Application view

2024-10-08

**Last modified** : 2024-10-08

**Last global review** : *<perform regular global reviews of the view at least once a year while the project is active and mention the date here>*

**Document status** : *<'WIP', 'DRAFT', …​>*

# Introduction

This is the application point of view of the project. It describes the application modules in play and their exchanges.

The other views of the document are accessible [from here](./README.adoc).

The project glossary is available [here](glossary.adoc). We will not redefine the functional or technical terms used here.

## Reference Documentation

Mention here the reference (defined at a IS level) architecture documents. This file should never summarize their content under penalty of quickly becoming obsolete and unmaintainable.

Documentary references

| N ° | Version | Document title / URL | Detail |
| --- | --- | --- | --- |
| *1* | *2.0.4* | *XX\_Urba\_POS.pdf* | *IS mapping* |

# Not ruled

## Points subject to further study

Points subject to further study

| Subject | Detail | Status | Subject bearer | Deadline |
| --- | --- | --- | --- | --- |
| *Use of Y services* | *Depending on the progress of project Y, this component could call the services of the latter or those of the former component Z* | *PENDING* | *Project Y team* | *BEFORE 2040* |

## Assumptions

Assumptions

| ID | Detail |
| --- | --- |
| *HA1* | *Even if the decision to generalize the centralized directory is not fully endorsed, the application will rely on it and not on a local directory.* |

# General context

## Objectives

Briefly describe the project and recall its objectives. Highlight those which are structuring for the architecture.

Example 1: This application allows suppliers invoices dematerialization and easy consultation of these documents by the accounting services.

Example 2: This project is the rewrite in web technologies of the X legacy application. It should ease the maintenance.

Example 3: The X application is one of the main components of the Y program. It leverages the person and billing repositories to enrich the CMS with real-time customer data.

## Existing

If this document presents a redesign or migration project, describe briefly the existing application. Do not repeat the documentation, simply refer to it and point to its architecture document if available. Beware of mentioning any information with a strong impact on the new project.

Example 1: The GOLD application is a Client-Server application in FORMS 4 pointing to an Oracle 9i database. Its architecture document is given in [REFxyz].

Example 2: The existing application is based on an LDAP directory for its authorizations. The new project has to coexist temporary with the former one. Thus, it is important to manage concurrent accesses as well as the coherence of LDAP during the tiling period.

## Positioning in the IS

If the IS is urbanized, identify the block concerned by the project.

## Actors

### Internal actors

By the term "internal", the IT team project refers to actors belonging to the organization. These actors can be humans or application components.

List of internal actors

| Actor | Description | Population | Location |
| --- | --- | --- | --- |
| *Administration system B* | *Provides company accounting data* | *N/A* | *Berlin site* |
| *Agent* | *Back-office agent* | *100* | *London site* |

### External actors

List of external actors

| Actor | Description | Population | Location |
| --- | --- | --- | --- |
| *Web client* | *A company from a PC* | *Max 1M* | *10 calls to the GUI per session, one session per day and per actor* |
| *Mobile client* | *A company from a mobile* | *Max 2M* | *Worldwide* |

# Constraints

## Budget

Give the budget constraints of the project

Example 1: Overall envelope of $1M

Example 2: Cloud infrastructure should cost less than $20K a month

## Planning

Without detailing the project schedules, it is suggested to highlight interesting elements for the architecture.

Example 1: Application Launch before February 2034, prerequisite for the HEAVY program in May 2034.

## Urbanization

List here the constraints relating to urbanization, this includes for example but not only:

* The rules applicable for calls between components (SOA)
* Call rules between network zones
* The rules concerning the localization of data (MDM)
* The rules concerning the propagation of updates by events (EDA)

Example 1: Calls between two services are prohibited except service calls to a nomenclature service.

Example 2: to ensure freshness, it is forbidden to replicate data from the PERSON repository. The latter must be interrogated synchronously if necessary.

Example 3: When modifying an order, the accounting and invoicing areas will be updated asynchronously via an event.

Example 4: All the batches must be able to operate in competition with the UIs without locking the resources.

Example 5: Services cannot be called directly. The calls must be made via an exposed route at the level of the company bus which will in turn call the service. It is then possible to control, prioritize, orchestrate or manage the calls.

Example 6: The components of this application follow the SOA architecture as defined in the reference document X.

Example 7: Components in an Internet zone cannot call components in an Intranet zone for security reasons.

## Legals

List here (without detailing too much) any legal constraints related to the project.

Example 1: The framework contract established with the ESN XYZ provides for the transfer to our company of the copyright on the source code.

Example 2: The project code will be under the free and open source license GPL V3.

Example 3: The data exposed by the project will be licensed under ODS-By.

Example 4: The EULA of the software package provides access to sources for users with shares in the company.

# Requirements

List here the application architecture requirements that may apply to the project. Depending on your context, feel free to add sub-sections.

## Strategic Requirements

Describe here the requirements related to the overall strategy of the project in terms of trajectory, budget, and organization.

Example 1: Development must be able to take place within distributed teams, each working on distinct modules.

Example 2 (migration project): Legacy modules should require as few adaptations as possible due to a lack of human resources.

## Interoperability

Describe here the requirements regarding protocols, formats, and semantics to be followed to facilitate exchanges with organizations or third parties.

Example: Our XYZ modules must be exposed to X organizations from the Internet in the form of authenticated REST APIs.

## Archiving

Archiving is the copying of important data to a dedicated offline medium for occasional consultation, unlike backup which is intended for restoration. Archives are often required for legal reasons and kept for thirty years or more.

Specify if application data needs to be kept long-term. Specify the reasons for this archiving (usually legal).

Specify if specific integrity protection mechanisms (mainly to prevent any modification) need to be put in place.

Example 1: As required by the law, accounting data must be kept for at least ten years.

Example 2: Accounting documents must be kept online (in the database) for at least two years and then can be archived for at least ten more years. A SHA256 hash will be calculated at the time of archiving and stored separately to verify the integrity of the documents if needed.

## Retention Periods

Specify here how long data and documents persisted by your application modules should be kept. Note that these durations may be legally constrained (see legal constraints above), for example in the context of the GDPR right to be forgotten.

Don’t forget to mention technical data (such as logs or technical tables) as well as archives.

Example:

Retention period for data and documents

| Data | Maximum Retention Period |
| --- | --- |
| *Payment Data (Credit Card)* | *2 months* |
| *Order List* | *2 years* |
| *Access Logs* | *1 month* |
| *Archived Accounting Data* | *30 years* |

# Target architecture

## General application architecture

Present here the application as a whole (without detailing its sub-components) in relation to the other applications of the IS. Also present the macro-data exchanged or stored.

Summarize:

* The kind of architecture (client-server, monolithic Web, SOA, micro-service…​).
* Large network flows between components or between applications in the case of monoliths.
* Any derogation to applicable architectural rules.

If the application is planned to be implemented in several stages, briefly describe the target trajectory.

The choice of representation is free but a C4 diagram from System Landscape or a UML2 component diagram seems the most suitable.

Numbering the steps in chronological order ensures a better understanding of the diagram. Group the sub-steps by the notation x, x.y, x.y.z, …​

Do not include specific infrastructure system (SMTP server, security device, reverse proxy, LDAP directories, etc.) which are in the domain of technical architecture. On the contrary, mention Enterprise Service Buses, API Gateway or similar components if they play an application role (service orchestration for example).

Example 1: AllMyData allows a company to retrieve by email a document summarizing all the information the administration has on it. The administration can supplement its data with those of another administration.

Example 2: AllMyData is made up of several independent microservices (GUI components, batches or REST services).

Example 3: Thanks to the August 03, 20xx derogation, the GUI will be written using an SPA (Single Page Application) technology.

## Detailed application architecture

Detail here all the components of the application, the flows between them and with the other applications of the IS.

Provide one or more diagrams (preferably C4 diagrams of the container type or UML2 component diagram).

Ideally, the diagram will fit on an A4 page and be self-supporting and understandable by non-ICT staff. It should become one of the most important documentary artifacts and be in the war room or be printed by every developer.

If the application is particularly complex, draw a diagram for each linking chain.

Use a simple non-significant sequence (1, 2, …​, n) as the flow ID. The flows are logical and not technical (for example, we can represent a direct HTTP flow between two components when in reality, it passes through an intermediate load balancer: this level of detail will be given in the infrastructure section).

For each stream, give the protocol, a synchronous/asynchronous attribute, a read/write/execute attribute and a description so that the scheme is self-supporting.

### Principles that dictated the choices

Give here the intention in the architecture conception.

Example: we will use a monolithic and non-micro-service approach due to a lack of expertise within the IT project team.

### Static view

Expose the application modules in their different zones or domains.

Example: module X, Y and Z in the ACCOUNTING domain. Modules A, B in the PERSON domain.

### Dynamic view

Expose the application modules in their different areas or domains with their main application flows.

Do not detail technical flows (such as flows related to monitoring or clustering).

If the application is complex, propose a global diagram listing all the application flows and then a diagram for each main linking chain by numbering the exchanges (use a sequence diagram or (better) a C4 Dynamic Diagram). It is also possible to detail the linking chains by main functionality.

Example:

## Archiving

Describe here the measures to meet archiving requirements. This section will mainly include:

* Technology: Ideally, for security, the archive will be duplicated on multiple media of different technologies: magnetic tape type LTO, optical disk (Blu-ray Disc Recordable for example), cloud storage (such as AWS 'Glacier' or GCP 'Coldline'), SMR mode hard drives, etc.
* A specific storage location distinct from traditional backups (e.g., Cloud, bank vault).

Example: Bank statements older than 10 years will be archived on LTO tape and hard drive. A set of each medium will be stored in a vault in two different banks.

## Purges

Describe here the technical measures to meet purge requirements.

Example 1: The consultation history will be archived by a dump with an SQL query like COPY (SELECT \* FROM my\_table WHERE …​) TO '/tmp/dump.tsv' and then purged by an SQL DELETE query after the operator has validated the completeness of the dump.

Example 2: Each API is responsible for purging the data it exposes. For this, plan internal processes that delete data according to a schedule (cron expression) and configurable criteria.

## Matrix of application flows

List here the main network flows of the application.

Do not detail the monitoring or clustering streams for example. Indicate the type of network (LAN, WAN).

Partial example of an application flow matrix

| Source | Destination | Network type | Protocol | Mode.[[1]](#footnote-60) |
| --- | --- | --- | --- | --- |
| *Company* | *PC / tablet / external mobile* | *WAN* | *gui-allmydata* | *R* |
| *batch-process-requests* | *service-compo-pdf* | *LAN* | *HTTP* | *C* |

1. Read(R), Write (W) or Call(C) to a stateless system [↑](#footnote-ref-60)