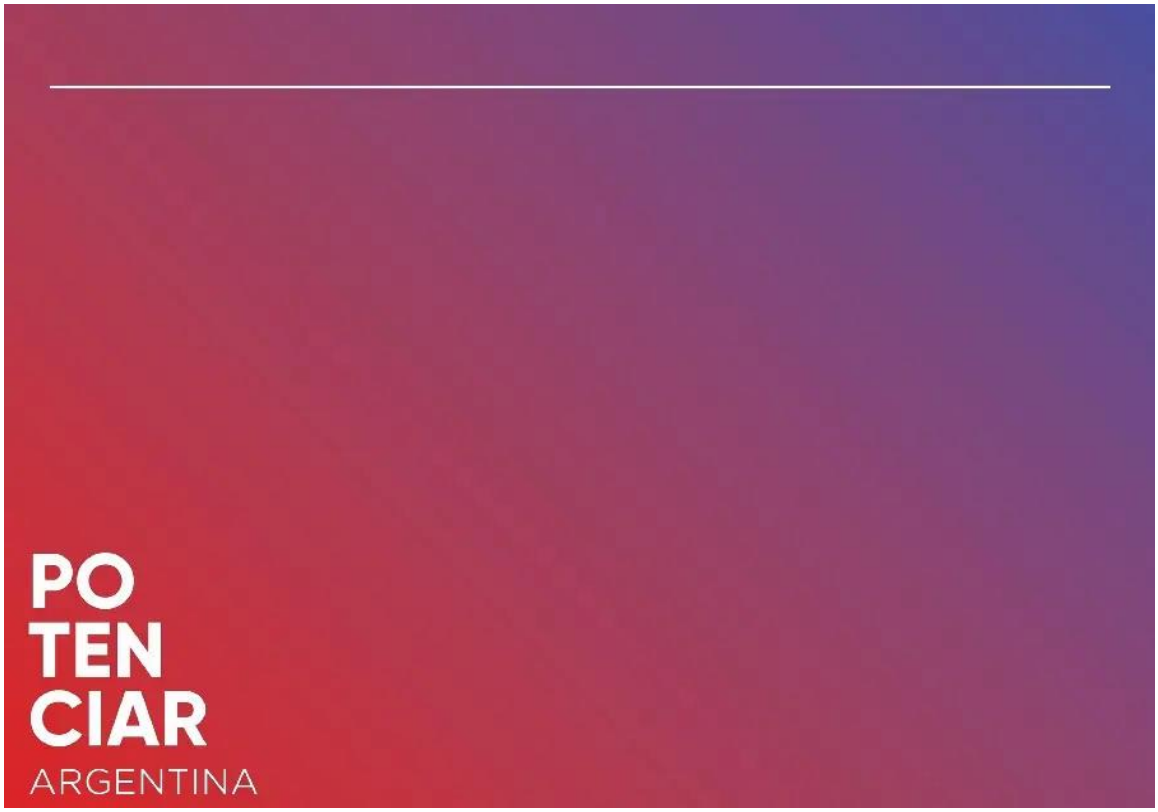

TRABAJO PRÁCTICO FINAL

Potenciar Argentina
FullCoders: curso introductorio
Comisión 28/21613

Septiembre 2023
Proyecto: KcalBalance aplicación para cuantificar balance calórico diario de un adulto
Alumno: Beorlegui Diego Tomas – DNI 30926139
Tutor/a: Gustavo Trinitario

Repositorio GitHub:
https://github.com/tomasomtas/Beorlegui_TPFullCodersFinal.git

Aplicación para calcular balance calórico diario/ KcalBalance

Un personal trainer necesita una aplicación que pueda recibir datos sobre características de sus clientes, como el sexo y la actividad física diaria, ingesta de calorías en cuatro comidas diarias (desayuno, almuerzo, merienda y cena) procesar estos datos, realizar un balance calórico diario (como todo balance, tendrá en consideración egresos e ingresos de calorías, hará la diferencia y dará un resultado al usuario, mostrando las calorías finales, en Kcal) y brindar una conclusión final al cliente (en este caso, los deportistas que entrenan para bajar de peso, mantenerse, o ganar masa muscular). Si el resultado da negativo, el usuario habrá sufrido una pérdida neta de calorías ese día, si da positivo, habrá una ganancia neta de calorías para el usuario, y si da cero, se mantendrá en un estado metabólico llamado basal o de mantenimiento, sin ganancia ni pérdida de peso. En resumen, lo que el cliente necesita (en este caso, un personal trainer, pero podría ser un gimnasio, o una escuela de pilates, o un centro de entrenamiento en artes marciales, por ejemplo) es un contador de calorías diario, que pida ingresar datos, como el sexo, horas de caminata y ejercicio diario, porciones de alimentos en desayuno, almuerzo, merienda y cena, procese estos datos ingresados y los traduzca a calorías (la medida utilizada en análisis de metabolismo digestivo es Kcal o Kilocaloría, y es la que utilizará el programa) procese los datos ingresados, realice un cálculo resultado del balance neto, y arroje resultados negativos si el deportista gastó más calorías de las que ingirió, positivo si gastó menos calorías, o neutro en caso de que el balance dé como resultado un número cero, y termine, una vez realizado el cálculo, dando una conclusión y un consejo al usuario (con texto en una pantalla, sin voz, ni imágenes, solo texto, bastante sencillo).

características y funcionalidades básicas de la aplicación

Para dicho propósito, desarrollaremos una aplicación de escritorio, portable y con una interfaz gráfica amigable, con uso de colores llamativos (por ejemplo, verde para valores positivos y rojo para valores negativos), que calcule primero el gasto calórico diario, pidiendo para este propósito, con un mensaje en pantalla luego de dar la bienvenida al usuario, que este ingrese sexo (ya que los requerimientos calóricos de mantenimiento varían según el sexo, siendo para un hombre adulto de aproximadamente 2500 Kcal y para una mujer, 2000 Kcal) y partiendo de este valor, sumar ingresos calóricos por ingesta, y restar egresos por actividad física (caminata por un lado, y actividad física intensa, cardio y levantamiento de pesas, por otro)

También la ingesta calórica deberá ser ingresada por el usuario, y para facilitar el cálculo del ingreso calórico, el programa separa ingreso por desayuno, almuerzo, merienda y cena. Luego de calcular el gasto calórico, el programa le avisará al usuario que debe ingresar lo ingerido durante el desayuno. En el desayuno, el programa le indica al usuario con un mensaje ya establecido en el programa (que se ejecuta en pantalla una vez calculado el gasto calórico) el equivalente a "porción", siendo una porción, en alimento sólido, lo que entra en la palma de la mano, dos porciones, el doble, y así. También le dará la opción de ingresar proteínas (huevos y tocino en desayuno) carbohidratos (panificados y facturas, porción de torta) porción de azúcar (1 porción equivale a una cucharada) y bebida (una porción equivale a un vaso de 300 ml) porción de frutas, y chocolate. Cada porción tendrá un valor a modo constante en el programa, en Kcal.

Luego, el programa calculará el ingreso calórico en el desayuno, y guardará el valor de esa variable en memoria. Hará lo mismo en el almuerzo, merienda y cena, pidiendo al usuario que ingrese las porciones ingeridas en estas comidas. La merienda será igual al desayudo, en cuanto a la oferta de comida disponible (lo que tradicionalmente se desayuna o merienda, bebida con cucharadas de azúcar, puede ser té o café, lo que cuenta en este caso es solo el azúcar, vaso de leche, pan o facturas o tortas, proteína, por ejemplo, huevos revueltos, porción de frutas, etc. Y el almuerzo, igual que la cena, porción de carbohidratos, por ejemplo, arroz o papas fritas, porción de proteína (carne magra, por ejemplo, un apechuga de pollo) porción de ensalada, pan, vaso de gaseosa, etc.

El programa debería tener una base de datos con constantes ya preestablecidas, ya que por ejemplo el gasto calórico (lo que comúnmente se conoce como "quemar calorías") es más o menos constante en seres humanos adultos y sus valores ya conocidos. Una hora de caminata a paso normal, equivale a "quemar" o gastar alrededor de 300 kcal, y una hora de ejercicio intenso, aproximadamente unas 700 Kcal. También, en la base de datos, una tabla con alimentos comunes y en porciones, como pizza, sándwiches, proteína magra, carbohidratos como porción de arroz, porción de pastas o papas, ensalada, pan, porción de torta, y bebidas, como, por ejemplo, vaso de gaseosa, etc. Y su equivalente por porción, en Kcal. Cabe aclarar, que el programa no busca una precisión exacta en estos cálculos, ya que su uso es para estimaciones aproximadas que servirán a modo de guía o consejo en un ámbito de entrenamiento deportivo sin consideraciones nutricionales médicas o de salud, por lo que su uso estará supeditado al consejo de un entrenador deportivo y no al análisis de un médico y por esta última razón, los datos ingresados como constantes serán aproximados y para nada exactos.

Una vez calculados los ingresos y egresos calóricos diarios, el programa realizará el balance calórico y arrojará un resultado, en Kcal, que podrá ser negativo, positivo o neutro. En base a ese resultado, el programa recomendará aumentar la ingesta de proteína magra y carbohidratos saludables si el resultado es negativo, disminuir la ingesta de carbohidratos y aumentar la de fibra y el ejercicio intenso si es positivo, y mantener los hábitos si da cero, pero recomendando la ingesta de proteína magra si se quiere obtener ganancia de masa muscular.

Utilizaríamos JAVA para desarrollar la aplicación, ya que es un lenguaje de programación ampliamente utilizado para el desarrollo de aplicaciones de escritorio. Podríamos utilizar la biblioteca Swing para crear interfaces de usuario de escritorio y SQLite u otras bases de datos para gestionar la información de alimentos y calorías. Otras opciones serían Python, otro lenguaje popular para aplicaciones de escritorio

Esta es la idea central y resumida de lo que sería la aplicación, y es lo que se acerca más a la idea que podemos plasmar escribiendo pseudocódigo en PSeInt, pero una aplicación de este tipo, desarrollada en JAVA, tendría una complejidad muchísimo mayor, en cuanto a funciones, posibilidades y características de la misma.

La aplicación desarrollada en JAVA para escritorio debería proporcionar a los usuarios las siguientes características:

Gestión de perfil de usuario: Permite a los usuarios crear y administrar perfiles personales, incluyendo detalles como edad, género, peso, altura y niveles de actividad física.

Base de datos de alimentos y calorías: Incluye una amplia base de datos de alimentos y bebidas con información detallada sobre las calorías, macronutrientes y otros nutrientes para facilitar la entrada de datos de alimentos consumidos.

Seguimiento de la ingesta de alimentos: Permite a los usuarios registrar y realizar un seguimiento de los alimentos y bebidas consumidos, incluyendo la cantidad y la hora de consumo.

Búsqueda de alimentos y escaneo de códigos de barras: Ofrece la capacidad de buscar alimentos y bebidas en la base de datos utilizando palabras clave o escanear códigos de barras para agregar alimentos de manera rápida y precisa.

Cálculo de calorías y nutrientes: Realiza cálculos automáticos de calorías y otros nutrientes basados en los alimentos y bebidas registrados por el usuario.

Seguimiento de actividad física: Permite a los usuarios registrar y realizar un seguimiento de su actividad física diaria, incluyendo ejercicios, tiempo dedicado y calorías quemadas.

Sincronización con dispositivos de seguimiento: Integra la capacidad de sincronizar datos con dispositivos de seguimiento de actividad física, como relojes inteligentes o rastreadores de fitness.

Gráficos y estadísticas: Ofrece gráficos y estadísticas visuales que muestran el progreso del usuario en términos de calorías consumidas, calorías quemadas y otros datos relevantes.

Consejos y recomendaciones personalizadas: Proporciona consejos y recomendaciones personalizadas sobre la ingesta de alimentos y el ejercicio físico en función de los objetivos de salud y condición física del usuario.

Registro de metas y objetivos: Permite a los usuarios establecer metas y objetivos de salud, como la pérdida o el aumento de peso, y realiza un seguimiento del progreso hacia esas metas.

Historial y análisis de datos: Almacena un historial completo de datos de alimentos y actividad física para permitir análisis a largo plazo y seguimiento de tendencias.

Notificaciones y recordatorios: Ofrece recordatorios y notificaciones para ayudar a los usuarios a mantener un seguimiento constante de su ingesta calórica y actividad física.

Compatibilidad multiplataforma: Desarrolla la aplicación de manera que sea compatible con múltiples sistemas operativos, como Windows, macOS y Linux, para llegar a la mayor cantidad de usuarios posible.

Exportación e importación de datos: Permite a los usuarios exportar e importar datos para realizar copias de seguridad y sincronización con otras aplicaciones y servicios.

Privacidad y seguridad: Asegura que los datos del usuario estén protegidos y cumplan con las regulaciones de privacidad aplicables.

Como se describe arriba, esta sería la idea más completa de una aplicación para escritorio desarrollada en JAVA.

La aplicación también debería proporcionar a los usuarios los siguientes requisitos funcionales:

Registro de usuario: Los usuarios deben poder crear cuentas personales con información básica, como nombre, edad, género, peso y altura.

Inicio de sesión: Los usuarios deben poder iniciar sesión en sus cuentas utilizando credenciales seguras.

Gestión de perfiles de usuario: Los usuarios deben poder editar y actualizar la información de su perfil, incluyendo cambios en la altura, el peso y otros detalles personales.

Ingreso de alimentos y bebidas: Los usuarios deben poder registrar los alimentos y bebidas que consumen, incluyendo la cantidad y la hora del consumo. Deben poder buscar alimentos en una base de datos o escanear códigos de barras para agregar alimentos de manera rápida.

Cálculo de calorías y nutrientes: La aplicación debe calcular automáticamente las calorías y otros nutrientes basados en los alimentos y bebidas ingresados por el usuario.

Ingreso de actividad física: Los usuarios deben poder registrar su actividad física diaria, incluyendo ejercicios, tiempo dedicado y tipo de actividad.

Sincronización con dispositivos de seguimiento: La aplicación debe ser capaz de sincronizarse con dispositivos de seguimiento de actividad física, como relojes inteligentes o rastreadores de fitness, para importar datos automáticamente.

Seguimiento de objetivos: Los usuarios deben poder establecer metas y objetivos personales, como pérdida de peso, aumento de masa muscular o mantenimiento de peso.

Gráficos y estadísticas: La aplicación debe mostrar gráficos y estadísticas visuales que representen el progreso del usuario en términos de calorías consumidas, calorías quemadas y otros datos relevantes.

Notificaciones y recordatorios: Deben proporcionarse notificaciones y recordatorios para alentar a los usuarios a registrar sus comidas y actividades.

Historial y análisis de datos: Los usuarios deben poder acceder a un historial completo de datos de alimentos y actividad física para realizar análisis a largo plazo y seguimiento de tendencias.

Consejos y recomendaciones: La aplicación debe proporcionar consejos y recomendaciones personalizadas sobre la ingesta de alimentos y el ejercicio físico según los objetivos y la información del usuario.

Exportación e importación de datos: Debe permitir a los usuarios exportar e importar datos para realizar copias de seguridad y sincronización con otras aplicaciones y servicios.

Privacidad y seguridad: Los datos de los usuarios deben estar protegidos y la aplicación debe cumplir con las regulaciones de privacidad aplicables.

Compatibilidad multiplataforma: La aplicación debe ser compatible con múltiples sistemas operativos, como Windows, macOS y Linux, para llegar a la mayor cantidad de usuarios posible.

Ser una aplicación de escritorio, idealmente portable, para PC's o laptops con sistemas operativos Windows, Mac o Linux. No sería óptimo su desempeño en smartphone o tablets, por las siguientes

razones: El usuario deberá cargar algunos datos con paciencia y leyendo atentamente las instrucciones, y necesitará una interfaz de usuario amigable con el ingreso de datos numéricos. Las PC's de escritorio y laptops generalmente tienen pantallas más grandes que los smartphones, esto facilita la visualización de datos detallados, gráficos y tablas, lo que es importante para un seguimiento detallado de calorías y actividad física. También las aplicaciones de escritorio generalmente ofrecen una mejor capacidad para exportar e importar datos, lo que facilita la sincronización con otras herramientas y servicios de seguimiento, y una mayor capacidad de almacenamiento.

Planeamiento y desarrollo del software: Git y GitHub

En este proyecto introductorio, se presenta un diagrama de flujo general, un pseudocódigo que simula el sistema de turnos y la carga de productos. Además, se teoriza sobre el proceso de gestión del proyecto con las metodologías ágiles trabajadas en el curso, Kanban y Scrum.

Inicia el proyecto, presentándose una problemática que el programa informático debe, y puede resolver: Un cliente, en este caso un personal trainer, necesita realizar un balance calórico, y en base a ese balance, dar un consejo. No de un usuario, ni de dos, sino de muchos, tal vez decenas de ellos, y claramente no podría estar rellenando planillas de papel, acumulando pilas de archivos en hojas, de todos sus deportistas, a los que entrena y asesora, por lo que recurre al desarrollador de aplicaciones, y le encarga la aplicación "KcalBalance".

Para iniciar el proyecto, primero escuchamos los requerimientos de nuestro cliente, qué es lo que necesita, cuales son sus urgencias, y qué cosas serían descartadas por éste, por considerarlas no importantes.

Pensamos primero cómo sería el algoritmo básico que resuelva el problema, o sea, que realice el balance calórico por ingresos y egresos, y obteniendo el resultado, dar un consejo o

asesoramiento, para luego presentarlo en pseudocódigo y hacerlo correr en PseInt, probando si de verdad funciona su lógica. Aquí está el algoritmo, paso a paso:

1) Se definen las variables necesarias para el cálculo, como sexo, horasCaminadas, horasEjercicio, gastoMetabolicoBasal, gastoCaminatas, gastoEjercicio, gastoTotal, y balanceCaloricoFinal, junto con valores predefinidos para el gasto metabólico basal y los costos calóricos por hora de caminata y ejercicio.

2) Se solicita al usuario que ingrese su sexo, horas caminadas y horas de ejercicio.

3) Según el sexo ingresado, se asigna el valor adecuado de gastoMetabolicoBasal (ya sea el valor masculino o femenino).

4) Se calcula el gasto calórico por caminatas y ejercicio multiplicando las horas correspondientes por los costos calóricos por hora.

5) Se calcula el gastoTotal sumando el gasto metabólico basal, el gasto por caminatas y el gasto por ejercicio.

6) Se muestra el gasto calórico total al usuario.

7) Se solicita al usuario que ingrese los detalles de su desayuno, incluyendo la cantidad de porciones de diferentes alimentos y bebidas.

8) Se realiza un cálculo de las calorías totales consumidas en el desayuno sumando las calorías de cada componente, multiplicando la cantidad de porciones por las calorías por porción correspondientes.

9) Se muestra el total de calorías consumidas en el desayuno.

10) Se solicita al usuario que ingrese los detalles de su almuerzo, incluyendo la cantidad de porciones de diferentes alimentos y bebidas.

11) Se realiza un cálculo de las calorías totales consumidas en el almuerzo de manera similar al desayuno.

12) Se muestra el total de calorías consumidas en el almuerzo.

13) Se solicita al usuario que ingrese los detalles de su merienda, incluyendo la cantidad de porciones de diferentes alimentos y bebidas.

14) Se realiza un cálculo de las calorías totales consumidas en la merienda de manera similar a los otros momentos del día.

15) Se muestra el total de calorías consumidas en la merienda.

16) Se solicita al usuario que ingrese los detalles de su cena, incluyendo la cantidad de porciones de diferentes alimentos y bebidas.

17) Se realiza un cálculo de las calorías totales consumidas en la cena de manera similar a los otros momentos del día.

18) Se muestra el total de calorías consumidas en la cena.

19) Se calcula el balance calórico final restando el gasto calórico total de las calorías totales consumidas a lo largo del día.

20) Se muestra el balance calórico diario y se proporcionan recomendaciones basadas en su valor:

21) Si el balance es negativo, se sugiere aumentar la ingesta calórica, especialmente de proteínas y carbohidratos saludables.

22) Si el balance es cero, se informa que se mantendrá el peso actual pero no se ganará masa muscular sin aumentar la ingesta de proteínas magras.

23) Si el balance es positivo, se sugiere disminuir la ingesta de carbohidratos, aumentar la ingesta de fibra y proteínas magras, y aumentar el tiempo de actividad física.

Este algoritmo es una implementación detallada del proceso de cálculo del balance calórico diario y proporciona recomendaciones basadas en los resultados. Cada paso se realiza de manera ordenada, y los resultados se presentan de manera clara al usuario.

Luego, se presenta el pseudocódigo en PSeInt, y es en esta instancia en la que comenzamos a trabajar escribiendo pseudocódigo y subiendo a GitHub un repositorio con los avances del mismo. Presentamos el enlace al repositorio de este proyecto en GitHub a continuación: https://github.com/tomasomtas/Beorlegui_TPFullCodersFinal.git.

Dividimos el proyecto en cinco partes (vamos a desarrollar más este punto cuando expliquemos las metodologías ágiles aplicadas al mismo) y en cada parte, hicimos un “commit to main” y luego

un “push origin”. En nuestro ejemplo, usamos para este desarrollo GitHub Desktop, la aplicación de escritorio. Tenemos una cuenta ya abierta en www.github.com, creamos un repositorio con las siguientes características: público, inicializado con un archivo README.TXT en el que explicamos lo que contiene el repositorio en sus carpetas, luego, conectamos a nuestro repositorio, con GitHub desktop, elegimos la opción “Clone a repository” (clonar un repositorio en el programa de escritorio) copiamos la URL en nuestro repositorio remoto, y la pegamos en la opción Repository URL, y creando una carpeta local (o Local Path en GitHub Desktop) en donde se irán guardando nuestros archivos de código. En este casi presentamos el trabajo en pseudocódigo en PSeInt, pero si luego trabajamos en JAVA, sería igualmente el directorio local en donde se guarda nuestro código.

A nivel local, se usaría Git como un controlador de versiones, comparable a la bitácora del capitán de un barco, en donde registra todos los eventos. Registrar los avances en desarrollo del software, las modificaciones en el código, es de mucha importancia cuando se trabaja en equipo, ya que se puede realizar un seguimiento de todos los cambios hechos en el código a lo largo del tiempo. Puedes ver quién hizo qué cambio, cuándo se hizo y por qué se hizo. También permite crear ramas independientes del código principal (rama principal o "master"), esto es útil para desarrollar nuevas características o solucionar problemas sin afectar el código base. Una vez que la nueva característica está lista, se puede fusionar de nuevo en la rama principal. También tiene otras tantas ventajas, como respaldo y restauración (¡uy, se cambió un montón de código y el programa ya no funciona, y no sabemos dónde inició el problema!) Distribución y Copias de Seguridad, Comunidad y Código Abierto, etc.

Continuando con el trabajo en PSeInt, al escribir el pseudocódigo, guardamos los avances y éstos quedan en el archivo guardado en la carpeta a nivel local. Lo más cómodo de GitHub DeskTop, es que te avisa cuando hay un cambio inmediatamente, te da la opción de escribir una descripción, un título del avance, apretamos botón commit to main (que sería main sería principal, “rama principal” en GitHub) push, e inmediatamente se realiza el cambio en el repositorio remoto. La contra del GitHub desktop, es que no es tan configurable como GitHub por comandos, pero para un trabajo sencillo como este, con Pseudocódigo en PSeInt, basta y sobra.

Una vez terminado el pseudocódigo en PSeInt, se sube completamente al repositorio en GitHub, y presentamos aquí lo que sería un diagrama de flujo del proyecto, en formato PNG:

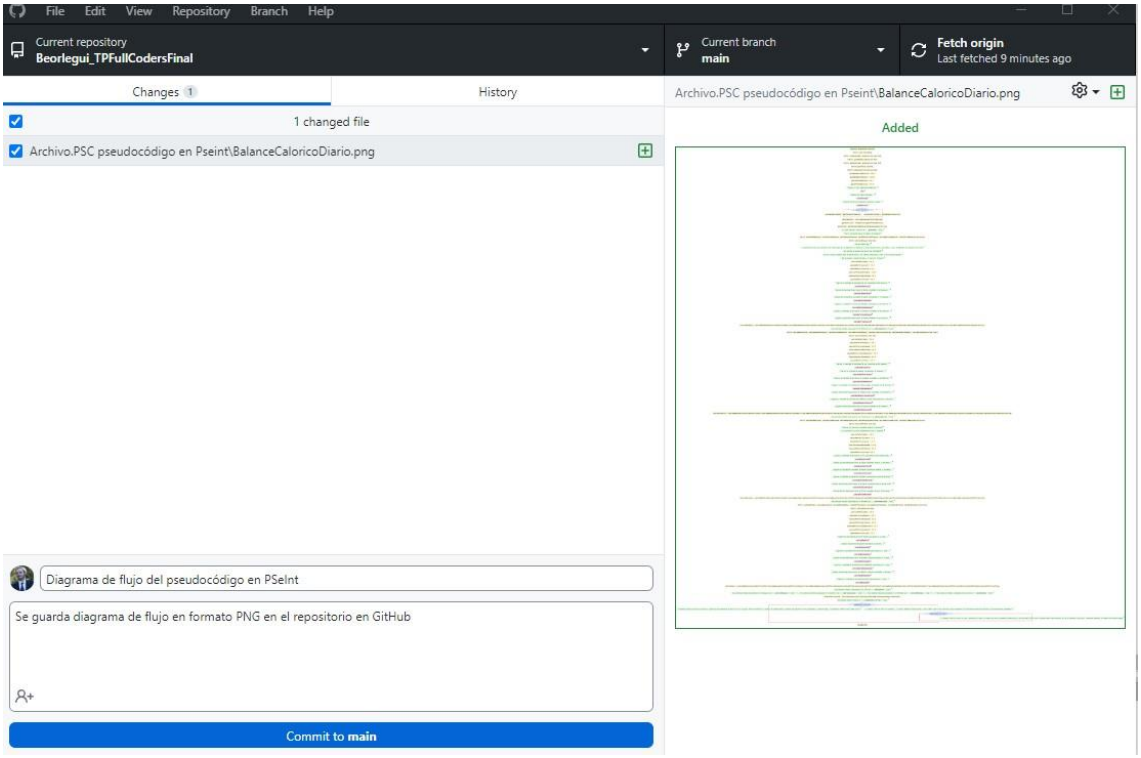

Imagen 1: commit y push de diagrama de flujo del pseudocódigo

El archivo está guardado en el repositorio.

Estas son algunas imágenes del proceso, trabajando con GitHub desktop:

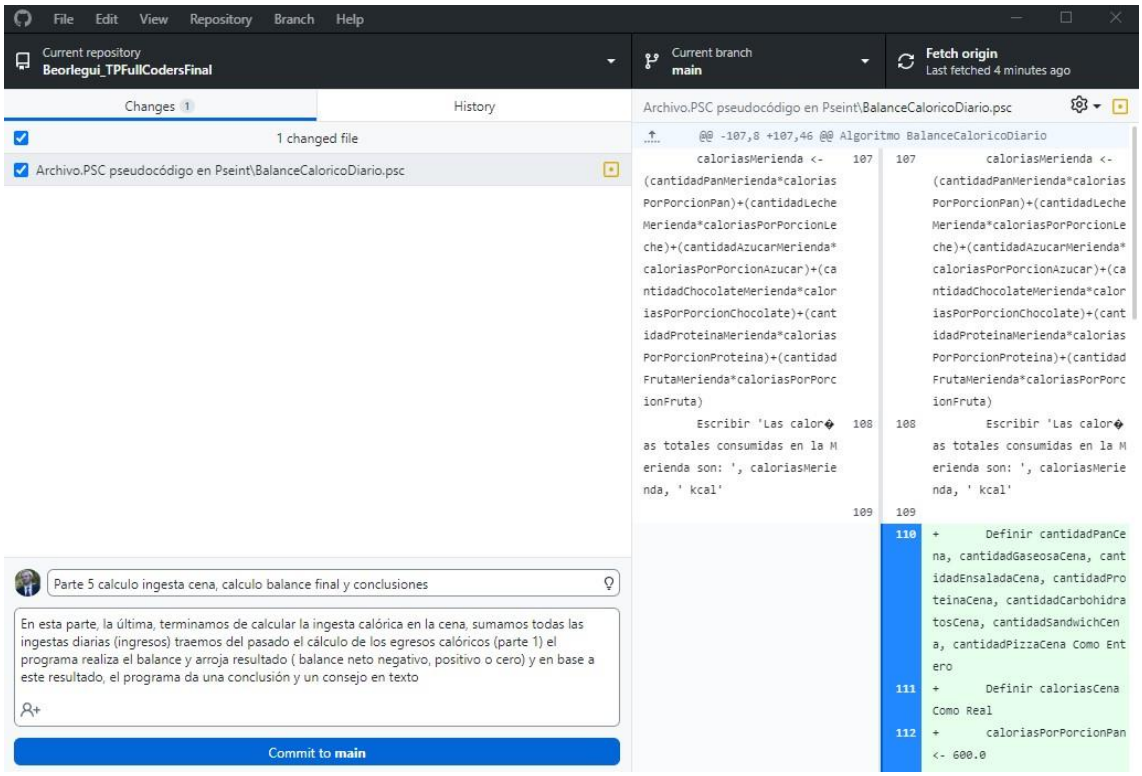

Imagen 2: Ejemplo de como vamos subiendo los progresos al repositorio remoto con GitHub desktop

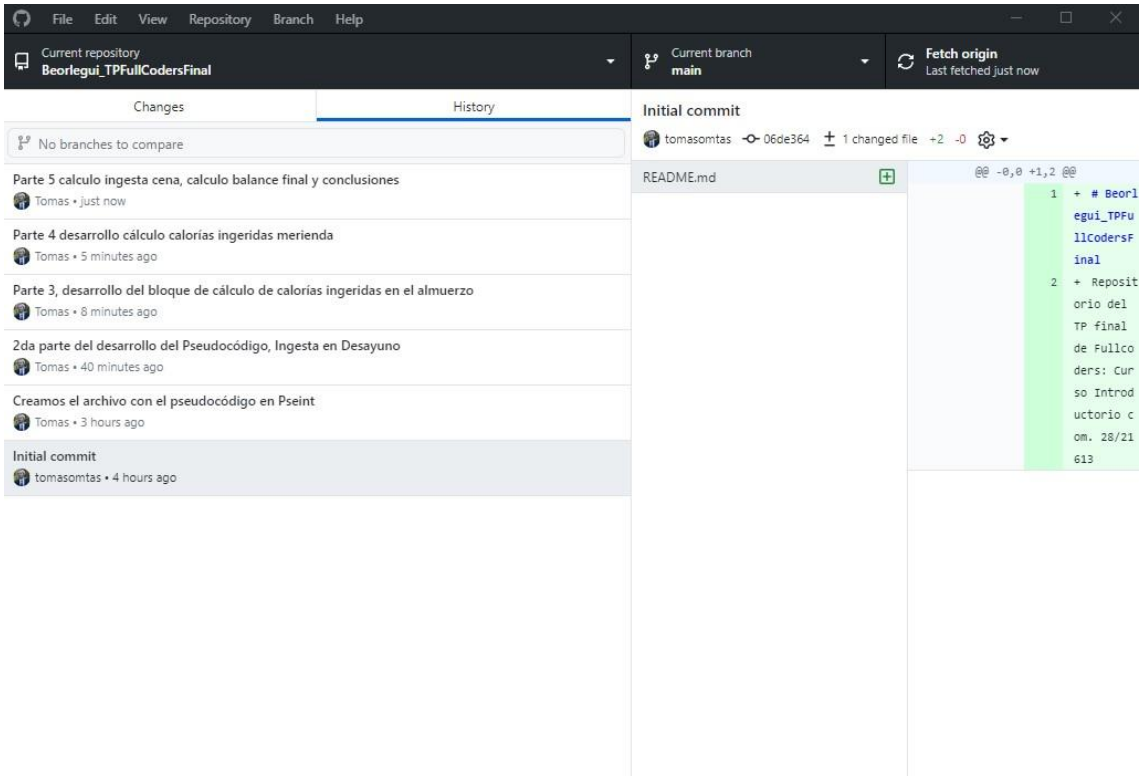

Imagen 3: Las 5 partes con sus commit y push en GitHub desktop

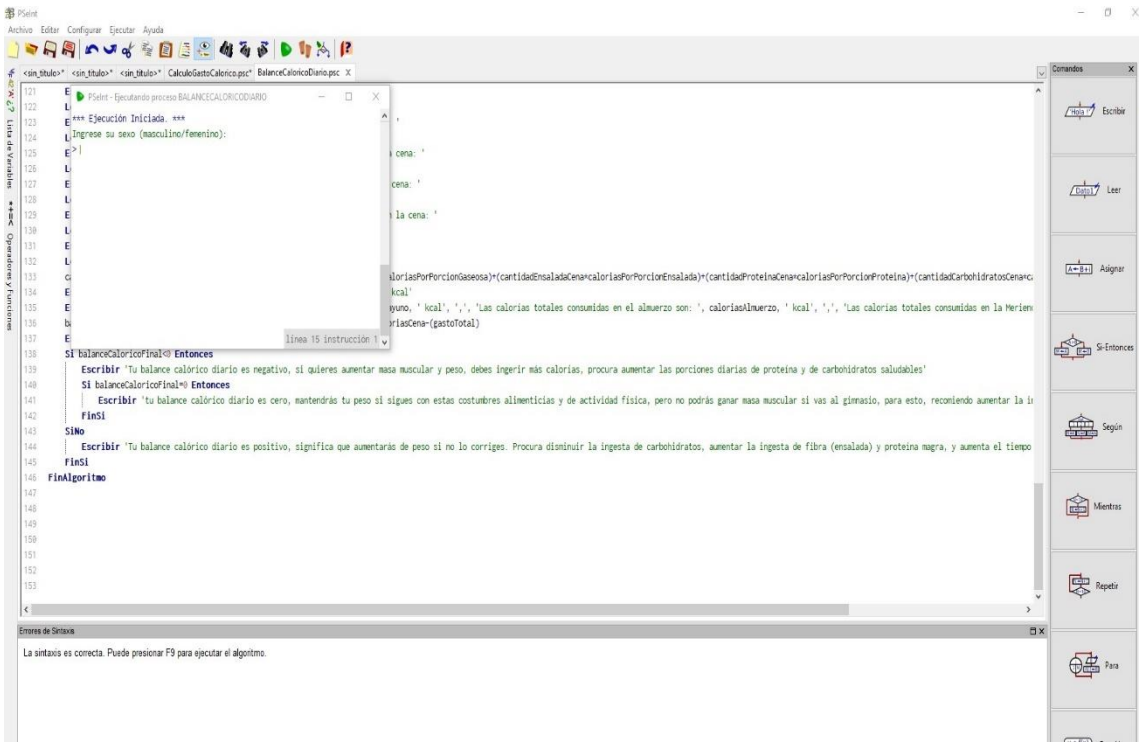

Imagen 4: El programa corriendo en PSeInt, interfaz de ingreso de datos

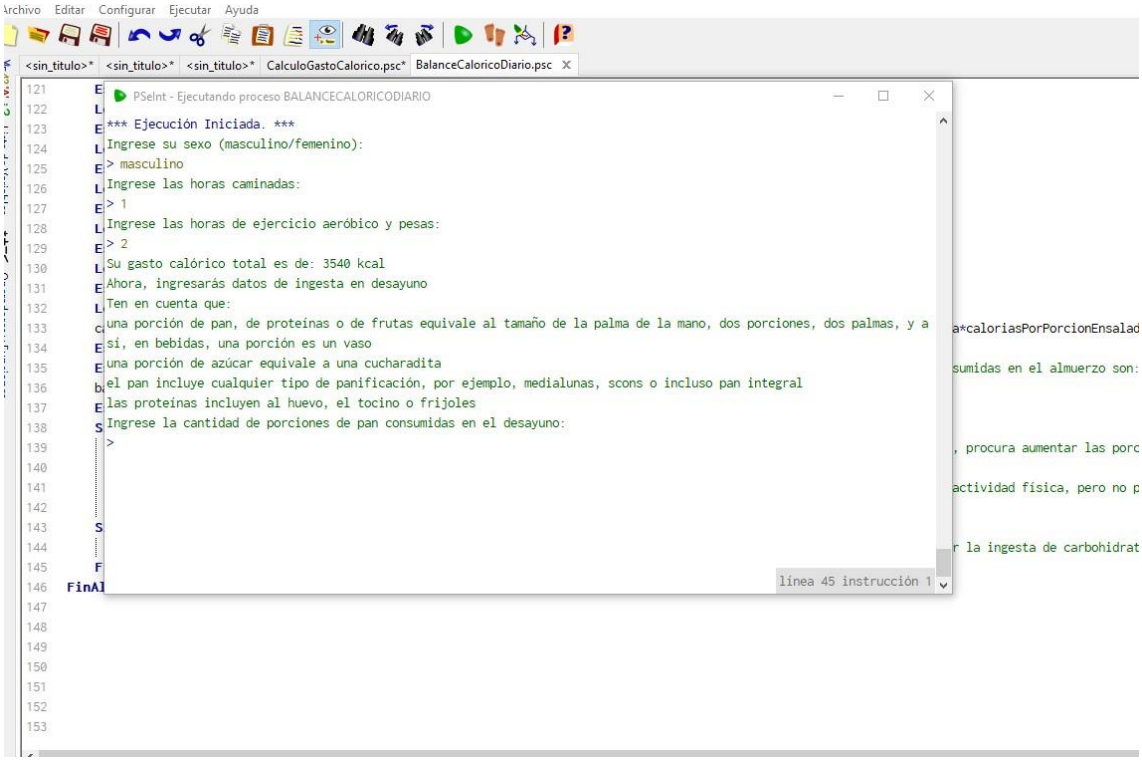

Imagen 5: El programa corriendo en PSeInt, interfaz de ingreso de datos

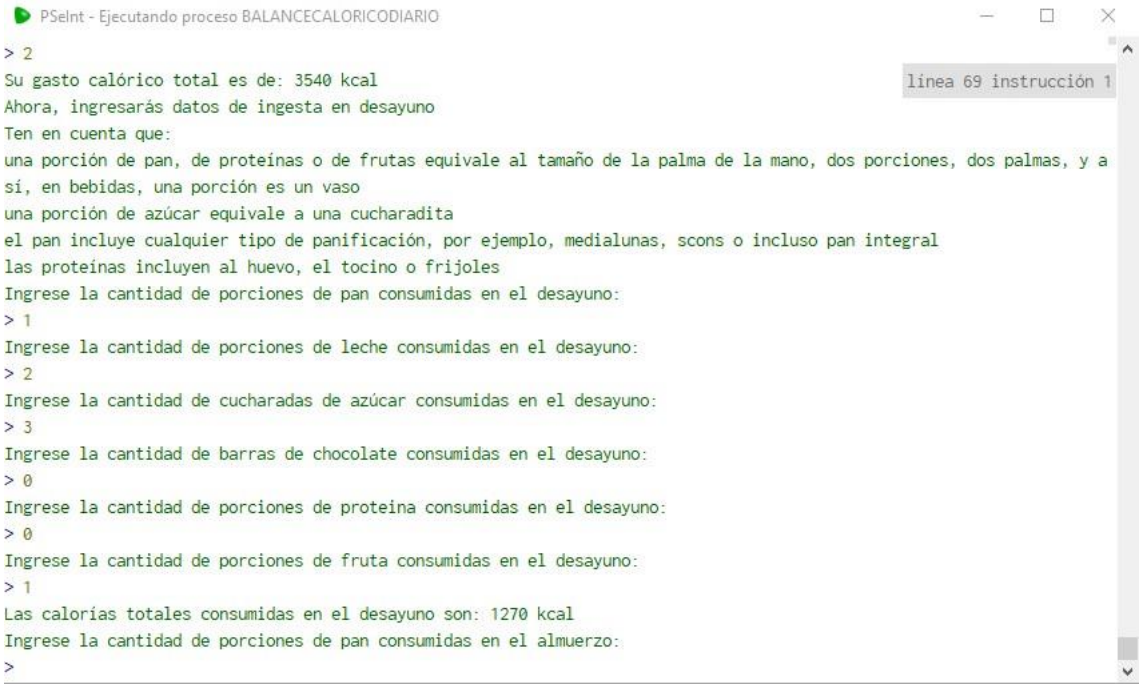

Imagen 6: El programa corriendo en PSeInt, interfaz de ingreso de datos

Y, por último, antes de pasar a explicar las metodologías ágiles aplicadas a este proyecto, compartimos el pseudocódigo generado en PSeInt para su copiado fácil y rápido:

```
Algoritmo BalanceCaloricoDiario
	Definir sexo Como Cadena
	Definir horasCaminadas, horasEjercicio Como Real
	Definir gastoMetabolicoBasal Como Real
	Definir gastoCaminatas, gastoEjercicio Como Real
	Definir gastoTotal Como Real
	Definir balanceCaloricoFinal  Como Real
	gastoMetabolicoMasculino <- 2500.0
	gastoMetabolicoFemenino <- 2000.0
	gastoPorHoraCaminata <- 300.0
	gastoPorHoraEjercicio <- 370.0
	Escribir 'Ingrese su sexo (masculino/femenino): '
	Leer sexo
	Escribir 'Ingrese las horas caminadas: '
	Leer horasCaminadas
	Escribir 'Ingrese las horas de ejercicio aeróbico y pesas: '
	Leer horasEjercicio
	Si sexo='masculino' Entonces
		gastoMetabolicoBasal <- gastoMetabolicoMasculino
	SiNo
		gastoMetabolicoBasal <- gastoMetabolicoFemenino
	FinSi
	gastoCaminatas <- horasCaminadas*gastoPorHoraCaminata
	gastoEjercicio <- horasEjercicio*gastoPorHoraEjercicio
	gastoTotal <- gastoMetabolicoBasal+gastoCaminatas+gastoEjercicio
	Escribir 'Su gasto calórico total es de: ', gastoTotal, ' kcal'
	Escribir 'Ahora, ingresarás datos de ingesta en desayuno'
	Definir cantidadPanDesayuno, cantidadLecheDesayuno, cantidadAzucarDesayuno,
cantidadChocolateDesayuno, cantidadProteinaDesayuno, cantidadFrutaDesayuno Como Entero
	Definir caloriasDesayuno Como Real
```

```
Escribir 'Ten en cuenta que:'
Escribir 'una porción de pan, de proteínas o de frutas equivale al tamaño de la palma de la mano, dos porciones, dos palmas, y así, en bebidas, una porción es un vaso'
Escribir 'una porción de azúcar equivale a una cucharadita'
Escribir 'el pan incluye cualquier tipo de panificación, por ejemplo, medialunas, scons o incluso pan integral'
Escribir 'las proteínas incluyen al huevo, el tocino o frijoles'
caloriasPorPorcionPan <- 600.0
caloriasPorPorcionLeche <- 125.0
caloriasPorPorcionAzucar <- 40.0
caloriasPorPorcionChocolate <- 300.0
caloriasPorPorcionProteina <- 200.0
caloriasPorPorcionFruta <- 300.0
Escribir 'Ingrese la cantidad de porciones de pan consumidas en el desayuno: '
Leer cantidadPanDesayuno
Escribir 'Ingrese la cantidad de porciones de leche consumidas en el desayuno: '
Leer cantidadLecheDesayuno
Escribir 'Ingrese la cantidad de cucharadas de azúcar consumidas en el desayuno: '
Leer cantidadAzucarDesayuno
Escribir 'Ingrese la cantidad de barras de chocolate consumidas en el desayuno: '
Leer cantidadChocolateDesayuno
Escribir 'Ingrese la cantidad de porciones de proteina consumidas en el desayuno: '
Leer cantidadProteinaDesayuno
Escribir 'Ingrese la cantidad de porciones de fruta consumidas en el desayuno: '
Leer cantidadFrutaDesayuno
caloriasDesayuno <- (cantidadPanDesayuno*caloriasPorPorcionPan)+(cantidadLecheDesayuno*caloriasPorPorcionLeche)+(cantidadAzucarDesayuno*caloriasPorPorcionAzucar)+(cantidadChocolateDesayuno*caloriasPorPorcionChocolate)+(cantidadProteinaDesayuno*caloriasPorPorcionProteina)+(cantidadFrutaDesayuno*caloriasPorPorcionFruta)
Escribir 'Las calorías totales consumidas en el desayuno son: ', caloriasDesayuno, ' kcal'
Definir cantidadPanAlmuerzo, cantidadGaseosaAlmuerzo, cantidadEnsaladaAlmuerzo, cantidadProteinaAlmuerzo, cantidadCarbohidratosAlmuerzo, cantidadSandwichAlmuerzo, cantidadPizzaAlmuerzo Como Entero
```

```
        Definir caloriasAlmuerzo Como Real
        caloriasPorPorcionPan <- 600.0
        caloriasPorPorcionGaseosa <- 250.0
        caloriasPorPorcionEnsalada <- 250.0
        caloriasPorPorcionProteina <- 200.0
        caloriasPorPorcionCarbohidratos <- 370.0
        caloriasPorPorcionSandwich <- 450.0
        caloriasPorPorcionPizza <- 270.0
        Escribir 'Ingrese la cantidad de porciones de pan consumidas en el almuerzo: '
        Leer cantidadPanAlmuerzo
        Escribir 'Ingrese la cantidad de gaseosa consumida en el almuerzo: '
        Leer cantidadGaseosaAlmuerzo
        Escribir 'Ingrese la cantidad de porciones de ensalada consumidas en el almuerzo: '
        Leer cantidadEnsaladaAlmuerzo
        Escribir 'Ingrese la cantidad de porciones de proteína magra consumidas en el
almuerzo: '
        Leer cantidadProteinaAlmuerzo
        Escribir 'Ingrese la cantidad de porciones de carbohidratos consumidas en el almuerzo:
'
        Leer cantidadCarbohidratosAlmuerzo
        Escribir 'Ingrese la cantidad de porciones de sandwich o pizzas consumidas en el
almuerzo: '
        Leer cantidadSandwichAlmuerzo
        Escribir 'Ingrese la cantidad de porciones de pizza consumidas en el almuerzo: '
        Leer cantidadPizzaAlmuerzo
        caloriasAlmuerzo <-
(cantidadPanAlmuerzo*caloriasPorPorcionPan)+(cantidadGaseosaAlmuerzo*caloriasPorPorcion
Gaseosa)+(cantidadEnsaladaAlmuerzo*caloriasPorPorcionEnsalada)+(cantidadProteinaAlmuerz
o*caloriasPorPorcionProteina)+(cantidadCarbohidratosAlmuerzo*caloriasPorPorcionCarbohidr
atos)+(cantidadSandwichAlmuerzo*caloriasPorPorcionSandwich)+(cantidadPizzaAlmuerzo*calo
riasPorPorcionPizza)
        Escribir 'Las calorías totales consumidas en el almuerzo son: ', caloriasAlmuerzo, ' kcal'
        Definir cantidadPanMerienda, cantidadLecheMerienda, cantidadAzucarMerienda,
cantidadChocolateMerienda, cantidadProteinaMerienda, cantidadFrutaMerienda Como Entero
```

```
        Definir caloriasMerienda Como Real
        Escribir 'Ingresa las porciones ingeridas durante la merienda'
        Escribir 'Las porciones equivalen igualmente que en el desayuno'
        caloriasPorPorcionPan <- 600.0
        caloriasPorPorcionLeche <- 125.0
        caloriasPorPorcionAzucar <- 40.0
        caloriasPorPorcionChocolate <- 300.0
        caloriasPorPorcionProteina <- 200.0
        caloriasPorPorcionFruta <- 300.0
        Escribir 'Ingrese la cantidad de porciones de pan consumidas durante la merienda: '
        Leer cantidadPanMerienda
        Escribir 'Ingrese la cantidad de porciones de leche consumidas durante la merienda: '
        Leer cantidadLecheMerienda
        Escribir 'Ingrese la cantidad de cucharadas de azúcar consumidas durante la merienda:
'
        Leer cantidadAzucarMerienda
        Escribir 'Ingrese la cantidad de barras de chocolate consumidas durante la merienda: '
        Leer cantidadChocolateMerienda
        Escribir 'Ingrese la cantidad de porciones de proteína consumidas durante la merienda:
'
        Leer cantidadProteinaMerienda
        Escribir 'Ingrese la cantidad de porciones de fruta consumidas durante la merienda: '
        Leer cantidadFrutaMerienda
        caloriasMerienda <-
(cantidadPanMerienda*caloriasPorPorcionPan)+(cantidadLecheMerienda*caloriasPorPorcionL
eche)+(cantidadAzucarMerienda*caloriasPorPorcionAzucar)+(cantidadChocolateMerienda*cal
oriasPorPorcionChocolate)+(cantidadProteinaMerienda*caloriasPorPorcionProteina)+(cantidad
FrutaMerienda*caloriasPorPorcionFruta)
        Escribir 'Las calorías totales consumidas en la Merienda son: ', caloriasMerienda, ' kcal'
        Definir cantidadPanCena, cantidadGaseosaCena, cantidadEnsaladaCena,
cantidadProteinaCena, cantidadCarbohidratosCena, cantidadSandwichCena, cantidadPizzaCena
Como Entero
        Definir caloriasCena Como Real
        caloriasPorPorcionPan <- 600.0
```

```
caloriasPorPorcionGaseosa <- 250.0
caloriasPorPorcionEnsalada <- 250.0
caloriasPorPorcionProteina <- 200.0
caloriasPorPorcionCarbohidratos <- 370.0
caloriasPorPorcionSandwich <- 450.0
caloriasPorPorcionPizza <- 270.0
Escribir 'Ingrese la cantidad de porciones de pan consumidas en la cena: '
Leer cantidadPanCena
Escribir 'Ingrese la cantidad de gaseosa consumida en la cena: '
Leer cantidadGaseosaCena
Escribir 'Ingrese la cantidad de porciones de ensalada consumidas en la cena: '
Leer cantidadEnsaladaCena
Escribir 'Ingrese la cantidad de porciones de proteína magra consumidas en la cena: '
Leer cantidadProteinaCena
Escribir 'Ingrese la cantidad de porciones de carbohidratos consumidas en la cena: '
Leer cantidadCarbohidratosCena
Escribir 'Ingrese la cantidad de porciones de sándwich o pizzas consumidas en la cena: '
Leer cantidadSandwichCena
Escribir 'Ingrese la cantidad de porciones de pizza consumidas en la cena: '
Leer cantidadPizzaCena
caloriasCena <-
(cantidadPanCena*caloriasPorPorcionPan)+(cantidadGaseosaCena*caloriasPorPorcionGaseosa)
+(cantidadEnsaladaCena*caloriasPorPorcionEnsalada)+(cantidadProteinaCena*caloriasPorPorci
onProteina)+(cantidadCarbohidratosCena*caloriasPorPorcionCarbohidratos)+(cantidadSandwic
hCena*caloriasPorPorcionSandwich)+(cantidadPizzaCena*caloriasPorPorcionPizza)
Escribir 'Las calorías totales consumidas en la cena son: ', caloriasCena, ' kcal'
Escribir 'Las calorías totales consumidas en el desayuno son: ', caloriasDesayuno, ' kcal',
',', 'Las calorías totales consumidas en el almuerzo son: ', caloriasAlmuerzo, ' kcal', ',', 'Las
calorías totales consumidas en la Merienda son: ', caloriasMerienda, ' kcal', ',', 'Y las calorías
totales consumidas en la cena son: ', caloriasCena, ' kcal'
balanceCaloricoFinal <-
caloriasDesayuno+caloriasAlmuerzo+caloriasMerienda+caloriasCena-(gastoTotal)
Escribir 'El balance calórico diario es : ', balanceCaloricoFinal, ' kcal'
Si balanceCaloricoFinal<0 Entonces
```

```
            Escribir 'Tu balance calórico diario es negativo, si quieres aumentar masa
muscular y peso, debes ingerir más calorías, procura aumentar las porciones diarias de
proteína y de carbohidratos saludables'
            Si balanceCaloricoFinal=0 Entonces
                Escribir 'tu balance calórico diario es cero, mantendrás tu peso si sigues
con estas costumbres alimenticias y de actividad física, pero no podrás ganar masa muscular si
vas al gimnasio, para esto, recomiendo aumentar la ingesta de proteína magra'
            FinSi
        SiNo
            Escribir 'Tu balance calórico diario es positivo, significa que aumentarás de
peso si no lo corriges. Procura disminuir la ingesta de carbohidratos, aumentar la ingesta de
fibra (ensalada) y proteína magra, y aumenta el tiempo de actividad física'
        FinSi
FinAlgoritmo
```

Link al repositorio en GitHub: https://github.com/tomasomtas/Beorlegui_TPFullCodersFinal.git

Metodologías Ágiles Aplicadas a la gestión de este proyecto con Scrum y Kanban

Al iniciar el proyecto, se pensó dividir el mismo en 5 partes. La primera parte, sería la porción de código que dé la bienvenida, pida el ingreso de datos al usuario (nombre y sexo en PSeInt, pero en una aplicación ya más desarrollada y compleja, como el código desarrollado en JAVA, pediría otros datos para conseguir resultados más precisos, como, por ejemplo, edad, peso actual, etc. Obviamente, contando con una base de datos, esto sería posible) y a partir de este dato (masculino o femenino) el programa bifurca hacia un lado o hacia el otro, ya que el gasto metabólico cambia según el sexo, y es un dato constante y de partida, a partir del cual se irán agregando gastos e ingresos. Representa un costo fijo, que paga sí o sí el cuerpo, sin siquiera tener que realizar actividad alguna, por el solo hecho de respirar. Pide las horas de caminata diarias, y las horas de ejercicio intensivo diario, para luego, realizar el cálculo del gasto metabólico calórico.

Luego, guardando este dato en la variable gastoTotal, realizamos la parte 2, en la que el proyecto tiene que escribir código para realizar el ingreso calórico en el desayuno, estableciendo qué se considera una “porción”, y los valores de cada porción de determinados alimentos típicos de un desayuno (declarando dichos valores en el pseudocódigo, entendiendo que en PSeInt no se pueden declarar constantes, solo otorgarles un valor, y solo pudiendo declarar variables) para luego, pedirle al usuario que ingrese la cantidad de porciones que ingirió en el desayuno. Como en el código ya están esos valores por porción declarados, el solo hecho de agregar las porciones de cada alimento hará que el programa realice un cálculo, obteniendo la cantidad de kcal ingeridas durante el desayuno. Lo mismo para el almuerzo, merienda y cena, para las partes 3 4 y 5, pero cambiando algunos alimentos. La parte 5 del proyecto, además de incluir el último cálculo de ingreso calórico (cena) trae al presente los otros valores obtenidos (gasto total, e ingresos calóricos por desayuno, almuerzo y merienda) y realiza el balance neto, dando un resultado y escribiendo en la interfaz gráfica, o sea en pantalla, un mensaje a modo de consejo.

Esta fue la idea principal e inicial del proyecto, separar el código en 5 partes, e incluso reutilizamos código en parte del proyecto, ya que desayuno y merienda, o almuerzo y cena, tienen alimentos típicos similares, y el cálculo es igual, sólo tuvimos que cambiar nombres de algunas variables, para que el código lea estos bloques por separado, y no reasigne valores nuevos, ya que, si por ejemplo, usamos el código escrito en el cálculo desayuno en merienda, sin cambiar los nombres de las variables, estas perderán su valor de la parte previa (desayuno) y el cálculo será erróneo.

Este es el plan de aplicación de Scrum en el desarrollo del proyecto del "Cálculo del Balance Calórico Diario". Scrum es una metodología ágil que se basa en la iteración y la colaboración continua, y a continuación, presentamos un plan simplificado para su implementación:

Fase de Inicio, Formación del Equipo Scrum:

Identificar a los miembros del equipo, que pueden incluir desarrolladores, un Scrum Master y un Product Owner.

Asegurarse de que todos comprendan su rol y las responsabilidades asociadas.

Creación del Product Backlog:

El Product Owner (PO) trabaja en conjunto con expertos en nutrición y salud para definir los requisitos del proyecto.

Enumerar todas las funcionalidades y características que se desean en la aplicación, como la entrada de datos, cálculos y generación de informes.

Planificamos varios sprint

Planificación del Sprint 1

Selección de Historias de Usuario:

El equipo revisa el Product Backlog y selecciona las historias de usuario más importantes para el Sprint 1. Por ejemplo, podrían seleccionar la historia "Cálculo del Gasto Calórico Basal".

Estimación de Tareas:

El equipo descompone las historias de usuario seleccionadas en tareas más pequeñas y estima el esfuerzo requerido para cada una.

Creación del Sprint Backlog:

El equipo crea un conjunto de tareas con las que comprometerse para el Sprint 1.

Ejecución del Sprint 1:

Daily Standup:

El equipo realiza reuniones diarias de seguimiento para discutir el progreso y los posibles obstáculos.

Desarrollo y Pruebas:

Los desarrolladores trabajan en la implementación de las funcionalidades asignadas.

Se realizan pruebas unitarias y de integración para garantizar la calidad del código.

Revisión del Sprint 1:

Sprint Review:

Al final del Sprint 1, el equipo realiza una revisión de las funcionalidades completadas y presenta los resultados al Product Owner.

El Product Owner puede proporcionar retroalimentación y aceptar o rechazar las funcionalidades implementadas.

Retrospectiva del Sprint 1:

El equipo lleva a cabo una retrospectiva para analizar lo que funcionó bien y lo que se puede mejorar en el próximo Sprint.

Se identifican obstáculos y se proponen soluciones.

Planificación del Sprint 2

Selección de Historias de Usuario para el Sprint 2:

Basado en la retroalimentación del Sprint 1 y las prioridades del Product Owner, el equipo selecciona nuevas historias de usuario para el próximo Sprint.

Estimación de Tareas para el Sprint 2:

El equipo estima el esfuerzo necesario para las nuevas tareas.

Creación del Sprint Backlog para el Sprint 2:

El equipo forma un conjunto de tareas para el próximo Sprint.

Repetición:

Este proceso se repite en ciclos sucesivos (Sprints) hasta que se complete el proyecto. En cada Sprint, se construyen nuevas funcionalidades y se obtiene retroalimentación del Product Owner y otros stakeholders. Se ajusta el Product Backlog en consecuencia para reflejar cambios y prioridades actualizadas.

Reflexiones:

Lo que funcionó bien:

La colaboración continua con el Product Owner y expertos en nutrición ayudó a garantizar que el producto final satisficiera las necesidades reales.

Las reuniones diarias de seguimiento (Daily Standup) mantuvieron a todos informados y permitieron abordar problemas rápidamente.

Lo que se puede mejorar:

La estimación de tareas podría mejorar con el tiempo para ser más precisa.

Asegurarse de que el equipo tenga un buen equilibrio de habilidades técnicas y conocimiento en nutrición y salud.

Algunos errores en el planeamiento del pseudocódigo, que no necesariamente se pasen o contaminen el código final, por ejemplo, asignamos valores reales cuando definimos, pero ¿qué pasa si un usuario pone valores negativos por error, por ejemplo, en las horas de caminata? Nadie puede caminar menos dos horas, sin embargo, los números reales incluyen números negativos también, por lo que ese error puede existir. Esos errores se corrigen escribiendo más pseudocódigo, más condiciones restrictivas en el mismo.

A continuación, el plan implementado según la metodología Kanban en el desarrollo del proyecto del "Cálculo del Balance Calórico Diario". Kanban es una metodología ágil que se centra en la visualización y gestión de tareas en un tablero. Aquí, una adaptación de cómo se podría utilizar Kanban para este proyecto:

Configuración Inicial

Definición del Tablero Kanban:

Se crea un tablero Kanban digital o físico que consta de columnas que representan diferentes etapas del proceso, por ejemplo, la división en 5 etapas escrito más arriba, como "Backlog", "En Progreso", "Revisión", "Listo", etc. Ya que, entendiendo la estructura del código, los ingresos y egresos calóricos son independientes entre sí, y por lo tanto pueden resolverse por separado

Creación del Backlog Inicial:

Se enumeran todas las funcionalidades y características deseadas (por ejemplo, buena optimización para que se ingresen datos de forma entendible y correcta, y en sucesivos

desarrollos, código cada vez más complejo, que permita trabajar con bases de datos de múltiples alimentos, con la cantidad de calorías exactas, o la posibilidad de incorporar gráficas, por ejemplo) en tarjetas individuales y se colocan en la columna "Backlog".

Flujo de Trabajo Continuo

Visualización y Priorización:

El equipo revisa el tablero Kanban y prioriza las tarjetas en la columna "Backlog" en función de su importancia y valor, por ejemplo, el incorporar gráficos a la aplicación, no es tan importante, como el resultado final del consejo que da el programa, una vez analizado el balance.

Selección de Tareas:

Los miembros del equipo toman tarjetas del "Backlog" y las mueven a la columna "En Progreso" cuando están listos para trabajar en ellas. Aquí se presentó un proyecto de aplicación en pseudocódigo, pero en el caso de un proyecto mas serio, como una aplicación desarrollada en JAVA, habrá desarrolladores que trabajen con base de datos, otros con la interfaz gráfica, y así en todo el proyecto.

Desarrollo y Pruebas:

Los desarrolladores trabajan en la implementación de las funcionalidades asignadas.

Se realizan pruebas unitarias y de integración durante el proceso.

Revisión y Aprobación:

Cuando una tarea está completa, se mueve a la columna "Revisión" para su revisión y aprobación por parte del Product Owner o expertos en nutrición.

Entrega y Listo para Lanzar:

Después de la revisión y aprobación, las tarjetas se mueven a la columna "Listo para Lanzar" y están listas para su implementación en la versión final de la aplicación.

Gestión Continua:

Reuniones de Standup (Opcionales):

El equipo puede llevar a cabo reuniones de Standup periódicas para sincronizar y discutir el progreso.

Actualización del Backlog:

A medida que surgen nuevas ideas o se realizan cambios en las prioridades, se agregan tarjetas al "Backlog" y se reorganizan según corresponda.

Reflexiones:

Lo que funcionó bien:

La visualización en el tablero Kanban proporciona una visión clara del flujo de trabajo y el estado de las tareas en todo momento.

La flexibilidad para agregar o cambiar tareas en el "Backlog" permite adaptarse a las necesidades cambiantes del proyecto.

Lo que se puede mejorar:

Se debe establecer un límite de trabajo en progreso (WIP) para evitar la sobrecarga de trabajo.

Asegurarse de que todas las tarjetas estén claramente definidas para evitar malentendidos en el equipo.

Hasta aquí el trabajo final.

Links: