

DGA/DIT/DTK(2023)07

English

Developer's Toolkit

DIT – DEVELOPER'S USER GUIDE

This document is the property of the Council of Europe. It is part of a consultation launched by the Directorate of Information Technologies (DIT) of the Council of Europe.

The service provider will treat this document as strictly confidential. This document may not be reproduced or distributed without the prior agreement of the author.

The content of the document may be passed on in good faith by the service provider to any of its partners. No part of this document may constitute a commitment on the part of the Council of Europe or its staff.

CONTENTS

1	Introduction.....	5
1.1	Objectives of the document	5
1.2	Glossary	5
2.	Technologies	6
2.1	For the front-end.....	6
2.2	For the back-end	6
2.3	Web services development.....	6
2.3.1	Web services authentication	6
2.3.2	Securing Web Services	7
2.3.3	For databases.....	7
2.4	For printing reports	7
2.5	Application logging.....	7
2.6	Storing files in the application	8
2.7	Third-party technologies available at the COE	8
2.7.1	Monitoring and supervision: Prometheus	8
2.7.2	Cartography.....	8
2.8	Tools.....	9
2.8.1	Development tools.....	9
2.8.2	DevOps (Microsoft Azure).....	9
2.8.2.1	Project method	9
2.8.3	Continuous integration.....	9
2.9	Standards	9
2.9.1	Standards applicable to the front-end	9
2.9.2	Back-end standards.....	10
2.9.3	Communication standard (format, envelope, etc.).....	10
2.9.4	Security standards.....	10
2.9.4.1	Compliance with safety advice.....	10
2.9.4.2	API on Web Services.....	10
2.9.4.3	Authentication.....	11
2.9.4.4	Identity and Accreditation Governance (IAG).....	11
3.	Council of Europe environnements	12
3.1	Environnements.....	12
3.2	Validation environnement	12
3.3	Sending emails and notifications.....	13
3.4	Production environment.....	13
3.5	Environment on demand.....	13

4.	Deliverables	14
4.1	Documentary deliverables	14
4.2	Source code deliverables.....	14
4.3	Structuring projects Development	14
4.3.1	Composition of a .NET Core application	14
4.3.1.1	SRC Section	14
4.3.1.2	TEST section: Unit tests	15
4.3.2	Composition of an Angular application.....	15
4.4	Integration process	15
4.4.1	Delivery process	15
4.4.2	Integration process	16
4.4.3	Unit Testing	16
4.4.4	Integration test methodology.....	16
4.4.5	Going into production.....	16
5.	General Ergonomic Requirements	16
5.1	Presentation	16
5.2	Multilingualism	17
5.3	Performance requirements	17
5.4	GUI requirements (responsive design).....	17
5.5	Layout and grid requirements	18
5.6	Requirements linked to accessibility	19
5.6.1	General rules	20
5.6.2	Textual content	20
5.6.3	Hypertext links	20
5.6.4	Images & media.....	20
5.6.5	Tables.....	20
5.6.6	Colours & contrasts	20
5.6.7	Navigation	21
5.6.8	Data entry and forms	21
5.6.9	Error message	21
5.6.10	Checking the code	21
5.7	Internet browsers	21
5.8	Application charter.....	22
5.8.1	Intranet application	22
5.8.2	Internet application (external audience)	23
6.	Data protection	24
6.1	Confidentiality requirements	24

6.2 RGDP compliance 24

1 Introduction

1.1 OBJECTIVES OF THE DOCUMENT





This document presents the Developer's Toolkit, which defines the technical requirements for application development at the Council of Europe's Directorate of Information Technology (DiT).

The main challenge of the Toolkit is to provide a **uniform technical framework** enabling:

- develop specific applications.
- limit regressions during development and/or maintenance cycles.
- optimise the processes to be followed in developments.
- Follow secure development principles

The field of application of the Toolkit concerns the specific applications developed by the CoE to meet its needs. It is aimed both at the DiT's in-house developers and at service providers in charge of development and/or maintenance projects on behalf of the DiT.

Throughout this document, the pictograms below are used to highlight important points or concepts.

	Important information
	Specific action
	Action to avoid
	Mandatory action

1.2 GLOSSARY

Term	Definition
CoE	Council of Europe
DiT	Directorate of Information Technologies
SLA	Service Level Agreement
DOM	Document Object Model [reference to HTML DOM]
DDL	Data Definition Language (in SQL - creating tables, for example)
DML	Data modification language (in SQL - data modification script)
IAG	Identity and accreditation governance

GUI	Graphical User Interface - the application's human-machine interface
ORM	Object Relational Mapping - programming device enabling an application object to be saved in a database
CISO	Information System Security Manager

2. Technologies

2.1 FOR THE FRONT-END

The following frameworks are authorised by the Council of Europe for the front end of the application:

- Angular framework
- Bootstrap

The versions to be used are the latest LTS versions of the frameworks available to date.

2.2 FOR THE BACK-END


The application back-end is built in .NET Core (Latest LTS).

This technology, supported by the Council of Europe, enables deployment to be automated via a continuous integration pipeline and a Docker container.

2.3 WEB SERVICES DEVELOPMENT

Shared API web services must:

- Be authenticated
- Being secure
- Be documented with a Swagger File descriptor.

	The swagger file must not be accessible in the production environment
---	---

2.3.1 Web services authentication

Web services provided by the application must be authenticated.

The Council of Europe currently supports two authentication methods:

- OAuth 2
- Open ID Connect

2.3.2 Securing Web Services

Communications between the Angular front-end and the application's web services will be secured by tokens (JWT) to guarantee the authenticity of requests.

2.3.3 For databases

SQL Server 2019 must be used for development purposes.

DDL and DML SQL scripts should be isolated in the source code delivery.

There are two possible scenarios:

- Either the development tool (.NET) has an ORM tool for managing database version upgrades: this is the preferable case, which does not require human intervention.
- Otherwise, each commit must specify the scripts to be run on the environment to update the base structure, tables, stored procedures or data.

In all cases, scripts (if any) will have the .sql extension so that they can be identified in deliveries.

Database creation, collation and other settings will have to be scripted and run manually by the Council of Europe's operations unit on the database engines of the environments delivered.

2.4 FOR PRINTING REPORTS

Depending on the complexity of the reports requested, it is possible to use different technologies available within the Council of Europe:


Technology	Use
SQL Server 2019 Reporting Service	Printing simple reports / Listing
Customised development	For individual exports

For more complex statistics, if required by the project, contact the project coordinator.

2.5 APPLICATION LOGGING

A specific library should be used to log an application.

It must be possible to set the log level from an application configuration variable.

	<p>The logs should have several levels:</p> <ul style="list-style-type: none"> - TRACE - DEBUG - WARNING - ERROR - CRITICAL <p>The log management method must be explained: file running for X days and adapted to each environment.</p>
---	---

The logs are not currently centralised.

However, the log output format must be documented.

2.6 STORING FILES IN THE APPLICATION

There are two types of files:

- Text documents must be stored
 - o in the DMS (Document Management System) together with metadata for collaboration purpose
 - o in the RMS (Record Management System) together with metadata for record and archives purpose.
- Application or system files must be stored on a suitable medium, particularly in cloud mode.



Under no circumstances should files be stored in a database.
Cloud storage is preferable.

2.7 THIRD-PARTY TECHNOLOGIES AVAILABLE AT THE COE

2.7.1 Monitoring and supervision: Prometheus

The Council of Europe currently uses the Prometheus metrology/alerting tool.

The development must include a health check type probe providing information on the application's operation (typically UP or DOWN).

These sensors will be integrated into the Prometheus dashboards to provide information on how the application is working.

Developments must incorporate the following probes as standard:

Probe	Type	Description
Healthcheck	Jumper	Informs you that the application is working properly

2.7.2 Cartography

When an application requires the use of a mapping tool, the technical coordinator should be contacted for the political aspects.



For example: we need a tool that displays the countries of the world as defined politically by the Council of Europe.

The choice is guided by the final use of the cartography (restricted public, internet, etc.).

2.8 TOOLS

2.8.1 Development tools

At the time of writing, the preferred development tool is **Visual Studio 2022**. However, **Visual Studio Code** for .Net Core is an alternative.

2.8.2 DevOps (Microsoft Azure)

2.8.2.1 Project method

The tools provided support V-cycle and Agile development methods.


The DevOps platform also supports :

- CI/CD chain (in particular deployment via Azure Pipelines)
- The use of Docker containers.
- Agile project task workflow (broken down into Epic / User Story / Sprint / Release / Bugs / Tasks)
- V-cycle project task workflow (broken down into Bugs / Change Request / Patches / Requirements / Tasks / User stories / Risks / Support Request)

2.8.3 Continuous integration

It must be included in the project:

- One or more Docker containers to host the application and its dependencies.
- An Azure Pipeline to trigger the integration deployment process when an action is taken on the application's Git repository.

	<p>At the start of a project, the integration arrangements will be agreed between the development teams and the tech leads at the Council of Europe.</p> <p>The Council of Europe's technical teams will be able to provide scripting templates for Docker containers and for the Azure DevOps pipeline.</p>
---	--

2.9 STANDARDS

The programming environment to be used for development projects is a back-end in the form of a RESTful .NET API and an Angular front-end.

2.9.1 Standards applicable to the front-end

Development will have to follow best practice in the technologies used.

Technology	Standard
Angular	Development using the Angular model CamelCase Ecma Script 6

2.9.2 Back-end standards

In the various technologies, the structure of the source code must respect the "layer" model:

- With a DATA layer (Data Access Layer)
- A Services layer (Business Logic Layer)
- A surface layer :
 - o Web service / Web API access
 - o Front-end access - GUI

Technology	Standard
.NET	Compliance with Microsoft coding conventions Structuring code in layers
SQL	SQL best practice

	Supply the : <ul style="list-style-type: none"> ■ Technical Architecture File ■ Operating File
---	--

2.9.3 Communication standard (format, envelope, etc.)

Exchanges between components/APIs are made in JSON on a RESTful API-type application architecture.

2.9.4 Security standards

2.9.4.1 Compliance with safety advice

The application developed must be robust, reliable and not compromise the data. To this end, and to limit the main risks, development security standards must be respected (OWASP top 10 <https://cheatsheetseries.owasp.org/IndexTopTen.html>).

2.9.4.2 API on Web Services

Access to APIs must be encrypted and authenticated in order to protect access to data.

Use of OAuth / JWT for calls between front-end and back-end.

Public web services will require web api authentication to authorise the client application to consume their services.

Api development should respect the secure api standards ([OWASP API Security Project | OWASP Foundation](#))

2.9.4.3 Authentication

Users will access the application via their web browser. User sessions must be authenticated and traced.

For sensitive applications, integration with multi-factor authentication is necessary, following the Council of Europe standards.

2.9.4.3.1 For the production environment

Authentication will be carried out with the COE identification tool using the SAML V2 or OIDC protocol.

Authentication requirements :

- The application must allow people from outside the Council of Europe to authenticate themselves using a form and where needed a multi factor authentication
- The application should enable self-logon for Council of Europe Agents
- SSO authentication must be transitive on the web services consumed by the application:
 - o *If Michel Durand is authenticated on the main application, it is Michel Durand who uses the APIs of application A/Application B.*

2.9.4.3.2 For the integration environment

In the integration environment, the application can be either in auto-logon mode or in COE identification tool bypass authentication mode. The aim is to be able to take on the role of any user in the application.

2.9.4.3.3 Anonymous mode

Public non sensitive applications will be able to use anonymous authentication.

2.9.4.4 Identity and Accreditation Governance (IAG)

IAG at the Council of Europe provides identification and accreditation of users to enable them to access a given application.

2.9.4.4.1 Identification

IAG enables the creation of a unique identity for an individual, based on a combination of :

- Surname / First name (from passport / ID card)
- Date of birth

These elements are essential to the creation of an identity.

In return, the IAG provides an identifier (UIP) that links an application to its account/identity.

The IAG lists both internal users at the Council of Europe and external users.



At the start of a project, the integration methods between the IAG and the application to be developed need to be defined.

2.9.4.4.2 Security logs

Every new sensitive application should provide a minimum level of logging concerning security events.

Security logs are of multiple kind; but as a minimum every new application should be able to trace:

- User actions in the application
- Data modifications and if applicable data transfer
- Connection to the application (user, type and location)
- If a database is used, logs for the specific access and actions to the database should be provided.

Logs should be made available in a standard format for ingestion in the centralised SIEM system (json, syslog, xml...)

Every collected log must be timestamped.

2.9.4.4.3 Databases

For applications containing sensitive information, the database should be able to provide encryption at rest.

In some case anonymization of data can be requested.

2.9.4.4.4 Connection

Every connection to the application should be made securely according to the security standards of the market and provide encryption in transit.

Access restriction via geolocalisation or IP address can be made available for highly sensitive applications.

3. Council of Europe environnements

3.1 ENVIRONNEMENTS

The Council of Europe provides 2 environments:

- The validation environment (characterised by URLs containing VAL), the objective of which is the functional and technical acceptance of the application.
- The production environnement.

3.2 VALIDATION ENVIRONNEMENT

The validation environment is only accessible from the Council of Europe intranet. It is used to carry out functional and technical tests of the application in a "real" environment.

The continuous integration chain deploys the Docker container containing the application and its dependencies. Once deployed, the application becomes accessible.

The application parameters relate to the environment:

- COE identification tool bypass
- Routing emails to a group of testers / SMTP validation server
- Access to third-party application components in the acceptance environment

These parameters are supplied by the project coordinator in order to provide a correctly configured environment.

3.3 SENDING EMAILS AND NOTIFICATIONS

The application can use an SMTP relay to send emails or notifications.

In the integration environment, must be sent using COE email relay system – Mailhog.

3.4 PRODUCTION ENVIRONMENT

The production environment includes a proxy server as input.

This proxy supports SSL/TLS security for public access to the application.

As a result, the application will have to be developed with this specificity in mind. Communication between the proxy and the application is generally via HTTP.

The developer must provide the elements required to configure the communication between the proxy and the application, in particular the HTTP headers to be provided and the various protocol options.

3.5 ENVIRONMENT ON DEMAND

By default, the validation environments are not open to anyone outside the Council of Europe. However, it is possible to obtain a test environment (Virtualised PC under Windows) in order to run the application in the real context of the Council of Europe and to have access to the various components linked to the application.

Virtualized PC with development tools installed can also be setup to allow test and debug on COE premise.

As a reminder, in the development environment, the links to the APIs will have been simulated and plugged.

4. Deliverables

4.1 DOCUMENTARY DELIVERABLES

Depending on the criticality level of the application, the following documentation is required:

- Detailed Functional Specifications File
- Alternatively, with the agreement of the project coordinator:
 - Documenting user stories to better adapt to Agile mode.
- Technical architecture file
 - Describing the technical architecture of the application in terms of infrastructure, network and protocols
- Application installation file
- Test booklet
- User manual
- Operating procedures
 - Procedures for monitoring links and interfaces
 - Monitoring to be applied
 - Application monitoring
 - Batch and automated processing

4.2 SOURCE CODE DELIVERABLES

A source code unit repository consists of the following elements:

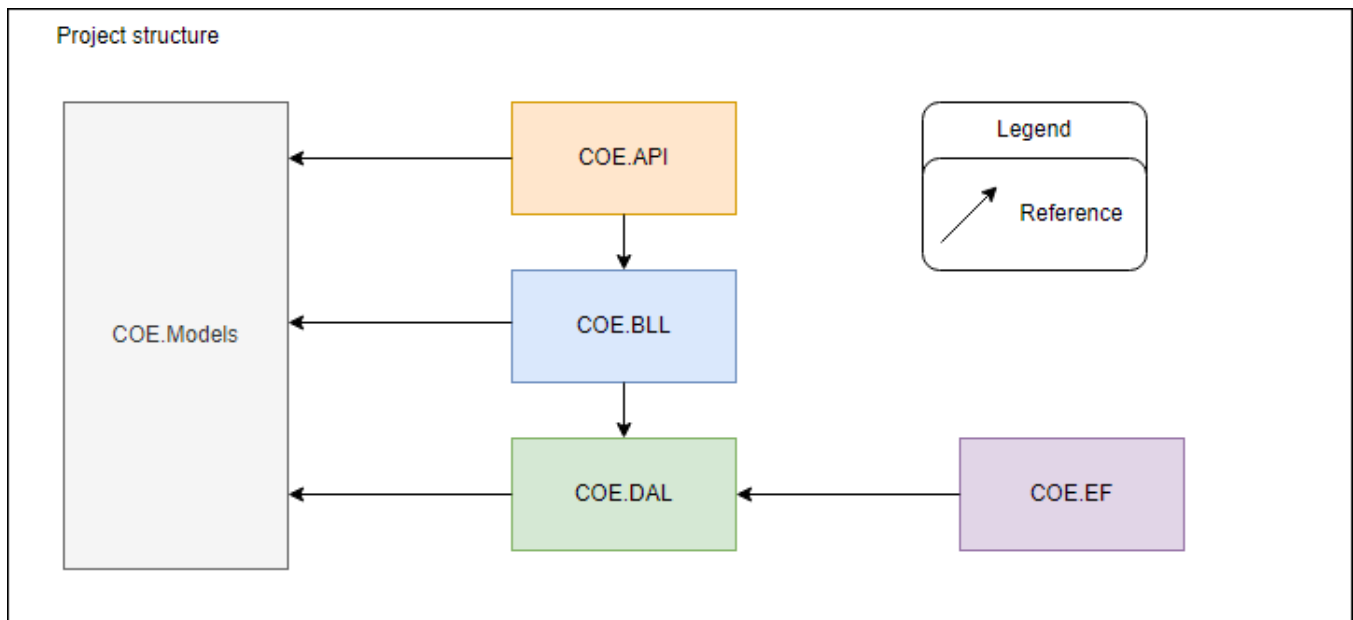
- Integration: reserved for the integration team
 - Contains scripts for Docker and Azure Pipelines
- Source: contains all the application's source code
 - Folder dedicated to SQL scripts
 - File dedicated to the .NET Core/ Angular application
 - Structured in two sections:
 - The src section: containing source code projects
 - The test section: containing unit test projects
- Documentation: contains the service provider's documentation

4.3 STRUCTURING PROJECTS DEVELOPMENT

4.3.1 Composition of a .NET Core application

4.3.1.1 SRC Section

The .NET application will necessarily be multi-layered:



4.3.1.2 TEST section: Unit tests

The unit tests must cover the elements of the various projects.

The following test frameworks can be used:

- Nunit
- Xunit

Unit tests are implemented in accordance with the conventions associated with each Framework.



The execution of unit tests must be integrated into the continuous integration pipeline.

4.3.2 Composition of an Angular application

An Angular application is made up of several elements:

- View templates (in HTML)
- TypeScript components representing blocks of display logic
- Modules (groups of Templates and components)

It is up to the developer to ensure that the Human Machine Interface layer only integrates display logic and does not embed business logic (dedicated to the Service layer of the .NET Core application).



Development in Angular must respect the state of the art of the Framework.

4.4 INTEGRATION PROCESS

4.4.1 Delivery process

The source code and associated documentation are to be delivered in a Git repository on Azure DevOps provided by the CoE.

4.4.2 Integration process

The integration process is as follows:

- Delivery to Git repository + Version tag
- Automated process triggering at the tag:
 - o Compiling the application
 - o Deploying the solution in an integration environment
 - o Launching unit tests

At the end of the process, the application is operational and ready to be tested by users.

4.4.3 Unit Testing

The application must include automated unit tests written by the service provider. These tests are run each time the application is pushed or committed to the Git branch.

4.4.4 Integration test methodology

Integration tests cover the following areas:

- Functional area
- Technical domain and interface exchanges

Integration tests are run manually by testers in the integration environment.

The application's use cases are executed as well as more extensive use scenarios.

Test results are recorded for arbitration and resolution.

4.4.5 Going into production

Production environments are made available for deployment of the application.

The service provider supplies :

- The installation file, which must be sufficiently detailed to enable the Council of Europe to deploy the application independently.
- The Technical Architecture file including
 - o The machines needed for the application.
 - o Network flow matrix.

The application is put into production once the acceptance test has been successfully completed.

5. General Ergonomic Requirements

5.1 PRESENTATION

The general principles of ergonomics are applicable to IT development projects. These requirements cover the UX, ergonomics and accessibility aspects, to ensure that the application is easy to use.

5.2 MULTILINGUALISM

The application must be fully translated into the two official languages of the Council of Europe, French and English.

5.3 PERFORMANCE REQUIREMENTS

Over 40% of users abandon a site that takes more than 3 seconds to load. Slow loading is frustrating and leads to a drop in attention.

Loading time depends on the following factors:

- DOM complexity
- Volume of information to download
- Page generation time

As the application can be used all over the world and Internet access may be limited in certain geographical areas, the application will need to be bandwidth-efficient.

Page size limit for extranet application	< 2 Mb
Time limit for page generation at server output	3 seconds

To improve the user experience, we recommend using delayed loading of information.

5.4 GUI REQUIREMENTS (RESPONSIVE DESIGN)

Responsive design ensures that users can access the site regardless of the device they are using (computer, smartphone, tablet), offering a layout that adapts to the size of the screen. Responsive design development should be assessed at the start of the project; although desirable, it is not necessarily a prerequisite.

The following table shows the screen resolutions of our PC fleet in February 2020:

RESOLUTION	NUMBER OF PCS
1024X768	26
1152X864	2
1280X1024	132
1280X720	10
1280X768	5
1280X800	2
1280X960	1
1360X768	4
1366X768	65
1400X1050	4
1440X900	33

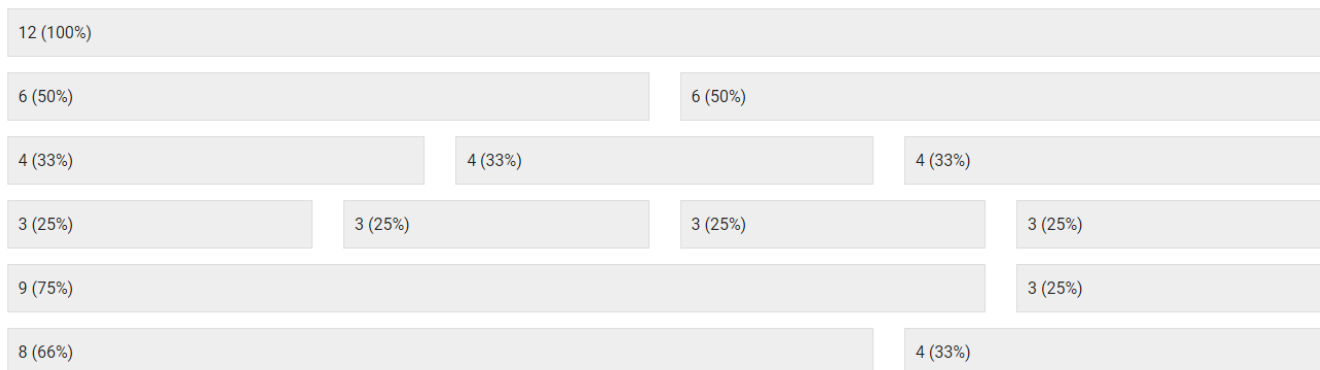
1600X1200	49
1600X900	92
1680X1050	280
1920X1080	405
1920X1200	1047
2560X1440	7
2560X1600	5
TOTAL	2169

The majority of our users have a 1920 pixel wide screen. However, a significant number of users work with smaller screens (1680 pixels, 1280 pixels). These resolutions will be tested as part of future development.

It should also be noted that the Windows zoom of 150% is a default setting on some of our laptops with a resolution of 1920 pixels, so on these devices the site will be displayed as on a screen of 1280 pixels (1920 / 1.5). To ensure good accessibility, we recommend that the application remains visible if the user zooms up to 200%.

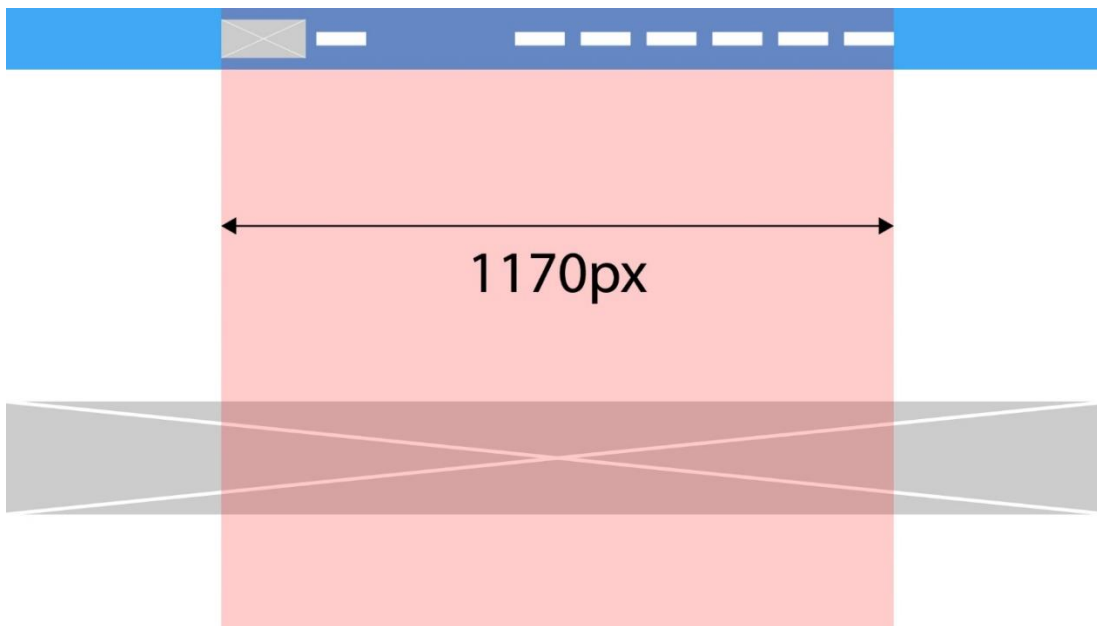
5.5 LAYOUT AND GRID REQUIREMENTS

The application will use a fluid grid system of 12 columns, with breakpoints.

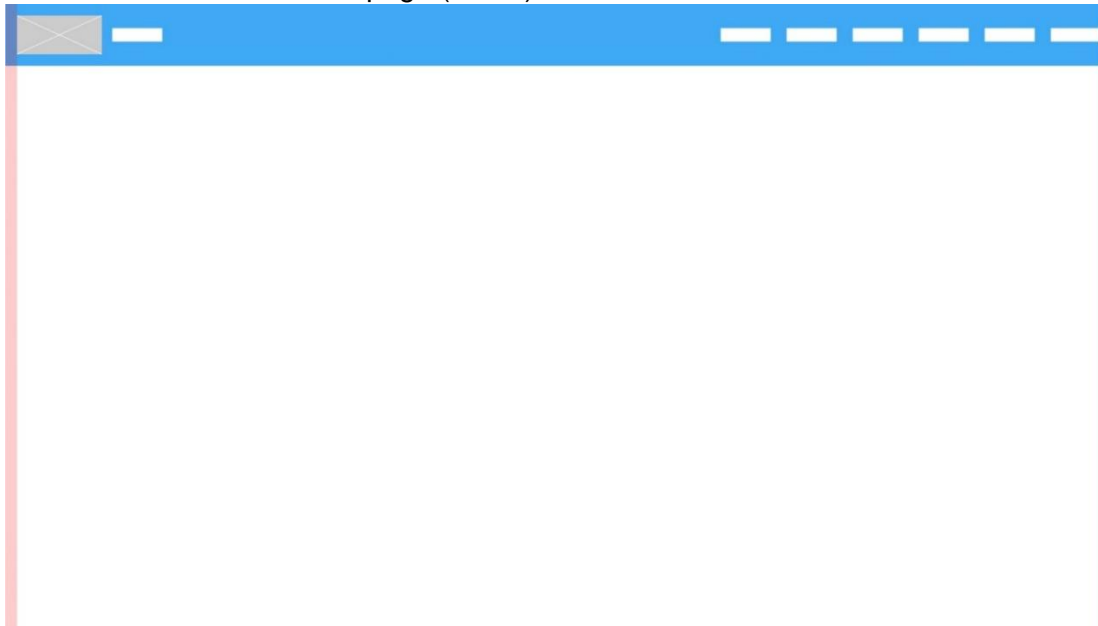


There are two possible scenarios for content placement:

1. Use a 1170 pixel container. All the content is placed in a 1170 pixel block, with the exception of the backgrounds, which can use the full width of the page.



2. Use the full width of the page (100%).



These two scenarios should be considered depending on the content and use of the application. The encompassing block makes the text easier to read and is therefore appropriate for editorial content. For an application that offers fairly dense calendars or tables, for example, full width is more appropriate.

5.6 REQUIREMENTS LINKED TO ACCESSIBILITY

Difficulties in interacting with a system can arise from a number of factors: physical disabilities, injuries or temporary situations, age, language barriers, low-speed connection, obsolete devices, etc. Inclusive practices prevent these difficulties and often prove beneficial for all users. The Council of Europe strives to make its sites accessible to a wide range of people.

The [Web Content Accessibility Guidelines \(WCAG\)](#) list a number of guidelines for providing good accessibility for disabled people with hearing, visual, motor, cognitive and neurological impairments, as well as for people with age-related disabilities (older people whose abilities have diminished). The

WCAG defines three levels of accessibility: A, AA, AAA. The level of accessibility will be specified at the start of the project.

5.6.1 General rules



Pages must appear and function in a predictable way. Components with the same functionality are identified in the same way and have the same appearance. The use of web standards and norms reduces the need for users to learn the interface by drawing on their previous experience.

5.6.2 Textual content

All the textual information on a site must be clear and simple enough to be understood by everyone. Pages must have a title that concisely describes their subject or purpose. Long paragraphs should be avoided: all information should be concise and clear.

Rare words, trade-specific expressions and abbreviations should be defined using a specific mechanism (e.g. tooltips or glossary).

We recommend the use of action verbs for buttons, so that the user understands and identifies the action to be performed. Headings such as "yes" or "no" should be avoided as they do not allow the user to understand the action to be performed.

	
<p>✗ To be avoided. The user cannot identify the action.</p>	<p>✓ The actions that can be carried out are easy to understand.</p>

5.6.3 Hypertext links

The link must have a heading explaining its function.

For reasons of accessibility and user experience, we recommend that links in the application open in the same context as the current page and not in a new window or tab. It is preferable to inform users of the opening of new windows or tabs by indicating this textually in the link to indicate the change of context.

5.6.4 Images & media

All non-text content must have a text alternative. It is essential to fill in the alt attribute of an image, an alternative text to the image that enables blind and partially-sighted people to obtain a description of it.

Subtitles must be provided for videos. Sound control and a pause mechanism must be available for audio and video content lasting more than 3 seconds.

5.6.5 Tables

Blind people cannot make visual associations between headings and cell data, so it is essential to use semantic HTML tags, and in particular to define the table headings so that the table can be read properly by screen readers.

It may also be desirable to offer the following functions:

- Visual elements indicate which row or column the user is hovering over.
- Alternating colours are used between rows and columns to make the table easier to read.
- The data displayed in the tables can be sorted by clicking in the header of each column.

5.6.6 Colours & contrasts

The expected contrast ratio is at least 4.3:1 to enable people with colour perception deficits to perform adequately when reading.

You shouldn't rely solely on colours to convey information, even if these colours have a strong meaning (green and red, for example).

5.6.7 Navigation

Navigation is crucial: it must guide users through the system and enable them to achieve their objectives. Efficient navigation has a major impact on the user experience and the overall perception of the system.

Navigation must be consistent: within a set of pages, the navigation mechanisms must be repeated. The path to be followed by the user to reach their goal must be clear: the actions and the order in which they are carried out must be easily identifiable.

Provide users with information to help them find their way around a set of pages: mark active pages in the menu, use a breadcrumb trail if necessary.

5.6.8 Data entry and forms

Provide tooltips for input fields to help users fill in forms. By hovering over the tooltip pictogram, the user will see the text in the selected language. The texts in the tooltips can be managed. Tooltips are available in French and English as standard.

5.6.9 Error message

Application error messages must be self-explanatory and in English or French. The texts of error messages must be configurable from an administration access.

The application must be able to display different levels of error messages:

Information	Declarative message
Fatal	Critical error message generally associated with a technical error. Blocking message.
Warning	Blocking information message
Debug	Displays a debug trace

All error messages must be filtered so that blocking messages are not displayed directly from the server.

5.6.10 Checking the code

The [W3C Markup Validation Service](#) is a tool for checking the validity of code and identifying errors or omissions. The code delivered must not contain any errors.

5.7 INTERNET BROWSERS

The application must be usable on the following browsers:

- Chrome
- Firefox
- EDGE
- Safari

For information, TLS 1.0 and 1.1 security is prohibited, so the application must support 1.2.

5.8 APPLICATION CHARTER

5.8.1 Intranet application

If the application is intended for use by Council of Europe staff, essential elements that identify the Council of Europe must be used.

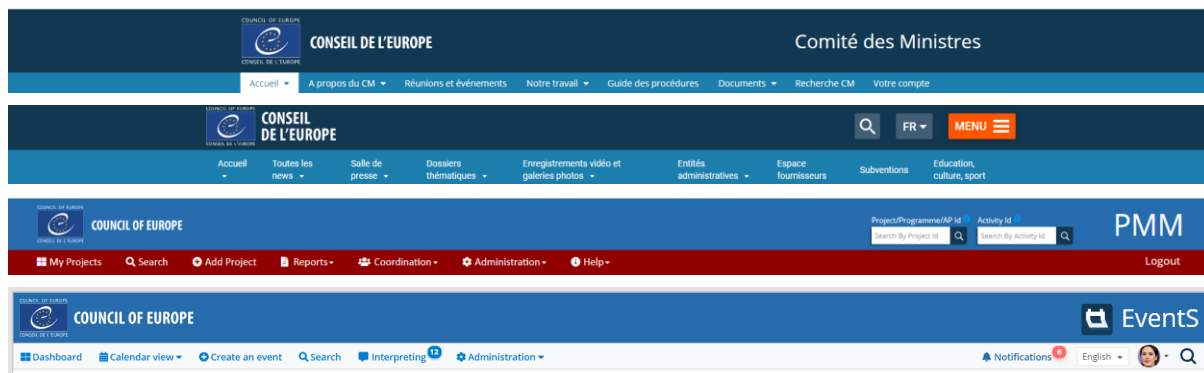
Header banner

The header banner is a fundamental element of our sites and enables users to identify the Council of Europe. The Council of Europe logo must appear here. It must be placed in the top left-hand corner of the banner.

Logo download links :

- [French version](#)
- [English version](#)
- [Without text](#)

Examples of standard banners:





There must also be a specific mechanism for selecting the language.

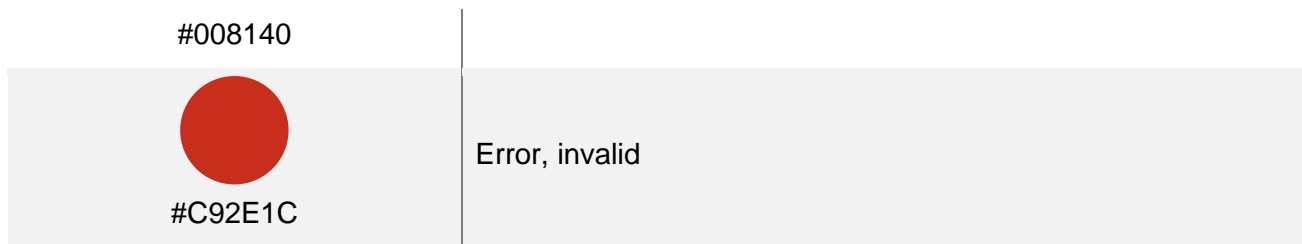
Typography

The font used on the Council of Europe Internet portal and third-party sites is Open Sans. We recommend a font size of 15 pixels for the body of the text. The font size should not be less than 9 pixels. This font uses 5 fat variants, each with an italic version.

Colours

Depending on the nature of the application and the graphic identity of the beneficiary, the colours can be chosen. Otherwise, here are our recommendations:

COLOUR AND HEXADECIMAL CODE	USAGE
 #266BAC	Main colour, header background, links
	Success, confirmed



For the page background, we recommend using white or grey.

Iconography

A specific iconography is used for our sites: <https://commtools.coe.int/vdd/#icons> It can be completed with FontAwesome, an icon library.

5.8.2 Internet application (external audience)

If the application is created to communicate with an external audience, it must comply with the Council of Europe's design standards and adopt its visual identity.

To create an application that is open to the Internet, the development team can contact the Communications Department to obtain the rules for developing an application that can be integrated (or not) into the public portal. Online resources are available to facilitate these developments, see the [frontend starter kit](#).

The essential elements that must be scrupulously respected in all cases are :

- The official logo for the web ;
- The URL: nomdevotresite.coe.int ;
- Disclaimers (see website footers) ;
- Search engine optimisation (SEO - applying at least the basic 'natural' referencing techniques) ;
- WAI: Accessibility for people with disabilities: the Council of Europe's interactive tools must be part of a commitment to excellence in this area;
- Responsive design (and layouts for mobile devices), so that your site can be viewed on mobile devices;
- Browser compatibility.

Sites can be customised to give them an individual touch, while maintaining a common uniform look and feel. The following elements can be customised:

- The colour palette ;
- The title of the site ;
- Visual identity ;
- The footer.

In some cases, websites are created to meet a very precise or specific objective. DC does not wish to restrict creativity in external communication. Consequently, although the essential elements must be scrupulously respected, there is greater scope for freedom when :

- A website is dedicated to a specific theme or event, for which the image of the Organisation is of secondary importance;
- You are targeting a specific audience whose needs you have clearly identified;
- A website is a collaboration between equals and one or more partner organisations.

6. Data protection

6.1 CONFIDENTIALITY REQUIREMENTS

If an application handles sensitive personal data (surname/first name/personal home address/email address, etc.), these must under no circumstances be altered by a third party, nor consulted by an unauthorised third party.

The developer must take into account the data access channel:

- Internet access
- Private network access only

and set up the appropriate encryption systems:

- Data transport security, with systematic encryption of data between access points and data consumers.
- Data access security to prevent any modification by a user other than the administrator

Private data in the database must be encrypted to prevent unauthorised access.

6.2 RGDP COMPLIANCE

Developments will have to comply with European directives on the protection of personal data.



As part of the project, the Council of Europe's DPO must be consulted and issue recommendations in the context of the RGDP.

[End of document]