Quick Justice

Owner	Reviewer

Summary:

Actualmente el estudio de abogados Mendoza depende de Excel, Access y servidores físicos para gestionar sus casos, pagos y documentos. Nuestro equipo a manera de solución y como propuesta tiene planeado reemplazar esas herramientas por una arquitectura escalable basada en una API optimizada, migrando a PostgreSQL para gestionar y almacenar la data.

Requirements:

- Tablas en Access: clientes, casos, pagos, planilla
- Migrar base de datos de Access a Postgresql
- Cortar excel del diseño para que no reciba nuevos datos de casos.
- Escalar de manera vertical el mismo espacio de trabajo.
- Indexar según los campos de búsqueda más frecuentes.
- Mejorar el endpoint de búsqueda de casos GET/cases/search, añadir el de registro de casos (POST/cases) y gestión de pagos POST/payments.
- Permitir ingresos para registro de vouchers físicos, mediante escaneo y reconocimiento de datos.

Out of Scope:

- Servicios en la nube por altos costos.
- Crecimiento horizontal, mejora del hardware actual.
- el reconocimiento de datos del voucher

Keywords:

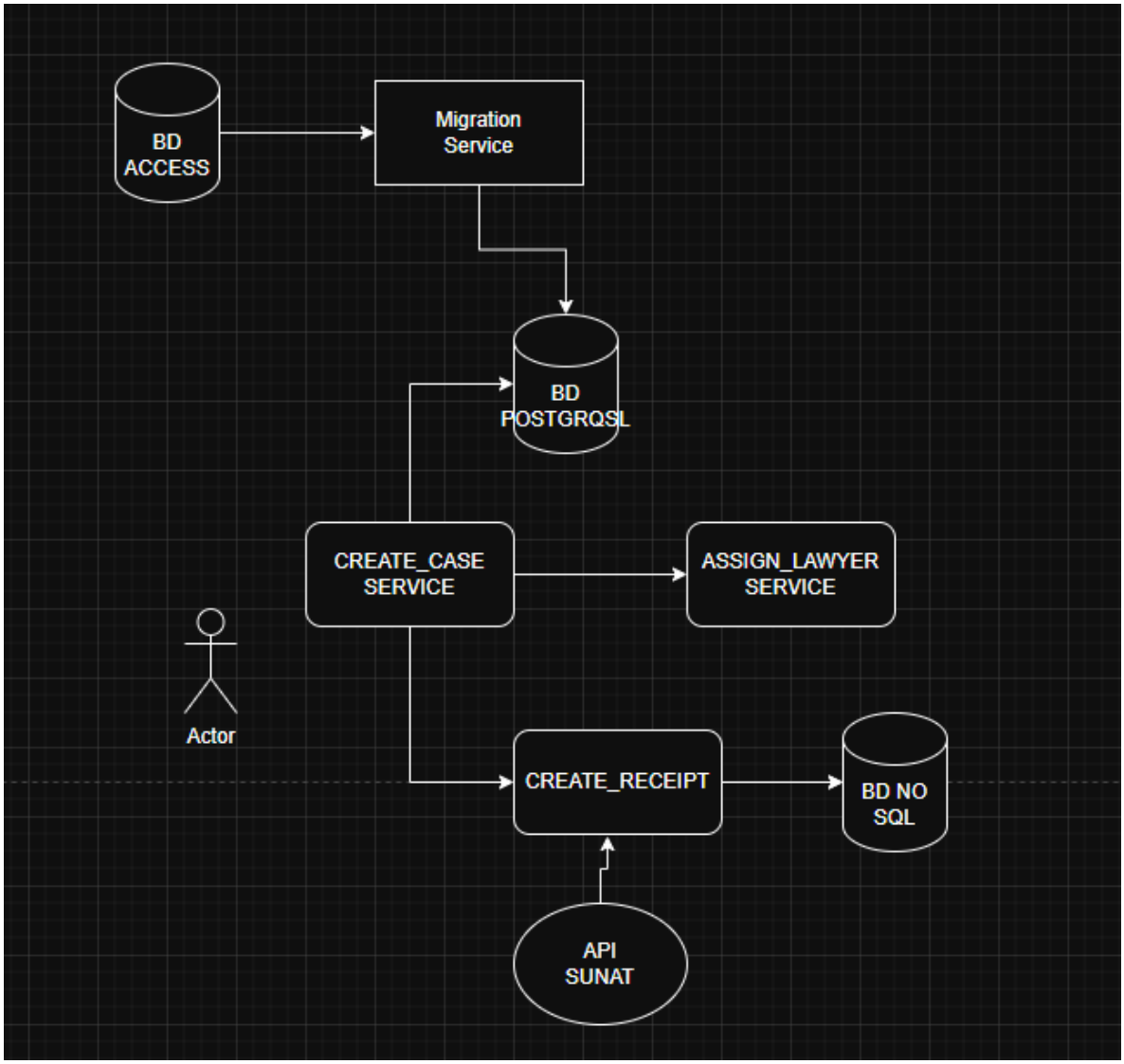
- Migración de datos: proceso realizado para transferir la información almacenada en Excel y Access hacía PostgreSQL con el uso de scripts locales, python y pandas.
- PostgreSQL: reemplaza a Access como sistema de base de datos relacional ofreciendo eficiencia y soporte para índices y consultas más rápidas y complejas (Bajo costo ya que funciona en una misma máquina local)

- API REST: la creación de esta api permitirá registrar casos, buscar información y gestionar pagos.
- Caché: con ello almacenaremos temporalmente resultados de búsqueda más frecuentes.
- Almacenamiento Local: sustituye el servidor físico y los servicios en la nube.
- Bajo costo: sin servicios en la nube y con una infraestructura local. Además, se prioriza la simplicidad reduciendo gastos operativos ya mencionados.

Current status:

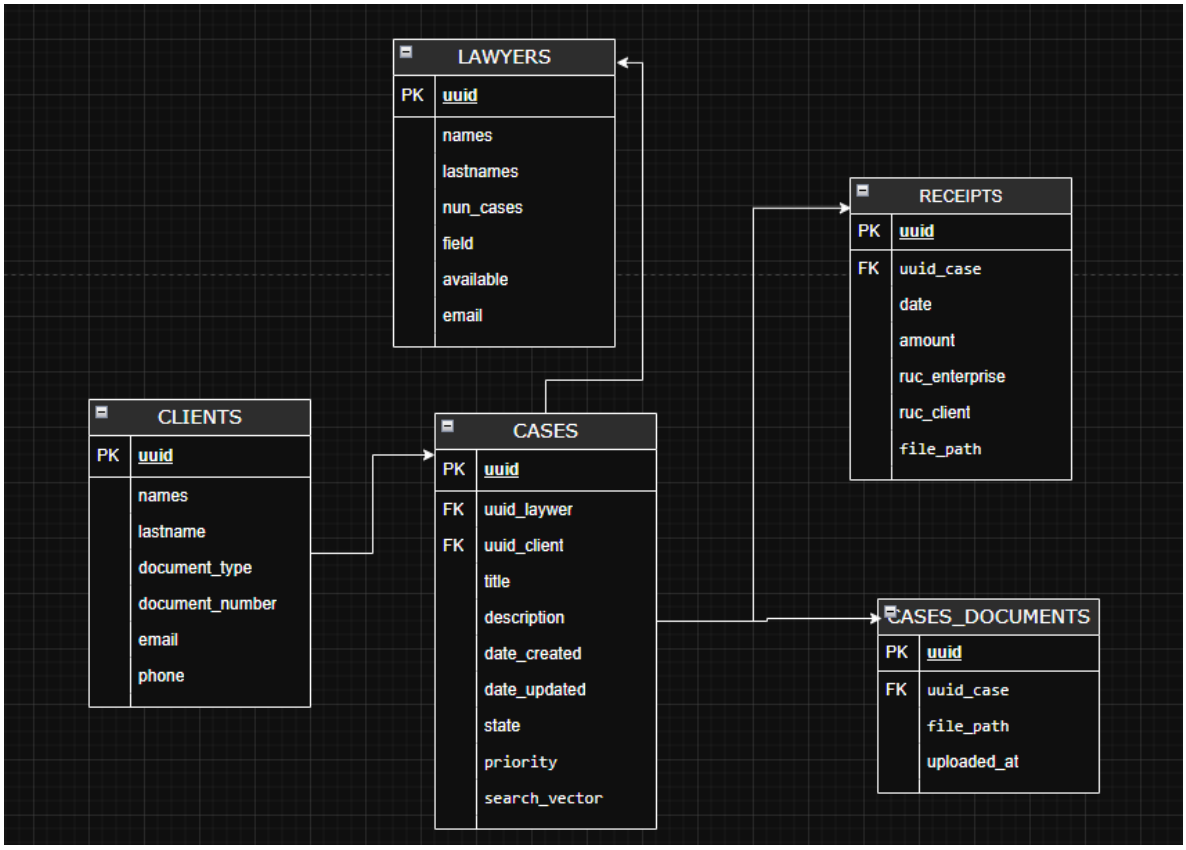
Un nuevo caso es guardado en excel, luego es segmentado y almacenado en ACCESS. De la misma manera los recibos, que son alojados id, path en access.

Technical proposal:



Database design proposal:

- /search GET
- /lawyer GET, PUT, DELETE, POST
- /lawyer/assing {uuid_cliente} POST
- /cases/{casesId} GET PUT
- /cases/search GET
- /cases/{casesId}/attachments POST GET
- /receipt POST
- /receipt {receiptId} GET
- /clients POST
- /clients/search GET
- /clients/ {clientId} GET



Additional information:

- el search va a usar un ts vector de postgresql para optimización en búsqueda.
- orden de migracion:
    - Primero migrar los CLIENTES y ABOGADOS, generando UUIDs.
    - Luego migrar los CASOS, validando que cliente y abogado existan.
    - Migrar RECIBOS. Si no se pueden vincular a un CASE, se deja uuid_case nulo o se asocia por lógica (fecha cercana, cliente).
    - Crear un script para mover archivos y renombrarlos por uuid_case.
- tabla de receipts indexada por hash en el uuid