Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Bases de Datos - CE3101

Profesor:
Marco Rivera Meneses

Documentación Externa

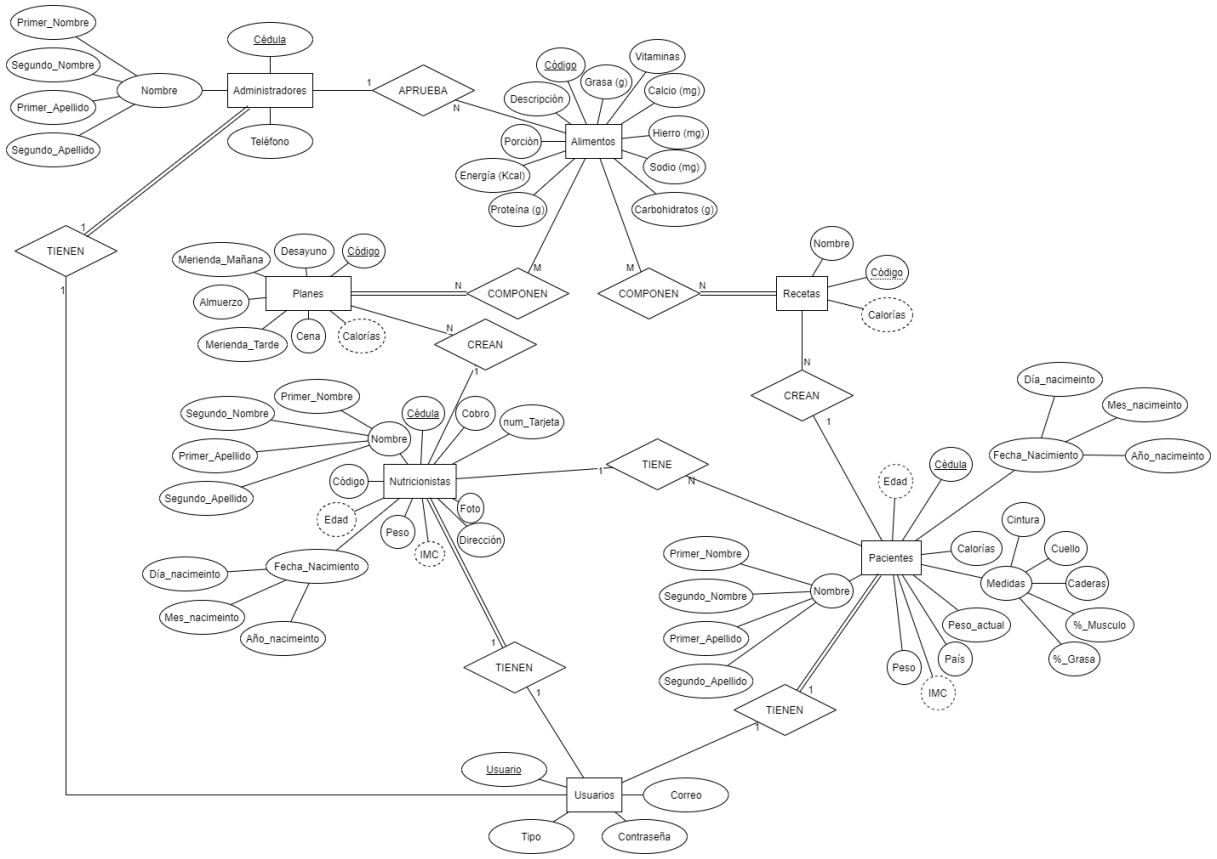
Estudiantes:
Brandon Gomez Gomez
Erick Madrigal Zavala
Marco Rivera Serrano
Kevin Rodriguez Lanuza

Grupo:
1

II Semestre

2021

Modelo Conceptual:



Modelo Relacional:

1. Mapeo de tipos de entidades fuertes

Para cada entidad fuerte construimos una relación que contenga los atributos simples, en caso de los atributos compuestos, solo ponemos los atributos simples que los componen.

Estas son las relaciones resultantes.

Relación resultante	Llave primaria	Atributos
ADMINISTRADORES	Cedula_admin	Cedula_admin Pnombre_admin Snombre_admin Papellido_admin Sapellido_admin Telefono_admin
ALIMENTOS	Codigo_alimento	Codigo_alimento Descripcion_alimento Porcion_alimento Grasa_alimentos Vitaminas_alimentos Calcio_alimentos Hierro_alimentos Sodio_alimentos Carbohidratos_alimentos Energía_alimentos Proteína_alimentos
PLANES	Codigo_planes	Codigo_planes Desayuno_planes Merienta_mañana_planes Almuerzo_planes Merienta_tarde_planes Cena_planes
RECETAS	Codigo_recetas	Codigo_recetas Nombre_recetas
NUTRICIONISTAS	Cedula_nutri	Cedula_nutri Pnombre_Nutri Snombre_Nutri Papellido_Nutri Sapellido_Nutri Día_na_nutri Mes_na_nutri Año_na_nutri Peso_nutri Descripción_nutri Num_tarjeta_nutri Cobro_nutri Foto_nutri Código_nutri
PACIENTES	Cedula_pac	Cedula_pac Pnombre_pac Snombre_pac Papellido_pac Sapellido_pac Día_na_Pac Mes_na_Pac Año_na_pac Peso_pac Peso_actual_pac Calorías_pac País_pac Medida_cintura_pac Medida_cuello_pac Medida_caderas_pac Medida_muslo_pac

		Medida_grasa_pac
USUARIOS	Usuario	Usuario Contraseña Correo_usuario Tipo_usuario

2. Mapeo de tipos de entidades débiles

En el modelo conceptual no se definieron entidades débiles.

3. Mapeo de tipos de asociaciones binarias 1:1

Primero identificamos las relaciones de este tipo, en nuestro caso solo hay 3: Nutricionista-Usuario ,Paciente-Usuario y Adminsitrador-Usuario.

Utilizamos referencia cruzada, donde hacemos una nueva relación, incluimos como llave foránea las llaves primarias de las entidades relacionadas, y se suman los atributos propios de la relación.

Relación	Llaves foráneas	atributos
NUTRICIONISTAS_USUARIO	Cedula_nutri Usuario	Cedula_nutri Usuario
PACIENTES_USUARIO	Cedula_pac Usuario	Cedula_nutri Usuario
ADMINISTRADORES_USUARIO	Cedula_nutri Usuario	Cedula_nutri Usuario

4. Mapeo de tipos de asociaciones binarias 1:N

Primero identificamos las relaciones de este tipo:

- Administradores-Alimentos
- Nutricionistas-Planes
- Pacientes-Recetas
- Nutricionistas-Pacientes

Relaciones 1:N	
Lado 1	Lado N
Administradores	Alimentos
Nutricionistas	Planes
Pacientes	Recetas
Nutricionistas	Pacientes

Incluimos como llave foránea en el lado N de la relación, la llave primaria del lado 1, no creamos una nueva relación, solo se hace la agregación en la relación existente.

Relación resultante	Nueva llave foranea	Atributos
ALIMENTOS	Cedula_admin	Codigo_alimento Descripcion_alimento Porcion_alimento Grasa_alimentos Vitaminas_alimentos Calcio_alimentos Hierro_alimentos Sodio_alimentos Carbohidratos_alimentos Energía_alimentos Proteína_alimentos Cedula_admin
PLANES	Cedula_nutri	Codigo_planes Desayuno_planes Merienta_mañana_planes Almuerzo_planes Merienta_tarde_planes Cena_planes Cedula_nutri
RECETAS	Cedula_pac	Codigo_recetas Nombre_recetas Cedula_pac
PACIENTES	Cedula_nutri	Cedula_pac Pnombre_pac Snombre_pac Papellido_pac Sapellido_pac Día_na_Pac Mes_na_Pac Año_na_pac Peso_pac Peso_actual_pac Calorías_pac País_pac Medida_cintura_pac Medida_cuello_pac Medida_caderas_pac Medida_muslo_pac Medida_grasa_pac Cedula_nutri

5. Mapeo de tipos de asociaciones binarias N:M

Para cada asociación de este tipo hacemos una relación intermedia, y la llave primaria de ambas relaciones asociadas componen la llave primaria de esta nueva relación.
Para nuestro modelo de entidad-relación, tenemos dos asociaciones de este tipo, que es entre Planesy Alimentos, y Recetas y Alimentos.
Haciendo referencia cruzada

Relación	Llaves foráneas	atributos
PLANES_ALIMENTOS	Codigo_planes Codigo_recetas	Codigo_planes Codigo_recetas

RECETAS_ALIMENTOS	Cedula_admin Codigo_recetas	Cedula_admin Codigo_recetas

6. Mapeo de atributos multivaluados

Para nuestro modelo de entidad-relación no hay definidos atributos multivaluados.

Descripción de las estructuras de datos desarrolladas (Tablas)

Se tienen un total de doce entidades, las cuales son:

admins :

- id_admin: es la llave primaria de la tabla y es la cual identifica a cada administrador, es de tipo int.
- first_name_admin: es el primer nombre del administrador, es de tipo text.
- second_name_admin: es el segundo nombre del administrador, es de tipo text.
- first_last_name_admin: es el primer apellido del administrador, es de tipo text.
- second_last_name_admin: es el segundo apellido del administrador, es de tipo text.
- phone_admin: número de teléfono del administrador, es de tipo int.

foods :

- id_food : es la llave primaria de la tabla y es la cual identifica a cada alimento, es de tipo int.
- description_food : es la descripción de la comida, es de tipo text.
- portion_food : es la cantidad de porciones del alimento, es de tipo int.
- fat_food : cantidad de grasa del alimento, es de tipo float.
- vitamins_food : cantidad de vitaminas del alimento, es de tipo float.
- calcium_food : cantidad de calcio del alimento, es de tipo float.
- iron_food : cantidad de hierro del alimento, es de tipo float.
- sodium_food : cantidad de sodio del alimento, es de tipo float.
- carbs_food : cantidad de carbohidratos del alimento, es de tipo float.
- energy_food : cantidad de energía del alimento, es de tipo float.
- protein_food : cantidad de proteina del alimento, es de tipo float.
- food_state : es el estado del alimento, es decir, aceptado o no, es de tipo int.

plans :

- id_plan : es la llave primaria de la tabla e identifica a cada plan, es de tipo int.
- breakfast_plan : el desayuno del plan, es de tipo text.
- morning_snack_plan :la merienda de la mañana del plan, es de tipo text.

- lunch_plan : el almuerzo del plan, es de tipo text.
- afternoon_snack_plan :la merienda de la tarde del plan, es de tipo text.
- diner_plan : la cena del plan, es de tipo text.
- id_nutritionist_plan : es la llave primaria de la tabla e identifica a cada plan con un nutricionista específico, es de tipo int.

recipes :

- id_recipe : es la llave primaria de la tabla, que identifica a cada receta, es de tipo int.
- name_recipe : es el nombre de la receta, es de tipo text.
- patient_id_recipe : es la llave foránea que asocia a cada receta con un paciente específico, es de tipo int..

nutritionists :

- id_nutritionist : es la llave primaria de la tabla e identifica cada nutricionista, es de tipo int.
- first_name__nutritionist : es el primer nombre de nutricionista, es de tipo text.
- second_name__nutritionist : es el segundo nombre de nutricionista, es de tipo text.
- first_last_name__nutritionist : es el primer apellido de nutricionista, es de tipo text.
- second_last_name__nutritionist : es el segundo apellido de nutricionista, es de tipo text.
- birth_date_nutritionist : fecha de nacimiento de nutrisionista, es de tipo DATE .
- weight_nutritionist : peso del nutricionista, es de tipo float.
- imc_nutritionist : peso del nutricionista, es de tipo float.
- code_nutritionist : peso del nutricionista, es de tipo int.
- pfp_nutritionist : fotografía del nutricionista, es de tipo text.
- card_nutritionist : número de tarjeta del nutricionista, es de tipo int.
- payment_nutritionist : forma de pago del nutricionista, es de tipo int.
- id_dirdirection_nutritionist : identificador del nutricionista, es de tipo tex.

patients :

- id_patient : es la llave primaria de la tabla e identifica cada paciente, es de tipo int.
- first_name_patient : es el primer nombre de paciente, es de tipo text.
- second_name_patient : es el segundo nombre de paciente, es de tipo text.
- first_last_name_patient : es el primer apellido del paciente, es de tipo text.
- second_last_name_patient : es el segundo apellido del paciente, es de tipo text.
- birth_date_patient : fecha de nacimiento de paciente, es de tipo DATE .
- weight_patient : peso del paciente, es de tipo float.

- actual_weight_patient : peso actual del paciente, es de tipo float.
- imc_patient : índice de masa corporal del paciente, es de tipo float.
- calories_patient : calorías del paciente, es de tipo float.
- country_patient : país del paciente, es de tipo text.
- waist_patient : medición de la cintura del paciente, es de tipo float.
- neck_patient : medición cuello de la cintura del paciente, es de tipo float.
- hip_patient : medición cadera de la cintura del paciente, es de tipo float.
- thigh_patient : medición del muslo del paciente, es de tipo float.
- fat_patient : grasa del paciente, es de tipo float.
- id_nutritionist_patient : llave foránea que enlaza los nutricionistas con los pacientes, es de tipo int.

users :

- id_user :es el identificador del usuario, es de tipo int.
- username : es el usuario único para ingresar a la aplicación, es de tipo text.
- email : es el correo que usa el usuario, es de tipo varchar.
- password : es la contraseña, es de tipo text.
- user_type : es un número que identifica el tipo de usuario, es de tipo int.

nutritionists_users:

- id_nutritionist :llave foránea que identifica a los nutricionistas, es de tipo int.
- id_user : llave foránea que identifica a los usuarios, es de tipo int.

patients_users:

- id_patient :llave foránea que identifica a los pacientes, es de tipo int.
- id_user : llave foránea que identifica a los usuarios, es de tipo int.

admins_users:

- id_admin :llave foránea que identifica a los administradores, es de tipo int.
- id_user : llave foránea que identifica a los usuarios, es de tipo int.

plans_foods:

- id_plan :es el identificador del plan, es de tipo int.
- id_food : es el identificador para los alimentos, es de tipo int.

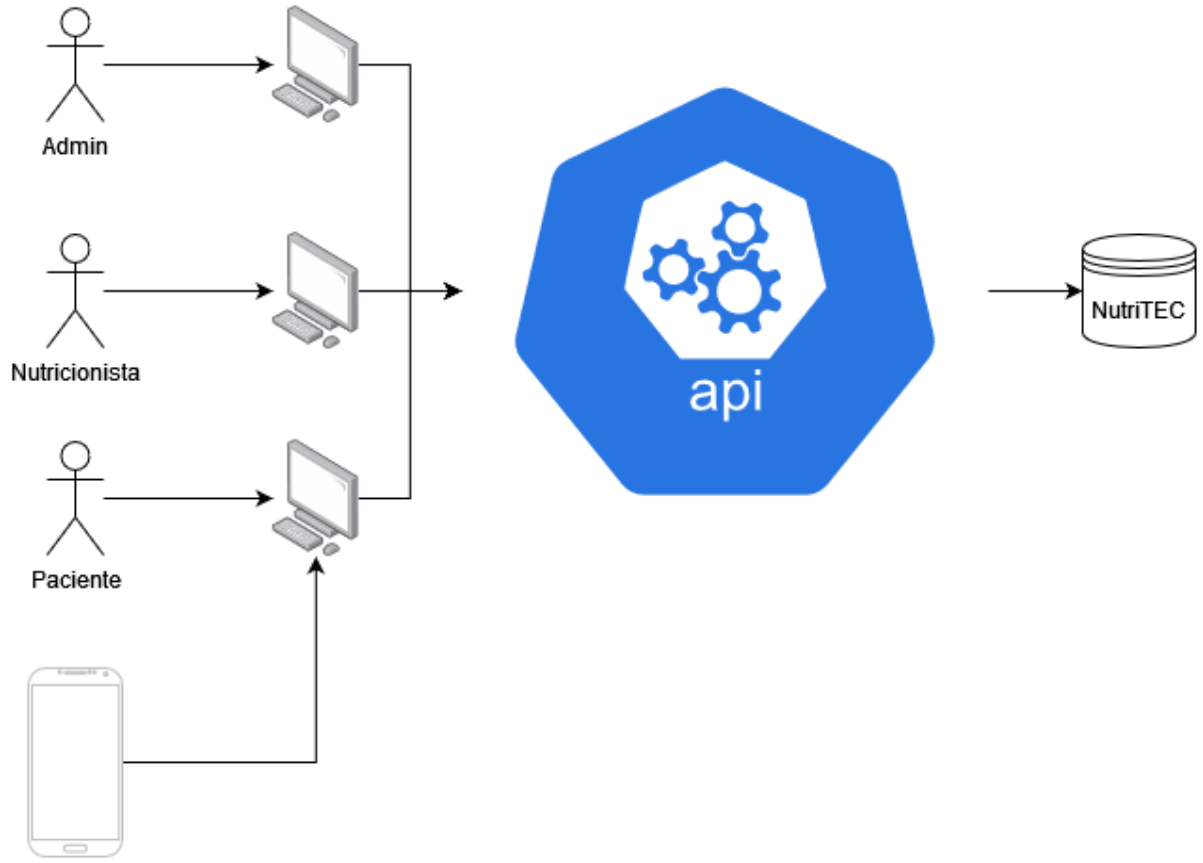
recipes_foods:

- id_recipe :es el identificador de una receta, es de tipo int.

- id_food : es el identificador para los alimentos, es de tipo int.

Descripción de la arquitectura desarrollada

Como se tiene previsto, se debe tener el API funcionando. Cuando un cliente o administrador utiliza la aplicación, la aplicación realiza la conexión que necesite al API y este utiliza los queries necesarios para obtener la información desde la base de datos. El siguiente diagrama explica la funcionalidad de la aplicación con sus componentes:



Problemas conocidos

A nivel de API:

- Dado que no se pudieron implementar triggers, la manipulación externa de los datos puede inferir en el proyecto. Es decir, no existe una verificación entre usuarios y esto permite que puedan realizar acciones no autorizadas.

A nivel de la aplicación móvil:

- No se logró enviar los datos a la api para añadir las recetas y los consumos diarios por lo que se crea no se ve reflejado en la base de datos. Este problema es resultado de retrasos en el desarrollo del app.

A nivel de la aplicación web:

- No se pudo implementar la generación de reportes, se decidió enfocar el tiempo en otras funcionalidades.

- No se logró generar una factura para el cliente que contuviera toda la información necesaria en ella debido a que no se podía obtener cierta información de los gets utilizando typescript, pero dicha información sí aparece en el API.
- No pudimos implementar la funcionalidad de búsqueda y asociación de pacientes, seguimiento de paciente y registro de consumo diario, pues, no se logró obtener la función completa entre las el back y el front end.

Problemas encontrados

A nivel de API:

- No se podía hacer inserción de si se recibían otro tipo de datos diferentes a los de la base. Para solucionar este error, se utilizaron diferentes queries buscando el id del o los strings que se reciben, luego, se confirma si estos queries contienen algo o están vacíos y luego, se realiza la inserción a la base de datos.
- No se podían obtener nombres de otras tablas, solo sus ids. Una manera de solucionar esto fue creando otro modelo que contenga toda la información que se quería mostrar. Finalmente, se realizaron queries entre las tablas que se quería obtener dicha información.

A nivel de la aplicación móvil:

- A la hora de entrar en las ventanas de recetas y consumo diario no se podía cargar automáticamente el spinner ya que esto producía que la aplicación crasheara. Por la experiencia adquirida en trabajos anteriores se recurrió a poner un botón que hace la llamada a la función de carga del spinner para evitar el cierre de la aplicación.

Reuniones

- 25 de octubre del 2021

Primera reunión del equipo. En este día, el grupo discutió el proyecto y se asignaron las tareas para cada miembro del equipo. Asimismo, se establecieron algunas de las rutas principales que son necesarias para el funcionamiento de la aplicación.

- 1 de noviembre del 2021

Segunda reunión del equipo. Para esta fecha se tienen adelantadas algunas plantillas para ser utilizadas en el desarrollo de la aplicación. Algunas de estas fueron, la base de datos, api y front end.

- 8 de noviembre del 2021

Tercera reunión del equipo. Para este día, se lograron algunas conexiones con el front end para realizar algunos métodos básicos (get, post, update, delete) y establecer nuevos modelos según lo que necesite el front end. En esta reunión se tomaron en cuenta todos aquellos puntos restantes para la solución del proyecto y trabajar en ellos.

- 15 de noviembre del 2021

Reunión entre Kevin, Brandon y Marco. Resolución de algunos choques entre el front end y el back end. Se probaron más rutas compuestas de inserción utilizando datos desde el front end. Se establecieron nuevos modelos para ser mostrados en el front end y se entregó el proyecto.

Actividades realizadas (Bitácora)

Brandon:

- 30/10/2021
  Creación de navbar para el home, la vista nutricionista, y la vista administrador.
- 02/11/2021
  Creación de la vista administrador y nutricionista.
- 04/11/2021
  Creación de gestión de productos.
- 06/11/2021
  Creación de la vista de generación de planes.
- 08/11/2021
  Creación de la vista de asignación de planes.
- 09/11/2021
  Creación de la vista registro de productos.
- 11/11/2021
  Creación de la vista lista de pacientes.
- 13/11/2021
  Comunicación del front end y el api.

Erick:

- 30/10/2021
  Se crea la interfaz que se utilizara para crear el login de la aplicación móvil
- 31/10/2021
  Se crea la interfaz con la que se registrará el consumo diario desde la aplicación móvil
- 31/10/2021
  Se crea la interfaz desde la cual se generarán las recetas que el usuario quiera desde la app móvil
- 2/11/2021
  Se crea la lógica necesaria para validar los datos que ingrese el usuario para el login y la respectiva acción en caso de que sean correctos
- 3/11/2021
  Se implementa la lógica para mostrar los alimentos que se tienen almacenados y la lógica para mostrar los que se escojan y tomar sus respectivos datos y agregarlos al consumo diario
- 4/11/2021
  Se implementa la lógica para mostrar los alimentos que se tienen almacenados y la lógica para mostrar los que se escojan y tomar sus respectivos datos y agregarlos a las recetas
- 10/11/2021

Se establece la conección de la app móvil con el api para recibir datos

- 11/11/2021-14/11/2021

  Se realiza una adaptación de la aplicación para que solicite los datos pertinentes al api y que a su vez los maneje de la manera que se necesita

Marco:

- 25/10/21
  Creación de un API básico (CRUD) para comida, nutricionista y admin
- 01/11/21
  Creación de un API básico (CRUD) para planes, recetas y usuarios.
- 05/11/21
- Creación de un API básico (CRUD) para pacientes.
  Creación de las llaves foráneas y adición de estas mismas en la base de datos
- 09/11/21
  Creación de métodos para poder trabajar las llaves foráneas desde el API
- 13/11/21
  Solución al error CORS dentro del API.
  Se agregaron nuevas tablas para algunas referencias.
- 14/11/21
  Solución a algunos errores con respecto al front end.

Kevin:

- 30/10/21
  Creación de tres páginas de registro.
- 01/11/21
  Creación de la página del login.
- 03/11/21
  Creación de navbar para registro.
- 04/11/21
  Creación de navbar para cliente.
- 06/11/21
  Creación de registro de productos.
- 05/11/21
  Creación de registro de medidas.
- 08/11/21
  Creación de registro de recetas.
- 09/11/21
  Creación de registro de avances.
- 12/11/21
  Creación de registro de consumo diario.

Conclusiones

Las bases de datos son muy versátiles para el manejo de información masiva. Además, lograr las relaciones entre ellas permite mantener un control adecuado de la información. Sin embargo, manejar dicha información necesita su cuidado dado que borrar información es sencillo si se tiene el query adecuado y no se cuentan con los triggers o seguridad necesaria.

Lograr que la interfaz de usuario sea amigable con este mismo, permite que el cliente, nutricionista o el administrador no se pierda o confunda a la hora de navegar por la aplicación. Esto permite una experiencia agradable y que este pueda manipular la información con mucha facilidad. Sin embargo, si a este se le presenta una aplicación en la cual no pueda entender con facilidad dónde se encuentran las páginas o información, la experiencia no será gratificante y podría manipular la información erróneamente.

La implementación de aplicaciones para dispositivos móviles permite que los usuarios tengan formas de acceso al servicio que se desea brindar ya que el usuario no depende de un ordenador para poder acceder al contenido de la misma y puede estar seguro de que dicha aplicación funciona de una mejor manera que una página web en su dispositivo

Recomendaciones

En época de covid, casi todo se volvió virtual, el manejo de herramientas como CSS, Angular, Bootstrap y HTML5, se vuelven aún más importantes cada día. Recomendamos que el diseño sea minuciosamente realizado previamente, el orden en el desarrollo web es muy importante y es una buena práctica el previo planeamiento del diseño.

Tener una comunicación constante con el front end para tener en cuenta todos las rutas para el API, qué tipos de parámetros se requieren o qué tipo de respuesta se espera. Por otra parte, si existen varios métodos, pero con diferentes parámetros de entrada, se necesita saber cual ruta debe ser llamada para evitar choques con la base de datos.

Bibliografía

Developers, G. (n.d.). *Android Developers*. obtenido de: https://developer.android.com/docs?hl=es-419

Código Estudiante. (1 de diciembre del 2020). Crear una Web Api - Rest sencillo CRUD con C# + SQL Server [Archivo de Video]. YouTube. https://www.youtube.com/watch?v=-rhIFSDgFjk&t=1782s

Erkec, E. (18 de Noviembre de 2019). *SQL Variables: Basics and usage*. Obtenido de SQLShack: https://www.sqlshack.com/sql-variables-basics-and-usage/

Microsoft. (09 de Noviemnbre de 2021). *Tutorial: Deploy an ASP.NET app to Azure with Azure SQL Database*. Obtenido de Microsoft: Tutorial: Deploy an ASP.NET app to Azure with Azure SQL Database

w3schools. (s.f). *SQL FOREIGN KEY Constraint*. Obtenido de w3schools: https://www.w3schools.com/sql/sql_foreignkey.asp

Diagrama de clases

https://github.com/SlimeVRS/NutriTEC/blob/main/Documentation/Class%20Diagram.jpg