



DOCKER Y KUBERNETES AVANZADO

Proyecto Final del Curso

Despliegue de Backend en Kubernetes para
Neuromotion

Integrantes

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

Contenido

Proyecto Curso Docker y Kubernetes.....	2
Objetivo:.....	2
Planteamiento:.....	2
Entregables del Proyecto Final.....	2
1. Código Fuente.....	2
1.1. Repositorios de Microservicio:.....	2
1.2. Repositorios de Front End (Angular):.....	4
2. Artefactos Contenerizados	5
2.1. Imágenes Docker:.....	5
3. Configuración para MiniKube (Kubernetes)	9
3.1. Archivos YAML:.....	9
4. Ingress.	13
5. Documentación.....	14
5.1. Manual de Despliegue:.....	14
5.2. Diagrama de Arquitectura:.....	14
6. Resultados	15
6.1. Validación del Despliegue:	15

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>

1. Código Fuente

1.1. Repositorios de Microservicio:

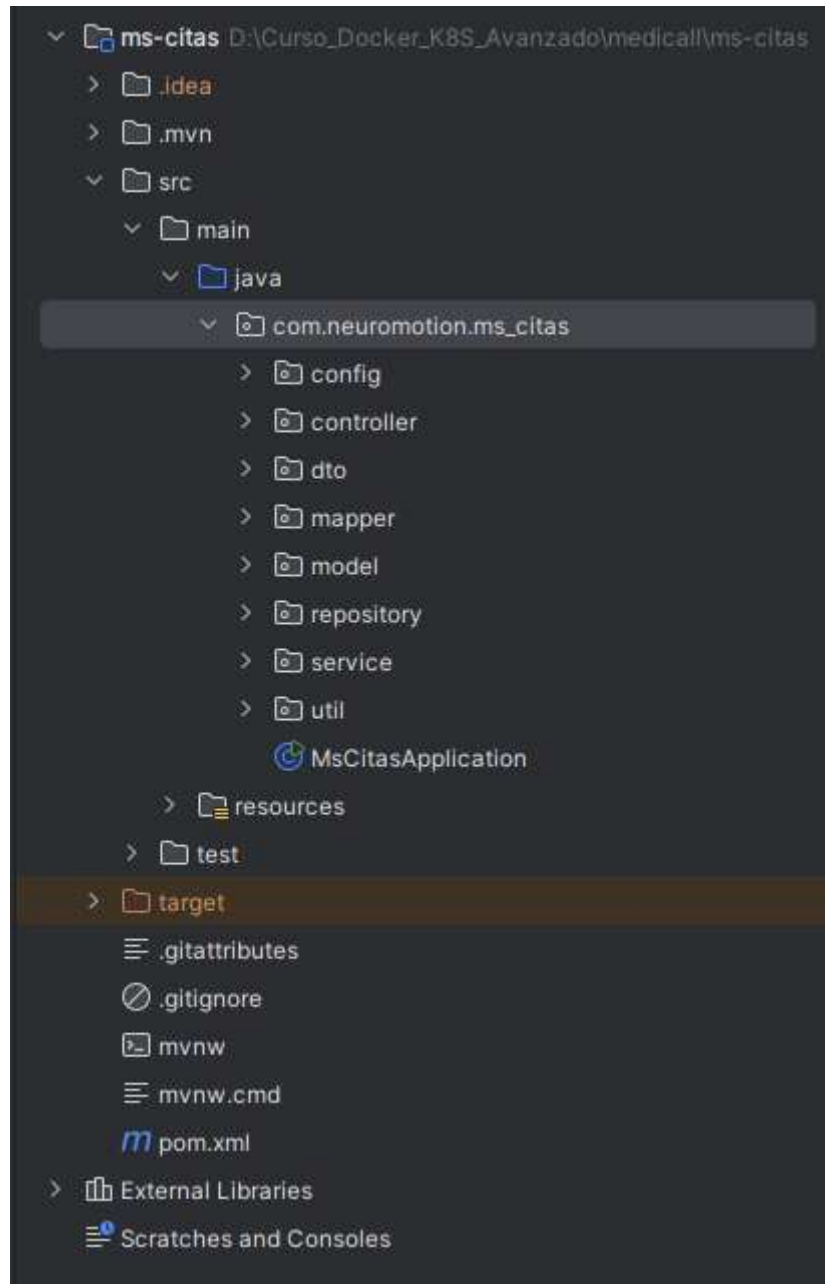
- Código fuente del Micro servicio:

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



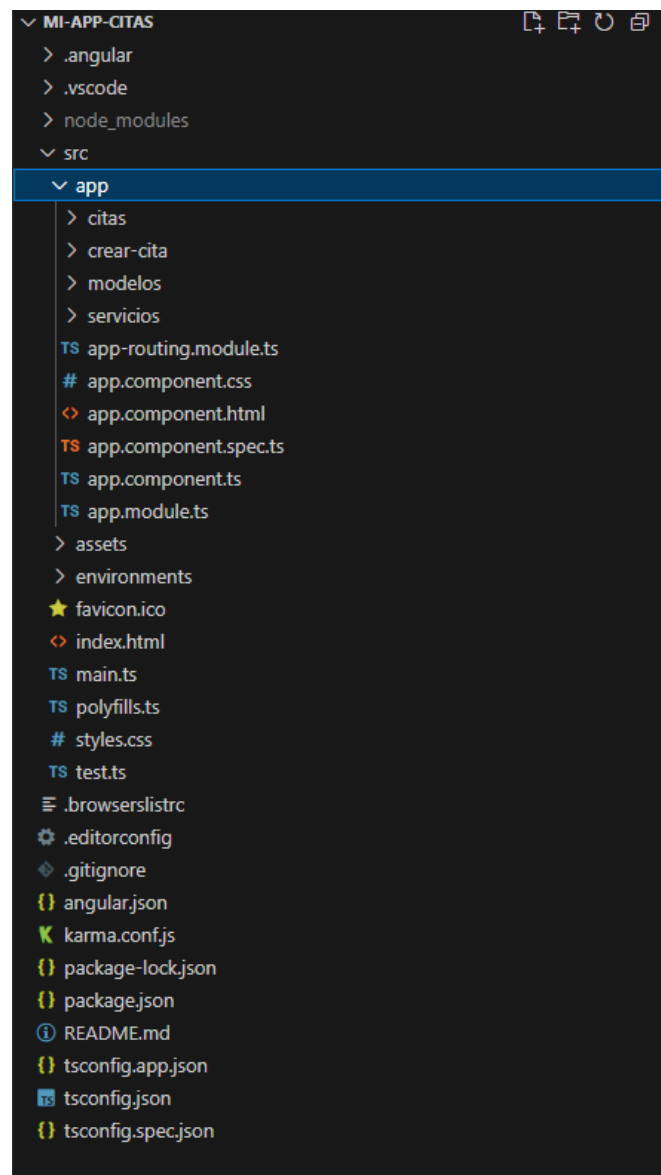
1.2. Repositorios de Front End (Angular):

- Código fuente del Front End en Angular:

mi-app-citas:

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

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



2. Artefactos Contenerizados

2.1. 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)

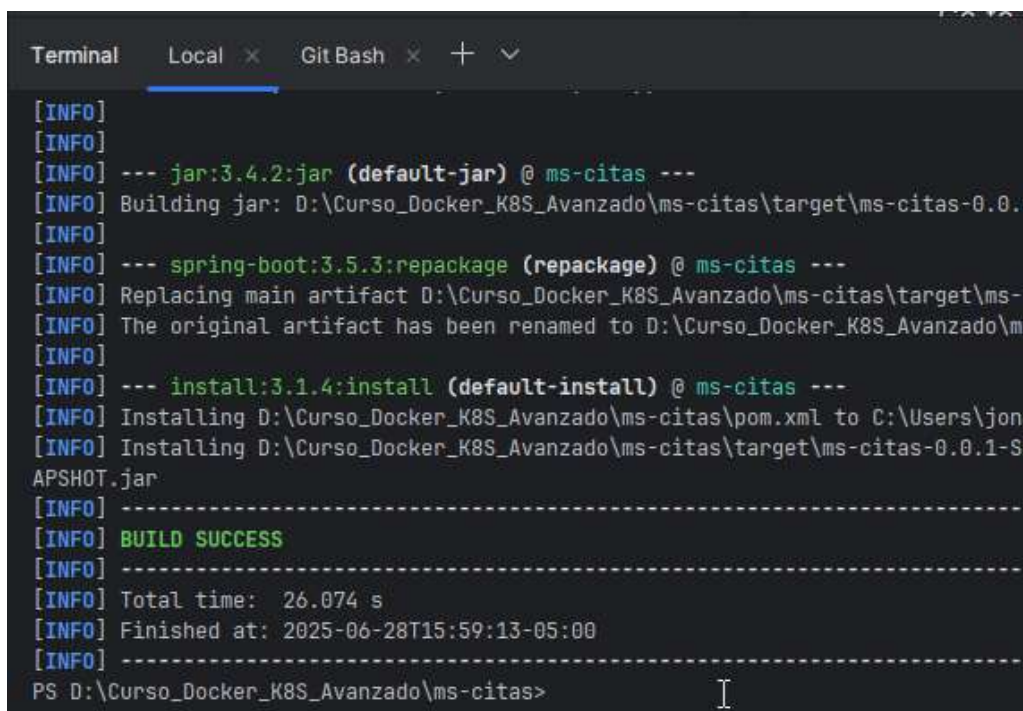
<https://github.com/laquinoq/medical/blob/main/ms-citas/Dockerfile>

```
1 FROM openjdk:21-jdk-slim
2 WORKDIR /app
3 COPY target/ms-citas-0.0.1-SNAPSHOT.jar app.jar
4 EXPOSE 9090
5 ENTRYPOINT ["java", "-jar", "app.jar"]
```

Comando para compilar el microservicio

=====

.mvnw clean install



```
Terminal Local x Git Bash x + v
[INFO]
[INFO]
[INFO] --- jar:3.4.2:jar (default-jar) @ ms-citas ---
[INFO] Building jar: D:\Curso_Docker_K8S_Avanzado\ms-citas\target\ms-citas-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:3.5.3:repackage (repackage) @ ms-citas ---
[INFO] Replacing main artifact D:\Curso_Docker_K8S_Avanzado\ms-citas\target\ms-citas-0.0.1-SNAPSHOT.jar with repackaged one
[INFO] The original artifact has been renamed to D:\Curso_Docker_K8S_Avanzado\ms-citas\target\ms-citas-0.0.1-SNAPSHOT.jar.original
[INFO]
[INFO] --- install:3.1.4:install (default-install) @ ms-citas ---
[INFO] Installing D:\Curso_Docker_K8S_Avanzado\ms-citas\pom.xml to C:\Users\jona\AppData\Local\Temp\maven-repository\org\springframework\boot\spring-boot-starter-parent\3.5.3\pom.xml
[INFO] Installing D:\Curso_Docker_K8S_Avanzado\ms-citas\target\ms-citas-0.0.1-SNAPSHOT.jar to C:\Users\jona\AppData\Local\Temp\maven-repository\org\springframework\boot\spring-boot-starter-parent\3.5.3\ms-citas-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 26.074 s
[INFO] Finished at: 2025-06-28T15:59:13-05:00
[INFO]
PS D:\Curso_Docker_K8S_Avanzado\ms-citas>
```

Crear imagen Docker

=====

docker build -t ms-citas:2.0 .

```

C:\Windows\System32\cmd.exe X + -
Microsoft Windows [Versión 10.0.26100.4349]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Curso_Docker_K8S_Avanzado\medicall\ms-citas>docker build -t ms-citas:1.0 .
[*] Building 44.8s (9/9) FINISHED                                     docker:desktop-Linux
=> [internal] load build definition from Dockerfile                    0.1s
=> => transferring dockerfile: 179B                                   0.0s
=> [internal] load metadata for docker.io/library/openjdk:21-jdk-slim 2.0s
=> [auth] library/openjdk:pull token for registry-1.docker.io         0.0s
=> [internal] load .dockerignore                                       0.1s
=> => transferring context: 2B                                          0.0s
=> [1/3] FROM docker.io/library/openjdk:21-jdk-slim@sha256:7072053847a8a05d7f3a14ebc778a90b38c50ce7e8f199382128 36.0s
=> => resolve docker.io/library/openjdk:21-jdk-slim@sha256:7072053847a8a05d7f3a14ebc778a90b38c50ce7e8f199382128a 0.1s
=> => sha256:af800cd8441e394f9ec3832393ff933c52e165c0965089937e3fb16ea395ea19 204.31MB / 204.31MB 29.6s
=> => sha256:b4972576c83dad66aa1e4f6d08e74f9e551e721a7cb2dc6370fe8da1af5b11e8 4.01MB / 4.01MB 2.3s
=> => sha256:a883e7c4b030119420574a882a52b6431e160fceb7620f61b525d49bc2d58886 29.12MB / 29.12MB 9.7s
=> => extracting sha256:a803e7c4b030119420574a882a52b6431e160fceb7620f61b525d49bc2d58886 6.2s
=> => extracting sha256:b4972576c83dad66aa1e4f6d08e74f9e551e721a7cb2dc6370fe8da1af5b11e8 1.0s
=> => extracting sha256:af800cd8441e394f9ec3832393ff933c52e165c0965089937e3fb16ea395ea19 5.9s
=> [internal] load build context                                       9.1s
=> => transferring context: 29.43MB                                     0.9s
=> [2/3] WORKDIR /app                                                1.4s
=> [3/3] COPY target/ms-citas-0.0.1-SNAPSHOT.jar app.jar           0.3s
=> exporting to image                                                4.5s
=> => exporting layers                                                 3.7s
=> => exporting manifest sha256:9c8ffee168bc0b3d4da817c2444fed692ef102fd83ba2d443a59e3f7e250453e 0.0s
=> => exporting config sha256:7f06b1801d15bc784d55406e73df20ab5a3db5e19504ab4c998bbf833f7ad484 0.0s
=> => exporting attestation manifest sha256:4be9394d478b1432acdd6bd40e608445d97a1abe101c1b7a1fb0a8e76573de59 0.1s
=> => exporting manifest list sha256:8c5ac15acfc888248ae37ca35e8baf49e9628736434399e4ce4b50f0be7f6793 0.0s
=> => naming to docker.io/library/ms-citas:1.0                       0.0s
=> => unpacking to docker.io/library/ms-citas:1.0                    0.6s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/d4mn3smiobhn6cnzta1bgrm6c
D:\Curso_Docker_K8S_Avanzado\medicall\ms-citas>

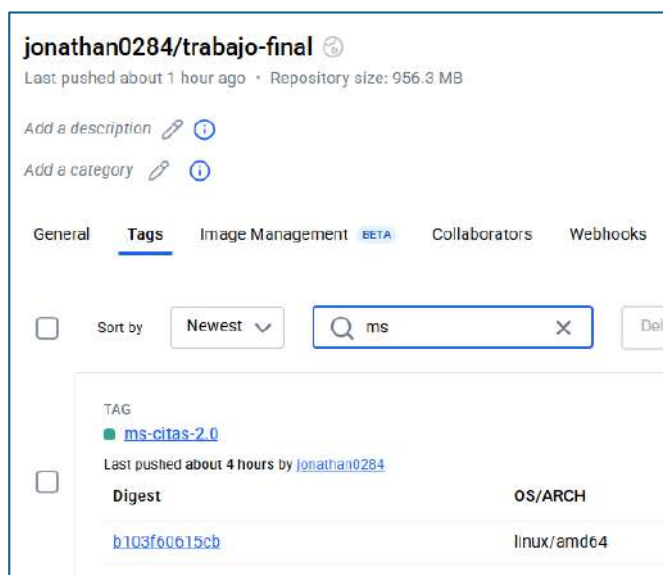
```

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>

```

1  # Etapa 1: Build de Angular con Node 16.20.2
2  FROM node:16.20.2 AS build
3  WORKDIR /app
4
5  # Copia los archivos del proyecto
6  COPY package*.json ./
7  RUN npm install
8
9  COPY . .
10 RUN npx ng build --configuration production --project mi-app-citas
11
12 # Etapa 2: NGINX para servir Angular
13 FROM nginx:stable-alpine
14 COPY --from=build /app/dist/mi-app-citas /usr/share/nginx/html
15
16 # Opcional: archivo personalizado para redirección SPA
17 COPY nginx.conf /etc/nginx/conf.d/default.conf
18
19 EXPOSE 80
20 CMD ["nginx", "-g", "daemon off;"]
21

```

Comando para compilar el microservicio

=====

ng build --configuration production

```

Build at: 2025-06-28T23:10:26.322Z - Hash: df4951add0264d52 - Time: 51496ms

jonat@Jonathan MINGW64 /d/Curso_Docker_K8S_Avanzado/medicall/mi-app-citas (main)
● $ ng build --configuration production
✓ Browser application bundle generation complete.
✓ Copying assets complete.
✓ Index html generation complete.

Initial Chunk Files | Names | Raw Size | Estimated Transfer Size
main.80bfd9bbb7dd123a.js | main | 626.67 kB | 138.13 kB
styles.3c5ac9246b66d5e4.css | styles | 506.27 kB | 33.68 kB
polyfills.3b93124c722fe128.js | polyfills | 33.08 kB | 10.63 kB
runtime.cc940ee8f5ec591e.js | runtime | 1.05 kB | 593 bytes
| Initial Total | 1.14 MB | 183.02 kB

Build at: 2025-06-29T00:40:34.619Z - Hash: 6f198ee18cff4b61 - Time: 40688ms
jonat@Jonathan MINGW64 /d/Curso_Docker_K8S_Avanzado/medicall/mi-app-citas (main)
○ $

```


Crear imagen Docker

=====

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

```

D:\Curso_Docker_K8S_Avanzado\medical\mi-app-citas>docker build -t mi-app-citas:1.0 .
[+] Building 248.0s (17/17) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 570B
=> [internal] load metadata for docker.io/library/node:16.20.2
=> [internal] load metadata for docker.io/library/nginx:stable-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 4.11MB
=> [build 1/6] FROM docker.io/library/node:16.20.2@sha256:f77a1aef2da8d83e45ec990f45df50f1a286c5fe8bbfb8c6e4246c6389705c0b
=> resolve docker.io/library/node:16.20.2@sha256:f77a1aef2da8d83e45ec990f45df50f1a286c5fe8bbfb8c6e4246c6389705c0b
=> CACHED [stage-1 1/3] FROM docker.io/library/nginx:stable-alpine@sha256:aed99734248e851764f1f2146835ecad42b5f994081fa6631cc5d79240891ec9
=> resolve docker.io/library/nginx:stable-alpine@sha256:aed99734248e851764f1f2146835ecad42b5f994081fa6631cc5d79240891ec9
=> CACHED [build 2/6] WORKDIR /app
=> CACHED [build 3/6] COPY package*.json ./
=> CACHED [build 4/6] RUN npm install
=> [build 5/6] COPY . .
=> [build 6/6] RUN npm run build --configuration production --project mi-app-citas
=> [stage-1 2/3] COPY --from=build /app/dist/mi-app-citas /usr/share/nginx/html
=> [stage-1 3/3] COPY nginx.conf /etc/nginx/conf.d/default.conf
=> exporting to image
=> exporting layers
=> exporting manifest sha256:f85a3e07e97928e666c7dcee97e7364fc11b2ee98413f6ae8daa5f4e111c0aca
=> exporting config sha256:b9cde478bbae4f8afe79b9b5f1c4da78878530fa8132e5e4acc67f4561c2ef89
=> exporting attestation manifest sha256:f87a10201aa62bbc32939a1ef51a01f8b28059ba918b8397bc75031b9eafad36
=> exporting manifest list sha256:d618783a1fcea26085afa9766397f82c9874f418321a8ed20856e982d791c351
=> naming to docker.io/library/mi-app-citas:1.0
=> unpacking to docker.io/library/mi-app-citas:1.0

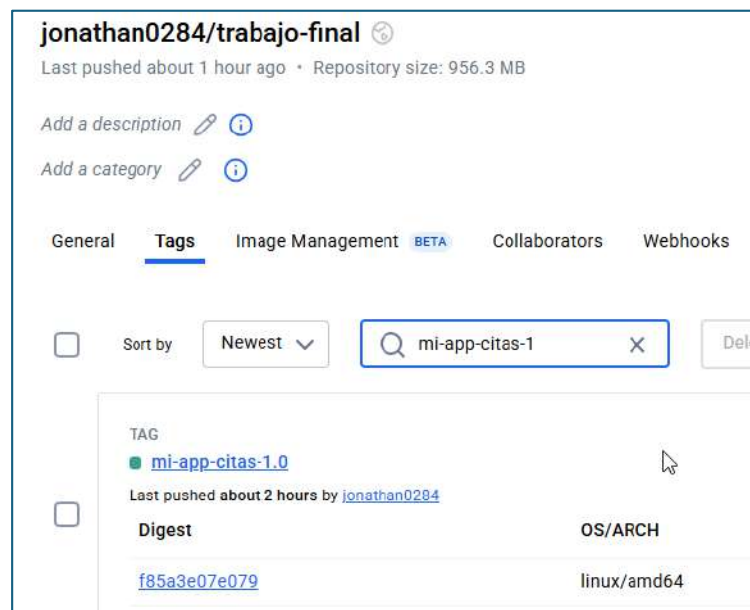
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/x51t0un2a5zgav3pc5e2h4a5f
  
```

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



3. Configuración para MiniKube (Kubernetes)

3.1. Archivos YAML:

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

Microservicio: ms-citas

Archivo: spring-deployment.yaml

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

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: ms-citas
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        app: ms-citas
10   template:
11     metadata:
12       labels:
13         app: ms-citas
14     spec:
15       containers:
16         - name: ms-citas
17           image: jonathan0284/trabajo-final:ms-citas-2.0
18           ports:
19             - containerPort: 9090
20           env:
21             - name: SPRING_PROFILES_ACTIVE
22               value: "default"

```

App Angular: mi-app-citas

Archivo: angular-deployment.yaml

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

```

1  # angular-deployment.yaml
2  apiVersion: apps/v1
3  kind: Deployment
4  metadata:
5    name: angular-app
6  spec:
7    replicas: 1
8    selector:
9      matchLabels:
10       app: angular-app
11    template:
12      metadata:
13        labels:
14          app: angular-app
15      spec:
16        containers:
17          - name: angular-app
18            image: jonathan0284/trabajo-final:mi-app-citas-1.0
19            ports:
20              - containerPort: 80

```

Mongo DB: mongo

Archivo: mongo-deployment.yaml

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

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: mongo
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        app: mongo
10   template:
11     metadata:
12       labels:
13         app: mongo
14     spec:
15       containers:
16         - name: mongo
17           image: mongo
18           ports:
19             - containerPort: 27017
20           env:
21             - name: MONGO_INITDB_ROOT_USERNAME
22               value: root
23             - name: MONGO_INITDB_ROOT_PASSWORD
24               value: rootpass

```

Listado de despliegues

kubectl get deployments

```
D:\Curso_Docker_K8S_Avanzado\medicall\proyecto_final\yaml-files>kubectl get deployments
NAME          READY    UP-TO-DATE    AVAILABLE    AGE
angular-app   1/1      1              1            43m
mongo         1/1      1              1            5h5m
ms-citas      1/1      1              1            4h19m

D:\Curso_Docker_K8S_Avanzado\medicall\proyecto_final\yaml-files>
```

- **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>

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: ms-citas-service
5  spec:
6    selector:
7      app: ms-citas
8    ports:
9      - protocol: TCP
10        port: 80
11        targetPort: 9090
12    type: ClusterIP
```

App Angular: mi-app-citas

Archivo: angular-service.yaml

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

```

1      # angular-service.yaml
2      apiVersion: v1
3      kind: Service
4      metadata:
5        name: angular-service
6      spec:
7        selector:
8          app: angular-app
9        ports:
10         - protocol: TCP
11           port: 80
12           targetPort: 80
13        type: ClusterIP

```

Mongo DB: mongo

Archivo: mongo-service.yaml

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

```

1      apiVersion: v1
2      kind: Service
3      metadata:
4        name: mongo
5      spec:
6        selector:
7          app: mongo
8        ports:
9         - protocol: TCP
10           port: 27017
11           targetPort: 27017

```

4. 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.

```

1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4    name: miapp-ingress
5    annotations:
6      nginx.ingress.kubernetes.io/ssl-redirect: "false"
7  spec:
8    rules:
9      - host: miapp.local
10      http:
11        paths:
12          - path: /api
13            pathType: Prefix
14            backend:
15              service:
16                name: ms-citas-service
17                port:
18                  number: 80
19          - path: /
20            pathType: Prefix
21            backend:
22              service:
23                name: angular-service
24                port:
25                  number: 80

```

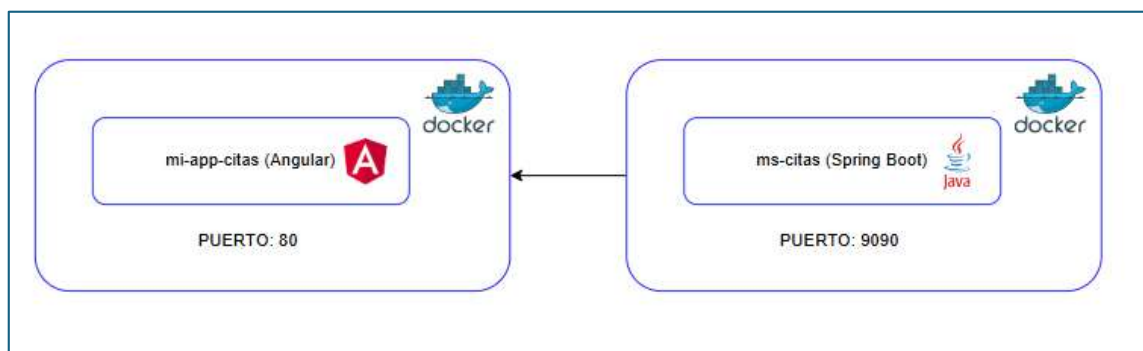
5. Documentación

5.1. Manual de Despliegue:

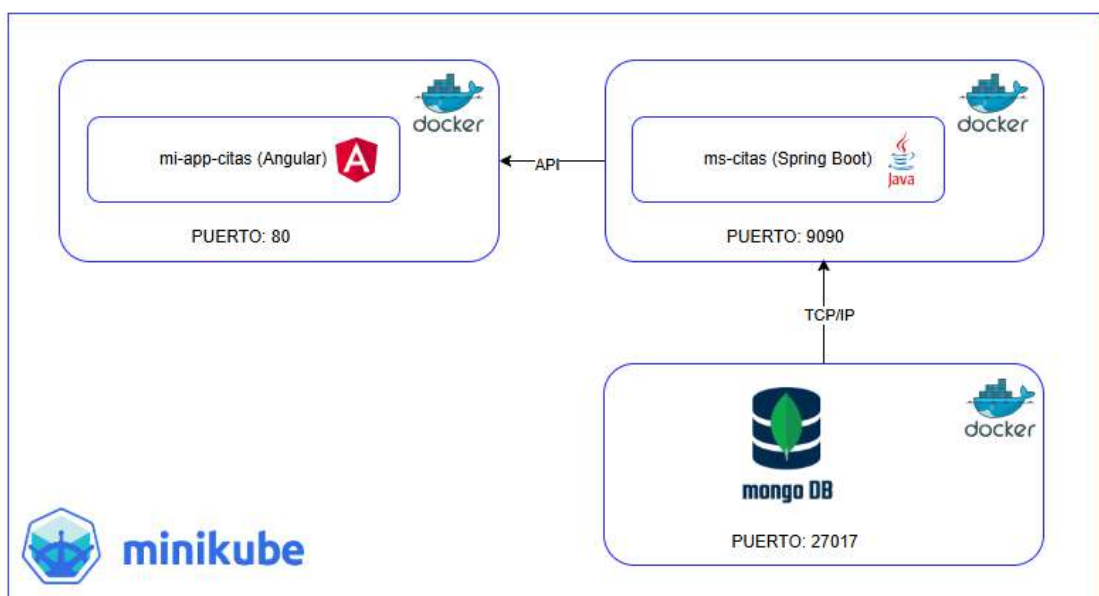
- Pasos detallados para:
 - Contenerizar y construir las imágenes:
 - Desplegar en Kubernetes utilizando los manifiestos YAML.

5.2. Diagrama de Arquitectura:

- Representación gráfica de la solución:
 - Estructura de componentes:



- Despliegue en Kubernetes (MiniKube):



6. Resultados

6.1. 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

```
D:\Curso_Docker_K8S_Avanzado\medicall\proyecto_final\yaml-files>kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
angular-app   1/1     1             1           110m
mongo         1/1     1             1           6h12m
ms-citas      1/1     1             1           5h26m

D:\Curso_Docker_K8S_Avanzado\medicall\proyecto_final\yaml-files>
```

kubectl get service

```
D:\Curso_Docker_K8S_Avanzado\medicall\proyecto_final\yaml-files>kubectl get service
NAME            TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
angular-service ClusterIP    10.111.126.207 <none>       80/TCP           122m
kubernetes      ClusterIP    10.96.0.1      <none>       443/TCP          6h21m
mongo           ClusterIP    10.97.41.181   <none>       27017/TCP        6h12m
ms-citas-service ClusterIP    10.107.87.23   <none>       80/TCP           122m

D:\Curso_Docker_K8S_Avanzado\medicall\proyecto_final\yaml-files>
```

kubectl get pods

```
D:\Curso_Docker_K8S_Avanzado\medicall\proyecto_final\yaml-files>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
angular-app-fd6f4d4f7-fq2cf        1/1     Running   0           111m
mongo-79bbbf84-r4jbj               1/1     Running   0           6h13m
ms-citas-8486d79bbd-qlcp6          1/1     Running   0           5h27m

D:\Curso_Docker_K8S_Avanzado\medicall\proyecto_final\yaml-files>
```


- Pruebas realizadas en los servicios mediante Postman

Video DEMO:

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

Controller Usuarios

Method GET: Obtener todos los usuarios

The screenshot shows a Postman interface for a GET request to `http://miapp.local/api/usuarios`. The response status is 200 OK. The response body is displayed in JSON format, showing two user objects.

Key	Value
id	"686062736059d1814111115c"
dni	"46325889"
nombre	"Juan Perez"
edad	30
direccion	"Lima"
telefono	" +51 965 855 447"

Key	Value
id	"6860629b6059d1814111115d"
dni	"46398745"
nombre	"Miguel Angel Diaz Perez"
edad	25
direccion	"Lima"
telefono	" +51 965 852 777"

Method POST: Registrar Usuario

The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** http://miapp.local/api/usuarios
- Body Type:** JSON
- Request Body (JSON):**

```

1  {
2    "dni": "40369852",
3    "nombre": "Ronald Saenz",
4    "edad": 33,
5    "direccion": "Lima",
6    "telefono": "+51 963 654 888"
7  }

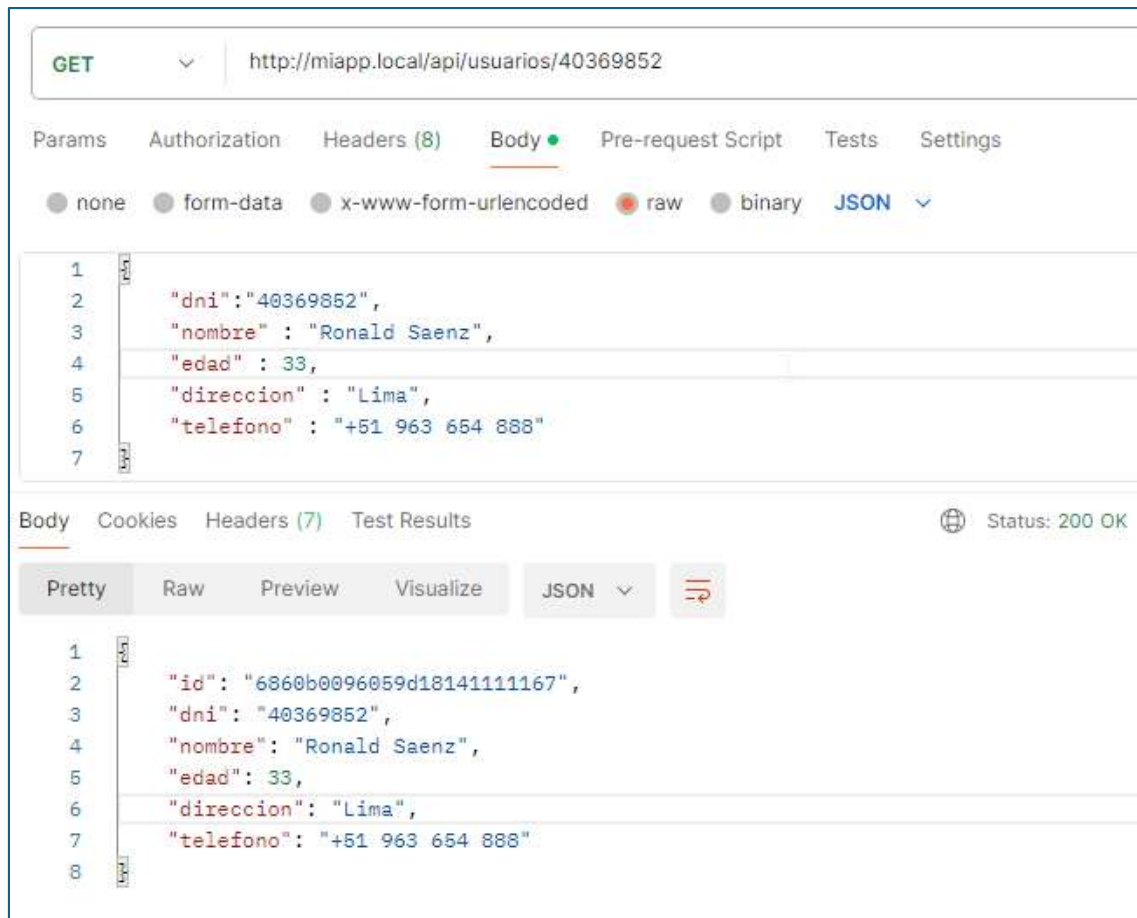
```
- Status:** 200 OK
- Response Body (JSON):**

```

1  {
2    "id": "6860b0096059d18141111167",
3    "dni": "40369852",
4    "nombre": "Ronald Saenz",
5    "edad": 33,
6    "direccion": "Lima",
7    "telefono": "+51 963 654 888"
8  }

```

Method GET: Buscar Usuario por DNI



The screenshot displays a REST client interface with a GET request to `http://miapp.local/api/usuarios/40369852`. The 'Body' tab is selected, showing the request body in JSON format. Below the request, the 'Body' tab of the response is also selected, showing the response body in JSON format. The status of the response is '200 OK'.

Request Body:

```

1  {
2    "dni": "40369852",
3    "nombre": "Ronald Saenz",
4    "edad": 33,
5    "direccion": "Lima",
6    "telefono": "+51 963 654 888"
7  }

```

Response Body:

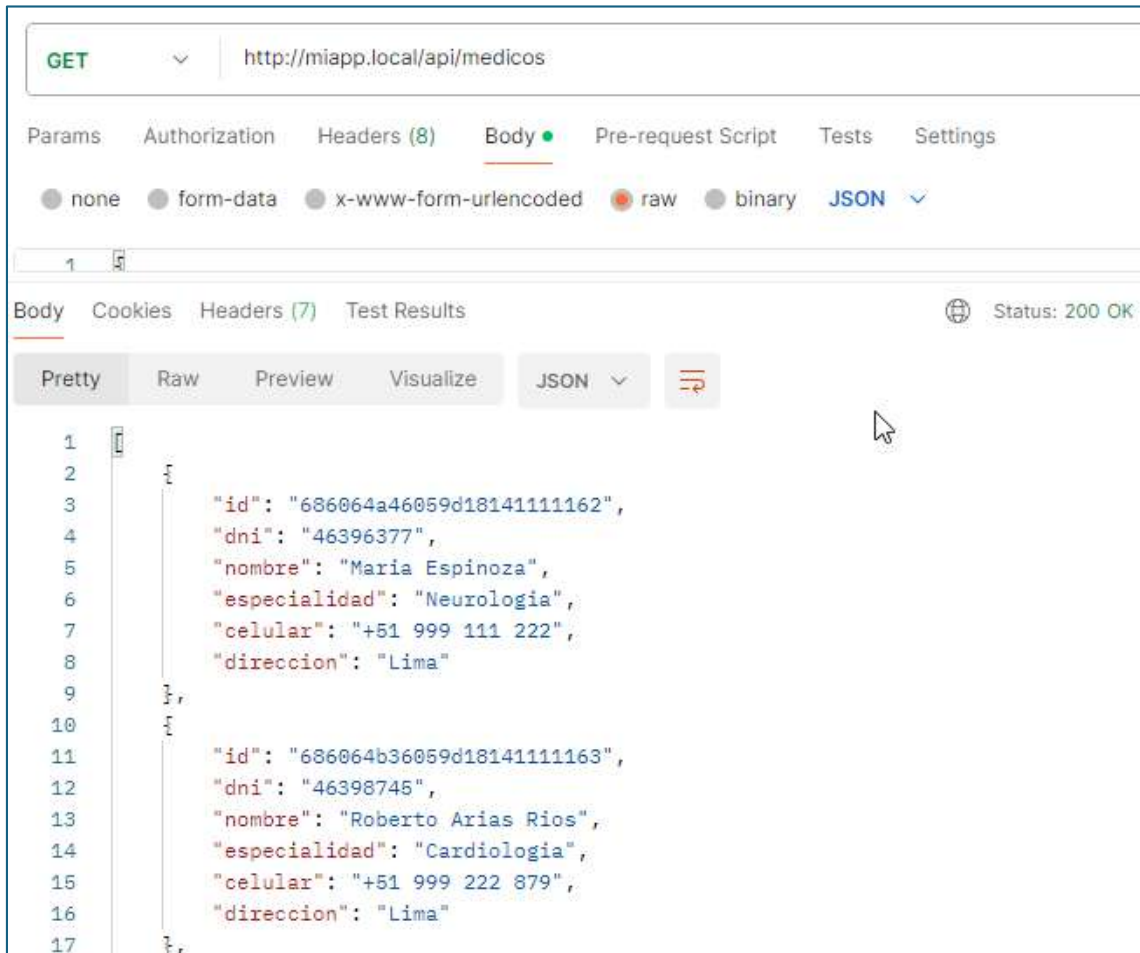
```

1  {
2    "id": "6860b0096059d18141111167",
3    "dni": "40369852",
4    "nombre": "Ronald Saenz",
5    "edad": 33,
6    "direccion": "Lima",
7    "telefono": "+51 963 654 888"
8  }

```

- **Controller Medicos**

Method GET: Obtener todos los medicos



Method POST: Registrar medico

The screenshot displays a REST client interface for a POST request to the endpoint `http://miapp.local/api/medicos`. The request body is a JSON object containing doctor information. The response status is 200 OK, and the response body is a JSON object containing the created doctor's details, including a new ID.

Request:

```

1  {
2    "dni": "42630598",
3    "nombre": "Julio Velasquez",
4    "especialidad": "Dermatologia",
5    "celular": "+51 999 222 879",
6    "direccion": "Lima"
7  }

```

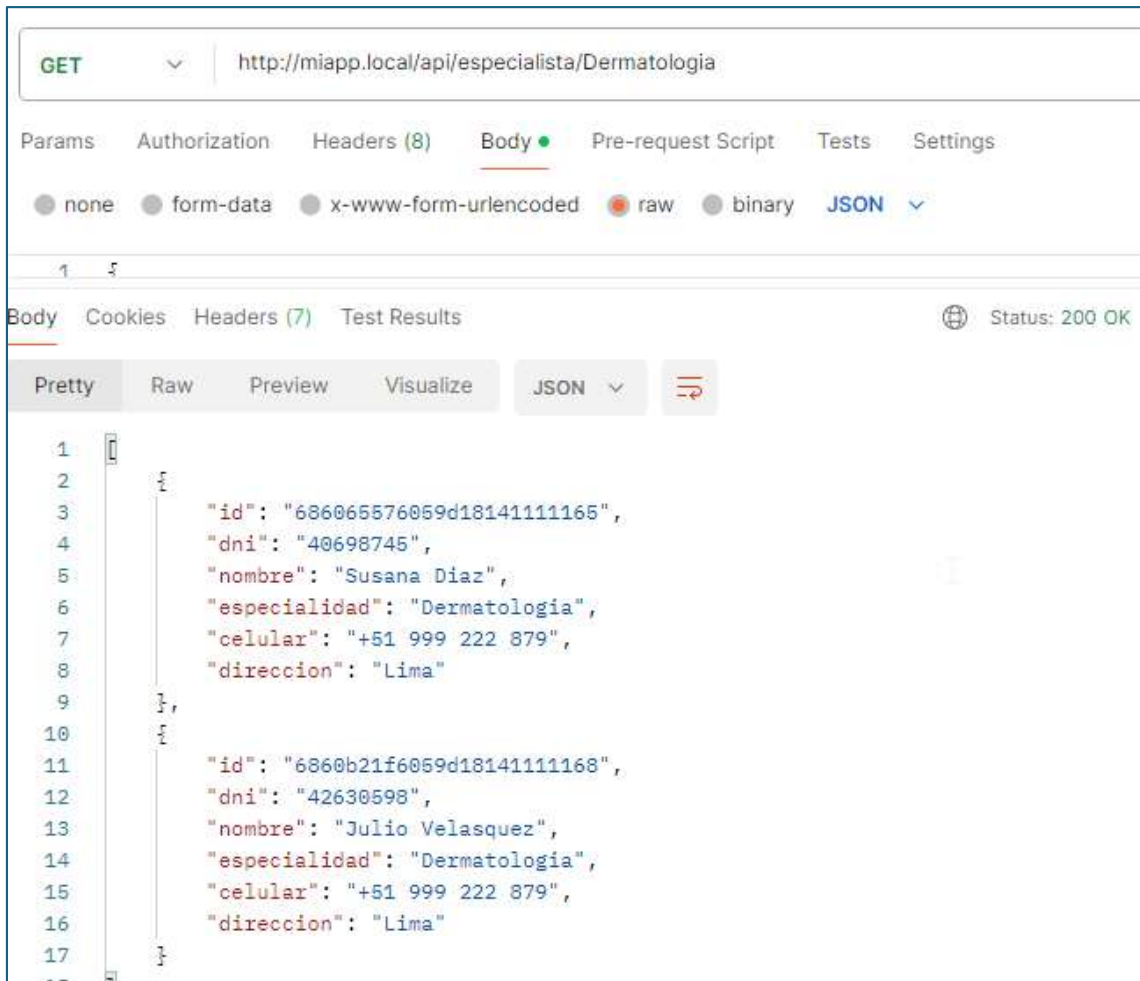
Response:

```

1  {
2    "id": "6860b21f6059d18141111168",
3    "dni": "42630598",
4    "nombre": "Julio Velasquez",
5    "especialidad": "Dermatologia",
6    "celular": "+51 999 222 879",
7    "direccion": "Lima"
8  }

```

Method GET: Obtener médicos por especialidad



The screenshot shows a web browser's developer tools interface. The top bar indicates a GET request to the URL `http://miapp.local/api/especialista/Dermatologia`. The 'Body' tab is selected, showing a JSON response. The status bar at the bottom right indicates 'Status: 200 OK'.

The JSON response is as follows:

```

1  {
2    {
3      "id": "686065576059d18141111165",
4      "dni": "40698745",
5      "nombre": "Susana Diaz",
6      "especialidad": "Dermatologia",
7      "celular": "+51 999 222 879",
8      "direccion": "Lima"
9    },
10   {
11     "id": "6860b21f6059d18141111168",
12     "dni": "42630598",
13     "nombre": "Julio Velasquez",
14     "especialidad": "Dermatologia",
15     "celular": "+51 999 222 879",
16     "direccion": "Lima"
17   }
18 }

```

- **Controller Citas**

Method GET: Obtener todas las citas

The screenshot shows a REST client interface with a GET request to `http://miapp.local/api/citas`. The response is a JSON array containing one object, which is displayed in a 'Pretty' format. The status is 200 OK.

```

1  [
2    {
3      "id": "6860949d6059d18141111166",
4      "codigoCita": "WTAD7X",
5      "fechaHoraCita": "2025-06-24 08:00:00",
6      "motivo": "Dolor de torax",
7      "paciente": {
8        "dni": "46325889",
9        "nombre": "Juan Perez",
10       "direccion": "Lima"
11      },
12      "medico": {
13        "dni": "46398745",
14        "nombre": "Roberto Arias Rios",
15        "especialidad": "Cardiologia",
16        "celular": "+51 999 222 879"
17      }
18    }
19  ]
  
```

Method POST: Registrar cita

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://miapp.local/api/citas
- Body Type:** JSON
- Request Body:**

```

1 {
2   "dniPaciente": "46398745",
3   "dniMedico" : "46398745",
4   "fechaHoraCita" : "2025-07-15 09:00:00",
5   "motivo" : "Dolor de torax",
6   "lugar" : {}

```
- Status:** 200 OK
- Response Body (JSON):**

```

1 {
2   "id": "6860b3f06059d18141111169",
3   "codigoCita": "00AQY7",
4   "fechaHoraCita": "2025-07-15 09:00:00",
5   "motivo": "Dolor de torax",
6   "paciente": {
7     "dni": "46398745",
8     "nombre": "Miguel Angel Diaz Perez",
9     "direccion": "Lima"
10  },
11  "medico": {}

```

The screenshot shows the 'Gestión de Citas Médicas' web application. The left sidebar contains navigation links: Inicio, Citas, Crear cita, Ver citas, and Perfil. The main content area displays the 'Listado de Citas' with the following table:

Código	Fecha-Hora	Paciente	Médico	Especialidad	Motivo	Estado
WTAD7X	2025-06-24 08:00:00	Juan Perez	Roberto Arias Rios	Cardiología	Dolor de torax	Creada
00AQY7	2025-07-15 09:00:00	Miguel Angel Diaz Perez	Roberto Arias Rios	Cardiología	Dolor de torax	Creada

Below the table is a pagination control showing page 1 of 1.

© 2025 Castro Médico - Todos los derechos reservados