Instituto Tecnológico de Costa Rica

Bases De Datos

II Semestre 2021

Carmen Valentina Araya Chacon.

Harold Espinoza Matarrita.

Armando Fallas Garro.

Fatima Leiva Chinchilla.

Profesor:

Marco Rivera Meneses

Documentación Técnica

Mapeo **3**

Tablas Desarrolladas **3**

Store procedures **6**

Vistas **10**

Triggers **11**

Arquitectura Desarrollada **12**

Minutas **13**

Bitácora de Actividades **13**

Bitácora de Actividades 13

Problemas encontrados **15**

Conclusiones **15**

Recomendaciones **15**

Bibliografía **15**

Mapeo

1) Mapeo de entidades fuertes

Todas las entidades presentes en el modelo conceptual son fuertes las cuales son: Consumo_Diario, Medida, Paciente, Plan, Producto, Receta, Usuario, Rol.

2) Mapeo de entidades débiles

No existen entidades débiles en el modelo por lo que este paso se omite

3) Mapeo de tipos de asociaciones binarias 1:1

No se encontró ninguna relación de este tipo en el modelo, por lo que se omite este paso

4) Mapeo de tipos de asociaciones binarias 1:N

Se asigna como Foreign Key de Medida la Primary Key de Paciente, la cual es cédula.

Se asigna como Foreign Key de Receta el primary key de paciente, la cual es cédula.

Se asigna como Foreign Key de Paciente el Primary Key de nutricionista, el cual es Código de nutricionista

Se asigna como Foreign Key de Consumo Diario la Primary Key de Paciente, la cual es cédula.

Se asigna como Foreign Key de Plan la Primary Key de Nutricionista, el cuál es código nutricionista.

Se asigna como Foreign Key de Plan la Primary Key de Paciente, la cual es cédula.

5) Mapeo de tipos de asociaciones binarias N:M
Se realizó una tabla intermedia llamada Producto por Plan donde se guarda el primary key de plan (nombre) y el primary key(código de barras) de producto.

Se realizó una tabla intermedia Producto por receta donde se guarda el primary key de Producto (código de barras).

Se realizó una tabla intermedia llamada Plan por paciente donde se guarda el primary key de plan (nombre) y el primary key de paciente (cedula).

6) Mapeo de atributos multivaluados
Se encuentra como atributo multivaluado “vitaminas” dentro de la tabla “producto”
Se encuentra como atributo multivaluado “vitaminas” dentro de la tabla “receta”
Sin embargo dado que este atributo es el mismo, es decir, se encuentra repetido, se toma la decisión de crear una tabla “vitamina” para las vitaminas donde su primary key es el atributo vitaminas que contendrá el tipo de vitamina.
Posteriormente se genera una tabla intermediaria entre “vitamina” y “producto” “vitaminas_por_producto” donde se toma como foreing key el primary key “vitamina” de la tabla “vitamina” y el primary key “codigo_de_barras” de la tabla “producto”. Por otra parte se genera una tabla intermedia entre la tabla “vitamina” y “receta” denominada “vitaminas_por_receta” donde se toma como foreing key el primary key “vitamina” de la tabla “vitamina” y el primary key “nombre” de la tabla “receta”.

Formas normales

Tablas	1FN	2FN	3FN	4FN	5FN	Motivos
Producto por receta	☑	☐	☐	☑	☑	No tiene segunda ni tercera forma ya que no se le asignó una primary key, este error se pasó por alto y no se pudo solucionar por falta de tiempo
Plan por paciente	☑	☑	☑	☑	☑	

Producto por plan	☑	☑	☑	☑	☑	
Vitaminas_por_producto	☑	☑	☑	☑	☑	
Vitamina	☑	☑	☑	☑	☑	
Producto	☑	☑	☑	☑	☑	
Plan	☑	☑	☑	☑	☑	
Usuario	☑	☑	☑	☑	☑	
Receta	☑	☑	☑	☑	☑	
Consumo_Diario	☐	☑	☑	☑	☑	No cumple con la primera forma normal ya que no tiene valores atómicos en algunas de sus columnas. No es posible corregir este error por falta de tiempo
Medida	☑	☑	☑	☑	☑	
Vitaminas_por_recetas	☑	☑	☑	☑	☑	
Rol	☑	☑	☑	☑	☑	

1FN: Casi todas las tablas tienen valores atómicos y cuentan con una llave primaria, por lo que se cumple esta forma, de no ser así se encuentra indicado el porque en la tabla anterior

2FN: No se encuentra una falta a esta forma normal en la mayoría de tablas, de no ser así se indica en la tabla anterior

3FN: No se encuentra una dependencia transitiva en los atributos de las diferentes tablas, a excepción de algunas, las cuales están determinadas en la tabla de arriba

4FN: Todas las tablas cumplen la cuarta forma normal, pues no hay varios atributos multivaluados que no tengan correlación, por lo que se no se da una falta a la norma

5FN: De igual manera la quinta forma normal, ya que se cumple con la 4FN en todas las tablas, y no existe información redundante que genere una falta a la normalización

Tablas Desarrolladas

A partir del modelo conceptual y relacional se crean las siguientes tablas

Rol: esta tabla contiene lo roles existentes, donde tiene un atributo “nombre” para el nombre de rol el cual es llave primaria, además tiene un atributo “descripción” que almacena el trabajo que realiza el rol

Usuario: esta tabla contiene los usuarios que ingresan a la página ya sea administradores o nutricionistas, sus atributos son “nombre1”, “nombre2”, “apellido1”, “apellido2” estos son para almacenar el nombre completo del usuario, “cedula”, “fecha_de_nacimiento”, “edad”, “email”, “dirección”, “foto” que como indica el nombre almacenan estos datos del usuario, “código_nutricionista” este atributo es la llave primaria que es un código que se asigna a cada nutricionista, “pass” este es la contraseña del usuario, “numero_tarjeta” este será el número de tarjeta donde se harán los cobros, “tipo_cobro” este será la modalidad en la que se realiza el cobro , “rol” este atributo almacena el rol que desempeña el usuario este es una llave foránea hacia “nombre” de la tabla “Rol”.

Paciente: esta tabla contiene los pacientes que se registran en NutriTec, sus atributos son “nombre1”, “nombre2”, “apellido1”, “apellido2” estos son para almacenar el nombre completo del paciente, “cedula” este atributo se utiliza como llave primaria, “fecha_de_nacimiento”, “edad”, “email”, “pais”, que como indican almacenan estos datos, “peso”, “imc”, son datos de estado físico del paciente, “estado” este atributo refiere a si es un paciente activo de la página, “pass” este refiere a la contraseña del paciente y por último “codigo_nutricionista” este es una llave foranea a “codigo_nutricionista” de la tabla “usuario” que corresponde al nutricionista encargado del paciente.

ConsumoDiaro: esta tabla contiene el consumo diario que realiza un paciente, este tiene como llave primaria “fecha” que sería la fecha del consumo y “cedula_paciente” que es una llave foranea a “cedula” de la tabla “paciente”, “desayuno”, “merienda_manana”, “almuerzo”, “merienda_tarde”, “cena” estos son los tiempos de consumo del paciente

Medida: esta tabla contiene las medidas registradas por un paciente, sus atributos son “cadera”, “cintura”, “cuello”, “peso”, “porcentaje_musculo”, “porcentaja_grasa”, estas son las diferentes medidas del estado físico de un paciente, “fecha_ingreso” es la fecha en la que

se registra la medida este actúa como llave primaria junto a “cedula_paciente” que es a su vez una llave foranea a “cedula” de la tabla “paciente”

Plan_Alimentacion: esta tabla contiene los planes creados por los nutricionistas, consta de “desayuno”, “merienda_manana”, “almuerzo”, “cena”, “merienda_tarde” que son los tiempos de consumo, “nombre” este es el nombre del plan que actúa como llave primaria, “nutricionista” que es el nutricionista que creó el plan, es una llave foránea a “codigo_nutricionista” de la tabla “usuario”.

Plan_por_paciente: este es una tabla intermediaria para relacionar los planes con los pacientes, consta de “cedula” que es una llave foránea a “cedula” de la tabla “paciente”, “nombre_plan” que es una llave foránea a “nombre” de la tabla “plan_alimentacion”, éstos son llave primaria de la tabla, tambien contiene “periodo_inicio”, “periodo_fin” que es la fecha inicial y final del plan.

Producto: esta tabla contiene los productos para el consumo, compuesto por “energia”, “grasa”, “sodio”, “carbohidrato”, “proteina”, “calcio”, “hierro”, “tamano_porcion” estos son los datos nutricionales del producto, “descripcion” es una descripción del contenido del producto, “estado” este refiere al estado de aprobación en que se encuentra el producto, “codigo_de_barras” este es el código de barras del producto que funciona como llave primaria.

Producto_por_plan: esta tabla contiene la relación de los productos que se asignan a un plan, “codigo_barras” es el código del producto asignado es una llave foranea de “codigo _de_barras” de la tabla “producto”, “nombre_plan” que es el plan al que se asigna y es una llave foránea a “nombre” de la tabla “Plan_alimentacion”. Ambos atributos están asignados como llave primaria.

Producto_por_receta: esta tabla contiene la relación de los productos que se asignan a una receta, “codigo_barras” es el código del producto asignado es una llave foranea de “codigo _de_barras” de la tabla “producto”, “nombre_receta” que es la receta al que se asigna y es una llave foránea a “nombre” de la tabla “Receta”.

Receta: esta tabla contiene las recetas para el consumo, compuesto por “grasa”, “sodio”, “carbohidrato”, “proteina”, “calcio”, “hierro”, “tamano_porcion” estos son los datos nutricionales de la receta, “nombre” este es el nombre asignado a la receta que actúa como

llave primaria, “cedula paciente” esta es la cédula del paciente que crea la receta y es una llave foránea a “cedula” de la tabla “paciente” ,

Vitamina: esta tabla contiene las diferentes vitaminas existentes, formado por “vitamina” que es el tipo de vitamina y actua como llave primaria

Vitaminas_por_producto: esta tabla contiene la relación de vitamina a producto para asignar vitaminas a un producto, contiene “vitamina” es una llave foránea a “vitamina” de la tabla “vitamina”, “codigo_barras” es una llave foranea a “codigo_de_barras” de la tabla “producto”. Ambos actuan como llave primaria.

Vitaminas_por_receta: esta tabla contiene la relación de vitamina a receta para asignar vitaminas a una receta, contiene “vitamina” es una llave foránea a “vitamina” de la tabla “vitamina”, “nombre_receta” es una llave foranea a “nombre” de la tabla “receta”. Ambos actuan como llave primaria.

Store procedures

GetUsuarios: este procedimiento se encarga de obtener un listado de los usuarios existentes con sus atributos, no recibe ningún parámetro de entrada.

GetNuricionistaXcod: este procedimiento obtiene un nutricionista con sus atributos según su código, así recibe como parámetro de entrada @NutriCod VARCHAR(20) con el código del nutricionista deseado

InsertUsuario: este procedimiento se encarga de ingresar un nuevo usuario a la base de datos, recibe como parámetros: @nombre1 VARCHAR(20), @nombre2 VARCHAR(20), @apellido1 VARCHAR(20), @apellido2 VARCHAR(20), @cedula INTEGER, @fecha_de_nacimiento DATE, @edad INTEGER, @codigo_nutricionista VARCHAR(20), @pass VARCHAR(30) (contraseña), @email VARCHAR(40), @numero_tarjeta INTEGER, @tipo_cobro VARCHAR (50),@rol VARCHAR(50), @direccion VARCHAR (80), @foto VARCHAR(80))

GetValidarLogin: Este procedimiento se encarga de validar el email y contraseña del paciente, recibe parámetros: @User VARCHAR(60), @pass VARCHAR(100), @Return int output

DeleteUsuario: este procedimiento se encarga de eliminar un usuario por su código, así recibe como parámetro el código del usuario a eliminar @Codigo VARCHAR(20)

GetValidarLoginRol: Este procedimiento se encarga de validar el inicio de sección ya sea para un nutricionista o administrador, parámetros que recibe: @User VARCHAR(60), @pass VARCHAR(100), @rol VARCHAR(100), @Return int output.

GetPlanesDeNutri: este procedimiento obtiene un listado de todos los planes con sus atributos creado por un nutricionista en específico, recibe como parámetro el código del nutricionista @Codigo VARCHAR(20)

GetPlan: este procedimiento obtiene un plan de alimentación y sus atributos por su nombre, así recibe como parámetro el nombre del plan @nombre VARCHAR(50)

GetNombreRecetasPaciente: Este procedimiento obtiene el nombre de todas las recetas ligadas a un paciente, recibe parametros: @Email VARCHAR(60).

GetProductosPlan: Este procedimiento se encarga de obtener todos los productos que tiene un plan, parámetros que recibe: @Email VARCHAR(60).

GetProductoPorReceta: Este procedimiento obtiene todos los productos de una receta, parámetros que recibe: @NombreReceta VARCHAR(60).

UpdatePlan: este procedimiento actualiza los valores de un plan, recibe como parámetros el nombre del plan a modificar @nombre VARCHAR(50), y los atributos a modificar @desayuno VARCHAR(120), @merienda_manana VARCHAR(120), @almuerzo VARCHAR(120), @cena VARCHAR(120), @merienda_tarde VARCHAR(120), @nutricionista VARCHAR(20)

DeletePlan: este procedimiento elimina un plan de alimentación y sus relaciones existentes, recibe como parámetro el nombre del plan a eliminar @nombre VARCHAR(50)

GetMedidasXCedula: este procedimiento obtiene un listado de las medidas con sus atributos de un paciente, recibe como parámetro de entrada la cédula del paciente a buscar @cedula_paciente INT

InsertMedidas: este procedimiento ingresa unas nuevas medidas para un usuario, así recibe como parámetros la cédula del paciente @cedula_paciente INT, y los atributos de las medidas

que se desean registrar @fecha DATE, @cadera INT, @cintura INT, @cuello INT, @peso INT, @musculo INT, @grasa INT

InsertConsumo: este procedimiento ingresa un nuevo consumo diario para un paciente, así recibe como parámetros la cédula del paciente @cedula_paciente INT, y los atributos del consumo a ingresar @fecha DATE, @almuerzo VARCHAR(250), @cena VARCHAR(250),@desayuno VARCHAR(250), @merienda_manana VARCHAR(250), @merienda_tarde VARCHAR(250), @consumo_calorias INT,

AgregarReceta: Este procedimiento se encarga de realizar un insert de una receta, parámetros que recibe: @Email VARCHAR(100), @NombreReceta VARCHAR(50), @TamanoPorcion INT, @Producto VARCHAR(100).

GetProductos: este procedimiento se encarga de obtener un listado de todos los productos registrados y sus atributos, no recibe parámetros de entrada

InsertPlan: Este procedimiento se encarga de ingresar un nuevo plan de alimentación a la base de datos, recibe como parámetros de entrada los atributos del plan a ingresar @nombre VARCHAR(50), @desayuno VARCHAR(120), @merienda_manana VARCHAR(120), @almuerzo VARCHAR(120), @cena VARCHAR(120), @merienda_tarde VARCHAR(120), @nutricionista VARCHAR(20)

GetProductosXEstado: este procedimiento obtiene un listado de los productos y sus atributos según sea el estado de aprobación del producto, recibe como parámetro el estado del producto que se desea @estado VARCHAR(50)

GetProducto: este procedimiento obtiene un producto y sus atributos por su código de barras, recibe como parámetro de entrada el código del producto buscado @codigo VARCHAR(50)

InsertProducto: este procedimiento ingresa un nuevo producto a la base de datos, recibe como parámetros de entrada los atributos del producto a ingresar @codigo_de_barras VARCHAR(50), @descripcion VARCHAR(500), @tamano_porcion INT, @energia INT, @grasa INT, @sodio INT, carbohidrato INT, @proteina INT, @calcio INT, @estado VARCHAR(50), @hierro INT

AprovacionProducto: este procedimiento cambia el estado de aprobación de un producto por el código de barras, recibe como parámetros de entrada el código del producto y el estado por el que se va a cambiar @codigo VARCHAR(50), @estado VARCHAR(50)

BorrarProdReceta: Este procedimiento se encarga de realizar un delete de un producto de una receta, parámetros que recibe: @NombreReceta VARCHAR(50), @ProductoBorrado VARCHAR(100).

AgregarProdRecetaNuevo: Este procedimiento se encarga de hacer un insert de un nuevo producto en una receta, parámetros que recibe: @NombreReceta VARCHAR(50), @Producto VARCHAR(100), @Porcion INT

BorrarReceta: Este procedimiento se encarga de borrar una receta, parámetros que recibe: @Email VARCHAR(100), @NombreReceta VARCHAR(50).

RegistroDiario: Este procedimiento se encarga de realizar un insert de consumo diario, parámetros que recibe: @Email VARCHAR(50), @Producto VARCHAR(250), @Horario VARCHAR(250).

GetRecetasPaciente: este procedimiento obtiene el nombre de las recetas que posee un paciente, recibe como parámetros de entrada el email del paciente solicitado @Email VARCHAR(50)

GetProductosDePaciente: este procedimiento obtiene la descripción de los productos que se asignaron en el plan de un paciente, recibe como parámetros de entrada el email de paciente que se solicita @Email VARCHAR(50)

PacienteSinNutri: este procedimiento obtiene un listado de pacientes y sus atributos que no tienen un nutricionista asociado, no recibe parámetros de entradas

AsignarNutriAPaciente: este procedimiento se encarga de asociar un nutricionista a un paciente, recibe como parámetros de entrada la cédula del paciente y el código de nutricionista del nutricionista @cedula_paciente INT,@codigo_nutri VARCHAR(20)

DesligarNutricionista: este procedimiento se encarga de desligar el nutricionista que tiene un paciente, recibe como parámetro de entrada la cédula del paciente @cedula_paciente INT

AsignarPlanAPaciente: este procedimiento asocia un plan a un paciente, recibe como parámetros de entrada la cédula del paciente, el nombre del plan a asignar y el intervalo de tiempo del plan @cedula INT ,@nombre_plan VARCHAR(50),@inicio date,@fin date

GetProductosPlanPorNombre: Este procedimiento se encarga de obtener los productos de un plano por su nombre, parámetros que recibe: @Email VARCHAR(60), @nombre VARCHAR(60).

GetProductosPlanPorCodigo: Este procedimiento se encarga de obtener los productos de un plan por su código, recibe parámetros: @Email VARCHAR(60), @codigo VARCHAR(60).

GetNombreRecetasPorNombre: Este procedimiento se encarga de obtener una receta por su nombre, recibe parametros: @Email VARCHAR(60), @Nombre VARCHAR(60).

GetNutriXEmail: este procedimiento obtiene un usuario y sus atributos por su email, recibe como parámetro de entrada el email del usuario buscado @email VARCHAR(40)

GetPacientesDeUnNutri: este procedimiento se encarga de obtener un listado con los pacientes que tiene un nutricionista, recibe como parámetro de entrada el código del nutricionista buscado @nutri VARCHAR(20)

GetConsumosPaciente: este procedimiento obtiene un listado de los consumos diarios de un paciente, recibe como parámetros de entrada la cédula del paciente buscado @cedula INT

GetPacienteXEmail: este procedimiento obtiene un paciente y sus atributos por su email, recibe como parámetro de entrada el email del usuario buscado @email VARCHAR(40)

GetPorcionReceta: Este procedimiento se encarga de obtener la porción de una receta, recibe parámetros: @Email VARCHAR(60), @Nombre VARCHAR(100).

AgregarProductoPlan: Este procedimiento se encarga de agregar un producto a un plan, recibe parámetros: @NombrePlan VARCHAR(50), @Producto VARCHAR(100).

ReporteCobro: este procedimiento genera el reporte de cobro para los nutricionistas, no recibe parámetros de entrada

Vistas

ProductosAprobados: esta vista almacena un listado de los productos que se encuentran en estado de aprobación

ProductosEnEspera: Esta vista almacena un listado de los productos que se encuentran en estado de espera

email_plan_pacientes: Esta vista se encarga de almacenar un listado de los plans junto a los emails de cada uno de los pacientes.

PdtsPacientes:

tablaCobro: esta vista guarda el tipo de cobro, el código de nutricionista, el email, el nombre completo, el número de tarjeta y la cantidad de pacientes que tenga, donde el rol sea ‘nutricionista’ y agrupado por el tipo de cobro

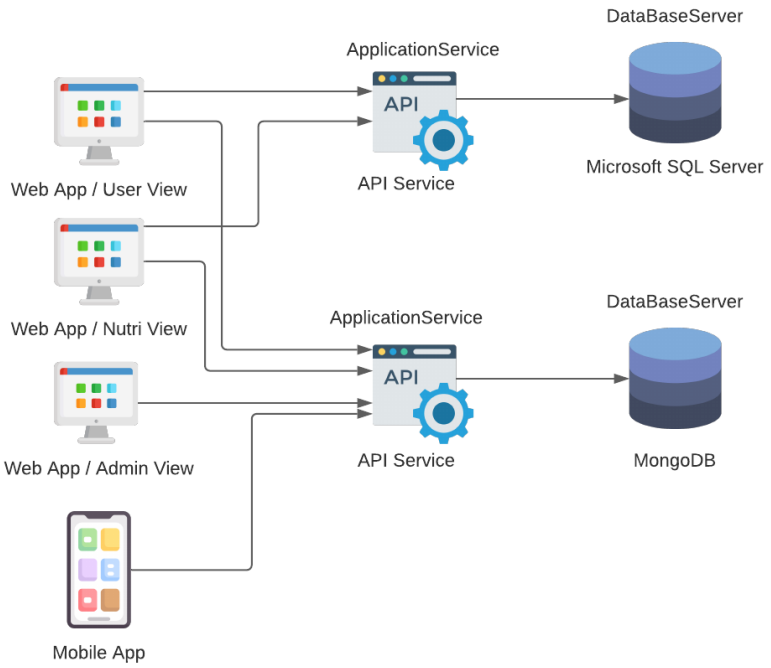
Triggers

TR_DropTable: este trigger impide que se pueda eliminar una tabla completa de la base de datos

TR_validarCantidades: este trigger se encarga de validar que cuando se hace el insert de producto las cantidades nutricionales del producto no sea menores a 0

Arquitectura Desarrollada

Desde el para esta arquitectura se tienen tres aplicaciones Web, la aplicación para la vista Nutricionista, Usuario y Administrador, además existe una aplicación móvil para usuarios y pacientes. Del lado del backend se tienen dos distintas bases de datos, la primera se desarrolló con Microsoft SQL Server y MongoDB, la primera contiene los datos esenciales para el funcionamiento de las tres aplicaciones web y la aplicación móvil. La base de datos MongoDB, solo contiene la información para la funcionalidad del foro de las aplicaciones web User View y Nutri View. Para realizar una correcta comunicación y flujo de datos se tienen dos Rest API para cada una de las bases de datos, estas fueron desarrolladas en C#.



Minutas

Fecha	Asistentes	Canal de comunicación	Tiempo
22/10/2021	Harold, Armando	Discord	1 hora 30
Actividades			
● Lectura del proyecto ● Realizar la primer propuesta del modelo conceptual.			
Acuerdos y/o propuestas			
● Desarrollo en conjunto de las dos Rest Api y base de datos			

Fecha	Asistentes	Canal de comunicación	Tiempo
24/10/2021	Carmen, Fátima	Discord	2 horas
Actividades			
● Lectura del proyecto ● Realizar la segunda propuesta del modelo conceptual.			
Acuerdos y/o propuestas			
● Se propone realizar un mayor desarrollo en conjunto de las aplicación web al leer los requerimientos del proyecto.			

Fecha	Asistentes	Canal de comunicación	Tiempo
25/10/2021	Carmen, Fátima, Harold, Armando	Discord	2 horas
Actividades			
● Selección del modelo conceptual para la solución del problema. ● Terminar documento de Anexo del proyecto. ● División de roles y actividades ● Revisión del cronograma, fijar fechas a las actividades			

Acuerdos y/o propuestas
• Tener una reunión más adelante para realizar los scripts de la base de datos.

Fecha	Asistentes	Canal de comunicación	Tiempo
1/11/2021	Carmen, Fátima, Armando, Harold	Discord	3 horas
Actividades			
• Crear scripts de población • Crear scripts de base de datos			
Acuerdos y/o propuestas			
• Hacer una reunión más adelante para tener un registro del avance de los integrantes.			

Fecha	Asistentes	Canal de comunicación	Tiempo
3/11/2021	Carmen, Fátima, Armando, Harold	Discord	1 hora 30 mi
Actividades			
• Revisión de avance de tareas por persona • Métodos de la API que se necesitan para utilizar en las web app			
Acuerdos y/o propuestas			
• Estar avisando del avance de cada integrante			

Fecha	Asistentes	Canal de comunicación	Tiempo
14/11/2021	Carmen, Fátima, Harold, Armando	Discord	3 horas
Actividades			

• Documentación externa del proyecto
Acuerdos y/o propuestas
• Avanzar la documentación y al final que la encargada de la documentación la suba al respositorio.

Bitácora de Actividades

Bitácora de Actividades

Fecha	Descripción actividades	Asistentes
22/10/21	Se realizó la primer propuesta del modelo conceptual para el proyecto	Armando, Harold
24/10/21	Se realizó la segunda propuesta de modelo conceptual para el proyecto.	Carmen, Fátima
25/10/21	Se realizó un avance del documento de anexo, introducción, plantilla para bitácoras, cronograma para las páginas web y trabajos previos al desarrollo de la solución	Carmen
26/10/21	Se realizó el proyecto inicial en android studio para la app móvil, se dió inicio con la vista del login y parte de la interfaz de menú de inicio	Armando
26/10/21	Se realiza la creación de los proyectos en angular para las tres vistas de las páginas web.	Carmen
27/10/21	Se crea la funcionalidad de registro para las vistas de usuario y pacientes.	Carmen
27/10/21	Se realiza la creación inicial de un API en c#	Harold
28/10/21	Se terminó la vista del menú y se dió inicio con la interfaz del registro diario la cual se logró terminar.	Armando
30/10/21	Se realizó la vista del consumo diario, se elaboró el store procedure del login para el paciente y también el login para nutricionistas y administradores, a partir de estos store procedures se hizo el método para la conexión con el api, por último se hizo la	Armando

	interfaz de conexión del login para la app móvil y se hizo la conexión y validación de credenciales	
31/10/21	Se crea una cuenta estudiantil para ingresar al portal de azure	Harold
31/10/21	Se realiza lo necesario para la gestión de productos en la vista administrador, esto incluye el despliegue de los productos en una tabla y permite realizar la aprobación de productos.	Carmen
31/10/21	Se despliega la base de datos en azure	Harold
1/11/21	Se realizó los store procedures para obtener los nombres de las recetas que tiene el paciente, el store procedure para obtener la información de una receta por su nombre, a partir de estos store procedure se hizo el método de la rest api, por último se realizó la interfaz de conexión para cada método en la app móvil y su debida conexión.	Armando
2/11/21	Se realizaron los elementos necesarios para la visualización de los productos en las vistas nutricionista y usuario, esto para poder visualizar los productos en una tabla y permite además realizar la función de agregar productos.	Carmen
3/11/21	Se realizó el despliegue del API en Azure	Harold
4/11/21	Se realizó el store procedure para agregar una nueva receta en el cúal se realiza el cálculo de carbohidratos y demás atributos, a partir de este se hace el método en la rest api y la conexión con la app móvil.	Armando
4/11/21	Se realizó lo necesario para la gestión de mediciones, esto incluye el ingreso de mediciones de mediciones para cierto día.	Carmen
5/11/21	Se crean los métodos para la gestión de usuarios, productos y planes Con estos se generaron los respectivos store procedure que permitan la gestion	Harold
5/11/21	Se realizó el store procedure para actualizar una receta a partir de los productos borrados o agregados, a partir de este su método en la rest api y la conexión con la app móvil.	Armando
5/11/21	Se crean los login para las tres páginas y una primera conexión con los servicios de la base	Fátima

	de datos.	
5/11/21	Se realizó lo necesario para la gestión de mediciones, esto incluye el ingreso de mediciones de mediciones para cierto día. Esto incluye la búsqueda de productos y asignarlos a los tiempos de comida necesarios.	Carmen
6/11/21	Se implementaron al API los métodos faltantes para asociar pacientes, planes y nutricionistas	Harold
6/11/21	Se realizó el store procedure para agregar los datos del consumo diario, además de los productos y recetas disponibles para agregar, se realizó el método en la rest api y la conexión con la app móvil	Armando
7/11/21	Se genera otro API para la coneccion con Mongobd y se despliega en azure	Harold
7/11/21	Se realiza el despliegue de las aplicaciones en Azure DevOps.	Carmen
8/11/21	Se realizó la búsqueda y asociación de pacientes dentro de la vista de nutricionista, conectándose con el API y obteniendo los usuarios pacientes que no tienen un nutricionista asociado, así como los que se asocian con el nutricionista logueado.	Fátima
10/11/21	Se realizó múltiples pruebas a la app móvil y se modificó errores al momento de cargar productos nuevos en la receta y la debida documentación del código de la app móvil.	Armando
10/11/21	Se realizó la asignación de un plan a un paciente, donde se obtienen un rango de fecha que el nutricionista debe elegir y todos los planes que tiene disponibles y así seleccionar un paciente para asignarle el plan.	Fátima
10/11/21	Se despliega el atlas la base de mongodb y se realiza la coneccion con el API	Harold
11/11/21	Se crean los métodos del API de mongo para gestionar el foro	Harold
11/11/21	Se realizó el seguimiento de un paciente, donde se obtienen todos los usuarios asociados al nutricionista y se despliega todo el avance de ese paciente, todo esto con un servicio que se conecta con la base de datos.	Fátima

12/11/21	Se realizó el store procedure para el cálculo del reporte de cobro	Armando y Harold
12/11/21	Se realiza la creación de los triggers y vistas necesitadas	Armando y Harold
12/11/21	Se realizó un foro en el seguimiento de paciente con la base de datos de Mongo para que hubiese una comunicación entre un paciente y nutricionista.	Fátima
13/11/21	Hice la gestión de recetas, para la vista del usuario y que pueda insertar una nueva receta con productos aprobados. Se realizó el reporte de avance del usuario.	Fátima
14/11/21	Se hizo una vista para desplegar los cobros de los usuarios nutricionistas desde la vista de admin, así como la generación de un pdf con esta información.	Fátima

Problemas encontrados

- A la hora de registrarse un nuevo usuario nutricionista no se podía utilizar el usuario recién creado para loguearse en las vistas a pesar de que se guardaba en la base de datos.
- No se encriptaba una contraseña para un usuario recién logueado.
- No se envía el usuario recién registrado en la vista usuario a la base de datos, es decir, no se crea el nuevo usuario.
- No se presenta una manera de avisar al usuario que se ha realizado una petición

Problemas solucionados

- Para desplegar las páginas web en azure no se podían desplegar debido a que las versiones que se utilizaban no eran compatibles con las web app, esto se solucionó cambiando las versiones a .net y agregando un web.config a cada página, que establecía una configuración necesaria para subir las aplicaciones a azure.
- No se podían hacer peticiones desde las páginas en el localhost debido a problemas de CORS que no permitían hacer peticiones a la red de las API, esto se solucionó al

desplegar las páginas en Azure al agregar la ruta de la página al API que permitiera hacer estas peticiones.

Conclusiones

- El despliegue en Azure permite una facilidad para trabajar en conjunto, dado que todos y todas pueden acceder a los distintos elementos como base de datos y aplicaciones web.
- El uso de Azure permite solucionar problemas como el caso de CORS Policy para las web app, y solucionarlo para todos los contribuidores del proyecto.

Recomendaciones

- Crear casos de usuario a la hora de terminar una vista de la web app, para así conocer qué debilidades tiene la app y saber qué cambios podrían hacerse para mejorar la experiencia de usuario.
- Utilizar Azure para trabajar un proyecto en conjunto y se necesitan compartir los recursos, gracias a su facilidad de uso.

Bibliografía

Angular Material (s.f.) *Angular Material UI component library*. https://material.angular.io/

Retrofit. (s. f.). Retrofit. https://square.github.io/retrofit/

Robles, V(2021) *Master en Javascript: Aprender Js, jQuery, Angular, NodeJs.* https://www.udemy.com/course/master-en-javascript-aprender-js-jquery-angular-nodejs-y-mas/