Documentación de Proyecto de Ingeniería de Software - MultiCalc

Aran Colina (S22002244)
Orlando Madrigal Lagunes (S23002508)
Miguel Yael Montero Guevara (S22002214)

12 de octubre de 2025

Índice

1. Metodología de Desarrollo y Equipo

1.1. Conformación de Equipos y Roles

El equipo de desarrollo está conformado por 3 integrantes y utiliza **Scrum** como marco de trabajo. Todos participan en el desarrollo técnico.

Cuadro 1: Asignación de Roles del Equipo MultiCalc

Rol Principal	**Integrante**	**Habilidades Clave (Justificación)**
Product Owner (PO) / Scrum Master (SM)	Orlando Madrigal Lagunes	Lógica de Backend y Frontend. (Liderazgo y visión general del producto)
Developer (DEV)	Aran Colina	Frontend, diseño y empaquetado. (Responsable de la UI/UX).
Developer (DEV)	Miguel Yael Montero Guevara	Lógica de Backend. (Responsable de los algoritmos y métodos numéricos).

1.2. Responsabilidades de los Roles

Las responsabilidades se definen según las directrices de la asignatura.

Product Owner (PO) / Scrum Master (SM)

- Define visión del producto.
- Construye y prioriza el Product Backlog.
- Aclara requerimientos y valida incrementos.
- Asegura el cumplimiento de Scrum.
- Supervisa avance documental del equipo.

Developer (DEV)

- Implementa, diseña y prueba funcionalidades (Unit Test).
- Desarrolla incrementos funcionales.
- Participa en estimaciones y decisiones técnicas.

2. Product Backlog (PB)

El Product Backlog contiene el listado de Historias de Usuario (HU) y es el único origen de trabajo para el Equipo de Desarrollo.

Cuadro 2: Product Backlog Inicial - MultiCalc

ID	Historia de Usuario (Formato Çomo..., quiero..., para...")	Prioridad	Story Points
HU-01	Como estudiante, quiero crear un perfil de usuario para mantener mi información, foto y racha de práctica.	Alta	5
HU-02	Como estudiante, quiero acceder a la calculadora de derivadas e integrales.	Alta	8
HU-03	Como estudiante, quiero usar el módulo de conversores de unidades físicas para verificar rápidamente mis resultados en problemas de física.	Alta	5
HU-04	Como estudiante, quiero consultar el historial de las gráficas que he generado.	Media	3
HU-05	Como usuario, quiero acceder al módulo de Graficador de Funciones para visualizar el comportamiento de las ecuaciones que estoy estudiando.	Media	8
HU-06	Como estudiante, quiero utilizar el módulo de métodos numéricos para resolver un sistema de ecuaciones no lineales.	Alta	13
HU-07	Como estudiante, quiero generar exámenes personalizados para ponerme a prueba con temas, dificultad y número de preguntas a elegir.	Alta	13
HU-08	Como estudiante, quiero acceder al conversor de sistemas numéricos para traducir valores entre binario, decimal y hexadecimal.	Media	5
HU-09	Como estudiante, quiero configurar el modo de mis exámenes (contra reloj o cronometrado) para practicar bajo condiciones de tiempo específicas.	Media	5
HU-10	Como usuario, quiero tener un módulo para fórmulas importantes.	Baja	3

2.1. Detalle de Historias de Usuario y Criterios de Aceptación

A continuación, se detallan los criterios de aceptación (Definition of Done) para asegurar la calidad de cada Historia de Usuario implementada.

- **HU-01: Perfil de Usuario**
  1. El sistema debe permitir registrar un usuario nuevo validando que el nombre no exista previamente en el JSON.
  2. El usuario debe poder iniciar sesión y ver su nombre, foto y racha recuperados correctamente.
  3. Al cerrar y abrir la app, se le solicita al usuario ingresar sus credenciales.

- **HU-02: Calculadora Simbólica**

  1. El sistema debe aceptar expresiones algebraicas complejas (ej. $x^2 + 2x + 1$).
  2. Debe retornar el resultado simplificado o factorizado correctamente usando `SymPy`.
  3. Debe mostrar un mensaje de error amigable si la sintaxis de la ecuación es inválida.

- **HU-03: Conversor de Unidades Físicas**

  1. El usuario debe poder seleccionar categoría (Longitud, Masa, Tiempo) y unidades de origen/destino.
  2. La conversión debe ser precisa con al menos 4 decimales.
  3. El sistema debe bloquear conversiones entre unidades incompatibles (ej. Metros a Segundos).

- **HU-04: Historial de Gráficas**

  1. Cada gráfica generada debe guardarse automáticamente.
  2. El historial debe ser persistente (guardarse en almacenamiento local) y único por usuario.
  3. El usuario debe tener la opción de limpiar su historial.

- **HU-05: Graficador de Funciones**

  1. El sistema debe generar una imagen (PNG/Bitmap) a partir de una función $f(x)$ válida.
  2. El usuario debe poder definir el rango de visualización (ejes X e Y).
  3. La gráfica debe mostrarse dentro de la interfaz de la aplicación sin cierres inesperados.

- **HU-06: Métodos Numéricos**

  1. El sistema debe implementar correctamente el algoritmo seleccionado (ej. Newton-Raphson).
  2. Debe mostrar las iteraciones o el resultado final de la raíz aproximada.
  3. Debe manejar casos de no-convergencia con un límite de iteraciones.

- **HU-07: Generador de Exámenes**

  1. El usuario debe poder seleccionar la dificultad (Fácil, Medio, Difícil).
  2. El sistema debe generar preguntas aleatorias basadas en los módulos de estudio.
  3. Al finalizar, se debe mostrar una calificación.

- **HU-08: Conversor de Sistemas Numéricos**
    1. Debe permitir la entrada de valores en Binario, Decimal y Hexadecimal.
    2. La conversión debe actualizarse en tiempo real o al presionar calcular para los otros dos sistemas.
    3. Debe validar que la entrada corresponda al sistema base (ej. no permitir '2' en Binario).
- **HU-09: Modos de Examen (Crono/Reloj)**
    1. En modo Çontra Reloj", el examen debe cerrarse automáticamente al terminar el tiempo.
    2. En modo Çronometrado", el sistema debe registrar el tiempo total tomado al finalizar.
    3. El tiempo debe guardarse en las estadísticas de desempeño del usuario.
- **HU-10: Módulo de Formulario**
    1. El usuario debe poder visualizar las fórmulas en formato de imagen.
    2. Las fórmulas deben poder estar clasificadas por temática o asignatura.
    3. El usuario debe tener obtener una breve descripción de la fórmula solicitada.

3. Objetivo y Alcance del Proyecto

3.1. Objetivo del Proyecto

El objetivo principal es **diseñar e implementar un compendio digital de herramientas de cálculo y estudio**, accesible a través de una plataforma unificada. Este proyecto busca proporcionar a los estudiantes de primeros semestres de ingeniería un recurso integral para:

1. **Reforzar su aprendizaje** de conceptos matemáticos fundamentales.
2. **Comprobar la exactitud** de sus resultados de ejercicios y tareas.
3. **Facilitar el estudio y repaso** mediante funcionalidades de autoevaluación (exámenes y formularios).

3.2. Alcance del Proyecto

El alcance de este proyecto incluye el desarrollo de los siguientes módulos funcionales:

- **Calculadora de Funciones Avanzadas:** Integración de una herramienta capaz de manipular y calcular **valores simbólicos** y aplicar **métodos numéricos** específicos.
- **Graficador de Funciones:** Módulo para la visualización gráfica de funciones matemáticas, incluyendo la gestión de un **historial de funciones** graficadas, asociado a cada usuario.
- **Herramientas de Conversión:** Implementación de dos tipos de conversores:
  - Conversor de Unidades Físicas (p. ej., masa, longitud, tiempo).
  - Conversor de Sistemas Numéricos (p. ej., binario, decimal, hexadecimal).
- **Módulo de Evaluación y Estudio:** Desarrollo de una funcionalidad que permita la creación y presentación de **formularios** y **exámenes**. Estos podrán configurarse bajo modalidades de tiempo limitado (**a contra reloj**) o simplemente **cronometradas**, permitiendo la autoevaluación del estudiante.

Nota: El proyecto está específicamente orientado a estudiantes de primeros semestres de cualquier ingeniería y su alcance se limita a las funcionalidades de cálculo, graficación, conversión y evaluación antes mencionadas.

4. Requerimientos del Sistema

Este apartado detalla las capacidades específicas que debe cumplir la aplicación MultiCalc, divididas en requerimientos funcionales (lo que el sistema *hace*) y no funcionales (cómo el sistema *funciona*).

4.1. Requerimientos Funcionales (RF)

Los requerimientos funcionales describen las tareas que el sistema debe ejecutar en sus módulos de cálculo e interacción con el usuario.

- **Cálculo Simbólico:**
  - RF1.1: Calcular derivadas de funciones (ej: $3x^2 + \sin(x) \rightarrow 6x + \cos(x)$).
  - RF1.2: Calcular integrales definidas e indefinidas (ej: $\int 2x dx \rightarrow x^2 + C$).

- **Métodos Numéricos:**
  - RF2.1: Implementar el Método de Bisección para la búsqueda de raíces (requiere ingresar $f(x)$, intervalo, y tolerancia).
  - RF2.2: Implementar el Método de Newton-Raphson para la búsqueda de raíces (requiere ingresar $f(x)$, $f'(x)$, y un punto inicial).
- **Graficador:**
  - RF3.1: Graficar funciones en 2D, soportando hasta 3 funciones simultáneas, con capacidades de zoom y desplazamiento.
- **Conversores:**
  - RF4.1: Ofrecer conversión de unidades físicas (longitud, masa, tiempo).
  - RF4.2: Ofrecer conversión entre sistemas numéricos (binario, decimal, hexadecimal).
- **Gestión de Usuario y Gamificación:**
  - RF5.1: Proporcionar un sistema de registro y login seguro (usuario y contraseña).
  - RF5.2: Guardar y recuperar el historial de gráficas por usuario.
  - RF5.3: Crear exámenes aleatorios según los parámetros ingresados por el usuario (temas, dificultad).

4.2. Requerimientos No Funcionales (RNF)

Los requerimientos no funcionales definen los criterios de calidad y las restricciones del entorno del sistema.

- **Usabilidad (RNF1):** La Interfaz Gráfica de Usuario (GUI) debe ser intuitiva y de fácil navegación para el usuario objetivo (estudiantes de ingeniería).
- **Compatibilidad (RNF2):** El sistema debe asegurar compatibilidad y estabilidad con una amplia gama de versiones de Android, utilizando las APIs compatibles.

5. Tecnologías Utilizadas

- **Backend:** Python con las librerías `sympy` y `numpy`.
- **Frontend:** Kotlin y Android Studio (para empaquetar).

subsectionArquitectura Técnica y Stack de Tecnología

El proyecto MultiCalc se construyó utilizando una arquitectura de doble capa para aprovechar la potencia de cálculo de Python con la plataforma móvil nativa de Android (Kotlin).

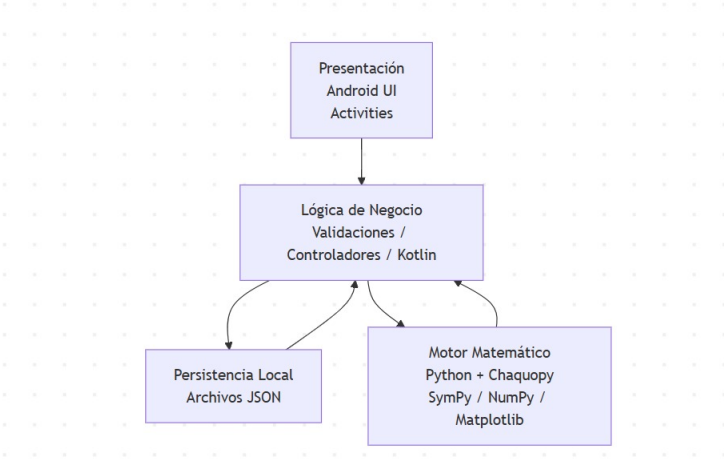

Figura 1: Diagrama de Arquitectura de MultiCalc

5.1. Diseño de Datos y Persistencia

Debido a la naturaleza ligera y portátil de la aplicación, MultiCalc no utiliza una base de datos relacional (SQL). En su lugar, implementa un sistema de persistencia basado en archivos **JSON** estructurados, gestionados por la capa de Backend (Python).

5.2. Esquema Lógico (usuarios.json)

El archivo principal de almacenamiento sigue una estructura de diccionario anidado, donde la clave primaria es el nombre de usuario. A continuación se presenta el esquema de datos:

```
{
  "nombre_usuario_unico": {
    "password_hash": "SHA-256 string",
    "es_admin": boolean,
    "racha_dias": integer,
    "foto_url": "string (path local)",
    "cursos_completados": {
      "Simbolico": integer (0-100),
      "Metodos_Numericos": integer (0-100),
      "Conversores": integer (0-100)
    },
    "stats_examenes": {
      "racha_examen": integer,
      "facil_completados": integer,
      "medio_completados": integer,
```

```
      "dificil_completados": integer
    }
  }
}
```

Justificación Técnica: Este modelo NoSQL permite una lectura rápida mediante la librería `json` de Python y facilita la serialización de objetos `Usuario` entre las capas de Kotlin y Python sin necesidad de un ORM complejo.

1. **Capa de Presentación (Frontend):** Desarrollada en **Kotlin** (Android Nativo). Esta capa maneja la interfaz de usuario (UI/UX), la navegación, el sistema de autenticación (actividades de Login/Perfil) y la gestión de la persistencia básica (Shared Preferences).
2. **Capa de Lógica de Negocio (Backend):** Desarrollada en **Python 3.x**. Esta capa aloja la lógica pesada, incluyendo:
   - Algoritmos de Métodos Numéricos (`numpy`).
   - Cálculo Simbólico (Derivadas, Integrales, Ecuaciones) utilizando la librería `SymPy`.
   - Generación de Gráficas de Funciones (se genera una imagen PNG) utilizando librerías como `matplotlib`.
   - Persistencia de usuarios en formato `JSON` (`auth_manager.py`).

6. UI/UX

6.1. Flujo de pantallas

Se adjunta el siguiente link con la imagen de los flujos de pantalla era demasiado grande.
https://uvmx-my.sharepoint.com/:f:/g/personal/zs22002244_estudiantes_uv_mx/IgBGE7BDYyp2RaouadHDx57wAaHiokfLJmp4SAj6fN-qaJI?e=IBChyd.

6.2. Estilo visual y decisiones de diseño

Para el diseño de la app consideramos en el logotipo un estilo colorido representando los módulos principales de la aplicación con un colores distintivos para que el usuario asocie algunos colores inmediatamente con los módulos, los botones tienen un estilo visual redondeado y la fuente que se utilizó en su mayoría corresponde al MonoSpace genérico de Android Studio.

Figura 2: Logotipo de MultiCalc

7. Bitácora de Decisiones Técnicas (ADR)

Registro de las decisiones arquitectónicas clave tomadas durante el desarrollo para resolver impedimentos.

- **Decisión 1 (Sprint 1): Uso de JSON sobre SQLite.**
  - *Contexto:* Necesidad de persistencia rápida y flexible para datos no relacionales complejos.
  - *Decisión:* Usar archivos JSON planos gestionados por Python.
  - *Resultado:* Desarrollo más rápido del módulo de perfil, aunque requiere cuidado con la concurrencia de escritura.

- **Decisión 2 (Sprint 1): Integración Chaquopy.**
  - *Contexto:* Kotlin no tiene librerías simbólicas nativas tan potentes como SymPy.
  - *Decisión:* Integrar el plugin Chaquopy para ejecutar Python dentro de Android.
  - *Resultado:* Éxito en funcionalidad matemática, pero aumentó el tamaño del APK y la complejidad del *build*.
- **Decisión 3 (Sprint 3): Estructura de Directorios para Gráficas.**
  - *Contexto:* Las gráficas generadas se sobrescribían o no eran accesibles por usuario.
  - *Decisión:* Crear subdirectorios dinámicos basados en el nombre de usuario (`/files/user/graficas/`).
  - *Resultado:* Aislamiento correcto de datos y privacidad entre usuarios en el mismo dispositivo.