08

Fall

**IMPLANTACIÓN DE APLICACIONES WEB**

**Apartamentos AWS**

***Alvaro S-O***

CONTENIDO

[1. Fase Local (Docker): 2](#_Toc190535013)

[Base de Datos: 2](#_Toc190535014)

[Backend: 4](#_Toc190535015)

[Frontend: 6](#_Toc190535016)

[2. Fase Nube (AWS): 8](#_Toc190535017)

[RDS: 8](#_Toc190535018)

[EBS (Backend): 11](#_Toc190535019)

[EBS (Frontend): 12](#_Toc190535020)

# Fase Local (Docker):

En primer lugar, vamos a configurar todo de forma local usando Docker Desktop para asegurarnos de que funciona antes de lanzarlo desde la página web. Para saltar esta parte pulsa [aquí](#_Fase_Nube_(AWS):).

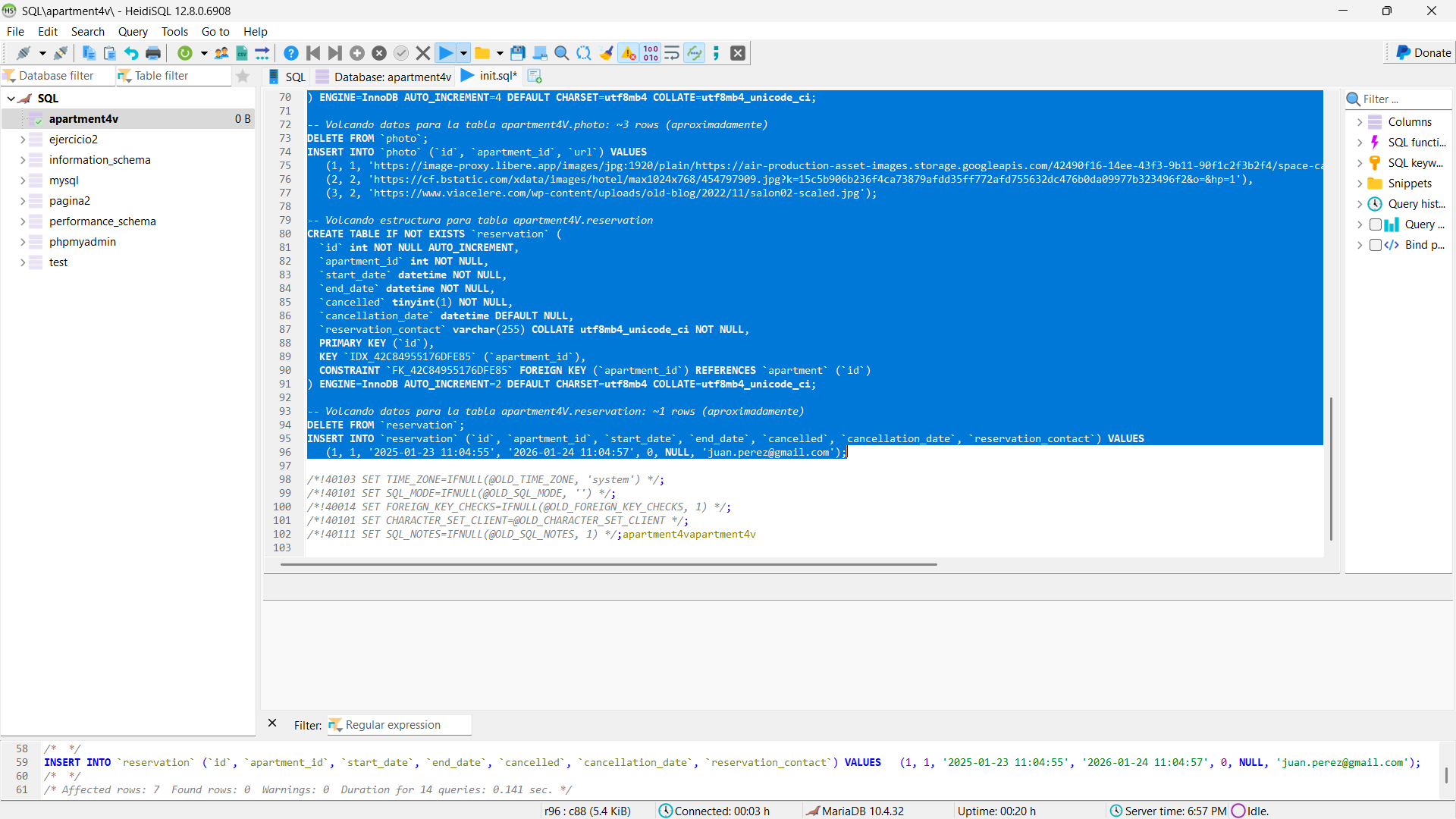
## Base de Datos:

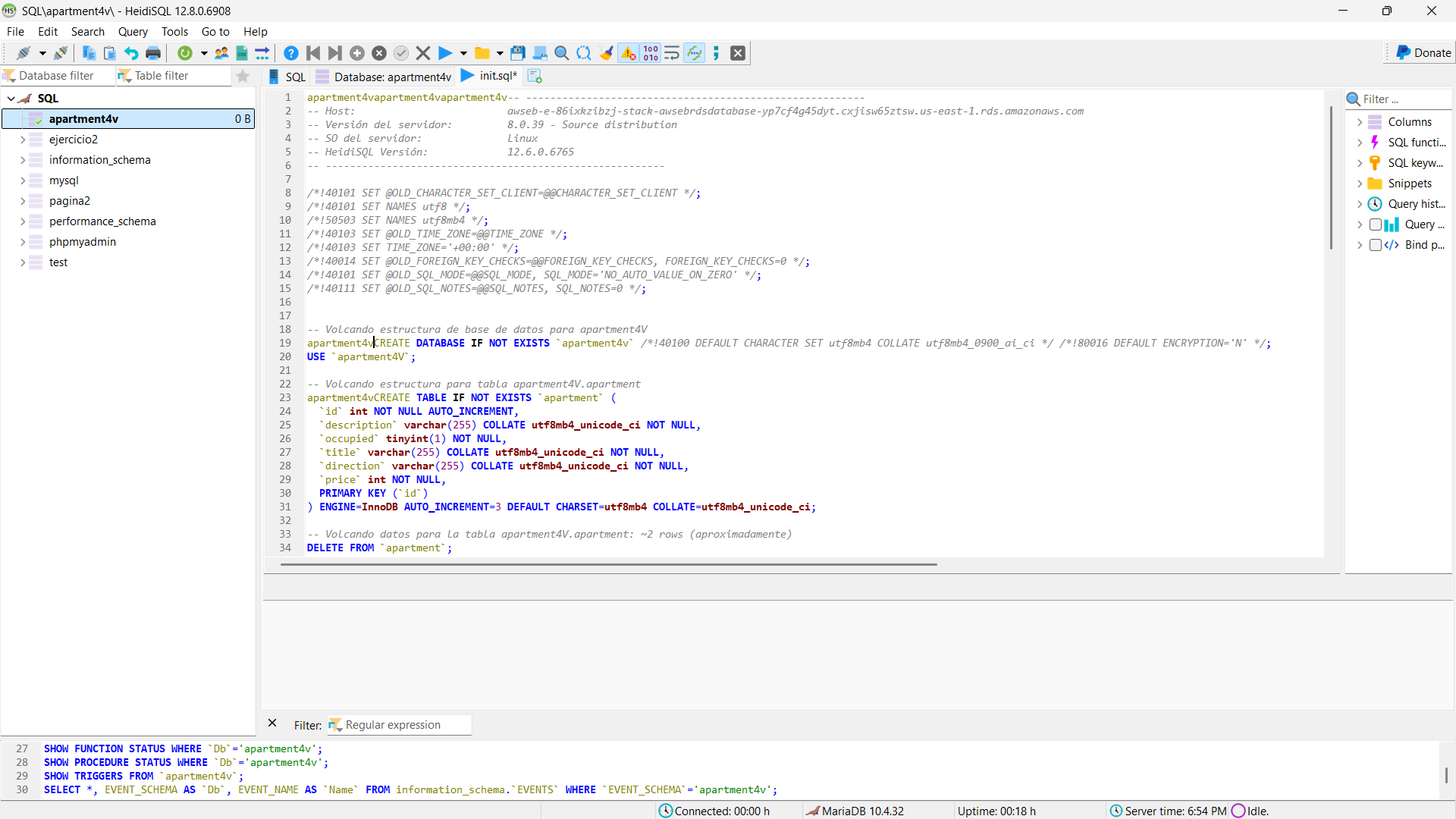
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamentePara esta parte tenemos que crear la base de datos de la que el backend tomará los datos del apartamento. Para ello abriremos el XAMPP y encenderemos tanto el servidor web Apache como el servidor MySQL y nos quedamos con el puerto que esté disponible.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteUna vez dentro, usaremos HeidiSQL para gestionar las bases de datos. Nos conectamos al puerto que vimos en el xampp, y meteremos el usuario y la contraseña que configuramos en su momento.

Ahora abriremos el archivo .sql que teníamos, ejecutaremos la línea de crear la base de datos y seleccionaremos la de apartment4v.

Por lo demás ejecutaremos el resto del sql para que cree las tablas, relaciones y datos.

## Backend:

Ahora haremos la imagen en Docker para que conecte con la base de datos y ejecute todo el backend. El Dockerfile quedó de esta manera:

# 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\

    DocumentRoot /var/www/html/public\n\

    <Directory /var/www/html/public>\n\

        AllowOverride All\n\

        Require all granted\n\

        # Reescritura para Symfony (equivalente a .htaccess)\n\

        <IfModule mod\_rewrite.c>\n\

            RewriteEngine On\n\

            RewriteCond %{REQUEST\_FILENAME} !-f\n\

            RewriteCond %{REQUEST\_FILENAME} !-d\n\

            RewriteRule ^(.\*)$ index.php [QSA,L]\n\

        </IfModule>\n\

    </Directory>\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

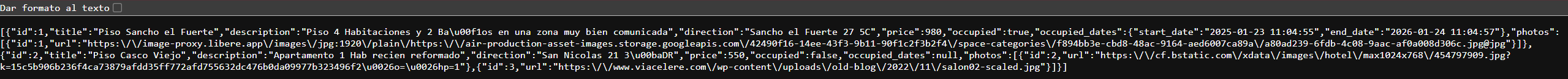
CMD ["apache2-foreground"]

Nos meteremos en el archivo .env en el mismo directorio y editaremos el archivo para que cuando quiera cargar los datos se comunique con el servidor de la base de datos:

DATABASE\_URL="mysql://root@host.docker.internal:3307/apartment4v?serverVersion=8.0.39&charset=utf8mb4

Por último, crearemos la imagen del Apache para Symfony y el contenedor que será nuestro Backend local:

* docker build -t “nombre imagen” .
* docker run -p 8080:80 “nombre imagen”

Para comprobarlo nos meteremos en una ventana del navegador y escribiremos localhost:8080/apartments?apiKey=1234. Esto nos devolverá los datos de la base de datos de nuestros apartamentos.

## Frontend:

Para esta parte solo tenemos que editar el archivo main-ZKT6JCA3.js para que vaya por el puerto que configuramos en el Backend. Buscamos en el archivo una parte que ponga algo como http://174.129.173.204 y lo reemplazamos por esto: <http://localhost:8080>

Por otra parte, el Dockerfile quedó de esta manera:

FROM ubuntu:20.04

# Evitar preguntas interactivas durante la instalación

ENV DEBIAN\_FRONTEND=noninteractive

# Actualizar repositorios e instalar paquetes en una sola instrucción

RUN apt-get update && apt-get install -y \

    apt-utils \

    git \

    rsync \

    nano \

    vim \

    unzip \

    curl \

    wget \

    software-properties-common \

    default-mysql-client \

    apache2 \

 && rm -rf /var/lib/apt/lists/\*

# Configurar zona horaria a UTC

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

# Variables de entorno para Apache

ENV APACHE\_RUN\_USER=www-data

ENV APACHE\_RUN\_GROUP=www-data

ENV APACHE\_LOG\_DIR=/var/log/apache2

# Ajustar permisos en /var/www

RUN chown -R www-data:www-data /var/www && chmod -R 755 /var/www

# Habilitar módulos necesarios de Apache

RUN a2enmod rewrite headers

# Directorio de trabajo y copia de la aplicación

WORKDIR /var/www/html

COPY . /var/www/html

# Exponer el puerto 80

EXPOSE 80

# Iniciar Apache en primer plano al arrancar el contenedor

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

Por último, crearemos la imagen del Apache y el contenedor que será nuestro frontend local:

* docker build -t “nombre imagen” .
* docker run -p 8081:80 “nombre imagen”

Para comprobarlo nos meteremos en una ventana del navegador y escribiremos localhost:8081/apartments?apiKey=1234.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

# Fase Nube (AWS):

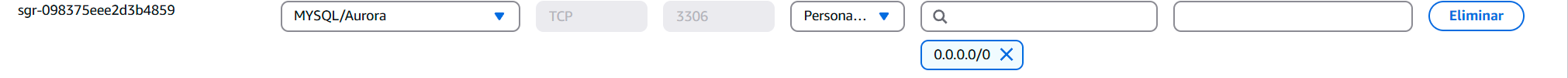
Ahora que ya lo hemos comprobado en local vamos a lanzar nuestro proyecto en la nube usando AWS.

## RDS:

Para esta parte nos iremos a la sección de Relacional Database Service (RDS), crear base de datos y lo configuraremos de la siguiente manera:

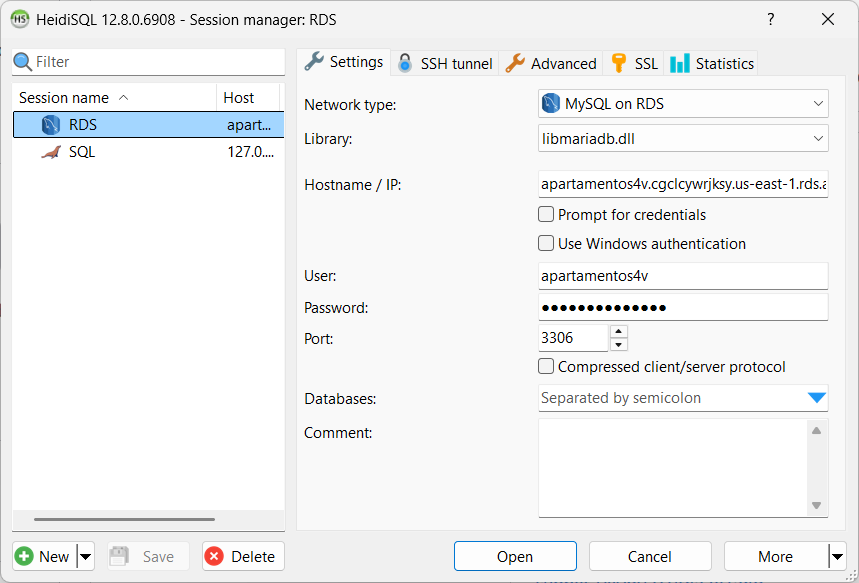
* Elegir un método de creación de base de datos: Creación sencilla
* Configuración:
  + Tipo de motor: Como el Dockerfile del backend se planteó para ser usado con MySQL usaremos MySQL.
  + Tamaño de la instancia de base de datos: Como no va necesitar muchos recursos escogeremos Capa gratuita
  + Identificador de instancias de bases de datos: apartamentos4v
  + Nombre de usuario maestro: apartamentos4v
  + Administración de credenciales: Como no va a ser una base de datos que vaya a tener datos sensibles le daremos a Autoadministrado para crear nuestra propia contraseña
  + Contraseña maestra: apartamentos4v
  + Confirmar la contraseña maestra: apartamentos4v

Para poder conectarnos a la base de datos tenemos que permitir el tráfico del puerto 3306. Para esto nos meteremos en el grupo de seguridad del RDS, Editar reglas de entrada, Agregar regla y configuraremos para que el Tipo sea MYSQL/Aurora y el Origen desde cualquier dirección 0.0.0.0/0. Nos quedaría de la siguiente manera:



Tabla

Descripción generada automáticamente con confianza mediaTambién tenemos que cambiar un parámetro para que sea accesible públicamente. Le damos a modificar, apartado Conectividad, desplegamos la opción de Configuración adicional, seleccionamos la de Accesible públicamente y al continuar le diremos que aplique inmediatamente

Ahora nos meteremos en el HeidiSQL, crearemos una nueva conexión, en el apartado de Hostname / IP pondremos el puerto de enlace que nos da el RDS pondremos los datos que configuramos al crear el RDS

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteUna vez dentro abriremos el archivo .sql que teníamos, ejecutaremos la línea de crear la base de datos y seleccionaremos la de apartment4v.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamentePor lo demás ejecutaremos el resto del sql para que cree las tablas, relaciones y datos.

## EBS (Backend):

Ahora nos iremos a la sección de Elastic Beanstalk (EBS) y crearemos el entorno configurándolo de la siguiente manera:

1. Configuración de entorno:
   * Nivel de entorno 🡪 Entorno de servidor web
   * Información de la aplicación ponemos un nombre 🡪 Apartamentos4v-backend
   * Información del entorno 🡪 por defecto
   * Plataforma:
   * Tipo de plataforma 🡪 por defecto
   * Plataforma 🡪 Docker
   * Ramificación de la plataforma 🡪 por defecto
   * Versión de la plataforma 🡪 por defecto
   * Código de aplicación:
   * Código de aplicación 🡪 Cargar el código
   * Etiqueta de versión ponemos una 🡪 0.1

Ahora tenemos que editar el .env de nuestro código backend para que en vez de comunicar con el host interno del Docker trate con la instancia RDS que configuramos antes. Para ello nos metemos en un editor de texto, abrimos el archivo .env y lo configuramos para que nos quede de la siguiente manera:

DATABASE\_URL="mysql://”Nombre de usuario maestro”:”Contraseña maestra”@”Puerto de enlace RDS”:3306/apartment4v?serverVersion=8.0.39&charset=utf8mb4"

DATABASE\_URL="mysql://apartamentos4v:apartamentos4v@apartamentos4v.cgclcywrjksy.us-east-1.rds.amazonaws.com:3306/apartment4v?serverVersion=8.0.39&charset=utf8mb4"

Comprimimos el código entero del backend en un archivo .zip

* + Eliges archivo local 🡪 Metes el zip
  + Valores preestablecidos 🡪 Instancia única (compatible con la capa gratuita)

1. Acceso al servicio:
   * Roles de servicio existentes 🡪 LabRole
   * Par de claves 🡪 vockey
   * Perfil de instancia de EC2 🡪 LabInstanceProfile
2. Revisión: enviar y esperamos

Captura de pantalla de computadora

Descripción generada automáticamenteUna vez esté el entorno montado, pondremos en una ventana nueva la dirección del dominio de nuestro EBS y añadiremos /apartments?apiKey=1234 como en el backend de la fase de Docker

Y con esto ya tendríamos el EBS del backend de nuestra página de apartamentos

## EBS (Frontend):

Para esta parte repetiremos el proceso que hicimos en el EBS del backend, cambiando lo siguiente:

1. Configuración de entorno:
   * ~~Nivel de entorno 🡪 Entorno de servidor web~~
   * Información de la aplicación ponemos un nombre 🡪 Apartamentos4v-frontend
   * ~~Información del entorno 🡪 por defecto~~
   * ~~Plataforma:~~
   * ~~Tipo de plataforma 🡪 por defecto~~
   * ~~Plataforma 🡪 Docker~~
   * ~~Ramificación de la plataforma 🡪 por defecto~~
   * ~~Versión de la plataforma 🡪 por defecto~~
   * ~~Código de aplicación:~~
   * ~~Código de aplicación 🡪 Cargar el código~~
   * ~~Etiqueta de versión ponemos una 🡪 0.1~~

Para esta parte solo tenemos que editar el archivo main-ZKT6JCA3.js para que se comunique con el EBS del backend. Para ello, nos meteremos en el entorno EBS del backend y copiaremos el dominio. Ahora, en el archivo buscaremos una parte que ponga algo como <http://localhost:8080> y lo reemplazamos por esto: http://”dominio EBS Frontend”

Por último, comprimiremos el código en .zip

* + ~~Eliges archivo local 🡪 Metes el zip~~
  + ~~Valores preestablecidos 🡪 Instancia única (compatible con la capa gratuita)~~

1. ~~Acceso al servicio:~~
   * ~~Roles de servicio existentes 🡪 LabRole~~
   * ~~Par de claves 🡪 vockey~~
   * ~~Perfil de instancia de EC2 🡪 LabInstanceProfile~~
2. Revisión: enviar y esperamos

Ya por último nos meteremos en el dominio del entorno del frontend y ya nos saldría la página web completamente implementada en AWS a través de RDS y EBSs

Una captura de pantalla de una red social

El contenido generado por IA puede ser incorrecto.