## 1️⃣ App: `proyectos`

**Propósito:** Gestión de proyectos, miembros y metodologías aplicadas.

**Modelos:**

- `Proyecto` → nombre, descripción, propietario
- `MiembroProyecto` → usuario, proyecto, rol (DEV, QA, ADMIN, Invitado)
- `ProyectoMetodologia` → tabla intermedia Proyecto ↔️ Metodología
  Opcional:
- `Metodologia` (si quieres centralizar aquí en vez de app separada)
- Relaciones con `Sprint` o `Reporte`

---

## 2️⃣ App: `sprints`

**Propósito:** Gestión de sprints para proyectos ágiles.

**Modelos:**

- `Sprint` → nombre, descripción, fecha_inicio, fecha_fin, estado (Planificado, Activo, Finalizado), relación a `Proyecto`

Opcional:

- Relación con `Ticket` o `Bug` para asignar tareas al sprint

---

## 3️⃣ App: `tickets`

**Propósito:** Gestión de incidencias, tareas y bugs tipo Jira.

**Modelos:**

- `Ticket` → título, descripción, proyecto, sprint, autor, asignado_a, estado, prioridad
- `ComentarioTicket` → ticket, autor, contenido
- `HistorialTicket` → ticket, usuario, acción, descripción

---

## 4️⃣ App: `bugtracker`

**Propósito:** Seguimiento de bugs específicos y comentarios asociados.

**Modelos:**

- `Bug` → título, descripción, severidad, estado, proyecto, sprint, reportado_por, asignado_a
- `ComentarioBug` → bug, usuario, comentario

---

## 5️⃣ App: `testsuite`

**Propósito:** Gestión de casos de prueba, suites y ejecuciones.

**Modelos:**

- `TestSuite` → nombre, descripción, proyecto
- `CasoPrueba` → nombre, descripción, test_suite, estado
- `EjecucionPrueba` → caso_prueba, ejecutado_por, resultado, observaciones, version_app, entorno

---

## 6️⃣ App: `reports`

**Propósito:** Generación de reportes QA y de gestión.

**Modelos:**

- `Reporte` → título, descripción, tipo (BUGS, TESTS, GENERAL), proyecto, testsuite, metodologia, generado_por
- `ArchivoReporte` → reporte, archivo

---

## 7 App: `autenticacion`

**Propósito:** Usuarios, MFA, tokens de activación, auditoría de login.

**Modelos:**

- `Usuario` → nickname, correo, slug, intentos_fallidos, bloqueado_hasta, etc.
- `AutenticacionMultifactor` → usuario, código_verificación, fecha_expiracion, ip, user_agent
- `TokenActivacion` → usuario, token, fecha_expiracion, usado
- `Metadatos` → abstracto, con fecha_creacion, fecha_actualizacion, estado

---

## 8 App: `notificaciones`

**Propósito:** Alertas y notificaciones para usuarios.

**Modelos:**

- `Notificacion` → usuario, título, mensaje, tipo, leida
- `ConfiguracionNotificacion` → usuario, recibir_correos, recibir_alertas_app

---

## 9 App: `historial` (opcional, si quieres centralizar)

**Propósito:** Auditar acciones de usuarios sobre proyectos, tickets y bugs.

**Modelos:**

- `Historial` → usuario, acción, descripción, proyecto, ticket, bug
- `DetalleHistorial` → historial, campo, valor_anterior, valor_nuevo

---

## ✅ Resumen visual de apps y modelos

| App | Modelos principales |
|---|---|
| proyectos | Proyecto, MiembroProyecto, ProyectoMetodologia |
| sprints | Sprint |
| tickets | Ticket, ComentarioTicket, HistorialTicket |
| bugtracker | Bug, ComentarioBug |
| testsuite | TestSuite, CasoPrueba, EjecucionPrueba |
| reports | Reporte, ArchivoReporte |
| autenticacion | Usuario, AutenticacionMultifactor, TokenActivacion, Metadatos |
| notificaciones | Notificacion, ConfiguracionNotificacion |
| historial (opcional) | Historial, DetalleHistorial |
| metodologias (opcional) | Metodologia |

---

## 1 app_proyectos

**Descripción:**
Gestiona toda la información de proyectos, incluyendo su nombre, descripción, propietario, miembros y metodologías aplicadas. Permite organizar sprints y asignar tickets y bugs.
**Entidades principales:**

- `Proyecto`
- `MiembroProyecto`
- `Sprint`
- `ProyectoMetodologia`

---

## 2️⃣ app_tickets

**Descripción:**
Administra los tickets de tareas o incidencias dentro de los proyectos. Permite asignarlos a usuarios, registrar comentarios, historial y estados (Pendiente, En Progreso, En Pruebas, Finalizado).
**Entidades principales:**

- `Ticket`
- `ComentarioTicket`
- `HistorialTicket`

---

## 3️⃣ app_bugtracker

**Descripción:**
Gestión de bugs y errores detectados en los proyectos, incluyendo severidad, estado, comentarios y asignaciones. Integra seguimiento por sprints y vinculación con tickets.
**Entidades principales:**

- `Bug`
- `ComentarioBug`

---

## 4️⃣ app_testsuite

**Descripción:**
Permite crear y organizar **Test Suites** y **Casos de Prueba**, registrar ejecuciones de QA, resultados y observaciones. Incluye auditoría con versión de la app y entorno de prueba.
**Entidades principales:**

- `TestSuite`
- `CasoPrueba`
- `EjecucionPrueba`

---

## 5️⃣ app_reports

**Descripción:**
Genera reportes de QA y bugs, vinculados a proyectos, test suites o metodologías. Permite adjuntar archivos de evidencia y centralizar resultados de pruebas.
**Entidades principales:**

- `Reporte`
- `ArchivoReporte`

---

## 6️⃣ app_autenticacion

**Descripción:**
Gestiona usuarios, autenticación multifactor (MFA), tokens de activación, permisos y control de acceso al sistema.
**Entidades principales:**

- `Usuario`
- `AutenticacionMultifactor`
- `TokenActivacion`

---

## 7️⃣ app_notificaciones

**Descripción:**
Sistema de notificaciones para usuarios, con configuración personalizada (alertas en app y correo). Permite marcar como leídas, diferenciar tipo de notificación y almacenar historial.
**Entidades principales:**

- `Notificacion`
- `ConfiguracionNotificacion`

---

## 8️⃣ app_historial

**Descripción:**
Auditoría de acciones dentro del sistema. Permite registrar cambios en proyectos, tickets, bugs y ejecutar seguimiento de cada modificación con detalle de campo y valor anterior/nuevo.
**Entidades principales:**

- `Historial`
- `DetalleHistorial`

---

```
Error parsing Mermaid diagram!

Cannot read properties of null (reading 'getBoundingClientRect')
```

---

## ✅ Qué representa este mapa

1. **Usuarios centrales**: `app_autenticacion` conecta con casi todas las apps para asignación, ejecución y reportes.
2. **Proyectos como eje**: `app_proyectos` contiene Tickets, Bugs, TestSuites y Reportes.
3. **QA / Testeo**: `app_testsuite` genera ejecuciones que alimentan reportes (`app_reports`) y se registran en historial (`app_historial`).
4. **Alertas y notificaciones**: `app_notificaciones` recibe información de cambios en Tickets, Bugs y Ejecuciones.
5. **Auditoría completa**: `app_historial` registra cambios de todas las apps críticas para QA y gestión de proyectos.

---

## Diagrama de relaciones (conceptual)

```
[Usuario] ──<MiembroProyecto>──> [Proyecto] ──<Sprint>──> [Ticket]
     │                                │                 │
     │                                │                 └──<HistorialTicket>
     │                                │
     │                                └──<ProyectoMetodologia>──> [Metodologia]
     │
     ├──<AutenticacionMultifactor>
     ├──<TokenActivacion>
     ├──<Notificacion>
     └──<ConfiguracionNotificacion>


[Proyecto] ──<Bug>──> [ComentarioBug]
       │
       └──<Reporte> ──<ArchivoReporte>
       │                 │
       │                 └── (relacion opcional con TestSuite y Metodologia)
       │
       └──<TestSuite> ──<CasoPrueba> ──<EjecucionPrueba>
                            │
                            └── version_app, entorno, ejecutado_por (Usuario)


[Historial] ──<DetalleHistorial>
```

```
    ↑
    └── Registra acciones de Usuario sobre Proyecto, Ticket o Bug
```

---

## Explicación de relaciones

1. **Proyectos y usuarios**
   - `MiembroProyecto` conecta usuarios con proyectos y define su rol (DEV, QA, ADMIN).
   - `ProyectoMetodologia` indica qué metodologías se aplican en cada proyecto.
2. **Sprints y tareas**
   - Cada proyecto puede tener varios sprints (`Sprint`).
   - Tickets y Bugs pueden estar asociados a un sprint.
3. **Tickets y bugs**
   - Cada `Ticket` o `Bug` tiene comentarios y un historial de acciones (`HistorialTicket` o `ComentarioBug`).
   - `Historial` centraliza auditoría general para proyectos, tickets o bugs.
4. **QA y pruebas**
   - `TestSuite` pertenece a un proyecto.
   - Cada `TestSuite` tiene varios `CasoPrueba`.
   - Cada `CasoPrueba` puede tener varias `EjecucionPrueba`, donde se registra **usuario**, **resultado**, **version_app**, **entorno**, etc.
5. **Reportes**
   - `Reporte` puede vincularse a un proyecto, un TestSuite y opcionalmente a una metodología.
   - Los archivos adjuntos se guardan en `ArchivoReporte`.
6. **Usuarios y autenticación**
   - `Usuario` gestiona login, MFA y tokens de activación.
   - Las notificaciones se envían según la configuración del usuario.
7. **Historial**
   - Centraliza auditoría sobre acciones importantes de usuarios en proyectos, tickets y bugs.
   - `DetalleHistorial` permite ver **qué cambió exactamente** (valor anterior y nuevo).

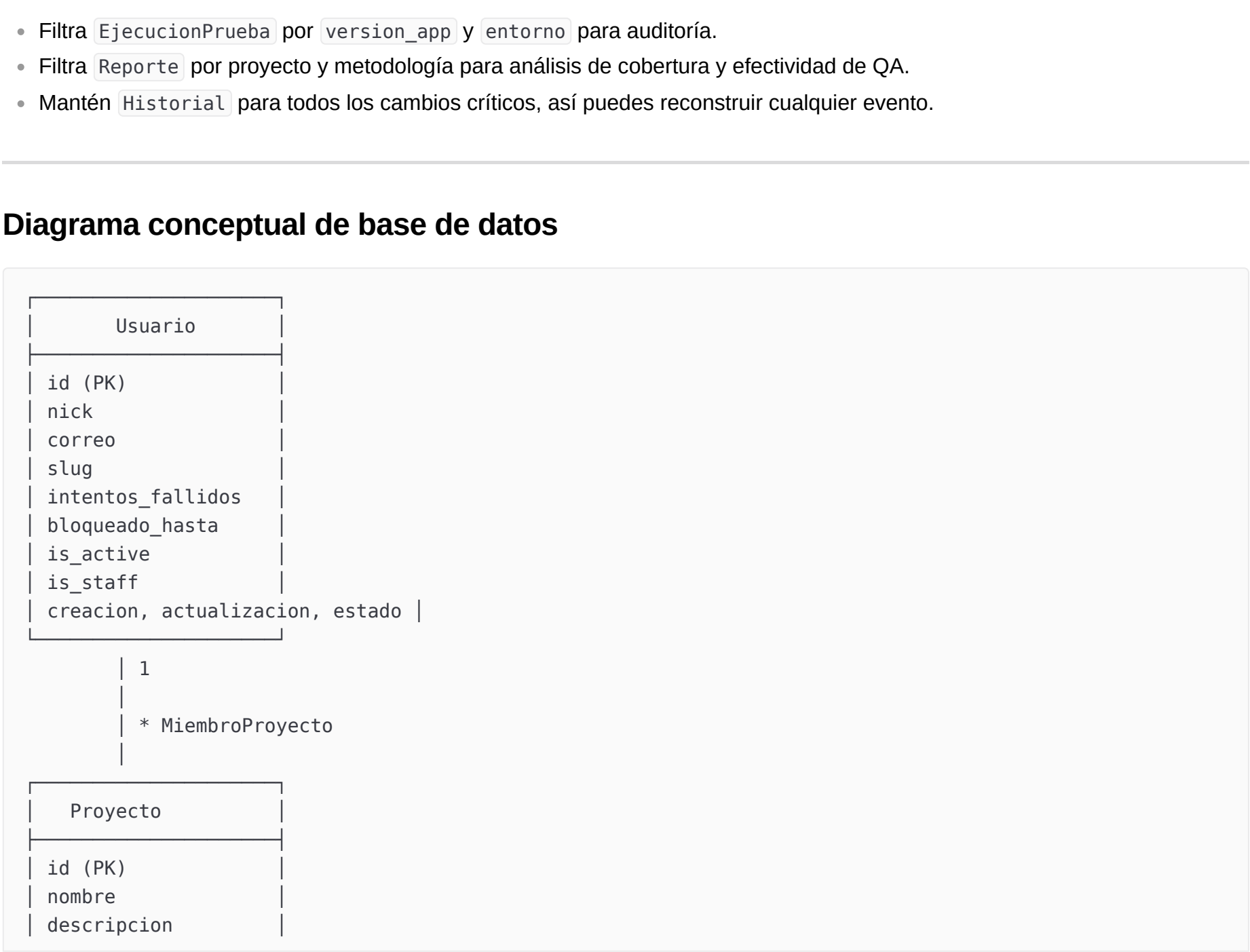
---

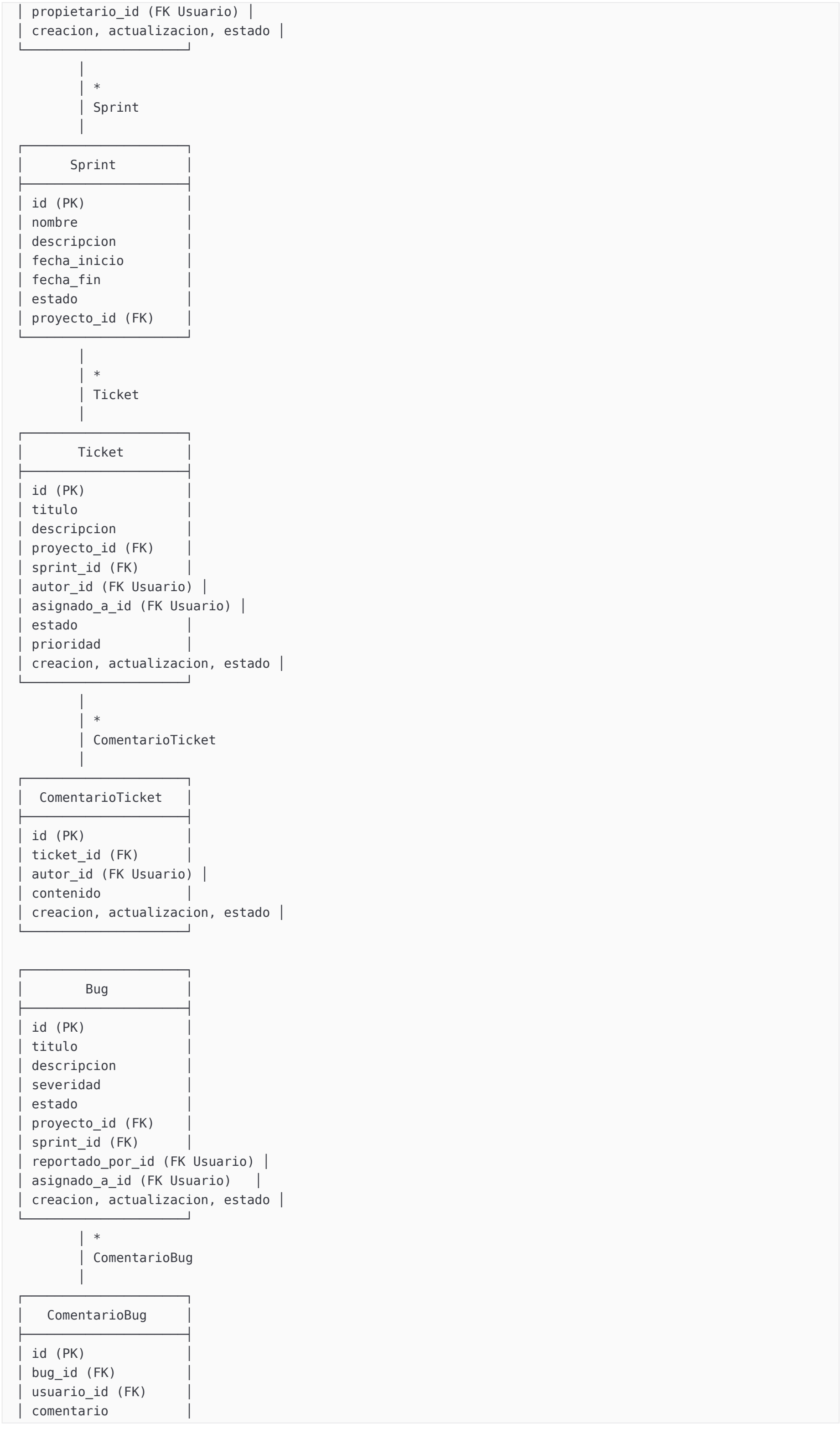
💡 **Sugerencia QA avanzada:**

- Filtra `EjecucionPrueba` por `version_app` y `entorno` para auditoría.
- Filtra `Reporte` por proyecto y metodología para análisis de cobertura y efectividad de QA.
- Mantén `Historial` para todos los cambios críticos, así puedes reconstruir cualquier evento.

---

## Diagrama conceptual de base de datos




propietario_id (FK Usuario)
creacion, actualizacion, estado
Sprint
Sprint
id (PK)
nombre
descripcion
fecha_inicio
fecha_fin
estado
proyecto_id (FK)
Ticket
Ticket
id (PK)
titulo
descripcion
proyecto_id (FK)
sprint_id (FK)
autor_id (FK Usuario)
asignado_a_id (FK Usuario)
estado
prioridad
creacion, actualizacion, estado
ComentarioTicket
ComentarioTicket
id (PK)
ticket_id (FK)
autor_id (FK Usuario)
contenido
creacion, actualizacion, estado
Bug
Bug
id (PK)
titulo
descripcion
severidad
estado
proyecto_id (FK)
sprint_id (FK)
reportado_por_id (FK Usuario)
asignado_a_id (FK Usuario)
creacion, actualizacion, estado
ComentarioBug
ComentarioBug
id (PK)
bug_id (FK)
usuario_id (FK)
comentario

```
| creacion, actualizacion, estado |
└──────────────────────────┘


┌──────────────────────────┐
│    TestSuite             │
├──────────────────────────┤
| id (PK)                  |
| nombre                   |
| descripcion              |
| proyecto_id (FK)         |
| creacion, actualizacion, estado |
└──────────────────────────┘
        | *
        | CasoPrueba
        |
┌──────────────────────────┐
│    CasoPrueba            │
├──────────────────────────┤
| id (PK)                  |
| nombre                   |
| descripcion              |
| estado                   |
| testsuite_id (FK)        |
| creacion, actualizacion, estado |
└──────────────────────────┘
        | *
        | EjecucionPrueba
        |
┌──────────────────────────┐
│  EjecucionPrueba         │
├──────────────────────────┤
| id (PK)                  |
| caso_prueba_id (FK)      |
| ejecutado_por_id (FK Usuario) |
| resultado                |
| observaciones            |
| version_app              |
| entorno                  |
| creacion, actualizacion, estado |
└──────────────────────────┘


┌──────────────────────────┐
│      Reporte             │
├──────────────────────────┤
| id (PK)                  |
| titulo                   |
| descripcion              |
| tipo                     |
| proyecto_id (FK)         |
| testsuite_id (FK)        |
| metodologia_id (FK)      |
| generado_por_id (FK Usuario) |
| creacion, actualizacion, estado |
└──────────────────────────┘
        | *
        | ArchivoReporte
        |
┌──────────────────────────┐
│  ArchivoReporte          │
├──────────────────────────┤
| id (PK)                  |
| reporte_id (FK)          |
| archivo                  |
| creacion, actualizacion, estado |
└──────────────────────────┘


┌──────────────────────────┐
│   Metodologia            │
├──────────────────────────┤
| id (PK)                  |
| nombre                   |
| tipo (Ágil/Tradicional/Híbrida) |
| creacion, actualizacion, estado |
└──────────────────────────┘
```


---

## Notas importantes

1. `version_app` y `entorno` van en `EjecucionPrueba` para auditoría QA.
2. `Metodologia` se puede vincular a proyectos y reportes.
3. `ProyectoMetodologia` permite **muchos a muchos** y registrar detalles de cómo se aplica.
4. `Historial` y `DetalleHistorial` centralizan auditoría de todo: cambios en tickets, bugs y proyectos.
5. Todas las entidades heredan `Metadatos` (`creacion`, `actualizacion`, `estado`) para trazabilidad.

---

```
Error parsing Mermaid diagram!

Cannot read properties of null (reading 'getBBox')
```

---

## ✅ Qué incluye este diagrama

1. **Usuarios y su autenticación:** MFA, token de activación, notificaciones y configuración.
2. **Proyectos y metodologías:** Varios proyectos pueden aplicar varias metodologías (`ProyectoMetodologia`).
3. **Sprints, Tickets y Bugs:** Gestión tipo Jira/Trello.
4. **QA y pruebas:** TestSuites, Casos de prueba, Ejecuciones con `version_app` y `entorno`.
5. **Reportes y archivos adjuntos:** Vinculados a proyectos, TestSuite y metodologías.
6. **Historial y auditoría:** Historial centralizado con detalle de cambios.

---

```
Error parsing Mermaid diagram!

Cannot read properties of null (reading 'getBoundingClientRect')
```

---

## ✅ Qué muestra este tablero visual

1. **Tickets por estado:** Pendiente → En Progreso → En Pruebas → Finalizado.
2. **Bugs por estado:** Abierto → En Progreso → Resuelto → Cerrado.
3. **Casos de prueba y ejecuciones:** Cada ejecución tiene `version_app` y `entorno`.
4. **Reportes:** Vinculados a Ejecuciones de prueba o Bugs.
5. **Sprints y Metodologías:** Sprint 1 → XP/Ágil, Sprint 2 → Waterfall/Tradicional.
6. **Usuarios asignados:** QA y DEV conectados a Tickets, Bugs y Ejecuciones.

---

💡 **Sugerencia de mejora:**

- Podrías agregar **columnas de prioridad** (Baja, Media, Alta, Crítica) para resaltar tickets críticos.
- También se podría mostrar **historial de cambios** como pequeños íconos debajo de cada ticket/big bug.

---

```
Error parsing Mermaid diagram!

Cannot read properties of null (reading 'getBoundingClientRect')
```

---

## ✅ Qué representa este Kanban

1. **Columnas por estado:** Pendiente, En Progreso, En Pruebas, Finalizado.
2. **Tickets:** Prioridad, asignado a un usuario.
3. **Bugs:** Severidad y asignado.
4. **Casos de Prueba:** Estado, versión de la app (`version_app`), entorno, ejecutado por QA.
5. **Relaciones entre entidades:** Tickets relacionados con Bugs, Casos de Prueba vinculados a Tickets para reportes QA.

---

💡 **Ideas para mejorar:**

- Agregar **iconos de prioridad** y severidad para visualización rápida.
- Diferenciar **entornos** con colores: `Staging`, `QA`, `Producción`.
- Incluir **historial resumido** de cada ticket o bug con mini-tags (cambios, comentarios, ejecuciones).
- Integrar un **tooltip** con más detalles al pasar el cursor (si se hace en frontend).