Instituto Tecnológico de Costa Rica
Área Académica de Administración de Tecnologías de Información
Curso: Bases de Datos Avanzados
Profesor: María José Artavia Jiménez
Grupo: 01

FarmaTEC

Estudiantes:
José Pablo Esquivel Morales
Randy Alejandro Martínez Sandí
Luis José Martínez Ramírez
Gustavo Alonso Fallas Carrera

Cartago, 26 de septiembre

Tabla de Contenidos

Resumen

El presente documento es una ficha técnica del proyecto programado llamado FarmaTEC, el cual consiste en desarrollar un sistema de bases distribuido (fragmentación y replicación). El tema es la una compañía farmacéutica que tiene tres puntos de ventas en tres provincias de Costa Rica: Heredia, Cartago y San José. La primera es nodo central que contiene toda la información y réplica extractos a los dos dos nodos. Cartago y San José son secundarios y contiene una fragmentación de los datos de Heredia. Logrando un sistema balanceado, el programa es robusto para dar respuesta a las consultas necesarias y saber soportar caídas o pérdidas de alguno de los nodos secundarios.

1. Arquitectura General

El programa FarmaTEC está compuesto por tres capas a gran escala: el Cliente, la Aplicación y los Datos o Persistencia. El Cliente es punto de entrada donde interactúan los usuarios con el programa. La Aplicación es la fracción donde se maneja las solicitudes, conexiones y controles. Los Datos es donde guarda toda la información que se consume. En la Figura 1 se muestra un diagrama de arquitectura que lo visualiza.

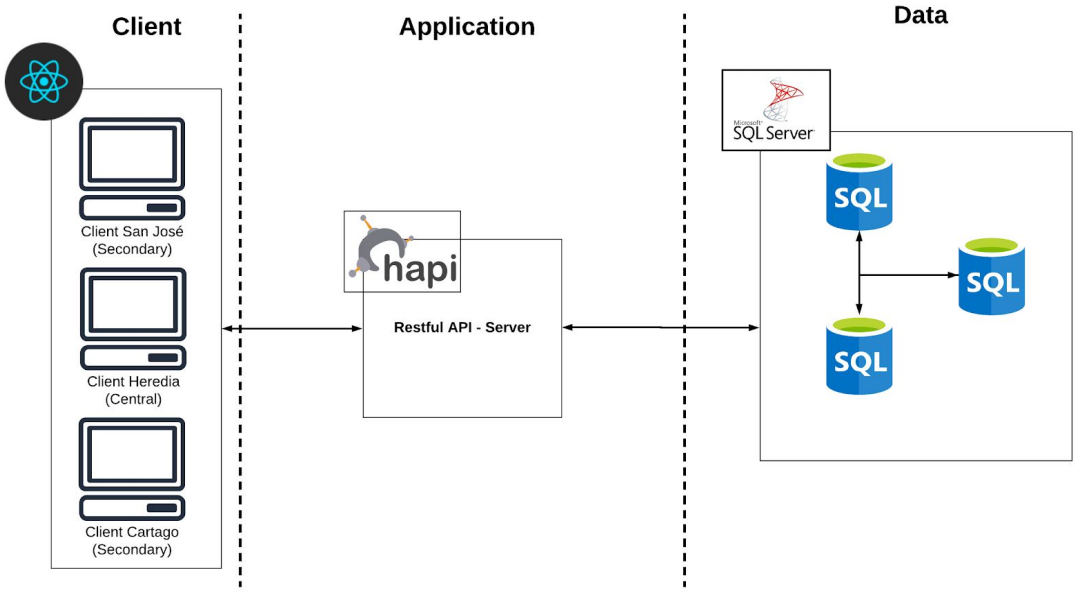

Figura 1. Diagrama de Arquitectura de FarmaTEC.

Como se puede observar anteriormente, el proyecto está compuesto por tres capas. El Cliente está construido usando la biblioteca de Interfaz de Usuario (*Front-End*) de JavaScript llamada ReactJS. La Aplicación es un servidor en NodeJS usando la biblioteca de HapiJS, este se comporta como un Restful API para dirigir las peticiones del Cliente a hacia los Datos (*Back-End*). Por último las Bases de Datos están basadas en el motor relacional de Microsoft SQL Server, MSSQL. En las siguientes secciones se entra en detalle de cómo está compuesta cada sección.

2. Base de Datos

2.1. Herramientas y Equipo

Para el manejo de los datos se utilizó Microsoft SQL Server 2017 v 14.0, y se corrieron en Windows 10 Pro y Windows 8.1. El sistema operativo es indiferente siempre y cuando al versión de la instancia de MSSQL sea la misma.

2.2. Diagrama ER Centralizada

El diseño central de la base de datos de forma centralizada se realizó en dos versiones: con y sin normalizar. La versión normalizada es la interpretación de cómo sería la base en tercera forma normalizada en caso de fuese un único servidor centralizado. Esta es meramente teórica y representativa, no se llegó a implementar en el proyecto.

La versión sin normalizar es aquella que sí se utilizó y se aplicó en el proyecto. El motivo fue poder trabajar con un diseño más sencillo para realizar las fragmentaciones y distribuciones respectivas. Este diseño es mucho más pequeño y sencillo, tiene repetición de algunos datos puesto a que no está pensada para ser eficiente, sino poner en práctica la división y asignaciones de datos en distintos nodos. Asimismo como la replicación de los datos en los nodos secundarios al nodo principal. A continuación se muestran los diagramas de entidad relación respectivos a cada tipo.

2.2.1. Normalizada

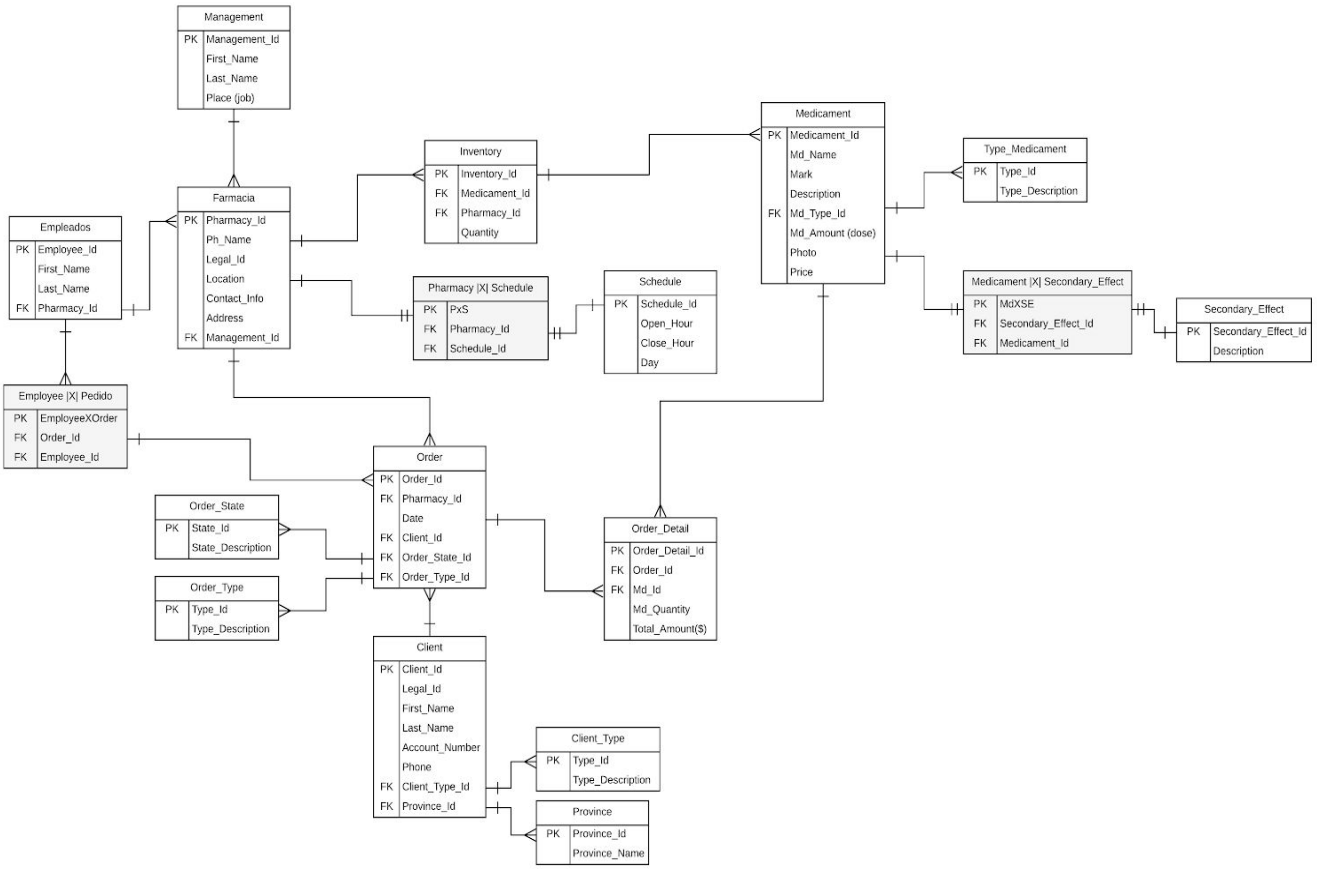

Figura 2. Diagrama de Entidad Relación de la base de datos en su tercera forma normalizada.

2.2.2. Sin Normalizar

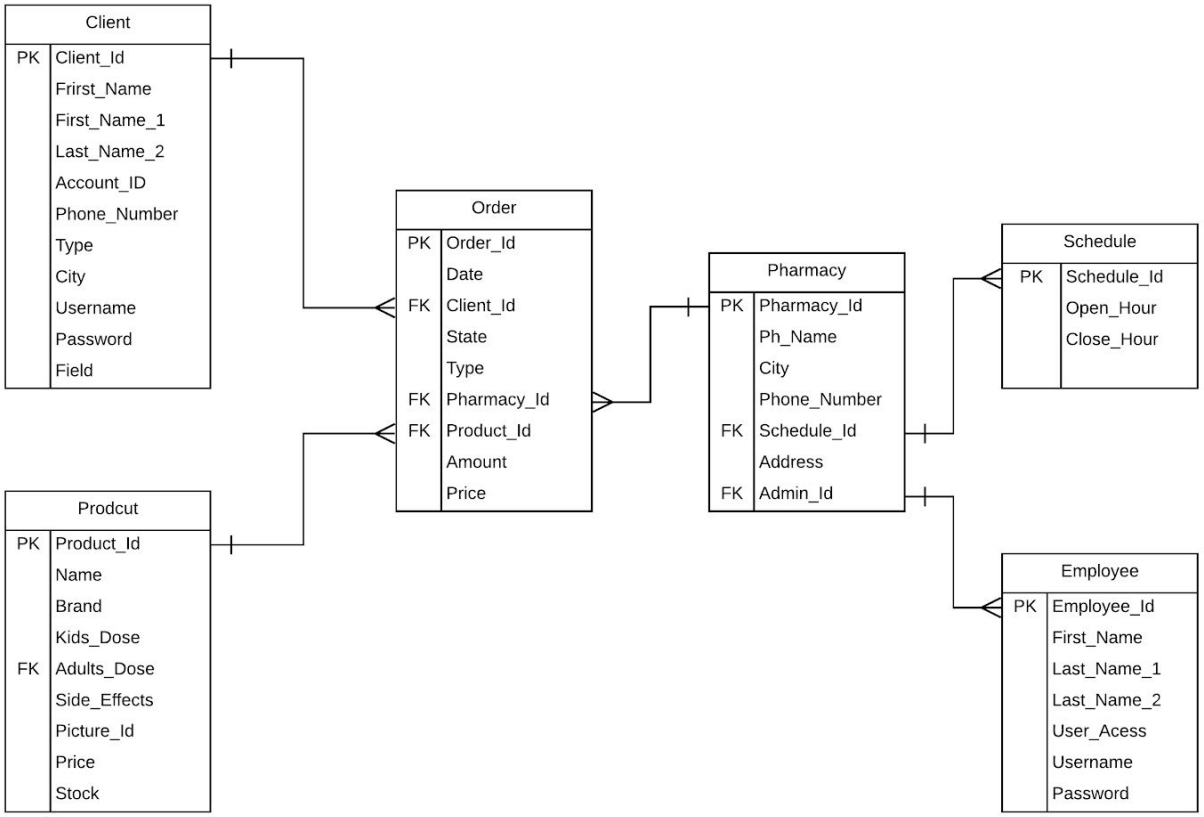

Figura 3. Diagrama de Entidad Relación de la base de datos en su tercera forma normalizada.

2.3. Fragmentación

Utilizando el diagrama de la Figura 3, ver página 5, se creó la fragmentación según cada nodo. Como la información tiene dos nodos secundarios que manejan sólo la información pertinente a esos lugares, la fragmentación tomó como guía los identificadores a cada lugar respectivo. El nodo central queda con toda la información disponible debido a que representan las oficinas centrales y tienen acceso a todo total.

- **Fragmentación Cartago:**

  Para el fragmentación de Cartago se hicieron los siguientes predicados.

1. $Pharmacy_{Cartago}$: PharmacyId = 'x';
2. $Schedule_{Cartago}$: Schedule ⋈ $Pharmacy_{Cartago}$;
3. $Order_{Cartago}$: Order ⋈ $Pharmacy_{Cartago}$;
4. $Client_{Cartago}$: Client ⋈ $Order_{Cartago}$

Tomando el "id" de la farmacia que corresponde a Cartago, se pueden obtener los datos que se asocian a ese lugar, tales como el horario, las órdenes, acá se efectúa una fragmentación primaria, ya que se buscan sólo el identificador dónde hay coincidencia. Los clientes tienen una fragmentación derivada debido a que para obtenr los clientes que hayan efectuado alguna actividad en la farmacia de Cartago, se debe buscar en la tabla de $Order_{Cartago}$ debido a ahí se asocian el identificador clientes.

- **Fragmentación San José:**

Para el fragmentación de San José se hicieron los siguientes predicados.

5. $Pharmacy_{SanJose}$: PharmacyId = 'y';
6. $Schedule_{SanJose}$: Schedule ⋈ $Pharmacy_{SanJose}$;
7. $Order_{SanJose}$: Order ⋈ $Pharmacy_{SanJose}$;
8. $Client_{SanJose}$: Client ⋈ $Order_{SanJose}$

En el caso de San José sucede lo mismo, la con diferencia de que acá se toman el identificador de esa farmacia, el resto de los pasos se repiten.

- **Fragmentación Heredia:**

El nodo de Heredia, es el nodo principal y acá están la información de todos los nodos secundarios en adición a la información que le corresponde a la farmacia de Heredia. Por lo tanto acá hay una replicación de los datos y los registros de Heredia están apilados junto con las fragmentaciones de Cartago y San José.

2.4. Asignación y Distribución

La información de cada fragmentación se encuentra ubicada en la base de que corresponde a cada nodo. Es decir, cada nodo tiene un propio cliente, aplicación y datos. Es decir, La información de propia de Cartago está ubicada en el nodo de Cartago y en el nodo de Heredia (por ser nodo principal). Asimismo sucede en San José. La Figura 4 muestra esto a manera de diagrama.

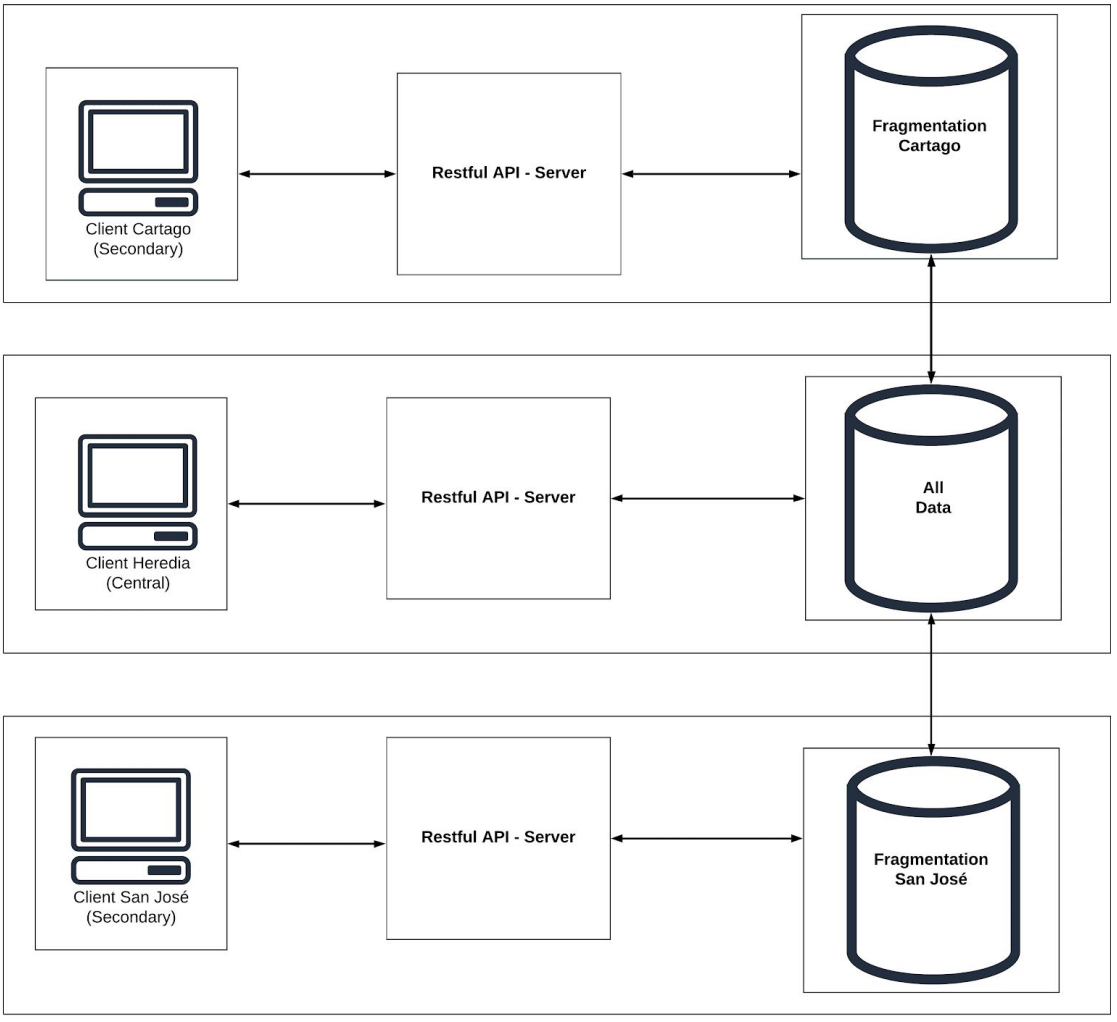

Figura 3. Diagrama general de la distribución de los datos en FarmaTEC.

2.5. Capturas de Pantalla Ejemplares

A continuación se muestran algunas capturas de pantalla de los pasos efectuados a la hora de realizar la replicación, fragmentación y distribución de la base de datos. Utilizando el diseño mostrado en en la Figura 3 (página 5). Esto con el propósito de ilustrar cómo se configuró lo explicado con anterioridad.

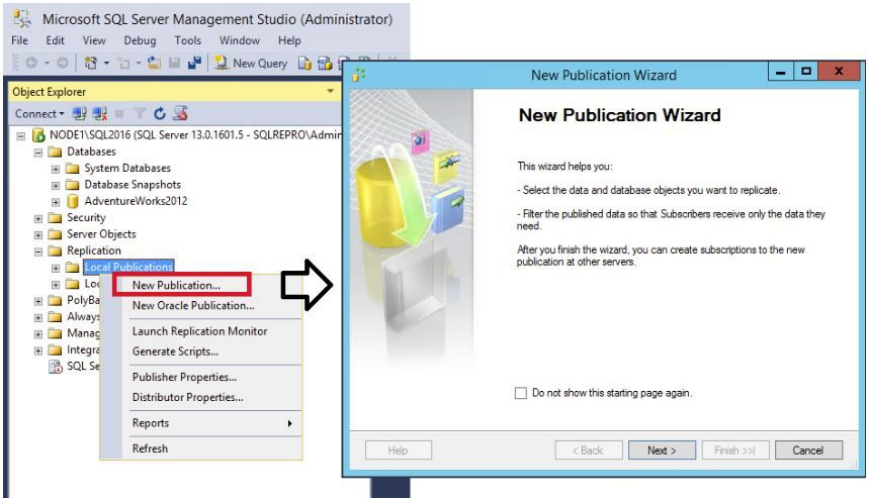

Figura 4. Crear una subscripción sobre una base de datos ya existente.

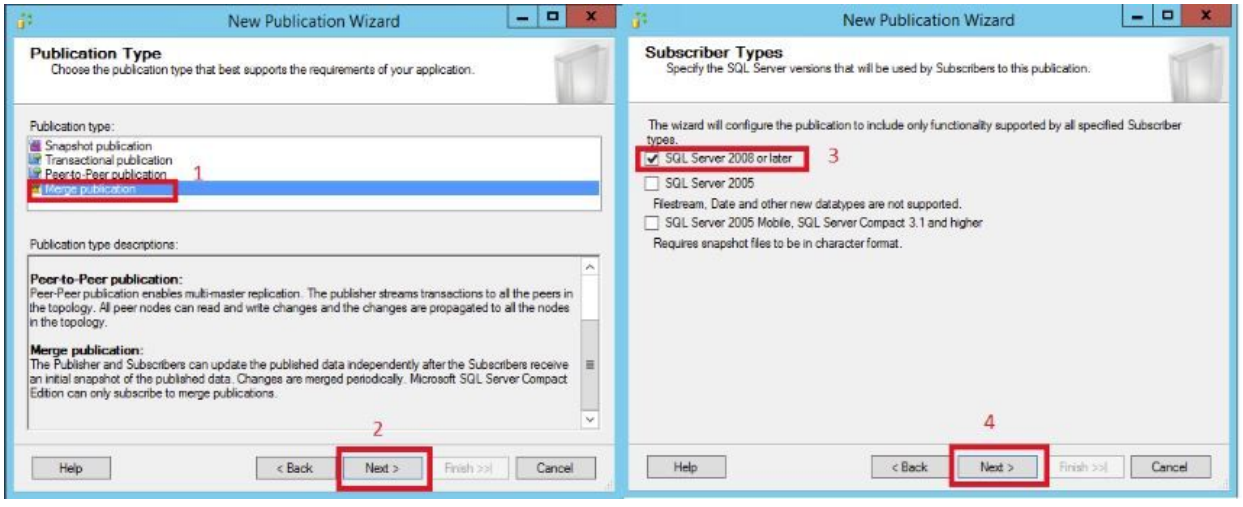

Figura 5. Selección de la configuración de la subscripción.

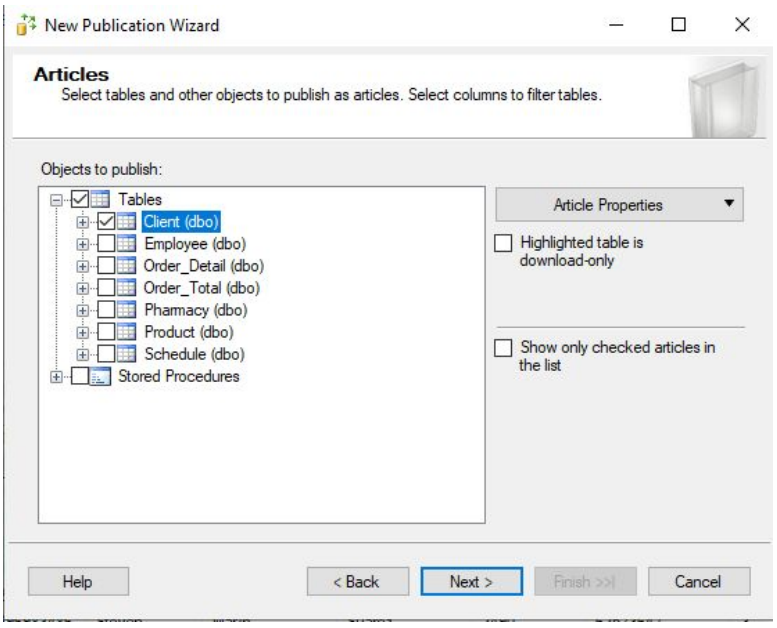

Figura 5 Selección de las tablas de la base de datos.

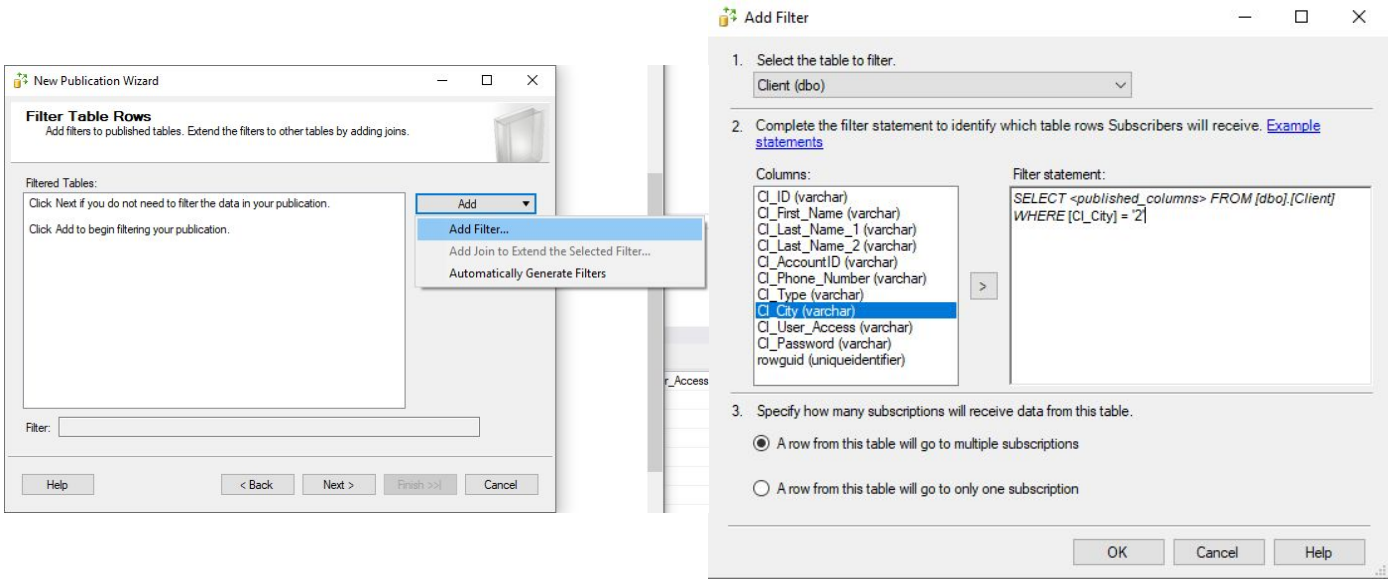

Figura 6. Selección de los filtros para la fragmentación.

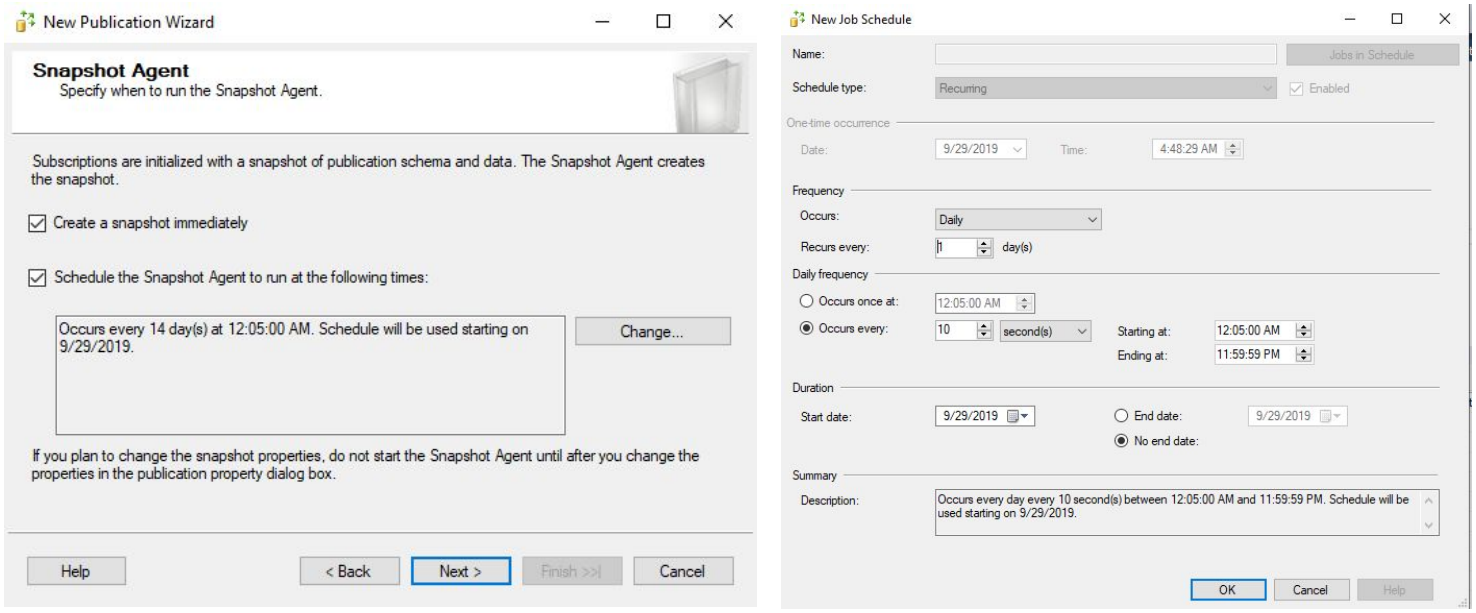

Figura 7.

Programación de los *snapshots* a ejecutar para los nodos secundarios (San José y Cartago).

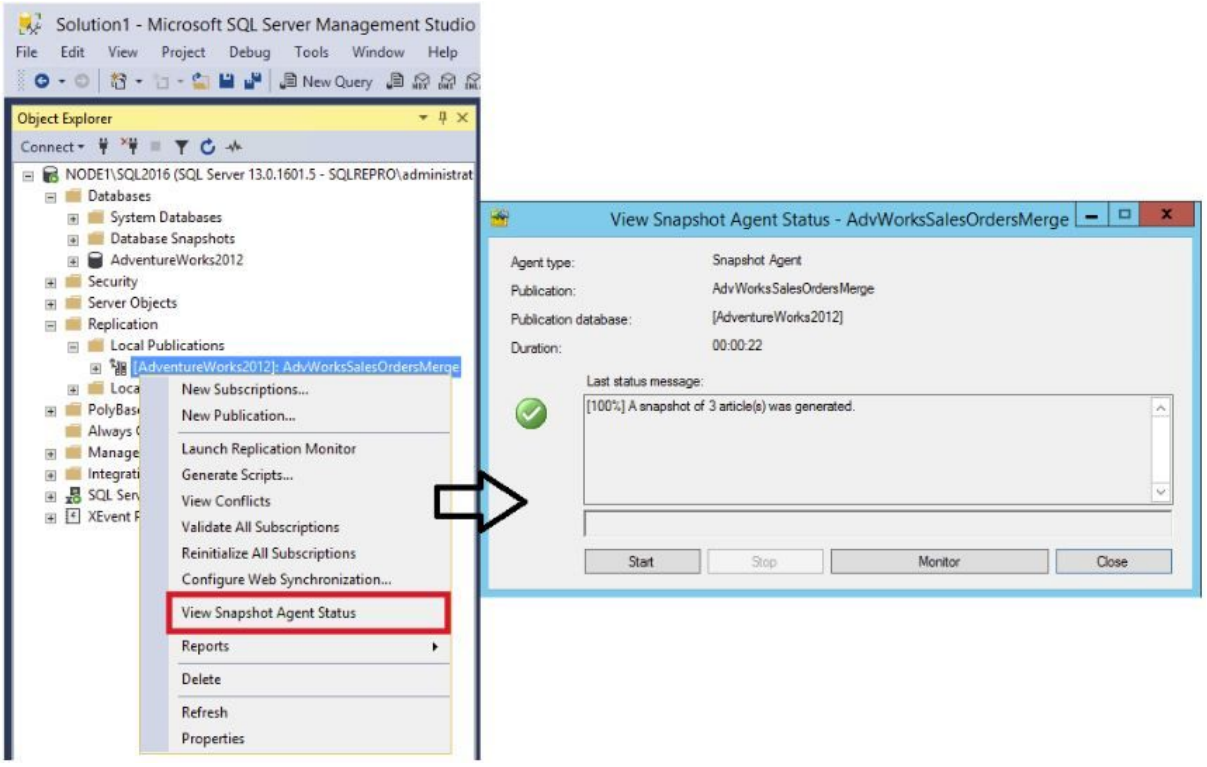

Figura 8. *Snapshot* realizado y listo para distribuirse..

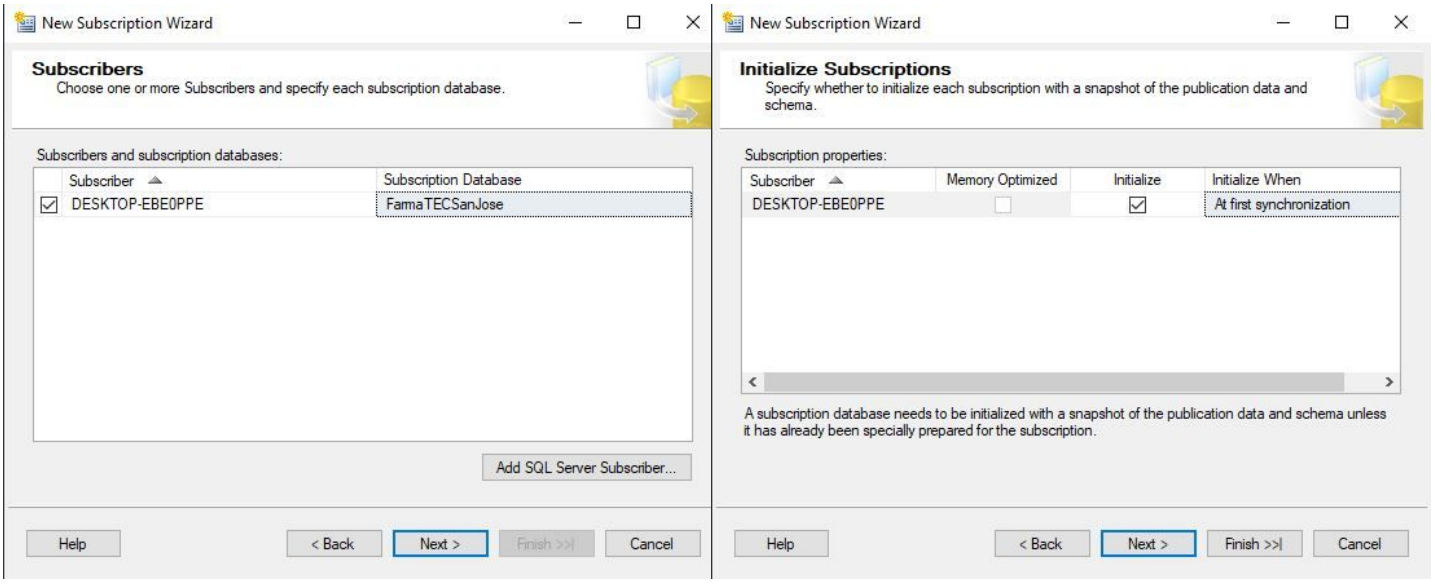

Figura 9. Del lado del suscriptor creación de la suscripción al nodo principal.

3. Cliente

3.1. Herramientas y Equipo

El Cliente es la sección donde el usuario interactúa con el sistema, visto en la Figura 1, página 2. Para desarrollarlo se utilizó ReactJS v16.9, usando NodeJS v10.16 y NPM v6.10. Al una aplicación web, un página dinámica, corre en todo los principales navegadores (Mozilla Firefox, Google Chrome y Apple Safari) y todo computador sin importar el sistema operativo.

3.2. Funcionamiento

El Cliente está compuesto por una serie componentes usando la premisa nativa de ReactJS, un conjunto de componentes trabajando juntos que reaccionan según cada evento, haciendo el código reutilizable en múltiples partes.

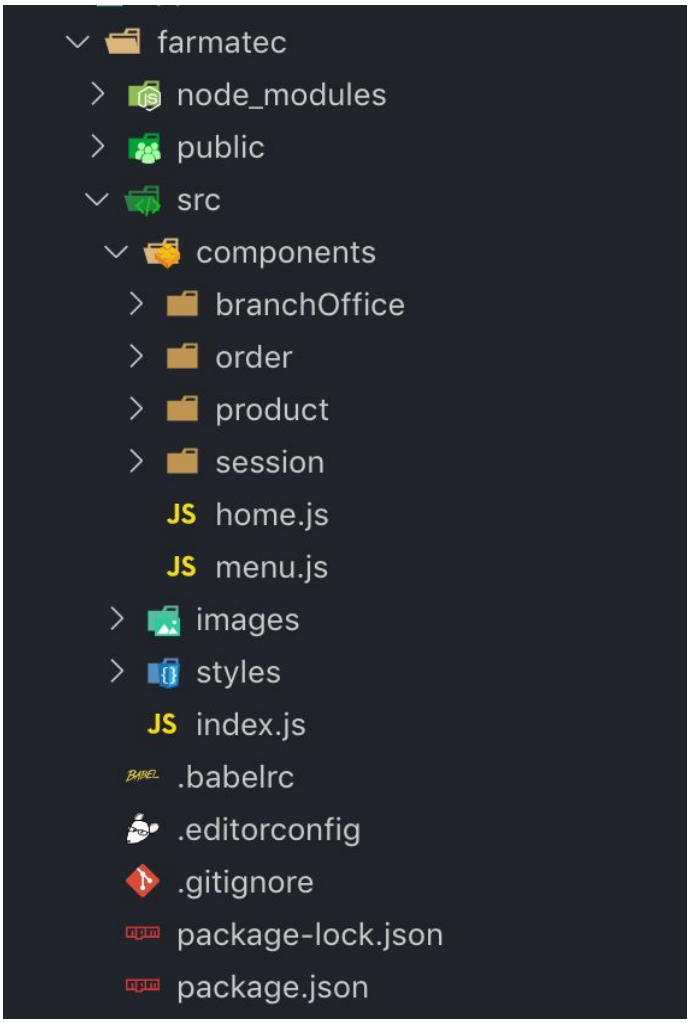

Figura 10. Anatomía de la solución del cliente usando ReactJS.

La carpeta de *node_modules* son las dependencias propias del proyecto instaladas y usadas por medio de NPM y NodeJS. La carpeta *src* es donde está el código: las vistas de las páginas corresponden a los componentes que ya se mencionaron. El resto de archivos son meramente de configuración y de mantener corriendo la aplicación. Es importante notar que se requiere conocimiento en JavaScript ya que estas son bibliotecas basada en este lenguaje.

4. Aplicación

4.1. Herramientas y Equipo

La Aplicación es la sección que actúa como intermediaria entre el Cliente y los Datos, como está ilustrado en cada en la Figura 1 en la página 2. Es un servidor RESTful API para poder conectar y. realizar las consultas hacia la base de datos. Está construido usando HapiJS v18.1, que es una biblioteca de NodeJS y NPM, utilizando las mismas versiones que se usaron para el Cliente.

4.2. Funcionamiento

El RESTful API es un servidor que redirige las consultas a la base de datos, utilizando métodos de HTTPS, se hacen las consultas del CRUD por medio de las acciones POST, GET, PUT, DELETE, UPDATE. Asimismo, en caso de que la conexión se pierda debido a algún fallo de la base de datos, éste redirige al nodo central. Para el este caso el nodo principal no presenta fallas por lo cual siempre está corriendo.

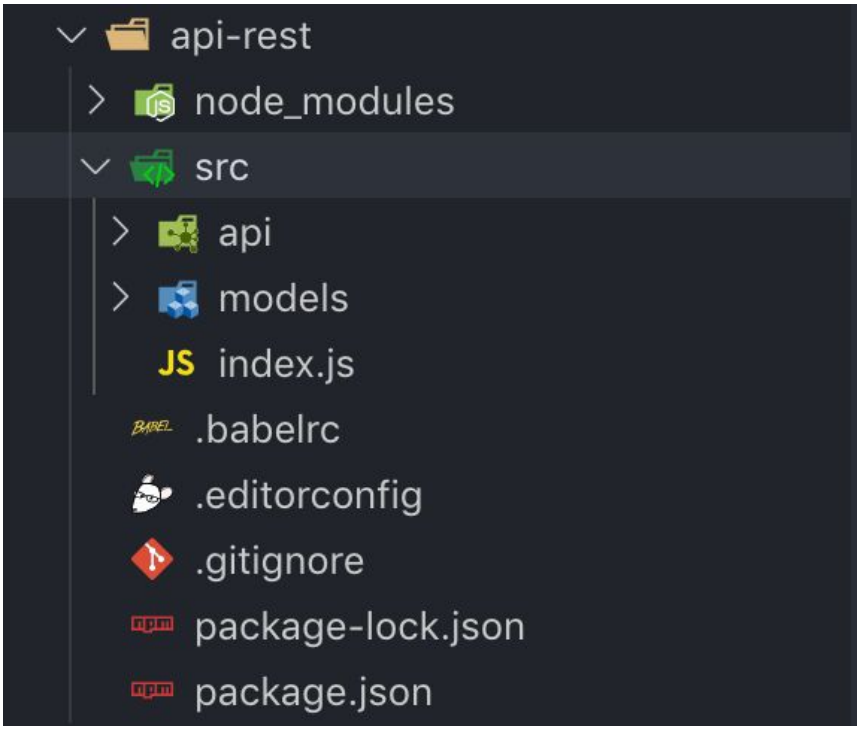

Figura 11. Anatomía de la solución del servidor usando HapiJS.

Como se puede observar, la estructura del servidor es muy parecida al del Cliente puesto que están construidos usando soluciones de NodeJS y NPM. La diferencia radica en que en *src* hay dos carpetas *models* y *api*. La primera presenta la conexión y la reconexión a las bases de datos iniciales y de recuperación en caso de fallo. La segunda carpeta, tiene una subcarpeta *v1* que contiene los las rutas y *paths* para efectuar las consultas respectivas.

5. Conclusiones

En términos generales el proyecto es una escala pequeña y sencilla. La parte de la base de datos consistió más en configuración. Hay diferentes maneras de realizar la replicación y fragmentación, todas al final llevan al mismo resultado quizá con distintas opciones.. Es importante probar, y así como se hizo, velar por las alternativas y ver cual es la mejor adapta a lo que se requiere. No obstante, siempre hay errores en los pasos, equivocaciones en los nombres, uso de redes distintas son algunos ejemplos.

El diseño de la base tomó su tiempo debido al cuidado de que las tablas que existieran fueran las necesarias, que no faltaran para que no existiera inconsistencia ni que sobraran para que haya redundancia. El diseño de la base de datos centralizada con la tercera forma normalizada, el diseño es amplio, pero aunque nunca se desarrolló se considera eficiente y completa.

El segundo diseño es un más simple y se creó con el objetivo de poder tener una fragmentación y distribución más sencilla, esto implica que la base no está normalizada, obvio cierta información y hay repeticiones de algunas tuplas. La fragmentación de este diseño es muy simple, tal como se explicó en clase se empleó fragmentación primaria y derivada. No presentó se hizo rápido y va acorde a lo que se hizo.

El cliente, la página web, es producto sencillo, al tener experiencia usando esta herramienta es simple de utilizar, no toma tiempo por la práctica, es simplemente ajustar las necesidades a lo que se especifica. Asimismo, la parte del servidor RESTful y la API ya se tenía práctica de proyectos anteriores por lo que se recicló código útil y se ajustó.

El problema de usar herramientas con NodeJS es lo pesado que puede llegar a ser por las dependencias de *node_modules* entre más dependencias y requerimientos llega a ocupar más espacio y un poco más complicado de manejar. Otro detalle que ya se tiene contemplado es que se requiere una curva de aprendizaje empezando desde JavaScript, ya que, puede ser difícil de entender o seguir debido a la complejidad que tiene. Sin embargo, es una solución sencilla, rápida y fácil una vez que se entiende.

Bibliografía

M. de la Rosa (2005). *Bases de Datos Distribuidas* [Online]. Tomado de: https://ebookcentral.proquest.com/lib/itcrsp/reader.action?docID=3174866\&query=sql\%2Bdistribuidas

Microsoft. (2017). *Types of Replication*. Tomado de: https://docs.microsoft.com/en-us/sql/relational-databases/replication/types-of-replication?view=sql-server-2017

Microsoft. (2017). *Merge Replication*. Tomado de: https://docs.microsoft.com/en-us/sql/relational-databases/replication/merge/merge-replication?view=sql-server-2017

Microsoft. (2018). *Tutorial: Prepare SQL Server for replication (publisher, distributor, subscriber)*. Tomado de: https://docs.microsoft.com/en-us/sql/relational-databases/replication/tutorial-preparing-the-server-for-replication?view=sql-server-2017

Microsoft. (2018). *Tutorial: Configure replication between a server and mobile clients (merge)"*. Tomado de: https://docs.microsoft.com/en-us/sql/relational-databases/replication/tutorial-replicating-data-with-mobile-clients?view=sql-server-2017