ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

INGENIERÍA EN SOFTWARE

Construcción y Evolución de Software

Manual Para el Programador

Integrantes:
Karina Arichavala
Gilson Cango
Christian Nazate
Victor Rodriguez

Curso: GR2SW

Fecha de entrega: 06 de junio de 2024

Período académico: 2024-A

Contenido

Introducción 3
1. Objetivo 3
2. Requisitos del proyecto 3
Software: 3
Hardware: 3
3. Preparación para el desarrollo 3
4. Estructura del proyecto 3
__pychache__ 4
Font 4
BrailleApp.py 4
brailleDictorionary.txt 4
BrailleImageGenerator.py 4
BrailleTranslator.py 4
LICENCE 5
main.py 5
Readme.md 5
5. Estándares 5
Documentación del código 5
Commits 5
6. Contacto 6

Introducción

El presente documento tiene como objetivo describir aspectos esenciales del sistema al lector, en este caso programador, que desea entender el sistema. Su principal función es familiarizar al personal encargado de actividades de mantenimiento, revisión, solución de problemas, futuros desarrolladores, instalación y configuración del sistema.

1. Objetivo

- Instruir al personal que manipulará el sistema proporcionando una guía detallada para entender, modificar o expandir el código del aplicativo según sea necesario.

2. Requisitos del proyecto

Software:

Las versiones de software detalladas a continuación son las mas recientes usadas para crear el software, con el fin de mantener la seguridad del sistema es necesario actualizarlo a la versión estable más reciente posible.

- Python 3.13.3 – April 9, 2024; Versión estable.
- Librería de Python pillow – 10.3.0
- Git – 2.45.2 Mayo 31, 2024

Hardware:

- Computador con al menos 4GB de RAM
- Sistema operativo Windows, macOS o Linux

3. Preparación para el desarrollo

El repositorio es público y se encuentra alojado en github. Para clonarlo en una terminal del computador ejecutar el comando presentado a continuación:

git *clone https://github.com/VicttorAE/magickeydevelopment-traductor-braille.git*

4. Estructura del proyecto

El proyecto presenta la estructura de directorios observada en la Figura 1.

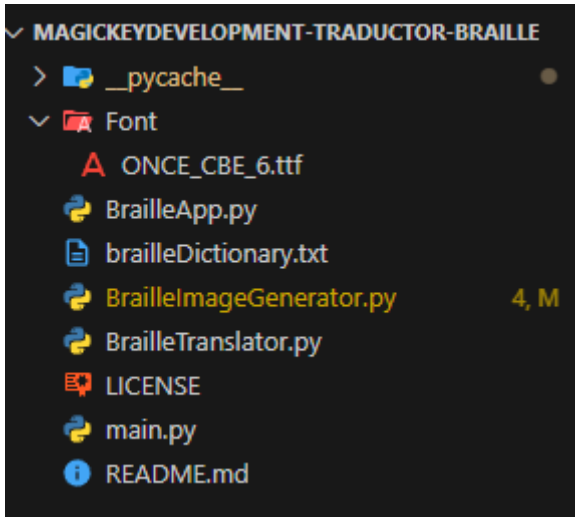

Figura 1. Estructura del proyecto.

A continuación, se presenta un detalle de cada elemento.

__pychache__
Directorio propio de Python que almacena archivos compilados en bytecode. No es necesario modificarlo y tampoco es necesario incluirlo en el repositorio.

Font
Fuentes usadas en el proyecto. El único archivo de este directorio es usado para generar la señalética.

BrailleApp.py
Archivo de Python que contiene la clase que crea la interfaz grafica de usuario para el programa de traducción texto – braille.

Métodos:

- **convertir_esp_a_brl():**
  Convierte el texto en español del campo de entrada a braille y lo muestra en el campo de texto en braille.
- **convertir_brl_a_esp():**

  Convierte el texto en braille del campo de entrada a español y lo muestra en el campo de texto en español.

- **generar_senaletica():**

  Genera una imagen de señalética en braille a partir del texto en español ingresado y la guarda como 'senaletica.png'.

- **imprimir_espejo():**

  Convierte el texto en español a texto en braille en modo espejo y lo muestra en el campo de texto en braille.

brailleDictorionary.txt
Archivo de texto que contiene los caracteres alfabéticos con su respectivo equivalente a braille, separados por “:”.

BrailleImageGenerator.py
Archivo de Python que contiene métodos útiles para generar la señalética en formato de imagen.

Métodos:

- **generar_senaletica_braille():**
  Genera una imagen en formato .png que contiene la señalética braile, es decir, a partir de un texto ingresado genera una imagen de la respectiva traducción.
- **imprimir_en_espejo_braille():**
  Genera una representación en braille de un texto ingresado e invierte el resultado para retornalo.

BrailleTranslator.py
Archivo de Python que contiene métodos útiles para realizar la traducción de texto a alfabeto braille y viceversa.

Métodos:

- **load_dict():**

  A partir del archivo de texto brailleDictionary.txt crea un diccionario de Python que contiene cada carácter alfabético y su respectivo equivalente en braille.

- **texto_a_braille():**

  Genera una representación en braille de un texto dado buscando cada carácter del texto ingresado en el diccionario y seleccionando su respectiva traducción braille.

- **braille_a_texto():**

  Genera texto a partir de una cadena de alfabeto braille invirtiendo el diccionario de caracteres y buscando cada uno de ellos seleccionando su traducción a texto.

LICENCE
Documento que especifica los términos y condiciones bajo los cuales se puede usar, copiar, modificar y distribuir el software. No debe ser modificado.

main.py
Clase ejecutable que permite ejecutar el programa en el entorno de desarrollo.

Readme.md
Información del proceso de desarrollo del proyecto.

5. Estándares

Documentación del código
La documentación del código sigue el estándar de pydocs para cada clase o método. A continuación, se observa un ejemplo.

"""
General detail of the method.

Specific detail of the method if needed.

Parameters:
Variable name (variable type): *Description of the entry variable.*

Returns:
(type like str, int, etc. or None if not return): *Description of the returned variable.*
"""

Commits
Para mantener un historial de commits claro y estructurado, utiliza el estándar de Conventional Commits. Este sistema de nomenclatura mejora la legibilidad del historial de commits y facilita la generación automática de versiones y changelogs. A continuación, se muestra un ejemplo.

<tipo>(<alcance>): <descripción>

[cuerpo opcional]

[pie opcional]

Tipos comunes:

- **feat:** Una nueva característica.
- **fix:** Una corrección de errores.
- **docs:** Cambios en la documentación.
- **style:** Cambios que no afectan el significado del código (espacios, formato, etc.).
- **refactor:** Un cambio de código que no corrige un error ni agrega una característica.
- **perf:** Un cambio en el código que mejora el rendimiento.
- **test:** Añadir o corregir tests.
- **build:** Cambios que afectan el sistema de construcción o dependencias externas (ej. gulp, npm).
- **ci:** Cambios en los archivos y scripts de configuración de CI (ej. Travis, Circle, etc.).
- **chore:** Otras tareas que no modifican ni el código ni los tests.
- **revert:** Revertir un commit anterior.

6. Contacto

En caso de ser necesario información mas especifica o se requiere conocer algún detalle no incluido en el presente manual contactar con el dueño del repositorio generando un nuevo Issue en la página del repositorio de Github: https://github.com/VicttorAE/magickeydevelopment-traductor-braille