UNIVERSIDAD HISPANOAMERICANA

ESCUELA DE INGENIERÍA

ARQUITECTURA DE DATOS

SISTEMA DE VOTO ELECTRÓNICO

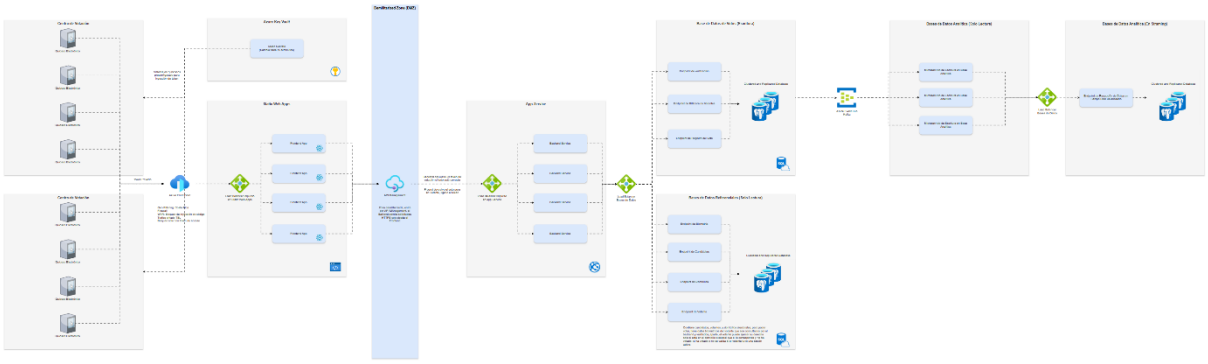
KENNY RICARDO RAMOS WONG

SAN JOSÉ, JULIO 2025

Contents

Esquema de aplicación de votos

Diagrama de aplicación



Fuente: Elaboración propia

Funcionamiento de la aplicación

La aplicación consiste en un frontend desplegado en el servicio de Azure Static Web Apps, este servicio ofrece nativamente un balanceador de carga auto escalable. Utilizar esta herramienta nos da una alta resistencia a fallos con un SLA brindado por Azure superior al 99%, indirectamente, al estar alojado en Azure, nos permite eventualmente aislar al backend en una zona desmilitarizada.

Primero centrémonos en los quioscos, estos quiscos son preprogramados con un script que les permite actualizar su token mediante consultas constantes al administrador de secretos de Azure, Key Vault, este gestor de secretos actualiza el token de autenticación al sitio web cada 15 minutos. Cada vez que el token es renovado inmediatamente el quisco recibe el token, mismo que es necesario para poder acceder al frontend.

Por medio del header el quisco envía el token nuestro frontend, pero este, tiene un intermediario, que es otro servicio de CDN llamado Azure Front Door, no aplicamos bloqueo por IP porque su administración ería muy complicada, en cambio, usamos el token que solo los quiscos reciben cada 15 minutos.

Además de brindarnos esta capa de seguridad por token, el servicio también nos permite aplicar políticas WAF, *"Las políticas WAF pueden impedir que ataques como inyección SQL, secuencias de comandos entre sitios y otros ataques HTTP o HTTPS con firmas conocidas accedan a sus sistemas. Los conjuntos de reglas administradas en los niveles premium protegen contra*

muchos ataques comunes y bots. También puede restringir el acceso a sus sitios a regiones o direcciones IP específicas." (Nash, 2023). Lo cual es indispensable para un proceso electoral propenso a ataques.

Una vez autenticado el quiosco en el frontend entrando por el CDN, este hará un proceso de verificación del votante, se leerán datos biométricos, que pueden ser firma digital, huella dactilar o el código único de la cédula, el frontend enviará una solicitud de validación del votante al backend, el backend opera en una zona desmilitarizada, accesible únicamente desde el frontend a través del servicio de API Management.

Este servicio, nos permite poner el backend privado, solo accesible para el servicio que nosotros queramos, el sistema está limitado para que solo el propio frontend pueda realizar solicitudes al backend.

El backend está alojado en App Service de Azure, escrito en Python y funciona como un orquestador que puede consultar a las bases de datos o escribir, todo mediante Endpoints habilitados en ambas bases de datos.

Tenemos dos bases de datos, una de lectura y otra de escritura, la de lectura contiene los candidatos, votantes, domicilios electorales, mientras que la de escritura una bitácora de votantes que ya han votado, votos y logs para auditoría. Ambas bases de datos son transaccionales, están geo-replicadas, así como trabajan en clúster para balancear la carga, ningún servicio consulta a otro sin ingresar por un balanceador de carga.

Finalmente, tenemos otro servicio basado en Kafka, Azure Event Hub, que es un streaming de los votos y la bitácora de votos, esto es enviado a la base de datos analítica, ingresando por microservicios que gestionan el envío de estos datos. No se envían mas datos ya que los votantes, domicilios electorales y candidatos son conjuntos de datos fijos que no varían durante la elección, por lo tanto, solo se envían los votos para tener un conteo en tiempo real.

Volviendo al backend, este es el orquestador central, que recibe consultas del frontend, como por ejemplo verificar si una persona ya ha votado, consulta a la bitácora en la base de datos de escritura y devuelve una respuesta, también escribe los votos en la misma base, así como verifica datos biométricos del votante y si su domicilio electoral es correcto. Una medida de seguridad implementada es que el frontend no abre la sesión de votación hasta que no valida estas tres

variables, ya ha votado, es válido el votante y coincide con el domicilio del quiosco, las respuestas a estas interrogantes las responde realizando las consultas a nuestro backend.

Por otro lado, el voto es anónimo, el sistema lo que hace es que una vez validadas las 3 interrogantes el backend genera un token único habilitado solo pro 5 minutos, este va firmado digitalmente con RSA, que para dar mayor contexto *"El cifrado RSA consta de una **clave pública**, a la que se puede acceder libremente, y de una **clave privada**, que preferiblemente solo debe conocer una persona. El cifrado original se realiza con la clave pública RSA. En cambio, para descifrar se necesita la clave RSA privada. Si por alguna razón se pierde o no se conoce la clave privada, es prácticamente imposible descifrar un archivo, por ejemplo."* (IONOS, 2022). Esto con fines de trazabilidad y auditoría, permitiéndonos tener un control de validación de votos sin violar el derecho al voto secreto.

Una vez ejercido el votante selecciona sus candidatos, se abre una vista previa con un botón de confirmar votación, al momento que el votante confirma su voto el frontend envía dos respuestas al backend, la primera, un mensaje confirmando que el votante ha ejercido su derecho, marcándolo en la bitácora de votos sin revelar nada de por quién votó, únicamente captura la hora y el nombre del votante, impidiendo así un sufragio doble ya que esto es un requisito previo para abrir una sesión de votación. Al mismo tiempo devuelve el token de votación con el candidato elegido, escribiéndolo así en la base de datos.

Es importante aclarar que son dos servicios separados, únicamente registra el hecho de que esa persona ha votado (sin registrar por quién), y se almacena de forma separada a los votos, de ahí que existan dos Endpoints de escritura.

Todo esto, como se mencionó, está con un streaming en tiempo real a una base de datos analítica que nos brinda el recuento de los votos inmediato, la información se escribe en una capa bronce de datos que eventualmente es procesada en silver y escrita en un repositorio de datos curado en gold que nos brinda los datos relacionados que nos permiten determinar el ganador de las elecciones según el recuento de votos.

Flujo de validación para votar

El votante llega al quiosco e inicia su sesión.

El frontend (a través del backend) consulta la base de lectura:

- Valida datos biométricos.
- Verifica que esté habilitado para votar.
- Verifica que esté en el domicilio electoral correcto.

Si todo es correcto:

- Se permite votar. Abre una sesión de votación.
- El frontend muestra candidatos válidos.
- El votante realiza su selección (Aunque también permite votos nulos)
- Votante confirma y se le despliega una prevista de su voto con un botón de confirmar.
- Al confirmar, se registra el voto y cierra sesión.

Cuando el votante confirma su voto:

- Se registra el voto en la base de datos de escritura.
- Se marca al votante como "ya votó".

Si el votante intenta volver a iniciar sesión:

- El backend consulta y detecta que ya votó, y lo rechaza.

Flujo para registro del voto

El votante ya está autenticado.

- Fue verificado en el flujo anterior.
- Abre una sesión de votación.

El backend genera un token de votación para el frontend.

- Fecha y hora
- UUID (Identificador único de voto.
- Firma digital (RSA, para auditorias, el sistema lo utiliza para verificar tokens validos generados solo por el backend).

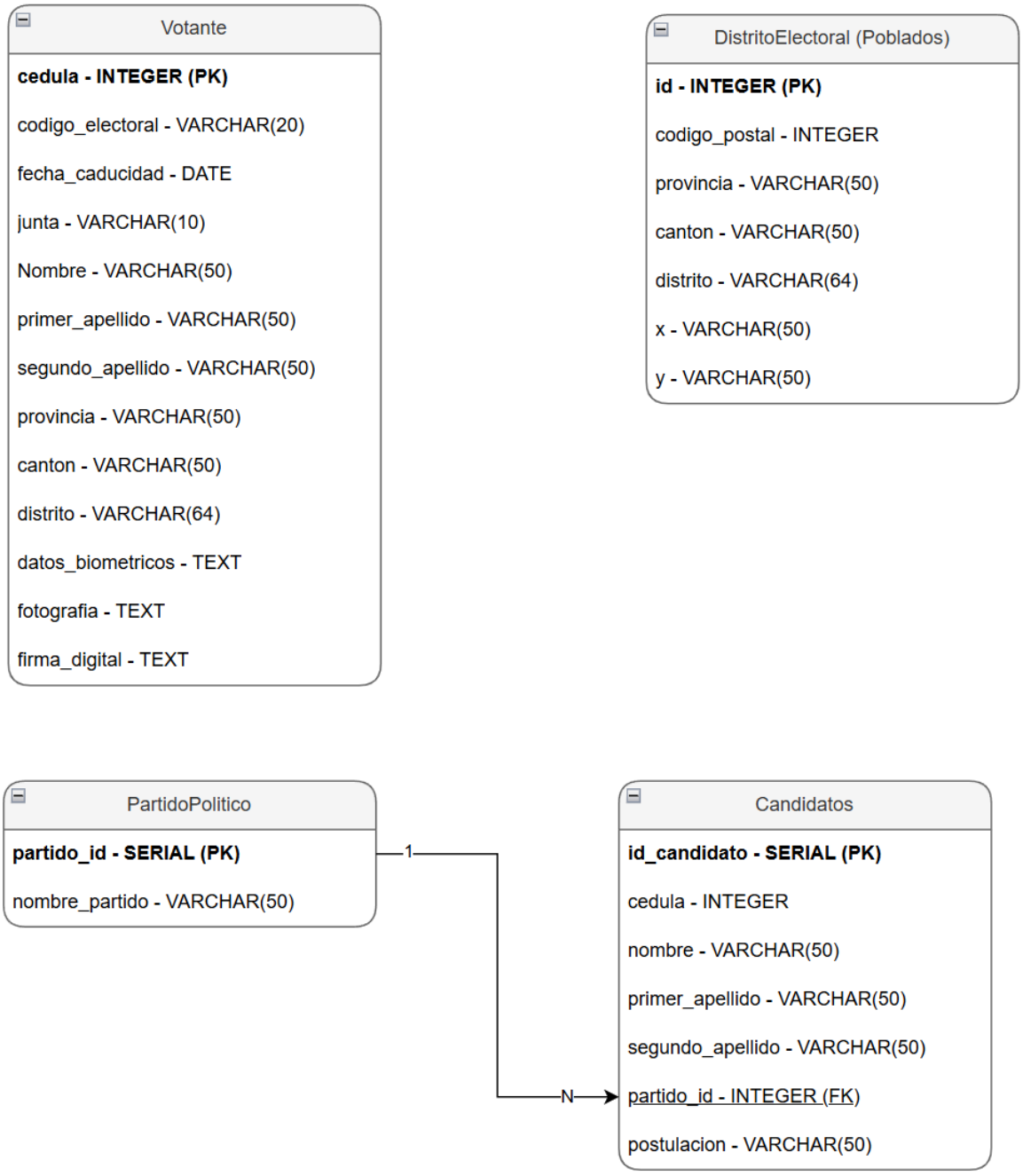
El frontend recibe el token.

El frontend verifica que la firma sea válida.

Esto sucede por detrás cuando el usuario se autentica, creando un token de votación firmado que es el que eventualmente se registra en la base de datos cuando el votante ejerce su derecho.

Bases de Datos

Base de datos de Lectura

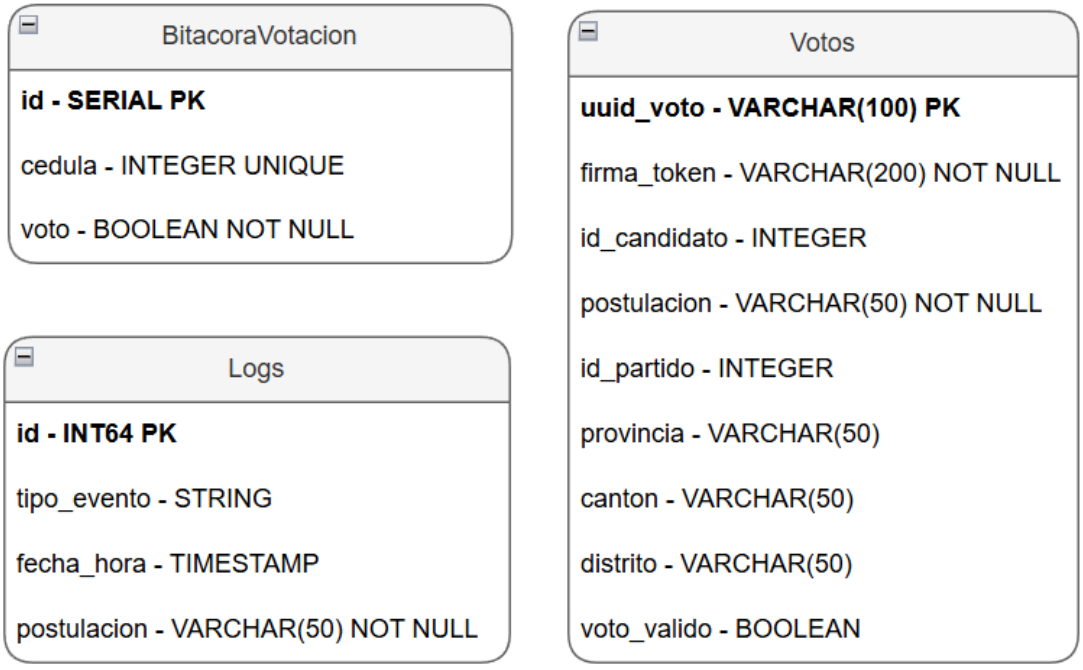


Fuente: Elaboración propia

La base de datos de lectura fue construida a partir de la información brindada por el profesor, por un lado, tenemos una lista de poblados, tenía un elemento llamado *object_id*, sin embargo, se determina utilizar el *distrito_id* (Se renombra a *código_postal*), esto para tener una llave primaria única, el JOIN dado que no hay llave se realiza por provincia, cantón y distrito al conjunto de datos brindado llamado **Votante**, del cual si poseemos más datos.

En el caso de los candidatos y partidos políticos, aquí no hay conjunto de datos suministrado, por lo tanto, se construye uno simple para el partido político con su respectiva llave a los candidatos que los conforman.

Base de datos de escritura



Fuente: Elaboración propia

Para la base de datos de escritura, aquí no hay relaciones, ye esto está bien, cada objeto o tabla es documentado por los procesos que realiza el backend en paralelo, pero sin violentar el derecho al voto secreto. Para ello, tenemos 3 elementos, la bitácora que captura las personas que ya votaron, sin indicar hora ni candidato por el que votó, logs para auditoria, donde el sistema

indica cada vez que se efectúa un voto, con su timestamp y ID, pero también captura errores internos o logs informativos, finalmente la tabla que contiene el voto anónimo.

Para la tabla votos, cada voto es un UUID generado por el backend cuando el votante se autentica, aparte de esto, también incluye una firma, que solo el backend con su clave privada puede autenticar la clave pública y verificar si es un voto valido o un ataque de inserción de SQL (Previamente controlado también con la política WAF), esto con el fin de tener varias capas de seguridad.

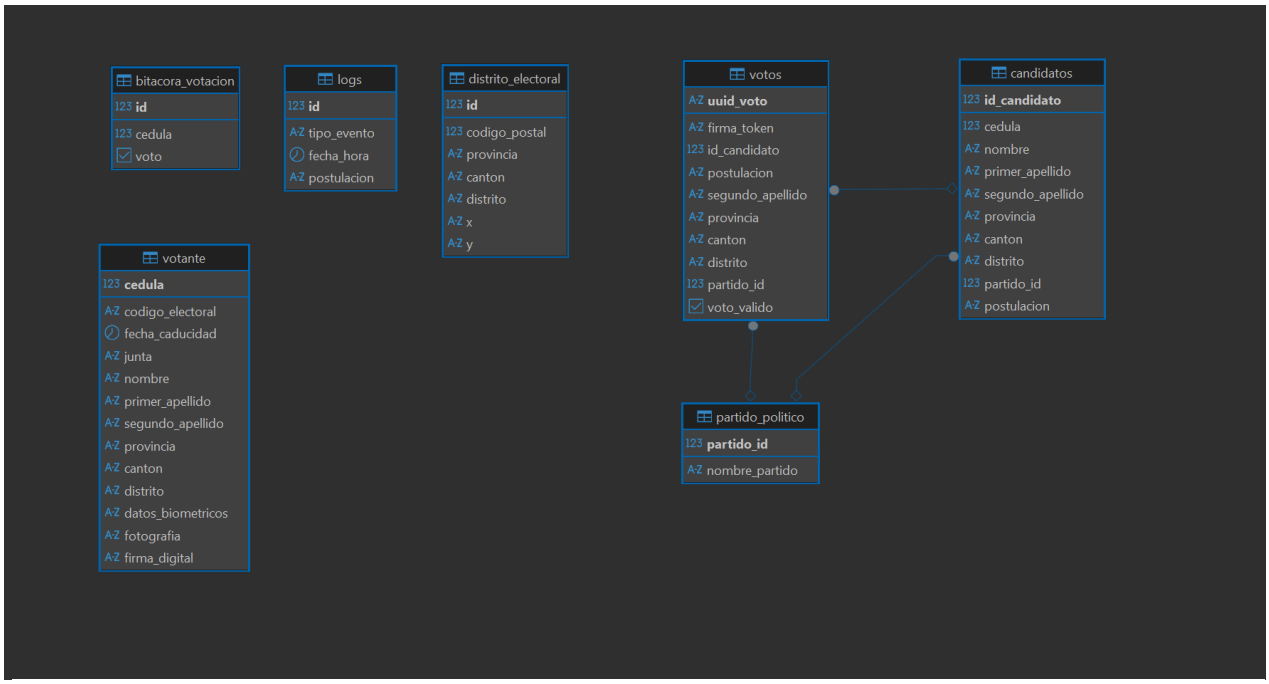
Aparte de la validación, se registra por quién se votó, el partido, y la postulación, esto porque recordemos que el sistema debe estar preparado tanto para diputados como presidencia, y en Costa Rica se puede hacer doble postulación, aparte, el quisco tiene los datos geográficos, así que dado que el backend los orquesta y tiene en memoria los escribimos para facilitar el eventual análisis, finalmente si el voto no es en blanco o nulo es false, mientras que si es true es valido, así podemos contar fácilmente cuantos votos tiene cada candidato a la presidencia y cada partido a los escaños de los diputados.

Aquí es importante algo, aunque el votante realiza solo un voto, son dos registros que se escriben, uno para diputados y otro para la presidencia, ambos se escriben con sus respectivos parámetros de seguridad en la base de datos.

Base de datos Analítica

Para la base de datos OLAP se construyó el modelo Medallón, en la capa bronce nos trajimos toda la información que pueda ser necesaria para análisis, el esquema quedó de la siguiente forma

Bronze Layer



Fuente: Elaboración propia

Donde, los datos de logs no se moverán a Silver ya que están reservados para auditoria, además, no se definió una relación con la bitácora de votación para realizar JOINs manuales en Silver que permitan calcular el porcentaje de abstencionismo, de igual forma, esto busca que el voto sea siempre anónimo.

Por otro lado, si existen relaciones entre los votos, candidatos y partido político con el fin de verificar el ganador de las elecciones. Esta información se modela en la capa Silver por medio de vistas.

En la capa Silver planteamos dos vistas, una que une la bitácora en contraste con el total de votantes, esto para verificar el porcentaje de abstencionismo, y otra que tiene todo el detalle del voto, para determinar el partido y el ganador de la presidencia, así como definir los curules para los diputados.

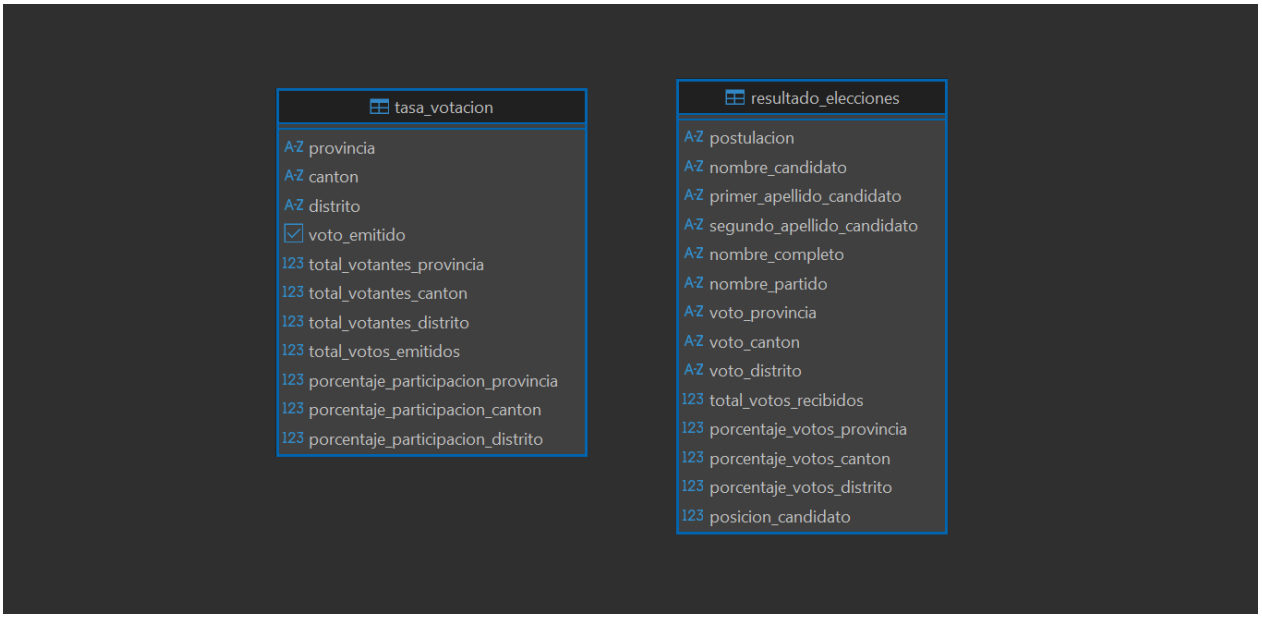
Silver Layer

Fuente: Elaboración propia

Finalmente, en nuestra capa Gold, realizamos calculos que nos permiten determinar el porcentaje de abstencionismo, así como definir quien es el ganador de las elecciones, ver el porcentaje de cada candidato así como de los diputados.

Aquí se utiliza una arquitectura relativamente moderna, descartando el modelo Snowflake y Star, utilizando en su lugar el modelo OBT (One Big Table) aunque en nuestro pequeño escenario no veremos esa "Big Table", para tener un poco mas de contexto "*OBT es una técnica de modelado que almacena todos los atributos de datos necesarios para el análisis en una tabla amplia y desnormalizada. A diferencia de los modelos de datos tradicionales, como el esquema en estrella y el esquema de copo de nieve, OBT elimina la necesidad de unir datos entre tablas de dimensiones y de hechos."* (Medium, 2023) Esto nos permitira optimizar nuestras consultas y realizar una conexión a herramientas de BI de forma mas sencilla.

Gold Layer



Fuente: Elaboración propia

References

IONOS. (3 de January de 2022). ¿Cómo funcionan las claves RSA? *IONOS*. Obtenido de https://www.ionos.com/es-us/digitalguide/servidores/seguridad/claves-rsa/

Medium. (14 de December de 2023). What is One Big Table? Exploring When to Use it, When Not to Use it and My Personal Experience. *Medium*. Obtenido de https://gbamezai.medium.com/what-is-one-big-table-exploring-when-to-use-it-when-not-to-use-it-and-my-personal-experience-3f740f56529d

Nash, M. (7 de September de 2023). Knock Knock; Who's there? *Medium*. Obtenido de https://medium.com/contino-engineering/knock-knock-whos-there-azure-front-door-920cffad2e3a