

Trabajo Práctico Final

Laboratorio III

Integrantes:

Gonzalez, Crisanto Gabriel.

García Irizar, Ezequiel.

Quiroga, Manuel.

Universidad Tecnológica Nacional de Mar del Plata.

Docente: Gonzalo Benoffi.

Ayudante: Emmanuel Etcheber.

Fecha de entrega: 21/06/2023.

Introducción del Informe:

A continuación presentaremos un informe técnico respecto a la aplicación que hemos desarrollado en conjunto los integrantes del grupo, explicaremos brevemente el funcionamiento de la misma y como ha sido el desarrollo hasta la fecha de su finalización.

La aplicación, apodada “Nutribros”, está enfocada hacia aquel público que desea controlar su situación nutricional o quiera cumplir con algún objetivo específico ya sea bajar o subir de peso, para ello es que la hemos desarrollado, ni bien el usuario ingresa le pediremos cual es su objetivo y todos aquellos datos necesarios para calcular su BMR(Índice Metabólico Basal), lo cuál nos permitirá conocer cuantas calorías al día necesita para poder cumplir con su objetivo y en base a esto último darle a conocer una dieta personalizada la cual le servirá como guía.

Para comenzar con el desarrollo de la aplicación los primeros días fueron exclusivamente dedicados al armado del diseño de clases(UML), también a entender cómo funcionaba la aplicación de escritorio Github la cuál nos iba a permitir que cada integrante del grupo pueda ir haciendo avances/modificaciones en un mismo proyecto y por último también entender a cómo calcular las calorías necesarias para cada usuario y así de esta forma poder generarle su dieta correspondiente . Cuando ya todo esto se había podido lograr con éxito comenzamos a codificar según lo que habíamos diagramado, en el camino hacia lo que hoy es un resultado satisfactorio fue largo y con muchos errores los cuáles para poder subsanarlos fueron necesarias muchas horas de discusiones entre los integrantes y búsquedas infinitas en distintos foros para poder tener una idea de como resolver algo muy específico como por ejemplo, que se envíe un email a quien se registre.

Desarrollo del Informe:

Funcionamiento del Sistema

La aplicación va a poder ser utilizada por aquellas personas que quieran bajar de peso, subir de peso o mantener su peso y en base a esto poder armarles una dieta en base a lo que deseen. Para esto, les pediremos que se registren en nuestra aplicación y completen toda la información requerida. Una vez que el usuario se registre se le pedirá que inicie sesión, hecho esto, el usuario tendrá acceso a generar su dieta, ver su perfil (información personal) o salir de la aplicación. En el apartado de ver perfil el usuario podrá hacer cambios de sus datos personales y si lo desea podrá seleccionar clickear en “get premium” para acceder a una mejor experiencia.

Para que nuestra aplicación sea exitosa definimos dos clases principales Food y User(abstracta), de ésta última extenderán Premium User, Basic User y Admin User. Para poder almacenar en un tipo de colección a los objetos de ambas clases se nos ocurrió crear un mapa genérico ya que el tipo de dato que debíamos almacenar era irrelevante para nuestra

colección sino que lo más importante eran las operaciones generales que estas dos clases iban a compartir (agregar-borrar-listar-buscar), Food su key sería el id mientras que en User su email.

Para poder persistir la información de datos en el sistema fueron necesarios crear archivos de tipo JSON pero para esto creamos dos interfaces IToJson (para pasar un objeto a formato JSON) y IFromJson (para desglosar un objeto de un JSON) las cuales fueron implementadas por Food y User, y quienes extendían de este último le dieron su comportamiento correspondiente. También fue necesario la creación de un paquete llamado Handlers quién contendría aquellas clases que iban a ser necesarias para la manipulación de la información. La clase Intermediary, que forma parte de Handlers, sería la encargada de tener los mapas genéricos de Food y User, para luego con sus métodos poder establecer una conexión con dos clases estáticas que se encargarán de escribir y leer los archivos correspondientes (FileHandler y JSONHandler).

Decisiones de Diseño

Al comenzar a construir nuestro diagrama de clases nos dimos cuenta que destacaban dos, Food y User, decidimos que esta última sea abstracta ya que el concepto de User en sí no existe, es por eso que optamos que Basic User, Premium User y Admin User extiendan de User. Como ambas clases compartían operaciones generales (agregar-borrar-buscar-listar) decidimos que para manipularlas se almacenen en un mismo mapa genérico ya que el tipo de dato que necesitaríamos manipular sería irrelevante. Decidimos un Hashmap ya que de ambas clases no queríamos repetidos y podían ser identificados por una Key, las Foods por su id y los Users por su email. Debido a esto tomamos la decisión que nuestra clase envoltorio contenga ambos mapas y se quien actúe como intermediario con aquellas clases que contengan los métodos para almacenar en los archivos Json y los datos puedan persistir en el sistema.

En nuestro sistema decidimos que existan dos interfaces IToJson y IFromJson, ambas serían implementadas por aquellas clases que manipulen la información y necesiten pasar a JSON o ser leídas de un JSON, (debido a que cada clase implementaría el método con un comportamiento diferente). Para que esta tarea se realice eficientemente pensamos en la creación de un paquete llamado Handlers, en la que también se encuentra nuestra clase envoltorio, pero además clases que sus métodos se encargan en persistir la información en archivos Json y también una clase DataValidation que se encargue de validar la información ingresada por el usuario.

En lo que respecta de Excepciones decidimos que exista un paquete el cuál contendrá la creación de todas las excepciones de nuestro sistema y que recibirá el tratamiento correspondiente en donde sean llamadas.

Clases Utilizadas y sus Relaciones

Para explicar de forma clara y concisa comenzaremos con User:

Tendremos un paquete llamado Users el cuál contendrá todas clases relacionadas a la clase abstracta User. En primer lugar tenemos la UserData que servirá como atributo de la clase User y contendrá toda la información necesaria para la elaboración de la dieta y también la almacenará en una lista llamada Diet. La clase User tendrá tres subclases, BasicUser, PremiumUser y AdminUser. BasicUser es la situación en la que se encuentra el usuario ni bien inicia sesión luego de registrarse, tendrá funciones limitadas con respecto a la generación de dietas. PremiumUser, para poder acceder a mayores funcionalidades, si así lo desea el usuario, deberá efectuar un pago para el acceso a las mismas. AdminUser, este último es exclusivamente para quienes forman parte del desarrollo de la aplicación si bien no harán uso de la misma podrán tener acceso a funcionalidades admin para llevar un control.

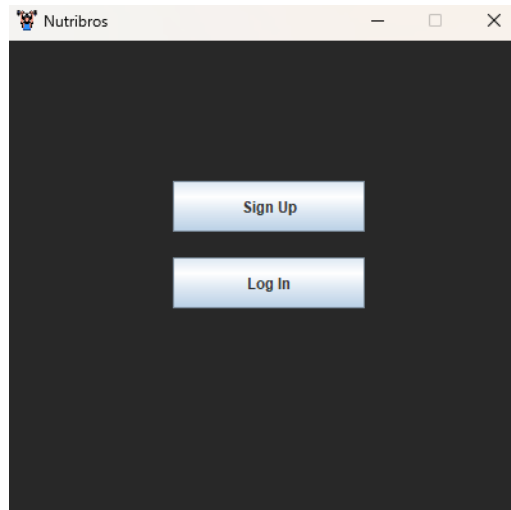
El paquete también contendrá dos Enums, Objective y PhysicalActivity, ambas serán atributos de User, la primera será para conocer el objetivo del usuario que únicamente podrán ser tres (subir de peso, bajar de peso, mantener su peso) y PhysicalActivity es una situación similar, será para saber si su actividad física es moderada, activa o nula. Ambos Enums servirán como parámetro para la creación de la dieta personalizada.

Por otro lado, tendremos otro paquete llamado FoodModels el cuál contendrá dos clases, Food y FoodType, la primera contendrá todos los atributos necesarios para luego nosotros generarle la dieta al usuario por ejemplo las calorías, la segunda será un Enum para que nosotros sepamos que tipo de comida es la que se está creando (desayuno, almuerzo, cena, tentempié).

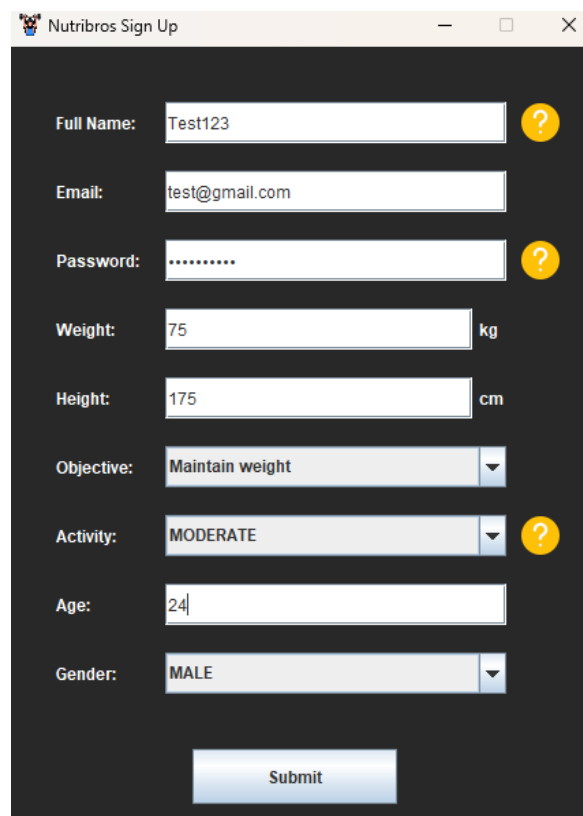
Para poder almacenar y manipular la información tendremos un paquete llamado Handlers el cuál contendrá la clase Intermediary que contiene ambos mapas genéricos de Foods y Users, actuará como clase envoltorio entre lo que ocurre en el sistema en tiempo de ejecución y las clases FileHandler y JSONHandler quienes se encargarán de persistir y obtener la información. También tendremos la clase DataValidation quién se encargará de comprobar que la información ingresada por el usuario sea correcta.

Manual de Usuario

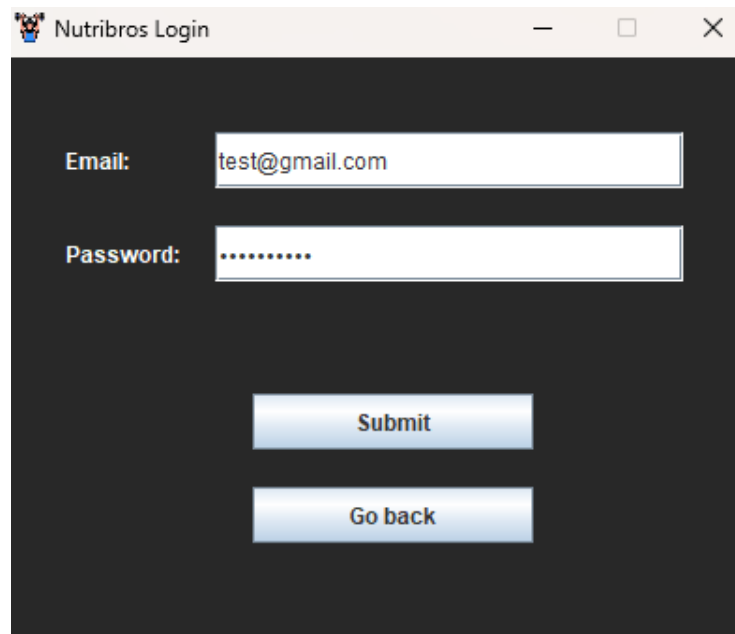
La primera impresión que tendrá el usuario una vez inicie el programa será encontrarse con esta ventana la cual le indicará con dos botones que acción desea realizar si registrarse (Sign Up) o iniciar sesión (Log In).



Si el usuario clickea sobre el botón Sign Up se le abrirá la siguiente ventana la cuál le pedirá que ingrese toda la información necesaria para quedar registrado en nuestra base de datos y la misma pueda ser utilizada para brindarle un óptimo servicio.

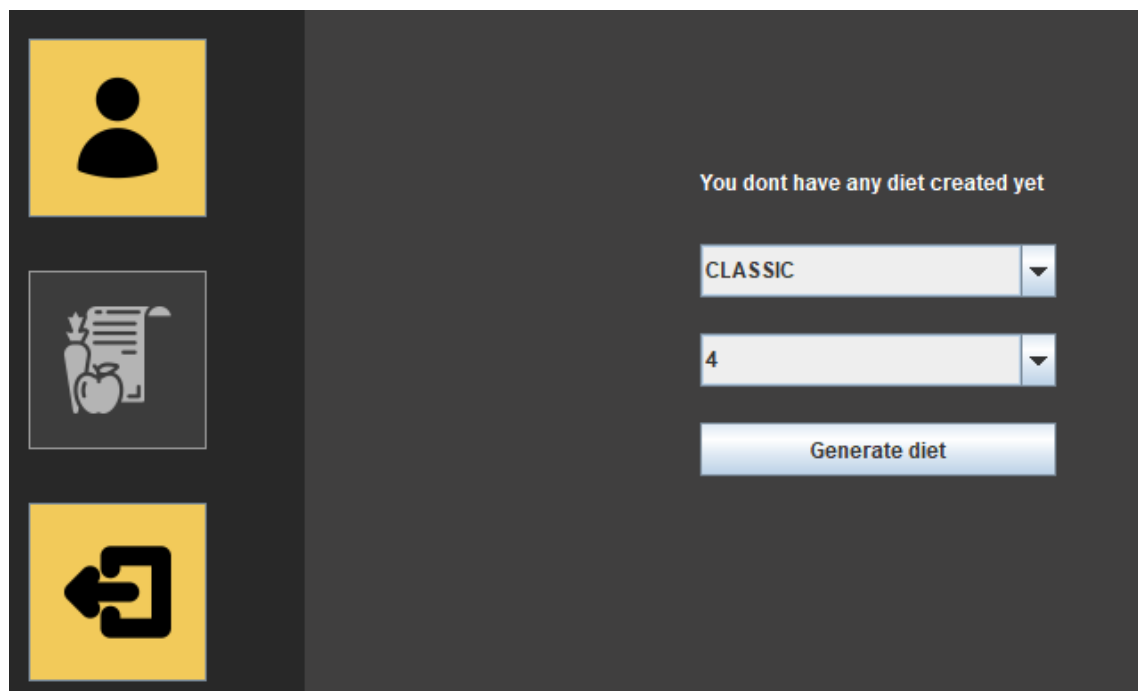
A screenshot of a software window titled "Nutribros Sign Up". The window has a dark gray background. It contains a form with several input fields and dropdown menus. The fields are labeled as follows: "Full Name:" with the value "Test123", "Email:" with the value "test@gmail.com", "Password:" with a masked value ".....", "Weight:" with the value "75" and a unit "kg", "Height:" with the value "175" and a unit "cm", "Objective:" with a dropdown menu showing "Maintain weight", "Activity:" with a dropdown menu showing "MODERATE", "Age:" with the value "24", and "Gender:" with a dropdown menu showing "MALE". There are yellow circular help icons (question marks) next to the "Full Name:", "Password:", and "Activity:" fields. At the bottom of the form is a light blue "Submit" button.

Una vez el usuario se haya registrado o al comienzo de la aplicación hubiera clickeado Log In se le abrirá la siguiente ventana en la que le pediremos que ingrese su email con el que se ha registrado y su contraseña.



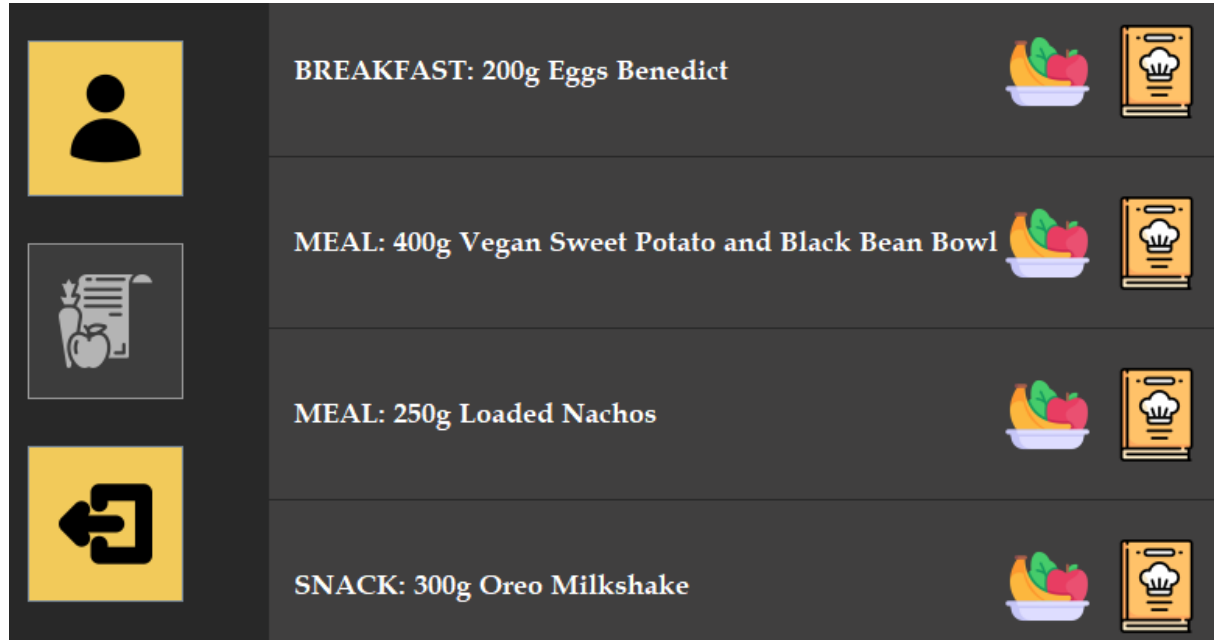
A screenshot of a web browser window titled "Nutribros Login". The window has a dark background. It contains two input fields: "Email:" with the text "test@gmail.com" and "Password:" with masked characters ".....". Below the fields are two buttons: "Submit" and "Go back".

Una vez clickeado el botón de Submit se le abrirá la siguiente interfaz, en el lado izquierdo se pueden apreciar tres botones, de arriba hacia abajo serían: ver perfil, generar nueva dieta (solo para usuarios premium) y salir. En la parte central de la interfaz es donde aparecerá nuestra dieta, en el caso de no tener una se generará en base a los requerimientos del usuario, podrá elegir el tipo de comidas (clásica,celíaca,vegetariana,vegana) y la cantidad de comidas que quiere en el día.

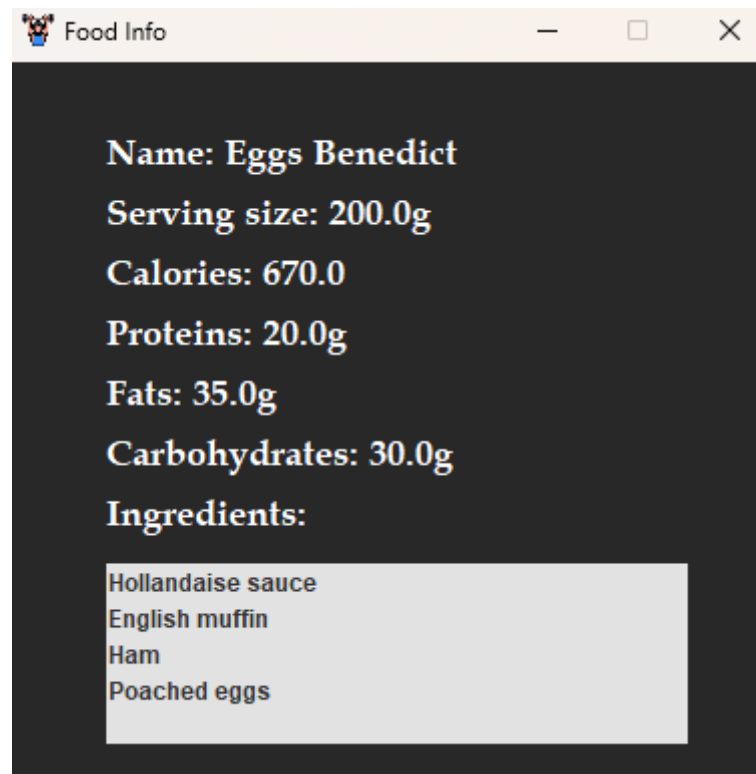


A screenshot of the main interface of the Nutribros application. On the left side, there is a vertical sidebar with three yellow buttons: a person icon (profile), a fork and knife icon (diet), and a back arrow icon (logout). The main area has a dark background. At the top, it says "You dont have any diet created yet". Below this, there are two dropdown menus: the first is labeled "CLASSIC" and the second is labeled "4". At the bottom of the main area is a button labeled "Generate diet".

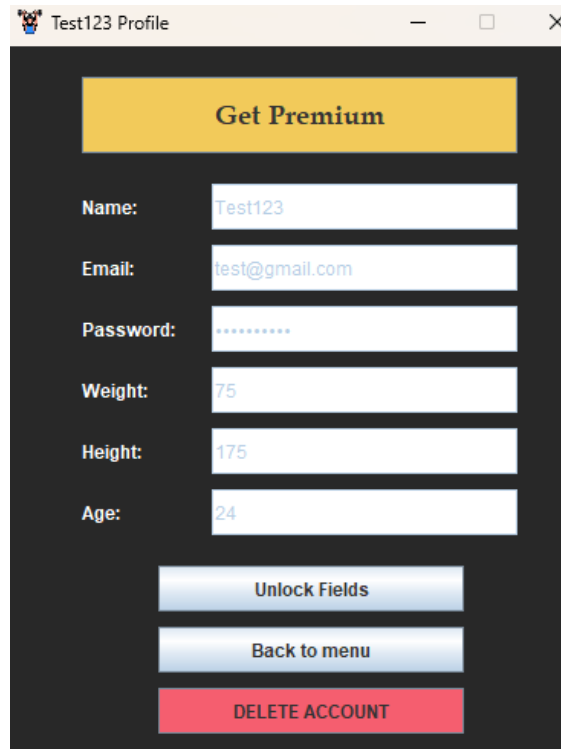
Si hacemos click en el botón Generate Diet se agregará a la interfaz la dieta creada y en cada renglón una comida las cuales tendrán dos botones, el ícono de las frutas será para ver cómo está compuesta la comida y la otra te redirigirá a una página en la que te mostrará los pasos para poder cocinarla.



Si hacemos click en el ícono de las frutas de la primer comida se abrirá esta ventana.

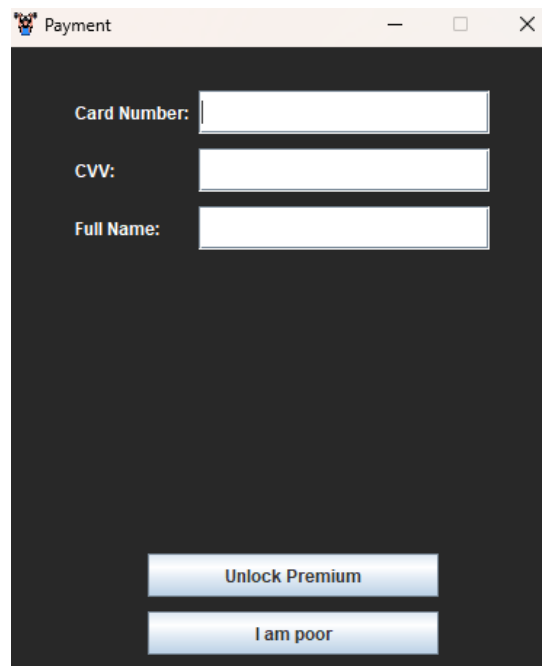


Ahora centrándonos en los botones del sector izquierdo de la interfaz si ingresamos al de ver perfil se abrirá una ventana donde el usuario podrá ver su información personal y en el caso que necesite realizar una modificación deberá clicar en Unlock Fields, realizar las modificaciones y posteriormente aparecerá un botón de Save Data para guardar la nueva información. Si el usuario lo desea, habrá un Get Premium el cual le habilitará el acceso a las funcionalidades de usuario premium pero para esto deberá efectuar un pago. También habrá un botón rojo que le permitirá al usuario borrar su cuenta.



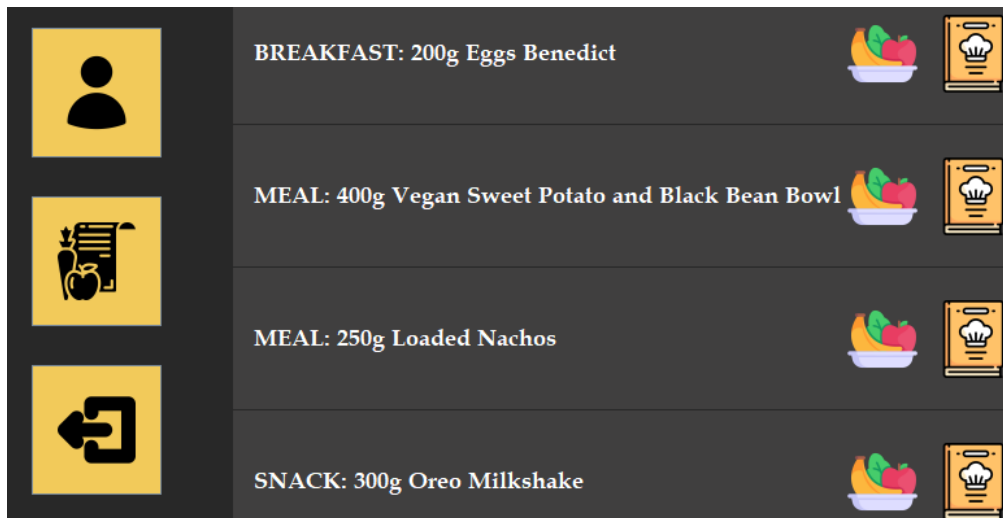
The screenshot shows a window titled "Test123 Profile". At the top is a yellow button labeled "Get Premium". Below it, on a dark background, are several input fields with labels: "Name:" (containing "Test123"), "Email:" (containing "test@gmail.com"), "Password:" (containing "*****"), "Weight:" (containing "75"), "Height:" (containing "175"), and "Age:" (containing "24"). At the bottom of the form are three buttons: "Unlock Fields" (light blue), "Back to menu" (light blue), and "DELETE ACCOUNT" (red).

Si el usuario hace click en Get Premium se abrirá la siguiente ventana pidiendo el número de su tarjeta, el código de seguridad de la misma y su nombre.



The screenshot shows a window titled "Payment". It contains three input fields with labels: "Card Number:" (empty), "CVV:" (empty), and "Full Name:" (empty). At the bottom of the form are two buttons: "Unlock Premium" (light blue) and "I am poor" (light blue).

Una vez efectuado el pago se le pedirá al usuario que inicie sesión nuevamente y ahora en su interfaz aparecerá habilitado el botón para poder generar una nueva dieta con un máximo de diez.



Por último y no menos importante tendremos el menú de admin el cuál iniciará sesión como un usuario normal con la diferencia que no tendrá una interfaz visual y tendrá funcionalidades de administrador.

Email:

Password:

```
Welcome admin
Select option
1. DISPLAY ALL USERS
2. DISPLAY ALL FOODS
3. DELETE ONE USER
4. DELETE ONE FOOD
5. SEARCH ONE FOOD
6. SEARCH ONE USER
7. ADD ONE FOOD
```

Diario de Trabajo

- [22/5 13:00] Decidimos la idea principal del proyecto.
- [22/5 13:35] Gabriel Gonzalez buscó API's para integrar al proyecto.
- [22/5 15:43] Manuel Quiroga creó el repositorio en GitHub.
- [22/5 18:00] Los tres probamos hacer commits y push.
- [22/5 18:22] Manuel Quiroga hizo chanchada y tuvo que eliminar el repositorio de GitHub.
- [22/5 18:22] Manuel Quiroga creó otro repositorio en GitHub. (este si funciona).
- [23/5 11:24] Establecimos la conexión con la primera API, con fines de prueba.
- [23/5 11:30] Establecimos la conexión con la segunda API, con fines de prueba.
- [23/5 12:34] Arrancamos a diagramar la estructura del proyecto.
- [27/5 15:34] Utilizamos LucidChart para continuar definiendo nuestro diseño de clases.
- [29/5 09:23] Decidimos quitar las API del proyecto.
- [29/5 09:40] Terminamos la planificación del proyecto.
- [29/5 17:11] Comenzamos a codificar las Clases, Interfaces, Excepciones en el IDE.
- [29/5 19:00] Armamos los JSON de las clases que corresponde para guardarlo en archivos.
- [1/6 11:30] Armamos un prototipo de UI en Swing.
- [1/6 15:23] Creamos una "interface" con métodos de validación de datos (name, password, email).
- [1/6 16:25] Agregamos las librerías y funciones para enviar un email.
- [1/6 16:50] Creamos una contraseña de aplicación para darle acceso a la app.
- [1/6 17:00] Conectamos el email para poder notificar a los usuarios de su registro.
- [2/6 00:00] Acordamos un descanso hasta la finalización de los exámenes.
- [10/6 18:30] Hicimos el algoritmo para generar dietas.
- [11/6 00:00] Descanso hasta el término de parciales.
- [14/6 15:35] Realizamos las funcionalidades de la colección genérica.
- [15/6 16:45] Agregamos métodos para manejar los usuarios y comidas en la colección genérica(modificar, borrar, buscar, agregar)
- [15/6 16:45] Empezamos a armar el menú con el cual va a interactuar el usuario, mediante java SWING.
- [16/6 09:15] Finalizamos el apartado de generación de dietas.
- [16/6 09:15] Comenzamos a planear el perfil del usuario
- [16/6 12:30] Terminamos el perfil del usuario.
- [16/6 12:30] Comenzamos a planear el apartado de usuarios premium.
- [16/6 18:20] Terminamos el apartado de usuarios premium.
- [17/6 10:00] Realizamos un debuggeo general y acomodamos algunos métodos.
- [17/6 18:30] Comenzamos a planear y armar el apartado de administrador.
- [17/6 20:00] Terminamos el apartado de administrador.
- [18/6 14:00] Agregamos una API para las recetas de las comidas.
- [18/6 16:00] Dejamos a punto todo lo necesario para la creación del Java.doc.
- [19/6 11:00] Realizamos testeos finales al programa.
- [19/6 13:00] Hicimos el manual de usuario.

Matriz de Soluciones

Fecha	Problemas	Solución
23/05/2023	No aplicábamos herencia	Hicimos la clase User abstracta y la dividimos en tres.
25/05/2023	Las API's encontradas no eran del todo útiles.	Crear nuestro propio archivo JSON de comidas.
30/05/2023	Agregar un objeto JSON al archivo JSON cuando ingresábamos mal los datos de un usuario.	Desconocemos cómo se solucionó.
03/06/2023	Se borra el archivo cada vez que agregamos un usuario.	Creamos un método que si el archivo existía lo reescriba.
03/06/2023	Agregar más de un usuario al archivo JSON.	Arreglamos keys en el método de JSONHandler.
13/06/2023	Cuando se borraba la cuenta de un usuario, se borraba otra.	Cambiamos las id de tipo int a tipo UUID.
15/06/2023	No podía registrarse un usuario por más que el email no estaba en nuestra base de datos.	Cambiamos la lógica dentro de un if(), en la clase DataValidation.
16/06/2023	Mismo problema de borrado de los usuarios en las comidas.	Creamos un método que refactoriza las id para acomodarlas.
18/06/2023	Cuando el administrador quería agregar o borrar una comida el archivo JSON de las mismas quedaba mal formado.	Estábamos utilizando un método que no correspondía.

Conclusión del Informe:

Finalmente, como grupo hemos llegado a un mismo pensamiento, llevar a cabo las exigencias demandadas por el trabajo práctico final con éste tipo de lenguaje (Java), comparado con C, fué mucho más llevadero y cómodo, ya que nos proporcionaba muchísimos métodos que nos ahorran trabajo y nos daban la posibilidad de realizar funcionalidades más complejas. También que sea un lenguaje fuertemente tipado nos permitía encontrar los errores con mucha más facilidad. Sentimos que hicimos un buen proyecto para afianzar los conocimientos adquiridos durante la cátedra.

Bibliografía:

- Oracle[<https://docs.oracle.com/javase/7/api/javax/mail/package-summary.html>].
- Oracle[<https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>].
- Geeksforgeeks[<https://www.geeksforgeeks.org/send-email-using-java-program/>].
- Benoffi, G[<https://github.com/benoffi7/Programacion-Laboratorio-3-UTN-MDP/commit/5cb412ec5d84b1b0159cd80b7403179d2c35d9a3>].
- Benoffi, G[<https://docs.google.com/presentation/d/1v6mxAAk3XUizLmXqOFgYFRv8Kn1RMUU-iOO4XQYoq9M/edit#slide=id.p9>].