# DOCUMENTACIÓN:

# Programa 3 - Kakuro


**Estudiante:**

Daniel Alejandro Arrieta Víquez


**Semestre y año:**

I semestre 2025

**Curso:**

Taller De Programación

**Grupo:**

5


**Profesor:**

William Mata Rodríguez.

**Institución:**

Instituto Tecnológico de Costa Rica.


TEC | Tecnológico de Costa Rica

# Contenido:

# Enunciado del Proyecto:

**Enunciado del Proyecto:**
Desarrollar una aplicación en Python usando Tkinter para el juego de lógica Kakuro, que permita al usuario interactuar con una cuadrícula 9x9 de sumas cruzadas mediante una interfaz gráfica, personalizable y completa. Incluyendo funciones de configuración, cronómetro o temporizador, gestión de jugadas (deshacer/rehacer), guardar/cargar partidas, control de récords y validaciones estrictas para garantizar el cumplimiento de las reglas del juego.

**Objetivo general:**
Diseñar e implementar un sistema completo e interactivo para el juego Kakuro, que permita al usuario configurar el entorno de juego, resolver tableros con sumas cruzadas respetando las reglas matemáticas, y registrar sus tiempos como récords, todo con una experiencia gráfica atractiva.

**Objetivos específicos:**

- Aplicar estructuras de datos como listas, diccionarios y pilas para la gestión del estado del juego.
- Permitir que el usuario configure el nivel de dificultad y el tipo de reloj (cronómetro, temporizador o sin reloj).
- Validar cada jugada según las reglas del Kakuro (sin repetir números, sumas correctas).
- Implementar funciones para deshacer y rehacer jugadas usando pilas.
- Mostrar un tablero 9x9 de forma clara y visualmente atractiva.
- Guardar y cargar partidas en archivos JSON para mantener la persistencia.
- Registrar récords por nivel, jugador y tiempo, y mostrarlos ordenados ascendentemente.
- Documentar todo el proyecto, incluyendo su código fuente y el manual de usuario.
- Utilizar herramientas de IA de manera ética, consciente y crítica para el apoyo en diseño, codificación y documentación.

**Requerimientos funcionales:**

- Cargar partidas de un archivo con claves numéricas y disposición de tablero por nivel de dificultad.
- Generar tableros aleatorios sin repetir hasta agotar el conjunto por nivel.
- Validar jugadas (número repetido, sumas incorrectas por fila o columna).
- Permitir ingresar y borrar números sobre casillas válidas.
- Iniciar juego solo después de ingresar el nombre del jugador.
- Soporte completo para:
    - Guardar partida actual.
    - Cargar partida existente por nombre.

    - Finalizar o borrar el juego con confirmación.
    - Mostrar récords por nivel, jugador o todos.
- Control de tiempo:
    - Temporizador con validación de tiempo ingresado.
    - que inicia al presionar “Iniciar juego”.
    - Manejo automático si el tiempo se agota.
- Registro de jugadas realizadas y jugadas deshechas mediante pilas.
- Generación y almacenamiento de récords en un archivo.
- Configuración persistente del juego guardada en archivo JSON.

**Requerimientos no funcionales:**

- Interfaz gráfica desarrollada con Tkinter, con estilo visual definido
- Validación completa de entradas del usuario.
- Uso de archivos JSON para configuraciones, partidas guardadas y récords.
- Código modular, limpio, con nombres significativos y documentado con comentarios.
- División del código en funciones/módulos reutilizables bajo el principio “divide y vencerás”.
- Uso del software Git y Github como sistema de control de versiones.

# Temas investigados:

Durante el desarrollo del sistema del juego de lógica Kakuro, se investigaron y aplicaron diversos conceptos técnicos con el fin de implementar una solución completa, funcional y amigable para el usuario. A continuación, se describen los principales aspectos abordados:

**Software de control de versiones y colaboración ¿Qué es y cuál es su importancia?:**

El software de control de versiones permite gestionar y registrar de forma organizada los cambios realizados en los archivos de un proyecto a lo largo del tiempo. Es fundamental en el desarrollo de software moderno porque:

- Facilita la colaboración entre varios programadores de forma simultánea.
- Permite mantener un historial completo del código fuente, incluyendo qué cambios se hicieron, cuándo y por quién.
- Ayuda a detectar errores y revertir versiones fallidas sin perder el trabajo anterior.
- Mejora la organización y la calidad del desarrollo al dividir el proyecto en versiones o ramas (branches) para experimentar sin afectar la versión principal.

En este proyecto de Kakuro, el control de versiones permitió mantener un seguimiento ordenado del desarrollo de funcionalidades como la cuadrícula, la validación de sumas, el temporizador y el manejo de archivos. También garantizó respaldo constante y facilitó la prueba de nuevas ideas sin temor a romper el código base.

**Fuente:** https://github.com/resources/articles/software-development/what-is-version-control

**Ejemplos de software de control de versiones:**

**1. Git**
Desarrollado por Linus Torvalds en 2005, Git es un sistema de control de versiones distribuido y el más utilizado hoy en día. Permite crear múltiples ramas locales y remotas, combinar cambios, colaborar en paralelo y resolver conflictos.
Fue la herramienta utilizada en este proyecto para:

- Registrar avances importantes.
- Mantener respaldo en GitHub.
- Crear versiones estables y experimentar con nuevas funcionalidades.

**2. Mercurial**
Es una alternativa distribuida a Git, enfocada en la simplicidad. Aunque no fue utilizada en este proyecto, se estudió por su enfoque intuitivo y su menor curva de aprendizaje, ideal para proyectos con colaboradores menos técnicos.

**3. Perforce**
Sistema de control centralizado, diseñado para grandes empresas y equipos que manejan muchos archivos o recursos gráficos, como videojuegos. Ofrece rendimiento sobresaliente pero requiere mayor conocimiento técnico.

**Fuente:** https://sentrio.io/blog/herramientas-control-de-versiones-devops/

**Git y GitHub:**

Git es la herramienta que se instala localmente para gestionar cambios en los archivos de un proyecto.
GitHub es una plataforma en la nube donde se almacenan repositorios Git, se comparten, y se colabora con otros usuarios.

**Fuente:** https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

**Comandos de Git utilizados en este proyecto:**

Durante el desarrollo del juego Kakuro, se utilizaron los siguientes comandos de Git,entre otros:

- git push -u origin main
- git commit –m
- git add
- git status
- git init
- ls
- git –version

**Nuevos componentes de Tkinter utilizados en este proyecto:**

Comparado con el proyecto anterior de **parqueos callejeros**, que usaba principalmente Button, Label, Entry, messagebox y Tk, este proyecto **Kakuro** implicó una interfaz más visual, dinámica y centrada en la interacción gráfica directa con un tablero. Por eso se incorporaron **nuevos componentes y técnicas de Tkinter**:

| Componente | Descripción | Uso en Kakuro |
|---|---|---|
| Canvas | Área para dibujar gráficos personalizados | Se utilizó para construir visualmente la cuadrícula 9x9 y dibujar claves numéricas |
| Frame | Contenedor para organizar elementos visuales | Se usó para dividir áreas del juego: panel numérico, temporizador, tablero |
| Label + StringVar() | Etiquetas dinámicas | Se utilizó para mostrar el cronómetro o temporizador actualizado en tiempo real |
| messagebox.askyesno() | Cuadro de diálogo con opciones Sí/No | Validación para borrar, terminar o guardar juego |
| after() | Método para ejecutar funciones repetidamente tras un retardo | Implementación del cronómetro que se actualiza cada segundo |

Estos componentes enriquecieron la experiencia del jugador y permitieron construir una interfaz visualmente cercana a un tablero real de juego, lo cual representa una evolución significativa con respecto al diseño más clásico del sistema de parqueo.

# Recursos utilizados en el proyecto

Durante el desarrollo del sistema se consultaron y utilizaron los siguientes recursos técnicos, además del apoyo de herramientas de inteligencia artificial:

1. **Git y GitHub**

**Marco teórico:**
Git es un sistema de control de versiones distribuido que permite a múltiples desarrolladores trabajar simultáneamente en un proyecto sin sobrescribir el trabajo de otros. Guarda el historial de cambios, permite comparar versiones, revertir errores y trabajar con ramas.
GitHub es una plataforma basada en la nube donde se alojan repositorios Git. Facilita la colaboración remota, la gestión de versiones, y el trabajo en equipo con herramientas como pull requests, issues y control de ramas.

**Aplicación en el proyecto:**
Durante el desarrollo del sistema de gestión de parqueos y del juego Kakuro, Git fue fundamental para organizar y respaldar los cambios del código fuente. Se usaron comandos como:

git init, git add, git commit -m, git push -u origin main, git status, git --version, etc.
GitHub permitió almacenar el proyecto en línea, compartirlo y sincronizarlo desde distintos dispositivos.

**Recursos utilizados:**

https://www.youtube.com/watch?v=DinilgacaWs

https://www.youtube.com/watch?v=rBPBBCpqP5s

https://www.youtube.com/watch?v=Z6VM-Gp3OGw&list=PL-gX0xg7VLB-1O02yLPCBsPUZyV_c9Owg

https://www.youtube.com/watch?v=MAHkItbZD5c

https://www.youtube.com/watch?v=44ziZ12rJwU&list=PL-gX0xg7VLB-1O02yLPCBsPUZyV_c9Owg&index=9

https://git-scm.com/downloads/win

https://www.youtube.com/watch?v=ytSoabxSQ6E

https://www.youtube.com/watch?v=VdGzPZ31ts8

https://www.solucionex.com/blog/borrar-ultimo-commit-con-reset-y-revert-en-git

https://es.stackoverflow.com/questions/320220/borrar-un-commit-en-github

https://es.stackoverflow.com/questions/1267/c%C3%B3mo-modificar-el-mensaje-de-un-commit-en-particular

https://www.reddit.com/r/git/comments/117itus/how_to_add_a_description_to_a_commit_during_an/?tl=es-419

2. **Variables globales en Python**

**Marco teórico:**
Una variable global es aquella que se define fuera de cualquier función y puede ser utilizada en cualquier parte del programa. Son útiles para mantener información compartida entre funciones, pero deben utilizarse con cuidado para evitar errores o colisiones.

**Aplicación en el proyecto:**
Se utilizaron variables globales para mantener el estado del usuario logueado, controlar el tiempo en el juego Kakuro y compartir configuraciones entre ventanas, como el nivel de dificultad o la información del tablero.

**Recursos utilizados:**

https://www.w3schools.com/python/python_variables_global.asp

https://www.freecodecamp.org/espanol/news/variables-globales-en-python/

3. **Tkinter avanzado: diseño, fuentes y estética**

**Marco teórico:**
Tkinter es una biblioteca gráfica de Python que permite crear interfaces de usuario con botones, etiquetas, menús y más. Para mejorar la experiencia visual se pueden cambiar colores, bloquear el tamaño de las ventanas, personalizar fuentes y usar lambda para funciones dinámicas.

**Aplicación en el proyecto:**
Se personalizó el fondo de las ventanas del juego Kakuro con colores específicos, se definieron fuentes modernas para los textos, y se bloqueó el tamaño de la ventana principal con resizable(False, False) para mantener la estética. También se usaron expresiones lambda para asignar funciones interactivas a botones del tablero.

**Recursos utilizados:**

https://es.stackoverflow.com/questions/321609/como-cambia-el-color-de-fondo-en-python-con-tkinter

https://www.reddit.com/r/Python/comments/z8cvus/list_of_fonts_for_tkinter/?tl=es

https://stackoverflow.com/questions/39614027/list-available-font-families-in-tkinter

https://www.tutorialspoint.com/how-can-i-prevent-a-window-from-being-resized-with-tkinter

https://www.tutorialspoint.com/tkinter-button-commands-with-lambda-in-python

### 4. Algoritmo de selección aleatoria utilizado

La función random.choice() se basa en el generador de números pseudoaleatorios de Python, se utilizó para elegir tableros.

**Recursos utilizados:**

https://www.geeksforgeeks.org/python/random-choices-method-in-python/

https://www.w3schools.com/python/ref_random_choice.asp

https://docs.python.org/3/library/random.html

# Uso de IA:

| Objetivo del uso | Herramienta utilizada | Prompt o pregunta | Respuesta de la IA | ¿Cómo se adaptó? | Reflexión crítica | Otros |
|---|---|---|---|---|---|---|
| Diseñar el tablero 9x9 con claves numéricas válidas y sin errores | ChatGPT | ¿Cómo puedo generar un tablero Kakuro 9x9 con sumas válidas para cada fila y columna? | Código base para crear estructuras de tablero con control de claves y sumas permitidas | Se usó como guía para diseñar tableros con sumas posibles, sin repeticiones en grupos, y evitando sumas imposibles | Muy útil para garantizar la jugabilidad real del tablero | |
| Validar jugadas del usuario y restricciones de Kakuro | ChatGPT | ¿Cómo valido que un número no se repita en un grupo y que la suma coincida con la clave en Kakuro? | Función de validación que revisa unicidad y suma total por grupo horizontal o vertical | Se integró al evento de ingresar número en Tkinter para validar jugada en tiempo real | Fundamental para cumplir las reglas del juego y mejorar la experiencia del usuario | |
| Implementar cronómetro con pausa automática | ChatGPT | ¿Cómo creo un cronómetro en Tkinter que se pueda pausar y continuar según eventos? | Ejemplo de implementación de cronómetro con `after()` y funciones de pausa/reanudar | Se usó para medir el tiempo de juego y detenerlo cuando se abre el menú de récords | Muy útil para dar una experiencia más profesional y justa al jugador | |
| Mostrar récords por nivel y jugador con orden ascendente | ChatGPT | ¿Cómo puedo ordenar una lista de récords por tiempo y mostrar solo los mejores de cada jugador? | Ejemplo con `sorted()` y claves personalizadas de ordenamiento | Se usó para leer archivo de récords y mostrar los top por nivel y jugador en pantalla aparte | Permitió crear una tabla clara y justa para fomentar competencia entre usuarios | |
| Crear interfaz estilo moderno con colores, fuentes y distribución personalizada | ChatGPT | ¿Cómo adapto una interfaz Tkinter con fondo oscuro, botones grandes y estilo limpio como esta imagen? | Propuesta de diseño usando `place()`, colores personalizados y fuentes modernas | Se adaptó para que la interfaz del tablero, panel de números y botones siga el estilo mostrado | Muy útil para atraer visualmente y ofrecer una estética profesional | |

# Conclusiones del Trabajo:

El desarrollo del sistema de juego Kakuro con interfaz gráfica en Python fue una experiencia enriquecedora que integró principios de programación modular, validación lógica de reglas y diseño de interfaces con Tkinter. A lo largo del proyecto se logró implementar un juego completo con tablero de 9x9, claves numéricas coherentes, validaciones en tiempo real, control de errores y un sistema de récords por jugador y dificultad, generando así una experiencia interactiva y funcional.

La interfaz gráfica fue diseñada cuidadosamente siguiendo una estética retro, con botones grandes, colores llamativos y elementos visuales bien distribuidos, inspirándose en referencias visuales. Además, se implementó un cronómetro en tiempo real, con pausa automática al consultar los récords, lo cual mejoró la jugabilidad y profesionalismo del juego.

El sistema se diseñó de forma completamente modular, lo que permitió desarrollar componentes separados para el menú principal, el tablero del juego, los controles de ingreso, validación de jugadas, y la gestión de récords. Se empleó el almacenamiento persistente mediante archivos JSON, lo que permite conservar los mejores tiempos entre sesiones.

También se hizo un uso responsable y estratégico de herramientas de inteligencia artificial (IA), las cuales brindaron apoyo en el diseño de la estructura del tablero, la lógica de validación de sumas, la implementación del temporizador, y el diseño visual. Estas herramientas sirvieron de guía técnica y solución a bloqueos durante el desarrollo, sin sustituir el criterio ni la decisión final del programador.

**Problemas Encontrados y Soluciones**

- **Generación de claves imposibles o tableros inválidos:** Al inicio, los tableros contenían claves numéricas que no podían lograrse con la cantidad de casillas disponibles

**Solución:** Se diseñó un generador de claves que respeta los límites matemáticos del juego y se revisaron manualmente los tableros generados.

- **Diseño visual poco atractivo y confuso:** La primera versión del tablero era básica, con botones pequeños y fondo blanco.

  **Solución:** Se rediseñó completamente la interfaz siguiendo una imagen de referencia, aplicando colores oscuros, botones grandes y estilos uniformes.

- **Errores de validación lógica de jugadas:** Las primeras validaciones no evitaban correctamente repeticiones ni detectaban errores de suma.

  **Solución:** Se mejoró la lógica con condiciones que comparan listas de celdas por grupo, unicidad y comparación de sumas acumuladas vs clave.

- **Falta de pausa del cronómetro al consultar récords:** El cronómetro seguía corriendo, aunque el jugador no estuviera jugando.

  **Solución:** Se programó la pausa automática del cronómetro al abrir la ventana de récords, y su reanudación al cerrarla.

- **Problemas al guardar los récords por nivel y jugador:** Al inicio no se diferenciaban bien los tiempos por jugador ni por dificultad.

  **Solución:** Se diseñó una estructura en JSON que guarda los récords por nombre y nivel, permitiendo ordenarlos y consultarlos fácilmente.

# Estadística de tiempos:

| Actividad realizada | Horas |
|---|---|
| Análisis del problema | 2 |
| Diseño de algoritmos | 3 |
| Investigaciones | 4 |
| Programación | 6 |
| Documentación interna | 2:30 |

| | |
|---|---|
| Pruebas | 11 |
| Elaboración del manual de usuario | 5 |
| Elaboración de documentación del proyecto | 4 |
| **TOTAL** | 32 horas aprox |

**Aprendizajes Obtenidos**

- Desarrollo de interfaces gráficas con Tkinter adaptadas a un juego lógico: se implementó una interfaz visualmente atractiva con menús principales, temporizador, tablero de juego interactivo, selección de números, validaciones visuales y pantallas emergentes al completar el juego.

- Aplicación del diseño modular en Python, separando el sistema en módulos independientes como el menú principal, la lógica del tablero, la validación del juego y la gestión de récords. Esto facilitó el mantenimiento, la reutilización del código y permitió escalar el sistema con nuevas funciones como niveles o desafíos.

- Manejo de archivos JSON como medio de almacenamiento persistente, utilizado para guardar configuraciones, tableros predefinidos, y récords de usuarios con tiempo y nivel alcanzado. Se trabajó con lectura, escritura y actualización de datos de forma estructurada.
-
- Validación de reglas específicas del juego de Kakuro, aplicando lógica para comprobar sumas correctas, evitar repeticiones y asegurar que cada jugada cumpla con los requisitos del tablero, lo cual representó un reto de programación lógica con estructuras condicionales y listas.

- Uso de herramientas de inteligencia artificial (IA) como apoyo en la depuración de errores, refactorización de código, generación de ejemplos de validación, diseño de tableros y asistencia en el diseño visual del sistema, sin reemplazar la toma de decisiones ni el diseño lógico del estudiante.

- Gestión del tiempo y estructura de juego, implementando un cronómetro en tiempo real con pausas automáticas durante visualizaciones externas (como los récords), lo que reforzó habilidades en el manejo de eventos y control del flujo del programa.

# Lista de Revisión del Proyecto:

| Concepto | Puntos | Avance (100/%/0) | Puntos obtenidos | Análisis de resultados |
|---|---|---|---|---|
| Opción Jugar: despliegue ventana de juego | 15 | 100% | 15 | Ventana de juego funciona correctamente con tablero dinámico. |
| Seleccionar partida | 5 | 100% | 5 | La selección de partida permite cargar niveles correctamente. |
| Botón Iniciar Juego | 15 | 100% | 15 | Inicia el juego y el temporizador sin errores. |
| Creación del archivo de Récords | 5 | 100% | 5 | Se generan y actualizan récords en archivo externo correctamente. |
| Borrar casillas | 2 | 100% | 2 | Funcionalidad para limpiar casillas implementada y estable. |
| Botón Deshacer Jugada | 8 | 100% | 8 | Deshace jugadas correctamente manteniendo estado del tablero. |
| Botón Rehacer Jugada | 8 | 100% | 8 | Rehace jugadas previamente deshechas sin fallos. |
| Botón Borrar Juego | 2 | 100% | 2 | Limpia completamente el tablero para iniciar nuevo juego. |

| | | | | |
|---|---|---|---|---|
| Botón Terminar Juego | 2 | 100% | 2 | Finaliza la partida y valida el resultado correctamente. |
| Botón Récords | 8 | 100% | 8 | Muestra los récords de forma ordenada y pausando el temporizador. |
| Botón Guardar Juego | 5 | 100% | 5 | Guarda estado actual del juego en archivo para cargar después. |
| Botón Cargar Juego (incluye el despliegue del mismo) | 10 | 100% | 10 | Carga partidas guardadas sin errores ni pérdida de información. |
| Opción Configurar | 5 | 100% | 5 | Permite configurar parámetros básicos y ajustar nivel. |
| Ayuda en el programa: Manual de usuario | 5 | 100% | 5 | Manual claro e integrado para facilitar uso a los jugadores. |
| Cronómetro o Temporizador en tiempo real | 5 | 100% | 5 | Cronómetro visible y funcional con pausa automática en diálogos. |
| TOTAL | 100 | 100% | 100 | |

# Partes desarrolladas adicionales:

- Interfaz estilo retro con colores, fuentes y botones personalizados según la imagen de referencia.
- Validación de jugadas según reglas reales de Kakuro (suma correcta, sin repetir dígitos).
- Tableros organizados por dificultad (fácil a experto) generados manualmente para evitar errores de lógica.
- Opción de ver récords solo del usuario actual (“Yo”) además de globales.
- Control de finalización de juego automático cuando se completa el tablero correctamente.
- Diseño modular del sistema en archivos separados (ej. tablero, cronómetro, gestor de récords).

- Temporizador con pausa automática al abrir ventanas emergentes (ej. récords o ayuda).