

DISEÑO DE ARQUITECTURAS DE SISTEMAS

**Trabajo Unidad 2**

**Tema:** Segmentar la arquitectura de servicios

**Docente:** Msc. Jorge Oswaldo Loja Cajas

**Alumno:** Saúl Marcelo Guamán Cabrera

Contenido

[Configuración. 2](#_Toc139181001)

[Descomprimir el proyecto 2](#_Toc139181002)

[Modificar el proyecto Java 2](#_Toc139181003)

[Compilar proyecto Java. 2](#_Toc139181004)

[Eliminar pgadmin 3](#_Toc139181005)

[API Layer 3](#_Toc139181006)

[Configuración de Nginx como proxy inverso 3](#_Toc139181007)

[Configuración de Nginx en docker-compose.yml 4](#_Toc139181008)

[Construir imagen myapp. 4](#_Toc139181009)

[Ejecutar el proyecto 4](#_Toc139181010)

[Funcionamiento 5](#_Toc139181011)

[Wildfly 5](#_Toc139181012)

[Proxy 5](#_Toc139181013)

[Validación 6](#_Toc139181014)

[Accediendo a través del servidor de aplicaciones 6](#_Toc139181015)

[Accediendo a través del proxy inverso 6](#_Toc139181016)

[Análisis de la diferencia 7](#_Toc139181017)

[Conclusión 7](#_Toc139181018)

SEGMENTAR LA ARQUITECTURA DE SERVICIOS

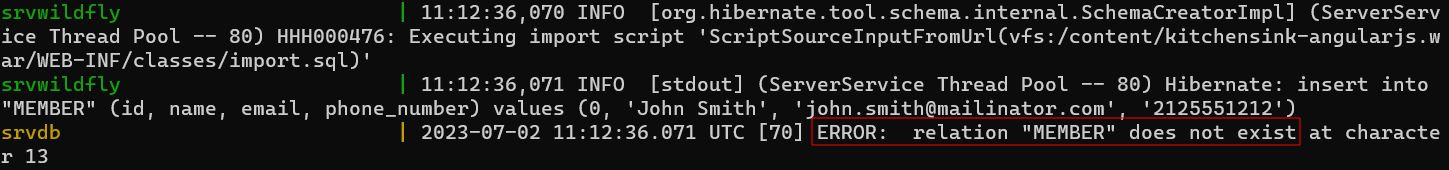
# Configuración.

## Descomprimir el proyecto

El archivo descargado se descomprime utilizando cualquier herramienta que permita extraer los archivos.

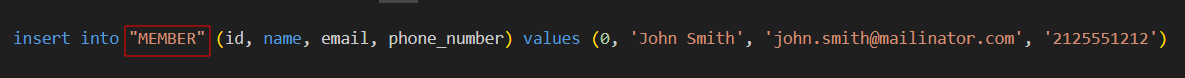
## Modificar el proyecto Java

Si se ejecuta el proyecto tal como se descarga, en el despliegue se encontrará con el siguiente error:

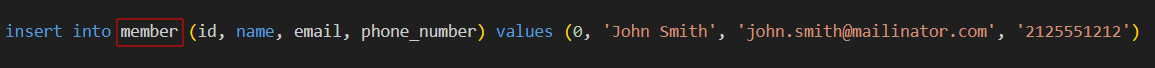


Para evitar el error, se modifica el archivo: **import.sql** ubicado en la ruta: quickstart-23.0.0.Final\kitchensink-angularjs\src\main\resources\

Cambiando de:

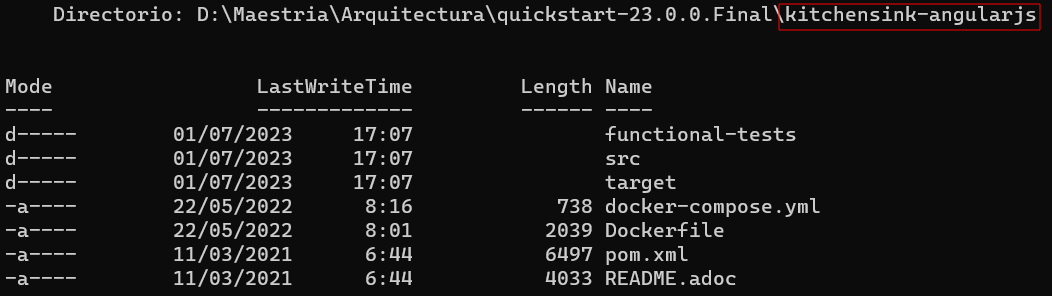


A:



## Compilar proyecto Java.

Ingresar hasta la carpeta: **kitchensink-angularjs**



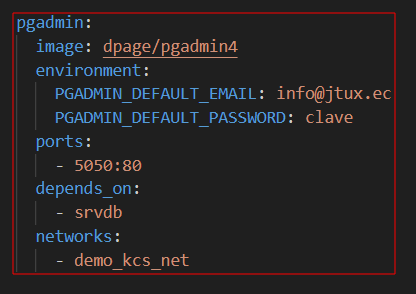
En esta path se abre una terminal y se ejecuta el comando:

mvn clean package



## Eliminar pgadmin

En el archivo: **docker-compose.yml** se elimina la configuración correspondiente a: pgadmin



## API Layer

Para la implementación de API Layer, se utiliza nginx como proxy inverso, permitiendo que las peticiones que lleguen al puerto 80 se redirijan al servidor de aplicacones srvwildfly, para lo cual se debe realizar el siguiente proceso:

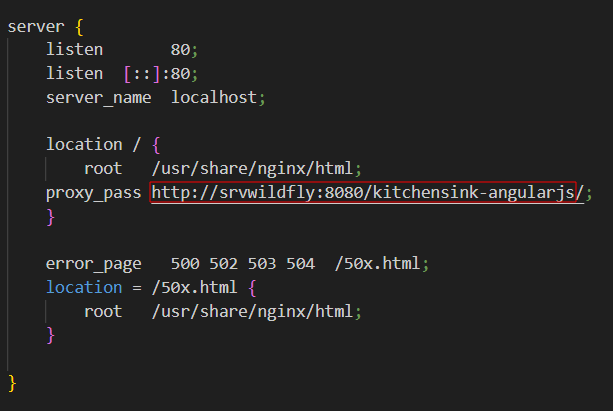
### Configuración de Nginx como proxy inverso

En la ruta:

quickstart-23.0.0.Final\kitchensink-angularjs

Se crea el archivo default.conf que contiene la configuración específica que requiere Nginx, para funcionar como proxy inverso.

En el archivo se ingresa la siguiente configuración:



Se determina que las peticiones que llegan a nginx en el puerto 80 se redirijan al:

Contenedor: **srvwildfly**

En el puerto: **8080**

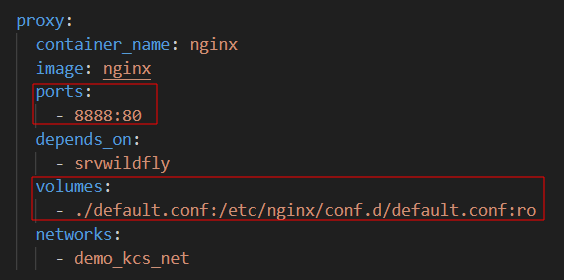
Con el dominio: **kitchensink-angularjs**

### Configuración de Nginx en docker-compose.yml

Para que se ejecute un contenedor de nginx con la configuración establecida en el archivo:

default.conf.

Al archivo: docker-compose.yml se agrega la siguiente configuración:



Se determina que el puerto 80 del contenedor se va a mapear en el puerto 8888 del host y que el contenedor utilizará el archivo default.conf con la configuración establecida.

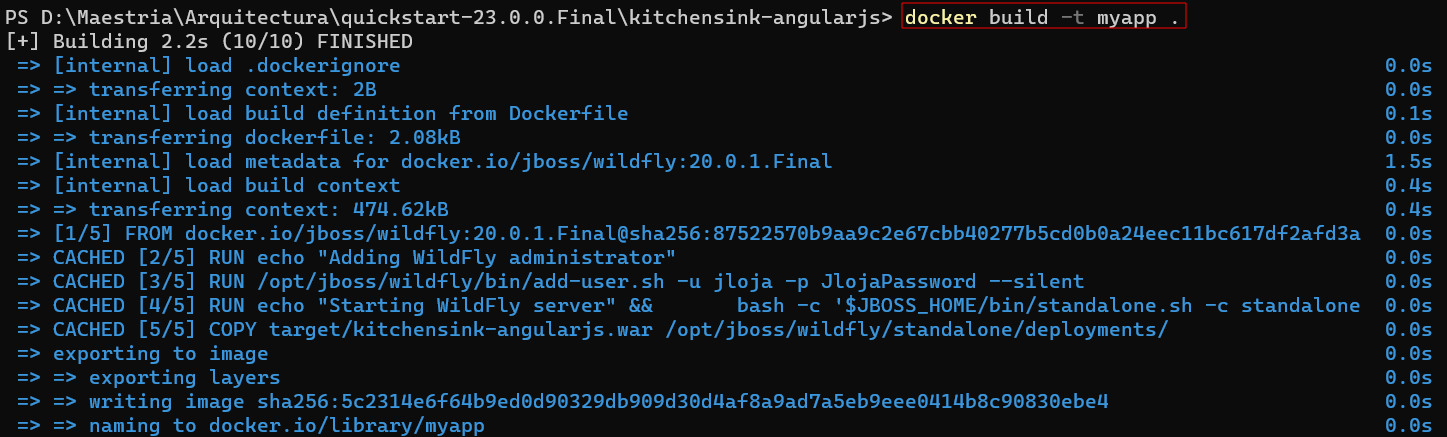
## Construir imagen myapp.

Abrir una terminal donde se encuentra el archivo: **docker-compose.yml,** la ruta es:

quickstart-23.0.0.Final\kitchensink-angularjs

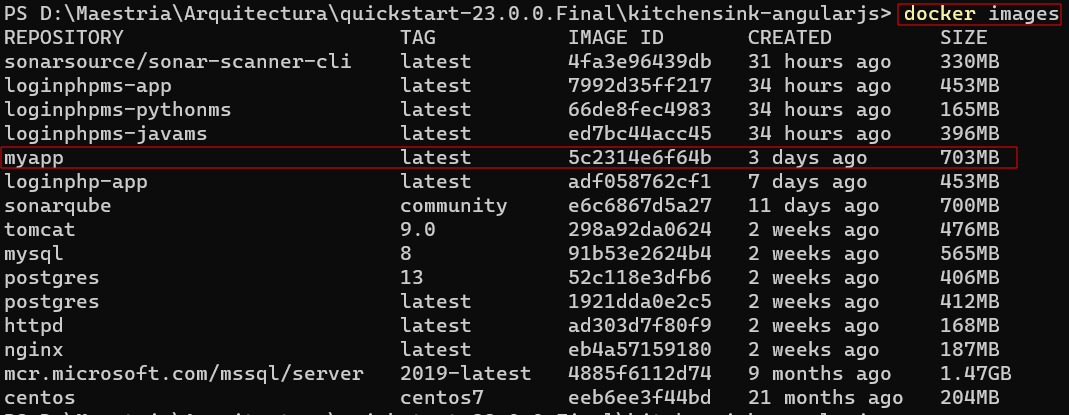
y ejecutar el comando:

docker build -t myapp .



Finalizada la construcción se puede visualizar la imagen construida con el comando:

docker images



## Ejecutar el proyecto

Para ejecutar el Proyecto, abrir una terminal en la ruta:

quickstart-23.0.0.Final\kitchensink-angularjs\

Ejecutar el comando:

docker compose up

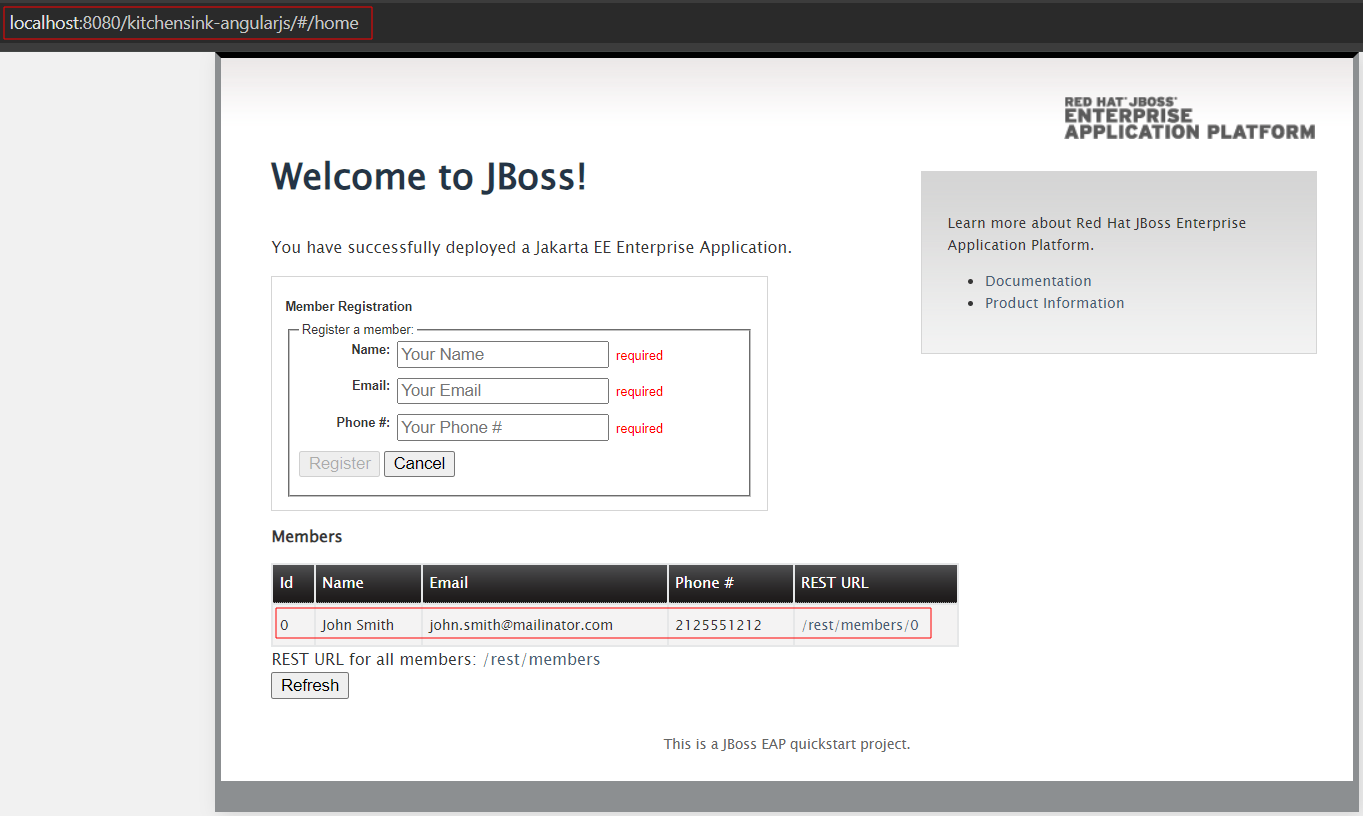
# Funcionamiento

Para acceder a través del navegador nos basamos en la configuración establecida en el archivo:docker-compose.yml.

## Wildfly

El puerto **8080** del servidor wildfly se mapea al puerto **8080** del host

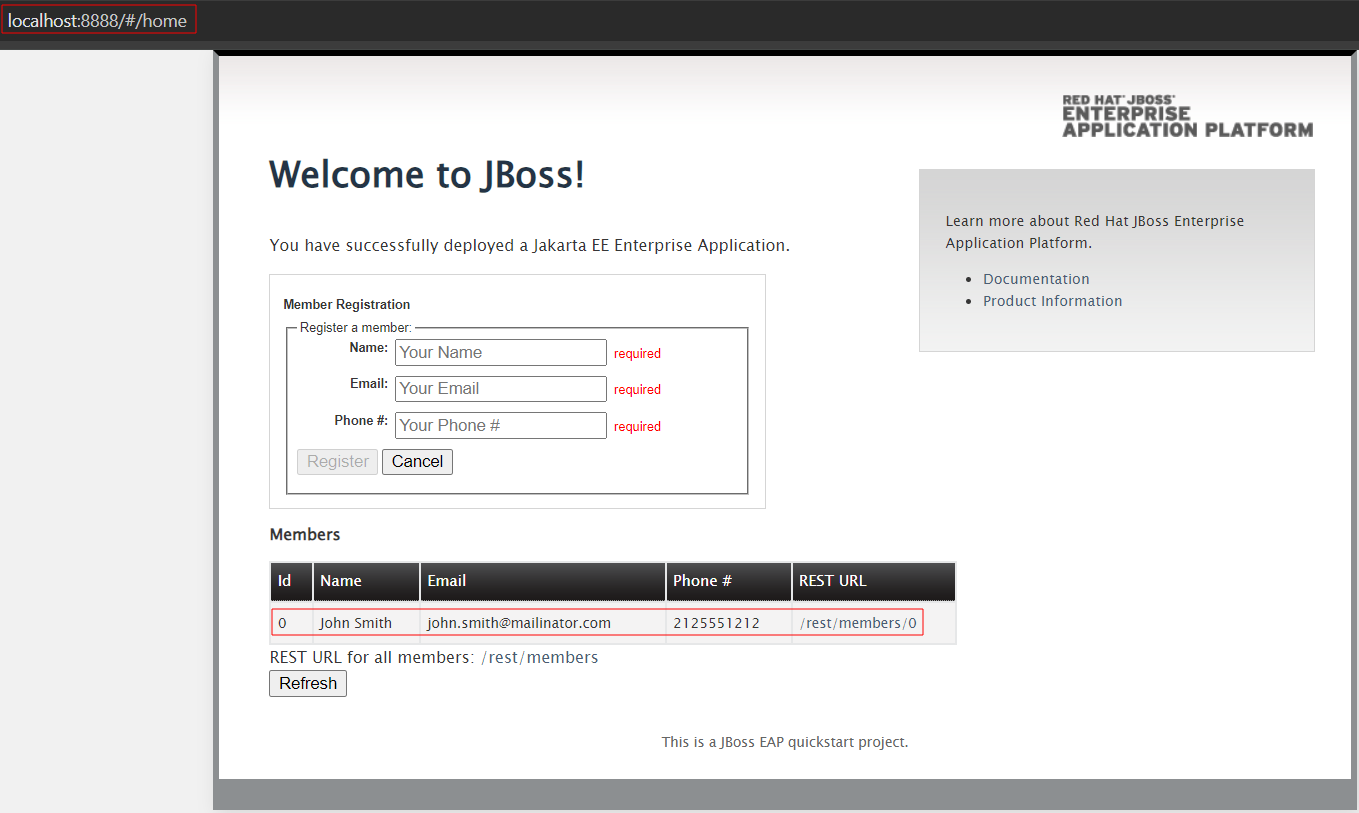
<http://localhost:8080/kitchensink-angularjs>



## Proxy

El puerto **80** de proxy se mapea al puerto **8888** del host

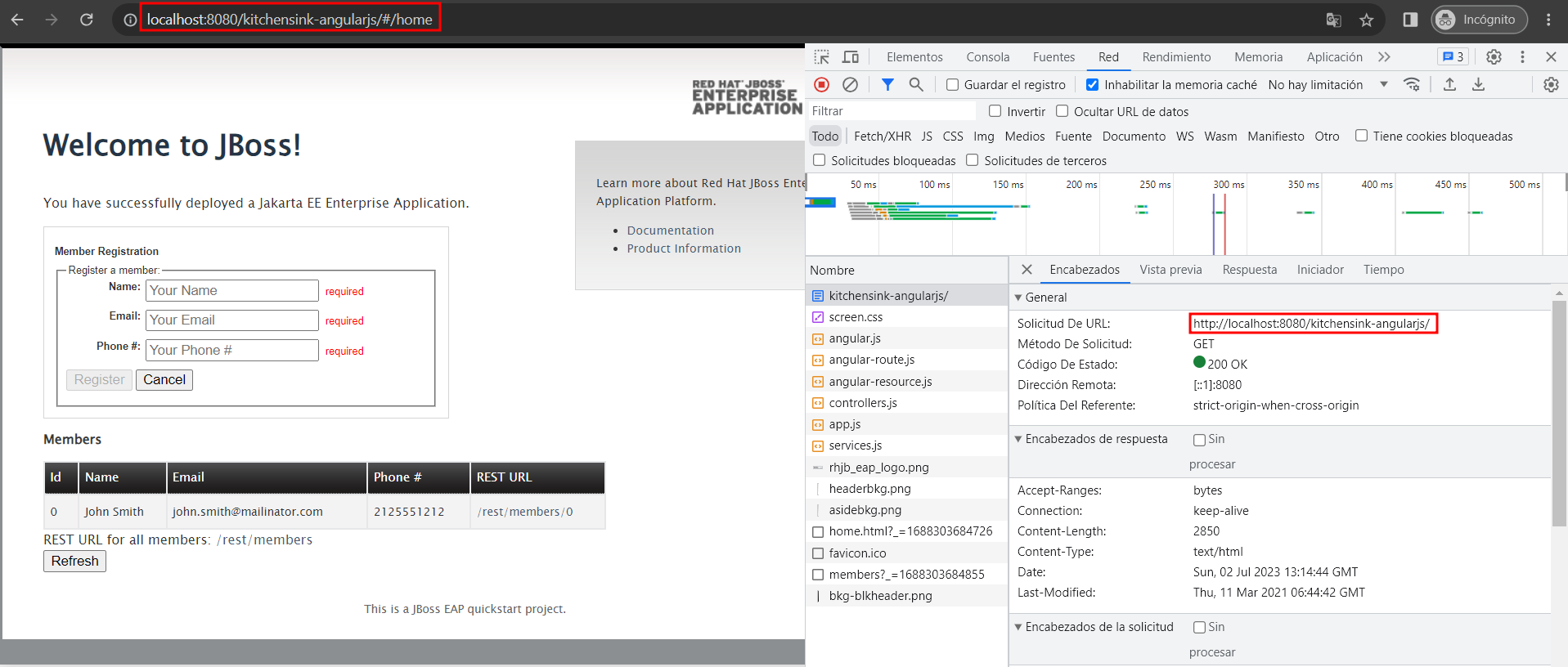
<http://localhost:8888>



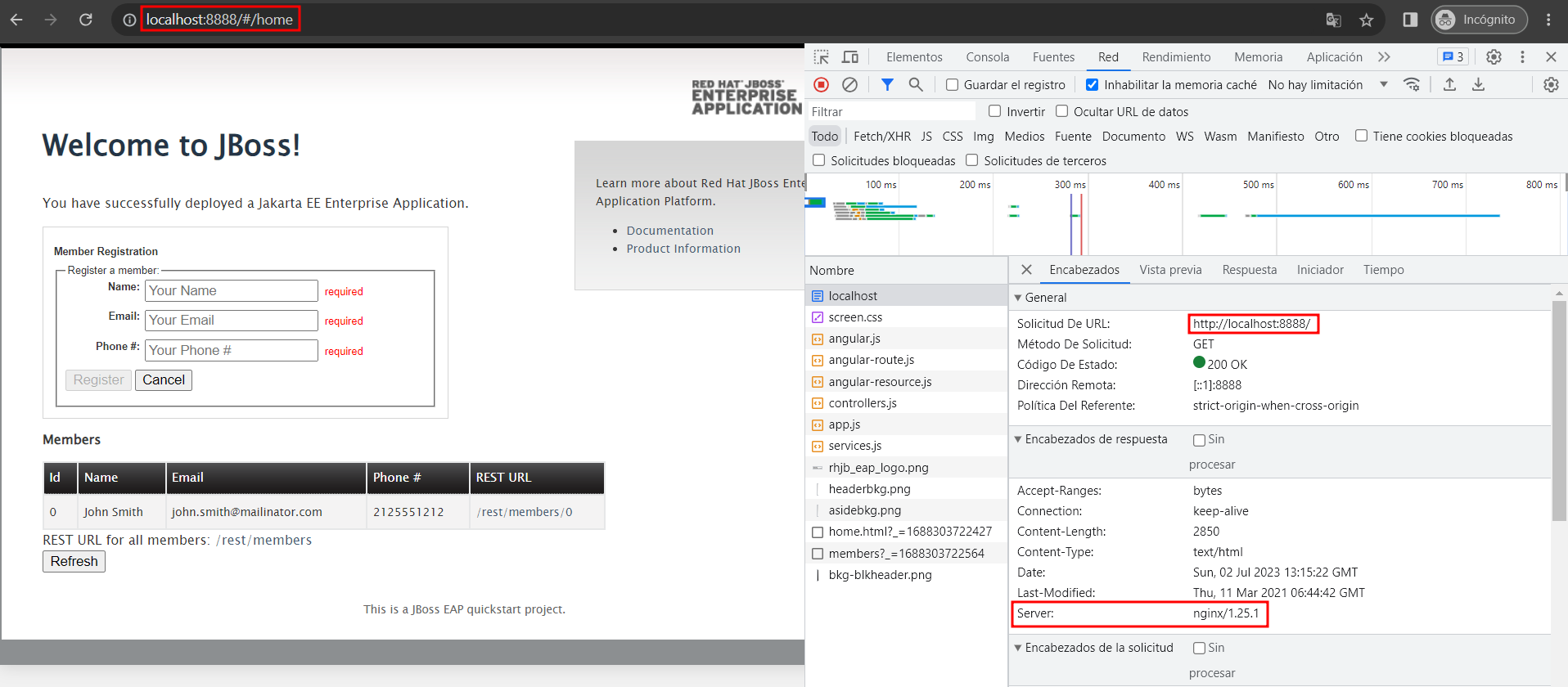
# Validación

Para validar el funcionamiento del ApiLayer compararemos los headers del navegador cuando se accede mediante el servidor de aplicaciones y luego cuando se accede a través de proxy inverso.

## Accediendo a través del servidor de aplicaciones



## Accediendo a través del proxy inverso



## Análisis de la diferencia

Si observamos las imágenes de cuando se accede a través de l servidor de aplicaciones y luego a través del proxy podemos observar las siguientes diferencias:

|  |  |  |
| --- | --- | --- |
|  | Servidor de aplicaciones | Proxy |
| Puerto | 8080 | 80 |
| URL | http://localhost:8080/kitchensink-angularjs | http://localhost:8888/ |
| Tag Proxy | - | Server: nginx/1.25.1 |

Cuando se accede a través del API Layer en los headers de respuesta se puede observar que se incrementa el tag Server que determina que para llegar al servidor de aplicaciones las peticiones pasan por nginx/1.25.1.

# Conclusión

De acuerdo al proyecto desarrollado se determina que en este caso implementar un servidor proxy evita el acceso directo a Wildfly; lo cual en la realidad es una mejora en la seguridad, puesto que de esta manera se impide que el usuario final pueda conocer el servidor de aplicaciones.