CloudMiner

Proyecto de Sistemas Informáticos

2013 – 2014

Facultad de Informática

Universidad Complutense de Madrid

Autores:

Juan Arratia Mamani

Arturo Pareja García

Tomás Restrepo Klinge

Director:

José Luis Vázquez Poletti

Autorización

Los abajo firmantes, matriculados en la asignatura Sistemas Informáticos de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado durante el curso académico 2013-2014.

Juan Arratia Mamani

Arturo Pareja García

Tomás Restrepo Klinge

Agradecimientos

Como grupo, queremos agradecer a nuestro tutor José Luis Vázquez Poletti, quien nos ha ayudado durante todo el proceso de diseño y guiado a lo largo del desarrollo del proyecto. Asimismo, a todos quienes de manera directa o indirecta, nos ayudaron y apoyaron para hacer CloudMiner una realidad.

A mi familia, quienes han sido siempre para mí un apoyo imprescindible. A mis compañeros por su ayuda y sus oportunos consejos. Y más que nunca, a Nancy, por estar ahí en los momentos más difíciles y ser siempre la mitad más importante del equipo.

Tomás

A mi novia, Marta, por llenar mi vida de ilusión y alegría, por todo el ánimo que me has dado y por no dejar que me rindiese nunca. A mi familia y a mis amigos por estar ahí cuando más se os necesita, por creer en mí y por guiarme siempre por la buena senda.

Arturo

A mi padres Genaro y Octavina, que en paz descansen, por su esfuerzo por sacarme adelante. A mis hermanos por sus consejos y apoyo que han sido imprescindible en esta batalla. A mi cuñado Paulino Aldana por su apoyo incondicional. A mis amigos y compañeros por su amistad y sus consejos. Y a todos lo que han hecho que esto sea una realidad. ¡Los quiero!

Juan

Resumen

CloudMiner es un proyecto cuyo objetivo principal es conseguir un mejor aprovechamiento del hardware existente, consiguiendo un beneficio económico mediante el uso de cripto-monedas virtuales. La idea principal es crear un "cloud" de recursos, compuesto por distintas máquinas con arquitecturas potencialmente distintas. Este "pool" será monitorizado en tiempo real por la aplicación, permitiendo al usuario comenzar/parar el "minado" en cualquier momento en cualquiera de las máquinas disponibles, asimismo dándole información respecto al estado actual de estas. Se pondrán a disposición opciones adicionales, tales como añadir o quitar recursos (bajo arquitecturas y SSOO soportados). Adicionalmente, se podrá usar inteligencia artificial basada en estadísticas, consiguiendo un cierto nivel de automatización de la aplicación. Estas estadísticas también se harán visibles al usuario, para ayudar en la toma de decisiones.

Palabras clave: Cloud, Bitcoin, Monedas virtuales, Minería

Abstract

CloudMiner is a project that aims for a better exploitation of the existing hardware, achieving economical benefit through the mining of virtual crypto-currencies. The main idea is to create a cloud-computing resource pool, composed by diverse machines under potentially different architectures. This pool will be monitorized in real-time by the application, enabling the user to start/stop mining at any given point on any of the available machines, also providing information on their current status. Additional options will be available, like adding or removing resources (supported architectures). Artificial Intelligence based on statistics may be used in order to allow automated control of the mining cloud. This statistics are also visible to the user, to aid decision taking.

Keywords: Cloud, Bitcoin, Virtual currencies, Mining

Índice

1. Introducción

1.1. Divisas electrónicas, BitCoin

1.1.1. Historia

El concepto de dinero electrónico aparece en respuesta a situaciones que se presentan en el ámbito de internet. Visto en perspectiva, y dado el rápido crecimiento que han tenido los intercambios virtuales en las últimas décadas, parece inevitable que tarde o temprano se popularizara este concepto.

Inicialmente, es en el mundo de los videojuegos donde se dan las primeras referencias a esto. Podemos tomar como ejemplo el caso de la red social-videojuego SecondLife, en donde se podía hacer uso del "Linden Dollar" de manera similar a una moneda real para adquirir diversidad de objetos, entre otros. (1)

A partir de entonces, empezaron a surgir juegos multijugador online donde los jugadores pueden acceder a objetos, misiones o mejoras mediante la compra de créditos.

Con el paso del tiempo, el uso de internet se ha ido masificando, y asimismo la utilización de medios virtuales de pago también se ha generalizado. El objetivo es entonces facilitar este tipo de transacciones, además de mejorar la seguridad relacionada a los intercambios en este ámbito. Comienzan entonces a ofrecerse servicios profesionales como PayPal, que sirven como monederos virtuales, útiles a la hora de realizar o recibir pagos relativos a transacciones realizadas por internet.

Se hace patente entonces un nuevo objetivo: el anonimato. Surgen en respuesta las divisas electrónicas, con la diferencia respecto a las "tangibles" que no existe ninguna entidad central que las controle.

Una de estas divisas, la que quizás más popularidad ha ganado, es el BitCoin. Para poder analizar con cierto detalle el funcionamiento de las divisas electrónicas, tomamos a partir de ahora como caso de estudio el BitCoin.

1.1.2. Origen del BitCoin

Bitcoin fue concebido en 2008 por una persona (o grupo de personas) bajo el seudónimo "Satoshi Nakamoto". La creación de la primera aplicación para operar con *Bitcoins* también se le(s) atribuye. Respecto a esto, existen dudas sobre su nacionalidad y hay múltiples especulaciones sobre su identidad real. (2)

Bajo este seudónimo se publica también un libro que propone un sistema de transacciones electrónicas que no depende de la confianza, sino que permite realizar transferencias de forma directa sin la necesidad de un intermediario. (3) Al contrario de la mayoría de las monedas, el Bitcoin no está respaldado por ningún gobierno ni depende de ningún emisor central, sino que utiliza un sistema de prueba de trabajo para impedir el doble gasto y alcanzar el consenso entre todos los nodos que integran la red.

Desde que se puso en funcionamiento en 2009 el sistema ha ido ganando popularidad gracias a las características de anonimato con las que permite realizar transacciones comerciales y al interés especulativo que ha despertado la evolución de su cotización al cambio con monedas reales.

1.1.3. Seguridad

La seguridad de la mayoría de las criptomonedas (y en particular de *Bitcoin*) se basa en utilizar técnicas de criptografía para la protección del saldo del usuario. La verificación de las solicitudes de transacción se realiza usando criptografía asimétrica.

La criptografía asimétrica (en inglés asymmetric key cryptography), también llamada criptografía de clave pública o criptografía de dos claves1, es el método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que ha enviado el mensaje. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. (4)

El sistema también asegura que las transacciones son correctas y que el poseedor de un saldo no puede gastarlo más de una vez. Para ello se utilizan técnicas de sellado temporal con las que se registra el momento exacto en el que se solicita una transacción de *bitcoins*. Los nodos de procesado tienen en cuenta estos valores para determinar si una transacción es válida o no.

1.1.4. Protocolo

Direcciones

Todo participante de la red Bitcoin tiene una cartera electrónica que contiene un número arbitrario de claves criptográficas. A partir de la clave pública, se obtiene la dirección Bitcoin, que funciona como la entidad remitente y receptora para todos los pagos. Su clave privada correspondiente autoriza el pago solo para ese usuario. Las direcciones no tienen ninguna información sobre su propietario, son generalmente anónimas y no requieren de ningún contacto con los nodos de la red para su generación.

Las direcciones son secuencias alfanuméricas aleatorias de 33 caracteres de largo, en formato legible para personas, como puede verse en este ejemplo: 1LtU9rMsQ41rCqsJAvMtw89TA5XT2dW7f9. Se utilizan codificación en Base58, que resulta de eliminar los siguientes seis caracteres del sistema Base64: 0 (cero), I (i mayúscula), O (o mayúscula), l (L minúscula), + (más) y / (barra). De esta forma, se componen únicamente de caracteres alfanuméricos que se distinguen entre sí en cualquier tipo de letra. Las direcciones Bitcoin también incluyen un checksum de 32 bits para detectar cambios accidentales en la secuencia de caracteres.

Transacciones

Cada BitCoin contienen la dirección pública de su dueño. Cuando un usuario A transfiere algo a un usuario B, A entrega la propiedad agregando la clave pública de B y después firmando con su clave privada. A entonces incluye esos bitcoins en una transacción, y la difunde a los nodos de la red P2P a los que está conectado. Estos nodos validan las firmas criptográficas y el valor de la transacción antes de aceptarla y retransmitirla. Este procedimiento propaga la transacción hasta alcanzar a todos los nodos de la red P2P.

Cadena de bloques

Todos los nodos que forman parte de la red Bitcoin mantienen una lista colectiva de todas las transacciones conocidas, a la que se denomina la cadena de bloques. Los nodos generadores, también llamados mineros, crean los nuevos bloques, añadiendo en cada uno de ellos el hash del último bloque de la cadena más larga de la que tienen conocimiento, así como las nuevas transacciones publicadas en la red.

Cuando un minero encuentra un nuevo bloque, lo transmite al resto de los nodos a los que está conectado. En el caso de que resulte un bloque válido, estos nodos lo agregan a la cadena y lo vuelven a retransmitir. Este proceso se repite indefinidamente hasta que el bloque ha alcanzado todos los nodos de la red. Eventualmente, la cadena de bloques contiene el historial de posesión de todas las monedas desde la dirección creadora a la dirección del actual dueño. Por lo tanto, si un usuario intenta reutilizar monedas que ya usó, la red rechazará la transacción.

Mining

La generación de bloques se conoce en inglés como mining y puede traducirse al español como 'extracción' por analogía con la minería del oro. Todos los nodos generadores de la red compiten para ser el primero en encontrar la solución al problema criptográfico de su bloque candidato actual, mediante un sistema de pruebas de trabajo, resolviendo un problema que requiere varios intentos repetitivos, por fuerza bruta, no determinista, de manera que se evita que mineros con gran nivel de procesamiento dejen fuera a los menos capaces.

De esta forma, la frecuencia de localización de cada bloque sigue una distribución de Poisson y la probabilidad de que un minero lo encuentre depende del poder computacional con el que contribuye a la red en relación al poder computacional de todos los nodos combinados, lo que permite que el sistema funcione de manera descentralizada. Los nodos que reciben el nuevo bloque solucionado lo validan antes de aceptarlo, agregándolo a la cadena. La validación de la solución proporcionada por el minero es trivial y se realiza inmediatamente.

La red reajusta la dificultad cada 2016 bloques, es decir, aproximadamente cada 2 semanas, para que un bloque sea generado cada diez minutos. La cantidad de Bitcoins creada por bloque nunca es más de 25 BTC, y los premios están programados para disminuir con el paso del tiempo hasta llegar a cero, garantizando que no puedan existir más de 21 millones de BTC.

Los mineros no tienen la obligación de incluir transacciones en los bloques que generan, por lo que los remitentes de Bitcoins pueden pagar voluntariamente una tarifa para que tramiten sus transacciones más rápidamente. Como el premio por bloque disminuye con el paso del tiempo, en el largo plazo todas las recompensas de los nodos generadores provendrán únicamente de las tarifas de transacción.

1.1.5. Rendimiento computacional de la minería

Las estrategias para la extracción de Bitcoins se han ido perfeccionando progresivamente. En los primeros meses de funcionamiento de la red era posible extraer en solitario con una CPU estándar y obtener un bloque y sus 50 BTC asociados con una frecuencia relativamente alta. Posteriormente, la aparición de software de minería adaptado a tarjetas gráficas, mucho más eficiente, desplazó completamente a las CPUs.

La minería por GPUs se ha ido profesionalizando, con grandes instalaciones en países con energía barata, configuraciones personalizadas con uso generalizado de overclocking y sistemas especiales de refrigeración. Con el aumento sostenido de la dificultad, los mineros comenzaron a organizarse en grupos independientes (en inglés, pools) para extraer de manera colectiva, desplazando así a los mineros en solitario que podían tardar meses o incluso años en encontrar un bloque de manera individual. El propietario del pool se lleva una comisión por encontrar un bloque. Los pools también compiten entre ellos para intentar atraer al mayor número de mineros.

Desde el año 2013 se han comenzado a comercializar FPGAs y ASICs para extraer Bitcoins de manera más eficiente. Si con la minería con CPUs y tarjetas gráficas, el coste de explotación provenía fundamentalmente del gasto energético, la comercialización de equipos especializados de bajo consumo está desplazando las inversiones de los mineros hacia hardware más sofisticado, e indirectamente hacia la investigación necesaria para el desarrollo de estos productos.

1.2. Planteamiento del problema

Teniendo en cuenta la situación actual en el ámbito de la "minería", hay que tener en cuenta que resulta sumamente ineficiente en la actualidad plantear la minería en solitario. Es más recomendable unirse a un "pool", según se ha explicado previamente. Evidentemente, entre más máquinas se dispongan a tal efecto, mayor el beneficio asociado.

El problema que subyace entonces, en el caso de disponer de más de una máquina con la que hacer minería, es el poderlas controlar y manejar de manera eficiente y cómoda.

1.3. Objetivos y alcance del proyecto

El objetivo es entonces ofrecer una plataforma de gestión, desde la cual se pueda controlar un número potencialmente alto de máquinas. No se pone límite tampoco en cuanto a diversidad de arquitecturas se refiere. La idea es facilitar el proceso de minería con cualquier dispositivo habilitado para tal fin.

Se pre-configura la herramienta con los casos más frecuentes en el equipamiento de las aulas de laboratorio: windows 7 de 32 bit y 64 bits, linux debian también en versiones de 32 y 64 bits. En particular, los equipos de más reciente adquisición resultarían más eficientes para este tipo de trabajos, dado el tipo de procesador que llevan.

Se pretende también hacer lo más transparente al usuario el funcionamiento del software existente (usados para minería), centrándose en el rendimiento que pueda obtenerse, basándose principalmente en estadísticas, además de otros factores como la cotización actual de las diversas cripto-monedas. Se busca acercar al usuario a este mundo gradualmente, aunque sin necesidad de adentrarse demasiado.

1.4. Situación actual de la Fdi-UCM

El principal aliciente de este planteamiento viene de observar el porcentaje de utilización de los recursos informáticos actuales de la facultad. A lo largo de cada curso académico podemos observar periodos de mayor/menor actividad en las aulas de laboratorios, por ejemplo. Igual situación nos encontramos en los ordenadores de la biblioteca.

Por otra parte, existen equipos de utilización menos frecuente, como por ejemplo los asignados al personal administrativo, cuya jornada es más reducida. Estos podrían ser aprovechados en multitud de situaciones, una de ellas es la que nosotros planteamos. Esta idea es extensible a centros informáticos de otras facultades/centros, teniendo así un margen de trabajo mucho más amplio.

1.4.1. Hardware existente - rentabilización

Tengamos en cuenta la cantidad de equipos ya en propiedad de la facultad (aulas de laboratorios, biblioteca, etc). El uso de este proyecto se puede considerar como una manera

de rentabilizar su adquisición, e incluso una manera de permitir la renovación de los mismos. Con que cada equipo este en posición de producir (minar) lo suficiente como para amortizar su precio de coste, ya sería suficiente.

1.4.2. Consumo energético

Es de notar la constante subida en el coste de la energía, especialmente en los últimos años. Los centros de proceso de datos más recientes se caracterizan por la búsqueda de la mayor eficiencia energética posible para así reducir al máximo los costes de explotación. Ante la actual situación de crisis económica, la Universidad Complutense no puede quedarse atrás en la mejora de la eficiencia energética: el ahorro energético permite reducir el efecto que las sucesivas reducciones del presupuesto tienen sobre otras partidas (como las dedicadas a becas y a investigación). Por ello, es fundamental racionalizar el consumo energético de las aulas de informática.

La primera fuente de consumo es el sistema de climatización y alumbrado de las aulas de informática infrautilizadas, cuyo uso resulta imprescindible con independencia del número de alumnos que estén trabajando en el aula. La segunda fuente de consumo es el propio equipamiento informático asignado a las aulas infrautilizadas: los PCs, servidores y también el equipamiento de red. Si se aprovechase al máximo el potencial de estos equipos en desuso, se podría llegar a conseguir mucha mayor eficiencia a nivel energético.

2. Estado del arte

2.1. Cloud Computing

(5)

2.1.1. Introducción

Cloud Computing es un modelo de computación relativamente reciente, en el cual se independizan los diferentes recursos que intervienen a la hora de utilizar una aplicación o sistema dado. EL usuario no necesita conocer en detalle sobre que recursos "reales" se está trabajando, interesándole únicamente los recursos virtuales.

Estos recursos son potencialmente de naturaleza distinta: capacidad de cómputo, espacio de almacenamiento, tráfico de red, etc. El objetivo es generalmente combinarlos para conformar una aplicación completa.

En este capítulo abordaremos los conceptos principales relacionados con "Cloud Computing", así como los diferentes modelos que hay.

2.1.2. Características

Las características generales que cumple una plataforma de este tipo son[1]:

- **Sevicio bajo demanda:** Un consumidor puede hacer uso de recursos computacionales, tales como tiempo de servidor y almacenamiento en red, sin requerir interacción con cada proveedor de servicios.

- **Acceso uniforme a la red:** Los servicios están disponibles en red y son accedidos usando mecanismos estándar que promueven el uso de plataformas heterogéneas (móviles, tablets, ordenadores portátiles y de escritorio, etc).

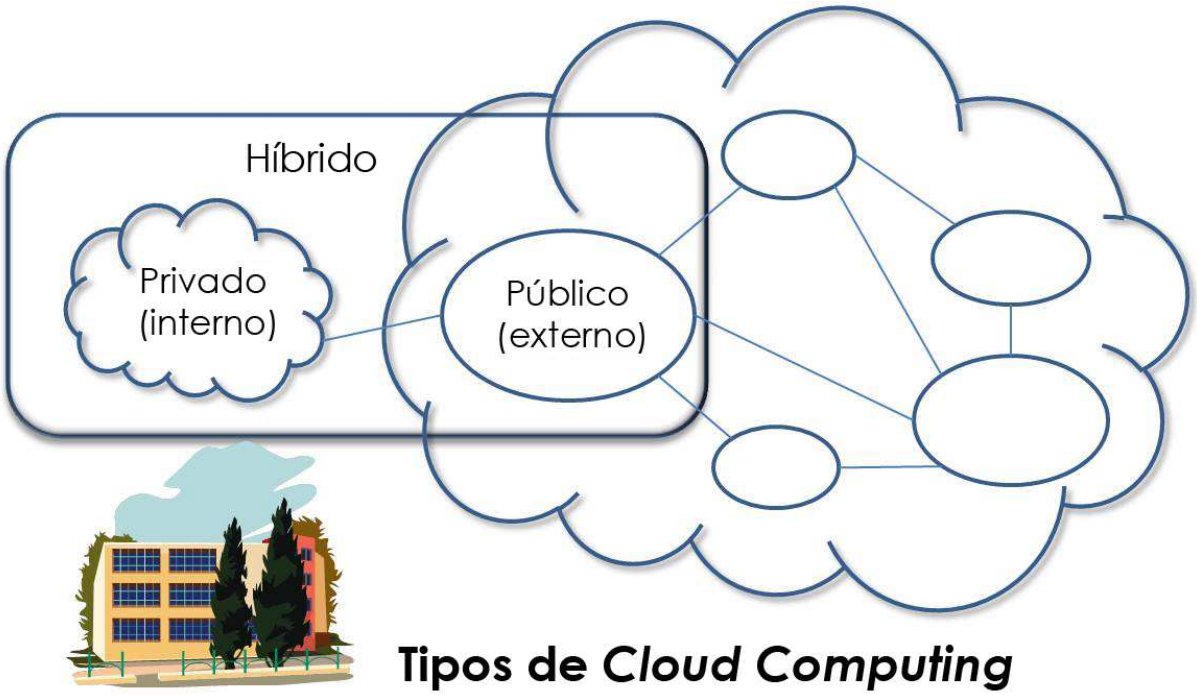
- **"Pooling" de recursos:** Los recursos computacionales del proveedor de servicios son provistos para atender múltiples consumidores usando un modelo multi-usuario, con diferentes recursos físicos y virtuales asignados dinámicamente, y reasignados basándose en la demanda del consumidor. Hay un sentido de independencia de ubicación en la que el el cliente generalmente no tiene ni conocimiento ni control sobre la ubicación exacta de los recursos provistos pero puede especificar su ubicación a un nivel más alto de abstracción (país, ciudad, centro de datos).

- **Elasticidad rápida:** Los servicios pueden ser provisionados y liberados de manera elástica, en algunos casos de manera automática, escalando de acuerdo con la demanda. Para el consumidor, los servicios disponibles usualmente aparentan ser ilimitados y pueden ser utilizados en cualquier medida y en cualquier momento.

- **Servicio a medida:** Los sistemas Cloud permiten el control y optimización de los recursos mediante unidades de medida con cierto nivel de abstracción relativo al tipo de servicio. Los recursos se pueden monitorizar, controlar, y reportar, proveyendo transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

[1] Dichas características han sido extraídas del proyecto de sistemas informáticos “Cygnuscloud” (30)

2.1.3. *Cloud* privado, *Cloud* público, otros



En función de la ubicación de los servidores y del público al que están destinados los servicios, podemos distinguir principalmente entre tres tipos de cloud: público, privado e híbrido.

Cloud privado

Cualquier centro de cómputo en propiedad de una organización privada puede considerarse como de este tipo, dado que es destinado a uso exclusivo de la misma. Las principales ventajas que ofrecen los clouds privados son el aislamiento y la exclusividad de su propia infraestructura, así como máximas garantías de seguridad y privacidad en sus sistemas de procesamiento de información. (5) (6) (7) (8)

Los proveedores de este tipo de servicio suelen tener por clientes a grandes empresas, las cuales a su vez requieren una mayor atención en aspectos como seguridad e integridad de datos. Por esto, pueden resultar un poco más costosos.

El uso de *clouds* privados resulta muy adecuado cuando:

- El negocio se basa en su información y sus aplicaciones. Por lo tanto, el control y la seguridad son primordiales.
- La organización es lo suficientemente grande como para poder mantener un sistema en la nube propio y eficiente.
- El negocio está relacionado con la gestión de la información de otras empresas. [6]

Cloud público

En el lado opuesto tenemos el modelo privado, que es a su vez el más utilizado, y también el que mejor representa el concepto de *Cloud Computing*. En general, son de coste reducido, o bien de pago bajo demanda.

Las principales ventajas que se pueden tener al usar este tipo de servicios son la flexibilidad y el ahorro principalmente en infraestructura. Además, se evitan las inversiones iniciales, muy importantes en los casos de negocios pequeños o medianos.

Algunos clouds públicos son el *Elastic Compute Cloud* de Amazon y *Windows Azure* de Microsoft. (5) (6)

Cloud híbrido

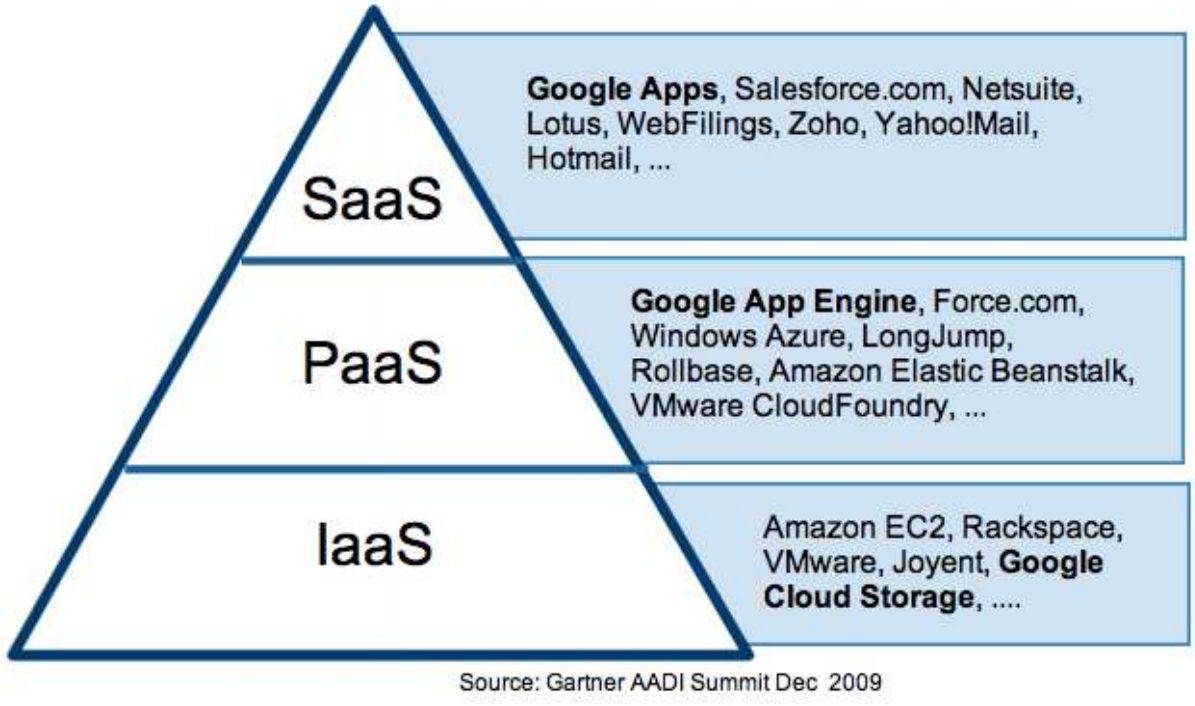
En este último tipo se mezclan las características de los modelos vistos anteriormente. El cliente gestiona y hace uso exclusivo de su infraestructura, y a su vez dispone de acceso al *cloud* público de algún proveedor.

El uso de un cloud de este tipo resulta muy adecuado cuando una organización:

- Utiliza una infraestructura privada para realizar sus actividades y hace uso de servicios públicos a la hora de almacenar la información.

- Utiliza un modelo de SaaS y necesita una infraestructura privada para almacenar información con requisitos de seguridad elevados [6].

2.1.4. Modelos de servicio



Existen tres modelos de servicios relativos a *Cloud Computing*, diferenciados por la forma en que los consumidores de servicio los utilizan. Es de notar que un sistema no tiene necesariamente pertenecer a un único de estos modelos (no son excluyentes).

Software como servicio (SaaS)

Este es quizá el tipo de modelo más clásico de los basados en *cloud*. Desde servicios sencillos como correo electrónico hasta modelos de gestión empresarial completos, podemos encontrar múltiples ejemplos de proveedores de este tipo casi desde los inicios de la computación.

Hacer uso de este tipo de servicios ofrece numerosas ventajas: el precio del software es pagado en un régimen de uso (bajo demanda), y no incluye inversión inicial, haciéndolo especialmente atractivo para pequeñas y medianas empresas en fase de crecimiento. Tradicionalmente, las compañías de mayor tamaño tienden a requerir software a medida, teniendo en cuenta otros factores como la eficiencia y seguridad.

Algunos ejemplos de software como servicio son *Google Apps, innkeypos, Quickbooks Online, Successfactors Bizx, Limelight Video Platform, Salesforce.com y Microsoft Office 365*.

Infraestructura como servicio (IaaS)

Este modelo se traduce en ofrecer equipamiento computacional (servidores web, tecnologías de red, centros de datos) entendido como un servicio. En lugar de adquirir estos recursos, los usuarios los alquilan, a menudo bajo demanda.

El servicio puede incluir escalado dinámico, de manera que si la aplicación requiriese en cualquier momento disponer de más recursos, los pudiese obtener de manera prácticamente inmediata.

Actualmente, el proveedor más importante de este tipo de servicios es Amazon EC2. (9) EC2 ofrece escalabilidad controlada por el usuario, pagando por los recursos a nivel de horas de uso. Otros servicios a destacar: *Rackspace Cloud, Terremark, Windows Azure Virtual Machines y Google Compute Engine*. (10)

Los usuarios más naturales de este tipo de servicio son las compañías y personas físicas enfocadas en la investigación. A menudo, los estudios científicos y médicos requieren de mucha capacidad de cómputo durante las fases de análisis y test, los cuales serían imposibles sin poder disponer de recursos adicionales en situaciones puntuales.

Plataforma como servicio (PaaS)

En este modelo se ofrece una plataforma en su totalidad, que incluye sistema operativo, un conjunto de aplicaciones, un gestor de BBDD y un conjunto de servicios web. Así, contiene todo lo que un desarrollador necesita para construir una aplicación.

Asimismo, puede ser visto como la evolución del web hosting, dado que una mayoría de proveedores de este tipo de servicio ya ofrecen una variedad de facilidades que conforman un conjunto bastante completo a la hora de desarrollar sitios web.

El beneficio principal de este modelo es disponer de software y de la capacidad de despliegue, basado en su totalidad en la nube, sin tener que preocuparse por adquirir o mantener la infraestructura. *PaaS* habitualmente nace de la necesidad de escalabilidad, y en referencia a esto se acuña el término "escalado dinámico", que significa que el software puede crecer o decrecer con facilidad.

Algunos servicios de este tipo son *Amazon Elastic Beanstalk, Cloud Foundry, Heroku, Force.com, EngineYard, Mendix, Google App Engine, Windows Azure Compute y OrangeScape*.

Comparativa modelos principales (11)

Modelo	Nivel típico de control cedido al cliente	Funcionalidad típica a disposición del cliente
IaaS	Uso y configuraciones de uso	Acceso a la interfaz de usuario
SaaS	Administrativo limitado	Control moderado sobre recursos relativos a la plataforma de uso
PaaS	Administrativo sin restricciones	Acceso completo a los recursos (virtualizados) de infraestructura, y posiblemente a los recursos físicos subyacentes

Otros modelos

Existen también otros modelos menos "difundidos" que surgen como solución a situaciones muy concretas. Se pueden entender también como una mezcla de dos de los modelos principales.

- **Escritorio como Servicio (DaaS):** Es conocido también como escritorio virtual, y ofrece al cliente un escritorio con todas las aplicaciones instaladas y listas para su ejecución. Cuando el usuario inicia sesión utiliza su configuración personal y toda la información que hubiese podido guardar en sesiones anteriores.

- **Entorno de pruebas como Servicio (TEaaS)**: Como su nombre lo indica, ofrece al usuario un entorno donde probar aplicaciones/software.

2.1.5. Un estándar abierto

Si tenemos en cuenta el desarrollo de la computación en nube hasta la fecha, podemos observar que esta tecnología es el resultado de la convergencia de muchas normas diferentes.

El crecimiento del Cloud Computing ha dado lugar a un importante impulso por parte de la industria para crear estándares abiertos basados en la nube. Así, en la actualidad, el Cloud Computing se rige por los siguientes estándares tecnológicos:

- **Framework orientado a aplicaciones web:** Consiste en un conjunto de conceptos, prácticas y criterios orientados a resolver ciertos problemas de las aplicaciones web, que sirven como referencia para resolver nuevos problemas de índole similar.

- **Despliegue de software de código abierto:** Consiste en el desarrollo de aplicaciones de código abierto, que puedan ser utilizadas y ampliadas por una comunidad de usuarios.

- **Arquitectura orientada a servicios:** Permite crear de sistemas de información altamente escalables que reflejan el negocio de una organización. Además, también brinda una forma bien definida de exposición e invocación de servicios, lo cual facilita la interacción entre diferentes sistemas propios o de terceros.

- **Virtualización de la plataforma de recursos:** Esencialmente, consiste en la abstracción de un sistema hardware completo, que permite que diversas instancias de sistemas operativos se ejecuten sobre ella.

- **Servicios web estandarizados:** Consiste en el uso de servicios web que sigan los estándares establecidos. Así, se facilita la incorporación de dichos servicios a las aplicaciones.

- **Uso de sistemas autónomos:** Un sistema autónomo es un conjunto de redes IP que poseen una política de rutas propia e independiente. Cada sistema autónomo gestiona el tráfico que fluye entre él y los restantes sistemas autónomos que forman internet.

- **Computación grid:** Permite utilizar de forma coordinada recursos de tipos muy diversos (como recursos cómputo, de almacenamiento y aplicaciones específicas) que no están sujetos a un control centralizado.

2.1.6. Ventajas y desventajas

Tomando como ejemplo el software de una organización y a sus empleados como usuarios, veamos los pros y los contras de usar un sistema de este tipo:

Ventaja	Desventaja
Accesibilidad: Permite que todos los empleados accedan a la plataforma de forma fácil y rápida con independencia del lugar en el que nos encontremos.	-
Reducción de costes: Al utilizar *Cloud Computing*, se reducen los gastos de adquisición y mantenimiento de los servidores sobre los que se ejecutan las aplicaciones.	-
Escalabilidad: Las infraestructuras cloud son capaces de suministrar recursos, software y/o datos en función de la demanda existente sin que los usuarios perciban diferencias.	**La escalabilidad debe planificarse con cuidado:** A medida que más usuarios empiecen a compartir una misma infraestructura, los servidores se encontrarán más sobrecargados. Si la organización no ha hecho las previsiones oportunas, la calidad del servicio puede degradarse en gran medida.
Seguridad: para mantener la integridad de los datos. Esta dependerá del modelo de *Cloud* que se utilice.	**Privacidad:** Es más una cuestión de confianza de los usuarios que un problema de la computación en la nube en sí. Al estar almacenados los datos de los usuarios en equipos externos, estos temen que otros puedan acceder fácilmente a su información.

Ventaja	Desventaja
Fácil integración de servicios: Las aplicaciones en la nube pueden integrarse en con otras de forma rápida y sencilla.	-
Prestación de servicios a nivel mundial: Con independencia de la ubicación de la infraestructura, los servicios que prestan los proveedores pueden utilizarse en cualquier parte del mundo de manera rápida y sencilla.	**Disponibilidad:** La disponibilidad de los servicios está estrechamente ligada al correcto funcionamiento de la red. **Dependencia:** Cuando un cliente opta por no utilizar un *Cloud* privado, pasa a depender de su proveedor para desarrollar total o parcialmente su actividad [11]
Actualizaciones automáticas: En los sistemas convencionales, cada vez que se realiza una actualización de la aplicación el cliente debe dedicar tiempo y esfuerzo para volver a personalizarla e integrarla. Con la computación en la nube, estas actualizaciones se realizan de manera automática y sin que el cliente se vea afectado.	

2.2. Aplicaciones existentes

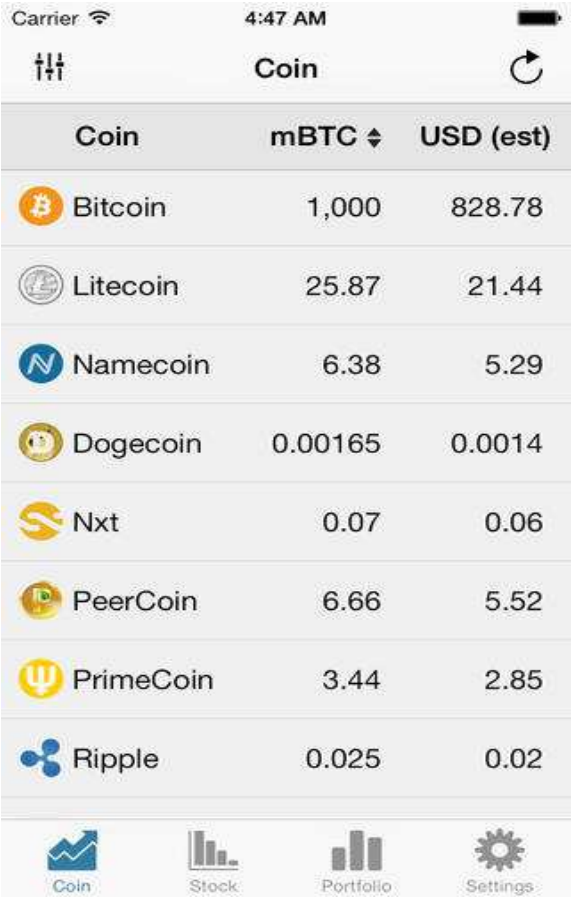
Vamos a hacer un repaso breve de algunas de las plataformas alternativas, con funcionalidad similar, disponibles actualmente en el mercado / la web.

2.2.1. Aplicaciones móviles

Estos son algunos ejemplos de aplicaciones móviles (12):

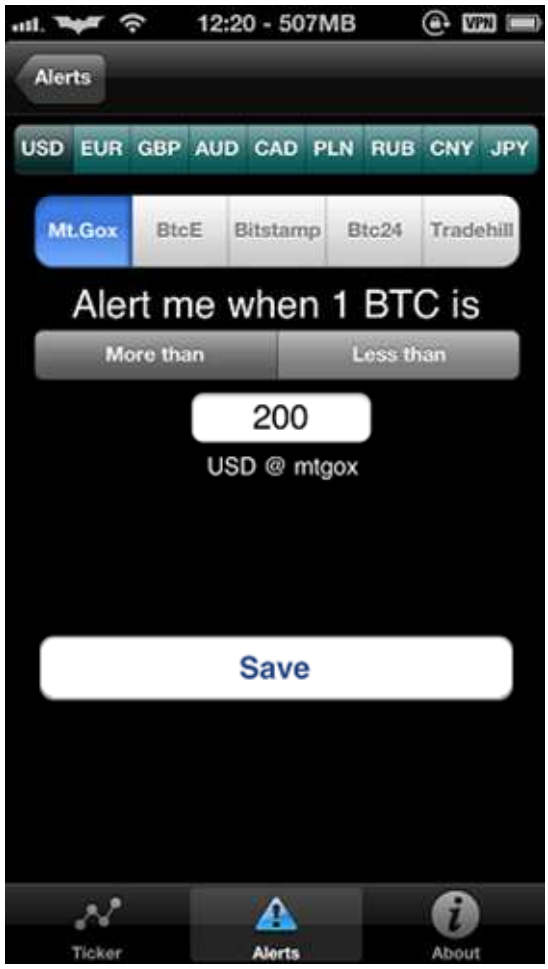
Coin Ticker

Se puede utilizar para comprobar Bitcoin y otros precios de cripto-divisas sobre la marcha. Las tasas de cambio se actualizan de forma automática cada minuto. También se puede utilizar esta aplicación para gestionar la cartera de todas las cripto-monedas.



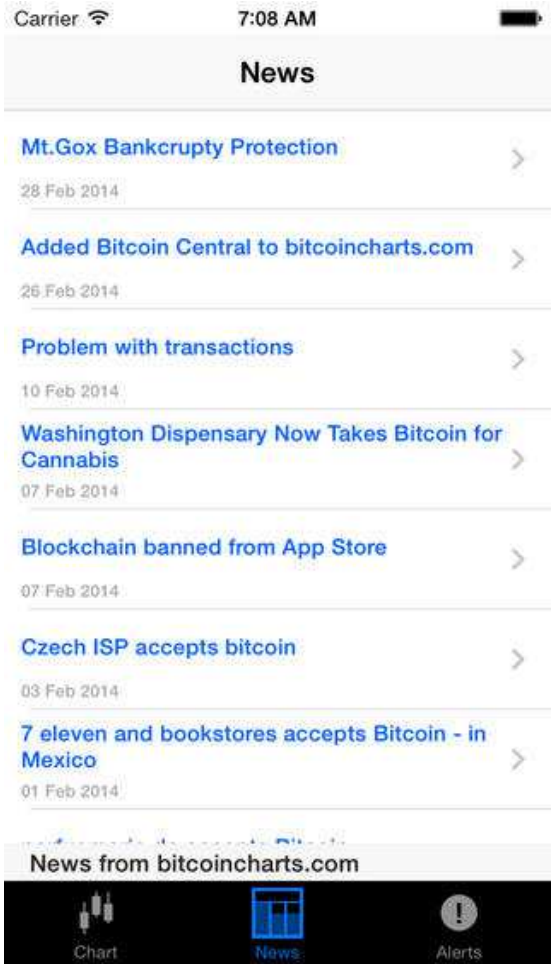
BitCoin Ticker

Como su nombre lo sugiere, esta aplicación muestra y monitorea las tasas de cambio actuales respecto a BitCoin (BTC). Las alertas se envían como notificaciones push directamente al teléfono.



BTC charts

Es otra aplicación gratuita que permite a los usuarios comprobar los datos más recientes relativos a las tasas de cambio actuales en BTC, así como el volumen en tráfico de estos.



Mobile Miner

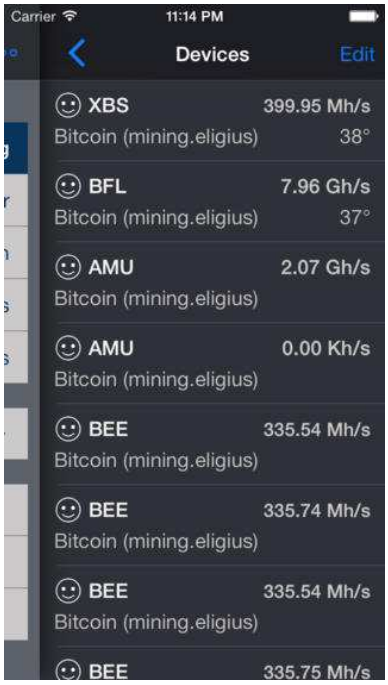
A día de hoy, es la única aplicación que permite convertir el móvil en un aparato de minería. En cualquier caso, al nivel de dificultad actual, resulta muy demorado ganar algo por este medio. Se cita aquí por completitud.

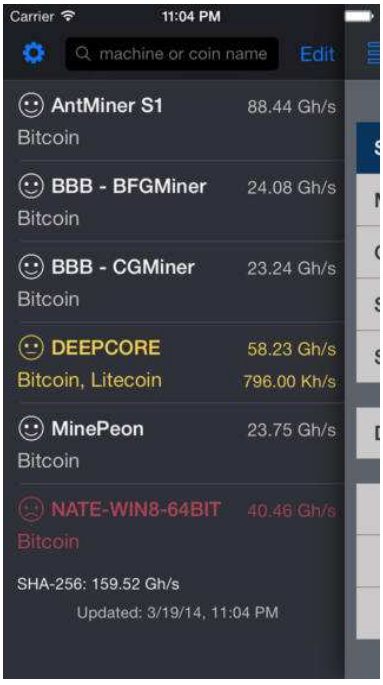
BTCstatus Slush's Pool, BTCguild view

Aplicaciones que permiten monitorizar los *miners* que se tengan enlazados a estos *pools* de minería. Son para casos muy concretos de uso, y su funcionalidad es muy básica.

MobileMiner App

Es una aplicación que permite controlar de manera remota máquinas que estén minando cripto-monedas (BitCoin, LiteCoin principalmente). De las estudiadas es la más completa, aunque contiene reducida documentación/soporte.





3. Arquitectura del sistema

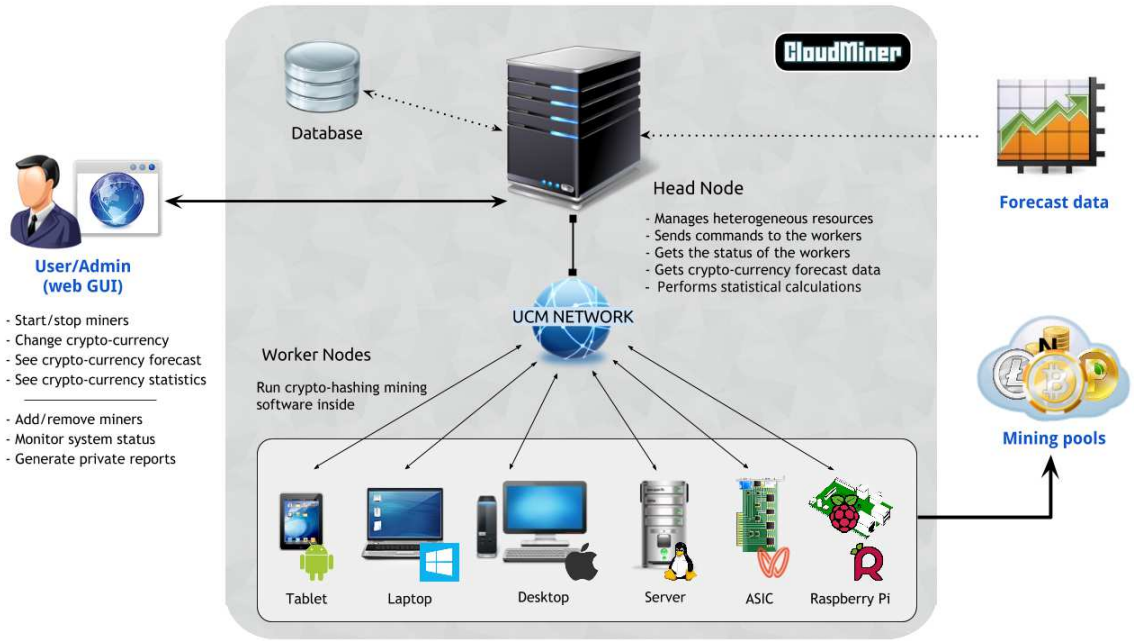
3.1. Diseño de alto nivel

Para poder resolver de manera eficaz el problema detallado en los capítulos anteriores, hemos de aprovechar que el proceso de minería de cripto-monedas es ejecutado en cada nodo de computación sin compartir memoria con otros procesos. En esta aproximación de alto nivel, omitiremos detalles como que cada uno de ellos podrá ejecutarse dentro de una unidad de procesamiento (CPU, GPU, etc.).

El principal problema que se plantea es cómo gestionar de forma eficiente, escalable y distribuida los procesos de minería sobre los recursos hardware de los que se disponga. Para ello, basta con implantar un sistema Cloud con una infraestructura como servicio (IaaS) y una plataforma como servicio (PaaS) como se detalla en el capítulo 2.

Por un lado, debemos disponer de un sistema centralizado que permita que los usuarios puedan controlar todos los procesos de minería, insertar nuevos recursos hardware y recopilar información sobre el estado de cada proceso. Por otro lado, los procesos de minería deben adaptarse a recursos hardware heterogéneos, sabiendo sobre qué arquitecturas hardware podrán ejecutarse correctamente, ejecutándose bajo demanda del usuario e informando al sistema centralizado sobre su estado y sobre posibles errores.

En la siguiente figura, para tener una visión global del sistema, se mostrará un diagrama del diseño a alto nivel de la solución.



Diseño de alto nivel de CloudMiner 1

3.2. Diseño de la arquitectura

3.2.1. Objetivos

La arquitectura del sistema debe garantizar el cumplimiento de los siguientes objetivos para que el sistema se implemente satisfactoriamente acorde al diseño de alto nivel especificado en la sección anterior:

- Escalable respecto del número de nodos que ejecutarán procesos de minería de cripto-monedas.
- Compatible con cualquier tipo de arquitectura hardware que soporte la ejecución de miners para cada plataforma concreta.
- Extensible para futuros miners y futuras cripto-monedas.
- Modular para permitir un control de usuario centralizado y multiplataforma.

3.2.2. Restricciones

Para que los objetivos se lleven a cabo, debemos tener presentes las restricciones que nos imponen tanto los propios objetivos, como el diseño de alto nivel:

- La escalabilidad sobre el número de nodos nos impone tres restricciones:
    - Las comunicaciones entre los nodos y el usuario deben ser lo más livianas posible.
    - Si establecemos una base de datos como almacén de información del estado de cada nodo, el motor de base de datos debe gestionar bien la concurrencia de inserción de datos.
    - Como consecuencia de la restricción anterior, el diseño de la base de datos debe permitir que las consultas sean eficientes.

- La compatibilidad nos impone restricciones sobre el lenguaje de programación, ya que éste ha de ser un lenguaje interpretado y multiplataforma para no depender de ningún modo de la arquitectura sobre la que se vaya a lanzar.

- Que sea extensible tan solo afecta al diseño de la base de datos.

- Que el control del usuario deba ser centralizado impone restricciones sobre el tipo de arquitectura que elegiremos.

3.2.3. Decisiones de diseño

Después de analizar los diferentes paradigmas para computación distribuida y teniendo en cuenta también las soluciones ya existentes comentadas en el capítulo 2, no llegamos a encontrar ninguna arquitectura que encajara perfectamente con las restricciones impuestas por el propio diseño de alto nivel descrito anteriormente.

Tras fijarnos detenidamente en cómo trabajan las arquitecturas PRC (Public Resource Computing), encontramos muchas similitudes con la forma en la que tenía que trabajar nuestra arquitectura. Un ejemplo de ello es el proyecto SETI@Home que utiliza la infraestructura BOINC (Berkeley Open Infrastructure for Network Computing), cuya arquitectura cliente-servidor hace que, numerosos clientes a lo largo del mundo, puedan ofrecer la potencia del cálculo de sus computadores y enviar las unidades de trabajo (en inglés work-units) a un servidor central. (13)

Sin embargo, aunque las fases de procesamiento y envío de las unidades de trabajo se asemejen mucho, no encontramos en dicha infraestructura una forma eficiente de crear una plataforma como servicio, para dar la capacidad de poder administrar y controlar cada uno de los nodos de computación.

Para que todo ello sea posible, se hace necesario utilizar una infraestructura ad-hoc (como variante de una arquitectura por capas) que proporcionará un diseño modular a nuestro sistema:

- **Nodo central** (Head Node): será el encargado de transmitir los comandos que deberá ejecutar posteriormente un nodo trabajador concreto.

- **Nodo trabajador** (Worker Node): deberá ejecutar el miner indicado por el nodo central y registrará la cantidad de hashes por unidad de tiempo que consiga.

- **Capa de integración**: hará de intermediario entre un nodo central y un nodo trabajador. Gestiona la distribución de los trabajos generados desde el nodo central y recopila los resultados arrojados desde el nodo trabajador.

- **Capa de gestión y presentación**: proporcionará al usuario una forma efectiva para poder trabajar con el sistema completo de forma centralizada, traduciendo sus órdenes para que el nodo central pueda transmitirlas y recopilando la información recogida en la capa de integración.

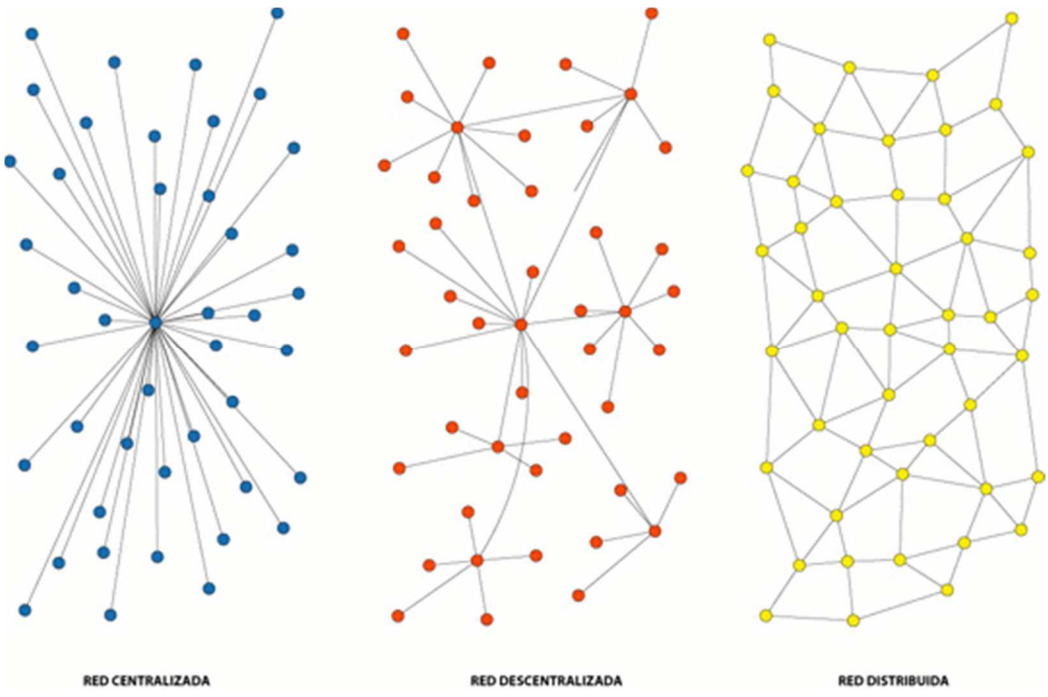
Dado que los componentes de dicha infraestructura pueden interactuar entre sí de formas muy diversas, hemos tomado las siguientes decisiones de diseño respecto a cada uno de ellos:

- El nodo central y el nodo trabajador serán procesos ejecutables en máquinas distintas y, por tanto, creímos en la conveniencia de hacer uso de alguna arquitectura de programación distribuida de tipo **cliente-servidor** debido a la heterogeneidad de los potenciales nodos trabajadores y de las redes que los interconectarán con el nodo central.

  Sin embargo, siendo la funcionalidad principal del nodo central el envío de comandos y la de un nodo trabajador será transmitir los resultados del proceso de minería al nodo central, no podemos establecer claramente los roles de cliente y de servidor. Por lo tanto, usaremos redes ***Peer to Peer*** (P2P o entre iguales) para que ambos sean capaces de enviar y recibir lo que se estime conveniente.

  Entre las características de las redes *Peer to Peer* que más nos han llamado la atención han destacado su escalabilidad y su robustez, que nos han sido útiles para poder montar toda la infraestructura sin ningún tipo de problemas a posteriori.

  De toda la gama de redes *Peer to Peer* que existen según su centralización, hemos tomado la decisión de que sea totalmente **centralizada**.



  En este diseño (como se puede apreciar en la imagen de la izquierda), todos los nodos trabajadores deben estar conectados a un mismo nodo central, que serán los que pueda controlar el usuario final. Si el número de nodos trabajadores creciera desmesuradamente, podríamos usar un sistema de balanceo de carga dividiendo la red P2P en subconjuntos de un tamaño concreto.

- Para poder establecer la arquitectura de comunicaciones entre los nodos, en la capa de integración utilizaremos el protocolo **TCP/IP** (14) para lograr interconectar cualquier tipo de nodo trabajador, pudiendo incluso comunicarse a través de Internet. Esto nos dará mucha versatilidad a la hora de trabajar como un Cloud híbrido, además de la robustez que nos proporciona el propio protocolo a la hora de entregar los mensajes de forma controlada. Esto exigirá que ambos nodos soporten dicho protocolo a nivel de capa de aplicación, es decir, que cuente con un interfaz para los *sockets TCP*.

  También se requerirá que los puertos asignados para cada extremo de la conexión se encuentren visibles entre ellas. Al tratarse de *peers* y no de clientes y servidores, si la comunicación se realiza a través de Internet, se hace necesario el uso del protocolo **NAT** (Network Address Translation) (15) tanto en la red del nodo trabajador como en la del nodo central.

- De cara a la creación de un sistema que interrelacione la interfaz que se le presentará al usuario con la información de los nodos trabajadores y que traduzca las peticiones del usuario a comandos tratables por el nodo central, hemos aplicado el patrón arquitectural **MVC** (Modelo Vista Controlador).

  Hemos apostado por dicho patrón en pro de conseguir separar los datos y la funcionalidad del sistema (lógica de negocio) de la forma en que presenta la información al usuario (lógica de diseño).

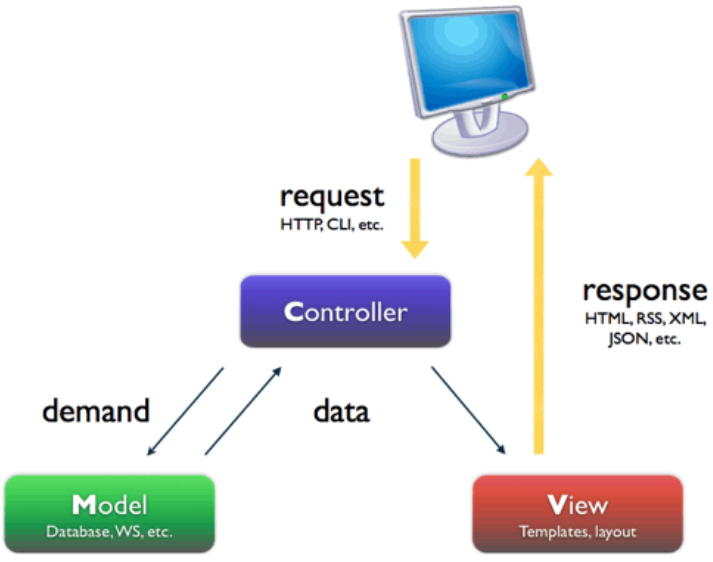


Ilustración del patrón MVC 1

El patrón se puede implementar perfectamente como aplicación de escritorio pero, para cumplir con los objetivos del diseño, se implementará como una aplicación web, lo que permitirá un acceso *on-line* a la plataforma, aunque impondrá como requisito fundamental su despliegue en un servidor web.

3.3. Implementación

3.3.1. Nodo central

El nodo central se encarga de traducir y encapsular las órdenes del usuario, para transmitírselas posteriormente al nodo trabajador. Toda la información para construir el comando, le vendrá suministrada a través de la capa de gestión y presentación.

Dichos comandos serán principalmente los siguientes:

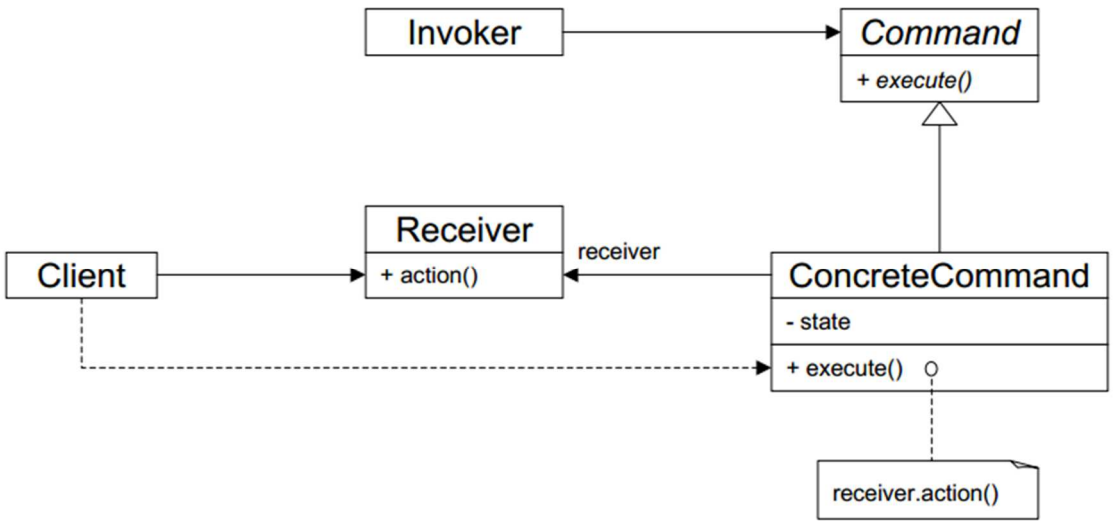
- Inicio y detención de un proceso de minería concreto en un nodo trabajador.
- Detención del proceso entero del nodo trabajador.

La idea es que el formato sea lo más versátil posible con miras a poder extenderlo con muchas otras órdenes pero, a su vez, debe establecer una sintaxis lo más estricta posible para evitar problemas de parametrización y ejecución no controlada.

Patrón Command

Para su correcta implementación haremos uso del patrón *Command* (16), que nos proporcionará las siguientes características:

- Encapsulación de peticiones en forma de objetos.
- Parametrizar a los clientes con distintas peticiones.
- Invocar las acciones de modo uniforme.
- Extensible de forma sencilla.



Estructura del Patrón Command 1

Sus participantes serán:

- **Comando** (Command): interfaz de ejecución de operaciones.
- **Comando concreto** (ConcreteCommand): interfaz de ejecución invocando operaciones del receptor. Relaciona una acción con un receptor.
- **Cliente**: crea un comando concreto dirigido a un receptor.
- **Invocador**: contiene el comando asociado a una petición.
- **Receptor**: sabe cómo realizar las operaciones asociadas a una petición. Esta parte se explicará en la sección siguiente ya que forma parte del nodo trabajador.

Por otro lado, se necesita asegurar la serialización de los comandos, es decir, que se ejecuten en el mismo orden que se enviaron, se ha optado por implementar un envío síncrono de dichos comandos. Esto implica, como se verá posteriormente en la capa de integración, que el nodo central deberá esperar a que el nodo trabajador le comunique que ha recibido el comando correctamente (*Callback*).

3.3.2. Nodo trabajador

Como se explica en la sección anterior, el nodo trabajador debe actuar como receptor en el patrón *Command*, por tanto, dispondrá de un hilo principal de **escucha asíncrona** (explicado con más detalle en la capa de integración) que lo primero que hará tras haber recibido un comando por parte del nodo central, será devolverle una señal de confirmación (*Callback*) para que pueda seguir enviando otros comandos.

A continuación, al incorporar un **intérprete de comandos** acorde a la sintaxis común, clasificará la petición y ejecutará la operación asociada.

La ejecución de los comandos se realizará en *Threads* separados (*WorkerThread*) del proceso principal impidiendo que se bloquee el proceso de escucha asíncrona de peticiones y poder así actuar sobre cada proceso de minería de forma autónoma con comandos sucesivos.

Para proceder a ejecutar un *miner,* el nodo trabajador ha de consultar a la capa de gestión (más concretamente a la base de datos) el comando específico que iniciará el ejecutable en el sistema operativo destino.

Una vez que esté todo listo, se procede a ejecutarlo por línea de comandos mediante un *Pipe* (tubería del sistema operativo) que canalizará la salida del *miner* de la línea de comandos de vuelta al *WorkerThread*. La idea es que dicho proceso sea multiplataforma y evitar así tener que adaptarnos a cada sistema operativo en particular.

La salida del *Pipe* ha de sufrir un proceso de *parsing* para encontrar en ella la cantidad de hashes criptográficos que está minando por unidad de tiempo (*hashing*).

Por último, el nodo trabajador debe enviar dicho valor de forma homogénea, para ello se normalizará en "mega hashes por segundo" (MH/s) y se enviará a la capa de gestión. Es importante destacar que, en beneficio de la escalabilidad, no es conveniente mandar los resultados constantemente, sino que ha de hacerse cada cierto intervalo de tiempo (30 segundos como promedio).

Esto último tiene fuertes implicaciones en el cálculo exacto de la tasa de *hashing* que percibirá el usuario, puesto que al discretizar en intervalos temporales tan grandes, dicha tasa puede sufrir errores considerables. Sin embargo, para tranquilidad del usuario, los propios procesos de minería producen tasas de *hashing* muy estables y, además, incluyen por defecto ese intervalo de "refresco" (aunque puede ser modificado por línea de comandos pero no es lo más conveniente).

3.3.3. Capa de integración

La capa de integración está compuesta por los diferentes conectores, albergados en cada uno de los subsistemas de la infraestructura, capaces de comunicarse entre sí usando el protocolo TCP/IP como base.

El envío de comandos, como se explicó en la sección dedicada a la implementación del nodo central, debe ser **síncrono** para garantizar su serialización.

La recepción de los comandos, sin embargo, deber ser **asíncrona** para garantizar que el proceso de minería no bloquee la comunicación del nodo central mientras está a la espera.

Por todo ello, necesitaremos usar el patrón *Callback.*

Patrón Callback

El patrón *Callback* (17), también llamado patrón Reactor, lo usaremos para realizar una petición (*Request*) con el nodo central como emisor y con el nodo trabajador como receptor, y para obtener una respuesta (*Response*) siendo ahora el nodo trabajador el emisor y el nodo central el receptor.

La petición la usaremos para poder enviar el comando y la respuesta (confirmación del comando) para poder desbloquear el proceso en el nodo central y que pueda seguir enviando.

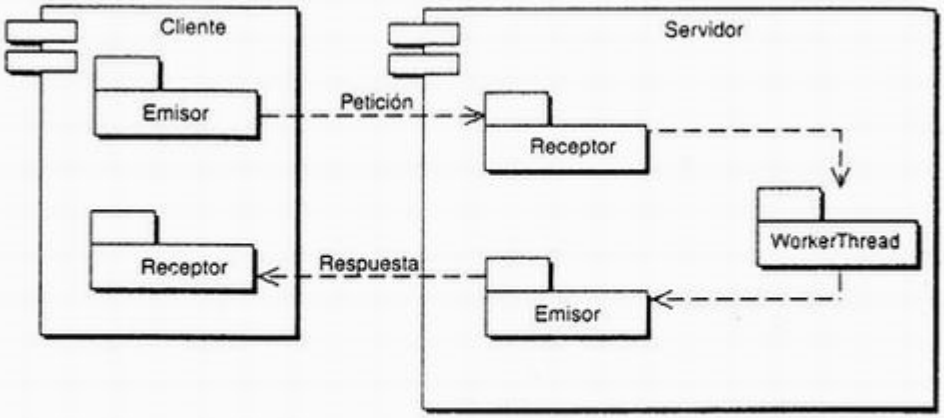


Diagrama de componentes del patrón Callback.

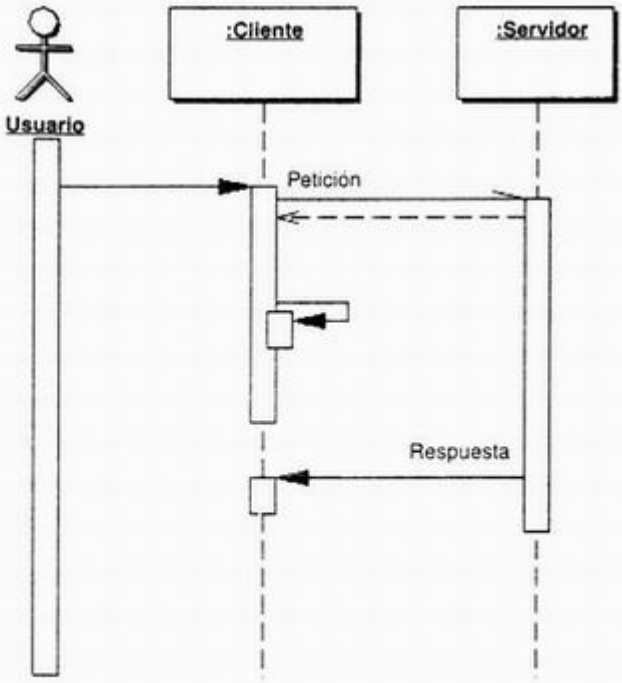
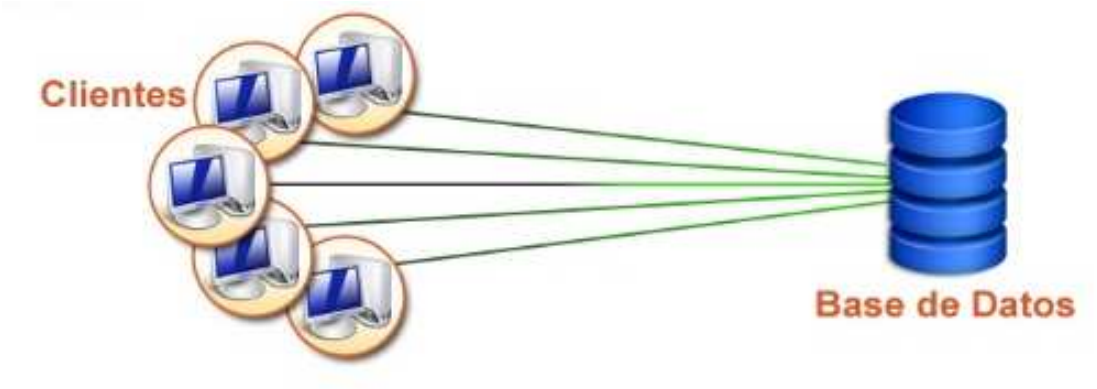


Diagrama de secuencia del patrón Callback.

Diagramas del patrón Callback 1

Conector del SGBD

También hay que considerar como parte de la capa de integración al conector con la base de datos, ya que la comunicación del estado de los nodos trabajadores no se realiza por medio del nodo central directamente (en ese caso saturaría al proceso del nodo central), sino que se realiza por medio de la conexión directa con la base de datos.



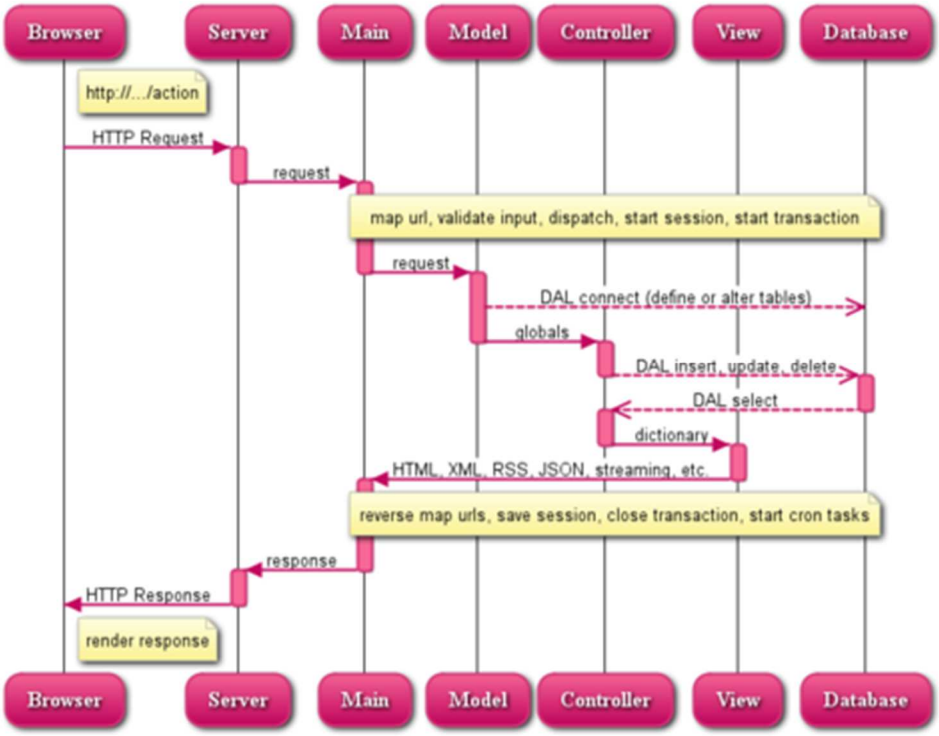
Esta es, simplemente, una decisión de diseño para evitar el colapso absoluto del nodo central y distribuir su carga, aprovechando que los sistemas gestores de bases de datos (SGDB) modernos disponen de una gestión eficiente de conexiones concurrentes (mediante un *pool* de conexiones o similar)

3.3.4. Capa de gestión y presentación

La capa de gestión y presentación está formada por un sistema gestor de bases de datos (SGBD) y por un marco de desarrollo (*framework*) de aplicaciones web como solución completa para implementar el modelo-vista-controlador (MVC).

En este punto, debemos tener en cuenta que necesitaremos dos servicios activos, un servidor web y un servidor de bases de datos. Sobre el primero desplegaremos el *framework y* sobre el segundo crearemos la base de datos a la que se conectará el *framework* y donde se establecerán los modelos.

El funcionamiento interno del *framework MVC* que hemos utilizado (descrito en el siguiente capítulo) es el que se muestra en la siguiente figura:



Funcionamiento del framework MVC Web2py 1

Toda la funcionalidad de nuestro sistema, detallada en el capítulo 5, es posible gracias a la modularidad y robustez que nos proporciona esta estructura.

Además, este *framework* que hemos utilizado es capaz de integrar como módulo al nodo central y es capaz, a través de un controlador, de llamar a sus funciones con los datos que ordene el usuario.

4. Tecnologías

4.1. Tecnologías utilizadas

4.1.1. Python

¿Qué es Python?

Python es un lenguaje de programación creado por Guido van Rossum a finales de 1980 cuyo nombre fue inspirado en el grupo de cómicos ingleses 'Monty Python'. Es un lenguaje de programación cuya filosofía hace hincapié en una sintaxis muy limpia y legible.

Se trata de un lenguaje interpretado, con tipado dinámico, fuertemente tipado, orientado a objetos y multi-plataforma. Es administrado por la *Python Software Foundation*. Posee una licencia de código abierto, denominada *Python Software Foundation License*, (18) que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

¿Por qué Python?

Teniendo en cuenta las características mencionadas anteriormente, es decir, por sintaxis simple, clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías que dispone y la gran potencia del lenguaje hacen que desarrollar una aplicación en Python sea rápido, sencillo, con un código limpio y legible.

Algunos casos de éxito en el uso de Python son *Google, MSN, Yahoo, Wikipedia, Youtube, NASA* y todas las distribuciones de Linux.

4.1.2. Web2py

¿Qué es Web2py?

Web2py es un framework de desarrollo web de código abierto de muy fácil aprendizaje, incluye las últimas tecnologías de una forma simple y clara, tales como MVC, ORM, sistemas de plantillas, soporte para JavaScript, AJAX, etc., lo que convierte en una solución completamente funcional para crear aplicaciones web de manera totalmente interactiva. (19)

Está escrito en Python y es programable en Python. Separa la representación de datos (el modelo) de la presentación de datos (la vista) y también de la lógica de la aplicación y flujo de trabajo (el controlador). Proporciona bibliotecas para ayudar a los desarrolladores a diseñar, implementar y probar cada una de estas tres partes por separado, para luego trabajar juntas.

Su objetivo principal es dar soporte al desarrollo ágil de software de aplicaciones web escalables, seguras y portables. Resuelve muchos de los problemas que pueden dar lugar a vulnerabilidades de seguridad, siguiendo prácticas bien establecidas. Por ejemplo, valida todas las entradas (para evitar las inyecciones de código), escapa todas las salidas (para evitar el cross-site scripting), cambia el nombre de los archivos cargados (para prevenir ataques que recorran el directorio). Web2py deja poco espacio de maniobra a los desarrolladores de aplicaciones en cuestiones relativas a la seguridad.

Además incluye una capa de abstracción de base de datos (DAL por su acrónimo en inglés) que escribe SQL dinámicamente para que el desarrollador no tenga que hacerlo. El DAL sabe cómo generar SQL de forma transparente para SQLite, MySQL, PostgreSQL, MSSQL, FireBird, Oracle, IBM DB2, Informix e Ingres. El DAL también puede generar llamadas a funciones para Google BigTable cuando se ejecuta en el Google App Engine (GAE). Una vez que una o más tablas de bases de datos están definidas, Web2py también genera una interfaz de administración de base de datos basada en la web, la cual es totalmente funcional para acceder a la base de datos y las tablas.

Difiere de otras plataformas web porque es la única que acoge plenamente el paradigma Web 2.0, donde la red es el computador. De hecho, Web2py no requiere instalación o configuración, sino que funciona en cualquier arquitectura en la que se puede ejecutar python (Windows, Windows CE, Mac OS X, iPhone, y Unix/Linux), y las fases de desarrollo, despliegue y mantenimiento para las aplicaciones se pueden hacer mediante una interfaz web local o remota. Web2py se ejecuta con Cpython (la implementación C) y Jython (la implementación Java), versiones de 2.4, 2.5 y 2.6 aunque “oficialmente” sólo soporta 2.5 para así poder garantizar la compatibilidad con versiones previas de aplicaciones.

Web2py proporciona un sistema de ‘tickets’. Si ocurre un error, los ‘tickets’ se expiden para el usuario, y el error se registra para el administrador.

Web2py es de código abierto y está liberado bajo la licencia GPL 2.0, pero las aplicaciones desarrolladas con Web2py no están sujetas a ninguna restricción de licencia. De hecho, siempre y cuando no contengan el código fuente Web2py, no se consideran “obra derivada”.

También le permite al desarrollador compilar en bytecode las aplicaciones y distribuirlas como de código cerrado, aunque se requiera Web2py para ejecutarlas. La licencia Web2py incluye una excepción que permite a los desarrolladores web enviar sus productos con binarios originales Web2py pre-compilados, sin el acompañamiento del código fuente.

¿Por qué Web2Py?

Elegimos Web2py por una diversidad de factores, entre ellas podemos destacar:

1) Tiene una curva de aprendizaje muy llana respecto a otros frameworks similares y provee un entorno de desarrollo completamente basado en web (se puede desarrollar en "cualquier parte").

2) Como se menciona en la introducción, Web2py tiene un foco muy importante en la seguridad prestando mecanismos predeterminados seguros, previniendo las vulnerabilidades más comunes.

3) Es ligero. Sus librerías centrales, incluyendo la capa de abstracción de bases de datos, el lenguaje de la plantilla y todas las ayudas constan en total de 1.4MB. Todo el código fuente incluyendo aplicaciones de muestra e imágenes alcanzan los 10.4MB

4) Ocupa poco espacio y es muy rápido. Utiliza el servidor Web Rocket(22) WSGI desarrollado por Timothy Farrell. Es 30% más rápido que Apache con mod_proxy. E en un PC promedio, éste sirve una página dinámica media sin acceso de base de datos en aproximadamente 10 ms. La DAL tiene una tasa de sobrecarga muy baja, por lo general inferior al 3%.

Características distintivas

A continuación enumeramos sus características más relevantes (20):

- **Entorno de desarrollo integrado (IDE) basado en web:** Todo el desarrollo puede ser realizado (opcionalmente) vía una interfaz web, sin herramientas de terceros, dentro de la misma aplicación web2py. Cada aplicación tiene una interfaz de base de datos integrada, similar a la de Django. La IDE web también incluye un editor de código, editor html para las vistas, herramienta de pruebas y una consola basada en web. Las aplicaciones también pueden ser creadas desde la línea de comandos o desarrolladas con otras IDEs.

- **Vistas Flexibles:** Web2py incluye un lenguaje de plantilla rápido, basado en Python puro, sin requerimientos de Indentacion y un modelo de objeto de documento (DOM). El lenguaje de plantillas funciona sin web2py. Las plantillas Joomla 1.x pueden ser convertidas a web2py. Un controlador sin una vista automáticamente usa una vista genérica para mostrar las variables devueltas por el controlador, permitiendo desarrollar la lógica del negocio antes de escribir el HTML.

- **Sistema de tiquets:** Cada aplicación web2py viene con un Sistema de seguimiento de incidentes; si ocurre un error, es registrado y un ticket es generado para el usuario. Esto permite seguimiento de errores. Los errores y el código fuente son solo accesibles por el administrador, quien puede buscar y recuperar los errores por fecha o IP del cliente. El código nunca se expone al usuario.

- **Protección del código fuente:** Web2py puede compilar las aplicaciones web y estas pueden ser distribuidas en forma de bytecode, sin el código fuente. Esto ayuda pero no garantiza la protección del código fuente debido a la existencia de

desensambladores y decompilador para Python con efectividad variada (archivos *.pyc y *.pyo) (21)

- **Capa de abstracción de BBDD (DAL):** Web2py implementa una DAL, no un ORM. Un ORM mapea las tablas de una base de datos en clases y los registros en instancias de estas clases. La DAL en cambio mapea, de manera transparente, las tablas en instancias de una clase (Table) y los registros en instancias (DAL Set, DAL Rows, DAL Fields). Tiene una sintaxis muy similar a un ORM pero es más rápido, (22) y puede manejar casi cualquier expresión SQL. La DAL puede ser usada sin web2py. (23) SQLite está incluido en Python y es la base de datos de web2py por defecto. Un cambio en la cadena de conexión permite usar Firebird, IBM DB2, Informix, Ingres, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, y la Google App Engine (GAE).

- **Migraciones automáticas de esquema:** Web2py soporta migración automática al cambiar la definición de una tabla, web2py la ALTERa (puede ser deshabilitado). Las migraciones, sus intentos y los cambios son registrados (logged) y documentados.

Plugins

Web2py admite usar *plugins*, funcionalidades autónomas agregables a cualquier aplicación Web2py existente. Usualmente definen componentes, por ejemplo objetos que pueden ser embebidos en una página y comunicarse con el servidor vía Ajax. Frecuentemente incluyen el correspondiente plugin jQuery y su funcionalidad del lado del servidor para hacerlos funcionar.

La interfaz administrativa de web2py provee una interfaz web para subir y administrar los plugins. Una vez que el plugin es parte de su aplicación, puede adaptarlo sin afectar otras aplicaciones. Su apariencia puede ser adaptada usando CSS.

Énfasis en la seguridad

Como ya hemos mencionado antes, este framework hace mucho hincapié en el apartado de la seguridad. Esto es muy importante al estar desarrollando una plataforma de *Cloud Computing*.

El Proyecto de Seguridad de Aplicaciones Web Abiertas (19) (OWASP por sus siglas en inglés) es una comunidad mundial abierta y libre enfocada en la mejora de la seguridad de aplicaciones de software. Web2py aborda los diez principales temas de seguridad definidos por OWASP. Veamos esto en detalle:

Cross Site Scripting (XSS): Las fallas de XSS ocurren cuando una aplicación toma los datos suministrados por el usuario y los envía a un navegador web sin validar o codificar el contenido. XSS permite a los atacantes ejecutar scripts en el navegador de la víctima lo cual puede secuestrar sesiones de usuario, deformar sitios web, posiblemente introducir gusanos, etc.	Web2py, por defecto, escapa todas las variables presentadas en la vista, previniendo que ocurra XSS.
Las fallas de inyección, en particular inyección SQL, son comunes en aplicaciones web. La inyección se produce cuando los datos ingresados por el usuario son enviados a un intérprete como parte de un comando o consulta. Los datos hostiles del atacante engañan al intérprete para que ejecute comandos no deseados o cambie los datos.	Web2py incluye una Capa de abstracción de base de datos que hace imposible la inyección SQL. Normalmente, las declaraciones SQL no están escritas por el desarrollador. En su lugar, el código SQL es generado dinámicamente por el DAL, asegurando que todos los datos insertados sean escapados correctamente.
Ejecución de archivos maliciosos: Código vulnerable a la inclusión de archivos remotos (RFI) permite a los atacantes incluir código hostil y datos, resultando en ataques devastadores, como el compromiso total del servidor. web2py sólo permite que sean ejecutadas las funciones expuestas, previniendo la ejecución de archivos maliciosos. Funciones importadas nunca son expuestas; sólo las acciones son expuestas.	Web2py utiliza una interfaz de administración basada en Web que hace muy fácil hacer un seguimiento de lo que se expone y lo que no.
Una referencia directa a objetos se produce cuando un desarrollador expone una referencia a un objeto de implementación interna, tal como un archivo, directorio, registro de base de datos, o clave, como una URL o un parámetro de formulario. Los atacantes pueden manipular esas referencias para tener acceso a otros objetos sin autorización. web2py no expone ningún objeto interno y además valida todas las direcciones URL, con lo que logra la prevención de ataques que recorran el directorio.	Web2py también proporciona un mecanismo simple para crear formularios que validan automáticamente todos los valores ingresados.

Requerimiento falsificado Cross-site (CSRF): Un ataque CSRF obliga a una sesión de usuario en el navegador de la víctima a enviar una solicitud pre-autentificada a una aplicación web vulnerable, lo que entonces obliga al navegador de la víctima a realizar una acción hostil en beneficio del atacante. CSRF puede ser tan poderosa como la aplicación web que ataca.	Web2py previene CSRF, así como el doble ingreso de los datos en formularios mediante la asignación de una única ficha aleatoria a cada formulario. Además Web2py utiliza UUID para la cookie de sesión.
Fugas de información y manejo de errores inapropiado: Las aplicaciones pueden provocar fugas no intencionales de información relacionada con su configuración, funcionamiento interno, o violar la privacidad a través de una variedad de problemas de aplicación. Agresores utilizan esta debilidad para robar datos importantes, o realizar ataques más graves.	Web2py incluye un sistema de tickets. Ningún error puede resultar en código que se expone a los usuarios. Todos los errores son registrados y los tickets se expiden al usuario lo que permite el seguimiento del error. Errores y códigos fuente son accesibles sólo para el administrador.
Administración de sesión y autentificación rota: las credenciales de cuentas y las fichas de sesión a menudo no son debidamente protegidas. Los atacantes comprometen las contraseñas, claves o las fichas de autenticación para asumir las identidades de otros usuarios.	Web2py proporciona un mecanismo integrado para la autenticación de administrador, y gestiona de forma independiente sesiones para cada aplicación. La interfaz de administración también obliga a la utilización de cookies de sesión segura cuando el cliente no es “localhost”. Para aplicaciones, incluye un potente API de control de acceso basado en roles.
Almacenamiento criptográfico inseguro: las aplicaciones web rara vez utilizan de manera correcta las funciones criptográficas para proteger los datos y las credenciales. Agresores utilizan datos protegidos débilmente para llevar a cabo el robo de identidad y otros delitos, tales como fraude de tarjeta de crédito.	Web2py utiliza los algoritmos hash MD5 o HMAC la + SHA-512 para proteger las contraseñas almacenadas. Otros algoritmos también están disponibles.
Comunicaciones inseguras: Las aplicaciones frecuentemente fallan en encriptan el tráfico de red cuando es necesario para proteger las comunicaciones sensibles.	Web2py incluye el servidor con SSL(21) Rocket WSGI, pero también puede usar Apache o Lighttpd y mod_ssl para proporcionar cifrado SSL de las comunicaciones.

Falla en restringir acceso URL: Con frecuencia una aplicación sólo protege la funcionalidad sensible al impedir la visualización de los enlaces o direcciones URL a usuarios no autorizados. Atacantes pueden usar esta debilidad para acceder y realizar operaciones no autorizadas mediante el acceso a las URL directamente.	Web2py mapea las solicitudes de URL a los módulos y funciones de python. Además proporciona un mecanismo para que se especifique cuáles son funciones públicas y cuales requieren autenticación y autorización. El API de control de acceso basado en roles incluido, permite a los desarrolladores restringir el acceso a cualquier función basados en la sesión, pertenencia a grupos o permisos basados en grupos. Los permisos son muy granulares y se pueden combinar con CRUD para permitir, por ejemplo, acceso a los tablas y/o registros específicos.

4.1.3. SB Admin v2.0

Se trata de una plantilla web (24) que funciona con otras tecnologías de última generación como jQuery y Bootstrap. Es una versión actualizada del tema original (también gratuito) *SB Admin free Bootstrap*. Permite utilizar componentes muy prácticos como:

- Gráficas de línea temporal
- Widgets (chat, login, etc)
- Botones circulares y tipo página social
- Gráficas tipo Morris.js

4.2. Tecnologías descartadas

4.2.1. Django

Introducción

Según sus desarrolladores, Django es “El framework Web para perfeccionistas con fechas límite”. (25)

Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el paradigma conocido como Model Template View. Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kansas, y fue liberada al público bajo una licencia BSD en julio de 2005; el framework fue nombrado en alusión al guitarrista de jazz gitano Django Reinhardt. (26)

La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas (DRY, del inglés Don't Repeat Yourself). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

Características

- “Mapeador” objeto-relacional.
- *Plugins* que pueden instalarse en cualquier página gestionada con Django.
- API de base de datos robusta.
- Sistema incorporado de "vistas genéricas" que ahorra tener que escribir la lógica de ciertas tareas comunes.
- Sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.
- Despachador de URLs basado en expresiones regulares.
- Sistema "middleware" para desarrollar características adicionales; por ejemplo, la distribución principal de Django incluye componentes middleware que proporcionan cacheo, compresión de la salida, normalización de URLs, protección CSRF y soporte de sesiones.
- Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.
- Documentación incorporada accesible a través de la aplicación administrativa (incluyendo documentación generada automáticamente de los modelos y las bibliotecas de plantillas añadidas por las aplicaciones).

Soporte de Base de Datos

Se recomienda el uso de PostgreSQL, pero también soporta MySQL y SQLite 3. Se encuentra en desarrollo un adaptador para Microsoft SQL Server. Una vez creados los data models, Django proporciona una abstracción de la base de datos a través de su API que permite crear, recuperar, actualizar y borrar objetos. También es posible que el usuario ejecute sus propias consultas SQL directamente. En el modelo de datos de Django, una clase representa un registro de una tabla en la base de datos y las instancias de esta serán las filas en la tabla.

Soporte de servidores Web

Django incluye un servidor web liviano para realizar pruebas y trabajar en la etapa de desarrollo. En la etapa de producción, sin embargo, se recomienda Apache 2 con mod_python. Aunque Django soporta la especificación WSGI, por lo que puede correr sobre una gran variedad de servidores como FastCGI o SCGI en Apache u otros servidores.

Motivos de descarte

El motivo principal por el que descartamos este framework es por la curva de aprendizaje. Se trata de un framework muy completo, y que permite hacer cosas sencillas en un par de minutos. Ahora bien, en cuanto queremos complicar un poco las cosas y añadir más funcionalidad, requiere ahondarse con cierta profundidad en la documentación y los grupos de soporte.

Lo anterior nos lleva a nuestro segundo motivo principal: la documentación. Si bien existe mucha y variada a disposición del usuario/programador, es excesivamente difícil de navegar, y en multitud de ocasiones, para buscar la solución a un problema concreto, hay que buscar mucho para finalmente encontrarlo.

5. Funcionalidad del sistema

5.1. Casos de uso

5.1.1. Registro en CloudMiner

	Nombre: registro en CloudMiner	
Descripción	Es la primera vez que el usuario/admin usa la aplicación y para esto debe registrarse.	
Objetivo	Registrarse en el sistema	
Precondición	El usuario no debe estar todavía registrado	
Actor principal	Usuario	
Canal del actor principal	Interfaz web	
Disparador	El usuario hace click en el desplegable de “log in” y posteriormente en el enlace de “sign in”	
Secuencia	Paso	Acción
	1	El usuario accede a la pagina ppal. de la aplicación
	2	El usuario ubica el mouse sobre el desplegable de "log in"
	3	El usuario presiona en el enlace de "sign in"
	4	El usuario rellena el formulario con sus datos y lo envía haciendo click en “aceptar”.
	5	El sistema crea un usuario con los datos introducidos
	6	El sistema redirecciona al usuario a la página principal de la aplicación
Postcondición	Se crea un nuevo usuario en el sistema	
Excepciones	1	Error de servidor, al fallar el acceso a la plataforma. El usuario es notificado del error.
	4	Ya existe el usuario, se notifica al usuario y se repite el paso 4.
	5	Error de conexión. Se notifica al usuario y se vuelve al paso 4.
Frecuencia de uso	Habitual	
Importancia	Alta	
Estabilidad	Alta	
Comentarios	-	

5.1.2. Modificar datos de usuario

	Nombre: modificar datos de usuario	
Descripción	Un usuario/admin requiere modificar sus datos de acceso	
Objetivo	Modificar en BBDD los datos personales	
Precondición	El usuario debe estar dado de alta	
Actor principal	Usuario	
Canal del actor principal	Interfaz web	
Disparador	El usuario hace click en el desplegable de “Welcome <user>” y posteriormente en el enlace de “profile”	
Secuencia	Paso	Acción
	1	El usuario accede a la página ppal. de la aplicación
	2	El usuario ubica el mouse sobre el desplegable de "Welcome <user>"
	3	El usuario presiona en el enlace de "profile"
	4	El usuario rellena el formulario con los datos que quiere modificar y hace click en el boton de aceptar.
	5	El sistema modifica los datos del usuario en BBDD
	6	El sistema redirecciona al usuario a la página en la que estuviese previamente
Postcondición	Quedan modificados los datos del usuario	
Excepciones	1	Error de servidor, al fallar el acceso a la plataforma. El usuario es notificado del error.
	4	Ya existe un usuario con los datos introducidos, se notifica al usuario y se repite el paso 4.
	5	Error de conexión. Se notifica al usuario y se vuelve al paso 4.
Frecuencia de uso	Variable	
Importancia	Media	
Estabilidad	Alta	
Comentarios	-	

5.1.3. Iniciar un nodo

	Nombre: iniciar un nodo	
Descripción	Se quiere agregar un recurso hardware para que este disponible para minería	
Objetivo	Habilitar el nodo en el sistema	
Precondición	El nodo no debe estar activo	
Actor principal	Usuario	
Canal del actor principal	Consola (linux: shell, windows: ms-dos)	
Disparador	El usuario inicia el ejecutable WorkerNode.py	
Secuencia	Paso	Acción
	1	El usuario accede, usando la consola, al directorio donde se encuentra alojado el WorkerNode
	2	El usuario introduce el comando "python WorkerNode.py"
	3	El software correspondiente al Nodo se inicia y efectua comprobaciones en BBDD
	3.1	Si el nodo no estaba dado de alta en el sistema, se añade con todos los datos pertinentes y marca como activo en BBDD
	3.2	Si el nodo ya existía en el sistema, se actualizan los datos IP y puerto, y se marca como activo en BBDD
	6	Se notifica al usuario de la ID del nodo, así como de la IP y puerto por los que recibirá comandos del nodo central
Postcondición	Se activa el nodo en el sistema	
Excepciones	2	Error de conexión a BBDD, el usuario es notificado del error.
	3.1	No existe un tipo de plataforma correspondiente a la arquitectura del nodo, se notifica al usuario y se termina la ejecución
Frecuencia de uso	Habitual	
Importancia	Alta	
Estabilidad	Alta	
Comentarios	-	

5.1.4. Desactivar un nodo

	Nombre: desactivar un nodo	
Descripción	Se quiere detener un nodo activo, utilizando la interfaz web	
Objetivo	Detener un nodo	
Precondición	El nodo debe estar activo	
Actor principal	Usuario	
Canal del actor principal	Interfaz web	
Disparador	El usuario hace click en el boton de "parar" en el panel de control	
Secuencia	Paso	Acción
	1	El usuario accede al panel de control de las máquinas/nodos (machine_cp)
	2	El usuario selecciona el nodo que quiere detener y hace click en el boton de "stop"
	3	El sistema envía el comando correspondiente al nodo seleccionado
	4	El sistema devuelve automáticamente al panel de control de las máquinas/nodos
Postcondición	El nodo deja de estar activo	
Excepciones	3	Error de conexión, no se detiene el nodo. El usuario es notificado del error
Frecuencia de uso	Habitual	
Importancia	Alta	
Estabilidad	Alta	
Comentarios	-	

5.1.5. Iniciar un trabajador

	Nombre: iniciar un trabajador	
Descripción	Se quiere iniciar un nuevo trabajador en un nodo activo, utilizando la interfaz web	
Objetivo	Iniciar un miner	
Precondición	El nodo debe estar activo	
Actor principal	Usuario	
Canal del actor principal	Interfaz web	
Disparador	El usuario hace click en el boton de "iniciar"	
Secuencia	Paso	Acción
	1	El usuario accede al panel de control de las máquinas/nodos (machine_cp)
	2	El usuario selecciona el nodo donde quiere iniciar un trabajador/miner y hace click en el boton de iniciar
	3	Elusuario escoge a continuación que miner quiere iniciar, haciendo click en el boton de iniciar
	4	El sistema envía el comando correspondiente al nodo seleccionado
	5	El sistema devuelve automáticamente al panel de control de las máquinas/nodos
Postcondición	Queda iniciado el miner en el nodo	
Excepciones	4	Error de conexión, no se inicia el miner. El usuario es notificado del error
Frecuencia de uso	Habitual	
Importancia	Alta	
Estabilidad	Alta	
Comentarios	-	

5.1.6. Detener un trabajador

	Nombre: detener un trabajador	
Descripción	Se quiere detener un trabajador/miner en un nodo activo, utilizando la interfaz web	
Objetivo	Detener un miner	
Precondición	El nodo debe estar activo y el miner iniciado	
Actor principal	Usuario	
Canal del actor principal	Interfaz web	
Disparador	El usuario hace click en el boton de "parar" en el panel de control	
Secuencia	Paso	Acción
	1	El usuario accede al panel de control de las máquinas/nodos (machine_cp)
	2	El usuario selecciona el nodo donde quiere parar el trabajador/miner y hace click en el botón de detener
	3	El sistema envía el comando correspondiente al nodo seleccionado
	4	El sistema devuelve automáticamente al panel de control de las máquinas/nodos
Postcondición	Queda iniciado el miner en el nodo	
Excepciones	3	Error de conexión, no se detiene el miner. El usuario es notificado del error
Frecuencia de uso	Habitual	
Importancia	Alta	
Estabilidad	Alta	
Comentarios	-	

5.1.7. Añadir una plataforma

	Nombre: añadir una plataforma	
Descripción	Se quiere agregar una plataforma a la BBDD, para habilitar a su vez agregar/iniciar nodos que correspondan a esta	
Objetivo	Añadir una plataforma a la BBDD	
Precondición	La plataforma debe no existir en BBDD	
Actor principal	Usuario	
Canal del actor principal	Interfaz web	
Disparador	El usuario hace click en Manager -> Platforms	
Secuencia	Paso	Acción
	1	El usuario accede al grid de plataformas haciendo click en el menu Manager -> Platforms
	2	El usuario hace click en el botón "Add Record"
	3	El usuario rellena todos los datos del formulario y hace click en "send"
	4	El sistema añade un registro a la BBDD con los datos introducidos
	5	El sistema redirige al usuario al grid de plataformas
Postcondición	Queda añadida la plataforma en el sistema	
Excepciones	3	Error de conexión a BBDD, el usuario es notificado del error.
	4	Existe una plataforma con los mismos datos. El usuario es notificado y se vuelve al paso 3
Frecuencia de uso	Habitual	
Importancia	Alta	
Estabilidad	Alta	
Comentarios	-	

5.1.8. Añadir un miner

	Nombre: añadir un miner	
Descripción	Se quiere agregar un miner a la BBDD, para habilitar a su vez iniciar trabajadores que le utilicen	
Objetivo	Añadir un miner a la BBDD	
Precondición	El miner debe no existir en BBDD	
Actor principal	Usuario	
Canal del actor principal	Interfaz web	
Disparador	El usuario hace click en Manager -> Miners	
Secuencia	Paso	Acción
	1	El usuario accede al grid de miners haciendo click en el menu Manager -> Miners
	2	El usuario hace click en el botón "Add Record"
	3	El usuario rellena todos los datos del formulario y hace click en "send"
	4	El sistema añade un registro a la BBDD con los datos introducidos
	5	El sistema redirige al usuario al grid de miners
Postcondición	Queda añadido el miner al sistema	
Excepciones	3	Error de conexión a BBDD, el usuario es notificado del error.
	4	Existe un miner con los mismos datos. El usuario es notificado y se vuelve al paso 3
Frecuencia de uso	Habitual	
Importancia	Alta	
Estabilidad	Alta	
Comentarios	-	

5.1.9. Añadir un pool de minería

	Nombre: añadir un pool de minería	
Descripción	Se quiere agregar un pool a la BBDD, para habilitar a su vez iniciar trabajadores que le utilicen	
Objetivo	Añadir un pool a la BBDD	
Precondición	El pool debe no existir en BBDD	
Actor principal	Usuario	
Canal del actor principal	Interfaz web	
Disparador	El usuario hace click en Manager -> Pools	
Secuencia	Paso	Acción
	1	El usuario accede al grid de pools haciendo click en el menu Manager -> pools
	2	El usuario hace click en el botón "Add Record"
	3	El usuario rellena todos los datos del formulario y hace click en "send"
	4	El sistema añade un registro a la BBDD con los datos introducidos
	5	El sistema redirige al usuario al grid de pools
Postcondición	Queda añadido el pool al sistema	
Excepciones	3	Error de conexión a BBDD, el usuario es notificado del error.
	4	Existe un pool con los mismos datos. El usuario es notificado y se vuelve al paso 3
Frecuencia de uso	Variable	
Importancia	Alta	
Estabilidad	Alta	
Comentarios	-	

5.1.10. Añadir una cripto-moneda (divisa virtual)

	Nombre: añadir una cripto-moneda	
Descripción	Se quiere agregar una cripto-moneda a la BBDD, para habilitar a su vez hacer minería de esta	
Objetivo	Añadir una cripto-moneda a la BBDD	
Precondición	La cripto-moneda debe no existir en BBDD	
Actor principal	Usuario	
Canal del actor principal	Interfaz web	
Disparador	El usuario hace click en Manager -> Currencies	
Secuencia	Paso	Acción
	1	El usuario accede al grid de cripto-monedas haciendo click en el menu Manager -> Currencies
	2	El usuario hace click en el botón "Add Record"
	3	El usuario rellena todos los datos del formulario y hace click en "send"
	4	El sistema añade un registro a la BBDD con los datos introducidos
	5	El sistema redirige al usuario al grid de cripto-monedas
Postcondición	Queda añadida la cripto-moneda en el sistema	
Excepciones	3	Error de conexión a BBDD, el usuario es notificado del error.
	4	Existe una cripto-moneda con los mismos datos. El usuario es notificado y se vuelve al paso 3
Frecuencia de uso	Variable	
Importancia	Alta	
Estabilidad	Alta	
Comentarios	-	

6. Manual de uso

Este manual tiene como objetivo:

- Brindar una descripción clara y detallada sobre el funcionamiento y uso e instalación del sistema *CloudMiner*.
- Guiar al usuario por las distintas partes dentro de la aplicación a la hora de navegar, para así apoyarlo en la búsqueda de la información que ofrece *CloudMiner*.

6.1. Requisitos del sistema

Para poder desplegar la aplicación *CloudMiner* en el sistema, la máquina del usuario debe cumplir los siguientes requisitos.

- Tener instalado el sistema operativo, ya sea *Linux* o *Windows*.
- Navegador web en cualquiera de sus versiones con acceso a internet, a continuación indicamos algunos navegadores más comunes:
  - *Google Chrome*.
  - *Mozilla Firefox*.
  - *Internet Explorer*.
  - *Safari*.
  - *Opera*
- Intérprete de *Python* versión 2.7.
- El *framework* de desarrollo web *Web2py.*
- El gestor de bases de datos *MySQL.*
- La librería *PyMySQL* para usar como conector de MySQL en *Python*.

6.2. Instalar la aplicación

El usuario antes de ejecutar la aplicación *CloudMiner* en su máquina, su equipo debe cumplir los requisitos previamente indicados.

A continuación se indican los pasos de instalación de los requisitos previamente indicados.

6.2.1. Instalación de *Python*

Para poder instalar *Python* en se deben seguir las siguientes instrucciones:

Windows

Descargar el instalador, para ello acceder a la página web oficial de *Python* (http://www.python.org/), una vez descargado el instalador, haciendo doble clic en él se inicia la instalación.

Para una correcta instalación, el usuario sólo debe seguir las instrucciones del asistente de instalación.

Linux

El proceso de instalación en el sistemas operativo *Linux* es sencilla, sólo debe ejecutar el siguiente comando para poder instalar *Python* en su equipo.

Comando de instalación de *Python:*

```
$ sudo apt-get install python2.7
```

6.2.2. Instalación de *Web2py*

Web2Py viene empaquetado para varios sistemas operativos, por lo cual, su instalación es muy sencilla, y su filosofía "baterías incluidas" hace prácticamente no tengamos que descargar o instalar otras dependencias, es decir, ni bibliotecas ni paquetes extras (27).

Windows

Para este sistema operativo viene en un paquete *.zip*, para descargar Web2py solo debe acceder a dirección web
http://www.web2py.com/init/default/download

Una vez descargado, descomprimirlo y ejecutar haciendo doble clic en web2py.exe

Linux

Descargar y desempaquetar *Web2Py* para el sistemas operativo *Linux* accediendo a la siguiente dirección web:
http://www.web2py.com/init/default/download, ejecutar el archivo *web2py* correspondiente.

Para ejecutar el *web2py* desde consola, debe acceder al directorio de we2py que se haya descargado y ejecutar el siguiente código:

```
user@user:~/web2py$ python web2py.py
```

6.2.3. Instalación de *MySQL*

Windows

Para instalar *MySQL,* el usuario, primero debe descargar el software accediendo a la página correspondiente para descargar *MySQL* para *Windows* (http://www.mysql.com/why-mysql/windows/).

El software tiene un asistente fácil de usar basado en la interfaz de usuario, el cual guiará durante el proceso de instalación.

Linux

Para instalar del servidor y el cliente de *MySQL*, el usuario debe instalar los paquetes *MySQL-Server, MySQL-Common y MySQL-Cliente*.

El proceso de instalación se realiza ejecutando el siguiente comando.

- Instalación de *MySQL-Server*
```
$ sudo apt-get install mysql-server
```
- Instalación de *MySQL-Common*
```
$ sudo apt-get install mysql-common
```
- Instalación de *MySQL-Client.*
```
$ sudo apt-get install mysql-client
```

6.2.4. Instalación de *PyMySQL*

Windows

Para instalar *PyMySQL* debe acceder a la página web oficial de *Python* (https://pypi.python.org/pypi/PyMySQL/) y descargar el instalador para *Windows,* una vez descargado el instalador ejecutar y seguir las instrucciones del asistente de la instalación de *PyMySQL*.

Linux

Si el usuario tiene disponible *PyPI* en su equipo, PyMySQL se puede instalar con *pip*, ejecutando el siguiente comando en la consola de *Linux*

```
$ pip install PyMySQL
```

Si el usuario no dispone *PyPI* en su equipo, se puede instalar PyPI de dos formas

- Descargar desde la página oficial de *Python* (https://pypi.python.org/pypi), donde éste viene empaquetado. Una vez descargado, desempaquetarlo e instalarlo ejecutando el siguiente comando
```
$ python setup.py install
```
- La otra forma más fácil y rápida de instalar PyPI es ejecutado
```
$ python easy_install pip
```

Con cualquiera de las dos opciones, el usuario lo tendrá listo *PyPI* en su equipo, de esta forma ya puede utilizar el comando para instalar *PyMySQL*:

```
$ pip install PyMySQL
```

6.2.5. Creación de la base de datos

Se debe crear la base de datos, para ello vamos a utilizar *MySQL,* donde se almacería toda la información que vaya a generar la aplicación *CloudMiner.*

Windows

Para crear la base de datos, en la consola de Windows(CMD) deben ejecutar los siguientes comandos.

- Empezamos a ejecutar como usuario *root*:

```
> mysql -u root -p
```

```
> password: <el password que ha elegido el usuario en la configuración a la hora de installar MySQL>
```

```
> mysql -u root -p
```

- Una vez que esté en la consola de *MySQL,* el usuario crea la base de datos con el siguiente comando:

```
mysql> CREATE DATA BASE nombreBBDD
```

- usar la base de datos creada en el apartado anterior.

```
mysql> USE nombreBBDD
```

Linux

Para ejecutar y crear la base de datos en *MySQL*, abrir el terminal de comandos de *Linux,* y ejecutar los siguientes comandos:

- Conectar con el servidor *MySQL*

```
$ mysql -h localhost -u nombre_usuario -p
```

  Lo primero que nos preguntará será el *password* para el usuario *root*
- Una vez que se acceda a *MySQL*, para crear la base de datos, ejecutar el comando:

```
mysql> create database miBaseDatos;
```

- Lo más normal es que primero te tenga que conectar con la base de datos creada en el apartado anterior. Eso se hace con el comando *use*, seguido del nombre de la base de datos.

```
mysql> use miBaseDatos;
```

- Si no es la primera vez se accede a la base de datos, es decir, si antes ya se había creado la base de datos. Para conectar *MySQL* con una base de datos ya creada con anterioridad, ejecutamos el siguiente comando desde la consola de *Linux*:

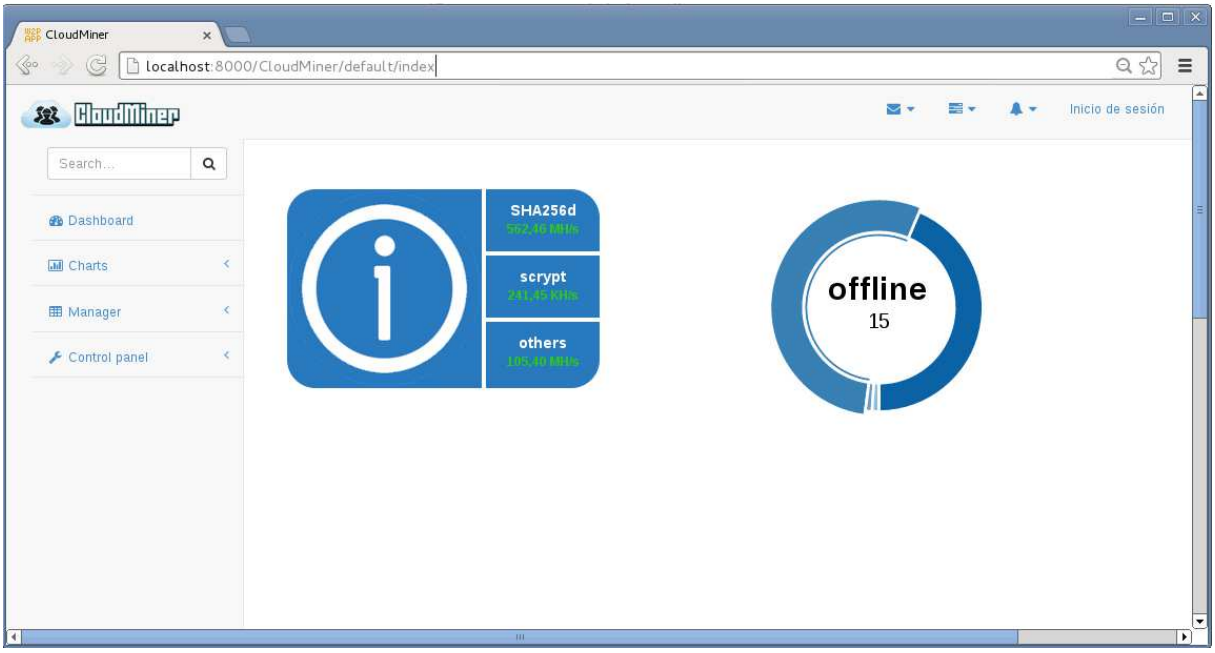
```
$ mysql -u nombre_usuario -p'password' nombreBBDD
```

6.3. Ejecutar la aplicación

Si la aplicación *CloudMiner* se encuentra en el equipo local del usuario, los pasos a seguir son los siguientes:

- Conectar las base de datos con el servidor local.
- Acceder a la siguiente dirección: http://localhost:8000/CloudMiner/default/index

Una vez que el usuario acceda a dicha dirección, el sistema se ejecutará y presentará al usuario el *Index* de la aplicación *CloudMiner*.

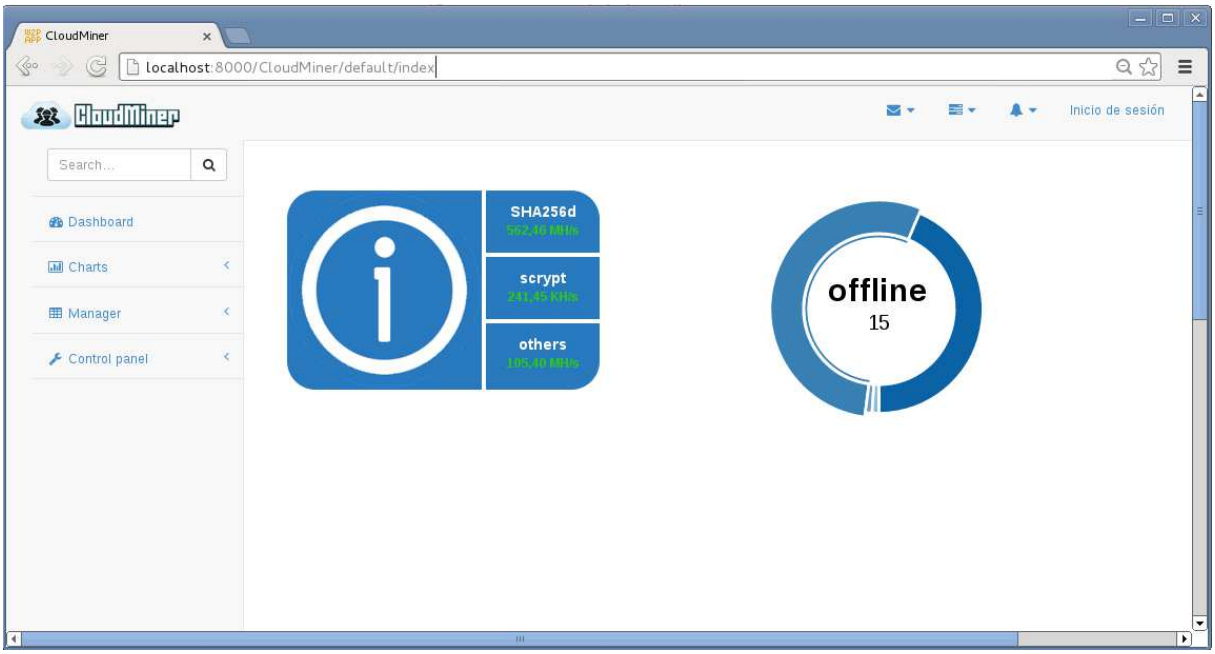


6.4. Frontend

En este apartado se le muestra y se le explica al usuario la parte del sistema que interactúa con él.

Para acceder a la aplicación de *Cloudminer*, el usuario debe visitar el siguiente enlace: http://localhost:8000/CloudMiner/default/index

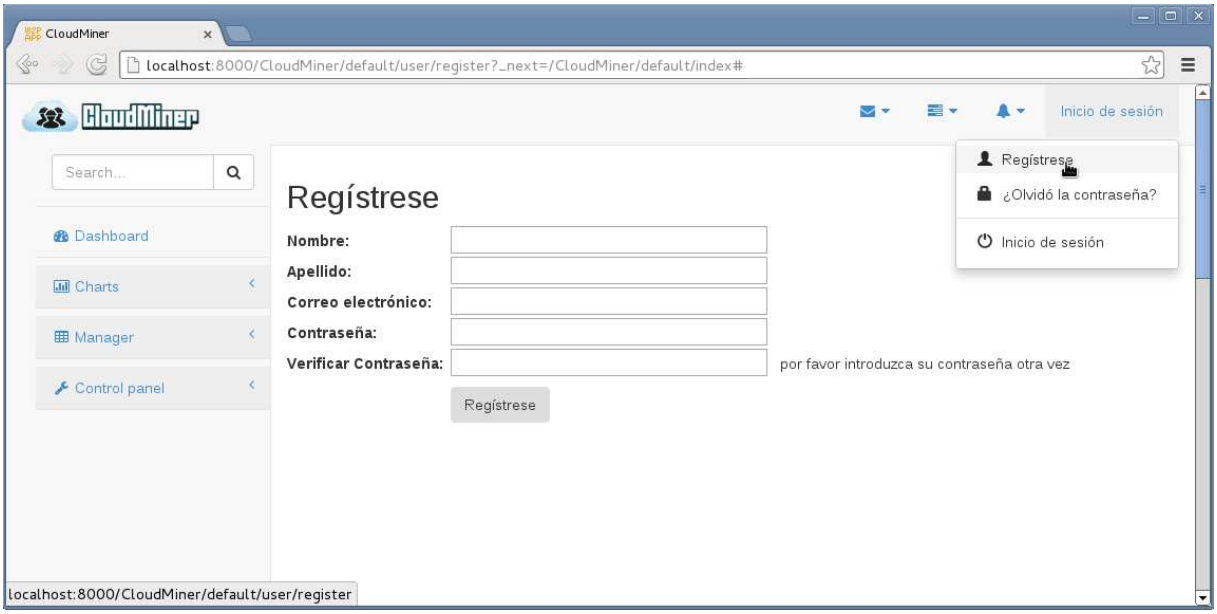
Una vez accedido a esta dirección, al usuario se le presentará la siguiente interfaz de *CloudMiner.*



6.4.1. Formulario de registro

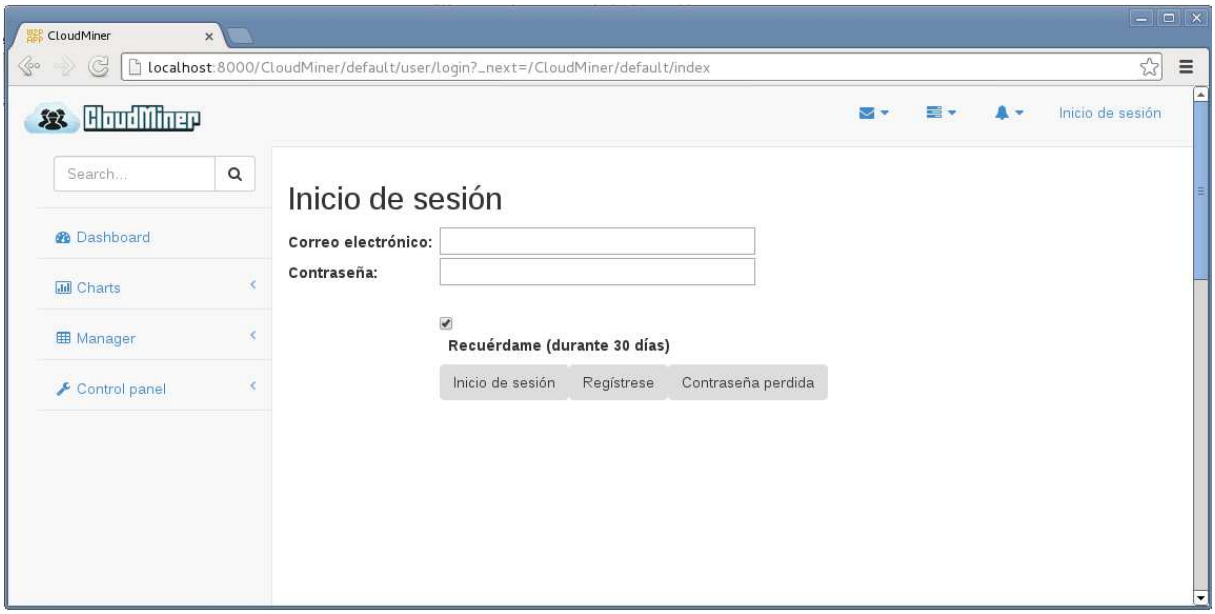
Para poder registrarse, el usuario debe introducir sus datos en los campos correspondientes en el siguiente formulario de registro

.



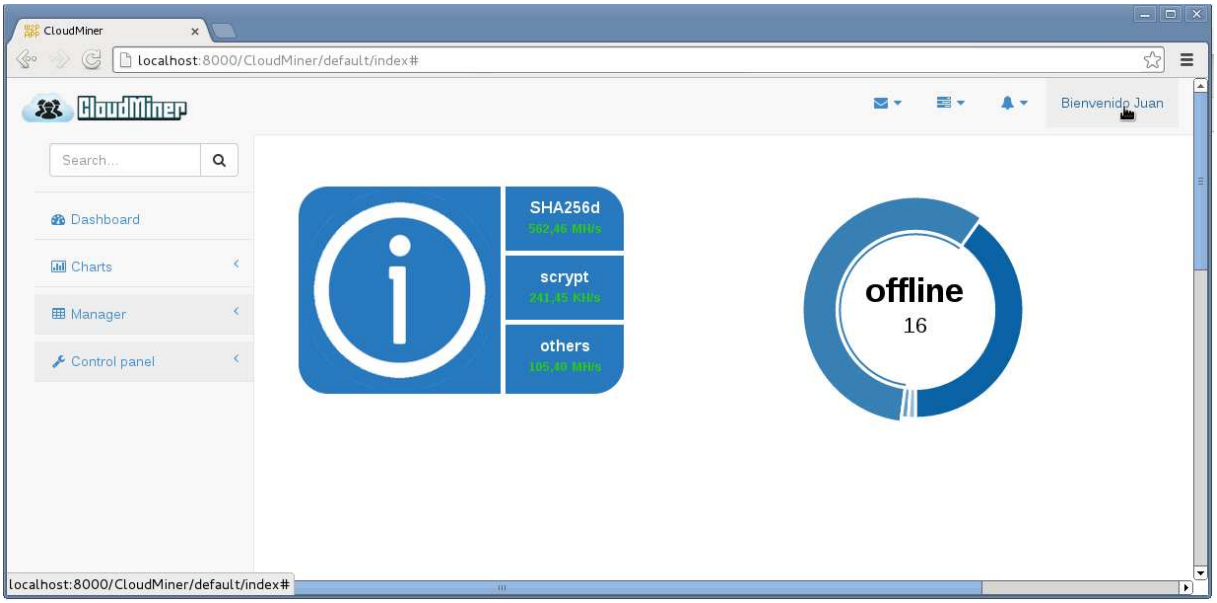
6.4.2. Formulario de acceso

Una vez que el usuario se haya registrado, para hacer el login, el sistema pide al el correo electrónico y el password con el que se haya hecho el registro.



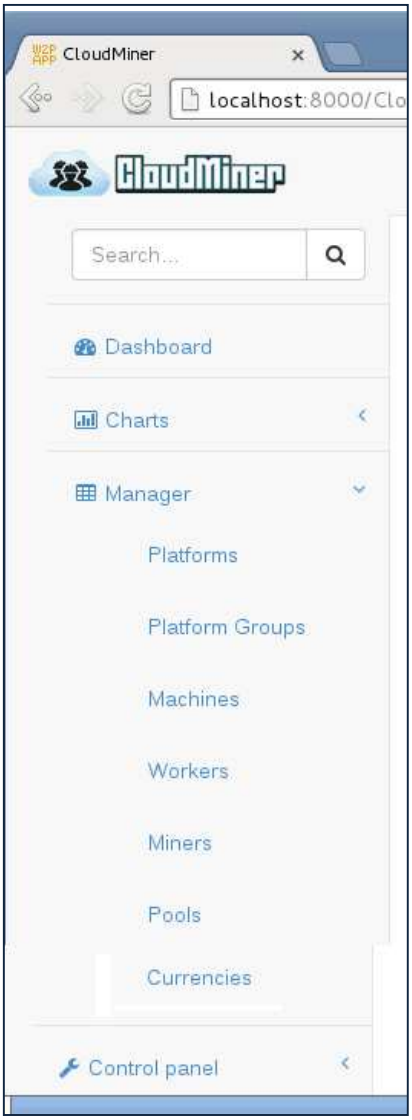
6.4.3. Interfaz principal de usuario

Una vez introducidos los datos correctos, el sistemas de bienvenida al usuario es como se muestra en la siguiente imagen.



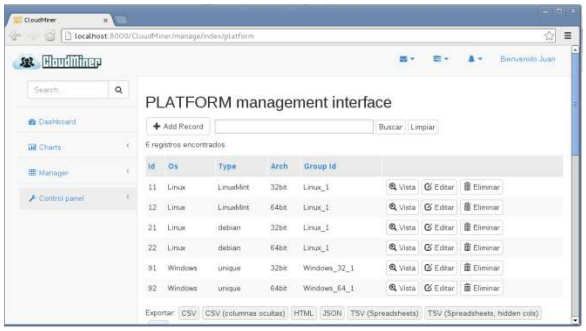
6.4.4. Menú manager

El menú manager tienes varios submenús, que permite al usuario visualizar la información de los distintos submenús que se muestran en la siguiente imagen (*Plataforms, Plataforms groups, Machines, Miners, Pools, Currencies*)

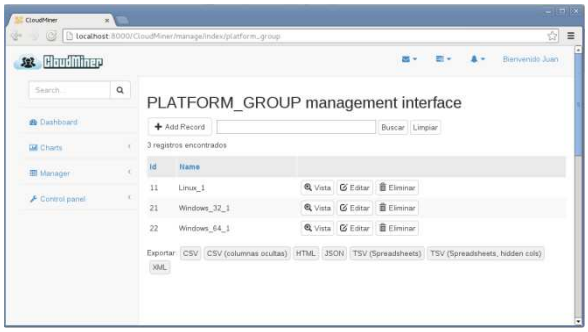


Submenús de manager

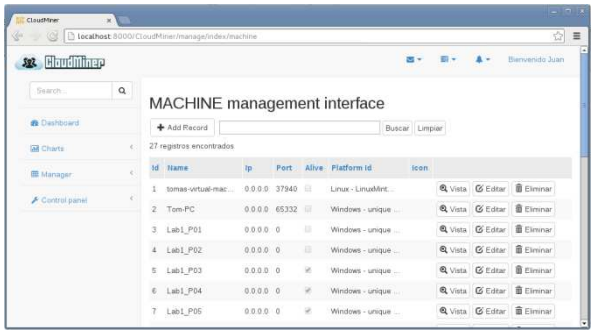
Plataforms



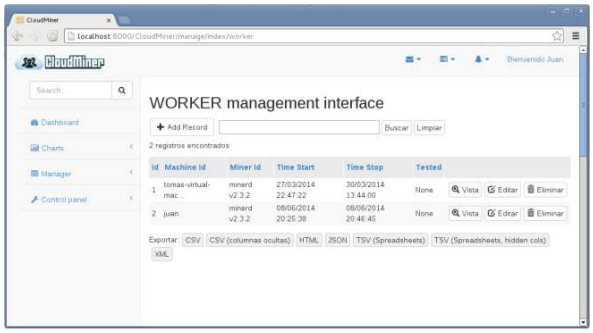
Platform groups



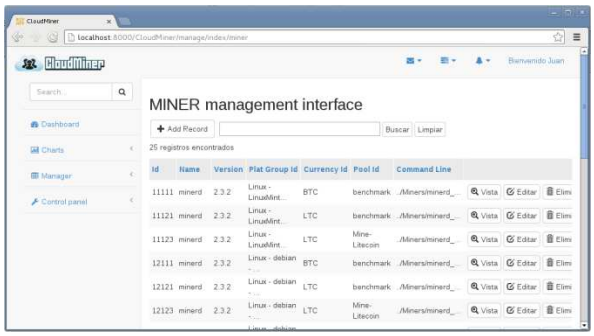
Machines



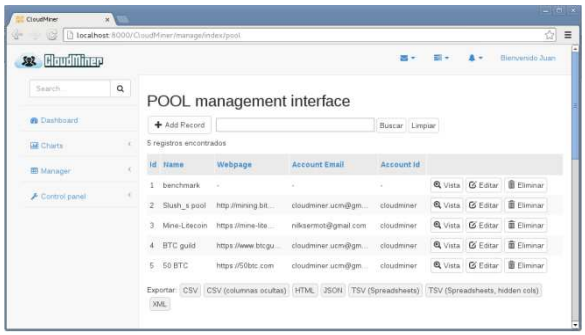
Workers



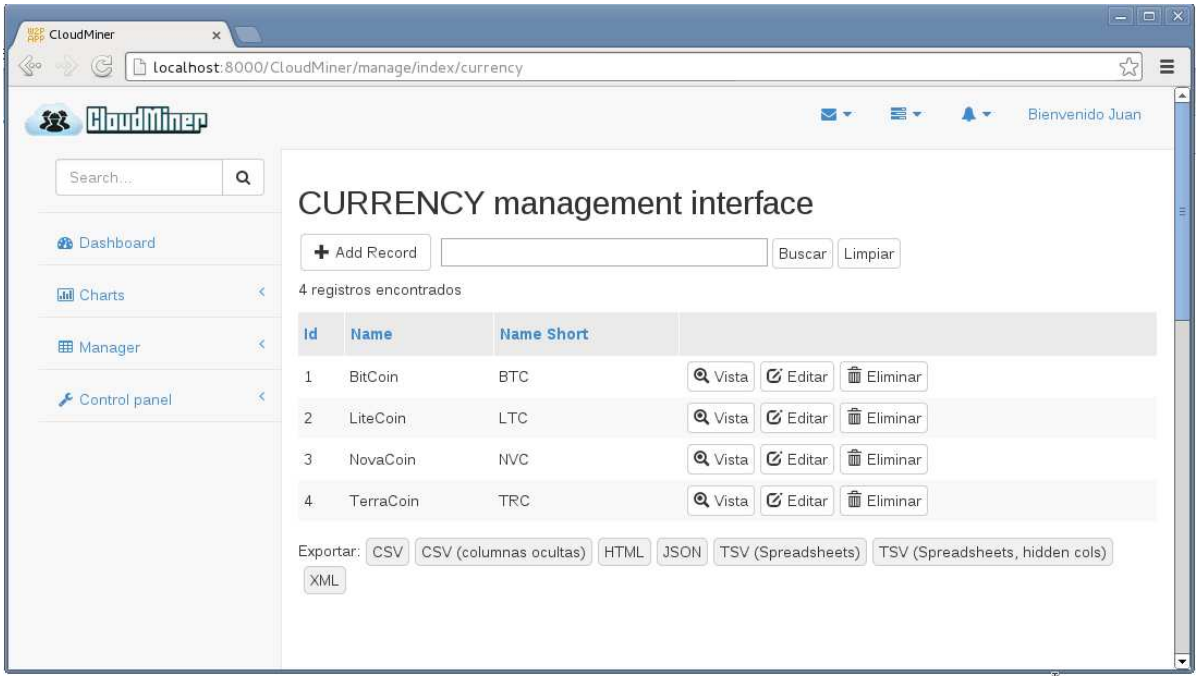
Miners



Pools



Currencies

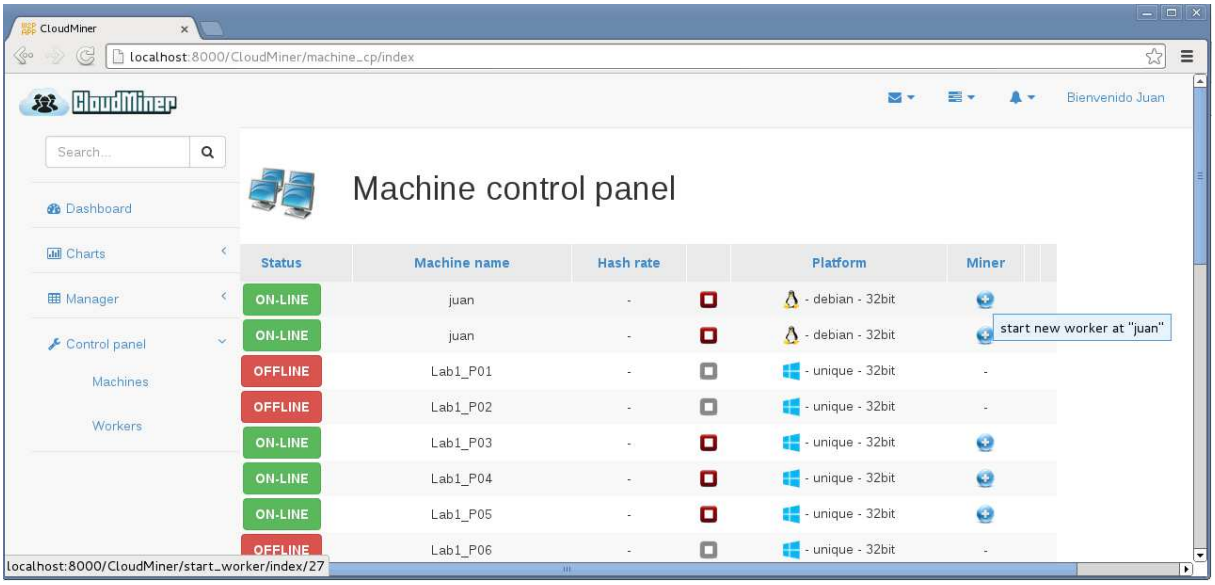


6.4.5. Panel de control de las máquinas

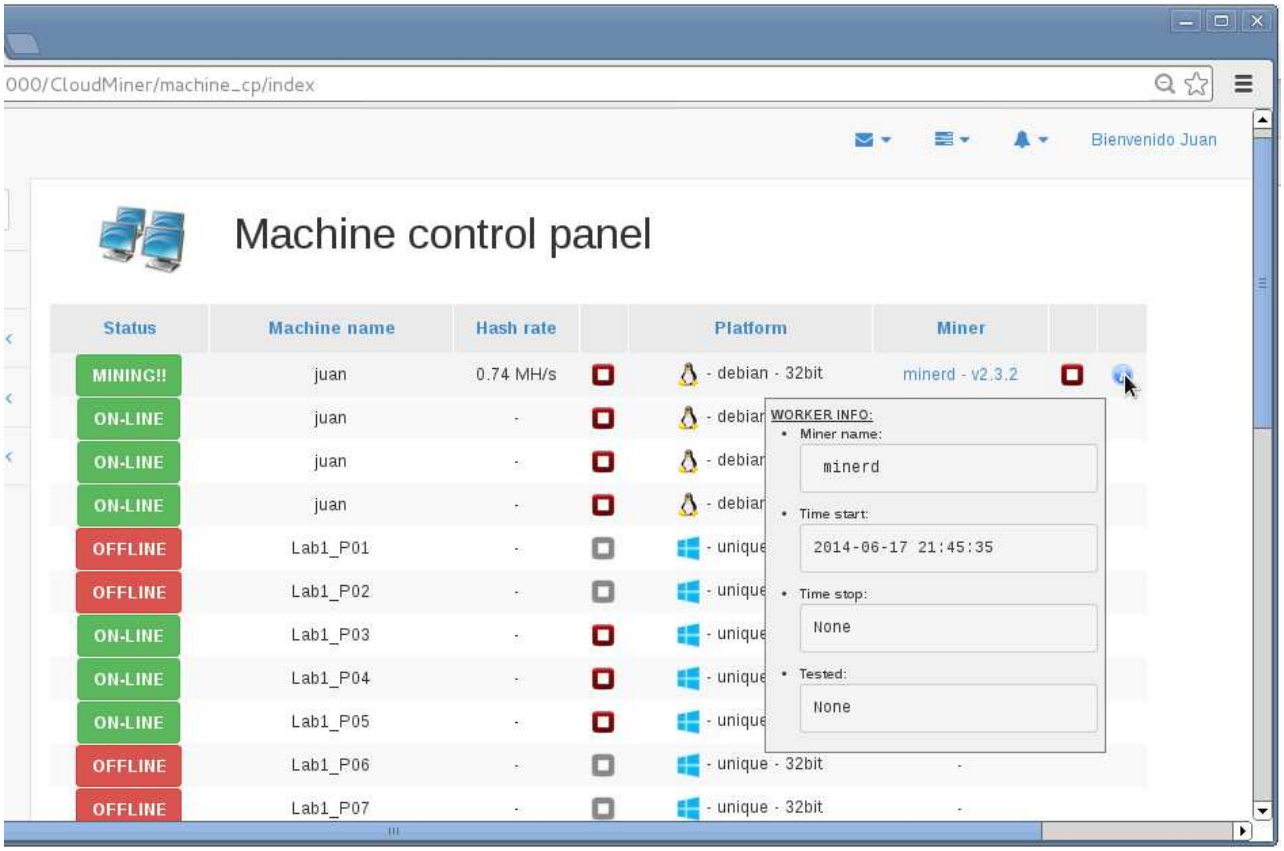
Antes de arrancar un miner

Este panel de control de máquinas, es donde el usuario puede:

- Arrancar worker haciendo clic en el botón
- Parar worker pulsando en el rojo de stop minerd - v2.3.2
- Ver el estado de las máquinas (ON-LINE, OFFLINE, MINING!!)
- Parar máquina
- Arrancar máquina



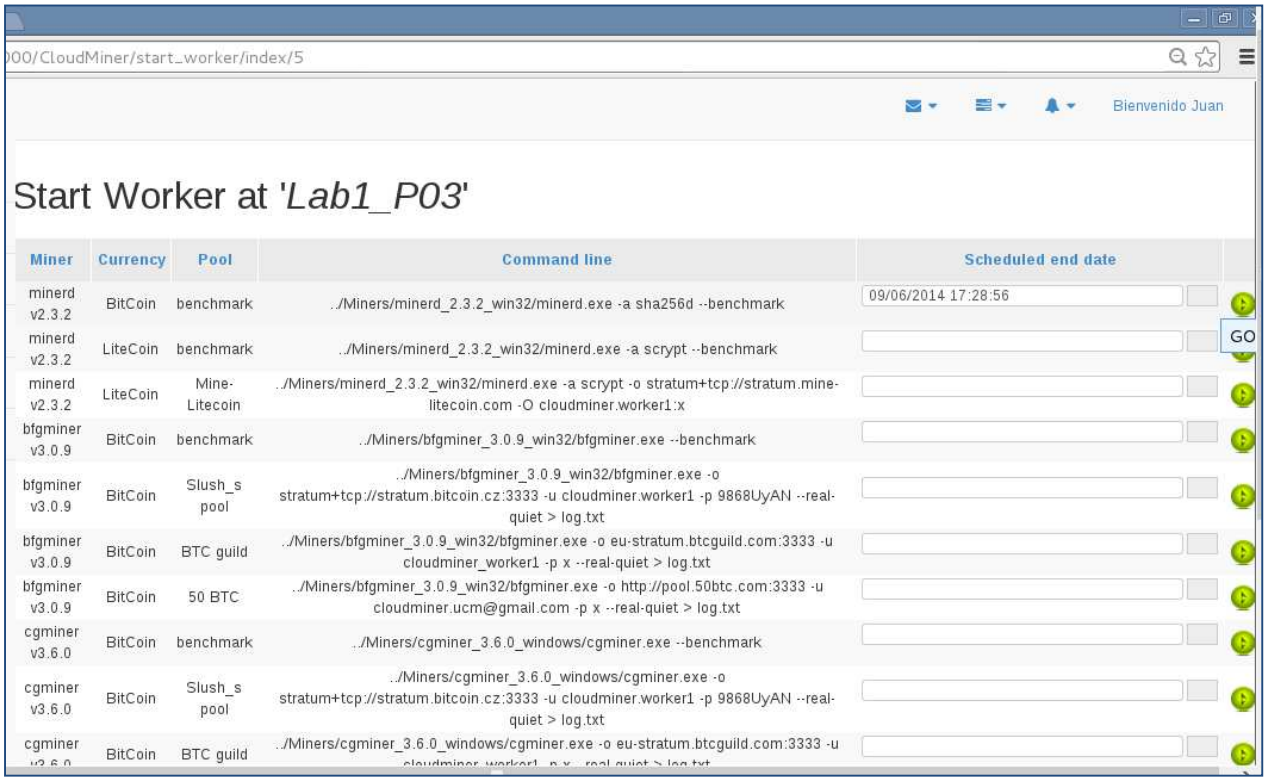
Una vez que se ha arrancado un worker y miner, el usuario puede realizar cualquiera de las acciones anteriormente indicadas y además puede consultar la información detallada de los workers activos pasando el puntero del mouse sobre el símbolo del worker correspondiente.



6.4.6. Pantalla de arranque de los workers

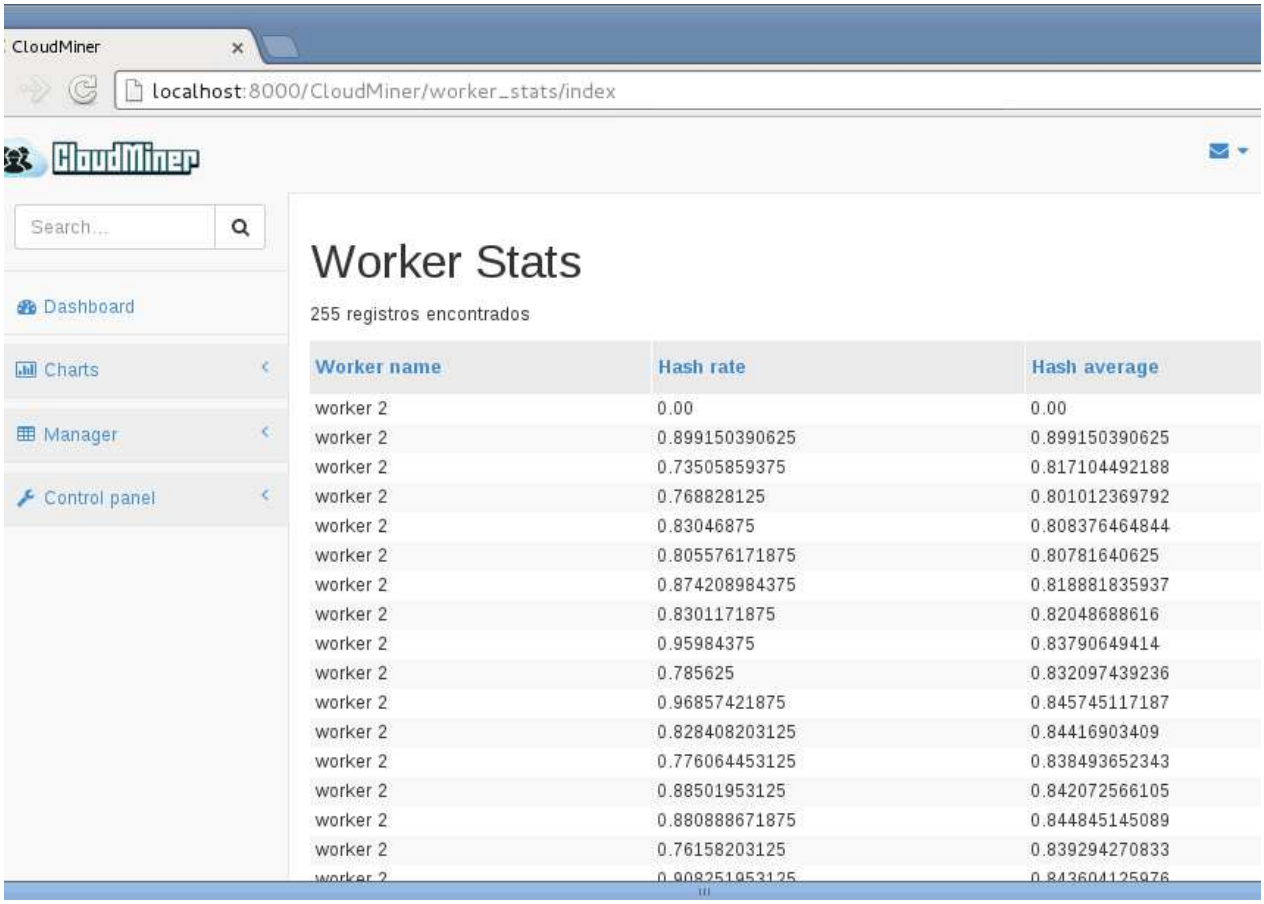
El usuario puede planificar la fecha de finalización de un worker, para ellos se introduce la fecha en el campo "*Schedule end date*" como se muestra en la siguiente figura, una vez programada la fecha de finalización para el worker que se vaya a arrancar, pulsar sobre el botón *start* ( )

EL usuario si quiere puede omitir la fecha de finalización del worker y arrancar directamente worker presionando sobre el botón *start* ( )



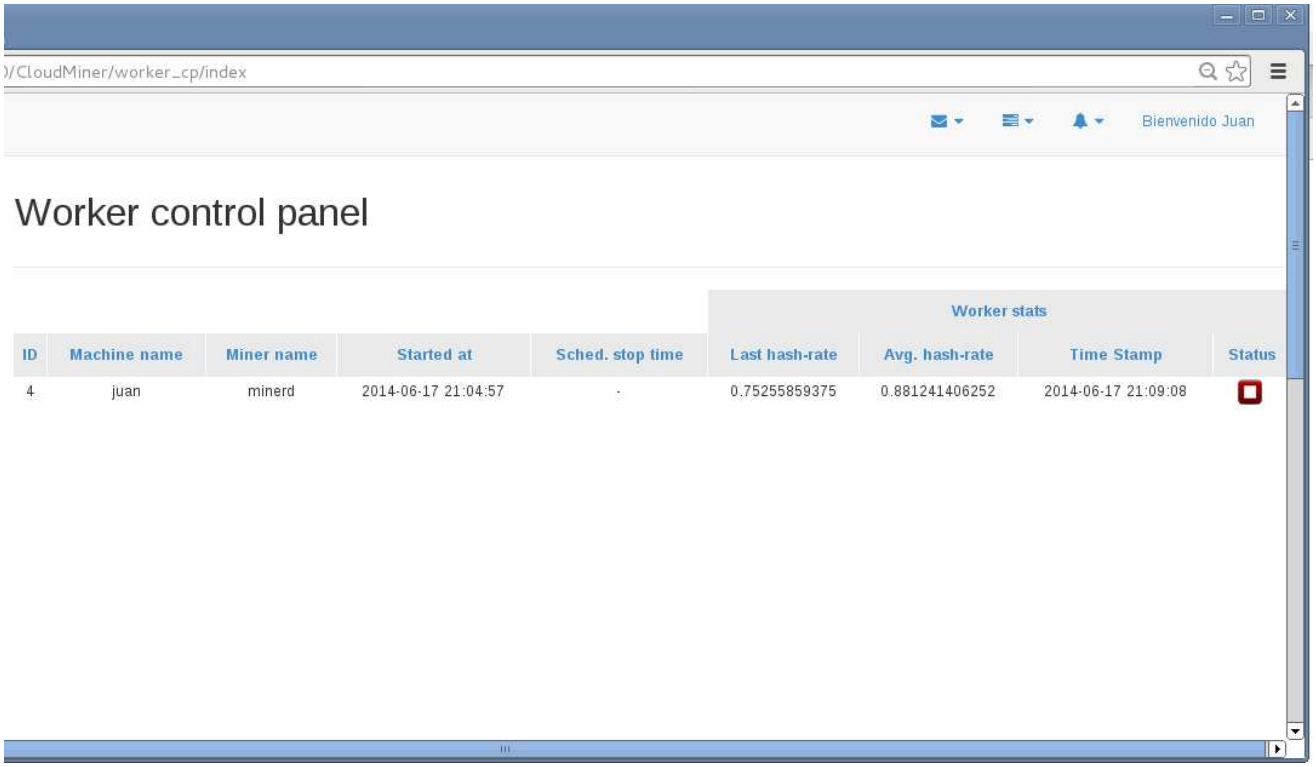
6.4.7. Estadísticas de los workers

Esta es la interfaz donde la aplicación muestra al usuario las estadísticas de los workers.



6.4.8. Workers activos

Esta interfaz de panel de control de los workers está dedicado para que muestre al usuario toda la información sobre el worker en marcha. El usuario puede parar el worker en cualquier momento presionando sobre el botón *stop* ( )



7. Conclusiones

Desde los inicios de este proyecto cuando se comenzó a planear y a lo largo de todo su desarrollo (en total algo más de un año), hemos aprendido mucho en un área de la informática que hasta ahora era para nosotros una incógnita: *Cloud Computing*.

Hemos tenido también la oportunidad, así como el reto, de aprender varios lenguajes nuevos de programación (*Python, Javascript*). Esto, además de ser enriquecedor desde el punto de vista didáctico, lo consideramos un punto esencial en la completitud de nuestras carreras profesionales.

Queremos presentar ahora un resumen de las dificultades que se nos han presentado, así como una consideración de los posibles trabajos futuros y ampliaciones posibles de este proyecto.

7.1. Trabajos futuros y posibles mejoras

7.1.1. Agrupación máquinas

Consideremos el siguiente escenario: tenemos un conjunto de ordenadores de idénticas características (principalmente SSOO, CPU y GPU). Bien pueden ser los equipos de aula informática (habitualmente de 15 – 20) o incluso aunando 'virtualmente' 2 o más aulas de laboratorio. Resultaría muy práctico poder desplegar un mismo *miner* en todos estos ordenadores a la vez, dado que el mismo comando funciona en cualquiera de ellos.

Lo anterior se podría extender al resto de la funcionalidad de la plataforma:

- Añadir un nuevo grupo de ordenadores iguales (un aula de laboratorio, p. ej.)
- Modificar la instalación (SSOO) de un subgrupo de estos.
- Editar grupalmente sus especificaciones técnicas, en caso de que se les cambie algún componente crítico.

7.1.2. Directorio FTP

Sería ideal disponer de un servidor FTP desde donde desplegar los miners. De esta manera, se evitaría el tener que agregarlos al repositorio manualmente. Los nodos tendrían que tener la capacidad de recibirlos usando un comando de autenticación tipo SH o similar.

7.1.3. Demonio vigía

Un problema con el que nos podemos encontrar a la hora de monitorizar los *miners* desplegados en las máquinas disponibles es al momento de fallar una ejecución, bien por un corte de energía, o por un fallo inesperado en el SSOO.

En este tipo de situación, con el diseño actual de BBDD, el nodo afectado quedaría en estado "*zombie*" hasta que se recuperase del error y volviera a ejecutarse el demonio principal.

Para solventar estas circunstancias, se podría desarrollar una aplicación auxiliar, también con conexión a la BBDD, que se encargase de vigilar si hay alguna máquina en este estado, y actuando para devolverla a un estado consistente, y preservar así la coherencia completa del sistema.

7.1.4. Módulo de IA

Otro de los objetivos "opcionales" planteados en la fase de diseño del proyecto, era el incluir un módulo de automatización a la aplicación. Por diversos motivos, el principal de ellos el tiempo disponible, no se pudo llevar a cabo.

Dotado de un cierto nivel de independencia, este módulo permitiría llevar a cabo toma de decisiones relativas a casos comunes como:

- Cuando desplegar un miner,
- En qué máquina iniciarlo,
- Que cripto-moneda utilizar,
- Cuando detenerlo

Para poder tomar estas decisiones, el módulo posiblemente necesitaría también de una cantidad elevada de información relevante, como la información actual relativa a la cotización de las cripto-monedas y la tendencia del mercado relativo a las mismas.

7.1.5. Detección dinámica de recursos disponibles

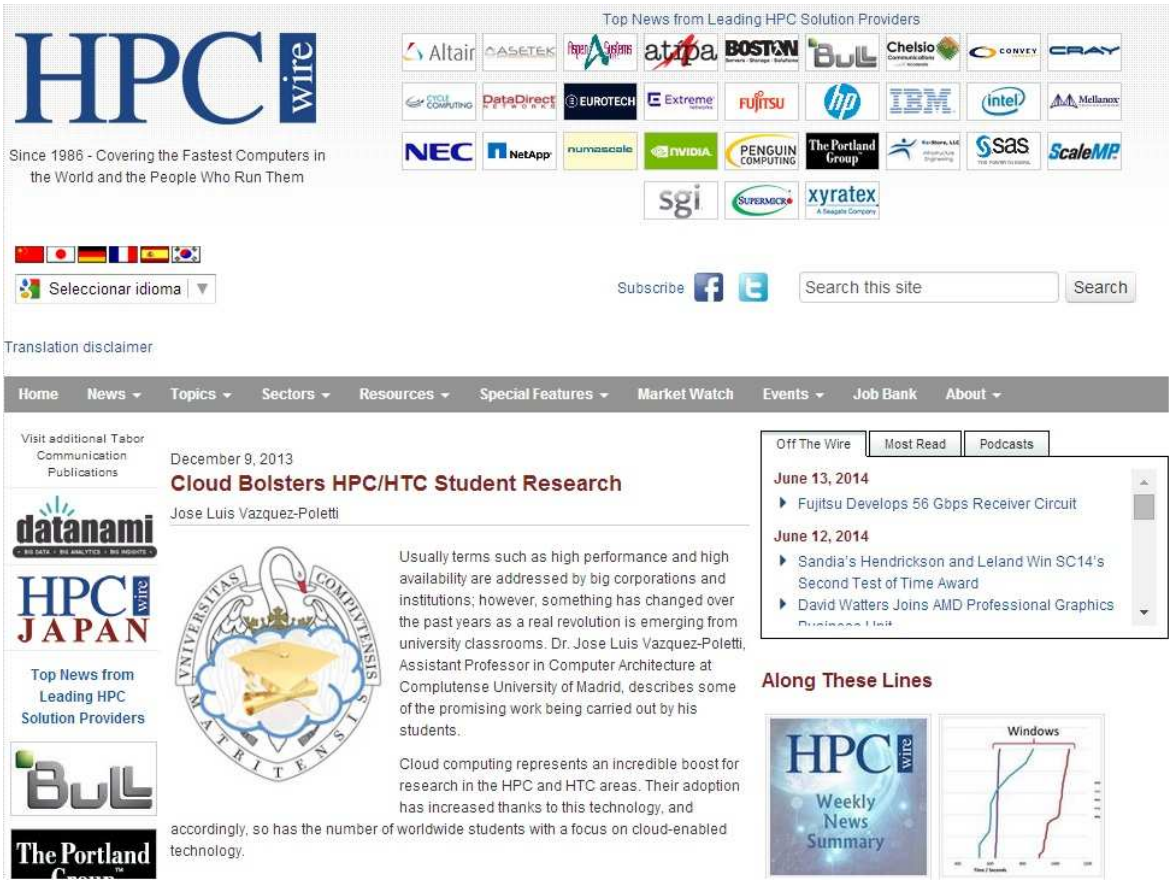
Nosotros desde un principio hemos asumido que las máquinas que aparecen como "disponibles" en el panel de control son realmente recursos de los que se puede disponer por completo. Esto es correcto fuera de horario académico, donde realmente están en desuso dichos ordenadores.

Ahora bien, si se quisiera optimizar al máximo el uso de la aplicación, se podría añadir una funcionalidad que detectara cuando un ordenador está siendo utilizado por un alumno, y marcarlo entonces como "recurso no disponible". Adicionalmente, si un ordenador está siendo utilizado por la plataforma (en horario académico), se podría detectar también un intento de inicio de sesión, deteniendo entonces el minado en dicha máquina.

8. Repercusión

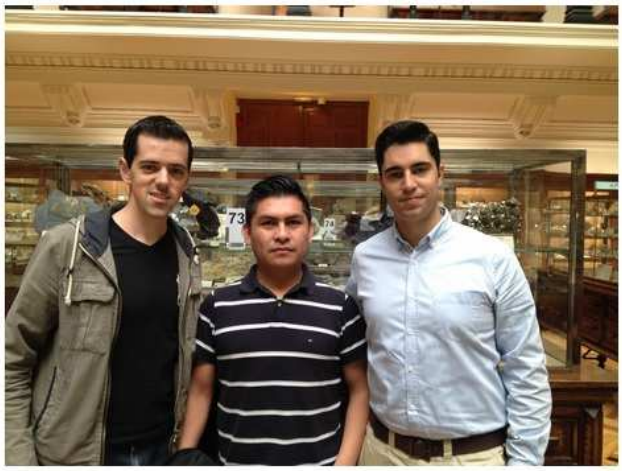
8.1. Artículo en HPC Wire

(Ver artículo completo en http://www.hpcwire.com/2013/12/09/cloud-bolsters-hpchtc-student-research/)



CloudMiner (2013-2014, ongoing)

Nobody would argue that mining bitcoins or one of the many other digital currencies is a great example of HPC gone monetary. Problems arise when having heterogeneous mining rigs and non-monitored maintenance costs. This challenge was taken up by Tomás Restrepo, Juan Arratia and Arturo Pareja, all of them doing their Master Thesis in Computer Science.



CloudMiner team: Tomás Restrepo, Juan Arratia and Arturo Pareja at Madrid's Mining Engineering Museum.

CloudMiner is a project that aims for a better exploitation of the existing hardware, achieving economical benefit through the mining of virtual crypto-currencies. The main idea is to create a cloud computing resource pool, composed by diverse machines under potentially different architectures. This pool is to be monitored in real-time, enabling the user to start/stop mining at any given point on any of the available machines, also providing information on their current status. Additional options are also possible, like adding or removing resources (supported architectures). Artificial Intelligence based on statistics can be used in order to allow automated control of the mining cloud. These statistics are also visible to the user, to aid decision-making.

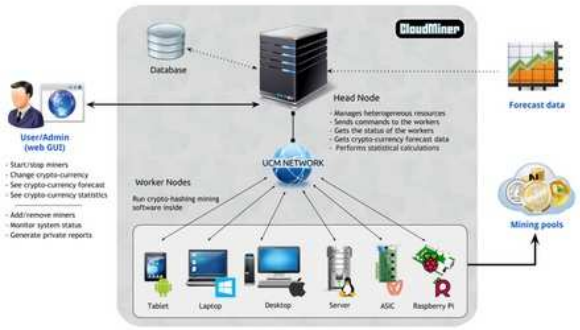


Diagram showing CloudMiner features.

9. Bibliografía

1. Morán, David Ramírez (analista IEEE). Fundamentos de las Divisas Virtuales: BitCoin. 2014.

2. BitCoin.org (Satoshi Nakamoto Whitepaper). [En línea] https://bitcoin.org/bitcoin.pdf.

3. Inteco.es. [En línea] http://www.inteco.es/blogs/post/Seguridad/BlogSeguridad/Articulo_y_comentarios/bitcoin_funcionamiento.

4. Criptografía Asimétrica (Universidad de Comillas). [En línea] http://www.iit.upcomillas.es/palacios/seguridad/cap05.pdf.

5. NIST. [En línea] http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

6. Gens, F. Defining "Cloud services" and "Cloud computing". [En línea] http://blogs.idc.com/ie/?p=190.

7. Murray, A. C. There's no such thing as a private cloud. [En línea] http://www.informationweek.com/cloud-computing/theres-no-such-thing-as-a-private-cloud/229207922.

8. Haff, G. Just don't call them private clouds. [En línea] Enero de 2009. http://news.cnet.com/8301-13556_3-10150841-61.html.

9. Amazon EC2. [En línea] http://aws.amazon.com/es/ec2/.

10. McFedries, P. The Cloud Is The Computer. [En línea] http://spectrum.ieee.org/computing/hardware/the-cloud-is-the-computer.

11. Hurwitz, Judith. *Cloud Computing for Dummies.* s.l. : Wiley Pub, 2010. 9780470484708.

12. Prashnar, Vinay. Ten Must Have Bitcoin Related iPhone Apps. [En línea] http://www.btcpedia.com/bitcoin-related-iphone-apps/.

13. BOINC. [En línea] http://boinc.berkeley.edu/.

14. RFC 793. [En línea] http://www.rfc-es.org/rfc/rfc0793-es.txt.

15. RFC 2663. [En línea] http://www.rfc-es.org/rfc/rfc2663-es.txt.

16. Patrón Command (UAM). [En línea] http://arantxa.ii.uam.es/~eguerra/docencia/0708/09%20Command.pdf.

17. The Callback Design Pattern. [En línea] http://people.bu.edu/azs/teaching/cs108/2006fall/callback_pattern.pdf.

18. [En línea] https://docs.python.org/2/license.html.

19. Web2py. [En línea] http://www.web2py.com.

20. Web2py (manual de referencia). [En línea] http://web2py.com/books/default/chapter/41.

21. [En línea] https://docs.python.org/2/library/dis.html.

22. [En línea] http://www.web2py.com/AlterEgo/default/show/76.

23. [En línea] http://www.web2py.com/AlterEgo/default/show/215.

24. SB Admin v2.0. [En línea] http://startbootstrap.com/sb-admin-v2.

25. Django. [En línea] https://www.djangoproject.com/.

26. Djangoproject.com (Django 1.6 overview). [En línea] https://docs.djangoproject.com/en/1.6/intro/overview/.

27. www.web2py.com.ar. [En línea] http://www.web2py.com.ar/wiki/default/_page/Web2py_para_todos1.

28. SB Admin v2.0. [En línea] http://startbootstrap.com/sb-admin-v2.

29. SB Admin v2.0. [En línea] http://startbootstrap.com/sb-admin-v2.

30. Barrios Hernández, Luis and Fernández Hernández, Adrián and Guayerbas Martín, Samuel. *CygnusCloud : provisión de puestos de laboratorio virtuales bajo demanda.* Madrid : UCM, 2013.