



Cazasteroides

Memoria Final

07/07/2017

realizado por Miguel Núñez Díaz-Montes

tutelado por Francisco Rosales García

febrero-junio del 2017

A Google y Stack Overflow sin ellos esto no hubiese sido posible,
a mi novia por obligarme a esforzarme para acabar este trabajo,
a todos mis amigos que me han apoyado y ayudado, en especial a,

- A Michael que hizo algo por mí por una vez
- Al experto Revi que sus buenos consejos
- A Víctor y sus conocimientos matemáticos

a mi familia (que siempre se les suele nombrar),
y al magnífico equipo de cazasteroides,
en especial a Esteban

Trabaja en lo que te guste y no tendrás que trabajar ni un día de tu vida

Confucio

Resumen

Este trabajo de fin de grado se va a centrar en cómo aplicar técnicas de gamificación a una aplicación ya desarrollada con el objetivo de aumentar la participación de los usuarios. Para ello se ha realizado un estudio de las distintas técnicas actuales, así como su evolución y tendencias. Con el fin de complementar este estudio se han analizado distintas aplicaciones ya gamificadas.

Por otro lado, se ha realizado un rediseño de la aplicación en forma de prototipo que sirva de guía para futuras versiones, buscando un acercamiento a las guías de diseño actuales logrando una interfaz limpia, intuitiva y completa.

Por último, se ha implementado un módulo de karma en forma de microservicio en una API REST, utilizando el framework Flask de Python. Este módulo permite realizar las labores de cálculo de nivel, validación y selección de observaciones basados en el nivel de karma del usuario.

Palabras Clave

Gamificación, Sistemas de Reputación, Asteroides, Python, Flask, Microservicios

Abstract

This end-of-degree paper focuses on how to apply gamification techniques to an application already developed with the goal of increasing user participation. For this end, a study of digital techniques, as well as their evolution and trends, will be carried out. In order to complement this study, gamified applications will be analysed.

On the other hand, a redesign of the application has been carried out with an interactive prototype that serves as a guide for future versions, seeking an approach to the current design guides to achieve a clean, intuitive and complete interface.

Finally, a karma module has been implemented as a REST API using the Python Flask Framework. This module allows to perform the tasks of level calculation, validation and selection of observations based on the karma level of the user.

Keywords

Gamification, Reputation Systems, Asteroids, Python, Flask, Microservices

Índice de Contenido

Introducción	8
Metodología de Trabajo	9
Estructura del Proyecto	10
Estado del Arte	11
Crowdsourcing	11
Gamificación.....	14
Componentes de la Gamificación	14
Games With a Purpose	16
Ejemplos de Games With a Purpose	17
Reputation Systems	23
Componentes de los Sistemas de Reputación.....	23
Ejemplos de Sistemas de Reputación	24
Análisis y Propuesta de Mejora	26
Sistema Actual	26
Aplicación	27
Navegación	27
Apariencia	30
Sistema de Gamificación	31
Plataforma	31
Mecánicas.....	31
Dinámicas	32
Estéticas.....	33
Sistemas de Reputación.....	33
Karma	33
Cambios Realizados.....	34
Servidor de Karma	34
Herramientas y lenguajes	36
Arquitectura	37
Algoritmos.....	39
Testing e Integración Continua	50
Script de Preparación de Entorno	52
Conclusiones	54

Anexos.....	55
1. Instalación y Primer Arranque del Servidor de Karma.....	55
Prerrequisitos	55
Instalación.....	55
2. Uso del Servidor	57
Formato de Respuesta	57
Cálculo del Karma	57
Selección de la Observación a Mostrar	59
Validación de la Observación	60
3. Principios S.O.L.I.D.....	62
4. Arquitectura del Servidor de Karma	63
Bibliografía.....	64

Índice de Ilustraciones

Ilustración 1: Ejemplo de reCaptcha.....	12
Ilustración 2: Apitopia: Pantalla del juego.....	17
Ilustración 3: Apitopia: Posición y puntuación total.....	17
Ilustración 4: Apitopia, Ranking	18
Ilustración 5: Apitopia, Puertas.....	18
Ilustración 6: ARTigo, Pantalla del Juego	19
Ilustración 7: ARTigo, Pantalla de Rankings.....	19
Ilustración 8: EteRNA, Pantalla Principal	20
Ilustración 9: EterRNA, Pantalla nivel completado.....	21
Ilustración 10: Eyewire, Pantalla Principal	22
Ilustración 11: Eyewire, Pantalla nivel completado.....	22
Ilustración 12: Arquitectura cazasteroides Original.....	26
Ilustración 13: Pantallas Aplicación Actual	27
Ilustración 14: Pantallas Aplicación Propuesta.....	28
Ilustración 15: Pantallas Aplicación Rediseñada	30
Ilustración 16: Esquema Servidores	34
Ilustración 17: Caso de uso módulo de karma.....	35
Ilustración 18: Graficas Puntuación.....	41
Ilustración 19: Proceso de selección de una imagen.....	43
Ilustración 20: Nivel de Filtrado Según el Nivel de Karma	46
Ilustración 21: División Observaciones por Nivel de Filtrado.....	46
Ilustración 22: Maquina de Estados, Algoritmo de Validación	48
Ilustración 23: Resultado Test Script con Errores.....	51
Ilustración 24: Resultado Test Script Correcto.....	51
Ilustración 25: Clase TestAbstract	52
Ilustración 26: Resultado Script Preparación Entorno	53
Ilustración 27: Salida Esperada al lanzar el servidor	56
Ilustración 28: Arquitectura Servidor de Karma	63

Índice de Ecuaciones

Ecuación 1: Cálculo de los puntos por Observación	40
Ecuación 2: Cálculo de los puntos por nivel.....	40
Ecuación 3: Pseudocódigo de un EFES	43
Ecuación 4: Cálculo de la Dificultad de una imagen.....	44
Ecuación 5: Cálculo de la parte estática de la Dificultad	44
Ecuación 6: Cálculo del Valor de la Probabilidad	44
Ecuación 7: Cálculo del Valor del Seeing	45
Ecuación 8: Cálculo del Valor del Brillo	45
Ecuación 9: Cálculo de la parte dinámica	45
Ecuación 10: Cálculo de la Certeza de una Observación	49
Ecuación 11: Cálculo del Límite de Certeza	49

Introducción

Cazasteroides se trata de un proyecto de ciencia ciudadana desarrollado en conjunto entre la Universidad Politécnica de Madrid y el Instituto Astrofísico de Canarias.

Con este proyecto se pretende realizar un experimento que acerque la astronomía a la ciudadanía. Esto se va a conseguir convirtiendo a los usuarios de la aplicación en cazadores de asteroides.

Para ello los usuarios deberán localizar asteroides cercanos a la Tierra utilizando imágenes animadas obtenidas por la red de telescopios GLORIA. Estas imágenes animadas son realmente un quinteto de imágenes que van cambiando, iterando en bucle, del mismo modo que lo haría un GIF.

Todas las observaciones probables, es decir, las confirmadas por un astrónomo, serán enviadas al *Minor Planet Center* de la Unión Internacional de Astronomía (IAU, *International Astronomy Union*).

Con el fin de reducir la carga del astrónomo ofreciéndole solo las observaciones que puedan ser probables, los usuarios van a poder votar las observaciones realizadas por otros usuarios.

Para lograr una mayor aceptación de los usuarios se ha decidido utilizar técnicas de gamificación para acercarse al concepto de *Games With A Purpose*.

El objetivo de este proyecto de fin de grado es conseguir avanzar en el desarrollo de este sistema de gamificación, además de implementar un Sistema de Karma con el que ponderar las acciones de los usuarios según su nivel de experiencia, es decir, implementar un Sistema de Reputación.

Metodología de Trabajo

Este proyecto se ha decidido dividir en dos partes.

- **Investigación:** donde se han estudiado las tendencias actuales y las bases necesarias para implantar un sistema gamificado.
- **Desarrollo:** donde se implementó un servidor de Karma con los algoritmos necesarios para el cálculo del nivel, selección de imágenes y validación de estas.

Para la primera parte no se ha seguido una metodología concreta. Mientras que para la segunda se ha decidido llevar a cabo una metodología ágil. Destacando las siguientes dinámicas:

- **Reuniones semanales:** en estas reuniones se hacían controles con los supervisores donde se mostraba el desarrollo de la semana, y apuntando los cambios que fueran necesarios.
- **Desarrollo de partes separadas:** se ha decidido ir implementando poco a poco el sistema sin intentar abarcar demasiadas características de golpe, cuando se terminaba una se pasaba a la siguiente.
- **Sistemas de Control de Versiones:** se han utilizado GitHub como plataforma para el control de versiones.
- **Sistemas de Integración Continua:** se ha decidido utilizar una plataforma de Integración continua para permitir realizar tareas de testeo automático con cada cambio.

Estructura del Proyecto

Este proyecto se va a dividir en 5 secciones. Estas secciones se van a explicar a continuación

1. **Introducción:** en esta primera parte, se va a explicar cuál es el contexto de la aplicación además de la metodología de trabajo y cuáles van a ser las partes de las que se va a componer este proyecto.
2. **Estado del Arte:** en esta segunda sección se va a estudiar cuales son los orígenes, como ha sido el progreso y cuál es el estado actual de los sistemas gamificados. Además de todos los aspectos clave, así como las técnicas mejor valoradas y los mejores ejemplos de sistemas gamificados.
3. **Análisis y Propuesta de Mejora:** en esta tercera sección se va a estudiar el sistema actual y se van a proponer las mejoras necesarias las cuales se han determinado gracias a la anterior etapa.
4. **Cambios a Realizar:** en esta cuarta sección se van a explicar y documentar los cambios implementados en el sistema. En el caso de este proyecto, concretamente el servidor de Karma y el script para la preparación del entorno.
5. **Conclusión:** en esta última sección se van a realizar las conclusiones finales sobre el proyecto.

Por último, se incluirá una sección adicional de anexos con las partes de la memoria que ha sido necesario separar para su mejor comprensión.

Estado del Arte

Estado del Arte proviene del termino anglosajón *State of Art*¹ (a partir de ahora SoA). SoA se podría traducir de forma no literal al español como Tecnología Punta. En este SoA se van a estudiar las distintas metodologías, así como distintos ejemplos, sobre las que se van a sustentar las posteriores mejoras a implementar.

En concreto se van a estudiar 4 metodologías distintas, estas son, Crowdsourcing, Gamificación, *Games with a Purpose* y Sistemas de Reputación.

CrowdSourcing y Gamificación sientan las bases sobres las que se asienta, Games with a Purpose, como más adelante se va a ver en profundidad. Con esto lo que conseguiremos será ofrecer una mejor experiencia de uso, logrando así aumentar el rendimiento del sistema.

Mientras tanto, con un Sistema de Reputación, se va a conseguir implantar un sistema de karma, que permitirá, de forma resumida, dar mayor valor a las interacciones de los usuarios más expertos.

Crowdsourcing

El Crowdsourcing es una técnica que se basa en la colaboración abierta y distribuida. Actualmente, muchas organizaciones usan el Crowdsourcing para externalizar distintas tareas.

Esta técnica puede resultar muy útil cuando se aplica a campos donde los ordenadores tengan problemas para trabajar y las personas seamos capaces de realizarla sin ningún problema.

También es indispensable para la educación de las Inteligencias Artificiales basadas en *Machine Learning*. En este campo aún es necesario el trabajo de usuarios cuya labor es catalogar distintos *Datasets* para así poder utilizar este conocimiento para clasificar muestras posteriores.

¹ <http://normasapa.net/que-es-el-estado-del-arte/>

Un ejemplo a destacar de esto último se trata de *reCaptcha* [1], plataforma desarrollada por Luis Von Ahn uno de los pioneros del CrowdSourcing que analizaremos más adelante. Los captcha son sistemas que utilizan las páginas web para evitar el tráfico de bots siguiendo el lema "Fácil para humanos, complicado para ordenadores". Un ejemplo de *reCaptcha* se puede apreciar en la Ilustración 1.



Ilustración 1: Ejemplo de reCaptcha

Imagen Obtenida de <http://www.eternagame.org/>

Originalmente, la finalidad de estos sistemas era simplemente la anteriormente mencionado. En la actualidad, este sistema se utiliza además para el perfeccionamiento de sistemas ROC/OCR (Reconocimiento Óptico de Caracteres/*Optic Character Recognition*).

El Crowdsourcing, según Geiger & Schader en *Crowdsourcing Information Systems* [2], se puede dividir en 4 categorías:

- **Crowdsolving:** Buscan una respuesta heterogénea global entre una gran cantidad de respuestas de un gran colectivo. Se suelen utilizar para encontrar respuestas a problemas complicados.
- **Crowdcreation:** Usuarios trabajan en común para ofrecer información. Como puede ser Wikipedia o YouTube
- **Crowdrating:** Sistema basado en votos, la solución aparece por la homogeneidad de los votos. Fue utilizado por la NASA en su aplicación Clickworkers², utilizada para detectar cráteres en asteroides

² <http://www.nasaclickworkers.com/>

- **Crowdprocessing:** modelo que se caracteriza por la realización de trabajos por distintas personas, que dos personas realicen el mismo trabajo y obtengan el mismo resultado añade veracidad a la prueba. La unión de todos los resultados es la solución final. Un ejemplo de esto lo podemos encontrar en Galaxy Zoo³.

Esta técnica depende de que la gente que esté dispuesta a realizar tareas gratis (o por una pequeña compensación económica). Ya que prácticamente todos los métodos de Crowdsourcing se basan completamente en la información suministrada por los *CrowdSourcers*, mantenerlos motivados es la tarea más importante para conseguir un buen sistema.

El hecho de la auto superación, aprender cosas nuevas o simplemente el hecho de entretenerse suelen predominar a las originadas por el dinero o motivos sociales externos. El objetivo principal es diseñar un sistema que genere sensaciones positivas sobre el Crowdsourcing consiguiendo que quede perfectamente cohesionado.

En la actualidad uno de los campos más estudiados para conseguir estos efectos ha sido el de la gamificación, es por ello que una de las mayores áreas donde se aplica esta sea el Crowdsourcing. Se ha detectado según una gran cantidad de estudios realizados sobre sistemas de Crowdsourcing, una relación directa entre la presencia de sistemas de gamificación y la implicación de los usuarios.

Por esta razón, cuanto mejor aplicada este la gamificación, mayor va a ser la implicación de la gente, ya que muchas veces una compensación económica no es suficiente. Aun así, es importante destacar que no todas las formas de gamificación se pueden aplicar en todos los métodos de Crowdsourcing

³ <https://www.galaxyzoo.org>

Gamificación

El termino gamificación proviene del inglés, *gamification*, el cual, a su vez, proviene nuevamente del termino anglosajón, *games*. Esto se debe a que, con esta técnica, lo que se pretende, es llevar las técnicas de los juegos a otros ambientes. Al hacerlo, lo que conseguimos es aumentar la fidelización y la atracción del sistema, consiguiendo así aumentar el volumen de usuarios activos. [3] [4]

Esta palabra es relativamente nueva, apareció alrededor del 2008 en el ámbito empresarial. Y ha visto como aumentaba rápidamente su popularidad en los últimos años. Además, no se prevé que sea una tendencia temporal, estudios como el realizado por el IEEE en *Everyone's a Gamer* [5] avisan de que para el 2020 el 85% de las tareas diarias estén gamificadas.

Para entender porque la gamificación ha tenido tanta importancia en los últimos años hay que comprender el contexto del término del que proviene, los juegos, más concretamente los videojuegos.

Según *The Entertainment Software Association* en su informe sobre 2016 [6] podemos sacar 3 datos muy interesantes sobre los juegos en Estados Unidos:

- Indican que en el 65% de los hogares hay al menos un dispositivo que puede ser utilizado para jugar.
- Que hay unos 155 millones de jugadores habituales, es decir, que juegan más de 3 horas por semana.
- Que se gastaron un total de 23.5 billones de dólares en juegos en 2015.

Estas cifras aumentan año tras año y se espera que siga así.

Componentes de la Gamificación

Para la definición de los componentes de la gamificación se ha decidido seguir un modelo derivado del desarrollado por la empresa *gamemarketing* [7], el cual divide en 9 componentes el sistema. En el modelo de cazasteroides vamos a dividirlo en 6 componentes, pues son los que este trabajo requiere.

Los cambios frente a este modelo, consistirían en la unión de las mecánicas y los componentes; la unión de comportamientos y dinámicas; y por último se han eliminado costes y beneficios pues no se va a tener en cuenta en este trabajo. [8]

Estos componentes serían los siguientes:

- **Plataformas:** Son los distintos sistemas sobre los que se va a desarrollar la aplicación y sobre los que se va a tener acceso.
- **Mecánicas:** son los componentes que forman el juego. Pueden ser tanto jugables como no.
- **Dinámicas:** definen como los jugadores interactúan con las mecánicas del juego.
- **Estéticas:** muestran cómo se tiene que sentir un usuario al realizar determinadas acciones.
- **Comportamientos:** patrones que se espera que los usuarios tengan al utilizar el sistema.
- **Jugadores:** Tipos de usuarios que va a haber en el sistema y sobre los que hay que desarrollar los comportamientos. Según Richard Bartle en su libro *Designing Virtual Worlds* [9], en un juego, se suelen determinar 4 patrones distintos de usuarios, estos son:
 - **El ambicioso:** busca ganar por encima de todo. Dando importancia a los rankings se mantienen a este tipo de usuarios, sobre todo si en ellos aparecen sus amigos o compañeros.
 - **El triunfador:** tipo de jugador cuya principal motivación es la de descubrir la mayor cantidad de contenido, obtener todos los logros y reconocimientos. Muy ligado al anterior tipo de jugador.
 - **El sociable:** su interés es social por encima de estratégico. Intenta compartir todos sus logros con el fin de aumentar su red de contactos. Le mantiene el poder relacionarse con los distintos jugadores.
 - **El explorador:** jugador cuya motivación es la auto superación, superar los desafíos más complejos o ser el primero en encontrar la solución.

Games With a Purpose

Games with a Purpose (a partir de ahora GWAP) o Human-Based Computation Games es una técnica de Crowdsourcing gamificada. Fue propuesta en 2004 por Luis von Ahn con el juego llamado ESP [10]. Este empresario guatemalteco es uno de los pioneros del Crowdsourcing, además del creador de *reCaptcha*, anteriormente mencionada, y Duolingo⁴, una de las mayores plataformas para el estudio de idiomas.

Esta técnica se podría considerar como una combinación de las dos técnicas vistas hasta este momento, Crowdsourcing y Gamificación.

El fin de esta técnica es que las personas realicen el trabajo que los ordenadores no pueden realizar por si solos. Estas tareas deben triviales para los humanos, pero complicadas para los ordenadores, por ejemplo, se pueden tratar de reconocimiento de formas o para la realización de estudios donde la interacción humana es requisito.

Al utilizar esta técnica, las personas realizamos tareas sencillas, las cuales, al unir las forman algo mucho más grande.

Pero el problema que tiene esta mentalidad es que los humanos necesitamos algún tipo de incentivo para realizar las cosas, aquí es donde entran en escena los juegos. La motivación principal de los GWAP es la de entretener por encima del interés de realizar tareas. Esto ayuda a que estos juegos puedan ser acogidos por una gran cantidad de personas.

Los campos sobre los que se pueden aplicar estas técnicas son innumerables ya que es una técnica flexible que permite adaptarse a cualesquiera que sean las necesidades. En el siguiente punto veremos ejemplos de juegos donde se han utilizados estas técnicas.

⁴ <https://es.duolingo.com>

Ejemplos de Games With a Purpose

Apetopia⁵

Apetopia es un juego desarrollado por la empresa Visual Computing⁶. Este juego entra en el género de *endless runner* en el cual tendrás que ir avanzando recogiendo monedas, las cuales aumentan los puntos al igual que la distancia recorrida.

Mientras tanto se deben ir esquivando los objetos que aparecen los cuales disminuyen la vida, mostrado en la Ilustración 2.

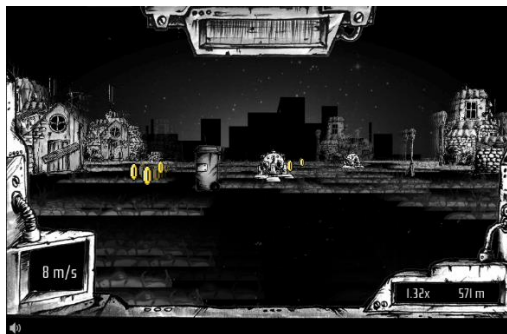


Ilustración 2: Apitopia: Pantalla del juego

Imagen Obtenida de <http://apetopia.visual-computing.com>

Al finalizar el juego puedes introducir un nombre para que aparezca en el ranking global además de tu posición en este, Ilustración 3. Además, se puede ver el estado del ranking global, Ilustración 4.



Ilustración 3: Apitopia: Posición y puntuación total

Imagen Obtenida de <http://apetopia.visual-computing.com>

⁵ <http://apetopia.visual-computing.com>

⁶ <http://visual-computing.com>



Ilustración 4: Apetopia, Ranking

Imagen Obtenida de <http://apetopia.visual-computing.com>

La tarea detrás de este juego es la de educar al ordenador sobre como las personas percibimos la diferencia entre los colores. Esto está implementado en forma de portales. Cada cierta distancia aparece un muro de piedra y dos portales con distintos colores, el usuario tiene que pasar a través del que crea que más se parece al cielo. Esto se puede apreciar en la Ilustración 5.



Ilustración 5: Apetopia, Puertas

Imagen Obtenida de <http://apetopia.visual-computing.com>

ARTigo⁷ (Nuevo ESP)

Siguiendo la estela del ESP (el primer GWAP), ARTigo es un juego de etiquetado de imágenes.

Este juego se trata de una competición con otro usuario online, donde, en una secuencia de imágenes, durante un minuto cada usuario tendrá que etiquetar las imágenes, si el sistema la detecta como probable te otorga puntos, esto se puede ver en la Ilustración 6. Al finalizar este minuto se presenta una imagen nueva.

Cuando finaliza la secuencia de imágenes aparecen los resultados finales y te dan una puntuación final y una posición en el ranking, Ilustración 7.



Ilustración 6: ARTigo, Pantalla del Juego
Imagen Obtenida de <http://www.artigo.org>

A screenshot of the ARTigo game interface showing the rankings page. The page has a sidebar on the left with navigation links and a main content area on the right. The main area displays three tables of rankings: 'Daily Highscores', 'Weekly Highscores', and 'Monthly Highscores'. Each table lists users and their scores. The 'Daily Highscores' table shows users like 'User' and 'Score'. The 'Weekly Highscores' table shows users like 'User' and 'Score'. The 'Monthly Highscores' table shows users like 'User' and 'Score'. The interface is clean and modern, with a white background and blue accents.

Ilustración 7: ARTigo, Pantalla de Rankings
Imagen Obtenida de <http://www.artigo.org>

⁷ <https://artigo.org/>

EteRNA⁸

EteRNA es un juego gamificado cuyo fin es el de ayudar a los ordenadores a mejorar las predicciones sobre como las cadenas de ARN (Ácido Ribonucleico) se doblan. Algunas de las soluciones planteadas son probadas en la realidad y comparadas con los resultados obtenidos.

No se necesita ningún tipo de conocimientos previos para utilizar este sistema ya que tiene un tutorial en los primeros niveles, tras esto, se podrán empezar a realizar experimentos reales, como podemos apreciar en la Ilustración 8.



Ilustración 8: EteRNA, Pantalla Principal

Imagen obtenida de www.eternagame.org

Cuando completamos un nivel aparece un ranking según la puntuación obtenida, además de una pequeña tenemos un sistema de puntuación con rankings en cada nivel, cuando completamos un nivel tenemos un recordatorio sobre cómo se unen los distintos componentes del RNA, como se puede ver en la Ilustración 9.

⁸ www.eternagame.org/

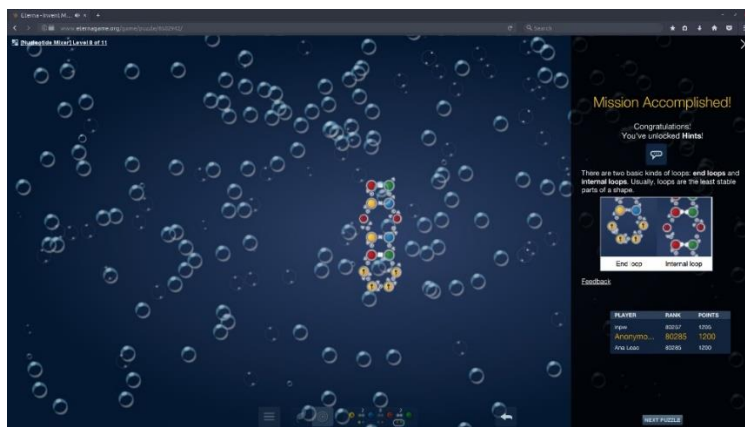


Ilustración 9: EterRNA, Pantalla nivel completado

Imagen Obtenida de <http://www.eternagame.org>

Eyewire⁹

Eyewire es una aplicación para el mapeado de neuronas. Estas imágenes se obtienen de la retina de un ratón. Para la realización de estos reconocimientos se dividen las imágenes en zonas de unos 4,5 micrones donde ya está descubierta parte de la neurona y se tienen que descubrir el resto del sector. No se necesita ningún tipo de trasfondo científico.

La primera vez que se abre la aplicación, y tras registrarte, el uso de la aplicación resulta muy sencillo al ser correctamente guiado. Tras unas primeras pantallas de tutorial donde las ayudas van disminuyendo hasta finalizar el tutorial y empezar a mapear nuevas zonas, las cuales no han sido aún exploradas aún por nadie.

Esta aplicación posee el sistema de gamificación más completo de los que se han estudiado para este trabajo. Algunas de las dinámicas implementadas en este sistema son las siguientes:

- **Sistema de puntuación:** el cual depende de velocidad, habilidad y precisión de la observación.
- **Sistema de Ranking:** el cual podríamos dividir por bloque de fechas (hoy, esta semana, este mes), y por el grupo de gente que aparece (global, solo amigos).

⁹ <https://eyewire.org>

- **Insignias:** obtenidas al completar desafíos o ir completando sectores.
- **Niveles:** posee un sistema de niveles en función del trabajo desarrollado, estos otorgan mayores privilegios en el sistema.
- **Retos:** que permiten desafiar a tus amigos.

Además, se muestra a los usuarios distintas zonas en función de su experiencia, es decir, se aplica un Sistema de Reputación, técnica que será estudiada más adelante.

En la ilustración 10 podemos apreciar la pantalla principal del sistema, mientras que en la 11 podemos apreciar la pantalla de finalización de un mapeado, en ambas están presentes las dinámicas descritas anteriormente, como pueden ser los sistemas de ranking, niveles, insignias.

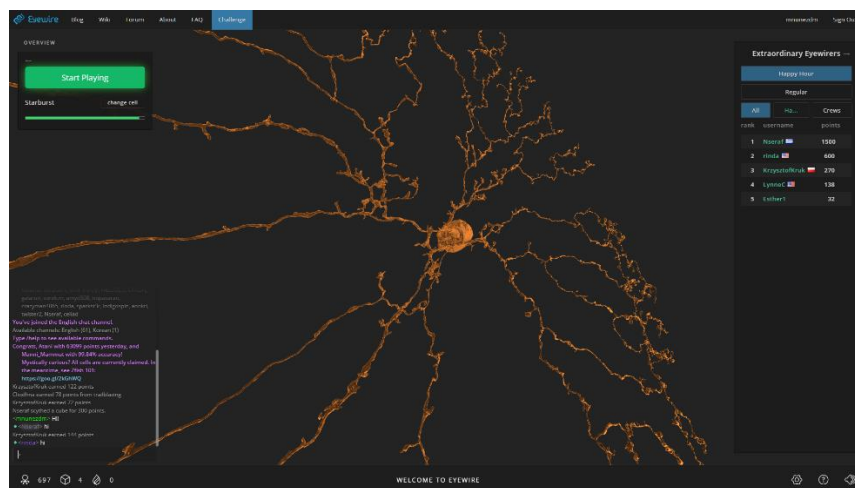


Ilustración 10: Eyewire, Pantalla Principal
Imagen Obtenida de <http://eyewire.org>

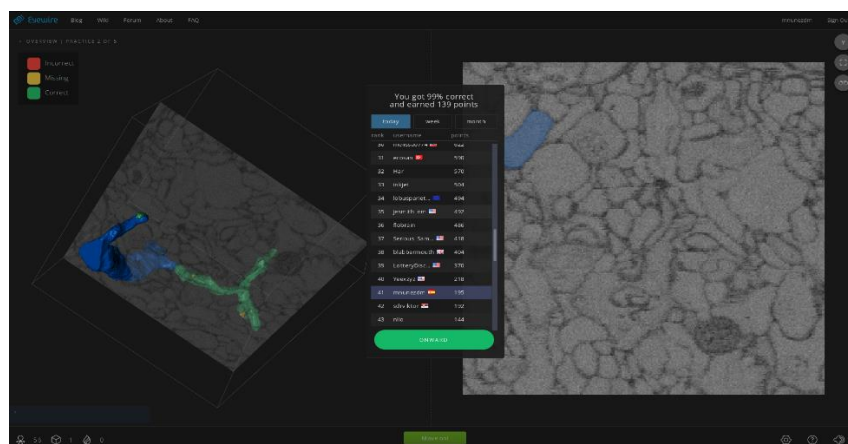


Ilustración 11: Eyewire, Pantalla nivel completado
Imagen Obtenida de <http://eyewire.org>

Reputation Systems

Los Reputation Systems o Sistemas de Reputación son programas que permiten valorar a los usuarios según las acciones que han realizado con el fin de otorgarles credibilidad a ellos mismos y por lo tanto a las acciones que realizan.

Actualmente son usados de la misma manera que los sistemas de gamificación, además de para otras razones que veremos más adelante, y es muy común verlos juntos. De la misma manera que en los sistemas gamificados se premia la cantidad de las aportaciones, en los sistemas de reputación prima la calidad de estas. Esto quiere decir que en un sistema de reputación realizar una aportación errónea penalizaría al usuario mientras que en un sistema gamificado simplemente no aportaría nada.

El objetivo de estos sistemas es asegurar un sistema confiable esto se consigue asegurando que los componentes lo sean, por lo tanto, con un sistema donde se premie estas acciones; y además tengan más valor las acciones realizadas por personas confiables se permitirá llegar a este sistema buscado.

Componentes de los Sistemas de Reputación

La reputación en estos sistemas se puede obtener de distintas formas, como, por ejemplo, según indica la página feverbee [11], se pueden dividir en 7 distintas maneras. Todos estos componentes se pueden implementar en el mismo sistema, siempre y cuando estén correctamente utilizados. Los componentes son los siguientes:

- **Por Niveles de Acceso:** dar distintos niveles (de cosas que pueden hacer) a distintos usuarios del sistema con el fin de eliminar tareas automáticas del sistema. Esta herramienta suele ser utilizada en comunidades, a modo de dar privilegios de moderador a usuarios permitiéndoles eliminar contenido, bloquear usuarios etcétera.
- **Por Antigüedad:** premiar a los usuarios más antiguos del sistema, esto ayuda a mantener a los usuarios más antiguos.

- **Por Cantidad de Contribuciones:** premiar a cada contribución, esto ayuda a que los usuarios sean más activos, aunque esta herramienta, con el fin de evitar que se haga mal uso de ella, debe ser implementada junto a otras.
- **Por Calidad de Contribuciones:** herramienta principal de los sistemas de reputación donde se premia a los usuarios por buenas contribuciones y se penaliza por malas.
- **Por Frecuencia:** realizar contribuciones de forma regular aumenta la reputación.
- **Por Distinciones Especiales:** otorgar insignias positivas o negativas por acciones realizadas, estas acciones no deberían ser las relacionadas con las actividades principales del sistema, pues de eso ya se encarga la herramienta de Calidad de las Contribuciones vista anteriormente. En esta, una mala acción supondría una insignia que reste reputación, como por ejemplo una disputa fuera de tono con otro usuario o violar las reglas del sistema; mientras que una buena otorgaría un distintivo que aporte reputación, como podría ser una donación.
- **Por Ranking:** los resultados en los rankings añaden respeto.

Ejemplos de Sistemas de Reputación

- **Portales de ventas:** es uno de los campos donde en más porcentaje se usan sistemas de reputación. Resulta difícil encontrar un portal de eCommerce donde no se valore los artículos, vendedores o compradores. El fin de aplicar estas técnicas varía según donde se realice, por ejemplo:
 - **En vendedores:** en portales como Aliexpress¹⁰ o eBay¹¹ resulta imprescindible tener una alta reputación, pues va muy ligada a las ventas que se obtienen, algo que refleja claramente esta postura es el comportamiento de los vendedores ante cualquier problema, donde proceden al reenvío o devolver el dinero con tal de no obtener una mala valoración, y en el caso de no obtener las 5 estrellas, te escriben por privado preguntándote si has tenido algún problema.
 - **En productos:** destacar las valoraciones de productos que tiene Amazon. Esta empresa, a los artículos con mayor valoración se encarga ella misma de realizar

¹⁰ <https://es.aliexpress.com>

¹¹ www.ebay.es

la distribución y el servicio post venta, aumentando así la visibilidad de ellos. Actualmente no creo que haya nadie que compre artículos de otras alternativas cuando Amazon Premium esté disponible.

- **En compradores:** cobra la mayor importancia en portales de venta de segunda mano donde los vendedores quieren evitar en gran medida un fraude. Esto puede verse en sistemas como *WallaPop*¹².
- **Comunidades online:** otro de los capos donde resulta muy común añadir estos sistemas es en foros o comunidades. Como por ejemplo XDA-Developers¹³, el mayor portal de desarrolladores móviles, donde, los usuarios expertos pueden comentar en distintos temas.
- **Motores de Búsqueda:** todos los grandes motores de búsqueda tienen complejos sistemas de posicionamiento donde premian, es decir, ordenan en función de los valores obtenidos de esta puntuación los resultados obtenidos.
- **Seguridad Online:** portales como *TrustedSource*¹⁴, solución ofrecida por Intel y McAfee, catalogan sitios según su tipo y según las vulnerabilidades que pueden tener.

¹² <https://es.wallapop.com>

¹³ <https://www.xda-developers.com/>

¹⁴ <https://trustedsource.org/>

Análisis y Propuesta de Mejora

Se va realizar un análisis del sistema actual y una propuesta de mejora. Se han propuesto una gran cantidad de cambios los cuales no se van a implementar todos en este Trabajo de Fin de Grado, pero podrán servir como guía para trabajo posterior.

Sistema Actual

El sistema actual se trata de una arquitectura REST. El esquema del sistema actual se puede ver en la Ilustración 12 y va a ser explicada a continuación:

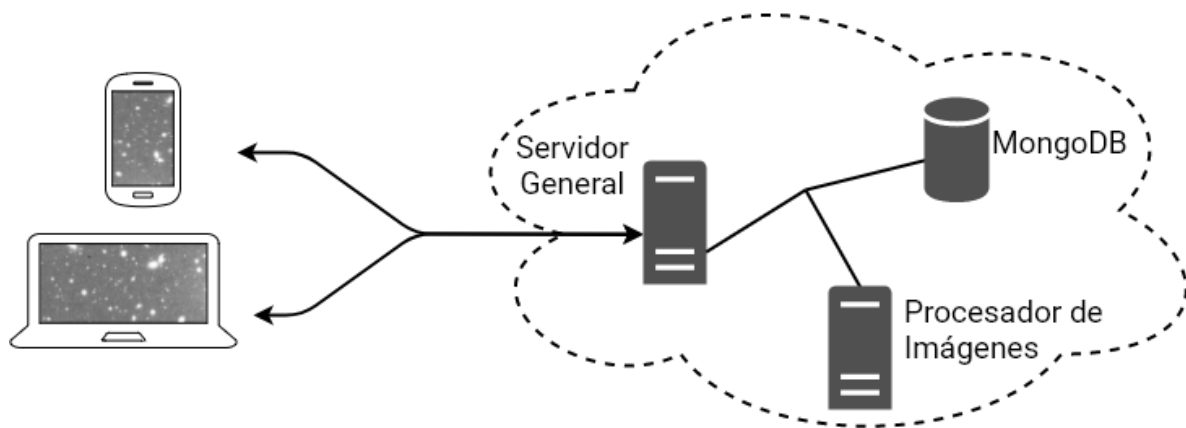


Ilustración 12: Arquitectura cazasteroides Original

Generado con la Aplicación Draw.io

- **Cliente:** programado en HTML5, CSS y JavaScript, concretamente utilizando el framework AngularJS. Partiendo de este código, utilizando la herramienta Apache Cordova se construyen aplicaciones nativas para Android e iOS. Ahora mismo solo hay dos versiones accesibles, estas son:
 - **Aplicación Android:** la cual puede ser descargada desde la Play Store
 - **Aplicación Web Responsive:** a la cual se puede acceder desde la página web de cazasteroides.
- **Servidor:** tendríamos un servidor programado en Node.js.
- **Base de datos:** concretamente MongoDB, tipo de base de datos que se caracteriza por tener un esquema dinámico de datos, aunque utilizaremos Mongoose, un plugin de Node.js, que aparte de realizar las consultas, permite modelar los datos.

- **Procesador de imágenes:** es el encargado de realizar todo el procesamiento de las imágenes de los telescopios, como es el recorte de la imagen original, la escala, y la compresión. Además, genera la animación de la observación.

Aplicación

Se ha realizado un rediseño de la aplicación siguiendo las guías de Material Design¹⁵, guía desarrollada por Google. Estos patrones de diseño son, actualmente, una de las referencias más importantes para el desarrollo de UI (User Interface, Interfaz de Usuario).

Este rediseño se debe a dos razones fundamentales:

- Actualmente, la interfaz de la aplicación es poco vistosa y anticuada.
- La navegación en la aplicación no es la adecuada para esta aplicación

Navegación

Para lograr esta mejor navegación se ha decidido aumentar el número de pantallas que tiene la aplicación. Para ilustrar estos cambios tenemos en la Ilustración 13 la estructura de la aplicación actual mientras que en la Ilustración 14 tenemos la estructura de la aplicación propuesta.

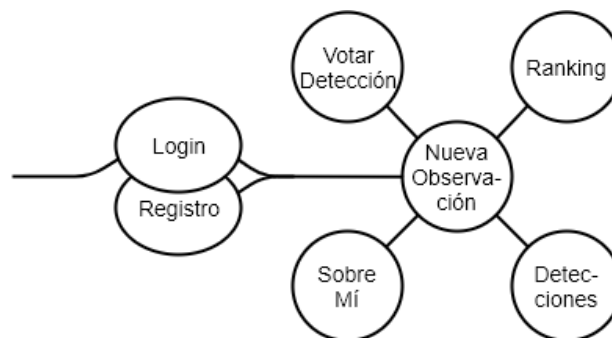


Ilustración 13: Pantallas Aplicación Actual

Generado con la Aplicación Draw.io

¹⁵ <https://material.io/>

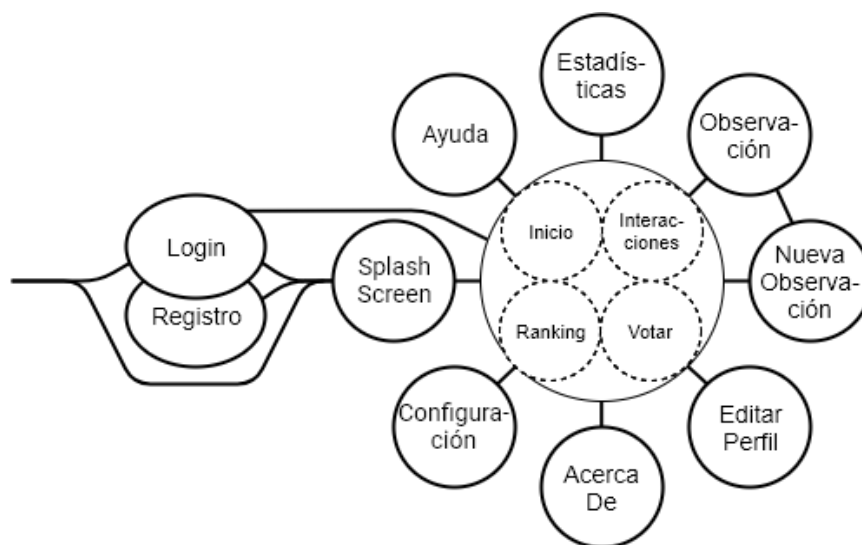


Ilustración 14: Pantallas Aplicación Propuesta

Generado con la Aplicación Draw.io

1. **Login:** esta pantalla va a ser la primera pantalla que cargara la aplicación. Permitirá a los usuarios iniciar sesión con una cuenta ya creada, registrarse o pedir un reseteo de contraseña.
2. **Registro:** permite al usuario registrarse, ya sea por medio de correo y contraseña o utilizando el login por token¹⁶ con Google o Facebook.
3. **Splash Screen:** esta va a ser una pantalla de carga, cuando ya hayamos iniciado sesión en la aplicación esta pasará a ser la primera pantalla que abrirá la aplicación. Realizara la carga de todos los datos que necesite cargar para su correcto funcionamiento.
4. **Pantalla Principal:** sería la primera pantalla en la pila, es decir, al pulsar atrás deberíamos salir de la aplicación. Esta pantalla se ha pensado como una barra de navegación inferior con los botones a las acciones principales que va a tener nuestra aplicación. Estas acciones cambiarían el panel mostrado en la pantalla. Los botones serían los siguientes:
 - a. **Nueva Observación:** llevaría a la pantalla **Nueva Observación**.

¹⁶ <https://developers.google.com/identity/protocols/OAuth2>

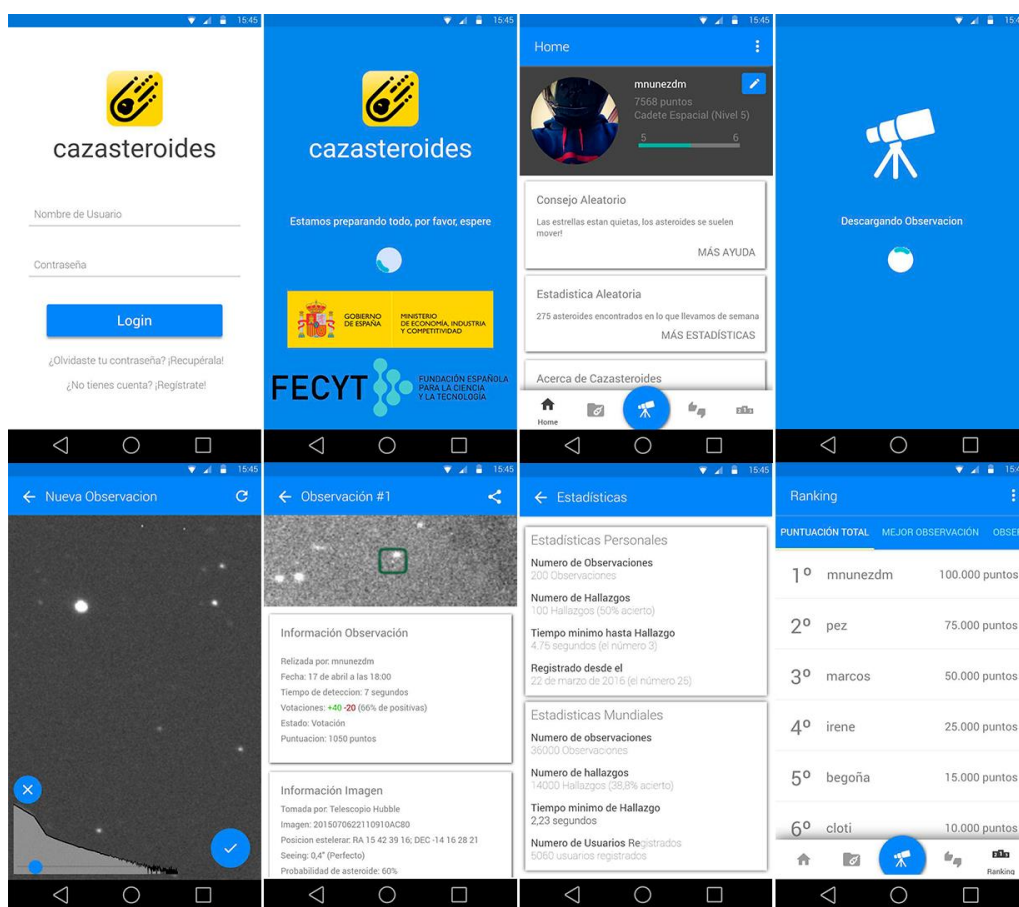
- b. **Home:** se trata del panel por defecto, este panel se ha pensado como un resumen de toda la actividad del usuario. Contiene información del usuario así como un acceso a la edición de esta información, estadísticas y él acerca de.
 - c. **Interacciones:** en este panel aparecerán todas las observaciones y las visualizaciones realizadas por el usuario (información separada en dos pestañas). Esto permite al usuario ver el resultado de sus interacciones.
 - d. **Votaciones:** en este panel aparecerá un listado de unas pocas observaciones realizadas por otros usuarios permitiendo al usuario ver la imagen y votar si creen o no que se trate de un asteroide.
 - e. **Ranking:** este panel tendrá distintas pestañas con el top 10 de usuarios de distintos rankings, además de la posición actual del usuario.
5. **Editar Perfil:** en esta pantalla el usuario podrá editar su perfil.
 6. **Estadísticas:** en esta pantalla se mostrarán distintas estadísticas sobre el uso de la aplicación, tanto globales como del propio usuario.
 7. **Acerca De:** en esta pantalla se mostrará información de la aplicación, como por ejemplo mostrar el propósito de la aplicación (ya que muchos usuarios la usan sin saberlo), información sobre los integrantes y más datos relevantes.
 8. **Carga Observación:** esta pantalla será una pantalla de carga la cual no se superará hasta que no se hayan descargado todas las imágenes, es necesaria esta pantalla pues en conexiones lentas se han llegado a medir 20 segundos de carga lo cual repercutía directamente en la puntuación obtenida. Tras esta pantalla aparecerá la pantalla Nueva Observación.
 9. **Nueva Observación:** esta pantalla será muy similar a la de la versión anterior, salvo que la comunicación con el sistema se hará al confirmar una observación y no con cada toque lo cual suponía demasiadas peticiones al sistema que influían en el rendimiento de la aplicación.
 10. **Observación Detallada:** esta pantalla tendrá información sobre la observación, la pantalla será la misma ya sea una observación propia o una observación de otra persona. Tendrá información simple, compleja y datos del ranking, como podría ser tiempo, puntuación o dificultad. Además, mostraría en el estado en el que se encuentra esta observación.

Apariencia

Utilizando la aplicación Justinmind¹⁷ se ha realizado un prototipo de alto nivel con el rediseño propuesto, algunas capturas de este rediseño se pueden ver en la Ilustración 15.

Este diseño se encuentra desplegado en un servidor de Heroku¹⁸, un proveedor de plataformas como servicio (PaaS¹⁹), y puede ser accedido mediante un navegador (siempre y cuando no sea Google Chrome) desde la siguiente URL:

<http://prototype-cazasteroides.herokuapp.com/>



*Ilustración 15: Pantallas Aplicación Rediseñada
Generadas con la Aplicación Justinmind*

¹⁷ <https://www.justinmind.com>

¹⁸ <https://www.heroku.com/about>

¹⁹ <https://azure.microsoft.com/es-es/overview/what-is-paas/>

Sistema de Gamificación

Utilizando los conocimientos adquiridos al analizar las distintas técnicas de gamificación y reputación vistas en el anterior capítulo (Estado del Arte) se ha decidido ampliar el actual sistema con el fin de lograr unos resultados más acordes. Los distintos apartados a desarrollar, así como las mejoras planteadas se van a especificar a continuación:

Plataforma

En estos momentos las plataformas donde está integrado el sistema son solamente el propio servidor Node.js y el sistema operativo Android donde hay una aplicación funcional.

Como propuesta de mejora para este apartado se propone que la aplicación esté disponible en los dos grandes sistemas operativos móviles, Android e iOS. Además de lograr una mayor integración con las Redes Sociales, convirtiéndolas en fundamentales para conseguir la parte social de este sistema.

Mecánicas

Las mecánicas de juego propuestas para Cazasteroides son las siguientes.

- **Puntuación:** actualmente ya se encuentra implantado.
- **Karma:** un sistema de reputación que aumentaría del mismo modo que la puntuación. Nunca se reduce este nivel. Una mayor descripción de esta mecánica se encuentra en el apartado Ampliaciones al final de este capítulo.
- **Insignias:** distintas medallas que obtendrían los usuarios de distintas maneras.
- **Retos:** misiones que los usuarios pueden lanzar a otros.
- **Premios reales:** en el caso de cazasteroides, se va a permitir utilizar los puntos obtenidos para canjearlos por tiempo de operación de la red de telescopios Gloria, esta característica también se encuentra incluida.
- **Clasificaciones:** al igual que el sistema de puntuación, ya se encuentra implementando y se va a rediseñar para incrementar la información que aportan.

Dinámicas

Las dinámicas pensadas para cazasteroides son las siguientes:

- **Realizar observaciones:** uno de los objetivos principales de la aplicación. Realizando observaciones (es decir buscando asteroides en imágenes). Se obtienen puntos y karma según distintos parámetros.
- **Realizar votaciones:** el otro objetivo principal del juego. Permitir votar a los usuarios las observaciones realizadas por otros usuarios.
- **Completar misiones diarias:** simples misiones que den puntos por completar algunas tareas, ayudan a que el usuario sienta la necesidad de entrar diariamente.
- **Completar logros:** misiones más complejas y largas que obliguen al usuario a jugar mucho y a usar todas las características. Estos logros otorgaran insignias al usuario. Muy importante conectar esto con plataformas como Google Play Games, para que estos logros aumenten también el perfil del usuario no solo en el juego lo que aumenta las ganas de jugar.
- **Visualizar Rankings:** permitir al usuario ver como se encuentra el con respecto al mundo.
- **Compartir observaciones:** permite al usuario compartir por redes sociales una observación que ha realizado.
- **Lanzar retos:** permite al usuario lanzar retos por redes sociales para competir de forma directa con sus amigos.
- **Comprar tiempo** en el observatorio: permitir a los usuarios gastar los puntos acumulados en algo real, dando sentido al tiempo invertido en el juego.

Las dos dinámicas principales del sistema de cazasteroides, son las dos primeras, Realizar Observaciones y Realizar Votaciones. Son el objetivo final del proyecto y son las únicas que no están directamente relacionadas con los procesos de gamificación.

Estéticas

En nuestro caso las distintas estéticas presentes son:

- **Desafiante:** queremos que el usuario se esfuerce en superarse a sí mismo y a los demás.
- **Descubrimiento:** queremos que el usuario aprenda y explore un mundo que nunca haya visto.
- **Expresivo:** queremos que el usuario se comunique con otros usuarios.

Sistemas de Reputación

Karma

El sistema de Karma, el cual se encuentra en el punto de Mecánicas. Este Karma se trata de un Sistema de Reputación analizado en el capítulo del estado del arte.

Como ya vimos, lo que se pretende conseguir con estos sistemas de reputación es premiar a los mejores jugadores dándoles mayor poder en el sistema. Con esto, lo que se pretende expresar, es que las observaciones de los usuarios con mayor nivel de karma van tener mayor importancia, las votaciones mayor relevancia y las imágenes a mostrar van a tener mayor dificultad, permitiéndoles así, conseguir más puntos.

Esta mejora se expandirá en el próximo capítulo al ser el sobre lo que se ha centrado el trabajo de desarrollo.

Cambios Realizados

Este capítulo va a constar de toda la documentación de la implementación que se va a realizar en este trabajo en el sistema.

Para el proceso de desarrollo de la aplicación se ha utilizado la plataforma de control de versiones GitHub. Actualmente, el código, se encuentra en la siguiente URL.

<https://github.com/mnunezdm/cazasteroides/tree/master/codigo>

Servidor de Karma

El equipo de cazasteroides ha solicitado 3 algoritmos para implementar distintas dinámicas del sistema. Estos algoritmos tienen en común el karma como mecánica.

El primero de estos algoritmos sirva para obtener el nivel del karma, el cual en el sistema actual no se encuentra implementado, mientras que los dos siguientes serán variaciones de algoritmos ya desarrollados que no tienen en cuenta el karma.

Se busca que estos algoritmos sean modulables y fácilmente configurables, permitiendo así, el poder variar las ponderaciones de las entradas incluso en caliente. De este modo, permitimos realizar pequeños ajustes utilizando el tráfico real de la aplicación para lograr el equilibrio en el sistema.

Este servidor está pensado como un módulo adicional totalmente transparente al usuario. La comunicación se haría entre servidor general y servidor del karma tal y como se puede ver en la Ilustración 16.

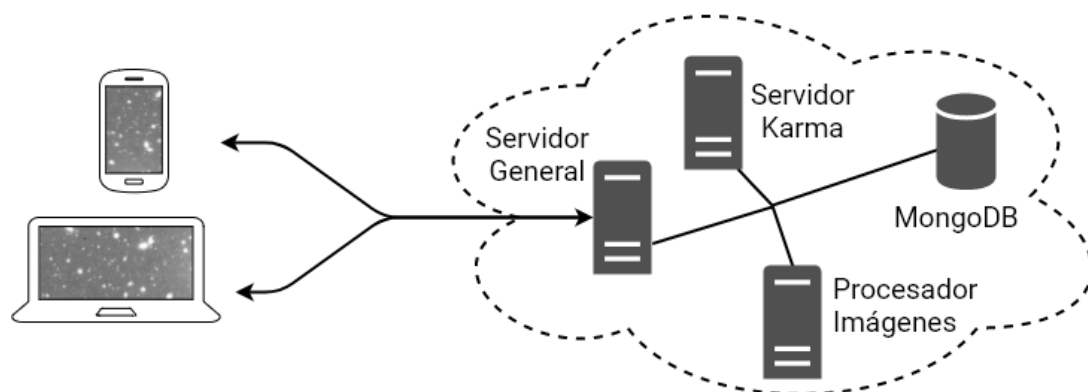


Ilustración 16: Esquema Servidores

Un ejemplo de caso de uso de esta aplicación de todos sus módulos funcionando de forma coordinada se puede ver en la Ilustración 17.

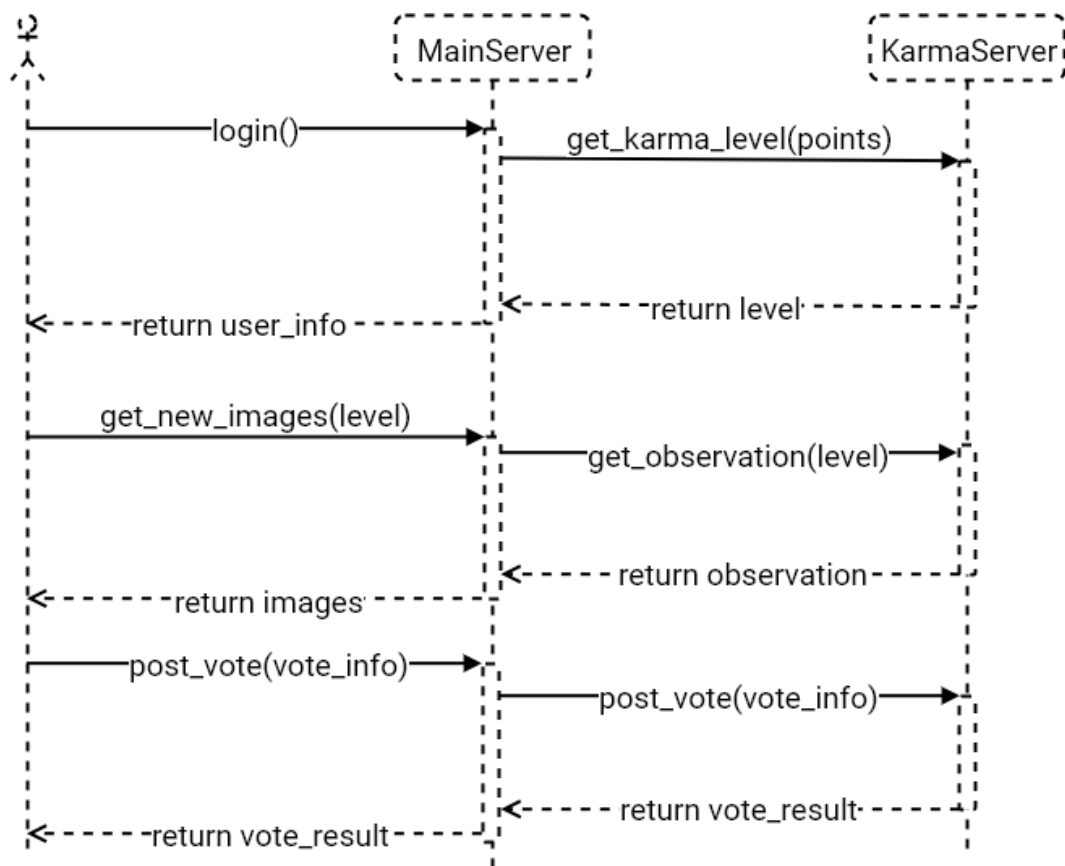


Ilustración 17: Caso de uso módulo de karma

Generado con la Aplicación Draw.io

En este diagrama de secuencia podemos ver cuáles serían las operaciones iniciales de un usuario al abrir la aplicación.

1. Según entrase haría una petición de inicio de sesión, el servidor, a su vez este haría una petición al servidor del karma para calcular el nivel a los puntos del usuario, finalmente se devolvería al usuario toda la información.
2. El siguiente paso sería solicitar un nuevo quinteto de imágenes sobre el que realizar la observación. Para ello, el usuario mandaría una petición al servidor en la que enviaría su nivel de karma. El servidor de karma realizaría una petición al módulo del karma y devolvería su sugerencia de observación. Si no se devolviese ninguna observación, el servidor general la asignaría automáticamente.

3. Por último, el usuario enviaría una nueva observación, por lo que el servidor redirigiría la petición al servidor de karma que realizaría todos los cálculos necesarios y devolvería el estado de la observación al servidor general y este al usuario.

Además, se ha decidido añadir una guía para la instalación del entorno y un resumen con la declaración de las interfaces, podemos encontrarlos en los Anexo 1 y Anexo 2 respectivamente.

Herramientas y lenguajes

Para implementar este algoritmo se ha decidido hacerlo montando un microservicio con una arquitectura API REST.

Se ha decidido utilizar Python como lenguaje de programación por 3 razones:

- **Rendimiento** con el trabajo con listas y operaciones aritméticas complejas, operaciones necesarias para la implementación de los algoritmos a desarrollar.
- **Ligereza**, es capaz de trabajar en equipos de pocas prestaciones.
- **Multiplataforma**, funciona de la misma manera independientemente del sistema operativo sobre el que esté funcionando,
- **Código legible**, lo que favorece la reutilización del código, necesario ya que va a tener que ser utilizado por otros desarrolladores.

Para implementar el servidor se ha decidido utilizar el framework Flask²⁰, el cual implementa a su vez el servidor Werkzeug²¹. Con esto lo que conseguimos es realizar todo el enrutamiento que nuestro servidor requiere.

Para facilitar el diseño de la arquitectura se ha decidido realizar un diseño modular de la aplicación. Esto lo conseguimos utilizando el concepto de *Blueprint*²² en Flask

Un Blueprint no es más que una Aplicación Flask con menos funcionalidades. Este módulo se conecta a otra aplicación Flask asociándola a uno o varios prefijos y/o

²⁰ <http://flask.pocoo.org/>

²¹ <http://werkzeug.pocoo.org/>

²² <http://flask.pocoo.org/docs/0.12/blueprints/>

subdominios. Con los Blueprints, aparte de simplificar la aplicación conseguimos una mayor configuración individual, pudiendo asociar un tipo u otro de control de errores, de plantillas o permisos.

La persistencia de datos la logramos utilizando una base de datos SQLITE²³. La característica principal de una base de datos SQLITE. Las razones principales por las que se ha decidido utilizar esta base de datos son:

- **Ligereza:** decenas de miles de registros no alcanzan los 500 KB.
- **Portabilidad:** se trata únicamente de un archivo, independiente de contexto.
- **Sin Servidor:** No necesita estar sobre un servidor para interactuar con ella.

Por último, se ha utilizado un ORM²⁴ (Object Relational Management, o Mapeo Objeto Relacional) para abstraer las interacciones con la base de datos. Se ha utilizado específicamente la herramienta SQLALCHEMY²⁵ junto con la herramienta Flask-Migrate²⁶. Con estas herramientas lo que conseguimos es:

- Automatizar la generación y actualización de tablas según los modelos configurados.
- Trabajar con múltiples bases de datos lo que nos permite tener entornos de testeo.
- Posibilidad de trabajar directamente sobre la base de datos sin abstracción.
- Permite la configuración de las relaciones, permitiendo configurar como se va a traer la información de la base de datos.

Arquitectura

Se ha buscado conseguir una arquitectura limpia y comprensible, para ello se ha procurado seguir las guías de diseño S.O.L.I.D.²⁷ propuestas en por Robert C Martin en su libro *Code Clean* [12]. Esta guía de diseño se divide en 5 principios, estos principios

²³ <https://www.sqlite.org>

²⁴ <http://www.tuprogramacion.com/glosario/que-es-un-orm>

²⁵ <https://www.sqlalchemy.org>

²⁶ <https://flask-migrate.readthedocs.io/en/latest/>

²⁷ <https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design>

se encuentran explicados en el Anexo 3. Además de seguir las guías de diseño específicas de Python descritas en el PEP8²⁸.

La arquitectura final del sistema se puede ver descrita en la Ilustración 28 la cual se puede encontrar en el Anexo 4. Donde cada uno de los módulos del servidor se encuentra colocado acorde al nivel de esquema. Se ha buscado que esta arquitectura sea lo más clara posible. A continuación, se explicarán cuáles son las funciones de cada uno de los componentes.

- **manage**: se trata del manager de nuestra aplicación, todas las operaciones con el servidor se realizan desde este módulo. Actualmente, los parámetros que tiene implementado el manage son:
 - **db**: permite crear la base de datos (db init), realizar las migraciones (db migrate), actualizarla (db upgrade) entre otras.
 - **runserver**: lanza el servidor de la aplicación.
 - **runtests**: lanza la batería de tests configurados.
- **app**: se trata de la aplicación principal del servidor, tiene tres funcionalidades principales.
 - Servir de enrutador para el resto de módulos.
 - Realiza la tarea de registrar todas las interacciones y errores.
 - Maneja los errores que no han podido ser tratados por los módulos.
- **modules**: en este módulo encontraríamos todos los blueprints con los algoritmos que hemos mencionados al comienzo de este capítulo y que serán explicados en profundidad en el siguiente punto. La estructura de todos los módulos es la misma, en ambos tendremos dos partes:
 - **init**: esta parte se encargará de la erutación, además iniciaremos el modulo y, por último, comprobaremos que las llamadas sean las correctas.
 - **provider**: este módulo implementara el algoritmo asociado al módulo.

²⁸ <https://www.python.org/dev/peps/pep-0008/>

- **data:** este paquete contendrá todos los módulos asociados con los objetos y el acceso a los datos. Tendremos dos tipos de componente:
 - **content_resolver:** este módulo se encarga de toda la interacción con la base de datos. Siendo este el único modulo que opera con ella.
 - **models:** este paquete tiene todas las clases asociadas a los datos que hay en el sistema, se encarga además de determinar cuáles van a ser las tablas, con los atributos y las relaciones entre ellos.
- **Utils:** en este paquete tendremos módulos para la realización de tareas comunes entre módulos. Tendremos tres tipos de métodos:
 - **print:** este módulo tendrá todos los métodos para la depuración y el formateo de las salidas por consola del sistema.
 - **responses:** en este módulo tendremos el método generador de las respuestas.
 - **timer:** tendremos los métodos para realizar las mediciones, exactamente tenemos dos métodos, un método que calculo el tiempo actual, y otro método que pasado un tiempo calcula la diferencia con el tiempo actual.
- **Config:** tiene los parámetros de configuración de la aplicación Flask además de todos los valores por defecto de los distintos módulos.

Algoritmos

Cálculo del Karma

Para el cálculo de este sistema de reputación se ha decidido dividir el progreso en niveles, a los cuales se iría llegando acumulando puntos, los cuales pueden ser obtenidos de distintas maneras. La manera principal para la obtención de estos puntos, y la que se va a utilizar de referencia en este algoritmo es la dinámica de la realización de observaciones.

Actualmente, para el cálculo de esta puntuación se utiliza el siguiente, el cual consta de 5 entradas ponderadas. Estas entradas son las siguientes.

1. **Tiempo (40%):** El usuario tendrá más puntos si encuentra el Asteroide más rápido.
2. **Orden en la detección (30%):** El primero en detectar un candidato tendrá la máxima puntuación.

3. **Probabilidad de encontrar un asteroide en el campo (10%):** (se calcula a partir de la latitud eclíptica del campo). Hay un header PROBA con ese valor. Para campos donde la probabilidad de encontrar un asteroide sea alta los puntos serán menos.
4. **Brillo o magnitud del asteroide (10%):** Si el brillo del asteroide es alto tendrá menos puntos que la detección de asteroides débiles. Al final del apartado coloco los detalles para calcular la magnitud de un asteroide a partir del máximo de luz.
5. **Seeing o FWHM (10%):** Existirá un parámetro en el header llamado FWHM que indicará el grado de turbulencia o "seeing" de la noche. Detectar un asteroide con mal seeing debería tener más puntos que con una buena noche.

La ecuación final sumando todas las entradas es la siguiente Ecuación 1.

PuntosPorObservacion

$$= 40\% \text{ Tiempo} + 30\% \text{ Orden} + 10\% \text{ Prob} + 10\% \text{ Brillo} + 10\% \text{ Seeing}$$

Ecuación 1: Cálculo de los puntos por Observación

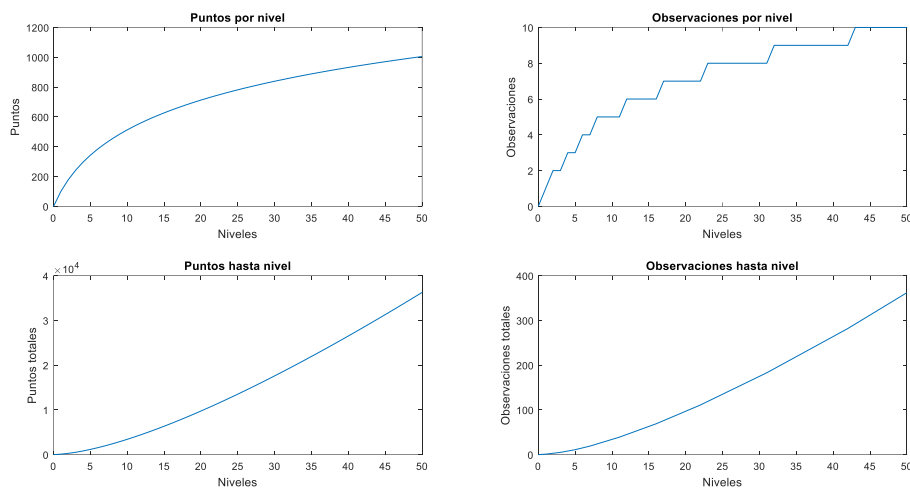
Para el cálculo de los puntos necesarios para subir de nivel en este sistema se ha decidido utilizar una ecuación logarítmica, la cual se caracteriza por tener un valor muy bajo en un principio, subir muy rápido y acabar manteniéndose en un valor elevado, pero sin que aumente demasiado.

La ecuación utilizada para la obtención de estos valores se trata de la Ecuación 2 que se encuentra a continuación.

$$PuntosPorNivel(nivel) = \log\left(\frac{nivel}{3} + 1\right) * 350$$

Ecuación 2: Cálculo de los puntos por nivel

Como podemos ver, esta ecuación se trata de una logarítmica escalada y decelerada.



*Ilustración 18: Graficas Puntuación
Generado con la Aplicación Matlab*

Como podemos ver estas graficas parten de un valor muy bajo donde los usuarios subirían rápidamente de nivel, durante los primeros niveles, este valor se iría incrementando de forma muy rápida también pero este aumento de puntuación se reduciría según va aumentando los niveles hasta apenas diferenciarse entre nivel y nivel. Esto se debe a que si se aumenta siempre en la misma proporción los niveles más altos requerirían demasiado trabajo lo que llevaría a los usuarios a dejar de intentar subir de nivel porque lo verían demasiado complicado.

Gestión de políticas

Además, se ha decidido configurar este módulo para poder gestionar la política de puntuación. Esto permite realizar configuraciones de forma programática sobre el modulo evaluador, además de tener varias políticas para distintas interacciones con el servidor. Como puede ser una política para el nivel asociado a la detección de asteroides, y una política distinta para el nivel asociado a la votación de observaciones.

Actualmente soporta CRUD, es decir, las cuatro operaciones básicas con bases de datos, estas son:

- **Create:** crear registros en bases de datos.
- **Retreive:** leer registros de la base de datos.
- **Update:** actualizar registros de la base de datos.
- **Delete:** eliminar registros de la base de datos.

Estas políticas se configuran por medio de tres parámetros, el primero se pasa directamente en la URL mientras que los otros dos se pasan en el cuerpo

- **Id de la política:** una cadena de texto de hasta 64 caracteres, servirá como clave primaria dentro de la base de datos por lo que no puede haber políticas con el mismo identificador.
- **Formula:** es la ecuación para el cálculo del nivel en base a la puntuación. Está contenida dentro de una cadena de texto y debe estar en formato Python.
- **Número máximo de niveles:** permite configurar el número máximo de niveles que va a aceptar la política.

Para realizar la evaluación de forma dinámica de forma sencilla se ha decidido utilizar el método eval que provee Python. Con este método lo que conseguimos es evaluar cadenas de texto como si de código se tratase, lanzando una excepción al detectar errores. Esta evaluación depende de contexto, es decir, puede coger el valor de las variables declaradas, módulos importados o métodos dentro del propio modulo.

Ya que el método eval puede llegar a suponer una brecha de seguridad inmensa, podría, con solamente dos mandatos, eliminar todo el disco duro, por ejemplo. Se ha decidido implementar un validador de fórmulas restrictivo.

Este validador de fórmulas está configurado para solo admitir:

- **Módulo math:** métodos de dentro del módulo math de Python,
- **Valores enteros y decimales**
- **Operadores Aritméticos**
- **Identificador 'level':** el cuál sería la única variable a la que puede acceder, será propia del módulo e ira aumentando con cada nivel de karma.

Esta validación se realiza previo a inserción y previo a modificación, además de previo a utilización, con ello lo que conseguimos es evitar cualquier posibilidad de cargar código maligno.

Selección de la Imagen a Mostrar

Para la selección de la imagen a mostrar se ha decidido implementar y diseñar un algoritmo de tipo EFES (Evaluación-Filtrado-Eliminado-Selección). La característica principal de estos algoritmos es que permiten realizar selecciones sobre listas aplicado distintos criterios para lograr mejores resultados.

Este algoritmo fue diseñado pensando en la modularidad, donde se pueden intercambiar fases, siempre y cuando sean del mismo tipo, u omitirse en el caso de que alguna de las fases no fuese necesaria. Un esquema de cómo funcionan estos algoritmos puede verse en la Ecuación 3.

ImagenAMostrar(nivel)

= selection(eliminacion(filtrado(nivel, evaluacion(observaciones))))

Ecuación 3: Pseudocódigo de un EFES

Para este caso, se ha decidido diseñar el EFES que se puede apreciar en la Ilustración 19.



Ilustración 19: Proceso de selección de una imagen

Generado con la Aplicación Draw.io

1. **Evaluación, Cálculo de la Dificultad:** en esta primera fase se evaluarán las imágenes según una serie de parámetros configurables.
2. **Filtrado, Clasificación según la Dificultad:** en este método, y utilizando el valor obtenido de la fase anterior, clasificaremos todas las imágenes en distintos niveles. Estos niveles servirán para que los usuarios puedan acceder de forma rápida a ellos, ya que estas clasificaciones estarán cacheadas y no tendrán que ser calculados cada vez que un usuario requiera una nueva imagen. Estas dos fases van a ser ampliadas a continuación.
3. **Eliminado, Descartar Repeticiones:** con las imágenes seleccionadas por nivel, será necesario eliminar de esta lista las imágenes que ya hayan sido procesadas.

4. **Selección, Selección Pseudoaleatoria:** Para este último paso recibiremos como entrada las imágenes filtradas y no repetidas y, utilizando algún método para calcular valores pseudoaleatorios, se seleccionará una imagen y se mandará al usuario.

Al dividir este proceso en distintas fases lo que conseguimos es incrementar la modularidad deseada además de favorecer la testabilidad. Además de poder utilizar las distintas fases de forma independiente y decidir cuál es la más adecuada. Estas fases están pensadas como métodos que reciben entradas y producen salidas.

Cálculo de la Dificultad de la Imagen

En este primer paso del algoritmo se evaluarán las observaciones, este valor se calcula desde dos partes, una parte estática, que no varía, y depende de las características de la imagen y de la observación, y una parte dinámica que se calcula a partir del número de votaciones que tenga una imagen. Cuantas más votaciones tenga una observación, más fácil será.

$$Dificultad = ParteEstatica - ParteDinamica$$

Ecuación 4: Cálculo de la Dificultad de una imagen

- **Parte Estática:** se calcula utilizando la Ecuación 5.

$$ParteEstatica = \frac{1}{3}V_{probabilidad} + \frac{1}{3}V_{seeing} + \frac{1}{3}V_{brillo}$$

Ecuación 5: Cálculo de la parte estática de la Dificultad

- **Probabilidad:** este valor, como ya se analizó anteriormente, depende de la latitud eclíptica del campo. Este valor se calcula como se puede ver en la Ecuación 6

$$V_{probabilidad} = 100 - probabilidad$$

Ecuación 6: Cálculo del Valor de la Probabilidad

- **Seeing**²⁹: turbulencia que posee la imagen. El seeing se obtiene desde los metadatos de la imagen, tras esto se pasa a calcular su valor utilizando la Ecuación 7.

$$V_{seeing} = 100 * \frac{1}{1 + e^{-fwhm+1}}$$

Ecuación 7: Cálculo del Valor del Seeing

- **Brillo**: valor obtenido de la observación, se calcula a partir del brillo que tiene la imagen en el punto de la observación. La parte de este valor se calcula de la siguiente manera.

$$V_{brillo} = 100 * \frac{1}{1 + e^{-0.6*(brillo-15)}}$$

Ecuación 8: Cálculo del Valor del Brillo

Parte Dinámica: Se calcula utilizando la Ecuación 9. Que consiste en simplemente multiplicar el número de votos que tenga la observación por un multiplicador en función del estado.

$$ParteDinamica = \#votos * multiplicador$$

Ecuación 9: Cálculo de la parte dinámica

Este nivel de dificultad servirá para filtrar las imágenes que le pueden aparecer a los usuarios. Con esto lo que conseguiremos es mostrar a los usuarios inexpertos (con poco nivel de karma) imágenes que sean consideradas como fáciles. Mientras que a los usuarios expertos les enseñaremos imágenes de mayor dificultad.

Filtrado de Imágenes por nivel

En esta configuración se ha decidido dividir la fase del filtrado en 3 distintas.

1. **Detección del nivel de filtrado:** primero dividiremos todos los niveles, en 5 grupos distintos de niveles. Esta división se hará de forma equitativa. Como puede verse puede ver en la Ilustración 20.

²⁹ <http://www.astronomyhints.com/seeing.html>

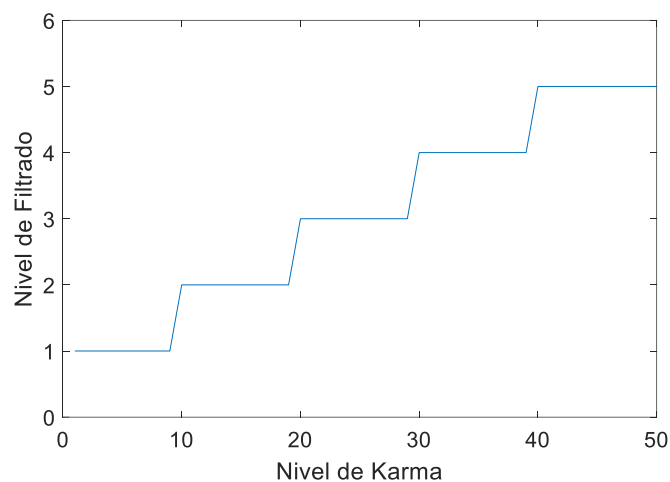


Ilustración 20: Nivel de Filtrado Según el Nivel de Karma

Imagen obtenida de Matlab

2. **Distribuir observaciones por dificultad:** la siguiente fase de este algoritmo será dividir todas las observaciones en 3 subgrupos según la dificultad.
 - **Fáciles:** tendrán todas las imágenes aprobadas
 - **Medias:** tendrán las observaciones pendientes que tengan un nivel de dificultad inferior al 50%.
 - **Difíciles:** tendrán todas las imágenes con un nivel de dificultad superior al 50% y todas las observaciones que se encuentren en estado Disputado.
3. **Selección del subgrupo a filtrar:** por ultimo devolveremos las observaciones asociadas al nivel de filtrado del usuario. Esta asociación se realiza como se puede ver en la Ilustración 21.

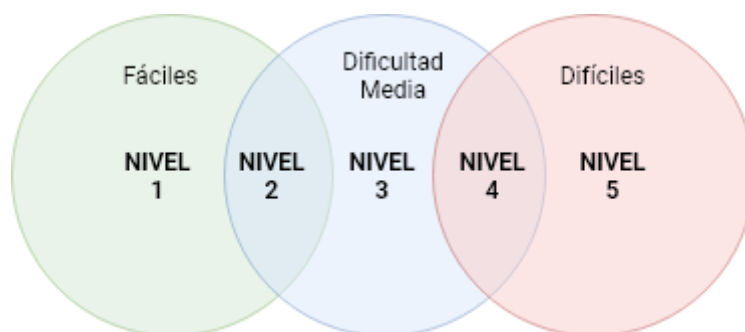


Ilustración 21: División Observaciones por Nivel de Filtrado

Generado con la Aplicación Draw.io

Validación de la Observación

Este algoritmo va a ser utilizado para determinar cuándo una imagen debe cambiar del estado "En votación" (*Pending* dentro del sistema) al estado "En comprobación" (*Accepted*, dentro del sistema). Actualmente, como está pensado el sistema, es que un astrónomo debe confirmar que la observación pueda ser correcta para enviarla al MPC (Minor Planet Center) el encargado de finalmente comprobar si la observación es correcta.

Para liberar la carga humana en la mayor parte posible, se ha decidido plantear un algoritmo que, en función de las votaciones, cambie el estado de las observaciones avisando al astrónomo de cuándo debe comprobarla él.

Además, utilizaremos el sistema de karma anteriormente planteado para ponderar las votaciones en función de la experiencia del usuario que ha votado. Las detenciones realizadas por otros usuarios en el mismo lugar equivaldrían a votaciones positivas.

Además, ya que vamos a tener votaciones positivas y negativas, este algoritmo debería ser capaz también de rechazar las observaciones.

Para cambiar el estado de la observación, se va a necesitar imponer un mínimo de votaciones, ya que, si no introduyésemos ningún mínimo, el sistema cambiaría este estado con la primera votación.

Por último, en las observaciones con un alto nivel de votaciones y donde no se pueda sacar un resultado claro, el sistema cambiara el estado de la observación a "Disputado" (*Disputed*, dentro del sistema) para que aparezca a todos los usuarios con un alto nivel de Karma.

Diagrama

Por lo tanto, aplicando todas las condiciones anteriormente descritas, se ha realizado un diagrama mostrando la máquina de estados en la que está basado este sistema. Este diagrama se puede apreciar en la Ilustración 22.

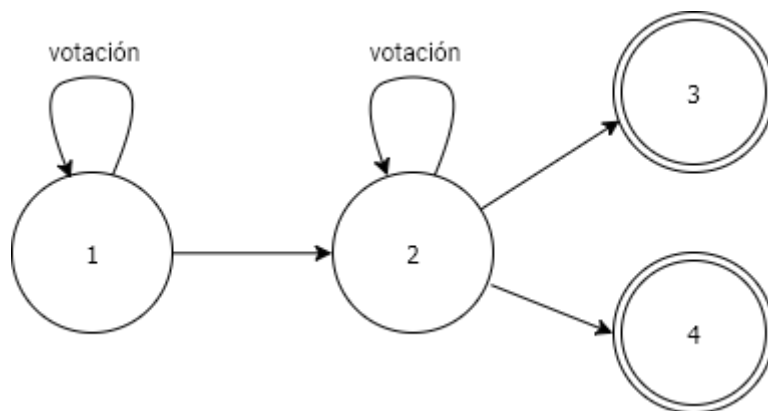


Ilustración 22: Máquina de Estados, Algoritmo de Validación

Generado con la aplicación Draw.io

Parámetros y constantes

Este algoritmo dependerá de distintas variables y constantes, las cuales o estarán configuradas en el servidor o serán obtenidas como parámetro de entrada. Estos valores son:

Variables

- **Certeza:** este valor variará entre -1 y 1 donde, cuando más cerca esté este valor de -1 mayor seguridad habrá de que esa observación no va a ser válida. Mientras, cuanto más cercano sea este valor del 1, mayor seguridad tendremos de que esa observación va a ser válida. Por último, los valores cercanos al 0 no tendremos ninguna certeza.
- **LímiteCerteza:** comenzara en un valor fijo y pasado un umbral de votaciones, en nuestro caso determinado por el número mínimo de votaciones comenzara a descender con cada votación hasta un valor fijado.

Constantes

- **NúmeroMínimo:** será el número mínimo de votaciones que se van a tener que realizar para que se pueda cambiar el estado.
- **NúmeroMáximo:** será el número máximo de votaciones que se van a realizar, en cuanto se alcance esta cifra se actualizará el estado de la observación a disputado.
- **LímiteInicial:** será el valor de certeza con el que se empezará.

- **LímiteFinal**: será el valor de certeza mínimo al que podrá descender.
- **Parámetros de entrada**
 - **Observación**: se pasará la imagen sobre la que se ha realizado la observación. Dentro de este objeto se van a utilizar los siguientes campos.
 - **NúmeroVotaciones**
 - **PuntuaciónNegativa**
 - **PuntuaciónPositiva**
 - **Karma Usuario**: el nivel del karma del usuario que ha realizado la votación.
 - **Tipo De Voto**: puede ser Positivo o Negativo.

Movimientos

Estos movimientos dependen de constantes y variables (en cursiva) que hemos visto anteriormente. Todos estos movimientos tendrán unos pasos comunes, y otros específicos según el tipo de movimiento.

1. Se incrementará el *NúmeroDeVotaciones*.
2. Se aumentará la *PuntuaciónPositiva* o *PuntuaciónNegativa* según el *KarmaUsuario* del usuario.
3. Se calculará la nueva *Certeza* utilizando la Ecuación 4.

$$Certeza = \frac{PuntuaciónPositiva - PuntuaciónNegativa}{PuntuaciónPositiva + PuntuaciónNegativa}$$

Ecuación 10: Cálculo de la Certeza de una Observación

4. Con este valor de *Certeza* se comprobará que este dentro de los límites configurados tras su previo cálculo utilizando la Ecuación 11.

LimiteCerteza(#votaciones) =

$$\begin{cases} \begin{matrix} \text{LímiteInical} \\ \text{LimiteInicial} - \frac{\text{LimiteInicial} * \#votaciones}{\text{LimiteFinal} * \text{NumeroMáximo}} \\ \text{LímiteFinal} \end{matrix} & \begin{matrix} \text{sii } votaciones < \text{NúmeroMinimo} \\ \text{Resto} \\ \text{sii } votaciones > 3 * \text{NúmeroMáximo} \end{matrix} \end{cases}$$

Ecuación 11: Cálculo del Límite de Certeza

5. Por último, y tras realizar los pasos específicos se actualizará en la base de datos todos los cambios.

Los movimientos específicos según la transición son los siguientes:

- **1:1:** este movimiento se va a producir siempre que se reciba una votación y el *NúmeroDeVotaciones* sea menor de *NúmeroMínimo*.
- **1:2:** este movimiento se va a producir cuando se reciba una votación y el *NúmeroDeVotaciones* sea *NúmeroMínimo*.
- **2:3:** en este movimiento el cual se puede producir por dos razones, se actualizará el estado de la observación a "Aprobada" cuando la *Certeza* sea mayor que el *LímiteSuperior*.
- **2:4:** este movimiento se producirá cuando la *Certeza* sea menor del *LímiteInferior*. En este estado se actualizará el estado de la observación a "Rechazada"
- **2:2:** este movimiento se va a producir siempre que no se pueda realizar ninguna de las 2 anteriores transiciones.

Testing e Integración Continua

Se ha desarrollado un módulo para el testeo de todo el sistema. El objetivo de este módulo es de poder realizar una gran cantidad de pruebas de forma automática con el fin de poder comprobar con cada actualización que el sistema sigue siendo estable.

Actualmente este módulo está dividido en 3. Uno para cada algoritmo, dentro de estos módulos se van implementado una serie de Tests individuales. Para una mayor automatización, se ha realizado un script que detecta todos los tests dentro de este módulo y los va lanzando uno a uno.

Este script puede ser llamado desde el manage del servidor. Actualmente requiere que el servidor haya sido ejecutado en otra consola del mismo equipo.

Los comandos para lanzar este script serían los siguientes.

```
python manage.py runtests
```

Los resultados esperados, se podrían ver en la Ilustración 23 resultado erróneo, mientras que en la Ilustración 24 tendríamos un resultado correcto.

```
#####
##### Starting Test Module #####
#####
[INFO] Starting ValidationTests:
- CreateRandomObservationTest: OK
- DoubleUserObservationTest: ERROR Expected error 400 received error 404
- MultipleVotesInSameObservationTest: OK
- NoObservationTest: ERROR Expected error 400 received error 405
- StressValidationTest: OK (Average request time 111963 ns)
[ERROR] 3/5 completed successfully
```

Ilustración 23: Resultado Test Script con Errores
Obtenido de la Aplicación cmdr

```
#####
##### Starting Test Module #####
#####
[INFO] Starting ValidationTests:
- CreateRandomObservationTest: OK
- DoubleUserObservationTest: OK
- MultipleVotesInSameObservationTest: OK
- NoObservationTest: OK
- StressValidationTest: OK (Average request time 98188 ns)
[SUCCESS] All test were completed successfully (5/5)
```

Ilustración 24: Resultado Test Script Correcto
Obtenido de la Aplicación cmdr

Para la creación de nuevos tests, gracias a la lectura dinámica utilizando el módulo *inspect* de Python. Simplemente habría que crear una nueva clase dentro de alguno de los modules de testeo que se encuentran en `/testers/*_tester` que fuese hija de `TestAbstract` la cual se encuentra dentro del módulo `testers/test`, esta clase abstracta

obliga a tener configurados los métodos `run` y `get_result`. Esta clase la podemos ver en la Ilustración 25.

```
1  ''' Test module '''
2
3  class TestAbstract:
4      ''' Test class '''
5      def run(self):
6          ''' Runs the test class '''
7          raise NotImplementedError('Abstract method')
8
9      def get_result(self):
10         ''' Prints the result of the test class '''
11         raise NotImplementedError('Abstract method')
12
```

*Ilustración 25: Clase TestAbstract
Obtenida de la Aplicación Visual Studio Code*

Por último, se va a implementar este módulo de tests con una plataforma de integración continua y el repositorio de control de versiones con el que estoy trabajando. Para así realizar un análisis del código que acabo de ser subido y compruebe su correcto funcionamiento.

Script de Preparación de Entorno

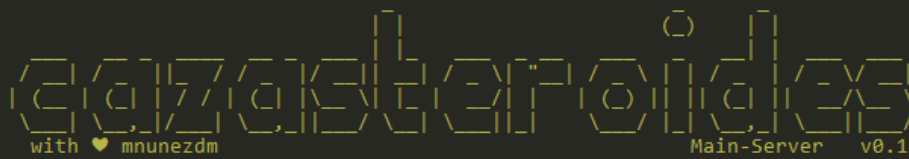
Por último, se ha preparado un script en Bash para la preparación del entorno donde se va a alojar el servidor general de cazasteroides.

Este script permite instalar todas las dependencias necesarias con control de errores y detectando las ya instaladas.

Para la utilización de este script el único prerequisite será tener un maquina con alguna distribución de Ubuntu/Debian, tener permisos de superusuario y que el script tenga permisos de ejecución.

Este script ha sido probado en una máquina virtual con Ubuntu Server 17.04 instalado. Aunque debería funcionar en cualquier entorno.

Un ejemplo del resultado obtenido al lanzar este Script y terminado de forma correcta se puede apreciar en la Ilustración 26.



```
[INFO]: Updating apt-get OK!
[INFO]: Installing necessary programs:
- Installing 'wget' ... OK!
- Installing 'pkg-config' ... OK!
- Installing 'mongodb' ... OK!
- Installing 'gcc' ... OK!
- Installing 'make' ... OK!
- Installing 'g++' ... OK!
- Installing 'cmake' ... OK!
- Installing 'git' ... OK!
- Installing 'subversion' ... OK!
- Installing 'npm' ... OK!
- Installing 'postgis' ... OK!
- Installing 'postgresql' ... OK!
- Installing 'postgresql-contrib' ... OK!
[INFO]: Installing necessary libraries:
- Installing 'libpng12-dev' ... OK!
- Installing 'libfreetype6-dev' ... OK!
- Installing 'libjpeg-dev' ... OK!
- Installing 'libproj-dev' ... OK!
- Installing 'libgeos-dev' ... OK!
- Installing 'libfcgi-dev' ... OK!
- Installing 'libcairo2-dev' ... OK!
- Installing 'libxml2-dev' ... OK!
- Installing 'libgif-dev' ... OK!
- Installing 'libgdal-dev' ... OK!
- Installing 'libapache2-mod-fcgi' ... OK!
[INFO]: Installing FITS ... downloading ... configuring ... making binaries ... installing ... OK!
[INFO]: Installing MapServer ... downloading ... cmake ... making binaries ... installing ... OK!
[INFO]: Configuring Apache cgi ... OK!
[INFO]: Configuring NPM ... OK!
[INFO]: Preparing workspace ... moving to correct branch ... installing dependencies ... OK!
```

Ilustración 26: Resultado Script Preparación Entorno

Obtenido de la Aplicación cmdr

Conclusiones

En la actualidad, donde cada vez es más complicado captar la atención del consumidor y más aún mantenerla debido a la gran cantidad de oferta fácilmente accesible desde cualquier dispositivo en cualquier momento o lugar; la gamificación se podría considerar como la tendencia más importante a seguir en los próximos años para lograr este objetivo.

La metodología desarrollada en este trabajo de fin de grado, aunque está centrado concretamente en el nicho de mercado de entender el universo que nos rodea; podría servir para enamorar a cualquier otro grupo de interés, porque combina las ansias de superación y competición con las ganas de profundizar en un tema específico.

Lo que se ha buscado con este módulo es ir enganchando al usuario de forma gradual, consiguiendo una curva de dificultad exponencial ligada a la práctica. Busca enviciar al usuario al ver claramente recompensados sus logros según va aumentando su experiencia, esto se percibe no como un simple marcador, sino también en los retos presentados y en la influencia de sus acciones.

Durante la etapa de análisis se han identificado un gran número de mejoras que podrían ser implementadas, dentro del sistema, en futuras fases del proyecto y que no han podido ser abordados en este trabajo, destacables serían las mejoras en la interfaz gráfica indispensables para una aplicación hoy en día; o las ampliaciones en el sistema de gamificación del sistema.

El efecto positivo de los algoritmos en el rendimiento de la solución es previsible ya que no ha podido ser validado con casos de uso reales. Las pruebas han sido únicamente realizadas en entornos controlados donde se ha mostrado un alto grado de fiabilidad. Las evidencias de estas pruebas no forman parte de los datos presentados.

Se ha buscado en todo momento la profesionalidad en el código desarrollado utilizando guías de diseño con alta aceptación en entornos productivos. La solución es fácilmente mantenible, parametrizable y escalable en funcionalidad. Se han incorporado técnicas de testeo automático, control de versiones y gestión de errores con el objetivo de conseguir un sistema fiable para el uso real del mismo.

Anexos

1. Instalación y Primer Arranque del Servidor de Karma

Prerrequisitos

Sera necesario tener Python3 instalado. No se asegura el funcionamiento sobre Python2 ya que se utilizan funciones específicas de Python como es el formateado de textos utilizando la función f-strings³⁰. Para la instalar se puede realizar directamente desde la página web de Python³¹.

Además, será necesario tener instalado pip³² para descargar todos los módulos necesarios, este paquete se instala por defecto al instalar cualquier versión de Python por encima de 2.7.8 y por encima de 3.3 (no inclusive).

Por último, es recomendable utilizar virtualenv³³ para aislar las dependencias de otros desarrollos.

Instalación

Será necesario descargarse la última versión del servidor desde este repositorio Git, para ello lo podemos hacer utilizando el siguiente mandato.

```
git clone https://github.com/mnunezdm/cazasteroides.git
```

Con esto activado habrá que navegar hasta la carpeta donde está el servidor, esta es:

```
cd cazasteroides/código/karma-server
```

Ahora, si se quiere utilizar un virtualenv habrá que inicializarlo y activarlo.

```
virtualenv .  
Scripts/activate
```

Ahora, habrá que instalar todos los módulos necesarios, para ello, simplemente lanzaremos el mandato

³⁰ <https://www.python.org/dev/peps/pep-0498/>

³¹ <https://www.python.org/downloads/>

³² <https://pypi.python.org/pypi/pip>

³³ <https://virtualenv.pypa.io/en/stable/>

```
pip install -r requirements.txt
```

Por último, tendremos que inicializar la base de datos. Para ello, dentro de la carpeta src, lanzaremos los tres siguientes mandatos.

```
python manage.py db init
```

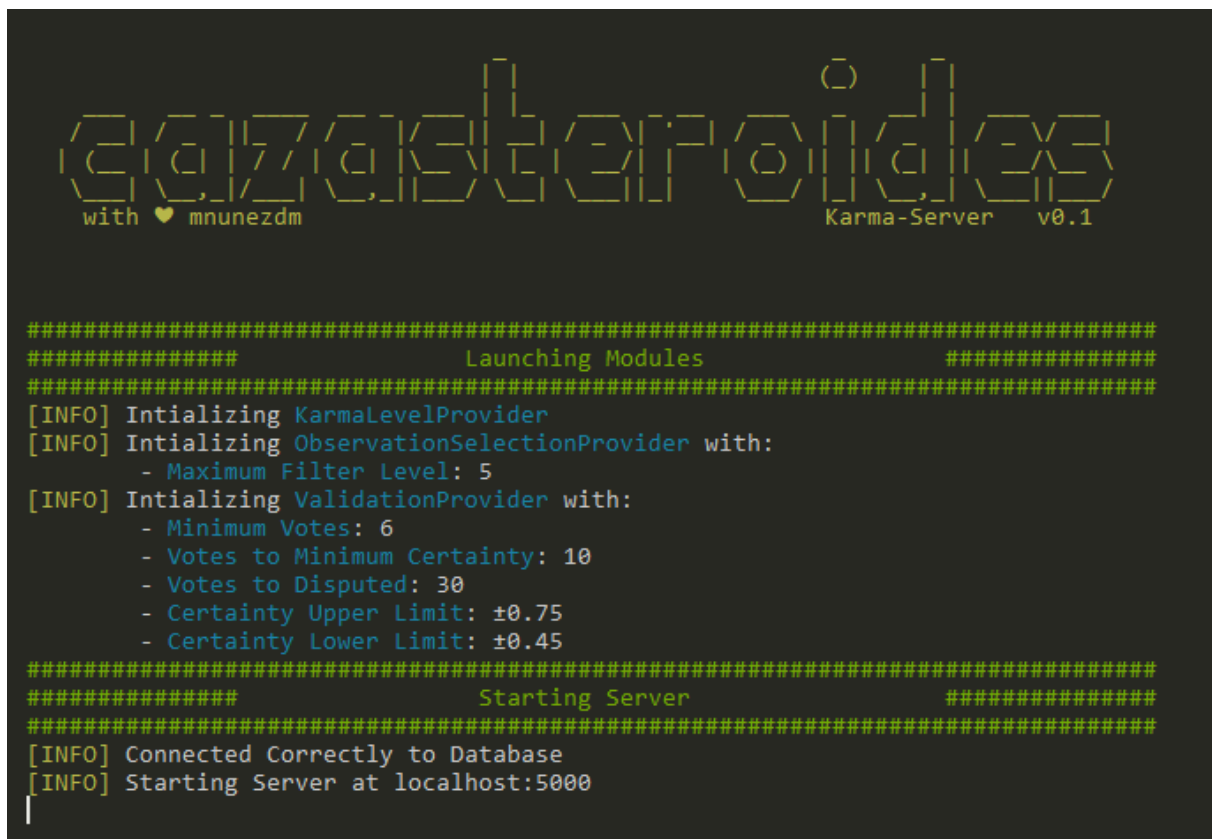
```
python manage.py db migrate
```

```
python manage.py db upgrade
```

Con esto ya tendremos nuestro sistema preparado para poder ejecutar el servidor y podremos ejecutar nuestro servidor con el siguiente mandato.

```
python manage.py runserver
```

La salida esperada, en el caso de todo hubiese salido correctamente, sería la mostrada en la Ilustración 27.



```
#####
#####                               Launching Modules                               #####
#####
[INFO] Intializing KarmaLevelProvider
[INFO] Intializing ObservationSelectionProvider with:
- Maximum Filter Level: 5
[INFO] Intializing ValidationProvider with:
- Minimum Votes: 6
- Votes to Minimum Certainty: 10
- Votes to Disputed: 30
- Certainty Upper Limit: ±0.75
- Certainty Lower Limit: ±0.45
#####
#####                               Starting Server                               #####
#####
[INFO] Connected Correctly to Database
[INFO] Starting Server at localhost:5000
|
```

*Ilustración 27: Salida Esperada al lanzar el servidor
Obtenido de la aplicación cmdr*

2. Uso del Servidor

Formato de Respuesta

Todas las respuestas tienen el mismo formato, se trata de un JSON con cuatro atributos.

1. **Code:** se trata del código HTTP que devuelve el servidor.
2. **Status:** se trata de un mensaje de estado asociado con el código HTTP
3. **Description:** se trata de un mensaje descriptivo que trae información adicional, por ejemplo, porque ha fallado la petición.
4. **Payload:** datos devueltos en la respuesta, puede volver vacío.

Cálculo del Karma

Tendremos 5 métodos distintos en este módulo. Este módulo está levantado sobre el prefijo /level por lo que todas las URL deben contener este prefijo. A continuación, se detallan cuáles son los métodos y su funcionalidad.

Calcular Nivel

Endpoint	máquina:puerto/level/<política>/<punto>
Método HTTP	GET
Descripción	Calcula el nivel de karma asociado a la política y a los puntos pasados.
Parámetros	<ul style="list-style-type: none">• Política (String): indica el id de la política• Puntos (Integer): con los puntos del usuario
Payload	<pre>{ "level": 21, "points_to_next": 400, "policy": "1" }</pre>

Crear Nueva Política

Endpoint	máquina:puerto/level/<política>
Método HTTP	POST
Descripción	Crea una nueva política en función de los campos pasados
Parámetros	Política (String): indica el id de la política
Petición	<pre>{ "formula": "math.log ((level / 3) + 1) * 350", "max_level": 25 }</pre>

Obtener Información de una Política

Endpoint	máquina:puerto/level/<política>
Método HTTP	GET
Descripción	Devuelve información sobre la política:
Parámetros	Política (String): indica el id de la política
Payload	<pre>{ "formula": "math.log ((level / 3) + 1) * 350", "levels": [{ "level": 1, "necessary_points": 0 }, [...] { "level": 25 }], "max_level": 25, "policy": "1" }</pre>

Actualizar Política

Endpoint	máquina:puerto/level/<política>
Método HTTP	PUT
Descripción	Actualiza uno o dos de los campos
Parámetros	Política (String): indica el id de la política
Petición	<pre>{ "formula": "math.log ((level / 3) + 1) * 350", "max_level": 25 }</pre>

Eliminar Política

Endpoint	máquina:puerto/level/<política>
Método HTTP	DELETE
Descripción	Elimina la política
Parámetros	Política (String): indica el id de la política

Selección de la Observación a Mostrar

Tendremos 2 métodos distintos en este módulo, aunque debido a que los dos tienen las mismas entradas y salidas se han decidido resumir en uno. Este módulo está levantado sobre el prefijo /selection por lo que todas las URL deben contener este prefijo. A continuación, se detallan cuáles son los métodos y su funcionalidad.

Obtener nueva observación para Votación o Descubrimiento

Endpoint	máquina:puerto/selection/(vote ó /discover) ?user=mnunezdm&karma=100
Método HTTP	GET
Descripción	Devuelve la observación recomendada para el usuario pasado, en función de si es voto o descubrimiento
Query String	User (String): id del usuario Karma(Integer): nivel de karma del usuario
Payload	<pre>{ "filter_level": 5, "image_id": "2", "observation_id": "test-1498992170.559663", "position": { "x": 12, "y": 15 } }</pre>

Validación de la Observación

Tendremos 2 métodos distintos en este módulo. Este módulo está levantado sobre el prefijo /validation por lo que todas las URL deben contener este prefijo. A continuación, se detallan cuáles son los métodos y su funcionalidad.

Añadir Voto

Endpoint	máquina:puerto/validation/vote
Método HTTP	POST
Descripción	Añade el voto pasado a la observación pasada según el karma
Cuerpo Petición	<pre>{ "user_info": { "_id": "prueba7" }, "vote_info": { "karma_level": 3, "vote_type": true }, "observation_info": { "_id": "prueba11", "brightness": 5, "image": { "_id": "prueba11", "fwhm": 1, "probability": 60, "x_size": 100, "y_size": 100 }, "votes": { "upvotes": 0, "downvotes": 0 }, "position": { "x": 12, "y": 15 } } }</pre>
Payload	<pre>{ "_id": "1", "image_id": "-5", "position": { "x": 12, "y": 15 }, "punctuation": { "certainty": 0.8, "negative": 113, "positive": 1318 }, "state": "approved", "users_who_voted": ["33", [...], "329"], "votes": { "downvotes": 4, "upvotes": 16 } }</pre>

Obtener información sobre una Observación

Endpoint	máquina:puerto/validation/<observation_id>
Método HTTP	GET
Descripción	Devuelve toda la información de la observación
Parámetros	Observation_id (String): id de la observación
Payload	<pre>{ "_id": "1", "image_id": "-5", "position": { "x": 12, "y": 15 }, "punctuation": { "certainty": 0.8 "negative": 113, "positive": 1318 }, "state": "approved", "users_who_voted": ["33", [...] "329"], "votes": { "downvotes": 4, "upvotes": 16 } }</pre>

3. Principios S.O.L.I.D.

El termino SOLID fue por primera vez descrito por George C Martin a principios del siglo XXI con el fin de mejorar la legibilidad, mantenibilidad y escalabilidad del código. Para ello decidió utilizar el mnemónico S.O.L.I.D. el cual se compone de los siguientes 5 principios:

1. Single Responsibility Principle - Principio de Responsabilidad Única

Este principio considera que las clases y métodos solo deberían tener una única funcionalidad. Es decir, una clase debería solo hacer una tarea, y no intentar aglutinar una gran cantidad de funcionalidades, al igual que los métodos.

2. Open/Closed Principle – Principio de Abierto/Cerrado

Este principio considera que las clases deberían estar cerradas para modificación, pero abiertas para extensión, es decir, si se quiere añadir una funcionalidad no se debería editar lo que actualmente está funcionando

3. Liskov Substitution Principle – Principio de Sustitución de Liskov

Este principio considera que los subtipos de una clase, deberían poder ser substituidos por su clase padre sin que esta fallase.

4. Interface Segregation Principle – Principio de la Segregación de Interfaces

Este principio considera que es necesario dividir las interfaces para que tengan sentido único, más que tener una interfaz que implemente gran cantidad de métodos los cuales no van a ser implementados.

5. Dependency Inversion Principle – Principio de Inversión de Dependencia

Este principio considera dos patrones:

- Las abstracciones no deberían depender de detalles: esto quiere decir que una clase abstracta debería únicamente hacer referencia a otras clases no abstractas.
- Los detalles deberían depender únicamente de abstracciones: este segundo punto, explica que las implementaciones deberían referenciar a clases abstractas no a otras implementaciones.

4. Arquitectura del Servidor de Karma

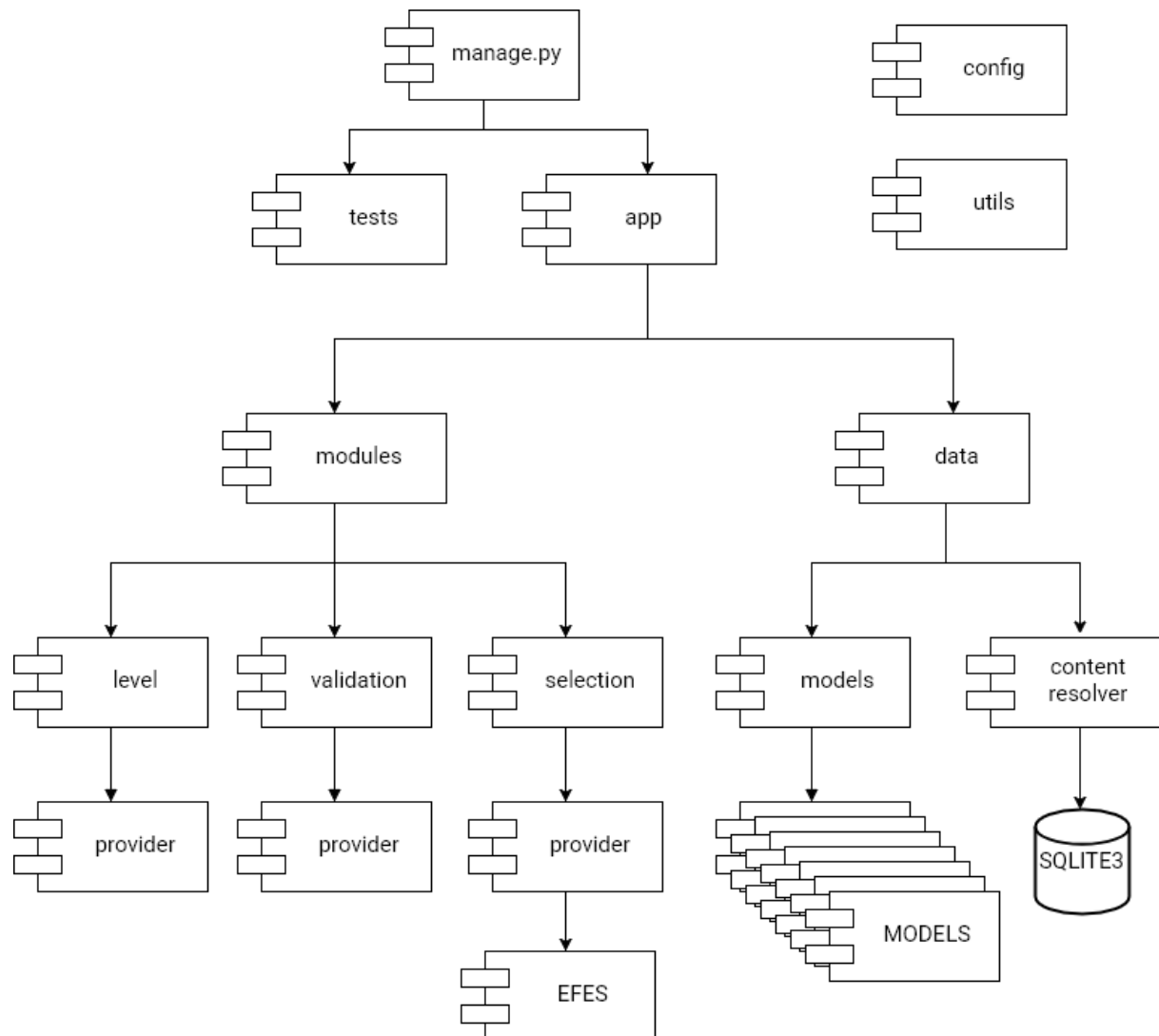


Ilustración 28: Arquitectura Servidor de Karma

Generado con la Aplicación Draw.io

Bibliografía

- [1] VON AHN, Luis, et al. recaptcha: Human-based character recognition via web security measures. Science, 2008, vol. 321, no 5895, p. 1465-1468.
· Disponible en https://www.cs.cmu.edu/~biglou/reCAPTCHA_Science.pdf
- [2] GEIGER, David; ROSEMAN, Michael; FIELT, Erwin. Crowdsourcing information systems: a systems theory perspective. En Proceedings of the 22nd Australasian Conference on Information Systems (ACIS 2011). 2011.
· Disponible en http://eprints.qut.edu.au/47466/1/acis_-_2011_-_Crowdsourcing_Information_Systems_-_A_Systems_Theory_Perspective.pdf
- [3] MOLL, Santiago. Gamificación: 7 claves para entender qué es y cómo funciona. 2014.
· Disponible en <http://justificaturespuesta.com/gamificacion-7-claves-para-entender-que-es-y-como-funciona/>
- [4] GOIKOLEA, Marcos. ¿Qué es la Gamificación? Definición y características. 2013.
· Disponible en <http://noticias.iberestudios.com/que-es-gamificacion/>
- [5] TARDO, Tardo. Everyone's a Gamer. 2014.
· Disponible en <http://www.prnewswire.com/news-releases/everyones-a-gamer---ieee-experts-predict-gaming-will-be-integrated-into-more-than-85-percent-of-daily-tasks-by-2020-247100431.html>
- [6] Entertainment Software Association. Essential-Facts. 2016.
· Disponible en <http://essentialfacts.theesa.com/Essential-Facts-2016.pdf>
- [7] JIMÉNEZ ARENAS, Sergio. Gamification Model Canvas. 2013.
· Disponible en http://www.gameonlab.com/downloads/gamification_model_canvas_poster.pdf
- [8] BOADA, Josep. Diferencia entre las mecánicas y dinámicas de los juegos en fidelización. 2013.
· Disponible en <https://jboadac.com/2013/03/05/diferencia-entre-las-mecanicas-y-dinamicas-de-los-juegos-en-fidelizacion/>
- [9] BARTLE, Richard A. Designing virtual worlds. New Riders, 2004.
· Disponible en <https://books.google.es/books?id=z3VP7MYKqalC&printsec=frontcover>
- [10] VON AHN, Luis; DABBISH, Laura. Labeling images with a computer game. En Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 2004. p. 319-326.
· Disponible en <https://www.cs.cmu.edu/~biglou/ESP.pdf>
- [11] FeverBee Limited, Reputation Systems Are Better Than Game Mechanics. 2011.
· Disponible en <https://www.feverbee.com/reputationsystems/>
- [12] MARTIN, Robert C. Clean code: a handbook of agile software craftsmanship. Pearson Education, 2009.
· Disponible en <https://www.casadellibro.com/libro-clean-code/9780132350884/1954618>