Instituto Tecnológico de Celaya

Ingeniería en sistemas computacionales

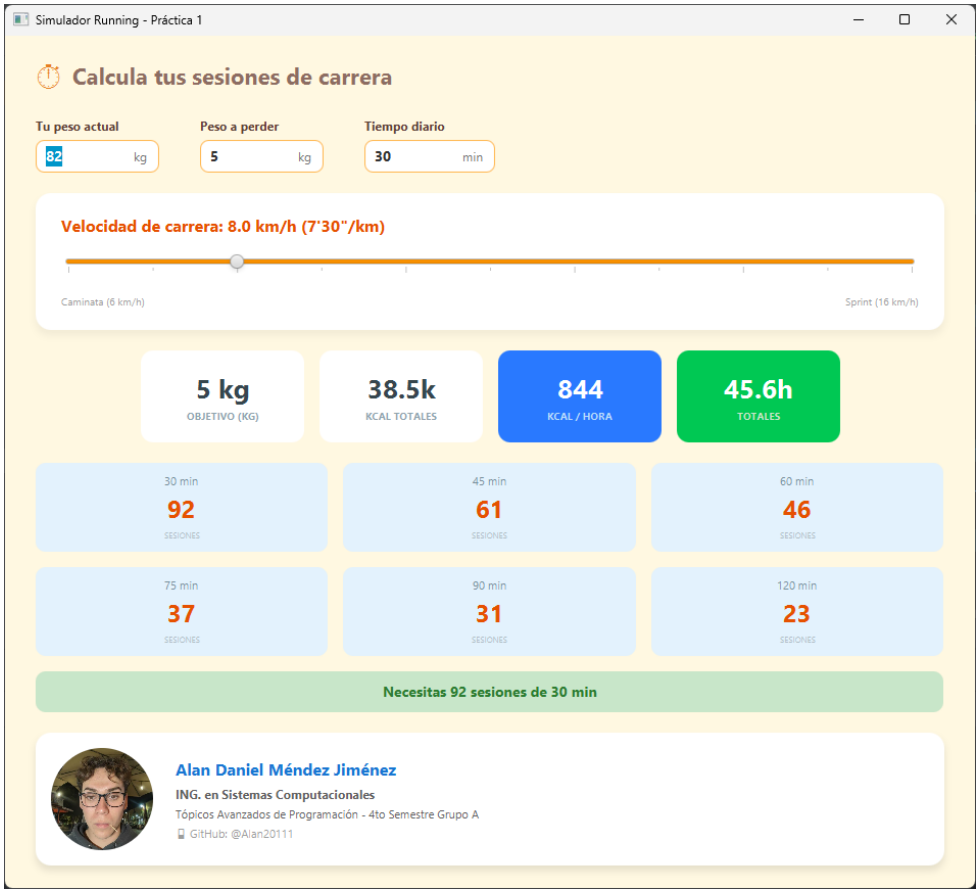
PRÁCTICA 1

Nombre: Alan Daniel Méndez Jiménez 24030976	**Fecha de entrega:** 17-02-2026	**Calificación:**
Materia: Tópicos Avanzados de programación		

1. Introducción
2. Desarrollo
3. Resultados
4. Conclusiones

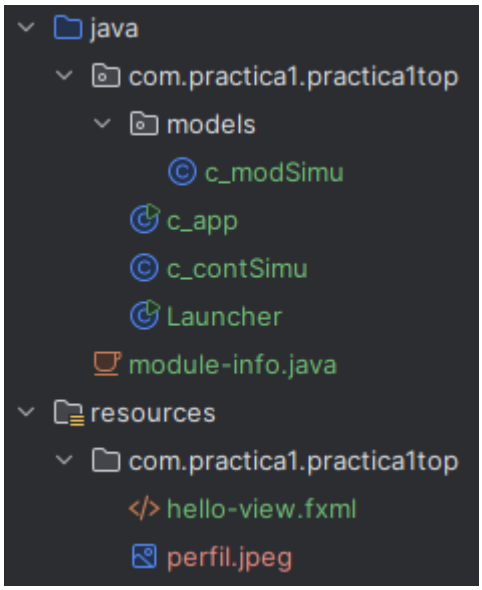
SIMULADOR DE RUNNING

Proyecto alojado en https://github.com/Alan20111/Practica1TOP



INTRODUCCIÓN

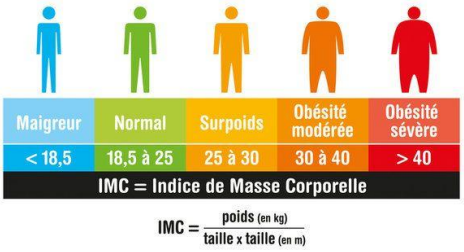
En el ámbito de la ingeniería de software moderna, el desarrollo de interfaces gráficas de usuario (GUI) eficientes e intuitivas es fundamental para garantizar una interacción fluida entre el humano y la máquina. El presente proyecto, titulado "Simulador de Running", consiste en el diseño e implementación de una aplicación de escritorio desarrollada bajo el entorno de trabajo JavaFX, correspondiente a la Práctica 1 de la asignatura Tópicos Avanzados de Programación.



El objetivo principal de esta aplicación es simular el proceso de quema calórica y pérdida de peso mediante la actividad física de correr. Aunque se trata de una aproximación matemática y no de una herramienta médica, el software integra fórmulas fundamentadas en el Equivalente Metabólico de Tarea (MET) para realizar estimaciones basadas en variables críticas proporcionadas por el usuario, tales como el peso actual, el peso objetivo, la velocidad de desplazamiento y el tiempo dedicado diariamente a la actividad.

Desde una perspectiva técnica, el desarrollo de este simulador permite aplicar los paradigmas de la Programación Orientada a Objetos (POO) y, específicamente, el patrón de arquitectura de software Modelo-Vista-Controlador (MVC). Esta separación de responsabilidades asegura que la lógica de negocio (cálculos de calorías y sesiones), la interfaz de usuario (archivos FXML) y la gestión de eventos (controladores) operen de manera desacoplada y mantenible.

DESARROLLO



Para la implementación del "Simulador de Running", se optó por una arquitectura basada en el patrón de diseño Modelo-Vista-Controlador (MVC). Esta decisión permite desacoplar la lógica matemática de la interfaz gráfica, facilitando la escalabilidad y el mantenimiento del código. El desarrollo se llevó a cabo utilizando el lenguaje Java y la biblioteca gráfica JavaFX.

IMPLEMENTACIÓN DE LA LÓGICA MATEMÁTICA (MODELO)

El núcleo de la simulación reside en la clase c_modSimu. Se implementaron métodos para calcular el gasto calórico basándose en el Equivalente Metabólico de Tarea (MET).

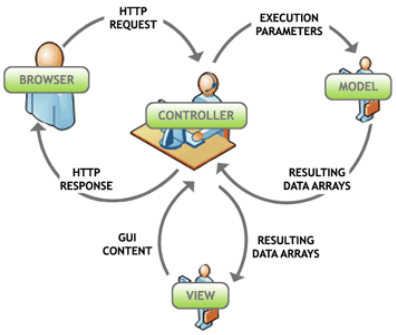
Selección del MET: Se utilizó una estructura condicional para asignar el valor MET según la velocidad (km/h), variando desde 8.3 (trote suave) hasta 12.8 (carrera muy rápida).

Cálculo de Calorías por Minuto: Se aplicó la fórmula fisiológica estándar:

$$Kcal/min = \frac{MET \times PesoActual \times 3.5}{200}$$

Proyección de Sesiones: Para determinar cuántas sesiones son necesarias, se implementó un algoritmo que divide el total de calorías objetivo entre el gasto calórico de una sesión de duración variable (n minutos), redondeando siempre hacia el entero superior (Math.ceil) para asegurar el cumplimiento de la meta.

DISEÑO DE LA INTERFAZ GRÁFICA (VISTA)



La interfaz de usuario se diseñó priorizando la experiencia de usuario (UX) y la claridad de la información. En lugar de utilizar bibliotecas externas de estilos, se implementó **Generado con inteligencia artificial** para el mejor manejo de tiempo y de lógica optimizando tiempos y siendo agradable a la vista directamente sobre los componentes de JavaFX para lograr un diseño moderno y limpio ("Material Design").

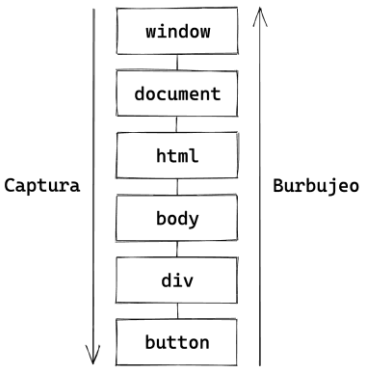
Componentes Clave:

- **Tarjetas Informativas (Cards):** Se utilizaron contenedores VBox con bordes redondeados y sombras (dropshadow) para agrupar métricas relacionadas (Objetivo, Intensidad, Totales).
- **Slider Interactivo:** Se incorporó un control deslizante para ajustar la velocidad, proporcionando retroalimentación visual inmediata.

  Matriz de Resultados: Se empleó un GridPane para mostrar de forma organizada las sesiones necesarias para diferentes duraciones (30, 45, 60, 75, 90 y 120 minutos).

INTERACTIVIDAD Y GESTIÓN DE EVENTOS (CONTROLADOR)

Para garantizar una experiencia fluida, se implementaron **Listeners** (escuchas) en las propiedades de los campos de texto y el slider. Esto permite que el sistema recalcule todos los valores en **tiempo real** cada vez que el usuario modifica un dato, eliminando la necesidad de un botón explícito de "Calcular".



Adicionalmente, se programó una conversión matemática para transformar la velocidad lineal (km/h) a ritmo de carrera o *Pace* (minutos por kilómetro), un dato técnico valorado en el ámbito deportivo (ej. 8.0 km/h == 7'30"/km).

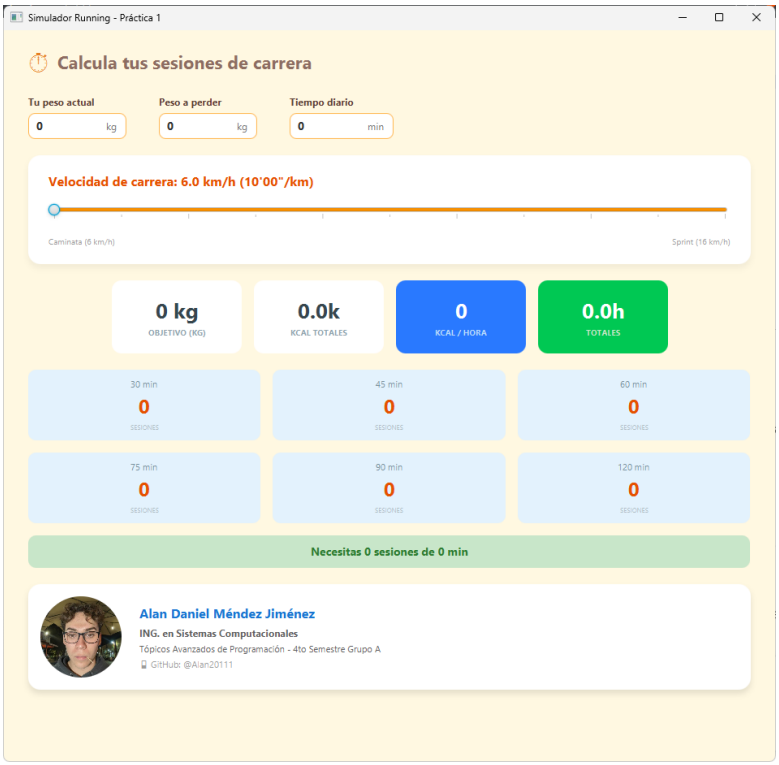
RESULTADOS

Tras la fase de codificación y depuración, se obtuvo una aplicación de escritorio estable y funcional que cumple con todos los requerimientos planteados en la práctica. La interfaz gráfica, diseñada mediante FXML y estilizada con CSS nativo, permite una interacción fluida y presenta los datos de manera clara y organizada.

A continuación, se describen los escenarios de prueba y los resultados obtenidos:

INTERFAZ PRINCIPAL Y CARGA INICIAL

Al ejecutar la aplicación, el sistema inicializa correctamente todos los componentes visuales. Se observa la carga exitosa de la imagen de perfil del desarrollador (con recorte circular mediante código Java) y los valores por defecto definidos en los campos de texto. La distribución de elementos respeta la jerarquía visual propuesta: configuración en la parte superior y resultados en la inferior.

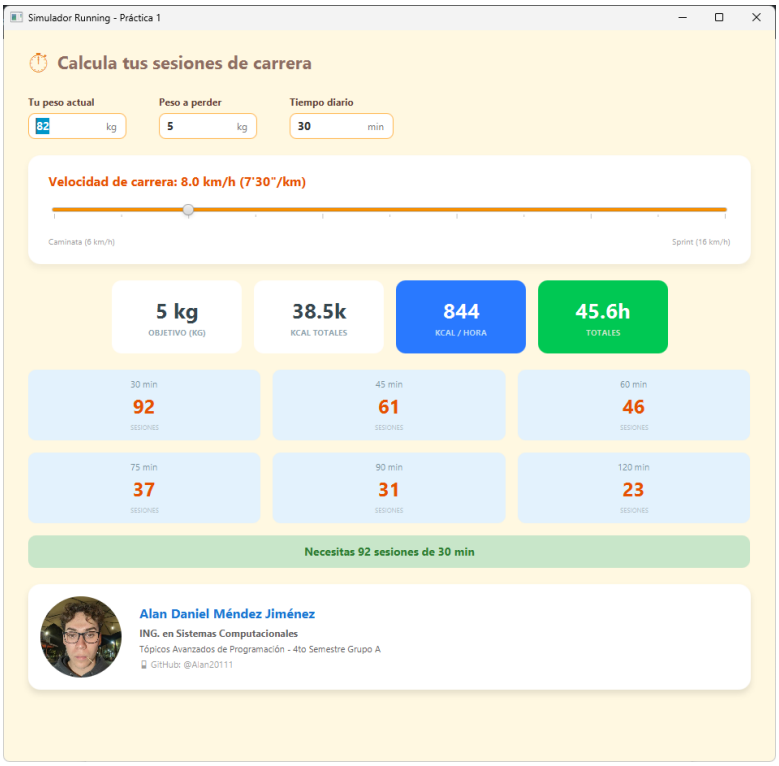


Vista general del Simulador de Running en ejecución

CÁLCULO DINÁMICO DE RITMO (PACE) Y CALORÍAS

Uno de los resultados técnicos más relevantes es la capacidad de respuesta en tiempo real. Al interactuar con el *Slider* de velocidad, la aplicación recalcula instantáneamente el ritmo de carrera (minutos por kilómetro).

Prueba: Se configuró una velocidad de **8.0 km/h**.



Detalle del cálculo de ritmo y configuración de usuario.

Resultado: El sistema calculó automáticamente un ritmo de **7'30"/km** y proyectó un gasto calórico de **844 Kcal/hora** (basado en un peso de 82kg), coincidiendo con las fórmulas del MET implementadas.

Proyección de Sesiones La sección inferior de la interfaz muestra la matriz de sesiones necesarias. El algoritmo de redondeo hacia arriba (Math.ceil) funcionó correctamente para garantizar que el usuario cumpla su meta de 38,500 Kcal (5kg).

Caso de Prueba: Para una meta de 5kg y sesiones de 30 minutos diarios.

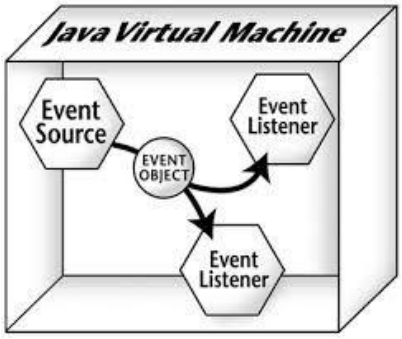
Resultado: El sistema determina que son necesarias **92 sesiones**, lo que concuerda con la división del objetivo total entre el gasto por sesión.

Matriz de resultados mostrando las sesiones requeridas por duración.

CONCLUSIONES

El desarrollo del "Simulador de Running" cumplió exitosamente con el objetivo de aplicar la arquitectura Modelo-Vista-Controlador (MVC) en un entorno JavaFX. Se logró desacoplar la lógica matemática de la interfaz gráfica, garantizando un código modular, limpio y escalable.

La implementación de *Listeners* para realizar cálculos en tiempo real (como el ritmo y las calorías) y el uso de CSS personalizado, permitieron crear una experiencia de usuario fluida y moderna, superando la funcionalidad de los controles básicos. Este proyecto consolidó conocimientos clave sobre el manejo de eventos, la precisión en los tipos de datos y el diseño de software orientado a objetos.



REFERENCIAS

- Google. (2026). *Gemini* (Versión 1.5 Pro) [Modelo de inteligencia artificial]. Recuperado de https://gemini.google.com/
- Oracle. (2025). *JavaFX Documentation: The JavaFX Architecture*. Java Platform, Standard Edition. Recuperado de https://openjfx.io/
- Gradle Inc. (2025). *Gradle User Manual: Building Java Applications*. Gradle Documentation. Recuperado de https://docs.gradle.org/current/userguide/building_java_projects.html
- Baeldung. (2024). *JavaFX with Gradle*. Baeldung Technical Tutorials. Recuperado de https://www.baeldung.com/javafx-gradle

LIBRERIAS

- org.openjfx:javafx-controls

  Descripción: Esta es la librería que contiene todos los botones, *labels*, *sliders* y *text fields*

- org.openjfx:javafx-fxml

  Descripción: Esta librería es la que permite cargar fxml