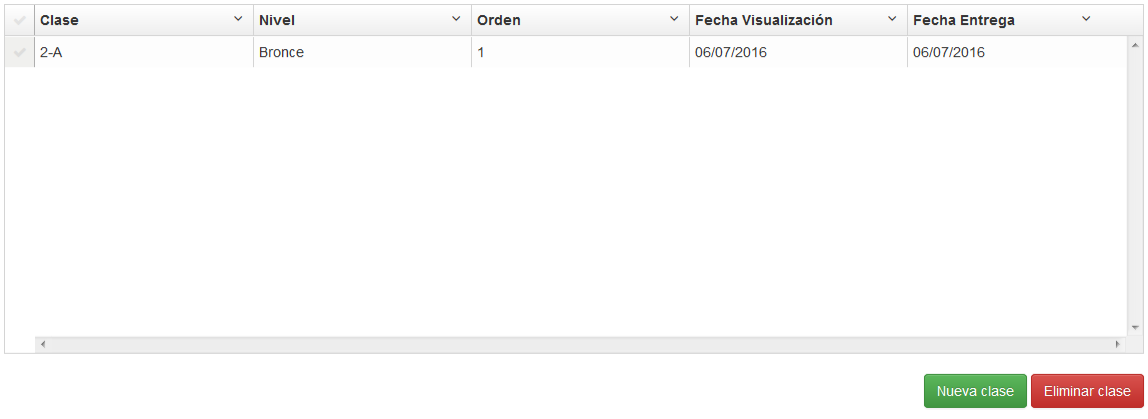

Figura 66 Clases Ejercicios

Al pulsar sobre un nuevo ejercicio aparece la siguiente ventana modal.

Figura 67 Añadir ejercicio

Al pulsar sobre añadir nueva clase, de tal forma que el ejercicio seleccionado se asocie a otra clase.

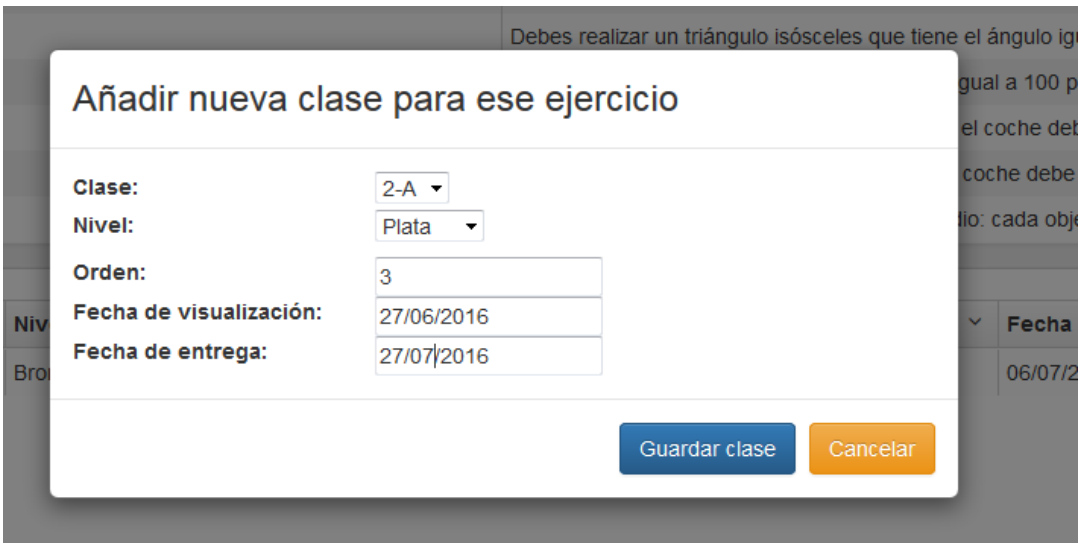

Figura 68 Añadir clase ejercicio

Puntuación

Muestra las puntuaciones obtenidas en ese momento de cada grupo.

Figura 69 Puntuación

Al seleccionar un grupo, se muestra en otro grid los datos obtenidos por ese grupo en cada ejercicio

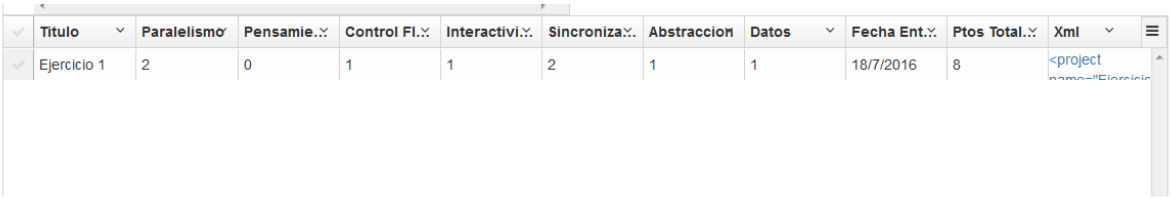

Figura 70 Puntuación Ejercicio

CAPÍTULO 5

CONCLUSIONES

5. CONCLUSIONES

El presente proyecto consiguió el objetivo general, la construcción de una aplicación web (MEAN) que permita a los usuarios evaluar los proyectos creados en Snap!

Dentro de los objetivos específicos, se puede valorar positivamente la creación de los diferentes métodos creados para evaluar los criterios del pensamiento computacional y leer el programa de Snap!

Además hay que valorar también el desarrollo usando las tecnologías que están en auge en el mercado a día de hoy. La conexión entre AngularJS, NodeJS y MongoDB, haciendo así una web multiplataforma.

CAPÍTULO 6

TRABAJOS FUTUROS

6. TRABAJOS FUTUROS

La herramienta desarrollada en el actual proyecto debe considerarse una primera aproximación.

Para conseguir que la herramienta pueda ser comercializada , se deben solventar ciertos problemas y desarrollar otros aspectos importantes.
Se describen a continuación las posibles mejoras y nuevas funcionalidades:

- **Seguridad**. Uno de los trabajos futuros debería ir dirigido a la fiabilidad de la aplicación. Una revisión y plan de pruebas unitarias para la validación de software.
  Otro tema a tener en cuenta es la seguridad a la hora de guardar las contraseñas y los datos del usuario. La herramienta debe cumplir los condiciones de la ley de protección de datos.
- **Idiomas**. Uno de los aspectos que no se ha podido realizar es una aplicación web multidioma. AngularJs posee el módulo gettext para dar soporte (https://angular-gettext.rocketeer.be/)
- **Múltiples centros**. Otra funcionalidad a desarrollar es que la herramienta pudiera registrar varios centros de estudio, de tal forma que cada centro pudiera tener acceso a la herramienta.
- **Red social**. Para permitir la interacción de los usuarios entre centros, una funcionalidad interesante sería el añadir un chat a la herramienta. La posibilidad de que los usuarios puedan competir por mejorar sus habilidades de pensamiento computacional dejando comentarios o chateando con otros usuarios de diferentes centros.

CAPÍTULO 7

BIBLIOGRAFÍA

7. BIBLIOGRAFIA

HTML

http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=435:ique-es-y-para-que-sirve-html-el-lenguaje-mas-importante-para-crear-paginas-webs-html-tags-cu00704b&catid=69:tutorial-basico-programador-web-html-desde-cero&Itemid=192
https://es.wikipedia.org/wiki/HTML

Bootstrap

http://getbootstrap.com/

Javascript

https://librosweb.es/libro/javascript/capitulo_1.html

Angular JS

https://angularjs.org/
http://www.w3schools.com/angular//default.asp
http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html
http://ui-grid.info/

NodeJS

https://nodejs.org/en/
http://www.nodebeginner.org/index-es.html
http://www.campusmvp.es/recursos/post/Que-es-el-stack-MEAN-y-como-escoger-el-mejor-para-ti.aspx

Casos de usos

https://parasitovirtual.wordpress.com/tag/diagrama-de-casos-de-uso/
http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416

REST

http://asiermarques.com/2013/conceptos-sobre-apis-rest/

Pensamiento computacional

https://programamos.es/que-es-el-pensamiento-computacional/
http://formacion.educalab.es/pluginfile.php/9364/mod_imscp/content/2/pensamiento_computacional_y_scratch.html
http://www.um.es/ead/red/46/Basogain.pdf
http://drscratch.programamos.es/

APÉNDICE

APÉNDICE 1. INSTALACIONES

NODEJS

Para usar NodeJS se necesitan dos herramientas:

- Servidor Node.js
- El instalador de paquete npm (Node package manager) que permite instalar extensiones en nuestro servidor Node.

Desde la web oficial http://nodejs.org/ se accede a la descarga de los instaladores. Para windows se descarga el msi correspondiente y se instala como cualquier otra aplicación.

Por defecto se instala en la carpeta *NodeJS* dentro de la carpeta *Archivos de programa* Para probar el funcionamiento del servidor, se accede desde la consola de comandos de windows e ir a la carpeta NodeJS. Escribir *node* y pulsar *Intro*. Una vez ahí, teclear 1+1, pulsar *Intro* y ver el resultado.

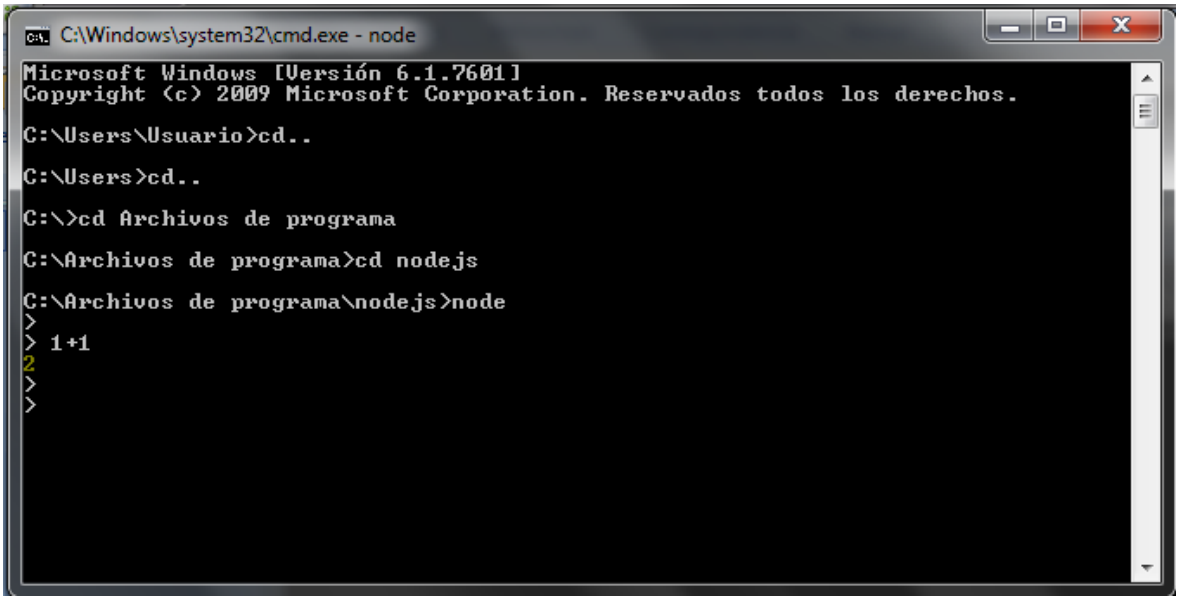

Figura 70 Instalación NodeJS

MONGODB

Descargar la versión oportuna según el sistema que tengamos. La descarga se realiza desde la web de Mongo.

La descarga consta de un archivo instalador msi que se ejecuta como cualquier otro instalador. Si no se especifica nada instala el producto en una carpeta de la carpeta *Archivos de programa.* Desde la instalación personalizada seleccionar una nueva carpeta a C:\mongodb

Crear en C un directorio *data* y dentro de este un directorio *db*. Esta es la configuración que por defecto se utiliza para almacenar las bases de datos y que hay que respetar.

Poner en marcha el servidor

Desde la consola de comandos, entrar en la carpeta *bin* de la carpeta descomprimida y ejecutar el archivo *mongod* para iniciar el servidor MongoDB. Si todo va bien, el servidor queda funcionando. Para comprobarlo poner la instrucción mongod --help. Debe aparecer todas las instrucciones.

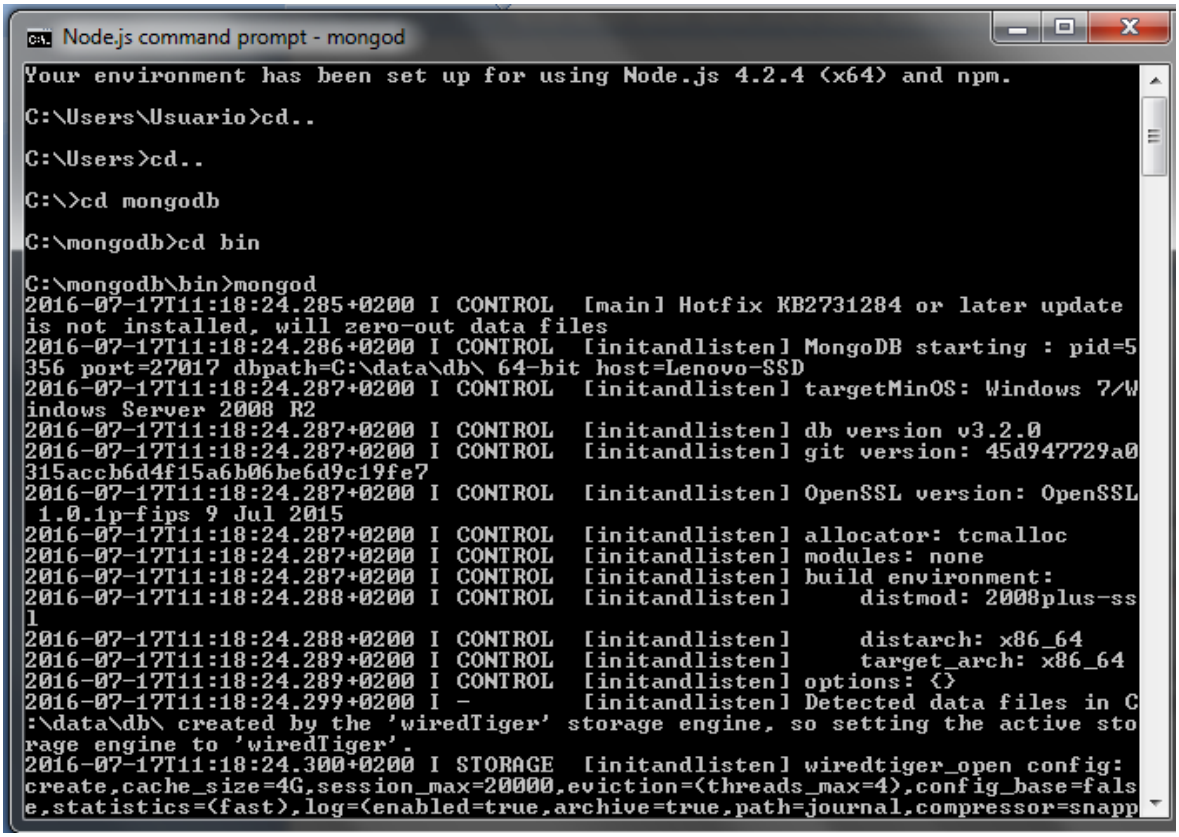

Figura 71 Instalación MongoDB

Posteriormente probar a conectarse a él. Para ello, ejecutar el archivo mongo (también desde la carpeta bin) en otra ventana de comandos y ver que aparece el Shell de Mongo. Escribir *show dbs* para ver las bases de datos.

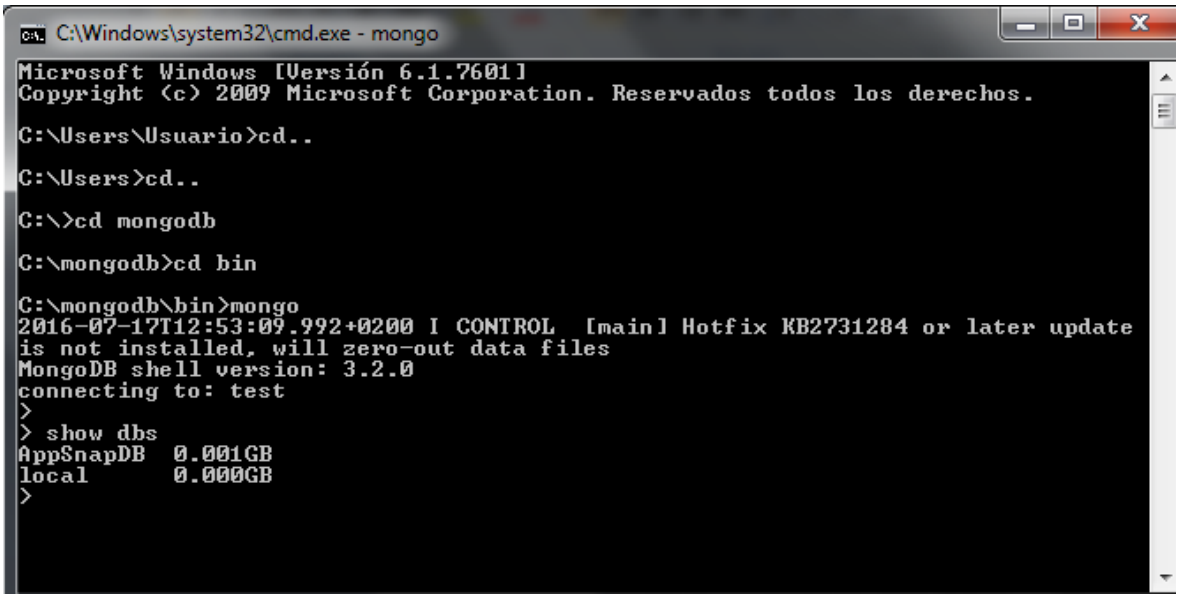

Figura 72 Collections MongoDB

Creación de la base de datos

Desde la Shell, escribir *use AppSnapDB*
Use crea la base de datos si existe y se introduce en ella.

A partir de aqui se crean las colecciones que se necesitan.
Niveles y roles

```
> db.Niveles.insert({_id:1,Nivel:"Oro",Imagen:"Oro.jpg"})
WriteResult({ "nInserted" : 1 })
> db.Niveles.insert({_id:2,Nivel:"Plata",Imagen:"Plata.jpg"})
WriteResult({ "nInserted" : 1 })
> db.Niveles.insert({_id:3,Nivel:"Bronce",Imagen:"Bronce.jpg"})
WriteResult({ "nInserted" : 1 })
```

ANGULARJS

Para trabajar con AngularJs se debe incluir el script del framewrok en la página. Esto se puede hacer de varias maneras, o bien descargando la librería por completo y colocandola en un directorio del proyecto, o bien usando un CDN para traer la librería desde un servidor remoto.

Al acceder a la web oficial https://angularjs.org al pulsar sobre el botón de descarga muestra las diferentes opciones. Se dispone además de varias opciones de descarga del framework en versión normal o comprimida.

Incorporada de forma local se asegura la entrega de la librería, aunque incorporada mediante CDN puede mejorar un poco la rapidez por el tema de la recarga y caché.

Figura 73 Instalación AngularJS

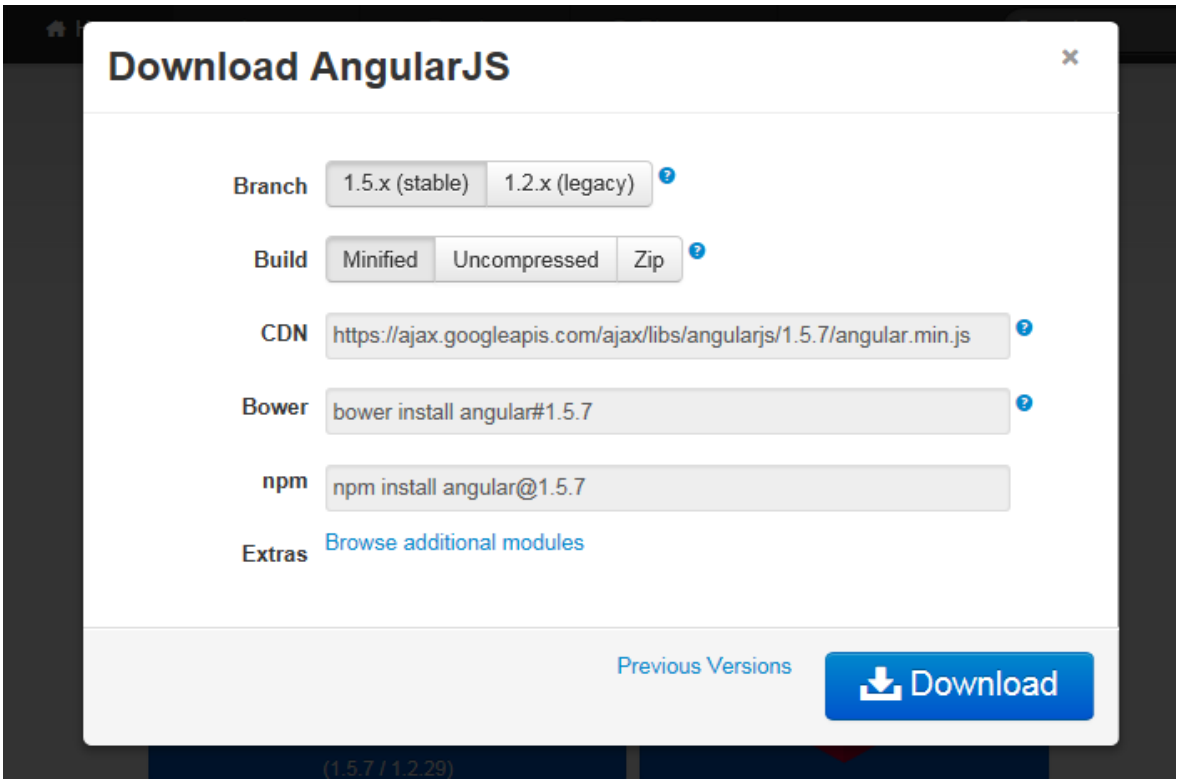

Figura 74 Download AngularJS

Incluir AngularJS en una página web

Se debe incluir el script de Angular en la página con la etiqueta SCRIPT. Ese script se pude colocar en el HEAD o bien antes del final del BODY. En principio no hay diferencias en

lo relativo a la funcionalidad, pero sí hay una pequeña mejora si se coloca antes de cerrar el cuerpo.

Si se coloca en el HEAD se está obligando a que el navegador descargue la librería de AngularJS, retrasando quizás la descarga de las áreas de la página con contenido.

Si se coloca antes de cerrar el BODY primeramente se descarga todo el código HTML y se va renderizando en la pantalla del usuario.

EJECUTAR LA APLICACION EN LOCAL

1. Levantar la base de datos

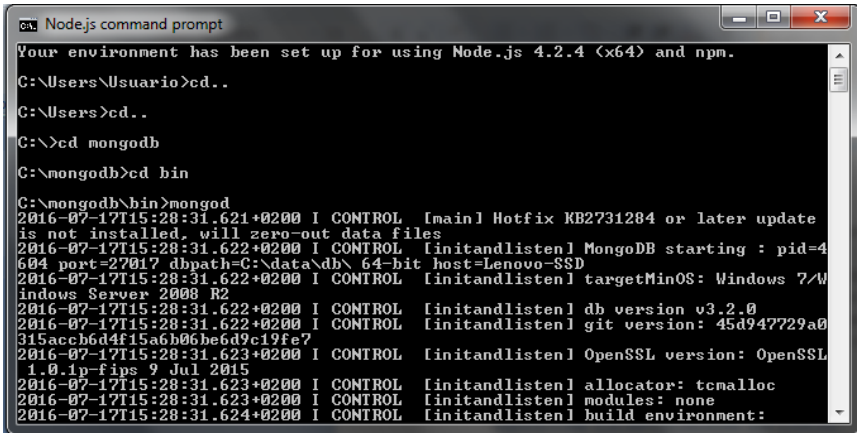

Figura 75 Levantar MongoDB

2. Acceder desde la ventana de comandos hasta la carpeta de la aplicación y ejecutar Node

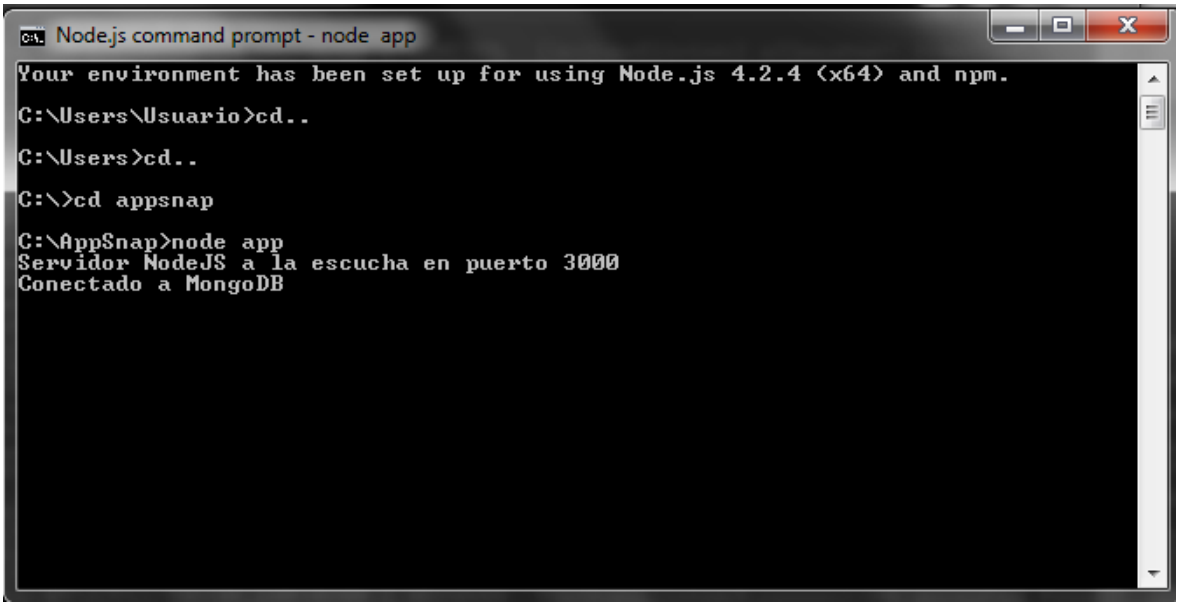

Figura 76 Levantar Node

3. Hacer doble click sobre la página de Inicio.html

APÉNDICE 2 - MANUAL DE USUARIO

Página de inicio

Desde esta pantalla se realiza la evaluación de un proyecto de Snap! según los criterios del pensamiento computacional.
Para ello se debe pulsar sobre el botón *Examinar* y seleccionar el archivo obtenido de descargar el programa de Snap! con extensión XML.

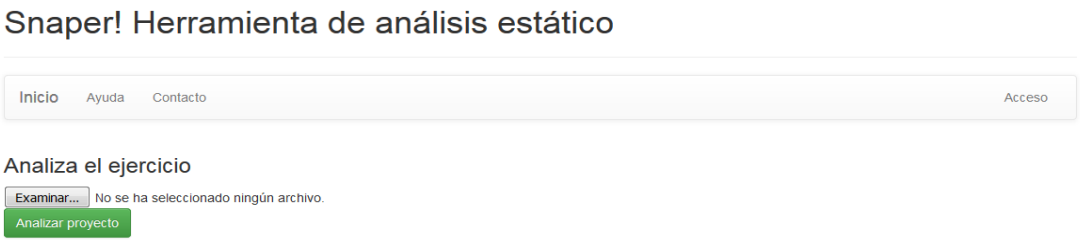

Figura 77 Manual Inicio

Una vez seleccionado el archivo, pulsar sobre Analizar proyecto para evaluarlo y obtener los resultados.

Figura 78 Manual Análisis

Los criterios de evaluación se muestran a continuación:

- **Convención**: Muestra los nombres de los personajes o objetos que no han sido modificados en el programa.

Figura 79 Manual Objetos

- **Duplicados**: Muestra el grupo de instrucciones si existen que son varias veces repetidas.

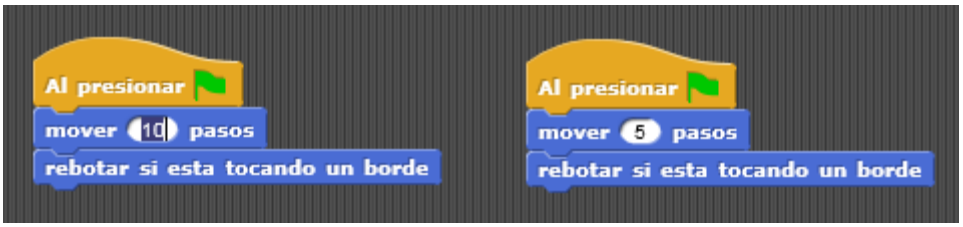

Figura 80 Manual Duplicados

- **Código muerto**: Se dice del código que nunca llega a ejecutarse. Muestra el objeto donde existe código muerto.

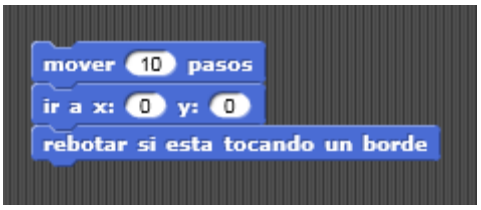

Figura 80 Manual Código muerto

- **Paralelismo**: La posibilidad de que varias cosas ocurran al mismo tiempo.
  1 punto: Cuando en el programa no existen bloques que comiencen a ejecutarse en el mismo momento. Por ejemplo, si existen al menos dos bloques de "presionar bandera verde"

Figura 81 Manual Paralelismo 1

  2 puntos: Cuando en el programa existen al menos dos bloques del tipo "presionar una tecla" , dos scripts en ‘when I am’

Figura 82 Manual Paralelismo 2

3 puntos: Cuando en el programa existen al menos dos bloques del tipo "cuando se reciba un mensaje"

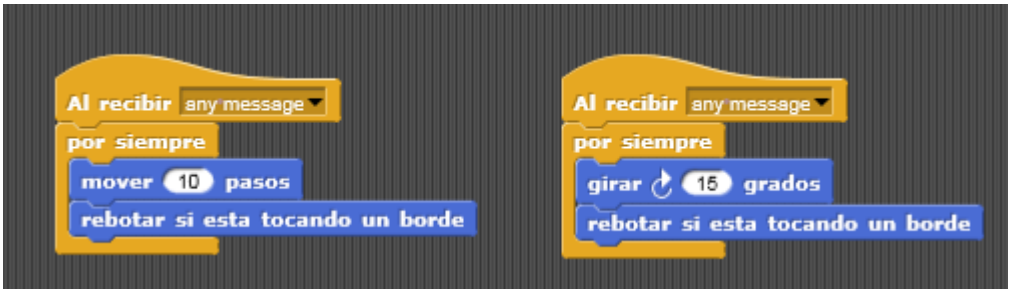

Figura 83 Manual Paralelismo 3

- **Pensamiento lógico**: Si hay instrucciones que el personaje actué de diferente manera según una condición. Ejemplo de estas instrucciones es el *"si"*
  1 punto: Si se utiliza la sentencia *"si"*

Figura 84 Manual Pensamiento lógico 1

2 puntos: Si el programa utiliza la sentencia *"si/sino"*. Este bloque evalúa la condición y según el resultado, si es verdadera ejecuta una instrucción y sino otra.

Figura 85 Manual Pensamiento lógico 2

3 puntos: En ocasiones es necesario evaluar más de una condición a la vez, en este caso, se utilizan las operaciones lógicas

Figura 86 Manual Pensamiento lógico 3

- **Control de flujo**: Si existen instrucciones que repitan ciertos bloques, por ejemplo, instrucción *"repetir hasta"*
  1 punto: Se valida cuando el programa ejecuta bloques uno a uno

Figura 87 Manual Control flujo 1

2 puntos: Si se utiliza en el programa instrucciones de repetición

Figura 88 Manual Control flujo 2

3 puntos: Cuando se utiliza la instrucción repetir hasta que

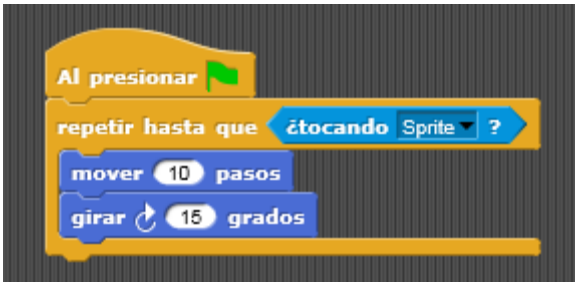

Figura 89 Manual Control flujo 3

- **Interactividad**: El usuario que lo está ejecutando el programa puede interactuar , por ejemplo, moviendo el personaje, respondiendo preguntas, etc..
  1 punto: Cuando el programa comienza con un bloque sombrero del tipo "Al presionar bandera verde"

Figura 90 Manual Interactividad 1

2 puntos: Cuando el programa utiliza bloques del tipo "Al presionar la tecla"

Figura 91 Manual Interactividad 2

3 puntos: Cuando el programa utiliza bloques de sonido

Figura 92 Manual Interactividad 3

- **Representación de datos**, se valora las instrucciones tipo cambio de disfraz, de tamaño...
  1 punto: Cuando no se utiliza en el programa ningún tipo de variables

Figura 93 Manual Datos 1

2 puntos: Cuando el programa define variables para almacenar valores

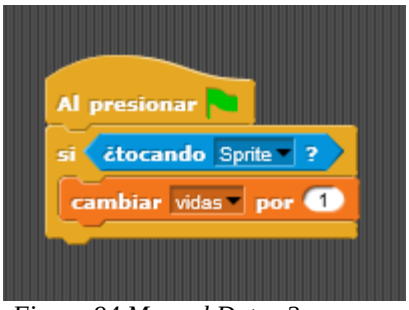

Figura 94 Manual Datos 2

3 puntos: Cuando el programa define variables de tipo lista

Figura 95 Manual Datos 3

- **Abstracción**: Consiste en dividir un problema en partes más pequeñas. Lo ideal es que el comportamiento del personaje sea controlado por diferentes programas y que cada uno de estos programas se ocupe de una cuestión concreta
  1 punto: Cuando se utiliza un único programa

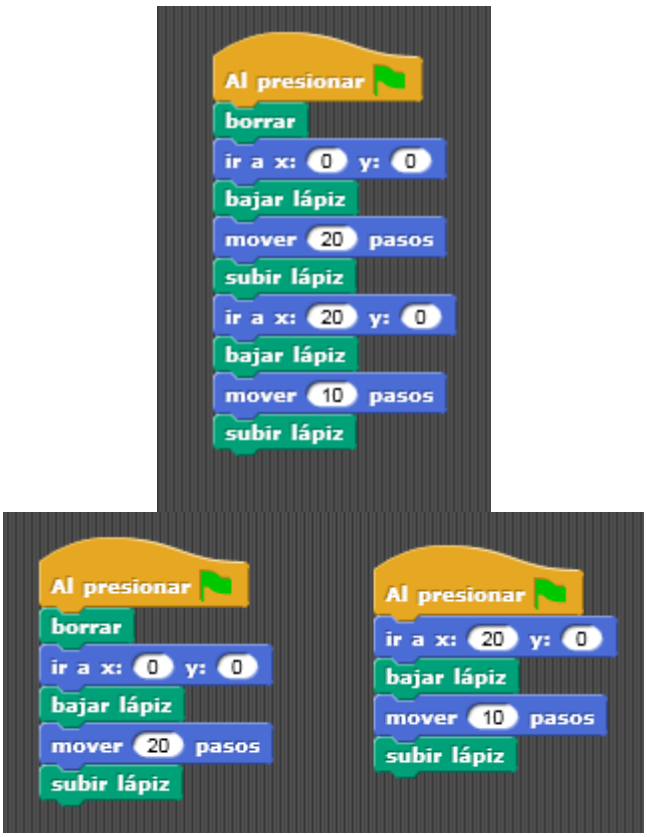

Figura 96 Manual Abstracción 1

2 puntos: Snap! permite crear bloques personalizados que se pueden llamar desde cualquier parte del programa.

Se define un bloque para posteriormente utilizarlo en el programa.

Figura 97 Manual Abstracción 2

3 puntos: Comprueba si el programa utiliza clones. Se crea un objeto y a partir de él se pueden crear otros con los mismos atributos.

Figura 98 Manual Abstracción 3

- **Sincronización**: permiten organizar a nuestros personajes para que las cosas ocurran en el orden que nosotros queremos.
  1 punto: Cuando el programa contiene el bloque "esperar x segundos". Es la forma más fácil de sincronización.

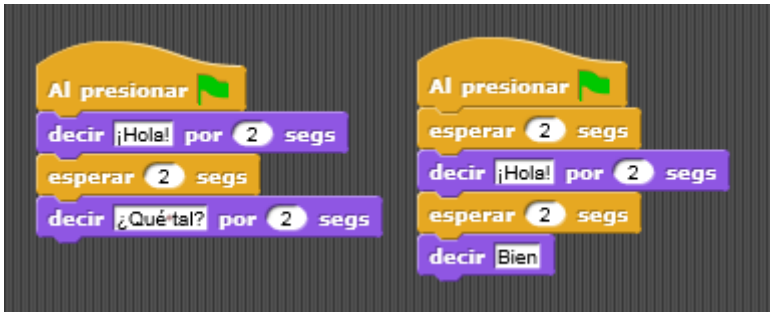

Figura 99 Manual Sincronización 1

2 puntos: Cuando se hace uso del envío de mensajes en el programa

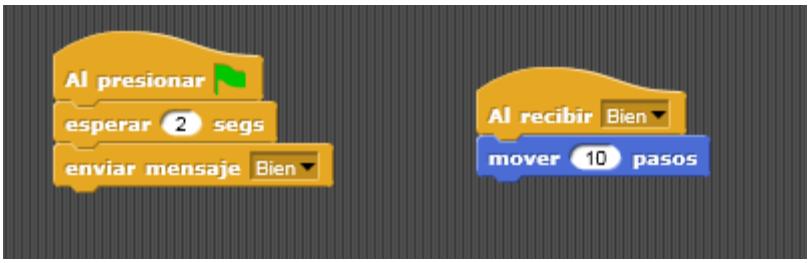

Figura 100 Manual Sincronización 2

3 puntos: Cuando se utiliza en el programa bloques del tipo "esperar hasta que" o "cuando el fondo cambie a...".

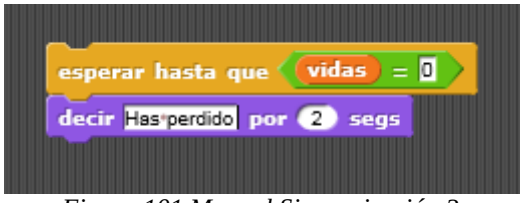

Figura 101 Manual Sincronización 3

Página de Login

Desde esta página el usuario registrado accede a la aplicación. Desde esta pantalla acceden tanto alumnos y alumnas como profesores registrados.

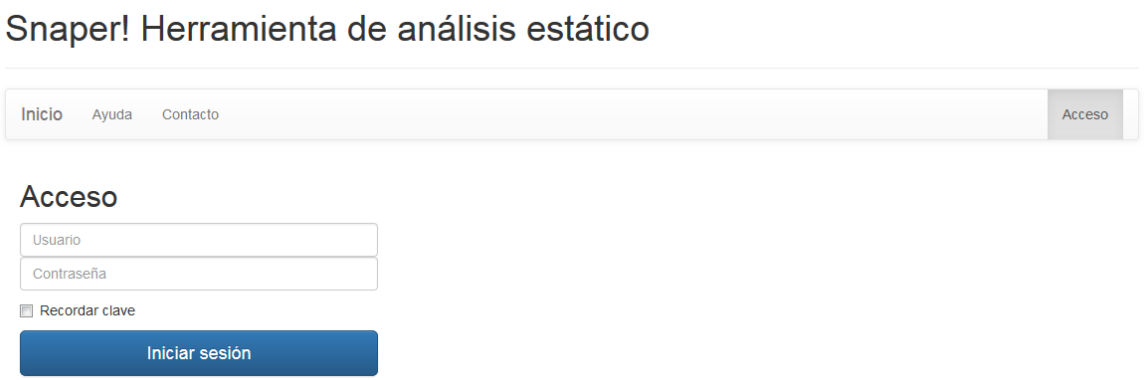

Figura 102 Manual Login

Para acceder introducir el Nick del usuario y la Contraseña que proporciona el profesor. Pulsar sobre iniciar sesión.

Página inicio de un alumno

El alumno que ha accedido a la web desde el login, accede a una página de inicio con los datos del siguiente ejercicio que debe realizar y a los anteriormente realizados con sus notas.

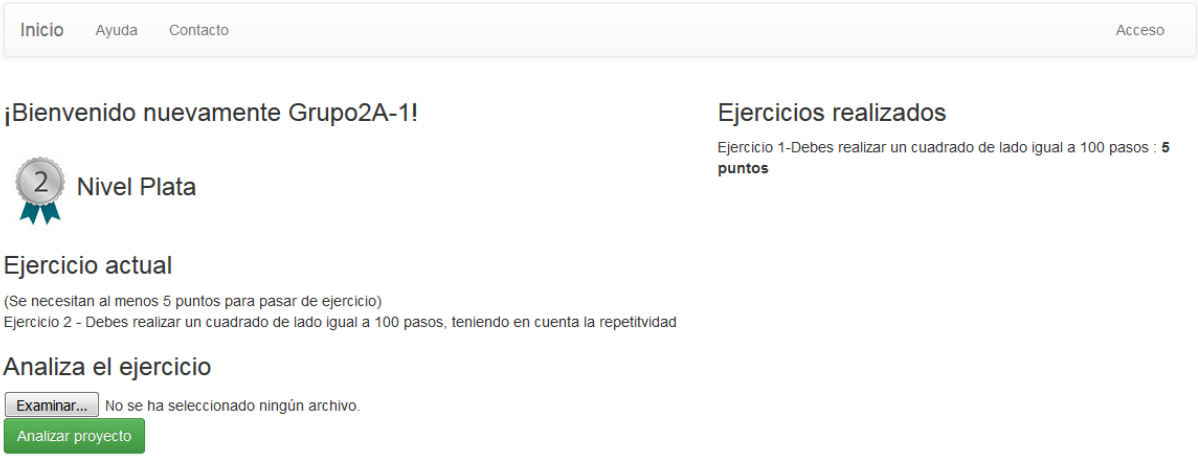

Figura 103 Manual Análisis

El usuario puede realizar nuevamente un ejercicio que ya había hecho pulsando sobre el ejercicio que quiere realizar.

Seleccionar el archivo del ejercicio que se quiere realizar y pulsar examinar.

Página niveles - Solo para profesores y administradores

Pulsar sobre el menú Niveles. Se accede a una pantalla con los datos de los niveles actualmente

Figura 104 Manual Niveles

Página clases - Solo para profesores y administradores

Pulsar sobre el menú Clases. Se accede a la siguiente pantalla.

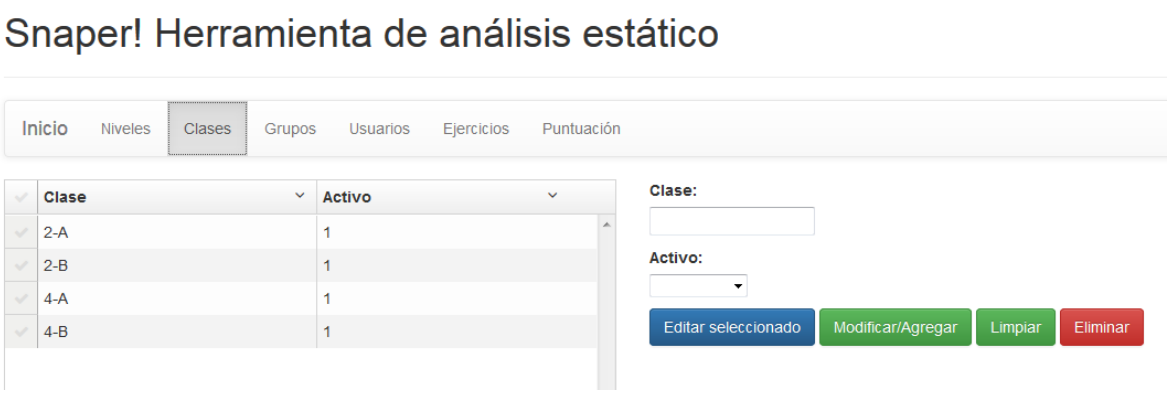

Figura 105 Manual Clases

Crear una nueva clase

Rellenar el nombre de la clase y su estado. Pulsar sobre el botón Modificar/Agregar

Editar una clase

Seleccionar desde la tabla una clase. Pulsar sobre Editar seleccionado, se rellenan los datos de la clase a editar. Modificar estos datos en el campo correspondiente. Pulsar sobre el botón Modifica/Agregar para que se guarden los datos.

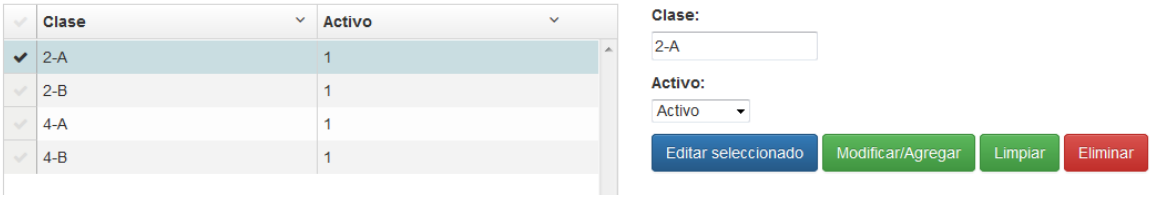

Figura 106 Manual Editar Clases

Eliminar una clase

Seleccionar desde la tabla una clase. Pulsar sobre el botón Eliminar para eliminar esa clase.

Página grupos - Solo para profesores y administradores

Pulsar sobre el menú Grupos. Se accede a la siguiente pantalla.

Figura 107Manual Grupos

Crear una nuevo grupo

Rellenar el nombre del grupo y su estado. Pulsar sobre el botón Modificar/Agregar. En el alta de un grupo la contraseña se genera automáticamente y el valor de los puntos es de cero puesto que todavía no ha realizado ningún ejercicio.

Editar un grupo

Seleccionar desde la tabla un grupo. Pulsar sobre Editar seleccionado, se rellenan los datos del a editar. Modificar estos datos en el campo correspondiente. Pulsar sobre el botón Modifica/Agregar para que se guarden los datos.

Figura 108 Manual Editar Grupos

Eliminar un grupo

Seleccionar desde la tabla un grupo. Pulsar sobre el botón Eliminar para eliminar ese grupo.

Página alumnos y alumnas - Solo para profesores y administradores

Pulsar sobre el menú Usuarios y en el submenú Alumnos y Alumnas. Se accede a la siguiente pantalla.
Seleccionar una clase para que se muestren los alumnos y alumnas asociados a la clase seleccionada.

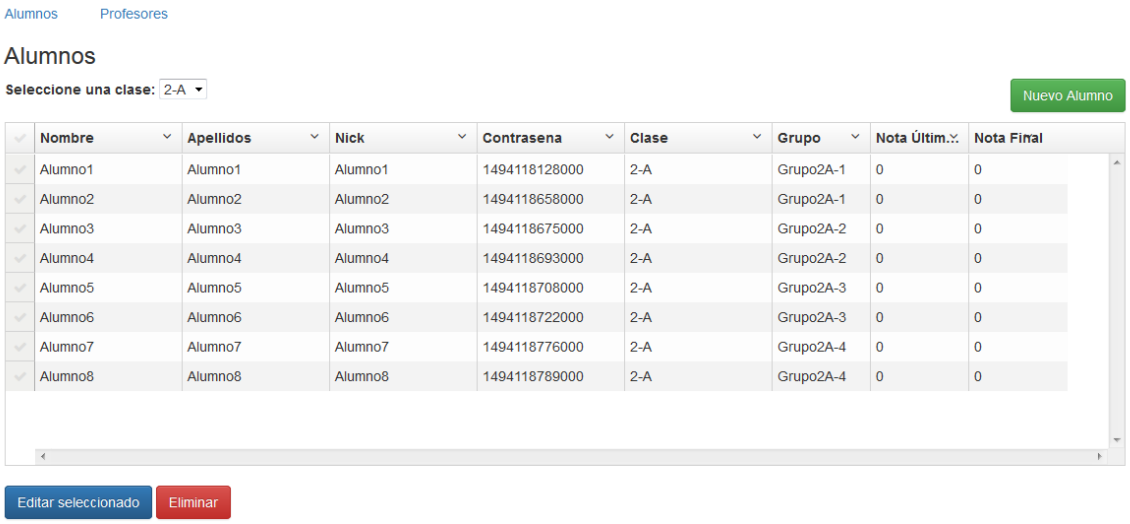

Figura 109 Manual Alumnos y alumnas

Crear una nuevo alumno

Pulsar sobre el botón Nuevo Alumno. En la ventana modal que aparece, rellenar los datos del alumno.

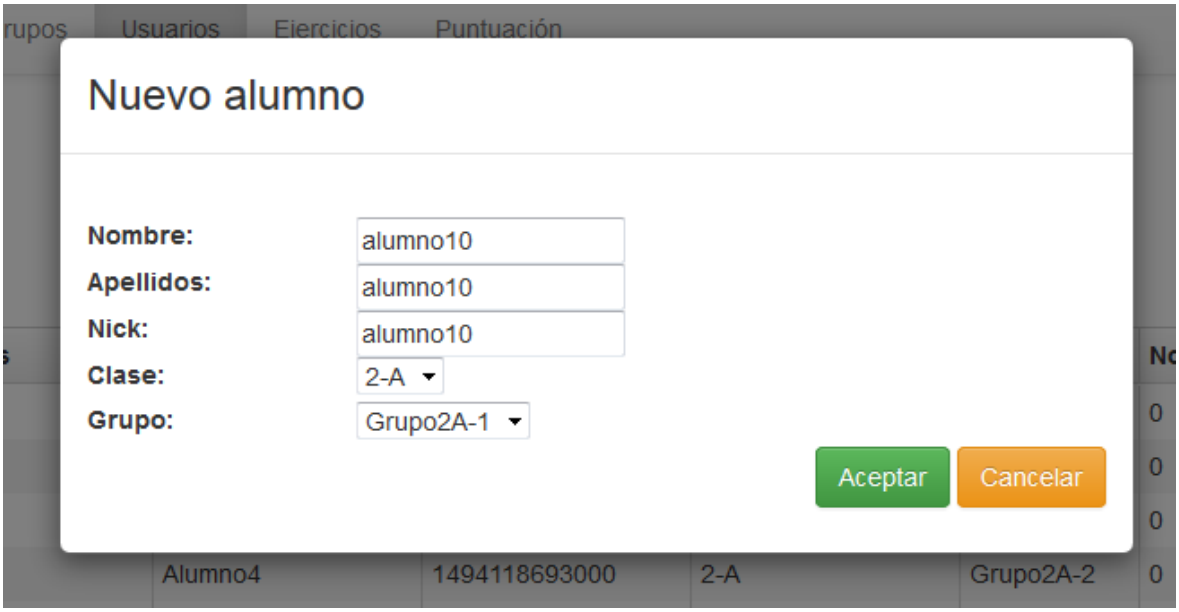

Figura 110 Manual Nuevo alumno

Todos los datos son obligatorios.
Pulsar Aceptar para dar de alta al Alumno.

En el alta de un alumno la contraseña se genera automáticamente y el valor de las notas es de cero puesto que todavía no ha realizado ningún ejercicio.

Editar un alumno

Seleccionar desde la tabla un alumno. Pulsar sobre Editar seleccionado, aparecerá la ventana modal con los datos del usuario que tiene hasta ahora. Modificar estos datos en el campo correspondiente. Pulsar sobre el botón Aceptar para que se guarden los datos.

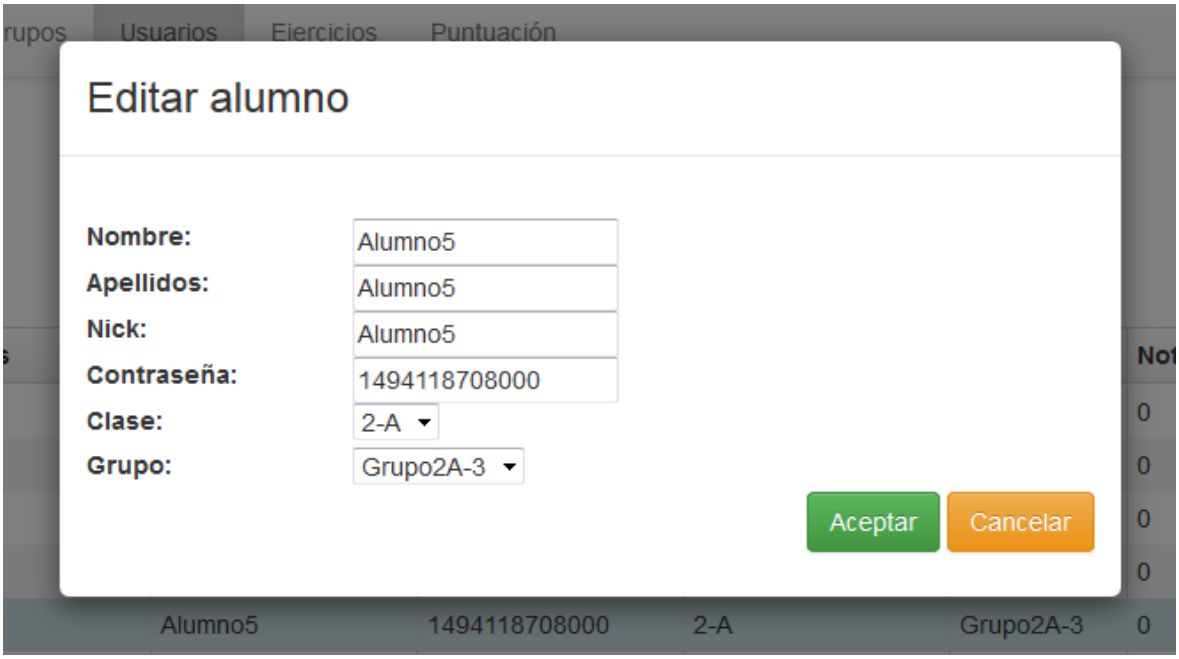

Figura 111 Manual Editar alumno

Eliminar un alumno

Seleccionar desde la tabla un alumno. Pulsar sobre el botón Eliminar para eliminar ese alumno.

Página profesores - Solo para administradores

Pulsar sobre el menú Usuarios y en el submenú Profesores. Se accede a la siguiente pantalla.

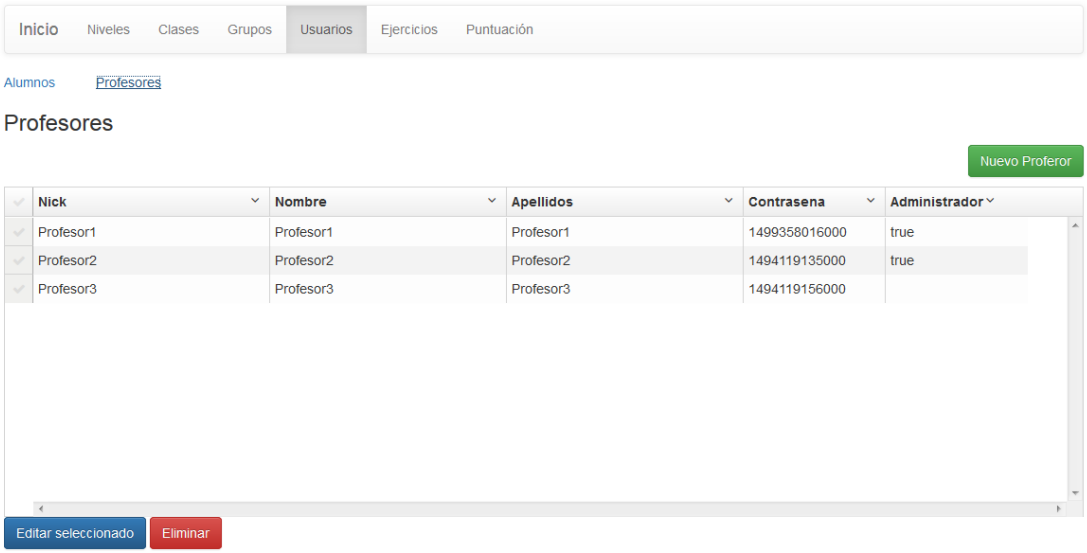

Figura 112 Manual Profesores

Crear un nuevo profesor

Pulsar sobre el botón Nuevo Profesor. En la ventana modal que aparece, rellenar los datos del profesor

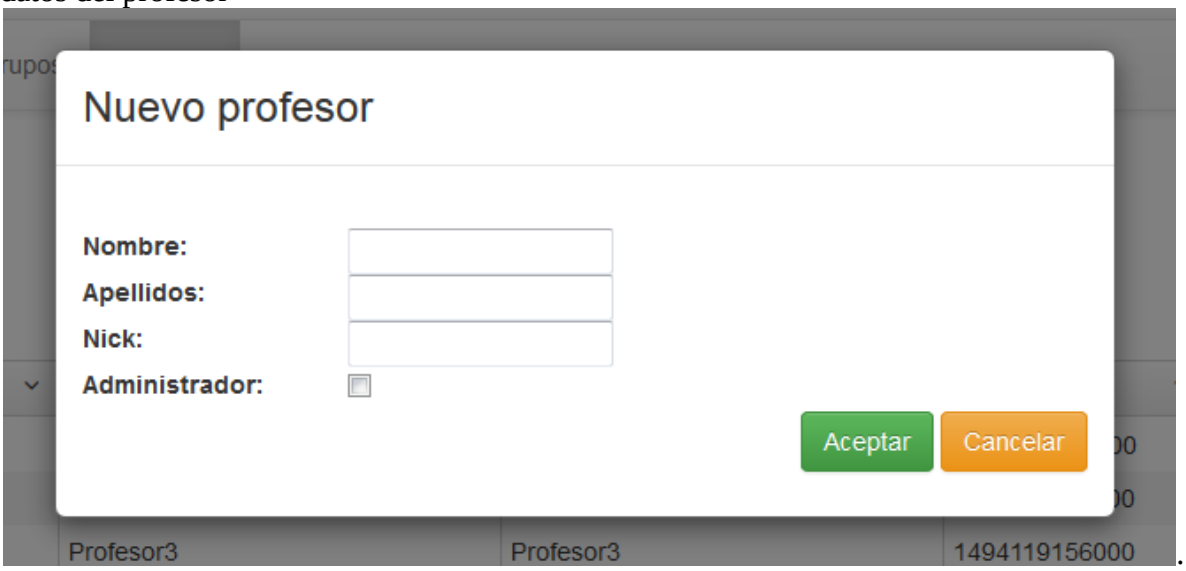

Figura 113 Manual Nuevo profesor

En el alta de un profesor la contraseña se genera automáticamente. Podemos decidir si ese profesor puede ser también administrador chequeando el check de Administrador

Editar un profesor

Seleccionar desde la tabla un profesor. Pulsar sobre Editar seleccionado, aparecerá la ventana modal con los datos del profesor que tiene hasta ahora. Modificar estos datos en el campo correspondiente. Pulsar sobre el botón Aceptar para que se guarden los datos.

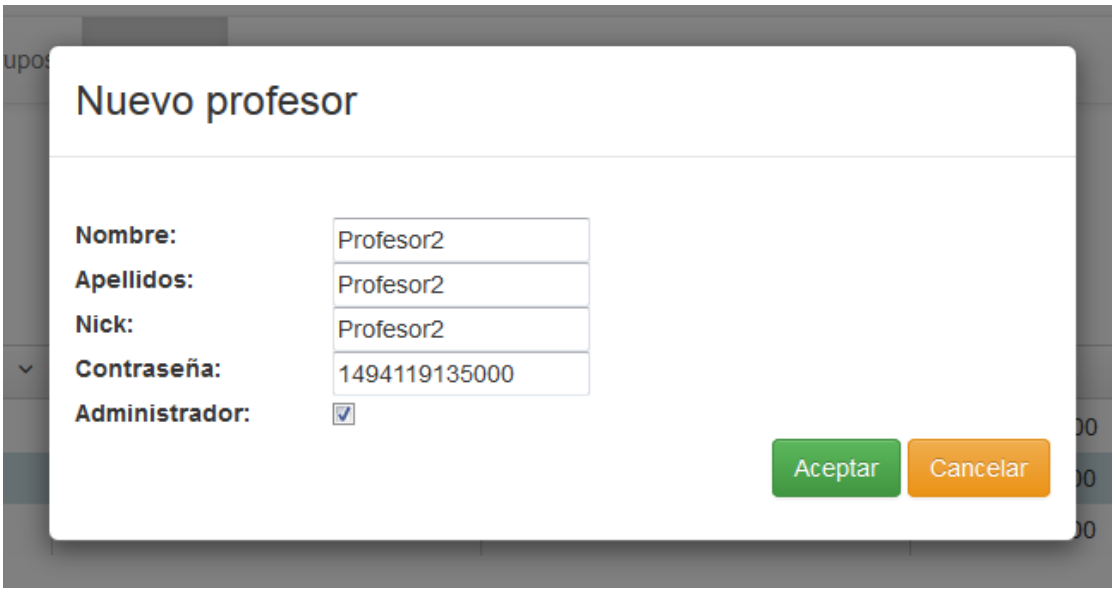

Figura 114 Manual Editar profesor

Eliminar un profesor

Seleccionar desde la tabla un profesor. Pulsar sobre el botón Eliminar para eliminar ese profesor.

Página ejercicios - Solo para profesores y administradores

Pulsar sobre el menú Ejercicios. Se accede a la siguiente pantalla. Se selecciona el idioma, en este caso el castellano.

Figura 115 Manual Ejercicios

Crear un nuevo ejercicio

Pulsar sobre el botón Nuevo ejercicio. En la ventana modal que aparece, rellenar los datos del ejercicio

Figura 116 Manual Añadir ejercicio

Pulsar Aceptar para guardar los datos.

Seleccionar un ejercicio de la tabla. Se muestran las clases asociadas a ese ejercicio en la tabla de más abajo.

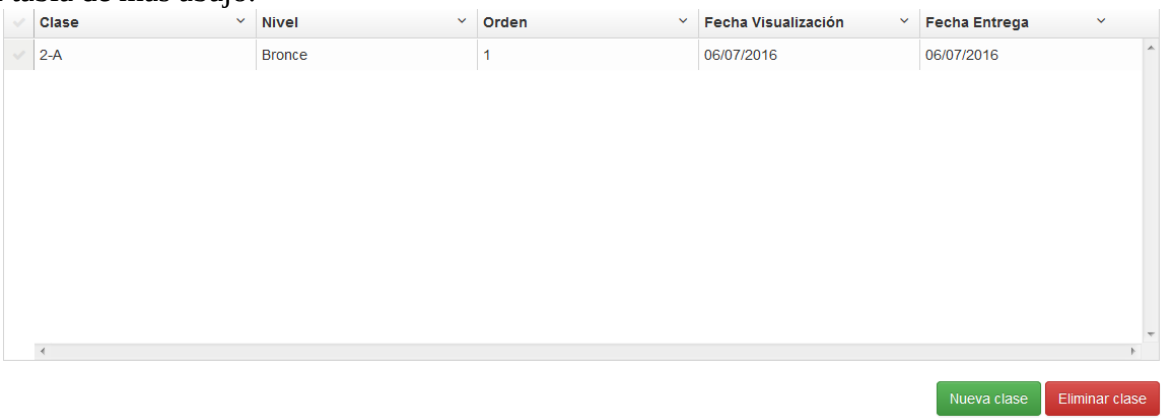

Figura 117 Manual Clases ejercicio

Pulsar sobre Nueva clase para asignarlo a la clase.

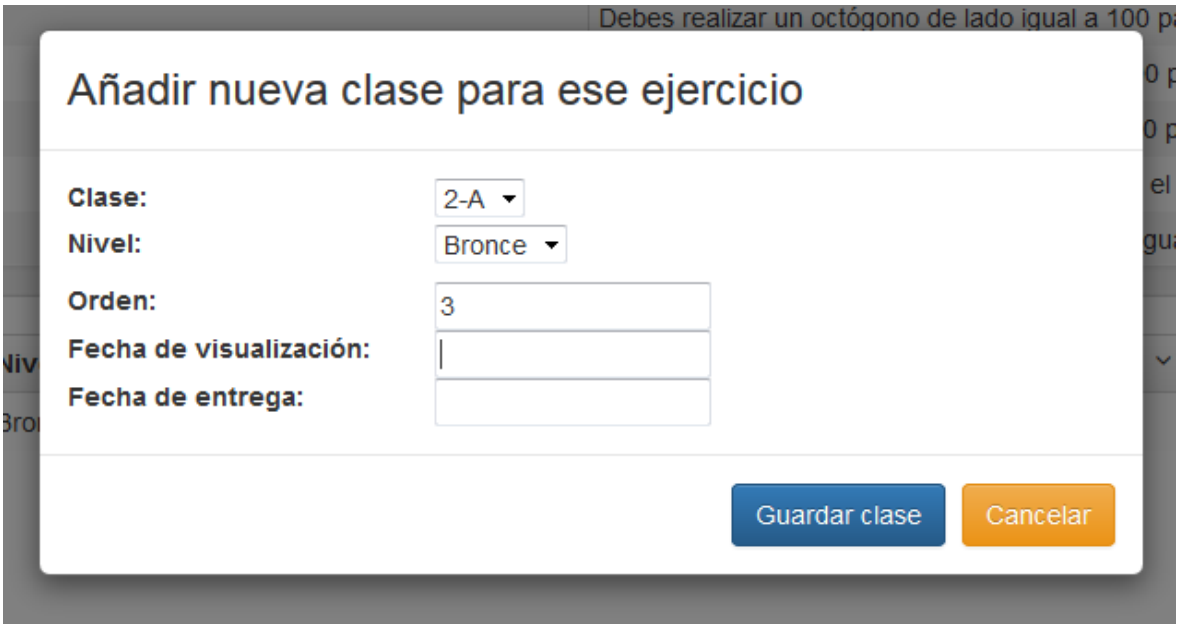

Figura 118 Manual Nueva clase ejercicio

Eliminar una clase asignada a un ejercicio

Seleccionar desde la tabla un ejercicio. En la tabla que se muestran las clases a las que pertenece ese ejercicio, seleccionar la que se quiere eliminar. Pulsar sobre Eliminar para borrar el vínculo.

Editar un ejercicio

Seleccionar desde la tabla un ejercicio. Pulsar sobre Editar seleccionado, aparecerá la ventana modal con los datos del ejercicio que tiene hasta ahora. Modificar estos datos en el campo correspondiente. Pulsar sobre el botón Aceptar para que se guarden los datos.

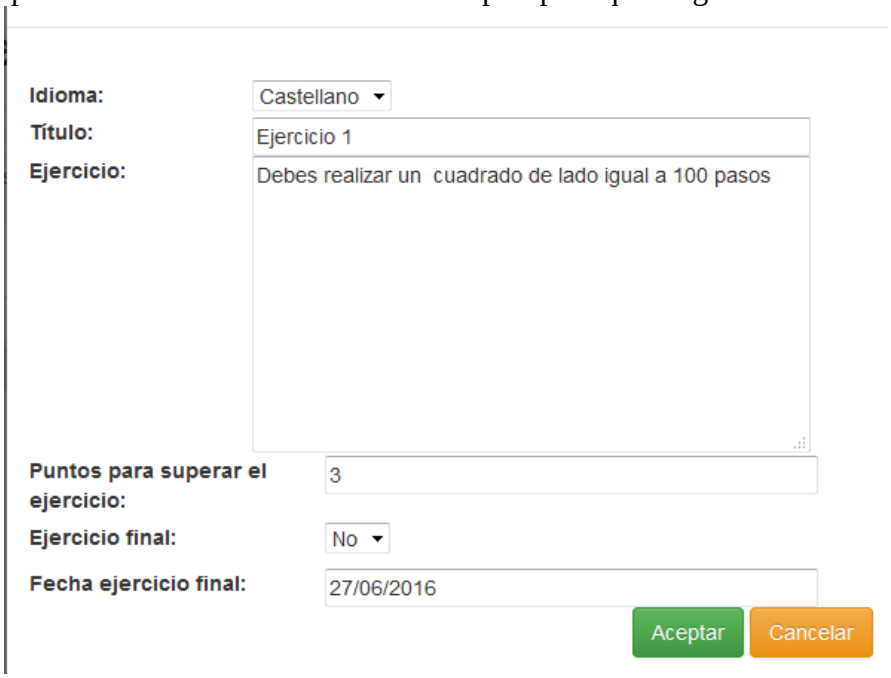

Figura 119 Manual Editar ejercicio

Eliminar un ejercicio

Seleccionar desde la tabla un ejercicio. Pulsar sobre el botón Eliminar para eliminar ese ejercicio.

Página puntuaciones - Solo para profesores y administradores

En esta pantalla se muestran las puntaciones de los grupos que se han obtenido hasta ahora.
Seleccionamos una clase para mostrar los datos de los grupos.

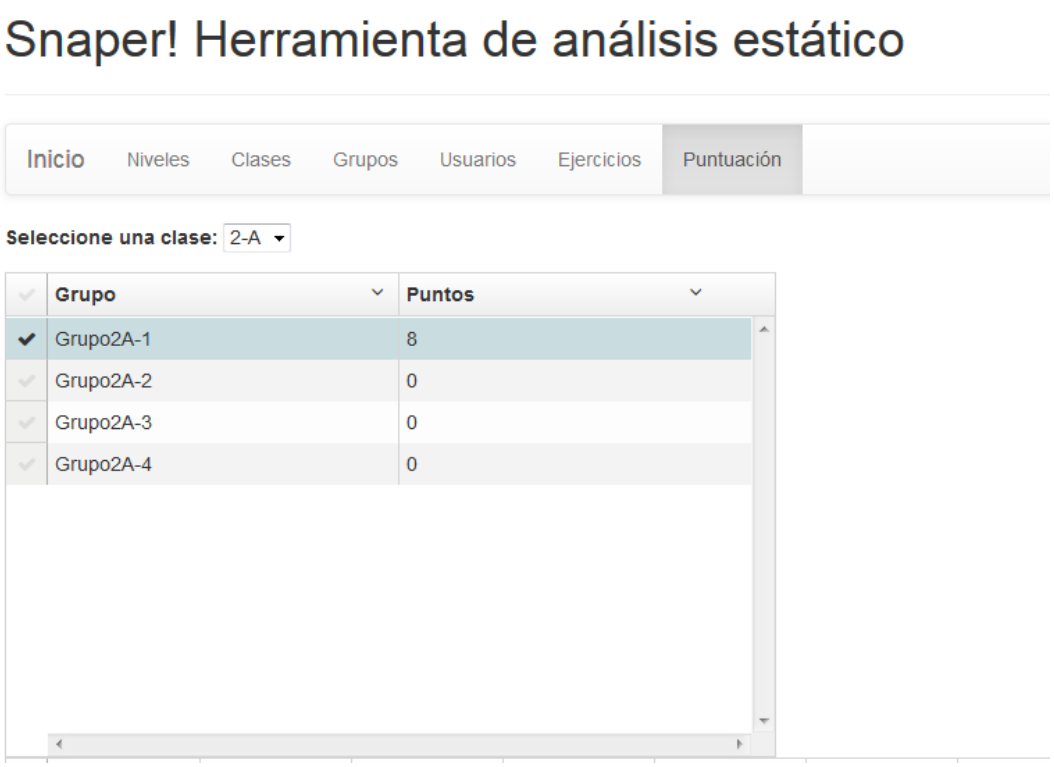

Figura 120Manual Puntuación

Para ver con más detalle los puntos obtenidos en cada ejercicio seleccionar un grupo.

Figura 121 Manual Puntuación ejercicios