**Manual Técnico**

**Aplicación Web para la Gestión de PQRSD**

|  |  |
| --- | --- |
| Versión 1.1 | 26 de octubre 2025 |

Un dibujo de un perro

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

# Índice de tablas

[Tabla 1. Requisitos mínimos de hardware para la instalación del proyecto 2](#_Toc212632046)

# REQUISITOS DEL SISTEMA

## Características del sistema

Antes de continuar con la instalación del proyecto en cualquier máquina, es fundamental verificar que el servidor cuente con los siguientes requisitos y configuración previa.

|  |  |
| --- | --- |
| **Características** | **Especificaciones recomendadas** |
| CPU asignada | 1 núcleo virtual (vCPU) |
| Memoria RAM | 2 GB a 4 GB (dependiendo del plan) |
| Almacenamiento | Mínimo 40 GB |
| Base de Datos | PostgreSQL (Instalación nativa con herramientas con pg\_dump) |
| Ancho de Banda | Escalable según la demanda y límites del plan contratado |

Tabla 1. Requisitos mínimos de hardware para la instalación del proyecto

*Fuente: Elaboración por los autores, Año 2025*

## Requisitos de software y configuración previa

|  |  |
| --- | --- |
| **Requisito** | **Descripción** |
| Leguaje | Java Versión 21 o superior |
| JDK | JDK 21 (Adoptium / Oracle) instalado y configurado en el PATH |
| Gestor de Dependencias (Backend) | Apache Maven 3.9+, usado para compilar y ejecutar el proyecto Spring Boot |
| Framework backend | Spring Boot 3.x, basado en Java y Maven |
| Framework frontend | React 18+, desarrollado en Node.js 18+ |
| Gestor de paquetes (Frontend) | npm (incluido con Node.js) |
| Base de datos | PostgreSQL 15.5, instalada localmente |
| Cliente de base de datos (opcional) | pgAdmin 4 o herramienta de línea de comandos psql |
| Sistema operativo recomendado | Windows 10 / 11 (64 bits) |
| Navegador compatible | Google Chrome, Microsoft Edge o Firefox (últimas versiones) |
| Editor o IDE sugerido | IntelliJ IDEA / Visual Studio Code (con extensiones para Java y React) |
| Git | Para clonado y versionado del proyecto |

Tabla 2. Requisitos técnicos del entorno

*Fuente: Elaboración por los autores, Año 2025*

# Requisitos de hardware y software del equipo del cliente

## Usuarios

Los usuarios del sistema no necesitan realizar instalaciones locales ni configuraciones avanzadas, ya que el acceso se efectúa mediante navegadores web. A continuación, se especifican los requisitos mínimos recomendados para un uso óptimo:

* Sistema operativo: Compatible con cualquier sistema operativo que cuente con un navegador web, incluyendo Windows 11, Windows 10, Windows 8.1, Windows 8, Linux y Android.
* Navegador web: Compatible con los principales navegadores, como Google Chrome, Microsoft Edge y Mozilla Firefox. Se recomienda utilizar versiones recientes para garantizar un mejor rendimiento y compatibilidad.
* Resolución de pantalla: El diseño del sistema es completamente adaptable (responsive), permitiendo una visualización óptima en diferentes dispositivos, incluyendo computadoras, tabletas y teléfonos celulares. La interfaz se ajusta automáticamente al tamaño de pantalla, asegurando una experiencia de usuario fluida y cómoda.
* Conectividad: Es necesario contar con una conexión estable a Internet para el correcto funcionamiento del sistema. Se recomienda una conexión de banda ancha con una velocidad mínima de 5 Mbps para garantizar un acceso rápido y sin interrupciones a todas las funciones disponibles.

## Asignadores, funcionarios y administrador

Los usuarios con roles administrativos o de gestión, como asignadores, contratistas y administradores, tampoco requieren instalaciones locales ni configuraciones avanzadas. El acceso se realiza mediante navegadores web desde equipos de escritorio o portátiles. A continuación, se detallan los requisitos mínimos recomendados para un funcionamiento óptimo:

* Sistema operativo: Compatible con sistemas operativos de escritorio que cuenten con un navegador web, incluyendo Windows 11, Windows 10, Windows 8.1, Windows 8 y distribuciones Linux.
* Navegador web: Compatible con los principales navegadores web, como Google Chrome, Microsoft Edge y Mozilla Firefox. Se recomienda utilizar las versiones más recientes para asegurar el correcto funcionamiento y la máxima compatibilidad con todas las funciones del sistema.
* Resolución de pantalla: La interfaz para estos usuarios está optimizada exclusivamente para computadoras, garantizando una visualización clara, ordenada y eficiente en pantallas de mayor tamaño. No se recomienda el uso desde dispositivos móviles, ya que ciertas funcionalidades pueden no mostrarse correctamente.
* Conectividad: Es indispensable contar con una conexión a Internet estable y de buena calidad. Se recomienda una velocidad mínima de 10 Mbps para asegurar la carga rápida de datos, el acceso simultáneo a múltiples módulos y un desempeño fluido en operaciones administrativas.

# Portabilidad del sistema

El sistema presenta una alta portabilidad al estar desarrollado bajo una arquitectura cliente-servidor, utilizando tecnologías web modernas y de código abierto. Esta arquitectura permite una clara separación entre el frontend y el backend, facilitando el mantenimiento, la escalabilidad y la implementación en distintos entornos sin necesidad de herramientas adicionales como Docker u orquestadores externos.

Sus principales ventajas incluyen:

* Arquitectura cliente-servidor: El sistema está dividido en dos capas principales: un cliente desarrollado en React, encargado de la interfaz de usuario y la interacción con el usuario final, y un servidor implementado con Spring Boot, responsable de la lógica de negocio y la gestión de datos. Esta separación permite mantener y actualizar cada componente de forma independiente, mejorando la escalabilidad y la estabilidad del sistema.
* Backend en Spring Boot (Java): El servidor está desarrollado con Spring Boot, lo que permite un despliegue nativo directo en el servidor sin requerir contenedores ni configuraciones complejas. Puede ejecutarse fácilmente mediante un archivo .jar o integrarse como un servicio del sistema operativo. Su estructura modular y la gestión automática de dependencias proporcionan un entorno robusto y flexible.
* Frontend en React: El cliente, desarrollado con React, ofrece una interfaz moderna, dinámica y eficiente. Su diseño basado en componentes facilita la actualización y personalización de la interfaz sin afectar la lógica del backend. Además, el diseño es responsive, asegurando compatibilidad con diferentes resoluciones de pantalla y dispositivos.
* Base de datos PostgreSQL: El sistema utiliza PostgreSQL como gestor de base de datos, reconocido por su estabilidad, seguridad y rendimiento. Está configurado para integrarse directamente con el backend sin depender de servicios externos, y puede migrarse fácilmente a otras instancias o servidores manteniendo la integridad y consistencia de los datos

# Adaptabilidad a cambios del entorno

El sistema ha sido diseñado bajo una arquitectura cliente-servidor modular y flexible, lo que permite una alta capacidad de adaptación a nuevas necesidades funcionales y técnicas. Su estructura favorece el mantenimiento, la escalabilidad y la integración con otros servicios o aplicaciones externas.

Entre sus principales características se destacan:

* Arquitectura modular y flexible: La separación entre el cliente (React) y el servidor (Spring Boot) permite realizar actualizaciones, mejoras o correcciones de manera independiente, sin afectar la operatividad global del sistema. Esta independencia facilita la evolución tecnológica y la incorporación de nuevas funcionalidades.
* Escalabilidad adaptable: El sistema admite escalabilidad vertical, mediante la ampliación de los recursos del servidor (CPU, memoria o almacenamiento), y escalabilidad horizontal, al distribuir las solicitudes entre múltiples instancias del backend. Esto garantiza un rendimiento estable incluso con un alto número de usuarios concurrentes.
* Backend estructurado en Spring Boot: La organización modular del backend, basada en controladores, servicios y repositorios, permite un crecimiento ordenado y facilita el mantenimiento del código. Además, el uso de controladores REST simplifica la comunicación con el frontend y con servicios externos.
* Compatible con APIs RESTful: La arquitectura REST implementada en el backend facilita la interoperabilidad con otros sistemas o aplicaciones externas, permitiendo el intercambio de datos mediante peticiones HTTP seguras y estandarizadas.
* Frontend en React: Las interfaces están desarrolladas en React, utilizando componentes reutilizables que aseguran consistencia visual y eficiencia en la renderización. El diseño es responsivo, adaptándose a distintas resoluciones y dispositivos, garantizando una experiencia de usuario óptima tanto en equipos de escritorio como en dispositivos móviles.
* Base de datos PostgreSQL: La base de datos está implementada en PostgreSQL, ofreciendo integridad transaccional, seguridad y soporte para operaciones complejas. Su configuración permite una conexión directa y eficiente con el backend, facilitando la administración y la portabilidad a otros entornos si es necesario.

# Plan de contingencia

El sistema, desplegado de forma nativa en un servidor local, cuenta con medidas preventivas y procedimientos de recuperación orientados a garantizar la continuidad del servicio ante fallos técnicos, errores humanos o incidentes de seguridad. Estas estrategias buscan minimizar el tiempo de inactividad y asegurar la integridad de la información.

Entre las acciones recomendadas se incluyen:

* Recuperación ante desastres:

En caso de pérdida de datos o fallas críticas en la aplicación, el sistema permite restaurar la información a partir de respaldos automáticos de la base de datos PostgreSQL, generados diariamente a las 2:00 a.m.

Adicionalmente, los administradores pueden generar copias de seguridad manualmente desde la interfaz de administración mediante un botón dedicado.

En caso de una pérdida total del servicio, el sistema puede ser reinstalado desde el repositorio de código fuente en GitHub, lo que permite restablecer el entorno rápidamente.

* Escalabilidad y continuidad operativa:

Aunque el sistema se ejecuta en un servidor local, se recomienda mantener un servidor de respaldo con una copia actualizada del sistema y de la base de datos para garantizar la recuperación inmediata en caso de fallas graves o mantenimiento no programado.

La estructura modular del sistema permite su migración a otro servidor o entorno de producción con configuraciones mínimas, sin alterar el funcionamiento general.

Medidas de seguridad preventiva: La seguridad del sistema depende principalmente de las configuraciones implementadas en el servidor local donde se aloje.

* Se recomienda:

Mantener el servidor protegido mediante firewall, acceso restringido por roles y protocolos seguros de conexión (HTTPS o SSH).

* + Limitar el acceso a puertos o servicios innecesarios para reducir la superficie de ataque.
  + Actualizar regularmente el sistema operativo, dependencias de Java, librerías de React y PostgreSQL.
  + Restringir el acceso a la interfaz de administración solo a personal autorizado.
* Gestión de registros (logs):

El sistema genera registros internos de actividad, almacenando información sobre el usuario, ruta accedida, tipo de petición (GET, POST, PATCH, PUT o DELETE) y el estado de la respuesta.

Estos registros permiten realizar auditorías básicas y verificar la correcta interacción de los usuarios con el sistema, aunque no contemplan métricas de rendimiento ni alertas automáticas.

* Código fuente y control de versiones:

El proyecto se encuentra disponible de forma pública en GitHub, lo que facilita la recuperación del código, el seguimiento de cambios y la colaboración en futuras mejoras. Esta práctica contribuye a la transparencia, mantenibilidad y posibilidad de restauración en caso de pérdida del entorno local.

# Diccionario de datos

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| id | SERIAL | Identificador único de cada registro en la tabla. Se genera automáticamente. | PK |
| pq\_id | INTEGER | Identificador del registro asociado en la tabla principal de PQRSD. Representa la relación entre el adjunto y la solicitud correspondiente. | FK → pqs(id) |
| nombre\_archivo | VARCHAR | Nombre original del archivo adjunto cargado por el usuario. |  |
| ruta\_archivo | VARCHAR | Ruta o ubicación en el servidor donde se almacena físicamente el archivo. |  |
| created\_at | TIMESTAMP | Fecha y hora en que se realizó la carga del archivo. Permite llevar control temporal de los registros |  |
| Respuesta | BOOLEAN | Indica si el archivo corresponde a una respuesta emitida por la entidad (TRUE) o si fue un archivo adjunto por el ciudadano (FALSE). |  |
| deleted\_at | TIMESTAMP | Fecha y hora en la que el registro fue eliminado lógicamente del sistema. |  |

Tabla 3. Diccionario de Datos – Adjuntos\_pq

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| Id | SERIAL | Identificador único de cada área registrada. Se genera automáticamente por el sistema. | PK |
| codigo\_dep | VARCHAR | Código identificador del departamento o dependencia dentro de la Secretaría de Tránsito y Transporte. |  |
| Nombre | VARCHAR | Nombre del área o dependencia responsable (por ejemplo: Atención al Ciudadano, Jurídica, Técnica, etc.). |  |
| deleted\_at | TIMESTAMP | Fecha y hora en que el registro fue eliminado lógicamente del sistema. Si el valor es nulo, el área sigue activa. |  |

Tabla 4. Diccionario de Datos – Areas\_resp

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| Id | SERIAL | Identificador único del registro de auditoría. Se genera automáticamente por el sistema. | PK |
| username | VARCHAR | Nombre de usuario que realizó la acción dentro del sistema. Permite rastrear quién ejecutó una operación. |  |
| action | VARCHAR | Acción realizada por el usuario (por ejemplo: crear, actualizar, eliminar, iniciar sesión). |  |
| method | VARCHAR | Método HTTP utilizado en la solicitud (por ejemplo: GET, POST, PUT, DELETE). |  |
| endpoint | TEXT | Ruta o endpoint del sistema que fue accedido durante la acción. |  |
| status\_code | INTEGER | Código de estado HTTP devuelto por el servidor tras la ejecución de la acción (por ejemplo: 200, 404, 500). |  |
| timestamp | TIMESTAMP | Fecha y hora exacta en la que ocurrió la acción registrada. |  |

Tabla 5. Diccionario de Datos – Audit\_log

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| id | SERIAL | Identificador único del departamento. Se genera automáticamente por el sistema. | PK |
| nombre | VARCHAR | Nombre del departamento (por ejemplo: Cundinamarca, Tolima, Huila). |  |
| codigo\_dane | VARCHAR | Código asignado por el DANE que identifica oficialmente el departamento dentro del territorio nacional. |  |

Tabla 6. Diccionario de Datos – Departamentos

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| id | SERIAL | Identificador único del estado de la PQRSD. Se genera automáticamente por el sistema. | PK |
| nombre | VARCHAR | Nombre del estado asignado a una PQRSD (por ejemplo: “Radicado”, “En proceso”, “Resuelto”, “Cerrado”). |  |
| color | VARCHAR | Código de color (en formato hexadecimal o texto) que representa visualmente el estado dentro de la interfaz del sistema. |  |
| descripcion | VARCHAR | Descripción breve del significado o propósito del estado. |  |

Tabla 7. Diccionario de Datos – Estados\_pq

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| id | SERIAL | Identificador único del género. Se genera automáticamente por el sistema. | PK |
| nombre | VARCHAR | Nombre del género registrado (por ejemplo: Masculino, Femenino, Otro). |  |

Tabla 8. Diccionario de Datos – Géneros

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| id | SERIAL | Identificador único del municipio. Se genera automáticamente por el sistema. | PK |
| nombre | VARCHAR | Nombre oficial del municipio (por ejemplo: Girardot, Nilo, Tocaima). |  |
| codigo\_dane | VARCHAR | Código asignado por el DANE que identifica de forma única al municipio dentro del país. |  |
| departamento\_id | INTEGER | Identificador del departamento al que pertenece el municipio. |  |

Tabla 9. Diccionario de Datos – Municipios

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| Id | SERIAL | Identificador único del tipo de documento. Se genera automáticamente por el sistema. | PK |
| nombre | VARCHAR | Nombre del tipo de documento (por ejemplo: Cédula de Ciudadanía, Tarjeta de Identidad, Pasaporte, NIT, etc.). |  |

Tabla 10. Diccionario de Datos – Tipos\_docs

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| id | SERIAL | Identificador único del tipo de persona. Se genera automáticamente por el sistema. | PK |
| nombre | VARCHAR | Nombre o categoría del tipo de persona dentro del sistema (por ejemplo: Ciudadano, funcionario STTG, Asignador STTG, Administrador). |  |

Tabla 11. Diccionario de Datos – Tipos\_personas

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| Id | SERIAL | Identificador único de la persona. Se genera automáticamente por el sistema. | PK |
| Nombre | VARCHAR | Nombres completos de la persona registrada. |  |
| Apellido | VARCHAR | Apellidos de la persona registrada. |  |
| tipo\_doc | INTEGER | Identificador del tipo de documento (por ejemplo: cédula, tarjeta de identidad, pasaporte). | FK → tipos\_docs(id) |
| Dni | VARCHAR | Número del documento de identificación personal. |  |
| fecha\_nacimiento | DATE | Fecha de nacimiento de la persona. |  |
| tipo\_persona | INTEGER | Define el tipo de persona (por ejemplo: ciudadano, funcionario, asignador, administrador). | FK → tipos\_personas(id) |
| Telefono | VARCHAR | Número de teléfono de contacto. |  |
| Direccion | VARCHAR | Dirección de residencia o contacto de la persona. |  |
| tratamiento\_datos | BOOLEAN | Indica si la persona autoriza el tratamiento de sus datos personales según la política de privacidad. |  |
| municipio\_id | INTEGER | Identificador del municipio al que pertenece la persona (relacionado con la tabla de municipios). | FK → municipios(id) |
| Genero | INTEGER | Identificador del género de la persona (masculino, femenino u otro). | FK → generos(id) |
| created\_at | TIMESTAMP | Fecha y hora de creación del registro. |  |
| updated\_at | TIMESTAMP | Fecha y hora de la última actualización del registro. |  |
| deleted\_at | TIMESTAMP | Fecha y hora en que el registro fue eliminado lógicamente. Si es nulo, el registro sigue activo. |  |

Tabla 12. Diccionario de Datos – Personas

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| id | SERIAL | Identificador único del tipo de PQRSD. Se genera automáticamente por el sistema. | PK |
| nombre | VARCHAR | Nombre del tipo de solicitud registrada (por ejemplo: Petición, Queja, Reclamo, Sugerencia, Denuncia). |  |
| dias\_habiles\_respuesta | INTEGER | Número de días hábiles establecidos para dar respuesta según el tipo de PQRSD, conforme a la normativa institucional. |  |

Tabla 13. Diccionario de Datos – Tipos\_pq

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| Id | SERIAL | Identificador único de la PQRSD. Se genera automáticamente por el sistema. | PK |
| numero\_radicado | VARCHAR | Código o número de radicado asignado automáticamente a la solicitud. |  |
| detalle\_asunto | TEXT | Descripción principal del asunto o motivo de la petición, queja, reclamo, sugerencia o denuncia. |  |
| tipo\_pq\_id | INTEGER | Identificador del tipo de solicitud (Petición, Queja, Reclamo, Sugerencia o Denuncia). | FK → tipos\_pq(id) |
| solicitante\_id | INTEGER | Identificador del ciudadano o persona que radica la solicitud. | FK → personas(id) |
| fecha\_radicacion | DATE | Fecha en la que la solicitud fue registrada en el sistema. |  |
| hora\_radicacion | TIME | Hora exacta en la que se realizó la radicación. |  |
| fecha\_resolucion\_estimada | DATE | Fecha estimada para la resolución de la solicitud, de acuerdo con los tiempos legales. |  |
| fecha\_resolucion | DATE | Fecha real en la que se resolvió la solicitud. |  |
| radicador\_id | INTEGER | Identificador del funcionario que realizó la radicación en el sistema (en caso de que no sea vía web). | FK → usuarios(id) |
| responsable\_id | INTEGER | Identificador del funcionario asignado para atender y responder la solicitud. | FK → usuarios(id) |
| Web | BOOLEAN | Indica si la solicitud fue radicada directamente desde la página web (TRUE) o por un funcionario (FALSE). |  |
| Respuesta | TEXT | Respuesta o resolución emitida por el funcionario responsable. |  |
| detalle\_descripcion | VARCHAR | Observaciones o detalles adicionales del proceso de atención de la solicitud. |  |
| deleted\_at | TIMESTAMP | Fecha y hora en que la solicitud fue eliminada lógicamente. Si es nula, la solicitud sigue activa. |  |

Tabla 14. Diccionario de Datos – Pqs

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| Id | SERIAL | Identificador único del registro en el historial de cambios de estado de una PQRSD. Se genera automáticamente por el sistema. | PK |
| pq\_id | INTEGER | Identificador de la PQRSD a la que pertenece el historial. | FK → pqs(id) |
| estado\_id | INTEGER | Identificador del estado al que cambió la PQRSD (por ejemplo: Radicada, En trámite, Resuelta, Cerrada). | FK → estados\_pq(id) |
| cambiado\_por\_id | INTEGER | Identificador del funcionario o usuario que realizó el cambio de estado. | FK → usuarios(id) |
| observacion | TEXT | Comentarios u observaciones relacionadas con el cambio de estado (por ejemplo, motivo del cambio o avance del caso). |  |
| fecha\_cambio | TIMESTAMP | Fecha y hora exacta en la que se realizó el cambio de estado. |  |
| deleted\_at | TIMESTAMP | Fecha y hora en que el registro fue eliminado lógicamente. Si es nulo, el registro sigue activo. |  |

Tabla 15. Diccionario de Datos – Historial\_estados\_pq

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| id | SERIAL | Identificador único del permiso dentro del sistema. Se genera automáticamente por el sistema. | PK |
| tabla | VARCHAR | Nombre de la tabla o módulo del sistema al que aplica el permiso (por ejemplo: usuarios, pqs, personas). |  |
| accion | VARCHAR | Acción o tipo de operación que puede realizar el rol sobre la tabla (por ejemplo: crear, leer, actualizar, eliminar). |  |
| descripcion | TEXT | Descripción detallada del permiso, indicando su propósito o alcance dentro del sistema. |  |

Tabla 16. Diccionario de Datos – Permisos

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| Id | SERIAL | Identificador único del registro en la relación entre roles y permisos. Se genera automáticamente por el sistema. | PK |
| rol\_id | INTEGER | Identificador del rol asociado al permiso, relaciona con la tabla roles. | FK → roles(id) |
| permiso\_id | INTEGER | Identificador del permiso asignado al rol, relaciona con la tabla permisos. | FK → permisos(id) |

Tabla 17. Diccionario de Datos – Roles\_permisos

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| Id | SERIAL | Identificador único del rol dentro del sistema. Se genera automáticamente por el sistema. | PK |
| Nombre | VARCHAR | Nombre del rol asignado al usuario (por ejemplo: Administrador, Asignador STTG, Funcionario STTG, Ciudadano). |  |
| created\_at | DATE | Fecha en la que se creó el registro del rol en el sistema. |  |
| updated\_at | DATE | Fecha de la última actualización del registro del rol. |  |

Tabla 18. Diccionario de Datos – Roles

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| id | SERIAL | Identificador único del usuario dentro del sistema. Se genera automáticamente por el sistema. | PK |
| correo | VARCHAR | Correo electrónico del usuario utilizado para el inicio de sesión y la comunicación oficial. Debe ser único. |  |
| contrasena | VARCHAR | Contraseña cifrada del usuario para el acceso al sistema. |  |
| is\_enable | BOOLEAN | Indica si la cuenta del usuario está activa (TRUE) o deshabilitada (FALSE). |  |
| reset\_token | VARCHAR | Token temporal utilizado para restablecer la contraseña del usuario cuando lo solicita. |  |
| reset\_token\_expiration | TIMESTAMP | Fecha y hora de expiración del token de restablecimiento de contraseña. |  |
| persona\_id | INTEGER | Identificador de la persona asociada al usuario, permitiendo vincular la información personal registrada. | FK → personas(id) |
| rol\_id | INTEGER | Identificador del rol asignado al usuario, el cual determina los permisos y accesos dentro del sistema. | FK → roles(id) |
| created\_at | TIMESTAMP | Fecha y hora de creación del registro del usuario. |  |
| updated\_at | TIMESTAMP | Fecha y hora de la última actualización del registro. |  |
| deleted\_at | TIMESTAMP | Fecha y hora en que el registro fue eliminado lógicamente. Si es nulo, el usuario sigue activo en el sistema. |  |

Tabla 19. Diccionario de Datos – Usuarios

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo de dato** | **Descripción** | **Clave** |
| Id | SERIAL | Identificador único del registro de asignación de responsables dentro del sistema. Se genera automáticamente por el sistema. | PK |
| persona\_responsable\_id | INTEGER | Identificador de la persona designada como responsable de atender una o varias PQRSD. | FK → personas(id) |
| area\_id | INTEGER | Identificador del área o dependencia a la que pertenece el responsable. | FK → areas\_resp(id) |
| fecha\_asignacion | TIMESTAMP | Fecha y hora en que se asignó al responsable dentro del sistema o para un caso específico. |  |
| is\_active | BOOLEAN | Indica si el responsable actualmente se encuentra activo dentro del sistema o área asignada. |  |
| deleted\_at | TIMESTAMP | Fecha y hora en que el registro fue eliminado lógicamente. Si es nulo, el registro sigue activo. |  |

Tabla 20. Diccionario de Datos – Responsables\_pq

*Fuente: Elaboración por los autores a partir de PostgreSQL, Año 2025*