jamk.fi

GDPR and its solution in Django

Maksim Rusinau

Bachelor's thesis
December 2018
School of Technology, Communication and Transport
Degree Program in Information and Communication Technology

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s)	Type of publication	Date
Rusinau, Maksim	Bachelor's thesis	14.12.2018
		Language of publication: English
	Number of pages 46	Permission for web publication: x

Title of publication
GDPR and its solution in Django

Degree programme
Software Engineering

Supervisor(s)
Salmikangas, Esa

Assigned by
Qvantel Finland Oy

Abstract

The General Data Protection Regulation applies to the personal data and the parties concerned. It has become applicable in the force on 25 May 2018. Every party processing the personal data is under impact of the GDPR.

The document is divided into two parts – theoretical and practical. First part consists of four chapters. The first chapter of the document is an introductory point. The second chapter explains the eight principles of the Organization for Economic Cooperation and Development as answer to privacy regulations. The third part – what has been brought with Directive 95 and why it has been repealed. The fourth part contains the main aspects of the GDPR, the bullet points.

The practical part of the document – is an application developed for the specific business need, the implementation also considers all the rights of the data subject introduced with GDPR and ensures the established principles of data processing. The application consists of two parts. The first part is built upon Django Web Framework, the second part is on Rest Framework. The Rest application rather represents the data flow between other similar Django Web applications and how to achieve the right of data portability.

The goal was to analyse the GDPR, to build the IT solution that meets the requirements of the GDPR. The goals were achieved successfully.

Keywords/tags (subjects)
OECD, Directive 95, GDPR, Django, REST, Python

Miscellaneous

Contents

Figures

Tables

Terminology

API – Application Programming Interface

Amendment – change or addition to the legislation

Authorization – process of giving user access to the system

Celery – software for distributing asynchronous tasks

Coding – writing data in such format, that accessible to every person

Cron job – scheduled task running automatically

DDOS – Distributed Denial of Service

DOS – Denial of Service

DPO – Data Protection Officer

EDPB – European Data Protection Board

Eligible – allowed to do something

Encryption – writing data in such format that only authorized persons can read it

GDPR – General Data Protection Regulation

HTTP – Hypertext Transfer Protocol

Intelligible – able to be understood

JSON – Java Script Object Notation

Key – data used to encrypt and decrypt the message

OECD - Organization for Economic Cooperation and Development

PKI – Public Key Infrastructure

Party – natural or legal person, public authority, agency or body

REST – Representational State Transfer

1 Introduction

1.1 Purpose

The thesis presents a study of the GDPR, its predecessors, the development of IT solution. The analysis is conducted from the point of view of the developer involved in information technologies and business solutions. How has the life of private persons become more secure and reliable in terms of data flow, i.e. creation, transformation and deletion, and prevention of loss and/or misuse of data?

The GDPR´s appearance was inevitable with the changing environment and broadening of technologies. The approach to how data obtained, the way it is used, how it is transformed and transferred further in the context of service has caused it. Before GDPR there was a Directive 95, which has been the predecessor and the first attempt to harmonize the society rights and the businesses that use their information to perform their contractual obligations. The impact of the GDPR is permanent, but the side effects of it appear frequently. Thus, the GDPR analysis could not be completed before it came into force, it is better to compare the hassle going on before 25th of May and the perceived changes that were put into different systems and affected humans' life.

1.2 Modern technologies in society

The invention of the internet has brought many conveniences into every day life. The internet has services which enable people to buy and sell goods, study materials located in different countries in many languages, to educate people, make travel plans, create content on specific platforms such as YouTube, Wikipedia, or communicate with others using social media such as MySpace, ICQ, and many other possibilities that just keep increasing.

However, the services as a commerce have grown very rapidly in the internet services. Marketing takes its specific place inside of it. Nowadays, global technology companies utilize the features of the internet as an income or as a research base of its customer browsing over, but they tend to make no harm, to be not evil.

With the internet commerce, the interest of how pages are viewed emerged; meaning what are the points of interest when browsing the Web of the customer-consumer. Knowing the preferences of the customers it is much easier to target them to find a specific product. The location interest helps with marketing and so on; however, this also could lead to pursue the owner of the data. Web sites nowadays use cookies, key-value storages. Request banner about usage of data and storing in cookies explains the use of the information and collection by the site, and if one is agreed with them, then the data are stored. At least, that is how it should work, however, the request to receive information is not explicit, for example, “cookies are used to give the best experience”; however, as a customer giving consent so broad it becomes undefined and dangerous how the data could be used and by whom. Or sometimes consent is not possible to give at all. Some companies even sell such browsing data, some use obsolete security technologies. Could be that the reason for the new change in the existing legislation?

2 OECD

The privacy laws were established in many countries to meet the new technology solutions that operate on personal data. However, the difference in national legislations can hamper the flow of such data across border. Such restrictions on the flows could cause economic harm, e.g. in banking or insurance. The Organization for Economic Cooperation and Development set recommendations endorsed by both the EU and the U.S. to protect the fundamental right of privacy. The guidelines were adopted on 23 September 1980, proposing the principles for the processing of personal data (OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data 1980, 1).

The principles were created to harmonize national privacy legislation, while upholding such human rights, at the same time prevent interruptions in international flows of data. They represent a consensus on basic principles which can be built into existing national legislation or serve as a basis for legislation in those countries which do not yet have it. The eight principles for protection of personal data were (OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data 1980, 1, 3):

Limited collection – personal data obtained in a lawful and fair manners and to the extent what is necessary, if possible with consent of the data subject.

Quality – data are relevant for the purposes for which it is collected, to the extent necessary for those purposes; it is accurate, complete and kept up-to-date.

Purpose – data are collected for the purposes that are specified at the time of collection and used only for those purposes.

Limited Use – data are not used outside of the purpose, otherwise the data subject should expel their consent about the changes.

Security – personal data are protected against risks by reasonable safeguards.

Openness – about development, practices and policies which apply to personal data.

Individual Participation – the subjects have the right to know whether the controller of data possesses the data, have access to data in an intelligible form for a charge if any is applied to it that is not excessively large. The right to challenge the controller for refusing to grant access to their data remains with the subject, as well as challenging the accuracy of data. If not, the data should be erased or updated.

Accountability – data controllers are accountable for complying with the measures detailed above.

3 Directive 95

On 24th of October 1995, the European Union answered to the division of privacy regulations across the EU which yielded into The Data Protection Directive 95/46/EC. Basically it stood for spreading data to countries outside of the Union, to third countries. An independent public authority was established in every member state, the Data Protection Authority, which stood for supervising the application of the directive as a harmonization of data between businesses and citizens. The prerequisite to retrieve the personal data by third party is that it has an adequate level of protection of data, and that level is comparable to the protections used within the EU.

Directive 95/46/EC has adopted eight principles established by the OECD. It also placed a general obligation to report the processing of personal data to the supervisory authorities. That obligation produced administrative and financial burdens, it did not completely contribute to improving the protection of personal data. Thus, the notification obligations should have been abolished and replaced by effective mechanisms. The notification instead should be focusing on types of processing operations that can likely result in a high risk to the rights and freedoms of natural persons by their nature, scope, context and purposes. Such types of processing operations may involve new technologies or a new kind of automation processing and where no data protection impact assessment has been carried out before by the controller, or where they become necessary in the light of time elapsed since the initial processing (European Parliament and the Council of the European Union 1995, (49)).

Directive 95 was meant to bring together the laws of different member states; however, it was still a directive, it left a space for misinterpretation during the transposition into individual national law. Along with today's agile changing data processing landscape it become necessary for another update to the regulatory environment of the EU. The new regulation will become immediately enforceable law in all member states.

4 GDPR

4.1 Preface

The General Data Protection Regulation was conducted on 27 April 2016 and became applicable on 25 May 2018. The Regulation set the rules about the processing of personal data and the free movement of personal data. It is designed at its core to protect fundamental rights and freedoms of the data subjects. GDPR lays responsibilities on shoulders of the controller bodies of personal data in contemporary circumstances and environment. It is not surprising that it repeals Directive 95 completely as obsolete today. What does GDPR actually mean and brings altogether? This chapter is about to address (European Parliament and the Council of the European Union 2016, Article 94, 99).

4.2 The Board

The EDPB is an institution of the European Union, it serves as a controller in execution of rules established in the GDPR. According to the European Parliament and the Council of the European Union (2016, Article 68) “board is composed of the head of one supervisory authority of each Member State and of the European Data Protection Supervisor, or their respective representatives”. The second important task of the Board is to advise the Commission in issues related to the protection of personal data and on any **amendment.** The Commission of the European Union takes a role in promoting the general interests of the EU, it brings new legislations within the EU budget (Institutions bodies 2018).

4.3 Processing

4.3.1 Definitions

The most important terms related to the processing are conducted below (European Parliament and the Council of the European Union 2016, Article 4):

Personal data is information that could lead to a subject, directly or indirectly discovering their identity, for example:

- First and last names
- Electronic mail
- Phone number
- Internet Protocol address
- Latitude and longitude
- Gender
- Date of birth
- Social security number
- Any physical, physiological, genetic, mental, economic, cultural attribute of the data subject

Processing refers to an operation which is performed on personal data such as collection, structuring, disclosure by transmission, destruction.

Restriction of processing refers to limiting the processing of personal data.

Profiling is an automated analysis serving to evaluate personal aspects and make predictions about condition and behaviour of a natural person.

Pseudonymisation refers to the transformation of the personal data in such a way that the output information could not lead to a subject without additional information, i.e. an algorithm, which is kept separately.

Controller is the processing party, it defines by what means and how to process personal data.

Processor is the processing party engaged by the controller.

Third party refers to the processing party authorized by controller or processor.

Recipient is the party to which the personal data are disclosed.

Consent refers to an explicit indication of will which signifies an agreement to the processing.

Personal data breach is a breach of security in storage of personal data.

Main establishment is the establishment of the controller or processor, which defines its activities.

Representative is a party which represents the controller or processor.

Group of undertakings refers to controlling undertaking and its controlled undertakings.

"Supervisory authority is an independent public authority which is established by a Member State".

Supervisory authority concerned is regarded as concerned because the processing is carried, or the data subjects are residing on the territory of that supervisory authority.

Cross-border processing refers to processing which is carried out or data subjects affected in the Union.

4.3.2 Principles

The principles related to data processing represented below (European Parliament and the Council of the European Union 2016, Article 5):

Lawfulness, fairness and transparency mean that data should be processed with lawfulness and fairness and in a transparent manner to the data subject.

Purpose limitation means that data collected for specified, explicit and legitimate purposes. If the purpose changes, the data subject shall be communicated by the controller.

Data minimization means an adequate, relevant and limited collection of data to what is necessary.

Accuracy means that the data kept up to date, otherwise – erased or corrected.

Storage limitation means that the personal data are stored only to the extent period estimated or defined for the processing.

Integrity and confidentiality mean protection of the personal data against losses and undesired disclosures to untargeted parties.

The controller must be transparent to the data subject, which means that the subject knows how personal data used, where, by whom and how long. The controller must be able to demonstrate the compliance with the principles.

4.3.3 Lawfulness of processing

Processing is considered lawful if (European Parliament and the Council of the European Union 2016, Article 6):

Consent is given to the processing of personal data for a specific purpose.

Contract performance requires the processing, to which the data subject is party.

Legal obligation processing is required for compliance by the controller.

Vital interest processing is carried out for protection of data subject or another natural person.

Public interest or exercise of official authority task is carried out.

Legitimate interest is pursued, and processing is lawful. The restriction does not apply to a public authority.

4.3.4 Consent

Consent must be requested in a clear and plain language. The part which constitutes the infringement shall not be binding. The given consent should be possible to demonstrate. It is easy to give and withdraw consent, and it is the right of data subject to do so at any time. The holder of parental responsibility over the child, who is less than 16 years old, shall express consent of the child. It is the controller's responsibility to verify consent from the holder. In some cases, the age could not be below 13 years (European Parliament and the Council of the European Union 2016, Article 7, 8).

4.4 Rights

The General Data Protection Regulation has introduced the following data subject's rights, to which the controller is subject to execute:

- Right of transparency and modalities
- Right to access
- Right to rectification
- Right to be forgotten
- Right to data portability
- Right to object
- Right to compensation and liability

4.4.1 Right of transparency and modalities

Any information related to processing should be given in human readable format by the controller to the data subject concerned. Exercising the rights should be done by the subject's request if the subject is identifiable, and this should be done without undue delay. The controller can request the additional information for identification and execute rights within one month, considering that the request has not a repetitive signature. If actions are not taken by the controller, the data subject should be informed and should have the possibility to seek a judicial remedy. Any communication and any actions taken under rights and communication of a breach shall be provided free of charge unless it is excessive and makes inadequate effort. The controller shall express if it intends to transfer personal data to a third party and the information about existing safeguards. If the profiling takes place, the impact of such profiling should be exposed to the data subject (European Parliament and the Council of the European Union 2016, Article 12).

4.4.2 Right of access

The right to obtain from the controller the additions to Transparency and modalities paragraphs if the personal data are being processed (European Parliament and the Council of the European Union 2016, Article 15):

a) information about the categories of personal data, the existence of the right to rectify, erase, restrict, object the processing;

b) access to and copy of the personal data.

4.4.3 Right to rectification

The right to rectification of inaccurate data refers to having incomplete personal data completed by providing a supplementary information (European Parliament and the Council of the European Union 2016, Article 16).

4.4.4 Right to be forgotten

The request for the erasure of the personal data shall not be executable only if there are legal or legitimate grounds for the processing (European Parliament and the Council of the European Union 2016, Article 17).

4.4.5 Right to restriction of processing

Processing of restricted data shall be done with the data subject's consent or for the interest of a natural person or public. Before lifting the restriction, the data subject shall be communicated about the execution of such restriction. To receive such right from the controller one of the following conditions is required (European Parliament and the Council of the European Union 2016, Article 18):

a) the accuracy principle is broken;

b) "the processing is unlawful, and the data subject opposes the erasure of the personal data and requests the restriction of their use instead";

c) "the controller no longer needs the personal data for the purposes of the processing, but they are required by the data subject for the establishment, exercise or defense of legal claims";

d) the data subject's rights are overriding the legitimate grounds of the controller.

4.4.6 Right to data portability

The right to send the personal data to another controller if the processing is based on consent or contract and is carried out by automated means (European Parliament and the Council of the European Union 2016, Article 20, paragraph 1, 2).

4.4.7 Right to object

The request to object processing shall be executed unless the compelling legitimate grounds for such processing have been demonstrated. Processing for scientific, historical or statistical purposes can be objected by the data subject, unless the processing is necessary for reasons of public interest (European Parliament and the Council of the European Union 2016, Article 21).

4.4.8 Right to compensation and liability

The data subject shall have the right to seek for a compensation if the damage has been caused the fault of the processing body. If the controller is fully liable, the procession is only partially. According to European Parliament and the Council of the European Union (2016, Article 82) “a processor shall be liable for the damage caused only where it has not complied with obligations of this Regulation specifically directed to processors or where it has acted outside or contrary to lawful instructions of the controller”.

4.5 Controller and Processor

4.5.1 Processor

The processor and its undertakings are authorized by the controller to process personal data if they provide appropriate security measures. The authorization shall be governed by a legal binding act, which defines the terms for the processing, such as duration, purpose, categories of personal data and the obligations and rights of the controller. If instruction infringes existing data protection provisions the processor is obliged to inform the controller. However, if the undertaking processor caused failure in the data protection obligations, the initial processor remains fully liable to the controller. The processor shall help with integrity and confidentiality of processing,

with exercise of the rights. Send the notification of a personal data breach if any occur, to prevent such, the processor shall carry out the protection assessments. The processor can show the demonstration of compliance with the obligations and audits. At the end of processing activities, the processor shall return or delete all the personal data to the controller unless the opposite is required by the law (European Parliament and the Council of the European Union 2016, Article 28, paragraph 1 - 4).

4.5.2 Records of processing activities

Every processing activity must be recorded. According to European Parliament and the Council of the European Union (2016, Article 30) the requirement does not apply to a smaller organisation, less than 250 employees “unless the processing it carries out is likely to result in a risk to the rights and freedoms of data subjects, the processing is not occasional, or the processing includes special categories of data or personal data relating to criminal convictions and offences”.

4.5.3 Security of processing

The pseudonymisation and encryption of personal data is considered as appropriate. The next is to ensure the execution of the integrity and confidentiality principle. The system availability ratio should be real and, in a time-wise period. The regular testing and assessment are a plus (European Parliament and the Council of the European Union 2016, Article 32, paragraph 1).

4.5.4 Notification of a personal data breach

After the awareness of the personal data breach, the notification shall be sent to the supervisory authority within 72 hours, to data subject concerned without undue delay. If the notification is not sent within an indicated time limit, it shall be accompanied by reasons for the delay. The notification shall describe the nature of the breach and it consequences, the approximate number of data subjects and records concerned. The supplementary information as contact details of the DPO and what measures have been taken is a must (European Parliament and the Council of the European Union 2016, Article 33, paragraph 1 - 5).

However, the notification to the data subject is not required if the controller has taken measures to ensure that the risk to the rights and freedoms of data subject is not likely to appear. If the notification causes a disproportionate effort, it can be done through a public communication to the data subjects concerned (European Parliament and the Council of the European Union 2016, Article 34, paragraph 1, 3).

4.6 Data protection officer

A data protection officer is designated by controller or processor by professional qualities and expert knowledge of data protection law and practices as well as by the ability to fulfil the tasks. The processing body must publish the contact details of the data protection officer and provide them to the supervisory authority. The processing body must provide support to the data protection officer in the performance of its tasks by providing resources and access to personal data and processing operations, and to maintain expert knowledge. The data protection officer must help with data protection provisions and assessment, with the training of staff involved in processing operations and the related audits (European Parliament and the Council of the European Union 2016, Article 37, 39).

4.7 Transfer of personal data to third countries

The transfer to the third party can only occur if it is compliant with the processing body and the third party has an adequate level of protection of data and listed in the Official Journal of the European Union by the decision of the Commission (European Parliament and the Council of the European Union 2016, Article 45).

4.8 Fines

The base penalty fee for the infringements of the controller's or processor's obligations is counted to 10 000 000 EUR. The violations in other circumstances can be fined with up to 20 000 000 EUR (European Parliament and the Council of the European Union 2016, Article 83, paragraph 4, 5).

5 Implementation of Django Web application

5.1 Foreword

Application development is a process of executing the need for something into a digital solution and it starts from the business requirement divided into smaller parts, as stories, to coding and testing. The Application has been developed for the demonstrating purposes only, and should be used only for this, used to show how to develop the features laid down in the GDPR. It serves as a service management tool of memberships of the gym facility, its customers, paying system, debt collection reporting and registration system. What features needed to be developed in the application to meet the requirements of the GDPR and how? This chapter is to show practical examples and describe theory concepts.

The following figure (Figure 1) shows the project structure.

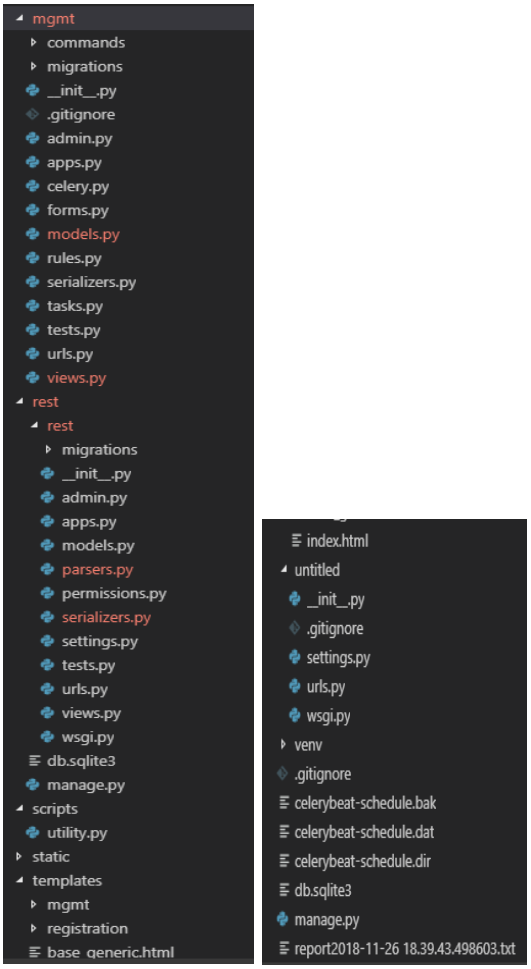

Figure 1. Project structure

5.2 Introduction, set up and theory concepts

5.2.1 Development

The Operating system used for the application implementation is Windows 10 Home Edition. However, it lacks the libraries that will simplify the coding needs and requirements in comparison with Linux. To start development it is necessary to pull some libraries. Python for Windows could be retrieved from the python.org website. With that package there is also a handy tool called pip, which makes easy installations of the libraries. Just by executing *pip install pip* (pip cannot install pip, as the requirement is already satisfied) command in the Shell, package pulling is done to the system.

To download the installer of the used IDE the one could refer to code.visualstudio.com. The Integrated Development Environment is used for the main reasons as:

- Its simple installation
- It is free
- It is fast

5.2.2 Programming language

Python 3 used as a programming language and lays in the base of Django Web Framework. The below table (Table 1) demonstrates the differences between Python 2 and Python 3.

Table 1. Python 2 versus Python 3

	Python 2	Python 3
ASCII	default	-
Unicode	u'Hyvä'	default
type (4/2)	2 (integer)	2.0 (floating point number)
xrange ()	faster iteration in comparison with range ()	-

print “Hello World!”	a) print … b) print (…)	print (…)
raise of exceptions	a) raise Exception, “Exception” b) raise Exception(“Exception”)	raise Exception(“Exception”)

5.2.3 Framework

Django is a free and open source high-level Python based Web Framework which simplifies the development in the Web and can play a good role as an example of how to meet the requirements of the GDPR and build a good business solution with agile methods. Django uses an MVT architectural pattern, Model-View-Template. Model is a class with its own fields and methods, View represents the Model with the help of the Template, which is an HTML and CSS combination. The instance’s attributes and the class’s object are changed with the help of the Form. Django is configurable with settings.py file, which is just a Python module with module-level variables, not allowing for Python syntax errors, assigning of settings dynamically, importing the values from other settings files.

To create a project in Django, the following is used: “*django-admin startproject example*”. Then the project structure will look like in Figure 2 (left). Then the application needs to be created within this project, the creation is achieved with the “*django-admin startapp example_app*” command. After the application structure creation, the project will look like as shown in Figure 2 (right). For development purposes, the following could be used as shown in figure (Figure 3) (Note! Not in production).

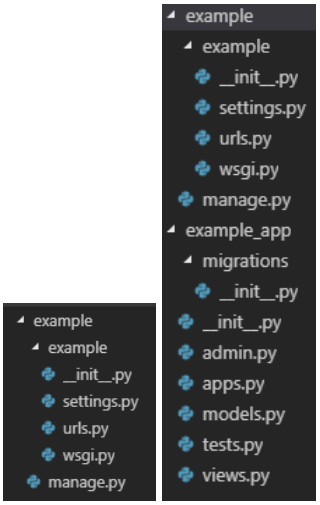

Figure 2. Default Django project and application structure

```
from django.conf import settings

if settings.DEBUG:
    #do something
    pass
```

Figure 3. Example of how DEBUG object could be imported

To start the development server, the following should be run:

```
> python .\manage.py runserver 8000
```

Figure 4. Execution of server on port 8000

Testing of the application is done with unit tests as shown in below figure (Figure 5).

```
from django.test import TestCase

from .forms import CreateSubjectForm


class SubjectFormTest(TestCase):
    def test_if_subject_is_not_adult(self):
        form_data = {'first_name': 'fname',
                     'last_name': 'lname',
                     'date_of_birth': '2018-01-01',
                     'consent_given': 'true',
                     'social_number': 'XYZ',
                     'email': 'n@n.com',
                     'contact_phone': '09901234',
                     'user': '3',
                     'address': 'A',
                     'language': '5',
                     'occupation': '3'}
        form = CreateSubjectForm(data=form_data)
        self.assertFalse(form.is_valid())

    def test_if_subject_is_adult(self):
        form_data = {'first_name': 'fname',
                     'last_name': 'lname',
                     'date_of_birth': '2001-01-01',
                     'consent_given': 'true',
                     'social_number': 'XYZ',
                     'email': 'n@n.com',
                     'contact_phone': '09901234',
                     'user': '3',
                     'address': 'A',
                     'language': '5',
                     'occupation': '3'}
        form = CreateSubjectForm(data=form_data)
        self.assertTrue(form.is_valid())
```

Figure 5. Unit tests examples

5.2.4 Type of attack techniques

The following covers the most common attack techniques terms in Web:

Sniffing – capture of packets by unauthorized persons, hackers, sent in local network and Wi-Fi.

Scanning – protocol scanning, which can give an information about OS version, and thus revealing its leaks and holes.

Spoofing – replace of data by hacker in the between of transmission, “man in the middle”.

Poisoning – replacement of records in ARP table, making possible to receive data from the victim.

ARP – address resolution protocol, mapping of logical addresses to physical addresses.

DoS – attack on computer system to deny the functioning, usually based on overloading of resources.

DDoS – form of DoS attack, based on sending requests from many places, usually the traffic comes from bots.

5.2.5 Security attacks and their preventions in Django

This chapter is an overview of Django’s security features (Django web application security 2018).

Cross site scripting (XSS)

XSS attack based on injection of script on client-side into browser with user input field. The malicious script could be stored in the database where it will be retrieved and displayed to other users, or the link can contain a script which will be executed by the user’s browser. Django templates escape specific characters which are particularly dangerous to HTML. Mark_safe function to mark data as safe can be used when the content is for sure needed and trusted. But it is better to have escape and encode functions in use (Figure 6), the script would call a pop-up window with “Hello World” text; it is safe but worse things could be achieved.

```
In [24]: import re

In [25]: val = '<script>alert(\'Hello World\')<script>'

In [26]: print(val)
<script>alert('Hello World')<script>

In [27]: print(re.escape(val.encode()))
b"<script>alert\\('Hello\\ World'\\)<script>"
```

Figure 6. Encoding and escaping of script, example

Cross site request forgery (CSRF)

CSRF attacks allow a malicious user to execute actions using the credentials of another. The protection works by checking for a secret in each POST request, which is created with csrf_token pasted in a template. This ensures that a malicious user cannot simply “replay” a form POST to website and have another logged in user unwittingly submit that form. The malicious user would have to know the secret, which is user specific (using a cookie). The csrf_exempt decorator disables CSRF tag check.

SQL injection

SQL injection is a type of attack where a malicious user can execute arbitrary SQL code on a database. This can result in records being deleted or data leakage. Django’s query sets are protected from SQL injection since their queries are constructed using query parameterization. A query’s SQL code is defined separately from the query’s parameters. Since parameters may be user-provided and therefore unsafe, they are escaped by the underlying database driver.

DOS

With Django 1.10 came into consideration the security vulnerability such as the amount of data sent in GET or POST methods. It takes time to process those requests, thus, it can lead to the Denial of Service. The object field DATA_UPLOAD_MAX_NUMBER_FIELDS was defined to be 1000 only but could be disabled if set to None.

5.3 Database

The used management database system is SQLite, which is locally deployed and runs fast; however, it has its limitations but for the demonstration purposes it works well. As Django has migrations, which eases the alteration of the database schema, its build, with just changing an attribute of the model it is just necessary to run: *python manage.py makemigrations | python manage.py migrate*, and that is it; if of course, nothing is wrong with the database or data of it. Redundancy issues and architectural issues are not taken into consideration. Use of indexes is strictly recommended for the performance improvements to access the database data. The index creation example is showed in Figure 7.

```
class Meta:
    ordering = ['id']
    indexes = [
        models.Index(fields=['social_number']),
        models.Index(fields=['consent_given'])
    ]
```

Figure 7. Index definition in Django

The personal data could be stored as a plain text and showed to the staff only if consent of the subject is given, the code example can be found in Figure 8. However, the data safe storage could be achieved by one of the following methods:

a) Modern encryption techniques

b) Pseudonymization techniques

The first method only has one problem to consider: where to storage the encryption key and how it is retrieved when the personal data need to be altered or rectified. A solution could be the usage from the Application's key; however, it is not a variable and hence, it could be broken. The second method is easily achieved and gives the consistent results every time; yet, then it does not discover the initial data itself. Below are example code snippet extracts with the conversion of string of chars to string of numbers and colon separator and backwards.

The pseudo_* function used a predefined alphabet as a starting point if it is not provided. Then it converts given input to string representation and to list. In the For loop the input values are substituted from the alphabet position as shown in code snippet in Figure 10 or look up the alphabet to find the corresponding value, shown in Figure 9. The last step is to gather all elements altogether.

The similar principle can work on date data type, such as the date of birth stored as a date field in the Django example application. To make the fake value of it, the shift with the help of relativedelta from dateutil.relativedelta could be done on real data. An example is shown in Figure 11.

```
def get_details(self):
    if self.consent_given:
        return (self.id, self.first_name, self.last_name, self.date_of_birth, self.social_number, self.occupation,
                self.email, self.consent_given, self.contact_phone)
    else:
        return u"{0} {1} {2}".format(self.id, 'x', 'y')
```

Figure 8. Function that displays personal information if consent is given

```
def pseudo_chr_num(cod, alphabet=''):
    if not alphabet:
        alphabet = 'UdfiyGbEvXmDNWzCaMxKsAchVjSBtgIoZPFHrklwQJpuTeYnqRLO'

    cod = str(cod)
    cod = list(cod)

    for i in range(len(cod)):
        cod[i] = str(alphabet.find(cod[i])) + ':'

    cod = ''.join(cod)
    return cod
```

Figure 9. Function to convert string of chars to string of numbers and separator

```
def pseudo_num_chr(cod, alphabet=''):
    if not alphabet:
        alphabet = 'UdfiyGbEvXmDNWzCaMxKsAchVjSBtgIoZPFHrklwQJpuTeYnqRLO'

    cod = str(cod)

    if not cod.find(':'):
        return None

    cod = list(cod.split(':'))

    for i in range(len(cod)-1):
        cod[i] = str(alphabet[int(cod[i])])

    cod = ''.join(cod)
    return cod
```

Figure 10. Function to convert string of numbers and separator to string of chars

```
from dateutil.relativedelta import relativedelta

import datetime

date_of_birth = datetime.datetime(2000, 10, 10)

date_of_birth.date() - relativedelta(years=42)
datetime.date(1958, 10, 10)
```

Figure 11. How date data type can be pseudonymized

The database schema represented in Figure 12, and its design has an idea to have a starting base for the Membership application and with only one table containing the personal data.

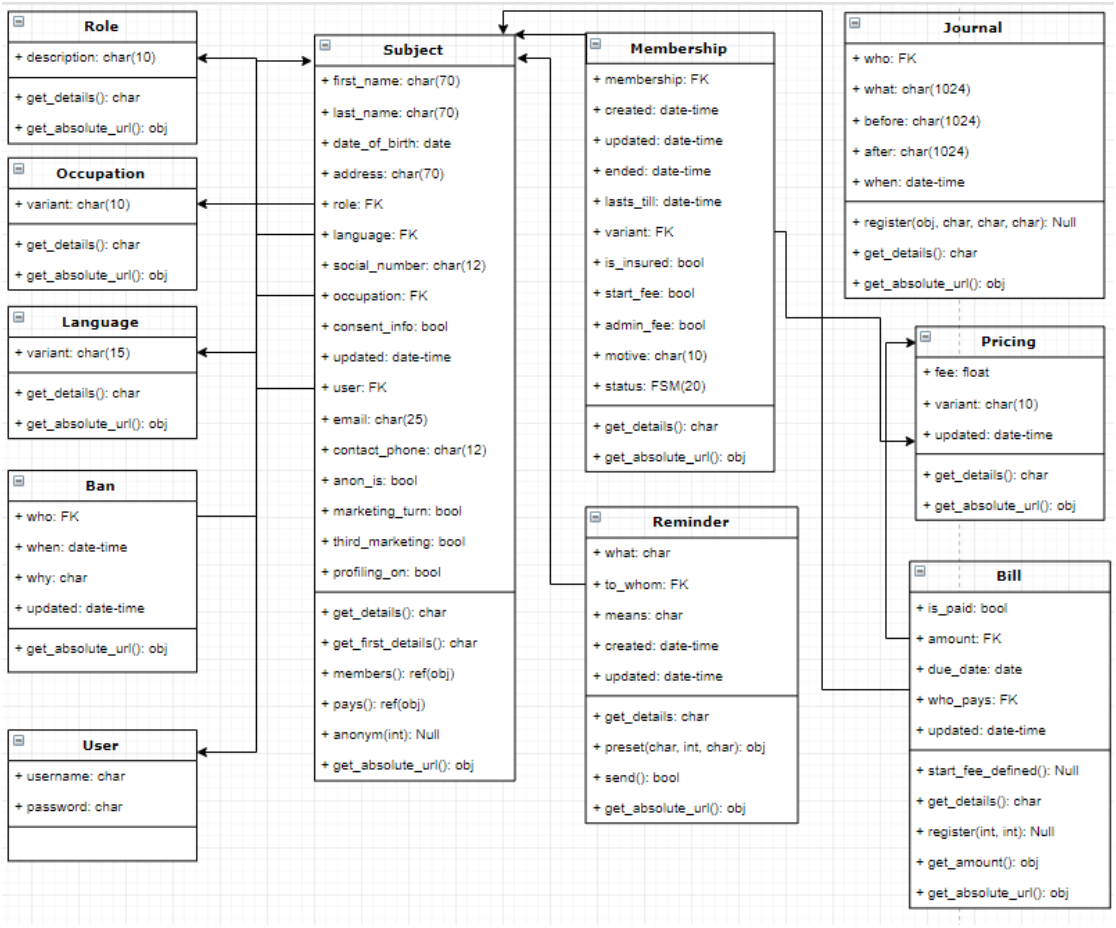

Figure 12. Application's database schema

Role – the table contains the roles of the users of the system:

- Admin – all rights given
- Personal – all rights as Admin has, except not erasure or send of data
- Subject – GDPR rights given
- Extra – not yet defined

Occupation – the table contains the status of the user of the Subject customer for discounting and other promoting marketing features:

- Senior
- Student
- Other

Language – the table contains customer specific spoken language for better customer support, and it has the service available in spoken language.

Ban – the table, currently, is used only to store identification of the customers that have not paid the fee for the service during the defined term (30 days of service payment, + 7 of debt collection term).

User – the table extends the Abstract User model from the *django.contrib.auth.models*, with fields only required for authentication of the customer into self-service to use from the service and with the ability to execute the GDPR rights.

Subject – the table containing customer related personal data; it has marks for the anonymization, marketing usage in local market and in third, profiling. Later, those marks will be used in automatic processing scripts.

Membership – the table indicating the status of the service for the specific customer. Status is Finite State Machine Field, which means that the transition from one status to another is possible, however, not vice versa, in defined cases. Example schema of Finite State Machine is in Figure 13 (adapted from Dmitry Stepanenko 2017).

Figure 13. Finite state machine schema

Reminder – the table containing the data used to remind the customer of their obligations in payment for the service.

Journal – the table is designed to provide a live record of the performed changes on the system. It is saved in the following format:

What was done, by whom, when, comparison to the previous state of the system.

Pricing – the table contains the predefined fee charged for the use of the service.

Bill – the table contains the information on confirmation of payment or that it has been refuted.

Django provides permission mechanism that defines what could be done, where and by whom. Permissions are divided into two types: object level and model level. As the model level indicates, the permission can be set to all objects related to the model. The object level permission defines concrete actions towards the related object. Example is shown in below figure (Figure 14).

```
class Meta:
        permissions = (
            ('can_view_self_service'), ('can_edit_subject')
        )
```

Figure 14. Permission set up in Meta class

Following figure (Figure 15) shows the way the Member is created in the system, the way it is giving consent, and its user profile for self-service. Create Subject Form performs data conditional check, as shown in Figure 16.

```
class CreateMemberView(generic.CreateView):
    def post(self, request, *args, **kwargs):
        subject_form = CreateSubjectForm(request.POST)
        membership_form = CreateMembershipForm(request.POST)
        sign_up_form = SignUpForm(request.POST)
        if subject_form.is_valid():
            if membership_form.is_valid():
                if sign_up_form.is_valid():
                    sign_up_form.save()
                    username = sign_up_form.cleaned_data.get('username')
                    raw_password = sign_up_form.cleaned_data.get('password')
                    user = User.objects.get(username=username)
                    user.set_password(raw_password)
                    user.save()
                    subject_form.instance.role_id = Role.objects.get(description='Subject').pk
                    subject_form.instance.user_id = sign_up_form.instance.pk
                    subject_form.save()
                    membership_form.instance.membership_id = subject_form.instance.pk
                    membership_form.instance.status = Membership.STATUS_ACTIVE
                    membership_form.save()
                    Bill.register(
                        who_pays_id=subject_form.instance.pk,
                        amount_id=membership_form.instance.variant_id
                    )
                    Journal.register(
                        who=subject_form.instance,
                        what='create_member',
                        before=None,
                        after='{} {} {}'.format(subject_form.instance.pk, membership_form.instance.pk,
                                                sign_up_form.instance.pk)
                    )

                    return render(request, 'mgmt/success_request.html', {'action': 'Creation member'})

    def get(self, request, *args, **kwargs):
        subject_form = CreateSubjectForm()
        membership_form = CreateMembershipForm()
        sign_up_form = SignUpForm()

        return render(request, 'mgmt/create_member.html', {'subject_form': subject_form,
                                                           'membership_form': membership_form,
                                                           'sign_up_form': sign_up_form, })
```

Figure 15. Form to register data subject to the system

```python
class CreateSubjectForm(ModelForm):
    class Meta:
        model = Subject
        exclude = ['role']

    def clean(self):
        date = datetime.date(datetime.now())
        minus_16 = date - relativedelta(years=16)

        first_name = self.cleaned_data['first_name']
        last_name = self.cleaned_data['last_name']
        date_of_birth = self.cleaned_data['date_of_birth']
        consent_given = self.cleaned_data['consent_given']

        if not consent_given:
            raise ValidationError(_('You need give us your consent about transforming of your personal data to deliver'
                                    ' you our service'))
        if len(first_name) > 102:
            raise ValidationError(_('Too long first name'))
        if len(last_name) > 102:
            raise ValidationError(_('Too long last name'))
        if date_of_birth >= date:
            raise ValidationError(_('You can\'t be born today or in the future :/'))
        if date_of_birth >= minus_16:
            raise ValidationError(_('You are too young!'))
```

Figure 16. Subject's provided data validation

If somehow consent is not given in time of registration of subject in the system, the form validation will give a corresponding error about this. Validation considers the amount of chars in the first and last names of the subject; less than 102 should be enough. Another validation considers the date of birth set as today or in the future; it could also be checked with delta if the date of birth is too old. It is better to have less hassle about non-adults and exclude them from the client base, as they are the special category of data subjects. Thus, clients below 16 years old will not be registered into the system. In a further application extension, consent of the non-adult could be asked from the beholder of parental responsibility via email way of communication.

5.4 Flow of data

5.4.1 Rest

REST Framework allows to create an API to interact with different systems and interfaces. Rest authentication policies allow to send and receive data with a proper authentication mechanism, pair of login and password or with a token. The request methods are the following: GET, POST, PUT, DELETE. They allow to perform retrieval, creation, update, or deletion action respectively. Response on the request contains data in JSON format and the status from which is the result of the sent request that it is clear. The following HTTP status codes are the most frequently met:

- HTTP_201_CREATED
- HTTP_202_ACCEPTED
- HTTP_400_BAD_REQUEST
- HTTP_401_UNAUTHORIZED
- HTTP_405_METHOD_NOT_ALLOWED

With extension of generics.ListCreateAPIView the list of existing data subjects is shown and with generics.RetrieveUpdateDestroyAPIView the individual data subject is listed. To restrict the access to retrieve the data it is enough to set "permission_classes" variable to "(permissions.IsAuthenticated,)". To get all subjects – queryset = Subject.objects.all () is sufficient.

Rest framework extends Django's libraries plus provides the validation of the received data with the help of serializers. Serializers have a similar behaviour as Django's forms: if the data are invalid, then raise an error. In the subject list the serializer definition is set with variable "serializer_class". The customized validation is easily achieved by extending the serializers.py classes.

URLs allow to navigate the views. DefaultRouter gives the possibility to wrap the URL of subjects with regular expression just like:

router = DefaultRouter()

router.register(r'subjects', views.SubjectViewSet)

5.4.2 Authentication

Authentication is a process of verification of the user identity in the system based on possessed knowledge. For authentication, the following steps are needed to consider:

1. Establishing a connection with urllib3 library's Pool Manager.
2. If in settings.py Is Authenticated permission is set, it requires providing the credentials in the header part of the request.

3. When those steps have been done, then type of request method is defined and final request sent. The response object will contain data and status code.

Additionally, to send data, a body argument is provided in the request method.

Figure 17 presents an example code in Django with urllib3 library.

```
credentials = 'root:queryset'
url = "http://localhost:8008/"
http = urllib3.PoolManager()
method = 'GET'
credentials = 'username:password'
header = urllib3.util.make_headers(basic_auth=credentials)
response = http.request(method, url=url, headers=header)
```

Figure 17. Example of basic authentication with urllib3

Secure HTTP connection is achieved with the HTTPS Connection Pool, as shown in Figure 18. Dots can be replaced with the exampled in Figure 17 http.request method arguments.

```
from urllib3.connectionpool import HTTPSConnectionPool

conn = HTTPSConnectionPool('httpbin.org', ca_certs='/etc/pki/cert.pem', cert_reqs='REQUIRED')

response = conn.request(...)
```

Figure 18. Secure authentication with urllib3

5.4.3 API

Application programming Interface is a way of communication which consists of requests and responses from the client to the server side. With API, it becomes possible to send data from one server to another, which means from one controller to another one if the details of API are known, of course.

5.5 Processing

The used platform does not have a **cronjob**, in contrast to Linux. Windows XP, seven editions have such called crontab analogue. However, those editions lack, nowadays, support from Microsoft, thus, there are no up-to-date security patches or Windows 8 and later schtasks utility. Nevertheless, it has Celery which brings the distributed task capability into equation.

5.5.1 Tasks

Celery is a one solution as a task processing mechanism in Windows. The configuration is done in celery module (Figure 19). Tasks definition lay in tasks.py module (Figure 20). A configuration of the task that is scheduled to run once every 21 hours, as the assumptions of low activity of the users, illustrated in Figure 21 and explicit definition is shown in Figure 22, “Do not store data unless you need it” example. To run such tasks, it is necessary to use the following commands on Windows, running first the broker and its beat (Periodic Tasks 2016):

celery -A mgmt beat

celery -A mgmt worker -P eventlet -E

```
from __future__ import absolute_import, unicode_literals
import os
from celery import Celery
import django


os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'untitled.settings')
django.setup()

app = Celery('untitled',
             broker='pyamqp://guest@localhost//',
             backend='amqp://',
             include=['mgmt.tasks'])

app.config_from_object('django.conf:settings', namespace='CELERY')

app.conf.broker_heartbeat = 0

app.autodiscover_tasks()
```

Figure 19. Celery task configuration

```
import logging
from datetime import datetime

from dateutil.relativedelta import relativedelta
from django.utils.timezone import make_aware

from mgmt.celery import app
from mgmt.models import Bill, Reminder, Membership, Subject

logger = logging.getLogger(__name__)


@app.task(bind=True)
def debug_task(self):
    print('Request: {0!r}'.format(self.request))


@app.task
def test(arg):
    print(arg)
    return arg
```

Figure 20. Celery task examples

```
@app.on_after_configure.connect
def setup_periodic_tasks(**kwargs):

    app.add_periodic_task(
        crontab(hour=21),
        check_consent.s(),
        name='check consent'
    )
```

Figure 21. Setting up the periodic task for checking if consent is not given of data subject

```
@app.task
def check_consent():
    subjects = Subject.objects.filter(consent_given=False)
    for subject in subjects:
        Subject.anonym(subject.pk)
        Membership.transit(subject.pk, 'deactivate', 'Self')
    logger.info('Checked consent')
    return 'Checked consent'
```

Figure 22. Definition of task for checking if consent is not given of data subject

Fields for special processing where set on the Subject table. In below example it is shown how to filter subjects that have consented their profiling, Figure 23. The use of an iterator is recommended to eliminate the chance of OOM (Out of Memory) error.

```
subjects_allowed_profiling = Subject.objects.filter(profiling_on=True)
for sap in subjects_allowed_profiling.iterator():
    # do something
    pass
```

Figure 23. Filtering on allowed profiling for data subjects

5.5.2 Logging of actions

It is better to use a trailing mechanism in the case of breach to find the guilty person. The following overriding could be used as shown in Figure 24:

```
def save(self, *args, **kwargs):
    before = Subject.objects.get(id=self.pk)
    after = self.initial_value
    Journal.register(
        who=self.user,
        what='save',
        before=before.get_details(),
        after=self.subject,
    )
    super(Subject, self).save(*args, **kwargs)
```

Figure 24. Overriding of base save method to inject logging functionality

5.6 Rights execution

In the given application “the provisioning of information from the controller to the data subject” is not required. It is not applied due to the demonstration purposes of the application and because the personal data are generated randomly.

5.6.1 Right to accurate the inaccurate data

Django framework provides the class based views and function based views. The extension of UpdateView can be used to modify the object of the Subject model in a human-readable format as illustrated in Figure 25.

```
class SubjectUpdate(UpdateView):
    model = Subject
    fields = '__all__'
    template_name = './mgmt/template_form.html'
```

Figure 25. Class based view for Subject model

5.6.2 Right to rectification

With the help of DeleteView the data could be deleted as shown in Figure 26. Context data allow to define which data are being rendered in the template. In the example the header of the HTML is a “subject”, same in “what” context. The details of the subject are retrieved with get_details() function.

```
class SubjectDelete(DeleteView):
    model = Subject
    success_url = reverse_lazy('subjects')
    template_name = './mgmt/template_confirm_delete.html'

    def get_context_data(self, *args, **kwargs):
        context = super(eval('{}Delete'.format(self.model.__name__)), self).get_context_data(**kwargs)
        context['header'] = self.model.__name__
        context['what'] = self.model.__name__
        context['exact'] = self.model.objects.get(pk=self.kwargs['pk']).get_details()
        return context
```

Figure 26. Basic DeleteView Class to delete personal data

5.6.3 Right to access

Compliance with the right to access personal data and related data achieved with the simple Get method sent from the client-side. The main models are: Subject, Membership, Bill. The function generates the text file from converting the Django's query set into a data dictionary format and excluding unnecessary fields such as id, user, role (Figure 27).

```
class ReportView(generic.CreateView):
    def post(self, request, *args, **kwargs):
        pass

    def get(self, request, *args, **kwargs):
        subject = get_object_or_404(Subject, pk=self.kwargs['pk'])
        exclude_fields_subject = ['id', 'user', 'role', 'language', 'occupation']
        exclude_fields_membership = ['id', 'variant', 'membership']
        exclude_fields_bill = ['id', 'who_pays']
        membership = subject.membership.first()
        if subject.pays:
            bill = subject.pays
        else:
            bill = Bill.objects.get(who_pays=subject)
        # write
        file = 'report{}.txt'.format(str(datetime.now()).replace(':', '.'))
        with open(file, 'w') as f:
            f.write('Your personal data\n')
            for k, v in model_to_dict(subject).items():
                if k not in exclude_fields_subject:
                    f.write(u'{}: {}\n'.format(k, v))

            f.write('\nMembership\n')

            for k, v in model_to_dict(membership).items():
                if k not in exclude_fields_membership:
                    if k == 'ended' and v is None:
                        v = ''
                    f.write(u'{}: {}\n'.format(k, v))

            for k, v in model_to_dict(bill).items():
                if k not in exclude_fields_bill:
                    f.write(u'{}: {}\n'.format(k, v))
        # download
        if os.path.exists(file):
            with open(file, 'rb') as fh:
                response = HttpResponse(fh.read(), content_type="application/vnd.ms-excel")
                response['Content-Disposition'] = 'inline; filename=' + os.path.basename(file)

                Journal.register(
                    who=subject,
                    what='get_report',
                    before=None,
                    after='{}'.format(subject.id)
                )

                return response
        raise Http404
```

Figure 27. Retrieveal of used data of data subject

5.6.4 Right to be forgotten

The execution can be easily achieved with the form sent as shown in Figure 28. The view function is presented in Figure 29. But The subject cannot be anonymized if the invoice has not been paid as shown in Figure 30. The effect of anonymization is shown in Figure 31.

```python
class AnonymousForm(forms.Form):
    anonymize = forms.BooleanField(help_text='the Right to Be Forgotten')

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        for field in self.fields:
            help_text = self.fields[field].help_text
            self.fields[field].help_text = None
            if help_text != '':
                self.fields[field].widget.attrs.update(
                    {'class': 'has-popover', 'data-content': help_text, 'data-placement': 'right',
                     'data-container': 'body'})
```

Figure 28. Form used to provide the right to be forgotten

```python
class AnonView(generic.CreateView):
    msg = """By ticking this check box we will, if possible, erase your personal data,
             deactivate your subscription,
             Thank you for using our service"""

    def post(self, request, *args, **kwargs):
        anon_form = AnonymousForm(request.POST)
        if anon_form.is_valid():
            Subject.anonym(kwargs['pk'])
            Membership.transit(kwargs['pk'], where='deactivate', motive='Anon')
            Journal.register(
                who=anon_form.instance,
                what='register',
                before=None,
                after='{}'.format(request.user.pk)
            )
            return render(request, 'mgmt/success_request.html', {'action': 'Anonymous operation'})
        return render(request, 'mgmt/opt_out.html', {'action_form': anon_form, 'msg': self.msg, 'button': 'Request'})

    def get(self, request, *args, **kwargs):
        get_object_or_404(Subject, pk=kwargs['pk'])
        anon_form = AnonymousForm()
        return render(request, 'mgmt/opt_out.html', {'action_form': anon_form, 'msg': self.msg, 'button': 'Request'})
```

Figure 29. View class to handle anonymization form

```python
@classmethod
def anonym(cls, pk=None):
    subject = cls.objects.get(pk=pk)
    if not subject.pseud_od:
        if subject.pays.is_paid and subject.members.status != Membership.STATUS_BANNED:
            subject.first_name = '*'
            subject.last_name = '*'
            subject.date_of_birth = datetime(1900, 1, 1)
            subject.address = 'X as 0'
            subject.social_number = 'ABCDEF*GHIJ'
            subject.consent_given = False
            subject.pseud_od = True
            subject.email = 'noni@noni.com'
            subject.save()
```

Figure 30. Function used to erase personal data, or fake it

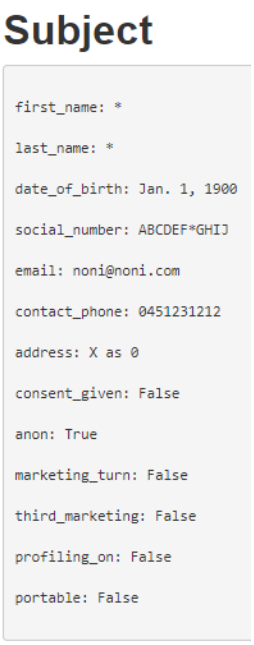

Figure 31. View of anonymized subject

5.6.5 Right to restriction of processing

The ease of withdrawal of consent is illustrated in the code snippet in Figure 32, in the browser it shown as presented in Figure 33. The general function to decide which FSM function to call can be seen in Figure 34. The FSM function to switch between states is illustrated in Figure 35 (Building for Flexibility Using Finite State Machine in Django 2017).

```
class OptOutView(generic.CreateView):
    msg = """By un-ticking this check box we will stop processing your data, and we will deactivate your
        subscription, if possible, otherwise, until last bill is paid, to secure your obligations towards us,
        Thank you for using our service"""

    def post(self, request, *args, **kwargs):
        subject = get_object_or_404(Subject, pk=self.kwargs['pk'])
        opt_out_form = OptOutForm(request.POST)
        if opt_out_form.is_valid():
            subject.consent_given = opt_out_form.cleaned_data['consent_given']
            subject.save()
            Membership.transit(subject.pk, 'deactivate', 'Self')
            return render(request, 'mgmt/success_request.html', {'action': 'Opt-out of subject'})
        return render(request, 'mgmt/opt_out.html', {'action_form': opt_out_form, 'msg': self.msg, 'button': 'Opt-Out'})

    def get(self, request, *args, **kwargs):
        subject = get_object_or_404(Subject, pk=self.kwargs['pk'])
        opt_out_form = OptOutForm(initial=model_to_dict(subject))
        return render(request, 'mgmt/opt_out.html', {'action_form': opt_out_form, 'msg': self.msg, 'button': 'Opt-Out'})
```

Figure 32. Application’s opt-out view (code)

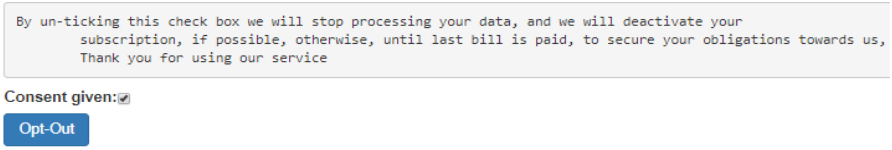

Figure 33. Application's consent withdrawal (opt-out)

```
@classmethod
def transit(cls, ref=None, where=None, motive=None):
    membership = cls.objects.get(membership_id=ref)
    if where == 'deactivate' and motive == cls.MOTIVE_SELF or cls.MOTIVE_ANON:
        if membership.status == cls.STATUS_ACTIVE:
            membership.deactivate(motive=motive)
            membership.save()
    elif where == 'suspend' and motive == cls.MOTIVE_ILLNESS:
        if membership.status == cls.STATUS_ACTIVE:
            membership.suspend(motive=motive)
            membership.save()
    elif where == 'ban' and motive == cls.MOTIVE_DEBT:
        if membership.status == cls.STATUS_ACTIVE:···
    elif where == 'reactivate' and motive == cls.MOTIVE_HEALTHY:
        membership.reactivate(motive=motive)
        membership.save()
```

Figure 34. Application's general function for status transition

```
@transition(field=status, source=STATUS_ACTIVE, target=STATUS_DEACTIVATED)
def deactivate(self, motive=None):
    now_aware = make_aware(datetime.now())
    self.ended = now_aware
    self.lasts_till = now_aware
    self.motive = motive
```

Figure 35. Application's membership deactivation

The give of consent functionality could be extended further to specify if the specific category of data could be used, e.g. specific such as racial or ethnic origin, political opinions, religious, philosofical beliefs or trade union membership, and genetic and biometric and other (European Parliament and the Council of the European Union 2016, Article 9).

5.6.6 Right to data portability

Data transmission is achieved with the REST API. The steps to do before sending the data are:

- To ensure that the customer has requested the transfer;
- Deactivation of membership takes place if the constraints are met, mainly this means all bills are paid;

- Open secure connection, and post data;
- The notification should be sent to the customer in case of success or failure of transmission.

Data should be sent over HTTPS. HTTPS uses the TLS (Transport Layer Security) or SSL (Secure Socket Layer), in another word, the asymmetric PKI system. PKI uses a public and private key for encryption and decryption. A private key belongs to recipient and sender, as public can be disclosed. The data can be encrypted with private key and decrypted with public, this also works vice-versa.

5.6.7 Right to object

The marketing purposes could be imposed only on data subjects who have explicitly given consent for such purposes. To achieve that, the special mark could be added and filtered with automatic commands.

Communication with data subject

As the application uses only registered users with electronic means, all communication towards customer about processing of personal data, requests from data subjects and its execution and response to those will be communicated via electronic means. The electronic email is used as one way of communication. Django provides a built-in method send_email under django.core.mail, so just two more steps are needed to do, the setup of configuration of email service provider (Figure 36). To send mail the following function is called:

send_email(header, content, sender's email, list of recipients)

```
# EMAIL CONFIG

EMAIL_HOST = 'email service provider'
EMAIL_PORT = 123
EMAIL_HOST_USER = 'username at service provider'
EMAIL_HOST_PASSWORD = 'password at service provider'
EMAIL_USE_TLS = True
```

Figure 36. Configuration to send email

6 Conclusion

The objective of the thesis was to analyse the Genereal Data Protection Regulation and to develeop the Web application. Some articles gave a help to me to briefly understand the concept of the GDPR and what side effects it brings. Nevertheless, they did not specify some exemptions defined in the GDPR, such as public interests, statistical or research purposes. The most important task to resolve for me was to consider the privacy, security of personal data and related data, how the solution can be done. I have learned how to write the code and use tricks to meet the requirements set in the GDPR and how to build a solid Django application. I have designed the database structure to minimize the spread of personal data across different tabes and to conclude the data in one place. Considering the data minimisation principle (I don't know why GDPR is so funny, but it is 2018; What is GDPR? 2018).

There are few examples on the internet how to be GDPR complaint in Django. However, they are just fragments, the do not represent the application wholly, but partially. The implemented application provides the service and compliance with the Regulation. The the requirements can be easily achieved in the new-coming solutions. The development of the solution could be concentrated firstly on the implementation of the requirements of the GDPR, later – on the implementation of other features. One should be clear about the purpose of collecting personal data and one should not store data if there is not need for it. Lastly, be aware of human privacy.

The application works as intended, but I would like to implement and deploy a real life case in the future. I have dropped the handling of repetitive customers inquries about the rights. That could be achieved with the extra table CustomerInquiry, for example. Which would store when the inquiry was done, and if so, disable the button of access to data, for example.

The gap when the legislation was created and when it came into force was long, there are a few cases of the GDPR violations or accusations of it, which can only

prove that IT giants need to to take care of their products in a reasonable time. Below could be found the examples of such cases in big companies:

Facebook – (Facebook Faces Potential $1.63 Billion Fine in Europe Over Data Breach 2018).

Google – (Google already sued for $3.7 billion for alleged Android GDPR violations 2018).

Yahoo, Ebay, Equifax – (GDPR: The Biggest Data Breaches and The Shocking Fines (That Would Have Been) 2018).

There are appeared new services to help with the GDPR compaint. For example, a cookie bot, it analysis if one's web site is complaint. The service can be found via the link: https://www.cookiebot.com/en.

References

14. 12. 1980. *OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data*. Accessed on 14 July 2018. Retrieved from: http://www.oecd.org/sti/ieconomy/oecdguidelinesontheprotectionofprivacyandtransborderflowsofpersonaldata.htm

Android Authority. 25. 05. 2018. C. Scott Brown. *Google already sued for $3.7 billion for alleged Android GDPR violations.* Accessed on 25 September 2018. Retrieved from: https://www.androidauthority.com/google-gdpr-violations-869642

Dmitry Stepanenko. 07. 06. 2017. Distillery. *Building for Flexibility Using Finite State Machine in Django.* Accessed on 10 June 2018. Retrieved from: https://medium.com/@distillerytech/building-for-flexibility-using-finite-state-machines-in-django-2e36ddbd7708

Docs Celery Project. 2016. Ask Solem. *Periodic Tasks.* Accessed on 11 June 2018. Retrieved from: http://docs.celeryproject.org/en/latest/userguide/periodic-tasks.html

Europa. Communication department of the European Commission on behalf of the EU institutions. *Institutions bodies*. Accessed on 30 October 2018. Retrieved from: https://europa.eu/european-union/about-eu/institutions-bodies_en

European Parliament and the Council of the European Union. 23. 11. 1995. *Directive 95/46/EC*. Accessed on 5 June 2018. Retrieved from: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:31995L0046

European Parliament and the Council of the European Union. 4. 5. 2016. *Regulation (EU) 2016/679.* Accessed on 3 April 2018. Retrieved from: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

Forbes. 11. 06. 2018. Bernard Marr. *GDPR: The Biggest Data Breaches and The Shocking Fines (That Would Have Been).* Accessed on 25 September 2018. Retrieved from: https://www.forbes.com/sites/bernardmarr/2018/06/11/gdpr-the-biggest-data-breaches-and-the-shocking-fines-that-would-have-been/#270615c56c10

MDN web docs. 05. 01. 2018. Hamish Willee. *Django web application security.* Accessed on 14 June 2018. Retrieved from: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/web_application_security

The Verge. 25. 05. 2018. Sarah Jeong. *I don't know why GDPR is so funny, but it is.* Accessed on 12 September 2018. Retrieved from: https://www.theverge.com/2018/5/25/17395210/gdpr-spam-funny-memes

The Wall Street Journal. 30 September 2018. Sam Schechner. *Facebook Faces Potential $1.63 Billion Fine in Europe Over Data Breach.* Accessed on 4 October 2018. Retrieved from: https://www.wsj.com/articles/facebook-faces-potential-1-63-billion-fine-in-europe-over-data-breach-1538330906

Varonis. *What is GDPR?* Accessed on 14 March 2018. Retrieved from: https://www.varonis.com/guides/what-is-gdpr