Universitat Oberta de Catalunya

uoc.edu

Learning analytics en cursos sobre TICS utilizando Moodle, Spark y Jupyter notebooks

Iñaki Pérez García
Posgrado en Sistemas de Información para Inteligencia de Negocio y Big Data
Área de Big Data y Sistemas NoSQL

Francesc Julbe
Josep Curto

25/06/2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Gamificación y learning analytics en cursos sobre TICS utilizando Moodle y Jupyter notebooks*
Nombre del autor:	*Iñaki Pérez García*
Nombre del consultor/a:	*Francesc Julbe López*
Nombre del PRA:	*Josep Curto Díaz*
Fecha de entrega (mm/aaaa):	06/2018
Titulación:	*Sistemas de información de Inteligencia de Negocio y Big Data*
Área del Trabajo Final:	*Sistemas de Información y Big Data*
Idioma del trabajo:	***Castellano***
Palabras clave	*eLearning, Big Data, Data Warehouse*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

El presente proyecto se plantea con un doble objetivo. Por un lado se quiere crear una plataforma tecnológica de eLearning necesaria para atender las necesidades de un proyecto emprendedor real. En concreto se trata de un club de robótica y programación, a modo de actividades extraescolares, en el que los alumnos tendrán los contenidos formativos a su disposición a través de dicha plataforma. Por otro lado se quiere aprovechar la información generada por los alumnos para crear un almacén de datos y posteriormente generar modelos de análisis que permitan mejorar la calidad de los contenidos formativos de la plataforma.

El proyecto consta de dos partes bien diferenciadas. En primer lugar se ha creado la arquitectura de la plataforma de eLearning. Cabe destacar la configuración de los diversos servicios en modo cluster, que permitirá una futura escalabilidad del sistema. Las tecnologías utilizadas en esta parte son Moodle, PostgreSQL, Spark, Jupyter notebooks y MongoDB. La segunda parte consiste en la explotación de los datos, para lo cual se recurre al servidor Pentaho, se crea un almacén de datos con Spoon y se diseña un cubo de análisis multidimensional con Schema Workench que finalmente se analiza con Saiku Analytics.

El resultado ha sido satisfactorio, ya que se ha conseguido integrar todas estas tecnologías correctamente, lo que ha permitido generar varios ejemplos de utilización y prototipos de análisis. A partir de este punto, será necesario como trabajo futuro, incluir contenidos en la plataforma para poner el sistema en producción y atender a alumnos reales.

Abstract (in English, 250 words or less):

The present project is proposed with two goals. On the one hand we want to create an eLearning technology platform necessary to meet the needs of a real startup project. It is about a robotics and programming club, offered as extracurricular activities, in which the students will have the training content available through that platform. On the other hand, we want to take advantage of the information generated by the students to create a data warehouse and then generate analysis models that allow us to improve the quality of the training contents of the platform.

The project consists of two well differentiated parts. First of all, we have worked in the architecture. It is worth highlighting that the configuration of the services in cluster mode will allow the future scalability of the system. The technologies used in this part are Moodle, PostgreSQL, Spark, Jupyter notebooks and MongoDB. The second part consists in the exploitation of the data. We have used Pentaho server and a data warehouse that has been created with Spoon, then we have designed a multidimensional analysis cube with Schema Workench and finally we have analyzed it with Saiku Analytics.

The result has been satisfactory, since we have achieved to integrate all these technologies correctly, which has allowed us to generate several examples of use and prototypes of analysis. From this point, it will be necessary, as future work, to include contents in the platform to put the system into production and serve real students.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Se parte de la necesidad de crear una plataforma que posibilite el análisis del trabajo realizado por los estudiantes de cursos relacionados con las TIC en entornos de eLearning.

Hasta el momento existen plataformas abiertas de eLearning como Moodle, que permiten gestionar grupos de alumnos así como crear cursos online e incluso colaborar en ellos en tiempo real.

También hay gran cantidad de plataformas que ofrecen cursos MOOC (Masive Open Online Course) y además, cuando se trata de materias técnicas, como por ejemplo lenguajes de programación, bases de datos, inteligencia artificial, etc. disponemos de entornos computacionales interactivos, como Databricks, Code Anywhere, Atom, etc.

El presente proyecto surge de la necesidad de disponer de una plataforma centralizada desde la que organizar cursos y administrar usuarios/alumnos y, a la vez, disponer de una herramienta que permita enseñar materias TIC de forma interactiva además de poder analizar los resultados y la evolución del aprendizaje de dichos alumnos.

De momento no existe una plataforma abierta que permita disponer de ambas funcionalidades al mismo tiempo, por lo que a día de hoy, o cubrimos una de ellas con una plataforma de terceros sin tener la ventaja de la gestión personalizada, o prescindimos de la capacidad de análisis de resultados y de la interactividad.

En este trabajo se pretende integrar varios frameworks para ofrecer a los alumnos un único portal desde el que acceder a todas las herramientas de estudio, y visto desde el lado del profesorado disponer de un sistema de gestión y análisis que permita optimizar los cursos y aumentar su productividad.

Por último, además de la integración de las plataformas, que sería un primer trabajo de administración de sistemas, se añadirá al sistema un almacén de datos formado por un lado por una base de datos relacional y por otro lado por una base de datos NoSQL basada en documentos. Esta sería la aportación en cuanto a Big Data y sistemas NoSQL necesarios para realizar learning analitics. Como trabajo futuro, este Datawarehouse facilitará mediante técnicas de gamificación, la evaluación y mejora del aprendizaje de los usuarios del sistema.

1.2 Objetivos del Trabajo

El objetivo de este trabajo es el diseño e implementación de un sistema de Big Data que facilite la adquisición, el almacenamiento y la explotación de datos de un entorno de eLearning.

Los objetivos parciales serían los siguientes:

1. Instalar un servidor Moodle en modo cluster y gamificar un curso de programación.
2. Instalar un servidor Spark en cluster utilizando OpenStack sobre el que configurar y ejecutar Jupyter notebook con acceso multiusuario.
3. Integrar los notebooks en Moodle.
4. En el cluster de Moodle instalar un servidor PostgreSQL.
5. En el cluster Openstack instalar un servidor MongoDB.
6. Crear una datawarehouse entre ambas bases de datos sobre el que analizar la evolución del aprendizaje de los estudiantes.
7. Definición de las fuentes de datos, caracterización del modelo de datos y desarrollo de los procesos ETL
8. Creación de un prototipo de análisis tipo Word Count o similar.

A continuación se puede observar un ejemplo del resultado deseado, con un notebook dentro de un curso de moodle.

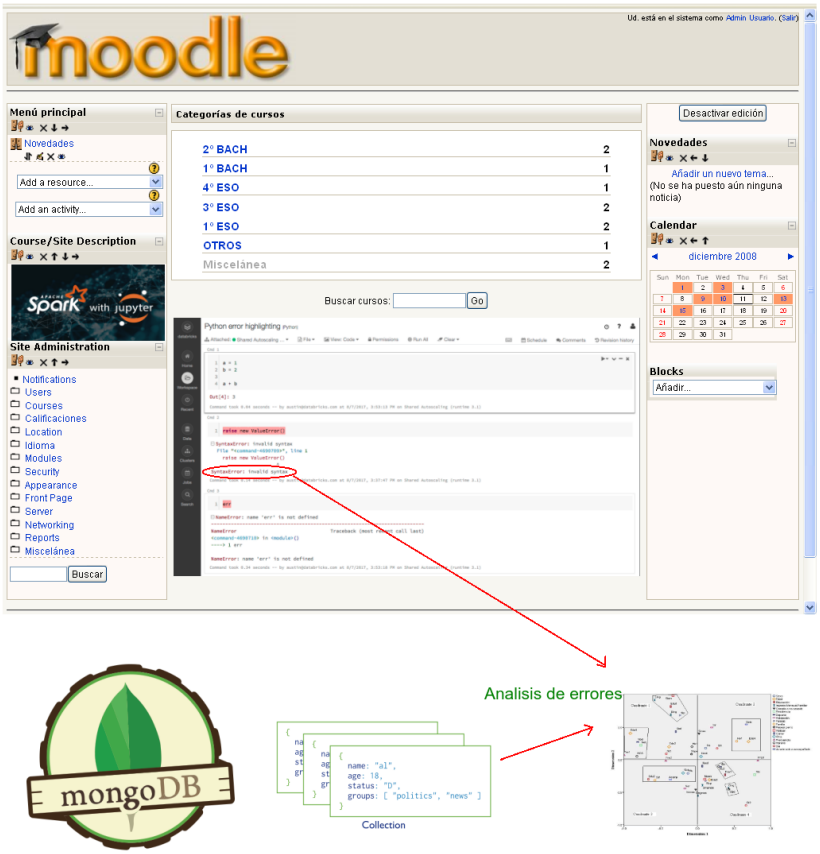


Fig. 1. Funcionalidad deseada con el proyecto.

1.3 Enfoque y método seguido

El enfoque del trabajo es del tipo Profesionalizador/Empresarial. Se pretende crear una microempresa real a partir de los resultados del mismo, por lo que el producto será el facilitador del inicio de una etapa emprendedora. En concreto se pretende crear un club de programación y robótica a modo de extraescolares para alumnos de 7 a 17 años.

En este club se promoverá el uso de software y hardware libre, por lo tanto en el proyecto también se utilizará en todo momento open software. La plataforma que se pretende crear es algo ambiciosa, por lo tanto la metodología a seguir será la de adaptar productos ya existentes apoyándome en la gran cantidad de documentación publicada en Internet gracias a la gran comunidad que existe alrededor del ecosistema que se quiere utilizar y que incluye las siguientes tecnologías entre otras: Apache Spark, Postgres, MongoDB, Moodle, Pentaho.

Esta estrategia de adaptación del software libre existente es más adecuada dada la complejidad que supondría crear todo este entorno desde cero y también por el escaso tiempo disponible. Además se trata de un proyecto que se desarrolla de forma totalmente individual por lo que los recursos son limitados.

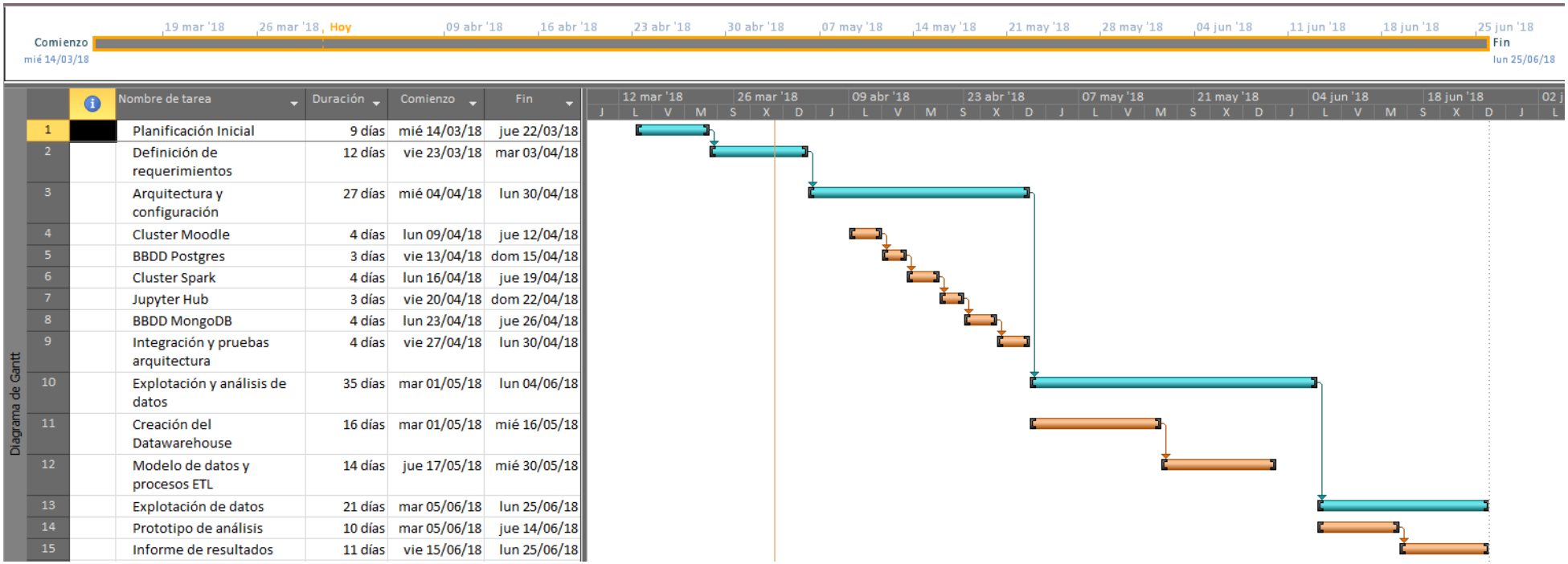
1.4 Planificación del Trabajo

Se disponen de los siguientes recursos materiales para realizar el proyecto en un entorno lo más real posible. Si el proyecto es viable, habrá que mejorar los recursos hardware para asegurar la disponibilidad de la plataforma en un entorno de producción, pero de momento las especificaciones básicas de las computadoras disponibles para realizar el prototipo son las siguientes:

- 1 equipo con procesador 8x AMD FX 4 GHz y 32 GB de RAM.
- 1 equipo con procesador 2x AMD 3,2 GHz y .4 GB de RAM.
- 1 equipo con procesador 2x Intel core i7 3 GHz y 8 GB de RAM.

Por otro lado, se muestra a continuación un listado de las tareas a realizar y una planificación temporal de las mismas mediante un diagrama de Gantt.

Diagrama de Gantt



	Nombre de tarea	Duración	Comienzo	Fin
1	Planificación Inicial	9 días	mié 14/03/18	jue 22/03/18
2	Definición de requerimientos	12 días	vie 23/03/18	mar 03/04/18
3	Arquitectura y configuración	27 días	mié 04/04/18	lun 30/04/18
4	Cluster Moodle	4 días	lun 09/04/18	jue 12/04/18
5	BBDD Postgres	3 días	vie 13/04/18	dom 15/04/18
6	Cluster Spark	4 días	lun 16/04/18	jue 19/04/18
7	Jupyter Hub	3 días	vie 20/04/18	dom 22/04/18
8	BBDD MongoDB	4 días	lun 23/04/18	jue 26/04/18
9	Integración y pruebas arquitectura	4 días	vie 27/04/18	lun 30/04/18
10	Explotación y análisis de datos	35 días	mar 01/05/18	lun 04/06/18
11	Creación del Datawarehouse	16 días	mar 01/05/18	mié 16/05/18
12	Modelo de datos y procesos ETL	14 días	jue 17/05/18	mié 30/05/18
13	Explotación de datos	21 días	mar 05/06/18	lun 25/06/18
14	Prototipo de análisis	10 días	mar 05/06/18	jue 14/06/18
15	Informe de resultados	11 días	vie 15/06/18	lun 25/06/18

Fig. 2. Diagrama de Gantt

1.5 Breve sumario de productos obtenidos

Las soluciones software obtenidas serían las siguientes:

- Plataforma de eLearning Moodle en modo cluster.
- Plataforma Spark + Jupyter como entorno de programación interactivo.
- Datawarehouse formado por la base de datos relacional Postgres y la base de datos NoSQL MongoDB.

Todos estos productos estarán integrados para formar un único portal de elearning con capacidades de análisis de la actividad de los usuarios y la evaluación y seguimiento de su aprendizaje.

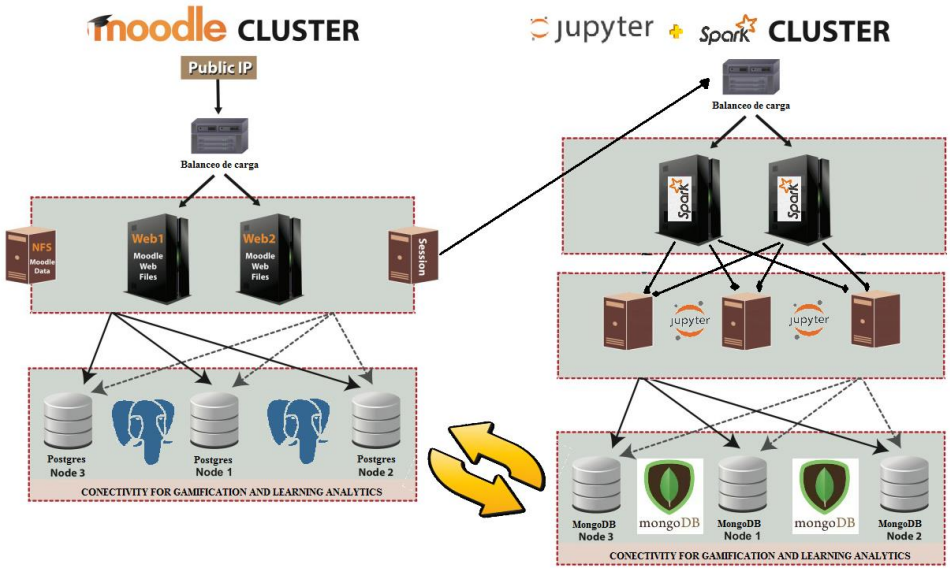


Fig. 3. Diagrama conceptual de la arquitectura objetivo.

1.6 Breve descripción de los otros capítulos de la memoria

Parte 1: Diseño y configuración de la plataforma de eLearning

- Capítulo 2. Arquitectura del sistema de administración de cursos.
- Capítulo 3. Arquitectura del sistema de programación interactivo.
- Capítulo 4. Escenario de análisis. Simulación de datos.

Parte 2: Análisis y explotación de los datos generados por los alumnos

- Capítulo 5. Modelo de datos y procesos ETL.
- Capítulo 6. Creación del Datawarehouse.
- Capítulo 7. Prototipo de análisis multidimensional. Cubo OLAP con Mondrian.
- Capítulo 8. Trabajo futuro y conclusiones.
- Capítulo 9. Bibliografía
- Capítulo 10. Anexos

Parte 1: Desarrollo de una plataforma de eLearning y un entorno de programación interactivo.

La plataforma de eLearning propuesta tiene dos partes bastante diferenciadas que posteriormente habrá que integrar. Por un lado, tenemos un sistema gestor de cursos, alumnos y contenidos, compuesto por Moodle y por la base de datos PostgreSQL.

Por otro lado, tenemos un entorno de programación interactivo que tiene tres subsistemas, Spark, que es un entorno de computación distribuida, Jupyter, que es un servidor de notebooks que permiten programar de forma interactiva, y por último, también en este caso tenemos una base de datos basada en documentos, MongoDB.

En ambos casos se pretende implementar los distintos servidores sobre un cluster de varios equipos, de este modo tendremos un sistema que a futuro será escalable.

Esta parte del proyecto tiene que ver con la administración de sistemas. Hay una gran cantidad de configuraciones, ejecución de comandos e instalaciones que se incluyen como anexos dada su extensión. En los tres capítulos de esta parte se describe el proceso que está replicado en detalle en dichos anexos.

2. Arquitectura del sistema de administración de cursos.

Podría parecer que la primera de las herramientas que se van a instalar en un entorno de eLearning es el servidor Moodle, sin embargo, para el correcto funcionamiento del mismo, tenemos que tener previamente un sistema gestor de bases de datos, que en este caso será PostgreSQL.

PostgreSQL es una base de datos relacional de código abierto que se distribuye bajo licencia BSD. Dispone de versiones para prácticamente todos los sistemas operativos y cumple totalmente con ACID.

PostgreSQL ofrece características tales como replicación asíncrona y write ahead logging para ser tolerante a fallos de hardware y del que se hace uso en este proyecto. Soporta juegos de caracteres internacionales, codificaciones de caracteres multibyte, Unicode y es altamente escalable tanto en la cantidad bruta de datos que puede manejar como en el número de usuarios concurrentes que puede atender. Hay sistemas activos en producción con PostgreSQL que manejan más de 4 terabytes de datos.

2.1 Instalación y configuración del cluster PostgreSQL.

2.1.1 Instalación y configuración de Turnkey Linux con imagen Postgresql

Turnkey Linux es un proyecto a través del cual se distribuyen dispositivos virtuales libre, que permiten la implementación (llave en mano) de plataformas tecnológicas en máquinas virtuales, en pocos minutos. En este caso se ha optado por una imagen que incluye un servidor Postgresql en su versión 9.4

Se han instalado dos servidores postgresql en dos máquinas virtuales para configurar el cluster, la réplica de datos y el balanceo de carga y que en el futuro el sistema sea escalable pudiendo añadir más servidores fácilmente.

2.1.2 Configuración de Postgres en modo cluster y con replicado

Para que los dos servidores funcionen en modo cluster es necesario hacer ciertas modificaciones en varios archivos de configuración. Además es necesario instalar el programa pgpool-ii. Se trata de un middleware que trabaja entre un cliente PostgreSQL y los servidores, ofreciendo al usuario la sensación de conectarse a un único servidor, y proporcionando herramientas para gestionar el balanceo de cargam la replicación de los datos, etc. Para ello se ha cargado otra imagen de Turnkey, en esta ocasión un sistema básico llamado core, para instalar y configurar este servidor intermedio.

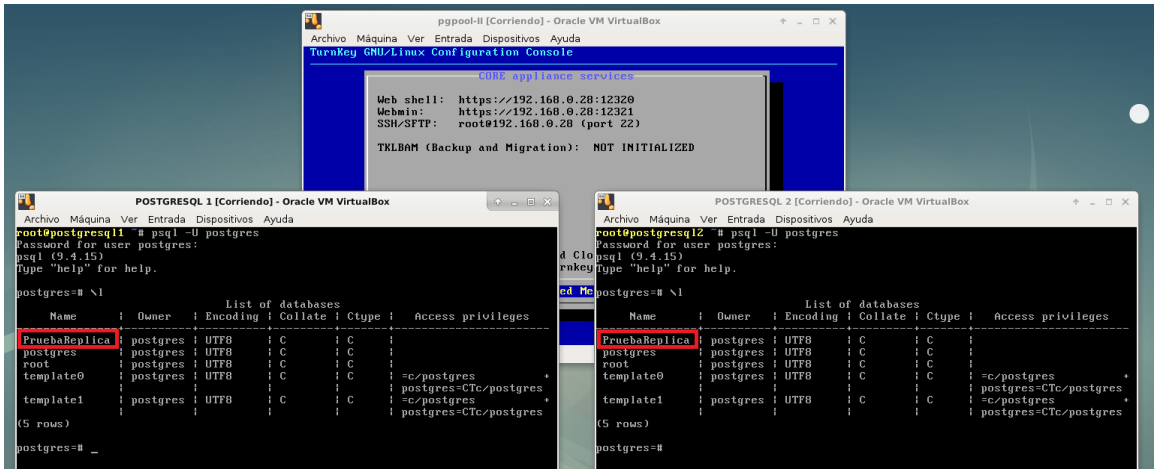


Fig. 4. Comprobación del replicado del cluster PostgreSQL.

2.2 Instalación y configuración del cluster Moodle.

2.2.1 Instalación y configuración de Turnkey Linux con imagen Moodle

Al igual que en el caso de los servidores de bbdd anterior, se han instalado dos servidores Turnkey en dos máquinas virtuales para configurar el cluster, en este caso con una imagen con Moodle preconfigurado.

La imagen de Turnkey Linux viene con Moodle configurada por defecto para trabajar con una base de datos MYSQL. En el anexo se detallan los pasos para crear una nueva base de datos con Postgresql, crear el usuario moodleuser y configurar Moodle para que la utilice.

2.3 Configuración de Moodle con Postgres como base de datos.

En el primero de los arranques, si la conexión con el servidor PostgreSQL es correcta, tendremos que hacer los siguientes pasos, con lo que se creará el esquema en la base de datos con aproximadamente 370 tablas. Una vez realizados los siguientes pasos podremos acceder a nuestro portal Moodle por primera vez:

1) Aceptar condiciones y licencia
2) Comprobar dependencias
3) Creación de la estructura de la base de datos
4) Crear el usuario administrador
5) Último paso, configuración de la página principal

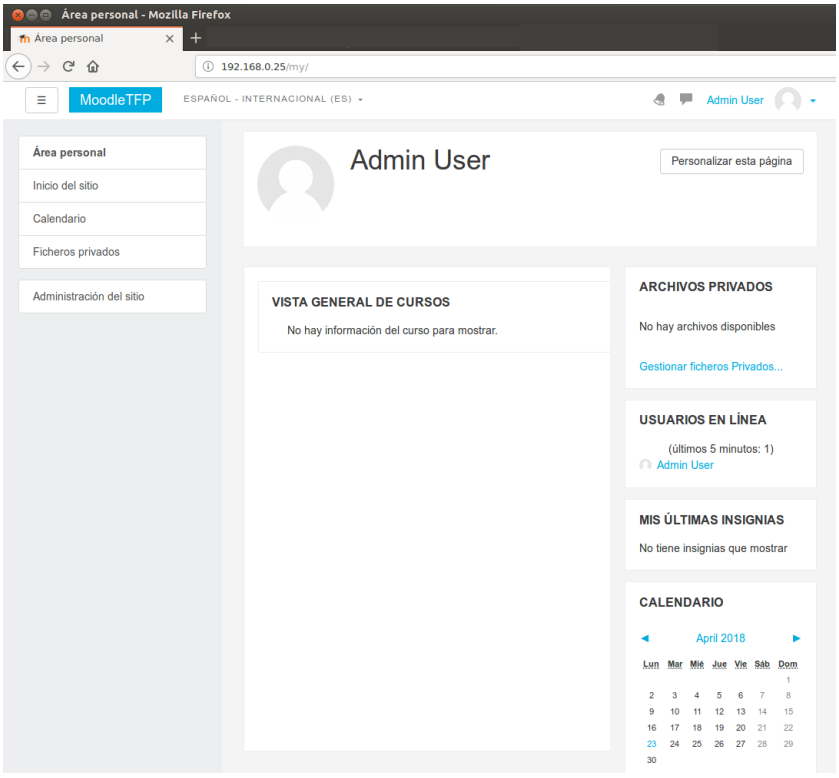


Fig. 5. Página principal del usuario administrador en Moodle.

3. Arquitectura del sistema de programación interactivo.

3.1 Instalación y configuración del cluster Spark.

3.1.1 Instalación y configuración de Ubuntu Server 16.04 y las dependencias

En este caso no existe una imagen preconfigurada en el proyecto Turnkey que incluya el servidor Spark, por lo tanto es necesario instalar en primer lugar el sistema operativo. Se ha elegido Ubuntu Server 16.04.

Las dependencias se refieren por ejemplo a java. Es necesario instalar la máquina virtual o el entorno de desarrollo que también la incluye. Por otro lado es necesario el soporte para el lenguaje Scala. Se trata del lenguaje nativo de Spark, por lo que es fundamental que esté instalado.

Cuando ya tenemos uno de los equipos correctamente configurado procedemos a clonarlo dos veces. Una de las máquinas hará de equipo master y las otras dos de equipos esclavos o workers.

Fig. 6. Clonación de máquinas virtuales.

3.1.2 Instalación y configuración de Apache Spark

Ahora vamos a proceder a instalar Spark tanto en el servidor Master como en los “workers”.

Se puede descargar el código fuente completo de Spark y compilarlo, pero como depende de una instalación base de Hadoop, y existe una versión precompilada que incluye las dos tecnologías, se opta por descargar esta opción, más cómoda y que seguramente generará menos fallos, sólo es necesario decomprimir y listo.

Posteriormente hay que editar la configuración de los archivos $SparkHome/conf/slaves y spark-env.sh. El proceso completo se detalla en los anexos. Para arrancar y parar todo el cluster se utilizan los comandos start-all.sh y stop-all.sh y finalmente accederemos vía web a este entorno de computación distribuida.

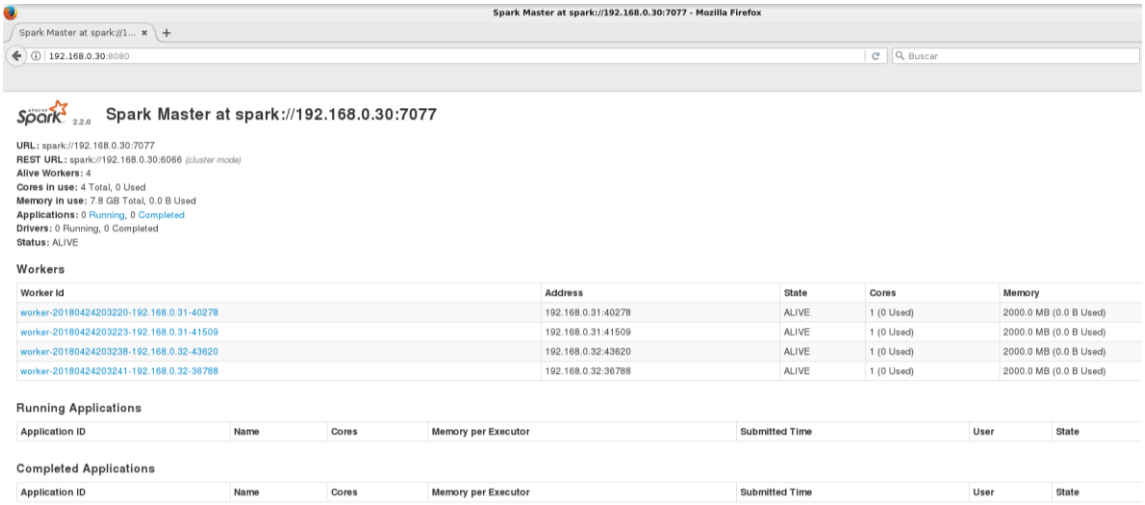


Fig. 7. Comprobación del cluster de computación distribuida.

3.2 Instalación y configuración de Jupyter Notebooks

Jupyter Notebooks es una aplicación web que permite crear y compartir documentos que contienen código fuente, ecuaciones, visualizaciones y texto explicativo. Entre sus usos está la limpieza y transformación de datos, la simulación numérica, el modelado estadístico, el aprendizaje automático y mucho más.

3.2.1 Instalación y configuración de las dependencias en el nodo Master

En este caso no se va a utilizar un cluster independiente de máquinas virtuales para ejecutar Jupyter, sólo es necesario instalar una serie de paquetes en la máquina master del cluster Spark y serán los kernels de los distintos lenguajes los que harán uso del cluster o sólo de una de las máquinas, dependiendo de si disponen o no de capacidad para hacer uso de computación distribuida. Algunos lenguajes que si disponen de esta capacidad son Python y R.

Las dependencias son las siguientes: python3-pip npm nodejs-legacy y también será necesario instalar un servidor http-proxy: configurable-http-proxy.

3.2.2 Instalación y configuración del entorno multiusuario JupyterHub

En nuestro proyecto necesitamos que accedan al servidor gran cantidad de alumnos, por lo que necesitamos que el entorno sea multiusuario.

Gracias al gestor pip3 instalado anteriormente, vamos a descargar y ejecutar tanto JupyterHub como el nucleo de Jupyter, por lo que ya estaremos en condiciones de acceder con el usuario administrador:

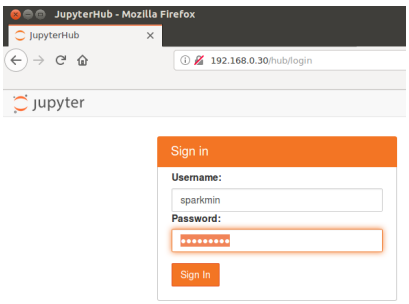


Fig. 8. Inicio de sesión en JupyterHub.

Ahora ya podríamos crear usuarios nuevos e incluso crear algún notebook del lenguaje por defecto que sería Python 3.

3.2.3 Configuración del kernel pyspark y otros lenguajes

En nuestro proyecto queremos que los alumnos aprendan distintos lenguajes de programación, entre ellos el lenguaje de programación python. Python es un lenguaje muy utilizado en BigData para análisis masivo de datos, machine learning, etc ya que tiene capacidades de procesado en paralelo.

Vamos a configurar un kernel llamado pyspark para hacer uso del cluster spark a la hora de ejecutar las aplicaciones escritas en python. El problema es que pyspark es compatible sólo con Python2 por lo que crearemos también la compatibilidad con este lenguaje. Se trata de una configuración bastante complicada que queda detallada en los anexos. Finalmente se prueba que todo funciona correctamente con una aplicación de prueba.

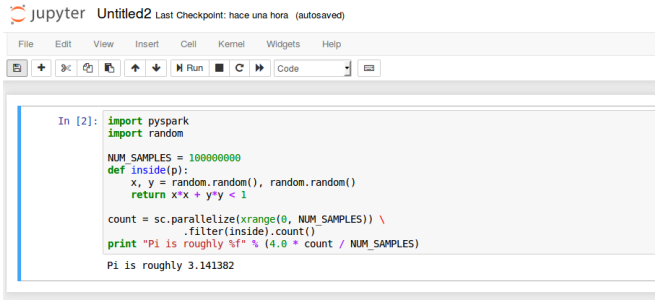


Fig. 9. Código de ejemplo en el kernel pyspark.

A modo de ejemplo se instala también un kernel para el lenguaje de programación javascript:

Fig. 10. Kernel del lenguaje Javascript.

3.3 Instalación y configuración de MongoDB

3.3.1 Instalación y configuración de Turnkey Linux con imagen MongoDB

Al igual que en el caso de los servidores de Postgresql y Moodle, se ha instalado un servidor Turnkey en una máquina virtual con una imagen con MongoDB preconfigurado.

De momento únicamente comprobamos su funcionamiento iniciando sesión como administradores vía web:

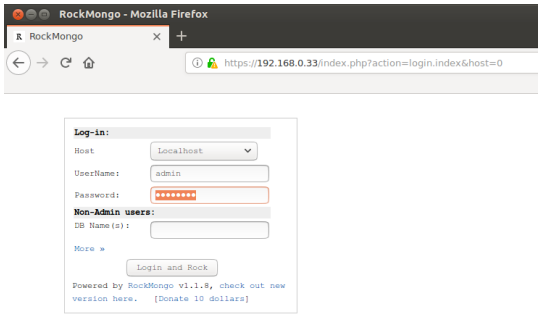


Fig. 11. Cliente RockMongo.

3.3.2 Configuración del driver pyMongo

Para poder utilizar la base de datos MongoDB en conjunción con los notebooks de JupyterHub es necesario utilizar un driver llamado pyMongo cuya instalación se realiza a través de un simple comando, pip install pymongo.

Se crean ciertas configuraciones para el correcto funcionamiento con jupyter y spark, se crea también una base de datos de prueba, usuarios, pero como en capítulos anteriores, estas explicaciones quedan detalladas en anexos.

En este caso, lo importante era comprobar el correcto funcionamiento del driver, por lo que se ejecuta una aplicación sencilla y se comprueba el resultado con un cliente de MongoDB llamado robo3T:

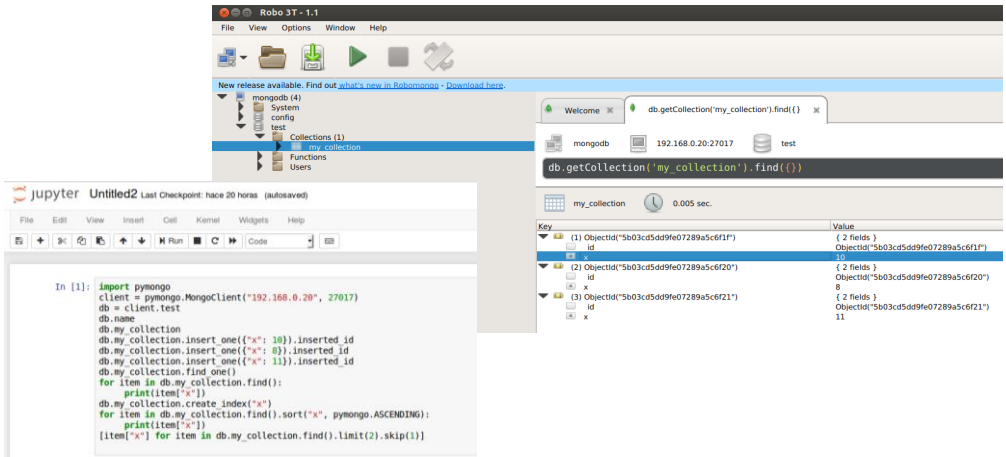


Fig. 12. Código conexión con MongoDB y cliente Robo 3T.

4. Escenario de análisis. Simulación de datos.

Llegados a este punto, en el que tenemos la plataforma creada en su totalidad, vamos a proceder a crear un escenario simulado que nos permitirá crear las tablas y los datos necesarios para posteriormente hacer los análisis propuestos.

4.1 Creación del escenario de análisis

4.1.1 Creación de cursos y alumnos en la plataforma Moodle.

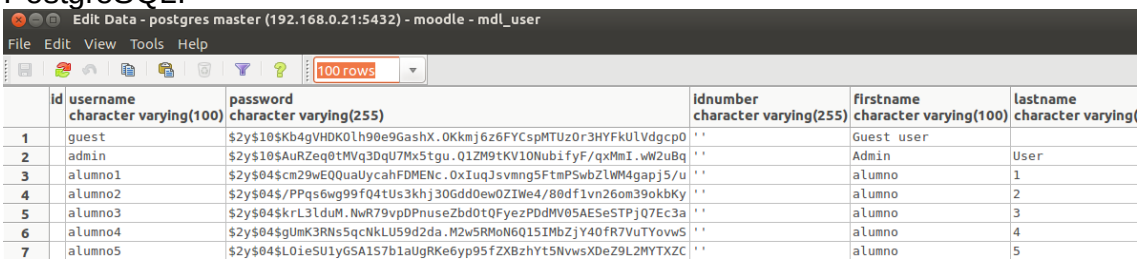
Hasta este momento, en cuanto a la plataforma Moodle se refiere, solamente teníamos un usuario de ejemplo y poco más. Si bien es verdad que en la instalación del servidor y la base de datos se crean las tablas con ciertos datos por defecto, ahora tenemos que crear un ejemplo que simule la realidad de un campus.

Se crea una categoría programación, y dos cursos dentro de esta categoría, Python y Javascript. Dentro de cada curso se crean 10 sesiones, y para probar la funcionalidad, uno de ellos tiene formato de semanas y el otro formato de temas.

Se crean también los alumnos, tal como se describe en el anexo asociado a este capítulo, existe un procedimiento para crearlos de forma masiva, y por último, es necsario matricular a estos alumnos en los cursos.

Fig. 13. Curso y usuarios simulados.

Se comprueba la ctualización de todos estos datos en la base de datos PostgreSQL.



	id	username character varying(100)	password character varying(255)	idnumber character varying(255)	firstname character varying(100)	lastname character varying(
1		guest	$2y$10$Kb4gVHDKOlh90e9GashX.OKkmj6z6FYCspMTUzOr3HYFkUlVdgcpO	''	Guest user	
2		admin	$2y$10$AuRZeq0tMVq3DqU7Mx5tgu.Q1ZM9tKV1ONubifyF/qxMmI.wW2uBq	''	Admin	User
3		alumno1	$2y$04$cm29wEQQuaUycahFDMENc.OxIuqJsvmng5FtmPSwbZlWM4gapj5/u	''	alumno	1
4		alumno2	$2y$04$/PPqs6wg99fQ4tUs3khj3OGddOewOZIWe4/80df1vn26om39okbKy	''	alumno	2
5		alumno3	$2y$04$krL3lduM.NwR79vpDPnuseZbdOtQFyezPDdMV05AESeSTPjQ7Ec3a	''	alumno	3
6		alumno4	$2y$04$gUmK3RNs5qcNkLU59d2da.M2w5RMoN6Q15IMbZjY4OfR7VuTYovwS	''	alumno	4
7		alumno5	$2y$04$LOieSU1yGSA1S7b1aUgRKe6yp95fZXBzhYt5NvwsXDeZ9L2MYTXZC	''	alumno	5

Fig. 14. Usuarios actualizados en tabla PostgreSQL.

4.1.2 Creación de usuarios en JupyterHub, integración de Moodle con Spark y generación de datos de la actividad de los usuarios en MongoDB.

Para la correcta integración de los dos tipos de bases de datos, postgresql y mongodb, necesitamos que en ambas los usuarios tengan la misma denominación. En el caso de JupyterHub, que hará uso de mongoDB, los usuarios son los mismos que los del sistema, por lo tanto será necesario crear 100 usuarios en la máquina que aloja el servidor.

Se trata de un trabajo puro de administración de sistemas, por lo que queda detallado en anexos.

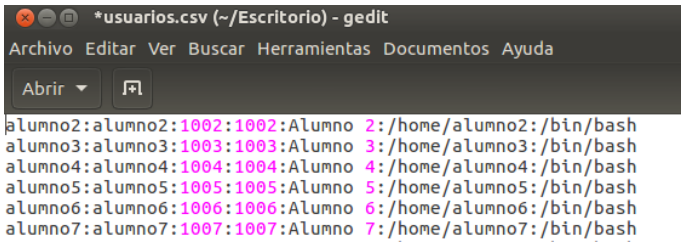


Fig. 15. Usuarios del sistema Spark.

Un aspecto clave del proyecto era la conexión desde un curso de Moodle a la plataforma de programación interactiva Spark/JupyterHub. Esto se ha conseguido mediante un módulo de navegador incrustado en uno de los temas del curso de Python.

La creación de un notebook con código real que genere los campos propuestos a analizar queda fuera de los objetivos de este proyecto, ya que no habría tiempo material para diseñarlo así como para generar los datos con usuarios reales. Por lo tanto, a partir del archivo que se utilizó para comprobar el funcionamiento de mongoDB, se ha modificado y, gracias a funciones de aleatoriedad, se ha introducido información hasta tener un volumen cercano a los 100.000 documentos.

Por último, conectándonos con el cliente web Rockmongo, podemos observar un ejemplo de documento en formato JSON.

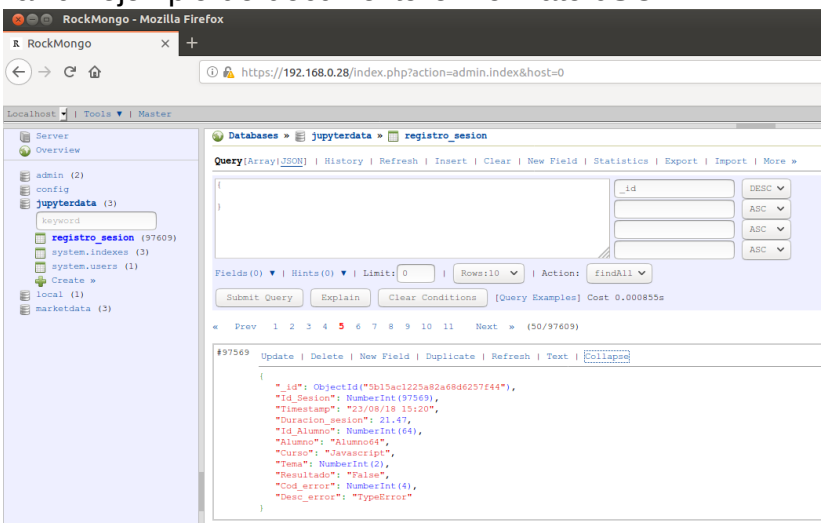


Fig. 16. Ejemplo de documento JSON en MongoDB.

Parte 2: Almacenamiento y explotación de datos.

5. Modelo de datos y procesos ETL.

5.1 Análisis de requerimientos

Objetivo general del proyecto: Como se deduce del título del caso de estudio, se trata de diseñar un almacén de datos que permita la gestión y reutilización de información de la actividad realizada por los estudiantes en las distintas sesiones de estudio. El propósito principal es ofrecer:

- Informes predefinidos o personalizados.
- Cuadros de mando específicos.
- Acceso al análisis libre de datos.

El sistema tiene que ser capaz de dar respuesta a las siguientes cuestiones:

- Datos sobre el grado de utilización de la plataforma.
- Datos sobre la evolución del aprendizaje de los alumnos.
- Mejora de la calidad de los contenidos.
- Datos históricos sobre los distintos cursos para mejorar la regularidad y la eficiencia.
- Análisis de los errores más comunes en las distintas tareas.

5.2 Análisis de fuentes de datos

Se dispone de 2 dos bases de datos que habrá que integrar para formar el almacén de datos. La primera de ellas es una base de datos relacional, en la que se utiliza como servidor PostgreSQL y que sirve como base para el correcto funcionamiento de la plataforma Moodle. En la página siguiente se muestra el esquema completo de la versión de Moodle 3.2.2 utilizada en el proyecto. Existe una gran cantidad de información y de tablas en esta base de datos. Para nuestro almacén de datos sólo utilizaremos parte de la información de las dcuatro primeras tablas, aunque existen otras tablas que merece la pena mencionar por su importancia y potencial para el análisis en proyectos futuros:

- **{user}** → Información sobre usuarios
- **{course}** → Cursos dentro del campus **(Incluye la tabla categoría y secciones)**
- {config} → Parametrización del sitio web
- {logstore_standard_log} → Los logs, todo lo que se ha hecho en Moodle. En este caso, la información sobre las sesiones de programación está en la base de datos MongoDB.
- {quiz} y {quiz_attempts} → Elementos calificativos cuestionario e intentos de usuarios.
- {scorm} y {scorm_scoes_track} → Prácticamente toda la información sobre el scorm y el intento del usuario.

Las últimas tablas nos proporcionarían un entorno de análisis de otros tipos de actividad como cuestionarios, talleres, sesiones en la propia plataforma Moodle, etc.

En este proyecto el objetivo es el de analizar la actividad del entorno de programación interactivo, por lo que la actividad a analizar será la de los usuarios y cursos recogida en la base de datos MongoDB a partir de los notebooks de JupyterHUB y el driver pyMongo.

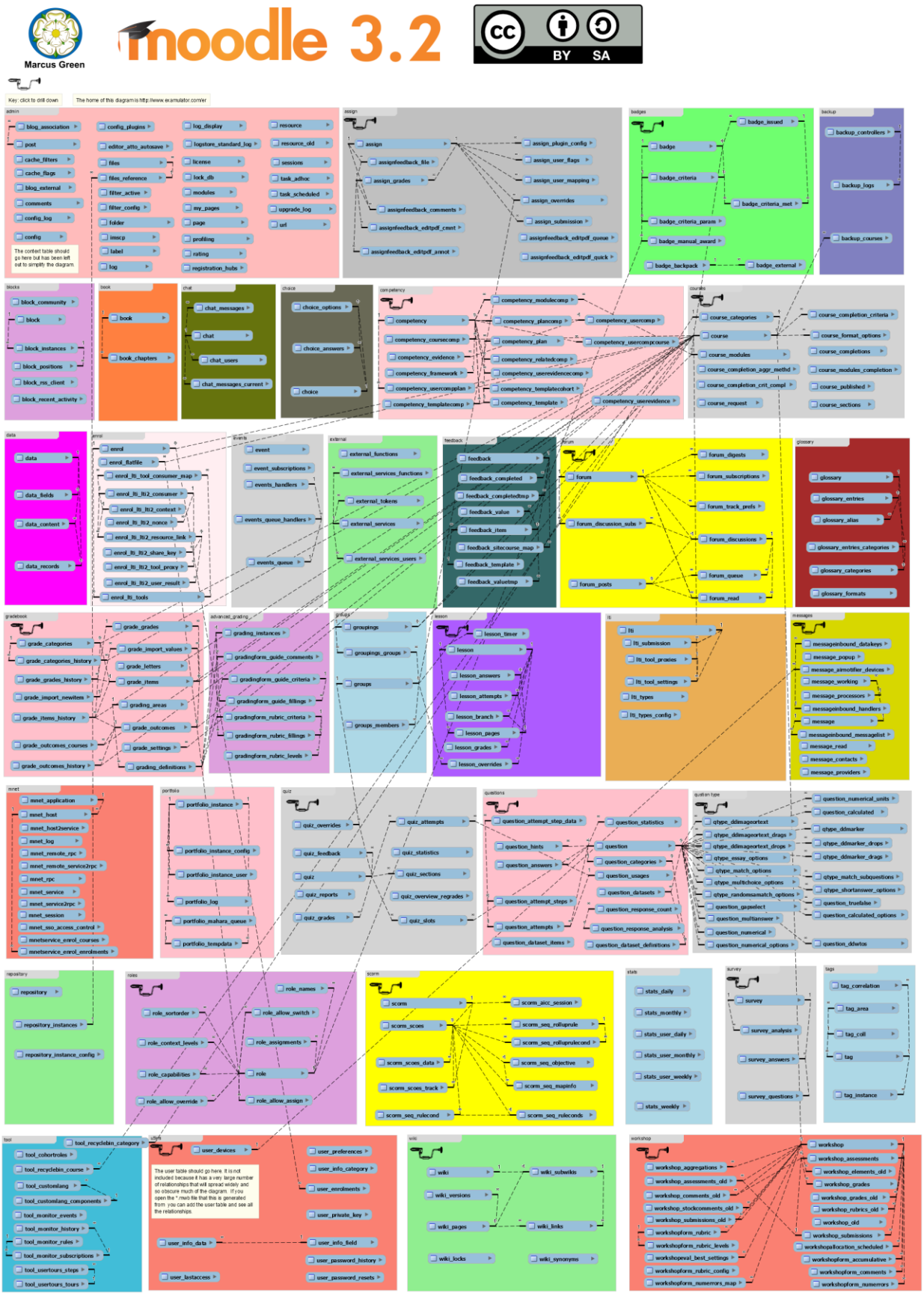


Fig. 17. Esquema completo de BBDD de Moodle 3.2.

En cuanto a la segunda base de datos, MongoDB, se trata de una base de datos NoSQL basada en documentos. El origen de datos en este caso es una única tabla llamada registro_sesion que contiene información sobre los usuarios, los cursos y los temas, información sobre fecha y hora, duración de las sesiones, resultado de las mismas, e información sobre los errores.

Al igual que en el caso de la base de datos de Moodle, y a pesar de que sobrepasa los objetivos de este proyecto, se incluye a continuación información sobre como se guardan los notebooks de Jupyter en formato JSON y la información principal que puede ser susceptible de análisis:

- Estructura de alto nivel:

Top-level structure

```
{
  "metadata" : {
    "signature": "hex-digest", # used for authenticating unsafe outputs on load
    "kernel_info": {
        # if kernel_info is defined, its name field is required.
        "name" : "the name of the kernel"
    },
    "language_info": {
        # if language_info is defined, its name field is required.
        "name" : "the programming language of the kernel",
        "version": "the version of the language",
        "codemirror_mode": "The name of the codemirror mode to use [optional]"
    }
  },
  "nbformat": 4,
  "nbformat_minor": 0,
  "cells" : [
      # list of cell dictionaries, see below
  ],
}
```

Fig. 18. Documento JSON de un notebook de Jupyter.

Contiene información sobre el notebook en general, como por ejemplo metadatos, el lenguaje utilizado en el kernel, otros formatos y por último un listado de tipos de celdas. Estas celdas son las que nos proporcionarán la información más interesantes y son las siguientes:

- Markdown cells → Son celdas con texto informativo o de cuerpo con explicaciones.
- Code cells → Son las celdas con el código a ejecutar.
- Code cell outputs:
  - Streaming de salida → Datos que se van generando
  - Display_data → Pueden ser imágenes, gráficos, etc.
  - Execute_result → Resultado de la ejecución, cálculos, etc.
  - Error → Tipos de error generados.

Code cells

```
{
  "cell_type" : "code",
  "execution_count": 1, # integer or null
  "metadata" : {
      "collapsed" : True, # whether the output of the cell is collapsed
      "autoscroll": False, # any of true, false or "auto"
  },
  "source" : ["some code"],
  "outputs": [{
      # list of output dicts (described below)
      "output_type": "stream",
      ...
  }],
}
```

Fig. 19. Documento JSON de una celda de código.

En nuestro caso se han generado una cantidad importante de documentos de forma sintética. No existe tiempo material, ni alumnos reales, para elaborar un notebook con código que genere información sobre sesiones, duración, errores, etc. por lo que se ha generado de forma manual en un archivo csv con fórmulas aleatorias:

dataset_val.csv - LibreOffice Calc

Archivo Editar Ver Insertar Formato Hoja Datos Herramientas Ventana Ayuda

Liberation Sans 10

O28

	A	B	C	D	E	F	G	H	I	J
1	Id_Sesion	Timestamp	Duracion_sesion	Id_Alumno	Alumno	Curso	Tema	Resultado	Cod_error	Desc_error
2	1	24/08/17 09:30	41.93	18	Alumno18	Javascript	6	True		
3	2	24/08/17 09:31	42.86	52	Alumno52	Python	5	True		
4	3	24/08/17 09:32	59.25	22	Alumno22	Javascript	1	False	5	TypeError
5	4	24/08/17 09:33	54.78	40	Alumno40	Python	3	True		
6	5	24/08/17 09:34	45.33	89	Alumno89	Python	8	True		
7	6	24/08/17 09:35	25.78	18	Alumno18	Python	2	True		
8	7	24/08/17 09:36	49.81	63	Alumno63	Javascript	3	False	3	ValueError
9	8	24/08/17 09:37	46.27	6	Alumno6	Python	2	True		
10	9	24/08/17 09:38	37.01	14	Alumno14	Javascript	3	True		
11	10	24/08/17 09:39	58.48	50	Alumno50	Javascript	9	False	4	SyntaxError
12	11	24/08/17 09:40	5.12	12	Alumno12	Javascript	8	True		
13	12	24/08/17 09:41	37.05	5	Alumno5	Javascript	4	True		
14	13	24/08/17 09:42	8.45	98	Alumno98	Python	7	True		
15	14	24/08/17 09:43	32.28	71	Alumno71	Python	1	True		
16	15	24/08/17 09:44	1.12	5	Alumno5	Python	4	True		
17	16	24/08/17 09:45	52.45	95	Alumno95	Python	9	True		
18	17	24/08/17 09:46	4.57	38	Alumno38	Javascript	3	False	3	SyntaxError
19	18	24/08/17 09:47	19.88	99	Alumno99	Python	9	True		
20	19	24/08/17 09:48	27.64	79	Alumno79	Javascript	9	True		
21	20	24/08/17 09:49	58.16	35	Alumno35	Python	3	True		
22	21	24/08/17 09:50	25.43	74	Alumno74	Javascript	9	False	5	TypeError
23	22	24/08/17 09:51	37.69	71	Alumno71	Python	4	True		
24	23	24/08/17 09:52	39.96	7	Alumno7	Javascript	10	False	4	TypeError
25	24	24/08/17 09:53	29.11	66	Alumno66	Javascript	3	False	2	SyntaxError

Fig. 20. Dataset sintetico utilizado en el proyecto.

5.3 Análisis de fuentes de datos. Tipos de datos y niveles de información

En esta primera aproximación se analiza la información “util” de las tablas de las dos bases de datos. Existen 5 tablas a partir de las cuales se va a crear el datawarehouse.

En primer lugar la tabla cursos, con 118 campos totales, de los cuales 29 campos son útiles para nuestro análisis. A partir de estos campos se indica el tipo de datos y niveles de las dimensiones.

En principio existen campos de un sólo nivel, que posteriormente se podrán agrupar en diferentes niveles generando nuevas dimensiones.

La precisión de los números decimales es de 1 posición decimal.

Base de datos: moodle	**Tabla: mdl_course_category**		
Nombre del atributo	Tipo de dato	Nivel de información	Comentarios
id	Bigserial	1 nivel (id)	Clave Primaria
name	Character Varying (254)	1 nivel (Nombre corto)	
description	Text	1 nivel (Descripción)	

Base de datos: moodle	**Tabla: mdl_course**		
Nombre del atributo	Tipo de dato	Nivel de información	Comentarios
id	Bigserial	1 nivel (id)	Clave Primaria
category	Bigint	1 nivel (Categoría)	Clave Foránea
fullname	Character Varying (254)	1 nivel (Nombre completo)	
shortname	Character Varying (255)	1 nivel (Nombre corto)	
summary	Text	1 nivel (Descripción)	
format	Character Varying (21)	1 nivel (Formato de temas)	Puede ser por semanas o por temas

Base de datos: moodle	**Tabla: mdl_course_sections**		
Nombre del atributo	Tipo de dato	Nivel de información	Comentarios
id	Bigserial	1 nivel (id)	Clave Primaria
course	Bigint	1 nivel (curso)	
section	Bigint	1 nivel (tema)	

Base de datos: moodle Tabla: mdl_users			
Nombre del atributo	Tipo de dato	Nivel de información	Comentarios
id	Bigserial	1 nivel (id)	Clave Primaria
username	Character Varying (100)	1 nivel (nombre)	Clave Foránea
password	Character Varying (254)	1 nivel (contraseña)	
firstname	Character Varying (100)	1 nivel (nombre)	
lastname	Character Varying (100)	1 nivel (apellido)	En este ejemplo puede ser clave foranea
email	Character Varying (100)	1 nivel (email)	

Base de datos: jupyterdata Tabla: registro_sesion			
Nombre del atributo	Tipo de dato	Nivel de información	Comentarios
id_sesion	Int32	1 Nivel (id)	Clave primaria
timestamp	String	1 Nivel (fecha y hora)	Transformacion ETL → Dia, mes, año, hora y minuto
duracion_sesion	Double	1 Nivel (Duración)	Transformación ETL → Minutos y segundos
id_alumno	Int32	1 Nivel (id del alumno)	Clave foranea
alumno	String	1 Nivel (alumno)	
curso	String	1 Nivel (curso)	Clave foranea
tema	Int32	1 Nivel (tema)	Clave foranea
resultado	String	1 Nivel (resultado)	
cod_error	String	1 Nivel (codigo de error)	
desc_error	String	1 Nivel (descripción del error)	

En conclusión, tenemos 29 campos que no son prescindibles, a partir de los cuales crearemos el almacén de datos.

5.4 Análisis funcional

En este apartado se describen los requisitos funcionales del sistema. A partir del enunciado se establecen unos requerimientos mínimos que nos permitirán elegir la arquitectura más adecuada.

- Con un nivel de prioridad máximo, el DW deberá poder atender las necesidades del profesorado.

- En primer lugar, a partir del análisis de las fuentes, el DW tendrá que extraer de manera adecuada la información, pero además tendrá que transformarla para agruparla de la forma más adecuada.

- Será necesaria una carga inicial de la información, y después sería necesario definir como se actualiza dicha información. Se trata de una solución sencilla, pero que puede servir de base para un modelo de datos mayor con más tablas de hecho y dimensiones.

- Existirá un modelo OLAP que esté disponible para dar respuestas a las distintas cuestiones analíticas planteadas.

- Sería deseable disponer de un soporte a los metadatos de gestión.

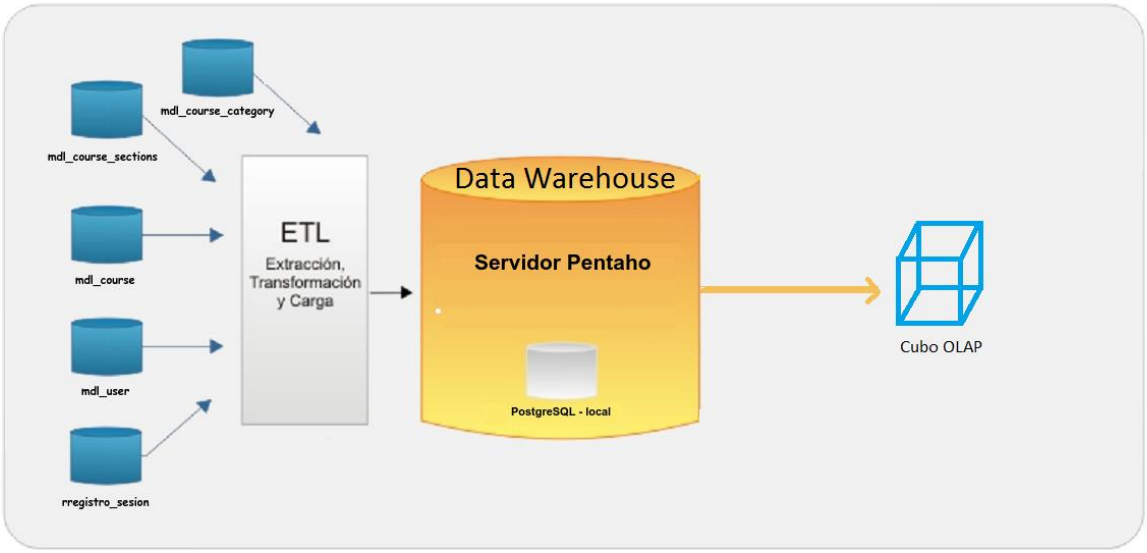


Fig. 21. Diarama conceptual de la arquitectura de análisis de datos.

5.5 Diseño del modelo conceptual, lógico y físico del almacén de datos.

5.5.1 Diseño del modelo conceptual.

A partir del análisis de requerimientos y del análisis de las fuentes de datos, se han identificado 1 tablas de hechos, 4 dimensiones y 25 atributos

La tabla de hecho es la siguiente:

h_registro → Registro de la actividad de los alumnos en cuanto a tiempo y en cuanto a contenidos y resultados.

Por otro lado, las dimensiones y sus correspondientes atributos estarían organizados de la siguiente manera:

Nombre	d_curso
Descripción	Datos que describen los cursos.
Atributos	id_categoria, nom_categoria, desc_categoria id_curso, fullname, shortname, descripcion, formato_temas id_tema, tema
Jerarquía	3 Niveles: Categoría → Curso →Tema

Nombre	d_usuario
Descripción	Datos que describen a los usuarios/alumnos
Atributos	id_usuario, username, password, nombre, apellido, email
Jerarquía	1 Nivel

Nombre	d_sesion
Descripción	Datos con información sobre las sesiones de estudio
Atributos	id_sesion, timestamp, duracion, resultado, cod_error, desc_error
Jerarquía	1 Nivel

Nombre	d_tiempo
Descripción	Datos de tiempo
Atributos	fecha, año, mes, día
Jerarquía	3 Niveles: Año → Mes → Día

5.5.2 Diseño lógico.

A continuación se presenta el gráfico del diseño lógico para el hecho registro. Por claridad se presenta un modelo en estrella, pero en realidad, las distintas dimensiones están relacionadas entre ellas, por lo que sería un modelo de copo de nieve. Las relaciones entre dimensiones se definen en el diseño físico a través de las claves foráneas.

h_registro → Registro de la actividad de los alumnos

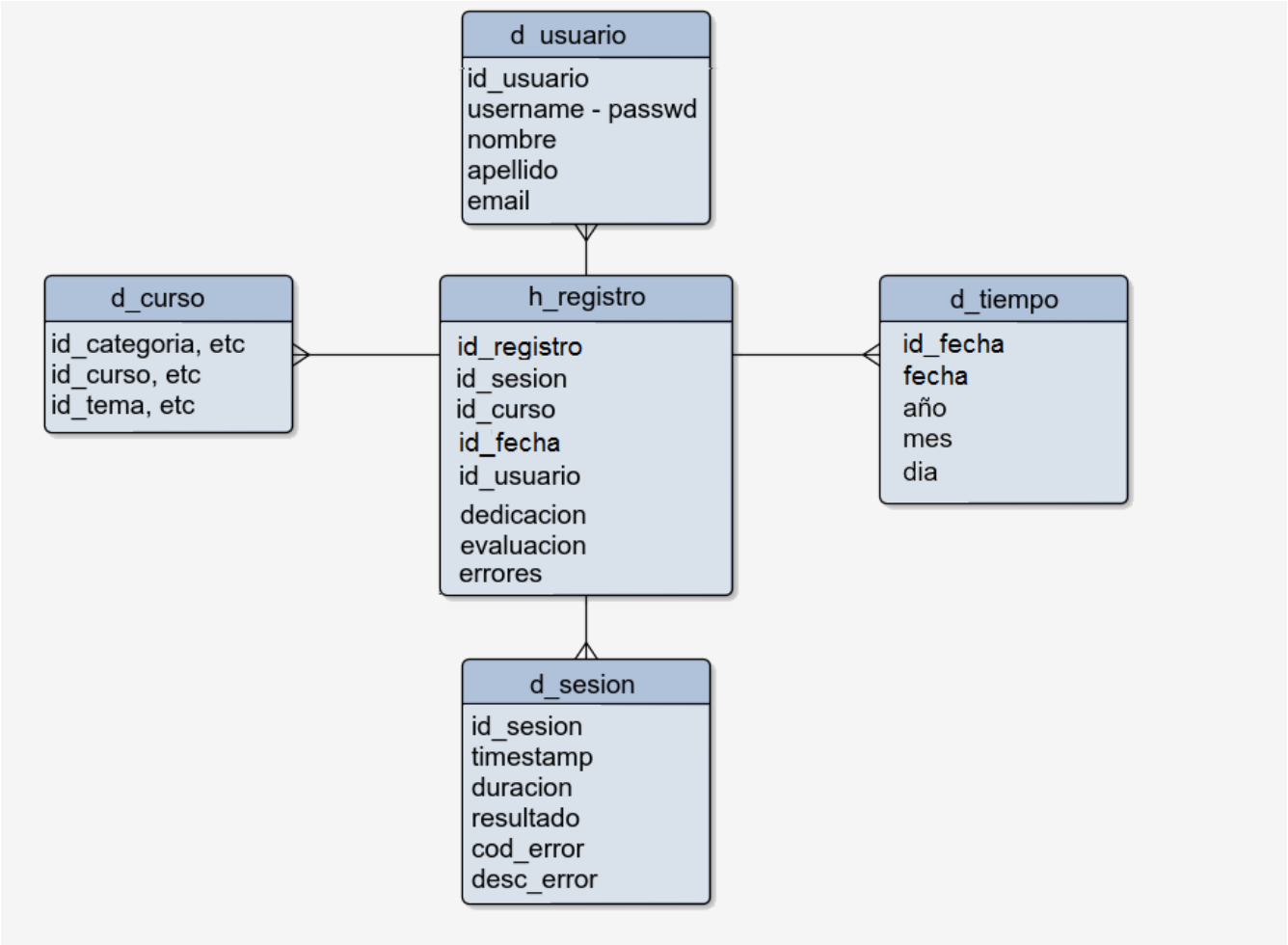


Fig. 22. Diseño lógico del almacén de datos.

*Nota. Por extensión no se muestran todos los atributos de la dimensión curso.

5.5.3 Diseño físico.

El último paso del diseño del Data Warehouse, será el diseño físico, por lo que se presentan a continuación los scripts de creación de tablas. Cabe comentar, que una vez vistos el modelo conceptual y el diseño lógico, se ve claramente que se trata de un diseño ROLAP, ya que tenemos gran cantidad de tablas con indices para las claves y particiones horizontales, sobre todo en la dimensión temporal, años y meses.

Se presentan a continuación los scripts de creación de las tablas de las distintas dimensiones del almacén de datos.

```
CREATE TABLE "D_CURSO"
(
"ID_CATEGORIA" BIGSERIAL NOT NULL,
"NOM_CATEGORIA" VARCHAR(254),
"DESC_CATEGORIA" TEXT NOT NULL,
"ID_CURSO” BIGSERIAL NOT NULL,
"FULLNAME" VARCHAR(254),
"SHORTNAME" VARCHAR(254),
"DESCRIPCION" TEXT,
"FORMATO_TEMAS" VARCHAR(21) NOT NULL,
"ID_TEMAS” BIGSERIAL NOT NULL,
"TEMA” BIGINT NOT NULL,
CONSTRAINT "D_CURSO_PK" PRIMARY KEY ("ID_CURSO")
) ;


CREATE TABLE "D_USUARIO"
(
"ID_USUARIO" BIGSERIAL NOT NULL,
"USERNAME" VARCHAR(100) NOT NULL,
"PASSWORD" VARCHAR(254) NOT NULL,
"NOMBRE" VARCHAR(100) NOT NULL,
"APELLIDO" VARCHAR(100) NOT NULL,
"EMAIL" VARCHAR(100) NOT NULL,
CONSTRAINT "D_USUARIO_PK" PRIMARY KEY ("ID_USUARIO")
) ;


CREATE TABLE "D_SESION"
(
"ID_SESION" BIGSERIAL NOT NULL,
"TIMESTAMP" TIMESTAMP NOT NULL,
"DURACION" DOUBLE NOT NULL,
"RESULTADO" VARCHAR(254) NOT NULL,
"COD_ERROR" VARCHAR(254) NOT NULL,
"DESC_ERROR" VARCHAR(254) NOT NULL,
CONSTRAINT "D_SESION_PK" PRIMARY KEY ("ID_SESION")
) ;
```

```
CREATE TABLE "D_TIEMPO"
(
"ID_FECHA" BIGSERIAL NOT NULL,
"FECHA" DATE NOT NULL,
"DIA" NUMBER(2,0) NOT NULL,
"MES" NUMBER(2,0) NOT NULL,
"ANYO" NUMBER(4,0) NOT NULL,
CONSTRAINT "D_FECHA_PK" PRIMARY KEY ("ID_FECHA")
) ;
```

```
CREATE TABLE "H_REGISTRO"
(
"ID_REGISTRO" BIGINT NOT NULL,
"ID_SESION" BIGINT NOT NULL ENABLE,
"ID_USUARIO" BIGINT NOT NULL ENABLE,
"ID_CURSO" BIGINT NOT NULL ENABLE,
"ID_TEMA" BIGINT NOT NULL ENABLE,
"ID_FECHA" BIGINT NOT NULL ENABLE,
"DURACION" DOUBLE NOT NULL,
"RESULTADO" VARCHAR(254) NOT NULL,
"COD_ERROR" VARCHAR(254) NOT NULL,
CONSTRAINT "D_ REGISTRO_PK" PRIMARY KEY ("ID_REGISTRO") ENABLE,
CONSTRAINT "D_REGISTRO_FK_SESION" FOREIGN KEY ("ID_SESION")
REFERENCES "D_SESION" ("ID_SESION") ENABLE,
CONSTRAINT "D_REGISTRO_FK_USUARIO" FOREIGN KEY ("ID_USUARIO")
REFERENCES "D_USUARIO" ("ID_USUARIO") ENABLE,
CONSTRAINT "D_REGISTRO_FK_CURSO" FOREIGN KEY ("CURSO")
REFERENCES "D_CURSO" ("ID_CURSO") ENABLE,
CONSTRAINT "D_FECHA_FK_FECHA" FOREIGN KEY ("ID_FECHA")
REFERENCES "D_TIEMPO" ("ID_FECHA") ENABLE
) ;
```

6. Creación del DataWareHouse.

6.1 Conexión a las bases de datos

El primer paso para poder acceder a las fuentes de datos, realizar transformaciones y cargar los datos seleccionados en el almacén de datos será configurar las conexiones:

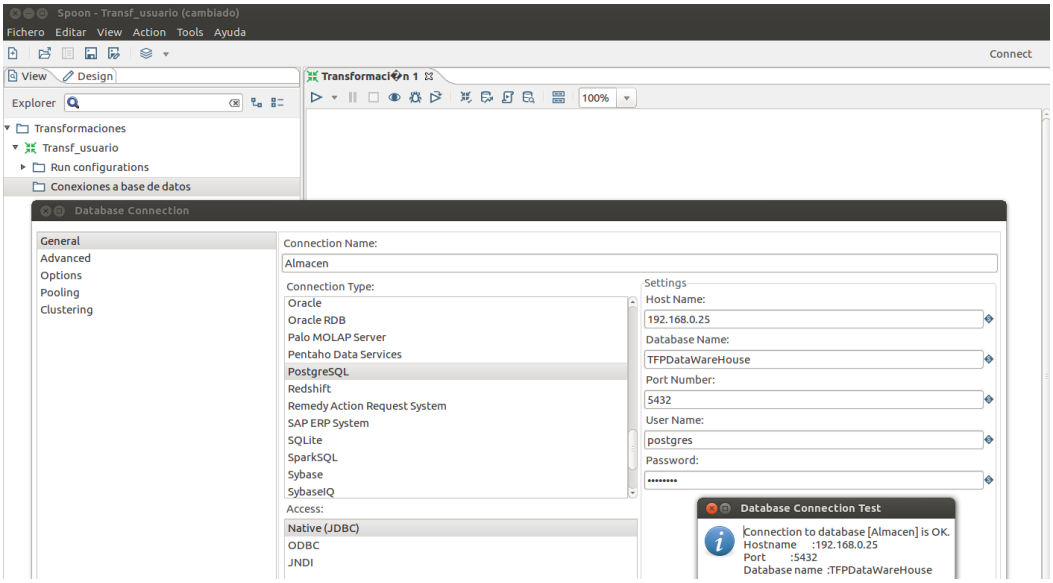


Fig. 23. Conexión a BBDD desde Spoon.

Las opciones más importantes a configurar serán el tipo de conexión, en el ejemplo se trata de una base de datos PostgreSQL que utiliza el conector JDBC. Hay que indicar la dirección a modo de nombre del host, la base de datos a la que vamos a conectar, puerto, nombre de usuario y contraseña. Además hay que indicar un nombre para la conexión. En este proyecto las dos conexiones a PostgreSQL son Almacen y MoodleDB y existe la posibilidad de configurarlas y compartirlas para poder utilizarlas en todas las transformaciones que creemos. Además, una vez configuradas las conexiones podemos probar y explorar. A continuación se muestran las tablas del almacén creadas mediante los scripts del diseño físico:

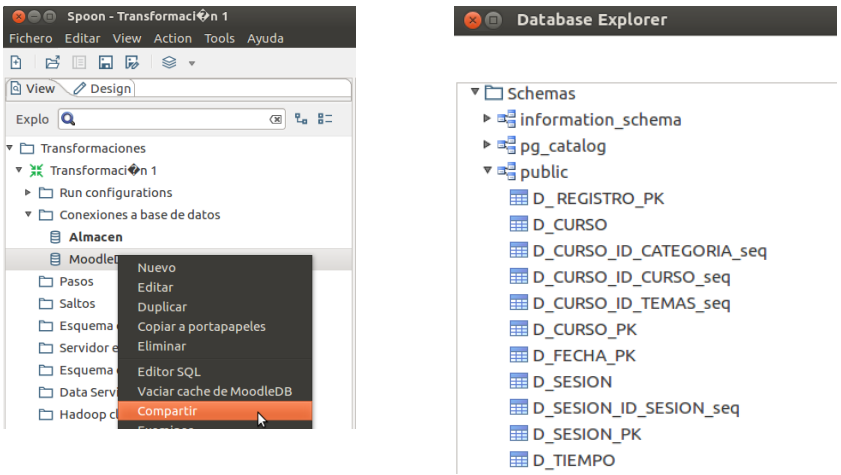


Fig. 24. Compartir conexiones entre transformaciones.

La primera transformación que vamos a hacer es la de carga de los datos en la dimensión usuario. Tendremos que utilizar como origen de los datos la tabla mdl_user de la base de datos de moodle, para lo cual utilizamos el paso de transformación “table input”. Elegimos la conexión y escribimos la query SQL para seleccionar los campos que nos interesan:

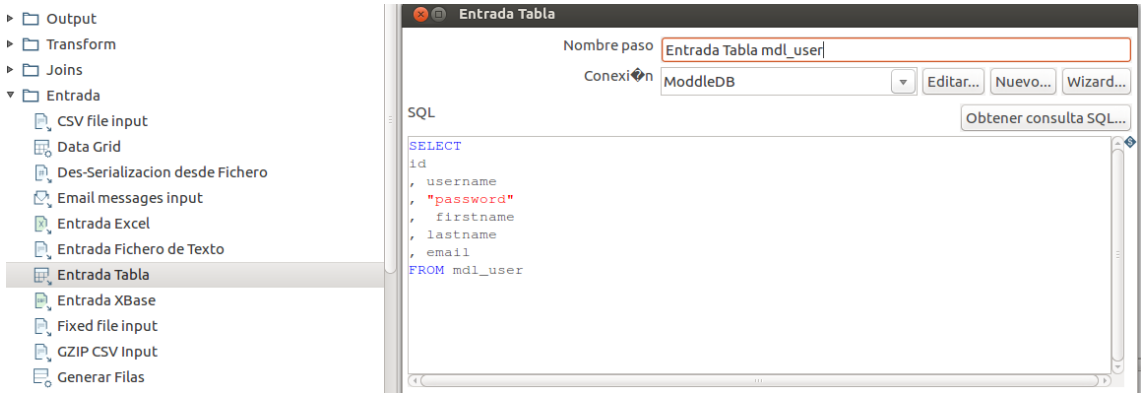


Fig. 25. Paso ETL de entrada de datos mediante tabla.

A través del paso “Crear datos” crearemos una tabla intermedia en el almacén de datos. Estas tablas tendrá el prefijo dim_ y nos servirán para trabajar con los datos en el siguiente paso, “Seleccionar/Renombrar valores”, en el que eliminamos la información que no queremos y renombramos los campos de las tablas. Estos dos pasos se utilizan en todas las transformaciones del proyecto.

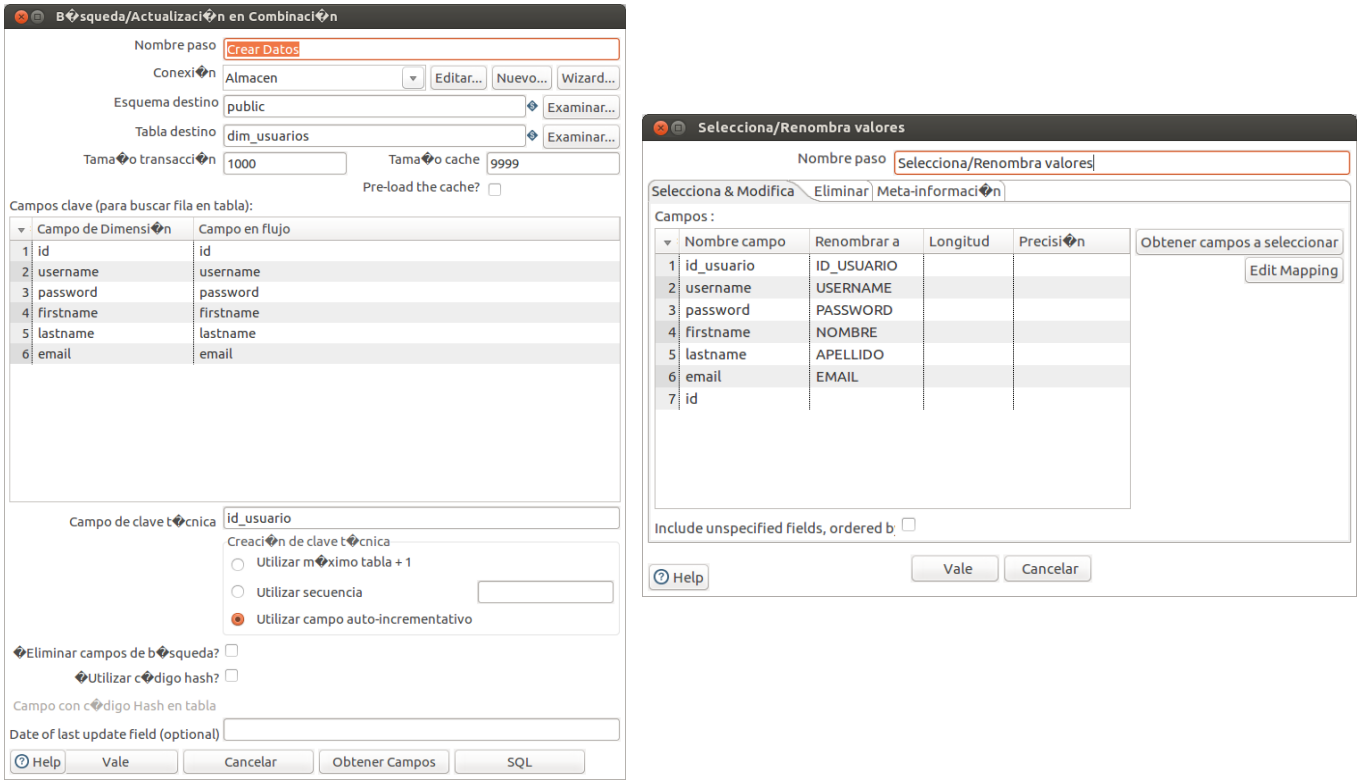


Fig. 26. Pasos ETL de actualización y selección de campos.

Finalmente, después de ordenar los datos por la clave principal, se cargan los datos en la dimensión usuario del almacén:

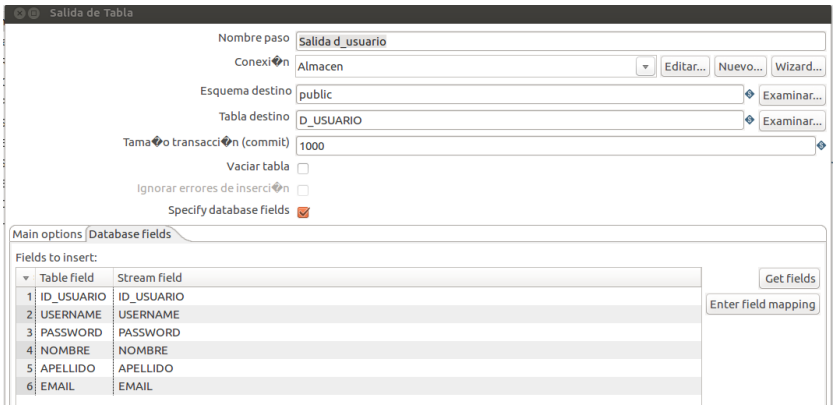


Fig. 27. Paso ETL de salida al almacén de datos.

Por último, en cuanto a la transformación para la dimensión usuario, sólo resta ejecutarla y a continuación se muestra el proceso y su correcta ejecución:

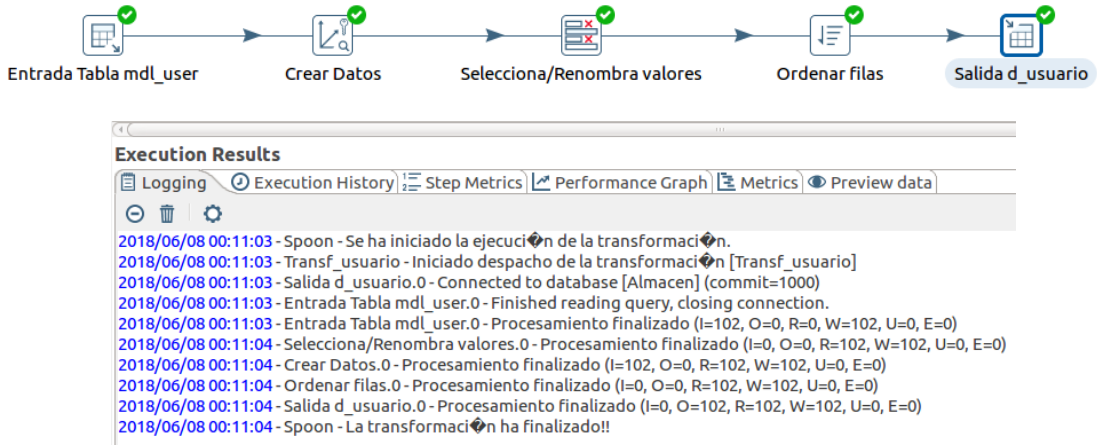


Fig. 28. Transformación de la dimensión usuario.

La siguiente transformación es muy similar a la anterior, pero el origen de los datos proviene de 3 tablas distintas, que además crearán una jerarquía de 3 niveles en el cubo OLAP.

Creamos como en el caso anterior la dimensión intermedia:

Fig. 29. Detalle sentencia SQL dentro de un paso ETL.

Vemos que antes de aplicar los pasos de crear datos y seleccionar/renombrar, tenemos que hacer un join de las 3 tablas, pero por lo demás, el proceso es el mismo. Cargaremos en esta ocasión los datos en la tabla d_cursos del almacén:

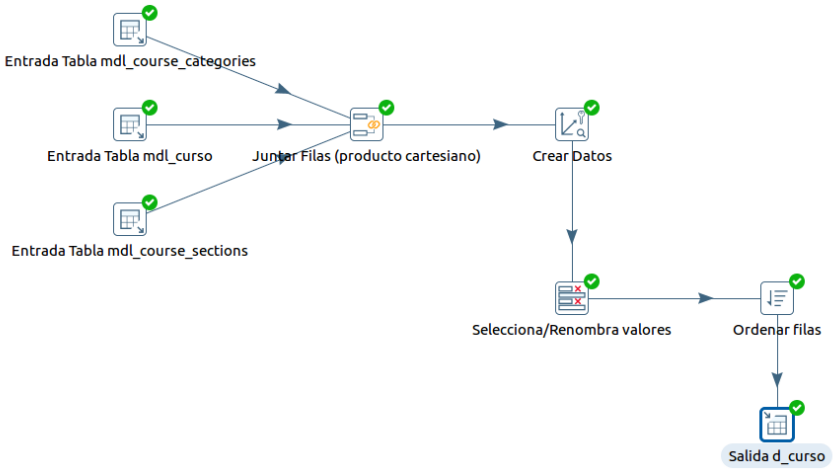


Fig. 30. Transformación de la dimensión cursos.

Para la dimensión d_sesión, así como para la dimensión d_tiempo, necesitamos crear una conexión a la base de datos MongoDB. En este caso no se trata de configurar la conexión como en el caso anterior, sino que se configura a través de una paso como entrada de datos (MongoDB Input):

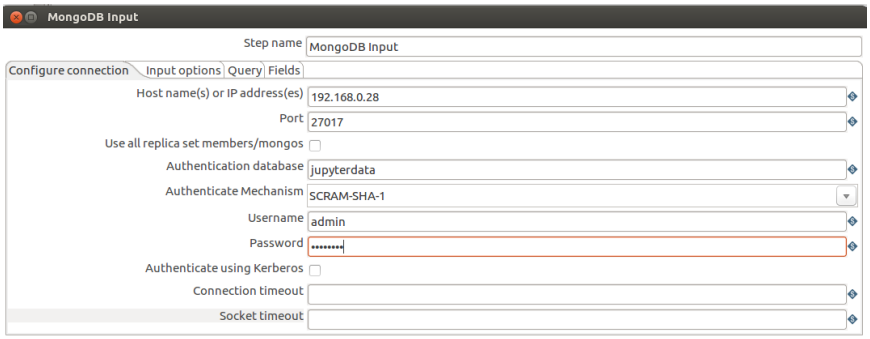


Fig. 31. Entrada de datos desde MongoDB.

La salida de este paso puede ser un único campo JSON, o una serie de campos, esta ultima opción es la que nos interesa.

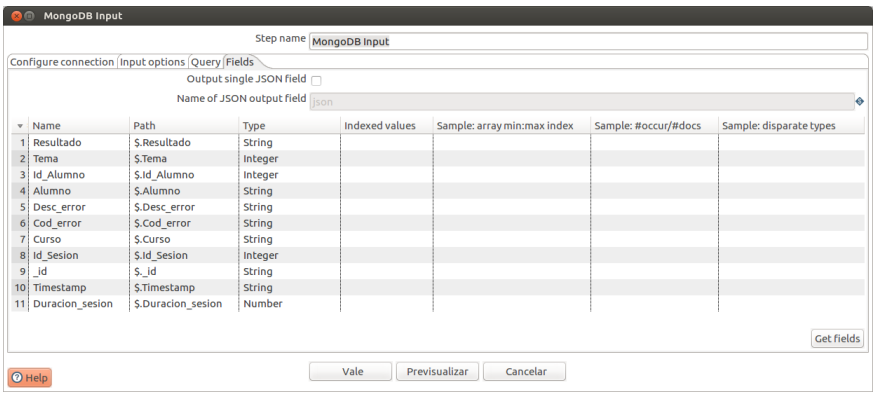
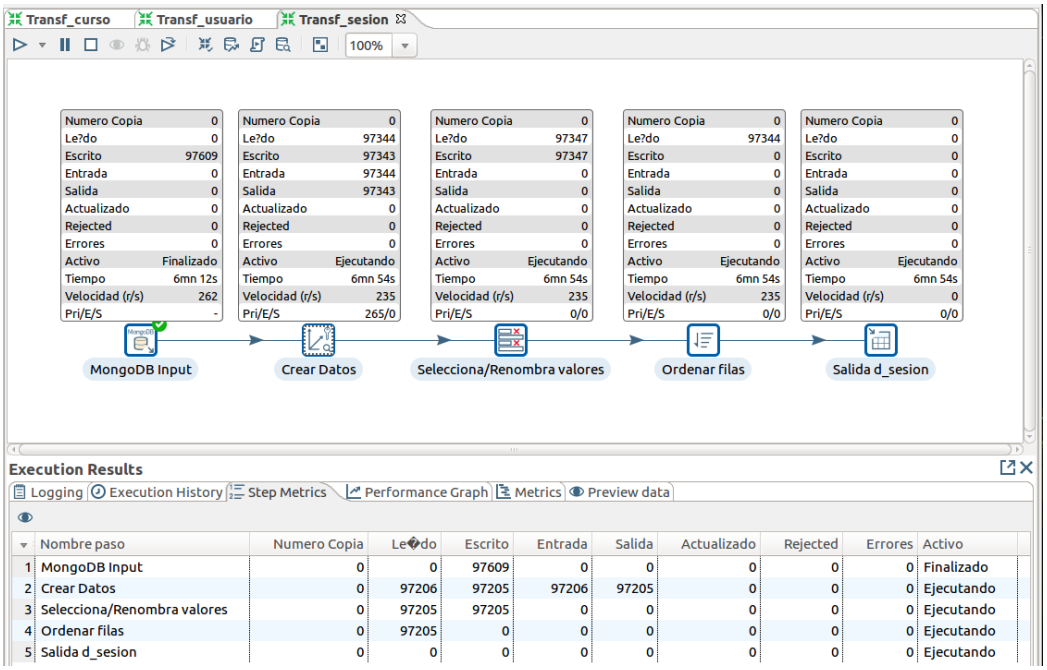


Fig. 32. Selección de campos de la estructura JSON.

Por lo demás, esta transformación es igual que las anteriores, salvo también por la cantidad de datos. En esta ocasión, tal como podemos ver a través de las métricas, hemos cargado los casi 100.000 registros. El proceso de transformación ha tardado aproximadamente 7 minutos en completar la carga en la tabla d_sesión del almacén de datos.



	Nombre paso	Numero Copia	Le?do	Escrito	Entrada	Salida	Actualizado	Rejected	Errores	Activo
1	MongoDB Input	0	0	97609	0	0	0	0	0	Finalizado
2	Crear Datos	0	97206	97205	97206	97205	0	0	0	Ejecutando
3	Selecciona/Renombra valores	0	97205	97205	0	0	0	0	0	Ejecutando
4	Ordenar filas	0	97205	0	0	0	0	0	0	Ejecutando
5	Salida d_sesion	0	0	0	0	0	0	0	0	Ejecutando

Fig. 33. Ejecución de la transformación de la dimensión sesion.

Por último, en cuanto a las transformaciones de las dimensiones, tenemos d_tiempo.

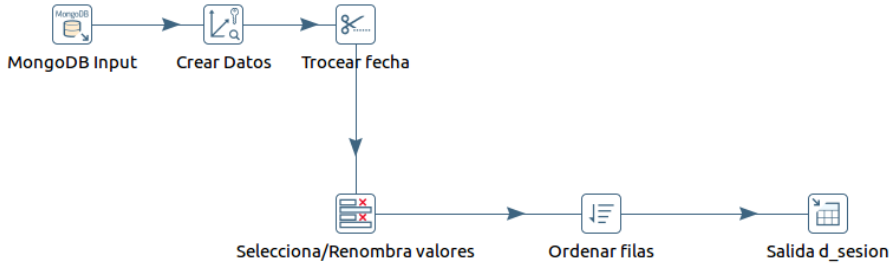
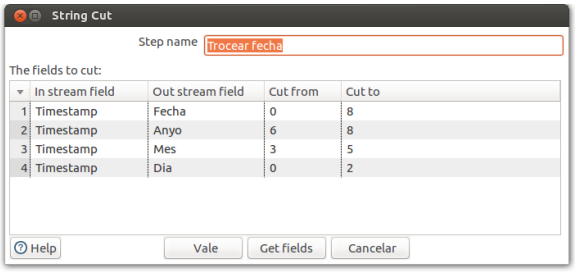


Fig. 34. Transformación de la dimensión tiempo.

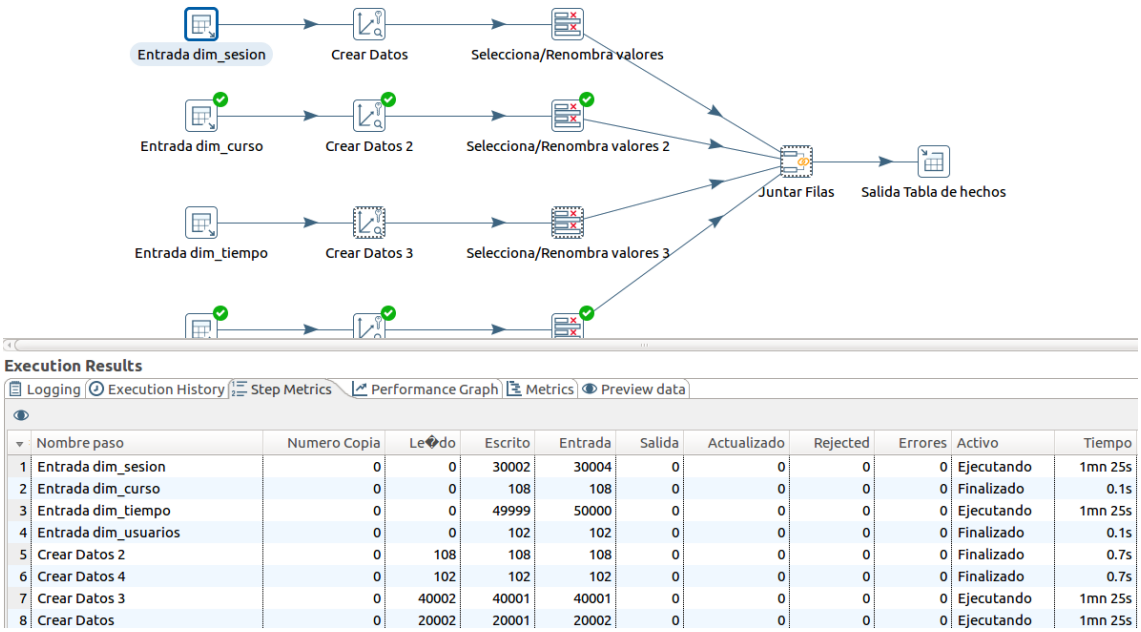
Al igual que en la transformación anterior, partimos de la base de datos MongoDB y los pasos son similares, sólo es necesario pasar datos de una tabla a otra. Sin embargo en esta transformación, la particularidad es que vamos a crear nuevos campos a través del campo Timestamp, troceando a partir de las posiciones de la cadena de caracteres:



	In stream field	Out stream field	Cut from	Cut to
1	Timestamp	Fecha	0	8
2	Timestamp	Anyo	6	8
3	Timestamp	Mes	3	5
4	Timestamp	Dia	0	2

Fig. 35. Paso ETL de troceo de cadena de caracteres.

Una vez que tenemos las 4 dimensiones del diseño lógico cargadas de datos, para terminar este capítulo sólo nos quedaría completar la carga de la tabla de hechos h_registro. Esta carga se hará aprovechando las tablas intermedias localizadas en el propio almacén de datos y cabe comentar que también existe un paso de juntar filas como producto cartesiano que hace que la cantidad de datos sea considerable. El proceso completo de carga ha superado la media hora.



	Nombre paso	Numero Copia	Le&#65533;do	Escrito	Entrada	Salida	Actualizado	Rejected	Errores	Activo	Tiempo
1	Entrada dim_sesion	0	0	30002	30004	0	0	0	0	Ejecutando	1mn 25s
2	Entrada dim_curso	0	0	108	108	0	0	0	0	Finalizado	0.1s
3	Entrada dim_tiempo	0	0	49999	50000	0	0	0	0	Ejecutando	1mn 25s
4	Entrada dim_usuarios	0	0	102	102	0	0	0	0	Finalizado	0.1s
5	Crear Datos 2	0	108	108	108	0	0	0	0	Finalizado	0.7s
6	Crear Datos 4	0	102	102	102	0	0	0	0	Finalizado	0.7s
7	Crear Datos 3	0	40002	40001	40001	0	0	0	0	Ejecutando	1mn 25s
8	Crear Datos	0	20002	20001	20002	0	0	0	0	Ejecutando	1mn 25s

Fig. 36. Transformación de carga de datos en la tabla de hechos.

Para terminar, se ha preparado un “job”, trabajo, para automatizar la ejecución de las distintas transformaciones. De esta forma se podrá ir actualizando el almacén de datos a medida que se vayan generando nuevos datos por parte de los usuarios.

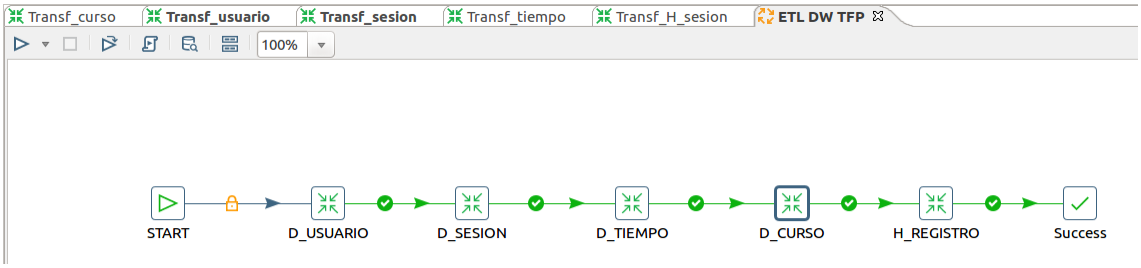


Fig. 37. Trabajo para la ejecución automática de las transformaciones.

7. Prototipo de análisis multidimensional. Cubo OLAP con Mondrian.

7.1 Creación del cubo multidimensional con Schema Workbench

Llegados a este punto del proyecto, sólo quedaría ponerle la guinda al pastel. Tenemos la plataforma funcionando correctamente y se han generado los datos a analizar. Estos datos se han recogido en distintas bases de datos que hemos integrado en un almacén de datos, en distintas dimensiones y en una tabla de hechos. Ahora sólo queda diseñar un cubo de análisis multidimensional OLAP, para plantear ciertas cuestiones y obtener respuestas de los datos. Esta parte se desarrollará con Schema Workbench, un software cliente que se comunica con el módulo Mondrian del servidor Pentaho para hacer análisis de datos.

La instalación de Schema Workbench es muy sencilla, al igual que con spoon, se descomprime una version del programa precompilada en un ordenador cliente, y después se ejecuta un script para arrancar el programa.

El primer paso será establecer la conexión de Schema Workbench con el Data Warehouse. Es una pantalla muy similar a otras conexiones a bbdd que se han hecho desde otras aplicaciones:

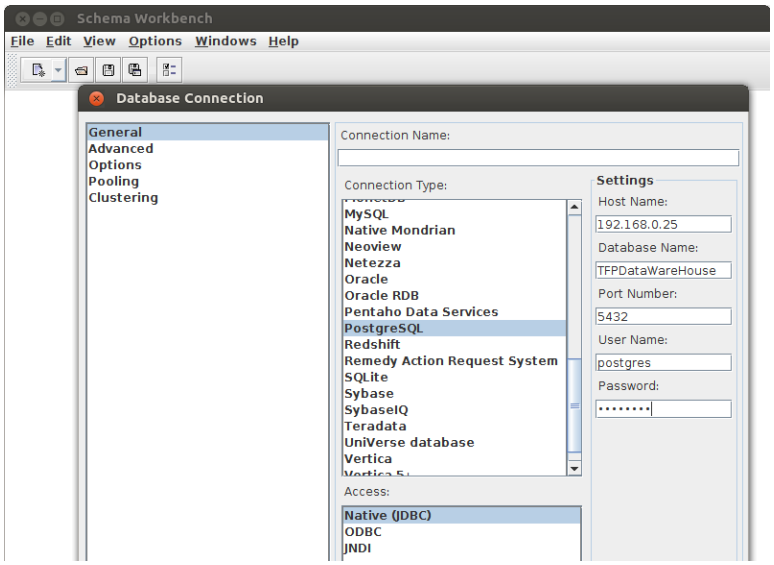


Fig. 38. Conexión al Data Warehouse desde Schema Workbench.

Es importante que para que esta conexión funcione tengamos el driver jdbc en las librerías de schema workbench, al igual que en el servidor Pentaho y en Spoon.

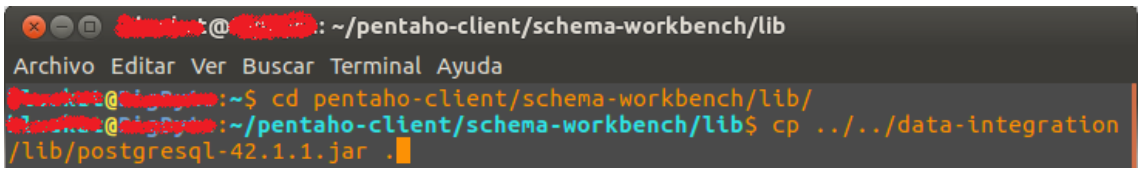


Fig. 39. Driver JDBC.

Cuando creamos un nuevo proyecto partimos de un esquema vacío al que tenemos que poner nombre, en este caso "Esquema Analisis Actividad". A partir de aquí tendremos que añadir un cubo y a este cubo le tendremos que añadir también la tabla de hechos.

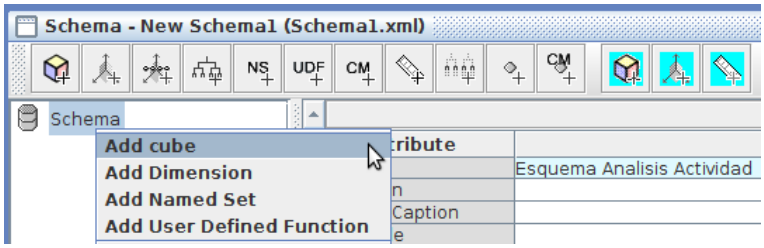


Fig. 40. Añadir Cubo.

Por otro lado tenemos que añadir dimensiones al esquema. Se van a explicar en primer lugar las dimensiones tiempo y curso, ya que tienen jerarquías. En el caso de la dimensión tiempo, tenemos la jerarquía año -> mes -> día y en cuanto a la dimensión curso tenemos la jerarquía categoría -> curso -> tema como se puede ver a continuación:

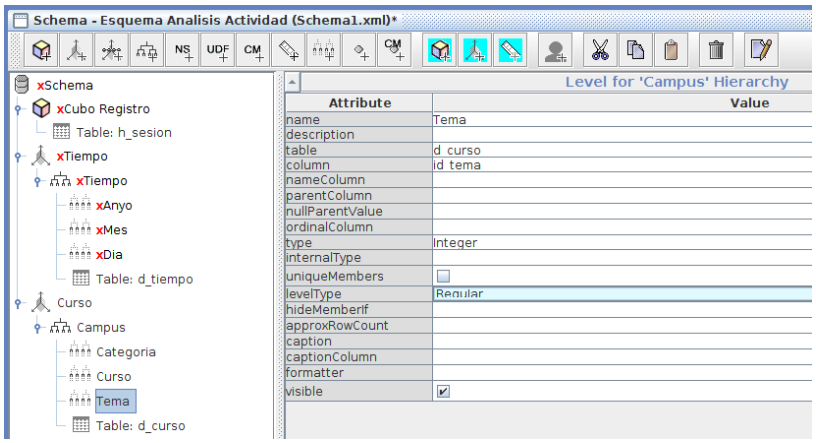


Fig. 41. Jerarquías en las dimensiones.

En las otras dos dimensiones, usuario y sesión, a pesar de no tener distintos niveles, hay que añadir una jerarquía de un único nivel. Cuando ya tenemos un cubo y distintas dimensiones, tenemos que relacionarlas mediante "Add Dimension Usage":

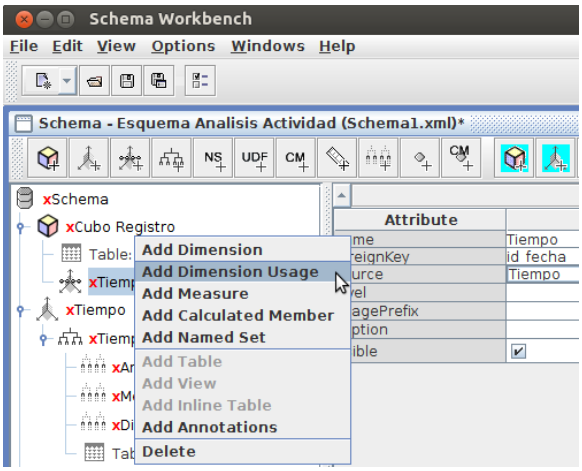


Fig. 42. Añadir dimensiones al cubo.

También tendremos que añadir “medidas” al cubo. A pesar de que se pueden realizar todo tipo de consultas, estas medidas serán el campo principal sobre el que realizar los análisis, y por otro lado, dependiendo de los campos de las dimensiones, podremos ir filtrando el resultado. Estas medidas pueden ser el resultado de alguna operación sobre algún campo, como por ejemplo la media en el caso de la dedicación:

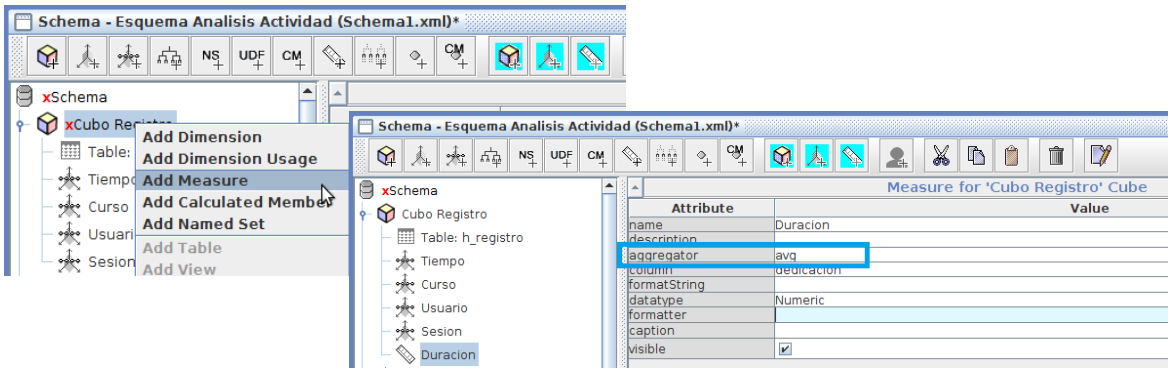


Fig. 43. Añadir métricas al cubo.

Una vez añadidas las medidas tendremos el cubo diseñado por completo. Sólo será necesario guardarlo y publicarlo:

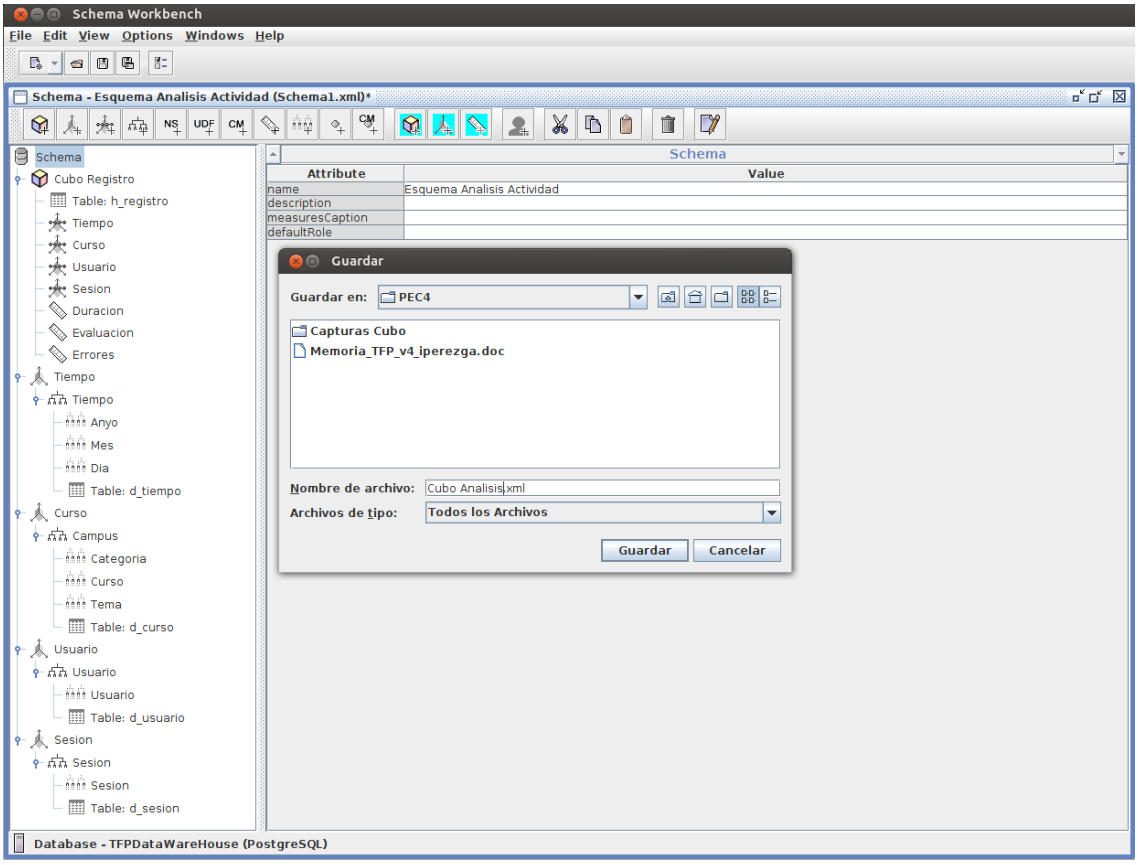


Fig. 44. Diseño del cubo finalizado.

7.2 Creación de un prototipo de modelo de análisis con Saiku Analytics

Hemos llegado a la última parte del proyecto, la creación de un prototipo de análisis, para lo cual nos vamos a apoyar en una herramienta llamada Saiku Analytics. Se trata de un plugin para Pentaho Server de la empresa Meteorite.bi. Nos permitirá manejar el cubo de una forma muy intuitiva y sencilla, proporcionando una gran cantidad de modelos de gráficos para representar la información.

Se trata de una herramienta propietaria, por lo que será necesario descargarlo desde el Marketplace de Pentaho y después hay que registrarse para obtener una licencia que habrá que copiar en una ruta determinada:

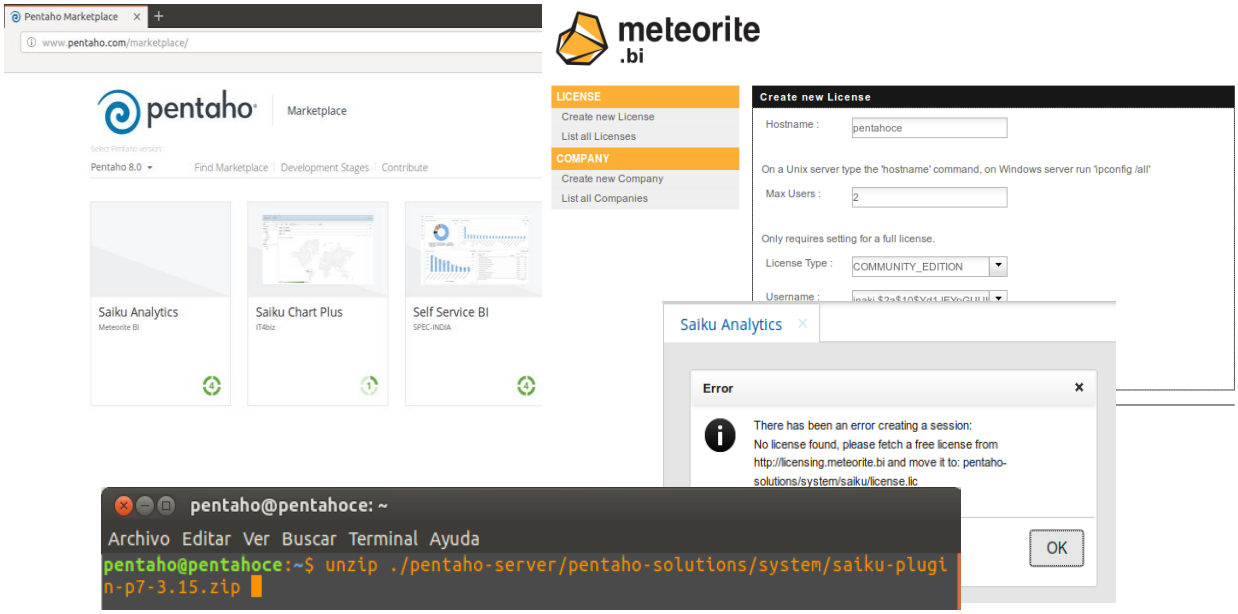


Fig. 45. Saiku en Marketplace y su licencia.

Una vez que hemos hecho los pasos necesario para que la aplicación se inicie si generar errores, tendremos que reiniciar el servidor Pentaho y el plugin estará disponible para crear un nuevo análisis.

Fig. 46. Crear nuevo proyecto de Saiku.

Como hemos dicho al principio, se trata de una herramienta muy intuitiva, únicamente es necesario arrastrar los campos de medidas, arrastrar otros campos que harán de filas y columnas y después, opcionalmente, configurar ciertos filtros:

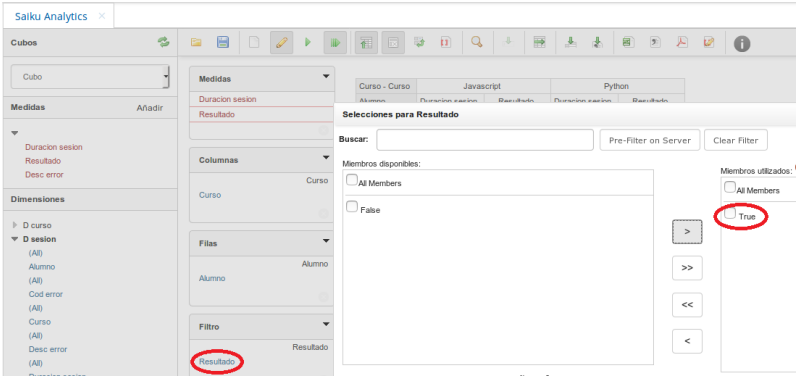


Fig. 47. Filtro por resultado.

En este caso se ha hecho un análisis en el que se analiza la duración de las sesiones de JupyterHub, en los distintos lenguajes de programación, para el caso en el que el resultado es que el ejercicio se ha resuelto correctamente:

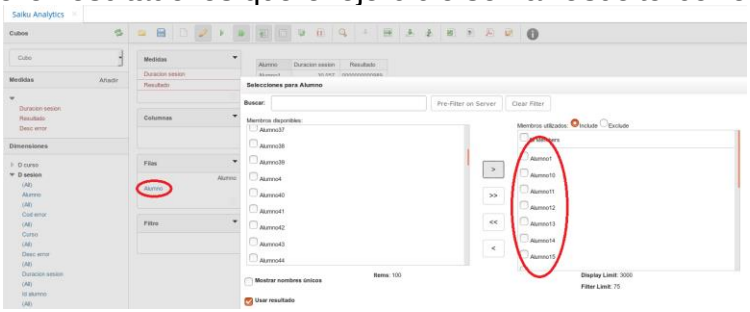
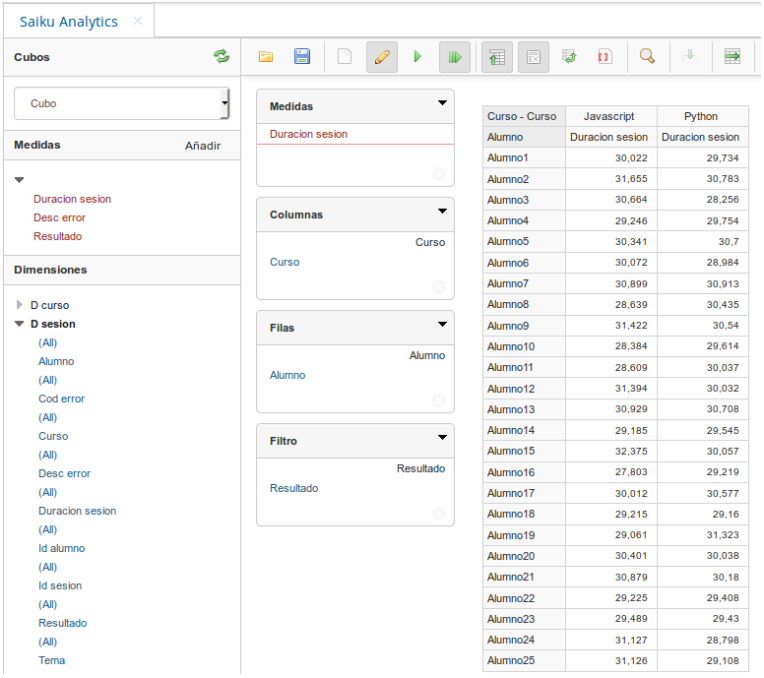


Fig. 48. Miembros del campo alumnos utilizados.

Se realiza el análisis sobre los 25 alumnos que forman parte de uno y otro curso y obtenemos la siguiente tabla de resultado a la consulta:



Curso - Curso	Javascript	Python
Alumno	Duracion sesion	Duracion sesion
Alumno1	30,022	29,734
Alumno2	31,655	30,783
Alumno3	30,664	28,256
Alumno4	29,246	29,754
Alumno5	30,341	30,7
Alumno6	30,072	28,984
Alumno7	30,899	30,913
Alumno8	28,639	30,435
Alumno9	31,422	30,54
Alumno10	28,384	29,614
Alumno11	28,609	30,037
Alumno12	31,394	30,032
Alumno13	30,929	30,708
Alumno14	29,185	29,545
Alumno15	32,375	30,057
Alumno16	27,803	29,219
Alumno17	30,012	30,577
Alumno18	29,215	29,16
Alumno19	29,061	31,323
Alumno20	30,401	30,038
Alumno21	30,879	30,18
Alumno22	29,225	29,408
Alumno23	29,489	29,43
Alumno24	31,127	28,798
Alumno25	31,126	29,108

Fig. 49. Tabla resultado de la consulta.

Una vez obtenidos los resultados podemos representarlos gracias a la gran cantidad de gráficos disponibles en Saiku, además, en todos los casos, dichos gráficos son configurables. Podemos hacer drill down o drill across sobre los datos, cambiar filas por columnas y también son interactivos, ya que nos ofrecen detalles pasando el ratón por las distintas partes de los mismos. En los siguientes ejemplos de consultas se irán mostrando distintos tipos de gráficos.

En este primer caso tenemos un gráfico de barras en el que, como tenemos dos medidas en ela consulta, nos representa la información en la misma barra con distintos colores y por otro lado obtenemos, en el detalle el porcentaje de acierto respecto del total de intentos. El detalle sería distinto si pasáramos por la otra métrica.

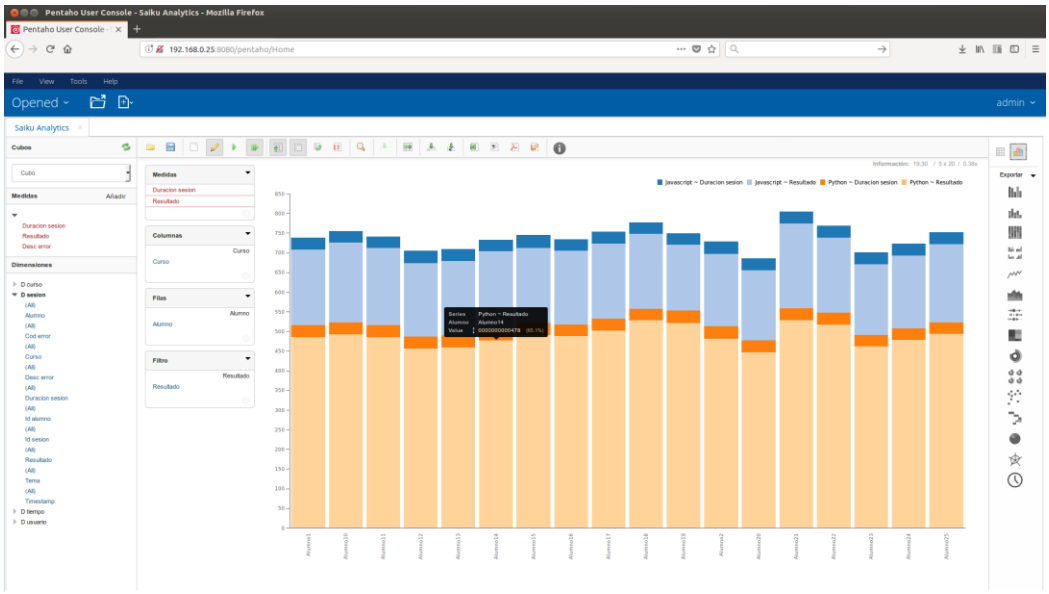


Fig. 50. Gráfico de barras con varias métricas.

Analizando la duración en función del lenguaje de programación/curso, obtenemos un gráfico de tipo radar que resulta interesante:

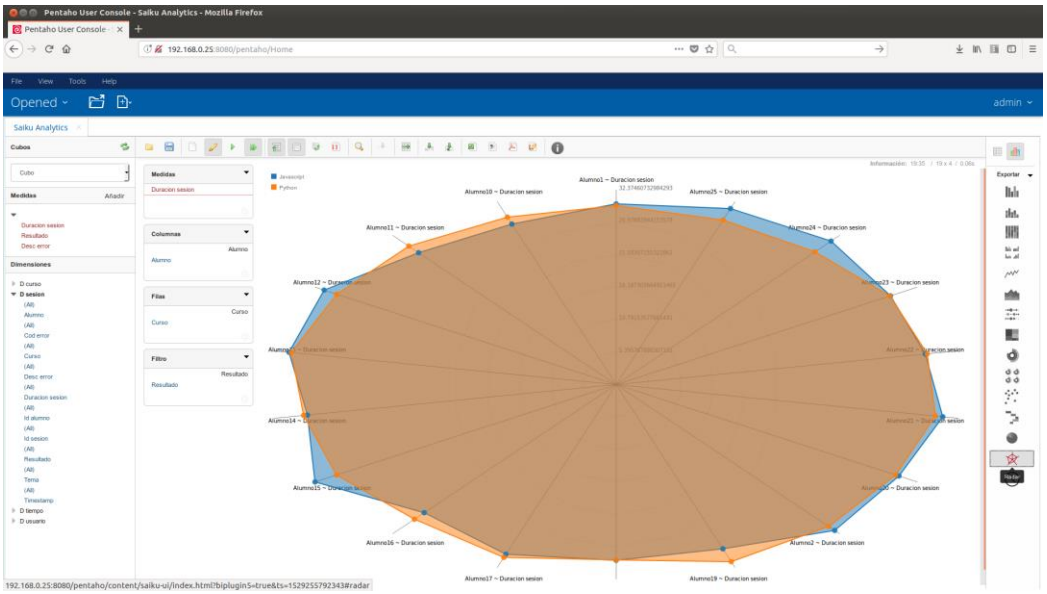
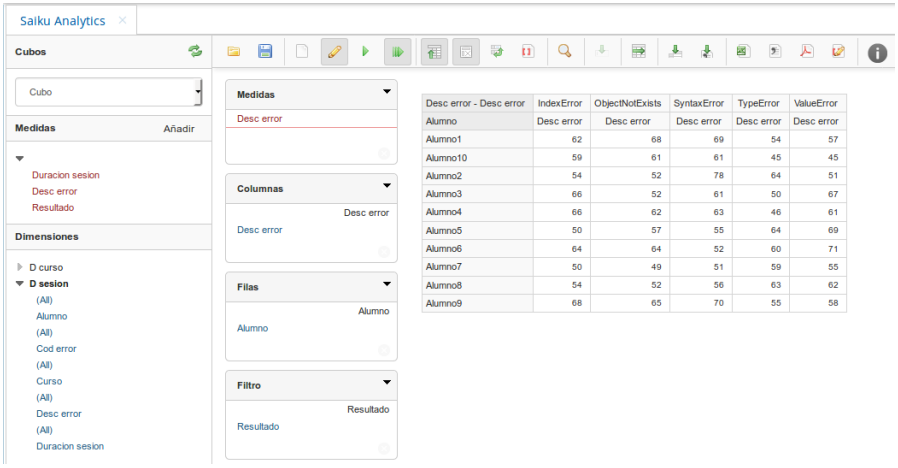


Fig. 51. Gráfico de tipo radar.

En esta segunda consulta, se van a analizar los errores, por lo que cambiamos el filtro resultado a false, es decir, no se ha superado el ejercicio. En este caso tendremos los distintos caso de error, que en una primera instancia representamos como tabla.



Desc error - Desc error	IndexError	ObjectNotExists	SyntaxError	TypeError	ValueError
Alumno	Desc error	Desc error	Desc error	Desc error	Desc error
Alumno1	62	68	69	54	57
Alumno10	59	61	61	45	45
Alumno2	54	52	78	64	51
Alumno3	66	52	61	50	67
Alumno4	66	62	63	46	61
Alumno5	50	57	55	64	69
Alumno6	64	64	52	60	71
Alumno7	50	49	51	59	55
Alumno8	54	52	56	63	62
Alumno9	68	65	70	55	58

Fig. 52. Tabla resultado consulta tipo de error.

A continuación se muestran distintas opciones de representación gráfica de los resultado en función de los distintos tipos de error de esta consulta al cubo. En primer lugar, en un gráfico de barras similar al de la primera consulta, mostramos los procentajes de cada tipo de error de cada alumno.

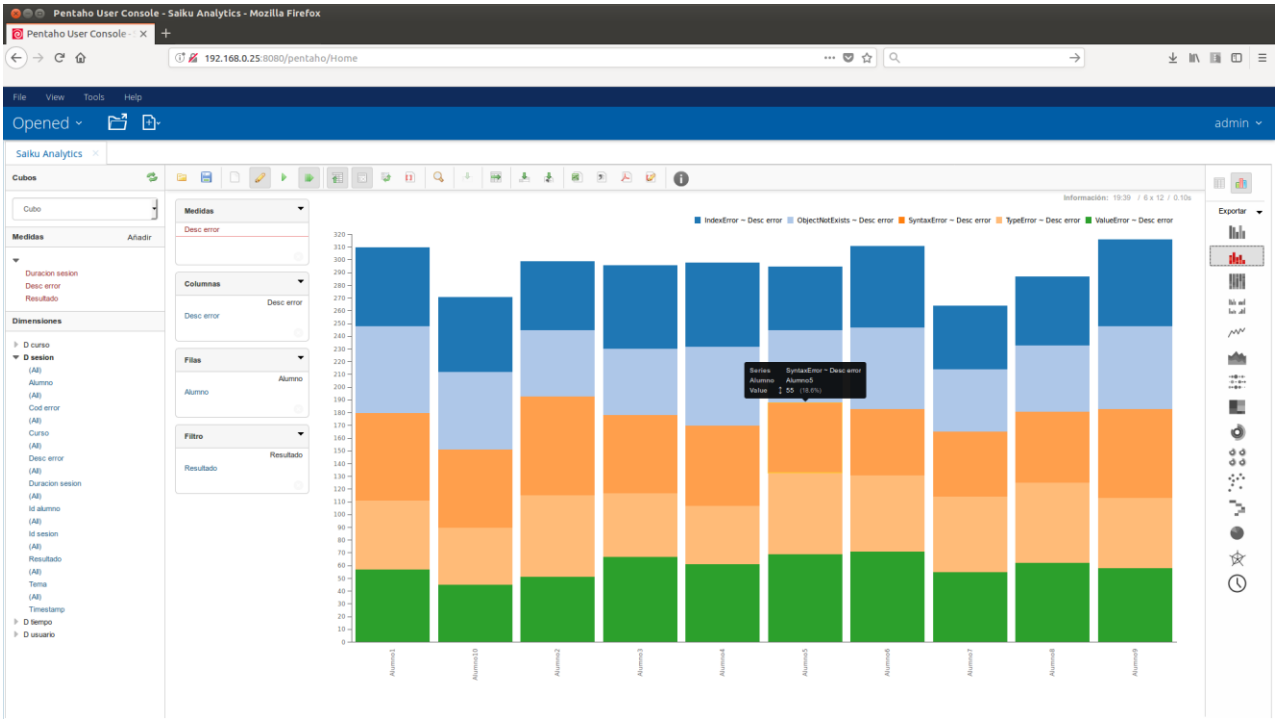


Fig. 52. Tabla resultado consulta tipo de error.

Este gráfico proporciona información detallada “navegando” a través de las barras. En este caso vemos que el alumno 5 tiene un 18% de errores del tipo SyntaxError.

También podemos representarlo con el típico gráfico de forma de queso, pero en este caso intercambiamos filas por columnas, proporcionando aún más información y de una forma muy atractiva. Se aprecia cual es el alumno que más falla en cada tipo de error.

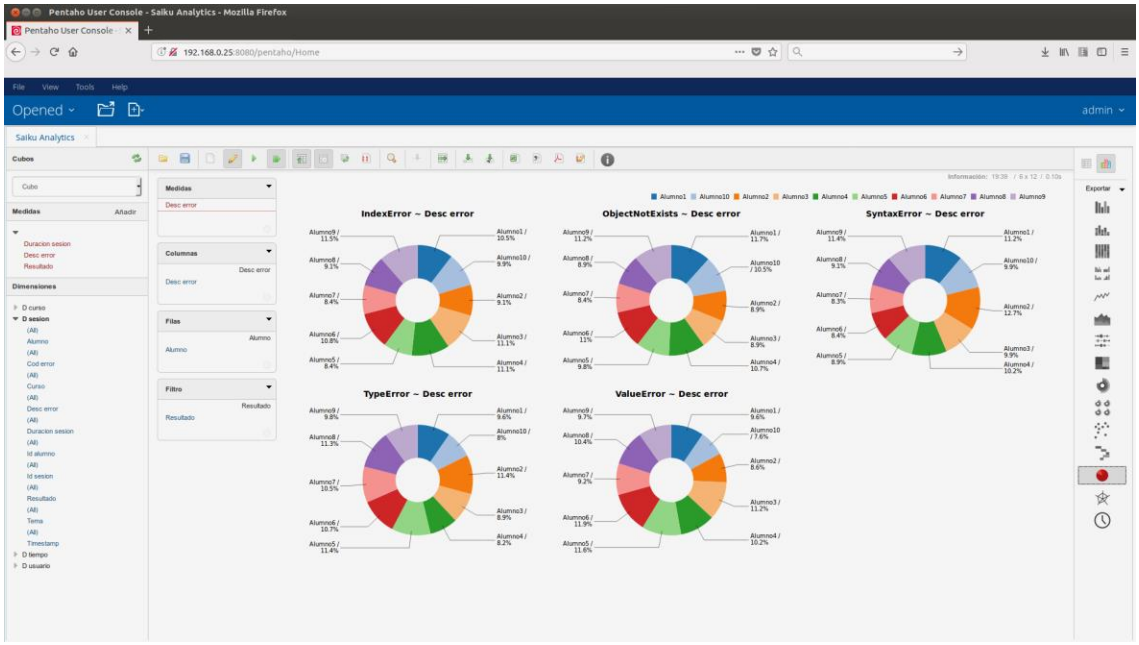


Fig. 53. Gráfico circular cambiando filas por columnas.

Otro tipo de gráfico muy utilizado en análisis de datos es la cuadrícula activa, en la que dependiendo del valor obtenido, varía el tamaño de la representación.

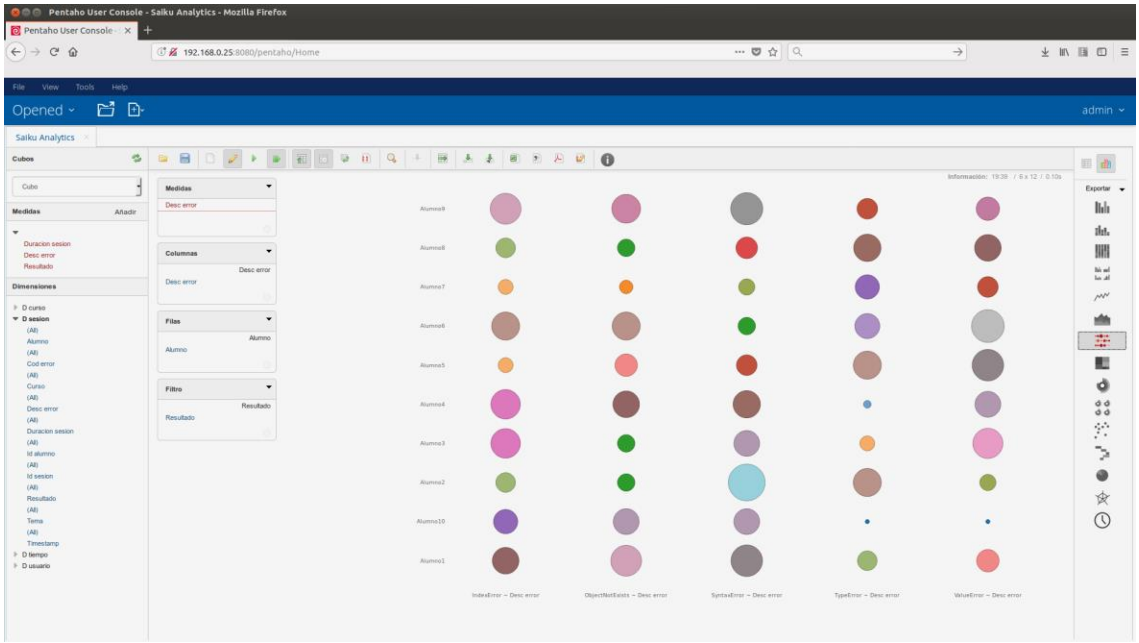


Fig. 54. Gráfico de tipo cuadrícula activa.

Se podrían hacer infinidad de análisis, pero para terminar, se muestran dos últimos gráficos que dan información sobre los porcentajes de error y resultado correcto por cada alumno, y por otra parte un análisis individual de los errores que comete cada alumno.

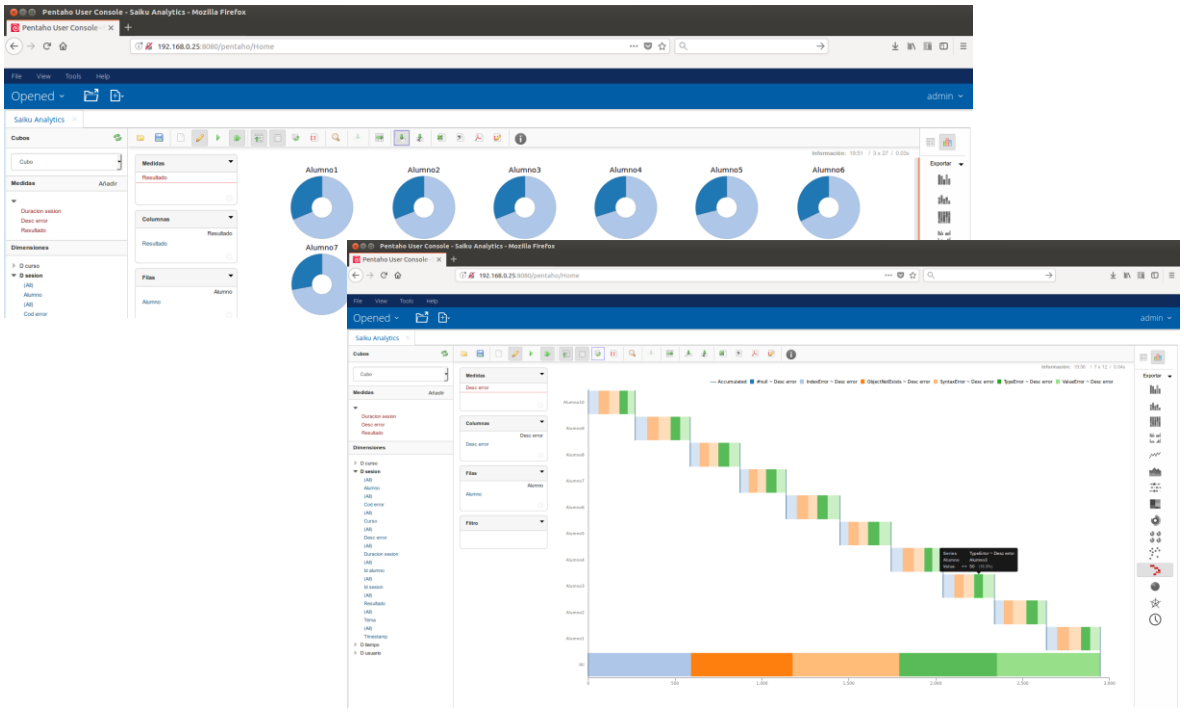


Fig. 55. Otros tipos de gráficos.

8. Trabajo futuro y conclusiones.

8.1 Conclusiones

Más allá de los resultados obtenidos en los análisis del almacén de datos del capítulo anterior, siendo además esta información simulada, se va a valorar en las conclusiones el nivel de cumplimiento de los objetivos planteados al inicio.

A modo de justificación del trabajo se había detectado que en el mercado no existe una solución que permita la gestión de un entorno de eLearning, con capacidades de interactividad entre los usuarios y que además permita el análisis de la información generada por el sistema. En una primera valoración general del proyecto, se puede afirmar que se ha cubierto la necesidad de un producto que cumpla las tres características simultáneamente y por lo tanto se considera que se ha logrado el objetivo principal.

A partir del análisis de los objetivos parciales se desprende que la integración de varios frameworks puede llegar a ser un trabajo arduo y costoso. Una de las partes más complejas de la arquitectura ha sido la instalación de drivers, conectores y el tunning de ciertas configuraciones, para que las distintas herramientas pudieran comunicarse entre ellas y ofrecer la sensación de disponer de una plataforma única e integrada. En este punto la parte positiva ha sido el aprendizaje sobre el funcionamiento de las distintas partes de la arquitectura en modo cluster. Este último punto, de cara a la aplicabilidad en un entorno real, nos proporciona sobre todo escalabilidad.

Se ha seguido prácticamente la planificación, excepto dos semanas en las que por cuestiones de salud no ha sido posible trabajar en el proyecto, pero este contratiempo se ha gestionado correctamente y finalmente se han cumplido todos los hitos.

El proyecto se ha dividido en dos partes, la parte de diseño y configuración de la plataforma de elearning y la parte de análisis y explotación de los datos. Ambas partes han quedado documentadas de forma detallada en distintos anexos, y es otro aspecto positivo de los resultados obtenidos, ya que permitirá su posterior consulta para la aplicación en el proyecto de emprendizaje planteado.

El enfoque del trabajo desde un principio ha sido profesionalizador/empresarial. Se plantea la creación de una microempresa consistente en un club de programación y robótica a modo de extraescolares. En este aspecto se ha dado un primer paso de gigante, ya que se valora como muy positivo disponer de la plataforma completamente configurada para empezar a crear usuarios/alumnos que puedan consultar información en ella y además, que puedan practicar distintos lenguajes de programación de forma tutorizada e interactiva.

Por último, en cuanto a las conclusiones, cabe comentar que se ha conseguido también el objetivo transversal de que todo el software utilizado en el proyecto sea de código abierto. Es un aspecto importante, ya que en el club tecnológico se pretende inculcar esta filosofía, además de la ventaja de que el coste en licencias es nulo.

8.2 Trabajo futuro

El trabajo futuro consistirá principalmente en añadir contenidos a la plataforma y ponerla en funcionamiento en algún colegio como prueba piloto.

En la parte de programación habrá que diseñar ejercicios guiados, pero además habrá que incluir otro tipo de actividades como cuestionarios, talleres, etc. y sobre todo hacer que este contenido sea atractivo de cara a los estudiantes, ya que principalmente serán niños y jovenes menores de edad.

En la parte de robótica, además de las actividades comunes que incluyen también la programación, se ha pensado en configurar algún laboratorio remoto en el que se puedan manejar los dispositivos a través de la plataforma, en un entorno controlado de colaboración.

Para hacer todos estos contenidos más divertidos y que los alumnos se involucren más en las actividades será necesario recurrir a la gamificación. Es uno de los objetivos del proyecto empresarial que ha quedado fuera del proyecto inicial, pero que será imprescincible a futuro.

Fig. 56. Gamificación.

Gracias a la suite de inteligencia de negocio y a partir del análisis de los datos generados por la actividad de los alumnos, se podrán mejorar los contenidos así cómo observar que partes de la gamificación son las que más gustan a los estudiantes para potenciarlas y en definitiva poder ofrecer una formación de calidad.

Por último, en cuanto a la parte de hardware, también queda pendiente el estudio de la mejora del rendimiento de los recursos y la gestión del crecimiento de la plataforma que ya se ha previsto inicialmente y que permite su escalabilidad.

9. Bibliografía

La bibliografía utilizada se basa casi exclusivamente en la información extraida de páginas web, a excepción del siguiente libro:

- Thottuvaikkatumana, Rajanarayanan. Apache Spark 2 for Beginners, Packt Publishing, 2016.

Las referencias a páginas web son las siguientes, se indica la fecha de la primera visita:

- OpenBadges.me (Gamificación) https://www.openbadges.me/ → 01/03/2018
- TurnKey Hub https://hub.turnkeylinux.org/ → 12/03/2018
- Pgpool2 modo Balanceo de carga para PostgreSQL modo replicación https://juantrucupei.wordpress.com/2015/07/17/pgpool2-modo-balanceo-de-carga-con-streaming-replication-en-postgresql-9-4/ → 18/03/2018
- PostgreSQL HA with pgpool-II https://www.fatdragon.me/blog/2016/05/postgresql-ha-pgpool-ii-part-3 → 20/03/2018
- Replication and Load Balance in PosgreSQL 9.4 with PgPool2 https://sonnguyen.ws/replication-load-balance-in-posgresql-replication-with-pgpool2/ → 22/03/2018
- NAU Moodle Cluster https://snapdev.net/2015/10/26/nau-moodle-cluster/ → 26/03/2018
- Moodle en Español: CONFIGURAR MOODLE CON POSTGRESQL https://moodle.org/mod/forum/discuss.php?d=258085 → 09/04/2018
- Moodle in English: Moodle integration with Jupyter notebooks https://moodle.org/mod/forum/discuss.php?d=362641 → 11/04/2018
- Server cluster – MoodleDocs https://docs.moodle.org/34/en/Server_cluster#Step_by_step_guide_for_server_clustering_in_Moodle_2.6 → 12/04/2018
- Spark cluster on OpenStack with multi-user Jupyter Notebook https://arnesund.com/2015/09/21/spark-cluster-on-openstack-with-multi-user-jupyter-notebook/ → 16/04/2018
- Cómo crear un clúster de servidores con Apache Spark https://geekytheory.com/como-crear-un-cluster-de-servidores-con-apache-spark → 17/04/2018
- Montando un entorno multiusuario para Jupyter con JupyterHub https://cetatech.ceta-ciemat.es/2016/05/instalalacion-de-un-entorno-jupyterhub/ → 18/04/2018
- Get Started with PySpark and Jupyter Notebook in 3 Minutes https://blog.sicara.com/get-started-pyspark-jupyter-guide-tutorial-ae2fe84f594f → 20/04/2018
- Jupyter kernels · jupyter/jupyter Wiki · GitHub https://github.com/jupyter/jupyter/wiki/Jupyter-kernels → 23/04/2018

- Examples | Apache Spark
  https://spark.apache.org/examples.html → 25/04/2018
- Using MongoDB with Apache Spark - The Databricks Blog
  https://databricks.com/blog/2015/03/20/using-mongodb-with-spark.html
  → 02/05/2018
- pymongo · PyPI
  https://pypi.org/project/pymongo/ → 03/05/2018
- MongoDB to Postgres - Data Extraction and Loading Instructions
  http://mongodb.topostgres.com/ → 04/05/2018
- MongoDB to PostgreSQL in minutes | Stitch
  https://www.stitchdata.com/integrations/mongodb/postgresql/?utm_source=topostgres.com&utm_medium=microsite-referral&utm_campaign=microsite-referral+mongodb-topostgres.com
  → 05/05/2018
- Python Tutorial: MongoDB with pyMongo I – 2018
  http://www.bogotobogo.com/python/MongoDB_PyMongo/python_MongoDB_pyMongo_tutorial_installing.php → 22/05/2018
- The Jupyter Notebook Format — Estructura JSON
  https://ipython.org/ipython-doc/3/notebook/nbformat.html → 23/05/2018
- Installing Pentaho CE 8.0 with PostgreSQL on Ubuntu Server 16.04 LTS
  http://nunobarreiro.com/2018/01/22/Installing-Pentaho-CE-8-0-with-PostgreSQL-on-an-Ubuntu-Server-16-04-LTS/ → 27/05/2018
- Instalación Pentaho biserver 3.8 en Linux y base PostgreSQL
  https://docs.google.com/document/d/1YBXNtLwtcRBfqlcX45qftTxedf6KLuGOIKY1-GKHToE/edit → 27/05/2018
- Pentaho con PostgreSQL
  https://forums.pentaho.com/threads/152221-Pentaho-con-PostgreSQL/
  → 01/06/2018
- Pentaho | Hitachi Vantara Community
  https://community.hitachivantara.com/community/products-and-solutions/pentaho/ → 04/06/2018
- OLAP OK Schema Workbench. Installation of Pentaho components and setting up connections
  https://wiki.civicrm.org/confluence/display/CRMDOC/Installation+of+Pentaho+components+and+setting+up+connections → 08/06/2018
- Creación de los cubos OLAP Mondrian con el Pentaho Cube Designer
  http://pentaho.almacen-datos.com/cube-designer.html → 11/06/2018

10. Anexos

A1. Instalación y configuración del cluster PostgreSQL.

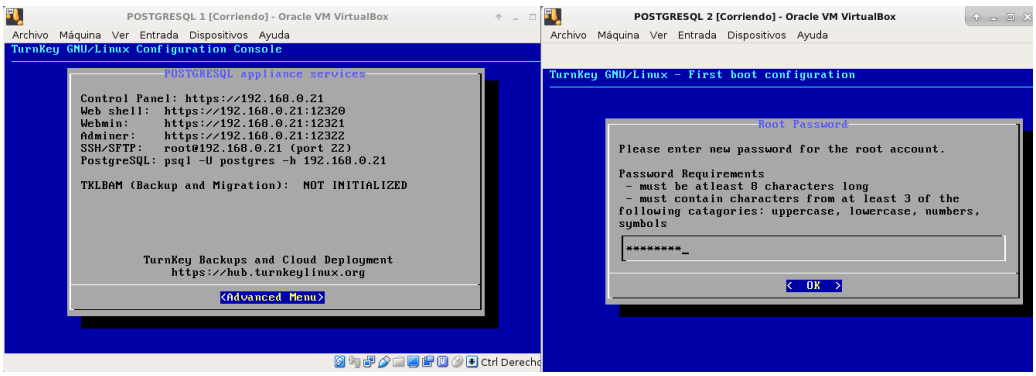
PostgreSQL es una base de datos relacional de código abierto que se distribuye bajo licencia BSD. Dispone de versiones para prácticamente todos los sistemas operativos y cumple totalmente con ACID (del inglés, Atomicity, Consistency, Isolation, Durability). Tiene soporte para claves extranjeras, joins, vistas, disparadores y procedimientos almacenados.

PostgreSQL ofrece características tales como replicación asíncrona y write ahead logging para ser tolerante a fallos de hardware y del que se hace uso en este proyecto. Soporta juegos de caracteres internacionales, codificaciones de caracteres multibyte, Unicode y es altamente escalable tanto en la cantidad bruta de datos que puede manejar como en el número de usuarios concurrentes que puede atender. Hay sistemas activos en producción con PostgreSQL que manejan más de 4 terabytes de datos.

A1.1 Instalación y configuración de Turnkey Linux con imagen Postgresql

Turnkey Linux es un proyecto a través del cual se distribuyen dispositivos virtuales libre, que permiten la implementación (llave en mano) de plataformas tecnológicas en máquinas virtuales, en pocos minutos. En este caso se ha optado por una imagen que incluye un servidor Postgresql en su versión 9.4

Se han instalado dos servidores postgresql en dos máquinas virtuales para configurar el cluster, la réplica de datos y el balanceo de carga y que en el futuro el sistema sea escalable pudiendo añadir más servidores fácilmente. El primer paso es importar el servicio virtualizado en Virtualbox, y una vez arrancada la máquina habrá que configurar las contraseñas para el usuario “root” y el usuario “postgres”:



Se puede observar en la captura anterior que existen ciertos servicios, como por ejemplo Webmin, SSH o webshell, que se proporcionan por defecto en cualquier distribución Turnkey. Se trata de herramientas que facilitan la administración de los sistemas muy cómodamente.

A1.2 Configuración de Postgres en modo cluster y con replicado

Para que los dos servidores funcionen en modo cluster es necesario hacer ciertas modificaciones en varios archivos de configuración. El primero de ellos, que se trata del archivo de configuración principal, es postgresql.conf. Lo modificaremos para permitir "wal" (Write ahead logging):

```
#------------------------------------------------------------------------------
# WRITE AHEAD LOG
#------------------------------------------------------------------------------

# - Settings -

wal_level = hot_standby                 # minimal, archive, hot_standby, or logical
                                        # (change requires restart)
```

Otras configuraciones que se han retocado para permitir la replicación son las que aparecen a continuación seleccionadas:

```
# - Archiving -

archive_mode = on               # allows archiving to be done
                                # (change requires restart)
archive_command = 'scp /BBDD/rel/%p 192.168.0.26:/BBDD/wal/%f'          # command
                                # placeholders: %p = path of file to archive
                                #               %f = file name only
                                # e.g. 'test ! -f /mnt/server/archivedir/%f && cp
#archive_timeout = 0            # force a logfile segment switch after this
                                # number of seconds; 0 disables


#------------------------------------------------------------------------------
# REPLICATION
#------------------------------------------------------------------------------

# - Sending Server(s) -

# Set these on the master and on any standby that will send replication data.

max_wal_senders = 1             # max number of walsender processes
                                # (change requires restart)
wal_keep_segments = 32          # in logfile segments, 16MB each; 0 disables
#wal_sender_timeout = 60s       # in milliseconds; 0 disables
```

Ahora tendremos que crear una clave publica en el servidor maestro para posteriormente copiarla en el servidor esclavo y que se permita la comunicación entre ambos sin necesidad de estar autenticando cada vez que se haga una conexión. Creamos la clave pública:

```
postgres@postgresql:/etc/postgresql/9.4/main$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/var/lib/postgresql/.ssh/id_dsa):
/var/lib/postgresql/.ssh/id_dsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/postgresql/.ssh/id_dsa.
Your public key has been saved in /var/lib/postgresql/.ssh/id_dsa.pub.
The key fingerprint is:
cd:87:91:72:b7:cb:ca:de:b0:72:78:45:78:d7:21:df postgres@postgresql
The key's randomart image is:
+---[DSA 1024]----+
```

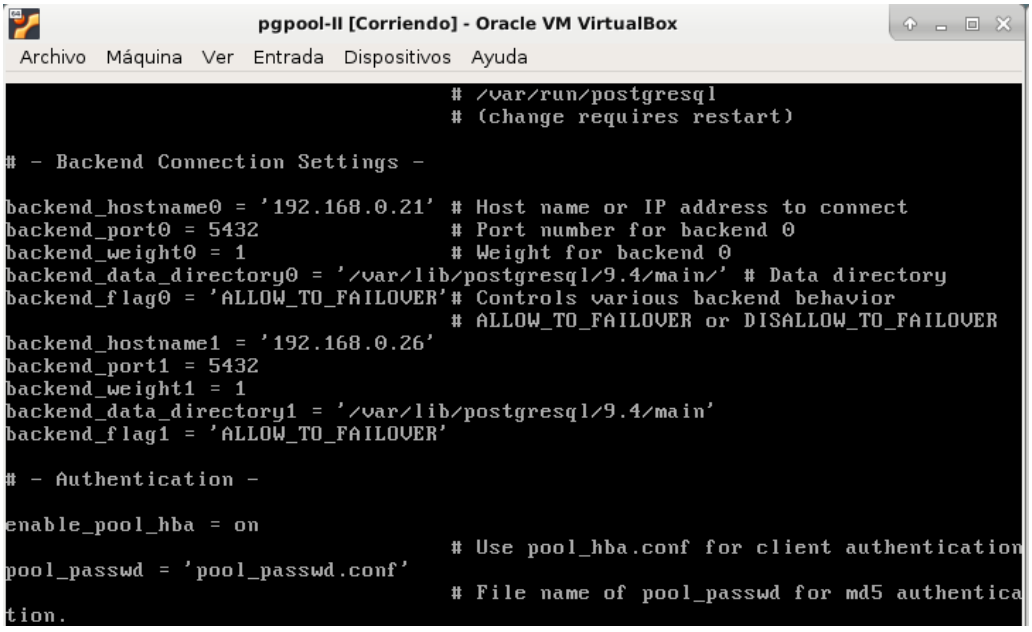
Y procedemos a enviarla al otro servidor:

```
postgres@postgresql:/etc/postgresql/9.4/main$ ssh-copy-id -i /var/lib/postgresql/.ssh/id_dsa.pub postgres@192.168.0.26
The authenticity of host '192.168.0.26 (192.168.0.26)' can't be established.
ECDSA key fingerprint is c2:a8:53:b8:8c:cd:d7:b5:2b:59:35:18:b2:34:5a:ac.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
postgres@192.168.0.26's password:
Permission denied, please try again.
postgres@192.168.0.26's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'postgres@192.168.0.26'"
and check to make sure that only the key(s) you wanted were added.
```

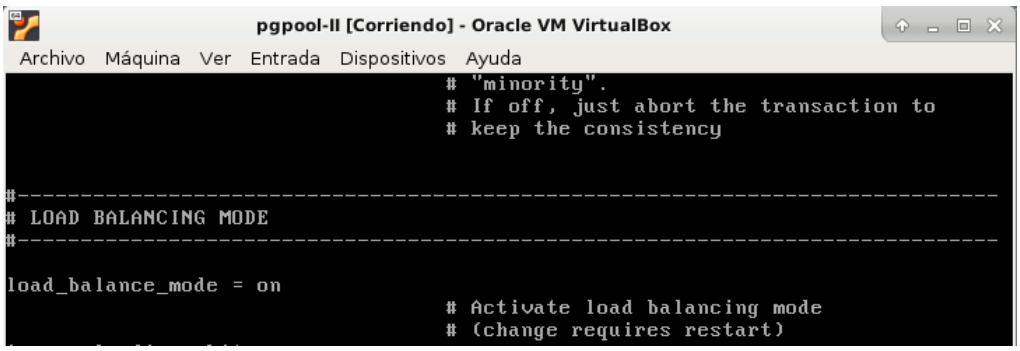
Por último modificaremos otro archivo de configuración, en este caso pg_hba.conf, tanto en el servidor maestro como en el esclavo para permitir las conexiones para la replica:

```
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                peer
host    replication     postgres        192.168.0.26/32         md5
```

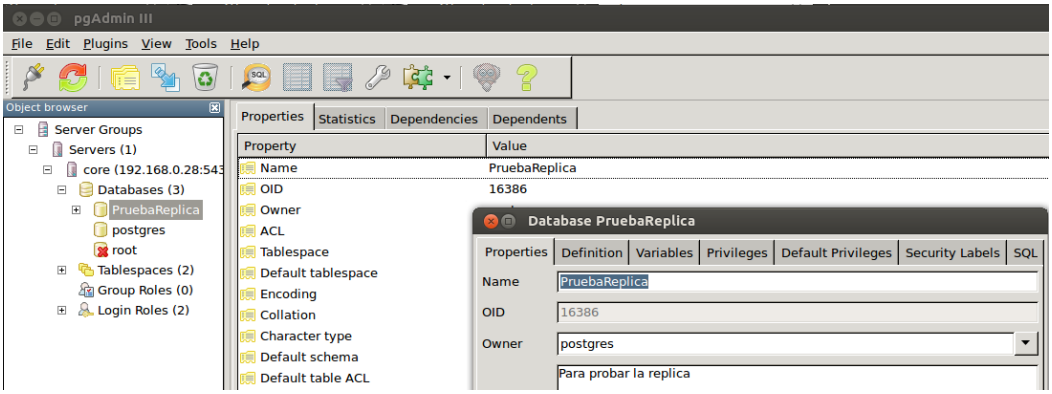
Una vez instalado pgpool-ii en un servidor Turnkey core, procedemos a configurarlo para nuestros servidores postgres:



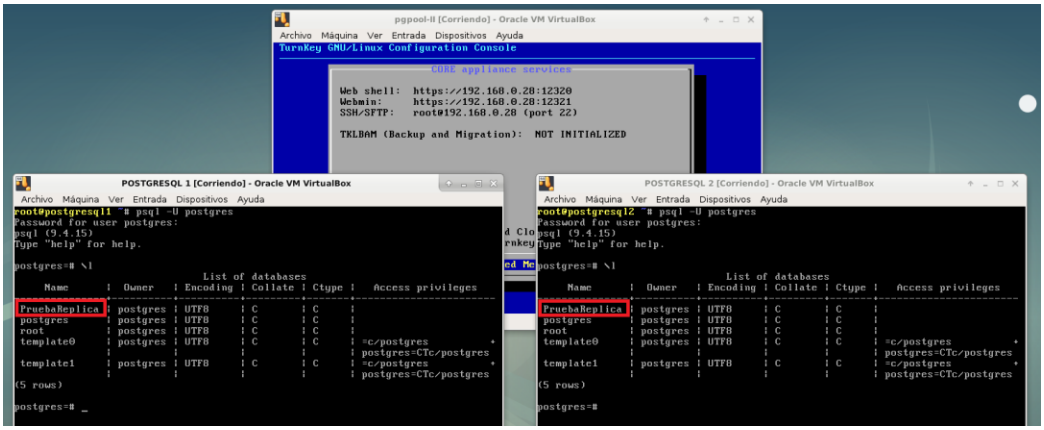
En ese mismo archivo editamos la opción de balanceo de carga:



Ayudandonos de pgAdmin3, el administrador de bbdd para postgresql instalado sobre un cliente ubuntu 16.04, creamos una tabla de prueba, para comprobar que la réplica se hace correctamente, conectándonos al equipo de pgpool-ii:



Vemos a continuación, conectándonos por consola en los propios servidores, que la tabla se ha replicado correctamente en el esclavo:

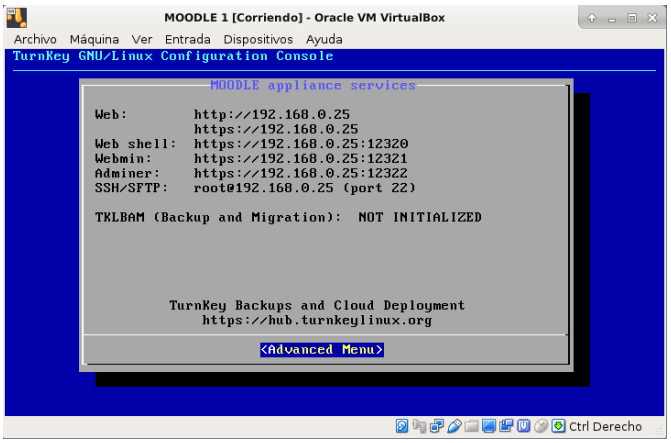


A2. Instalación y configuración del cluster Moodle.

A2.1 Instalación y configuración de Turnkey Linux con imagen Moodle

Al igual que en el caso de los servidores de bbdd anterior, se han instalado dos servidores Turnkey en dos máquinas virtuales para configurar el cluster, en este caso con una imagen con Moodle preconfigurado.

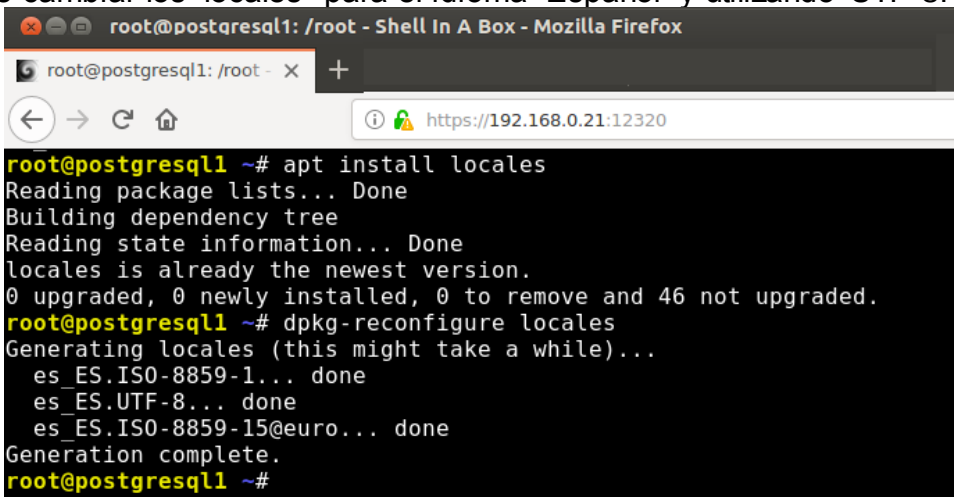
Al importar el servicio a la máquina virtual y arrancar la primera vez, nos pide también en este caso un usuario administrador "root" y otro usuario "admin" para Moodle. También tendremos acceso vía webshell, ssh, etc.



A2.2 Configuración de Moodle con Postgres como base de datos.

La imagen de Turnkey Linux viene con Moodle configurada por defecto para trabajar con una base de datos MYSQL. A continuación se presentan los pasos para crear una nueva base de datos con Postgresql, crear el usuario moodleuser y configurar Moodle para que la utilice.

El primer paso es arreglar un tema de configuración regional. Tenemos que cambiar los "locales" para el idioma "Español" y utilizando UTF-8:



Con el comando reconfigure anterior elegiremos los locales en castellano:

```
[*] es_ES ISO-8859-1
[*] es_ES.UTF-8 UTF-8
[*] es_ES@euro ISO-8859-15
```

Ahora conectamos a Postgresql con el usuario postgres y creamos la base de datos. Si listamos las tablas vemos la tabla moodle y como propietario a moodleuser:

```
root@postgresql1 ~# psql -U postgres
Password for user postgres:
psql (9.4.15)
Type "help" for help.

postgres=# CREATE DATABASE moodle WITH OWNER moodleuser ENCODING 'UTF8' LC_COLLATE='es_ES.UTF-8' LC_CTYPE='es_ES.UTF-8' TEMPLATE=template0;
CREATE DATABASE
postgres=# \l
                                  List of databases
   Name    |   Owner    | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+------------+----------+-------------+-------------+-----------------------
 moodle    | moodleuser | UTF8     | es_ES.UTF-8 | es_ES.UTF-8 |
 postgres  | postgres   | UTF8     | C           | C           |
 root      | postgres   | UTF8     | C           | C           |
 template0 | postgres   | UTF8     | C           | C           | =c/postgres          +
           |            |          |             |             | postgres=CTc/postgres
 template1 | postgres   | UTF8     | C           | C           | =c/postgres          +
           |            |          |             |             | postgres=CTc/postgres
(5 rows)

postgres=#
```

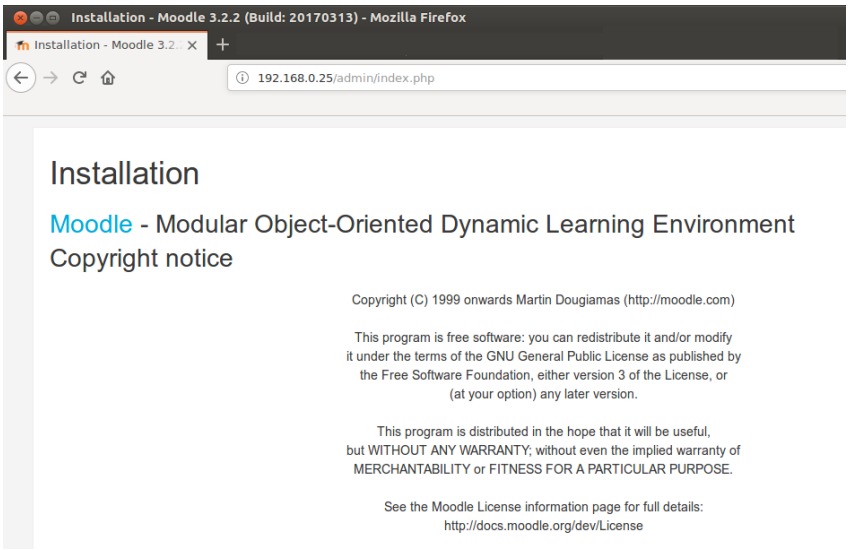
Para que el servidor moodle haga uso de esta tabla tendremos que modificar el archivo config.php, en el que los parámetros más importantes son la ip del balanceador pgpool-ii, que hace de frontend del cluster postgresql, nombre de la base de datos, usuario y contraseña y el directorio de los datos.

```
<?php  // Moodle configuration file

unset($CFG);
global $CFG;
$CFG = new stdClass();

$CFG->dbtype    = 'pgsql';
$CFG->dblibrary = 'native';
$CFG->dbhost    = '192.168.0.21';
$CFG->dbname    = 'moodle';
$CFG->dbuser    = 'moodleuser';
$CFG->dbpass = 'd6f27a1cdeeb97a6b7ee953fe392b8d6';
$CFG->prefix    = 'mdl_';
$CFG->dboptions = array (
  'dbpersist' => 0,
  'dbport' => '',
  'dbsocket' => '',
  'dbcollation' => 'utf8_general_ci',
);

$protocol='http://';
$hostname='192.168.0.25';
if (isset($_SERVER['HTTPS'])) { $protocol='https://'; }
if (isset($_SERVER['HTTP_HOST'])) { $hostname=$_SERVER['HTTP_HOST']; }
$CFG->wwwroot = $protocol.$hostname;

$CFG->dataroot  = '/var/www/moodledata';
$CFG->admin     = 'admin';

$CFG->directorypermissions = 0750;

require_once(__DIR__ . '/lib/setup.php');
```

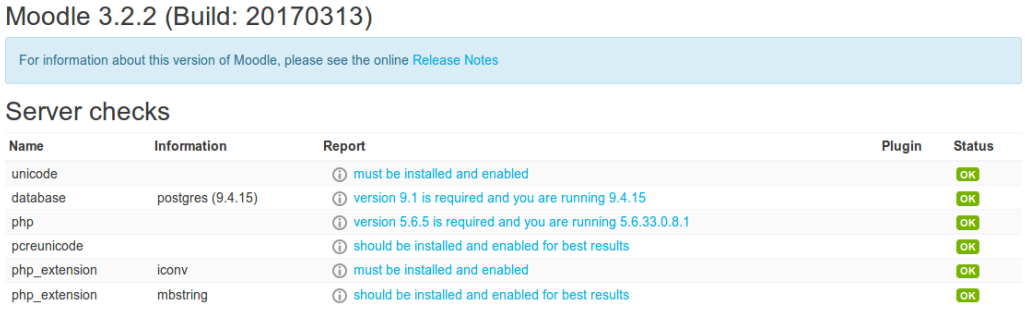
Como se puede ver en la última línea, en la primera ejecución se abrirá un script de configuración. Antes de conseguir ejecutar este script se han obtenido dos errores, el prefijo para las tablas que se creen no puede estar vacío, por lo que se indica “mdl_” y es necesario instalar php.

Cuando consigamos ejecutar el asistente de configuración del primer inicio de Moodle, sólo habrá que seguirlo y configurar los parámetros a nuestro gusto:

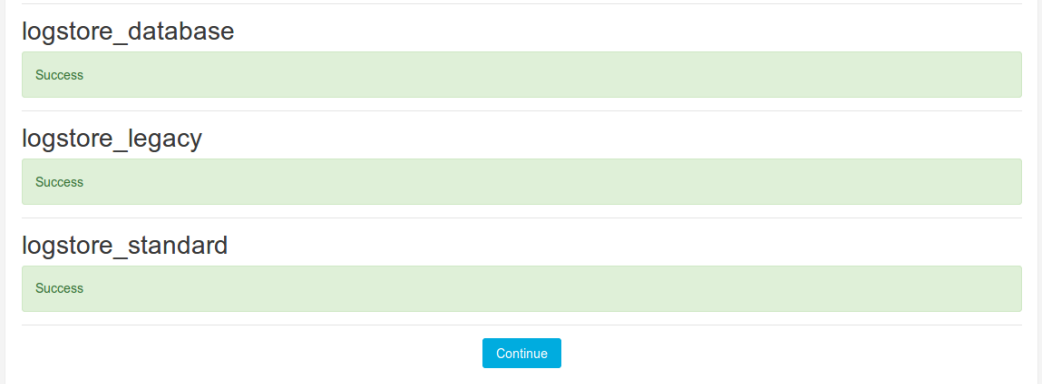
6) Aceptar condiciones y licencia:



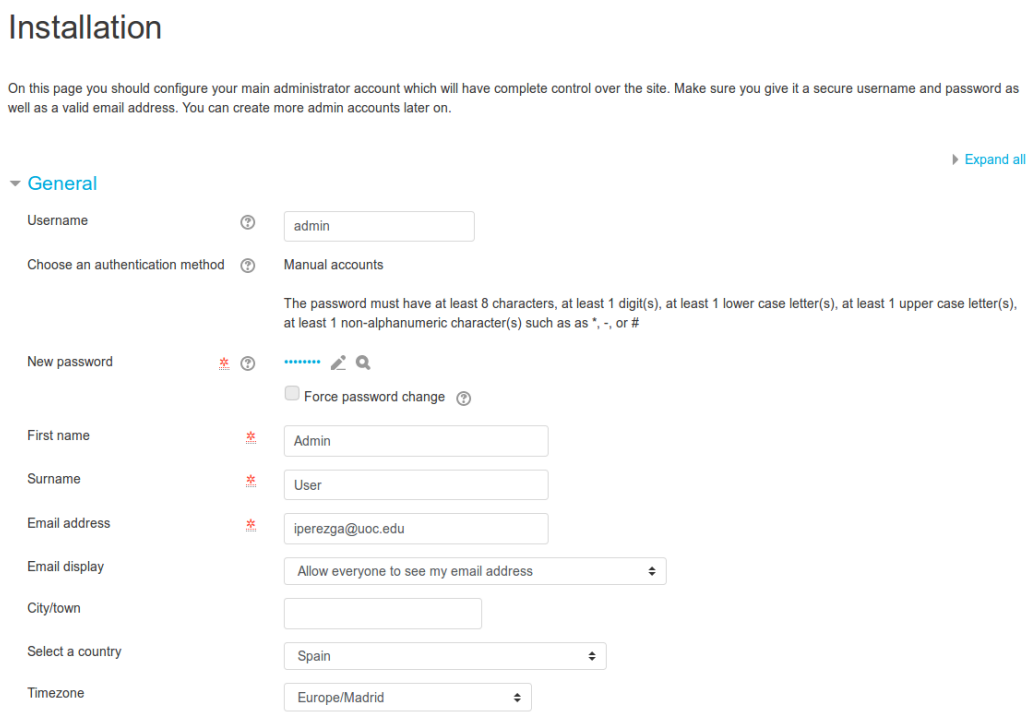
7) Comprobar dependencias:



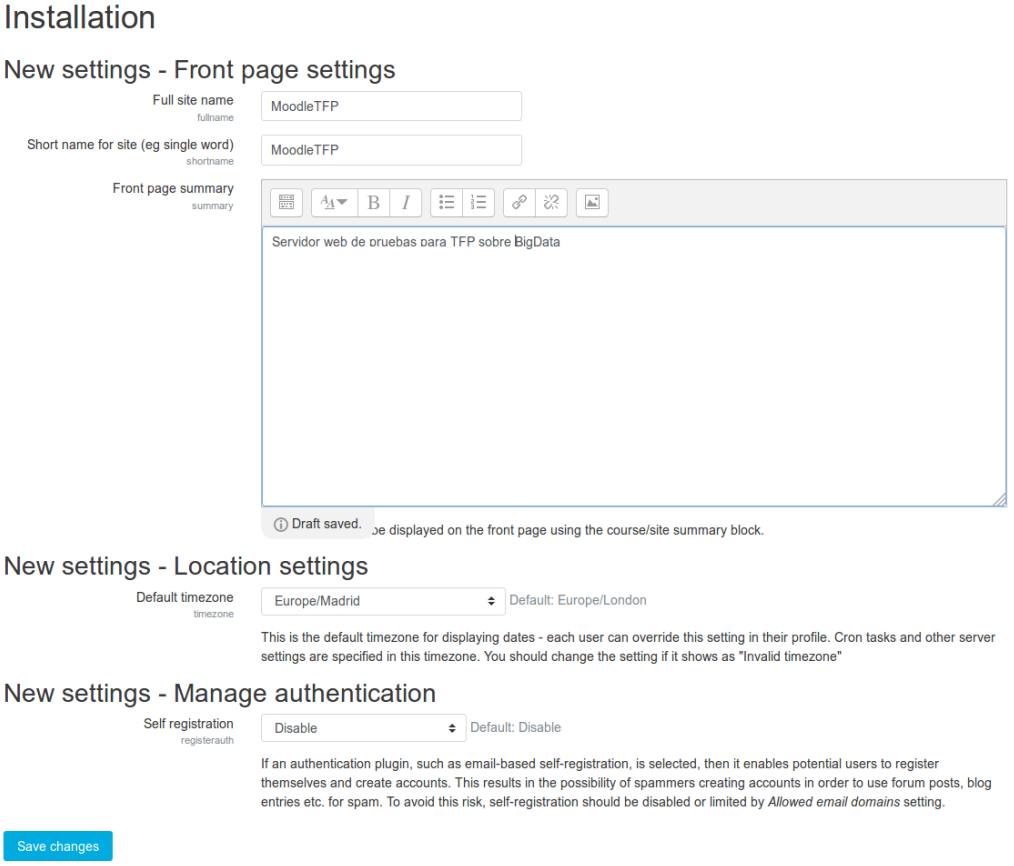
8) Creación de la estructura de la base de datos:



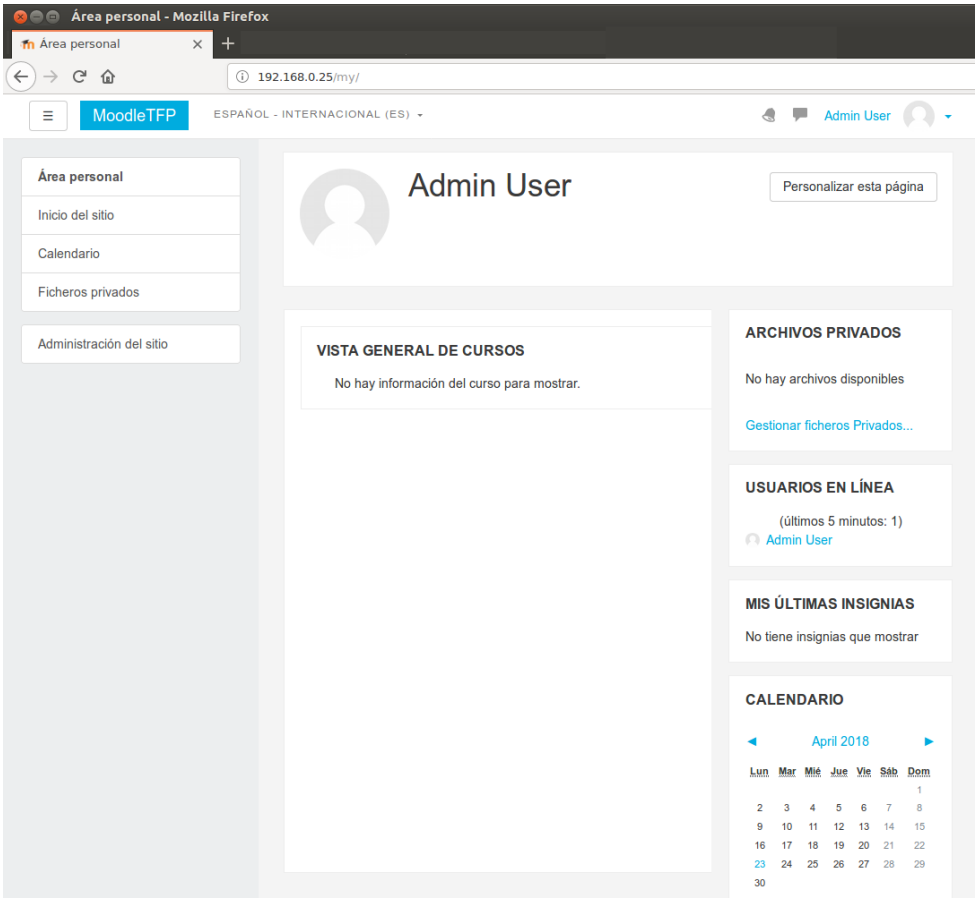
9) Crear el usuario administrador:



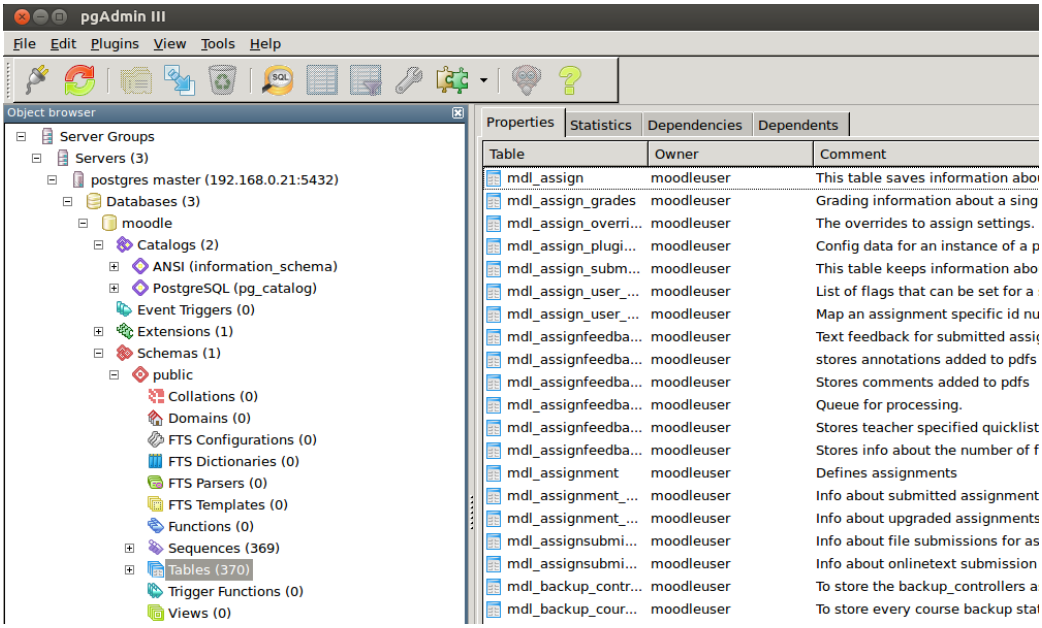
10) Último paso, configuración de la página principal:



Con esto habremos configurado Moodle con la base de datos Postgresql y podemos ejecutar por primera vez el portal de eLearning:



Comprobamos con el cliente pgAdmin3 que se han creado las tablas necesarias en la base de datos. Hay un total de 370 tablas y se puede observar una pequeña descripción de su función:



A3. Instalación y configuración del cluster Spark.

A3.1 Instalación y configuración de Ubuntu Server 16.04 y las dependencias

En este caso no existe una imagne preconfigurada en el proyecto Turnkey que incluya el servidor Spark, por lo tanto es necesario instalar en primer lugar el sistema operativo. El elegido ha sido Ubuntu 16.4 en su versión server. Se trata de la version LTS, y a pesar de que existe una nueva verión con soporte de larga duración, la 18.04, prefiero hacer uso de esta versión más probada.

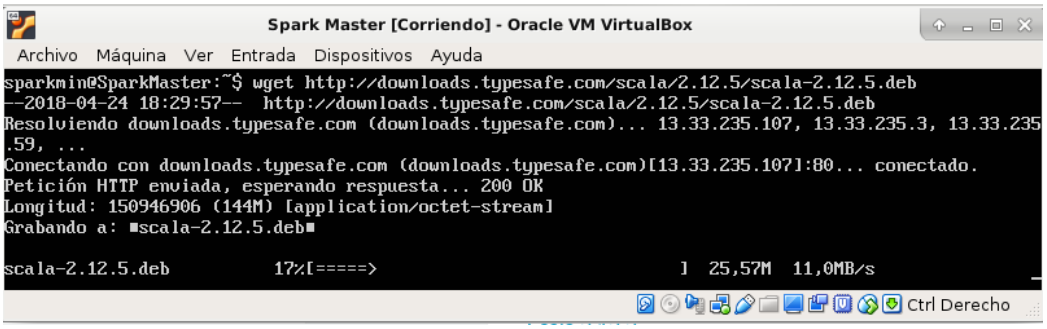
Instalamos el sistema base, en la versión server cabe recordar que no existe escritorio, y el primer paso es elegir un nombre para la máquina, será SparkMaster. Después para el correcto funcionamiento de Spark, es necesario tener instalado tanto java, como Scala. En el caso de java podemos elegir entre OpenJDK o la versión de Oracle. Para descargar esta última hay que añadir los repositorios:

```
sparkmin@SparkMaster:~$ sudo apt-add-repository ppa:webupd8team/java
```

Una vez instalado comprobamos la versión:

```
sparkmin@SparkMaster:~$ java -version
java version "1.8.0_171"
Java(TM) SE Runtime Environment (build 1.8.0_171-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)
sparkmin@SparkMaster:~$ _
```

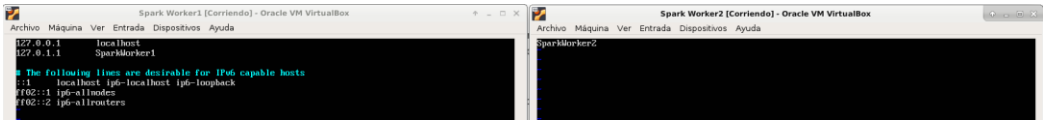
El otro lenguaje del que tenemos que tener soporte es Scala. Se trata del lenguaje nativo de Spark, por lo que es fundamental que esté instalado. Elegimos la versión y la descargamos con wget e instalamos:




```
sparkmin@SparkMaster:~$ sudo dpkg -i scala-2.12.5.deb
Seleccionando el paquete scala previamente no seleccionado.
(Leyendo la base de datos ... 58012 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar scala-2.12.5.deb ...
Desempaquetando scala (2.12.5-400) ...
Configurando scala (2.12.5-400) ...
Creating system group: scala
Creating system user: scala in scala with scala daemon-user and shell /bin/false
Procesando disparadores para man-db (2.7.5-1) ...
```

Una vez que tenemos el sistema base instalado, es el momento de clonar la máquina del servidor master para obtener los esclavos. En el caso de Spark estos esclavos se llaman workers. La clonación se realiza de una forma muy sencilla utilizando las herramientas de VirtualBox. Cabe destacar que se va a crear una clonación enlazada:

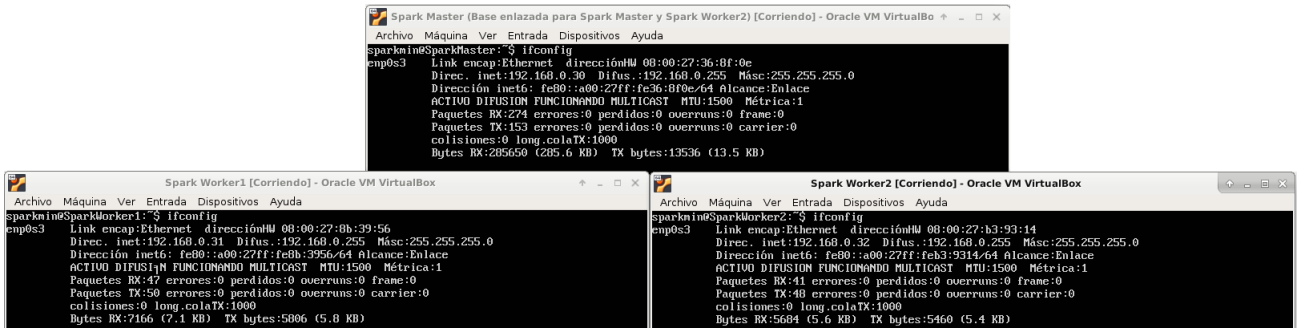
El último paso para dejar el sistema base preparado es cambiar el nombre a estas dos nuevas máquinas, así como sus tarjetas de red, para que no sean exactamente la misma que el servidor Master. Es necesario cambiar tanto el archivo /etc/hosts como /etc/hostname:

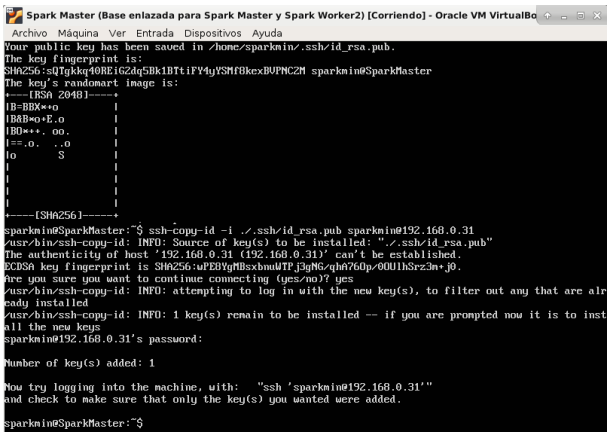


Como se trata de máquinas virtuales, es posible cambiar la dirección física de las tarjetas de red. Además aprovechamos la ocasión para indicar que se trata de conexiones puentes y de esta manera todos los equipos estarán en la misma red:

A3.2 Instalación y configuración de Apache Spark

Ahora vamos a proceder a instalar Spark tanto en el servidor Master como en los “workers”, por lo tanto vamos a necesitar conectarnos de uno a otro, por lo que creamos una clave púlbica y una privada y así no es necesario introducir la contraseña en cada ocasión. Verificamos las dirección ip del cluster y enviamos la clave pública:





Se puede descargar el código fuente completo de Spark y compilarlo, pero como depende de una instalación base de Hadoop, y existe una versión precompilada que incluye las dos tecnologías, se opta por descargar esta opción, más cómoda y que seguramente generará menos fallos, sólo es necesario decomprimir y listo. Las versiones son Spark 2.2 y Hadoop 2.6:

```
sparkmin@SparkMaster:~$ wget http://apache.rediris.es/spark/spark-2.2.0/spark-2.2.0-bin-hadoop2.6.tg
z
--2018-04-24 19:16:29--  http://apache.rediris.es/spark/spark-2.2.0/spark-2.2.0-bin-hadoop2.6.tgz
Resolviendo apache.rediris.es (apache.rediris.es)... 130.206.13.2, 2001:720:418:cafd::2
Conectando con apache.rediris.es (apache.rediris.es)[130.206.13.2]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 201706782 (192M) [application/x-gzip]
Grabando a: ■spark-2.2.0-bin-hadoop2.6.tgz■

spark-2.2.0-bin-hadoop2. 100%[==================================>] 192,36M  11,2MB/s    in 17s

2018-04-24 19:16:46 (11,0 MB/s) - ■spark-2.2.0-bin-hadoop2.6.tgz■ guardado [201706782/201706782]

sparkmin@SparkMaster:~$ ls
scala-2.12.5.deb  spark-2.2.0-bin-hadoop2.6.tgz
sparkmin@SparkMaster:~$ tar -xvzf spark-2.2.0-bin-hadoop2.6.tgz _
```

Se comprueba el correcto funcionamiento del servidor ejecutando una instancia de la Shell de Spark:

```
sparkmin@SparkMaster:~$ ./bin/spark-shell
-bash: ./bin/spark-shell: No existe el archivo o el directorio
sparkmin@SparkMaster:~$ ./spark-2.2.0-bin-hadoop2.6/bin/spark-shell
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
18/04/24 19:20:22 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... u
sing builtin-java classes where applicable
18/04/24 19:20:22 WARN Utils: Your hostname, SparkMaster resolves to a loopback address: 127.0.1.1;
using 192.168.0.30 instead (on interface enp0s3)
18/04/24 19:20:22 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
18/04/24 19:20:31 WARN ObjectStore: Version information not found in metastore. hive.metastore.schem
a.verification is not enabled so recording the schema version 1.2.0
18/04/24 19:20:31 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
18/04/24 19:20:32 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectExcept
ion
Spark context Web UI available at http://192.168.0.30:4040
Spark context available as 'sc' (master = local[*], app id = local-1524590423077).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_171)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Para que obtengamos este mismo resultado pero funcionando en modo cluster, es necesario indicar en el archivo $SparkHome/conf/slaves, las ip de los servidores esclavos. Además, hay que repetir los pasos anteriores de descarga y descompresion de Spark y Hadoop en cada esclavo y este mismo archivo tiene que estar en las 3 máquinas:

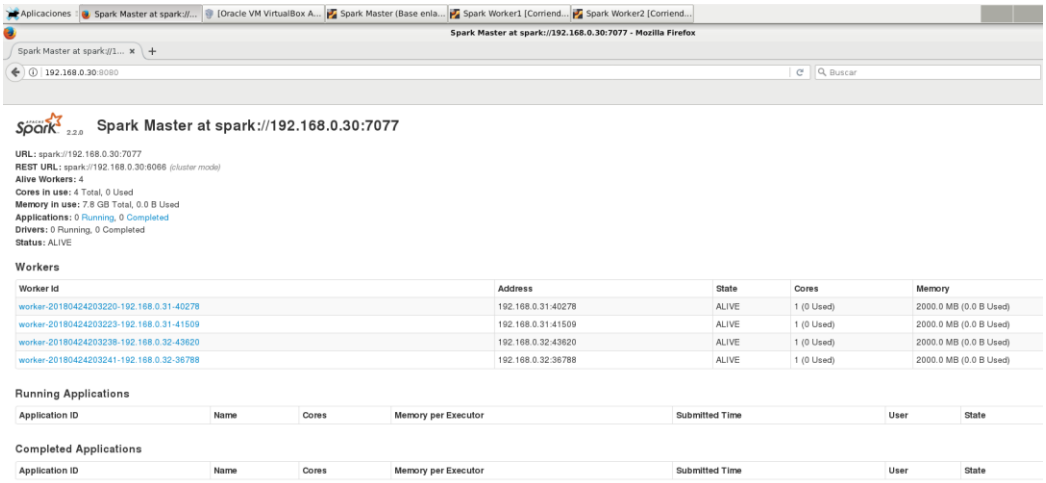
```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# A Spark Worker will be started on each of the machines listed below.
# localhost
192.168.0.31
192.168.0.32
```

También hay que editar un script de inicialización, que está en la misma ruta que el anterior y que se llama spark-env.sh. En el mismo deberemos indicar cual es la ip del nodo master, número de núcleos en cada esclavo, número de instancias de servidor en cada esclavo y cantidad de memoria:

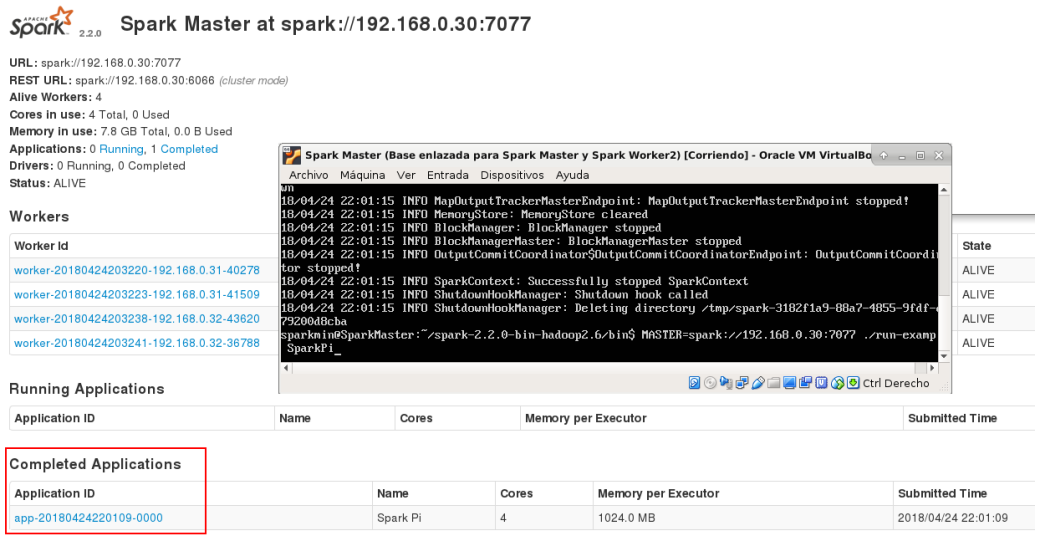
```
export SPARK_MASTER_IP=192.168.0.30
export SPARK_WORKER_CORES=1
export SPARK_WORKER_MEMORY=2000m
export SPARK_WORKER_INSTANCES=2
"./spark-env.sh" 67L, 3829C escritos
sparkmin@SparkMaster:~/spark-2.2.0-bin-hadoop2.6/conf$
```

Ahora tendremos que arrancar el cluster. Se puede iniciar cada máquina por separado, pero desde el servidor master se puede arrancar todo de golpe con el script ./start-all.sh. De igual manera se detiene todo el cluster ejecutando ./stop-all.sh

A continuación podemos apreciar, conectándonos a Spark vía web, que todos los nodos están conectados y a la espera.



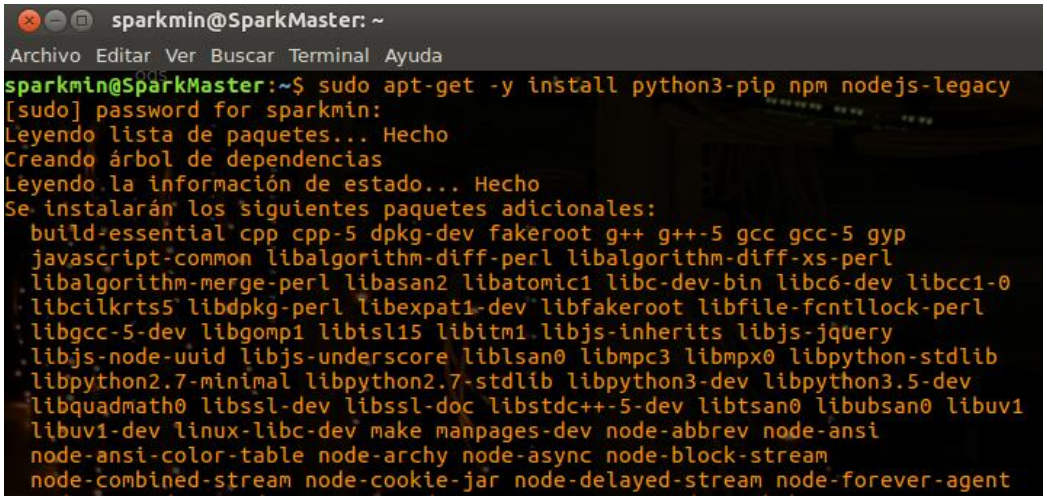
Probamos el correcto funcionamiento del cluster, ejecutando en paralelo un ejemplo de aplicación.



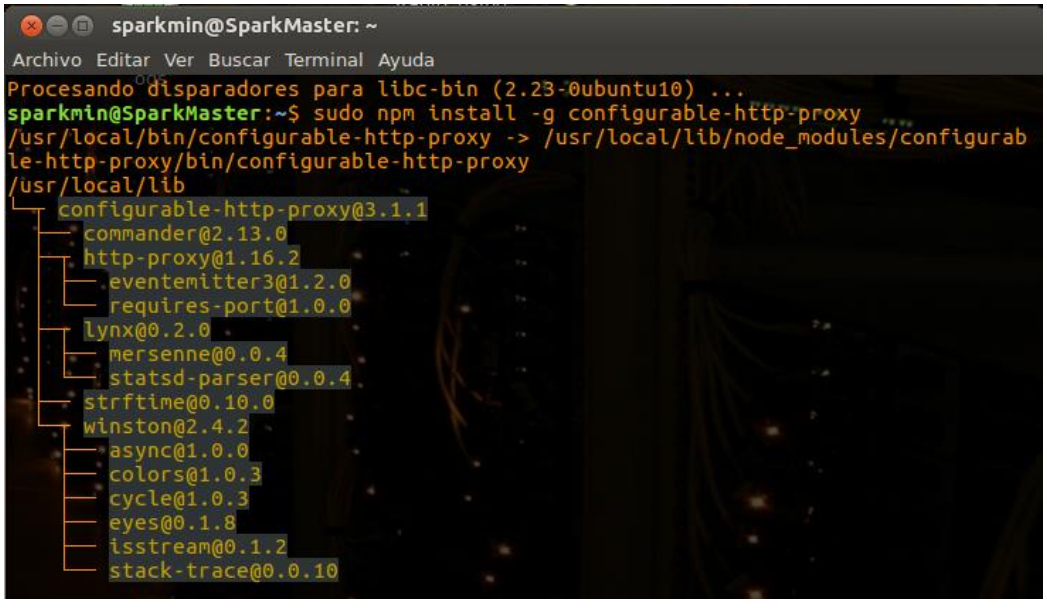
A4. Instalación y configuración de Jupyter Notebooks

A4.1 Instalación y configuración de las dependencias en el nodo Master

En este caso no se va a ejecutar un cluster de máquinas virtuales para ejecutar Jupyter, de momento sólo es necesario instalar una serie de paquetes en la máquina master del cluster Spark. Uno de los primeros de estos paquetes será “pip”, un sistema de gestión de paquetes escritos en lenguaje python, en este caso pip3, que es la versión para gestionar python3, que utilizaremos en nuestro proyecto. Se instalan gran cantidad de paquetes con dependencias:

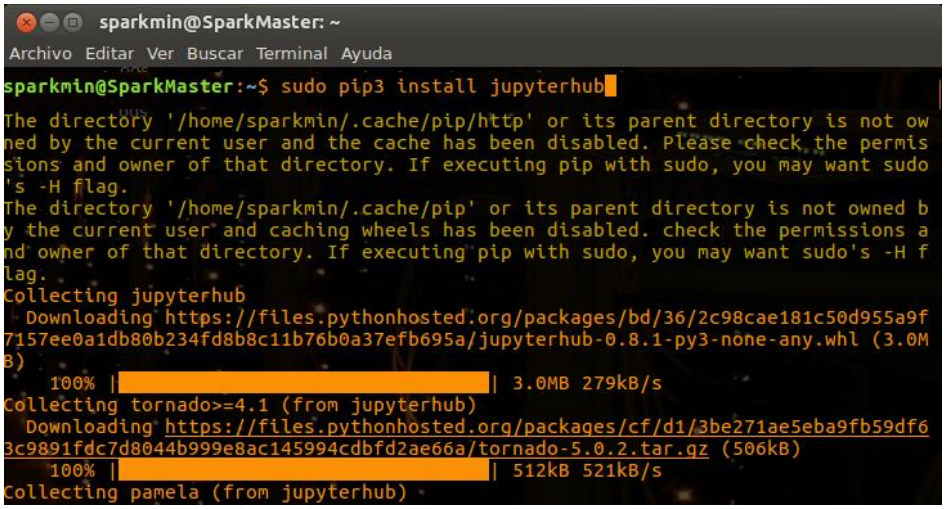


Para acceder a Jupyter también es necesario un servidor hhtp-proxy, por lo que también los instalamos vía apt-get, recordemos que estamos en un sistema Ubuntu:

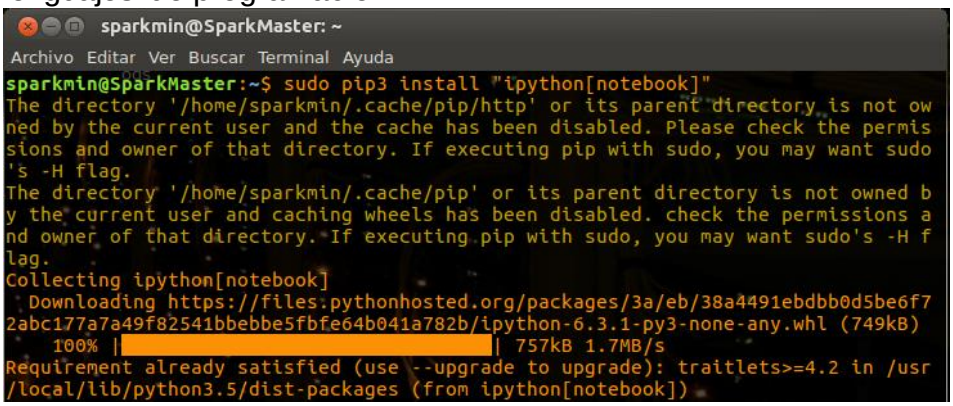


A4.2 Instalación y configuración del entorno multiusuario JupyterHub

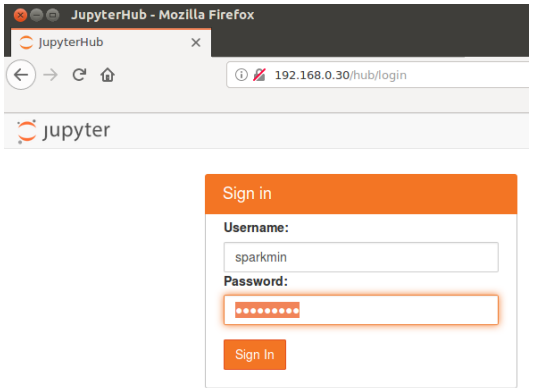
Jupyter es una aplicación web que permite crear y compartir documentos que contienen código fuente, ecuaciones, visualizaciones y texto explicativo. En nuestro caso necesitamos que accedan a la misma gran cantidad de alumnos, por lo que necesitamos que el entorno sea multiusuario. Por este motivo procedemos a instalar JupyterHub:



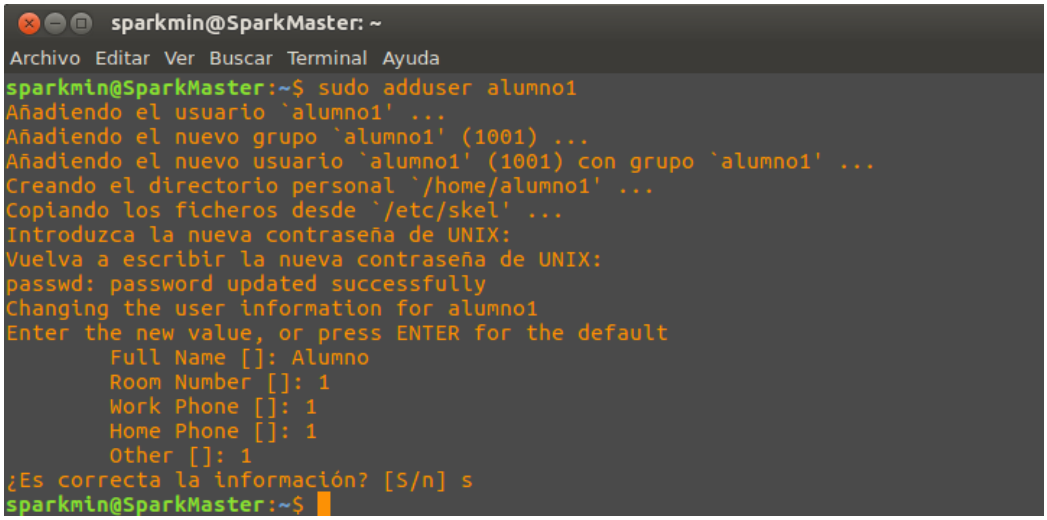
Ahora, ayudándonos del gestor pip3, vamos a instalar el nucleo de Jupyter. Se trata de ipython notebooks, uno de los kernels principales. Más adelante veremos como instalar kernels para Spark y para otros lenguajes de programación:



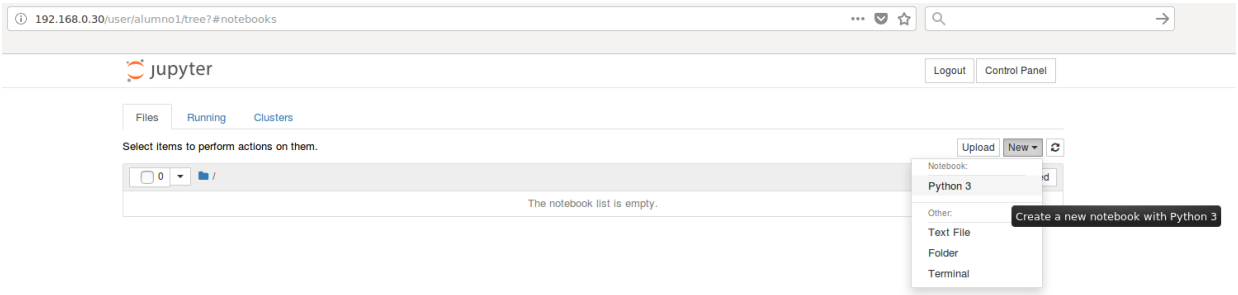
Sólo con estos paquetes ya podemos conectarnos a Jupyter con el usuario sparkmin:



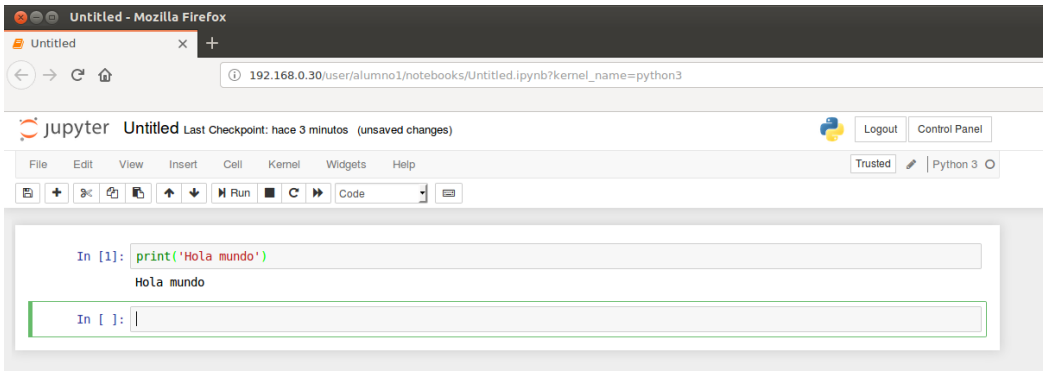
Como vemos, se trata de un usuario del sistema operativo. Tendremos que crear un usuario, para cada alumno al que queramos dar acceso, creando tantos usuarios en el sistema operativo como sea necesario:



Ya podemos acceder a JupyterHub e incluso crear un nuevo Notebook del tipo Python3:



Para probar el funcionamiento creamos un nuevo notebook y ejecutamos un código de prueba:

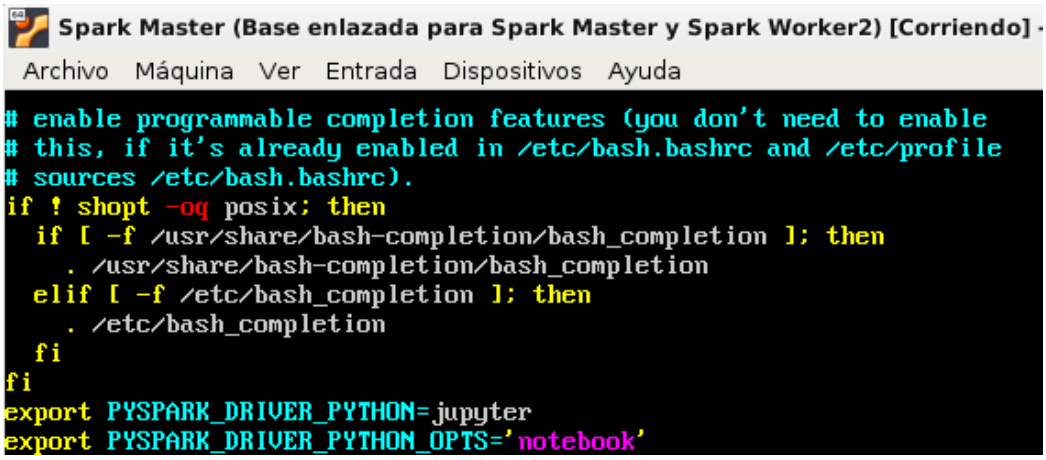


A4.3 Configuración del kernel pyspark y otros lenguajes

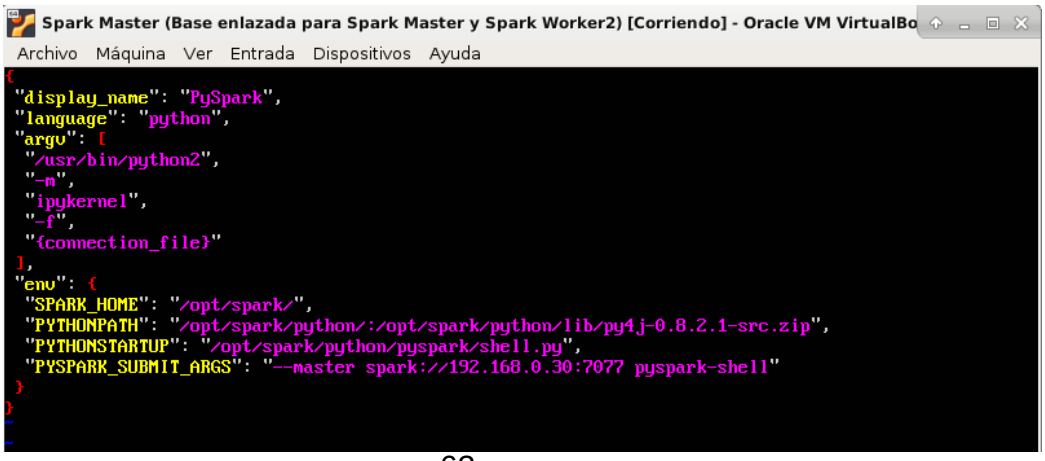
En nuestro proyecto queremos que los alumnos aprendan distintos lenguajes de programación, entre ellos el lenguaje de programación python. Python es un lenguaje muy utilizado en BigData para análisis masivo de datos, machine learning, etc ya que tiene capacidades de procesado en paralelo. Vamos a configurar un kernel llamado pyspark para hacer uso del cluster spark a la hora de ejecutar las aplicaciones escritas en python. El problema es que pyspark es compatible sólo con Python2, y tenemos Python3 en nuestro sistema, por lo que es necesario volver a instalar pip, ipython[notebook], etc. para la versión 2.

```
sudo apt-get -y install python-dev python-setuptools
sudo easy_install pip
sudo pip install py4j
sudo pip install "ipython[notebook]"
```

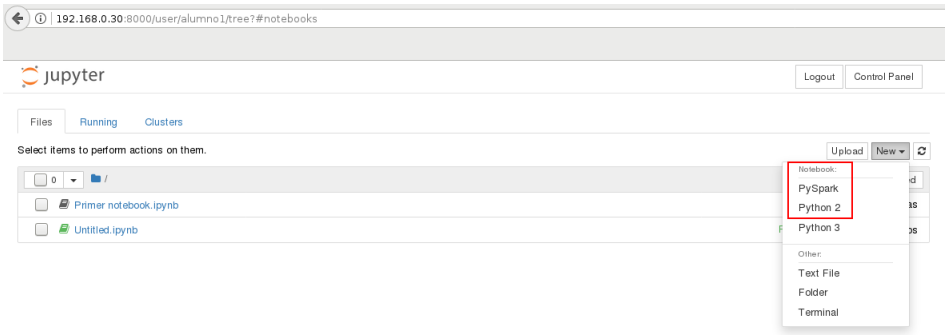
Una vez instalado tendremos que configurar el sistema para que utilice el driver de pyspark para python. Es necesario "tocar" el fichero de inicialización de la consola de comandos para asegurar que en cada reinicio se configure ese driver. Editamos el fichero .bashrc que se encuentra en el directorio $HOME de cada usuario del sistema y añadimos las dos últimas líneas:



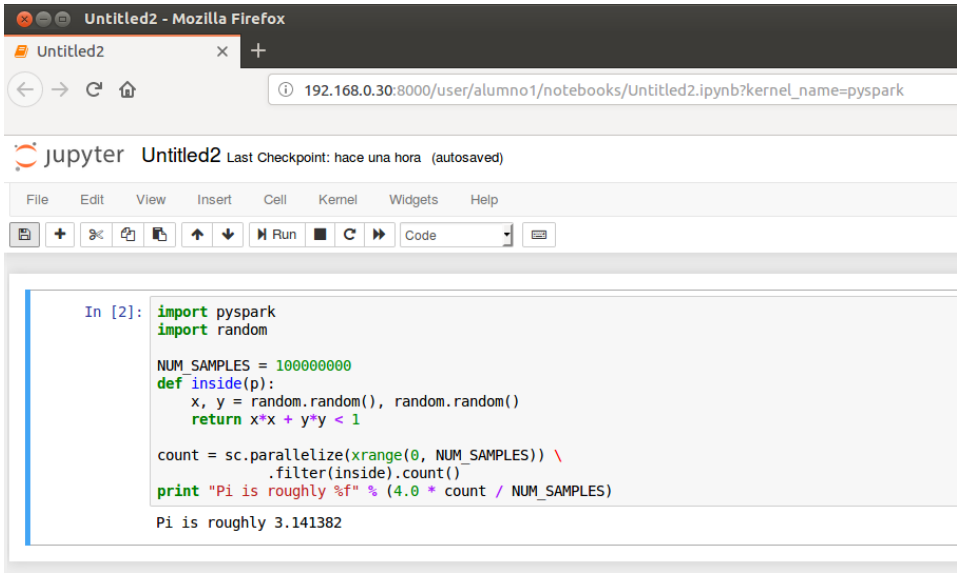
También tenemos que crear un fichero JSON en la ruta /usr/local/share/jupyter/kernels/pyspark/kernel.json para que el kernel esté disponible para todos los usuarios:



Una vez hechos estos cambios tenemos el nuevo kernel disponible en JupyterHub, además del kernel de Python2:



Para probar el funcionamiento creamos un nuevo notebook y ejecutamos un ejemplo del cálculo del número pi en el cluster spark:

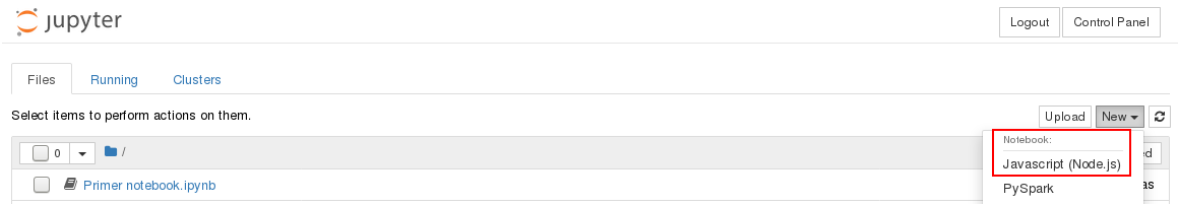


En la web de Git para Jupyter existen numerosos kernels para otros lenguajes de programación:
https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

A continuación se muestra como instalar el kernel por ejemplo para javascript:

```
sudo add-apt-repository ppa:chronitis/jupyter
sudo apt-get update
sudo apt-get install ijavascript
```

Actualizando Jupyterhub ya tendremos disponible el nuevo kernel:

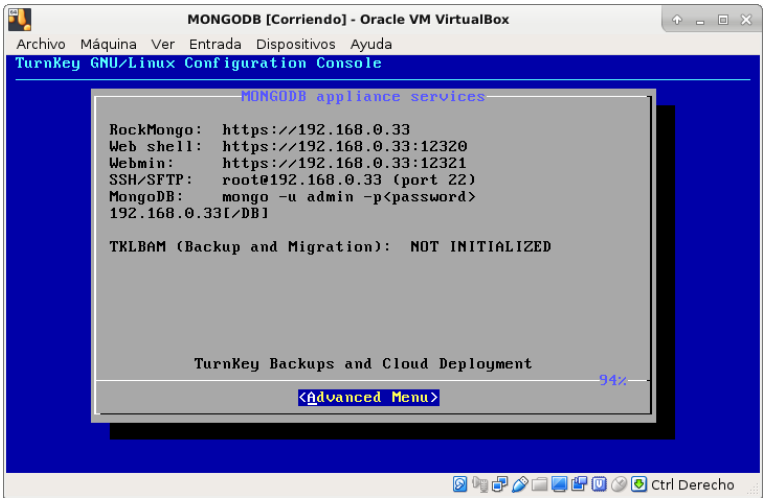


A5. Instalación y configuración de MongoDB

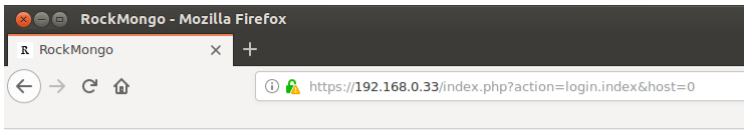
A5.1 Instalación y configuración de Turnkey Linux con imagen MongoDB

Al igual que en el caso de los servidores de Postgresql y Moodle, se ha instalado un servidor Turnkey en una máquina virtual con una imagen con MongoDB preconfigurado.

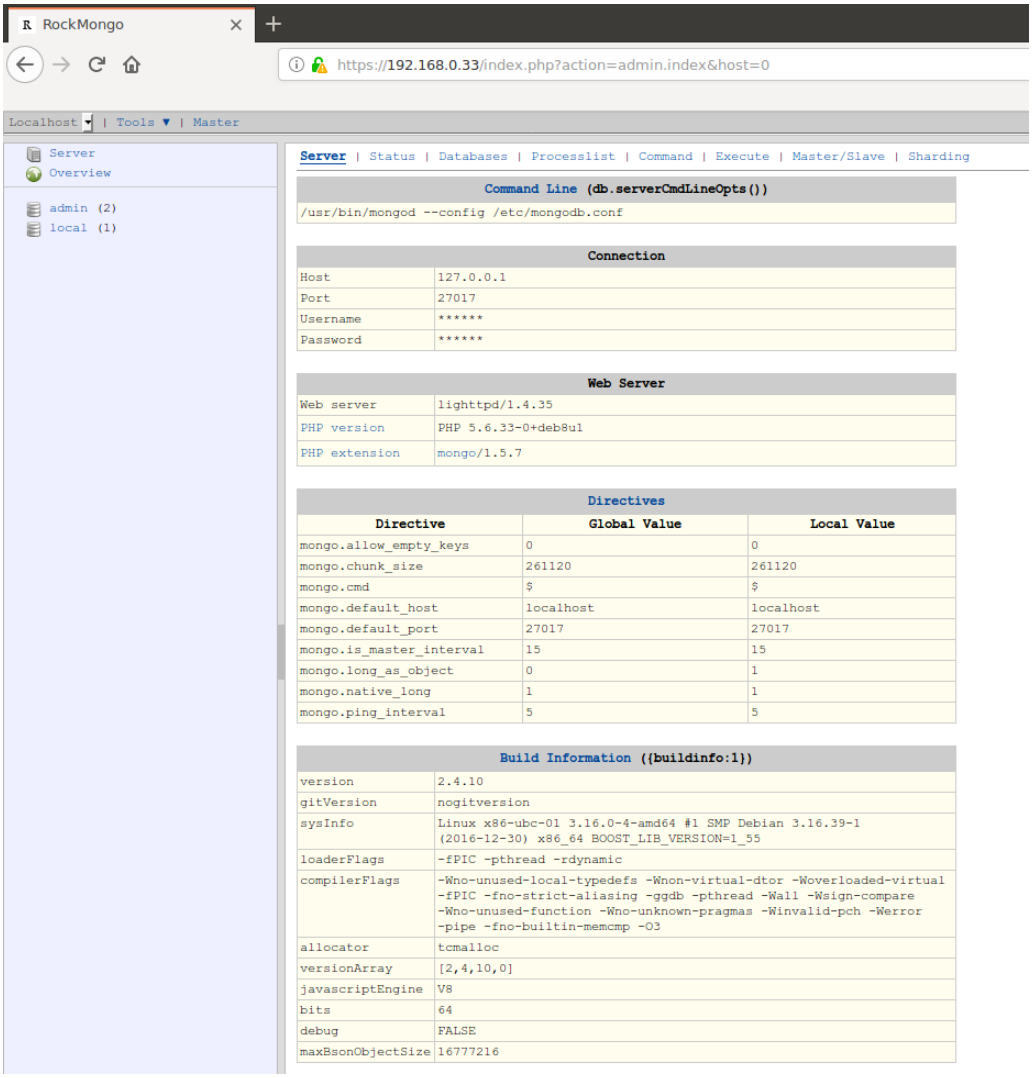
Al importar el servicio a la máquina virtual y arrancar la primera vez, nos pide también en este caso un usuario administrador "root". También tendremos acceso vía webshell, ssh, etc.



De momento no es necesario realizar cambios en la configuración, en el capítulo siguiente se explicara como hacer uso de los notebooks almacenados como documentos en la base de datos MongoDB desde Spark. De momento únicamente comprobamos su funcionamiento iniciando sesión como administradores vía web:

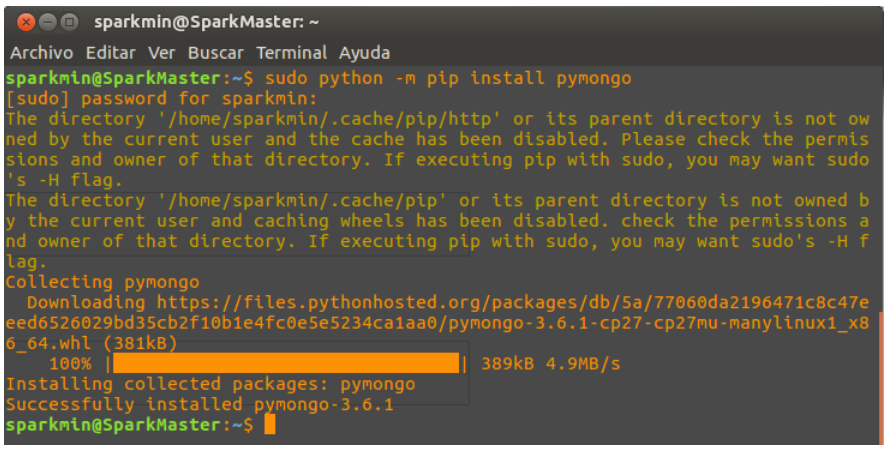


Página principal del servidor utilizando RockMongo:



A5.2 Configuración del driver pyMongo

Para poder utilizar la base de datos MongoDB en conjunción con los notebooks de JupyterHub es necesario utilizar un driver llamado pyMongo. Su instalación es sencilla a través de los comandos pip o easy_install:



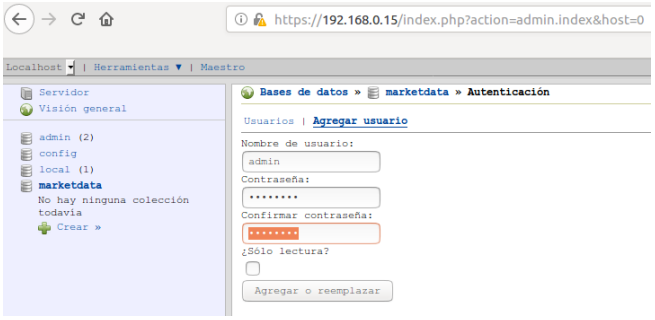
Una vez instalado comprobamos su correcto funcionamiento arrancando el servidor con el comando “mongod -f /etc/mongodb.conf”. A modo de ejemplo se va a descargar un archivo csv con datos de la bolsa. Este archivo contiene aproximadamente 100.000 registros y será utilizado posteriormente para el experimento real:

```
root@mongodb ~# wget www.barchartmarketdata.com%2Fdata-samples%2Fmstf.csv&sa=D&sntz=1&usg=AFQjCNErKsHfV7l6PKvm6igSQIpuVbfabQ
[1] 930
[2] 931
[3] 932
[2]-  Done                    sa=D
[3]+  Done                    sntz=1
```

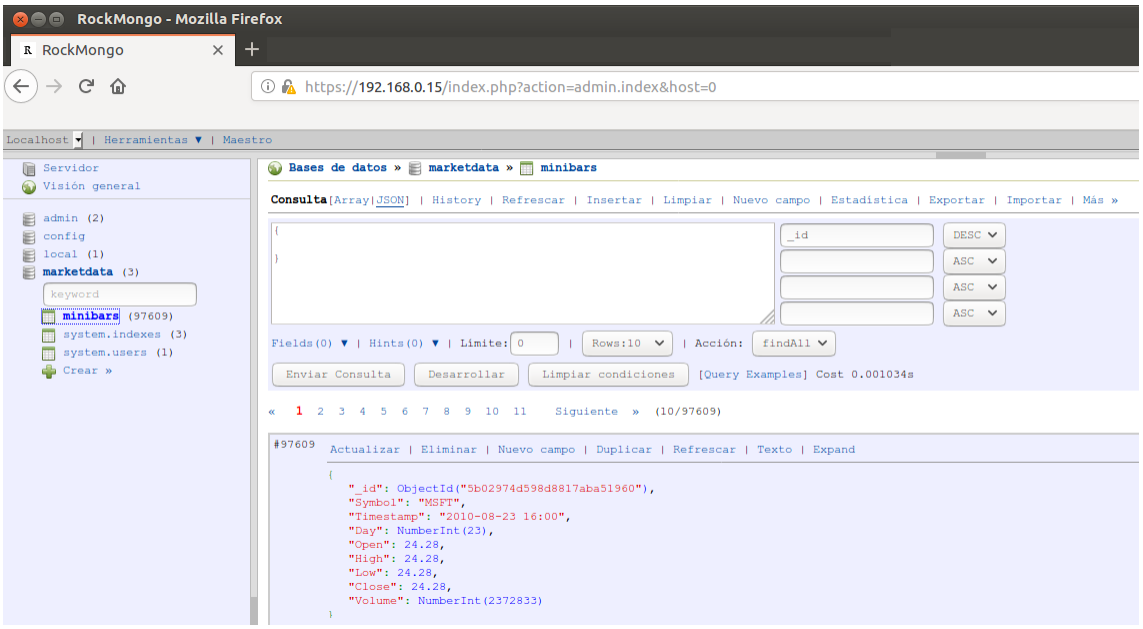
A partir de este archivo vamos a crear una tabla llamada “minibars” y una base de datos llamada “marketdata” mediante el siguiente comando:

```
root@mongodb ~# mongoimport mstf.csv --type csv --headerline -d marketdata -c minibars -u [redacted] -p[redacted]
connected to: 127.0.0.1
Mon May 21 09:54:19.680 check 9 97610
Mon May 21 09:54:21.054 imported 97609 objects
```

Previamente es necesario añadir un usuario a dicha base de datos con permiso de escritura:

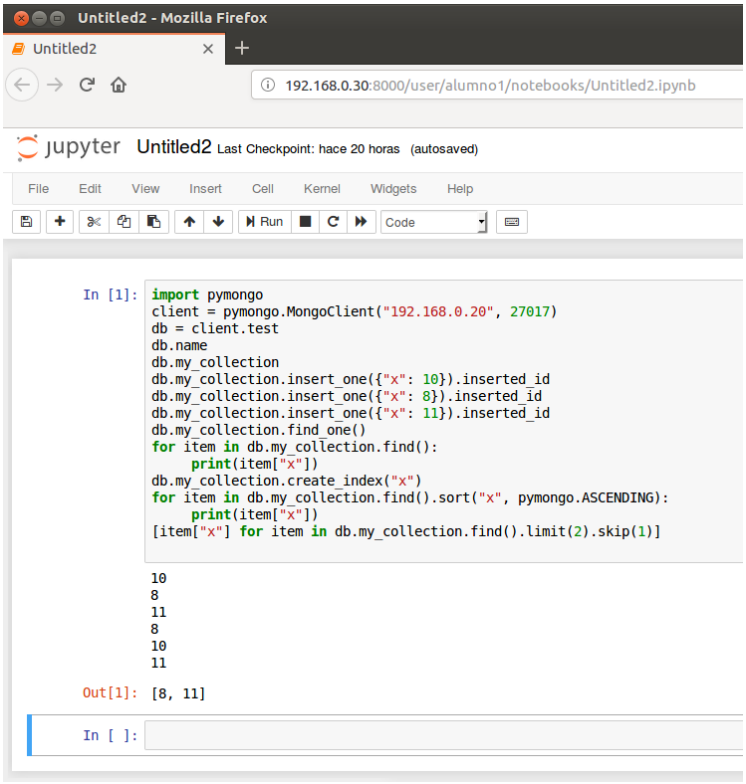


Comprobamos finalmente como se han creado los documentos y se muestra a continuación el formato de un documento en este tipo de base de datos:

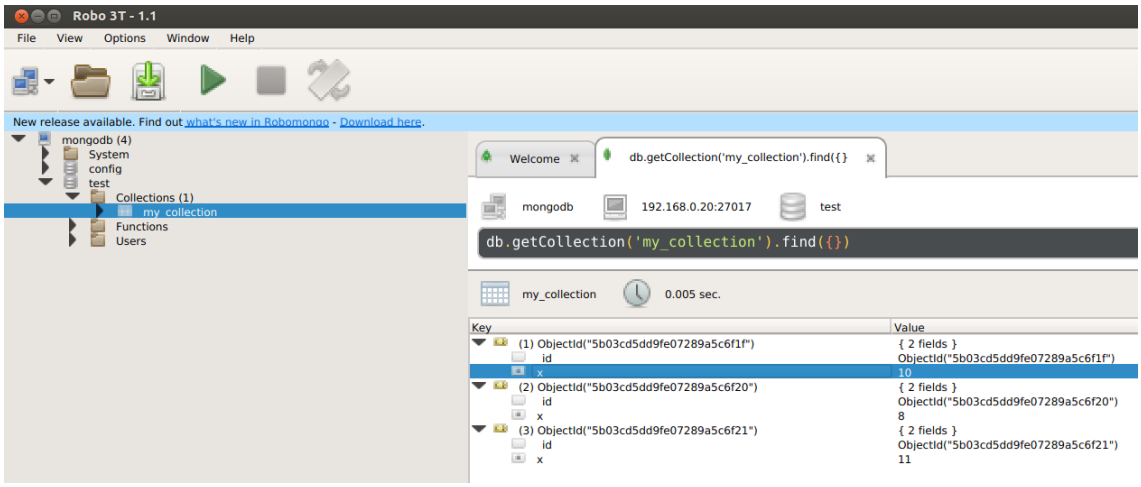


Ya hemos comprobado que la base de datos funciona correctamente, el siguiente paso será comprobar que, a través de un notebook pyspark, utilizando el driver pymongo, podemos crear tablas y registros en una nueva base de datos.

Para ello utilizaremos un sencillo código de prueba que permite ordenar una serie de valores de menor a mayor y además muestra el menor y el mayor valor:



Conectándonos a través del cliente Robo 3T, observamos que se han creado 3 objetos en la tabla “my_collection” de la base de datos “test”:



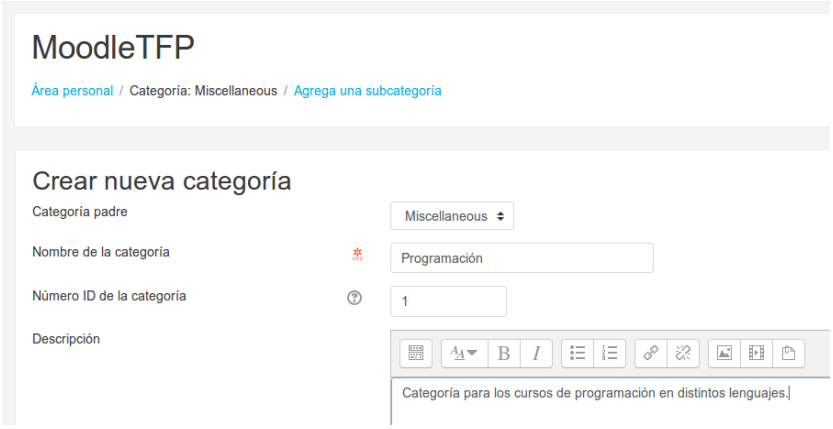
A6. Creación del escenario de análisis

Llegados a este punto, en el que tenemos la plataforma creada en su totalidad, vamos a proceder a crear un escenario simulado que nos permitirá crear las tablas y los datos necesarios para posteriormente hacer los análisis propuestos.

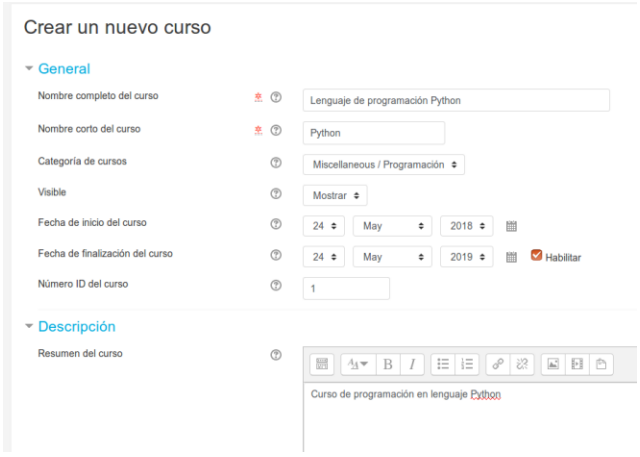
A6.1 Creación de cursos y alumnos en la plataforma Moodle.

Hasta este momento, en cuanto a la plataforma Moodle se refiere, solamente teníamos un usuario de ejemplo y poco más. Si bien es verdad que en la instalación del servidor y la base de datos se crean las tablas con ciertos datos por defecto, ahora tenemos que crear un ejemplo que simule la realidad de un campus. El primer paso será crear un curso, para lo cual tenemos que dirigirnos a la opción "Administración del sitio" -> "Administrar cursos y categorías":

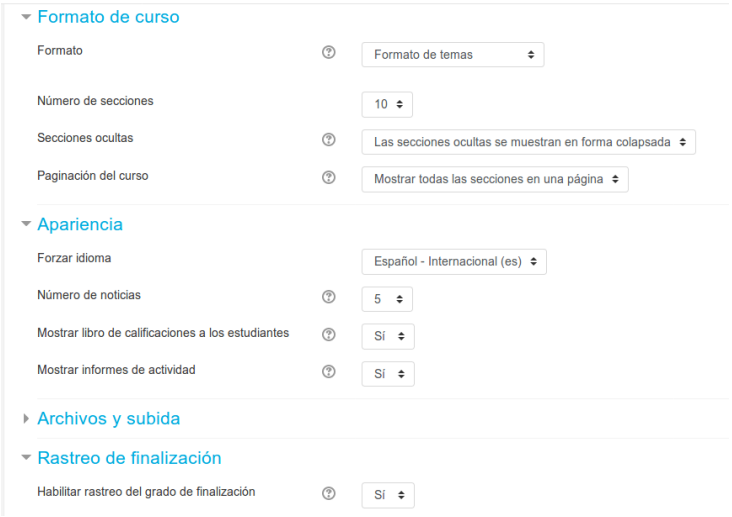
Existe una categoría padre por defecto llamada "Miscellaneous", podríamos crear categorías "padre" que jerarquicamente podrían ser como distintos centros de estudio o campus. Nosotros vamos a crear una categoría para los cursos de programación dentro de la opción por defecto:



El siguiente paso será crear un curso dentro de la nueva categoría. En nuestro ejemplo vamos a crear los cursos de programación en lenguaje Python y Javascript.



Cada curso tiene una serie de opciones que deberemos configurar, entre las que está el formato del curso, pudiendo ser por temas o por semanas:



En cuanto a los cursos no hay mucho más que explicar, sólo quedaría añadir los contenidos en las distintas secciones, como por ejemplo cuestionarios, talleres, etc. Podemos ver a continuación como se han creado las correspondientes tablas de categorías y cursos en la base de datos:

Edit Data - postgres master (192.168.0.21:5432) - moodle - mdl_course_categories

File Edit View Tools Help

No limit

	id [PK] bigserial	name character varying(255)	idnumber character varying(100)	description text
1	1	Miscellaneous		
2	2	Programación	1	<p>Categoría para los cursos de programación en distintos lenguajes.
*				

Edit Data - postgres master (192.168.0.21:5432) - moodle - mdl_course

File Edit View Tools Help

100 rows

	id [PK] bigserial	category bigint	sortorder bigint	fullname character varying(254)	shortname character varying(255)	idnumber character varying(100)
1	1	0	1	MoodleTFP	MoodleTFP	''
2	2	2	20001	Lenguaje de programación Python	Python	1
3	3	2	20002	Lenguaje de programación Javascript	Javascript	2
*						

A continuación vamos a crear los usuarios. Para nuestro ejemplo crearemos 100 usuarios que se podrán generar de una forma semiautomática mediante un fichero csv, indicando la información correspondiente en los distintos campos:

	A	B	C	D	E	F
1	username	password	firstname	lastname	email	course1
2	alumno1	alumno1	alumno	1	alumno1@tfp.es	Python1
3	alumno2	alumno2	alumno	2	alumno2@tfp.es	Python1
4	alumno3	alumno3	alumno	3	alumno3@tfp.es	Python1
5	alumno4	alumno4	alumno	4	alumno4@tfp.es	Python1
6	alumno5	alumno5	alumno	5	alumno5@tfp.es	Python1
7	alumno6	alumno6	alumno	6	alumno6@tfp.es	Python1
8	alumno7	alumno7	alumno	7	alumno7@tfp.es	Python1
9	alumno8	alumno8	alumno	8	alumno8@tfp.es	Python1
10	alumno9	alumno9	alumno	9	alumno9@tfp.es	Python1
11	alumno10	alumno10	alumno	10	alumno10@tfp.es	Python1
12	alumno11	alumno11	alumno	11	alumno11@tfp.es	Python1
13	alumno12	alumno12	alumno	12	alumno12@tfp.es	Python1
14	alumno13	alumno13	alumno	13	alumno13@tfp.es	Python1
15	alumno14	alumno14	alumno	14	alumno14@tfp.es	Python1
16	alumno15	alumno15	alumno	15	alumno15@tfp.es	Python1

Subiremos este archivo en Moodle y los usuarios se crearán con sus distintos campos.

Vemos como estos usuarios se han creado correctamente.

Nuevo usuario	99	100	alumno98	alumno	98	alumno98@tfp.es	alumno98	manual
Nuevo usuario	100	101	alumno99	alumno	99	alumno99@tfp.es	alumno99	manual
Nuevo usuario	101	102	alumno100	alumno	100	alumno100@tfp.es	alumno100	manual

Usuarios creados: 100
Usuarios con contraseña débil: 0
Errores: 0

Y por último nos queda matricularlos en los cursos. A modo de ejemplo matricularemos 25 alumnos en el curso de Python.

Al igual que en el caso de los datos sobre categorías y cursos, comprobamos la información sobre los usuarios directamente en la tabla mdl_usuario de PostgreSQL:

Edit Data - postgres master (192.168.0.21:5432) - moodle - mdl_user

File Edit View Tools Help

100 rows

	id	username character varying(100)	password character varying(255)	idnumber character varying(255)	firstname character varying(100)	lastname character varying(
1		guest	$2y$10$Kb4gVHDKOlh90e9GashX.OKkmj6z6FYCspMTUzOr3HYFkUlVdgcpO	''	Guest user	
2		admin	$2y$10$AuRZeq0tMVq3DqU7Mx5tgu.Q1ZM9tKV1ONubifyF/qxMmI.wW2uBq	''	Admin	User
3		alumno1	$2y$04$cm29wEQQuaUycahFDMENc.OxIuqJsvmng5FtmPSwbZlWM4gapj5/u	''	alumno	1
4		alumno2	$2y$04$/PPqs6wg99fQ4tUs3khj3OGddOewOZIWe4/80df1vn26om39okbKy	''	alumno	2
5		alumno3	$2y$04$krL3lduM.NwR79vpDPnuseZbdOtQFyezPDdMV05AESeSTPjQ7Ec3a	''	alumno	3
6		alumno4	$2y$04$gUmK3RNs5qcNkLU59d2da.M2w5RMoN6Q15IMbZjY4OfR7VuTYovwS	''	alumno	4
7		alumno5	$2y$04$LOieSU1yGSA1S7b1aUgRKe6yp95fZXBzhYt5NvwsXDeZ9L2MYTXZC	''	alumno	5
8		alumno6	$2y$04$SLgy/RAkd3w1tr1k1GbPNegzOefCkW8pTTmD/vHl9VrESBrmDN5Ba	''	alumno	6
9		alumno7	$2y$04$RFmHfUdjbavsNm4yrLvqiOTNLU8ByN3aXoJkb3MWi2sNmoCKxghfa	''	alumno	7
10		alumno8	$2y$04$F/KmyUDFx3QNplu7tB2roeboyJpMhZb9dU7ObRTuDv8RG1sEN5RK.	''	alumno	8

A6.2 Creación de usuarios en JupyterHub, integración de Moodle con Spark y generación de datos de la actividad de los usuarios en MongoDB.

Para la correcta integración de los dos tipos de bases de datos, postgresql y mongodb, necesitamos que en ambas los usuarios tengan la misma denominación. En el caso de JupyterHub, que hará uso de mongoDB, los usuarios son los mismos que los del sistema, por lo tanto será necesario crear 100 usuarios en la máquina que aloja el servidor.

Afortunadamente también existe un método para crear usuarios de forma automatizada en el sistema operativo linux a partir de un archivo csv:

*usuarios.csv (~/Escritorio) - gedit

Archivo Editar Ver Buscar Herramientas Documentos Ayuda

Abrir

```
alumno2:alumno2:1002:1002:Alumno 2:/home/alumno2:/bin/bash
alumno3:alumno3:1003:1003:Alumno 3:/home/alumno3:/bin/bash
alumno4:alumno4:1004:1004:Alumno 4:/home/alumno4:/bin/bash
alumno5:alumno5:1005:1005:Alumno 5:/home/alumno5:/bin/bash
alumno6:alumno6:1006:1006:Alumno 6:/home/alumno6:/bin/bash
alumno7:alumno7:1007:1007:Alumno 7:/home/alumno7:/bin/bash
```

Copiamos el archivo al servidor, nos conectamos y ejecutamos el comando:

sparkmin@SparkMaster: ~

Archivo Editar Ver Buscar Terminal Ayuda

```
bluekat@BigByte:~$ scp usuarios.txt sparkmin@192.168.0.30:/home/sparkmin
sparkmin@192.168.0.30's password:
usuarios.txt                                  100% 6209     6.1KB/s   00:00
bluekat@BigByte:~$ ssh sparkmin@192.168.0.30
sparkmin@192.168.0.30's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-124-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Pueden actualizarse 17 paquetes.
1 actualización es de seguridad.

sparkmin@SparkMaster:~$ sudo newusers usuarios.txt
[sudo] password for sparkmin:
```

Listamos los usuarios del sistema y efectivamente encontramos los alumnos:

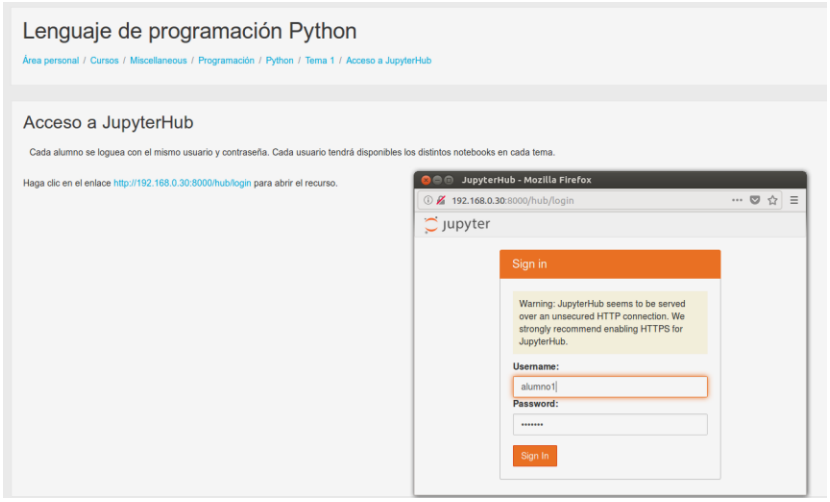
sparkmin@SparkMaster: ~

Archivo Editar Ver Buscar Terminal Ayuda

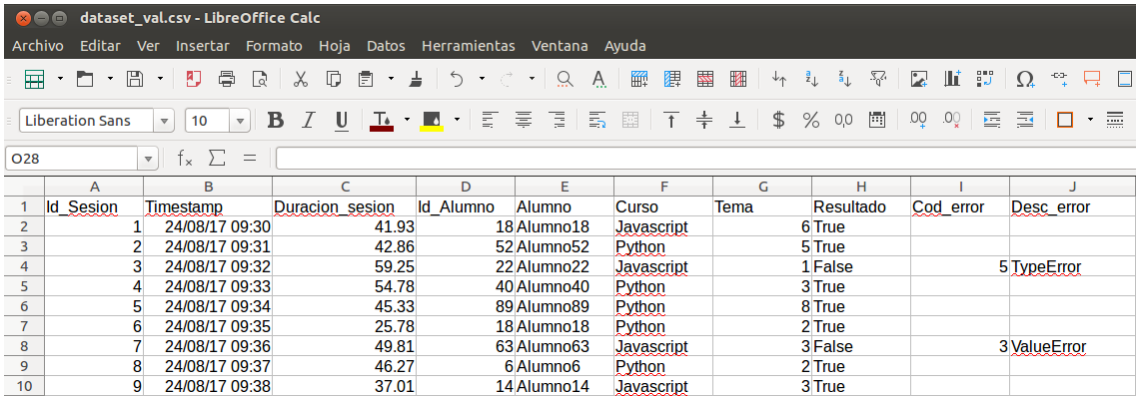
```
sparkmin@SparkMaster:~$ cat /etc/passwd | cut -d":" -f1 |grep alumno*
alumno1
alumno2
alumno3
alumno4
alumno5
alumno6
alumno7
```

Una vez creados los usuarios en ambos sistemas vamos a proceder a la integración de los mismos. En el curso de Python que hemos tomado como ejemplo, en el tema 1 creamos un módulo de navegador para conectar a la plataforma de JupyterHub. Tendremos que indicar la url del servidor, así como el puerto, y después cada usuario se logueará con su contraseña:

Vemos un ejemplo de conexión por parte de un alumno:



La creación de un notebook con código real que genere los campos propuestos a analizar queda fuera de los objetivos de este proyecto, ya que no habría tiempo material para diseñarlo así como para generar los datos con usuarios reales. Por lo tanto, a partir del archivo que se utilizó para comprobar el funcionamiento de mongoDB, se ha modificado y, gracias a funciones de aleatoriedad, se ha introducido información hasta tener un volumen cercano a los 100.000 documentos.



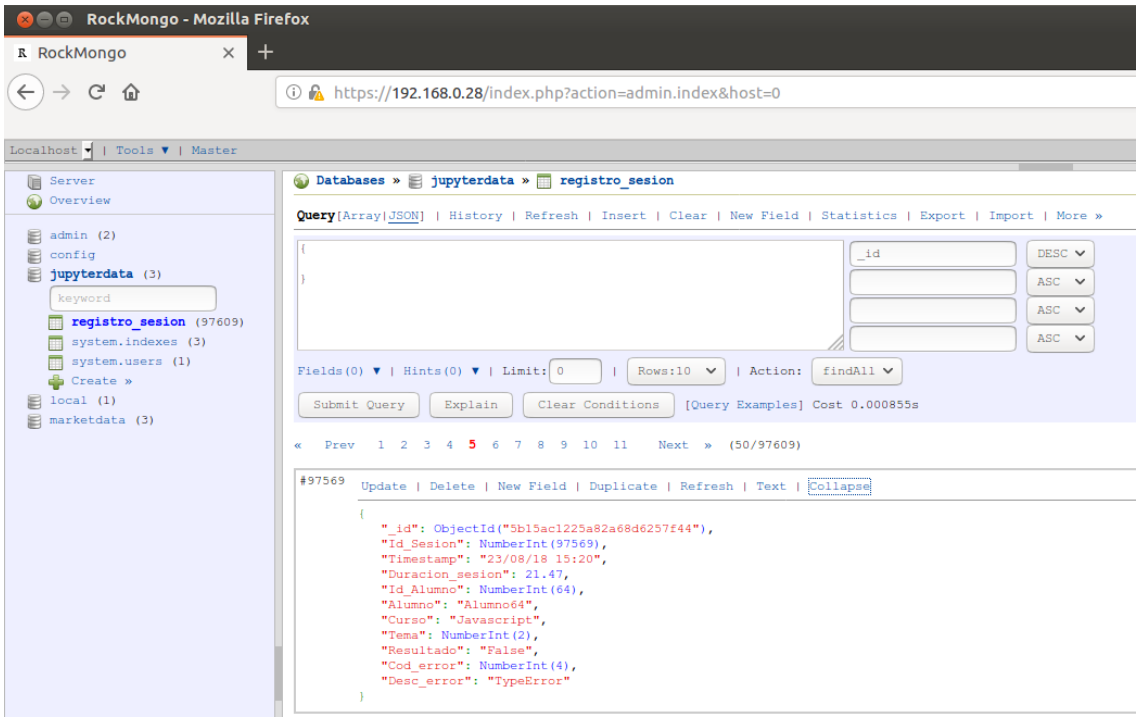
	A	B	C	D	E	F	G	H	I	J
1	Id_Sesion	Timestamp	Duracion_sesion	Id_Alumno	Alumno	Curso	Tema	Resultado	Cod_error	Desc_error
2	1	24/08/17 09:30	41.93	18	Alumno18	Javascript	6	True		
3	2	24/08/17 09:31	42.86	52	Alumno52	Python	5	True		
4	3	24/08/17 09:32	59.25	22	Alumno22	Javascript	1	False	5	TypeError
5	4	24/08/17 09:33	54.78	40	Alumno40	Python	3	True		
6	5	24/08/17 09:34	45.33	89	Alumno89	Python	8	True		
7	6	24/08/17 09:35	25.78	18	Alumno18	Python	2	True		
8	7	24/08/17 09:36	49.81	63	Alumno63	Javascript	3	False	3	ValueError
9	8	24/08/17 09:37	46.27	6	Alumno6	Python	2	True		
10	9	24/08/17 09:38	37.01	14	Alumno14	Javascript	3	True		

Una vez creado el archivo, que contiene los nombre de los distintos campos en las cabeceras, procedemos a importarlo a la base de datos:

```
root@mongodb ~# mongoimport dataset_val.csv --type csv --headerline -d jupyterdata -c registro_sesion -u [illegible] -p [illegible]
connected to: 127.0.0.1
Mon Jun  4 21:16:00.987 check 9 97610
Mon Jun  4 21:16:02.381 imported 97609 objects
root@mongodb ~#
```

En el comando anterior podemos distinguir que se ha creado la base de datos con nombre jupyterdata y la tabla registro_sesion.

Por último, conectándonos con el cliente web Rockmongo, podemos observar un ejemplo de documento en formato JSON. En concreto se han creado 97609 documentos de este tipo:



A7. Instalación de la plataforma de Business Intelligence.

A7.1 Instalación del servidor Pentaho

A lo largo de este capítulo se explicará como instalar una plataforma de BI que nos permitirá por un lado crear un almacén de datos, y por otro lado, podremos hacer un experimento de análisis masivo de datos.

En este primer punto vamos a proceder a instalar Pentaho Server Community Edition 8.0. Se trata de la versión gratuita del servidor, pero que permite todas las funcionalidades, eso sí, no obtendremos soporte por parte de la empresa Hitachi, pero la comunidad es muy extensa y la documentación es abundante.

Como en casi todas las herramientas, comenzaremos instalando un par de paquetes necesarios para el correcto funcionamiento de la aplicación. En este caso son unzip y java 8:

```
pentadmin@pentaho:~$ sudo apt install unzip
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Preparando para desempaquetar .../unzip_6.0-20ubuntu1_amd64.deb ...
Desempaquetando unzip (6.0-20ubuntu1) ...
Procesando disparadores para mime-support (3.59ubuntu1) ...
Procesando disparadores para man-db (2.7.5-1) ...
Configurando unzip (6.0-20ubuntu1) ...
pentadmin@pentaho:~$ sudo apt install default-jre
```

Acto seguido descargamos la imagen precompilada del servidor comprimida en un zip desde la web de SorceForge. El archivo, lógicamente hay que descomprimirlo y copiarlo a la ruta que nos guste.

```
pentadmin@pentaho: ~
Archivo Editar Ver Buscar Terminal Ayuda
pentadmin@pentaho:~$ wget https://sourceforge.net/projects/pentaho/files/Pentaho
%208.0/server/pentaho-server-ce-8.0.0.0-28.zip/download -O pentaho-server-ce-8.0
.0.0-28.zip
--2018-05-24 13:43:08--  https://sourceforge.net/projects/pentaho/files/Pentaho%
208.0/server/pentaho-server-ce-8.0.0.0-28.zip/download
Resolviendo sourceforge.net (sourceforge.net)... 216.105.38.13
Conectando con sourceforge.net (sourceforge.net)[216.105.38.13]:443... conectado
.
```

Es importante crear el grupo y el usuario pentaho, y hacerlo propietario de la carpeta donde hemos descomprimido los archivos:

```
pentadmin@pentaho:~$ sudo addgroup pentaho
Añadiendo el grupo `pentaho' (GID 1001) ...
Hecho.
pentadmin@pentaho:~$ sudo adduser --system --ingroup pentaho --disabled-login pentaho
Añadiendo el usuario del sistema `pentaho' (UID 111) ...
Añadiendo un nuevo usuario `pentaho' (UID 111) con grupo `pentaho' ...
Creando el directorio personal `/home/pentaho' ...
pentadmin@pentaho:~$ sudo chown -R pentaho:pentaho /opt/pentaho
```

Pentaho puede funcionar con distintas bases de datos. En este proyecto se están utilizando PostgreSQL y mongoDB. Esta última no parece apropiada para un almacén de datos, por lo que vamos a configurar Pentaho para que funcione con PostgreSQL:

```
pentadmin@pentaho: /opt/pentaho/pentaho-server/data/postgresql
Archivo Editar Ver Buscar Terminal Ayuda
Configurando postgresql-contrib-9.5 (9.5.12-0ubuntu0.16.04) ...
Configurando sysstat (11.2.0-1ubuntu0.2) ...

Creating config file /etc/default/sysstat with new version
update-alternatives: utilizando /usr/bin/sar.sysstat para proveer /usr/bin/sar (sar) en modo automático
Procesando disparadores para libc-bin (2.23-0ubuntu10) ...
Procesando disparadores para systemd (229-4ubuntu21.2) ...
Procesando disparadores para ureadahead (0.100.0-19) ...
pentadmin@pentaho:~$ sudo -u postgres psql postgres
psql (9.5.12)
Type "help" for help.

postgres=# \password
Enter new password:
Enter it again:
postgres=# \q
pentadmin@pentaho:~$ sudo vim /etc/postgresql/9.5/main/
environment      pg_hba.conf      postgresql.conf
pg_ctl.conf      pg_ident.conf    start.conf
pentadmin@pentaho:~$ sudo vim /etc/postgresql/9.5/main/pg_hba.conf
pentadmin@pentaho:~$ sudo service postgresql restart
```

Una vez instalado el servidor de BBDD que albergará nuestro almacén de datos, es necesario crear ciertas bases de datos locales para que Pentaho funcione correctamente:

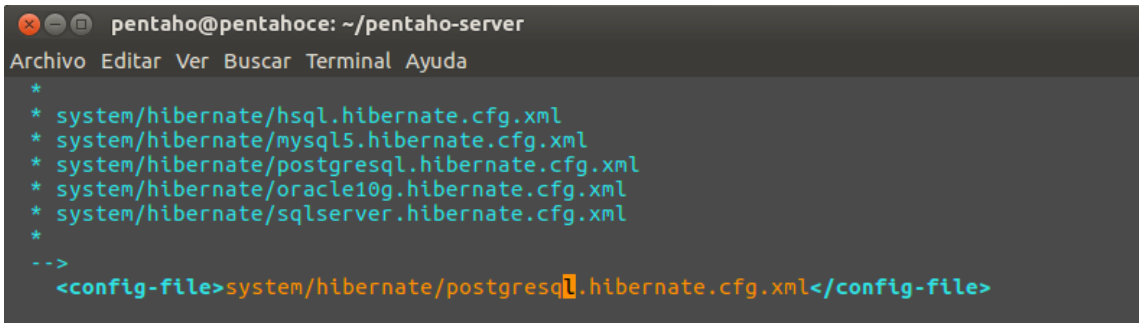
```
pentadmin@pentaho:/opt/pentaho/pentaho-server/data/postgresql$ psql -U postgres -h 127.0.0.1 -p 5432 -f create_quartz_postgresql.sql
Password for user postgres:
DROP DATABASE
DROP ROLE
CREATE ROLE
CREATE DATABASE
GRANT
Password for user pentaho_user:
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
You are now connected to database "quartz" as user "pentaho_user".
BEGIN
```

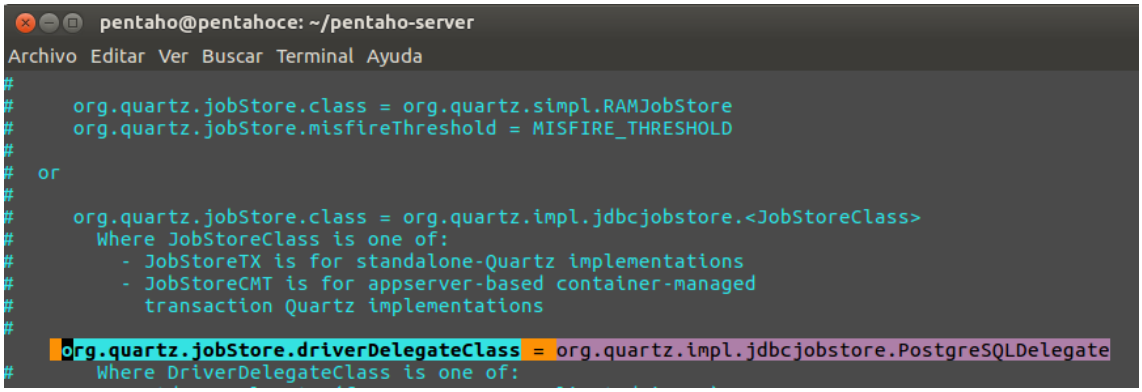
```
pentadmin@pentaho: /opt/pentaho/pentaho-server/data/postgresql
Archivo Editar Ver Buscar Terminal Ayuda
pentadmin@pentaho:/opt/pentaho/pentaho-server/data/postgresql$ psql -U postgres -h 127.0.0.1 -p 5432 -f create_repository_postgresql.sql
Password for user postgres:
psql:create_repository_postgresql.sql:7: NOTICE:  database "hibernate" does not exist, skipping
DROP DATABASE
psql:create_repository_postgresql.sql:8: NOTICE:  role "hibuser" does not exist, skipping
DROP ROLE
CREATE ROLE
CREATE DATABASE
GRANT
pentadmin@pentaho:/opt/pentaho/pentaho-server/data/postgresql$ psql -U postgres -h 127.0.0.1 -p 5432 -f create_jcr_postgresql.sql
```

Es necesario instalar también un driver JDBC, ya que Pentaho se ejecuta sobre un servidor Tomcat y es necesario configurar correctamente el conector de java hacia la base de datos:

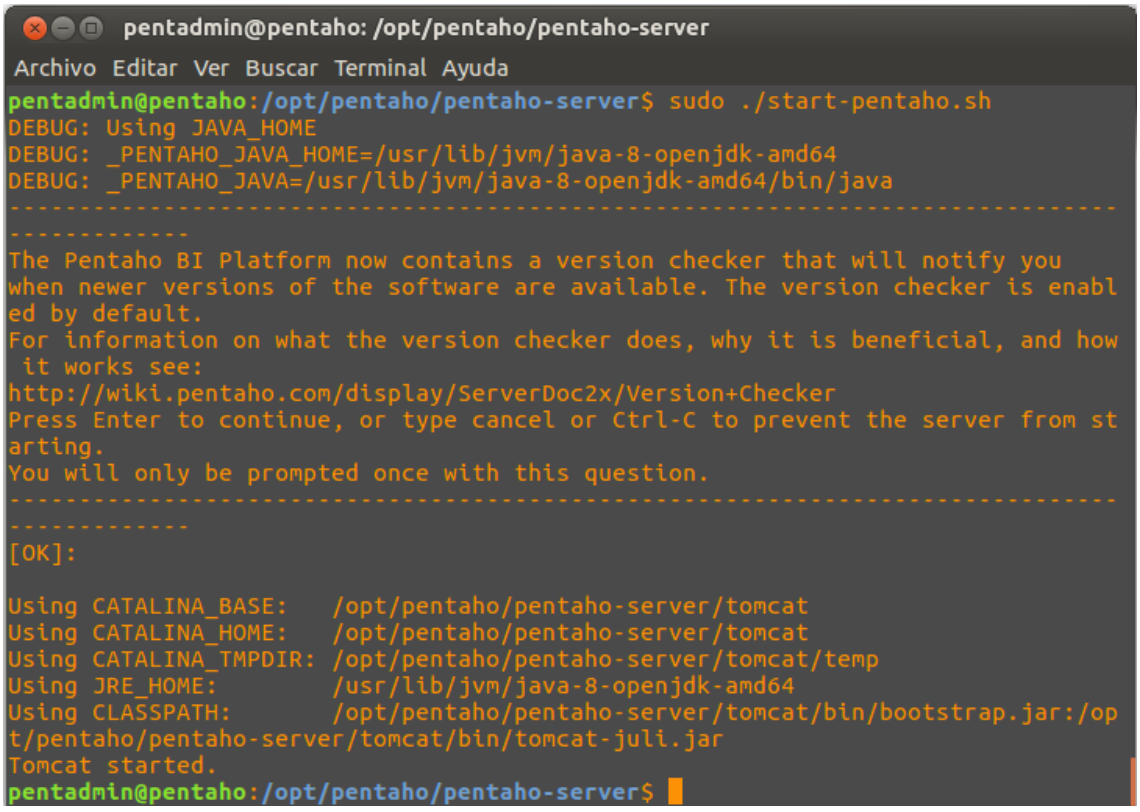
```
pentadmin@pentaho: /opt/pentaho/pentaho-server/tomcat/webapps/pentaho/META-INF
Archivo Editar Ver Buscar Terminal Ayuda
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/pentaho" docbase="webapps/pentaho/">
        <Resource name="jdbc/Hibernate" auth="Container" type="javax.sql.DataSource"
                factory="org.apache.tomcat.jdbc.pool.DataSourceFactory" maxActive="20" minIdle="0" maxIdle="5" initialSize="0"
                maxWait="10000" username="hibuser" password="password"
                driverClassName="org.postgresql.Driver" url="jdbc:postgresql://127.0.0.1:5432/hibernate"
                validationQuery="select count(*) from INFORMATION_SCHEMA.SYSTEM_SEQUENCES" />

        <Resource name="jdbc/Quartz" auth="Container" type="javax.sql.DataSource"
                factory="org.apache.tomcat.jdbc.pool.DataSourceFactory" maxActive="20" minIdle="0" maxIdle="5" initialSize="0"
                maxWait="10000" username="pentaho_user" password="password"
                driverClassName="org.postgresql.Driver" url="jdbc:postgresql://127.0.0.1:5432/quartz"
                validationQuery="select count(*) from INFORMATION_SCHEMA.SYSTEM_SEQUENCES"/>

</Context>
```

También es necesario indicar que en el arranque se utilicen las bases de datos que hemos nombrado anteriormente, hibernate y quartz:

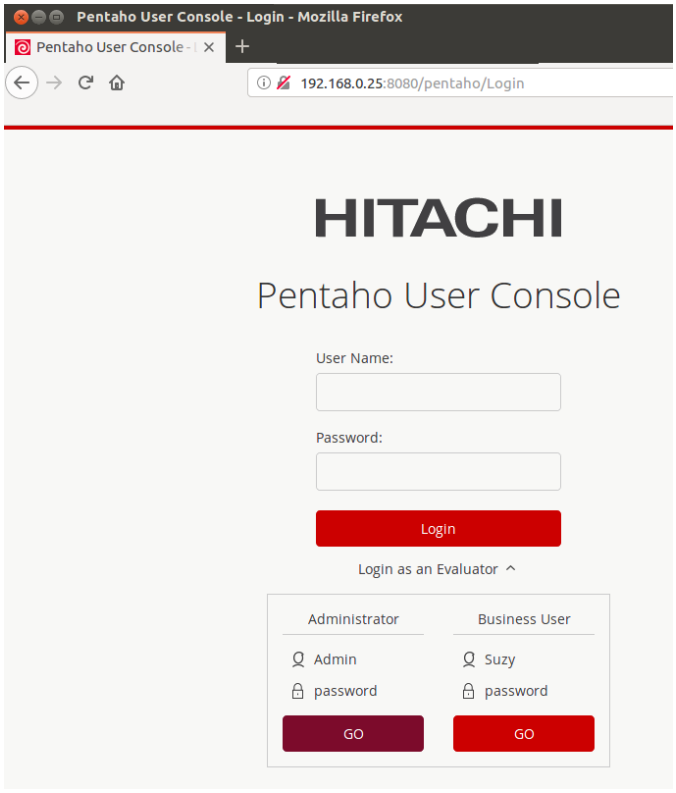




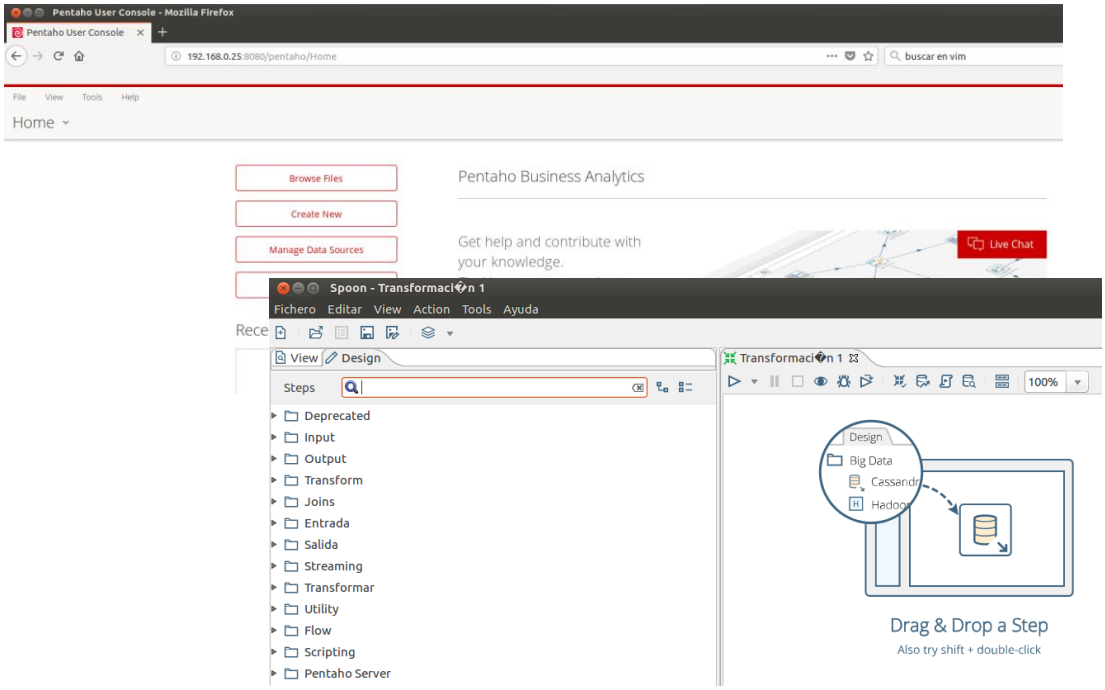
Finalmente podemos arrancar Pentaho mediante el siguiente comando:



Ahora ya podremos conectarnos a través de un navegador con dos perfiles diferentes, como administrador, o como un usuario que por defecto está creado con el nombre Suzy:



Las opciones de la consola de administrador son las siguientes:



Las opciones para el usuario Suzy se reducen a la creación de análisis, para lo cual son necesarias las herramientas del siguiente apartado.

A7.2 Instalación de las herramientas cliente Mondrian y Spoon.

La instalación de las herramientas de análisis es muy sencilla. Como en el caso del servidor, en cualquier ordenador "cliente", por ejemplo en el portátil de trabajo, descargamos y descomprimimos los programas Pentaho data-analysis, más conocido como Spoon, o Kettle, que nos permitirá crear el almacén de datos. Por otro lado, descargaremos Pentaho Schema-Workbench, más conocido como Mondrian, y que nos permitirá hacer cubos de análisis multidimensional o cubos OLAP.

En ambos casos son archivos ejecutables, spoon.sh y workbench.sh. Se muestra a continuación sus pantallas de inicio respectivamente: