Integrantes

Ramirez Vásquez, Jonathan  
Aquino Quiñones, Luis Alberto  
Luque Goycochea, Luis Alberto   
Quintana Cubas, Lenin Alexander

Docker y Kubernetes AVANZADO

Proyecto Final del Curso

Despliegue de Backend en Kubernetes para Neuromotion

Contenido

[Proyecto Curso Docker y Kubernetes 2](#_Toc202043995)

[Objetivo: 2](#_Toc202043996)

[Planteamiento: 2](#_Toc202043997)

[Entregables del Proyecto Final 2](#_Toc202043998)

[1. Código Fuente 2](#_Toc202043999)

[1.1. Repositorios de Microservicio: 2](#_Toc202044000)

[1.2. Repositorios de Front End (Angular): 4](#_Toc202044001)

[2. Artefactos Contenerizados 5](#_Toc202044002)

[2.1. Imágenes Docker: 5](#_Toc202044003)

[3. Configuración para MiniKube (Kubernetes) 9](#_Toc202044004)

[3.1. Archivos YAML: 9](#_Toc202044005)

[4. Ingress. 13](#_Toc202044006)

[5. Documentación 14](#_Toc202044007)

[5.1. Manual de Despliegue: 14](#_Toc202044008)

[5.2. Diagrama de Arquitectura: 14](#_Toc202044009)

[6. Resultados 15](#_Toc202044010)

[6.1. Validación del Despliegue: 15](#_Toc202044011)

# Proyecto Curso Docker y Kubernetes

## Objetivo:

Desplegar el backend desarrollado en Spring Boot por la empresa Neuromotion en un entorno Kubernetes, garantizando la alta disponibilidad y escalabilidad del servicio. Se busca realizar este despliegue localmente en Minikube, optimizando la administración y el rendimiento de la infraestructura.

## Planteamiento:

El software de Neuromotion gestiona usuarios, doctores y citas médicas, y su desarrollo ya está finalizado en Spring Boot. Con el fin de mejorar la disponibilidad y asegurar que el backend pueda manejar altos volúmenes de solicitudes sin interrupciones, se ha decidido desplegarlo en un entorno de orquestación de contenedores como Kubernetes. La implementación en Minikube permitirá al cliente ejecutar el backend en su máquina local. Este proyecto abordará la configuración del entorno Kubernetes, la creación de manifiestos YAML para los recursos y las mejores prácticas de despliegue.

## Entregables del Proyecto Final

Los entregables se organizan en categorías que abarcan desde el código fuente hasta la documentación y los artefactos de despliegue.

Se creó el siguiente repositorio Git para todos los entregables del presente proyecto: <https://github.com/laquinoq/medicall.git>

### Código Fuente

### Repositorios de Microservicio:

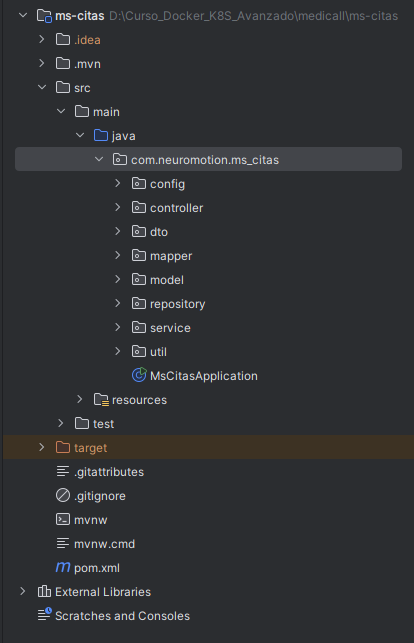
* Código fuente del Micro servicio:

ms-citas:

<https://github.com/laquinoq/medicall/tree/main/ms-citas>

* Estructura del proyecto conforme a las mejores prácticas (separación en capas: controlador, servicio, repositorio, entidades, etc.).

**ms-citas**



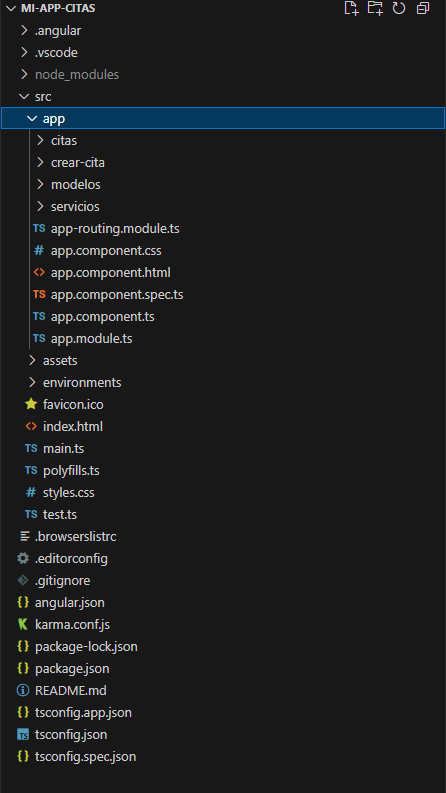
### Repositorios de Front End (Angular):

* Código fuente del Front End en Angular:

mi-app-citas:

<https://github.com/laquinoq/medicall/tree/main/mi-app-citas>

* Estructura del proyecto conforme a las mejores prácticas (separación en capas: modelo, servicio, componentes, interfaces, etc.).



### Artefactos Contenerizados

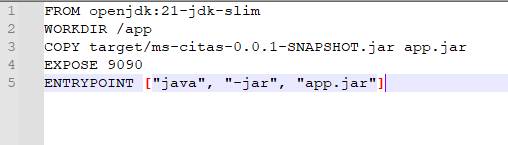
### Imágenes Docker:

* Imagen Docker del microservicio, construidas y publicadas en un repositorio como Docker Hub.

<https://hub.docker.com/repository/docker/jonathan0284/trabajo-final/general>

Dockerfile ms-citas(Microservicio Spring Boot)

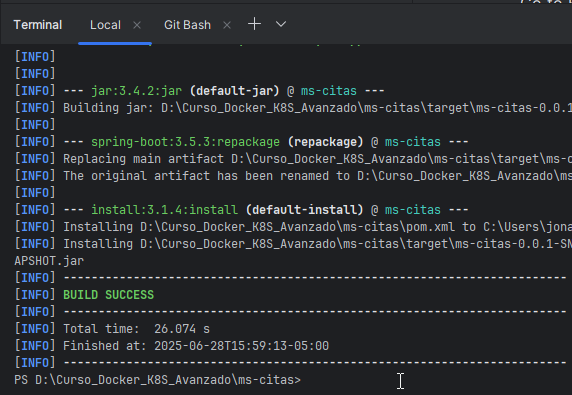
<https://github.com/laquinoq/medicall/blob/main/ms-citas/Dockerfile>



Comando para compilar el microservicio

===============================

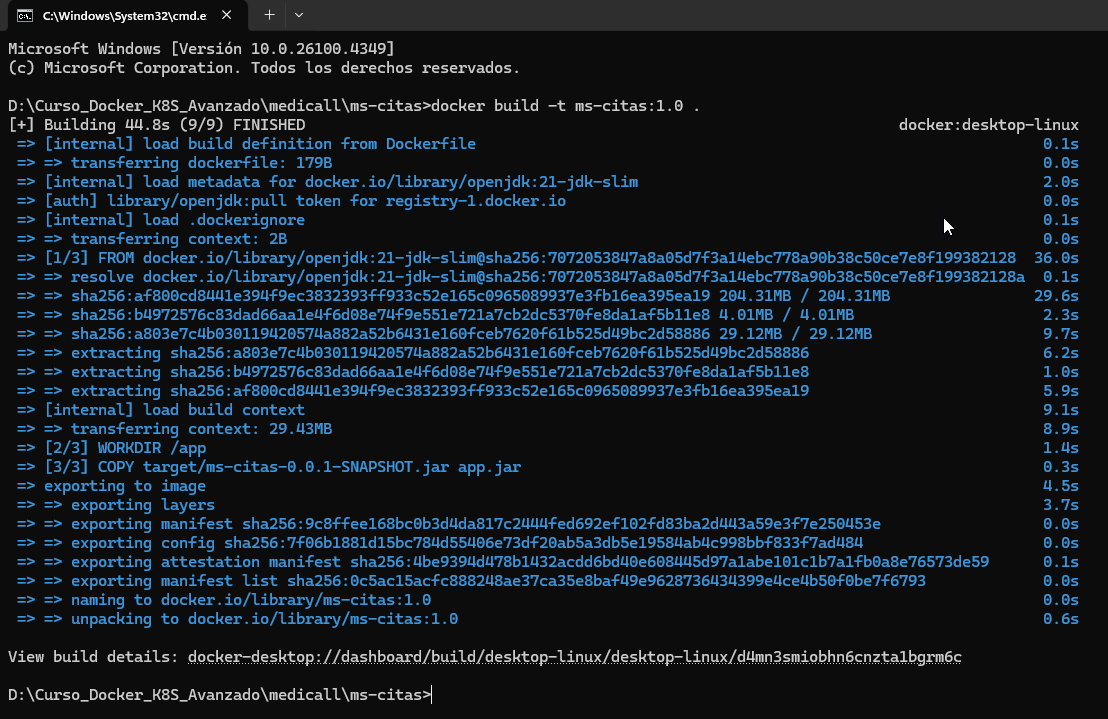
.\mvnw clean install



Crear imagen Docker

=====================

docker build -t ms-citas:2.0 .

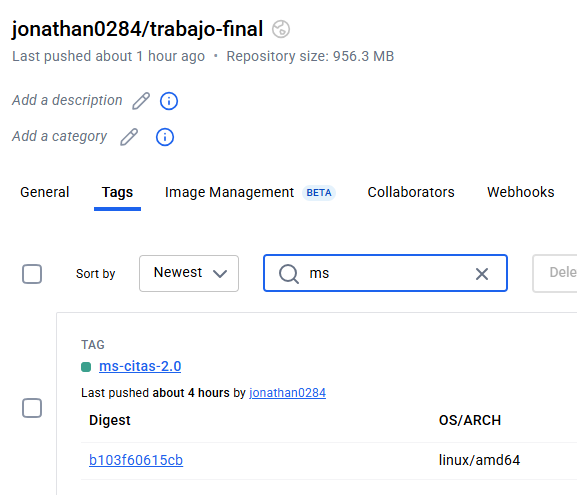


Subir imagen a Docker HUB

======================

docker tag ms-citas:2.0 jonathan0284/trabajo-final:ms-citas-2.0

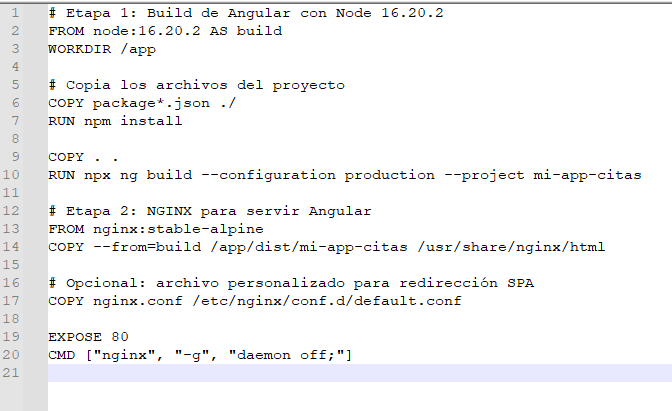
docker push jonathan0284/trabajo-final:ms-citas-2.0



* Imagen Docker de la aplicación angular, construidas y publicadas en un repositorio como Docker Hub.

Dockerfile mi-app-citas (Angular)

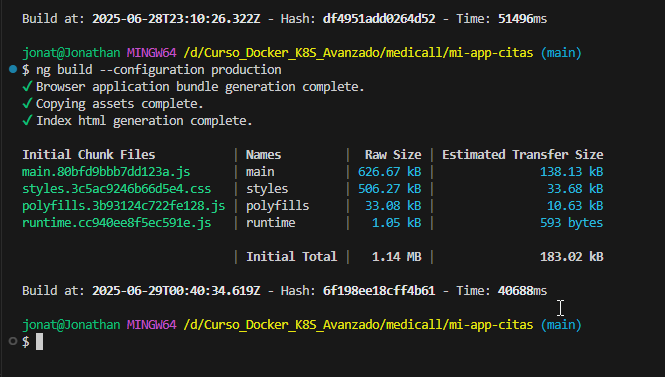
<https://github.com/laquinoq/medicall/blob/main/mi-app-citas/Dockerfile>



Comando para compilar el microservicio

===============================

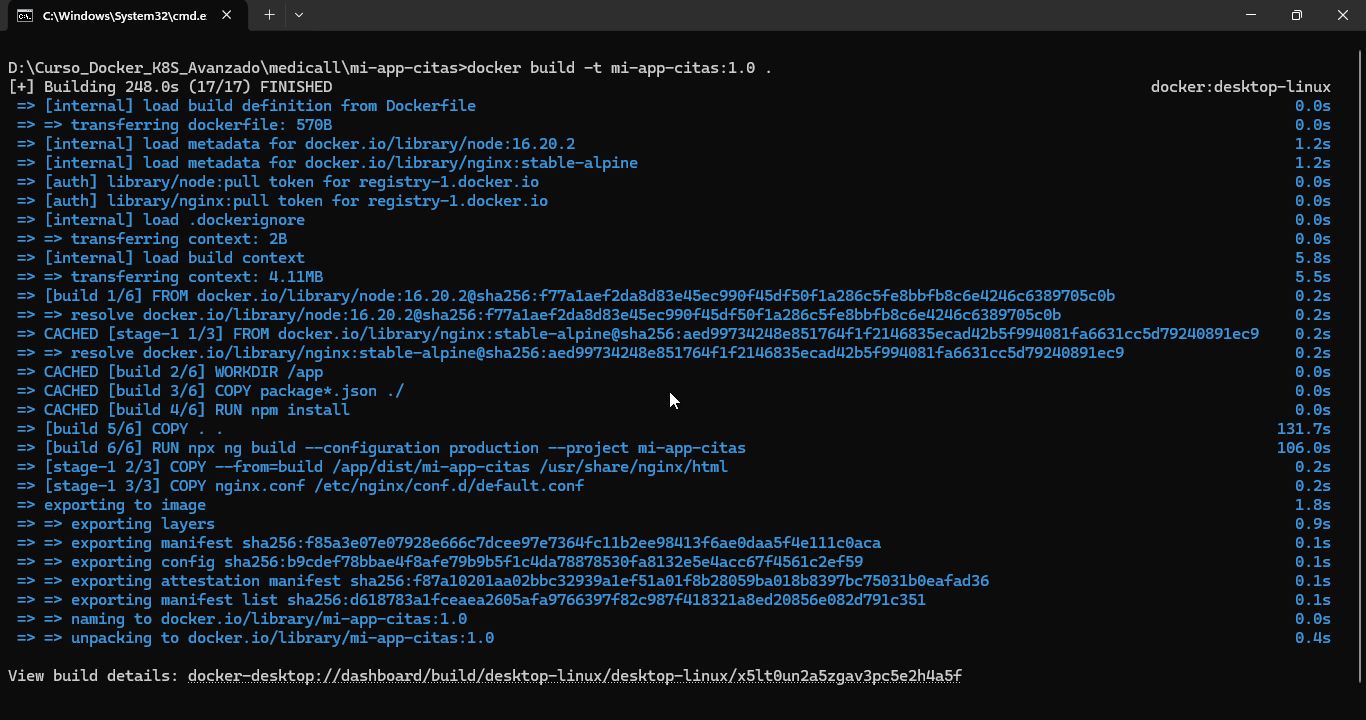
ng build --configuration production



Crear imagen Docker

=====================

docker build -t mi-app-citas:1.0 .

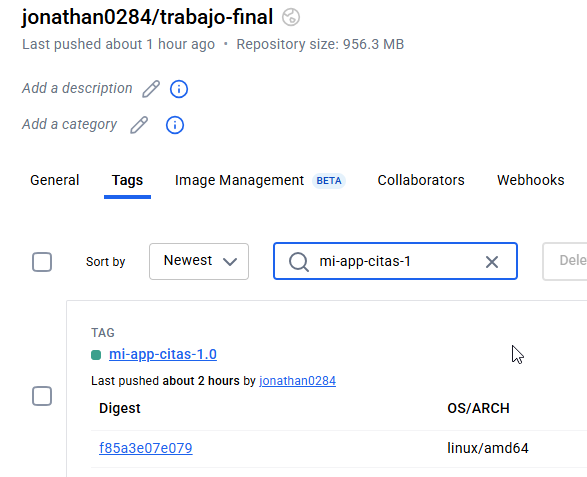


Subir imagen a Docker HUB

======================

docker tag mi-app-citas:1.0 jonathan0284/trabajo-final:mi-app-citas-1.0

docker push jonathan0284/trabajo-final:mi-app-citas-1.0



### Configuración para MiniKube (Kubernetes)

### Archivos YAML:

* **Deployments:** Configuración de pods y réplicas

**Microservicio: ms-citas**

**Archivo:** spring-deployment.yaml

<https://github.com/laquinoq/medicall/blob/main/ms-citas/yaml-files/spring-deployment.yaml>

****

**App Angular: mi-app-citas**

**Archivo:** angular-deployment.yaml

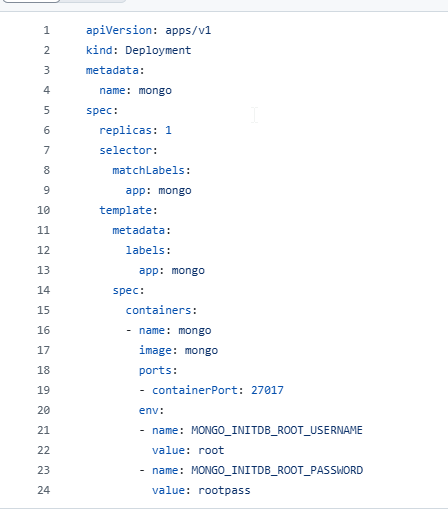
<https://github.com/laquinoq/medicall/blob/main/mi-app-citas/yaml-files/angular-deployment.yaml>

****

**Mongo DB: mongo**

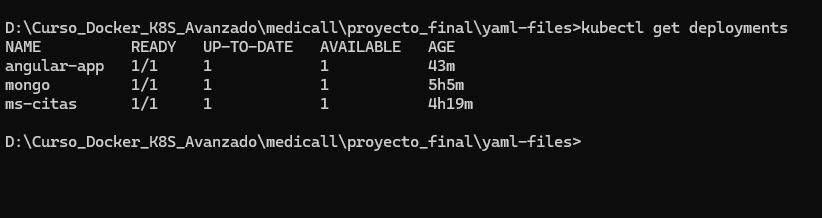
Archivo: mongo-deployment.yaml

<https://github.com/laquinoq/medicall/blob/main/mongo-db/yaml-files/mongo-deployment.yaml>

****

**Listado de despliegues**

**kubectl get deployments**

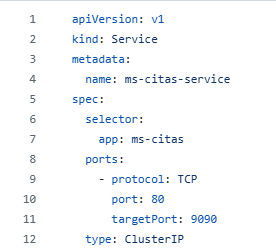
****

* **Services:** Definición de servicios ClusterIP

**Microservicio: ms-citas**

**Archivo:** spring-service.yaml

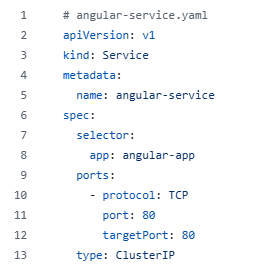
<https://github.com/laquinoq/medicall/blob/main/ms-citas/yaml-files/spring-service.yaml>

****

**App Angular: mi-app-citas**

**Archivo:** angular-service.yaml

<https://github.com/laquinoq/medicall/blob/main/mi-app-citas/yaml-files/angular-service.yaml>

****

**Mongo DB: mongo**

Archivo: mongo-service.yaml

<https://github.com/laquinoq/medicall/blob/main/mongo-db/yaml-files/mongo-service.yaml>

****

### Ingress.

* **Ingress:** Configuración de rutas basadas en contexto para exponer las APIs externamente

**Archivo: ms-citas-ingress.yaml**

<https://github.com/laquinoq/medicall/blob/main/k8s/ms-citas-ingress.yaml>

Descripción general del archivo Ingress: Este manifiesto enruta el tráfico de la siguiente manera:

🌐 Las solicitudes a http://miapp.local/ van al servicio Angular

📡 Las solicitudes a http://miapp.local/api van al servicio Spring Boot (ms-citas-service)

Ambos servicios deben estar dentro del clúster de Kubernetes (tipo ClusterIP) y escuchando en el puerto 80.



### Documentación

### Manual de Despliegue:

* Pasos detallados para:

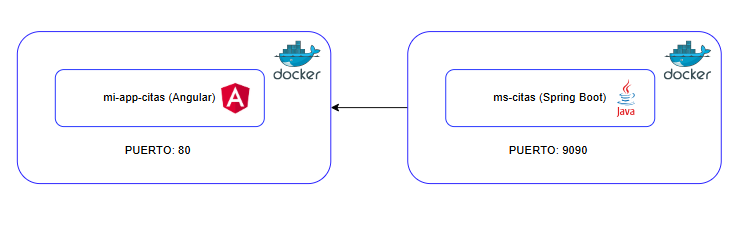
▪ Contenerizar y construir las imágenes:

▪ Desplegar en Kubernetes utilizando los manifiestos YAML.

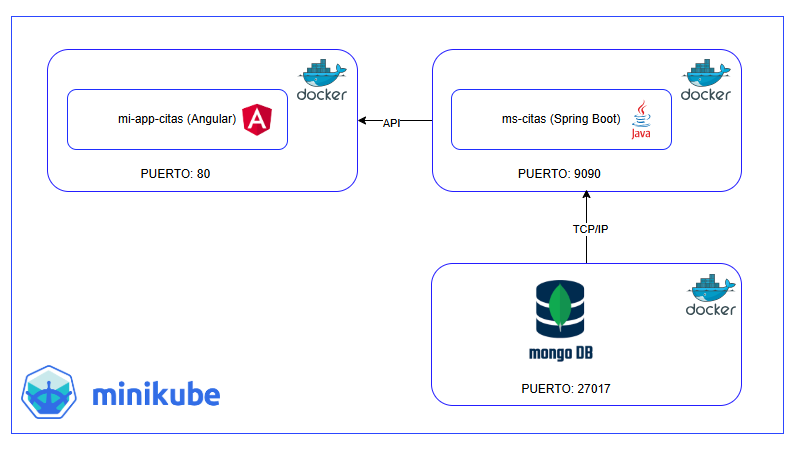
### Diagrama de Arquitectura:

* Representación gráfica de la solución:

▪ Estructura de componentes:



▪ Despliegue en Kubernetes (MiniKube):

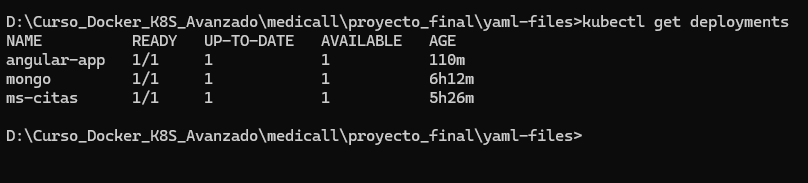


### Resultados

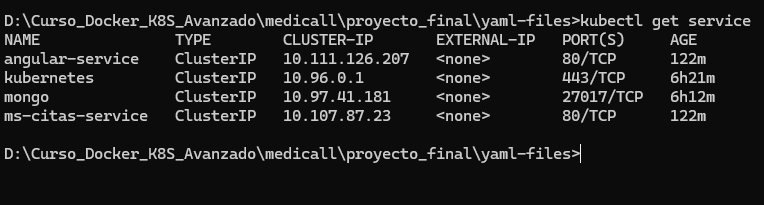
### Validación del Despliegue:

* Evidencia del despliegue exitoso en Kubernetes (Minikube) (por ejemplo, capturas de pantalla o logs que muestren los pods corriendo).

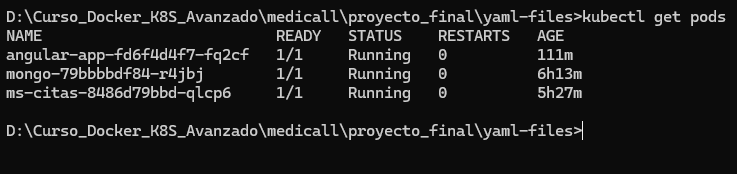
kubectl get deployments



kubectl get service



kubectl get pods



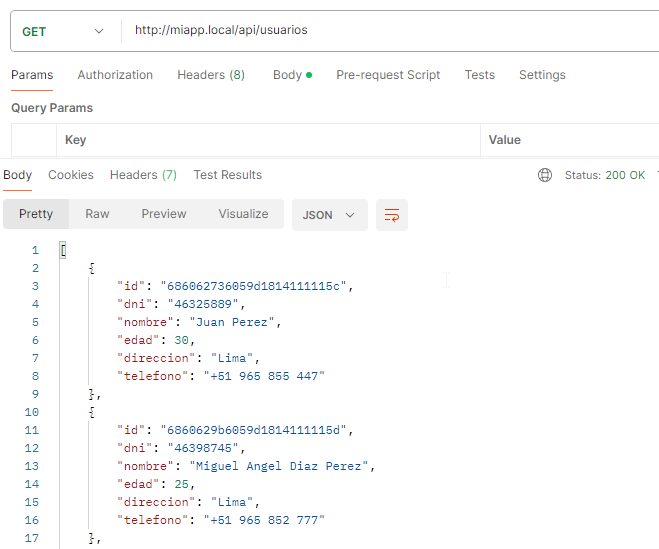
* Pruebas realizadas en los servicios mediante Postman

**Video DEMO:**

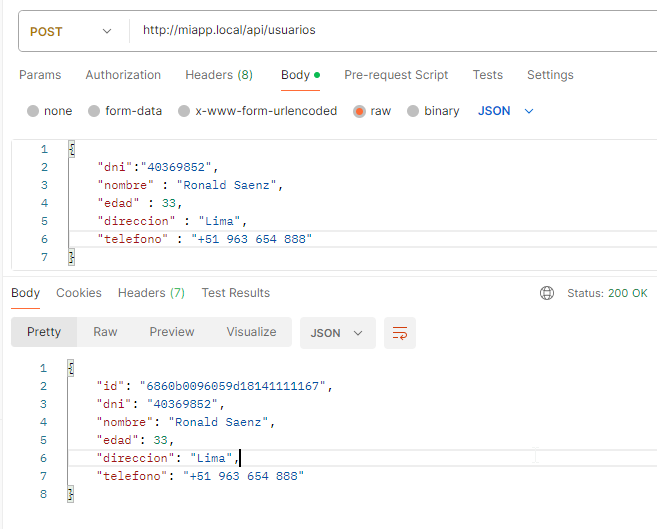
<https://www.youtube.com/watch?v=gLRaoOBWsZ8>

**Controller Usuarios**

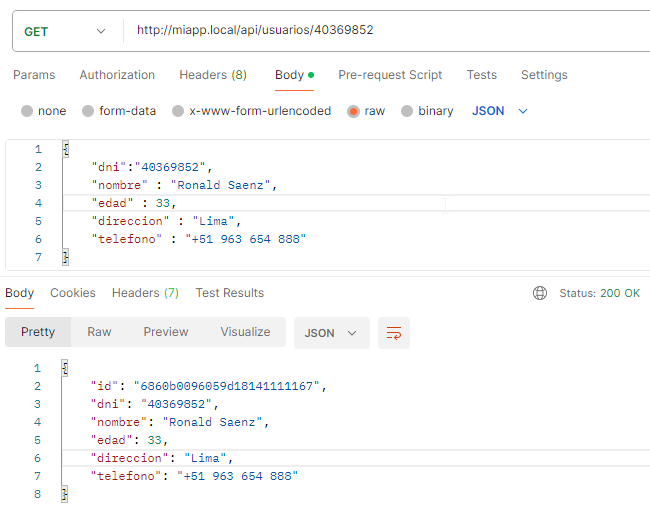
**Method GET:** Obtener todos los usuarios



**Method POST:** Registar Usuario

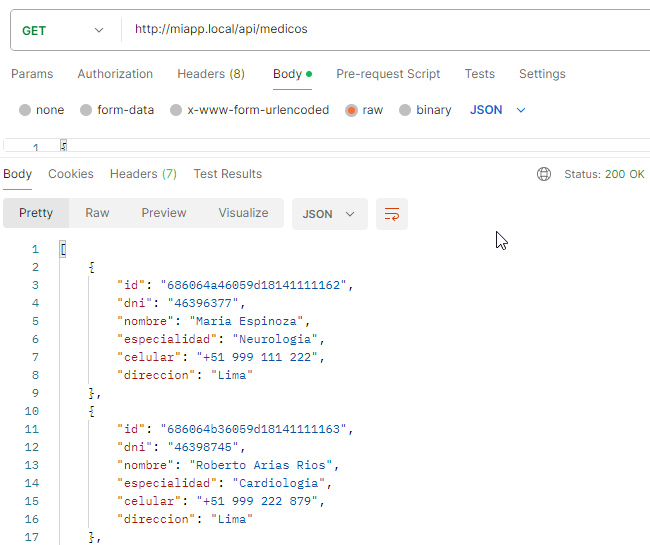


**Method GET:** Buscar Usuario por DNI

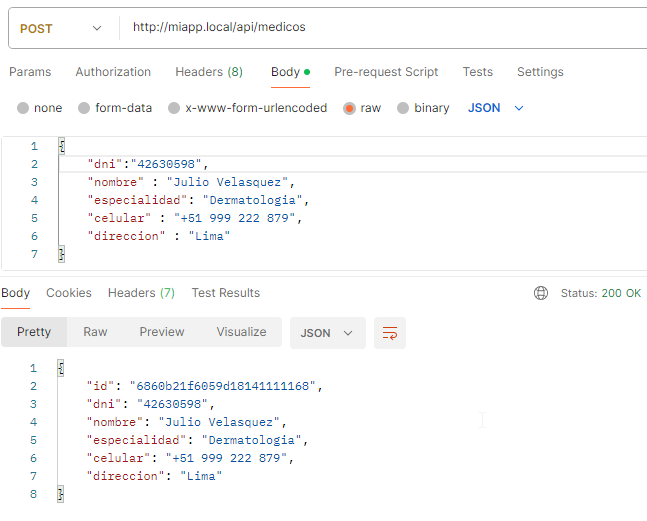


* **Controller Medicos**

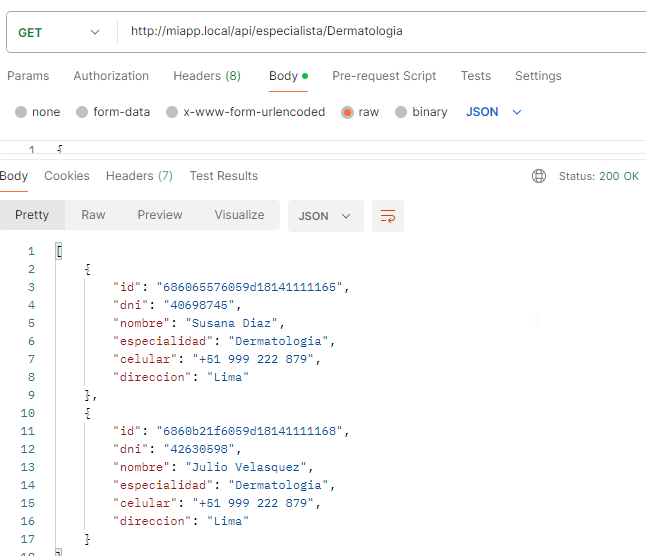
**Method GET:** Obtener todos los medicos



**Method POST:** Registar medico

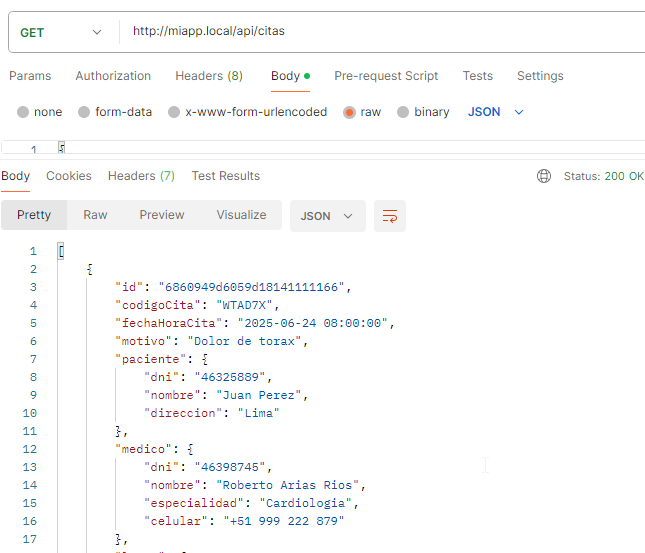


**Method GET:** Obtener médicos por especialidad



* **Controller Citas**

**Method GET:** Obtener todas las citas



**Method POST:** Registar cita

