GGML-Generic GDPR Management Layer

João Carlos Teixeira
joao.carlos.teixeira@ist.utl.pt

Instituto Superior Técnico,
Lisbon, Portugal

September 2020

Abstract

The directives implemented in May 2018 in the General Data Protection Regulation (GDPR) [8] are responsible for regulating the processing and circulation of data classified as personal within the European Union. With the implementation of GDPR, the need to develop a set of modifications in the methods of processing and storing personal data by systems and web applications was created.

This thesis developed one of the first help systems for the implementation of GDPR in web systems and applications with automatic classification and data management features, this system was developed using meta programming and study of relational trees and relational graphs. The data can be classified automatically by the personal or public system according to the GDPR rules through information obtained by the programmer of the application where it was applied. The system was developed to be applicable in various programming languages and operating systems without having compatibility problems and with few adaptations to be made, in the case of proof of concept of the thesis the system is demonstrated in SQLAlchemy and was tested in the implementation of an application generic data collection.

It has been proven that due to the use of this system, the workload of web application programmers is reduced in the regulation of data processing and storage without, however, the creation of high overheads in the system.

Keywords: General Data Protection Regulation, automatic classification,relational trees, relational graphs, SQLAlchemy

1. Introduction

The General Data Protection Regulation (GDPR) [8] was published in May 2016 and on 25th May 2018 this new regulations on the privacy and protection of personal data of individuals within the European Union were established. The fundamental purpose of these new regulations is to force data storage entities to guarantee the rights of the owners of personal data records.

Such rights are mainly formed by the knowledge about what data is being collected and stored, the purpose of such collection, the storage time as well as all processes to which personal data undergo.This issue with the consent request initiated the creation of some tools for introducing and processing consent requests as discussed in the article "Managing Consenting Workflows under GDPR"[6].

In the new regulation implemented, the differences between personal data, which maintain a traceable link between the data and an individual person, and non-personal data, which do not have that link, were defined and established the rules and regulations for the treatment of each of them. In the present day the development of software that collects, processes and stores data according to GDPR standards is mainly based on the developer's ability to program and structure a system that follows the regulations of the European Union. Because it is necessary for system designers to study these standards and create a programming logic that handles specifically the treatment of personal data, it is also part of the developer's responsibility to distinguish between personal and non personal data records.

The programmer, according to the current model of web application development and for the development of an application according to GDPR regulations, is responsible for the following essential points:

- Define and structure the data model used by the application to be developed.
- Classify and mark the data that will be considered personal data, for these personal data the programmer will need to develop forms and make their implementation to obtaining consent from the data owners to do the collection

and storage of its records.

- Manually define and implement the queries to the database to make the data available to the user.
- Establish an application logic responsible for making sure that personal data receives a different treatment, imposed by the GDPR, from public data and that the storage conditions are those regulated.

In addition, part of the programmer's responsibility is to develop and to implement all the application logic responsible for the operation of the application, data processing, data iteration for the user, as well as the entire graphical interface, responsible for the connection between the user and the application server.

Currently there is an absence of tools that help the developer to apply the GDPR. This thesis main objective is to evaluate the possibility of creating an automatic support layer/middleware. That will help the programmer to apply the necessary modifications to the personal data records for the execution of the system and management of records of personal data stored in accordance with the regulations imposed by the European Union.

For the development of the system created in this thesis it has been considered that most web applications currently are developed in an architecture called as multitier architecture, or tier N applications. This consists in an architecture of separated physical layers, each responsible for runnig part of the application code. In this architecture the presentation functions, application logic and data management may be physically separated in system components of the application, as can be seen in Figure 1. This architecture was studied in this project because it is considered a strong, solid technique and is very common in the web application developers community nowadays.

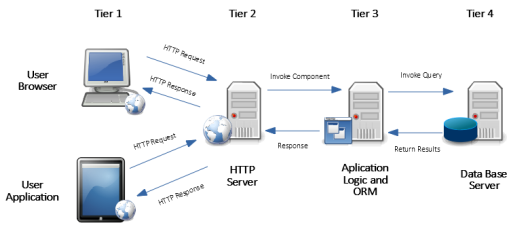

Figure 1: System multitier Scheme

1.1. Objectives

With the development and implementation of this new layer it will be possible to extend the functionality of a standard ORM (Object-relational mapping) to provide a set of code and primarily generic functions that automate and assist the programmer in his current responsibilities. This extension should be generic, and its focus should not be a single system or ORM, so the programmer is not forced to choose a framework due to the need to implement this system.

This new layer will be formed by memory elements and a set of functions and functionalities made available to the programmer with the main objective of helping to:

- Automate the classification of intermediate data classes as well as their management, according to GDPR regulations.
- Automatically infer the classification of personal class by the system, through graphs obtained due to the relationships included in the relational databases, making its differentiated treatment, if they are classified as personal data.
- Create an event manager to assist in the construction and submission of forms for obtaining and renewing consent from data owners.

For the features described above, this middleware should have a set of features and functions that will be grouped in two groups, passive features, primarily responsible for the classification, marking and storage of data, and active features, whose main objective is the manipulation and treatment of data classified as personal, helping in the implementation of GDPR in addition to making this functions available to the user with the purpose of giving him control over his data.

2. Background

Since May 2018, the storage and processing of data within the European Union has undergone several modifications due to the introduction of the new GDPR regulations. This regulations forced companies dealing directly with personal data collection and processing were required to revise their systems and standards for storage and processing of personal data.

The systems currently implemented for the storage and management of personal data, previously collected to be handled by data storage entities, resort to manual classification and manual segmentation of the collected data. For this reason this systems are very dependent on the knowledge and competence of the software developer or the person in charge of data processing who is responsible for the executing of the implemented system, thus decreasing the possible efficiency value for the system.

2.1. General Data Protection Regulation

One of the main purposes of this regulation is to give back to the users the control over their personal data records. To this end, user's rights as well as the obligations of the database controllers were regulated in chapter III and IV of the GDPR [8]. Some of these regulations are more structural policies, and not necessarily implemented in the system / applications, while the following regulations have received practical modifications, that is, these regulations have an active impact on the implementation of the storage of personal data carried out by applications and other systems.

2.2. Data owners' rights

The following list groups the main changes made to the GDPR [8] (with focus on the the rights of legal owners of personal data) that have an influence on the implementation of the data layer of web applications created to carry out data storage. These changes are all listed and explained in Chapter III of the GDPR [8] document and are as follows:

- **Transparency of information and notifications (sec 1, art 12-a) -** The data controller is required to provide the user with information on the reason for data collection, how long the data will be stored and other relevant information. This information should be concise, written in plain and clear language.
- **Rules for exercising rights of data subjects (sec 1, art 12-b) -** If the user requests information other than the available at that moment (meta-data), the responsible entity should grant the requested information if relevant.
- **Information of personal data (sec 2, art 13-a) -** During the collection of users' personal data, the responsible entity shall provide the user with the identity and contact details of the entity responsible for processing their data, their purpose, and whether they will be transferred to a third party in the future.
- **Access to personal data (sec 2, art 13-b) -** The responsible entity for the personal data previously collected should also provide the user with a copy of the collected data and a means of access them while stored.
- **Rectification right (sec 3, art 16) -** The owner of the data has the right to have his personal data, in a suitable period of time, modified or supplemented if he wishes so.
- **Right to be forgotten (sec 3, art 17) -** The user is entitled to have his / her data deleted, not affecting the results already obtained from previous data processing.
- **Right to treatment limitation (sec 3, art 18-a) -** The data holder may, in special situations of misuse of his data, require the processing of their data be terminated.
- **Renewal of data processing rights (sec 3, art 18-b) -** In case of limitation of data processing, this data could only be reused with a renewal of the user's explicit consent.
- **Obligation to notify the deletion of personal data or limitation of treatment (sec 3, art 19-a) -** The entity in charge of processing a user's personal data is responsible for communicating, if possible, to all recipients of them if the data was modified, deleted or the processing limited.
- **Access to the data of the entities responsible for processing personal data (sec 3, art 19-b) -** If the data owner requests the data processing entity, he / she is responsible for providing the data of the recipients of their personal data.
- **Data portability right (sec 3, art 20) -** The data owner has the right to be provided with the same data, in a structured and automatic reading format, if he requests to transmit this data to a third party for processing.
- **Right to oppose data processing (sec 3, art 21) -** The data owner has the authority to oppose the processing of his personal data and the responsible entity must immediately cease processing of his personal data.
- **Right to oppose automated individual decisions (sec 3, art 22) -** The user should not be subject to any decision made solely based on automated processing of their data, including profiling from their data.

2.3. Obligations of database controllers

In this case, the list refers to the modifications made to the regulations concerning the entities responsible for the storage and processing of personal data collected by web applications. The changes mentioned in the following list can be found in chapter IV of the GDPR document issued by the European Union.

- **Data protection (sec 1, art 24-a)-** Given the purpose, cost in applying, the nature and context of the personal data to be processed, the controller of the database shall employ all technical measures to ensure the protection of such data as standard.

- **Data storage period (sec 1, art 24-b)-** The responsible entity should ensure that the data to be processed should be only necessary for that purpose, thus limiting the extent of data collection as well as its storage period.

- **Records of data processing activities (sec 1, art 30) -** Each data handling entity shall ensure that a record is kept of all processing activities carried out under its responsibility. This record shall include all metadata of such processing, such as the contact details of the executor, purpose, description of the categories of data used and, where appropriate, third parties to whom the personal data were transmitted. In addition, each subcontractor must have its to-do list with all the items described above.

- **Treatment Safety (sec 2, art 32) -** It is the responsibility of the data storage entity to employ all possible data protection measures such as pseudonymization and encryption of data, the ability to ensure confidentiality, establish users' availability and access to their data and to regularly test the effectiveness of the measures used to protect data. Data controllers have the task of assessing the level of security that needs to be implemented against the data being collected and stored.

- **Notification of a personal data breach to the supervisory authority (sec 2, art 33) -** In the event of breach of the confidentiality of stored data, the entity responsible for notifying this fact shall be authorized within 72 hours. If a subcontractor becomes aware of a breach, it shall notify the lead data entity, which shall in turn notify the authorities. This notification should consist of the nature of the violation, number of users affected, description of possible consequences, and possible steps to mitigate the consequences of this violation.

- **Notification of a personal data breach to the data subject (sec 2, art 34) -** In the event of a data leakage that undermines the data subject's rights or freedoms, the data owner shall be notified, without delay, of the occurrence and the measures needed to mitigate the effects of such data leakage.

- **Data protection impact assessment (sec 3, art 35)-** Prior to the processing of the data, the responsible entity shall carry out an impact assessment of the operations carried out in that processing in the protection of the stored data. This assessment should group all high-risk operations to minimize the number of executions if possible.

- **Designation of Data Protection Officer (sec 4, art 37) -** The data storage officer is also responsible for appointing the data protection officer who should take into account the professional qualities of the data protection officer as well as his or her data protection expertise. The duties of this officer shall be to advise the responsible entity on all data protection matters and to cooperate with supervisory authorities.

2.4. Penalties

According to the GDPR in article 83 [8], data collection, processing and storage companies that do not comply with the imposed obligations or other GDPR regulations will be fined.

These fines are divided in two groups, in the first group the company will be penalized up to 2% of its annual earnings or 10 million euros, in the second group the company will be penalized up to 4% of its annual earnings or 20 million euros. The amount of the penalty will be the highest amount made available to the group where the entity is included and the choice of the group will depend on the violations made by the entities, also depending on the amount of data that was used improperly as well as on the importance and sensitivity of personal data.

It is possible to group and analyze cases of penalties imposed on companies located within the territory of the European Union with the help of the GDPR Enforcement Tracker website [1].

2.5. GDPR supporting tools

At the beginning of 2018 studies and questionnaires [4] were conducted which showed that, although 98% of Fortune 500 companies believed they comply with the new GDPR, only 39% of UK companies and 47% of United States of America had formed internal teams to establish the changes imposed by the new regulation and only a third hired third parties or experts for this goal. In companies specialized or involved in the monitoring of personal information from individuals only 29% of UK and 18% of USA companies were hiring teams of privacy experts to deal with these modifications.

Nevertheless, a study [7] carried out shows that a broad part of the companies that work in data storage and processing are directing a considerable part of their funds to implement procedures necessary to enforce compliance with GDPR regulations, to avoid future failures in inspections and monetary penalties.

Much of the work carried out by these teams of specialists consists of manual development of application logic and manual data classification, but there are also tools available to programmers that facilitate the implementation and imposition of GDPR.

3. Implementation

This chapter provides a detailed description of all the decisions made in order to implement the system **Generic GDPR Management Layer (GGML)** that provides the features to regularize the changes implemented by GDPR in the storage and treatment of personal data. This implementation and practical tests were carried out as proof of concept for this thesis.

3.1. System Functionalities

In a relational database it is possible to represent the classes that constitute the database and their relationships between classes through graphs, in these graphs the classes without higher levels are called root classes.

In order to simplify the work of web application developers, this system provides the possibility for the programmer to simply mark as personal classes the classes considered as application roots, leaving the system to mark the classes that are considered personal due to the relationships they have with the roots and that will later lead to the creation of personal data sets referred to as **personal records**. This feature is achieved by studying the graphs created based on database relationships and the privatization function which is a backend function that takes an intermediate class and transforms it into a personal class.

As previously mentioned, this system was developed with the main objective of helping in the development of web applications according to GDPR regulations. For that, it will be the responsibility of the programmer to make the calls of all the functions present in this library but in addition to use of the passive features, features developed and implemented to provide help in the implementation of the GDPR regulations according to the optics of the **application programmer**, the utility functions can be divided into two groups taking into account the type of user who will benefit from their use. The first group associates all the functionalities that are used by the **Data Protection Officer (DPO)**, a type of user whose objective is to prove the implementation of the GDPR regulations, which manage data in terms of classes and affecting a large number of objects, such as all those derived from *show* and *change function*, the second group that is mostly used by the **average user** manipulates one personal record at a time, or sets of related records, such as the *show class data* and *forgets class data* functions.

3.2. Programing Enviroment

To demonstrate the concepts idealized in this thesis as well as prove their successful implementation, some ORMs and frameworks were studied. The **GGML** is implemented as an ORM extension providing the programmer with a set of new features to manage information regarding the implementation of GDPR in web applications and systems.

The selected ORM, to demonstrate the implementation og this system , was SQLAlchemy [2], which is an ORM library developed in python[3], this choice was due to be an open source library, while still providing its user with a large selection of persistence models and features, in addition this library treats the databases as a collection of objects which perfectly adapts to the use that we intend of the ORM for this thesis.

SQLAlchemy consists of a core and an ORM that allow python objects to be mapped in relation to relational data tables. Changes made to the library of SQLACHEMY[2], used for the practical demonstration of this system, were kept to the minimum possible maintaining, in that way, the compatibility with previous versions of it.

3.3. System implementation

3.3.1 Classical SQLAlchemy architecture

The library called SQLAlchemy is mainly made up of two components called core and ORM, as shown in figure 2.

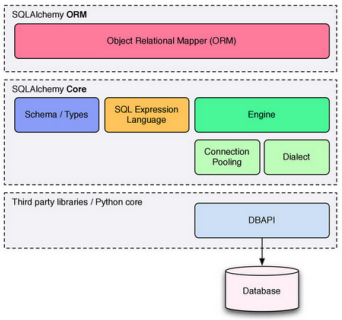

Figure 2: SQLAlchemy Core and ORM [5]

The core of the module is in turn formed by the following components:

- **Engine** - records and connects to a specific database
- **Dialect** - interprets generic SQL language and database commands
- **Connection Pool** - Make a list of specific database connections
- **SQL Language** - Translate Python into SQL statemens
- **Schema/Types** - Represent tables, data types and columns in Python objects.

On the other hand, the SQLAlchemy ORM performs the construction of Python objects, mapping

with relational tables, allowing the persistence of these objects. This module was developed over the core previously explained, thus using the core to generate SQL and commands in order to manage the database and data.

3.3.2 New components of the architecture

The Generic GDPR Management Layer (GGML) was developed and created based on the ORM already present in the SQLAlchemy library, thus lowering maintenance problems, learning curve and compatibility of the new system.

This system was developed to prove the concept of this thesis and was created as an extension to the ORM module present in the SQLAlchemy library, thus adding existing functionalities in order to facilitate the implementation of GDPR directives in the development of web applications.

This system takes advantage of the persistence of objects made available by the classic ORM to manage and save the metadata necessary for the implementation of GDPR in a generic web application.

This system developed in the new layer is mainly composed of tables with the purpose of being in the backend of the applications to be developed, these tables or meta-tables will be used to implement the guidelines presented in the GDPR. The system is also composed of a set of data structures and functionalities, made available by the system library, in order to manage the backend tables and the metadata stored in parallel with the personal data.

3.4. Metadata

To achieve the goal of creating a system that helps in the implementation and enforces the necessary modifications to implement the new GDPR regulations, internal date structures were created and extended in order to store all the relevant metadata and data enrichment necessary to support the idealized system. The main internal data structures, that have information attached to each object of each class, are as follows:

- **List of classes classified as Personal Data** - consists of a list stored in the database, parallel to the data, and used to know if a class is a personal class without access to it according to GDPR regulations.
- **Collection date information** - data enrichment where the date an object was created by the system and stored is combined with the corresponding data.
- **Personal tag information** - data enrichment where the name of the nearest root is combined with the corresponding data. In other words, a field to be filled with the name of the root class that is closest to the class, which creates this object, in the graph.

In these cases, data enrichment is a classification given when information needed to support the developed system has been linked to the data and stored together with them in the databases.

3.5. Inference of personal classes

For the development and implementation of the self-classification functionality of intermediate classes as personal or public, it was necessary to build graphs during the creation of new classes. These acyclic and directional graphs, as exemplified in figure 3, were developed by analyzing the relationships between classes stored in the databases and the following concepts were taken from this analysis:

- **Root Class -** Classification attributed to a node of a graph that does not have nodes at higher levels, in this case and attributed to classes that do not have classes with a child-to-father relationship with them. These classes will be marked by the application programmer if they contain personal information.
- **Intermediate Class -** Nodes belonging to levels lower than nodes previously classified as roots. For this case, specifically the intermediate classes are those treated by the self-classification functionality, that is, the system is in charge of classifying these classes as personal or public.

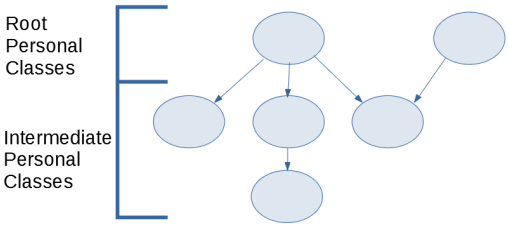

Figure 3: Directional Graph

- **Path -** Sequence of relationally interconnected classes that form a link between a previously established source class and a destination class. These classes belonging to this set are consecutive and without repeated arcs.

4. Results

4.1. Programming procedures

The ORM in this library has been extended to provide the developer new features and help to enforce the GDPR regulations, with the following main features:

- Mechanism that allows the programmer to easily classify the roots of the relationships between the classes of his application, and existing in his database, as personal and to classify and automatically mark the intermediate classes in those same relationships.
- Functions that enable software developers to control and manipulate stored metadata used in the system to help enforce the GDPR regulations, this metadata is common to all objects of the same class.
- Functions that allow the programmer to easily control the expiration date of stored data, as well as listing and deleting data that is out of date. This feature can be used by the programmer to perform a cyclic cleaning of the data in question or as a warning of need for renewal of consent.
- A set of tests with the main purpose of calling the programmer's attention if there is a need to complete the metadata used to support the system or to test whether the data is personal or not.
- Function that searches the database for specific personal data as well as all personal data that is related to the former in descending form, that is, with a parent-child relationship.

4.2. Comparison with classic programming model

With the implementation of this library, developed for this thesis, web application programmers will have at their disposal a set of functions and features that will simplify the creation of a personal data maintenance system that will simplify and help the implementation of GDPR.

This system will apply improvements to the classical programming model in which the programmer was responsible for the creation and collection of personal data meta data, retention of this meta data up to date, as well as the creation of the personal data maintenance system in accordance with the GDPR. In the classic programming model, the programmer would also have to implement functions to present the information to the application's users and all the application logic needed for its correct functioning, just as in this new model.

The main advantage of this new model is to provide the programmer with help in regulating the handling of personal data records according to the GDPR regulations thus leaving more time and workload for the development of the stored information display system. Therefore it is no longer totally the programmer's responsibility to develop the system that imposes the regulation of the European Union, putting aside no aspect in terms of system compatibilities.

4.3. Application examples

In order to prove the concepts studied and the correct functioning of the system designed and implemented in this thesis, web applications were developed and implemented, which perform the collection and treatment of personal data, as practical examples of the use of this system by programmers to analyze the advantages of its use. As a proof of concept an application was developed and implemented applying the architecture model present in figure 1 and a data and relational structure sufficiently complex to study the functionalities provided by the system efficiently.

4.4. Practical example: Restaurant Classification system

For testing the functioning of the library and system, developed during the course of this thesis, the structural model in figure 4 was used. This example in addition to collecting and storing , according to the regulation present in the GDPR, of the personal data of the users and of the restaurants as well as the creation of visit records that keep the data related to a person's visit to a restaurant also introduces the structural fragment related to the restaurant workers, making the collection and storage of their personal data, working hours and classification given by the user.

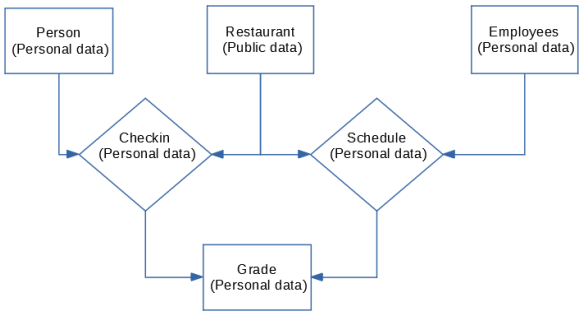

Figure 4: Restaurant rating Web application data template

As a complement to the proof of concept of this system, an android application was developed running parallel to the system accessed by the web browser. This application would act as a proxy that could make a REST requests and that endpoint would make a call to the code in the server-side. This application would implement the restaurant rating system with the data structure shown in figure 4.

The main purpose of this mobile application is to quickly collect and store records of users' visits to restaurants as well as their rating, thus developing a more practical part of the practical example used as a proof of concept. Figures 6 and 7 shows screen mockups that would work as data entry forms for

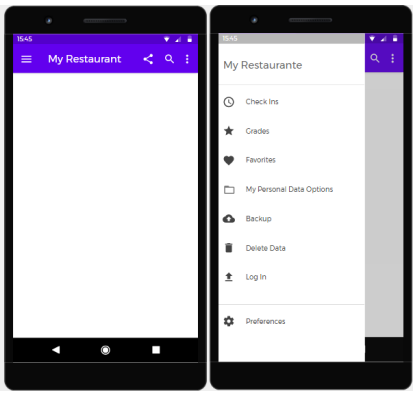

Figure 5: Android application index screen

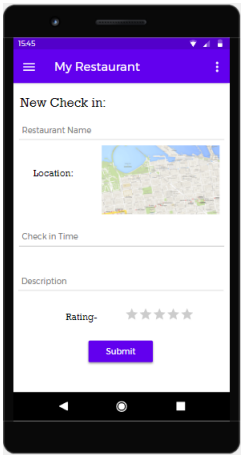

Figure 6: a) Check in screen example

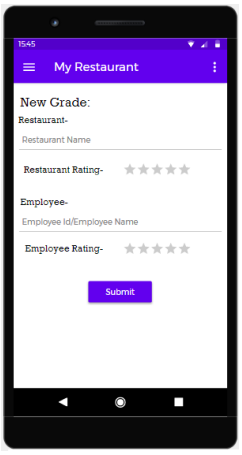

Figure 7: b) Grade screen example

the practical example, in the screen demonstrated in the point a) is allowed to the user to identify the restaurante visited, by name and location, to identify the time frame in which the visit took place, to write a short description, which is optional, and finally to introduce a general classification of the visit to the establishment.

On the other hand, on the screen shown in point b) of the image 7, is made available to the user the possibility to grade specifically the restaurante where he was, as well to grade the employee responsible for the service provided during the visit to the restaurant.

In this practical example the records collected by the logic displayed on the screen in point a) are classified as personal records under the influence of the root class person, and belong to the user who performed the classification, the records stored by the screen in the point b) are also personal but they can bellong to the root user or to the root employee, because the class grade, where these records are stored, is at the same distance from the class person and from the class employee, this varying with the way the data structure was done by the programmer and the consent obtain from the user, in this specific case, the grade class is classified as personal and belongs to the user.

4.5. Time Overheads

To test the impact of the introduction of the new system on the development of web applications according to the GDPR guidelines, it was first studied where in the system flow it would be the point with the worst performance in terms of time.

As you can see in table 1, a comparison was made between an application using only the classic model and an application using the new GGML system. In this comparison, a local database was created to avoid delays due to communications and the number of objects introduced from a class was increased to simulate the increase in the amount of personal data to be processed in the case of the publication of the web application.

As it is possible to observe in table 1 with an exponential increase in the amount of data and without the delay of communications, the time overhead value is always close to the 10% increase.

In the case of the tests presented in table 1, where the objects inserted in the database are of a reduced size (approximately 20 bytes), in table 2 the objects

Num. of obj.	Classic Model	GGML
10	0.03 s	0.08 s
100	0.497 s	0.623 s
1000	4.49 s	5.53 s
10000	56.025 s	1 m 0.5 s
100000	9 m 23.59 s	10 m 32.38 s

Table 1: Overhead Times (20 bytes)

were artificially increased to test the impact of the system developed in object inserts, of size extensive (1 KB) in the database.

Num. of obj.	Classic Model	GGML
10	0.06 s	0.106 s
100	0.8 s	0.961 s
1000	9.92 s	11.48 s
10000	1 m 37.7 s	1 m 54.96 s
100000	16 n 41.59 s	19 m 25.78 s

Table 2: Overhead Times (1 KB)

As you can see from the second table, the time values increase with the size of the objects, but the increase of the time interval between the use of the system and the classic model remains, for high amounts of data, in the gap between 10% and 15% as shown in table 1 and table 2.

With this differentiation of tests it was possible to conclude that, despite a residual increase in the running times of the system, the temporal overheads, regardless of the size of the objects and information, remain limited in an interval appropriate to the use of the system.

4.6. Memory Overheads

Another test developed with the aim of testing the impact of the introduction of GGML in applications with data storage features was the monitoring of the evolution of the size of the database with the use of the system and without.

Num. of obj.	Classic Model	GGML
10	24.6 kB	53.2 kB
100	24.6 kB	61.4 kB
1000	73.7 kB	135.2 kB
10000	512 kB	892 kB
100000	5.4 MB	8.9 MB

Table 3: Memory Overheads(20 bytes)

As for the tests of overheads related to times of use, memory tests were made with ojects with a artificially size fixed. In the values that can be seen in table 4 the objects have a size of at least 1 Kbyte.

As it is possible to observe in table 3, in tests with the insertion of objects with a size in the order of 20 Bytes the size of the local database varies due to the implementation of the system in the order of 40% to 50% increase in memory occupation. On the other hand, with table 4 it is possible to observe that in tests with inserts of objects of larger size (1 kB) the percentage of increase between the size of the databases is very small, reaching below 1%.

Num. of obj.	Classic Model	GGML
10	90.1 kB	118.8 kB
100	634.9 kB	663.6 kB
1000	6.1 MB	6.1 MB
10000	60.3 MB	60.4 MB
100000	603.3 MB	603.4 MB

Table 4: Memory Overheads(1 KB)

5. Conclusions

In this thesis it was possible to show what the new **General Data Protection Regulation (GDPR)** directives implemented in the European Union consist of, as well as their importance and the resulting needs. A presentation and summary of the tools that already address the needs of part of the needs created by **GDPR** is also made.

Then the presentation and description of the tool that was developed in the thesis, ggml, is made, as well as its advantages, operation mode and overhead accumulation tests. A study was made on the impact of this new system model on the implementation of web applications, during this development and study it was always taken into account that the tool remained generic, being able to be easily adapted to each ambience or Programming language

As a product of the study carried out in this thesis and as a proof of concept, a system was developed, **GGML- Generic GDPR Management Layer**, which provides a set of features that help the programmer to implement systems and web applications following the GDPR guidelines.

With the use of this new layer it is possible to take advantage of a set of advantages that are presented in the following sections depending on the group of users it affects.

5.1. Benefits for programmers

The layer developed in this thesis, **GGML**, offers a series of advantages that help programmers who design and implement web applications, which perform personal data storage functionalities according to GDPR directives. These advantages are:

- Reduce the workload of programmers in implementing and adapting web applications according to European Union regulations,
- Simplify marking of data as personal,

- Decrease the time needed to mark all data as personal,
- Facilitate the increase of data to be processed according to the GDPR.

5.2. Benefits for DPOs

Another class of users that benefits from the advantages provided by the new layer of personal data processing and **DPO**. These users are professionals responsible for the maintenance and management of personal data according to the GDPR and Their main advantages are:

- Avoid human errors in classifying and marking data as personal data,
- Reduce errors in data markup due to the amount of data,
- Automate the maintenance of personal data in the web application storage system,
- Increase knowledge about personal data existing in the organization.

5.3. Benefits for data holders

Finally, users classified as data holders are users of applications and systems developed by programmers and by connection users of the **GGML** system in second degree. These users also benefit from some advantages provided by the introduction of this layer in web applications, these advantages are:

- Provide features to the user make decisions regarding their personal data,
- Bring together in a system features that address all the problems created by the GDPR directives,
- Facilitate access to stored personal data and its meta data.

5.4. Future Work

For future ideas to improve the developed tool, it is possible to incorporate a system to control access to personal data to reinforce the security aspect of the system. The development and implementation of an automatic classification system for classes classified as roots, with the help of a list of fields that a class with personal information normally contains, in order to remove this responsibility from the programmer.

Some higher level audit features should be introduced, using them to take a sample of data from the web application database to demonstrate that they are well classified and marked, these features would be used as a way to test the proper functioning of this tool.

Finally, this tool is a demonstration of possibilities on the topic of GDPR implementation, not being an ideal solution to the problem. In this demonstration, the problem of deleting data at the request of the data holder was not addressed, this problem can be the cause of memory inconsistencies within the database because interconnection objects can be erased, creating the inaccessibility of certain data within the database.

Acknowledgements

The author would like to thank Professor João Nuno de Oliveira e Silva, who without his help and clarifications this thesis would not be possible.

References

[1] Gdpr enforcement tracker- list gdpr fine. https://www.enforcementtracker.com/. accessed: March 27th 2020.

[2] Sqlalchemy-the database toolkit for python. https://www.sqlalchemy.org/. accessed: December 3th 2019.

[3] Welcome to python.org. https://www.python.org/. accessed: December 3th 2019.

[4] W. Ashford. Top uk and us firms still overestimate their readiness for gdpr, study shows. *Computer weekly*, 2018.

[5] M. Bayer. Introduction to sqlalchemy. In *Pycon*, April 2013.

[6] S. I. Besik. Managing consent in workflows under gdpr, June 2020.

[7] P. Fisher, T. Grosser, and L. Iffert. Managing personal data beyond the gdpr. *tech. rep., BARC -Business Application Research Center*, 2018.

[8] M. SCHULZ. Regulation (eu) 2016/679 of the european parliament and of the council. *Official Journal of the European Union*, Apr. 2016.