



## PySudoku: Implementación en Python usando PyQt

Iván Aveiga  
Kevin Campuzano

Escuela Superior Politécnica del Litoral

4 de agosto de 2013

## Interfaz, 5 Puntos

- Pantalla de Inicio
- Juego Nuevo
- Cargar Partida
- Estadísticas

# Interfaz

## Pantalla de Inicio

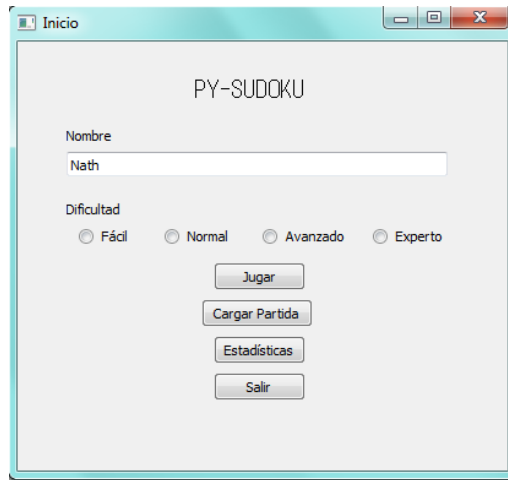
- Pantalla de Inicio donde se puede iniciar un nuevo juego, cargar una partida o ver las estadísticas



# Interfaz

## Pantalla de Inicio

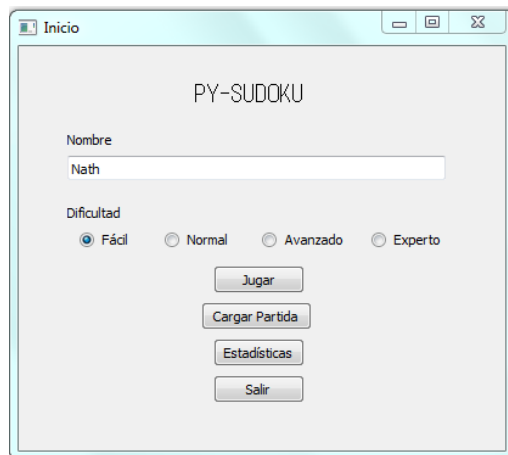
- Si no se indica el nivel no se puede iniciar un nuevo juego.



# Interfaz

## Pantalla de Inicio

- Pantalla de Inicio con los datos correctos para iniciar un nuevo juego.



# Interfaz

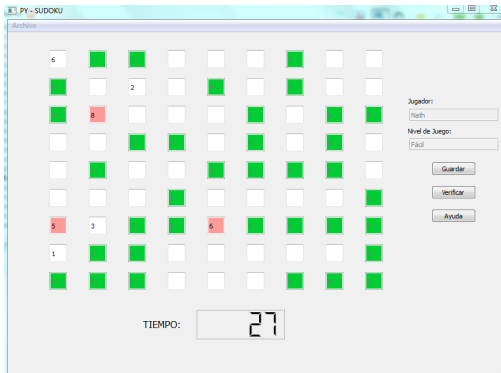
## Funcionamiento del Widget Casilla

- Es un QLineEdit.
- Tiene una longitud máxima de un caracter.
- Tiene una **Input Mask** de valores enteros del 0 al 9.
- Se valida que no se ingrese el número 0.
- Generados dinámicamente agregados a un **QtGridLayout**

# Interfaz

## Jugadas Erradas

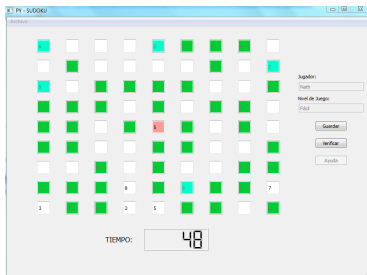
- Las celdas erróneas se muestran en un color distinto al blanco (valor correcto o vacío del tablero) o al verde (valor fijo del tablero).



# Interfaz

## Ayuda

- Se resuelve una celda aleatoria y se marca de color turquesa, además se inhabilita para editar la celda ya que no tiene sentido eliminar el valor resuelto por el juego. El jugador cuenta con 5 ayudas, en el momento que se usa por lo menos una ayuda deja de ser considerado para formar parte del ranking. Una vez usada las 5 ayudas se deshabilita el botón de ayuda.





# Funcionalidad, 12 Puntos

## Algoritmo

- Se utiliza un Generador de tableros.
- Se genera un tablero eliminando celdas aleatorias en el tablero generado.
- La dificultad está basada en el número de casillas a jugar.

Nivel	Celdas
Fácil	36 - 41 Fichas
Normal	32 - 35 Fichas
Avanzado	28 - 31 Fichas
Experto	23 - 27 Fichas

# Funcionalidad

## Algoritmo

```
proc poblar_matriz( )
    matriz = [9][9]
    col = 0
    mientras col < 9:
        posibles = [1,2,3,4,5,6,7,8,9]
        mezclar(posibles)
        fila = 0
        prueba = 0
        mientras fila < 9 y prueba < 200
            malas = 0
            parar = 0
            para cada elemento i en posibles
                si posibles[i] está en columna
                    malas += 1

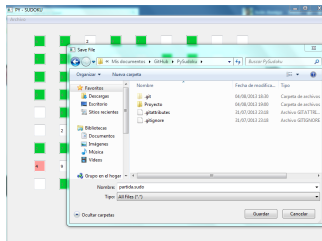
            si malas == 9
                limpiar matriz
                parar = 1
                fila = 0
                prueba += 1
            si parar == 0
                numero = aleatorio(1,9)
                mientras conflicto(numero)
                    numero = aleatorio(1,9)
                matriz[fila][col] = numero
                remover numero de posibles
                filas += 1

            col += 1
            si prueba == 20
                limpiar matriz
                col = 0

    retornar matriz
```

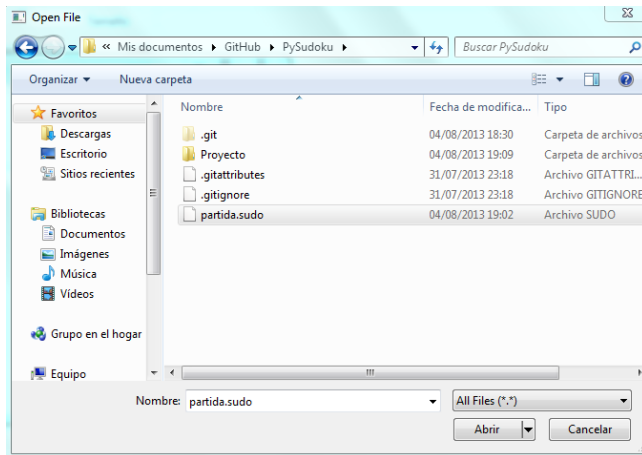
# Funcionalidad

## Guardar Partida



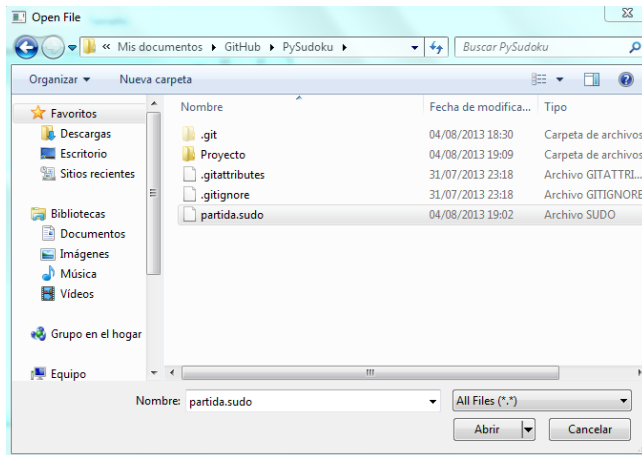
# Funcionalidad

## Cargar Partida



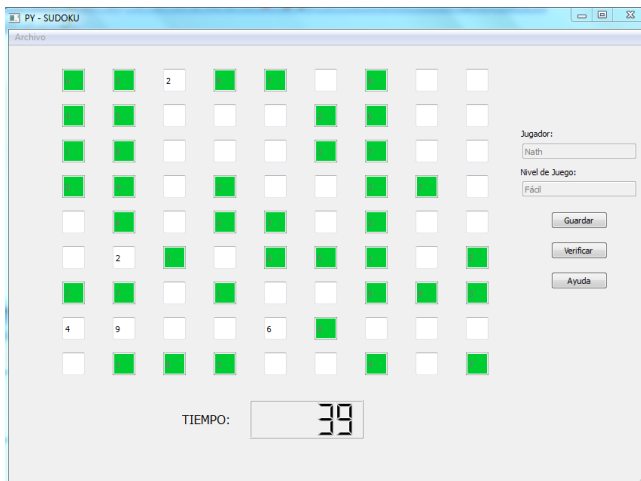
# Funcionalidad

## Cargar Partida



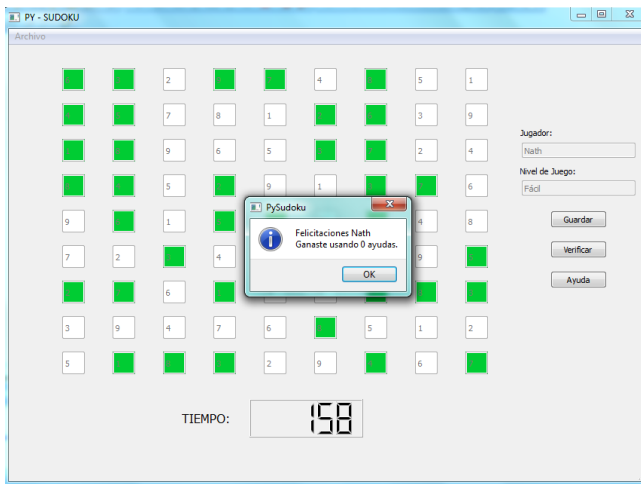
# Funcionalidad

## Cargar Partida



# Funcionalidad

## Finalización de Partida



# Manual - L<sup>A</sup>T<sub>E</sub>X, 5 puntos

## Lista de Funcionalidades

- Jugar nueva partida.
- Cargar nueva partida.
- Salvar partida.
- Ranking.



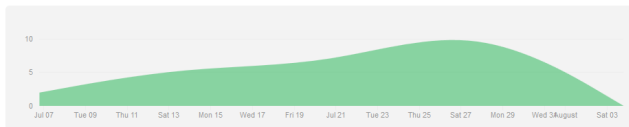
# Manual - L<sup>A</sup>T<sub>E</sub>X, 5 puntos

## Responsabilidades

- Iván Aveiga:
  - Algoritmo de generación de tableros.
  - Guardar y Cargar partida (incluida serialización).
  - Jugar (nueva partida, verificación de partida, partida ganada).
- Kevin Campuzano:
  - Interfaz Gráfica.
  - Estadísticas.
  - Conectar UI con parte lógica.

# Colaboración - Git, 10 puntos

Picos de Commit



# Colaboración - Git, 10 puntos

## Commits por Integrante

