GGML-Generic GDPR Management Layer
A middleware to support the development of GDPR related software requirements

João Carlos Teixeira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): João Nuno De Oliveira e Silva

Examination Committee
Chairperson: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. João Nuno De Oliveira e Silva
Member of the Committee: Prof. António Manuel Ferreira Rito da Silva

September 2020
Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.
To my family and girlfriend for their support.
Acknowledgments

With the completion of this thesis another important phase of my life ends, in this thesis I found myself working on a current topic where I felt interested and was happy with the final result. First I would like to thank Professor João Nuno de Oliveira e Silva, who without his help and clarifications this thesis would not be possible.

I would like to thank my colleagues for their help in times of difficulties and lack of ideas, to my parents, brother and sister-in-law for the support and help they provided me. Finally, I would like to thank my girlfriend for the enormous patience she always has with me and for her help that I will never be able to fully repay.
Resumo

As diretivas implementadas em maio de 2018 no Regulamento Geral Sobre a Proteção de Dados (GDPR) [1] são responsáveis pela regulamentação do processamento e circulação de dados classificados como pessoais dentro da União Europeia. Com a entrada em vigor do RGPD foi criada a necessidade de implementação de um conjunto de modificações nos métodos de processamento e armazenamento de dados pessoais por sistemas e aplicações web.

O problema focado nesta tese é a necessidade, criada pela implementação do RGPD, dos programadores de software desenvolverem lógica funcional de administração do armazenamento de dados pessoais. Esta Lógica é usualmente comum e genérica entre as aplicações que possuem capacidades de armazenamento e processamento de dados pessoais.

Esta tese desenvolveu um dos primeiros sistemas de apoio a implementação do RGPD em sistemas e aplicações web, fornecendo funcionalidades de classificação automática e administração de dados. Este sistema foi desenvolvido em Python e utilizando meta-programação e estudo de árvores e grafos relacionais. Os dados podem ser classificados automaticamente pelo sistema de pessoais ou públicos de acordo com as regras do RGPD e através de informações obtidas pelo programador da aplicação onde foi aplicado.

O sistema foi desenhado de forma genérica para ser aplicável em várias linguagens de programação, sistemas operativos e middlewares com poucas adaptações a efetuar. No caso da prova de conceito da tese o sistema foi demonstrado em SQLAlchemy e foi testado na implementação de uma aplicação de recolha de dados genérica.

Ficou comprovado que devido ao uso deste sistema a carga de trabalho dos programadores de aplicações web fica menor na regulamentação do processamento e armazenamento de dados sem, no entanto, existir a diminuição de desempenho do sistema.

Palavras-chave: Regulamento Geral Sobre a Proteção de Dados (GDPR), classificação automática, árvores e grafos relacionais, MOR (Mapeamento objeto-relacional)
Abstract

The directives implemented in May 2018 at General Data Protection Regulation (GDPR) [1] 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 implement a set of modifications in the methods of processing and storing personal data by systems and web applications was created.

The problem focused on this thesis is the need, created by the implementation of the GDPR, for software developers to develop functional management logic for the storage of personal data. This logic is usually common and generic among applications that have capacities for storing and processing personal data.

This thesis developed one of the first systems to support the implementation of GDPR in web systems and applications, providing automatic classification and data management features. This system was developed in Python and using meta-programming and the study of trees and relational graphs. The data can be classified automatically by the personal or public system according to the rules of GDPR and through information obtained by the programmer of the application where it was applied.

The system was designed in a generic way to be applicable in several programming languages, operating systems and middlewares with few adaptations to make. In the case of proof of concept of the thesis the system was demonstrated in SQLAlchemy and was it was tested in the implementation of a generic data collection application.

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 (GDPR), automatic classification, relational trees, relational graphs, ORM (Object-relational mapping)
Contents

Acknowledgments ........................................................................................................ vii
Resumo .......................................................................................................................... ix
Abstract ........................................................................................................................ xi
List of Tables .................................................................................................................. xvii
List of Figures ................................................................................................................ xix
Acronyms ....................................................................................................................... xxi

1 Introduction ................................................................................................................. 1
  1.1 Problem Statement ................................................................................................. 2
  1.2 Objectives .............................................................................................................. 3
  1.3 Results ..................................................................................................................... 4
  1.4 Thesis Outline ........................................................................................................ 5

2 Background .................................................................................................................. 7
  2.1 General Data Protection Regulation ..................................................................... 7
    2.1.1 Data owners’ rights ......................................................................................... 8
    2.1.2 Obligations of database controllers .............................................................. 9
    2.1.3 Penalties ........................................................................................................ 10
  2.2 GDPR supporting tools ......................................................................................... 11
    2.2.1 Data Flow Mapping Tool ............................................................................... 12
    2.2.2 Data Protection Impact Assessment ............................................................ 13
    2.2.3 GDPR Manager ............................................................................................ 13
    2.2.4 Wordpress plugin gdpr .................................................................................. 14
    2.2.5 DataDefender ............................................................................................... 15
    2.2.6 Virgil-crypto .................................................................................................. 15
    2.2.7 DBMS support .............................................................................................. 16
    2.2.8 Summary table ............................................................................................... 17
    2.2.9 Current Auditing ........................................................................................... 19
  2.3 Web application development Architectures ....................................................... 21
    2.3.1 Multitier architecture .................................................................................... 21
    2.3.2 Model-View-Controller ................................................................................ 22
## 2.3.3 Object-relational mapping

### 3 Generic GDPR enforcement Layer

3.1 Motivation Applications
3.2 Current Adaptations of an application to GDPR
3.3 GDPR-related features
  3.3.1 Data owners’ rights
  3.3.2 Obligations of database controllers
3.4 Generic GDPR enforcement System Funcionalities

### 4 GGML-GDPR Management Layer Implementation

4.1 Programming Environment
4.2 System implementation
  4.2.1 Metaclass
  4.2.2 Metadata
4.3 Personnal classes
  4.3.1 Labeling of personal classes
  4.3.2 Inference of personal classes
  4.3.3 Personal Class GGML Constructor
4.4 Application Programming Interface

### 5 Validation / Evaluation

5.1 Example Application
5.2 Comparison with classic programming model
  5.2.1 Definition of personal data
  5.2.2 Implementation of metadata storage system
  5.2.3 UI development and access to metadata
  5.2.4 Implementation of GDPR requirements
  5.2.5 GDPR Administrator Data Handling
  5.2.6 GDPR User Data Manipulation
  5.2.7 Restaurant Classification mobile application
5.3 Overheads Evaluation
  5.3.1 Time Overheads
  5.3.2 Memory Overheads
5.4 Relationship between system features and GDPR articles

### 6 Conclusions

6.1 General description and Advantages
  6.1.1 Benefits for programmers
  6.1.2 Benefits for DPOs
  6.1.3 Benefits for data holders
6.2 Future Work .............................................................. 73

Bibliography ............................................................... 75

A GGLM Classes ......................................................... 77
   A.1 Restaurant rating classes ...................................... 77
## List of Tables

1.1 Synthesis API functions .................................................. 5

2.1 GDPR Fines ................................................................. 11
2.2 CMS/websites pubins summary table .................................. 18
2.3 DBMS support summary table. ........................................... 18

3.1 Personal data owner rights from chapter III of GDPR. .............. 29
3.2 Obligations of database controllers from chapter IV of GDPR. .... 31
3.3 GDPR enforcing assessment by GGML. ................................. 39
3.4 GDPR requirements out of scope of GGML. ............................ 40

4.1 Metadata Categories ....................................................... 46
4.2 Function calls and their results ........................................... 50

5.1 Backend table of the practical example of figure 5.1 .................. 60
5.2 Overhead Times (20 bytes) ............................................... 68
5.3 Overhead Times (1 KB) .................................................... 68
5.4 Memory Overheads (20 bytes) .......................................... 69
5.5 Memory Overheads (1 KB) ............................................... 69
List of Figures

1.1 System multitier Scheme ................................................. 4
2.1 DIK pyramid ................................................................. 20
2.2 Model-View-Controller scheme ........................................ 22
2.3 Object-relational mapping operation .................................. 23
2.4 Object-relational mapping scheme .................................... 24
3.1 Diagram of the simpler practical example ............................ 26
3.2 Android application personal check in screen ....................... 27
3.3 Android application personal data management screen ............ 28
3.4 Simplified schema of the show class data function ............... 38
4.1 SQLAlchemy Core and ORM [27] ........................................ 42
4.2 GGML layers ................................................................. 43
4.3 Directional Graph ......................................................... 48
4.4 Part of the practical example .......................................... 48
5.1 Restaurant rating Web application data template ................... 53
5.2 Templates for input information from users ......................... 55
5.3 Administrator page ....................................................... 61
5.4 Administrator: Result from the function to show the expiration date 62
5.5 Administrator: Metadata Manipulation .............................. 62
5.6 Administrator: Expired data management ......................... 63
5.7 Database Contents Example ........................................... 63
5.8 Introduction of class index in the function ......................... 64
5.9 Actions for manipulating a metadata value ......................... 64
5.10 Result from the function to show all data related to a specific object 65
5.11 Web application template for personal data processing .......... 65
5.12 Templates for input information from users ....................... 66
5.13 Web application template for personal data processing (personal data access) 67
Acronyms

API  Application Programming Interface. 4
DML  Data manipulation language. 24
DPO  Data Protection Officer. 34
DSAR  Data Subject Access Rights. 13
GDPR  General Data Protection Regulation. xi, 1–5
GGML  Generic GDPR Management Layer. 41
MVC  Model View Controller. 22
ORM  Object-relational mapping. 4, 5
REST  Representational State Transfer. 65
RGPD  Regulamento Geral Sobre a Proteção de Dados (GDPR). ix
UML  Unified Modeling Language. 48
Chapter 1

Introduction

The General Data Protection Regulation (GDPR) [1] was published in May 2016. On 25th May 2018 this new regulations on the privacy and protection of personal data of individuals within the European Union was enforced. 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 new regulation defined, 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, and established the rules and regulations for the treatment of each of them.

The records of personal data referred in this regulation were divided into two groups: common personal data records (for example name, age, address and credit information) and sensitive personal data records (such as race, genetic and biometric data, religion and others). In this second group, data processing is unlawful unless there is explicit consent to do so or if the data has already been made public by the owner as it is previously described in Chapter II, Article 9 of the GDPR[1].

One of the other changes in the collection of personal data records was that consent changed from passive consent (where if the user uses the service provided by the Web application or data company he would be agreeing to the storage of their data) to active consent, where the user comes across a clear and concise form that explicitly reports the purpose of data collection, as it is described in the Chapter II, Article 14 Section 2 of the GDPR[1].

The user must also agree to the information usage provided, so his personal data records can be collected and stored. It should be also possible for the user to withdraw the consent given at any time, or to rectify the previously stored data, as described in chapter II section 3 articles 16,17,18 and 21[1]. This issue with the consent request drove the creation of some tools for introducing and processing consent requests as discussed in the article "Managing Consenting Workflows under GDPR"[2].

Another main change was that companies that are responsible for data storage should ensure that the user has control over their personal data records, and guarantee that the user is notified if there is a security breach or information leakage on their servers.
With these modifications, companies and data storage controllers became responsible for the entire life cycle of the personal data collected by them. Due to these changes, the need to develop tools for the application and verification of the regulation in a simple, optimized and automatic way, arose and no optimal answers have been developed yet.

1.1 Problem Statement

Nowadays 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. Since 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:

a) Define and structure the data model used by the application to be developed.

b) Classify and mark the data that will be considered personal data,

c) for the data referred in b) 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.

d) Manually define and implement the queries to the database to make the data available to the user.

e) Establish an application logic responsible to ensure that personal data receives a different treatment, imposed by the GDPR, 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.

Analyzing the main tools currently available and used for the development of web applications, with the main objective of storing and processing data, it is possible to conclude the existence of a lack of tools that automate and facilitate the work of the programmer, in the enforcement of the new GDPR.

The current method of software development has several disadvantages and large-scale optimization problems with the implementation of GDPR related requirements. These problems are a result of the need for program development, of applications with data storage features, to be based and supported solely on the programmer’s work and knowledge of the regulations, for the storage of the data enforcing the GDPR. It is the software programmer who is also responsible for supervising the classification of data in personal data, which requires different treatment and storage, and public data. Making this process slow on a large scale and less reliable due to being subject to human error. This concept of
human error can be composed mainly of poor classification of data, treatment of personal data ignoring the regulations and necessary steps imposed by the GDPR and lack of request for consent or renewal of it during the time of data storage in the database.

1.2 Objectives

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.

With the development and implementation of this new layer it will be possible 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 ORM or Middleware, 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, through the relationships included in the relational databases.
- Provide differentiated treatment to data 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.

This layer should be able to automatically assess and distinguish between personal and non-personal data, mark the data and store it together with a group of metadata, necessary for data processing according to the GDPR. Simultaneously this middleware should make available to the programmer a set of functions in charge of manipulating the stored metadata or processing for storage the data classified as personal with the help of these metadata. For the features described above, this layer 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 these functions available to the user with the purpose of giving him control over his data.
1.3 Results

To implement this system it was necessary to study the behavior and structure of various frameworks used to develop software and web applications as well as the analysis of databases, used for data storage, and the relational links between data. In order to enforce GDPR regulations on the storage and processing of personal data.

For the development of the system created in this thesis it has been considered that most web applications are currently developed in an architecture called as multitier architecture, or tier N applications. This consists in an architecture of separated physical layers, each responsible for running 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 one can see in Figure 1.1. This architecture was studied in this project because it is considered as a strong, solid technique and it is very common in the web application developers community nowadays.

![Figure 1.1: System multitier Scheme](image)

In the system present in figure 1.1 it is possible to observe that the multitier architecture was implemented, this project focused more on Tier 3 represented in the figure. This tier is divided into layers with the different functionalities present in the application logic as well as the ORM.

In the elaboration of this thesis is considered the possibility of creating a layer that extends the ORM and works in parallel with it. In this hypothetical layer is found an API that will be formed by the functions explained in table 1.1 and made available to the programmer of the application that uses this system. This tool was called **GGML-Generic GDPR Management Layer**.

To exemplify the functioning of the concepts studied in this thesis and to collect practical data, a layer was created that extends the ORM present in SQLAlchemy, a python framework and sql toolkit. This layer contains an API that was developed with the previously described functionalities implemented for the use of the programmer, as well as the possibility of creating a file with the structuring of the objects in charge of data storage.
Functions | Definition
---|---
Show Functions | Functions that allow the programmer to query the saved metadata values, these metadata were created according to the GDPR and for its imposition in the implementation of web applications.
Change Functions | Functions that allow the programmer to manipulate and manage saved metadata.
Data management Functions | Functions responsible for handling and storing, according to the GDPR, data classified as personal, using the help of the previously stored metadata.

Table 1.1: Synthesis API functions

To further validate the proposed system a web application was developed, using this new extended ORM layer, as an example for its correct functioning and proof of the achieving of its main objectives. The application aims to demonstrate the development and implementation of a web application, with data storage functionalities according to the GDPR. It was possible to evaluate the benefits achieved for the programmer using the GGML.

With the implementation of this example system it was possible to observe several positive results. During its testing it was possible to notice that among the advantages of the implementation of this system was the decrease of the need for knowledge of GDPR regulations and operation of the ORM by the programmer, thus lowering the workload of the same and simplifying the remaining, and for the same reason lowered the system’s exposure to human error.

Moreover, by automating the classification of intermediate classes, in large cases, it was possible to observe an increase in data processing speed, without greatly increasing the memory used by the system. Finally it was possible to conclude that with the use of this tool it was possible to implement all the desired features and drastically decrease the lines of code to be developed by the programmer without a significant increase in overhead.

1.4 Thesis Outline

The second chapter describes the background of the problem that is being studied by giving a brief explanation of what GDPR consists of and what are the changes in the daily lives of companies with the changes in regulations present in GDPR. A summary of the actions taken by companies to work in accordance with the new regulations is presented, as well as a description of the technological solutions implemented today in order to help enforcing GDPR regulations on data storage and management.

In the third chapter of this thesis, a detailed description of the features included in the developed library is made, as well as the level of abstraction that will work and provide the software developer. The fourth chapter is a description of the programming environment in which it was developed as well as a presentation of the practical implementation of the system and its entire support structure.

The fifth chapter consists of a comparison between the developed system and the current method used to deal with the problem previously exposed and a set of practical examples that help to prove the concept of the idea presented in the thesis. A demonstration of the system operation was also made.
from the point of view of both the entity that is responsible for the processing and storage of data as well
as the user to whom the personal data belongs.

In the sixth chapter of this thesis is a discussion about the functioning of the developed library, its
practical applications, strengths and weaknesses. Finally, the last chapter summarizes the work done
during the thesis, as well as the hypotheses of future work to continue the work started in this thesis.
Chapter 2

Background

Since May 2018, the storage and processing of data within the European Union has undergone several modifications due to the enforcement of the new GDPR regulations. These regulations forced companies to deal directly with personal data collection and processing to revise their systems and standards for storage and processing of such personal data.

The systems currently implemented for the storage and management of personal data resort to manual classification and manual segmentation of the data. For this reason, these systems are very dependent on the knowledge and competence of the software developer or the person in charge of data processing thus decreasing the possible effectiveness of the system.

This chapter presents a detailed characterization of the changes suffered by the regulations and laws that govern and regulate the storage of data classified as personal in the European Union, and some examples of penalties imposed are also elicited, if the GDPR is not strictly followed by any entity. This chapter also includes a review on the most important tools that are available for the programmers to create or to adapt systems to the new GDPR rules. There is also a presentation on the current audit methods used as well as a description and representation of the main software architecture used in the creation of web applications oriented to the storage and processing of personal data.

2.1 General Data Protection Regulation

Within the GDPR guidelines, some basic concepts for the implementation of the rules for the storage and processing of personal data were also outlined. These concepts are:

- End User or data holder- Person or entity the data are referring to and the primary owner of the data.

- Database controller- Entity responsible for the processing of the data in the database and its storage.

- Data Protection Officer- Entity responsible for the application of the rules in the processing of data and its storage in databases.
• Personal Data- Set of information that contains a noticeable link between the data and the data
holder.

Another of the main purposes of this regulation is to give back to the users the control over their per-
sonal data records. With this purpose, users’ rights as well as the obligations of the database controllers
were regulated in chapter III and IV of the GDPR [1]. Some of these regulations are more related poli-
cies, and not necessarily implemented in the system / applications, while others have an active impact
on the implementation of the storage of personal data carried out by applications and other systems.

The following sections (2.1.1 and 2.1.2) present the changes in the regulations and rules of the GDPR
that had a direct influence on the method and requirements that are necessary to implement in a web
application to comply with the rules of the European Union. These changes are thematically grouped
into Data owners’ rights and Obligations of database controllers as it can be seen in the sections that
follow.

2.1.1 Data owners’ rights

The following list groups the main changes made to the GDPR [1] (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 [1] 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 informa-
tion 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 responsi-
able 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.

### 2.1.2 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...
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.

**2.1.3 Penalties**

According to the GDPR in article 83 [1], 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 into 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 [3]. Some examples of these penalties, that have their details and information public, are shown in table 2.1.

<table>
<thead>
<tr>
<th>Country</th>
<th>Description</th>
<th>Fine [Euros]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iceland</td>
<td>Insufficient technical and organisational measures to ensure information security and deletion of data outside the validity period established by the consent given by the data owners.</td>
<td>20,600</td>
</tr>
<tr>
<td>Spain</td>
<td>Non-compliance with general data processing principles, missing of consent requests</td>
<td>3,600</td>
</tr>
<tr>
<td>Greece</td>
<td>Insufficient fulfilment of data subjects rights</td>
<td>5,000</td>
</tr>
<tr>
<td>Spain</td>
<td>The company processed personal data of customers without required consent.</td>
<td>5,000</td>
</tr>
<tr>
<td>Italy</td>
<td>The company repeatedly sent advertising messages to a data subject, although the data subject had objected to the processing of his data.</td>
<td>6,670</td>
</tr>
<tr>
<td>Belgium</td>
<td>The company failed to act on requests from the data subject to get access to his data and to have his data erased.</td>
<td>2,000</td>
</tr>
<tr>
<td>Poland</td>
<td>No data processing agreement has been concluded with the company whose servers contained the resources with no limitation date</td>
<td>9,380</td>
</tr>
</tbody>
</table>

Table 2.1: GDPR Fines

As can be seen from Table 2.1, most of the infractions committed by companies consist of lack of consent from the data holder, improper processing of personal data and incorrect storage of personal data. All these problems can be avoided with the implementation of mechanisms and functionalities in software that help in the implementation of the GDPR rules.

2.2 GDPR supporting tools

At the beginning of 2018 studies and questionnaires [4] were conducted and 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 [5] 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. One of the tools developed for the case is the tool discussed in the article “Design Challenges for GDPR RegTech” [6], this tool performs a study on the level of GDPR compliance of an organization that processes and stores personal data.
The remaining of this section presents the list of main tools that provide help to programmers in the implementation of GDPR accompanied by a short description and explanation of its importance regarding the development and adaptation of systems to follow the GDPR. Most of the tools publicly available are plugins for CMS, a system that offers a variety of libraries that help programmers in content management, that is, it is a system that helps in the creation and editing of digital content in a dynamic way.

There were many evolutions and methods created for the imposition of the GDPR in websites and applications, such as plugins and add-ons, these plugins are a software component designed to modify behaviors or add specific new features to websites and applications, in this case to implement new application logic as a response to the new regulation. The main types of plugins and addons, found in internet content control systems are regarding to the following features:

a) Terms & conditions
b) Event and Communications Administrator
c) Audit administrators

Some of the specific plugins that deal with the themes mentioned above can be analyzed in the following chapters as well as the features they present to the programmer.

2.2.1 Data Flow Mapping Tool

Data Flow Mapping Tool [7] is an online tool developed by IT Governance [8] that creates the mapping of the organizations data flows and a personal data inventory with the goal of studying and controlling the data of their users. For this main purpose the tool has the following features:

- Creation and usage of a personal data inventory, consisting primarily of metadata, to log details of the personal data items involved in each process;
- Specify your lawful basis for processing, the types of personal data being processed and the categories of data;
- Generate a version-controlled data flow report that compiles information from your data flow map in an easy-to-read format to share with users;
- Update the process map and details whenever changes are made to the process.

This tool may, using some of the features previously described, address the needs created by the changes implemented by GDPR. The articles are as follows:

- **Sec 3, art 18-a** - In the case of personal data used outside the parameters consented by the users, mark this personal data in its personal data inventory in order to stop or limit their treatment
- **Sec 1, art 30** - This inventory of personal data can also store the main logs of the procedures performed on a set of marked data.
2.2.2 Data Protection Impact Assessment

Data Protection Impact Assessment[9] is a tool developed by IT Governance [8] that studies the risks that stored data suffers and assesses their possible repercussions to the data entity and its users. The online tool in question contains the following functions, significant for the GDPR theme:

- Systematic and extensive evaluation of data based on automated processing and on which decisions are based in legal effects concerning users;
- Large-scale processing and monitoring of special categories of data or personal data;
- Identify risks and determine the likelihood of their occurrence and impact;
- Share information with users and your supervisory authority
- Systematic monitoring of a publicly accessible area on a large scale.

When using the previously described functionalities, this tool can suppress the necessary data due to the modifications made in the GDPR. These needs are related to the following articles:

- **Sec 2, art 32** - This module is very suitable for analyzing the security risk of stored personal data and analyze the security measures implemented by the system in question.

Despite not making real-time notifications to users and entities responsible for storing data, reports are made on the large-scale processing of predefined categories and the break points of those treatments.

2.2.3 GDPR Manager

GDPR Manager[10] is a tool developed by IT Governance [8], which consists of a set of modules that aim to help the organization to impose the introduction of the GDPR in company applications that have data storage and processing functionalities. The most influential modules and features in the web applications are the following:

- **Gap Analysis module** - Assess your level of compliance with the personal information management system standard aligned with the GDPR and identify the actions you need to take to protect personal data in compliance with the GDPR.
- **Data Subject Access Rights (DSAR) module** - Keep a record of all DSARs received and how they have been followed up.
- **Breach Report module** - Record all data breaches and incidents affecting personal data, and streamline your data breach notification process.
- **Third Party Management module** - Keep track of all third parties your organization works with access to process personal data

Using the modules previously analyzed, this tool will mitigate the implementation needs in web applications created by the following articles:
• **Sec 2, Art 13-a** - A well-developed module, Third Party Management module, that creates a list of all the entities for which data has been assigned, for various reasons, and is responsible for make this list available to the data owners if requested.

• **Sec 2, Art 13-b** - This tool has a module, Data Subject Access Rights (DSAR) module, that lists all the requests for obtaining the data that is stored, as well as the result of those requests, but it does not have the functionality to resolve the requests that are on the list, not fully solving the problem.

• **Sec 2, art 33 and 34** - This system has a very precise module, Breach Report module, in keeping a record of all data breaches as well as performing the real time notifications to the data owners and storage entities. Even so, the full implementation of the articles is not done with the provision of mitigation steps for the consequences of data leakage.

There is also software outside these large companies, much of which open source, that produce digital management mechanisms and digital rights enforcers to help implement GDPR regulations, some examples of these plugins are:

### 2.2.4 Wordpress plugin gdpr

Wordpress plugin [11] gdpr is a software extension that implements the automatic consent management as well as the user’s right to transfer data to third parties and their data access and deletion. It also manages the notifications automatically made to the user in case of data leakage. The features used to obtain the results previously described are as follows:

- Consent and re-consent notifications management functionalities
- Privacy Preference management for Cookies with front-end preference UI & banner notifications
- Rights to erasure & deletion of website data
- Pseudonymization of user website data
- Front-end requests button to access data by Data Subject
- Right to portability & export of data by Admin or Data Subject
- Encrypted audit logs for the lifetime of Data Subject compliance activity
- Data breach notification logs to Data Owners

This plugin has several features to help the implementation of GDPR particularly in the following articles:

• **Sec 1, art 12 a** - One of the features of the system is a set of functions that help in the elaboration of requests for consent and sending to the users of the web applications.
• **Sec 2, art 13-b** - Makes a list of requests received through the front end button and send requested data.

• **Sec 3, art 20** - Makes a list of requests received through the front end button and send requested data.

• **Sec 3, art 17** - It has data erasure features to delete personal data if the user, owner of such data, makes that request to the system.

• **Sec 1, Art 30** - It has a list containing all the logs referring to all the processes that the data undergoes within the storage and treatment system.

• **Sec 2, art 34** - It has the necessary features for sending real-time notification messages from data owners, classified as personal, of the occurrence of information leaks.

### 2.2.5 DataDefender

DataDefender[12] is a software that identifies and analyzes data risks, implements data anonymization allowing to anonymize the sensitive data and transfer information between organizations, while reducing the risk of unintended disclosure. The main features present in this code extension are:

- Auto identification features of sensitive personal data;
- Creates plan to define what personal data and how should it be anonymized;
- Anonymizes the data;

This extension of features helps the programmer to implement the following GDPR articles.

- **Sec 2, art 32** - Analyzes what data should be anonymized and anonymizes data classified as personal.

It also identifies the data that is considered sensitive but does not perform the analysis of the treatment that these data undergo by the system, implementing only part of Article 35, section 3 of the fourth chapter of the GDPR.

### 2.2.6 Virgil-crypto

Virgil-crypto[13] is a high level cryptographic library that helps software developers to perform all the necessary modifications to their software in order to secure data storing and data transferring, in compliance with the GDPR regulations. For this purpose, this library provides the programmer with features that:

- Generate keys;
- Encrypt data;
- Decrypt data;
• Provide Sign data to the application store;

• Verify data safety;

This extension of features helps the programmer to implement the following GDPR articles.

• Sec 2, art 32 - The features in this plugin are excellent for protecting data sets classified as personal, through their encryption and secure decryption.

2.2.7 DBMS support

DBMS are systems that link end users and the database itself to take samples of data and process data in order to study the dataflow, this type of system is another widely used tool for implementing GDPR regulations and increasing the security of stored data is database management software and techniques. These techniques and systems are not dependent on the types of databases they are managing or the software on which they were created, so DBMS can be generally broke down into the following modules:

• Monitor[14] - Module that performs a real time assessment of the data base enforcing good security practices, identifies vulnerabilities, monitors the creation and usage of passwords, and oversees the backup of the metadata related to accesses of the data and of the data itself.

The inclusion of this module in the web application under development helps in the implementation of the following GDPR articles.

– Sec 1, art 32 - Introduces and helps to structure the need to implement data security and use passwords.

– sec 3, art 35 - Performs impact and vulnerability assessments in the last part of the data treatment and storage flow.

• Authentication[15] - External authentication module that integrates a centralized authentication infrastructure, a central account management, and defines a password policy management. This module suppresses the necessity to manages credentials in an individual level for the system modules.

The inclusion of this module in the web application under development helps in the implementation of the following GDPR articles.

– Sec 2, art 32 - It takes steps to ensure that the data is safe within the database and to prevent data leakage.

• Firewall[16] - Module that in real time protects the data base against cyber security treats using a whitelist, blocking SQL injections attacks and suspicious traffic. Is also responsible for intrusion detection, blocking and alerting the data storage entity for these events.

The inclusion of this module in the web application under development helps in the implementation of the following GDPR articles.
Sec 2, art 32 - It takes steps to ensure that the data is safe within the database and to prevent data leakage.

Sec 2, art 33 - It is also responsible for the possible leaks of information that occur and notifies the responsible entities

- **Audit[17]** - System that helps the data storage company to implement a better policy-based auditing solution and better security controls testing the workflow of the database in question. It also produces an audit trail of information to assess the database compliance level of the storage entity standards.

The inclusion of this module in the web application under development helps in the implementation of the following GDPR articles.

- **Sec 2, art 32** - This module is used in audits with the possibility of assessing the security of the system and its data flow.

- **Sec 2, art 35** - It is also used to assess the risk of operations carried out on certain data stored in the database.

- **Cloud Service** - Cloud based module that manages disaster and backup recovery of the data and logs related to certain data, classified as sensitive and stored in the database. Also owns and displays consultative support for de software developer and responsible entity.

The inclusion of this module in the web application under development helps in the implementation of the following GDPR articles.

- **Sec 1, art 30** - Module responsible for maintaining logs on some operations performed by the system on stored personal data.

With the help of these modules, it is possible to implement and impose various modifications to the personal data storage regulations created by the new GDPR, these modules can help programmers in implementing solutions to the data security problems created by GDPR, such as the monitor and authentication module. Programmers can also be helped to deal with problems of management and classification of personal data with modules such as the audit, previously presented.

### 2.2.8 Summary table

It is possible to observe in table 2.2 which GDPR requirements and needs can be met by introducing the systems, previously presented, in the applications to be developed.

With this study of the table 2.2 it is possible to verify that the functionalities used to mitigate the needs created by the implementation of the GDPR are spread over several different modules or plugins and that even with the use of all there are regulations that are not covered by the insertion of them in the systems in question.
<table>
<thead>
<tr>
<th>GDPR Article</th>
<th>Data Flow mapping tool</th>
<th>Data Protection Impact Assessment</th>
<th>GDPR Monitor</th>
<th>Wordpress Plugin</th>
<th>Data defender</th>
<th>Virgil</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sec 1, art 12-a</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1, art 12-b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2, art 13-a</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2, art 13-b</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 17</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 18-a</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 18-b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 19-a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 19-b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 20</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1, art 24-a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1, art 24-b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1, art 30</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2, art 32</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Sec 2, art 33</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2, art 34</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Sec 3, art 37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 4, art 37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: CMS/websites plugins summary table

<table>
<thead>
<tr>
<th>GDPR Article</th>
<th>Monitor</th>
<th>Authentication</th>
<th>Firewall</th>
<th>Audit</th>
<th>Cloud Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sec 1, art 12-a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1, art 12-b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2, art 13-a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2, art 13-b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 18-a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 18-b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 19-a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 19-b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1, art 24-a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1, art 24-b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1, art 30</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.3: DBMS support summary table.

Continued on next page
<table>
<thead>
<tr>
<th>GDPR Article</th>
<th>Monitor</th>
<th>Authentication</th>
<th>Firewall</th>
<th>Audit</th>
<th>Cloud Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sec 2, art 32</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Sec 2, art 33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2, art 34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 35</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Sec 4, art 37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It can be concluded, analyzing the table 2.3, that DBMS systems focus more on the implementation of GDPR on the side of the obligations of the entities responsible for data storage, which was expected because it is where the database is implemented. These implementations are applied mainly on the security side of the data stored in the database.

Analyzing the summary of these tables, it is possible to observe that there are GDPR requirements that are not implemented. It is also possible to conclude that there is no optimal tool that implements all requirements simultaneously, thus creating a void in the offer of tools designed to help the implementation of GDPR in web applications by their developers. Similar conclusions were found in the study linking current tools and the articles to be implemented in the GDPR carried out under the article "Transparency Enhancing Tools and the GDPR: Do They Match?" [18].

2.2.9 Current Auditing

This theme is very significant in the implementation and enforcement of the regulations presented in the GDPR, as the audit can be used to observe data flows within the system, thus proving and testing the correct classification of data as well as its treatment according to the GDPR. In this way, the audit functions as a test module for the good functioning and the correct imposition of the modifications of the regulations currently implemented.

Today's audits of web applications and databases consist primarily of studying the possible breakpoints of a system flow to prevent information leakage. For this purpose and to prove the proper functioning of the application in question it is possible to make a sample access to the database through the application, not mattering what data is accessed but focusing how the data flow through the system works.

This generally implemented audit system is classified as low level because, if we take into account the data pyramid (DIK pyramid) [19], where knowledge is ordered in different hierarchical layers as shown in the figure 2.1, these layers are the data layer, which is the base and first layer of the pyramid and is formed of symbols, facts and mathematical values, the information layer, structured, organized and processed data, and on the top the knowledge layer, conclusions drawn from these structuring and processing of the information, the current audit system only works mostly in relation to the creation and modification of data tables stored in the database, thus functioning at the level of the data and its processing. This leaves out all meta information regarding the accesses, the types of data accessed and the operations made with them as all the conclusions reached with those information.
In this way it is possible to observe that, failing to process the metadata on the accesses to the data as well as on the operations that were performed on the raw data, the audits currently implemented are of low level and very incomplete.

Although some techniques, tools and solutions have been developed for helping to enforce the GDPR regulation on web pages and applications, as the tool analyzed in “HS. Register–An Audit-Trail Tool to Respond to the General Data Protection Regulation (GDPR)”[20], they are very dependent on database controllers and software developers. For that very reason there is still a need for the development of a system that automates the application and enforcement of GDPR regulation in the processing and storage of personal data, thus removing from the programmer some of the responsibility.

**Information level auditing**

The software developer may recognize the need for the system to perform an external audit that focus mainly the GDPR regulations, this audit would consist in monitoring and analyzing the actions and executions made to the stored personal data that are treated by the system in question. That is, this audit module, unlike current ones that control data-level information, would control and store information at the level of queries made to stored personal data.

Therefore this external audit module shall be responsible by making a list of logs at the level of queries, this list will be composed of:

- **User’s** - Identification of all users who access data marked as personal or performed other operations on them.

- **Internal Operations** - Maintenance or management operations carried out on the database without the need for an event outside the database.

- **External Operations** - Operations carried out through the web application that affect stored personal data.
• **Data** - List of personal data affected by these operations.

• **Date and Time** - All points previously mentioned must be stored together with the date and time of the operations.

The lists resulting from this module will can be compared with the metadata obtained from the collection of the personal data in question by the applications or system so can be verified that all operations performed on the data are in accordance with the permissions granted by the data owners and all the accesses are in accordance with GDPR regulations.

### 2.3 Web application development Architectures

Currently software developers to create a web application in accordance with GDPR must develop an application logic that considers the problems created by the new regulation and solve or mitigate them with no external help. For this, the programmer must have specific knowledge of GDPR regulation, as well as choosing a type of software structural architecture that adapts to the resolution of the problem and is advantageous for the development of the application.

#### 2.3.1 Multitier architecture

Other of the patterns of structural systems architecture commonly used in the development of web applications is called multitier architecture [21], usually composed by three tiers. This model is composed usually by the following three main layers:

• **Client or presentation tier** - The client layer is the front end tier in this system and consists of the user interface. This user interface is often a graphical one accessible through a web browser or web-based application and which displays content and information useful to a user.

• **Application logic tier** - The application logic tier is responsible for data processing required by the application for later presentation.

• **data storage system** - The data tier compresses all the data storage and data access functionalities.

The typical 3 tier structure previously described must have the client layer deployed in the user device, the logic tier hosted in one or more application servers and the data tier in a big database system hosted in local or in the cloud. The main advantages that made this architecture extremely popular, in order to be used as a model in several systems, is the ease and speed of development, scalability and great performance, due to the division of system it is possible to develop each tier independently or to scale a tier taking into account its use and requests balancing the work load, augmenting the overall performance [22].
2.3.2 Model-View-Controller

One of the most popular structural architecture pattern of web applications, that benefit from most of the current functionalities and advantages, is called Model View Controller (MVC)[23]. This architectural pattern, defined in the year 1970 [24], MVC main objective is the reuse of code and separation of concepts in three interconnected layers, as shown in figure 2.2. These layers are:

- **Model** - Layer that manages the behavior of data through specific rules and remains in standby mode waiting for the call of functions that allow access, manipulation and storage of data. This layer is responsible for the connection, management and manipulation of the database implemented under the system in question.

- **View** - Code layer responsible for any representation and output of the required data to the Model layer, this layer also ignites reactions from the user, who then goes on to interact with the Controller, a simple example of this is the click of a button displayed by the view application. Traditionally in web apps built using MVC, the View is the part of the system where the HTML is generated and displayed.

- **Controller** - The last layer of this architecture is the controller, which is responsible for managing user inputs, sending commands to both the model and the view. The functioning of this code segment can be seen as a set of triggers initiated only by the interaction between the user and the view, then the controller will send commands to the model layer in order to update the data state, perform manipulation or access.

![Figure 2.2: Model-View-Controller scheme](image)

Many frameworks were developed based on this model, varying its interpretation, mainly in the way that MVC responsibilities are separated between the client and the server. Currently the approaches of MVC frameworks are those classified as thin client, where the two model and controller layers are
implemented on the server and the client only makes the connection between the application and the user, thus facilitating the introduction of multiple clients.

2.3.3 Object-relational mapping

Another particularly important mechanism currently used in web applications, with data handling and data storage functionalities, is the Object-relational mapping (ORM). The ORM is an additional layer implemented between the application logic and the database, as illustrated in figure 1.1, within the MVC architectural model this mechanism is contained within the layer called Model.

This layer is responsible for the conversion between the Object-oriented programming languages and the relational database. The programmer uses the objects and their methods to access the data but the storage of them is done in relational tables.

Since in the programming and development of web applications most logic is implemented using Object-oriented programming languages, but the data is stored using scalar values. It is necessary to implement a layer that, in addition to creating and managing the tables used in data storage, performs the necessary transformations to make the introduction of the values and information represented in objects in the tables created in the database, as illustrated in figure 2.3.

That is, the introduction of an ORM layer in the development of the web application makes a huge reduction in the amount of code needed and creates a higher level of abstraction, avoiding that the programmer needs to know all the code used in the previously mentioned conversions.

![Figure 2.3: Object-relational mapping operation](image)

There are several implementations of ORM that reproduce the behavior previously described, within these implementations there are overlaps and common points between the various architectures used. It is possible to see in figure 2.4 the components that are included in most of the implementations of the ORM layers, these generic modules are the following ones:

- **ORM API** - Set of functions or communication protocols that allows the application to use the
orm functionalities without having to have knowledge about the details of its implementation.

- **DML / Object-oriented programming language** - Module where the representation of the classes necessary for the implementation of the application and data collection is made, this representation can be done through data manipulation language or by making a direct reception of an object-oriented programming language.

- **Parser** - Component that loads the information and classes represented in the DML and checks the grammatical structure according to a formal grammar previously established.

- **Class repository + Class Mapper** - Module that based on a set of standard classes performs the mapping of the information present in the classes already analyzed by the parser and making the compatibility, converting in scalar values to be possible the introduction of information in tables implemented database.

- **ORM drivers** - Component responsible for the communication between the ORM layer and the database, communication made through session management, queries formation and their syntactic analysis and finally the execution of information transactions or their rollback.

If the programmer chooses to not use an ORM layer, it becomes necessary to manually convert the values and information contained in the objects into simpler values for introduction in the database tables, and convert them back in their access, or develop the application using only simple scalars values.
Chapter 3

Generic GDPR enforcement Layer

This chapter presents the requirements introduced by the GDPR that can be implemented as functionalities in web applications that handle personal data. This section is illustrated by a sample application, implemented with the technologies and methods commonly used for implementing web applications and to address the problems arising from the GDPR.

3.1 Motivation Applications

For this work the software architectures used were MVC and the multilayer architecture, previously presented, (the MVC architecture presented in 2.3.2 or the Multitier architecture presented in 2.3.1) the software developed is well divided in layers or modules according to its functionalities.

These software architecture structures are widely used since the programmers responsible for the development of these systems can use the advantages that come from this decision in order to facilitate their development. That is, the programmer can define the data model used by the system with the help of the ORM layer, presented in 2.3.3, later the programmers can develop the system back end, responsible for data processing, and finally the front end that makes the connection between the system or application and the users, presenting the previously processed personal data.

This previously referred front end of the system or user interface is the part of the application responsible for the interaction between user and database and is responsible for making available and presenting the data. For this purpose, this application layer must be programmed by the developer with the primary functionalities of the application in question as well as the necessary functionalities for the implementation of GDPR, the programmer should also be sure that the storage of data classified as personal, by the web application, is legal in the European Union, following and implementing the directives present in the GDPR.

Nowadays the development of a web application with the capacity to store and process personal data can be divided into two parts of the project, these parts are:

- **Specific features** - Set of features and structures that are developed and implemented to achieve the objectives for which the application was developed.
- **GDPR-related features** - Set of features and structures developed and implemented with the purpose of imposing directives and rules for the legal storage of personal data in the European Union.

Using the modern programming model, nowadays GDPR related functionalities are idealized and developed by web application programmers without any help or outside intervention, which leads to a high dependence on the developer's knowledge and no reuse of code.

In order to demonstrate the development of an application using the current programming model and the current software implementation problems, a simple practical example was planned and implemented. This practical example consists of a system of restaurant reviews, service ratings and comment issuing attached to data originated from a user’s visit to a restaurant establishment.

The data layer of this system mainly consists of the implementation of the three classes, shown in figure 3.1, their relational diagram and the management of the records formed by them. For the implementation of the relational flow present in the example of figure 3.1 with the current programming model, the programmer must be responsible for the creation of the three classes required in addition to the corresponding tables for the data storage.

Finally, the software developer is in charge of developing in the logic layer all the application logic responsible for the maintenance and processing of all data stored in the tables mentioned above.

![Figure 3.1: Diagram of the simpler practical example](image)

The second tier of the system described consists of a mobile web application with the main function of the system’s user interface, where the data is collected from the users for later storage and processing. Figure 3.2 shows the screen where the user is offered the possibility to classify a visit to a certain restaurant.

This user interface is also responsible for processing user requests to access stored personal data or data resulting from their processing, always following GDPR guidelines.

### 3.2 Current Adaptations of an application to GDPR

Nowadays it is also part of the programmer's responsibility to manually create a data structure, which lists the names of the personal classes and all the meta information that is required from the application
logic to retain about the personal data, to ensure that data classified as personal data is processed differently from public data, in the layer responsible for calling important functionalities for the treatment of data stored by the web application.

After these specific implementations, essential in all applications and systems developed, for the adaptation to GDPR enforcement. The programmer should be responsible for including in his application logic layer the following items:

- The creation of additional tables for storing metadata relating to the classification of personal or public data,
- Development and implementation of the functions necessary to control access and access to the stored data besides the creation of the pages to present the data taken from the database and its routing,
- Classification system for data stored, in personal or public data,
- Implementation of all functions needed for, using previously saved metadata, the implementation of European regulations and to keep all metadata up to date.

Another extension of functionalities necessary for this application is the inclusion of the user interface, shown in figure 3.3, that allows the management of the data collected by the application and belonging to the user, as well as the management of all metadata aggregated to them.

To achieve the implementation of the application, the software developer needs to build all the application logic related to data collection at the user level, implementing all communication with the database
and finally develops all the necessary functions for the user be able to manage the storage and processing of your personal data, as described in lists 2.1.1 and 2.1.2.

With the development of this sample system it was possible to conclude that a large part of the implementation of a data collection and processing system, according with the GDPR, depends on the programmer’s knowledge and competence. In this case, the developer of the web application in question needs to manually implement the functions necessary to achieve the application’s objective and all the necessary functionalities to correctly implement the directives presented in the GDPR.

It was also possible to observe that the manual classification of the personal data and its labeling as such, is made by the programmer or by the data administrator, opens space for the introduction of human errors when the number of data is of a high magnitude.

3.3 GDPR-related features

To make a system compliant with the GDPR, a set of features must be implemented. Some of requirements from the GDPR that can be implemented inside computer applications are presented in the following Table 3.1 and Table 3.2. These tables also describe how such requirements can be implemented in the applications to help address the needs created by the introduction of GDPR.
### 3.3.1 Data owners’ rights

Table 3.1 presents the rights of the owners of personal data, established in the GDPR regulations, in relation to their collection and treatment, as well as functionalities that can be implemented in software to guarantee these rights.

This table consists of two columns, the first containing the article belonging to the GDPR responsible for the modification and a small summary where the modification is explained and the last column describes features that, if included in the application, correct or mitigate the problems created by the new regulation.

**Table 3.1: Personal data owner rights from chapter III of GDPR.**

<table>
<thead>
<tr>
<th>GDPR article</th>
<th>System Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sec 1, art 12- a</strong>&lt;br&gt;The data controller or web application are required to provide the user with information on the reason for data collection, how long the data will be stored and other relevant information.&lt;br&gt;This information should be concise, written in plain and clear language.</td>
<td>The web application should provide a mechanism for the creation of a link between the user and the database where the necessary information.&lt;br&gt;A mechanism for the creation of forms, for obtaining consent to store the data, are automatically developed and sent.</td>
</tr>
<tr>
<td><strong>Sec 1, art 12- b</strong>&lt;br&gt;If the user requests information other than the available at the moment, the responsible entity should grant the requested information if relevant.</td>
<td>The web application should provide a mechanism for the creation and management of requests for the user request information not available to the public.</td>
</tr>
<tr>
<td><strong>Sec 2, art 13- a</strong>&lt;br&gt;During the collection of users’ records of personal data, the responsible entity shall provide the customer 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.</td>
<td>For each data record the web application should provide a report with the contact data from the data administrator to make it available to the user, if necessary.</td>
</tr>
<tr>
<td><strong>Sec 2, art 13- b</strong>&lt;br&gt;The data administrator should also provide the user with a copy of the collected data and a means of access to them while stored.</td>
<td>The web application should provide a mechanism to automatically present to the user with a copy of their data if he/she asks for it.</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>GDPR article</th>
<th>System Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sec 3, art 16</strong>&lt;br&gt;The owner of the data has the right to have his personal data, in a suitable period of time, modified or supplemented if he so wishes.</td>
<td>The web application should provide a mechanism that gives the user the possibility to request the change of the user personal data records. If the data has been modified, the secondary data resulting from its processing must be marked as out of date.</td>
</tr>
<tr>
<td><strong>Sec 3, art 17</strong>&lt;br&gt;Similarly to the last topic, the user is entitled to have his / her personal data records deleted, not affecting the results already obtained from previous data processing.</td>
<td>The web application should provide a mechanism that gives the user the possibility to request the deletion of the user personal data records. If the user data is deleted the results already obtained, with their previous treatment, can be marked but don’t need to be deleted also.</td>
</tr>
<tr>
<td><strong>Sec 3, art 18-a</strong>&lt;br&gt;The owner of the data may, in special situations of misuse of his personal data records, require that the processing of your data be terminated.</td>
<td>The web application should provide a mechanism able to distinguish and to terminate the treatment of a specific data set, in case of wrong use or use outside of the consent obtained.</td>
</tr>
<tr>
<td><strong>Sec 3, art 18-b</strong>&lt;br&gt;In the case of a previous limitation in data processing, this data can only be reused with a renewal of the user’s explicit consent.</td>
<td>The web application should provide a mechanism that, in the case of termination of data processing, request to the owner of the data the renewal of the consent. If the renewal of the consent is given the system should resume its treatment.</td>
</tr>
<tr>
<td><strong>Sec 3, art 19-a</strong>&lt;br&gt;The data administrator in charge of processing a user’s personal data is responsible for communicating, in the eventuality of a data transmission has occurred, to all recipients of the personal data transmission if the data was modified, deleted or the processing limited.</td>
<td>The web application should provide a report to notify the user about the modification or deletion of some data and, in case of a data transmission has occurred, to notify all the recipients of the data transmission.</td>
</tr>
<tr>
<td><strong>Sec 3, art 19-b</strong>&lt;br&gt;If the data owner requests the data processing entity, he / she is responsible for providing the data of the recipients of their personal data.</td>
<td>The web application should provide a mechanism that gives access to the data of the receivers of the data in question, if the data was transmitted by the application.</td>
</tr>
</tbody>
</table>
The data owner has the right to be provided with the personal data records that were stored in the entity in charge for their processing, in a structured and automatic reading format, if he requests the transmission of this data to a third party for processing.

The users always have the right to place an order to receive their data and the web application should automatically respond to the request.

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.

The web application should provide a mechanism that gives the user the possibility to request to cease the personal data treatment, in case of the user opposition.

The user should not be subject to any decision made solely based on automated processing of your data, including profiling from your data.

The data processing system of the web application should not make processing decisions of the user's data with base in profiling, should be always asked the owner of the data for his consent.

### 3.3.2 Obligations of database controllers

In table 3.2, it is possible to observe the main regulations, with practical applications at the software level, carried out by GDPR in terms of the responsibilities of the entities collecting, storing or processing personal data.

<table>
<thead>
<tr>
<th>GDPR article</th>
<th>System Functionalities</th>
</tr>
</thead>
</table>
| **Sec 1, art 24-a**
Given the purpose, cost in applying, the nature and context of the personal data to be processed, the web application with data storage features shall employ all technical measures to ensure the protection of such data as standard rule. | The web application should contain all the necessary mechanisms and techniques to ensure the security of the data stored by it. |
<table>
<thead>
<tr>
<th>GDPR article</th>
<th>System Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sec 1, art 24-b</strong>&lt;br&gt;The responsible entity should likewise 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.</td>
<td>The web application should provide a mechanism to ensure that data is only kept in the database long enough to be processed to the objective that the owner gave the consent.</td>
</tr>
<tr>
<td><strong>Sec 1, art 30</strong>&lt;br&gt;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.</td>
<td>The web application should provide a mechanism that maintains a list of all operations to the records of personal data in the database should be created and stored in parallel to the records of personal data, as well as a listing of all data metadata regarding the consent given by the data owners.</td>
</tr>
<tr>
<td><strong>Sec 2, art 32</strong>&lt;br&gt;It is the responsibility of the data storage system 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.</td>
<td>The web application should provide a mechanism that regularly tests data protection measures, such as encryption and pseudonymization of the personal data records, as well testing the level of security used to protect this data, while ensuring the user’s access to their personal data records in all times.</td>
</tr>
<tr>
<td><strong>Sec 2, art 33</strong>&lt;br&gt;In the event of breach of the confidentiality of stored data, the entity responsible for the treatment and storage of data must be notified 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.</td>
<td>The web application should provide a mechanism that notifies the authorities in the event of an information leak as well describing the consequences, and possible steps to mitigate the consequences of this event. The web application should also provide a mechanism that, if some data has been leaked, maintains a catalog with all the data records that were leaked.</td>
</tr>
</tbody>
</table>

Continued on next page
GDPR article | System Functionalities
---|---
**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. | The web application should provide a mechanism that, automatically and instantly, notifies the owners of the data in the event of an information leak and describe the possible steps to mitigate the effects of the occurrence.

**Sec 3, art 35**  
Prior to the processing of the data, the data administrator or 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. | The web application should provide a mechanism that maintains a list of operations classified and ranked by risk level so that high-risk operations only will be performed in last case and in a controlled environment, for the purpose of data protection.

**Sec 4, art 37**  
The data storage officer is also responsible for appointing the data protection officer or data administrator who should consider the professional qualities of the data protection officer as well as his or her data protection expertise. | The web application must have a different user, named data officer, the application must provide this user access to the actual data from backend tables created by the system with metadata values of the data so it can be used to help him to advise the data controller to enforce GDPR regulation. Making the classification and differentiation between personal and public data explicit and more intuitive for the software developer and for the data storage officer.

### 3.4 Generic GDPR enforcement System Functionalities

The system designed to help deal with the new requirements, created with the implementation of GDPR, for data storage entities, targets web applications that have an architectural model generically composed four tiers, each described in the section 3.1.

This system and its implementation are intended to help the software developer implement the GDPR regulation in the application he is developing. The system provides a mechanism that suppress the needs created by the regulation of the data owners’ rights and the obligations of database controllers by the GDPR.

If the applications were developed in multitier architecture based on the ORM layer, it is possible to automatically implement most of the features presented in the previous tables, delegating a large part of them to the ORM layer.
The system in web applications runs between the ORM layer and the layer responsible for the application logic execution, in this way this system will work as an extension of the ORM itself and the developed web application will be able to apply both the new system functionalities as well as the generic ORM functionalities. It is possible to observe in the architectural model of Figure 1.1 where this implementation will occur to the application developed get the most out of its features.

In a relational database it is possible to represent the classes that constitute them 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/ data holder manipulates one personal record at a time, or sets of related records, such as the show class data and forgets class data functions.

The main goal set for the development of this project was to simplify and automate the application of GDPR regulations. Because of this, and in order to make it easier for web application developers to work, this middleware provides a list of features and functions that are available for the developer to use, these functionalities are classified into two groups. The first group consists of passive features, necessary for data management (either during the programming of the web application or during its execution), they are:

- **Classification mechanism** - mechanism implemented that allows the developer of software to distinguish personal data classes from public data classes and label them, the mechanism in question is only required for classes considered as roots of the relations implemented by the programmer. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:

  – Sec 3, art 18-a
• **Auto labeling and classification system** - mechanism which, considering the classification of the roots made by the application developer, automatically classifies and labels the descending related classes. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:

  – Sec 3, art 18-a

• **Auto dating function** - If the class is considered a personal class, either by classification of the programmer or by automatic marking, whenever data records of that class are made, the collection date is saved and combined with the data. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:

  – Sec 1, art 24-b

• **Log Operation** - List that contains all the operations performed by the application to personal data records, as well as information regarding the moment in which they were performed. These features can be adapted to collect other processing operations outside the library as well as to carry out a risk classification for the personal data of the operations. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:

  – Sec 1, art 30

The second group of functionalities consists of functions that are only executed if an event or user input previously occurs. These functions will be used in the implementation of graphical interfaces accessible by several users and which will allow access by users to stored data. These features, previously referred, are:

• **Show Operation Log** - Function that provides a list of logs, which contains all the operations performed on a set of personal data records as well as dates related to the operations. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:

  – Sec 1, art 12-a

  – Sec 1, art 30

• **Show expiration date** - Function that receives a personal class and returns:

  – The value in days that determines for how long a data record of this class can be kept in the database,

  – If the value of the arguments is empty, the function performs for all classes contained in the list of personal classes.

This feature implements the following GDPR articles presented in tables 3.1 and 3.2:

  – Sec 1, art 12-a
• **Change expiration date** - receives a personal class and a value and modifies the expiration date value to this new value. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  – Sec 3, art 16

• **Show goal** - Function that receives a personal class and returns the goal of the data, this value is required for the GDPR implementation and attests the purpose for which the data was collected as well as the consent obtained from the data owner obtained by the system, if the value of the argument is empty, the function performs for all classes contained in the list of personal classes. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  – Sec 1, art 12-a

• **Change goal** - receives a personal class and a value and modifies the goal value to this new value. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  – Sec 3, art 16

• **Show category** - Function that receives a personal class and returns the type of the data, the data is subdivided into the following types:
  – Direct data - which can directly identify the person concerned,
  – Indirect data - which with a set of different records can lead to the identification of the data owner,
  – If the value of the argument is empty - the function performs for all classes contained in the list of personal classes.

This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  – Sec 1, art 12-a

• **Change category** - receives a personal class and id and a value and modifies the category value to this new value. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  – Sec 3, art 16

• **Show data owner** - receives a personal record ID and returns:
  – the owner of the data- value that stores the person or entity that has rights over the data to be processed,
  – if the value of the argument is empty- the function performs for all classes contained in the list of personal classes.

This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
- Sec 1, art 12-a

- **Change data owner** - receives a personal class and ID and a value and modifies the owner value to this new value. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  - Sec 3, art 16

- **Show data source** - receives a personal class and returns the origin of the data. Value that keeps the origin or details of the data collection, if the value of the argument is empty, the function performs for all classes contained in the list of personal classes. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  - Sec 1, art 12-a

- **Change data source** - receives a personal class and a value and modifies the origin value to this new value. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  - Sec 3, art 16

- **Clean** - Function that receives a personal class and, by checking its expiration interval and its storage date, erases all data from that class that is out of date, if the value of the argument is empty, the function performs for all classes contained in the list of personal classes. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  - Sec 3, art 17
  - Sec 3, art 24-b

- **Stop processing** - Function receives an identifier from a personal record and puts the data of that personal record in a waiting state, by order of the user, in which it cannot be processed. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  - Sec 3, art 18-a

- **Retake processing** - Function receives an identifier from a personal record and see if this record is in the waiting state, if so it puts the record in an active state by order of the user. In this way, the system can resume processing the data found in this record. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  - Sec 3, art 18-b

- **Data out of date** - function that, when receiving a personal class, returns a list of objects that are out of date for that class. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  - Sec 1, art 24-b
• **Personal list** - Function that makes a query to the system in which the answer is a list composed of all classes classified as personal. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  – Sec 1, art 12-b

• **Is personal** - Function that receiving a class examines whether it was classified as personal data, by the programmer or system, without accessing its data maintaining the data owner’s right of confidentiality. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  – Sec 1, art 12-b

• **Alert empty** - A function that, when receiving alerts the programmer if any of the common fields in the personal classes are unfilled. These common values are expiration interval, goal and category. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:
  – Sec 1, art 12-b

• **show class data** - Function that given a personal class and a unique identifier, to singularize a specific record, returns:
  – all the data in that primary record,
  – a set of records or list constituted of all values of data that are in the personal records that are directly and indirectly related to the primary record.

Being this relationship restricted to parent child direction.

As it is possible to analyze in figure 3.4, where the performance of the show class data function is shown, first the function receives the record identifier with the number 1 and labels it as a primary
record. The function then searches for and returns the records directly and indirectly linked to that record, creating a list with the data contained in records 2, 3, 4 as well as the data from the primary record. This feature implements the following GDPR articles presented in tables 3.1 and 3.2:

- Sec 1, art 12-a
- Sec 1, art 13-b
- Sec 3, art 20

**forgets class data** - Function that receives a personal class and a unique identifier, to singularize a specific record, and:

- deletes all data in the primary record,
- deletes all the data from personal records related to that primary record (starting in the lower levels of the relationship tree).

The general operation of this function is remarkably similar to the operation of the search carried out in the show data class function, previously described. The set of data described in the second point can be only marked and not deleted, if they do not maintain a direct link with the deleted personal data.

This feature implements the following GDPR articles presented in tables 3.1 and 3.2:

- Sec 3, art 17
- Sec 13, art 18-a
- Sec 1, art 24-b

Table 3.3: GDPR enforcing assessment by GGML.

<table>
<thead>
<tr>
<th>GDPR article:</th>
<th>GGML Functionalities:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter III</td>
<td></td>
</tr>
<tr>
<td>Sec 1,art 12- a</td>
<td>Show Operation Log, Show expiration date, show category, show class data</td>
</tr>
<tr>
<td>Sec 1,art 12- b</td>
<td>Personal list, alert empty</td>
</tr>
<tr>
<td>Sec 2,art 13- b</td>
<td>show class data</td>
</tr>
<tr>
<td>Sec 3,art 16</td>
<td>change expiration date, change goal</td>
</tr>
<tr>
<td>Sec 3,art 17</td>
<td>forgets class data</td>
</tr>
<tr>
<td>Sec 3,art 18- a</td>
<td>Classification mechanism, Auto labeling and classification system, stop processing, forgets class data</td>
</tr>
<tr>
<td>Sec 3,art 18- b</td>
<td>retake processing</td>
</tr>
<tr>
<td>Sect 3,art 19- a)</td>
<td>show funtions</td>
</tr>
<tr>
<td>Sec 3,art 20</td>
<td>show class data</td>
</tr>
<tr>
<td>Sect 3,art 21</td>
<td>stop processing</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 3.3: Functionalities of GGML and GDPR Articles

<table>
<thead>
<tr>
<th>GDPR article:</th>
<th>GGML Functionalities:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter IV</td>
<td></td>
</tr>
<tr>
<td>Sec 1, art 24- b</td>
<td>Auto dating function, Clean, Data out of date, forgets class data</td>
</tr>
<tr>
<td>Sec 1, art 30</td>
<td>Log Operation, Show Operation Log</td>
</tr>
<tr>
<td>Sect 3, art 35</td>
<td>Log Operation, Show Operation Log</td>
</tr>
</tbody>
</table>

Table 3.3 summarizes the functionalities of GGML and the articles of the GDPR that help with implementation. Table 3.4 gives an overview of the gaps still present in the system and the reasons for them.

### Table 3.4: GDPR Requirements Out of Scope of GGML

<table>
<thead>
<tr>
<th>GDPR article:</th>
<th>Reasons for Not Implementing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter III</td>
<td></td>
</tr>
<tr>
<td>Sec 3, art 19- a</td>
<td>Functionality associated with DPO, functionality related to notifications and information spreading by users.</td>
</tr>
<tr>
<td>Sec 3, art 22</td>
<td>Outside the context of the thesis as it is necessary to study the user’s self profiling to avoid its use.</td>
</tr>
<tr>
<td>Chapter IV</td>
<td></td>
</tr>
<tr>
<td>Sec 1, art 24- a</td>
<td>Outside the context of the thesis as it is a more relative topic to the security of stored data and databases.</td>
</tr>
<tr>
<td>Sec 2, art 32</td>
<td>Outside the context of the thesis as it is a more relative topic to the security of stored data and databases.</td>
</tr>
<tr>
<td>Sec 2, art 33</td>
<td>Outside the context of the thesis as it is a more relative topic to the security of stored data and databases.</td>
</tr>
<tr>
<td>Sec 2, art 34</td>
<td>Outside the context of the thesis as it is a more relative topic to the security of stored data and notification of its owners.</td>
</tr>
<tr>
<td>Sec 4, art 37</td>
<td>Article without computational inclusion, choice of DPO.</td>
</tr>
</tbody>
</table>
Chapter 4

GGML-GDPR Management Layer 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 described in section 3.4. This implementation and practical tests were carried out as proof of concept for this thesis.

4.1 Programing Environment

To demonstrate the concepts idealized in this thesis as well as to prove their successful implementation, one ORM was selected. The **Generic GDPR Management Layer (GGML)** was 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.

To develop this thesis, other ORMs for Java and Python were studied (FENIX framework, Django, Spring) but due to the simplicity of adding new features without changing the base code, it was decided to use the ORM from SQLAlchemy [25]. SQLAlchemy, which is an ORM library developed in Python[26] and was chosen 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 is mainly made up of two components called core and ORM, as shown in figure 4.1. 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** - Makes a list of specific database connections
- **SQL Language** - Translates Python into SQL statements
- **Schema/Types** - Represents tables, data types and columns in Python objects.
The SQLAlchemy ORM performs the construction of Python objects, mapping with relational tables, allowing the persistence of these objects. The Layer GGML was developed over the core previously explained, thus using the core to generate SQL and commands to manage the database and data.

For the development and implementation of the **Generic GDPR Management Layer (GGML)** system, a new layer of higher level functions was created. This layer was implemented on top of the original ORM which has not undergone any modifications, thus allowing the use of this system regardless of the ORM version used and maintaining the compatibility with previous versions of it.

```python
class Example_class(Base):
    __tablename__ = 'example_class'
    Id = Column('id', Integer, primary_key=True)
    Info = column('info', String)
```

Listing 4.1: SQLAlchemiy Class model example

The listing 4.1 presents one example of a Class model defined in SQLAlchemy. For the regular use of this original version of SQLAlchemy ORM it is necessary to introduce a field, called `tablename` (line 2), and the indication that the class in question inherits the properties and methods from the superclass called `Base` on line 1 (superclass implemented in the ORM) in addition to the class arguments. The `Base` superclass is used as an ORM feature used to convert the class to a table, with the name given in `tablename`, for storing the data in the database and the class arguments in table columns where the values of the data will be stored. This implementation example can be seen in figure 4.1

In addition to the chosen ORM, the application programmer must be responsible for choosing a web framework for implementing their web application. In the case of the implementation of the practical example described in this thesis, the Bottle framework [28] was chosen, a micro-framework also developed in python, to create an Application Programming Interface to support the application, this choice was due
to the simplistic nature of the framework and not having more dependencies than Python Standard Library.

4.2 System implementation

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. As you can see in figure 4.2, a metaclass was developed that attaches the classes developed for the web application by the software developer in order to trigger the operation of GGML.

![Figure 4.2: GGML layers](image)

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 back end 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 back end tables and the metadata stored in parallel with the personal data.

4.2.1 Metaclass

An important part of Generic GDPR Management Layer (GGML) was the implementation of a metaclass, that must be included in the definitions of the classes created by the programmer for the functioning of its web application. This metaclass includes all the code necessary to initialize the fields required for the processing of the application code. It is executed before classes are loaded and processed by SQLAlchemy.

Listing 4.2 presents the use of the GGML metaclass in the definition of the data models.

```python
1 class GGML_class_model(Base):
2     __metaclass__ = GGLMetaClass
```
The programmer must only insert the definition of the metaclass as shown on line 2, everything else remains the same as on regular SQLAlchemy.

With the inclusion of these directive metaclasses, their metaclass code runs at the same time as the creation of the classes and before SQLAlchemy processes the classes, making all classifications and modifications of the class to implement GDPR without conflicting with the functionalities present in SQLAlchemy. In this way the current ORM can be used in the implementation of GDPR without changing its functionality. Using the metaclass code as a complement in the storage of metadata in parallel with the data to be processed.

```python
class CustomMetaClass(type(Base)):
    def __new__(meta, name, bases, dct):
        if PersonalData in bases:
            personalClasses[name.lower()] =
            creategraph_lista(name.lower(), dct)
            aux_tag = Personal_tag_router_listas(name.lower(), grafo_lista, personalClasses)
            bases, dct = Personalization(name, bases, aux_tag, personalClasses, dct)
            if PersonalData in bases:
                dct['__original_init__'] = dct['__init__']
                dct['__init__'] = GGML_personnalInit
                consents = search_forms
                Personal_meta_global_initiator(self, consents)
                print("bases after"+str(bases))
                print("\n\ndct after\n\n"+str(dct))
        return super(CustomMetaClass, meta).__new__(meta, name, bases, dct)

def __init__(cls, name, bases, dct):
    print '-----------------------------------'
    print("Initializing class\n", name)
    super(CustomMetaClass, cls).__init__(cls, name, bases, dct)
```

Listing 4.3: GGML modified metaclass

It is possible to analyze the structure of the code and meta code for initializing the system and application classes, as can be seen in Listing 4.3. The overall functionalities of the metaclass are the following:

1. Pre-classification of the type of class (personal or not) following the steps described in section 4.3 (lines 3-7).

2. Modification of the class constructor if it is classified as personal (to include all the run-time code required during the application execution) as described in section 4.3.3 (lines 8-10).
3. Collection of metadata common to all objects created by the same class (line 11).

4. Creation of backend tables where metadata related to personal classes will be stored and storage of this metadata (line 12).

5. SQLAlchemy creates the tables to save data of objects created by application classes. After these initialization steps of the system and its tables, the data will be labeled in accordance with the classification of its class and combined with metadata related to the object. This classification will result in the binding of the records, resulting from this class, with a label and a date of collection, as well as the alteration of the `init` method of the class so that it is possible to create and initialize to become responsible for the creation and initialization of the fields of the metatable with common data between personal classes, each time a personal record is created.

With this information attached to the records, GGML provides the programmer with a mechanism that easily classifies 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.

### 4.2.2 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 data 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.

To support the functioning of the implemented system, it is also necessary to store some metadata common to all data records of the same class. The fields of metadata referred to are shown in table 4.1 and are saved in a back-end table stored in the database in parallel with the other tables, but without users having access to it, except when using the functions provided by the developed library.

The metadata used in this project can be classified into two generic groups:
• Global metadata (class specific)- Metadata for the classes in question with information about their classification, processing procedures and information common to all objects of the same class.

• Specific metadata (object specific)- Metadata for the classes in question with information about their classification, processing procedures and information common to all objects of the same class.

<table>
<thead>
<tr>
<th>Metadata Categories</th>
<th>Definition</th>
<th>Metaclass Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expiration Interval</td>
<td>Value in days that determines how long a data record of this class can be kept in the database without renewal of consent</td>
<td>Class Specific.</td>
</tr>
<tr>
<td>Data Goal</td>
<td>This value attests to the purpose for which the data was collected as well as the consent obtained from the data owner obtained by the system.</td>
<td>Class Specific</td>
</tr>
<tr>
<td>Data Categories</td>
<td>This value subdivides the data into two types, direct data, which can directly link the data to the person concerned, and Indirect data, which with a set of different records can lead to the identification of the data owner.</td>
<td>Class Specific</td>
</tr>
<tr>
<td>Data Owner</td>
<td>Person or entity that has rights over the data to be processed.</td>
<td>Object Specific</td>
</tr>
<tr>
<td>Data Source</td>
<td>Value that keeps the origin or details of the data collection. This value can refer to whether the data was collected by the entity itself or received from a third party.</td>
<td>Object Specific</td>
</tr>
</tbody>
</table>

Table 4.1: Metadata Categories

4.3 Personnal classes

The GDPR, a defines and makes a standardization of the concept of personal data, this definition consists of information relating to a single person identified or identifiable through these data. This means that it is data in which it is possible to make a link between that data and a person, that link can be classified as direct if there is a direct and unique identifier or indirect if there are several data that together lead to the identification of a person.

In an application data this classification of data in personal data occurs with the agglutination of personal data with a tag and an expiration date, in this tag it is possible to observe whether the data was classified by the programmer or automatically by the system, and with the storage of the metadata necessary for data processing of personal data as the GDPR imposes. In case the system was responsible for the automatic classification of the data it is possible to extract the root's name positioned closer and therefore responsible for the classification of this data in personal data.

4.3.1 Labeling of personal classes

For the programmer of the application under development to use the classification and manual marking functionality of the Root classes as personal classes, he should indicate as meta-class the GGML_MetaClass, as exemplified in figure 4.4, class implemented in the new library extension, this implementation is described in chapter 4.2.1. In the examples of listings 4.4 and 4.5 it is possible to observe that the introduction of the metaclass is common in both GGML classes models, even if for different reasons
and functionalities, while the introduction of the **Personal_data** class as a superclass makes the class manual marking as a personal Root, necessary for the correct functioning of the implemented system.

```python
class GGML_labeled_class_model(Base, PersonalData):
    __metaclass__ = GGML_MetaClass
    __tablename__ = 'GGML_labeled_class'
    Id = Column('id', Integer, primary_key=True)
    Name = column('name', String)
```

**Listing 4.4: Personal class labeling**

```python
class GGML_public_class_model(Base):
    __metaclass__ = GGML_MetaClass
    __tablename__ = 'GGML_public_class'
    id_r = Column('id_r', Integer, primary_key=True)
    name = column('name', String)
```

**Listing 4.5: Public class implementation**

After the introduction of the metaclass in both class models that GGML processes, the programmer is responsible for the introduction of the **Personal_data** super class that marks the class as containing personal data. This class is made available and implemented in the core of the system developed in this thesis, and its introduction is exemplified in Listing 4.4.

This introduction creates a differentiation between the two class models so that the system recognizes which of the two contains personal data and changes its storage, with the creation and merging of metadata, and its processing. This differentiation also allows the system to carry out a later classification of classes that are not roots of the application.

### 4.3.2 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 4.3, 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.

- **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.
In the system presented in this thesis the classification of data into personal and public data can be done by manual marking by the programmer in the case of the root classes of the application, as demonstrated in the previous chapter, or automatically by the system. For the system to be able to classify intermediate data classes, the programmer must first manually tag the personal root classes of his application and then the system automatically classifies intermediate classes as personal, using the meta-class present in all application classes, if they are relationally linked to a root classified as personal by the programmer, as exemplified in figure 4.4.

In the simplified example of an application’s UML it is possible to analyze that one of the classes is considered as Root and the other as an intermediate class, according to the classification system present in chapter 4.3.2, so if the application programmer has previously marked the Person class as Root the system will study its relationships, using graphs, eventually concluding that this class has a direct relationship with the Checkin class.

After the system proves the relationship between the classes, the intermediate class is automatically marked by the system, with the storage of the class metadata with a view to its treatment according to the current legislation. This classification is performed since the class may have data that indirectly
leads to the identification of the person who owns records of the person class.

With class marking the system performs a two-field binding, expiration date and label, these values will be set on all objects created from this class with the label value initialized with the name of the class which is the reason for its classification as personal, in this case the label will have the value of person. Finally, the system searches the stored relationships of the intermediate class to search if there are more classes in the same conditions to be marked, thus marking intermediate classes in a cascade effect.

4.3.3 Personal Class GGML Constructor

With the marking of a class with the GGML meta class, a new constructor is associated with each class, which runs when the objects are created and which manages the specific information of each object.

```python
def __GGML_personnalInit__(self, *args, **kwargs):
    Personal_tag_router(self)
    self.created_date = datetime.now().replace(microsecond=0)
    Personal_meta_initiator(self)
```

Listing 4.6: GGML modified builder

The code for this constructor, shown in listing 4.6, consists of the following steps:

1. Metaclass class studies the relational database in order to find out if the class is classified by GGML and the classifications are inferred.

2. The association of a new modified constructor made available by the GGML layer.

3. Study of the relationships kept between the database tables to associate with an object which is the root class that leads to its classification as a personal class (line 2).

4. Determination and association of the object's creation date,(line 3).

5. Initialization of values that are linked to each object created by the application, according to the consent forms obtained from the data holder (line 4).

4.4 Application Programming Interface

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:

- 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.

The set of auxiliary tables and data structures were created in section 4.2.2, can only be accessed by the programmer using the developed API. For this, after the system already has the information of which classes are classified as personal, the software developer should call the functions and features available in the API to facilitate the implementation of GDPR standards. Examples and explanations of the use of the library extension developed to make the connection between the programmer and the system developed in the thesis, some examples are shown in table 4.2.

<table>
<thead>
<tr>
<th>Function Code Example</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>show_expiration_interval(Class)</td>
<td>Function that receives a class name and performs a database query about the class expiration interval in the backend table and returns its value in days.</td>
</tr>
<tr>
<td>change_expiration_interval(Class, value)</td>
<td>Function that modify the expiration interval value stored in the backend table to the value received by this function.</td>
</tr>
<tr>
<td>stop_processing(Class, Index)</td>
<td>Function that receives the necessary inputs to individualize a record and makes an addendum to its label so that the application differentiates the data in order to help prevent its processing.</td>
</tr>
<tr>
<td>personal_list()</td>
<td>Function that queries the metatable present in the database, returning the list with the names of all classes marked as personal, both by the programmer and by the system.</td>
</tr>
<tr>
<td>show_class_data(Class, Index)</td>
<td>Performs a database query about the class data with the index as primary key, making a listing of this data and all data of the related classes descending to the concerned class.</td>
</tr>
<tr>
<td>forgets_class_data(Class, Index)</td>
<td>If receives a class and an index as an argument and queries the database about the record with that index and belongs to that class, subsequently listing all the personal records related to that particular object, finally deleting that object and, depending on the programmer's choice, it deletes all records contained in the list or marks them for later analysis and deletion. Because results obtained from forgotten personal records can be maintained if there is no direct link between them and the person to whom the personal data belongs.</td>
</tr>
<tr>
<td>Data_out_of_date(Class)</td>
<td>Performs a query to the meta-table about the value of the expiration interval of the class and then lists all existing class records in database out of date.</td>
</tr>
<tr>
<td>clean(Class)</td>
<td>Clean all the data expired of the table with the class name</td>
</tr>
<tr>
<td>is_personal(Class)</td>
<td>Function that tests the class to see if it is a personal or public class</td>
</tr>
<tr>
<td>alert_empty()</td>
<td>Backend function that sends a warning to the developer if there is an empty or badly populated meta table field</td>
</tr>
</tbody>
</table>

Table 4.2: Function calls and their results

In the functions presented in table 4.2 the programmer can send classes to make queries at a higher level or indices to make queries at the level of a singular record.

As the system already has all information regarding data classification in addition to access to all metadata related to that data the library allows the programmer to use the features provided by the library
to access and manage the information provided and only needs to iterate the information received from the functions to present to the user. These features, also described in section 3.4, can be differentiated into two sets, utility functions that are used in specific situations by programmer and system support functions, both sets have examples in table 4.2.

Examples of the functions provided by the library and considered as utility functions are the *show* and *change* functions and their derivatives, one for each category of metadata saved due to GDPR, which are functions implemented for the programmer to be able to access and manipulate the values of the metadata that are saved by the system in the backend tables. With these functions, the knowledge needed to use the system is considered to be of a high level of abstraction because the programmer does not need to have detailed knowledge about the implementation of the memory elements used to support the system.

Another example of a function developed in the library with the goal of lower the work of the programmer and facilitating the implementation of GDPR in web applications is the function of *showclassdata* where the programmer only needs to call the function with a class and primary key as arguments, for the system to return all data relating to that primary key as well as data from all personal classes related to it, making the exchange between the primary keys of the classes in the query. Therefore, it is only necessary for the programmer to create and implement functions to iterate the received data, thereby lowering the programmer’s workload as well as the required level of knowledge about the functioning of the database and GDPR regulations.

The other functions attached in the second set have as main objective to help the implementation of the system developed by issuing notices and carrying out cyclical tasks. One such example is the function *Data out of date()* this function is responsible for, after receiving a class as an argument, looking for the expiration interval and, taking into account the current date, compare with the storage date value of all records of the class in question. Finally, this function returns a list of all the records that were out of their expiration date, this function can be manipulated in a way and send a warning if the expiration date is close.

The function *Clean()* is very similar to the previous function, having all the same initial steps, but as a result it does not return any list of records but deletes all the invalid records of personal classes from the database. Finally, the *Is personal()* and *alert_empty()* functions are the main functions responsible for issuing warnings that help the programmer to implement the system and to fill in the backend tables without having knowledge about its implementation.

These features and functions become very useful for the software developer because, in case of different classes or even different data configurations, the programmer will only have to call the functions provided by the library and change their arguments to create applications in accordance with the modifications implemented by GDPR.

In figure 4.7 it is possible to observe the implementation of a simplified functionality, made available by the GGML library, which gives access or control to the metadata the programmer previously stored in the backend tables created by GGML. This functionality in the classic model would have to be implemented by the application programmer and with the use of this system it is already available for use.
def clean(value):
    Session = sessionmaker(bind=engine)
    session = Session()
    Personal_classes = session.query(Metatable.validity).filter(Metatable.l_pessoal==value.\_\_tablename\_\_)
    for data in Personal_classes:
        date_value=data.validity
        date=datetime.now().replace(microsecond=0)
        session.query(value).filter((value.created_date)<date-timedelta(days=date_value)).
        delete()
    session.commit()
    session.close()

Listing 4.7: Clean function (simplified)

In Listing 4.7 it is possible to observe the functioning of a function included in the GGML layer, in lines 2, 3 and 8, 9 it is possible to observe the management of communications with the database. In lines 4, 5 there is a loop where the validity interval values are searched, in line 6 the current date is saved. Finally, line 7 deletes all data from the record if the creation date is past the current date minus the expiration interval.
Chapter 5

Validation / Evaluation

This chapter has as main objective the demonstration of the Implementation of the concepts studied in this thesis in order to prove the good functioning of the system, through the development of practical examples where they are used.

5.1 Example Application

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 1.1 and a data and relational structure sufficiently complex to study the functionalities provided by the system efficiently.

![Figure 5.1: Restaurant rating Web application data template](image)

For testing the functioning of the library and system, developed during the course of this thesis, the structural model in figure 5.1 was used, which corresponds to a more complete practical example than the one shown in figure 3.1. This example in addition to collecting and storing, according to the
Web application classes

For the design and implementation of the application the following classes were implemented using the **Generic GDPR Management Layer (GGML)** system and its functionalities.

```python
class Person(Base, PersonalData):
    __metaclass__ = GGML_MetaClass
    __tablename__ = 'person'
    Id = Column('id', Integer, primary_key=True)
    Name = column('name', String)
    Email = Column('email', String, unique=True)
    Checkin_p = relationship("Checkin")

Listing 5.1: Person class implementation
```

```python
class Checkin(Base):
    __metaclass__ = GGML_MetaClass
    __tablename__ = 'checkin'
    id_c = Column('id_c', Integer, primary_key=True)
    id = Column(Integer, ForeignKey('person.id'))
    id_r = Column(Integer, ForeignKey('restaurant.id_r'))
    description = Column('description', String)
    rating = Column('rating', Integer)
    grade_c = relationship("Grade")

Listing 5.2: Checkin class implementation
```

In these figures with code, it is possible to observe that the programmer used the automatic marking classification mechanism available in GGML, as shown in section 4.3.1. The programmer only marked two classes as personal in listings 5.1 and A.1, the rest of the classes the system either classified automatically as personal or maintained its classification as public, reproducing the data structure drawn in figure 5.1.

In Annex A, it is possible to observe and analyze the classes used in the implementation of the restaurant classification application. These classes are very similar to the examples shown in listings 5.1 and 5.2.

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 5.1.

In this practical example, shown in the mockups represented in 5.2, the records collected by the logic displayed on the screen in point b) are classified as personal records under the influence of the root 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.
class person, and belong to the user who performed the classification, the records stored by the screen in the point c) are also personal but they can belong 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.

![Figures 5.2: Templates for input information from users](image)

5.2 Comparison with classic programming model

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

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 should implement all functions related to the GDPR.

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 therefore the responsibility of verifying which date to comply with the GDPR rules is no longer part of the programmer’s responsibilities during the development of the application, leaving only the programmer’s obligation, when designing the data classes, to mark the classes classified as personal.
5.2.1 Definition of personal data

In the classic model of programming in which the programmer is responsible for developing a system for classifying data from scratch, with the GGML the programmer has to introduce the metaclass provided by the library (GGML.MetaClass) and mark the personal root classes with the introduction of the super class Personal_data in the classes that will create objects with personal data.

These introductions simplify the programmer's work, as they classify and mark all objects of classes previously marked or automatically marked as personal, and are shown in chapter 4.3.1.

5.2.2 Implementation of metadata storage system

The programmer, when using the classic programming model, must have the following list of responsibilities during the development of his personal data storage system:

- The design and implementation of backend tables for storing metadata relating to the classification of personal or public data,
- Development and implementation of the necessary functions to control access and access to the stored data,
- Classification system for data stored, for separating data into personal and public data,
- Implementation of all functions needed for, using previously saved metadata, the implementation of European regulations and to keep all metadata up to date.

Observing the practical examples shown in the following figures it is possible to notice that there is a great simplification in the programmer work because with the use of the new system:

- the programmer doesn’t need to know the implementation and details of the backend tables used as system support
- the programmer just needs to call the functions implemented in the library.

5.2.3 UI development and access to metadata

In the next subsections, a comparison is made between the use or not of GGML in code development by software developers to impose the rules presented in GDPR on their data storage systems. These comparisons were made and grouped in some examples of GDPR guidelines that the system helps to address.

Information of personal data (Sec 2, art 13-a)

In the first example presented in figures 5.3 and 5.4 it is possible to analyze the difference between the use of this system in the development of applications that work with personal data.

In the following two figures it is possible to observe the pseudocode related to the query of the values of the validity intervals, a value saved due to the implementation of the GDPR regulations, of
the database, in figure 5.3 it is possible to observe that the programmer is the entity responsible for managing sessions and communications between the database and application, as well as the person responsible for implementing application logic that makes the application aware of which classes need an expiration date value.

As it is possible to observe in figure 5.3, for the implementation of the directive 13-a) of the GDPR the programmer is obliged to perform session management, in line 4, 5 and 16, to make a query about which data classes are personal, in line 7 - 9, and finally a query to associate a validity value to each of these classes in lines 11-14.

In figure 5.4 it is already possible to see improvements in the amount of code needed by the programmer to achieve the same functionality, since all the values and additional logic are found in the system back-end, being unnecessary for the programmer to know its detailed operation and decreasing the possibility of human error in the imposition of GDPR rules regarding the storage of personal data.
Data storage period (Sec 1, art 24-b)

In these images it is possible to see the necessary logic implementation so that the web application does not keep stored personal records whose deadline, obtained through the Consent of the data owner, has been exceeded.

```python
@route('/cleanall')
def cleanall():
    Session = sessionmaker(bind=engine)
    session= Session()
    metadata=session.query(Metatable).all
    for data in metadata:
        all_priv_classes.append(data.l_personal)
    for class in all_priv_classes:
        for data in session.query(Metatable.expirationdate).filter(Metatable.l_pessoal==class.__tablename__):
            value=data.expirationdate
        date=datetime.now().replace(microsecond=0)
        session.query(class).filter((class.created_date)<date-timedelta(days=value)).delete()
    session.commit()
    session.close()
return '<h1> All Classes are within expiration date </h1>'
```

Listing 5.5: Code example related to cleaning all out-of-date records (classic model)

As it is possible to observe in figure 5.5, for the implementation of the directive 13-a) of the GDPR the programmer is obliged to perform session management, in line 4, 5, 19 and 20, to make a query about which data classes are personal, in line 7 - 9, a query to associate a validity value to each of these classes in lines 11-13 and finally a search for the current date is made on line 15 and all data where the difference between the storage date and the current date is greater than the validity range is deleted on line 17.

```python
@route('/cleanall')
def cleanall():
clean()
return '<h1> All Classes are within expiration date </h1>'
```

Listing 5.6: Code example related to cleaning all out-of-date records (new model)
Access to personal data (Sec 2, art 13-b)

The two examples shown previously are practical examples of the simplification, due to the introduction of the new system, of two functionalities used mainly in the DPO optics, in this way an example of the use mostly of the common user was chosen. In figure 5.7 and 5.8 it is possible to analyze the simplification of the implementation of the functionality responsible for making available all personal records related to a specific class, used mainly to search all personal records related to a specific person.

In figure 5.7 it is possible to verify that the programmer is responsible for, in addition to managing the sessions in lines 5,6 and 18, communications and implementing additional application logic, implementing features that store the relationships between classes so that it is possible to perform a search in the database receiving all the personal records contained in related classes, at lower levels with the initial class. This search for relationships and constant index exchange is carried out on lines 8 to 16 of figure 5.7.

```
@route('/show_all_data')
def show_all_data(class, index):
    data=[]
    Session = sessionmaker(bind=engine)
    session= Session()
    id_pk=inspect(class).primary_key[0].name
    object = session.query(class).filter(class.__dict__[id_pk]==index)
    for key, value in object.__dict__.items():
        data.append(object)
    descendants=[]
    descendants=ordered_find_direct_descend(graphs, class.__tablename__)
    for object in descendants:
        data.append(object)
    session.close()
    return results
```

**Listing 5.7:** Example code for searching all personal records (classic model)

```
@route('/show_all_data')
def show_all_data(class, index):
    results=[]
    results=showclassdata(class, index)
    return results
```

**Listing 5.8:** Example code for searching all personal records (new model)

With the inclusion of the three previous practical examples, formed by a demonstration of the implementation of the functionality in the current programming model and in the model with the introduction of
this system, it is possible to realize the main advantages, for the software developer, of the introduction of the system in the implementation of the web application, with the implementation of this new system an increase in the level of abstraction in the development of web applications is made. These examples also demonstrate the impulse of the decrease in the programmer’s working hours when implementing a web application, following the regulations instituted in European territory, using the help provided by the system developed in this thesis.

It is possible to observe that the implementation and development of web applications take place without such a high level of dependence on the knowledge of the programmer, because in addition to the active functionalities the passive functionalities help in imposing GDPR regulations, without the programmer needing to have specific knowledge. The entire session management and communications are carried out by the system, thus reducing the factor of human error in that field.

5.2.4 Implementation of GDPR requirements

To incorporate the system, developed during this thesis, in the implementation of the web application of the practical example, shown in figure 5.1, the programmer is responsible for creating a file named DML, where the structure of the classes of data is developed and structured, for storage in the database, and where the relationship between them is indicated. This file also introduces the meta class, in all classes, and is where the Person and Employees classes are marked, as described in chapter 4.3, as personal classes.

With the contribution of the information in this file, the first step of the system is the creation of a table, kept parallel to the data, composed of the metadata common among all personal records belonging to the same class, as shown in figure 5.1. This metatable is made up of a list composed of all classes classified as personal, or by the programmer in the DML file or automatically by the system, and all global metadata values, explained in table 4.1, these metadata are obtained by sending forms to obtain the consent of the entities that have the legal right over the data, which may vary from example to example.

<table>
<thead>
<tr>
<th>List Personal Classes</th>
<th>Experation Interval</th>
<th>Data Goal</th>
<th>Data Categories</th>
<th>Data Owner</th>
<th>Data Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>180 days</td>
<td>Statistical Studies</td>
<td>Direct data</td>
<td>Data Holder</td>
<td>Direct</td>
</tr>
<tr>
<td>Employees</td>
<td>180 days</td>
<td>Management</td>
<td>Direct data</td>
<td>Employer</td>
<td>Direct</td>
</tr>
<tr>
<td>Checkin</td>
<td>200 days</td>
<td>Statistical Studies</td>
<td>Indirect data</td>
<td>Data Holder</td>
<td>Direct</td>
</tr>
<tr>
<td>Schedule</td>
<td>200 days</td>
<td>Statistical Studies</td>
<td>Indirect data</td>
<td>Restaurant</td>
<td>Indirect</td>
</tr>
<tr>
<td>Grade</td>
<td>200 days</td>
<td>Statistical Studies</td>
<td>Indirect data</td>
<td>Data Holder</td>
<td>Direct</td>
</tr>
</tbody>
</table>

Table 5.1: Backend table of the practical example of figure 5.1

Finally, with the help of the data present in table 5.1, the system provides the features that help the programmer to enforce the rules defined by the GDPR. The system automatically makes a subsequent review of the filling of the backend table, making a search for filling errors and if there was any contradiction in the rules for formatting the meta table or failure to fill in any meta data field belonging to a personal class, leaving the table with empty cells, the system activates a feature that issues warnings.
to the programmer’s console with information about the error found as well as its location and possible actions that lead to its correction.

5.2.5 GDPR Administrator Data Handling

To prove the functioning of the implemented system, sets of actions were carried out using the functions and functionalities, made available by the system library to the programmers, in the application of a practical example with the purpose of a future generalization of the results obtained.

In conjunction with the data model presented in Figure 5.1, a web application was developed to apply this same model and give the app user a restaurant and meal rating system. In Figure 5.3 it is possible to observe the application administrator template, where the data collected by the application is managed, during its operation. This template also contains links to some of the functions implemented in the library developed in this paper regarding the enforcing of GDPR regulations, so the administrator or DPO can use the functions made available by the library.

![Figure 5.3: Administrator page](image_url)

To test the operation and analyze the advantages of introducing the system and using its features in the development of web applications, several sets of actions were designed. The first set of actions was developed to test the manipulation and treatment of the collected metadata, due to the GDPR, for records classified passively as personal data.

For practical demonstration of the functions made available to developers by the library developed during this project, links to the calling of these same functions, for the person class present in the model...
represented in 5.1, were included in the administrator page. Present in 5.4 is the practical example of the result obtained for calling the function responsible for obtaining the expiration interval value of a specific class, in the case of this example the person class.

![Figure 5.4: Administrator: Result from the function to show the expiration date](image)

Demonstrated in figures 5.5 are the set of actions carried out by a user or by the logic of the program in order to manipulate one of the saved meta data fields used in the help provided for the imposition of GDPR regulations. In subfigure a) a server template of the developed application is displayed, as a practical example, where it is possible to observe the call, and admission of an argument, of the function, made available by the system, for manipulating the expiration interval value saved for a specific class. Subfigure b) shows the call to the function that shows the value of the expiration date was called again to prove that the value stored in the database was replaced.

![Figure 5.5: Administrator: Metadata Manipulation](image)

Another feature provided by the very advantageous system and used by DPO users is the functionality that lists all records, of classes labeled as personal, that are outside the legal validity period for their storage, a legal term obtained through Consent obtained from the data owners.

Another set of actions planned and executed in the web application, with the aim of testing the performance of the functionality previously described, is the manipulation of the date of collection and storage of two personal records, so that their creation date leads the records to be in a irregular situation, according to the GDPR. Finally, the function is called by the user of the application, which lists all personal records in irregularity due to the storage time and the template shown in figure 5.6 point a) is obtained, in this template it is possible to analyze the data that are found out of date and the time interval in which the data is in a irregular situation.

In figure 5.6 it is possible to observe the same template of the functionality present in figure 5.6 point a), but after having called the function that performs the cleaning of all records that are in an irregular state due to the time interval that is stored. It is possible to verify that the elements that are out of date
in the previous figure were removed from the database, as described in the GDPR regulations, because the list is now empty.

(a) Result from the function to show all data expired

(b) Result from the function to show all data expired after cleaning

Figure 5.6: Administrator: Expired data management

5.2.6 GDPR User Data Manipulation

For the study and test of the performance of more complex functionalities and in the perspective of the common user, a database was created, for the practical example used until now, and it was filled with fictitious data in order to obtain a specific result for the tests to be carried out later. It is possible to analyze the structure and content of the database created in figure 5.7, where all the data has the objective of testing a specific functionality of the new system implemented.

Figure 5.7: Database Contents Example
To analyze the performance and behavior of the features focused on the actions of users classified as common, a set of templates were built on the application server that activated the functions in relation to a specific user, in the figures 5.8 and 5.9 a fictitious personal record was chosen randomly to exemplify the use of these templates.

One of the main features focused on the average user is the functionality that returns all personal records to the user as well as all the results obtained from those records, as long as it is possible to make a connection between the record owners and the results obtained from of the same records. A link was created in the templates of the personal pages present in the application that triggers the functionality previously described, in figure 5.10 the result of calling this function is visible, with the database structure present in 5.7, with the starting point in a record of specified index and class, it is also noticeable the amount of level that the function traverses in the relational graph of the database observing the indentation present in the templates where the results are found.
5.2.7 Restaurant Classification mobile application

To prove the correct functioning of the library developed for this thesis and the ease of implementing the system’s functionalities in a web application, an example mobile application user interface was created for the web application, the next images available are mockups from this mobile application. This mobile application it would make REST requests that would in turn call the server side that is described in section 5.2.5.

Due to the implementation of the system functionalities in API's REST, mobile applications can easily use the functions provided by the system, transforming and using these functions in REST endpoints, regardless the application’s graphic template. As shown in the figure 5.11, users have access to a screen where the library’s main features, for managing data stored under GDPR standards, are displayed.

Figure 5.11: Web application template for personal data processing
In the personal data options screen shown in figure 5.12, it is possible to notice that it is divided into 3 parts. In the first part there is a dynamic list, marked in figure 5.12 point b), which makes a request to the system which classes are classified as personal, and a button that, after selecting a personal class from the list, activates the functionality responsible for making a list composed by all out-of-date personal records belonging to that class stored in the system.

This example proves the ease of including this system in a mobile application because to obtain these features, the programmer would only have to connect the graphical part of the application, under development, and the end point of the **Personal list** function, a function available in the system library, and from the button to the end point related to the **Clean list** function, automating part of the construction work of the mobile application.

![Figure 5.12: Templates for input information from users](image)

In the screen in figure 5.12, it is possible to observe the existence of a dynamic list, used by the user to choose a personal class, and two buttons marked in figure 5.12 point c), which are associated respectively with rest endpoints responsible for displaying or manipulating metadata of a previously chosen personal class, stored in the back end table shown in figure 5.1.

In the last fragment of the functionalities screen on the personal records it is possible to check the existence of a dynamic list explained and marked previously in figure 5.12 and a button where the user can make a query which the answer is a list with all records related to his personal data or personal class chosen by him, the result obtained by activating this functionality is similar to that shown in figure 5.10.

With this set of examples of screens and templates it is possible to demonstrate the simplicity of
incorporating this system in the creation of an interface for manipulating options on personal data of a user of the mobile application, according to GDPR regulations. Because the programmer is only responsible for the creation and design of the graphical part of the interface, it is only necessary to implement the connection between the graphical part and the endpoints REST related to the features that the programmer wishes to make available to the user in this mobile application.

5.3 Overheads Evaluation

In this section, it is studied what are the main impacts of implementation of the system developed in this thesis, Generic GDPR Management Layer (GGML), in examples of web applications and systems developed following the classic programming model. All tests and examples, present in this section, were developed on the Linux operating system.

5.3.1 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.

Since additional code is run when creating objects, it is necessary to evaluate the existence of overhead increases. To do these tests, several amounts of objects were created using GGML and not using GGML to understand the system’s influence on the overall performance of the application.

As you can see in table 5.2, 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.

<table>
<thead>
<tr>
<th>Number of objects</th>
<th>Classic Model</th>
<th>GGML</th>
<th>Overhead/object</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.03 sec</td>
<td>0.08 sec</td>
<td>5 μsec</td>
<td>62.5 %</td>
</tr>
<tr>
<td>100</td>
<td>0.497 sec</td>
<td>0.623 sec</td>
<td>1.26 μsec</td>
<td>20.22 %</td>
</tr>
<tr>
<td>1000</td>
<td>4.49 sec</td>
<td>5.53 sec</td>
<td>1.04 μsec</td>
<td>18.8 %</td>
</tr>
<tr>
<td>10000</td>
<td>56.025 sec</td>
<td>1 min 0.5 sec</td>
<td>0.4525 μsec</td>
<td>7.39 %</td>
</tr>
<tr>
<td>100000</td>
<td>9 min 23.59 sec</td>
<td>10 min 32.38 sec</td>
<td>0.6879 μsec</td>
<td>10.87 %</td>
</tr>
</tbody>
</table>

Table 5.2: Overhead Times (20 bytes)

As it is possible to observe in table 5.2 when a reduced number of objects is created and without the delay of communications, the overhead value is high, as it corresponds to the startup times of GGML. With a high number of objects this overhead is diluted, stabilizing around 10 % increase.

In the case of the tests presented in table 5.2, where the objects of type Person, shown in the 5.1 listing, are inserted in the database with a reduced size (approximately 20 bytes). In table 5.3 the objects were artificially increased to test the impact of the system developed in object inserts, of size extensive (1 KB) in the database.

As you can see in the first test of each of the tables, there is a system initialization overhead, observable in the considerable increase in the system's running time, due to the creation of a low number of objects, this overhead is then diluted in longer runs and with a high number of objects created.

<table>
<thead>
<tr>
<th>Number of objects</th>
<th>Classic Model</th>
<th>GGML</th>
<th>Overhead/object</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.06 sec</td>
<td>0.106 sec</td>
<td>4.6 μsec</td>
<td>43.39 %</td>
</tr>
<tr>
<td>100</td>
<td>0.8 sec</td>
<td>0.961 sec</td>
<td>1.61 μsec</td>
<td>16.75 %</td>
</tr>
<tr>
<td>1000</td>
<td>9.92 sec</td>
<td>11.48 sec</td>
<td>1.56 μsec</td>
<td>13.58 %</td>
</tr>
<tr>
<td>10000</td>
<td>1 min 37.7 sec</td>
<td>1 min 54.96 sec</td>
<td>1.726 μsec</td>
<td>31.40 %</td>
</tr>
<tr>
<td>100000</td>
<td>16 min 41.59 sec</td>
<td>19 min 25.78 sec</td>
<td>1.6419 μsec</td>
<td>14.08 %</td>
</tr>
</tbody>
</table>

Table 5.3: 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 5.2 and table 5.3.

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.

### 5.3.2 Memory Overheads

As metadata must also be stored in the database, it is necessary to assess whether the increase in storage memory is significant. In this way another test developed with the aim of testing the impact of the introduction of Generic GDPR Management Layer (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.
### Table 5.4: Memory Overheads (20 bytes)

As for the tests of overheads related to times of use, memory tests were made with objects with an artificially size fixed. In the values that can be seen in table 5.5 the objects have a size of at least 1 Kbyte.

<table>
<thead>
<tr>
<th>Number of objects</th>
<th>Classic Model</th>
<th>GGML</th>
<th>Overhead/object</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>24.6 kB</td>
<td>53.2 kB</td>
<td>2.86 kB</td>
<td>53.75 %</td>
</tr>
<tr>
<td>100</td>
<td>24.6 kB</td>
<td>61.4 kB</td>
<td>0.368 kB</td>
<td>59.93 %</td>
</tr>
<tr>
<td>1000</td>
<td>73.7 kB</td>
<td>135.2 kB</td>
<td>61.5 B</td>
<td>45.48 %</td>
</tr>
<tr>
<td>10000</td>
<td>512 kB</td>
<td>892 kB</td>
<td>38 B</td>
<td>42.60 %</td>
</tr>
<tr>
<td>100000</td>
<td>5.4 MB</td>
<td>8.9 MB</td>
<td>35 B</td>
<td>39.32 %</td>
</tr>
</tbody>
</table>

### Table 5.5: Memory Overheads (1 KB)

As it is possible to observe in table 5.4, 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 5.5 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%.

### 5.4 Relationship between system features and GDPR articles

Finally, it is possible to conclude that with the introduction of this system in the development of an application it is possible to solve some problems created by the imposition of GDPR. With the introduction of some functionalities idealized and presented in tables 3.1 and 3.2.

The **show features**, responsible for making the metadata available to the programmer and the user, and the automatic classification of data into personal data are part of measures used by the system to mitigate the needs created by articles **Sec 1,art 12-a and b**, **sec 2,art 13-a** and **Sec 1,art 24** of the GDPR. Specifically for article described in **Sec 3,art 18**, the Clean and Clean list functions were created to keep the data in the database only for the time period referred to in the consent required from the user and only extendable with a re-consent action.

The function that mitigates the imposition problems of the GDPR, created by articles **Sec 2,art 13-b** and **Sec 3,art 20** of the same, is the show data class function that receives a unique index value for a data set, and returns that set as well as all the data that are are stored in the database and are related to it.

The **forget class data** and **change functions** are features belonging to the library that, similarly to the show class data function, take a specific data structure and search all the data that are related to it,
later erasing it or funtions that change the meta data of a specific data set. These functions address the concerns arising from articles Sec 3,art 16/17 and Sec 3,art 21/22 of the GDPR.

The change and alert empty functions can be used by the programmer to address the problem of notifications of changes made to the metadata and data of a user as referred to in Sec 3,art 19.

Finally, the log operation function deals with problems such as the listing of operations carried out on the data as well as their classification in terms of risk and their differentiation, referred to in articles Sec 1,art 30 and Sec 3,art 35.
Chapter 6

Conclusions

In May 2018, the GDPR guidelines were implemented to regulate the processing and storage of personal data in the European Union. These regulations created a set of needs to be met by data storage systems, this thesis presents a tool that helps in suppressing some of those needs. This tool helps programmers to implement data storage systems according to GDPR directives.

Firstly, a study was carried out on the requirements met by the tools currently available and which of them are necessary to be completed by another tool, then the presentation and description of the tool that was developed in the thesis, GDPR, 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 considered that the tool remained generic, being able to be easily adapted to each ambience or Programming language

6.1 General description and Advantages

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

The developed tool works as an extension of ORM and provides programmers with the following functionalities:

- Help features for implementing a data storage system according to GDPR,
- Automatic mechanism for marking and classifying personal data,
- Personal data processing help features according to GDPR,
- Functionalities for managing stored personal data.

With the use of this new layer it is possible to take advantage of a set of features that are made available to various groups of GDPR target individuals (Programmers, DPOs and Data holders). The
advantages that come from this tool and the groups that they affect are described in the following sections.

6.1.1 Benefits for programmers

The layer developed in this thesis, **Generic GDPR Management Layer (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.

6.1.2 Benefits for DPOs

Another class of users that benefits from the advantages provided by the new layer of personal data processing and **Data Protection Officer (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.

6.1.3 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.
6.2 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.

In order to avoid these accessibility errors, it is necessary to implement complex database management techniques as discussed in article "Forgetting personal data and revoking consent under the GDPR: Challenges and proposed solutions" [29]. This problem is also responsible for creating difficulties in creating backups as discussed in article "Backups and the right to be forgotten in the GDPR: An uneasy relationship" [30].
Bibliography


Appendix A

GGLM Classes

In this annex it is possible to consult the implementation of the classes designed to implement the application, used as a practical example of this thesis.

A.1 Restaurant rating classes

```python
class Employee (Base , PersonalData):
    __metaclass__ = GGML_MetaClass
    __tablename__ = 'employee'
    id_e = Column ('id_e ', Integer , primary_key = True)
    name = Column ('name ', String)
    email = Column ('email ', String , unique = True)
    schedule_e = relationship ("Schedule")
```

Listing A.1: Employee class implementation

```python
class Restaurant (Base):
    __metaclass__ = GGML_MetaClass
    __tablename__ = 'restaurant'
    id_r = Column ('id_r ', Integer , primary_key = True)
    name = Column ('name ', String)
    adress = Column ('adress ', String , unique = True)
    Checkin_r = relationship ("Checkin")
```

Listing A.2: Restaurant class implementation

```python
class Schedule (Base):
    __metaclass__ = GGML_MetaClass
    __tablename__ = 'schedule'
    id_s = Column ('id_s ', Integer , primary_key = True)
    id_e = Column ('id_e ', Integer , ForeignKey ("employee.id_e"))
    id_r = Column (Integer , ForeignKey ("restaurant.id_r"))
    checkin = Column ("checkin", DateTime)
    checkout = Column ("checkout", DateTime)
```
Listing A.3: Schedule class implementation

class Grade (Base):
    _metaclass_ = GGML_MetaClass
    __tablename__ = 'grade'
    id_g = Column('id_g', Integer, primary_key=True)
    id_s = Column(Integer, ForeignKey('schedule.id_s'))
    id_c = Column(Integer, ForeignKey('checkin.id_c'))
    grade = Column('grade', Integer)

Listing A.4: Grade class implementation