**Informe de Paradigmas de Programación en Divergesia**

**Autor: Eldis Dennys González Pérez**

**Como artista plástico aficionado, mi proceso creativo implica múltiples etapas: exploración de conceptos, experimentación con materiales, prueba de composiciones y estudio de la luz y el color. En cada fase, busco formas de capturar y organizar mis hallazgos cromáticos para aplicarlos de modo coherente en mis obras. Por eso he concebido una aplicación de escritorio que facilite la extracción, el ajuste y la gestión de paletas de colores a partir de propias imágenes de referencia: permite guardar variaciones según cada proyecto, afinar tonos con precisión y mantener todo el trabajo local, respetando la confidencialidad del proceso de los artistas.**

**Divergesia es una aplicación de escritorio desarrollada en Python con PySide6, orientada al diseño y gestión de paletas de colores. Permite al usuario cargar imágenes, extraer los colores dominantes mediante técnicas de clustering, ajustar matiz, saturación y valor (HSV), generar armonías cromáticas (complementarios, análogos, triadas, etc.), visualizar un mockup de la paleta, y exportar o importar los resultados en formatos JSON, CSS y CSV.**

**Paradigma Imperativo**

**El núcleo de la interacción del usuario está construido con programación imperativa. La clase MainWindow, ubicada en el módulo ui/main\_window.py, organiza y controla la interfaz gráfica. Allí, los eventos como cargar una imagen, presionar un botón o seleccionar una paleta se manejan con métodos secuenciales (on\_cargar\_imagen, on\_guardar\_paleta, etc.), que modifican directamente el estado de la aplicación (self.paleta\_actual, self.lista\_paletas, etc.). Este enfoque permite actualizar componentes visuales como vistas de imagen, mockup y swatches de color en función de las acciones del usuario de forma clara y controlada.**

**Paradigma Funcional**

**La lógica de procesamiento de color se basa en funciones puras, lo que representa el paradigma funcional. En módulos como logic/color\_utils.py, logic/harmony.py y logic/mockup.py, se definen funciones que transforman datos sin producir efectos colaterales. Por ejemplo, adjust\_palette\_hsv devuelve una nueva lista de colores ajustados en HSV, sin modificar la lista original; generar\_complementarios y otras funciones de armonía reciben una paleta base y generan nuevas combinaciones de forma determinista. Asimismo, crear\_mockup\_ui genera una imagen a partir de colores de entrada sin alterar el estado global, facilitando pruebas y mantenibilidad.**

**Paradigma Asincrónico**

**Para mantener una interfaz fluida, las tareas intensivas se ejecutan en segundo plano usando hilos (QThread), cumpliendo con el paradigma asincrónico. En async\_tasks/workers.py, se definen ClusteringWorker y ExportWorker, que realizan operaciones como análisis de imagen o guardado de archivos sin bloquear la interfaz. La comunicación entre los hilos y la UI se realiza mediante señales y slots (resultado, error, finished), lo que permite actualizar los elementos visuales solo cuando la operación ha terminado, sin interferencias ni bloqueos perceptibles por el usuario.**

**Conclusión**

**Divergesia integra de forma coherente los tres paradigmas de programación. El paradigma imperativo organiza la interfaz y su comportamiento; el funcional asegura que la lógica de color sea predecible y testeable; y el asincrónico garantiza una experiencia de usuario fluida, incluso al ejecutar operaciones complejas. Esta combinación da lugar a una arquitectura modular, robusta y fácil de mantener.**

**Enlace de Repositorio de GitHub: https://github.com/Dennys1701/Divergesia**