

Componente 1. Entregable 1.8.10 Ministerio de Minas y Energía

Proyecto: CTO No. 116231-225-2024

Realizar el cargue de los datos generados al sistema, cumpliendo los esquemas de georreferenciación definidos.

Fecha: julio de 2025
Elaborado por, ADA S.A.S.
Versión: 7.0

Contenido

1. Introducción	6
2. Carga de Datos a la Base de Datos	6
2.1 Uso de MAGNA y conversión a EPSG:4326	7
2.2 Extracción y Preparación de los Datos	7
2.3 Verificación y Control de la Información	8
2.4 Evidencias de Carga a la Base de Datos	9
3. Procedimiento para montar la base de datos	11
3.1 Prerrequisitos	11
3.2 Permisos adecuados	12
3.3 Esquema predefinido	12
3.3.1 Identificación de principales entidades	12
3.3.2 Definición de tablas y campos	13
3.3.3 Modelo Entidad-Relación (ER) Detallado para el Aplicativo Web	14
3.3.3.1 Entidades Principales	14
3.3.3.2 Relaciones	15
3.3.3.3 Diagrama simplificado	16
4. Consideraciones técnicas	16
4.1 Normalización	16
4.2 Índices	17
4.3 Restricciones de Integridad Referencial	17
5. Validación por parte del DBA	17
5.1 Compatibilidad de los geodatos	18
6. Ejecución de scripts	18
6.1 Script Create_Table.sql: creación de las estructuras de base de datos	19
6.1.1 Iniciar PGAdmin	19
6.1.2 Conectarse al servidor deseado	19
6.1.3 Abrir la herramienta de consultas	19
6.1.4 Cargar el archivo del script	19
6.1.5 Ejecutar el script	19
6.1.6 Verificar que las tablas se hayan creado correctamente	20
6.2 Script div_territorial_zonas_rows.sql: inserción de zonas y datos territoriales	20
6.2.1 Iniciar PGAdmin y conectarse al servidor	20
6.2.2 Seleccionar la base de datos adecuada	20
6.2.3 Abrir una nueva ventana de consulta	20
6.2.4 Cargar el archivo div_territorial_zonas_rows.sql	20

6.2.5 Ejecutar el script.....	20
6.2.6 Validar que los datos hayan sido insertados correctamente	21
6.3 Carga y ejecución de scripts determinantes: una guía paso a paso	21
6.3.1 Iniciar PGAdmin y conectarse al servidor	21
6.3.2 Seleccionar la base de datos adecuada	21
6.3.3 Cargar el script SQL desde el archivo correspondiente	22
6.3.4 Ejecutar el script y generar los datos.....	22
6.3.5 Visualizar y verificar los resultados de la ejecución	22
6.4 Generación de territorios a nivel municipal.....	22
6.4.1 Inicio de PGAdmin y conexión a la base de datos	22
6.4.2 Ubicación y selección de la base de datos	23
6.4.3 Acceder al archivo del script SQL.....	23
6.4.4 Ejecutar el script para cargar los territorios.....	23
6.4.5 Verificación de resultados y validación final.....	23
6.5 Creación de datos de indicadores: indicadores_territorial_rows.sql	23
6.5.1 Inicio de pgAdmin.....	23
6.5.2 Conexión a la base de datos	24
6.5.3 Apertura del Query Tool	24
6.5.4 Carga del script indicadores_territorial_rows.sql	24
6.5.5 Ejecución del script	24
6.5.6 Verificación de resultados	24
6.6 Carga de información geoespacial: municipios_rows_espacial.sql.....	24
6.6.1 Apertura de la herramienta de administración	25
6.6.2 Conexión a la base de datos Oracle.....	25
6.6.3 Carga del script municipios_rows_espacial.sql.....	25
6.6.4 Ejecución del script	25
6.6.5 Revisión de resultados	25
6.7 Consideraciones finales	25
7. Instalación de Docker en Linux/Ubuntu.....	26
7.1. Limpieza de versiones anteriores de Docker.....	26
7.2. Actualización de repositorios de Docker en la máquina	27
7.3. Instalación de Docker.....	29
7.4 Configuración de permisos para el usuario Docker	29
8. Despliegue de la Aplicación	31
8.1. Creación de la Imagen Docker.....	31
8.2 Código Fuente del Desarrollo.....	31
8.3 Construcción de la imagen Docker.....	32

8.4 Subir la imagen al repositorio de Docker	32
8.5 Creación del archivo docker-compose.yml	33
8.6 Cree y edite el archivo Docker -compose.yml	34
8.7 Copie el contenido del archivo docker-compose.yml desde su entorno local al servidor	35
8.8 Ejecute el despliegue	35
8.9 Configuración y Ejecución de la Aplicación	36
9. Verificación	37
9.1 Determinantes	37
9.2 Indicadores	38
10. Solución de problemas	38
10.1 Puertos bloqueados	39
10.2 Problemas con Docker Compose. Verificar la versión correcta	40
10.2.1 Puertos necesarios	41
10.2.2 Pasos para el despliegue	41
10.2.2.1 Eliminar versiones anteriores de Docker	41
10.2.2.2 Instalar los repositorios de Docker en la máquina	41
10.2.2.3 Instalar Docker y otorgar permisos	41
10.2.2.4 Desplegar la aplicación	42
11. Presentación de la información en el Aplicativo web	42
11.1 Interfaz Gráfica de Usuario	43
11.2 Filtros y Búsquedas	43
12. Importancia del proceso	44
12.1 Eficiencia	45
12.2 Escalabilidad	45
12.3 Confiabilidad	45
13. Aprobación	46

Índice de ilustraciones

Ilustración 1. Evidencias de Carga a la Base de Datos Fuente: ADA.....	9
Ilustración 2. Evidencias de Carga a la Base de Datos Fuente: ADA.....	9
Ilustración 3. Evidencias de Carga a la Base de Datos Fuente: ADA.....	10
Ilustración 4. Evidencias de Carga a la Base de Datos Fuente: ADA.....	10
Ilustración 5. Evidencias de Carga a la Base de Datos Fuente: ADA.....	10
Ilustración 6. Evidencias de Carga a la Base de Datos Fuente: ADA.....	11
Ilustración 7. Evidencias de Carga a la Base de Datos Fuente: ADA.....	11
Ilustración 8. Modelo Entidad – Relación. Fuente: ADA.	16
Ilustración 9. Script div_territorial_zonas_rows.sql: inserción de zonas y datos territoriales. Fuente ADA.	21
Ilustración 10. Limpieza de versiones anteriores de Docker. Fuente ADA.	26
Ilustración 11. Actualización de repositorios de Docker en la máquina. Fuente ADA.	27
Ilustración 12. Actualización de repositorios de Docker en la máquina. Fuente ADA.	28
Ilustración 13. Guardar en repositorio. Fuente: ADA.....	28
Ilustración 14. Actualización de repositorios de Docker en la máquina. Fuente ADA.	28
Ilustración 15 Instalación de Docker. Fuente ADA.	29
Ilustración 16. Instalación de Docker. Fuente ADA.	29
Ilustración 17. Configuración de permisos para el usuario Docker. Fuente ADA.....	30
Ilustración 18. Configuración de permisos para el usuario Docker. Fuente ADA.....	30
Ilustración 19. Creación de la Imagen Docker. Fuente ADA.....	31
Ilustración 20. Construcción de la imagen Docker. Fuente ADA.	32
Ilustración 21. Subir la imagen al repositorio de Docker. Fuente ADA.	33
Ilustración 22. Creación del archivo docker-compose.yml. Fuente ADA.	34
Ilustración 23. Cree y edite el archivo Docker -compose.yml: Fuente ADA.....	34
Ilustración 24. Ejecute el despliegue. Fuente ADA.....	35
Ilustración 25. Configuración y Ejecución de la Aplicación. Fuente ADA.....	36
Ilustración 26. Configuración y Ejecución de la Aplicación. Fuente ADA.....	36
Ilustración 27. disponibilidad del frontend. Fuente ADA.	37
Ilustración 28. disponibilidad del frontend. Fuente ADA.	38
Ilustración 29. Solución de problemas. Fuente ADA.	38
Ilustración 30. Solución de problemas. Fuente ADA.	39
Ilustración 31. Problemas con Docker Compose: Verifique la versión correcta. Fuente ADA.....	40
Ilustración 32. Desplegar la aplicación. Fuente ADA.....	42
Ilustración 33. Desplegar la aplicación. Fuente ADA.....	42
Ilustración 34. Filtros y Búsquedas. Fuente ADA.	44
Ilustración 35. Filtros y Búsquedas. Fuente ADA.	44

1. Introducción

Desarrollar un aplicativo web bajo el contrato CTO No. 116231-225-2024, suscrito entre ADA S.A.S. y el Ministerio de Minas y Energía (MME), que provea información abierta sobre indicadores y determinantes territoriales relevantes para la toma de decisiones en proyectos del sector minero-energético dentro del marco de la Transición Energética Justa - TEJ.

El sistema permite a los usuarios acceder a datos estructurados que se presentan de forma clara y comprensible, con el propósito de que cualquier persona pueda acceder a información de interés general del sector minero-energético. Además, garantiza que la aplicación sea altamente eficiente, proporcionando la información necesaria con un mínimo de costos operativos. Esto se logra mediante la integración de herramientas avanzadas de búsqueda, filtrado y análisis de datos, que permiten a los usuarios generar informes, realizar consultas detalladas y obtener visualizaciones interactivas, todo dentro de una interfaz de usuario amigable y de fácil acceso.

Los datos provendrán exclusivamente de matrices en Excel y Shapefile de municipios, departamentos y polígonos suministrados por el Ministerio de Minas y Energía - MME.

2. Carga de Datos a la Base de Datos

El proceso de carga de datos al aplicativo web del Ministerio de Minas y Energía (MME) representa una fase esencial dentro del funcionamiento integral del sistema. Este procedimiento no se limita únicamente a la transferencia técnica de archivos o registros; más bien, constituye un paso estratégico para asegurar que la información que nutre al Sistema de Información de la Estrategia de Relacionamento Territorial conserve su valor, coherencia y utilidad a lo largo del tiempo.

Cada dato que ingresa al sistema atraviesa una serie de pasos meticulosamente definidos que garantizan su integridad estructural, su correcta organización y, sobre todo, su disponibilidad para ser consultado, interpretado y utilizado en la toma de decisiones. La finalidad principal es permitir un registro preciso, una consulta fluida y un seguimiento transparente de todas las acciones y actividades que se desarrollan en el marco de la Estrategia de Relacionamento Territorial.

En este contexto, resulta fundamental que toda la información cargada incluya georreferenciación. Esta condición no es opcional, ya que el valor del dato dentro del sistema se potencia cuando se asocia con una ubicación específica. Para ello, se adopta como estándar el sistema de referencia EPSG:4326, el cual responde a las necesidades técnicas del proyecto en términos de precisión espacial, estructura de datos y compatibilidad con los sistemas existentes.

La georreferenciación habilita herramientas de análisis espacial, facilitando la visualización geográfica de las actividades lo que garantiza una trazabilidad completa desde el nivel más general hasta el detalle local. Por esta razón, el proceso de cargue debe seguir los

esquemas previamente definidos y validados, respetando en todo momento las exigencias del sistema respecto al formato, la calidad y la organización de la información espacial.

Es igualmente importante señalar que el aplicativo web no impone al MME el uso exclusivo de una plataforma o entorno de desarrollo específico para ejecutar scripts o llevar a cabo la carga de datos. Esta flexibilidad es clave, ya que reconoce la diversidad de herramientas disponibles en el entorno tecnológico actual, algunas de las cuales son de libre uso y otras requieren licenciamiento. Dentro de las alternativas recomendadas se encuentran PGAdmin, ampliamente utilizado para la gestión de bases de datos PostgreSQL, una herramienta versátil que soporta múltiples motores y facilita la administración avanzada de bases de datos.

Como buena práctica, se sugiere que el funcionario designado para ejecutar este proceso cuente con un nivel avanzado de conocimientos en administración de bases de datos. Esto asegura una ejecución precisa, reduce el riesgo de errores estructurales y permite resolver eficazmente cualquier eventualidad técnica que pueda surgir durante el proceso.

2.1 Uso de MAGNA y conversión a EPSG:4326

En el marco del presente proyecto, la información geoespacial suministrada por el Ministerio de Minas y Energía proviene de archivos en formato Shapefile y tablas con geometrías previamente referenciadas en el sistema oficial colombiano MAGNA (Marco Geocéntrico Nacional de Referencia). Este sistema, identificado internacionalmente como EPSG: 4326, se basa en el elipsoide WGS84.

Toda la información fue manejada en el sistema de referencia EPSG: 4326 (WGS84). En la práctica, esto significó que, al momento de crear las columnas geométricas en la base de datos, se definió el tipo correspondiente en las tablas (ej. En la tabla “div_territorial_municipios” se agregó el tipo MULTIPOLYGON, 4326), lo que asegura que las coordenadas se almacenen en latitud y longitud, listas para ser interpretadas por la herramienta Leaflet.

Teniendo en cuenta lo anterior, la base de datos del aplicativo almacena y publica la información en EPSG:4326, garantizando su compatibilidad, trazabilidad y coherencia con el resto de componentes del sistema.

Nota: Para mayor detalle acerca de las tablas creadas, ver el documento entregable “2.6.17 Documentación técnica del proyecto”, numeral “12. Diccionario de datos”.

2.2 Extracción y Preparación de los Datos

Antes de proceder con la carga, es indispensable realizar una etapa previa de preparación y adecuación de los datos. Esta fase implica revisar los archivos fuente, asegurarse de que contienen la información requerida y transformarlos al formato apropiado que la base de datos espera recibir. Comúnmente, estos archivos se presentan en formatos como Excel (.xlsx) o Shapefile, según el tipo de información (tabular o geográfica). Dicha información

es cargada con la misma estructura entregada desde el Ministerio de Minas y Energía (MME).

Durante esta etapa, también se realiza una limpieza inicial que incluye, por ejemplo, la normalización de campos, la eliminación de valores vacíos o inconsistentes, y la verificación de coordenadas en caso de tratarse de datos espaciales. El objetivo es minimizar los errores al momento de la carga y garantizar que la información esté lista para integrarse al sistema de forma fluida.

La información que se revisó y se cargó al sistema durante este proceso fueron los siguientes:

- **Lista_recurrentes**
 - Las hojas del documento usadas para la carga de las listas recurrentes fueron: TablaApoyo_1, TablaApoyo_4 y TablaApoyo_6. Contiene los datos de las consultas más frecuentes.
- **Apoyo6: Hoja**
 - La hoja del documento usada para la carga de estos datos es la “Hoja1”. Contiene datos para los indicadores socioeconómicos.
- **2_TablaApoyo**
 - Las hojas del documento usadas para la carga de los datos fueron: TablaApoyo_1, TablaApoyo_2, TablaApoyo_3, TablaApoyo_4, TablaApoyo_5, y TablaApoyo_6. Contiene la información de la grilla para el cargue de los polígonos con sus respectivos datos (ej. Nombre Departamento, Nombre Municipio, código departamento, código municipio, etc).
- **1_Glosario_DataTER_Determinantes**
 - Las hojas del documento usadas para la carga de los datos fueron: Socioeconómico, Determinantes y TablasApoyoDicc. Contiene las descripciones de las variables de los determinantes (ej: PIB por habitante).

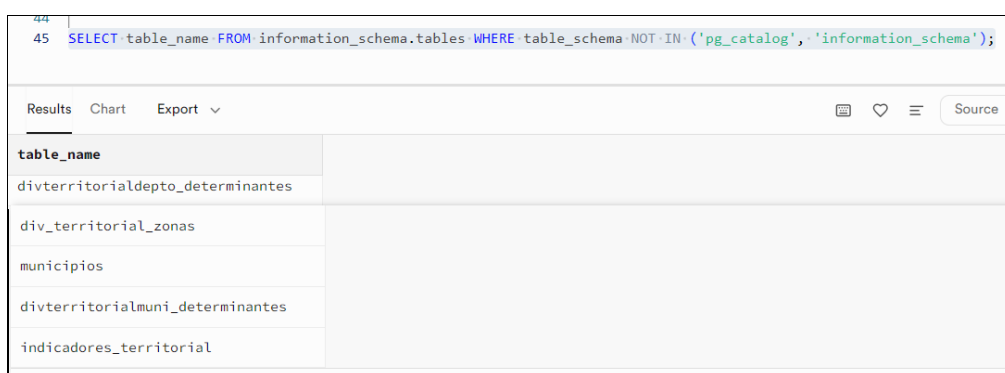
2.3 Verificación y Control de la Información

Una vez cargados los datos, se procede con una verificación para confirmar que la inserción se ha realizado correctamente y que no se ha comprometido la integridad referencial de la base de datos. Esta etapa se lleva a cabo mediante la ejecución de sentencias SQL específicas, diseñadas para validar la consistencia interna del sistema y detectar cualquier posible anomalía.

Estas consultas permiten revisar relaciones entre tablas, integridad de claves foráneas, existencia de duplicados, y cumplimiento de reglas de negocio previamente establecidas. Solo después de superar esta fase, los datos se consideran oficialmente cargados y listos para ser utilizados dentro del aplicativo web.

2.4 Evidencias de Carga a la Base de Datos

A manera de evidencia, se registra una captura de pantalla o un resultado visual de la consulta SQL realizada en la base de datos PostgreSQL, en la que se reflejan los datos recientemente ingresados. Esta evidencia cumple una doble función: por un lado, sirve como respaldo documental del proceso realizado; y por otro, permite una verificación visual directa, útil tanto para propósitos técnicos como de auditoría.

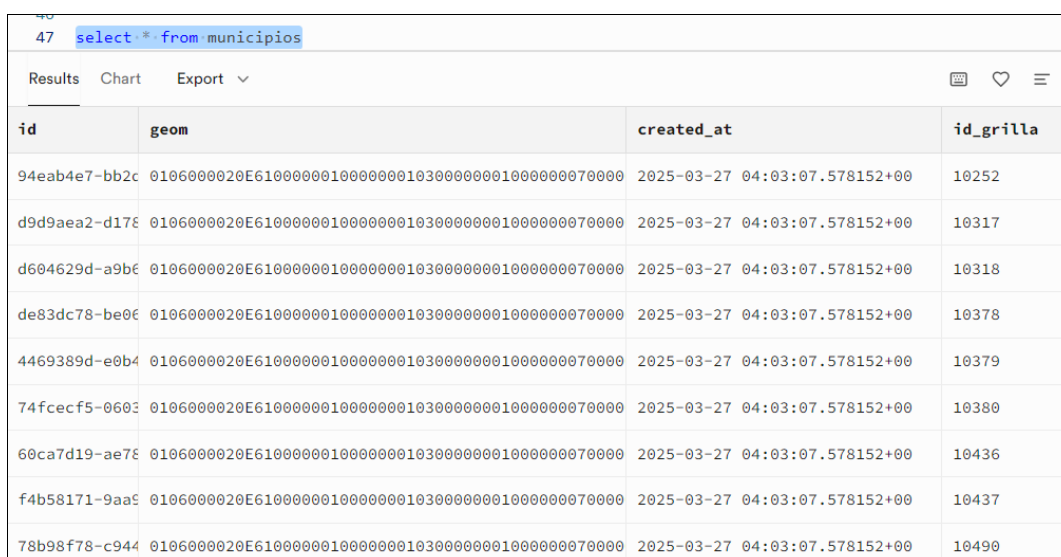


```
44
45 SELECT table_name FROM information_schema.tables WHERE table_schema NOT IN ('pg_catalog', 'information_schema');
```

table_name
divterritorialdepto_determinantes
div_territorial_zonas
municipios
divterritorialmuni_determinantes
indicadores_territorial

Ilustración 1. Evidencias de Carga a la Base de Datos Fuente: ADA.

Esta imagen tiene como propósito mostrar cómo se estructura y almacena la información geográfica de los municipios en la base de datos, útil para análisis territoriales o visualización en sistemas de información geográfica (SIG).



```
46
47 select * from municipios
```

id	geom	created_at	id_grilla
94eab4e7-bb2c	0106000020E610000001000000010300000001000000070000	2025-03-27 04:03:07.578152+00	10252
d9d9aea2-d178	0106000020E610000001000000010300000001000000070000	2025-03-27 04:03:07.578152+00	10317
d604629d-a9b6	0106000020E610000001000000010300000001000000070000	2025-03-27 04:03:07.578152+00	10318
de83dc78-be06	0106000020E610000001000000010300000001000000070000	2025-03-27 04:03:07.578152+00	10378
4469389d-e0b4	0106000020E610000001000000010300000001000000070000	2025-03-27 04:03:07.578152+00	10379
74fcec5-0603	0106000020E610000001000000010300000001000000070000	2025-03-27 04:03:07.578152+00	10380
60ca7d19-ae78	0106000020E610000001000000010300000001000000070000	2025-03-27 04:03:07.578152+00	10436
f4b58171-9aa5	0106000020E610000001000000010300000001000000070000	2025-03-27 04:03:07.578152+00	10437
78b98f78-c944	0106000020E610000001000000010300000001000000070000	2025-03-27 04:03:07.578152+00	10490

Ilustración 2. Evidencias de Carga a la Base de Datos Fuente: ADA.

El objetivo de esta instrucción es el de contar el total de registros (filas) existentes en la tabla denominada municipios. En la sección de resultados (Results) se muestra que la tabla incluye 13.788 registros, es decir, se tienen almacenadas en la base de datos 13.788 localidades o municipios, lo cual podría corresponder tanto con las localidades como con los datos históricos (dependiendo de la fuente de datos utilizada).

46	
47	<code>select count(*) from municipios;</code>
Results	Chart Export ▾
count	
13788	

Ilustración 3. Evidencias de Carga a la Base de Datos Fuente: ADA.

La imagen está mostrando el total de registros (filas) que hay en la tabla div_territorial_zonas. El resultado de la consulta muestra que hay 23,448 registros en la tabla. Este resultado puede ser utilizado para comprobar el número de zonas territoriales que se hallan registradas o bien que la carga de datos ha sido exitosa.

48	<code>select count(*) from div_territorial_zonas;</code>
49	
Results	Chart Export ▾
count	
23448	

Ilustración 4. Evidencias de Carga a la Base de Datos Fuente: ADA.

El objetivo primordial de esta consulta es contar los registros (filas) que hay en la tabla divterritorialdepto_determinantes. El resultado que aparece en la parte inferior menciona que hay 33 registros en la tabla. Es una manera habitual de comprobar qué cantidad de datos hay disponibles en una determinada tabla.

50	<code>select count(*) from divterritorialdepto_determinantes;</code>
51	
Results	Chart Export ▾
count	
33	

Ilustración 5. Evidencias de Carga a la Base de Datos Fuente: ADA.

Mediante la presente consulta, se quiere averiguar cuál es la cantidad total de registros (filas) que hay almacenados en la tabla denominada divterritorialmuni_determinantes.

El resultado de la consulta, que aparece mostrado abajo, en la sección resultados, indica que hay 1121 registros en la tabla correspondiente. Por lo tanto, la imagen tiene la intención de mostrar la cantidad de datos que hay acumulados en dicha tabla.

52	<code>select count(*) from divterritorialmuni_determinantes;</code>
Results	Chart Export ▾
count	
1121	

Ilustración 6. Evidencias de Carga a la Base de Datos Fuente: ADA.

El objetivo de esta consulta es contar el número total de registros (filas) en la tabla indicadores_territorial, cuyo resultado, representa el total de registros de la tabla, el cual se encuentra en la parte inferior de la imagen - el resultado aparece en la columna count con valor 1121, lo que significa que la tabla indicadores_territorial tiene 1121 registros. En definitiva, la imagen ilustra cómo obtener el total de los registros de una tabla con SQL.

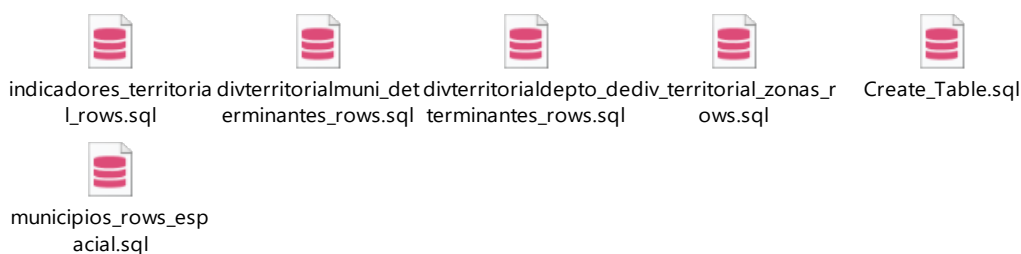
53	<code>select count(*) from indicadores_territorial;</code>
Results	Chart Export ▾
count	
1121	

Ilustración 7. Evidencias de Carga a la Base de Datos Fuente: ADA.

3. Procedimiento para montar la base de datos.

3.1 Prerrequisitos

Antes de iniciar cualquier tipo de procedimiento relacionado con la creación o modificación de la base de datos del aplicativo, es fundamental verificar que se cumplen ciertos prerrequisitos técnicos y operativos. Esta etapa previa resulta esencial, ya que garantiza que el entorno de trabajo esté debidamente preparado para ejecutar las acciones necesarias de forma segura, ordenada y eficiente. Ignorar esta verificación podría derivar en errores durante la implementación, pérdida de tiempo o incluso en inconsistencias dentro del modelo de datos.



3.2 Permisos adecuados

Uno de los primeros aspectos a revisar es que el usuario responsable de ejecutar los scripts cuente con los permisos necesarios dentro del sistema gestor de bases de datos. No se trata solo de una formalidad: sin estos privilegios, muchas de las operaciones críticas no podrían llevarse a cabo correctamente.

En términos más específicos, dicho usuario debe tener la capacidad de crear y modificar estructuras de base de datos, así como insertar registros y ejecutar procedimientos almacenados. Esto implica, entre otras cosas, poder definir nuevos esquemas, diseñar tablas con sus respectivos campos, establecer relaciones entre entidades y gestionar la carga inicial de datos. Estos permisos suelen estar reservados a perfiles administrativos o técnicos que cuentan con un rol claro dentro del equipo de desarrollo o de infraestructura tecnológica del Ministerio de Minas y Energía.

3.3 Esquema predefinido

Una vez se cuenta con los permisos adecuados, el siguiente paso es contar con un esquema de base de datos claramente definido, que responda a las necesidades del proyecto y que pueda ser validado por el DBA (Administrador de Base de Datos) del Ministerio. Este esquema debe estar alineado con los objetivos funcionales del aplicativo, y su diseño debe facilitar tanto la integridad como la escalabilidad de la información que se va a manejar.

En el caso específico del módulo relacionado con los municipios, se plantea una estructura base que puede servir como punto de partida. Este diseño no es rígido, pero sí proporciona una guía clara sobre cómo deben organizarse los datos y cómo deben relacionarse entre sí las distintas entidades involucradas.

3.3.1 Identificación de principales entidades

El núcleo de este esquema lo constituye la entidad Municipios, donde se concentrará la información básica correspondiente a cada municipio del país. Esta tabla funcionará como eje central del modelo y se complementará con otras entidades que permiten enriquecer y estructurar mejor los datos.

En primer lugar, se encuentra la entidad Departamentos, la cual representa las divisiones territoriales mayores que agrupan a los municipios. Esta relación es clave para mantener una organización coherente del territorio y para facilitar consultas o visualizaciones a nivel departamental.

3.3.2 Definición de tablas y campos

A continuación, se detallan las tablas principales y los campos sugeridos para cada una. Esta estructura está pensada para ser fácilmente interpretable, flexible y lo suficientemente robusta como para adaptarse a posibles cambios o ampliaciones futuras.

Tabla de Apoyo 1:

- dpto_ccdgo (PK): Identificador único para cada departamento. Funciona como clave primaria.
- dpto_cnabr: Nombre oficial del departamento.
- mpio_cdpmp: Identificador del municipio.
- mpio_cnabr: Nombre del municipio.
- Id_Grilla: Identificador único del polígono.

Tabla de Apoyo 3:

- dpto_ccdgo (PK): Identificador único para cada departamento. Funciona como clave primaria.
- dpto_cnabr: Nombre oficial del departamento.
- Id_Grilla: Identificador único del polígono.
- mpio_cdpmp: Identificador del municipio.
- Suma de S_112: Corresponde a la presencia o no de determinantes en la zona.

Tabla de Apoyo 4:

- dpto_ccdgo (PK): Clave primaria que identifica de forma única a cada departamento.
- dpto_cnabr: Nombre oficial del departamento.
- Suma de S_1: Corresponde a la presencia o no de determinantes en la zona.

El modelo relacional definido entre estas entidades responde a una lógica coherente y fácilmente escalable:

- Un departamento puede agrupar múltiples municipios, estableciendo así una relación de uno a muchos.

3.3.3 Modelo Entidad-Relación (ER) Detallado para el Aplicativo Web

El modelo relacional definido entre estas entidades responde a una lógica coherente y fácilmente escalable. Este tipo de modelado permite una gestión ordenada de los datos y facilita consultas complejas, visualizaciones jerárquicas y análisis detallados de gran utilidad.

3.3.3.1 Entidades Principales

1. Departamento

○ Atributos:

- dpto_ccdgo (PK): Código único del departamento.
- dpto_cnabr: Nombre del departamento.
- geom: Geometría espacial (MultiPolygon, EPSG:4326).

2. Municipio

○ Atributos:

- mpio_cdpmp (PK): Código único del municipio.
- mpio_cnabr: Nombre del municipio.
- dpto_ccdgo (FK): Código del departamento al que pertenece.
- geom: Geometría espacial (MultiPolygon, EPSG:4326).
- id_grilla: Identificador de la grilla territorial.

3. Zona

○ Atributos:

- id_zona (PK): Identificador único de la zona.
- nombre: Nombre de la zona (Caribe, Andina, Amazonia, etc.).
- descripción: Características de la zona.

4. Determinante

○ Atributos:

- id_determinante (PK): Identificador único del determinante.
- tipo: Categoría (social, ambiental, sectorial, infraestructura).
- descripción: Explicación del determinante.

5. Indicador

○ Atributos:

- id_indicador (PK): Identificador único del indicador.
- nombre: Nombre del indicador (ej. PIB per cápita, cobertura de energía).
- unidad_medida: Unidad asociada (ej. porcentaje, habitantes).
- año: Año de referencia del dato.

6. Sistema de Referencia Espacial

- **Atributos:**
 - srid (PK): Identificador único del sistema de coordenadas.
 - auth_name: Autoridad que define el sistema (ej. EPSG:4326).
 - srtxt: Definición en formato WKT.

3.3.3.2 Relaciones

1. Departamento - Municipio

- **Cardinalidad:** 1:N (Un departamento tiene muchos municipios).
- **Descripción:** Relación de composición territorial.

2. Municipio - Zona

- **Cardinalidad:** M:N (Un municipio puede pertenecer a múltiples zonas y viceversa).
- **Tabla de Unión:** div_territorial_zonas (registra atributos booleanos: Caribe, Andina, etc.).

3. Departamento - Determinante

- **Cardinalidad:** M:N (Un departamento tiene múltiples determinantes y un determinante puede aplicarse a varios departamentos).
- **Tabla de Unión:** divterritorialdepto_determinantes (almacena valores como suma_s_111, suma_a_221, etc.).

4. Municipio - Determinante

- **Cardinalidad:** M:N (Un municipio tiene múltiples determinantes y viceversa).
- **Tabla de Unión:** divterritorialmuni_determinantes (similar a la tabla departamental, pero a nivel municipal).

5. Municipio - Indicador

- **Cardinalidad:** M:N (Un municipio tiene múltiples indicadores y un indicador puede aplicarse a varios municipios).
- **Tabla de Unión:** indicadores_territorial (almacena valores como pobl_rur, ipm, PIBxhabitante, etc.).

6. Sistema de Referencia - Geometrías

- **Cardinalidad:** 1:N (Un sistema de referencia aplica a múltiples geometrías de municipios/departamentos).
- **Descripción:** Las geometrías (geom en municipios y departamentos) usan el srid definido en spatial_ref_sys.

3.3.3.3 Diagrama simplificado

```
Departamento (1) ----< Municipio (N)
Municipio (N) >----< Zona (N)
Departamento (N) >----< Determinante (N)
Municipio (N) >----< Determinante (N)
Municipio (N) >----< Indicador (N)
SistemaReferencia (1) ----< Geometrías (N)
```

Ilustración 8. Modelo Entidad – Relación. Fuente: ADA.

4. Consideraciones técnicas

Al momento de estructurar y gestionar la base de datos del aplicativo, resulta esencial tener en cuenta una serie de buenas prácticas técnicas que garantizan no solo el rendimiento, sino también la integridad y consistencia de la información almacenada. Estas consideraciones, aunque puedan parecer detalles técnicos menores, tienen un impacto directo en la experiencia del usuario final y en la robustez del sistema en el mediano y largo plazo.

4.1 Normalización

Uno de los pilares fundamentales en la construcción de una base de datos eficiente es la normalización. En este caso, se procura alcanzar al menos la Tercera Forma Normal (3NF), lo cual permite eliminar redundancias innecesarias y evitar inconsistencias que puedan surgir al manipular datos duplicados. Esta estructura facilita el mantenimiento de la base de datos y mejora la calidad de la información, ya que cada dato se encuentra en un único

lugar y se relaciona con los demás de manera lógica y ordenada. Además, la normalización contribuye a reducir el espacio de almacenamiento y optimiza las operaciones de actualización, inserción y eliminación, lo cual se traduce en una base de datos más limpia, coherente y confiable.

4.2 Índices

Otro aspecto clave para asegurar un buen desempeño del sistema es la implementación de índices sobre aquellas columnas que son más frecuentemente consultadas por los usuarios. En este caso, se identifican como prioritarios campos como `id_departamento`, `codigo_dane` e `id_municipio`, ya que son comúnmente utilizados para filtrar, ordenar o buscar información. Al crear índices sobre estas columnas, se mejora notablemente la velocidad de respuesta en las consultas, permitiendo que los tiempos de carga se mantengan bajos incluso cuando el volumen de datos crece. Esta optimización no solo beneficia al equipo técnico, sino también a los usuarios que acceden al sistema en busca de datos precisos y oportunos.

4.3 Restricciones de Integridad Referencial

Por último, pero no menos importante, se asegura que existan y se apliquen correctamente las restricciones de integridad referencial entre las distintas tablas de la base de datos. Esto implica que todos los valores presentes en las columnas `id_departamento` e `id_municipio` deben corresponder necesariamente a registros válidos en sus respectivas tablas de referencia. Estas restricciones no solo previenen la aparición de datos huérfanos o relaciones inválidas, sino que también refuerzan la coherencia lógica del sistema, brindando una mayor confianza sobre la veracidad y solidez de los datos almacenados. En definitiva, se trata de una práctica indispensable para garantizar que la estructura relacional de la base de datos se mantenga íntegra y funcional con el paso del tiempo.

5. Validación por parte del DBA

Antes de considerar finalizada la arquitectura de la base de datos, resulta fundamental someterla a una revisión detallada por parte del administrador de bases de datos (DBA) del Ministerio de Minas y Energía. Esta etapa no solo representa un punto de control técnico, sino también un ejercicio de verificación profunda para garantizar que el diseño cumpla con los lineamientos institucionales en materia de seguridad, eficiencia y escalabilidad.

El rol del DBA en este proceso trasciende la simple validación estructural; implica evaluar con criterio técnico y experiencia si la solución propuesta está en condiciones de operar de manera estable y segura bajo los estándares que la entidad exige. Se revisan aspectos clave como la integridad referencial, la configuración de índices, la coherencia en los tipos

de datos y, especialmente, la capacidad del sistema para soportar un crecimiento sostenido en el tiempo sin comprometer el rendimiento.

Además, se explora la posibilidad —cuando sea pertinente— de incorporar procedimientos almacenados que faciliten la gestión de las operaciones más críticas sobre los datos, así como la implementación de reglas de negocio directamente en el nivel de base de datos. Esta decisión se toma considerando tanto la conveniencia técnica como el impacto que tendría en la mantenibilidad del sistema. La intervención del DBA en este punto también permite identificar cuellos de botella potenciales, proponer mejoras orientadas al desempeño, y anticiparse a futuras necesidades operativas que pudieran emerger a medida que el aplicativo web se consolide.

5.1 Compatibilidad de los geodatos

Dado que la aplicación maneja información con componentes espaciales, la base de datos debe estar equipada con una extensión que le permita comprender, almacenar y operar sobre este tipo de datos. En este contexto, se establece como requisito indispensable el soporte para PostGIS, una extensión espacial para PostgreSQL que amplía sus capacidades y lo convierte en una plataforma robusta para el manejo de geodatos.

La solución debe permitir, como mínimo, la gestión eficiente de datos georreferenciados: esto incluye su almacenamiento adecuado, la ejecución de consultas espaciales, y el tratamiento de la información en función de coordenadas geográficas o proyecciones específicas. Todo esto debe realizarse bajo conformidad con los estándares definidos por el sistema de referencia espacial EPSG: 4326, que garantiza precisión y coherencia en el manejo de las ubicaciones geográficas que forman parte del dominio de esta aplicación.

Esta compatibilidad espacial no solo es un requerimiento técnico, sino una pieza clave para que la solución pueda ofrecer una experiencia completa y significativa a los usuarios que dependen de la representación geográfica de los datos para su análisis, monitoreo o toma de decisiones.

6. Ejecución de scripts

Dentro del proceso de despliegue y preparación del entorno de datos, uno de los pasos clave consiste en la correcta ejecución de los scripts SQL provistos. Este procedimiento garantiza que tanto la estructura de la base de datos como los registros iniciales queden establecidos de forma coherente con lo diseñado. A continuación, se describe, paso a paso, cómo realizar esta ejecución utilizando la herramienta PGAdmin, que permite interactuar de forma visual con servidores PostgreSQL.

6.1 Script Create_Table.sql: creación de las estructuras de base de datos

Este script contiene las instrucciones necesarias para crear todas las tablas definidas en el esquema de base de datos. Su ejecución debe realizarse en la instancia que el Ministerio de Minas y Energía (MME) determine como destino.

6.1.1 Iniciar PGAdmin

Se comienza abriendo PGAdmin desde el equipo local. Este entorno gráfico facilita la conexión, exploración y gestión de servidores PostgreSQL. Al iniciarlo, se despliega una interfaz donde se puede ver el árbol de servidores configurados.

6.1.2 Conectarse al servidor deseado

En el panel izquierdo de PGAdmin, se debe localizar el servidor PostgreSQL que se quiere utilizar. Si este ya está configurado, bastará con expandir su entrada. En caso contrario, se puede configurar uno nuevo haciendo clic derecho sobre la sección “Servers” y seleccionando “Create” → “Server”. Aquí se ingresan los datos de conexión necesarios, como el nombre del servidor, la dirección IP o dominio, el puerto y las credenciales de acceso.

6.1.3 Abrir la herramienta de consultas

Una vez conectado al servidor, se debe identificar la base de datos sobre la cual se ejecutará el script. Al seleccionarla, se hace clic derecho sobre su nombre y se elige la opción “Query Tool” o “Herramienta de consultas”. Esto abre una nueva pestaña en el editor, lista para recibir instrucciones SQL.

6.1.4 Cargar el archivo del script

En la barra de herramientas del editor de consultas, se encuentra el ícono para abrir archivos (generalmente representado con una carpeta o un documento). Al hacer clic en él, se despliega el explorador de archivos del sistema. Desde allí, se localiza y selecciona el archivo Create_Table.sql. Una vez cargado, su contenido aparecerá dentro del editor de texto, listo para su ejecución.

6.1.5 Ejecutar el script

Con el script ya visible en el editor, se procede a su ejecución. Esto se logra haciendo clic en el botón de ejecutar, que suele estar representado por un ícono de rayo, o simplemente presionando la tecla F5. PGAdmin interpreta y ejecuta cada línea de código contenida en el archivo.

6.1.6 Verificar que las tablas se hayan creado correctamente

Tras la ejecución, los resultados aparecen en la parte inferior del editor, en una sección dedicada a la salida de consola. Si no se presentan errores, se puede asumir que las tablas se han creado con éxito. Para confirmarlo visualmente, se puede volver al panel izquierdo y expandir la sección “Schemas” → “public” → “Tables”. Allí deben aparecer listadas las nuevas tablas creadas por el script.

6.2 Script `div_territorial_zonas_rows.sql`: inserción de zonas y datos territoriales

Este segundo script tiene como propósito cargar en la base de datos los datos correspondientes a las divisiones territoriales, de acuerdo con la estructura previamente creada. El procedimiento para ejecutarlo sigue una lógica similar a la del script anterior.

6.2.1 Iniciar PGAdmin y conectarse al servidor

Al igual que antes, se inicia PGAdmin y se establece conexión con el servidor PostgreSQL deseado. Esta conexión debe apuntar a la base de datos en la que ya se han creado las tablas mediante el script anterior.

6.2.2 Seleccionar la base de datos adecuada

Desde el panel izquierdo, se expande el servidor conectado y se hace clic sobre la base de datos donde se desea ejecutar el script. Es fundamental asegurarse de estar trabajando sobre la misma instancia que recibió las estructuras.

6.2.3 Abrir una nueva ventana de consulta

Con la base de datos seleccionada, se abre la herramienta de consultas mediante clic derecho sobre su nombre y seleccionando “Query Tool”. Esto habilita una nueva pestaña dentro del editor donde se podrán cargar y ejecutar instrucciones SQL.

6.2.4 Cargar el archivo `div_territorial_zonas_rows.sql`

Desde el editor, se utiliza nuevamente la opción para abrir archivos. Al seleccionar el archivo correspondiente, su contenido se carga automáticamente en el área de edición. Este archivo contiene las sentencias necesarias para poblar las tablas con los datos de zonificación territorial.

6.2.5 Ejecutar el script

Se ejecuta el contenido cargado utilizando el botón de ejecutar o la tecla F5. PGAdmin procesa el archivo y despliega los resultados de la ejecución en la parte inferior de la ventana.

6.2.6 Validar que los datos hayan sido insertados correctamente

Si el script se ha ejecutado correctamente, los datos deben haber sido insertados sin errores. Para verificar esto, se pueden realizar consultas simples desde la misma herramienta como, por ejemplo:

```
sql

SELECT * FROM nombre_de_la_tabla;
```

Ilustración 9. Script div_territorial_zonas_rows.sql: inserción de zonas y datos territoriales. Fuente ADA.

Esto permite comprobar visualmente que los registros se encuentren disponibles dentro de la tabla correspondiente.

Es importante recalcar que antes de ejecutar cualquier script, se debe revisar que los archivos .sql estén correctamente formateados, que no contengan errores sintácticos y que estén orientados a la base de datos correcta. De esta manera, se minimizan riesgos de errores y se garantiza que el sistema funcione conforme a lo esperado.

6.3 Carga y ejecución de scripts determinantes: una guía paso a paso

divterritorialdepto_determinantes_rows.sql: Generación de datos determinantes a nivel departamental

Este procedimiento tiene como objetivo principal la creación de registros clave que forman parte de la estructura territorial del sistema, específicamente a nivel de departamento. Para llevarlo a cabo, utilizamos PGAdmin, una herramienta visual que nos permite interactuar con nuestra base de datos PostgreSQL de manera práctica e intuitiva.

6.3.1 Iniciar PGAdmin y conectarse al servidor

Comienzo abriendo PGAdmin desde mi equipo. Una vez cargada la interfaz, localizo el servidor PostgreSQL al cual necesito acceder. Si el servidor no está conectado, simplemente hago clic derecho sobre él y selecciono la opción de conexión, ingresando mis credenciales si es necesario. Este paso inicial me permite establecer un puente entre mi entorno local y el motor de base de datos.

6.3.2 Seleccionar la base de datos adecuada

Ya dentro de PGAdmin, en el panel izquierdo navego cuidadosamente por la estructura de servidores y bases de datos hasta ubicar aquella en la que deseo trabajar. Al identificarla, hago clic derecho sobre ella y selecciono la opción Query Tool, que me abre una ventana interactiva para ejecutar consultas SQL de forma directa.

6.3.3 Cargar el script SQL desde el archivo correspondiente

Con la ventana de consultas abierta, ahora procedo a cargar el archivo `divterritorialdepto_determinantes_rows.sql`, que contiene el conjunto de instrucciones SQL necesarias para insertar los datos determinantes. Para ello, hago clic en el ícono de carpeta que se encuentra en la barra superior o presiono el atajo de teclado `Ctrl + O`. Esto me permite buscar el archivo en mi sistema local. Una vez lo localizo, lo selecciono y lo abro directamente en la herramienta de consultas.

6.3.4 Ejecutar el script y generar los datos

Con el script cargado y visible en la ventana de edición, doy clic en el botón de Ejecutar, representado por un ícono de rayo, o simplemente presiono `F5`. En ese momento, PGAdmin comienza a interpretar línea por línea el contenido del archivo, ejecutando cada instrucción conforme a la sintaxis SQL definida.

6.3.5 Visualizar y verificar los resultados de la ejecución

Al finalizar la ejecución, los resultados aparecen automáticamente en el panel inferior de la ventana del Query Tool. Aquí es posible visualizar mensajes de éxito, advertencias, errores de sintaxis, o incluso resultados de consultas si el script los incluye. Esta retroalimentación es fundamental para validar que el proceso se ha llevado a cabo correctamente.

Si en algún momento surge un inconveniente durante la ejecución, como un error en el código SQL o algún conflicto con la estructura de la base de datos, reviso detenidamente los mensajes de error que proporciona PGAdmin. Estos mensajes suelen señalar con precisión el tipo de problema y la línea específica donde ocurre, facilitando así la identificación y solución del inconveniente.

6.4 Generación de territorios a nivel municipal

divterritorialmuni_determinantes_rows.sql. A través de este script, se busca establecer los registros que representan los distintos territorios a nivel municipal, los cuales son esenciales para una adecuada organización y segmentación de la información geográfica dentro del sistema.

6.4.1 Inicio de PGAdmin y conexión a la base de datos

Abro PGAdmin desde mi equipo, asegurándome de tener acceso al servidor de base de datos correspondiente. Si el servidor no aparece conectado, realizo la conexión manualmente desde el panel izquierdo, haciendo clic derecho y eligiendo la opción Connect. Esto me habilita para trabajar directamente sobre la base de datos deseada.

6.4.2 Ubicación y selección de la base de datos

Una vez conectado al servidor, localizo la base de datos sobre la cual voy a trabajar. Si aún no está abierta, simplemente la conecto desde el panel lateral. Luego hago clic derecho sobre su nombre y selecciono la opción Query Tool, lo que me permite abrir una nueva pestaña de consulta dentro del entorno de PGAdmin.

6.4.3 Acceder al archivo del script SQL

Con la ventana del Query Tool ya activa, busco el archivo `divterritorialmuni_determinantes_rows.sql` en mi equipo. Para abrirlo, utilizo el botón en forma de carpeta ubicado en la parte superior izquierda de la ventana, que permite explorar el sistema de archivos. Al encontrar el script, lo selecciono y lo abro dentro del editor de consultas.

6.4.4 Ejecutar el script para cargar los territorios

Con el contenido del script visible, presiono el botón de Ejecutar, o utilizo el atajo de teclado F5. Esto activa la ejecución del conjunto de instrucciones que permiten crear, actualizar o insertar información clave relacionada con los territorios municipales.

6.4.5 Verificación de resultados y validación final

Al completar la ejecución, la herramienta muestra en la parte inferior los mensajes que resumen el proceso. Aquí puedo observar si los datos se han cargado exitosamente o si ha ocurrido algún problema. En caso de modificaciones a tablas, inserciones de registros o creación de nuevas entidades, estos cambios quedan reflejados inmediatamente en la base de datos y pueden comprobarse navegando por su estructura o consultando directamente las tablas afectadas.

Cuando aparece algún error, el análisis del mensaje proporcionado por PGAdmin resulta fundamental para identificar si se trata de un problema en la sintaxis, una dependencia faltante o cualquier otro detalle técnico. En función de ese diagnóstico, realizo los ajustes necesarios antes de volver a ejecutar el script.

6.5 Creación de datos de indicadores: `indicadores_territorial_rows.sql`

Para generar los datos necesarios de los indicadores territoriales, comenzamos con el uso de pgAdmin, una de las herramientas más comunes para gestionar bases de datos PostgreSQL. El proceso se desarrolla paso a paso, de forma intuitiva, para asegurar que todo se cargue y ejecute correctamente.

6.5.1 Inicio de pgAdmin

Lo primero es abrir pgAdmin desde tu sistema. Esta herramienta, una vez ejecutada, te brinda un entorno visual cómodo para trabajar con bases de datos. Si no lo tienes instalado, puedes descargarlo desde la página oficial y seguir los pasos de instalación.

6.5.2 Conexión a la base de datos

Una vez abierto pgAdmin, en el panel de navegación ubicado generalmente a la izquierda, aparece el nombre de tu servidor. Si aún no lo has configurado, este es el momento de hacerlo. Al hacer clic sobre el servidor, se despliega la opción de conectarse. Aquí introduces las credenciales necesarias y seleccionas la base de datos sobre la que vas a trabajar.

6.5.3 Apertura del Query Tool

Luego, con la base de datos seleccionada, accedes al Query Tool o herramienta de consulta. Este editor te permite escribir, modificar y ejecutar comandos SQL. Lo abres haciendo clic en el botón con un icono de hoja o directamente desde el menú contextual sobre la base de datos.

6.5.4 Carga del script indicadores_territorial_rows.sql

En esta etapa puedes escribir directamente el contenido del script si lo tienes a la mano, aunque lo más habitual es cargar el archivo desde tu sistema. Para ello, haces clic en el icono de "Abrir archivo", ubicado en la barra superior del Query Tool, y buscas en tu equipo el archivo llamado indicadores_territorial_rows.sql. Una vez seleccionado, el contenido del script aparece automáticamente en el editor.

6.5.5 Ejecución del script

Con el script ya visible en el editor, lo siguiente es ejecutarlo. Esto se hace fácilmente haciendo clic en el botón de "Ejecutar", identificado con un ícono en forma de rayo. También puedes usar el atajo de teclado correspondiente. Si el script contiene múltiples sentencias SQL, se ejecutan en el orden en que aparecen, asegurando la secuencia lógica del proceso.

6.5.6 Verificación de resultados

Al terminar la ejecución, puedes revisar los resultados en la parte inferior del Query Tool. Aquí encuentras dos pestañas: Data Output, que muestra los datos resultantes y mensajes, donde se notifican errores o advertencias. En caso de que algo no funcione como esperabas, esta pestaña te brinda pistas claras para corregir y volver a ejecutar.

6.6 Carga de información geoespacial: municipios_rows_espacial.sql

Este segundo script se encarga de cargar información geográfica asociada a los municipios. Se ejecuta en un entorno diferente, ya que trabaja sobre una base de datos Oracle. Puedes utilizar herramientas como Oracle SQL Developer o PL/SQL Developer, según prefieras o tengas disponible en tu entorno de trabajo.

6.6.1 Apertura de la herramienta de administración

Inicias tu jornada abriendo la herramienta de tu elección. SQL Developer es muy utilizado por su entorno gráfico amigable y su capacidad para manejar grandes volúmenes de datos. Si prefieres PL/SQL Developer, también es completamente válido y funcional para este proceso.

6.6.2 Conexión a la base de datos Oracle

En la interfaz de la herramienta, accedes al panel de conexiones. Aquí introduces tus credenciales: nombre de usuario, contraseña y el identificador del servicio o SID. Al conectarte exitosamente, puedes ver los esquemas y objetos disponibles en la base de datos.

6.6.3 Carga del script municipios_rows_espacial.sql

Luego, creas un nuevo archivo SQL desde el menú o abres uno existente. En SQL Developer, simplemente vas a "Archivo > Abrir" y seleccionas el archivo municipios_rows_espacial.sql. También puedes copiar y pegar directamente el contenido del script dentro del editor SQL.

6.6.4 Ejecución del script

Para ejecutar el script, haces clic en el ícono de rayo o utilizas los atajos del teclado: Ctrl + Enter en SQL Developer o F8 en PL/SQL Developer. Si el script está correctamente estructurado, comienza a insertarse la información geoespacial en la tabla correspondiente.

6.6.5 Revisión de resultados

Al concluir la ejecución, los resultados aparecen en la sección inferior del editor. Si el script incluye consultas SELECT, verás los resultados en forma de tabla. En caso de errores, estos se reflejan claramente en la pestaña de errores, permitiéndote identificar y corregir cualquier problema con rapidez.

6.7 Consideraciones finales

Antes de ejecutar cualquier script que modifique la estructura o los datos de una base de datos (ya sea mediante INSERT, UPDATE o DELETE), es fundamental verificar que tu usuario tenga los permisos necesarios. También es importante que el script esté correctamente dividido en bloques si incluye múltiples instrucciones. Esto garantiza que la ejecución no se interrumpa y que cada sentencia se interprete de manera adecuada.

Una vez ejecutados ambos scripts, las tablas correspondientes deben quedar correctamente pobladas con los datos requeridos, tanto para los indicadores territoriales como para la información geoespacial de los municipios. Así, el sistema queda preparado para su posterior consulta, análisis o visualización.

7. Instalación de Docker en Linux/Ubuntu

En este apartado, se describen los pasos necesarios para instalar Docker en un sistema operativo Linux, específicamente en una distribución basada en Ubuntu. Docker es una herramienta muy utilizada para la creación, implementación y ejecución de aplicaciones en contenedores, lo que permite una gestión eficiente y segura de las aplicaciones. Para lograr una instalación limpia y exitosa, es importante seguir ciertos procedimientos previos que garantizarán que no haya conflictos con versiones anteriores de Docker que puedan estar instaladas en el sistema.

7.1. Limpieza de versiones anteriores de Docker

Antes de proceder con la instalación de Docker, es fundamental asegurarse de que cualquier versión anterior de Docker o sus componentes relacionados haya sido completamente eliminada. Esto evita posibles conflictos con versiones obsoletas que podrían afectar el rendimiento o el funcionamiento del sistema. Para llevar a cabo este proceso de manera efectiva, se puede ejecutar un comando específico en la terminal que eliminará las versiones anteriores de Docker y sus paquetes asociados.

El siguiente comando es útil para limpiar los paquetes más comunes que podrían estar instalados en el sistema:

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc;  
do  
    sudo apt-get remove $pkg  
done
```

Ilustración 10. Limpieza de versiones anteriores de Docker. Fuente ADA.

Este comando recorre una lista de paquetes relacionados con Docker, como docker.io, docker-doc, docker-compose, entre otros, y los elimina uno por uno utilizando el gestor de paquetes apt-get. Es importante destacar que este paso es crucial, ya que garantizar que no queden restos de instalaciones anteriores ayudará a evitar posibles errores al intentar instalar una versión más actualizada de Docker.

Al ejecutar este comando, se elimina no solo el software principal, sino también cualquier complemento o herramienta asociada que pudiera haber quedado instalada en el sistema, como es el caso de Docker Compose, que se utiliza para definir y ejecutar aplicaciones Docker multi-contenedor.

Este procedimiento de limpieza es recomendable especialmente si se ha realizado una instalación de Docker anteriormente sin haber seguido una guía adecuada, o si se ha tenido alguna versión preinstalada a través de otros métodos. Si se encuentra que el sistema ya está limpio de estas versiones, se puede proceder con la instalación de la última versión disponible sin ningún problema.

7.2. Actualización de repositorios de Docker en la máquina

Para asegurarnos de que nuestra máquina está lista para trabajar con Docker, primero es necesario actualizar los repositorios e instalar las dependencias requeridas. Esto nos garantiza que tenemos las versiones más recientes de los paquetes y que los entornos de desarrollo y producción estén completamente actualizados. Comenzamos con la actualización de los repositorios mediante el siguiente comando:

Este paso descarga la lista de los paquetes más recientes disponibles en los repositorios de la distribución de Ubuntu que estamos utilizando, asegurando que cualquier software adicional que instalemos esté en su versión más reciente.

Una vez que se completa esta actualización, es necesario instalar algunas dependencias esenciales para que Docker funcione correctamente en el sistema. En este caso, las dependencias que debemos instalar son ca-certificates y curl, herramientas que nos ayudarán a manejar conexiones seguras y realizar transferencias de datos a través de HTTP:

```
sudo apt-get update
sudo apt-get install -y ca-certificates curl
```

Ilustración 11. Actualización de repositorios de Docker en la máquina. Fuente ADA.

A continuación, es fundamental agregar la clave GPG oficial de Docker. Esta clave garantiza que los paquetes que descargamos e instalamos desde los repositorios de Docker sean auténticos y no hayan sido alterados. Para ello, creamos un directorio específico donde almacenaremos las claves de seguridad y descargamos la clave pública de Docker directamente desde su servidor:

1. bash
2. CopiarEditar
3. sudo install -m 0755 -d /etc/apt/keyrings
4. sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

Este paso es crucial para mantener la integridad y la seguridad de las instalaciones. La opción curl -fsSL descarga la clave de manera segura y confiable.

Una vez que tenemos la clave GPG, le damos permisos de lectura a todos los usuarios en el archivo de claves para que el sistema pueda acceder a ella de manera adecuada:

1. bash
2. CopiarEditar
3. sudo chmod a+r /etc/apt/keyrings/docker.asc

```
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Ilustración 12. Actualización de repositorios de Docker en la máquina. Fuente ADA.

El siguiente paso es añadir el repositorio de Docker a las fuentes de Apt. Este repositorio es el que nos permitirá acceder a las versiones más recientes de Docker y sus herramientas relacionadas. Usamos el siguiente comando para agregar la fuente de Docker al archivo de fuentes de Apt, asegurándonos de que se utilice la clave GPG previamente descargada para verificar la autenticidad de los paquetes:

```
1. bash
2. CopiarEditar
3. echo \
4. "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \ $(. /etc/os-release && echo
"${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
5. sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Ilustración 13. Guardar en repositorio. Fuente: ADA

En este comando, usamos `dpkg --print-architecture` para obtener la arquitectura del sistema, lo que asegura que estamos agregando el repositorio adecuado para nuestra configuración específica. Además, se emplea el comando `$(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}")` para obtener el nombre de la versión de Ubuntu que estamos utilizando, lo que garantiza que el repositorio se adapte a la versión correcta de nuestro sistema operativo.

Finalmente, después de agregar el repositorio, actualizamos nuevamente los repositorios de Apt para que reconozcan la nueva fuente y podamos instalar Docker sin problemas:

1. bash
2. CopiarEditar
3. sudo apt-get update

Este paso final asegura que todo está listo para que podamos proceder con la instalación de Docker o cualquier otro paquete relacionado. Ahora, nuestra máquina está configurada adecuadamente para trabajar con Docker, con repositorios actualizados y seguridad garantizada por la clave GPG oficial.

```
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://down
load.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Ilustración 14. Actualización de repositorios de Docker en la máquina. Fuente ADA.

7.3. Instalación de Docker

Para proceder con la instalación de Docker en su sistema, primero debe asegurarse de que su equipo cuenta con una versión compatible de Linux y de que se cumplen los requisitos previos para ejecutar Docker. A continuación, se describe el proceso de instalación paso a paso, el cual debe llevarse a cabo desde la terminal del sistema operativo.

El primer paso consiste en actualizar la lista de paquetes de su sistema para asegurarse de que cuenta con las últimas versiones disponibles de los paquetes necesarios. Para ello, ejecute el siguiente comando en la terminal:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Ilustración 15 Instalación de Docker. Fuente ADA.

Este comando actualizará la base de datos de paquetes de su sistema. Posteriormente, para instalar Docker, ejecute el siguiente comando que incluye todos los componentes esenciales necesarios para el funcionamiento del software, tales como Docker CE (Community Edition), el cliente de Docker, el motor de contenedores containerd, así como los complementos adicionales de Docker Compose y Buildx para facilitar la creación y gestión de contenedores:

Es importante destacar que este comando instalará tanto Docker como sus herramientas asociadas, permitiéndole gestionar contenedores, crear aplicaciones distribuidas y realizar diversas tareas relacionadas con la virtualización mediante contenedores. La instalación puede tardar algunos minutos dependiendo de su conexión a internet y las actualizaciones que necesite realizar el sistema.

Una vez finalizada la instalación, podrá verificar que Docker se ha instalado correctamente ejecutando el siguiente comando:

```
bash

sudo docker --version
```

Ilustración 16. Instalación de Docker. Fuente ADA.

Este comando le proporcionará la versión de Docker instalada, confirmando que el proceso se ha llevado a cabo exitosamente.

Con estos pasos, su sistema ya debería contar con Docker instalado y listo para su uso, permitiéndole comenzar a ejecutar y gestionar contenedores de manera eficiente.

7.4 Configuración de permisos para el usuario Docker

Cuando se trabaja con Docker desde la terminal, es común encontrarse con la necesidad de anteponer sudo a cada comando para que el sistema permita su ejecución. Si bien esto garantiza un nivel de seguridad adicional, también puede entorpecer la fluidez del trabajo diario, especialmente en entornos de desarrollo donde se requiere ejecutar múltiples comandos de forma constante.

Para facilitar el uso de Docker y evitar tener que escribir sudo cada vez, es posible asignar al usuario actual permisos específicos mediante su inclusión en el grupo del sistema denominado docker. Este grupo otorga los privilegios necesarios para interactuar con el demonio de Docker directamente, sin necesidad de elevar los privilegios de forma manual.

El procedimiento es bastante sencillo. Solo es necesario ejecutar el siguiente comando en la terminal:

```
bash

sudo usermod -aG docker $USER
```

Ilustración 17. Configuración de permisos para el usuario Docker. Fuente ADA.

Con esta instrucción, se añade el usuario actual (representado por \$USER) al grupo docker, manteniendo intactas las demás pertenencias a grupos del sistema gracias al modificador -aG.

Para que este cambio surta efecto de inmediato y sin necesidad de reiniciar el sistema, se recomienda ejecutar a continuación el comando:

```
bash

newgrp docker
```

Ilustración 18. Configuración de permisos para el usuario Docker. Fuente ADA.

Este comando inicia una nueva sesión de grupo en la terminal activa, permitiendo que el sistema reconozca de inmediato la nueva asociación del usuario al grupo docker.

Como recomendación adicional —y para asegurar que los cambios se apliquen correctamente en todas las terminales abiertas y futuras sesiones— es conveniente cerrar la terminal actual y volver a abrirla. De este modo, el entorno reconoce por completo los nuevos permisos asignados al usuario, y se puede comenzar a trabajar con Docker sin necesidad de utilizar sudo.

Esta configuración no solo agiliza el flujo de trabajo, sino que también mejora la experiencia general del desarrollo al eliminar fricciones innecesarias durante la ejecución de tareas frecuentes relacionadas con la gestión de contenedores.

8. Despliegue de la Aplicación

8.1. Creación de la Imagen Docker

El proceso de despliegue de la aplicación comienza con la creación de una imagen Docker, que actúa como una especie de contenedor autosuficiente en el que se encapsulan todos los componentes necesarios para ejecutar el sistema de forma consistente en cualquier entorno. Esta imagen incluye el código fuente de la aplicación, sus dependencias, configuraciones específicas y todo lo necesario para su ejecución sin depender del sistema operativo anfitrión.

Como primer paso, se accede al directorio donde se encuentra el código fuente del aplicativo web en el entorno local. Para ello, se abre una terminal y se navega hasta la ruta correspondiente mediante el siguiente comando:

```
bash

cd ruta/de_app/en_local/
```

Ilustración 19. Creación de la Imagen Docker. Fuente ADA.

Este directorio debe contener no solo los archivos de la aplicación como tal, sino también el archivo Dockerfile, que define las instrucciones necesarias para construir la imagen. Es fundamental asegurarse de que este archivo esté correctamente configurado, ya que servirá como base para el proceso de construcción.

Una vez ubicado en la carpeta correcta, se puede continuar con la construcción de la imagen, lo cual se aborda en el siguiente apartado. Esta etapa es crucial porque garantiza que la aplicación pueda ser empaquetada de manera uniforme y replicable, independientemente del entorno donde se despliegue posteriormente (servidores locales, entornos de pruebas o producción, o servicios en la nube).

8.2 Código Fuente del Desarrollo

El Código Fuente del desarrollo correspondiente al proyecto, será entregado como un documento adjunto para su revisión y análisis en formato txt. Dicho documento estará guardado en el servidor en la siguiente ruta:

/home/admgear/Documents/Instaladores/Codigo fuente/codigo fuente.txt

8.3 Construcción de la imagen Docker

Una vez que se tiene configurado el entorno de desarrollo y se cuenta con el archivo Dockerfile debidamente definido, se procede a la construcción de la imagen de Docker que encapsula la aplicación. Este proceso es esencial para garantizar que la aplicación pueda ejecutarse de manera consistente, tanto en ambientes locales como en entornos de producción o despliegue en la nube.

Para llevar a cabo esta tarea, se utiliza el comando `docker build`, especificando el nombre del archivo Dockerfile que contiene las instrucciones necesarias para crear la imagen, así como la etiqueta (tag) que identifica la versión y el repositorio de destino de la imagen resultante.

La ejecución del siguiente comando en la terminal inicia el proceso de construcción:

```
bash

docker build -f Dockerfile -t usuariodocker/municipios-viewer:aws .
```

Ilustración 20. Construcción de la imagen Docker. Fuente ADA.

Este comando le indica a Docker que utilice el archivo Dockerfile ubicado en el directorio actual (representado por el punto al final del comando) para generar una imagen. Al mismo tiempo, asigna a esta imagen una etiqueta personalizada —en este caso, `usuariodocker/municipios-viewer:aws`— que facilita su identificación posterior, ya sea para subirla a un repositorio remoto como Docker Hub o para utilizarla directamente en una instancia de AWS.

Durante el proceso de construcción, Docker interpreta las instrucciones contenidas en el Dockerfile en orden secuencial. Estas instrucciones definen desde la imagen base que se utiliza, hasta los pasos para instalar dependencias, copiar archivos del proyecto, configurar variables de entorno y establecer el punto de entrada de la aplicación. Al finalizar este proceso, se obtiene una imagen lista para ser ejecutada, replicada o desplegada en diversos entornos con total fidelidad respecto al entorno original de desarrollo.

Este paso, aunque técnico, representa una etapa fundamental dentro del ciclo de vida de despliegue de la aplicación, ya que permite empaquetar todos los componentes necesarios para su correcto funcionamiento en una unidad portable, reproducible y fácilmente escalable.

8.4 Subir la imagen al repositorio de Docker

Una vez construida correctamente la imagen del contenedor, el siguiente paso consiste en subirla al repositorio correspondiente en Docker Hub. Esta acción permite que la imagen esté disponible desde cualquier entorno autorizado que necesite desplegar el aplicativo, ya

sea en un servidor de pruebas, en un entorno de producción o dentro de un flujo de integración continua.

Para llevar a cabo esta publicación, se utiliza el comando `docker push`, seguido del nombre del usuario y del identificador completo de la imagen que se desea transferir. En este caso, el comando específico es:

```
bash

docker push usuariodocker/municipios-viewer:aws
```

Ilustración 21. Subir la imagen al repositorio de Docker. Fuente ADA.

Este comando inicia el proceso de subida de la imagen etiquetada como `municipios-viewer` con la versión o etiqueta `aws` al repositorio del usuario indicado en Docker Hub. Es importante asegurarse de haber iniciado sesión previamente en Docker (`docker login`) con las credenciales correspondientes, ya que de lo contrario el sistema rechazará la solicitud de subida.

Durante este proceso, Docker verifica si la imagen ya existe parcialmente en el repositorio. Si detecta capas repetidas que no han cambiado desde la última versión publicada, las omite para optimizar la transferencia, reduciendo así tanto el tiempo de carga como el uso del ancho de banda. Una vez finalizada la operación, la imagen queda disponible públicamente (o de forma privada, según la configuración del repositorio) para ser utilizada en despliegues posteriores.

Este paso es fundamental dentro del flujo de trabajo, ya que garantiza que la versión más reciente del aplicativo web esté accesible para su posterior implementación en los entornos definidos por el equipo de desarrollo o el área encargada de infraestructura.

8.5 Creación del archivo `docker-compose.yml`

Como parte del proceso de configuración del entorno de ejecución del aplicativo, uno de los pasos fundamentales consiste en la creación del archivo `docker-compose.yml`. Este archivo cumple un rol clave, ya que permite definir y orquestar de forma clara y estructurada los distintos servicios que componen el ecosistema de la aplicación. A través de este archivo, se especifican contenedores, redes, volúmenes y dependencias, lo que facilita no solo el despliegue, sino también la reproducción del entorno en diferentes etapas del desarrollo y mantenimiento.

El primer paso para iniciar esta configuración es acceder al directorio principal donde reside el código fuente de la aplicación, comúnmente ubicado en la ruta `/app`. Para ello, se abre una terminal con permisos adecuados y se ejecuta el siguiente comando:

```
bash  
  
cd /app
```

Ilustración 22. Creación del archivo docker-compose.yml. Fuente ADA.

Este sencillo comando permite ingresar al directorio raíz del proyecto, donde se alojarán tanto el archivo docker-compose.yml como otros archivos esenciales para la puesta en marcha del entorno. Es importante asegurarse de que esta ruta coincida con la estructura del proyecto local, ya que cualquier discrepancia puede generar errores al momento de levantar los servicios definidos.

Una vez dentro de este directorio, se procede a crear el archivo docker-compose.yml, que se convierte en el punto de partida para levantar todo el stack de servicios con una sola instrucción. La correcta estructuración y redacción de este archivo resulta vital para garantizar que cada servicio se comuniquen entre sí, que las rutas de red sean coherentes y que los volúmenes se gestionen de forma adecuada, asegurando así la persistencia de los datos y la estabilidad del entorno.

8.6 Cree y edite el archivo Docker -compose.yml

Para iniciar la configuración del entorno con Docker, el primer paso consiste en crear un archivo que nos permita definir y orquestar los distintos servicios necesarios para la aplicación. Este archivo se llama docker-compose.yml y actúa como una guía que Docker sigue para levantar los contenedores de forma conjunta y coordinada.

Primero, se crea un directorio destinado a alojar dicho archivo. Esto se hace ejecutando el comando `mkdir docker-compose.yml`, lo cual genera una carpeta con ese nombre dentro del proyecto. Este paso, aunque sencillo, resulta clave para mantener una estructura ordenada en el entorno de trabajo, sobre todo cuando se trabaja con múltiples servicios o configuraciones adicionales.

Una vez creado el directorio, se procede a generar el archivo en sí. Para ello, se utiliza el editor de texto nano, escribiendo en consola el comando `nano docker-compose.yml`.

```
nano docker-compose.yml
```

Ilustración 23. Cree y edite el archivo Docker -compose.yml: Fuente ADA.

Esta instrucción abre el editor en la terminal y permite comenzar a redactar el contenido del archivo directamente, de forma inmediata. Desde allí, se definen los distintos servicios que compondrán el sistema: contenedores, redes, volúmenes, variables de entorno, entre otros elementos que permiten que todo funcione de manera integrada.

Este archivo se convierte en una pieza central del entorno de desarrollo, ya que facilita la automatización del despliegue de todos los componentes necesarios, simplificando tareas que, de otro modo, requerirían configuraciones manuales más complejas.

8.7 Copie el contenido del archivo `docker-compose.yml` desde su entorno local al servidor.

Para dar continuidad al proceso de despliegue del aplicativo, se realiza la transferencia del archivo `docker-compose.yml` desde el entorno local de desarrollo hasta el servidor en el que se ejecutará la aplicación. Este paso es clave, ya que este archivo contiene toda la configuración necesaria para orquestar los distintos servicios que componen el sistema, incluyendo sus respectivas imágenes, volúmenes, variables de entorno y redes. La copia se lleva a cabo utilizando herramientas de transferencia seguras, como `scp` o `rsync`, lo cual garantiza que el contenido del archivo llegue íntegro y sin alteraciones al entorno de destino. Este procedimiento asegura que la infraestructura definida localmente pueda reproducirse de manera fiel en el servidor, facilitando así un despliegue coherente, predecible y alineado con lo que se ha probado previamente en desarrollo.

8.8 Ejecute el despliegue

Una vez finalizada la configuración del entorno y verificado que todos los archivos necesarios se encuentran en su lugar, procedo a ejecutar el despliegue de la aplicación utilizando Docker. Para ello, ingreso al directorio raíz del proyecto desde la línea de comandos y lanzo el siguiente comando:

```
bash
```

```
docker compose up -d
```

Ilustración 24. Ejecute el despliegue. Fuente ADA.

Este paso pone en marcha los contenedores definidos en el archivo `docker-compose.yml` de forma desacoplada, es decir, en segundo plano, permitiendo que cada servicio se inicie automáticamente con la configuración previamente establecida. Al ejecutar esta instrucción, Docker se encarga de construir las imágenes —si aún no existen—, crear los contenedores correspondientes, establecer las redes necesarias y asegurar que cada componente del sistema funcione de manera orquestada y sin interferencias.

La opción `-d`, que significa *detached mode*, resulta especialmente útil porque libera la terminal inmediatamente después del arranque, lo que me permite continuar con otras tareas sin interrumpir la ejecución de los servicios. Una vez finalizado este paso, puedo verificar el estado de los contenedores ejecutando `docker compose ps` y asegurarme de que cada uno se encuentra corriendo correctamente.

Este procedimiento automatizado no solo facilita la puesta en marcha del sistema, sino que también garantiza la consistencia del entorno, ya sea en un entorno local de desarrollo o durante una etapa de pruebas o producción.

8.9 Configuración y Ejecución de la Aplicación

En el proceso de despliegue y operación cotidiana del aplicativo, puede surgir la necesidad de reiniciar el entorno completo por motivos como actualizaciones, ajustes en la configuración, o simplemente para garantizar un funcionamiento limpio y estable después de una modificación significativa en el código fuente o los servicios asociados. Este procedimiento es simple y se lleva a cabo utilizando Docker Compose, una herramienta ampliamente adoptada para la orquestación de contenedores.

Para llevar a cabo un reinicio completo de la aplicación, se siguen dos pasos fundamentales desde la línea de comandos. En primer lugar, se detiene el entorno de ejecución actual con el comando:

```
nginx  
  
docker compose down
```

Ilustración 25. Configuración y Ejecución de la Aplicación. Fuente ADA.

Esta instrucción detiene y elimina todos los contenedores, redes y volúmenes definidos en el archivo docker-compose.yml, sin afectar los datos persistentes fuera de los volúmenes definidos explícitamente. Al hacer esto, se garantiza que no queden procesos residuales que puedan interferir con el nuevo ciclo de ejecución.

A continuación, para levantar nuevamente la aplicación en segundo plano y permitir que todos los servicios se inicien automáticamente, se ejecuta:

```
nginx  
  
docker compose up -d
```

Ilustración 26. Configuración y Ejecución de la Aplicación. Fuente ADA.

El parámetro -d (detached mode) permite que los contenedores se ejecuten en segundo plano, liberando así la terminal para otras tareas y asegurando que el entorno quede completamente operativo sin necesidad de mantener una consola activa.

Este proceso asegura que el aplicativo web se configure y ejecute correctamente, manteniendo la consistencia del entorno y facilitando una administración más predecible durante su ciclo de vida operativo.

9. Verificación

Una vez completado el proceso de despliegue del aplicativo, es fundamental realizar una verificación que asegure que todos los componentes involucrados se han instalado y configurado correctamente. Este paso no solo permite identificar posibles fallos o inconsistencias en el entorno, sino que garantiza que el sistema queda operativo, accesible y funcional para los usuarios finales.

El despliegue debe reflejar una estructura ordenada y coherente, en la cual cada uno de los servicios y módulos del aplicativo web se encuentre ubicado en su ruta correspondiente, accesible desde el entorno definido (ya sea de pruebas o de producción), y respondiendo adecuadamente a las peticiones que se generen desde la interfaz.

Durante esta etapa, se comprueba que todos los endpoints estén activos y entregando respuestas válidas; que las conexiones con bases de datos, servicios externos o APIs funcionen sin interrupciones; y que la configuración del servidor coincida con los parámetros establecidos previamente. Esto incluye verificar la correcta carga de los archivos estáticos, la disponibilidad del frontend, y el comportamiento general del backend.

Es recomendable documentar este proceso de verificación paso a paso, dejando evidencia de cada comprobación realizada. De este modo, se construye un registro que facilita futuras auditorías, mejora la trazabilidad del sistema y permite replicar el procedimiento con mayor precisión en futuros despliegues.

En resumen, esta etapa constituye una garantía de calidad antes de habilitar el sistema para su uso público, asegurando que todo el entorno cumple con los requisitos técnicos, funcionales y operativos previstos desde el inicio del proyecto.

9.1 Determinantes

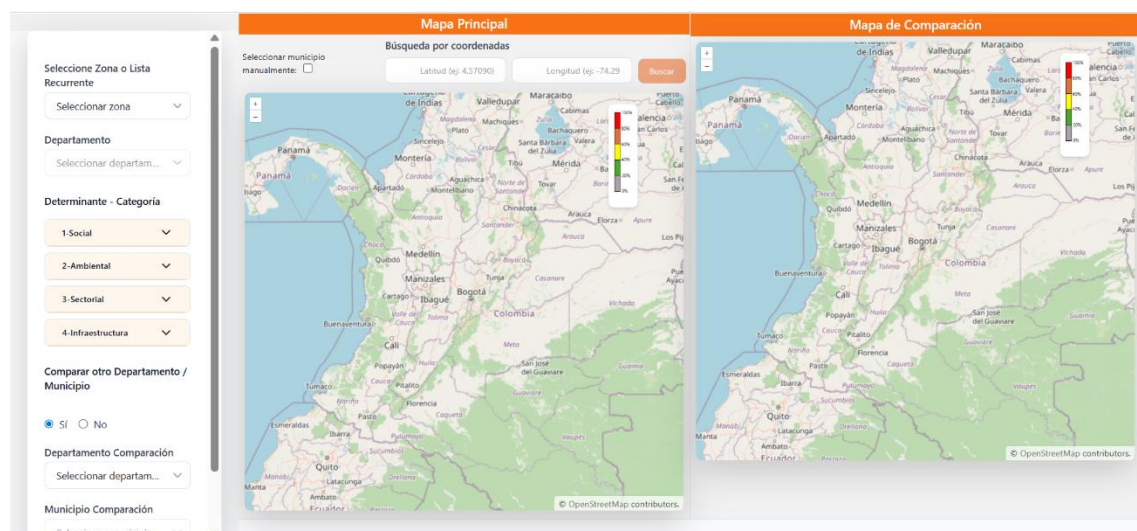


Ilustración 27. disponibilidad del frontend. Fuente ADA.

9.2 Indicadores

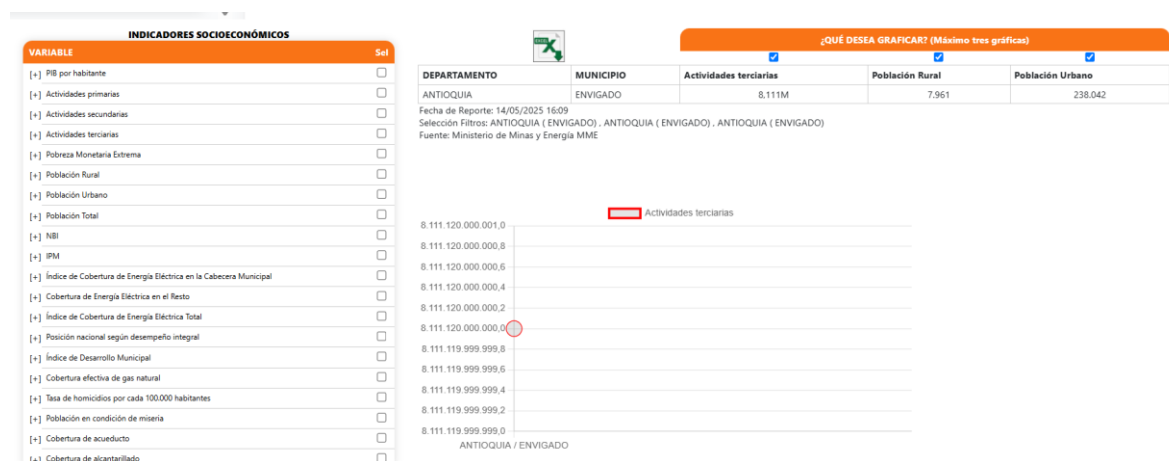


Ilustración 28. disponibilidad del frontend. Fuente ADA.

10. Solución de problemas

Cuando se trabaja con Docker, es posible encontrarse con ciertos inconvenientes que, aunque comunes, pueden resultar confusos si no se conocen los pasos adecuados para solucionarlos. Uno de los problemas más frecuentes está relacionado con los permisos, particularmente cuando Docker no se ejecuta correctamente sin utilizar el comando sudo. Este error es habitual si el usuario no forma parte del grupo de Docker o si hay algún problema de configuración relacionado con los permisos del sistema.

Para resolver esta situación, lo primero que debes intentar es cerrar la sesión y volver a iniciarla. Al hacerlo, el sistema puede actualizar los permisos y permitirte ejecutar Docker sin la necesidad de utilizar sudo cada vez que lo invocas.

Sin embargo, si este paso no soluciona el problema, hay una alternativa sencilla que puedes probar directamente desde la terminal. Ejecutando el siguiente comando:

```
bash

newgrp docker
```

Ilustración 29. Solución de problemas. Fuente ADA.

Este comando actualiza el grupo de trabajo del usuario, de manera que, sin necesidad de reiniciar el sistema, puedes incorporar los cambios de grupo que habilitan los permisos necesarios para que Docker funcione correctamente. Al ejecutar newgrp docker, el sistema aplicará los cambios de inmediato y podrás continuar utilizando Docker con los permisos adecuados.

Es importante destacar que este tipo de problemas no son exclusivos de Docker, sino que pueden ocurrir con otros programas que requieren permisos especiales para su funcionamiento. Por lo tanto, entender cómo administrar los grupos de usuarios y permisos en tu sistema operativo es fundamental para evitar que este tipo de inconvenientes se repitan en el futuro.

Si después de seguir estos pasos sigues teniendo problemas, sería conveniente revisar la configuración del sistema o buscar ayuda en los foros de Docker o en la documentación oficial para obtener una solución más específica.

10.1 Puertos bloqueados

Para verificar si un puerto está bloqueado en el sistema, se puede utilizar el comando `netstat`. Este comando permite inspeccionar las conexiones de red en el dispositivo y analizar el estado de los puertos que están en uso. Para identificar si un puerto específico se encuentra bloqueado o no está siendo utilizado correctamente, se debe emplear la siguiente sintaxis en la terminal del sistema:

```
bash

sudo netstat -tulnp | grep <puerto>
```

Ilustración 30. Solución de problemas. Fuente ADA.

Este comando realiza una serie de acciones en la terminal que nos permiten obtener una visión detallada de la situación de los puertos. Desglosando la sintaxis del comando:

1. **sudo**: Ejecuta el comando con privilegios de superusuario, lo que es necesario para acceder a información detallada sobre las conexiones de red y los puertos utilizados en el sistema.
2. **netstat**: Es una herramienta que se utiliza para obtener estadísticas de red, mostrando las conexiones de red activas, los puertos abiertos y los servicios que están asociados con esos puertos.
3. **-tulnp**: Son varias opciones que se combinan en el comando:
 - **-t**: Muestra las conexiones TCP activas.
 - **-u**: Muestra las conexiones UDP activas.
 - **-l**: Filtra y muestra solo los puertos que están en escucha (es decir, los puertos que están esperando conexiones).
 - **-n**: Muestra las direcciones y números de puerto en formato numérico, evitando la resolución de nombres de host.
 - **-p**: Muestra el identificador del proceso (PID) que está utilizando cada puerto.

4. **grep <puerto>**: Es un comando adicional que filtra los resultados de netstat para buscar únicamente el puerto que se desea verificar. Al sustituir <puerto> por el número de puerto específico que se está investigando, el comando devolverá solo las líneas que contienen ese número, permitiendo una revisión rápida del estado de dicho puerto.

Al ejecutar este comando, se obtiene una lista de las conexiones de red activas asociadas con el puerto indicado, junto con los procesos que están utilizando dichos puertos. Si no aparece ningún resultado relacionado con el puerto en cuestión, esto puede ser una señal de que el puerto está bloqueado o no está en uso. En ese caso, se recomienda revisar la configuración de los firewalls o las políticas de red que puedan estar impidiendo el acceso a dicho puerto. Además, verificar los registros del sistema puede ayudar a identificar posibles errores o conflictos relacionados con el puerto bloqueado.

Este procedimiento es fundamental para diagnosticar problemas de conectividad o de configuración en servidores y dispositivos que dependen de puertos específicos para la comunicación de servicios o aplicaciones.

10.2 Problemas con Docker Compose. Verificar la versión correcta

Uno de los problemas más comunes al trabajar con Docker Compose es el uso de versiones incompatibles o desactualizadas. Para garantizar que todo funcione correctamente, es crucial verificar que se está utilizando la versión adecuada para el entorno de desarrollo. Docker Compose es una herramienta que facilita la gestión de contenedores Docker, pero dependiendo de la versión instalada, pueden surgir incompatibilidades que afecten el rendimiento o la estabilidad del sistema.

Para comprobar la versión de Docker Compose instalada en su entorno, basta con ejecutar el siguiente comando en la terminal:

```
nginx

docker compose version
```

Ilustración 31. Problemas con Docker Compose: Verifique la versión correcta. Fuente ADA.

Este comando le proporcionará la versión exacta que está utilizando, lo que le permitirá confirmar si se encuentra trabajando con la versión más reciente o si necesita realizar una actualización. Mantener Docker Compose actualizado es esencial, ya que las nuevas versiones incluyen mejoras en la funcionalidad, corrección de errores y parches de seguridad que optimizan el rendimiento de los contenedores.

En caso de que la versión que tenga instalada no sea la correcta o esté desactualizada, le recomendamos seguir las instrucciones oficiales para actualizar Docker Compose a la última versión estable. Un desajuste en las versiones de Docker Compose puede llevar a problemas como la incompatibilidad con los archivos de configuración, errores en el

despliegue de contenedores o incluso fallos en la ejecución de los servicios, lo que podría retrasar su trabajo o generar inconvenientes en el entorno de producción.

Por lo tanto, revisar y asegurar que se cuenta con la versión adecuada de Docker Compose es un paso fundamental para evitar complicaciones en el desarrollo y en la administración de contenedores Docker.

10.2.1 Puertos necesarios

Para el correcto funcionamiento del sistema, es imprescindible abrir ciertos puertos en la infraestructura donde se desplegará la aplicación. Los puertos que se deben habilitar son los siguientes:

- **Puerto 80:** Usado para la comunicación HTTP estándar, es el puerto de acceso principal para la web.
- **Puerto 8080:** Se utiliza comúnmente para aplicaciones web que necesitan servir contenido a través de un servidor diferente al puerto 80, o para aplicaciones de pruebas y desarrollo.
- **Puerto 3306:** Este puerto es utilizado por el servicio de bases de datos MySQL, por lo que debe estar abierto para que la aplicación pueda realizar conexiones necesarias a la base de datos.
- **Puerto 3030:** Es necesario para la comunicación con otros servicios o aplicaciones dentro del entorno que utilicen este puerto para sus operaciones internas.

10.2.2 Pasos para el despliegue

10.2.2.1 Eliminar versiones anteriores de Docker

Antes de iniciar con la instalación y configuración de la nueva versión de Docker, es fundamental asegurarse de que no queden restos de instalaciones previas. Esto evita conflictos y problemas durante el proceso de despliegue. Se deben eliminar versiones anteriores de Docker que puedan estar en el sistema, garantizando un entorno limpio para la nueva instalación.

10.2.2.2 Instalar los repositorios de Docker en la máquina

Una vez eliminadas las versiones previas, el siguiente paso es instalar los repositorios de Docker en la máquina. Esto permite gestionar fácilmente las actualizaciones y dependencias necesarias para que Docker funcione de manera correcta. Es recomendable utilizar los repositorios oficiales para garantizar que la versión de Docker instalada esté actualizada y libre de vulnerabilidades conocidas.

10.2.2.3 Instalar Docker y otorgar permisos

Después de instalar los repositorios, se procede a la instalación de Docker. Este paso consiste en obtener la última versión estable de Docker y configurarlo en el sistema. Durante

la instalación, es esencial otorgar los permisos necesarios para que Docker pueda ejecutarse correctamente. Esto incluye la configuración de permisos de usuario y acceso a los recursos del sistema, lo que permitirá que Docker opere de manera óptima en el entorno deseado.

10.2.2.4 Desplegar la aplicación

Con Docker correctamente instalado y configurado, el siguiente paso es proceder al despliegue de la aplicación. En este proceso, se utilizarán imágenes de contenedores para que la aplicación se ejecute de manera aislada y eficiente. El despliegue puede incluir la configuración de variables de entorno, la asignación de puertos necesarios y la integración con los servicios de base de datos que se estén utilizando. Es fundamental verificar que todos los componentes estén funcionando como se espera, y realizar pruebas de conectividad para asegurar que los puertos 80, 8080, 3306 y 3030 estén abiertos y accesibles según sea necesario.

```
cd app
docker compose down
docker compose up -d
```

Ilustración 32. Desplegar la aplicación. Fuente ADA.

Archivo docker-compose.yml:

```
cd /app
nano docker-compose.yml
# Copiar el contenido desde el entorno local
docker compose up -d
```

Ilustración 33. Desplegar la aplicación. Fuente ADA.

11. Presentación de la información en el Aplicativo web

Una vez que los datos han sido correctamente integrados, validados y verificados en la base de datos, el siguiente paso consiste en su integración con el Aplicativo web del Ministerio de Minas y Energía (MME). Este Aplicativo web se presenta como una herramienta esencial para garantizar el acceso a la información de manera estructurada, dinámica y fácilmente accesible para todos los usuarios, promoviendo a su vez los principios fundamentales de transparencia, trazabilidad y participación ciudadana. El objetivo es que, a través de este sistema, los usuarios puedan acceder a la información de

forma clara, precisa y confiable, permitiendo que tanto la ciudadanía como los usuarios institucionales puedan consultar y analizar los datos de manera eficiente.

El diseño y desarrollo del Aplicativo está orientado a ofrecer una experiencia de usuario óptima. En este sentido, el sistema no solo facilita la navegación sino también la comprensión de los datos, proporcionando herramientas y recursos que permiten un uso sencillo y eficaz. Esta plataforma ha sido ideada para responder a las necesidades de quienes requieren una consulta detallada y organizada de la información relacionada con el sector de Minas y Energía.

11.1 Interfaz Gráfica de Usuario

La interfaz gráfica de usuario (GUI) del Aplicativo web se basa en un diseño intuitivo y funcional, que permite la visualización de los datos de manera interactiva y comprensible. A través de una serie de módulos interactivos, los usuarios pueden explorar las imágenes territoriales con facilidad, presentadas de manera clara, atractiva y bien estructurada. Estas imágenes son esenciales para la interpretación precisa de la información geográfica, lo que facilita la comprensión de datos que, en muchos casos, dependen de un contexto geoespacial.

Además de las imágenes, el sistema emplea una variedad de elementos visuales como mapas, tablas, gráficos y reportes, todos diseñados para ofrecer una visión integral y detallada de los datos disponibles. Cada una de estas representaciones permite una interacción directa con la información, lo que favorece la toma de decisiones informadas. Los usuarios internos del sistema, como los funcionarios del MME, y los usuarios externos, como la ciudadanía, tienen la posibilidad de acceder a diferentes tipos de análisis de datos, todos ellos estructurados y accesibles para su descarga en diversos formatos.

11.2 Filtros y Búsquedas

Una de las características más destacadas es la implementación de filtros y mecanismos de búsqueda avanzados, los cuales permiten a los usuarios acceder a la información de forma más segmentada y precisa. Estos mecanismos de búsqueda incluyen la opción de filtrar por departamentos, municipios, zonas estratégicas, indicadores específicos, así como por determinantes o factores relacionados con la viabilidad territorial para proyectos del sector. Este enfoque facilita la búsqueda y clasificación de datos en función de criterios geográficos, económicos o sociales, según las necesidades de cada usuario.

Los usuarios también tienen la posibilidad de combinar múltiples filtros de forma simultánea, lo que les permite afinar aún más los resultados de búsqueda. Esta flexibilidad en los filtros no solo mejora la eficiencia de la búsqueda, sino que también asegura que los resultados obtenidos sean altamente relevantes y ajustados a las consultas realizadas. Al proporcionar un acceso ágil y segmentado a la información, el Aplicativo web se convierte en una herramienta invaluable tanto para quienes gestionan los datos como para quienes los consultan, asegurando una experiencia de usuario fluida y altamente efectiva.

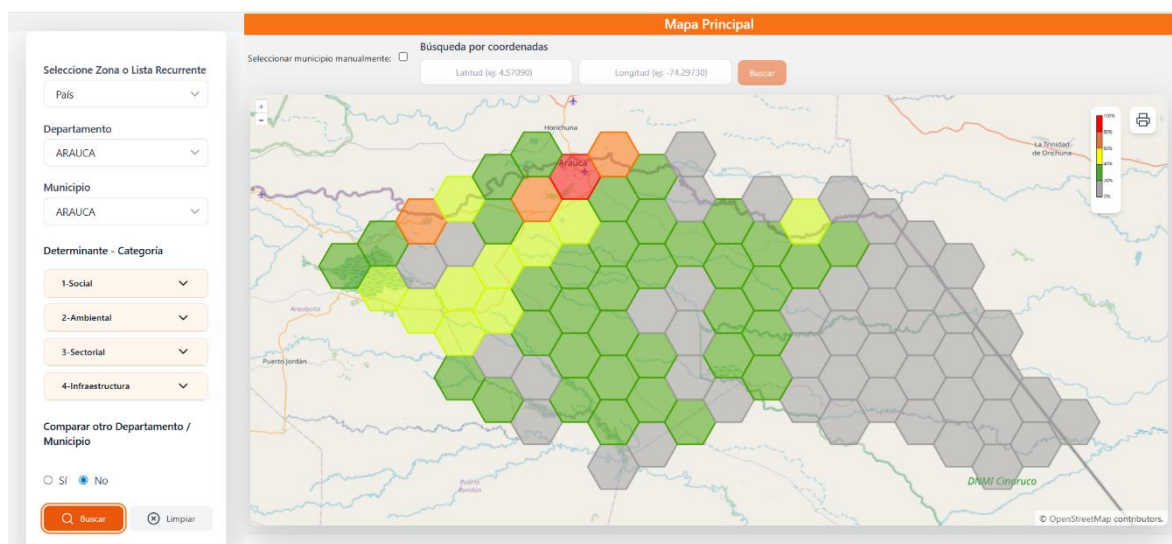


Ilustración 34. Filtros y Búsquedas. Fuente ADA.

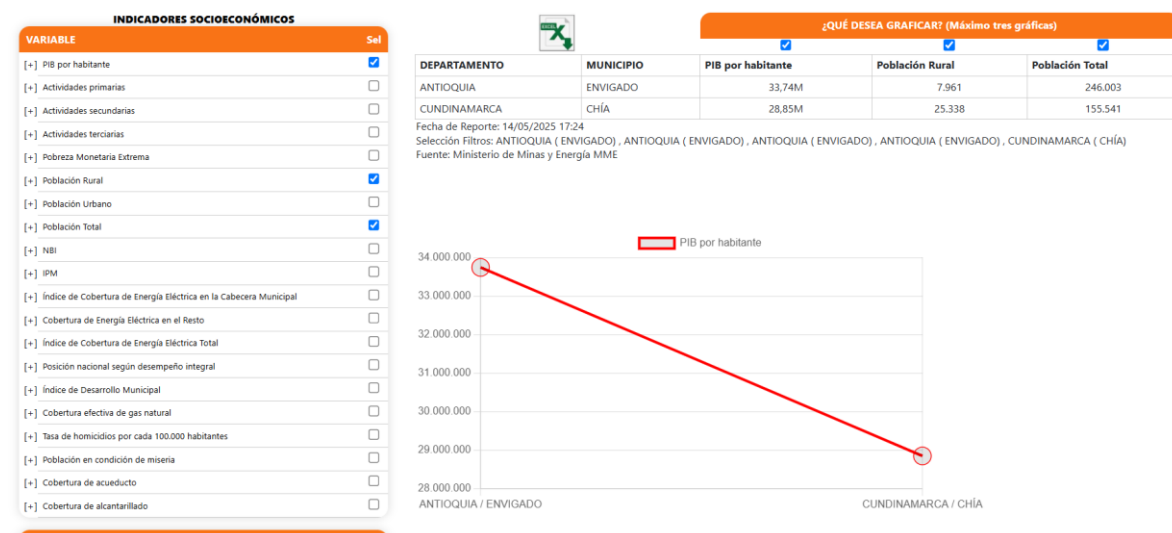


Ilustración 35. Filtros y Búsquedas. Fuente ADA.

12. Importancia del proceso

El proceso de carga de datos desempeña un papel fundamental dentro del funcionamiento del aplicativo web, ya que garantiza que la información que se ingresa en la base de datos sea no solo precisa, sino también fácilmente accesible y, sobre todo, útil para facilitar la toma de decisiones en tiempo real. Este proceso asegura que los datos, al estar bien organizados y correctamente estructurados, puedan ser consultados de manera eficiente por los usuarios, lo que optimiza el rendimiento del sistema y mejora la calidad de los

informes generados a partir de dicha información. Así, la correcta implementación y ejecución de este proceso se convierte en un pilar para el éxito de cualquier estrategia administrativa y operativa que dependa del uso de la plataforma.

12.1 Eficiencia

Uno de los principales beneficios que se obtiene al optimizar el proceso de carga de datos es la mejora en la eficiencia de la plataforma. La rapidez con la que se actualiza la información en la base de datos tiene un impacto directo en la velocidad con la que los usuarios pueden acceder a los datos más recientes. Este flujo constante de actualizaciones permite que el sistema funcione de manera fluida, evitando posibles cuellos de botella o retrasos que puedan afectar la experiencia del usuario. En resumen, las actualizaciones periódicas no solo mejoran la precisión de los datos, sino que también optimizan el rendimiento general del aplicativo, asegurando que los usuarios puedan obtener la información que necesitan de manera rápida y sin interrupciones.

12.2 Escalabilidad

El proceso de carga de datos bien diseñado también es clave para garantizar que el sistema pueda manejar grandes volúmenes de información sin comprometer su rendimiento. A medida que el volumen de datos aumenta, el proceso de carga debe ser capaz de adaptarse y escalar para gestionar este crecimiento sin sacrificar la velocidad o la precisión del sistema. La escalabilidad permite que el aplicativo web siga funcionando de manera eficiente, incluso cuando los datos crecen en tamaño y complejidad, asegurando que no se presenten problemas de lentitud o inestabilidad. Esto se logra mediante la configuración adecuada del sistema, el uso de herramientas de optimización de bases de datos y la implementación de mecanismos automáticos de actualización que gestionan de manera efectiva los grandes volúmenes de información.

12.3 Confiabilidad

La confiabilidad del proceso de carga de datos es otro aspecto esencial para asegurar la calidad de la información almacenada. Un proceso robusto y bien definido no solo garantiza que los datos sean precisos y actualizados, sino que también aumenta la confianza de los usuarios en el sistema. Cuando la información es confiable, la satisfacción del usuario se incrementa significativamente, ya que pueden tomar decisiones informadas basadas en datos verídicos y actualizados. Además, la confiabilidad del sistema facilita la generación de reportes detallados y consistentes, que son cruciales tanto para la gestión diaria de las operaciones como para la elaboración de estrategias a largo plazo. Esto permite que las administraciones operacionales y los responsables de la toma de decisiones a nivel estratégico cuenten con datos sólidos sobre los que basar sus planes de acción.

El proceso de carga de datos es, por lo tanto, esencial para garantizar la operatividad y el buen uso del aplicativo web del Ministerio de Minas y Energía (MME). Este proceso no solo facilita el acceso a datos precisos y útiles, sino que también permite que estos sean aprovechados de manera efectiva para optimizar tanto las tareas diarias como las

decisiones estratégicas. Sin un proceso adecuado de carga de datos, el sistema perdería gran parte de su valor, ya que los usuarios no podrían contar con la información necesaria para tomar decisiones fundamentadas y eficientes.

13. Aprobación

A continuación, firman aprobando:

Nombre	Cargo	Firma	Fecha
Responsable de Área (MME)			
Adriana Patricia Cante Chaparro	Supervisora Contrato designada por FENOGE.		
María Fernanda Ramírez	Supervisor Convenio. MME-OAAS		
Jhon Faustino Chaparro Sierra	Especializado TI funcionario MME		