Instalación del software necesario

En este documento daremos los pasos necesarios para instalar todo el software que utilizaremos durante el curso. Se darán las pautas de instalación en un sistema Linux basado en Debian, como por ejemplo Ubuntu, Lubuntu, Linux Mint, etc. Recuerda que dispones de una máquina virtual Lubuntu para VirtualBox, con todo el software ya preinstalado, en el caso de que no quieras perder tiempo con estos pasos.

En concreto, los elementos que vamos a instalar a lo largo de este documento son:

- El editor de código que emplearemos, **Visual Studio Code**.
- Un sistema **XAMPP** para poder trabajar con Apache, PHP y bases de datos MariaDB/MySQL, además de disponer de la aplicación *phpMyAdmin* para gestionar las bases de datos.
- La herramienta **Postman**, que emplearemos para probar los servicios REST que desarrollaremos en alguna sesión.
- El framework **Laravel**, como piedra angular de este curso.
- Adicionalmente, también instalaremos **Node.js**, ya que incorpora el gestor de paquetes **NPM**, que nos servirá para instalar algunas dependencias en la parte del cliente en los proyectos Laravel.

1. Visual Studio Code

Como IDE para desarrollar nuestras aplicaciones emplearemos **Visual Studio Code**, que es uno de los IDEs más versátiles que existen hoy en día para desarrollo web.

Desde la web oficial de Visual Studio Code podemos descargarlo para la plataforma deseada.

En el caso de **Linux Debian** (nuestra máquina virtual Lubuntu o una distribución similar), descargaremos un archivo *.deb*. Una vez descargado, accedemos por terminal a la carpeta donde esté y ejecutamos este comando para instalarlo:

```
sudo dpkg -i nombre_del_archivo.deb
```

Se creará automáticamente un acceso directo en el menú de inicio, dentro de la sección de *Programación* en el caso de Lubuntu.

2. Apache, PHP y MariaDB/MySQL con XAMPP

Para poder tener un sistema con Apache, PHP y un gestor de bases de datos (como MariaDB/MySQL), y poderlo gestionar cómodamente, trabajaremos con un sistema AMPP, paquetes que integran en una sola instalación todas estas cosas. El ejemplo más conocido de estos sistemas es **XAMPP**, aunque existen otros como WAMPP, para Windows. Una de las ventajas que ofrecen es que, además de instalar Apache, PHP y MySQL y dejarlo todo integrado, nos proporciona un cliente web llamado **phpMyAdmin** para poder administrar las bases de datos desde Apache. Esto nos vendrá bien para crear o importar las bases de datos de los distintos ejercicios.

2.1. Instalación

Para instalar XAMPP, basta con descargarlo de su web oficial y seguir los pasos del asistente. Nos basta con tener instalado Apache, MySQL y PHP, así que podemos descartar otras opciones que nos ofrezca, si nos da a elegir.

En el caso de Linux, debemos dar permisos de ejecución y ejecutar el archivo *.run* que descarguemos desde algún terminal, con permisos de administrador (*sudo*). Suponiendo que el archivo se llame *xampp-linux-x64-7.4.5-installer.run*, por ejemplo, los pasos son los siguientes (desde la carpeta donde lo hemos descargado):

```
sudo chmod +x xampp-linux-x64-7.4.5-installer.run
sudo ./xampp-linux-x64-7.4.5-installer.run
```

Se iniciará un asistente que instalará XAMPP. El *manager* de XAMPP será la herramienta que usaremos para ponerlo en marcha, y se encuentra en **/opt/lampp/manager-linux-x64.run**. Podemos acceder a la carpeta desde el terminal para ejecutarlo (con permisos de superusuario), o bien crear algún acceso directo en otra ubicación que nos resulte más cómoda.

Por ejemplo, podemos crear un acceso directo en el escritorio con el editor *nano* (suponiendo la carpeta */home/alumno/Escritorio/*, como la que tenemos en la máquina virtual):

```
nano /home/alumno/Escritorio/XAMPP.desktop
```

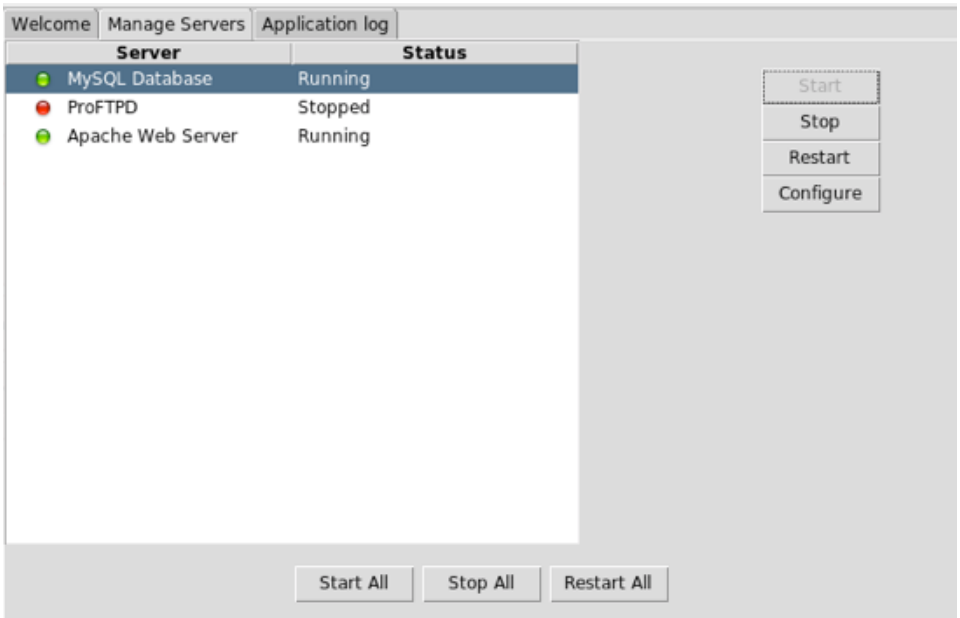
Editamos el contenido del archivo y añadimos las líneas de configuración para el acceso directo:

```
[Desktop Entry]
Encoding=UTF-8
Name=Manager XAMPP
Comment=Manager XAMPP
Exec=sudo /opt/lampp/manager-linux-x64.run
Icon=/opt/lampp/htdocs/favicon.ico
Categories=Aplicaciones;Programación;Web
Version=7.4.5
Type=Application
Terminal=1
```

> **NOTA**: la versión del programa (atributo *Version*) dependerá de la versión que hayamos instalado de XAMPP en el momento concreto. El atributo *Terminal* lo ponemos a 1 para poder especificar el password de superusuario al ejecutar, de lo contrario no funcionará.

2.2. Puesta en marcha y parada.

El manager nos permitirá lanzar o detener cada servidor. Para las pruebas que haremos deberemos tener iniciados tanto Apache como MySQL:



Por defecto, Apache estará escuchando en el puerto 80 (o 443 para conexiones SSL), y MySQL en el 3306. Podemos modificar estos puertos en los respectivos archivos de configuración ("*httpd.conf*" y "*my.cnf*"), dentro de las carpetas de la instalación de XAMPP (la ubicación concreta de estos archivos varía entre versiones y entre sistemas operativos).

3. Postman

Postman es una aplicación que permite probar aplicaciones cliente-servidor, simulando peticiones a un servidor y recogiendo la respuesta de éste.

Para descargar e instalar Postman, debemos ir a su web oficial, a la sección de descargas, y descargar la aplicación (versión gratuita). Es un archivo portable, que se descomprime y dentro está la aplicación lista para ejecutarse.

La primera vez que la ejecutemos nos preguntará si queremos registrarnos, de forma que podamos compartir los proyectos que hagamos entre los distintos equipos en que estemos registrados, pero podemos saltar este paso haciendo clic en el enlace inferior de la ventana que aparecerá.

4. Laravel

Para trabajar con Laravel, será necesario instalar el gestor de paquetes **composer** mediante el que podremos tanto crear proyectos Laravel como gestionar las dependencias de otros módulos en un proyecto.

4.1. Instalando *composer*

Como hemos comentado, la instalación de Laravel se realiza a través del gestor de paquetes **composer**. Éste es una herramienta muy habitual en ecosistemas PHP, y su labor es similar a la que desempeña el gestor NPM para aplicaciones JavaScript: gestionar las dependencias de un determinado proyecto, descargando, actualizando o desinstalando los paquetes necesarios. En este caso, lo utilizaremos para descargar e instalar el propio framework Laravel.

Composer puede instalarse localmente para cada proyecto web, o de forma global para todo el sistema. Esta última opción es la recomendable en el caso de querer gestionar varios proyectos en nuestro equipo, para no tener que instalarlo en todos ellos. Para hacer esto, debemos descargar el archivo `composer.phar` de la web oficial y copiarlo renombrado a `composer` desde donde lo hayamos descargado a alguna carpeta que forme parte del PATH del sistema, y activarlo como ejecutable. Por ejemplo:

```
sudo mv composer.phar /usr/local/bin/composer
sudo chmod +x /usr/local/bin/composer
```

Como último paso, y ya que Composer utiliza el ejecutable de PHP, necesitamos que dicho ejecutable esté también en el PATH del sistema, para lo que haremos lo siguiente:

```
echo "export PATH=$PATH:/opt/lampp/bin" >> ~/.bashrc
source ~/.bashrc
```

La primera línea añade la carpeta */opt/lampp/bin* al PATH, con lo que ya se puede localizar el comando *php* de XAMPP a nivel global. La segunda recarga el fichero *.bashrc* para hacer efectivos los cambios.

4.2. Instalando Laravel

A través de la herramienta `composer` se pueden crear directamente proyectos Laravel, como veremos en el curso. Sin embargo, la sintaxis del comando de creación es algo larga, si la comparamos con el instalador de Laravel, por lo que vamos a instalarlo también. Para hacerlo, usamos la propia herramienta `composer`, con este comando:

```
composer global require laravel/installer
```

Después, añadimos el instalador al PATH del sistema (en realidad, añadimos la carpeta con las utilidades que *composer* instala de forma global al sistema):

```
echo "export PATH=$PATH:$HOME/.config/composer/vendor/bin" >> ~/.bashrc
source ~/.bashrc
```

> **NOTA**: en algunos sistemas la carpeta que hay que incluir en el PATH es `$HOME/composer/vendor/bin` en lugar de la anterior.

Con esto, se habrá instalado un comando llamado `laravel`, que podemos utilizar a partir de ahora para crear los proyectos. Podemos probar a ejecutarlo en un terminal para que nos muestre las opciones disponibles, lo que indicará que está correctamente instalado y localizado.

4.3. Actualizando Laravel

Por defecto, el comando `composer` anterior instala la última versión disponible de Laravel. Con el paso del tiempo, evidentemente estas versiones se irán actualizando. Para poder actualizar a la versión más reciente de Laravel, tenemos dos opciones, aunque es cierto que ninguna de ellas está recogida en la documentación oficial de Laravel, y lo que aquí se menciona se basa en recomendaciones de webs externas a Laravel.

La primera forma de actualizar es utilizar el comando de actualización:

```
composer global update laravel/installer
```

Sin embargo, esta opción puede no ser suficiente si el cambio es demasiado brusco (por ejemplo, pasar de Laravel 5 a Laravel 7), ya que algunas dependencias que también haya instaladas harían inviable el cambio. En este caso, podemos optar por quitar la versión instalada por completo, e instalar la reciente:

```
composer global remove laravel/installer
composer global require laravel/installer
```

5. Node.js

A pesar de que podría parecer que *Node.js* es un ecosistema diferente a Laravel, lo cierto es que con la instalación de Node se incorpora una herramienta muy útil en cualquier aplicación web que utilice librerías JavaScript, como puedan ser Bootstrap o jQuery. Es la herramienta **NPM** (*Node Package Manager*), que permite instalar de forma sencilla estas librerías en cualquier proyecto.

Para instalar Node en un sistema Linux, haremos uso de otra herramienta llamada **NVM** (*Node Version Manager*), una herramienta que nos permite fácilmente instalar distintas versiones de Node, desinstalar versiones obsoletas, o elegir en cualquier momento cuál de las versiones que tenemos instalada queremos dejar activa. Podemos consultar información en su web oficial en GitHub.

Para instalar NVM, debemos descargarlo con el comando `curl` o `wget`, según se explica en la propia web de GitHub. Si optamos por `wget`, el comando es como sigue (en una sola línea):

```
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/
v0.35.3/install.sh | bash
```

En el caso de no disponer del comando `wget` instalado, podemos o bien instalarlo, o bien emplear este otro comando equivalente, con la orden `curl` (también en una sola línea):

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/
v0.35.3/install.sh | bash
```

> **NOTA**: el número de versión `v0.35.3` puede variar. Es preferible consultar la web de GitHub para obtener el comando actualizado.

> **NOTA**: después de ejecutar el comando anterior, será necesario cerrar el terminal y volverlo a abrir para poder utilizar el comando `nvm`.

Ya tenemos `nvm` instalado en el sistema. Aquí mostramos algunas de las opciones más interesantes que podemos utilizar:

- `nvm install node`: instala la última versión disponible de Node
- `nvm install --lts`: instala la última versión LTS disponible
- `nvm install 12.16.0`: instala la versión especificada de Node
- `nvm uninstall 12.16.0`: desinstala la versión especificada de Node
- `nvm ls-remote`: muestra todas las versiones disponibles para instalar

- `nvm list` : muestra todas las versiones instaladas localmente
- `nvm current` : muestra la versión actualmente activa
- `nvm use 12.16.0` : marca la versión indicada como actualmente activa
- `nvm use --lts` : marca como activa la última versión LTS instalada

En nuestro caso, vamos a instalar la última versión LTS disponible, ya que éstas son las versiones que tienen soporte a largo plazo. Por lo tanto, ejecutaremos el comando:

```
nvm install --lts
```

Podemos ejecutar ahora `node -v` en el terminal y comprobar que nos muestra el número de versión adecuado. También podemos ejecutar el comando `npm - v` para comprobar la versión que se ha instalado del gestor NPM (que no tiene por qué coincidir con la de Node).