NutriTEC: Documentación Técnica

Bases de Datos

Instituto Tecnológico de Costa Rica

Proyecto # 2

Integrantes:

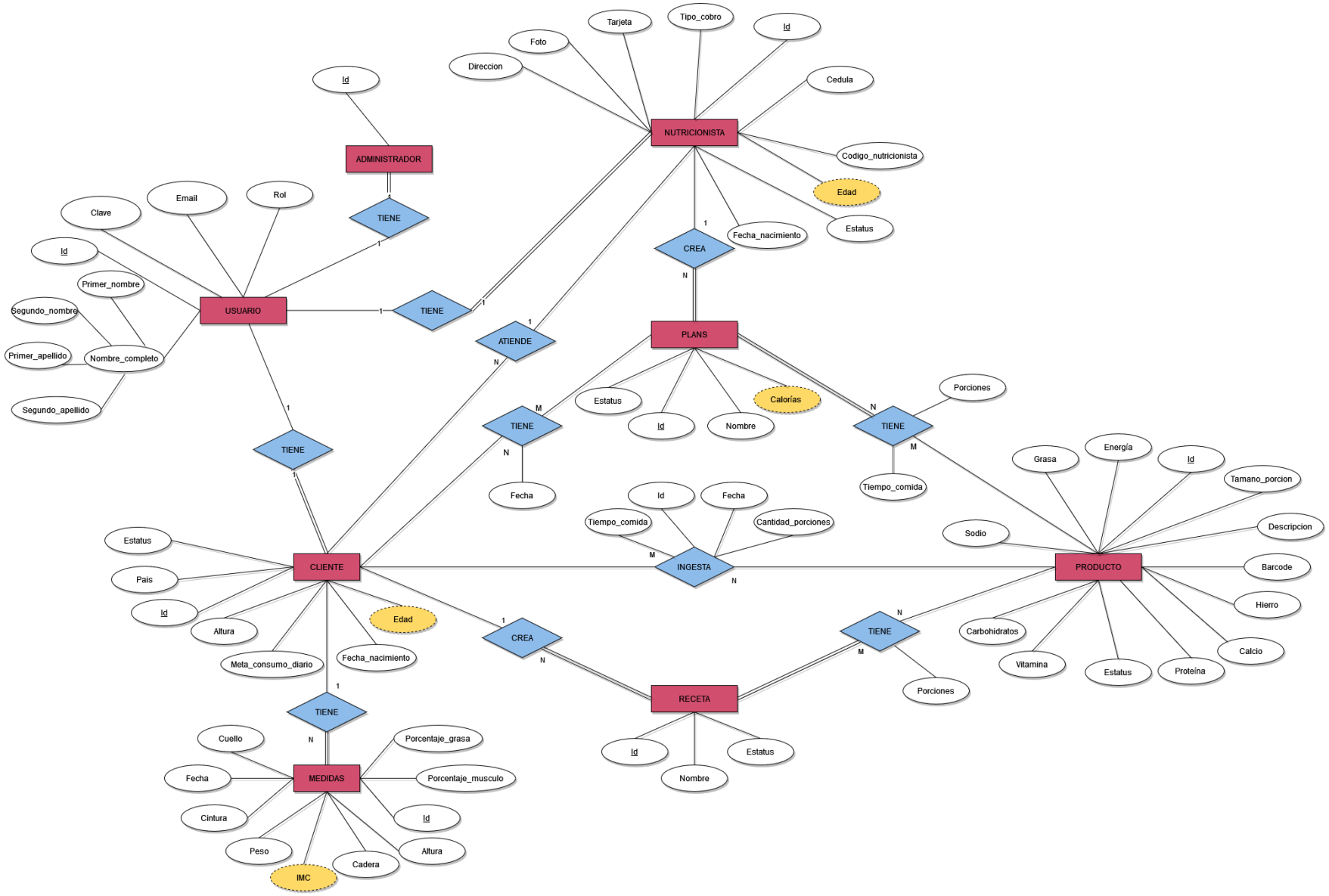
Carlos Adrián Araya Ramírez	*2018319701*
José Alejandro Chavarría Madriz	*2019067306*
Sebastián Mora Godínez	*2019227554*
Michael Shakime Richards Sparks	*2018170667*

Prof. Marco Rivera Meneses

Noviembre, 2021

Índice

Modelo Conceptual



Mapeo de Diagrama Relacional

☐Mapeo de entidades fuertes:

Las identidades fuertes identificadas fueron USUARIO, CLIENTE, NUTRICIONISTA, ADMINISTRADOR, MEDIDAS, PRODUCTO, RECETA Y PLANES, por lo que se creó una relación para cada una de estas entidades.

- Para *USUARIO* los atributos identificados fueron rol, email, clave, primer nombre, segundo nombre, primer apellido, segundo apellido y de llave primaria id autoincremental.
- Para *ADMINISTRADOR* se identifica como llave primaria id autoincremental.
- Para *NUTRICIONISTA* están dirección, foto, tarjeta, tipo de cobro, cédula, código de nutricionista, estatus, fecha de nacimiento y id como llave primaria autoincremental.
- Para *CLIENTE* los atributos fueron país, estatus, altura, meta consumo diario, fecha de nacimiento, edad como atributo derivado y id como llave primaria autoincremental.
- Para *MEDIDAS* se encuentran cuello, fecha, cintura, peso, cadera, altura, porcentaje músculo, porcentaje grasa, IMC como atributo derivado y id como llave compuesta.
- Para *RECETA* se identifican nombre y estatus, como llave primaria id.
- Para *PRODUCTO* están sodio, grasa, energía, tamaño de la porción, descripción, barcode, hierro, calcio, proteína, estatus, vitamina, carbohidratos y id como llave primaria autoincremental.
- Para *PLANES* los atributos son estatus, nombre, calorías como atributo derivado y id como llave primaria autoincremental.

☐Mapeo de entidades débiles

En el caso de las entidades débiles, no se identificó ninguna.

☐Mapeo de asociaciones binarias 1:1

En este paso entre las relaciones 1:1, se incluye la llave foránea en la relación del lado de participación total. Por lo que se procede a realizar los siguientes pasos.

Existen tres relaciones 1:1 *TIENE* entre *USUARIO - CLIENTE, USUARIO - NUTRICIONISTA* y *USUARIO - ADMINISTRADOR*, como la participación total se encuentra del lado de *CLIENTE, NUTRICIONISTA y ADMINISTRADOR*, se agrega a estas tabla una llave foránea (id_usuario), la cual corresponde a la llave principal (id) de *USUARIO*. Es necesario que exista un usuario para poder establecer un cliente, un nutricionista o un administrador.

☐Mapeo de asociaciones binarias 1:N

En este paso se incluye como llave foránea en la relación del lado N la llave primaria de la relación de lado 1. Por lo que se procede a realizar los siguientes pasos.

- Para la relación de *CLIENTE* a *MEDIDAS*, se incluye id_cliente como llave foránea en *MEDIDAS*, que corresponde a la llave primaria de *CLIENTE*.
- Para la relación de *CLIENTE* a *RECETA*, se incluye id_cliente como llave foránea en *RECETA*, que corresponde a la llave primaria de *CLIENTE*.
- Para la relación de *NUTRICIONISTA* a *CLIENTE*, se incluye id_nutricionista como llave foránea en *CLIENTE*, que corresponde a la llave primaria de *NUTRICIONISTA*.
- Para la relación de *NUTRICIONISTA* a *PLANES*, se incluye id_nutricionista como llave foránea en *PLANES*, que corresponde a la llave primaria de *NUTRICIONISTA*.

☐ Mapeo de asociaciones binarias N:M

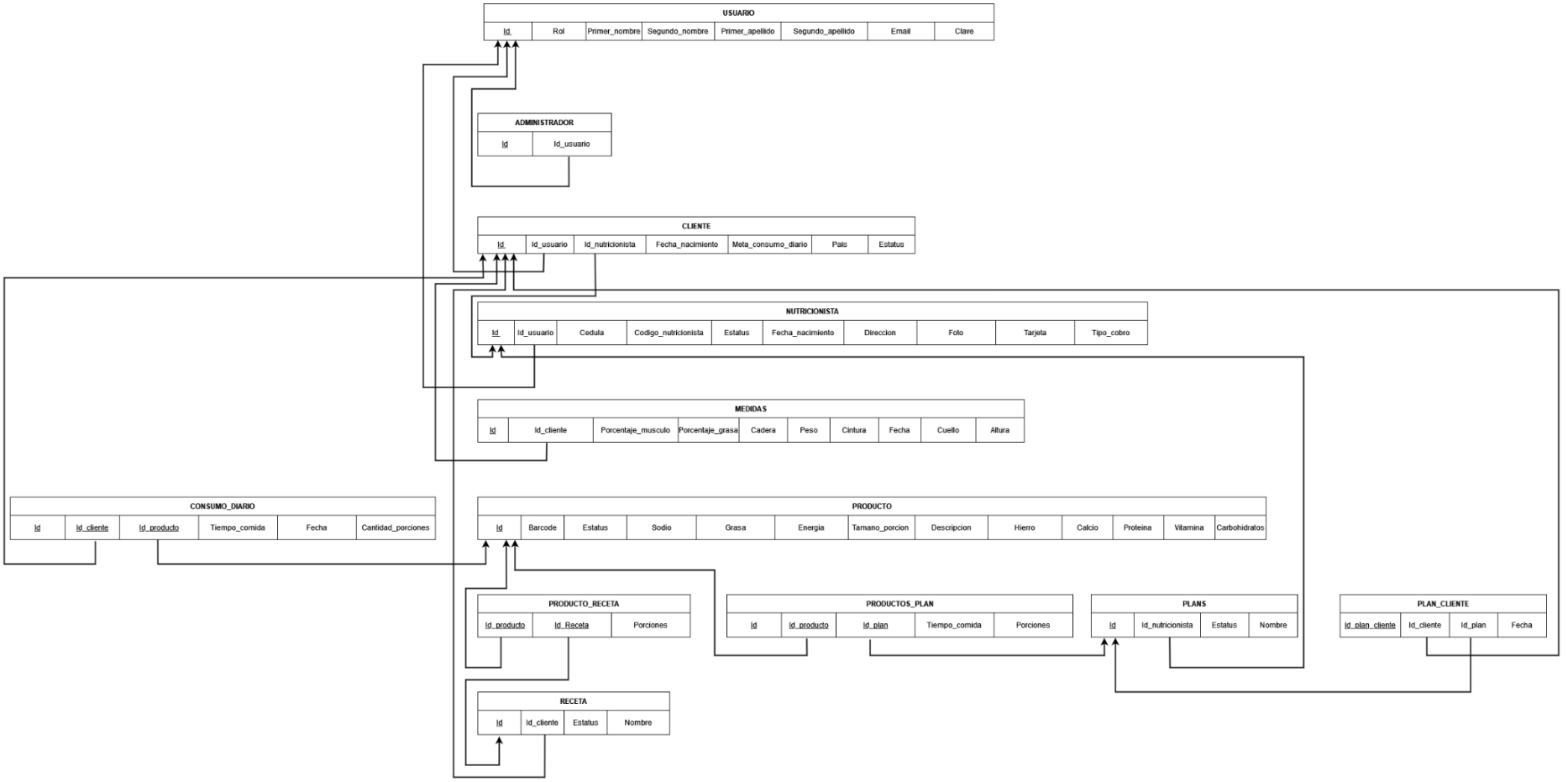
En este paso para cada relación N:M, se crea una nueva relación intermedia, en la cual la llave primaria de la nueva relación se compone de la llave primaria de ambas relaciones asociadas. Por lo que se procede a realizar los siguientes pasos.

- Para la relación de *RECETA* a *PRODUCTO*, se crea la entidad RECETA_PRODUCTO, para la cual su llave primaria es compuesta y está conformada por la llave primaria de *RECETA, PRODUCTO y una llave autoincremental*.
- Para la relación de *CLIENTE* a *PRODUCTO*, se crea la entidad CONSUMO_DIARIO, para la cual su llave primaria es compuesta y está conformada por la llave primaria de *CLIENTE, PRODUCTO y una llave autoincremental*.
- Para la relación de *CLIENTE* a *PLANS*, se crea la entidad PLAN_CLIENTE, para la cual su llave primaria es compuesta y está conformada por la llave primaria de *CLIENTE, PLANES y una llave autoincremental*.
- Para la relación de *PLANES* a *PRODUCTO*, se crea la entidad PRODUCTOS_PLAN, para la cual su llave primaria es compuesta y está conformada por la llave primaria de *PLANES, PRODUCTO y una llave autoincremental*.

☐ Mapeo de atributos multivaluados

Se omite pues no hay este tipo de atributos en la implementación propuesta.

Modelo Relacional



Estructuras de Datos Desarrolladas

Para el almacenamiento y organización de los datos se utilizaron tablas en una base de datos implementada haciendo uso de SQL Server, estas tablas tienen una estructura desarrollada para cada tipo de entidad del mini mundo tomando ciertos atributos que los caracterizan particularmente, todos tienen una llave primaria, la cual se encuentra denotada en negrita. En esta llave se utilizan uno o varios atributos para definir la unicidad de cada instancia de la entidad, de esta forma se puede hacer una búsqueda sencilla y evita duplicados.

A continuación se muestra la forma de cada una de estas tablas:

☐ USUARIO

id : int
rol : varchar(20)
primer_nombre : varchar(20)
segundo_nombre : varchar(20)
primer_apellido : varchar(20)
segundo_apellido : varchar(20)
email : varchar(20)
clave : varchar(max)

☐ CLIENTE

id : int
id_usuario : int
id_nutricionista : int
estatus : varchar(20)
fecha_nacimiento : Date
meta_consumo_diario : float
pais : varchar(20)

☐ ADMINISTRADOR

id : int
id_usuario : int

☐ NUTRICIONISTA

id : int
id_usuario : int
cedula : varchar(20)
codigo_nutricionista : int
estatus : varchar(20)
fecha_nacimiento : Date
tarjeta: varchar(20)
tipo_cobro : varchar(20)
direccion : varchar(20)

☐ PRODUCTO_RECETA

id_receta : int
id_producto : int
porciones : float

☐ PLANS

id : int
id_nutricionista : int
estatus : varchar(20)
nombre: varchar(20)

☐ MEDIDAS

id : int
id_cliente : int
fecha : Date
porcentaje_musculo : float
porcentaje_grasa : float
cadera : float
peso : float
altura : float
cintura : float
cuello : float

☐ PRODUCTO

id : int
id_cliente : int
estatus : varchar(20)
barcode : varchar(20)
descripcion : varchar(20)
tamano_porcion : float
sodio : float
grasa : float
energia : float
hierro : float
calcio : float
proteina : float
vitamina : float
carbohidratos : float

☐ RECETA

id : int
id_cliente : int
estatus : varchar(20)
nombre: varchar(20)

☐ PLAN_CLIENTE

id_plan_cliente : int
id_plan : int
id_cliente : int
fecha: Date

☐ PRODUCTOS_PLAN

id : int
id_plan : int
id_producto : int
tiempo_comida : varchar(20)
porciones : float

☐ CONSUMO_DIARIO

id : int
id_cliente : int
id_producto : int
tiempo_comida : varchar(20)
cantidad_porciones : int
fecha: Date

Las entidades cuentan con atributos que las relacionan con otras entidades, denominados foreign keys. Estos atributos toman el valor de la primary key de la entidad a la que se está referenciando. Las relaciones presentes son las siguientes:

Referencia Administrador → Usuario

```
ALTER TABLE "Administrador"
ADD CONSTRAINT ADMINISTRADOR_USUARIO_FK FOREIGN KEY(id_usuario)
REFERENCES "Usuario"(id)
```

Referencia Cliente → Nutricionista

```
ALTER TABLE "Cliente"
ADD CONSTRAINT CLIENTE_NUTRICIONISTA_FK FOREIGN KEY(id_nutricionista)
REFERENCES "Nutricionista"(id)
```

Referencia Cliente → Usuario

```
ALTER TABLE "Cliente"
ADD CONSTRAINT CLIENTE_USUARIO_FK FOREIGN KEY(id_usuario)
REFERENCES "Usuario"(id)
```

Referencia Nutricionista → Usuario

```
ALTER TABLE "Nutricionista"
ADD CONSTRAINT NUTRICIONISTA_USUARIO_FK FOREIGN KEY(id_usuario)
REFERENCES "Usuario"(id)
```

Referencia Medidas → Cliente

```
ALTER TABLE "Medidas"
ADD CONSTRAINT MEDIDAS_CLIENTE_FK FOREIGN KEY(id_cliente)
REFERENCES "Cliente"(id)
```

Referencia Consumo_diario → Cliente

```
ALTER TABLE "Consumo_diario"
ADD CONSTRAINT CONSUMO_DIARIO_CLIENTE_FK FOREIGN KEY(id_cliente)
REFERENCES "Cliente"(id)
```

Referencia Consumo_diario → Producto

```
ALTER TABLE "Consumo_diario"
ADD CONSTRAINT CONSUMO_DIARIO_PRODUCTO_FK FOREIGN KEY(id_producto)
```

```
REFERENCES "Producto"(id)
```

Referencia Receta → Cliente

```
ALTER TABLE "Receta"
ADD CONSTRAINT RECETA_CLIENTE_FK FOREIGN KEY(id_cliente)
REFERENCES "Cliente"(id)
```

Referencia Producto_receta → Producto

```
ALTER TABLE "Producto_receta"
ADD CONSTRAINT PRODUCTO_RECETA_PRODUCTO_FK FOREIGN KEY(id_producto)
REFERENCES "Producto"(id)
```

Referencia Producto_receta → Receta

```
ALTER TABLE "Producto_receta"
ADD CONSTRAINT PRODUCTO_RECETA_RECETA_FK FOREIGN KEY(id_receta)
REFERENCES "Receta"(id)
```

Referencia Productos_plan → Producto

```
ALTER TABLE "Productos_plan"
ADD CONSTRAINT PRODUCTOS_PLAN_PRODUCTO_FK FOREIGN KEY(id_producto)
REFERENCES "Producto"(id)
```

Referencia Productos_plan → Plans

```
ALTER TABLE "Productos_plan"
ADD CONSTRAINT PRODUCTOS_PLAN_PLANS_FK FOREIGN KEY(id_plan)
REFERENCES "Plans"(id)
```

Referencia Plans → Nutricionista

```
ALTER TABLE "Plans"
ADD CONSTRAINT PLANS_NUTRICIONISTA_FK FOREIGN KEY(id_nutricionista)
REFERENCES "Nutricionista"(id)
```

Referencia Plan_cliente → Plans

```
ALTER TABLE "Plan_cliente"
ADD CONSTRAINT PLAN_CLIENTE_PLAN_FK FOREIGN KEY(id_plan)
REFERENCES "Plans"(id)
```

Referencia Plan_cliente → Cliente

```
ALTER TABLE "Plan_cliente"
```

```
ADD CONSTRAINT PLAN_CLIENTE_CLIENTE_FK FOREIGN KEY(id_cliente)
REFERENCES "Cliente"(id)
```

Stored Procedures, Triggers y Vistas

Stored Procedures

Master Client

Gestiona la selección, inserción y actualización de un cliente, así como la asignación y desasignación de un nutricionista, el registro de medidas, consumo diario y el reporte de avance.

Master Nutricionista

Gestiona la selección, inserción y actualización de un nutricionista, así como la asignación de planes a un cliente, el seguimiento de pacientes y el consumo diario de estos.

Master Receta

Gestiona la selección, inserción, actualización y eliminación de una receta, así como la adición, actualización o eliminación de productos dentro de esta.

Master Plans

Gestiona la selección, inserción, actualización y eliminación de planes para un nutricionista en específico, además es responsable de agregar y eliminar los productos de un plan.

Master Products

Gestiona la selección, inserción y actualización de un producto, mediante la actualización se aprueba o se rechaza un producto.

Register

Realiza la acción de registrar, contempla el registro para cada uno de los roles en su respectiva tabla. Además en cada ejecución se crea un usuario automáticamente para el nuevo registro que se debe crear.

Log In

Realiza una búsqueda dentro de la tabla respectiva a cada rol y analiza si el email y el MD5 de la clave ingresados por el usuario coinciden con alguno de la tabla en la que se busca. De ser así retorna la información correspondiente, sino no retorna nada.

Triggers

RecipeRelations Trigger

El trigger Recipe Relations se encuentra aplicado y se dispara después de cada actualización sobre la tabla *RECETA*. En el caso específico en el que el estado de la instancia modificada en la tabla fue cambiado a *"INACTIVO" el trigger procede a* eliminar de la tabla *PRODUCTO_RECETA* todas las referencias de productos hacia la instancia de receta que se actualizó.

MD5 Trigger

El trigger MD5 se encuentra aplicado y se dispara después de cada inserción y actualización sobre la tabla *USUARIO*. Consiste en tomar la clave ingresada como parámetro por el usuario y aplicarle el algoritmo de reducción criptográfico *MD5*, inmediatamente se almacena el valor de la clave encriptada en el atributo clave de la tabla para la instancia o fila respectiva, lo cuál permite una mayor seguridad y confidencialidad de la clave para cada uno de los usuarios.

Vistas

VistaPacientesPorNutricionista

Esta vista muestra los pacientes que están asociados a un nutricionista y la información necesaria para suplir los requisitos de varias peticiones. Esta vista evita la repetición de sentencias join cada vez que se necesita la información de esta relación entre pacientes y nutricionistas.

VistaNutricionista

Esta vista muestra a los nutricionistas con atributos que están asociados a la petición de registro de cobro, presenta toda la información de los nutricionistas y por cada uno de estos, el número de pacientes asociados. De esta forma se evita la repetición de la sentencia para el conteo de la cantidad de pacientes.

VistaProductosPlan

Esta vista muestra detalladamente los productos asociados a un plan por tiempo de comida ordenados por el id del plan, permitiendo ver ordenadamente los productos que tiene cada tiempo de comida en un plan. Esta vista es útil ya que otra vez evita la repetición de sentencias join en la funcionalidad de gestión de plan y seguimiento paciente.

VistaRecetaProductos

Esta vista muestra detalladamente los productos asociados a una receta evitando la repetición de sentencias join cada vez que se consulta los productos contenidos en una receta.

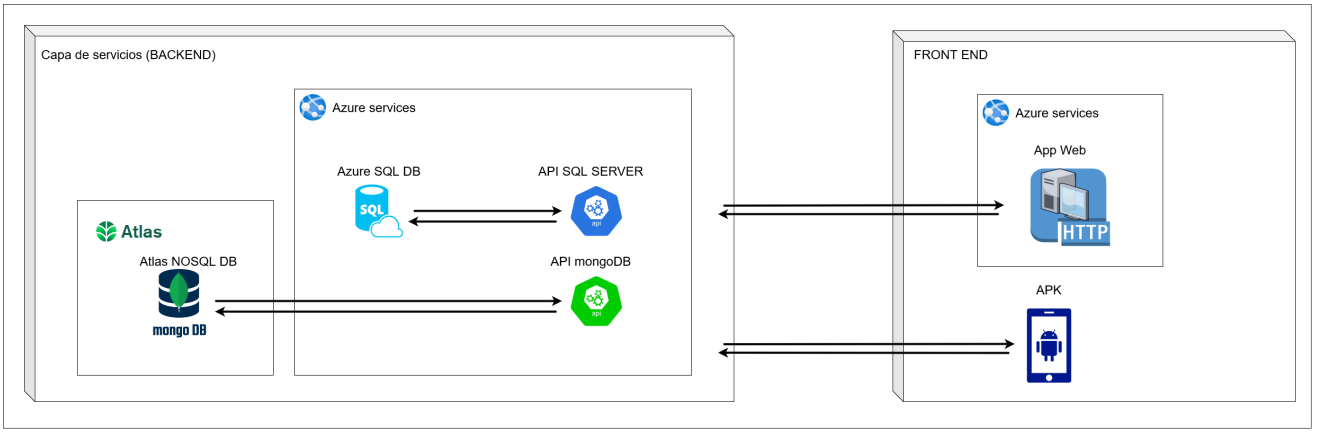
Arquitectura

Descripción

En la capa de servicios se utilizó Azure para desplegar la base de datos SQL y las dos APIs necesarias. Para desplegar la base de datos NOSQL en mongoDB se utilizó Atlas. Ambas APIs se comunican con las bases de datos e intercambian la información de interés para los clientes.

Para desplegar la aplicación web se utilizó Azure y la aplicación móvil simplemente se comunica desde la aplicación desarrollada para el cliente. Ambas aplicaciones de usuario se comunican con las APIs mediante el protocolo HTTP haciendo uso de las solicitudes GET, POST, PUT y DELETE.

Diagrama



Problemas Conocidos

Tamaño de interfaz

Cuando se está en la interfaz de administrador, si se minimiza el tamaño de la interfaz de vista, los cuadros de input para los distintos tipos de ítems modificables se sobreponen a los cuadros con la información de los ítems.

Repetición de productos en los tiempos de comida

Si se agregan productos repetidos en los diferentes tiempos de comida, al momento de realizar el registro diario, los productos repetidos se guardaban en solo uno de los tiempos de comida

Agregar una receta en el consumo diario

Si se agrega dos veces una misma receta al momento de registrar el consumo diario, los productos que conforman esta receta aparecerán repetidos.

Problemas Encontrados

PUT sobre una tupla con valor ID igual a cero

Se encontró un error al intentar realizar un PUT a una entidad que contaba con un ID autogenerado y que tomaba como valor el 0. Al finalizar el protocolo request, se obtenía un valor de que el proceso se realizó de forma exitosa, no obstante la entidad no recibía ningún cambio.

Al realizar pruebas contundentes, curiosamente se observó que solo ocurría con las tuplas de id igual a cero, así que la solución fue omitir el cero para los ids y empezar todos desde el uno, luego de este cambio no hubo problema alguno.

Log in API

Existía un error curioso relacionado al login, el cual consistía en que pasada cierta cantidad de tiempo las URLs para el inicio de sesión dejaban de funcionar sin justificación alguna. Al realizar pruebas exhaustivas se llegó a la conclusión de que era un error en el código, puesto que al realizar una petición específica de eliminar una receta, se eliminaba el stored procedure log in.

El problema era debido a que se tenían algunos triggers y stored procedures dentro de un mismo archivo; por causa de la falta de un GO después del END al final del trigger Recipe Relations se estaba contemplando la eliminación del stored procedure log in, si existía, razón por la cual cada vez que se ejecutaba el trigger ya no se podía volver a iniciar sesión.

Esto se solucionó agregando un GO después del END, para poder encapsular correctamente el trigger y excluir el retiro del stored procedure log in.

Cronograma

Semana	Área	Actividad	Duracion Estimada [h]	Responsable	Fecha de entrega
1	Web App	Selección de plantilla Angular-HTML5-Bootstrap	6	Alejandro Chavarria	23/10/21
		Análisis profundo de la funcionalidad de la página web	6	Todos	25/10/21
		Planteamiento del diseño general de las vistas y el flujo de uso	6	Todos	24/10/21
		Creacion del logo y seleccion de paleta de color	3	Alejandro Chavarria	24/10/21
		Planeamiento de comunicación con API	6	Todos	25/10/21
	API	Acordar las versiones de instalación de los paquetes .NET.	0.5	Todos	22/10/21
		Investigar sobre la conexión del API con bases de datos en MongoDB.	5	Adrián Araya Shakime Richards	25/10/21
		Establecer los URLS y protocolos CRUD necesarios para cada API.	6	Todos	25/10/21
		Investigar sobre la conexión del API con la base de datos SQL Server y cómo realizar consultas de triggers, stored procedures y vistas.	4	Adrián Araya Shakime Richards	25/10/21
	DB	Organizar la información requerida y convertirla en entidades con sus respectivos atributos y relaciones.	9	Todos	22/10/21
		Investigación de la herramienta SQL y la implementación de triggers, stored procedures y vistas.	4	Adrián Araya Shakime Richards	25/10/21
		Investigación de la herramienta MongoDB y el funcionamiento de las bases de datos no relacionales.	4	Adrián Araya Shakime Richards	25/10/21
		Implementar el modelo relacional utilizando el diseño conceptual escogido.	2	Todos	23/10/21
	Mobile App	Planteamiento del diseño general de las vistas de la aplicación	8	Sebastián Mora	24/10/21
		Planteamiento de la comunicación con el API	8	Sebastián Mora	25/10/21
2	Web App	Páginas de login y registro	12	Alejandro Chavarria	28/10/21
		Admin: Aprobación de productos	12	Sebastián Mora Godínez	29/10/21
		Gestión de productos/platillos	24	Alejandro Chavarria	30/10/21
		Cliente: Gestión de recetas	6	Alejandro Chavarria	31/10/21
		Nutricionista: Búsqueda y asociación de pacientes	12	Sebastián Mora	31/10/21
	API	Implementar esqueleto de la API utilizando el diagrama relacional.	8	Adrián Araya	30/10/21
		Asegurar una conexión del esqueleto con la DB en SQL.	4	Adrián Araya Shakime Richards	31/10/21
		Implementar parte de las operaciones CRUD en la API.	6	Adrián Araya	1/11/21
	DB	Implementar tablas y relaciones de la base de datos utilizando el diagrama relacional.	5	Shakime Richards	01/11/21
		Implementar algunos stored procedures, triggers y vistas.	7	Shakime Richards	1/11/21
	Mobile App	Implementación de la vista de Login	5	Sebastián Mora Godínez	25/09/21
		Implementación de la vista de sección de registro y su conexión al API	8	Sebastián Mora Godínez	26/09/21
		Implementación de la vista de gestión de recetas y su conexión al API	8	Sebastián Mora Godínez	27/09/21

3	**Web App**	Nutricionista: Gestion de planes	6	Alejandro Chavarria	5/11/21
		Admin: Reporte de cobro	12	Alejandro Chavarria	6/11/21
		Nutricionista: Asignación de planes	12	Alejandro Chavarria	6/11/21
		Registro diario de consumo (plan/calendario)	48	Alejandro Chavarria	7/11/21
		Cliente: Registro de medidas	6	Alejandro Chavarria	5/11/21
	API	Finalizar con las operaciones CRUD faltantes de la API.	4	Shakime Richards	05/11/21
		Implementar el API para la base de datos de MongoDB y establecer conexión.	9	Adrián Araya Shakime Richards	07/11/21
		Discutir bugs, tareas faltantes y verificar documentación interna del código.	1	Adrián Araya Shakime Richards	08/11/21
	DB	Finalizar los stored procedures, triggers y vistas faltantes.	6	Shakime Richards	05/11/21
		Implementar la base de datos en MongoDB para el foro entre nutricionistas y sus clientes.	5	Adrián Araya Shakime Richards	07/11/21
		Comunicación al equipo los logros, bugs y funcionalidad entre APIs y bases de datos.	3	Adrián Araya Shakime Richards	08/11/21
	Mobile App	Testeo de las aplicación móvil y corrección de errores	24	Sebastián Mora	06/11/21

4	**Web App**	Cliente: Reporte de avance	24	Alejandro Chavarria	12/11/21
		Nutricionista: Seguimiento del paciente	24	Alejandro Chavarria	12/11/21
		Corrección de errores	12	Alejandro Chavarria	13/11/21
		Limpieza y documentación de código.	12	Alejandro Chavarria	13//11/21
	API	Implementación de las restricciones de cada protocolo de request	10	Adrián Araya Shakime Richards	12/11/21
		Depuración del código y corrección de errores.	12	Adrián Araya Shakime Richards	13//11/21
		Limpieza y documentación de código.	12	Adrián Araya Shakime Richards	13//11/21
	DB	Pruebas finales del script de población.	12	TODOS	13//11/21
		Limpieza y documentación de código.	6	Shakime Richards	13//11/21
	Mobile App	Limpieza de código y documentación del código	6	Sebastián Mora	13/11/21

Minutas

18 de octubre

- 7:00 pm: Se realiza la lectura de la especificación y se discute la posible distribución del trabajo. Asimismo se asignan parejas para el desarrollo de los dos modelos conceptuales, los cuales serán revisados en la próxima reunión.
  Se crean las plantillas de los archivos referentes a la documentación y el repositorio, por otra parte se definen del anexo, la introducción, las reglas, los roles y la descripción del aprendizaje continuo.
  Finalmente, se agenda la próxima reunión el 21 de octubre.

21 de octubre

- 7:00 pm: Se realiza la lectura de la especificación nuevamente, se discute e implementa la primera propuesta de diagrama conceptual y se inicia la preparación del cronograma. Además, se finalizan las metas y la descripción del análisis del problema desde el punto de vista ingenieril. Finalmente, se agenda la próxima reunión el 23 de octubre.

23 de octubre

- 2:30 pm: Se realiza la segunda propuesta de modelo conceptual y se escoge como la definitiva, asimismo se realiza el respectivo mapeo al modelo relacional. Se continua con la partición y asignación del cronograma. Además, se finaliza la justificación de la propuesta escogida.

7 de noviembre

- 8:30 pm: Se realiza una reunión de seguimiento de cara a la última semana de desarrollo del proyecto .

15 de noviembre

- 9:00 pm: Se realizan pruebas finales de ambas aplicaciones y también se finaliza la documentación externa.

Bitácora

Miembro: Adrián Araya

22 de octubre

Realicé mi parte del cronograma.

3 de noviembre

Terminé los request para Producto, validé la existencia de un barcode único y creé todos parte de los modelos junto con Shakime.

6 de noviembre

Implementé el registro de usuarios.

8 de noviembre

Realicé la gestión de planes, seguimiento de paciente, el reporte de avance y la vista de productos por plan.

11 de noviembre

Realicé el API y la base de datos de MongoDB para el foro junto con Shakime.

12 de noviembre

Agregué más población al script de población.

15 de noviembre

Realicé documentación externa.

Miembro: Alejandro Chavarría

19 de octubre

Seleccioné la paleta de colores y diseñé templates para el logo del proyecto.

21 de octubre

Hice mi parte del cronograma.

23 de octubre

Redacté la justificación de la propuesta seleccionada.

2 de noviembre

Realice completamente y conecte con el api la vista que permite registrar las medidas del usuario.

También incluí el navbar con todos los componentes hechos, así como crear los componentes de las páginas que hacían falta.

Arregle el cors del api.

3 de noviembre

Implemente la vista de historial de medidas y agregue una gráfica para el despliegue de los datos.

4 de noviembre

Agregue botones a la gráfica para poder visualizar distintas medidas.

6 de noviembre

Realice la conexión del diagrama con el API. Además intenté implementar múltiples calendarios en angula pero no logré que ninguno funcionara.

7 de noviembre

Realicé todos los componentes necesarios y el planteamiento de los querys para la poder hacer el despliegue de los datos de consumo diario y planes.

8 de noviembre

Logre implementar full calendar en angular. Reconsiderar los querys necesarios para el despliegue de los datos de consumo diario y planes.

10 de noviembre

Realicé el login de administrador. Además implementa la generación de reportes de cobro en la vista de admin.

11 de noviembre

Implemente la generación de reporte de medidas en el historial de medias en la vista de cliente. Además estuve trabajando en la integración del calendario para la vista de consumo diario

13 de noviembre

Continúe trabajando en el calendario. Se logró el control de los eventos, como consumo y plan.

14 de noviembre

Se termina la implementación de calendario como vista de consumo y plan. Además se utiliza el calendario como vista de seguimiento del paciente. Se crea la vista de asignación de planes. Se arregla el cors de mongo y se corrigen muchos bugs.

Miembro: Sebastián Mora

22 de octubre

Realicé mi parte del cronograma.

24 de octubre

Se comienza a plantear y diseñar las vistas de la app móvil, así como la estructura para la comunicación con el API basándose en la experiencia de proyectos pasados.

25 de octubre

Se realiza la vista de registro de clientes y nutricionista en la página web.

26 de octubre

Se comienza la vista de aprobación de productos en la página web.

27 de octubre

Se termina la vista de aprobación de productos.

28 de octubre

El día de hoy terminaron los detalles de las vistas de aprobación de productos. Se dá por terminada esta vista.

29 de octubre

El día de hoy se trabaja en la vista de gestión de recetas. Se comienza a realizar los componentes necesarios. Se da por terminada la vista.

30 de octubre

El día comenzó y se terminó la vista de asignación de clientes. Además se realizaron cambios de diseños en diferentes vistas. Se comienza la vista de gestión de planes

31 de octubre

El día de hoy se corrigen varios errores en las vistas hechas hasta el momento, además continúa con la vista de gestión de planes.

1 de noviembre

El día de hoy termina la vista de gestión de planes. Además corrigen errores en diferentes vistas.

2 de noviembre

Se comienza a agregar los request al API para comenzar realizar la conexión con el front-end.

3 de noviembre

Se realizan las conexiones con el API a las vistas hechas hasta el momento. Se realizan diferentes pruebas para verificar que todo esté funcionando correctamente y se corrigen los errores que aparecen.

4 de noviembre

El día de hoy se corrigen varios errores en las vistas hechas hasta el momento, además continúa con la vista de gestión de planes.

5 de noviembre

Se continúan corrigiendo errores errores de las vistas hechas hasta el momento. Además se terminan las corrección al API que faltaban.

6 de noviembre

El día de hoy se comienza a implementar la app móvil. Se comienza a crear la estructura de las diferentes vistas.

7 de noviembre

Se continúan corrigiendo errores errores de las vistas hechas hasta el momento. Además se terminan las correcciones al API que faltaban.

8 de noviembre

Se terminan las primeras vistas de la app móvil y se comienza a realizar las conexión al API para la app móvil. Además, se realiza la conexión al API para los planes en la página web.

10 de noviembre

El día de hoy se corrigen errores y se siguen realizando conexiones al API conforme están disponibles. Se sigue trabajando en el app móvil y las vistas de gestión de recetas y registro de consumo diario.

11 de noviembre

El día de hoy se continúan realizando conexiones al API.

13 de noviembre

Se termina la aplicación de android y se termina la app. Se realiza las conexiones al API faltantes

14 de noviembre

Se afinan detalles de la aplicación móvil y la página web. Se corrigen errores y se realizan pruebas.

15 de noviembre

Realicé documentación externa e interna.

Miembro: Shakime Richards

21 de octubre

Realicé la descripción del análisis del problema desde el punto de vista ingenieril y mi parte del cronograma.

30 de octubre

Realicé la conexión entre API y MSSQL.

1 de noviembre

Realicé algunos de los request para Cliente junto con Adrián.

2 de noviembre

Terminé los request para Cliente, validé la existencia de un email único y creé todos parte de los modelos junto con Adrian.

6 de noviembre

Implementé el Log In, la encriptación MD5 para las claves de usuarios y el trigger MD5.

7 de noviembre

Implementé el stored procedure para la gestión de recetas y el trigger Recipe Relation.

8 de noviembre

Realicé el reporte de cobro y las vistas de nutricionistas, productos por receta y pacientes por nutricionista.

11 de noviembre

Agregué los atributos derivados faltantes de IMC por paciente y calorías por plan.

15 de noviembre

Realicé documentación externa.

Conclusiones

- La mayoría de los sistemas de reporte necesitan de NuGet packages disponibles en las bibliotecas de .NET de Visual Studio.
- Las bases de datos relacionales son más robustas y permiten relacionar una o más entidades mediante un identificador y junto el lenguaje de consultas SQL permite la manipulación compleja de los datos.
- Haciendo uso de Stored Procedures, Triggers y Vistas es posible implementar gran parte de la lógica de la capa de servicios en la base de datos, esto agiliza la tarea ya que el lenguaje SQL es más corto, sencillo y provee una gran cantidad de sentencias para el manejo de los datos.
- Las bases de datos no relacionales pueden manejar grandes volúmenes de datos con técnicas más flexibles y poseen gran escalabilidad.
- Es posible utilizar servicios en la nube para publicar la capa de servicios, tanto el API como la base de datos y así poder hacer uso de los servicios desde Internet, el proveedor Azure ofrece una prueba gratuita para estudiantes.
- También es posible publicar aplicaciones web en Azure para hacerlo público y consumirlo desde Internet.
- El entorno de desarrollo de Android para desarrollar una aplicación móvil permite crear aplicaciones móviles de manera sencilla y eficiente.
- Existen gran cantidad de plantillas de código abierto en angular bootstrap. Permiten facilitar un diseño rápido y estético de una página web.

Recomendaciones

- Es recomendable que al desarrollar un API en .NET, utilizar Visual Studio para tener acceso a mayor funcionalidad como sistemas de reporte.

- Se recomienda utilizar bases de datos no relacionales para la implementación de aplicaciones en las que se procesan grandes volúmenes de datos y que requieren estructuras flexibles, por ejemplo redes sociales, foros.

- Es favorable el uso servidores web gratuitos para hospedar APIs, esto permite un desarrollo en paralelo de una aplicación web que lo requiera, sin la necesidad de instalar los ambientes necesarios para ejecutar el API.

- Se recomienda utilizar el entorno de desarrollo Android para desarrollar aplicaciones móviles por las diferentes facilidades y funcionalidades que ofrece.

- Es recomendable utilizar plantillas preconstruidas de angular bootstrap para generar rápidamente aplicaciones con interfaces llamativas.

Bibliografía

[1] "Create a single database - Azure SQL Database", 2021. [Online]. Available: https://docs.microsoft.com/en-us/azure/azure-sql/database/single-database-create-quickstart?tabs=azure-portal. [Accessed: 10- Nov- 2021].

[2] C. Libre, “Creación de xml Factura Electrónica,” 2018. [Online]. Available: https://github.com/CRLibre/API_Hacienda/wiki/Creaci%C3%B3n-de-xml-Factura-Electr%C3%B3nica [Accessed: 9- Oct- 2021].

[3] "Documentation | Android Developers", Android Developers, 2021. [Online]. Available: https://developer.android.com/docs. [Accessed: 17- Oct- 2021].