Instituto Tecnológico de Costa Rica

Bases de Datos Grupo 1

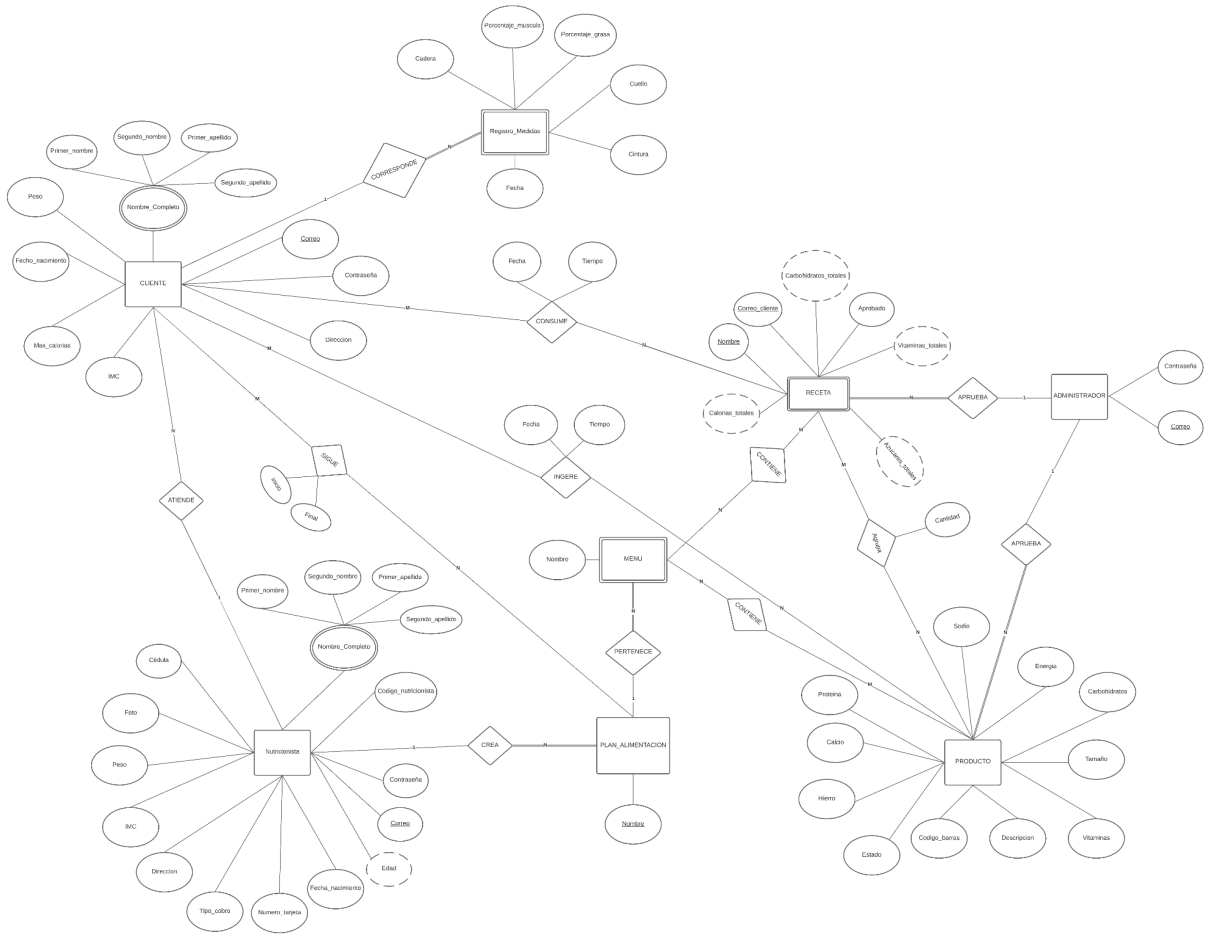
Documentación técnica NutriTEC

Grupo 2

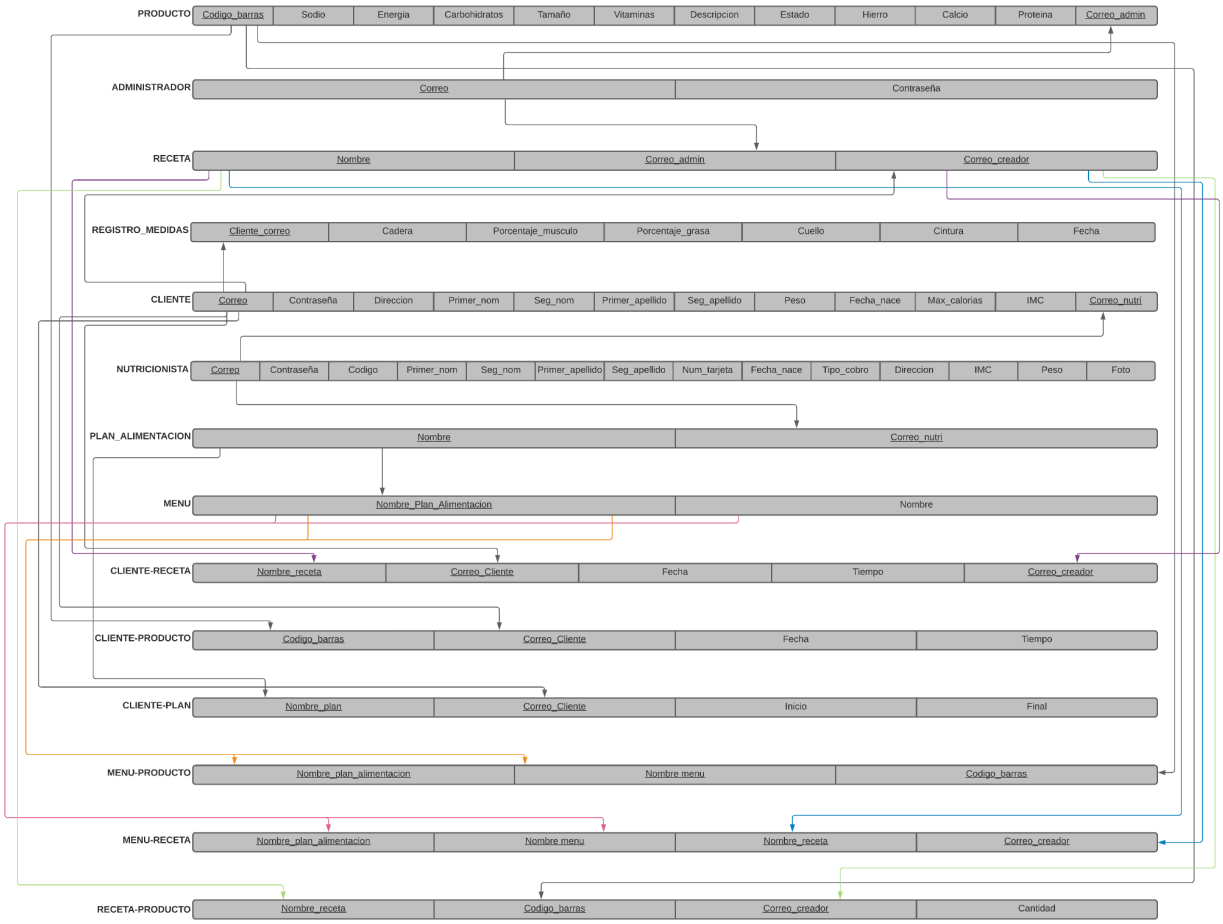
Alejandro Vásquez Oviedo
Luis Andrey Zuñiga Hernández
Adrián Gonzalez Jimenez
Brian Wagemans Alvarado

Semestre II

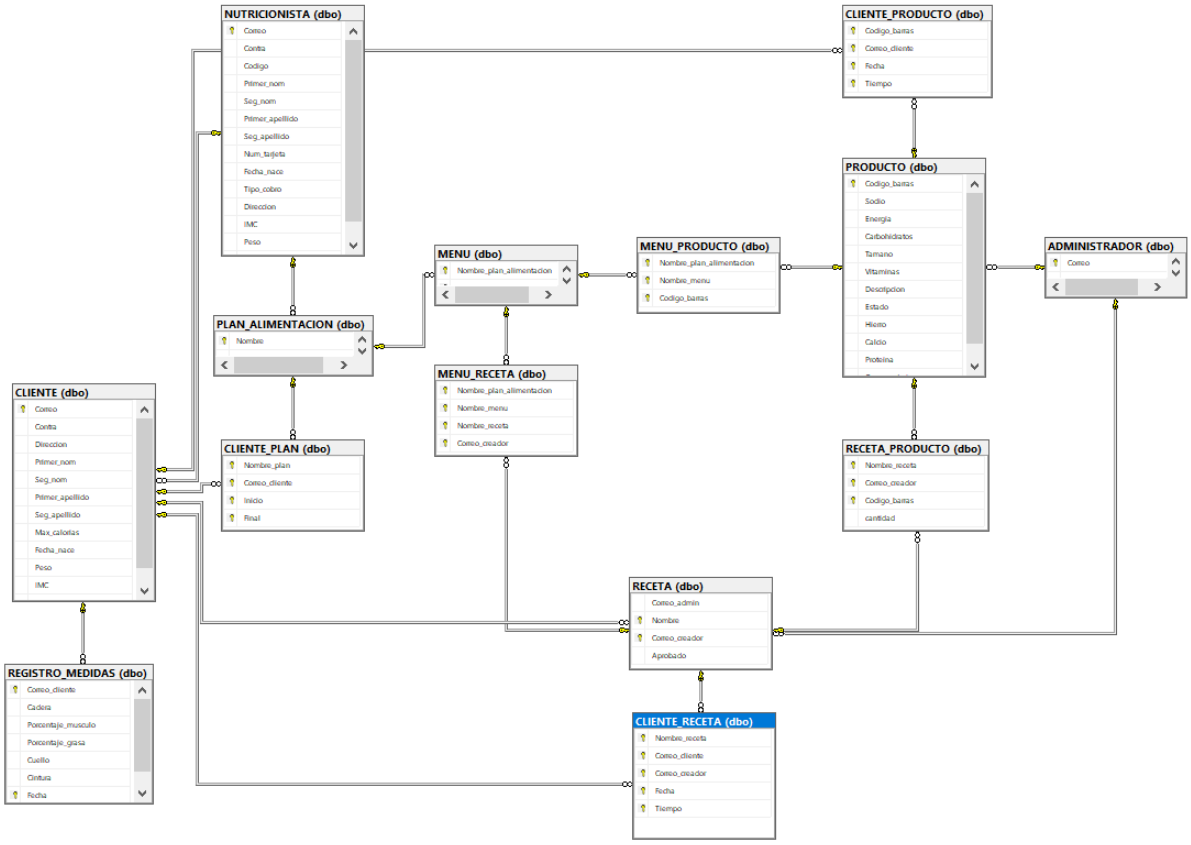
Modelo Conceptual:



Modelo Relacional:



Mapeo:


NUTRICIONISTA (dbo)
Correo
Contra
Codigo
Primer_nom
Seg_nom
Primer_apellido
Seg_apellido
Num_tarjeta
Fecha_nace
Tipo_cobro
Direccion
IMC
Peso
CLIENTE_PRODUCTO (dbo)
Codigo_barras
Correo_cliente
Fecha
Tiempo
PRODUCTO (dbo)
Codigo_barras
Sodio
Energia
Carbohidratos
Tamano
Vitaminas
Descripcion
Estado
Hierro
Calcio
Proteina
ADMINISTRADOR (dbo)
Correo
MENU (dbo)
Nombre_plan_alimentacion
MENU_PRODUCTO (dbo)
Nombre_plan_alimentacion
Nombre_menu
Codigo_barras
PLAN_ALIMENTACION (dbo)
Nombre
MENU_RECETA (dbo)
Nombre_plan_alimentacion
Nombre_menu
Nombre_receta
Correo_creador
CLIENTE (dbo)
Correo
Contra
Direccion
Primer_nom
Seg_nom
Primer_apellido
Seg_apellido
Max_calorias
Fecha_nace
Peso
IMC
CLIENTE_PLAN (dbo)
Nombre_plan
Correo_cliente
Inicio
Final
RECETA_PRODUCTO (dbo)
Nombre_receta
Correo_creador
Codigo_barras
cantidad
RECETA (dbo)
Correo_admin
Nombre
Correo_creador
Aprobado
REGISTRO_MEDIDAS (dbo)
Correo_cliente
Cadera
Porcentaje_musculo
Porcentaje_grasa
Cuello
Cintura
Fecha
CLIENTE_RECETA (dbo)
Nombre_receta
Correo_cliente
Correo_creador
Fecha
Tiempo

Estructuras de datos desarrolladas:

1. Nutricionista: Se crea una tabla nutricionista con el correo, la contraseña,el código de nutricionista, el nombre completo, el número de tarjeta, la fecha de nacimiento, el tipo de cobro, la dirección, el índice de masa corporal y el peso.
2. Producto: Se crea una tabla producto con la información nutricional del producto: miligramos de sodio, miligramos de calcio, miligramos de hierro, gramos de proteína, gramos de vitaminas, gramos de carbohidratos, gramos por porción, kcal de energía, el codigo de barras y correo del administrador.
3. Administrador: Se crea una tabla con el correo y la contraseña, esto se hace ya que solo se necesita saber las credenciales del administrador.
4. Receta: Se crea una tabla para las recetas creadas por el usuario, esta contiene el correo del administrador que validó dicha receta, el nombre de la receta, el correo del creador y el estado en el que se encuentra la receta.
5. Registro Medidas: Se crea una tabla para las medidas físicas de los usuarios, esta contiene el correo, tamaños de: cadera, cuello, cintura. Además tiene los porcentajes de músculo y grasa. Finalmente se guarda el IMC y el peso.
6. Cliente: Se crea una tabla que represente al cliente, para ello se guarda el correo, la contraseña, la dirección, el nombre, el IMC, el máximo de calorías, la fecha de nacimiento y el nutricionista al que está asociado.
7. Plan Alimentación: Se crea una tabla para almacenar la información del plan de alimentación, esta contiene el tiempo inicial, el final el correo del nutricionista que lo hizo y un nombre.
8. Menú: Es una tabla que sirve como puente entre el plan de alimentación y las comidas. Tiene almacenado el tiempo asignado y el nombre del plan de alimentación.
9. Cliente Receta: es una tabla que representa la relación entre Cliente y Receta. Además de contener las llaves de las tablas Cliente y Receta contiene la fecha y el tiempo en el que se comió.
10. Cliente Producto: Es una tabla que representa la relación entre el Cliente y el Producto. Además de contener las llaves de las tablas Cliente y Producto contiene la fecha y el tiempo en el que se comió.
11. Cliente Plan: Es una tabla para relacionar Cliente y el Plan de alimentación. Además, de los atributos llave tiene la fecha inicial y la fecha final.
12. Menu Producto:Es una tabla para relacionar Producto y Menú.Además, de los atributos llave tiene la cantidad.
13. Menú Receta:Es una tabla para relacionar Receta y Menú. Solo tiene los atributos llave.
14. Receta Producto: Es una tabla para relacionar Producto y receta.Además, de los atributos llave tiene la cantidad.

Se debe agregar que se cumplen las reglas de normalidad en el modelo elegido.

Stored Procedures, Triggers y Vistas implementadas:

Vistas:

1. Nutricionista pública: Es una view creada para encapsular la información importante del nutricionista como puede ser el tipo de cobro o el número de tarjeta.
2. Client public: Es una view creada para encapsular información importante del cliente.

3. Receta pública: Es una view con la funcionalidad de devolver la interpretación correcta de la tabla Receta Producto.

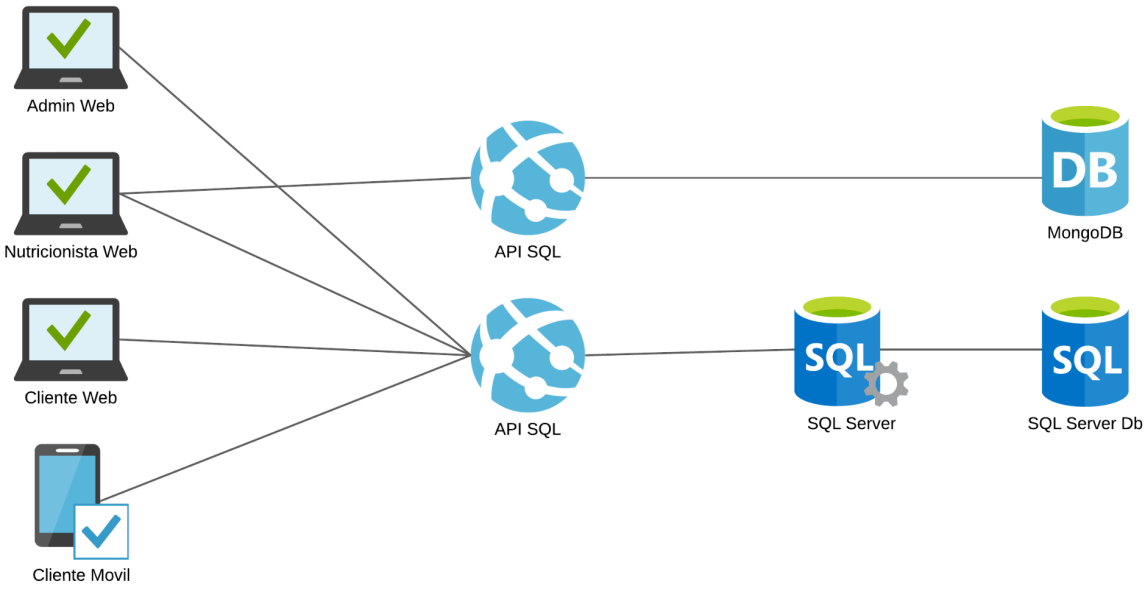
Triggers:

1. TR insert product: Este trigger se asegura de que los datos insertados en productos estén en un estado válido al inicio.
2. TR delete plan: Este trigger hace la funcionalidad de cascada para la tabla plan de alimentación.

Stored Procedures:

1. GetPaymentAmount: Este store procedure tiene la funcionalidad de devolver una tabla con la información necesaria para los cobros.
2. spAddPlan: Este store procedure verifica si el usuario que realizó la operación tiene los permisos para crear dicha operación y si los tiene, agrega un plan con los datos especificados.
3. spAddMenuToPlan: Este store procedure tiene la función de agregar un menú al plan de alimentación de un cliente, si la persona realiza esta llamada tiene el permiso para hacerlo.
4. spAddProductToMenu: Este store procedure tiene la funcionalidad de agregar un producto al menú, solamente si existe dicho menú.
5. spAddRecetaToMenu: Este store procedure tiene la funcionalidad de agregar una receta al menú, solamente si existe dicho menú.

Arquitectura desarrollada:



Problemas conocidos:

- La CRUD no está completa en lo que respecta, deletes y updates para las recetas, productos, clientes, nutricionistas.

Problemas encontrados:

- No se mostraban los toast en android: Para notificar los estados de las solicitudes con el server, es decir por ejemplo si un proyecto fue procesado, o existe algún fallo, sin embargo no se mostraban. Se encontró que la causa era que se estaban llamando de un

hilo diferente al de la UI, por lo que estos toast no se agregan. Se soluciono esto obligando a la acción a ejecutarse en dicho hilo mediante el método runOnUiThread()

- Problemas en el SSL en la aplicación móvil: En la fase de pruebas y desarrollo, cuando aún se utilizaba la base de datos en el localhost y se utilizaba IIS Express, al acceder en la aplicación móvil. Para solucionarlo se debió agregar comando en la configuración del cliente de okHttp para evitar el problema.
- Uso de store procedure desde el API: Cuando se estaba utilizando el API específicamente entity framework para acceder a los store procedure de la base de datos, la información no era procesada, sin embargo el store procedure funcionaba adecuadamente. Al parecer el problema se encontraba en que no se almacenaba la "respuesta" en .net, y debido a que el proceso en realidad se ejecuta de manera asíncrona, y no existía la referencia, no se actualizaba. La solución fue simplemente asignar una variable a la "respuesta"
- El API para MongoDB era incapaz de acceder a la base de datos: Al subir el API para la base de datos NOSQL, ubicada en Atlas, sin embargo, no se era capaz de realizar la conexión desde el api en azure a la base de datos en Atlas, el problema corresponde a que en la base de datos en Atlas no se había configurado para aceptar la ip de la API
- El API de SQL: Al cargar la base de datos a azure siguiente el proceso descrito en el manual de instalación el API era incapaz de conectar correctamente con la base de datos, el problema consistía en problemas con el String de conexion, el cual apesar de ser configurado en el proceso de creación para azure, no era accedido y utilizado en el contexto, la solución consistió en utilizar la clase Configuration de .net core y acceder a dicho parámetro.
- Display de la la web app en azure: Al subir el api a azure no detectaba correctamente la clase HttpClient, por lo que no funcionaba de manera correcta. Para solucionarlo debe de cambiarse el ambiente de desarrollo a .Net 5 para que contempla las dependencia

Conclusiones:

- Azure es una herramienta útil para el hosting de aplicaciones en la nube, pero esta puede tener problemas para la configuración, por lo que hay que tener cuidado al usarla.
- Los stored procedures, views y triggers son herramientas útiles para configurar y manejar la base de datos de una manera más efectiva.
- Entity Framework tiene una interfaz para la comunicación sencilla con bases de datos SQL aun, cuando estas peticiones son SQL Raw.

Recomendaciones:

- Se recomienda usar MongoDriver para la comunicación de .net core con Atlas u otra implementación de MongoDB
- Se recomienda la utilización de fragments en el desarrollo de android ya que permite la reutilización de interfaz y lógica
- Se recomienda la definición de una Navigation para el control del flujo entre los fragments, ya que facilitan las transiciones
- Se recomienda utilizar azure para el deployment de la aplicación ya que provee un hosting, con control interactivo.
- Se recomienda la utilización de entity framework para la comunicación de bases de datos sql con .net core
- Se recomienda utilizar Jira o alguna herramienta similar para registrar el progreso y las tareas por realizar.
- Se recomienda la utilización de un API Manager en conjunto con el API service, ya que permite visualizar las distintas solicitudes presentes en el API
- Se recomienda la utilización de triggers para verificaciones o etapas complejas previas o posteriores necesarias después de las distintas operaciones en la base de datos
- Se recomienda usar view para asociar información inherente a distintas tablas que sea requerida constantemente o para obviar columnas de tablas innecesarias para el usuario
- Se recomienda usar stored procedures para encapsular la lógica de operaciones importantes o frecuentes ya que permite unificar el código en un solo lugar, lo que facilita las correcciones.

Bibliografía consultada:

- **ZZZ Projects.*What's New in EF Core 5.* https://www.learnentityframeworkcore5.com/whats-new-in-ef-core-5**
- **MongoDB.*Get Started with Atlas.* https://docs.atlas.mongodb.com/getting-started/**
- **MongoDB. (2021). *MongoDB C#/.NET Driver.* https://docs.mongodb.com/drivers/csharp/**