Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Especialización en Tecnologías de la Información

Trabajo Final

“Construcción de una Estrategia para Anonimizar Bases de Datos de Preproducción en el Poder Judicial de la Provincia de Corrientes”

Autor: Jorge Abel Blanco

Director: Dr. Emanuel Irrazábal

Co-Director/a: Mgter. Martín Rey

Año: 2023

Resumen

El propósito fundamental de este trabajo es la creación de un módulo de anonimización de datos en tecnología Python que operacionaliza un procedimiento operativo para salvaguardar la integridad de información personal altamente sensible almacenada en una base de datos para que sea utilizada en entornos de pruebas y preproducción

Para ello se optó por seguir un modelo de desarrollo iterativo e incremental. Este enfoque se caracteriza por dividir el proceso en una serie de fases claramente definidas, lo que permite una adaptación eficaz a las particularidades del entorno de trabajo que se nos presenta. Se inició con un análisis de datos, seguido por el desarrollo de un módulo de anonimización basado en la técnica seleccionada, su ejecución en datos de prueba, una validación y refinamiento sobre los resultados obtenidos, y finalmente la documentación del proceso.

CAPÍTULO 1: INTRODUCCIÓN

1.1 Fundamentación

Las organizaciones tanto del ámbito público como privado, se encuentran en posesión de una gran cantidad de información de carácter personal, la cual es almacenada en bases de datos para ser explotadas por sistemas software. Estos datos abarcan desde nombres y direcciones hasta números de identificación y referencias personales.

Sin embargo, junto con la valiosa información surge la responsabilidad de proteger la privacidad y confidencialidad de los individuos involucrados. Es en este contexto donde surge el concepto de "anonimización de datos", que es el proceso de desidentificación de datos confidenciales conservando su formato y tipo de datos [1].

Según Abdul Majeed en [2], la anonimización es una solución práctica para preservar la privacidad del usuario en la publicación de datos. Los propietarios de datos, como hospitales, bancos, proveedores de servicios de redes sociales (SN) y compañías de seguros, anonimizan los datos de sus usuarios antes de publicarlos para proteger la privacidad de los mismos, mientras que los datos anónimos siguen siendo útiles para los consumidores legítimos de información.

Al eliminar o modificar los atributos identificables, se reduce el riesgo de que los datos puedan ser utilizados para identificar a individuos específicos. Esto ayuda a mitigar el riesgo de divulgación no autorizada o uso indebido de información personal.

Asimismo, la anonimización de datos juega un papel crucial en la protección de la privacidad y la seguridad de los individuos en la era digital, y más aún cuando se requiere trabajar con grandes volúmenes de datos. Se requiere anonimización de datos antes de que un proceso de big data pueda funcionar de manera efectiva sin comprometer la privacidad de la información personal que utiliza [3].

La calidad de la anonimización sigue siendo un desafío y es necesario proporcionar métricas de privacidad demostrables para garantizar que los datos compartidos públicamente se anonimizan adecuadamente sin comprometer significativamente la utilidad de los datos [4].

Se han propuesto muchas soluciones para la anonimización escalable de big data. Los enfoques existentes de anonimización de datos de subárbol se basan principalmente en plataformas MapReduce para aprovechar la escalabilidad y la rentabilidad [5].

Para lograr la anonimización, el anonimizador de datos debe determinar las tres cuestiones siguientes: En primer lugar, ¿qué datos se conservarán? En segundo lugar, ¿qué conocimiento previo del adversario utilizó para revelar los datos anonimizados? En tercer

lugar, ¿el uso de datos anónimos? Las diferentes técnicas de anonimización conducen a diferentes pérdidas de información [6].

Mediante una auditoría en el área de datos de una entidad pública se detectó que periódicamente se realiza una copia de la base de datos de producción y se lo utiliza en el ambiente de preproducción para verificar la compatibilidad de dichos datos con los servicios en desarrollo que se van a publicar.

Esto trae consigo la exposición de la información personal así como el acceso a otros usuarios. Por todo ello, se ha propuesto el desarrollo de un procedimiento para anonimizar los datos y su operacionalización mediante una herramienta y el uso de buenas prácticas.

1.2 Objetivo General

Desarrollar un módulo de anonimización de datos implementado en tecnología Python y un procedimiento operativo de uso para proteger información sensible de carácter personal almacenadas en una base de datos relacional en el marco de pruebas en pre producción del área de desarrollo software del Poder Judicial de la provincia de Corrientes.

1.3 Objetivos específicos

1.3.1 OE1

Investigar las diferentes estrategias y técnicas de anonimización de datos personales utilizadas en la industria y seleccionar aquella que mejor se adapte a la problemática.

1.3.2 OE2

Desarrollar e implementar un módulo de anonimización de datos implementado en tecnología Python que sea eficiente y escalable para garantizar la protección de los datos personales.

1.3.3 OE3

Establecer un procedimiento sobre el manejo y almacenamiento seguro de datos personales antes, durante y después del proceso de anonimización.

1.3.4 OE4

Evaluar el módulo desarrollado y su procedimiento de ejecución en términos de efectividad y completitud.

CAPÍTULO 2: METODOLOGÍA

En el presente capítulo se describe la metodología utilizada para el desarrollo del módulo de anonimización. Se adoptó el modelo iterativo e incremental para el desarrollo del mismo.

La aplicación de un modelo iterativo e incremental se revela como una estrategia fundamental para lograr un equilibrio entre la protección de la privacidad de los individuos y la preservación de la utilidad de los datos. El enfoque flexible y adaptable de esta metodología, permite la evolución continua del proceso de anonimización a medida que se identifican y abordan nuevos desafíos y se refinan las técnicas empleadas.

A lo largo de este apartado, se explora en detalle cómo esta metodología facilita un enfoque progresivo y efectivo para la anonimización de datos, garantizando la conformidad con las regulaciones de privacidad y la obtención de resultados óptimos en términos de privacidad y utilidad de los datos. Además, se describen las fases adoptadas para el desarrollo del módulo.

2.1 Modelo iterativo e incremental

Para el desarrollo del proyecto se utilizó el ciclo de vida iterativo e incremental. Este modelo es un enfoque de desarrollo de proyectos que implica dividir al mismo en pequeñas etapas o iteraciones, en lugar de tratar de completarlo en una única fase [7]. A medida que se avanza en cada iteración, se incorporan nuevas funcionalidades o mejoras al producto de manera incremental, lo que permite una evolución gradual y continua. Cada iteración produce una versión mejorada del producto, y el proceso se repite hasta que se alcanza la meta final del proyecto. Ver Fig. 1.

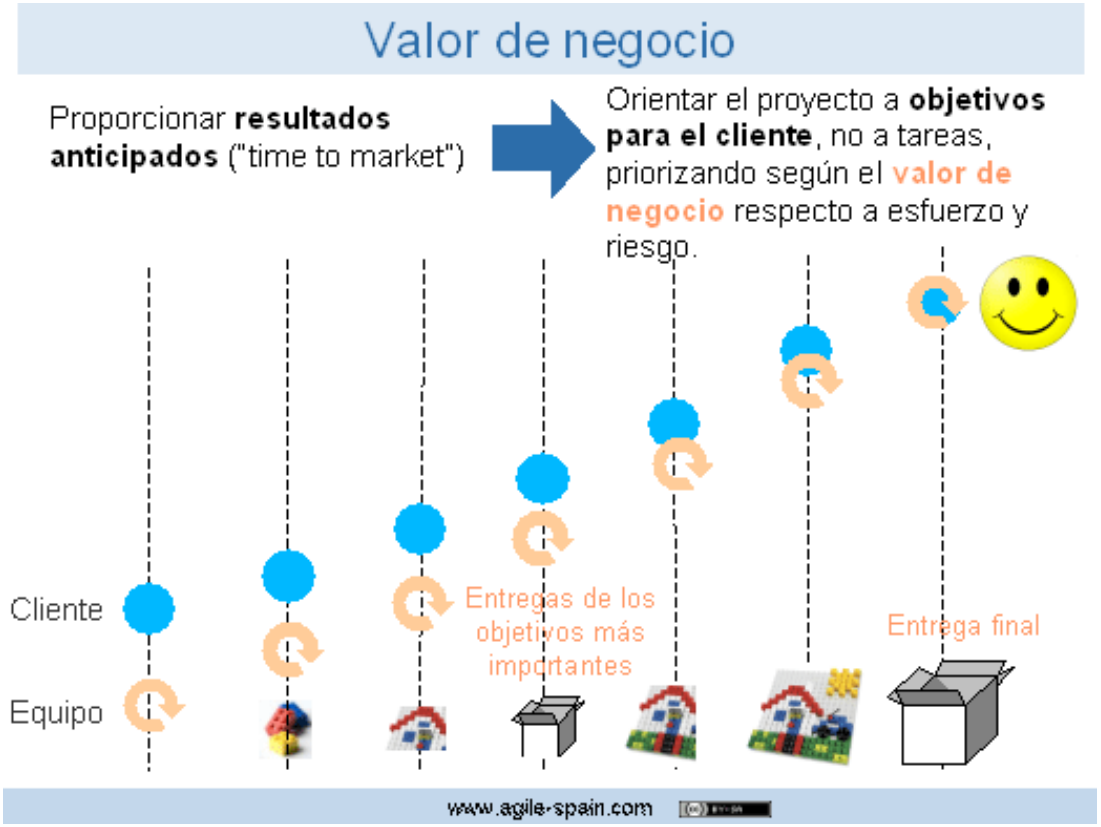


Fig. 1 "Ciclo de vida iterativo e incremental" [7].

2.2 Fases Propuestas

A continuación se definen las fases adoptadas para el desarrollo del proyecto.

Fase 1: Planificación Inicial

En esta fase, se establecen los objetivos generales de anonimización y se identifican las fuentes de datos.

- **Iteración 1 (Incremento 1):**
  - Identificar los datos sensibles.
  - Iniciar la construcción de la estrategia de anonimización, que contenga una planificación inicial.

Fase 2: Implementación

Se implementa la técnica de anonimización basada en el plan inicial.

- **Iteración 2 (Incremento 2):**
    - Desarrollar un módulo en Python para llevar a cabo el plan definido, aplicando las técnicas adoptadas en la Fase 1 a los datos de prueba.
    - Ejecutar el módulo de anonimización sobre los datos de prueba.

Fase 3: Validación y Refinamiento

En esta fase, se validan los resultados de la iteración anterior y se ajusta el proceso de anonimización.

- **Iteración 3 (Incremento 3):**
    - Evaluar la efectividad de la Fase 2 en términos de utilidad.
    - Refinar el proceso de anonimización, considerando técnicas adicionales como la ofuscación de datos en caso de ser necesario con sus respectivas modificaciones en el módulo.

Fase 4: Documentación

En esta fase se finaliza la construcción de la estrategia de anonimización iniciada en la Fase 1, adicionando los pasos faltantes de etapas anteriores..

- **Iteración 4 (Incremento 4):**
    - Finalizar la construcción del procedimiento de anonimización, incluyendo roles intervinientes y etapas restantes.

Este enfoque iterativo e incremental permite que el proceso de anonimización evolucione con el tiempo, adaptándose a nuevos desafíos de privacidad y mejorando la utilidad de los datos. Cada iteración agrega refinamientos y ajustes basados en la retroalimentación y la experiencia acumulada, lo que resulta en un proceso de anonimización más efectivo y confiable.

CAPÍTULO 3: MARCO TEÓRICO

En la era digital, la recopilación y el análisis de datos son componentes esenciales de la toma de decisiones en diversos campos, desde la atención médica hasta el marketing y la investigación académica. Sin embargo, este acceso a datos ricos también plantea preocupaciones fundamentales sobre la privacidad y la seguridad de la información personal. La anonimización de datos surge como una respuesta fundamental para equilibrar la necesidad de datos valiosos con la protección de la privacidad de los individuos [8].

Este apartado busca brindar un marco teórico más amplio en lo que respecta a la anonimización de datos, haciendo foco en las diferentes técnicas de anonimización más utilizadas en la actualidad. Además, se detallan los diferentes conocimientos adquiridos en las materias durante el cursado de la especialización.

3.1 Privacidad de la información personal

La privacidad consiste en mantener la información personal alejada del acceso público. La privacidad es necesaria para la autonomía personal, el individualismo y el respeto. Hay cuatro tipos de privacidad como la información, la corporal, la territorial y la comunicación [9]. La privacidad de la información consiste en recopilar, gestionar, analizar y publicar los datos personales. La privacidad corporal está relacionada con los daños físicos causados por cualquier tipo de procedimientos/medidas invasivas. La privacidad de la comunicación se refiere a cualquier forma de comunicación, como la privacidad de las llamadas telefónicas o los correos electrónicos. La privacidad territorial se refiere a poner límites a la irrupción en una localidad.

Según [10], se pueden identificar cuatro tipos de atributos en lo que respecta a información personal:

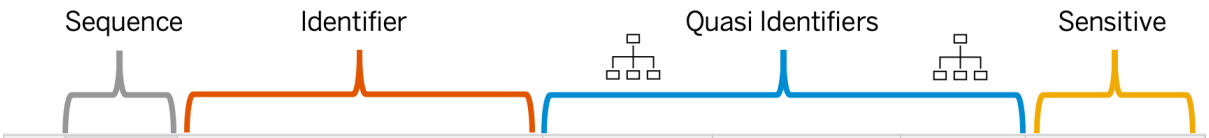
Identificadores directos (DI) : Pueden identificar directa y exclusivamente a un individuo.
Por ejemplo: nombres, DNIs, números telefónicos, fotografías, etc.

Cuasi identificadores (QI): Se pueden vincular con información auxiliar para revelar la identificación de alguien/el valor de SA.
Por ejemplo: edad, código postal, estado civil, etc.

Atributos sensibles (SA): Atributo/información que un usuario/individuo quiere ocultar a los demás.

Por ejemplo, si se tratase de una base de datos delictiva, el campo “crimen” (asesinato, robo, trafico, asalto, etc).

Atributos no sensibles (NSA): Todos los atributos distintos de los identificadores directos, los cuasi identificadores y los atributos confidenciales. Ver Fig. 2.



	ID	NAME	SITE	GENDER	AGE	SALARY
1	1	Henry Brubaker	Madrid	Male	29	91000
2	2	Jaylene Jennings	Paris	Female	19	67000
3	3	Dean Tavalouris	Boston	Male	20	76000
4	4	Lydia Wong	Dallas	Female	18	81000
5	5	Harvey Denton	Madrid	Male	19	45000
6	6	Pamela Doove	Nantes	Female	21	78000
7	7	Luciana Trujillo	Barcelona	Female	24	92000
8	8	Demarcus Collins	Bordeaux	Male	18	67000

Fig. 2 “Tipos de atributos” [12].

3.2 Técnicas de Anonimización

Las técnicas de anonimización de datos son un conjunto de estrategias y métodos diseñados para proteger la identidad de las personas en los conjuntos de datos, eliminando o modificando información personal identificable sin comprometer la utilidad de los datos para análisis legítimos.

Para proteger la privacidad de los usuarios en el conjunto de datos publicado, los propietarios de los datos pueden aplicar una de las siguientes operaciones de anonimización en los datos del usuario original proporcionados en forma de tabla [11].

3.2.1 Generalización

Esta operación transforma los valores del QI original en valores menos específicos pero semánticamente consistentes durante el proceso de anonimización. Por ejemplo, el valor 25 de una edad de QI se puede generalizar con un intervalo [25 – 30] o < 30. Esta operación se basa en la taxonomía de cada QI. Los esquemas de generalización existentes se pueden clasificar en cinco tipos, como generalización de subárbol, generalización de dominio completo, generalización de subárbol sin restricciones, generalización de celda y generalización multidimensional. Ver Fig. 3.

ID	NOMBRE	APELLIDO	EDAD	CUIL
1111	JORGE	AGUIRRE	36	20-34432123-1
1112	LUIS	FERNANDEZ	57	22-32123456-6
1113	JUAN	MEDINA	25	25-41123543-0

ID	NOMBRE	APELLIDO	EDAD	CUIL
1111	JORGE	AGUIRRE	36 - 40	20-******-1
1112	LUIS	FERNANDEZ	56 - 60	22-******-6
1113	JUAN	MEDINA	20 - 25	25-******-0

Fig. 3 "Técnica de generalización aplicada a una tabla" [Elaboración propia].

3.2.2 Supresión

Esta operación oculta un valor original de un QI con un valor especial (por ejemplo, '*'). Por ejemplo, para anonimizar el valor 25 de una edad de QI mediante la operación de supresión, 5 se puede reemplazar con un asterisco, lo que da como resultado 2* como el valor suprimido de QI. La supresión de registros, valores y celdas son las tres variantes de supresión más utilizadas en la anonimización de datos tabulares, subárbol sin restricciones, generalización de celda y generalización multidimensional. Ver Fig 4.

ID	NOMBRE	APELLIDO	EDAD	CUIL
1111	JORGE	AGUIRRE	36	20-34432123-1
1112	LUIS	FERNANDEZ	57	22-32123456-6
1113	JUAN	MEDINA	25	25-41123543-0

ID	NOMBRE	APELLIDO	EDAD	CUIL
1111	JORGE	*	36	*
1112	LUIS	*	57	*
1113	JUAN	*	25	*

Fig. 4 "Técnica de supresión aplicada a una tabla" [Elaboración propia].

3.2.3 Permutación

En esta operación, los registros se dividen en varios grupos y los valores de SA se mezclan dentro de cada grupo. Por lo tanto, las relaciones SA y QI están desasociadas dentro de cada grupo. Esta operación puede generar análisis inexactos en términos de utilidad de datos anónimos, pero la privacidad del usuario se preserva significativamente. Ver Fig. 5.

Tabla Original

ID	GENERO	ESTATURA CM)	PESO (KG)	EDAD	SITUACION LABORAL	SUELDO	PROFESION	FECHA NACIMIENTO	GRUPO
1111	M	177	67	36	EMPLEADO	600K	AGRIMENSOR	10/12/1967	1
1112	F	180	85	57	EMPLEADO	300K	ADMINISTRATIVO	25/04/1978	
1113	F	168	90	25	DESEMPLEADO	-	INFORMATICO	19/07/1990	
1114	M	166	65	37	EMPLEADO	500K	AGRIMENSOR	10/12/1967	
1115	F	161	50	56	EMPLEADO	250K	ADMINISTRATIVO	25/04/1978	2
1116	M	177	76	45	DESEMPLEADO	-	INFORMATICO	19/07/1990	
1117	M	182	90	43	EMPLEADO	550K	AGRIMENSOR	10/12/1967	
1118	M	198	100	65	EMPLEADO	200K	ADMINISTRATIVO	25/04/1978	
1119	F	156	60	25	DESEMPLEADO	-	INFORMATICO	19/07/1990	3
1120	M	170	68	33	DESEMPLEADO	-	AGRIMENSOR	10/12/1967	
1121	F	160	70	27	EMPLEADO	300K	ADMINISTRATIVO	25/04/1978	
1122	M	193	87	34	EMPLEADO	100K	AGRIMENSOR	10/12/1967	

Tabla Permutada

ID	GENERO	ESTATURA CM)	PESO (KG)	EDAD	SITUACION LABORAL	SUELDO	PROFESION	FECHA NACIMIENTO	GRUPO
1111	M	177	67	36	EMPLEADO	300K	INFORMATICO	19/07/1990	1
1112	F	180	85	57	EMPLEADO	500K	ADMINISTRATIVO	25/04/1978	
1113	F	168	90	25	DESEMPLEADO	-	AGRIMENSOR	10/12/1967	
1114	M	166	65	37	EMPLEADO	600K	AGRIMENSOR	10/12/1967	
1115	F	161	50	56	EMPLEADO	550K	AGRIMENSOR	19/07/1990	2
1116	M	177	76	45	DESEMPLEADO	-	ADMINISTRATIVO	25/04/1978	
1117	M	182	90	43	EMPLEADO	250K	INFORMATICO	25/04/1978	
1118	M	198	100	65	EMPLEADO	200K	ADMINISTRATIVO	10/12/1967	
1119	F	156	60	25	DESEMPLEADO	-	ADMINISTRATIVO	25/04/1978	3
1120	M	170	68	33	DESEMPLEADO	-	AGRIMENSOR	10/12/1967	
1121	F	160	70	27	EMPLEADO	100K	INFORMATICO	19/07/1990	
1122	M	193	87	34	EMPLEADO	300K	AGRIMENSOR	10/12/1967	

Fig. 5 "Técnica de permutación aplicada a una tabla" [Elaboración propia].

3.2.4 Perturbación

En esta operación, los valores de datos originales se reemplazan con algunos valores generados sintéticamente. Además, los valores sintéticos se generan de manera que la información estadística no difiere mucho en ambos conjuntos de datos (es decir, conjuntos de datos reales y generados sintéticamente). El grado de perturbación debe ser proporcional al rango de valores de los atributos. Ver Fig. 6.

ID	ESTATURA CM)	PESO (KG)	EDAD
1111	177	67	36
1112	180	85	57
1113	168	90	25

ID	ESTATURA CM)	PESO (KG)	EDAD
1111	177	70	35
1112	182	87	60
1113	170	88	27

Fig. 6 "Técnica de perturbación aplicada a una tabla" [Elaboración propia].

3.2.5 Anatomización

Esta operación no aplica ninguna modificación en los valores de datos originales y, en cambio, los QI y SA se separan en dos tablas. Al hacerlo, se rompe la asociación entre QI y SA y los datos se publican como tablas de QI y SA por separado. En algunos casos, la tabla SA contiene los valores de SA y su frecuencia en el conjunto de datos anonimizados para preservar la privacidad de manera efectiva. Ver Fig. 7.

Tabla Original

ID	GENERO	ESTATURA CM)	PESO (KG)	EDAD	SITUACION LABORAL	SUELDO	PROFESION	FECHA NACIMIENTO
1111	M	177	67	36	EMPLEADO	500K	AGRIMENSOR	10/12/1967
1112	F	180	85	57	EMPLEADO	300K	ADMINISTRATIVO	25/04/1978
1113	F	168	90	25	DESEMPLEADO	-	INFORMATICO	19/07/1990

Tabla QI

ID	GENERO	ESTATURA CM)	PESO (KG)	EDAD
1111	M	177	67	36
1112	F	180	85	57
1113	F	168	90	25

Tabla SA

ID	SITUACION LABORAL	SUELDO	PROFESION	FECHA NACIMIENTO
1111	EMPLEADO	500K	AGRIMENSOR	10/12/1967
1112	EMPLEADO	300K	ADMINISTRATIVO	25/04/1978
1113	DESEMPLEADO	-	INFORMATICO	19/07/1990

Fig. 7 "Técnica de anatomización aplicada a una tabla" [Elaboración propia].

3.3 Generación de datos ficticios

La generación de datos ficticios es una práctica esencial en la anonimización de datos porque ayuda a garantizar la privacidad de las personas, facilita el cumplimiento de regulaciones de privacidad, mejora la seguridad de los datos en entornos de desarrollo y pruebas, y reduce riesgos y costos asociados con el uso de datos reales.

3.3.1 Faker

Faker es un paquete de Python que genera datos ficticios. Ya sea que necesite iniciar su base de datos, crear documentos XML atractivos, completar su persistencia para realizar pruebas de estrés o anonimizar datos tomados de un servicio de producción [12].

Es una herramienta específica en el ámbito de la protección de datos y la privacidad, ya que se encarga de la generación de datos ficticios de alta calidad. Su capacidad para crear información que se asemeja a la realidad, pero que carece de cualquier conexión con individuos reales, desempeña un papel esencial en la anonimización de datos. Al proporcionar conjuntos de datos ficticios que imitan la estructura y la distribución de datos reales, Faker facilita el desarrollo, las pruebas y la capacitación sin comprometer la privacidad de las personas. Ver Fig 8.

	Student	TestScore
0	Amanda Davis	66
1	Shannon Roberson	80
2	Jordan Cowan	67
3	James Jackson	95
4	Brooke Mendez	90
5	Paul Vaughn	64
6	Jennifer Bell	98
7	Victoria Wallace	92
8	Teresa Goodwin	91
9	Andrea Dixon	89

	Student	TestScore	Fake_Student
0	Amanda Davis	66	Donald Thorpe
1	Shannon Roberson	80	Jane Taylor-Griffin
2	Jordan Cowan	67	Mark Bradley
3	James Jackson	95	Antony Brady
4	Brooke Mendez	90	Stephen Grant
5	Paul Vaughn	64	Nigel Wallace
6	Jennifer Bell	98	Mrs Zoe Pearce
7	Victoria Wallace	92	Dr Raymond Fox
8	Teresa Goodwin	91	Michelle Cook
9	Andrea Dixon	89	Lisa Smith-King

Fig. 8 "Datos ficticios mediante Faker" [13].

3.4 Conocimientos Adquiridos

Esta sección detalla cómo los conocimientos adquiridos a lo largo del estudio de materias específicas se aplican de manera práctica en el contexto del proyecto actual. Estos conocimientos disciplinarios son la base que impulsa la comprensión y ejecución de las tareas necesarias para alcanzar los objetivos del proyecto. A lo largo de esta sección, se identifican los campos de estudio clave y cómo se integran para abordar los desafíos y requerimientos del proyecto de anonimización de datos.

Pruebas de software: Se tomarán conceptos de la prueba de software con el objetivo de asegurar la calidad del código fuente del módulo. Se utilizarán conceptos referidos a la refactorización de código para asegurar la calidad del mismo en cuanto a mantenibilidad mientras se desarrolla el software para futuras mejoras.

Base de datos avanzada: Se aplicaran los conocimientos adquiridos en cuanto a buenas prácticas en base de datos relacionales, manejo de tablas y normalización para abordar el desarrollo del proceso de anonimización. Los mismos serán esenciales para llevar a cabo el proceso de anonimización, debido a que los datos están almacenados en tablas de una base de datos relacional.

Proceso de desarrollo de software: Se utilizarán conceptos adquiridos con el objetivo de desarrollar un procedimiento claro y conciso para llevar a cabo el proceso de anonimización. Detallando cada una de las etapas necesarias para que el proceso pueda converger en una anonimización aceptable de datos personales.

Gestión del conocimiento: Se utilizarán técnicas aprendidas para documentar el conocimiento necesario para llevar a cabo el proceso con todas sus etapas.

La importancia de documentar los pasos para una ejecución precisa del proceso de anonimización mediante un manual con todas sus etapas, para evitar futuras pérdidas de conocimiento del proceso como así también ir registrando las posibles mejoras que surjan mediante su uso.

CAPÍTULO 4: CONSTRUCCIÓN DEL ECOSISTEMA DE DESARROLLO

El propósito de este capítulo, es presentar un análisis del ecosistema de desarrollo empleado para la construcción de este proyecto. Este análisis se organiza en tres secciones claramente definidas, cada una con un enfoque específico, detallando las herramientas, lenguaje de programación, librerías e IDEs (entorno de desarrollo integrado) seleccionados y las razones detrás de esta selección.

La intención es proporcionar a los colaboradores y futuros usuarios una comprensión clara de cómo pueden involucrarse en el proyecto, replicar el entorno de desarrollo y colaborar de manera efectiva en la tarea de anonimización de datos. Esta sección no solo sirve como guía para el equipo actual, sino que también es una referencia esencial para aquellos interesados en comprender la base técnica del proyecto y su implementación.

Se detallan tanto bibliotecas y herramientas propias del entorno laboral como así también aquellas a seleccionar como complemento para el desarrollo del presente trabajo. Ver Fig.9.

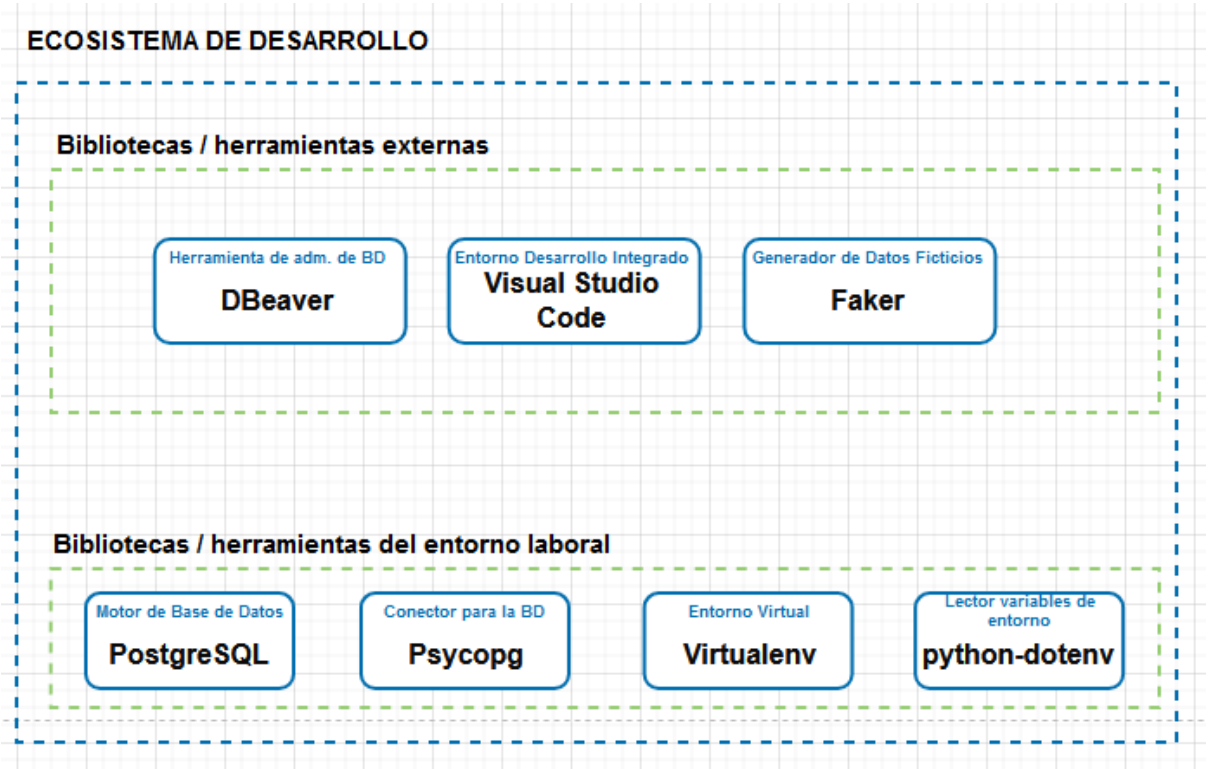


Fig. 9 "Ecosistema de Desarrollo" [Elaboración propia].

4.1 Sección 1: Base de datos

En esta sección, se detallan las herramientas que se han seleccionado y se utilizan para llevar a cabo la interacción con la base de datos, incluyendo aquellas que facilitaron el acceso y la gestión de la información sensible.

4.1.1 Herramienta de Administración de Bases de Datos

En esta sección, se realiza una exploración de la herramienta seleccionada para la tarea de administración de bases de datos. Aquí, no solo se identifican sus características y funcionalidades en profundidad, sino también se resalta su importancia fundamental para el logro de los objetivos específicos del presente proyecto.
Como se puede ver en la Tabla 1, para llevar a cabo la elección se realizó una análisis comparativo entre 3 opciones. Esta comparación proporciona una visión general de cómo las opciones evaluadas se comparan en términos de sus características, capacidades y aptitud para cumplir con las necesidades del proyecto.

Aunque las tres opciones consideradas inicialmente son válidas para su inclusión al ambiente de desarrollo, después de un análisis minucioso, se tomó la decisión de seleccionar DBeaver [14], como la herramienta principal de administración de bases de datos para el proyecto de anonimización.
Es importante destacar que tanto Navicat como DBeaver son herramientas ampliamente utilizadas en el entorno laboral y cuentan con características sólidas para la gestión de bases de datos. Sin embargo, DBeaver ha sido elegida por dos razones fundamentales. En primer lugar, su disponibilidad como software de código abierto gratuito lo hace altamente accesible para todo el equipo, y en segundo, presenta la ventaja adicional de integrar de manera nativa el control de versiones GIT, lo que simplifica la gestión de cambios y colaboración en el proyecto.

Tabla 1 ”Comparación entre Navicat, Dbeaver y HeidiSQL” [Elaboración Propia].

Característica	**Navicat**	**DBeaver**	**HeidiSQL**
Tipo de licencia	Pago	Gratuito	Gratuito
Soporte para múltiples DBMS	Sí	Sí	Sí
Conexión a través de SSH/Túneles	Sí	Sí	Sí
Interfaz gráfica de usuario (GUI)	Sí	Sí	Sí
Edición y ejecución de consultas SQL	Sí	Sí	Sí
Importación y exportación de datos	Sí	Sí	Sí
Generación de diagramas ER	Sí	Sí	No
Depuración y perfilado de consultas	Sí	Sí	No
Control de versiones	Requiere extensión	Integrado	No
Compatibilidad con extensiones y plugins	Sí	Sí	No
Soporte para Windows, macOS y Linux	Sí	Sí	Windows (principalmente)
Costo	Versión de prueba gratuita, luego licencia de pago	Gratuito	Gratuito

4.2 Sesión 2: Desarrollo

Esta sección se centra en las herramientas empleadas en el desarrollo del módulo de anonimización, desglosando las tecnologías y recursos que contribuyeron a la creación de esta pieza fundamental.

4.2.1 Entorno de Desarrollo Integrado

En el proceso de selección del Entorno de Desarrollo Integrado (IDE) [15], siguiendo el enfoque empleado en la sección anterior, se llevó a cabo una comparación entre las tres opciones de IDE más ampliamente reconocidas y utilizadas en la comunidad de desarrollo. Este análisis se realizó con el objetivo de determinar cuál de las alternativas mejor se alinea con las necesidades y los objetivos del proyecto de anonimización de datos. Ver Tabla 2.

Los tres Entornos de Desarrollo Integrado (IDE) que inicialmente se consideran para el proyecto son ampliamente reconocidos y utilizados en el ámbito laboral en diversas áreas y disciplinas. Después de una cuidadosa evaluación, se llegó a la conclusión de que Visual Studio Code es la elección ideal para desempeñar el papel principal de IDE en el proceso de anonimización de datos.

Esta elección se basa en varias consideraciones fundamentales. En primer lugar, su carácter gratuito lo hace altamente accesible para todos los miembros del equipo, en segundo lugar brinda soporte nativo para entornos virtuales (virtual environments), lo que facilita la organización y gestión de las dependencias de proyectos de manera eficiente y sin complicaciones, y por último, posee un amplio conjunto de herramientas de depuración y perfilado, lo que lo convierte en una opción poderosa para el desarrollo y mejora en el código de anonimización.

Tabla 2 "Comparación entre Sublime Text, PyCharm y Visual Studio Code" [Elaboración Propia].

Característica	Sublime Text	PyCharm	Visual Studio Code
Licencia	Pago	Comunidad/Comercial	Gratuito
Lenguajes de programación soportados	Amplia variedad	Python y más	Amplia variedad
Plugins para Python	Sí	Sí	Sí
Integración con frameworks	Requiere extensiones	Integración profunda	Amplia integración
Depuración y perfilado	Requiere extensiones	Requiere extensiones	Integrado
Control de versiones (Git)	Requiere extensiones	Integrado	Integrado
Autocompletado y sugerencias de código	Limitado	Potente	Potente
Soporte para venv.	Requiere extensiones	Requiere extensiones	Integrado
Rendimiento	Rápido	Moderado (depende del proyecto)	Rápido
Personalización	Personalizable	Personalizable	Muy personalizable
Costo	Versión de prueba gratuita, luego licencia de pago	Gratuito (Community) o pago (Professional)	Gratuito

4.3 Sección 3: Python

Esta sección se centra en las herramientas específicas en el entorno Python, destacando las bibliotecas y módulos que enriquecieron el proceso de anonimización y permitieron una implementación efectiva.

4.3.1 Generador de Datos Ficticios

La elección de la biblioteca para la generación de datos ficticios se realizó a través de un proceso de comparación entre las tres principales bibliotecas utilizadas en el mercado. Este proceso de selección se llevó a cabo con el objetivo de identificar la biblioteca más adecuada que cumpla con los requisitos específicos de un proyecto de anonimización de datos y generación de información ficticia. Se consideraron aspectos como la variedad de datos que podían generar, la facilidad de personalización, la capacidad de producir datos realistas y la compatibilidad con diferentes tipos de datos. Ver Tabla 3.

Si bien, Gretel y Mimesis son bibliotecas de generación de datos ficticios utilizadas comúnmente para pruebas y desarrollo de aplicaciones, se elige a Faker [16] porque se enfoca específicamente en la anonimización de datos sensibles, lo que lo convierte en una herramienta valiosa para proteger la privacidad de la información en entornos de desarrollo y análisis de datos.
La misma desempeña un papel crucial en la generación de datos ficticios y sintéticos para campos como nombres de personas, apellidos, números de DNI, CUIT y alias. Al emplear Faker, se logró una simulación precisa y realista de datos sensibles que, si bien mantienen la coherencia y la utilidad en el conjunto de datos, no comprometen la privacidad de los individuos.

Tabla 3 "Comparación entre Gretel, Mimesis y Faker" [Elaboración Propia].

Característica	**Gretel**	**Mimesis**	**Faker**
Tipo de biblioteca	Generador de datos ficticios	Generador de datos ficticios	Biblioteca de anonimización
Personalización de datos	Sí	Sí	Sí
Generación de datos realistas	Sí	Sí	Sí
Anonimización de datos sensibles	No	No	Sí
Soporte de idiomas	Sí	Sí	Sí
Tipos de datos compatibles	Diversos, incluyendo nombres, direcciones, fechas.	Diversos, incluyendo nombres, direcciones, fechas.	Variables personalizadas, textos y números
Simplicidad de uso	Fácil de usar	Fácil de usar	Configuración específica
Extensibilidad	Mediante complementos	Sí	Sí
Biblioteca de anonimización de datos	No	No	Sí
Personalización de estrategias	No	No	Sí
Comunidad y documentación	Comunidad activa y documentación extensa	Comunidad activa y documentación extensa	Comunidad activa y documentación extensa

CAPÍTULO 5: DESARROLLO Y RESULTADOS

En el presente capítulo se enfoca en el desarrollo de la solución propuesta. Incluyendo una breve descripción del escenario actual, siguiendo con el desarrollo del proyecto de acuerdo a las fases propuestas en el CAPÍTULO 3 y además un apartado donde se exponen los resultados de la ejecución del mismo.

5.1 Descripción del escenario actual

Como se mencionó anteriormente en el CAPÍTULO 1 del presente trabajo, una copia de la base de datos de producción es realizada de manera periódica, y se lo vuelca al ambiente de preproducción, con el fin de verificar compatibilidad de datos con los servicios en desarrollo que se van a publicar en producción. Esta compatibilidad de datos puede ser a nivel de estructura (tablas) o a nivel de configuración.

Al llevarse una copia exacta de la base de datos de producción al ambiente de preproducción, datos "reales" como ser información personal de individuos (detenidos, víctimas, imputados, querellas, testigos, funcionarios, etc.) como así también información procesal de causas, legajos, expedientes, quedan expuestas y son accesibles por usuarios que no son del sistema, como por ejemplo: analistas, programadores, testers, etc. Ver Fig. 10.

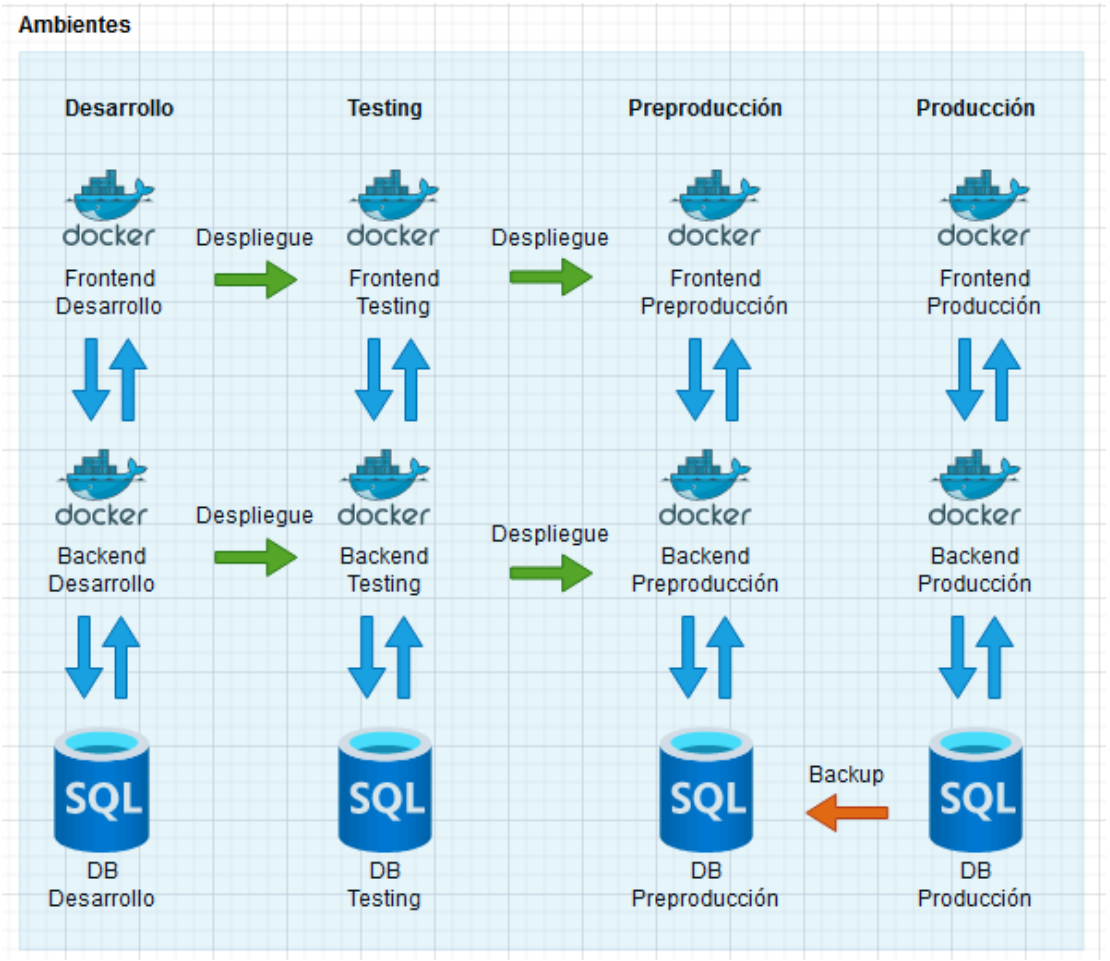


Fig. 10 "Flujo entre ambientes" [Elaboración propia].

5.1.1 Modelo de datos actual

Por razones de confidencialidad y seguridad, no se puede exponer el modelo de datos actual perteneciente al esquema de personas. Proponemos trabajar con un modelo aproximado, con el fin de demostrar que la estrategia va a funcionar con el modelo de datos real. Se procederá a cambiar nombres de tablas y campos para llevar a cabo el trabajo propuesto. Ver Fig. 11.

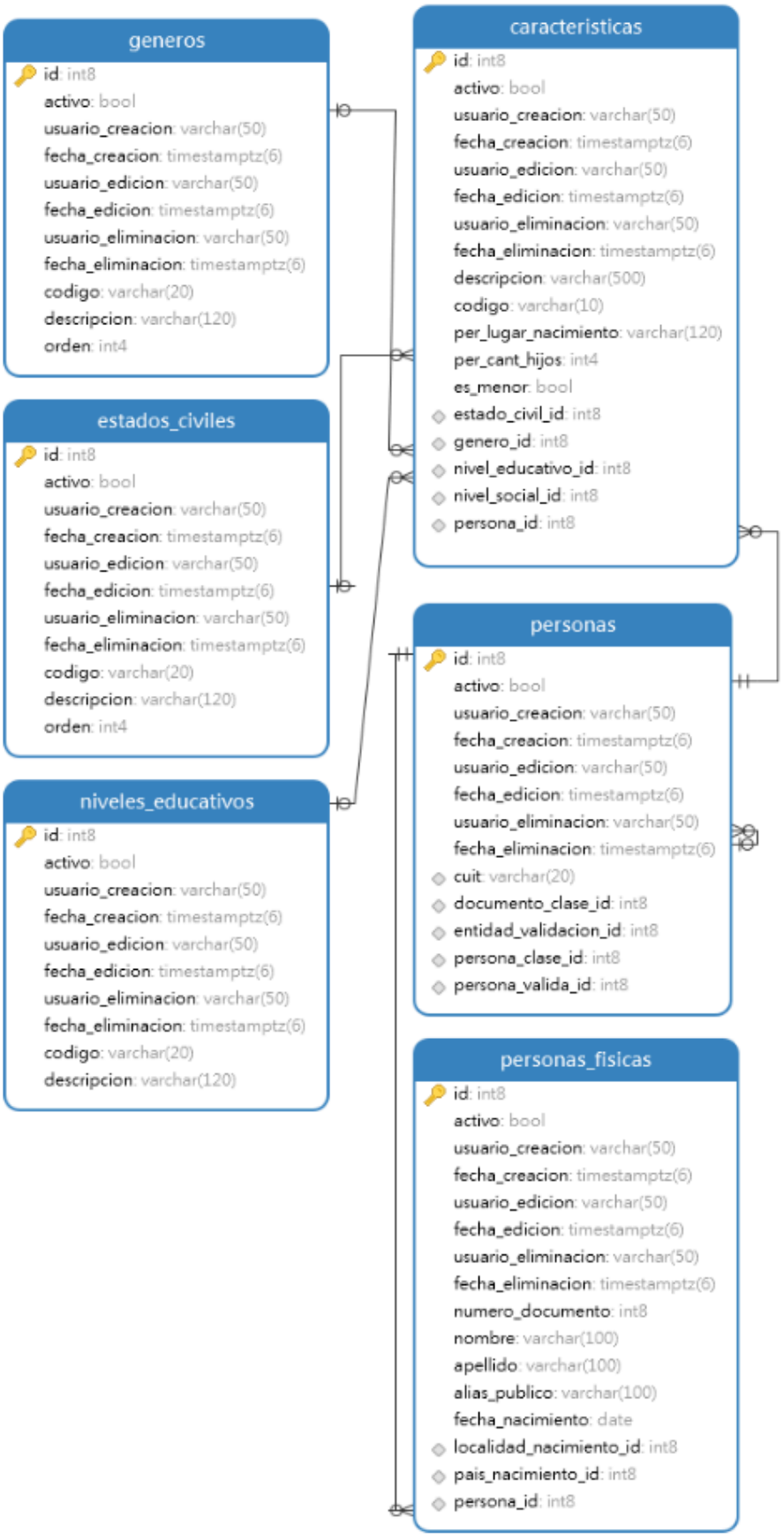


Fig. 11 "Modelo de datos personas" [Elaboración propia].

En la figura anterior se pueden apreciar las tablas presentes en el modelo que son de interés para el desarrollo del presente trabajo en lo que respecta a datos personales.
La tabla **"personas"**, es la tabla base, la misma almacena tanto personas físicas como personas jurídicas.
La tabla **“personas_fisicas”**, almacenan los datos personales como ser dni, nombre y apellido de las personas físicas o humanas.
La tabla **“características”** se utiliza para agrupar datos secundarios presentes en una persona, como ser: estado civil, nivel educativo, género, etc.

5.2 Fases del Desarrollo

Como se mencionó con anterioridad en el CAPÍTULO 2, el desarrollo de este proyecto sigue el ciclo de vida iterativo e incremental.
La misma se divide en fases o etapas bien definidas para facilitar su implementación efectiva. Cada fase representa un conjunto de actividades y tareas específicas que deben llevarse a cabo en un orden lógico para alcanzar los objetivos del proyecto de manera eficiente.

5.2.1 Fase 1 - Planificación Inicial

Durante esta fase se lleva a cabo un análisis exhaustivo de los requisitos de anonimización, se analizan las fuentes de datos en búsqueda de atributos sensibles de carácter personal y definen las primeras etapas de la estrategia de anonimización a desarrollar.

5.2.1.1 Identificación de datos sensibles

Como se dijo anteriormente, la tabla **"personas"**, es la tabla base. Se puede observar un campo llamado "entidad_validacion_id", la misma sirve para identificar qué entidad valió la carga de la persona en el sistema. Las posibles entidades son: 1-Operador. 2-Renaper.
Por razones de completitud de datos, se procederá a trabajar solo con personas validadas por Renaper, los mismos tienen un género asociado. Las personas validadas por Operador entran en la categoría de personas "no validadas", carecen de un género asociado, por lo que resultará difícil generar nombres ficticios en relación a los reales. Ver Fig. 12.

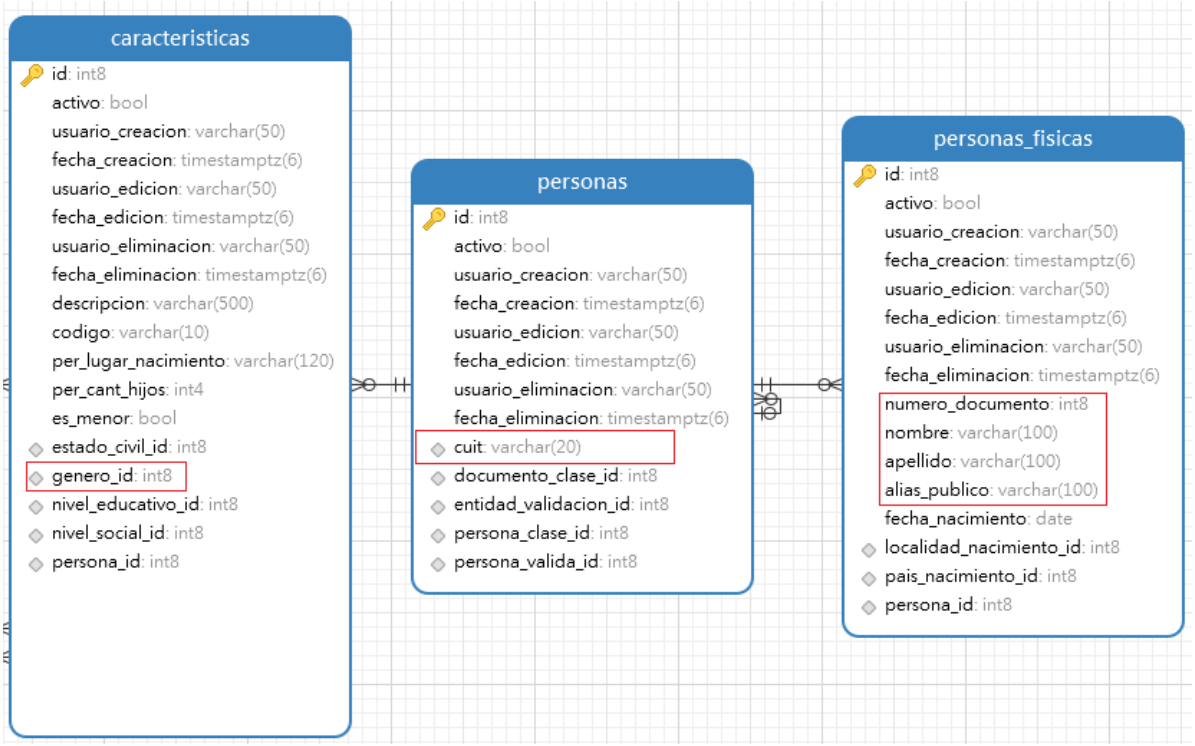


Fig. 12 "Tablas de interés" [Elaboración propia].

En la tabla **"características"** el campo "genero_id" es el que mayor interés presenta, el mismo va a indicar si la persona es masculina o femenina, la cual brindará una mejor correspondencia entre los nombres reales y los generados por el módulo de manera ficticia.

Como se puede observar la tabla **"personas_fisicas"** es la que almacena un mayor número de campos de carácter sensible, mientras que la tabla de "personas" solo almacena el campo cuit, la tabla características, se utilizará como complemento para detectar el sexo de la persona.

Mediante un análisis en conjunto con el área de datos, se detectaron DIs (identificadores directos) presentes en el modelo de datos con los cuales se podrían llegar a utilizar para identificar a una persona.

- En la tabla “personas” el campo **“cuit”**, el mismo es utilizado para almacenar el cuil o cuit de una persona física o una persona jurídica respectivamente.
- En la tabla “personas_fisicas” se detectaron los campos: **“numero_documento”**, **“nombre”**, **“apellido”** y **“alias_publico”**.

5.2.1.2 Estrategia de anonimización

A continuación se describen detenidamente las etapas iniciales de la estrategia para anonimizar bases de datos. Las mismas definen los cimientos de todo el proceso de anonimización, los métodos y prácticas que guiarán la transformación de datos para ocultar o eliminar la identificación de las personas o entidades relacionadas.

Etapa 1: Elección de la técnica de anonimización.

En este apartado se define la técnica de anonimización adoptada para desarrollar el módulo de anonimización del presente trabajo.
Se requiere que se oculten las identidades individuales de las personas, pero además se preserve un nivel aceptable de utilidad de los datos para su posterior análisis o procesamiento. Por este motivo la técnica adoptada fue la de "Perturbación". La razón principal detrás de la elección de la perturbación como técnica de anonimización radica en su capacidad para equilibrar la protección de la privacidad con la utilidad de los datos, lo que la hace especialmente relevante en situaciones donde es esencial mantener un equilibrio entre la confidencialidad y la utilidad.
Se procederá a generar datos ficticios o sintéticos mediante la librería Faker de Python para posteriormente reemplazarlos por los originales.

Etapa 2: Duplicidad de tablas y migración de datos

En esta etapa se procede a duplicar las tablas que contienen datos sensibles, en este caso son las tablas de "personas" y "personas_fisicas".
Cuando el proceso de anonimización de las tablas finalice, los datos originales serán reemplazados con los ficticios, con la imposibilidad de acceder a ellos nuevamente.
Al duplicar tablas, se crean copias idénticas las mismas en cuanto a estructura y datos, pero con un nombre diferente/similar. Con ello, en las tablas duplicadas se logra crear unas tablas del tipo “backup” con los datos originales para su posterior consulta en caso de que se lo requiera. Un caso extraordinario donde podrían ser útiles estas tablas se da cuando el área de estadísticas lleva a cabo un análisis y requieren los datos reales de un determinado individuo.
La nomenclatura adoptada para el nombre de las tablas es la siguiente:

Tabla original	**Tabla duplicada**
personas	personas_duplicada
personas_fisicas	personas_fisicas_duplicada

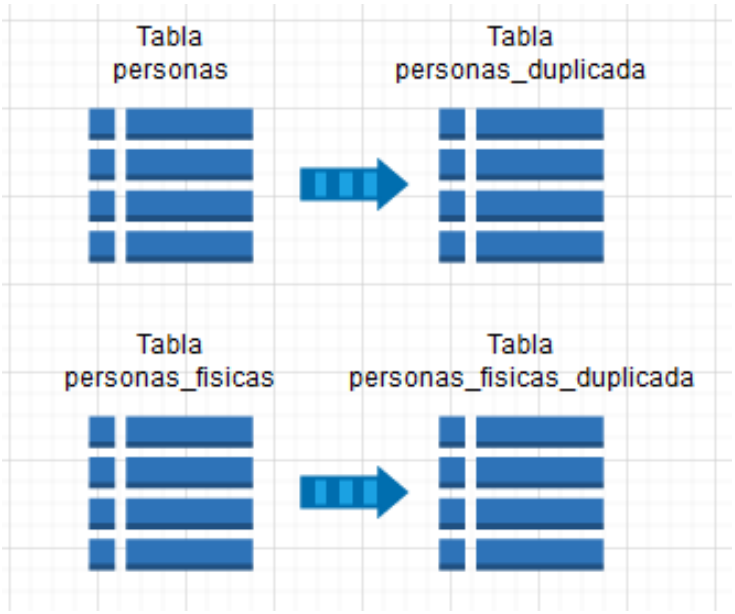


Fig. 13 "Duplicidad de tablas" [Elaboración propia].

La duplicidad y migración de tablas se llevó a cabo mediante consultas SQL. Las mismas duplican las tablas en cuanto a estructura y datos. Se detallan a continuación, cada una para su respectiva tabla.

Tabla personas:

```
CREATE TABLE IF NOT EXISTS personas_duplicada AS SELECT * FROM personas;
```

Tabla personas_fisicas:

```
CREATE TABLE IF NOT EXISTS fisicas_duplicada AS SELECT * FROM
personas_fisicas;
```

Etapa 3: Obtención de los datos de interés

En esta etapa se recopilan y extraen los datos necesarios para poder llevar a cabo el proceso de anonimización. En esta sección, se explica cómo se diseña y ejecuta una consulta SQL con "INNER JOIN" para obtener los datos necesarios en base a las tablas relevantes, cómo se establecen las condiciones de unión y cómo se extraen los datos que cumplen con los criterios específicos de interés.

La principal herramienta que se utiliza es una consulta SQL, particularmente una consulta con "INNER JOIN". La misma permite combinar datos de dos o más tablas relacionadas en una base de datos.
Esta etapa es fundamental para asegurar que los datos sean relevantes, precisos y estén listos para su posterior tratamiento.

Se utiliza esta consulta SQL para extraer información de varias tablas relacionadas en una base de datos. Los resultados contendrán datos de personas físicas, sus géneros, números de documento, CUIT, nombres, apellidos y alias públicos, y solo se incluirán registros que cumplan con la condición especificada en la cláusula WHERE. Ver Fig.14.

```sql
-- Query para obtener el lote de personas a anonimizar
SELECT pf.persona_id, pf.id as persona_fisica_id,
pg.codigo as genero, pf.numero_documento as dni,
pp.cuit as cuit, pf.nombre, pf.apellido,
pf.alias_publico
FROM personas_fisicas AS pf
INNER JOIN personas as pp
ON pf.persona_id=pp.id
INNER JOIN caracteristicas AS pc
ON pc.persona_id=pp.id
INNER JOIN generos as pg
ON pc.genero_id=pg.id
WHERE pp.entidad_validacion_id=2
ORDER BY persona_fisica_id ASC
```

Fig. 14 "Query de obtención de datos" [Elaboración propia].

La consulta SQL de la imagen anterior, es una consulta SELECT con múltiples INNER JOIN que extrae datos de varias tablas de una base de datos relacionada. Aquí hay una breve explicación de lo que hace esta consulta:

1. Selecciona las siguientes columnas de la consulta:
    - persona_id de la tabla fisicas.
    - id de la tabla fisicas con un alias persona_fisica_id.
    - codigo de la tabla generos con un alias genero.
    - numero_documento de la tabla fisicas con un alias dni.
    - cuit de la tabla personas.
    - nombre de la tabla fisicas.

- apellido de la tabla fisicas.
- alias_publico de la tabla fisicas

2. La consulta combina datos de varias tablas utilizando INNER JOIN:
    - La tabla fisicas se combina con la tabla personas a través de la columna persona_id.
    - La tabla personas se combina con la tabla caracteristicas a través de la columna persona_id.
    - La tabla caracteristicas se combina con la tabla generos a través de la columna genero_id.

3. Se aplica una condición WHERE para filtrar los resultados: solo se seleccionan los registros donde el valor de entidad_validacion_id en la tabla personas es igual a 2.

    Esta condición es esencial para sólo tratar con personas del tipo “validadas”, las mismas al estar validadas por la entidad **“RENAPER”,** estarán asociadas a un género específico y así se tendrá una mejor correspondencia entre los nombres reales de las persona y los ficticios generados de manera sintética.

4. La consulta se ordena por persona_fisica_id en orden ascendente (ASC).

5.2.2 Fase 2 - Implementación

En este apartado, se explica en detalle la fase de implementación, centrándose en el desarrollo concreto del módulo en Python y su posterior evaluación de efectividad.

5.2.2.1 Desarrollo del Módulo

Este apartado abordará el proceso de desarrollo del módulo en detalle, explorando la programación de las funciones esenciales y la gestión de datos. Aquí se seleccionan las bibliotecas, se escriben las funciones y se configuran los parámetros esenciales para llevar

a cabo la anonimización de los datos de manera eficiente y precisa. Además, se establecen las bases para futuras optimizaciones y mejoras en el módulo de anonimización.

Esta sección busca proporcionar una visión detallada y explicativa de cómo funciona el código subyacente en el módulo. Sirve como una guía esencial para entender el proyecto en su conjunto, destacando los principales componentes, algoritmos, patrones de diseño y decisiones de implementación.

Para el desarrollo del módulo se optó por el formato de script o scripting en Python.
Según [17], un script es un código particular creado para resolver una tarea específica. Normalmente cumple una o varias de las siguientes características:

- Se compone de un único módulo de python (un fichero de texto con extensión .py)
- Tiene un punto entrada/ejecución al final del mismo.
- Hace uso de librerías del sistema y de comandos de bash.
- Se lanza por consola.
- Se integra con facilidad con tareas tipo cron (cron jobs).
- Define diferentes funciones simples.
- Permite el uso de argumentos desde la consola.

multiplier.py

```python
def multiply(a, b):
    return float(a) * float(b)


def main():
    a = input('Inserte un número: ')
    b = input('Inserte otro número: ')
    res = multiply(a, b)
    print(f'La multiplicación es {res}')


if __name__ == '__main__':
    main()
```

Fig. 15 "Ejemplo de script en Python" [15].

La finalidad del módulo es la de "anonimizar" datos presentes en las tablas, tenemos una sola tarea específica, por lo cual se considera que este formato es ideal para el desarrollo del módulo sin necesidad de utilizar algún framework de terceros.

1 - Estructura del proyecto

Este apartado define la estructura elegida para organizar el proyecto en cuanto a carpetas y archivos presentes en el mismo.

Una estructura bien definida y ordenada facilita la colaboración entre desarrolladores y también asegura que el código sea mantenible y escalable a largo plazo.

Como se puede observar en la Fig.16, se procedió a estructurar el proyecto de la siguiente manera:

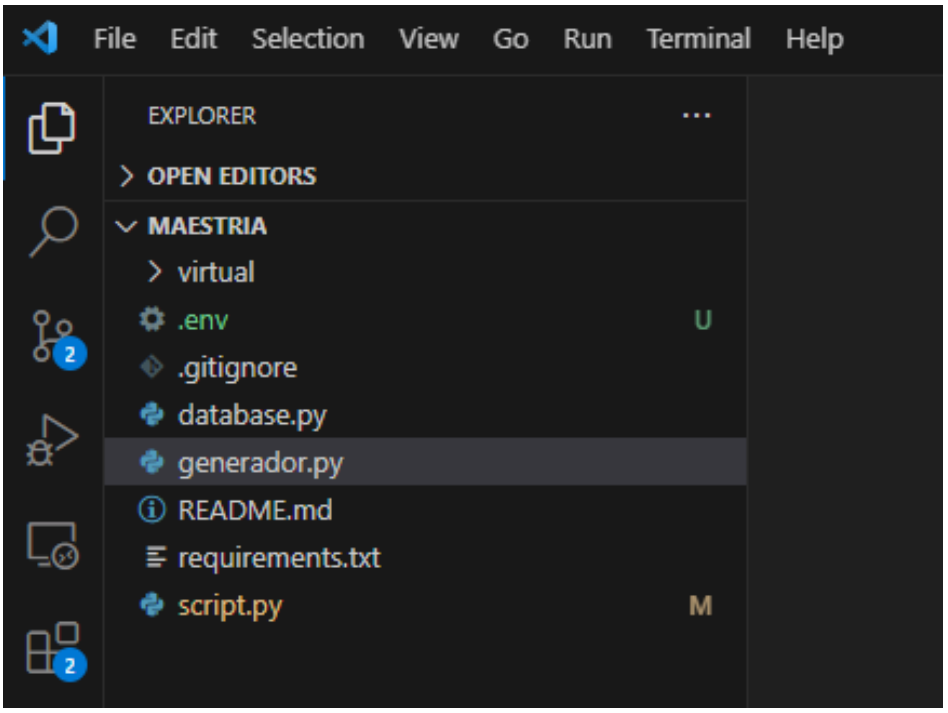


Fig. 16 "Estructura del proyecto" [Elaboración propia].

2 - Descripción de carpetas y archivos

Se describen las carpetas y archivos presentes en el proyecto en cuanto a contenido y funcionalidad.

1 - Carpeta “MAESTRÍA”:

Está ubicada en el nivel superior, es la carpeta principal con el nombre del proyecto. Ésta carpeta contiene todo el código fuente, archivos de configuración y recursos necesarios para la correcta ejecución del módulo.

2 - Carpeta “virtual”:

Está ubicada en el nivel inferior, es la utilizada para ejecutar el entorno virtual del módulo, y así poder aislar las dependencias del proyecto.

3 - Archivo .env:

El mismo es un archivo de configuración utilizado para almacenar variables de entorno del proyecto. Por ejemplo el usuario y contraseña de conexión a la base de datos. Ver Fig.17.

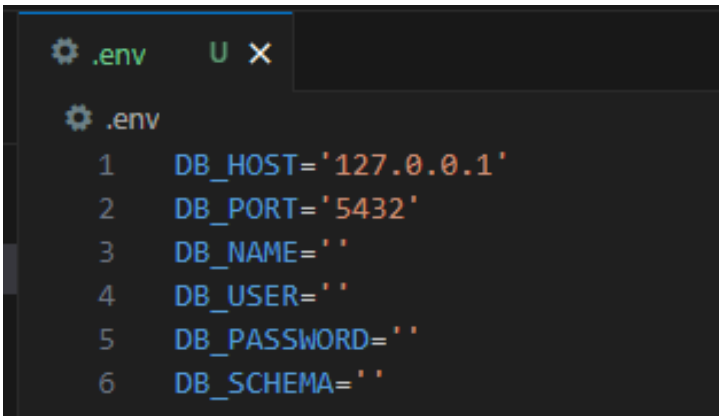


Fig. 17 "Archivo .env" [Elaboración propia].

4 - Archivo .gitignore:

Este es un archivo de configuración utilizado en sistemas de control de versiones, especialmente con Git, para especificar qué archivos y carpetas deben ser excluidos o ignorados durante el seguimiento y la sincronización con el repositorio.

5 - Archivo database.py:

En él, se encuentra el código fuente que se encarga de administrar las conexiones con la base de datos, lo que implica el establecimiento de comunicación, la realización de consultas y la gestión de transacciones.

6 - Archivo generador.py:

El archivo "generador.py" contiene el código fuente que se encarga de la generación de datos aleatorios mediante la librería "Faker".

7 - Archivo script.py:

El archivo "script.py" es el archivo principal del proyecto y actúa como punto de entrada para la ejecución del módulo. En él, se encuentran las instrucciones y llamadas a otros módulos o funciones que conforman el módulo.

8 - Archivo requirements.txt:

El archivo "requirements.txt" se utiliza para enumerar todas las bibliotecas y paquetes de Python que son necesarios para ejecutar el módulo de manera adecuada. Cada línea de este archivo contiene el nombre de una biblioteca y, opcionalmente, su versión específica. Ver Fig. 18.

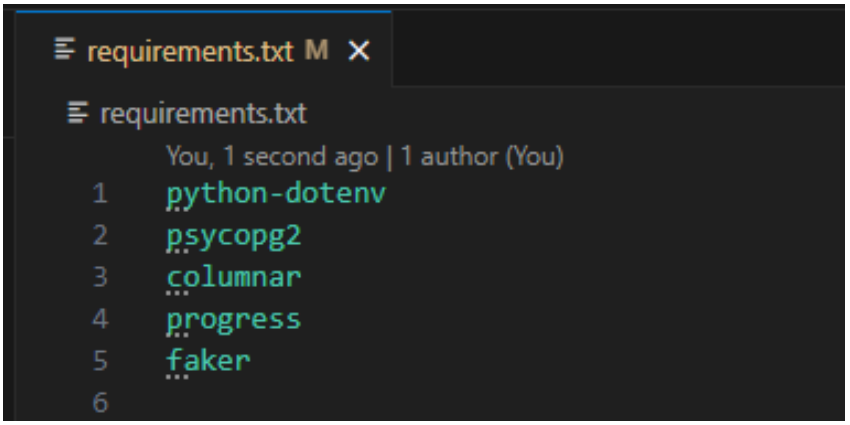


Fig. 18 "Archivo requirements.txt" [Elaboración propia].

3 - Diagramas de clases

En esta sección, se abordan los diagramas de clases relacionados con las clases esenciales para la ejecución del proceso de anonimización de datos. Estas clases, diseñadas con el objetivo de proporcionar una estructura modular y eficiente, son fundamentales para el éxito y la comprensión del proyecto de anonimización.

A través de la representación visual en los diagramas de clases, se busca proporcionar una comprensión más profunda de la arquitectura y las interrelaciones entre las clases esenciales para el desarrollo del presente proyecto.

A continuación, en la Fig. 19 se presenta el diagrama de las clases involucradas:

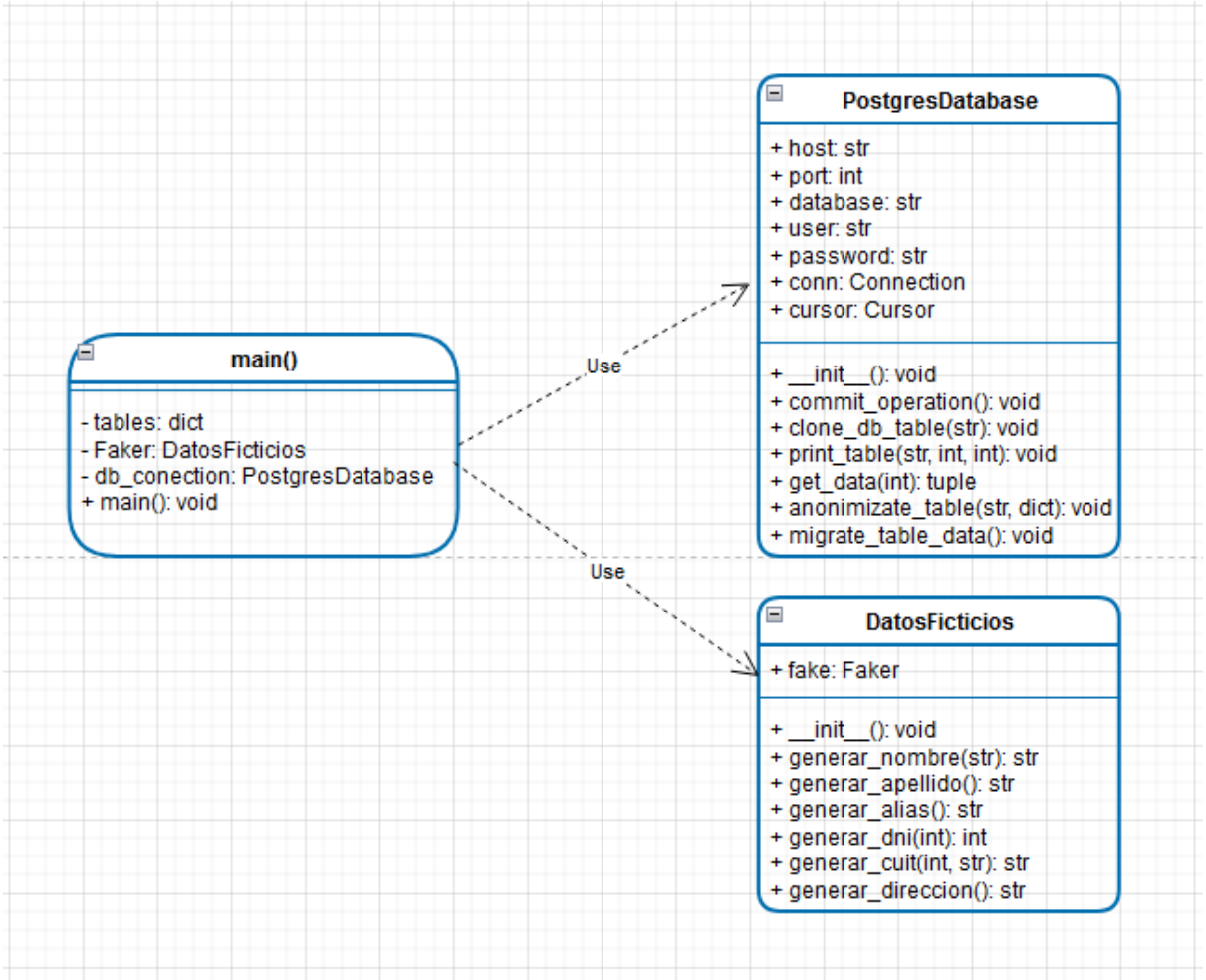


Fig. 19 "Diagrama de clases" [Elaboración propia].

Clase “DatosFicticios”: Generación de Datos Ficticios

La clase **DatosFicticios** desempeña un papel crucial en la generación de datos ficticios que reemplazan los valores originales de la base de datos. Este componente utiliza la biblioteca **Faker** para crear información simulada de manera realista y personalizada, permitiendo así la preservación de la integridad del conjunto de datos.

Se presenta a continuación una breve descripción de sus atributos y métodos:

Atributos:

- **fake:** Objeto de la clase Faker configurado para el idioma español. Se utiliza para generar datos ficticios realistas.

Métodos:

1. **__init__():** Constructor de la clase. Inicializa el objeto Faker para su uso en la generación de datos ficticios.

2. **generar_nombre(str):** Genera un nombre ficticio basado en el género proporcionado. Si no se especifica el género, se genera un nombre sin restricciones de género.
3. **generar_apellido():** Genera un apellido ficticio.
4. **generar_alias():** Genera un alias ficticio o nombre de usuario.
5. **generar_dni(int):** Genera un número de DNI ficticio dentro de un rango válido basado en el DNI original.
6. **generar_cuit(int, str):** Genera un nuevo CUIT ficticio basado en el CUIT original y el DNI generado.
7. **generar_direccion():** Genera una dirección ficticia.

Clase "PostgresDatabase": Interacción con la Base de Datos

La clase **PostgresDatabase** se encarga de establecer y gestionar la conexión con la base de datos PostgreSQL. Su diseño modular facilita la interacción con las tablas, desde la selección del esquema hasta la anonimización de datos. Además, esta clase es responsable de clonar tablas, imprimir información del servidor, y realizar operaciones de actualización de datos en la base de datos. A continuación, se presenta una breve descripción de sus atributos y métodos:

Atributos:

- **host:** Dirección del servidor de la base de datos.
- **port:** Puerto de conexión al servidor (predeterminado es 5432).
- **database:** Nombre de la base de datos.
- **user:** Nombre de usuario para la conexión.
- **password:** Contraseña para la conexión.
- **conn:** Objeto de conexión a la base de datos.
- **cursor:** Objeto cursor para ejecutar consultas SQL.

Métodos:

- **__init__(str, str, str, str, int):** Constructor de la clase que inicializa los atributos de conexión.
- **connect_to_db():** Establece la conexión con la base de datos.
- **print_server_info():** Imprime información sobre la versión del servidor PostgreSQL al que está conectado.
- **select_schema(str):** Selecciona el esquema especificado en la base de datos.
- **close():** Cierra la conexión a la base de datos.

- **commit_operation():** Confirma las operaciones pendientes en la base de datos.
- **clone_db_table(str):** Clona una tabla existente en la base de datos.
- **print table(str, int, int):** Imprime los datos de una tabla en formato tabular.
- **get_data(int):** Obtiene los datos de una tabla limitados por la cantidad especificada.
- **anonimizate_table(str, dict):** Anonimiza datos en una tabla específica.
- **migrate_table_data():** Mueve datos desde una tabla original a una tabla duplicada.

Método “main()”: Orquestación del Proceso de Anonimización

La función **main()** actúa como el punto de entrada principal del proceso de anonimización. En ella, se coordinan las instancias de las clases mencionadas anteriormente para lograr una ejecución ordenada y eficiente del proceso completo. Desde la conexión inicial con la base de datos hasta la generación y aplicación de datos ficticios, main() juega un papel central en la orquestación de estos elementos.

A continuación se presenta una breve descripción de sus componentes claves:

Atributos:

- **tables:** Un diccionario que contiene las tablas a anonimizar en formato 'schema.table_name'. Las tablas originalmente seleccionadas se duplicarán para preservar los datos originales.
- **Faker:** Una instancia de la clase DatosFicticios para generar datos ficticios.
- **db_conection:** Una instancia de la clase PostgresDatabase que facilita la conexión y la interacción con la base de datos PostgreSQL.

Métodos:

- **main():** La función principal que orquesta el proceso de anonimización completo. Coordina la conexión a la base de datos, la selección del esquema, la impresión de información del servidor, la clonación de tablas, la generación de datos ficticios y la aplicación de la anonimización en las tablas seleccionadas.

La función main() encapsula la lógica de ejecución del proceso de anonimización, utilizando las clases DatosFicticios y PostgresDatabase para garantizar un flujo de trabajo coherente y eficiente. Desde la conexión inicial hasta la impresión de resultados anonimizados, este método desempeña un papel central en el desarrollo exitoso del proyecto trabajo

4 - Workflow de ejecución

Este apartado proporciona una visión detallada de los pasos secuenciales que componen la ejecución del script, ofreciendo una comprensión de cómo se desarrolla y se asegura la ejecución exitosa del proceso completo de anonimización de datos.

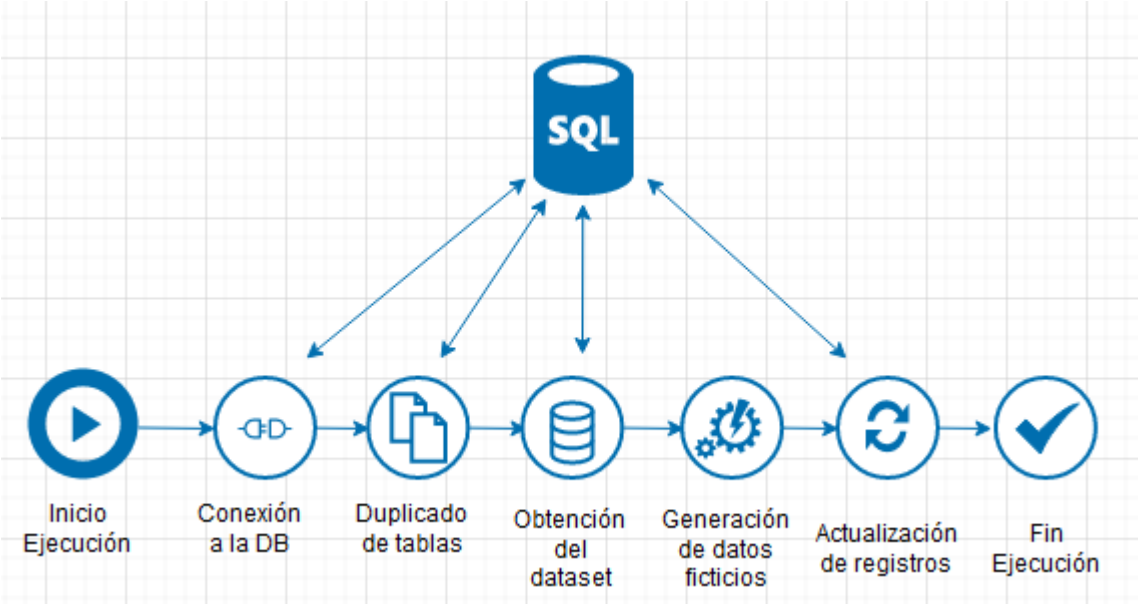


Fig. 20 "Workflow de ejecución" [Elaboración propia].

Pasos del Workflow:

1. **Inicio de ejecución:**
    - Marcando el inicio del proceso, esta fase establece el contexto y la preparación para la ejecución del workflow de anonimización.
2. **Conexión a la base de datos:**
    - La conexión segura con la base de datos PostgreSQL es el primer paso esencial. Esta etapa garantiza el acceso adecuado a los datos originales, estableciendo una base sólida para las operaciones subsiguientes.
3. **Duplicado de tablas:**
    - Con el objetivo de preservar los datos originales, se procede a la creación de duplicados de las tablas seleccionadas. Este paso asegura que la información inicial permanezca intacta durante todo el proceso.
4. **Obtención del dataset:**
    - La extracción del dataset original marca el inicio de la preparación para la generación de datos ficticios. Esta información sirve como base para la posterior aplicación de técnicas de anonimización.

5. **Generación de datos ficticios:**
    - La clase DatosFicticios entra en acción, generando datos ficticios de manera realista y coherente con el contenido original. Este paso es crucial para mantener la validez del dataset mientras se ocultan los detalles sensibles.
6. **Actualización de registros:**
    - Los datos ficticios recién generados se aplican a los registros correspondientes en la base de datos. La clase PostgresDatabase coordina esta actualización de manera segura y eficiente, garantizando la consistencia y precisión del dataset anonimizado.
7. **Fin de ejecución:**
    - La conclusión del workflow señala la finalización del proceso de anonimización. En esta fase, se asegura que todas las operaciones se hayan llevado a cabo con éxito y se han aplicado adecuadamente, culminando en un conjunto de datos protegido y listo para su uso en entornos seguros.

5 - Patrón de diseño

En esta sección, se explora y justifica la elección específica del patrón de diseño que guiará la implementación del módulo de anonimización de datos, proporcionando una visión detallada de cómo la selección cuidadosa del patrón de diseño contribuye a la creación de un módulo de anonimización de datos robusto y altamente funcional.

Como se puede observar en la Tabla 4, para llevar a cabo la selección del patrón se realizó una análisis comparativo entre los patrones de diseño más populares. Esta comparación proporciona una visión detallada de cómo las opciones son evaluadas en términos de propósito, implementación, ventajas y limitaciones de cada patrón de diseño.

Tabla 4 "Comparación entre patrones de diseño" [Elaboración Propia].

Característica	**Pipeline Pattern**	**Prototype Pattern**	**Facade Pattern**	**Singleton Pattern**
Propósito	Divide un proceso en etapas	Permite la creación de objetos nuevos duplicando objetos existentes	Proporciona una interfaz unificada para un conjunto de interfaces	Garantiza que una clase tenga solo una instancia y proporciona un punto global de acceso
Implementación	Secuencia de pasos o etapas	Clonación de objetos existentes	Interfaz unificada para subsistemas	Punto global de acceso con garantía de instancia única
Ventajas	Modularidad, reutilización	Creación de objetos sin conocer detalles de implementación	Simplificación y ocultación de complejidad	Control de instancias, acceso global y evita creación innecesaria
Limitaciones	Posible complejidad con muchos pasos	Manejo de objetos complejos y dependencias en clonación	Limitación de control directo sobre subsistemas	Puede introducir acoplamiento global y dificultar pruebas unitarias
Ejemplos de Uso Común	Procesamiento de datos	Creación de objetos similares	APIs de bibliotecas y frameworks	Configuración de conexiones a bases de datos o logs

Aunque cada opción presenta características potenciales que podrían ser utilizadas de manera efectiva, se ha tomado la decisión de emplear el patrón de diseño Pipeline. Esta elección se fundamenta en la necesidad de un enfoque secuencial y estructurado que sea coherente con la naturaleza del proceso de anonimización.

El patrón Pipeline ha sido seleccionado debido a su capacidad única para organizar de manera secuencial las diversas operaciones involucradas en el proceso de anonimización. Esta organización secuencial no solo facilita la comprensión del flujo de trabajo, sino que también contribuye a la modularidad y reusabilidad del código. Cada etapa del pipeline puede considerarse como un componente independiente, lo que permite una fácil extensión o modificación en el futuro.
Además, el patrón Pipeline se destaca por su capacidad para mejorar la legibilidad del código al representar cada fase del proceso como una entidad separada. Esto simplifica el mantenimiento continuo y permite a los desarrolladores abordar áreas específicas del módulo de anonimización de datos de manera más eficiente.

6 - Análisis estático del código fuente

Este apartado, busca realizar un análisis estático del código fuente del módulo de anonimización, utilizando la herramienta Pylint. Este análisis brinda una mejora en la calidad del software, permitiendo una revisión detallada de la estructura, estilo y posibles problemas en el código antes de su ejecución, como así también, evaluar el código en términos de buenas prácticas, convenciones de estilo y posibles problemas de diseño.

Se procedió a ejecutar la herramienta Pylint sobre los archivos script.py, database.py y generador.py. Se pueden observar los resultados en la Fig. 21, Fig. 22 y Fig. 23 respectivamente:

```
(virtual) C:\Users\Documents\Django Projects\Maestria>pylint script.py
************* Module script
script.py:19:0: C0116: Missing function or method docstring (missing-function-docstring)
script.py:19:0: R0914: Too many local variables (24/15) (too-many-locals)
script.py:19:0: R0915: Too many statements (53/50) (too-many-statements)

-------------------------------------------------------------------
Your code has been rated at 9.59/10 (previous run: 9.59/10, +0.00)
```

Fig. 21 "Pylint sobre archivo script.py" [Elaboración propia].

```
(virtual) C:\Users\Documents\Django Projects\Maestria>pylint database.py
************* Module database
database.py:1:0: C0114: Missing module docstring (missing-module-docstring)
database.py:6:0: C0115: Missing class docstring (missing-class-docstring)
database.py:17:4: C0116: Missing function or method docstring (missing-function-docstring)
database.py:33:4: C0116: Missing function or method docstring (missing-function-docstring)
database.py:39:4: C0116: Missing function or method docstring (missing-function-docstring)
database.py:46:4: C0116: Missing function or method docstring (missing-function-docstring)
database.py:53:4: C0116: Missing function or method docstring (missing-function-docstring)
database.py:57:4: C0116: Missing function or method docstring (missing-function-docstring)
database.py:70:4: C0116: Missing function or method docstring (missing-function-docstring)
database.py:102:4: C0116: Missing function or method docstring (missing-function-docstring)
database.py:130:4: C0116: Missing function or method docstring (missing-function-docstring)
database.py:140:4: C0116: Missing function or method docstring (missing-function-docstring)

-------------------------------------------------------------------
Your code has been rated at 8.75/10 (previous run: 8.75/10, +0.00)
```

Fig. 22 "Pylint sobre archivo database.py" [Elaboración propia].

```
(virtual) C:\Users\Documents\Django Projects\Maestria>pylint generador.py
************* Module generador
generador.py:1:0: C0114: Missing module docstring (missing-module-docstring)
generador.py:3:0: C0115: Missing class docstring (missing-class-docstring)
generador.py:7:4: C0116: Missing function or method docstring (missing-function-docstring)
generador.py:14:4: C0116: Missing function or method docstring (missing-function-docstring)
generador.py:17:4: C0116: Missing function or method docstring (missing-function-docstring)
generador.py:20:4: C0116: Missing function or method docstring (missing-function-docstring)
generador.py:46:4: C0116: Missing function or method docstring (missing-function-docstring)

-------------------------------------------------------------------
Your code has been rated at 7.31/10 (previous run: 7.31/10, +0.00)
```

Fig. 23 "Pylint sobre archivo generador.py" [Elaboración propia].

Se llevaron a cabo todas las mejoras y sugerencias propuestas por la herramienta de análisis estático. La Fig. 24 presenta de manera clara y detallada el resultado de estas modificaciones, reflejando la implementación exitosa de las recomendaciones sugeridas.

```
(virtual) C:\Users\Documents\Django Projects\Maestria>pylint script.py

-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.59/10, +0.41)

(virtual) C:\Users\Documents\Django Projects\Maestria>pylint database.py

-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 8.75/10, +1.25)

(virtual) C:\Users\Documents\Django Projects\Maestria>pylint generador.py

-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run:  7.31/10, +2.69)
```

Fig. 24 "Pylint sobre script.py, database.py y generador.py" [Elaboración propia].

5.2.2.2 Ejecución del módulo desarrollado

En esta sección, se aborda la ejecución del módulo diseñado, detallando los pasos necesarios para ejecutarlo y anonimizar las tablas de manera efectiva. Se exploran las características claves del módulo, su funcionamiento y cómo puede ser adaptado a diferentes necesidades. Además, se proporcionan ejemplos prácticos de cómo aplicar esta técnica de anonimización en diversos escenarios.

1 - Pasos previos a la ejecución

Un paso preliminar esencial antes de ejecutar el proceso de anonimización consiste en la identificación de las tablas que se requiere anonimizar. Para ello, se procede a la creación de un diccionario en el que las tablas se convierten en las llaves, mientras que sus valores correspondientes se configuran como listas vacías. Estas listas cumplen un propósito doble: en primer lugar, sirven para llevar un registro de las tablas que deben someterse a la anonimización, y, en segundo lugar, permiten la acumulación gradual de los registros generados ficticiamente, uno por uno.

```
# Tablas a anonimizar {schema_name}.{tabla_name}
    # Definimos los nombres para su posterior duplicado
    # Nomencaltura 'schema.table_name'
    #tables = ['personas', 'físicas']
    tables = {
        'personas': [],
        'fisicas': []
    }
```

El algoritmo se adapta a los datos identificados en la fase 1, ajustándose individualmente a las particularidades de cada tabla. La generación de datos ficticios se basa exclusivamente en los campos presentes en nuestras tablas y en su nivel de sensibilidad. Se generarán datos ficticios en una cantidad equivalente al número de tablas identificadas durante el proceso.

```
 # ----seccion datos ficticios----
        dni_fict = Faker.generar_dni(dat[3])
        cuit_fict = Faker.generar_cuit(dni_fict, dat[4])
        nombre_fict = Faker.generar_nombre(dat[2])
        apellido_fict = Faker.generar_apellido()
```

```
        alias_fict = Faker.generar_alias()
# ----fin datos ficticios
```

Este caso ejemplifica de manera nítida el principio previamente mencionado, en el sentido de que la generación de datos ficticios se limita estrictamente a aquellos datos de interés que han sido identificados con anterioridad. En otras palabras, se genera información ficticia únicamente en relación con los datos que han sido previamente catalogados como relevantes para la anonimización, lo que garantiza que el proceso de generación de datos ficticios sea coherente y enfocado en la protección de la privacidad de los datos específicos que requieren anonimización, evitando la creación innecesaria de información ficticia no relacionada.

2 - Ejecución

En esta sección, se aborda de manera detallada la ejecución del módulo o script, que es una parte fundamental del proyecto. A través de capturas de pantalla y una guía paso a paso, se irá proporcionando información necesaria para entender y llevar a cabo la ejecución de manera eficiente y sin complicaciones.

Como se mencionó anteriormente, el archivo "script.py" es el archivo principal en este contexto. Para iniciar la ejecución del script, simplemente se utiliza el siguiente comando en su terminal o línea de comandos:

python script.py

Este comando permite ejecutar el script de manera efectiva y llevar a cabo las tareas previstas dentro del script.
Al ejecutar el script, la primera información que se presenta en la pantalla se refiere al estado del servidor de la base de datos, en cuanto al status y la versión del mismo. Ver Fig.25.

```
(virtual) C:\Users\Million\Documents\Django Projects\Maestria>python script.py
Checkeando conexion a la DB...OK!
Conectado a PostgreSQL - Server version ('PostgreSQL 13.3, compiled by Visual C++ build 1914, 64-bit',)
Esquema persona seleccionado
```

Fig. 25 "Información del servidor de base de datos" [Elaboración propia].

Tras esta información inicial sobre el estado del servidor de la base de datos, lo siguiente que se muestra en pantalla son los primeros 10 registros de la tabla "per_personas". Esta visualización inicial proporciona una instantánea de los datos contenidos en la tabla, lo que facilita una rápida comprensión de la información y su estructura antes de proceder con las acciones adicionales del script. Ver Fig. 26.

Tabla actual -> persona.per_personas

id	cuit	documento_clase_id	entidad_validacion_id	persona_clase_id	persona_valida_id
2	3353853847	1	1	1	None
3	233467562	1	1	1	None
4	7154170114	1	1	1	None
7	23346756	1	1	1	None
8	3509219495	1	1	1	None
9	233467561	1	1	1	None
11	20316976248	None	None	1	None
12	27278853212	1	None	1	None
13	203169762484	1	1	1	None
14	203169762481	None	2	1	None

Fig. 26 "Instantánea tabla per_personas" [Elaboración propia].

Inmediatamente después de la visualización de los datos de la tabla "per_personas", se presenta en pantalla una vista de los 10 registros iniciales de la tabla "per_fisicas". Este paso permite a los usuarios obtener un vistazo inicial a los datos contenidos en la tabla "per_fisicas", lo que resulta valioso para comprender la estructura y el contenido de estos registros antes de continuar con las operaciones adicionales que el script pueda realizar.
Ver Fig. 27.

Tabla actual -> persona.per_fisicas

id	numero_documento	nombre	apellido	alias_publico	fecha_nacimiento	localidad_nacimiento_id	pais_nacimiento_id	persona_id
1	[redacted]	JOSE	AYALA	Alias publico	1986-07-22	1	1	2
2	[redacted]	MANUEL	ALTAMIRANO	Alias publico	1986-07-22	None	1	3
3	[redacted]	PEDRO	RODRIGUEZ	Alias publico	1986-07-22	1	1	4
5	[redacted]	MIGUELO	ANGELO	MichelTorino	1986-07-22	1	1	7
6	[redacted]	MIGUELO	ANGELO	MichelTorino	1986-07-22	1	1	8
7	[redacted]	MIGUELO	ANGELO	MichelTorino	1986-07-22	1	1	9
8	[redacted]	NICOLAS	MIÑO		1985-07-17	1	1	11
9	[redacted]	MIRTA CONSUELO	MOREL	mirta more	1980-03-28	1	1	12
10	[redacted]	NICOLAS	MIÑO	Nico	1985-07-17	None	1	13
11	[redacted]	NICOLAS PEDRO	MIÑO		1985-07-17	2	1	14

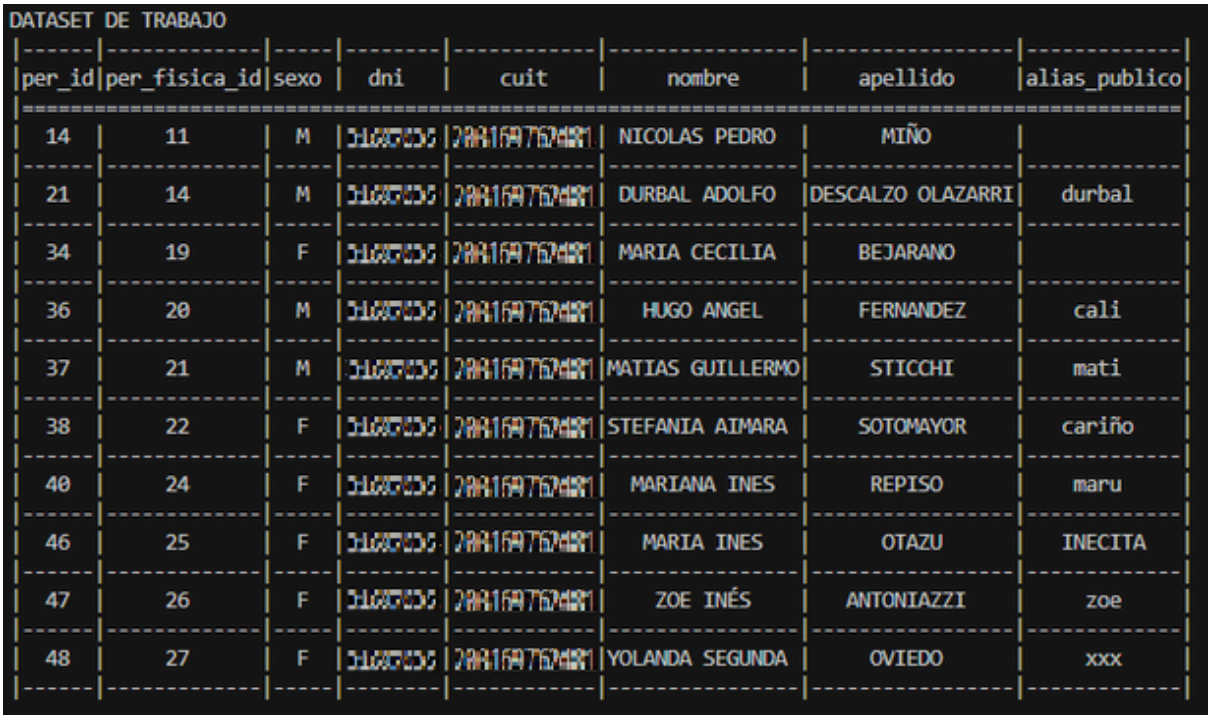
Fig. 27 "Instantánea tabla per_fisicas" [Elaboración propia].

A continuación, en la secuencia de operaciones que se despliegan en la pantalla, se presenta el proceso de impresión por pantalla que refleja la creación de duplicados o clones de las tablas de interés. Este paso tiene un propósito fundamental: respaldar y preservar los datos originales, lo que es esencial antes de cualquier modificación o manipulación adicional. La impresión detallada de este proceso garantiza que se mantenga un registro claro y completo de las tablas originales, lo que es crucial en la gestión segura de datos.
Ver Fig. 28.

```
Clonando tablas...
Tabla persona.per_personas_duplicada creada como duplicado de la tabla persona.per_personas
Tabla persona.per_fisicas_duplicada creada como duplicado de la tabla persona.per_fisicas
```

Fig. 28 "Duplicado de tablas originales" [Elaboración propia].

El siguiente paso en la secuencia de visualizaciones en pantalla muestra el dataset de trabajo en formato de tabla. Este dataset se obtiene mediante una consulta SQL que combina las tablas "per_personas" y "per_fisicas" a través de una operación INNER JOIN, y se enfoca en mostrar únicamente los campos de mayor relevancia. Se presentan, a modo de ejemplo, los primeros 10 registros de esta tabla generada a partir de la consulta SQL. Esta representación visual de datos destaca los elementos clave y proporciona una vista inicial de la información que se utilizará en las etapas posteriores del proceso. Ver Fig. 29.

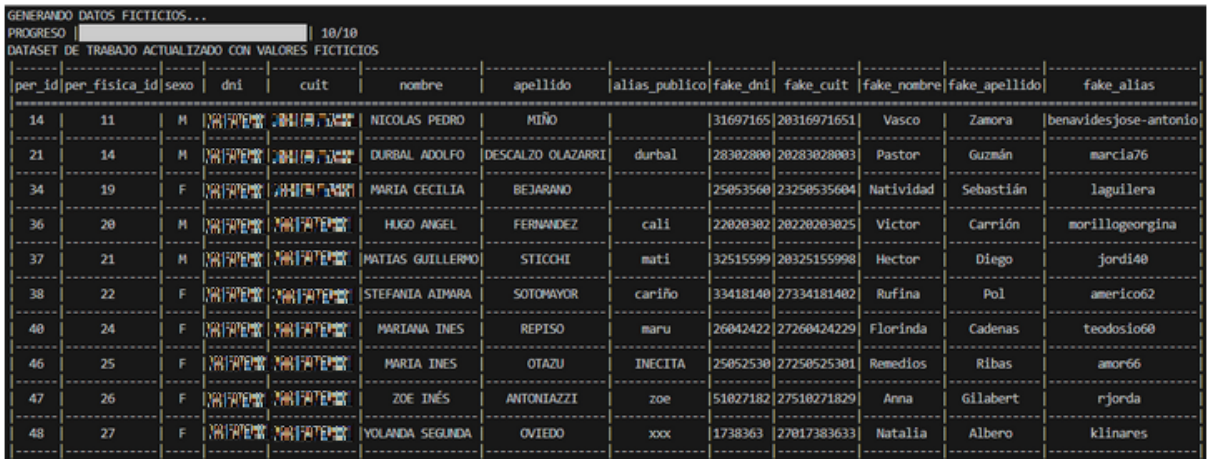


DATASET DE TRABAJO

per_id	per_fisica_id	sexo	dni	cuit	nombre	apellido	alias_publico
14	11	M	[illegible]	[illegible]	NICOLAS PEDRO	MIÑO	
21	14	M	[illegible]	[illegible]	DURBAL ADOLFO	DESCALZO OLAZARRI	durbal
34	19	F	[illegible]	[illegible]	MARIA CECILIA	BEJARANO	
36	20	M	[illegible]	[illegible]	HUGO ANGEL	FERNANDEZ	cali
37	21	M	[illegible]	[illegible]	MATIAS GUILLERMO	STICCHI	mati
38	22	F	[illegible]	[illegible]	STEFANIA AIMARA	SOTOMAYOR	cariño
40	24	F	[illegible]	[illegible]	MARIANA INES	REPISO	maru
46	25	F	[illegible]	[illegible]	MARIA INES	OTAZU	INECITA
47	26	F	[illegible]	[illegible]	ZOE INÉS	ANTONIAZZI	zoe
48	27	F	[illegible]	[illegible]	YOLANDA SEGUNDA	OVIEDO	xxx

Fig. 29 "Dataset de trabajo" [Elaboración propia].

El siguiente punto en la secuencia de visualizaciones consiste en el proceso de generación de datos ficticios, utilizando como entrada los datos previamente extraídos del dataset de trabajo. Este proceso se muestra en detalle, y se ilustra mediante una barra de progreso que proporciona una indicación visual del avance. Una vez que se completa la generación de datos ficticios, se presenta una tabla ampliada que muestra una comparativa entre los datos originales y los datos ficticios generados para cada registro original. Esta tabla es esencial para evaluar el impacto de la anonimización en los datos y comprender cómo se ha preservado la estructura y la utilidad de la información original. Ver Fig. 30.



GENERANDO DATOS FICTICIOS...
PROGRESO | 10/10
DATASET DE TRABAJO ACTUALIZADO CON VALORES FICTICIOS

per_id	per_fisica_id	sexo	dni	cuit	nombre	apellido	alias_publico	fake_dni	fake_cuit	fake_nombre	fake_apellido	fake_alias
14	11	M	[illegible]	[illegible]	NICOLAS PEDRO	MIÑO		31697165	20316971651	Vasco	Zamora	benavidesjose-antonio
21	14	M	[illegible]	[illegible]	DURBAL ADOLFO	DESCALZO OLAZARRI	durbal	28302800	20283028003	Pastor	Guzmán	marcia76
34	19	F	[illegible]	[illegible]	MARIA CECILIA	BEJARANO		25053560	23250535604	Natividad	Sebastián	laguilera
36	20	M	[illegible]	[illegible]	HUGO ANGEL	FERNANDEZ	cali	22020302	20220203025	Victor	Carrión	morillogeorgina
37	21	M	[illegible]	[illegible]	MATIAS GUILLERMO	STICCHI	mati	32515599	20325155998	Hector	Diego	jordi40
38	22	F	[illegible]	[illegible]	STEFANIA AIMARA	SOTOMAYOR	cariño	33418140	27334181402	Rufina	Pol	americo62
40	24	F	[illegible]	[illegible]	MARIANA INES	REPISO	maru	26042422	27260424229	Florinda	Cadenas	teodosio60
46	25	F	[illegible]	[illegible]	MARIA INES	OTAZU	INECITA	25052530	27250525301	Remedios	Ribas	amor66
47	26	F	[illegible]	[illegible]	ZOE INÉS	ANTONIAZZI	zoe	51027182	27510271829	Anna	Gilabert	rjorda
48	27	F	[illegible]	[illegible]	YOLANDA SEGUNDA	OVIEDO	xxx	1738363	27017383633	Natalia	Albero	klinares

Fig. 30 "Dataset de trabajo ampliado" [Elaboración propia].

Luego de este proceso, se presenta una visualización en formato de tabla para cada fila del dataset anterior. Este enfoque se adopta con el propósito de brindar una comprensión más profunda de los datos en términos de su contraste entre los datos reales y los datos ficticios. Estas tablas, al mostrar la información de cada fila de manera detallada y comparativa, permiten una evaluación minuciosa de cómo los datos ficticios se integran con los datos originales. Ver Fig. 31.

CAMPO	DATO REAL	DATO FICTICIO
per_id	14	None
per_fisica_id	11	None
sexo	M	None
dni	[redacted]	31697165
cuit	[redacted]	20316971651
nombre	NICOLAS PEDRO	Vasco
apellido	MIÑO	Zamora
alias_publico		benavidesjose-antonio

Fig. 31 "Fila en formato tabla" [Elaboración propia].

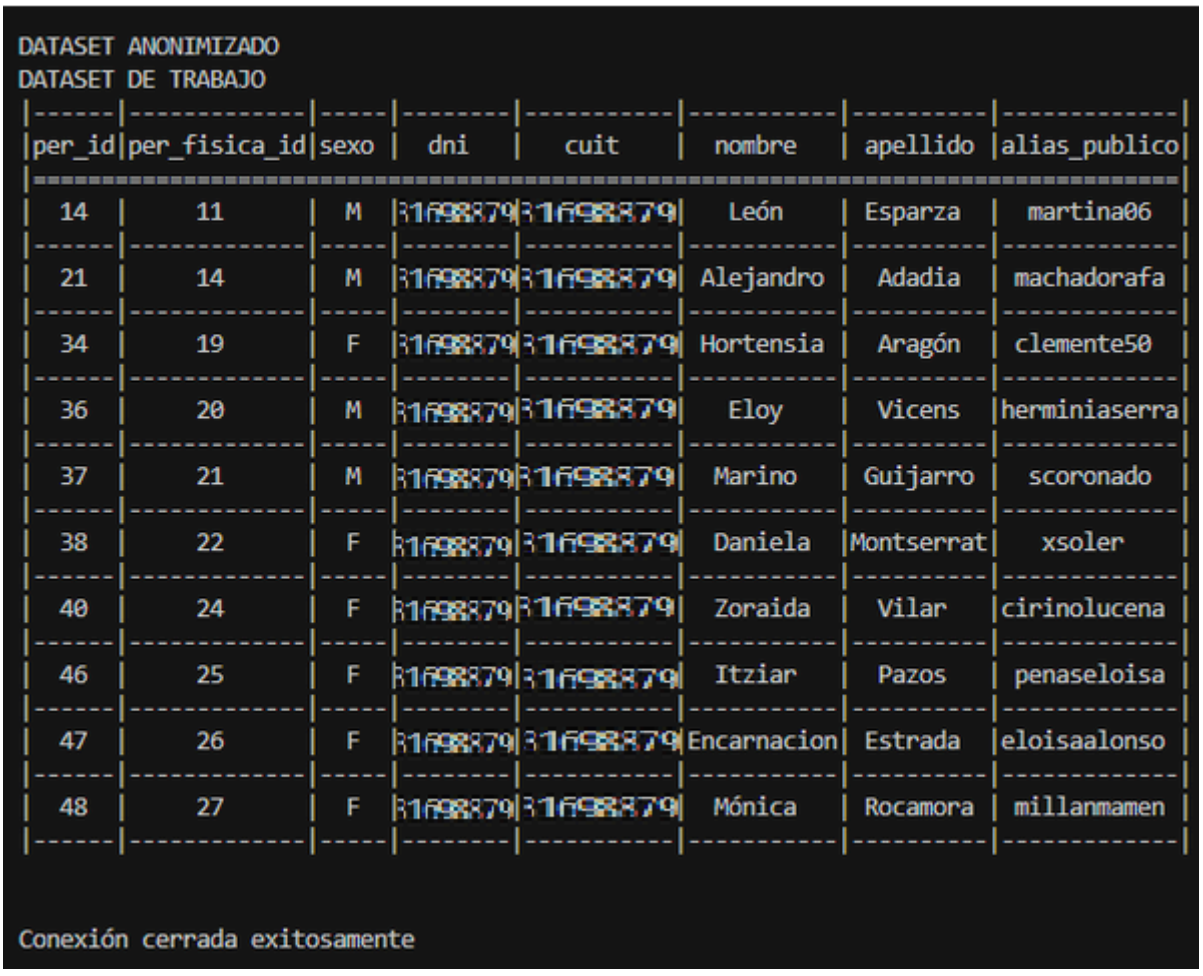
A continuación, en el desarrollo de la ejecución del script, se proporciona una visualización que rastrea el progreso de la anonimización de cada una de las tablas. Este seguimiento se realiza de manera efectiva a través de una barra de estado, que ofrece una representación gráfica del avance de la anonimización en tiempo real. Esta característica es invaluable para que los usuarios puedan supervisar el proceso y asegurarse de que se esté llevando a cabo de manera satisfactoria en cada una de las tablas identificadas. Ver Fig. 32.

```
ANONIMIZANDO DATOS DE LA TABLA: persona.per_personas...
PROGRESO |                                        | 10/10
ANONIMIZANDO DATOS DE LA TABLA: persona.per_fisicas...
PROGRESO |                                        | 10/10
Tablas anonimizadas correctamente
```

Fig. 32 "Progreso de anonimización de tablas" [Elaboración propia].

Siguiendo en el flujo de operaciones, se procede a presentar nuevamente en pantalla el dataset de trabajo, esta vez en formato de tabla, pero con la particularidad de que contiene los datos anonimizados. Este paso es fundamental para evaluar el impacto de la anonimización en los datos y comprender cómo se ha protegido la privacidad de la información sin comprometer su utilidad.

Finalmente, para concluir con el proceso de ejecución del script, se realiza el cierre de la conexión a la base de datos, acompañado del mensaje de confirmación correspondiente. Este cierre adecuado de la conexión es esencial para mantener la integridad de la base de datos y garantizar una ejecución segura y efectiva del script. Ver Fig. 33.



DATASET ANONIMIZADO
DATASET DE TRABAJO

per_id	per_fisica_id	sexo	dni	cuit	nombre	apellido	alias_publico
14	11	M	[illegible]	[illegible]	León	Esparza	martina06
21	14	M	[illegible]	[illegible]	Alejandro	Adadia	machadorafa
34	19	F	[illegible]	[illegible]	Hortensia	Aragón	clemente50
36	20	M	[illegible]	[illegible]	Eloy	Vicens	herminiaserra
37	21	M	[illegible]	[illegible]	Marino	Guijarro	scoronado
38	22	F	[illegible]	[illegible]	Daniela	Montserrat	xsoler
40	24	F	[illegible]	[illegible]	Zoraida	Vilar	cirinolucena
46	25	F	[illegible]	[illegible]	Itziar	Pazos	penaseloisa
47	26	F	[illegible]	[illegible]	Encarnacion	Estrada	eloisaalonso
48	27	F	[illegible]	[illegible]	Mónica	Rocamora	millanmamen

Conexión cerrada exitosamente

Fig.33 "Dataset de trabajo anonimizado" [Elaboración propia].

5.2.3 Fase 3: Validación y Refinamiento

La validación de efectividad es una fase crítica en el ciclo de desarrollo de un módulo de anonimización. Esta etapa permite determinar cuán exitosa es la implementación en la protección de datos sensibles y la preservación de la privacidad. Además, ayuda a identificar posibles debilidades y áreas de mejora en el módulo.
En este apartado, se explica a fondo el proceso de evaluación de la efectividad del módulo de anonimización en cuanto a utilidad de los datos obtenidos. Se examinan las pruebas y procedimientos utilizados para medir la eficacia de la anonimización y la capacidad del módulo para cumplir con los estándares de privacidad requeridos.

5.2.3.1 Validación de utilidad

Al realizar un proceso de anonimización se busca ocultar información personal, pero a su vez es esencial mantener la utilidad de los datos para que sigan siendo valiosos para análisis y aplicaciones legítimas sin exponer la identidad de las personas involucradas.
Este apartado se enfoca en evaluar si los datos anonimizados mantienen su valor y utilidad original.

El responsable de ejecutar el proceso de validación es el analista de datos, es quien lleva a cabo la minuciosa verificación de los registros anonimizados para asegurar su integridad y efectividad.
A continuación se definen las tareas intervinientes en el proceso de validación:

- Verificación visual de los registros anonimizados para identificar posibles errores o datos sensibles remanentes.
- Comparación de los registros anonimizados con los datos originales para asegurar la preservación de la información relevante y la eliminación adecuada de los datos personales.
- Aplicación de pruebas de reidentificación controladas para evaluar la resistencia de los datos anonimizados ante intentos de asociación con individuos originales.
- Revisión exhaustiva de la documentación del proceso de anonimización para asegurar la consistencia y validez de las técnicas empleadas.

Para llevar a cabo la validación, se empleó una copia de la base de datos de preproducción, que contenía alrededor de 15000 registros de personas. El proceso duró aproximadamente 15 minutos y se lograron anonimizar con éxito 12500 registros. Estos 12500 registros corresponden a personas físicas validadas por Renaper.

La estrategia de validación implementada se basa en la técnica de evaluación por pares, un enfoque que implica la comparación entre los datos originales y los datos anonimizados para determinar la preservación adecuada de la estructura y las características fundamentales del conjunto de datos original.
Para la evaluación se utilizan tanto el dataset original como el dataset anonimizado “reducido” a 10 registros, como componentes esenciales para determinar la "utilidad" de los datos anonimizados.

Dataset original

Se emplea la misma consulta de obtención de datos que se utilizó en la fase inicial del proceso, con la incorporación de ligeras modificaciones en las instrucciones de "inner joins". Estas adaptaciones están dirigidas hacia las tablas previamente duplicadas, con el propósito de establecer una suerte de retroceso o "back" a nivel de tabla. El objetivo es comparar y contrastar los registros originales con los datos anonimizados, permitiendo así verificar la efectividad del proceso de anonimización en la preservación de la utilidad de los datos, manteniendo la privacidad sin comprometer la calidad y relevancia de la información para aplicaciones posteriores.

Los resultados de la ejecución de la misma se exponen en la siguiente imagen. Ver Fig. 34.

Mensaje | Resumen | Resultado 1

persona_id	persona_fisica_id	genero	dni	cuit	nombre	apellido	alias_publico
14	11	M	[redacted]	[redacted]	NICOLAS PEDRO	MIÑO	
21	14	M	[redacted]	[redacted]	DURBAL ADOLFO	DESCALZO OLAZARF	durbal
34	19	F	[redacted]	[redacted]	MARIA CECILIA	BEJARANO	
36	20	M	[redacted]	[redacted]	HUGO ANGEL	FERNANDEZ	cali
37	21	M	[redacted]	[redacted]	MATIAS GUILLERMO	STICCHI	mati
38	22	F	[redacted]	[redacted]	STEFANIA AIMARA	SOTOMAYOR	cariño
40	24	F	[redacted]	[redacted]	MARIANA INES	REPISO	maru
46	25	F	[redacted]	[redacted]	MARIA INES	OTAZU	INECITA
47	26	F	[redacted]	[redacted]	ZOE INÉS	ANTONIAZZI	zoe
48	27	F	[redacted]	[redacted]	YOLANDA SEGUNDA	OVIEDO	xxx

Fig. 34 "Dataset original" [Elaboración propia].

Dataset anonimizado

En este caso se continúa haciendo uso de la consulta SQL idéntica a la empleada en la fase inicial del proceso, sin efectuar ninguna modificación en su estructura. Sin embargo, en este contexto, la diferencia radica en que los "joins" se realizan con las tablas que han sido previamente anonimizadas.
Los resultados de la ejecución de la misma se exponen en la siguiente imagen. Ver Fig. 35.

Mensaje | Resumen | Resultado 1

persona_id	persona_fisica_id	genero	dni	cuit	nombre	apellido	alias_publico
14	11	M	31698879	20316988791	León	Esparza	martina06
21	14	M	28304037	20283040373	Alejandro	Adadia	machadorafa
34	19	F	25053441	23250534414	Hortensia	Aragón	clemente50
36	20	M	22020660	20220206605	Eloy	Vicens	herminiaserra
37	21	M	32518287	20325182878	Marino	Guijarro	scoronado
38	22	F	33418848	27334188482	Daniela	Montserrat	xsoler
40	24	F	26041945	27260419459	Zoraida	Vilar	cirinolucena
46	25	F	25052841	27250528411	Itziar	Pazos	penaseloisa
47	26	F	51027556	27510275569	Encarnacion	Estrada	eloisaalonso
48	27	F	1739272	27017392723	Mónica	Rocamora	millanmamen

Fig. 35"Dataset anonimizado" [Elaboración propia].

A continuación se presenta un resumen aportado por el analista de datos, donde se exponen los resultados y observaciones generales de la validación en el escenario propuesto:

1. Cantidad de registros anonimizados sin problemas:

- Se identificaron 12500 registros que fueron anonimizados correctamente.
- De estos 12500 ninguno presentó problemas de identificación y mantuvieron su utilidad para usos posteriores.

2. Motivos por los cuales el proceso de anonimización fue exitoso:

- **Preservación de la Utilidad:** A pesar de la anonimización, la totalidad de registros (12500) mantuvieron información valiosa y útil para análisis estadísticos y de tendencias.
- **Protección de la Privacidad:** El proceso aseguró que la información sensible, como por ejemplos los nombres y DNIs, fueran protegidas de manera efectiva, reduciendo significativamente el riesgo de identificación de individuos.

3. Después de aplicar el riguroso proceso de anonimización a campos sensibles, que abarcan desde el género hasta el DNI, CUIT, nombre, apellido y alias público, se observa que la utilidad de los datos se mantiene en gran medida, lo que constituye un logro notable en la gestión de información sensible.

4. En particular, los campos de "DNI" generados exhiben una sutil similitud con los valores originales, manteniendo valores que se sitúan dentro de un rango admisible en comparación con los datos reales, pero presentando diferencias esenciales que salvaguardan la privacidad. Asimismo, los campos de "CUIT" se generan a partir de los DNIs ficticios, pero

retienen el patrón original de los CUITs reales, conservando los dos primeros dígitos y el último.

5. Respecto a los nombres generados, guardan correspondencia con el género original, dado que nuestro conjunto de datos corresponde exclusivamente a individuos validados por la entidad "Renaper" con género no nulo, lo que permite una mejor concordancia entre los nombres reales y los ficticios.

6. En cuanto a los apellidos, dado el enfoque basado en el género, cualquier apellido generado se considera válido. Por último, los valores generados para el campo "alias público" demuestran ser completamente válidos, y se aprecia que algunos registros previamente carecían de un "alias público," lo que indica que el proceso de anonimización mejora al completar estos campos vacíos de manera efectiva.

En conclusión, la validación del proceso se considera exitosa debido a que se utilizó una copia exacta de la base de datos de preproducción, conteniendo alrededor de 15000 registros, y se lograron anonimizar 12500 registros de manera eficiente. Esta validación se llevó a cabo mediante la técnica de evaluación por pares, permitiendo una comparación exhaustiva entre los datos originales y los datos anonimizados para garantizar la preservación adecuada de la estructura y las características fundamentales del conjunto de datos original, verificando así la efectividad del proceso de anonimización en la protección de la privacidad y la integridad de la información.

5.2.4 Fase 4 - Documentación

A continuación, se detallan las etapas subsiguientes de la estrategia para la anonimización de bases de datos. Es importante recordar que las etapas iniciales se establecieron en la fase 1 del proceso. Estas etapas adicionales son fundamentales para garantizar un proceso completo y efectivo de protección de datos sensibles. Cada una de estas fases desempeña un papel crucial en el logro de los objetivos de anonimización y en la preservación de la privacidad de los datos almacenados en la base de datos.

Procedimiento Operativo: Anonimización de Datos en el Poder Judicial de la Provincia de Corrientes

1 - Alcance

El alcance de este procedimiento es anonimizar bases de datos del área de desarrollo de software, que almacene información sensible de carácter personal y se requiera proteger esa información, tanto en ambientes de preproducción como en cualquier otro ambiente que lo requiera.

2 - Objetivos

El objetivo de este procedimiento es lograr anonimizar datos sensibles del tipo personal presentes en la base de datos de preproducción, con el fin de evitar que la información sea accedida por usuarios no autorizados.

3 - Roles intervinientes

3.1 Desarrollador

El rol del desarrollador desempeña un papel central en la implementación exitosa de la estrategia de anonimización de datos. Las responsabilidades del desarrollador abarcan varias áreas clave:

- **Desarrollo del Script de Anonimización:** El desarrollador es el encargado de programar el script de anonimización, lo que implica la escritura de código que aplique las técnicas de protección de privacidad al conjunto de datos. Esto incluye la importación de bibliotecas necesarias, la gestión de la conexión a la base de datos, la generación de datos ficticios y la ejecución de las consultas requeridas. El script de anonimización es la columna vertebral de la estrategia y debe ser diseñado y programado de manera eficiente y segura.

- **Selección de las Mejores Bibliotecas:** El desarrollador debe seleccionar las bibliotecas y módulos más adecuados para cada etapa del proceso. Esto implica elegir las herramientas que permitirán la generación de datos ficticios realistas y la conexión segura a la base de datos.

- **Adaptación del Script a Contextos Diversos:** En caso de que la estrategia de anonimización sea requerida por otra área o equipo, el desarrollador debe ser capaz de adaptar el script a su contexto específico. Esto implica realizar modificaciones en el código para que se ajuste a las necesidades y particularidades del nuevo entorno, garantizando así una implementación exitosa en diferentes contextos.

3.2 Administrador de base de datos (DBA)

El rol de Administrador de Base de Datos (DBA) desempeña una función crucial en la fase de anonimización de datos. Sus responsabilidades abarcan varios aspectos fundamentales:

- **Duplicación de Tablas de Interés:** El DBA es el encargado de duplicar las tablas de interés, lo que implica crear copias de seguridad de los datos originales que serán anonimizados. Este paso es esencial para garantizar que los datos sensibles permanezcan intactos y no se vean comprometidos durante el proceso de anonimización. La duplicación se lleva a cabo siguiendo las pautas y directrices establecidas, asegurando que los datos originales estén protegidos.

- **Ejecución de Consultas Definidas:** El DBA ejecuta las consultas definidas en el proceso de anonimización. Estas consultas pueden incluir la selección de datos específicos o la extracción de información requerida para la generación de datos ficticios. La ejecución precisa de estas consultas es esencial para la obtención de datos relevantes y la posterior aplicación de técnicas de anonimización.

- **Gestión de Permisos:** El DBA administra los permisos necesarios en las tablas duplicadas de manera que solo las personas autorizadas tengan acceso a ellas. Esto incluye la restricción de acceso a usuarios normales y la garantía de que solo aquellos con los permisos adecuados puedan acceder a los datos sensibles.

3.3 Analista de datos

El rol del analista de datos, perteneciente al área de datos abarca las siguientes responsabilidades:

- **Identificación de atributos sensibles:** El analista de datos es el encargado de identificar los atributos sensibles o que pueden conducir a la identificación de los

individuos. El mismo debe utilizar técnicas de análisis de datos para identificar estos atributos.

- **Verificación de efectividad:** El analista de datos debe verificar la efectividad del módulo de anonimización para corroborar que cumple con los requisitos y que no introduce errores en los datos.

4 - Etapas del procedimiento

A continuación se definen las etapas propias de la estrategia de anonimización propuesta en el presente proyecto.

Etapa 1: Elección de la técnica de anonimización.

El desarrollador selecciona la técnica de anonimización que mejor se ajuste a los requerimientos específicos del módulo en desarrollo. Esta elección se basa en un análisis exhaustivo de diversas técnicas disponibles, considerando meticulosamente las necesidades y particularidades del contexto en el que se implementará el módulo de anonimización.
Un ejemplo de ello se puede observar en el apartado 5.2.1.2 del presente trabajo, que se corresponde a esta precisa etapa.

Etapa 2: Duplicidad de tablas y migración de datos

El administrador de base de datos (DBA) duplica las tablas que contienen información sensible. Este proceso implica la creación de copias exactas de las tablas originales, conservando tanto su estructura como sus datos, pero asignándoles un nombre diferente o similar. Esta duplicación da lugar a tablas adicionales, comúnmente denominadas "backup", que almacenan los datos originales, permitiendo su consulta en el futuro si es necesario. Estas réplicas proporcionan una capa adicional de seguridad y respaldo, asegurando la preservación de la información original para posibles escenarios de recuperación o consulta posterior.
Un caso concreto que ejemplifica este proceso se encuentra detallado en la sección 5.2.1.2 de este documento, específicamente referida a esta fase precisa del procedimiento. En dicho apartado, se presenta un ejemplo ilustrativo que aclara y detalla la ejecución de esta etapa en particular.

Etapa 3: Obtención de los datos de interés

El desarrollador procede a recopilar y extraer los datos fundamentales requeridos para iniciar el proceso de anonimización, mediante el diseño y la ejecución de consultas SQL específicas con el propósito de seleccionar y recuperar únicamente los datos relevantes para el proceso de anonimización, asegurando así la obtención precisa y específica de la información requerida para el siguiente paso del procedimiento.
En la sección 5.2.1.2 de este trabajo se ilustra un ejemplo específico que ejemplifica este proceso detallado. Este ejemplo ofrece una representación concreta de cómo se lleva a cabo esta fase particular del procedimiento.

Etapa 4: Diseño del script de anonimización

En este apartado, se explora en detalle la planificación y desarrollo de un script que transforma datos confidenciales de manera que se mantenga su utilidad sin comprometer la privacidad.

4.1 Elección del lenguaje de programación

El desarrollador selecciona el lenguaje de programación más adecuado para llevar a cabo la implementación del script.. El lenguaje elegido no solo impacta en la eficiencia y efectividad del desarrollo, sino que también puede afectar la escalabilidad, el mantenimiento a largo plazo y la interoperabilidad del proyecto.

En este contexto, se llegó a la conclusión de que Python se erigía como la elección más idónea para la creación del módulo. Esta decisión se fundamenta en su innegable versatilidad, que abarca una amplia gama de aplicaciones, y en su predominante uso en el entorno laboral, particularmente en el ámbito del backend. Al optar por Python, no solo se capitaliza su potencial para el desarrollo de la solución de anonimización, sino que se garantiza una sinergia con las herramientas ya existentes en el entorno de trabajo.

4.2 Selección de librerías

El desarrollador realiza un análisis de las bibliotecas, módulos y recursos adicionales que se necesitan para la implementación del proyecto. Estas herramientas desempeñan un papel fundamental en la construcción de soluciones eficientes y eficaces, aportando funcionalidades predefinidas que ahorran tiempo y esfuerzo.

Un ejemplo de ello se encuentra detallado en la sección 4.3.1 del presente trabajo, específicamente en la selección de la biblioteca destinada a la generación de datos ficticios.

4.3 Pasos del algoritmo:

Esta sección proporciona una guía detallada y secuencial de las acciones y procesos que conforman la implementación de un algoritmo de anonimización de datos. Estos pasos son esenciales para llevar a cabo de manera efectiva la transformación de información sensible en datos protegidos y no identificables, garantizando la privacidad y seguridad de los individuos involucrados. A lo largo de este apartado, se explora cada paso en profundidad, destacando su importancia en el proceso global de anonimización y brindando una visión completa de cómo se logra la protección de datos sensibles.

A continuación, se proporciona una descripción de cada uno de los pasos que componen el algoritmo desarrollado:

- **1- Conexión a la base de datos**: Este paso implica establecer una conexión con la base de datos donde residen los datos que se desean anonimizar. Es el primer paso crucial para acceder a la información que será procesada.

- **2- Duplicado de tablas a anonimizar:** El DBA duplica las tablas de la base de datos que se deben anonimizar. Esta copia de seguridad se utiliza como una medida de seguridad para asegurarse de que los datos originales permanezcan intactos durante el proceso de anonimización.

- **3- Obtención del dataset de trabajo:** En este paso, se obtiene el conjunto de datos específico que se va a anonimizar. Esto puede implicar la selección de tablas, columnas o registros particulares que son el foco de la anonimización.

- **4- Impresión del dataset actual:** Antes de realizar cambios, se imprime o registra el conjunto de datos original para su referencia. Esto ayuda a tener una visión clara de cómo se ven los datos antes de aplicar las técnicas de anonimización.

- **5- Generación de datos ficticios:** En esta etapa, se generan datos ficticios o sintéticos que reemplazarán la información sensible. Los datos ficticios deben ser realistas y coherentes con el contexto de los datos originales.

- **6- Progreso:** Se registra el progreso del proceso de anonimización para realizar un seguimiento de las actividades realizadas, identificar posibles problemas y garantizar que se avance de manera eficiente.

- **7- Actualización de registros:** Los datos ficticios generados en el paso 5 se utilizan para reemplazar los datos sensibles en el conjunto de datos de trabajo. Esto implica la actualización de registros o columnas específicas de acuerdo con las técnicas de anonimización aplicadas.

- **8- Impresión del dataset anonimizado**: Finalmente, se imprime o registra el conjunto de datos anonimizado para su referencia. Esto permite comparar cómo se ven los datos después de aplicar las técnicas de anonimización y asegura que la información sensible se haya protegido de manera efectiva.

Etapa 5: Ejecución del script

El desarrollador ejecuta de manera manual el script de anonimización previamente diseñado y se visualizan por pantalla las secuencias de operaciones propias del proceso.
Dentro de la sección 5.2.2.2 del presente trabajo, se expone un ejemplo concreto que ilustra detalladamente la secuencia de operaciones implicadas en la ejecución del script. Este ejemplo ofrece una visión clara y precisa de cómo se desarrolla cada paso, permitiendo una comprensión más profunda de los procesos involucrados en la ejecución del código.
La ejecución de este script constituye un paso crítico en la estrategia de protección de datos, ya que es donde las técnicas y algoritmos se ponen en práctica para anonimizar la información sensible.

Etapa 6: Validación de resultados y refinamiento

El analista de datos verifica la efectividad de la anonimización y perfecciona el proceso si es necesario. Esta sección trata sobre cómo se lleva a cabo la validación de los resultados, garantizando que los datos sigan siendo útiles y coherentes. Además, se considera la implementación de ajustes y refinamientos, incluyendo técnicas adicionales si es necesario.
La validación y el refinamiento son pasos esenciales para garantizar que los datos se mantengan seguros y anónimos. Esta etapa, se corresponde con la sección 5.2.3 del presente trabajo, donde el analista de datos realiza la validación de los resultados obtenidos.

CAPÍTULO 6: CONCLUSIONES Y FUTUROS TRABAJOS

Conclusión

En resumen, este proyecto de anonimización de datos, es evidente que ha alcanzado los objetivos propuestos de manera satisfactoria. A través de la investigación exhaustiva y la evaluación de diversas estrategias y técnicas en el ámbito de la anonimización de datos personales, se logró seleccionar la metodología más adecuada para abordar la problemática específica del Poder Judicial de la provincia de Corrientes.

La implementación de un módulo de anonimización en tecnología Python no solo cumplió con la eficiencia y escalabilidad esperadas, sino que también se ajustó de manera integral a las necesidades particulares de protección de datos personales. Este módulo proporciona una capa adicional de seguridad crucial para la información sensible almacenada en la base de datos relacional, brindando así una solución robusta para el manejo de datos en el marco de pruebas en pre producción del área de desarrollo software.

La elaboración y establecimiento de un procedimiento detallado sobre el manejo y almacenamiento seguro de datos personales antes, durante y después del proceso de anonimización es un componente clave para garantizar la integridad y la confidencialidad de la información. Este enfoque proactivo hacia la seguridad de los datos contribuye significativamente a la mitigación de riesgos y al cumplimiento de normativas de privacidad.

La evaluación del módulo desarrollado y el procedimiento de ejecución proporciona información valiosa sobre la efectividad y completitud de la solución implementada. Los resultados positivos de esta evaluación respaldan la solidez de nuestro enfoque y la capacidad del sistema para cumplir con los objetivos establecidos.

En conjunto, este proyecto no solo cumple con los objetivos específicos propuestos, sino que también sienta las bases para un manejo seguro y ético de datos personales en entornos de desarrollo. La implementación de este módulo y procedimiento no solo es un logro técnico, sino también un paso importante hacia la protección de la privacidad y la integridad de la información sensible en el ámbito del Poder Judicial de la provincia de Corrientes.

Trabajos Futuros

En cuanto a futuros trabajos, este proyecto sienta las bases para una continua mejora y expansión en la protección de datos en bases de datos PostgreSQL. Una dirección prometedora podría ser la incorporación de técnicas de aprendizaje automático para la detección de datos sensibles de manera más automática y precisa. Además, la adaptación de este módulo de anonimización para ser compatible con diferentes bases de datos, no solo PostgreSQL, podría ser un paso importante para ofrecer una solución más amplia en el ámbito de la seguridad de datos. Asimismo, considerar la implementación de mecanismos de auditoría que permitan un seguimiento más detallado de las actividades de anonimización y el monitoreo constante de posibles vulnerabilidades podría ser esencial. En resumen, este proyecto proporciona una base sólida para futuras investigaciones y desarrollos que continúen fortaleciendo la protección de datos sensibles en entornos de bases de datos.

ANEXO

1 - Descripción del código fuente

En esta sección se busca permitir a los desarrolladores, colaboradores y otros interesados obtener un mejor entendimiento e información valiosa sobre la estructura, la lógica y el funcionamiento del código presente en los archivos.

1 - Archivo database.py:

El código fuente presente en el archivo database.py es una implementación en Python de una clase llamada PostgresDatabase, que está diseñada para interactuar con una base de datos PostgreSQL. Se encarga de la conexión, selección de esquemas y cierre de la conexión de manera estructurada. El uso concreto de esta clase en una aplicación requeriría llamar a los métodos apropiados para llevar a cabo operaciones en la base de datos.

```
import psycopg2
import time
from columnar import columnar
import json

class PostgresDatabase:

    def __init__(self, host:str, database:str,
                 user:str, password:str,
                 port:int=5432) -> None:
        self.host = host
        self.post = port
        self.database = database
        self.user = user
        self.password = password
        self.conn = None
        self.cursor = None

    def connect_to_db(self):
        print(
           "Checkeando conexion a la DB...", end='',
           flush=True)
        try:
```

```
            self.conn = psycopg2.connect(
                host=self.host,
                database=self.database,
                user=self.user,
                password=self.password)
            self.cursor = self.conn.cursor()
            time.sleep(2)
            print("OK!")
        except Exception as e:
            print(f"Error de conexión: {str(e)}")
            raise
    def print_server_info(self) -> str:
        time.sleep(2)
        self.cursor.execute("select version()")
        data = self.cursor.fetchone()
        print(f"Conectado a PostgreSQL - Server version {data}")

    def select_schema(self, schema_name):
        try:
            self.cursor.execute(
                f"SET search_path TO {schema_name}")
            print(f"Esquema {schema_name} seleccionado \n")
        except Exception as e:
            print(f"Error al seleccionar el esquema: {str(e)}")

    def close(self):
        if self.cursor:
            self.cursor.close()
        if self.conn:
            self.conn.close()
            print("Conexión cerrada")

    def commit_operation(self):
        if self.conn:
            self.conn.commit()
```

Aquí se presenta una breve descripción de las partes clave del código:

1. **Importaciones**: El código comienza importando varios módulos necesarios, incluyendo psycopg2 para la conexión a PostgreSQL, time para pausas en el programa, columnar para imprimir datos de una manera tabular, y json para manejar datos en formato JSON.
2. **Clase PostgresDatabase**: Esta clase es una abstracción que se utiliza para interactuar con una base de datos PostgreSQL. Sus métodos y propiedades son los siguientes:
    - **Constructor (__init__)**: Se utiliza para inicializar los parámetros de conexión a la base de datos, como el host, la base de datos, el usuario y la contraseña. El puerto es opcional y tiene un valor predeterminado de 5432.
    - **connect_to_db**: Este método intenta establecer una conexión a la base de datos PostgreSQL utilizando los parámetros proporcionados en el constructor. Imprime un mensaje indicando si la conexión se realizó con éxito o si hubo un error.
    - **print_server_info**: Este método consulta la versión del servidor PostgreSQL al que se ha conectado y la imprime en la consola.
    - **select_schema**: Permite seleccionar un esquema específico en la base de datos. Establece el esquema activo para la conexión actual.
    - **close**: Cierra la conexión a la base de datos. Primero cierra el cursor, si está abierto, y luego cierra la conexión. Imprime un mensaje indicando que la conexión se ha cerrado.
    - **commit_operation**: Realiza una operación de confirmación en la conexión. Esto se utiliza para confirmar cualquier cambio pendiente en la base de datos.

Además se presentan una serie de métodos adicionales dentro de la clase para la obtención de datos y llevar a cabo el proceso de anonimización.

```
def clone_db_table(self, table_name=None) -> None:
    # Definir el nombre de la nueva tabla duplicada
    new_tabla = f"{table_name}_duplicada"
    schema_name = ''
    # Ejecutar la consulta SQL para duplicar la tabla
    try:
        self.cursor.execute(
           f'''CREATE TABLE IF NOT EXISTS {new_tabla}
            AS SELECT * FROM {table_name}''')
        self.conn.commit()
        print(
```

```
            f"Tabla {new_tabla} creada como duplicado de la tabla
            {table_name}")
    except Exception as e:
     # En caso de error, deshacer los cambios
        self.conn.rollback()
        print(f"Error al duplicar la tabla: {str(e)}")


def print_table(self, table_name, limit=10, head=3) -> None:
     fields_exclude=['activo','usuario_creacion','fecha_creacion',
          'usuario_edicion','fecha_edicion',
          'usuario_baja','fecha_baja']        ]
     #listado para almacenar los índices de las columnas
     index_include = []
     #listado para almacenar los nombres de las columnas
     column_names = []

     # Ejecuta una consulta SQL para seleccionar los datos de la
tabla
     self.cursor.execute(
     f"SELECT * FROM {table_name} LIMIT {limit};")

     # Obtiene los resultados de la consulta
     data = self.cursor.fetchall()
     #list_data = list(map(list, data))

     # Obtiene los nombres de las columnas
     for i, value in enumerate(
          [desc[0] for desc in self.cursor.description]):
           if value not in fields_exclude:
               column_names.append(value)
               index_include.append(i)
        list_data2 = [[value for i, value in enumerate(dat)
          if i in index_include] for dat in data]

        table_format = columnar(
          list_data2, column_names, justify='c', head=1)
```

```
        print('Tabla actual -> ', table_name)
        print(table_format)
        print(' ')



 def get_data(self, limit:int=10):
     # Ejecuta una consulta SQL para seleccionar los datos de la
     tabla
     self.cursor.execute(
     "SELECT pf.persona_id AS per_id, pf.id as per_fisica_id, "
     "pg.codigo as sexo, pf.numero_documento as dni, pp.cuit, "
     "pf.nombre, pf.apellido, pf.alias_publico "
     "FROM fisicas AS pf "
     "INNER JOIN personas as pp "
     "ON pf.persona_id=pp.id "
     "INNER JOIN caracteristicas AS pc "
     "ON pc.persona_id=pp.id "
     "INNER JOIN generos as pg "
     "ON pc.genero_id=pg.id "
     "WHERE pp.entidad_validacion_id=2 "
     "ORDER BY per_fisica_id ASC LIMIT "+str(limit))

     # Obtiene los resultados de la consulta
     data = self.cursor.fetchall()
     list_data = list(map(list, data))

     # Obtiene los nombres de las columnas
     column_names = [desc[0] for desc in self.cursor.description]
     table_format = columnar(
         list_data, column_names, head=5, justify='c')
     print('DATASET DE TRABAJO')
     print(table_format,'\n')
     return list_data, column_names

def anonimizate_table(
    self, table_name=None, data=None, *args, **kwargs) -> None:
```

```
where = data.pop('id')
set = ",".join([str(item[0])+"='"+str(item[1])+"'"
      for item in data.items()])
query = f"UPDATE {table_name} SET {set} WHERE id='{where}'"
    try:
        self.cursor.execute(query)
    except Exception as e:
       # En caso de error, deshacer los cambios
       self.conn.rollback()
       print(f"Error al anonimizar los datos: {str(e)}")
```

Estos métodos proporcionan funcionalidades específicas para interactuar con la base de datos PostgreSQL, como duplicar tablas, imprimir datos de tablas, recuperar datos complejos y actualizar registros en la base de datos. Cada uno de ellos está diseñado para realizar una tarea particular en la gestión de la base de datos. A continuación se expone una breve descripción de los métodos en si:

1 - clone_db_table(self, table_name=None) -> None:

- Este método se utiliza para duplicar una tabla en la base de datos PostgreSQL. Puedes especificar el nombre de la tabla que deseas duplicar como argumento (table_name).
- Primero, genera un nuevo nombre para la tabla duplicada al agregar "_duplicada" al nombre original.
- Luego, ejecuta una consulta SQL que crea una nueva tabla (si aún no existe) con el nombre generado y copia todos los datos de la tabla original a la tabla duplicada.
- Realiza un commit para guardar los cambios y muestra un mensaje indicando que la tabla duplicada se ha creado con éxito. En caso de error, se realiza un rollback para deshacer los cambios y se muestra un mensaje de error.

2 - print_table(self, table_name, limit=10, head=3) -> None:

- Este método se utiliza para imprimir los datos de una tabla en formato tabular.
- Recibe el nombre de la tabla que se desea imprimir (`table_name`), un límite para el número de filas a imprimir (`limit`) y un encabezado (`head`) que indica cuántas filas del conjunto de resultados se considerarán como encabezado.
- Realiza una consulta SQL para seleccionar los datos de la tabla y luego filtra las columnas que no están en la lista `fields_exclude`.

- Formatea los datos en una tabla utilizando el módulo `columnar` y muestra la tabla en la consola.

3 - get_data(self, limit:int=10):

- Este método se utiliza para realizar una consulta SQL compleja que selecciona datos de varias tablas y devuelve un conjunto de resultados.
- La consulta selecciona información de diferentes tablas relacionadas y utiliza un límite (limit) para limitar el número de filas devueltas.
- Después de ejecutar la consulta, los resultados se almacenan en una lista y se devuelven junto con los nombres de las columnas.

4 - anonimizate_table(self, table_name=None, data=None, *args, **kwargs) -> None:

- Este método se utiliza para actualizar datos en una tabla de la base de datos PostgreSQL. Se espera que se proporcione el nombre de la tabla (table_name) y un diccionario de datos (data) que especifica qué columnas y valores se deben actualizar.
- El método construye una consulta SQL de actualización que utiliza los datos del diccionario para actualizar la fila con un identificador específico (en este caso, con el campo "id").
- Si la actualización se realiza con éxito, no se muestra ningún mensaje. En caso de error, se realiza un rollback para deshacer los cambios y se muestra un mensaje de error.

2 - Archivo generador.py:

El archivo generador.py alberga el código fuente que introduce una clase esencial denominada "DatosFicticios". Esta clase desempeña un papel fundamental en la generación de datos ficticios, empleando principalmente la potente biblioteca Faker. Los datos generados por esta clase se enfocan en áreas críticas, como nombres, apellidos, alias, números de DNI y CUIT (Clave Única de Identificación Tributaria). La incorporación de esta clase y su conexión con la biblioteca Faker permiten un control preciso sobre la creación de datos simulados que, a pesar de ser ficticios, mantienen una apariencia y coherencia realistas en el contexto del conjunto de datos.

```python
from faker import Faker

class DatosFiciticios:
    def __init__(self) -> None:
        self.fake = Faker('es_ES') # Faker para español

    def generar_nombre(self, genero=None):
        if genero == 'M':
            return self.fake.first_name_male()
        elif genero == 'F':
            return self.fake.first_name_female()
        else:
            return self.fake.first_name()

    def generar_apellido(self):
        return self.fake.last_name()

    def generar_alias(self):
        return self.fake.user_name()

    def generar_dni(self, dni:int)->int:
     # Generar un número de DNI aleatorio dentro de un rango
     válido
        return str(
           self.fake.random_int(min=dni-2000, max=dni+2000))

    def generar_cuit(self, dni:int, cuit:str)->str:
        """
        Genero un nuevo cuit en base a:
        el cuit anterior y
        el dni generado

        Args:
            dni (_type_): _description_
            cuit (_type_): _description_
```

```
    Returns:
        str: _description_
    """
    if cuit in ['null', None]:
        return cuit
    else:
        # obtengo los dos primeros digitos el cuit
        first2 = cuit[:2]
        last_digit = cuit[-1]
        dni = str(dni) #convierto el dni a str
        dni = '0'+dni if len(dni)==7 else dni
        return first2+dni+last_digit
```

A continuación, se presenta una descripción de los métodos de esta clase:

1. **Constructor (__init__)**:
    - El constructor inicializa una instancia de la clase DatosFicticios.
    - Utiliza la biblioteca Faker con el idioma español ('es_ES') para generar datos ficticios en ese idioma.
2. **generar_nombre(self, genero=None)**:
    - Este método se utiliza para generar nombres ficticios.
    - Puede recibir un argumento genero que puede ser 'M' para un nombre masculino, 'F' para un nombre femenino o ninguno para un nombre genérico.
    - Utiliza la instancia de Faker para generar nombres según el género especificado o de manera genérica.
3. **generar_apellido(self)**:
    - Este método se utiliza para generar apellidos ficticios.
    - Utiliza la instancia de Faker para generar apellidos.
4. **generar_alias(self)**:
    - Este método se utiliza para generar alias ficticios, que son nombres de usuario ficticios.
    - Utiliza la instancia de Faker para generar alias.
5. **generar_dni(self, dni:int)->int**:
    - Este método se utiliza para generar números de DNI ficticios.

- Recibe un argumento dni que se utiliza como base para generar un número de DNI aleatorio dentro de un rango válido. El número de DNI generado estará dentro del rango [dni - 2000, dni + 2000].
- Devuelve el número de DNI generado como una cadena de texto.

6. **generar_cuit(self, dni:int, cuit:str)->str**:
    - Este método se utiliza para generar números de CUIT ficticios.
    - Recibe dos argumentos: dni y cuit. El dni se utiliza para generar una parte del número de CUIT, y cuit representa un CUIT anterior.
    - El método verifica si el valor de cuit es 'null' o None, en cuyo caso devuelve ese valor. De lo contrario, genera una nueva parte del número de CUIT utilizando el dni y el CUIT anterior.
    - Devuelve el nuevo número de CUIT como una cadena de texto.

3 - Archivo script.py:

El código fuente presente en el archivo script.py, desempeña un rol central en el proceso de anonimización de datos en una base de datos PostgreSQL. Este script se caracteriza por su capacidad para ejecutar una serie de operaciones secuenciales que garantizan la protección de la privacidad de los datos almacenados. En su funcionamiento, script.py importa dos clases fundamentales previamente descritas: "PostgresDatabase" y "DatosFicticios". La primera, "PostgresDatabase", se encarga de gestionar la conexión con la base de datos PostgreSQL, lo que incluye la ejecución de consultas y la obtención de información del servidor. La segunda, "DatosFicticios", juega un papel esencial al generar datos ficticios que reemplazan la información sensible, asegurando al mismo tiempo la integridad y utilidad del conjunto de datos.

```
from database import PostgresDatabase
from generador import DatosFiciticios
from columnar import columnar
from click import style
from progress.bar import ShadyBar
import time
import os
from dotenv import load_dotenv

load_dotenv()

def main():
```

```
# Tablas a anonimizar {schema_name}.{tabla_name}
# Definimos los nombres para su posterior duplicado
# Nomencaltura 'schema.table_name'
#tables = ['personas', 'fisicas']

tables = {
    'personas': [],
    'fisicas': []
}

# Creo mi instancia de datos ficticios basada wn faker
Faker = DatosFiciticios()

# Crear una instancia de la clase PostgresDatabase
db_conection = PostgresDatabase(
    host = os.environ.get('DB_HOST'),
    database = os.environ.get('DB_NAME'),
    user = os.environ.get('DB_USER'),
    password = os.environ.get('DB_PASSWORD'),
    port = os.environ.get('DB_PORT')
)

# Conectamos a la DB
db_conection.connect_to_db()

# Informacion del servidor
db_conection.print_server_info()

# Seleccionar el esquema
db_conection.select_schema(os.environ.get('DB_SCHEMA'))

# Imprimimos por pantalla las tablas originales a anonimizar
 [db_conection.print_table(table_name=tb) for tb in
 tables.keys()]

# ----Clonacion de tablas----
print('Clonando tablas...')
```

```
 [db_conection.clone_db_table(table_name=tb) for tb in
 tables.keys()]
print('\n')
# Fin clonacion de tablas----

# dataset- nombres de columnas
# data es una lista de listas
# column_names es una lista de strings
data, column_names = db_conection.get_data()

# ----variables auxiliares----
 data_print = list() # list para almacenar datos originales y
 ficticios
# fin variables auxiliares----

# Seccion generacion de datos ficticios
# y almacenamiento de los mismos en variables
print("GENERANDO DATOS FICTICIOS...")
bar = ShadyBar('PROGRESO', max = len(data))
for dat in data:
    # list para guardar los datos que voy generando
    data_row= list()

    # ----seccion datos ficticios----
    dni_fict = Faker.generar_dni(dat[3])
    cuit_fict = Faker.generar_cuit(dni_fict, dat[4])
    nombre_fict = Faker.generar_nombre(dat[2])
    apellido_fict = Faker.generar_apellido()
    alias_fict = Faker.generar_alias()
    # ----fin datos ficticios

    # seccion data_print
    # seteo los primeros valores con null
    # corresponden a persona_id per_fisica_id y sexo
    # y no generamos datos ficticios para ellos
    row_fict = [None] * 3
    row_fict.extend(
```

```
        [dni_fict, cuit_fict, nombre_fict, apellido_fict,
        alias_fict])

    # agrego a mi listado original los datos generados
        dat.extend([dni_fict, cuit_fict, nombre_fict,
        apellido_fict, alias_fict])

    # guardo en data_row mis datos generados
    # formato [nombre_columna, datos_original, dato_ficticio]
    for i, value in enumerate(column_names):
          data_row.append([value, dat[i], row_fict[i]])

    # agrego data_row al listado data_print
    data_print.append(data_row)
    # fin data_print----


    # ----Seccion datos para UPDATE ----
     # Agrego a mi listado de personas fisicas los datos
   generados
    tables['fisicas'].append(dict(
          id=dat[1], numero_documento=dni_fict,
          nombre=nombre_fict,apellido=apellido_fict,
          alias_publico=alias_fict)
      )
    # Agrego a mi listado de personas los datos generados
    tables['personas'].append(dict(
          id=dat[0], cuit=cuit_fict))
    # fin UPDATE----
    bar.next()
    time.sleep(0.05)
bar.finish()
time.sleep(1)
#-- end for

# -----Seccion print dataset actualizado-----
# Imprimimos noestro dataset orginal actualizado
```

```
    # con los valores ficticios
    column_names_fict=[
        'fake_dni', 'fake_cuit','fake_nombre',
        'fake_apellido', 'fake_alias']
    column_names.extend(column_names_fict)

    patterns = [
        ('dni_ficticio', lambda text: style(
            text, fg='white', bg='red')),
        ('cuit_ficticio', lambda text: style(
            text, fg='white', bg='red')),
        ('nombre_ficticio', lambda text: style(
            text, fg='white', bg='red')),
        ('apellido_ficticio', lambda text: style(
            text, fg='white', bg='red')),
        ('alias_ficticio', lambda text: style(
            text, fg='white', bg='red')),]
    print('DATASET DE TRABAJO ACTUALIZADO CON VALORES FICTICIOS')
    table_format = columnar(
        data, column_names, justify='c',
        head=1, patterns=patterns)
    print(table_format)
    # -----Fin seccion print dataset actualizado-----

    patterns = [
        ('DATO FICTICIO', lambda text: style(text, fg='white',
bg='red')),
    ]

    # Imprimo por pantalla data_print
    for drp in data_print:
        table_format = columnar(drp, ['CAMPO','DATO REAL',
           'DATO FICTICIO'], justify='c', head=1)
        print(table_format)

    # ----Seccion UPDATE tables en db----
    # Recorro mi diccionario de personas fisicas y personas
```

```
    for table in tables.items():
        print(f"ANONIMIZANDO DATOS DE LA TABLA: {table[0]}...")
        bar = ShadyBar('PROGRESO', max = len(table[1]))
        for pfl in table[1]:
            db_conection.anonimizate_table(
                table_name=table[0], data=pfl)
            bar.next()
            time.sleep(0.05)
        bar.finish()
        time.sleep(1)
    # ----fin seccion UPDATE tables----

    # Guardamos los cambios en la DB
    db_conection.commit_operation()

    print("Tablas anonimizadas correctamente")

    # Imprimimos por pantalla las tablas originales anonimizadas
     [db_conection.print_table(table_name=tb) for tb in
     tables.keys()]

    # Cerramos la conexión a la DB
    db_conection.close()

if __name__ == '__main__':
    main()
```

Como se puede observar, este script realiza una serie de operaciones para anonimizar datos en una base de datos PostgreSQL. Genera datos ficticios, los reemplaza en las tablas originales, actualiza la base de datos y muestra el progreso y los resultados en la consola.
A continuación se realiza una breve descripción el funcionamiento general del código:

1 - Importaciones:

El script importa varias bibliotecas y módulos necesarios para su funcionamiento. Estas importaciones incluyen:

- PostgresDatabase: Se refiere a una clase que se utiliza para interactuar con una base de datos PostgreSQL.
- DatosFicticios: Representa una clase que genera datos ficticios para anonimizar los datos en la base de datos.
- columnar: Se utiliza para formatear y mostrar datos tabulares en la consola.
- click.style: Se utiliza para aplicar estilos de texto a la salida en la consola.
- progress.bar.ShadyBar: Ofrece una barra de progreso en la consola para rastrear el progreso de las operaciones.
- time, os: Proporcionan funcionalidades relacionadas con el tiempo y el sistema operativo.
- dotenv.load_dotenv: Se usa para cargar variables de entorno desde un archivo .env.

2 - Carga de Variables de Entorno:

- Se utiliza dotenv.load_dotenv() para cargar variables de entorno desde un archivo .env, que generalmente se utiliza para almacenar configuraciones sensibles como las credenciales de la base de datos.

3 - Función main:

- La función main es el punto de entrada del programa. Aquí se realiza la mayor parte del trabajo.

4 - Definición de Tablas a Anonimizar:

- Se define un diccionario llamado tables que contiene los nombres de las tablas de la base de datos que se desean anonimizar. Cada tabla está asociada a una lista vacía, que se utilizará para almacenar datos ficticios que reemplazarán los datos reales.

5 - Creación de Instancias:

- Se crea una instancia de la clase DatosFicticios para generar datos ficticios.
- Se crea una instancia de la clase PostgresDatabase para interactuar con la base de datos PostgreSQL, utilizando las variables de entorno cargadas previamente.

6 - Conexión a la Base de Datos:

- Se establece una conexión a la base de datos utilizando la instancia de PostgresDatabase.
- Se imprime información sobre el servidor de la base de datos.

7 - Selección de Esquema:

- Se selecciona un esquema en la base de datos a través de la instancia de PostgresDatabase.

8 - Impresión de Tablas Originales:

- Se utiliza un bucle para imprimir por pantalla las tablas originales que se desean anonimizar.

9 - Clonación de Tablas:

- Se ejecuta un bucle para clonar las tablas originales y crear copias duplicadas de las mismas.

10 - Obtención de Datos Originales:

- Se obtienen los datos originales de las tablas a través de la instancia de PostgresDatabase.

11 - Generación de Datos Ficticios:

- Se inicia una sección para generar datos ficticios y reemplazar los datos originales.
- Se utiliza un bucle para recorrer los datos originales y, para cada registro, se generan datos ficticios, como DNI, CUIT, nombre, apellido y alias. Estos datos ficticios se almacenan en una lista llamada data_print.

12 - Impresión de Dataset Actualizado:

- Se imprime un dataset actualizado que incluye tanto los datos originales como los datos ficticios generados.

13 - Actualización de Tablas en la Base de Datos:

- Se actualizan las tablas de la base de datos con los datos ficticios generados.

14 - Guardado de Cambios en la Base de Datos:

- Se guarda(n) el(los) cambio(s) en la base de datos. (Comentado, puedes descomentar esta línea si es necesario).

15 - Impresión de Tablas Anonimizadas:

- Se imprime por pantalla las tablas anonimizadas.

16 - Cierre de la Conexión a la Base de Datos:

- Se cierra la conexión a la base de datos.

17 - Ejecución del Programa:

- La función main se ejecuta si el script se ejecuta como un programa independiente.

REFERENCIAS

[1] Balaji Raghunathan, “The Complete Book of Data Anonymization: From planning to Implementation”, 1st Edition, pp. 4-5, May 21, 2013.

[2] A. Majeed y S. Lee, «Anonymization Techniques for Privacy Preserving Data Publishing: A Comprehensive Survey», IEEE Access, vol. 9, pp. 8512-8545, 2021, doi: 10.1109/ACCESS.2020.3045700.

[3] Kikuchi, H., Yamaguchi, T., Hamada, K., Yamaoka, Y., Oguri, H., Sakuma, J. (2016). A Study from the Data Anonymization Competition Pwscup 2015 . In: Livraga, G., Torra, V., Aldini, A., Martinelli, F., Suri, N. (eds) Data Privacy Management and Security Assurance. DPM QASA 2016 2016. Lecture Notes in Computer Science(), vol 9963. Springer, Cham. https://doi.org/10.1007/978-3-319-47072-6_16

[4] C. Ni, L. S. Cang, P. Gope, y G. Min, «Data anonymization evaluation for big data and IoT environment», Information Sciences, vol. 605, pp. 381-392, ago. 2022, doi: 10.1016/j.ins.2022.05.040.

[5] Bazai, Sibghat Ullah, Julian Jang-Jaccard, y Hooman Alavizadeh. «Scalable, High-Performance, and Generalized Subtree Data Anonymization Approach for Apache Spark». Electronics 10, n.º 5 (enero de 2021): 589. https://doi.org/10.3390/electronics10050589.

[6] M.Ead, Waleed, y Et Al. «A General Framework Information Loss of Utility-Based Anonymization in Data Publishing». Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12, n.o 5 (11 de abril de 2021): 1450-56. https://doi.org/10.17762/turcomat.v12i5.2102.

[7] Proyectos Ágiles. «Desarrollo iterativo e incremental», 27 de septiembre de 2008. https://proyectosagiles.org/desarrollo-iterativo-incremental/.

[8] S. Murthy, A. Abu Bakar, F. Abdul Rahim, y R. Ramli, «A Comparative Study of Data Anonymization Techniques», en *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, may 2019, pp. 306-309. doi: 10.1109/BigDataSecurity-HPSC-IDS.2019.00063.

[9] R. Mendes and J. P. Vilela, "Privacy-preserving data mining: Methods metrics and applications", *IEEE Access*, vol. 5, pp. 10562-10582, 2017

[10] A. Majeed y S. Lee, «Anonymization Techniques for Privacy Preserving Data Publishing: A Comprehensive Survey», *IEEE Access*, vol. 9, pp. 8512-8545, 2021, doi: 10.1109/ACCESS.2020.3045700.

[11] L. Xu, C. Jiang, J. Wang, J. Yuan and Y. Ren, "Information security in big data: Privacy and data mining", *IEEE Access*, vol. 2, pp. 1149-1176, 2014.

[12] «Faker - The Blue Book». Accedido: 8 de octubre de 2023. [En línea]. Disponible en: https://lyz-code.github.io/blue-book/coding/python/faker/

[13] K. Klingensmith, «How to Quickly Anonymize Personal Names in Python», Medium. Accedido: 8 de octubre de 2023. [En línea]. Disponible en:

https://towardsdatascience.com/how-to-quickly-anonymize-personal-names-in-python-6e78115a125b

[14] «DBeaver Community | Free Universal Database Tool». Accedido: 8 de noviembre de 2023. [En línea]. Disponible en: https://dbeaver.io/

[15] «¿Qué es un IDE? - Explicación de los entornos de desarrollo integrado - AWS», Amazon Web Services, Inc. Accedido: 8 de noviembre de 2023. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/ide/

[16] «Welcome to Faker's documentation! — Faker 18.13.0 documentation». Accedido: 3 de noviembre de 2023. [En línea]. Disponible en: https://faker.readthedocs.io/en/master/

[17] tuxskar, «▷ Scripts en Python», El Pythonista. Accedido: 22 de octubre de 2023. [En línea]. Disponible en: https://elpythonista.com/script-en-python