UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Servicio Web para Anonimización y Pseudonimización de Datos Estructurados

David Alberto Mena Madrid

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 19 de diciembre de 2021

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Servicio Web para la Anonimización y Pseudonimización de Datos Estructurados

David Alberto Mena Madrid

Nombre del profesor, Título académico **Daniel Riofrio, PhD**

Quito, 19 de diciembre de 2021

© DERECHOS DE AUTOR

Nombres y apellidos: David Alberto Mena Madrid

Código: 00215169

Cédula de identidad: 1717914533

Lugar y fecha: Quito, 19 de diciembre de 2021

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

En la era actual, el análisis de datos constituye un área esencial dentro de las ciencias de la computación. En este campo, existe un grave problema asociado con el manejo de la privacidad individual, debido a que la identidad u otra información sensible de los usuarios son expuestas; si esta información cae en malas manos, los resultados podrían ser devastadores. En el contexto de las empresas y organizaciones, si no se maneja correctamente las políticas de información, su reputación e imagen pública podrían verse afectadas porque, cada vez, los usuarios están tomando este tema con mayor importancia. En el presente proyecto, se analizaron técnicas de anonimización y pseudonimización para la protección de la privacidad de los datos. El objetivo fue comprender los conceptos principales de cada técnica y los problemas que presentan. Además, se desarrolló un prototipo de aplicación web para probar la implementación de estas. El prototipo se lo expone como una propuesta inicial para mejorar la transferencia de información intra e inter departamental dentro la Universidad San Francisco de Quito USFQ. Finalmente, se espera que esta investigación y prototipo sean de gran utilidad para otras universidades, empresas u organizaciones en el futuro.

Palabras clave: Privacidad, Datos, Análisis, Web, Anonimización, Pseudonimización.

ABSTRACT

Nowadays, data analysis is an essential area of computer science. In this field, there is a serious problem associated with the management of individual privacy as the identity or other sensitive information of the users are exposed; if this information is in the wrong hands, the results could be devastating. In the context of companies and organizations, if information policies are not managed correctly, their reputation and public image could be affected because users are taking this issue with greater importance. In this project, anonymization and pseudonymization techniques were analyzed to protect data privacy. The objective was to understand the main concepts of each technique and the problems they present. Additionally, a web application prototype was developed to test the implementation of these techniques. The prototype is presented as an initial proposal to improve intra and inter departmental information transfer within the Universidad San Francisco de Quito USFQ. Finally, it is expected that this research and prototype will be of great use to other universities, companies, or organizations in the future.

Key words: Privacy, Data, Analytics, Web, Anonymization, Pseudonymization.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCTION

Data science is an area of computer science which has grown exponentially in the past years. In fact, according to IBM, there was a predicted estimated growth of 28% in data science jobs from 2015 to 2020 in the United States (IBM et al, 2017). The fact that there are more opening jobs in this field shows that companies are being more interested in this area to be applied for their businesses. This does not only apply to companies, but also to organizations or any other type of entities. Besides, something to consider is that the main ingredient for data science is the "data" or information that is used for analysis or for training different data models. Therefore, data science is inextricably bound to the access, quantity, and quality of information.

Meanwhile, data privacy has become a high issue among this field. Also, there are regulations on privacy that have been implemented across the planet, with General Data Protection Regulation (GPDR) being the most known. GPDR is the privacy regulation that is used for the European Union (Burgess, 2020). Taking this into consideration, it becomes an essential priority for a business or organization or any entity that needs to work with the information of individuals, to have a good privacy management. This is not only since there are regulations that exist to control privacy practices, but also because this issue has a direct impact on the entity reputation.

To overcome the privacy problem, but at the same time to be able to have valuable information from data analysis privacy methods must be used. There are two main branches within privacy for data analysis: anonymization and pseudonymization (Tokenex, 2018). These are different groups of techniques which protect the identifiable information of individuals within banks of data (Tokenex, 2018). By applying these methods correctly, the entity can disentangle itself in a greater level from privacy issues, maintain a good image relating to user

information management and at the same time be able to apply data analysis to acquire valuable insights.

Anonymization is the process by which personally identifiable information also known as (PII) is secured from a dataset or database (Rivas, 2020). This is an important area to consider for privacy reasons, the wellbeing of user information among organizations, and to have a more ethical and open data analysis (UserCentrics, 2021). Information that goes through a correct process of anonymization can lower its privacy breach and be shared with more openness. It consists of multiple techniques, which all attempt to reduce identification of a specific individual within a set of information that holds values from multiple individuals. The idea is to alter the information in a way that preserves the value for analysis, but at the same time does not expose information about specific persons (UserCentrics, 2021). In other words, when the data is analyzed, it will hold characteristics about the group of individuals but no information about a determined person within the group (Tokenex, 2018).

This is a complex process because making information not personally identifiable does not consist of eliminating attributes that directly identify an individual like the name, the identification number, a specific address, etc. The individual can be identified through other not direct attributes like information about his/her behavior, physical attributes, emotional characteristics, psychological description, money spending patterns or preferences (Record Evolution, 2020). Any information that with reverse engineering could be used to determine a particular person presents a hazard.

Additionally, there exists a high possibility that the indirect information of an individual could exist on another database, place within the internet or the physical world. There is also a chance that this indirect information could allow access to direct identifiable data about an individual. Therefore, all attributes on a dataset hold a risk of identification even if this risk is

not evident by sight. The main objective of anonymization is that no matter the supplementary information given, an individual will not be identified (Deepomatic, 2019). Therefore, a database or dataset that is correctly anonymized will hold a level of security by itself.

On the other hand, pseudonymization is another way to protect the identification of specific individuals within a database or dataset that also consists of a series of techniques. Although it has a similar name to anonymization, it is a completely different process. The main distinction lies in the fact that pseudonymization allows reidentification of an individual when granted additional information (Deepomatic, 2019). This aggregated information which can allow the identification of a particular person is guarded with high security (Maldoff, 2016). The performance of pseudonymization depends inextricably on the difficulty of access to this external database.

The aggregated information can be any attribute that allows direct identification of a person or an attribute that has a high possibility of indirectly permitting the identification of an individual. Nevertheless, as previously mentioned, any attribute even the indirect columns have a chance of allowing someone´s identification. Therefore, in pseudonymization, it is important to select correctly the public and private columns or attributes of the protected information. This is also the case for anonymization, when not all the attributes are anonymized but only a subgroup of the dataset.

There is another consideration to take into account about these two techniques. Anonymization modifies the information directly, therefore it presents a tradeoff between the value within the information and the quantity it is anonymized. This is also known as the privacy-utility tradeoff (Tabakovic, 2020). On the other hand, pseudonymization does not modify the original information, this does not guarantee that the value within information will not be reduced, since it is replacing attributes in the original dataset by pseudonym attributes

(Tokenex, 2018). Therefore, both methods present a threshold between the value of information and privacy, but in a different form.

In this work, anonymization and pseudonymization methods were explored, identifying some of the different techniques both methods present. The techniques were analyzed and applied only for structured data. This refers to information that has a defined structure, as a set of rows and columns commonly known as a dataset or an entity within a database. Other structures of information can also be anonymized and pseudonymized, but those cases are not covered here.

Furthermore, also a prototype implementing some of the methods both for anonymization and pseudonymization was developed. The prototype was designed as a web application, with high detail on its implementation. Therefore, after reading and analyzing this document, the reader will have a better understanding of privacy protection methods, their implementation and also web development.

DEVELOPMENT

Anonymization

Anonymization is in charge of modifying or generalizing information (Octopize, 2021). The idea is that it becomes increasingly harder in complexity to identify a specific individual from the data. At the same time, it must preserve the maximum amount of value within the information (structure and patterns) to be applied correctly. This is important because the quality of the data analysis depends strictly on this fact as previously mentioned.

Although, anonymization is said to be an irreversible process this is not a real fact since there is always a risk of deanonymization, which is the inverse process of anonymization (Frankenfield, 2020). Especially in the current age, with all the present sources of information available, and where the same information could be at multiple places at once. As mentioned in the introduction, if not direct information is stored on another information bank and this information gives access to direct information for identification, then this presents a hazard for the guarded information in the original information bank. Thus, anonymization must not be seen as a black box which modifies data in an irreversible way. Instead, the different techniques that implement it, must be understood well in their strengths and their weaknesses.

Anonymization Techniques

Some of the techniques for the anonymization are mentioned next. Also, visual examples and analysis are offered.

Data Masking

This technique can be applied to one or multiple columns of the dataset. The idea is to introduce random characters in part of the data in each record. The random characters can be any, such as "*" or "x", to hide part of the information (Imperva, n.d.). This technique is used

by payment institutions, to hide part of a credit card number (Complior, n.d.), a balance, or an account number, so that no one can recognize it except the owner of the information. An example of a credit card number using this technique is shown below:

Before	After
3622 3311 4320 7570	XXXX XXXX XXXX 7570

Table 1. *Data masking example*

Table 1 shows a hypothetical credit card number, safeguarded by applying a data masking anonymization technique.

Data Perturbation

It involves adding statistical noise to the dataset (New America, n.d.). The noise is aggregated on the selected columns, depending on the respective data type (Imperva, n.d.). Modifications consist of rounding numbers in the data (Imperva, n.d.), multiplying the data by a base (Imperva, n.d.), adding or subtracting a random value to each record of the data (New America, n.d.), or adding random alterations to the data (New America, n.d.). The objective of these methods is that the data is slightly modified, but at the same time the statistical properties (New America, n.d.), patterns, and relationships are preserved as much as possible. An example is shown below:

Identifier	Before			After		
ID	**Name**	**Age**	**Balance**	**Name**	**Age**	**Balance**
1	Peter	19	1.500,30	Peter	29	225,05
2	Tom	22	1.210,40	Tom	33	181,56
3	Kelly	21	940,50	Kelly	32	141,08
4	Austin	21	1.840,30	Austin	32	276,05
5	Emma	23	1.420,00	Emma	35	213,00
6	Sophie	19	1.720,80	Sophie	29	258,12
7	Kevin	20	1.770,90	Kevin	30	265,64

Table 2. *Data perturbation example*

Table 2 displays the Data perturbation anonymization technique. The columns “Age” and “Balance” were modified by data perturbation. Column “Age” was modified by multiplying the original value by 1.5 and rounding up to an integer. Column “Balance” was modified by multiplying by 0.15 and rounding up to two decimals.

Data Swapping

This method consists of interchanging the records of one or multiple columns of the dataset (Szoc, n.d.). The interchange or shuffling is done randomly (Peinoit, 2019). It is important to consider which attributes are relevant for the application of this technique (Imperva, n.d.). Swapping must be done to break any relationship on the data that could identify specific individuals.

There are multiple ways to swap the data on each of the desired columns of the dataset. The first way of data swapping is to shuffle independently each selected column (Peinoit, 2019). A table illustrating this data swapping with independent columns is presented below:

Identifier	**Before**			**After**		
ID	**Name**	**Age**	**Balance**	**Name**	**Age**	**Balance**
1	Peter	19	1.500,30	Austin	19	1.840,30
2	Tom	22	1.210,40	Kelly	22	1.770,90
3	Kelly	21	940,50	Kevin	21	1.420,00
4	Austin	21	1.840,30	Tom	21	1.500,30
5	Emma	23	1.420,00	Sophie	23	940,50
6	Sophie	19	1.720,80	Emma	19	1.720,80
7	Kevin	20	1.770,90	Peter	20	1.210,40

Table 3. *Data swapping with independent columns example*

Table 3 illustrates Data Swapping Anonymization technique is used in columns "Name" and "Balance" from the dataset. It is important to note that changes are independent in both columns.

The second way of data swapping is to swap columns in groups (Peinoit, 2019). This means that a group of columns in the dataset is shuffled, retaining the relationship between their records. An example is displayed below:

Identifier	**Before**			**After**		
ID	**Name**	**Age**	**Balance**	**Name**	**Age**	**Balance**
1	Peter	19	1.500,30	Austin	21	1.500,30
2	Tom	22	1.210,40	Sophie	19	1.210,40
3	Kelly	21	940,50	Emma	23	940,50
4	Austin	21	1.840,30	Peter	19	1.840,30
5	Emma	23	1.420,00	Kelly	21	1.420,00
6	Sophie	19	1.720,80	Kevin	20	1.720,80
7	Kevin	20	1.770,90	Tom	22	1.770,90

Table 4. *Data swapping with group columns example*

In Table 4, Data Swapping with grouped columns is portrayed. The columns that were modified are "Name" and "Age", they are shuffled vertically, maintaining the relationships of the records of those two columns originally.

The third way of data swapping is to shuffle the records within partitions (Peinoit, 2019). This means that a base column is selected, from which all the range of values define groups that the other dependent column records can be shuffled in. Each of these groups is called a partition. An example is exhibited below:

Identifier	Before			After		
ID	**Name**	**Age**	**Balance**	**Name**	**Age**	**Balance**
1	Peter	19	1.500,30	Sophie	19	1.500,30
2	Tom	22	1.210,40	Tom	22	1.210,40
3	Kelly	21	940,50	Austin	21	940,50
4	Austin	21	1.840,30	Kelly	21	1.840,30
5	Emma	23	1.420,00	Emma	23	1.420,00
6	Sophie	19	1.720,80	Peter	19	1.720,80
7	Kevin	20	1.770,90	Kevin	20	1.770,90

Table 5. *Data swapping with partitions example*

Table 5 displays Data Swapping with partitions. The base column selected is “Age”, from which two partitions are formed: “19” and “21” (because nineteen and twenty-one are the only values in the column repeated at least twice). Considering these two partitions, rows from column “Name” are shuffled within each partition. Therefore, the names Austin and Kelly are swapped for partition “21”. While the names Sophie and Peter are swapped for partition “19”.

Nulling Out

This technique is based on deleting the records in the selected columns of the dataset (Record Evolution, 2020). It is important to consider any external or internal database that could still hold the deleted information or any relationship with the information that was not removed in the dataset. If there is no external or internal database and the desired columns records are erased, this is the best way of anonymizing. The only consequence is that the information deleted is completely lost, so it becomes a completely irreversible process considering the privacy-utility tradeoff (Tabakovic, 2020). An illustration of the operation is shown below:

Identifier	Before			After		
ID	**Name**	**Age**	**Balance**	**Name**	**Age**	**Balance**
1	Peter	19	1.500,30	Peter	19	null
2	Tom	22	1.210,40	Tom	22	null
3	Kelly	21	940,50	Kelly	21	null
4	Austin	21	1.840,30	Austin	21	null
5	Emma	23	1.420,00	Emma	23	null
6	Sophie	19	1.720,80	Sophie	19	null
7	Kevin	20	1.770,90	Kevin	20	null

Table 6. *Data nulling example*

In Table 6, the column “Balance” content is completely erased, which makes the process if well-handled irreversible.

Data Generalization

The practice is based on reducing the specificity or granularity of data (Record Evolution, 2020). In this sense, it adds complexity in the attempt of the identification of a specific individual. This means that the selected column or columns for anonymization in the dataset become less specific. The data type and content of the column is important for the considerations in the process of generalizing data. An example of information in a numeric data type is shown below:

Identifier	Before			After		
ID	**Name**	**Age**	**Balance**	**Name**	**Age**	**Balance**
1	Peter	19	1.500,30	Peter	[11-20]	1.500,30
2	Tom	22	1.210,40	Tom	[21-30]	1.210,40
3	Kelly	21	940,50	Kelly	[21-30]	940,50
4	Austin	21	1.840,30	Austin	[21-30]	1.840,30
5	Emma	23	1.420,00	Emma	[21-30]	1.420,00
6	Sophie	19	1.720,80	Sophie	[11-20]	1.720,80
7	Kevin	20	1.770,90	Kevin	[11-20]	1.770,90

Table 7. *Data generalization for numeric content example*

Table 7 illustrates when generalization technique is applied in the column “Age”. The specific numeric values of the age of an individual become value ranges. This makes it harder to determine a specific individual by their age.

As mentioned previously, this method can function correctly for diverse data types. An illustration of the process for text information is demonstrated below:

Identifier	**Before**			**After**		
ID	**Name**	**Age**	**Location**	**Name**	**Age**	**Location**
1	Peter	19	Quito	Peter	19	Ecuador
2	Tom	22	Montreal	Tom	22	Canada
3	Kelly	21	Lima	Kelly	21	Peru
4	Austin	21	Toronto	Austin	21	Canada
5	Emma	23	Guayaquil	Emma	23	Ecuador
6	Sophie	19	Cusco	Sophie	19	Peru
7	Kevin	20	Zürich	Kevin	20	Switzerland

Table 8. *Data generalization for text content example*

In Table 8, the column “Location” was generalized. Therefore, the location became less specific and, instead of mentioning the city of the individual person, it relates the country. This makes it harder to identify the individual by their locality.

An example of data generalization for date information is shown below:

Identifier	**Before**			**After**		
ID	**Name**	**Age**	**Date**	**Name**	**Age**	**Date**
1	Peter	19	16/01/2018	Peter	19	2018
2	Tom	22	08/04/2013	Tom	22	2013
3	Kelly	21	23/07/2016	Kelly	21	2016
4	Austin	21	29/10/2013	Austin	21	2013
5	Emma	23	01/03/2021	Emma	23	2021
6	Sophie	19	16/12/2021	Sophie	19	2021
7	Kevin	20	11/05/2018	Kevin	20	2018

Table 9. *Data generalization for date content example*

Table 9 portrays the Generalization of the “Date” attribute. This is a small example of the principle in use, but in a big database with user data, this makes it more complex to identify a determined user by the “Date” column.

Pseudonymization

Pseudonymization is not based on adding complexity in the dataset to make it harder to identify a specific individual information. In fact, pseudonymization does not directly modify the values on the original dataset. It uses a different approach to protect the information of specific individuals (Anonyome Labs, 2020). Although, pseudonymization and anonymization have similar names and focus on a same objective, it is important to avoid confusing both terms, because they are very different.

There are many ways of implementing pseudonymization. Thus, the most important thing is not the implementation, but understanding well the core idea. The main idea, is that pseudonymization does not modify the dataset values. In contrast, what it does instead is adding pseudonyms which replace the desired sensitive columns on the original dataset (Tokenex, 2018). These pseudonyms have no value on themselves but are the keys for retrieving the sensitive information. The sensitive information is stored on another database or dataset, which also holds the relationships to their corresponding pseudonyms. This database must be highly secured since the strength of pseudonymization is directly related to the difficulty of access of the sensitive data.

There are various techniques for applying the pseudonymization concept. Two of these techniques are regular pseudonymization (Tokenex, 2018) and pseudonymization with tokenization (Hamidovic et al, 2019). Regular pseudonymization does not alter the dimension or number of columns in the dataset or database. Instead, it replaces all the values in the

sensitive columns by pseudonym values (Tokenex, 2018). These pseudonyms can represent the distribution frequency in the original column or not (Tan, 2020). Later, these pseudonyms are also saved in a different dataset or database that must be stored with high security. An example of this process preserving the original distribution is shown below.

ID	Name	Age	Balance	Account	City	Satisfaction	Secret Code
1	Peter	19	1.500,30	AE34	Quito	8	81453
2	Tom	22	1.210,40	CB62	Montreal	9	73497
3	Kelly	21	940,50	NJ28	Lima	7	12052
4	Austin	21	1.840,30	AE34	Toronto	6	63849
5	Emma	23	1.420,00	NM73	Guayaquil	8	49635
6	Sophie	19	1.720,80	NJ28	Cusco	9	57060
7	Kevin	20	1.770,90	OL09	Zürich	10	94826

Table 10. *Pseudonymization Initial Data*

Looking at Table 10, the initial data for an example of Pseudonymization is manifest. It has eight attributes which are “ID”, “Name”, “Age”, “Balance”, “Account”, “City”, “Satisfaction” and “Secret Code”.

Examining Table 10, from the eight present attributes three of them can be considered sensitive information. These are “Balance”, “Account” and “Secret Code”. According to regular pseudonymization technique, the present dataset will be divided in two datasets. One that holds the sensitive information with the relationship to the corresponding pseudonyms and one that holds the public information with the replacing pseudonyms for analysis. The public dataset with pseudonyms is shown next.

ID	Name	Age	Balance	Account	City	Satisfaction	Secret Code
1	Peter	19	AXY79	19832	Quito	8	75HNM
2	Tom	22	KXB06	29780	Montreal	9	243KL
3	Kelly	21	TLE32	13248	Lima	7	PDR79
4	Austin	21	BFW87	19832	Toronto	6	12RET
5	Emma	23	XZR32	92288	Guayaquil	8	49HFD
6	Sophie	19	JNUL4	13248	Cusco	9	985THY
7	Kevin	20	E3RF2	90933	Zürich	10	47NBE

Table 11. *Public Dataset for Regular Pseudonymization Preserving Distribution*

Table 11 shows a publicly accessible dataset generated by the regular pseudonymization technique. Contrasting with Table 10, it is evident that the columns "Balance", "Account" and "Secret Code" where replaced by pseudonyms. The distribution frequency of the columns was also preserved by the pseudonyms, as seen on the attribute "Account". Indexes 1 and 4 share the same value, as well as indexes 3 and 6. This matches the repetitions that occur for the same column on Table 10 and is a useful feature for analysis.

The private dataset generated for the regular pseudonymization with preserving frequency is illustrated below.

ID	Balance Pseudo	Account Pseudo	Secret Code Pseudo	Balance	Account	Secret Code
1	AXY79	19832	75HNM	1.500,30	AE34	81453
2	KXB06	29780	243KL	1.210,40	CB62	73497
3	TLE32	13248	PDR79	940,50	NJ28	12052
4	BFW87	19832	12RET	1.840,30	AE34	63849
5	XZR32	92288	49HFD	1.420,00	NM73	49635
6	JNUL4	13248	985THY	1.720,80	NJ28	57060
7	E3RF2	90933	47NBE	1.770,90	OL09	94826

Table 12. *Private Dataset for Regular Pseudonymization Preserving Distribution*

Table 12 shows the protected columns with their respective pseudonyms. This dataset must be protected with high security for the pseudonymization technique to work appropriately.

On the other hand, regular pseudonymization without preserving distribution frequency is the same process, only that every pseudonym is a random value without holding any relationship to the distribution in the original column (Tan, 2020). This means that for the previous example, only the pseudonyms used for the "Account" attribute on Table 11 change. The values of this attribute for index one, three, four and six will be randomized to avoid any relationship with the original column. These changes will also be reflected on the private dataset, shown in Table 12, with new pseudonyms for the "Account" attribute.

Pseudonymization with tokenization is another type of implementation for this concept. This method is also known as tokenization by itself (Tokenex, 2018). The key idea that changes is that all the sensitive attributes in the dataset are replaced by one attribute which holds a token (Hamidovic et al, 2019). This token has a relationship with all the sensitive information in one or multiple columns. The public dataset generated will have all the original columns, except for the sensitive ones. Also, it will have a new column with the token that relates with the sensitive information. Meanwhile, the private dataset generated will only have the token attribute with all the sensitive columns related to it. It is important that the token utilized holds no repeated values. An example of this type of pseudonymization is portrayed next.

The same initial information as for the regular pseudonymization is used for this example, this is found in Table 10. The public dataset generated for this initial information for the pseudonymization with tokenization algorithm is displayed below.

ID	Name	Age	City	Satisfaction	Token
1	Peter	19	Quito	8	AXN44
2	Tom	22	Montreal	9	DRE21
3	Kelly	21	Lima	7	LJK07
4	Austin	21	Toronto	6	PUI43
5	Emma	23	Guayaquil	8	BVH67
6	Sophie	19	Cusco	9	XCB72
7	Kevin	20	Zürich	10	ZET80

Table 13. *Public Dataset for Pseudonymization with Tokenization*

Table 13 illustrates the public dataset generated for the pseudonymization with tokenization. The sensitive columns “Balance”, “Account” and “Secret Code”, shown in Table 10 were replaced by the “Token” attribute.

The private dataset generated by the pseudonymization technique with tokenization is displayed next.

ID	Token	Balance	Account	Secret Code
1	AXN44	1.500,30	AE34	81453
2	DRE21	1.210,40	CB62	73497
3	LJK07	940,50	NJ28	12052
4	PUI43	1.840,30	AE34	63849
5	BVH67	1.420,00	NM73	49635
6	XCB72	1.720,80	NJ28	57060
7	ZET80	1.770,90	OL09	94826

Table 14. *Private Dataset for Regular Pseudonymization with Tokenization*

On Table 14, the protected columns with their respective token are displayed. This dataset must be protected with high security for the pseudonymization technique to work appropriately.

In Table 14, both “ID” and “Token” are attributes with no repeated values, which means they are useful for identifying all the attribute values of the sensitive information. In other words, they may serve as a key for identification. Therefore the “ID” column could have been used also as the token attribute. This is an important consideration to take in account, later in the document when the implementation for this technique is illustrated.

Finally, addressing both anonymization and pseudonymization, it is important to reiterate that the most important thing for these methods is to understand well the main concept. There is a lot of confusion on the theory for both subjects and it is easy to confuse them and their corresponding application. Sometimes the terms are used interchangeably for the same technique application, for example, with data masking. In the next section, some of the techniques for anonymization and pseudonymization methods will be applied within a web application.

Prototype

A web application was developed in order to portray the anonymization techniques previously mentioned. It is offered as a solution to improve the data sharing among departments within Universidad San Francisco de Quito USFQ in Ecuador.

The application has a server-client structure for the web. The server was built on the FastAPI Python framework, while the client was constructed on the React framework in JavaScript. The objective of the application is to allow users at the university to upload structured data (dataset) in the form of a *".parquet"* file. Parquet files are specifically suited for the management of big quantities of information. They store data more efficiently, present faster reading times, and are also very convenient for compression purposes (Databricks, n.d.). In order to convert a dataset from ".csv" format to ".parquet", the pandas library of python offers all the necessary methods with the functions "read_csv", "read_parquet", "to_csv" and "to_parquet". Thus, a file can be preprocessed and prepared in a ".csv" format, later converted to ".parquet" format locally, and lastly upload the ".parquet" converted file to the web application.

The file uploading in the application is followed by the insertion of a JSON string that specifies the anonymization techniques or pseudonymization to be applied in the different columns of the uploaded data. The *".parquet"* file with the JSON string is sent, and once the dataset is processed in the server, a new file is forwarded to download in the client. This file consists of the same input data but anonymized according to the selected techniques and columns in the JSON string. This received file is in the ".parquet" format but it's also compressed by "gzip". To be able to display it or convert it in a ".csv" readable format, the same pandas functions mentioned previously must be considered locally. Especially the

method “read_parquet” which can read a “.parquet” file directly even if it is compressed. Therefore, there is no necessity for explicit decompression of the file.

Specifically, the program covers four anonymization techniques and pseudonymization. The anonymization techniques implemented are data masking, data perturbation, data swapping and data generalization. They will be expanded and explained in further detail later in the document. On the other hand, the pseudonymization technique with tokenization was implemented also.

In further detail, the application consists of one repository containing the frontend and backend in different folders. The backend folder contains all the backend programs related to the system. In this case, there is only one backend server, but more related backend servers could be added in this section in the future if required. The frontend folder contains all the frontend programs; in this situation, there is only a web client, but the addition of other clients like a mobile or I/O device client can be introduced here.

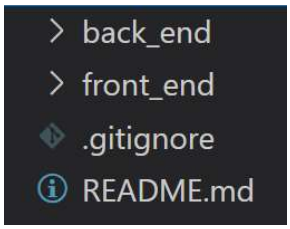


Figure 1. *Root path of the application.*

Figure 1 shows the backend and frontend structures of the system. Also, a *“README.md”* file explaining the project in full scope and a “.ignore” to mention the local files not to be uploaded.

Repository

The repository with the program can be accessed through GitHub on the following address https://github.com/Davidmenamm/Anonymization

Structure

The backend and frontend folders contain only one element that corresponds to the currently active backend server and client, as previously mentioned.

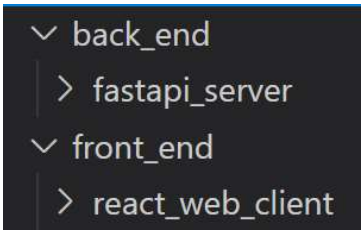


Figure 2. *Portrays the currently active participants in the client-server application*

Figure 2 portrays the active server and client for the web application. The active server corresponds to the directory “fastapi_server” inside the “back_end” folder. Whereas the active client is in the “react_web_client” directory inside the “front_end” folder.

Backend

Inside the server, the main functionality is in the “main.py” file. This element oversees the initialization and management of the base of the web server. Besides this, there is also a folder which hosts all the different endpoints logic and routing, called *“endpoints”*.

Figure 3. *The Server Architecture*

In Figure 3, the server structure is displayed. The elements shown in Figure 3, which are not the main file and the endpoints folder, represent the following: the *"README.md"* file mentions the server application description and a JSON string example, and the *"requirements.txt"* describes the necessary dependencies, in order to install the program in another computer.

In more detail, the main file manages Cross-Origin Resource Sharing (CORS), which is in charge of determining which client URLs or hosts are able to communicate with the server. Also, it oversees the annexing of all the endpoints, serves as the web server's principal endpoint and initializes the server. The main file contents are shown below (please see Figure 4).

```python
''' main file '''
''' imports '''
import uvicorn
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from endpoints import router
''' main app '''
app = FastAPI()
''' CORS '''
origins = ['http://localhost:3000']
app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=['*'],
    allow_headers=['*'],
)
''' include router '''
app.include_router(router.router)
''' root endpoint '''
@app.get('/')
async def root():
    return { 'message': 'Welcome to the Anonymization web server!'}
''' run uvicorn server '''
if __name__=='__main__':
    uvicorn.run('main:app',host='localhost',
        port=4557, reload=True, debug=True, workers=3)
```

Figure 4. *Main Application Structure*

The imported libraries are shown at the beginning of the Figure 4. Next, on the main app section, the app variable contains an object representing all the API that is being generated through FastAPI (FastAPI, n.d.-b). The generated API is built based on a standard called OpenAPI (FastAPI, n.d.-a, n.d.-b). The OpenAPI standard is a project which is in charge of defining a base structure for API's language independent (OAI, 2021).

Later, in the CORS section, the web server CORS policy is defined. This controls the cross-origin access to the web server (FastAPI, n.d.-c). In other words, all the clients that are allowed to access the web server and the configurations associated with the access they are given. The CORS definition acts as a middleware for the web server application (FastAPI, n.d.-

c). This means that before any request accesses an endpoint and before the response is returned to the client, it goes first through the CORS definition (FastAPI, n.d.-c).

Continuing, in the include router section, a router is added. This will allow the program to declare endpoints on different files. In this case, the endpoint definition of all endpoints in the project, with exception to the root endpoint, is in the *"endpoints"* folder. This is done to better organize the application, though all endpoints could be in this file; nevertheless, it's not recommended. Additionally on the root endpoint section, the root path of the server is assigned an endpoint. It only serves as a way for the client, to test the connection initially.

Following, at the end of the file, the uvicorn server is initiated. Uvicorn is a simple Asynchronous Server Gateway Interface (ASGI) for Python (Uvicorn, n.d.). ASGI, is a protocol of communication between the web server and the web application. It is the successor of Web Server Gateway Interface (WSGI), which was previously implemented universally (Manca, 2019). The main difference is that ASGI allows asynchronous communication between the parts (web server and web application) (Manca, 2019). This means that it can handle requests in a more effective manner, which can allow it to manage more clients at the same time, and faster on a long-term communication. The *"main:app"* connects the uvicorn server with the API application. The host, the port, and the allowed workers are specified on the server. The workers are the programs in charge of receiving the web server's requests; they can handle multiple threads each (Rayward, 2018). In other words, each worker can handle multiple clients isolated from other workers (Rayward, 2018). All of this together serves as the basis for the program.

On the other hand, the endpoints' structure is another important element of the program. As mentioned before, all the routing and respective logic are on the endpoints folder. This folder is explained next.

Figure 5. *Routing Structure*

Figure 5 shows the endpoints structure: routing and associated logic. There are two main subfolders. These are “data_science” and “web”. The first folder holds all endpoints related to data science, currently it only holds anonymization. The second folder contains all endpoints related to the web application. Currently no endpoints are added, but here it could be aggregated processes specifically related to the web application, like user login, search functionality or alike features, if needed in the future. Other subfolders could be included in this section, if the web application added other important areas with related necessary endpoints, on future versions.

Figure 5, there is a router file on every level within the endpoints directory and its subfolders. These routers are connected to each other in a hierarchical manner, which means that the highest router in the folder structure, connects the routers within directories inside of the same folder, and so on recursively. For example, router on the first level of the endpoints folder, at the bottom of the image, connects to the routers inside the folders “data_science” and “web”, which themselves are also in the first level of the endpoints folder. The router inside

the “data_science” directory, connects to the router inside of the “anonymization” folder, which itself is inside of the “data_science” directory.

On the other hand, the router in the first level is directly connected to the main application of the server. This can be seen on Figure 4, on the “include router” section of the file. All the routing connections presented allow the endpoints of the application to be in different files. As mentioned before, all endpoints could be located on the main application, but doing so is not recommended for scalability and better organization.

To be able to see this in more detail, the file corresponding to the router on the first level is displayed below.

```python
''' base router file'''
''' imports '''
from fastapi import APIRouter
from .data_science import router as router_data
from .web import router as router_web

''' router '''
router = APIRouter(
            prefix="/endpoints",
            tags=["endpoints"],
            responses={404: {"description": "Not found"}}
            )
router.include_router(router_data.router)
router.include_router(router_web.router)

''' default endpoint '''
@router.get("/")
async def dafault_endpoint():
    return { 'message': 'Currently, no endpoints are present for this path!'}
```

Figure 6. *Base Router for Endpoints*

Figure 6 portrays the connection between the base router (directly inside routers directory) and the routers within the folders of “data_science” and “web”.

More in detail, in Figure 6 the routers are added with the framework’s class “APIRouter” (FastAPI, n.d.-d), through the method “include_router”. When the “APIRouter” instance is created, the parameters provide information to be used by default by all routers at this level (FastAPI, n.d.-d). Prefix refers to the base path for all endpoints at this level (FastAPI,

n.d.-d). In other words, any endpoint added at this file will have the URL path "/endpoints" before any path the endpoint defines for itself. Tags is related to an interesting functionality from the FastAPI server, which allows the testing of the endpoints in the application server more directly (FastAPI, n.d.-a, n.d.-d). This is an attractive feature of FastAPI but has no relevance with the project, therefore it will not be mentioned. Responses, provide extra responses for all endpoints declared on the file (FastAPI, n.d.-d). At the bottom of the image, a GET default endpoint is added, displaying a message indicating that there are no endpoints with functionality at this level of the routing.

Continuing, on Figure 5 the router inside the folder "data_science" is displayed below.

```
1  ''' data_science router file'''
2  ''' imports '''
3  from fastapi import APIRouter
4  from .anonymization import router as anonymization_router
5
6  ''' router '''
7  router = APIRouter(
8                  prefix="/data_science",
9                  tags=["data_science"],
10                 responses={404: {"description": "Not found"}}
11                 )
12 router.include_router(anonymization_router.router)
13
14 ''' default endpoint '''
15 @router.get("/")
16 async def dafault_endpoint():
17     return { 'message': 'Currently, no endpoints are present for this path!'}
```

Figure 7. *Data Science Router*

Figure 7 presents the router within the folder "data_science" indicated on Figure 5. The connection between this router and the router inside the directory "anonymization" of the same figure is also shown. More descriptive information about the parameters of the "APIRouter" and the default endpoint are discussed below Figure 6.

Later, the router at the last level of the endpoints directory, inside the folder "anonymization" is examined below.

```python
''' imports '''
from fastapi import APIRouter, File, Form
from starlette.responses import StreamingResponse
from io import BytesIO
from .anonymize import anonymize
import pandas as pd
import timeit
import json
''' router '''
router = APIRouter(
                prefix="/anonymization",
                tags=["anonymization"],
                responses={404: {"description": "Not found"}}
                )
''' anonymization main endpoint '''
@router.post("/files", response_class = StreamingResponse)
async def anonymization(file: bytes = File(...), config: str = Form(...)):···
```

Figure 8. *Anonymization Router*

Figure 8 shows the router within the folder “anonymization” illustrated on Figure 5. Considering that on Figure 5, the “anonymization” folder has no other directories inside, with their respective router files, there are no routers connected to this file. This is the router where the anonymization process begins, specifically this holds the anonymization endpoint. The anonymization POST endpoint content is hidden, but it will be displayed and explained later in the document in the Main Endpoint section. Also, more descriptive information about the parameters of the “APIRouter” are discussed below Figure 6.

As seen on Figure 5, there are also other files at the same level than the anonymization router, shown in Figure 8. These are “techniques.py”, “mask.py”, “noise.py”, “swap.py” and “generalize.py”. These files are all in charge of applying the specific anonymization techniques and pseudonymization. They will be mentioned briefly in the section of Anonymization and Pseudonymization Implementation.

Besides, there is one more router to consider. As seen on Figure 5, this is the router inside the “web” directory. The contents of the file are shown next.

```python
''' web directory router file'''
''' imports '''
from fastapi import APIRouter

''' router '''
router = APIRouter(
                prefix="/web_external",
                tags=["web_external"],
                responses={404: {"description": "Not found"}}
                )

''' default endpoint '''
@router.get("/")
async def dafault_endpoint():
    return { 'message': 'Currently, no endpoints are present for this path!'}
```

Figure 9. *Web Directory Router*

On Figure 9, the router within the folder “web” which can be seen on Figure 5, is portrayed. Considering that on Figure 5, the “web” directory has no other directories inside, with their respective router files, there are no routers connected to this file. More descriptive information about the parameters of the “APIRouter” and the default endpoint are discussed below Figure 6.

Frontend

On the other hand, the frontend application was built with the react framework. The architecture of the client program is inside the “react_web_client” directory shown in Figure 2. This folder is expanded and explained in detail next.

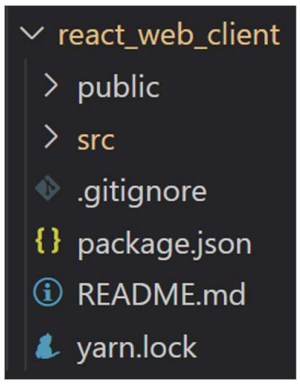


Figure 10. *Client Architecture*

Figure 10 indicates the web client architecture. It consists of two main folders called “public” and “src”. Additionally, there are system related files which are “.gitignore”, “package.json”, “README.md” and “yarn.lock”.

The “public” directory consists of static files that react needs to load without being processed by webpack (GeeksforGeeks, 2021). This includes JavaScript libraries, CSS libraries, HTML, assets and etc (GeeksforGeeks, 2021). Webpack is the static bundling module solution that react applications use (Singh, 2019).

Figure 11. *Public Folder*

Figure 11 indicates the “public” directory in the client. It contains three main files which are “index.html”, “manifest.json” and “robots.txt”, and one asset “favicon.ico”.

The “manifest.json” file includes metadata about the application, for example relevant information for the moment it is loaded on mobile phones (Abhishek Singh, 2020). Additionally, “robots.txt” file contains permissions allowed to web crawlers, spiders, and scrapers (Abhishek Singh, 2020). Also, the root HTML file is included in this directory (index.html); this is the first page to be loaded on the application (Sridhar, 2018). From the three mentioned files, the most relevant one to consider is the “index.html”, it will be expanded following.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Anonymization USFQ</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Figure 12. *Root HTML*

Figure 12 portrays the root HTML file. This is the first page loaded when the frontend program is initiated (Sridhar, 2018).

Analyzing Figure 12, the main parts to consider are detailed:

- The public URL corresponds to an environment variable that allows the access to any of the content in the “public” directory. The public URL must be used before in the path.
- The browser tab icon is located on the tag with “rel=icon”. If there’s need to update this icon, a new file must be uploaded to be used as tab icon. It must be added to the public folder with the same name as “favicon.ico” and replace the current “favicon.ico” file. Another option is to add a file with a new name and update the path at the “href” of the tag.
- At the end of the file, an important section to observe is the “div” with root id. All the react components and application will be located at this point. This connection will be mentioned later in the document, when the “index.js” file is examined.

The most relevant information, concerning this project, about the “public" directory was mentioned. Currently it is being used only to configure a specific tab icon and application name. In the future more functionality from the “public” folder could be implemented in the application.

Continuing, observing Figure 10, the next directory to consider is “src”. The folder is expanded and explained next.

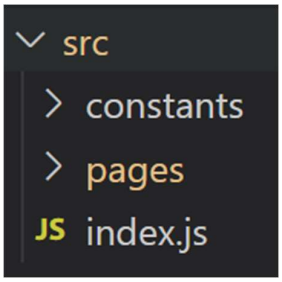


Figure 13. *Source Code*

In Figure 13, the source code section of the frontend application can be observed. There are two directories: “constants” and “pages”, and one file: “index.js”. The “constants” folder is used to store any important constant value that will be used in the application. This accounts for texts, URLS, UI styling parameters and other values alike. It is a useful feature, since these constant values can be changed in one place, which means that all files that implement any of the constant values will update immediately. The “constants” directory is opened to explore more in depth.

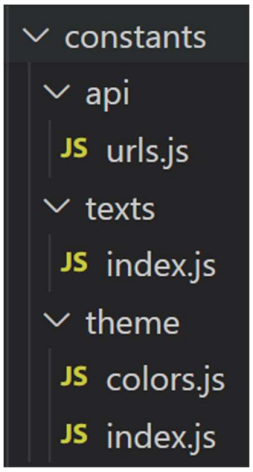


Figure 14. *Centralized Constant Values*

Figure 14 illustrates the “constants” directory. There are three main folders: “api”, “texts” and “theme”. The contents of the folders are also displayed. The “api” directory holds all the constant values related to API’s. There is only one element in the folder, which is the file “urls.js”. At this file, the URLs from all the endpoints used at the client are located. The contents are shown below.

```
/* endpoints list */

/* base url */
const baseProtocolAndHost = 'http://localhost'
const basePort = 4557
const localBaseUrl = `${baseProtocolAndHost}:${basePort}`
const baseUrl = localBaseUrl

/* endpoints */
export const urlAnonymizeData =
    `${baseUrl}/endpoints/data_science/anonymization/files`
```

Figure 15. *API URL*

At the beginning of Figure 15, the base URL is configured, the application is running locally. The second part of the file mentions the only accessed endpoint currently on the application. This is the endpoint where all the anonymization and pseudonymization processes requested by the client are initiated and given a response by the server.

Later, speaking about the “texts” folder mentioned in Figure 14. This contains only an “index.js” file, which holds all the common or reused texts of the application. The file details are displayed.

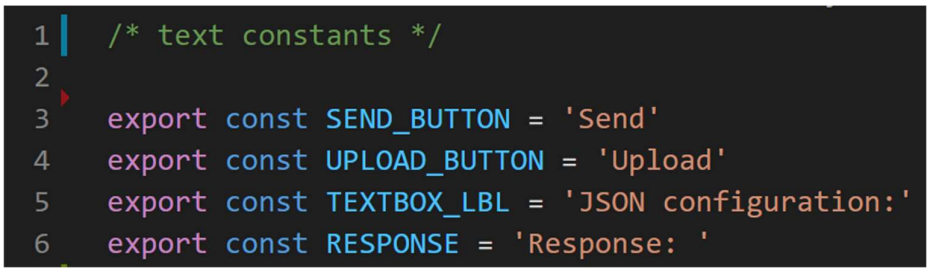


Figure 16. *Text Constants*

Figure 16 shows a very simple file for saving the text constants, currently it holds only a hand fold of values, but more could be added in the future.

Subsequently, looking at Figure 14, the last directory to consider is “theme”. It contains two files: “colors.js” and “index.js”. The role of this directory is to manage the application’s general visual theme. Both files are expanded and explained below.

```
/* colors */
export const Colors = {
    // main
    principal: '#1976d2', // blue
    secondary: '#ffe64b', // yellow
    third: '',
    fourt: '',
    // contrasts
    topMenu: '#f6f6f6', // white-grey
    background: '#d8d8d8', // grey
    dark: '#000000', // black
    white: '#FFFFFF', // white
    // event colors
    success: '#31b87b', // green
    important: '#1976d2', // blue
    error: '#b83131', // red
    warning: '#e3ac14' // yellow
  };
```

Figure 17. *Colors Palette*

Figure 17 shows the palette of colors for the theme in the frontend application. These colors can be changed for all the program from this file.

```
/* Main Theme */

/* imports */
import { createMuiTheme } from '@material-ui/core';
import { Colors } from './colors';

/* theme */
export const theme = createMuiTheme({
    palette: {
      primary: {
        main: Colors.principal,
      },
      secondary: {
        main: Colors.secondary,
      },
      contrast: {
        background: Colors.background,
        dark: Colors.dark,
        white: Colors.white,
      },
      specific: {
        topMenu: Colors.topMenu,
      },
      events: {
        success: Colors.success,
        error: Colors.error,
        warning: Colors.warning,
        important: Colors.important,
      },
    },
    styles: {
      borderRadius: '',
      border: {
        backgroundColor: 'white',
        borderColor: '#ccc',
        borderRadius: '',
        borderStyle: 'solid',
        borderWidth: '1px',
        boxShadow: '0 0.125rem 0.25rem rgba(0, 0, 0, 0.075)',
      },
    },
    overrides: {
      MuiRadio: {
        root: {
          color: Colors.principal,
          '&$checked': {
            color: Colors.principal,
          },
        },
        checked: {},
      },
    },
  });
```

Figure 18. *Theme Structure*

On Figure 18 the theme structure is illustrated. Material UI, which is a recognized library for react, on version 4 is used (Material UI, 2021). There are three main sections which are the “palette”, “styles” and “overrides”.

More extensively in Figure 18, Material UI was used to structure the theming. The three areas present are: “palette”, “styles” and “overrides”. The “palette” oversees the color management of the application; the “colors.js” is imported to use the color palette constants already defined. The “styles” manage the common styles that can be used throughout the program. The “overrides” modifies the base styling of the material UI components, which are used in the application in Figure 21 and Figure 28. Not all the theme is currently used on the application, principally only the palette is being used. Although the other areas are shown, to give an example of how to add them on the theming configuration.

Continuing, with the examination of the “src” folder shown in Figure 13. The “pages” directory is opened and described next.

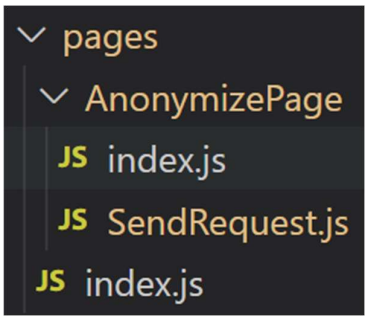


Figure 19. *Frontend Page Components*

Figure 19 portrays the Frontend Pages with their constituent components. There is one subfolder named “AnonymizePage” that holds two files: “index.js” and “SendRequest.js”. There is one independent file called “index.js”. There is only one subfolder under the “pages” directory. This is because there is only one active page currently on the application, the

anonymization page. If more pages had to be aggregated in future developments of the program, they would be added as subfolders at this section.

The anonymization page contains only two files: “index.js” and “SendRequest.js”. Both are extended and explained coming.

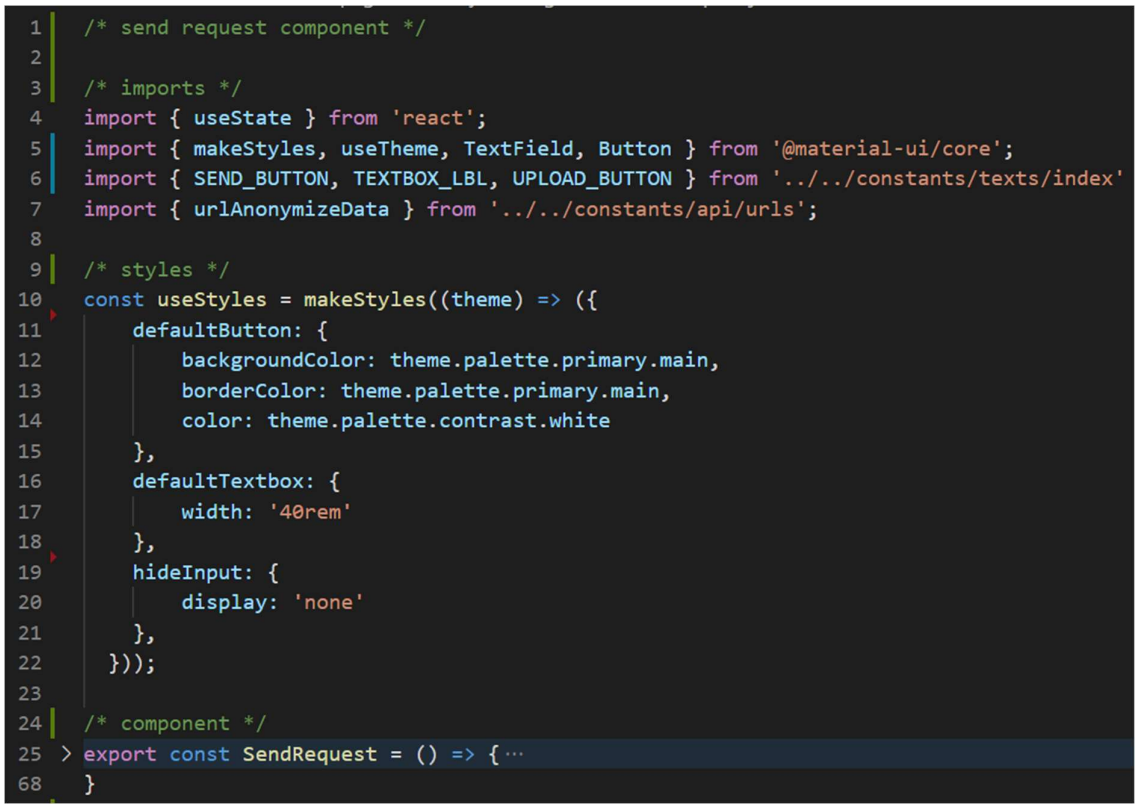
```
/* anonymization page index */

/* imports */
import {SendRequest} from './SendRequest'

/* component */
export const AnonymizePage = () => {
  return (
    <>
      <h1>{'Anonymization USFQ'}</h1>
      <SendRequest/>
    </>
  );
}
```

Figure 20, *Anonymization Page Index*

Looking at Figure 20, the base react component for the anonymization page is shown, it is called “AnonymizePage”. Additionally, the “SendRequest” react component and the title of the page are located inside the base component.



```
/* send request component */

/* imports */
import { useState } from 'react';
import { makeStyles, useTheme, TextField, Button } from '@material-ui/core';
import { SEND_BUTTON, TEXTBOX_LBL, UPLOAD_BUTTON } from '../../constants/texts/index'
import { urlAnonymizeData } from '../../constants/api/urls';

/* styles */
const useStyles = makeStyles((theme) => ({
    defaultButton: {
        backgroundColor: theme.palette.primary.main,
        borderColor: theme.palette.primary.main,
        color: theme.palette.contrast.white
    },
    defaultTextbox: {
        width: '40rem'
    },
    hideInput: {
        display: 'none'
    },
  }));

/* component */
export const SendRequest = () => {···
}
```

Figure 21. *Anonymization Page Send Request*

In Figure 21, the “SendRequest” react component is illustrated. It has two main sections portrayed in the file which are the styling and the component itself. The contents of the component “SendRequest” are hidden and will be displayed and explained later in the document.

Observing Figure 21 more in detail. Initially, all the imports related to react, material UI, text constants and URL constants are imported. Posterior, on the styling section, the CSS is defined on the same file as the component and in plain JavaScript. This is done with the aid of a useful functionality from material UI called “makeStyles”. Additionally, one more thing to note is that “makeStyles” receives a parameter, which is called “theme”. This parameter represents the theme itself declared on Figure 18. This is the way material UI uses to allow access of the theme by any specific react component.

Later, at the middle of the file, a react component is generated. This corresponds to the "SendRequest" component, which is the main component where the anonymization process starts. It holds a Form where the configuration string and the file to be anonymized or pseudonymized is included. The component is closed at this moment, since this section is mainly focused on the structure of the web application for the frontend, but it will be analyzed in a coming section.

Next, there is one more file to consider from Figure 19. This is the "index.js" which does not belong to a subfolder of the "pages" directory but it is directly inside the "pages" folder. This file contains the main react page of the application, in other words the page that will be shown when the program initiates. Exactly, it is not the first page loaded since the "index.html" mentioned previously on Figure 12 and also the contents of the main "index.js" file inside the "src" directory (Figure 23), launch earlier. Usually this first two pages do not show any relevant visual UI but are in charge of functionality. Therefore, any component added inside the "MainPage" should be displayed at the beginning of the react application.

```
1  /* pages index*/
2
3  /* imports */
4  import {AnonymizePage} from './AnonymizePage/index'
5
6  /* main page component */
7  export const MainPage = () => {
8      return (
9          <AnonymizePage/>
10     );
11 }
```

Figure 22. *Main Page Index*

Figure 22 shows the main react UI page. It contains one react component called "MainPage", which currently holds only the "AnonymizePage" mentioned inside Figure 20.

Additionally, it is important to reiterate that this file is the “index.js” directly located inside of the “pages” folder, as shown on Figure 19, to avoid confusion of index files.

Continuing, observing Figure 13 that portrays the content inside the “src” folder. There is one more index file to consider, this is the “index.js” file directly inside the “src” directory and not in any of the subfolders. This file will be expanded and described.

```
/* src index */

/* imports */
import React from 'react';
import ReactDOM from 'react-dom';
import {MainPage} from './pages/index';
import { MuiThemeProvider } from '@material-ui/core';
import { theme } from './constants/theme';


/* render react components */
ReactDOM.render(
  <React.StrictMode>
      <MuiThemeProvider theme={theme}>
        <MainPage/>
      </MuiThemeProvider>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Figure 23. *Main React Rendering File Index*

Figure 23 displays the main react source code file index. All the react components will be located inside the “ReactDOM.render”. Currently, there are three main containers inside this section. These are: “React.StrictMode”, “MuiThemeProvider” and “MainPage”.

Explaining Figure 23 in more detail, this file is directly connected with the “index.html” file, presented earlier. In Figure 12 the “index.html” can be seen, it is important to observe the “div” element with an id of “root”. Inside this “div” element, all the contents rendered with “ReactDOM.render” in Figure 23, will be located. The “React.StrictMode” component

corresponds to a react configuration, which gives more strict coding rules, it can be removed but it is usually useful to maintain better code practices. The “MuiThemeProvider” is a content provider, that Material UI uses, to pass the theme created in Figure 18, to be accessed by any component on the react application. The “MainPage” corresponds to the main page component of the application, referenced on Figure 22.

Finally, observing Figure 10, the last files to be mentioned of the frontend application are: “.gitignore”, “package.json”, “README.md” and “yarn.lock”. The first is a file to avoid uploading specific files or types of files to the online repository, noted in the “Repository” section. The second file serves to handle the dependencies for JavaScript and other configurations. The third file holds a description about the frontend application. Lastly, the fourth file is a system specific file that yarn package manager generates, but it is not considered in the scope of the project.

Configuration and Execution of the Program

To configure and execute the program locally, there are some conditions that must be accounted. If all the mentioned factors are followed up correctly, the program should run with ease in the local machine. The requisites to run the application are mentioned below.

Knowledge

- An intermediate knowledge both in JavaScript and Python programming languages is necessary.
- A proficient knowledge of the React framework for the frontend (with hook components) and the FastAPI framework for the backend is required.
- A minimum knowledge of NodeJS to download it in the LTS version, which comes with the default NPM package manager.

- The knowledge of dependency managers is important both for the frontend and backend of the application.
    - For the Frontend, the package manager is Yarn, which is installed through NPM once the LTS version of NodeJS is installed.
    - In the backend, the package manager can be “pip”, “anaconda” or any dependency manager that allows to add the required libraries mentioned in the “requirements.txt” file shown in Figure 3.

Considerations

- The backend must be executed first by running the “main.py” application mentioned on Figure 4.
    - This will launch the “uvicorn” server locally and activate the backend application on “localhost” with the port 4557.
- Later the frontend must be run by executing the “yarn start” command in the shell of choice. This means PowerShell, command prompt, terminal or etc.
    - This will prompt a new window on the default browser on “localhost” and the port 3000.
- The connection between the two applications is already set up, and there is no need for extra configuration.

Once all these conditions are followed up in the local computer, the application should launch correctly and display as it is shown below.

The shell mentions the following message once the backend is started adequately.

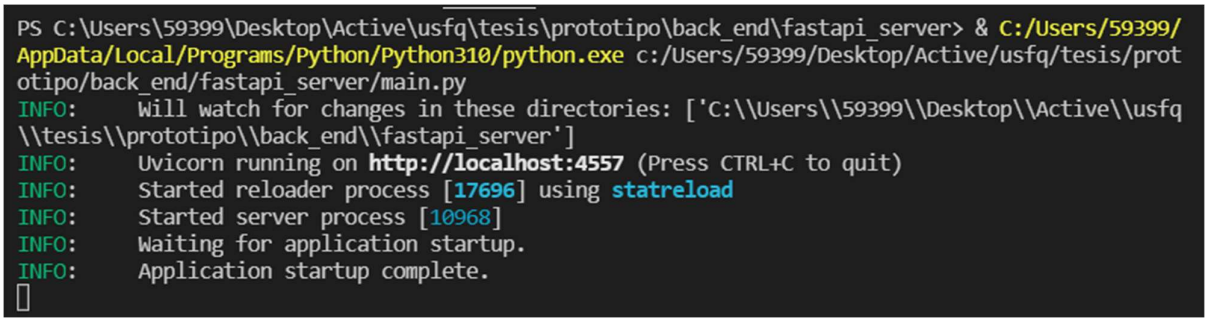


Figure 24. *Backend Execution*

Figure 24 shows the execution of the backend application. These are the messages that are displayed on the shell once the program is runed correctly. The path shown in the image is of no importance since this will change when executed on a different computer.

On the other hand, the shell displays the following text once the frontend is launched correctly.

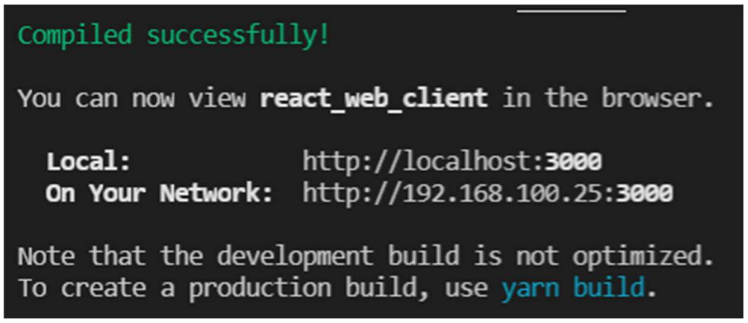


Figure 25. *Frontend Execution*

Figure 25 shows the frontend application execution when run successfully. The path shown in the image is of no importance since this will change when executed on a different computer.

Finally, once the backend is launched and later the frontend is launched, the following window will prompt in the default browser. This is the only present window of the application and were all the anonymization process occurs.

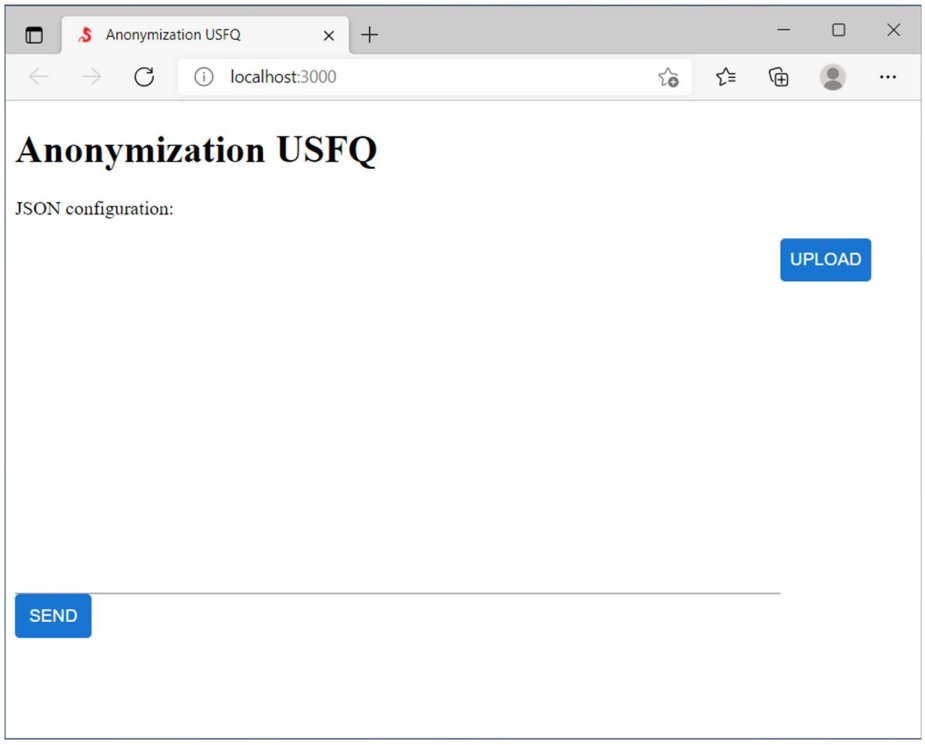


Figure 26. *Web Application Main Page*

In Figure 26 the web application main screen is shown. This is the only page currently on the application and where the main anonymization process is executed.

Main Endpoint

As shown earlier in Figure 8, the anonymization endpoint is located within the anonymization router. This is the main endpoint of the application and is in charge of initiating the anonymization process. It is also the only active endpoint currently on the application. Its contents are expanded and explained below.

```python
''' anonymization main endpoint '''
@router.post("/files", response_class = StreamingResponse)
async def anonymization(file: bytes = File(...), config: str = Form(...)):
    ''' initiate timer '''
    tic = timeit.default_timer()

    ''' input in-memory file to dataframe'''
    inMemoryFile = BytesIO(file)
    df = pd.read_parquet(inMemoryFile)
    inMemoryFile.close()

    ''' json configuration and apply techniques '''
    config_obj = json.loads(config)
    results_df = techniques(df, config_obj)

    ''' output in-memory file to streaming response '''
    outMemoryFile = BytesIO()
    results_df.to_parquet(
        outMemoryFile, index=True, compression='gzip')
    response = StreamingResponse(
        iter([outMemoryFile.getvalue()]),
        media_type='application/gzip',
        headers={
        'Content-Disposition': 'attachment; filename=dataset.parquet.gzip',
        'Access-Control-Expose-Headers': 'Content-Disposition'
        }
    )
    outMemoryFile.close()

    ''' end timer '''
    elapsed = timeit.default_timer() - tic
    print( f'Time elapsed is aproximately {elapsed/60} minutes.' )

    ''' return streaming response'''
    return response
```

Figure 27. *Anonymization Endpoint*

Figure 27 shows the anonymization endpoint. This is the main endpoint of the application. It receives two arguments which are a file and a string and returns a response as a stream of data. This allows the client to download the anonymized file. Besides this, its code has three main sections “input in-memory file to dataframe”, “json configuration and apply techniques” and “output in-memory file to streaming response”. Also, it has a section to take the time the endpoint takes to respond to the client.

Analyzing Figure 27 more in detail, the anonymization endpoint is a POST endpoint. It receives two parameters which are the input file and the configuration string, “file” and

“config” respectively. The default format used by “FastAPI” framework for file transfer is to encode the body in a multipart format, known as “multipart/form-data” (FastAPI, n.d.-e). This is an important consideration to take in account when sending the information from the client. This will be seen in a moment.

Inside the endpoint, the first section, “input in-memory file to dataframe”, stores the input file in program memory and later proceeds to place it on a DataFrame object. The second part, “json configuration and apply techniques”, loads the JSON configuration string and sends the DataFrame object to be applied the corresponding techniques of anonymization or pseudonymization mentioned in the JSON. The techniques applied will be explained later in the document. The third part, “output in-memory file to streaming response” generates a file in program memory with the results of the anonymization, this is a “.gzip” compressed file. It sends this file back to the client for download through a streaming response. Within all this steps something important to consider is that, for security reasons, the input file is not allowed to enter the disk memory but runs completely on the program memory. This is done through Python API “BytesIO”, which allows program memory byte files (Python, 2021).

Lastly, to be able to better visualize this process, the frontend function responsible for calling the main endpoint in the server will be displayed. This is the function that was hidden when explaining Figure 21 previously on the document. This file contains a react component with JSX code which is the way of writing applications in the react framework. The component is very simple and only displays a page with a form, as shown in Figure 26. The main part to consider from the code is the “enctype” mentioned in the form. This must be set to the “multipart/form-data” as previously noted on this section.

```
/* component */
export const SendRequest = () => {
    // hooks
    const theme = useTheme()
    const classes = useStyles(theme);
    const [text, setText] = useState('')
    // return
    return (
        <>
            {/* form */}
            <form action={urlAnonymizeData}
                enctype="multipart/form-data"
                method="post">
                <div>
                    {/* textfield */}
                    <p>{TEXTBOX_LBL}</p>
                    <TextField name='config'
                        className={classes.defaultTextbox}
                        multiline={true}
                        minRows={15}
                        value={text}
                        onChange={e => setText(e.target.value)}/>
                    {/* upload button */}
                    <Button className={classes.defaultButton}>
                        <label>
                            <input name='file'
                                type="file"
                                accept='.parquet'
                                className={classes.hideInput}/>
                            {UPLOAD_BUTTON}
                        </label>
                    </Button>
                </div>
                <div>
                    {/* send button */}
                    <Button name='sendButton'
                        type='submit'
                        className={classes.defaultButton}>
                            {SEND_BUTTON}
                    </Button>
                </div>
            </form>
        </>
    );
}
```

Figure 28. *Anonymization Endpoint Call*

On Figure 28 the anonymization endpoint is called from the client. The code displays a the react component that makes the calling. The contents can be omitted, but the important part to consider is the “enctype” mentioned in line 35. This is the “multipart/form-data” format, which the form needs to specify in order to send files to the FastAPI server.

Techniques

The application contains four anonymization techniques and the pseudonymization technique with tokenization. The anonymization techniques used are data masking, data perturbation, data swapping and data generalization. Data masking was implemented to work on numeric, text or date domains; basically, any data type that can be translated to a string value works. Data perturbation was developed to be used only on numeric input; therefore, it functions adequately with integer or decimal values. Data swapping doesn't consider the domain type; therefore, it allows any type of data. Additionally, data swapping was implemented for independent columns and grouped columns, as displayed in Table 3 and Table 4; columns with partitions were not developed. Lastly, data generalization is configured to function with numeric, text or date domains, but no other than these.

More in detail, the data masking implementation developed has some configurable hyperparameters. These are the "direction", the "quantity" and the "symbol". The "direction" has only two options, "right" and "left". In the first option, the mask is applied from the right (as shown in Table 1), whereas in the second option, the mask is placed from the left. Regarding the "quantity", it refers to the length of the employed mask. It can have determined units of length or a percentage length. Respecting the "symbol", this addresses the character used for the applied mask. Besides these parameters, also the list of columns on which to apply data masking must be provided. The following table denotes this type of anonymization.

Parameter	Description	Example	Default
Direction	The orientation of the mask	"right" or "left"	"right"
Quantity	The size of the mask: integer number or float percentage	3 or 0.20 or etc	1
Symbol	The character for the mask	"*", "#", "-" or etc	"*"
Columns	Columns on which to apply the technique	["column1", "column4", "column7"]	required

Table 15. *Data masking parameters*

Table 15 illustrates the parameters for the implemented data anonymization technique. It contains the parameter, its description, an example, and the default value or required in case there is no default value.

On other hand, data perturbation also has some configurable conditions, which are: “operator”, “quantity”, “rounding”, “rand” and “seed”. The “operator” covers the type of operator that will be used to add statistical noise to the data. As mentioned on the “data perturbation” section, this technique consists of adding statistical noise to the information. The “quantity” represents the base number that will be applied with the operator. In case the “rand” argument is habilitated, this means a range of random numbers, from 0 to the value of “quantity”, will be applied with the operator. For example, on Table 2, the “operator” used was “*” and the “quantity” 1.5 and 0.15. Continuing, the “rounding” indicates the number of decimals to be used for rounding, after the noise is aggregated. It rounds to the nearest decimal. The “rand” allows random numbers to be used, instead of a constant value. This allows the generation of more random noise but must be used carefully and lightly so that it doesn’t distort data too much, making it lose too much value. The “seed” is only applicable when “rand” is habilitated, it represents the pseudorandom seed for reproducibility. Apart from the mentioned arguments, also the “columns” parameter is necessary. A table is illustrated below.

Parameter	**Description**	**Example**	**Default**
Operator	Operator to use for the noise	"+", "-", "*" or "/"	"*"
Quantity	The base number to use for the noise	5, 12, 23 or etc	2
Rounding	Number of decimals to round	1, 2, 3, 4 or etc	2
Rand	If random numbers in range, rather than a constant value	true or false	false
Seed	The initialization for the pseudorandom numbers	0, 112, 1140 or etc	0
Columns	Columns on which to apply the technique.	["column1", "column4", "column7"]	required

Table 16. *Data perturbation parameters*

Table 16 portrays the parameters for data perturbation or statistical noise anonymization technique. It contains the parameter, its description, an example, and the default value or required in case there is no default value. For the rounding parameter, if the number 0 is selected, the number is rounded and casted to the nearest integer.

Subsequently, mentioning data swapping, it holds two parameters for tuning. The first one is the "type" of grouping. As noted at the beginning of this section, only two types of grouping are being used: independent and group columns. The second one is the "seed". Data swapping shuffles records randomly within the column or group of columns. Therefore, a seed for the reproducibility of the randomness is necessary. In top of that, as in the other cases it also has a "columns" argument. The respective table is shown below.

Parameter	**Description**	**Example**	**Default**
Type	Type of column grouping	"independent" or "group"	"independent"
Seed	The initialization for the pseudorandom shuffling	0, 112, 1140 or etc	0
Columns	Columns on which to apply the technique.	["column1", "column4", "column7"]	required

Table 17. *Data swapping parameters*

Looking at Table 17, the arguments for data swapping anonymization technique. It contains the parameter, its description, an example, and the default value or required in case there is no default value.

Regarding data generalization, there are three main parameters that can be modified for the implementation, these are "num_range", "str_level" and "date_type". The "num_range" applies only for numeric domains, this is the interval length to which the technique generalizes

numbers. For example, if the number was 97 and the “num_range” was 10, the result after applying generalization would be “90.0 – 100.0”. The “str_level” is used only for columns of string values, this corresponds to the number of levels that will be returned on the string. If a particular string does not contain enough levels as required by this argument, then it is raised to the maximum level possible. Such as, a string containing the hierarchical categories for books in a specific library. If the string was “History World War WW2” and the “str_level” was two, then the string would be generalized by two levels leaving it on “History World”. It is important to preprocess the dataset to arrange the string values in the column to be hierarchical or generalizable in this format, otherwise the generalization of strings does not work.

There are also three secondary parameters for generalization, these are the “date_type”, “day_first” and “year_first”. The “date_type” corresponds to the format dates are generalized. In other words, if specific dates are generalized leaving only the month or leaving only the year or both. Also, dates can be generalized on five and ten year intervals. The other secondary arguments will be explained later. Also, as the preceding techniques the “columns” argument is included.

Parameter	Description	Example	Default
num_range	The interval range for numeric generalization	2, 10, 50, 100 or etc	10
str_level	The levels to return on a string for generalization	1, 2, 3, 4 or etc	1
date_type	The format for date generalization	"month", "year", "month-year", "five" or "ten"	"year"
day_first	Interpret the first element on a date as the day and second as the month. Only when the “year_first” parameter is not defined. Otherwise, it defines the order of the day and the month. The month will come after or before the day, depending on this parameter.	true or false	true

year_first	Interpret the first element on a date as a year. The month and day will come after in the respective order corresponding to the "day_first" parameter.	true or false	false
columns	Columns on which to apply the technique	["column1", "column4", "column7"]	required

Table 18. *Data generalization parameters*

In Table 18, the variables for data swapping anonymization technique are exhibited. It contains the parameter, its description, an example, and the default value or required in case there is no default value.

Additionally, mentioning data generalization, for the date domains it is important to consider the date format used. Currently, the “DD-MM-YYYY”, “MM-DD-YYYY”, “YYYY-DD-MM” and “YYYY-MM-DD” are the allowed formats for the application. These formats are specified based on the arguments “day_first” and “year_first” mentioned on Table 18. Although, behind the scenes, the date parsing is done with the method parser of the “dateutil” python library (Dateutil, n.d.), which can handle most known date formats. The user can attempt distinct formats to the expected formats and see how the application results.

Regarding the pseudonymization technique with tokenization, it consists of generating a new dataset that does not contain the private columns, that do not want to be displayed, but maintains a relation to the old dataset by a unique identifier. Therefore, the values of the private columns can be identified by tracing the unique identifier between the two datasets. It contains two main arguments which are “unique_index” and “private_columns”. The first argument refers to the unique identifier of the dataset which must be unique to work. The second argument refers to the private columns that will be removed from the new generated dataset. The idea, as previously mentioned, is that the new generated dataset can be publicly accessed, and the previous dataset, that holds the private columns, must be kept with high security.

Parameter	Description	Example	Default
unique_index	The unrepeated identifier. It can be a specific column or a list of columns that together identify each record. This must not interfere with the columns mentioned in the "private_columns" argument	["column_id"] or ["column_id1", "column_id2", "column_id3"]	required
private_columns	The sensitive column or list of columns to remove	["column2"] or ["column2", "column5", "column6"]	required

Table 19. *Pseudonymization parameters*

In Table 19, the arguments for pseudonymization with tokenization are displayed. It contains the parameter, its description, an example, and the default value or required in case there is no default value.

To pass the different anonymization techniques and pseudonymization simultaneously and on different columns of the dataset, a general structure is utilized. It has only one argument called "techniques", which is a dictionary that holds the different anonymization techniques or pseudonymization being implemented. The techniques inside this dictionary are themselves arrays of objects. The objects represent a configuration for the specific technique. This means that each technique can have multiple configurations, this is important for when the same technique wants to be applied with different parameters for different columns. The configurations as mentioned are objects, they hold the arguments defined on the preceding tables of this section. The following table portrays these considerations.

Parameter	Description	Example	Default

Techniques	A dictionary that holds the different techniques to be applied. Each technique inside is an array of objects itself. The objects represent the specific configuration for a particular technique.	{ "mask": [{…},{…}], "noise": [{…},{…},{…}], "swap": [{…}], "generalize": [{…},{…},{…}], "pseudonymization": [{…},{…}] }	required

Table 20. *General structure parameters*

Table 20 displays the general structure for passing anonymization techniques or pseudonymization with tokenization to the application. It contains the parameter, its description, an example, and the default value or required in case there is no default value. For the value in the example column of the table, it is important to consider that this is a union of different data structures. The elements "{…}" denote the specific configuration to be used, they were not expanded. The configuration consists of the arguments mentioned on the previous tables of this section. In other words, every "{…}" will have the specific variables according to the technique that is being used. As an arbitrary example, the "{…}" for data masking would be: {"columns": ["name"], direction": "right", "quantity": 4, "symbol": "*"}.

Finally, it is important to consider that all the tables mentioned on this section are implemented in practice on JSON strings. This means that all the general structure mentioned on Table 20 and the rest of the tables mentioned in this section, are all introduced in the application using a JSON format. This will be examined in more detail on the "JSON Configuration" section, later in the document.

Preprocessing

The preprocessing of information is very important on any data science analysis. As mentioned previously, anonymization and pseudonymization are important areas to preserve the privacy of information within data science. Preprocessing can become a complex process

depending on the initial data. Also, it can be very specific to the type of data utilized. The program as it is now, does not include a preprocessing unit since that is a very broad area and will make the program much more complex. Therefore, the preprocessing of the input data is left as a complete responsibility for the user to apply, before submitting the file to the program.

There are important considerations to take into account in the dataset preprocessing. The columns must have only one domain type, and not mixed domains. Also, every column must have the correct format as illustrated on the Techniques section. Finally, there must be no empty values, this means that the dataset must be treated correctly with techniques for filling values, e.g., the mean of the column for numeric values.

JSON Configuration

The JSON configuration follows the structure mentioned in the "Techniques" section. It is inputted as a string to the program in the text area shown in the main page of the application on Figure 26. An example of this JSON string is shown below, it is important to note that the JSON string does not require to pass all the techniques at the same time as the one shown in the example. The only thing to consider is that at least one technique or more must be introduced to the program.

```
{
    "techniques" : {
        "mask" : [
            {
                "columns": ["columnA"],
                "direction": "right",
                "quantity": 4,
                "symbol": "*"
            }
        ],
        "noise" : [
            {
                "columns": ["columnB"],
                "operator": "+",
                "quantity": 50,
                "rounding": 2,
                "rand": false,
                "seed": 30
            }
        ],
        "swap" : [
            {
                "columns": ["columnC"],
                "type": "independent",
                "seed": 30
            }
        ],
        "generalize" : [
            {
                "columns": ["columnD"],
                "num_range": 10,
            },
            {
                "columns": ["columnE"],
                "str_level": 2,
            },
            {
                "columns": ["columnF"],
                "date_type": "year",
                "day_first": true,
                "year_first": false
            }
        ],
        "pseudonymization" : [
            {
                "unique_index": ["columnID"],
                "private_columns": ["columnG"]
            }
        ]
    }
}
```

Figure 29. *JSON Configuration Example*

Figure 29 addresses an example of the JSON configuration. It is important to note as mentioned precedingly, that not all the techniques must be mentioned at the same time as in the example. At least one or more techniques must be added, all depends on the requirements of the user for the specific dataset that is utilized. Additionally, every technique holds an array, therefore it can have more than one configuration of parameters. This is useful when configuring the same technique for different columns with distinct conditions. Besides, it is not recommended to share columns among the different configurations within a same technique or within different techniques. Therefore, in the image they are not shared but in case sharing is done, the techniques will be applied sequentially in the order they are declared on the JSON string, from top to bottom. Also, errors could occur in the application when attempting to share columns. Lastly, all the parameters for the techniques seen in the JSON string are explained in the "Techniques" section and the column names shown in the image are hypothetical, so they must be replaced by the specific ones in the dataset used.

This is how the main page of the application will look like when introduced the JSON string.

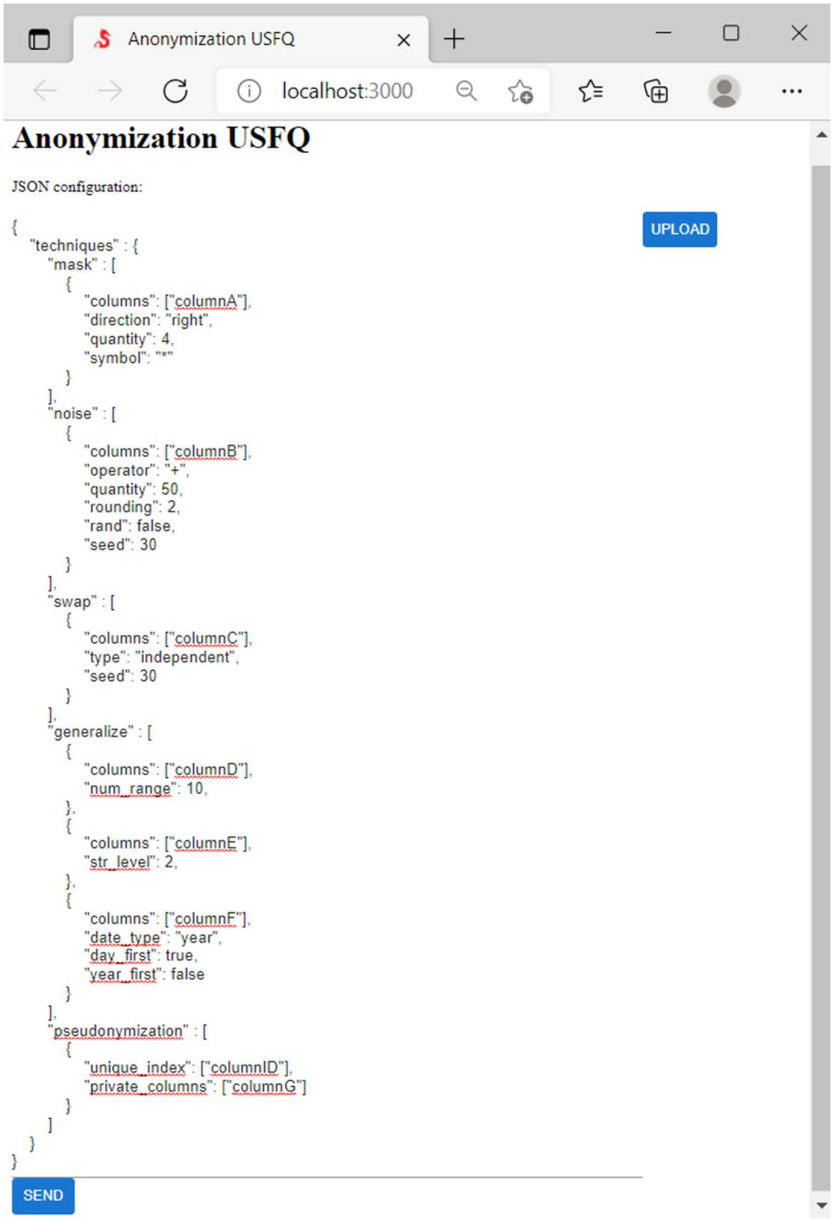


Figure 30. Web Application Main Page with JSON

Figure 30 shows the main page of the web application with the JSON configuration string. Before the send button is clicked, the ".parquet" file must be added through the upload button. Once the file is added, if it is correctly preprocessed and the JSON string is adequately configured, the download will be loaded in a moment, depending on the size of the input file.

Anonymization and Pseudonymization Implementation

The anonymization techniques and pseudonymization with tokenization technique was specifically implemented on the following files: “techniques.py”, “mask.py”, “noise.py”, “swap.py” and “generalize.py” as seen on Figure 5.

The techniques file receives the dataset that will be modified with the techniques of privacy protection according to the JSON configuration. This is seen in Figure 27, the techniques method, from the techniques file, is called to apply the privacy techniques to the dataset and return the result to the main endpoint. Additionally, it also is the file that releases the pseudonymization with tokenization.

The rest of the files mentioned are each responsible for a specific anonymization technique: “mask.py” file refers to data masking, “noise.py” signifies data perturbation technique (which adds statistical noise), “swap” addresses data swapping and “generalize.py” implements data generalization. All these techniques were explained in detail in the theory previously in the document. The pseudonymization with tokenization is the only technique that does not have its own file, since this is a very simple technique in practice, thus it was added directly in the “techniques” file. The contents of all these files can be revised if desired by the user but all work accordingly and produce the desired result when the input dataset is correctly preprocessed and the JSON configuration string is adequately build.

Errors

In case there is an error in the JSON string or if the input dataset was incorrectly preprocessed, there will appear an error on the screen once the send button shown in Figure 30 is clicked. To try again the web page must be reloaded in “localhost:3000” as mentioned in the Configuration and Execution section of the Program.

There is no validation of the JSON string or the input dataset in the program. Therefore, the input dataset and configuration string must be entered adequately or else an error will display. Additionally, the server must be examined to see if it is still running, if this not the case it must be relaunched before the client is reloaded. An example of the error page is illustrated next.

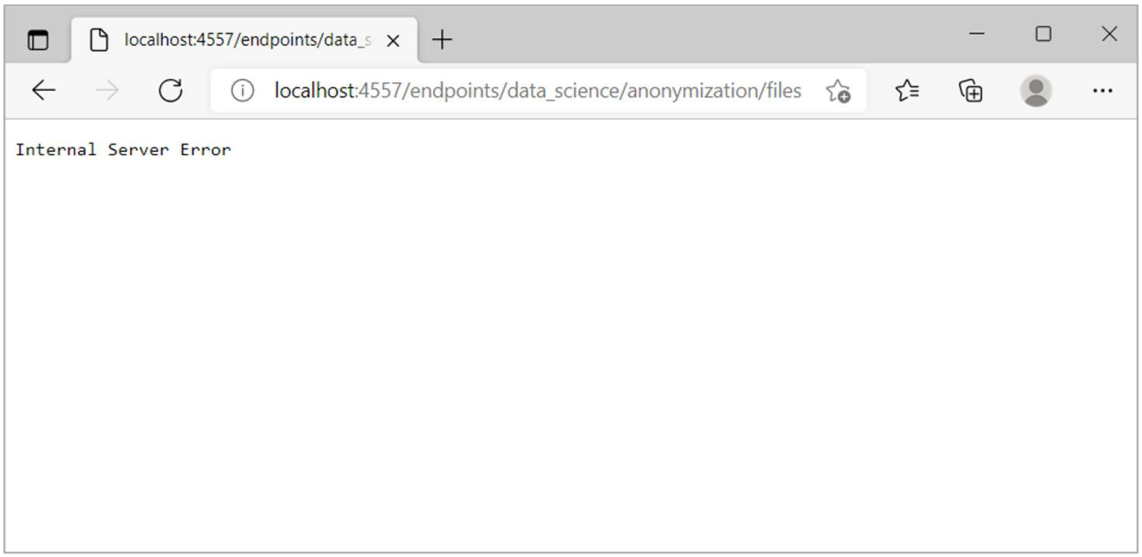


Figure 31. Error Page

Figure 31 shows the error page of the application. It is important to note that the URL is changed from "localhost:3000" portrayed at Figure 26 to "localhost:4557/endpoints/data_science/anonymization/files". This makes it necessary to reload at "localhost:3000" to try again. Also, an error message with the text "Internal Server Error" displays on the screen. This is the error message that appears for Microsoft Edge web browser, a different error message and UI will probably appear for other browsers, but the idea is the same.

CONCLUSIONS

Data science is an essential field for the development of businesses, organizations, or any entity on today's society's computer systems. Privacy is a recurring issue that is strongly related to this field as it requires the access of information as its base and most of the time this information is related to specific individuals. Anonymization and Pseudonymization are important methods to consider to prevent privacy problems, manage better user privacy ethics, and continue doing data analysis. In this document, both methods were studied and analyzed. Also, a prototype of a web application implementing techniques for both methods was released.

An important concept present in these two methods of privacy protection is the threshold that occurs between the value of data (patterns and structure) and the privacy it holds. No matter how good anonymization or pseudonymization is done, there will always be some value lost in the original data. Therefore, it is extremely important to spend the necessary time finding the adequate threshold, to correctly apply any of these two methods. This is an arduous job since there is no determined way to know exactly the amount of privacy and utility of information. As previously mentioned, privacy is influenced by the amount of information accessible from other sources. This means that all the information in the internet and in the physical world present a hazard, since there is the possibility that information that can reveal identities from the original source is present somewhere. With respect to utility is also difficult to determine the amount of utility, since this can change depending on the model used for the data analysis. Besides, it is important to note that both methods have different ways of removing the value in data. Anonymization removes value in data by modifying values directly, whereas pseudonymization by replacing sensitive columns with pseudonyms that can be or not equally distributed or removing sensitive columns.

The idea of the application is that it will be hosted and utilized by Universidad San Francisco de Quito USFQ in Ecuador to better manage data transferring among all departments. There is an important concern to take into account when hosting the application on a server for it to be accessible online. Due to privacy concerns, all the process of the application is done in program memory as Figure 27 portrays. The anonymization or pseudonymization of the dataset files can also occur at the gigabyte level, which adds an enormous weight to the active program memory. If this was done for various users at the same time a server with a high amount of Random-Access memory (RAM) would be necessary for hosting. Usually, servers with higher amount of RAM memory are significantly more expensive.

Furthermore, taking in account the country of Ecuador, this is a country which has a lower level of technological development presently compared to other more developed countries like the United States, Germany, or Russia. Therefore, the present methods mentioned on this document are not a common practice within the country. In fact, the correct application of these methods is also a challenge for the higher technologically developed countries. This means that there is a big opportunity for implementing this technology among the business and organizations within the country since all of them manage sensitive data about their users. Any company or venture that would like to attempt expanding these techniques among the country would have a good idea for business.

Finally, it is interesting that the difficulties present to protect the privacy of information hold a similar problem to what occurs in computer encryption. In encryption there is no perfect algorithm to protect the security of information, all algorithms have vulnerabilities and can be broken. Therefore, the quality of encryption is not measured on how unbreakable an algorithm is, but instead in how difficult, how much computer power and how much time it would be necessary to break an encryption. Also, the value of the information in time is an important aspect that is considered in encryption. In that sense a good encryption is an algorithm that can

protect the information for enough time until it loses most of its value. Besides, another important consideration in encryption is that computing power is always advancing, hence algorithms that are considered strong in the present era will probably not be so in the future. All these aspects apply to anonymization and pseudonymization as well.

The key idea for a good anonymization or pseudonymization of information is not that it becomes unbreakable, but that the concepts for a good encryption are also applied. In other words, that the identification of a specific individual has a high complexity, a high computational cost and is not possible to be performed on the time of value of the information. Also, it must be considered that all of these applies for the current state of technology since as technology advances most of this privacy protection methods will probably become obsolete. Thus, the area of privacy protection of information is an active field which will always be presenting new techniques and solutions.

RECOMENDATIONS

The techniques analyzed and implemented for both areas of privacy protection are not all the techniques present for these two methods. These are some of the multitudes of techniques that exist. The idea of the project was to bring a glimpse of what anonymization and pseudonymization are. It is not a complete guide on this highly broad and complex topic, but an introduction. Other important techniques that can be considered for future implementations of the application are differential privacy, and synthetic data.

Subsequently, to solve the RAM limitation among servers for web hosting, implementing temporary files is a solution to consider. Temporary files are stored temporarily on the disk of the computer, but they are immediately eliminated as soon as the program stops requiring these files for its processing. The framework used of FastAPI in Python offers the option of utilizing “Spoiled Temporary Files” (FastAPI, n.d.-e). These are files that are saved in memory until a threshold size and once the threshold is exceeded, they are stored temporarily on disk. The only thing needed for this change to be applied is to modify the definition for the anonymization POST endpoint. Instead of using “bytes” as the type of data for one of its arguments it must use “UploadFile”. In other words, replace the word “bytes” by the word “UploadFile” from the definition of the endpoint shown in Figure 27. Also, updating the respective import must be considered, “UploadFile” is imported from FastAPI and must be reflected on the import statement of Figure 8, which is speaking about the same file.

Furthermore, the application presented for this protype is a very simple full stack application. Thus, there is a vast list of improvements that can be done for the program. From the upgrades that can be applied some of the most relevant would be: to add user login or management, add routing for more pages, to make a more dynamic web interface which takes advantage of react components for the input configuration, adding a more robust validation of

the input dataset and configuration and applying vectorization methods for pandas which is the main library used currently in the application. From all the areas of advancement presented the most significant on the performance of the application is the addition of vectorization methods. This would mean rethinking the logic of the problems already solved for the techniques on the files “mask.py”, “noise.py”, “router.py” and “swap.py” illustrated in Figure 5 and applying vectorization. Vectorization is several times faster in terms of performance (Wyndham, n.d.). It is a necessity if big amounts of data want to be analyzed on a reasonable amount of time.

Subsequently, all the implementations done for the anonymization and pseudonymization techniques were done directly with Python code and the pandas library. This could be improved if specific libraries for anonymization and pseudonymization were used. Hence an investigation of python libraries relating to these subjects to apply them, would be an important consideration. Adding these libraries would also help on code readability and performance. Besides, if more complex techniques such as differential privacy or synthetic data were to be added in the future. These are highly complex techniques thus, implementing them directly in python would be a difficult and long work. Here libraries come very handy and are seen as a practical solution.

Finally, the objective of a web application is to be launched for it to be publicly accessible on the internet. Currently the application is only allowed to run locally as manifest in the “Configuration and Execution of the Program” section. Therefore, it is highly encouraged that if Universidad San Francisco de Quito USFQ, any other university, person, or entity, reading this document, finds the program useful, to upload it on a web host for it to be publicly available. This program serves as a base for anonymization and pseudonymization techniques, but it is highly encouraged that its development is continued by other parties that find it interesting. This way the program can evolve in a much more advance version helping in the

understanding and application of this complex topic, which is to find the ideal point between privacy of information and data analysis.

REFERENCES

Anonyome Labs. (2020). *What is Pseudonymization?* Retrieved on December 19 of 2021 from https://anonyome.com/2020/10/what-is-pseudonymization/

Burgess, M. (2020). *What is GDPR? The summary guide to GDPR compliance in the UK.* Retrieved on December 19 of 2021 from https://www.wired.co.uk/article/what-is-gdpr-uk-eu-legislation-compliance-summary-fines-2018

Complior. (n.d.). *Pseudonymization and Anonymization of Personal Data*. Retrieved on October 18 of 2021 from https://complior.se/pseudonymization-and-anonymization-of-personal-data-what-is-the-difference/

Databricks. (n.d.). *Parquet.* Retrieved on December 19 of 2021 from https://databricks.com/glossary/what-is-parquet

Dateutil. (n.d.). *Parser.* Retrieved on November 11 of 2021 from https://dateutil.readthedocs.io/en/stable/parser.html

Deepomatic. (2019). *Data Anonymization a Challenge to Face!* Retrieved on December 19 of 2021 from https://deepomatic.com/data-anonymization-a-challenge-to-face

FastAPI. (n.d.-a). *First Steps.* Retrieved on November 11 of 2021 from https://fastapi.tiangolo.com/tutorial/first-steps/

FastAPI. (n.d.-b). *Extending OpenAPI.* Retrieved on November 11 of 2021 from https://fastapi.tiangolo.com/advanced/extending-openapi/

FastAPI. (n.d.-c). *CORS (Cross-Origin Resource Sharing).* Retrieved on November 13 of 2021 from https://fastapi.tiangolo.com/tutorial/cors/

FastAPI. (n.d.-d). *Bigger Applications - Multiple Files.* Retrieved on December 10 of 2021 from https://fastapi.tiangolo.com/tutorial/bigger-applications/

FastAPI. (n.d.-e). *Request Files.* Retrieved on December 17 of 2021 from https://fastapi.tiangolo.com/tutorial/bigger-applications/

Frankenfield, J. (2020). *De-Anonymization.* Retrieved on December 19 of 2021 from https://www.investopedia.com/terms/d/deanonymization.asp

GeeksforGeeks. (2021). *How to use files in public folder in ReactJS?* Retrieved on December 12 of 2021 from https://www.geeksforgeeks.org/how-to-use-files-in-public-folder-in-reactjs/

Hamidovic, H., Kabil, J. Šehić, E. (2019). *EU General data protection regulation (GDPR) - Anonymisation and pseudonymisation in function of data protection.* Retrieved on December 19 of 2021 from https://www.researchgate.net/figure/Example-of-pseudonymization-by-tokenization_fig5_332686691

IBM., Business Higher Education Forum., Burning Glass Technologies. (2017). *The Quant Crunch How The Demand For Data Science Skills Is Disrupting The Job Market.* Retrieved on December 26 of 2021 from https://www.ibm.com/downloads/cas/3RL3VXGA

Imperva. (n.d.). *Anonymization*. Retrieved on October 18 of 2021 from https://www.imperva.com/learn/data-security/anonymization/

Maldoff, G. (2016). *Top 10 operational impacts of the GDPR: Part 8 – Pseudonymization.* Retrieved on December 19 of 2021 from https://iapp.org/news/a/top-10-operational-impacts-of-the-gdpr-part-8-pseudonymization/

Manca, F. (2019). *Introduction to ASGI: Emergence of an Async Python Web Ecosystem.* Retrieved on December 8 of 2021 from https://florimond.dev/en/posts/2019/08/introduction-to-asgi-async-python-web/

Material UI. (2021). *Material UI.* Retrieved on November 11 of 2021 from https://v4.mui.com/

New America. (n.d.). *Data Perturbation Methods*. Retrieved on October 18 of 2021 from https://www.newamerica.org/oti/reports/primer-disclosure-limitation/disclosure-limitation-techniques/

OAI. (2021). *The OpenAPI Specification.* Retrieved on November 11 of 2021 from https://github.com/OAI/OpenAPI-Specification

Octopize. (2021). *What anonymization techniques to protect your personal data?* Retrieved on December 19 of 2021 from https://octopize-md.com/en/2021/07/28/what-anonymization-techniques-to-protect-your-personal-data/

Peinoit, P. (2019). *Data Privacy through shuffling and masking – Part 1.* Retrieved on October 18 of 2021 from https://www.talend.com/blog/2019/08/13/data-privacy-shuffling-masking-part-1/

Python. (2021). *IO - Core tools for working with streams*. Retrieved on November 11 of 2021 from https://docs.python.org/3/library/io.html#io.BytesIO

Rayward. O. (2018). *Better performance by optimizing Gunicorn config*. Retrieved on December 26 of 2021 from https://medium.com/building-the-system/gunicorn-3-means-of-concurrency-efbb547674b7

Record Evolution. (2020). *Data Anonymization Techniques and Best Practices: A Quick Guide.* Retrieved on October 19 of 2021 from https://www.record-evolution.de/en/data-anonymization-techniques-and-best-practices-a-quick-guide/

Rivas, S. (2020). *Personal data anonymization: key concepts & how it affects machine learning models.* Retrieved on December 19 of 2021 from https://tryolabs.com/blog/2020/06/11/personal-data-anonymization-key-concepts--how-it-affects-machine-learning-models

Singh, A. [Abhishek] (2020). *Create-react-app files/folders structure explained*. Retrieved on December 12 of 2021 from https://medium.com/@abesingh1/create-react-app-files-folders-structure-explained-df24770f8562

Singh, A. [Ashish] (2019). *An intro to Webpack: what it is and how to use it*. Retrieved on December 12 of 2021 from https://www.freecodecamp.org/news/an-intro-to-webpack-what-it-is-and-how-to-use-it-8304ecdc3c60/

Sridhar, A. (2018). *A quick guide to help you understand and create ReactJS apps*. Retrieved on December 12 of 2021 from https://www.freecodecamp.org/news/quick-guide-to-understanding-and-creating-reactjs-apps-8457ee8f7123/

Szoc, S. (n.d.). *Data Anonymization*. Retrieved on October 18 of 2021 from https://ec.europa.eu/cefdigital/wiki/download/attachments/68329287/Anonymization%20explained_sara%20szoc.pdf?version=1&modificationDate=1558525207019&api=v2

Tabakovic, A. (2020). *Only a little bit re-identifiable?! Good luck with that.* Retrieved on December 19 of 2021 from https://mostly.ai/blog/only-a-little-bit-re-identifiable/

Tan, S. (2020). *Pseudonymization 101.* Retrieved on November 11 of 2021 from https://www.privitar.com/blog/pseudonymization-101/

Tokenex. (2018). *Pseudonymization vs. Anonymization: GDPR.* Retrieved on December 19 of 2021 from https://www.tokenex.com/blog/general-data-protection-regulation-pseudonymization-vs-anonymization

UserCentrics. (2021). *Data Anonymization: The What, Why, and How of Data Anonymization*. Retrieved on December 19 of 2021 from https://usercentrics.com/knowledge-hub/data-anonymization/

Uvicorn. (n.d.). *Introduction.* Retrieved on November 11 of 2021 from https://www.uvicorn.org/

Wyndham, J. (n.d.). *Fast, Flexible, Easy and Intuitive: How to Speed Up Your Pandas Projects.* Retrieved on November 11 of 2021 from https://realpython.com/fast-flexible-pandas/