## Testing summary:

|  |  |
| --- | --- |
| Project Name | Gestordocs |
| Release Version | V. 1.0 |
| Tester Name | David Fernando Carrillo Ramírez  Daniel Jurado Alfonso |
| Testing Date(s) | 11 de Agosto de 2025 |

## Description of testing information fields:

|  |  |
| --- | --- |
| Field | Description |
| Project Name | The name of the project being tested. |
| Release Version | The version of the project/software (first test could be considered 1.0). |
| Tester Name | Name of tester who carried out these tests. |
| Testing Date(s) | Date(s) when tests carried out - this could be a single day or a series of days. If tests are spread across a larger period of time, a testing date could be noted for each test case. |
| Test Case # | Unique ID for each test case. Follow some convention to indicate types of test. E.g. ‘TC\_UI\_1′ indicating ‘user interface test case #1′. |
| Test Priority  (Low/Medium/High) | How important each test is. Test priority for business rules and functional test cases can be medium or higher whereas minor user interface cases can be low priority. |
| Test Title/Name | Test case title. E.g. Verify login page with valid username and password. |
| Test Summary | A description of what the test should achieve. |
| Test Steps | List all test execution steps in detail. Write test steps in the order in which these should be executed. Make sure to provide as much detail as you can. A numbered list would be a good idea. |
| Test Data | List/describe any test data used for this test case. This way the actual inputs used can be tracked against test results. E.g. username & password for verifying a login. |
| Expected Result | What should be the system output after test execution? Describe the expected result in detail including any messages/errors that should be displayed on screen. |
| Actual Result | What is the actual result after the test has been executed? Describe any relevant system behaviour after test execution. |
| Pre-condition | Any prerequisite that must be fulfilled before execution of this test case. List all pre-conditions in order to successfully execute this test case. |
| Post-condition | What should be the state of the system after executing this test case? |
| Status  (Pass/Fail) | If actual result is not as per the expected result mark this test as failed. Otherwise update as passed. |
| Notes/Comments | Use this area for any additional notes/comments/questions. This area is to support the above fields (e.g. if there are some special conditions which can’t be described in any of the above fields or if there are questions related to expected or actual results). |

## Test case #1: Verificación de creación de tablas y claves

Hecho por: David Carrillo

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 1 |
| Test Priority | 5 |
| Test Title/Name | Validar estructura y creación de claves primarias y foráneas |
| Test Summary | Confirmar que las tablas se crean correctamente y que las claves primarias y foráneas están definidas y activas. |
| Test Steps | * Ejecutar el script de creación completo. * Listar tablas creadas * Listar claves primarias * Listar claves foráneas |
| Test Data | No aplica (solo estructura). |
| Expected Result | Todas las tablas y restricciones se crean sin errores. |
| Actual Result | Todas las tablas y restricciones se crearon de manera correcta sin errores. |
| Status | Funcional. |
| Pre-condition | BD vacía. |
| Post-condition | Estructura creada correctamente. |
| Notes/Comments |  |

## Test case #2: CHECKs y UNIQUE (validación de constraints)

Hecho por: David Carrillo

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 2 |
| Test Priority | 5 |
| Test Title/Name | Validar UNIQUE en correo y CHECK en estado/rol en usuarios |
| Test Summary | Insertar datos válidos e inválidos y confirmar que BD acepta o rechaza según constraints. |
| Test Steps | * Insertar usuario válido * Intentar insertar usuario con mismo correo (esperado: falla por UNIQUE) * Intentar insertar usuario con estado inválido (esperado: falla por CHECK) * Intentar insertar usuario con rol inválido (esperado: falla por CHECK) |
| Test Data | * Usuario válido [juan@example.com](mailto:juan@example.com). * Usuario duplicado [juan@example.com](mailto:juan@example.com). * Usuario con estado = 5 (inválido). * Usuario con rol = 9 (inválido). |
| Expected Result | Primer INSERT OK (devuelve id).  Segundo INSERT falla por violación de UNIQUE en correo.  Tercero y cuarto INSERT fallan por CHECK en estado/rol. |
| Actual Result | Se insertó el usuario de manera válida con el id=1.  Al insertar el usuario con el mismo correo, se muestra el error esperado por violación de restricción unique.  Al insertar el usuario con estado inválido, se muestra el error esperado por violación de restricción check.  Al insertar el usuario con el rol inválido, se muestra el error esperado por violación de restricción check. |
| Status | Funcional. |
| Pre-condition | Script DDL ejecutado (BD limpia). |
| Post-condition | Solo existe juan@example.com si las inserciones inválidas fueron rechazadas. |
| Notes/Comments |  |

## Test case #3: Integridad referencial y ON DELETE

Hecho por: David Carrillo

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 3 |
| Test Priority | 5 |
| Test Title/Name | Probar cascada y set null en eliminaciones (usuarios → carpetas → archivos) |
| Test Summary | Crear un usuario, carpeta y archivo; eliminar el usuario y verificar comportamiento de FK (ON DELETE). |
| Test Steps | 1. Crear usuario propietario 2. Crear carpeta para ese usuario 3. Crear archivo en la carpeta (y con id\_usuario = propietario) 4. Verificar registros creados. 5. Eliminar usuario 6. Verificar estado después de eliminar |
| Test Data | * Usuario propietario: prop1@example.com (rol 0). * Carpeta: Documentos\_test. * Archivo: miarchivo.txt (tamano 1024). |
| Expected Result | * Al eliminar usuario: carpetas del usuario desaparecen por ON DELETE CASCADE. * archivos asociados se eliminan por fk\_archivos\_usuario ON DELETE CASCADE (aunque id\_carpeta tiene ON DELETE SET NULL, la eliminación por id\_usuario debe borrar el archivo). * No deben quedar registros huérfanos. |
| Actual Result | Se elimina correctamente el usuario y junto al usuario la carpeta y archivo que están asociados a este. |
| Status | Funcional. |
| Pre-condition | Script DDL ejecutado. No colisiones de correo. |
| Post-condition | Usuario, carpeta y archivo borrados. |
| Notes/Comments |  |

## Test case #4: Triggers: update\_usuario\_fecha y notificaciones por solicitud

Hecho por: David Carrillo

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 4 |
| Test Priority | 5 |
| Test Title/Name | Probar trigger de actualización de fecha y trigger notificador de solicitudes\_registro |
| Test Summary | Actualizar usuario y crear solicitud para validar triggers. |
| Test Steps | 1. Crear dos admins 2. Insertar solicitud de registro (disparará trigger notificar\_solicitud\_registro) 3. Consultar notificaciones creadas (debe insertar una notificación por cada admin con rol=1) 4. Actualizar un admin y verificar fecha en usuarios (trigger update\_usuario\_fecha antes UPDATE) |
| Test Data | Admin A: admin1@example.com (rol = 1)  Admin B: admin2@example.com (rol = 1)  Nueva solicitud: datos de solicitante@example.com. |
| Expected Result | Al insertar la solicitud, se crean notificaciones para cada admin rol=1 con id\_solicitud =1.  Al actualizar usuario, campo fecha se actualiza a timestamp actual. |
| Actual Result | Se insertar correctamente una solicitud de registro a cada admin, y al actualizar el usuario el campo de fecha se actualiza correctamente. |
| Status | Funcional. |
| Pre-condition | Script DDL ejecutado. No admins existentes (o limpiar admin de prueba). |
| Post-condition | Notificaciones creadas. Admins actualizados. |
| Notes/Comments |  |

## Test case #5: Funciones: obtener\_archivos\_compartidos\_conmigo / obtener\_detalle\_archivo\_compartido / verificar\_archivo\_compartido\_usuario

Hecho por: David Carrillo

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 5 |
| Test Priority | 4 |
| Test Title/Name | Validar funciones de obtención y verificación de compartidos |
| Test Summary | Crear datos de prueba y ejecutar las funciones para validar resultados. |
| Test Steps | 1. Crear propietario y receptor 2. Insertar carpeta (opcional) y archivo 3. Crear solicitud compartido 4. Ejecutar función obtener\_archivos\_compartidos\_conmigo 5. Ejecutar obtener\_detalle\_archivo\_compartido(sol\_comp\_id, 'rec1@example.com') 6. Ejecutar verificar\_archivo\_compartido\_usuario(sol\_comp\_id, 'rec1@example.com') |
| Test Data | Propietario: prop2@example.com (id: prop\_id)  Receptor: rec1@example.com (id: rec\_id)  Archivo: reporte.pdf (id archivo\_id)  Solicitud compartido: sol\_comp\_id con correo = rec1@example.com, estado = 1 |
| Expected Result | Las funciones retornan las filas con los datos correctos del propietario, archivo y carpeta. |
| Actual Result | Las funciones retornan correctamente las filas esperadas con los datos correctos. |
| Status | Funcional. |
| Pre-condition | Script DDL ejecutado. No colisiones de correos. |
| Post-condition | Registros de prueba creados y visibles. |
| Notes/Comments | Sugerencia: En el Casting de las funciones utilizar %TYPE para mayor facilidad a la hora de cambiar algún tipo de dato. |

## Test case #6: Cambiar y marcar estado (funciones de update)

Hecho por: David Carrillo

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 6 |
| Test Priority | 4 |
| Test Title/Name | cambiar\_estado\_archivo\_compartido y marcar\_archivo\_como\_visto |
| Test Summary | Verificar que las funciones actualizan estado y devuelven filas afectadas correctas. |
| Test Steps | 1. Ejecutar cambiar\_estado\_archivo\_compartido a 0. 2. Revertir estado a 1 para la segunda función 3. Ejecutar marcar\_archivo\_como\_visto |
| Test Data | Id de archivo = sol\_comp\_id |
| Expected Result | * cambiar\_estado\_archivo\_compartido devuelve 1 y actualiza estado a 0. * marcar\_archivo\_como\_visto solo cambia si estado = 1 y devuelve 1; si ya es 2 devuelve 0. |
| Actual Result | El sistema no permite marcar el estado del archivo como visto ya que la restricción check no lo permite, la función marcar\_archivo\_como\_visto está mal hecha. |
| Status | No funcional. |
| Pre-condition | Tener un solicitudes\_compartidos con estado = 1. |
| Post-condition | solicitudes\_compartidos.estado = 2 |
| Notes/Comments | Revisar la función marcar\_estado\_como\_visto, ya que viola la restricción check, es decir la función intenta cambiar el estado a 2 y la restricción solo permite que el estado sea 0/1. |

## Test case #7: Estadísticas y búsquedas

Hecho por: David Carrillo

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 7 |
| Test Priority | 4 |
| Test Title/Name | obtener\_estadisticas\_compartidos\_usuario, obtener\_estadisticas\_compartidos\_globales, buscar\_archivos\_compartidos, contar\_archivos\_compartidos\_usuario |
| Test Summary | Validar conteos totales/hoy y búsqueda ILIKE. |
| Test Steps | 1. Insertar solicitudes 2. Ejecutar funciones |
| Test Data | Receptor estuser@example.com  2 solicitudes con fecha hoy, 1 solicitud fecha ayer |
| Expected Result | obtener\_estadisticas\_compartidos\_usuario debería devolver total\_recibidos = 3 y recibidos\_hoy = 2.  obtener\_estadisticas\_compartidos\_globales refleja totales globales incluyendo las filas creadas.  buscar\_archivos\_compartidos devuelve los archivos que contengan 'informe' en nombre o propietario.  contar\_archivos\_compartidos\_usuario devuelve 3. |
| Actual Result | Se ejecutan correctamente las funciones de estadísticas y búsquedas con los resultados esperados. |
| Status | Funcional. |
| Pre-condition | Insertar varios solicitudes\_compartidos con fechas (hoy y días anteriores). |
| Post-condition | Registros de pruebas creados. |
| Notes/Comments |  |

## Test case #8: Tokens: limpiar e invalidar

Hecho por: David Carrillo

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 8 |
| Test Priority | 5 |
| Test Title/Name | limpiar\_tokens\_expirados e invalidar\_tokens\_usuario |
| Test Summary | Insertar tokens expirados y válidos, ejecutar limpieza e invalidación. |
| Test Steps | 1. Crear usuario 2. Insertar tokens 3. Ejecutar limpiar\_tokens\_expirados() 4. Reinsertar (si fue eliminado) o verificar token válido permanece. 5. Ejecutar invalidar\_tokens\_usuario(token\_user\_id) |
| Test Data | Usuario tokenuser@example.com (id token\_user\_id)  Token válido con fecha\_expiracion > now()  Token expirado con fecha\_expiracion < now() |
| Expected Result | limpiar\_tokens\_expirados() elimina solo tokens con fecha\_expiracion < now().  invalidar\_tokens\_usuario marca usado = 1 para tokens no usados y no expirados (y devuelve filas afectadas). |
| Actual Result | Se ejecutan correctamente las funciones con los datos esperados. |
| Status | Funcional. |
| Pre-condition | Script DDL ejecutado. |
| Post-condition | Tokens expirados eliminados / tokens previos marcados usado=1. |
| Notes/Comments |  |

## Test case #9: Índices y plan de ejecución (rendimiento básico)

Hecho por: David Carrillo

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 9 |
| Test Priority | 3 |
| Test Title/Name | Verificar que consultas críticas usan índices (EXPLAIN ANALYZE) |
| Test Summary | Ejecutar EXPLAIN ANALYZE sobre consultas por correo, estado, tipo, y búsquedas ILIKE para revisar uso de índices. |
| Test Steps | 1. Ver plan para búsqueda por correo 2. Por estado compuesto 3. Por tipo de archivo 4. Para ILIKE |
| Test Data | Muestras de datos o dataset de prueba. |
| Expected Result | Consultas por correo, estado, tipo usan índices (Index Scan).  ILIKE puede requerir índice GIN+trgm para ser eficiente; si no está, aparece Seq Scan. |
| Actual Result | Funcionan Correctamente los índices. |
| Status | Funcional. |
| Pre-condition | Datos suficientes en tablas (solicitudes\_compartidos, archivos, usuarios) — inserta unos miles de filas si quieres prueba real. |
| Post-condition | Ninguno (solo lectura). |
| Notes/Comments | Seq Scan en ILIKE y mucho volumen, crea índices. |

## Test case #10: Validaciones adicionales y casos borde

Hecho por: David Carrillo

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 10 |
| Test Priority | 3 |
| Test Title/Name | Validar chk\_diferentes\_usuarios en compartidos, chk\_archivos\_tamano etc. |
| Test Summary | Intentar insertar datos que rompan checks: compartir archivo a sí mismo, tamaño negativo. |
| Test Steps | 1. Insertar usuario y archivo 2. Intentar compartir archivo del usuario con él mismo 3. Intentar insertar archivo con tamaño = -10 |
| Test Data | Usuario: userchk@example.com  Archivo: archivo\_chk.txt |
| Expected Result | La inserción en compartidos falla por constraint chk\_diferentes\_usuarios.  Inserción de archivo con tamaño negativo falla por chk\_archivos\_tamano. |
| Actual Result | Funcionan correctamente las restricciones de diferentes usuarios y tamaño negativo. |
| Status | Funcional. |
| Pre-condition | Tener un usuario y un archivo que pertenezca a ese usuario. |
| Post-condition | No se deben crear registros inválidos. |
| Notes/Comments |  |

## Test case #11: Estructura de dirección

Hecho por: Daniel Jurado

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 11 |
| Test Priority | |  | | --- | |  |   3 |
| Test Title/Name | |  | | --- | |  |  |  | | --- | | Validación de estructura de dirección durante el registro | |
| Test Summary | Probar que el sistema no acepte direcciones en formato libre que no estén estructuradas ni normalizadas. |
| Test Steps | 1. Ir al formulario de registro de usuario.  2. Ingresar los datos personales válidos.  3. Ingresar dirección: Mi casa frente al parque al lado de la tienda.  4. Enviar el formulario. |
| Test Data | Dirección: Mi casa frente al parque al lado de la tienda |
| Expected Result | El sistema debería **rechazar** el registro con un mensaje: “Dirección inválida: seleccione una estructura válida compuesta por tipo de vía, número y barrio”. |
| Actual Result | El sistema **aceptó** la dirección sin validación y la guardó como texto plano. |
| Status | No funcional. |
| Pre-condition | El campo de dirección aún está como VARCHAR(255) en la tabla usuarios |
| Post-condition | El usuario quedó registrado con una dirección mal estructurada, difícil de consultar o validar. |
| Notes/Comments | No se encuentra |

## Test case #12: numero de telefono

Hecho por: Daniel Jurado

|  |  |
| --- | --- |
| Field | Description |
| Test Case # | 12 |
| Test Priority | 3 |
| Test Title/Name | Validación del número de teléfono en el formulario y script. |
| Test Summary | Verificar que el sistema impida registrar números de teléfono inválidos (no numéricos o de longitud incorrecta). |
| Test Steps | 1. Ir al formulario de registro. 2. Llenar los campos obligatorios. 3. Ingresar como teléfono: abc123defg. 4. Enviar el formulario. 2. Verificar de la misma manera insertando el mismo teléfono desde el motor. |
| Test Data | Teléfono: abc123defg |
| Expected Result | El sistema debería rechazar el número e indicar: “Teléfono inválido: debe contener solo números, empezar en 3 y tener 10 dígitos |
| Actual Result | El sistema **aceptó el número alfanumérico** y permitió el registro sin advertencia en el motor, en el backend si hay validación. |
| Status | No funcional. |
| Pre-condition | No existe validación SQL. |
| Post-condition | Usuario registrado con teléfono inválido, afecta futuras validaciones, contactos o integraciones externas. |
| Notes/Comments | Esto valida que el teléfono tenga exactamente 10 dígitos y comience en 3 |