GDPR developer guide

The CNIL publishes a GDPR guide for developers

CNIL.
COMMISSION NATIONALE
INFORMATIQUE & LIBERTÉS

Introduction

In order to assist web and application developers in making their work GDPR-compliant, the CNIL has drawn up a new guide to best practices under an open source license, which is intended to be enriched by professionals.

This guide is published under license GPLv3 and under open license 2.0 (explicitly compatible with CC-BY 4.0 FR). You can freely contribute to its redaction.

The French version is the authentic version of this guide.

Is this guide for developers only?

This guide is mainly aimed at developers working alone or in teams, team leaders, service providers but also at anyone interested in web or application development.

It provides advice and best practices, and thus gives useful keys to understand the GDPR for every stakeholder, regardless of the size of their structure. It can also stimulate discussions and practices within the organisations and in customer relationships.

What does the guide contain?

This guide is divided into **16 thematic sheets** which cover most of the needs of developers at each stage of their project, from the preparation of the development to the use of analytics.

The General Data Protection Regulation (or GDPR) specifies that the protection of the rights and freedoms of natural persons requires that **"appropriate technical and organisational measures be taken to ensure that the requirements of this Regulation are met"** (Recital 78).

The determination of these measures is necessarily **related to the context of the processing operations put in place**, and the controller (the public or private entity processing personal data) must therefore ensure the security of the data it is called upon to process.

The good practices in this guide **are therefore not intended to cover all the requirements of the regulations nor to be prescriptive**, they provide a first level of measures to take into account privacy protection issues in IT developments that are intended to be applied to all data processing projects. Depending on the nature of the processing carried out in certain cases, additional measures will have to be implemented in order to fully comply with the regulations.

How can I contribute to this guide?

This guide is available in two versions:

- A web version on the CNIL website and in the tab the "Releases" tab of this repository;
- This GitHub version, which offers the possibility for everyone to contribute.

The contribution is done in a few steps:

- Register on Github;
- Go to the project page;
- You can:
  - Use the "Issue" tab to open comments or participate in the discussion
  - Use the "Fork" option to make your own modifications and propose their inclusion via the "Pull Requests" button.

Your contribution proposal will be examined by the CNIL before publication. The web version of the RGPD developer's guide will be regularly updated.

Table of contents

Sheet n°0: Develop in compliance with the GDPR

Whether you work alone, are part of a team developing a project, manage a development team, or are a service provider carrying out developments for third parties, it is essential to ensure that user data and all personal data processing are suffisiently protected throughout the lifecycle of the project.

The following steps will help you in the developing privacy-friendly applications or websites:

1. **Be aware of the GDPR core principles**. If you work in a team, we recommend that you identify a person responsible for monitoring compliance. If your company has a Data Protection Officer (DPO), then that person is a key asset in understanding and meeting the GDPR obligations. The appointment of a DPO may also be mandatory in some cases, for example if your programs or applications process so-called "sensitive" data (see examples) on a large scale or conduct regular and systematic monitoring on a large scale.

2. **Map and categorize the data and processing in your system**. Accurately mapping the data processing performed by your program or application will help you ensuring that they comply with legal requirements. Keeping a record of processing activities (an example of which can be found on the CNIL website), allows you to have an overall view of these data, and to identify and prioritize the associated risks. Indeed, personal data may be present in unexpected places such as in server logs, cache files, Excel files, etc., and may be stored in a number of different places. Such record-keeping is mandatory in most cases.

3. **Prioritize the required actions**. On the basis of the data processing registry, identify the required actions to comply with the obligations of the GDPR in advance of the development and prioritize the attention points with regard to the risks for the data subjects by the processing. These points of attention concern in particular the necessity and types of data collected and processed by your software, the legal basis on which your data processing operations are based, the information mentions of your software or application, the contractual clauses binding you to your contractors, the terms and conditions for exercising rights, the measures implemented to secure your processing.

4. **Manage the risks**. When you identify that a processing of personal data is likely to create high risks for data subjects, make sure that you manage those risks appropriately in the context. A Privacy Impact Assessment (PIA) can help you manage those risks. The CNIL has developed a method, model documents and a tool that will help you to identify those risks, as well as a catalogue of good practices that will assist you in implementing measures to address the identified risks. Furthermore, a privacy impact assessment is mandatory for all processing operations that are likely to create high risks to the rights and freedoms of data subjects. The CNIL proposes, on its website, a list of types of processing operations for which a DPA is required or not.

5. **put in place internal processes** to ensure compliance during all development stages, ensure that internal procedures guarantee that data protection is taken into account in all aspects of your project and into all events that may occur (e.g. security breach, requests for rectification or access fulfillment, modification of data collected, change of provider, data breach, etc.). The requirements of the governance label (even if this one no longer granted by the CNIL since the entry into force of the GDPR) can constitute a useful basis of inspiration to help you set up the necessary organization.

6. **Document developments compliance** to prove your compliance with the GDPR at all times: the actions performed and the documents produced at each stage of development must be mastered. This implies in particular a regular review and update of the documentation of your developments so that it is constantly consistent with the features deployed on your program.

The CNIL website provides numerous practical files which will assist you in setting up law-abiding treatments according to your sector of activity.

Sheet n°1: Identify personal data

Understanding the notions of "personal data", "purpose" and "processing" is essential for the development of law enforcement and user data. In particular, be careful not to confuse "anonymisation" and "pseudonymization", which have very precise definitions in the GDPR.

Definition

- The notion of **personal data** is defined in the General Data Protection Regulation (GDPR) as "any information relating to an identified or identifiable natural person (referred to as "data subject")". It covers a broad scope that includes both directly identifying data (e.g. first and last name) and indirectly identifying data (e.g. telephone number, license plate, terminal identifier, etc.).

- Any operation on this type of data (collection, recording, transmission, modification, dissemination, etc.) constitutes **processing within the meaning of the GDPR** and must therefore meet the requirements laid down by that regulation. Such processing operations must be lawful and have a specified purpose. The personal data collected and processed must be relevant and limited to what is strictly necessary to achieve the purpose.

Examples of personal data

- Where they relate to natural persons, **the following data are personal data**:
  - Surname, first name, pseudonym, date of birth;
  - photos, sound recordings of voices;
  - fixed or mobile telephone number, postal address, email address;
  - IP address, computer connection identifier or cookie identifier;
  - Fingerprint, palm or venous network of the hand, retinal print;
  - License plate number, social security number, ID number;
  - Application usage data, comments, etc...
- **Identification of natural persons can be carried out**:
  - from a single piece of data (examples: surname and first name);
  - from crossing of a set of data (example: a woman living at such and such an address, born on such and such a day and member of such and such an association).

- Some data are considered **particularly sensitive**. The GDPR prohibits the collection or use of such data, unless, in particular, the data subject has given his/her express consent (active, explicit and preferably written consent, which must be free, specific and informed).

- These requirements concern the following data:

  - data relating to the **health of individuals**;
  - data concerning **sexual life** or **sexual orientation**;
  - data revealing an alleged **racial** or **ethnic** origin;
  - political opinions, religious beliefs, philosophical beliefs or trade union membership;
  - **genetic** and **biometric data used for the purpose of uniquely identifying an individual**.

Anonymisation of personal data

- An **anonymisation process of personal data** aims at making impossible to identify individuals within data sets. It is therefore an irreversible process. When this anonymisation is effective, the data are no longer considered as personal data and the requirements of the GDPR are no longer applicable.

- By default, we recommend that you **never consider raw datasets as anonymous**. Anonymisation results from processing personal data in order to irreversibly prevent identification, whether by:

  - *singling out*: it is not possible to isolate some or all records which identify an individual in the dataset;
  - *linkability*: the dataset does not allow to link at least, two records concerning the same data subject or a group of data subjects;
  - *inference*: it is not possible to deduce, with significant probability, the value of an attribute from the values of a set of other attributes.

- These data processing operations imply in most cases a **loss of quality on the produced dataset**. The Article 29 Working Party (Art. 29 WP) opinion on anonymisation techniques describes the main anonymisation techniques used today, as well as examples of datasets wrongly considered anonymous. It is important to note that anonymisation techniques have short comings. The choice to anonymize or not the data as well as the selection of an anonymisation technique must be made on a case by case basis according to contexts of use and need (nature of the data, usefulness of the data, risks for people, etc.).

Pseudonymization of personal data

- **Pseudonymization is a compromise between retaining raw data and producing anonymized datasets**.

- It refers to the processing of personal data in such a way that **data relating to a natural person can no longer be attributed without additional information**. The GDPR insists that this additional information must be kept separately and be subject to technical and organisational measures to avoid re-identification of data subjects. Unlike anonymisation, pseudonymization can be a reversible process.

- In practice, a pseudonymization process consists of **replacing directly identifying data (surname, first name, etc.) in a dataset with indirectly identifying data** (alias, number in a filing system, etc.) in order to reduce their sensitivity. They may result from a cryptographic hash of the data of individuals, such as their IP address, user ID, e-mail address.

- Data resulting from pseudonymization are considered as **personal data and therefore remain subject to the obligations of the DPMR**. However, the European Regulation encourages the use of pseudonymization in the processing of personal data. Moreover, the GDPR considers that pseudonymization makes it possible to reduce the risks for data subjects and to contribute to compliance with the Regulation.

Sheet n°2: Prepare your development

The principles of personal data protection must be integrated into IT developments from the design phase onwards in order to protect the privacy of the people whose data you are going to process, to give them better control over their data and to limit errors, losses, unauthorised modifications or misuses of their data in applications.

Methodological choices

- **Put privacy protection at the center of your developments** by adopting a Privacy By Design methodology.
- If you use agile methods for your developments, consider **integrating security at the center of your process**. The ANSSI has made available a guide "digital security & agility" (in French only) which indicates how to conduct your developments within the framework of an agile method while taking into account the security aspects. Don't hesitate to draw inspiration from it.
- For any development aimed at the general public, **consider the privacy settings**, and in particular the default settings, such as the characteristics and user content visible by default.
- **Conduct a Privacy Impact Assessment (PIA)**. For certain processing operations it is mandatory. In other cases it is a good practice that will allow you to identify and deal with all the risks upstream of your developments. The CNIL has a special section on its website and provides a free software dedicated to this type of analysis.

Technological choices

Architecture and features

- **Include privacy protection, including data security requirements, at the design stage of the application or service**. These requirements should influence architecture choices (e.g. decentralized vs. centralized) or functionality (e.g. short term anonymization, data minimization). The default settings of the application must meet minimum security requirements and comply with the law. For example, the default complexity of passwords must comply at least with the CNIL recommendation on passwords.
- **Maintain control of your system**. It is important to keep control of your system, both to ensure proper operation and to guarantee a high level of security. Keeping a system simple allows you to understand precisely how it works and to identify its weak points. If a certain complexity is required, it is advisable to start with a simple, correctly designed and secure system. Then, it is possible to increase the complexity little by little, while continuing to secure the new features that are added.
- **Don't rely on a single line of defense**. In spite of all the steps taken to design a secure system, it may happen that some components added later may not be sufficiently secure. To minimize the risk to end users, it is advisable to defend the system in depth. For example, checking the data entered in an online form is part of the periphery defenses. If this defense is hijacked, database query protection can take over.

Tools and practices

- **Use programming standards that take into account safety**. Often, lists of standards, best practices or coding guides improving the security of your developments are already available. Ancillary tools can also be integrated into your integrated development environments ("**IDE**") in order to automatically check that your code complies with the various rules that are part of these standards or good practices. You can easily find lists of good practices for your favourite programming language on the Internet. For example here for C, C++ or Java. For web application development, specific good practice guides exist, such as those published by OWASP.
- **The technological choices are critical.** A few parameters need to be taken into account:

  - Depending on the field of application or functionality developed, one language or technology may be more appropriate than another.
  - Time-tested languages and technologies are safer. They have, in general, been audited to correct the most known vulnerabilities. However, you should be careful to use the latest versions of each of the technology building blocks you will be using.
  - You must avoid coding your final solution in a language you have just learned and not yet mastered. Otherwise, you expose yourself to an increased risk of a security flaw due to lack of experience.

- **Set up a secure development environment that allows versioning of the code** by following the dedicated sheet in this guide.

Sheet n°3: Secure your development environment

The security of production, development and continuous integration servers as well as developer workstations must be a priority because they centralize access to a large amount of data.

Assess your risks and adopt the appropriate security measures

- **Assess the risks** in the tools and processes used for your developments. Make an inventory of your existing security measures and define an action plan to improve your risk coverage. Appoint a person responsible for its implementation.
- Consider the risks on all the tools you use, including risks related to SaaS (Software as a Service) and collaborative tools in the cloud (such as Slack, Trello, GitHub, etc.).

Secure your servers and workstations in a homogeneous and reproducible way

- Lists of **recommendations** concerning the security of servers, workstations and internal networks are available in the sheets n° 5 to 8 of the **security of personal data guide** of the CNIL.
- Write a **document listing those measures and explaining their configuration** to ensure that security measures are implemented uniformly on servers and workstations. In order to reduce the workload, **configuration management tools**, such as Ansible, Puppet or Chef, can be used.
- Update servers and workstations, if possible automatically. You can set up a watchlist of the most important vulnerabilities, for example the NVD Data Feeds.

Put special emphasis on access management and traceability of operations

- Remember to document the management of your **SSH keys** (use of state of the art cryptography and key length algorithms, protection of private keys with a passphrase, key rotation). For examples of good practice, see the document on the secure use of (open)SSH.
- Encourage strong authentication on the services used by the development team.
- **Trace** access to your machines and, if possible, implement **automated log analysis**. In order to keep reliable traces, the use of generic accounts is to be avoided.

Sheet n°4: Manage your source code

Whatever the size of your project, it is highly recommended to use a source code management tool, such as a *version control system*, to track its different versions over time.

Set up your version control system efficiently, thinking about its security.

- A version control system is a software program that allows you to store **all your source code and associated files**, while keeping the **chronology of all changes** that have been made. A simple FTP server is not a version control system.
- Set up your environment correctly using the features offered by your version control system. It is recommended that you implement strong **authentication** and/or **authentication with SSH keys** at the beginning of your project.
- In addition, assign *levels of access* to your project to the users of your version control system and define for each level the corresponding **permissions** (for example, a "guest" level with limited read rights, a "developer" level with write rights, etc.).
- Make regular **backups** of your source code management system. In particular, remember to back up your main server where all changes are saved.
- Set up development procedures to work efficiently even if **several people are developing at the same time**. For example, you may decide not to work on the same branch (*master*), but to set up feature-based branches, which will be merged into the main branch as development progresses. Such development strategies are already well documented, for example in Git Flow. In addition, some version control systems offer to set up **protected branches** that prevent unauthorized changes to the files in these branches.

Be aware of your source code content.

- Implement **code quality metrics tools** that will scan your code as soon as it is *committed* to check its good quality. You can also add scripts to check these metrics in the version control system configuration: the *commit* will be cancelled if the source code is not of sufficient quality.
- Keep your secrets and passwords out of your source code repository:
  - in separate **files, which have not been *committed***. Remember to use special files from your version control system (such as *.gitignore* for *Git*) so that you don't *commit* sensitive files by mistake.
  - in **environment variables**, take care to check that environment variables are not accidentally written to *logs* or displayed when an application error occurs.
  - using **specific secret or configuration management software**.

Finally, if you need to include such data in your repository, consider **automatically encrypting/decrypting** the files using a *plugin* from your version control system (e.g. *git-crypt*).

- After a *commit* that contains personal or other critical data, don't forget to purge completely the source code repository: even after modification, the data may still be available in your repository history.
- Be careful before **publishing your source code online**. Review **its entire contents** to make sure that no personal data, passwords or other secrets are present, including the entire change history.

Examples of tools

- Unlike tools such as Subversion, which need a central server to run, the main version control systems (Git, Mercurial for example) are **decentralized**.

- For most of these tools, a **web interface and related tools** (bug management, wiki for your documentation, etc.) are provided. These solutions can either be accessible via the internet (GitHub, Bitbucket, etc.), or they can be integrated into your own servers.

Sheet n°5: Make an informed choice of architecture

When designing the architecture of your application, you must identify personal data that will be collected and define a path and life cycle for each of them. The choice of supporting assets (local storage, server, cloud service) is a crucial step, which must be adapted to your needs, but also to your technical knowledge. The registry and conduction a privacy impact assesment can assist you in this choice.

Examining life cycle of data and processes, from collection to erasure

- Represent and describe how the product generally works before starting your project, with a diagram of data flows and a detailed description of the processes carried out.
- When data is only **stored on the user's terminal** (local storage) or remains **confined on communication networks under the control of the user** (e.g. Wi-Fi or other local network), the main point of attention is data security. The duration for which data is stored and the actual deletion should be determined by the individuals.
- **When the data transits through online services**, the choice of hosting the data yourself or using a service provider must be made according to your security knowledge and the expected quality of service. Recognized cloud offerings may offer higher levels of security. However, they generate new risks that need to be mastered. Recommendations for companies planning to use Cloud computing services can guide at this selection stage.

In case of use of external hosting

- **Choose a service provider that ensures appropriate security and confidentiality measures and is sufficiently transparent**.
- **Make sure you know the geographical location of the servers that will host your data**. You may be required to transfer data outside the European Union (EU) and the European Economic Area (EEA). While data can move freely within the EU/EEA, transfers outside the EU/EEA are possible, provided that sufficient and appropriate level of data protection is ensured. The CNIL provides an on-site map showing the different levels of data protection in countries around the world.
- **If you need to host health data**, make sure that provider used is certified or approved for this activity.
- Other points to be aware of include:
  - the existence of an accessible security policy;
  - physical security and safety measures at the hosting site;
  - data encryption and other processes to ensure that the provider does not have access to the data entrusted to it;
  - the management of updates, the management of authorizations, the authentication of personnel and the security of application developments;
  - the easy reversibility/portability of data in a structured and commonly used format, on request and at any time.

Sheet n°6: Secure your websites, applications and servers

Any website, application or server must incorporate basic state-of-the-art security rules, not only on network communications but also on authentication and infrastructure.

Securing communication networks

- **Implement TLS version 1.2 or 1.3** (replacing SSL) on all websites and for data transmissions of your mobile applications, for example with LetsEncrypt, using only the most recent versions and checking its correct implementation.
- **Make the use of TLS mandatory** for all pages of your site and for your mobile applications.
- **Limit the communication ports** strictly necessary for the proper functioning of the installed applications. If access to a web server is only possible using the HTTPS protocol, only ports 443 and 80 of this server must be accessible, all other ports can be blocked by the firewall.
- **The OWASP has published on its website some cheatsheets** for exemple to correctly implement TLS or to secure a webservice.

Securing Authentications

- **Follow the CNIL recommendation on passwords**. In particular, remember to limit the number of access attempts.
- **Never store passwords in clear text**. Store them as a hash using a proven library, such as bcrypt.
- **If cookies are used for authentication**, it is recommended:
  - to force the use of HTTPS via HSTS;
  - to use the `secure` flag;
  - use the `HttpOnly` flag.
- **Test the cryptographic suites installed on the systems** and disable obsolete ones (RC4, MD4, MD5 etc.). Encourage the use of AES256. Read the OSWAP note on the subject.
- **Adopt a specific password policy for administrators**. Change the passwords, at least, each time an administrator leaves and in case of suspected breach. Encourage strong authentication when possible.
- **Limit access to administration tools and interfaces to qualified staff.** Encourage the use of lower-privilege accounts for day-to-day operations.
- **Remote access to administration interfaces should be subject to increased security measures.** For example, for internal servers, implementing a VPN with strong authentication of the user and the workstation he or she is using may be a good solution.

Securing infrastructures

- **Make backups, if possible encrypted and checked regularly**. This is especially useful in case of a ransomware attack on your systems as having backups for all your systems will be the only measure that will allow you to restore your systems.
- **Limit the size of the software stack used,** and for each element of the stack:
  - **Install critical updates** without delay by scheduling an automatic weekly check;
  - **Automate a vulnerability watch** by subscribing to the NVD Data Feeds for example.

- **Use vulnerability detection tools** for the most critical processes to detect possible security breach. Systems for detecting and preventing attacks on critical systems or servers can also be used. These tests must be conducted regularly and before any new software version is put into production.
- **Restrict or fordbid physical and software access to diagnostic and remote configuration ports.** For example, you can list all open ports using the *netstat* tool.
- **Protect the databases you make available on the Internet**, at least by restricting access as much as possible (for example, by IP filtering) and by changing the default password for the administrator account.
- In terms of database management, good practices include:
  - **using nominative accounts** for database access and create specific accounts for each application;
  - **revoking the administrative privileges** of user or application accounts to avoid modification to database structure (table, vues, process, etc);
  - having protection against SQL or script injection attacks;
  - encouraging at rest disk and database encryption.

Sheet n°7: Minimize data collection

You shall only collect personal data that is adequate, relevant and necessary in relation to the purposes for which they are processed, as defined at the time of collection.

Before collection, think about the different types of data you need to collect and try to limit your collection to what is strictly necessary.

- Think about the different **types of data** that will need to be collected before an application is implemented and **document** this thinking.
- If specific data is not **needed for a certain category of people**, do not collect it.
- Process and store data in a way that **reduces accuracy** (similar to pseudonymization). For example, store only the year of birth instead of a full date of birth if the application only needs the year.
- If collecting particularly sensitive data, such as health or criminal convictions data, be sure to collect only the **minimum required**. Due to the regulatory constraints, the simplest solution is still to **not collect them** if you can do without them.
- Minimize the amount of data collected also in the **log data** and do not store sensitive or critical data (health data, passwords, etc.).
- Some features may improve the user experience, but are **not strictly necessary for your application to work properly** (e.g. geolocation to simplify a geographic search). In this case, the end user must be able to **choose whether or not to use** this functionality. If he uses it, the data that you are led to collect for its operation must only be kept for the time strictly necessary for its operation and never be used for other purposes.
- Remember to associate **retention periods** for each category of data, depending on the purpose of the processing and the legal or regulatory obligations relating to their retention. Logs must also have a retention period. Document the defined retention durations. You will need to be able to justify them.

Once the data has been collected, set up automatic deletion mechanisms.

- Implement an automatic **purge** system at the end of the shelf life. You can also implement manual reviews of stored data on a periodic basis.
- To ensure complete erasure, erase **physically** all data that is no longer needed using specialized tools or by destroying the physical media.
- If the data is still useful, you can reduce its sensitivity by using **pseudomisation** or even **anonymisation** methods. In case of pseudonymization, these data remain subject to the regulations on personal data (see Sheet 1).
- Log the **automatic deletion procedures**. The corresponding logs can be used as a **proof of deletion** of a data item.

Sheet n°8: Manage user profiles

The way to manage profiles of your collaborators and your end-users must be thought out upstream of your developments. It consists in defining different access and authorization profiles so that each person can access only the data he or she actually needs.

Good practices for user management

- It all starts with the **use of unique and individual identifiers**, whether they are users of your application or collaborators in development.
- Make sure to **impose authentication** before any access to personal data, in accordance with the recommendations of the CNIL.
- To ensure that each person (user or collaborator) can only access to **data he or she actually needs**, your system must provide **differentiated data access management policies** (read, write, delete, etc.) according to people and needs. A global user profile management mechanism will allow you to group different rights according to a role exercised by a group of users within the application.
- The management of user profiles can be used along with **logging systems in order to trace activities, and detect anomalies or events related to security**, such as fraudulent access and misuse of personal data. These devices must not be used for any purpose other than ensuring the proper use of the computer system. Logs must also not be kept longer than necessary. In general, a period of six months is adequate.
- You can also plan code audits or penetration testing within your development environment to **ensure robustness of your profile management system**.

Streamline the management of clearance profiles

- Plan to **document or automate the movement of your collaborators**. For example, these procedures should lead the actions to be taken when people are no longer authorized to access a room or an IT resource, or at the end of their contract.
- Managing your users and collaborators implies **a regular review of the permission** according to the evolution of uses and organizational movements within your project. The use of directory services, such as *Lightweight Directory Access Protocol* (*LDAP*), will help you monitor these changes and allow you to refine your access strategies, for example by assigning roles based on usage profiles. This allows you to better respect the principle of least privilege.
- The use of "supreme" accounts (type *root*, administrator, etc.) has to be avoided for conventional operations, as it constitutes the keystone of your system and a privileged target for a possible external attacker. We recommend that you associate a strong password policy with it (10 to 20 characters or multi-factor) and that you limit the number of people with knowledge of it to the strictest necessary.
- **Favour the use of a password manager within your project** and the transition to strong authentication when possible. Avoid generic accounts shared by several people.

Sheet n°09: Control your libraries and SDKs

Do you use libraries, SDKs, or other software components written by third parties? Here are a few tips on how to integrate these tools while keeping control of your developments.

Make an informed choice

- **Assess the value of adding each dependency.** Some commonly used software bricks are only a few lines long. However, each added element is an increase in your system's attack surface. In the case where a single library offers several functionalities, integrate only the functionalities you actually need. By activating the minimum number of functionalities, you reduce the number of potential bugs that could occur.
- **Choose maintained software, libraries and SDKs:**
  - If you want to use free or open source software, try to choose projects or solutions with an active community, regular updates and good documentation.
  - If you use other types of solutions with commercial support, contractually ensure that the code will be maintained and updated for the life of your project.
- **Take privacy into account.** Some SDKs or libraries pay for themselves by using personal data collected from the applications or sites on which they are integrated. Make sure that such third parties comply with applicable laws regarding personal data, including a mechanism for obtaining user consent.
- **If you use cryptographic mechanisms, it is strongly discouraged to implement cryptographic algorithms or protocols yourself**, but rather try to choose cryptographic libraries that are maintained, recognized and easy to use.

Evaluate the selected elements

- **Read the documentation and change the default configurations**. It is important to know how your dependencies work. Third party libraries and SDKs often come with default configuration files, which are rarely changed due to lack of time, which causes many security holes.
- **Audit your libraries and SDKs.** Do you really know what all the libraries and SDKs you integrate do? What data is sent through these dependencies and to whom? This audit will allow you to determine the data protection obligations to be respected and to establish the responsibility of the actors.
- **Map you dependencies.** Third-party libraries and SDKs can also integrate other components: auditing their code will allow you to better map all your dependencies and to better act if a problem affects one of them. It is also recommended that you perform security audits of your third-party components and monitor them.
- **Beware of typosquatting and other malicious techniques.** Check the names of dependencies, as well as their own dependencies to avoid attacks. Do not copy and paste command lines from unknown sites.

Maintain libraries and SDKs

- **Use dependency management systems** (such as yum, apt, maven, pip, etc.) to maintain an up-to-date list of your dependencies.
- **Manage updates to your dependencies,** especially in the case of security updates that fix vulnerabilities. You must set up a documented procedure to manage and deploy them as soon as possible.
- **Be aware of the versions of libraries and SDKs at the end of support** that will no longer be maintained: try to find another solution (choose a new library, renew commercial support).

- **Check the status of open-source projects,** especially the change of domain or package ownership, some attacks using malicious updates of popular dependencies.

Sheet n°10: Ensure quality of the code and its documentation

It is essential to adopt good code-writing techniques as soon as possible. Code readability reduce the effort of maintenance and bug fixes over time for you and your (possibly future)collaborators.

Document code and architecture

- Documentation is sometimes left out during development, due to lack of time or visibility on the project. However, it is **crucial for the maintainability of your project**: it allows you to understand how the code works globally, and to know which parts of the code are affected by a modification.
- **Document the architecture, not just the code**: you need to be able to keep the big picture in mind when you write your documentation and help developers understand how all your components work together. Therefore, focus on diagrams and clear explanations when documenting your project.
- **Maintain the documentation along with the code**: The best way to keep your documentation up to date is to modify it as you go along with the code.
- If you use a source code manager, you can also include documentation changes for each "*commit*" that modifies your code (see in particular the "Manage your source code" form).
- **Do not forget to address security in your documentation**. In particular, you can document the different configuration choices for your application, and explain which settings are the most secure.

Check the quality of your code and its documentation.

- A quality code involves **adoption of good practices and coding conventions** applied consistently throughout the program. It is also best to refer to existing conventions. Here are a few examples of good practice:
  - **Using explicit variable and function names** makes it easier to understand what is going on at first glance.
  - **Correctly indenting your code** allows you to see the structure of the code more quickly.
  - **Avoiding code redundancy** reduces the correction efforts that have to be made in several places. An oversight is quickly forgotten.
- **Tools can help you control the quality of your code**. Once correctly set up, they will avoid re-reading the code to check the correct implementation of coding conventions. Example of these tools are:
  - **Integrated development environments** ("*IDE*"), possibly using plugins ("*plugins*"), can be configured to respect code indentation rules, line breaks between different portions of code, or the position of braces and other parentheses.
  - **Source code quality measurement software** can report code duplications, compliance with programming rules or potential bugs.

Sheet n°11: Test your applications

Testing your product allows you to check its correct operation, to ensure a good user experience and to find and prevent defects before it goes into production. Testing your product also reduces the risk of personal data breaches.

Automate testing

- The **development tests** (unit, functional, etc.) will verify the adequacy between the specifications and the functioning of the product. The **security tests** (random data tests also called "*fuzzing*", *scan* of vulnerabilities, etc.), will check that the product continues to function acceptably when you move away from its normal use and that it does not present any vulnerability that could allow third parties to compromise its security. Both types of tests are important for the proper functioning of your application.
- Set up a **continuous integration system** to run the tests automatically after each change in your source code.

Integrate testing into your business strategy.

- Add the implementation of the test environment into the company's strategy. The **acceptable metrics** must be defined jointly by all parties prior to development.
- Metrics to consider are for example:
  - The **coverage rate** of tests and their type;
  - the **duplication rate** of your code;
  - the **number of vulnerabilities** (as defined by the tools) and their type, etc.

Watch out for your test data!

- "Real" production data should not be used during the development and testing phase. Using personal data from your production database for testing purposes is tantamount to **diverting it from its original purpose**.
- If personal data is used outside of production, it should be noted that the **security risks** are also **increased**: access to the data by people who do not have a need to know, multiple storage locations, etc.
- So build a **dummy data set** that will look like the data that will be processed by your application. A dummy data set will ensure that disclosure of this data will not have any impact on people.
- If you need to **import existing configurations** from production into your test cases, consider **anonymizing the personal data** that may be present.

Sheet n°12: Inform users

The transparency principle of the GDPR requires that any information or communication relating to the processing of personal data should be concise, transparent, comprehensible and easily accessible in plain and simple language.

Who to inform and when?

- Data subjects must be informed:
  - both **in the case of direct data collection** i.e. when data are collected directly from individuals (examples: form, online purchase, subscription of a contract, opening of a bank account) or when they are collected via devices or technologies for observing the activity of individuals (examples: analysis of Internet navigation, geolocation and Wi-Fi analytics/tracking for audience measurement, etc.) ;
  - and **in the case of indirect collection of personal data**, when data are not collected directly from individuals (examples: data retrieved from trading partners, *data brokers*, publicly available sources, or others).
- This information is needed:
  - **during the data collection** in the case of direct collection;
  - **as soon as possible in the case of indirect collection** (in particular at the time of first contact with the person) and no later than a month from the collection (with exceptions);
  - **in the event of a substantial change or a particular event**. For example: new purpose, new recipients, change in the way rights are exercised, data breach.

What information do I have to give?

- In all cases, you must specify:
  - **The identity and contact details of the organization** that collects the data (who processes the data?) ;
  - **The purposes** (what will the collected data be used for?);
  - **The lawful basis** on which the data processing is based (find all the **information on the lawful basis**);
  - **The compulsory or optional nature of the data collection** (which implies an upstream reflection on the usefulness of collecting the data in view of the objective pursued - the principle of "minimisation" of the data) and the **consequences for the person** in case of failure to provide the data;
  - **Recipients or categories of recipients of the data** (who needs to access or receive them for the defined purposes, including processors?) ;
  - **The data retention period** (or criteria for determining it);
  - **The existence of data subjects' rights and the means to exercise them** (the rights of access, rectification, erasure and restriction are applicable to all processing operations) ;
  - **The contact details of the Data Protection Officer** of the body, if appointed, or of a contact point on personal data protection issues ;
  - **The right to file a complaint with your local Data Protection Agency.**
- In certain specific cases, additional information must be provided, for example in the case of data transfers outside the EU, fully automated decision-making or profiling, or when the lawful basis for the processing is the legitimate interest pursued by the body collecting the data (see the guidelines on transparency for more information).
- In the case of indirect collection, the following must be added:
  - **Categories of data** collected ;

- **The source of the data** (indicating in particular whether it comes from publicly available sources).

In what form should I provide this information?

- The information must be **easy to access**: the user must be able to find it without difficulty.
- **It must be provided in a clear and comprehensible manner**, i.e. with simple vocabulary (short sentences, no legal or technical terms, no ambiguities) and information adapted to the target audience (with particular attention to children and vulnerable persons).
- **It should be written in a concise manner**. In order to avoid the pitfall of a flood of information drowning out the user, it is necessary to **bring the most relevant information at the right time**.
- Data protection related information must be **distinguishable from information that is not specifically related to privacy (such as contractual clauses or general terms and conditions of use).**

What communication should be made when data security is compromised?

- **An organization may mistakenly or negligently suffer, accidentally or maliciously, a personal data breach, i.e. the destruction, loss, alteration or unauthorized disclosure of data**. In this case, the organization must report the violation to the local data protection agency within **72 hours** if it is likely to pose a risk to the rights and freedoms of individuals.
- If these risks are high, the organization must also inform the persons concerned as soon as possible and provide them with advice on how to protect their data (e.g. cancellation of a compromised bank card, modification of a password, modification of privacy settings, etc.).
- Notification of the violation to the CNIL must be made via the CNIL website.

Useful resources

- The Data & Design site developed by the CNIL's Digital Innovation Laboratory develops these concepts and contains interface examples.
- The CNIL site also contains many examples of information notices in French.
- The personal data violations page on the CNIL website (in French).

Sheet n°13: Prepare for the exercise of people rights

The persons whose data you process have rights on his or her data: right of access, to rectification, to object, to erasure, to data portability and to restriction of processing. You must give them the means to effectively exercise their rights and provide in your computer systems the technical tools that will allow their rights to be properly taken into account.

Preparing in advance how they will contact you and how you will deal with their requests will enable you to manage the exercise of these rights effectively.

Minimum measures to be put in place

- All organisations that use personal data have **the obligation to indicate where and how** individuals can exercise their rights in relation to this data. For example, you can mention an e-mail address or a web form when informing individuals, as well as in your privacy policy.
- In order to facilitate the exercise of people's rights, these rights may also be **implemented**, in whole or in part, directly in **the application or software you develop**. This specific implementation is not mandatory, but it allows you to meet users' expectations and reduce the time and complexity of processing this type of request.
- Above all, in case of access or operations directly performed by a person who exercises his or her rights, do not forget to manage his **authentication** in a secure way. Overall, **trace** also all operations that have an impact on his or her personal data.

Here are some examples of rights and their possible implementation

- **Right of access**: people have the right to obtain a copy of all the information you have about them. This allows, among other things, a person to know whether data concerning him or her is being processed and to obtain a readable copy in an understandable format. In particular, it allows the accuracy of the data to be checked.
  Possible implementation: Provide a functionality to display all data relating to a person. If there is a lot of data, you can split the data into several displays. If the data is too large, offer the person to download an archive containing all his or her data.
- **Right to erasure**: Persons have the right to request the deletion of all the data you hold on them.
  Possible implementations:
  1. Provide a functionality to erase all data relating to a person.
  2. Also provide for automatic notification of processors to also erase the data relating to that person.
  3. Provide for data erasure also in backups, or provide an alternative solution that does not restore erased data relating to that person.
- **Right to object**: individuals have the right to object in certain cases to their data being used for a specific purpose.
  Possible implementation: provide a functionality allowing the data subject to object to the processing. When the data subject exercises his or her right to object in this way, the controller must delete data already collected, and must not subsequently collect any more data related to that person.
- **Right to data portability**: Individuals have the right to retrieve their data in a machine-readable format for their own use or for transfer to another organisation.
  Possible implementation: Provide a feature that allows the data subject to download his or her data in a standard machine-readable format (CSV, XML, JSON, etc.).
- **Right to rectification**: Individuals have the right to request the modification of their data when it is incorrect in order to limit the use or dissemination of erroneous information.
  Possible implementation: Allow to directly modify data in the user account.

- **Right to restriction of processing**: individuals have the right to request that the processing of their data be blocked for a certain period of time, e.g. the time to examine a dispute on their part regarding the use of their data or a request to exercise rights.
  Possible implementation: Allow administrators to put data about a person in "quarantine": this data can then no longer be read or modified.

In conclusion

- The Data & Design site developed by the CNIL's Digital Innovation Laboratory develops these concepts and contains examples of interfaces for exercising rights.
- Finally, be **inventive**! (In case of doubt, ask the CNIL for advice).

Sheet n°14: Define a data retention period

Personal data cannot be kept for an indefinite period of time: this must be defined according to the purposes of the processing. Once this purpose has been achieved, the data should be archived, deleted or made anonymous (e.g. in order to produce statistics).

Data retention cycles

- The personal data retention cycle can be divided into **three distinct successive phases**:
  - The active database;
  - Intermediate archiving;
  - Final archiving or deletion.
- The mechanisms for deleting personal data from the active bases ensure that the data are kept and accessible by the operational services only **for the time necessary to achieve the purpose of the processing operation**.
- Ensure that **data is not kept in active databases** by simply noting them **as being archived**. The archived data (intermediate archive) must be accessible only to a specific service responsible for accessing and removing them from the archive if necessary.
- Please also ensure that you have **specified access modes** for the archived data, as the use of an archive must be on an ad hoc and exceptional basis.
- If possible, use the same implementation when implementing the **data purging or anonymisation** as the one managing the **right to erasure** (see [sheet on the exercise of rights](Sheet_n°13:*Prepare*for_the_exercise_of_people's_rights)), in order to guarantee a homogeneous operation of your system.

Some examples of shelf life

- The **data relating to payroll management or employee time control** can be kept for 5 years.
- The **data in a medical file** must be kept for 20 years.
- The **data of a prospect not responding to any solicitation** can be kept for 3 years.
- The **log data** can be kept for 6 months.

Sheet n°15: Take into account the legal basis in the technical implementation

Processing of personal data must be based on one of the "legal basis" mentioned in Article 6 of the GDPR. The legal basis of a processing operation is in a way the justification of the existence of the processing operation. The choice of a legal basis has a direct impact on the conditions for implementing the processing operation and the rights of individuals. Thus, anticipating the legal basis of the processing operations prior to any development will help you integrating the necessary functions to ensure that these processing operations comply with the law and respect the individuals rights.

Definition of the legal bases in the RGPD

- In the context of a development for a private organization (companies, associations, etc.), the legal basis often used are:
  - **The contract**: the processing is necessary for the performance or preparation of a contract between the data subject and the body carrying out the processing operation;
  - **The legitimate interest**: the organization has a "legitimate" interest in carrying out the processing and it is not likely to adversely affect the rights and freedoms of the data subjects;
  - **Consent**: the data subject has given his or her explicit consent to the processing.
- If you are a public authority or a body pursuing tasks in the public interest, other legal bases may also be used:
  - **The legal obligation**: the processing is imposed by regulatory texts.
  - **The public-interest mission**: the processing is necessary for the performance of a task carried out in the public interest.
- Finally, in very specific cases, **protect of vital interests** may be used as a legal basis, for example when processing is necessary to monitor the spread of epidemics or in cases of humanitarian emergency.

Choose the appropriate legal basis

- First of all, check on the CNIL website that **a text does not impose any particular constraints** (for example: cookies and other trackers).
- **Only one legal basis must be chosen** for a given purpose. Legal basis cannot be cumulated for the same purpose. The same data processing operation may pursue several purposes and a legal basis must then be defined for each of them.
- As mentioned above, if you are a **public authority**, the legal obligation and the public interest mission will be the most relevant in most cases.
- If your processing operation is part of a contractual relationship and its purpose is objectively and strictly necessary for the provision of the user's service (e.g. name, first name and address to create an account on an e-commerce site) then **contract should be appropriate**.
- If your processing is not part of a contractual relationship with the user, then **the legal basis of consent or legitimate interest** may be invoked. If your processing is potentially intrusive (profiling, collection of geolocation data, etc.) then **consent is likely to be the appropriate legal basis**.
- If your processing contains **sensitive data** (health data, data concerning life or sexual orientation, etc.), then you will need to identify, in addition to the legal basis, an exception provided for by Article 9 of the GDPR.

Rights exercises and modalities of information to be provided according to legal basis

- The following table summarizes the exercises of rights to be provided in accordance with legal basis:

	Right of access	Right to rectification	Right to erasure	Right to restriction of processing	Right to data portability	Right to object
Consent	✓	✓	✓	✓	✓	**Withdraw of consent**
Contract	✓	✓	✓	✓	✓	✗
Legitimate interest	✓	✓	✓	✓	✗	✓
Legal obligation	✓	✓	✗	✓	✗	✗
Public interest	✓	✓	✗	✓	✗	✓
Protect of vital interests	✓	✓	✓	✓	✗	✗

- The legal basis used must **always appear in the information transmitted to the person**.
- **Where your processing is based on legitimate interest**, you must also indicate the legitimate interests pursued (fight against fraud, system security, etc.).
- It is recommended to **document your choice of legal basis**. As an example, these choices can be indicated in a processing map or associated with your technical documentation.

The specific case of cookies and other trackers

- The European ePrivacy Directive requires the user's consent before any action is taken to store information - via cookies, identifiers or other tracers (software fingerprints, pixels) or to access information stored in the user's terminal equipment.
- However, an exception is made when cookies are for the sole purpose of carrying out electronic communication, or are strictly necessary to provide a service requested by the user.
- Furthermore, the use of a single tracer for multiple purposes does not exempt from obtaining consent for the purposes that require it. For example, if an authentication cookie is also used for advertising targeting purposes, consent must be obtained for the latter purpose, in the same way as for a non-logged site.

Sheet n°16: Use analytics on your websites and applications

Audience measurement tools are used to obtain information about the navigation of visitors on a website or mobile application. In particular, they make it possible to understand how users arrive at a site and to reconstruct their journey. Using cookies, they are subject to the rule of consent, except in one particular case. Please note that this section is relative to the ePrivacy directive and may be subject to national variation. Contact you local data protection agency to know its position.

Obtaining consent

- Generally speaking, **before depositing or reading a cookie or tracer,** editors of sites or applications must:
  - inform Internet users of the purpose of cookies;
  - obtain their consent;
  - provide them with a means of refusing them.
- Unless they fall exactly within the perimeter defined below, **this obligation applies to tracers used for audience measurement**.

To benefit from the exemption from consent

- **Subject to a number of conditions**, cookies used for audience measurement are exempt from consent.
- **These conditions, as specified in the guidelines on cookies and other trackers, are**:
  - To inform users of their use;
  - To give them the ability to object to their use;
  - To limit to the following purposes only:
    - audience measurement;
    - A/B testing;
  - Not to cross-check the data processed with other processing (customer files, statistics on visits to other sites, etc.);
  - To limit the scope of the tracer to a single site or application editor;
  - To truncate the last byte of the IP address;
  - To limit the lifetime of the trackers to 13 months.
- Provided that the conditions are met, **we therefore switch from an opt-in to an opt-out regime**.
- It is also possible for the same third party (subcontractor) to provide a comparative audience measurement service to multiple publishers, provided that **the data is collected, processed and stored independently for each publisher and that the trackers are independent of each other**.

In practice

- **Most large audience measurement offerings do not fall within the scope of the exemption, regardless of their configuration**.

- In order to benefit from this exemption, please contact your solution provider or use open source software such as Matomo that you can configure yourself.