

[Subtítulo del documento]

Contenido

PRACTICA 7: DESPLIEGUE	¡Error! Marcador no definido.
Conceptos indispensables.....	1
Definiciones.....	1
Grpc	3
MLFlow.....	5
CML.....	5
DVC.....	5
PM2	6
ENTREGA	6
Servidores web	6
MI PLAN	8
Elaboración de la practica.....	10
I. Preparación del entorno de producción	10
Breve repaso de IP y DNS	12
II. Configuración del sistema	17
Configuración de la base de datos.....	17
III. Gestor de dependencias	20
IV. Automatización de tareas.....	21
Adios, adios.....	¡Error! Marcador no definido.

Conceptos indispensables

Definiciones

1.Blob: En programación, el término "blob" es una abreviatura de "Binary Large Object" (Objeto Binario Grande, en español). Se refiere a un tipo de dato utilizado para almacenar

datos binarios, como imágenes, videos, archivos de audio u otros tipos de información no estructurada.

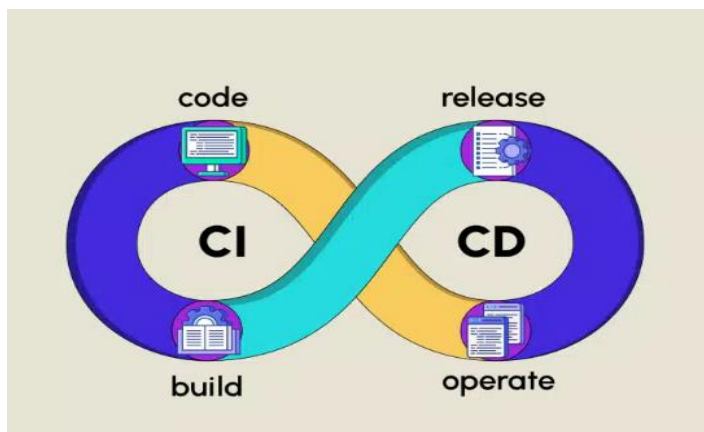
2.DevOps (acrónimo inglés de development -desarrollo- y operations -operaciones-) es un conjunto de prácticas que agrupan el desarrollo de software (Dev) y las operaciones de TI (Ops). Su objetivo es hacer más rápido el ciclo de vida del desarrollo de software y proporcionar una entrega continua de alta calidad.

Las características de DevOps provienen de la metodología Agile (término en inglés para la metodología de desarrollo ágil).

3.CI/CD :En ingeniería de software, IC/EC, ICEC, CI/CD, o CICD generalmente refiere a las prácticas combinadas de integración continua y entrega continua (también conocida como despliegue continuo).

3.1 Integración continua: Hacer *integraciones automáticas* de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes.

3.2 Entrega continua: Los equipos de desarrollo producen software en ciclos cortos, asegurando que el software puede ser liberado de forma confiable en cualquier momento.



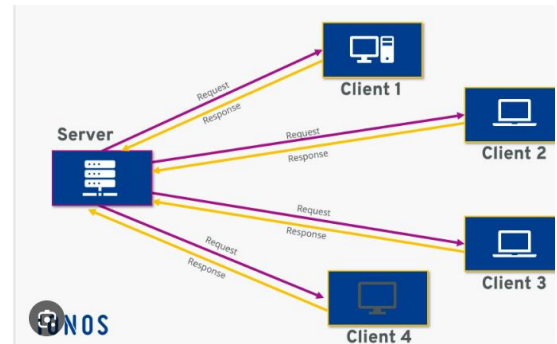
4.Proxy: Un proxy, en el contexto de redes y tecnología de internet, es un intermediario que actúa entre un usuario y el servidor al cual está realizando una solicitud. Su función principal es servir como un punto de conexión entre estos dos extremos, facilitando el intercambio de información.



5.Node.js: Es un entorno de ejecución de JavaScript en el lado del servidor. Se diferencia de otros archivos .js en que en vez de ser un archivo de código fuente escrito en Javascript es un entorno de ejecución para Javascript.

6.Servidor: En el ámbito de la informática, un servidor es un sistema o software diseñado para proporcionar servicios, recursos o funcionalidades a otros dispositivos o programas, conocidos como clientes. Los servidores desempeñan un papel fundamental en la comunicación y el intercambio de información en redes de computadoras.

Los clientes utilizan navegadores web que se conectan a los servidores y proveen la pagina web HTML en cuestión, aunque también pueden ofrecer otros servicios como proveer CSS



Grpc

gRPC (gRPC Remote Procedure Call) **es una herramienta que facilita la comunicación entre aplicaciones distribuidas**. Es un framework de código abierto desarrollado por Google que utiliza el protocolo HTTP/2 para la comunicación entre servicios.

En una aplicación distribuida en lugar de ejecutarse en una única máquina, los diversos componentes se ejecutan en múltiples máquinas interconectadas a través de una red. Estos componentes colaboran y se comunican entre sí para lograr un objetivo común.

Aquí se tiene un sencillo ejemplo de uso:

You
me puedes poner un ejemplo de uso de grpc?

ChatGPT

Claro, aquí te proporcionaré un ejemplo sencillo de uso de gRPC con Python. En este ejemplo, crearemos un servicio de calculadora con dos métodos: uno para sumar números y otro para restar. Utilizaremos archivos de definición de servicios (protobuf) para especificar la interfaz del servicio.

1. Definición del servicio (calculator.proto):

Crearemos un archivo "calculator.proto" que define la interfaz del servicio utilizando protobuf.

```
protobuf
Copy code

syntax = "proto3";

package calculator;

service Calculator {
  rpc Add (AddRequest) returns (AddResponse);
  rpc Subtract (SubtractRequest) returns (SubtractResponse);
}

message AddRequest {
  int32 num1 = 1;
  int32 num2 = 2;
}

message AddResponse {
  int32 result = 1;
}

message SubtractRequest {
  int32 num1 = 1;
  int32 num2 = 2;
}

message SubtractResponse {
  int32 result = 1;
}
```

1. Generación de Código:

Utilizaremos el compilador de protobuf para generar el código cliente y servidor en Python.



```
if __name__ == '__main__':
    run_server()
```

1. Implementación del Cliente (calculator_client.py):

Crearemos un cliente gRPC que utiliza los métodos del servicio.

```
python
Copy code

import grpc
import calculator_pb2
import calculator_pb2_grpc

def run_client():
    channel = grpc.insecure_channel('localhost:50051')
    stub = calculator_pb2_grpc.CalculatorStub(channel)

    # Llamada al método Add
    add_request = calculator_pb2.AddRequest(num1=10, num2=5)
    add_response = stub.Add(add_request)
    print(f"Add Result: {add_response.result}")

    # Llamada al método Subtract
    subtract_request = calculator_pb2.SubtractRequest(num1=10, num2=5)
    subtract_response = stub.Subtract(subtract_request)
    print(f"Subtract Result: {subtract_response.result}")

if __name__ == '__main__':
    run_client()
```

1. Ejecución del Servidor y Cliente:

Ejecuta el servidor y luego el cliente en terminales separadas.

```
bash
Copy code

# Terminal 1: Ejecutar el servidor
python calculator_server.py

# Terminal 2: Ejecutar el cliente
python calculator_client.py
```

Este ejemplo básico muestra cómo definir un servicio con gRPC, generar el código en Python, implementar un servidor que ofrece los métodos del servicio, y crear un cliente que llama a esos métodos. Este es solo un punto de partida, y gRPC ofrece muchas más características y opciones avanzadas para implementaciones más complejas y eficientes en entornos distribuidos.

Message ChatGPT...



```
calculator.proto
}
```

1. Generación de Código:

Utilizaremos el compilador de protobuf para generar el código cliente y servidor en Python.

```
bash
Copy code

# Instalar grpcio-tools
pip install grpcio-tools

# Generar archivos Python a partir del archivo de definición de servicio
python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. calc
```

1. Implementación del Servidor (calculator_server.py):

Crearemos un servidor gRPC que implementa los métodos definidos en el archivo "calculator.proto".

```
python
Copy code

import grpc
import calculator_pb2
import calculator_pb2_grpc

class CalculatorServicer(calculator_pb2_grpc.CalculatorServicer):
    def Add(self, request, context):
        result = request.num1 + request.num2
        return calculator_pb2.AddResponse(result=result)

    def Subtract(self, request, context):
        result = request.num1 - request.num2
        return calculator_pb2.SubtractResponse(result=result)

def run_server():
    server = grpc.server(grpc.ThreadPoolExecutor(max_workers=10))
    calculator_pb2_grpc.add_CalculatorServicer_to_server(CalculatorServicer(), server)
    server.add_insecure_port('[::]:50051')
    server.start()
    server.wait_for_termination()

if __name__ == '__main__':
    run_server()
```

1. Implementación del Cliente (calculator_client.py):

Crearemos un cliente gRPC que utiliza los métodos del servicio.

MLFlow

MLflow es una plataforma de código abierto diseñada para gestionar el ciclo de vida completo de un proyecto de aprendizaje automático (machine learning). Proporciona herramientas y funcionalidades para ayudar en la experimentación, el seguimiento, la reproducción y la implementación de modelos de aprendizaje automático.

En el siguiente enlace se puede ver un ejemplo de uso:

<https://chat.openai.com/share/2c5c01ae-0199-4ab7-86fe-e050021801ba>

aws

CML

La herramienta "CML" (Continuous Machine Learning) es una plataforma que facilita la **integración de flujos de trabajo** de aprendizaje automático (machine learning) en **entornos de entrega continua** (CI/CD) utilizando GitHub Actions.

CML busca mejorar la eficiencia y la automatización en el desarrollo y despliegue de modelos de aprendizaje automático, integrándose de manera estrecha con las capacidades de CI/CD proporcionadas por GitHub Actions.

Configurar **GitHub Actions** implica definir un flujo de trabajo (workflow) en tu repositorio de GitHub. Un flujo de trabajo de GitHub Actions es una serie de tareas automatizadas que se ejecutan cada vez que se produce un evento específico en tu repositorio. Puedes utilizar GitHub Actions para realizar tareas como pruebas, construcción, despliegue y más.

Se utilizan para ejecutar pruebas de Python cada vez que se realiza un push en el repositorio.

Ejemplo de uso:

DVC

DVC, que significa "Data Version Control", es una herramienta de código abierto diseñada para gestionar y versionar conjuntos de datos en proyectos de ciencia de datos y aprendizaje automático (machine learning).

DVC es una herramienta valiosa para equipos que trabajan en proyectos de aprendizaje automático y ciencia de datos, proporcionando un mecanismo eficiente y estructurado para gestionar y versionar grandes conjuntos de datos y facilitar la colaboración y la reproducibilidad en estos proyectos.

PM2

PM2 es un administrador de procesos para aplicaciones Node.js. Su nombre completo es "Process Manager 2". PM2 facilita la gestión y el despliegue de aplicaciones Node.js. Es especialmente útil en entornos de producción donde es esencial garantizar la disponibilidad continua de las aplicaciones.

ENTREGA

Servidores web

Existen varios tipos de servidores web, cada uno diseñado para satisfacer diferentes necesidades y escenarios. Aquí te presento algunos de los tipos más comunes:

1. Servidores Web Tradicionales:

- **Apache HTTP Server:** Uno de los servidores web más antiguos y ampliamente utilizados. Es de código abierto y es conocido por su flexibilidad y capacidad para manejar una variedad de configuraciones.
- **Nginx:** Un servidor web y proxy inverso que se ha vuelto muy popular por su eficiencia en el manejo de conexiones simultáneas y su uso común como servidor proxy y equilibrador de carga.

2. Servidores de Aplicaciones:

- **Tomcat (Apache Tomcat):** Diseñado para ejecutar aplicaciones Java en el servidor web. Es un servidor de aplicaciones de código abierto desarrollado por la Apache Software Foundation.
- **Microsoft Internet Information Services (IIS):** Un servidor web de Microsoft que también puede actuar como servidor de aplicaciones para aplicaciones basadas en tecnologías Microsoft como ASP.NET.

3. Servidores Ligeros y Rápidos:

- **Caddy:** Un servidor web moderno y fácil de configurar que se destaca por su simplicidad y soporte integrado para HTTPS.
- **LiteSpeed:** Un servidor web de alto rendimiento que pretende superar las limitaciones de servidores web tradicionales como Apache.

4. Servidores Específicos para Tecnologías:

- **Node.js:** Mientras que Node.js en sí mismo no es un servidor web, es común utilizar frameworks como Express para construir servidores web utilizando JavaScript en el lado del servidor.

- **Django (Gunicorn):** Gunicorn es un servidor WSGI (Web Server Gateway Interface) que se utiliza comúnmente para ejecutar aplicaciones web construidas con Django.

5. Servidores Empotrados:

- Algunos lenguajes de programación, como Python (con el módulo **http.server**) o Ruby (con WEBrick), vienen con servidores web integrados que son útiles para el desarrollo y pruebas locales.

6. Servidores de Contenido Estático:

- **Caddy:** Además de sus características como servidor web, Caddy también se destaca por su facilidad de configuración para servir contenido estático.
- **Netlify:** Es un servicio que proporciona un servidor de contenido estático junto con funciones de implementación continua para sitios web estáticos.

7. Servidores Proxy y Equilibradores de Carga:

- **HAProxy:** Aunque es comúnmente utilizado como equilibrador de carga, HAProxy también puede actuar como un servidor proxy para manejar el tráfico web.
- **Traefik:** Diseñado para ser un servidor proxy y equilibrador de carga moderno, especialmente utilizado en entornos de contenedores.

Estos son solo algunos ejemplos y hay muchos otros servidores web y herramientas que pueden adaptarse a diferentes necesidades. La elección del servidor web depende de factores como los requisitos específicos de la aplicación, la tecnología utilizada, la carga esperada y las preferencias del administrador del sistema.

Tras contemplar múltiples servidores con el objetivo de desarrollar una aplicación basada en flujos de trabajo de ML en entornos de entrega continua (CI/CD) utilizando GitHub Actions, un sistema operativo Windows y queriendo servir contenido dinámico he elegido el servidor Nginx (versión Stable)

Como dependencia del entorno de ejecución Node.js he elegido Sails.js que es un framework MVC (Modelo-Vista-Controlador) y Hapi por si quiero crear API's robustas

Ambas han sido añadidas como dependencias de producción pero también podían haber sido añadidas como dependencias de desarrollo

Ya se ha comprobado que con el comando `node app.js` se puede ejecutar la aplicación y no solo eso sino que yendo tanto al puerto 300 como al puerto 1337 se pueden ver diferentes facetas de la aplicación

MI PLAN

Recomendable utilizar flask??

Recomendable utilizar React en el entorno del cliente??

En Windows que server utilizo??-->nginx

Es mejor Windows o maquina virtual??-->windows

Gunicorn, Ngnix o IIS??-->nginx

Puedo utilizar IIS con proxys como pfSense o incorporan el suyo??-->da igual

AJAX como cliente servidor??--> da igual

El paso 5 es CML de manual

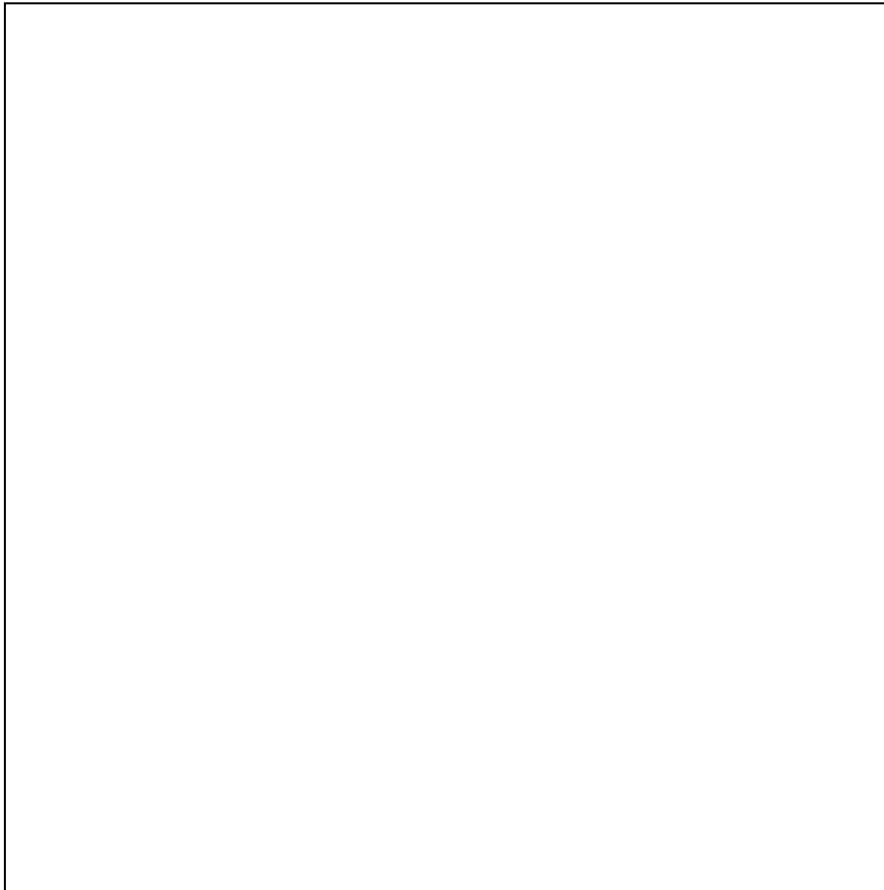
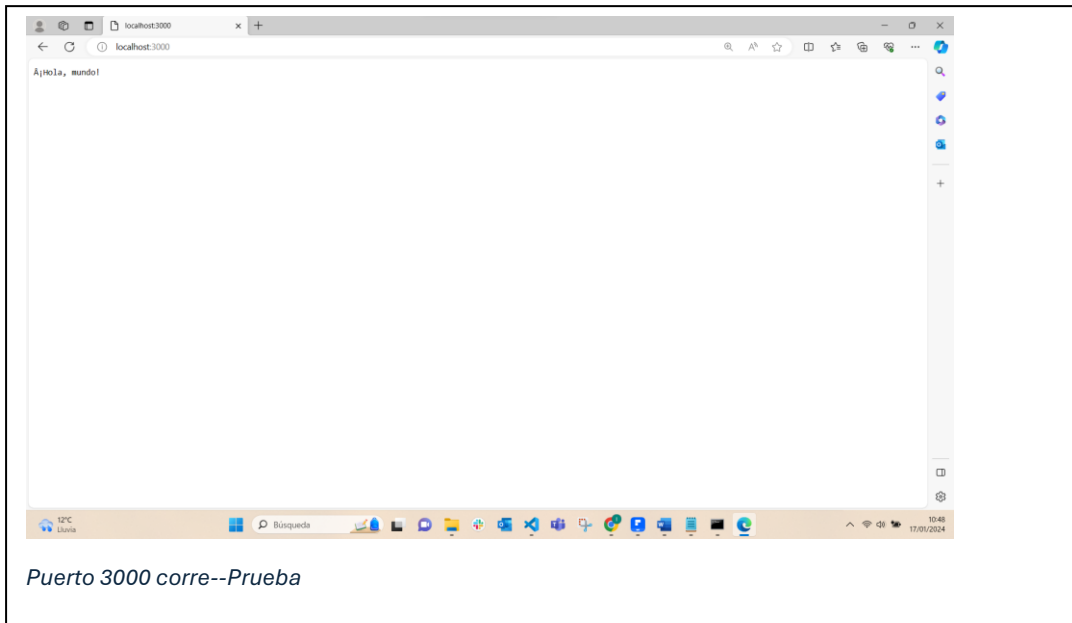
```
C:\Users\Lorenzo\Downloads>start nginx
El sistema no puede encontrar el archivo nginx.

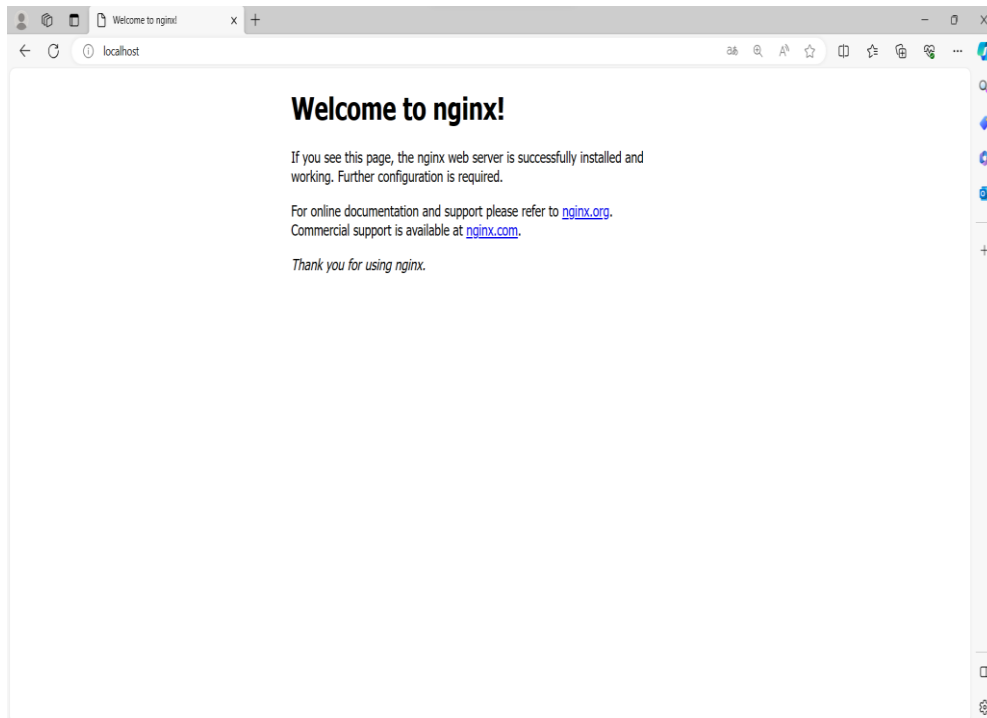
C:\Users\Lorenzo\Downloads>cd nginx-1.24.0

C:\Users\Lorenzo\Downloads\nginx-1.24.0>start nginx
C:\Users\Lorenzo\Downloads\nginx-1.24.0>start nginx
C:\Users\Lorenzo\Downloads\nginx-1.24.0>start nginx
C:\Users\Lorenzo\Downloads\nginx-1.24.0>start nginx
C:\Users\Lorenzo\Downloads\nginx-1.24.0>nginx -s stop
C:\Users\Lorenzo\Downloads\nginx-1.24.0>nginx -s stop
nginx: [error] CreateFile() "C:\Users\Lorenzo\Downloads\nginx-1.24.0/logs/nginx.pid" failed (1)
C:\Users\Lorenzo\Downloads\nginx-1.24.0>start nginx
C:\Users\Lorenzo\Downloads\nginx-1.24.0>nginx -s stop
C:\Users\Lorenzo\Downloads\nginx-1.24.0>nginx -s reload
nginx: [error] CreateFile() "C:\Users\Lorenzo\Downloads\nginx-1.24.0/logs/nginx.pid" failed (1)
C:\Users\Lorenzo\Downloads\nginx-1.24.0>start nginx
```

```
C:\Users\Lorenzo\Downloads\nginx-1.24.0>start nginx
El sistema no puede encontrar el archivo nginx.
```

iD





Elaboración de la practica

Objetivo:

Crear un flujo de trabajo automatizado para implementar y desplegar una aplicación web en un entorno de producción utilizando herramientas de entrega continua.

La elaboración se desarrolla en 5 pasos:1

I. Preparación del entorno de producción

Ya se ha comprobado que con el comando `node app.js` se puede ejecutar la aplicación y no solo eso sino que yendo tanto al puerto 3000 como al puerto 1337 se pueden ver diferentes facetas de la aplicación

Quiere **EDGE COMPUTING**

Que es un entorno de producción?

Un entorno de producción es un ambiente configurado y destinado específicamente para **ejecutar aplicaciones y servicios en un entorno real**, es decir, en condiciones operativas reales en lugar de en un entorno de prueba o desarrollo.

Entorno de producción→ aplicación interactúa con los usuarios

Entorno de desarrollo→ lugar donde la aplicación sufre pruebas

Aclaración→ En contraste con los entornos de desarrollo y prueba, donde los desarrolladores prueban y depuran aplicaciones, el entorno de producción es donde los usuarios finales interactúan con la aplicación de manera cotidiana. Cambios significativos y actualizaciones se prueban primero en entornos de desarrollo y prueba antes de implementarse en un entorno de producción para minimizar el impacto en los usuarios finales y garantizar la estabilidad del sistema.

El entorno de producción de nuestra aplicación consta de:

1. Servidor web→ se ha elegido Nginx (apartado de los servidores web)

```
❏ Símbolo del sistema
C:\Users\Lorenzo\Documents\Programacion_IV\1.Aprendizaje automatico\Practicas\Despliegue>node app.js
Servidor escuchando en el puerto 3000
^C
C:\Users\Lorenzo\Documents\Programacion_IV\1.Aprendizaje automatico\Practicas\Despliegue>node app.js
Servidor escuchando en el puerto 3000
^C
C:\Users\Lorenzo\Documents\Programacion_IV\1.Aprendizaje automatico\Practicas\Despliegue>node app.js
Servidor escuchando en el puerto 3000
debug: Warning: no session secret was set! In development mode, we'll set this for you,
debug: but session secret must be manually specified in production.
debug: To set up a session secret, add or update it in 'config/session.js':
debug: module.exports.session = { secret: 'extremely-secure-keyboard-cat' }
debug:
debug: (Or if you don't need sessions enabled, disable the hook.)
debug:
debug: For more help:
debug:   • https://sailsjs.com/config/session
debug:   • https://sailsjs.com/support
debug:
error: Server failed to start.
error: (received error: EADDRINUSE)
error:
error: Troubleshooting tips:
error:
error:   -> Is something else already running on port 3000 ?
error:
error:   -> Are you deploying on a platform that requires an explicit hostname, like OpenShift?
error:       (Try setting the 'explicitHost' config to the hostname where the server will be accessible.)
error:       (e.g. 'mydomain.com' or '183.24.244.42' )

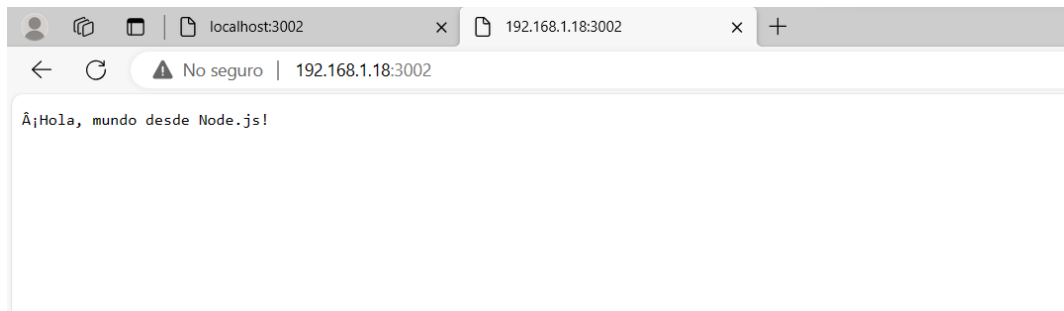
C:\Users\Lorenzo\Documents\Programacion_IV\1.Aprendizaje automatico\Practicas\Despliegue>node app.js
Servidor de Node.js escuchando en el puerto 3000
debug: Warning: no session secret was set! In development mode, we'll set this for you,
debug: but session secret must be manually specified in production.
debug: To set up a session secret, add or update it in 'config/session.js':
debug: module.exports.session = { secret: 'extremely-secure-keyboard-cat' }
debug:
debug: (Or if you don't need sessions enabled, disable the hook.)
debug:
debug: For more help:
debug:   • https://sailsjs.com/config/session
debug:   • https://sailsjs.com/support
debug:
info:
info:   .....
info:
info: Sails      <|  .....
info:
```

npm install -g nodemon→ para que cada vez que se ejecuten cambios simplemente con el código nodemon app.js se ejecute

Comando que se utiliza para saber la dirección IP del servidor Nginx

```
C:\Users\Lorenzo\Downloads\nginx-1.24.0>curl ifconfig.me
90.167.246.110
```

Con todo esto si en el navegador web buscamos <http://localhost:4000> nos muestra la pagina web del node.js que hemos configurado y que a su vez el servidor escucha



Tanto si ejecutas localhost como si ejecutas la dirección IP de tu ordenador los resultados son los mismos, ahora bien, el objetivo es poder acceder a la pagina web aún no ejecutando desde la terminal los comandos. Ahí es donde entra el servidor web Nginx que actúa como proxy. Además también se usa PM2(Process Manager), que como ya se ha mencionado facilita la gestión y despliegue de Node.js

```
Pm2 start app.js
Pm2 stop app
```

Comandos para iniciar la aplicación y pararla respectivamente para desplegar dicha aplicación sin la necesidad de utilizar comandos.

Breve resumen de donde nos encontramos; ahora mismo, tenemos nuestra aplicación app.js la cual la podemos inicializar gracias al product manager(PM2) sin necesidad de estar ejecutando un comando en la terminal. Sin embargo, dicha aplicación se encuentra en el puerto 3002(lo hemos elegido nosotros como puerto aleatorio) y queremos que escuche en el puerto 80. Ahí es donde entra el servidor Nginx que actúa como proxy.

Breve repaso de IP y DNS

Los servidores DNS(domain name server) se encargan de relacionar el dominio con la IP. Es decir, es un servidor que se encarga de transformar el dominio en palabras dado en una dirección IP.

Los ordenadores constan de una IP publica y una privada. La IP es el nombre del dispositivo dentro de la red. La IP privada es personal de cada dispositivo(0.0.0.0—255. 255. 255. rango de 8 bits), sin embargo la IP publica es la misma para todos aquellos dispositivos que estén conectados a internet, esta IP es una IP que esta protegida por el **router**(que utiliza el método **NAT** que cambia la dirección privada a la publica; DHCP para el proceso inverso) ya que se encarga de administrarle una IP privada a cada dispositivo que este conectado a internet. Es decir, si alguien es conocedor de nuestra IP publica

sabr  a qu  red nos estamos conectando y si sabe nuestra IP privada sabe cu l es nuestro dispositivo. La direcci n privada del router y la direcci n privada de tu dispositivo privado comparten la secuencia de los primeros 8 n meros y los siguientes 8 son asignados por el router, el cual siempre tiene reservado el “.”.0.1 como direcci n la 0.0 es la del wifi.

Cada direcci n IP tiene 8 bits en cada apartado y como hay 4 cada una tiene 32 bits.

Es importante recalcar el hecho de que dos ordenadores pueden tener la misma IP privada ya que sino hay un malgasto de IP lo que es imposible es que dos ordenadores conectados al mismo wifi compartan IP privada o que dos ordenadores conectados a diferentes wifi compartan IP privada

Los dominios web est n conectados a servidores y por lo tanto tambi n tienen una IP p blica. Cuando buscamos Google.com el servidor DNS transforma Google.com en la secuencia de 16 n meros que es la IP publica del dominio Google.com

```
ping google.com  
nslookup google.com
```

Comandos para ver la direcci n IP de una pagina web

Cada direcci n IP consta de un servidor y como cada dominio consta de una direcci n IP por lo tanto cada dominio consta de un servidor

En resumen, Google.com tiene una direcci n IP que se conecta a un servidor DNS el cual es el encargado de a la direcci n IP darle el nombre de Google.com

HASTA AQU  EL REPASO DE IP

Ahora que ya hemos hecho un repaso del concepto de IP. Nosotros en esta practica utilizamos el servidor web Nginx el cual consta de una IP. Paralelamente nos creamos una p gina web que tiene una direcci n IP local(localhost.””). El objetivo es poder relacionar la direcci n IP de nuestro servidor con la de nuestra pagina web, de tal manera que cada vez que busquemos nuestra pagina web con escribir la direcci n IP del servidor baste. Si queremos profesionalizar este proceso podemos comprar un dominio propio que sea el nombre (ejemplo.com) de nuestro servidor web.

Lo mas com n es asociar un subdominio propio(app.ejemplo.es) a nuestra direcci n IP.

Para la realización de esta practica no voy a utilizar un dominio web ya que supone un coste de dinero(es minimo) y nuestro objetivo es puramente académico no comercial. Sin embargo, el uso de los dominios web es altamente recomendado por:

1.Acceso más fácil

Un dominio proporciona una forma más fácil de acceder a tu aplicación en lugar de utilizar direcciones IP numéricas. Esto hace que la aplicación sea más accesible para los usuarios y facilita la memorización.

2.Configuración de SSL/TLS

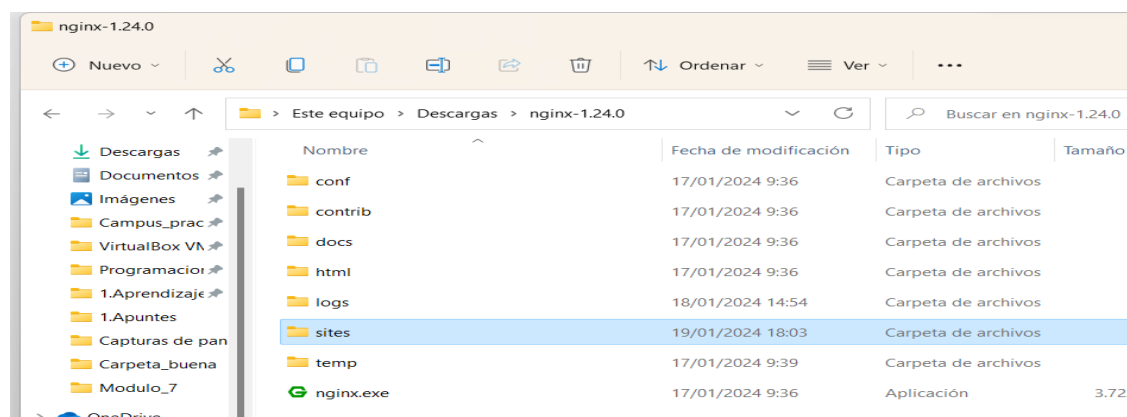
Si planeas implementar HTTPS para cifrar la comunicación entre el navegador y el servidor, tener un dominio es esencial. Los certificados SSL/TLS se emiten para específicos, y no es práctico usar certificados válidos para direcciones IP.

3.Facilita la Migración

Si alguna vez necesitas migrar tu aplicación a un nuevo servidor o cambiar la dirección IP, tener un dominio te permite hacer la transición sin cambiar la forma en que los usuarios acceden a tu aplicación.

Existen otros motivos por los cuales utilizar un dominio web tales como el SEO, la administración de subdominios o el dar un aspecto mas profesional a la pagina web.

De todas formas, como se ha mencionado, no se usará un dominio web y solamente se usará una dirección IP privada que es a la que tendría que apuntar la pagina web.(app.ejemplo.com apunta a la dirección IP 192.168.56.76 por ejemplo)



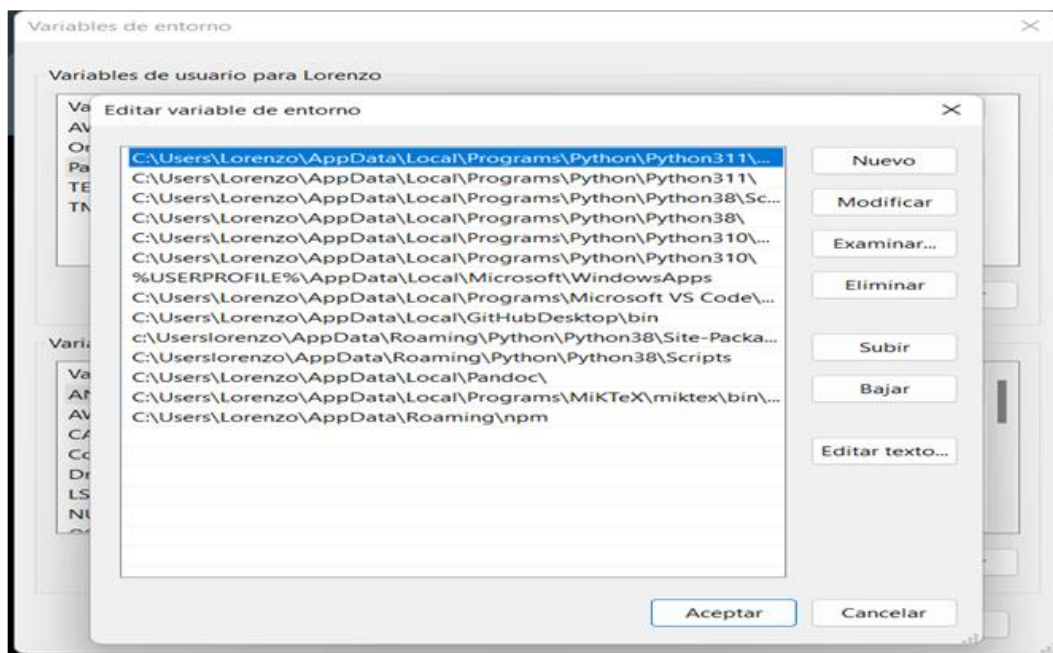
En esta pagina web: <https://pm2.keymetrics.io/docs/tutorials/pm2-nginx-production-setup> se ve un codigo genérico de la pagina web utilizando PM2 y nginx se modifica un poco tal que así:

Es necesario cambiar el archivo .conf de Nginx para poder recargar el programa correctamente.

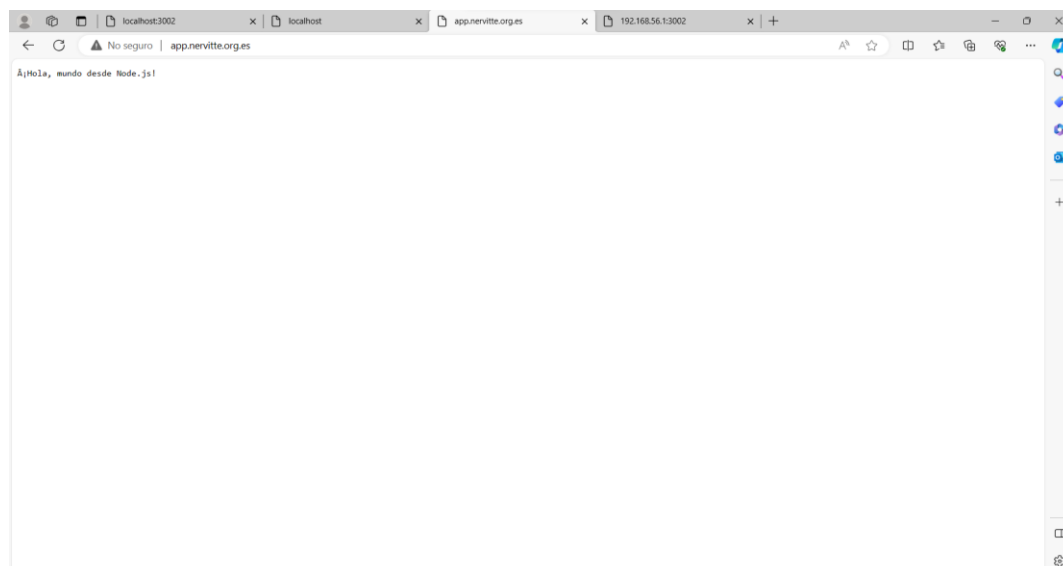
```
copy nul nombre_del_archivo
```

Creación del archivo a modificar(my_app)

Advertencia: Hacer que la línea de comandos reconozca a nginx como comando ejecutable siempre. A través de la variables de entorno:



He conseguido que desde mi directorio se conecte al localhost o la dirección IP de mi ordenador(<http://192.168.56.1>) pero, no entiendo por qué, a la hora de configurar el archivo `app.nervitte.org.es` que pertenece a mi carpeta `sites` dentro de `nginx` ignora completamente el hecho de que redirijo hacia el puerto 3002. Por lo tanto escucha a la dirección IP(localhost) pero no a la redirección al puerto 3002. No entiendo por qué ocurre.



Después de haber hecho cambios, que ya detallaré mas adelante he conseguido que la aplicación por fin se conecte con el servidor como se ve en la imagen de encima. De tal manera que el entorno de producción ya esta preparado

II. Configuración del sistema

Una variable de entorno es una variable cuyo valor está disponible para todos los procesos en un sistema operativo. Estas variables contienen información sobre el entorno del sistema y se utilizan para configurar y personalizar el comportamiento de los programas y aplicaciones. Existen variables de entorno como **path** que sirven para configurar sistemas operativos tales como Windows que son reconocibles, sin embargo, hay muchas mas variables de entorno para otras configuraciones como **MONGODB_URL** que se utiliza para configurar la URL de la base de datos. En esta practica se verán variables de entorno tanto para **configurar una base de datos** a la aplicación como para **crear credenciales de autenticación**.

Configuración de la base de datos

Configurar una URL de la base de datos se refiere a proporcionar una dirección de la base de datos del sistema que la utiliza. Las principales razones son:

- 1.Conexion remota:** En muchos casos, las bases de datos residen en servidores separados de las aplicaciones que las utilizan. Configurar una URL permite establecer una conexión remota, lo que permite que la aplicación acceda y gestione la base de datos a través de la red.
- 2.Escalabilidad:** Al configurar una URL de la base de datos, se facilita la escalabilidad del sistema. Puedes migrar la base de datos a otro servidor o cambiar su ubicación sin tener que actualizar la configuración de la aplicación en sí, siempre y cuando la URL de la base de datos permanezca constante.
- 3. Seguridad:** Al utilizar una URL de la base de datos, puedes implementar medidas de seguridad como firewalls y cifrado para proteger la comunicación entre la aplicación y la base de datos. Esto es especialmente importante cuando se trata de acceder a bases de datos a través de redes públicas o de Internet.

Existen otros motivos para configurar una base tales como la flexibilidad, la administración centralizada o la portabilidad de la aplicación,

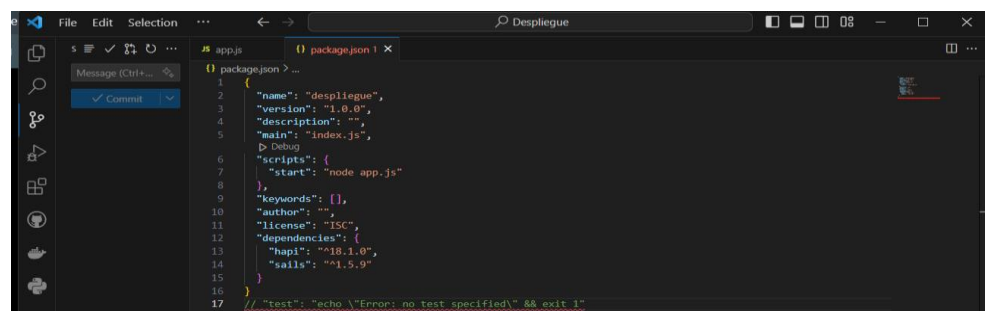
Antes de continuar:

Existe un error a la hora de ejecutar el comando

```
Npm start app.js
```

Esto se debe a que el node product manager(npm) tiene fallos en la configuración y a pesar de ejecutar la aplicación correctamente se

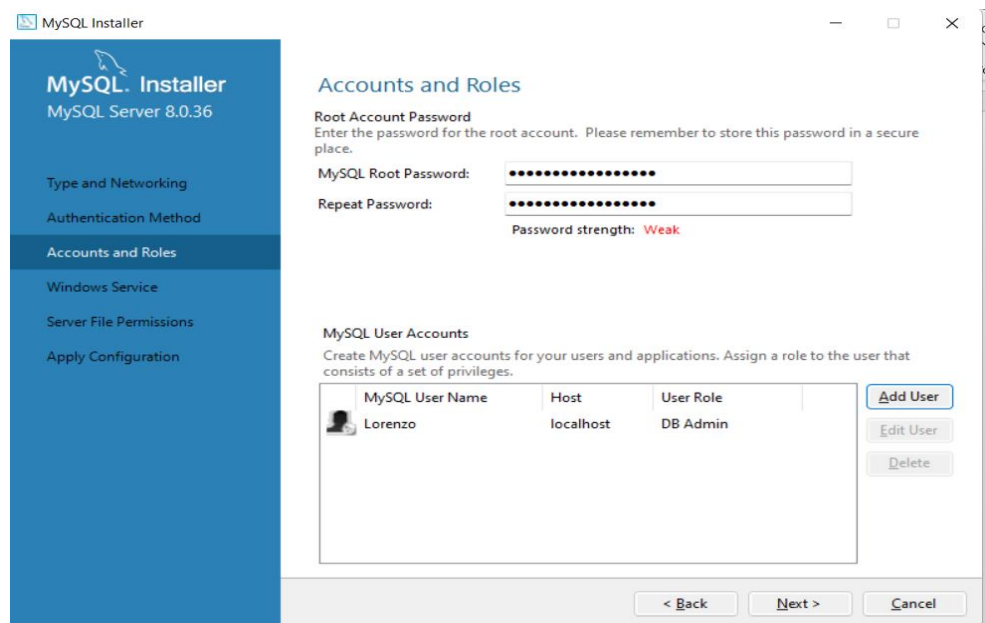
abre y se cierra la terminal constantemente haciendo que haya un proceso que sea casi un bucle indefinido. Para arreglar esto tienes que ir al package.json que consta en tu aplicación y realizar el siguiente cambio en la sección **scripts**.



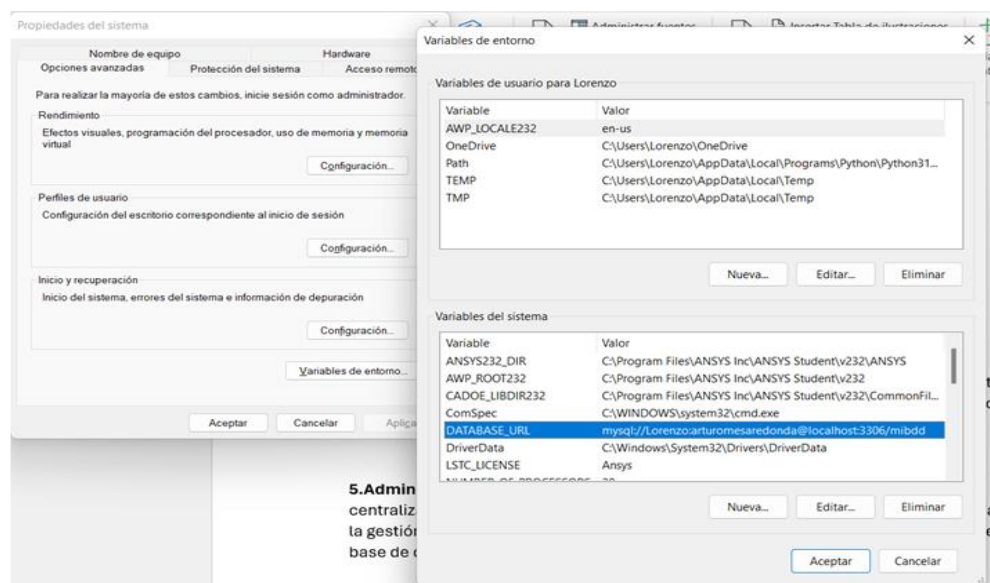
```
1 {
2   "name": "despliegue",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "start": "node app.js"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "hapi": "^18.1.0",
14    "sails": "^1.5.9"
15  }
16 }
17 // "test": "echo \"Error: no test specified\" && exit 1"
```

En esta practica se va a utilizar la base de datos de MySQL ya que es la más común en aplicaciones web. Sin embargo, para aplicaciones más robustas y de mayor complejidad recomiendo utilizar PostgreSQL.

Para ello, hay que tener instalado la aplicación de MySQL cosa que doy por hecha



Arturomesaredonda



Creación de la base de datos y configuración de la variable de entorno

```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 8.0.36 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> exit;
```

Dentro del archivo de configuración `mysql` es donde se meten los comandos que aparecen a continuación tales como **exit**;

```
CREATE DATABASE nombre_de_tu_basededatos;

CREATE USER 'tu_usuario'@'localhost' IDENTIFIED BY
'tu_contraseña';

GRANT ALL PRIVILEGES ON nombre_de_tu_basededatos.*
TO 'tu_usuario'@'localhost';

FLUSH PRIVILEGES;
```

Con estos comandos se ha ejecutado una base de datos que se la he asignado a un usuario que ya existía en la aplicación que Lorenzo, al cual le he dado todos los privilegios

```
Npm install dotenv
```

Para instalar la dependencia `dotenv` dentro del directorio de la raíz.

Además hay que crear un archivo `.env` con el directorio a la base de datos(`DATABASE_URL=mysql://usuario:contraseña@localhost:3306/nombre_de_la_base_de_datos`; en mi caso)

```
Npm install mysql2
```

Utilizare la biblioteca **mysql2** para el modelado de datos

```
C:\WINDOWS\system32\cmd.exe - node app.js
C:\Users\Lorenzo\Documents\Programacion_IV\1.Aprendizaje automatico\Practicas\Despliegue>node app.js
Servidor de Node.js escuchando en el puerto 3002
debug: Warning: no session secret was set! In development mode, we'll set this for you,
debug: but session secret must be manually specified in production.
debug: To set up a session secret, add or update it in 'config/session.js':
debug: module.exports.session = { secret: 'extremely-secure-keyboard-cat' }
debug:
debug: (Or if you don't need sessions enabled, disable the hook.)
debug:
debug: For more help:
debug:   • https://sailsjs.com/config/session
debug:   • https://sailsjs.com/support
debug:
info:
info:
info:
info: Sails
info: v1.5.9
info:
info:
info:
info:
info:
info:
info:
info:
info:
info:
info: Server lifted in 'C:\Users\Lorenzo\Documents\Programacion_IV\1.Aprendizaje automatico\Practicas\Despliegue'
info: To shut down Sails, press <CTRL> + C at any time.
info: Read more at https://sailsjs.com/support.
debug: -----
debug: :: Mon Jan 22 2024 19:49:16 GMT+0100 (hora estándar de Europa central)
debug: Environment : development
debug: Port         : 1337
debug: Local        : http://localhost:1337
debug:
Sails.js está levantado y listo para recibir solicitudes
Conexión exitosa a la base de datos MySQL
Conexión cerrada correctamente
```

Como se puede ver se conecta correctamente a la base de datos, ya que hemos configurado el archivo app.js de tal manera que si la conexión es exitosa nos lo muestre por mensaje.

De esta manera ya se ha conseguido configurar la variable de entorno DATABASE_URL tal que las credenciales de autenticación que en este caso son **Usuario:** Lorenzo y **Contraseña:** arturomesaredonda estén configuradas junto con la URL de la base de datos.

III. Gestor de dependencias

A lo largo de la practica, inconscientemente, para el correcto funcionamiento del servidor y de la aplicación se ha utilizado el gestor de dependencias **npm**, que además ya se ha explicado su uso. Por lo tanto, en este apartado únicamente se va a implementar una definición de un gestor de dependencias ya que se ha ido explicando el uso de npm.

Un gestor de dependencias es una herramienta que se utiliza para gestionar las bibliotecas y módulos que una aplicación o proyecto de software necesita para funcionar correctamente. Estas dependencias pueden incluir bibliotecas de código, frameworks, utilidades o cualquier otro componente de software que sea necesario para que la aplicación pueda ejecutarse y cumplir con sus funciones.

Los gestores de dependencias automatizan el proceso de instalación, actualización, y desinstalación de estas dependencias, simplificando así la administración de las bibliotecas utilizadas en un proyecto. Algunas de las funciones clave de un gestor de dependencias incluyen:

1.Instalación de Dependencias: Descarga e instala automáticamente las dependencias necesarias para la aplicación. Esto se realiza a través de un archivo de configuración, como el package.json en proyectos Node.js.

2.Resolución de Dependencias: Determina las versiones específicas de las bibliotecas que deben ser instaladas, asegurándose de que todas las dependencias funcionen bien juntas sin conflictos.

3.Actualización de Dependencias: Facilita la actualización de las dependencias a nuevas versiones cuando sea necesario. Algunos gestores de dependencias permiten actualizar todas las dependencias o solo aquellas que no rompan la compatibilidad con la versión actual de la aplicación.

4.Gestión de Versiones: Permite especificar las versiones específicas de las dependencias requeridas y manejar problemas relacionados con la compatibilidad de versiones.

5.Manejo de Conflictos: Resuelve automáticamente los conflictos que pueden surgir cuando varias dependencias requieren diferentes versiones de una misma biblioteca.

6.Eliminación de Dependencias: Permite desinstalar y eliminar dependencias que ya no son necesarias para la aplicación.

7.Scripts y Tareas Automatizadas: Algunos gestores de dependencias, como npm y Yarn, permiten la ejecución de scripts y tareas automatizadas relacionadas con las dependencias, lo que facilita la integración de flujos de trabajo personalizados.

En el contexto de Node.js, npm (Node Package Manager) y Yarn son dos gestores de dependencias populares.

IV. Automatización de tareas

Esta parte empezará creando un **script de construcción**

Un script de construcción, también conocido como "build script" o "script de compilación", es un conjunto de instrucciones y comandos automatizados que se utilizan para compilar, ensamblar y preparar una aplicación o proyecto de software para su ejecución o distribución.

Antes de crear dicho script build crearemos una aplicación con react con el siguiente

```
npm install react react-dom
```

comando:

```
npm install -D webpack webpack-cli webpack-dev-server babel-loader @babel/core
```

Además en nuestra aplicación se va a configurar un archivo .babelrc el cual sirve para especificar cómo Babel debe transpilar (convertir) tu código fuente JavaScript moderno

(por ejemplo, ES6/ES2015 y versiones posteriores) a una forma que sea compatible con versiones anteriores de JavaScript que puedan ser interpretadas por navegadores y entornos de ejecución que no admiten las últimas características de ECMAScript.

```
C:\Users\Lorenzo\Documents\Programacion_IV\1.Aprendizaje automatico\Practicas\Despliegue>npm run build
> despliegue@1.0.0 build
> webpack --mode production

asset bundle.js 137 KiB [emitted] [minimized] (name: main) 1 related asset
modules by path ./node_modules/react/ 6.94 KiB
  ./node_modules/react/index.js 190 bytes [built] [code generated]
  ./node_modules/react/cjs/react.production.min.js 6.75 KiB [built] [code generated]
modules by path ./node_modules/react-dom/ 130 KiB
  ./node_modules/react-dom/index.js 1.33 KiB [built] [code generated]
  ./node_modules/react-dom/cjs/react-dom.production.min.js 129 KiB [built] [code generated]
modules by path ./node_modules/scheduler/ 4.33 KiB
  ./node_modules/scheduler/index.js 198 bytes [built] [code generated]
  ./node_modules/scheduler/cjs/scheduler.production.min.js 4.14 KiB [built] [code generated]
./src/index.js 256 bytes [built] [code generated]
webpack 5.89.0 compiled successfully in 2379 ms
```

Si hemos ejecutado correctamente y hemos configurado los scripts de manera correcta podremos ejecutar el comando run build haciendo que el webpack se lance de manera correcta. De tal manera que se generarán los archivos optimizados en la carpeta dist.

A continuación, se implementarán un conjunto de pruebas automatizadas para verificar la integridad y el correcto funcionamiento de la aplicación antes de su despliegue. Para ello, voy a utilizar jest como framework de Javascript con el siguiente comando:

```
npm install --save-dev jest @testing-library/react
```

Debido a pruebas con la anterior instalación se va a proceder a esta otra: **npm install --save-dev jest-environment-jsdom**

Sin embargo esto introduce un nuevo problema de conflicto de versiones ya que Jest no puede manejar directamente los módulos ES6 (ECMAScript Modules) en este momento, ya que los archivos de prueba están utilizando declaraciones import, que son parte de la sintaxis de módulos ES6.

Para resolver este problema, puedes configurar Jest para manejar módulos ES6. Para ello se va a hacer lo siguiente: **npm install --save-dev babel-jest @babel/core @babel/preset-env**

Después de muchas interacciones con el error:

```
C:\Users\Lorenzo\Documents\Programacion_IV\1.Aprendizaje automatico\Practicas\Despliegue>npm test
> despliegue@1.0.0 test
> jest
FAIL tests /app.tests.js
  ● Test suite failed to run

    createRoot(...): Target container is not a DOM element.

    24 |
    25 | // Utiliza createRoot en lugar de ReactDOM.render
    26 | const rootInstance = createRoot(root);
    27 | rootInstance.render(<App />);
    28 |
    29 |
    at createRoot (node_modules/react-dom/cjs/react-dom.development.js:29345:11)
    at Object.createRoot [as createRoot] (node_modules/react-dom/cjs/react-dom.development.js:29800:10)
    at Object.<anonymous> reports/createRoot (node_modules/react-dom/client.js:12:10)
    at Object.<anonymous> (src/index.js:26:32)
    at Object.require (__tests__/app.tests.js:16:13)

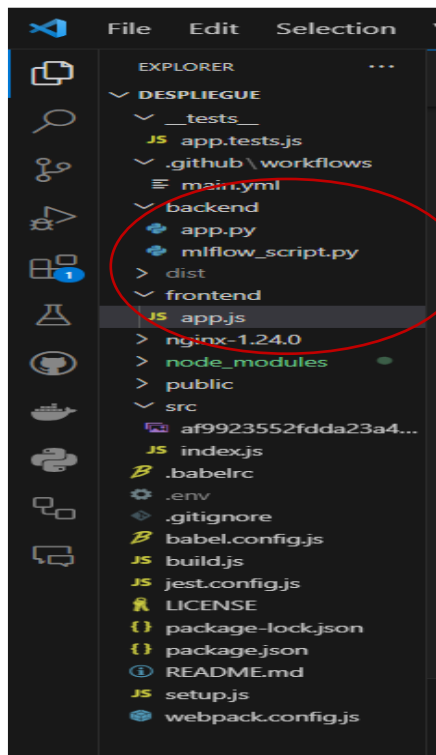
Test Suites: 1 failed, 1 total
Tests:       0 total
Snapshots:  0 total
Time:        1.254 s
Ran all test suites.

C:\Users\Lorenzo\Documents\Programacion_IV\1.Aprendizaje automatico\Practicas\Despliegue>babelrcq
```

Sigue persistiendo a pesar de cambiar la configuración babel y quedar todo registrado en mi archivo package.json:

```
"jest": {
  "testEnvironment": "jest-environment-jsdom",
  "transform": {
    "^.+\\.jsx?$": "babel-jest"
  }
}
```

Además hemos separado frontend y backend de la siguiente manera:



```
C:\Users\Lorenzo\Documents\Programacion_IV\1.Aprendizaje automatico\Practicas\Despliegue\backend>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 844-358-388
127.0.0.1 - - [24/Jan/2024 16:32:30] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [24/Jan/2024 16:32:30] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [24/Jan/2024 16:32:47] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [24/Jan/2024 16:32:47] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [24/Jan/2024 16:32:57] "GET / HTTP/1.1" 404 -
```

A la hora de desplegar la aplicación hay que añadir el siguiente código en el archivo app.js:

Además hay que tener en cuenta que para que el frontend de la aplicación(app.js) se ejecute correctamente tiene que estar el puerto del localhost:5000 (app.py) activo y en escucha. Por lo tanto se ejecutara desde el directorio correcto en la terminal el comando Python app.py que a su vez se conecta con el archivo que contiene el script de mlflow.

Desde el navegador se debe introducir el siguiente comando:

<http://127.0.0.1:5000/startMLflowScript>

Por alguna razón nos sale error a la hora de iniciar la aplicación



Con esto conseguimos comprobar que no es por el script de mlflow sino por la app.

Ahora si metemos la aplicación .js con los comandos debería conectarse correctamente a la aplicación

BIBLIOGRAFIA

<https://mlflow.org/docs/latest/introduction/index.html#core-components-of-mlflow>

<https://nginx.org/en/download.html>

[https://github.com/vercel/ai-chatbot/blob/main/app/\(chat\)/chat/%5Bid%5D/page.tsx](https://github.com/vercel/ai-chatbot/blob/main/app/(chat)/chat/%5Bid%5D/page.tsx)

<https://docs.github.com/es/actions>

<https://dvc.org/>

<https://nifi.apache.org/documentation/nifi-2.0.0-M1/html/developer-guide.html#introduction>

<https://grpc.io/>

<https://www.youtube.com/watch?v=7VAI73roXaY>

https://www.youtube.com/watch?v=Sfj--bhgm_8

<https://www.youtube.com/watch?v=m5vmqfio610>

https://www.celiagm.es/posts/servicios/nginx/instalacion_nginx/

https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/8/html/deploying_different_types_of_servers/adding-tls-encryption-to-an-nginx-web-server_setting-up-and-configuring-nginx

<https://www.redeszone.net/2016/12/02/instalar-configurar-servidor-paginas-web-nginx/>

<https://tutecnoinfor.wordpress.com/2020/11/01/configuracion-de-servidor-web-nginx-en-debian-10/>