

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

	Pág.
INTRODUCCIÓN	5
1. DESCRIPCIÓN GENERAL DEL SISTEMA	6
2. CARACTERÍSTICAS DE LOS USUARIOS DEL SISTEMA	6
3. REQUISITOS DEL HARDWARE Y DE SOFTWARE	6
4. INSTRUCCIONES DE INSTALACIÓN	6
4.1. INSTALACIÓN DE LA BASE DE DATOS	7
4.2. CREACIÓN DE LA IMAGEN DE DOCKER DEL PROYECTO JAVA	12
4.3. CREACIÓN DE LA TAREA EN ECS (ELASTIC CONTAINER SERVICE) PARA QUE CORRA LA IMAGEN DOCKER DE LA APLICACIÓN	15
4.4. CREACIÓN DEL API EN EL SERVICIO API GATEWAY	20
4.5. DESPLIEGUE DE LA APLICACIÓN FRONT-END EN AWS	25
4.6. CREACIÓN DE LA INSTANCIA EN CLOUDFRONT	29
5. DESINSTALACIÓN DEL SISTEMA	31
5.1. ELIMINACIÓN DE BASE DE DATOS EN EL RDS	31
5.2. ELIMINACIÓN DEL CLUSTER EN ECS	31
5.3. ELIMINACIÓN DEL API EN API GATEWAY	32
5.4. ELIMINACIÓN DEL BUCKET CREADO EN EL S3	32
5.5. ELIMINACIÓN DE LA INSTANCIA EN CLOUDFRONT	33
6. SOLUCIÓN DE PROBLEMAS	34
6.1. ERRORES AL CONSUMIR SERVICIOS	34

TABLA DE FIGURAS

	Pág.
Figura 1. Inicio de base de datos en RDS	7
Figura 2. Base de datos iniciada en RDS	8
Figura 3. Creación de instancia de la base de datos en workBench.....	8
Figura 4. Ejecución del script para crear base de datos.....	11
Figura 5. Base de datos creada sobre la conexión	12
Figura 6. Creación del .jar de la aplicación java.	12
Figura 7. Repositorio creado en DockerHub.	13
Figura 8. Creación archivo Dockerfile.....	13
Figura 9. Ejecución comando docker para crear contenedor archivo .jar	14
Figura 10. Subida de imagen docker a DockerHub.....	14
Figura 11. Comprobación de la ejecución aplicación en DockerHub.	15
Figura 12. Clúster creado en ECS.....	16
Figura 13. Creación de la tarea en ECS.....	16
Figura 14. Parámetros de creación de la tarea en ECS.....	17
Figura 15. Tarea creada en el ECS.....	17
Figura 16. Tarea corriendo en ECS.....	18
Figura 17. Propiedades de la tarea en ECS.....	18
Figura 18. Logs en CloudWatch de los servicios utilizados en AWS.	19
Figura 19. Registros de la tarea en CloudWatch.....	19
Figura 20. Logs de la aplicación back-end corriendo en la tarea.	20
Figura 21. Creación del API en API Gateway (REST API).....	21
Figura 22. Datos a establecer para crear el API.....	21
Figura 23. Creación de un recurso en el API.....	22
Figura 24. Creación de un recurso en el API estableciendo la ruta raíz.	22
Figura 25. Creación del metodo HTTP en el recurso.	23
Figura 26. Selección del método en el recurso creado.	23
Figura 27. Establecimiento del endpoint en el método creado.....	24

Figura 28. Punto de enlace (endpoint) generado por el API.	24
Figura 29. Compilación del proyecto Angular para producción.	25
Figura 30. Creación del bucket en S3.	26
Figura 31. Bucket creado en S3.	27
Figura 32. Carga de los archivos de la carpeta “dist” al S3.	27
Figura 33. Selección de los archivos de la carpeta “dist” para subir al S3.	28
Figura 34. Archivos cargados al S3.	28
Figura 35. Instancia de CloudFront creada.	29
Figura 36. Borrado de cache en la instancia CloudFront.	29
Figura 37. Eliminación de la base de datos en RDS.	31
Figura 38. Eliminación del cluster en ECS.	32
Figura 39. Eliminación del API en API Gateway.	32
Figura 40. Eliminación del bucket en S3.	33
Figura 41. Eliminación de la instancia en CloudFront.	33
Figura 42. Proceso para abrir la consola del navegador.	34
Figura 43. Ejemplo de caída del servicio de autenticación.	35

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.

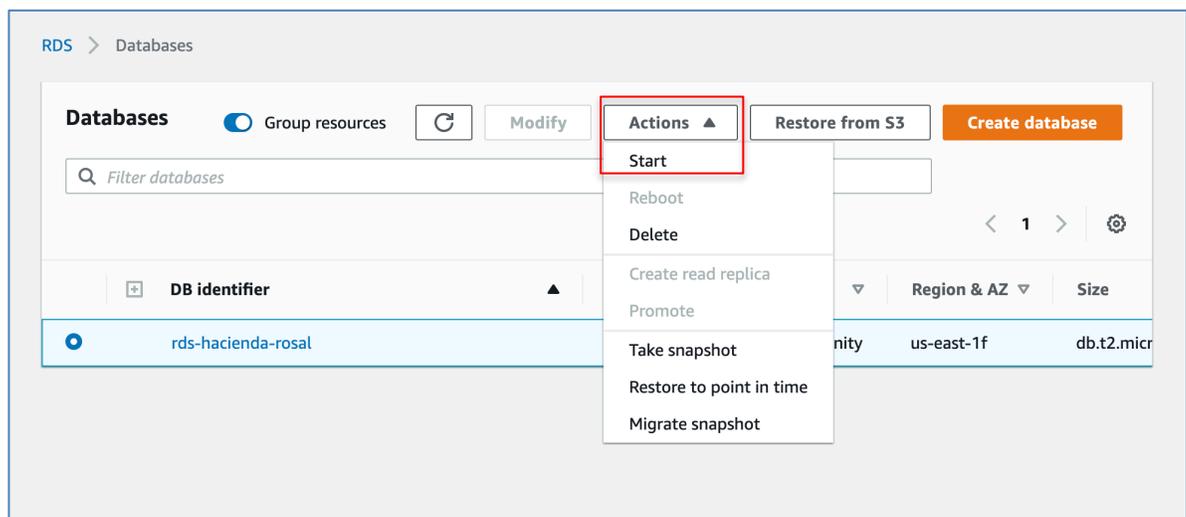
4.1. INSTALACIÓN DE LA BASE DE DATOS

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.

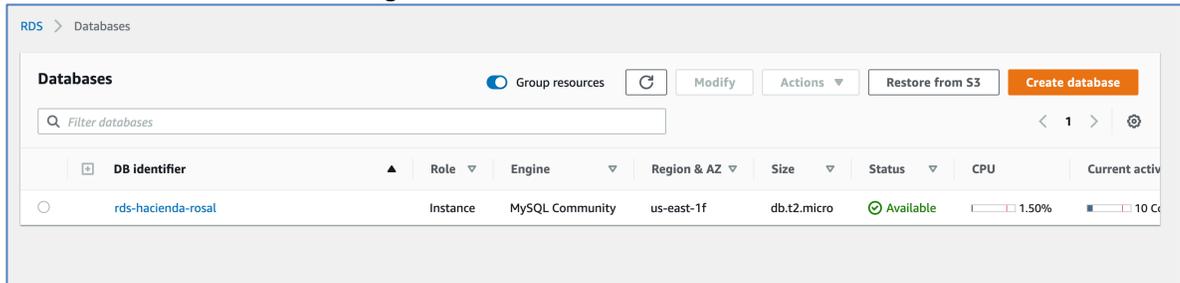
Figura 1. Inicio de base de datos en RDS



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 Workbench (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:

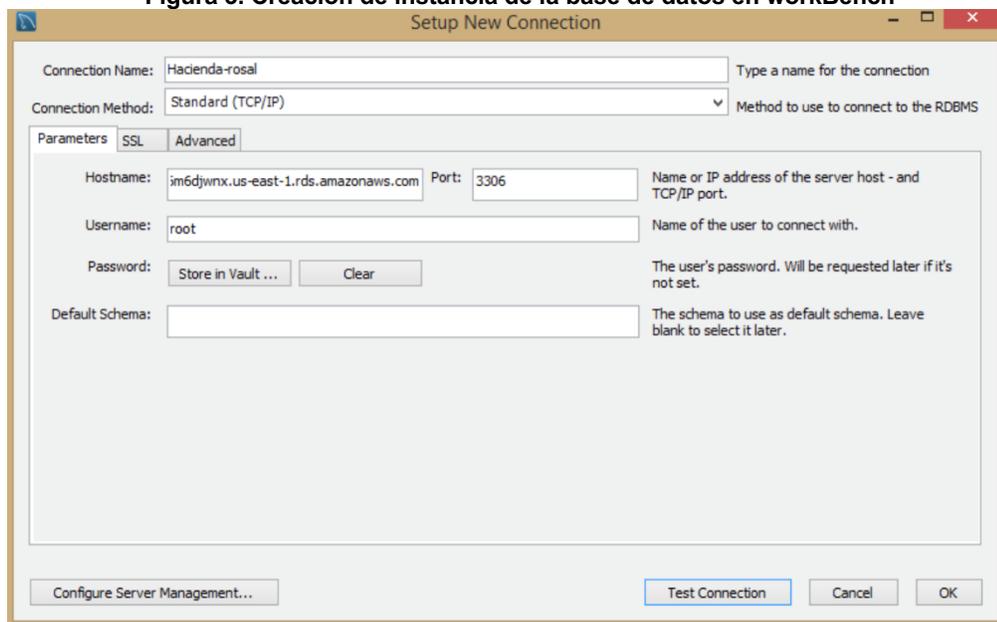
User: root

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_c_ibfk_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)
);*

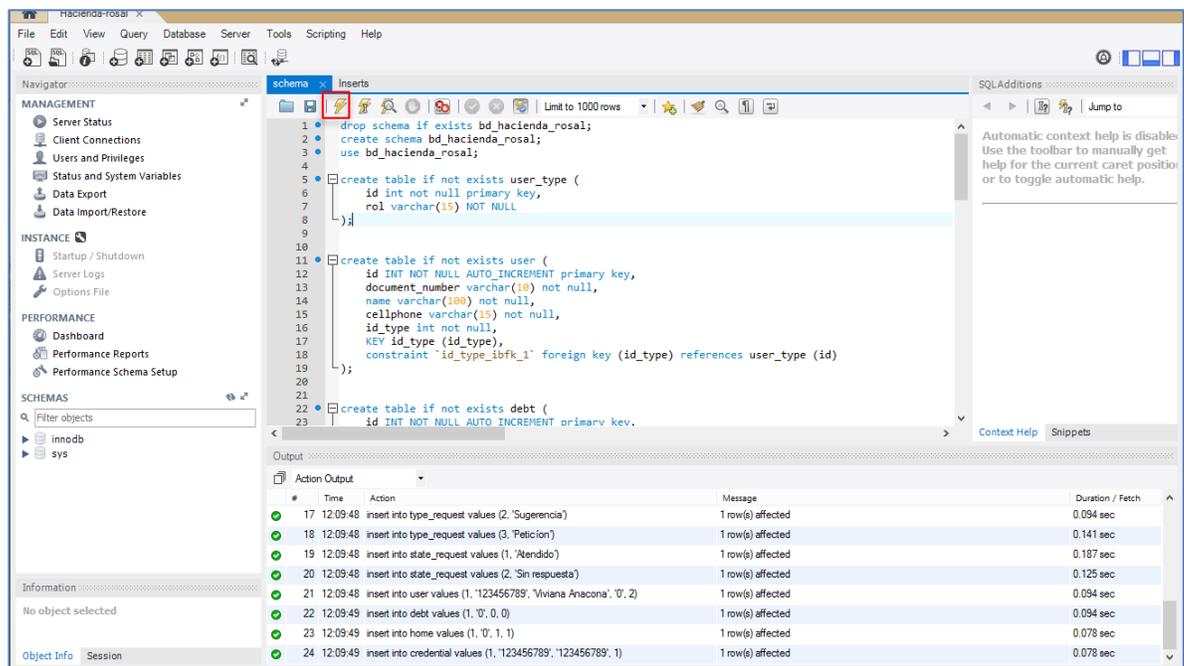
*insert into user_type values (1, 'Residente');
insert into user_type values (2, 'Administrador');*

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

*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);*

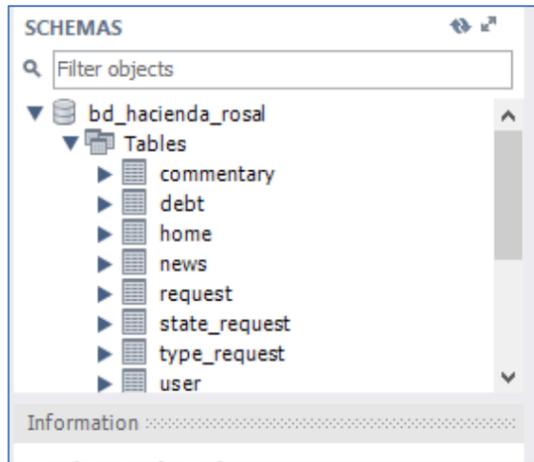
Figura 4. Ejecución del script para crear base de datos



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

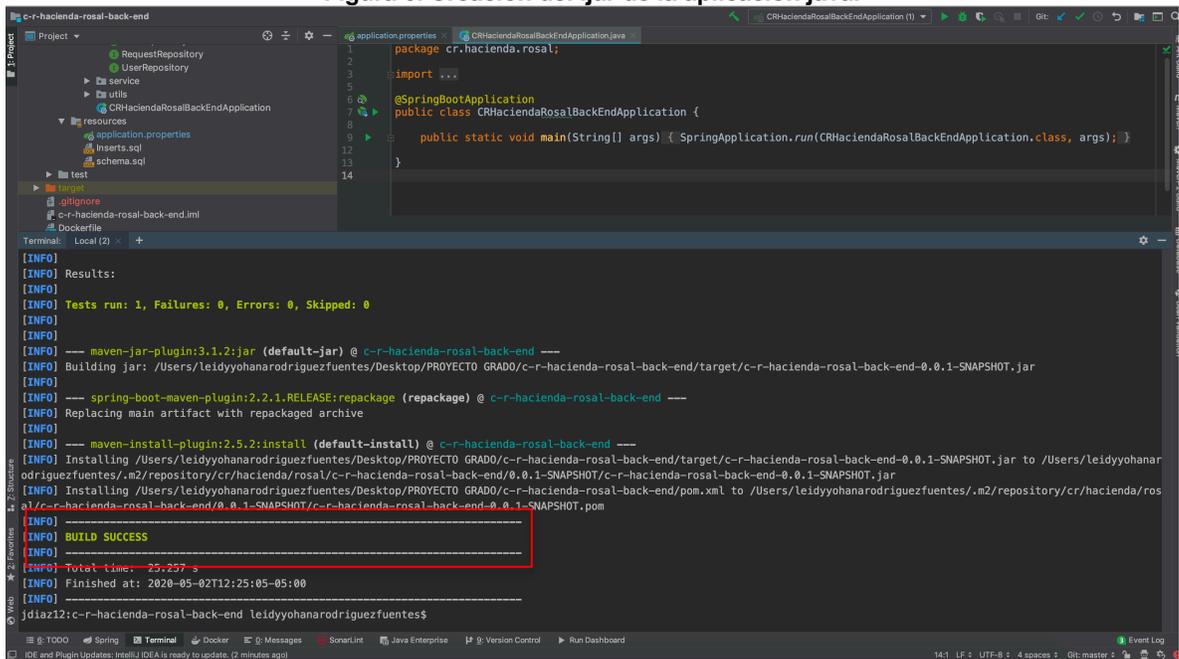


Fuente: elaboración propia

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.

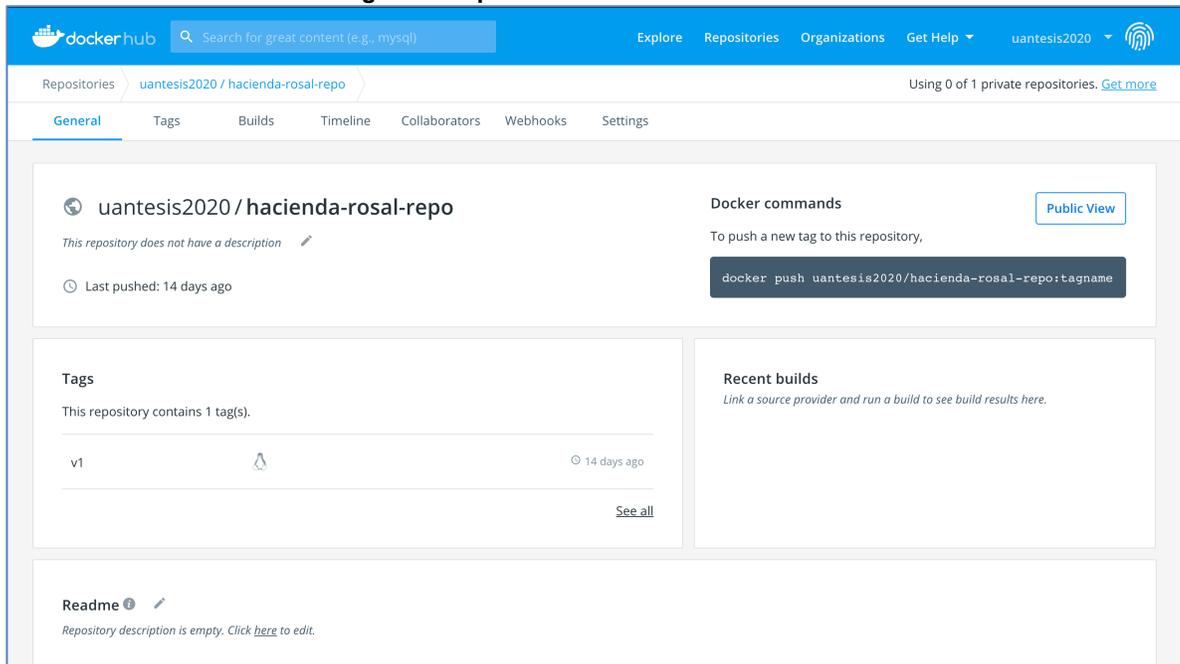
Figura 6. Creación del .jar de la aplicación java.



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

```
Terminal: Local (2) × +
jldiaz12:c-r-hacienda-rosal-back-end leidyohanarodriguezfuentes$ docker build -t hacienda-rosal .
Sending build context to Docker daemon 138MB
Step 1/5 : FROM java:8-jdk-alpine
----> 3fd9dd82815c
Step 2/5 : COPY ./target/c-r-hacienda-rosal-back-end-0.0.1-SNAPSHOT.jar /usr/app/
----> a7ba4ed6eb80
Step 3/5 : WORKDIR /usr/app
----> Running in 3e5718597664
Removing intermediate container 3e5718597664
----> fb405b65b0d0
Step 4/5 : EXPOSE 8080
----> Running in 3e16cb48793b
Removing intermediate container 3e16cb48793b
----> 7fc54ced980c
Step 5/5 : ENTRYPOINT ["java", "-jar", "c-r-hacienda-rosal-back-end-0.0.1-SNAPSHOT.jar"]
----> Running in 5cc620a8befb
Removing intermediate container 5cc620a8befb
----> 4ed9725d24fb
Successfully built 4ed9725d24fb
Successfully tagged hacienda-rosal:latest
```

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.

```
jldiaz12:c-r-hacienda-rosal-back-end leidyohanarodriguezfuentes$ docker login
Authenticating with existing credentials...
Login Succeeded
jldiaz12:c-r-hacienda-rosal-back-end leidyohanarodriguezfuentes$ docker tag hacienda-rosal uantesis2020/hacienda-rosal-repo:v1
jldiaz12:c-r-hacienda-rosal-back-end leidyohanarodriguezfuentes$ docker push uantesis2020/hacienda-rosal-repo:v1
The push refers to repository [docker.io/uantesis2020/hacienda-rosal-repo]
5e39c43fbe9f: Pushed
a1e7033f082e: Mounted from library/java
78075328e0da: Mounted from library/java
9f8566ee5135: Mounted from library/java
v1: digest: sha256:3fb5015f092d166d5426c693895eb572b75e451abe6708cde40d32a0910fab07 size: 1160
```

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 funcionando correctamente dentro del contenedor de docker en DockerHub, como se observa en Figura 11.

Figura 11. Comprobación de la ejecución aplicación en DockerHub.

```

jdiaz12@c-r-hacienda-rosal-back-end leidyhohanarodriguezfuentess$ docker run -p 4000:3000 uantesis2020/hacienda-rosal-repo:v1

:: Spring Boot ::
               (v2.2.1.RELEASE)

2020-05-02 17:59:16.041 INFO 1 --- [main] c.h.r.CRHaciendaRosalBackEndApplication : Starting CRHaciendaRosalBackEndApplication v0.0.1-SNAPSHOT on 8cc70c70fa49 with PID
1 (/usr/app/c-r-hacienda-rosal-back-end-0.0.1-SNAPSHOT.jar started by root in /usr/app)
2020-05-02 17:59:16.053 INFO 1 --- [main] c.h.r.CRHaciendaRosalBackEndApplication : No active profile set, falling back to default profiles: default
2020-05-02 17:59:18.312 INFO 1 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data repositories in DEFAULT mode.
2020-05-02 17:59:18.606 INFO 1 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 264ms. Found 8 repository interfaces.
2020-05-02 17:59:19.740 INFO 1 --- [main] trationDelegatesBeanPostProcessorChecker : Bean 'org.springframework.transaction.annotation.ProxyTransactionManagementConfigur
ation' of type [org.springframework.transaction.annotation.ProxyTransactionManagementConfiguration] is not eligible for getting processed by all BeanPostProcessors (for example: no
t eligible for auto-proxying)
2020-05-02 17:59:20.628 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-05-02 17:59:20.672 INFO 1 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-05-02 17:59:20.673 INFO 1 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.27]
2020-05-02 17:59:20.989 INFO 1 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-05-02 17:59:20.910 INFO 1 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 4646 ms
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading
of the driver class is generally unnecessary.
2020-05-02 17:59:21.278 INFO 1 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2020-05-02 17:59:21.292 WARN 1 --- [main] com.zaxxer.hikari.util.DriverDataSource : Registered driver with driverClassName=com.mysql.jdbc.Driver was not found, trying
direct instantiation.
2020-05-02 17:59:24.612 INFO 1 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2020-05-02 17:59:24.950 INFO 1 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2020-05-02 17:59:25.197 INFO 1 --- [main] org.hibernate.Version : HHH000412: Hibernate Core [5.4.8.Final]
2020-05-02 17:59:25.666 INFO 1 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations [5.1.0.Final]
2020-05-02 17:59:26.235 INFO 1 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL57Dialect
2020-05-02 17:59:33.913 INFO 1 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000499: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.
platform.internal.NoJtaPlatform]
2020-05-02 17:59:33.941 INFO 1 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'

```

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.

The screenshot shows the AWS ECS Clusters console. At the top, there's a 'Clusters' header with a description and a 'Create Cluster' button. Below that, there's a 'View' section with 'list' and 'card' options. The main content area shows the cluster 'c-r-hacienda-rosal' with a 'FARGATE' type. It displays a summary of resources: 0 Services, 1 Running tasks, and 0 Pending tasks. Below this, there's a section for 'EC2' resources: 0 Services, 0 Running tasks, and 0 Pending tasks. There are also two 'No data' boxes for 'CPUUtilization' and 'MemoryUtilization', and a '0' for 'Container instances'.

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.

The screenshot shows the AWS ECS console for the 'c-r-hacienda-rosal' cluster. The 'Tasks' tab is selected, and the 'Run new Task' button is highlighted with a red box. The console shows the cluster's status as 'ACTIVE' and provides a detailed view of resources: 0 Registered container instances, 0 Pending tasks count (0 Fargate, 0 EC2), 0 Running tasks count (0 Fargate, 0 EC2), 0 Active service count (0 Fargate, 0 EC2), and 0 Draining service count (0 Fargate, 0 EC2). There are also buttons for 'Update Cluster', 'Delete Cluster', 'Stop', 'Stop All', and 'Actions'. The 'Desired task status' is set to 'Running'. A table at the bottom shows 'No results' for the task list.

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.

The screenshot shows the 'Create Task' configuration page in the AWS ECS console. The form is divided into several sections:

- Task Definition:** 'hacienda-rosal-task2'
- Platform version:** 'LATEST'
- Cluster:** 'c-r-hacienda-rosal'
- Number of tasks:** '1'
- Task Group:** (empty)
- VPC and security groups:**
 - Cluster VPC:** 'vpc-09f9f2895be0a8 (10.88.1.0/25) | h...'
 - Subnets:** 'subnet-0734285b21cdd5c1c (10.88.1.0/28) | hc-dev-dmz-0 - us-east-1a assign ipv6 on creation: Disabled'
 - Security groups:** 'hacien-5545' with an 'Edit' button.
- Auto-assign public IP:** 'ENABLED'

 At the bottom right, there is a 'Run Task' button highlighted with a red box. A blue informational message at the bottom states: 'Tagging requires that you opt in to the new ARN and resource ID format. The IAM user/role has not opted in to the new ARN format. Opt-in to the new format to use this feature. Manage your opt-in settings.'

Fuente: elaboración propia

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

Figura 15. Tarea creada en el ECS.

The screenshot shows the AWS ECS console after a task has been successfully created. At the top, a green notification banner reads: 'Created tasks successfully' with the task IDs: ['7d50c1b0-a17d-4976-9227-47e0f8c58562']. Below this, the console shows the details for the cluster 'c-r-hacienda-rosal', which is in an 'ACTIVE' status. The cluster has 0 registered container instances, 1 pending task (Fargate), 0 running tasks, 0 active services, and 0 draining services. The 'Tasks' tab is selected, showing a table with one task in the 'PROVISIONING' state.

Task	Task definition	Container instan...	Last status	Desired status	Started By	Group	Launcher type	Platform version ...
<input type="checkbox"/>	7d50c1b0-a17d-4...	hacienda-rosal-tas...	--	PROVISIONING		family:hacienda-ro...	FARGATE	1.3.0

Fuente: elaboración propia

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

Figura 16. Tarea corriendo en ECS.

The screenshot shows the AWS ECS console interface. At the top, there are tabs for Services, Tasks, ECS Instances, Metrics, Scheduled Tasks, Tags, and Capacity Providers. Below the tabs, there are buttons for 'Run new Task', 'Stop', 'Stop All', and 'Actions'. A status indicator shows 'Desired task status: Running Stopped'. A filter box is present with 'Filter in this page' and 'Launch type ALL'. Below this is a table with columns: Task, Task definition, Container instan..., Last status, Desired status, Started By, Group, Launch type, and Platform version. One task is listed with ID '7d50c1b0-a17d-4...', definition 'hacienda-rosal-tas...', status 'RUNNING', and launch type 'FARGATE'.

Task	Task definition	Container instan...	Last status	Desired status	Started By	Group	Launch type	Platform version ...
<input type="checkbox"/>	7d50c1b0-a17d-4...	hacienda-rosal-tas...	--	RUNNING		family:hacienda-ro...	FARGATE	1.3.0

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.

The screenshot shows the 'Task : 7d50c1b0-a17d-4976-9227-47e0f8c58562' details page in the AWS ECS console. It has tabs for Details, Tags, and Logs. The 'Details' tab is active, showing various properties: Cluster (c-r-hacienda-rosal), Capacity provider (FARGATE), Launch type (FARGATE), Platform version (1.3.0), Task definition (hacienda-rosal-task:2), Group (family:hacienda-rosal-task), Task role (None), Last status (RUNNING), Desired status (RUNNING), Created at (2020-05-02 13:19:59 -0500), and Started at (2020-05-02 13:20:25 -0500). A 'Network' section shows Network mode (awsipc), ENI Id (eni-06601ab49f8afedc), Subnet Id (subnet-0734285b21ccd5c1c), Private IP (10.88.1.11), Public IP (3.231.50.222), and Mac address (02:d0:77:8d:c7:95). A 'Containers' section at the bottom shows a table with columns: Name, Container Runtime I..., Status, Image, Image Digest, CPU Units, Hard/Soft m..., Essential, and Resource ID... One container is listed with Name 'h-c-hacien...', Status 'RUNNING', and Image 'docker.io/uantesis2020/hacienda-r...'.

Name	Container Runtime I...	Status	Image	Image Digest	CPU Units	Hard/Soft m...	Essential	Resource ID...
h-c-hacien...	f81a92bace3fc6da17...	RUNNING	docker.io/uantesis2020/hacienda-r...		0	--/512	true	a3e3765f-f3...

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.

Log Groups	Insights	Expire Events After	Metric Filters	Subscriptions
<input type="radio"/> /aws/apigateway/welcome	Explore	Never Expire	0 filters	None
<input type="radio"/> /ecs/hacienda-rosal-task	Explore	Never Expire	0 filters	None
<input type="radio"/> API-Gateway-Execution-Logs_8nuv5w91y3/dev-stage	Explore	Never Expire	0 filters	None
<input type="radio"/> RDSOSMetrics	Explore	1 month (30 days)	0 filters	None

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.

Log Streams	Last Event Time
<input type="checkbox"/> ecs/h-c-hacienda-rosal/7d50c1b0-a17d-4976-9227-47e0f8c58562	2020-05-02 13:20 UTC-5
<input type="checkbox"/> ecs/h-c-hacienda-rosal/72604024-1707-41c9-bc90-5b0657053fef	2020-05-02 00:21 UTC-5
<input type="checkbox"/> ecs/h-c-hacienda-rosal/3ca5e739-0fb6-4ef2-a0a8-56c3b523490e	2020-04-18 17:23 UTC-5
<input type="checkbox"/> ecs/h-c-hacienda-rosal/0ed6189a-c782-4e29-a273-64c431ada152	2020-04-15 21:21 UTC-5
<input type="checkbox"/> ecs/h-c-hacienda-rosal/c5e43728-9455-4ec0-8b22-fd7c4b26a4df	2020-04-14 10:33 UTC-5
<input type="checkbox"/> ecs/h-c-hacienda-rosal/21bb56a3-2f01-4139-a2c2-88790c4487c8	2020-04-14 10:18 UTC-5

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.

Time (UTC -05:00)	Message
2020-05-02 18:20:39	2020-05-02 18:20:39.954 INFO 1 --- [main] trationDelegate\$BeanPostProcessorChecker : Bean 'org.springframework.transaction.annotation.ProxyTransactionManagementConfiguration' of type [org.springframework.transaction.annotation.ProxyTransactionManagementConfiguration] is not eligible for proxying by JDK dynamic proxies because no proxies can be generated for it because of the type [org.springframework.transaction.annotation.ProxyTransactionManagementConfiguration]. Please check the @ProxyConfiguration classes and the ProxyConfigurationAnnotationProcessor class in the classpath.
2020-05-02 18:20:41	2020-05-02 18:20:41.559 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-05-02 18:20:41	2020-05-02 18:20:41.652 INFO 1 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-05-02 18:20:41	2020-05-02 18:20:41.653 INFO 1 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.27]
2020-05-02 18:20:41	2020-05-02 18:20:41.959 INFO 1 --- [main] o.s.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-05-02 18:20:41	2020-05-02 18:20:41.959 INFO 1 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 9299 ms
2020-05-02 18:20:42	Loading class 'com.mysql.jdbc.Driver'. This is deprecated. The new driver class is 'com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
2020-05-02 18:20:42	2020-05-02 18:20:42.473 INFO 1 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2020-05-02 18:20:42	2020-05-02 18:20:42.550 WARN 1 --- [main] com.zaxxer.hikari.HikariDataSource : Registered driver with driverClassName=com.mysql.jdbc.Driver was not found, trying direct instantiation.
2020-05-02 18:20:44	2020-05-02 18:20:44.563 INFO 1 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2020-05-02 18:20:45	2020-05-02 18:20:45.276 INFO 1 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2020-05-02 18:20:45	2020-05-02 18:20:45.957 INFO 1 --- [main] org.hibernate.Version : HHH000412: Hibernate Core [5.4.8.Final]
2020-05-02 18:20:46	2020-05-02 18:20:46.659 INFO 1 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations [5.1.0.Final]
2020-05-02 18:20:47	2020-05-02 18:20:47.461 INFO 1 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL57Dialect
2020-05-02 18:20:51	2020-05-02 18:20:51.550 INFO 1 --- [main] o.h.e.t.j.p.l.JpaPlatformInitiator : HHH000403: Using JpaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2020-05-02 18:20:51	2020-05-02 18:20:51.559 INFO 1 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2020-05-02 18:20:55	2020-05-02 18:20:55.165 WARN 1 --- [main] JpaBaseConfiguration\$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view = false if you need to suppress this behavior.
2020-05-02 18:20:55	2020-05-02 18:20:55.753 INFO 1 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-05-02 18:20:57	2020-05-02 18:20:57.665 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2020-05-02 18:20:57	2020-05-02 18:20:57.667 INFO 1 --- [main] c.h.r.CRHaciendaRosalBackEndApplication : Started CRHaciendaRosalBackEndApplication in 29.301 seconds (JVM running for 31.889)
2020-05-02 18:23:57	2020-05-02 18:23:57.164 INFO 1 --- [nio-8080-exec-1] o.s.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2020-05-02 18:23:57	2020-05-02 18:23:57.164 INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2020-05-02 18:23:57	2020-05-02 18:23:57.173 INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 9 ms

Fuente: elaboración propia

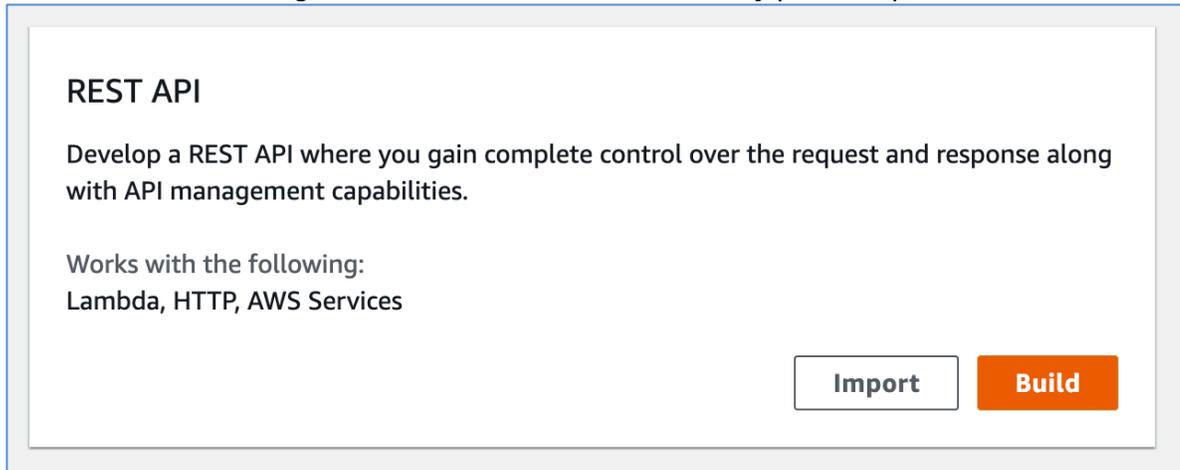
4.4. CREACIÓN DEL API EN EL SERVICIO API GATEWAY

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.

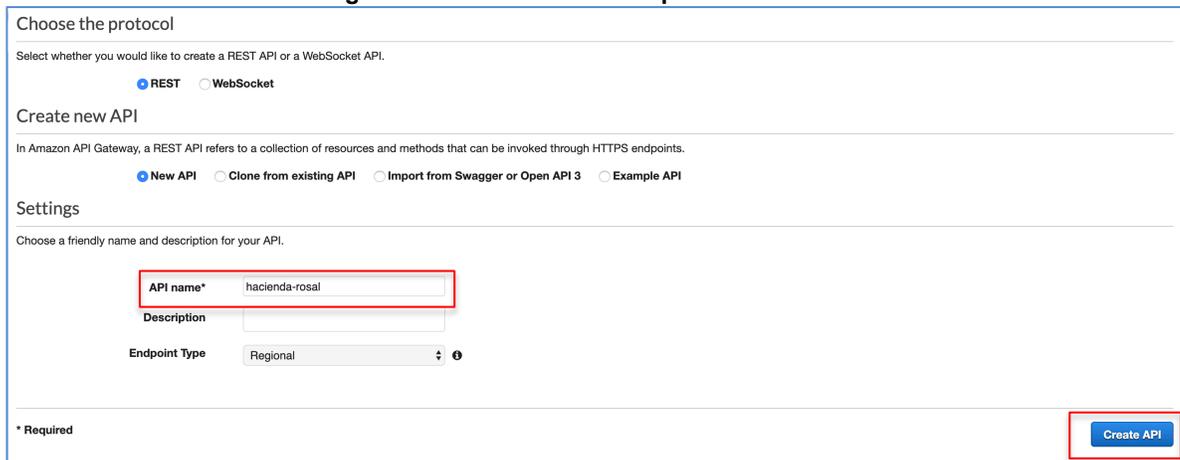
Figura 21. Creación del API en API Gateway (REST API).



Fuente: elaboración propia

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

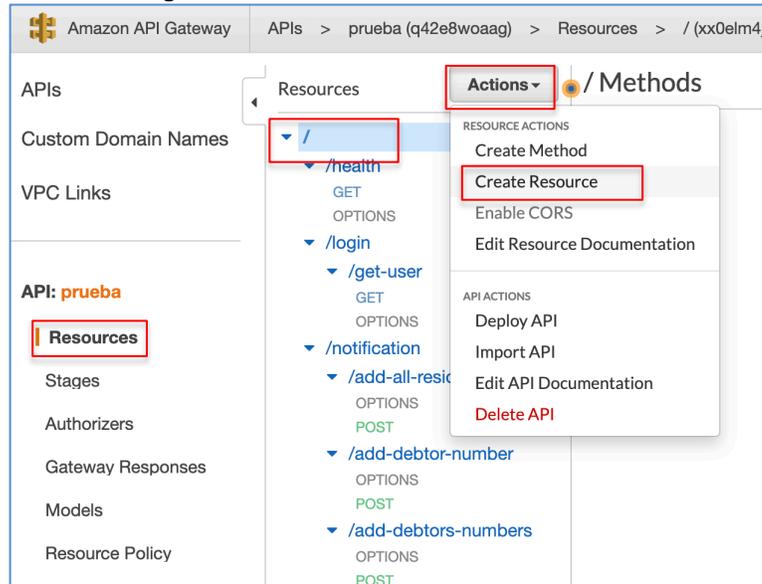
Figura 22. Datos a establecer para crear el API.



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.

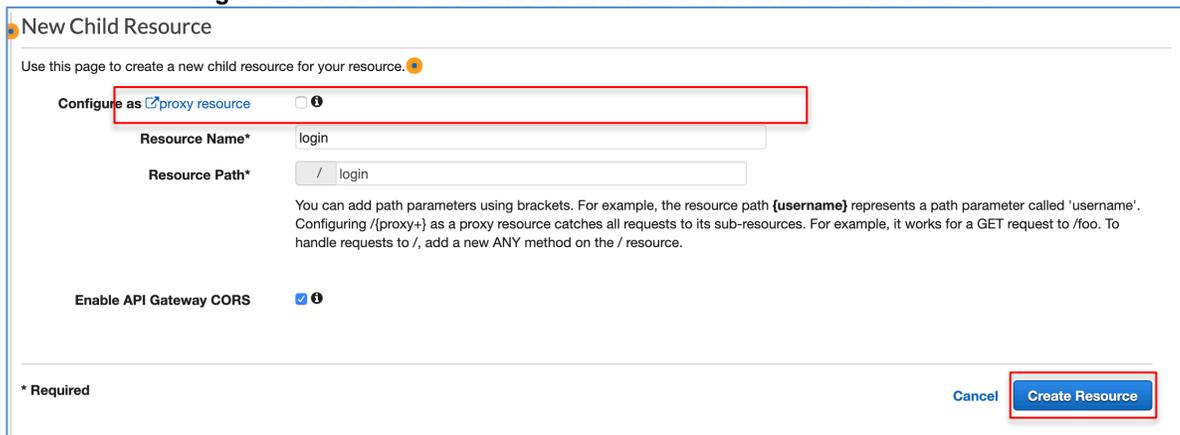
Figura 23. Creación de un recurso en el API.



Fuente: elaboración propia

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

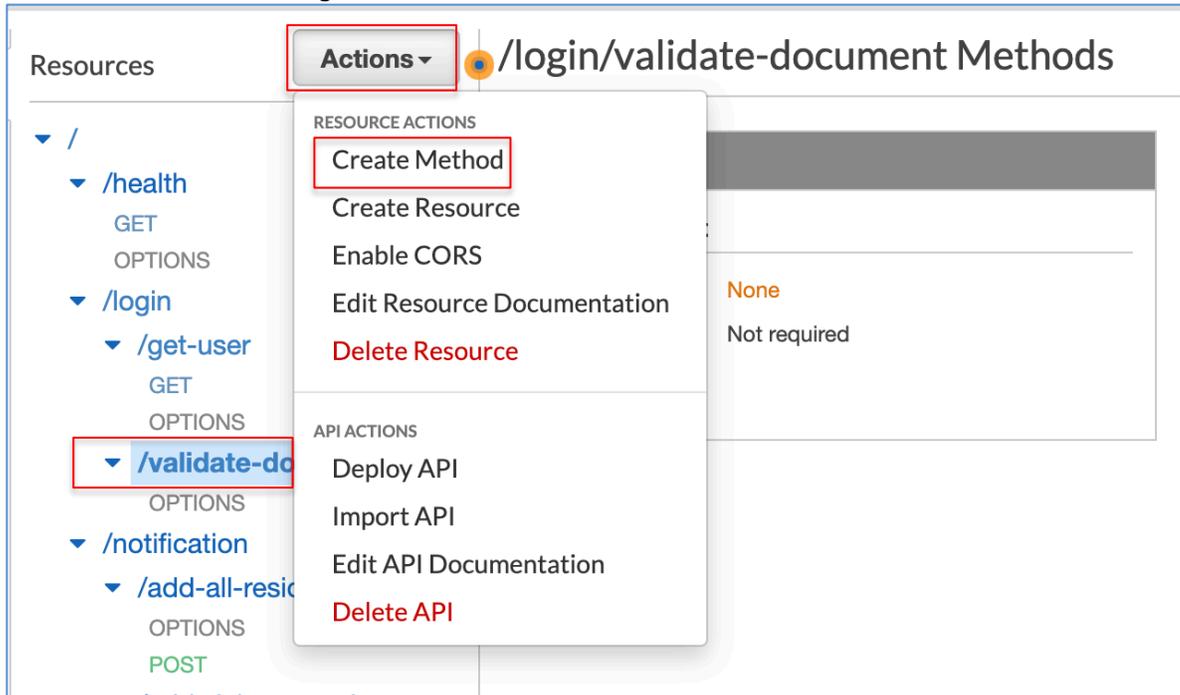
Figura 24. Creación de un recurso en el API estableciendo la ruta raíz.



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.

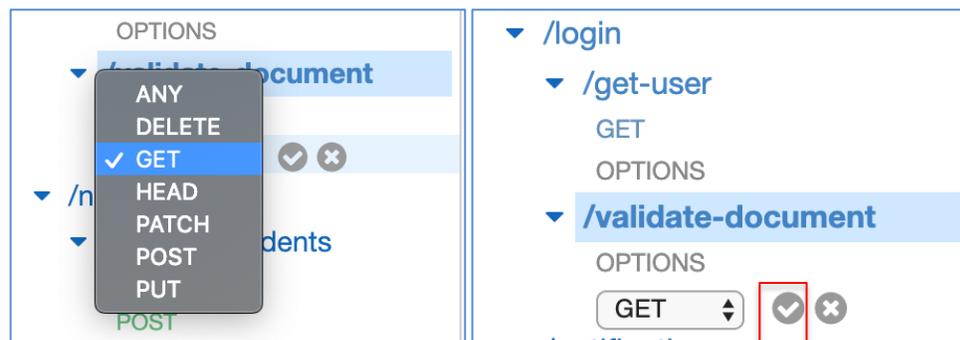
Figura 25. Creación del método HTTP en el recurso.



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.

The screenshot shows the 'Setup' page for a new method in the AWS API Gateway console. The title is '/login/validate-document - GET - Setup'. Below the title, it says 'Choose the integration point for your new method.' There are several configuration options: 'Integration type' with radio buttons for 'Lambda Function', 'HTTP' (selected), 'Mock', 'AWS Service', and 'VPC Link'; 'Use HTTP Proxy integration' (unchecked); 'HTTP method' dropdown set to 'GET'; 'Endpoint URL' text box containing 'http://3.231.50.222/login/validate-docun'; 'Content Handling' dropdown set to 'Passthrough'; and 'Use Default Timeout' checkbox (checked). A blue 'Save' button is located in the bottom right corner.

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.

The screenshot shows the 'hacienda-rosal Stage Editor' in the Amazon API Gateway console. The breadcrumb navigation is 'APIs > prueba (q42e8woaag) > Stages > hacienda-rosal'. The 'Invoke URL' is displayed as 'https://q42e8woaag.execute-api.us-east-1.amazonaws.com/hacienda-rosal'. Below this, there are tabs for 'Settings', 'Logs/Tracing', 'Stage Variables', 'SDK Generation', 'Export', 'Deployment History', 'Documentation History', and 'Canary'. The 'Settings' tab is active, showing 'Cache Settings' with 'Enable API cache' unchecked, 'Default Method Throttling' with a note about the current account level throttling rate of 10000 requests per second, and 'Enable throttling' checked with a rate of 10000 requests per second and a burst of 5000 requests.

Fuente: elaboración propia

4.5. DESPLIEGUE DE LA APLICACIÓN FRONT-END EN AWS

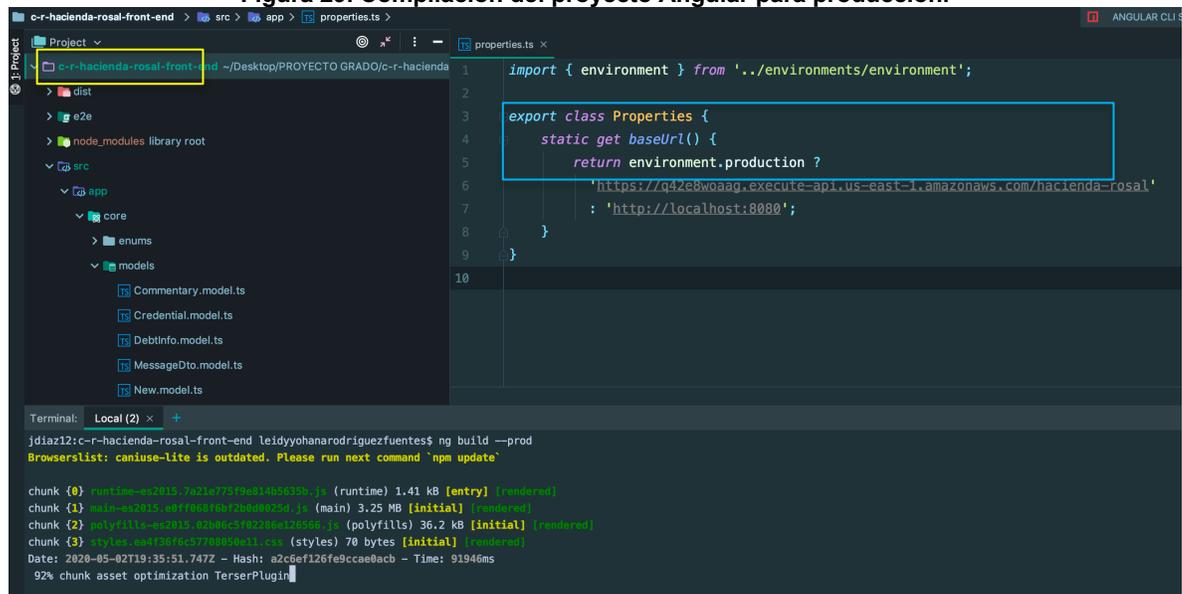
Para poder hacer este despliegue es necesario realizar los siguientes pasos:

- 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.

General configuration

Bucket name
hacienda-rosal

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

Region
US East (N. Virginia) us-east-1

Bucket settings for Block Public Access

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

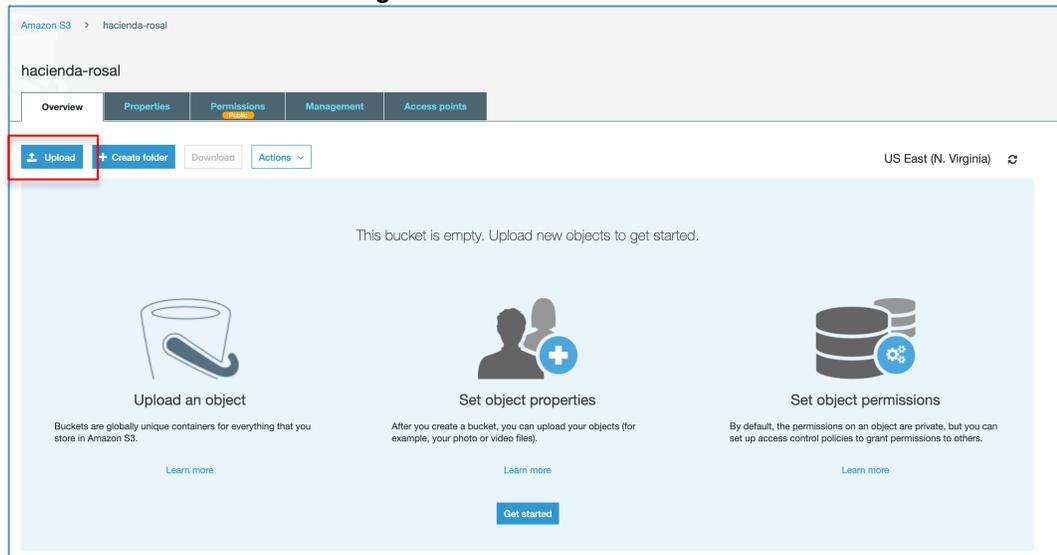
► **Advanced settings**

Cancel **Create bucket**

Fuente: elaboración propia

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

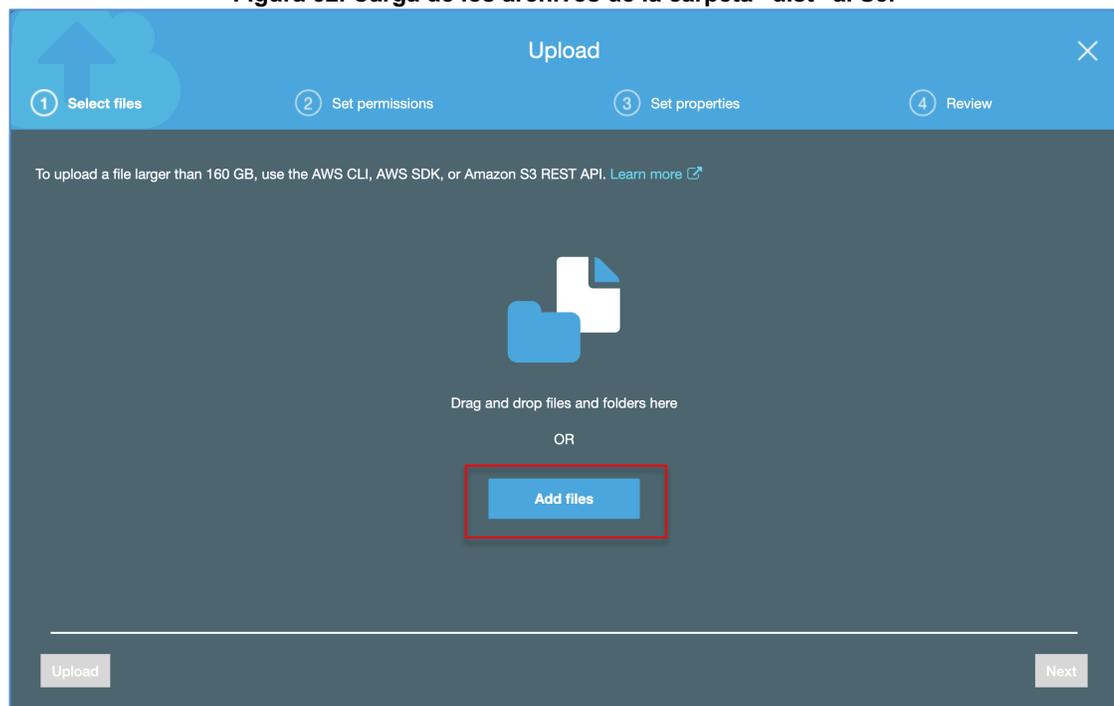
Figura 31. Bucket creado en S3.



Fuente: elaboración propia

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

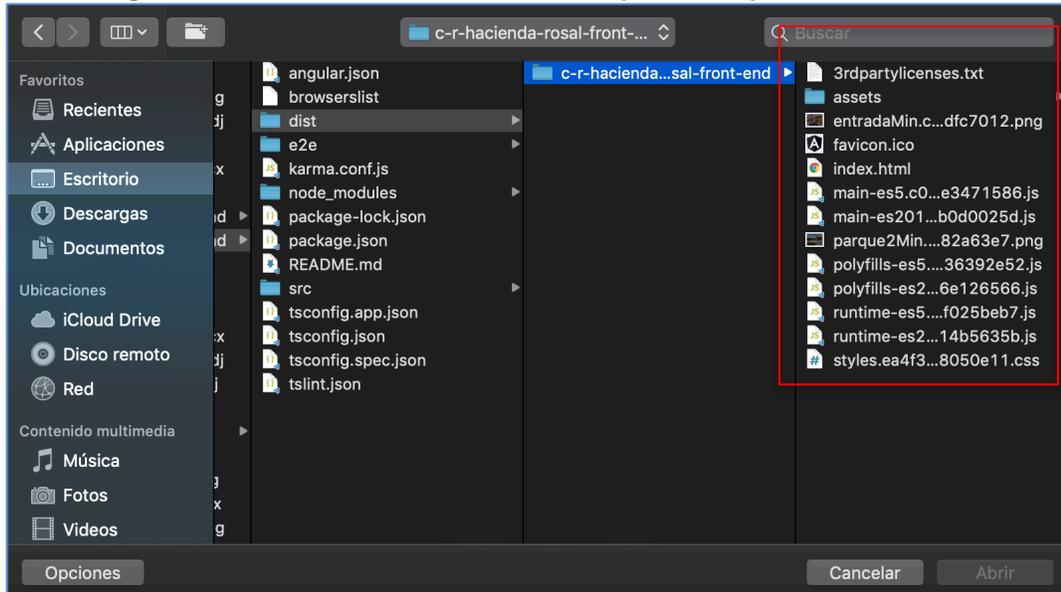
Figura 32. Carga de los archivos de la carpeta “dist” al S3.



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.

hacienda-rosal				
Overview Properties Permissions Management Access points				
<input type="text" value="Type a prefix and press Enter to search. Press ESC to clear."/>				
<input type="button" value="Upload"/> <input type="button" value="Create folder"/> <input type="button" value="Download"/> <input type="button" value="Actions"/>				US East (N. Virginia)
Viewing 1 to 13				
<input type="checkbox"/>	Name	Last modified	Size	Storage class
<input type="checkbox"/>	assets	--	--	--
<input type="checkbox"/>	3rdpartylicenses.txt	May 2, 2020 2:56:29 PM GMT-0500	37.1 KB	Standard
<input type="checkbox"/>	entradaMin.cb3f23b235a76dfc7012.png	May 2, 2020 2:56:32 PM GMT-0500	240.4 KB	Standard
<input type="checkbox"/>	favicon.ico	May 2, 2020 2:56:28 PM GMT-0500	948.0 B	Standard
<input type="checkbox"/>	index.html	May 2, 2020 2:56:28 PM GMT-0500	2.2 KB	Standard
<input type="checkbox"/>	main-es2015.e0f068f6bf2b0d0025d.js	May 2, 2020 2:56:43 PM GMT-0500	3.3 MB	Standard
<input type="checkbox"/>	main-es5.c0744d7951bde3471586.js	May 2, 2020 2:56:42 PM GMT-0500	3.6 MB	Standard
<input type="checkbox"/>	parque2Min.bf1ecab64c89182a63e7.png	May 2, 2020 2:56:32 PM GMT-0500	213.4 KB	Standard
<input type="checkbox"/>	polyfills-es2015.02b06c5f02286e126566.js	May 2, 2020 2:56:32 PM GMT-0500	36.2 KB	Standard
<input type="checkbox"/>	polyfills-es5.1c07da2168c236392e52.js	May 2, 2020 2:56:31 PM GMT-0500	112.8 KB	Standard
<input type="checkbox"/>	runtime-es2015.7a21e775f9e814b5635b.js	May 2, 2020 2:56:33 PM GMT-0500	1.4 KB	Standard
<input type="checkbox"/>	runtime-es5.96ca82998da1f025beb7.js	May 2, 2020 2:56:33 PM GMT-0500	1.4 KB	Standard
<input type="checkbox"/>	styles.ea4f36f6c57708050e11.css	May 2, 2020 2:56:33 PM GMT-0500	70.0 B	Standard

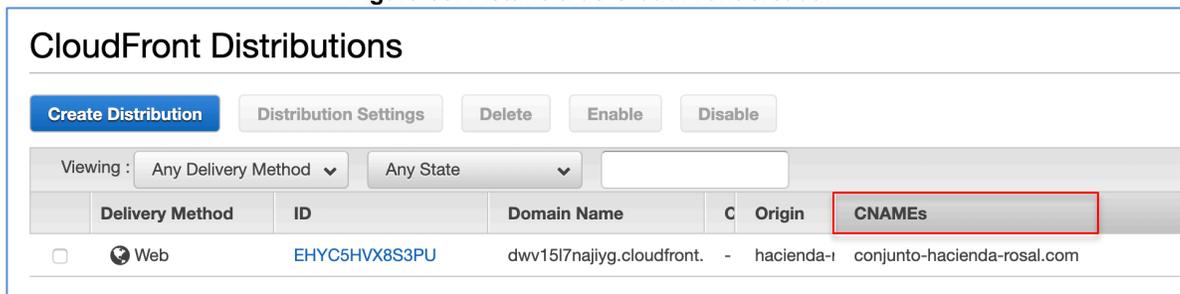
Fuente: elaboración propia

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.

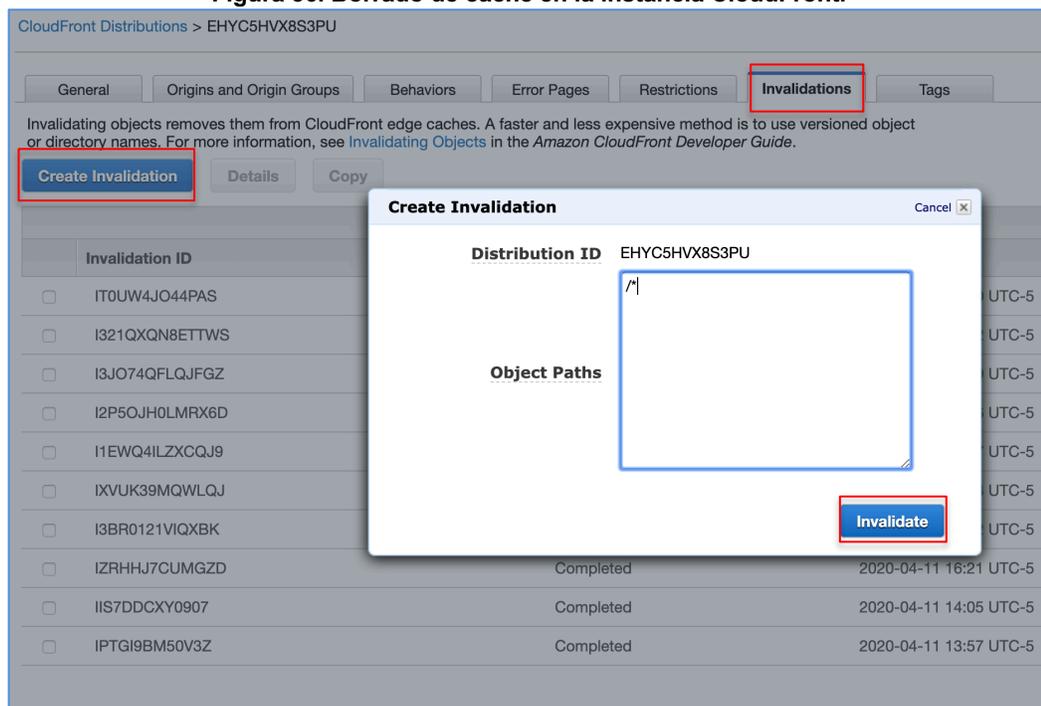
Luego de crear el dominio en CloudFront, se verá la instancia como se observa en la Figura 35. El dominio de la aplicación es <https://conjunto-hacienda-rosal.com>.

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.

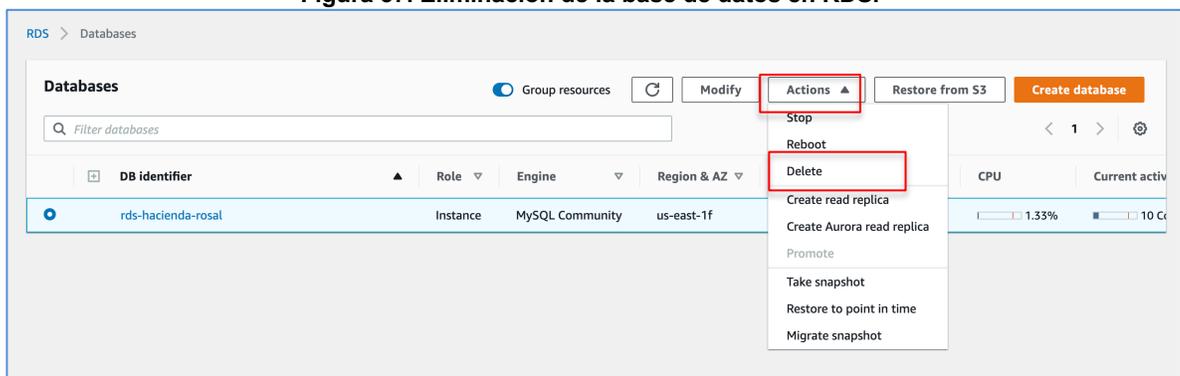
5. DESINSTALACIÓN DEL SISTEMA

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.

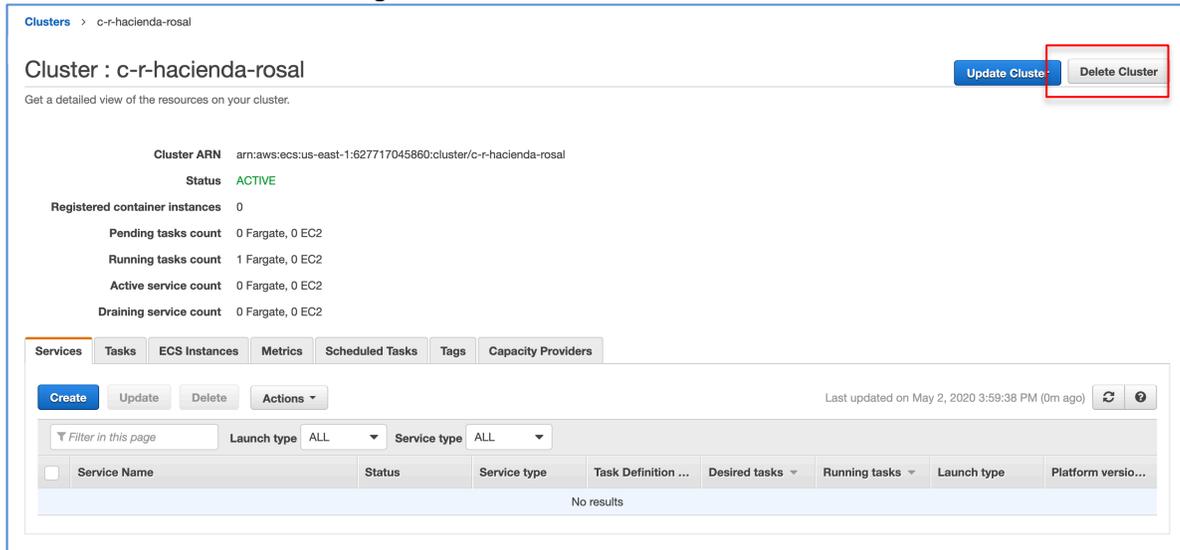


Fuente: elaboración propia

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.

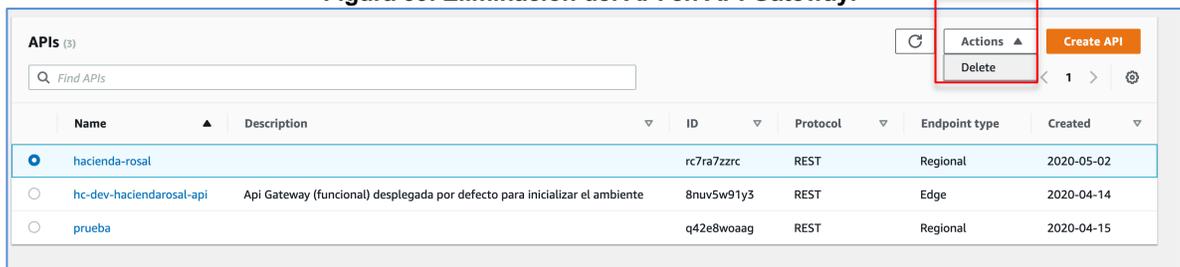


Fuente: elaboración propia

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.

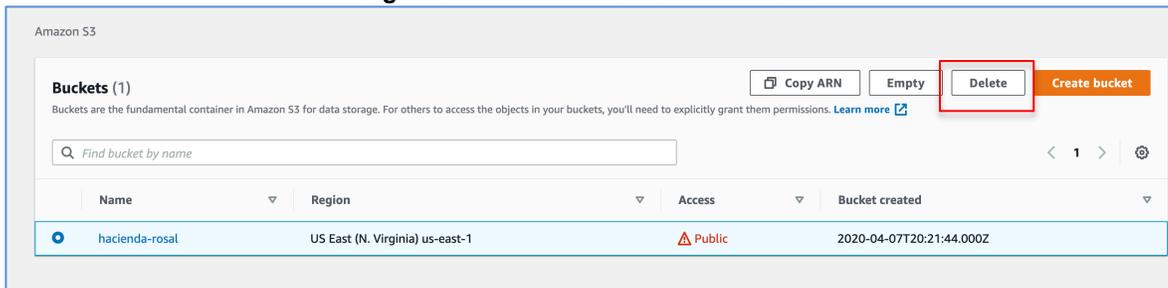


Fuente: elaboración propia

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.

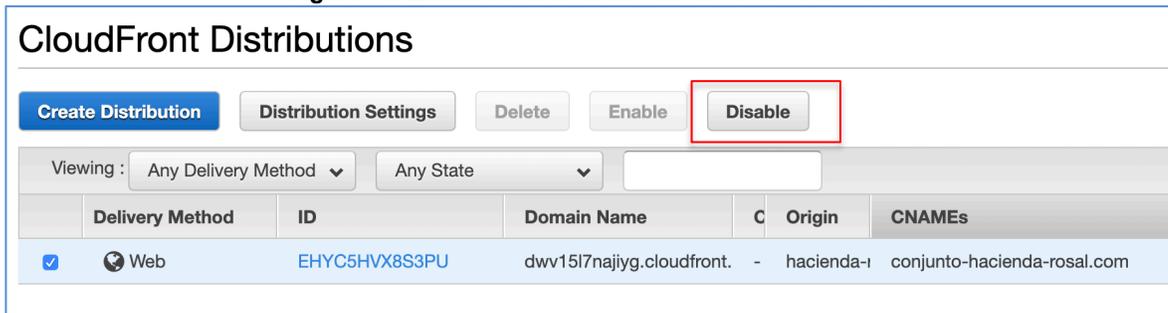


Fuente: elaboración propia

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

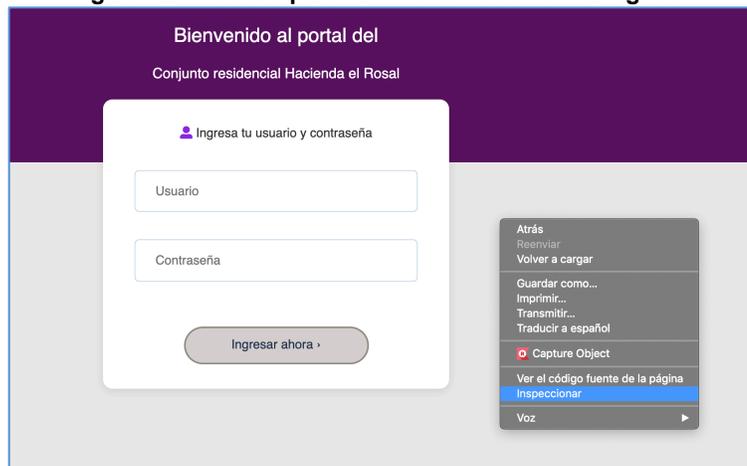
A continuación, se nombran algunos posibles errores de infraestructura que se pueden dar en la aplicación web y la solución.

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.



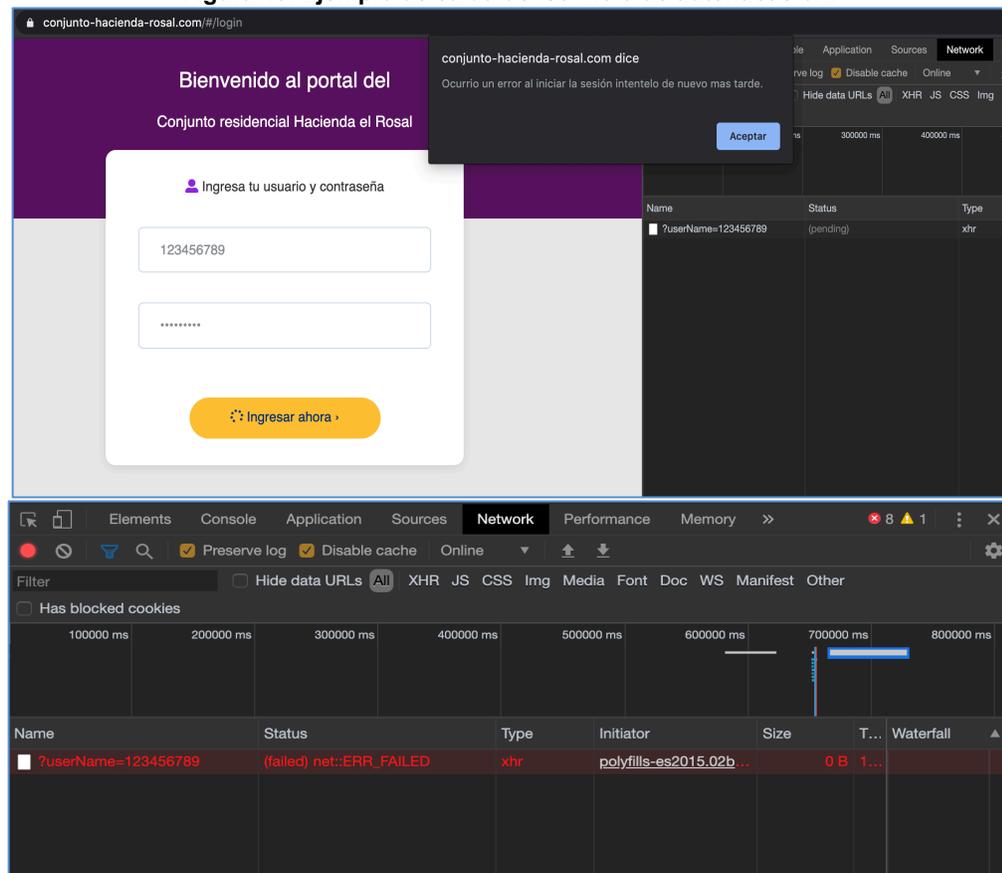
Fuente: elaboración propia

- 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 código 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 caída de servicio que toca solucionar desde la infraestructura. De esta forma se comprueba si hay caída 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.