F. Terroso-Sáenz and A. Hernández, "A Tool for Automatic Generation of Data Mining Practice Environments" in IEEE-RITA, vol. 16, no. 2, pp. 171-179, MAY. 2021.
doi: 10.1109/RITA.2021.3089922

Una Herramienta para la Creación Automática de Entornos de Prácticas en Análisis de Datos

F. Terroso-Sáenz and A. Hernández

***Title*— A Tool for Automatic Generation of Data Mining Practice Environments.**

***Abstract*— Data analysis subjects are an important part of the computer science degree. In these subjects, seminars and practical sessions where students can practice different data mining algorithms and techniques are a paramount part. This usually involves the installation and deployment of a common development environment for all the students including libraries, databases, datasets and so forth. Due to the heterogeneity of these environments and the characteristics of the undergraduates, such a configuration and deployment task might be an important step in the seminars schedule. Consequently, the present work puts forward an open-source tool for the automatic deployment of data-mining environments based on Docker containers. By means of this tool, the teacher can specify the parameters of the environment to deploy. This results in a set of virtual images comprising the whole practice environment that can be easily distributed among students. This way, we meaningfully optimize the aforementioned deployment step and we also guarantee that all students work with the same practice environment. Finally, we have evaluated the performance of the tool and the level of satisfaction of the students with the generated environments showing quite promising results.**

***Index Terms*— Virtualization, Containers, Machine Learning, Environment.**

I. Introducción

En los planes de estudios del grado en Ingeniería Informática, las asignaturas relacionadas con el análisis y tratamiento de datos están cobrando cada vez más relevancia debido al creciente interés en la sociedad por paradigmas como el Big Data. Dichas asignaturas están incorporando poco a poco el lenguaje de programación *Python* como parte de su programa de enseñanza práctica [1].

Generalmente, el objetivo de dichas prácticas es que los alumnos se familiaricen con determinadas librerías de análisis de datos que implementan algunas de las técnicas y algoritmos vistos en la parte teórica de la materia [2]. Para ello se usan *frameworks* muy activos en la comunidad Python de *machine learning* como scikit-learn[1], scipy[2] o Tensorflow[3].

Para poder hacer viables dichas prácticas es imprescindible que todos los alumnos trabajen con la misma versión de las librerías y frameworks. Sin embargo, en los últimos años la tendencia *bring your own device* (BYOD) se ha extendido entre los alumnos y ya son mayoría los que prefieren traer sus propios portátiles y realizar los seminarios de prácticas con ellos [3]. Esto puede provocar que los alumnos no tengan instaladas en sus ordenadores las mismas versiones de las librerías que el docente, lo cual puede dificultar la dinámica de los seminarios. Por ejemplo, la versión más reciente de *Scikit-learn* (v 0.23), uno de los *frameworks* más importantes en el ámbito del *machine learning*, requiere para su instalación Python 3.x. Sin embargo, muchas distribuciones de Linux y Mac todavía usan Python 2.x como lenguaje por defecto [4].

Para resolver dicho problema de heterogeneidad de versiones en el aula, el presente trabajo propone la herramienta de código abierto *Docker Machine Learning Environment Creator* (DMLEC). Dicha herramienta permite distribuir entre los alumnos máquinas virtuales equipadas con todos los recursos necesarios para realizar las prácticas incluyendo un Entorno de Desarrollo Integrado (IDE) de trabajo, bases de datos, ficheros de usuario o librerías a utilizar.

Sin embargo, para evitar los problemas de complejidad y gasto de recursos que la virtualización *completa* implica [5] optamos por una *virtualización por contenedores* basados en Docker[4] que representan una opción más ligera y flexible que las máquinas virtuales tradicionales [6].

El uso de Docker ha permitido además desarrollar dos formas de distribución diferentes del entorno de prácticas entre los alumnos, cada uno con un nivel diferente de automatismo adaptándose así al nivel de competencias de cada grupo de alumnos.

Finalmente, el código fuente de dicha aplicación se encuentra disponible para su descarga y uso por parte de la comunidad educativa mediante licencia GPL 3.0 en la plataforma GitHub[5].

El resto del presente artículo se estructura como sigue. La sección II describe los trabajos relacionados con la solución propuesta, mientras que la sección III describe la herramienta

[1] https://scikit-learn.org/stable
[2] https://www.scipy.org
[3] https://www.tensorflow.org
[4] https://www.docker.com
[5] https://github.com/xxx

DMLEC en detalle. Una evaluación del rendimiento de la herramienta se realiza en la sección IV y del grado de satisfacción del alumnado en la sección V. Finalmente, la sección VI detalla las principales conclusiones y vías de trabajo futuras que se derivan de este artículo.

II. Trabajos Relacionados

En la Tabla 1 podemos observar un resumen comparando las funcionalidades de diferentes herramientas que hacen uso de tecnologías de virtualización en el entorno docente aplicados a diferentes ámbitos.

Tabla 1. Comparativa de herramientas de virtualización en el entorno docente

Herramienta	Dominio de Aplicación	Características		
		Tipo Virtualización	Interfaz Gráfica	Repositorio Remoto
Anaconda	Análisis de Datos	Contenedores / Docker	Sí	Sí
Repo2Docker [7]	Programación	Contenedores / Docker	No	Sí
[8]	Programación	Contenedores / Docker	Sí	No
[9]	Programación	Contenedores / Docker	Sí	Sí
[10]	Programación / Big Data	Contenedores / Docker	Sí	Sí
[11]	Programación / Evaluación online	Contenedores / Docker	-	
[12][13]	Redes de ordenadores	Completa	Sí	No
[14]	Programación	Completa	Sí	No
Blinder	Análisis de Datos	Completa / Cloud	No	Sí
[15]	Análisis de Datos	Aplicación / JVM	Sí	No
DMLEC	**Análisis de Datos**	**Contenedores / Docker**	**Sí**	**Sí**

Uno de los *frameworks* más usado para la realización de tareas de análisis de datos en los lenguajes Python y R es Anaconda[6]. Dicho *framework* permite la descarga de gran número de librerías con diferentes funcionalidades de análisis de datos, así como otras herramientas para el pre-procesamiento y visualización de datos. Además, las últimas versiones cuentan con una interfaz gráfica con diferentes funcionalidades. Por último, dicho framework permite también, a través del repositorio *continuumio*[7], la creación de imágenes Docker conteniendo diferentes distribuciones y configuraciones del framework. Sin embargo, la interfaz gráfica nativa de Anaconda no incluye ninguna funcionalidad que permita generar las imágenes Docker personalizadas basándose en dicho repositorio. En este caso, el usuario debe editar manualmente de forma programática la generación de las imágenes deseadas.

Repo2Docker es una herramienta genérica que permite acceder a un repositorio Git indicado por el usuario y crear una imagen Docker para dicho repositorio [7]. Sin embargo, dicha herramienta sólo proporciona una interfaz basada en línea de comandos sin incluir ninguna interfaz gráfica que permita facilitar la tarea al docente a la hora de generar los entornos a distribuir.

La virtualización por contenedores basados en Docker ha sido utilizada con cierta notoriedad a la hora de generar entornos de prácticas para programación. Así, en [8] los autores proponen un sistema virtual basado en Docker para la realización de cursos de programación en C, C++ y Java. Otro trabajo interesante es el propuesto en [9] en donde se describe un entorno basado en Docker para la implantación de grandes laboratorios virtuales de programación en la educación escolar. En este sentido, la herramienta se basa en imágenes Docker pre-configuradas en las que no existe posibilidad de configuración. Un enfoque similar se ha propuesto también en [10], pero esta vez para la programación en plataformas de Big Data como Apache Hadoop. En este sentido, los autores proponen un entorno basado en tecnología Docker para generar un entorno de prácticas online enfocado en el *e-learning*. Por otro lado, el trabajo descrito en [11] sigue la misma línea de aplicar la tecnología Docker en laboratorios de programación, pero centrándose en la generación de jueces online que permita a los alumnos evaluar sus ejercicios de prácticas de forma automática. En este sentido, el uso de la virtualización por contenedores permite aligerar los costes y mantenimiento de la infraestructura.

En [12] y su posterior revisión en [13] encontramos una estrecha relación con nuestro proyecto, sobre todo en los aspectos de virtualización, donde se implementa un sistema virtualizado de redes orientado a la docencia, obteniendo las claras ventajas que supone la virtualización, como el no deterioro de *routers* y *switches* por mal uso, ni causar desperfectos en los aparatos físicos, permitiendo así que las asignaturas de prácticas se desarrollen como si de un sistema real se tratara. Sin embargo, dicha solución se basa en una virtualización completa al emplear tecnologías VMware[8]. Dicho tipo de virtualización se basa en que las máquinas virtuales simulan todos los elementos del hardware físico (ej: CPU, memoria, disco, tarjetas de red, etc.). Por tanto, dichas máquinas suelen virtualizar una máquina física de forma completa.

En [14], los autores proponen el uso en clase de la herramienta VMD (*Virtual Machine on Demand*). El funcionamiento de este herramienta consiste en que cada estudiante lleve en un medio de almacenamiento extraíble (USB, DVD, etc.) al aula sus máquinas virtuales (MVs), y al acceder a los ordenadores de clase despliegue dicha herramienta y ejecute el contenido del medio extraíble. VMD ofrece una interfaz intuitiva para desplegar MVs, pero sin usar contenedores al emplear una virtualización completa.

Por otro lado, la herramienta Binder[9] permite la ejecución en la nube de determinados repositorios en la plataforma GitHub mediante Jupyter Notebooks. Esta solución es interesante a la hora de permitir reproducir determinados experimentos de forma sencilla por parte de los alumnos, sin embargo, al realizar su ejecución en la nube, dificulta a los mismos el almacenar determinados resultados intermedios de

[6] https://www.anaconda.com/distribution/
[7] https://hub.docker.com/u/continuumio
[8] https://www.vmware.com/es.html
[9] https://mybinder.org

sus experimentos. Además, dicha herramienta no incluye ningún soporte ni interfaz gráfica a la hora de generar los repositorios que constituyen la base los notebooks.

Finalmente, WEKA (*Waikato Environment for Knowledge Analysis*) es una herramienta de software libre orientada al aprendizaje automático y a la minería de datos diseñada en Java [15]. Sin embargo, dicha herramienta no ofrece virtualización ni permite la replicación y distribución de determinadas configuraciones de entornos para el análisis de datos entre múltiples usuarios a través de repositorios remotos.

A modo de conclusión, podemos ver que el uso de entornos virtualizados ha resultado ser una solución satisfactoria en numerosos ámbitos docentes. En este sentido, es cierto que ya existen algunas soluciones que permiten la generación de entornos basados en Docker para realizar tareas de análisis de datos de forma online. Sin embargo, DMLEC enriquece dicho ecosistema como una solución enfocada a facilitar la generación y distribución entre los estudiantes de entornos de prácticas virtualizados mediante una sencilla interfaz gráfica para el docente.

III. DOCKER MACHINE LEARNING ENVIRONMENT CREATOR (DMLEC).

La presente sección se centra en describir la solución propuesta. Puesto que la misma se basa en gran medida en la tecnología Docker en primer lugar se describen los conceptos básicos de dicha tecnología, para a continuación explicar en detalle la herramienta DMLEC.

A. *Conceptos Relevantes de Docker*

La tecnología Docker se basa en cuatro conceptos diferentes pero interrelacionados como son Dockerfile, imagen Docker, contenedor Docker y Docker Hub. En este sentido la Fig. 1 muestra la relación entre estos cuatro conceptos.

La tecnología Docker se basa en un motor (Docker engine) que tiene que estar instalado en cualquier máquina en la que se quiera ejecutar dicha tecnología. En este sentido, dicho motor es mucho más ligero que un hipervisor utilizado en la virtualización completa.

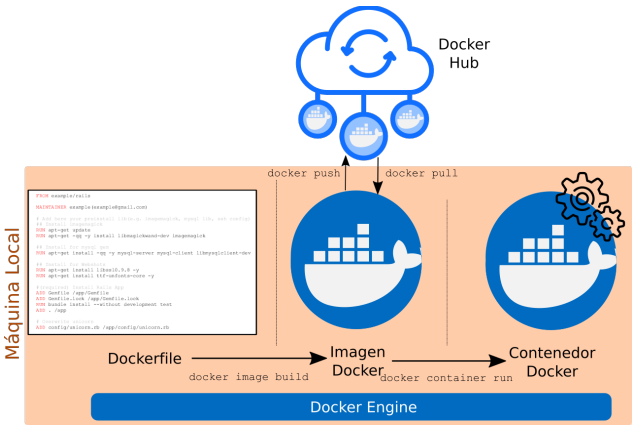

Figura 1. Conceptos básicos de Docker. Las flechas indican las relaciones entre dichos conceptos.

Una imagen Docker es una plantilla de solo lectura que contiene las instrucciones necesarias que deben ejecutarse en el Docker engine para generar un nuevo contenedor. Normalmente una imagen Docker se basa en otra imagen con alguna *personalización* adicional. Para generar una nueva imagen, es necesario crear un fichero de configuración textual llamado *Dockerfile* con una sintaxis simple. Dicho fichero contiene las instrucciones que definen los pasos necesarios para crear la imagen y ejecutarla. Más en concreto, cada instrucción en un Dockerfile crea una capa en la imagen. De esta forma, cuando se cambia el Dockerfile y se reconstruye la imagen, solo se vuelven a crear las capas que han cambiado. Esto es uno de los principales motivos que hace que las imágenes sean tan livianas, pequeñas y rápidas, en comparación con otras tecnologías de virtualización.

Un contenedor Docker es una instancia en ejecución de una imagen Docker. De esta forma, un contenedor alberga sólo los elementos mínimos e imprescindibles para que las aplicaciones o servicios que contiene funcionen sin problemas y pueda ejecutarse en cualquier maquina que tenga un motor Docker instalado. Esto permite que un contenedor Docker generalmente consuma muchos menos recursos de computación que una MV tradicional [16].

Docker Hub es el repositorio público de Docker. Los desarrolladores pueden subir sus imágenes y Dockerfiles en dicho repositorio y así compartirlos con el resto de la comunidad.

Por último, es interesante remarcar una herramienta incluida en Docker llamada *Docker Compose*. Mediante dicha herramienta es posible coordinar la ejecución de varios contenedores para que puedan operar de forma conjunta. Mediante sencillos ficheros en formato *yml* un desarrollador puede definir un entorno de ejecución que abarque a diferentes contenedores cada uno proporcionando un determinado servicio, así como otros elementos relativos al almacenamiento y a la conectividad.

B. *Entorno de prácticas a generar*

DMLEC genera un entorno de prácticas compuesto por dos imágenes Docker diferentes tal y como muestra en la Fig. 2.

Una primera imagen alberga el conjunto de datos sobre el que los alumnos van a realizar las prácticas de análisis junto con la base de datos que alberga dichos datos.

Una segunda imagen alberga las diferentes herramientas que los alumnos van a utilizar para analizar los datos contenidos en el primer contenedor. En concreto, este contenedor incluye las diferentes librerías de análisis de datos en python, un código fuente base que usa dichas librerías y sobre los cuales los alumnos deberán trabajar, así como un IDE para poder editar el código fuente proporcionado.

Mediante esta estructura separamos la parte de almacenamiento de los datos y de tratamiento de dichos datos en dos imágenes diferentes.

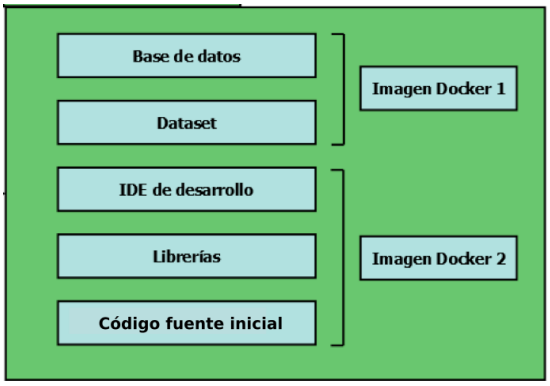

Figura 2. Escenario de prácticas a generar incluyendo dos imágenes diferentes.

C. *Arquitectura general de DMLEC*

Esta sección se centra en describir de forma general los principales componentes internos de la herramienta DMLEC. Dichos componentes tienen como objetivo generar el entorno de prácticas compuesto por las dos imágenes Docker cuya estructura se ha descrito en la sección anterior.

Como podemos observar en la Fig. 3, DMLEC sigue una arquitectura cliente-servidor. Por un lado, la parte cliente incluye una interfaz web que permite al docente interaccionar con la herramienta para detallar los parámetros de configuración de los entornos de prácticas a generar. Por otro lado, la parte servidor, es la encargada de generar los entornos de prácticas basados en Docker. Por ello, es necesario que esta parte back-end se ejecute en una máquina física que cuente con un Docker engine instalado de forma nativa.

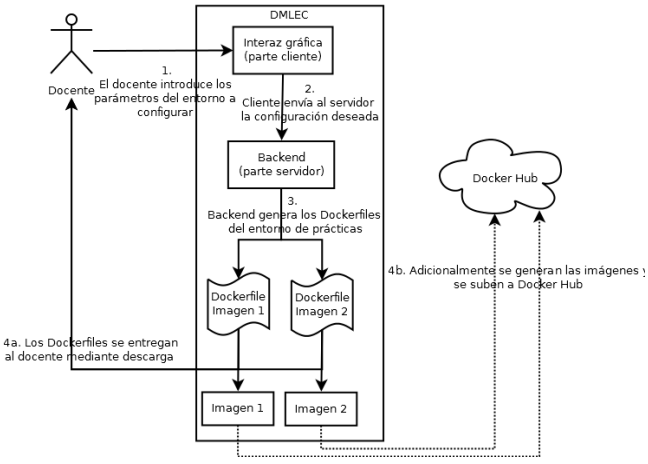

Figura 3. Diagrama de flujo de DMLEC

De esta forma, cuando el docente define todos los parámetros de los entornos a generar (flecha 1 de la Fig. 3), la parte cliente los remite al servidor (fecha 2) para que esta componga los Dockerfiles (flecha 3) que permitirán dar lugar a las correspondientes imágenes descritas en la sección anterior (flecha 4a). Adicionalmente, el sistema puede también componer automáticamente dichas imágenes y subirlas a Docker Hub (flecha 4b) tal y como se describe en la sección III.F.2.

D. *Detalles de implementación*

DMLEC ha sido desarrollada mediante diferentes tecnologías de programación. En particular, para el desarrollo de la interfaz web se ha usado el framework Bootstrap[10] para su diseño y realización además del lenguaje de programación Javascript. Para el módulo servidor, se ha usado Java Servlet como herramienta software para su implementación. Además, para la ejecución de los diferentes comandos Docker necesarios para componer los escenarios, la parte servidor hace uso de una librería Docker-Java[11] que permite accede a la API de Docker de manera programática.

Internamente, la parte servidor está construida como una serie de reglas SI-ENTONCES anidadas que, en función de los parámetros remitidos por el cliente, van generando de manera secuencial las diferentes instrucciones a incluir en los dos ficheros Dockerfile. Así, por ejemplo, una regla para instalar la librería *scikit-learn* en su versión 0.22.1 en la Imagen Docker 2, toma la siguiente forma:

```
  if(lib=='scikit-learn){
     dockerfile2Writer.writeln("RUN apt-get update; \
      apt-get install -y \
        python python-pip \
        python-numpy python-scipy \
        build-essential python-dev python-setuptools \
        libatlas-dev libatlas3gf-base")

     dockerfile2Writer.writeln("RUN pip install -U scikit-
learn==0.22.1")
  }
```

Como vemos dicha regla añade las instrucciones RUN necesarias para instalar dicho framework dentro del fichero Dockerfile destinado a crear la segunda imagen del entorno de prácticas.

Por último, la presente versión de la herramienta está parametrizada para instalar en las imágenes una versión en particular de las librerías y herramientas para evitar problemas de compatibilidad entre dichos componentes.

E. *Interfaz gráfica de DMLEC (módulo cliente)*

A la hora de generar las dos imágenes descritas en la sección III.B, DMLEC hace uso de una interfaz web compuesta de dos vistas diferentes tal y como se muestra en las Figs. 4 y 5. Dicha interfaz actúa como módulo cliente de la herramienta (ver Fig. 3) y se ejecuta sobre un navegador web estándar.

En primer lugar, la vista inicial de la herramienta (Fig. 4) está destinada a que el docente especifique determinados parámetros de su instalación local de Docker necesarias para que DMLEC pueda generar las imagines Docker a distribuir tales como su credenciales y determinadas rutas de configuración de la plataforma. Dichos parámetros son necesarios por parte de la parte servidor de la herramienta a fin de que este pueda ejecutar correctamente los comandos Docker necesarios para componer los Dockerfiles e imágenes asociadas del entorno de prácticas.

Una vez el docente ha especificado dichos parámetros, se transiciona a una segunda vista en donde se especifican el contenido concreto de las dos imágenes a distribuir entre los alumnos (ver Fig. 5). En particular, esta segunda vista permite especificar:

La base de datos a utilizar pudiendo elegirse entre MySQL, PostgreSQL y MariaDB.

[10] https://getbootstrap.com/

[11] https://github.com/docker-java/docker-java

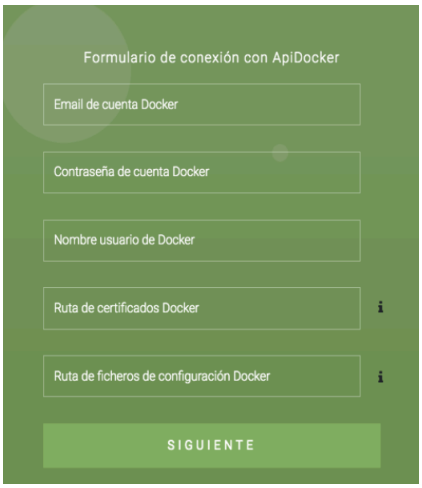

Figura 4. Vista inicial de DMLEC con los parámetros de configuración del entorno Docker local.

Las librerías de análisis de datos a utilizar pudiéndose seleccionar entre Scikit-learn[12], Tensorflow[13], Keras[14] y Scipy[15].

El IDE a usar para programar el código proporcionado pudiendo elegirse entre soluciones bastante complejas como NetBeans[16],IntelliJIdea[17] y Eclipse[18] y otras más ligeras como SublimeText3[19].

Además, la vista permite subir un fichero comprimido incluyendo el código fuente inicial sobre el que trabajarán los alumnos en las prácticas. Dicho fichero se descomprimirá al generar la imagen definitiva.

Figura 5. Segunda vista de DMLEC para la introducción de los parámetros del entorno a generar por parte del docente.

Finalmente, DMLEC permite también especificar el *dataset* a almacenar en la base de datos. En este sentido, debemos tener en cuenta que la estructura y contenido del *dataset* va guiar en gran medida la sesión de prácticas a realizar. Así, por ejemplo, un conjunto de datos tabulares será más apropiado para la realización de prácticas basadas en algoritmos "clásicos" de análisis de datos como Random Forest o Logistic Regression. Sin embargo, los conjuntos de imágenes pueden ser más apropiados para la realización de prácticas asociadas al campo del *Deep Learning*.

Por tanto, es importante que la herramienta ofrezca un amplio abanico de conjuntos de datos a ser seleccionados para garantizar su usabilidad. En este contexto, existe gran cantidad de repositorios en Internet de conjuntos de datos disponibles para su análisis. Uno de los que ofrece una mayor variedad de conjuntos es *Kaggle*[20]. Dicha plataforma ofrece una API basada en Python que permite listar y descargar sus datasets disponibles.

Nuestra herramienta hace uso de dicha API para que el docente pueda seleccionar e incorporar cualquiera de los datasets de dicha plataforma dentro de la imagen Docker a generar de manera sencilla. Como se puede apreciar en la Fig. 6, esto se ha conseguido mediante un formulario de texto con la opción de autocompletado.

Para hacer extensible DMLEC y que pueda integrase fácilmente en un futuro con otras plataformas tales como *Google Dataset Search*[21], se ha desarrollado un módulo externo a la propia herramienta que hace las funciones de intermediario o proxy para conectarse con los diferentes proveedores de conjuntos de datos tal y como se muestra en la Fig. 7. En este sentido, la numeración de las flechas en dicha figura indica el orden de los pasos para listar y descargar un determinado dataset.

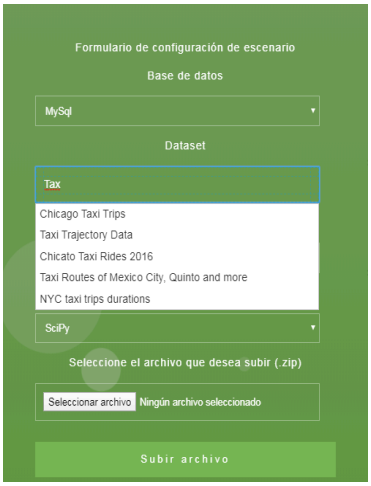

Figura 6. Ejemplo de listado de autocompletado para la selección del conjunto de datos a incluir en el entorno de prácticas.

[12] https://scikit-learn.org/stable/

[13] https://www.tensorflow.org

[14] https://keras.io/

[15] https://www.scipy.org

[16] https://netbeans.org

[17] https://www.jetbrains.com/idea/

[18] https://www.eclipse.org/ide/

[19] https://www.sublimetext.com/3

[20] https://www.kaggle.com/

[21] https://datasetsearch.research.google.com/

En la versión actual de la herramienta, dicho proxy es el encargado usar a la API en Python de Kaggle arriba comentada. En particular, el proxy inicialmente ejecuta el comando `kaggle datasets list` para obtener un listado inicial de todos los conjuntos de datos en la plataforma. Conforme el usuario va introduciendo caracteres en el campo de texto, el proxy va relanzando dicho comando incluyendo el parámetro –s con los términos de búsqueda. Cuando el usuario decide el dataset a usar, el proxy lo descarga mediante el comando Python `kaggle datasets download <nombre-dataset>`. Para ello, es necesario que dicho proxy almacene las credenciales del docente en la plataforma Kaggle para poder hacer uso de la API.

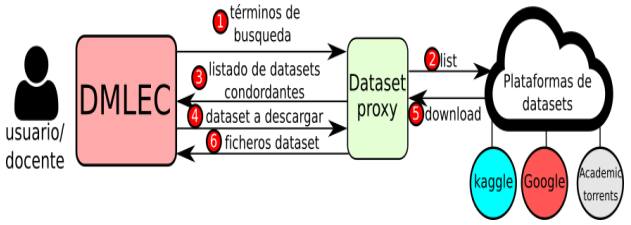

Figura 7. Esquema de funcionamiento del proxy dataset para el acceso a conjuntos de datos de forma dinámica.

Como vemos, la herramienta permite un gran abanico de opciones a la hora de generar el entorno de prácticas para los alumnos, pudiendo combinar librerías de análisis de datos e IDEs con diferentes complejidades de uso y aplicarlos a diferentes datasets.

Tal y como muestra la Fig. 3, una vez el usuario ha indicado los parámetros a través de la interfaz gráfica del cliente (flecha 1 en Fig. 3), estos se transfieren al servidor de DMLEC así como los parámetros de la instalación local de Docker en el ordenador donde se está ejecutando la herramienta (fecha 2).

A continuación, dicho módulo servidor se encarga de generar los Dockerfiles e imágenes del entorno de prácticas. En este sentido, la interfaz web permite establecer por parte del docente cómo dichas entorno va a distribuirse entre los alumnos tal y como se describe en la siguiente sección (ver Fig. 5).

F. Generación y distribución de las imágenes (módulo servidor)

Tal y como se describió en la sección III.E, la segunda de las vistas web de la interfaz gráfica permite al usuario especificar si desea o no que la herramienta genere y suba a Docker Hub las imágenes del entorno de prácticas a generar. En función de la selección realizada por el docente, la herramienta proporciona dos opciones diferentes de distribución tal y como se describe a continuación.

1) Distribución de ficheros fuente Dockerfile

En caso de que el usuario no marque la opción de subir las imágenes a Docker Hub, el módulo servidor de DMLEC simplemente genera los Dockefiles necesarios para la creación de las dos imágenes Docker que componen el entorno de prácticas. Para su generación, el módulo servidor hace uso del sistema de reglas anidadas SI-ENTONCES descritas en la sección III.D para ir componiendo cada una de las instrucciones de los Dockerfiles. Finalmente, una vez que dicho sistema de reglas ha generado todas las líneas de código de los Dockerfiles, éstos se proporcionan al usuario como una descarga directa desde el navegador.

Posteriormente, tal como indica la Fig. 8, dichos ficheros pueden distribuirse entre los alumnos a través de algún mecanismo de almacenamiento compartido. Una vez que los alumnos se descargan los ficheros, deben generar las imágenes a partir de ellos mediante una serie de comandos Docker.

Mediante esta opción de distribución damos opción al docente de que la generación del entorno de prácticas sobre los cuales los alumnos van a trabajar no sea una caja negra, sino que los propios alumnos puedan ejecutar los comandos Docker necesarios para generar ese entorno. Obviamente, esta primera manera de distribuir las imágenes tiene sentido cuando los alumnos ya están familiarizados con la tecnología Docker por haberse impartido en otras materias del grado relacionadas con la administración de sistemas.

Volviendo a la Fig. 3, esta opción de distribución abarca los pasos 1, 2, 3 y 4a del flujo de información de la herramienta.

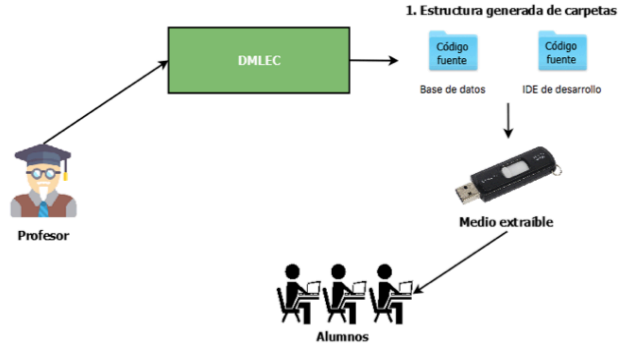

Figura 8. Distribución del entorno de prácticas mediante ficheros Dockerfile.

2) Distribución de imágenes mediante Docker Hub

Si el docente marca la opción para la generación automática de imágenes en la interfaz gráfica, DMLEC compone directamente las imágenes a partir de los ficheros Dockerfile generados por el modo anterior y subirlas finalmente al repositorio online Docker Hub tal y como muestra la Fig. 9. Para ello, el módulo servidor ejecuta el comando `docker image build` tomando como parámetros cada uno de los Dockerfiles para generar, en local, cada una de las imágenes. A continuación, dicho módulo ejecuta el comando `docker push` para subir dichas imágenes a Docker Hub. En este sentido, dichas imágenes se subirán al repositorio del usuario Docker indicado como parámetro en la primera de las vistas web de la aplicación (ver Fig. 4)

De esta forma, los alumnos solo necesitan descargarse dichas imágenes a sus máquinas locales mediante el comando `docker pull` para tener todos exactamente el mismo entorno de prácticas que el docente ha generado.

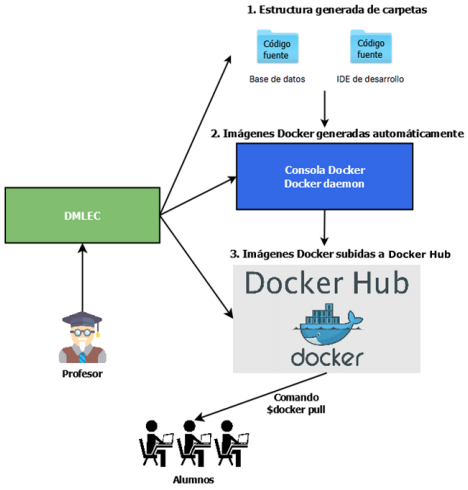

Figura 9. Distribución del entorno de prácticas mediante Docker Hub.

Por último, y volviendo a la Fig. 3, esta opción de distribución abarca los pasos 1, 2, 3, 4a y 4b del flujo de información de la herramienta.

G. *Ejecución y coordinación de los contenedores*

A fin de coordinar la ejecución de los dos contenedores asociados a las imágenes, DMLEC hace uso de la herramienta Docker Compose comentada en la sec. IIIA. Para ello, DMLEC genera automáticamente un fichero `docker-compose.yml` que permite a los dos contenedores ejecutarse en la misma red y por tanto poder conectar el entorno de programación contenido en la "Imagen Docker 2" y el entorno de datos de la "Imagen Docker 1" (ver Fig. 2).

El contenido de dicho fichero es generado dinámicamente por DMLEC dependiendo del modo de distribución seleccionado (Dockerfiles o Docker Hub). Así, por ejemplo, el contenido de dicho fichero cuando se opta por una distribución mediante Dockerfiles (sec. III.F.1) es el siguiente:

```
version: "3.7"
services:
  dev-environment:
    build:
      context: ./img2
      dockerfile: dmlec-dev-image.Dockerfile
    image:dmlec-dev-image
  data-environment:
    build:
        context: ./img1
        dockerfile: dmlec-data-image.Dockerfile
    image:dmlec-data-image
```

Como podemos ver el fichero define dos servicios (`dev-environment` y `data-environment`) para cada una de las dos imágenes generadas. Dicho fichero puede distribuirse entre el alumnado mediante algún repositorio de datos.

Finalmente, para poder lanzar ambos servicios de forma coordinada, el alumno simplemente tiene que ejecutar la instrucción `docker-compose up` en la terminal de comandos de su ordenador y sobre el mismo directorio donde se encuentra el fichero `docker-compose.yml`. Esto les abre automáticamente el IDE a usar y arranca la base de datos conteniendo el dataset sobre el que trabajar.

IV. Análisis de rendimiento de la herramienta

Con el objetivo de evaluar el rendimiento de DMLEC, hemos testeado nuestra herramienta en tres casos de uso diferentes en donde en cada uno de ellos se ha generado diferentes contenedores. Además, hemos comparado los resultados con los obtenidos si se usara un entorno de virtualización basado en máquinas virtuales completas.

La razón por la cual es interesante realizar esta evaluación del rendimiento es que, pese a que el número de prácticas y seminarios a realizar por los alumnos es limitado a lo largo de un curso escolar, también es cierto que el docente puede necesitar regenerar los contenedores en muchas ocasiones y circunstancias durante la preparación y planificación de dichos seminarios para obtener un entorno apropiado en cada nuevo curso. Por lo tanto, contar con una herramienta ágil que le permita realizar dichos testeos de una manera rápida es también importante desde el punto de la propia funcionalidad de la herramienta.

El entorno de ejecución ha consistido en un PC con 16 GB de RAM, procesador Intel core i7 a 3,40 GHz, Windows 7 como sistema operativo y Docker.

A. *Caso de uso I: Entorno para tratamiento de datos*

La configuración de este caso de uso se muestra en la Tabla 2.

Tabla 2. Configuración del caso de uso I.

Imagen 1		Imagen 2	
BBDD	**Dataset**	**IDE**	**Librería**
MySQL	*Students Performance*	*NetBeans*	*Scipy*

Este primer caso de uso se utilizó para generar un entorno de prácticas en el que los alumnos empezaron a trabajar conceptos básicos del manejo de datos mediante las librerías Numpy y Pandas embebidas en el framework Scipy. Además dicho entorno se configuró para que se pudiera usar mediante el modo de distribución basado en Docker Hub

Al lanzar DMLEC con dicha configuración obtuvimos los siguientes resultados de rendimiento. Para obtener una estimación lo más realista posible, lanzamos dicha configuración 4 veces diferentes:

- Tiempo medio para la construcción de las imágenes: 37 minutos 46 segundos.
- Desviación típica del tiempo de construcción de las imágenes: 163 segundos.
- Tamaño de la imagen 1 en Docker Hub: 94MB.
- Tamaño de la imagen 2 en Docker Hub: 721MB.

Como podemos ver en aproximadamente 40 minutos se tuvo el entorno de prácticas listo para descargar desde Docker Hub.

B. *Caso de uso II: Entorno para Deep Learning*

En este segundo caso de uso generamos un entorno de prácticas para la iniciación de los alumnos en el uso de mecanismos de *Deep Learning* mediante el framework Keras.

La Tabla 3 muestra la configuración elegida para este segundo entorno de prácticas.

Tabla 3. Configuración del caso de uso II.

Imagen 1		Imagen 2	
BBDD	**Dataset**	**IDE**	**Librería**
MariaDB	*BlackFriday*	*IntelliJIdea*	*Keras*

A la hora de generar dicho entorno, se obtuvieron los siguientes resultados de rendimiento (después de 4 ejecuciones):

- *Tiempo medio para la construcción de las imágenes: 29 minutos 11 segundos*
- *Desviación típica del tiempo de construcción de las imágenes: 200 segundos.*
- Tamaño de la imagen 1 en Docker Hub: 121MB
- Tamaño de la imagen 2 en Docker Hub: 973MB

En este segundo escenario tardamos en generar los dos contenedores aproximadamente 29 minutos.

C. *Caso de uso III: Entorno para análisis de datos*

Finalmente, hemos generado un entorno para que los alumnos usaran algunos de los algoritmos más relevantes del área del análisis de datos como Random Forest, Support Vector Machines o Linear Regresion. Todos estos mecanismos están incluidos en la librería *scikit-learn*. La Tabla 4 muestra la configuración usada para dicho entorno.

Tabla 4. Configuración del caso de uso III.

Imagen 1		Imagen 2	
BBDD	**Dataset**	**IDE**	**Librería**
PostgreSQL	*Students Performance*	*Eclipse*	*Scikit-learn*

En cuanto a los parámetros de rendimiento de la aplicación para este tercer entorno obtuvimos los siguientes valores (después de 4 ejecuciones):

- *Tiempo medio para la construcción de las imágenes: 20 minutos 17 segundos*
- *Desviación típica del tiempo de construcción de las imágenes: 155 segundos.*
- Tamaño de la imagen 1 en Docker Hub: 84MB
- Tamaño de la imagen 2 en Docker Hub: 774MB

Como podemos observar DMLEC genera los entornos de prácticas en un rango de tiempo variable entre los 20 y 40 minutos aproximadamente. Además, el tamaño de las imágenes generadas estuvo entre 800MB y 1GB de tamaño de disco. Dicho tamaño es lo suficientemente pequeño como para poder ser descargado por los alumnos en sus propios dispositivos sin problemas. A modo de resumen, la Fig. 10 muestra los tiempos de ejecución de cada escenario.

En este análisis de ejecución hemos comprobado que el factor que afecta en mayor medida al tiempo de ejecución para la generación de los entornos es el IDE incluido en el entorno a generar. Esto es debido a que es el elemento que suele requerir más tiempo para su descarga e instalación. En este sentido se observan diferencias significativas en dicho proceso entre los IDEs NetBeans (caso de uso I) e IntelliJIdea (caso de uso II) en favor del segundo respecto al primero.

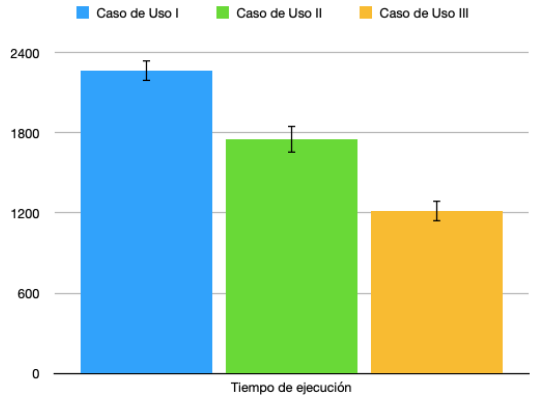

Figura 10. Tiempo de construcción de las imágenes Docker en segundos de DMLEC para la generación de cada uno de los casos de uso.

D. *Comparativa con Máquinas Virtuales Completas*

Para tener una visión más global del valor y utilidad de DMLEC, se han generado los tres casos de usos citados anteriormente, pero en este caso usando un sistema de virtualización completa. Para ello, se ha usado el software VMware Workstation 12.0 como hipervisor. En los tres casos, se usó el sistema operativo Linux Debian 10.3 como base puesto que es posible instalar todos los elementos de los tres casos de uso en él. Así, se generaron tres máquinas virtuales diferentes, una por cada caso de uso descrito en las tres secciones anteriores.

En este sentido, es cierto que podría haberse usado como hipervisor Oracle VirtualBox[22]. Dicho hipervisor es una solución de virtualización de libre distribución en contraste con VMware Workstation, sin embargo, estudios recientes indican que Workstation ofrece un rendimiento muy similar a VirtualBox permitiendo además capacidades de virtualización más completas [17,18]. Por este motivo, se decidió optar por crear las máquinas virtuales usando la herramienta de VMware ya que dicha herramienta supone una mejor solución más eficaz para entornos virtualizados con gran demanda que la herramienta de Oracle.

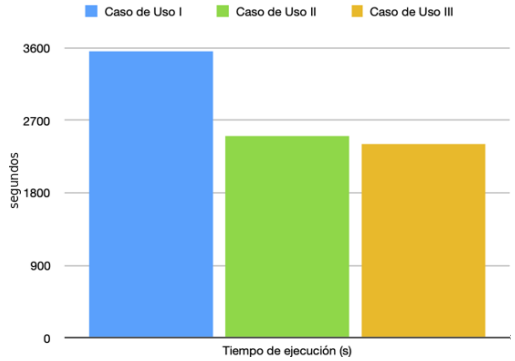

Figura 11. Tiempo en segundos usando un sistema de virtualización completa para la generación de cada uno de los casos de uso.

En la Fig. 11 se muestra el tiempo que se tardó en crear cada caso de uso desde cero. En este sentido, es importante aclarar que dichos tiempos se han tomado manualmente. Además, sólo consideran las fases durante el periodo de construcción de los casos de uso en donde no se necesitaba ninguna intervención humana. Es decir, aquellos instantes el

[22] https://www.virtualbox.org/

entorno operaba de forma automática para instalar y arrancar el sistema operativo, herramientas y librerías de cada entorno. Aquellos instantes en los que el usuario debía interaccionar con la máquina virtual a fin de realizar determinados pasos de la instalación no se han tenido en cuenta.

Como podemos ver, los tiempos registrados son sensiblemente superiores a los conseguidos por nuestra herramienta. Esto se debe a que la construcción de los tres escenarios debe ser realizada desde cero instalando y configurando el sistema operativo, librerías asociadas y aplicaciones. Por el contrario, DMLEC se beneficia del concepto de imagen, en donde una imagen Docker se construye en extendiendo una imagen anterior. Así, por ejemplo si un usuario de DMLEC decide incluir MySQL como base de datos a usar, la herramienta compondrá la primera imagen del entorno de prácticas extendiendo la imagen *mysql/mysql-server*[23] incluida en Docker Hub. Dicha imagen contiene ya un SO Linux y un servidor MySQL instalado, por lo que DMLEC no necesita ni instalar y pre-configurar dichos dos elementos con el significativo ahorro de tiempo que esto implica.

E. *Conclusiones de la evaluación de rendimiento*

Como principales conclusiones a la evaluación de rendimiento de DMLEC es importante remarcar que los tiempos de creación de los entornos con la herramienta propuesta están por debajo de los 40 minutos en los tres casos de uso tal y como se describe en las secciones IV.A, IV.B y IV.C. Estos son unos tiempos que permiten al docente generar diferentes alternativas de entornos mediante ensayos prueba-error. Por el contrario, si la herramienta necesitara de varias horas para generar dichos entornos probablemente dicho ensayo-error no sería una metodología adecuada.

Además, debemos tener en cuenta que DMLEC solo requiere que el docente interaccione con la herramienta al comienzo del proceso para definir los parámetros del entorno a generar. Una vez introducidos, la herramienta opera de forma autónoma tal y como se describe en la sección III.C. Esta es otra ventaja importante respecto a una solución basada en el uso de hipervisores (ej: VWware Workstation) que es mucho más disruptiva al necesitar la intervención del docente en diferentes fases de la instalación y configuración de las librerías y aplicaciones.

V. Satisfacción del Alumnado con los Entornos Creados Mediante DMLEC

Finalmente, se quiso estudiar el nivel de satisfacción del alumnado con los entornos de prácticas generados mediante DMLEC al usarse en la asignatura de *Sistemas Inteligentes* del grado en Ingeniería Informática de la UCAM durante el curso 2018-2019.

En dicho curso, el docente usó DMLEC para generar los tres entornos de prácticas comentados en la sección anterior. Posteriormente, los alumnos usaron las imágenes Docker creadas como entornos de prácticas de la asignatura. Además, para distribuir dichos entornos entre los alumnos, se usó una distribución basada en Docker Hub tal y como se describe en la sec. III.F.2 y los alumnos arrancaron dichos entornos siguiendo el procedimiento descrito en la sec. III.G.

A. *Características del alumnado*

Como podemos ver en la Tabla 5, durante el curso 2018-2019, se matricularon en la asignatura un total de 25 alumnos. De ellos, un total de 17 asistieron a más del 90% de las sesiones de prácticas de la asignatura. Mientras que un total de 19 alumnos se presentó en la convocatoria de Junio, 3 lo hicieron en la convocatoria de recuperación de Septiembre. En este sentido, todos los alumnos presentados en la convocatoria ordinaria como en la de recuperación tuvieron que usar la herramienta DMLEC.

Respecto al curso 2019-2020, los resultados obtenidos fueron similares respecto a la asistencia a las sesiones de prácticas. Sin embargo, sí que se apreció un incremento significativo en el porcentaje de alumnos presentados en la convocatoria de Junio respecto a otros años.

Tabla 5 Características del alumnado

Característica	2019-2020	2018-2019
Total de alumnos matriculados	27	25
Alumnos con más del 90% de asistencia	20	17
Alumnos con más del 70% de asistencia	24	21
Alumnos presentados en la convocatoria ordinaria (Junio)	26	19
Alumnos presentados en la convocatoria de recuperación (Septiembre)	2	3

B. *Resultados de la evaluación del alumnado*

La evaluación del alumnado se hizo mediante la herramienta de encuestas de valoración del profesorado de la Universidad xxx. Mediante un formulario web institucional, los alumnos pueden valorar la función del profesorado tomando como referencia diferentes parámetros al final de cada cuatrimestre.

Uno de los mismos es la valoración que los alumnos hacen de los recursos proporcionados en las asignaturas a nivel de materiales de estudio y de prácticas. En la Fig. 12 se muestra la puntuación (sobre 5) conseguida en este apartado en la asignatura de Sistemas Inteligentes para los cursos 2017-2018, 2018-2019 y 2019-2020.

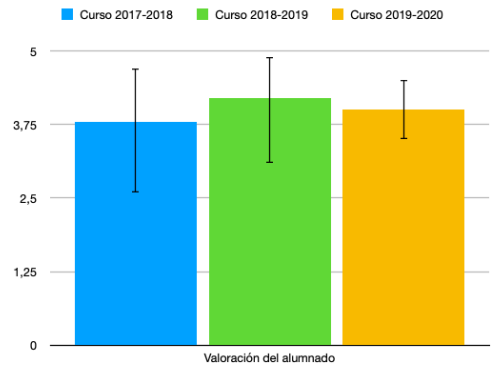

Figura 12. Valoración de los recursos de la asignatura por parte de los alumnos el curso previo (2017-2018) y los dos siguientes de instauración de DMLEC (2018-2019 y 2019-2020).

Como podemos ver, en el curso 2018-2019 conseguimos subir en 0.4 puntos la valoración media de los alumnos de los

[23] https://hub.docker.com/r/mysql/mysql-server/

recursos de la asignatura respecto al curso anterior al pasar de un 3,8 de valoración a un 4,2.

Pese a que la herramienta institucional usada para dicha encuesta no proporciona más detalles respecto a dicho apartado de recursos, es cierto que el incremento en dicho apartado fue sensiblemente superior a otros aspectos valorados tales como organización de la asignatura (de 3,9 puntos de valoración media en 2017-2018 a 4,0 en 2018-2019) o los mecanismos de evaluación (de 3,5 a 3,7 en valoración media).

La mejora en la valoración de los recursos de la asignatura respecto al curso 2017-2018 se mantuvo durante el curso 2019-2020. Es cierto que dicha valoración descendió ligeramente respecto al curso anterior (de 4,2 a 4,0) pero debemos de tener en cuenta que dicho curso se vio afectado por la pandemia del COVID-19 que obligó a desarrollar gran parte de las sesiones prácticas mediante videoconferencia. Así, otros aspectos como la organización de la asignatura (de 4,0 a 3,6) o los mecanismos de evaluación (de 3,7 a 3,4) sufrieron una mayor degradación de la valoración de los alumnos. El análisis de los motivos que ha llevado a dicha degradación está fuera del alcance del presente artículo.

Finalmente, podemos concluir que la mejora a nivel general de la valoración de los alumnos respecto a los recursos de la asignatura en estos últimos dos cursos se debe a que los estudiantes pudieron usar el entorno de prácticas para cada uno de los seminarios de una forma mucho más homogénea que en anteriores cursos. Es cierto que, en algunos casos puntuales, el docente tuvo que dar soporte a determinados alumnos para la instalación de Docker en su ordenador personal. Sin embargo, la mayoría de ellos descargaron y ejecutaron dichos entornos de forma rápida y sin incidencias. El ahorro de tiempo que ello supuso favoreció el desarrollo de las sesiones de prácticas de una forma mucho más fluida.

VI. Conclusiones

Actualmente, el alumnado universitario es cada vez más propenso a llevar sus propios dispositivos para realizar las prácticas de las diferentes materias siguiendo la tendencia de *Bring Your Own Device* (BYOD) imperante en el mercado.

Esto ha provocado una gran heterogeneidad de dispositivos que dificulta aún más si cabe el desarrollo de determinados componentes prácticos de las materias. Dicha problemática es especialmente sensible en las asignaturas relacionadas con el área del análisis de datos, en donde la variedad actual de lenguajes de programación, librerías y frameworks es muy significativa.

Por ello, el presente trabajo presenta DMLEC, una herramienta que permite la generación automática de entornos de práctica de análisis de datos utilizando la versatilidad de la virtualización basada en contenedores. Mediante una interfaz web, el docente puede generar fácilmente un entorno de prácticas con el conjunto de datos a generar y el código fuente para manipular dichos datos.

La evaluación del rendimiento de la herramienta ha determinado unos tiempos de generación de los diferentes casos de uso entre los 20 y 40 minutos. A su vez, los entornos de prácticas generados con la herramienta han sido valorados positivamente por el alumnado en base a las encuestas de evaluación realizadas por la propia universidad.

Finalmente, como trabajos futuros, se estudiará la integración de la herramienta con nuevos repositorios de datos online como *Google Datasets* o *Github*. Además, se optimizará el código de la herramienta para acelerar el proceso de generación de los escenarios.

Referencias

[1] P. J. Guo, "Online python tutor: embeddable web-based program visualization for cs education" in *Proc. 44th ACM technical symposium on Computer science education*, 2013.

[2] I-Y. Song, Il-Yeol, and Y. Zhu, "Big data and data science: what should we teach?", *Expert Systems*, vol 33, no. 4, pp. 364-373, 2016.

[3] R. Afreen. "Bring your own device (BYOD) in higher education: opportunities and challenges", *International Journal of Emerging Trends & Technology in Computer Science*, vol. 3, no. 1, pp. 233-236, 2014.

[4] B. Arshdeep, and V. Madisetti. "Cloud computing: A hands-on approach", *CreateSpace Independent Publishing Platform*, 2013.

[5] M. Portnoy, "Virtualization Essentials", *John Wiley & Sons Inc*, 2012.

[6] J. Nickoloff, and S. Kuenzli, "Docker in Action", *Manning Publications*, 2018.

[7] C. Holdgraf, "Introducing repo2docker", 2018. URL https://blog.jupyter.org/introducing-repo2docker-61a593c0752d.

[8] S., František, R. Sohlich, and T. Dulík, "Docker as platform for assignments evaluation", *Procedia Engineering,* vol. 100, pp. 1665-1671 2015.

[9] M. Chae, S. Han, and H. Lee, "Docker-based Cloud System for Computer Programming Labs", in *Proc. 2019 14th International Conference on Computer Science & Education (ICCSE),* pp. 622-626, 2019.

[10] F. Tuo, Y. Bai, S. Long, Y. Luo, T. Wang, R. Wang, and G. Yin, "A New Model of Docker-based E-learning in Hadoop", in Proc. *2018 International Conference on Distance Education and Learning (ICDEL '18)*, pp. 22–31, 2018.

[11] H. Yibo, B. O. Zheng Zhang, B. Haixia, N.S. Mohammad, and L. Lu, "An Experimental Online Judge System Based on Docker Container for Learning and Teaching Assistance", in *Proc. 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 1462-1467, 2019.

[12] J. Martínez, J.S. Ortega, and J.A. Fernández, "Laboratorios virtuales de redes: SÍ, inténtelo en casa" in *Proc. XVII Jornadas sobre la Enseñanza Universitaria de la Informática*, 2011.

[13] A. Ruiz-Martínez, and P. Martínez, "Enseñanza práctica de Arquitectura de Redes mediante la Virtualización de Escenarios de Red", *Ocnos: Revista de estudios de literatura*, 2015.

[14] J. Muñoz-Calle, F. J. Fernández-Jiménez, T. Ariza, A. J. Sierra, and J. M. Vozmediano, "Laboratorios Informáticos sobre Entornos Virtuales: Un Modelo Flexible, Portable y Multidisciplinar", *Revista Iberoamericana de Tecnologías del Aprendizaje*, vol. 4, no. 1, pp. 1-8, 2016.

[15] M. Hall, F. Eibe, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update", *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10-18, 2009.

[16] A.M. Joy, "Performance comparison between linux containers and virtual machines", in *Proc. 2015 International Conference on Advances in Computer Engineering and Applications*, 2015.

[17] D. T. Vojnak, B. S. Đorđević, V. V. Timčenko y S. M. Štrbac, "Performance Comparison of the type-2 hypervisor VirtualBox and VMWare Workstation", in *Proc. 2019 27th Telecommunications Forum (TELFOR)*, pp. 1-4, 2019.

[18] L. Peng, "Selecting and using virtualization solutions: our experiences with VMware and VirtualBox", *Journal of Computing Sciences in Colleges*, vol. 25, no. 3, pp. 11-17, 2010.