TEC | Tecnológico de Costa Rica

BASES DE DATOS | CE – 3101

NutriTEC

Documentación

Estudiantes:

Saúl Gómez Ramírez

Esteban MoralesUreña

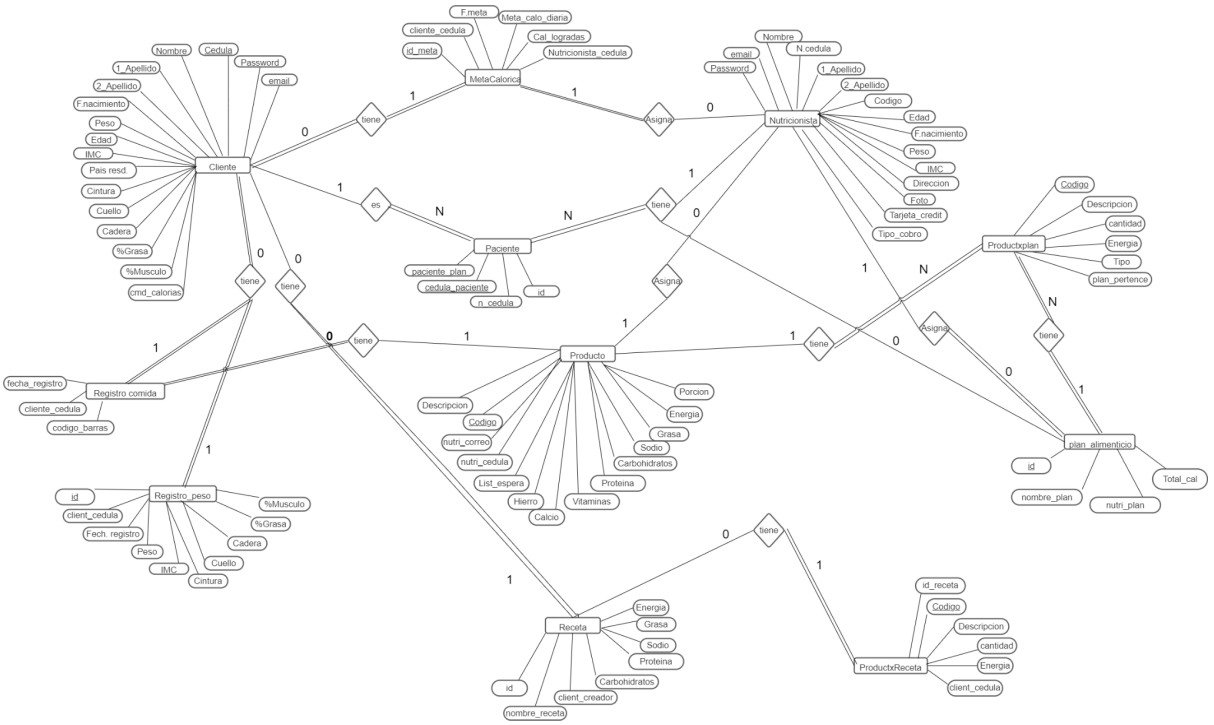
Steven Castro Medina

Profesor:

Marco Rivera Menese

SII-2020

◆ Modelo conceptual utilizando la notación de Chen.



◆ Modelo relacional.

◆ Descripción de las estructuras de datos desarrolladas (Tablas).

- Producto: aquí se almacenan todos los productos a los cuales tiene acceso el nutricionista para asignar plan alimenticio. Los guarda en lista de espera.

- Cliente: registra todos los datos de potenciales pacientes, ya que todo paciente es un cliente, pero no todo cliente es un paciente.
- Nutricionista: almacena todos los datos de los nutricionistas y crea las

diversas relaciones con plan alimenticio, producto, meta calórica, y paciente.

- Recetas: toma distintos productos y los conjuga entre sí, para así asignarlo a un debido plan.

- Paciente: toma un cliente apto y lo asocia a un nutricionista, así que el cliente cambia la condición y se crean otras relaciones más.

- Plan alimenticio: todo nutricionista asigna un plan calórico a un paciente, por lo que cada paciente tiene un plan asignado.

- Registro comida: registra los productos disponibles de un plan alimenticio.

- Registro peso: toma los datos de un cliente y una vez expuesto este como paciente copia los datos de peso en esta tabla.

- Meta calórica: Tomando datos de diversas tablas el nutricionista asigna esta meta a un paciente determinado, así se crean relaciones como las de plan alimenticio.

- Productos por plan: son los productos disponibles de un plan asignado.

- Productos por receta: de los productos disponibles toma ciertos y los conjuga para la creación de una receta.

◆ **Descripción de los Store Procedures, Triggers y Vistas implementados.**

User store procedures

- Reporte cobro: acá se toman datos tales como cédula, nombre, apellidos, tarjeta, tipo de cobro, monto, número de clientes, y con los pacientes tomados de la tabla pacientes, hace un count de esta misma tabla a partir de número de cédula, el monto es decidido o calculado según la cantidad de pacientes y el tipo de cobro que corresponde a este, funciona a través de un case, para dividir cobro semanal, mensual y anual y sus respectivos descuentos, a aparte usa un join para que no me muestren los nutricionistas sin pacientes.

- Registro peso: a partir de la toma del dato cédula cliente, busca y selecciona el registro de los pesos de ese cliente en concreto.

- Pacientes por nutricionista: en este caso se busca y seleccionan todos los datos relevantes que posee la tabla cliente para con esto asociarlo a un nutricionista.

- Insertar producto en plan: se toma el código, tiempo y el plan al que pertenece dicho producto, teniendo en cuenta también la cantidad de ese producto, dado esto lo selecciona como insert into productos plan y con el código de barras, descripción y calorías, y de esta manera lo agrega en un plan.

- Productos por plan: devuelve todos los productos por plan determinado.

- Crear plan alimenticio: en este caso teniendo en cuenta la cédula nutricionista y un plan alimenticio, se ejecuta a través de un insert. Existe una llave compuesta dónde está el id_nombre del plan y calculado calorías se toma el id del plan, se declara dentro una variable “calorías”, y esa variable se le asigna un select de la suma de los valores de energía de productos por plan,

o sea calcula todas las calorías dentro de un plan y así se actualiza este.

Triggers

- TR_BorrarDatosPlan: cuando se borra un plan de tabla de plan alimenticio, también borra los productos, y luego que haga un update a los pacientes con este plan y que lo ponga en nulo.

- TR_BorrarDatosNutri: Cuando se borra un nutricionista, borre todos los pacientes asignados a este y todos los planes que este asigno.

- TR_menorDe13: verifica si el paciente es menor a 13 años, si lo es no lo deja asignarse a un nutricionista.

- TR_insertPeso: cuando un cliente se registra, copia los datos de pesos de este paciente y los pega en registro peso.

- TR_BorrarDatosCliente: si elimino un cliente, eliminó todo lo asignado a este, ya que este deja mucho rastro en muchas otras tablas.

Vistas implementados

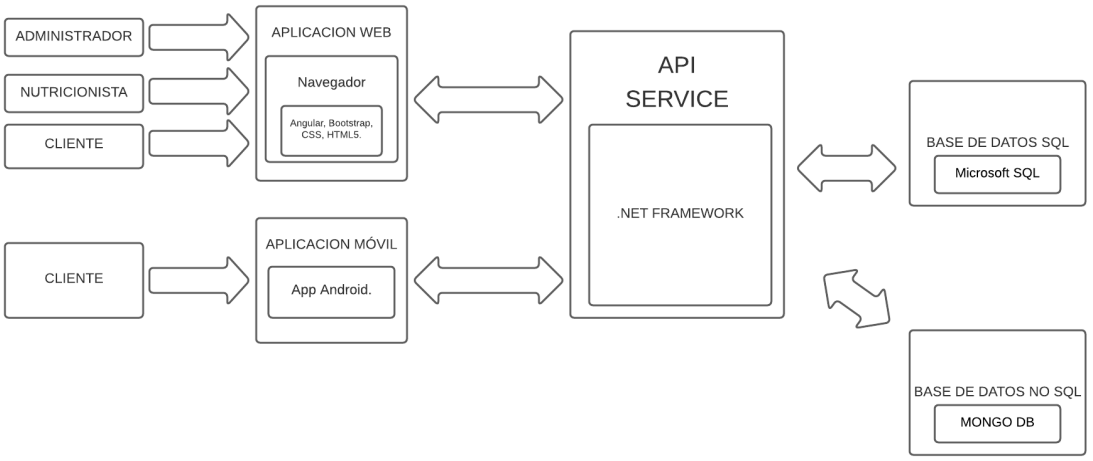
- Vista Web Administrador: en este caso se llega un 100% de implementación de la vista administrador, esta posee los aspectos de configuración de la plataforma.

- Vista Web Nutricionista: en este caso se permite a los nutricionistas buscar y asociar clientes como sus pacientes y crearles y asignarles el plan alimenticio,

sin embargo se estima en un 80% de un completitud.

- Vista Web Cliente: esta vista permite a los clientes darse de alta, buscar productos, agregar metas, aunque existe un faltante de gráficos en el historial de peso, las recetas, y al registrar su consumo diario existe una faltante de botones de manera de interfaz.

◆ **Descripción detallada de la arquitectura desarrollada.**



◆ **Problemas conocidos: (sin solución)**

En el caso del frontend en las vistas web, a manera de buscar una interfaz amigable con el usuario se implementa un fondo llamativo, alusivo al tema de la nutrición y los alimentos, este fondo al navegar en las diferentes vistas suele verse cortado, e irse desplegando conforme se despliegan las tablas correspondientes, es decir la imagen de fondo nunca llega a rellenar el campo visual completamente.

En el caso del API, se llega a presentar un problema muy curioso y que requirio

extensas horas de estabilización, sin embargo, no se llegó a una solución en concreto, en extensión lo que sucedió fue que cada ejecución del API el entity framework en la clase “NutriTecmodel.context” se da un proceso automáticamente, que ejecuta y borra la linea de codigo “Configuration.ProxyCreationEnabled = false; ”, lo cual bota el API, lo que llegó a estabilizar este problema es tener en cuenta que cada que se borra se agregue manualmente esta línea de código.

◆ **Problemas encontrados: (solucionados)**

Se presenta error del IIS por parte de un desarrollador, este se soluciono de manera tal que se descargó nuevamente todo de github donde se almacena todo el proyecto y se limpio el proyecto de manera local en la pc donde estabas las afectaciones, se eliminó carpetas de publish, y un historial de archivos duplicados y con esto se dio resultado positivo.

- *Intentos de solución 4.*
- *Horas de implicación 8.*
- *Recursos consultados*

  https://stackoverflow.com/questions/5275615/what-does-iis-consist-of-and-how-does-it-work

  https://www.daveaglick.com/posts/debugging-stack-overflows-on-iis

De parte de la base de datos SQL, existe un problema de manera tal que la tabla “productos por plan”, no se generó inicialmente con una llave compuesta, la única llave compuesta que había era código de barras, que también resultó ser foránea, se modificó la cardinalidad y esto resultó como solución al problema, la modificación fue de 0 a 1 se cambió a de 1 a muchos, así no pueden haber atributos nulos en las últimas dos tablas.

- *Intentos de solución 2.*

- *Horas de implicación 2.*
- *Recursos consultados*

  ninguno.

➔ Conclusiones del proyecto.

El proyecto presentado genera una arquitectura bastante avanzada y robusta por lo que el proceso de desarrollo conllevo extensas horas de implementación así como de investigación, a partir de esto se concluye con un proyecto bastante acertado a lo que se desea como solución al problema, atacando todos sus apartados como lo fueron bases de datos relacional y no relaciones, correcta comunicación de estas con un API, y a partir de este con productos web y android, lo que arroja un producto disponible al usuario en proyección cumpliendo especificaciones y requerimientos.

➔ Recomendaciones del proyecto.

El proceso de investigación siempre resulta fundamental en el desarrollo de software, el uso de tecnologías de frontend requieren actualización muy precisa a la hora de su utilización, para realizar esta con más aserción al objetivo que se tiene en mente como solución, por lo que se recomienda un manejo e investigación constante de herramientas de utilidad que devuelven soluciones mas faciles en implementacion robustas como la del presente proyecto.

El servicio de IIS, presente en todo el desarrollo de este proyecto se vuelve fundamental y es por esto que el manejo correcto de este va a evitar errores, que generan atrasos de gran importancia en el desarrollo, por lo que se recomienda un manejo preciso de IIS, completamente orientado a la solución que se busca dar a determinado problema.

➔ **Bibliografía consultada en todo el proyecto.**

Ruiz, T.(2012, 8 marzo). Angular Material Tutorial. Recuperado 25 de octubre de 2021, de https://www.youtube.com/watch?v=rWOwTVSMfPw

Consuming APIs with Retrofit | CodePath Android Cliffnotes. (s. f.). CODEPATH. Recuperado 26 de octubre de 2021, de https://guides.codepath.com/android/consuming-apis-with-retrofit

Ramos, J. (s. f.). Android: Aprende a consultar una API y procesar la respuesta (con Retrofit). Programación y más. Recuperado 26 de septiembre de 2021, de https://programacionymas.com/blog/consumir-una-api-usando-retrofit

Saunders, J. (2019, 2 abril). Angular HTTP GET Example with JSON. Recuperado 25 de octubre del 2021, de

https://www.youtube.com/watch?v=SYBzE68Ee-g

Saunders, J. (2019, 2 abril). Angular HTTP GET Example with JSON. Recuperado 25 de octubre del 2021, de https://www.youtube.com/watch?v=74X18AoZ2Gk