APLICACIÓN WEB PARA LA ADMINISTRACIÓN DE PROPIEDAD HORIZONTAL IMPLEMENTADO EN EL CONJUNTO RESIDENCIAL HACIENDA EL ROSAL UBICADO EN FUNZA CUNDINAMARCA

MANUAL TÉCNICO

KATHERINE AGUIRRE TIQUE
LEIDY YOHANA RODRÍGUEZ FUENTES

UNIVERSIDAD ANTONIO NARIÑO
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS
BOGOTÁ D.C.
2020

CONTENIDO

TABLA DE FIGURAS

Pág.

INTRODUCCIÓN

En el siguiente documento se podrá evidenciar todas las características técnicas de la aplicación web desarrollada para el conjunto residencial Hacienda el Rosal. En este manual se describen los usuarios que la utilizan, los requisitos de software y hardware que se requieren para poder ejecutarla. Por último, los pasos a seguir para poder crear toda la infraestructura de la aplicación en AWS y la solución a los problemas de indisponibilidad de servicios que se pueden dar al momento de manipularla.

1. DESCRIPCIÓN GENERAL DEL SISTEMA

Esta aplicación web es un sistema para la gestión de la información de los residentes, notificaciones por mensajes de texto y de noticias acerca de lo que sucede dentro del conjunto.

Para acceder a dicha aplicación, se puede hacer desde el celular, tablet y computador portátil o de escritorio en cualquier lugar y a cualquier hora del día.

2. CARACTERÍSTICAS DE LOS USUARIOS DEL SISTEMA

Para esta aplicación web los usuarios que la van a usar son:

Administrador: es el que se encarga de gestionar toda la información de los residentes, el envío de mensajes de texto, la publicación de noticias y estar pendiente de las solicitudes que llegan de los residentes.

Residente: son propietarios o arrendatarios del conjunto residencial, usan la aplicación para estar más informados acerca de lo que pasa dentro del conjunto, saber sobre sus pagos y enviar solicitudes a la administración de acuerdo con su interés.

3. REQUISITOS DEL HARDWARE Y DE SOFTWARE

Los requisitos para utilizar la aplicación web son:

- Tener internet.
- Tener un computador, celular o tablet en casa u oficina.
- Tener un navegador de internet instalado, puede ser Google Chrome, Firefox, Safari, entre otros.

4. INSTRUCCIONES DE INSTALACIÓN

Para que la aplicación corra correctamente en la web, lo primero que se debe hacer es el despliegue de toda la parte del back-end, comenzando por la base de datos, la aplicación back-end y por último la aplicación front-end.

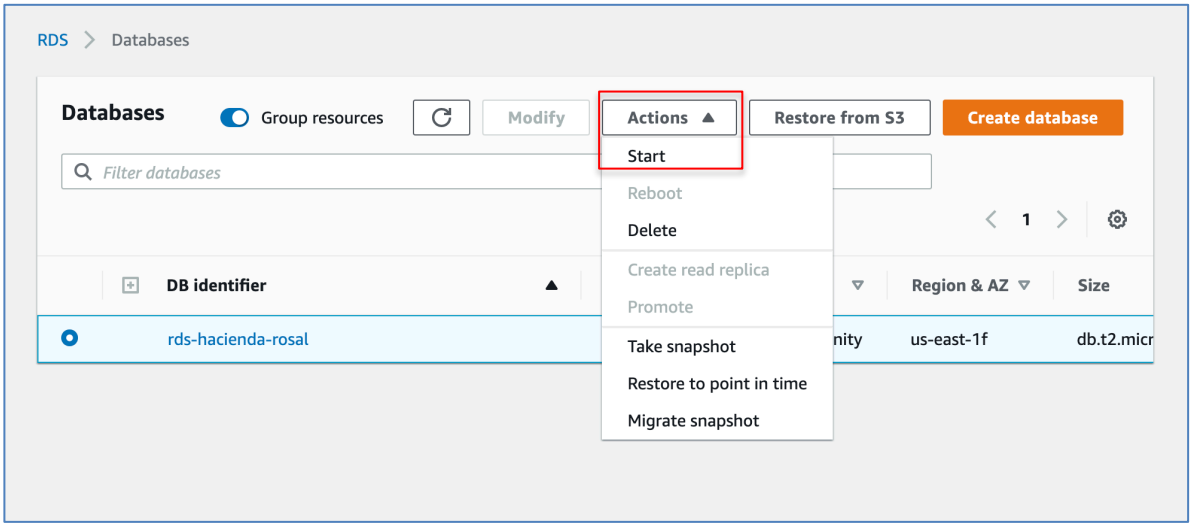
4.1. INSTALACIÓN DE LA BASE DE DATOS

Para correr la base de datos en el servicio RDS (Relational Data base Service) de AWS (Amazon Web Services) siga los siguientes pasos:

- Entrar a la consola de administración de AWS.
- Entrar al servicio RDS.
- Crear la base de datos en RDS, para poder hacerlo, los pasos están en la página oficial de AWS.
- Seleccionar Databases.
- Seleccionar la base de datos rds-hacienda-rosal.
- Seleccionar Actions y la opción start.

De esta forma la base de datos está lista para usar, como se muestra en la Figura 1.

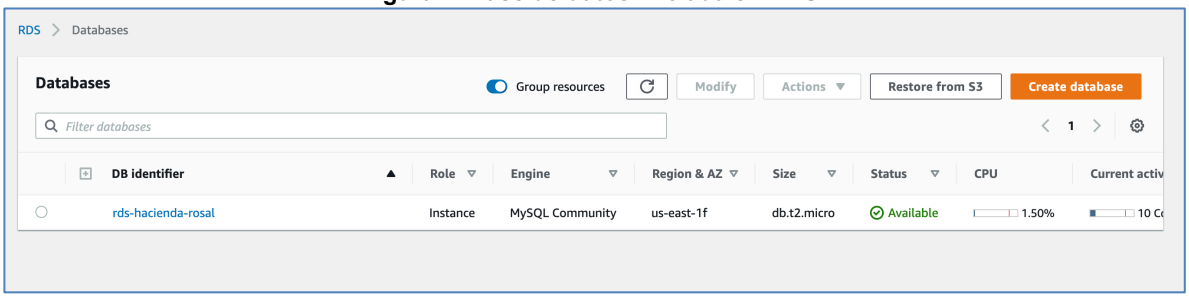
Figura 1. Inicio de base de datos en RDS



Fuente: elaboración propia

La base de datos tiene que estar disponible (Status: Available), como se muestra en la 2.

Figura 2. Base de datos iniciada en RDS



Fuente: elaboración propia

Se debe conectar por medio de un cliente. Para este caso, con Workbech (cliente gráfico de MySQL). Descargar el cliente siguiendo las instrucciones de la página oficial, en el siguiente enlace: https://www.mysql.com/products/workbench/. Luego de instalar el cliente, establecer la conexión con los datos de la base de datos en el RDS, como se muestra en la Figura 3. Colocar los siguientes datos:

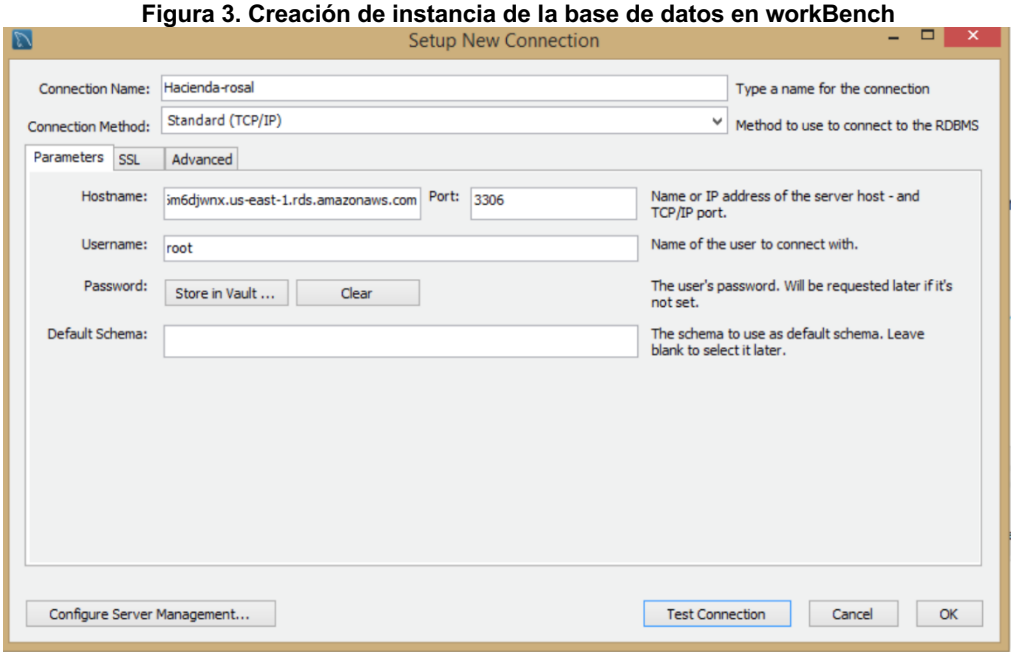
User: root

Password: temroot*

Hostname: rds-hacienda-rosal.chg16m6djwnx.us-east-1.rds.amazonaws.com

Dar clic en “OK” y se hace la conexión con la instancia creada en AWS.

Figura 3. Creación de instancia de la base de datos en workBench



Fuente: elaboración propia

Luego de tener la conexión, crear la base de datos en MySQL. Acceder a la base de datos y realizar los siguientes pasos:

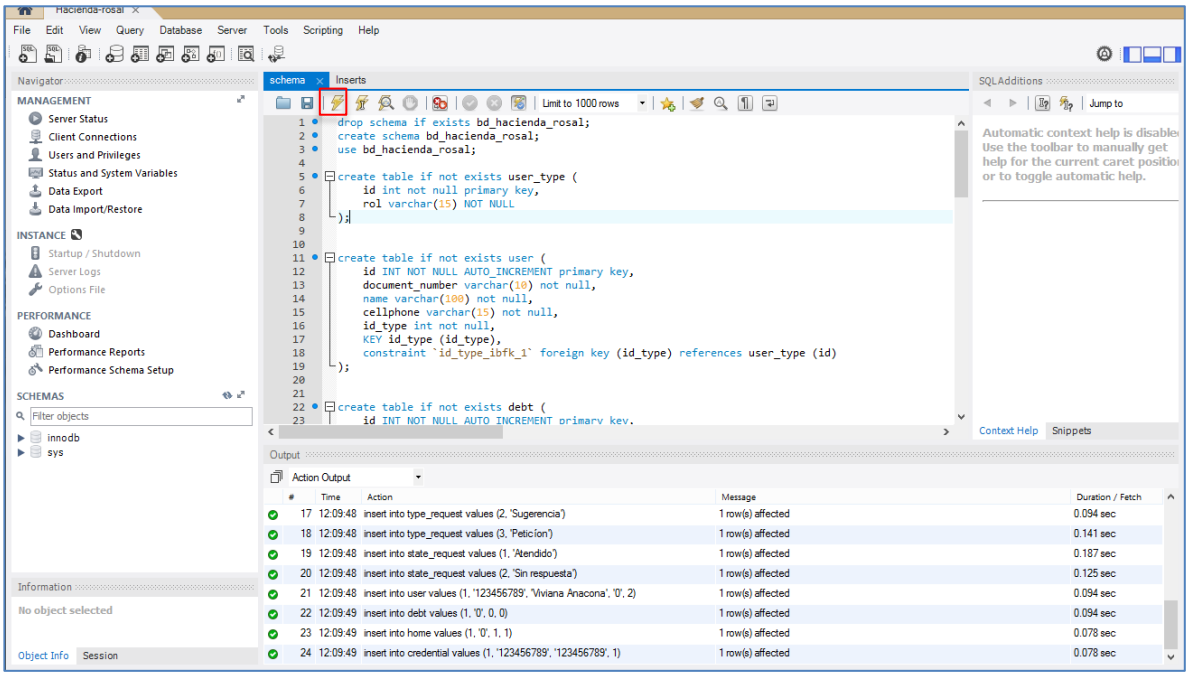
- Copiar el siguiente script y pegarlo en la consola de workbench. En la última sentencia, en los datos entre corchetes escribir la información del administrador. Dar clic en el ícono ejecutar, como se muestra en la Figura 4.

```
drop schema if exists bd_hacienda_rosal;
create schema bd_hacienda_rosal;
use bd_hacienda_rosal;

create table if not exists user_type (
        id int not null primary key,
   rol varchar(15) NOT NULL
);

create table if not exists user (
        id INT NOT NULL AUTO_INCREMENT primary key,
        document_number varchar(10) not null,
   name varchar(100) not null,
   cellphone varchar(15) not null,
   id_type int not null,
   KEY id_type (id_type),
   constraint `id_type_ibfk_1` foreign key (id_type) references user_type (id)
);

create table if not exists debt (
        id INT NOT NULL AUTO_INCREMENT primary key,
        tower_number_home varchar (15) not null,
        amount double not null,
   months int not null
);

create table if not exists home (
        id INT NOT NULL AUTO_INCREMENT primary key,
        tower_number_home varchar (15),
   id_user int not null,
   id_debt int not null,
   KEY id_user (id_user),
   constraint `id_user_ibfk_1` foreign key (id_user) references user (id),
   KEY id_debt (id_debt),
   constraint `id_debt_ibfk_2` foreign key (id_debt) references debt (id)
);
```

```
create table if not exists credential (
        id INT NOT NULL AUTO_INCREMENT primary key,
        user varchar(100) not null unique,
        password varchar(100) not null,
   id_home int not null,
   KEY id_home (id_home),
   constraint `id_home_ibfk_1` foreign key (id_home) references home (id)
);

create table if not exists type_request (
        id int  not null primary key,
   affair varchar(15) not null
);

create table if not exists state_request (
        id int  not null primary key,
   state varchar(15) not null
);

create table if not exists request (
        id INT NOT NULL AUTO_INCREMENT primary key,
   message varchar(500) not null,
        id_state int  not null,
   id_type int  not null,
   publish_date date not null,
        id_home int not null,
   response varchar(300),
        KEY id_state (id_state),
   constraint `id_state_ibfk_1` foreign key (id_state) references state_request (id),
   KEY id_type (id_type),
   constraint `id_type_ibfk_2` foreign key (id_type) references type_request (id),
   KEY id_home (id_home),
   constraint `id_home_ibfk_3` foreign key (id_home) references home (id)
);

create table if not exists news (
        id INT NOT NULL AUTO_INCREMENT key,
   information varchar(800) not null,
   publish_date  varchar(40) not null
);
create table if not exists commentary (
        id INT NOT NULL AUTO_INCREMENT key,
   message varchar(800) not null,
   publish_date varchar(40) not null,
   id_user_c int not null,
   id_news int not null,
   KEY id_user_c (id_user_c),
   constraint `id_user_cibfk_1` foreign key (id_user_c) references user (id),
   KEY id_news (id_news),
```

```
    constraint `id_news_ibfk_2` foreign key (id_news) references news (id)
);

insert into user_type values (1, 'Residente');
insert into user_type values (2, 'Administrador');

insert into type_request values (1, 'Queja');
insert into type_request values (2, 'Sugerencia');
insert into type_request values (3, 'Peticíon');

insert into state_request values (1, 'Atendido');
insert into state_request values (2, 'Sin respuesta');

insert into user values (1, '{cedula}', '{nombre-admin}', '0', 2);
insert into debt values (1, '0', 0, 0);
insert into home values (1, '0', 1, 1);
insert into credential values (1, '{user}', '{password}', 1);
```

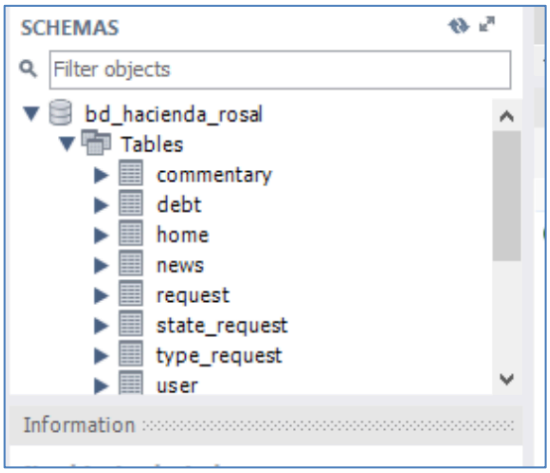
Figura 4. Ejecución del script para crear base de datos



Fuente: elaboración propia

- Sobre la conexión, aparecen las tablas que conforman la base de datos, como se observa en la Figura 5.

Figura 5. Base de datos creada sobre la conexión

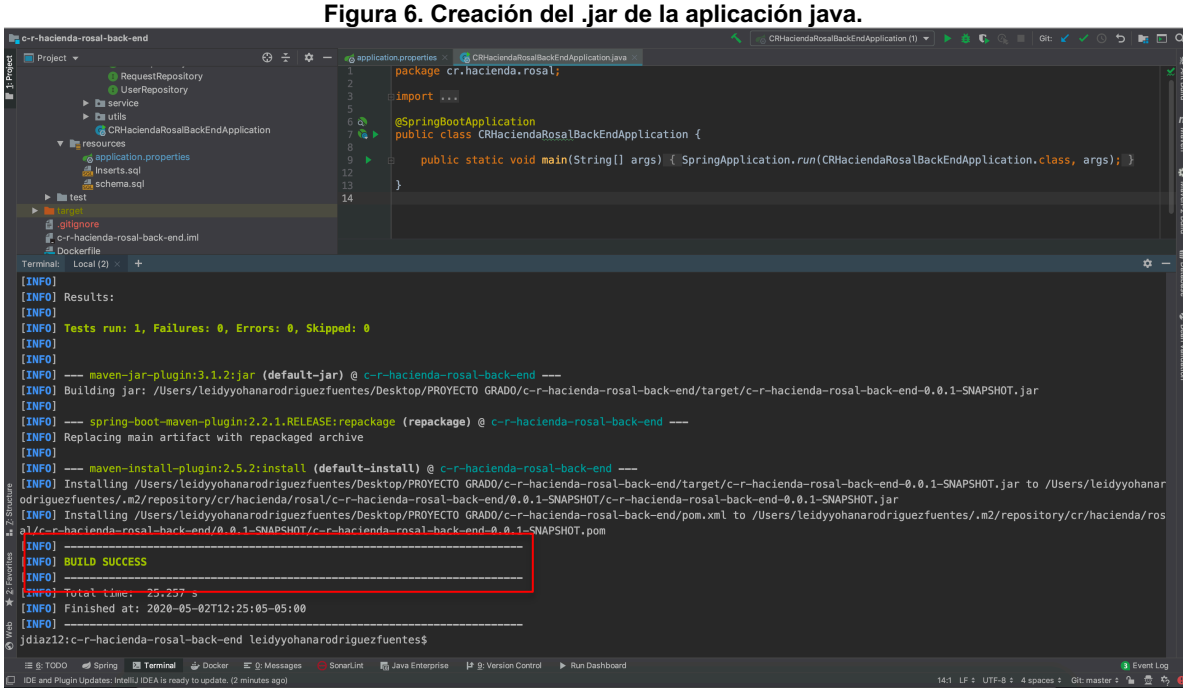


Fuente: elaboración propia

4.2. CREACIÓN DE LA IMAGEN DE DOCKER DEL PROYECTO JAVA

Abrir el proyecto java en el IDE Intellij Idea. Ir a la consola y escribir el siguiente comando: **mvn clean install**: este comando crea el .jar de la aplicación java.
En la consola aparece la salida BUILD SUCCESS, como en la Figura 6.

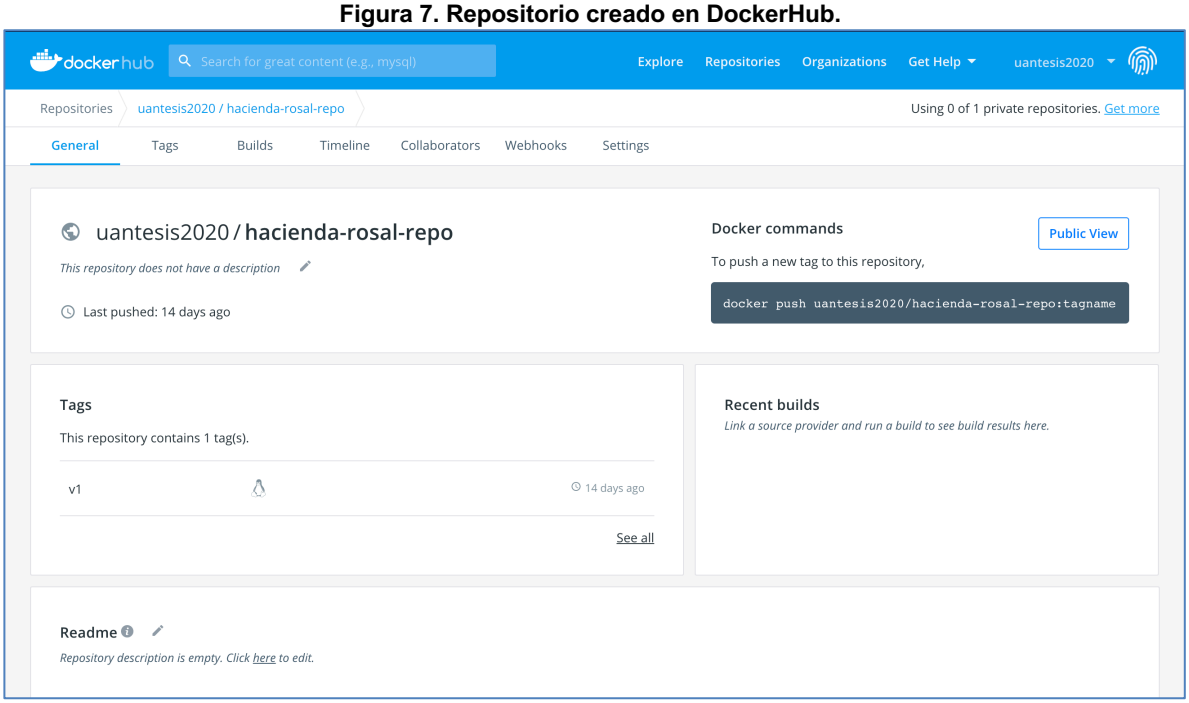
Figura 6. Creación del .jar de la aplicación java.



Fuente: elaboración propia

Crear el repositorio en DockerHub. Seguir las instrucciones de creación de un repositorio en la página oficial de DockerHub. El repositorio creado se observa en la Figura 7.

Figura 7. Repositorio creado en DockerHub.



Fuente: elaboración propia

Instalar docker de acuerdo con las indicaciones de la página https://www.redeszone.net/2019/04/21/instalar-docker-windows-10/.

Luego de tener instalado Docker, proceder a usar los comandos para crear la imagen docker del .jar, que generó la aplicación java. En el proyecto crear un archivo llamado Dockerfile que contenga lo que se observa en la Figura 8.

Figura 8.Creación archivo Dockerfile

```
FROM java:8-jdk-alpine
COPY ./target/c-r-hacienda-rosal-back-end-0.0.1-SNAPSHOT.jar
/usr/app/
WORKDIR /usr/app
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "c-r-hacienda-rosal-back-end-
0.0.1-SNAPSHOT.jar"]
```

Fuente: elaboración propia

En la consola correr el comando: **docker build -t hacienda-rosal.** La salida se tiene que ver como la Figura 9.

Figura 9. Ejecución del comando docker para crear contenedor del archivo .jar

Fuente: elaboración propia

Luego de que la imagen docker queda creada con el .jar de la aplicación, subir la imagen a dockerHub. Correr los siguientes comandos:

docker login: este comando pide las credenciales de la cuenta en dockerHub.

docker tag hacienda-rosal uantesis2020/hacienda-rosal-repo:v1: con este comando se asocia la imagen docker con el nombre del repositorio en dockerHub.

docker push uantesis2020/hacienda-rosal-repo:v1: este comando sube la imagen docker a dockerHub.

La salida de estos comandos se muestra en la Figura 10.

Figura 10. Subida de imagen docker a DockerHub.

Fuente: elaboración propia

Correr el comando: **docker run -p 4000:3000 uantesis2020/hacienda-rosal-repo:v1,** para comprobar que la imagen de docker quedó montada en dockerHub, y esta funcionado correctamente dentro del contenedor de docker en DockerHub, como se observa en Figura 11.

Figura 11. Comprobación de la ejecución aplicación en DockerHub.

Fuente: elaboración propia

4.3. CREACIÓN DE LA TAREA EN ECS (ELASTIC CONTAINER SERVICE) PARA QUE CORRA LA IMAGEN DOCKER DE LA APLICACIÓN

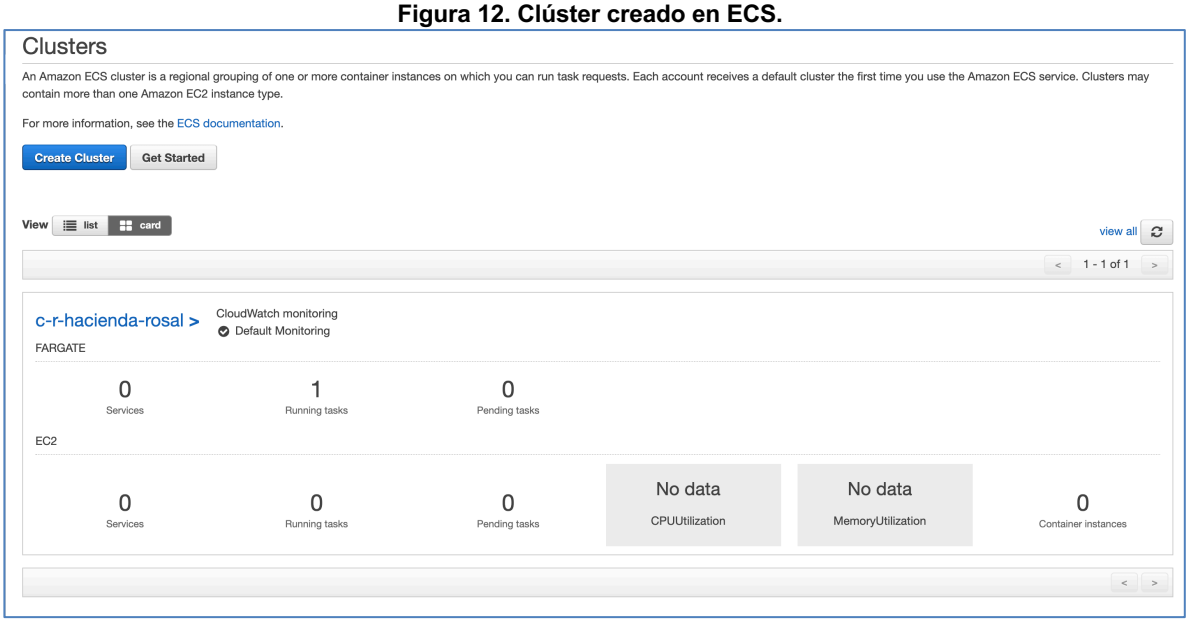
Para crear la tarea en el ECS es necesario realizar los siguientes pasos:

- Ingresar a la consola de AWS.
- Ingresar al servicio ECS.
- Crear un clúster con Fargate y asociar la imagen docker de DockerHub. Seguir el tutorial de la página: https://www.youtube.com/watch?v=3yBIRmUJhio.
- Crear la tarea que asocia la imagen de docker y dockerHub.

La creación de la tarea se hace de la siguiente manera:

Tener el clúster creado, para este proyecto se creó uno llamado c-r-hacienda-rosal como lo muestra la Figura 12.

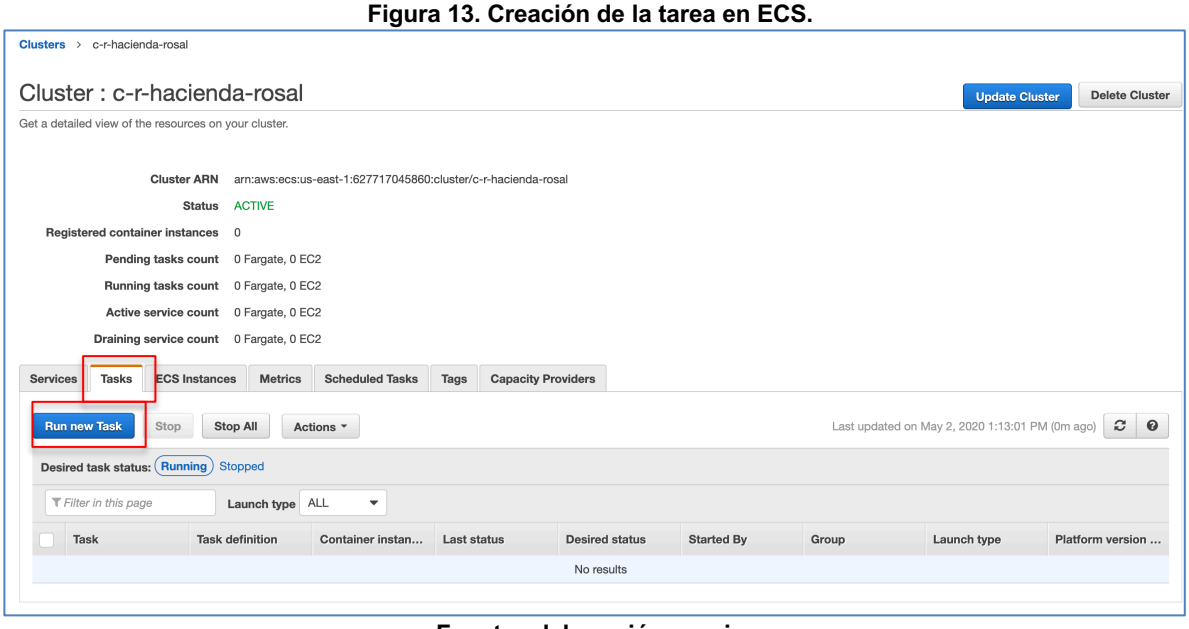
Figura 12. Clúster creado en ECS.



Fuente: elaboración propia

Ingresar al clúster para crear la tarea. Dar clic en la pestaña “Tasks” y luego en “Run new Task”, como lo muestra la Figura 13.

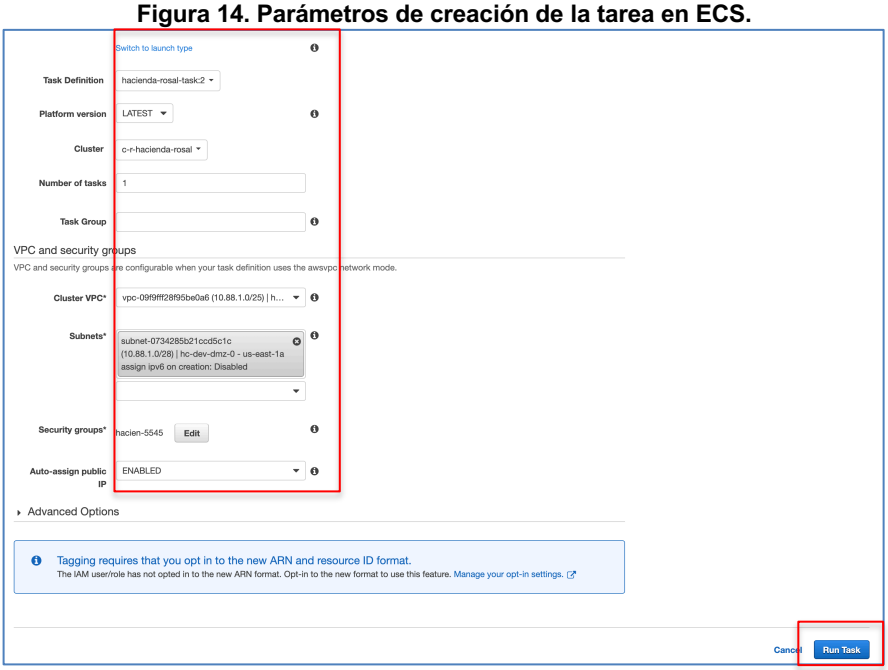
Figura 13. Creación de la tarea en ECS.



Fuente: elaboración propia

Se despliega una pantalla como se muestra en la Figura 14, en donde se establecen los parámetros necesarios para poder correr la tarea y por último clic en “Run Task”.

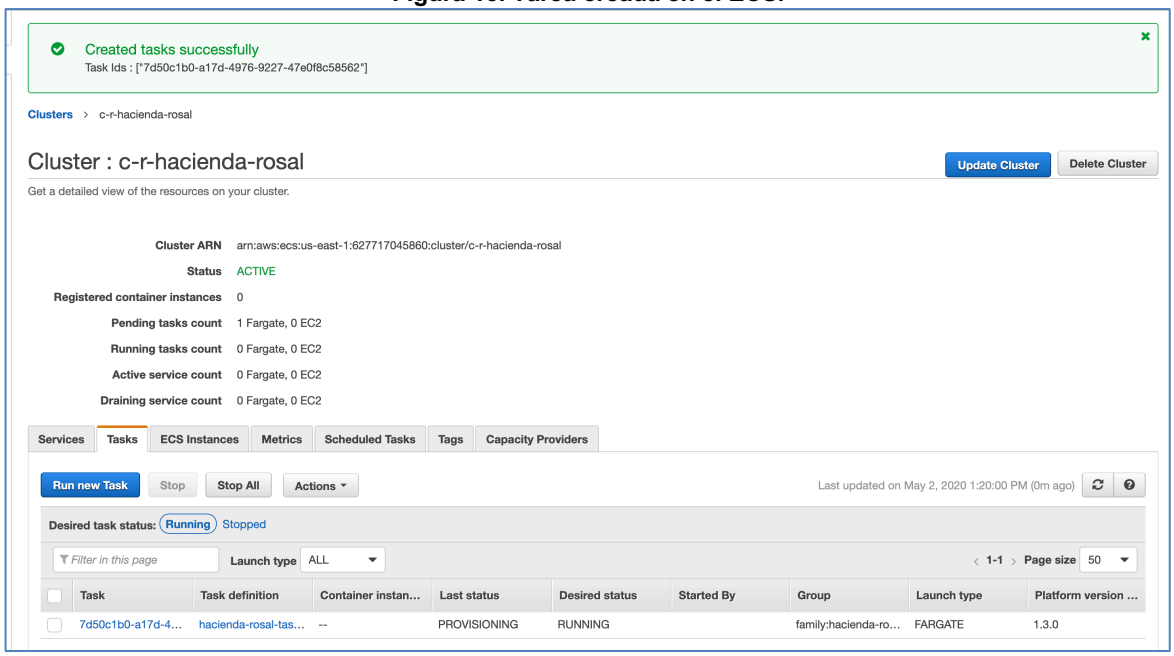
Figura 14. Parámetros de creación de la tarea en ECS.



Fuente: elaboración propia

Aparece un mensaje de creación exitosa, como se ve en la Figura 15.

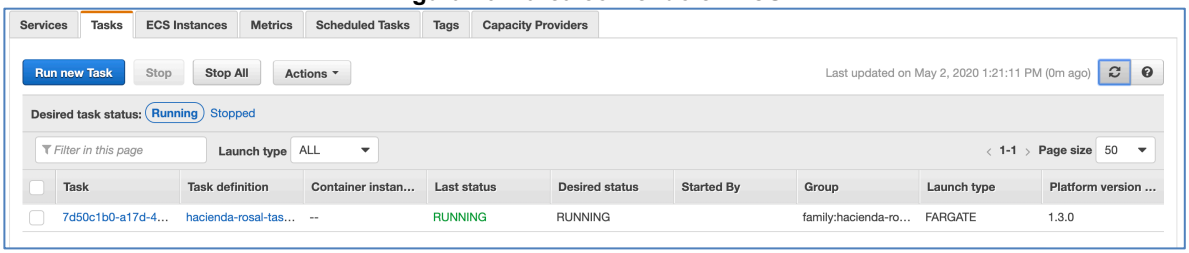
Figura 15. Tarea creada en el ECS.



Fuente: elaboración propia

La tarea se debe ver corriendo, como lo muestra la Figura 16.

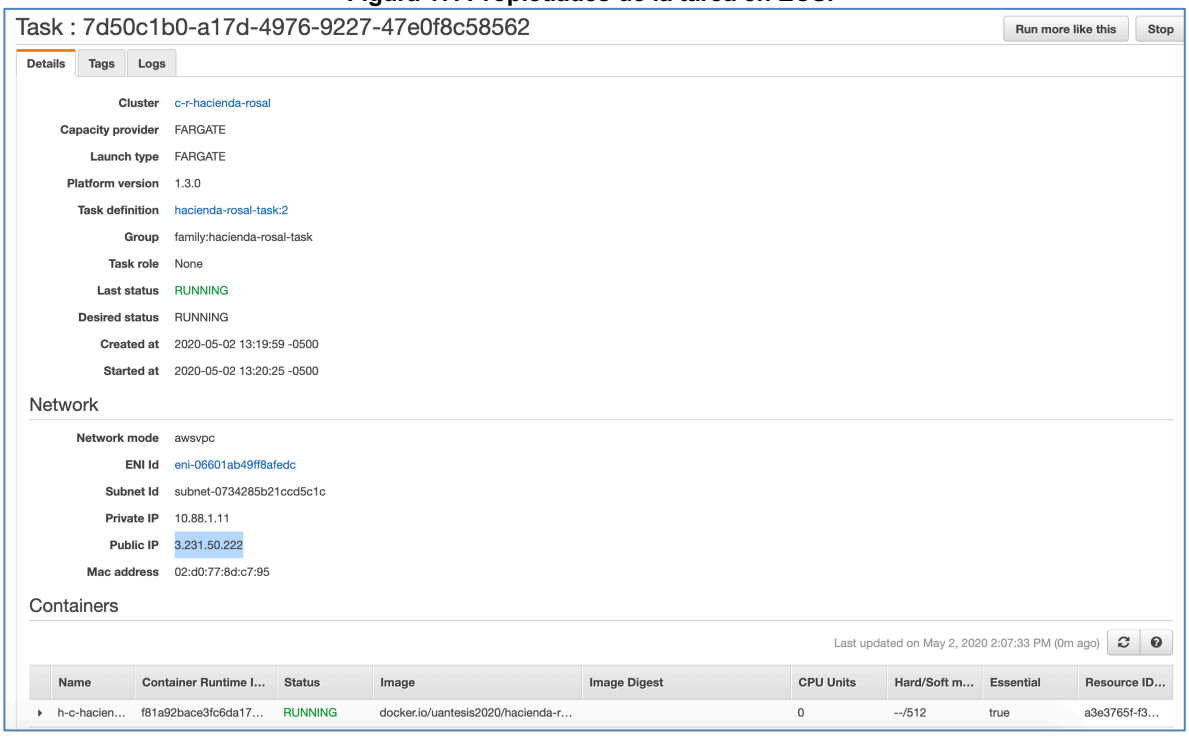
Figura 16. Tarea corriendo en ECS.



Fuente: elaboración propia

De esta forma, la tarea queda corriendo correctamente y expuesta en la ip 3.231.50.222, como lo muestra la Figura 17.

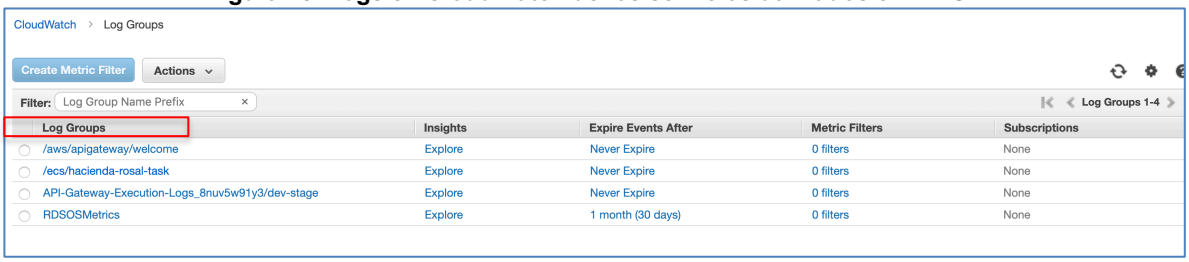
Figura 17. Propiedades de la tarea en ECS.



Fuente: elaboración propia

Ver los logs y verificar que la aplicación este corriendo perfectamente. Ir al servicio CloudWatch, dar clic en “Log Groups”. Aparece un panel con los logs que deja AWS de los servicios que se estén usando. En seguida dar clic en /ecs/hacienda-rosal-task, como se observa en la Figura 18.

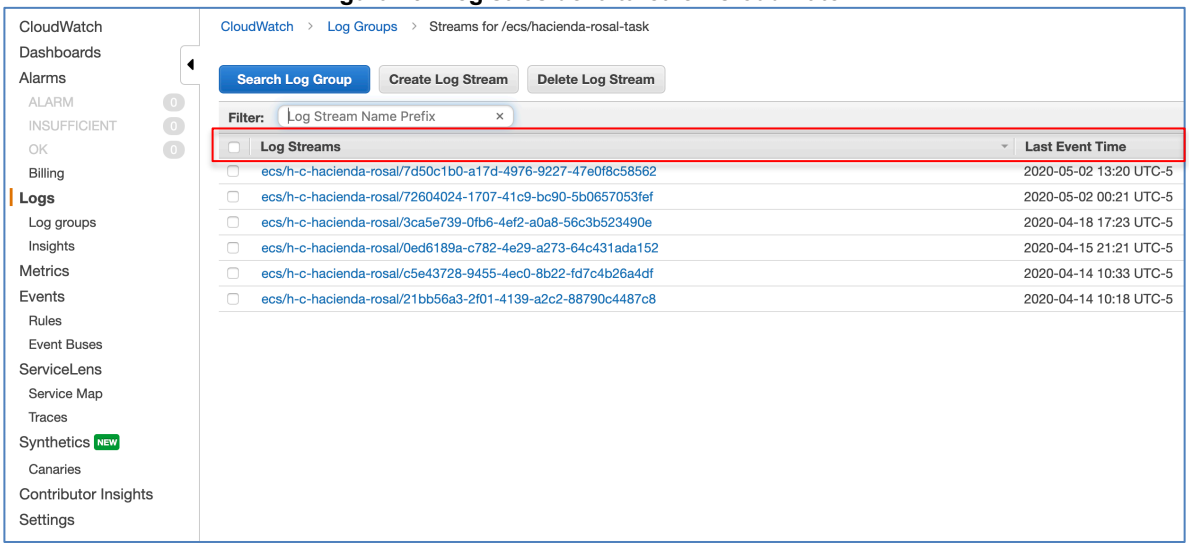
Figura 18. Logs en CloudWatch de los servicios utilizados en AWS.



Fuente: elaboración propia

Escoger los logs de la tarea que se creó en el clúster. Se despliegan una serie de registros de todas las tareas creadas, como en la Figura 19. Para ver la última, dar clic en el primer registro.

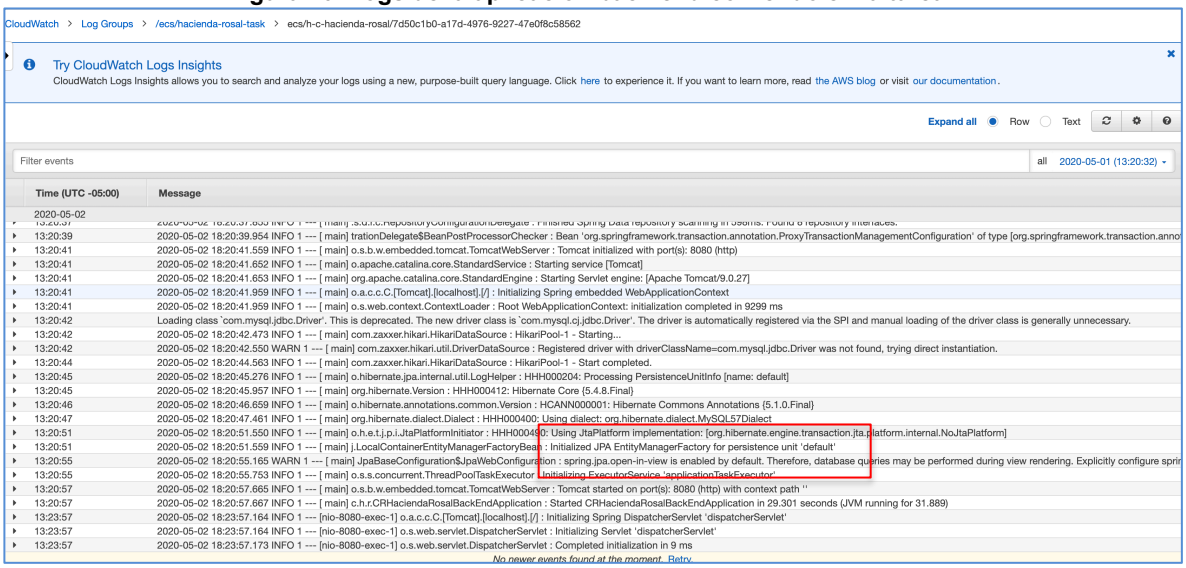
Figura 19. Registros de la tarea en CloudWatch.



Fuente: elaboración propia

En los logs que deja la aplicación, verificar que se haya levantado por el puerto 8080 como se muestra en la Figura 20. Con estos pasos, la aplicación back-end queda corriendo sobre una máquina expuesta en el puerto 8080.

Figura 20. Logs de la aplicación back-end corriendo en la tarea.



Fuente: elaboración propia

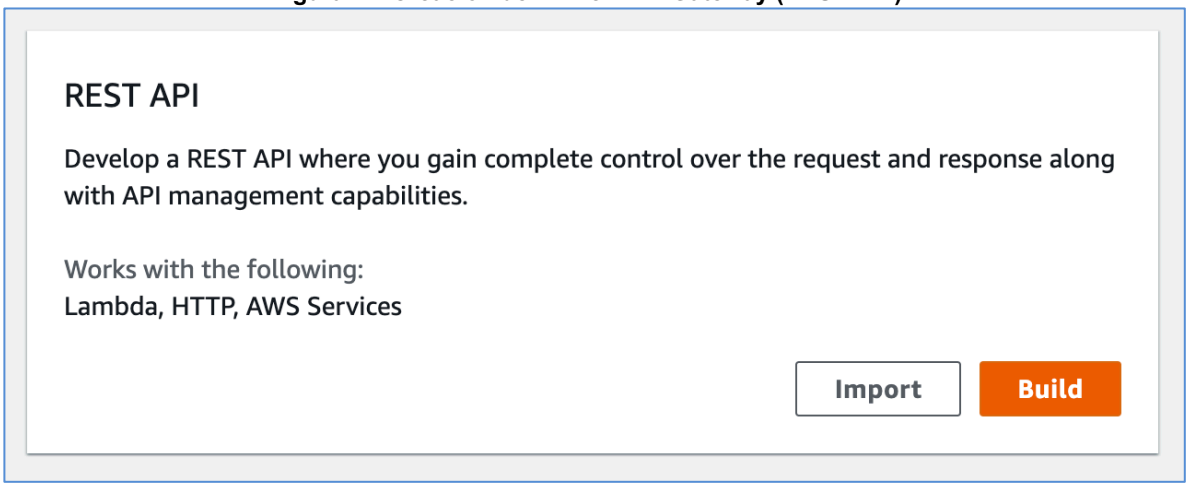
4.4. CREACIÓN DEL API EN EL SERVICIO API GATEWAY

Para consumir los recursos expuestos de la aplicación back-end, es necesario hacerlo mediante una capa de seguridad, no es correcto hacerlo directamente a la IP, en donde quedó expuesta la aplicación back-end. Para crear la capa de seguridad y acceder al punto de enlace o endpoint que genera, se utiliza el servicio API Gateway, con esto el consumo se hace por medio de HTTPS.

Lo primero que toca hacer es crear el API, seguir los siguientes pasos:

- Ingresar a la consola de AWS.
- Ingresar al servicio API Gateway.
- Crear el API. Dar clic en “Create API” y escoger el método REST API. Luego dar clic en “Build”, como se observa en la Figura 21.

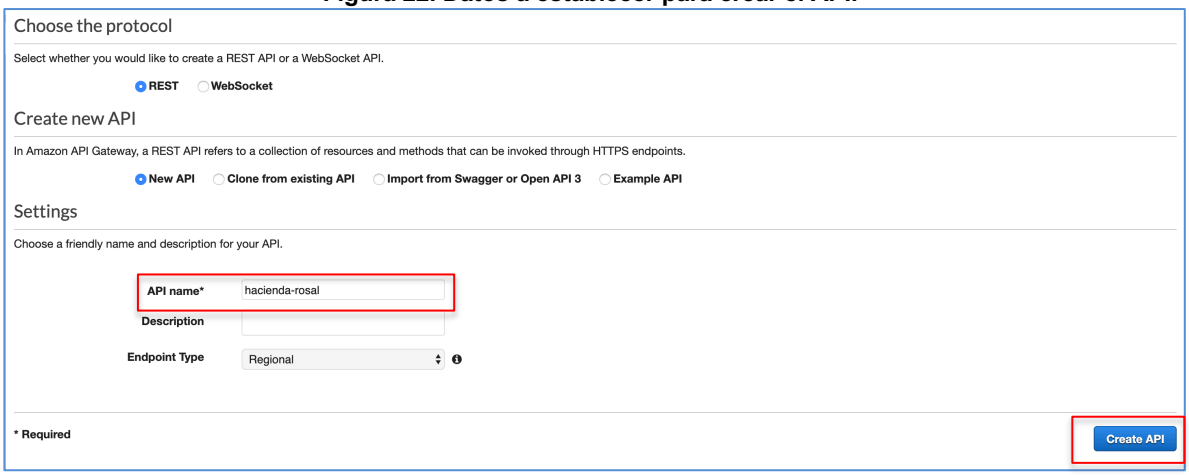
Figura 21. Creación del API en API Gateway (REST API).



Fuente: elaboración propia

Establecer el nombre en API name (hacienda-rosal) y luego clic en “Create API”, como se muestra en la Figura 22.

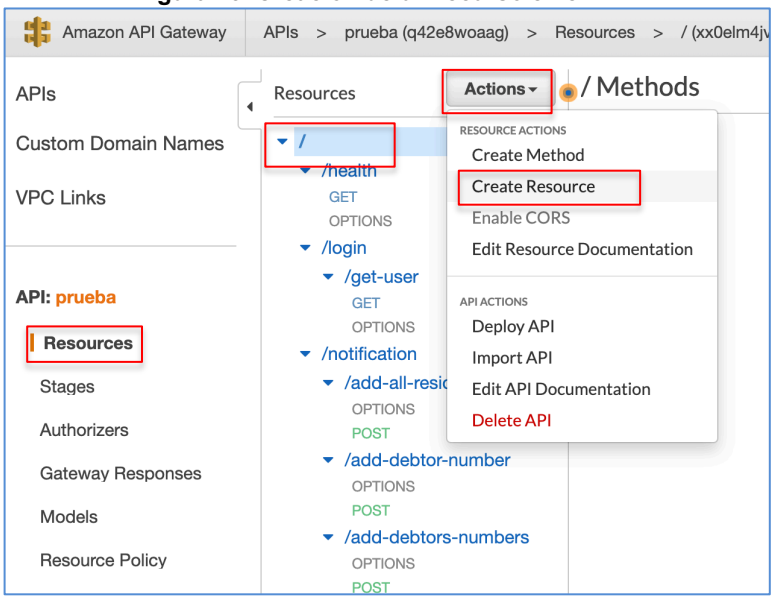
Figura 22. Datos a establecer para crear el API.



Fuente: elaboración propia

- Crear los recursos y los métodos. Se crean todos los recursos que expone la aplicación back-end con sus métodos HTTP (GET, PUT, POST, DELETE). Dar clic en “Resources”, luego en la ruta raíz y por último en “Actions” y seleccionar “Create Resource”, como se observa en la Figura 23.

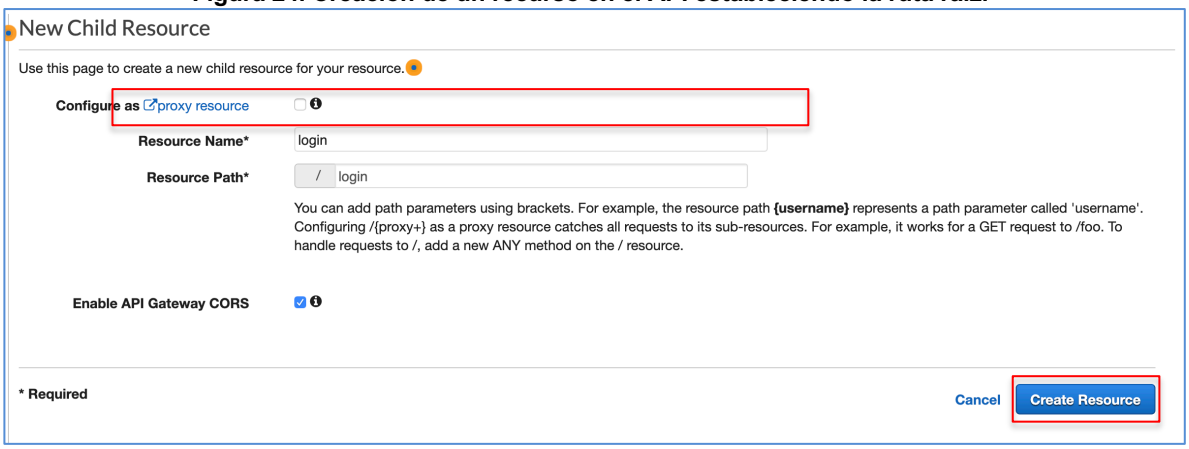
Figura 23. Creación de un recurso en el API.



Fuente: elaboración propia

Luego establecer el nombre del recurso expuesto y dar clic en “Create Resource”, como se observa en la Figura 24.

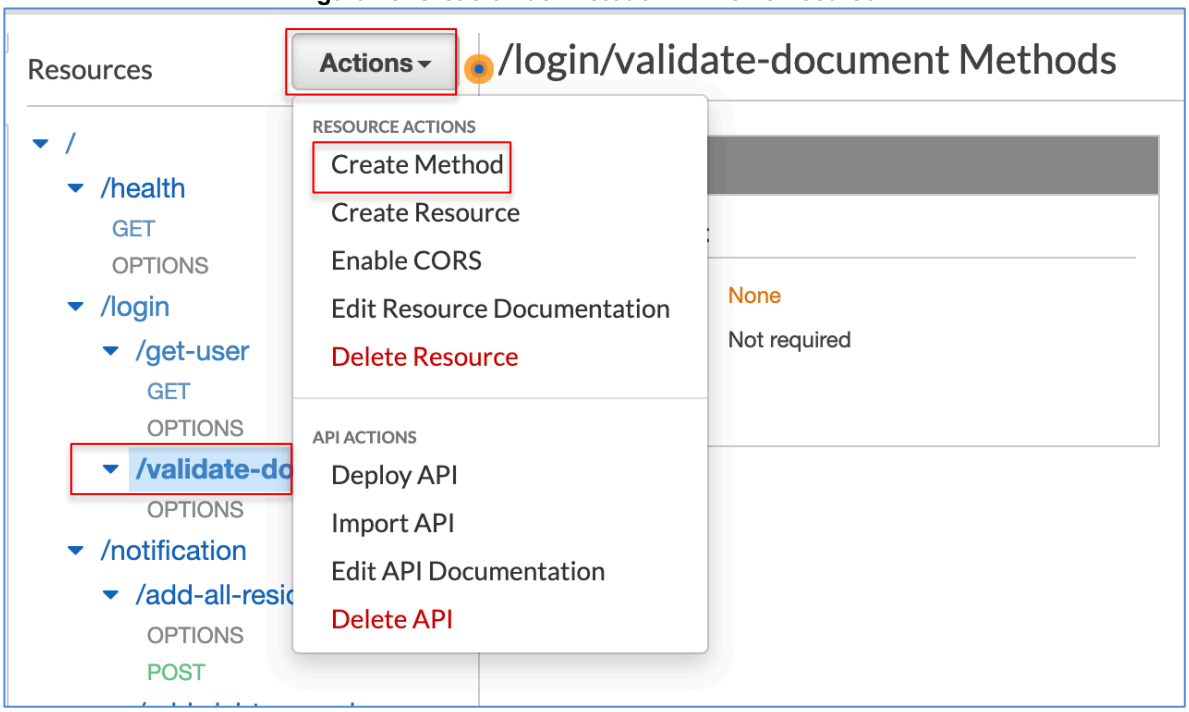
Figura 24. Creación de un recurso en el API estableciendo la ruta raíz.



Fuente: elaboración propia

Para crear el método, se da clic en el recurso ya creado. Luego clic en “Actions” y clic en “Create Method”, ver Figura 25.

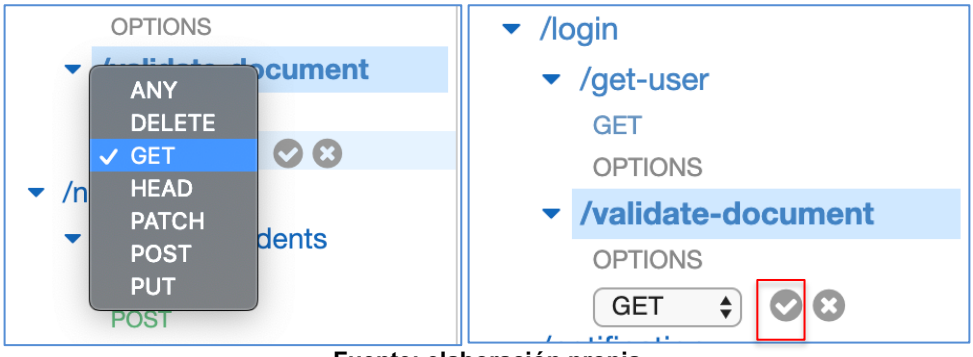
Figura 25. Creación del método HTTP en el recurso.



Fuente: elaboración propia

En el recurso login/validate-document, seleccionar el método GET y dar clic en el ícono de aceptar, como se muestra en la Figura 26.

Figura 26. Selección del método en el recurso creado.

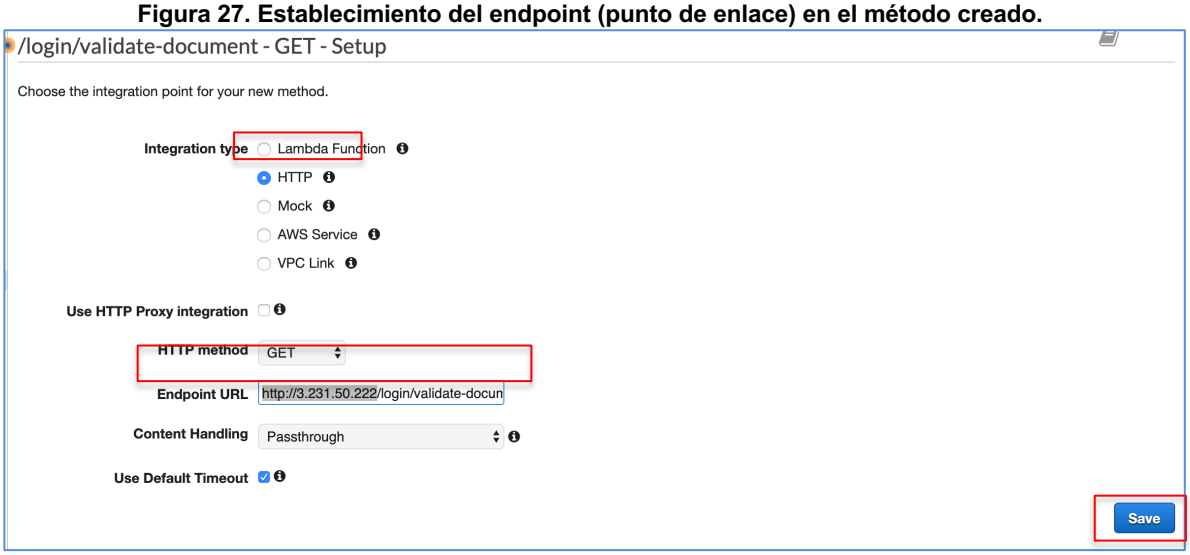


Fuente: elaboración propia

Aparece una pantalla en donde se define el método por el cual se quiere comunicar (Integration type) y el endpoint donde quiere que se conecte el API. Para este caso se establece la ip generada por la tarea creada en el ECS y el nombre del recurso.

Por último, dar clic en “Save” para guardar los cambios, como se muestra en la Figura 27. Estos pasos para crear un recurso se tienen que hacer con todos los recursos que se crearon en el proyecto back-end en java.

Figura 27. Establecimiento del endpoint (punto de enlace) en el método creado.



Fuente: elaboración propia

Se procede a obtener el endpoint o (punto de enlace) que genera el API para conectarse a ella mediante el proyecto front-end. Dar clic en “Stages”, luego en “hacienda-rosal”. Aparece la URL (endpoint) para conectarse al back-end mediante esta API, como se muestra en la Figura 28.

Figura 28. Punto de enlace (endpoint) generado por el API.

Fuente: elaboración propia

4.5. DESPLIEGUE DE LA APLICACIÓN FRONT-END EN AWS

Para poder hacer este despliegue es necesario realizar los siguientes pasos:

- Compilar el proyecto Angular para producción:
- Entrar a la consola de AWS.
- Ingresar al servicio S3 (Simple Storage Service).
- Cargar los archivos generados del proyecto compilado.

Para compilar el proyecto Angular para producción es necesario entrar al proyecto en la terminal e ingresar el siguiente comando:

ng build –prod: con este comando se compila el proyecto para usarlo en producción. Se genera la carpeta “dist”, en donde se encuentran los archivos compilados, como se ve en la Figura 29, encerrado con un recuadro amarillo.

Figura 29. Compilación del proyecto Angular para producción.

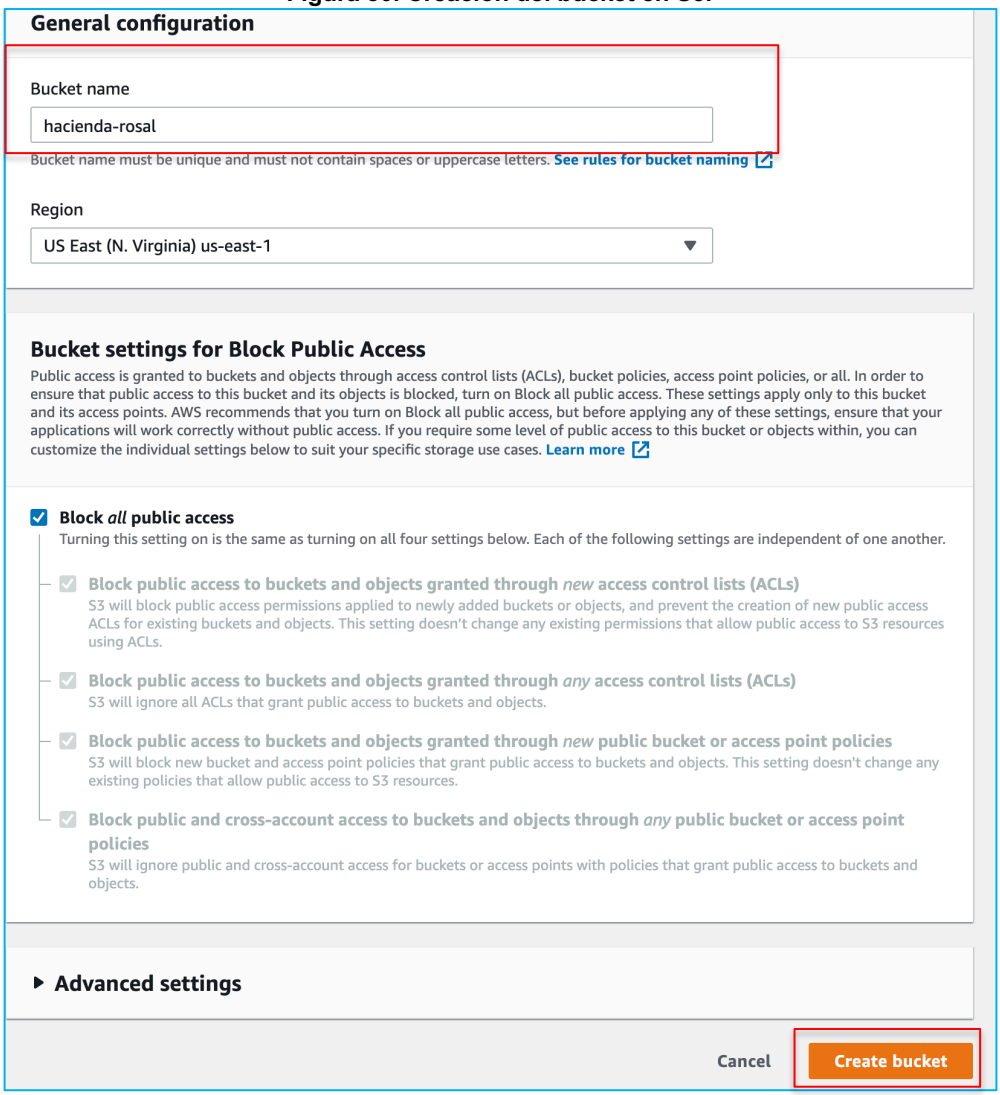
Fuente: elaboración propia

En el recuadrado azul de la Figura 29 se ven las variables de entorno, en el archivo properties.ts, se definen estas variables según el ambiente en donde se encuentre el proyecto. Una de esas variables es el punto de enlace para realizar las peticiones

HTTP al back-end, como se puede ver, está el punto de enlace que generó el API Gateway.

Luego crear el espacio de almacenamiento en la nube o bucket en el servicio S3 de AWS. Entrar a la consola de AWS y luego al servicio S3. Establecer el nombre “hacienda-rosal” y luego dar clic en “Create bucket”, como se observa en la Figura 30.

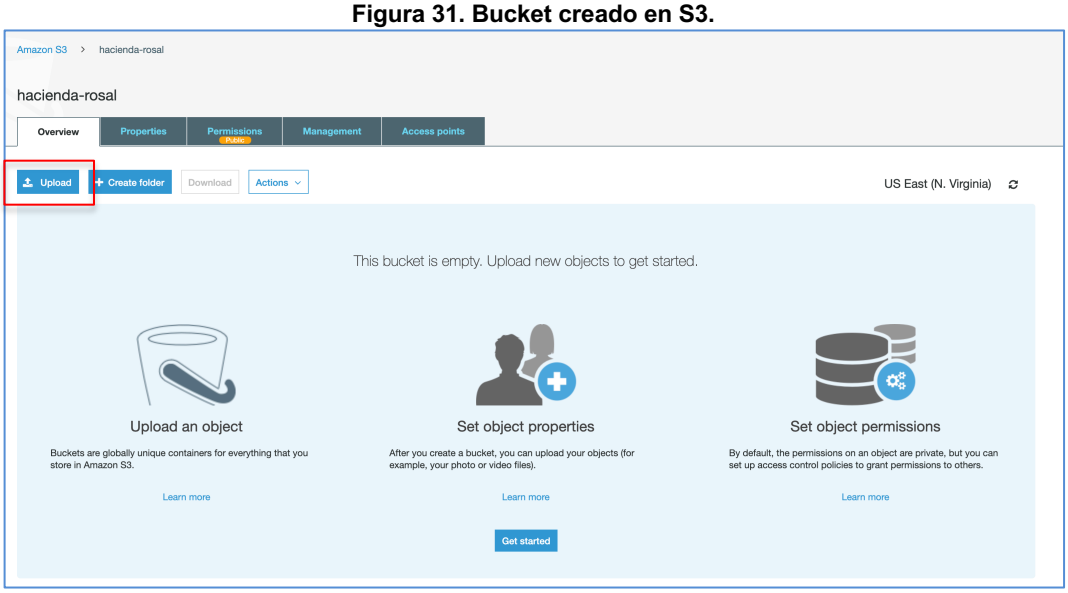
Figura 30. Creación del bucket en S3.



Fuente: elaboración propia

Aparece una pantalla nueva, dar clic en “Upload”, como se observa en la Figura 31.

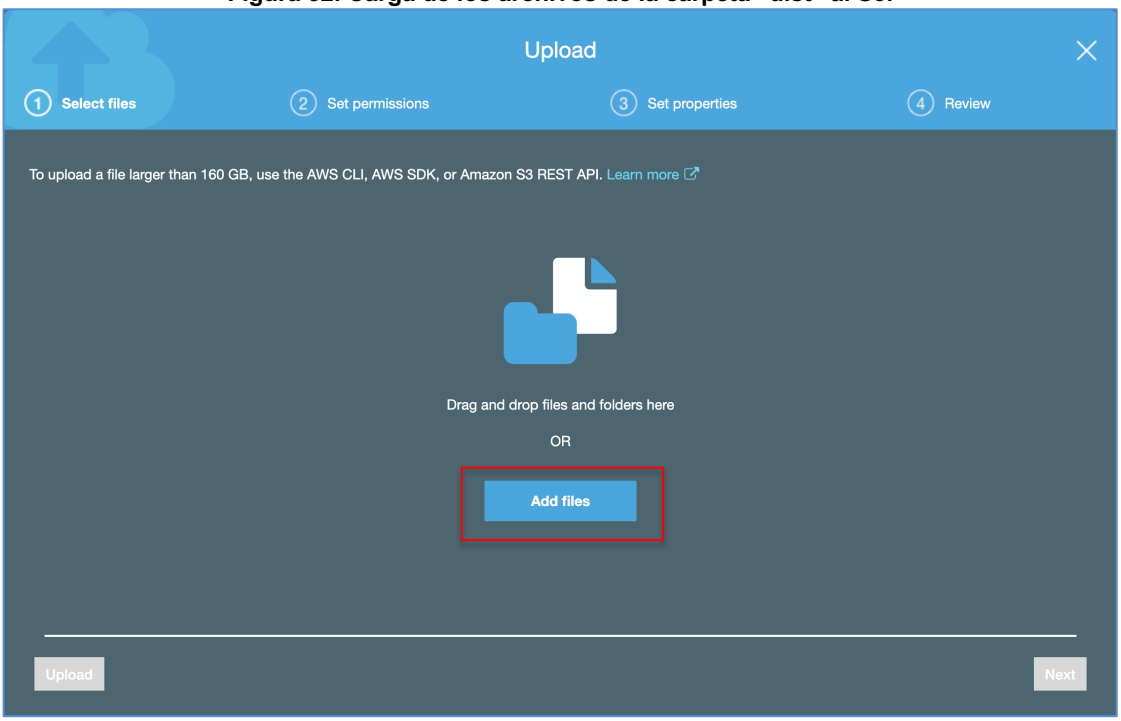
Figura 31. Bucket creado en S3.



Fuente: elaboración propia

En la nueva ventana, dar click en “Add files”, como se muestra en la Figura 32.

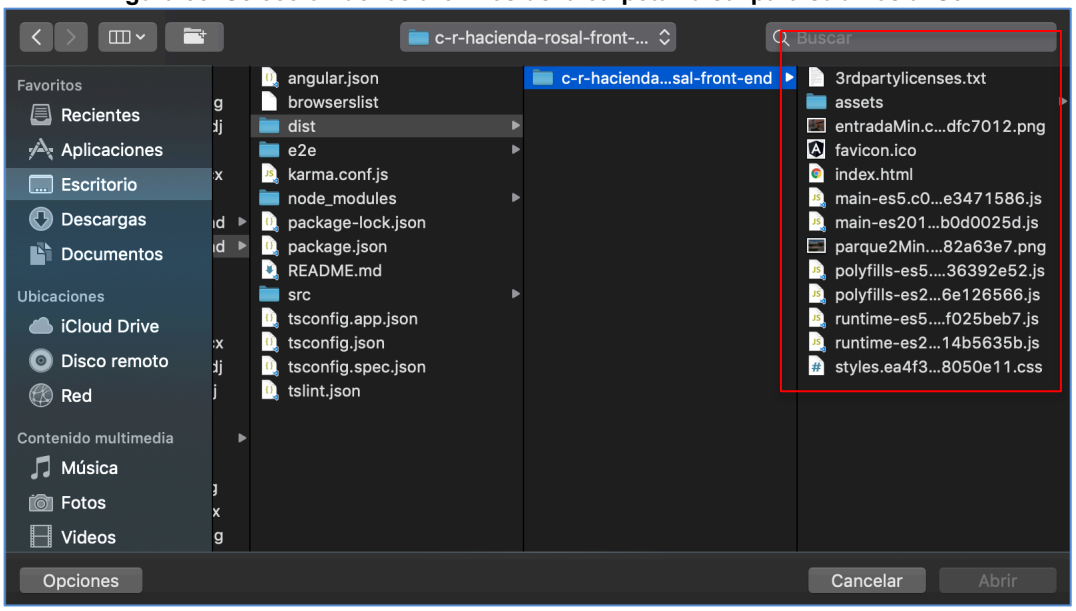
Figura 32. Carga de los archivos de la carpeta “dist” al S3.



Fuente: elaboración propia

Seleccionar todos los archivos que están dentro de la carpeta c-r-hacienda-rosal-front-end dentro de la carpeta “dist”, como se observa en la Figura 33.

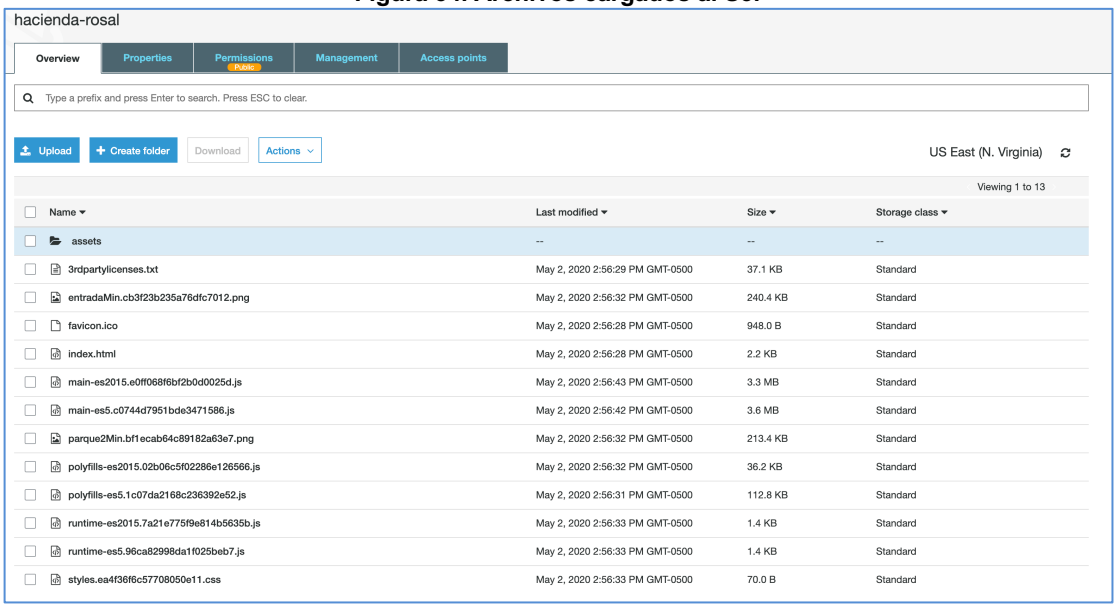
Figura 33. Selección de los archivos de la carpeta “dist” para subirlos al S3.



Fuente: elaboración propia

Cuando se carguen los archivos, se ven en el bucket como se muestra en la Figura 34.

Figura 34. Archivos cargados al S3.



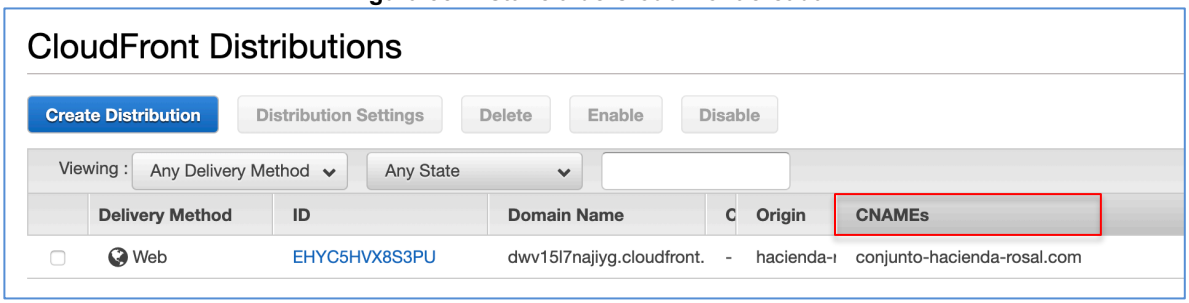
Fuente: elaboración propia

4.6. CREACIÓN DE LA INSTANCIA EN CLOUDFRONT

Luego de cargar los archivos en el bucket, se crea una instancia en el servicio CloudFront de AWS. Los pasos para obtener un dominio de la aplicación web con certificado de seguridad SSL con CloudFront, se pueden encontrar en la página oficial de AWS.

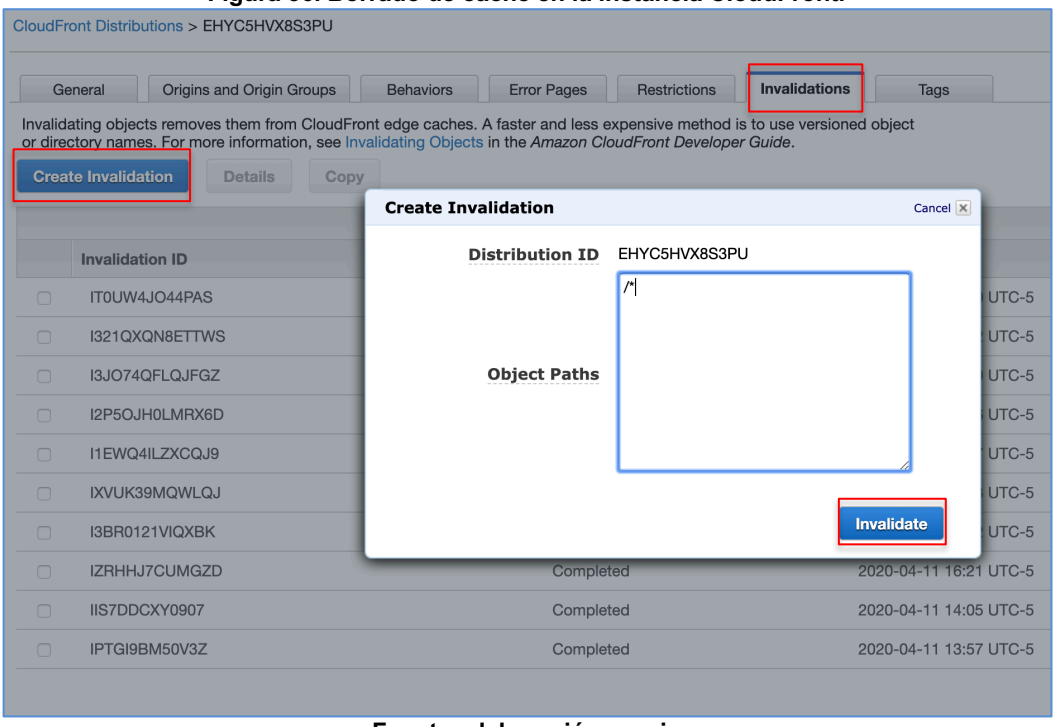
Luego de crear el dominio en CloudFront, se verá la instancia como se observa en la Figura 35. El dominio de la aplicación es https://conjunto-hacienda-rosal.com.

Figura 35. Instancia de CloudFront creada.



Fuente: elaboración propia

Figura 36. Borrado de cache en la instancia CloudFront.



Fuente: elaboración propia

Para borrar el cache de la instancia en CloudFront y la aplicación front refresque los cambios que se hicieron en el bucket, dar click sobre la instancia, luego en la opción “Invalidations”, en “Create Invalidation” y escribir el comodín /*, con este se toma todo el cache y se limpia. Por último, dar clic en el botón “Invalidate”, como se muestra en la Figura 36.

Los pasos anteriormente expuestos, es todo lo que toca hacer para poder realizar el despliegue de la aplicación.

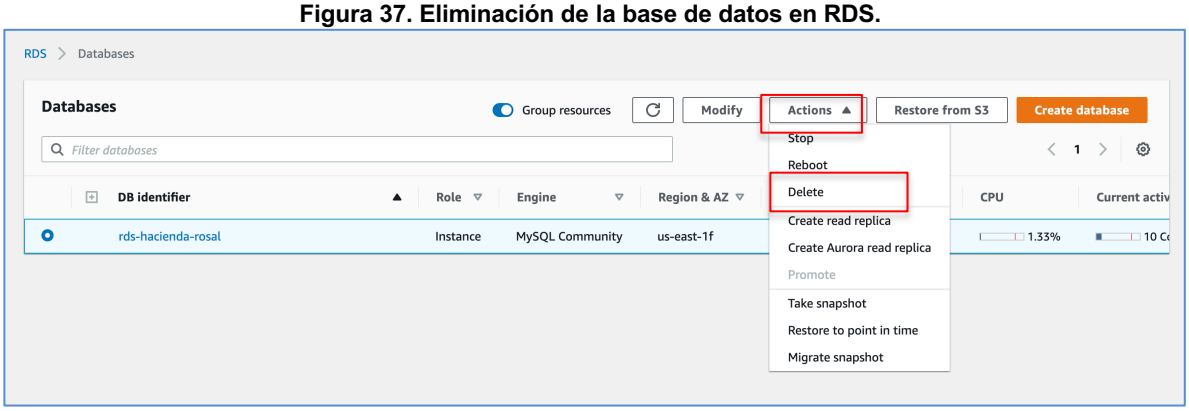
5. DESINSTALACIÓN DEL SISTEMA

Para hacer el proceso de desinstalación de la aplicación web, toca borrar todas las instancias creadas en AWS, ya que toca bajar toda la infraestructura que se tiene montada en AWS. Con este proceso la aplicación queda eliminada de la nube. Por lo tanto, no se podrá acceder a ella.

5.1. ELIMINACIÓN DE BASE DE DATOS EN EL RDS

Para borrar la instancia de la base de datos en el RDS se tienen que realizar los siguientes pasos: seleccionar la base de datos, luego clic en “Actions” y luego en “Delete”, como se observa en la Figura 37.

Figura 37. Eliminación de la base de datos en RDS.

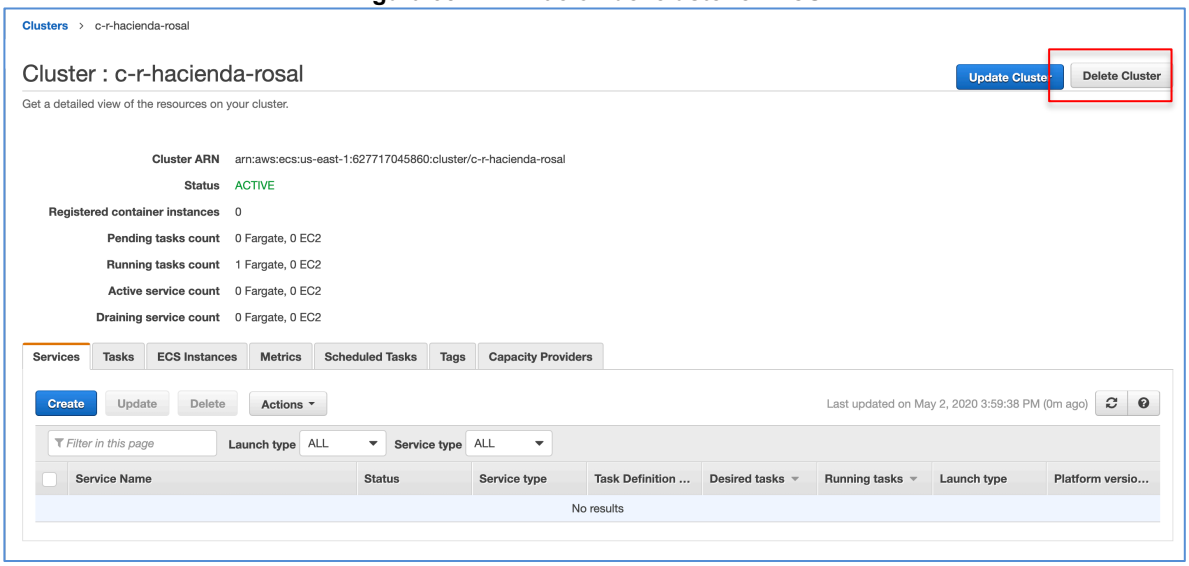


Fuente: elaboración propia

5.2. ELIMINACIÓN DEL CLÚSTER EN ECS

Para eliminar el clúster creado en el ECS, entrar al clúster y dar clic en “Delete Clúster”, como se observa en la Figura 38.

Figura 38. Eliminación del clúster en ECS.

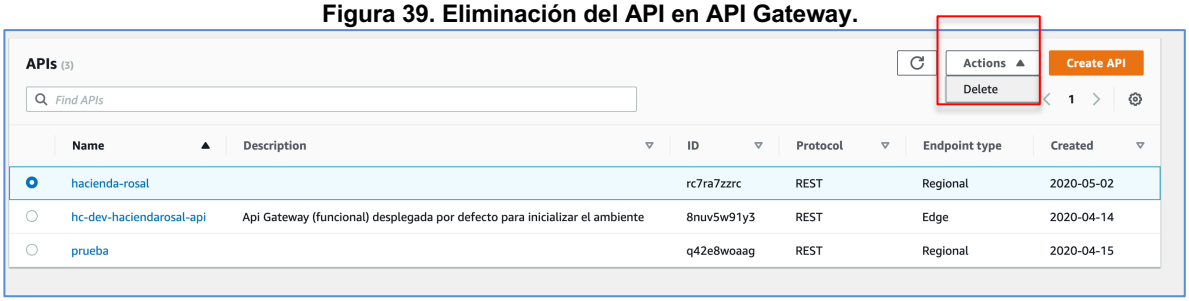


Fuente: elaboración propia

5.3. ELIMINACIÓN DEL API EN API GATEWAY

Para eliminar el API, entrar al API Gateway, seleccionar el API creado, luego seleccionar la opción “Actions” y luego clic en “Delete”, como se muestra en la Figura 39.

Figura 39. Eliminación del API en API Gateway.

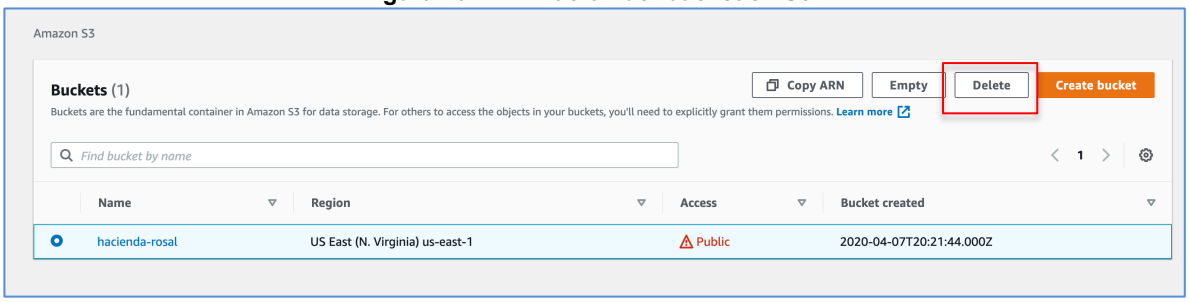


Fuente: elaboración propia

5.4. ELIMINACIÓN DEL BUCKET CREADO EN EL S3

Para eliminar el Bucket, entrar al servicio S3, seleccionar el Bucket y luego la opción “Delete”, como se observa en la Figura 40.

Figura 40. Eliminación del bucket en S3.

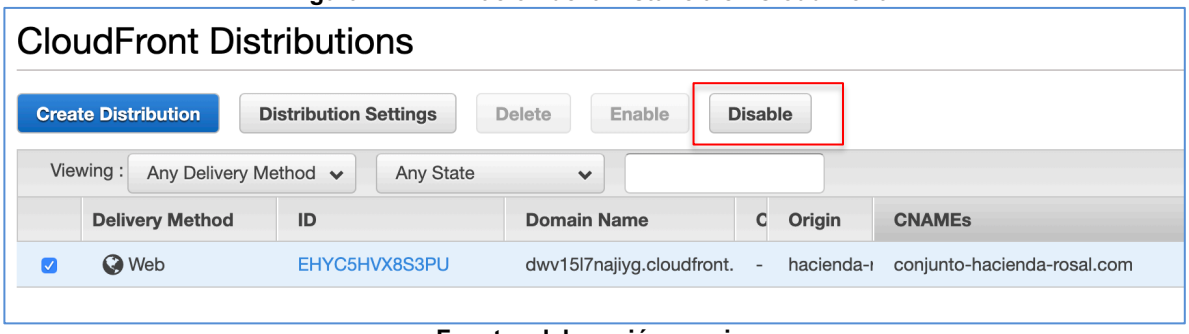


Fuente: elaboración propia

5.5. ELIMINACIÓN DE LA INSTANCIA EN CLOUDFRONT

Para eliminar la instancia en CloudFront, entrar al servicio CloudFront, seleccionar la instancia y la opción “Disable”, ver Figura 41.

Figura 41. Eliminación de la instancia en CloudFront.



Fuente: elaboración propia

6. SOLUCIÓN DE PROBLEMAS

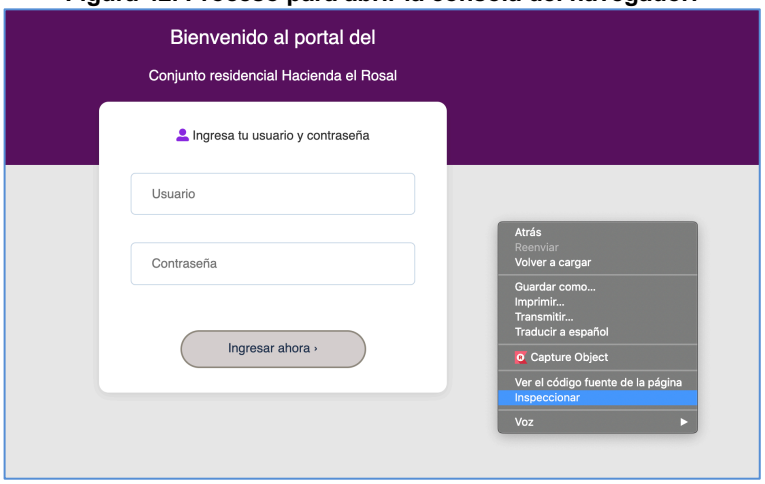
A continuación, se nombran algunos posibles errores de infraestructura que se pueden dar en la aplicación web y la solución.

6.1. ERRORES AL CONSUMIR SERVICIOS

Para visualizar los errores al consumir los servicios de la aplicación web, seguir el siguiente procedimiento:

- Clic derecho sobre la aplicación y en seguida clic en inspeccionar. Se abre la consola del navegador, como se observa en la Figura 42.

Figura 42. Proceso para abrir la consola del navegador.



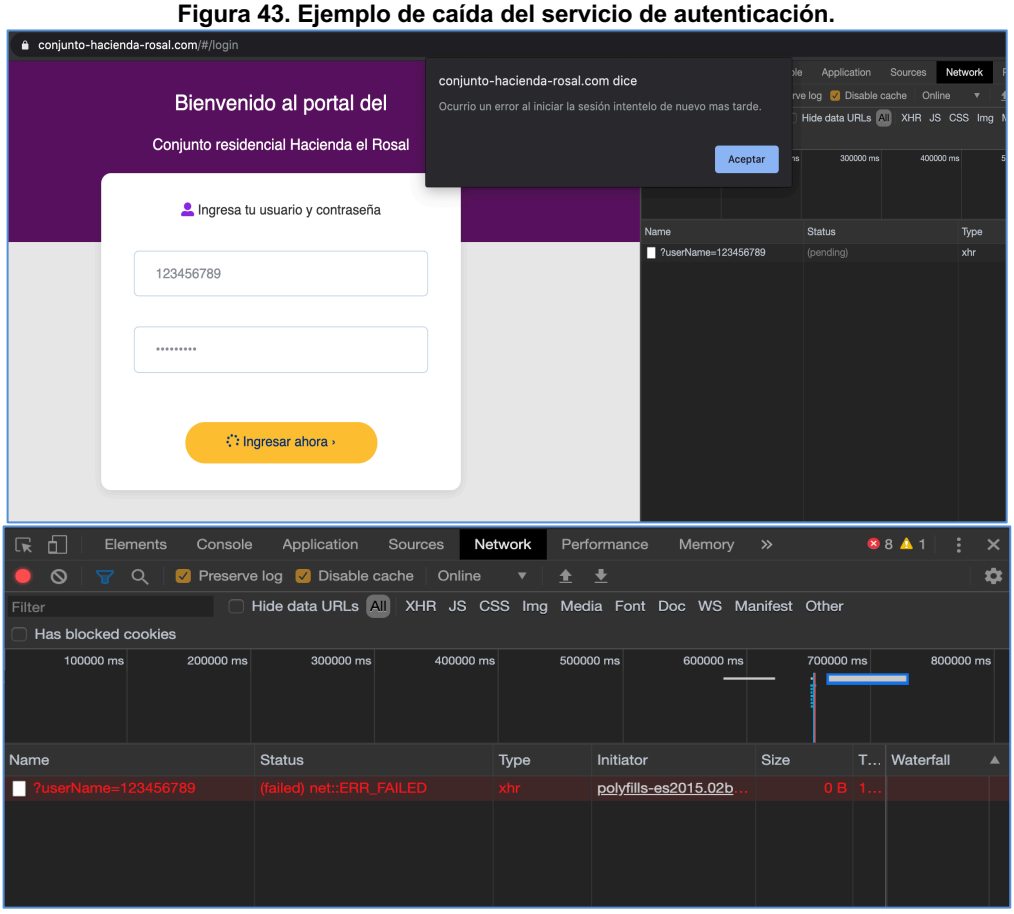
Fuente: elaboración propia

- Buscar la opción “Network”, en donde se ven todos los recursos y servicios que consume la aplicación web, un ejemplo de un servicio caído sería el siguiente:

  Este es el servicio de autenticación, si este servicio responde en su Status (estado) un codigo diferente de 200 (OK), como por ejemplo 504 (TimeOut),

FAILED o Cancel, la aplicación de inmediato muestra un error como el que se observa en la Figura 43. Este es un tema de caida de servicio que toca solucionar desde la infraestructura. De esta forma se comprueba si hay caida de servicios en la aplicación.

Figura 43. Ejemplo de caída del servicio de autenticación.



Fuente: elaboración propia

Para solucionar este tipo de inconvenientes es necesario revisar que todos los servicios e instancias creadas en AWS estén corriendo de forma correcta.