|  |  |
| --- | --- |
|  | desarrollado |
|  | avance |
|  | Por terminar |

Situación  
=========

Se requiere implementar la funcionalidad SUMAR mediante un servicio REST, que consiste en:

1. Recibir sólo dos números y sumarlos. (ejemplo: http:///xxxxxxxx:yy/zzzzzz/sumar/10/20 )  
2. Almacenar los números y el resultado en una base de datos.  
3. Devolver un json como respuesta.  
   
Arquitectura Solicitada  
=======================

1. 01 nginx como puerta de entrada (reverse proxy)  
2. 01 backend, en el lenguaje que tú escojas. (sug. java,node)  
3. 01 base de datos, la que tú escojas (sug. postgres)  
4. Contenedores (sug. Docker, Podman)  
5. Una librería de log (log4j, slf4j, otra dependiendo del lenguaje escogido)  
   
Problemática  
============

1. Generar un bashero/yml que realice la compilación del backend de forma automática, ejecutando los test unitarios.  
2. Generar un DockerFile, que permita construir el build de una imagen de Nginx como reverse-proxy.  
3. Generar un DockerFile, que permita construir el build de una imagen con en backend.  
4. Generar un DockerFile, que permita construir el build de una imagen personalizada con la base de datos.  
5. Generar un bashero/yml que realice la construcción automática de las imágenes mediante los archivos DockerFile respectivos.  
6. Generar un bashero/yml que permita ejecutar las imágenes de Nginx y el Backend conectando   
    a) Nginx ---> Backend ----> Base Datos  
7. Continuous Integration  : Realizar el despliegue mediante un toolchain de ALM por ejemplo Github/Gitlab/Bitbucket + Jenkins/Bamboo + Sonarqube  
8. Continuous Delivery     : Implementar Docker Registry en local o Cloud Provider (AWS, Azure, GCP, Ibm Cloud, Ali Cloud) para las imágenes de docker, Artifactory, Nexus Sonatype  
8. Continuous Deployment   : Realizar un Pipeline (Jenkins, vía código) / Plan (vía código \*Bamboo Specs\*) para el deploy automático  
9. ChatOps - Al construir correctamente o presentarse error el pipeline/plan debe enviar un mensaje a un grupo de Telegram/Slack creado para informar sobre los builds del proyecto. Al momento de la prueba agregaremos un número/email para comprobar la configuración.  
10. AgileOps: Integrar el stack ELK o Graylog+Grafana a la solución, mostrando un "Visualization"/"Dashboard" con la cantidad de peticiones por minuto.  
11. SRE: Integrar una tool de monitoreo de contenedores (por ejemplo: Portainer) a la solución.

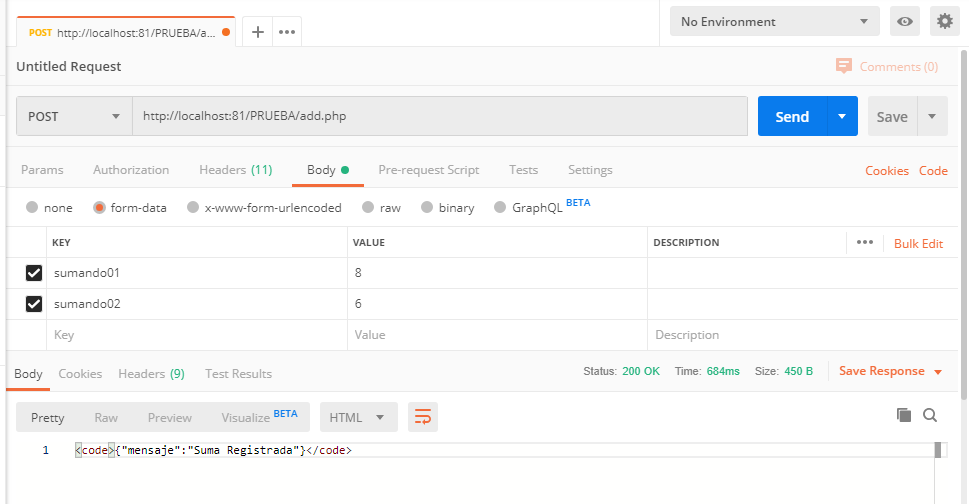
 Consideraciones  
===============

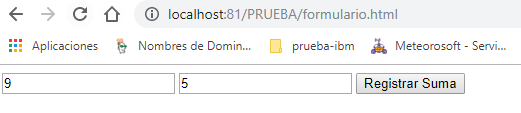
1. Subir los archivos solicitados a Github/Gitlab/Bitbucket en un repositorio en modo público, de manera que todos los archivos puedan ser descargados para su revisión.  
2. Considerar al menos un test unitario para el backend.  
3. Considerar el build automático del backend mediante un gestor de dependencias (maven, gradle, etc...)  
4. Para la base de datos, considerar solo una tabla simple (sumando01, sumando02, resultado)  
5. Considerar la cláusula HEALTCHECK dentro del Dockerfile del backend  
   
Sugerencias  
===========

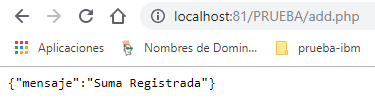
1. Puedes usar Springboot o Micronaut.io, con los cuales tendrás el backend en custión de minutos.  
2. En caso algún punto te resulte complicado puedes avanzar con los demás.  
3. El Backend debe poder ser multiréplicas, ten cuidado de no exponer el puerto como external.

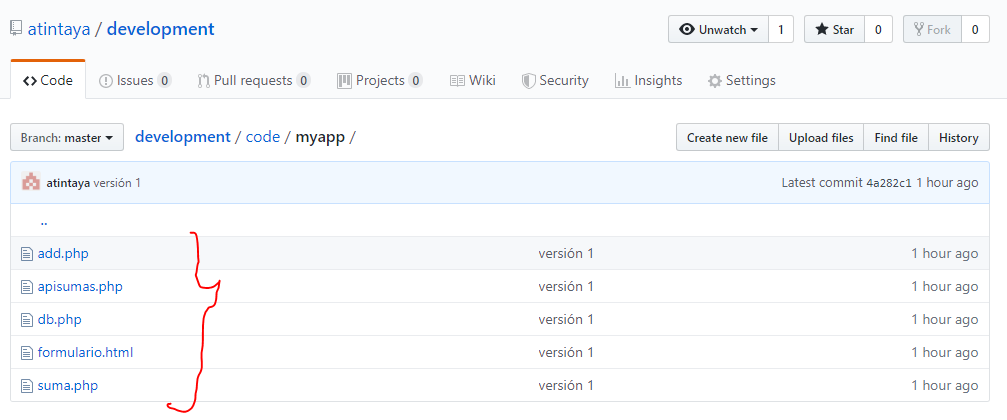
**Desarrollo del Reto**

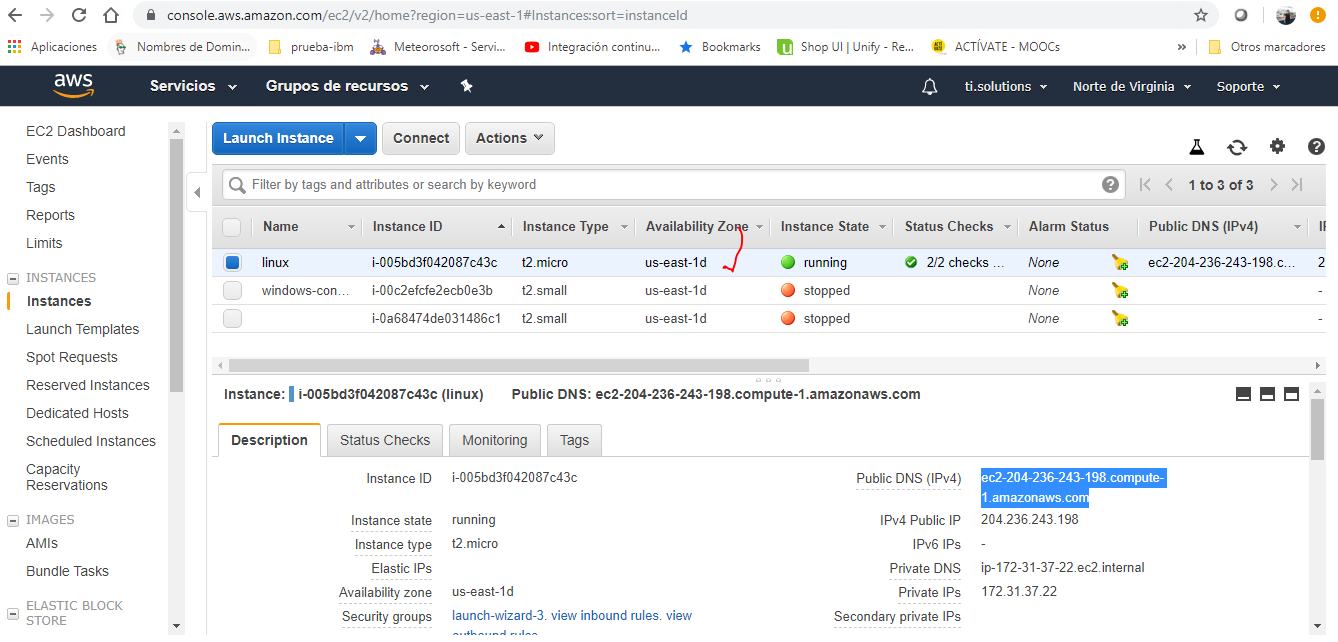
**API suma**

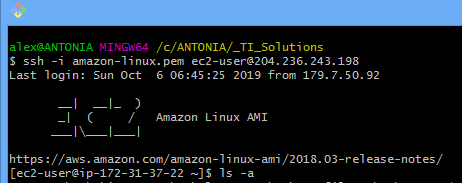


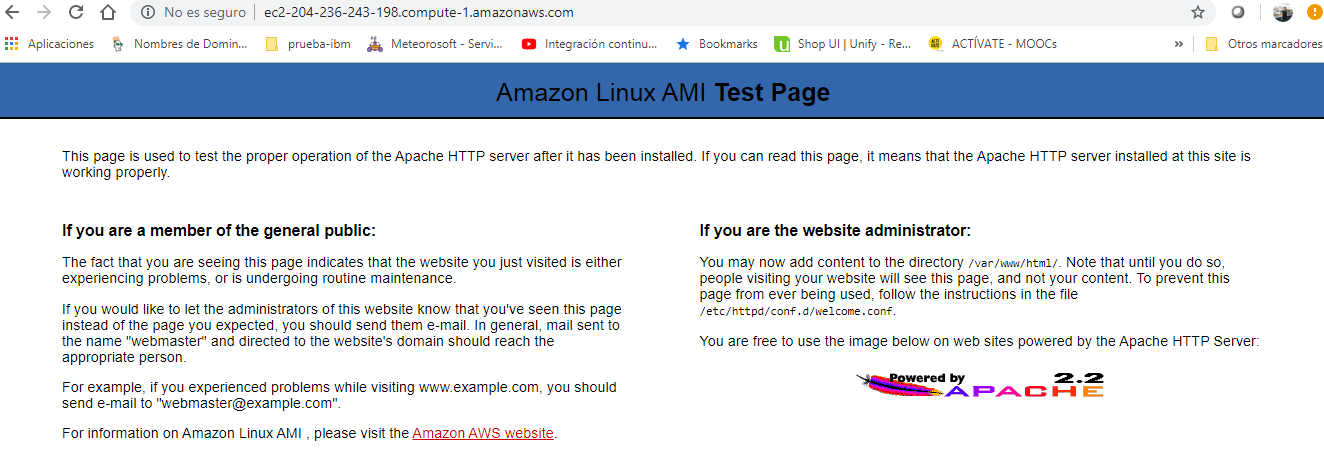








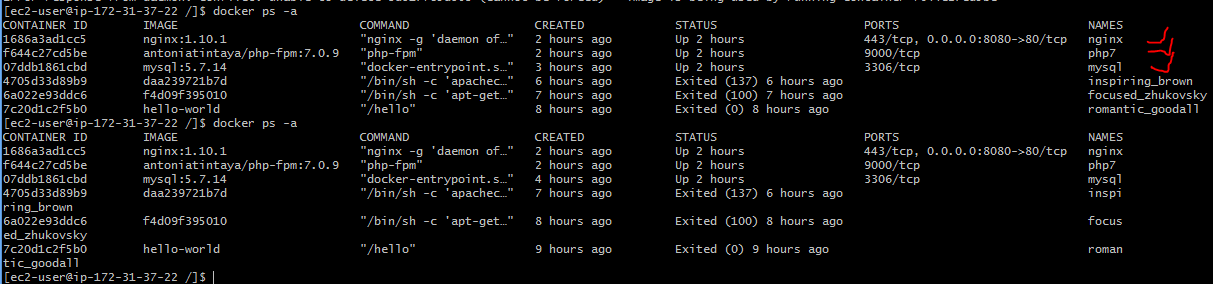




Imágenes



Contenedores



**Contenedor mysql**

docker run -d --name mysql -e MYSQL\_ROOT\_PASSWORD=password -e MYSQL\_DATABASE=dbprueba -v /Users/Antonia/development/mysql/data:/var/lib/mysql mysql:5.7.14

**script db**

use dbprueba;

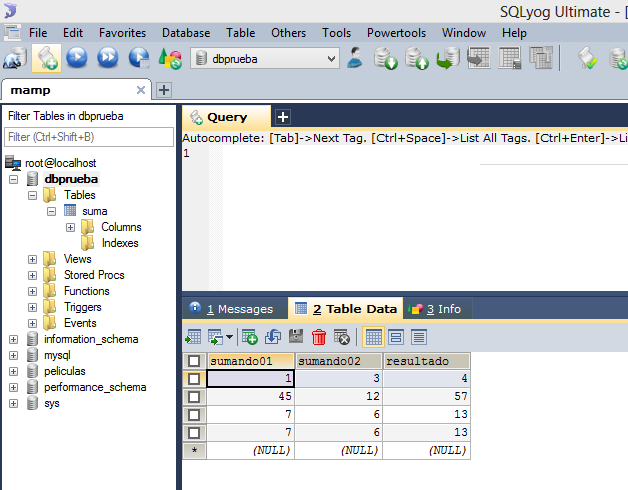
create table `suma` (

`sumando01` int (11),

`sumando02` int (11),

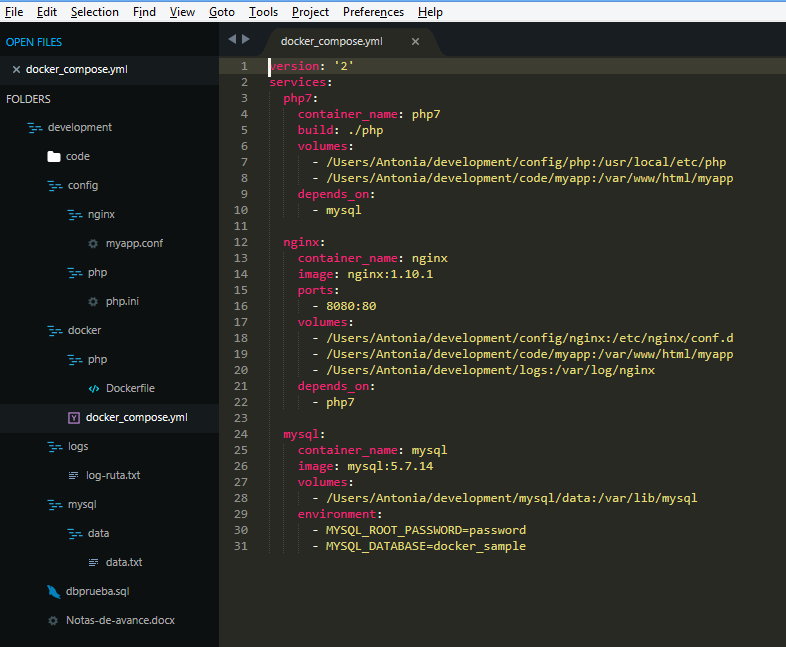
`resultado` int (11)

);



**Estructura del proyecto: descargarlo de:**

<https://github.com/atintaya/development>



**Dificultad con la instalación de**

Docker-engine

Docker-compose