Proyecto:

Gestión del Ciclo de Vida del Desarrollo del Software (SDLC) en la Carrera de Ingeniería en Sistemas/Computación de la UNL

Manual de Instalación del Sistema Web

1. Introducción

El presente documento contiene información relacionada al uso de herramientas y los procesos necesarios a seguir para llevar a cabo la correcta instalación del sistema web desarrollado durante el presente TT en distintos entornos (Desarrollo y Producción).

1.1. Objetivo

El propósito del presente documento es detallar el proceso necesario a seguir para realizar la instalación del sistema web para la Gestión del Ciclo de Vida de Desarrollo de Software dentro de un entorno de desarrollo usando diferentes sistemas operativos (Windows y Ubuntu) así como dentro de un entorno de producción.

1.2. Alcance

Este documento va dirigido a todos los integrantes, tanto administrativos como estudiantes de la Carrera de Ingeniería en Sistemas/Computación de la Universidad Nacional de Loja, con la finalidad que puedan instalar y ejecutar de forma adecuada el sistema web para la Gestión del Ciclo de Vida de Desarrollo de Software.

1.3. Requisitos Previos

Entorno de Desarrollo

- Tener previamente instalado MongoDB.
- Tener previamente instalado Node.js.
- Tener previamente instalado Angular.

Ambos Entornos

- Tener acceso al repositorio de backend (https://gitlab.com/franzflores21/backend.git).
- Tener acceso al repositorio de frontend (https://gitlab.com/franzflores21/pm-frontend.git).
- Tener previamente instalado un Editor de código.
- Tener previamente instalado Git.
- Tener previamente instalado herramientas para uso de SSH.

1.4. Restricciones

- No podrá subir cambios al repositorio en caso de no ser un usuario autorizado.
- Las versiones antiguas de Node.js pueden ser incompatibles con las dependencias de los proyectos (Revisar las versiones compatibles con Angular 13).

2. Instalación y Configuración del Software

A continuación se presenta el procedimiento para realizar la instalación del sistema web en diferentes entornos junto con sus respectivas configuraciones y puesta en marcha.

2.1. Entorno de Desarrollo

2.1.1. Instalación de dependencias en Windows

Para empezar con la instalación se debe ingresar a la consola (CMD) y dirigirse a la carpeta donde se desea clonar el código, luego ejecutar el siguiente comando para descargar el código del backend:

```
git clone https://gitlab.com/franzflores21/backend.git
```

De igual manera, ejecutar el siguiente comando para la clonación del código del frontend

```
git clone https://gitlab.com/franzflores21/pm-frontend.git
```

Una vez clonados los repositorios, se debe acceder a cada uno de los proyectos y ejecutar el siguiente comando:

```
npm install
```

2.1.2. Instalación de dependencias en Ubuntu

La instalación del sistema web en Ubuntu inicia abriendo una terminal e ingresando dentro de la carpeta en la cual se desea clonar el código. Luego se ejecuta el comando:

```
apt-get update
```

Posteriormente, se procede a la clonación del código del backend.

```
git clone https://gitlab.com/franzflores21/backend.git
```

De igual manera, se ejecuta el comando para la clonación del código del frontend.

```
git clone https://gitlab.com/franzflores21/pm-frontend.git
```

Una vez clonados los repositorios, se debe acceder a cada uno de los proyectos y ejecutar el siguiente comando:

```
npm install
```

2.1.3. Configuraciones

Una vez instaladas las dependencias en ambos proyectos, se procede a abrir mediante un editor de código el proyecto de backend y en el directorio raíz, crear un archivo llamado *.env,* donde se agregará el siguiente código:

```
NODE_ENV=development
PORT=3000
```

```
DB_URI=<uri de la BD de MongoDB>

TOKEN_SECRET=secret
TOKEN_EXPIRA=24h

CLOUD_NAME=<nombre de la nube>
API_KEY=<llave del api>
API_SECRET=<token secreto del servidor de archivos>

EMAIL_SMTP_HOST=<host del email>
EMAIL_SMTP_PORT=<puerto del servicio de email>
EMAIL_SMTP_USER=<usuario del email>
EMAIL_SMTP_PASS=<clave del email>

FRONTEND_URL=<URL del frontend>
```

Adicionalmente, en el directorio raíz del backend se debe crear una carpeta llamada *uploads,* dentro de la misma se deben crear las carpetas *attachments*, *projects* y *users*, además se debe crear la carpeta *reports*.

2.1.4. Ejecución del Proyecto

Una vez instalado el sistema web y realizada las respectivas configuraciones, dentro de ambos proyectos se ejecuta el siguiente comando

```
npm start
```

En caso de querer ejecutar las pruebas unitarias e integración se debe ejecutar el siguiente comando:

```
npm run test
```

2.2. Entorno de Producción (Ubuntu 20.04)

La instalación en este tipo de entornos por lo general se realiza a través de SSH. Para el propósito de esta guía, se asume que ya se tienen las herramientas necesarias para ejecutar la conexión y sus respectivos comandos.

2.2.1. Instalación de dependencias requeridas

Es necesario preparar la instancia de Ubuntu antes de proceder con la instalación de Node.js y MongoDB, por lo que se deben ejecutar los siguientes comandos en la consola:

```
sudo apt-get update && sudo apt-get upgrade -y
sudo apt-get install build-essential -y
```

```
sudo apt install build-essential checkinstall zlib1g-dev -y
sudo apt-get install -y software-properties-common
```

Luego se instala y se configura el SSH para la conexión remota:

```
sudo apt-get install openssh-server -y
sudo nano /etc/ssh/sshd_config -> PasswordAuthentication yes
sudo /etc/init.d/ssh restart
sudo systemctl status ssh
sudo passwd ubuntu -> ubuntu se puede reemplazar con el usuario actual
```

2.2.3. Instalación de Nginx

Debido a que Nginx está disponible en los repositorios predeterminados de Ubuntu, es posible instalarlo desde los mismos utilizando el sistema de empaquetado apt:

```
sudo apt install nginx -y
```

Antes de proceder a la configuración de Nginx, es necesario ajustar el software del firewall para permitir el acceso al servicio. Para este propósito, se registra a Nginx como un servicio en el ***ufw*** al momento de la instalación, lo que facilita su acceso.

Para ver el listado se debe ejecutar el siguiente comando:

```
sudo ufw app list
```

Para permitir el tráfico, es necesario habilitar el puerto 80 de Nginx:

```
sudo ufw allow 'Nginx HTTP'
```

Por último se verifica si la acción se completó correctamente con el siguiente comando:

```
sudo ufw status
```

Una vez realizada la acción anterior, es necesario configurar los bloques del servidor, los cuales permiten encapsular los detalles de configuración y alojar más de un dominio desde un solo servidor. En este punto se da por hecho que el ***dominio*** y el ***subdominio*** (perteneciente a la API) apuntan a la IP del servidor, para lo cual se ejecutan los siguientes comandos:

```
cd ~sudo mkdir -p /var/www/dominio/html
sudo chown -R $USER:$USER /var/www/dominio/html
sudo chmod -R 755 /var/www/dominio/html
```

Se debe tener en cuenta que dentro de esta carpeta se alojarán los archivos generados del comando ***ng build*** del frontend de angular. Más adelante se indicará la configuración para el backend usando el ***subdominio***.

Una vez creados los directorios, se procede a crear el archivo de la configuración:

```
sudo nano /etc/nginx/sites-available/dominio
```

La configuración recomendada para los archivos generados de **Angular**, es la siguiente:

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/dominio/html;
        index index.html index.htm index.nginx-debian.html;

        server_name dominio www.dominio;

        location / {
                try_files $uri $uri/ /index.html?$args;
        }
}
```

Es posible repetir el mismo proceso anterior para crear el bloque del API del backend con un subdominio, por lo que se presenta el último comando para el archivo y la configuración recomendada:

```
sudo nano /etc/nginx/sites-available/sudominio
```

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/subdominio/html;
        index index.html index.htm index.nginx-debian.html;

        server_name subdominio www.subdominio;

        location / {
              proxy_pass http://localhost:3000;
              proxy_http_version 1.1;
              proxy_set_header Upgrade $http_upgrade;
              proxy_set_header Connection 'upgrade';
```

```
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;

        }
}
```

Luego de hacer las configuraciones respectivas, se debe reiniciar el servicio de **Nginx**:

```
sudo nginx -t
sudo systemctl restart nginx
```

2.2.2. Instalación de Node.js (v16)

Una vez efectuadas las configuraciones necesarias para que la instancia funcione correctamente, se procede a instalar Node.js. Para ello, se ejecutan los siguientes comandos:

Primero, se debe instalar el PPA de NodeSource para obtener acceso a su contenido. Hay que asegurarse de estar en el directorio de inicio y utilizar ***curl*** para recuperar el script de instalación para la versión LTS más reciente de Node.js de sus archivos.

```
cd ~
curl -sL https://deb.nodesource.com/setup_16.x -o nodesource_setup.sh
sudo bash nodesource_setup.sh
```

El PPA se agregará a la configuración y el caché de paquetes locales se actualizará automáticamente. Después de ejecutar el script de instalación desde Nodesource, ya es posible instalar el paquete Node.js:

```
sudo apt install nodejs -y
```

Una vez instalado se puede verificar la versión con el siguiente comando:

```
node -v
npm -v
```

2.2.3. Instalación de PM2

Otro paquete necesario para el despliegue de una aplicación de Node.js es PM2, el cual es un administrador de procesos que permite demonizar aplicaciones para que se ejecuten en segundo plano, como un servicio. Para su instalación, se usa ***npm***, por lo cual se ejecuta el siguiente comando:

```
sudo npm install pm2@latest -g
```

Una vez instalado, es necesario ejecutar el archivo principal del backend de la aplicación, para lo cual se clona el proyecto:

```
git clone https://gitlab.com/franzflores21/pm-backend.git
```

Una vez clonado, es necesario ingresar a la carpeta y apuntando al archivo principal, se ejecuta el comando de pm2:

```
pm2 start bin/www
```

Para este caso, en la lista de demonios de ***pm2***, se guardará automáticamente con el nombre ***www*** lo que permitirá identificar y ejecutar varios comandos de administración sobre la aplicación recién desplegada, como por ejemplo:

```
pm2 restart www -> para reiniciar la aplicación
pm2 logs www -> para ver los logs
```

2.2.2. Configuraciones

Como se ha detallado previamente, es necesario crear un archivo ***.env*** en la raíz de la carpeta clonada del proyecto, que permita fijar las variables de entorno en el backend de la aplicación, por lo cual se detalla a continuación la configuración recomendada:

```
NODE_ENV=development
PORT=3000

DB_URI=<uri de la BD de MongoDB>

TOKEN_SECRET=secret
TOKEN_EXPIRA=24h

CLOUD_NAME=<nombre de la nube>
API_KEY=<llave del api>
API_SECRET=<token secreto del servidor de archivos>

EMAIL_SMTP_HOST=<host del email>
EMAIL_SMTP_PORT=<puerto del servicio de email>
EMAIL_SMTP_USER=<usuario del email>
EMAIL_SMTP_PASS=<clave del email>

FRONTEND_URL=<URL del frontend>
```

Se reinicia el demonio nuevamente y se comprueba el funcionamiento:

```
pm2 restart www -> para reiniciar la aplicación
pm2 logs www -> para ver los logs
```

Hay que tener en cuenta que es posible fijar dichas variables a través de ***pm2*** por lo que se recomienda usar cualquiera de las dos alternativas.