

Utilizando AWS EBS

16/02/2025



XAMPP	2
HeidiSQL	4
Backend/Frontend Local	7
Backend.....	7
Frontend.....	12
RDS	15
EBS	21
Backend.....	21
Frontend.....	27

Instalación local

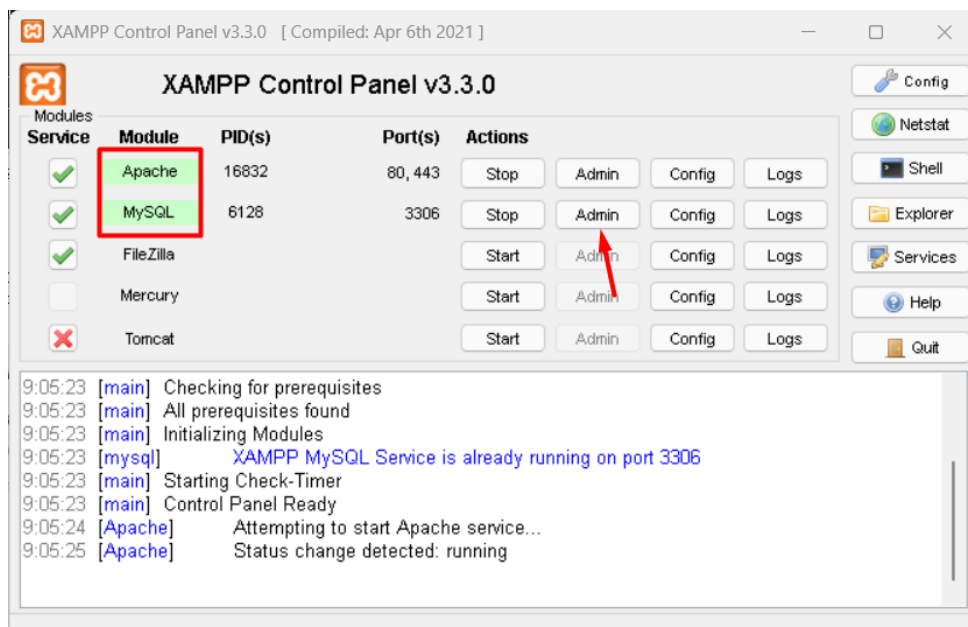
XAMPP

Empezaremos este trabajo con la instalación y el uso en local mediante Docker ya que luego será más fácil implementarlo en AWS una vez ya lo tengamos en local.

El primer paso y más importante para realizar esto en local será que tengamos creada una base de datos en XAMPP usando MySQL, en mi caso ya la tenía creada con el usuario "root".

(Si no esta la base de datos creada usar este video:

https://www.youtube.com/watch?v=tHoBYIBx2zs&ab_channel=Init)



Para comprobar que nos funciona correctamente todo podemos clicar sobre "Admin" en la parte de "MySQL" y se nos abrirá lo siguiente en nuestro navegador alojado en localhost.

localhost/phpmyadmin/

phpMyAdmin

Reciente Favoritas

- Nueva
- apartment4v
- information_schema
- mysql
- performance_schema
- phpmyadmin
- pwpersonal
- test

Base de datos SQL Estado actual Cuentas de usuarios Exportar Más

Configuraciones generales

Cotejamiento de la conexión al servidor:

utf8mb4_unicode_ci

Más configuraciones

Configuraciones de apariencia

Idioma (Language)

Español - Spanish

Tema pmahomme Ver todo

Servidor de base de datos

- Servidor: 127.0.0.1 via TCP/IP
- Tipo de servidor: MariaDB
- Conexión del servidor: No se está utilizando SSL
- Versión del servidor: 10.4.32-MariaDB - mariadb.org binary distribution
- Versión del protocolo: 10
- Usuario: root@localhost
- Conjunto de caracteres del servidor: UTF-8 Unicode (utf8mb4)

Servidor web

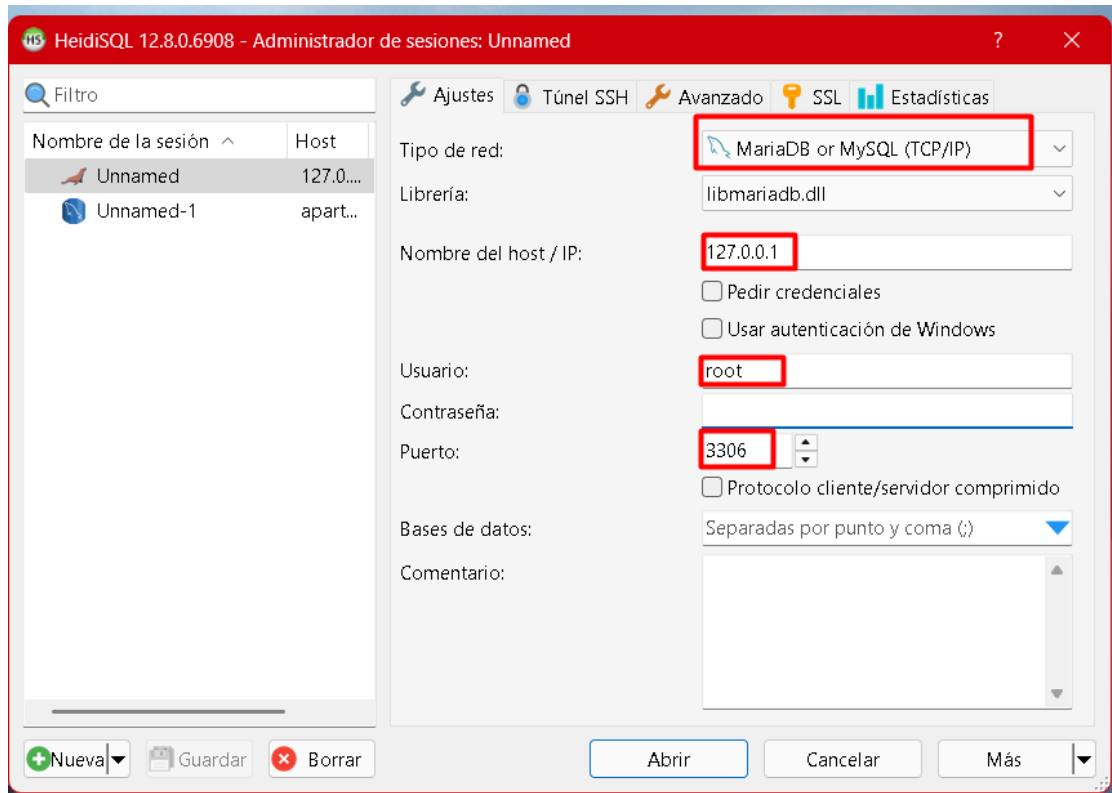
- Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
- Versión del cliente de base de datos: libmysql - mysqlnd 8.2.12
- extensión PHP: mysqli curl mbstring
- Versión de PHP: 8.2.12

phpMyAdmin

- Acerca de esta versión: 5.2.1, versión estable más reciente: 5.2.2
- Documentación
- Página oficial de phpMyAdmin
- Contribuir
- Obtener soporte
- Lista de cambios
- Licencia

HeidiSQL

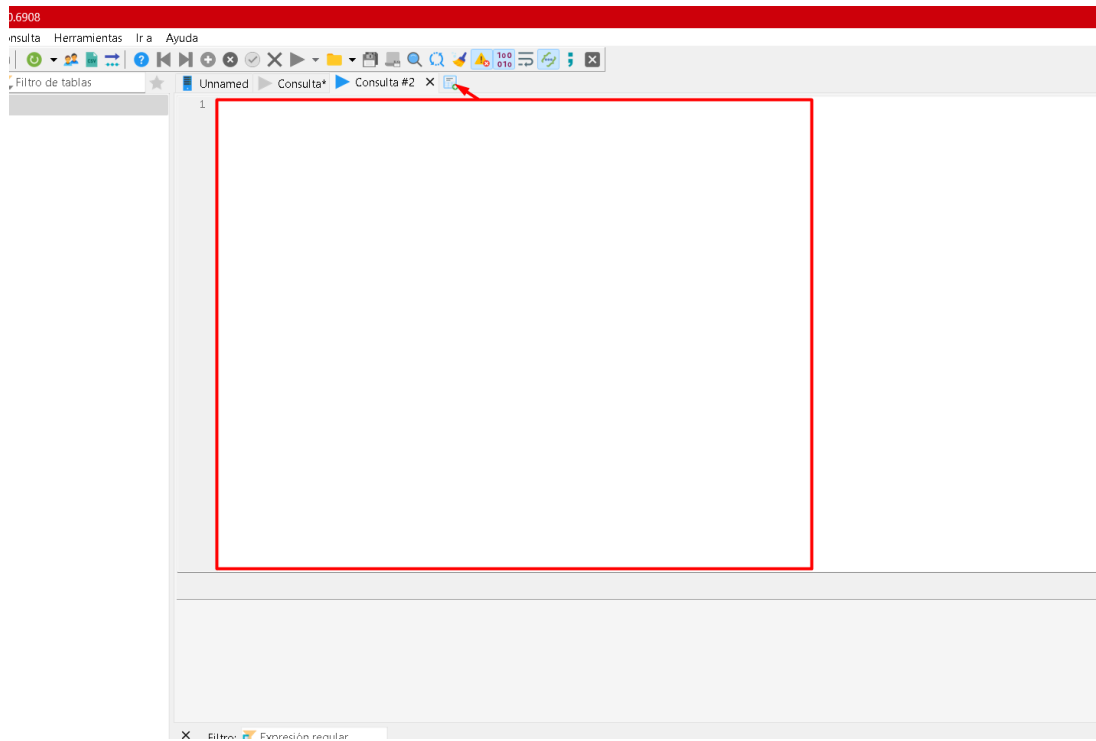
Una vez realizada la parte de XAMPP podremos utilizar HeidiSQL como gestor de bases de datos para poder gestionar nuestra base de datos creada en XAMPP.



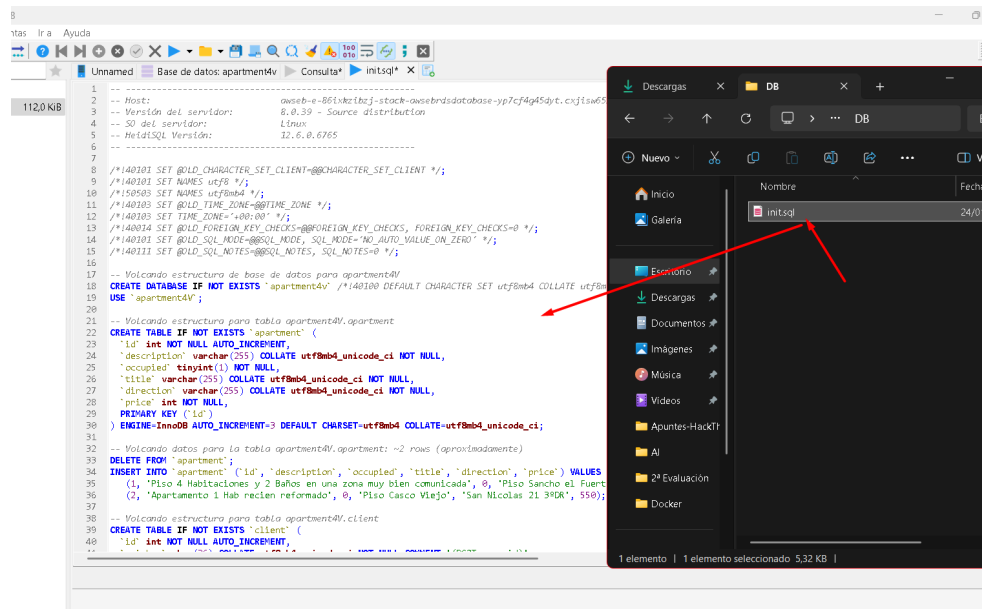
Como se puede observar estamos utilizando la IP/Host 127.0.0.1 ya que sería nuestro localhost y también utilizamos el puerto 3306 que es el que estaba usando el MySQL que teníamos en ejecución en XAMPP.



Una vez abramos la conexión con HeidiSQL se nos abrirá la siguiente interfaz:



En ella tendremos que abrir una nueva consulta para ejecutar el archivo init.sql del que disponemos.



Copiamos el sql al heidi para que se nos copien las sentencias. Lo primero que ejecutaremos serán estas dos sentencias, tenemos que seleccionarlasy clicar sobre “Ejecutar selección” de esta forma se nos creará la base de datos “apartment 4v” y la seleccionaremos para usarla.

```

-- Volcando estructura de base de datos para apartment4v
CREATE DATABASE IF NOT EXISTS `apartment4v` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;
USE `apartment4v`;

```

Una vez ejecutadas estas dos sentencias ejecutaremos el resto de la query. Y tendremos la base de datos necesaria para continuar con la instalación en local.

Backend/Frontend Local

Backend

Empezaremos con esta parte creando una carpeta en la que iremos almacenando los diferentes archivos de docker que vayamos creando, ya contamos con unos archivos .zip que contienen algunas configuraciones ya hechas pero tendremos que realizar en el Dockerfile y en el archivo .env

Empezaremos realizando la parte del backend, más concretamente el dockerfile, que en mi caso lo tengo de la siguiente forma:

```
# Imagen base de PHP con Apache
FROM php:8.2-apache

# Instala las extensiones necesarias para Symfony
RUN apt-get update && apt-get install -y \

    git \

    unzip \

    libicu-dev \

    libzip-dev \

    libonig-dev \

    mariadb-client \

    && docker-php-ext-install intl pdo pdo_mysql zip opcache

# Instalar Composer
COPY --from=composer:latest /usr/bin/composer /usr/bin/composer

# Configura el directorio de trabajo
```

```

WORKDIR /var/www/html

# Copia los archivos del proyecto al contenedor
COPY . .

# Configura permisos para Symfony
RUN chown -R www-data:www-data var/cache var/log

# Configuración del VirtualHost de Apache
RUN echo "<VirtualHost *:80>\n\
\n\
    DocumentRoot /var/www/html/public\n\
\n\
    <Directory /var/www/html/public>\n\
\n\
        AllowOverride All\n\
\n\
        Require all granted\n\
\n\
        # Reescritura para Symfony (equivalente a .htaccess)\n\
\n\
        <IfModule mod_rewrite.c>\n\
\n\
            RewriteEngine On\n\
\n\
            RewriteCond %{REQUEST_FILENAME} !-f\n\
\n\
            RewriteCond %{REQUEST_FILENAME} !-d\n\
\n\
            RewriteRule ^(.*)$ index.php [QSA,L]\n\
\n\
        </IfModule>\n\
\n\
    </Directory>\n\
\n\
</VirtualHost>" > /etc/apache2/sites-available/000-default.conf

# Habilita el módulo rewrite de Apache para Symfony

```



```
RUN a2enmod rewrite

# Expone el puerto 80

EXPOSE 80

# Comando por defecto para iniciar Apache

CMD ["apache2-foreground"]
```

Una vez realizado el Dockerfile ya tendremos la parte que nos conectara a la base de datos como tal, pero nos falta indicarle la URL de la base de datos a la que se tiene que conectar, y esto lo indicaremos en el archivo .env con el que contamos y modificaremos la linea "DATABASE_URL" añadiendo lo siguiente:

```
DATABASE_URL="mysql://root:4vientos@apartamentos4v-clase.cnza4epefm3p.us-east-1.rds.amazonaws.com:3306/apartment4v?serverVersion=8.0.39&charset=utf8mb4"
```

Una vez hayamos realizado la configuración la ejecutaremos como quien dice, para ello lo primero será ejecutar el comando **docker build -t imagen** .

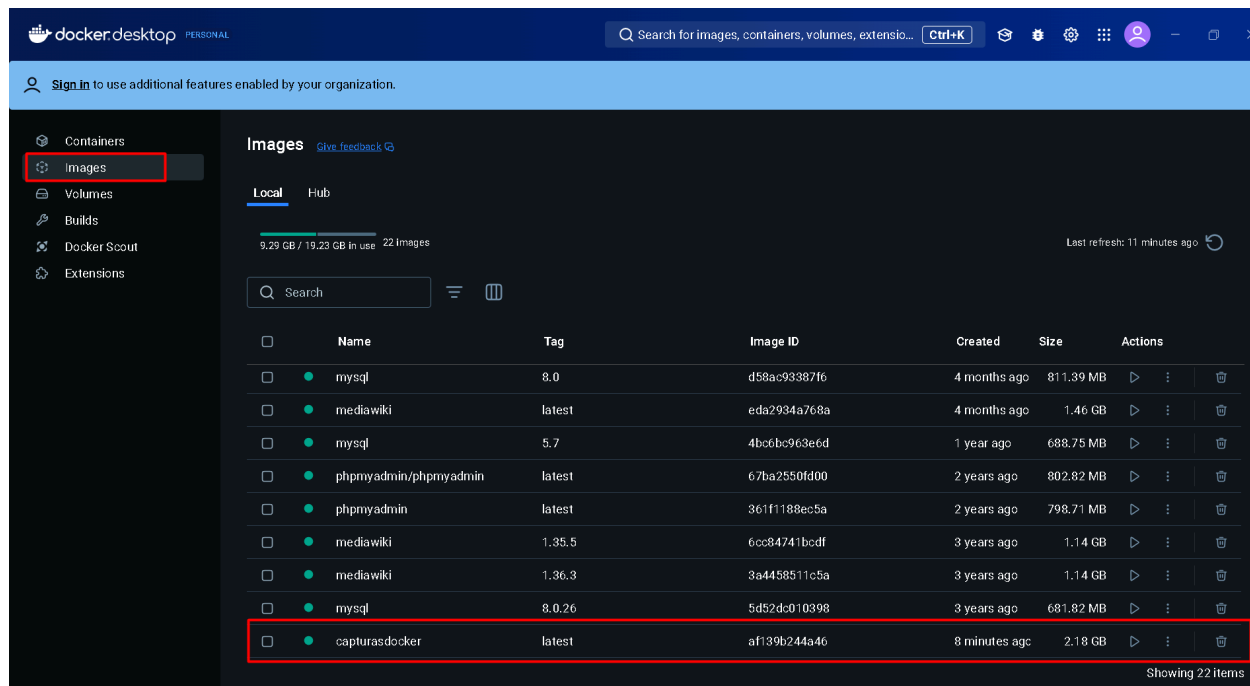
```

C:\Users\jesus\Desktop\Docker\Para2ASIR\Code_Backend>docker build -t capturasdocker .
[+] Building 40.3s (15/15) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 1.46kB                             0.0s
=> [internal] load metadata for docker.io/library/php:8.2-apache  0.0s
=> [internal] load metadata for docker.io/library/composer:latest 1.7s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> [stage-0 1/8] FROM docker.io/library/php:8.2-apache@sha256:95fad7ae0f0b5e60972eff3a2d38902fdcad5d32c7cc3435098f28f39b4 1.1s
=> => resolve docker.io/library/php:8.2-apache@sha256:95fad7ae0f0b5e60972eff3a2d38902fdcad5d32c7cc3435098f28f39b4e3 1.1s
=> FROM docker.io/library/composer:latest@sha256:e0c9ac329256c25b0dee572df37d986570fb26bb6baaa7d0abe69b84181701e1 0.0s
=> => resolve docker.io/library/composer:latest@sha256:e0c9ac329256c25b0dee572df37d986570fb26bb6baaa7d0abe69b84181701e1 0.0s
=> [internal] load build context                                  8.4s
=> => transferring context: 275.29MB                               8.2s
=> CACHED [stage-0 2/8] RUN apt-get update && apt-get install -y  git      unzip      libicu-dev      libzip-dev      libonig  0.0s
=> CACHED [stage-0 3/8] COPY --from=composer:latest /usr/bin/composer /usr/bin/composer 0.0s
=> CACHED [stage-0 4/8] WORKDIR /var/www/html                    0.0s
=> [stage-0 5/8] COPY . .                                        4.3s
=> [stage-0 6/8] RUN chown -R www-data:www-data var/cache var/log 1.7s
=> [stage-0 7/8] RUN echo "<VirtualHost *:80>\n    DocumentRoot /var/www/html/public\n    <Directory /var/www/html/public>\n" 0.5s
=> [stage-0 8/8] RUN a2enmod rewrite                             0.6s
=> exporting to image                                           21.0s
=> => exporting layers                                           18.2s
=> => exporting manifest sha256:6efd1996abc2154001ce080139e9b6ee33f25f0e4e8a05bd6fff2bc12131d048 0.0s
=> => exporting config sha256:69c426748f2299b07803dc3dc65022f7b8f430b3cae3e39803cd02ba9b4a891a 0.0s
=> => exporting attestation manifest sha256:867bc293b0816f610eb937c8c66be49937acedd7f041ce8d1465be2f8cbda07b 0.0s
=> => exporting manifest list sha256:af139b244a46ead56aa1873f8b3ef7806b3ee4368f38c1031bad44d3f6694501 0.0s
=> => naming to docker.io/library/capturasdocker:latest          0.0s
=> => unpacking to docker.io/library/capturasdocker:latest      2.7s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/5imb6hxa2oikynk218h7g85zj

```

Si sale todo correctamente cuando haya terminado podremos ver en la parte de “Images” la imagen que acabamos de construir.



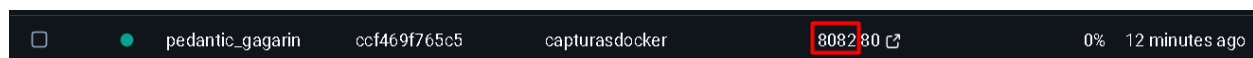
Una vez se haya construido deberemos de ejecutar el segundo comando necesario que es el que sirve para iniciar como tal la imagen creada. Para ello usaremos el comando **“docker run -p 8082:80 imagen”** después de la etiqueta -p yo he puesto el puerto 8082 pero podemos usar el que mejor creamos conveniente si tenemos ocupado el 8082. Yo en mi caso use el 8082 porque tenía ya en uso el puerto 8080 en otra imagen.

```

C:\Users\jesus\Desktop\docker\Para2ASIR\Code_Backend: docker run -p 8082:80 capturasdocker
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this message
[Mon Feb 10 08:45:18.376397 2025] [mpm_prefork:notice] [pid 1:tid 1] AH00163: Apache/2.4.62 (Debian) PHP/8.2.26 configured -- resuming normal operations
[Mon Feb 10 08:45:18.376439 2025] [core:notice] [pid 1:tid 1] AH00094: Command Line: 'apache2 -D FOREGROUND'
[Mon Feb 10 08:46:19.224485 2025] [php:notice] [pid 17:tid 17] [client 172.17.0.1:44192] [error] Uncaught PHP Exception Symfony\Component\HttpKernel\Exception\NotFoundHttpException: "No route found for "GET http://localhost:8082/" at RouterListener.php line 127
172.17.0.3:80 172.17.0.1 - - [10/Feb/2025:08:46:19 +0000] "GET / HTTP/1.1" 404 1301 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36"
172.17.0.3:80 172.17.0.1 - - [10/Feb/2025:08:46:24 +0000] "GET /apartments?apiKey=1234 HTTP/1.1" 200 1458 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36"
[Mon Feb 10 08:46:24.341794 2025] [php:notice] [pid 18:tid 18] [client 172.17.0.1:44200] [error] Uncaught PHP Exception Symfony\Component\HttpKernel\Exception\NotFoundHttpException: "No route found for "GET http://localhost:8082/favicon.ico" (from "http://localhost:8082/apartments?apiKey=1234") at RouterListener.php line 127, referer: http://localhost:8082/apartments?apiKey=1234
172.17.0.3:80 172.17.0.1 - - [10/Feb/2025:08:46:24 +0000] "GET /favicon.ico HTTP/1.1" 404 1300 "http://localhost:8082/apartments?apiKey=1234" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36"
172.17.0.3:80 172.17.0.1 - - [10/Feb/2025:08:47:15 +0000] "-" 408 0 "-" "-"

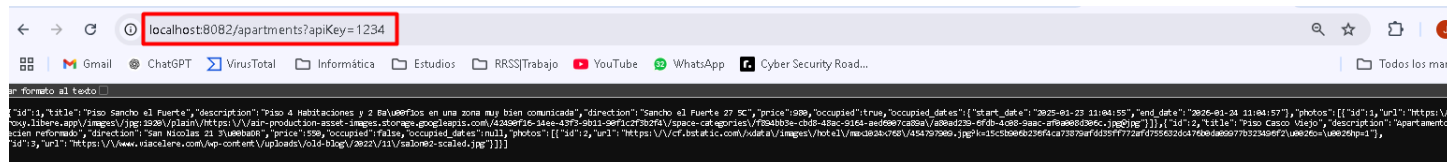
```

Cuando esté ya en ejecución la imagen habremos terminado como tal la instalación y ya podremos ver nuestra imagen en ejecución en el apartado de “Containers” y también podremos observar que está en ejecución en el puerto que hemos usado para hacer el docker run.



Comprobaremos el correcto funcionamiento de nuestra imagen ingresando en la URL de nuestro navegador web lo siguiente:

http://localhost:8082/apartments?apiKey=1234



Frontend

Ahora vamos a realizar básicamente lo mismo pero con el frontend por así decirlo. Tendremos que realizar el dockerfile de la misma forma que en el backend pero modificando las líneas que lo contiene para que cuadre con la parte que estamos realizando. El dockerfile que tengo contiene lo siguiente:

```
FROM ubuntu:20.04

# Update repositories index

RUN apt update --fix-missing && apt upgrade -y

# Install util tools

RUN ln -snf /usr/share/zoneinfo/UTC /etc/localtime && echo "UTC" >
/etc/timezone

RUN apt install -y apt-utils git rsync nano vim unzip curl wget
software-properties-common mysql-client

# Install Apache

RUN apt-get install -y apache2

# Config Apache

ENV APACHE_RUN_USER=www-data

ENV APACHE_RUN_GROUP=www-data

ENV APACHE_LOG_DIR=/var/log/apache2

RUN chown www-data:www-data -R /var/www

RUN chmod -R 755 /var/www

RUN a2enmod rewrite headers

RUN service apache2 start
```

```

WORKDIR /var/www/html

COPY . /var/www/html

EXPOSE 80

# Ejecutar Apache2 en primer plano

CMD ["apache2ctl", "-D", "FOREGROUND"]

```

Una vez hecho el dockerfile tendremos que editar el archivo .js que tenemos. En el realizaremos Ctrl + F y buscaremos "<http://174.129>" y nos aparecerá una o varias líneas en las que cambiaremos la IP por esto:

```

Pn("",o.apartamento.price," \u20AC/Mes"),G(),oe("ngIf",o.apartamento.occupied),G(),oe("ngIf",!o.apartamento.
occupied)))},dependencies:[Ct,Yr,dL,Vl,fw,YD,nm],styles:[".apartamento-card[_ngcontent-%COMP%]{text-align:center;
border:1px solid #ccc;border-radius:0;overflow:hidden;margin:10px;box-shadow:0 4px 8px #0000001a;position:relative}.
card-img[_ngcontent-%COMP%]{width:100%;max-height:300px;border-bottom:1px solid #ccc}.card-content[_ngcontent-%COMP%]
{padding:15px}.card-title[_ngcontent-%COMP%]{font-size:1.25rem;margin-bottom:10px;font-weight:700}.card-description
[_ngcontent-%COMP%]{font-size:1rem;margin-bottom:10px}.card-info[_ngcontent-%COMP%]{font-size:.875rem;color:#555;
margin-bottom:5px}.card-button[_ngcontent-%COMP%]{display:inline-block;padding:8px 40px;background-color:#007bff;
color:#fff;text-decoration:none;border-radius:20px;font-size:.875rem;transition;background-color .3s}.card-button
[_ngcontent-%COMP%]:hover{background-color:#0056b3}.card-price[_ngcontent-%COMP%]{position:absolute;top:10px;
right:10px}.card-price[_ngcontent-%COMP%] span[_ngcontent-%COMP%]{font-size:1.25rem;color:red;font-weight:700}.
card-content[_ngcontent-%COMP%] hr[_ngcontent-%COMP%]{margin:0 0 10px;border:none;height:2px;background-color:#ccc;
font-weight:700}.card-occupied[_ngcontent-%COMP%]{font-weight:700;font-size:1rem}.picsum-img-wrapper
[_ngcontent-%COMP%]{width:100%;height:300px;margin-bottom:10px;object-fit:contain}};let t=e;return t})();var vd=
((>=>{let e=class e{constructor(n){this.http=n,this.apiUrl="http://localhost:8080/apartments?apiKey=1234"}
getApartments(){return this.http.get(this.apiUrl)};e.\u0275fac=function(r){return new(r|e)(m(q1));e.\u0275prov=v
({token:e,factory:e.\u0275fac,providedIn:"root"});let t=e;return t})();function dP(t,e){if(t&1&&(Qc(0),S(1,"div",6),
pe(2,"app-apartamento-card",7),I(),Zc()),t&2){let i=e.$implicit;G(2),oe("apartamento",i)}var tC=t=>({selected:t}),
nC=()=>=>{let e=class e{constructor(n){this.apartamentosService=n,this.apartamentos=[],this.apartamentosFiltrados=[],
this.filtro="Todos",this.fechaFiltro=null,this.filtroSeleccionado=""}.ngOnInit(){this.getApartments()}getApartments()
{this.apartamentosService.getApartments().subscribe(n=>{this.apartamentos=n,this.filterApartments()})}setFilter(n)
{this.filtroSeleccionado=n,this.filtro=n,this.filterApartments()}setDateFilter(n){let r=n.target;this.
fechaFiltro=new Date(r.value),this.filterApartments()}filterApartments(){this.apartamentosFiltrados=this.
apartamentos.filter(n=>!(this.filtro==="Libres"&&n.occupied||this.fechaFiltro&&new Date(n.occupied_dates.end_date)
<this.fechaFiltro))};e.\u0275fac=function(r){return new(r|e)(b(vd))};e.\u0275cmp=ie({type:e,selectors:
[[["app-apartamentos"],standalone:!0,features:[re],decls:9,vars:7,consts:[["tags"],[1,"filter-tags"],[1,"tag",3,
"ngClass","click"],["type","date","id","date","name","date","value","2024-01-12",1,"tag",3,"change"],[1,

```

Una vez ya realizadas estas dos partes procederemos a ejecutar los dos comandos que ejecutamos anteriormente pero con el nombre de la imagen actual:

docker build -t imagen

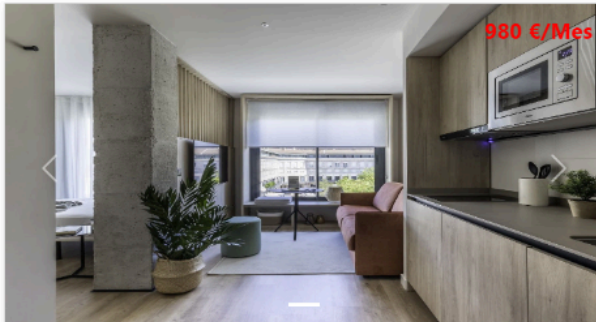
docker run -p 8083:80 imagen

Comprobamos el correcto funcionamiento accediendo a la siguiente URL desde nuestro navegador: localhost:8083/apartments?apiKey=1234.

Libres

Todos

12/01/2024

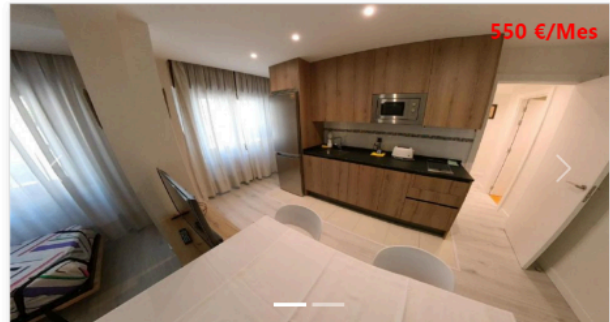


Piso Sancho el Fuerte

Piso 4 Habitaciones y 2 Baños en una zona muy bien comunicada

Sancho el Fuerte 27 5C

Hasta 2026-01-24 11:04:57



Piso Casco Viejo

Apartamento 1 Hab recién reformado

San Nicolas 21 3ºDR

Reserva

2024 By Cuatrovientos

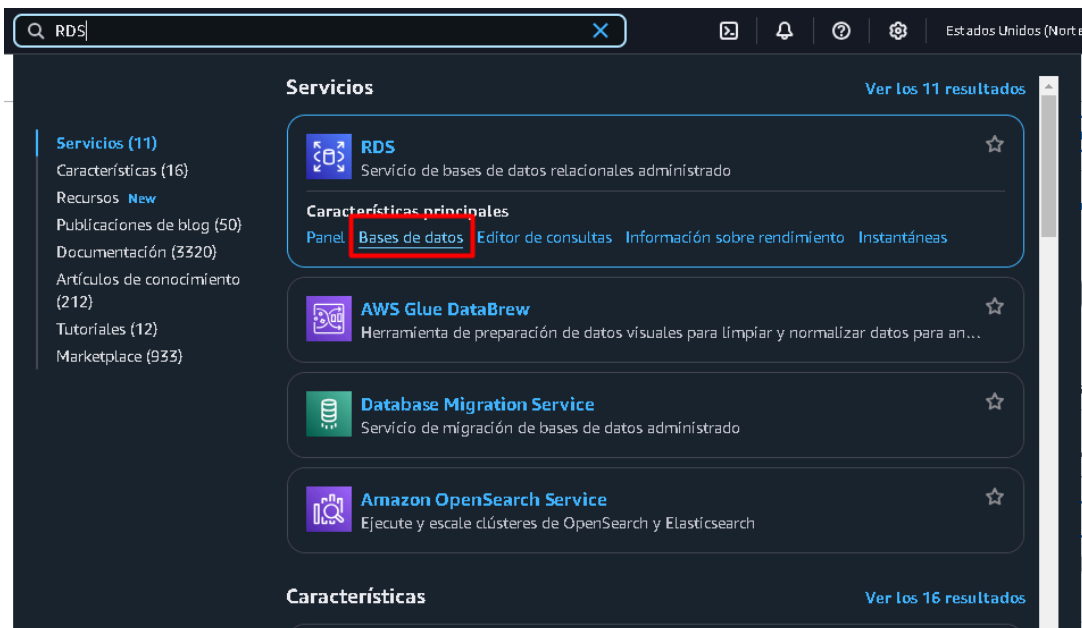
Como podemos observar al entrar en la url nos tiene que salir de la siguiente manera.

Instalación en nube (AWS)

Ahora que ya tenemos realizada la parte en local podemos empezar a realizarlo en la nube ya que el planteamiento es básicamente el mismo pero con los servicios que nos proporciona AWS.

RDS

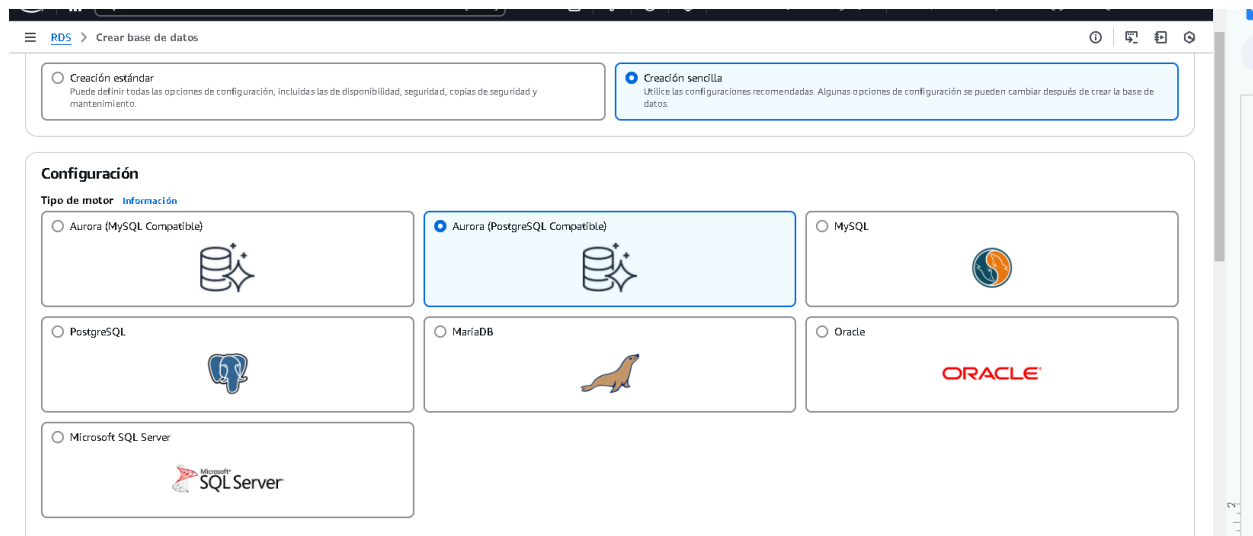
Comenzaremos iniciando la instalación en AWS para crear la base de datos en RDS, para empezar tendremos que iniciar sesión en la consola de AWS y buscaremos en la barra de búsqueda “RDS” y seleccionamos la opción “Bases de datos”



Una vez abierto el apartado de bases de datos ya podremos crear una nueva base de datos clicando sobre “Crear base de datos”



Una vez ya dentro del menú de creación de BBDD que tiene la siguiente estética:



Escogeremos las siguientes opciones de creación para la base de datos:

- Modo de creación: Sencilla
- Motor de base de datos: MySQL
- Capa: Gratuita
- Identificador de instancia: “nombredetuinstancia”
- Nombre de usuario maestro: root
- Administración: autoadministrador
- Contraseña: X

Ya escogidas estas opciones haremos clic sobre “Crear base de datos” y esperaremos a que la instancia se cree.

Ya creada la instancia lo que haremos será realizar algunas configuraciones de seguridad en AWS RDS, para ello empezaremos haciendo clic sobre el nombre de la instancia que acabamos de crear:

Bases de datos (1) Recursos del grupo Modificar Acciones Restaurar desde S3 Crear base de da

<input type="checkbox"/>	Identificador de base de datos	Estado	Rol	Motor	Región ...	Tamaño	Recomendaciones
<input type="radio"/>	apartamentos4v-clase	Disponible	Instancia	MySQL Co...	us-east-1c	db.t4g.micro	2 Informativo

apartamentos4v-clase

Modificar Acciones

Resumen

Identificador de base de datos apartamentos4v-clase	Estado Disponible	Rol Instancia	Motor MySQL Community	Recomendaciones 2 Informativo
CPU 3.78%	Clase db.t4g.micro	Actividad actual 0 Conexiones	Región y AZ us-east-1c	

[Conectividad y seguridad](#) [Supervisión](#) [Registros y eventos](#) [Configuración](#) [Integraciones sin extracción, transformación y carga \(ETL\)](#)

Conectividad y seguridad

Punto de enlace y puerto Punto de enlace apartamentos4v-clase.cnza4epfm3pu s-east-1.rds.amazonaws.com Puerto 3306	Redes Zona de disponibilidad us-east-1c VPC vpc-03338e02a11b3a50a Grupo de subredes default-vpc-03338e02a11b3a50a Subredes subnet-039e4fedabf9c1309 subnet-fb121712a6c1b9a0	Seguridad Grupos de seguridad de la VPC default (sg-0f2e088d3b9e31060) Activo Accesible públicamente Sí Entidad de certificación rds-ca-rsa2048-g1 Fecha de la entidad de certificación Mar 26, 2021 01:24:11 UTC+02:00
---	---	---

Una vez dentro de la instancia clicamos sobre “Grupos de seguridad de la VPC”, a continuación clicamos sobre el “ID de grupo de seguridad”

Grupos de seguridad (1) Información Acciones Exportar los grupos de seguridad a CSV Crear grupo de seguridad

Clear filters

<input type="checkbox"/>	Name	ID de grupo de seguridad	Nombre del grupo de seguridad	ID de la VPC	Descripción
<input type="checkbox"/>	-	sg-0f2e088d3b9e31060	default	vpc-03338e02a11b3a50a	default VPC security group

sg-0f2e088d3b9e31060 - default

Acciones ▾

Detalles

Nombre del grupo de seguridad
default

ID del grupo de seguridad
sg-0f2e088d3b9e31060

Descripción
default VPC security group

ID de la VPC
vpc-03338e02a11b3a50a

Propietario
595316730222

Número de reglas de entrada
2 Entradas de permisos

Número de reglas de salida
1 Entrada de permiso

Reglas de entrada

Reglas de salida

Compartiendo : *novedad*

Asociaciones de VPC : *novedad*

Etiquetas

Reglas de entrada (2)

Buscar



Administrar etiquetas

Editar reglas de entrada

<input type="checkbox"/>	Name	ID de la regla del g...	Versión de IP	Tipo	Protocolo	Intervalo de puertos	O
<input type="checkbox"/>	-	sgr-0f097042161a35d1b	IPv4	MySQL/Aurora	TCP	3306	0.
<input type="checkbox"/>	-	sgr-07b86cc707145cf6e	-	Todo el tráfico	Todo	Todo	sg

Ahora estaríamos ya sobre el grupo de seguridad en el que podremos editar las reglas de entrada. Clicamos en “Editar reglas de entrada” e ingresamos las siguientes reglas:

Editar reglas de entrada

Información

Las reglas de entrada controlan el tráfico entrante que puede llegar a la instancia.

Reglas de entrada

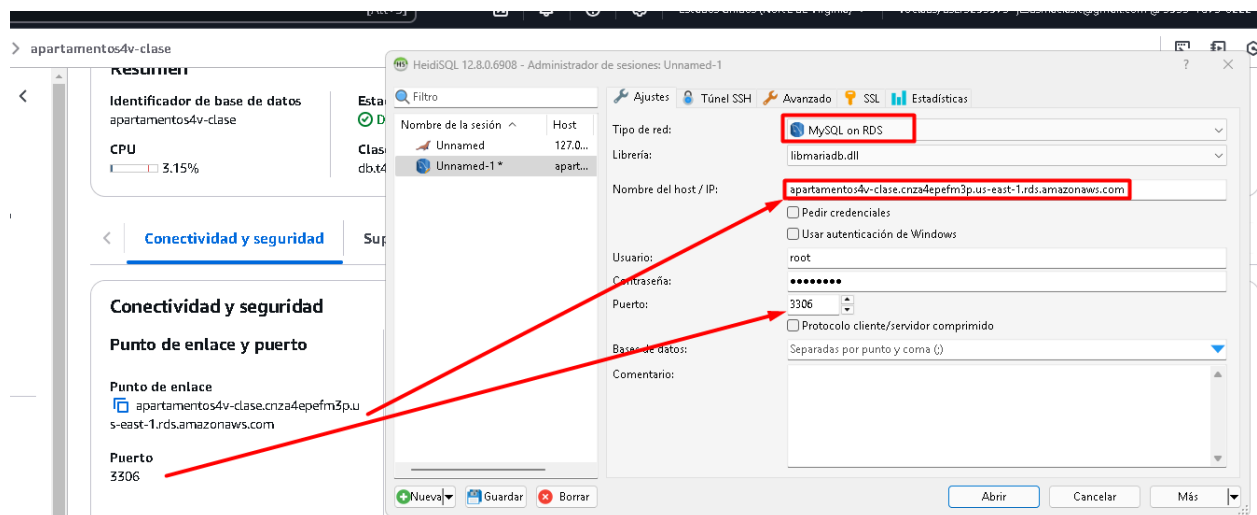
Información

ID de la regla del grupo de seguridad	Tipo	Protocolo	Intervalo de puertos	Origen	Descripción: opcional	
sgr-0f097042161a35d1b	MySQL/Aurora	TCP	3306	Person...		Eliminar
sgr-07b86cc707145cf6e	Todo el tráfico	Todo	Todo	Person...		Eliminar

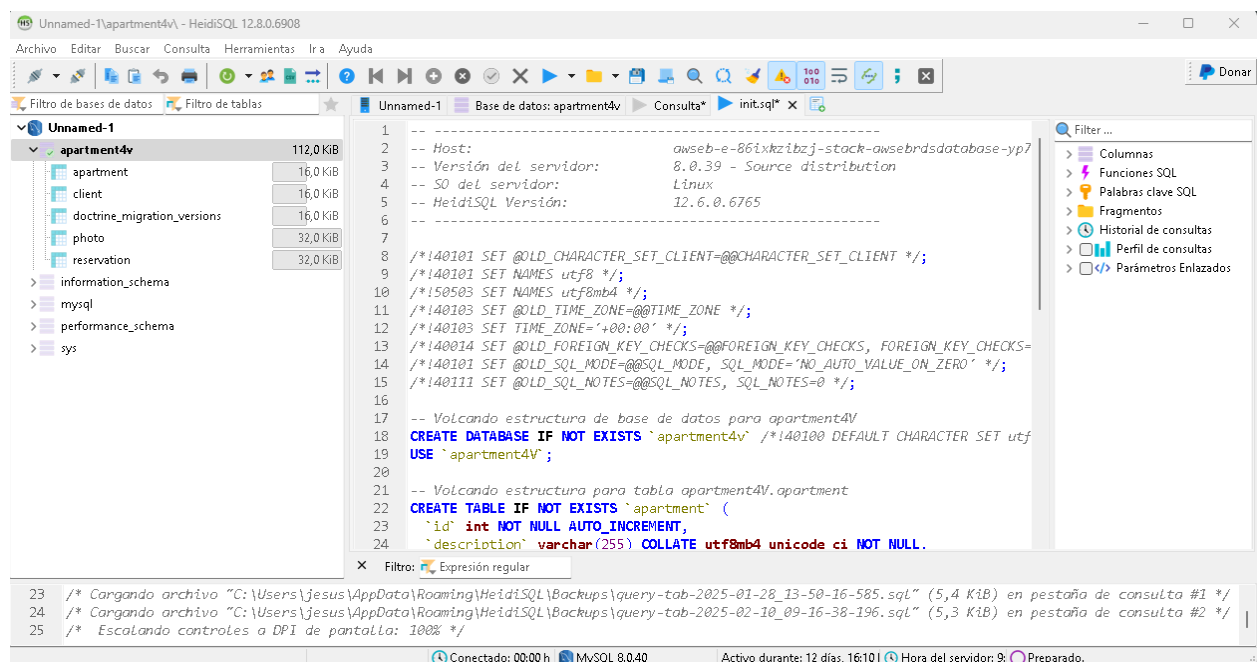
Agregar regla

⚠ Las reglas cuyo origen es 0.0.0.0/0 o ::/0 permiten a todas las direcciones IP acceder a la instancia. Recomendamos configurar reglas de grupo de seguridad para permitir el acceso únicamente desde direcciones IP conocidas.

Una de ellas permitirá la conexión de tipo MySQL/Aurora y otra todo el tráfico en general para permitir que todo el mundo se pueda conectar. Cuando ya tenemos todo esto configurado podremos probar la conexión a la base de datos.



Simplemente será como conectarnos en local pero cambiando el tipo de red y la IP. Y obviamente indicando el usuario y contraseña que hayamos introducido. Una vez nos conectamos tendremos la misma interfaz que anteriormente desde el modo local.

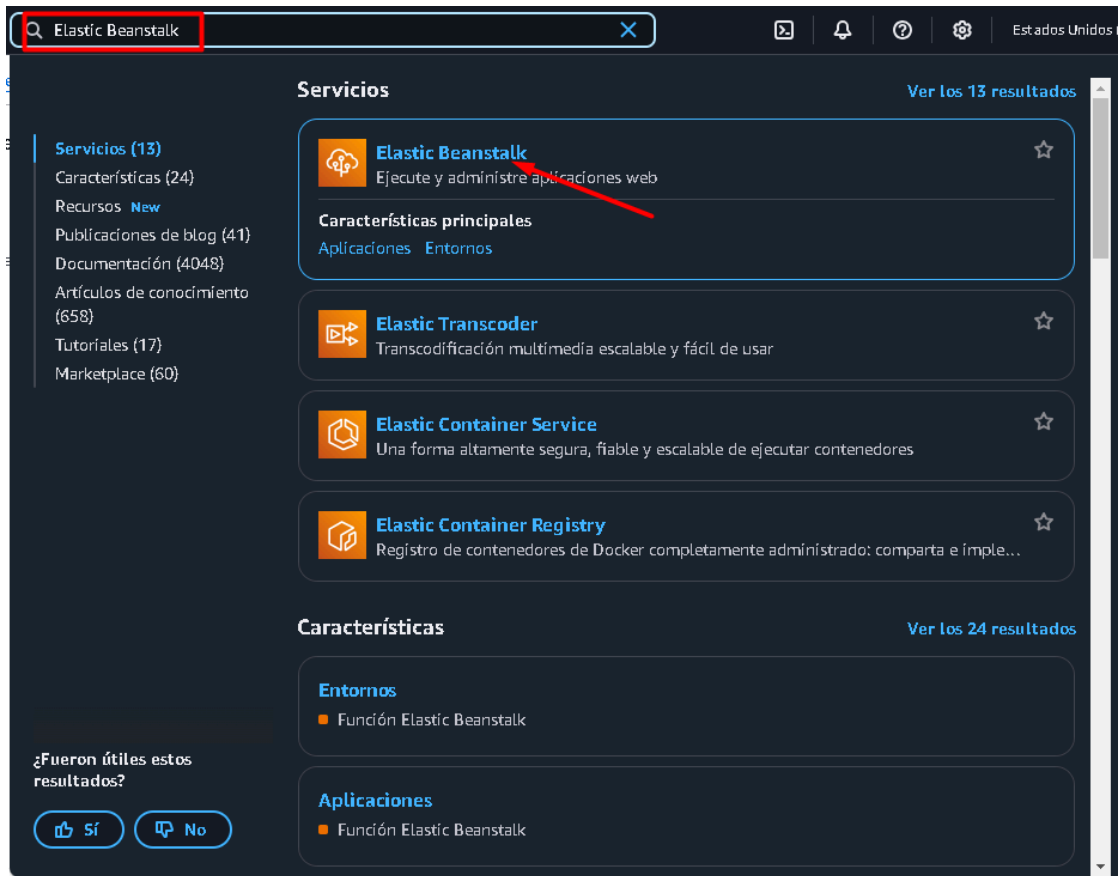


Simplemente tendremos que crear ahora en la nube nuestra base de datos de la misma forma que anteriormente, creando primero la base de datos y luego ejecutando las queries necesarias para crear el contenido que tenemos el archivo init.sql

EBS

Backend

A continuación vamos a empezar con Elastic Beanstalk (EBS) y crearemos el entorno, lo primero que vamos a hacer será buscar el servicio.



Cuando cliqueamos sobre EBS se nos abrirá la siguiente pestaña en la que tendremos que darle a "Creación de entorno"

Entornos (1) [Información](#)



Acciones ▾

Creación de entorno

< 1 > ⚙

Nombre del entorno ▲ Estado ▾ Nomb... ▾ Platafo... ▾ Dominio

▾ Versio... ▾ Nomb... ▾ Fecha ... ▾ Última ... ▾

Cuando se nos abra la configuración del entorno solo tendremos que modificar el nombre de la aplicación indicando el nombre que deseemos, el nivel de entorno elegiremos “entorno de servidor web” y en plataforma “docker”.

Más abajo en este apartado tendremos que configurar el código de aplicación el cual tendremos que cargarlo desde un archivo local con la etiqueta de versión 0.1.

Código de aplicación [Información](#)

☐ Aplicación de ejemplo

☐ Versión existente
Versiones de la aplicación que ha cargado.

☒ Cargar el código
Cargue un paquete de código fuente desde su equipo o copie uno desde Amazon S3.

Etiqueta de versión
Nombre único para esta versión del código de la aplicación.

Origen del código fuente. Tamaño máximo de 500 MB

☒ Archivo local

Cargar aplicación

☒ Nombre del archivo: **Code_Backend.zip**
El archivo debe tener un tamaño máximo de archivo inferior a 500 MB

☐ URL pública de S3

Ya habremos configurado todo lo necesario y ahora tendremos que ir al siguiente paso que es la configuración del acceso al servidor, esta parte la dejaremos como sigue:

Configuración del acceso al servicio [Información](#)

Acceso al servicio

Los roles de IAM, asumidos por Elastic Beanstalk como rol de servicio, y los perfiles de instancia de EC2 permiten a Elastic Beanstalk crear y administrar su instancia deben estar asociados a políticas administradas de IAM que contengan los permisos necesarios. [Más información](#)

Rol de servicio

- ☐ Crear y utilizar un nuevo rol de servicio
- ☒ Usar un rol de servicio existente

Roles de servicio existentes

Elija un rol de IAM existente para que Elastic Beanstalk asuma como rol de servicio. El rol de IAM existente debe tener las políticas administradas de IAM necesarias.

LabRole 

Par de claves de EC2

Seleccione un par de claves de EC2 para iniciar sesión de forma segura en sus instancias de EC2. [Más información](#)

vockey 

Perfil de instancia de EC2

Elija un perfil de instancia de IAM con políticas administradas que permitan a las instancias de EC2 realizar las operaciones necesarias.

LabInstanceProfile 

[Ver los detalles de los permisos](#)

Cuando hayamos finalizado nos vamos a revisión clicando en la opción “Ir a revisión”

[Cancelar](#)

[Ir a revisión](#)

[Anterior](#)

[~ Siguiente](#)

Una vez completado esto clicamos en “Ir a revisión”. Por último si no ha habido ningún problema ya podremos dar clic en “Enviar” y habremos finalizado el proceso.

Al terminar el proceso ya podremos comprobar el correcto funcionamiento simplemente clicando sobre “Ir al entorno”

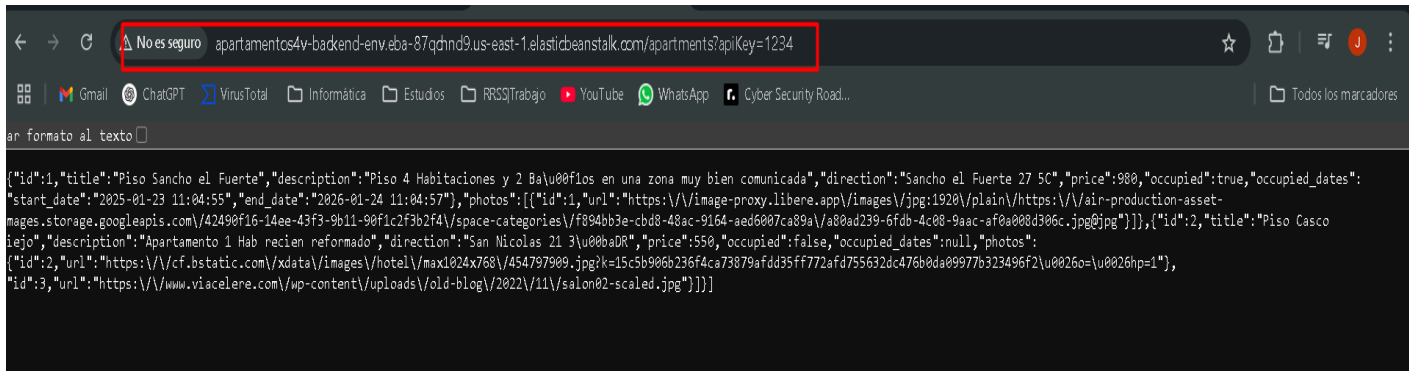
The screenshot shows the AWS Elastic Beanstalk console. The left sidebar has a menu with 'Entornos' selected. Under 'Entornos', there is a list of environments. The environment 'Apartamentos4v-BACKEND-env' is highlighted, and a red arrow points to the 'Ir al entorno' link next to it. The main content area shows the 'Información general del entorno' for 'Apartamentos4v-BACKEND-env'. It includes details like 'Estado' (Ok), 'ID del entorno' (e-2t2q6zrpx), 'Dominio' (Apartamentos4v-BACKEND-env.eba-87qhnd9.us-east-1.elasticbeanstalk.com), and 'Nombre de apli' (Apartamentos4v). Below this, there are tabs for 'Eventos', 'Estado', 'Registros', 'Monitoreo', and 'Alarmas'. The 'Eventos' tab is active, showing a list of 11 events with columns for 'Hora' and 'Tipo'.

Al hacer eso se nos abrirá una pestaña de la siguiente forma:

The screenshot shows a web browser window with the address bar displaying 'apartamentos4v-backend-env.eba-87qhnd9.us-east-1.elasticbeanstalk.com'. The browser's address bar shows 'No es seguro'. The main content area displays a red error message: 'Oops! An Error Occurred'. Below this, it says 'The server returned a "404 Not Found".' and provides a message: 'Something is broken. Please let us know what you were doing when this error occurred. We will fix it as soon as possible. Sorry for any inconvenience caused.'

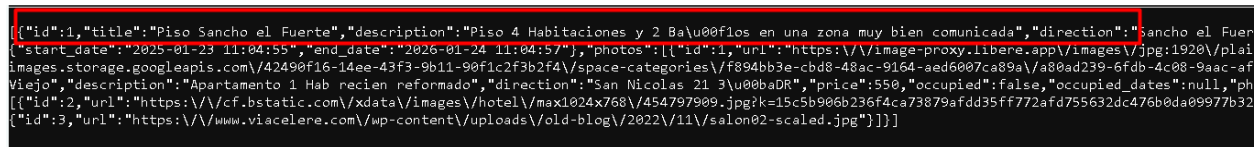
Pero no hay ningún problema ya que no tenemos configurado el frontend y obviamente no va a mostrar nada. Simplemente un error 404 de que no ha encontrado nada, para comprobar el correcto funcionamiento tendremos que añadir al final “/apartments?apiKey=1234”, al añadir

esto al final de la URL el resultado de la búsqueda si hemos realizado todo correctamente debe ser el siguiente:



```
{
  "id": 1,
  "title": "Piso Sancho el Fuerte",
  "description": "Piso 4 Habitaciones y 2 Ba\u00f1os en una zona muy bien comunicada",
  "direction": "Sancho el Fuerte 27 5C",
  "price": 980,
  "occupied": true,
  "occupied_dates": {
    "start_date": "2025-01-23 11:04:55",
    "end_date": "2026-01-24 11:04:57"
  },
  "photos": [
    {
      "id": 1,
      "url": "https://image-proxy.libere.app/images/jpg:1920/plain/https://air-production-asset-images.storage.googleapis.com/42490f16-14ee-43f3-9b11-90f1c2f3b2f4/space-categories/f894bb3e-cbd8-48ac-9164-aed6007ca89a/a80ad239-6fdb-4c08-9aac-af0a000d306c.jpg@jpg"
    },
    {
      "id": 2,
      "title": "Piso Casco Viejo",
      "description": "Apartamento 1 Hab recien reformado",
      "direction": "San Nicolas 21 3\u000baDR",
      "price": 550,
      "occupied": false,
      "occupied_dates": null,
      "photos": [
        {
          "id": 2,
          "url": "https://cf.bstatic.com/xdata/images/hotel/max1024x768/454797909.jpg?k=15c5b906b236f4ca73879afdd35ff772afd755632dc476b0da09977b323496f2\u0026hp=1"
        },
        {
          "id": 3,
          "url": "https://www.viacelene.com/wp-content/uploads/old-blog/2022/11/salon02-scaled.jpg"
        }
      ]
    }
  ]
}
```

A primera vista podr\u00eda parecer que incluso el resultado que nos est\u00e1 dando la visita a la URL es un error pero si nos fijamos podemos ver que son los datos que tenemos introducidos en la tabla de apartments dentro de la base de datos.

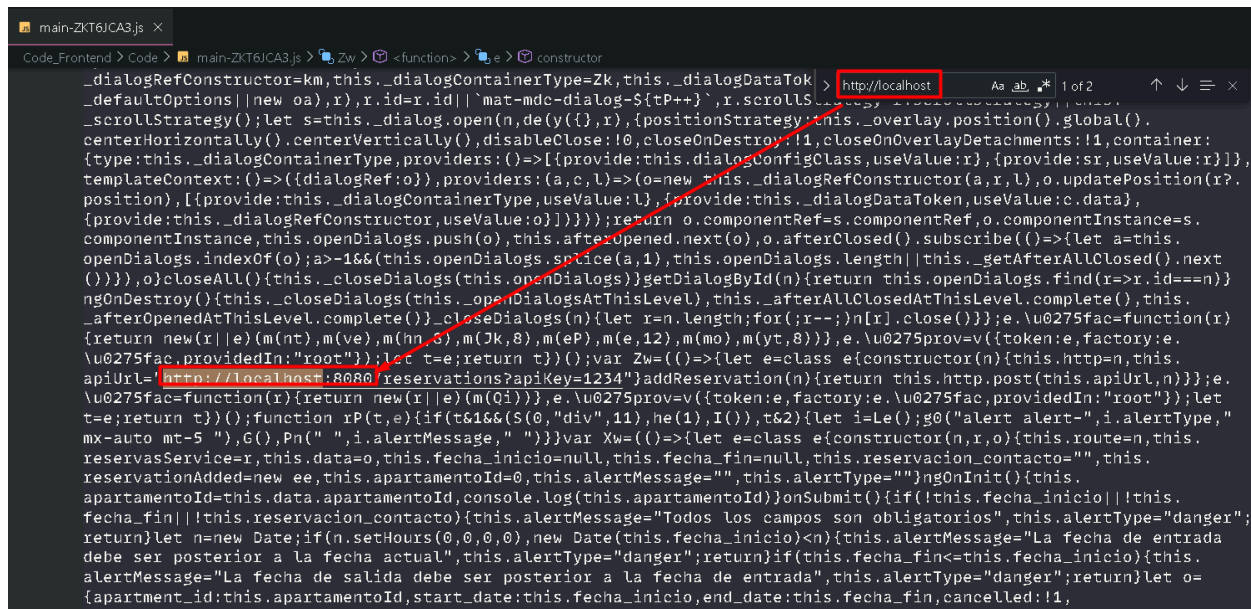


```
{
  "id": 1,
  "title": "Piso Sancho el Fuerte",
  "description": "Piso 4 Habitaciones y 2 Ba\u00f1os en una zona muy bien comunicada",
  "direction": "Sancho el Fuerte 27 5C",
  "price": 980,
  "occupied": true,
  "occupied_dates": {
    "start_date": "2025-01-23 11:04:55",
    "end_date": "2026-01-24 11:04:57"
  },
  "photos": [
    {
      "id": 1,
      "url": "https://image-proxy.libere.app/images/jpg:1920/plain/https://air-production-asset-images.storage.googleapis.com/42490f16-14ee-43f3-9b11-90f1c2f3b2f4/space-categories/f894bb3e-cbd8-48ac-9164-aed6007ca89a/a80ad239-6fdb-4c08-9aac-af0a000d306c.jpg@jpg"
    },
    {
      "id": 2,
      "title": "Piso Casco Viejo",
      "description": "Apartamento 1 Hab recien reformado",
      "direction": "San Nicolas 21 3\u000baDR",
      "price": 550,
      "occupied": false,
      "occupied_dates": null,
      "photos": [
        {
          "id": 2,
          "url": "https://cf.bstatic.com/xdata/images/hotel/max1024x768/454797909.jpg?k=15c5b906b236f4ca73879afdd35ff772afd755632dc476b0da09977b323496f2\u0026hp=1"
        },
        {
          "id": 3,
          "url": "https://www.viacelene.com/wp-content/uploads/old-blog/2022/11/salon02-scaled.jpg"
        }
      ]
    }
  ]
}
```

Frontend

Ya hemos terminado y comprobado el correcto funcionamiento de nuestro backend por lo que ahora vamos a proceder con el frontend. Para comenzar empezaremos modificando el archivo main.js que teníamos ya configurado para realizarlo en local lo único que tendremos que cambiar es la misma parte que configuramos la última vez.

Buscaremos mediante el atajo de teclado Ctrl + F "http://localhost" de este modo nos saldrán las líneas que tenemos que cambiar. Aquí tendremos que poner la URL que tenemos del EBS del Backend.



```
main-ZKT6JCA3.js x
Code_Frontend > Code > main-ZKT6JCA3.js > Ctrl+F > http://localhost > 1 of 2
...
_dialogRefConstructor=km,this._dialogContainerType=Zk,this._dialogDataTok
...
_defaultOptions|new oa(r),r.id=r.id||`mat-mdc-dialog-${tP++}`,r.scrollS
...
_scrollStrategy();let s=this._dialog.open(n,de(y({},r),{positionStrategy:this._overlay.position().global(),
centerHorizontally().centerVertically(),disableClose:!0,closeOnDestroy:!1,closeOnOverlayDetachments:!1,container:
{type:this._dialogContainerType,providers:()=>[provide:this.dialogConfigClass,useValue:r],{provide:sr,useValue:r}}}],
templateContext:()=>({dialogRef:o}),providers:(a,c,l)=>(o=new this._dialogRefConstructor(a,r,l),o.updatePosition(r?.
position),[provide:this._dialogContainerType,useValue:l],{provide:this._dialogDataToken,useValue:c.data},
{provide:this._dialogRefConstructor,useValue:o}}));return o.componentRef=s.componentRef,o.componentInstance=s.
componentInstance,this.openDialogs.push(o),this.afterOpened.next(o),o.afterClosed().subscribe(()=>{let a=this.
openDialogs.indexOf(o);a>-1&&(this.openDialogs.splice(a,1),this.openDialogs.length||this._getAfterAllClosed().next
()))},o.closeAll(){this._closeDialogs(this.openDialogs)}getDialogById(n){return this.openDialogs.find(r=>r.id===n)}
ngOnDestroy(){this._closeDialogs(this._openDialogsAtThisLevel),this._afterAllClosedAtThisLevel.complete(),this.
_afterOpenedAtThisLevel.complete()}_closeDialogs(n){let r=n.length;for(;r--;)n[r].close();e.\u0275fac=function(r)
{return new(r||e)(m(nt),m(ve),m(hn,6),m(7k,8),m(eP),m(e,12),m(mo),m(yt,8))},e.\u0275prov=v({token:e,factory:e.
\u0275fac.providedIn:"root"});let t=e;return t})();var Zw={()=>{let e=class e{constructor(n){this.http=n,this.
apiUrl=`http://localhost:8080/reservations?apiKey=1234`addReservation(n){return this.http.post(this.apiUrl,n)}};e.
\u0275fac=function(r){return new(r||e)(m(Qi))},e.\u0275prov=v({token:e,factory:e.\u0275fac.providedIn:"root"});let
t=e;return t})();function rP(t,e){if(t&16&(S(0,"div",11),he(1),I()),t&2){let i=Le();g0("alert alert-",i.alertType,"
mx-auto mt-5 "),G(),Pn(" ",i.alertMessage," ")}var Xw={()=>{let e=class e{constructor(n,r,o){this.route=n,this.
reservasService=r,this.data=o,this.fecha_inicio=null,this.fecha_fin=null,this.reservacion_contacto="",this.
reservationAdded=new ee,this.apartamentoId=0,this.alertMessage="",this.alertType=""}ngOnInit(){this.
apartamentoId=this.data.apartamentoId,console.log(this.apartamentoId)}onSubmit(){if(!this.fecha_inicio||!this.
fecha_fin||!this.reservacion_contacto){this.alertMessage="Todos los campos son obligatorios",this.alertType="danger";
return}let n=new Date;if(n.setHours(0,0,0),new Date(this.fecha_inicio)<n){this.alertMessage="La fecha de entrada
debe ser posterior a la fecha actual",this.alertType="danger";return}if(this.fecha_fin<this.fecha_inicio){this.
alertMessage="La fecha de salida debe ser posterior a la fecha de entrada",this.alertType="danger";return}let o=
{apartment_id:this.apartamentoId,start_date:this.fecha_inicio,end_date:this.fecha_fin,cancelled:!1,
...
}
```

Podemos copiar la URL desde la que nos metimos para comprobar el backend:

<http://apartamentos4v-backend-env.eba-87qchnd9.us-east-1.elasticbeanstalk.com/>






```
ac,providedIn:"root"});let t=e;return t})();var Zw={()=>{let e=class e{constructor(n){t
:"http://apartamentos4v-backend-env.eba-87qchnd9.us-east-1.elasticbeanstalk.com/reservat
reservation(n){return this.http.post(this.apiUrl,n)}};e.\u0275fac=function(r){return new(r|
rov=v({token:e,factory:e.\u0275fac.providedIn:"root"});let t=e;return t})();function rP
```

Tendremos que realizar el mismo proceso dos veces ya que hay dos líneas diferentes que apuntaban al localhost.

```
...object["url"].concat("}");let url="http://localhost:8083/apartments?apiKey=1234"}getApartments()};re...  
...fac=function(x){return new(x)(x)};...u0275prox=y/{tokenio:f
```

Ya hemos terminado con el cambio de configuración en el archivo js por lo que ahora solo tendremos que comprobar que nuestro Dockerfile esté correctamente configurado y podemos continuar con el proceso, el siguiente paso será comprimir la carpeta Code y el archivo Dockerfile en un archivo .zip para cargarlo después en el entorno de AWS.

 Code	15/02/2025 17:45	Carpeta de archivos	
 Code_Frontend.zip	15/02/2025 17:47	Archivo WinRAR ZIP	214 KB
 Dockerfile	15/02/2025 17:42	Archivo	1 KB

Configuramos a continuación el entorno como tal, el proceso es muy similar al del backend pero incluyendo los archivos del frontend.

Configuración del entorno [Información](#)

Nivel de entorno [Información](#)

Amazon Elastic Beanstalk tiene dos tipos de niveles de entorno para admitir diferentes tipos de aplicaciones web.

☒ Entorno de servidor web

Ejecute un sitio web, una aplicación web o una API web que atienda solicitudes HTTP. [Más información](#)

☐ Entorno de trabajo

Ejecute una aplicación de proceso de trabajo que procese cargas de trabajo de ejecución prolongada bajo demanda o realice tareas de forma programada. [Más información](#)

Información de la aplicación [Información](#)

Nombre de aplicación

Apartamentos4v-FRONTEND

La longitud máxima es de 100 caracteres.

Plataforma [Información](#)

Tipo de plataforma

☒ Plataforma administrada

Plataformas publicadas y mantenidas por Amazon Elastic Beanstalk. [Más info](#)

☐ Plataforma personalizada

Plataformas creadas y de su propiedad. Esta opción no está disponible si no tiene

Plataforma

Docker

Ramificación de la plataforma

Docker running on 64bit Amazon Linux 2023

Versión de la plataforma

4.4.3 (Recommended)

Código de aplicación [Información](#)

☐ Aplicación de ejemplo

☐ Versión existente

Versiones de la aplicación que ha cargado.

☒ Cargar el código

Cargue un paquete de código fuente desde su equipo o copie uno desde Amazon

Etiqueta de versión


Nombre único para esta versión del código de la aplicación.

0.1

Origen del código fuente. Tamaño máximo de 500 MB

☒ Archivo local

Cargar aplicación

 Elegir archivo

☒ Nombre del archivo: **Code_Frontend.zip**

El archivo debe tener un tamaño máximo de archivo inferior a 500 MB

☐ URL pública de S3

Cambiamos los parámetros en el entorno para que coincidan con los que siguen, incluyendo el archivo .zip. También configuraremos los parámetros de acceso al servicio cambiando el rol de servicio existente, el par de claves de EC2 y el perfil de instancia de EC2 que tendrán los mismos parámetros que el EBS de backend. Hecho todo esto hemos terminado con la configuración y podemos darle a “Ir a revisión” y “Enviar”.

Acceso al servicio

Los roles de IAM, asumidos por Elastic Beanstalk como rol de servicio, y los perfiles de instancia de EC2 permiten a Elastic Beanstalk asumir el rol de IAM como el perfil de instancia deben estar asociados a políticas administradas de IAM que contengan los permisos necesarios.

Rol de servicio

- ☐ Crear y utilizar un nuevo rol de servicio
- ☒ Usar un rol de servicio existente

Roles de servicio existentes

Elija un rol de IAM existente para que Elastic Beanstalk asuma como rol de servicio. El rol de IAM existente debe tener las políticas administradas de IAM que contengan los permisos necesarios.

LabRole



Par de claves de EC2

Seleccione un par de claves de EC2 para iniciar sesión de forma segura en sus instancias de EC2. [Más información](#)

vockey



Perfil de instancia de EC2

Elija un perfil de instancia de IAM con políticas administradas que permitan a las instancias de EC2 realizar las operaciones necesarias.

LabInstanceProfile



[Ver los detalles de los permisos](#)

Como siempre al crear un nuevo entorno tardará unos cuantos minutos en crearse correctamente por lo que seremos pacientes y esperaremos hasta que este termine de crearse.

Elastic Beanstalk está lanzando su entorno. Este proceso tardará unos minutos.

Apartamentos4V-FRONTEND-env [Información](#)



[Acciones](#)

Información general del entorno

Estado

Unknown

ID del entorno

e-riu8pia2qx

Dominio

-

Nombre de aplicación

[Apartamentos4V-FRONTEND](#)

Plataforma

Plataforma

Docker running on 64bit Amazon Lin

Ejecución de la versión

-

[Eventos](#)

[Estado](#)

[Registros](#)

[Monitoreo](#)

[Alarmas](#)

[Actualizaciones administradas](#)

[Etiquetas](#)

Eventos (2) [Información](#)

✓ Entorno lanzado correctamente.

Entornos (3) [Información](#)

🔍 *Filtrar entornos*

Nombre del entorno ▲

- ☐ [Apartamentos4v-backend-Clase-env \(terminado\)](#)
- ☐ [Apartamentos4v-BACKEND-env](#)
- ☐ [Apartamentos4V-FRONTEND-env](#)

¡Ya estaría! Ya tenemos nuestros dos entornos creados, el backend y el frontend. Ahora solo nos queda comprobar el correcto funcionamiento del frontend para ello:

Nos dirigimos al entorno del frontend y en la información general del mismo podemos encontrar el dominio.

Apartamentos4V-FRONTEND-env Información

Acciones Cargar e implementar

Información general del entorno

Estado
Ok

ID del entorno
e-riu8pia2qx

Nombre de aplicación
Apartamentos4V-FRONTEND

Dominio
Apartamentos4V-FRONTEND-env.eba-2pfseert.us-east-1.elasticbeanstalk.com

Plataforma
Cambiar la versión

Plataforma
Docker running on 64bit Amazon Linux 2023/4.4.3

Ejecución de la versión
0.1

Estado de la plataforma
Supported

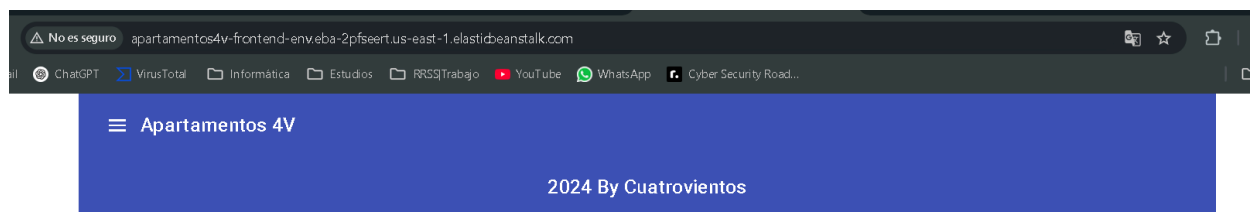
Eventos Estado Registros Monitoreo Alarmas Actualizaciones administradas Etiquetas

Eventos (11) Información

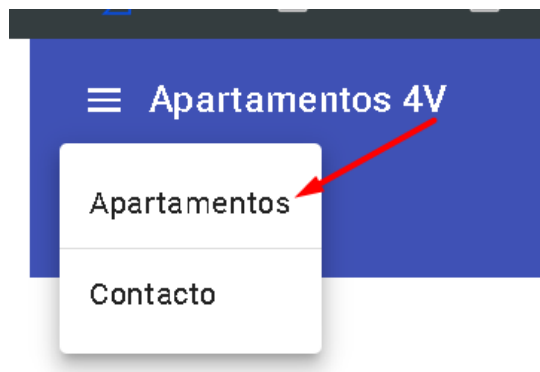
Q Filtrar eventos por texto, propiedad o valor

Hora	Tipo	Detalles
Febrero 15, 2025 18:05:19 (UTC+1)	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 34 seconds ago and took 3 minutes.
Febrero 15, 2025 18:05:13 (UTC+1)	INFO	Successfully launched environment: Apartamentos4V-FRONTEND-env
Febrero 15, 2025 18:05:11 (UTC+1)	INFO	Application available at Apartamentos4V-FRONTEND-env.eba-2pfseert.us-east-1.elasticbeanstalk.com

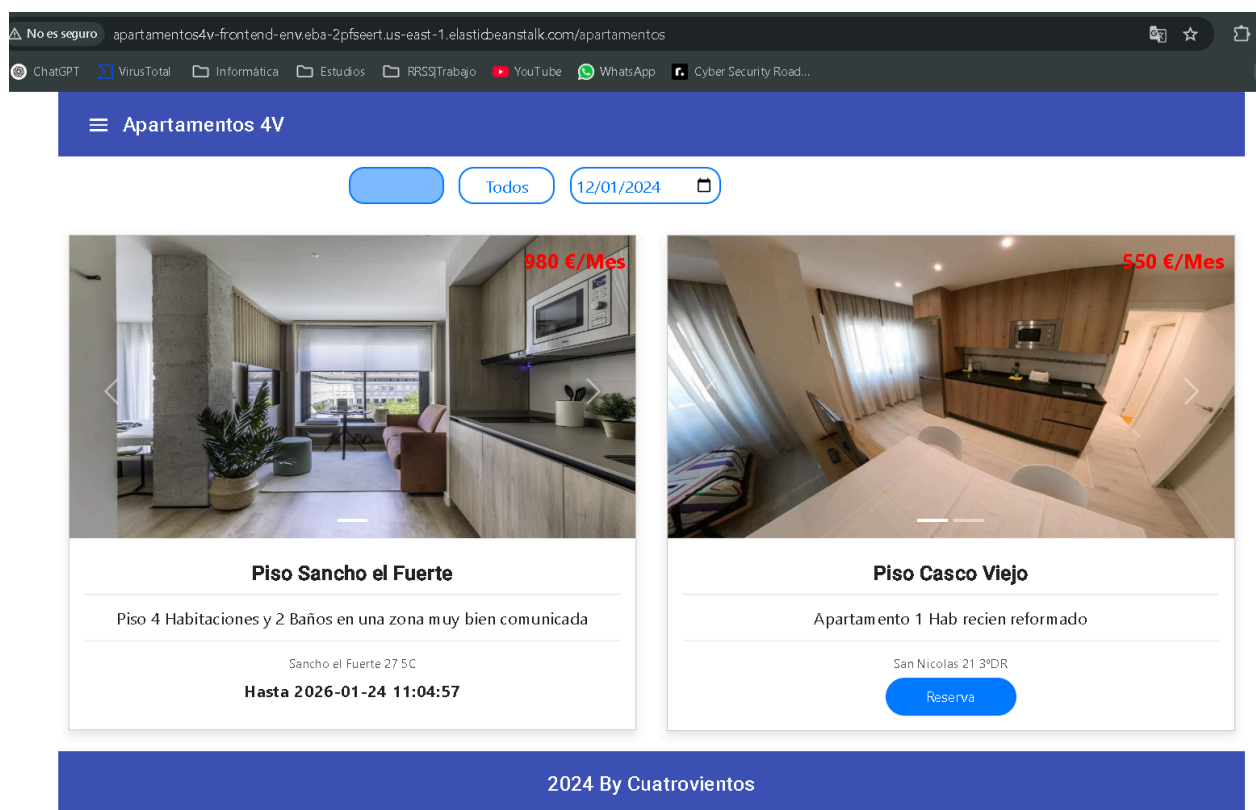
Simplemente clicamos sobre el dominio y nos aparecera la pagina web:



Esto nos confirmará el correcto funcionamiento del frontend como tal, pero ahora tenemos que comprobar que el frontend coja correctamente mediante el js los datos del backend.



Clicamos en Apartamentos dentro del menú de la web.



Y ya estaría, nos devuelve los pisos correspondientes que tenemos en la base de datos del backend. Por lo que esto nos confirma la correcta conexión del archivo js que hemos configurado con el backend creado en AWS.