

Universidad Autónoma de Yucatán

Licenciatura en Ingeniería de Software

(LIS)

Construcción de Software

Primera entrega de proyecto

Integrantes:

Adiel Elioenai Herrera Herrera

Héctor Joaquín Pérez Can

Alan Alfonso Pérez Romero

23/05/2024

| No. | Consideraciones para el Nombramiento de Clases | Cumplido (Sí/No) |
| --- | --- | --- |
| 1 | Sigue convenciones de nomenclatura consistentes | SI |
| 2 | Utiliza nombres descriptivos y sustantivos | SI |
| 3 | Utiliza PascalCase para nombres de clases | SI |
| 4 | Evita nombres demasiado largos | SI |
| 5 | Evita nombres confusos o similares | SI |
| 6 | Utiliza nombres que reflejen el propósito de la clase | SI |
| 7 | Utiliza nombres singulares y no plurales, y no verbos | NO |
| 8 | Evita acrónimos y abreviaturas confusas | SI |
| 9 | Sé consistente en la traducción de nombres si es necesario | SI |
| 10 | Evita prefijos o sufijos innecesarios | SI |
| 11 | Utiliza nombres que sean fácilmente comprensibles y memorables | SI |

| No. | Consideraciones para la creación de interfaces de clase | Cumplido (Sí/No) |
| --- | --- | --- |
| 12 | La clase implementa un sólo TDA | SI |
| 13 | Prestar servicios por parejas con sus opuestos | SI |
| 14 | Sólo hay información de la clase | SI |
| 15 | Las interfaces deben ser programáticas en lugar de semánticas | SI |
| 16 | No agregar métodos públicos que sean inconsistentes con la abstracción de la interfaz | SI |
| 17 | La clase es fácil de entender y utilizar desde el exterior (abstracción) | SI |
| 18 | Mantiene una estructura interna organizada y relacionada (cohesión). | SÍ |

| No. | Consideración para el Nombramiento de Variables | Cumplido (Sí/No) |
| --- | --- | --- |
| 1 | Sigue convenciones de nomenclatura consistentes | -- |
| 2 | Utiliza nombres descriptivos | -- |
| 3 | Evita nombres demasiado largos | Si |
| 4 | Elige un estilo de nomenclatura coherente | Si |
| 5 | Evita nombres confusos o similares | Si |
| 6 | Utiliza nombres plurales para colecciones | No |
| 7 | Prefija variables globales o de clase si es necesario | No |
| 8 | Evita acrónimos y abreviaturas confusas | Si |
| 9 | Revisa y actualiza nombres regularmente | No |
| 10 | Sé consistente en la traducción de nombres | Si |
| 11 | Utiliza nombres que eviten comentarios innecesarios | Si |

| No. | Consideración para el Nombramiento de funciones | Cumplido (Sí/No) |
| --- | --- | --- |
| 1 | Describe todo lo que la función realiza | Sí |
| 2 | No se basa en números para diferenciar las rutinas | Sí |
| 3 | Evita nombres demasiado largos | Sí |
| 4 | El nombre de la función usa en él el valor a devolver | Sí |
| 5 | Para nombrar un procedimiento, menciona un verbo descriptivo junto a un objeto. | Sí |
| 6 | Usa los opuestos de manera correcta | Sí |

| No. | Home.py | Cumplido (Sí/No) |
| --- | --- | --- |
| 1 | La clase implementa un sólo TDA | Si |
| 2 | Prestar servicios por parejas con sus opuestos | No |
| 3 | Sólo hay información de la clase | Si |
| 4 | Las interfaces deben ser programáticas en lugar de semánticas | Si |
| 5 | No agregar métodos públicos que sean inconsistentes con la abstracción de la interfaz | Si |
| 6 | La clase es fácil de entender y utilizar desde el exterior (abstracción) | Si |
| 7 | Mantiene una estructura interna organizada y relacionada (cohesión) | No |

| No. | Consideraciones para el Nombramiento de Variables: | Cumplido (Sí/No) |
| --- | --- | --- |
| 1 | Sigue convenciones de nomenclatura consistentes: | Si |
| 2 | Utiliza nombres descriptivos: | Si |
| 3 | Evita nombres demasiado largos: | Si |
| 4 | Elige un estilo de nomenclatura coherente: | Si |
| 5 | Evita nombres confusos o similares: | Si |
| 6 | Utiliza nombres plurales para colecciones: | N/A |
| 7 | Prefija variables globales o de clase si es necesario: | Si |
| 8 | Evita acrónimos y abreviaturas confusas: | Si |
| 9 | Revisa y actualiza nombres regularmente: | N/A |
| 10 | Sé consistente en la traducción de nombres: | Si |
| 11 | Utiliza nombres que eviten comentarios innecesarios: | Si |

| No. | Consideraciones para el Nombramiento de funciones: | Cumplido (Sí/No) |
| --- | --- | --- |
| 1 | Describe todo lo que la función realiza: | Si |
| 2 | No se basa en números para diferenciar las rutinas: | No |
| 3 | Evita nombres demasiado largos: | Si |
| 4 | El nombre de la función usa en él el valor a devolver: | No |
| 5 | Para nombrar un procedimiento, menciona un verbo descriptivo junto a un objeto: | Si |
| 6 | Usa los opuestos de manera correcta: | N/A |

| No. | AreaPDF.py  Consideraciones para la creación de interfaces de clase: | Cumplido (Sí/No) |
| --- | --- | --- |
| 1 | La clase implementa un sólo TDA | Si |
| 2 | Prestar servicios por parejas con sus opuestos: | No |
| 3 | Sólo hay información de la clase: | Si |
| 4 | Las interfaces deben ser programáticas en lugar de semánticas: | Si |
| 5 | No agregar métodos públicos que sean inconsistentes con la abstracción de la interfaz: | Si |
| 6 | La clase es fácil de entender y utilizar desde el exterior (abstracción): | Si |
| 7 | Mantiene una estructura interna organizada y relacionada (cohesión): | Si |

| No. | AreaDefiicion.py  Consideraciones para la creación de interfaces de clase: | Cumplido (Sí/No) |
| --- | --- | --- |
| 1 | La clase implementa un sólo TDA | Si |
| 2 | Prestar servicios por parejas con sus opuestos: | No |
| 3 | Sólo hay información de la clase: | Si |
| 4 | Las interfaces deben ser programáticas en lugar de semánticas: | Si |
| 5 | No agregar métodos públicos que sean inconsistentes con la abstracción de la interfaz: | Si |
| 6 | La clase es fácil de entender y utilizar desde el exterior (abstracción): | Si |
| 7 | Mantiene una estructura interna organizada y relacionada (cohesión): | Si |

| No. | SeleccionarPDF.py  Consideraciones para la creación de interfaces de clase: | Cumplido (Sí/No) |
| --- | --- | --- |
| 1 | La clase implementa un sólo TDA | Si |
| 2 | Prestar servicios por parejas con sus opuestos: | No |
| 3 | Sólo hay información de la clase: | Si |
| 4 | Las interfaces deben ser programáticas en lugar de semánticas: | Si |
| 5 | No agregar métodos públicos que sean inconsistentes con la abstracción de la interfaz: | Si |
| 6 | La clase es fácil de entender y utilizar desde el exterior (abstracción): | Si |
| 7 | Mantiene una estructura interna organizada y relacionada (cohesión): | Si |

| No. | AgregarPalabra.py  Consideraciones para la creación de interfaces de clase: | Cumplido (Sí/No) |
| --- | --- | --- |
| 1 | La clase implementa un sólo TDA | Si |
| 2 | Prestar servicios por parejas con sus opuestos: | No |
| 3 | Sólo hay información de la clase: | Si |
| 4 | Las interfaces deben ser programáticas en lugar de semánticas: | Si |
| 5 | No agregar métodos públicos que sean inconsistentes con la abstracción de la interfaz: | Si |
| 6 | La clase es fácil de entender y utilizar desde el exterior (abstracción): | Si |
| 7 | Mantiene una estructura interna organizada y relacionada (cohesión): | Si |

| No. | BuscadorDeDefinicion.py  Consideraciones para la creación de interfaces de clase: | Cumplido (Sí/No) |
| --- | --- | --- |
| 1 | La clase implementa un sólo TDA | Si |
| 2 | Prestar servicios por parejas con sus opuestos: | No |
| 3 | Sólo hay información de la clase: | Si |
| 4 | Las interfaces deben ser programáticas en lugar de semánticas: | Si |
| 5 | No agregar métodos públicos que sean inconsistentes con la abstracción de la interfaz: | Si |
| 6 | La clase es fácil de entender y utilizar desde el exterior (abstracción): | Si |
| 7 | Mantiene una estructura interna organizada y relacionada (cohesión): | Si |

**ERS**

1. Introducción

1.1 Propósito

El propósito surge de la necesidad de los departamentos de lingüística de utilizar el software para facilitar la extracción de términos, partiendo de los contextos definitorios; lo que es aquel fragmento de texto de un documento donde se define un término. Esta se vuelve una herramienta bastante útil en el área académica e investigativa, pues permite acceder a un corpus muy grande de terminología a partir de estos pequeños fragmentos.

1.2 Audiencia

La audiencia prevista para este proyecto abarca principalmente a los departamentos de lingüística, así como a estudiantes y docentes interesados en la materia. Además, está dirigido a cualquier persona que necesite acceder a definiciones de un libro para diversos propósitos, ya sea para estudios académicos, investigación, o simplemente para ampliar su conocimiento sobre el tema.

2. Descripción general

2.1 Perspectiva del producto

El proyecto final es un extractor de definiciones a partir de contextos definitorios, que se encarga de extraer textos de formato "‚pdf" en formato txt; para posteriormente ser depurado hasta extraer y buscar términos con sus definiciones. Posteriormente se crea un corpus el cual puede ser consultado donde se encuentre estas definiciones.

2.2 Tecnologías utilizadas

Fitz: interfaz para la librería PyMuPDF

El módulo Fitz proporciona funcionalidades para cargar archivos PDF, renderizar páginas, realizar operaciones de dibujo en las páginas y realizar manipulaciones básicas de PDF.

FreeLing, es una suite de software de procesamiento de lenguaje natural (PLN) de código abierto que proporciona herramientas lingüísticas para analizar texto en varios idiomas. Su objetivo principal es realizar tareas de procesamiento de lenguaje natural, como el análisis morfológico, análisis sintáctico, análisis semántico y otras tareas relacionadas con el procesamiento del lenguaje.

NLTK, es un módulo en la biblioteca Natural Language Toolkit (NLTK) de Python que proporciona acceso a varios corpus lingüísticos, es decir, colecciones de texto en diferentes idiomas y géneros que se utilizan con fines de investigación y desarrollo en procesamiento de lenguaje natural (PLN).

Los corpus contienen datos preprocesados y etiquetados que los desarrolladores y lingüistas pueden utilizar para entrenar y evaluar modelos de PLN, así como para realizar análisis lingüísticos y experimentos.

Librería "es \_core\_news\_sm" de spaCy es un modelo preentrenado específicamente para el procesamiento del lenguaje natural en español. spaCy es una biblioteca de procesamiento de lenguaje natural de código abierto que proporciona herramientas para realizar tareas como el análisis morfológico, análisis sintáctico, reconocimiento de entidades nombradas y más.

unittest, es un módulo incorporado en la biblioteca estándar de Python que proporciona un marco de pruebas unitarias. Las pruebas unitarias son un enfoque fundamental en el desarrollo de software para verificar que unidades individuales de código (como funciones, métodos o clases) funcionan correctamente de manera aislada.

Tkinter es una librería del lenguaje de programación Python y funciona para la creación y el desarrollo de aplicaciones de escritorio. Esta librería facilita el posicionamiento y desarrollo de una interfaz gráfica de escritorio con Python. Tkinter es el paquete estándar de Python para interactuar con Tk. Es el único toolkit o kit de herramientas para el desarrollo de una interfaz gráfica de usuario (GUI) que funciona en todos los sistemas operativos, es decir, funciona en Windows, Mac OS y Linux.

2.3 Estándares

En este proyecto utilizaremos el estándar de programación PEP 8 del lenguaje

Python, el cual proporciona convenciones de codificación para el código Python que comprende la biblioteca estándar en la distribución principal de Python. Es el estándar de programación más popular para Python y sus autores son parte de python.org, por lo que es prácticamente el oficial.

3. Características del sistema.

3.1 Requerimientos funcionales

RF-01. Convertir documentos .pdf en.txt: El código utiliza una librería que extrae únicamente el texto del documento.

RF-02. Extractor de contextos definitorios: El código busca las concurrencias en el texto para definir las oraciones que contienen CD's.

RF-03. Creación de corpus: Las definiciones extraídas se exportan a listas de CSV para poder ser consultadas.

RF-04. Interfaz de usuario: Implementa una interfaz de usuario que permite interactuar con los corpus.

RF-05. Buscador: Implementa un buscador que permite consultar un término y devolver su definición.

RF-06. Identificación de sustantivos: El código permite utilizar distintas librerías para poder identificar los sustantivos en las oraciones extraídas.

RF-07. Depurador: Depurar el contenido del texto para dejarlo lo más simplificado posible.

RF-08. Explorador de archivos: Permitirá a través de la interfaz cargar documentos pdf o txt para ser extraídos.

RF-09. Segmentación por áreas: Permitirá almacenar el corpus extraído a una carpeta del área al que corresponda.

RF-10. Permitir seleccionar archivo nuevo o existente: Permite al usuario realizar una búsqueda dentro de los corpus ya existentes o en convertir uno nuevo desde cero.

4. Requerimientos no funcionales.

4.1. Usabilidad: La interfaz de usuario debe ser intuitiva y permitir a los usuarios cargar archivos PDF fácilmente, así como visualizar y administrar el corpus de documentos extraídos de manera eficiente.

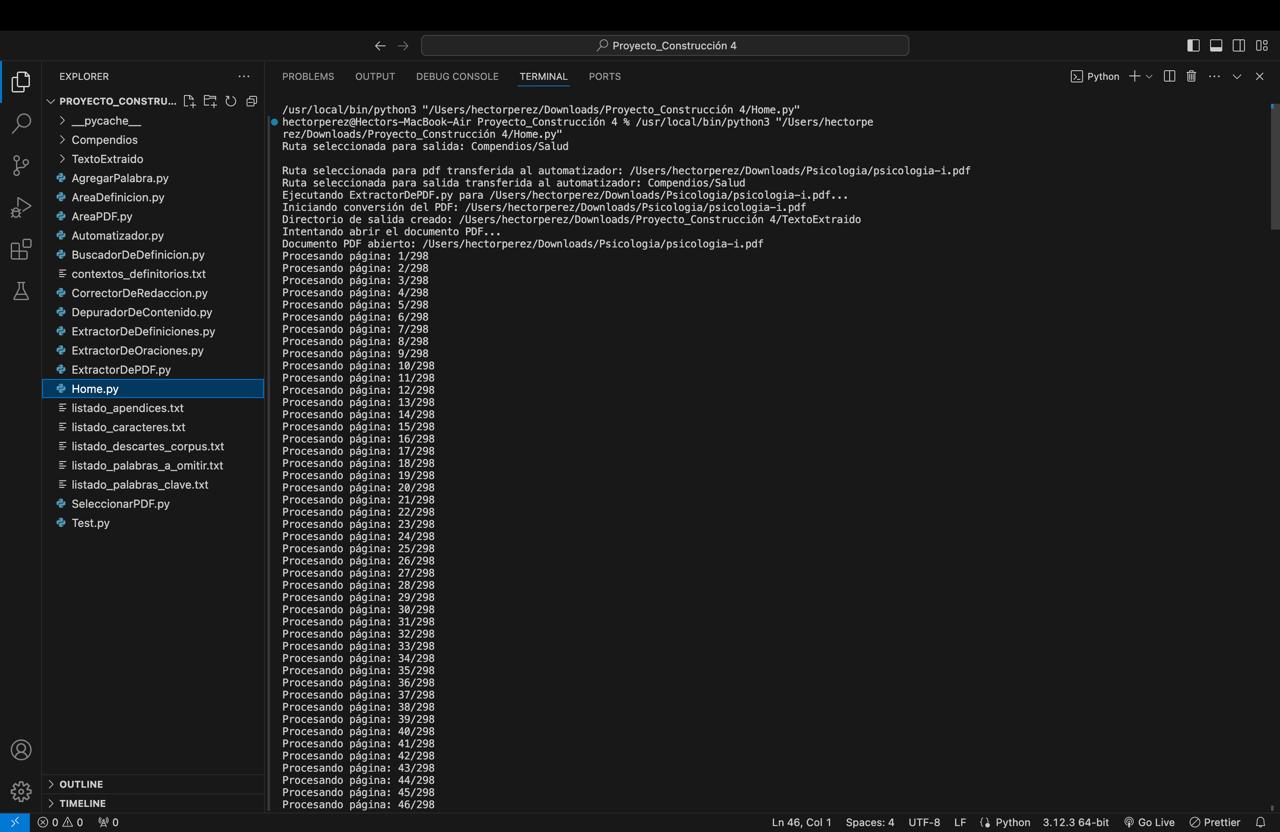
4.2. Rendimiento: El sistema debe ser capaz de procesar y extraer texto de múltiples archivos PDF de forma rápida y eficiente, con tiempos de procesamiento mínimos incluso para grandes volúmenes de documentos.

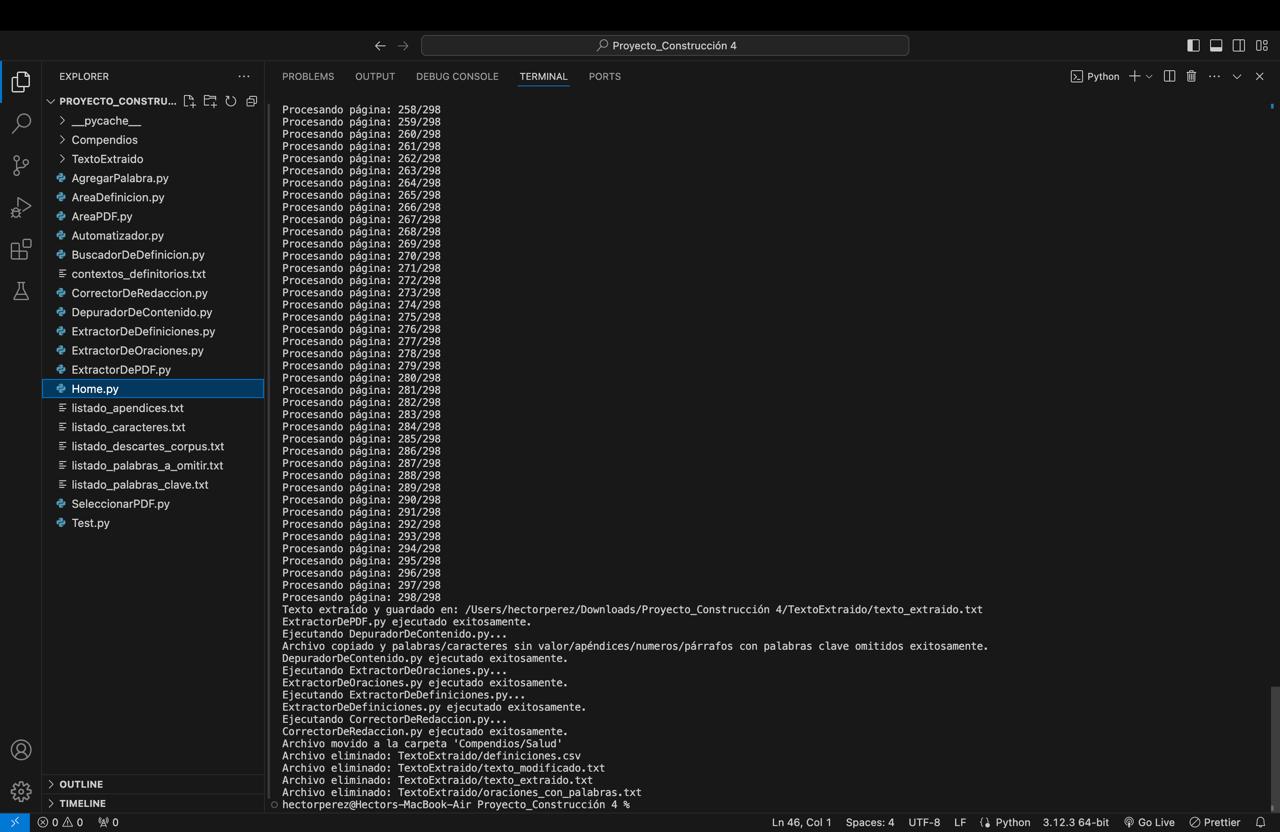
4.3. Fiabilidad: El sistema debe ser altamente confiable y estar disponible las 24 horas del día, los 7 días de la semana, con medidas de respaldo y recuperación de datos en caso de fallos.

4.4. Mantenibilidad: El código del sistema debe estar bien estructurado, modularizado y documentado para facilitar el mantenimiento y la escalabilidad en el futuro.

4.5. Portabilidad: El sistema debe ser compatible con diferentes sistemas operativos y entornos de ejecución, permitiendo su despliegue en una variedad de plataformas.

5. Pruebas de aceptación





Repositorio en GitHub: <https://github.com/Alambrep/CorpusDeDefiniciones>