INSTITUTO TECNOLÓGICO DE COSTA RICA

CAMPUS TECNOLÓGICO CENTRAL CARTAGO

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

ESTUDIANTE:
HANZEL RAÚL ALPÍZAR DÍAZ

TALLER DE PROGRAMACIÓN
CÓDIGO: IC1400

PARQUEOS

DOCUMENTACIÓN

Mayo, 2025

Introducción

Este documento fue creado con el fin de documentar el proceso de desarrollo del programa Kakuro. En él se detallará la funcionalidad principal del sistema, los aspectos técnicos necesarios para su implementación, así como los temas que fue necesario investigar para cumplir con los requerimientos de sus distintas funcionalidades.

Entre los temas abordados se encuentran el manejo de archivos en formato JSON para la persistencia de datos, la creación de interfaces gráficas utilizando la biblioteca Tkinter, el uso de módulos en Python para organizar el código de manera estructurada, y la implementación de mecanismos de validación de jugadas para asegurar la integridad del juego. También se incluye la gestión de funcionalidades como "Deshacer" y "Rehacer", que mejoran la experiencia del usuario al permitirle revertir o restaurar acciones.

Además, se documentará el uso de Git como software de control de versiones, lo que facilitó la colaboración y el seguimiento de cambios en el código a lo largo del desarrollo. Se explorarán las características de Git que fueron utilizadas en este proyecto, así como la diferencia entre Git y plataformas como GitHub y GitLab, que también se consideraron para la gestión del repositorio.

Finalmente, se documentará el tiempo requerido y el porcentaje de avance de cada componente clave del proyecto, considerando tanto el desarrollo de la lógica del programa como su interfaz y funcionalidades adicionales, como la visualización de récords y la gestión de configuraciones del usuario.

Contenidos

Tabla de contenido

Software de control de versiones y colaboración

Un software de versiones es un programa que permite gestionar y registrar los cambios realizados en el código fuente de un proyecto. Este es especialmente importante cuando se trabaja en equipo, ya que facilita la colaboración, el seguimiento del historial de cambios, la resolución de conflictos y la posibilidad de volver a versiones anteriores del código que ya teníamos. Por ejemplo, puedo trabajar con compañeros de la universidad en un proyecto (como el proyecto de Kakuro, si nos lo hubieran asignado en grupos) con estos software de control de versiones para que podamos acceder, modificar y enviar los programas en los que estemos trabajando de forma mas eficiente.

Lo bueno de estos tipos de software, es que es crucial para el desarrollo de software, ya que es mejora la organización del trabajo en equipo, porque se pueden compartir los archivos de forma mas efectiva. Tambien permite identificar y corregir errores mas fácilmente, nada mas basta con hacer los cambios y luego subirlos al software. Si hay un error, nada mas se vuelve al programa anterior que estaba funcionando y se puede seguir trabajando. Adicionalmente, faciilita la colaboración remota porque ahora se acostumbra a trabajar de forma no-presencial, ya sea porque no se puede o porque es parte del estilo de cada persona. Finalmente, respalda la evolución del proyecto de forma estructurada, porque estos software almacena todos los avances de los proyectos como si fuera un hisotiral, además se puede comentar y dar contexto de qué avances se han hecho con comentarios cada vez que se sube al respositorio.

Ejemplos de software de control de versiones y colaboración

Git

Es un sistema de control de versiones distribuido. Fue creado por Linus Torvalds y es uno de los más utilizados en el mundo. Lo bueno que tiene este software es que permite llevar un historial detallado de todos los cambios, trabajar en diferentes ramas del proyecto y colaborar sin necesidad de estar en línea todo el tiempo.

GitHub

Esta es una plataforma que está basada en la web que usa Git como sistema de control de versiones. Lo bueno de este es que facilita la colaboración entre desarrolladores al permitir almacenar repositorios en línea, realizar revisiones de código, crear solicitudes de extracción (pull requests) y gestionar problemas (issues).

GitLab

Es parecido a GitHub, porque GitLab también es una plataforma basada en git, pero tiene algunas diferencias, porque tiene la opción de ser instalada en servidores propios. Este ofrece herramientas integradas para CI/CD (Integración continua/Despliegue continuo), gestión de proyectos y control de versiones.

Diferencia entre Git y GitHub

Si bien ambas herramientas se parecen bastante, además por su nombre que se pueden confundir e incluso usar como sinónimos, tienen algunas características que los diferencian. Primero, es necesario aclarar que Git es un software de control de versiones, mientras que GitHub también cumple el mismo propósito, pero es más una plataforma de alojamiento colaborativo. Con respecto a la conectividad, Git funciona de manera local, mientas que GitHub es una plataforma web que no ocupa instalación si se usa desde un navegador como Chrome. Para que Git funciona, no necesita de una conexión a internet, ya que funciona localmente, por otra parte, GitHub requiere conexión a internet para sincronizar repositorios. Git se usa principalmente para dar un seguimiento de los cambios, creación de ramas, y el historial de versiones, mientras que GitHub se usa más para el almacenamiento remoto, colaboración, gestión de proyectos y revisión de código. Git usa una interfaz de línea de comandos; por otra parte, GitHub usa interfaz gráfica ya que se puede usar en la web. Con Git se puede colaborar de manera distribuida mediante repositorios locales, mientras que GitHub facilita la colaboración masiva, integra equipos y permite dar un seguimiento de tareas mediante issues, pull requests, y wiki. Finalmente, El control en Git está en manos del desarrollador que tiene el repositorio, mientras que GitHub tiene controles de acceso, permisos, autenticación y visibilidad de proyectos, que la gente puede acceder de manera pública.

Características de Git usadas en este proyecto

git init: Inicializa un repositorio local. Esto permitió empezar a llevar un control desde la primera línea de código del proyecto Kakuro.

git add: Se usa para añadir archivos específicos (o todos) al área de preparación (*staging area*). Esto permitió seleccionar qué cambios incluir en el siguiente commit, por ejemplo, después de agregar nuevas funciones al tablero o modificar la lógica del juego.

git commit -m: Guarda los cambios con un mensaje que describe qué se hizo.

git status: Muestra los archivos modificados, nuevos, o eliminados desde el último commit. Esto ayudó a detectar si había cambios sin registrar o archivos pendientes de seguimiento.

git branch **y** git switch: Se usaron para crear y alternar entre ramas, lo que permitió trabajar en funcionalidades nuevas (como animaciones o verificación de soluciones) sin afectar la rama principal (*main*). Ejemplo de rama: rama-validaciones.

git push **y** git pull: Se utilizaron para subir los cambios al repositorio remoto en GitHub y para actualizar la copia local con los cambios desde la nube. Esto permitió tener una copia de seguridad y colaborar si se trabaja con otros compañeros.

.

Nuevos componentes de Tkinter usados en este proyecto

tkinter.Frame (Contenedores): Me sirvió para organizar bien la ventana del juego. Usé varios frames para separar el tablero, los botones, el panel donde elijo los números y hasta el cronómetro. Con esto pude aplicar diferentes formas de organizar cada sección sin que todo queara desordenado o pegado.

tkinter.messagebox (Cuadros de diálogo): Lo usé para mostrar mensajes importantes al jugador. Me ayudó a dar avisos claros sin tener que llenar la pantalla de texto. Por ejemplo:

showerror() para cuando hay errores o el jugador hace algo inválido: "JUGADA NO ES VÁLIDA..." o "FALTA QUE SELECCIONE EL NÚMERO"

askyesno() para confirmar acciones importantes como: "¿ESTÁ SEGURO DE BORRAR EL JUEGO?" o "¿DESEA CONTINUAR EL MISMO JUEGO?"

grid() (Para ubicar elementos): Este fue clave para armar el tablero del Kakuro. Como todo es tipo cuadrícula, grid() me permitió colocar cada casilla en su posición exacta (fila y columna). También lo usé para organizar los botones y el panel de números.

widget.after() (Programación de eventos): Esta función la usé para el cronómetro del juego. Me permitió hacer que el tiempo avance y se actualice sin que la ventana se congele. Básicamente, con esto pude crear un temporizador que corre mientras el jugador juega.

Uso de prompts con inteligencia artificial

Objetivo de uso	Dame ejemplos para poner archivos json por defecto
Herramienta utilizada	Chatgpt
Prompt o pregunta	Dame archivos json para usarlos en mi programa, con información
Respuesta	Me tiró algunos ejemplos de archivos json para iniciar, y poderlos usar para no arrrancar mi proyecto con archivos vacíos.
¿Cómo usó o adatpó su respuesta?	Usé los datos para usarlos como por dejecto, a modo de ejemplo
Reflexión crítica	Me parece que esto es muy bueno, porque así me ahorro el tiempo de estar creando archivos manualmente, ya que puede tardar mucho tiempo
Otros	

Objetivo de uso	Uso de tkinter
Herramienta utilizada	Chatgpt
Prompt o pregunta	Funcionalidad de casillas
Respuesta	Me tiró algunas funciones que tiene tkinter para poder implementarlos en mi programa de kakuro, como el grid(), o el temporizador
¿Cómo usó o adatpó su respuesta?	Me dio ejemplos de cómo usarlo dentro de un programa. los use para acomodar las casillas.
Reflexión crítica	Lo veo de gran utilidad, si se implementa bien. En este caso, tuve que aprender a cómo usarlos pero ya adaptado a mi código, porque el ejemplo que me dio no se podía copiar y pegar y esperar a que funcionara en mi programa
Otros	

Objetivo de uso	Acomodar casillas
Herramienta utilizada	Chatgpt
Prompt o pregunta	Ayuda para implementar las casillas en mi programa de kakuro
Respuesta	El json para acomodar las casillas del kakuro
¿Cómo usó o adatpó su respuesta?	Se usó de la siguiente forma: para no escribir todo el código json de los distintos niveles del kakuro (porque había que poner las filas, columnas, contenido de cada casilla, etc.) le pedí a a la IA, a través de una forma en que yo le puse las casillas con cada contenido de las casillas, que me las acomodara en los datos de los json. Yo le daba la info de cada casilla por cada fila, y la IA me la acomodaba en formato json
Reflexión crítica	Considero que no hay problema en usar la IA de esta forma, ya que para mí la IA es una herramienta muy eficiente que le ahorra al ser humano hacer trabajo hormiga. Siempre y cuando se entienda la lógica detrás de lo que se le está pidiendo y lo único que uno quiere hacer es ahorrase el trabajo de escribir tanto código, uno se lo puede ahorrar usando IA.
Otros	

Archivos de datos

Nombre: kakuro2025_partidas.json

Tipo: Archivo JSON.

Uso: Contiene las configuraciones de los tableros de Kakuro predefinidos para los diferentes niveles de dificultad. El programa lee este archivo al iniciar un nuevo juego para seleccionar aleatoriamente una partida según el nivel configurado.

Estructura: Se organiza como una lista de objetos JSON. Cada objeto representa una partida de Kakuro y contiene los campos: "nivel_de_dificultad", "partida", y "claves" (una lista de objetos que describen cada pista numérica con "tipo_de_clave", "fila", "columna", "clave", y "casillas").

Nombre: kakuro2025_récords

Tipo: Archivo JSON.

Uso: Almacena los récords de los jugadores, registrando el nombre del jugador y el tiempo que tardaron en completar una partida en un nivel específico. Este archivo permite la persistencia de los mejores tiempos entre sesiones del juego.

Estructura: Se presenta como una lista de objetos JSON. Cada objeto representa un récord individual y contiene: "jugador", "tiempo", y "nivel".

Nombre: kakuro2025_juego_actual

Tipo: Archivo JSON.

Uso: Permite guardar el estado actual de una partida en curso para que el jugador pueda continuarla en una sesión posterior. Almacena toda la información necesaria para restaurar el juego exactamente donde se dejó.

Estructura: Es un único objeto JSON que encapsula el estado completo del juego. Los campos incluyen: "nivel", "jugador", "tiempo_transcurrido", "tablero" (representación del estado del tablero), "jugadas_realizadas" (para la función "Deshacer"), y "jugadas_deshechas" (para la función "Rehacer").

Nombre: kakuro2025_configuración

Tipo: Archivo JSON.

Uso: Almacena las preferencias de configuración del juego establecidas por el usuario (nivel de dificultad por defecto, tipo de reloj y tiempo predefinido para el temporizador). Esto asegura que las configuraciones persistan entre las ejecuciones del programa.

Estructura: Es un único objeto JSON con los campos: "nivel_por_defecto", "tipo_reloj", y "tiempo_temporizador".

Conclusiones del trabajo

El desarrollo del proyecto Kakuro presentó varios desafíos y oportunidades de aprendizaje. Uno de los principales problemas encontrados fue la gestión de la persistencia de datos. Al principio, había dificultades para asegurarse de que los récords de los jugadores y el estado de las partidas se guardaran y cargaran correctamente. Para solucionar esto, se implementó un sistema de archivos JSON que permitió guardar y leer datos de manera efectiva, utilizando json.dump() y json.load(). Además, se incorporó un manejo de excepciones que ayudó a gestionar situaciones como archivos faltantes o corruptos, mejorando así la experiencia del usuario.

Otro desafío importante fue mantener la interfaz gráfica responsiva. A veces, la GUI se congelaba o las actualizaciones eran lentas, lo que afectaba la jugabilidad. Para resolver esto, se utilizó el método widget.after() de Tkinter, que permitió programar actualizaciones del cronómetro sin bloquear la interfaz. También se implementaron variables vinculadas que aseguraron que cualquier cambio en los datos se reflejara automáticamente en la pantalla, haciendo que la experiencia fuera más fluida.

La validación de jugadas en Kakuro también resultó ser un reto. Se necesitaba asegurarse de que las reglas del juego se cumplieran, como no repetir números en una suma. Se desarrolló una función de validación que revisa si un número ya existe en la fila o columna y si la suma actual excede la clave. Esto garantizó que el juego funcionara correctamente y proporcionó retroalimentación inmediata al jugador.

Finalmente, la gestión de las funciones de "Deshacer" y "Rehacer" fue clave. Se utilizaron pilas para almacenar las jugadas realizadas y deshechas, lo que permitió a los jugadores revertir o restaurar acciones de manera sencilla. Esta implementación mejoró la experiencia general del juego y demostró la utilidad de las pilas en la gestión del historial de acciones.

En resumen, el proyecto Kakuro permitió aplicar conocimientos teóricos y brindó una valiosa experiencia práctica en el desarrollo de software. Desde la gestión de datos hasta el diseño de interfaces gráficas y la resolución de problemas, esta experiencia fue fundamental para el crecimiento en el ámbito del desarrollo de software y del resto de la carrera, sobretodo en cursos propidos de la misma, porque este proyecto, y todo el curso en general, ha fomentado bastante las habilidades de investigación.

Acitvidades realizadas

Actividad Realizada	Horas
Análisis del problema	30
Diseño de algoritmos	50
Investigación de los archivos json	1
Programación	16
Documentación interna	2
Pruebas	5
Elaboración del manual de usuario	6
Elaboración de documentación del proyecto	5
Investigación de como diseñar los niveles de Kakuro	3
Investigación de tkinter (nuevas funciones)	4
TOTAL	122

LISTA DE REVISIÓN DEL PROYECTO

Concepto	Puntos originales	Avance 100%-0	Puntos obtenidos	Análisis de resultados
Opción Jugar: despliegue ventana de juego	15	100	15	
Seleccionar partida	5	100	5	
Botón Iniciar Juego	15	100	15	
Creación del archivos de récords	5	80	4	Es posible que se presenten problemas
Borrar casillas	2	100	2	
Botón deshacer Jugada	8	100	8	
Botón rehacer jugada	8	50	4	Algunas veces no funcionea como es debido. No se pudo realizar por falta de tiempo
Botón borrar juego	2	100	2	
Botón Récords	2	20	0	Solo funcióna al principio, pero solo los records que tiene por defecto. No se pudo completar satisfactoria por falta de tiempo e intelecto.
Botón Guardar Juego	8	80	3	Funciona solo en algunos niveles
Botón Cargar Juego	5	70	3	Solo sirve bajo ciertas condiciones
Opción configurar	10	100	10	
Ayuda en el programa: manual de usuario	5	100	5	
Cronómetro o Temporizador en tiempo real	5	100	5	
TOTAL	100	85.71	91	
Partes desarrolladas adicionalmente				
Observaciones				Es posible que algunas cosas no funcionen de manera correcta.