Documentación

Kakuro

Carrera: Ingeniería en Computación

Curso: Taller de Programación

Autor: Santiago Villarreal Arley

Carné: 2025120897

Profesor: William Mata

Fecha de entrega: 30 de junio 2025

Tabla de contenidos

1. Enunciado del Proyecto

Nombre del proyecto:
KAKURO

Objetivo general del proyecto:

Desarrollar una aplicación de escritorio interactiva con interfaz gráfica en Python (usando tkinter) que permita jugar Kakuro (crucigrama de sumas) cumpliendo con reglas específicas, usando programación estructurada, estructuras de datos, manejo de archivos y buenas prácticas de desarrollo de software.

Objetivos específicos:

Aplicar el ciclo completo del desarrollo de software: análisis, diseño, codificación, pruebas y evaluación.

Utilizar estructuras de control, listas anidadas, pilas y manejo de archivos JSON.

Integrar una GUI funcional y clara, que represente el tablero de Kakuro y permita al jugador interactuar con ella.

Validar jugadas en tiempo real, según las claves numéricas de fila y columna.

Agregar funcionalidades como deshacer/rehacer, guardar/cargar partidas, temporizador y cronómetro.

Mantener registros históricos de los mejores tiempos por nivel (récords).

Implementar persistencia usando archivos para configuración, partidas y estado del juego.

Incorporar funcionalidades auxiliares: borrar juego, mostrar récords, ayuda y salida.

Definición del juego:

Kakuro es un juego de lógica similar a un crucigrama, pero con sumas. Cada casilla vacía debe llenarse con un número del 1 al 9, de forma que la suma total de una fila o columna coincida con una clave numérica dada. No se permite repetir números dentro del mismo bloque de suma. El sistema debe validar las reglas y determinar cuándo se ha completado correctamente la partida.

2. Temas Investigados

1. Software de control de versiones y colaboración

Marco teórico:
Un software de control de versiones permite gestionar el historial de cambios de un proyecto de software. Ofrece la capacidad de volver a versiones anteriores, comparar diferencias, trabajar en paralelo y resolver conflictos. Es esencial para equipos de desarrollo, ya que mejora la organización, reduce errores y promueve el trabajo colaborativo.

Uso en el proyecto:
Se utilizó **Git** para llevar un control ordenado por minitareas (commits). Cada avance fue registrado con un mensaje claro. Git permitió experimentar sin riesgo, y en combinación con GitHub, sirvió para respaldar y visualizar el proyecto en línea.

Lugar de investigación:
https://www.atlassian.com/git/tutorials/what-is-version-control

Uso de IA

Elemento	Contenido
Objetivo del uso	Entender cómo estructurar un flujo de trabajo con Git basado en minitareas para el proyecto Kakuro
Herramienta utilizada	ChatGPT (ChatPRD)
Prompt o pregunta	"¿Cómo puedo usar Git para llevar el control por módulos o funcionalidades pequeñas en un proyecto de interfaz gráfica?"
¿Cómo usó o adaptó la respuesta?	Se adaptó el consejo de "commits atómicos" y se usó git status / git log como se recomendó. La explicación fue combinada con práctica.
Reflexión crítica	Me ayudó a organizarme mejor. Ya conocía Git, pero esta estructura más disciplinada me sirvió para llevar mejor control del proyecto.
Otros	Se contrastó la información con la documentación oficial de Git para confirmar detalles.

2. Mención y explicación de tres software de control de versiones

Marco teórico:
Los sistemas de control de versiones permiten registrar, comparar y revertir cambios en archivos. Existen distintos tipos, siendo los más populares:

- **Git:** Distribuido, rápido y usado masivamente en la industria.
- **Subversion (SVN):** Centralizado; todo el historial se guarda en un servidor único.
- **Mercurial:** También distribuido, con sintaxis más simple que Git, pero menos popular.

Uso en el proyecto:
Se eligió Git por su velocidad, soporte multiplataforma, documentación abundante y porque es compatible con plataformas de colaboración como GitHub.

Lugar de investigación:
https://www.git-scm.com/, https://subversion.apache.org/, https://www.mercurial-scm.org/

3. Diferencia entre Git y GitHub

Marco teórico:
Git es una herramienta de control de versiones que se ejecuta localmente. GitHub es una plataforma en la nube que aloja repositorios Git y ofrece herramientas sociales para colaboración, revisión de código, issues, entre otras. Es la principal en el mundo y alberga la mayoría de repositorios Open Source.

Uso en el proyecto:
Se usó Git para versionar y GitHub para almacenar, visualizar y respaldar el avance. El repositorio permitió compartir el progreso con el docente y mantener una copia segura del trabajo.

Lugar de investigación:
https://docs.github.com/es/get-started/using-git/about-git

Uso de IA

Elemento	**Contenido**
Objetivo del uso	Redactar una explicación clara y académica sobre la diferencia entre Git y GitHub
Herramienta utilizada	ChatGPT (ChatPRD)
Prompt o pregunta	"¿Cuál es la diferencia entre Git y GitHub? Explicámelo como si fuera para incluir en un informe de universidad."
¿Cómo usó o adaptó la respuesta?	Se editó para adaptarla al estilo del documento, se mantuvieron los ejemplos, pero se cambiaron las frases por un lenguaje más técnico.
Reflexión crítica	Fue útil para evitar explicaciones confusas. Reforzó conocimientos y me ayudó a explicarlo con más claridad en la documentación.
Otros	Se verificaron los conceptos en la documentación oficial de GitHub.

4. Características de Git utilizadas en el proyecto

Marco teórico:
Git ofrece múltiples comandos y flujos de trabajo que permiten gestionar versiones. Entre los más comunes están: git init, git add, git commit, git push, git status, git log.

Uso en el proyecto:
Durante el desarrollo de Kakuro se usaron:

- git init para iniciar el repositorio
- git add . y git commit -m para registrar avances por minitarea
- git log para ver el historial
- git status para ver cambios pendientes
- git push para sincronizar con GitHub

Esto permitió tener trazabilidad de cada cambio y una estructura clara para corregir errores o implementar mejoras.

Lugar de investigación:
https://git-scm.com/docs

5. Nuevos componentes de tkinter usados en el proyecto

Marco teórico:
Tkinter es la biblioteca oficial de Python para crear interfaces gráficas. Permite usar múltiples widgets como botones, etiquetas, entradas de texto, y más. También incluye utilidades como temporizadores (after()), cuadros de diálogo y ventanas emergentes.

Uso en el proyecto:
Se utilizaron componentes nuevos que no se habían trabajado en cursos previos:

- tk.simpledialog.askstring(): para capturar el nombre del jugador
- after(): para implementar un temporizador que descuenta segundos
- messagebox: para mostrar mensajes de éxito, error o advertencia
- tk.Frame: para distribuir y modularizar visualmente la UI
- tk.Button con comandos dinámicos: celdas interactivas en el tablero

Lugar de investigación:
https://tkdocs.com/tutorial/
https://docs.python.org/3/library/tkinter.html

Uso de IA

Elemento	**Contenido**
Objetivo del uso	Encontrar componentes de Tkinter útiles para crear un temporizador y una grilla interactiva tipo tablero
Herramienta utilizada	ChatGPT (ChatPRD)
Prompt o pregunta	"¿Qué componentes de tkinter me permiten crear un tablero con celdas interactivas y un temporizador visual para juegos?"
¿Cómo usó o adaptó la respuesta?	Probé varias opciones sugeridas como after() y Button y las adapté a mi estructura modular. Algunas ideas fueron directamente integradas.
Reflexión crítica	Muy útil para descubrir herramientas que no conocía. La documentación de cada widget fue revisada posteriormente para entender detalles.
Otros	La IA me dio un punto de partida práctico, pero siempre probé visualmente para validar su comportamiento real.

3. Requerimientos del Sistema

Requerimientos funcionales:

Menú principal con acceso a:

- Jugar
- Configurar
- Guardar partida
- Cargar partida
- Récords
- Ayuda
- Acerca de
- Salir

Pantalla de juego con los siguientes componentes:

- Tablero de 9x9 casillas
- Panel numérico del 1 al 9
- Botón "INICIAR JUEGO"
- Botones para acciones: deshacer, rehacer, borrar casilla, borrar juego, terminar juego
- Cronómetro o temporizador según configuración
- Visualización del nivel de dificultad
- Campo para nombre del jugador
- Validaciones automáticas al colocar números

Validaciones del juego:

- No se repiten números en grupos de suma
- La suma debe coincidir con la clave
- No se puede jugar sin seleccionar un número o si el juego no ha iniciado
- Temporizador termina el juego si llega a cero
- Cronómetro se detiene al terminar juego

Persistencia con archivos:

- kakuro2025_configuración.json: configuración del juego
- kakuro2025_partidas.json: todas las partidas predefinidas
- kakuro2025_guardado.json: estado del juego actual
- kakuro2025_récords.json: mejores tiempos por jugador y nivel

Récords:

- Guarda los mejores tiempos por nivel
- Permite consultar récords propios o globales
- Ordenados de menor a mayor tiempo

Componentes tkinter usados:

Frame, Button, Label, Entry, messagebox, Canvas, PhotoImage, ttk.Style, StringVar, entre otros

Manejo de pilas para jugadas:

- Jugadas realizadas (stack para deshacer)
- Jugadas deshechas (stack para rehacer)
- Reutiliza estructuras tipo LIFO (append y pop)

4. Archivos de datos usados

Archivo	Tipo	Uso	Estructura
kakuro2025_configuración.json	JSON	Guarda configuración del juego antes de iniciarse. Incluye nivel de dificultad y tipo de reloj.	json { "nivel": "FÁCIL", "reloj": "CON RELOJ", "horas": 0, "minutos": 5, "segundos": 0 }
kakuro2025_partidas.json	JSON	Contiene las partidas base disponibles para cada nivel. Se seleccionan aleatoriamente según la dificultad elegida.	json { "facil": [{ "tablero": [[0, 0], [0, 0], ...] }] }
kakuro2025_guardado.json	JSON	Almacena el estado actual de una partida activa (tablero, tiempo restante, jugador, nivel). Se usa para cargar partidas guardadas.	json { "nivel": "MEDIO", "jugador": "Ana", "estado_tablero": [[1, null, ...]], "reloj": { "horas": 0, "minutos": 4, "segundos": 30 } }
kakuro2025_récords.json	JSON	Almacena los récords de tiempos más rápidos por nivel. Cada entrada contiene nombre del jugador, tiempo y fecha.	json { "FÁCIL": [{ "jugador": "Ana", "tiempo": "00:04:30", "fecha": "2025-06-19" }] }

5. Conclusiones del Trabajo

Problemas encontrados y soluciones aplicadas

- **Manejo de diseño visual en tkinter**
  Problema: Las celdas y componentes visuales no se redimensionaban correctamente.
  Solución: Se forzó el uso de tamaños fijos, se aplicaron pack() y grid() de forma ordenada, y se ajustaron los contenedores (Frame) con cuidado.
- **Persistencia de información incompleta** *Problema:* Al guardar y cargar partidas, algunas claves podían faltar o fallar si el JSON no estaba bien formado.
  Solución: Se hizo un robusto sistema de validación con try/except y valores por defecto.
- **Verificación de jugadas complejas (sumas y duplicados)**
  Problema: Comprobar la validez de todas las claves fue complejo debido a estructuras anidadas.
  Solución: Se implementó lógica modular para verificar cada clave y validar valores únicos.

Aprendizajes obtenidos

- Aplicar la programación modular con componentes reutilizables fue esencial para evitar duplicación y facilitar la colaboración.
- Aprendí a combinar estructuras de datos (listas, diccionarios, pilas) para representar el estado del juego de forma clara.
- Dominar tkinter me permitió crear una interfaz gráfica interactiva y profesional.
- Entendí la importancia del versionado con Git y la colaboración usando ramas, commits y pull requests.
- Integrar persistencia de datos con archivos JSON fue fundamental para una experiencia completa de usuario.
- La validación de datos me enseñó a anticipar errores del usuario y del sistema, y cómo proteger la aplicación.

Estadística de tiempos

Actividad	**Horas invertidas**
Análisis del problema y diseño inicial del tablero	4h
Implementación de interfaz gráfica (pantalla principal y de juego)	6h
Creación del sistema de validación de claves y sumas	5h

Actividad	**Horas invertidas**
Manejo de archivos JSON (configuración, partidas, guardado, récords)	4h
Lógica de interacción (botones, panel numérico, colocación)	4h
Temporizador y cronómetro	2h
Sistema de récords	2h
Guardar / Cargar partidas	3h
Depuración y pruebas	3h
Documentación del proyecto	4h
Total	**37h**

4. Lista de revisión

Concepto Evaluado	% de Avance	Puntos Obtenidos	Análisis
Persistencia de configuración	100%	1.5 / 1.5	El archivo kakuro2025_configuración.json guarda y carga correctamente el nivel y modo de reloj. Se valida la estructura antes de usar.
Carga de partidas desde archivo	100%	1.5 / 1.5	Se filtran aleatoriamente por nivel. Se evita repetir partidas hasta agotarlas. Formato interno adaptado.
Construcción dinámica del tablero	100%	2 / 2	El tablero se arma 9x9 en tiempo real usando datos cargados. Se manejan correctamente celdas negras, claves y blancas.
Interacción con el tablero	100%	2 / 2	El jugador puede seleccionar números y colocarlos. Se valida el estado del juego y número antes de colocarlo.
Temporizador visual y funcional	100%	1 / 1	El cronómetro inicia si el juego está en modo "con reloj". Muestra cuenta regresiva y termina automáticamente.
Sistema de validación de jugadas	100%	2 / 2	Verifica sumas y duplicados por clave. Solo permite finalizar si el tablero es correcto. Usa lógica clara y robusta.
Guardar juego actual	100%	1 / 1	Guarda la configuración, tablero, tiempo y jugador en un archivo .json. Integrado al botón "GUARDAR JUEGO".
Cargar juego guardado	100%	1 / 1	Restaura el tablero, tiempo, jugador y estado del juego desde kakuro2025_guardado.json. Todo queda funcional.
Historial de jugadas (Deshacer / Rehacer)	100%	1 / 1	Se registra cada movimiento válido. Permite deshacer/rehacer usando pilas. Funciona con botones laterales.
Sistema de récords	100%	1 / 1	Guarda los mejores tiempos por nivel. Se muestra desde botón "RÉCORDS". Guarda nombre, tiempo y fecha.
Documentación del proyecto	100%	1.5 / 1.5	Documento completo: incluye estructura, análisis, archivos, tiempos, reflexión crítica, IA y revisión.
Interfaz gráfica estructurada con tkinter	100%	1 / 1	GUI atractiva, clara y modular. Panel numérico, tablero, acciones y reloj bien organizados. Se usaron múltiples widgets.

Referencias Bibliográficas

1. Atlassian. (s.f.). *What is Version Control?*. https://www.atlassian.com/git/tutorials/what-is-version-control
2. Git-SCM. (s.f.). *Documentation*. https://git-scm.com/docs
3. GitHub Docs. (s.f.). *About Git*. https://docs.github.com/es/get-started/using-git/about-git
4. Subversion. (s.f.). *Apache Subversion*. https://subversion.apache.org/
5. Mercurial SCM. (s.f.). *Mercurial - SCM*. https://www.mercurial-scm.org/
6. Python Software Foundation. (s.f.). *Tkinter — Python interface to Tcl/Tk*. https://docs.python.org/3/library/tkinter.html
7. TkDocs. (s.f.). *Tkinter Tutorial*. https://tkdocs.com/tutorial/