1. **Definir el objetivo del juego**: Decide si quieres que los jugadores resuelvan un Sudoku diario o si también quieres ofrecer diferentes niveles de dificultad (fácil, medio, difícil).
2. **Generar Sudokus diarios**: Hay algoritmos para generar Sudokus automáticamente. Puedes utilizar una librería de programación como PySudoku para generarlos si tienes conocimientos de programación en Python.
3. **Diseñar la interfaz**: Decide cómo se verá el Sudoku. Puedes diseñar una cuadrícula simple donde los usuarios puedan ingresar números.
4. **Implementar la lógica del juego**: Necesitarás programar la lógica para verificar que las soluciones sean correctas. Cada fila, columna y subcuadro de 3x3 debe contener los números del 1 al 9 sin repetirse.
5. **Crear la página web**: Utiliza HTML, CSS y JavaScript para diseñar e implementar la página web. Asegúrate de que sea fácil de usar y visualmente atractiva.
6. **Añadir características adicionales**: Podrías incluir una tabla de clasificación, pistas para ayudar a los jugadores y un temporizador para los modos contrarreloj.
7. **Implementar almacenamiento**: Guarda el progreso diario de los jugadores y permite que comparen sus resultados con otros.
8. **Probar y lanzar**: Asegúrate de probar el juego a fondo antes de lanzarlo para corregir errores y mejorar la experiencia del usuario.

 **Algoritmo de Generación Aleatoria**: Puedes crear un algoritmo que no solo genere los Sudokus, sino que también asigne aleatoriamente una dificultad (fácil, medio, difícil) a cada juego diario. Existen parámetros que puedes ajustar, como el número de pistas iniciales y la complejidad de las reglas de generación.

 **Selección de Dificultad al Azar**: Al comienzo de cada día, el sistema puede seleccionar aleatoriamente un nivel de dificultad y luego generar un Sudoku acorde a ese nivel.

 **Dificultad Progresiva**: Podrías tener un sistema en el que la dificultad aumenta con cada intento fallido del jugador, proporcionando un reto creciente, o que varíe aleatoriamente dentro de un rango.

 **Interfaz de Usuario**: En la página web, podrías mostrar un indicador de la dificultad antes de que los jugadores comiencen, añadiendo una pequeña nota que diga “Hoy toca un Sudoku [dificultad]”.

Algoritmo para generar sudokus   
import random

def is\_valid(board, row, col, num):

# Check if the number is valid in the given position

for i in range(9):

if board[row][i] == num or board[i][col] == num:

return False

start\_row, start\_col = 3 \* (row // 3), 3 \* (col // 3)

for i in range(start\_row, start\_row + 3):

for j in range(start\_col, start\_col + 3):

if board[i][j] == num:

return False

return True

def solve(board):

for row in range(9):

for col in range(9):

if board[row][col] == 0:

for num in range(1, 10):

if is\_valid(board, row, col, num):

board[row][col] = num

if solve(board):

return True

board[row][col] = 0

return False

return True

def generate\_sudoku():

board = [[0 for \_ in range(9)]]

for \_ in range(random.randint(15, 30)): # Control the number of filled cells initially

row, col = random.randint(0, 8), random.randint(0, 8)

num = random.randint(1, 9)

if board[row][col] == 0 and is\_valid(board, row, col, num):

board[row][col] = num

solve(board)

return board

# Generate a Sudoku

sudoku = generate\_sudoku()

for row in sudoku:

print(row)

Versión con verificación de solución

import random

def is\_valid(board, row, col, num):

for i in range(9):

if board[row][i] == num or board[i][col] == num:

return False

start\_row, start\_col = 3 \* (row // 3), 3 \* (col // 3)

for i in range(start\_row, start\_row + 3):

for j in range(start\_col, start\_col + 3):

if board[i][j] == num:

return False

return True

def solve(board):

empty\_cell = find\_empty(board)

if not empty\_cell:

return True

row, col = empty\_cell

for num in range(1, 10):

if is\_valid(board, row, col, num):

board[row][col] = num

if solve(board):

return True

board[row][col] = 0

return False

def find\_empty(board):

for row in range(9):

for col in range(9):

if board[row][col] == 0:

return (row, col)

return None

def generate\_sudoku(difficulty='medium'):

board = [[0 for \_ in range(9)] for \_ in range(9)]

solve(board)

remove\_numbers(board, difficulty)

return board

def remove\_numbers(board, difficulty):

attempts = {'easy': 40, 'medium': 50, 'hard': 60}

cells\_to\_remove = attempts.get(difficulty, 50)

while cells\_to\_remove > 0:

row, col = random.randint(0, 8), random.randint(0, 8)

while board[row][col] == 0:

row, col = random.randint(0, 8), random.randint(0, 8)

backup = board[row][col]

board[row][col] = 0

copy\_board = [row[:] for row in board]

if solve(copy\_board):

cells\_to\_remove -= 1

else:

board[row][col] = backup

# Generar un Sudoku con dificultad aleatoria

difficulty = random.choice(['easy', 'medium', 'hard'])

sudoku = generate\_sudoku(difficulty)

for row in sudoke:

print(row)